Project 1: Attach an LDR (light dependent resistor) to Arduino Uno and employ Arduino Sketch to program it for detecting ambient light levels in the room/environment. When the light falls below a designated threshold, activate a lamp accordingly

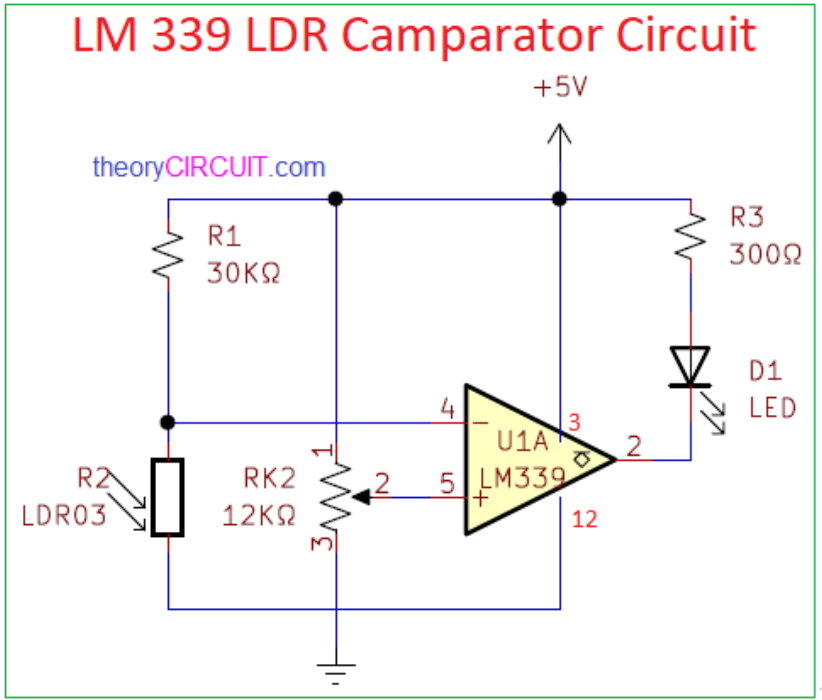
Know Your Sensor: Light Dependent Resistor (LDR)/Photoresistor

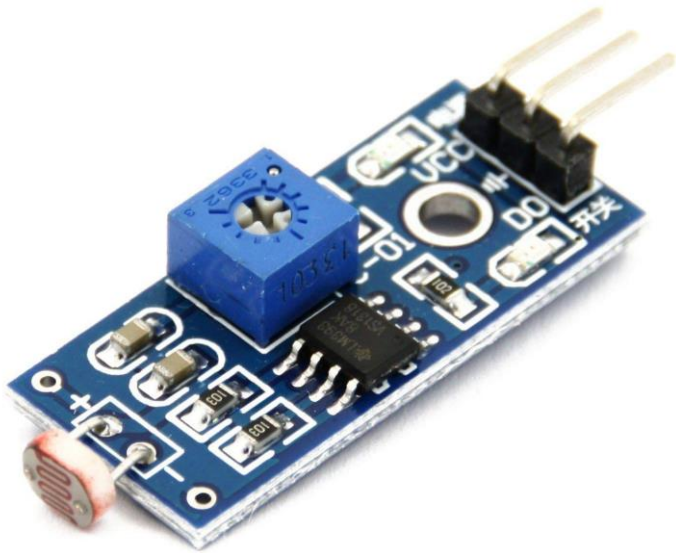
- Works on the principle of photoconductivity, where its electrical resistance changes based on the intensity of light falling on it.
- Made of semiconductor materials like cadmium sulfide (CdS) or cadmium selenide (CdSe). These materials exhibit photoconductivity, meaning their electrical conductivity increases with light exposure.
- When light (photons) strikes the LDR, the energy from the light excites electrons in the semiconductor material.

Project 1

Know Your Sensor: Light Dependent Resistor (LDR)/Photoresistor

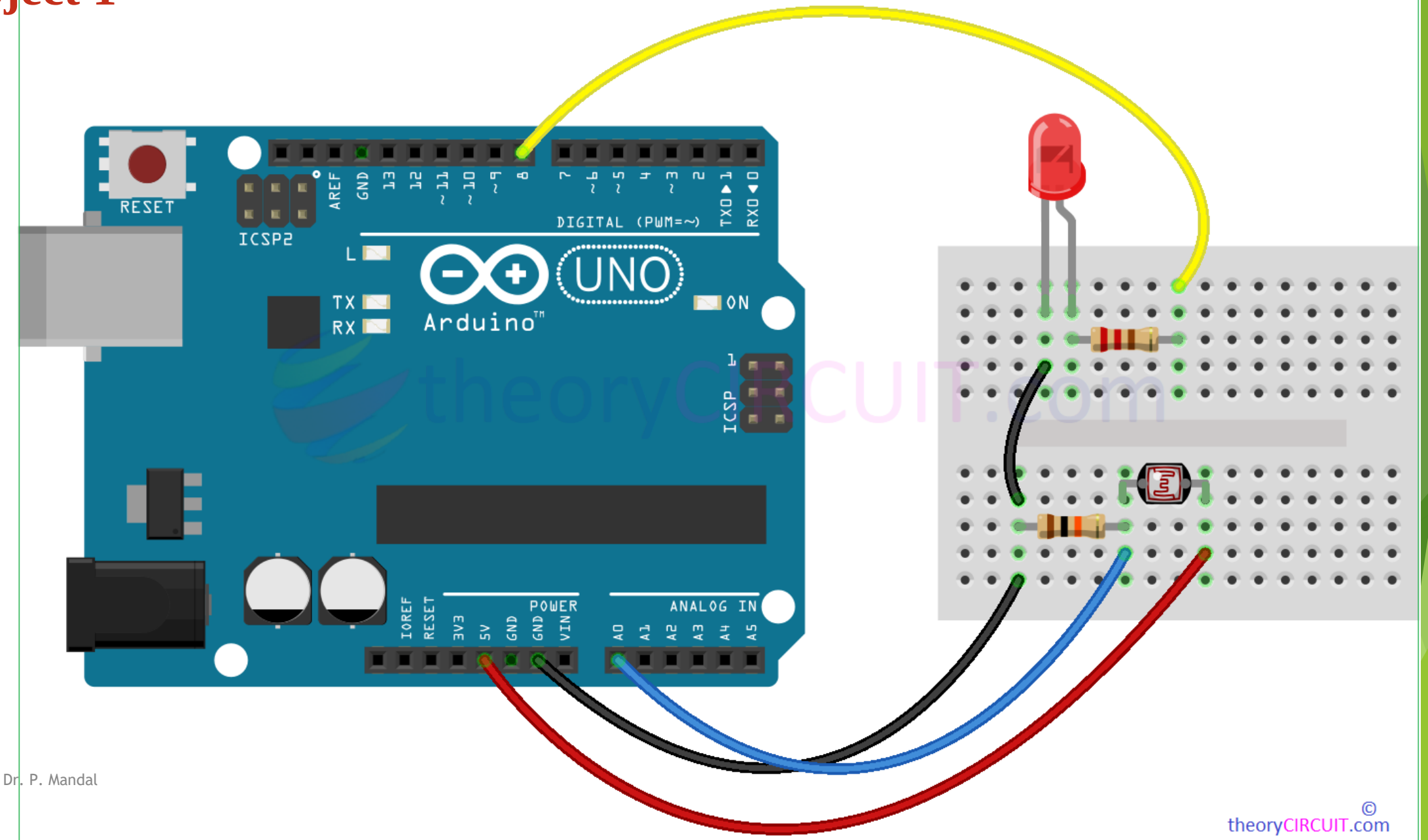
- These electrons jump from valence band to conduction band, creating free electrons and holes. As a result, the material's electrical conductivity increases, its resistance decreases.
- When more light falls on the LDR, more electrons are excited, significantly reducing its resistance. In the absence of light, fewer electrons are excited, and the resistance remains high.





