**IOT PROGRAM LIST**

1. **Blinking an on board LED**

void setup (){

// initialize digital pin LED\_BUILTIN as an output.

pinMode (LED\_BUILTIN, OUTPUT);

}

// the loop function runs over and over again forever

void loop () {

digitalWrite (LED\_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

delay (1000); // wait for a second

digitalWrite (LED\_BUILTIN, LOW); // turn the LED off by making the voltage LOW

delay (1000); // wait for a second

}

1. **Connecting external LED and blinking it.**

int led1=13;

void setup() {

// initialize digital pin LED\_BUILTIN as an output.

pinMode(13, OUTPUT);

}

// the loop function runs over and over again forever

void loop() {

digitalWrite(led1, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

digitalWrite(led1, LOW); // turn the LED off by making the voltage LOW

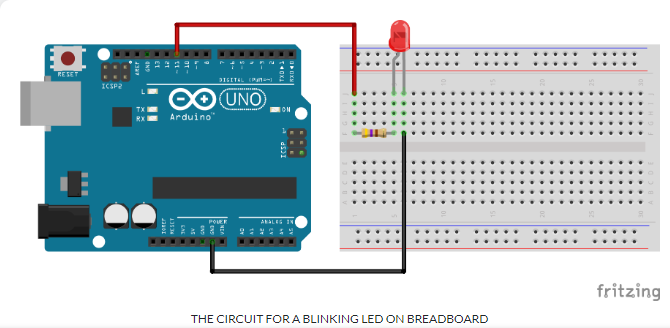
delay(1000); // wait for a second

}

**Note: Increase and decrease delay to see the variation in blinking.**

1. **Blinking an LED using Breadboard**

**Circuit Diagram**



**Code:**

void setup() {

// initialize digital pin 11 as an output.

pinMode(11, OUTPUT);

}

// the loop function runs over and over again forever

void loop() {

digitalWrite(11, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

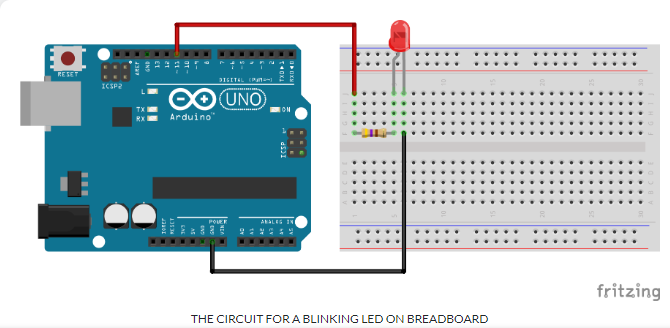
digitalWrite(11, LOW); // turn the LED off by making the voltage LOW

delay(1000);

}

**4. LED FADEIN/FADEOUT**

**Circuit diagram**



Code:

#define LED\_PIN 11

void setup()

{

pinMode(LED\_PIN, OUTPUT);

}

void loop()

{

for (int i = 0; i <= 255; i++) {

analogWrite(LED\_PIN, i);

delay(10);

}

for (int i = 255; i >= 0; i--) {

analogWrite(LED\_PIN, i);

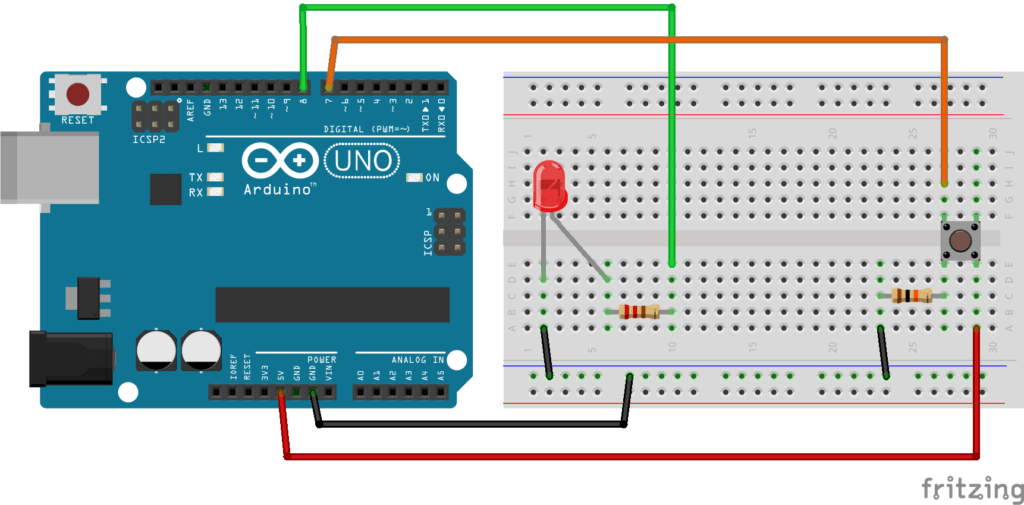
delay(10);

}

}

**5.Controlling LED using PUSH button**

Circuit Diagram



Code:

int led = 13;

int PinButton = 4;

void setup()

{

pinMode(PinButton, INPUT);

pinMode(led, OUTPUT);

}

void loop()

{

int stateButton = digitalRead(PinButton);

if (stateButton == 1)

{

digitalWrite(led, HIGH); // Turn on led

}

else

{

digitalWrite(led, LOW); //Turn off led

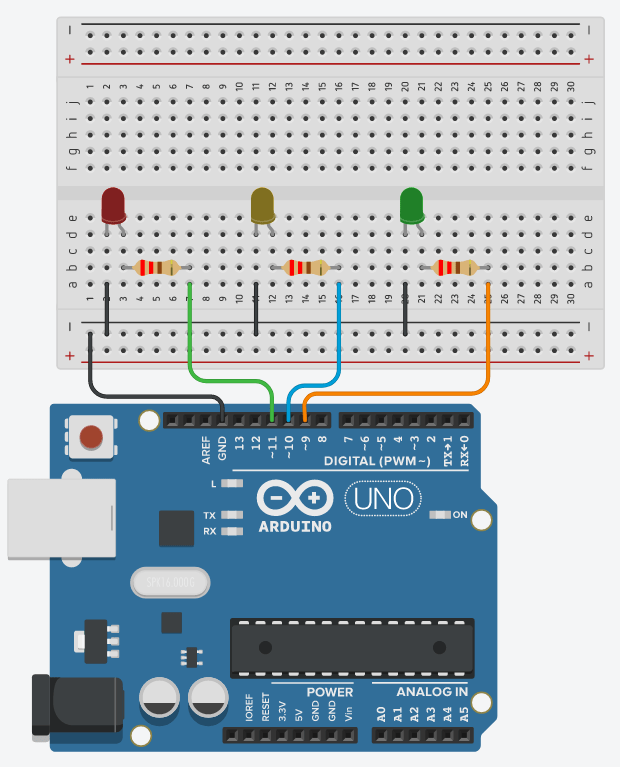
}

delay(20);

}

**6. Traffic Light Controller**

**Circuit diagram**



Program

#define LED\_PIN\_1 11

#define LED\_PIN\_2 10

#define LED\_PIN\_3 9

void setup()

{

pinMode(LED\_PIN\_1, OUTPUT);

pinMode(LED\_PIN\_2, OUTPUT);

pinMode(LED\_PIN\_3, OUTPUT);

}

void loop()

{

digitalWrite(LED\_PIN\_1, HIGH);

digitalWrite(LED\_PIN\_2, LOW);

digitalWrite(LED\_PIN\_3, LOW);

delay(1000);

digitalWrite(LED\_PIN\_1, LOW);

digitalWrite(LED\_PIN\_2, HIGH);

digitalWrite(LED\_PIN\_3, LOW);

delay(1000);

digitalWrite(LED\_PIN\_1, LOW);

digitalWrite(LED\_PIN\_2, LOW);

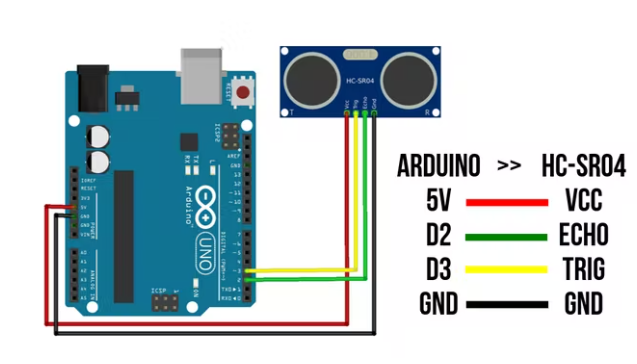
digitalWrite(LED\_PIN\_3, HIGH);

delay(1000);

}

7. Ultrasonic Sensor

**Circuit diagram**



**Code**

// defines variables

long duration; // variable for the duration of sound wave travel

int distance; // variable for the distance measurement

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 is starting with 9600 of baudrate speed

Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor

Serial.println("with Arduino UNO R3");

}

void loop() {

// Clears the trigPin condition

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Sets the trigPin HIGH (ACTIVE) for 10 microseconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);

