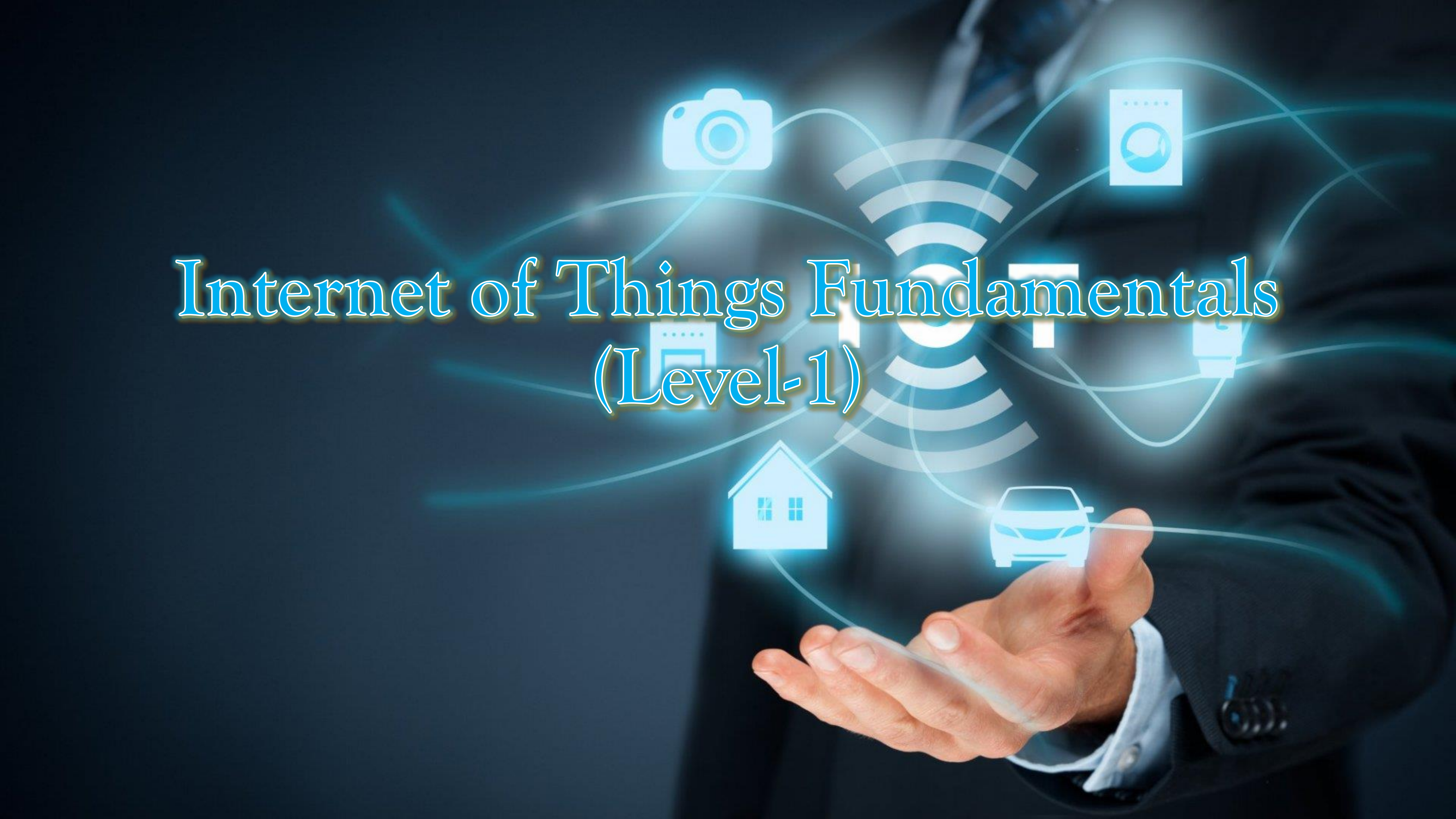
Internet of Things Fundamentals (Level-1)



Internet of Things Fundamentals (Level-1)

Course Code:	--	Credits:	--
		CIE Marks:	90
Exam Hours:	03	SEE Marks:	60

Course Learning Outcome (CLOs): After Completing this course successfully, the student will be able to…

CLO	Learning Outcome
CLO 1	Define and describe IoT, its significance, and common use cases in real-world applications.
CLO 2	Demonstrate proficiency in using IoT communication protocols (HTTP, MQTT, CoAP) for data exchange.
CLO 3	Interface sensors and actuators with microcontrollers (e.g., Arduino) for IoT applications.
CLO 4	Collect, analyze, and visualize IoT data.
CLO 5	Implement IoT projects that involve cloud integration for remote monitoring and control.
CLO 6	Troubleshoot and debug IoT systems, addressing performance and integration issues.
CLO 7	

Summary of Course Content

Serial No.	SUMMARY OF COURSE CONTENT	Hours	CLOs
1	Introduction to IoT, its significance, and real-world use cases	3	CLO 1
2	IoT communication protocols (HTTP, MQTT, CoAP)	6	CLO 2
3	Interfacing sensors and actuators with microcontrollers (e.g., Arduino)	8	CLO 3
4	Data collection, analysis, and visualization in IoT	5	CLO 4
5	IoT project implementation with cloud integration	10	CLO 5
6	Troubleshooting and debugging IoT systems	4	CLO 6
7	IoT system security and vulnerability protection	4	CLO 7

Textbooks:

- **"Internet of Things: A Hands-On Approach"** by Arshdeep Bahga, Vijay Madisetti
- **"Getting Started with Arduino"** by Massimo Banzi

Additional References:

- **"Arduino Cookbook"** by Michael Margolis
- **"Building the Internet of Things"** by Maciej Kran

Assessment Pattern

CIE- Continuous Internal Evaluation (30 Marks)

Bloom's Category Marks (out of 90)	Lab Participation (10)	Assignments (10)	Quizzes (10)
Remember			05
Understand	05		
Apply		05	
Analyze	05		
Evaluate		05	05
Create			

SEE- Semester End Examination (20 Marks)

Bloom's Category	Test
Remember	
Understand	
Apply	10
Analyze	
Evaluate	
Create	10

Course Plan

Week	Topics	Teaching-Learning Strategy(s)	Class Hour	Practice Hour	Assessment Strategy(s)	Mapping with CLO
01	Introduction to IoT: Ability to define IoT and explain its significance in modern systems.	Lecture, Demonstration, Interactive Q&A	5h	2h	Participation, Short Quiz	CLO 1: Define and describe IoT and its significance
02	IoT Communication Protocols: Ability to identify and work with IoT protocols for data communication.	Lecture, Hands-on Exercises with HTTP, MQTT, CoAP	5h	4h	Practical Exercise, Written Test	CLO 2: Demonstrate proficiency in using IoT protocols
03	IoT Hardware Components: Ability to choose hardware components for IoT projects.	Demonstration of sensors, actuators, and microcontrollers	5h	4h	Lab Participation, Practical Assignment	CLO 3: Interface sensors with microcontrollers
04	Arduino Basics for IoT: Ability to write basic Arduino programs and interface with sensors.	Hands-on sessions with Arduino, Coding tutorials	5h	4h	Lab Performance, Coding Test	CLO 3: Interface sensors with microcontrollers
05	Sensors and Actuators for IoT: Ability to select and connect sensors and actuators to microcontrollers.	Practical exercises with sensors and actuators	5h	4h	Hands-on Assignment, Peer Review	CLO 3: Interface sensors with microcontrollers
06	IoT Data Collection and Basic Analysis: Ability to collect and process IoT data for analysis and visualization.	Hands-on sessions on data collection, basic analysis, and visualization tools	5h	3h	Data Collection Exercise, Report Submission	CLO 4: Collect, analyze, and visualize IoT data
07	Project: Smart Temperature Monitor: Ability to integrate sensors and microcontrollers for a basic IoT project.	Guided project work: Build a temperature monitoring system	5h	5h	Project Progress Evaluation, Lab Report	CLO 5: Implement IoT projects with cloud integration
08	IoT System Integration: Ability to integrate different IoT components into a functioning system.	System integration workshop, IoT component interfacing	5h	3h	Integration Exercise, Group Discussion	CLO 5: Implement IoT projects with cloud integration
09	IoT Cloud Platforms: Introduction: Ability to connect IoT devices to cloud platforms for remote monitoring.	Introduction to IoT cloud platforms (AWS, Azure, Google IoT), hands-on work	5h	3h	Cloud Integration Exercise, Test on Cloud Tools	CLO 5: Implement IoT projects with cloud integration

Course Plan

Week	Topics	Teaching-Learning Strategy(s)	Class Hour	Practice Hour	Assessment Strategy(s)	Mapping with CLO
10	IoT Security Basics: Understanding IoT security risks and implementing basic safeguards.	Lecture, Hands-on exercises on encryption and security techniques	5h	3h	Security Challenge, Practical Application Test	CLO 7: Apply basic security techniques to IoT systems
11	IoT Project Integration: Ability to integrate IoT components with cloud platforms for remote control and monitoring.	Hands-on session to integrate sensors, microcontrollers, and cloud services	5h	4h	Practical Exercise, Integration Evaluation	CLO 5: Implement IoT projects with cloud integration
12	Project Work: Building a Smart Device: Ability to design and implement a smart IoT system with cloud connectivity.	Guided project work, Developing a complete smart device	5h	5h	Project Submission, Prototype Demonstration	CLO 5: Implement IoT projects with cloud integration
13	IoT System Testing and Debugging: Ability to debug and troubleshoot IoT systems.	Practical troubleshooting and testing sessions	5h	4h	Debugging Report, Lab Test	CLO 6: Troubleshoot and debug IoT systems
14	IoT Data Management: Ability to manage and structure large volumes of IoT data.	Introduction to data management, database integration, hands-on exercises	5h	3h	Data Management Assignment, Practical Test	CLO 4: Collect, analyze, and visualize IoT data
15	Final Project Development: Ability to implement a complete IoT project from start to finish.	Guided work on a final IoT project, including sensors, actuators, and cloud integration	5h	5h	Project Report, Final Project Evaluation	CLO 5: Implement IoT projects with cloud integration
16	Project Presentation: Ability to effectively communicate IoT project designs and results.	Presentation preparation and peer review	5h	1h	Project Presentation, Peer Feedback	CLO 6: Troubleshoot and debug IoT systems
17	Final Assessment: Evaluation of knowledge and practical skills in IoT development.	Written test, Practical assessment covering all IoT components	5h		Written Exam, Lab Performance Assessment	CLO 1-7: All CLOs

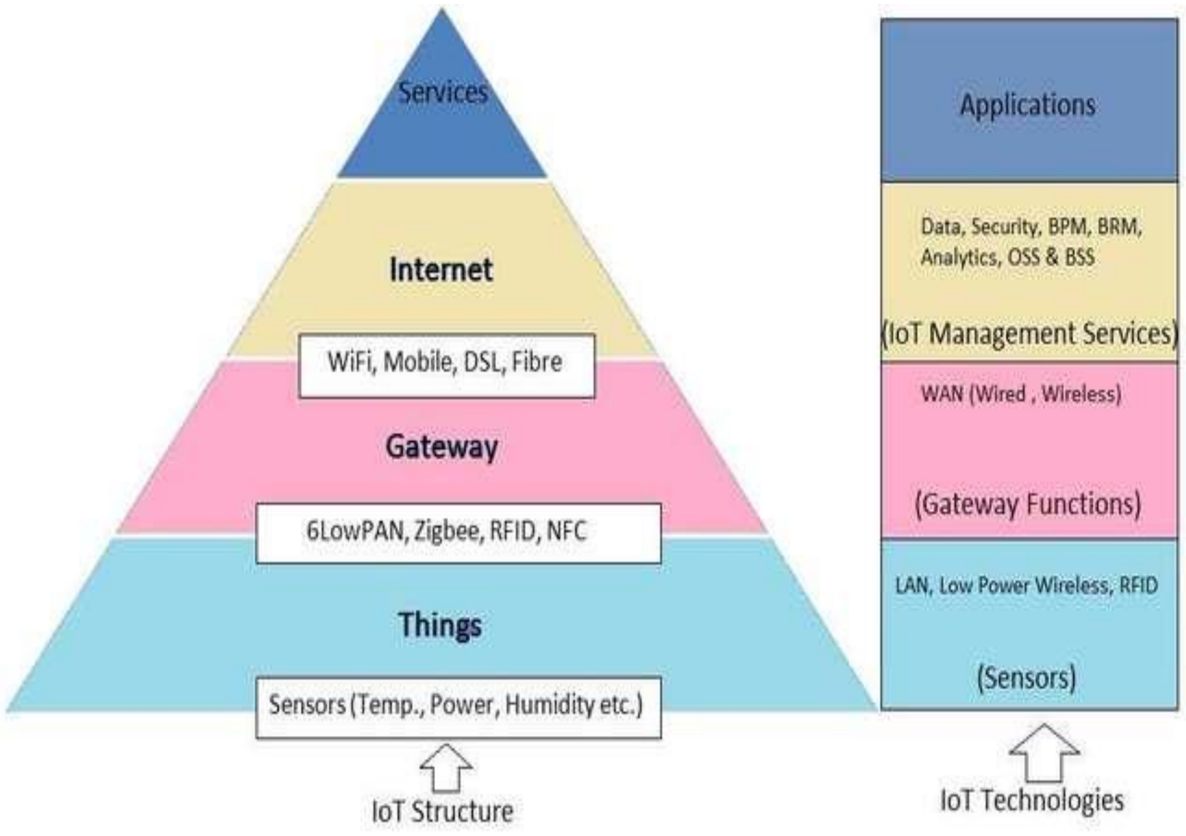
Internet of Things (IOT)

Introduction: IOT stands for "Internet of Things". The IOT is a name for the vast collection of "things" that are being networked together in the home and workplace (up to 20 billion by 2020 according to Gardner, a technology consulting firm).

