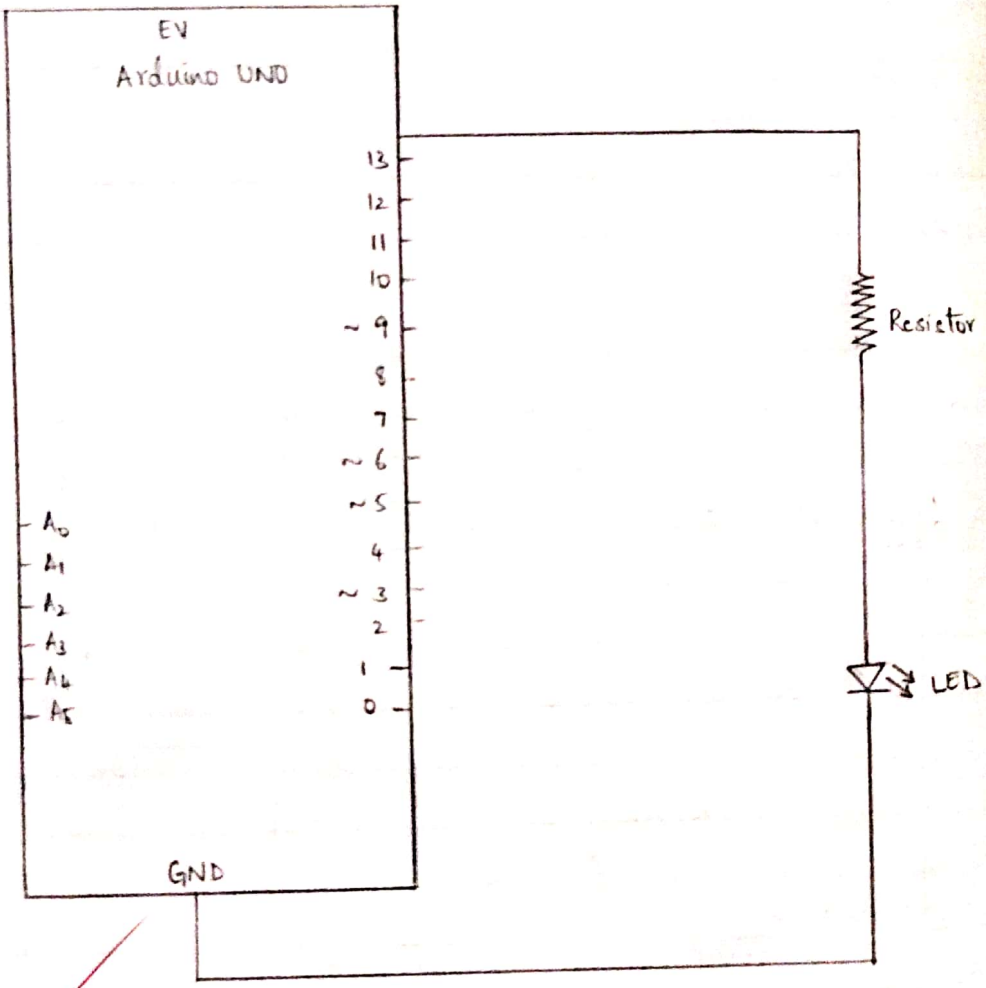
SCHEMATICS :



BLINKING OF AN LED USING ARDUINO UNO

Experiment No. : 1 Date : 4/7/2024

EXPERIMENT 1 : BLINK

AIM: To turn an LED on and off every second.

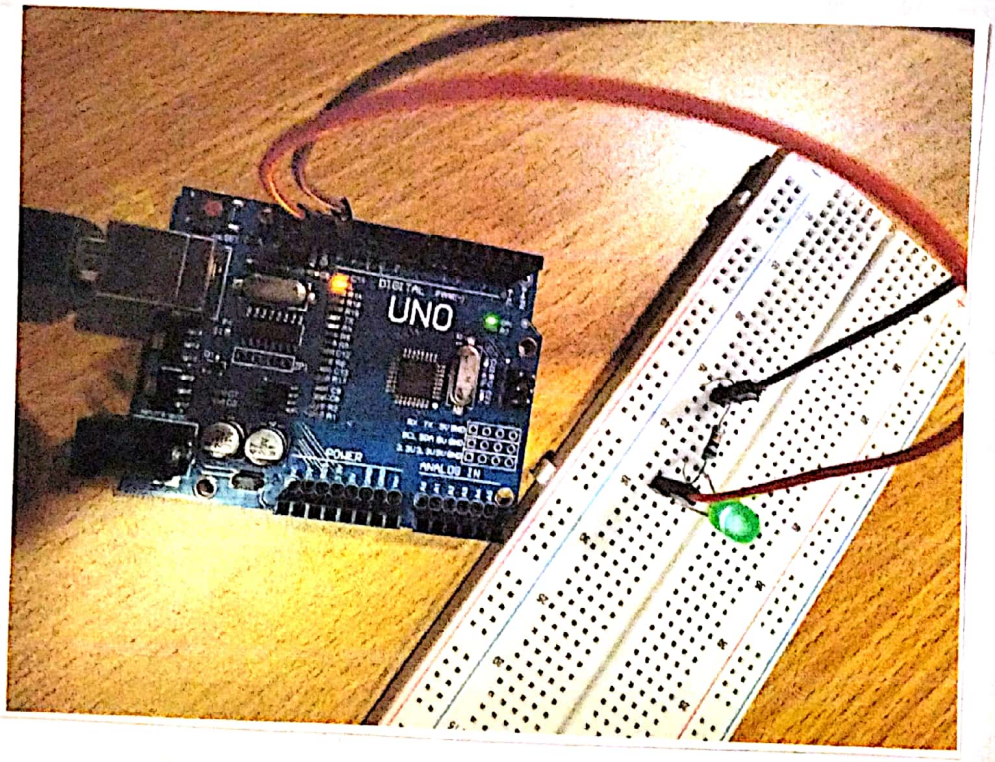
HARDWARE REQUIRED: Arduino UNO board, LED, Resistor, Jump wires, Bread board.

CIRCUIT: The LED is connected to a digital pin and its number may vary from board type to board type. We have a constant that is specified in every board description file. This constant is LED_BUILTIN constant. Connect the long leg of the LED (the positive leg called anode) to the other end of the resistor. Connect the short leg of the LED (the negative leg called cathode) to the GND in the diagram as shown on the UNO Board that has D-B as the LED_BUILTIN value. The value of the resistor in series with the LED may be of a different value than 220Ω. The LED will also light up with values upto 1kΩ.

CODE:

```
#define led_pin 13
void setup()
{
    pinMode(led_pin, OUTPUT);
}
void loop()
{
    digitalWrite(led_pin, HIGH);
    delay(1000);
    digitalWrite(led_pin, LOW);
```

OUTPUT :



