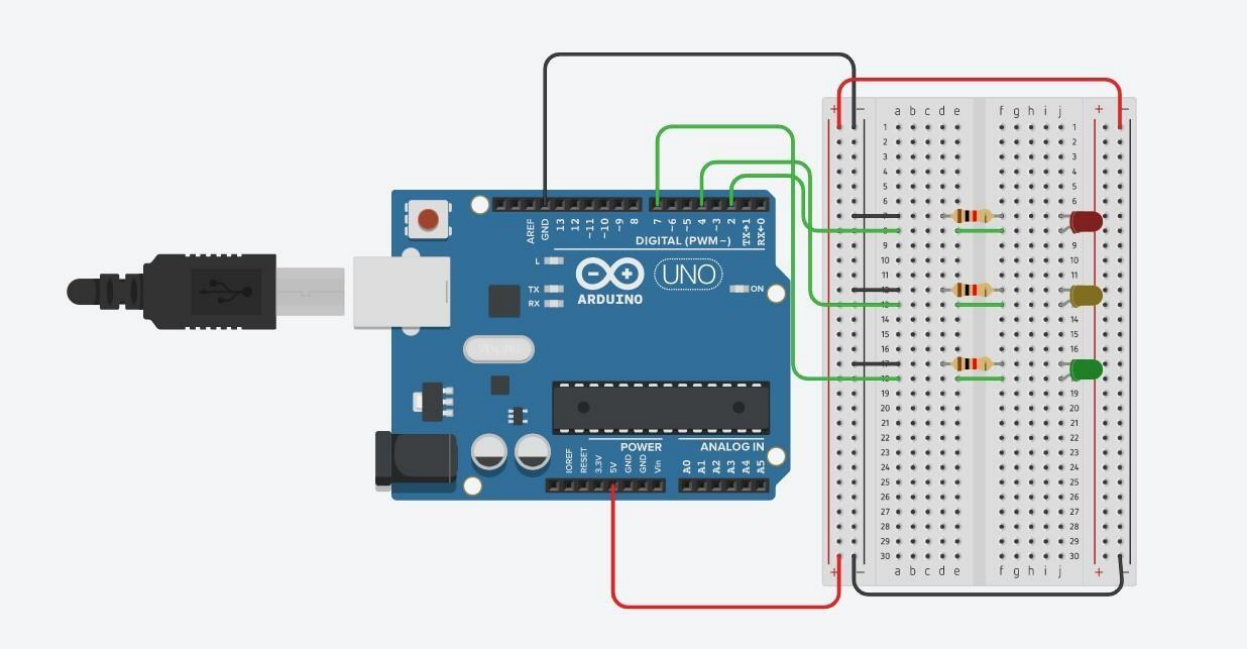
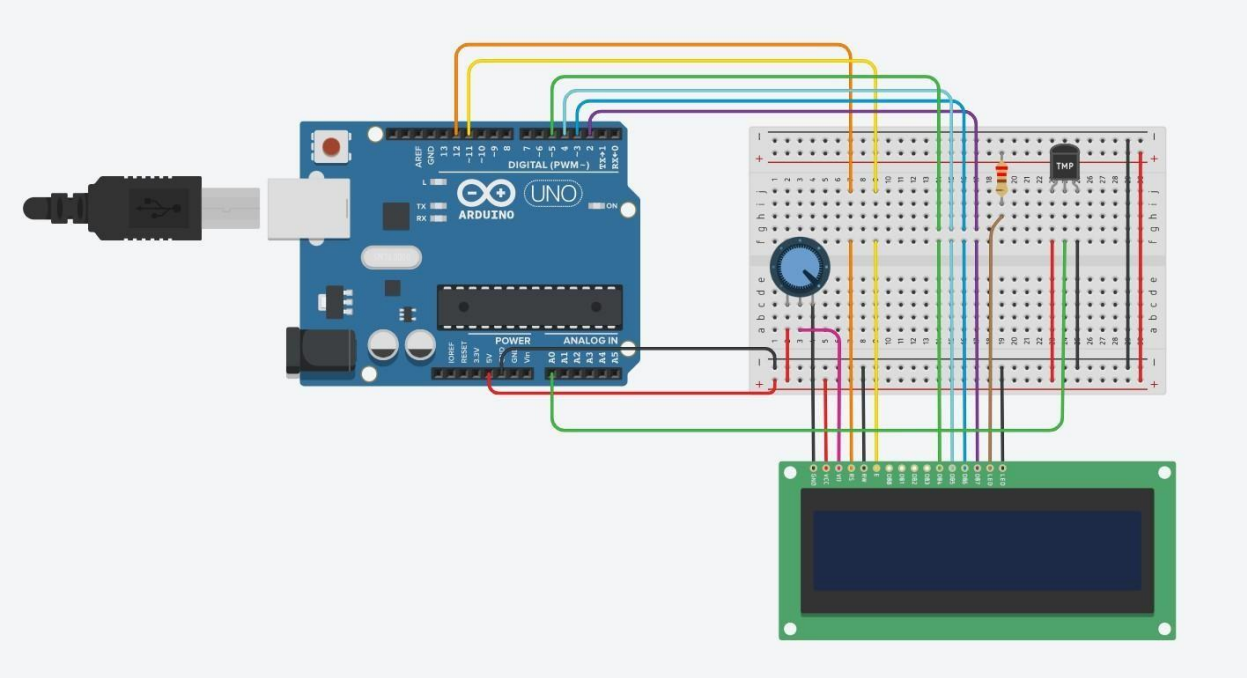
1. **Blinking LED**




```
// void
setup()
{
  pinMode(2, OUTPUT);
pinMode(4, OUTPUT);   pinMode(7,
OUTPUT);



}


void loop()
{ digitalWrite(2, HIGH);
delay(1000);   digitalWrite(2,
LOW);   digitalWrite(4,
HIGH);   delay(1000);
digitalWrite(4, LOW);
digitalWrite(7, HIGH);
```

```
delay(1000);   digitalWrite(7,
LOW);
}
```

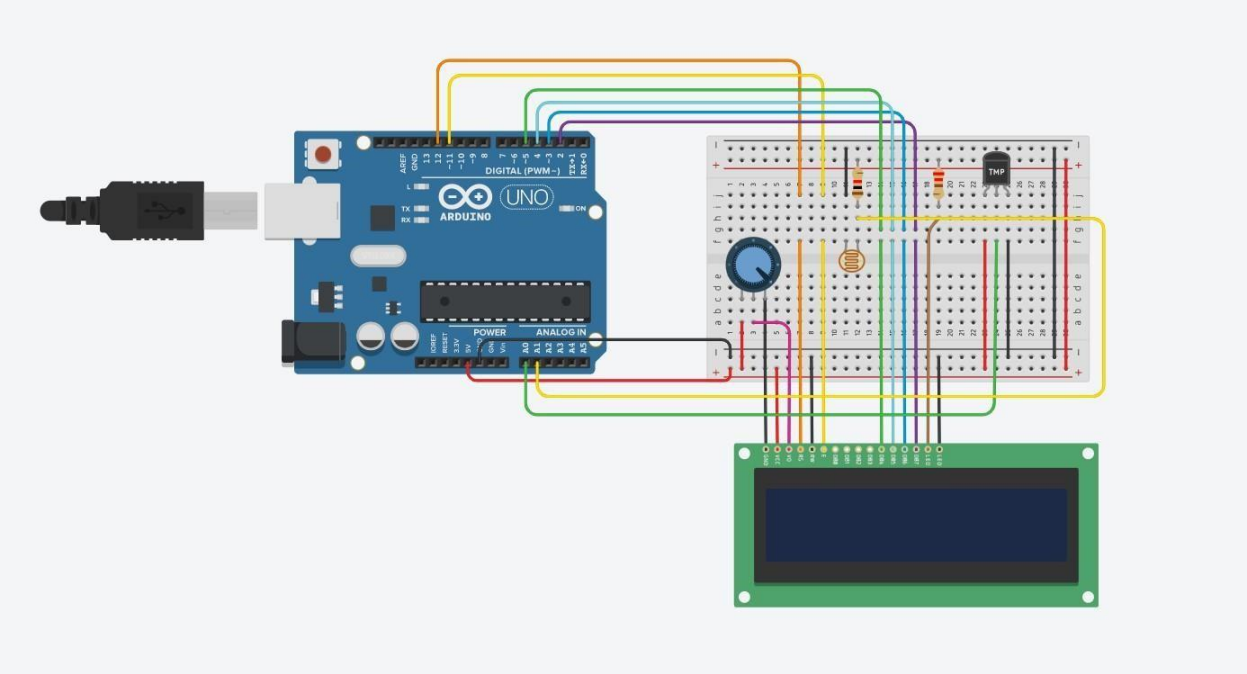
2. **Temperature Sensor**




```
#include <LiquidCrystal.h>

int seconds = 0;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
float value; int tmp = A0;

void setup()
{
  lcd.begin(16, 2);
pinMode(tmp,INPUT);
}

void loop()
{
```

```
   value = analogRead(tmp)*0.004882814;
value = (value - 0.5) * 100.0;
lcd.setCursor(0,1);         lcd.print("Tmp:");
lcd.print(value);           delay(1000);
lcd.clear();
}
```

3. **Temp and LDR Sensor**

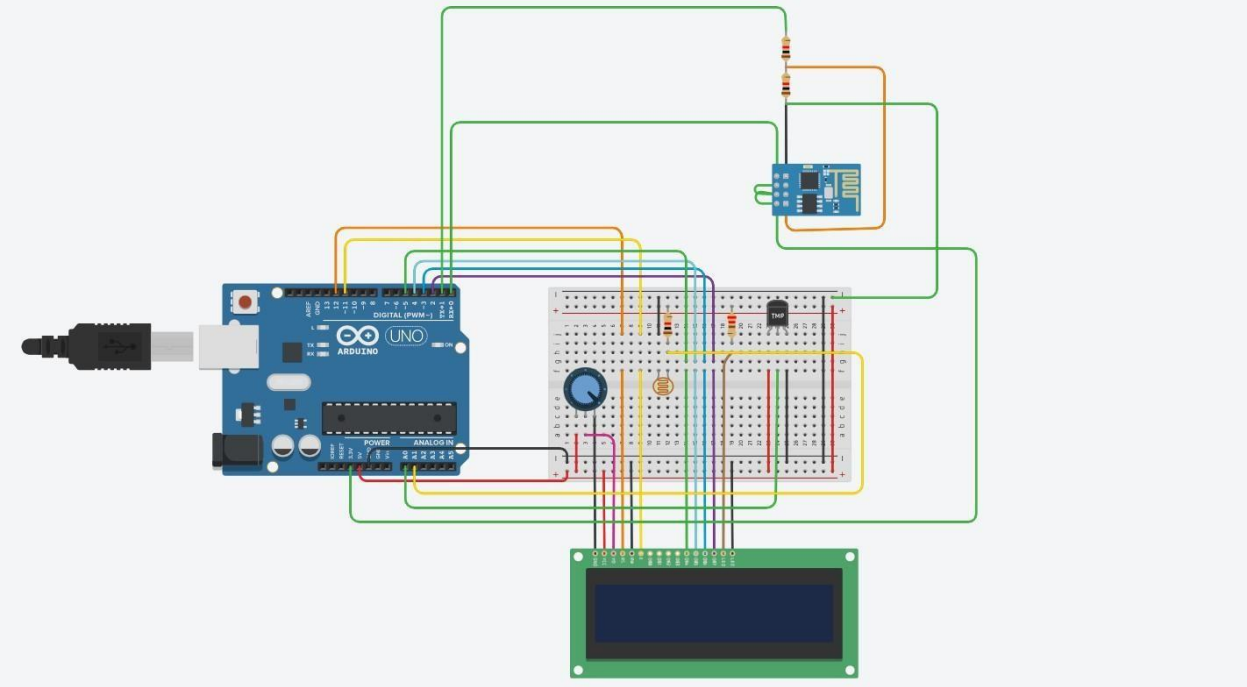



```
#include <LiquidCrystal.h>

int seconds = 0;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
float value; int tmp = A0; int ldr =
A1;