Characteristics of the IOT

Networking	These IOT devices talk to one another (M2M communication) or to servers located in the local network or on the Internet. Being on the network allows the device the common ability to consume and produce data.
Sensing	IOT devices sense something about their environment.
Actuators	IOT devices that do something. Lock doors, beep, turn lights on, or turn the TV on

Communications in IoT


Services
Internet
WiFi, Mobile, DSL, Fibre
Gateway
6LowPAN, Zigbee, RFID, NFC
Things
Sensors (Temp., Power, Humidity etc.)
IoT Structure
Applications
Data, Security, BPM, BRM, Analytics, OSS & BSS
(IoT Management Services)
WAN (Wired , Wireless)
(Gateway Functions)
LAN, Low Power Wireless, RFID
(Sensors)
IoT Technologies

Communications are important to IOT projects. In fact, communications are core to the whole genre. There is a trade-off for IOT devices. The more complex the protocols and higher the data rates, the more powerful processor needed and the more electrical power the IOT device will consume.

TCP/IP base communications (think web servers; HTTP-based commutation (like REST servers); streams of data; UDP) provide the most flexibility and functionality at a cost of processor and electrical power.

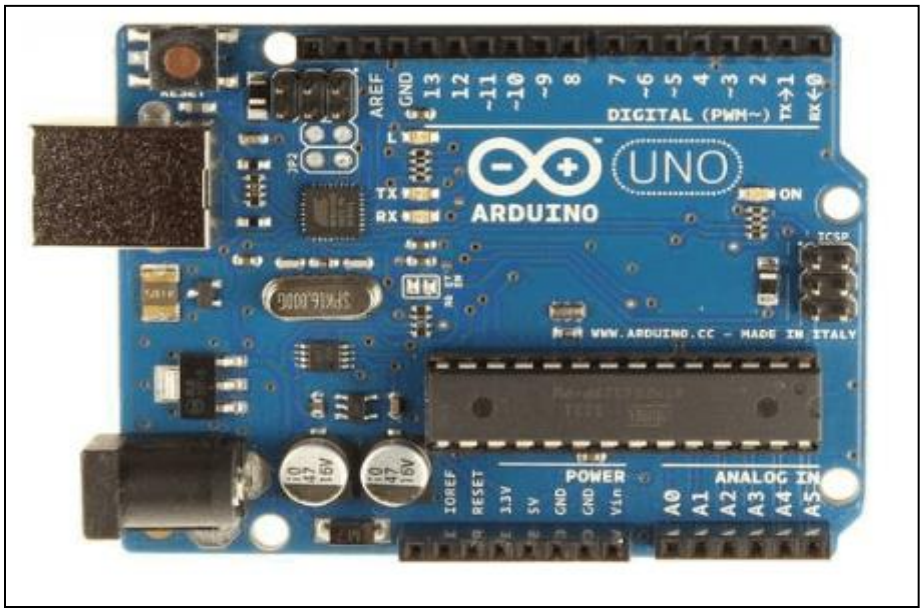
Low-power Bluetooth and Zigbee types of connections allow much lower power for connections with the corresponding decrease in bandwidth and functionality. IOT projects can be all over the map with requirements for communication flexibility and data bandwidth requirements.

Arduino in IoT

In IoT applications the Arduino is used to collect the data from the sensors/devices to send it to the internet and receives data for purpose of control of actuators.

Arduino Uno

Introduction: The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software.



Features of the Arduino

1. Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.

2. The board functions can be controlled by sending a set of instructions to the microcontroller on the board via Arduino IDE.
3. Arduino IDE uses a simplified version of C++, making it easier to learn to program.
4. Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

Arduino IDE

(Integrated Development Environment)

Introduction: The Arduino Software (IDE) is easy-to-use and is based on the Processing programming environment. The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

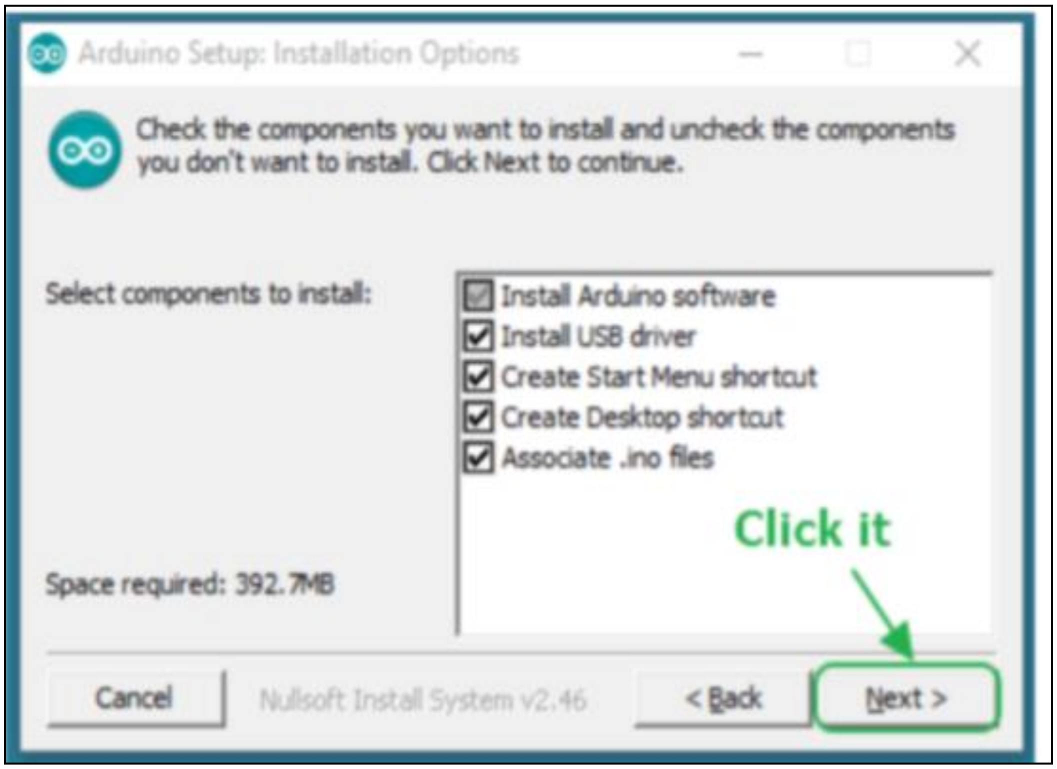
The Arduino Software (IDE) – contains:

- A text editor for writing code
- A message area
- A text consoles
- A toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

Installation of Arduino Software (IDE)

Step1: Downloading

- To install the Arduino software, download this page: http://arduino.cc/en/Main/Software and proceed with the installation by allowing the driver installation process.


Arduino Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

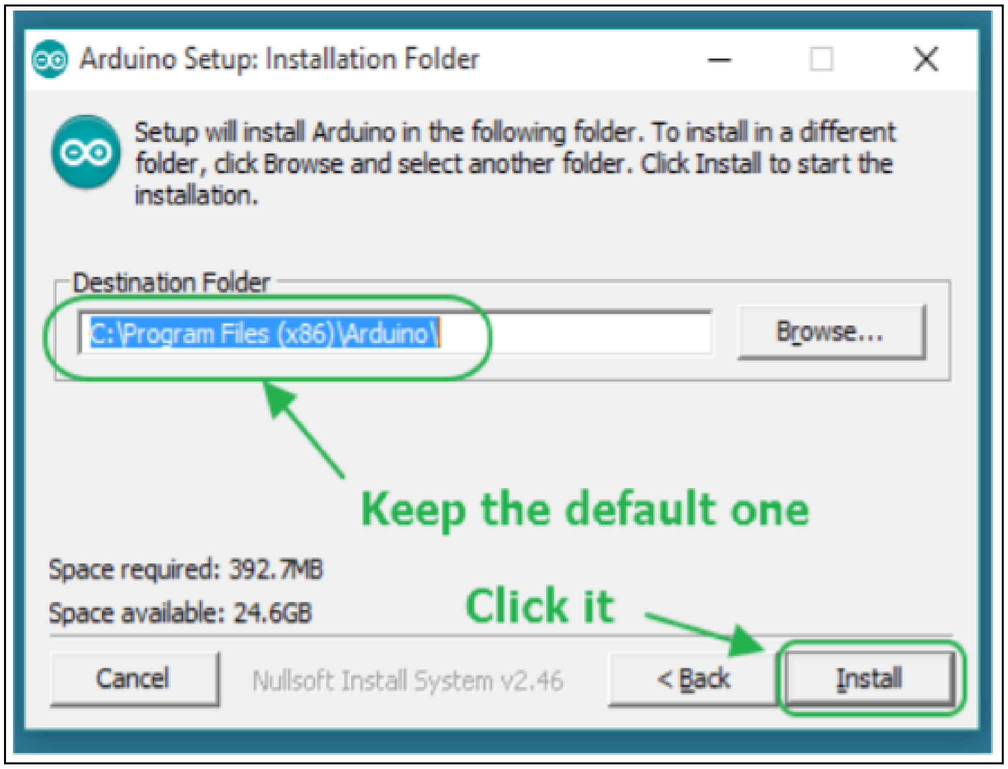
- [✓] Install Arduino software
- [✓] Install USB driver
- [✓] Create Start Menu shortcut
- [✓] Create Desktop shortcut
- [✓] Associate .ino files

Space required: 392.7MB

Click it
Cancel
Nullsoft Install System v2.46
< Back
Next >

Step 2: Directory Installation

- Choose the installation directory.



Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\Arduino\

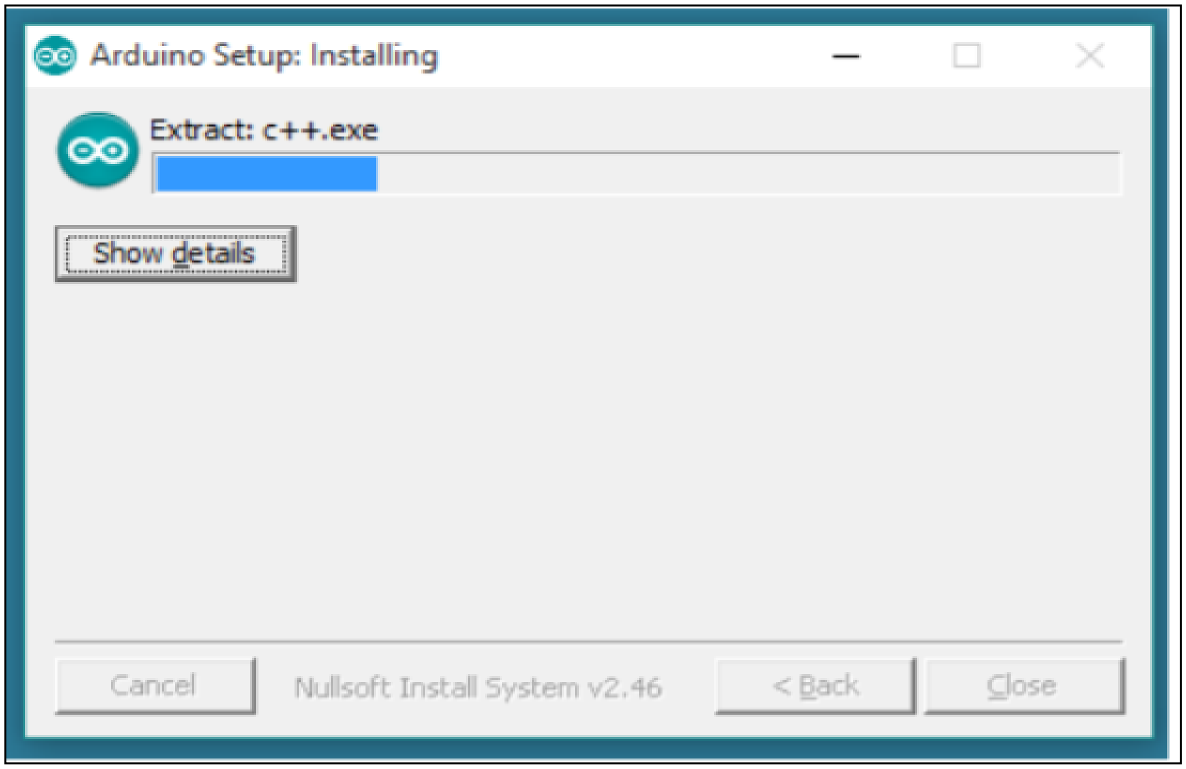
Space required: 392.7MB

Space available: 24.6GB

Nullsoft Install System v2.46

Step 3: Extraction of Files

- The process will extract and install all the required files to execute properly the Arduino Software (IDE)