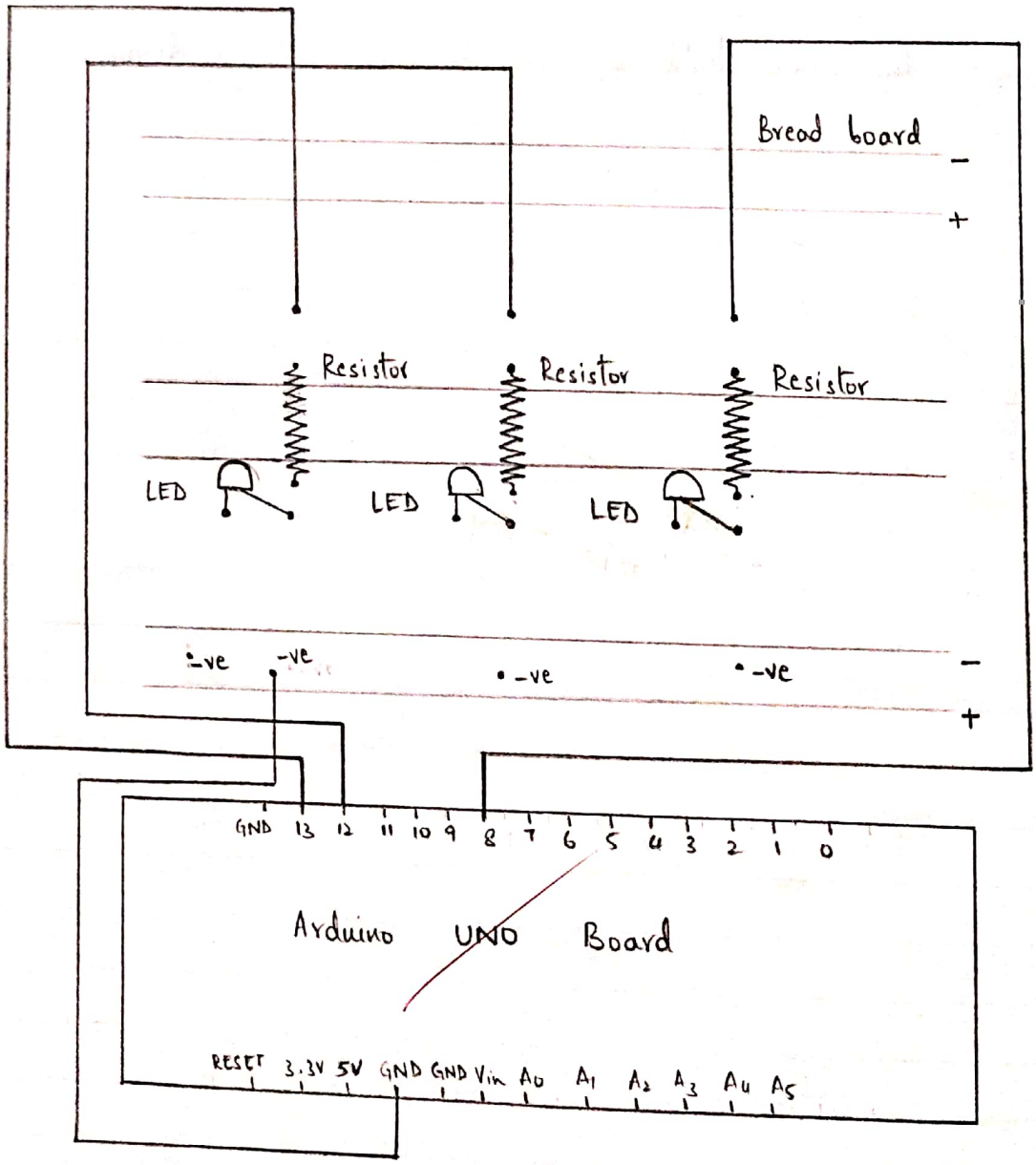
Experiment No. : 1 Date : 24/4/2024

```
    delay (10000);
}
```

RESULT : The LED light is seen blinking, with a delay.

08/05/2024

SCHEMATICS:



BLINKING OF 3 LEDs WITH ARDUIND UNO

Experiment No. : 2 Date : 8/5/2024

EXPERIMENT 2 : 3 LED BLINK

AIM: To demonstrate alternate blinking of LEDs with Arduino board.

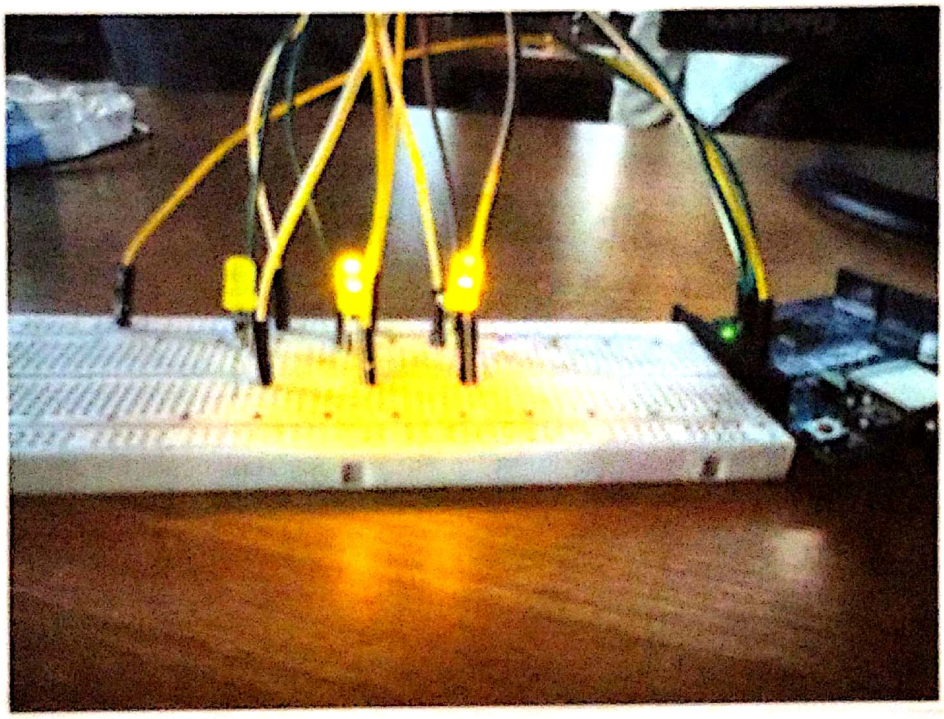
HARDWARE REQUIRED: Arduino UNO board, Bread board, 2 LEDs, Jump wires, 2(ohm) resistors.

CIRCUIT: We connect the three LEDs to pins of the Arduino board. The limiting value of resistance between 220-330 ohms is done to set the optimal current through the LEDs. The required resistance is enough to light up an LED without damaging the board and the LED will turn off individually. The circuit is connected in the same way as shown in the schematics. First, the LED is connected in series with the resistor. Then, the negative terminal of the LEDs is connected to GND (ground).

CODE:

```
void setup()
{
    pinMode(13, OUTPUT);
    pinMode(12, OUTPUT);
    pinMode(8, OUTPUT);
}
void loop()
{
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
```

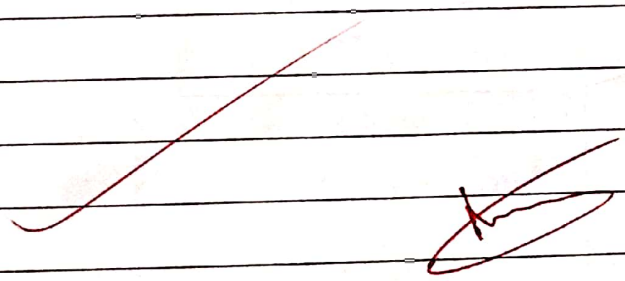
OUTPUT:



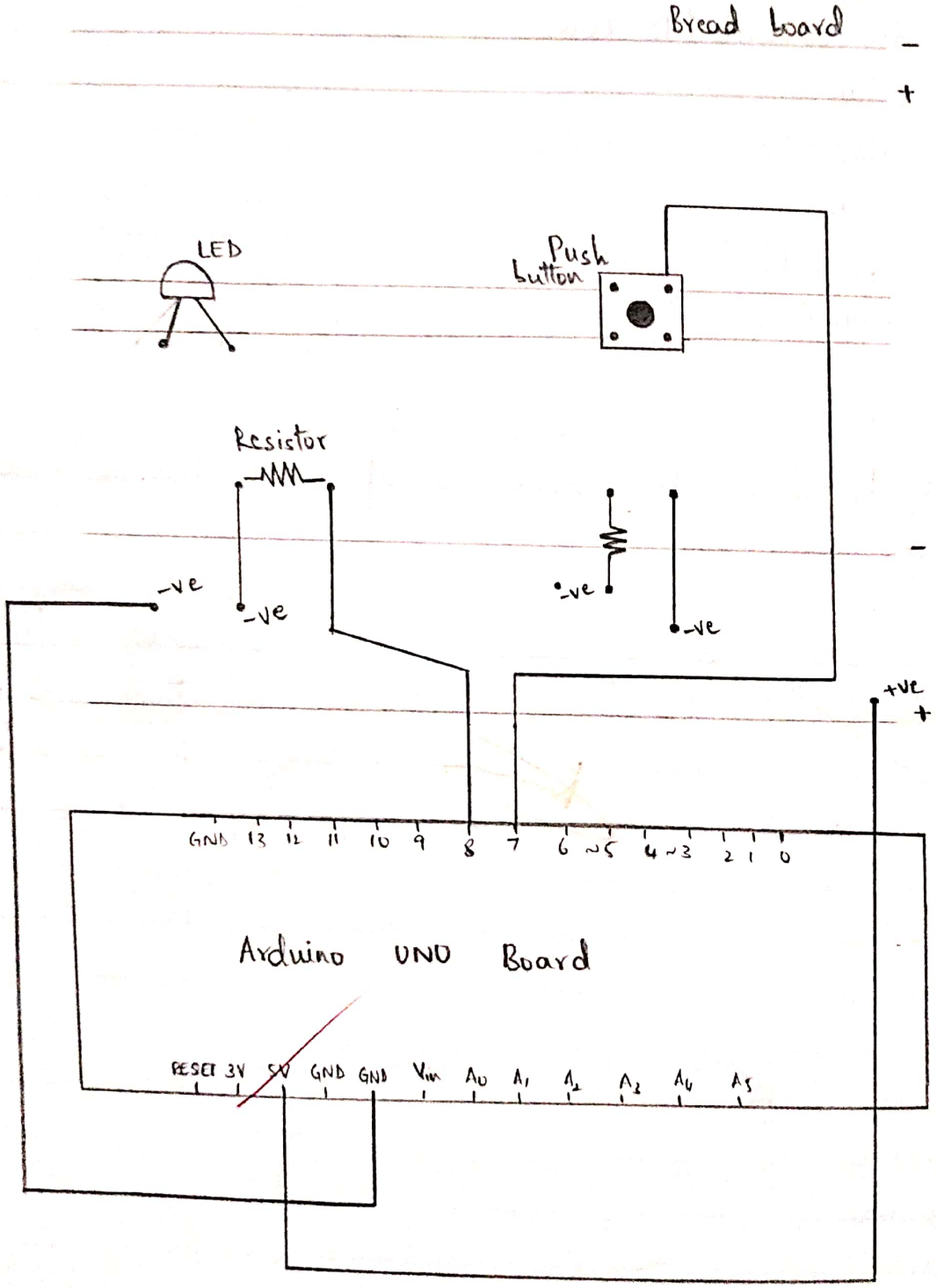
Experiment No. : 2 Date : 8/5/2024

```
    digitalWrite (12, HIGH);
    delay (1000);
    digitalWrite (12, LOW);
    delay (1000);
    digitalWrite (8, HIGH);
    delay (1000);
    digitalWrite (8, LOW);
    delay (1000);
}
```