float RLDR;
float Vout;

void setup()
{
  lcd.begin(16, 2);
  pinMode(tmp,INPUT);
pinMode(ldr,INPUT);

}
```

```
void loop()
{
   value = analogRead(tmp)*0.004882814;
value = (value - 0.5) * 100.0;
delay(1000);      lcd.clear();



            int sensorValue = analogRead(A1);

   Vout = (sensorValue * 0.0048828125);
   RLDR = (10000.0 * (5 - Vout))/Vout;

   lcd.clear();

            delay(100);

            if (((RLDR/500) >= 0) && ((RLDR/500) <= 5))
            {
                        lcd.clear();
            lcd.setCursor(0,0);
                        lcd.print("Tmp:");
lcd.print(value);
            lcd.setCursor(0, 1);
lcd.print("DARK");
            }
            else if (((RLDR/500) > 5) && ((RLDR/500) <= 14))              {
                        lcd.clear();
            lcd.setCursor(0,0);
```

```
                lcd.print("Tmp:");
lcd.print(value);
        lcd.setCursor(0, 1);
lcd.print("DIM");
        }
        else if (((RLDR/500) > 14) && ((RLDR/500) <= 50))
        {
                lcd.clear();
        lcd.setCursor(0,0);
                lcd.print("Tmp:");
