

**Project 9 & 10 & 11**

**Distance Measurement**

**&**

**Light Detection and Light Intensity Measurement**

**&**

**Obstacle Detection Using IR Sensor**

## Objective

Distance Measurement using Arduino and Ultra sonic Sensor.

## Required Equipment's

Arduino Uno, Ultra Sonic HC-SR04, 4 jumper Cables.

## What is a HC-SR04 Ultrasonic Distance Sensor?

The HC-SR04 Ultrasonic sensor is an inexpensive device that is very useful for robotics and test equipment projects. It consists of two ultrasonic transducers. The one acts as a transmitter which converts electrical signal into 40 **KHz** ultrasonic sound pulses. The receiver listens for the transmitted pulses. If it receives them it produces an output pulse whose width can be used to determine the distance the pulse travelled. It can be hooked directly to an Arduino or other microcontroller and it operates on 5 volts. This ultrasonic distance sensor is capable of measuring distances between 2 **cm** to 400 **cm**.

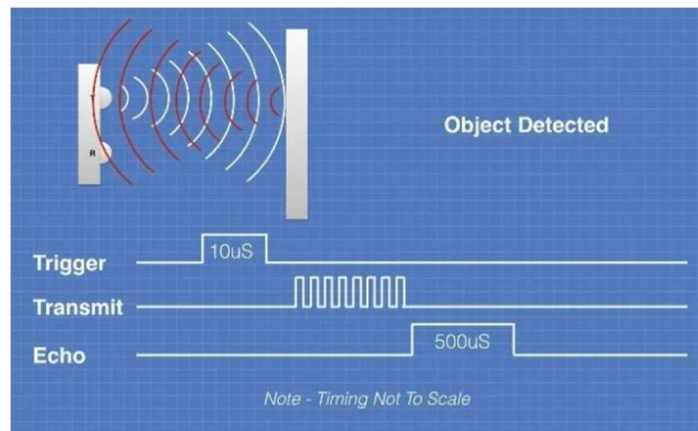
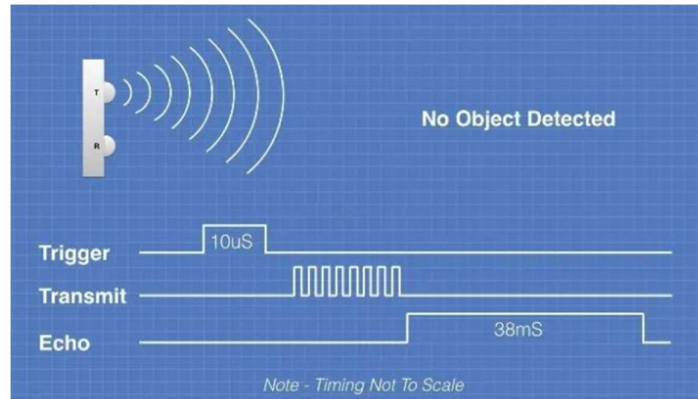


## How the HC-SR04 Ultrasonic Distance Sensor work's?

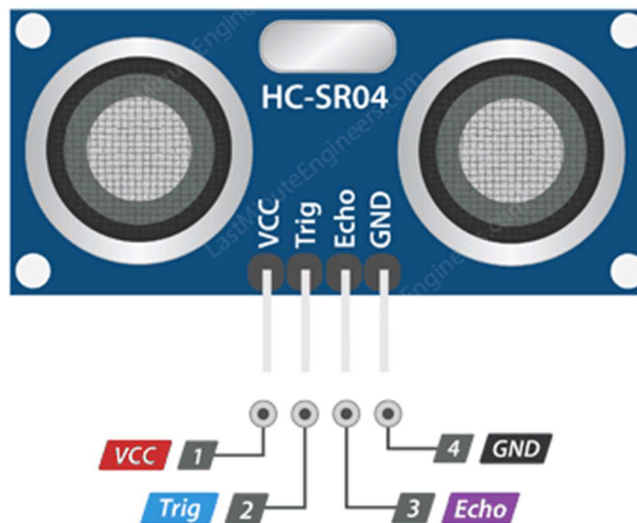
The device operates as follows:

- A 5 volt pulse of at least 10 **us** (10 microseconds) in duration is applied to the Trigger pin.
- The HC-SR04 responds by transmitting a burst of eight pulses at 40 **KHz**. This 8-pulse pattern makes the “ultrasonic signature” from the device unique, allowing the receiver to discriminate between the transmitted pattern and the ultrasonic background noise.
- The eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the Echo pin goes high to start forming the beginning of the echo-back signal.
- If the pulse is NOT reflected back then the Echo signal will timeout after 38 **ms** (38 milliseconds) and return low. This produces a 38 **ms** pulse that indicates no obstruction within the range of the sensor.
- If the pulse is reflected back, the Echo pin goes low when the signal is received. This produces a pulse whose width varies between 150 **us** to 25 **ms**, depending upon the time it took for the signal to be received.

