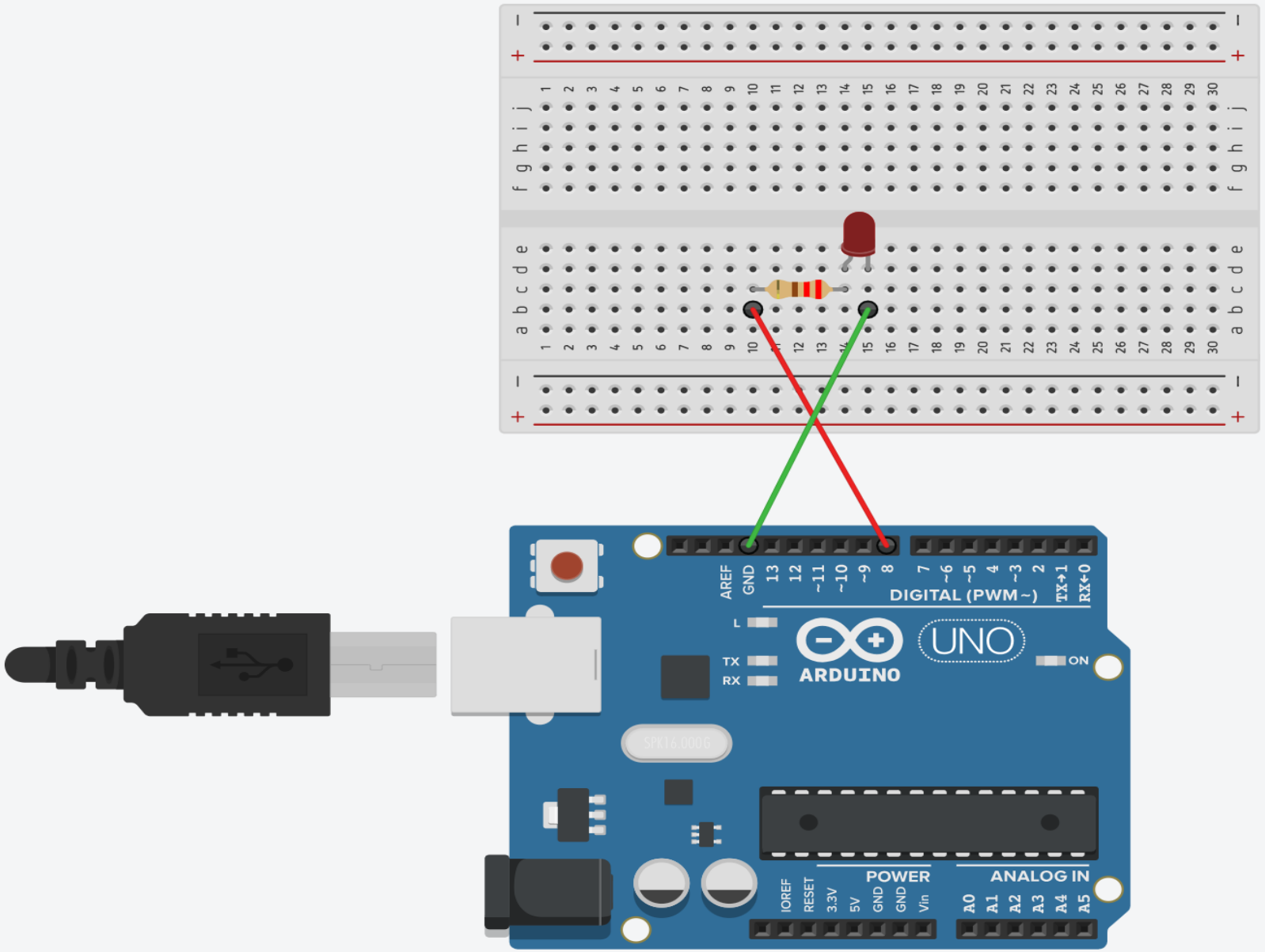
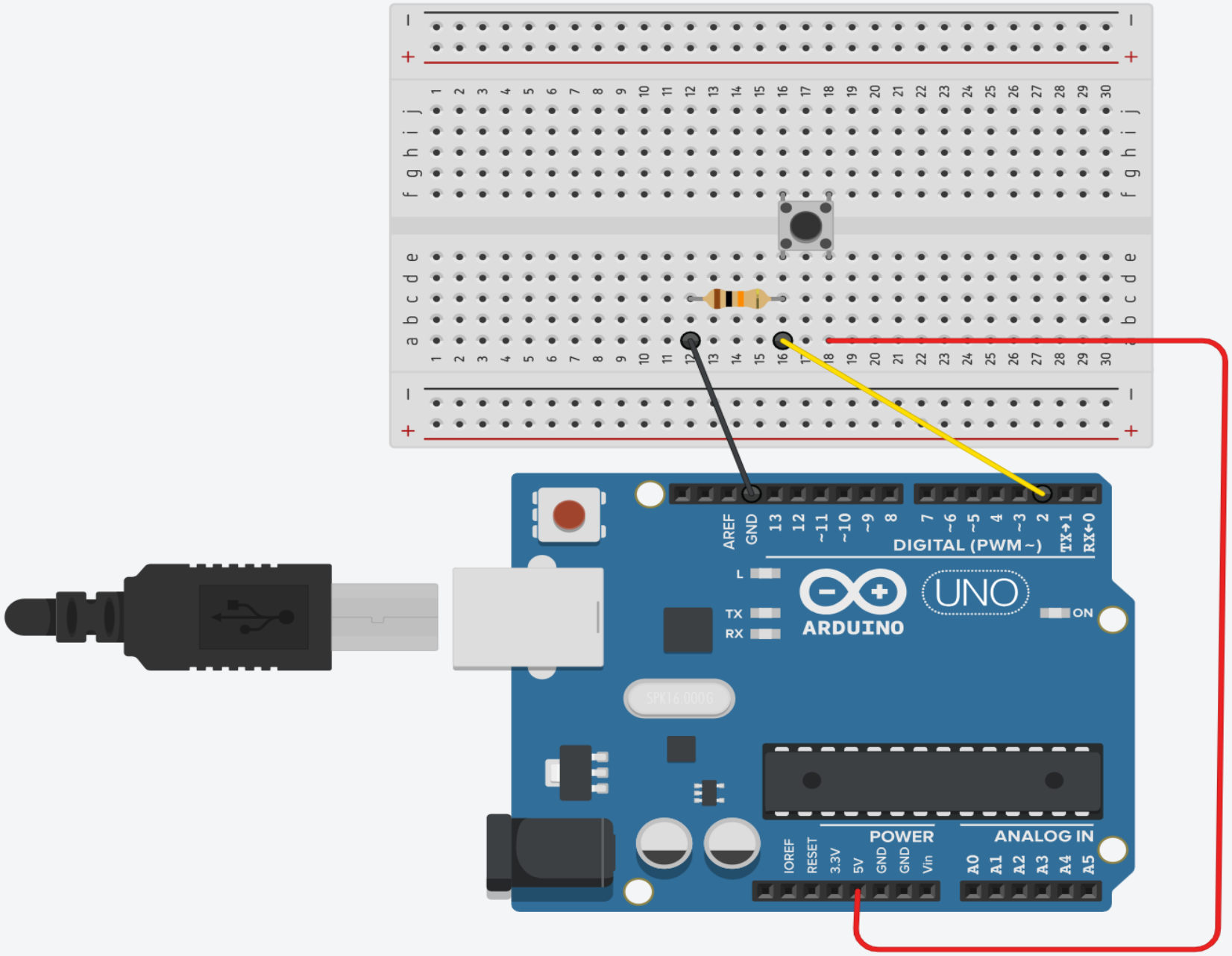
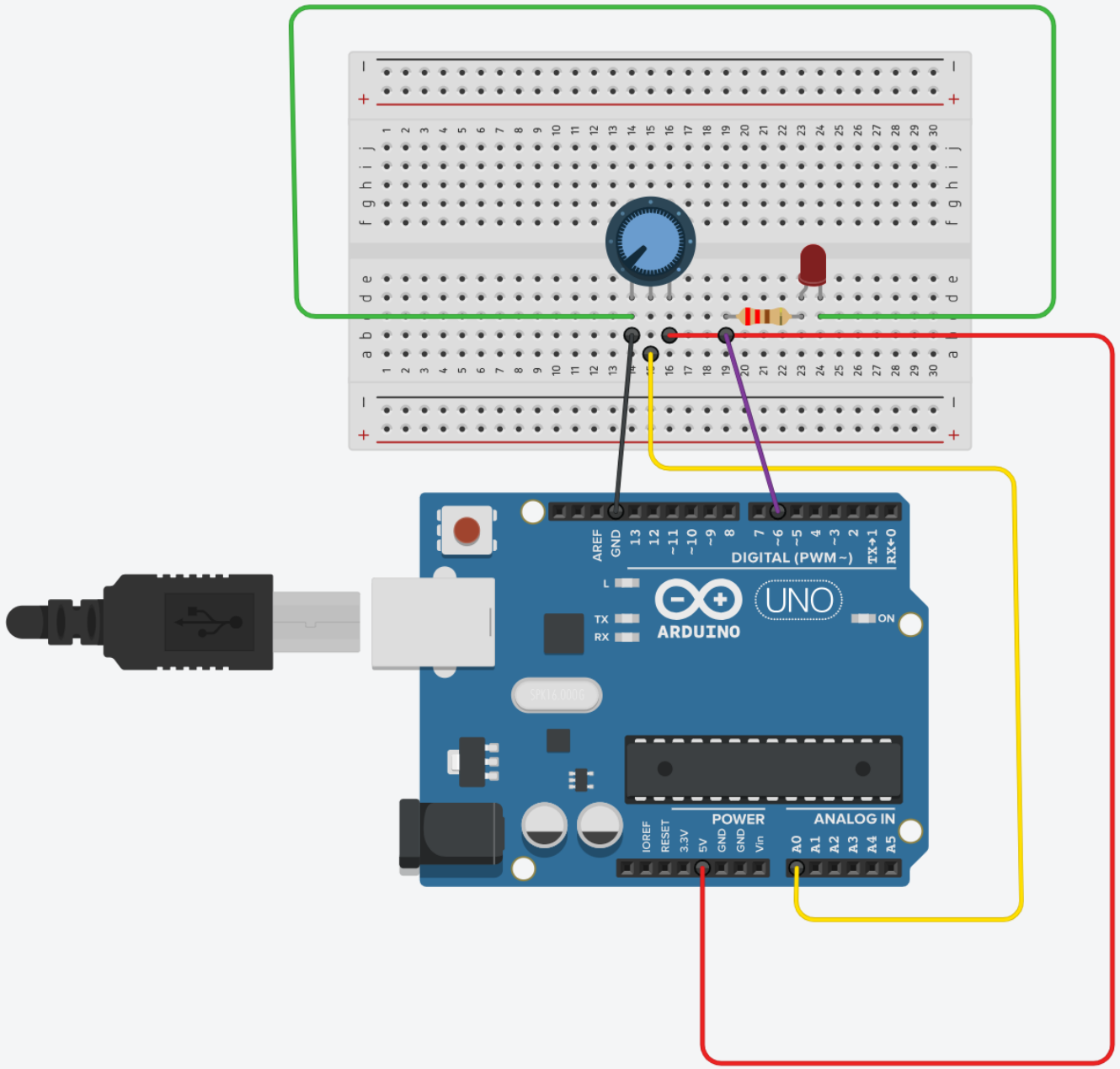
ARDUINO PROJECTS



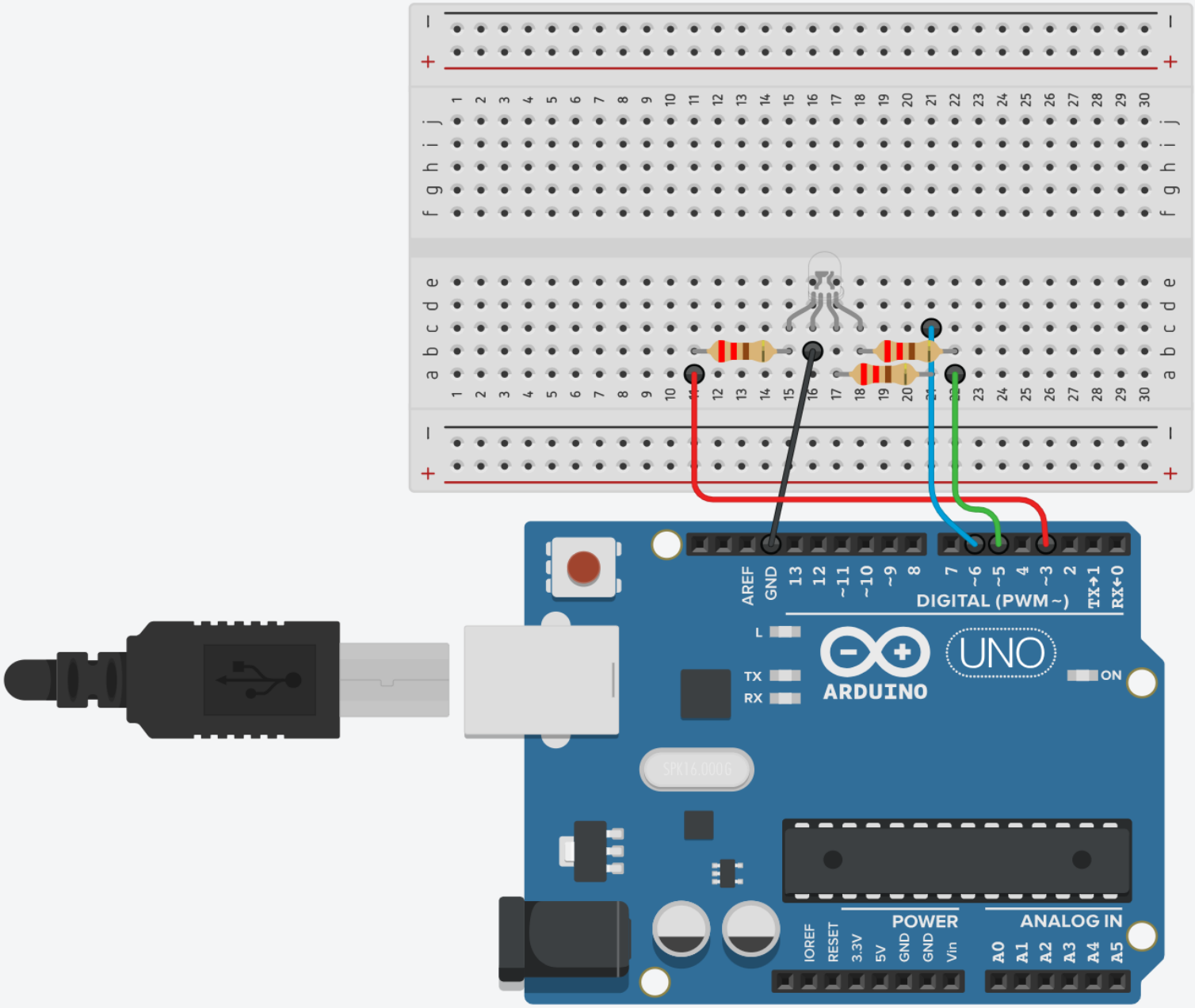

```
// Project 1: Blinking of light
//
void setup()
{
  pinMode(8, OUTPUT);
}

void loop()
{
  digitalWrite(8, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(8, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```



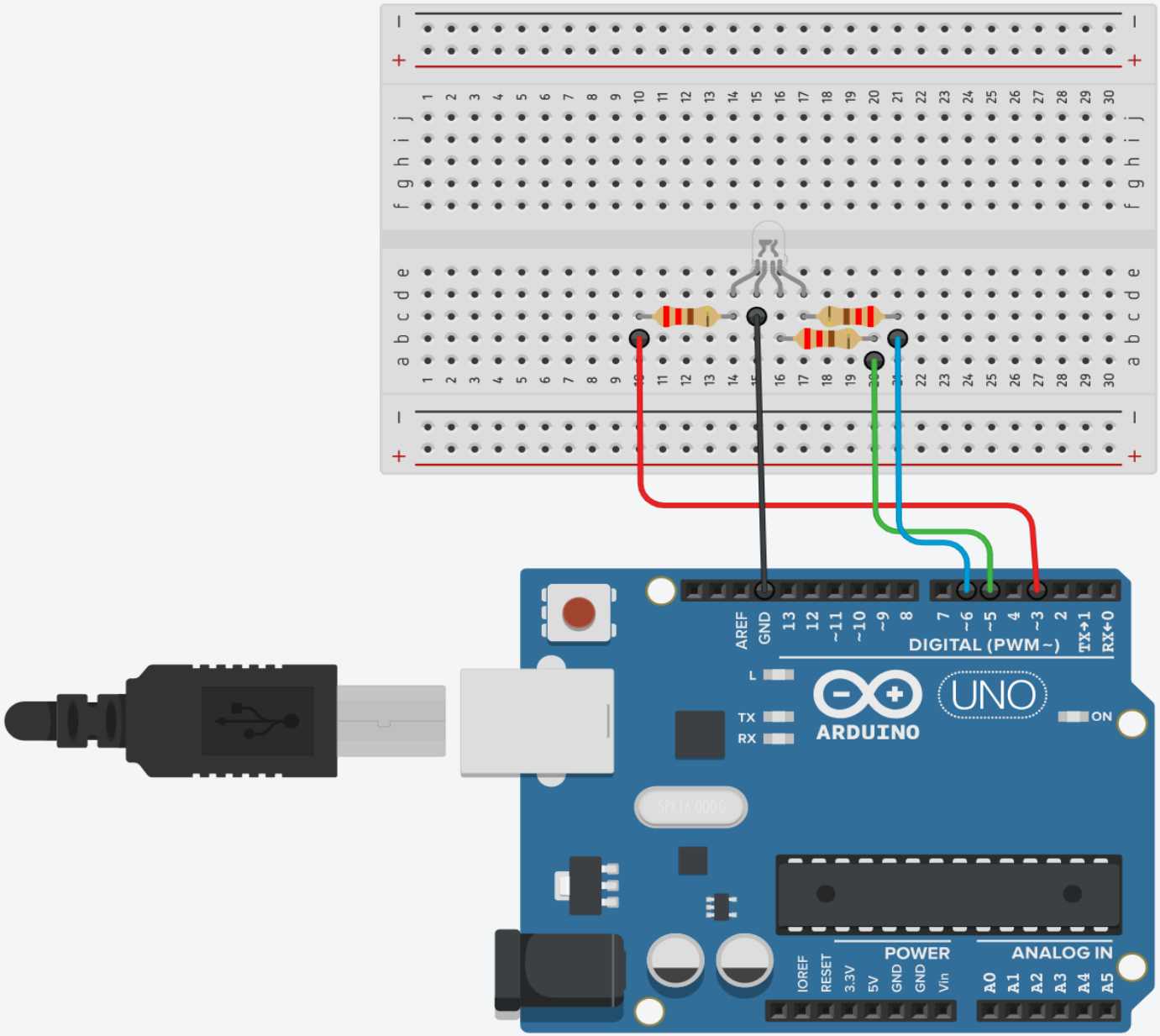