// Calculating the distance

distance = duration \* 0.034 / 2; // Speed of sound wave divided by 2 (go and back)

// Displays the distance on the Serial Monitor

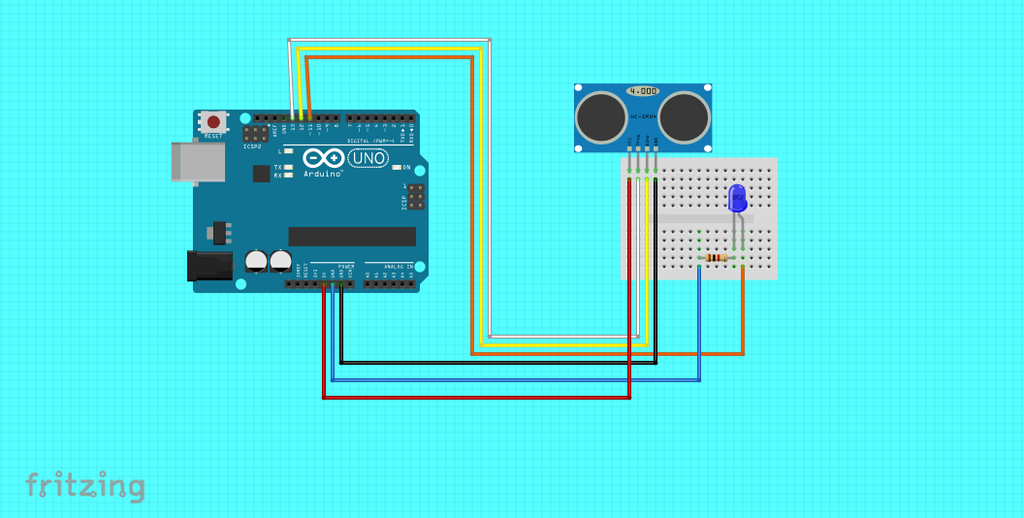
Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

}

8. Controlling LED using ultrasonic sensor



#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

// defines variables

long duration; // variable for the duration of sound wave travel

int distance; // variable for the distance measurement

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 is starting with 9600 of baudrate speed

Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor

Serial.println("with Arduino UNO R3");

}

void loop() {

// Clears the trigPin condition

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Sets the trigPin HIGH (ACTIVE) for 10 microseconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);

// Calculating the distance

distance = duration \* 0.034 / 2; // Speed of sound wave divided by 2 (go and back)

if (distance < 10)

{ digitalWrite(8,HIGH);

}

else {

digitalWrite(8,LOW);

}

// Displays the distance on the Serial Monitor

Serial.print("Distance: ");

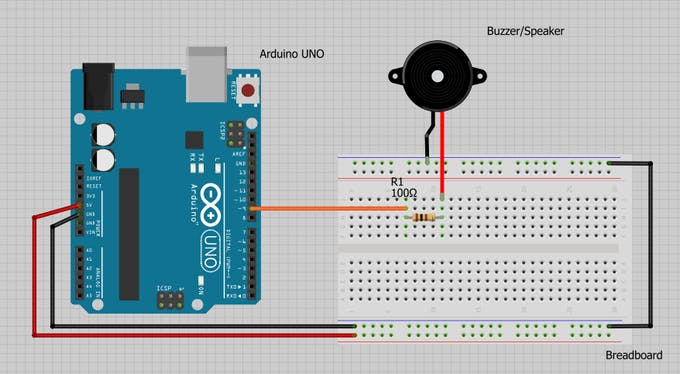
Serial.print(distance);

Serial.println(" cm");

}

9. Buzzer program

Circuit diagram



Code

const int buzzer = 9; //buzzer to arduino pin 9

void setup(){

pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output

}

void loop(){

tone(buzzer, 1000); // Send 1KHz sound signal...

delay(1000); // ...for 1 sec

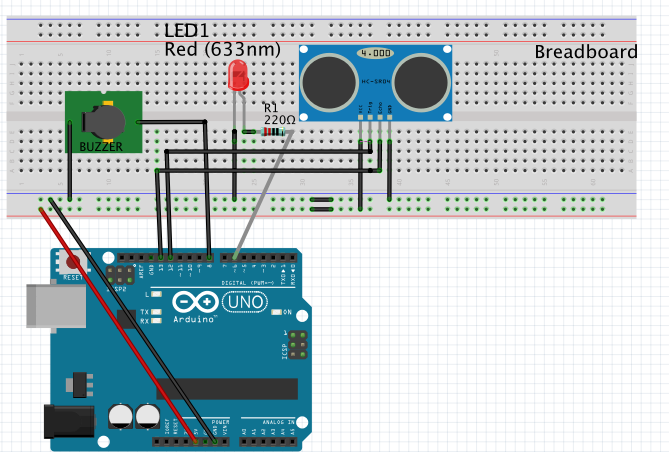
noTone(buzzer); // Stop sound...

delay(1000); // ...for 1sec

}

10. Program with ultrasonic sensor, led and buzzer (intrusion detection)

Circuit diagram



Code:

#define trigPin 12

#define echoPin 13

int Buzzer = 8; // Connect buzzer pin to 8

int ledPin= 6; //Connect LEd pin to 6

int duration, distance; //to measure the distance and time taken

void setup() {

Serial.begin (9600);

//Define the output and input objects(devices)

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(Buzzer, OUTPUT);

pinMode(ledPin, OUTPUT);

}

void loop() {

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration/2) / 29.1;

//when distance is greater than or equal to 200 OR less than or equal to 0,the buzzer and LED are off

if (distance >0 && distance<50)

{

Serial.println("object detected \n");

Serial.print("distance= ");

Serial.print(distance); //prints the distance if it is between the range 0 to 200

digitalWrite(ledPin,HIGH);

tone(Buzzer,50); // play tone of 400Hz for 500 ms

delay(1000);

}

else {

Serial.println("no object detected");

noTone(Buzzer);

digitalWrite(ledPin,LOW);

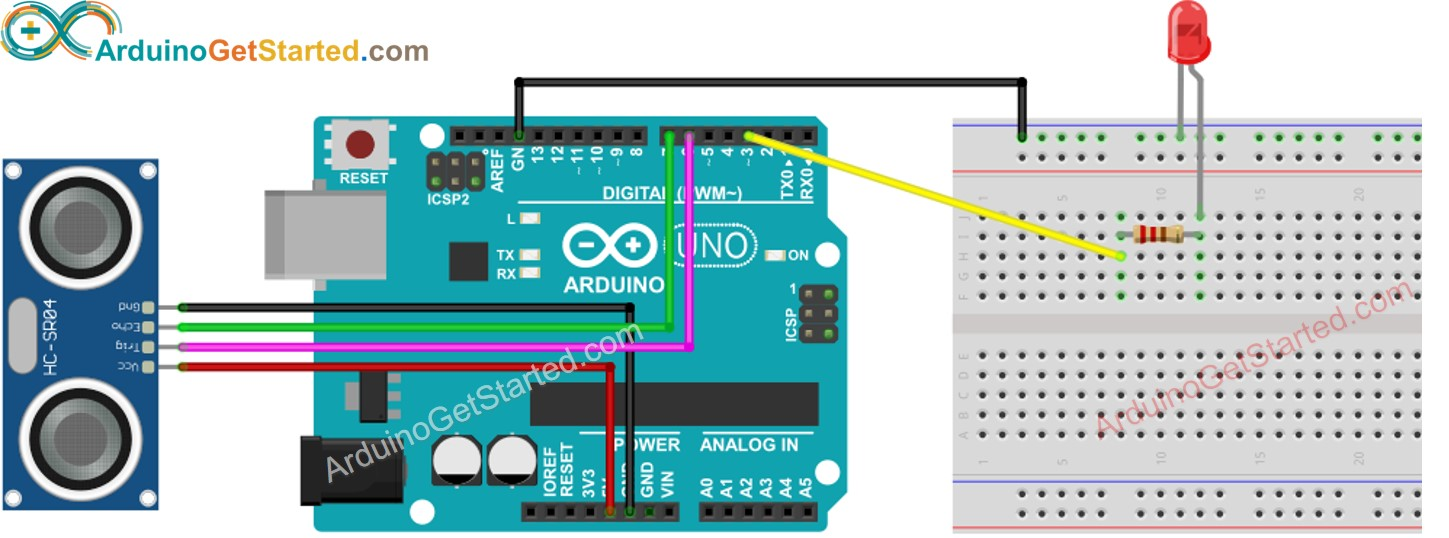
}

}

11. Controlling multiple LED’s with ultrasonic sensor

Circuit diagram:

(Connect 2 more LED’s in the same way)



Code:

const int TRIG\_PIN = 12; // Arduino pin connected to Ultrasonic Sensor's TRIG pin

const int ECHO\_PIN = 13; // Arduino pin connected to Ultrasonic Sensor's ECHO pin

const int LED1 = 3;// Arduino pin connected to LED's pin

const int LED2 = 4;

const int LED3 = 5;