- The width of the received pulse is used to calculate the distance to the reflected object. Remember that the pulse indicates the time it took for the signal to be sent out and reflected back so to get the distance you'll need to divide your result in half.



### HC-SR04 Ultrasonic Sensor Pinout



**VCC** is the power supply for HC-SR04 Ultrasonic distance sensor which we connect the 5V pin on the Arduino.

**Trig (Trigger)** pin is used to trigger the ultrasonic sound pulses.

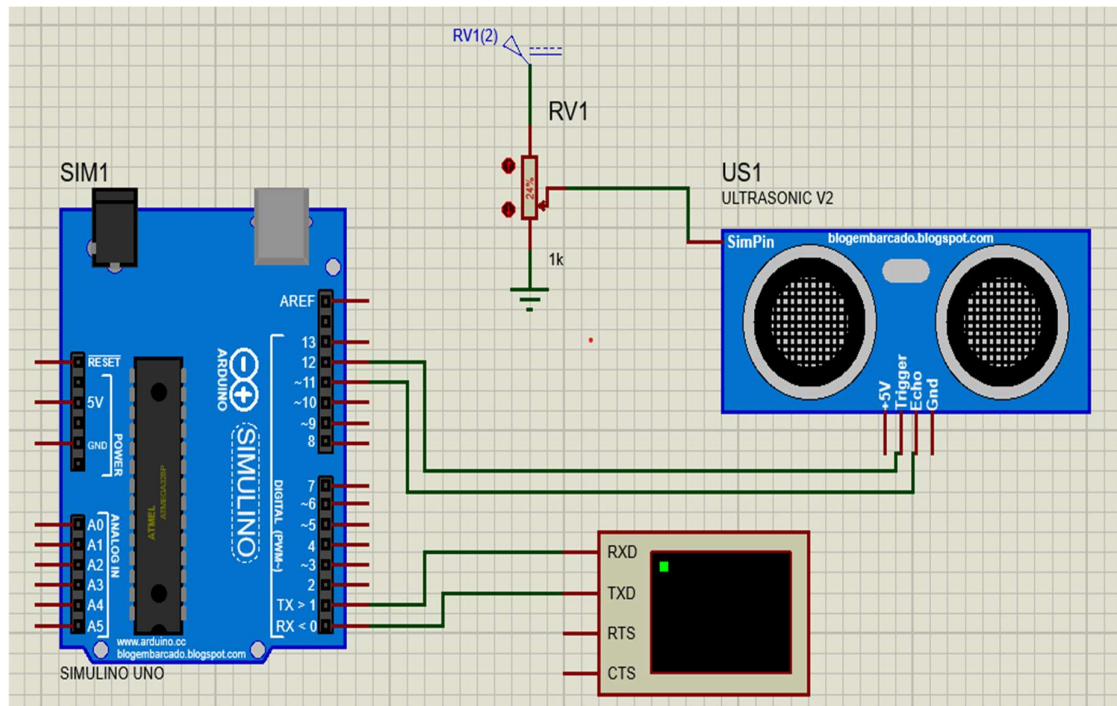
**Echo** pin produces a pulse when the reflected signal is received. The length of the pulse is proportional to the time it took for the transmitted signal to be detected.

**GND** should be connected to the ground of Arduino.

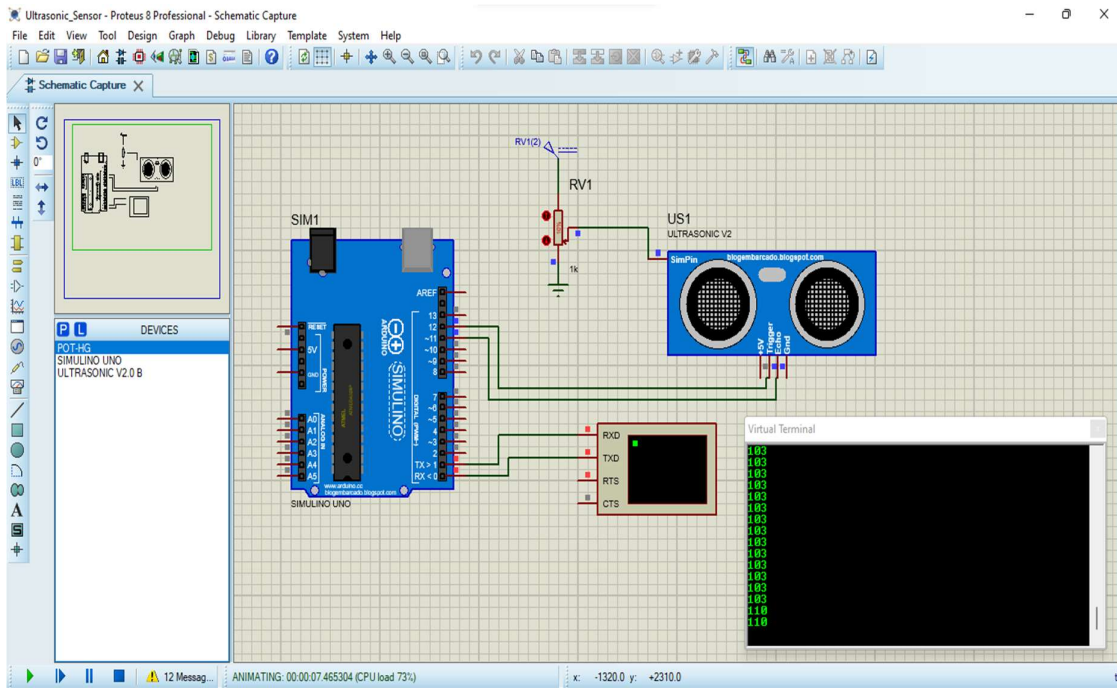
## Simulation of Ultrasonic Sensor on Proteus

