AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Engineering

Lab Report

Experiment # 08

Experiment Title: Implementation of a weather forecast system using the ADC modules of an Arduino.

Date of Perform:	5 May 2025	**Date of Submission:**	12 May 2025
Course Title:	Microprocessor and Embedded Systems Lab		
Course Code:	EE4103	**Section:**	P
Semester:	Spring 2024-25	**Degree Program:**	BSc in CSE
Course Teacher:	**Prof. Dr. Engr. Muhibul Haque Bhuyan**		

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case Study is my/our original work; no part has been copied from any other student's work or any other source except where due acknowledgment is made.
3. No part of this Assignment/Case Study has been written for me/us by any other person except where such collaboration has been authorized. by the concerned teacher and is acknowledged in the assignment.
4. I/we have not previously submitted or am submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared, and archived to detect plagiarism.
6. I/we permit a copy of my/our marked work to be retained by the Faculty Member for review by any internal/external examiners.
7. I/we understand that Plagiarism is the presentation of the work, idea, or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offense that may lead to expulsion from the University. Plagiarized material can be drawn from, and presented in, written, graphic, and visual forms, including electronic data, and oral presentations. Plagiarism occurs when the origin of the source is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or copy my/our work.

** Student(s) must complete all details except the faculty use part.*
** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group # 01

Sl No	Name	ID	PROGRAM	SIGNATURE
1	Md. Saikot Hossain	23-51242-1	BSc in CSE	
2	Md. Mosharof Hossain Khan	23-51259-1	BSc in CSE	
3	Rimal Banik	23-51260-1	BSc in CSE	
4	Md. Rahidul Islam	23-51269-1	BSc in CSE	
5	Rahat Ahmed	21-44911-2	BSc in CSE	

Faculty use only		
FACULTY COMMENTS	**Marks Obtained**	
	Total Marks	

Contents

Objectives:

The objectives of this experiment are to-

a) Familiarize the students with the Micro-controller-based weather forecast system.
b) Measure environmental parameters, such as temperature, pressure, and humidity.

Apparatus:

1. Arduino IDE 2.3.5
2. Arduino UNO Microcontroller board
3. BMP180
4. Inches96 inch OLED 128X64
5. Breadboard
6. Jumper wires

Circuit Diagram:

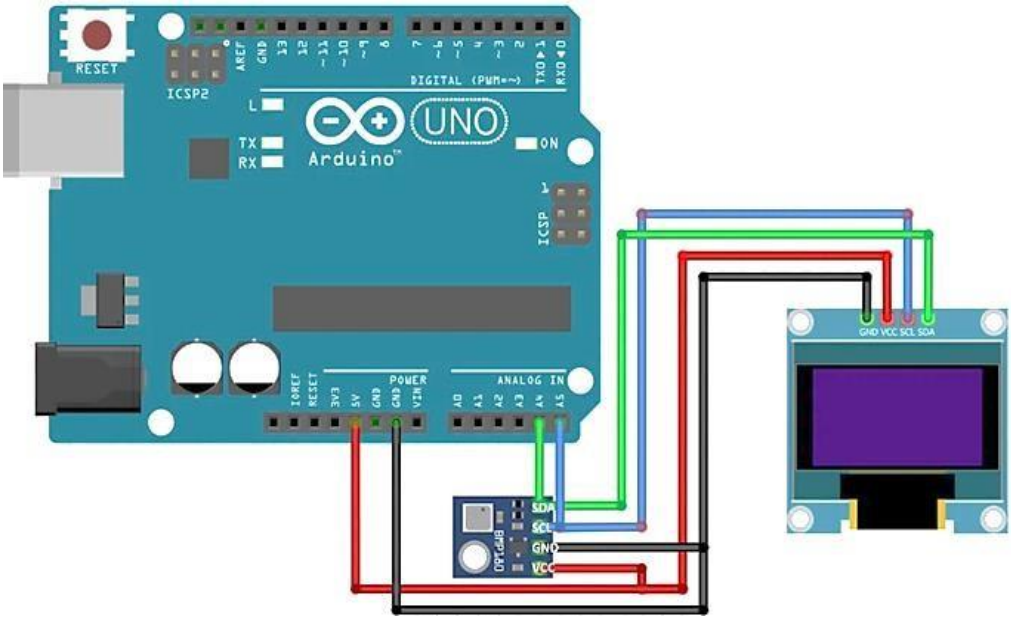

Figure 01: Arduino board's pin connections with a weather sensor and an OLED

Code Explanation:

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_BMP085.h>
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT);
Adafruit_BMP085 bmp;