// variables will change:

float duration\_us, distance\_cm;

void setup() {

Serial.begin (9600); // initialize serial port

pinMode(TRIG\_PIN, OUTPUT); // set arduino pin to output mode

pinMode(ECHO\_PIN, INPUT); // set arduino pin to input mode

pinMode(LED1, OUTPUT); // set arduino pin to output mode

pinMode(LED2, OUTPUT);

pinMode(LED3, OUTPUT);

}

void loop() {

// generate 10-microsecond pulse to TRIG pin

digitalWrite(TRIG\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG\_PIN, LOW);

// measure duration of pulse from ECHO pin

duration\_us = pulseIn(ECHO\_PIN, HIGH);

// calculate the distance

distance\_cm = 0.017 \* duration\_us;

if(distance\_cm < 20)

digitalWrite(LED1, HIGH); // turn on LED

else

digitalWrite(LED1, LOW); // turn off LED

if(distance\_cm > 20 && distance\_cm<50)

digitalWrite(LED2, HIGH); // turn on LED

else

digitalWrite(LED2, LOW); // turn off LED

if(distance\_cm > 50 && distance\_cm<100)

digitalWrite(LED3, HIGH); // turn on LED

else

digitalWrite(LED3, LOW); // turn off LED

// print the value to Serial Monitor

Serial.print("distance: ");

Serial.print(distance\_cm);

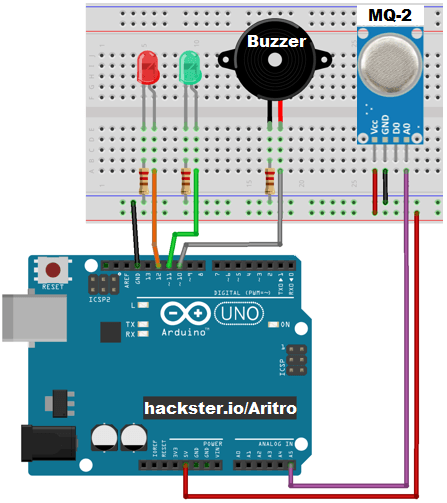
Serial.println(" cm");

delay(500);

}

12. Smoke detector with LED and Buzzer.

Circuit diagram



int redLed = 7;

int buzzer = 8;

int smokeA0 = A5;

// Your threshold value

int sensorThres = 400;

void setup() {

pinMode(redLed, OUTPUT);

pinMode(buzzer, OUTPUT);

pinMode(smokeA0, INPUT);

Serial.begin(9600);

}

void loop() {

int analogSensor = analogRead(smokeA0);

Serial.print("Pin A0: ");

Serial.println(analogSensor);

// Checks if it has reached the threshold value

if (analogSensor > sensorThres)

{

digitalWrite(redLed, HIGH);

tone(buzzer, 1000, 200);

}

else

{

digitalWrite(redLed, LOW);

noTone(buzzer);

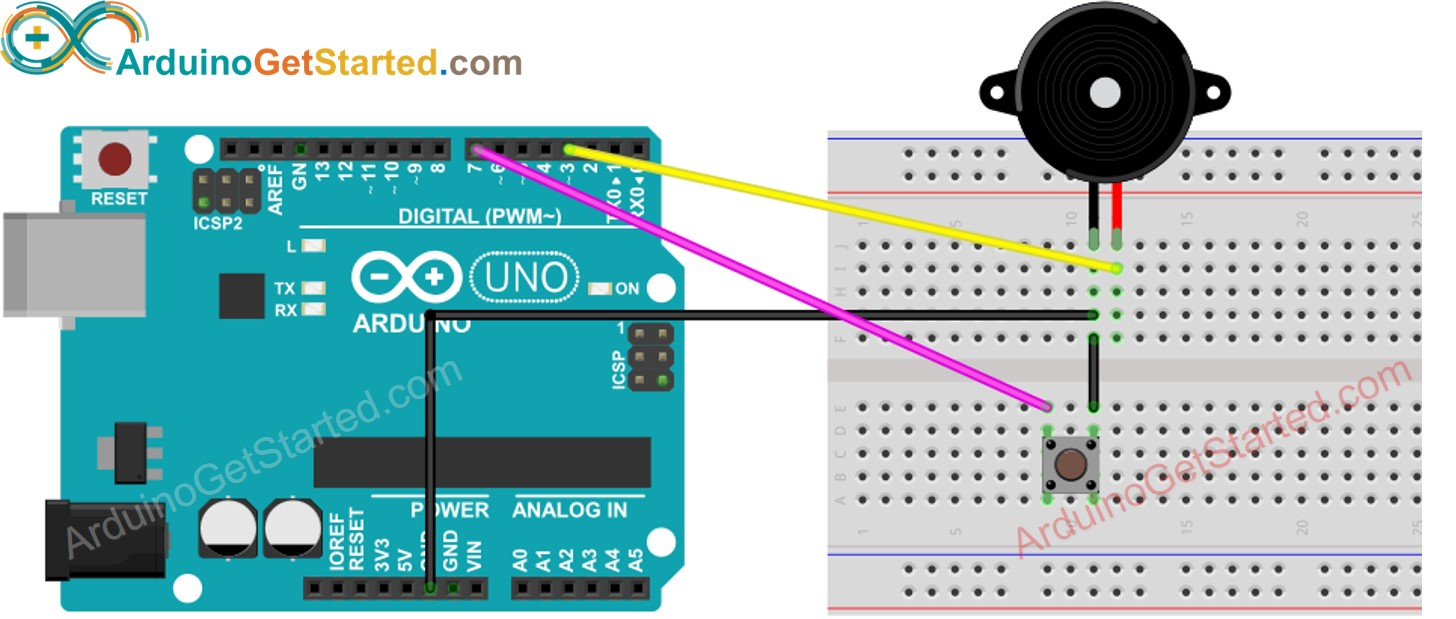
}

delay(100);

}

13. Controlling Buzzer using push button

Circuit diagram:



Code:

int buzzer = 13;

int PinButton = 4;

void setup()

{

pinMode(PinButton, INPUT);

pinMode(buzzer, OUTPUT);

}

void loop()

{

int stateButton = digitalRead(PinButton);

if (stateButton == 1)

{

tone(buzzer,100); // Turn on led

}

else

{

noTone(buzzer); //Turn off led

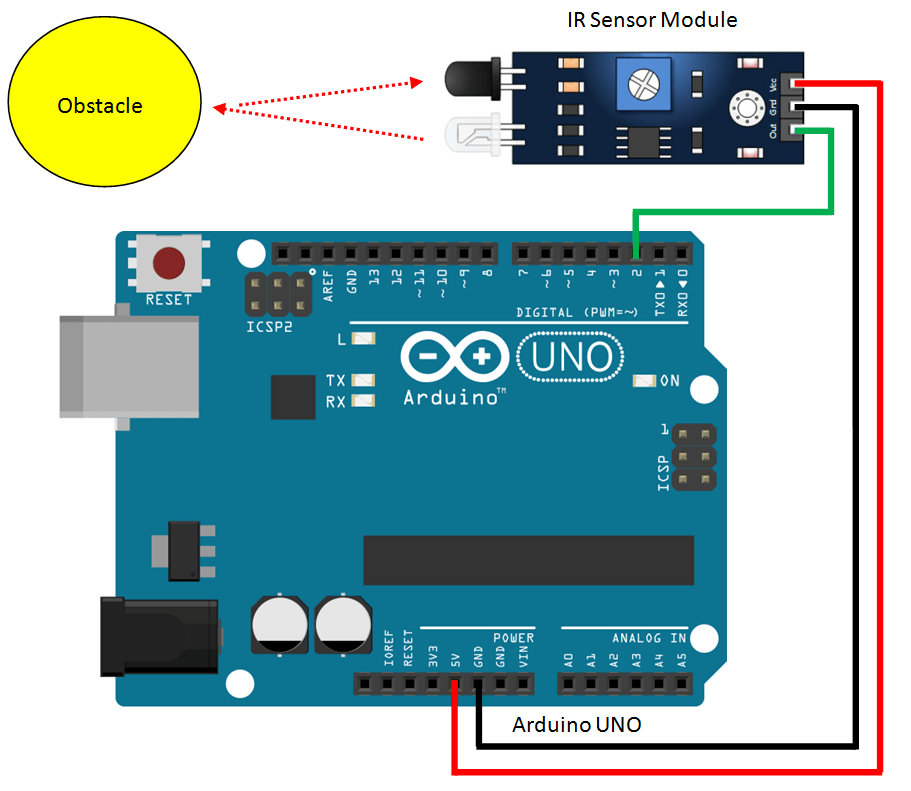
}

delay(20);

}

14.IR sensor with Led

ONLY IR SENSOR



Code:

int IRSensor = 7; // connect ir sensor to arduino pin 7

void setup()

{

pinMode (IRSensor, INPUT); // sensor pin INPUT

Serial.begin(9600); // open the serial port at 9600 bps:

}

void loop()

{

int statusSensor = digitalRead (IRSensor);

if (statusSensor == 1)

Serial.print("HIGH- No Obstacle/Black Surface\n");