```ino
// Project 2: Working of Button

int buttonState=0;
void setup()
{
  pinMode(2,INPUT);
  Serial.begin(9600);
}

void loop()
{
  buttonState= digitalRead(2);

  if(buttonState == HIGH)
  {
    Serial.println("HIGH");
}else
  {
    Serial.println("LOW");
  }
}

```




```ino
// Project 3: Working of Potentiometer

void setup()
{
  pinMode(6, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  int potvalue= analogRead(A0);
  Serial.println(analogRead(A0));
  int mappedValue= map(potvalue, 0, 1023, 0, 255);
  analogWrite(6,mappedValue);
}  //analogRead()  possible 0...1023
   //analogWrite() possible 0...255

   /* map function is used to convert the range of one function to
   another.*/
   /*connect pin 7 instead of 6 and compare potvalue and
   mappedValue by putting in analogWrite bracket.*/
```

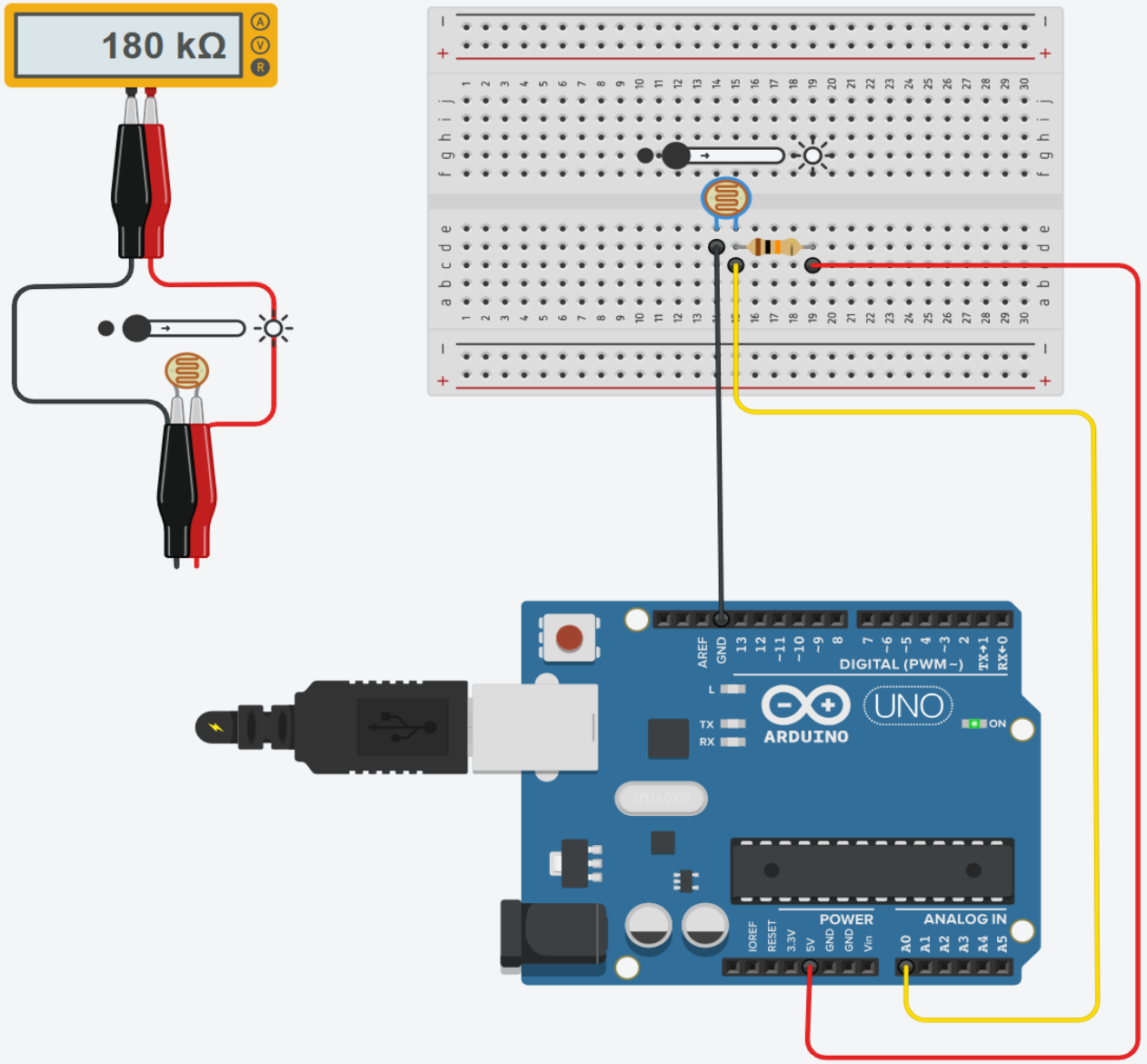



```ino
// Project 4: Working of RCGB light

//const intb red_pin=3;
#define RED_PIN   3
#define BLUE_PIN  6
#define GREEN_PIN 5

void setup()
{
  pinMode(RED_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
}

void loop()
{
  digitalWrite(GREEN_PIN, LOW);//turn the LED OFF
  digitalWrite(RED_PIN, HIGH); //turn the LED ON
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(RED_PIN, LOW); //turn the LED OFF
  digitalWrite(BLUE_PIN, HIGH); //turn the LED ON
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(BLUE_PIN, LOW); //turn the LED OFF
  digitalWrite(GREEN_PIN, HIGH); //turn the LED ON
  delay(1000); // Wait for 1000 millisecond(s)

}
```

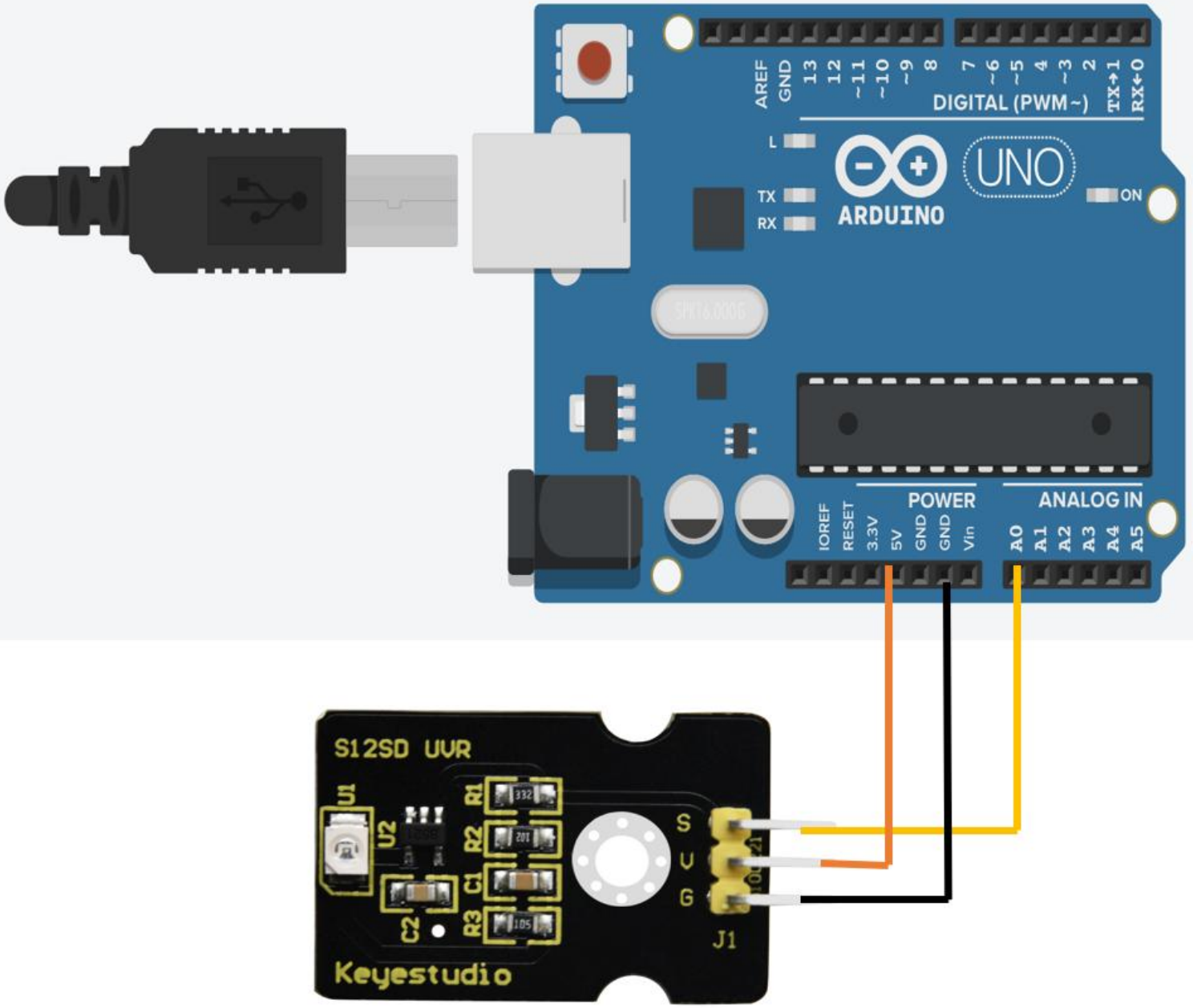



```ino
// Project 5: More on RGB Led by including RGB library

#include <AlaLedRgb.h>
#define red 3
#define blue 6
#define green 5
void setup()
{
  rgbLed.initPWM(red,blue,green)
    rgbLed.setBrightness(0x66FF44);
  rgbLed.setAnimation(ALA_FEDCOLORSLOOP,5000,alaPalRgb);
}

void loop()
{
  rgbLed.runAnimation();
}
```

Name brightness level



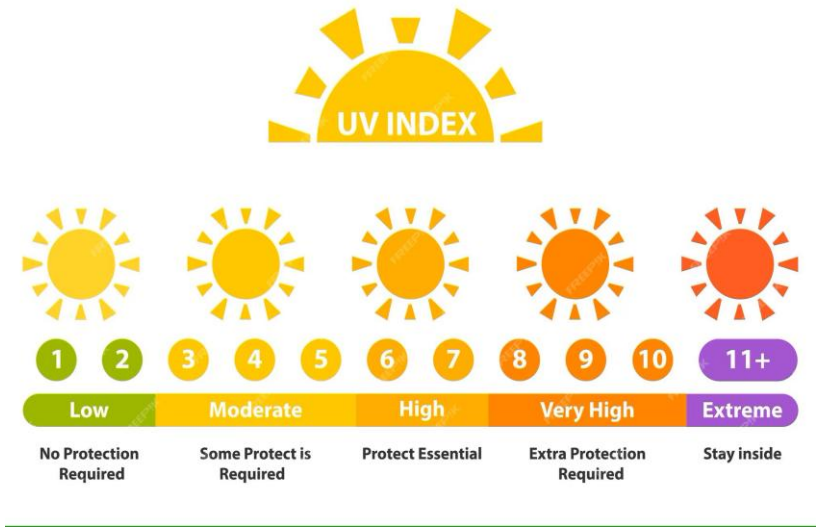

```
// Project 6: Working of Photoresistor

void setup(){
Serial.begin(9600);
}

void loop(){
int sensorValue = analogRead(A0); // ranges 0...1023
  Serial.println(sensorValue);
  delay(10);
}

//The circuit is known as "voltage divider".

/* Resistance of photoresistor is inversely proportional to
   intensity of light , brighter the light lesser the
   resistance. */

/* Photoresistor (resistance range)
Min   0.300k-ohm   Bright light
Max   70k-ohm      less light (black)
Med   1.3k-ohm     normal */

/* Serial monitor shows the value between the range 0....1023
   based on vout, vout= vin/(R1+R2)*R2
   where R1=photo-resistance, R2=resistance of resistor
   Vin=5V */

/* here, the range of photo-resistor is from 506ohm to 180k-ohm
   you can check on the multimeter by varying the brightness
   level. */

```

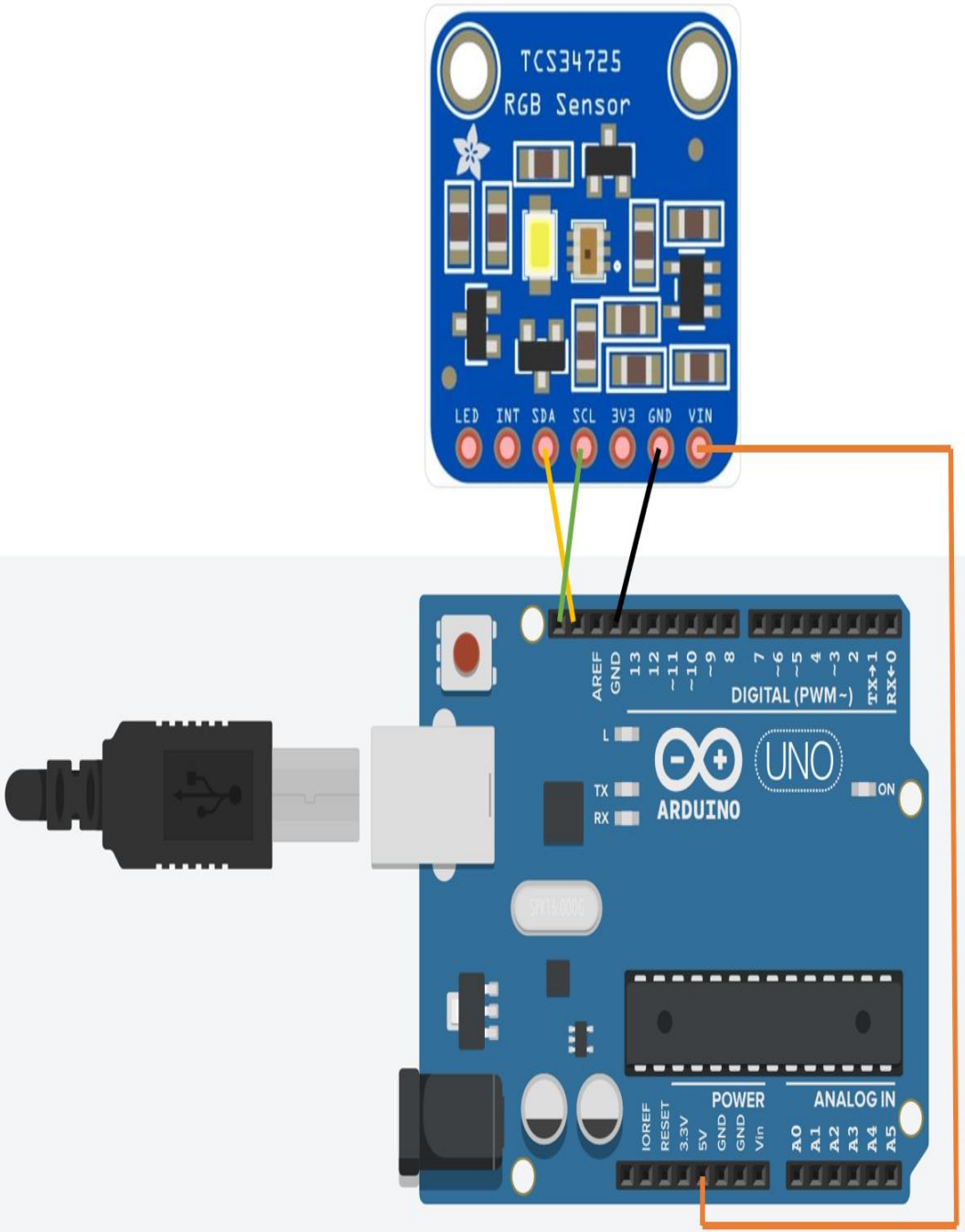