RESULT: The three LEDs blink one after the other, in series, with a delay.



SCHEMATICS:



BLINKING OF AN LED USING PUSH BUTTON

Experiment No. : 3 Date : 15/5/2024

EXPERIMENT 3: LED BLINKING WITH PUSH BUTTON

AIM: To demonstrate blinking of an LED using push button and Arduino UNO.

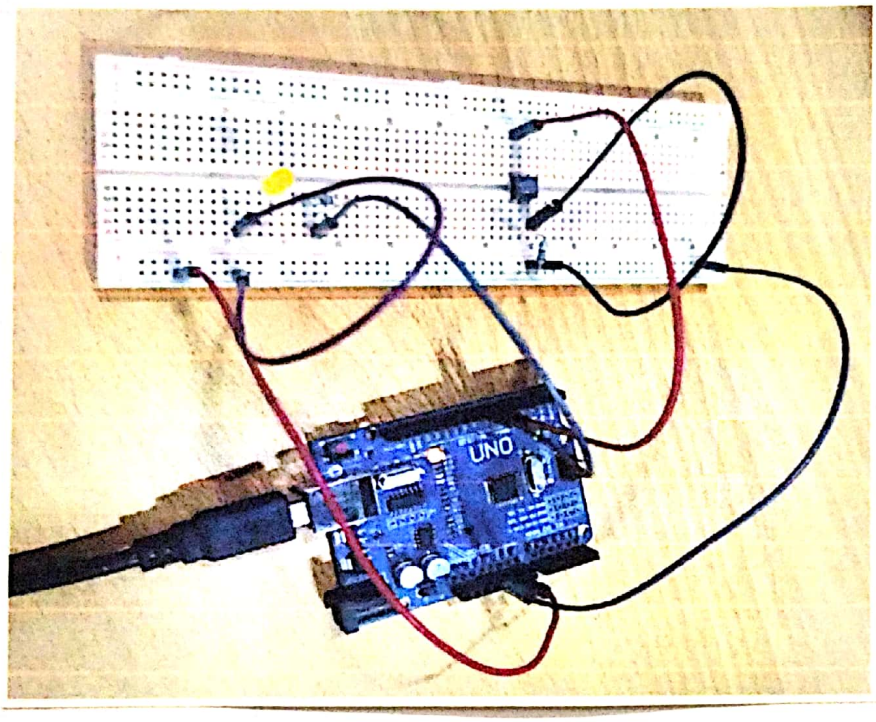
HARDWARE REQUIRED: Arduino UNO board, Bread board, Jumper wires, LED, momentarily tactile four-pin push button, two ohmic resistors.

CIRCUIT: A push button completes the circuit when pressed. As soon as the button is released, the connection will spring back and break the circuit, turning it off. The push button switch is also known as a momentary or normally open switch, and is used in computer keyboards. The connection for the circuit is made according to the schematics. The negative of the bread board is connected to the ground, and the positive of the bread board is connected to 6V supply of the Arduino UNO board. The LED and resistor are connected, along with the push button and the resistor, in the same way as depicted in the schematics.

CODE:

```
#define LED_PIN 8
#define BUTTON_PIN 7
void setup()
{
    pinMode(LED_PIN, OUTPUT);
    pinMode(BUTTON_PIN, INPUT);
}
void loop()
{
```

OUTPUT:

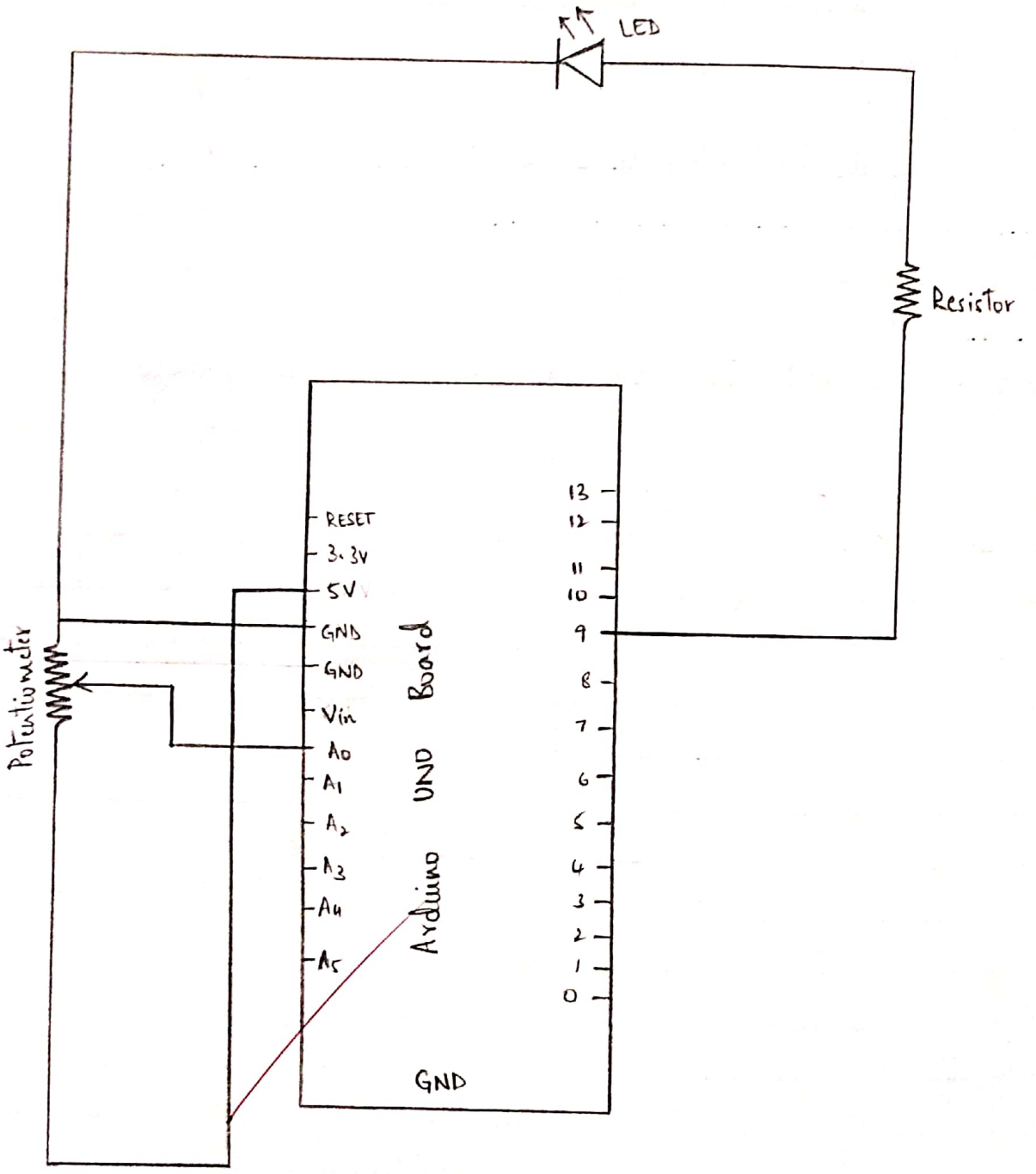


Experiment No. : 3 Date : 15/5/2024

```
    if (digitalRead( BUTTON_PIN == HIGH))
    {
        digitalWrite ( LED_PIN, HIGH);
    }
    else {
        digitalWrite ( LED_PIN, LOW);
    }
}
```

RESULT: The LED turns on when the button is pressed.