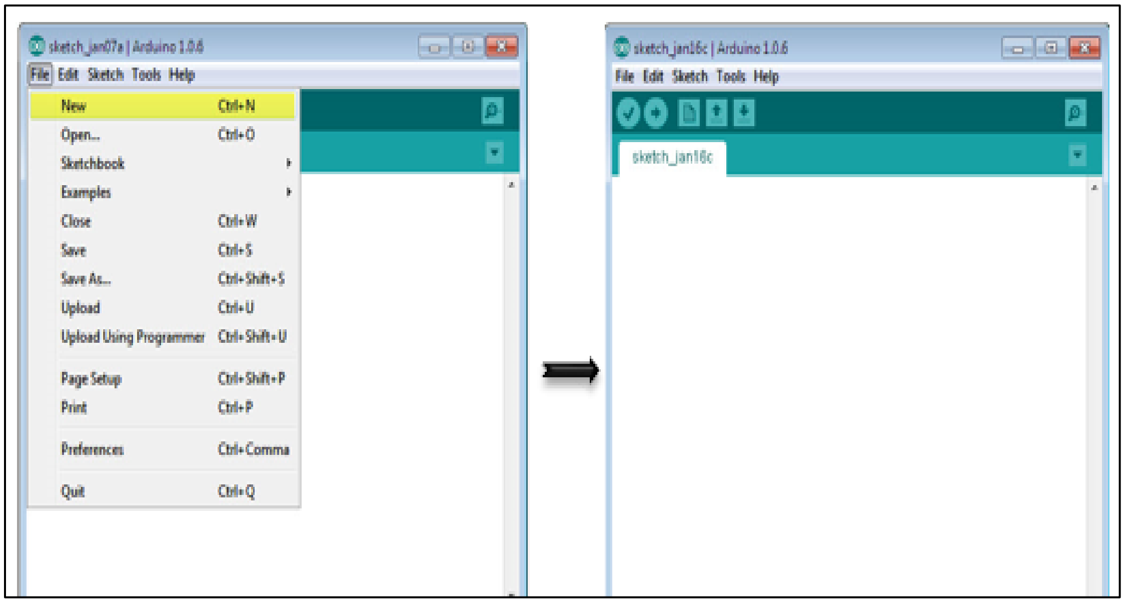


Step 4: Connecting the board

- The USB connection with the PC is necessary to program the board and not just to power it up. The Uno and Mega automatically draw power from either the USB or an external power supply. Connect the board to the computer using the USB cable. The green power LED (labelled PWR) should go on.

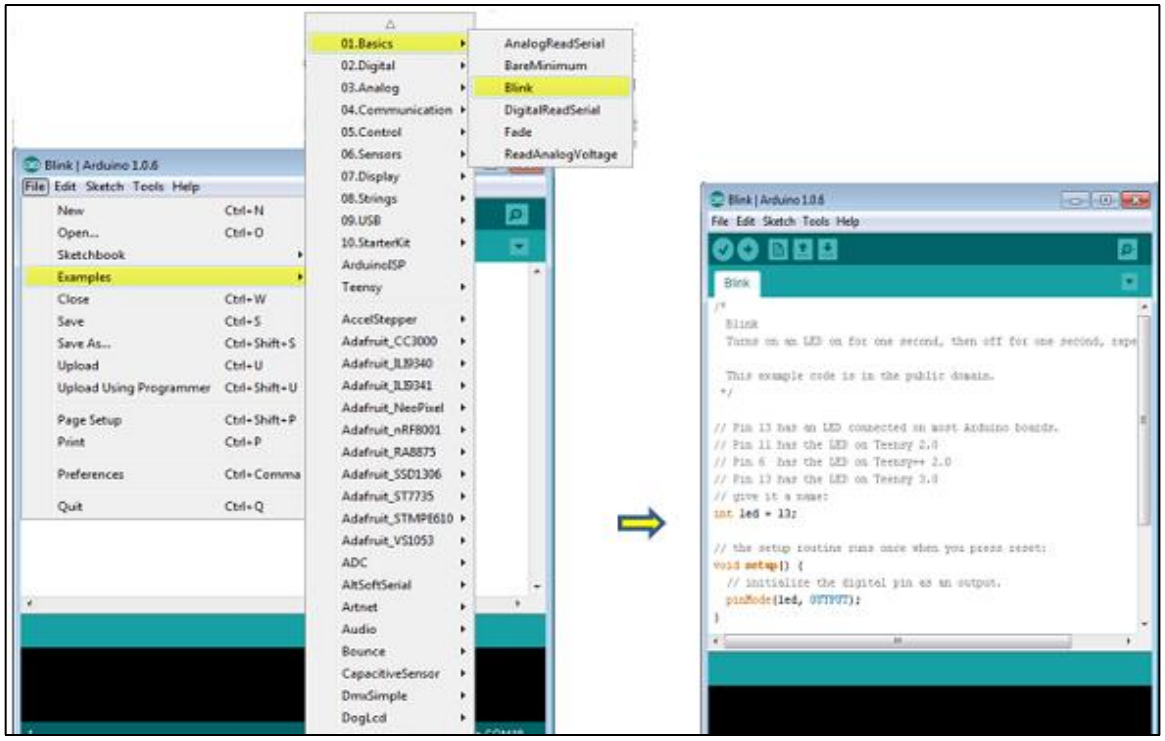
Step 5: Working on the new project

- Open the Arduino IDE software on your computer. Coding in the Arduino language will control your circuit.
- Open a new sketch File by clicking on New.



Step 6: Working on an existing project

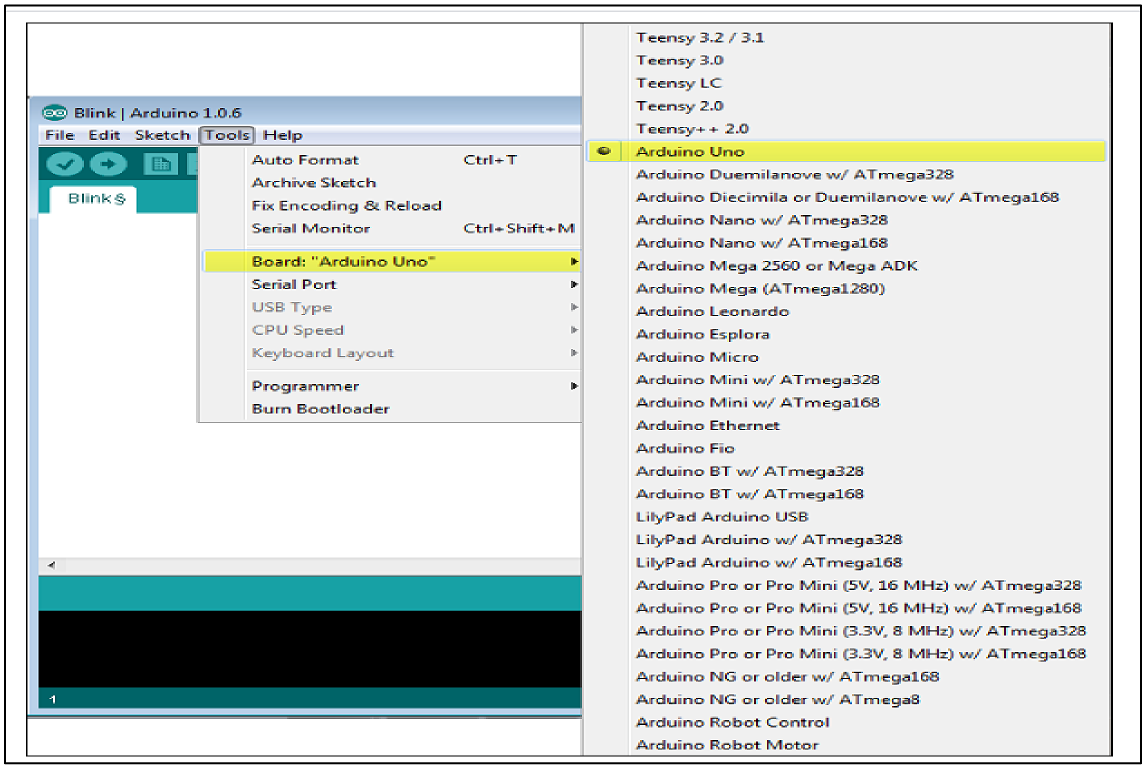
➢ To open an existing project example, select File → Example → Basics → Blink.



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// Pin 11 has the LED on Teensy 2.0
// Pin 6  has the LED on Teensy++ 2.0
// Pin 13 has the LED on Teensy 3.0
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
```

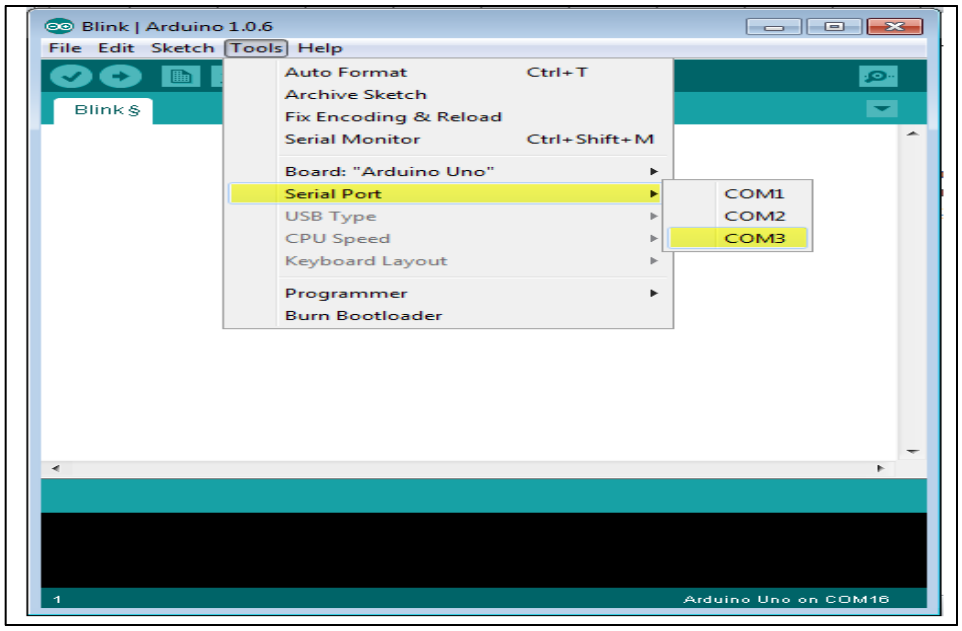
Step 7: **Select your Arduino board.**

- To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.
- Go to Tools → Board and select your board.


Blink | Arduino 1.0.6
File Edit Sketch Tools Help
Blink §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino Uno"
Serial Port
USB Type
CPU Speed
Keyboard Layout
Programmer
Burn Bootloader
Teensy 3.2 / 3.1
Teensy 3.0
Teensy LC
Teensy 2.0
Teensy++ 2.0
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor
1

Step 8: Select your serial port

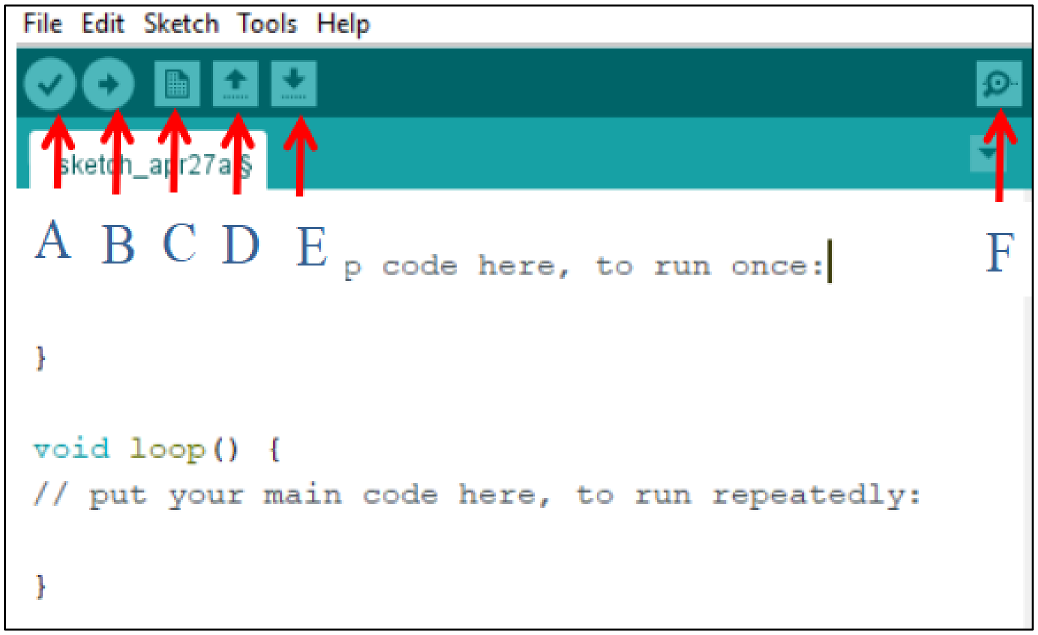
- Select the serial device of the Arduino board.
- Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports).
- To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



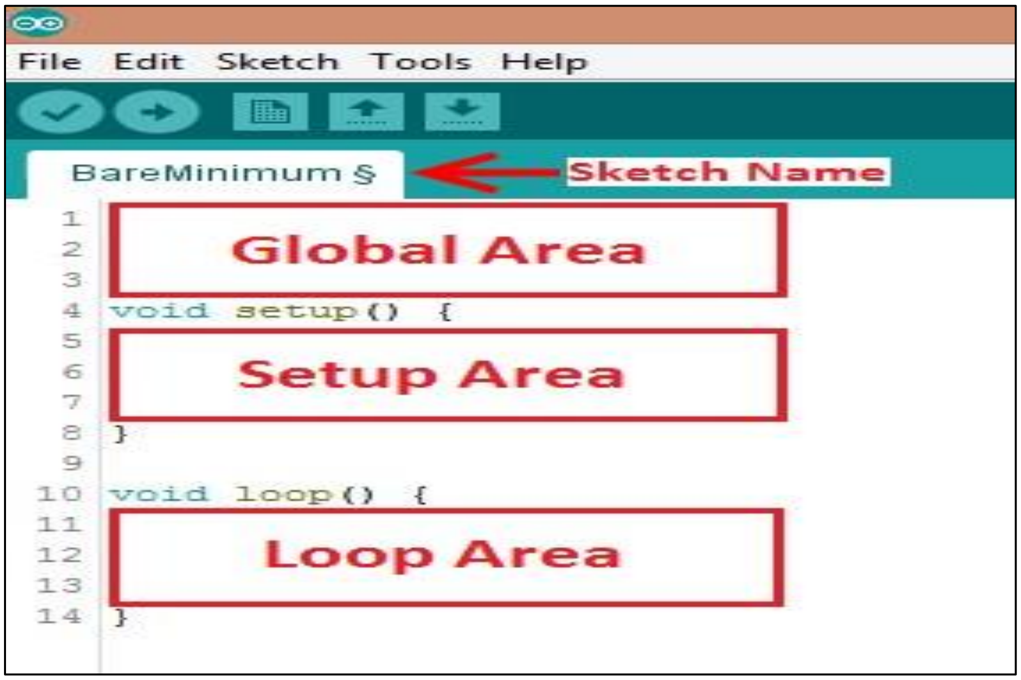
Step 9: Upload the program to your board.

- Click the "Upload" button in the environment.
- Wait a few seconds; you will see the RX and TX LEDs on the board, flashing.
- If the upload is successful, the message "Done uploading" will appear in the status bar.

A	Verify
B	Upload
C	New
D	Open
E	Save
F	Serial Motor