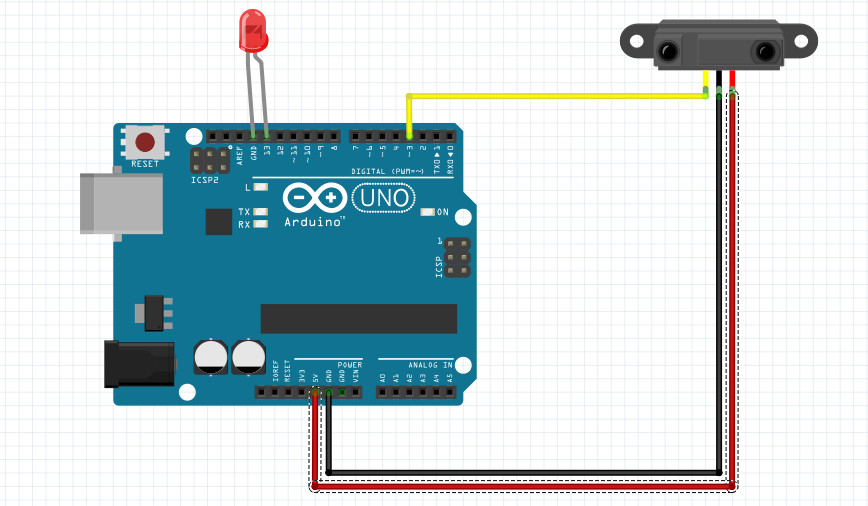
else

Serial.print("LOW- Obstacle Detected/White Surface\n");

}

15.IR SENSOR WITH LED

Circuit diagram



void setup()

{

pinMode(13,OUTPUT);

pinMode(7,INPUT);

Serial.begin(9600);

}

void loop()

{

if (digitalRead(7)== LOW)

{

digitalWrite(13,HIGH);

delay(10);

}

else

{

digitalWrite(13,LOW);

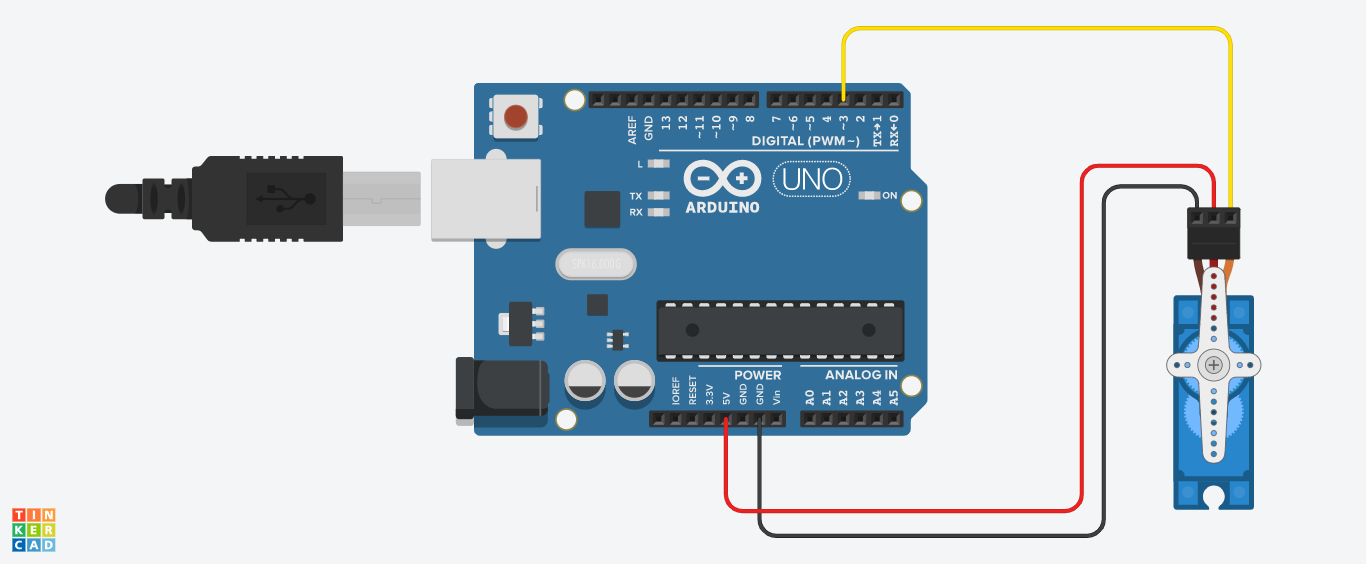
delay(10);

}

}

16. Servomotor using Arduino

Circuit diagram



Code:

#include<Servo.h>

Servo Myservo;

int pos;

void setup()

{

Myservo.attach(3);

}

void loop()

{

for(pos=0;pos<=180;pos++){

Myservo.write(pos);

delay(15);

}

delay(1000);

for(pos=180;pos>=0;pos--){

Myservo.write(pos);

delay(15);

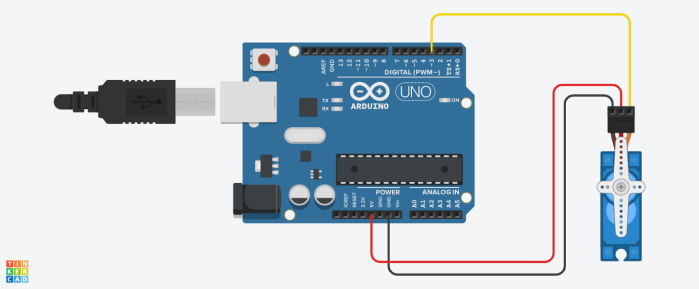
}

delay(1000);

}

Note: change delay and observe the movement of shaft.

16. rotating 0,90,180 arduino



Code:

#include <Servo.h>

Servo servo1;

int servoPin = 3;

void setup(){

servo1.attach(servoPin);

}

void loop(){

servo1.write(0);

delay(1000);

servo1.write(90);

delay(1000);

servo1.write(180);

delay(1000);

}

**LED ON and OFF using Serial input**

https://microdigisoft.com/data.store/projects/Sketches/Arduino\_Serialmonitor.ino

int LedPin = 2; // Connect RED LED to pin number 2

int counter = 0;

void setup() {

pinMode(LedPin, OUTPUT); //declare LED pin as a output

Serial.begin(9600); //Set serial to the 9600 band

while (! Serial);

Serial.println("Enter Yes to turn on the LED:");

}

void loop(){

if (Serial.available())

{

char ch = Serial.read();

if (ch == 'y' || ch == 'Y')

{

digitalWrite(LedPin, HIGH);

Serial.println("LED TURN ON!!");

Serial.print("The LED switched off for : ");

Serial.print(counter);

Serial.println(" Seconds");

Serial.println("Do you want to switch off LED, Enter NO or no!");

counter = 0;

}

if (ch == 'n' || ch == 'N')

{

digitalWrite(LedPin, LOW);

Serial.println("LED TURN OFF!!");

Serial.print("The LED Switched ON for : ");

Serial.print(counter);

Serial.println(" Seconds");

Serial.println("Do you want to switch ON, Enter YES or yes!");

counter = 0;

}

}

delay(1000);

counter += 1;

}

**2.Important Sensors based Projects**

* + - 1. **PIR Sensor**
      2. **Ultrasonic Sensor**
      3. **Temperature and Humidity Sensor**

What is a PIR sensor?(Arduino JavatPoint Tutorial)

The Passive Infra-Red sensors or PIR sensors detect motion or movement of an object that detect infrared radiations, such as the human body. Hence, the use of sensors is very common.

The advantages of using a PIR sensor are listed below:

* Inexpensive
* Adjustable module
* Efficient
* Small in size
* Less power consumption
* It can detect motion in the dark as well as light.

The PIR sensor is shown below:



The PIR sensor has three terminals, which are listed below:

* VCC
* Digital Output
* GND (Ground)

AD

We will connect the Vcc terminal of the sensor to the 5V on the [Arduino board](https://www.javatpoint.com/arduino-boards). The PIR's sensor output can be connected to any of the digital pins on the [Arduino](https://www.javatpoint.com/arduino) board.

The applications of the PIR sensor are automation, security systems, etc. Such sensors work great in detecting the entrance of a person in an area and leaving it.

The detection range of PIR sensors is from 5m to 12m.

Working of PIR Sensors

The working of the PIR sensor is entirely based on detecting the IR (Infra-Red) radiations, which are either emitted or reflected by the objects.

The infrared radiations are detected by the crystalline material present at the center of the sensor.