#define SEALEVELPRESSURE_HPA (101500)
float simpleweatherdifference, currentpressure, predictedweather, currentaltitude;

void setup()
{
// put your setup code here, to run once:
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
if (!bmp.begin())
{
Serial.println("Could not find a valid BMP085 sensor, check wiring!");
while (1) {}
}
}

void loop()
{
// put your main code here, to run repeatedly:
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(0,5);
display.print("BMP180");
display.setCursor(0,19);
display.print("T=");
display.print(bmp.readTemperature(),1);
display.println("*C");
/prints BME180 pressure in Hectopascal Pressure Unit/
display.setCursor(0,30);
display.print("P=");
display.print(bmp.readPressure()/100.0F,1);
display.println("hPa");
```

```
/prints BME180 altitude in meters/
display.setCursor(0,40);
display.print("A=");

display.print(bmp.readAltitude(SEALEVELPRESSURE_HPA),1);
display.println("m");
delay(6000);
display.display();

currentpressure=bmp.readPressure()/100.0;
predictedweather=(101.3*exp(((float)(currentaltitude))/(-7900)));
simpleweatherdifference=currentpressure-predictedweather;

//display.clearDisplay();
display.setCursor(0,50);
if (simpleweatherdifference>0.25)
display.print("SUNNY");
if (simpleweatherdifference<=0.25)
display.print("SUNNY/CLOUDY");
if (simpleweatherdifference<-0.25)
display.print("RAINY");
display.display();
delay(2000);
}
```

Explanation: This Arduino-based weather station project uses a BMP180 sensor and an OLED display to measure and display real-time environmental data including temperature, atmospheric pressure, and altitude. The code begins by initializing the required libraries for I2C communication, graphics display, and sensor interfacing. In the setup() function, the OLED display is started at address 0x3C and the BMP180 sensor is initialized. If the sensor is not detected, the system halts to avoid false readings. Within the loop(), the display is first cleared and then updated with the latest