```ino
// Project 7: Working of UV light sensor(GUVA-S12SD)

void setup()
{
 Serial.begin(9600);
}

void loop()
{
  // read the input on analog pin 0;
  int sensorValue= analogRead(A0);

  // calculate the actual voltage at sensor out
  float voltage= sensorValue*(5.0/1023.0);

  // print the value of analog input
  Serial.print(sensorValue);
  Serial.print(",");

  // print the UV index value. As per the specifications for the sensor,
  // this is done by dividing the sensor output voltage by 0.1
  Serial.println(voltage/0.1);
  delay(100);    // delay in between reads for stability
}
```

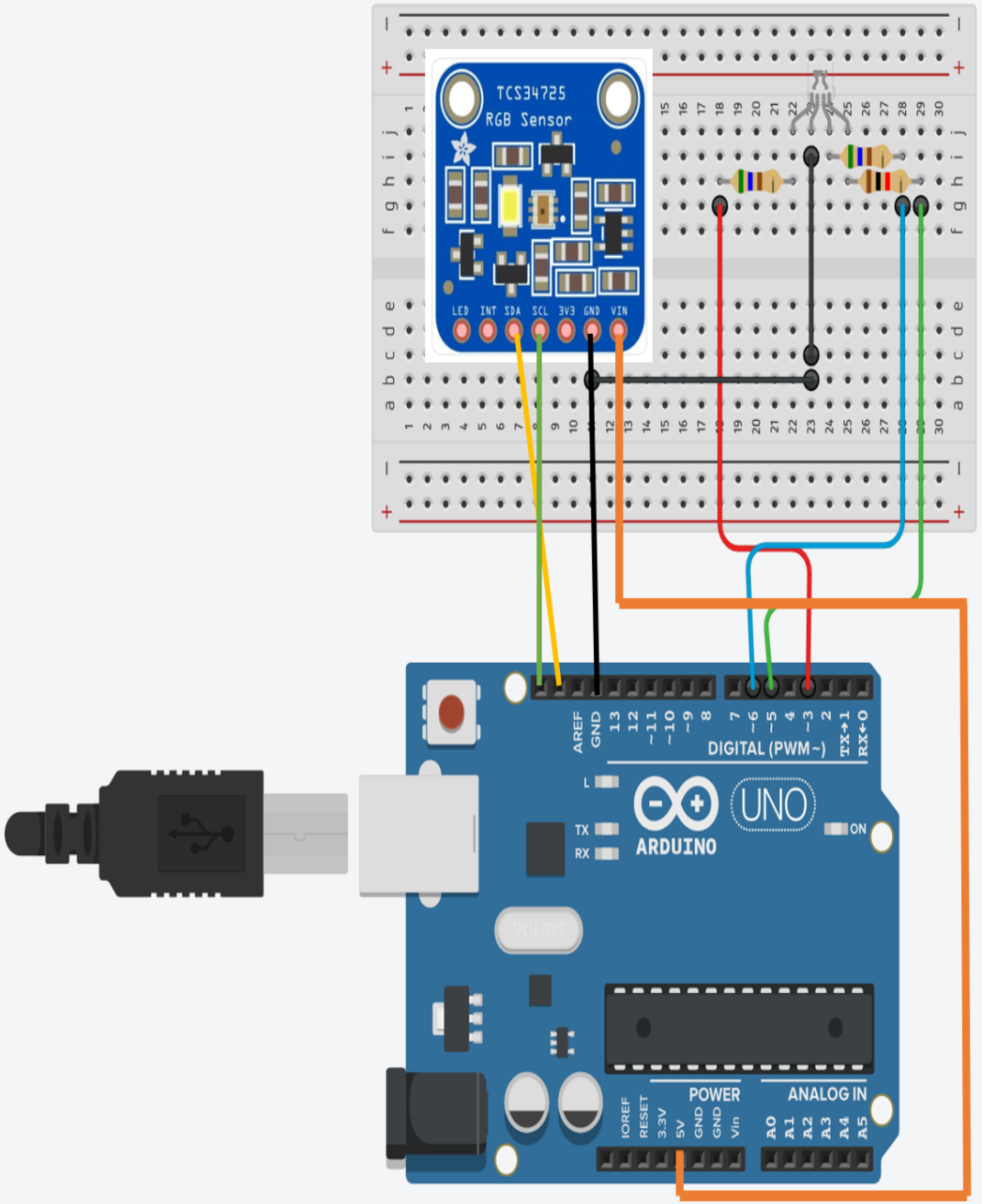
Serial Monitor

```
8,0.39
699,34.16
766,37.44
228,11.14
655,32.01
235,11.49
767,37.49
342,16.72
729,35.63
352,17.20
740,36.17
145,7.09
267,13.05
599,29.28
110,5.38
269,13.15
455,22.24
89,4.35
522,25.51
```



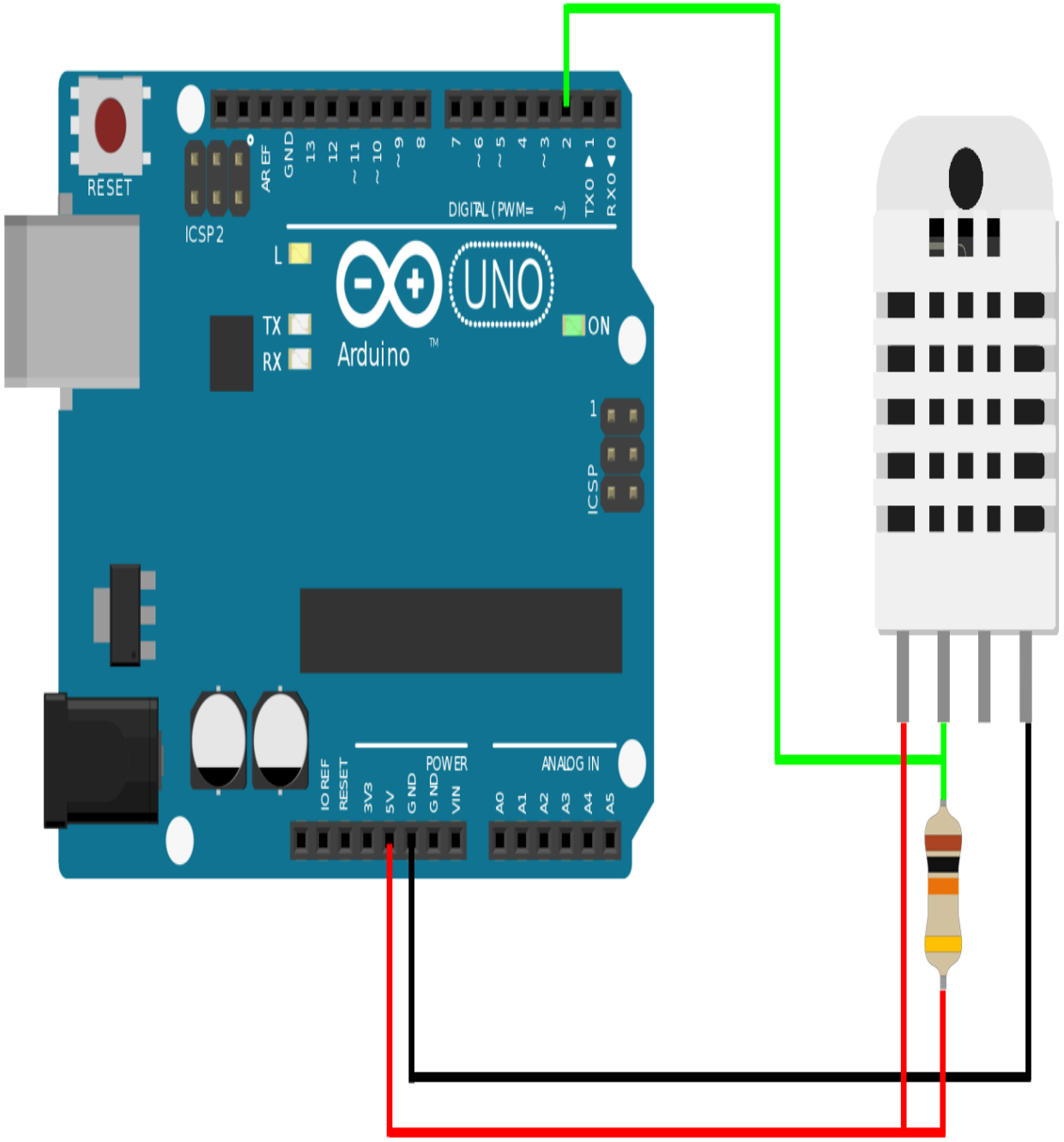



```ino
// Project 8: Working of TCS34725 RGB Light Sensor(Detecting color)

#include<Wire.h>
#include "Adafruit_TCS34725.h"

Adafruit-TCS34725 tcs= Adaafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4x);
/* TCS34725_INTEGRATIONTIME_50MS: The time the sensor takes to collect light data (50 milliseconds).
TCS34725_GAIN_4X: The sensor's sensitivity or gain is set to 4x. (Higher gain is useful for low-light conditions.) */


void setup()
{
  Serial.begin(9600);
  Serial.println("Color View Test!");

  if(tcs.begin()){
    Serial.println("Found sensor");
  }else{
  Serial.println("No TCS34725 found ... check your connections");
    while(1); /* tcs.begin(): Initializes the sensor. If not found, the program stops by entering an infinite
    loop (while(1);) */

  }

}

void loop()
{
  uint16_t clear, red, green, blue; // clear, red, green, blue are 16-bit unsigned integers to store light intensity values.

  tcs.setInterrupt(false);   // Activates the sensor's onboard LED for illumination. (turn on LED)

  delay(60); // takes 50ms to read

  tcs.getRowData(&red, &green, &blue, &clear);/* Reads the raw color data:
                                                 red, green, blue: Light intensity values for red, green, and blue wavelengths.
                                                 clear: Overall light intensity (combined light across all wavelengths). */

  tcs.setInterrupt(true); // Deactivates the LED after data collection. (turn off LED)

  Serial.print("C:\t"); Serial.print(clear);
  Serial.print("\tR:\t"); Serial.print(red);
  Serial.print("\tG:\t"); Serial.print(green);
  Serial.print("\tB:\t"); Serial.println(blue);
}

/* Output:
C: [clear_value]    R: [red_value]    G: [green_value]    B: [blue_value] */
```

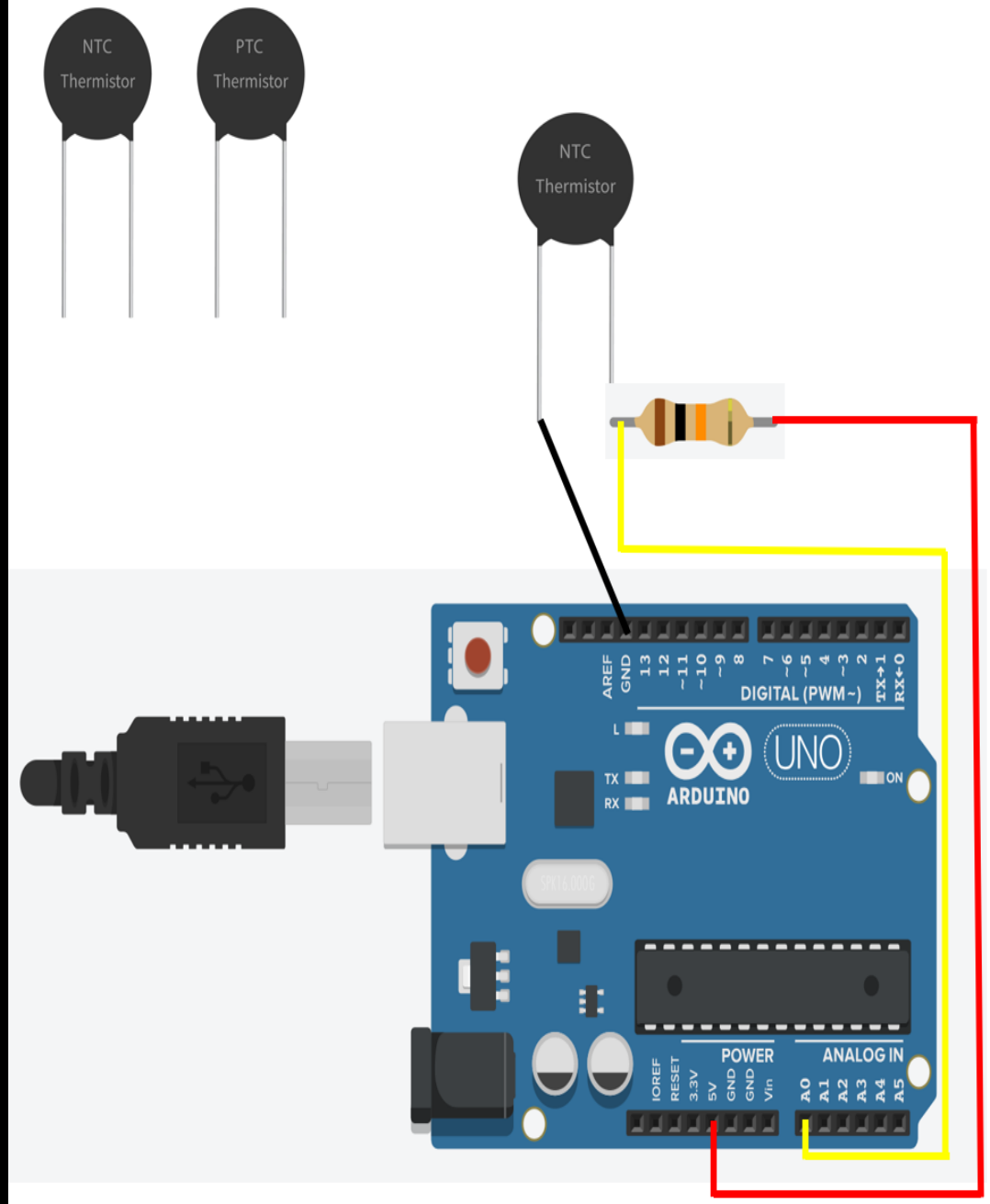



```
// Project 9: Copying colors from TCS34725 RGB Light Sensor to RGB light

#include<Wire.h>
#include "Adafruit_TCS34725.h"

//pick analog outputs, for the UNO these three work well
// use 560 ohm resistor between Red & blue, 1k resistor for green (its brighter)
#define redpin 3
#define greenpin 5
#define blue pin 6
#define commonAnode true

// our RGB -> eye-recognised gamma color
byte gammatable[256];

Adafruit-TCS34725 tcs= Adaafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4x);
/* TCS34725_INTEGRATIONTIME_50MS: The time the sensor takes to collect light data (50 milliseconds).
TCS34725_GAIN_4X: The sensor's sensitivity or gain is set to 4x. (Higher gain is useful for low-light conditions.) */


void setup()
{
  Serial.begin(9600);
  Serial.println("Color View Test!");

  if(tcs.begin()){
    Serial.println("Found sensor");
  }else{
  Serial.println("No TCS34725 found ... check your connections");
    while(1); /* tcs.begin(): Initializes the sensor. If not found, the program stops by entering an infinite
    loop (while(1);) */

  pinMode(redpin, OUTPUT);
  pinMode(greenpin, OUTPUT);
  pinMode(bluepin, OUTPUT);

}
```

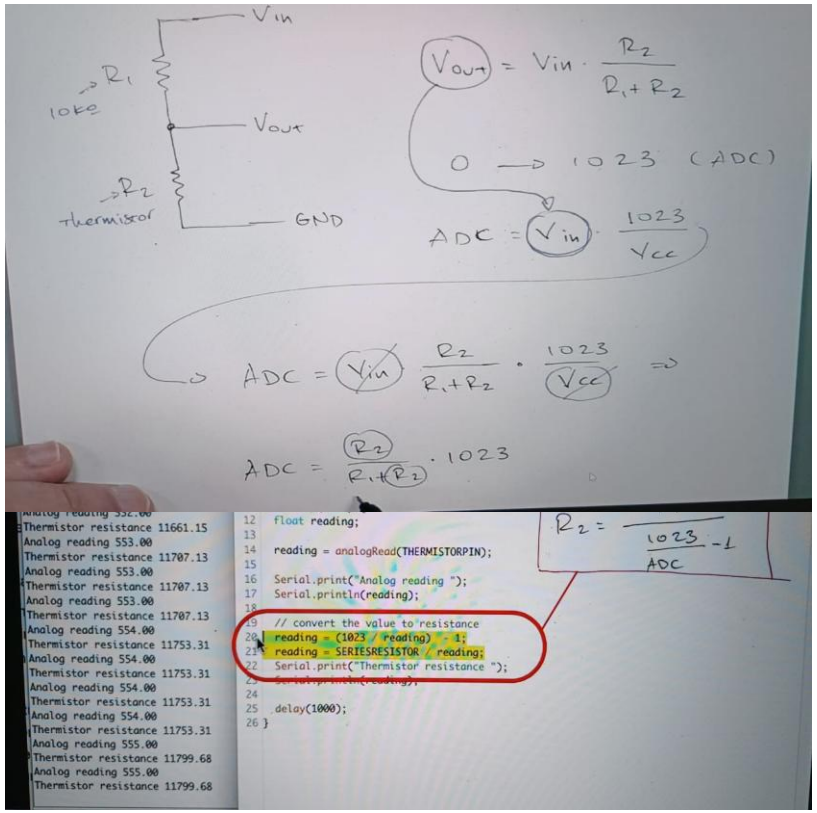
```
for(int i=0; i<256; i++){
  float x=i;
  x/=255;
  x= pow(x,2.5);
  x*=255;

  if(commonAnode){
    gammatable[i]=255-x;
}else{
      gammatable[i]= x;
}

}
}

void loop()
{
  uint16_t clear, red, green, blue; // clear, red, green, blue are 16-bit unsigned integers to store light intensity values.