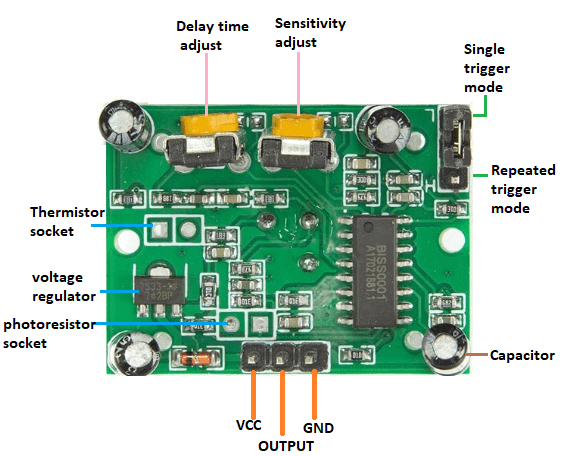
Consider a person passing in front of the background like a wall, etc. The temperature changes from room to body temperature and vice-versa within the sensor field. Changes arising in the arrival infrared radiations are converted by the sensor to the output voltage. It later detects the human body or object.

Structure of PIR Sensor

A round metal can is mounted on the center with the rectangular crystal that detects the IR radiations.

A ball like a lens present on some sensors helps in enhancing the viewing angle.

The bottom part of the sensor contains many circuits mounted on it, which is shown below:



Let's start with the project.

Hardware Required

The components required for the project are listed below:

* 1 x PIR motion sensor
* Arduino UNO R3 board (We can take any Arduino board).
* Jump wires
* 1 x red LED (we can take LED of any color)
* 1 x 220 Ohm resistor

Principle

The movement of jumper present on the sensor on the L side will cause a change in the state of the sensor whenever the motion is detected. Such a condition is defined as a single trigger mode.

AD

When the sensor resets the timer after every detection of motion, it is defined as repeated trigger mode.

The two potentiometers present on the sensor are called as **Sensitivity** Potentiometer and **Time** Potentiometer. We can adjust both the parameters (time and sensitivity) accordingly.

It should be restricted for atleast 15 seconds in front of the PIR sensor for proper calibration in the output. After 15 seconds, the sensor can easily detect movements.

If any movement is detected, the LED will be HIGH. If there is no such movement, the output will be LOW.

Connection

The steps to set up the connection are listed below:

AD

* Connect the Vcc terminal of the PIR sensor to the 5V pin of the Arduino board.
* Connect the Output terminal of the PIR sensor to pin 8 of the Arduino board.
* Connect the GND terminal of the PIR sensor to the Ground pin of the Arduino board.
* Connect the positive leg of the LED in series with 220 Ohm resistor to pin 13 of the Arduino board.
* Connect the negative terminal of the LED to the Ground pin of the Arduino board.

Sketch

Consider the below code:

1. **int** LEDpin = 13;   // LED pin
2. **int** PIRpin = 8;    // The pin of Arduino connected to the PIR output
3. **int** PIRvalue = 0;  // It specifies the status of PIR sensor
4. **void** setup() {
5. pinMode(LEDpin, OUTPUT);
6. pinMode(PIRpin, INPUT);
7. // the output from the sensor is considered as input for Arduino
8. Serial.begin(9600);
9. }
10. **void** loop()
11. {
12. PIRvalue = digitalRead(PIRpin);
13. **if** (PIRvalue == HIGH)
14. {
15. digitalWrite(LEDpin, HIGH);
16. // turn ON LED if the motion is detected
17. Serial.println("hello, I found you...heyyy..");
18. }
19. **else**
20. {
21. digitalWrite(LEDpin, LOW);
22. // LED will turn OFF if we have no motion
23. Serial.println("I cannot find you");
24. delay(1000);
25. }
26. }

Steps to upload the code to the project

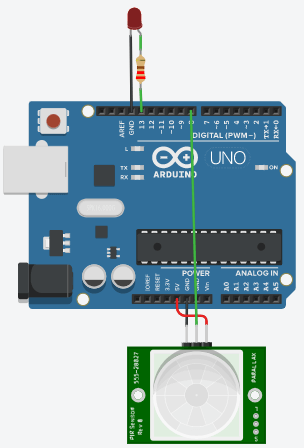
The steps are listed below:

* Open the Arduino IDE.
* Select the type of board from Tools -> Board -> Arduino UNO.
* Select the port from Tools -> Port -> COM..
* Upload the sketch to the connection diagram.

Connection Diagram

We will show the connection using the Simulator so that the connections become clearer and more precise.

We can make the same connection using the hardware devices.



The output will be based on the detection.

**2. Temperature Humidity Senor**

Source: <https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/>

**HOW TO SET UP THE DHT11 HUMIDITY SENSOR ON AN ARDUINO**

Posted by [Scott Campbell](https://www.circuitbasics.com/author/circuitbasicsgmail-com/) | [Arduino](https://www.circuitbasics.com/arduino/) | [276](https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/#comments)

The [DHT11 humidity and temperature](https://www.amazon.com/gp/product/B01DKC2GQ0/ref=as_li_qf_asin_il_tl?ie=UTF8&tag=circbasi-20&creative=9325&linkCode=as2&creativeASIN=B01DKC2GQ0&linkId=e60f134123941e579cc73e26d34fef60) sensor makes it really easy to add humidity and temperature data to your DIY electronics projects. It’s perfect for remote weather stations, home environmental control systems, and farm or garden monitoring systems.

In this tutorial, I’ll first go into a little background about humidity, then I’ll explain how the DHT11 measures humidity. After that, I’ll show you how to connect the DHT11 to an [Arduino](http://www.amazon.com/gp/product/B008GRTSV6/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B008GRTSV6&linkCode=as2&tag=circbasi-20&linkId=SO3YILYNYLXWVBKI) and give you some example code so you can use the DHT11 in your own projects.

Here are the ranges and accuracy of the DHT11:

* Humidity Range: 20-90% RH
* Humidity Accuracy: ±5% RH
* Temperature Range: 0-50 °C
* Temperature Accuracy: ±2% °C
* Operating Voltage: 3V to 5.5V

The DHT11 Datasheet:

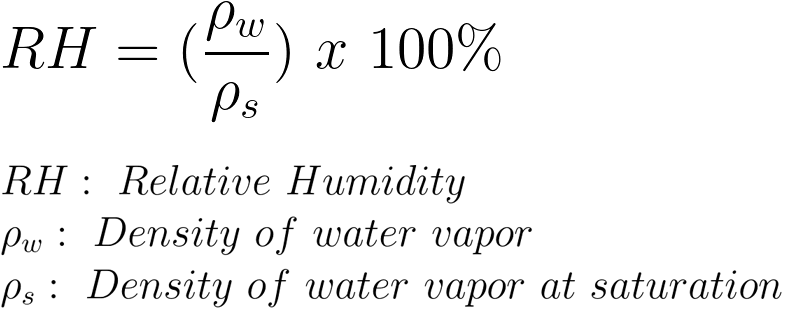
[](https://www.circuitbasics.com/wp-content/uploads/2015/11/DHT11-Datasheet.pdf) [DHT11 Datasheet](https://www.circuitbasics.com/wp-content/uploads/2015/11/DHT11-Datasheet.pdf)

**WHAT IS RELATIVE HUMIDITY?**

The DHT11 measures *relative humidity*. Relative humidity is the amount of water vapor in air vs. the saturation point of water vapor in air. At the saturation point, water vapor starts to condense and accumulate on surfaces forming dew.

The saturation point changes with air temperature. Cold air can hold less water vapor before it becomes saturated, and hot air can hold more water vapor before it becomes saturated.

The formula to calculate relative humidity is:



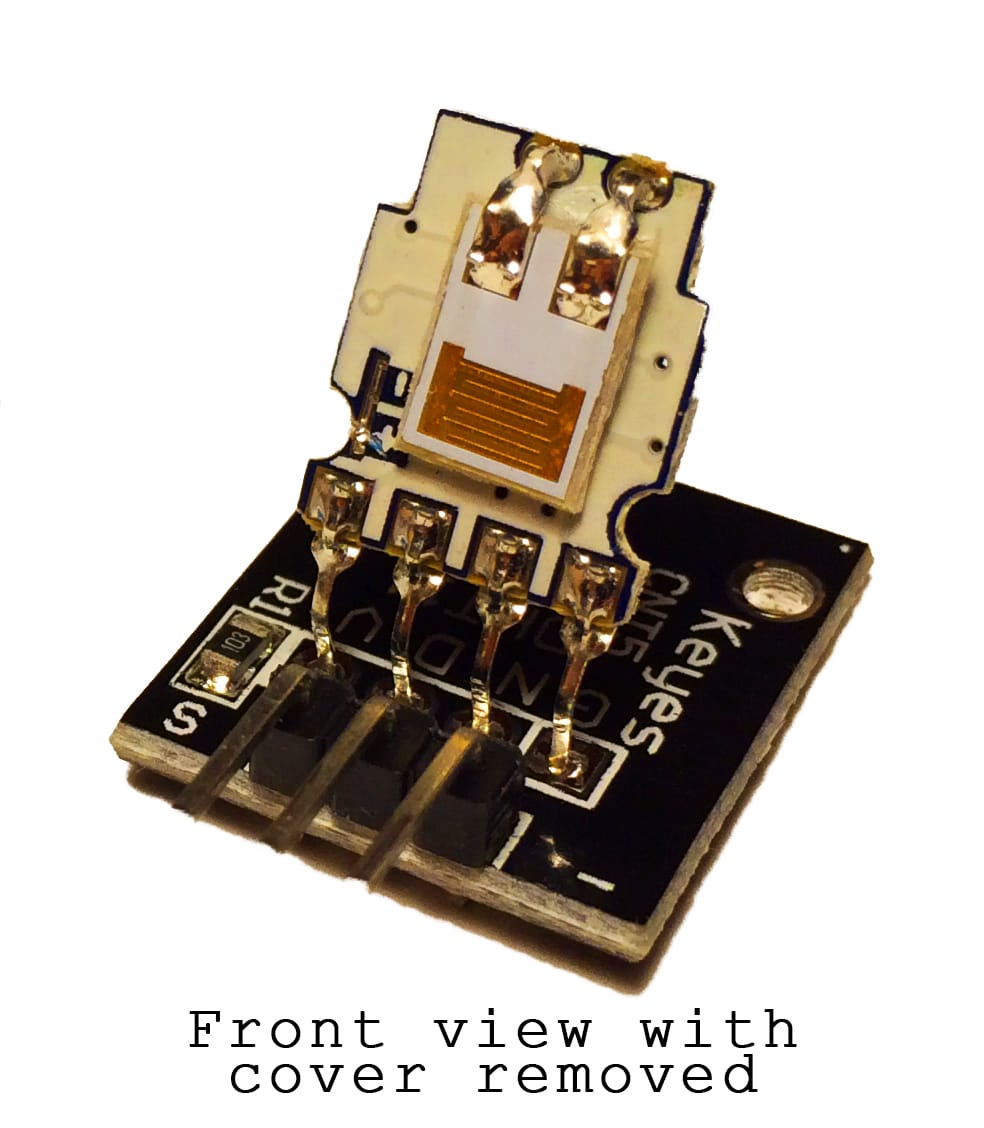
Relative humidity is expressed as a percentage. At 100% RH, condensation occurs, and at 0% RH, the air is completely dry.

