COLLEGE OF ENGINEERING
Since 1982
Engineering A Better Tomorrow

NBA Accredited

(An Autonomous Institute)
Affiliated to VTU, Belagavi,
Approved by AICTE, New Delhi,
Recognised by UGC with 2(f) & 12 (B),
Accredited by NBA & NAAC

DEPARTMENT OF Computer Science Engineering (Data Science)

VII SEMESTER

ACADEMIC YEAR 2023–2024[ODD]

Internet of Things Laboratory

MVJ20CDL76

LABORATORY MANUAL

NAM E OF THE STUDENT : .....

BRANCH : .....

UNIVERSITY SEAT NO. : .....

SEMESTER & SECTION : .....

BATCH : .....

Department of Computer Science Engineering (Data Science)

Institute Vision

To become an institute of academic excellence with international standards.

Institute Mission

- Impart quality education along with industry exposure.
- Provide world class facilities to undertake research activities relevant toindustrial and professional needs.
- Promote entrepreneurship and value added education that is sociallyrelevant with economic benefits

Department Vision

To be recognized as a department of repute in Computer Science and Engineering (Data Science) by adopting quality teaching learning process and impart knowledge to make students equipped with capabilities required for professional, industrial and research areas to serve society.

Department Mission

- **Innovation and technically competent:** To impart quality education in Information Science andEngineering by adopting modern teaching learning processes using innovation techniques that enable them to become technically competent.
- **Competitive Software Professionals:** Providing training Programs that bridges gap betweenindustry and academia, to produce competitive software professionals.
- **Personal and Professional growth**: To provide scholarly environment that enables valueaddition to staff and students to achieve personal and profession growth.

Program Outcomes (PO):

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis ofthe information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal,health, safety, legal and cultural issues and the consequent responsibilities relevant to the professionalengineering practice.

7. **Environment and sustainability**: Understand the impact of the professional engineering solutions insocietal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of

the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader indiverse teams.

10. **Communication**: Communicate effectively on complex engineering activities with the engineeringcommunity and with society at large, such as, being able to comprehend and write effective reports anddesign documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineeringand management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Educational Objectives (PEOs):

- **IT Proficiency:** Graduates will excel as IT Experts with extensive knowledge to analyze and designsolutions to Information Engineering problems.
- **Social &moral principles:** Graduates will work in a team, showcase professionalism; ethical valuesexpose themselves to current trends and become responsible Engineers.
- **Higher education:** Graduates will pursue higher studies with the sound knowledge of fundamentalconcepts and skills in basic sciences and IT disciplines.

Program Specific Outcomes (PSO):

PSO1. Software professional expertise: An ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, DBMS, and networking for efficient design of computer-based systems of varying complexity.

PSO2. Core competence: An ability to compete practically to provide solutions for real world problemswith a broad range of programming language and open source platforms in various computing domains

Course outcomes (CO):

On the completion of this laboratory course, the students will be able to:

- Install IoT applications and handling IoT tools.
- Illustrate the methods of deploying smart objects and connect them to network.
- Compare different Application protocols for IoT.
- Infer the role of Data Analytics and Security in IoT.
- Identify sensor technologies for sensing real world entities and understand the role ofIoT invarious domains of Industry.

SYLLABUS

INTERNET OF THINGS LABORATORY

[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2021 -2022)

SEMESTER – VI | Subject Code: MVJ20CDL76
SEE:50 CIE:50 | Total Marks: 100
Number of Lecture Hours:4 | Exam Hours:03

CREDITS – 02

Sr. No	Experiment Name	RBT Level	Hours
1	Create a program that blinks the LED on the Arduino development board.	L2	3
2	Create a program that Arduino can able to communicate with the attached PC.	L1	3
3	Create a program for displaying data from the sensor in regular intervals.	L3	3
4	Write a program to interface LDR (Module) Sensor using Arduino Uno	L1	3
5	Temperature monitoring using LM35 and Arduino.	L1	3
6	MQ2 sensor data accessing using Arduino.	L2	3
7	Ultrasonic sensor interfacing with Arduino.	L3	3
8	Create a program to blink LED in the following manner:- 1010, 1100, 1001,1011	L2	3
9	Playing Piezo buzzer using Arduino.	L1	3
10	Display “hello world” message using arduino LCD display.	L3	3

Course outcomes:	
CO1	Learn and understand IoT applications and tools
CO2	Interfacing sensor and Actuator with Arduino and Raspberry Pi development Modules
CO3	Implementing IoT device by interfacing communication modules
CO4	Developing real time examples using Python
CO5	Elaborate the use of smart objects for designing smart systems

CIE Assessment:
Regular Lab work :20 Record writing :5 Lab Tests (Minimum 2 tests shall be conducted for 15 marks and average of two will be taken) Viva 10 marks
SEE Assessment:
Examinations will be conducted for 100 marks and scaled-down to 50. The weightage shall be, Writeup: 20 marks Conduction: 40 marks
Analysis of results : 20 marks Viva : 20

CO-PO Mapping

CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	3	3	3	2							
CO2	3	3	3	2	2	2						
CO3	3	3	3	3	2							
CO4	2	1	3		2							
CO5	2	1	3		2							

High-3, Medium-2, Low-1

Experiment 1:

a) **Create a program that blinks the LED on the Arduino development board.**
b) **Create a program for blinking of 2 LEDs alternatively.**
c) **Create a program LED fade-in and fade-out.**

What is Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (basedon Wiring), and the Arduino Software (IDE), based on Processing.

Why Arduino?

Thanks to its simple and accessible user experience, 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. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step-by-step instructions of a kit, or sharing ideas online with other members of the Arduino community.

a) Create a program that blinks the LED on the Arduino development board.

This example shows the simplest thing you can do with an Arduino to see physical output: it blinks the on-board LED.

Hardware Required

- Arduino Board
- LED
- 220 ohm resistor

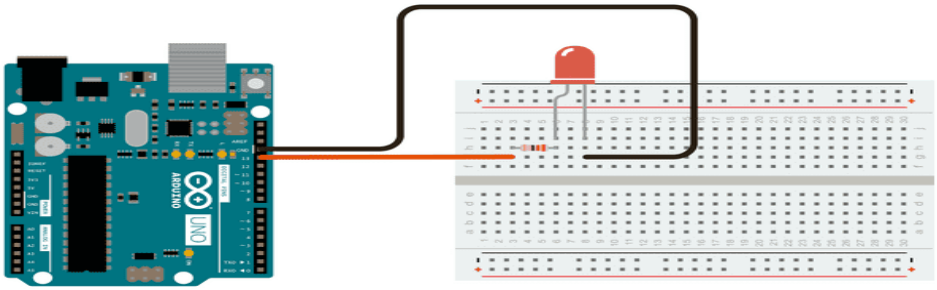
Circuit

This example uses the built-in LED that most Arduino boards have. This LED is connected to a digital pin and its number may vary from board type to board type. To make your life easier, we have a constant that is specified in every board descriptor file. This constantis *LED_BUILTIN* and allows you to control the built-in LED easily. Here is the correspondence between the constant and the digital pin.

