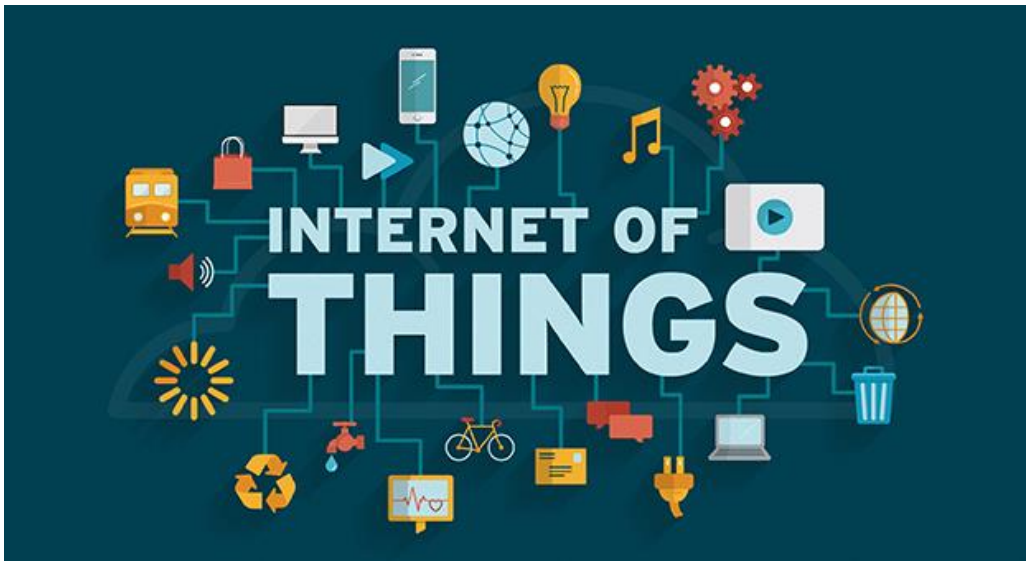
LABORATORY MANUAL
ON
INTERNET OF THINGS LAB



SIXTH SEMESTER

DEPARTMENT OF INFORMATION TECHNOLOGY

GOVERNMENT POLYTECHNIC, BHUBANESWAR
Government of Odisha
ସରକାରୀ ବହୁବୃତ୍ତି ଅନୁଷ୍ଠାନ, ଭୁବନେଶ୍ୱର

VISION AND MISSION OF THE INSTITUTE

VISION

The Vision of the Government Polytechnic, Bhubaneswar is to be a premier Diploma Institution in the Eastern India by transforming the learners into competent technocrats by imparting technical education in multiple disciplines.

MISSION

The mission of the Government Polytechnic, Bhubaneswar is to:

- Focus on quality teaching and learning by providing better academic environment for both faculties and students.
- Develop professionally and technically skilled students who work for betterment of the society
- Nurture positive attitude and social values amongst the students and staff.

VISION AND MISSION OF THE DEPARTMENT

DEPARTMENT OF INFORMATION TECHNOLOGY

VISION

The Vision of the Department of Information Technology is to develop knowledge and skill of students by quality education to make responsible and competent technocrats who will work for the needs of industries and betterment of the society.

MISSION

The Mission of the Department of Information Technology is to:

M-1. Provide an excellent learning environment for the students to be technically efficient in solving problems.

M-2. Prepare the students with strong fundamental concepts, analytical capability and programming skills.

M-3. Create a nurturing environment for lifelong learning.

M-4. Generate social awareness and responsibility within the students to serve the mankind and protect the environment.

Program Educational Objectives (PEOs)

Program Educational Objective of Diploma in Information Technology:

The Program Educational Objectives (PEOs) of the Department of Information Technology are:

PEO-1: ***Strong Fundamentals & Technically skilled:*** *To equip the students with sound knowledge in mathematics, science and basic engineering fundamentals and develop skill to solve engineering problems.*

PEO-2: ***Higher study and Professional responsibilities:*** *To enable students to pursue higher studies for professional growth and/or work as an entrepreneur for successful professional career.*

PEO-3: ***Teamwork and lifelong learning:*** *To inculcate soft skill, professional and ethical attitude, teamwork, leadership skills and multidisciplinary approach within the students which will enable them to correlate societal issue with engineering solution and pursue lifelong learning.*

Program Outcomes and Program Specific Outcomes (PSOs)

Program Outcomes for an Engineering diploma graduate:

i) **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems

ii) **Problem analysis:** Identify and analyze well-defined engineering problems using codified standard methods.

iii) **Design/ development of solutions:** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

iv) **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

v) **Engineering practices for society, sustainability and environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices.

vi) **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

vii) **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

Program Specific Outcome of Diploma in Information Technology:

PSO-1: Ability to design and develop methodologies in computer programming and apply those skills in the field of Information Technology.

PSO-2: Develop potentiality to apply the skills learnt for betterment of the society by updating it on the basis of the advanced technological changes and effectively communicate about well-defined engineering activities.

Pr-2 IOT LAB

Total Period:	60	**Maximum marks:**	100 Marks
Lab periods:	4P / week	**Term Work/ Sessional:**	50 Marks
Examination:	3 Hours	**Term End Examination:**	50 Marks

CONTENTS

1. Basics of C language using Arduino IDE
    - Understanding basics of Arduino IDE
    - Variables, datatype, loops, control statement, function
2. Practical using Arduino-interfacing sensors
    - Interfacing Light Emitting Diode(LED)- Blinking LED
    - Interfacing Button and LED – LED blinking when button is pressed
    - Interfacing Light Dependent Resistor (LDR) and LED, displaying automatic night lamp
    - Interfacing Temperature Sensor(LM35) and/or humidity sensor (e.g.DHT11)
    - Interfacing Liquid Crystal Display(LCD) – display data generated by sensor on LCD
    - Interfacing Air Quality Sensor-pollution (e.g. MQ135) – display data on LCD , switch on LED when data sensed is higher than specified value.
    - Interfacing Bluetooth module (e.g. HC05)- receiving data from mobile phone on Arduino and display on LCD
    - Interfacing Relay module to demonstrate Bluetooth based home automation application. (using Bluetooth and relay).

Books Recommended:

Sl.No	Name of Authors	Title of the Book	Name of Publisher
1	Vijay Madisetti, Arshdeep Bahga,	Internet of Things: A Hands-On Approach	University Press
2	Yashavant Kanetkar, Shrirang Korde,	Internet Of Things (IOT) experiments	
3	Neerparaj Rai	Arduino Projects For Engineers	

COURSE OUTCOMES:

After completion of this course the student will be able to

1. Understating basics of Arduino IDE
2. write programs for Arduino
3. interface various sensory devices with Arduino
4. implement the IoT technologies in practical domains of society

COs, POs and PSOs Mapping

Cos, Pos and PSOs mapping	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PSO1	PSO2
CO1	2	1	1	1	-	1	1	2	1
CO2	2	1	1	1	-	1	1	2	1
CO3	2	1	1	1	-	1	1	2	1
CO4	2	1	1	1	-	1	1	2	1
Total	6	3	3	3	-	4	4	8	4
Average	1.5	0.75	0.75	0.75	-	1	1	2	1

EXPERIMENT- 1.1

AIM OF THE EXPERIMENT: Understanding Basics Of Arduino Ide

INTRODUCTION:

The Arduino IDE (Integrated Development Environment) is a popular software tool used in the field of Internet of Things (IoT) to develop and program microcontroller-based systems using the C programming language. Arduino is an open-source platform that provides a range of microcontroller boards and a software development environment that allows programmers to write, compile, and upload code to these boards for various applications in the field of IoT.

