INDEX

PR-2 IoT LAB

Practical	**4 Periods per week**	**Term Work**	**50 Marks**
Total Periods	**60 Periods**	**Term End Exam**	**50 Marks**
Examination	**3 Hours**	**TOTAL MARKS**	**100 Marks**

CONTENTS

1. Basics of C language using Arduino IDE
   - Understating basics of Arduino IDE
   - Variables, datatype, loops, control statement, function

Practical using Arduino-interfacing sensors

2. Interfacing Light Emitting Diode(LED)- Blinking LED
3. Interfacing Button and LED – LED blinking when button is pressed
4. Interfacing Light Dependent Resistor (LDR) and LED, displaying automatic night lamp
5. Interfacing Temperature Sensor(LM35) and/or humidity sensor (e.g.DHT11)
6. Interfacing Liquid Crystal Display(LCD) – display data generated by sensor on LCD
7. Interfacing Air Quality Sensor-pollution (e.g. MQ135) – display data on LCD , switch on LED when data sensed is higher than specified value.
8. Interfacing Bluetooth module (e.g. HC05)- receiving data from mobile phone on Arduino and display on LCD
9. Interfacing Relay module to demonstrate Bluetooth based home automation application. (using Bluetooth and relay).

Books Recommended:

Sl.No.	Name of the Author	Title of the Book	Name of the Publisher
1	Vijay Madisetti, Arshdeep Bahga	Vijay Madisetti, Arshdeep Bahga	UniversityPress
2	Yashavant Kanetkar, Shrirang Korde,	21 Internet Of Things (IOT) experiments	
3	Neerparaj Rai	Arduino Projects For Engineers	

EXP. NO. 1(A)

AIM OF THE EXPERIMENT : Basics of C language using Arduino IDE

I. Understanding basics of Arduino IDE

Introduction: Arduino IDE is open source software that is used to program the Arduino controller board. It is based on variations of C and C-programming language. It can be downloaded from Arduino's official Website and installed into PC.

INSTALLATION OF ARDUINO IDE:

1. **Power the board by connecting it to a PC via USB cable:** The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port. The green power LED (labelled PWR) should glow.
2. **Launch the Arduino IDE** After the Arduino IDE software is downloaded, we need to unzip the folder. Inside the folder, we can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE
3. **Open the first project .** Once the software starts, we have two options - Create a new project or open an existing project example. To create a new project, select File – New To open an existing project example, select File Example.

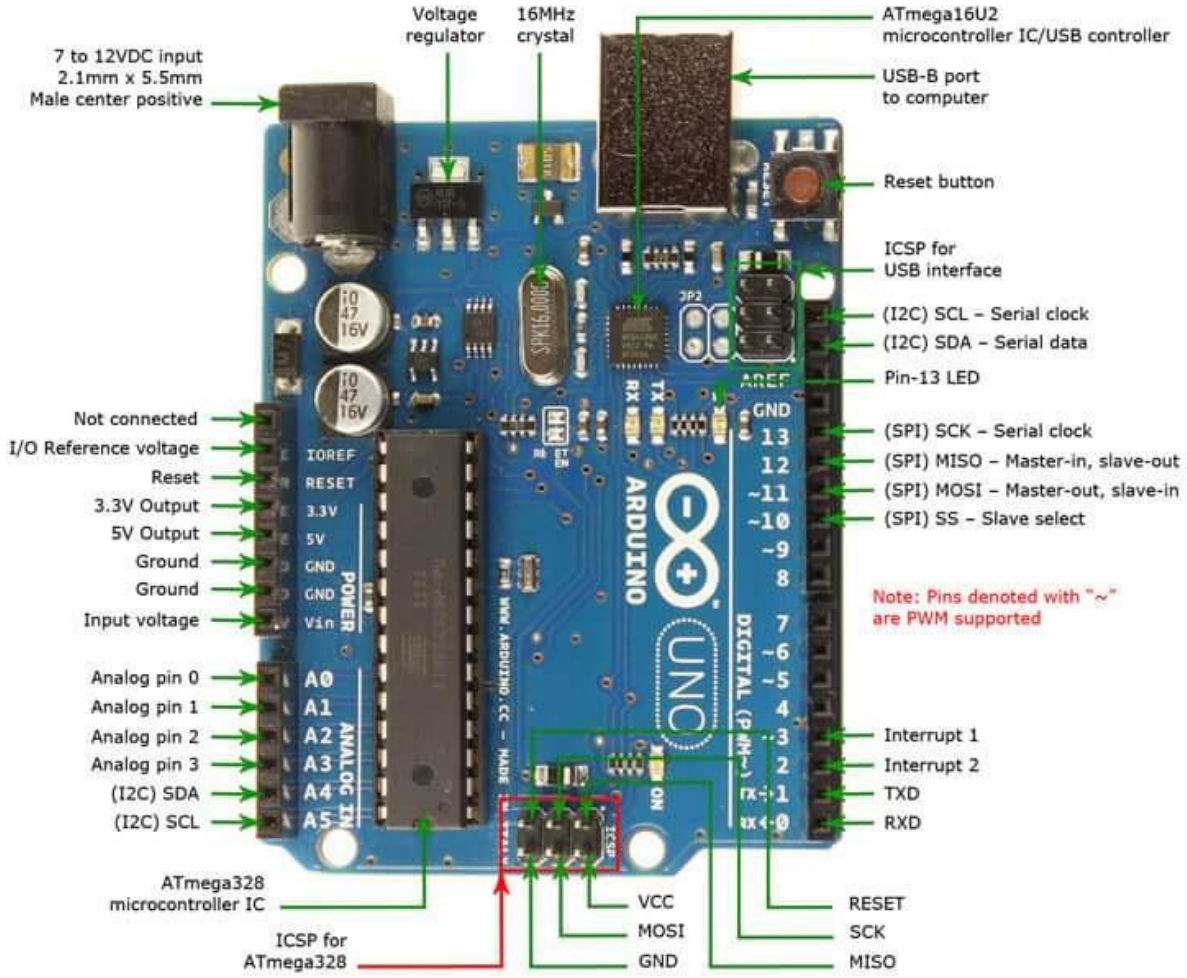
Following steps are required to set up Arduino board.

1. **Select the Arduino board**–To avoid any error while uploading our program to the board. We must select the correct Arduino board name, which matches with the board connected to your computer. Go to Tools-Board and select your board. Here, we will be selecting Arduino Uno board, but we must select the name matching the board that we are using.
2. **Select the serial port-**Select the serial device of the Arduino board: Go to Tools-SerialPort then. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports) To find out, we can disconnect our 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.
3. **Upload the program to our board -** Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar. Figure 1.0 explains various symbols on Arduino IDE toolbar.

Figure 1.0 Arduino IDE toolbar

Program coded in Arduino IDE is called a SKETCH. Following are the explanation of various symbols used in the above diagram.

A - used to check if there is any compilation error.
B - used to upload a program to the Arduino board.
C - shortcut used to create a new sketch
D - used to directly open one of the example sketch
E - used to save our sketch.
F - Serial monitor used to receive serial data from the board and send the serial data to the board.



CONCLUSION:

We successfully understood the basics of Arduino IDE and Arduino microcontroller.

EXP. NO. 1(B)

AIM OF THE EXPERIMENT: Basics of C language using Arduino IDE

II. Understanding Variables, datatype, loops, control statement, function in C using Arduino IDE.

PROGRAM ELEMENTS:

This section explains various program elements or components like structure of an Arduino sketch, variables, data types, operators, decision making, loops, functions, string objects and array.

i. Program Structure:

Arduino programs can be divided in three main parts Structure, Values (variables and constants) and Functions. Let us start with the Structure. Structure consist of two main functions-setup() function and loop() function.

a. **setup() function -** The setup() function is called when a sketch starts. It is used to utilize the variables, pin modes, start using libraries, etc. The setup() function will only run once, after each power up or reset of the Arduino board.
b. **loop() function -** After creating a setup() function, which initializes and sets the initial values, the loop() function allows our program to iteratively do the specified task in the program. This is used to actively control the Arduino hoard.

ii. Variables

A variable is a value that can change, depending on conditions or on information passed to
the program. Arduino uses C programming language. Variables use a property called scope. A scope is a region of the program and there are three places where variables can be declared. They are:

a. Variables declared inside a function or a block is called local variables.
b. Variables used in the definition of function parameters are called formal parameter
c. Variables declared outside of all functions are called global variables.

Program

```
float c = 0;//Global variable declaration
void setup () {
              }
void loop () {
   int z;//Local variable declaration
   z = 10;
}
```

iii.Data Types

Data types in C refer to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in the

storage and how the bit pattern stored is interpreted. The following data types are supported by Arduino programming: void, Boolean, char, unsigned char, byte, int, unsigned int, word, long, short, float, double etc...

iv. Decision Making

Decision making structures specify one or more conditions to be evaluated or tested by the program. It should be along with a statement or statements to be executed if the condition is determined true and optionally other statements to be executed if the condition is false. Following are the decision making statements supported by Arduino:

a. if statement
b. if .. else statement
c. if... else if...else statement
d. switch... case statement
e. conditional operator(?)

a.
```
int a=5;
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:
if (a>0)
a++;
else
a--;
}
```

c.
```
int a = 5;
void setup () {
}
void loop () {
  /* check the boolean condition */
  if (a>0) /* if condition is true then execute the following statement*/
  a++;
  else if(a<0)/* else if condition is true then execute the following
              statement*/
  a--;
  else /* else condition is true then execute the following statement*/
  a=0;
```

d.
```
Example Program
switch(phase) {
   case 0: Lo(); break;
   case 1: Mid(); break;
   case 2: Hi(); break;
   default: Message("Invalid state!");
}
```

v. Loops

A loop statement allows executing a statement or group of statements multiple times. Following are the loops supported by Arduino:

a. for loop
b. while loop
c. do.. while loop
d. nested loop

b.
```
n=10;
i=1;
while(i<=n) {
i++ //statements block will executed 10 times
}
```

c.
```
n=10;
i=1;
do{
i++ //statements block will executed 10 times
} while(i<=n);
```

d.
```
for(i=1;i<=10;i++) {
//statements block will executed 10 times
for(j=0;j<=99;j++) {
   //statements block will executed 100 times
}
}
```

vi. Functions

There are two required functions in an Arduino sketch or a program i.e. setup() and other functions must be created outside the brackets of these two functions. The no common syntax to define a function is as follows.

```
int sum(int,int ); // function prototype
void setup () {
   statements // group of statements
}
void loop () {
   int result = 0;
   result = sum(2,3); // function call
}

int sum(int x,int y) // function declaration {
   int z = 0;
   z = x+y ;
   return z; // return the value
}
```

CONCLUSION:

We successfully understood the implementation of variables, datatypes, loops, control statement, function in C using Arduino IDE.