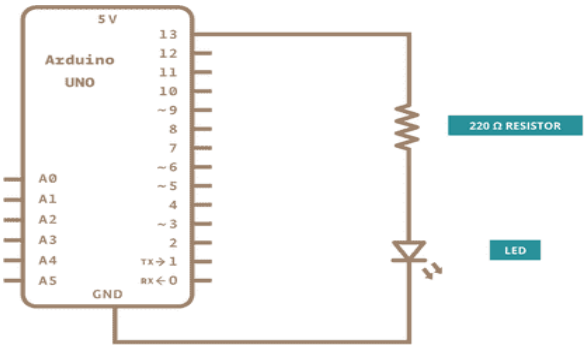
- D13 - 101
- D13 - Due
- D1 - Gemma
- D13 - Intel Edison
- D13 - Intel Galileo Gen2
- D13 - Leonardo and Micro
- D13 - LilyPad
- D13 - LilyPad USB
- D13 - MEGA2560
- D13 - Mini
- D6 - MKR1000
- D13 - Nano
- D13 - Pro
- D13 - Pro Mini
- D13 - UNO
- D13 - Yún
- D13 - Zero

If you want to lit an external LED with this sketch, you need to build this circuit, where you connect one end of the resistor to the digital pin correspondent to the *LED_BUILTIN* constant. Connect the long leg of the LED (the positive leg, called the anode) to the other end of the resistor. Connect the short leg of the LED (the negative leg, called the cathode) to the GND. In the diagram below we show an UNO board that has D13 as the LED_BUILTIN value.

The value of the resistor in series with the LED may be of a different value than 220 ohm; the LED will lit up also with values up to 1K ohm.



Circuit Diagram:



Program:

```
void setup() {  // initialize digital pin 13 as an output.
 pinMode(13, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}
```

After you build the circuit plug your Arduino board into your computer, start the Arduino Software (IDE) and enter the code above. The first thing you do is to initialize pin no.13 as an output pin with the line

pinMode(13, OUTPUT);

In the main loop, you turn the LED on with the line:

digitalWrite(13, HIGH);

This supplies 5 volts to the LED anode. That creates a voltage difference across the pins of the LED, and lights it up. Then you turn it off with the line:

digitalWrite(13, LOW);

That takes the pin no.13 back to 0 volts, and turns the LED off. In between the on and the off, you want enough time for a person to see the change, so the delay() commands tell the board to do nothing for 1000 milliseconds, or one second. When you use the delay() command, nothing else happens for that amount of time. Once you've understood the basic examples, check out the BlinkWithoutDelay example to learn how to create a delay while doing other things.

1.b) Create a program for blinking of 2 LEDs alternatively.

Program:

```
void setup() {  // initialize digital pin 2 as an output.
  pinMode(2, OUTPUT);
  pinMode(13, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED1 on (HIGH is the voltage level)
   digitalWrite(2, LOW);  //turn the LED2 off
  delay(1000); // wait for a second
  digitalWrite(13, LOW); // turn the LED1 off by making the voltage LOW
  digitalWrite(2, HIGH);// turn the LED2 on
  delay(1000); // wait for a second
}
```

1.c) Create a program LED fade-in and fade-out.

Program:

```
int led = 9; // the PWM pin the LED is attached to
int brightness = 0; // how bright the LED is
int fadeAmount =10; // how many points to fade the LED by
// the setup routine runs once when you press reset:

void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:

void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);
  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;
  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
   }
  // wait for 30 milliseconds to see the dimming effect
  delay(3000);
}
```

Experiment 2:

Create a program that Arduino can be able to communicate with the attached PC. Use a serial terminal for the communication.

2.a) Printing number and text messages on serial monitor.

2.b) Create a program to control the blink-rate of LED.

2.c) Create a program which can turn on LED if user enters '1' and turn off the LED if '0' is entered.

Components and supplies:

Arduino Uno.
Jumper wires.
Bread Board(generic).
16x2 LCD
Resister 10k ohm

Software: Arduino IDE

Serial communications provide an easy and flexible way for your Arduino board to interact with your computer and other devices. This chapter explains how to send and receive information using this capability.

To perform the experiment, connect the Arduino serial port to your computer to upload sketches. The upload process sends data from your computer to Arduino and Arduino sends status messagesback to the computer to confirm the transfer is working. The Sketches here show how you can usethis communication link to send and receive any information between Arduino and your computeror another serial device.

You can also send data from the Serial Monitor to Arduino by entering text in the text box to the left of the Send button. Baud rate is selected using the drop-down box on the bottom right. You can use the drop down labeled "No line ending" to automatically send a carriage return or a combination of a carriage return and a line at the end of each message sent when clicking the Sendbutton.

Your Arduino sketch can use the serial port to indirectly access (usually via a proxy program written in a language like Processing) all the resources (memory, screen, keyboard, mouse, network connectivity, etc.) that your computer has. Your computer can also use the serial link to interact with sensors or other devices connected to Arduino.

2.a) Printing number and text messages on serial monitor.

This sketch prints the constant, integers, number and text within double quotation on the Serial Monitor:

Program:

```
void setup()
 {
// Start serial communication for Uno R3
Serial.begin(9600); //Start serial communication for Arduino Mega at Serial port 3
}

void loop() {
```

```
  Serial.print("Hello World!\n");
  Serial.print("17\n");
  Serial.print("3.14159265359\n");
  Serial.print(25);
  Serial.print("\n"); // Outputs the ASCII string "25" to the serial port
  Serial.print(2.7345); // Outputs "2.73"
  delay(1000);
 }
```

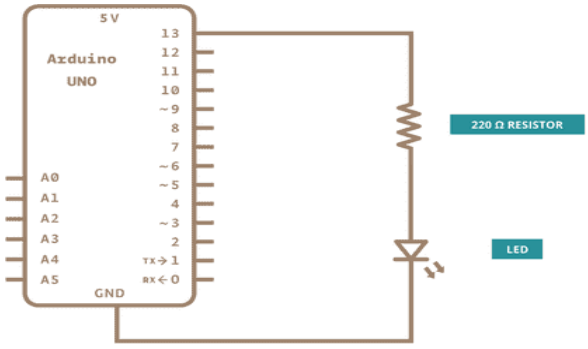
2.b) Create a program to control the blink-rate of LED

Program:

```
const int ledPin = 13; // pin the LED is connected to
int   blinkRate=0;     // blink rate stored in this variable

void setup()
{
  Serial.begin(9600); // Initialize serial port to send and receive at 9600 baud
  pinMode(ledPin, OUTPUT); // set this pin as output
}

void loop()
{
  if ( Serial.available()) // Check to see if at least one character is available
  {
    char ch = Serial.read();
    if(ch >= '0' && ch <= '9') // is this an ascii digit between 0 and 9?
    {
      blinkRate = (ch - '0');      // ASCII value converted to numeric value
      blinkRate = blinkRate * 100; // actual blinkrate is 100 mS times received
    }
  }
  blink();
}

// blink the LED with the on and off times determined by blinkRate
void blink()
{
  digitalWrite(ledPin,HIGH);
 delay(blinkRate); // delay depends on blinkrate value

 digitalWrite(ledPin,LOW);
```

```
delay(blinkRate);
 }
```

Circuit Diagram:



2.c) Create a program which can turn on LED if user enters '1' and turn off the LED if '0' is entered.

Program:

```
const int ledPin = 13; // pin the LED is connected to
void setup()
{
  Serial.begin(9600); // Initialize serial port to send and receive at 9600 baud
  pinMode(ledPin, OUTPUT); // set this pin as output
}

void loop()
{
  if ( Serial.available()) // Check to see if at least one character is available
  {
    char ch = Serial.read();
if (ch =='0')
  digitalWrite(ledPin,HIGH);
if(ch=='1')
digitalWrite(ledPin,LOW);
else
digitalWrite(ledPin,LOW);
}
```

Discussion

To display text and numbers from your sketch on a PC or Mac via a serial link, put the Serial.begin(9600) statement in setup(), and then use Serial.print() statements to print the textand values you want to see.

The Arduino Serial Monitor function can display serial data sent from Arduino. To start the Serial Monitor, click the Serial Monitor toolbar icon as shown in Figure 4-2. A new window will open for displaying output from Arduino.