In this lab manual, we will cover the basics of C language using Arduino IDE. C is a widely used programming language known for its efficiency and versatility, making it suitable for developing IoT applications that require low-level control over hardware. We will explore the fundamental concepts of C programming and their application in the context of IoT using Arduino IDE.

LAB SETUP:

To get started, you will need the following equipment and software:

- Arduino board: You can choose any Arduino board, such as Arduino Uno, Arduino Mega, or Arduino Nano, depending on your requirements.
- USB cable: To connect the Arduino board to your computer for programming.
- Arduino IDE: You can download the Arduino IDE from the official Arduino website (https://www.arduino.cc/en/software) and install it on your computer.
- LEDs, resistors, breadboard, and other electronic components: These will be used in the lab exercises to build basic IoT circuits.

LAB EXERCISES:

The lab exercises are designed to help you understand the basics of C programming using Arduino IDE. Each exercise builds upon the previous one and introduces new concepts gradually. Here is an overview of the lab exercises:

Exercise 1: Setting up Arduino IDE

- Learn how to install Arduino IDE on your computer.
- Understand the Arduino IDE interface, including the code editor, toolbar, and serial monitor.
- Write your first C program to blink an LED using Arduino IDE.

Exercise 2: Basic Data Types and Operators

- Learn about basic data types in C, such as int, float, and char.
- Understand arithmetic, relational, and logical operators in C.
- Write C programs to perform basic mathematical operations and control LED brightness using Arduino IDE.

Exercise 3: Control Statements

- Understand conditional statements, such as if, else, and switch, in C.
- Learn about looping statements, such as for, while, and do-while, in C.
- Write C programs to control LEDs based on sensor inputs and implement IoT applications, such as temperature monitoring and smart lighting using Arduino IDE.

Exercise 4: Functions and Libraries

- Understand the concept of functions and how to define and call functions in C.
- Learn about pre-defined functions in C and how to create your own functions.
- Write C programs to modularize code, create custom libraries, and implement IoT applications, such as motion sensing and remote control using Arduino IDE.

Exercise 5: Arrays and Pointers

- Learn about arrays and pointers in C and their application in IoT.
- Understand how to declare, initialize, and access arrays in C.
- Explore the concept of pointers and their use in C programs.
- Write C programs to store and manipulate sensor data, implement IoT applications, such as data logging and remote data retrieval using Arduino IDE.

CONCLUSION:

In this lab manual, we have covered the basics of C language using Arduino IDE, which is a powerful tool for developing IoT applications. By understanding the fundamental concepts of C programming and their application in the context of IoT, you can create a wide range of projects that involve microcontrollers, sensors, actuators, and other electronic components. With the knowledge gained from these lab exercises, you can further explore advanced topics in IoT and build more complex and sophisticated applications. Happy programming!

EXPERIMENT- 1.2

AIM OF THE EXPERIMENT: Variables, Datatype, Loops, Control Statement, Function

1. **Variables and Data Types in Arduino C:**

Variables are used to store and manipulate data in a program. In Arduino C, variables are declared using specific data types, which determine the size and type of data that can be stored. Common data types in Arduino C include:

i. int: Represents integer values, which are whole numbers without decimal points. It typically uses 2 bytes of memory and can store values ranging from -32,768 to 32,767.
ii. float: Represents floating-point values, which are numbers with decimal points. It typically uses 4 bytes of memory and can store values with a range of approximately ±3.4028235E+38 to ±1.1754944E-38.
iii. char: Represents characters, which are single alphanumeric characters. It typically uses 1 byte of memory and can store values ranging from -128 to 127, representing ASCII codes.
iv. bool: Represents boolean values, which are either true or false. It typically uses 1 byte of memory.

Variables are declared using the syntax: '**data_type variable_name;'** . For example, 'int count;' declares an integer variable named "count". Once a variable is declared, it can be assigned a value using the assignment operator (=), and its value can be accessed and manipulated in the program.

2. **Loops and Control Statements in Arduino C:**

Loops and control statements are used to control the flow of execution in a program. They allow you to repeat certain blocks of code or execute different parts of the code based on certain conditions. Common loops and control statements in Arduino C include:

i. for loop: Executes a block of code a specific number of times. It consists of an initialization statement, a condition statement, and an increment or decrement statement.
For example:

```
for (int i = 0; i < 10; i++)
  {
    // Code to be executed
  }
```

ii. while loop: Executes a block of code as long as a certain condition is true.
For example:

```
int i = 0;
while (i < 10) {
 // Code to be executed
 i++;
}
```

iii. if statement: Executes a block of code if a certain condition is true.
For example:

```
int value = 5;
if (value > 0) {
  // Code to be executed
}
```

iv. else statement: Executes a block of code if a certain condition is false. It can be used in conjunction with an if statement.
For example:

```
int value = 5;
if (value > 0) {
  // Code to be executed if condition is true
} else {
  // Code to be executed if condition is false
}
```

v. switch statement: Executes a block of code based on the value of a variable. It allows you to define multiple cases and specify code to be executed for each case.
For example:

```
int choice = 2;
switch (choice) {
  case 1:
    // Code to be executed for case 1
    break;
  case 2:
    // Code to be executed for case 2
    break;
  default:
    // Code to be executed for all other cases
    break;
}
```

3. Functions in Arduino C:

Functions are blocks of code that can be defined and called by name. They allow you to encapsulate a set of instructions into a single unit that can be reused in different parts of a program. Functions in Arduino C consist of a function declaration, function definition, and function call.

- Function Declaration: Specifies the return type, function name, and any parameters that the function accepts.

  For example:

```
int add(int a, int b); // Function declaration
```

EXPERIMENT- 2.1

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.

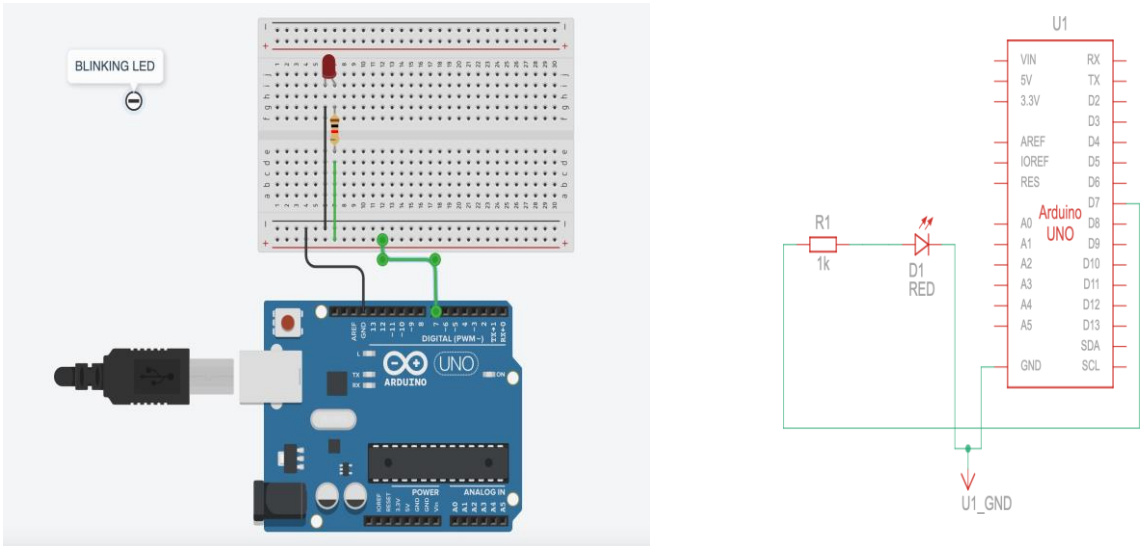
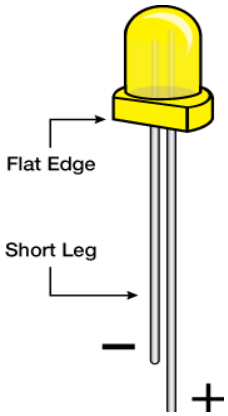
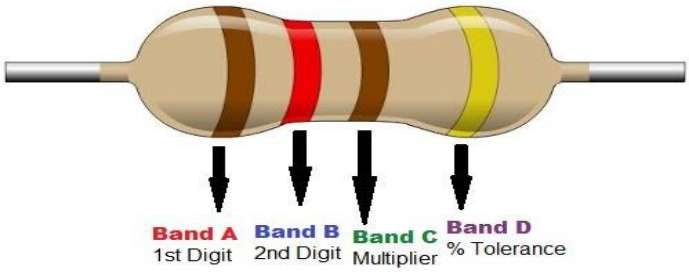