EXP. NO. 2

AIM OF THE EXPERIMENT: Interfacing Light Emitting Diode(LED)- Blinking LED.

INTRODUCTION:

LEDs are small, powerful lights that are used in many different applications. To start, we will work on blinking an LED, the Hello World of microcontrollers. It is as simple as turning a light on and off. Establishing this important baseline will give you a solid foundation as we work towards experiments that are more complex.

COMPONENTS REQUIRED:

We need the following components –

- 1 × USB Cable
- 1 × Breadboard
- 1×ArduinoUnoR3
- 1×LED
- 1 × 330Ω Resistor
- 2×Jumper

PROCEDURE:

•Follow the circuit diagram and hook up the components on the Arduino board as shown in the image given below.

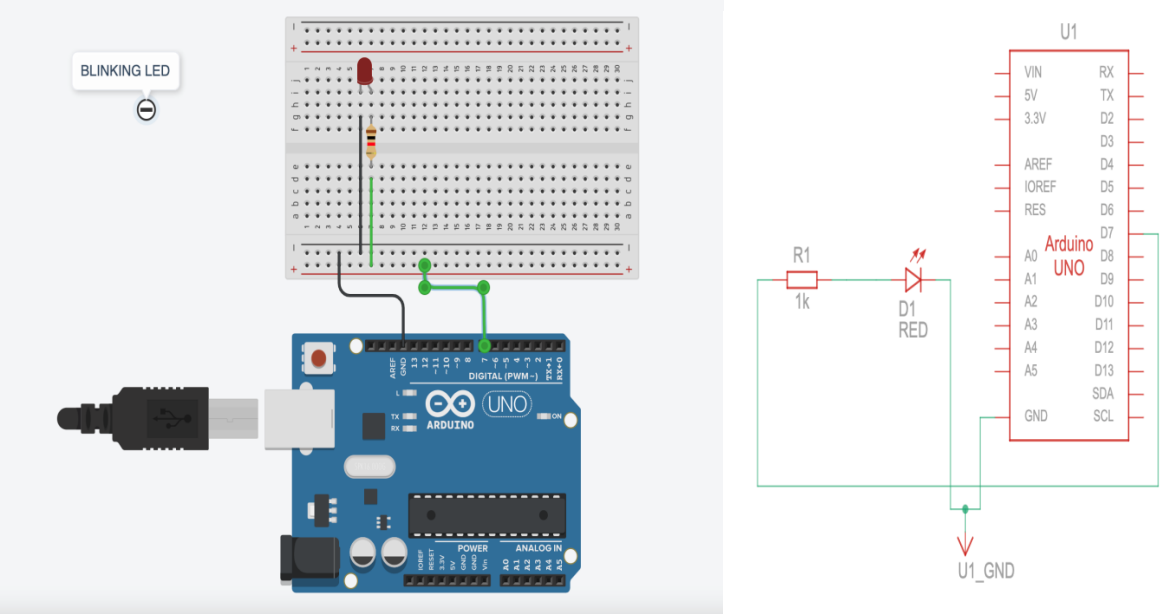
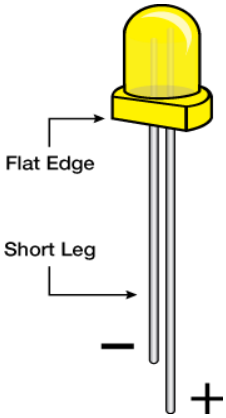
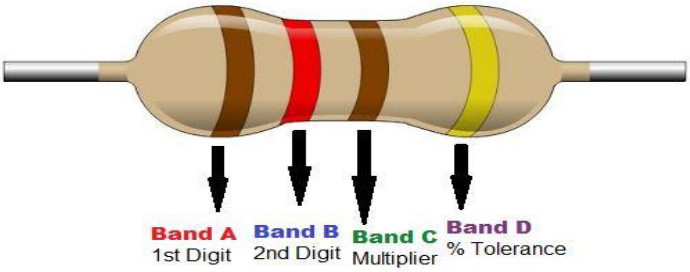


Fig. : Arduino circuit diagram for Blinking LED when OFF

Please note: Pay close attention to the LED. The negative side of the LED is the short leg, marked with a flat edge.



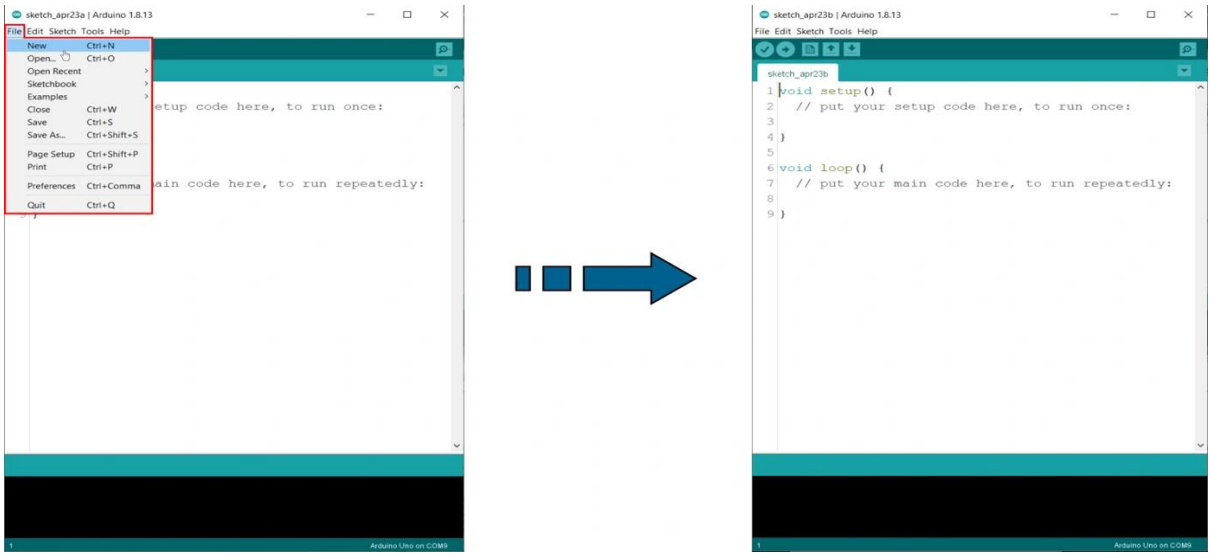
Components like resistor helps to limit or regulate the flow of electrical current in an electronic circuit.



SKETCH:

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

To open the code go to: **File > New**



ARDUINO CODE:

```
/*

  This program blinks pin 13 of the Arduino (the
  built-in LED)
*/
voidsetup()
{
pinMode(13, OUTPUT);
}

voidloop()
{
  // turn the LED on (HIGH is the voltage level)
digitalWrite(13, HIGH);
delay(1000); // Wait for 1000 millisecond(s)
  // turn the LED off by making the voltage LOW
digitalWrite(13, LOW);
delay(1000); // Wait for 1000 millisecond(s)
}
```

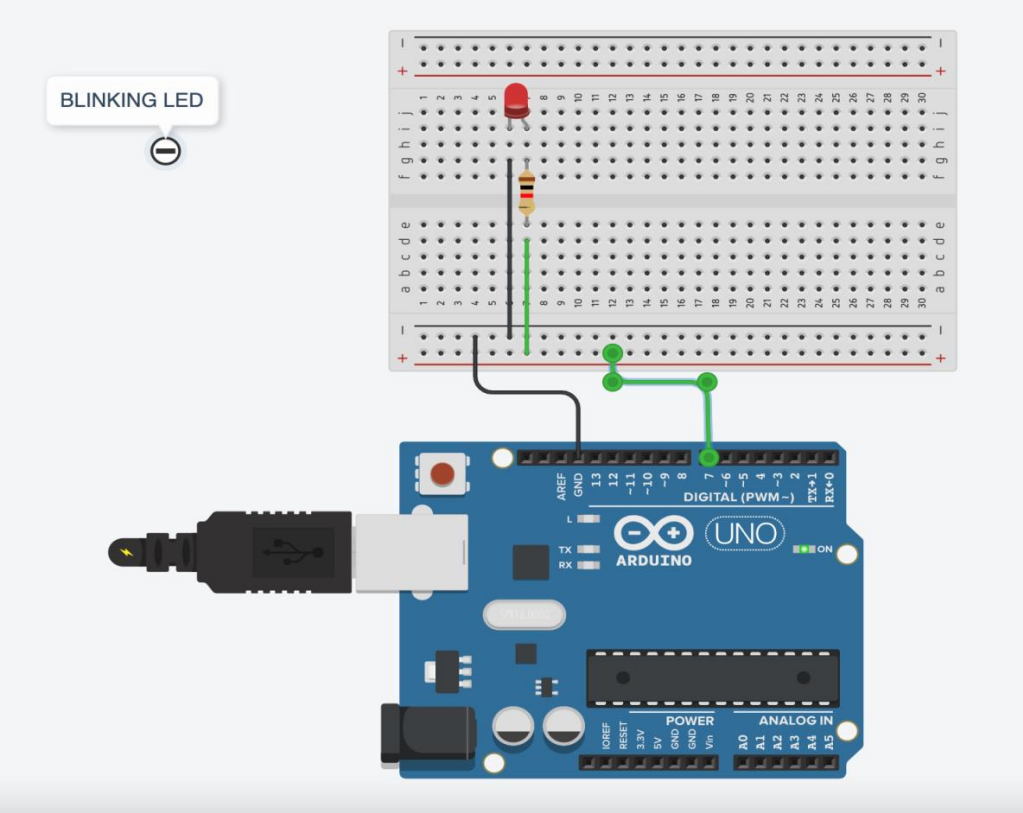


Fig. : Arduino circuit diagram for Blinking LED when OFF

CODE TO NOTE:

- **pinMode(13, OUTPUT) –** Before you can use one of Arduino's pins, you need to tell Arduino Uno R3 whether it is an INPUT or OUTPUT. We use a built-in "function" called pinMode() to do this.