  tcs.setInterrupt(false);   // Activates the sensor's onboard LED for illumination. (turn on LED)

  delay(60); // takes 50ms to read

  tcs.getRowData(&red, &green, &blue, &clear);/* Reads the raw color data:
                                                 red, green, blue: Light intensity values for red, green, and blue wavelengths.
                                                 clear: Overall light intensity (combined light across all wavelengths). */

  tcs.setInterrupt(true); // Deactivates the LED after data collection. (turn off LED)

  Serial.print("C:\t"); Serial.print(clear);
  Serial.print("\tR:\t"); Serial.print(red);
  Serial.print("\tG:\t"); Serial.print(green);
  Serial.print("\tB:\t"); Serial.println(blue);

  // Figure out some basic hex code for visualization
  uint32_t sum=clear;
  float r,g,b;
  r=red; r/=sum;
  g=green; g/=sum;
  b=blue; b/=sum;
  r*=256; g*=256; b*=256;
  Serial.print("\t");
  Serial.print((int)r,HEX); Serial.print((int)g,HEX); Serial.print((int)b,HEX);
  Serial.print();

  analogWrite(redpin,gammatable[(int)r]);
  analogWrite(greenpin,gammatable[(int)g]);
  analogWrite(bluepin,gammatable[(int)b]);
}
```

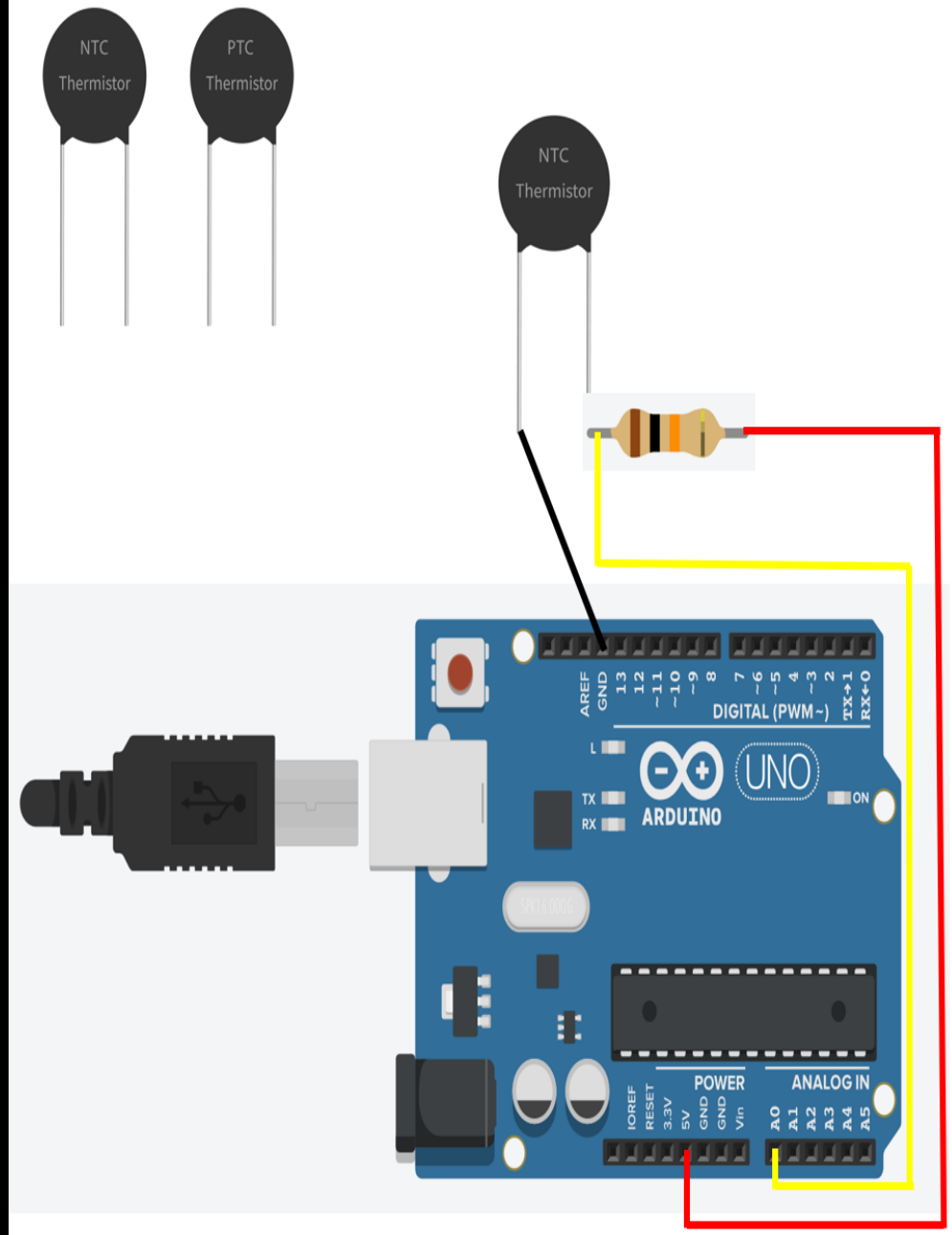



```
// Project 10: Working of DHT22 sensor to measure temperature and humidity

/* A "Strong" pull up resistor has a small value (10kohm). It
allows more current to flow through it compared to a "weak" pull up resistor (50kohm).*/

#include<DHT.h>
#define DHTPIN 2
// uncomment whatever type you are using!
//#define DHTTYPE DHT11 // DHT 11
  #define DHTTYPE DHT21   // DHT 21 (AM2302),AM2321
//#define DHTTYPE DHT11 //DHT 21 (AM2301)
DHT dht(DHTPIN,DHTTYPE);

void setup()
{
 Serial.begin(9600);
  Serial.println("DHTxx test!");

  dht.begin();
}

void loop()
{
  delay(2000);
  // reading temperature or humidity takes about 250 milliseconds!
  //sensor readings may also we up to 2 seconds 'old' (its a very slow sensor)
  float dht.readHumidity();
  // Read temperature as celcius (the default)
  float t=dht.readTemperature();
  // Read temperature as Fahrenhite (isFahrenhite = true)
  float f=dht.readTemperature(true);

  //Check if any reads failed and exist early (to try again).
  if(isnan(h) || isnan(t) || isnan(f)){  // isnan stands for "is not a number"
  Serial.println("Failed to read from DHT sensor!");
    return;
  }

  //compute heat index in fahrenhite (the default)
  float hif=dht.computeHeatIndex(f,h);
  // compute heat index in celcius (isFahrenheit= false);
   float hic=dht.computeHeatIndex(t,h,false);

  Serial.print("Humidity: ");
 Serial.print(h);
  Serial.print("%\t");
  Serial.print("Temperature");
  Serial.print(t);
  Serial.print(" *C ");
}
```

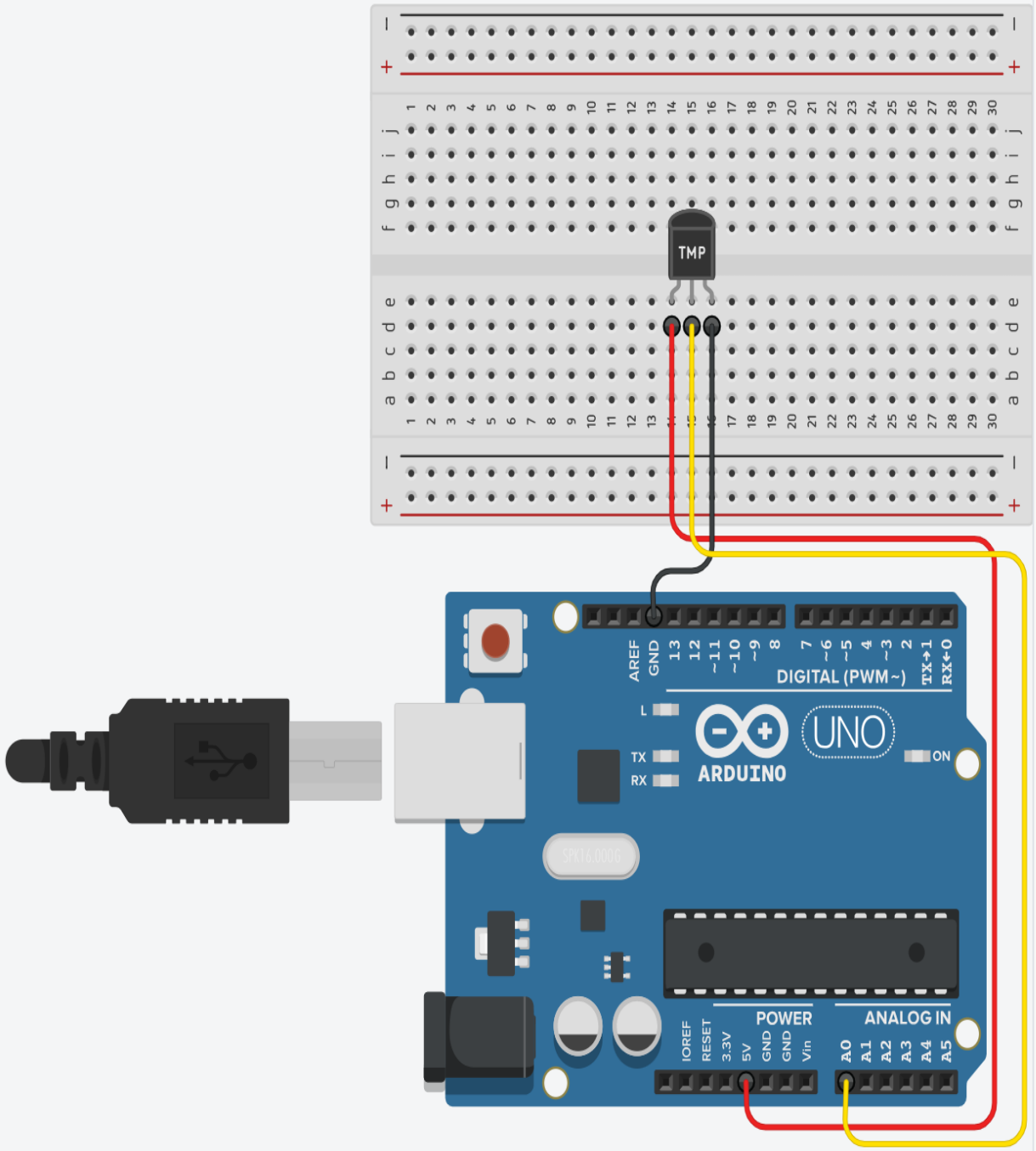



```
// Project 11: Working of Thermistor
//Two types of thermistor: PTC(positive coefficient thermistor,) and NTC(Negative coefficient thermistor).
//PTC:resistance increases with increase in temperature.
//NTC:resistance decreases with increase in temperature.
#define SeriesResistor 10000
#define ThermistorPin A0

void setup()
{
 Serial.begin(9600);
}

void loop()
{
  float reading;

  reading=analogRead(ThermistorPin);

  Serial.println("Analog Reading ");
  Serial.println(reading);

  reading= (1023/reading)-1;
  reading= SeriesResistor/reading; // convert analog value into thermistor resistance.

  Serial.print("Thermistor resistance ");
  Serial.println(reading);

  delay(1000);
}
```

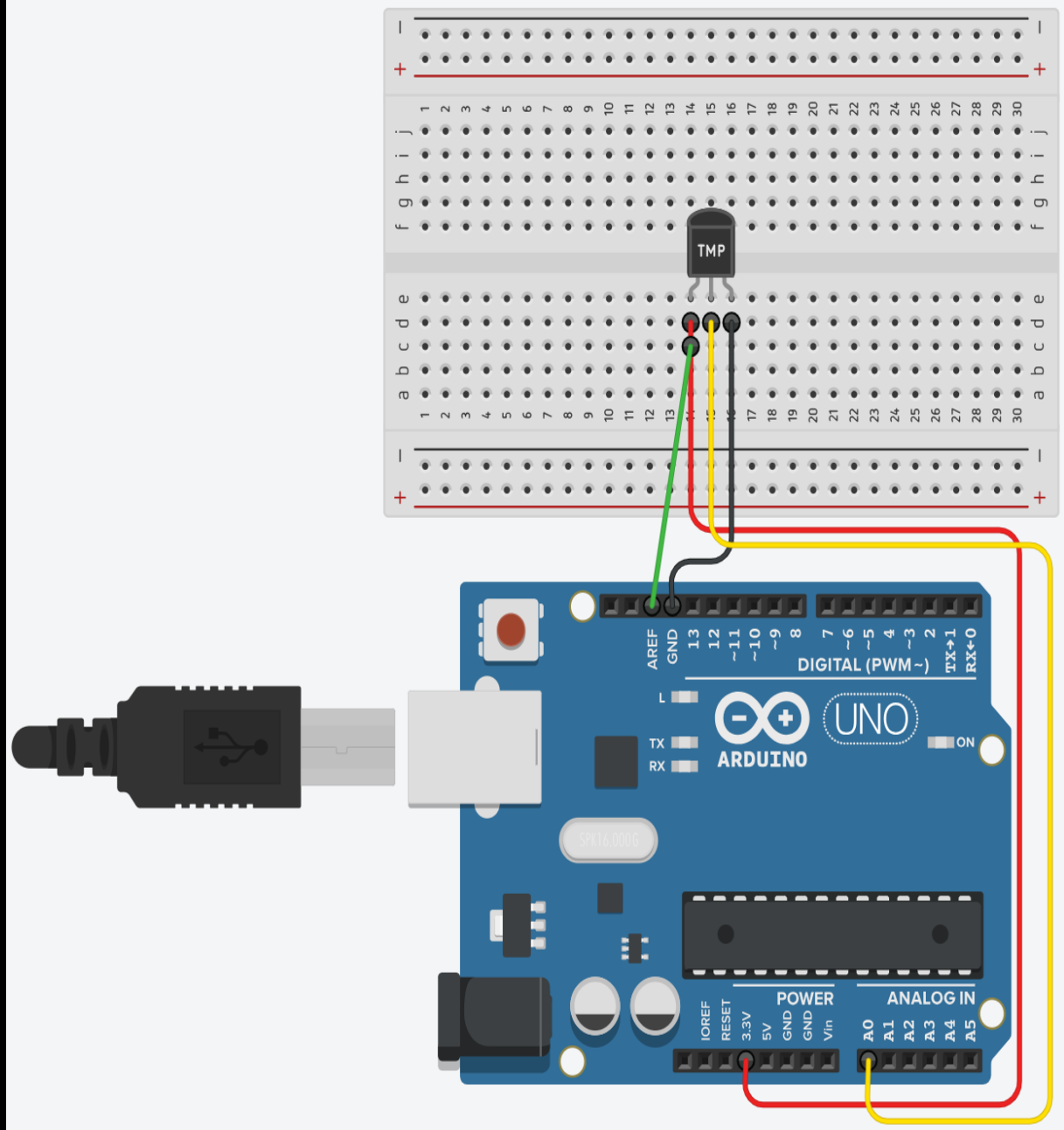




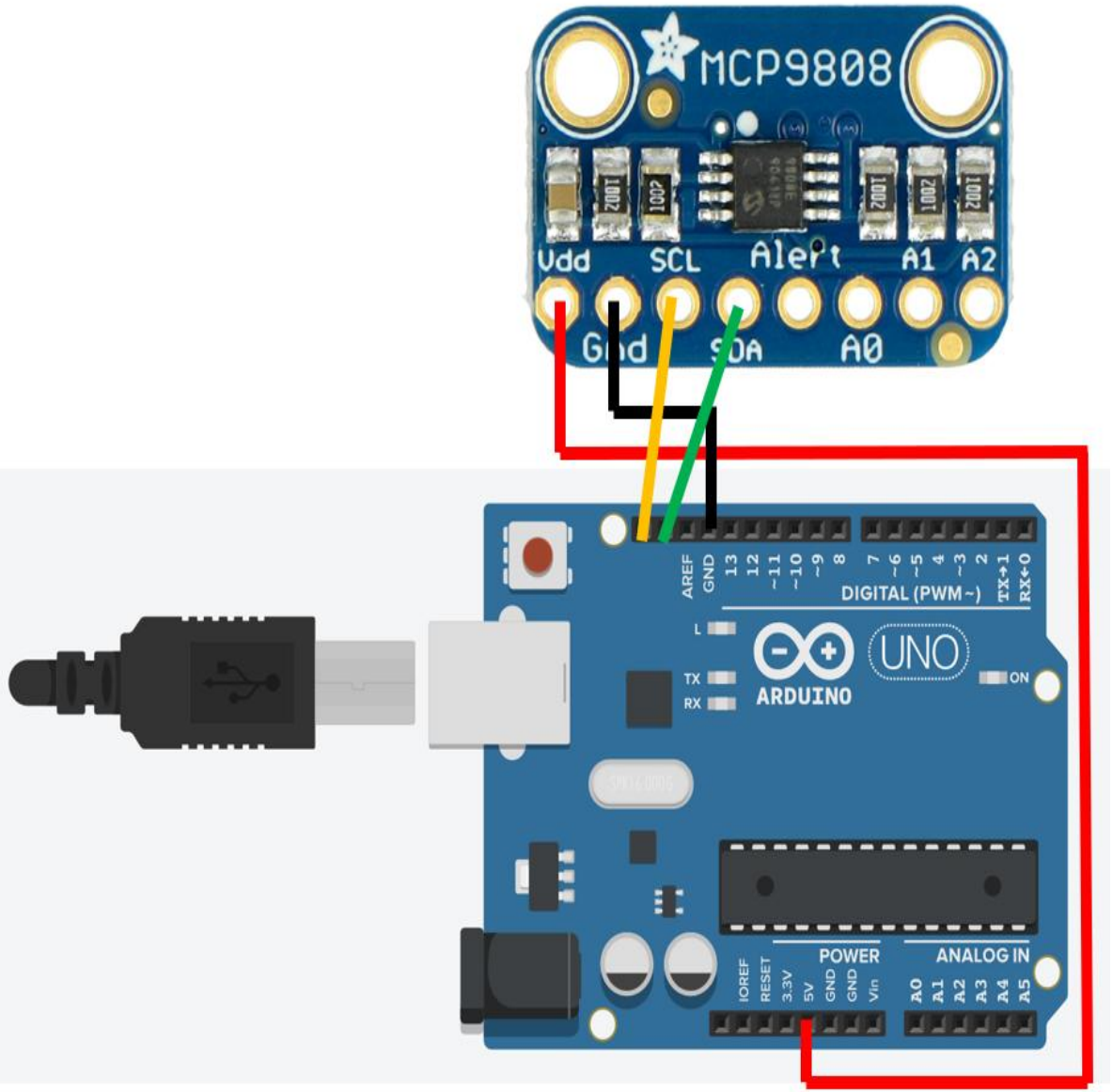

