Internet of Things

The Internet of Things (IoT) describes the phenomenon of everyday devices connecting to the Internet through tiny embedded sensors and computing power

We're entering a new era of computing technology that many are calling the Internet of Things (IoT). Machine to machine, machine to infrastructure, machine to environment, the Internet of Everything, the Internet of Intelligent Things, intelligent systems—call it what you want, but it's happening, and its potential is huge. We see the IoT as billions of smart, connected "things" that will encompass every aspect of our lives, and its foundation is the intelligence that embedded processing provides. The IoT is comprised of smart machines interacting and communicating with other machines, objects, environments and infrastructures

Applications of IOT

- Machine-to-machine communication
- Machine-to-infrastructure communication
- Telehealth: remote or real-time pervasive monitoring of patients, diagnosis and drug delivery
- Continuous monitoring of, and firmware upgrades for, vehicles
- Asset tracking of goods on the move
- Automatic traffic management
- Remote security and control
- Environmental monitoring and control
- Home and industrial building automation
- "Smart" applications, including cities, water, agriculture, buildings, grid, meters, broadband, cars, appliances, tags, animal farming and the environment, to name a few

Introduction to Arduino

The Arduino board is a small microcontroller board, which is a small circuit (the board) that contains a whole computer on a small chip (the microcontroller)

Arduino is composed of two major parts: the Arduino board, which is the piece of hardware you work on when you build your objects; and the Arduino IDE, the piece of software you run on your computer. You use the IDE to create a sketch (a little computer program) that you upload to the Arduino board. The sketch tells the board what to do.

Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. It has the specific advantages such as

• **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

• **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

• **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

• **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

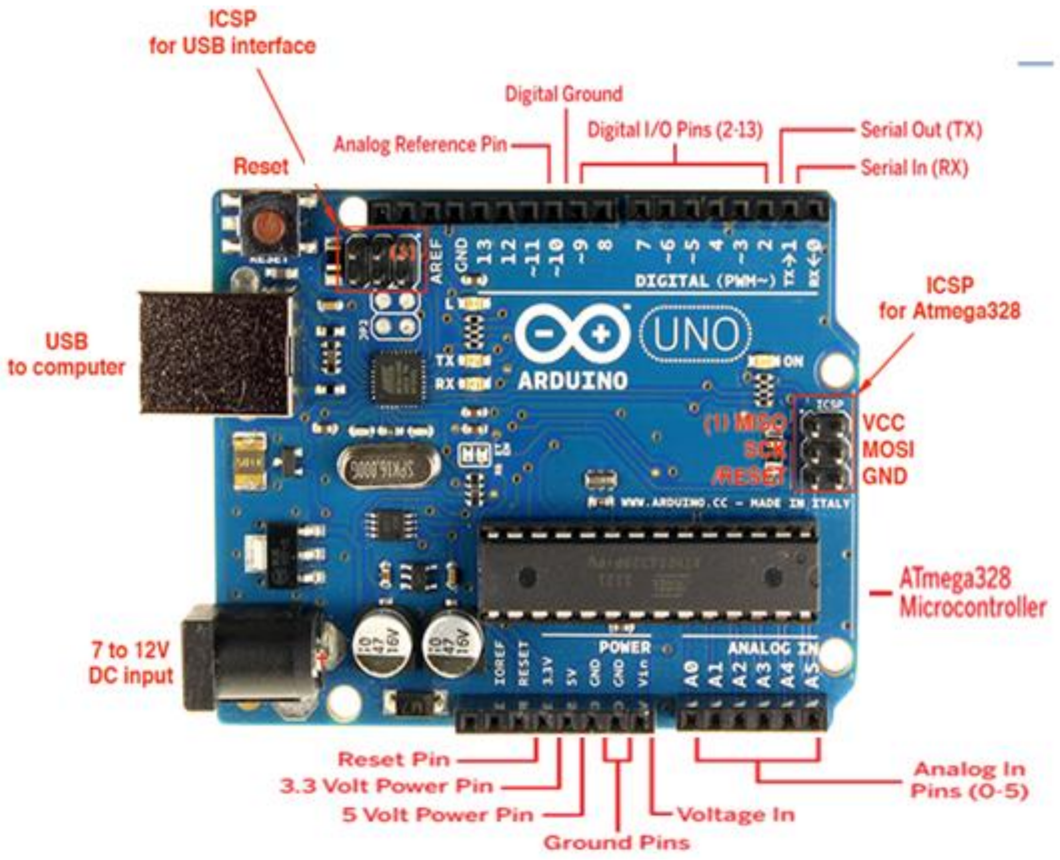
• **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Installing Arduino on Your Computer

To program the Arduino board, you must first download the development environment (the IDE) from here: www.arduino.cc/en/Main/Software. Choose the right version for your operating system. Download the file and double-click on it to open it it; on Windows or Linux, this creates a folder named arduino-[version], such as arduino-1.0. Drag the folder to wherever you want it: your desktop, your Program Files folder (on Windows), etc. On the Mac, double-clicking it will open a disk image with an Arduino application (drag it to your Applications folder). Now whenever you want to run the Arduino IDE, you'll open up the arduino (Windows and Linux) or Applications folder (Mac), and double-click the

Arduino icon. Don't do this yet, though; there is one more step.

1. The Arduino Development board



1.1.1 Digital Pins

The digital pins on an Arduino board can be used for general purpose input and output via the pinMode(), digitalRead(), and digitalWrite() commands. Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40 mA.