SCHEMATICS:



CONTROLLING BRIGHTNESS OF AN LED USING A POTENTIOMETER

Experiment No. : 4 Date : 15/5/2024

EXPERIMENT 4 : POTENTIOMETER

AIM: To control the brightness of an LED using a potentiometer.

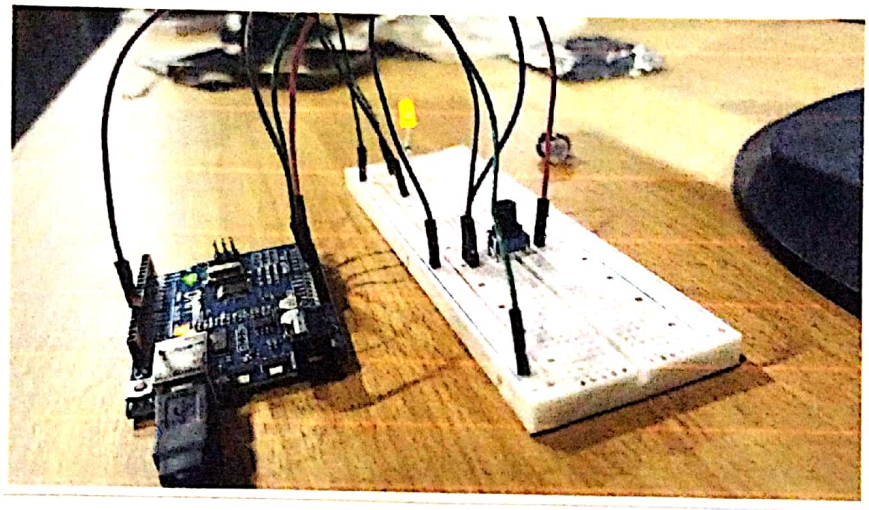
HARDWARE REQUIRED: Arduino UNO board, Bread board, LED, ohm resistor, Potentiometer, Jumper wires.

CIRCUIT: First, connect the GND pin of the Arduino UNO and the negative line on the bread board. From this negative line, we will be able to connect all the other grounds, which will make things easier. Connect the shorter leg to the GND of the Arduino UNO board. Connect the ohmic resistor to the long leg, and connect the resistor to the Arduino UNO board using a jumper to 9 pin. Plug 3 legs of the potentiometer to 3 different lines on the bread board. Connect middle to the pin of the Arduino UNO (A0). Then, connect the two legs to the negative and positive lines as shown. Connect the negative line of the bread board to GND of the Arduino UNO board, and the positive line to 5V of the Arduino UNO board.

CODE:

```
#define LED_PIN 9
#define POTENTIOMETER_PIN A0
void setup()
{
    pinMode(LED_PIN, OUTPUT);
}
void loop()
{
```

OUTPUT:



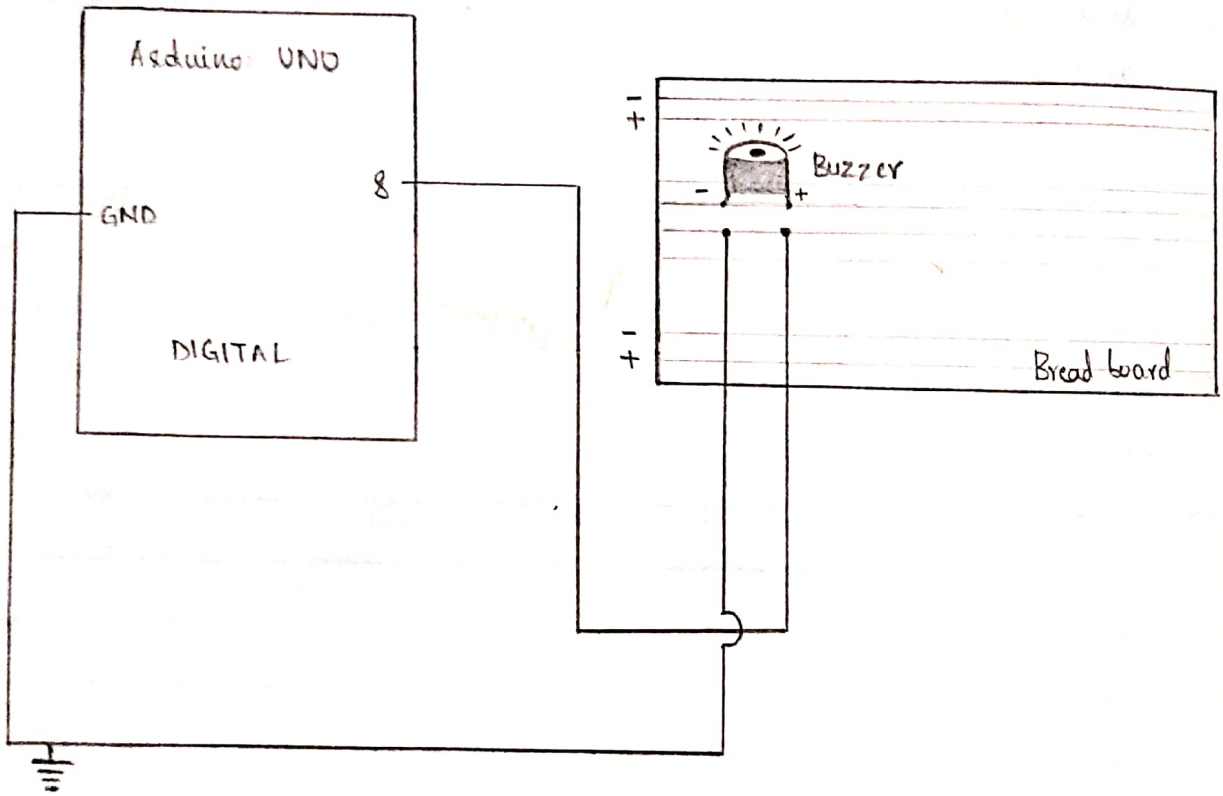
Experiment No. : 4 Date : 15/5/2024

```
    int potentiometerValue = analogRead (POTENTIOMETER_PIN);
    int brightness = potentiometerValue / 4;
    analogWrite (LED_PIN, brightness);
}
```

RESULT: As the potentiometer's regulator was turned clockwise and anti-clockwise, the brightness of the LED increased and decreased.

22/05/2024

SCHEMATICS:



PRODUCING DIFFERENT SOUNDS FROM A BUZZER USING ARDUINO UNO

Experiment No. : 5 Date : 22/5/2024

EXPERIMENT 5: BUZZER

AIM: To produce different sounds using a buzzer and Arduino UNO.

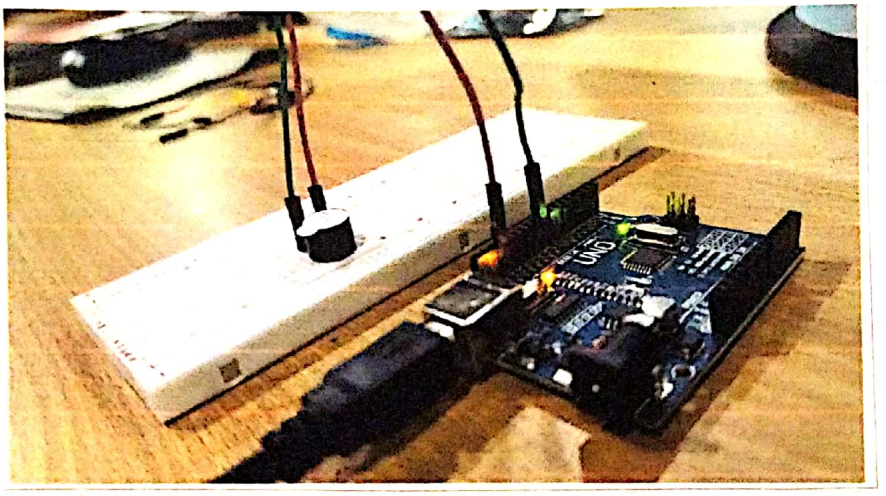
HARDWARE REQUIRED: Arduino UNO board, Buzzer, Bread board, Jumper wires.