```
// Working of Thermistor (part-2)

#include<Thermistor.h> // library allows to calculations relie on steinhart-Hart equation.
#define NTC_pin A0

// Thermistor object
THERMISTOR thermistor(NTC_pin,  // Analog pin
                      7500,     // Nominal resistance at 25 degree
                      3950,     // thermistor's beta coefficient
                      1000);    // Value of the series resistor

// Global temperature reading
uint16_t temp;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  temp = thermistor.read();

  Serial.print("Temp in 1/10 degree C: "); // Read temperature
  Serial.println(temp);

  delay(5000);
}
```



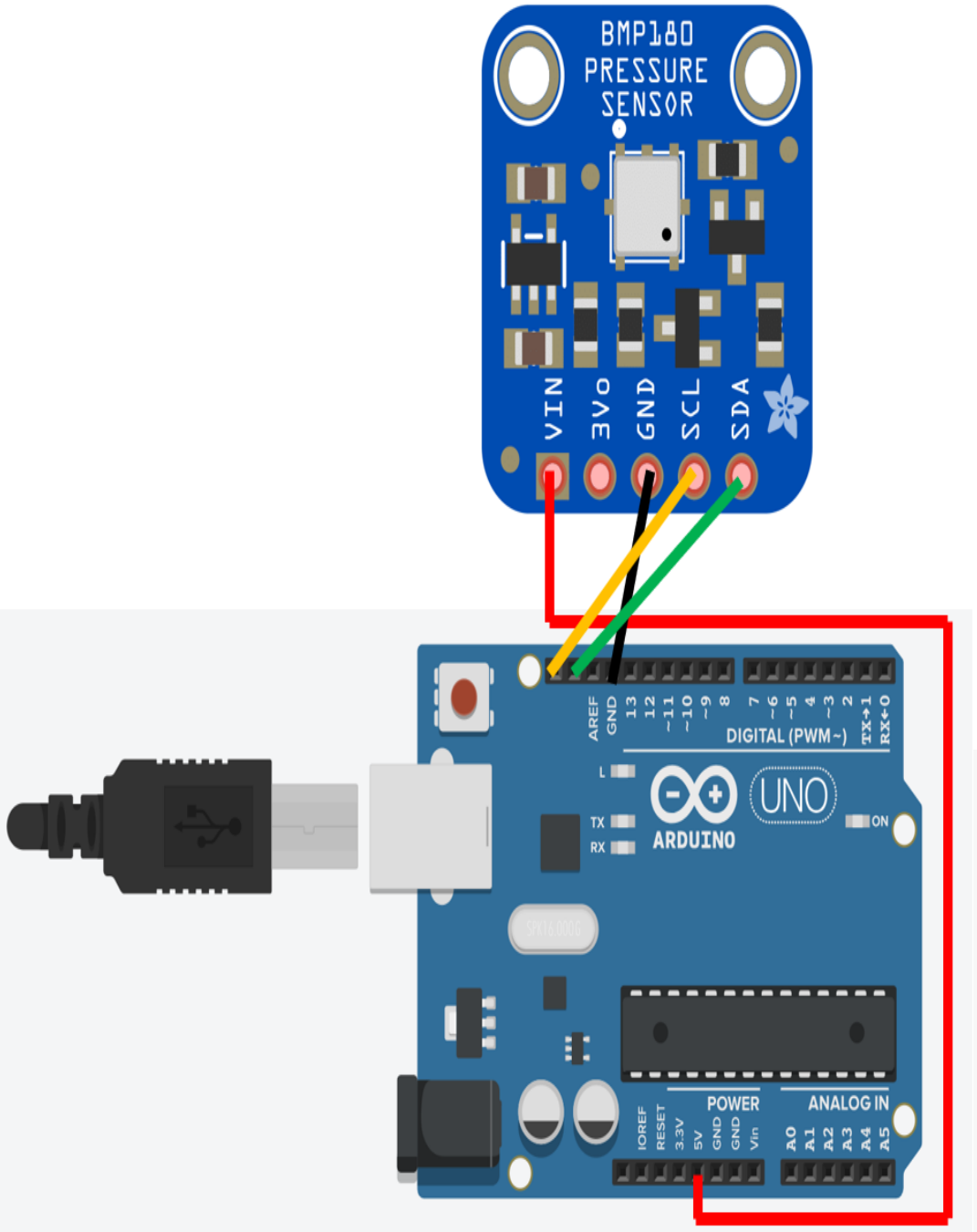

```ino
// Project 12: Working of TMP36 temperature sensor

int sensorPin=0; // the reading is obtained from analog pin 0 (A0)
float supply_voltage=5; //if you are using a 3.3V change it accordingly.

void setup()
{
 //analogReference(EXTERNAL); // if using a 3.3V as reference by bridging it to the AREF pin,
                              //then uncomment this line. If using 5V then it is not necessary.
  Serial.begin(9600); // start the serial conversion with the computer
                      // to view the result open the serial monitor;
}

void loop()
{
 // getting the voltage reading from the temperature sensor
  int reading=analogRead(sensorPin);

  // converting thet reading to voltage, for 3.3V arduino use 3.3
  float voltage= reading * supply_voltage/1024;

  float temperatureC=(voltage-0.5)*100; // converting from 10 mv per degree with 500 mv offset
                                        // to degrees ((voltage-500mv) times 100)
  Serial.print(temperatureC);
  Serial.println(" degrees C");

  // now convert to fahrernheit
  float temperatureF= (temperatureC *9.0/5.0) +32.0;

  Serial.print(temperatureF);
  Serial.println(" degrees F");

  delay(1000);
}
```



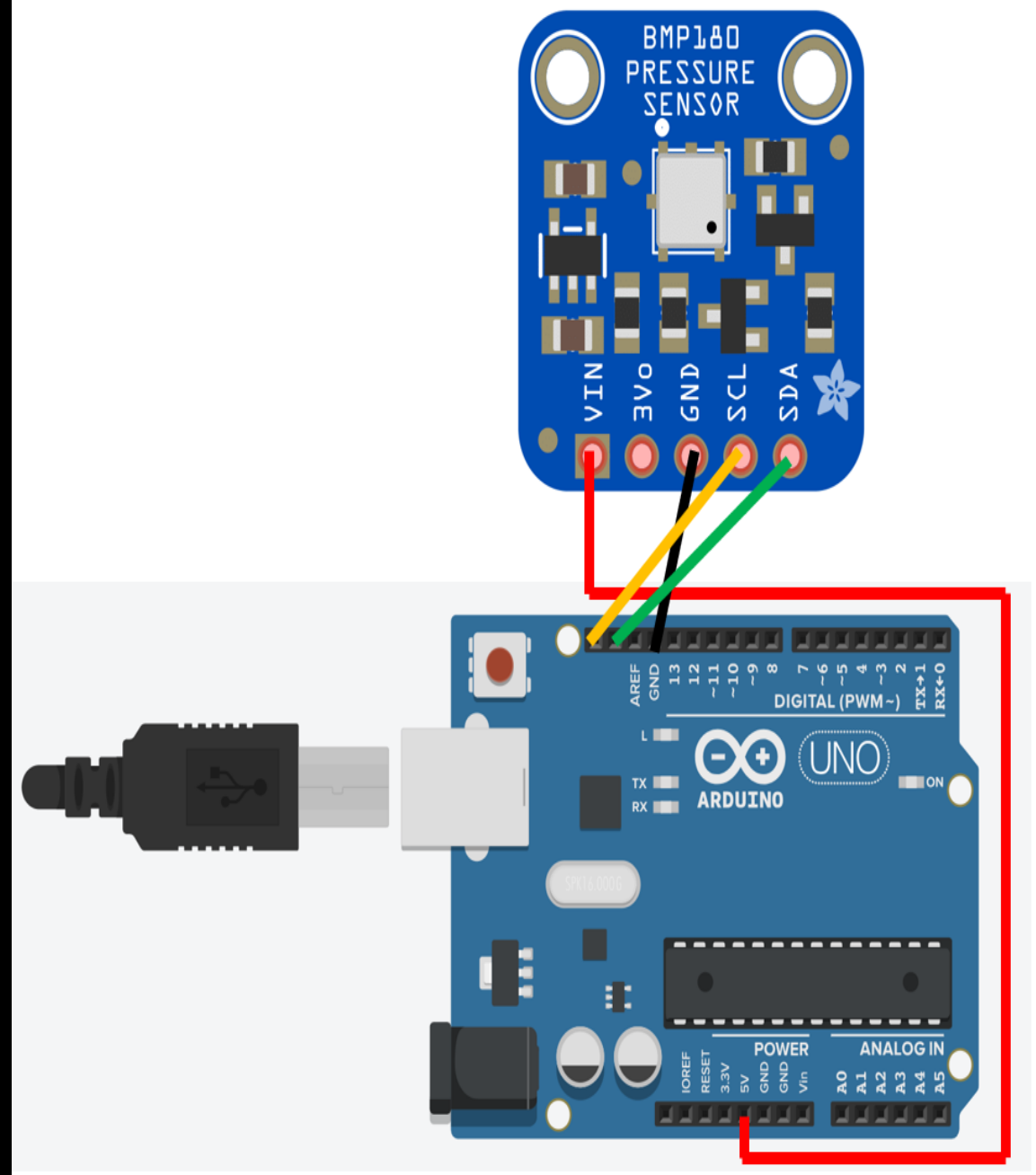

```ino
// Another wiring of TMP36 temperature sensor

int sensorPin=0; // the reading is obtained from analog pin 0 (A0)
float supply_voltage=3.3; //if you are using a 3.3V change it accordingly.

void setup()
{
 analogReference(EXTERNAL); // if using a 3.3V as reference by bridging it to the AREF pin,
                            //then uncomment this line. If using 5V then it is not necessary.
  Serial.begin(9600); // start the serial conversion with the computer
                      // to view the result open the serial monitor;
}

void loop()
{
 // getting the voltage reading from the temperature sensor
  int reading=analogRead(sensorPin);

  // converting thet reading to voltage, for 3.3V arduino use 3.3
  float voltage= reading * supply_voltage/1024;

  float temperatureC=(voltage-0.5)*100; // converting from 10 mv per degree with 500 mv offset
                                        // to degrees ((voltage-500mv) times 100)
  Serial.print(temperatureC);
  Serial.println(" degrees C");

  // now convert to fahrernheit
  float temperatureF= (temperatureC *9.0/5.0) +32.0;

  Serial.print(temperatureF);
  Serial.println(" degrees F");

  delay(1000);
}
```



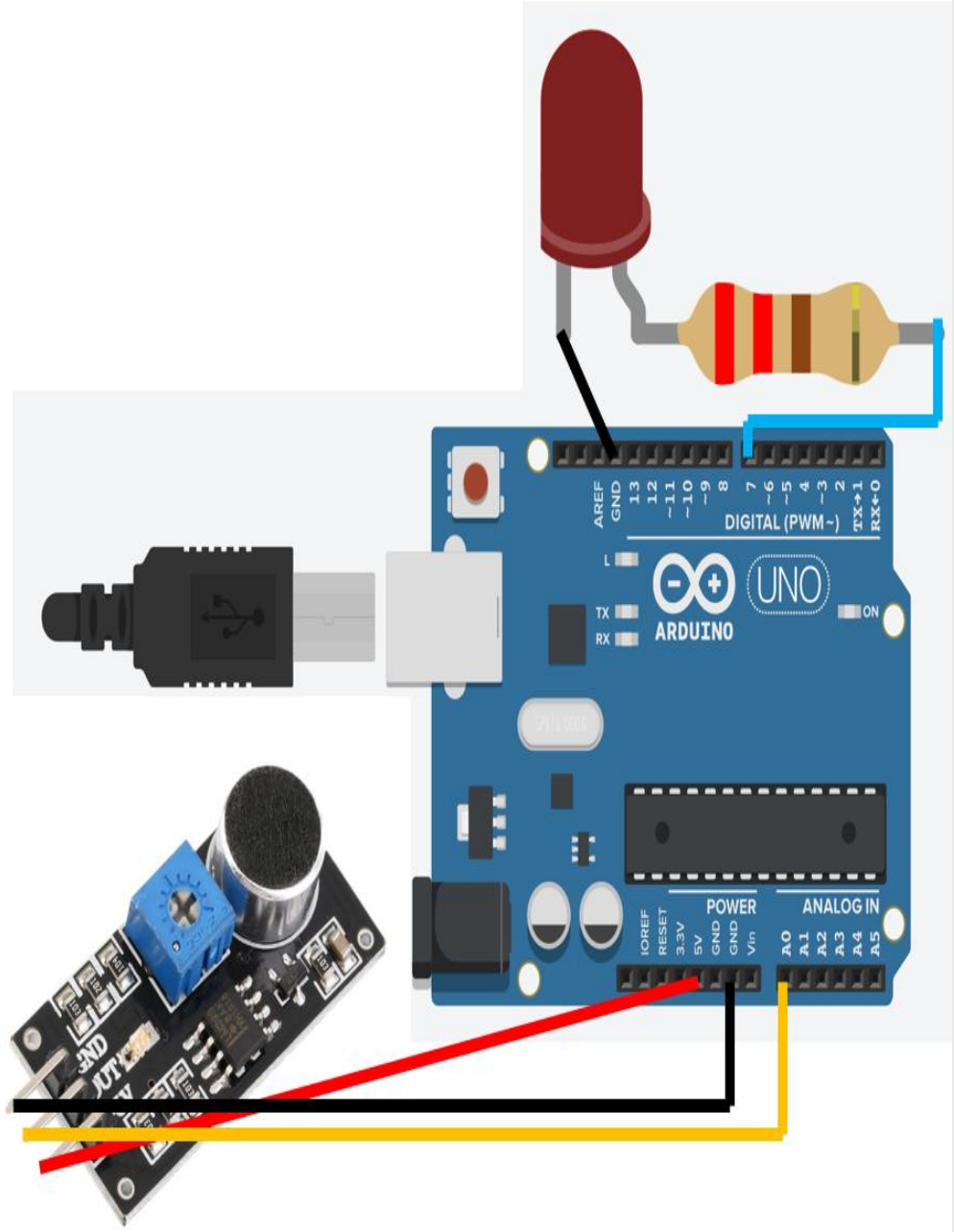

```ino
// Project 13: Working of MCP9808 temperature sensor

#include <Wire.h>
#include <Adafruit_MCP9808.h>

// Create an instance of the sensor
Adafruit_MCP9808 tempsensor = Adafruit_MCP9808();

void setup() {
  Serial.begin(115200);
  Serial.println("MCP9808 Temperature Sensor Test");

  // Initialize the sensor
  if (!tempsensor.begin(0x18)) { // Default I2C address is 0x18
    Serial.println("Couldn't find MCP9808! Check wiring.");
    while (1); // Stop execution
  }

  // Set resolution (optional, 0-3)
  // 0: +0.5°C, 1: +0.25°C, 2: +0.125°C, 3: +0.0625°C
  tempsensor.setResolution(3);
  Serial.println("MCP9808 initialized!");
}

void loop() {
  // Get temperature in Celsius
  float tempC = tempsensor.readTempC();
  // Convert to Fahrenheit
  float tempF = tempC * 9.0 / 5.0 + 32;