Project 1

Arduino LDR LED Interface



Project 1:

```
int ldrpin = A0; //LDR pin designated to analog pin A0
int ldrval = 0; //value of LDR

void setup()
{
  Serial.begin(9600); //Initializes the serial monitor
}

void loop()
{
  ldrval = analogRead(ldrpin); //Reads the analog value of the LDR
  Serial.println(ldrval); //Displays the value at the serial monitor
  delay(2);
}
```


https://www.instructables.com/Arduino-and-a-LDR-Light-Dependent-Resistor/

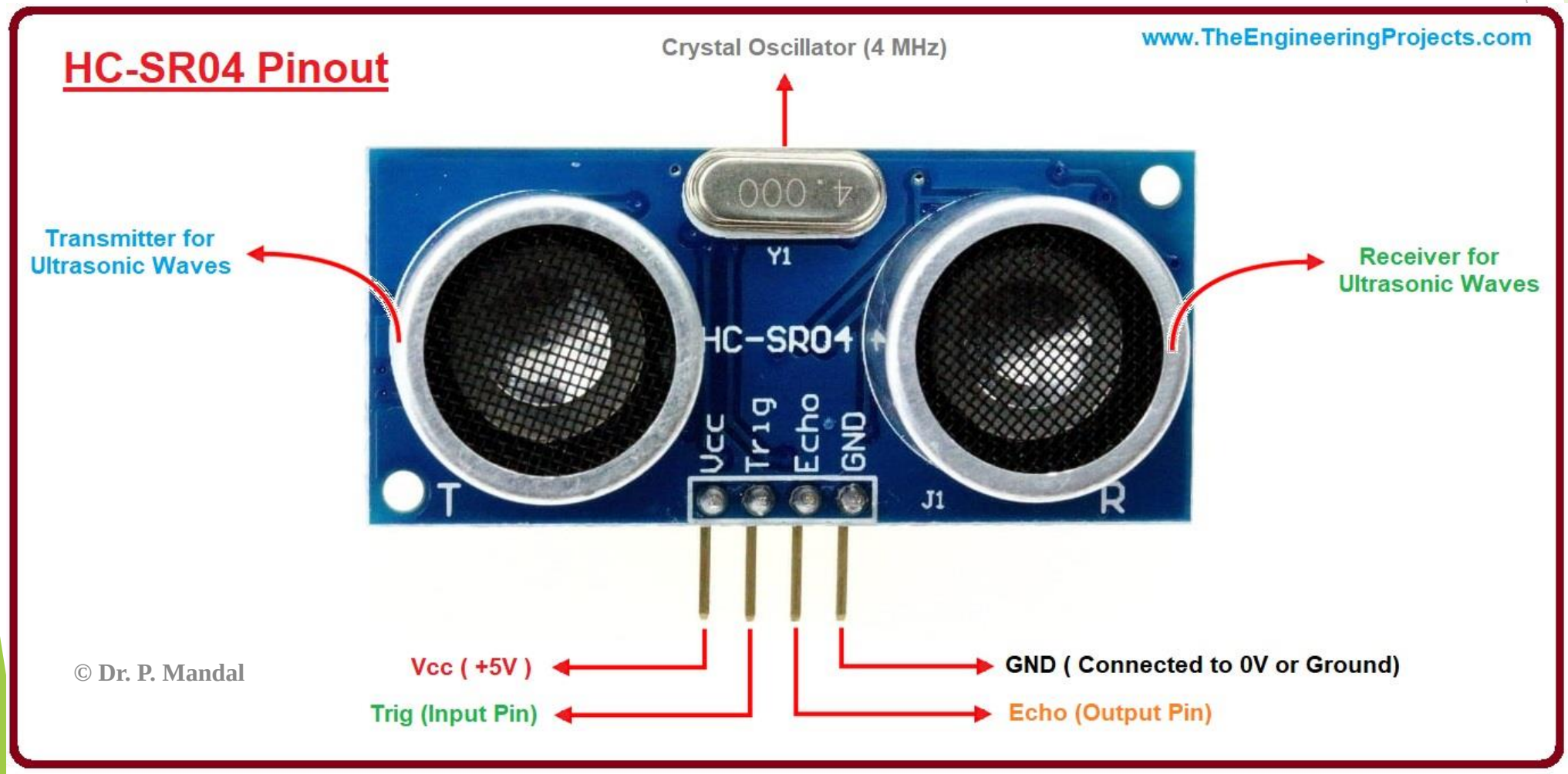
Project 1:

```
int ldrpin = A0;
int ldrvalue = 0;
int ledpin = 13;   // Digital pin number 13
void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);   // Output at the digital pin 13
  digitalWrite(13, LOW);   // Sets the output mode as low
}
void loop()
{
  ldrvalue = analogRead(ldrpin);
  Serial.println(ldrvalue);
  delay(2);

  if (ldrvalue > 300)     // If the LDR value is higher than 300
  {
    digitalWrite(ledpin, LOW); //Pin 13 is low, LED is off
  }
  else
  {
    digitalWrite(ledpin, HIGH); //Pin 13 is high, LED is on
  }
}
```

https://www.youtube.com/watch?v=9VYrGY142zQ&t=106s

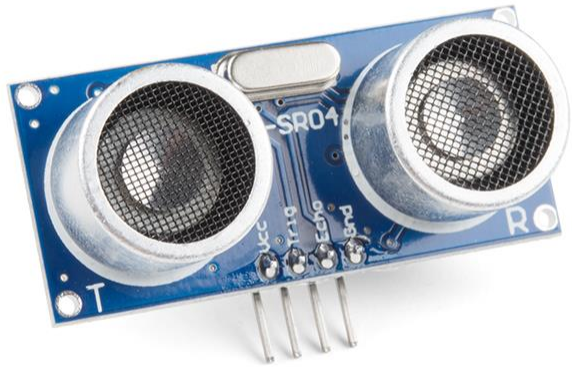
Project 3: Connect an ultrasonic distance sensor (HC-SR04) to Arduino Uno and utilize Arduino Sketch to program the measurement of object distance from the sensor and display it on the serial monitor.