- **digitalWrite(13, HIGH) –** When you are using a pin as an OUTPUT, you can command it to be HIGH (output 5 volts), or LOW (output 0 volts).

CONCLUSION:

We have successfully studied Interfacing Light Emitting Diode(LED)- Blinking LED. We should see your LED turn on and off. If the required output is not seen, make sure you have assembled the circuit correctly, and verified and uploaded the code to your board.

EXP.NO. 3

AIM OF THE EXPERIMENT: Interfacing Button and LED – LED blinking when pushbutton is pressed.

INTRODUCTION:

This experiment demonstrates the use of a push button to operate an LED. In this circuit, we'll be reading in one of the most common and simple inputs – a push button – by using a digital input. The way a push button works with your Arduino Uno R3 is that when the button is pushed, the voltage goes LOW. Your Arduino Uno R3 reads this and reacts accordingly.

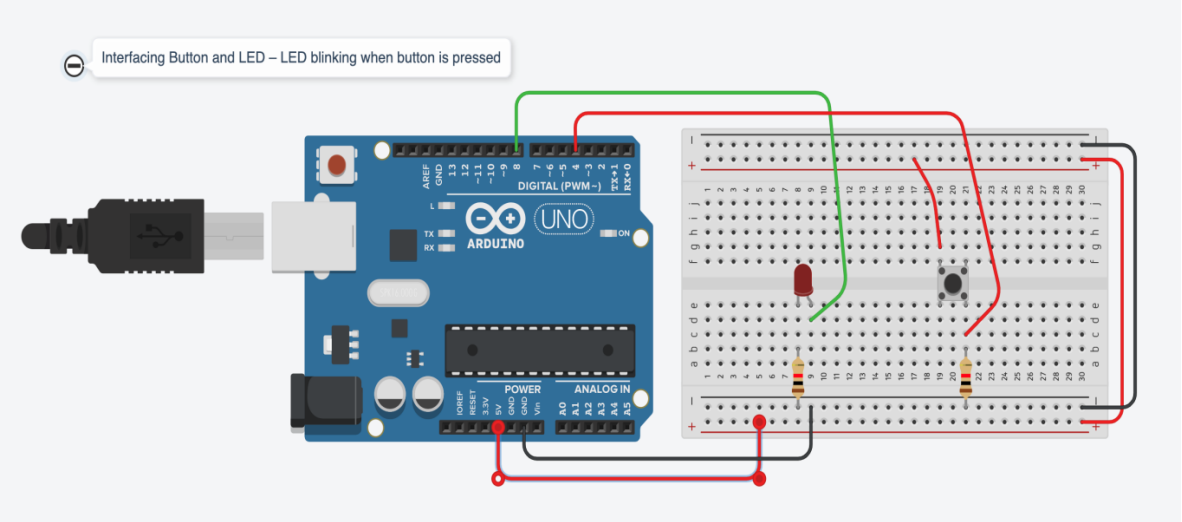
COMPONENTS REQUIRED:

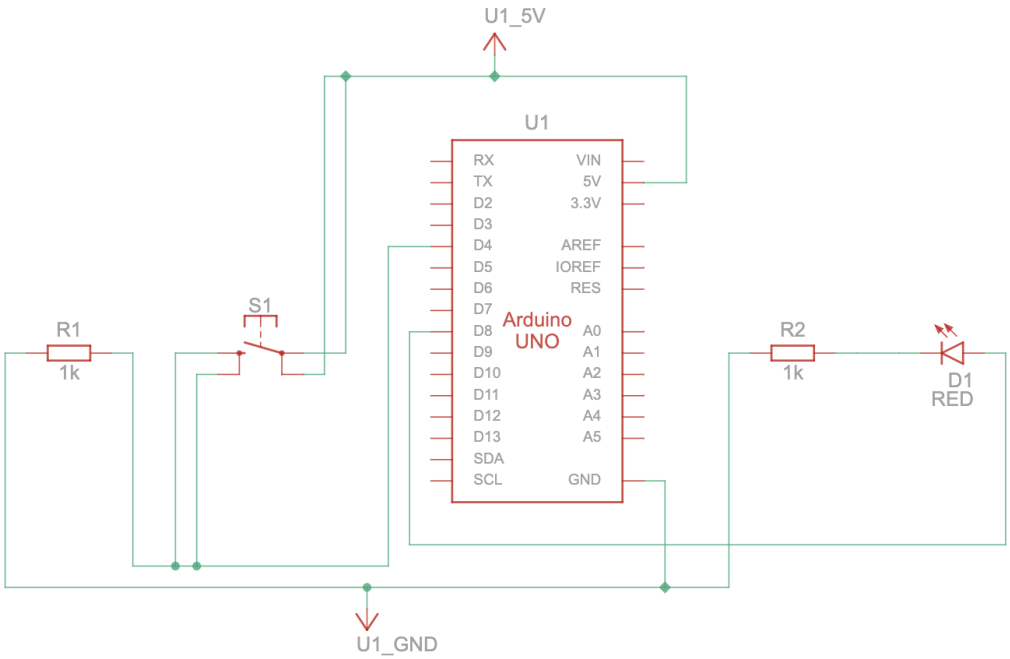
We need the following components –

- **1x** Breadboard
- **1x** Arduino Uno R3
- **1x** LED
- **7x** Jumper Wires
- **1x** Push Buttons
- **2x** 1kΩ Resistors

PROCEDURE:

•Follow the circuit diagram and hook up the components on the Arduino board as shown in the image given below.





ARDUINO CODE:

```
/*
This program blinks the LED connected to pin 8 of the Arduino
when pin 4 connected to push button is pressed
*/
void setup(){
pinMode(4,INPUT);
pinMode(8, OUTPUT);
  }
void loop(){
  if (digitalRead(4)==HIGH)
 {
digitalWrite(8,HIGH);
 }
  else {
digitalWrite(8,LOW);
  }
  delay(10);
}
```

CODE TO NOTE:

- **pinMode(4, INPUT);**
  The digital pins can be used as inputs as well as outputs. Before you do either, you need to tell the Arduino which direction you're going.

- **digitalRead(4)**
  To read a digital input, you use the digitalRead() function. It will return HIGH if there's 5V present at the pin, or LOW if there's0V present at the pin.

- **if (digitalRead(4) == HIGH)**
  Itwill read HIGH when it's being pressed and LOW when it's not being pressed . Here we're using the "equivalence" operator ("==") to see if the button is being pressed.
- **digitalWrite(8, HIGH) /digitalWrite(8, LOW)**
  When you are using a pin as an OUTPUT, you can command it to be HIGH (output 5 volts), or LOW (output 0 volts).

CONCLUSION:

We have successfully studied Interfacing Button and LED – LED blinking when push button is pressed. You should see the LED turn ON if you press the push button and the LED turns OFF if you do not press the push buttons.If the required output is not seen, make sure you have assembled the circuit correctly, and verified and uploaded the code to your boardor see the troubleshooting section.

EXP. NO. 4

AIM OF THE EXPERIMENT: Interfacing Light Dependent Resistor (LDR) and LED, displaying automatic night lamp.

INTRODUCTION:

A LDR (Light Dependent Resistor) or a photo resistor is a photo conductive sensor. It is a variable resistor and changes its resistance in a proportion to the light exposed to it. It resistance decreases with the intensity of light.

It senses the light intensity from surroundings and find whether its day or night then it automatically turns ON when the surrounding is dark and it turns OFF when it receives light from surroundings.

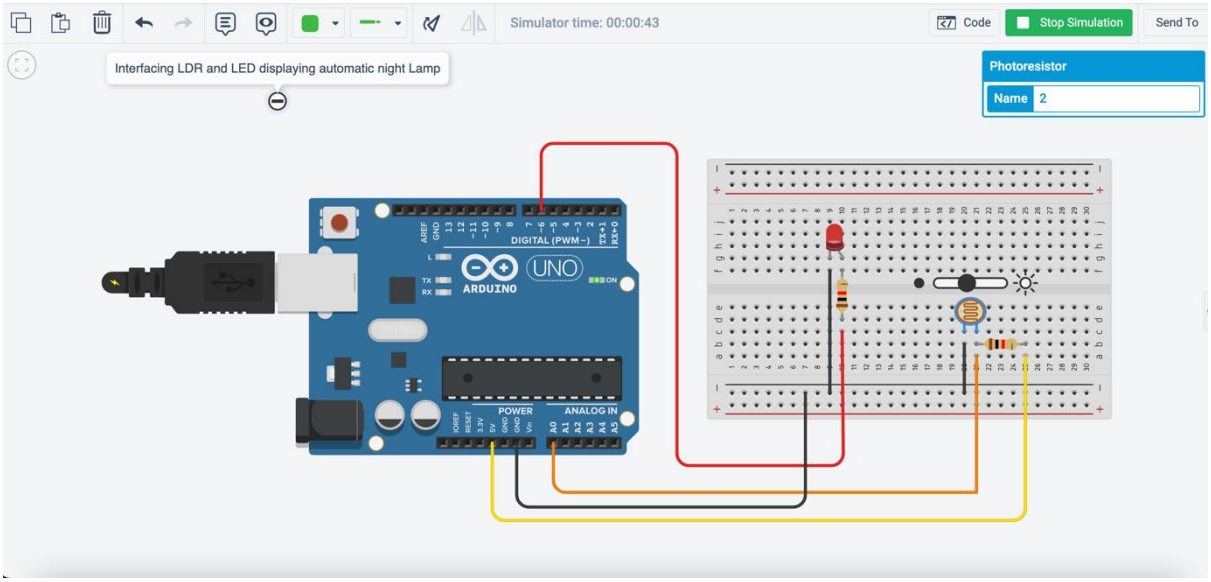
COMPONENTS REQUIRED:

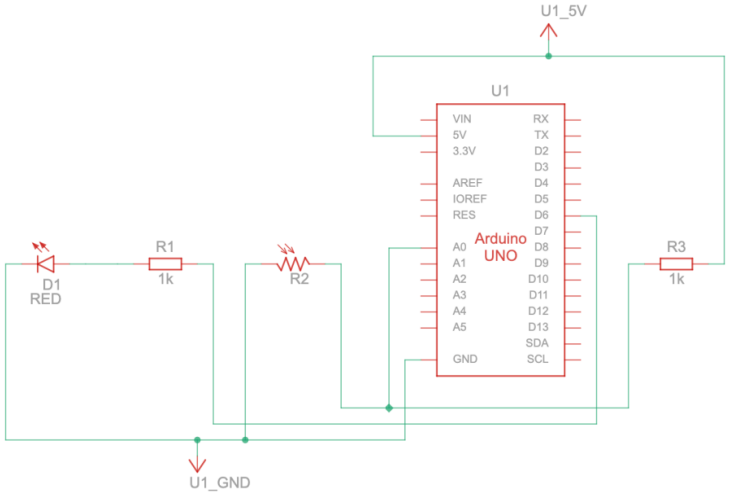
We need the following components –

- **1X** Arduino UNO R3
- **1X** LDR light sensor
- **2X** Resistor (100k)
- **1X**LED
- **1X**Jumper wire

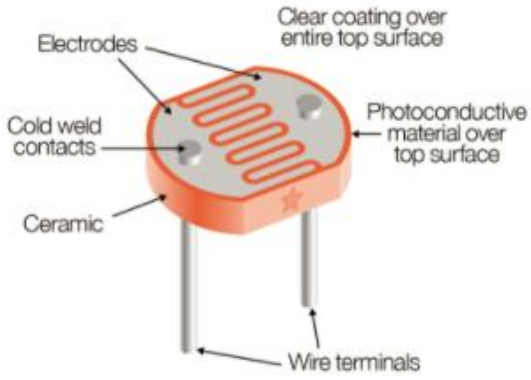
PROCEDURE:

Follow the circuit diagram and hook up the components on the Arduino board as shown in the image given below.





Please note: Photoresistor has two pins. Since it is a kind of resistor we do not need to distinguish these pins. They are symmetric.



ARDUINO CODE:

```
/*

This program uses photoresistorto sense the brightness and
accordingly switches ON/OFF the bulb automatically.
*/
const int LED=6;          // LED connect to Digital Pin
const int LDRSensor= A0;    //Sensor pin connects to analog pin A3

void setup()
{
pinMode (A3, INPUT);
pinMode (6, OUTPUT);
Serial.begin(9600);
}

void loop()
{
Serial.print("Brightness:");
```

```
Serial.println(analogRead(A0));      //sensor reading value stored in A3
 if (analogRead(A0) >600)            //if the light is above the threshold value, the LED
will turns on
  {
digitalWrite(12, HIGH);
        Serial.print("Night- Switch ON Light\n");
        delay(1000);
  }
 else
  {
digitalWrite(12, LOW);              //otherwise, the LED is off
        Serial.print("Day- Switch OFF Light\n");
        delay(1000);
  }
}
```

CODE TO NOTE:

- **const int LED=6;**
  It denotes a digital pin that will supply a trigger output of sufficient voltage.
- **const int LDRSensor= A0;**
  It is used to denote an analog pin that will receive the analog value from the circuit .
- **Serial.println(analogRead(A0));**
  We get this analog value from the analog pin A3 and store its value, then print its value in a new line.
- **if (analogRead(A0) >600)**
  Arduino senses the value from LDR in the range 0 to 1023. When the LDR value is less than600 then we call it as day condition and the bulb remains off. When the value is greater than 600 then we turn on the bulb.

Example of what you should see in the Arduino IDE's serial monitor:

```
Brightness:435
Day- Switch OFF Light
Brightness:464
Day- Switch OFF Light
Brightness:661
Night- Switch ON Light
Brightness:1017
Night- Switch ON Light
Brightness:700
Night- Switch ON Light
Brightness:516
Night- Switch ON Light
Brightness:449
Day- Switch OFF Light
```

CONCLUSION:

We have successfully studied Interfacing Light Dependent Resistor (LDR) and LED, displaying automatic night lamp. TheLED turns ON or OFF in accordance with how much light your photoresistor is reading. If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting section.

EXP. NO. 5

AIM OF THE EXPERIMENT: Interfacing Temperature Sensor(LM35)

INTRODUCTION:

A temperature sensor is exactly what it sounds like – a sensor used to measure ambient temperature. This particular sensor has three pins – a positive, a ground, and a signal. This is a linear temperature sensor. A change in temperature of one degree centigrade is equal to a change of 10 millivolts at the sensor output.

The TMP35 sensor has a nominal 750 mV at 25°C (about room temperature). In this circuit, you'll learn how to integrate the temperature sensor with your Arduino Uno R3, and use the Arduino IDE's serial monitor to display the temperature.

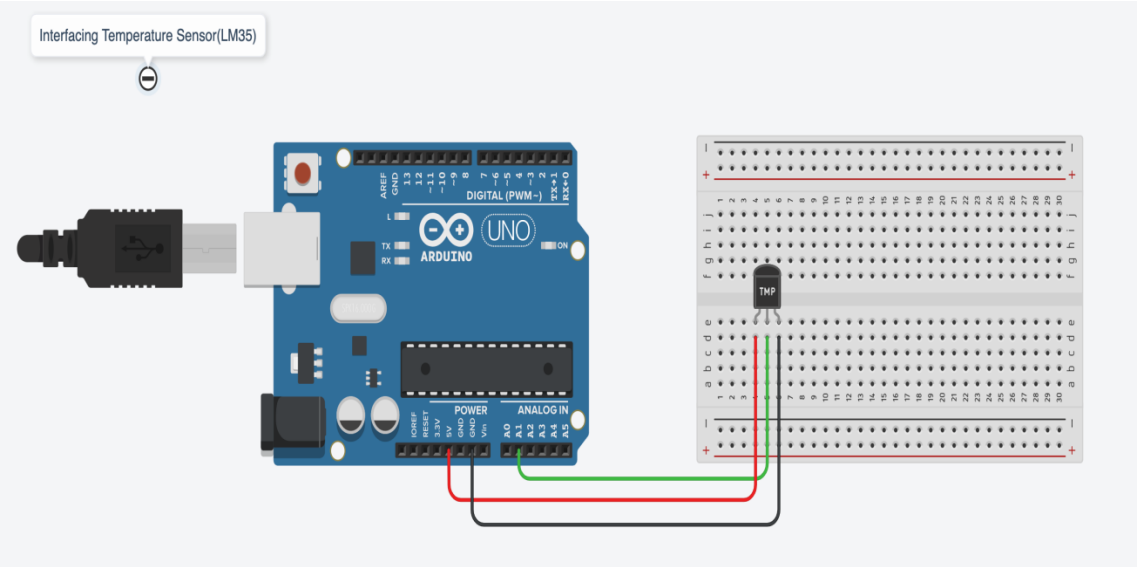
COMPONENTS REQUIRED:

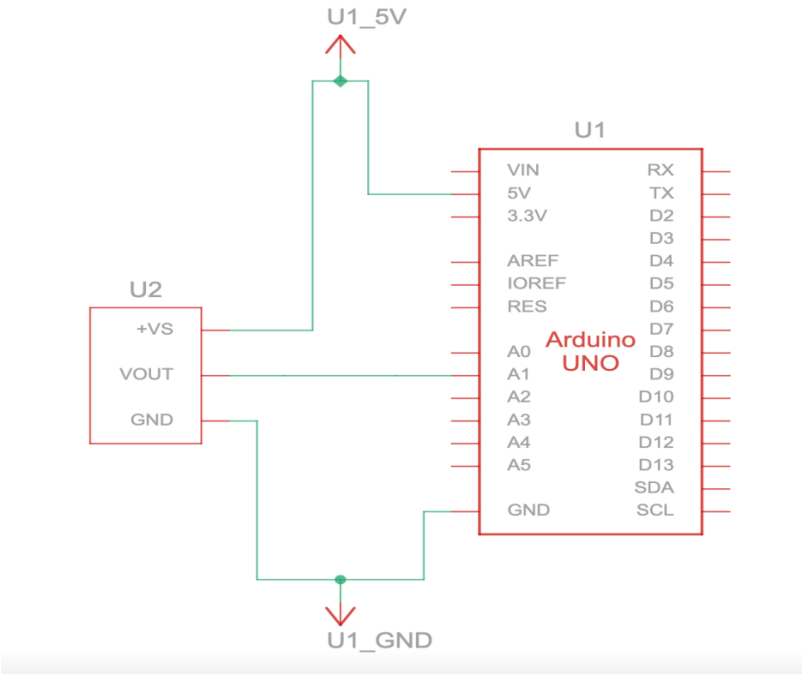
We need the following components –

- 1x Breadboard
- 1x Arduino Uno R3
- 3x Jumper Wires
- 1x Temperature Sensor

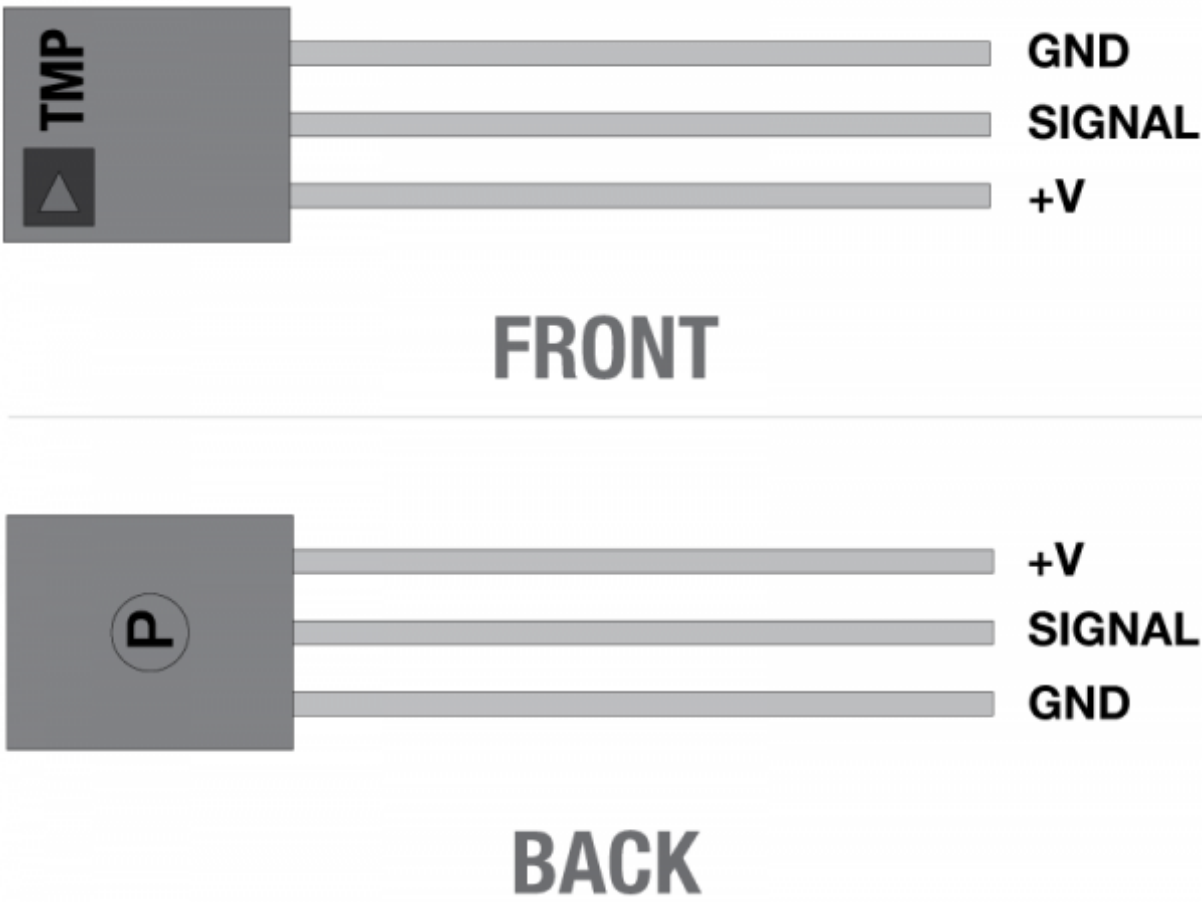
PROCEDURE:

•Follow the circuit diagram and hook up the components on the Arduino board as shown in the image given below.





Please note: The temperature sensor can only be connected to a circuit in one direction. See below for the pin outs of the temperature sensor - TMP35



ARDUINO CODE:

```
/*
This program uses temperature sensor to read analog input and
convert to its temperature equivalent i.e. digital output
*/
float temp;
int tempPin = 0;

void setup() {
Serial.begin(9600);
}

void loop() {
  temp = analogRead(tempPin);      //read analog volt from sensor and save to variable
                                   temp
  temp = temp * 0.48828125;        // convert the analog volt to its temperature
                                   equivalent

Serial.print("TEMPERATURE = ");
Serial.print(temp);              // display temperature value
Serial.print("*C");
Serial.println();
  delay(1000);           // update sensor reading each one second
}
```

CODE TO NOTE:

- **Serial.begin(9600);**
  Before using the serial monitor, you must call Serial.begin() to initialize it. 9600 is the "baud rate", or communications speed. When two devices are communicating with each other, both must be set to the same speed.

- **temp = analogRead(tempPin);**
  It will read analog volt from sensor and save to variable temp.

- **temp = temp * 0.48828125;**
  It will convert the analog volt to its temperature equivalent.

- **Serial.print("TEMPERATURE = "); and Serial.print("*C");**
  The Serial.print() command is very smart. It can print out almost anything you can throw at it, including variables of all types, quoted text (AKA "strings"), etc. See http://arduino.cc/en/serial/print for more info.

- **Serial.print(temp);**
  It will display the temperature value

- **Serial.println()**
  will move to the next line. By using both of these commands together, you can create easy-to-read printouts of text and data.

Example of what you should see in the Arduino IDE's serial monitor:

```
TEMPERATURE = 76.66*C
TEMPERATURE = 157.71*C
TEMPERATURE = 30.27*C
TEMPERATURE = 311.04*C
TEMPERATURE = 148.44*C
TEMPERATURE = 67.87*C
TEMPERATURE = 164.55*C
TEMPERATURE = 20.51*C
```

CONCLUSION:

We have successfully studied Interfacing Temperature Sensor(LM36) to the Arduino board. You should be able to read the temperature now, the temperature sensor is detecting on the serial monitor in the Arduino IDE. If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting section.

EXP. NO. 6

AIM OF THE EXPERIMENT:

Interfacing Liquid Crystal Display(LCD) – display data generated by sensor on LCD

INTRODUCTION:

In this experiment, you'll learn about how to use an LCD and a potentiometer. An LCD, or liquid crystal display, is a simple screen that can display commands, bits of information, or readings from your sensor - all depending on how you program your board. In this circuit, you'll learn the basics of incorporating an LCD into your project.

A potentiometer is also known as a variable resistor. When powered with 5V, the middle pin outputs a voltage between 0V and 5V, depending on the position of the knob on the potentiometer. A potentiometer is a perfect demonstration of a variable voltage divider circuit. The voltage is divided proportionate to the resistance between the middle pin and the ground pin.

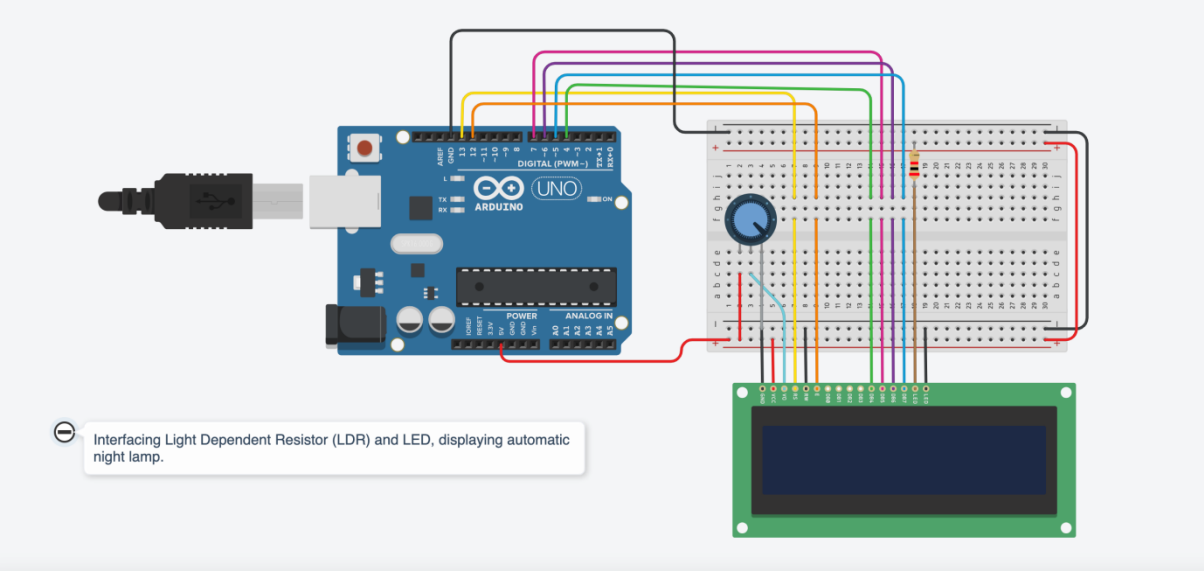
COMPONENTS REQUIRED:

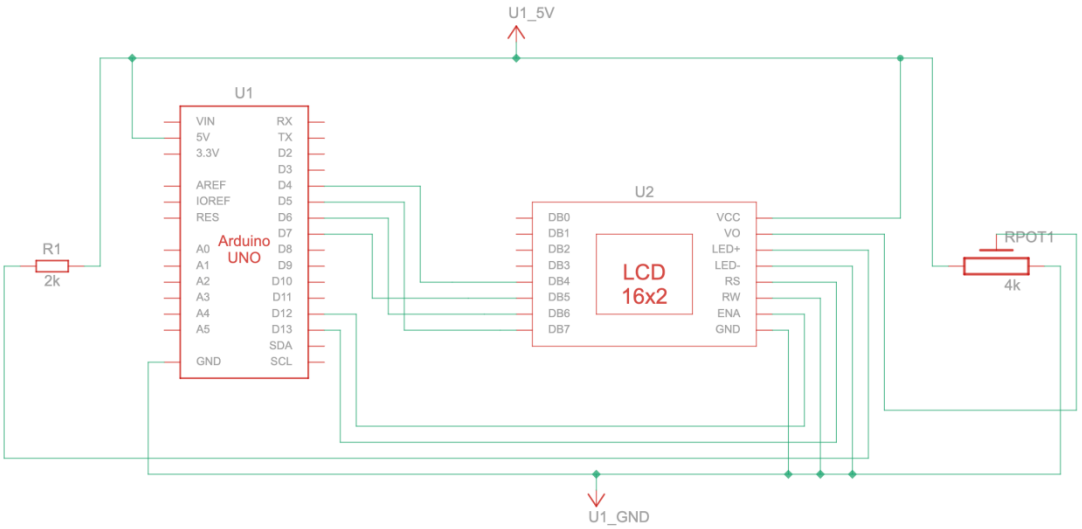
We need the following components –

- **1x** Breadboard
- **1x** Arduino Uno R3
- **1x** Potentiometer
- **1x** LCD
- **16x** Jumper Wires

PROCEDURE:

Follow the circuit diagram and hook up the components on the Arduino board as shown in the image given below.





