



SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade

Kuniamuthur, Coimbatore – 641008

Phone : (0422)-2678001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Accredited by NBA)

RECORD OF LABORATORY WORK

23ITC01 / INTERNET OF THINGS Laboratory

Name of the Student :

Register No. :

Year / Branch / Section : III / CSE & IT /

Semester : V

Academic Year : 2025 – 2026



SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade
Kuniamuthur, Coimbatore – 641008
Phone : (0422)-2678001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (Accredited by NBA)

BONAFIDE CERTIFICATE

Name of the Student :

Register No :

Year : **Branch :** **Section :** **Semester :**

Certified that this is the bonafide record of work done by the above student in
the **23ITC01 / Internet of Things Laboratory** during the academic year 2025
- 2026.

Faculty Incharge

HOD

Submitted for the End Semester Examination held on

Internal Examiner

External Examiner



SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade
Kuniamuthur, Coimbatore – 641008
Phone : (0422)-2678001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

To prepare professionals with high technical, research and entrepreneurial skills as well as ethical values who will contribute to the computational world

MISSION

M1: To develop human resources with the ability and attitude to adapt to emerging technological changes through academic and research-oriented events.

M2: To identify current socio, economic problems of national and international significance and provide solutions through competency centers.

M3: To impart ethics, social responsibilities and necessary professional, entrepreneurial and leadership skills through student lead activities.



SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade
Kuniamuthur, Coimbatore – 641008
Phone : (0422)-2678001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

PROGRAMME OUTCOMES

Engineering Graduates will be able to:

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/ Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct Investigations of Complex Problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade

Kuniamuthur, Coimbatore – 641008

Phone : (0422)-2678001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAMME SPECIFIC OUTCOMES (PSOs)

After the successful completion of the B.E. Computer Science and Engineering programme, the students will be able to:

PSO1: Apply the fundamental knowledge for problem solving and analysis as well as conduct investigations in computer science and engineering for sustainable development.

PSO2: Design and develop the solutions for real time problems and implement them by using modern software tools in lieu of deploying them in the society for its growth.

PSO3: Communicate effectively, adopt ethics and engage in life-long learning.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Be successful in their career in industries associated with Computer Science and Engineering.

PEO2: Comprehend, analyze, design, and create novel products and solutions for the real-life problems.

PEO3: Possess professional and ethical attitude, effective communication skills, team working skills, multi-disciplinary approach, and an ability to relate engineering issues to broader social context.

PEO4: Exhibit leadership qualities and progress through life-long learning.

COMMON INSTRUCTIONS

1. Students are advised to wear ID cards during the lab sessions.
2. Students should attend the lab sessions with proper formal dress code and shoes.
3. Lab manual is mandatory for the lab sessions.
4. Do not wear loose conductive jewelry including metallic rings, bangles, bracelets, wristwatches and neck chains.
5. Do not allow any parts of your body to contact with the live circuit or equipment.
6. Do not turn ON the circuit or equipment without the presence of the Faculty.
7. Use electrical cords/wires only if they are in good condition
8. Do not damage any safety devices such as a fuse or circuit breaker by plug in shorting.
9. Girl students should ensure that their long and loose hair is tied properly.
10. Students are instructed to avoid contacting live terminals with wet hands or wet materials.
11. Power must be switched off whenever an experiment is being assembled disassembled or modified.
12. Laboratory Record must be submitted with all necessary data on the immediate next laboratory class.



SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade
Kuniamuthur, Coimbatore – 641008
Phone : (0422)-2678001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

23ITC01 – INTERNET OF THINGS LABORATORY

Record of laboratory work

ODD SEMESTER-2025-2026

CONTINUOUS EVALUATION SHEET

REFERENCES RUBRICS TABLE

Criteria	Range of Marks			
	Excellent	Good	Average	Below Average
Aim, Identification of Components Required 10 Marks	9-10	7-8	5-6	0-4
Coding / Sketch 30 Marks	27-30	21-26	15-20	0-14
Compilation and Debugging 30 Marks	27-30	21-26	15-20	0-14
Wiring-Execution and Output 20 Marks	18-20	15-17	11-14	0-10
Documentation-Explanation of the experiment/Viva 10 Marks	9-10	7-8	5-6	0-4
Over all Marks	90-100	70-89	50-69	0-49



SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade
Kuniamuthur, Coimbatore – 641008
Phone : (0422)-2678001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

Department of Computer Science and Engineering

Reg. No:

Name of the Student:

Name of the lab:

Index

Exp No	Date	Experiment Name	Page No
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			

Ex No: 1

Date :

Study and Configuration of Arduino kit

Aim:

To study about the installation procedure of “Arduino Uno” and verifying the successful

installation.

Components Required:

Sl.No	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Power Adaptor – 12 V	1
4	Off board LED	1
5	Jumper Wire	2
6	PC with Windows OS -32/64 bit	1

Configuration of Arduino Uno & DIY-JRM TECH's Arduino Uno:

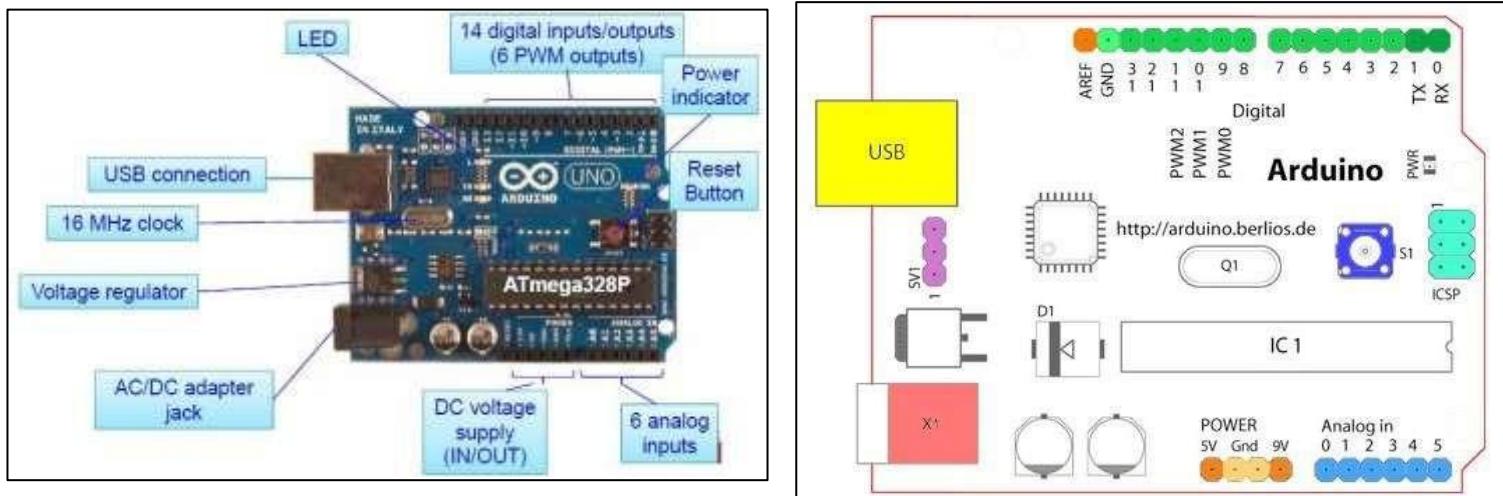


FIG: Standard Arduino UNO

- ❖ Digital I/O Pins 0 – 13;
 - ✓ 0 & 1 (Dark Green) can be **Serial In/Out – Tx/Rx**; These pins cannot be used for digital I/O (*digitalRead* and *digitalWrite*) if we are also using serial communication (e.g. `Serial.begin`)
- ❖ Analog In Pins 0-5 (light blue)
- ❖ ISP – **In System Programming** also called **In-circuit serial programming** (ICSP)
- ❖ External Power Supply In (9-12VDC) – X1 (pink)
- ❖ USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board)

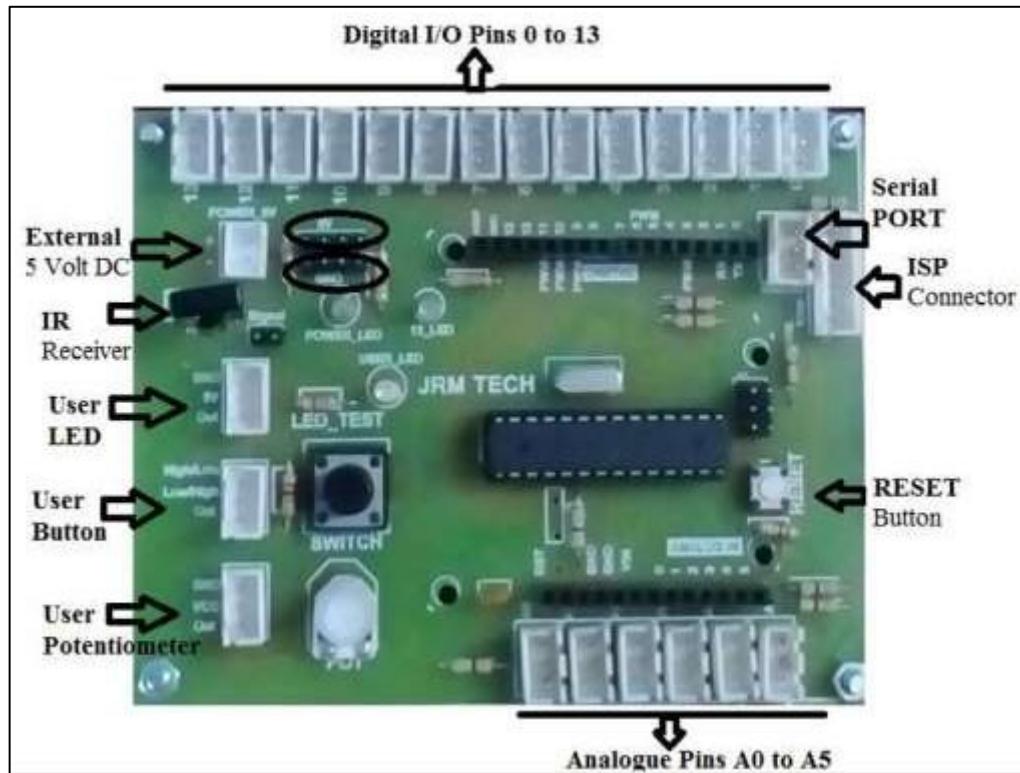


FIG: DIY – JRM Arduino Uno

- One Onboard LED 13th Pin
- One User LED
- One User Button-Switch
- One Potentiometer
- IR Receiver

Installation Procedure

1. Download & install the Arduino environment (IDE)
2. Connect the board to your computer via the USB cable
3. If needed, install the drivers
4. Launch the Arduino IDE
5. Select your board
6. Select your serial port
7. Open the blink example (for checking)
8. Upload the program

Result:

Thus, the study of Arduino Uno and installation of Arduino Uno is done.

Ex No: 2a

Date **Basic Programming using Arduino - LED and Switch Interface**

Aim:

To verify the basic Programming of LED and Switch Interface using Arduino.

Components Required:

Sl.No	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Power Adaptor – 12 V	1
4	Off board LED	1
5	Switch/Keypad Interface	1
6	Jumper Wire	2
7	PC with Windows OS -32/64 bit	1

Procedure

1. Open the Arduino environment (IDE)
2. Connect the board to your computer via the USB cable
3. If needed, install the drivers
4. Launch the Arduino IDE
5. Select your board
6. Select your serial port
7. Open the blink example (for checking)
8. Upload the program

A) Blink the onboard LED in Uno

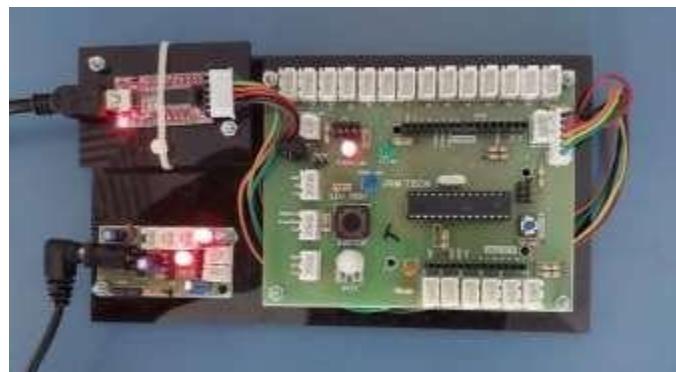
Program:

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

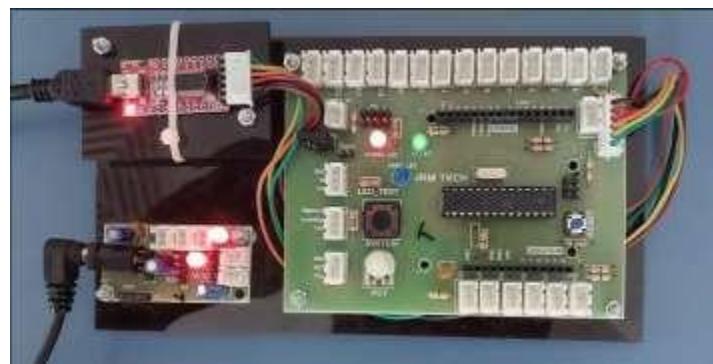
void loop()
{
    digitalWrite(13, HIGH);
    delay(1000); //Wait for 1000 millisecond(s)
    digitalWrite(13,LOW);
    delay(1000); // Wait for 1000 millisecond(s)
}
```

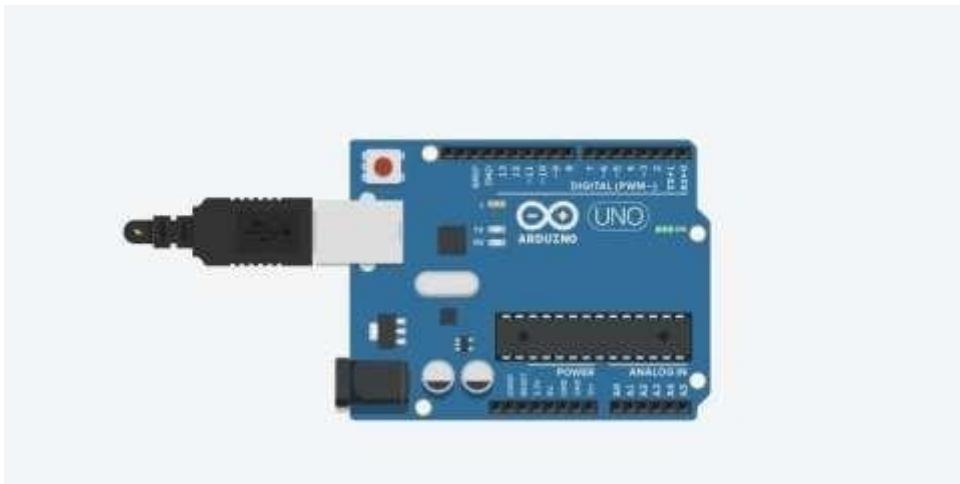
Output:

LED OFF:



LED ON:





B) Blink the user LED in Uno using the port D8.

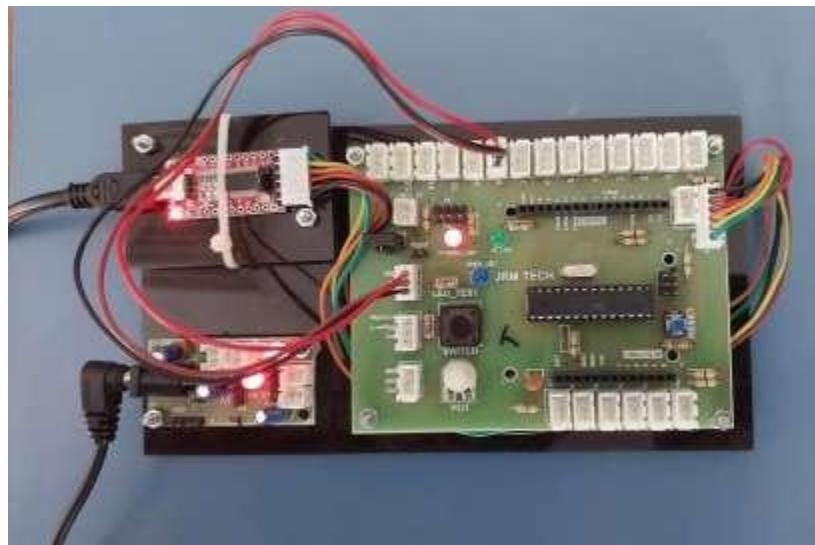
Program:

```
void setup()
{
    pinMode(8, OUTPUT);
}

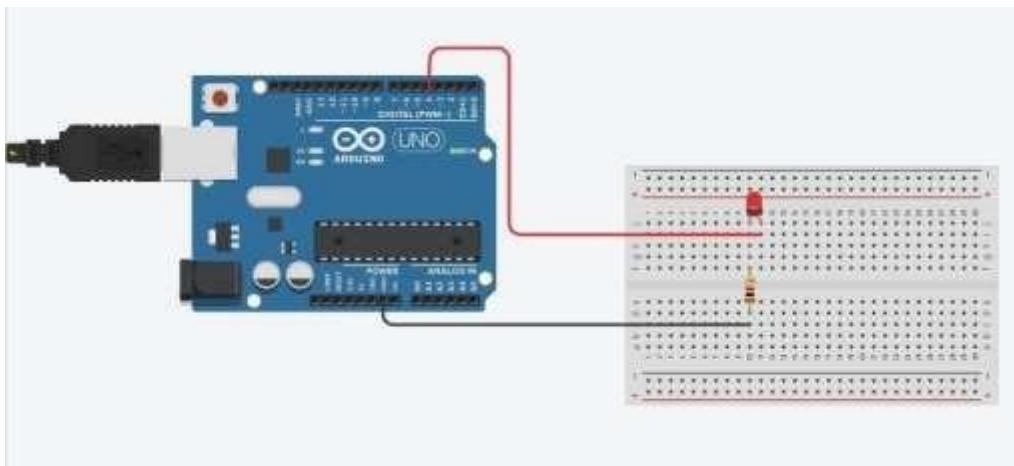
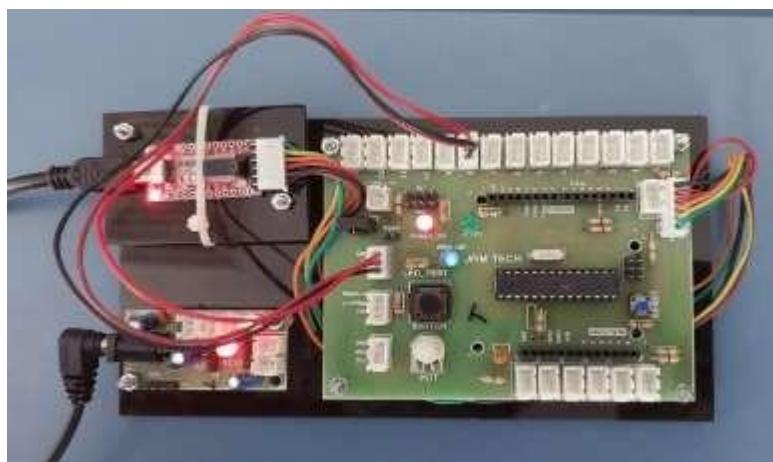
void loop()
{
    digitalWrite(8,HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(8,LOW);
    delay(1000); // Wait for 1000 millisecond(s)
}
```

Output:

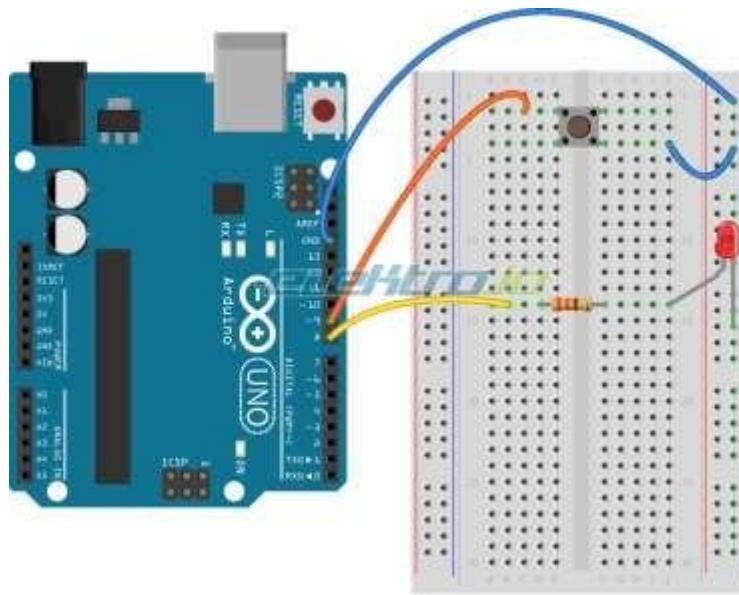
USER LED OFF:



USER LED ON:



C) Switch ON/OFF a LED using a digital Switch-button using External Pullup variable.

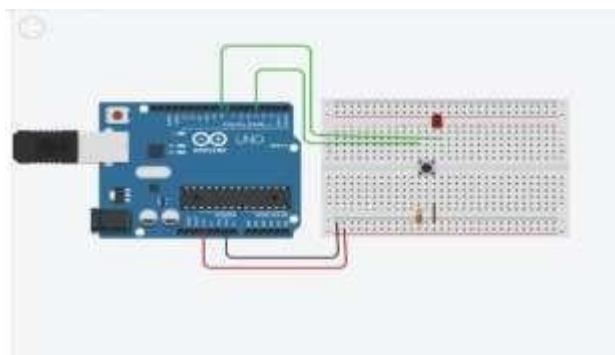


Program:

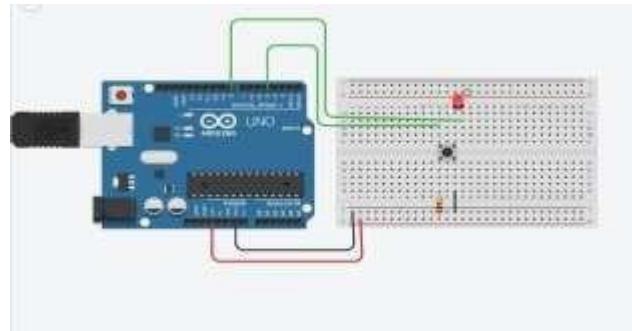
```
int button=0;  
void setup()  
{  
    pinMode(4,INPUT);  
    pinMode(8,OUTPUT);  
}  
void loop()  
{  
    button = digitalRead(4);  
    if(button==LOW)  
        digitalWrite(8,HIGH);  
    else  
        digitalWrite(8,LOW);  
}
```

Output:

Button state: LOW



Button state: HIGH



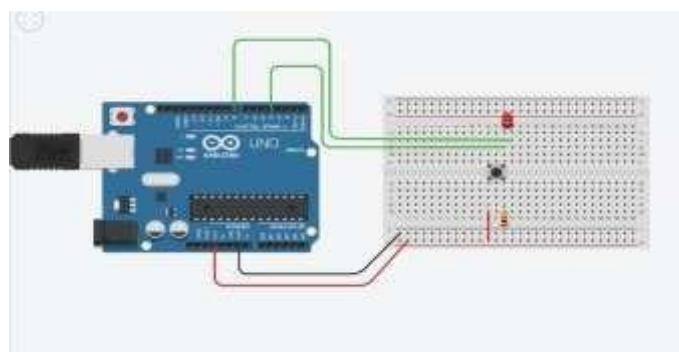
D) Switch ON/OFF a LED using a digital Switch-button using External Pulldown variable.

Program:

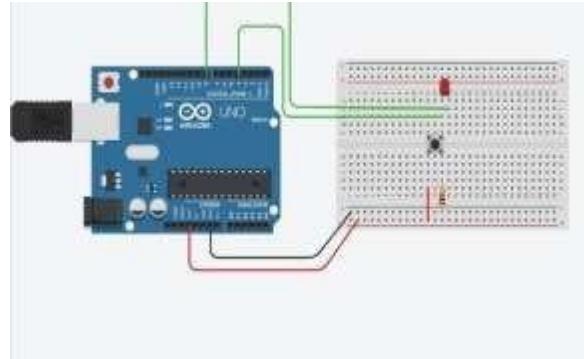
```
int button =0;  
void setup()  
{  
    pinMode(4,INPUT);  
    pinMode(8,OUTPUT);  
}  
void loop()  
{  
    button = digitalRead(4);  
    if(button==HIGH)  
        digitalWrite(8,HIGH);  
    else  
        digitalWrite(8,LOW);  
}
```

Output:

Button state: HIGH



Button state: LOW



E) Switch/Keypad Interface:

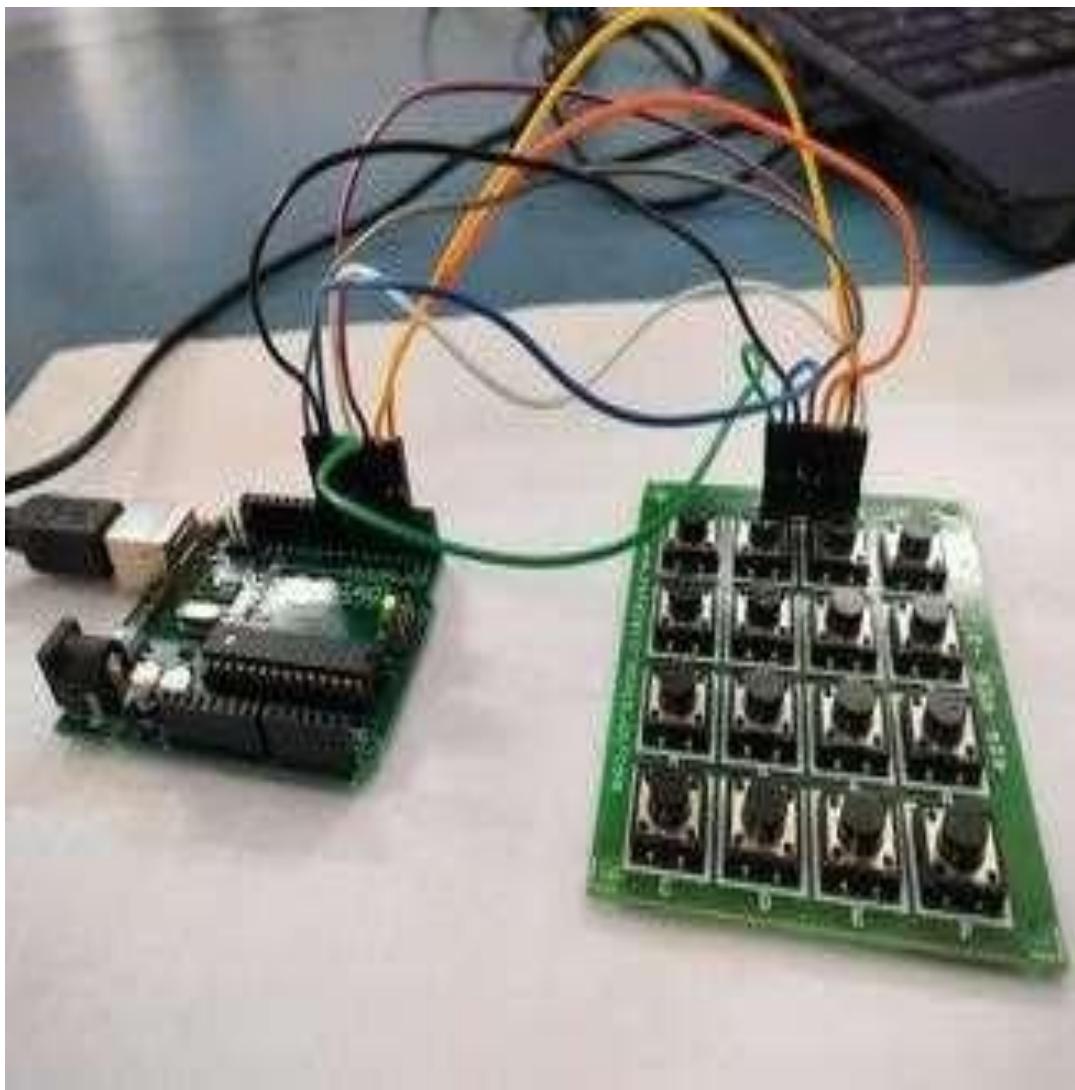
```
const int ROWS=4;
const int COLS=4;// put your setup code here, to run once:
int rowPins[ROWS]={9,8,7,6};
int colPins[COLS]={5,4,3,2};
char keys[ROWS][COLS]={
    {'0','1','2','3'},
    {'4','5','6','7'},
    {'8','9','A','B'},
    {'C','D','E','F'}
};

void setup() {
  Serial.begin (9600);
  for(int i=0;i<ROWS;i++)
  {
    pinMode(rowPins[i],INPUT_PULLUP);
  }
  for(int i=0;i<COLS;i++)
  {
    pinMode(colPins[i],OUTPUT);
    digitalWrite(colPins[i],HIGH);
  }
}

void loop()
{
  for(int col=0;col<COLS;col++)
  {
    digitalWrite(colPins[col],LOW);
    for(int row=0;row<ROWS;row++)
    {
```

```
if(digitalRead(rowPins[row])== LOW)
{
    Serial.println(keys[row][col]);
    delay(300);
}

digitalWrite(colPins[col],HIGH);
}
}
```



```
1  const int ROWS=4;
2  const int COLS=4; // put your setup code here, to run once
3  int rowPins[ROWS]={9,8,7,6};
4  int colPins[COLS]={5,4,3,2};
5  char keys[ROWS][COLS]={
6    {'0','1','2','3'},
7    {'4','5','6','7'},
8    {'8','9','A','B'},
9    {'C','D','E','F'}
10   };
11 void setup()
12 {
13   Serial.begin(9600);
14   for(int i=0;i<ROWS;i++)
15   {
16     pinMode(rowPins[i],INPUT_PULLUP);
17   }
18   for(int i=0;i<COLS;i++)
19   {
20   }
}
Output: Serial Monitor x
Message (Enter to send message to Arduino Uno on COM7)
New Line 9600 baud
11:28:59.121 -> F
11:28:59.122 -> E
11:28:59.123 -> D
11:28:59.124 -> C
11:28:59.125 -> B
11:28:59.126 -> A
11:28:59.127 -> 9
11:28:59.128 -> 8
11:28:59.129 -> 7
11:28:59.130 -> 6
11:28:59.131 -> 5
11:28:59.132 -> 4
11:28:59.133 -> 3
11:28:59.134 -> 2
11:28:59.135 -> 1
11:28:59.136 -> 0
```

Result:

Thus, the blinking of Inbuilt LED, User LED, LED Control using Button Switch and Switch/Keypad Interface is done using Arduino and Verified.

Ex No: 2b

Date: Basic Programming using Arduino - Analog & Digital Sensor Interface

Aim:

To write a program for analog and digital sensor interface using Arduino.

Components Required:

Sl.No	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Temperature Sensor LM35	1
4	LDR Sensor	1
5	Power Adaptor – 12 V	1
6	3pin RMC Cable	2
7	PC with Windows OS -32/64 bit	1

Procedure

1. Open the Arduino environment (IDE)
2. Connect the board to your computer via the USB cable
3. If needed, install the drivers
4. Launch the Arduino IDE
5. Select your board
6. Select your serial port
7. Open the blink example (for checking)
8. Upload the program

A) Temperature Sensor Analog Interface:

Program:

```
const int sensorPin = A0;

void setup() {
    // Start serial communication at 9600 baud rate
    Serial.begin(9600);
}

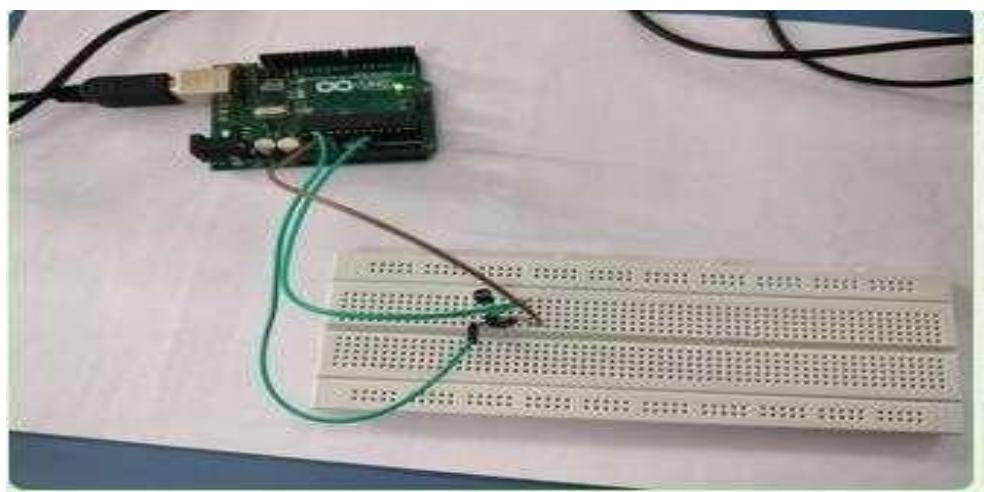
void loop() {
    // Read the analog value from LM35
    int sensorValue = analogRead(sensorPin);

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

    // Convert the voltage to temperature in Celsius
    float temperatureC = voltage * 100;

    // Print the temperature to the Serial Monitor
    Serial.print("Temperature: ");
    Serial.print(temperatureC);
    Serial.println(" °C");

    // Delay for a second before the next reading
    delay(1000);
}
```



The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, Help, and a dropdown for boards (set to Arduino Uno). Below the menu is a toolbar with icons for file operations. The main area displays the code for 'AnalogSensorInterface.ino'. The code reads an analog value from pin A0, converts it to a voltage, and then to a temperature in Celsius, printing the result to the Serial Monitor.