Know Your Sensor: Ultrasonic Distance Sensor (HC-SR04)



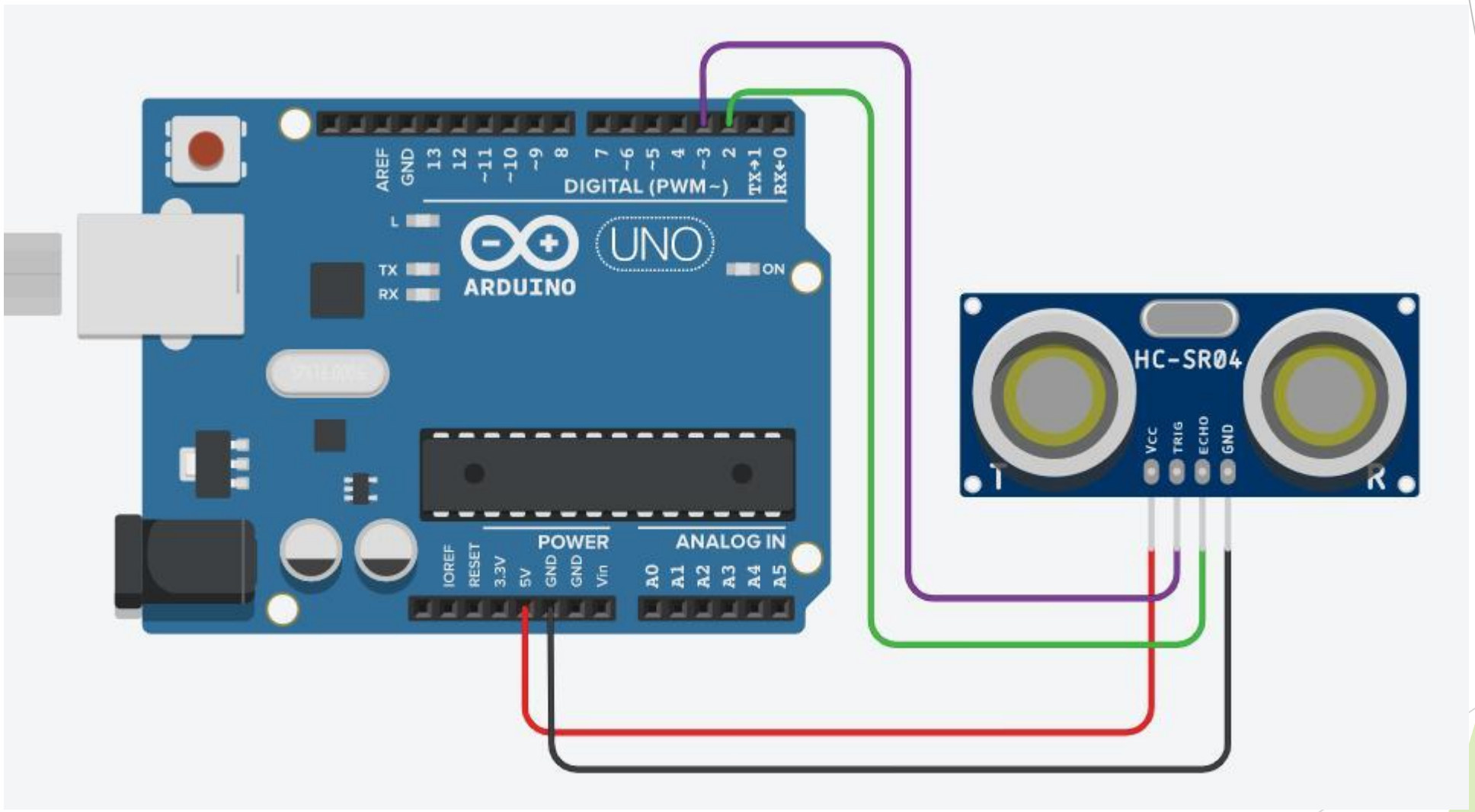
Project 3

Know Your Sensor: Ultrasonic Distance Sensor (HC-SR04)



- Microcontroller sends a trigger pulse of 10 µs to the sensor's Trigger pin which activates the sensor's transmitter to emit an ultrasonic pulse.
- The emitted ultrasonic waves travel through the air at the speed of sound (~340 m/s).
- When the waves encounter an object, they reflect back towards the sensor and picked up by the sensor's Echo pin.
- The time between the Trigger pulse and the Echo pin signal (high state duration) is measured by the microcontroller.
- $\text{Distance} = \frac{\text{Speed of Sound} \times \text{Time (in seconds)}}{2}$
- HC-SR04 can measure distances from 2 cm to 400 cm.

