

1. Interfacing sensors with Arduino (Write a program to interface a temperature sensor (e.g., LM35) with an Arduino Uno. Read the sensor data, convert it into temperature values in degrees Celsius, and display the results on the Serial Monitor.)

Aim:

To interface temperature sensor with an Arduino Uno, read the analog voltage from the sensor, convert it into temperature values in degrees Celsius, and display the results on the Serial Monitor.

Procedure:

i) **Components Required:**

- Arduino Uno
- LM35 Temperature Sensor
- Connecting wires
- Breadboard
- USB cable for connecting Arduino to the computer

ii) **Circuit Diagram:**

- Connect the **VCC** pin of the LM35 sensor to the **5V** pin of the Arduino.
- Connect the **GND** pin of the LM35 sensor to the **GND** pin of the Arduino.
- Connect the **OUT** pin of the LM35 sensor to the **A1** analog pin of the Arduino.

iii) **Working Principle:**

- The LM35 outputs a voltage proportional to the temperature, where the output is 10 mV/°C.
- The Arduino reads the analog output of the sensor (0–5V) and converts it into a digital value (0–1023) using its 10-bit ADC.
- Using the conversion formula, the temperature in °C is calculated.

iv) **Steps to Implement:**

- Set up the circuit as per the diagram.
- Write the Arduino code to read the analog values from the sensor and convert them to temperature.
- Compile and upload the code to the Arduino Uno using the Arduino IDE.
- Open the Serial Monitor to view the temperature readings.

Program:

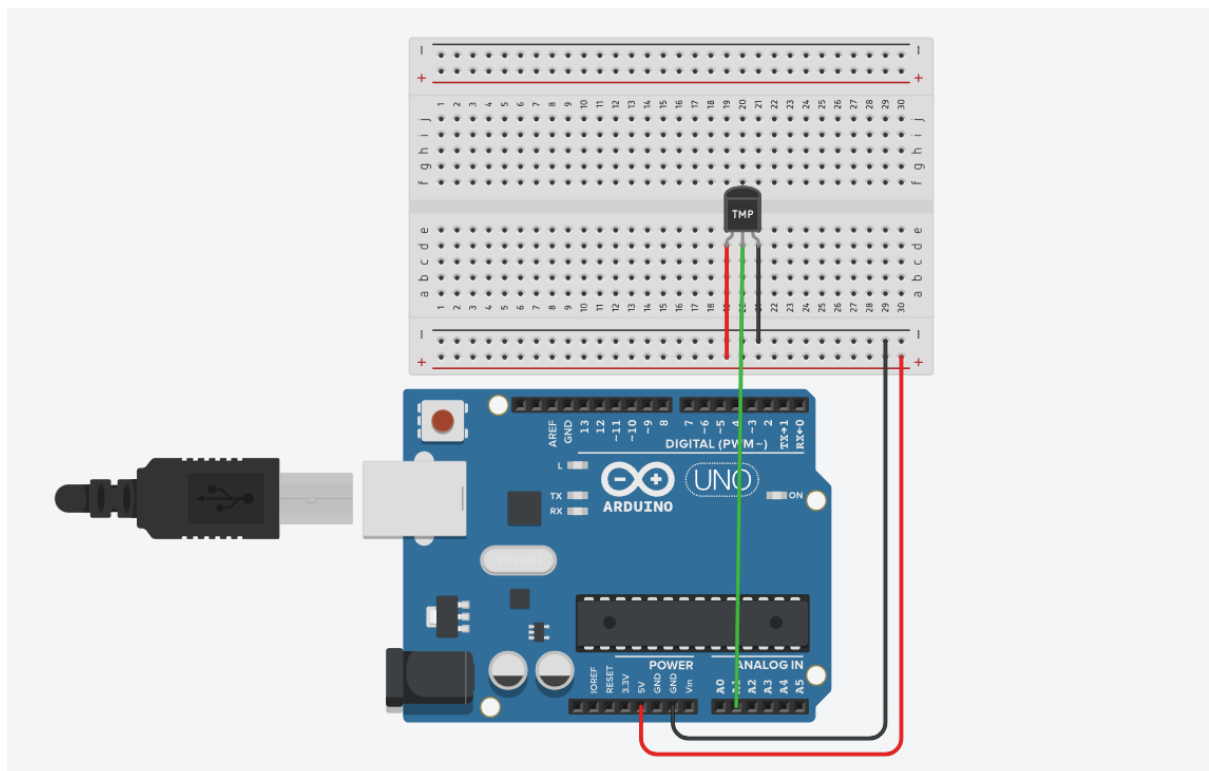
```
char degree = 176; // ASCII Value of Degree Symbol
const int sensor = A1; // Define the analog pin connected to the sensor

void setup() {
  pinMode(sensor, INPUT); // Set the sensor pin as INPUT
  Serial.begin(9600); // Start Serial Communication at 9600 bps
}

void loop() {
  int tmp = analogRead(sensor); // Read the analog value from the sensor
  float voltage = (tmp * 5.0) / 1024; // Convert analog value to voltage
  float millivolt = voltage * 1000; // Convert voltage to millivolts
  float tmpCel = (millivolt - 500) / 10; // Convert millivolts to Celsius

  Serial.print("Celsius: ");
  Serial.print(tmpCel); // Print the temperature
  Serial.println(degree); // Print the degree symbol
  delay(1000); // Wait for 1 second before the next reading
}
```