CIRCUIT: Here, we use an Arduino UNO board to control a buzzer and generate various sounds by programming the Arduino UNO, and controlling the frequency and duration of the buzzer's output to create different tones. To do this, connect the positive leg of the buzzer to the digital pin 8 on the Arduino UNO board, and the negative leg to the GND (ground) of the Arduino UNO board. The Arduino UNO board is then connected to the computer using USB. The code given below is written in the Arduino IDE software. On uploading the code, the Arduino UNO board will generate different sounds through the buzzer.

CODE:

```
#define buzzer 8
void setup()
{
    pinMode(buzzer, OUTPUT);
}
void loop()
{
    tone(buzzer, 10000, 1000); // frequency = 10000 Hz for 1000 ms
    delay(1000);
```

OUTPUT:



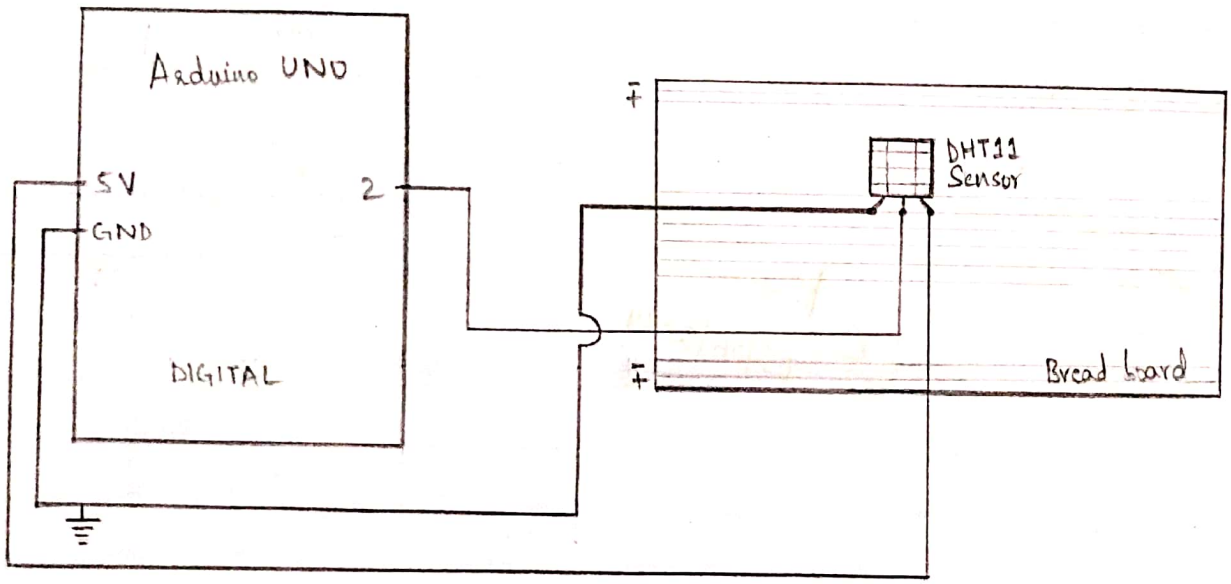
Experiment No. : 5 Date : 22/5/2024

```
    noTone(buzzer);
    delay(1000);
}
```

RESULT: The buzzer produces different sounds, the frequency and duration of which can be altered by modifying the code.

05/06/2024

SCHEMATICS:



DISPLAYING THE TEMPERATURE AND HUMIDITY WITH A DHT11 SENSOR USING ARDUINO UNO

Experiment No. : 6 Date : 5/6/2024

EXPERIMENT 6 : DHT11

AIM: To display the temperature and humidity readings by developing a program using Arduino UNO to be interfaced with a DHT11 sensor.

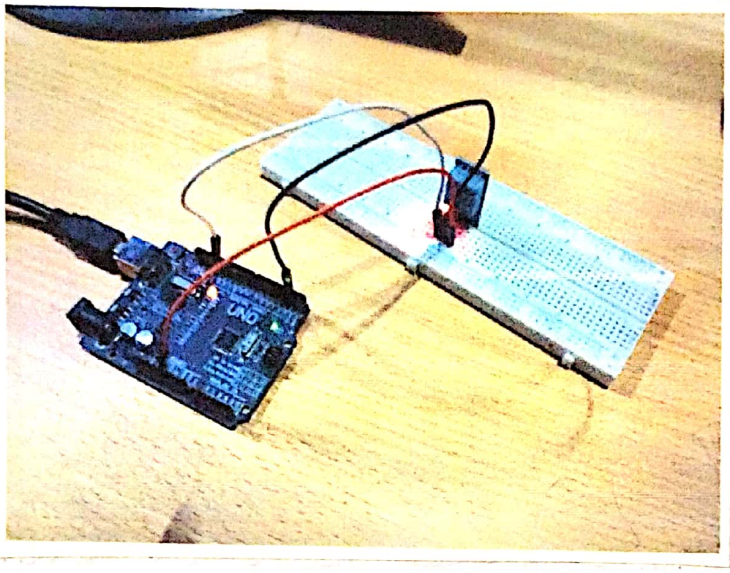
HARDWARE REQUIRED: Arduino UNO board, DHT11 sensor, Bread board, Jumper wires.

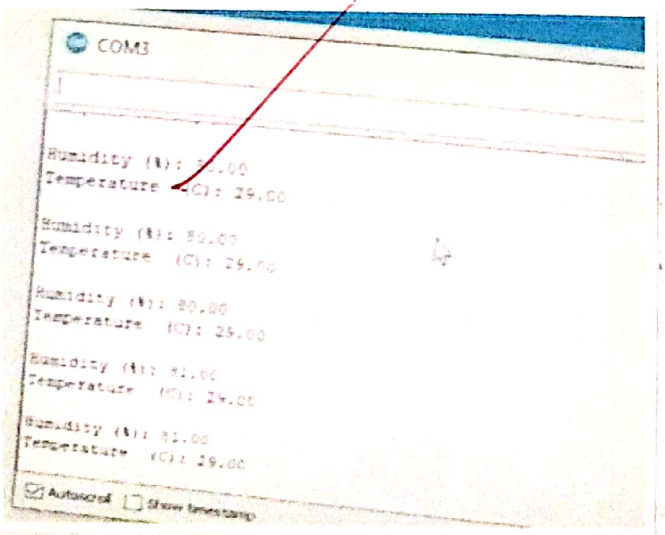
CIRCUIT: Connect the 3-pin DHT11 sensor, using three jumper wires, with the Arduino UNO board. Connect pin 2 of the DHT11 sensor to pin 2 of the Arduino UNO board. Connect pin 1 of the DHT11 sensor to 5V pin of the Arduino UNO board. Finally, connect pin 3 of the DHT11 sensor to GND of the Arduino UNO board. Click on "Serial Monitor" to view the output.

CODE:

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
void setup()
{
    Serial.begin(9600);
    Serial.println(F("DHTxx test!"));
    dht.begin();
}
void loop()
{
```

OUTPUT:



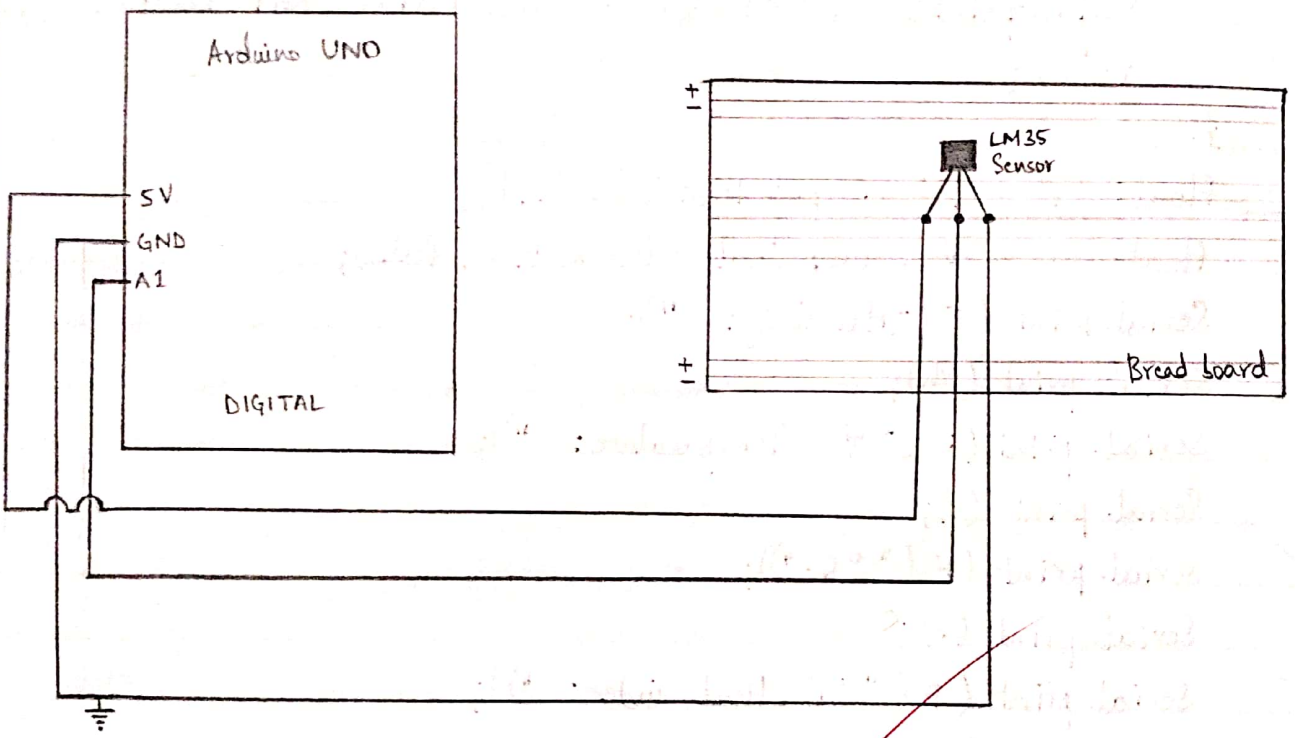


Experiment No. : 6 Date : 5/6/2024