```
p code here, to run once:

}

void loop() {
// put your main code here, to run repeatedly:

}
```



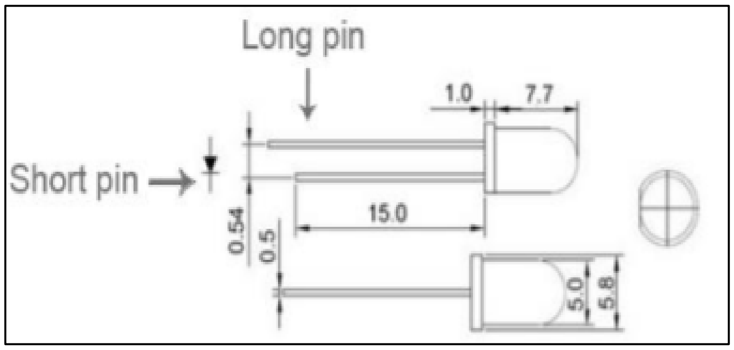
```
void setup() {
}

void loop() {
}
```

Practical 1

Controlling the Light Emitting Diode (LED) with a push button.

Introduction: Push-button is a very simple mechanism which is used to control electronic signal either by blocking it or allowing it to pass. This happens when mechanical pressure is applied to connect two points of the switch together. Push buttons or switches connect two points in a circuit when pressed. When the push-button is released, there is no connection between the two legs of the push-button. Here it turns on the built-in LED on pin 11 when the button is pressed. The LED stays ON as long as the button is being pressed.

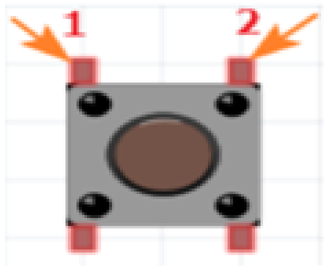
LED Specifications



Pin definition

Long pin	+5V
Short pin	GND

Push Button



Specifications:

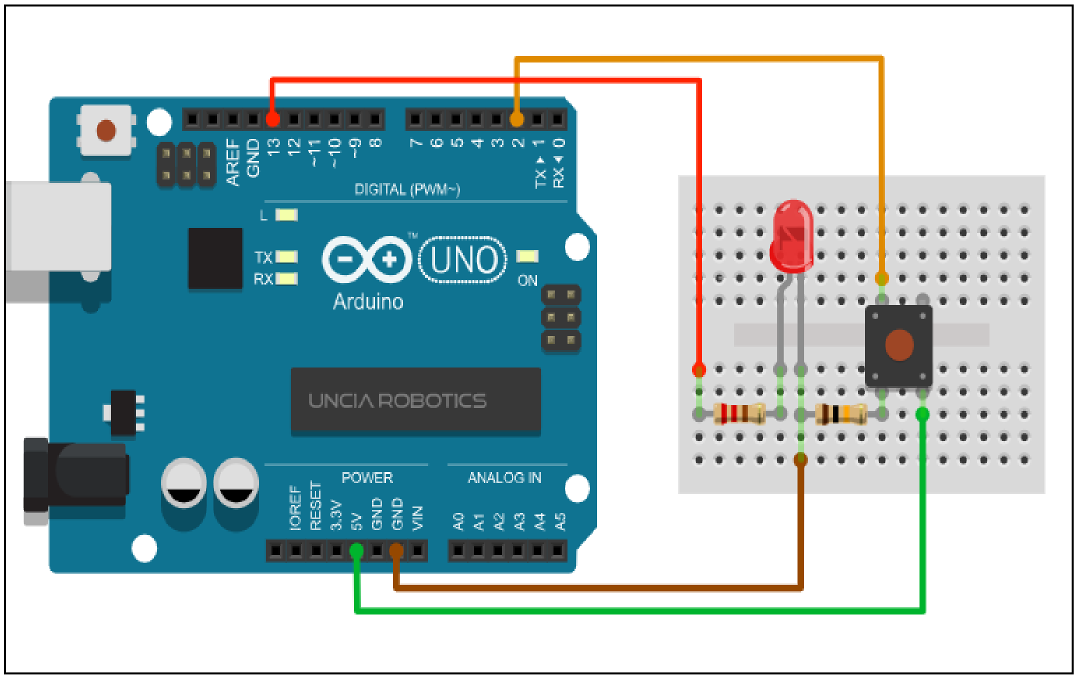
Size	6 x 6 x 5mm
Temperature	-30 ~ +70 Centigrade

Hardware Required:

Component Name	**Quantity**
Arduino UNO	1
LED	1
Push Button	1
220Ω resistor	1
10KΩ resistor	1
USB Cable	1

Breadboard	1
Jumper wires	Several

Connection Diagram:



Steps of working

1. Insert the push button into your breadboard and connect it to the digital pin 7(D7) which act as INPUT.
2. Insert the LED into the breadboard. Attach the positive leg (the longer leg) to digital pin 11 of the Arduino Uno, and the negative leg via the 220-ohm resistor to GND. The pin D11 is taken as OUTPUT.
3. The 10kΩ resistor used as PULL-UP resistor and 220 Ω resistors is used to limit the current through the LED.
4. Upload the code as given below.

5. Press the push-button to control the ON state of LED.

The Sketch

- This sketch works by setting pin D7 as for the push button as INPUT and pin 11 as an OUTPUT to power the LED.
- The initial state of the button is set to OFF.
- After that the run a loop that continually reads the state from the pushbutton and sends that value as voltage to the LED. The LED will be ON accordingly.

```
/****************Pressing Button LED*****/
const int buttonPin = 7; // choose the pin for the pushbutton
const int ledPin = 11;  // choose the pin for a LED
int buttonState = 0;   // variable for reading the pushbutton pin status
void setup()
{
  pinMode(ledPin, OUTPUT);  // declare LED as output
  pinMode(buttonPin, INPUT);    // declare pushbutton as input
}
void loop()
{
 buttonState = digitalRead(button Pin);  // read input value
 if (buttonState == HIGH)
{        // check if the input is HIGH (button pressed)
 digitalWrite(ledPin, HIGH);  // turn LED ON
 }
else
 {
  digitalWrite(ledPin, LOW);  // turn LED OFF}}
```

Observation Table:

Sr no.	Push button State	LED State
1		
2		

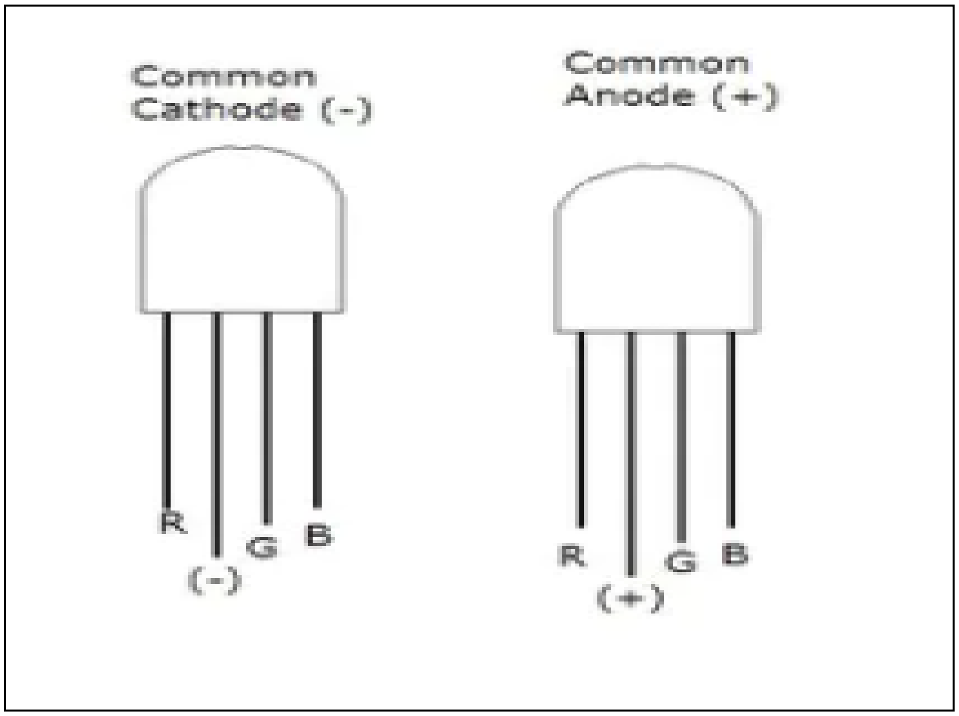
Precautions:

1. The pushbutton is square so it is important to set it appropriately on breadboard.
2. While making the connections make sure to use a pull-down resistor because directly connecting two points of a switch to the circuit will leave the input pin in floating condition and circuit may not work according to the program.
3. It is very important to set pinMode() as OUTPUT first before using digitalWrite() function on that pin.
4. If you do not set the pinMode() to OUTPUT, and connect an LED to a pin, when calling digitalWrite(HIGH), the LED may appear dim.

Practical 2

Interfacing the RGB LED with the Arduino

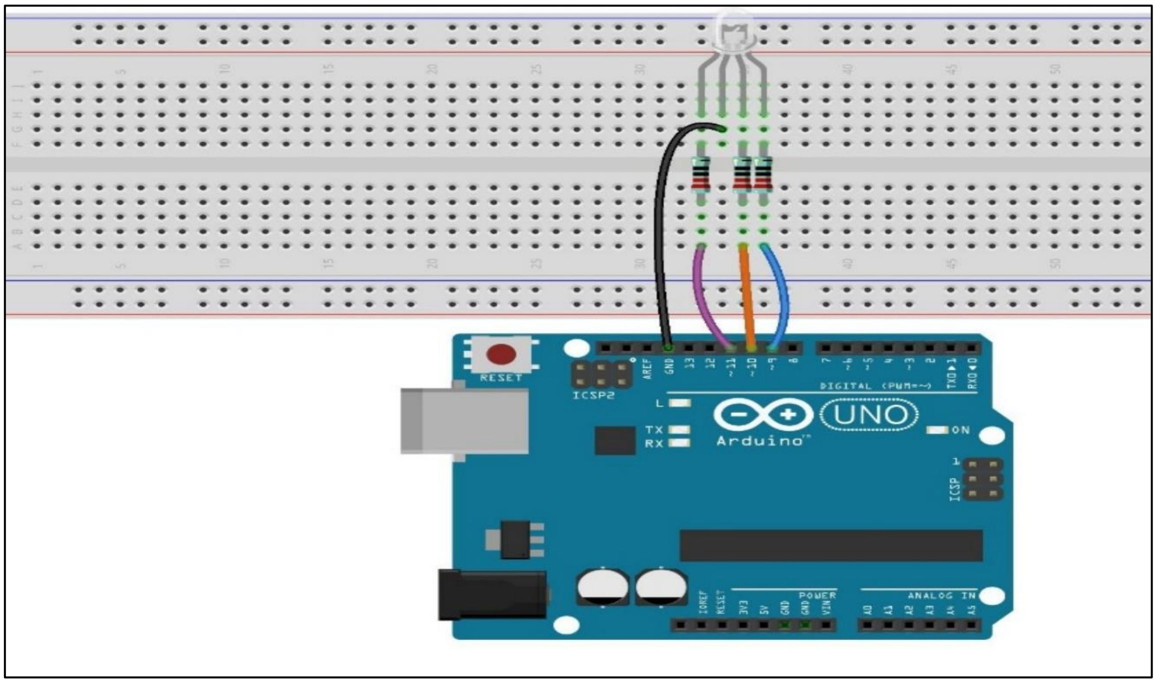
Introduction: There are actually two types of RGB LED's; the common cathode one and the common anode one. In the common cathode RGB led, the cathode of all the LED's is common and we give PWM signals to the anode of LED's while in the common anode RGB led, the anode of all the LED's is common and we give PWM signals to the cathode of LED's. Inside the RGB led, there are three more LED's. So, by changing the brightness of these LED's, we can obtain many other colors. To change brightness of RGB led, we can use the PWM pins of Arduino. The PWM pins will give signal different duty cycles to the RGB led to obtain different colors.



Hardware Required:

Component Name	Quantity
Arduino UNO	1
RGB LED	1
220Ω/330Ω resistor	3
USB Cable	1
Breadboard	1
Jumper wires	several

Connection Diagram:



Steps of working

1. Insert the RGB LED into your breadboard and connect its cathode pin to the GND of the Arduino.
2. Insert the LED into the breadboard. Attach Red pin to pin 8, Green pin to pin 9 and Blue pin to pin 10 of the Arduino via the 220-ohm resistor, and the negative leg to GND.
3. Upload the code as given below.
4. **4.** Observe the changes in the color of the RGB LED.

The Sketch

This sketch works by setting pinsD8, D9, D10 as for the different legs of RGB LED. After that the run a loop that continually reads the value from the pins and sends that value as voltage to the LED. The voltage value is between 0–5 volts, and the blinking of the LED will vary accordingly.