readings. The temperature is shown in degrees Celsius, the pressure in hectopascals (hPa), and the altitude is calculated relative to a predefined sea-level pressure. The system also includes a basic weather prediction feature. It compares the current measured pressure to a theoretically predicted pressure based on altitude. If the pressure difference is significantly positive, it indicates sunny weather. If it's near zero, it suggests partly cloudy conditions, and if the difference is significantly negative, it predicts rain. The OLED shows this forecast as "SUNNY", "SUNNY/CLOUDY", or "RAINY". This experiment effectively demonstrates the integration of environmental sensing with a visual output system and applies mathematical logic for basic weather forecasting, making it a useful tool for understanding sensor interfacing, data processing, and embedded display programming.

Hardware Implementation and Explanation:

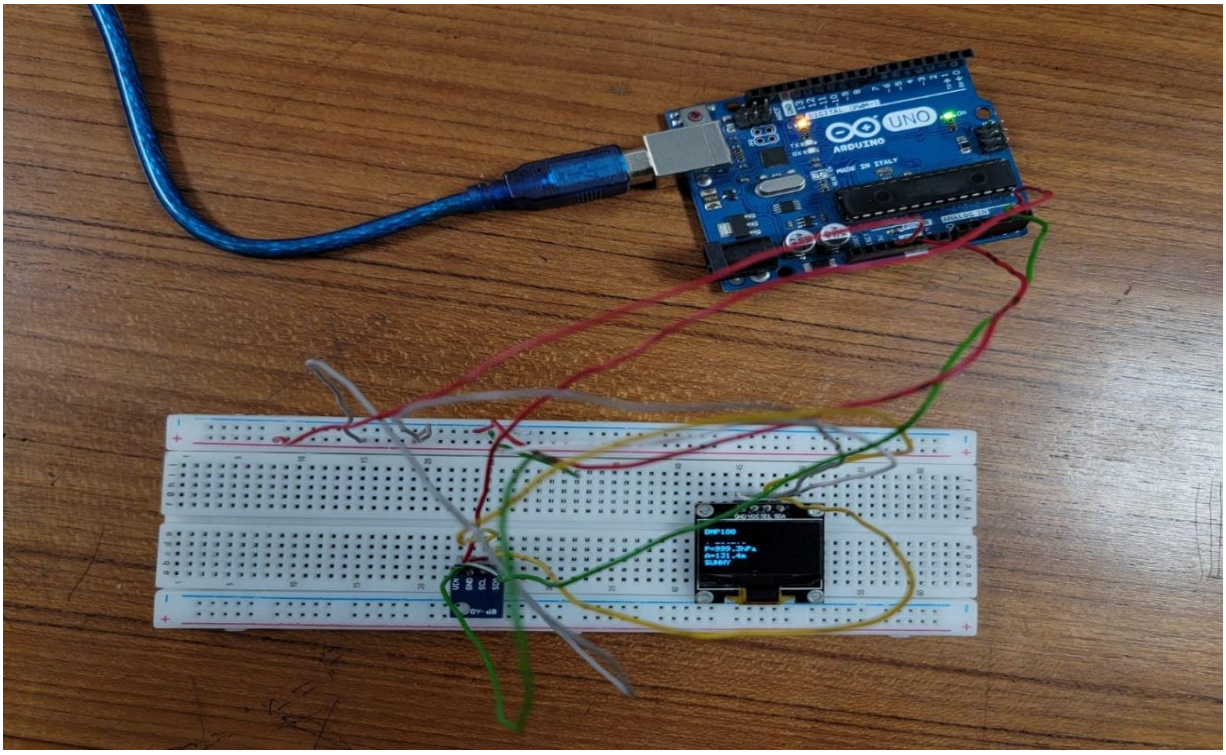

Figure 02: Hardware Implementation of a weather forecast system

Explanation: In the setup, an Arduino Uno is used as the main microcontroller. A BMP180 sensor module is connected to the Arduino to measure environmental data such as temperature and pressure. The BMP180 communicates using the I2C protocol, with its SDA (data line) connected to A4 and SCL (clock line) connected to A5 on the Arduino. The VCC and GND pins of the BMP180 are connected to the 5V and GND pins of the Arduino to supply power. Additionally, an OLED display is used to show the weather data in real time. The OLED also uses the I2C interface, sharing the same SDA (A4) and SCL (A5) lines as the BMP180 sensor. The OLED's VCC and GND are connected to the 5V and GND pins of the Arduino as well. The use of I2C allows both devices to operate simultaneously on the same communication lines. The analog-to-digital converter (ADC) module of the Arduino reads the sensor's output internally via I2C and converts it into digital values that are processed and displayed. The setup is powered via a USB cable connected to the Arduino, which also enables code uploading and serial communication for debugging if needed.