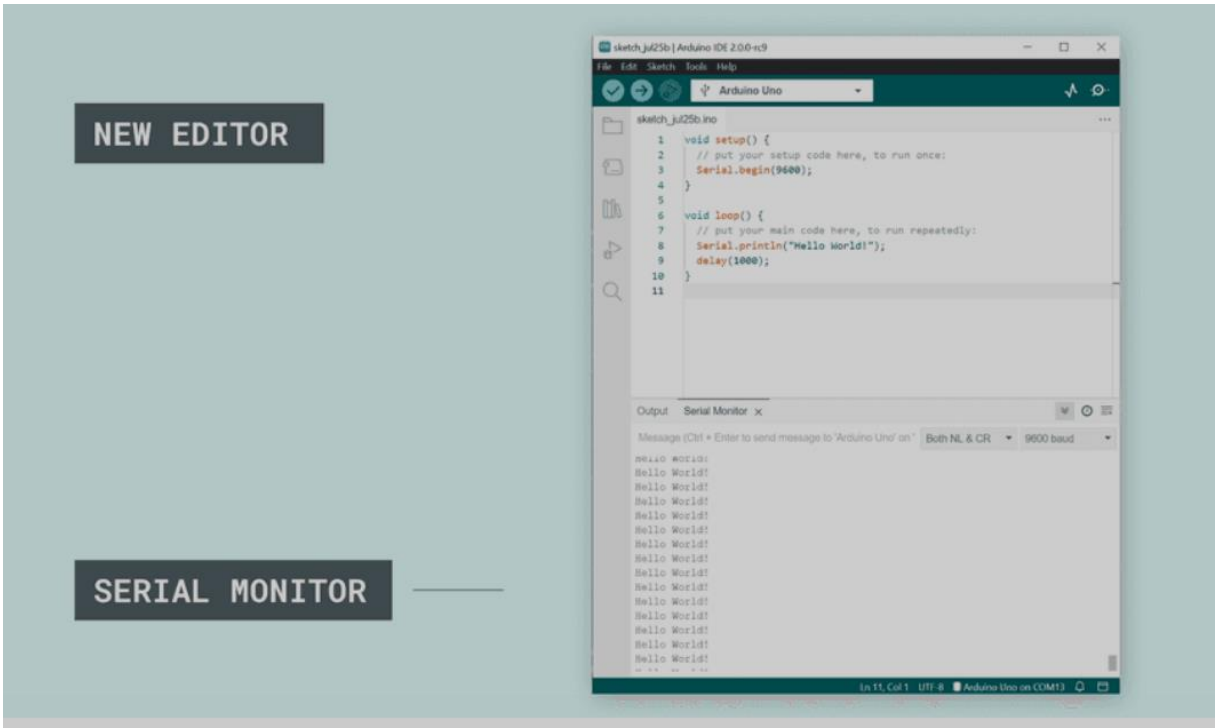


Figure 2: Clicking the Serial Monitor icon to see serial output

Your sketch must call the Serial.begin() function before it can use serial input or output. The function takes a single parameter: the desired communication speed. You must use the same speed for the sending side and the receiving side, or you will see gobbledygook (or nothing at all) on the screen. This example and most of the others in this book use a speed of 9,600 baud (*baud* is a measure of the number of bits transmitted per second). The 9,600 baud rate is approximately 1,000 characters per second. You can send at lower or higher rates (the range is 300 to 115,200), but make sure both sides use the same speed. The Serial Monitor sets the speed using the baud rate drop down (at the bottom right of the Serial Monitor window as shown in fig.2). If your output looks something like this:

`3??f<ÌxÌ□ □ □ü`³??f<

you should check that the selected baud rate on your computer matches the rate set by Serial.begin() in your sketch.

Experiment 3

a) Create a program that displays data from the sensor in regular intervals in a compact format.

Components and supplies:

Arduino Uno.

Jumper wires.

Bread Board(generic).

Light detecting resistor.

Resister 10k ohm

Software: Arduino IDE.

There are different type of sensors which can record different type of data. All sensors work differently e.g. light sensors, ultrasonic sensor, gas sensor, humidity sensors etc. Although all sensors work differently but when it come to interfacing these sensors with a micro controller (Arduino Uno in our case) the process is pretty much the same. Sensor records the physical data by changing the voltage at their output pin in response to different physical condition

Consider a light sensor (LDR). When it's in a dark environment the output voltage is low but when we place it in a brighter environment it's output voltage increases. Now, this change in voltage is noted by micro controller and it can programmed to respond accordingly. Arduino Uno has a set of Analog input pins which can are used to take analog input signalsfrom a sensor.

Remember there are two types of signals:

1. **Digital Signals**: These signals have only two values i.e. 1 or 0 (on or off).
2. **Analog Signals**: These signals have values in a range. In the case of Arduino it scales the value in the range from 0 to 255.

So, we will connect our LDR with A0 pin of the Arduino Uno.

Program:

```
int sensor_Value = 0;

void setup()

{
Serial.begin(9600);

pinMode(A0, INPUT); // Declare the A0 as an input

}

void loop() {
Sensor_Value = analogRead(A0); //Reads the value from the sensor connected at A0 and stores it.

Serial.println("LDR value is :");

Serial.println(Sensor_Value); //Prints the value of LDR to Serial Monitor.

}
```

Circuit Diagram:

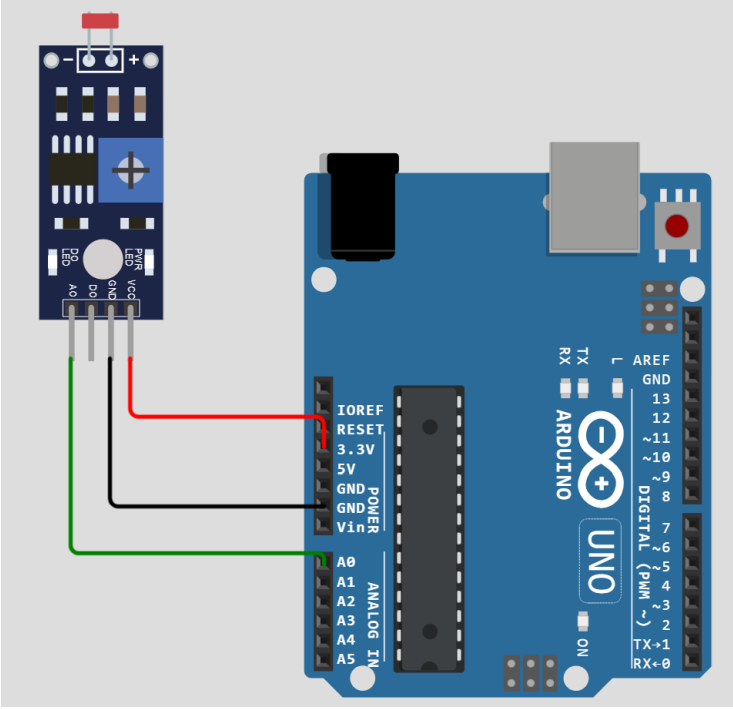


Figure 3: Arduino and LDR interfacing circuit diagram

Experiment No.4

4. Write a program to interface LDR (Module) Sensor using Arduino Uno

a) Interfacing LDR and Arduino using Analog pin and controlling LED based on intensity of light.

b) Interfacing LDR and Arduino using Digital pin to detect the presence of light.

Components and supplies:

Arduino Uno.
Jumper wires.
Bread Board(generic).
Light detecting resistor.
Resister 10k ohm

Working Principle:

LDR SENSOR is nothing but a light-dependent resistor, its resistance changes according to changes in light intensity. The LDR SENSOR is made of photosensitive material. The zig-zag lines you see on the sensor are nothing but a photosensitive material. When light falls on this material the resistance of the material changes and hence conductivity.