```
delay (2000);
float h = dht.readHumidity ();
float t = dht.readTemperature ();
float f = dht.readTemperature (true);
if (isnan(h) || isnan(t) || isnan(f))
{
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
}
float hif = dht.computeHeatIndex (f, h);
float hic = dht.computeHeatIndex(t, h, false);
Serial.print (F("Humidity: "));
Serial.print (h);
Serial.print(F(" %  Temperature: "));
Serial.print (t);
Serial.print (F("°C "));
Serial.print (f);
Serial.print (F("°F Heat index: "));
Serial.print (hic);
Serial.print (F("°C "));
Serial.print (hif);
Serial.println (F("°F"));
}
```

RESULT: The humidity and temperature is displayed on the serial monitor using the DHT11 sensor.

SCHEMATICS:



DISPLAYING THE TEMPERATURE READINGS USING LM35 AND ARDUINO UNO

Experiment No. : 7 Date : 5/6/2024

EXPERIMENT 7 : LM35

AIM: To display the temperature readings by developing a program using Arduino UNO to be interfaced with an LM35 sensor.

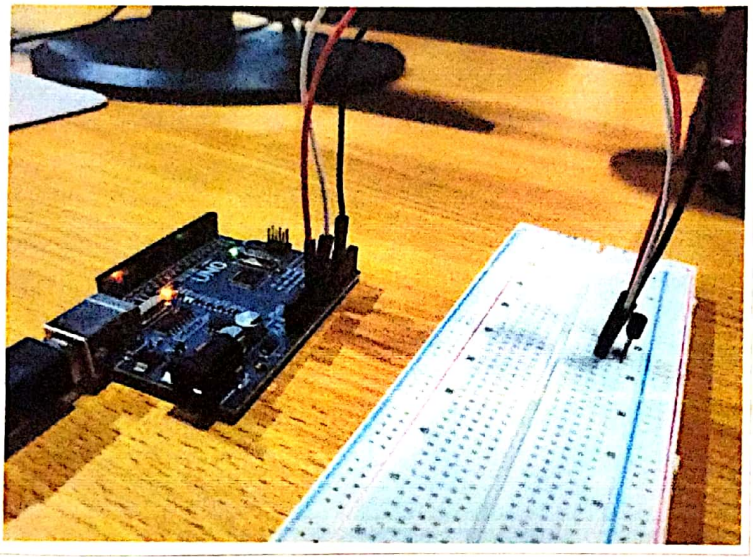
HARDWARE REQUIRED: Arduino UNO board, LM35 sensor, Bread board, Jumper wires.

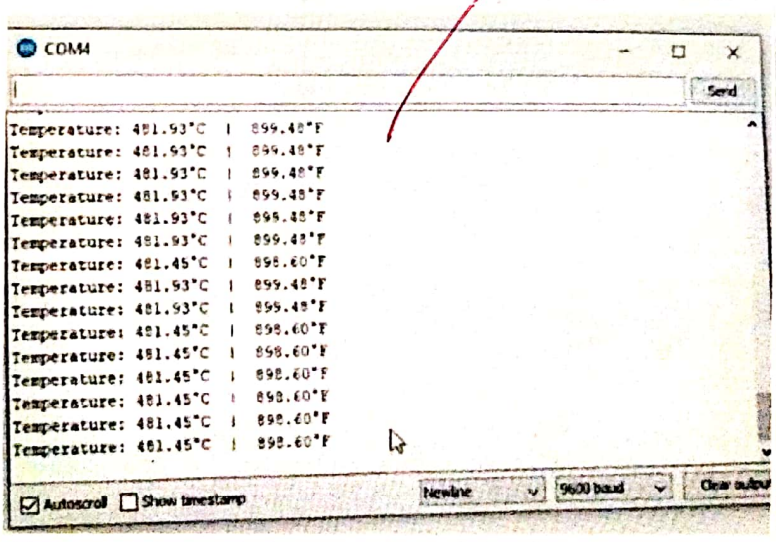
CIRCUIT: LM35 is a Temperature sensor which can measure temperature in the range of -55°C to 150°C. It is a three-Terminal device that provides analog voltage proportional to the temperature. Higher the temperature, higher is the output voltage. The output analog voltage can be converted to digital using ADC, so that a microcontroller can process it. To set up the circuit, connect the 3-pin LM35 sensor to the Arduino UNO using three wires. Connect the pin 2 of LM35 sensor to the analog pin A1 of the Arduino UNO board. Connect pin 1 of the sensor to 5V of the Arduino UNO. Connect pin 3 of the LM35 sensor to GND of the Arduino UNO board. Write and upload the code to the Arduino UNO board. Click on "Serial Monitor" to view the output.

CODE:

```
#define sensorPin A1
void setup()
{
    Serial.begin(9600);
}
void loop()
{
```

OUTPUT:





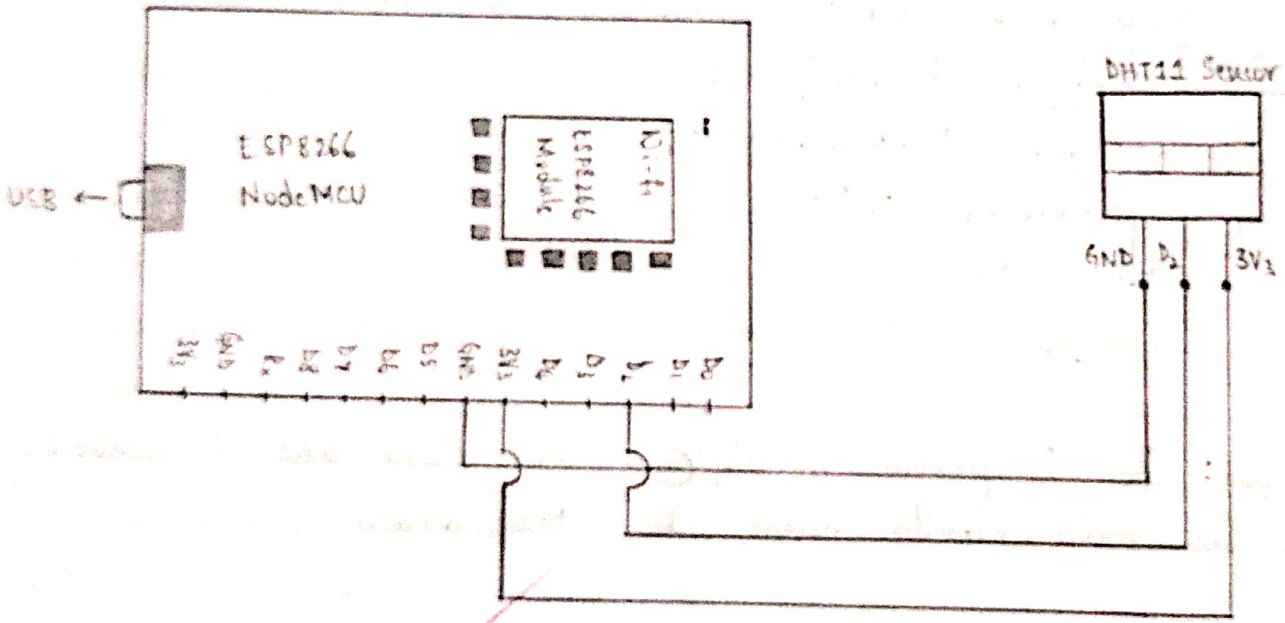
Experiment No. : 7 Date : 5/6/2024