Diagram:



Output: Celsius: 24.71 Degree

Result:

Thus, the interfacing of the temperature sensor with the Arduino Uno has been executed successfully, and the output has been verified on the Serial Monitor.

2. Blinking of LED light (Write the Arduino code for blinking an LED with a delay of 1 second. Modify the code to make the LED blink with a delay of 500 milliseconds. Explain how the pinMode(), digitalWrite(), and delay() functions work in Arduino programming.)

Aim:

To write program to blink an LED with a delay of 1 second and modify the program to blink the LED with a delay of 500 milliseconds.

Procedure:

i) Components Required:

- Arduino Uno
- LED
- 220-ohm resistor
- Breadboard
- Connecting wires

ii) Circuit Connection:

- Connect the longer leg (anode) of the LED to **digital pin 2** of the Arduino through a 220-ohm resistor.
- Connect the shorter leg (cathode) of the LED to the **GND** pin of the Arduino.
- Ensure the connections are made securely.

iii) Working Principle:

- The pinMode() function configures a specific pin of the Arduino as an input or output. Configures the specified pin to act as an input or output.
- The digitalWrite() function is used to set a pin HIGH (5V) or LOW (0V). Sets the output voltage of the pin to HIGH (5V) or LOW (0V).
- The delay() function pauses the program for a specified duration in milliseconds. Pauses the program for a specified time in milliseconds.

iv) Steps to Implement:

- Set up the circuit according to the above connection diagram.
- Write the Arduino program to blink the LED with a 1-second delay.

- Upload the code to the Arduino using the Arduino IDE.
- Modify the delay value to 500 milliseconds and observe the behavior.