Please note: Here Potentiometer is used to regulate the contrast of the LCD screen.

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 (V0)**, **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.

ARDUINO CODE:

```
/*
This program uses LCD to sense the brightness and accordingly
switch ON/OFF the bulb automatically.
*/
#include <LiquidCrystal.h>
int seconds = 0;
char txt1[]="hello World!";
LiquidCrystal lcd(13, 12, 4, 7, 6, 5);
void setup()
{
lcd.begin(16, 2);        // Set up the number of columns and rows on the LCD.
```

```
// set the cursor to column 0, line 0
  // (note: line 0 is the first row, since countingbegins with 0):
lcd.setCursor(0, 0);
lcd.print(txt1);            // Print a message to the LCD.
}
void loop()
{
// set the cursor to column 0, line 1
  // (note: line 1 is the second row, since countingbegins with 0):
lcd.setCursor(0, 1);
lcd.print(seconds);    // print the number of seconds since reset
  delay(1000); // Wait for 1000 millisecond(s)
  seconds += 1;
}
```

CODE TO NOTE:

- **#include <LiquidCrystal.h>**
  This bit of code tells your Arduino IDE to include the library for a simple LCD display. Without it, none of the commands will work, so make sure you include it!

- **LiquidCrystal lcd(13, 12, 4, 7, 6, 5);**
  Next we have the line that we had to modify. This defines which pins of the Arduino are to be connected to which pins of the display.
  The circuit:
  * LCD RS pin to digital pin 13
  * LCD Enable pin to digital pin 12
  * LCD D4 pin to digital pin 4
  * LCD D5 pin to digital pin 5
  * LCD D6 pin to digital pin 6
  * LCD D7 pin to digital pin 7
  * LCD R/W pin to ground
  * 2K resistor:ends to +5V and ground
  * wiper to LCD VO pin (pin 3)

- **lcd.begin(16, 2);**
  This function sets the dimensions of the LCD.A 16,2 LCD means it can display 16 characters per line and there are 2 such lines.

- **lcd.print(txt1);**
  This is the first time you'll fire something up on your screen. You may need to adjust the contrast to make it visible. Twist the potentiometer until you can clearly see the text.

- **lcd.setCursor(0, 0); /lcd.setCursor(0, 1);**
  The syntax of using the setCursor() function is as simple as just calling the function using "lcd" and mentioning the column and the row number where you want to set the cursor to start displaying the output.

- **lcd.print(seconds);**
  It will display the number of seconds or duration of the simulation time since reset.

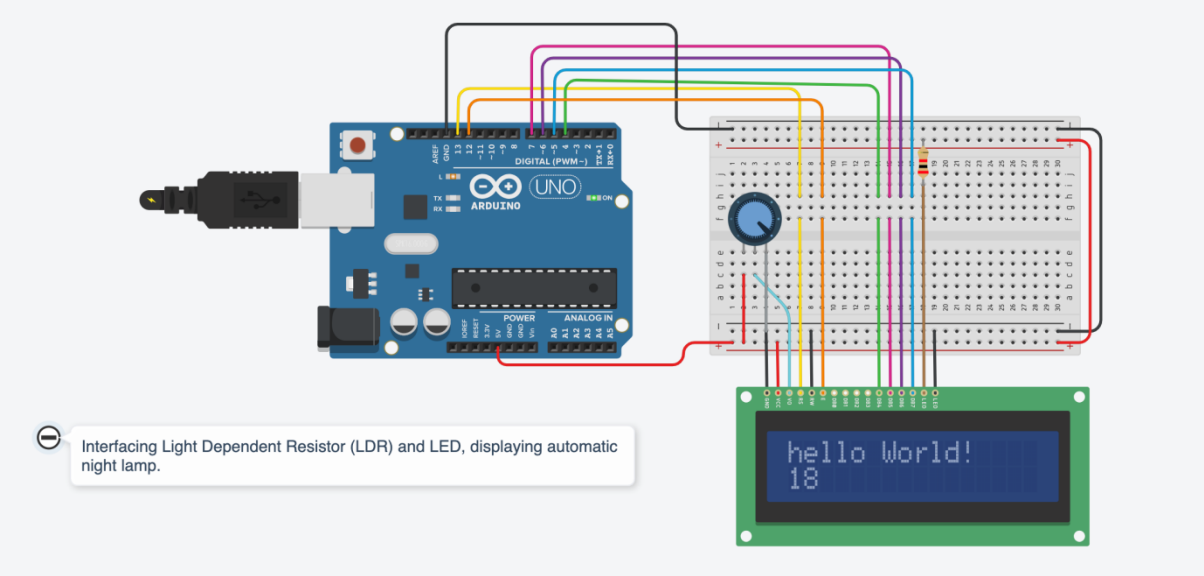


FIG. Output displayed when power supplied to the board

CONCLUSION:

We have successfully studied InterfacingLiquid Crystal Display(LCD) – display data generated by sensor on LCD.Initially, you should see the words “hello, world!” pop up on your LCD screen in the first line and displays number of seconds since reset in the second line. Remember you can adjust the contrast using the potentiometer. If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting section.

EXP. NO. 7

AIM OF THE EXPERIMENT:

Interfacing Air Quality Sensor-pollution (e.g. MQ135) – display data on LCD , switch on LED when data sensed is higher than specified value.

INTRODUCTION:

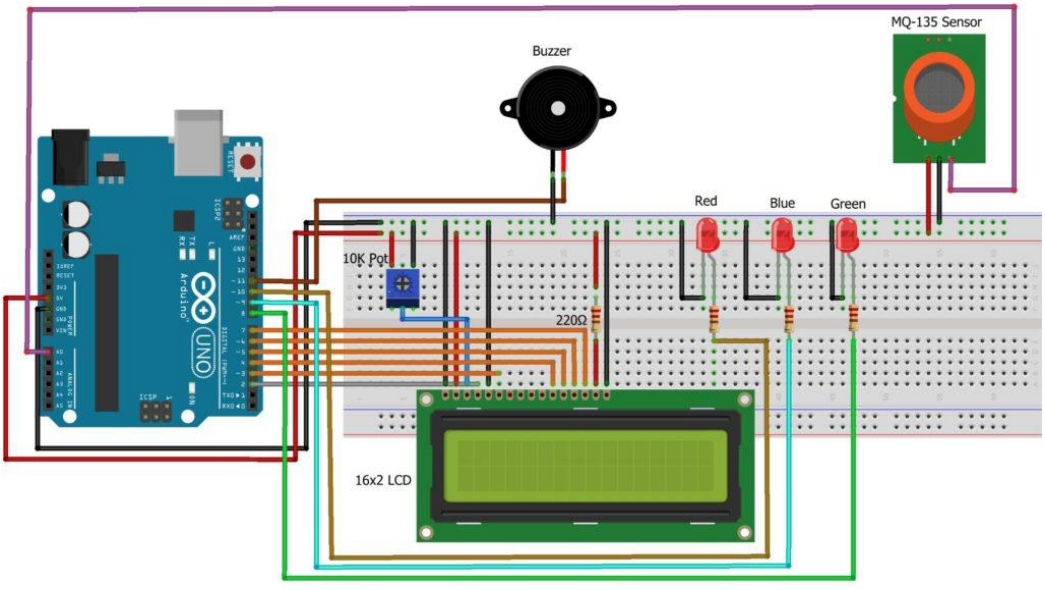
In this experiment, we will learn how to interface MQ-135 Gas Sensor with Arduino Board. MQ-135 sensor belongs to the MQ series that are used to detect different gasses present in the air. The MQ-135 sensor is used to detect gases such as NH3,NOx, alcohol, Benzene, smoke,CO2 ,etc. steel exoskeleton houses a sensing device within the gas sensor module. Arduino board is the main microcontroller board of this system. The gas sensor continuously measures air quality and sends data to the Arduino board. Then Arduino prints air quality value on the OLED display in the PPM unit. The LEDs and Buzzer used as indicators, that indicates the air quality is in good, Poor, or dangerous zone.

COMPONENTS REQUIRED:

- **1x**Arduino Nano
- **1x**MQ135 Gas Sensor
- **1x** LCDdisplay
- **3x** LEDs
- **10x** Jumper wires
- **1x** Breadboard

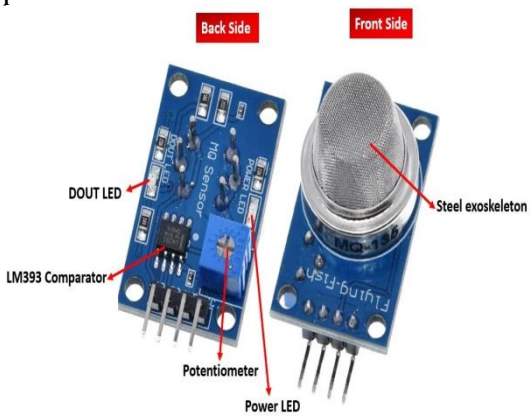
PROCEDURE:

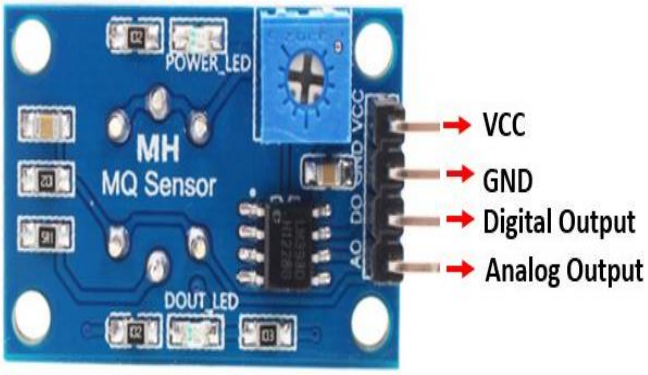
Follow the circuit diagram and hook up the components on the Arduino board as shown in the image given below.