**HOW THE DHT11 MEASURES HUMIDITY AND TEMPERATURE**

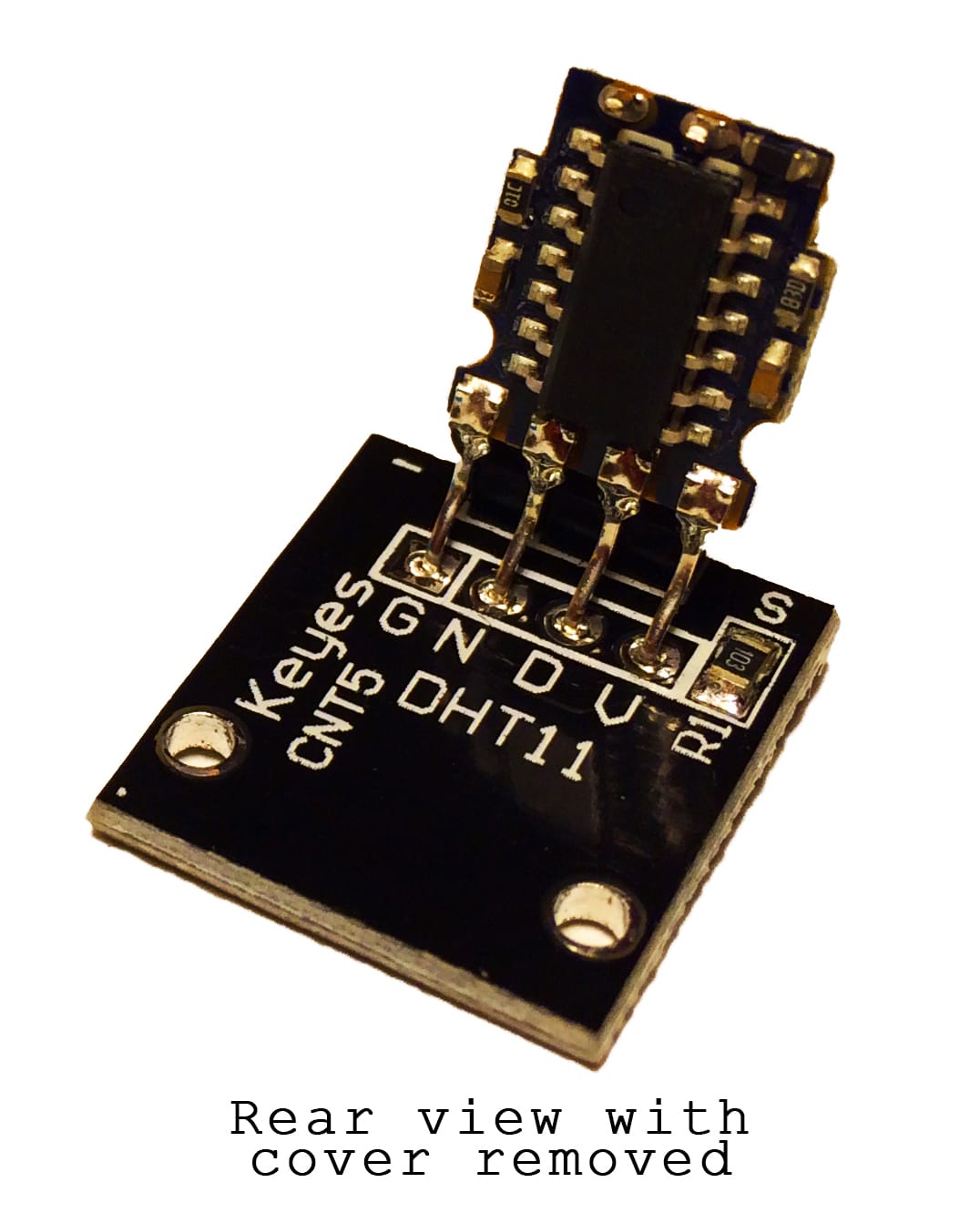
The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes.

The DHT11 measures temperature with a surface mounted [NTC temperature sensor](https://www.amazon.com/gp/product/B018QL5LPI/ref=as_li_ss_tl?imprToken=Xsq9niJAoOMwNpO9D8E6PQ&slotNum=0&ie=UTF8&linkCode=ll1&tag=circbasi-20&linkId=18ce70872cd2517e3d23e4d79957e233) (thermistor) built into the unit. To learn more about how thermistors work and how to use them on the Arduino, check out our [Arduino Thermistor Temperature Sensor Tutorial](https://www.circuitbasics.com/arduino-thermistor-temperature-sensor-tutorial).

With the plastic housing removed, you can see the electrodes applied to the substrate:

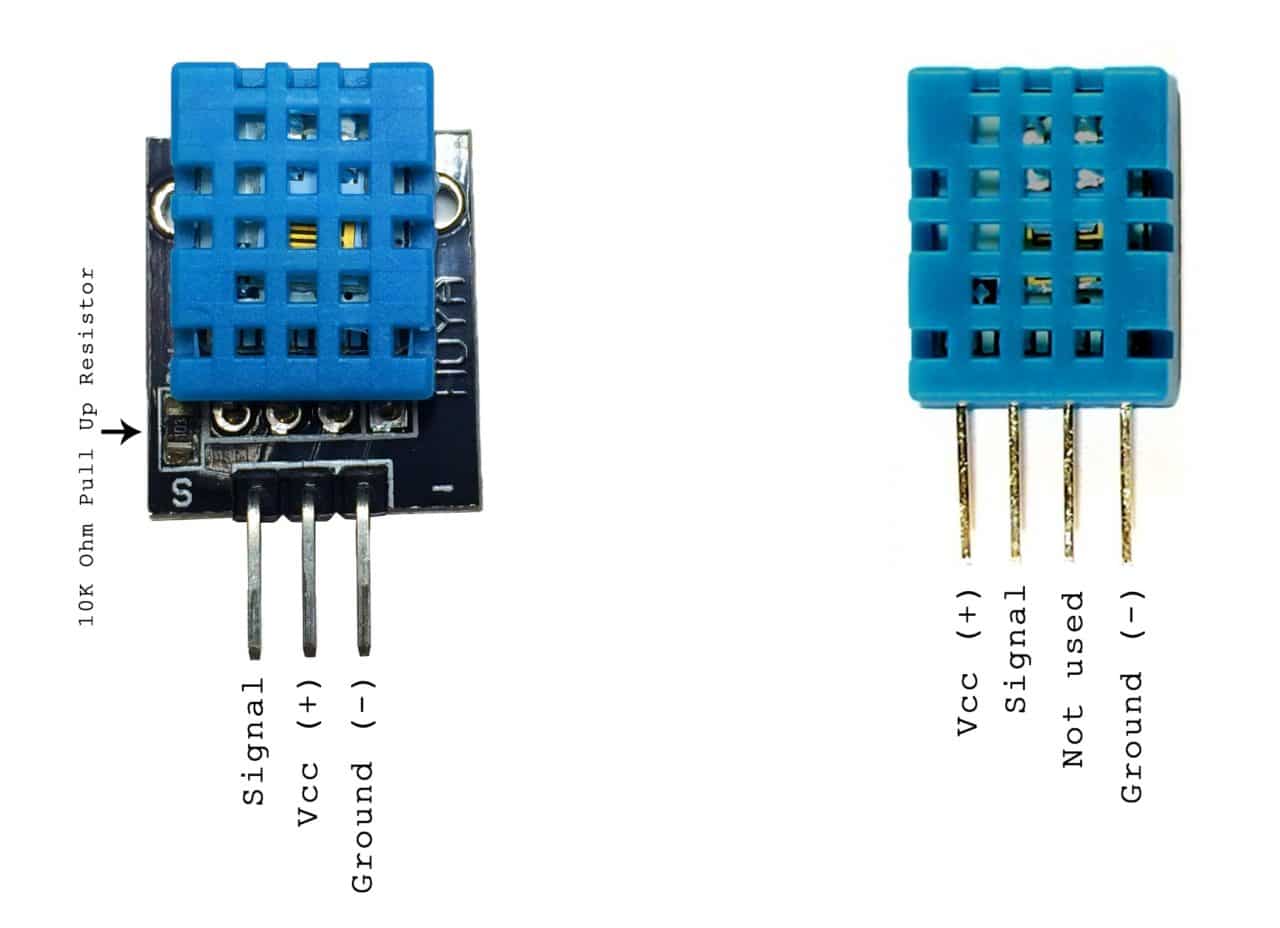
[](https://www.circuitbasics.com/wp-content/uploads/2015/12/DHT11-Temperature-and-Humidity-Sensor-Inside-Front-with-Cover-Removed.jpg)