Some other specific functions of pins are listed below:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.
- **BT Reset: 7.** (Arduino BT-only) Connected to the reset line of the bluetooth module.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

1.1.2 Analog Pins

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analogRead() function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

- **I^{2}C: 4 (SDA) and 5 (SCL).** Support I^{2}C (TWI) communication using the Wire library

1.1.3 Power Pins

- **VIN**: The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board FTDI chip.
- **GND.** Ground pins.

1.1.4 Other Pins

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

General Purpose Input / Output Interfacing

Aim

Turns on an LED on for one second, then off for one second, repeatedly

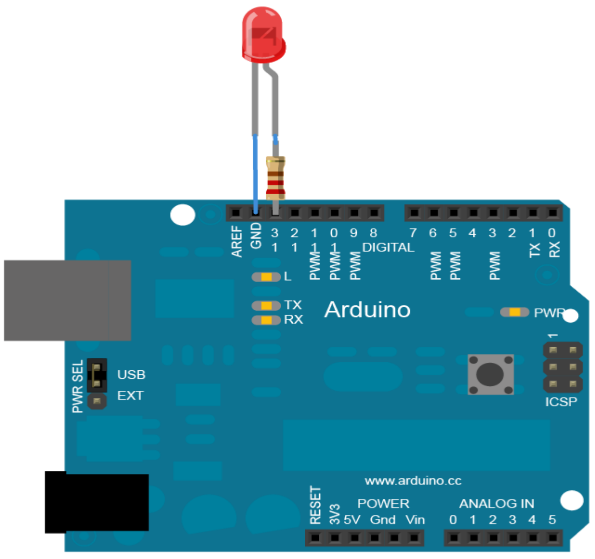
Hardware Required

- Arduino Board
- LEDs

Pin Configuration

Arduino	**LED**
Pin 13 (through resistor)	+ (Anode)
GND	- (Cathode)

Circuit Diagram



Code

```
// Pin 13 has an LED connected on most Arduino boards

int led = 13;

// the setup routine runs once when you press reset:
void setup()
```

```
{
// initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Aim

Traffic Signal Simulation (This program is example for digital read and digital write)

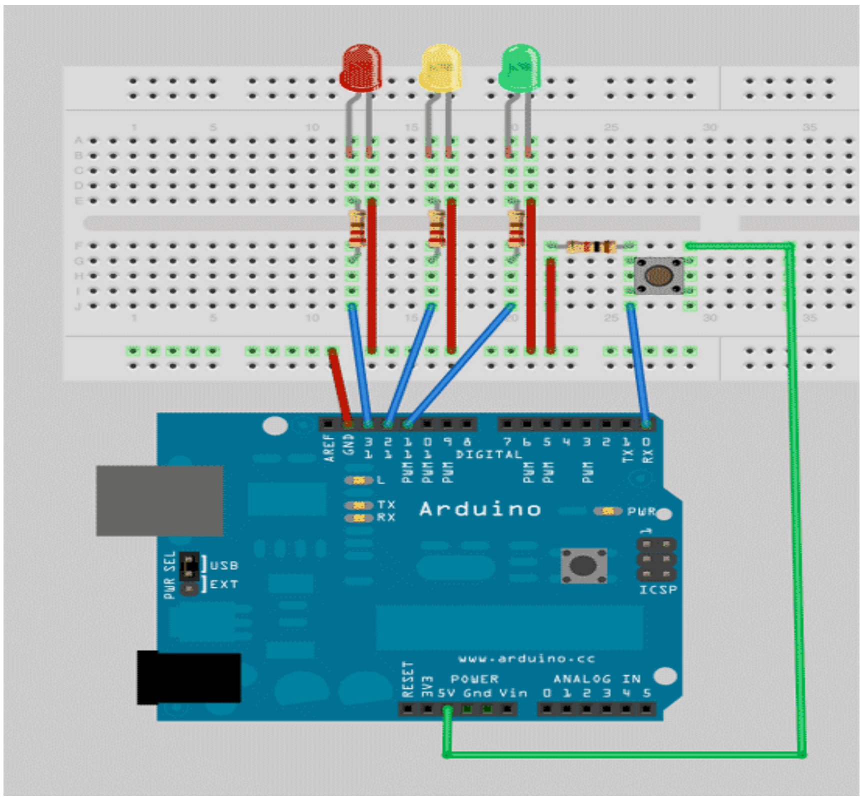
Hardware Required

- **Arduino Board**
- **LEDs**
- **Resistors – 10 K**
- **Bread board**

Pin Configuration

Arduino	Switch	LED
Pin 13 (through resistor)		+ (Anode) of RED LED
Pin 12 (through resistor)		+ (Anode) of YELLOW LED
Pin 11 (through resistor)		+ (Anode) of GREEN LED
GND		- (Cathode of all 3 LEDs)
5V	Switch Input	
D0	Switch output	

Circuit Diagram



Code

// defining variables so that we can address the lights by name rather than a number

```
int red = 13;
int yellow = 12;
int green = 11;
```

// add the setup function, where'll we define the red, yellow and green LEDs to be output mode