  // Print temperatures
  Serial.print("Temperature: ");
  Serial.print(tempC);
  Serial.print(" °C / ");
  Serial.print(tempF);
  Serial.println(" °F");

  delay(1000); // Wait 1 second before next reading
}
```



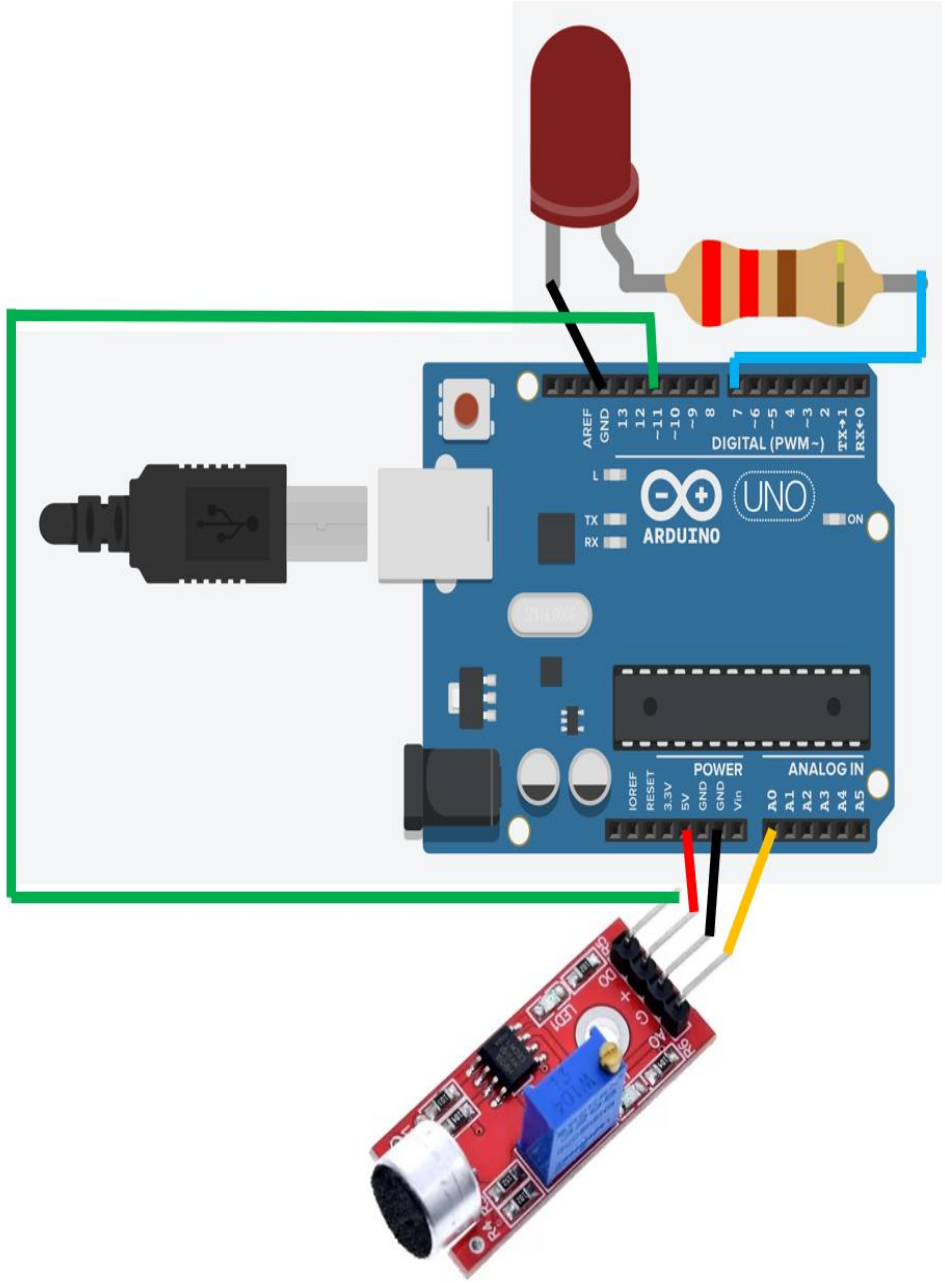

```ino
// Project 14: Working of BMP180 sensor
// In the place of SDA and SCL we also use A4 and A5 analog pins

#include <Wire.h>
#include <SFE_BMP180.h>

// Create an instance of the SFE_BMP180 class
SFE_BMP180 bmp;

// Variables for temperature and pressure
double temperature, pressure;

void setup() {
  Serial.begin(9600);
  Serial.println("BMP180 Sensor Test");

  // Initialize the BMP180
  if (!bmp.begin()) {
    Serial.println("BMP180 initialization failed. Check connections.");
    while (1);
  }
  Serial.println("BMP180 initialized successfully!");
}

void loop() {
  char status;
  double altitude;

  // Start a temperature measurement
  status = bmp.startTemperature();
  if (status != 0) {
    delay(status); // Wait for the measurement to complete

    // Retrieve the temperature
    status = bmp.getTemperature(temperature);
    if (status != 0) {
      Serial.print("Temperature: ");
      Serial.print(temperature, 2);
      Serial.println(" *C");
    } else {
      Serial.println("Error retrieving temperature.");
    }
  } else {
    Serial.println("Error starting temperature measurement.");
  }

  // Start a pressure measurement
  status = bmp.startPressure(3); // Oversampling setting: 0(fastest) to 3(most accurate).
  if (status != 0) {
    delay(status); // Wait for the measurement to complete

    // Retrieve the pressure
    status = bmp.getPressure(pressure, temperature);
    if (status != 0) {
      Serial.print("Pressure: ");
      Serial.print(pressure, 2);
      Serial.println(" Pa");

      // Calculate altitude based on standard sea-level pressure (101325 Pa)
      altitude = bmp.altitude(pressure, 101325);
      Serial.print("Altitude: ");
      Serial.print(altitude, 2);
      Serial.println(" m");
    } else {
      Serial.println("Error retrieving pressure.");
    }
  } else {
    Serial.println("Error starting pressure measurement.");
  }

  delay(2000); // Wait 2 seconds before the next measurement
}
```



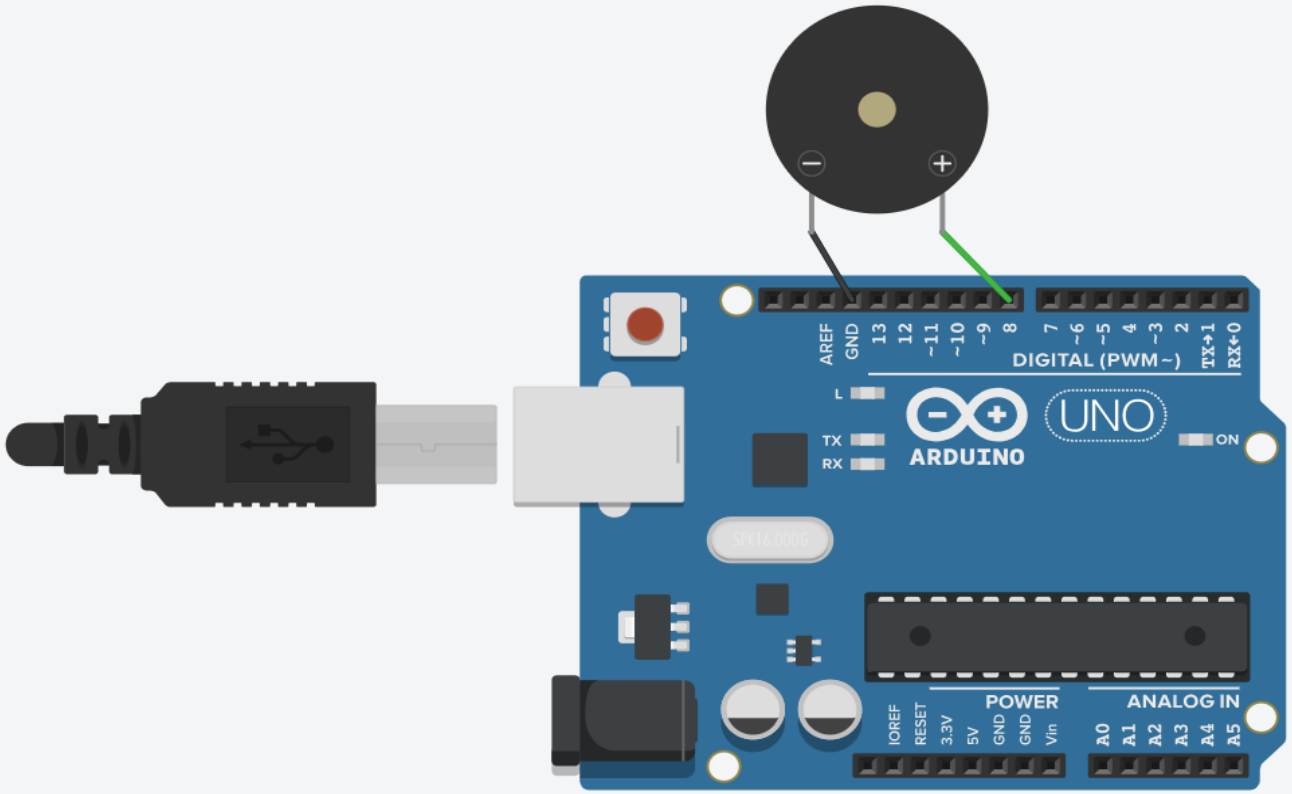

```ino
// Another code of working of BMP180 pressure sensor

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085_U.h>

// Create an instance of the sensor
Adafruit_BMP085_Unified bmp = Adafruit_BMP085_Unified(10085);

void setup() {
  Serial.begin(9600);
  Serial.println("BMP180 Sensor Test");

  // Initialize the BMP180 sensor
  if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP180 sensor, check wiring!");
    while (1);
  }
}

void loop() {
  sensors_event_t event;
  bmp.getEvent(&event);  // Get a new sensor event

  if (event.pressure) {
    // Display pressure in hPa
    Serial.print("Pressure: ");
    Serial.print(event.pressure);
    Serial.println(" hPa");

    // Get temperature for altitude calculation
    float temperature;
    bmp.getTemperature(&temperature);

    // Display temperature in Celsius
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println(" *C");

    // Calculate altitude based on standard sea-level pressure (1013.25 hPa)
    float seaLevelPressure = 1013.25;
    Serial.print("Altitude: ");
    Serial.print(bmp.pressureToAltitude(seaLevelPressure, event.pressure));
    Serial.println(" m");
  } else {
    Serial.println("Sensor error.");
  }

  delay(2000); // Wait 2 seconds before the next reading
}
```



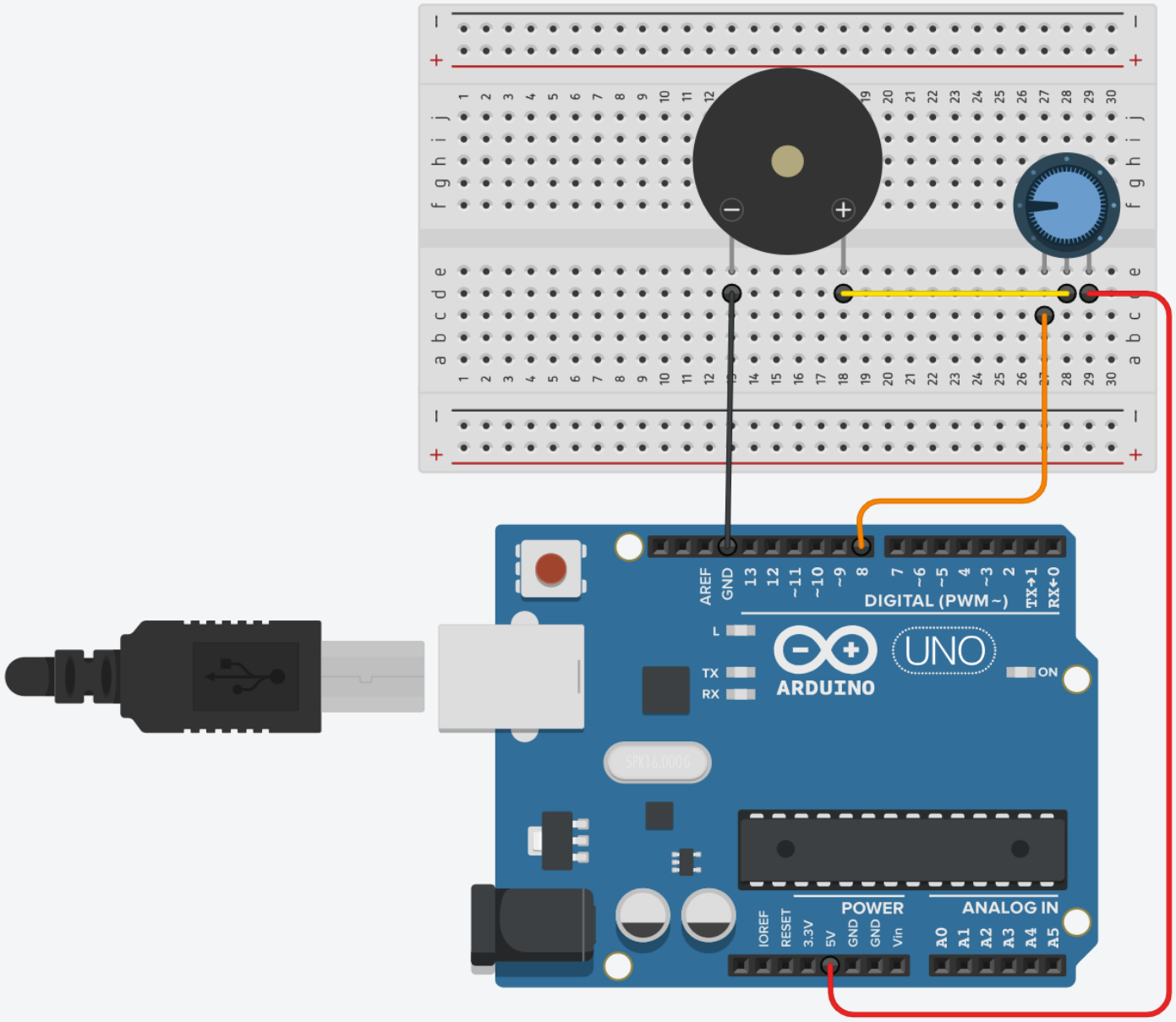

```ino
// Project 15: Working of Analog Sound detection sensor (sensing sound)

const int soundSensorPin = A0; // Analog pin connected to the sound sensor
const int ledPin = 7;          // Digital pin connected to the LED
const int threshold = 500;     // Threshold for sound intensity

void setup() {
  pinMode(ledPin, OUTPUT); // Set the LED pin as an output
  Serial.begin(9600);      // Initialize serial communication
}

void loop() {
  int soundLevel = analogRead(soundSensorPin); // Read sound sensor value

  Serial.print("Sound Level: ");
  Serial.println(soundLevel); // Print the sound level to Serial Monitor

  // Check if the sound level exceeds the threshold
  if (soundLevel > threshold) {
    digitalWrite(ledPin, HIGH); // Turn on the LED
  } else {
    digitalWrite(ledPin, LOW);  // Turn off the LED
  }

  delay(100); // Small delay for readability
}
```



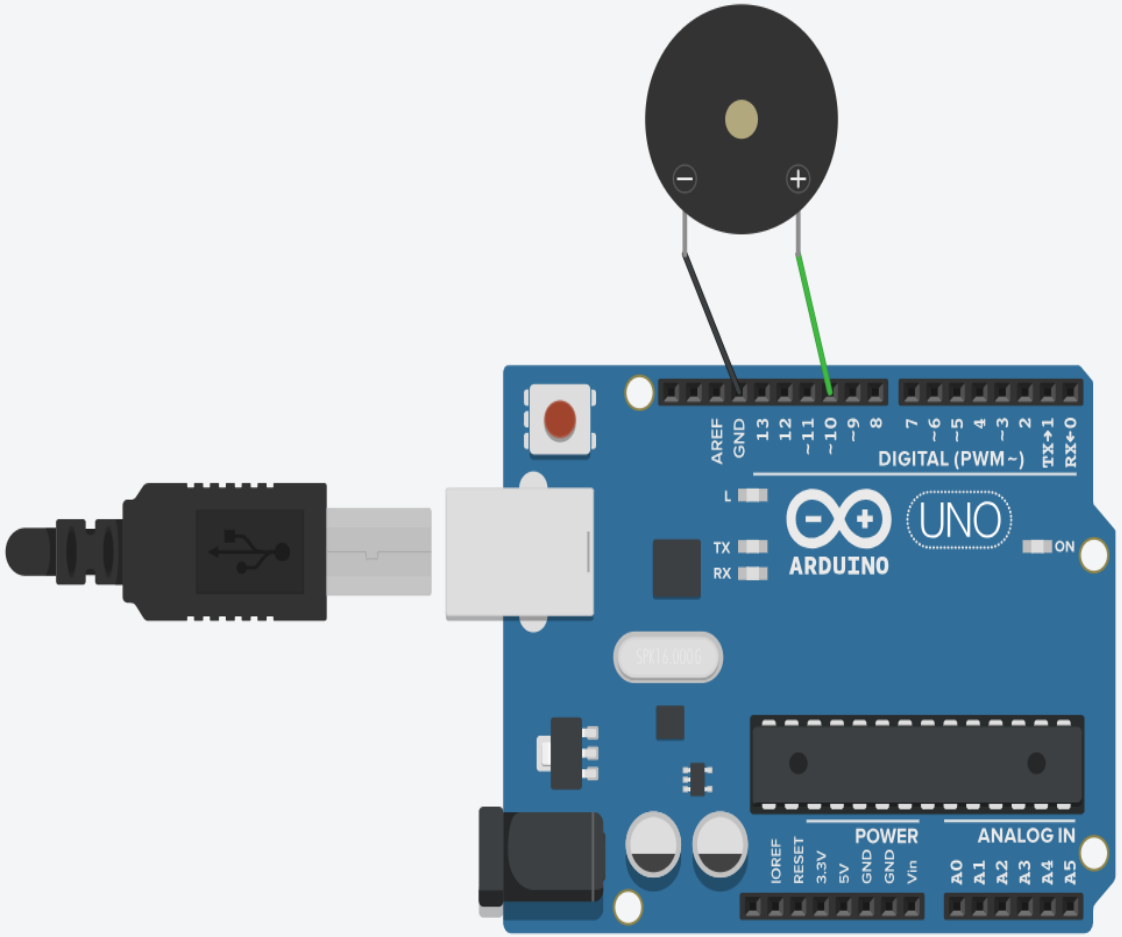