Experimental Output Results:

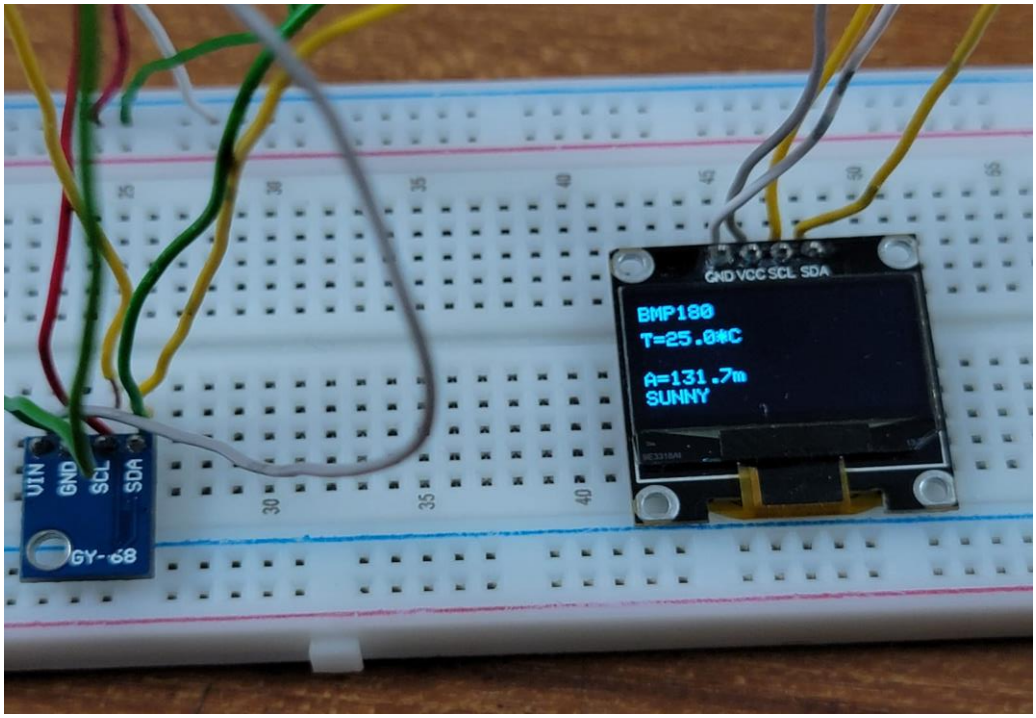

Figure 03:BMP180 readings on OLED display (25°C, 131.7m, Sunny).

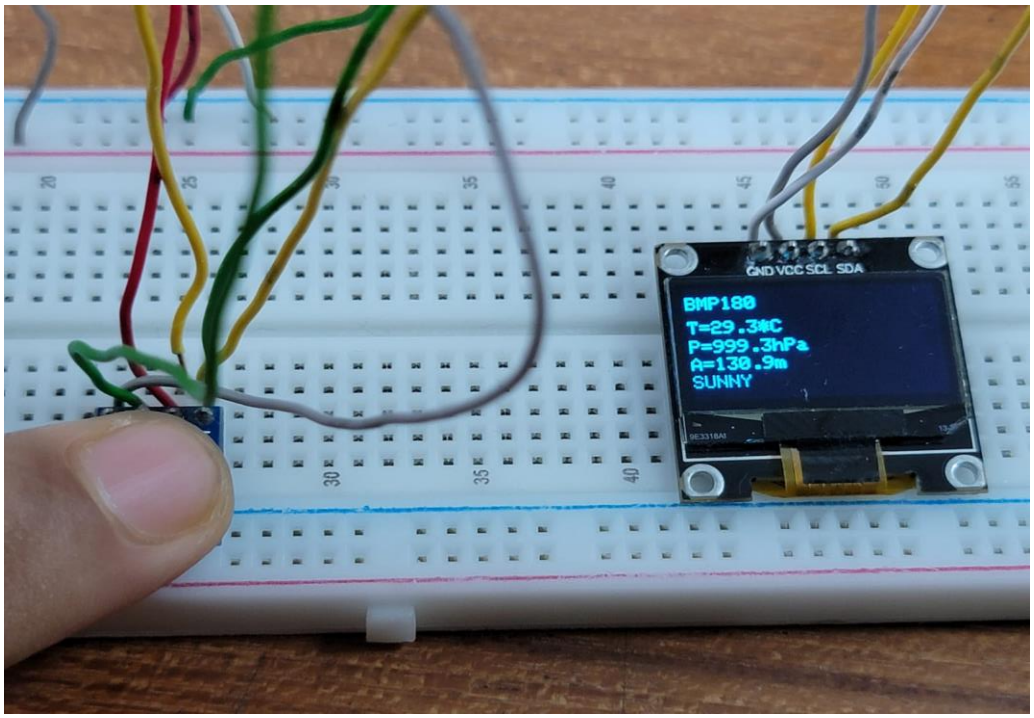

Figure 04: BMP180 readings on OLED display (29.3°C, 130.9m, Sunny).

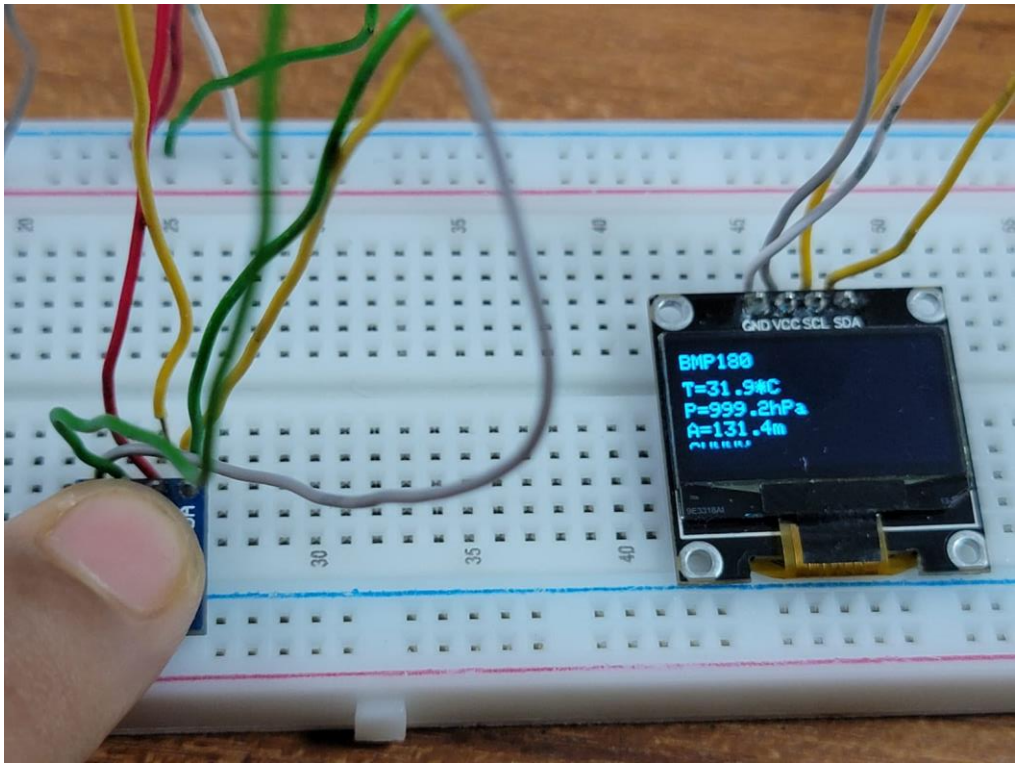

Figure 05: BMP180 readings on OLED display (31.9°C, 131.4m, Sunny).

Explanation: In this setup, a BMP180 sensor is used in conjunction with an Arduino Uno and an OLED display to implement a basic weather forecasting system. The BMP180 sensor measures both temperature and atmospheric pressure, which are then used to calculate altitude and infer simple weather conditions. The sensor is connected to the Arduino via I2C communication using the SDA and SCL pins, while the OLED display presents real-time readings such as temperature, altitude (denoted as "A"), and a weather status like “SUNNY.” When the system runs without any physical interaction, the sensor records the ambient room temperature initially around 25.0°C and displays "SUNNY" as the weather condition based on stable pressure readings.When the sensor is touched, the temperature reading increases rising to 29°C and later to 31°C due to heat transfer from the hand to the sensor. This temperature change can slightly influence the pressure reading or its interpretation, as the sensor uses temperature-compensated pressure values to calculate altitude. As a result, the altitude reading also fluctuates slightly: for example, it might be **131.7 m at 25°C**, drop to **130.9 m at 29°C,** and then rise to **131.4 m at 31°C.** These small variations are expected, as altitude is derived using both pressure and temperature in the barometric formula. Despite these fluctuations, the weather status may remain "SUNNY" if the pressure stays within the predefined threshold range in the code.