```
void setup()
{
pinMode(red,OUTPUT);
pinMode(yellow,OUTPUT);
pinMode(green,OUTPUT);
}

void loop()
{
changeLights();
delay(15000);
}

void changeLights(){
// green off, yellow for 3 seconds
digitalWrite(green,HIGH);
digitalWrite(yellow,LOW);
delay(3000);
// turn off yellow, then turn red on for 5 seconds
digitalWrite(yellow,LOW);
digitalWrite(red,HIGH);
delay(5000);

// red and yellow on for 2 seconds (red is already on though)
digitalWrite(yellow,HIGH);
delay(2000);

// turn off red and yellow, then turn on green
digitalWrite(yellow,LOW);
digitalWrite(red,LOW);
digitalWrite(green,HIGH);
}
```

Aim

Push button to control LED . This is an example for digital read and analog write

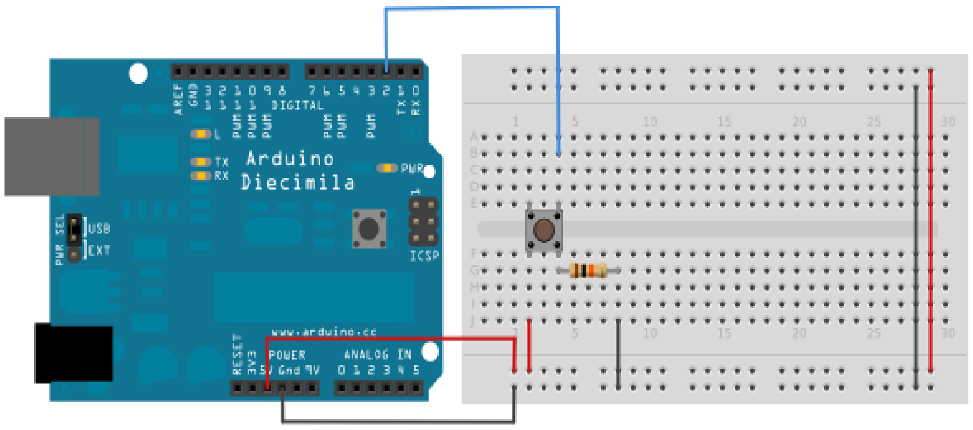
Hardware Required

- Arduino Board
- momentary button or switch
- 10K ohm resistor
- Breadboard

Pin Configuration

Arduino	Switch
VCC	Switch Input
GND	GND (pulled by resistor)
D2	Switch Output

Circuit Diagram



Code

```
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;     // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

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

```
 // check if the pushbutton is pressed.
 // if it is, the buttonState is HIGH:
 if (buttonState == HIGH) {
  // turn LED on:
  digitalWrite(ledPin, HIGH);
 }
 else {
  // turn LED off:
  digitalWrite(ledPin, LOW);
 }
}
```

Aim

Control brightness of led using potentiometer (Analog read and analog write)

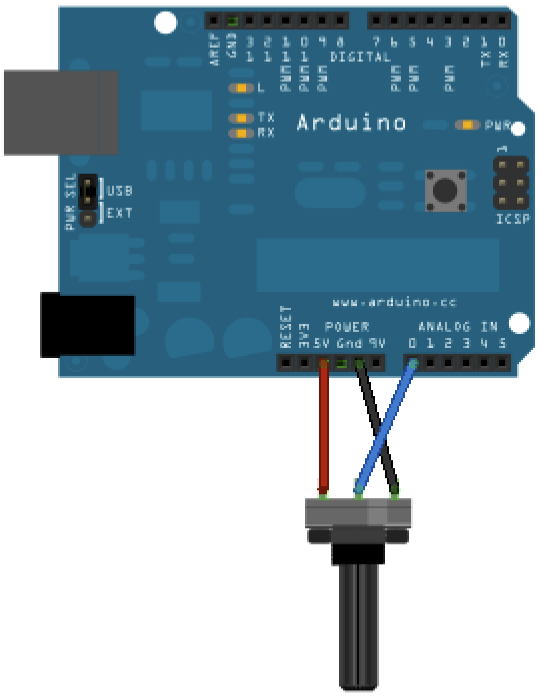
Hardware Required

- **Arduino Board**
- **Potentiometer**

  By turning the shaft of the potentiometer, you change the amount of resistance on either side of the wiper which is connected to the center pin of the potentiometer. This changes the voltage at the center pin. When the resistance between the center and the side connected to 5 volts is close to zero (and the resistance on the other side is close to 10 kilohms), the voltage at the center pin nears 5 volts. When the resistances are reversed, the voltage at the center pin nears 0 volts, or ground. This voltage is the analog voltage that you're reading as an input.

  The Arduino has a circuit inside called an analog-to-digital converter that reads this changing voltage and converts it to a number between 0 and 1023. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and the input value is 0. When the shaft is turned all the way in the opposite direction, there are 5 volts going to the pin and the input value is 1023. In between,analogRead() returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

Circuit Diagram



Pin Configuration

Arduino	Switch
VCC	Switch Input
GND	GND (pulled by resistor)
D2	Switch Output

Code

```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

Sensor Interfacing

Aim

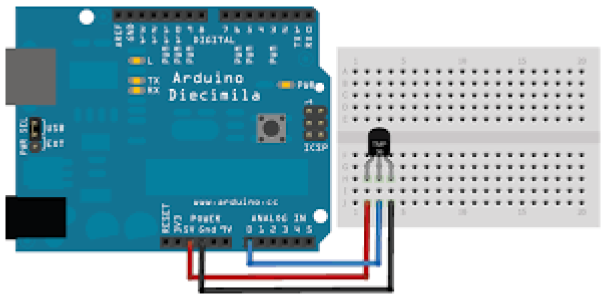
Read the temperature and print it in serial port

Hardware Required

- **Arduino Board**
- **Temperature Sensor (LM35 temperature sensor)**

Arduino	**LM35**
5V	VCC
A0	Middle pin
GND	GND

Circuit Diagram



Code