```
/************RGB LED Blink*******/
void setup() {
  // put your setup code here, to run once:
pinMode(8,OUTPUT);
pinMode(9,OUTPUT);
pinMode(10,OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
digitalWrite (8,HIGH);
digitalWrite (10,LOW);
delay(1000);
digitalWrite (9,HIGH);
digitalWrite (8,LOW);
delay(1000);
digitalWrite (10,HIGH);
digitalWrite (9,LOW);
delay(1000);
}
```

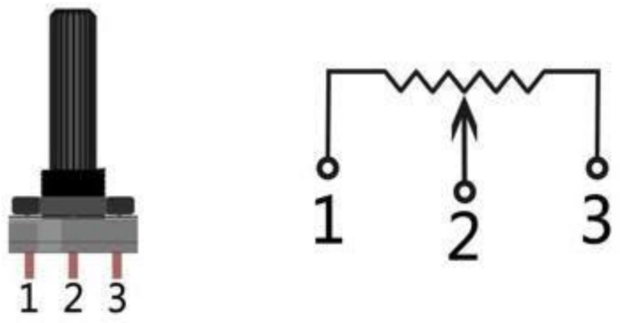
<u>Observations:</u>

Sr. No.	Time(ms)	Color of LED
1		
2		
3		

Practical 3

Controlling the LED blink rate with the potentiometer interfacing with Arduino

Introduction: A potentiometer is a variable resistor with a knob that allows altering the resistance of the potentiometer. The potentiometer manipulates a continuous analog signal, which represents physical measurements. The potentiometer is used with Arduino to control the blink rate of the LED. The potentiometer is an adjustable resistor, and its operating principle is shown in the following figure:

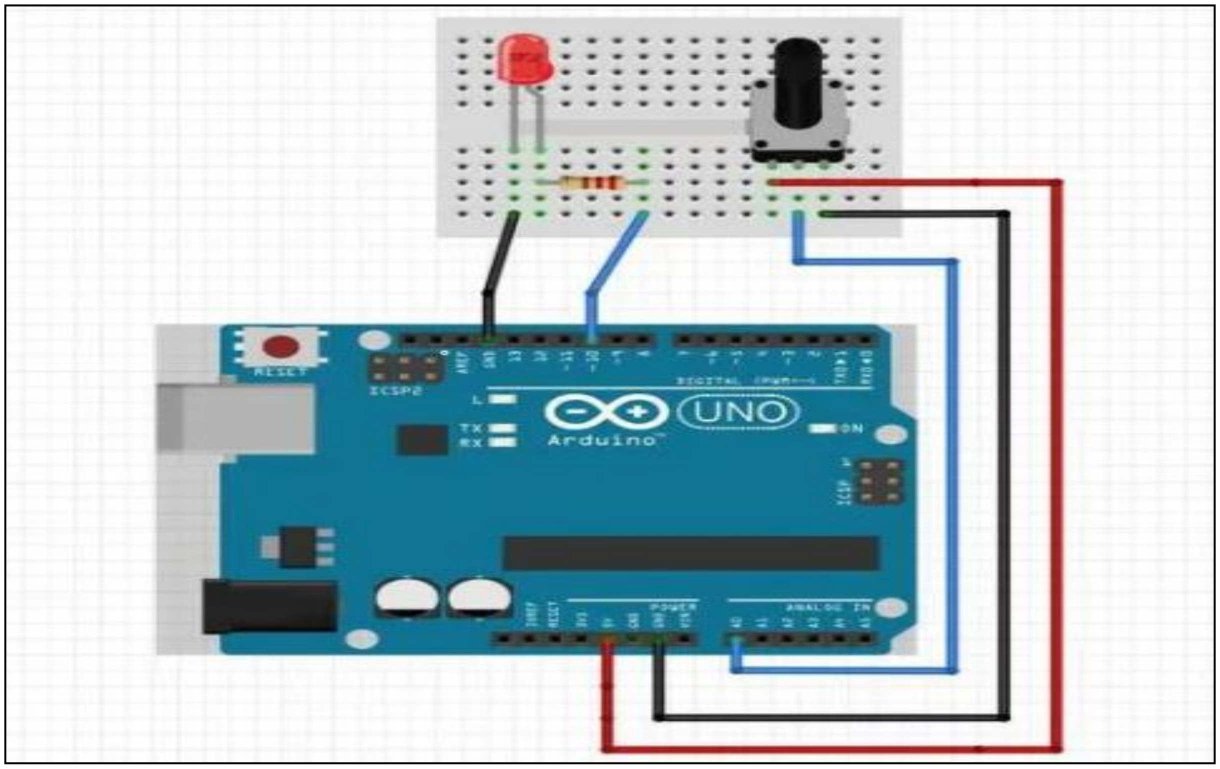


Hardware Required:

Component	Quantity
Arduino Uno	1
Bread board	1
220Ω current limiting resistor	1

5mm LED	1
10KΩ Potentiometer	1
Jumper Wires	Several
Supporting USB data cable	1

Working Diagram:



Steps of working

1. Insert the potentiometer into your breadboard and connect its center pin to the analog pin A2 and the remaining pin to GND on the breadboard.

2. Insert the LED into the breadboard. Attach the positive leg (the longer leg) to pin 13 of the Arduino via the 220-ohm resistor, and the negative leg to GND.
3. Upload the code as given below.
4. Turn the potentiometer to control the brightness of the LED and move the position of pin 2 by rotating the knob, changing the resistance value from pin 2 to both ends.
5. Observe the changes in the blinking rate of the LED.

The Sketch

This sketch works by setting pin A2 as for the potentiometer and pin 9 as an OUTPUT to power the LED. After that the run a loop that continually reads the value from the potentiometer and sends that value as voltage to the LED. The voltage value is between 0–5 volts, and the brightness of the LED will vary accordingly.

Observation Table:

Sr. no.	**Voltage**	**Light Intensity**
1		
2		
3		
4		
5		

Practical 4

Detection of the light using photo resistor

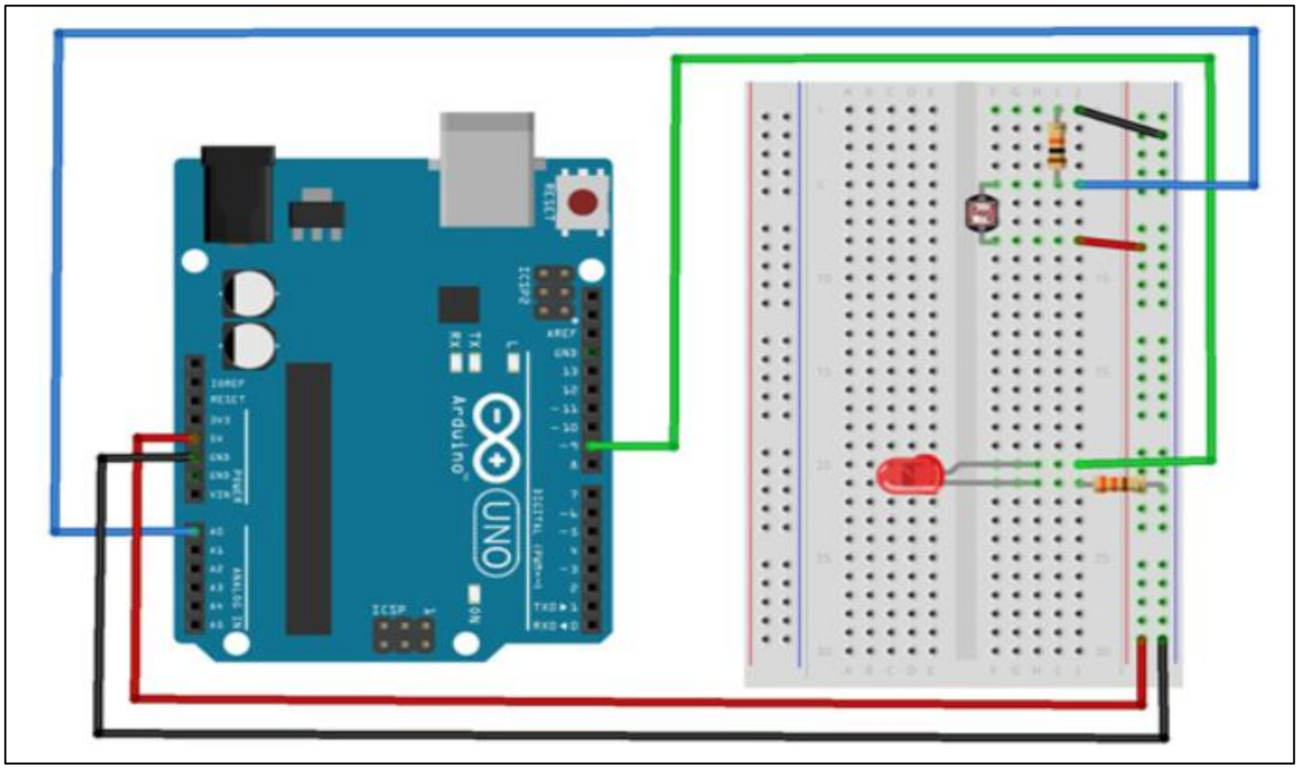
Introduction: A photo resistor or photocell is a light-controlled variable resistor made of a high resistance semiconductor. The resistance of a photo resistor decreases with increasing incident light intensity. A photo resistor can be applied in light-sensitive detector circuits, and light- and dark-activated switching circuits. It's also called light-dependent resistor (LDR).

Hardware Required:

Component Name	**Quantity**
Arduino UNO	1
LED	1
Photo Resistor	1
10KΩ Resistor	1
220Ω Resistor	1

USB Cable	1
Breadboard	1
Jumper wires	several

Connection diagram:



Steps of working

1. Insert the photo resistor into your breadboard and connect its pin to the analog pin A0 and the remaining pin to supply on the breadboard.
2. Insert the LED into the breadboard. Attach the positive leg (the longer leg) to pin 9 of the Arduino via the 220-ohm resistor, and the negative leg to GND.

3. Insert the 10K-ohm resistor
4. Upload the code
5. Turn the photo resistor to ON the LED
6. Observe the changes in the state of the LED.

The Sketch

This sketch works by setting pin A0 as for the photo sensor and pin 9 as an OUTPUT to power the LED. After that the run a loop that continually reads the value from the photo resistor and sends that value as voltage to the LED. The LED will vary accordingly.

