Title of Experiment:

Data Logging Using Arduino and Excel (Serial Communication)

Aim:

To acquire and log real-time temperature data from an LM35 sensor using Arduino and transmit it to a computer via serial communication for visualization and storage in Excel using PLX-DAQ or Python.

Apparatus:

- Arduino Uno board
- LM35 temperature sensor
- Breadboard
- Jumper wires
- USB cable
- Arduino IDE installed on PC
- Excel with **PLX-DAQ Add-in** (or Python with Serial library)
- Computer system with Windows OS

Precautions:

1. Ensure correct connections of LM35 (Vcc, Output, GND).
2. Do not exceed 5V input to LM35 to avoid damage.
3. Ensure proper baud rate matching between Arduino and PLX-DAQ or Python script.
4. Save logged data regularly to avoid loss during power failure.
5. Do not open the serial monitor in Arduino IDE when logging with PLX-DAQ/Python (only one application can access the COM port at a time).

Theory:

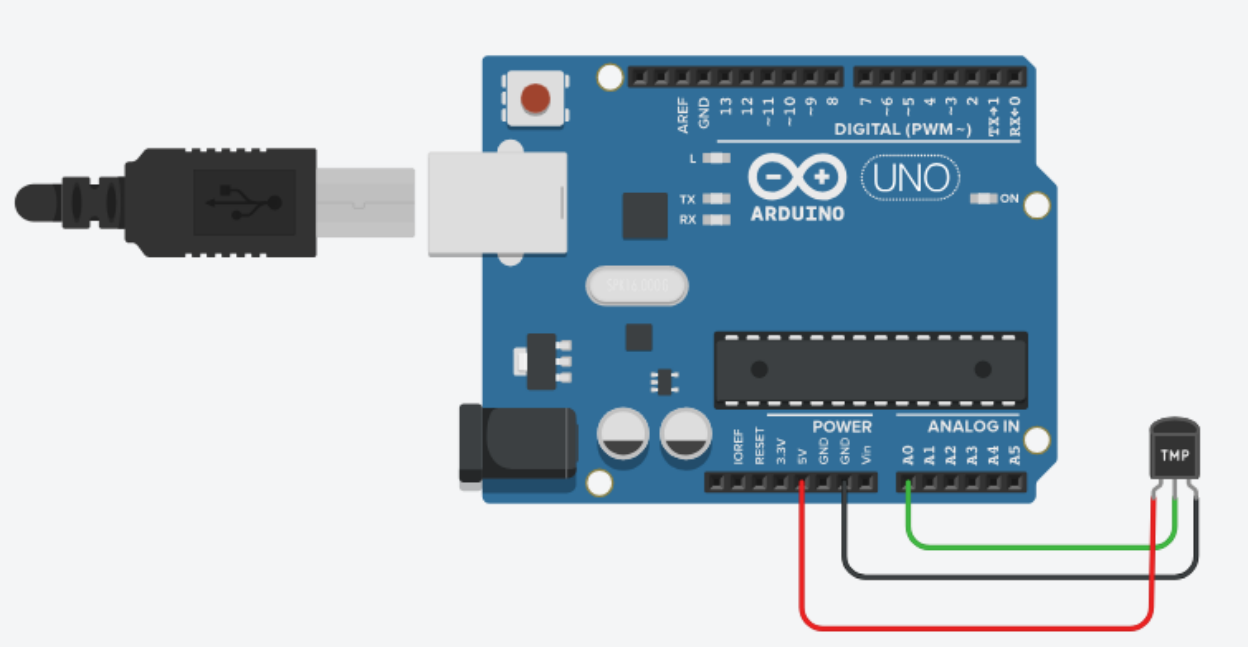
Data logging is the process of collecting and storing sensor data for later analysis. Arduino can send sensor data via **serial communication** (USB) using the `Serial.print()` or `Serial.println()` commands.

- **LM35 sensor** outputs analog voltage proportional to temperature (10 mV/°C).
- Arduino reads this analog value using its **ADC** and converts it to temperature.
- The temperature data is transmitted through the **Serial Port (COM)**.
- On the PC side, **PLX-DAQ** (a Microsoft Excel add-in) or **Python** with `pyserial` library can read and save the data for analysis.

Circuit Diagram:

LM35 Pinout:

- Pin 1 (Vcc) → 5V
- Pin 2 (Output) → A0
- Pin 3 (GND) → GND



Procedure:

Part A: Hardware Setup

1. Connect the LM35 sensor to Arduino as per the circuit diagram.
2. Connect Arduino to the computer using USB.

Part B: Arduino Code
3. Open Arduino IDE and write the program to read temperature and send it via serial.
4. Upload the code to the Arduino board.

Part C: Data Logging
5. **For Excel (PLX-DAQ)**:

- Open Excel and run PLX-DAQ Add-in.
- Select the COM port, set baud rate to 9600, and click "Connect".
- Temperature data will appear in real time and can be saved/graph plotted.

OR

6. **For Python**:
    - Run Python script with serial port reading code.
    - Save the data to a CSV file or plot it using matplotlib (optional).

Arduino Programming (with Comments):

```
cpp
CopyEdit
const int sensorPin = A0;      // LM35 connected to analog pin A0
float temperatureC = 0;        // Variable to store temperature in Celsius

void setup() {
  Serial.begin(9600);          // Start serial communication at 9600 baud
  Serial.println("CLEARDATA");  // Clear previous data in PLX-DAQ
  Serial.println("LABEL,Time,Temperature (C)"); // Labels for columns
}

void loop() {
  int sensorValue = analogRead(sensorPin);  // Read analog value (0-1023)

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

  // Convert voltage to temperature
  temperatureC = voltage / 0.01;  // LM35 gives 10mV per °C

  // Send data to PLX-DAQ or serial monitor
```

```
  Serial.print("DATA,TIME,");   // Required format for PLX-DAQ
  Serial.println(temperatureC); // Send temperature data

  delay(1000);  // Delay 1 second between readings
}
```

Conclusion:

In this experiment, real-time data from an LM35 temperature sensor was successfully read by Arduino and logged to a PC using serial communication. Using PLX-DAQ (or Python), the data was visualized and stored in Excel for future analysis. This approach is widely used in environmental monitoring, IoT applications, and data analytics.

---

4 Short Questions and Answers:

Q1: Why is serial communication used in this experiment?
Answer: Serial communication allows Arduino to send real-time sensor data to a computer for monitoring, visualization, and storage.

Q2: What is the output of the LM35 sensor?
Answer: The LM35 provides an analog voltage output of **10 mV per °C**, i.e., 250 mV for 25°C.

Q3: What does PLX-DAQ do in this experiment?
Answer: PLX-DAQ captures serial data from Arduino and logs it directly into Excel in real time for display and storage.

Q4: Why do we use `Serial.print()` and `Serial.println()` in Arduino?
Answer: These functions are used to send data from Arduino to the PC via the USB port, enabling data logging or debugging.