First install the Ultrasonic sensor Libraries the same way as the installation of the Arduino libraries on Proteus.

Then open Proteus and open the “**Ultrasonic\_Sensor.pdsprj**” from the **Ultrasonic Sensor** folder.



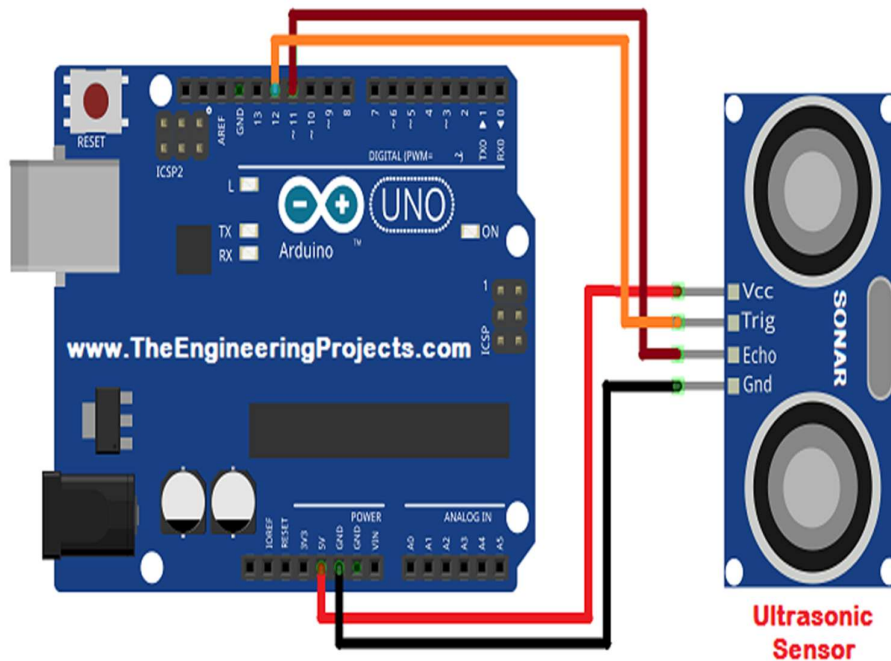
After that, double click on ultrasonic sensor and in the Program File upload the “**UltraSonicTEP.HEX**”. Then using the same steps upload the “**Ultrasonic\_Sensor.ino.hex**” file to Arduino and start the simulation.



Click on **Debug > Virtual Terminal** to open the Terminal output and use the Test Pin to play with Distance.

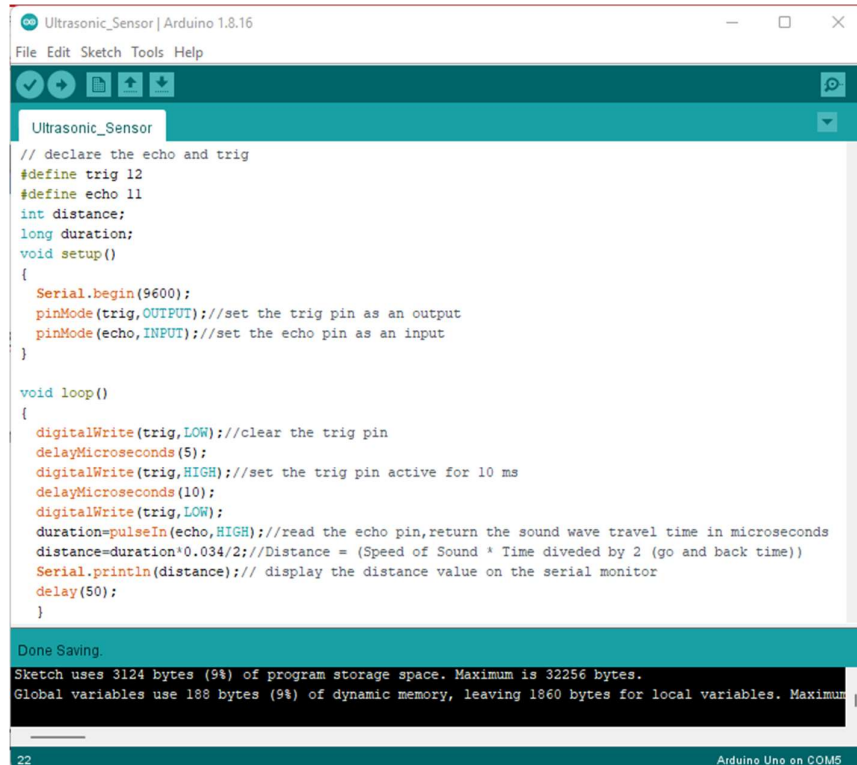
## Implementation using Arduino

Attach the Ultrasonic Sensor to the Arduino Board as follow:



## Arduino Code

From the **Ultrasonic\_Sensor** folder open “**Ultrasonic\_Sensor.ino**” file.