An IC mounted on the back of the unit converts the resistance measurement to relative humidity. It also stores the calibration coefficients, and controls the data signal transmission between the DHT11 and the Arduino:

[](https://www.circuitbasics.com/wp-content/uploads/2015/12/DHT11-Temperature-and-Humidity-Sensor-Inside-Back-with-Cover-Removed.jpg)

The DHT11 uses just one signal wire to transmit data to the Arduino. Power comes from separate 5V and ground wires. A 10K Ohm pull-up resistor is needed between the signal line and 5V line to make sure the signal level stays high by default (see the datasheet for more info).

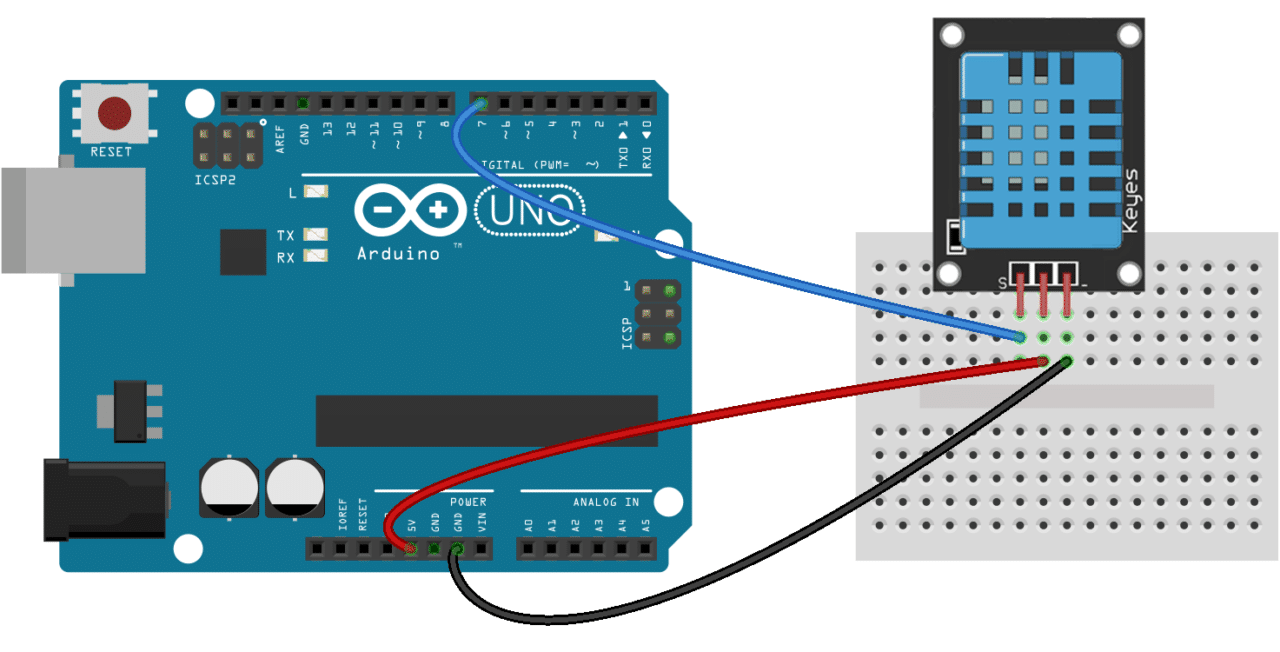
There are two different versions of the DHT11 you might come across. One type has four pins, and the other type has three pins and is mounted to a small PCB. The PCB mounted version is nice because it includes a surface mounted 10K Ohm pull up resistor for the signal line. Here are the pin outs for both versions:

[](https://www.circuitbasics.com/wp-content/uploads/2015/12/DHT11-Pinout-for-three-pin-and-four-pin-types-2.jpg)

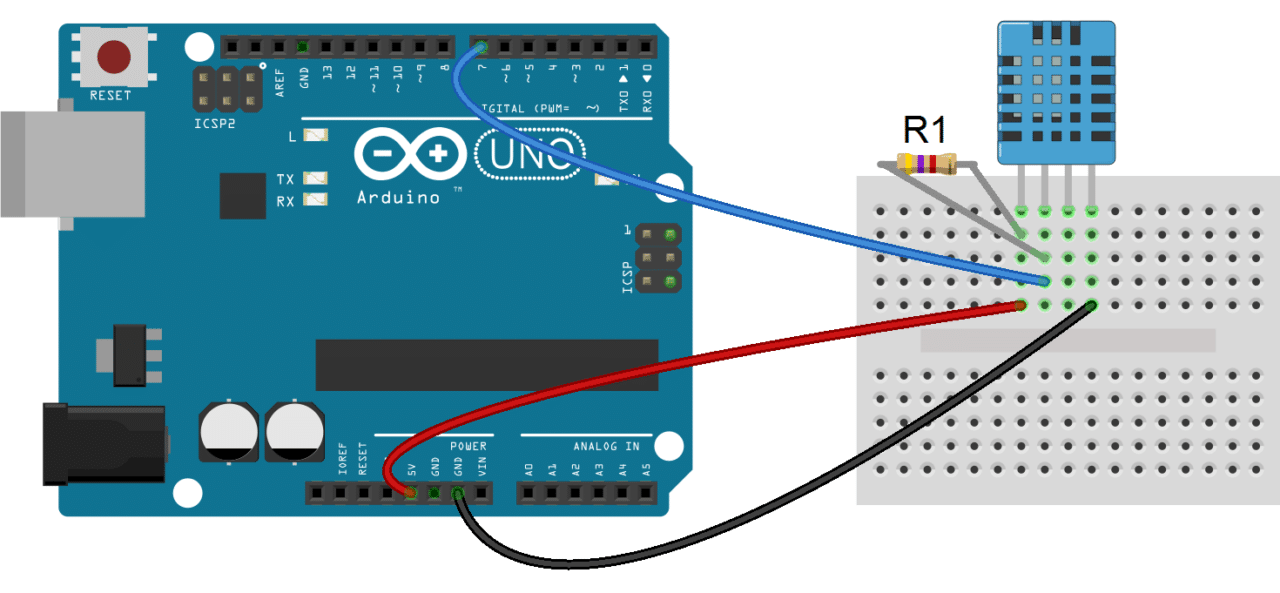
**HOW TO SET UP THE DHT11 ON AN ARDUINO**

Wiring the DHT11 to the Arduino is really easy, but the connections are different depending on which type you have.

**CONNECTING A THREE PIN DHT11:**

[](https://www.circuitbasics.com/wp-content/uploads/2015/10/Arduino-DHT11-Tutorial-3-Pin-DHT11-Wiring-Diagram.png)

**CONNECTING A FOUR PIN DHT11:**

[](https://www.circuitbasics.com/wp-content/uploads/2015/10/Arduino-DHT11-Tutorial-4-Pin-DHT11-Wiring-Diagram.png)

* R1: 10K Ohm pull up resistor

**DISPLAY HUMIDITY AND TEMPERATURE ON THE SERIAL MONITOR**

Before you can use the DHT11 on the Arduino, you’ll need to install the [DHTLib library](http://playground.arduino.cc/Main/DHTLib" \o "DHTLib Library" \t "_blank). It has all the functions needed to get the humidity and temperature readings from the sensor. It’s easy to install, just download the DHTLib.zip file below and open up the Arduino IDE. Then go to Sketch>Include Library>Add .ZIP Library and select the DHTLib.zip file.