```
/****************Photo Resistor to LED*****/
const int sensorPin = A0; // choose the pin for the Photo resistor
const int ledPin = 9;   // choose the pin for a LED
int lightCal;    // variable for reading the initial state of photo sensor
int lightVal;    // variable for reading the current state photo sensor
void setup()
 {
pinMode(ledPin, OUTPUT);  // declare LED as output
lightCal = analogRead(sensorPin);
}
void loop() {
lightVal =analogRead(sensorPin); // read input value
if(lightVal < lightCal-50) { // check if the input is less than threshold
digitalWrite(9,HIGH);    // turn LED ON}
else { digitalWrite(9, LOW);  // turn LED OFF
  }
}
```

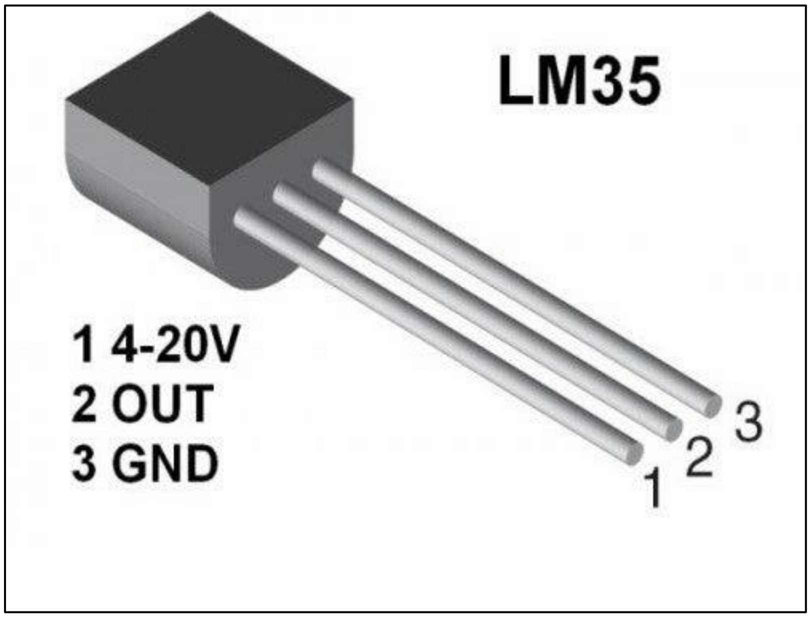
Observation Table:

Sr. no.	Light detected	LED state
1		
2		

Practical 5

Interfacing of temperature sensor LM35 with Arduino

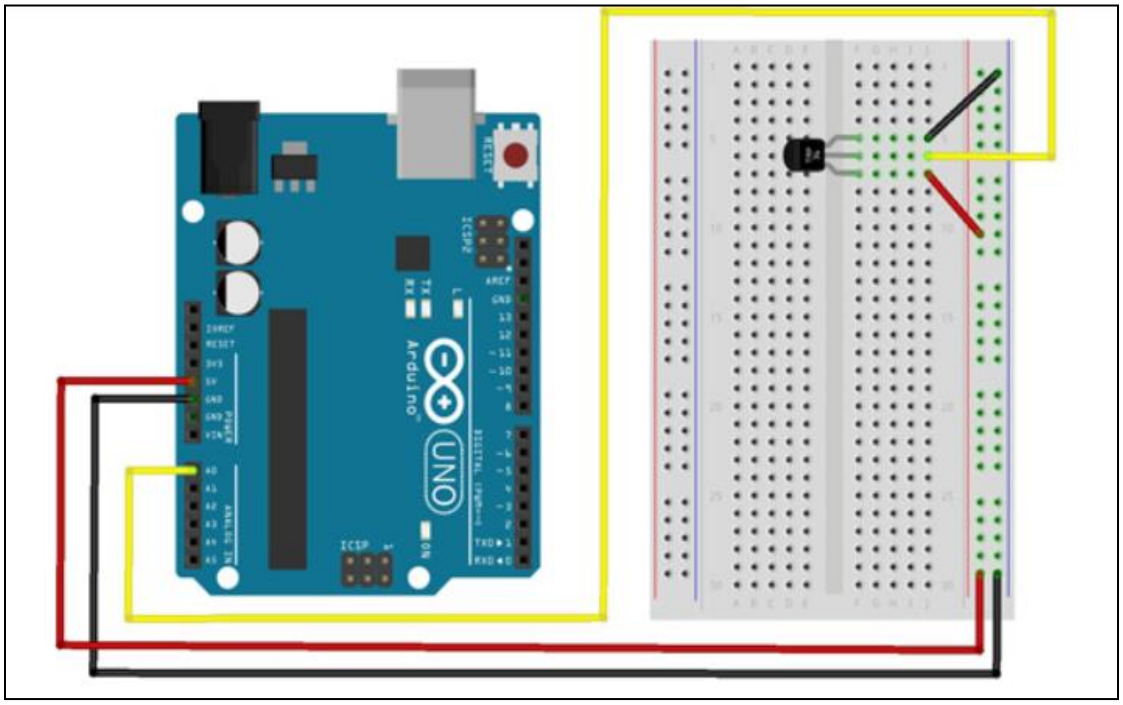
Introduction: The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly proportional to the Centigrade temperature. LM35 is three terminal linear temperature sensors from National semiconductors. It can measure temperature from -55 degree Celsius to +150 degree Celsius. The voltage output of the LM35 increases 10mV per degree Celsius rise in temperature. LM35 can be operated from a 5V supply and the stand by current is less than 60uA. The pin out of LM35 is shown in the figure below.



Hardware Required:

Component Name	Quantity
Arduino UNO	1
Lm35	1
USB Cable	1
Breadboard	1
Jumper wires	Several

Connection Diagram:



Steps of working

1. Insert the temperature sensor into your breadboard and connect its pin1 to the supply.
2. Connect its center pin to the analog pin A0 and the remaining pin3 to GND on the breadboard.
3. Upload the code as given below.
4. Vary the temperature and read the voltage changes.
5. Open the Arduino IDE's serial monitor to see the results.

The Sketch

This sketch works by setting pin A0 as for the temperature sensor. After that the run a loop that continually reads the value from the sensor and sends that value as voltage. The voltage value is between 0–5 volts, when temperature will vary accordingly.

```
/*************File name: LM 35 Temperature Sensor.ino Description: Lit
LM35 Temperature Sensor, let Precision Temperature sensor***/
int LM35Pin=A0;
void setup()
{
Serial.begin(9600);}
void loop ()
{int val;
int data;
val = analogRead(LM35Pin);
data= (val*5)/10;
Serial.print("Temp:");
Serial.print(data);
Serial.println(“C”);
delay(500);
}
```

Observation Table:

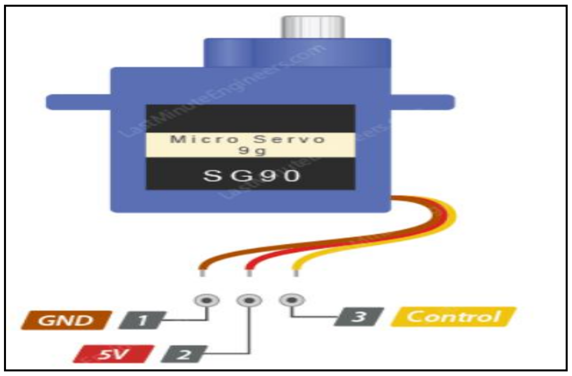
Sr. no.	Voltage	Temperature
1		
2		
3		
4		
5		

Practical 6

Interfacing Servo Motor with the Arduino

Introduction:

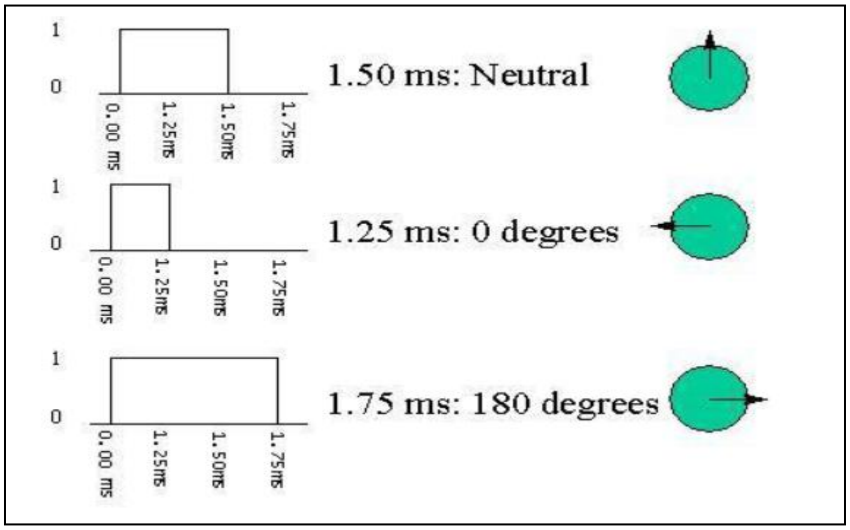
A Servo Motor is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. If the coded signal changes, the angular position of the shaft changes. Servo motors have three terminals – power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino. The ground wire is typically black or brown as shown in figure:



Specifications:

GND	common ground for both the motor and logic.
5V	positive voltage that powers the servo.
Control	Input for the control system.

The control wire is used to communicate the angle. The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse Coded Modulation. The servo expects to see a pulse every 20 milliseconds (.02 seconds). The length of the pulse will determine how far the motor turns. A 1.5 millisecond pulse, for example, will make the motor turn to the 90-degree position (often called as the neutral position). If the pulse is shorter than 1.5 milliseconds, then the motor will turn the shaft closer to 0 degrees. If the pulse is longer than 1.5 milliseconds, the shaft turns closer to 180 degrees.

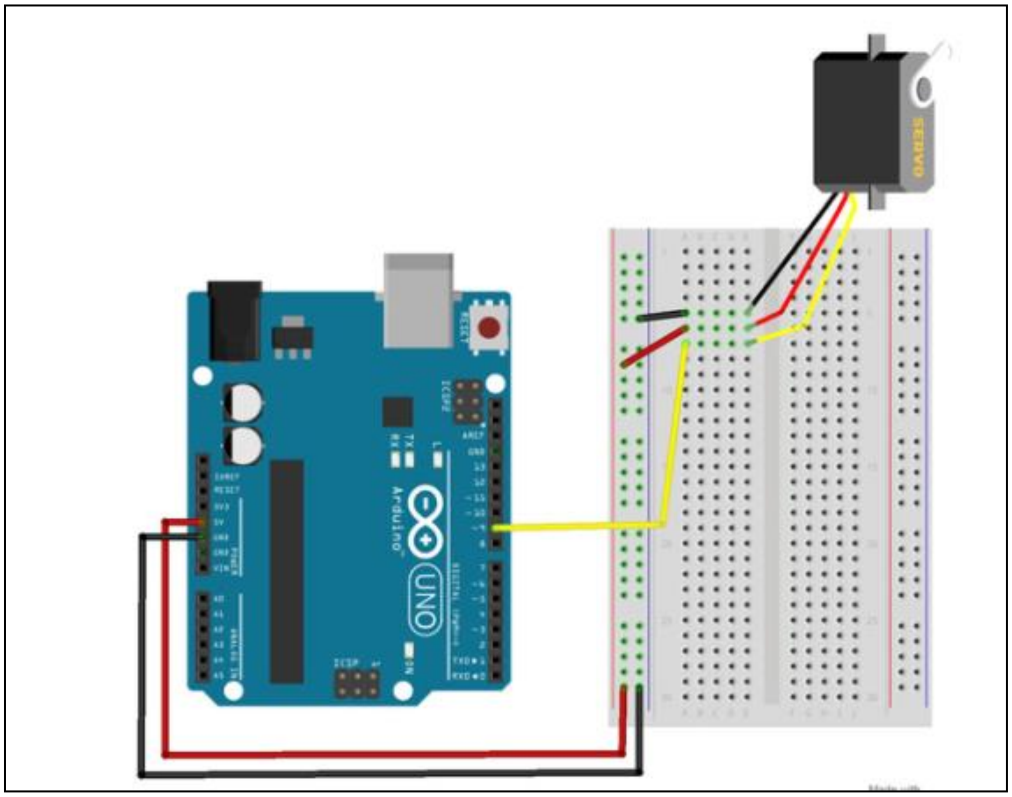


Hardware Required:

Component Name	**Quantity**
Arduino UNO	1
Servo motor	1
USB Cable	1

Breadboard	1
Jumper wires	several

Connection Diagram:



Steps of working

1. The servo motor has a female connector with three pins. The darkest or even black one is usually the ground. Connect this to the Arduino GND.
2. Connect the power cable that in all standards should be red to 5V on the Arduino.
3. Connect the remaining line on the servo connector to a digital pin on the Arduino.

4. Upload the code
5. Observe the position of the shaft.

The Sketch

This sketch works by setting pin D9 as for the control of servo motor. After that the run a loop that continually increment the value of the index of rotation angle and sends that value as voltage to the D9. The voltage value is between 0–5 volts, and the rotation angle of the servo motor will vary accordingly.