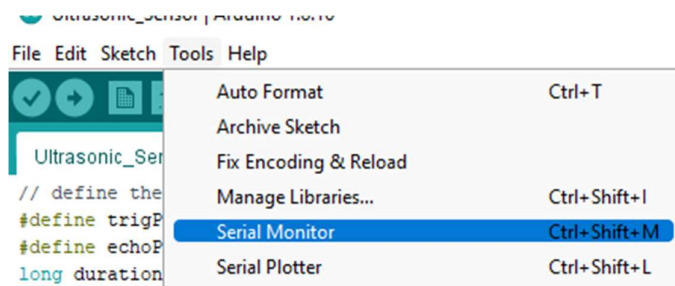


```
Ultrasonic_Sensor | Arduino 1.8.16
File Edit Sketch Tools Help
Ultrasonic_Sensor
// declare the echo and trig
#define trig 12
#define echo 11
int distance;
long duration;
void setup()
{
  Serial.begin(9600);
  pinMode(trig,OUTPUT);//set the trig pin as an output
  pinMode(echo,INPUT);//set the echo pin as an input
}

void loop()
{
  digitalWrite(trig,LOW);//clear the trig pin
  delayMicroseconds(5);
  digitalWrite(trig,HIGH);//set the trig pin active for 10 ms
  delayMicroseconds(10);
  digitalWrite(trig,LOW);
  duration=pulseIn(echo,HIGH);//read the echo pin,return the sound wave travel time in microseconds
  distance=duration*0.034/2;//Distance = (Speed of Sound * Time divided by 2 (go and back time))
  Serial.println(distance);// display the distance value on the serial monitor
  delay(50);
}

Done Saving.
Sketch uses 3124 bytes (9%) of program storage space. Maximum is 32256 bytes.
Global variables use 188 bytes (9%) of dynamic memory, leaving 1860 bytes for local variables. Maximum
22 Arduino Uno on COM5
```

Compile the code then send it to the Arduino board. Click on the **Tools > Serial Monitor** to see the sensor values.



# PROJECT 10: Light Detection and Light Intensity Measurement

## Objective

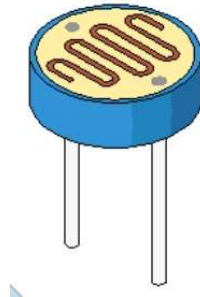
Detect and Measure Light Intensity using Arduino and LDR

## Required Equipment's

Arduino Uno, LDR, 1Kohm resistor, 3 jumper Cables.

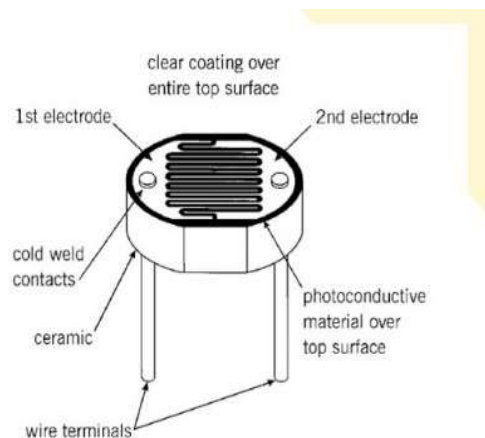
## What is an LDR?

LDR (**Light Dependent Resistor**) also known as a photoresistor, photoconductor or photocell, is a resistor whose resistance increases or decreases depending on the amount of light intensity. LDRs are a very useful tool in a light/dark circuits. A LDRs can have a variety of resistance and functions. Light dependent resistors are a vital component in any electric circuit which is to be turned on and off automatically according to the level of ambient light - for example, solar powered garden lights, and night security lighting.