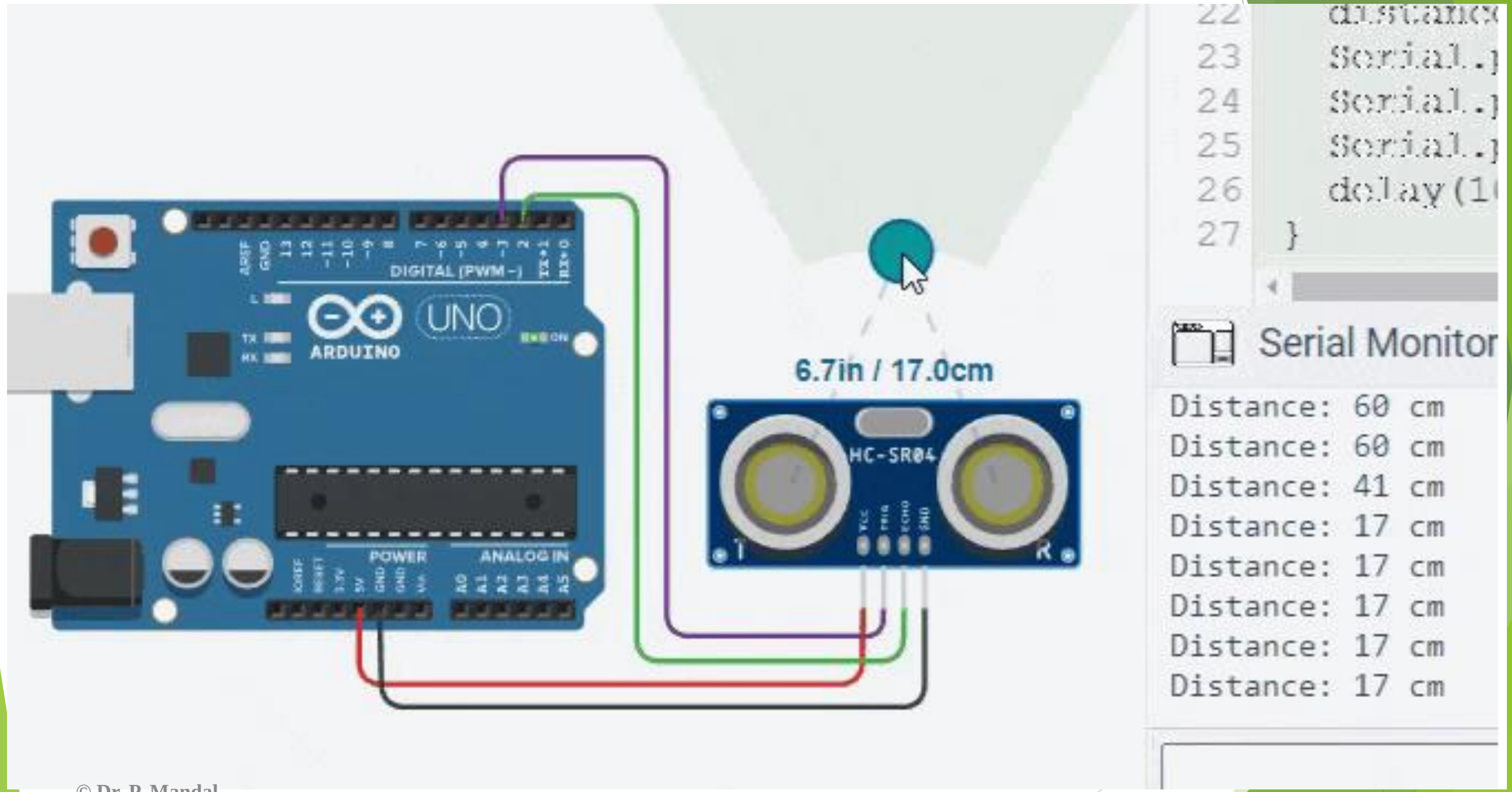
Project 3



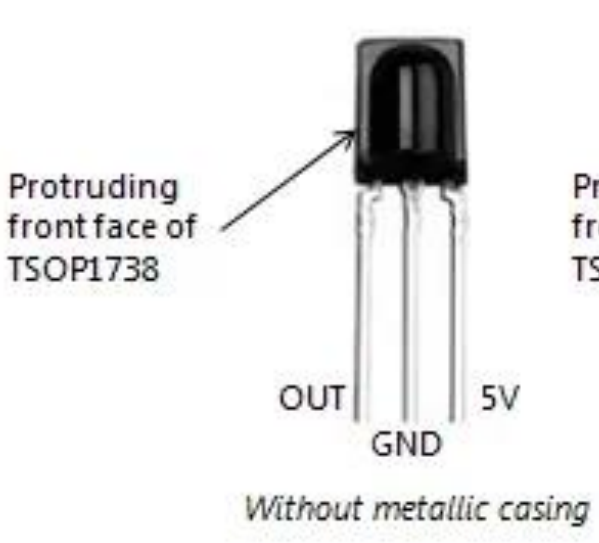
https://www.geeksforgeeks.org/distance-measurement-using-ultrasonic-sensor-and-arduino/

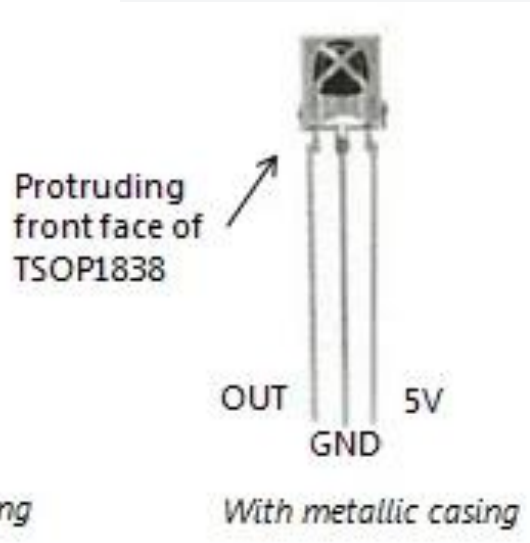
```
#define echoPin 2 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 3 // attach pin D3 Arduino to pin Trig of HC-SR04

long duration; // Variable to store time taken to the pulse to reach receiver
int distance; // Variable to store distance calculated using formula

void setup()
{
    pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
    pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
    Serial.begin(9600); // Serial Communication at 9600 of baudrate speed
    Serial.println("Distance measurement using Arduino Uno.");
    delay(500); // The text to be printed in serial monitor
}
```

```
void loop(){
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2); // wait for 2 µs to avoid collision in serial monitor
    digitalWrite(trigPin,HIGH); // turn on the Trigger to generate pulse
    delayMicroseconds(10); // trigger "ON" for 10 µs & generate pulse for 10 µs
    digitalWrite(trigPin,LOW); //turn off pulse trigger & stop pulse generation
    // If pulse reached the receiver echoPin become high then pulseIn() returns
       the time taken by the pulse to reach the receiver
    duration = pulseIn(echoPin, HIGH);
    distance = duration * 0.0340 / 2; // Expression for distance using time
    Serial.print("Distance: ");
    Serial.print(distance); // Print the output in serial monitor
    Serial.println("cm");
    delay(100);}
```


DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN
6.7in / 17.0cm
HC-SR04
VCC
TRIG
ECHO
GND
22
23
24
25
26
27
}
Serial Monitor
Distance: 60 cm
Distance: 60 cm
Distance: 41 cm
Distance: 17 cm
Distance: 17 cm
Distance: 17 cm
Distance: 17 cm
Distance: 17 cm

Project 4: Connect an IR sensor to Arduino Uno and utilize Arduino Sketch to detect Hex code generated from a TV remote for different button and print them on serial monitor





Project 4

Know Your Sensor: IR sensor (TSOP 1738/1838)

The **IR remote sensor 1738**, also known as the **TSOP1738**, is an **infrared (IR) receiver module** commonly used in remote control systems. It detects and demodulates **38 kHz** IR signals, which are commonly used in consumer electronics like TVs, DVD players, and home automation. TSOP1738 operates by detecting modulated IR signals (typically 38 kHz) from an IR remote and converting them into digital signals.

- ❑ The sensor has a photodiode that detects infrared light emitted from an IR remote control.
- ❑ It only responds to IR signals modulated at 38 kHz, rejecting continuous IR signals from other sources (e.g., sunlight, incandescent bulbs).
- ❑ The internal demodulator extracts the data signal from the modulated 38 kHz carrier wave.

Project 4

Know Your Sensor: IR sensor (TSOP 1738/1838) cotnd.

- The output is a clean digital signal representing the original pulse pattern of the remote control.
- The sensor includes automatic gain control (AGC) to filter out unwanted noise and disturbances.
- The received signal is amplified for better detection.
- The demodulated signal is sent to the OUT pin as a digital pulse train (high and low states).
- These pulses correspond to the encoded data sent by the remote.