```
1 // Define the analog pin connected to ADS1
2 const int sensorPin = A0;
3
4 void setup() {
5   // Start serial communication at 9600 baud rate
6   Serial.begin(9600);
7 }
8
9 void loop() {
10   // Read the analog value from A0
11   int sensorValue = analogRead(sensorPin),
12
13   // Convert the analog value to voltage
14   float voltage = sensorValue * (5.0 / 1023.0);
15
16   // Convert the voltage to temperature in Celsius
17   float temperatureC = voltage * 100;
18
19   // Print the temperature to the Serial Monitor
20 }
```

The bottom section shows the Serial Monitor window with the title 'Serial Monitor' and the message 'Message (Error to send message to Arduino (no on COM0))'. The monitor displays a series of temperature readings:

```
23:43:50.783 -> Temperature: 21.19 °C
23:43:51.783 -> Temperature: 21.17 °C
23:43:52.784 -> Temperature: 21.18 °C
23:43:53.784 -> Temperature: 21.16 °C
23:43:54.783 -> Temperature: 21.17 °C
23:43:55.783 -> Temperature: 21.17 °C
23:43:56.783 -> Temperature: 21.18 °C
23:43:57.783 -> Temperature: 21.17 °C
23:43:58.783 -> Temperature: 21.18 °C
```

B) LDR Sensor Digital Interface:

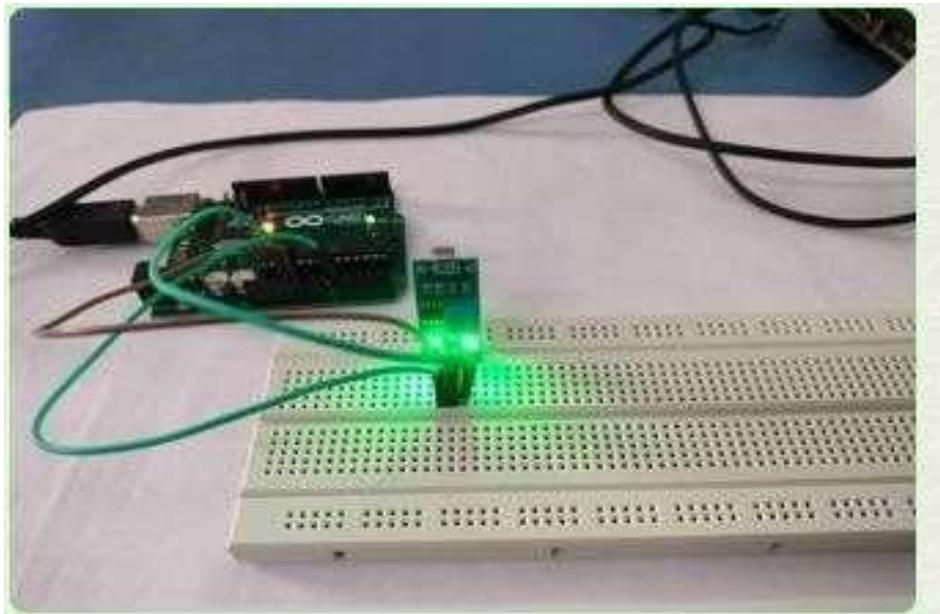
```
const int ldrPin = A0;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);
}

void loop() {
  // Read the analog value from the LDR
  int analogValue = analogRead(ldrPin);

  // Map the analog value (0-1023) to digital (0-1)
  int digitalValue = analogValue > 512 ? 1 : 0;

  // Display the values on the Serial Monitor
  Serial.print("Analog Value: ");
  Serial.print(analogValue);
  Serial.print(" | Digital Value: ");
  Serial.println(digitalValue);
  // Delay for readability
  delay(500);
}
```



The screenshot shows the Arduino IDE interface. The code in the editor is:

```
3
4 void setup() {
5   // initialize serial communication
6   Serial.begin(9600);
7 }
8
9 void loop() {
10  // Read the analog value from the LDR
11  int analogValue = analogRead(A0);
12
13  // Map the analog value (0-1023) to digital (0-1)
14  int digitalValue = analogValue > 512 ? 1 : 0;
15
16  // Display the values on the Serial Monitor
17  Serial.print("Analog Value: ");
18  Serial.print(analogValue);
19  Serial.print(" | Digital Value: ");
20  Serial.println(digitalValue);
21 }
```

The Serial Monitor window shows the following output:

```
[Message from Arduino Uno on COM3]
15:56:44.251 -> Analog Value: 78 | Digital Value: 0
15:56:45.423 -> Analog Value: 78 | Digital Value: 0
15:56:45.526 -> Analog Value: 78 | Digital Value: 0
15:56:45.494 -> Analog Value: 78 | Digital Value: 0
15:56:44.526 -> Analog Value: 78 | Digital Value: 0
15:56:45.457 -> Analog Value: 78 | Digital Value: 0
15:56:45.591 -> Analog Value: 78 | Digital Value: 0
15:56:44.457 -> Analog Value: 1023 | Digital Value: 1
15:56:45.825 -> Analog Value: 1023 | Digital Value: 1
```

Result:

Thus, the program for Analog and Digital Sensor Interface is completed and verified using Arduino.

Ex No: 2c

Date: **Basic Programming using Arduino - Serial Communication**

Aim:

To implement various programs for understanding Serial Communication in Arduino.

Components Required:

Sl.No	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Power Adaptor – 12 V	1
4	3pin RMC Cable	2
5	PC with Windows OS -32/64 bit	1

Task #1 : Displaying a message on the Serial Monitor

SKETCH:

```
void setup()
{
    Serial.begin(9600);
    delay(1000);
}

void loop()
{
    Serial.println("Internet of Things");
}
```

OUTPUT:



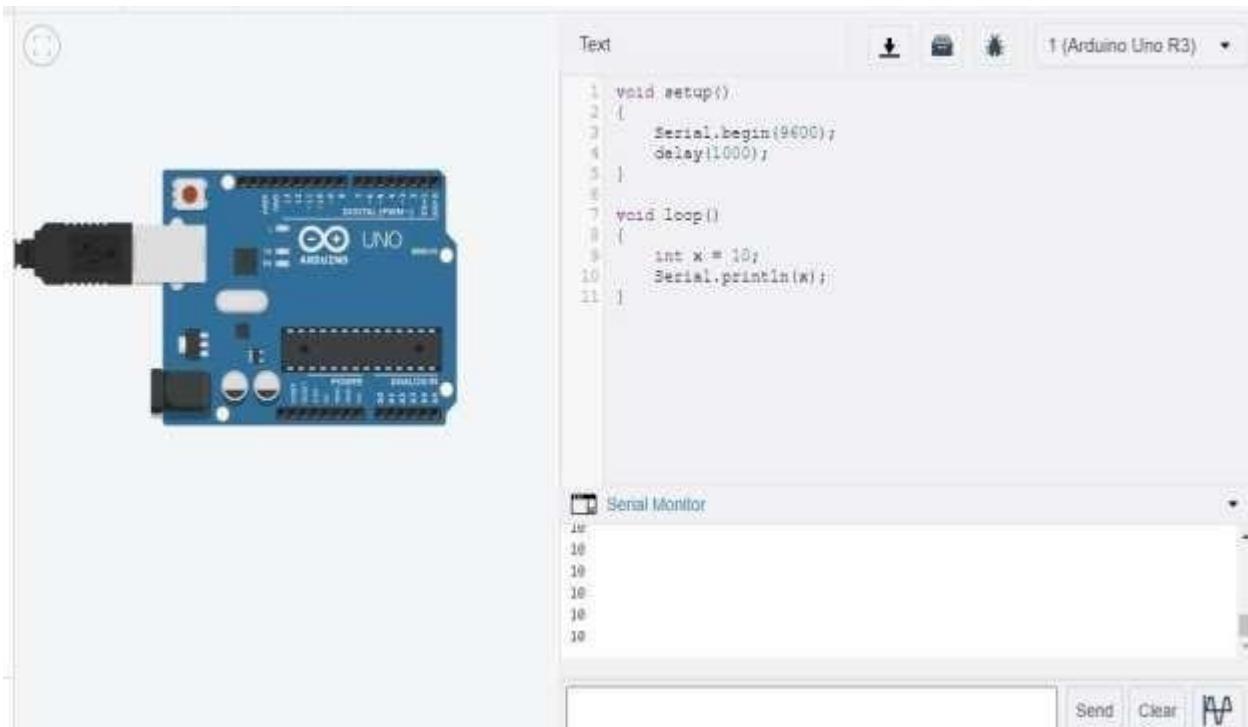
Task #2 : Displaying a stored variable value on the Serial Monitor

SKETCH:

```
void setup()
{
    Serial.begin(9600);
    delay(1000);
}

void loop()
{
    int x=10;
    Serial.println(x);
}
```

OUTPUT:



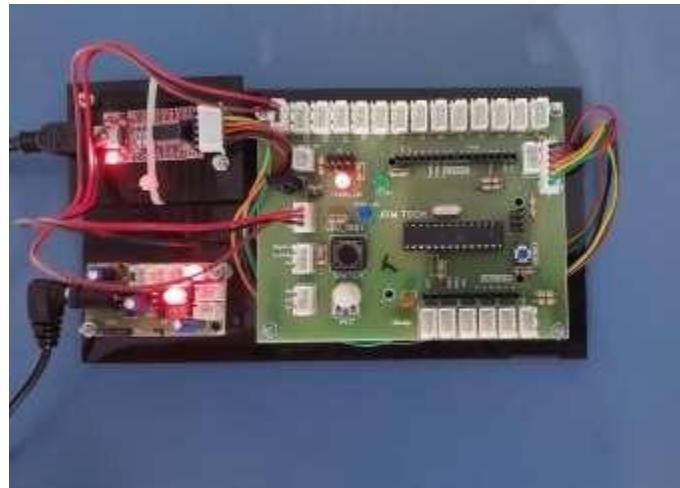
Task #3: Displaying state of a digital input pin 9

SKETCH:

```
void setup()
{
    Serial.begin(9600);
    delay(1000);
}

void loop()
{
    int value= digitalRead(13);
    Serial.prntln(value);
}
```

OUTPUT:



Task 4

- ❖ Interact with Arduino by saying your name through Serial monitor

SKETCH:

```
char input;
void setup()
{
    Serial.begin(9600);
}
void loop()
{
```

```

if(Serial.available()>0)
{
    input=Serial.read();
    Serial.println("Entered Character:");
    Serial.print(input);
}

```

OUTPUT:



Result:

Thus, the various operations on Arduino using Serial Communication is done.

Ex No: 2d

Date: Basic Programming using Arduino - Local display of sensor data using LCD

Aim:

To write and execute Arduino program for interfacing temperature sensors and displaying the measured temperature values on an LCD display.

Components Required:

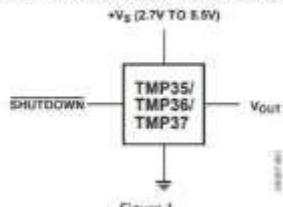
Sl.No	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Power Adaptor – 12 V	1
4	3pin RMC Cable	4
5	PC with Windows OS -32/64 bit	1
6	TMP36 Temperature Sensor	1
7	16x2 LCD Monitor	1
8	LCD Adaptor & Shield	1+1
9	Jumper Cable	Required amount

LM35 Temperature Sensor:

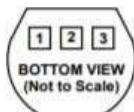
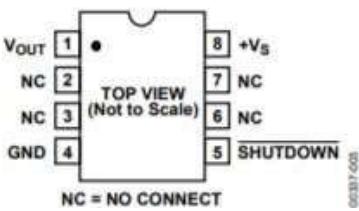
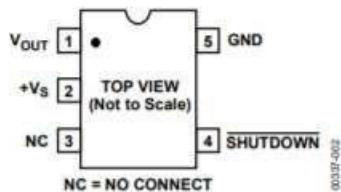
The TMP35/TMP36/TMP37 are low voltage, precision centigrade temperature sensors. They provide a voltage output that is linearly proportional to the Celsius (centigrade) temperature. TMP35/TMP36/TMP37 do not require any external calibration to provide typical accuracies of $\pm 1^{\circ}\text{C}$ at $+25^{\circ}\text{C}$ and $\pm 2^{\circ}\text{C}$ over the -40°C to $+125^{\circ}\text{C}$ temperature range. The low output impedance of the TMP35/TMP36/TMP37 and its linear output and precise calibration simplify interfacing to temperature control circuitry and ADCs. All three devices are intended for single-supply operation from 2.7 V to 5.5 V maximum. The supply current runs well below 50 μA , providing very low self-heating—less than 0.1°C in still air. In addition, a shutdown function is provided to cut the supply current to less than 0.5 μA . The TMP35 is functionally compatible with the LM35/LM45 and provides a 250 mV output at 25°C. The TMP35 reads temperatures from 10°C to 125°C. The TMP36 is specified from -40°C to $+125^{\circ}\text{C}$, provides a 750 mV output

at 25°C, and operates to 125°C from a single 2.7 V supply. The TMP36 is functionally compatible with the LM50. Both the TMP35 and TMP36 have an output scale factor of 10 mV/°C. The TMP37 is intended for applications over the range of 5°C to 100°C and provides an output scale factor of 20 mV/°C. The TMP37 provides a 500 mV output at 25°C. Operation extends to 150°C with reduced accuracy for all devices when operating from a 5 V supply. The TMP35/TMP36/TMP37 are available in low cost 3-lead TO-92, 8-lead SOIC_N, and 5-lead SOT- 23 surface-mount packages.

FUNCTIONAL BLOCK DIAGRAM



PIN CONFIGURATIONS



PIN 1, +V_S; PIN 2, V_{OUT}; PIN 3, GND

00337-004

❖ Other Famous Temperature Sensors in Market

- LM35
- DHT 11/22 by AOSONG
- PT100 Platinum by Generic

16×2 LCD Module:

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications.

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. Usually, each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, **Command and Data**.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The **data register** stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.

The LiquidCrystal library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. Usually, it has 16 pin parallel interface.

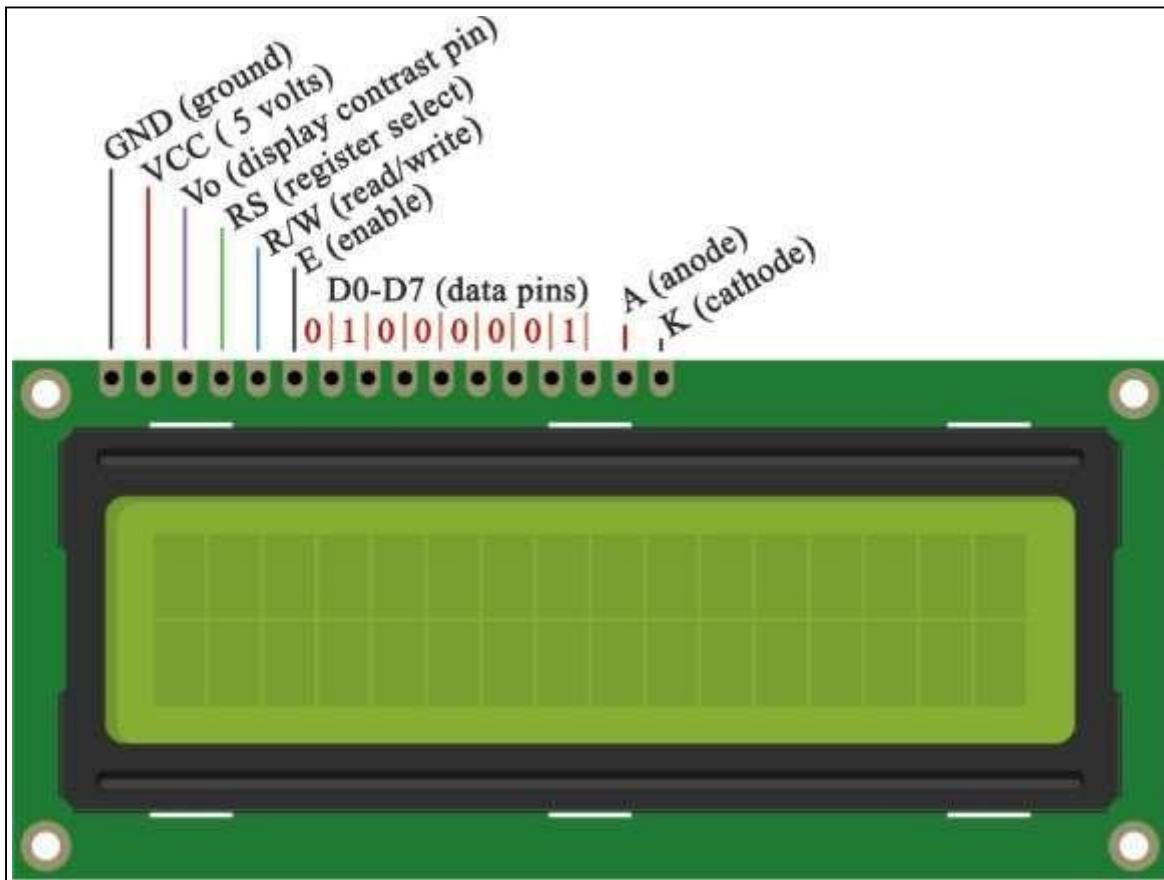


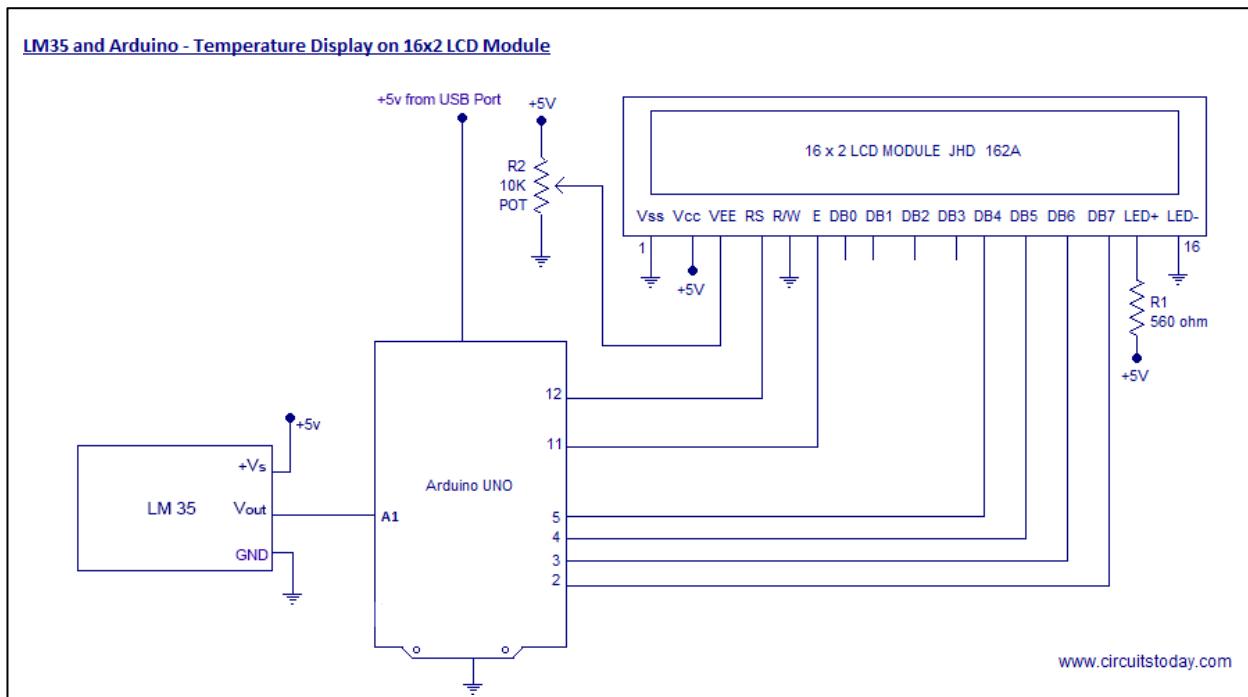
FIG: Pin Configuration

To wire your LCD screen to your board, connect the following pins:

- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2

Major Functions: lcd.begin(), lcd.print(), lcd.clear() and lcd.setCursor()

Temperature Display on 16x2 LCD Module – using TMP36

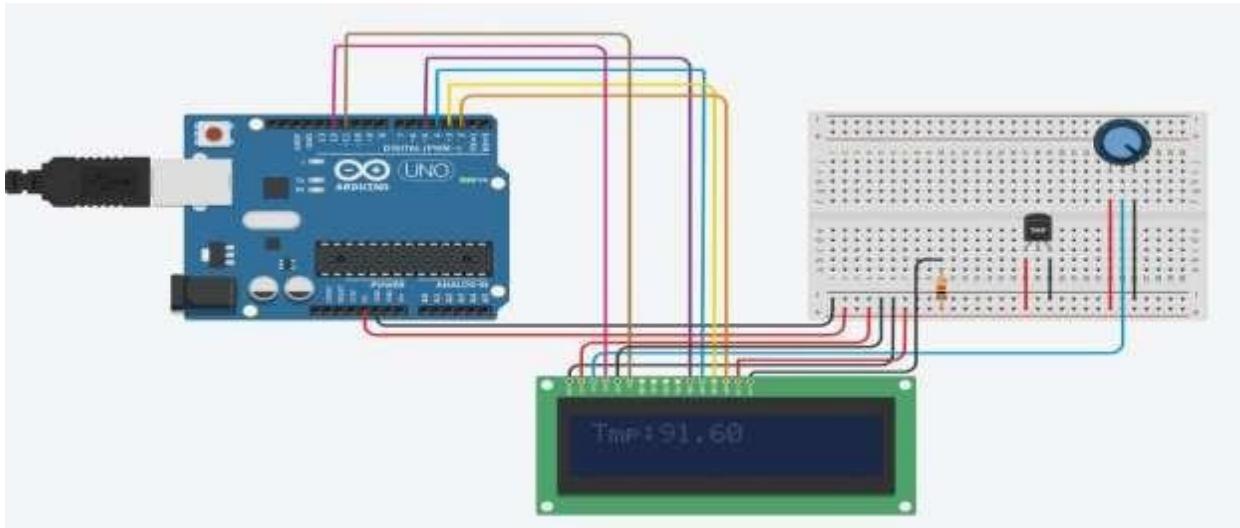


```

#include<LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
float value;
int tmp = A1;
void setup(){
    pinMode(tmp,INPUT);
}
void loop(){
    value = analogRead(tmp)*0.004882814;
    value = (value - 0.5) * 100.0;
    lcd.setCursor(0,1);
    lcd.print("Tmp:");
    lcd.print(value);
    delay(1000);
    lcd.clear();
}

```

Output



Result:

Thus, the Arduino program was successfully executed, and the temperature values were displayed on the LCD.

Ex No: 2e

Basic Programming using Arduino - Display of Sensor values in Mobile handset using Bluetooth

Aim:

To design and deploy an IoT-based buzzer (LED) control system using the Blue Bot platform, a Bluetooth module, and an Arduino microcontroller, demonstrating real-time remote control of hardware components via a mobile application.

Components Required:

- Arduino Board
- HC-05 Bluetooth
- Module Buzzer or
- LED Connecting
- Wires
- Smartphone with Blue Bot App Installed
- Laptop with Arduino IDE

Specifications

The "**Display of Sensor Values in Mobile Handset Using Bluetooth**" system consists of a microcontroller unit (MCU) such as ESP32, ESP8266, or Arduino with a Bluetooth module like HC-05 (Bluetooth 2.0), HC-06, HM-10 (BLE 4.0), or ESP32's built-in Bluetooth. It supports various sensors, including temperature and humidity sensors (DHT11/DHT22), gas sensors (MQ2, MQ135), light sensors (LDR), and motion sensors (PIR), operating at 3.3V or 5V DC from a regulated power supply or battery. Bluetooth communication is based on serial (UART) at a default baud rate of 9600 bps, with a range of approximately 10 meters for HC-05/HC-06 and up to 30 meters for BLE modules. The system interfaces with a mobile handset running Android or iOS, using either a custom app (developed with MIT App Inventor, Android Studio, or Flutter) or generic Bluetooth terminal apps. The mobile application displays real-time sensor data, provides graphical representations, enables notifications for threshold breaches, and optionally logs data for analysis. Power consumption varies, with HC-05/HC-06 consuming around 8-30 mA, while BLE modules like HM-10 and ESP32 BLE offer low-power modes (1-3 mA in sleep mode). This system is ideal for IoT applications requiring wireless sensor monitoring.

Background Theory

The Internet of Things (IoT) is a technology that connects physical devices to the internet or local networks, enabling real-time monitoring and control. One of the key aspects of IoT is remote device automation, where sensors and actuators interact using wireless communication protocols. Bluetooth-based IoT applications, such as those deployed using the Blue Bot platform, facilitate short-range, low-power communication between a microcontroller (Arduino) and a smartphone app. The HC-05 Bluetooth module acts as a bridge, transmitting commands from the smartphone to the Arduino via serial communication (UART protocol). When the user presses a button in the Blue Bot app, the command is sent to the Arduino, which processes it and controls the connected buzzer (LED) accordingly. This setup demonstrates the integration of wireless communication and embedded systems, providing a fundamental approach to IoT-based device automation.

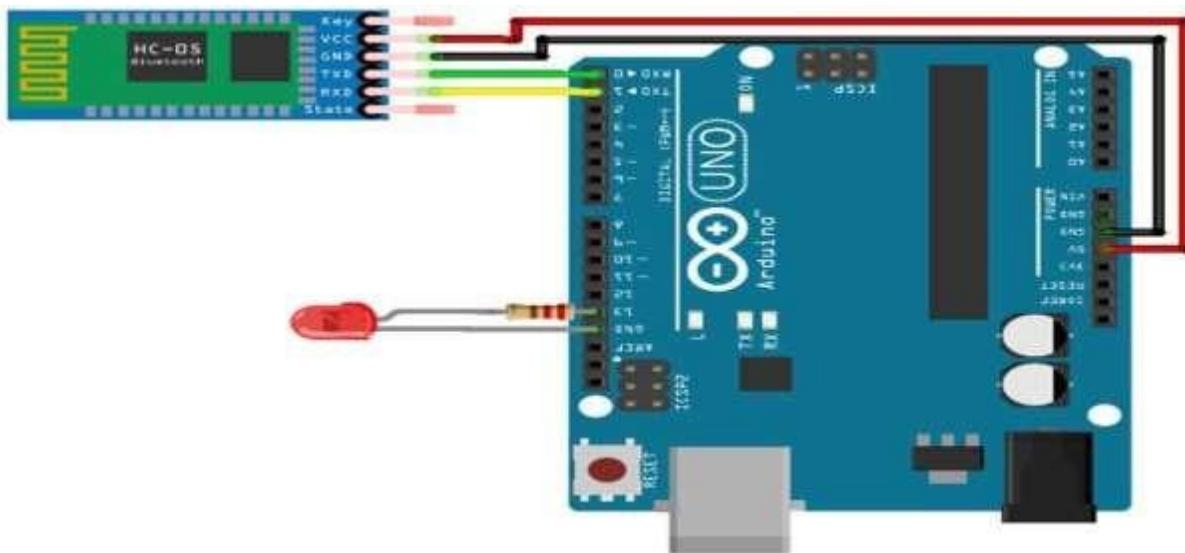
Procedure

Step 1: Hardware Setup

1. Connect the Bluetooth Module (HC-05) to the Arduino:
 - o VCC → 5V
 - o GND → GND
 - o TX (HC-05) → RX (Arduino, Pin 10 for SoftwareSerial)
 - o RX (HC-05) → TX (Arduino, Pin 11 for SoftwareSerial)
 - o
2. Connect the Buzzer (LED) to the Arduino:
 - o Buzzer (LED) Positive (+) → Digital Pin 13
 - o Buzzer (LED) Negative (-) → GND

Step 2: Program the Arduino

Open Arduino IDE.



Upload the following code to control the buzzer (LED) via Bluetooth commands:Program:

```

void setup() {
    Serial.begin(9600);          // Initialize serial communication at 9600 baud
    pinMode(12, OUTPUT);        // Set the built-in LED pin as an output
}

void loop() {
    if (Serial.available() > 0) { // Check if data is available to read from Bluetooth
        char data = Serial.read(); // Read the data received from the Bluetooth module

        switch (data) {
            case '1':           // If '1' is received, turn on the LED
                digitalWrite(12, HIGH);
                break;
            case '2':           // If '2' is received, turn off the LED
                digitalWrite(12, LOW);
                break;
            default:
                break;
        }