```
/******** Servo Motor Rotation******/
#include<Servo.h>
Servo myservo;
int pos=0;
void setup()
{
  // put your setup code here, to run once:
myservo.attach(7);}
void loop() {
  // put your main code here, to run repeatedly:
  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);
}
```

Observation Table:

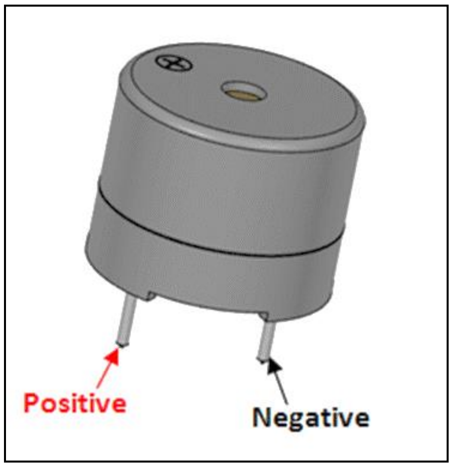
Sr. no.	**Voltage**	**Position of Shaft**
1		
2		
3		
4		
5		

Practical 7

Interfacing of the Active Buzzer with Arduino.

Introduction:

A piezo buzzer is a type of electronic device that’s used to produce beeps and tones. The working principle of the device is piezoelectric effect. The main component of this device is a piezo crystal, which is a special material that changes shape when a voltage applied to it. The active buzzer will only generate sound when it will be electrified. It generates sound at only one frequency. This buzzer operates at an audible frequency of about 2 KHz.



Specifications:

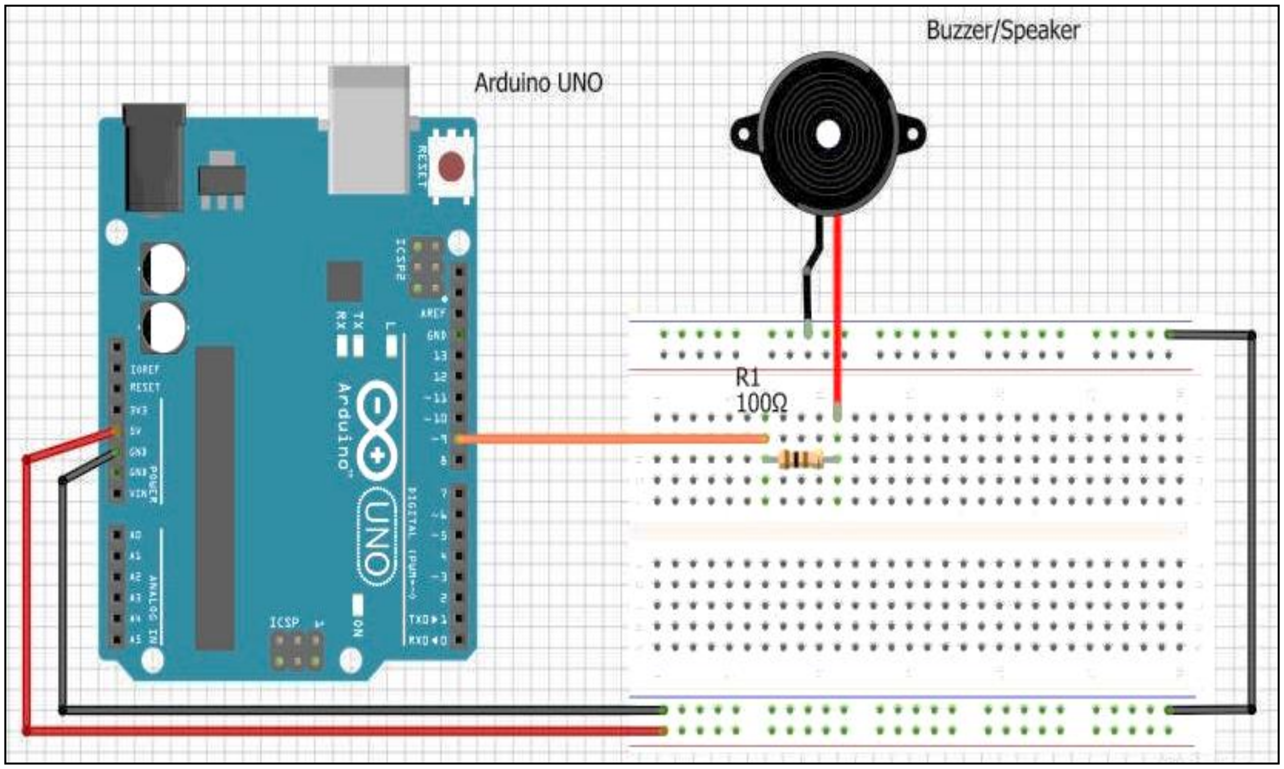
Specification	**Range**
VoltageRange	3.3-5V
Frequency	2KHz

Pin Name	Description
Positive	Identified by (+) symbol or longer terminal lead. Can be powered by 6V DC
Negative	Identified by short terminal lead. Typically connected to the ground of the circuit

<u>Hardware Required:</u>

Component Name	Quantity
Arduino UNO	1
Buzzer / piezo speaker	1
220-ohm resistors	1
USB Cable	1
Breadboard	1
Jumper wires	several

Connection Diagram:



Steps of working:

Connect the Supply wire (RED) of the buzzer to the Digital Pin 9 of the Arduino through a 100-ohm resistor.

Connect the Ground wire (BLACK) of the buzzer to any Ground Pin on the Arduino.

Upload the code

Observe the changes in the pitch and volume of the buzzer.

Sketch:

This sketch works by setting pin D9 as for the control the buzzer. After that the run a loop that continually sends that value as voltage high or low to the D9 using the function digitalWrite( ). The voltage value and the tone generated from the buzzer will vary accordingly.

```
/****Musical buzzer****/
int buzzer = 9;                          //the pin of the active buzzer
void setup()
{
pinMode (buzzer,OUTPUT);                 //initialize the buzzer pin as an output
}
void loop(){
unsigned char i;
while(1){
//output a frequency
for(i=0;i<80;i++)
{
digitalWrite(buzzer,HIGH);
delay(1);                                      //wait for 1ms
digitalWrite(buzzer,LOW);
delay(1);                                      //wait for 1ms
}
//output another frequency
for(i=0;i<100;i++){
digitalWrite(buzzer,HIGH);
delay(2);                                      //wait for 2ms
digitalWrite(buzzer,LOW);
delay(2);                                    //wait for 2ms
}
}
}
```

Observation:

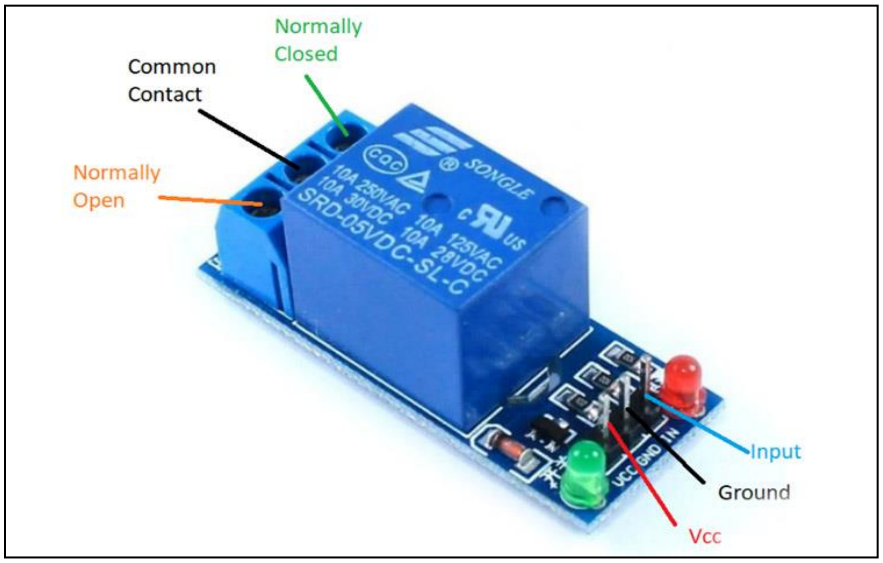
Sr. no.	Change the value	Frequency of tone
1		
2		
3		

Practical 8

Interfacing of the Relay with Arduino.

Introduction:

Relay is an electromagnetic switch, which is controlled by small current, and used to switch ON and OFF relatively much larger current. Means by applying small current we can switch ON the relay which allows much larger current to flow.

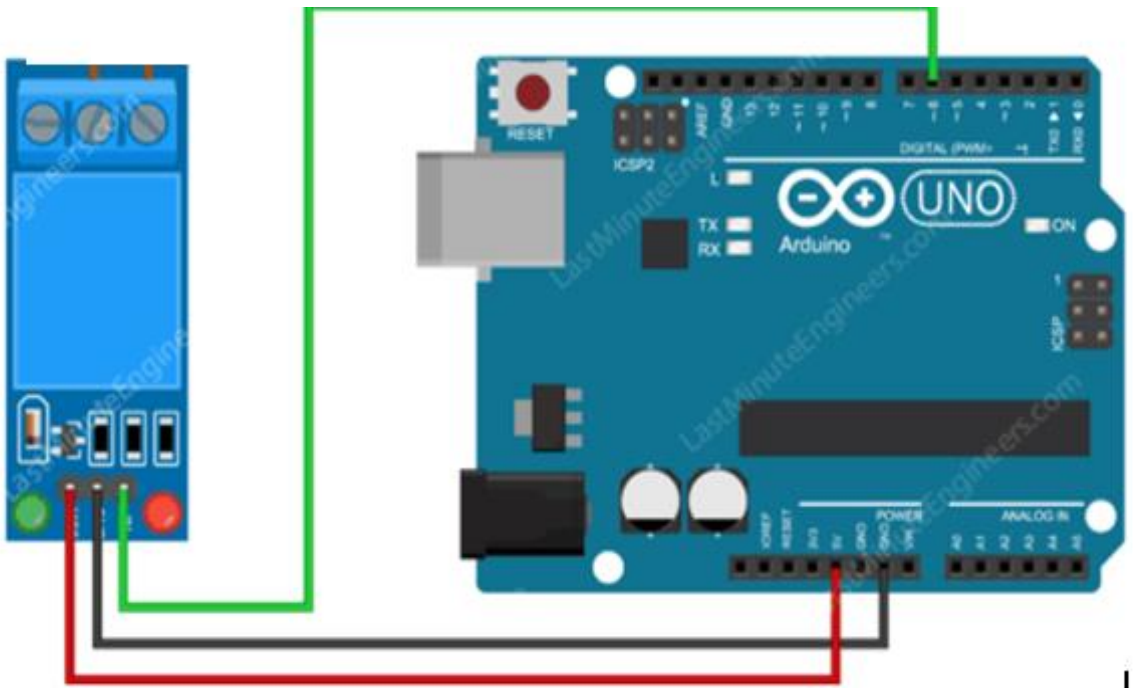


Hardware Required:

Component Name	Quantity
Arduino UNO	1
5V Relay	1

USB Cable	1
Breadboard	1
Jumper wires	several

Connection Diagram:



Steps of working:

1. The relay module connected with three pins. We will connect the relay module with Arduino in the normally open state. The black one of relay is usually the ground. Connect this to the Arduino GND.
2. Connect the red wire of relay module to 5V of the Arduino.
3. Connect the signal pin of relay module to a digital pin 6 of the Arduino.
4. Upload the code

5. Observe the clicking sound of the relay that states the ON and OFF constantly.

Sketch:

This sketch works by setting 5V supply pin of Arduino as for the control of relay module. After that the run a loop that continually sends that value as voltage to the D6 with the delay given.