Project 4

Project_04 | Arduino IDE 2.3.3

File Edit Sketch Tools Help

Arduino Uno

Project_04.ino

```ino
#include <IRremote.h>

// Pin connected to the IR receiver
const int RECV_PIN = 11;

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

    // Set up the IR receiver
    IrReceiver.begin(RECV_PIN, ENABLE_LED_FEEDBACK);
    Serial.println("IR Receiver is ready");
}

void loop() {
    // Check if a signal is received
    if (IrReceiver.decode()) {
        Serial.print("Received IR Code: ");
        Serial.println(IrReceiver.decodedIRData.decodedRawData, HEX); // Print the code in HEX format

        IrReceiver.resume(); // Ready to receive the next signal
    }
}
```

Output Serial Monitor

Message (Enter to send message to 'Arduino Uno' on 'COM7')

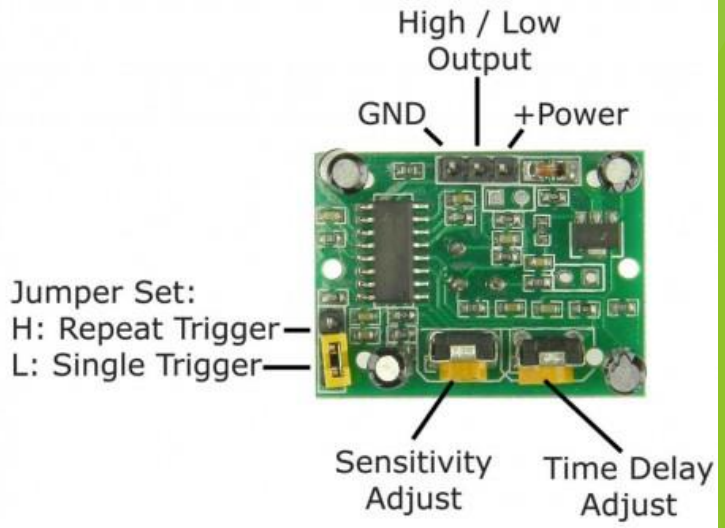
New Line 9600 baud

```
Received IR Code: EE11FB04
Received IR Code: EC13FB04
Received IR Code: 0
Received IR Code: E916FB04
Received IR Code: E619FB04
Received IR Code: E619FB04
Received IR Code: E916FB04
Received IR Code: EC13FB04
Received IR Code: F00F9B88
```

Project 5: Connect an HR-SC501 passive infrared (PIR) sensor to Arduino Uno and utilize Arduino Sketch to code for detecting movement in front of the PIR sensor. Illuminate an LED to indicate the detected movement.

Know Your Sensor: Ultrasonic Distance Sensor (HC-SR04)

The HR-SC501 Passive Infrared (PIR) sensor detects motion by sensing infrared (IR) radiation changes in its field of view. Its operation is based on the detection of infrared radiation emitted by warm objects, such as humans or animals.



Project 5:

Basic Components

Pyroelectric Sensor: Detects changes in IR radiation. It has two sensing elements to differentiate motion from ambient temperature. (pyroelectric effect refers to the ability of certain materials like ceramics or crystals, to generate a small electric charge in response to temperature changes.)

Fresnel Lens: Focuses IR radiation onto the pyroelectric sensor to improve sensitivity and expand the detection range.

Processing Circuit: Processes the sensor's signal and outputs a HIGH or LOW signal based on detected motion.

Control Pins: Typically, the HR-SC501 has adjustable sensitivity and delay time using onboard potentiometers.

Project 5:

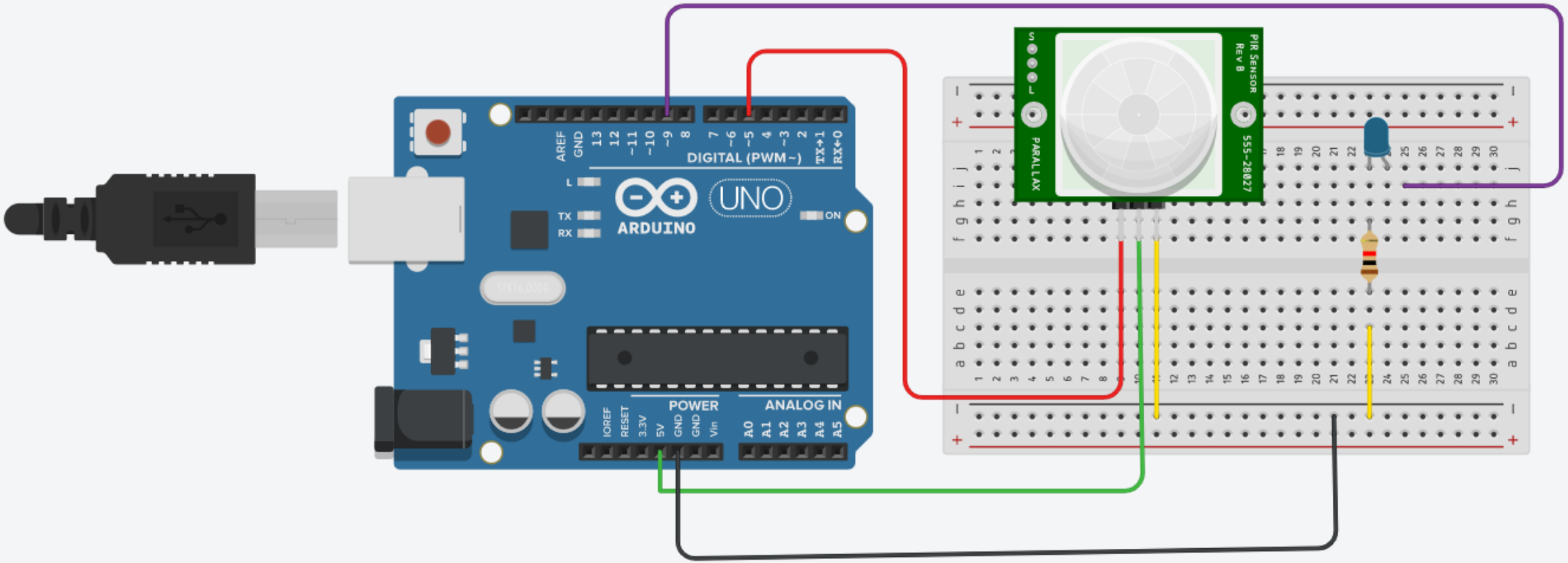
Working Principle

Background IR Detection: The PIR sensor constantly monitors the IR levels in its environment. Normally, it detects a stable amount of IR radiation from stationary objects like walls, furniture, or ambient heat.

Motion Detection: When a warm object (like a person or animal) moves across its field of view, the IR radiation changes rapidly as it moves from one sensing element to another.

- The dual sensing elements inside the PIR sensor detect this change in IR radiation. If the levels differ between the two sensing zones, the sensor triggers an output.
- The onboard processing circuit generates a HIGH output signal for a preset duration (delay).
- After the delay time expires, the output returns to LOW if no further motion is detected.