[](https://www.circuitbasics.com/wp-content/uploads/2015/10/DHTLib.zip) [DHTLib](https://www.circuitbasics.com/wp-content/uploads/2015/10/DHTLib.zip)

After it’s installed, upload this example program to the Arduino and open the serial monitor:

#include <dht.h>

dht DHT;

#define DHT11\_PIN 7

void setup(){

Serial.begin(9600);

}

void loop(){

int chk = DHT.read11(DHT11\_PIN);

Serial.print("Temperature = ");

Serial.println(DHT.temperature);

Serial.print("Humidity = ");

Serial.println(DHT.humidity);

delay(1000);

}

You should see the humidity and temperature readings displayed at one second intervals.

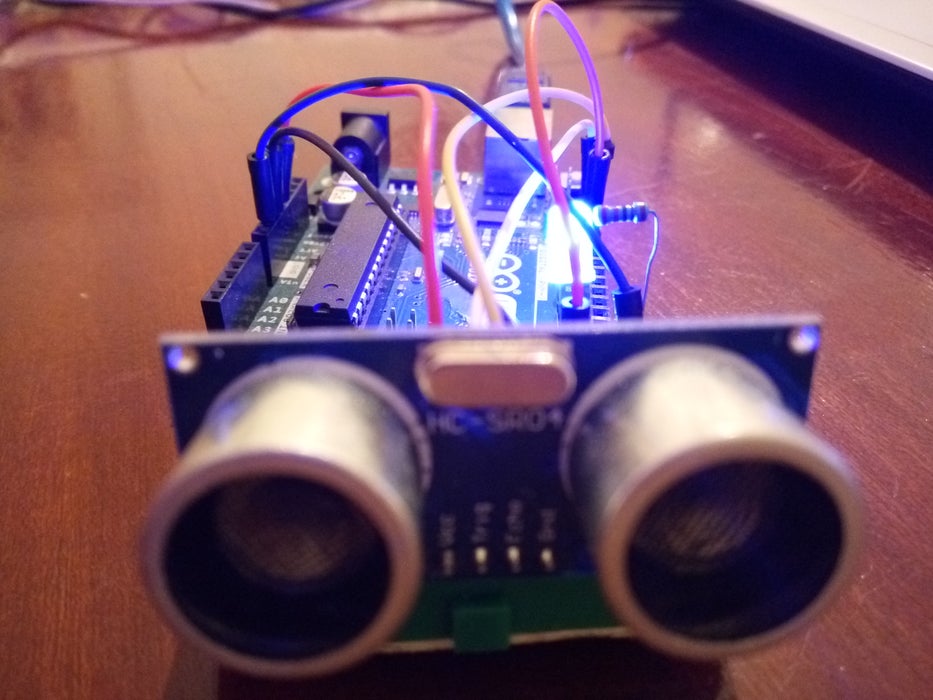
**3.Ultrasonic Sensor**

# https://www.instructables.com/Simple-Project-With-the-Ultrasonic-Sensor-HC-SR04-/

# Simple Project With the Ultrasonic Sensor (HC-SR04) +LED -Arduino Tutoriel-

By [SaidiDikra](https://www.instructables.com/member/SaidiDikra/) in [Circuits](https://www.instructables.com/circuits/)[Arduino](https://www.instructables.com/circuits/arduino/projects/)

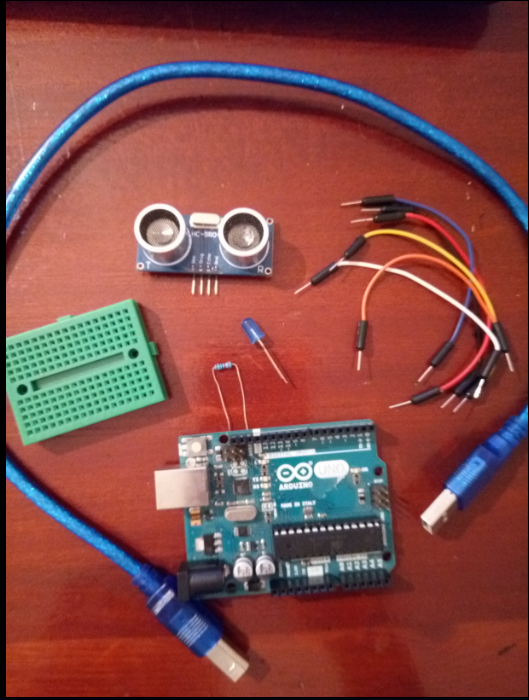
## Introduction: Simple Project With the Ultrasonic Sensor (HC-SR04) +LED -Arduino Tutoriel-

[](https://content.instructables.com/F0G/VBG3/JC0UCMVC/F0GVBG3JC0UCMVC.jpg?auto=webp&frame=1&width=1024&height=1024&fit=bounds&md=f79179c56e0126fe82fdf19129e000fc)

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit.

This is a simple example of using the ultrasonic sensor (HC-SR04) in Arduino where we will turn on a led with the variation of distance and print the distance from an object to the serial monitor.

## Step 1: All the Necessary Components

[](https://content.instructables.com/F2N/YOUA/JC0UCMUP/F2NYOUAJC0UCMUP.png?auto=webp&frame=1&fit=bounds&md=79fc86b442d9e30b0326dec56940c588)

1. Arduino Uno

2. Ultrasonic Sensor (HC-SR04).

3. Mini-BreadBoard

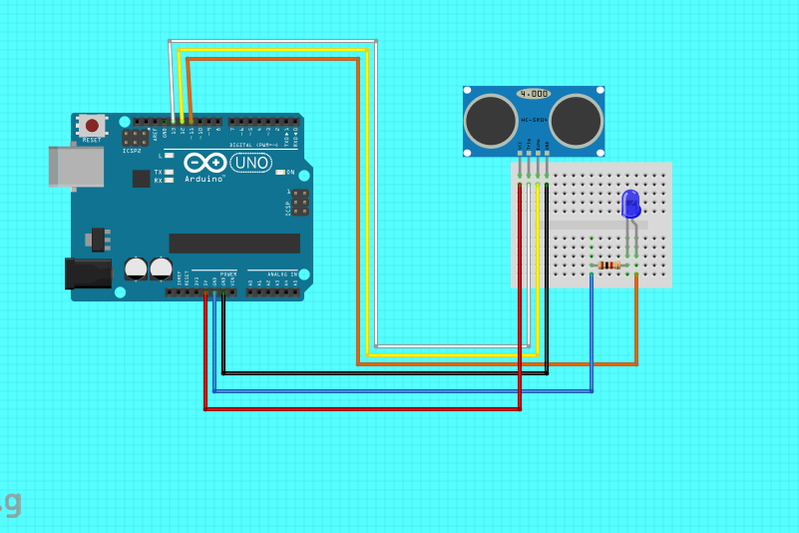
4. 1 kohm Resistor.

5. Jumpers.

6. Blue LED.

Add TipAsk QuestionCommentDownload

## Step 2: Connect the Components

[](https://content.instructables.com/FIZ/E1MF/JC0UCLKB/FIZE1MFJC0UCLKB.png?auto=webp&frame=1&width=1024&fit=bounds&md=5cc29e38f4a7858a76a6946ba1ce8f24)

The connection of the components is very easy it is only necessary to follow the following pictures.

## Step 3: Write Your Code

#define trigPin 13

#define echoPin 12

#define led 11

void setup()

{ Serial.begin (9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(led, OUTPUT);

}

void loop()

{ long duration, distance;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration/2) / 29.1;

if (distance < 10)

{ digitalWrite(led,HIGH);

}

else {

digitalWrite(led,LOW);

}

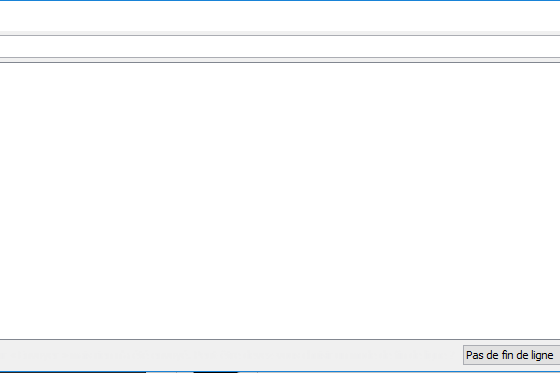
Serial.print(distance);

Serial.println(" cm");

delay(500);

}

## Step 4: Upload and Run .

[](https://content.instructables.com/FX0/AO89/JC0UCNQB/FX0AO89JC0UCNQB.png?auto=webp&frame=1&width=1024&fit=bounds&md=5251a51e375804b3e1f60bbe93b5ae5d)

You can change the value of distance by replacing 10 in the condition if (distance <=10) by the desired value.

1. Thingspeak – IOT data collector and Analyser

<https://www.youtube.com/watch?v=JKpRBg8YOdY&ab_channel=IndustrialITandAutomation>

1. <https://www.instructables.com/How-to-use-a-Buzzer-Arduino-Tutorial/>
2. <https://www.instructables.com/Traffic-Light-Controller-1/>
3. https://www.instructables.com/Fading-LED-With-Analog-Output/