```
int reading = analogRead(sensorPin);
float voltage = reading * (5.0 / 1024.0);
float temperatureC = voltage * 100;
Serial.print("Temperature: ");
Serial.print(temperatureC);
Serial.print("\xC2\xB0");
Serial.print("C  |  ");
float temperatureF = (temperatureC * 9.0/5.0) + 32.0;
Serial.print(temperatureF);
Serial.print("\xC2\xB0");
Serial.println("F");
delay(1000);
}
```

RESULT: The temperature is displayed in Celsius and Fahrenheit on the serial monitor using the LM35 sensor.

26/06/2024

SCHEMATICS



DISPLAYING THE TEMPERATURE AND HUMIDITY WITH A DHT11 SENSOR INTERFACED WITH ESP8266 NodeMCU BOARD

Experiment No. : 8 Date : 26/6/2024

EXPERIMENT 8: ESP8266 NODEMCU WITH DHT11

AIM: To design and develop a program using an ESP8266 NodeMCU board to interface with DHT11 sensor, and display the temperature and humidity readings.

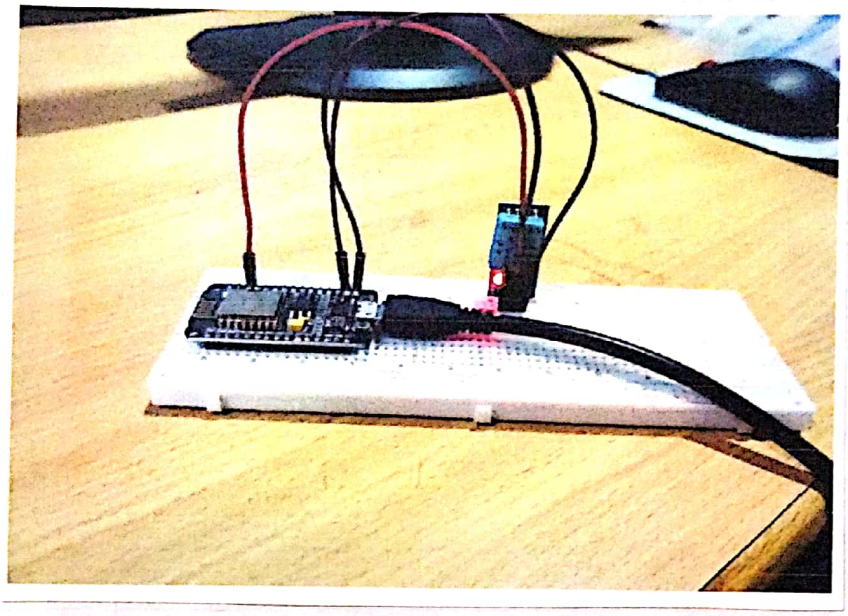
HARDWARE REQUIRED: ESP8266 NodeMCU board, DHT11 sensor, Breadboard, Jumper wires.

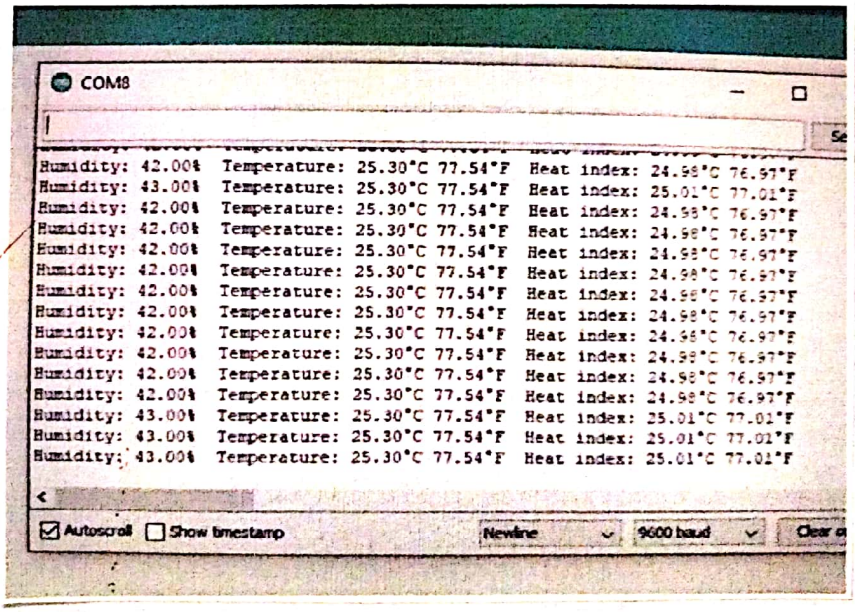
CIRCUIT: The DHT11 sensor is a digital temperature and humidity sensor that communicates with NodeMCU using a one-wire protocol. The program reads the sensor data and displays the readings on the serial monitor. Connect the Vcc pin of the DHT11 sensor to the 3.3V pin on the NodeMCU. Connect the GND pin of the DHT11 sensor to the GND pin on the NodeMCU. Connect the data pin of the DHT11 sensor to a digital input/output (I/O) pin such as D2 on the NodeMCU. Upload the Arduino sketch to the NodeMCU using the Arduino IDE. Open the serial monitor on the Arduino IDE to view the readings.

CODE:

```
#include "DHT.h"
#define DHTPIN D2
#define DHTTYPE DHT11
DHT dht (DHTPIN, DHTTYPE);
void setup ()
{
    Serial.begin (9600);
    Serial.println (F("DHTxx test!"));
```

OUTPUT:





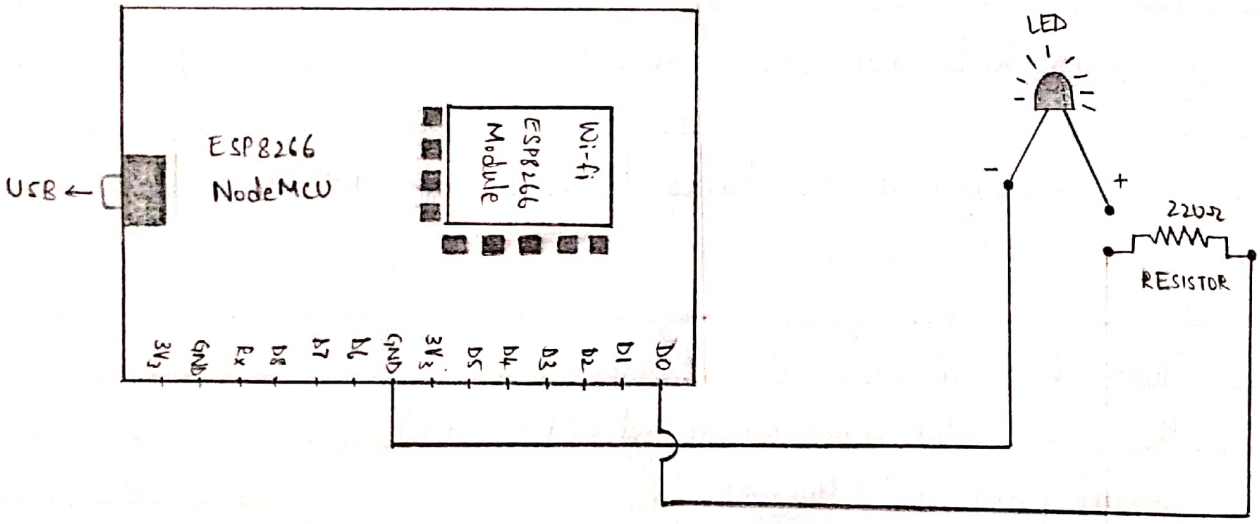
Experiment No. : 8 Date : 26/6/2024