Project 5:

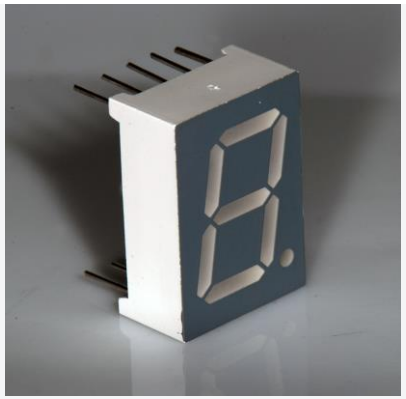


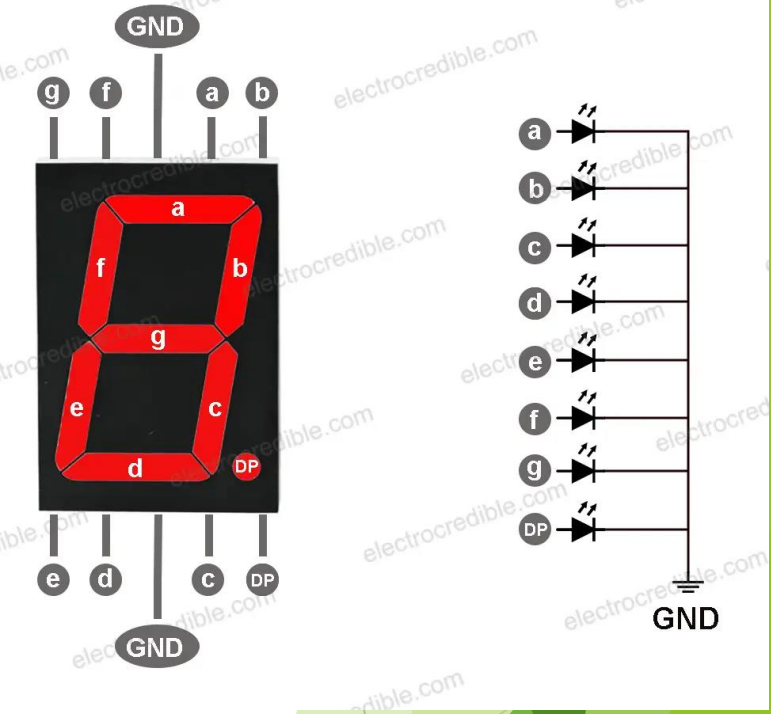
https://projecthub.arduino.cc/electronicsfan123/interfacing-arduino-uno-with-pir-motion-sensor-593b6b

Project 5:

```
// Define pins
const int pirPin = 2; // PIR sensor output pin
const int ledPin = 13; // LED pin

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

void loop() {
  int motionDetected = digitalRead(pirPin); // Read PIR sensor output

  if (motionDetected == HIGH) {
    // Motion detected
    digitalWrite(ledPin, HIGH); // Turn on LED
    Serial.println("Motion detected!");
  } else {
    // No motion
    digitalWrite(ledPin, LOW); // Turn off LED
  }

  delay(100); // Short delay to prevent excessive serial output
}
```

Project 7: Attach a Seven Segment Display to Arduino Uno and utilize Arduino Sketch to exhibit a counter that increments from 0 to 9, pausing for 2 seconds between each increment...





Common Cathode

Project 7:

```
// Define the Arduino pins connected to the segments
const int segmentPins[] = {2, 3, 4, 5, 6, 7, 8}; // Pins for a, b, c, d,
e, f, g

// Define the binary patterns for displaying digits 0-9 on a common
cathode display
const byte digits[] = {
  0b00111111, // 0
  0b00000110, // 1
  0b01011011, // 2
  0b01001111, // 3
  0b01100110, // 4
  0b01101101, // 5
  0b01111101, // 6
  0b00000111, // 7
  0b01111111, // 8
  0b01101111  // 9
};
```

Project 7:

```
void setup() {
  // Set all segment pins as OUTPUT
  for (int i = 0; i < 7; i++) {
    pinMode(segmentPins[i], OUTPUT);
    digitalWrite(segmentPins[i], LOW); // Initialize segments as
OFF
  }
}

void displayDigit(byte digit) {
  // Light up segments based on the binary pattern for the digit
  for (int i = 0; i < 7; i++) {
    bool state = (digit >> i) & 0x01; // Extract the i-th bit of
the binary pattern
    digitalWrite(segmentPins[i], state);
  }
}
```

Project 7:

```
void loop() {
  // Display digits 0 to 9 one by one
  for (int i = 0; i < 10; i++) {
    displayDigit(digits[i]); // Display the current digit
    delay(2000);             // Wait 2 second before showing the
next digit
  }
}
```

Modifications for Common Anode Displays:

If you're using a common anode display, invert the logic in `displayDigit()` like this:

```
digitalWrite(segmentPins[i], !state); // Invert the state
```

Project 11: Connect a temperature and humidity module such as DHT-11/22, LM35DZ, or DS18B20/LM75 to Arduino Uno and utilize Arduino Sketch to code for reading temperature and humidity. Display the data either on the serial monitor or as a plotted graph showcasing the captured readings from the sensor over a specified duration.

Know Your Sensor: DHT11 sensor module

- Digital humidity and temperature sensor that provides calibrated output via a single-wire communication protocol.
- Humidity Sensor: A resistive humidity sensing component absorbs moisture from the air, changing its resistance accordingly.
- Temperature Sensor: A thermistor (NTC) measures temperature by changing its resistance based on the ambient temperature.
- Microcontroller: An internal IC processes the sensor data and transmits it digitally.