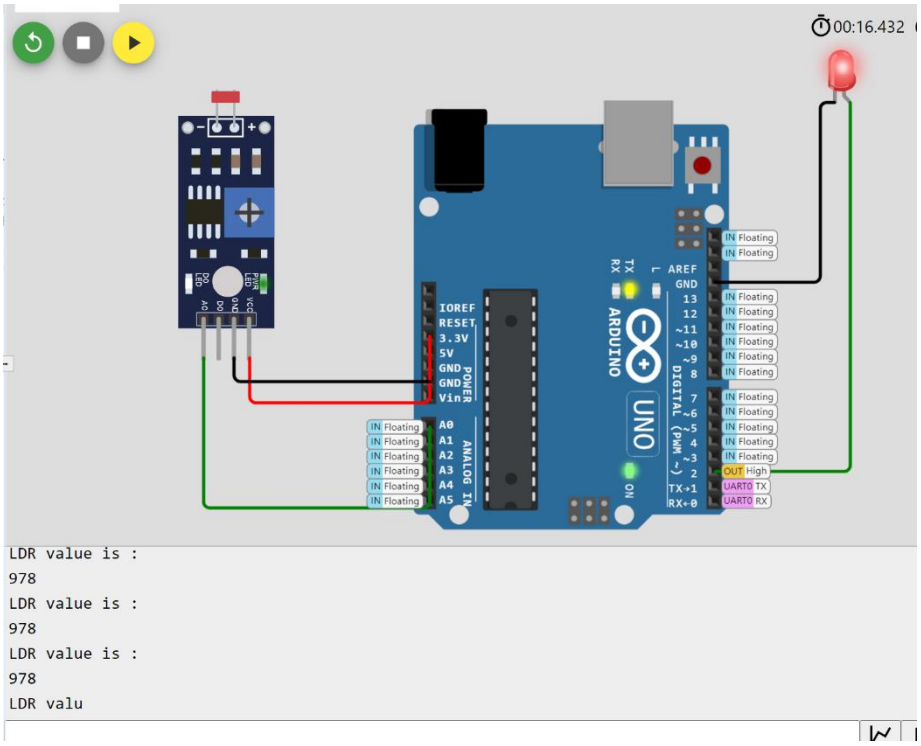
a) Interfacing LDR and Arduino using Analog pin and controlling LED based on intensity of light.

Program:

```
int LDRInput=A0; //Set Analog Input A0 for LDR.
int LED=2;
void setup() {
Serial.begin(9600);
pinMode(LDRInput,INPUT);
pinMode(LED,OUTPUT);
}
void loop() {
int value=analogRead(LDRInput);//Reads the Value of LDR(light).
Serial.println("LDR value is :");//Prints the value of LDR to Serial Monitor.
Serial.println(value);
if(value<300)
  {
    digitalWrite(LED,HIGH);//The LED turns ON in Dark.
  }
  else
```

```
 {

   digitalWrite(LED,LOW);//The LED turns OFF in Light.
 }

}
```

Circuit Diagram:



b) Interfacing LDR and Arduino using Digital pin to detect the presence of light.

Program:

```
int LDRInput=13; //Set Analog Input A0 for LDR.
int LED=2;
void setup()
 {
Serial.begin(9600);
pinMode(LDRInput,INPUT);
pinMode(LED,OUTPUT);
}
void loop()
 {

int value=digitalRead(LDRInput);//Reads the Value of LDR(light).
Serial.println("LDR value is:");//Prints the value of LDR to Serial Monitor.
```

```
Serial.println(value);


if(value)
  {
   digitalWrite(LED,HIGH);//The LED turns ON in Dark.
  }
  else
  {
   digitalWrite(LED,LOW);//The LED turns OFF in Light.
  }
}
```

Circuit Diagram:

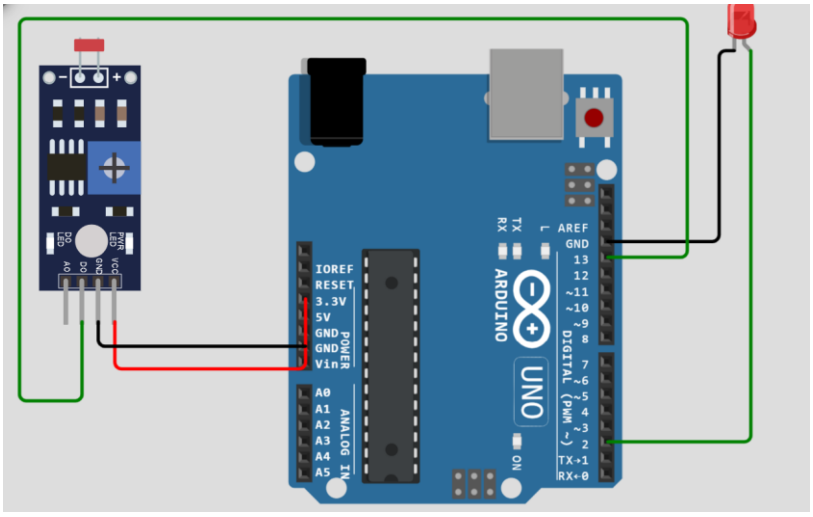
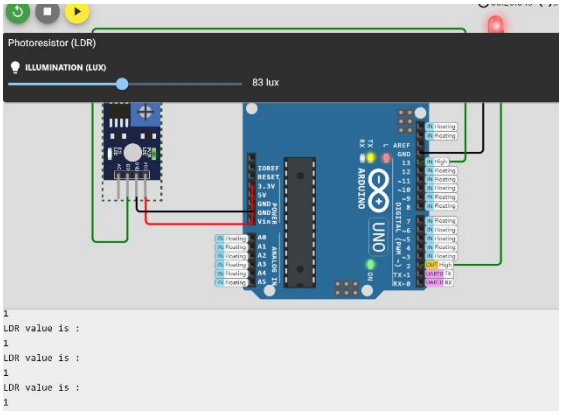


Figure: Interfacing LDR module with Arduino UNO

Output:



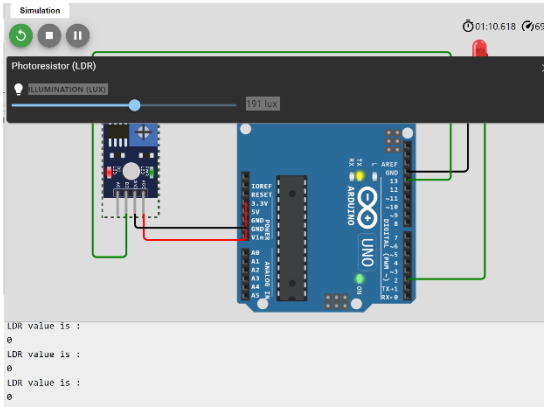


Figure: Light intensity less than 100, LED ON **Figure: Light intensity greater than 100, LED OFF**

When light intensity is less than 100, it turns on LED.

Procedure:

1. Make the connections as per the circuit diagram.
2. Check the ground connection precisely before switching on the power supply.
3. Connect Micro USB Cable to PC .
4. Write the Code.
5. Run the code through Arduino IDE and Observe the output.

Result:

We studied LDR working principle and interfacing of LDR module with Arduino UNO using digital and analog pin.