Simulation Output Results:

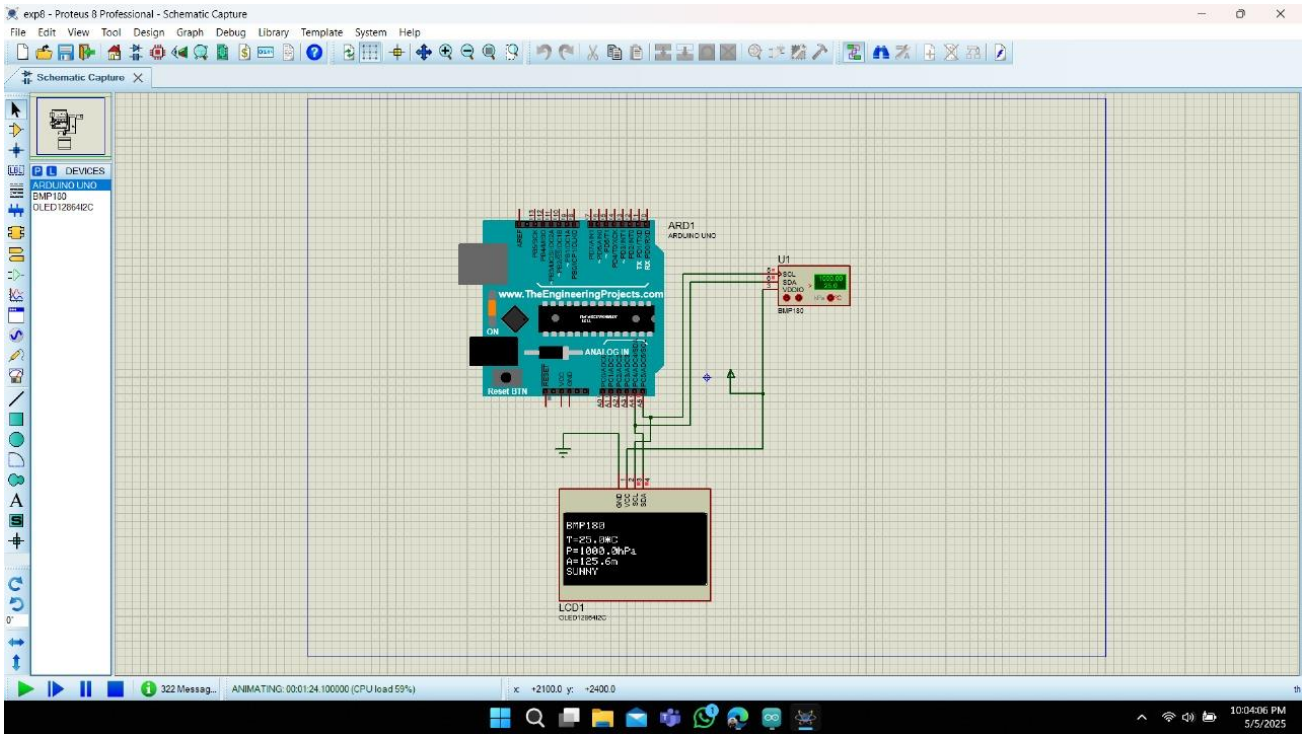

Figure 06: On Proteus BMP180 readings on OLED display (25°C, 125.6m, Sunny).

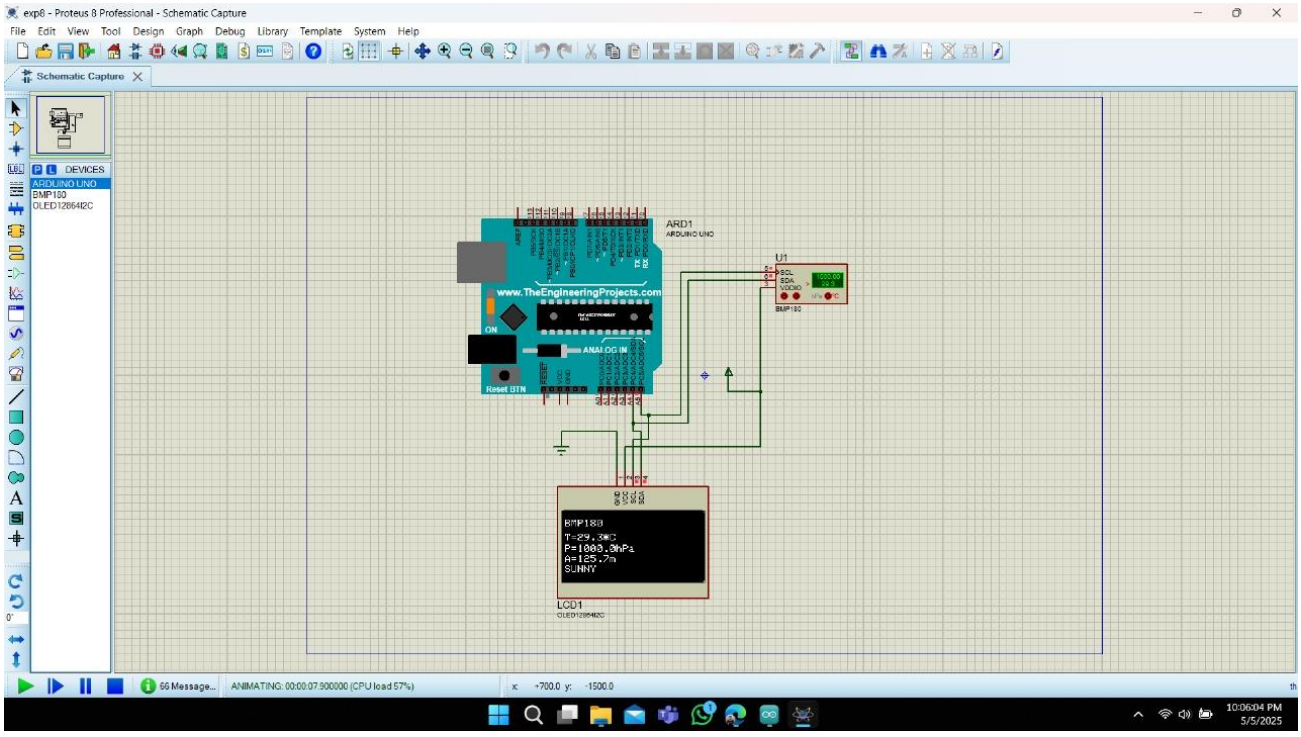

Figure 07: On Proteus BMP180 readings on OLED display (29.3°C, 125.7m, Sunny).

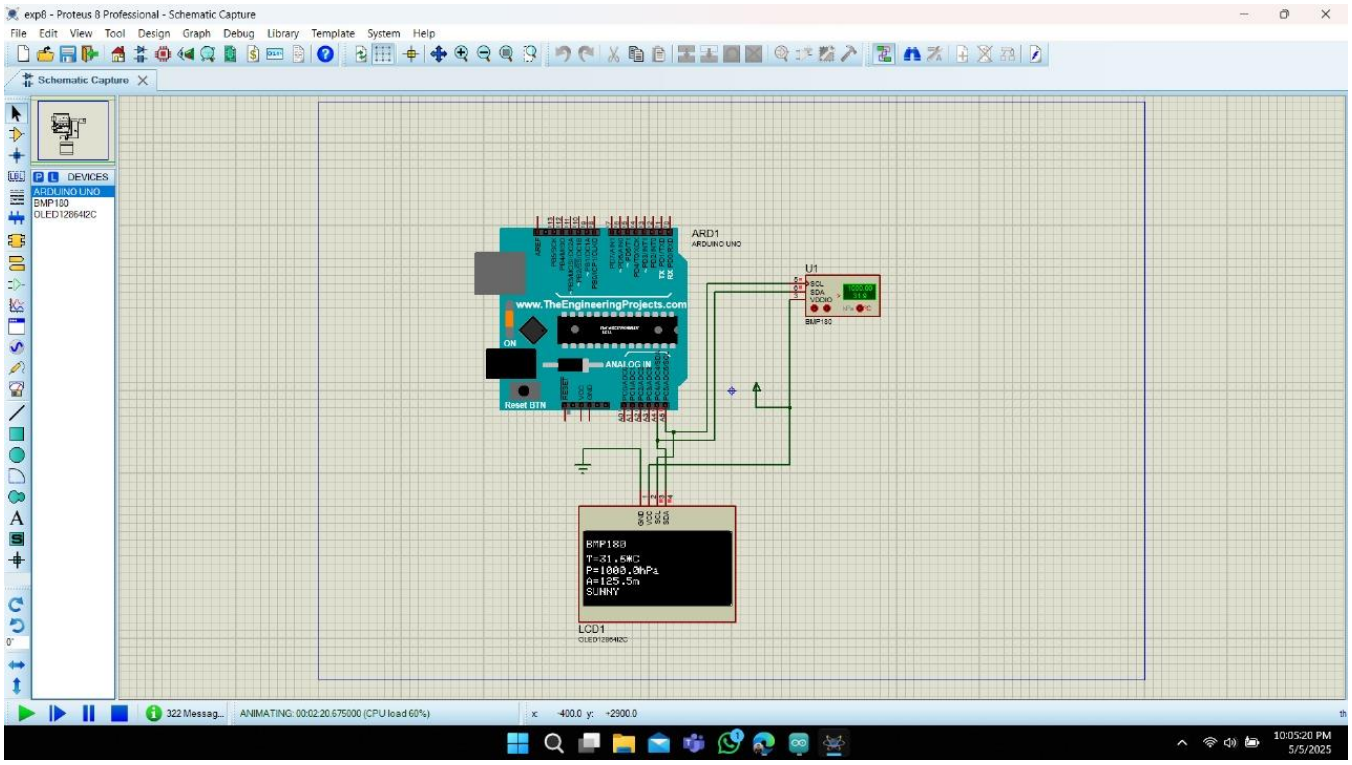

Figure 08: On Proteus BMP180 readings on OLED display (31.6°C, 125.5m, Sunny).

Explanation: In this simulation, the Arduino Uno board,a BMP180 and OLED display were set up according to the hardware configuration used in the lab. The program was written and verified in the Arduino IDE 2.3.5, which generated a HEX file. This file was then loaded into the Proteus simulation to simulate the behavior of the circuit. In the first image, the OLED display shows the initial output from the BMP180 sensor: a temperature of 25°C, pressure of 1000.9 hPa, and an altitude of 125.6 meters, with a weather status labeled as "SUNNY". This indicates standard atmospheric conditions likely associated with clear weather. The second image captures an updated reading as the simulation progresses. Here, the temperature has slightly increased to 29.3°C, while the pressure remains the same at 1000.9 hPa, and the altitude is now reported as 125.7 meters. The weather status continues to be "SUNNY", suggesting that the pressure is still within the range considered for clear weather conditions. In the third image, the sensor data updates once again, now showing a temperature of 31.6°C, pressure still at 1000.9 hPa, and a slightly lower altitude reading of 125.5 meters. The weather status remains consistent as "SUNNY", reinforcing the system's stability in interpreting atmospheric conditions. After running the simulation, the results were observed and compared with the actual hardware results to check for consistency and verify the performance of the system.

Answer to Question:

C) Ans: In this experiment, a weather forecast system was implemented using an Arduino Uno, a BMP180 sensor, and an OLED display module. The system relies on the **I2C communication protocol**, which requires the use of **A4 (SDA)** and **A5 (SCL)** pins on the Arduino board. These two pins are **fixed and reserved for I2C operation** and cannot be changed to other ports. According to the question instructions, we are asked to configure input/output ports based on digits from our student ID within the range of 0–6. From the given ID **23-51259-1**, the valid digits are **1, 2, 3, and 5**. However, since **A4 and A5** are mandatory for I2C communication and do not directly match any two digits from the ID, we are **unable** to apply the port in this case.

Therefore, **due to the fixed nature of the I2C ports A4 and A5, and their absence in the relevant ID digits**, the configuration requirement based on the student ID could not be followed for this specific experiment.

Discussion:

This experiment focused on understanding how environmental data such as temperature, pressure, and altitude can be measured and interpreted using an Arduino-based weather forecast system. The main objective was to explore the use of analog-to-digital conversion (ADC) in conjunction with sensor integration to predict basic weather conditions. For this purpose, a BMP180 sensor module was interfaced with an Arduino microcontroller and an OLED display to visualize real-time data.

In the setup, the BMP180 sensor was connected to the Arduino through I2C communication, allowing efficient serial data exchange. The Arduino continuously read analog signals from the sensor, representing temperature and pressure, and converted these into digital values for processing. The altitude was calculated using the pressure readings, based on a reference sea-level pressure. The OLED display was initialized to present temperature in degrees Celsius, pressure in hectopascals (hPa), and altitude in meters, offering a clear and concise user interface. The weather prediction logic was implemented using a simple comparison between current pressure and a theoretically predicted value at a given altitude. Depending on the difference, the system displayed a basic forecast: “SUNNY,” “SUNNY/CLOUDY,” or “RAINY.”