The MQ-135 gas sensor module's front side consists of the sensor enclosed in a steel exoskeleton. This is to protect the sensor as it gets heated up over time and will eventually come in contact with the gas vapours.

The backside of the module consists of two LEDs: D_{OUT} LED (lights when D_O is LOW) and the power LED (lights when the module is powered on), an LM393 comparator, and a potentiometer.





Pinout

The MQ-135 sensor module consists of four pins namely VCC, GND, DO, and DO. The table below gives a brief description of them.

Pin	Description
VCC	Positive power supply pin that powers up the sensor module.
GND	Reference potential pin.
AO	Analog output pin. It generates a signal proportional to the concentration of gas vapors coming in contact with the sensor.
DO	Digital Output pin. It also produces a digital signal whose limit can be set using the in-built potentiometer.

ARDUINO CODE:

```
/* Air Pollution Monitoring System using Arduino and MQ135 Air Quality Sensor*/

// Include library for LCD and define pins
#include <LiquidCrystal.h>
```

```
const int rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
// Define pins and variable for input sensor and output led and buzzer
const int mq135_aqi_sensor = A0;
const int green_led = 8;
const int blue_led = 9;
const int red_led = 10;
const int buzzer = 11;

int aqi_ppm = 0;        // Set threshold for AQI
void setup() {
// Set direction of input-output pins
pinMode (mq135_aqi_sensor, INPUT);
pinMode (green_led, OUTPUT);
pinMode (blue_led, OUTPUT);
pinMode (red_led, OUTPUT);
pinMode (buzzer, OUTPUT);
digitalWrite(green_led, LOW);
digitalWrite(blue_led, LOW);
digitalWrite(red_led, LOW);
digitalWrite(buzzer, LOW);

Serial.begin (9600);  // Initiate serial and lcd communication
lcd.clear();
lcd.begin (16, 2);
Serial.println("AQI Alert System");
lcd.setCursor(0, 0);
lcd.print("AQI Alert System");
delay(1000);
}
void loop() {
aqi_ppm = analogRead(mq135_aqi_sensor);
Serial.print("Air Quality: ");
Serial.println(aqi_ppm);
lcd.setCursor(0, 0);
lcd.print("Air Quality: ");
lcd.print(aqi_ppm);
if ((aqi_ppm>= 0) && (aqi_ppm<= 50))
{
lcd.setCursor(0, 1);
lcd.print("AQI Good");
Serial.println("AQI Good");
digitalWrite(green_led, HIGH);
digitalWrite(blue_led, LOW);
digitalWrite(red_led, LOW);
digitalWrite(buzzer, LOW);
}
```

```
elseif ((aqi_ppm>= 51) && (aqi_ppm<= 100))
{
lcd.setCursor(0, 1);
lcd.print("AQI Moderate");
Serial.println("AQI Moderate");
tone(green_led, 1000, 200);
digitalWrite(blue_led, HIGH);
digitalWrite(red_led, LOW);
digitalWrite(buzzer, LOW);
}
elseif ((aqi_ppm>= 101) && (aqi_ppm<= 200))
{
lcd.setCursor(0, 1);
lcd.print("AQI Unhealthy");
Serial.println("AQI Unhealthy");
digitalWrite(green_led, LOW);
digitalWrite(blue_led, HIGH);
digitalWrite(red_led, LOW);
digitalWrite(buzzer, LOW);
}
elseif ((aqi_ppm>= 201) && (aqi_ppm<= 300))
{
lcd.setCursor(0, 1);
lcd.print("AQI V. Unhealthy");
Serial.println("AQI V. Unhealthy");
digitalWrite(green_led, LOW);
tone(blue_led, 1000, 200);
digitalWrite(red_led, HIGH);
digitalWrite(buzzer, LOW);
}
elseif (aqi_ppm>= 301)
{
lcd.setCursor(0, 1);
lcd.print("AQI Hazardous");
Serial.println("AQI Hazardous");
digitalWrite(green_led, LOW);
digitalWrite(blue_led, LOW);
digitalWrite(red_led, HIGH);
digitalWrite(buzzer, HIGH);
}
delay (700);
}
```

CODE TO NOTE:

void setup() In this function, the Buzzer is initialized, and set LED as an output device, and MQ-135 as an input device to Arduino Uno.

lcd.begin(); In this function initialize 16×2 LCD, before initializing the 16×2 LCD you must clear using **lcd.clear();** function. Also, set the baud rate at 9600-speed rate using **Serial.begin (9600);**function.

lcd.setCursor(0, 0); After Initializing, the cursor in 16×2 LCD is set to home position.

Serial.println("AQI Alert System"); It will print the messageon the serial terminal/monitor.

lcd.print("AQI Alert System"); It will print the message on the 16×2 LCD.

void loop() In this function, the variable is initialized to catch up analog value, using **aqi_ppm = analogRead(mq135_aqi_sensor);**function. And print on message serial terminal and 16×2 LCD.

if((aqi_ppm>= 0) && (aqi_ppm<= 50))
If the Suppose Analog value is between 0 to 50, the green LED will blink. Print the messageon serial terminal and on LCD will print a message in the second row's first position.

Similarly, depending on AQI value, LED will blink and print on the message serial terminal and the 16×2 LCD following the table.

AQI value	Green LED	Blue LED	Red LED	Buzzer	Serial Terminal and16×2 LCD Message
0 – 50	ON	OFF	OFF	OFF	AQI Good
51 – 100	BLINK	ON	OFF	OFF	AQI Moderate
101 –200	OFF	ON	OFF	OFF	AQI Unhealthy
201 –300	OFF	BLINK	ON	OFF	AQI Very Unhealthy
> 301	OFF	OFF	ON	ON	AQI Hazardous

CONCLUSION:

After this experiment, you can develop your own Air Monitoring and Alert System using the MQ-135 sensor and Arduino Nano/Uno board.

EXP. NO. 8
AIM OF THE EXPERIMENT:

Interfacing Bluetooth module (e.g. HC05)- receiving data from mobile phone on Arduino and display on LCD

INTRODUCTION:

In this experiment, we will explain about interfacing HC-05 Bluetooth Module with Arduino Uno. HC-05 uses bluetooth classic and can be configured in master or slave modes. It can be interfaced with a microcontroller using UART. In other words, it uses serial communication to transmit and receive data serially over standard Bluetooth radio frequency. Therefore, we will use TX, RX pins of Arduino to connect with the module.

In this experiment, we will use an android application to communicate with the Arduino through the HC-05 Bluetooth module. We will see an example to control an LCD from the Android application using Bluetooth wireless communication. The LCD will be connected with a GPIO pin of the Arduino and in effect, we will be controlling that pin.

COMPONENTS REQUIRED

- Arduino Uno
- HC-05 Bluetooth Module
- LCD
- Jumper Wires
- Bread Board

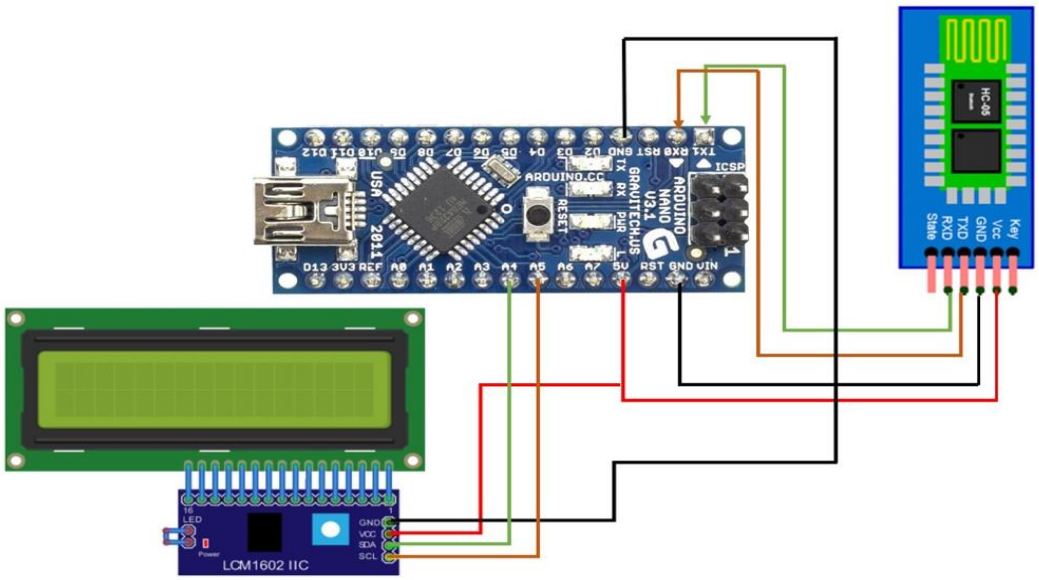
SOFTWARE REQUIRED

- Arduino IDE
- Android App – Arduino Bluetooth Controller

PROCEDURE:

Here, we will transmit data from Smartphone via Bluetooth to the Arduino Uno and display it on LCD.Download and install a **Bluetooth terminal** application on your phone and use it to connect to the HC-05 Bluetooth module.Data is sent from the Smartphone using the **Bluetooth terminal** application.

Follow the circuit diagram and connect the components on the Arduino board as shown in the image given below.



HC-05 Bluetooth Module

- Operating Voltage : 4 V to 6V (have internal 3.3V regulator).
- Operating Current : 30mA
- Integrated antenna and an edge connector.
- Range about 10 meters.
- Configurable in both master and slave modes.
- Pins : STATE, RXD, TXD, GND, VCC, KEY/ENABLE

Pin Description

- **EN:** It is the enable pin. When this pin is floating or connected to 3.3V, the module is enabled. If this pin is connected to GND, the module is disabled.
- **Vcc:** This is the supply pin for connecting +5V. As the Module has on-board 3.3V regulator, you can provide +5V supply.
- **GND:** It is the ground pin.
- **TX:** It is the Transmitter pin of the UART Communication.
- **RX:** It is the Receive Pin of UART.
- **STATE:** This is a status indicator pin. This pin goes LOW when the module is not connected to any device. When the module is paired with any device, this pin goes HIGH.