```
    dht.begin();
}
void loop()
{
    delay(2000);
    float h = dht.readHumidity();
    float t = dht.readTemperature();
    float f = dht.readTemperature(true);
    if(isnan(h) || isnan(t) || isnan(f))
    {
        Serial.println(F("Failed to read from DHT sensor!"));
        return;
    }
    float hif = dht.computeHeatIndex(f,h);
    float hic = dht.computeHeatIndex(t,h,false);
    Serial.print(F("Humidity: "));
    Serial.print(h);
    Serial.print(F("% Temperature: "));
    Serial.print(t);
    Serial.print(F("°C "));
    Serial.print(f);
    Serial.print(F("°F Heat index: "));
    Serial.print(hic);
    Serial.print(F("°C "));
    Serial.print(hif);
    Serial.println(F("°F")); }
```

03/07/2024

RESULT: The DHT11 sensor provides accurate temperature and humidity readings when interfaced with ESP8266 NodeMCU.

SCHEMATICS:



CONTROLLING AN LED USING ESP8266 NodeMCU BOARD

Experiment No. : 9 Date : 3/7/2024

EXPERIMENT 9: ESP8266 NodeMCU WITH ONE LED

AIM: To design and develop a program to control an LED connected to an ESP8266 NodeMCU board.

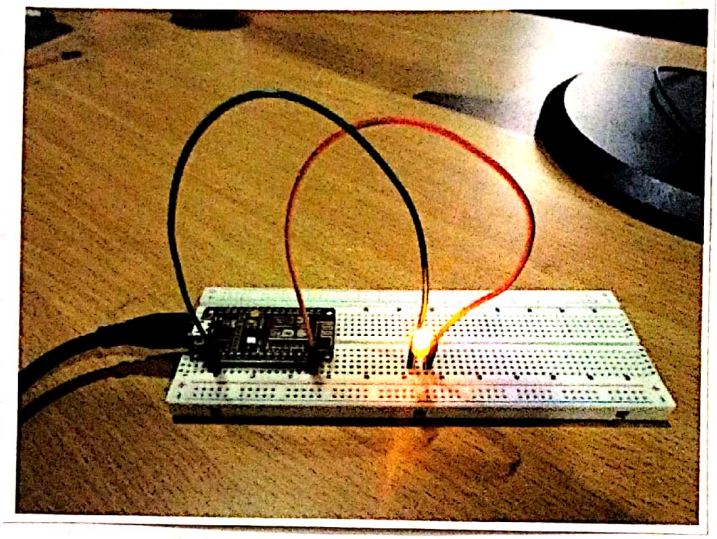
HARDWARE REQUIRED: ESP8266 NodeMCU board, LED, 220-ohm resistor, Bread board, Jumper wires.

CIRCUIT: The ESP8266 NodeMCU board has multiple GPIO (General Purpose Input Output) pins that can be used to control external devices like LEDs. We use one of these GPIO pins to connect the LED with a current-limiting resistor. Connect the anode of the LED to one end of the 220Ω resistor. Connect the other end of the resistor to GPIO pin (say, D0) on the NodeMCU board. Connect the cathode of the LED to the ground of the bread board. Upload the Arduino sketch to the NodeMCU board using the Arduino IDE.

CODE:

```
#define LED D0
void setup()
{
    pinMode(LED, OUTPUT);
}
void loop()
{
    digitalWrite(LED, HIGH);
    delay(1000);
    digitalWrite(LED, LOW);
```

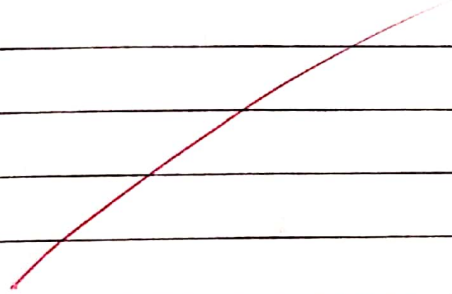
OUTPUT:



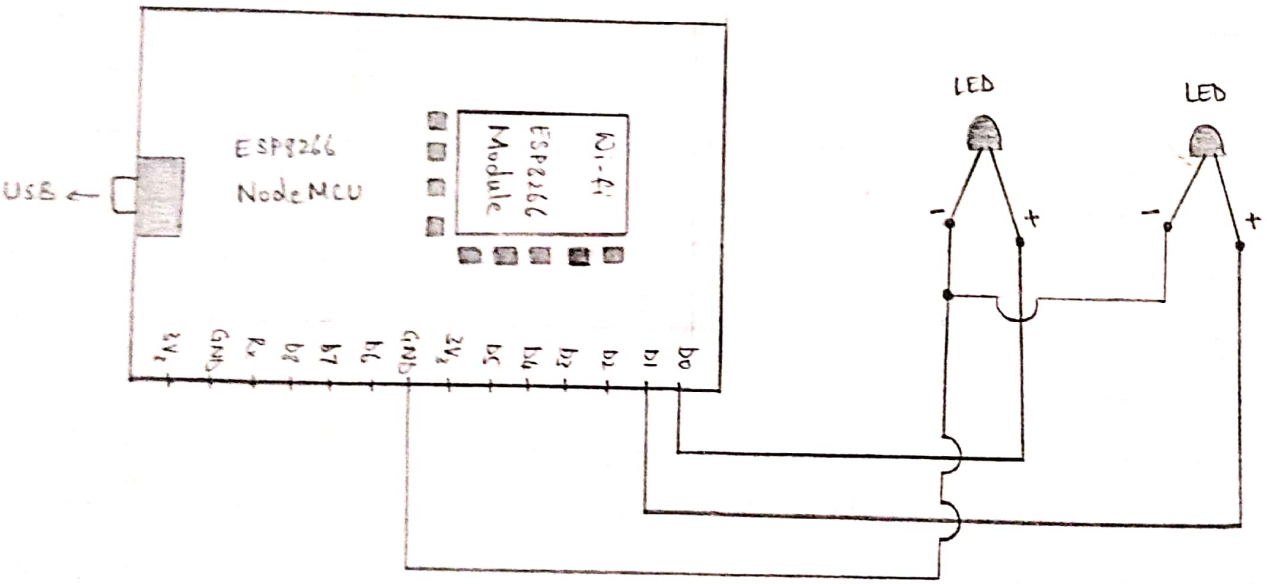
Experiment No. : 9 Date : 3/7/2024

```
    delay (1000);
}
```

RESULT: The LED, controlled using an ESP8266 NodeMCU board, switches on and off automatically, as per the program's logic.



SCHEMATICS:



CONTROLLING TWO LEDS USING AN ESP8266 NodeMCU BOARD

Experiment No. : 10 Date : 3/7/2024

EXPERIMENT 10: ESP8266 NodeMCU WITH TWO LEDs

AIM: To design and develop a program to control two LEDs connected to an ESP8266 NodeMCU board.

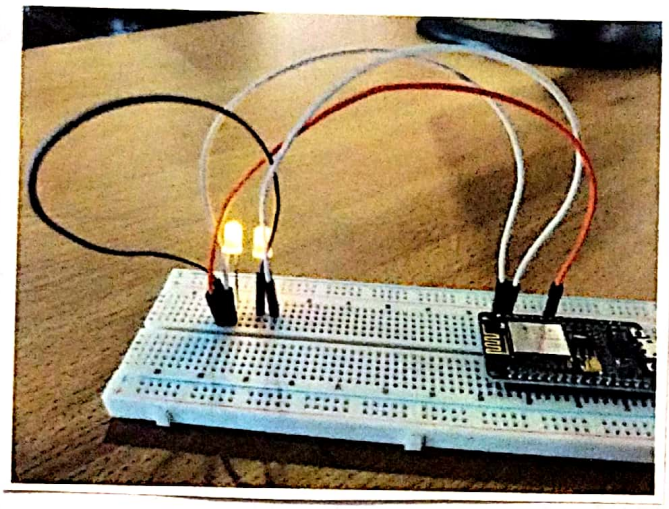
HARDWARE REQUIRED: ESP8266 NodeMCU board, 2 LEDs, Jumper wires, Bread board.

CIRCUIT: Place the two LEDs on the bread board. Using jumper wires, connect the cathode of each LED to a common ground on the breadboard. Connect the anode of one LED to GPIO pin D0. Connect the anode of anode LED to GPIO pin D1. Connect a GND pin on the NodeMCU board to the ground rail on the bread board using a jumper wire. Upload the Arduino sketch to the NodeMCU board using the Arduino IDE.

CODE:

```
#define LED D0
#define LED2 D1
void setup()
{
    pinMode(LED, OUTPUT);
    pinMode(LED2, OUTPUT);
}
void loop()
{
    digitalWrite(LED, HIGH);
    digitalWrite(LED2, HIGH);
    delay(1000);
```

OUTPUT:



Experiment No. : 10 Date : 3/7/2024

```
    digitalWrite(LED, LOW);
    digitalWrite(LED2, LOW);
    delay(1000);
}
```

RESULT: The LEDs, controlled using an ESP8266 NodeMCU board, turn on and off automatically, as per the program's logic.

16/07/2024