This hands-on implementation highlighted the role of ADC in embedded systems, where real-world analog signals are digitized for meaningful processing. It also emphasized the importance of real-time data acquisition, sensor calibration, and the integration of display modules to create interactive systems.

Here are some real life scenario and application of this experiment:

1. Smart Agriculture Monitoring Systems:

Weather forecast systems using pressure and temperature sensors help farmers plan irrigation, crop protection, and harvesting based on local atmospheric changes.

Example: An Arduino-based setup in a greenhouse predicts weather changes like rainfall or high temperature, triggering cooling or watering systems automatically.

2. Weather Stations in Remote Locations:

In areas without internet or power infrastructure, low-cost microcontroller-based weather stations collect and log environmental data.

Example: A solar-powered Arduino with a BMP180 sensor logs pressure and temperature data for later analysis by meteorologists or environmental researchers.

3. Personal Weather Monitoring Devices:

Hobbyists and enthusiasts use compact weather stations for real-time local monitoring.

Example: A portable Arduino weather gadget worn during hiking provides information about elevation and forecasts sudden weather changes.

4. Aviation and Drone Safety Systems:

Pressure and altitude sensing help drones adjust their flight and prepare for environmental conditions that may affect stability.

Example: A drone uses onboard sensors to determine if changing pressure could indicate approaching storms, allowing it to return or reroute.

5. Disaster Preparedness Systems:

These systems are vital in predicting and alerting for extreme weather conditions, especially in flood-prone or mountainous areas.