## How LDRs Work?

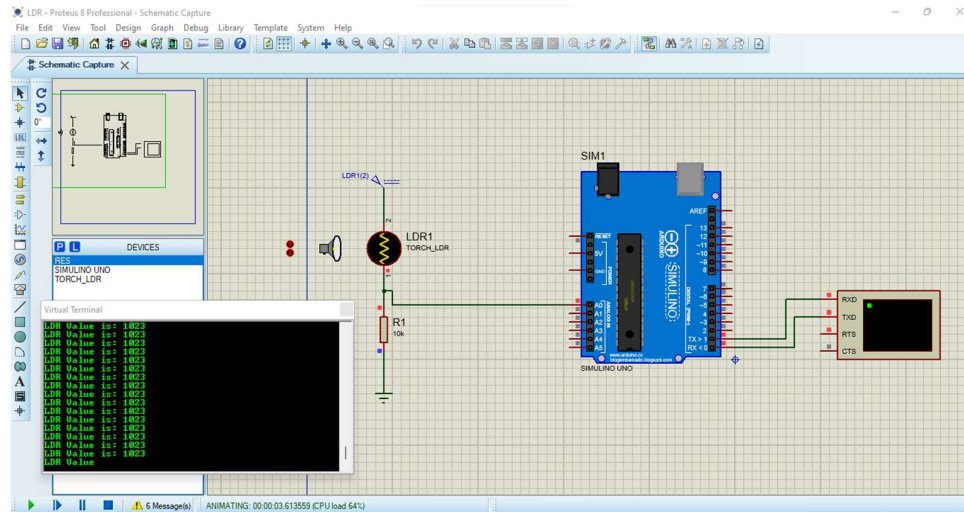
The way an LDR works is that they are made of many semi-conductive materials with high resistance. The reason they have a high resistance is that there are very few electrons that are free and able to move because they are held in a crystal lattice and are unable to move. When light falls on the semi-conductive material it absorbs the light photons and the energy is transferred to the electrons, which allow them to break free from the crystal lattice and conduct electricity and lower the resistance of the LDR.





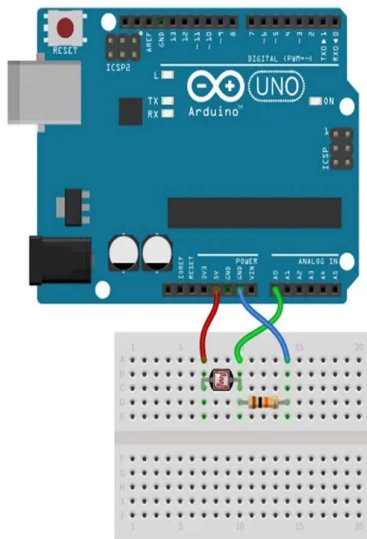
## Simulation of an LDR using Proteus

Navigate to the LDR folder and open the “LDR.pdsprj” Project, double click on Arduino then select the Program File field to upload the “LDR.ino.standard.hex” and start the simulation process. Open the virtual Terminal to see the output data.



## Implementation using Arduino

Achieve the following circuit:



## Arduino Code

Compile the “LDR.INO” File then upload it to the Arduino Board.



```
LDR $  
// Declare the A0 pin variable  
const int LDR = A0;  
int input_val = 0;  
  
void setup()  
{  
  Serial.begin(9600); // start the serial monitor  
}  
  
void loop()  
{  
  input_val = analogRead(LDR); // read data from A0 pin  
  Serial.println(input_val); // print data into serial  
  delay(50);  
}
```

14 Arduino Uno on COM5

# PROJECT 11: Obstacle Detection Using IR Sensor

## Objective

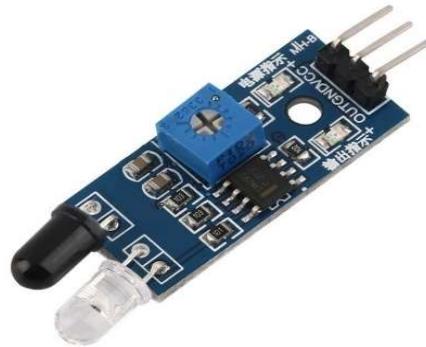
Detect Obstacle using Arduino and IR Sensor.

## Required Equipment's

Arduino Uno, IR Sensor, 3 jumper Cables.

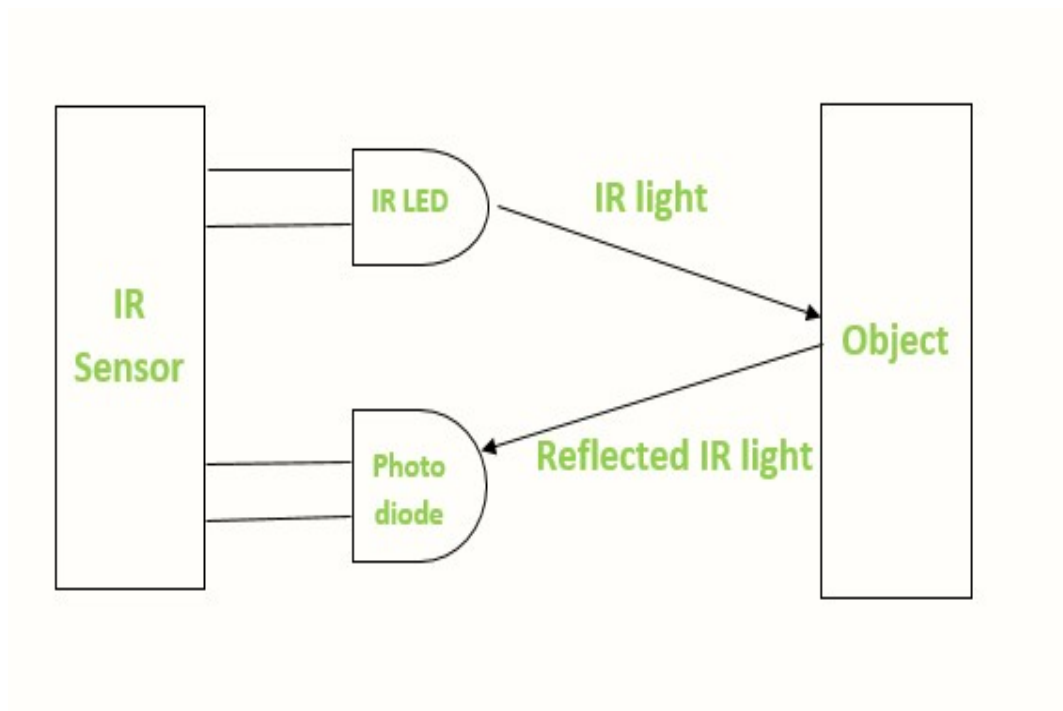
## What is an IR Sensor?