Fig. : Arduino circuit diagram for Blinking LED when OFF

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



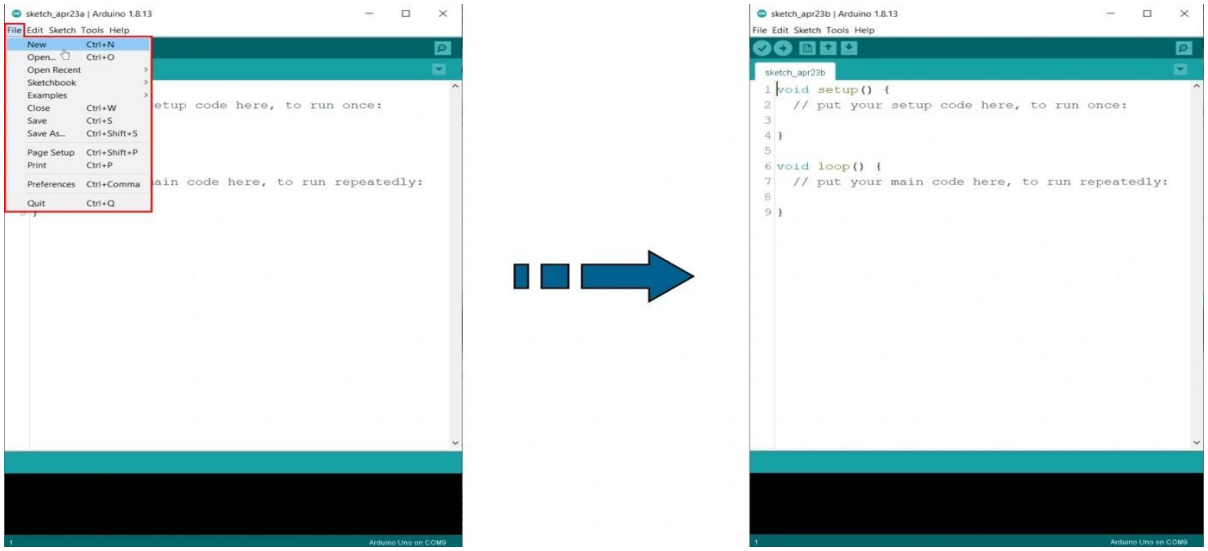
Components like resistor helps to limit or regulate the flow of electrical current in an electronic circuit.



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:

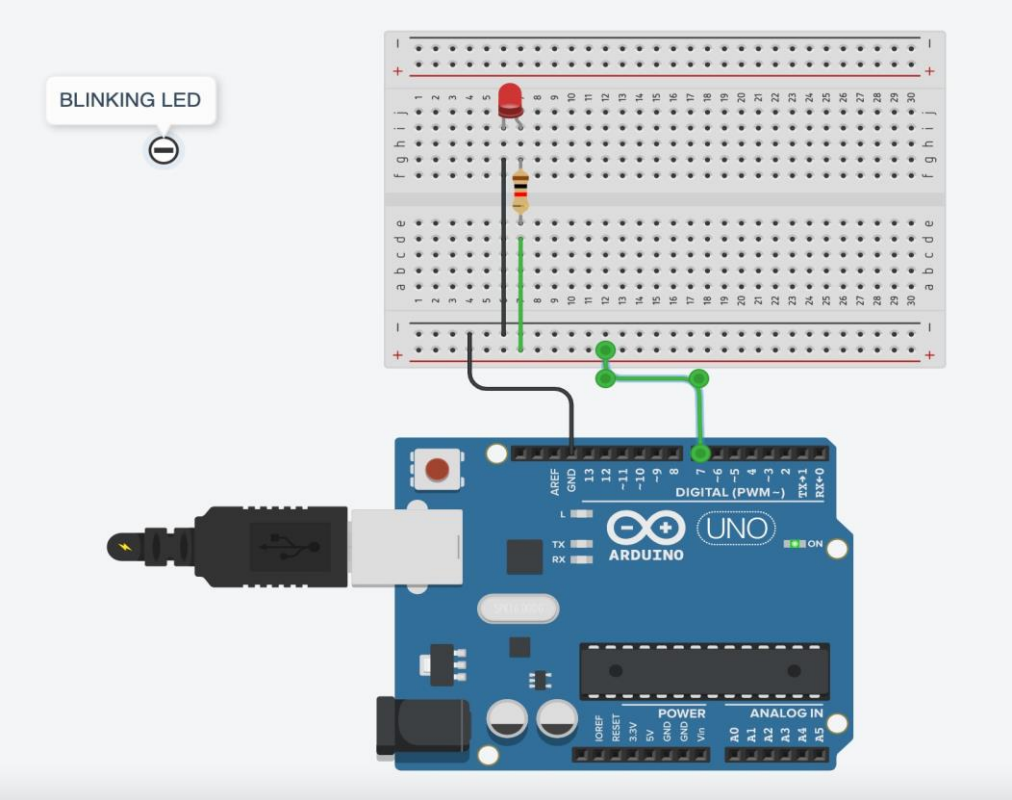
```
/*

    This program blinks pin 13 of the Arduino (the built-in LED)
*/

void setup()
{
    pinMode(13, OUTPUT);
}


void loop()
{
    // 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 the 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.

EXPERIMENT- 2.2

AIM OF THE EXPERIMENT: Interfacing Button and LED – LED blinking when push button 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.

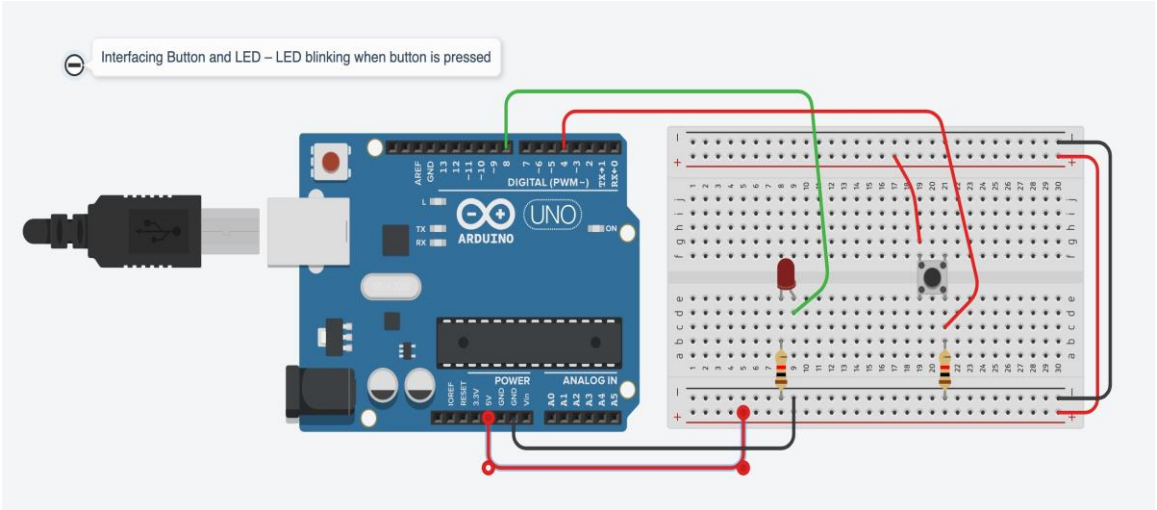
COMPONENTS REQUIRED:

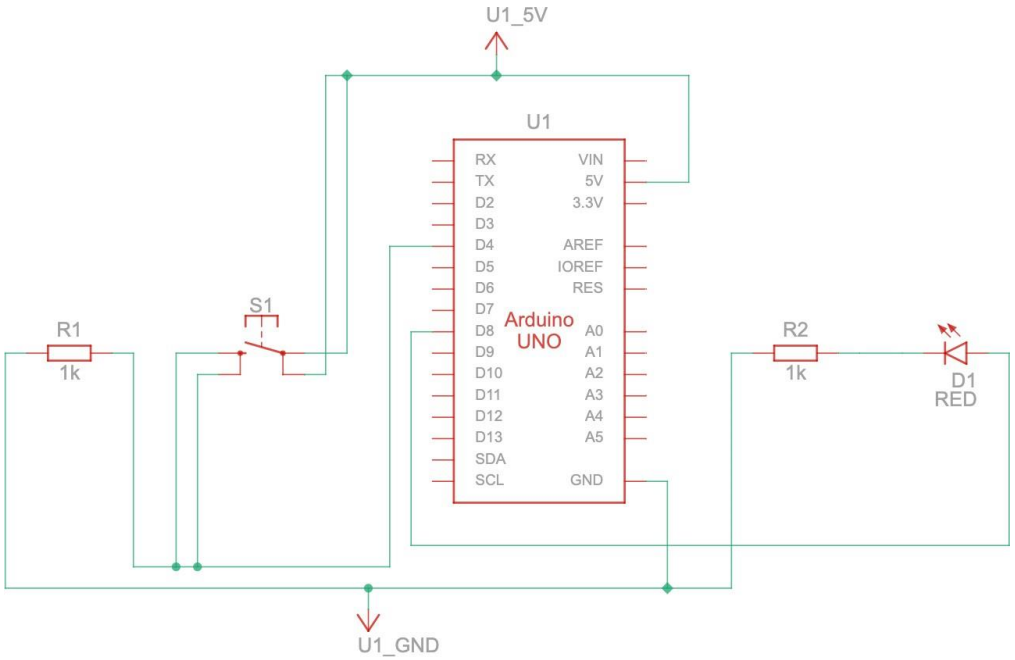
We need the following components –

- **1x** Breadboard
- **1x** Arduino Uno R3
- **1x** LED
- **7x** Jumper Wires
- **1x** Push Buttons
- **2x** 1k Ω 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's 0V present at the pin.
- if (digitalRead(4) == HIGH)
  It will 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 board or see the troubleshooting section.

EXPERIMENT- 2.3

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.

A relay is a programmable electrical switch which can be controlled by Arduino or any micro controller board. It is used to programmatically control on and off the devices, which use the high voltage and/or high current. It is a bridge between I do you know and high-voltage devices.

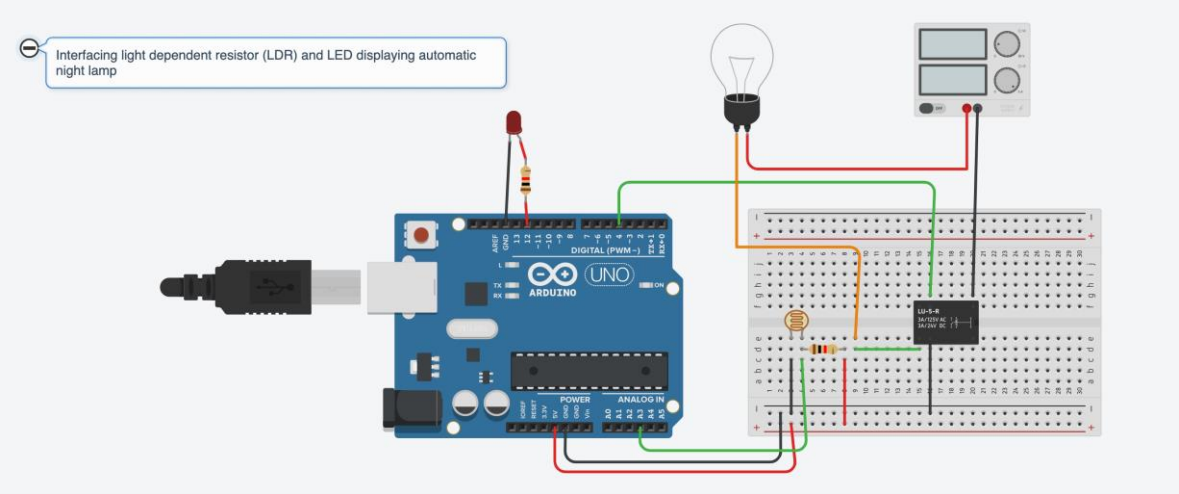
COMPONENTS REQUIRED:

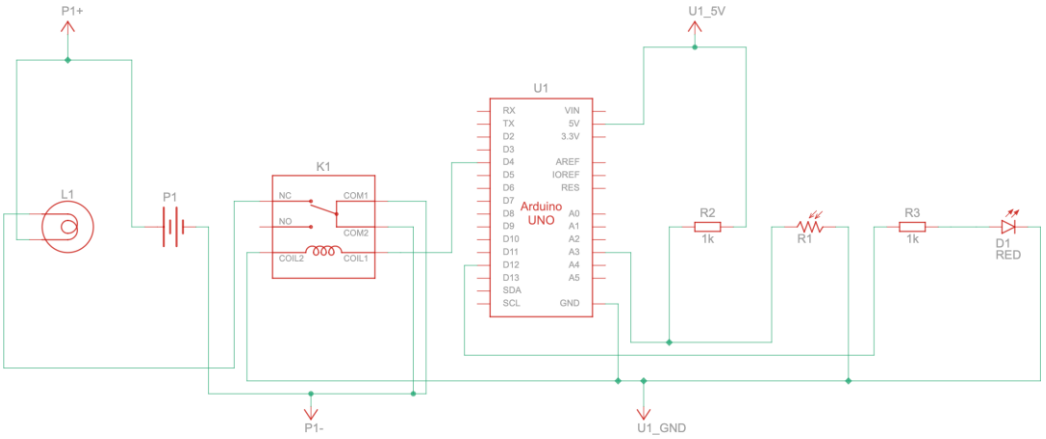
We need the following components –

- **1X** Arduino UNO R3
- **1X** LDR light sensor
- **1X** 5V Relay Module
- **2X** Resistor (100k)
- **1X** AC Bulb (25 or 100) Watt
- **1X** AC Power Supply

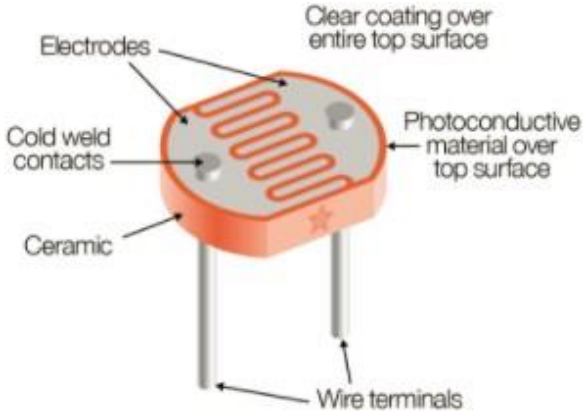
PROCEDURE:

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

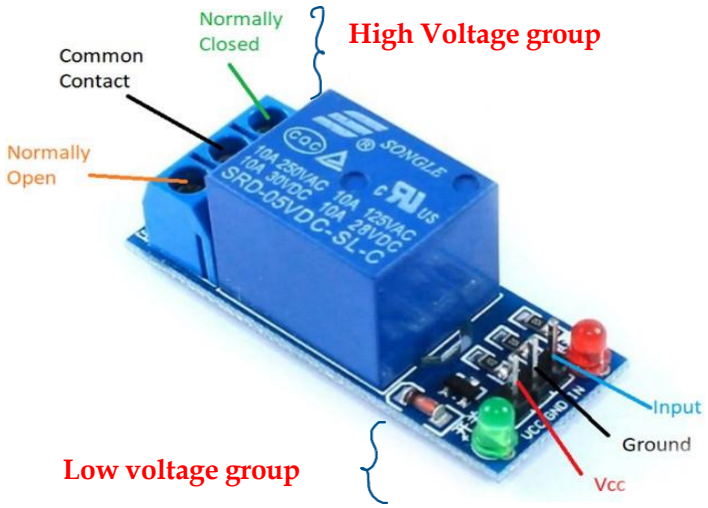




Please note: Photoresistor has two pins. Since it is a kind of resistor we do not need to distinguish these pins. They are symmetric.



Relay consists of a coil with metal contacts on one side (low-voltage side) that can be magnetized and demagnetized to open or close the circuit attached on the other side (high-voltage side). High voltage side consists of a connector with 3 sockets namely common (COM), normally closed (NC) and normally open (NO). The relay comes in different ratings like 12V, 9V, 5V, and 3V.



ARDUINO CODE:

```
/*

This program uses photoresistor to sense the brightness and accordingly switches ON/OFF the bulb automatically.

*/


const int LED=12;               // LED connect to Digital Pin
const int LDRSensor= A3;        //Sensor pin connects to analog pin A3

void setup()
{
  pinMode (A3, INPUT); pinMode (12,
  OUTPUT);
  pinMode          (4,          OUTPUT);
  Serial.begin(9600);
}


void loop()
{
  Serial.print("Brightness:");
  Serial.println(analogRead(A3));           //sensor reading value stored in A3
  if (analogRead(A3) > 500)                 //if the light is above the threshold value, the LED will
turns on
   {

     digitalWrite(12,               HIGH);
     digitalWrite(4, LOW);
           Serial.print("Night- Switch ON Light\n"); delay(1000);
   }

  else
   {

     digitalWrite(12, LOW);                 //otherwise,  the  LED  is  off
          digitalWrite(4, HIGH);            //otherwise, the BULB is off
                  Serial.print("Day- Switch OFF Light\n"); delay(1000);
   }

}
```

CODE TO NOTE:

- const int LED=12;

  It denotes a digital pin that will supply a trigger output of sufficient voltage.

- const int LDRSensor= A3;

  It is used to denote an analog pin that will receive the analog value from the circuit .

- Serial.println(analogRead(A3));

  We get this analog value from the analog pin A3 and store its value, then print its value in a new line.

- if (analogRead(A3) > 500)

  Arduino senses the value from LDR in the range 0 to 1023. When the LDR value is less than 500 then we call it as day condition and the bulb remains off. When the value is greater than 500 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. The LED 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

EXPERIMENT- 2.4

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.

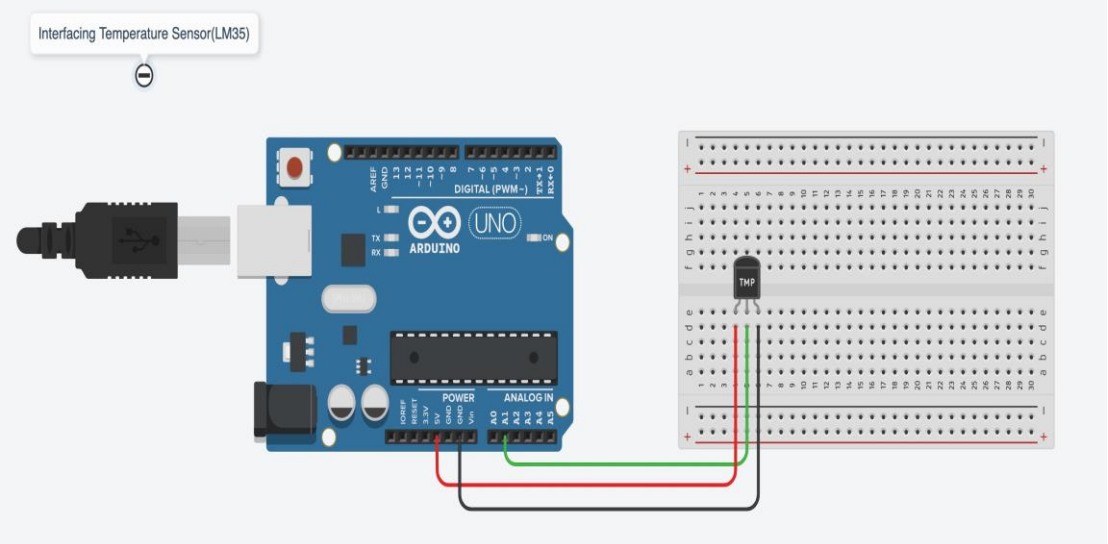
COMPONENTS REQUIRED:

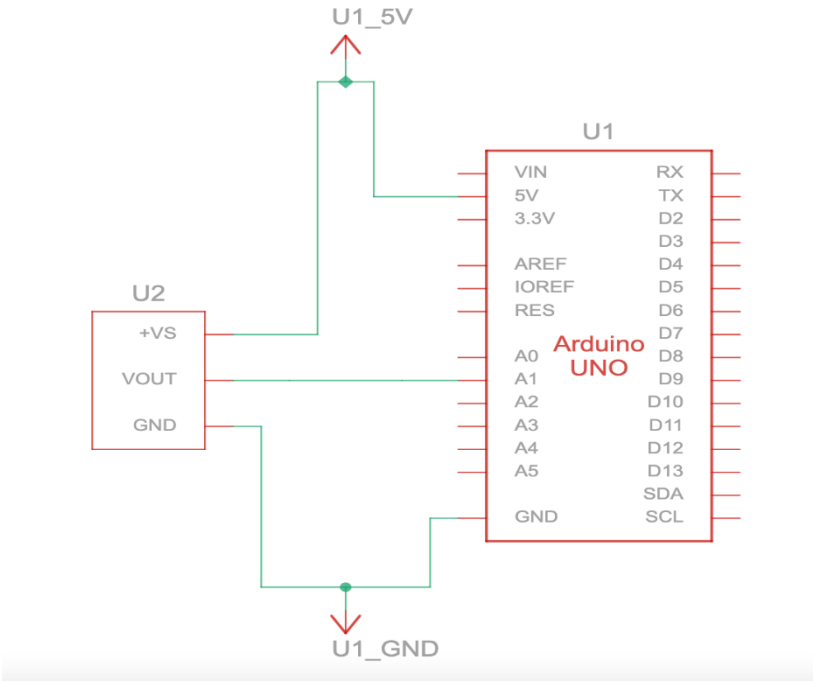
We need the following components –

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

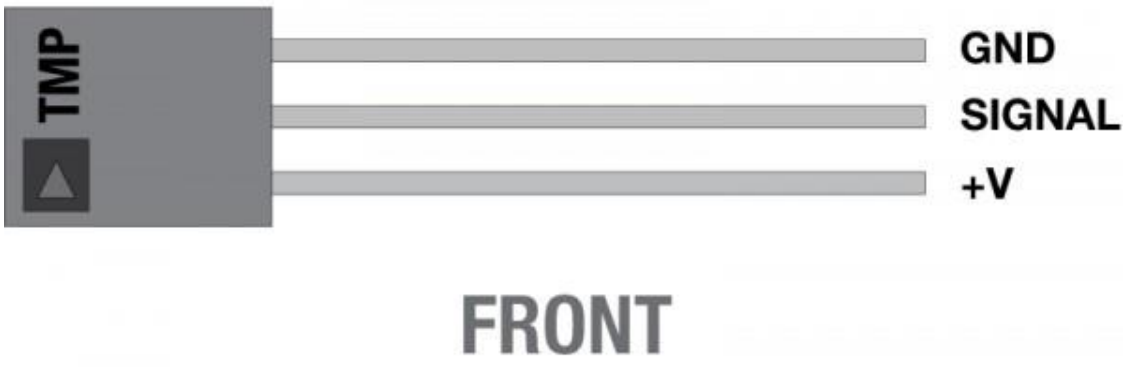
PROCEDURE:

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

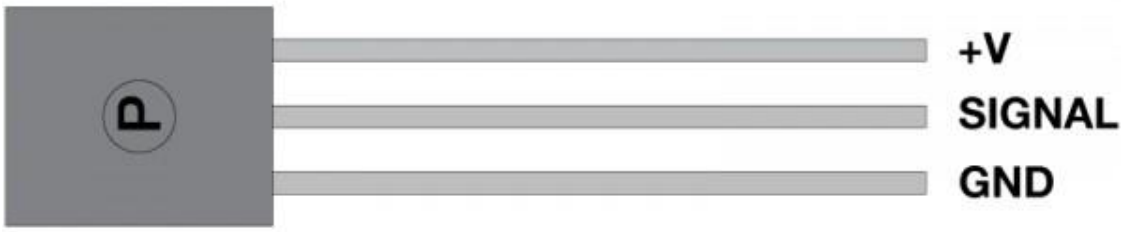




Please note: The temperature sensor can only be connected to a circuit in one direction. See below for the pin outs of the temperature sensor - TMP35



FRONT



BACK

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.

EXPERIMENT- 2.5

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.

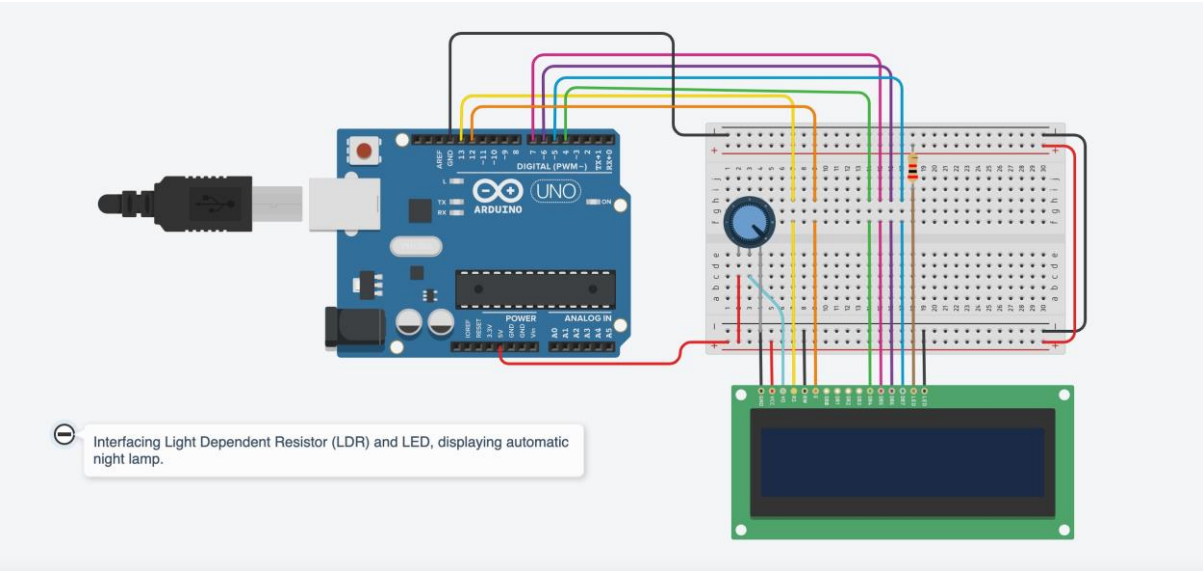
COMPONENTS REQUIRED:

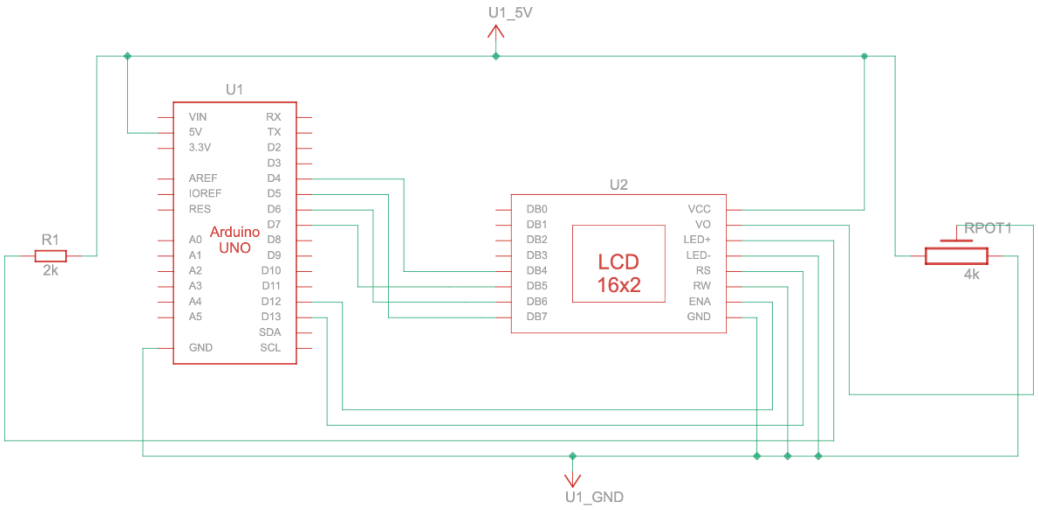
We need the following components –

- **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 counting begins 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 counting begins 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.