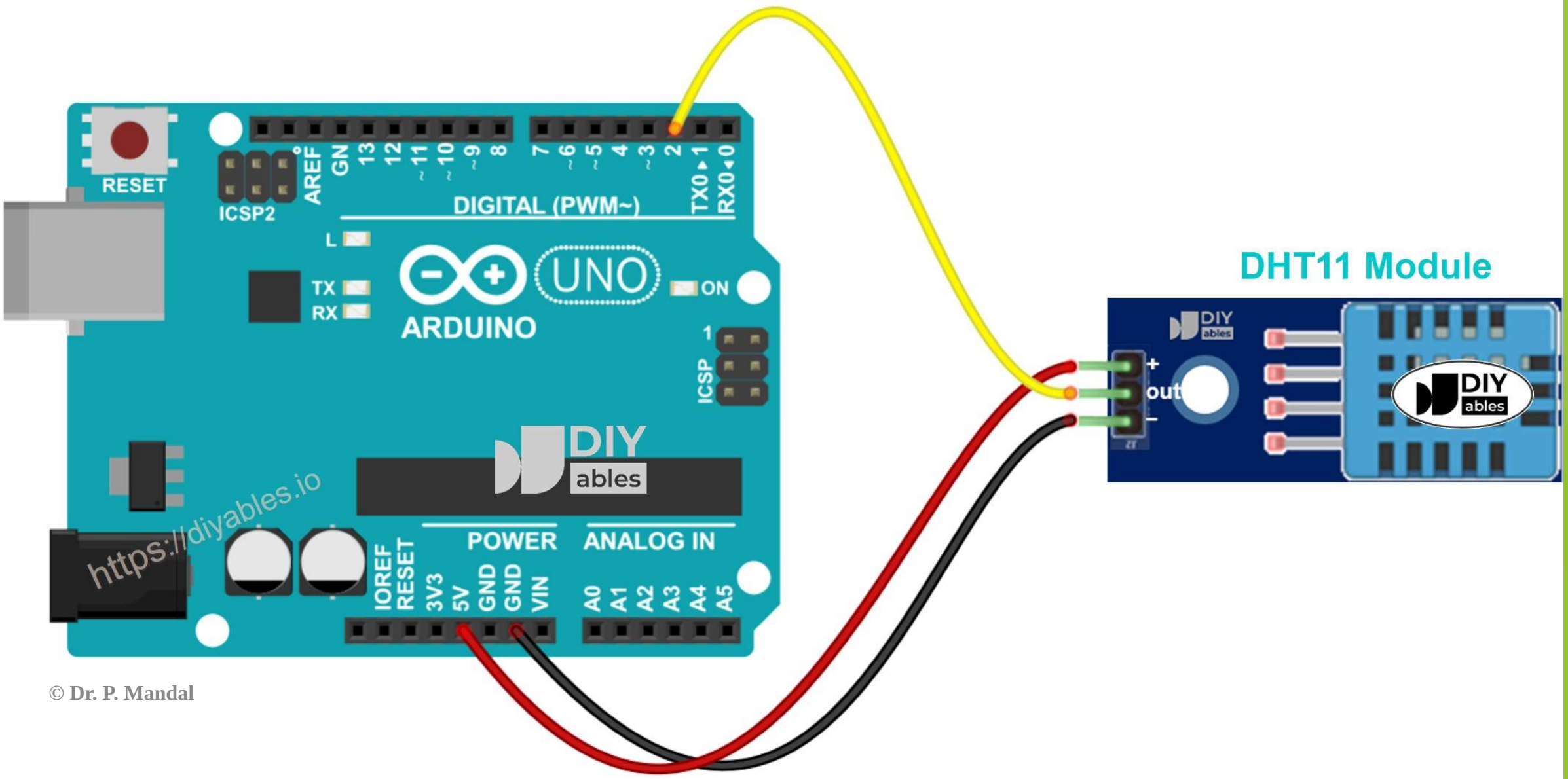
Project 11

Know Your Sensor: DHT11 sensor module (contd.)

- Humidity Measurement: The sensor has a moisture-sensitive resistive element. As humidity increases, the conductivity of this element changes. The internal microcontroller reads these changes and converts them into digital humidity values.
- Temperature Measurement: A Negative Temperature Coefficient (NTC) thermistor measures temperature. As temperature increases, the resistance of the thermistor decreases, and the microcontroller converts this data into digital form.
- Data Transmission: The DHT11 communicates using a single-wire digital protocol. The microcontroller in the sensor module processes the temperature and humidity readings and sends them as a 40-bit serial data signal to the external microcontroller or microprocessor.

WIRING DIAGRAM TO ARDUINO

Project 11



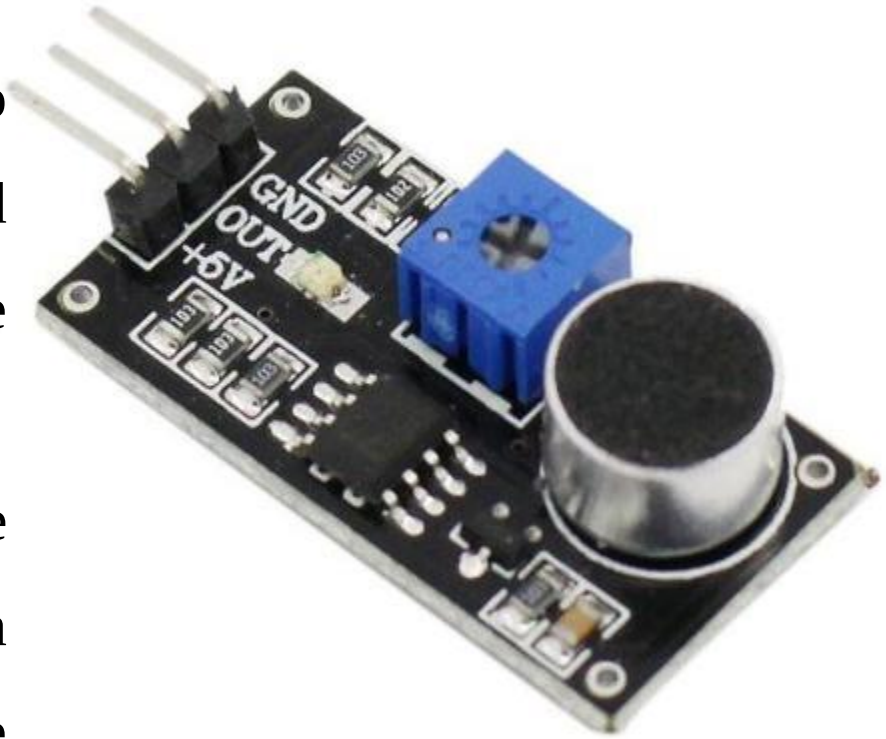
Project 11

Project_11.ino

```
#include <DHT.h>
//DHT sensor library by Adafruit
// Define DHT sensor type and pin
#define DHTPIN 2        // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11

// Initialize DHT sensor
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);   // Start the Serial communication
  Serial.println("DHT11 Temperature and Humidity Sensor");
  dht.begin();          // Initialize the DHT sensor
}

void loop() {
  // Wait a few seconds between readings
  delay(2000);

  // Reading temperature and humidity
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  // Check if any reads failed
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Print values to Serial Monitor
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");
}
```

Project 12: Attach an LM393 sound sensor to Arduino Uno and employ Arduino Sketch to program the detection of sound levels surpassing a set threshold, adjustable via the sensor's potentiometer. Within the sketch, illuminate an LED when sound exceeds the threshold, and indicate on the serial monitor whether a new sound has been detected or if the previous sound has ceased.