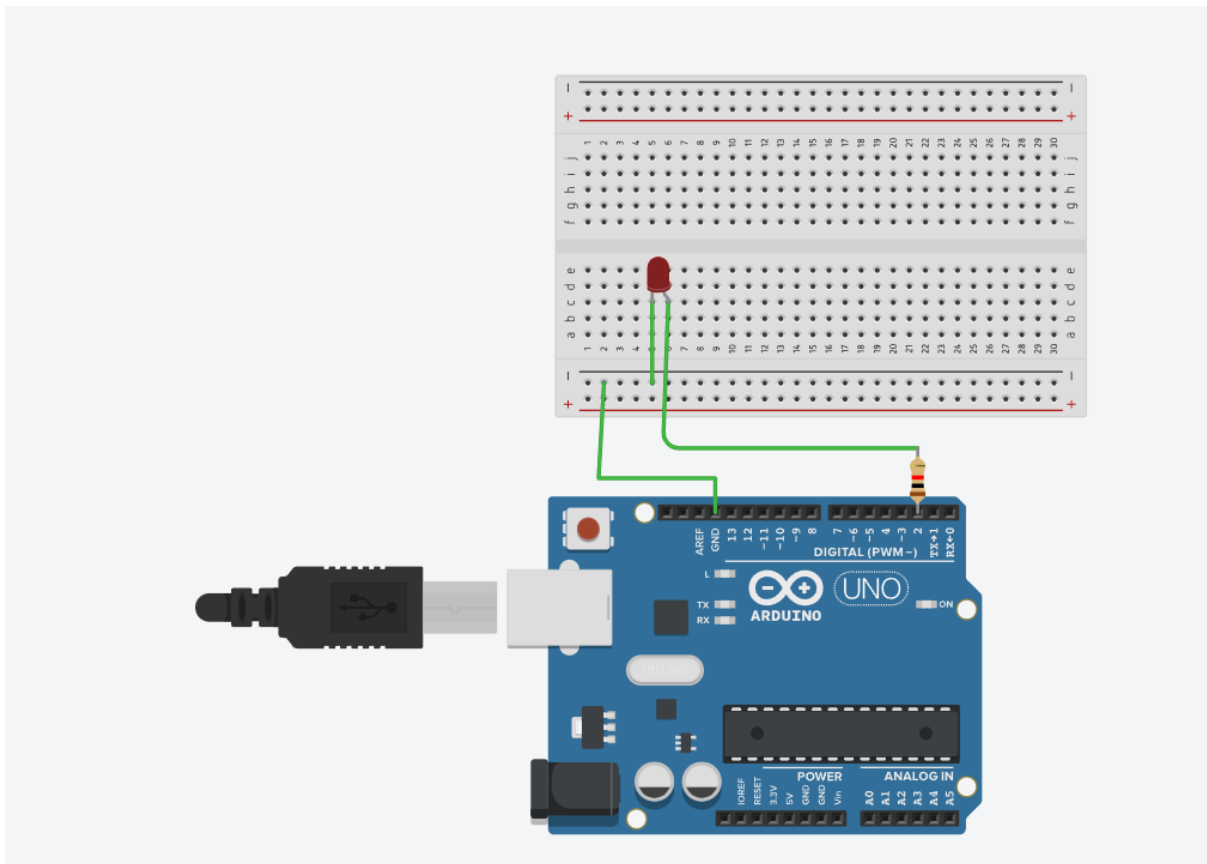
Program:

```
int a = 2; // Define pin 2 for the LED

void setup() {
  Serial.begin(9600); // Start Serial Communication at 9600 bps
  pinMode(a, OUTPUT); // Set pin 2 as OUTPUT
}

void loop() {
  digitalWrite(a, HIGH); // Turn on the LED
  Serial.println("LED is ON");
  delay(1000); // Wait for 1 second
  digitalWrite(a, LOW); // Turn off the LED
  Serial.println("LED is OFF");
  delay(1000); // Wait for 1 second
}
```

Diagram:



Output:

LED is ON

LED is OFF

Result:

Thus, the experiment for blinking an LED with delays of 1 second and 500 milliseconds has been successfully executed, and the output has been verified.

- 3. Design a real time IOT based system. (Design and implement a real-time IoT-based temperature monitoring system using Arduino and a temperature sensor (e.g., DHT11 or LM35). Send the sensor data to a cloud platform (e.g., ThingSpeak or Firebase) for real-time monitoring and visualization.)**

Aim:

To design and implement a real-time IoT-based temperature monitoring system using Arduino and a temperature sensor. The system measures the temperature, displays it on an LCD screen, and sends the data to a cloud platform (e.g., ThingSpeak or Firebase) for real-time monitoring and visualization.