IR sensor is an electronic device that emits the light in order to sense some object of the surroundings. An **IR sensor** can measure the heat of an object as well as detects the motion. Usually, in the infrared **spectrum**, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, but infrared sensor can detect these radiations.

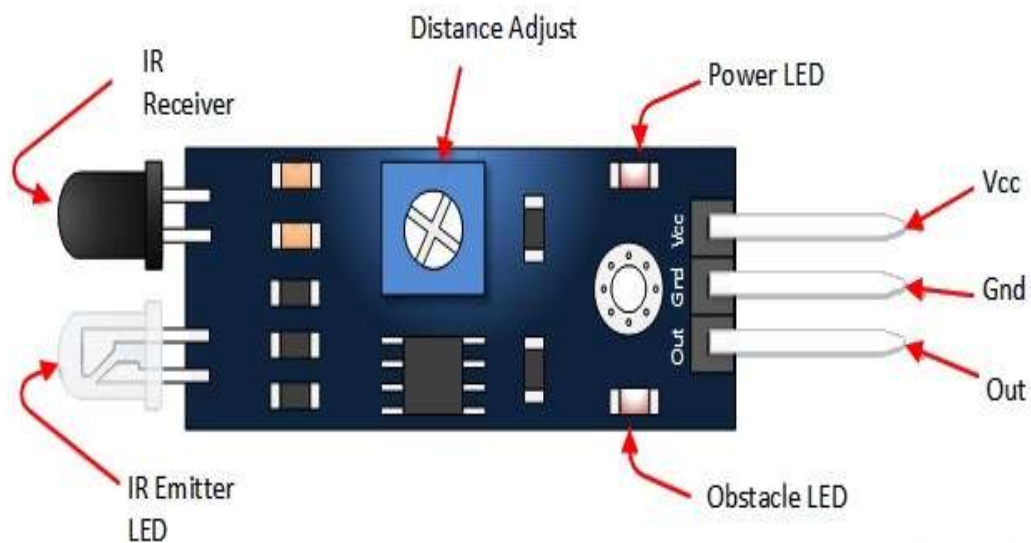


## How an IR Sensor work's?

An IR sensor consists of an IR LED (**transmitter LED**) and IR receiver (**Photodiode**), together they are called as PhotoCoupler or OptoCoupler. An Infrared Sensor works in the following sequence. IR transmitter transmits IR signal, as that signal detects any obstacle in its path, the transmitted IR signal reflects back from the obstacle and received by the receiver.