Modes of Operation

- The HC-05 Bluetooth Module can be configured in two modes of operation: Command Mode and Data Mode.
- In Command Mode, you can communicate with the Bluetooth module through AT Commands for configuring various settings and parameters of the Module like get the firmware information, change UART Baud Rate, change module name, set it as either Master or slave etc.
- An important point about HC-05 Module is that it can be configured as Master or Slave in a communication pair. In order to select either of the modes, you need to activate the Command Mode and sent appropriate AT Commands.
- Coming to the Data Mode, in this mode, the module is used for communicating with other Bluetooth device i.e. data transfer happens in this mode.

ARDUINO CODE:

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd (0x3F, 16, 2);
String msg;
void setup()
        {
// put your setup code here
        lcd.init();
        lcd.backlight ();
        Serial.begin(9600);
}

void loop()
{
// put your main code here
        readSerialPort ();
        if (msg.length() >0)
        {
                lcd.clear();
                lcd.print (msg);
        }
msg = "";
}
void readSerialPort()
        {
        while (Serial.available())
                {
                delay(10);
                char c = Serial.read();
                msg+= c;
                }
        }
```

CODE TO NOTE:

- **lcd.init();**
  It initializes the interface to the LCD.

- **lcd.backlight ();**
  The function 'lcd. backlight();' if this is put in the setup, it turns the backlight on.

- **Serial. available()**
  It returns the number of characters (i.e. bytes of data) which have arrived in the serial buffer and that are ready to be read.

- **readSerialPort()**
  It reads the incoming serial data in the Arduino. The int data type is used here. It returns the first data byte of the arriving serial data. It also returns -1/0 when no data is available on the serial port. The syntax used in the Arduino programming is Serial.

- **Serial.read();**
  It is a function of the Arduino Serial Library and what it does is read out the first available byte from the serial receive buffer. When it reads it out, it removes that byte from the buffer.

CONCLUSION:

We have successfully studied that the Bluetooth module is a small range wireless communication device that can be used for various purposes like sending of data or controlling of devices connected with it. Furthermore, it can be interfaced with Arduino Uno and can be used in multiple projects where wireless communication is required for a small range. In this experiment, we have interfaced the Bluetooth module with Arduino Uno and controlling the LCD by sending data through the Bluetooth module.

EXP. NO. 09

NAME OF THE EXPERIMENT: InterfacingRelay module to demonstrate Bluetooth based home automation application. (using Bluetooth and relay).

INTRODUCTION

In the modern day's everyone uses Smartphone and the internet. Therefore, every Smartphone has Bluetooth System. In this experiment, we will design a simple Arduino Bluetooth Control Home Automation using the HC-05 Bluetooth module, which is used to switch ON or OFF different electrical appliances remotely. Home Automation systems can make our life easy and secure. Here we will control 4 different home appliances using Smartphone App through Bluetooth communication.

REQUIRED MATERIAL:

In order to get started, you'll need the following components:

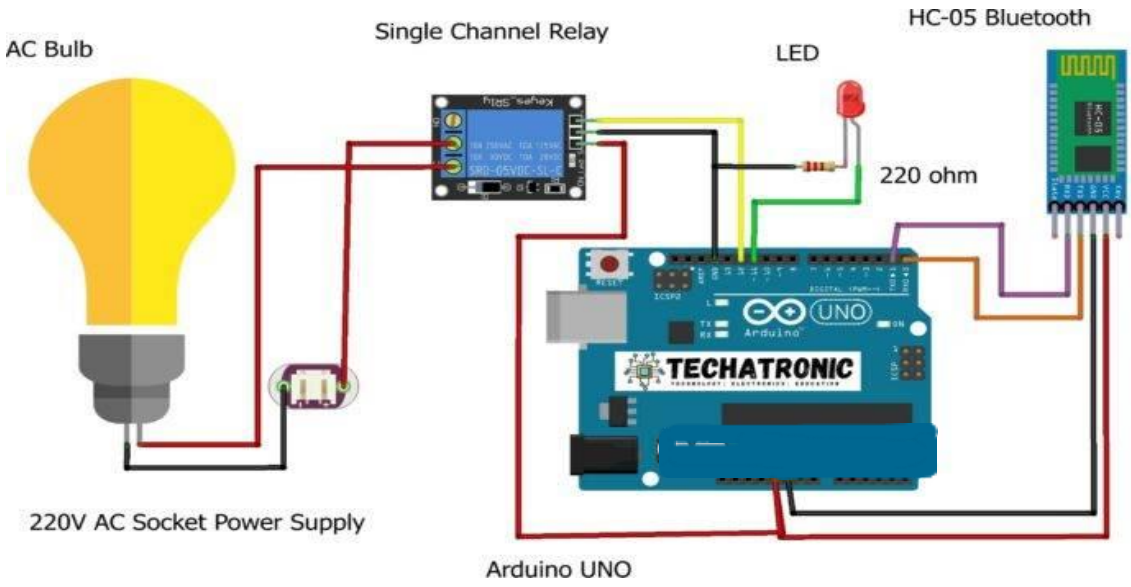
- 1x Arduino UNO
- 1x Single Channel Relay Module
- 1x HC 05 Wireless Bluetooth Module
- 1x AC Bulb Holder
- 1x AC Bulb
- 1x 2.2k ohm resistor
- 1x 1k ohm resistor
- 1x Breadboard
- Jumper wires

SOFTWARE:

- Arduino Bluetooth Controller (Android)
- Arduino IDE

PROCEDURE:

Follow the circuit diagram and hook up the components on the Arduino board as shown in the image given below.



Connection Table (Bluetooth HC-05 to Arduino)

Arduino UNO		**HC-05 Bluetooth**
(+5V) VCC		VCC
GND (Ground)		GND (Ground)
TX Pin		RX Pin
RX Pin		TX Pin
Arduino UNO		**Relay Module**
(+5V) VCC		VCC
GND (Ground)		GND
12 Pin		OUT Pin
Arduino UNO	**LED**	**220-ohm Resistor**
11 Pin	Anode Pin	
GND (Ground)		Terminal 1
	Cathode Pin	Terminal 2
220 Volt AC Supply	**AC Bulb**	**Relay Module**
		Normally Open
Phase		Common
	Terminal 1	Normally Closed
Neutral	Terminal 2	

A **Single Channel relay** is an automatic switch that is commonly used in an automatic control circuit and to control a high-current using a low-current signal. The input voltage of the relay signal ranges from 0 to 5V.
Common (COM): This pin is connected to the main terminal of the Load to make it active.

Normally Closed (NC): This second terminal of the load is connected to either NC/ NO pins. If this pin is connectedto the load then it will be ON before the switch.

Normally Open (NO): If the second terminal of the load is allied to the NO pin, then the load will be turned off before the switch.

VCC: This pin needs 5V DC to work. So 5V DC power supply is provided to this pin.

Ground(GND): This pin connects the GND terminal of the power supply.

Signal Pin: The signal pin is mainly used for controlling the relay.

ARDUINO CODE:

```
#define RELAY_ON 0
#define RELAY_OFF 1
#define RELAY_1  4
char data = 0;
void setup()
{
// Set pin as output.
pinMode(RELAY_1, OUTPUT);
```

```
// Initialize RELAY1 = off. So that on reset it would be off by default
digitalWrite(RELAY_1, RELAY_OFF);
Serial.begin(9600);
Serial.print("Type: 1 to turn on the bulb. 0 to turn it off!");
}
void loop()
{
 if (Serial.available() > 0)
{
data = Serial.read();     //Read the incoming data and store it into variable data
  Serial.print(data);      //Print Value inside data in Serial monitor
Serial.print("\n");       //New line
if(data == '1')
{
digitalWrite(RELAY_1, RELAY_ON);
Serial.println("Bulb is now turned ON.");
}
else if(data == '0'){
digitalWrite(RELAY_1, RELAY_OFF);
 Serial.println("Bulb is now turned OFF.");
}
}
}
```

CODE TO NOTE:

First, we initialize the relay first in setup() method. Then, wait for input on the Serial port in the loop() method.
If '1'is received input, we turn on the relay.
If '0' is received, we turn off the relay.

Now, when you'll try to upload the code to your Arduino, while the HC-04 module is connected, you'll get the following error:
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x00
An error occurred while uploading the sketch
That's because Arduino UNO operates on UART, which means it has common TX-RX lines. It can't communicate with your computer and Bluetooth module at the same time.

To solve the error,
Simply unplug the jumper wire connected to Pin 0 of Arduino (Rx pin), and re-attempt to upload code on Arduino. You should now be able to update the code successfully. The reason the simple hack works is that Arduino is no longer receiving data from two sources, and therefore can receive the code from the computer.

ARDUINO BLUETOOTH CONTROLLER: CONNECTING TO A SMARTPHONE

Now that we have configured the hardware and successfully uploaded the code, our next step is to control the setup from a smartphone. In order to that, you'll need to download the Arduino Bluetooth Controller app on your Android device.
Here's how you can configure your Android device to send commands to Arduino:
Step-1: Open the app on your smartphone. It will ask for Bluetooth permissions. Click 'Ok'.
Step-2: Next, it will list all the available devices in your vicinity. Click on HC-05.
Step-3: Once you click on the device, you would be connected to the transceiver. It would now prompt you to enter the mode that you wish to use. Select "Switch" mode.
Step-4: You should be redirected to the following screen. Click on the "Settings" tab in the top-right corner of the screen.
Step-5: It will ask you to set values for ON and OFF. Enter '1' in the ON text box and '0' in the OFFtext box. Click Submit.

CONCLUSION:

Using the above setup, you can turn any electrical appliance into a smart device that can be controlled from your smartphone. While we used a single relay in the example, you can expand the use-case with an optocoupler 8-channel relay.