        Serial.println(data);     // Print the received data to the Serial Monitor
    }

    delay(50); // Small delay to avoid overwhelming the microcontroller
}

```

Result:

Thus, the buzzer (LED) was controlled wirelessly via the Bluetooth module (HC-05) with Arduino is done successfully.

Exp No: 3

Date:

Study of ESP8266 12E Node MCU

Aim:

To study about NodeMCU Development Kit v1.0.

Components Required:

Sl.No	Components Name	Quantity
1	NodeMCU Development Kit v1.0	1
2	USB - Micro B Type	1
3	Arudino IDE	With NodeMCU packages Installed mode

ESP8266 WiFi SoC:

The ESP8266 is a low-cost **Wi-Fi microchip (esp 12x)** with full TCP/IP stack and **microcontroller** capability developed by Ai-thinker Team, produced by Shanghai-based Chinese manufacturer Espressif Systems for IoT platform. (**ie., wifi enabled μ C**)

The successor of ESP8266 is ESP32.

❖ Major Features:

- **Processor:** L106 32-bit RISC microprocessor core based on the Cadence Tensilica Xtensa Diamond Standard, 80MHz(default) or 160 MHz.
- **IEEE 802.11 b/g/n Wi-Fi (2.4 GHz)**
 - ✓ Integrated TR switch, balun, LNA, power amplifier and matching network
 - ✓ WEP or WPA/WPA2 authentication, or open networks
- Integrated TCP/IP protocol stack
- **Memory:** 32 KiB instruction, 80 KiB user data
- Integrated 10-bit ADC
- PWM (10 bit)

- Operating Voltage: 3.0 ~3.6V (Level: 3.3 V); accepts 5V.
- 16 GPIO pins
- SPI (Serial Peripheral Interface)
- I²C (software implementation)
- I²S interfaces with DMA (sharing pins with GPIO)
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2

ESP32 or ESP8266 are cheap Wi-Fi enabled SoC modules perfectly suited for DIY projects in the Internet of Things (IoT) field. The best part is that they come with wireless networking included, which makes them apart from other microcontrollers like the Arduino. This means that you can easily control and monitor devices remotely via Wi-Fi for a very low price.

Differences between ESP32 and ESP8266: <https://makeradvisor.com/esp32-vs-esp8266/>

ESP32: <https://openhomeautomation.net/getting-started-esp32>

💡 Node MCU:

NodeMCU is an open source **Lua** based **firmware** for the **ESP8266 WiFi SOC** from Espressif and uses an on-module flash-based SPIFFS file system. By exploring functionality with ESP8266 chip, NodeMCU firmware comes with ESP8266 Development board/kit i.e. NodeMCU Development board.

NodeMCU => LUA based firmware

NodeMCU development board:

NodeMCU development board = Development Board + ESP8266 (μc + wifi: ESP12x)
+ **NodeNCU**



A development kit based on ESP-12E version which has been released by the same NodeMCU guys. Also known as **NodeMCU 1.0 / NodeMCU devkit 1.0**, which usually comes in black colored PCB.

Fig: NodeMCU Development Board/kit v1.0 (Version2)

Since NodeMCU is open source platform, their hardware design is open for it/modify/build. There is Version1 (V0.9) available for NodeMCU Dev Kit which usually comes in blue colored PCB.

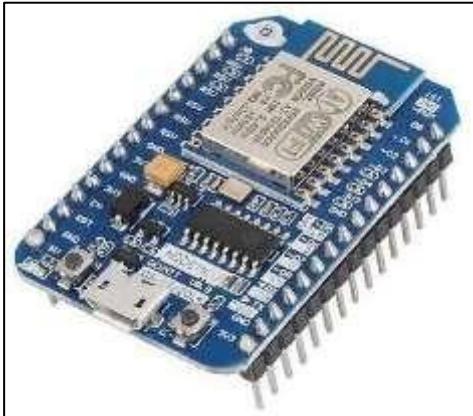


Fig: NodeMCU Development Board/kit v0.9 (Version1)

NodeMCU Dev Kit has Arduino like **Analog (i.e. A0) and Digital (D0-D10) pins** on its board. Also, supports serial communication protocols i.e. UART, SPI, I2C etc. Using such serial protocols, we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc.

Reference 1: <https://blog.squix.org/2015/03/esp8266-module-comparison-esp-01-esp-05.html>

Reference 2: <https://www.espressif.com/en>

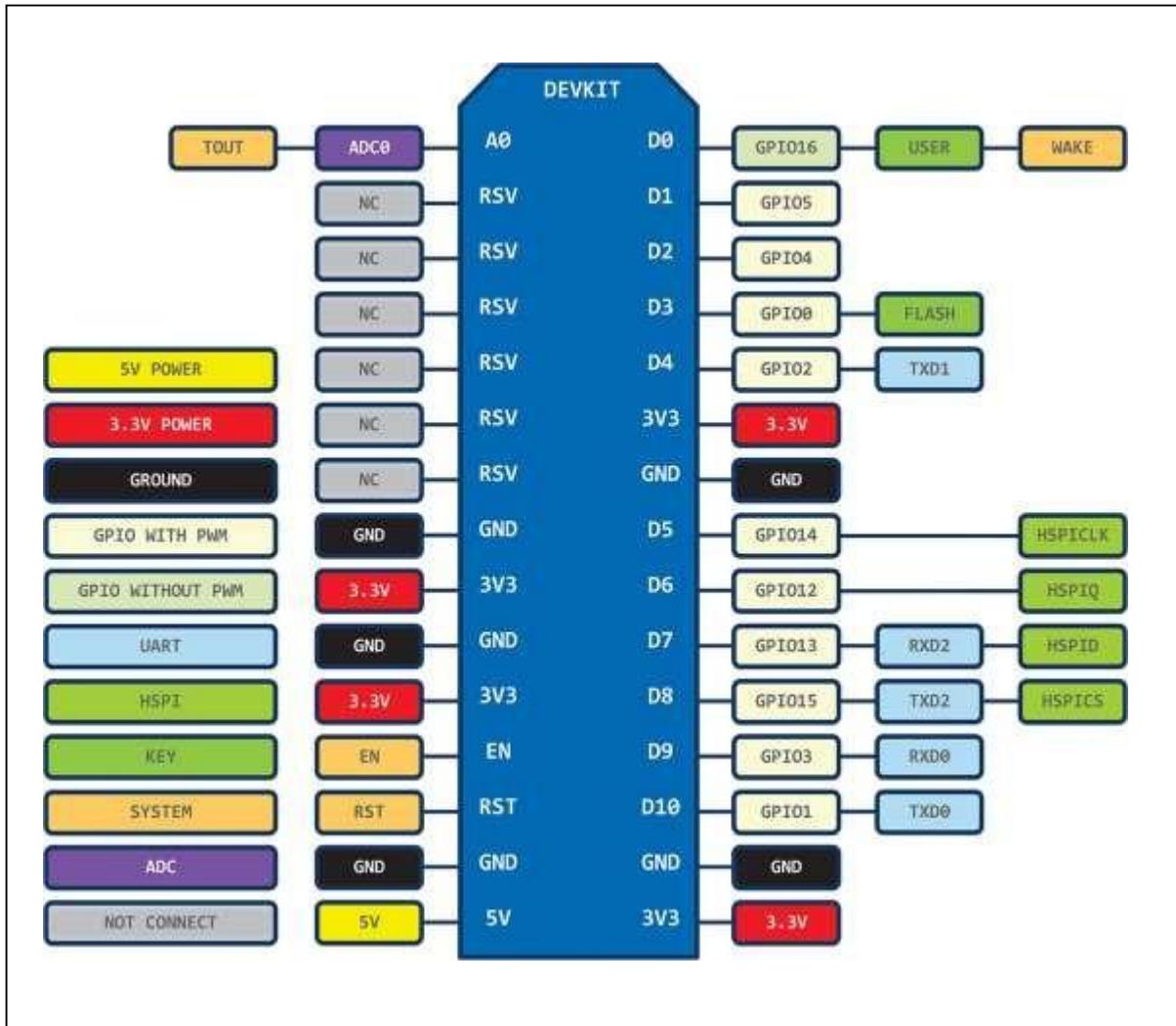
Reference 3: <https://www.ahirlabs.com/2017/10/21/what-is-nodemcu-esp8266/>

Reference 4: <https://www.electronicwings.com/nodemcu/nodemcu-development-kitboard>

Reference 4: <https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/>

❖ Pin Configuration

<http://www.electronicwings.com/nodemcu/nodemcu-development-kitboard>



SDKs / IDEs:

Major SDKs (mostly open-source) include:

- [NodeMCU](https://en.wikipedia.org/wiki/NodeMCU) – A Lua-based firmware. Lua is an open source, lightweight, embeddable scripting language built on top of C programming language.
<https://en.wikipedia.org/wiki/NodeMCU>
 - [ESPplorer IDE](#) – for Lua Scripts
 - [Arduino](#) – A C++-based firmware (IDE is implemented in JAVA). With this core, the ESP8266 CPU and its Wi-Fi components can be programmed like any other Arduino

device. *This makes easy for Arduino developers than learning new language and IDE for NodeMCU.* The ESP8266 Arduino Core is available through GitHub.

❖ Difference in using ESPlorer and Arduino IDE

<http://www.electronicwings.com/nodemcu/introduction-to-nodemcu>

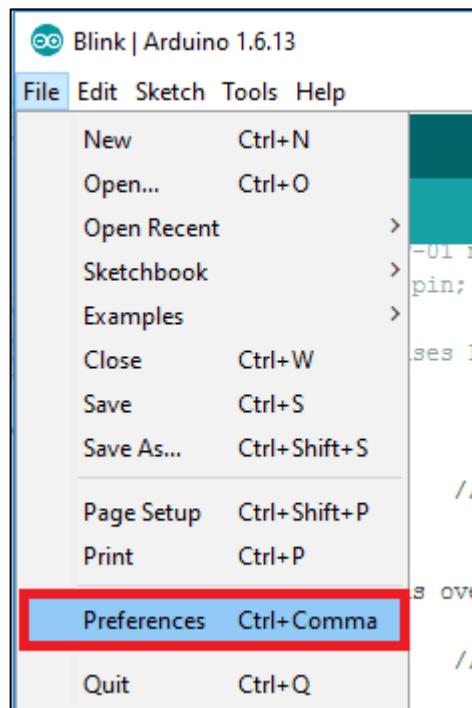
How to start with NodeMCU?

To get start with using NodeMCU for IoT applications first we need to know about how to write/download NodeMCU firmware in NodeMCU Development Boards.

<http://www.electronicwings.com/nodemcu/getting-started-with-nodemcu>

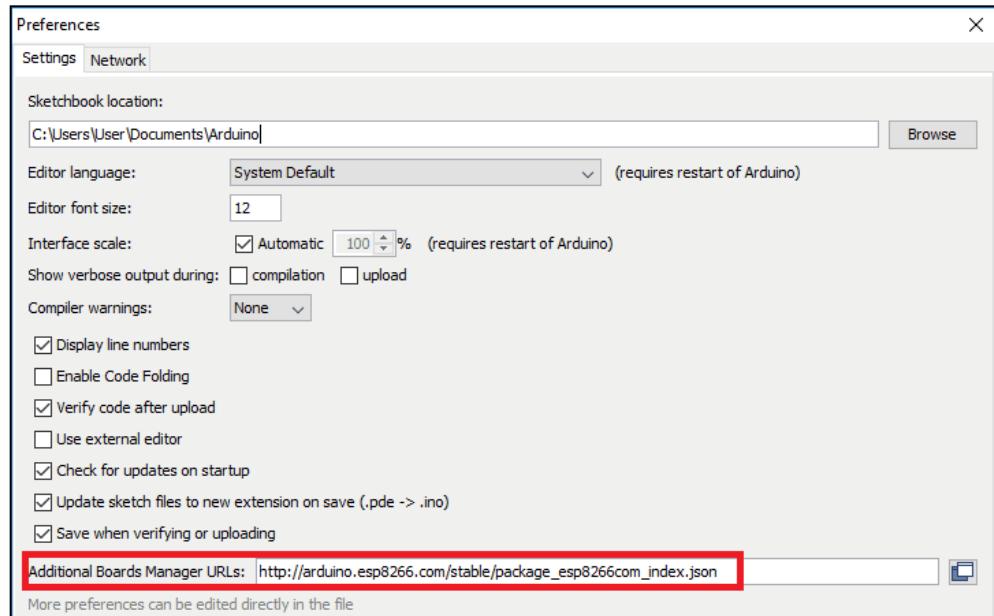
NodeMCU using Arduino IDE (**Note: One of the ways**)

- Open Arduino IDE and Go to File -> Preference

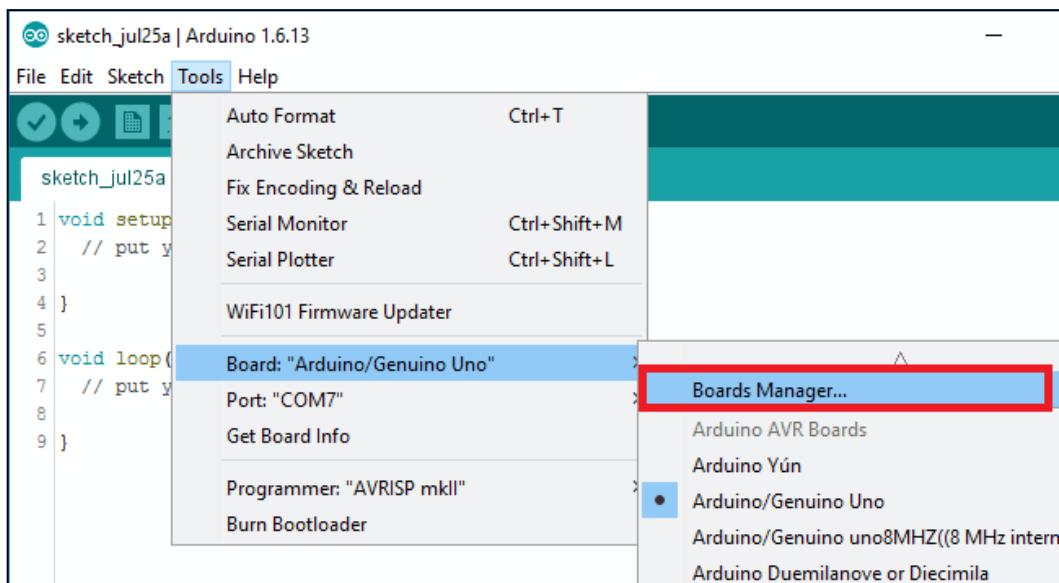


- Now on Preference window, Paste below link in Additional Boards Manager URLs

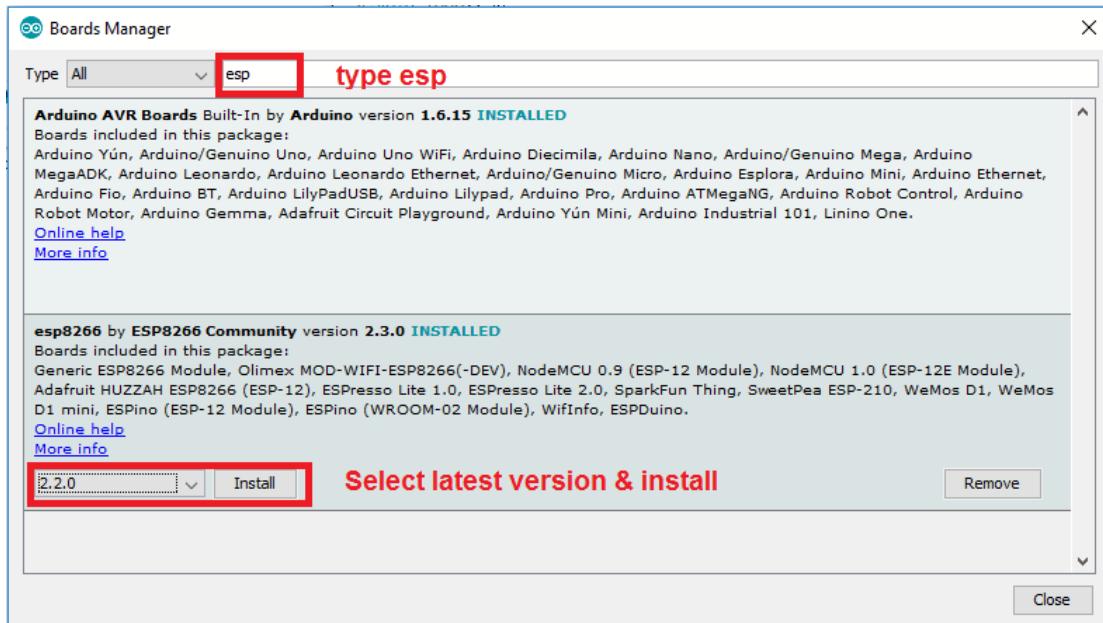
http://arduino.esp8266.com/stable/package_esp8266com_index.json



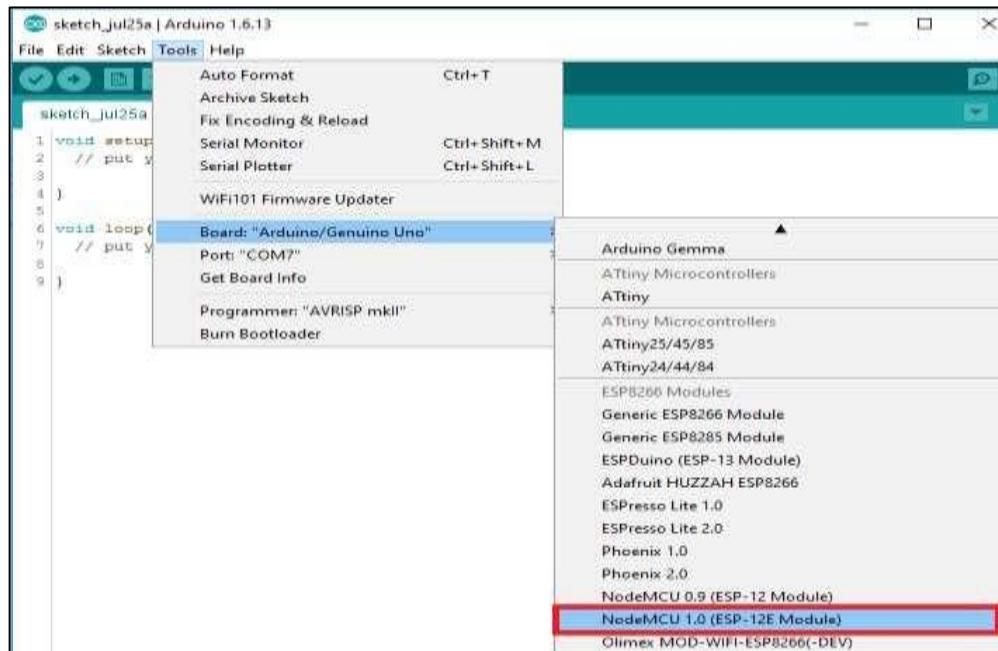
- Now close Preference window and go to Tools -> Board -> Boards Manager



- In Boards Manager window, Type “esp” in the search box, esp8266 will be listed there below. Now select latest version of board and click on install.



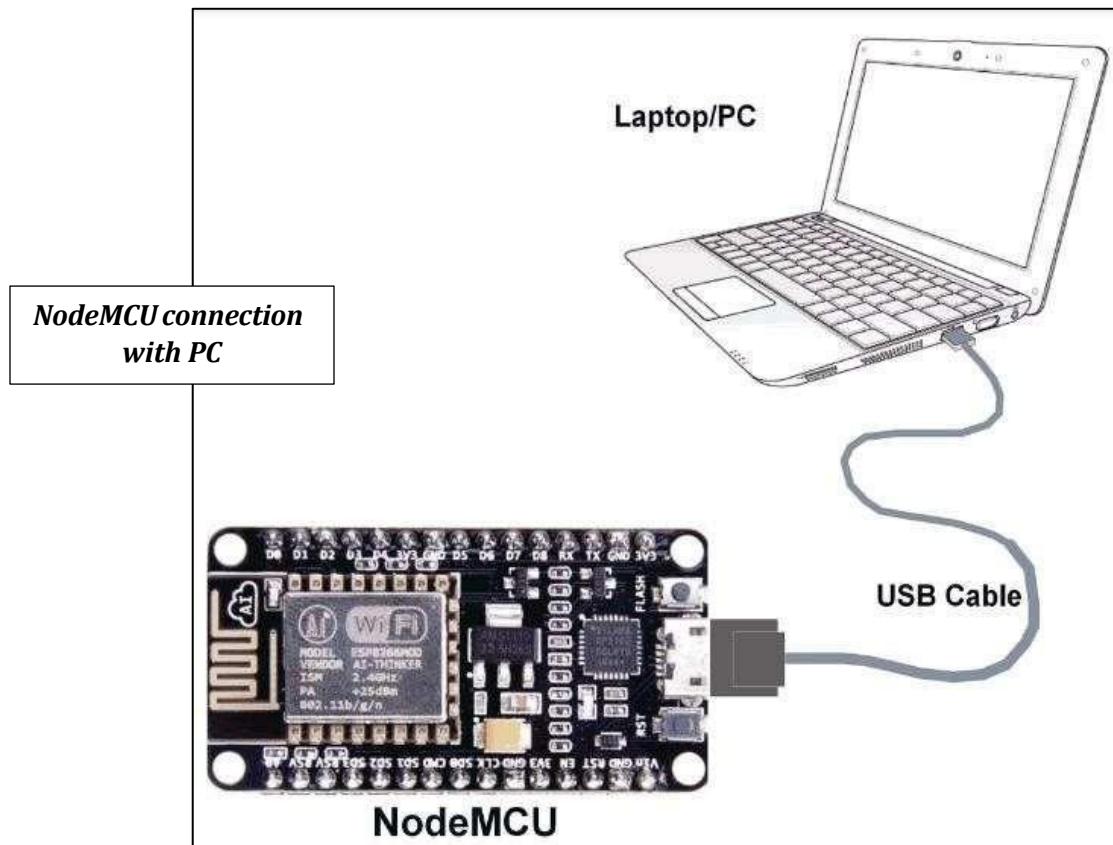
- After installation of the board is complete, open Tools->Board->and select NodeMCU 1.0(ESP-12E Module).



- Now Your Arduino IDE is ready for NodeMCU

❖ Example

Let's see how to write simple serial print sketch using Arduino IDE for NodeMCU. First connect NodeMCU Development Kit with PC as shown in below figure.

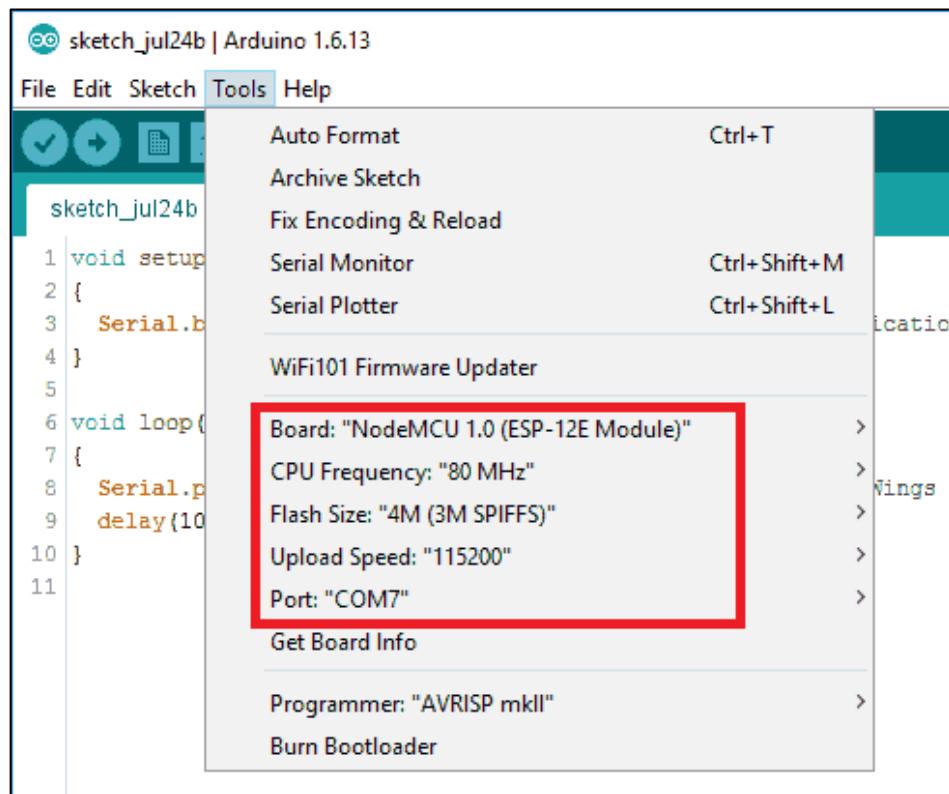


- After setting up Arduino IDE for NodeMCU, open Arduino IDE and write simple sketch of serial print as shown in below figure.

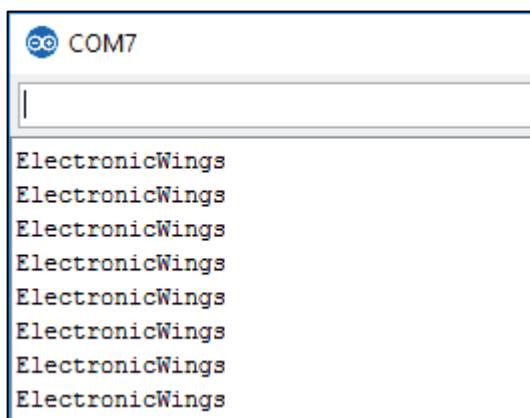
A screenshot of the Arduino IDE interface. The title bar says "sketch_jul24b | Arduino 1.0.15". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for file operations. The main area shows a code editor with the following text:

```
1 void setup()
2 {
3     Serial.begin(9600);          //initialise serial communication
4 }
5
6 void loop()
7 {
8     Serial.println("ElectronicWings"); //print Electronic Wings at new line per second
9     delay(1000);
10}
11
```

- Ensure that you have selected the correct board as shown in below figure. Also make sure that you have selected the appropriate COM port.



- Now compile & upload the written sketch directly to the NodeMCU Dev Kit by clicking on upload button.
- Serial monitor output window will pop up with output as shown in below figure.



NOTES

- ✓ To use NodeMCU V1 or V2 or V3 dev boards using Arduino IDE, we do not need to flash it with firmware using nodemcu flasher. It is required only if we intend to program NodeMCU using Lua script with esplorer etc.
- ✓ The ESP8266 chip requires 3.3V (working voltage) power supply voltage. It should not be powered with 5 volts like other arduino boards.
- ✓ NodeMCU ESP-12E dev board can be connected to 5V (accepting voltage) using **micro USB** connector or **Vin** pin available on board.
- ✓ The I/O pins of ESP8266 communicate or input/output max 3.3V only. i.e. the pins are **NOT** 5V tolerant inputs.
- ✓ In case you have to interface with 5V I/O pins, you need to use level conversion system (either built yourself using resistor voltage divider or using ready to use level converters e.g. these ones [adafruit](#) or [aliexpress](#) etc.).
- ✓ The pin mapping of NodeMCU dev board are different from those of ESP8266 GPIOs. (ie., like Raspberry Pi)

Result:

Thus, the study of NodeMCU Development kit is done.

Exp No: 4a

Date:

**Basic Programming using NodeMCU - Remote control of Electrical appliances
using Mobile handset and Wi-Fi**

Aim:

To write and execute a program using NodeMCU for the remote control of electrical appliances through a mobile handset over Wi-Fi.

Components Required:

Sl.No	Components Name	Quantity
1	NodeMCU Development Kit v1.0	1
2	USB - Micro B Type	1
3	Relay,LED	-
4	Arudino IDE	With NodeMCU packages Installed mode

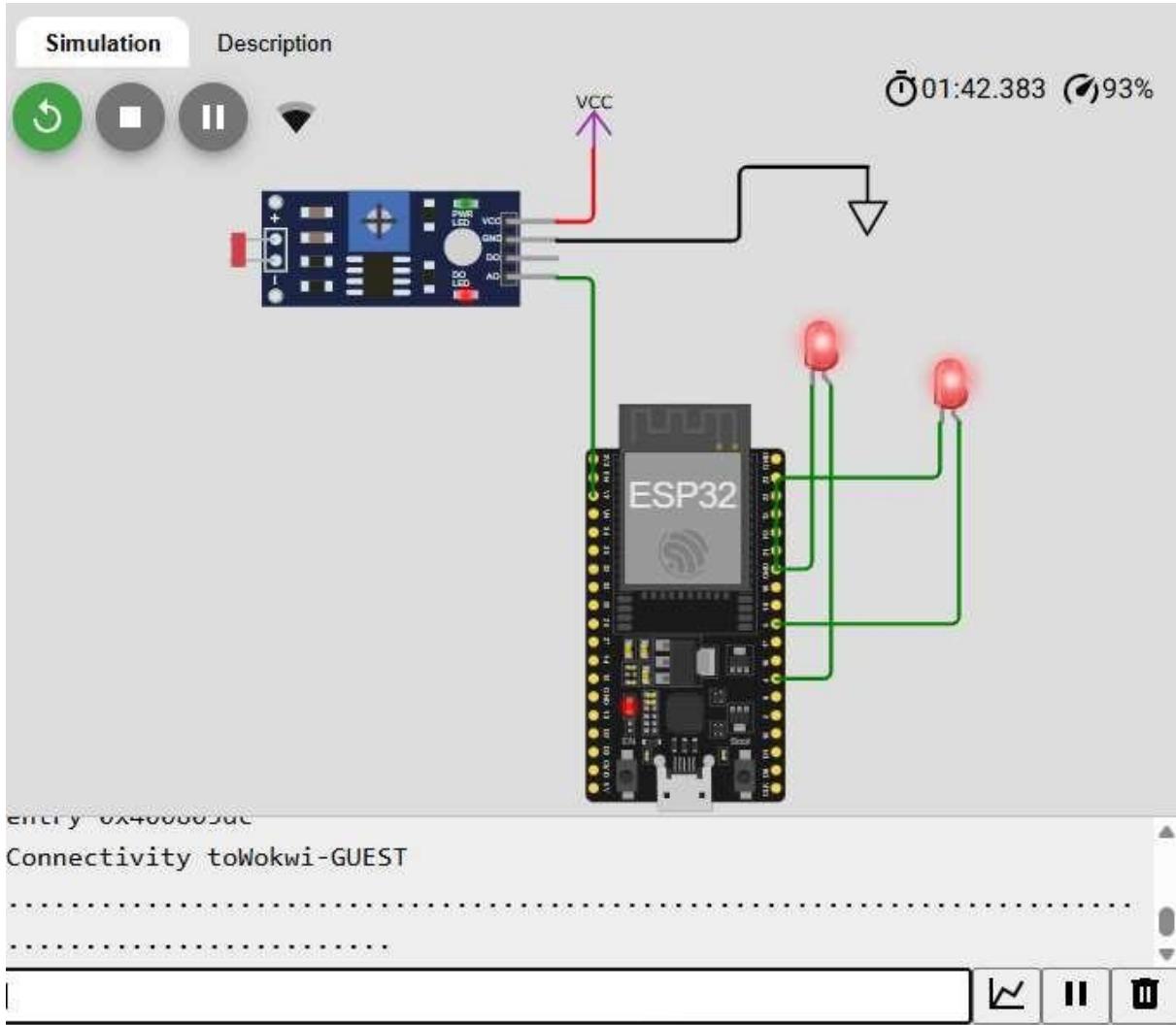
ESP8266 WiFi SoC:

The ESP8266 is a low-cost **Wi-Fi microchip (esp 12x)** with full TCP/IP stack and **microcontroller** capability developed by Ai-thinker Team, produced by Shanghai-based Chinese manufacturer Espressif Systems for IoT platform. (**ie., wifi enabled μC**)

ESP8266-12E Wi-Fi Interfacing with Electrical Appliances

To control electrical appliances, the ESP8266-12E can be used with:

1. **Relays** (for switching ON/OFF AC appliances)
2. **Triacs and Optocouplers** (for AC dimming control)
3. **MOSFETs or Transistors** (for DC motor/fan control)
4. **Smart Home Applications** (IoT-based automation with MQTT, Blynk, etc.)



PROGRAM: Node mcu connectivity through Thinkspeak

```

#include "ThingSpeak.h" // Header File
#include <WiFi.h>
#include <WiFiClient.h>
//enter wifi credentials
char ssid[] = "Wokwi-GUEST"; // enter network name
char pass[] = ""; // enter wifi password

//ThingSpeak credentials
unsigned long myChannelNumber = 2424075;// Channel ID From ThingSpeak
const char * myWriteAPIKey = "Y61RJEN1KGHQBRBH";// Key from ThingSpeak
int field=1;
int a;
WiFiClient client;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

```

```

delay(10);
Serial.print("Connectivity to");
Serial.println(ssid);
pinMode(4, OUTPUT); // onboard led Arduino
//pinMode(5, OUTPUT); // control through blynk cloud
WiFi.begin(ssid,pass);
while(WiFi.status() !=WL_CONNECTED)
{
  delay(500);
  Serial.print(".");
  digitalWrite(4,HIGH);
  delay(500);
  digitalWrite(4,LOW);
}
digitalWrite(4,HIGH);
Serial.println("WiFi connected to");
Serial.println(ssid);
Serial.println("IP address:");
Serial.println(WiFi.localIP());
ThingSpeak.begin(client); // Initialize ThingSpeak