Procedure:**i) Components Required:**

- Arduino Uno
- LM35 Temperature Sensor
- 16x2 LCD Display
- Potentiometer (for LCD contrast control)
- Connecting Wires
- Breadboard
- USB cable
- Wi-Fi module (e.g., ESP8266) or Ethernet Shield (for internet connectivity)

ii) Circuit Connection:

- Connect the **VCC** pin of the LM35 sensor to the **5V** pin of the Arduino.
- Connect the **GND** pin of the LM35 sensor to the **GND** pin of the Arduino.
- Connect the **OUT** pin of the LM35 sensor to **A0** (analog pin 0) of the Arduino.
- For the 16x2 LCD:

- Connect pins 1, 5, and 16 to **GND**.
- Connect pins 2 and 15 to **5V**.
- Connect pin 3 to the wiper of the potentiometer for contrast adjustment.
- Connect pins 4, 6, 11, 12, 13, and 14 to Arduino pins 8, 7, 6, 5, 4, and 3, respectively.

iii) Working Principle:

- The LM35 temperature sensor generates a voltage proportional to the surrounding temperature.
- The Arduino reads this analog voltage, converts it to temperature in °C, and displays it on the LCD.
- The temperature data is also sent to a cloud platform (e.g., ThingSpeak or Firebase) via an internet module for real-time monitoring.

iv) Steps to Implement:

- Assemble the circuit as per the connections mentioned.
- Write and upload the code to the Arduino using the Arduino IDE.
- Configure the cloud platform (ThingSpeak or Firebase):
 - Create an account on the platform.
 - Set up a new channel (ThingSpeak) or database (Firebase) to store temperature data.
 - Use appropriate libraries (e.g., WiFi.h or ESP8266WiFi.h) for internet connectivity.
- Modify the cloud connection settings in the Arduino code to match your API keys or credentials.

Program:

```
#include "LiquidCrystal.h"
LiquidCrystal lcd(8, 7, 6, 5, 4, 3);
int sensorPin = 0;

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
}
```

```

void loop() {
    int reading = analogRead(sensorPin);

    // Convert the analog reading to voltage
    float voltage = reading * 4.68;
    voltage /= 1024.0;

    // Convert voltage to temperature in Celsius
    float temperatureC = (voltage - 0.5) * 100;

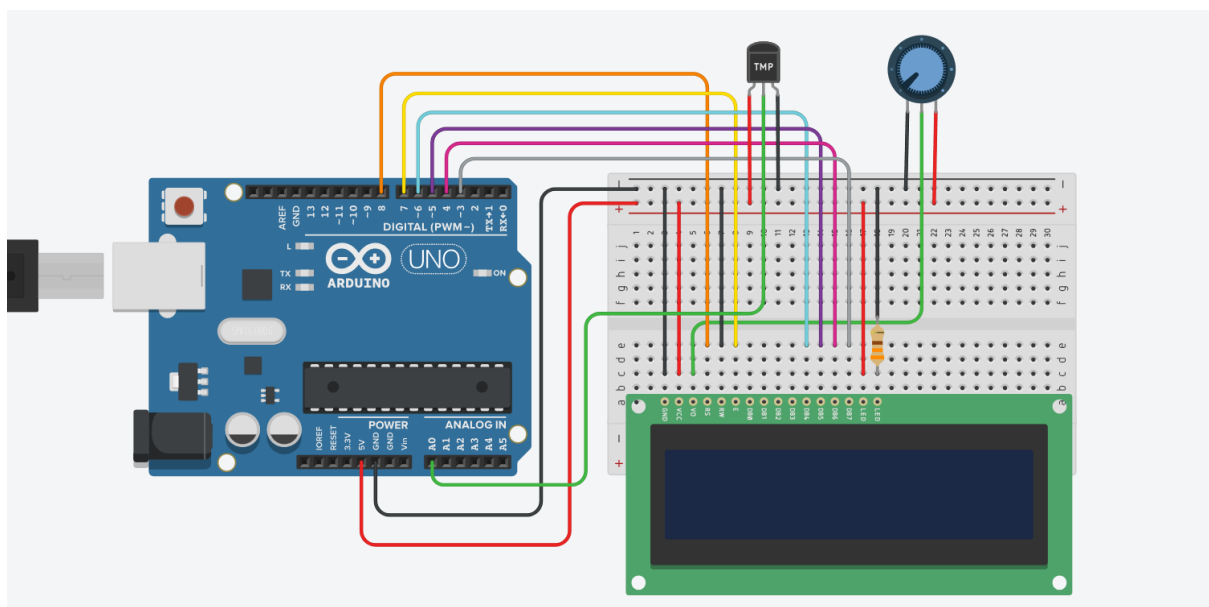
    // Print the temperature on Serial Monitor
    Serial.print(temperatureC);
    Serial.println(" degrees C");

    // Display the temperature on LCD
    lcd.setCursor(0, 0);
    lcd.print("Temperature Value");
    lcd.setCursor(0, 1);
    lcd.print(" degrees C");
    lcd.setCursor(11, 1);
    lcd.print(temperatureC);

    delay(100);
}