```
// Project 16: Working of Digital Sound detection sensor (sensing sound)

const int soundSensorPin = 11; // Digital pin connected to the sound sensor's D0 pin
const int ledPin = 7;        // Digital pin connected to the LED

void setup() {
  pinMode(soundSensorPin, INPUT); // Set the sound sensor pin as input
  pinMode(ledPin, OUTPUT);        // Set the LED pin as output
  Serial.begin(9600);             // Initialize serial communication
}

void loop() {
  int soundDetected = digitalRead(soundSensorPin); // Read the digital output of the sensor

  if (soundDetected == HIGH) {
    digitalWrite(ledPin, HIGH); // Turn on the LED if sound is detected
    Serial.println("Sound detected!");
  } else {
    digitalWrite(ledPin, LOW); // Turn off the LED if no sound is detected
    Serial.println("No sound detected.");
  }

  delay(100); // Small delay for stability
}
```



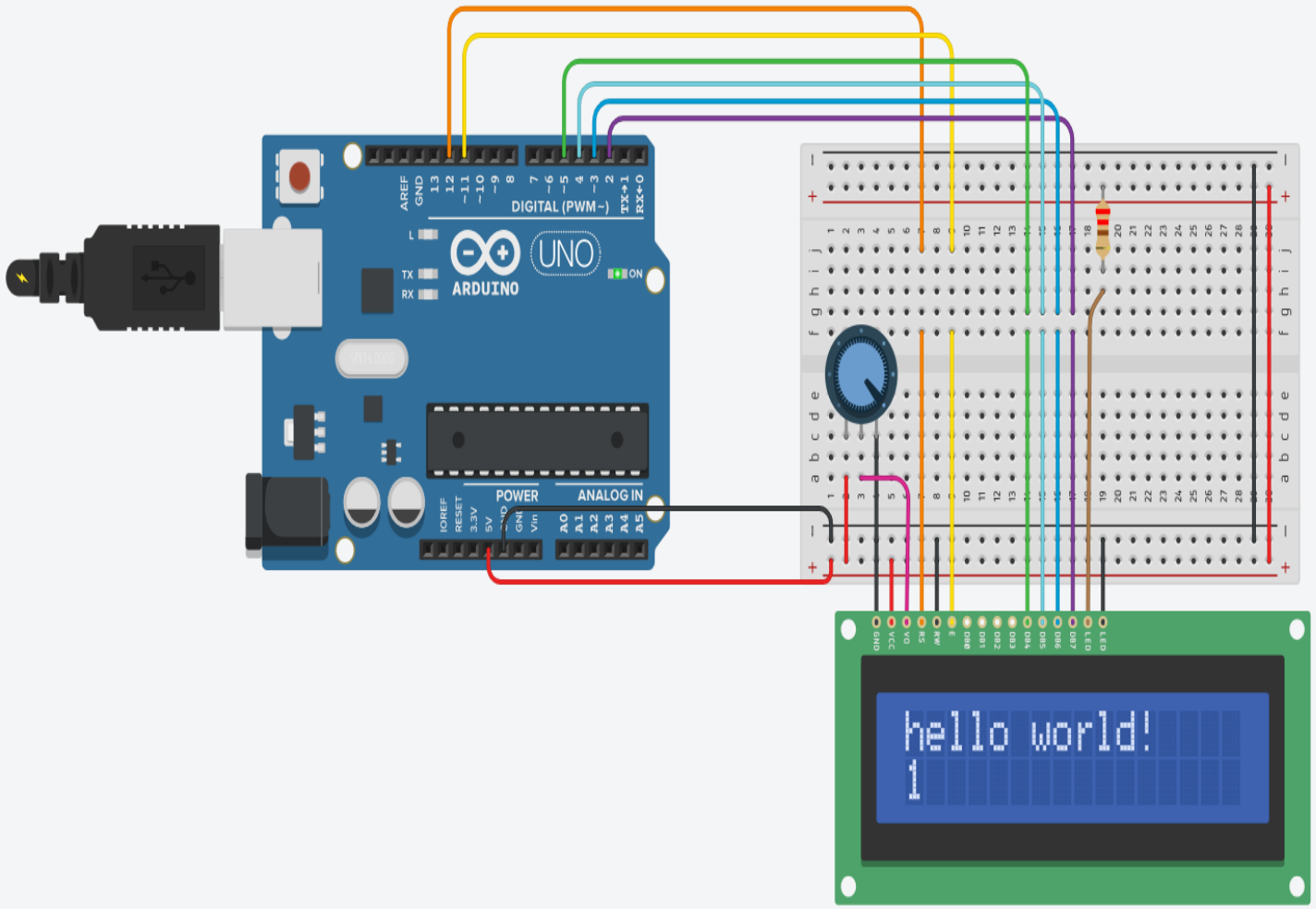

```arduino
// Project 17: Working of Buzzer (produce sound)

// Pin connected to the buzzer
const int buzzerPin = 8;

void setup() {
  // Set the buzzer pin as output
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  // Turn the buzzer on
  digitalWrite(buzzerPin, HIGH);
  delay(500); // Wait for 500 milliseconds (0.5 second)

  // Turn the buzzer off
  digitalWrite(buzzerPin, LOW);
  delay(500); // Wait for 500 milliseconds (0.5 second)
}

// ***************Advanced approach***************


// Pin connected to the buzzer
const int buzzerPin = 8;

void setup() {
  // No setup required for tone
}

void loop() {
  // Play a 1kHz tone for 500ms
  tone(buzzerPin, 1000, 500);
  delay(1000); // Wait for 1 second

  // Play a 500Hz tone for 300ms
  tone(buzzerPin, 500, 300);
  delay(1000); // Wait for 1 second

  // No tone for 1 second
  noTone(buzzerPin);
  delay(1000);
}
```



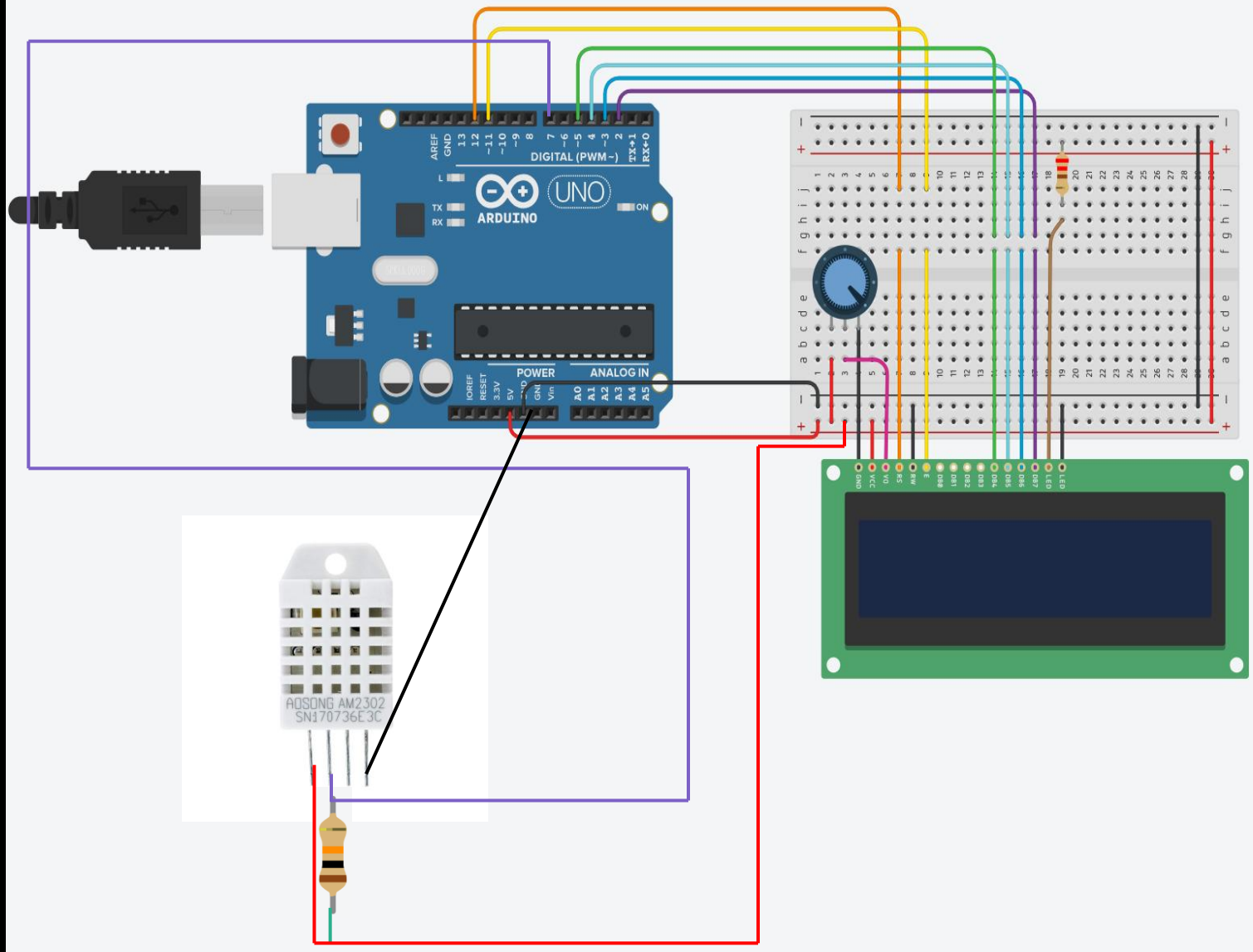

```arduino
// Working of passive Buzzer (produce sound)
// Adjusting volume by using potentiometer

// Pin connected to the buzzer
const int buzzerPin = 8;

void setup() {
  // Set the buzzer pin as output
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  // Turn the buzzer on
  digitalWrite(buzzerPin, HIGH);
  delay(500); // Wait for 500 milliseconds (0.5 second)

  // Turn the buzzer off
  digitalWrite(buzzerPin, LOW);
  delay(500); // Wait for 500 milliseconds (0.5 second)
}

// ***************Advanced approach***************


 Pin connected to the buzzer
const int buzzerPin = 8;

void setup() {
  // No setup required for tone
}

void loop() {
  // Play a 1kHz tone for 500ms
  tone(buzzerPin, 1000, 500);
  delay(1000); // Wait for 1 second

  // Play a 500Hz tone for 300ms
  tone(buzzerPin, 500, 300);
  delay(1000); // Wait for 1 second

  // No tone for 1 second
  noTone(buzzerPin);
  delay(1000);
}
```



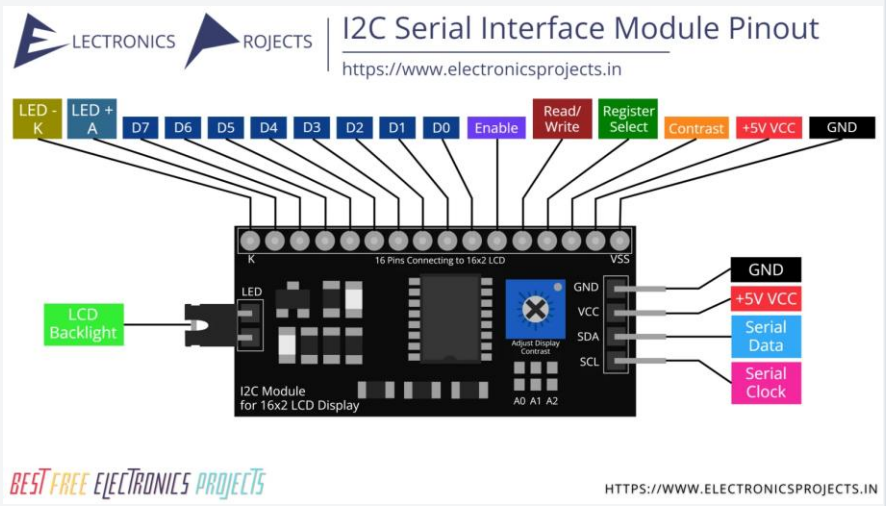

```ino
// Project 18: Melody tone by passive buzzer
#define NOTE_C4  262
#define NOTE_G3  196
#define NOTE_A3  220
#define NOTE_B3  247
#define NOTE_C4  262


// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {      //4 means a quarter note (1/4 of a whole note).
    4, 8, 8, 4, 4, 4, 4, 4}; //8 means an eighth note (1/8 of a whole note).

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc. (in milli sec)
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(10, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(10);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

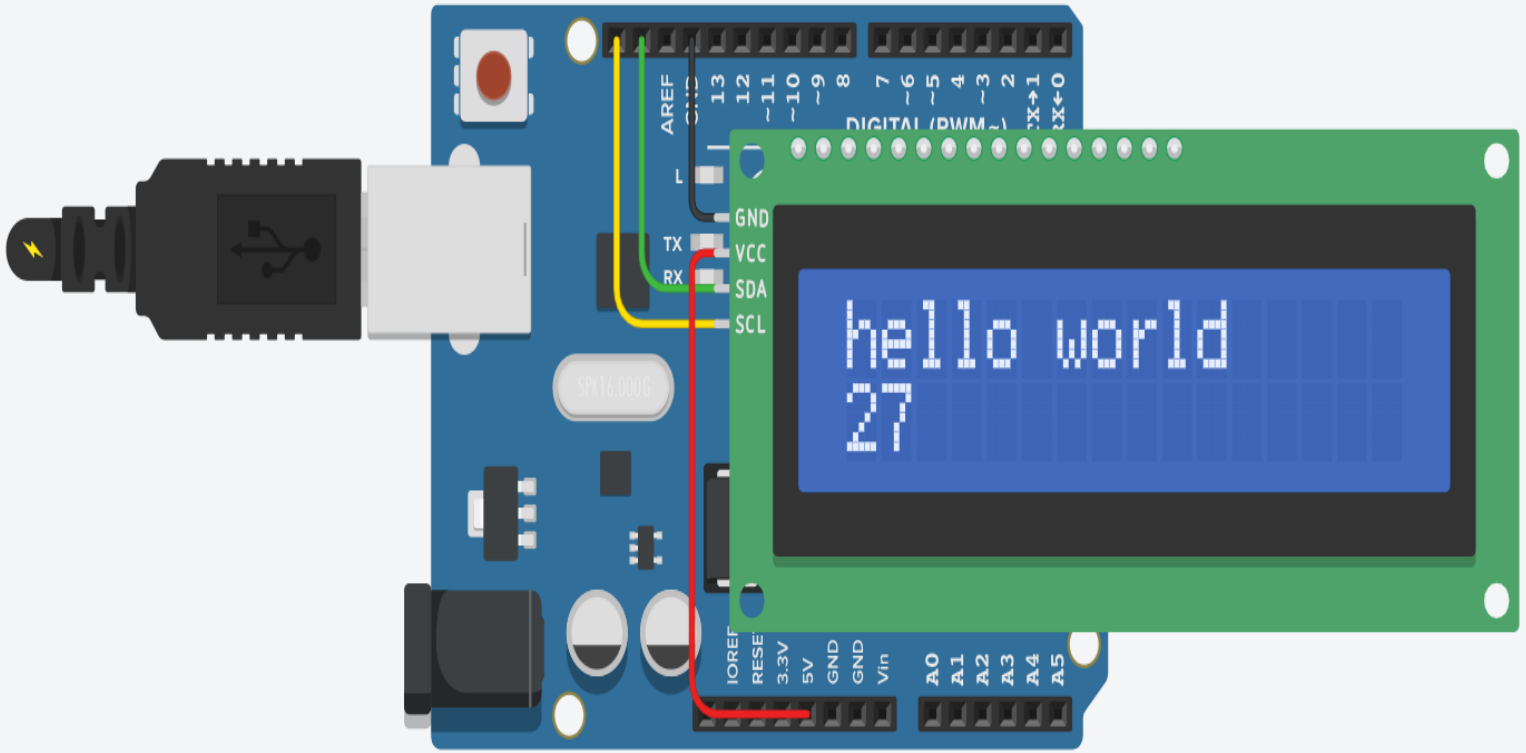



```ino
// Project 19: Working of LCD (16*2)

#include <LiquidCrystal.h>

int seconds = 0;

LiquidCrystal lcd_1(12, 11, 5, 4, 3, 2);

void setup()
{
  lcd_1.begin(16, 2); // Set up the number of columns and rows on the LCD.

  // Print a message to the LCD.
  lcd_1.print("hello world!");
}

void loop()
{
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting
  // begins with 0):
  lcd_1.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd_1.print(seconds);
  delay(1000); // Wait for 1000 millisecond(s)
  seconds += 1;
}
```