Experiment 5

5. Temperature monitoring using LM35 and Arduino.

Components and supplies:

Arduino Uno.
Jumper wires.
Bread Board(generic).
LM35
Resister 10k ohm
Software: Arduino IDE

Working Principle:

It is a 3-terminal device that provides analog voltage proportional to the temperature. Higher the temperature, higher is the output voltage. LM35 is a temperature sensor which can measure temperature in the range of -55°C to 150°C.The LM35 uses a solid-state technique to measure the temperature. It makes use of the fact that the voltage drops between the base and emitter (forward voltage – Vbe) of the Diode-connected transistor decreases at a known rate as the temperature increases. By precisely amplifying this voltage change, it is easy to generate an analog signal that is directly proportional to temperature. This linear relationship between forward voltage and temperature is the reason why diode-connected transistors are used as temperature measurement devices. Connect the left pin to 4V to 30V power supply and the right pin to ground assuming the flat side of the sensor is facing you. It just outputs a voltage that is linearly proportional to temperature.

The output analog voltage can be converted to digital form using ADC so that a microcontroller can process it. Here, LM35 output is given to analog pin A0 of Arduino UNO. This analog voltage is converted to its digital form and processed to get the temperature reading.

Program:

```
const int lm35_pin = A1;        // LM35 O/P pin
void setup() {
  Serial.begin(9600);
}
void loop() {
  int temp_adc_val;
  float temp_val;
  temp_adc_val = analogRead(lm35_pin);      //Read Temperature
  temp_val = (temp_adc_val * 4.88);  // Convert adc value to equivalent voltage //
  temp_val = (temp_val/10);    ///LM35 gives output of 10mv/°C
  Serial.print("Temperature = ");
  Serial.print(temp_val);
```

```
Serial.print(" Degree Celsius\n");
 delay(1000);
}
```

Circuit Diagram:

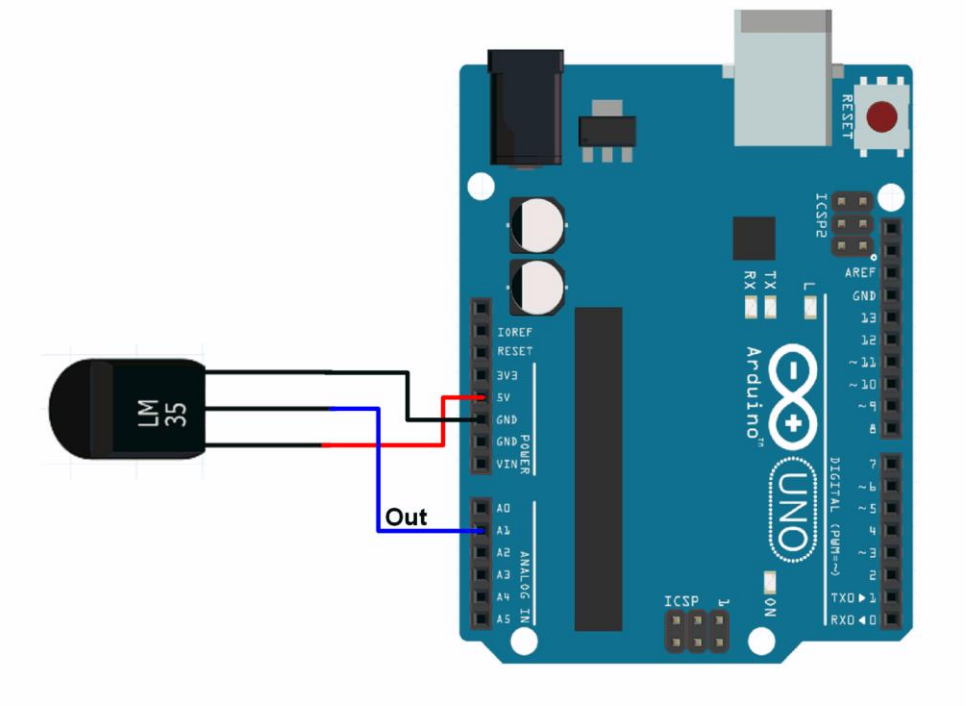


Figure: Interfacing LM35 with Arduino

Procedure:

1. Make the connections as per the circuit diagram.
2. Check the ground connection precisely before switching on the power supply.
3. Connect Micro USB Cable to PC.
4. Write the Code.
5. Run the code through Arduino IDE.

Result:

Successfully printed surrounding temperature using Arduino UNO and the LM35 sensor

Experiment 6

MQ2 sensor data accessing using Arduino.

6.a) Write a program to detect the smoke by using MQ2 Sensor.

6.b) Write a program to represent the concentration of gas present in air.

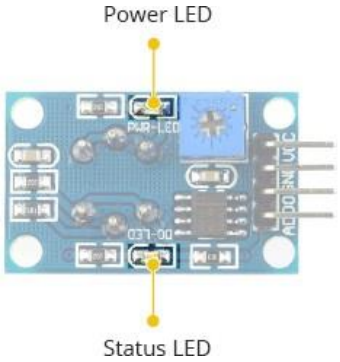
Components and supplies:

Arduino Uno.
Jumper wires.
Bread Board(generic).
MQ2 sensor.
Resister 10k ohm
Software: Arduino IDE

Working Principle:

It is a MOS (Metal Oxide Semiconductor) sensor. Metal oxide sensors are also known as Chemiresistors because sensing is based on the change in resistance of the sensing material when exposed to gasses. It is a versatile sensor that can detect LPG, smoke, alcohol, propane, hydrogen, methane, and carbon monoxide concentrations in the air. The MQ2 gas sensor operates on 5V DC and consumes approximately 800mW.

It can detect LPG, Smoke, Alcohol, Propane, Hydrogen, Methane and Carbon Monoxide concentrations ranging from 200 to 10000 ppm. the MQ2 gas sensor detects multiple gases, but cannot identify them. It is best suited for measuring changes in a known gas density rather than detecting which one is changing. The MQ2 gas sensor is simple to use and has two different outputs. It not only provides a binary indication of the presence of combustible gasses, but also an analog representation of their concentration in air. The sensor's analog output voltage (at the A0 pin) varies in proportion to the concentration of smoke/gas. The higher the concentration, the higher the output voltage; the lower the concentration, the lower the output voltage. This analog signal is digitized by an LM393 High Precision Comparator and made available at the Digital Output (D0) pin. The module includes a potentiometer for adjusting the sensitivity of the digital output (D0). You can use it to set a threshold so that when the gas concentration exceeds the threshold value, the module outputs LOW otherwise HIGH.

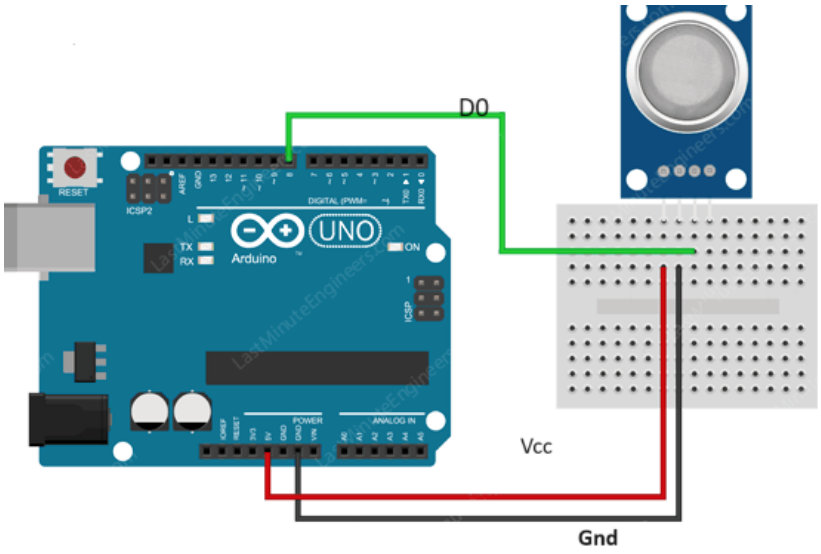


6.a) Write a program to detect the smoke by using MQ2 Sensor.