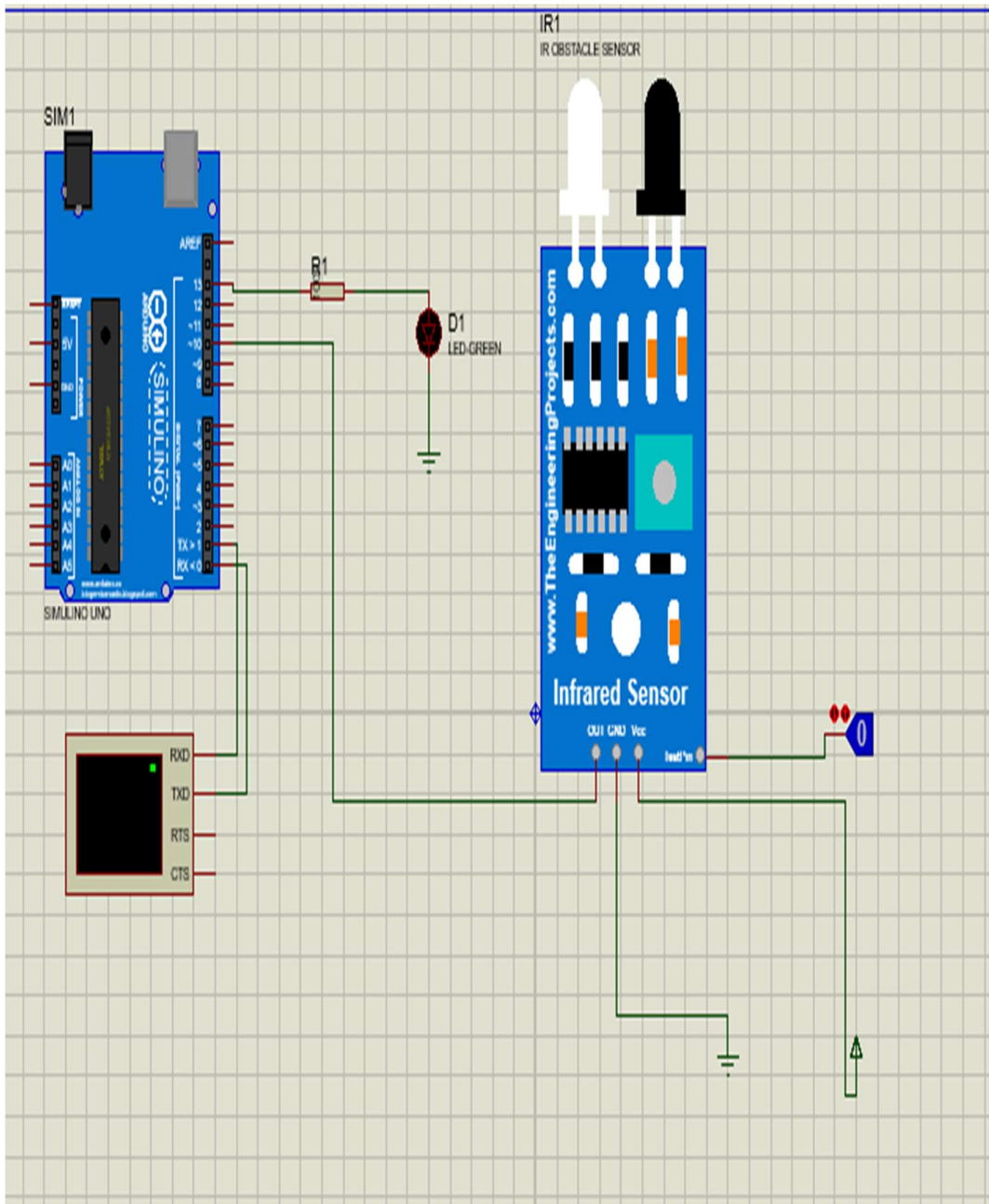


### IR Sensor Pinout



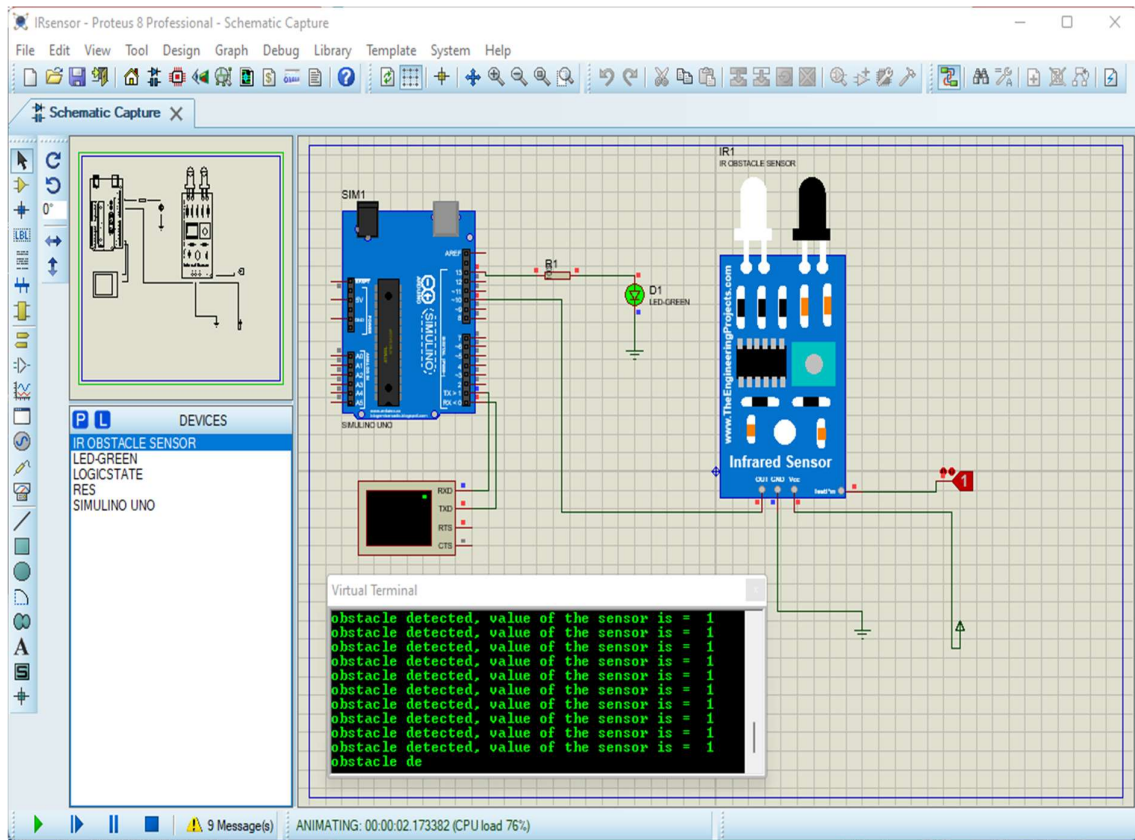
### Simulation of IR Sensor on Proteus

First install the IR Sensor Libraries the same way as the installation of the Arduino libraries on Proteus. Then open Proteus and open the "IR\_sensor.pdsprj" from the IR\_sensor folder.