```
int outputpin= 0;

//this sets the ground pin to LOW and the input voltage pin to high

void setup()

{

Serial.begin(9600);

}


//main loop

void loop()

{

int rawvoltage= analogRead(outputpin);
float millivolts= (rawvoltage/1024.0) * 5000;

float celsius= millivolts/10;

Serial.print(celsius);

Serial.print(" degrees Celsius, ");


Serial.print((celsius * 9)/5 + 32);

Serial.println(" degrees Fahrenheit");

delay(1000);


}
```

Aim

Reading the state of a photo resistor

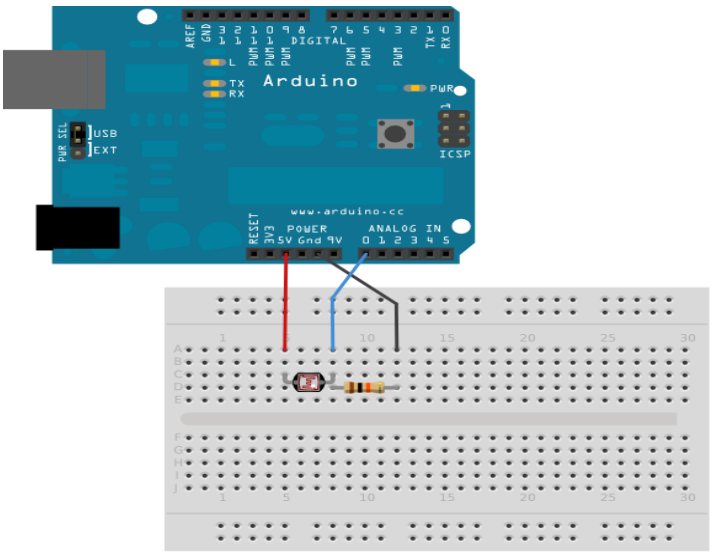
Hardware Required

- Arduino Board
- Photocell
- 10K ohm resistor
- Bread board

☐ **Attach one leg of LDR to 5V and another leg attach 110K register with that leg of LDR connected to A0**

☐ **Attach another leg of register to the ground**

Circuit Diagram



Code

```
const int sensorMin = 0;      // sensor minimum, discovered through experiment
const int sensorMax = 600;    // sensor maximum, discovered through experiment

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
}

void loop() {
  // read the sensor:
  int sensorReading = analogRead(A0);
  // map the sensor range to a range of four options:
  int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

  // do something different depending on the
  // range value:
  switch (range) {
  case 0:    // your hand is on the sensor
    Serial.println("dark");
    break;
  case 1:    // your hand is close to the sensor
    Serial.println("dim");
    break;
  case 2:    // your hand is a few inches from the sensor
    Serial.println("medium");
    break;
  case 3:    // your hand is nowhere near the sensor
    Serial.println("bright");
    break;
  }
  delay(1);        // delay in between reads for stability
}
```

Aim

Measuring the distance from obstacle (Proximity sensor)

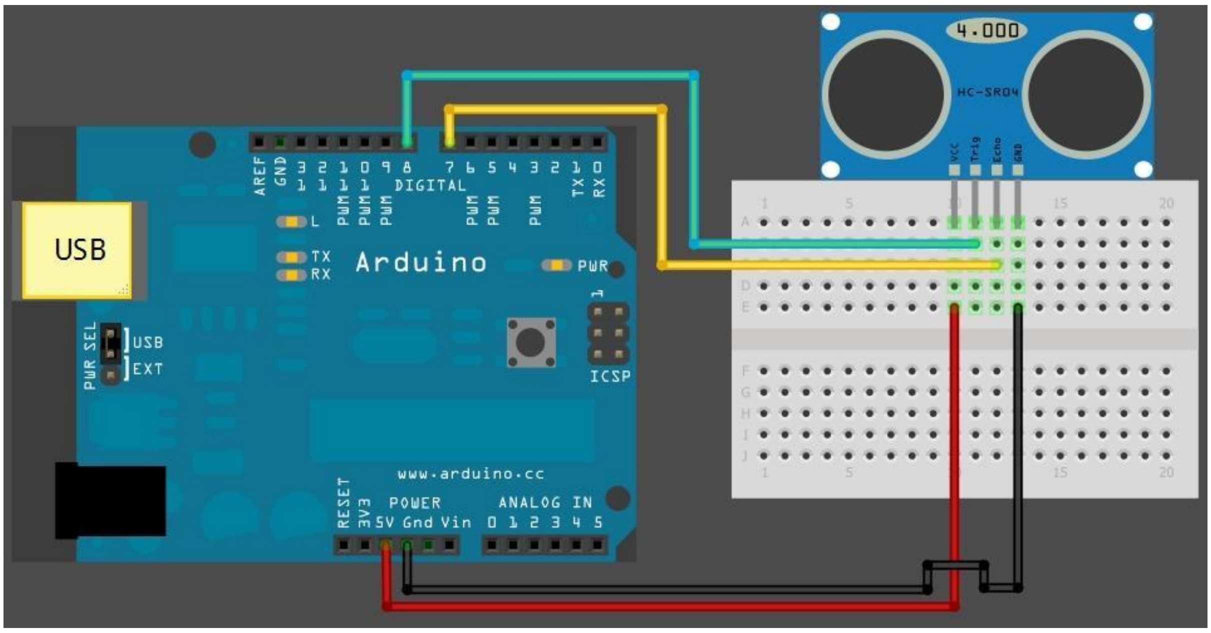
Hardware Required

- **HC-SR04 Ultrasonic Sensor**
  It is a very affordable proximity/distance sensor that has been used mainly for object avoidance in various robotics projects
- **Arduino Board**

Pin Configuration

Arduino	Ultrsonic Sensor
5V	VCC
GND	GND
Pin 7	Triger
Pin 8	Echo

Circuit Diagram



Code

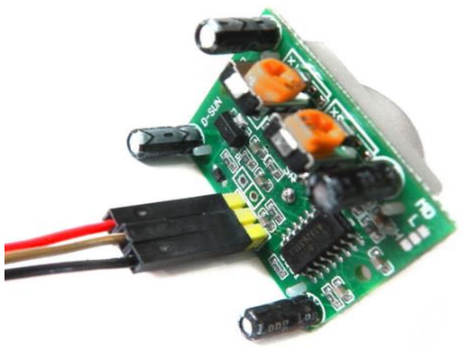
```
#define echoPin 7 // Echo Pin
#define trigPin 8 // Trigger Pin
#define LEDPin 13 // Onboard LED