Program:

```
int MQ2_digitalout=13; //Set Digital Input D0 for Mq2.
void setup()
 {
Serial.begin(9600);
pinMode(MQ2_digitalout,INPUT);
}
void loop()
 {
int value=digitalRead(MQ2_digitalout);
Serial.println("Sensor value is :");//Prints the value of MQ2 sensor to Serial Monitor.
Serial.println(value);
if(value)
  {
    Serial.println("| Smoke detected!");
  }
  else
  {
     Serial.println(" NO Smoke");
  }
}
```

Circuit Diagram:

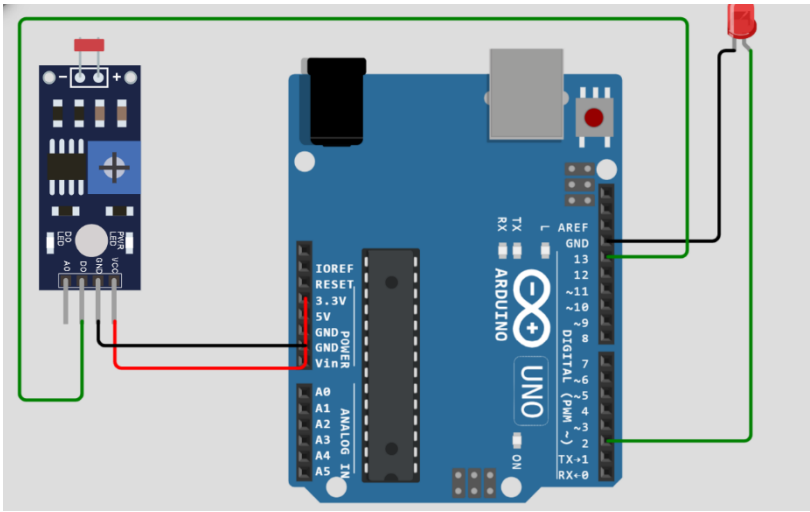


6.b) Write a program to represent the concentration of gas present in air.

Program:

```
#define MQ2pin A0
int LED=13
float sensorValue;  //variable to store sensor value

void setup()
{
        Serial.begin(9600); // sets the serial port to 9600
        pinMode(MQ2pin, INPUT);
        pinMode(13,OUTPUT);
        Serial.println("MQ2 warming up!");
        delay(20000); // allow the MQ2 to warm up
}

void loop()
{
 sensorValue = analogRead(MQ2pin); // read analog input pin 0
 Serial.print("Sensor Value: ");
 Serial.println(sensorValue);

 if(sensorValue > Threshold)
 {
  Serial.println(" | Smoke detected!");
  digitalWrite(LED,HIGH);
 }
 else
{

 Serial.println(" | Smoke detected!");
 digitalWrite(LED,HIGH);
}
 delay(2000); // wait 2s for next reading
}
```

Circuit Diagram



Procedure:

1. Make the connections as per the circuit diagram.
2. Check the ground connection precisely before switching on the power supply.
3. Connect Micro USB Cable to PC .
4. Write the Code.
5. Run the code through Arduino IDE and Observe the output.

Result:

Hence the gas sensor senses the harmful gases present around it, by using Arduino UNO.

Experiment 7

7. Ultrasonic sensor interfacing with Arduino.

Components and supplies:

Arduino Uno.
Jumper wires.
Bread Board(generic).
MQ2 sensor.
Resister 10k ohm
Software: Arduino IDE

Working Principle

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. An HC-SR04 ultrasonic distance sensor actually consists of two ultrasonic transducers. One acts as a transmitter that converts the electrical signal into 40 KHz ultrasonic sound pulses. The other acts as a receiver and listens for the transmitted pulses. When the receiver receives these pulses, it produces an output pulse whose width is proportional to the distance of the object in front. This sensor provides excellent non-contact range detection between 2 cm to 400 cm (~13 feet) with an accuracy of 3 mm. It operates on 5 volts. Figure below shows HC-SR04 sensors with its pin configuration.

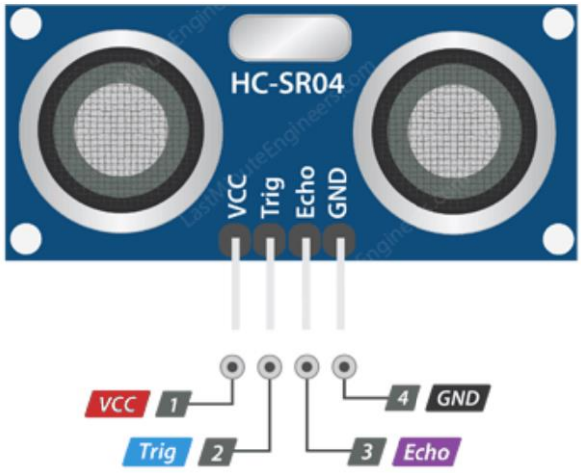


Figure: Ultrasonic(HC-SR04) Sensor

Vcc supplies power to the HC-SR04 ultrasonic sensor. You can connect it to the 5V output from your Arduino. Trig (Trigger) pin is used to trigger ultrasonic sound pulses. By setting this pin to HIGH for 10µs, the sensor initiates an ultrasonic burst. Echo pin goes high when the ultrasonic burst is transmitted and remains high until the sensor receives an echo, after which it goes low. By measuring the time the

Echo pin stays high, the distance can be calculated. GND is the ground pin.

It all starts when the trigger pin is set HIGH for 10µs. In response, the sensor transmits an ultrasonic burst of eight pulses at 40 kHz. This 8-pulse pattern is specially designed so that the receiver can distinguish the transmitted pulses from ambient ultrasonic noise. These eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the echo pin goes HIGH to initiate the echo-back signal. If those pulses are not reflected back, the echo signal times out and goes low after 38ms (38 milliseconds). Thus, a pulse of 38ms indicates no obstruction within the range of the sensor.
If those pulses are reflected back, the echo pin goes low as soon as the signal is received. This generates a pulse on the echo pin whose width varies from 150 µs to 25 ms depending on the time taken to receive the signal. The width of the received pulse is used to calculate the distance from the reflected object. This can be worked out using the simple distance-speed-time equation.

Program:

```
int trigPin = 9;   // TRIG pin
int echoPin = 8;    // ECHO pin

float duration_us, distance_cm;

void setup() {
  // begin serial port
  Serial.begin (9600);

  // configure the trigger pin to output mode
  pinMode(trigPin, OUTPUT);
  // configure the echo pin to input mode
  pinMode(echoPin, INPUT);
}

void loop() {
  // generate 10-microsecond pulse to TRIG pin
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // measure duration of pulse from ECHO pin
  duration_us = pulseIn(echoPin, HIGH);

  // calculate the distance
  distance_cm = 0.017 * duration_us;

  // print the value to Serial Monitor
  Serial.print("distance: ");
  Serial.print(distance_cm);
  Serial.println(" cm");

  delay(500);
}
```

Circuit Diagram:

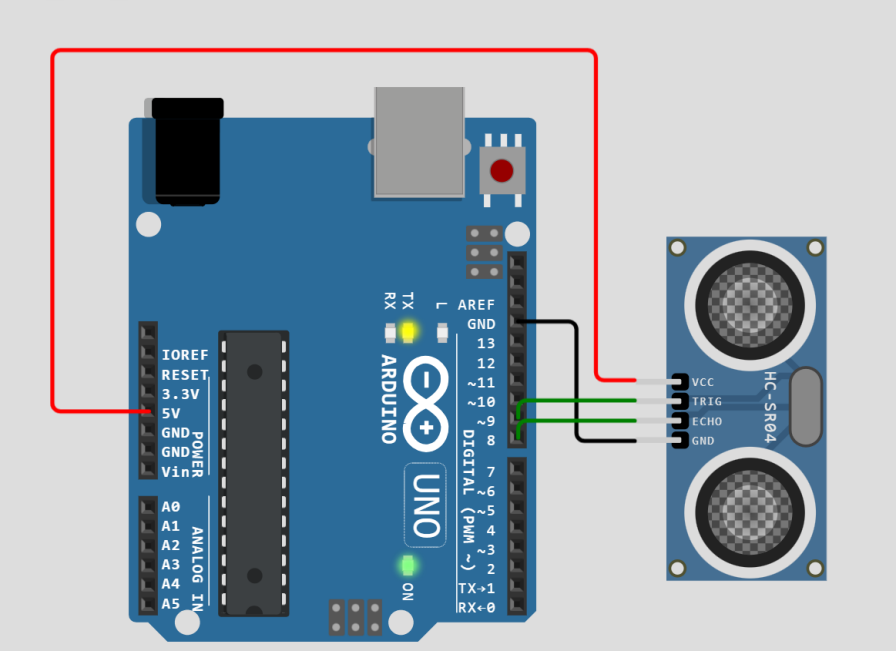


Figure: Interfacing HC-SR04 with Arduino

Procedure:

1. Make the connections as per the circuit diagram.
2. Check the ground connection precisely before switching on the power supply.
3. Connect Micro USB Cable to PC .
4. Write the Code.
5. Run the code through Arduino IDE
6. Observe the output.

Result:

We successfully interfaced HC-SR04 sensor with Arduino UNO and distance between sensor and the object in its vicinity was displayed on serial monitor.

Experiment 8

8. Create a program to blink LED in the following manner:
1010, 1100, 1001,1011

Components and supplies:

Arduino Uno.
Jumper wires.
Bread Board(generic).
4 LEDs
Resister 10k ohm
SOFTWARE: Arduino IDE

Program:

```
void setup()
{
// initialize digital pin 2 as an output.
pinMode(2, OUTPUT);
pinMode(4, OUTPUT);
pinMode(7, OUTPUT);
pinMode(13, OUTPUT);
}
// the loop function runs over and over again forever
void loop()
{
  digitalWrite(13, HIGH); // turn the LED1 on
  digitalWrite(2, LOW);  //turn the LED2 off
  digitalWrite(7, HIGH); // turn the LED3 on
  digitalWrite(4, LOW);  //turn the LED4 off
  delay(1000); // wait for a second
  digitalWrite(13, HIGH); // turn the LED1 on
  digitalWrite(2, HIGH);  //turn the LED2 on
  digitalWrite(7, LOW;  //turn the LED3 off
  digitalWrite(4, LOW);  //turn the LED4 off
  delay(1000); // wait for a second
  digitalWrite(13, HIGH); // turn the LED1 on
  digitalWrite(2, LOW);  //turn the LED2 off
  digitalWrite(7, LOW);  //turn the LED3 off
  digitalWrite(4, HIGH);  turn the LED4 on
  delay(1000); // wait for a second
  digitalWrite(13, HIGH); // turn the LED1 on
```

```
digitalWrite(2, LOW);  //turn the LED2 off
digitalWrite(7, HIGH); // turn the LED3 on
digitalWrite(4, HIGH);  //turn the LED4 on
delay(1000); // wait for a second

        }
```

Circuit Diagram:

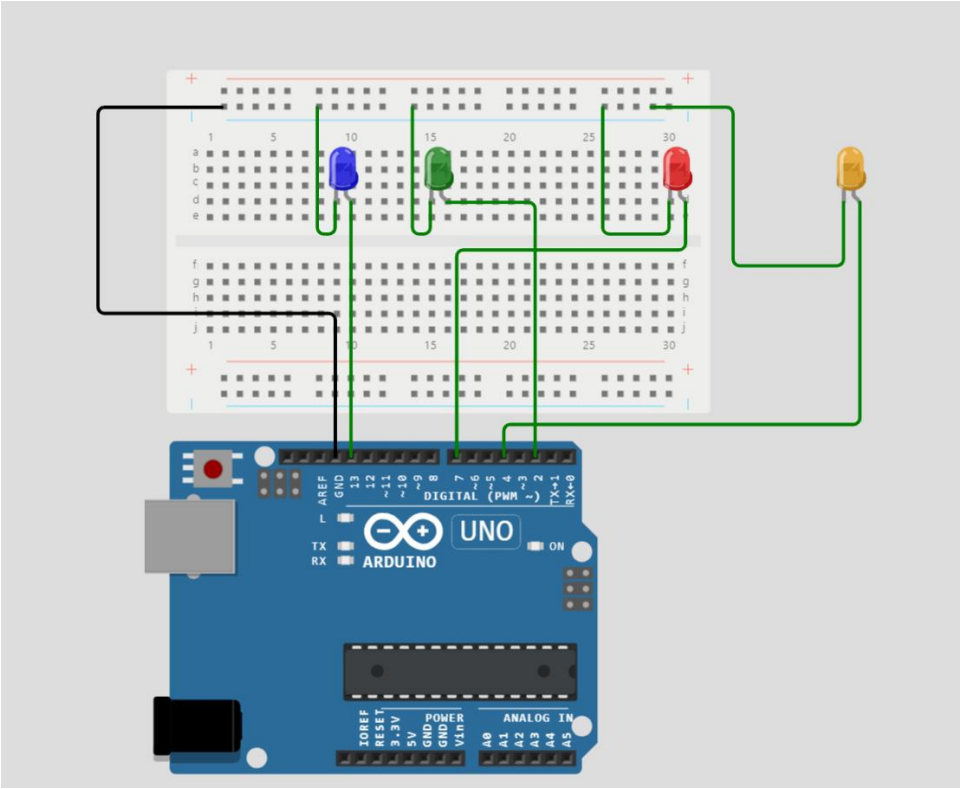


Figure: Connections for turning on-off LEDs in various patterns

Procedure:

1. Make the connections as per the circuit diagram.
2. Check the ground connection precisely before switching on the power supply.
3. Connect Micro USB Cable to PC .
4. Write the Code.
5. Run the code through Arduino IDE
6. Observe the output.

Result:

Blinking of LED in given pattern was successfully demonstrated.

Experiment 9

9.a) Playing Piezo buzzer using Arduino.

Components and supplies:

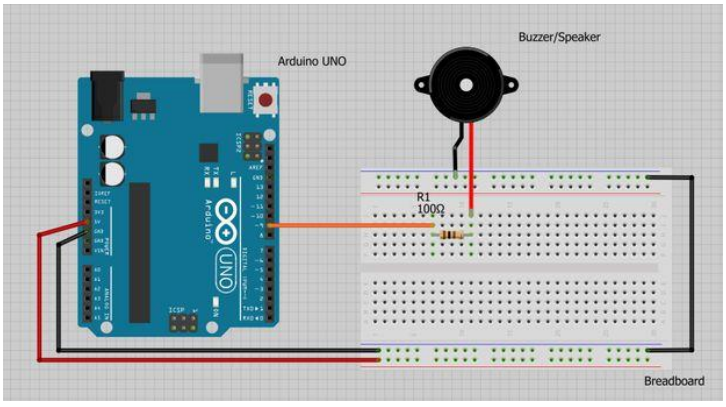
Arduino Uno.
Jumper wires.
Bread Board(generic).
MQ2 sensor.
Resister 10k ohm

Software: Arduino IDE

Program:

```
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

  }
```

Circuit Diagram:



Procedure:

1. Make the connections as per the circuit diagram.
2. Check the ground connection precisely before switching on the power supply.

3. Connect Micro USB Cable to PC .
4. Write the Code.
5. Run the code through Arduino IDE or uPycraft IDE and Observe the output.