```
// Arduino Relay Control Code
Int relayPin=6;
#define interval 2000
void setup() {
  pinMode(relayPin, OUTPUT);
}
void loop()
{
   digitalWrite(relayPin, HIGH);
   delay(interval);
   digitalWrite(relayPin, LOW);
   delay(interval);
}
```

Observations:

Sr. No.	**Delay**	**Relay Status**
1		
2		

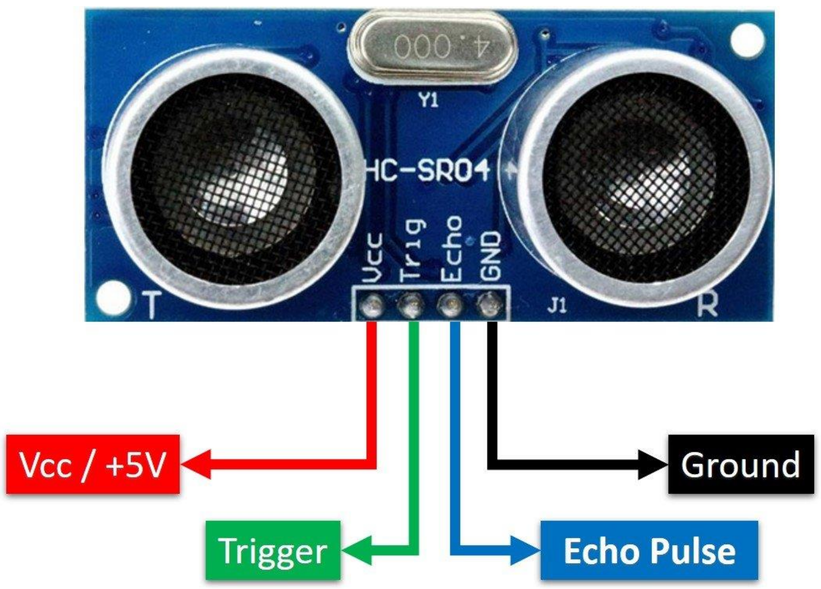
Practical 9

Building Intrusion Detection System with Arduino and Ultrasonic Sensor

Introduction:

An intrusion detection system (IDS) is a device or software application that monitors a network or systems for malicious activity

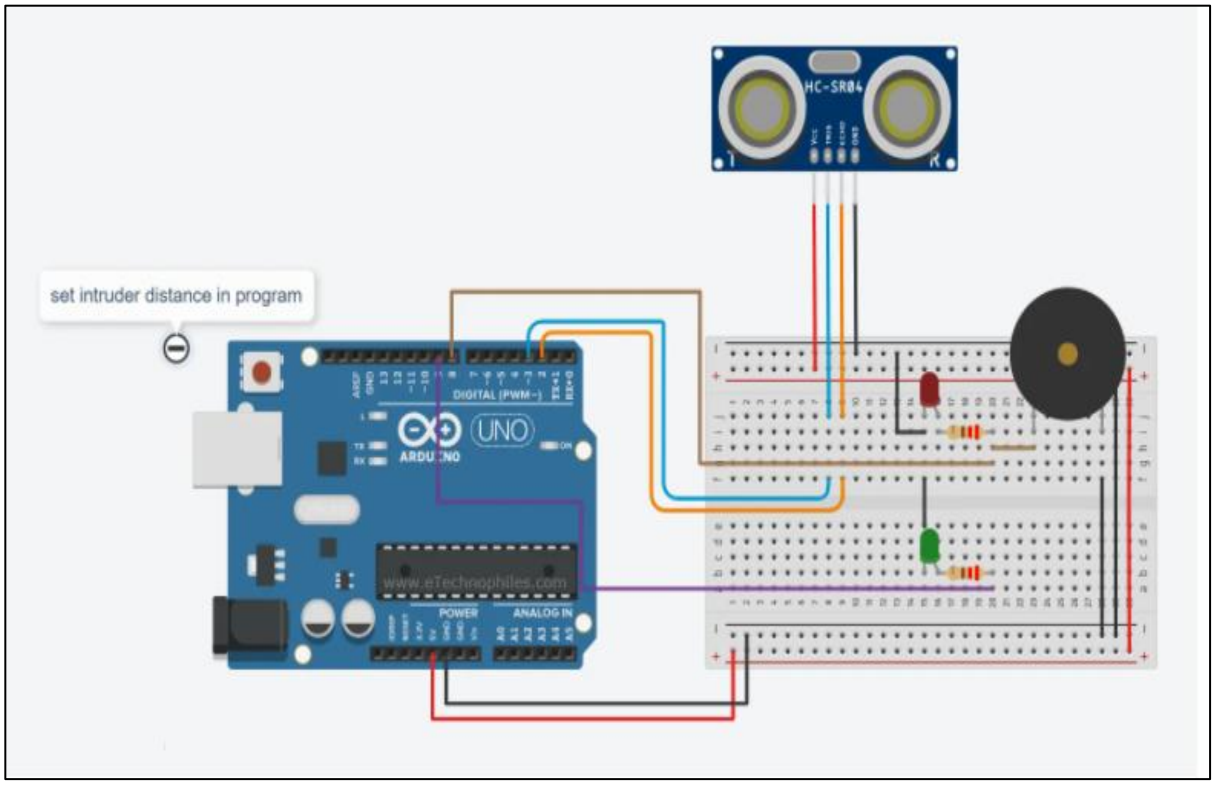
Ultrasonic Sensors: The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1” to 13 feet. It comes complete with ultrasonic transmitter and receiver module. The ultrasonic sensor uses the reflection of sound in obtaining the time between the wave sent and the wave received. It usually sent a wave at the transmission terminal and receives the reflected waves. The time taken is used together with the normal speed of sound in air (340ms-1) to determine the distance between the sensor and the obstacle. The Ultrasonic sensor is used here for the intruder detection. The sound via a buzzer occurs when an object comes near to the sensor. The distance to which the sensor will respond can be easily adjusted in the program.



Hardware Required:

Component Name	**Quantity**
Arduino UNO	1
Red LED	1
Green LED	1
HC-SR04 Ultrasonic Sensor	1
Buzzer	1
USB Cable	1
Breadboard	1
Jumper wires	several

Connection Diagram:



Steps of working

1. Insert the Ultrasonic sensor into your breadboard and connect its Echo pin to the digital pin 2 and the Trigger pin to digital pin 3 of the Arduino.
2. Insert the RED and Green LED into the breadboard. Attach the positive leg (the longer leg) of red LED to signal pin of the Buzzer via the 220-ohm resistor, and the negative leg to GND. The green LED is connected to digital pin 8 of the Arduino.
3. Upload the code.
4. Observe the LEDs and take some object in front of ultrasonic sensor.
5. Observe the changes in the LED and buzzer sound.

The Sketch

This sketch works by setting pin 2 as for the ultrasonic sensors and pin 8, pin9 & pin 10 as an OUTPUT to power the LEDs and buzzer. After that the run a loop that continually reads the value from the echo pin and sends that value as voltage to the LEDs. The color of the LED which glows will vary accordingly to the detection of object in the given range.

```
/**********Intrusion Detection******/
#define echo 2
  #define trig 3
  #define outA 8     // Red LED
  #define outB 9     // Green LED
  #define outC 10   // Buzzer
  float duration;      // time taken by the pulse to return back
  float distance;       // one way distance travelled by the pulse
 const int intruderDistance = 10; // the minimum distance up to which the
sensor is able to sense any object
  void setup() {
    pinMode(trig, OUTPUT);
    pinMode(echo, INPUT);
     pinMode(outA, OUTPUT);
    digitalWrite(outA, LOW);
    pinMode(outB, OUTPUT);
    digitalWrite(outB, LOW);
pinMode(outC, OUTPUT);
    digitalWrite(outC, LOW);
    Serial.begin(9600);
  }
 void loop() {
 time_Measurement();
 distance = (float)duration * (0.0343) / 2;
                     // calculate the one way distance travelled by the pulse
 Serial.println(distance);
 alarm_condition();
  }
    void time_Measurement()
  {                // function to measure the time taken by the pulse to return back
    digitalWrite(trig, LOW);
    delayMicroseconds(2);
    digitalWrite(trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(trig, LOW);
   duration = pulseIn(echo, HIGH);
  }
  void alarm_condition()
  {                //function to execute the output commands based on the sensor
input
    if(distance<=intruderDistance)
    {
 digitalWrite(outA,HIGH);
  digitalWrite(outB,LOW);
  analogWrite(outC,200);}
    else
    {
```

```
digitalWrite(outA,LOW);
digitalWrite (outB, HIGH);
analogWrite (outC,0);
}
}
```

Observation Table:

Sr no.	Object Detected	LED	Buzzer
1			
2			

Practical 10

Directional Control of the DC motor using Arduino

Introduction:

A DC motor (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

Specification

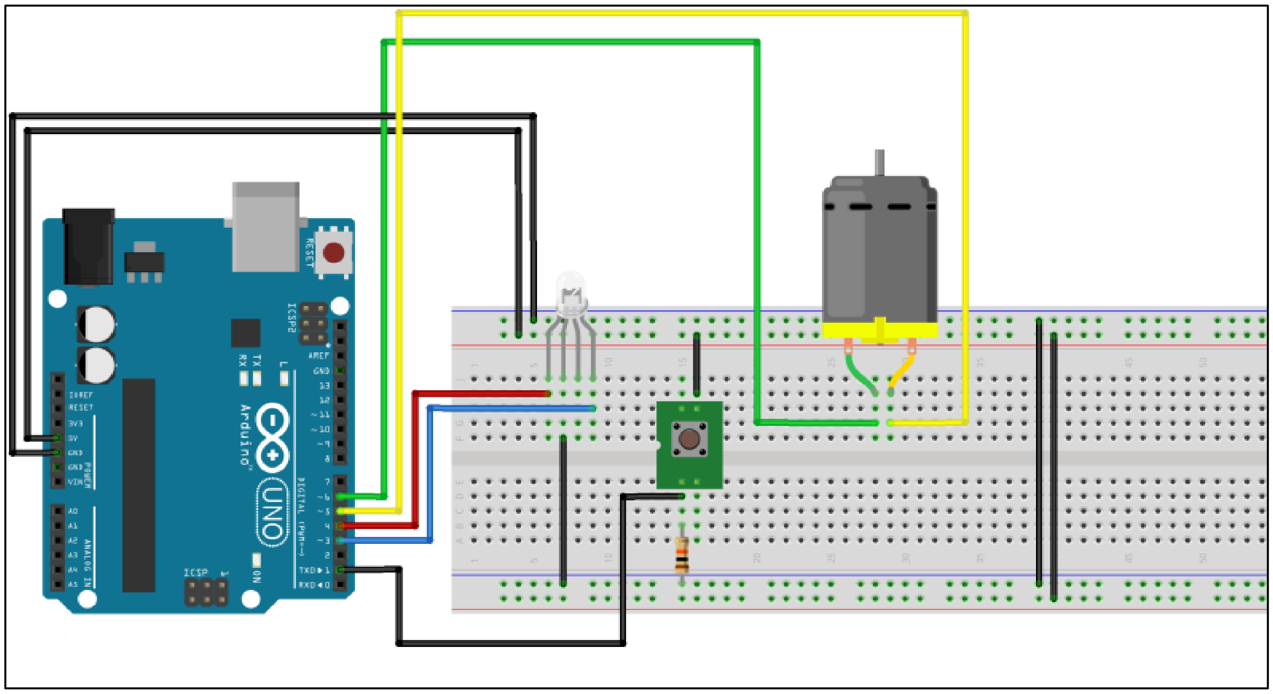
Pin	**Description**
GND	common ground for both the motor and logic
5V	positive voltage that powers the servo
Control	Input for the control system.

The control wire is used to communicate the angle. The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse Coded Modulation. The servo expects to see a pulse every 20 milliseconds (.02 seconds). The length of the pulse will determine how far the motor turns. A 1.5 millisecond pulse, for example, will make the motor turn to the 90-degree position (often called as the neutral position). If the pulse is shorter than 1.5 milliseconds, then the motor will turn the shaft closer to 0 degrees. If the pulse is longer than 1.5 milliseconds, the shaft turns closer to 180 degrees.

Hardware Required:

Component Name	**Quantity**
Arduino UNO	1
DC motor	1
RGB LED	1
Push button	1
10k-ohm resistor	1
USB Cable	1
Breadboard	1
Jumper wires	several

<u>Connection diagram:</u>



<u>Steps of working</u>

1. The servo motor has a female connector with three pins. The darkest or even black one is usually the ground. Connect this to the Arduino GND.
2. Connect the power cable that in all standards should be red to 5V on the Arduino.
3. Connect the remaining line on the servo connector to a digital pin on the Arduino.
4. Upload the code
5. Observe the position of the shaft.

<u>The Sketch</u>

This sketch works by setting pin A2 as for the potentiometer and pin 9 as an OUTPUT to power the LED. After that the run a loop that continually reads the value from the potentiometer and sends that value as voltage to the LED.

The voltage value is between 0–5 volts, and the brightness of the LED will vary accordingly.

```
/******** DC Motor Direction control by RGB******/
const int inputPin=1;
const int blue=3;
const int red=4;
const int motorPin1=5,motorPin2=6;
int dir=LOW;
int prevState=0,currentState=0;
void setup()
{
// put your setup code here, to run once:
pinMode(inputPin,INPUT);
pinMode(motorPin1,OUTPUT);
pinMode(motorPin2,OUTPUT);
pinMode(blue,OUTPUT);
pinMode(red,OUTPUT);
}
void loop()
{
// put your main code here, to run repeatedly:
currentState=digitalRead(inputPin);
if(currentState!=prevState)
{
  if(currentState==HIGH)
  { dir=!dir;
  }
}
prevState=currentState;
if(dir==HIGH)
{
  digitalWrite(motorPin1,HIGH);
  digitalWrite(motorPin2,HIGH);
  digitalWrite(blue,LOW);
digitalWrite(red,HIGH);
}
}
```

Observation Table:

Sr no.	**Voltage**	**Position of Shaft**
1		
2		
3		
4		
5		