lcd.print(value);
        lcd.setCursor(0, 1);
        lcd.print("BRIGHT");
        }
        else
        {
                lcd.clear();
        lcd.setCursor(0,0);
                lcd.print("Tmp:");
lcd.print(value);
    lcd.setCursor(0, 1);       lcd.print("VERY
BRIGHT");
        }
}
```

4. **Sending Multiple sensor Data to database**



```
#include <LiquidCrystal.h>

int seconds = 0;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
float value; int tmp = A0; int ldr =
A1;
float RLDR;
float Vout;
String env;

String ssid     = "Simulator Wifi";
String password = "";
String host     = "api.thingspeak.com";
const int httpPort   = 80;
```

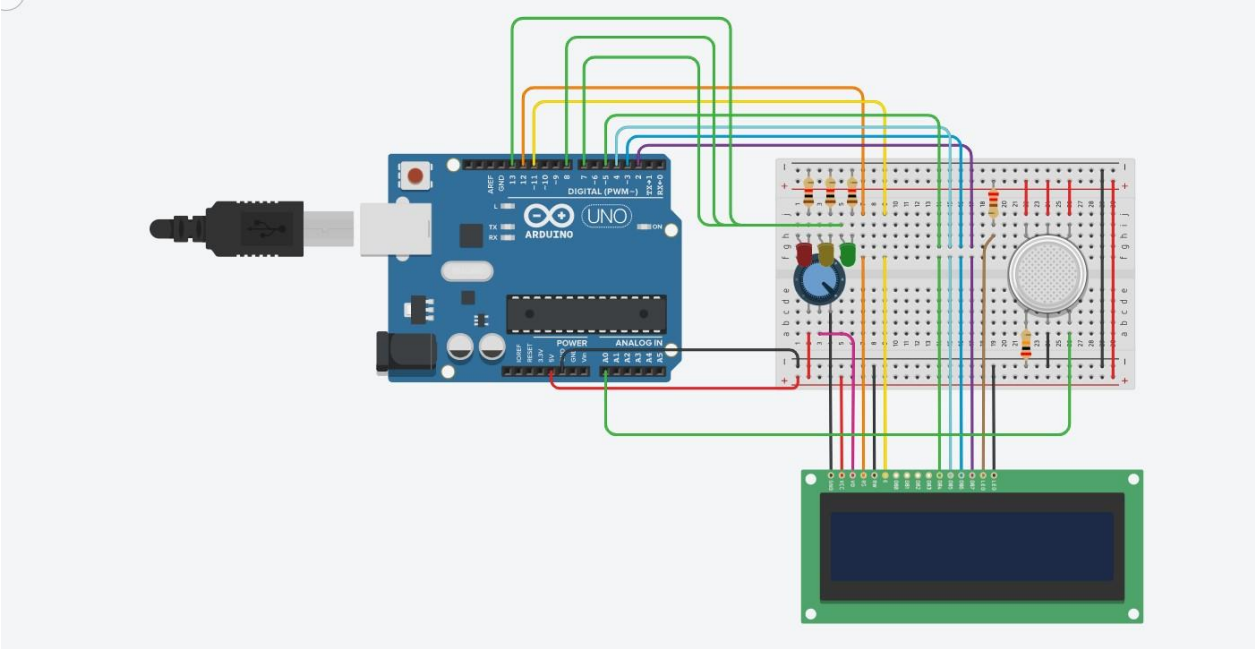
```
String uri_1     = "/update?api_key=0CNARR0CD2TJ6LR7&field1=";

int setupESP8266(void) {

  Serial.begin(115200);
Serial.println("AT");      delay(10);
  if (!Serial.find("OK")) return 1;



  Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"");
delay(10);   if (!Serial.find("OK")) return 2;

  // Open TCP connection to the host:
  Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\"," + httpPort);