int maximumRange = 200; // Maximum range needed
int minimumRange = 0; // Minimum range needed
long duration, distance; // Duration used to calculate distance

void setup() {
 Serial.begin (9600);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 pinMode(LEDPin, OUTPUT); // Use LED indicator (if required)
}

void loop() {
/* The following trigPin/echoPin cycle is used to determine the
 distance of the nearest object by bouncing soundwaves off of it. */
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);

 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);

 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH);

 //Calculate the distance (in cm) based on the speed of sound.
 distance = duration/58.2;

 if (distance >= maximumRange || distance <= minimumRange){
 /* Send a negative number to computer and Turn LED ON
 to indicate "out of range" */
 Serial.println("-1");
 digitalWrite(LEDPin, HIGH);
 }
 else {
 /* Send the distance to the computer using Serial protocol, and
 turn LED OFF to indicate successful reading. */
 Serial.println(distance);
 digitalWrite(LEDPin, LOW);
 }

 //Delay 50ms before next reading.
 delay(50);
}
```

Aim

Motion detection System

Hardware Requirement

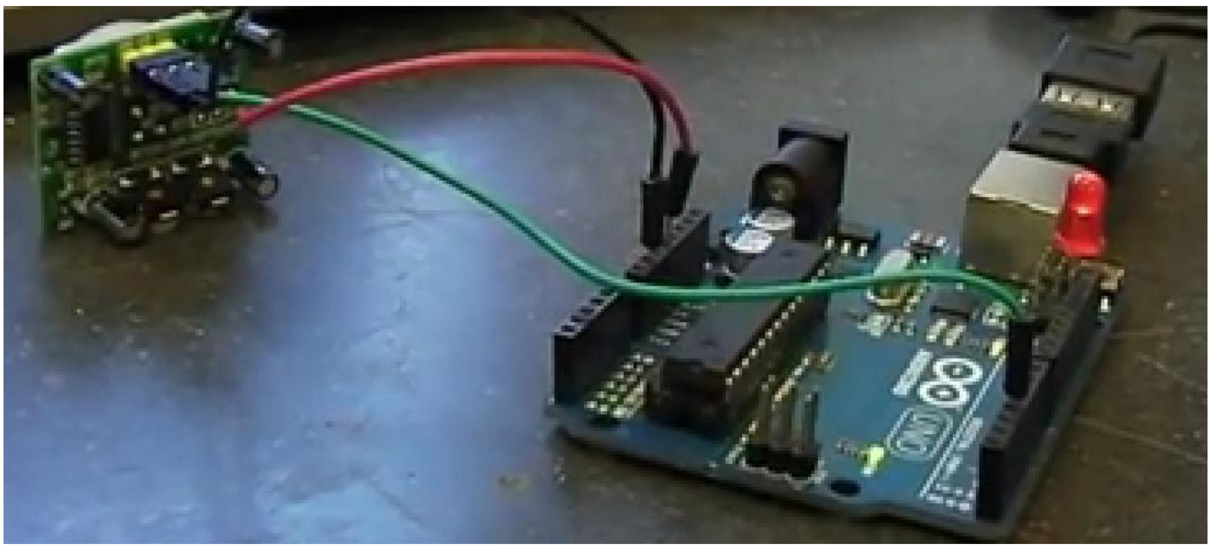
- **Arduino Board**
- **PIR Motion Sensor**



Pin Configuration

- ☐ **Connect positive leg of PIR to 5V**
- ☐ **Connect negative leg of PIR to GND**
- ☐ **Connect output pin of PIR to digital pin 3**

Circuit Diagram



Code

```
//the time we give the sensor to calibrate (10-60 secs according to the datasheet)
int calibrationTime = 30;

//the time when the sensor outputs a low impulse
long unsigned int lowIn;