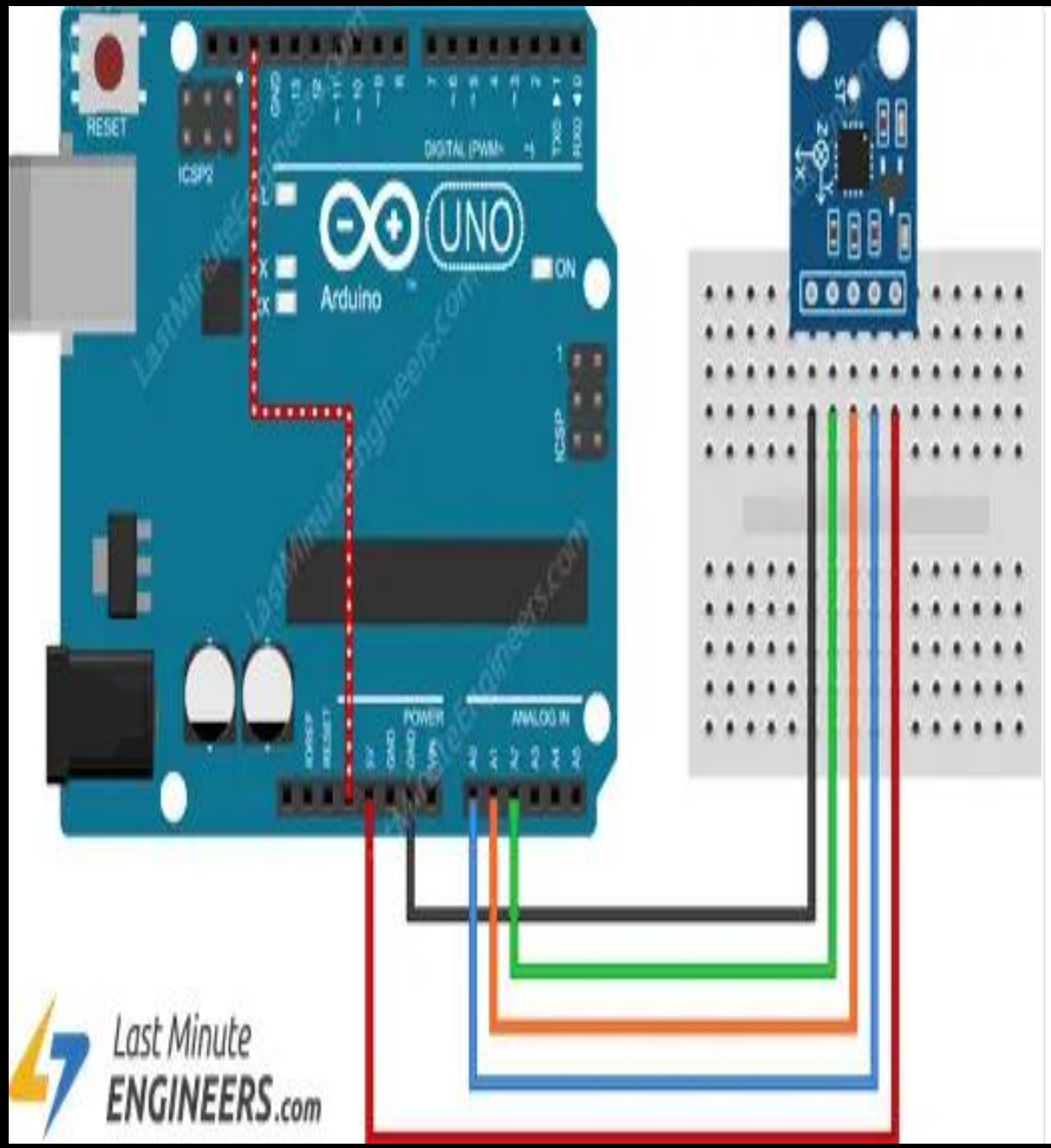



```ino
// Project 20: Display of DHT22 sensor readings on LCD (16*2)

#include <LiquidCrystal.h>
#include <DHT.h>

// Initialize the LCD (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Define the DHT sensor
#define DHTPIN 7       // Pin connected to the DATA pin of DHT22
#define DHTTYPE DHT22  // Specify DHT sensor type
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  // Initialize the LCD
  lcd.begin(16, 2); // Set the LCD size: 16x2
  lcd.print("Initializing...");

  dht.begin(); // Initialize the DHT sensor

  delay(2000); // Allow time to initialize
  lcd.clear();
}

void loop() {
  // Read temperature and humidity from DHT22
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (isnan(humidity) || isnan(temperature)) { // isnan(is not a number)
    lcd.setCursor(0, 0);
    lcd.print("Error reading");
    lcd.setCursor(0, 1);
    lcd.print("sensor data");
    delay(2000); // Wait before retrying
    return;
  }
            // Display temperature and humidity on LCD
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temperature);
  lcd.print(" C");

  lcd.setCursor(0, 1);
  lcd.print("Hum:  ");
  lcd.print(humidity);
  lcd.print(" %");

  delay(2000); // Update every 2 seconds
}
```

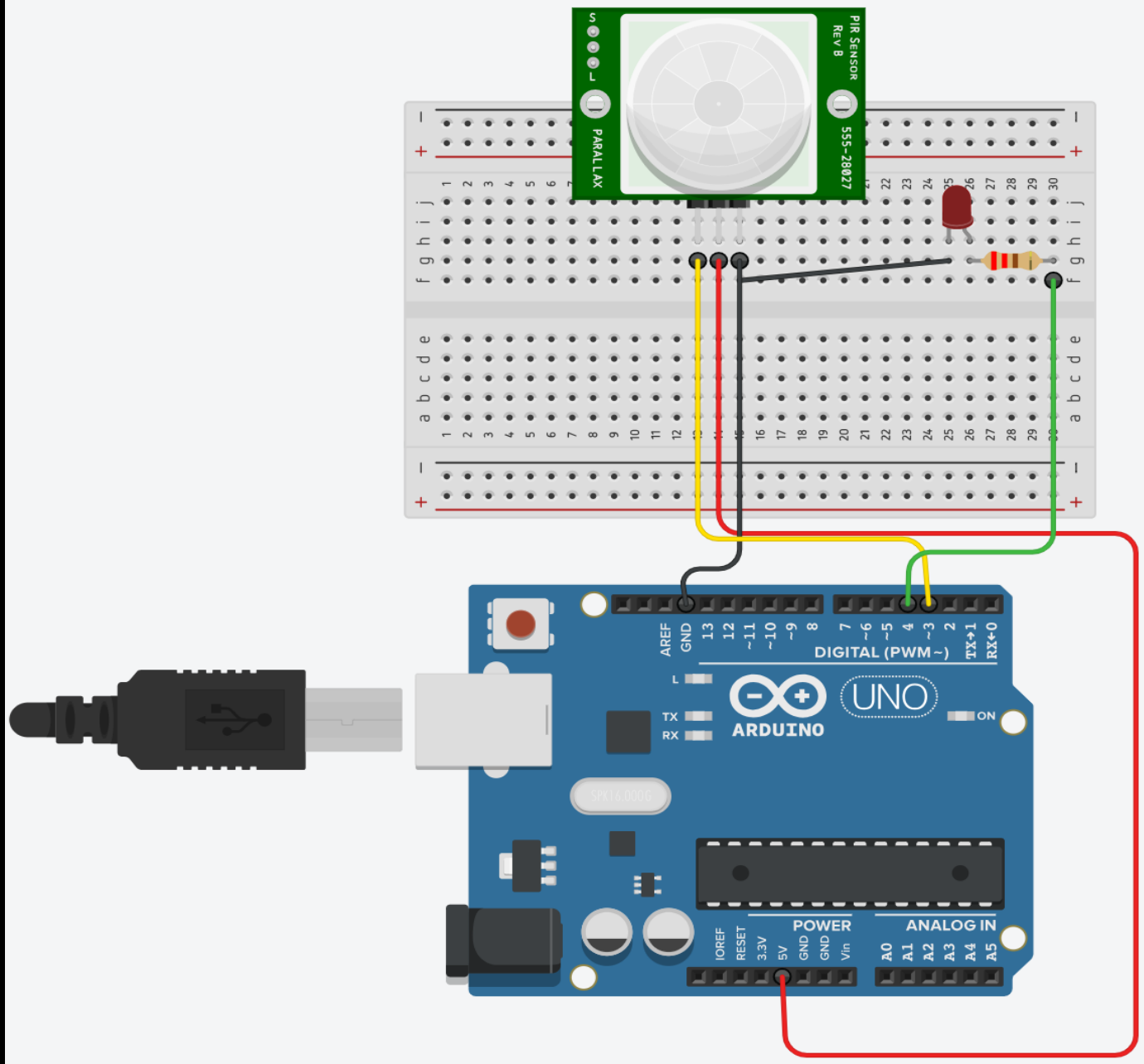




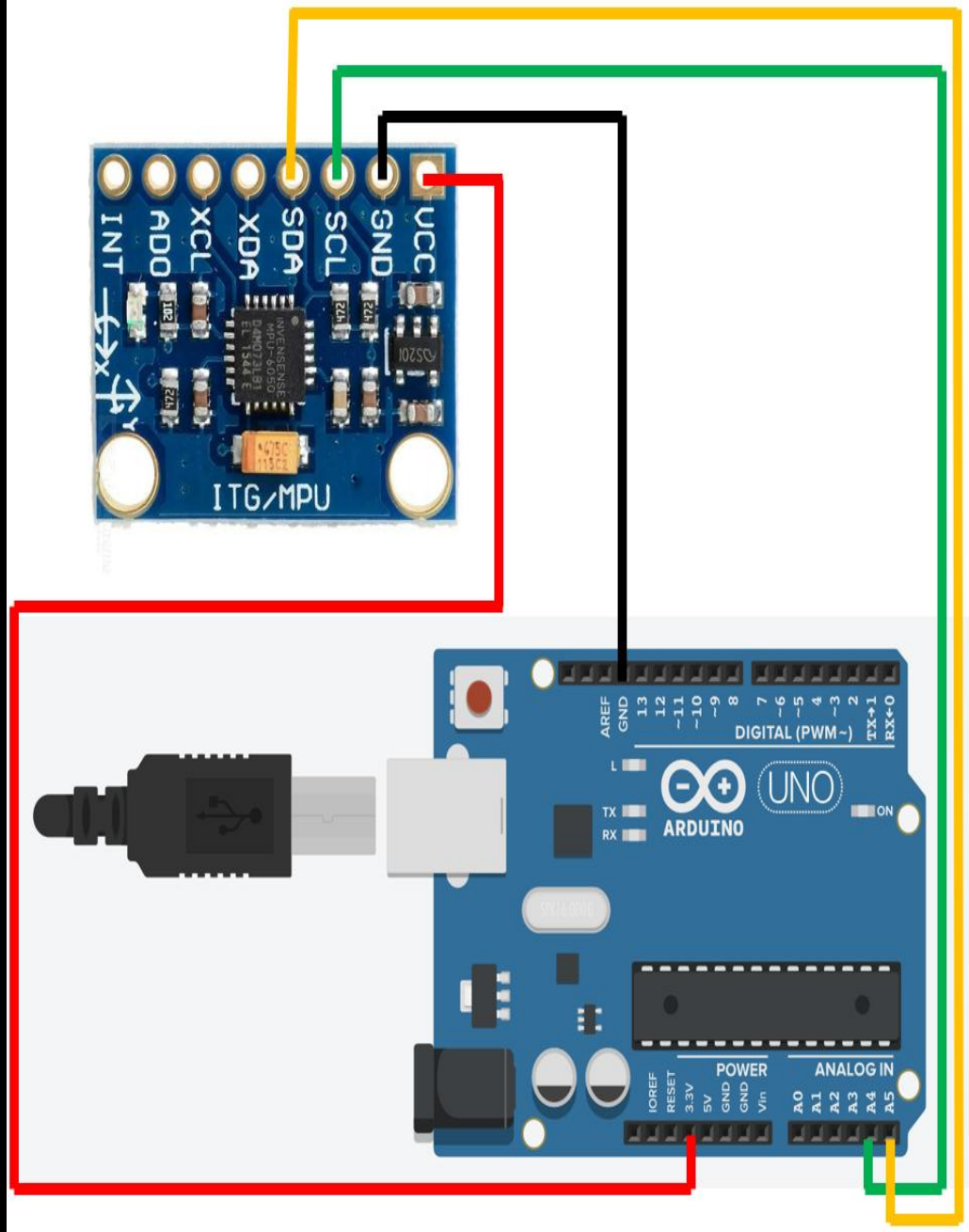

```arduino
// Project 21: Working of LCD (16*2) by I2C module

#include <Adafruit_LiquidCrystal.h>

int seconds = 0;

Adafruit_LiquidCrystal lcd_1(0);

void setup()
{
  lcd_1.begin(16, 2);

  lcd_1.print("hello world");
}

void loop()
{
  lcd_1.setCursor(0, 1);
  lcd_1.print(seconds);
  lcd_1.setBacklight(1);
  delay(500); // Wait for 500 millisecond(s)
  lcd_1.setBacklight(0);
  delay(500); // Wait for 500 millisecond(s)
  seconds += 1;
}
```




```ino
// Project 22: Working of ADXL355 accelaration sensor

const int xPin = A0; // X-axis analog output
const int yPin = A1; // Y-axis analog output
const int zPin = A2; // Z-axis analog output

void setup() {
  Serial.begin(9600); // Initialize Serial Monitor
}

void loop() {
  // Read analog values from the sensor
  int xValue = analogRead(xPin);
  int yValue = analogRead(yPin);
  int zValue = analogRead(zPin);

  // Convert raw values to acceleration in g (optional calibration)
  float xAcc = map(xValue, 0, 1023, -3, 3);
  float yAcc = map(yValue, 0, 1023, -3, 3);
  float zAcc = map(zValue, 0, 1023, -3, 3);

  // Print values to Serial Monitor
  Serial.print("X: ");
  Serial.print(xAcc);
  Serial.print(" g, Y: ");
  Serial.print(yAcc);
  Serial.print(" g, Z: ");
  Serial.print(zAcc);
  Serial.println(" g");

  delay(500); // Delay for readability
}
```




```ino
//Project 23: Workng of PIR sensor

// Pin Definitions
#define pirSensorPin 3 // PIR motion sensor pin
#define ledPin 4       // LED pin

void setup() {
  // Initialize Serial Monitor
  Serial.begin(9600);

  // Configure Pins
  pinMode(pirSensorPin, INPUT); // PIR sensor as input
  pinMode(ledPin, OUTPUT);      // LED as output

  Serial.println("PIR Sensor Test Initialized");
}

void loop() {
  // Read the PIR sensor state
  bool motionDetected = digitalRead(pirSensorPin);

  if (motionDetected) {
    // Turn on LED if motion is detected
    digitalWrite(ledPin, HIGH);
    Serial.println("Motion Detected: LED ON");
  } else {
    // Turn off LED if no motion is detected
    digitalWrite(ledPin, LOW);
    Serial.println("No Motion: LED OFF");
  }

  delay(100); // Short delay for stability
}

```




```
// Project 24: Working of MPU6050 Sensor

// Include necessary libraries for I2C and MPU6050
#include <Wire.h>          // For I2C communication
#include <MPU6050.h>       // MPU6050 library by Jeff Rowberg

// Create an object of the MPU6050 class
MPU6050 mpu;

void setup() {
  Serial.begin(9600);      // Start serial communication at 9600 baud rate
  Wire.begin();            // Initialize I2C communication

  Serial.println("Initializing MPU6050...");
  mpu.initialize();        // Wake up the MPU6050 and configure registers

  // Check if the MPU6050 is connected properly
  if (mpu.testConnection()) {
    Serial.println("MPU6050 connected successfully!");
  } else {
    Serial.println("MPU6050 connection failed!");
    while (1);  // Stop execution if sensor not detected
  }
}

void loop() {
  // Declare variables to hold raw accelerometer, gyroscope, and temperature values
  int16_t ax, ay, az;      // Accelerometer X, Y, Z
  int16_t gx, gy, gz;      // Gyroscope X, Y, Z
  int16_t tempRaw;         // Raw temperature

  // Read all 6 motion values from the sensor (accel + gyro)
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  // Read raw temperature
  tempRaw = mpu.getTemperature();

  // Convert raw accelerometer values to g (gravity units)
  float ax_g = ax / 16384.0;
  float ay_g = ay / 16384.0;
  float az_g = az / 16384.0;

  // Convert raw gyroscope values to degrees per second (°/s)
  float gx_deg = gx / 131.0;
  float gy_deg = gy / 131.0;
  float gz_deg = gz / 131.0;

  // Convert raw temperature to Celsius using datasheet formula
  float tempC = tempRaw / 340.0 + 36.53;
// Print accelerometer data in g
Serial.print("Accel (g): X=");
Serial.print(ax_g, 2);     // X-axis acceleration
Serial.print(" Y=");
Serial.print(ay_g, 2);     // Y-axis acceleration
Serial.print(" Z=");
Serial.print(az_g, 2);     // Z-axis acceleration

// Print gyroscope data in degrees/sec
Serial.print(" | Gyro (°/s): X=");
Serial.print(gx_deg, 2);   // X-axis rotation
Serial.print(" Y=");
Serial.print(gy_deg, 2);   // Y-axis rotation
Serial.print(" Z=");
Serial.print(gz_deg, 2);   // Z-axis rotation

// Print temperature in Celsius
Serial.print(" | Temp: ");
Serial.print(tempC, 2);
Serial.println(" °C");

// Wait for 500 milliseconds before next reading
delay(500);
```