}

void loop() {
// a==analogRead(A0);
a++;
int x = ThingSpeak.writeField(myChannelNumber, field, a, myWriteAPIKey);
if(x == 200){
  Serial.println("Channel update successful.");
  Serial.println(a);
  delay(15000); // Minimum delay 15 sec
}
else{
  Serial.println("Problem updating channel. HTTP error code " + String(x));
}
}

```

Result:

Thus, the design and implementation of Wi-Fi interfacing to control electrical appliances using NodeMCU, with output display on the Serial Monitor, was successfully completed.

Exp No: 4b

Date:

Basic Programming using NodeMCU - Local Web server using NodeMCU and displaying Sensor values.

Aim:

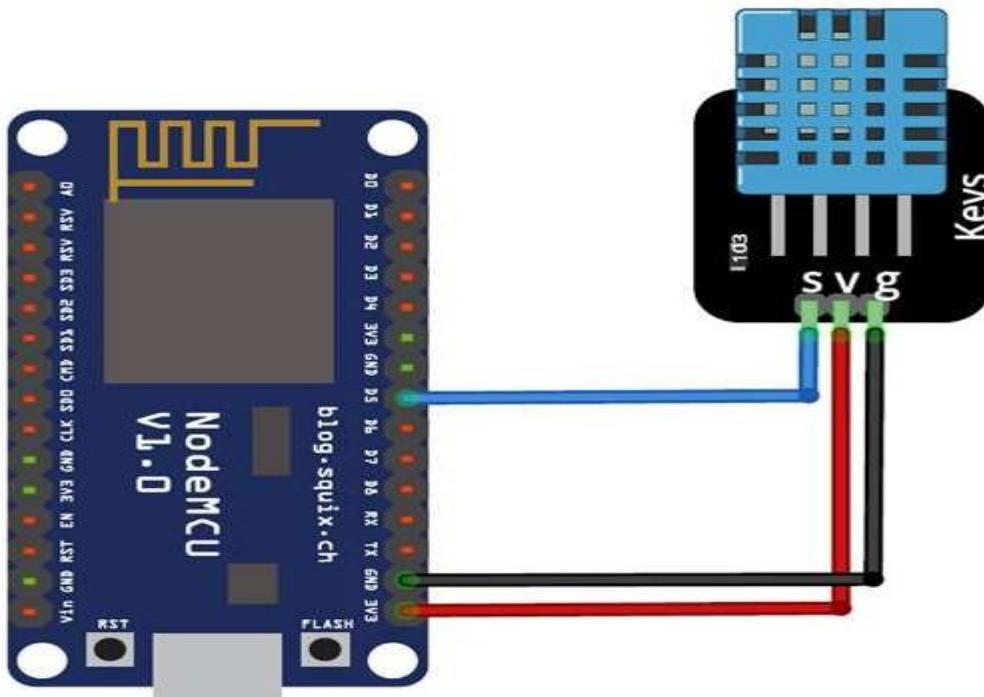
To design a local web server using NODEMCU to display sensor values.

Components Required:

- NodeMCU ESP8266
- DHT11/DHT22 (Temperature & Humidity Sensor) or any other sensor
- Jumper wires
- Breadboard
- USB cable (for programming NodeMCU)
- Computer with Arduino IDE installed

Theory

NodeMCU (ESP8266) is a microcontroller with built-in Wi-Fi that allows setting up a local web server. By interfacing sensors with NodeMCU, sensor data can be read and displayed on a web page hosted on the device.



Program

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <DHT.h>

#define DHTPIN 2
#define DHTTYPE DHT11 // Change to DHT22 if using DHT22
DHT dht(DHTPIN, DHTTYPE);

const char* ssid = "Your_WiFi_SSID";
const char* password = "Your_WiFi_Password";
ESP8266WebServer server(80);

void handleRoot() {
    float temp = dht.readTemperature();
    float hum = dht.readHumidity();
    String html = "<html><head><title>Sensor Data</title></head><body>";
    html += "<h1>Temperature: " + String(temp) + "°C</h1>";
    html += "<h1>Humidity: " + String(hum) + "%</h1>";
    html += "</body></html>";
    server.send(200, "text/html", html);
}

void setup() {
```

```

Serial.begin(115200);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {

    delay(1000);

    Serial.print(".");

}

Serial.println("Connected to WiFi");

server.on("/", handleRoot);

server.begin();

}

void loop() {

    server.handleClient();

}

```

Result:-

Thus, the sensor values were successfully displayed on a local web page hosted by the NodeMCU.

Ex No: 5

Date: **Study and Configuration of Raspberry PI**

Aim:

To study about the configurations of Raspberry Pi and get working experience on it.

Components Required:

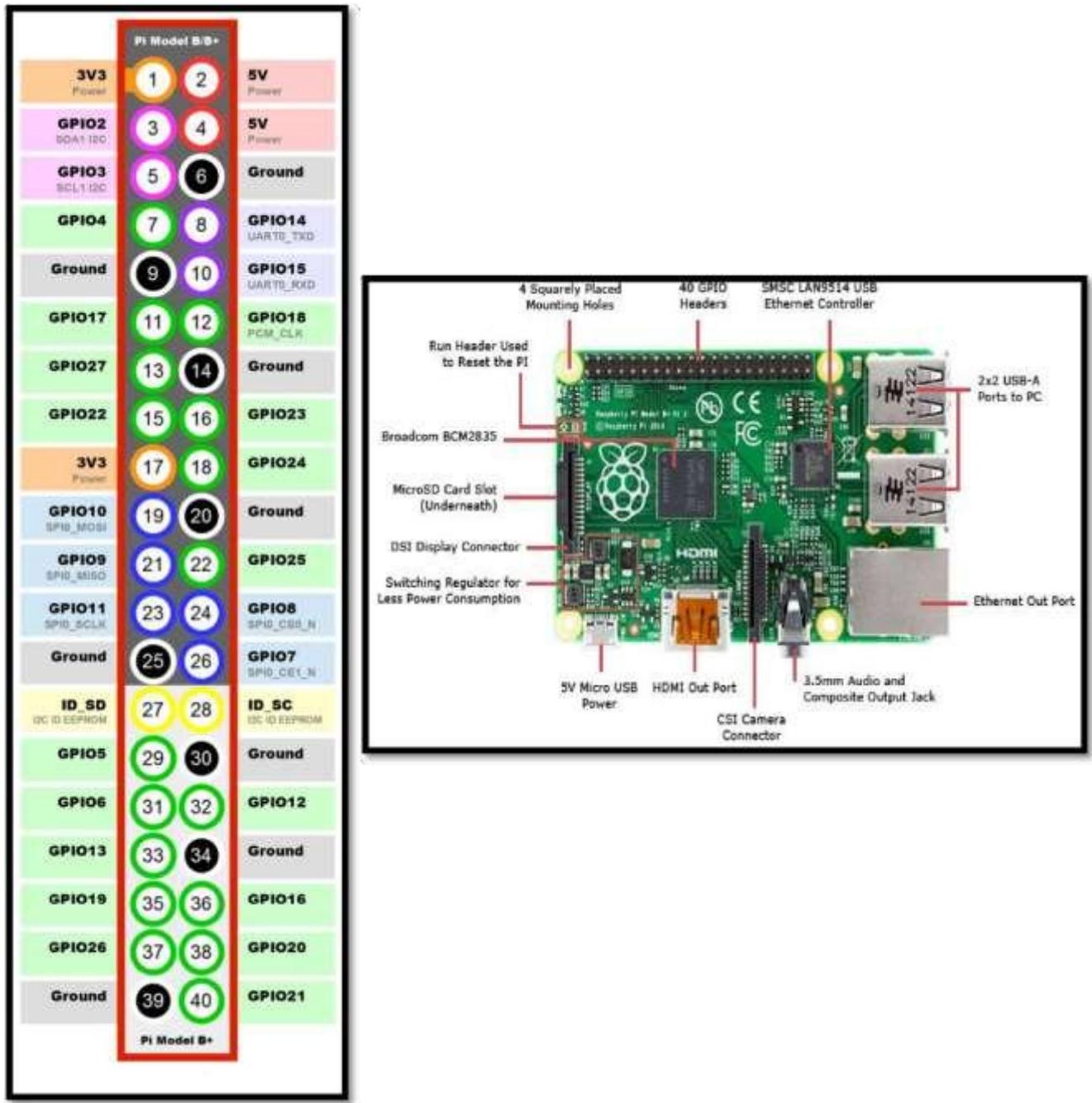
Sl.No	Components Name	Quantity
1	Raspberry Pi 3 -Model B+	1
2	HDMI to VGA Adapter	1
3	Power Guard (5V)	1
4	LED	1
5	Connecting Wires	As required
6	DHT11 Sensor	1

Specifications:

The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range.

- BroadcomBCM2837B0,Cortex-A53(ARMv8)64-bitSoC@1.4GHz
- 1GB LPDDR2SDRAM
- 2.4GHzand5GHzIEEE802.11.b/g/n/acwirelessLAN,Bluetooth4.2,BLE
- GigabitEthernetoverUSB2.0(maximumthroughput300Mbps)
- Extended 40-pinGPIO header
- Full-sizeHDMI
- 4 USB 2.0ports
- CSICameraportforconnectingaRaspberryPicamera
- DSIDisplayportforconnectingaRaspberryPitouchscreendisplay
- 4-polestereooutputandcompositevideoport
- MicroSDportforloadingyouroperatingsystemandstoringdata
- 5V/2.5ADCpowerinput
- Power-over-Ethernet(PoE)support(requiresseparatePoEHAT)

Pin Configuration:



As shown in above figure, there are 40 output pins for the PI. But all 40 pinout cannot be programmed to our use. Only 26 GPIO pins can be programmed. These pins go from **GPIO2 to GPIO27**. These **26 GPIO pins can be programmed** as per need. Some of these pins also perform some special functions. With special GPIO put aside, we have **17 GPIO** remaining (Light greenCirl).

Task1: Blinking the LED

```
import RPi.GPIO as gpio
from time import sleep
gpio.setmode(gpio.BOARD)           //or gpio.setmode(gpio.BCM)
gpio.setup(11,gpio.OUT)

while True:
    gpio.output(11,gpio.HIGH)
    sleep(1)
    gpio.output(11,gpio.LOW)
    sleep(1)                      //in Seconds
```

Task 2: Control LED using Digital Button

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.IN) GPIO.setup(11,
GPIO.OUT)

while True:
    input=GPIO.input(7) if
    (input == True):
        GPIO.output(11, GPIO.HIGH)
        print("Led On") elif
    (input == False):
        GPIO.output(11, GPIO.LOW)
        print("Led OFF")
```

Task3: Blinking the LED using GPIO Zero Library

```
from gpiozero import LED  
from time import sleep  
  
led = LED(17)  
  
while True:  
    led.on()  
    sleep(1)  
    led.off()  
    sleep(1)
```

Task 4: Control the LED using Digital Button with GPIOZero Library

```
from gpiozero import LED, Button  
  
led = LED(17)  
button = Button(2)  
  
while True:  
    if button.is_pressed:  
        led.on()          //Active low by default  
    else:  
        led.off()
```

Task 5: Toggle LED using GPIO Zero Library

```
from gpiozero import LED  
from time import sleep  
  
led = LED(17)  
while True: led.toggle() sleep(1)
```

Task6: Display the Temperature and Humidity of the surrounding environment

```
import Adafruit_DHT

while True:
    humidity, temperature = Adafruit_DHT.read_retry(11, 27)          //GPIO27BCMNotation
    print("Humidity={}%;Temperature={}C".format(humidity,temperature))
```

Prerequisites:

1. sudo apt-get install git-core

2. Download The Library:

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```

3. Navigate to Adafruit_Python_DHT directory (folder), in the terminal type:

```
cd Adafruit_Python_DHT
```

4. Run the following commands in the terminal.

```
sudo apt-get install build-essential python-dev # python2
```

```
sudo apt-get install build-essential python3-dev # python3
```

To see the version of python:

```
>python -v
```

5. To install the library, in the terminal type:

```
sudo python setup.py install#python2
```

```
sudo python3 setup.py install#python3
```

Result :

Thus, the Study of Raspberry Pi is done successfully.

Ex No: 6

Date: **Design and development of a Temperature Detection System using LM35 Temperature sensor**

Aim:

To design a temperature detection system using LM35 Temperature sensor and display using serial monitor.

Components Required:

Sl.No	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Power Adaptor – 12 V	1
4	3pin RMC Cable	2
5	PC with Windows OS -32/64 bit	1
6	PIR Sensor	1
7	Buzzer	1
8	LED	1

Task #1

Measure the temperature and print message in serial monitor if motion occurred.

```
// Define the analog pin for LM35 and digital pin for the buzzer
const int sensorPin = A0;
const int buzzerPin = 3;

// Temperature threshold in Celsius
const float tempThreshold = 35.0;

void setup() {
```

```

// Initialize serial communication
Serial.begin(9600);

// Set the buzzer pin as output
pinMode(buzzerPin, OUTPUT);

// Turn the buzzer off initially
digitalWrite(buzzerPin, LOW);
}

void loop() {
    // Read the analog value from the LM35 sensor
    int sensorValue = analogRead(sensorPin);

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

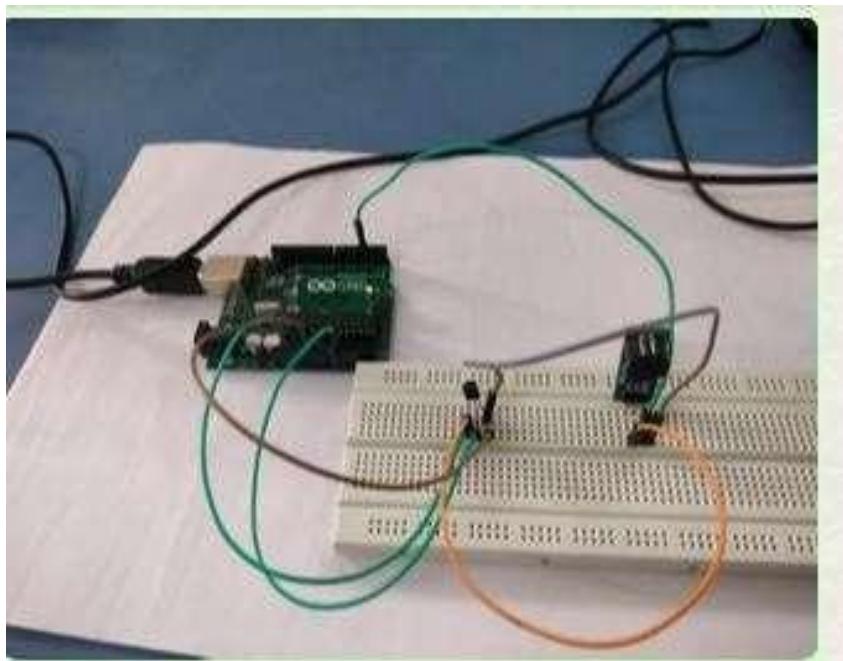
    // Convert the voltage to temperature in Celsius
    float temperatureC = voltage * 100;

    // Print the temperature to the Serial Monitor
    Serial.print("Temperature: ");
    Serial.print(temperatureC);
    Serial.println(" °C");

    // Check if the temperature exceeds the threshold
    if (temperatureC > tempThreshold) {
        // Turn on the buzzer
        digitalWrite(buzzerPin, HIGH);
        Serial.println("Buzzer ON: High temperature detected!");
    } else {
        // Turn off the buzzer
        digitalWrite(buzzerPin, LOW);
        Serial.println("Buzzer OFF: Temperature is normal.");
    }