```

Diagram:



Output: Temperature value: 25.3 Degree

Result:

Thus, the IoT-based temperature monitoring system using Arduino has been designed and implemented successfully. The temperature is displayed locally on the LCD, monitored on the Serial Monitor, and sent to the cloud for real-time visualization and tracking. The output has been verified.

4. Interfacing Arduino to Zigbee module.

Aim:

To interface an Arduino board with a Zigbee module for wireless communication and exchange of data.

Procedure:

1. Connect the Zigbee module to the Arduino board using jumper wires:
 - Connect the VCC pin of the Zigbee module to the 3.3V output of the Arduino.
 - Connect GND pin of the Zigbee module to the GND pin of the Arduino.
 - Connect TX pin of the Zigbee module to a digital pin (e.g., pin 2) of the Arduino.
 - Connect RX pin of the Zigbee module to another digital pin (e.g., pin 3) of the Arduino.
2. Ensure the Zigbee module is configured properly (e.g., baud rate, PAN ID, etc.) according to the specifications and requirements of your project. Refer to the module's datasheet or documentation for configuration details.
3. Upload a basic sketch onto the Arduino board to establish serial communication with the Zigbee module.
4. Close the Arduino IDE. 5. Use the following Python code to communicate with the Arduino and Zigbee module via serial communication.

Program:

```
void setup() {
  Serial.begin(9600);      // Initialize serial communication at 9600 baud
  pinMode(2, INPUT);       // Set digital pin 2 as input (receives data from Zigbee)
  pinMode(3, OUTPUT);      // Set digital pin 3 as output (sends data to Zigbee)
}

void loop() {
  if (Serial.available() > 0) {
    char received = Serial.read(); // Read incoming data from Zigbee
    Serial.print("Received: ");
    Serial.println(received);      // Debug print received data
    // Additional processing of received data can be added here
  }
  // Additional logic for sending data to Zigbee can be added here
}
```



```

import serial
import time

# Replace 'COM3' with the correct serial port for your Arduino
ser = serial.Serial('COM3', 9600, timeout=1)
time.sleep(2) # Allow time for the serial connection to establish

print("Enter commands to send to Arduino-Zigbee setup ('q' to quit).")
while True:
    command = input("Enter command: ")
    if command.lower() == 'q':
        break
    else:
        ser.write(command.encode()) # Send command to Arduino as bytes
        response = ser.readline().decode().strip() # Read response from Arduino
        print("Response:", response)

ser.close() # Close the serial connection

```

Output:

Result

Thus, the Arduino to Zigbee communication system was designed and implemented successfully. Data was exchanged between the Arduino and Zigbee module, controlled via a Python script. The setup has been tested, and the output is verified.

5. Interfacing Arduino to Bluetooth Module.

Aim:

To establish communication between an Arduino board and a Bluetooth module for wireless data transfer.