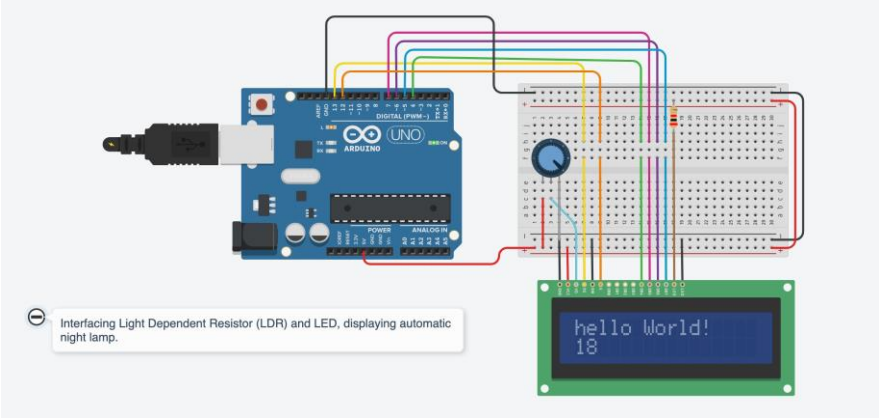


FIG. Output displayed when power supplied to the board

CONCLUSION:

We have successfully studied Interfacing Liquid 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.

EXPERIMENT- 2.6

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, 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.

COMPONENTS REQUIRED:

We need the following components –

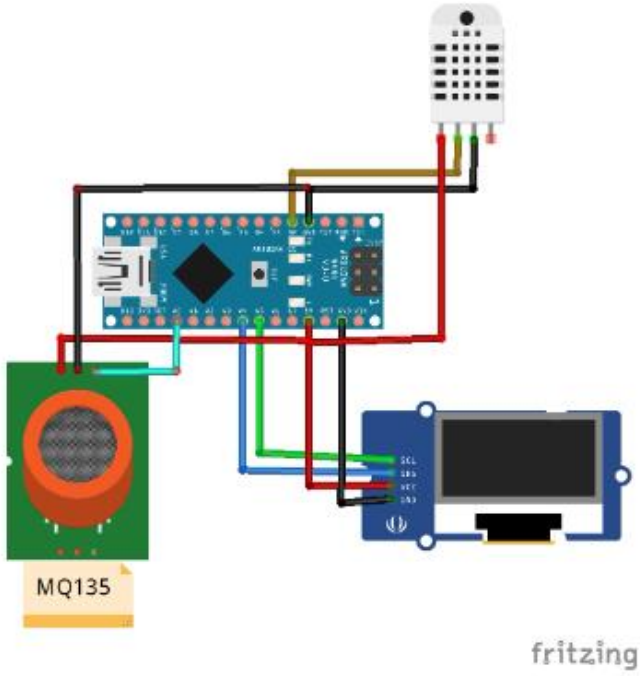
- Arduino Uno board
- MQ135 Air Quality Sensor
- LCD module (16x2 or 20x4)
- Breadboard
- Jumper wires
- Resistor (220 ohms)
- LED

PROCEDURE:

1. Gather all the required components and set up the Arduino board on a breadboard.
2. Connect the MQ135 sensor to the Arduino board. Use the following connections:
    - Connect the VCC pin of the MQ135 sensor to the 5V pin of the Arduino board.
    - Connect the GND pin of the MQ135 sensor to the GND pin of the Arduino board.
    - Connect the OUT pin of the MQ135 sensor to the A0 analog input pin of the Arduino board.
3. Connect the LED to the Arduino board. Use the following connections:
    - Connect the longer leg (anode) of the LED to a digital pin (e.g., 13) of the Arduino board.
    - Connect the shorter leg (cathode) of the LED to a 220-ohm resistor.
    - Connect the other end of the resistor to the GND pin of the Arduino board.
4. Connect the LCD module to the Arduino board. Use the following connections:
    - Connect the VSS pin of the LCD module to the GND pin of the Arduino board.
    - Connect the VDD pin of the LCD module to the 5V pin of the Arduino board.
    - Connect the VO pin of the LCD module to a potentiometer to adjust the contrast.

- Connect the RS, RW, and EN pins of the LCD module to digital pins (e.g., 12, 11, 5) of the Arduino board.
- Connect the D4, D5, D6, and D7 pins of the LCD module to digital pins (e.g., 4, 3, 2, 1) of the Arduino board.

5. Upload the provided Arduino code to the Arduino board using the Arduino IDE.
6. Open the Serial Monitor in the Arduino IDE to view the sensor data.
7. Power up the Arduino board and the sensor.
8. The LCD module will display the "Air Quality Sensor" message on the first line and the sensor data (in percentage) on the second line.
9. If the sensor data exceeds the specified threshold value (e.g., 200), the LED will turn on as an alert for high pollution levels.
10. Adjust the potentiometer on the LCD module to set the contrast for better readability.
11. Observe the sensor data on the Serial Monitor and the LCD module in real-time, and observe the LED alert based on the pollution levels.
12. To turn off the LED alert, lower the pollution levels below the threshold value.
13. To stop the Arduino board, click the "Reset" button in the Arduino IDE or disconnect the power supply.



ARDUINO CODE:

```
// Include required libraries
#include <LiquidCrystal.h>

// Define pin assignments
#define MQ135_PIN A0 // Analog pin for MQ135 sensor
#define LED_PIN 13    // Digital pin for LED
#define THRESHOLD 200 // Threshold value for pollution level

// Initialize LCD module
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // Set up serial communication
  Serial.begin(9600);

  // Set up LCD module
  lcd.begin(16, 2);
  lcd.print("Air Quality");
  lcd.setCursor(0, 1);
  lcd.print("Sensor");

  // Set LED pin as output
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  // Read analog value from MQ135 sensor
  int sensorValue = analogRead(MQ135_PIN);

  // Convert analog value to voltage
  float voltage = sensorValue * (5.0 / 1023.0);

  // Convert voltage to resistance using known equation
  float resistance = ((5.0 * 10.0) / voltage) - 10.0;

  // Calculate air quality index using resistance value
  float airQuality = 100.0 - ((resistance * 100.0) / 10000.0);

  // Display sensor data on LCD module
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Air Quality:");
  lcd.setCursor(0, 1);
  lcd.print(airQuality);
  lcd.print("%");
```

```
  // Activate LED alert if pollution level exceeds threshold
  if (airQuality > THRESHOLD) {
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }

  // Delay for stability
  delay(500);
}
```

CONCLUSION:

In conclusion, this lab provided a hands-on experience in interfacing an MQ135 Air Quality Sensor with an Arduino board and displaying the sensor data on an LCD module. The implementation of an LED alert for high pollution levels demonstrated the use of control statements and functions in Arduino code. This lab serves as a foundation for students to understand the basics of IoT, environmental sensing, and pollution control. Further enhancements can be made to expand the functionality of the system. Overall, this lab emphasized the significance of air quality monitoring in addressing the issue of air pollution and its impact on human health and the environment in the context of IoT applications.

EXPERIMENT- 2.7

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

INTRODUCTION:

In this lab, we will learn how to interface a Bluetooth module (e.g. HC05) with an Arduino board to receive data from a mobile phone, and then display the received data on an LCD module. Bluetooth technology enables wireless communication between devices, making it ideal for IoT applications that require remote control or data transfer.

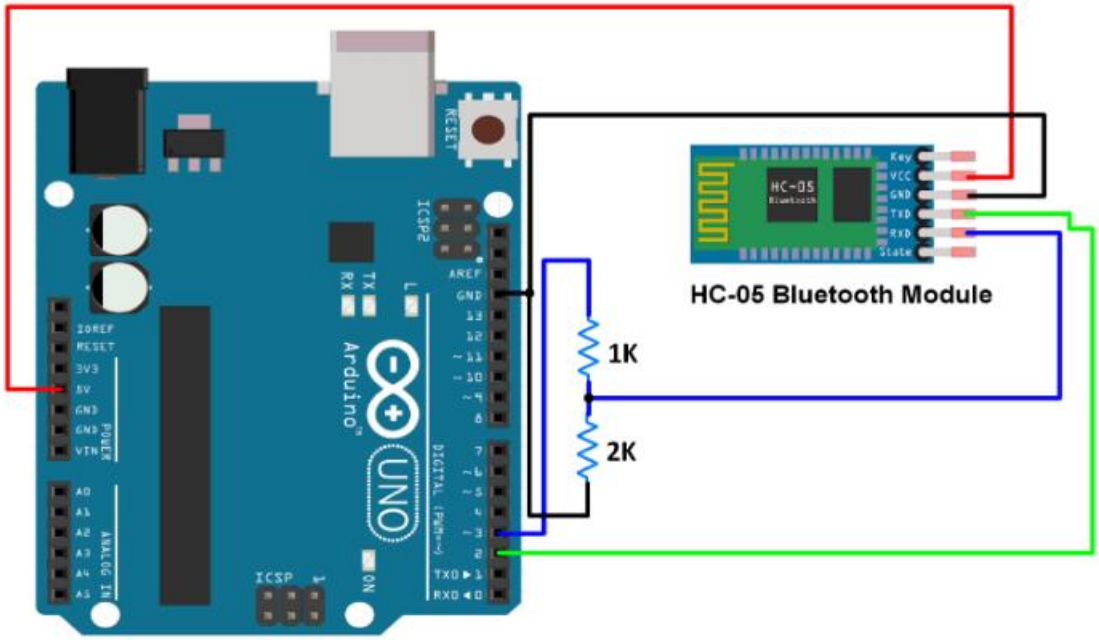
COMPONENTS REQUIRED:

We need the following components –

- Arduino board (e.g. Arduino UNO)
- Bluetooth module (e.g. HC05)
- LCD module (e.g. 16x2 character LCD)
- Jumper wires
- Breadboard
- Mobile phone with Bluetooth capabilities

PROCEDURE:

1. Connect the Bluetooth module to the Arduino board:
   - Connect the VCC pin of the Bluetooth module to the 5V pin of the Arduino.
   - Connect the GND pin of the Bluetooth module to the GND pin of the Arduino.
   - Connect the TXD pin of the Bluetooth module to the RXD pin of the Arduino.
   - Connect the RXD pin of the Bluetooth module to the TXD pin of the Arduino.
2. Connect the LCD module to the Arduino board:
   - Connect the VSS pin of the LCD module to the GND pin of the Arduino.
   - Connect the VDD pin of the LCD module to the 5V pin of the Arduino.
   - Connect the V0 pin of the LCD module to a potentiometer to adjust the contrast.
   - Connect the RS, EN, D4, D5, D6, and D7 pins of the LCD module to digital pins of the Arduino.



Interfacing HC-05 Bluetooth Module with Arduino UNO

ARDUINO CODE:

```
#include <LiquidCrystal.h> // Include the LiquidCrystal library

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // Define the pins for the LCD module

void setup() {
  Serial.begin(9600); // Initialize serial communication
  lcd.begin(16, 2); // Initialize the LCD module
  lcd.print("Bluetooth Data:"); // Display initial message on LCD
}

void loop() {
  if (Serial.available() > 0) { // Check if data is received from Bluetooth module
    char data = Serial.read(); // Read the data received
    lcd.setCursor(0, 1); // Set cursor to second row of LCD
    lcd.print("Received: "); // Display received data on LCD
    lcd.write(data); // Write the received data to LCD
  }
}
```

CODE TO NOTE:

1. Serial.begin(9600): This function initializes the serial communication at a baud rate of 9600 bits per second, which should match the baud rate of the Bluetooth module for proper communication.
2. Serial.available(): This function checks if there is any data available in the serial buffer.
3. Serial.read(): This function reads a character from the serial buffer.
4. lcd.setCursor(row, col): This function sets the cursor position on the LCD to the specified row and column.
5. lcd.print(): This function writes a string to the LCD.
6. lcd.write(): This function writes a character to the LCD.

CONCLUSION:

In this lab, we learned how to interface a Bluetooth module with an Arduino board to receive data from a mobile phone, and then display the received data on an LCD module. This can be useful in various IoT applications, such as remote control of devices or wireless data transfer. Understanding how to interface Bluetooth modules with Arduino expands the possibilities of IoT projects, providing wireless communication capabilities.

EXPERIMENT- 2.8

AIM OF THE EXPERIMENT: Interfacing Relay module to demonstrate Bluetooth based home automation application. (using Bluetooth and relay).

INTRODUCTION:

In this lab, we will learn how to interface a relay module with an Arduino board to demonstrate a Bluetooth-based home automation application. Relays are electromechanical switches that can be controlled by an Arduino to turn on/off high voltage devices, making them suitable for home automation projects. By integrating Bluetooth communication, we can control the relay module wirelessly using a mobile phone or any Bluetooth-enabled device.

COMPONENTS REQUIRED:

We need the following components –

- Arduino board (e.g. Arduino UNO)
- Relay module (e.g. 1-channel relay module)
- Bluetooth module (e.g. HC05)
- Jumper wires
- Breadboard
- High voltage device (e.g. bulb, fan, etc.)
- Mobile phone with Bluetooth capabilities

PROCEDURE:

1. Connect the relay module to the Arduino board:
    - Connect the VCC pin of the relay module to the 5V pin of the Arduino.
    - Connect the GND pin of the relay module to the GND pin of the Arduino.
    - Connect the IN pin of the relay module to a digital pin of the Arduino (e.g. D2).
2. Connect the Bluetooth module to the Arduino board:
    - Connect the VCC pin of the Bluetooth module to the 5V pin of the Arduino.
    - Connect the GND pin of the Bluetooth module to the GND pin of the Arduino.
    - Connect the TXD pin of the Bluetooth module to the RXD pin of the Arduino.
    - Connect the RXD pin of the Bluetooth module to the TXD pin of the Arduino.

3. Connect the high voltage device to the relay module:
    - Connect one terminal of the high voltage device to the Common (COM) terminal of the relay module.
    - Connect the other terminal of the high voltage device to the Normally Open (NO) terminal of the relay module.

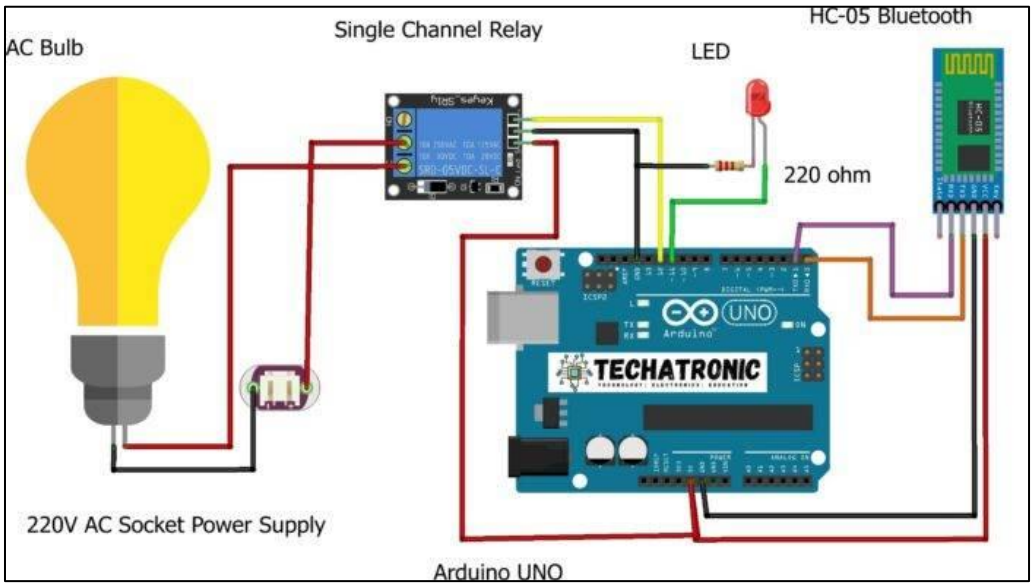


Fig. : Arduino circuit diagram for Bluetooth based home automation application.

CODE TO NOTE:

1. digitalWrite(pin, value): This function sets the specified digital pin to the given value (HIGH or LOW), which controls the relay to turn ON or OFF the high voltage device.
2. Serial.available(): This function checks if there is any data available in the serial buffer.
3. Serial.read(): This function reads a character from the serial buffer.

CONCLUSION:

In this lab, we learned how to interface a relay module with an Arduino board to demonstrate a Bluetooth-based home automation application. By using Bluetooth communication, we can remotely control the relay module and switch ON/OFF high voltage devices, showcasing the potential of IoT in home automation. This lab provides a practical hands-on experience in interfacing relay modules with Arduino and integrating Bluetooth communication for IoT applications.