delay(50);       // Wait a little for the ESP to respond   if
(!Serial.find("OK")) return 3;

  return 0;
}

void setup()
{
  lcd.begin(16, 2);
pinMode(tmp,INPUT);
pinMode(ldr,INPUT);   setupESP8266();
}

void loop()
{
```

```
   value = analogRead(tmp)*0.004882814;
value = (value - 0.5) * 100.0;
          delay(1000);
lcd.clear();



          int sensorValue = analogRead(A1);

   Vout = (sensorValue * 0.0048828125);
   RLDR = (10000.0 * (5 - Vout))/Vout;

   lcd.clear();

          delay(100);

          if (((RLDR/500) >= 0) && ((RLDR/500) <= 5))
          {
                    env = "DARK";
          lcd.clear();
          lcd.setCursor(0,0);
                    lcd.print("Tmp:");
lcd.print(value);
          lcd.setCursor(0, 1);
lcd.print(env);
          }
          else if (((RLDR/500) > 5) && ((RLDR/500) <= 14))          {
                    env = "DIM";
          lcd.clear();
```

```
        lcd.setCursor(0,0);
                lcd.print("Tmp:");
lcd.print(value);
        lcd.setCursor(0, 1);
        lcd.print(env);
        }
        else if (((RLDR/500) > 14) && ((RLDR/500) <= 50))
        {
                env = "BRIGHT";
        lcd.clear();
        lcd.setCursor(0,0);
                lcd.print("Tmp:");
lcd.print(value);
        lcd.setCursor(0, 1);
        lcd.print(env);
        }
        else
        {
                env = "VERY BRIGHT";
        lcd.clear();
lcd.setCursor(0,0);
                lcd.print("Tmp:");
lcd.print(value);
        lcd.setCursor(0, 1);
lcd.print(env);
        }
  String httpPacket = "GET " + uri_1 + String(value) + "&field2=" + env + " HTTP/1.1\r\nHost: " + host +
"\r\n\r\n";   int length =
httpPacket.length();
```

```
  // Send our message length
  Serial.print("AT+CIPSEND=");   Serial.println(length);   delay(10); // Wait a
little for the ESP to respond if (!Serial.find(">")) return -1;

  // Send our http request
Serial.print(httpPacket);   delay(10); // Wait a
little for the ESP to respond   if
(!Serial.find("SEND OK\r\n")) return;

  delay(1024);
}
```

5. **Gas Sensor**

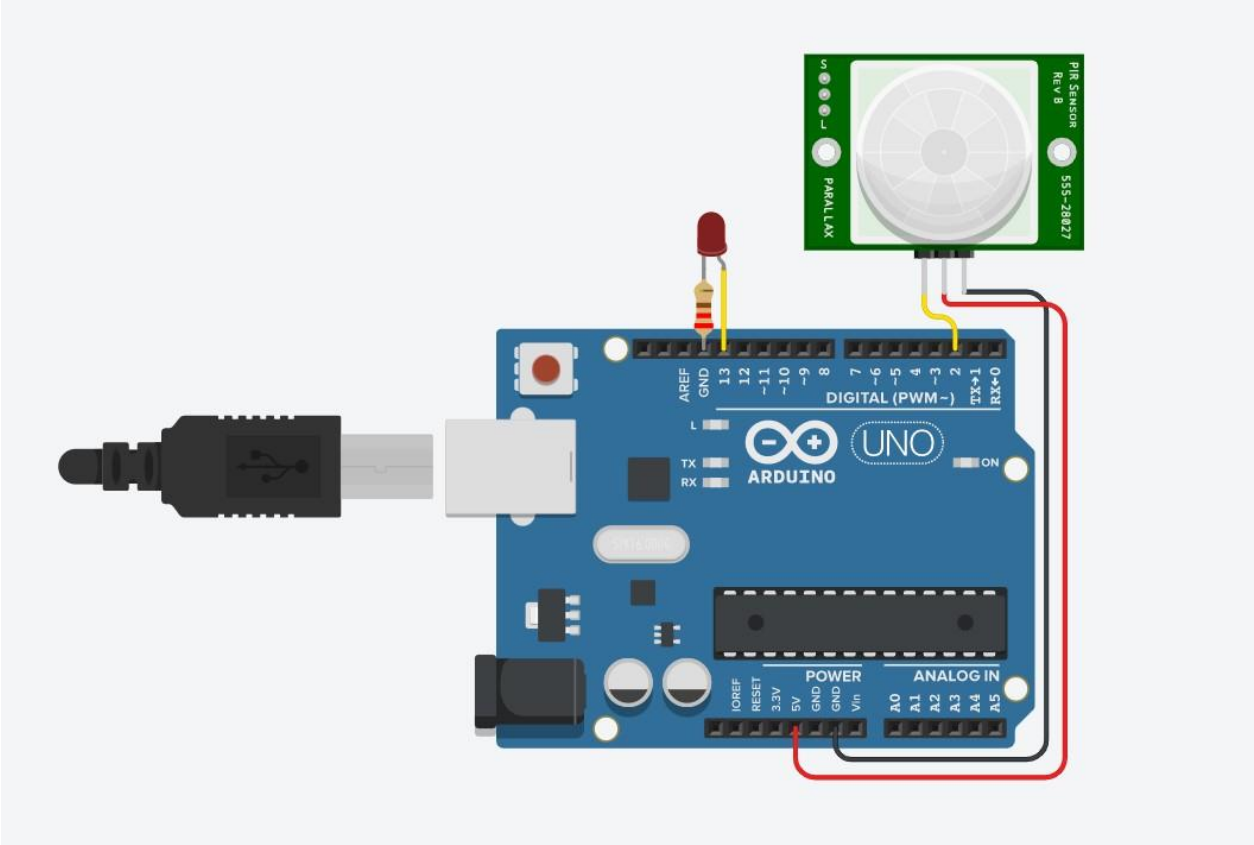



```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int red_led = 7;