//the amount of milliseconds the sensor has to be low
//before we assume all motion has stopped
long unsigned int pause = 5000;

boolean lockLow = true;
boolean takeLowTime;

int pirPin = 3;    //the digital pin connected to the PIR sensor's output
int ledPin = 13;



void setup(){
  Serial.begin(9600);
  pinMode(pirPin, INPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(pirPin, LOW);

  //give the sensor some time to calibrate
  Serial.print("calibrating sensor ");
    for(int i = 0; i < calibrationTime; i++){
     Serial.print(".");

      delay(1000);
      }
    Serial.println(" done");
    Serial.println("SENSOR ACTIVE");
    delay(50);
  }


//LOOP
void loop(){

if(digitalRead(pirPin) == HIGH){


  digitalWrite(ledPin, HIGH);
       if(lockLow){
 //makes sure we wait for a transition to LOW before any further output is made:
        lockLow = false;
        Serial.println("---");
        Serial.print("motion detected at ");
        Serial.print(millis()/1000);
        Serial.println(" sec");
        delay(50);
        }
        takeLowTime = true;
      }

    if(digitalRead(pirPin) == LOW){
      digitalWrite(ledPin, LOW);  //the led visualizes the sensors output pin state

      if(takeLowTime){
       lowIn = millis();          //save the time of the transition from high to LOW
       takeLowTime = false;       //make sure this is only done at the start of a LOW phase
       }
      //if the sensor is low for more than the given pause,
      //we assume that no more motion is going to happen
      if(!lockLow && millis() - lowIn > pause){
          //makes sure this block of code is only executed again after
          //a new motion sequence has been detected
          lockLow = true;
          Serial.print("motion ended at ");      //output
          Serial.print((millis() - pause)/1000);
          Serial.println(" sec");
          delay(50);
          }
      }
  }
```

Aim

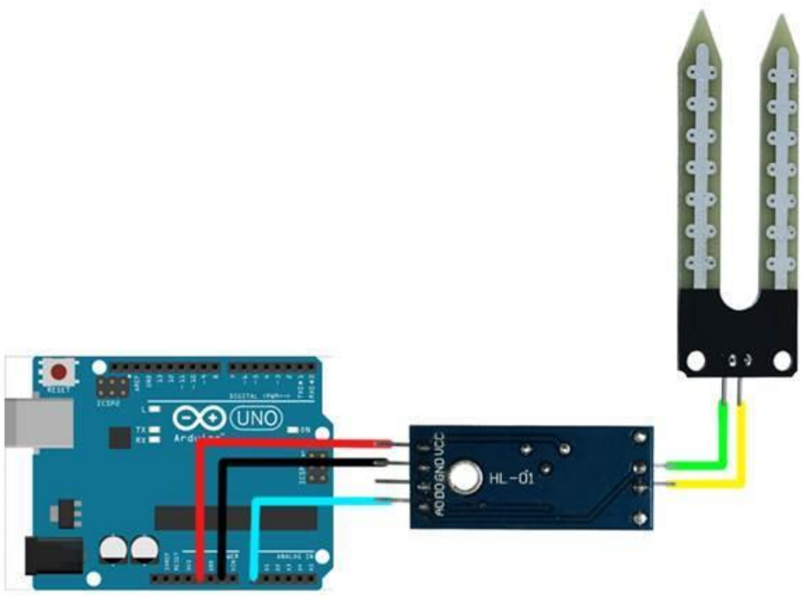
To find the Moisture level in the sand

Hardware Requirement

- **Arduino Board**
- **Moisture Sensor**

Arduino	**Moisture Sensor**
5v	**VCC**
GND	**GND**
A0	**A0**

Circuit Diagram



Code

```
int sensorPin = A0;    // select the input pin for the potentiometer

int sensorValue = 0;  // variable to store the value coming from the sensor


void setup() {

  Serial.begin(9600);