Know Your Sensor: LM 393 sound sensor module

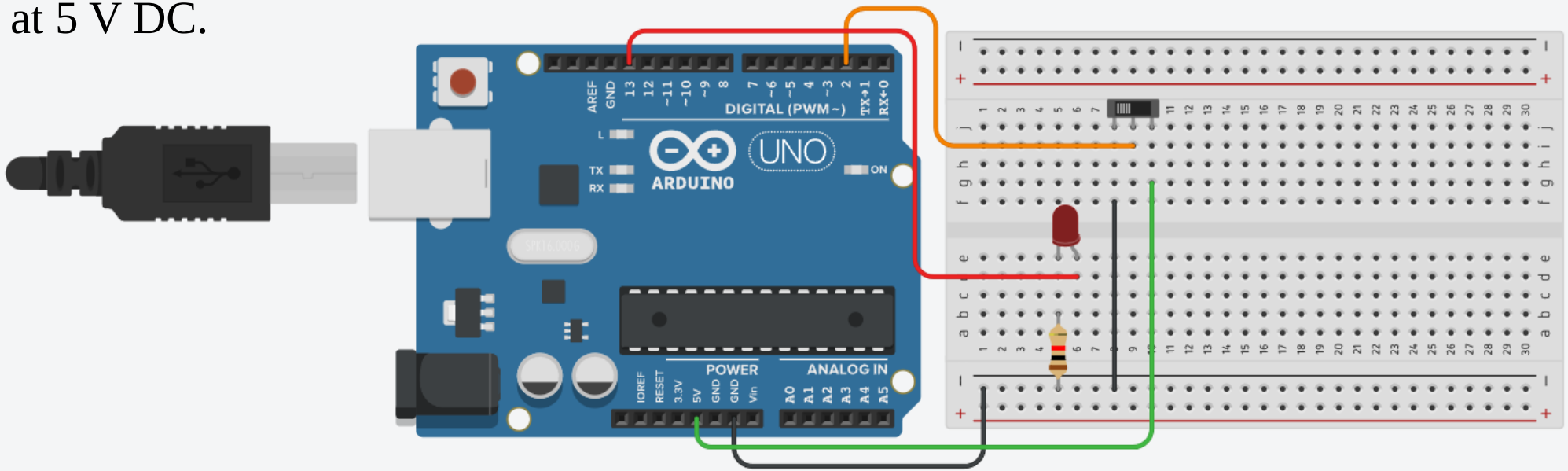


- Microphone: Captures sound waves and converts them into small electrical signals corresponding to the amplitude and frequency of the sound. The weak electrical signal from the microphone is amplified to make it usable.
- LM393 Comparator: Compares the voltage generated by the microphone with a reference voltage set by a potentiometer on the module. If the microphone's voltage exceeds the reference voltage, the comparator outputs a digital HIGH signal; otherwise, it outputs LOW.

Project 12

Know Your Sensor: LM 393 sound sensor module

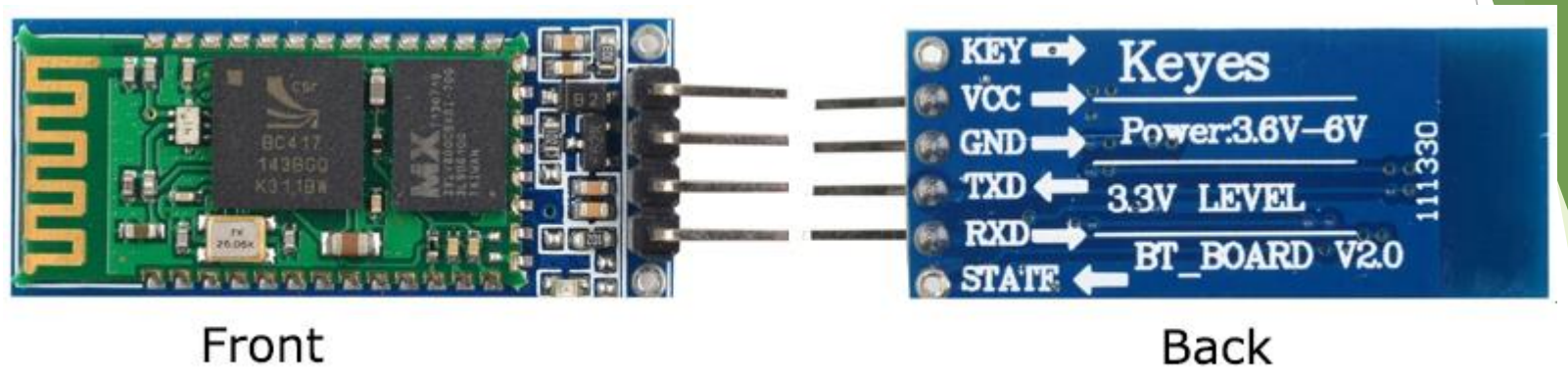
- Potentiometer: Adjusts the reference voltage, effectively setting the threshold for sound detection.
- Output Pin (Digital): Outputs a HIGH or LOW signal depending on whether the sound exceeds the threshold. HIGH output when sound intensity exceeds the threshold set by the potentiometer. LOW output when sound intensity is below the threshold.
- Operates at 5 V DC.



Project 12

```
#define SOUND_SENSOR_PIN 2  // LM393 OUT pin connected to digital pin 2
#define LED_PIN 13          // LED connected to digital pin 13

bool soundDetected = false; // Variable to track the current sound state

void setup() {
  pinMode(SOUND_SENSOR_PIN, INPUT); // Set sound sensor as input
  pinMode(LED_PIN, OUTPUT);         // Set LED pin as output
  Serial.begin(9600);               // Start serial communication
  Serial.println("Sound detection system initialized.");
}

void loop() {
  int soundState = digitalRead(SOUND_SENSOR_PIN); // Read sensor output (HIGH or LOW)

  if (soundState == HIGH && !soundDetected) {    // Sound detected for the first time
    soundDetected = true;
    digitalWrite(LED_PIN, HIGH); // Turn on LED
    Serial.println("New sound detected!");
  } else if (soundState == LOW && soundDetected) {    // Sound has ceased
    soundDetected = false;
    digitalWrite(LED_PIN, LOW); // Turn off LED
    Serial.println("Sound has ceased.");
  }
```

Project 15: Use of Bluetooth module HC-05 and operation through smartphone or voice.



Know Your Sensor: Bluetooth Module HC-05/06

This Bluetooth module can easily achieve serial wireless data transmission. Its operating frequency is among the most popular 2.4GHz ISM frequency band (i.e. Industrial, Scientific and Medical). It adopts Bluetooth 2.0+EDR standard. In Bluetooth 2.0, signal transmit time of different devices stands at a 0.5 seconds interval so that the workload of Bluetooth chip can be reduced substantially and more sleeping time can be saved for Bluetooth. This module is set with serial interface, which is easy to use and simplifies the overall design/development cycle.

Project 15

Pinout:

1. VCC: Power supply (3.6V–6V)

2. GND: Ground

3. TXD: Transmit data

4. RXD: Receive data

Default Name: HC-06

Default PIN Code: 1234 (used for pairing)

Smartphone App: Arduino Bluetooth Controller (or similar)

Arduino Bluetooth Controller

Giristudio

Contains ads

Project 15

Project_15.ino

```
#include <SoftwareSerial.h>
String value;
int TxD = 7;
int RxD = 6;
SoftwareSerial bluetooth(TxD, RxD);

void setup() {
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
Serial.begin(9600); // start serial communication at 9600bps
bluetooth.begin(9600);
}
void loop() {
Serial.println(value);
if (bluetooth.available())
{
value = bluetooth.readString();
if (value == "all LED turn on"){
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
}
if (value == "all LED turn off"){
digitalWrite(2, LOW);
digitalWrite(3, LOW);
}
if (value == "turn on Red LED"){
digitalWrite(2, HIGH);
}
if (value == "turn on green LED"){
digitalWrite(3, HIGH);
}
if (value == "turn off red LED"){
digitalWrite(2, LOW);
}
if (value == "turn off green LED"){
digitalWrite(3, LOW);
}
}
}
```