int yellow_led = 8;
int green_led = 13;
int gas = A0;
int gas_data;

String gas_state;

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);

  pinMode(red_led,OUTPUT);
```

```
  pinMode(yellow_led,OUTPUT);
  pinMode(green_led,OUTPUT);
  pinMode(gas,INPUT);
}

void loop() {

  digitalWrite(red_led,LOW);
  digitalWrite(yellow_led,LOW);
  digitalWrite(green_led,LOW);
  gas_data = analogRead(gas);
  lcd.setCursor(00,00);
  lcd.print("Gas:");
  lcd.setCursor(5,00);
  lcd.print(gas_data);

  if(gas_data > 800){
            digitalWrite(red_led,HIGH);
    gas_state = "DANGER";

  }
  else if(gas_data > 700){
    digitalWrite(yellow_led,HIGH);
    gas_state = "WARNING";

  }else {
    digitalWrite(green_led,HIGH);
    gas_state = "SAFE";
  }
```

```
  lcd.setCursor(00,00);
  lcd.print("Gas:");
  lcd.setCursor(5,00);
  lcd.print(gas_data);
  lcd.setCursor(00,1);
  lcd.print(gas_state);




  Serial.println(gas_data);
  delay(100);
  lcd.clear();

}
```

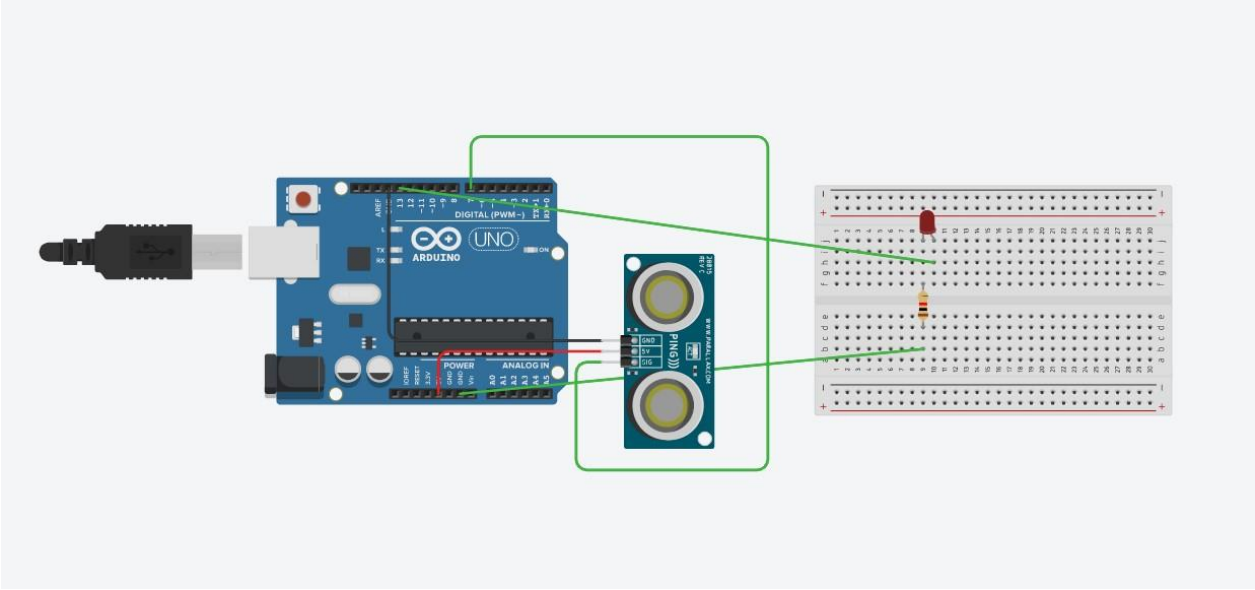
6. **PIR Sensor**




```
int buttonState = 0;


void setup()
{
  pinMode(2, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}


void loop()
{
  // read the state of the pushbutton
  buttonState = digitalRead(2);
```

```
  // check if pushbutton is pressed. if it is,
the
  // button state is HIGH
  if (buttonState == HIGH) {
    digitalWrite(LED_BUILTIN, HIGH);
  } else {
    digitalWrite(LED_BUILTIN, LOW);
  }
  delay(10); // Delay a little bit to improve
simulation performance
}
```

7. Ultrasonic Sensor




```
const int pingPin = 7;
const int ledPin = 13;

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

void loop() {

  long duration, cm;

  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
```

```
  digitalWrite(pingPin, LOW);



  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);



  cm =
microsecondsToCentimeters(duration);



  Serial.print("Distance: ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();



  if(cm < 100) {
    digitalWrite(ledPin, HIGH);
  }
  else {
    digitalWrite(ledPin, LOW);
  }

  delay(100);
}



long microsecondsToCentimeters(long
microseconds) {
```

```
  return microseconds / 29 / 2;
}
```