}
```

```
void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  Serial.println (sensorValue);
  delay (1000);
}
```

Aim

Reverse Alarm System using Proximity Sensor

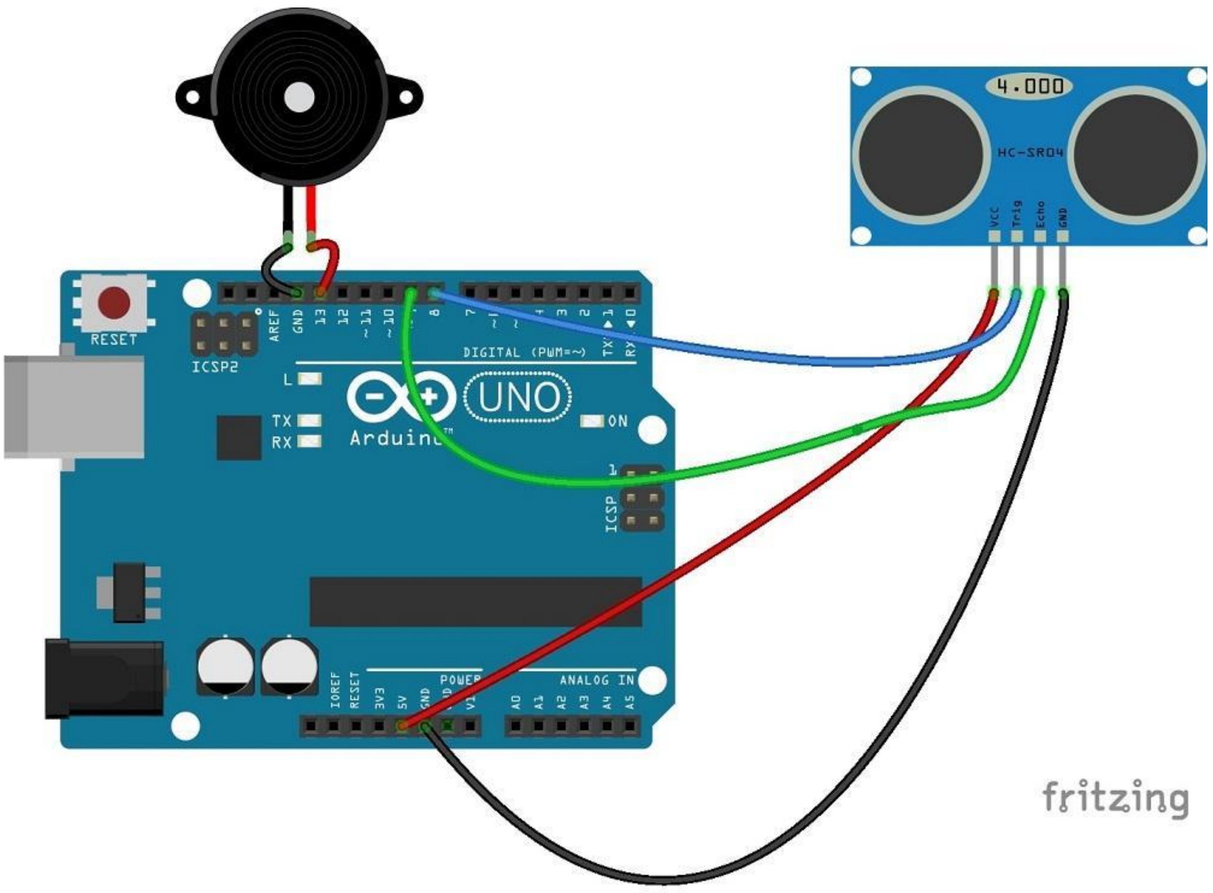
Hardware Requirement

- The HC-SR04 Ultrasonic Sensor

It is a very affordable proximity/distance sensor that has been used mainly for object avoidance in various robotics projects . It essentially gives your Arduino eyes / spacial awareness and can prevent your robot from crashing or falling off a table. It has also been used in turret applications, water level sensing, and even as a parking sensor. This simple project will use the HC-SR04 sensor with an Arduino and a Processing sketch to provide a neat little interactive display on your computer screen.

- Arduino Board
- Buzzer

Circuit Diagram



Code

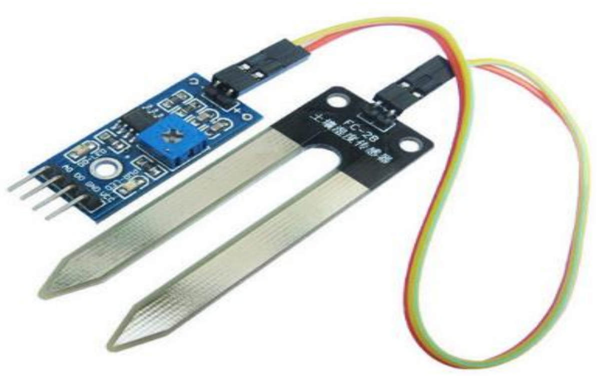
```
const int trigPin = 3;

const int echoPin = 2;

const int buzzerPin = 4;


void setup() {

  Serial.begin(9600);

  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

  pinMode(buzzerPin, OUTPUT);

   }

void loop()

{

  // establish variables for duration of the ping,

  // and the distance result in inches and centimeters:

  long duration, inches, cm;


  // The sensor is triggered by a HIGH pulse of 10 or more microseconds.

  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:


  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);


  // Read the signal from the sensor: a HIGH pulse whose

  // duration is the time (in microseconds) from the sending
  }


  // of the ping to the reception of its echo off of an object.


  duration = pulseIn(echoPin, HIGH);
  // convert the time into a distance
  inches = microsecondsToInches(duration);
  cm = microsecondsToCentimeters(duration);


  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.println("cm");


  if(inches<20)
  {
    digitalWrite(buzzerPin, HIGH);
  }
  else
    digitalWrite(buzzerPin, LOW);


 delay(1000);



}
long microsecondsToInches(long microseconds)
{
return microseconds / 74 / 2;
  }


long microsecondsToCentimeters(long microseconds)
```

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

To design and implement ON/OFF the light based on human presence in the room using PIR sensor and LED

```
int pirpin=2;
int ledpin=13;
int ct=30;
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
pinMode(pirpin,INPUT);
pinMode(ledpin,OUTPUT);
digitalWrite(pirpin,LOW);