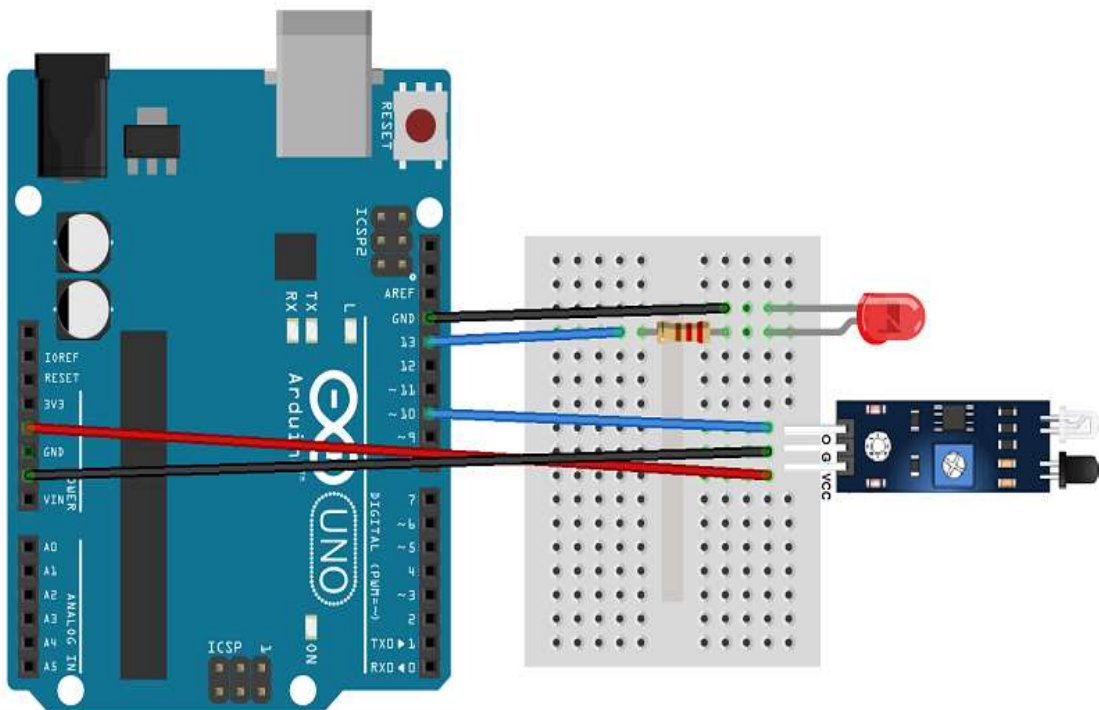
After that, double click on ultrasonic sensor and in the Program File upload the “**InfraredSensorsTEP.HEX**”. Then using the same steps upload the “**IR\_sensor.ino.hex**” file to Arduino and start the simulation.

Click on **Debug > Virtual Terminal** to open the Terminal output and use the Test Pin to play with Distance.



## Implementation using Arduino

Attach the IR Sensor to the Arduino Board as follow:



## Arduino Code

From the `IR_sensor` folder open "`IR_sensor.ino`" file.



```
IR_sensor | Arduino 1.8.16
File Edit Sketch Tools Help

IR_sensor

// define IR sensor pin and the led pin
int IRpin = 10;
int LEDpin = 13;
void setup() {
  Serial.begin(9600); // initialize the serial
  pinMode(IRpin,INPUT); // set ir sensor pin as input
  pinMode(LEDpin,OUTPUT); // set led pin as input
}

void loop() {

  if (digitalRead(IRpin) == HIGH){
    Serial.print("obstacle detected, value of the sensor is = ");
    Serial.println(digitalRead(IRpin)); // print the ir sensor value
    digitalWrite(LEDpin,HIGH); // turn on the led
    delay(10);
  }
  else {
    Serial.print("clear, value of the sensor is = ");
    Serial.println(digitalRead(IRpin)); // print the ir sensor value
    digitalWrite(LEDpin,LOW); // turn off the led
    delay(10);
  }
}

Done Saving.
Sketch uses 2562 bytes (7%) of program storage space. Maximum is 32256 bytes.
Global variables use 268 bytes (13%) of dynamic memory, leaving 1780 bytes for local
21 Arduino Uno on COM5
```

Compile the code then send it to the Arduino board. Click on the **Tools > Serial Monitor** to see the sensor values.