Procedure:

1. Connect the Bluetooth module to the Arduino board using jumper wires:
 - Connect VCC pin of the Bluetooth module to the 5V output of the Arduino.
 - Connect GND pin of the Bluetooth module to the GND pin of the Arduino.
 - Connect TX pin of the Bluetooth module to the RX pin (pin 0) of the Arduino.
 - Connect RX pin of the Bluetooth module to the TX pin (pin 1) of the Arduino.

2. Ensure the Bluetooth module is configured properly according to its specifications (e.g., baud rate, pairing settings). Refer to the module's datasheet or documentation for configuration details.
3. Upload a basic sketch onto the Arduino board to establish serial communication with the Bluetooth module:
4. Close the Arduino IDE.
5. Use a Bluetooth terminal app on a smartphone or computer to connect to the Bluetooth module (ensure its paired and connected).
6. Send data from the Bluetooth terminal to the Arduino. The Arduino will echo back the received data for verification.

Program:

```
void setup() {  
  Serial.begin(9600);      // Initialize serial communication at 9600 baud  
  Serial.println("Bluetooth is ready."); // Send status message  
}  
  
void loop() {  
  if (Serial.available() > 0) {  
    char received = Serial.read();    // Read incoming data from Bluetooth  
    Serial.println(received);         // Echo received data back for verification  
    // Additional processing of the received data can be added here  
  }  
  // Additional logic to send data can also be implemented here  
}
```

Output:

Result:

Thus, a Bluetooth communication system was designed and implemented successfully. Data was exchanged between the Arduino and a paired Bluetooth device. The setup was tested, and the output was verified.

6. Communicate between Arduino and Raspberry PI using any wireless medium.

Aim:

To establish wireless communication between an Arduino and a Raspberry Pi using Bluetooth, implementing a basic data exchange system.

Procedure:

1. Setting Up Arduino:

- Connect the VCC and GND of the Bluetooth module to the 5V and GND on the Arduino, respectively.
- Connect the TX pin of the Bluetooth module to the RX pin of the Arduino.
- Connect the RX pin of the Bluetooth module to the TX pin of the Arduino.

2.Upload the following Arduino code to enable Bluetooth communication.

- This code uses the SoftwareSerial library to establish communication with the Bluetooth module.

3.Setting Up Raspberry Pi:

- Ensure that your Raspberry Pi has Bluetooth capability.
- Connect the Raspberry Pi to the Internet to install required libraries.

4. Python Code for Raspberry Pi:

- Install the pyserial library on the Raspberry Pi:

5. Use the following Python script to receive and send data:

6.Replace "/dev/ttyS0" with the correct serial port based on your Raspberry Pi's configuration.

Program:


```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX, TX pins for Bluetooth communication

void setup() {
  Serial.begin(9600);           // Serial communication with PC
  BTSerial.begin(9600);        // Serial communication with Bluetooth
}

void loop() {
  if (Serial.available()) { // Check for data from PC
    char c = Serial.read();
    BTSerial.write(c);      // Send data to Bluetooth module
  }

  if (BTSerial.available()) { // Check for data from Bluetooth
    char c = BTSerial.read();
    Serial.write(c);         // Send data to PC for display
  }
}
```



```

import serial

# Replace '/dev/rfcomm0' with the correct Bluetooth serial port on Raspberry Pi
ser = serial.Serial('/dev/rfcomm0', 9600)

try:
    while True:
        # Receive data from Arduino
        data = ser.readline().decode().strip()
        if data:
            print("Arduino:", data)

        # Send data to Arduino
        message = input("Enter message to send to Arduino: ")
        ser.write(message.encode())
except KeyboardInterrupt:
    ser.close()
    print("Serial connection closed.")

```

Output:

Arduino: Hello, Raspberry Pi!
 Message from Raspberry Pi: Hello, Arduino!

Result:

Thus, a wireless communication system was successfully established between an Arduino and a Raspberry Pi using Bluetooth. The data exchange was tested, and the output was verified.