}

void loop() {
  // put your main code here, to run repeatedly:
if(digitalRead(pirpin)==HIGH)
{
  digitalWrite(ledpin,HIGH);
}
delay(5000);
if(digitalRead(pirpin)==LOW)
{
  digitalWrite(ledpin,LOW);
}
delay(1000);
}
```

To design and implement smart irrigation system using Soil Moisture sensor and Servo Motor

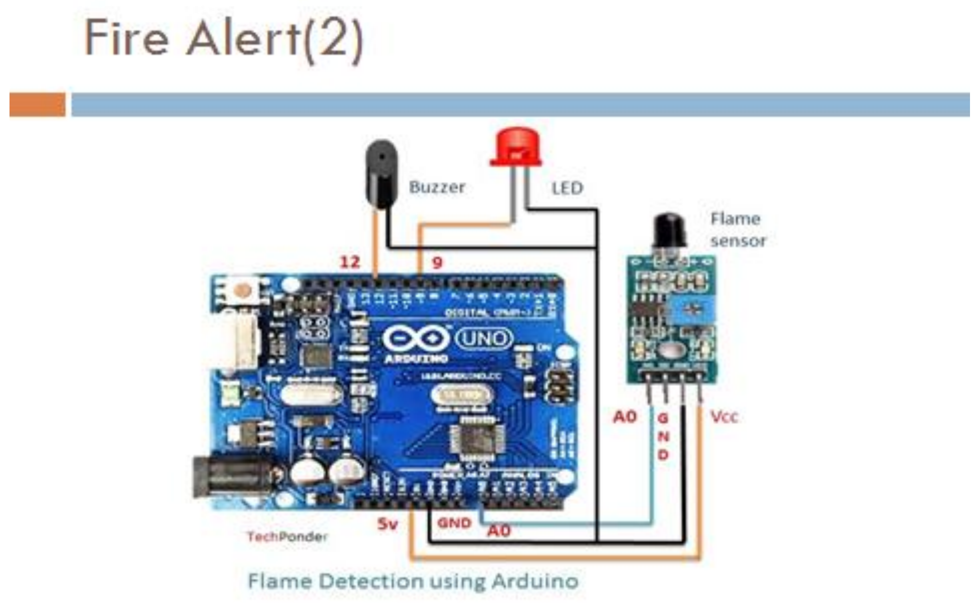


- ☐ **Moisture sensor VCC to Arduino 5V**
- ☐ **Moisture sensor GND to Arduino GND**
- ☐ **Moisture sensor A0 to Arduino A0**

- ☐ **Servo motor VCC to Arduino 5V**
- ☐ **Servo motor GND to Arduino GND**
- ☐ **Servo Motor Signal to Arduino digital pin 9**

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position



int sensorPin = A0;    // select the input pin for the potentiometer
int sensorValue = 0;  // variable to store the value coming from the sensor
void setup() {
 myservo.attach(9);  // attaches the servo on pin 9 to the servo object
Serial.begin(9600);
}
void loop() {
  // read the value from the sensor:
sensorValue = analogRead(sensorPin);
Serial.print(sensorValue);
if(sensorValue>500)
{
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
delay (1000);
}
```

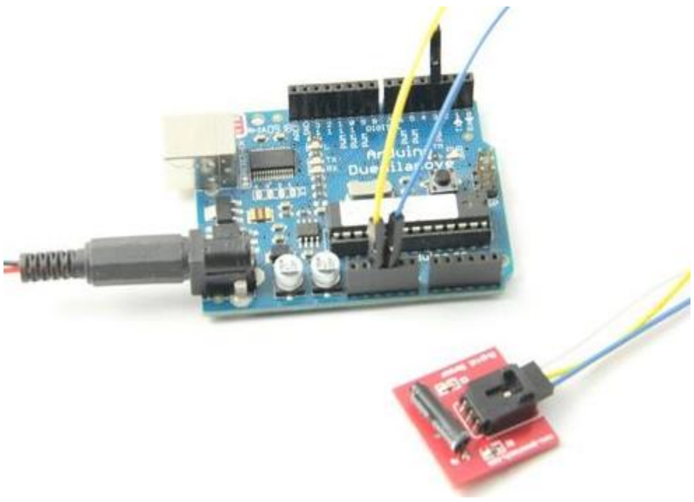
To design and implement Fire alarm system using flame sensor and buzzer



```
#include<SoftwareSerial.h>
#include <NewPing.h>

int sensorPin = A0; // select the input pin for the LDR
int sensorValue = 0; // variable to store the value coming from the sensor
int led = 9; // Output pin for LED
int buzzer = 12; // Output pin for Buzzer
void setup() {
// declare the ledPin and buzzer as an OUTPUT:
pinMode(led, OUTPUT);
pinMode(buzzer,OUTPUT);
Serial.begin(9600);
}
void loop()
{
```

```
sensorValue = analogRead(sensorPin);
Serial.println(sensorValue);
if (sensorValue < 100)
{
Serial.println("Fire Detected");
Serial.println("LED on");
digitalWrite(led,HIGH);
tone(buzzer,2000,1000);
delay(1000);
}
digitalWrite(led,LOW);
digitalWrite(buzzer,LOW);
delay(sensorValue);
}
```

To design and demonstrate the usage of Tilt sensor



```
int ledPin = 13;              // Connect LED to pin 13
int switcher = 3;               // Connect Tilt sensor to Pin3
 void setup()
{
 pinMode(ledPin, OUTPUT);     // Set digital pin 13 to output mode
 pinMode(switcher, INPUT);      // Set digital pin 3 to input mode
}
```

```
void loop()
{

  if(digitalRead(switcher)==HIGH) //Read sensor value
    {
      digitalWrite(ledPin, HIGH);   // Turn on LED when the sensor is tilted
      delay(300);
    }
  else
    {
      digitalWrite(ledPin, LOW);    // Turn off LED when the sensor is not triggered
    }
}
```