Result:
We successfully interfaced buzzer with Arduino.

Experiment 10
10. a) Display "hello world" message using arduino LCD display.
10. b) Distance Measurement Using Ultrasonic Sensor and Displaying on LCD.

Components and supplies:

Arduino Uno.
Jumper wires.
Bread Board(generic).
16x2 LCD
Resister 10k ohm
Software: Arduino IDE

Liquid Crystal Display
An LCD character display is a unique type of display that can only output individual ASCII characters with fixed size. Using these individual characters then we can form a text. The number of the rectangular areas define the size of the LCD. The most popular LCD is the 16×2 LCD, which has two rows with 16 rectangular areas or characters. Of course, there are other sizes like 16×1, 16×4, 20×4 and so on, but they all work on the same principle.

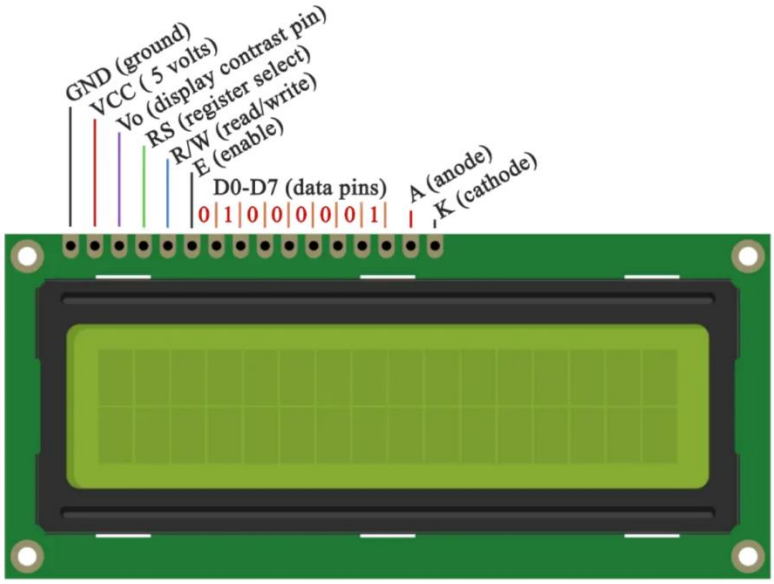


Figure: 16x2 Liquid Crystal Display

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

→ A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
→ A Read/Write (R/W) pin that selects reading mode or writing mode
→ An Enable pin that enables writing to the registers

→ 8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

There's also a display contrast pin (Vo), power supply pins (+5V and GND) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively. The LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode. To wire your LCD screen to your board, connect the following pins:

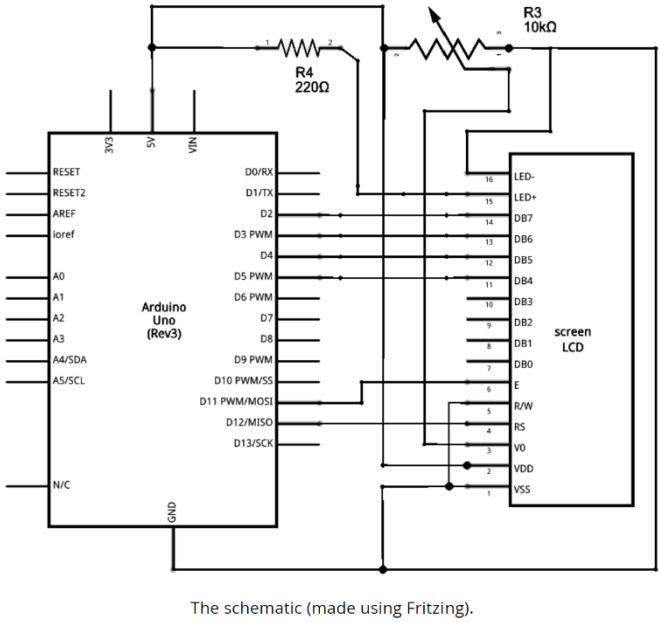
LCD RS pin to digital pin 12
LCD Enable pin to digital pin 11
LCD D4 pin to digital pin 5
LCD D5 pin to digital pin 4

LCD D6 pin to digital pin 3
LCD D7 pin to digital pin 2
LCD R/W pin to GND
LCD VSS pin to GND
LCD VCC pin to 5V
LCD LED+ to 5V through a 220 ohm resistor
LCD LED- to GND

10. a) Display “hello world” message using arduino LCD display.
Program:

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);

void setup(){

lcd.begin(16,2);
lcd.setCursor(0,0);
lcd.print("Hello World");
lcd.setCursor(0,1);
lcd.print("Welcome to MVJCE");
}
void loop()
{

}
```

Circuit Diagram:



The schematic (made using Fritzing).

Figure: Schematic for LCD and Arduino UNO

10. b) Distance Measurement Using Ultrasonic Sensor And Displaying on LCD

LCD circuit:

- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2
- LCD R/W pin to ground
- LCD VSS pin to ground
- LCD VCC pin to 5V

Program:

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //Interface pins of the LCD
const int trig_pin=8;
const int echo_pin=9;
long distance,duration;
```

```
void setup() {
lcd.begin(16,2);
lcd.setCursor(0,0); //set the cursor to column 0 and line 0



pinMode(8,OUTPUT);
pinMode(9,INPUT);
}

void loop() {
digitalWrite(8,HIGH);
delayMicroseconds(20);
digitalWrite(8,LOW);
delayMicroseconds(20);
duration = pulseIn(echo_pin, HIGH); //To receive the reflected signal.
distance= duration*0.034/2;
lcd.setCursor(0,0);
lcd.print("The distance is");
lcd.setCursor(0,1); //set the cursor to column 0 and line 1
lcd.print(distance);
lcd.print("cm");
delay(100);
}
```

Circuit Diagram:

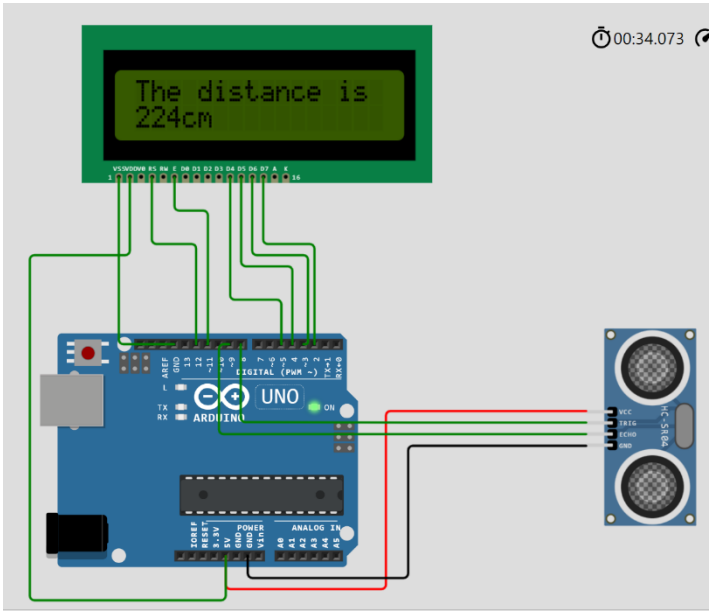


Figure: LCD and HC-SR04 interfacing with Arduino UNO

Procedure:

1. Make the connections as per the circuit diagram.

2. Check the ground connection precisely before switching on the power supply.
3. Connect Micro USB Cable to PC .
4. Write the Code.
5. Run the code through Arduino IDE or uPycraft IDE and Observe the output.

RESULT:

We successfully measured the distance using ultrasonic sensor and displayed it on 16x2 LCD.