Example: Arduino-based pressure drop monitoring in villages triggers community alerts to prepare for heavy rainfall or landslides.

6. IoT-Based Smart Cities:

Weather sensors can be connected to cloud platforms via Wi-Fi or GSM modules to provide city-wide atmospheric data.

Example: A network of Arduino-based weather nodes uploads live pressure and temperature data to a dashboard for city planning and traffic control.

7. Outdoor Sports and Adventure Gear:

Devices designed for trekkers or campers can include weather prediction features based on local sensor data.

Example: A wearable weather forecaster alerts a hiker if atmospheric pressure suddenly drops, indicating potential storms.

Conclusion:

This experiment successfully demonstrated the implementation of a basic weather forecast system using the ADC modules of an Arduino in conjunction with the BMP180 sensor and OLED display. By measuring key environmental parameters such as temperature, atmospheric pressure, and altitude, the system utilized analog-to-digital data processing to provide real-time feedback. A simple weather prediction model was applied based on pressure variation from the standard atmospheric value, allowing basic conditions like sunny, cloudy, or rainy to be displayed. This practical implementation highlighted the importance of serial and I2C communication protocols, sensor interfacing, and display handling in embedded systems. Furthermore, the experiment strengthened our understanding of how raw sensor data can be interpreted and used for environmental monitoring. It provided valuable experience in writing condition-based logic for real-time applications. Overall, this experiment laid a strong foundation for developing more complex weather monitoring or smart climate control systems in IoT and automation domains.

References:

[1] https://www.arduino.cc/.
[2] https://www.educba.com
[3] https://www.researchgate.net/publication/
[4] https://www.geeksforgeeks.org