    // Delay for a short interval before the next reading
    delay(1000);
}

```



```
4_temperaturesensordesign.ino

1 // Define the analog pin for LM35 and digital pin for the buzzer
2 const int sensorPin = A0;
3 const int buzzerPin = 3;
4
5 // Temperature threshold in Celsius
6 const float tempThreshold = 35.0;
7
8 void setup() {
9     // Initialize serial communication
10    Serial.begin(9600);
11
12    // Set the buzzer pin as output
13    pinMode(buzzerPin, OUTPUT);
14
15    // Turn the buzzer off initially
16    digitalWrite(buzzerPin, LOW);
17 }
18
19 void loop() {
20 }
```

Output Serial Monitor - x

Message (Enter to send message to 'Arduino Uno' on 'COM8')

16:05:12.801 -> Buzzer was high: temperature detected!	New Line
16:05:16.798 -> Temperature: 36.17 °C	
16:05:16.798 -> Buzzer ON: High temperature detected!	
16:05:17.804 -> Temperature: 35.88 °C	
16:05:17.804 -> Buzzer OFF: High temperature detected!	
16:05:18.793 -> Temperature: 36.18 °C	
16:05:19.823 -> Buzzer ON: High temperature detected!	
16:05:19.798 -> Temperature: 36.17 °C	
16:05:19.798 -> Buzzer OFF: High temperature detected!	

Result:

Thus, the design of temperature measurement system using LM35 sensor and output display using serial monitor is done successfully.

Ex No: 7

Date: **Design and development of a Gas Detection System using MQ5 sensor**

Aim:

To write programs for gas detection system using MQ5 Sensor and display using serial monitor.

Components Required:

Sl.No	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Power Adaptor – 12 V	1
4	3pin RMC Cable	2
5	PC with Windows OS -32/64 bit	1
6	Gas Sensor – MQ5	1
7	Buzzer	1
8	LED	1

Detect the gas and print message in serial monitor

```
#define MQ5_PIN A0 // Analog pin connected to MQ-5 sensor
#define BUZZER_PIN 8
void setup() {
    pinMode(BUZZER_PIN, OUTPUT);
    Serial.begin(9600); // Start serial communication
}

void loop() {
    int gasValue = analogRead(MQ5_PIN); // Read sensor value
    Serial.print("Gas Level: ");
    Serial.println(gasValue); // Print value on Serial Monitor

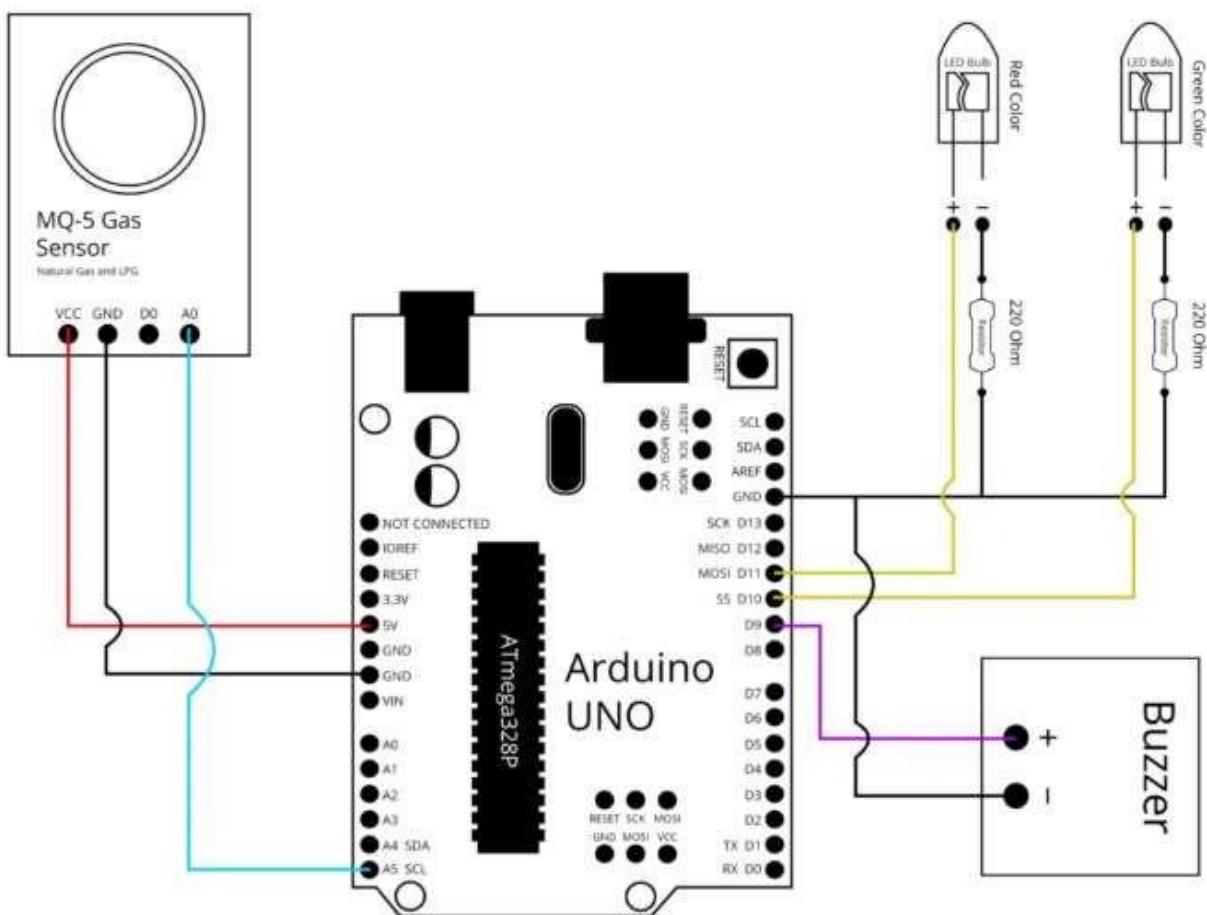
    // Optional: Trigger warning if gas level exceeds threshold
```

```

if (gasValue > 800) { // Adjust threshold as needed
    Serial.println("Warning! High gas concentration detected!");
    digitalWrite(BUZZER_PIN, HIGH); // Turn on buzzer
} else {
    digitalWrite(BUZZER_PIN, LOW); // Turn off buzzer
}

delay(1000); // Wait 1 second before next reading
}

```



Result :

Thus, the system for gas detection system, to detect gas using MQ-5 sensor and display using serial monitor is done successfully.

Ex No: 8

Date:

Design and development of a Moisture Detection System using Soil Moisture Sensor

Aim:

To write programs for moisture detection system using Soil Moisture Sensor and display using serial monitor.

Components Required:

Sl.No	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Power Adaptor – 12 V	1
4	3pin RMC Cable	2
5	PC with Windows OS -32/64 bit	1
6	Soil Moisture Sensor	1
7	Buzzer	1
8	LED	1

Detect the soil moisture and print message in serial monitor

```
#define sensorPin A0
#define sensorPower 7
#define LED_PIN 8
void setup () {
  pinMode(sensorPower, OUTPUT);
  Serial.begin(9600);
}

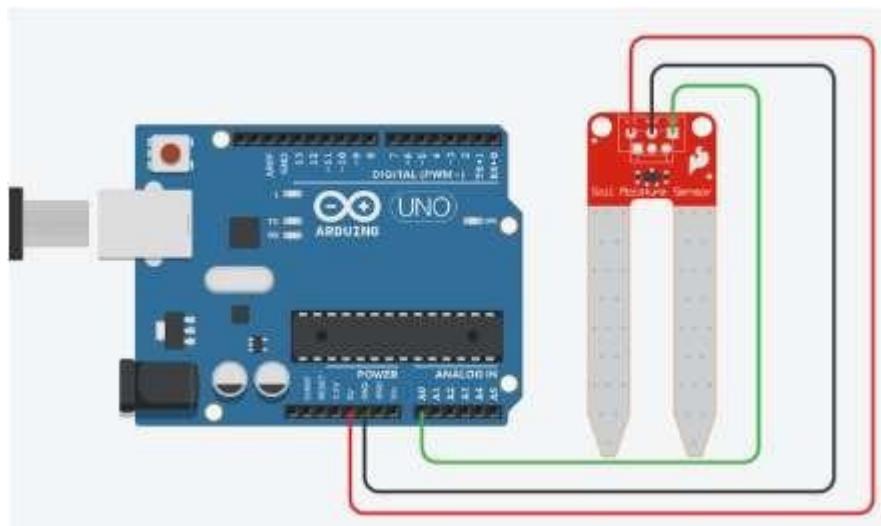
void loop () {
  // put your main code here, to run repeatedly:
  Serial.print("Analog output: ");
  int val = readSensor();
  Serial.println(val);
  if (val > 1000) {
```

```

Serial.println("Low moisture is detected! Water the plants");
digitalWrite(LED_PIN, HIGH);
} else {
  digitalWrite(LED_PIN, LOW);
}
Delay(1000);
}

int readSensor() {
  digitalWrite(sensorPower, HIGH);
  delay(10);
  int val = analogRead(sensorPin);
  digitalWrite(sensorPower, LOW);
  return val;
}

```



Result :

Thus, the design of soil moisture detection system using soil moisture sensor and output display using serial monitor is done successfully.

Ex No: 9

Date: **Design and development of an Intrusion Detection System using PIR sensor**

Aim:

To write programs for intrusion detection system to detect motion and display using serial monitor.

Components Required:

Sl.No	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Power Adaptor – 12 V	1
4	3pin RMC Cable	2
5	PC with Windows OS -32/64 bit	1
6	PIR Sensor	1
7	Buzzer	1
8	LED	1

Task #1

Detect the intrusion and print message in serial monitor if motion occurred.

```
#define PIR_PIN 2 // Digital pin connected to PIR sensor OUT
#define LED_PIN 8 // LED connected to pin 8

void setup() {
    pinMode(PIR_PIN, INPUT); // Set PIR sensor as input
    pinMode(LED_PIN, OUTPUT); // Set LED pin as output
    Serial.begin(9600); // Start serial communication
}
```

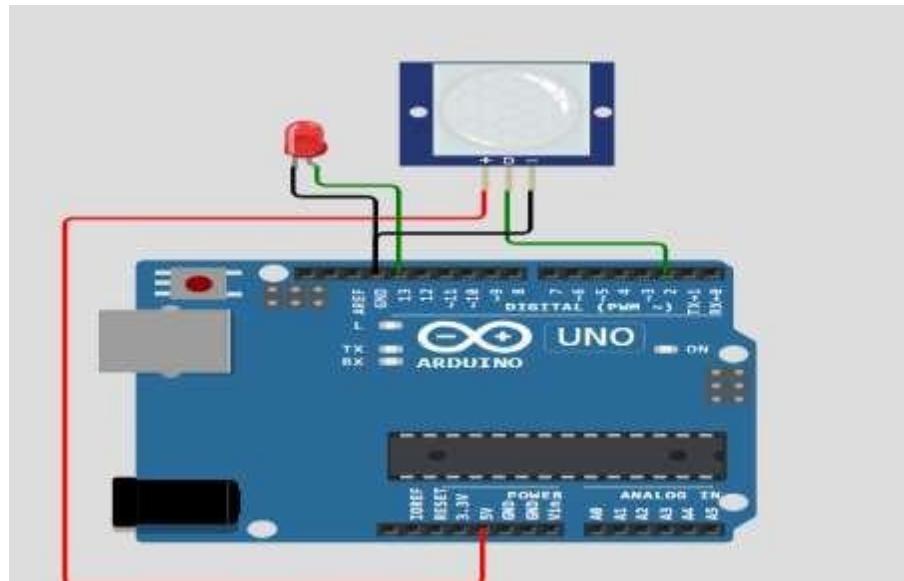
```

void loop() {
    int motionDetected = digitalRead(PIR_PIN); // Read PIR sensor state

    if(motionDetected == HIGH) { // If motion is detected
        Serial.println("Motion detected!");
        digitalWrite(LED_PIN, HIGH); // Turn ON LED
        delay(5000); // Keep LED on for 5 seconds
    } else {
        digitalWrite(LED_PIN, LOW); // Turn OFF LED
    }

    delay(1000); // Wait 1 second before checking again
}

```



Result :

Thus the design of intrusion detection system to detect intrusion using PIR sensor and display using serial monitor is done successfully.

Ex No: 10

Date:

Design and development of a Heartbeat Monitoring system using Heart beat sensor.

Aim:

To design and develop a heartbeat monitoring system using a heartbeat sensor and microcontroller for real-time heart rate measurement and display.

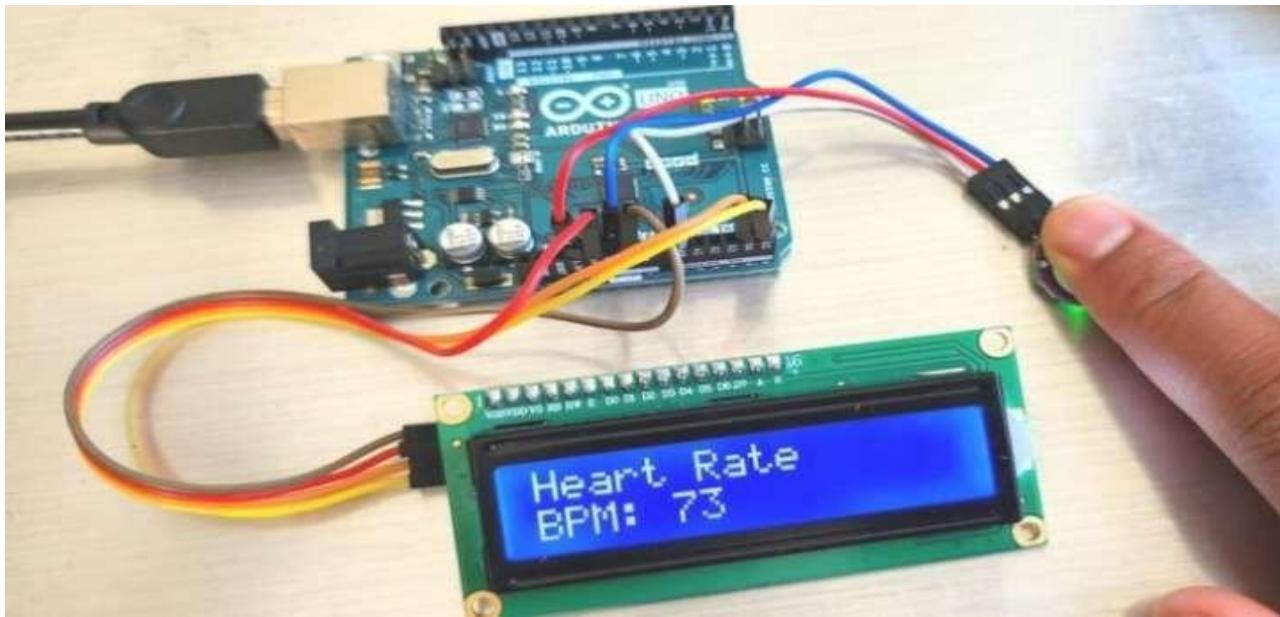
Components Required:

Sl.No	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Power Adaptor – 12 V	1
4	3pin RMC Cable	2
5	PC with Windows OS -32/64 bit	1
6	Pulse Sensor, MAX30100,	1
8	LED	1

THEORY

Working Principle of Heartbeat Sensor

The heartbeat sensor detects changes in blood flow based on optical sensing principles. It uses a light-emitting diode (LED) and a photodiode to measure variations in blood volume, which are converted into an electrical signal. The microcontroller processes the raw analog signals. Noise filtering is applied to remove unwanted variations. BPM is calculated using a time interval between successive pulses.



PROGRAM

```
#define SENSOR_PIN A0 // Heartbeat Sensor Pin
int bpm;

void setup() {
    Serial.begin(9600);
    pinMode(SENSOR_PIN, INPUT);
}

void loop() {
    int sensorValue = analogRead(SENSOR_PIN);
    bpm = map(sensorValue, 0, 1023, 60, 120); // Basic BPM mapping
    Serial.print("Heart Rate: ");
    Serial.print(bpm);
    Serial.println(" BPM");
    delay(1000);
}
```

Result :

Thus the design of a heartbeat monitoring system using a heartbeat sensor and display using serial monitor is done successfully.

