AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Engineering

Lab Report

Experiment # 8

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

Date of Perform:	Date Month 2023	**Date of Submission:**	31 July 2023
Course Title:	MICROPROCESSOR AND EMBEDDED SYSTEM LAB		
Course Code:	COE3104	**Section:**	N/A
Semester:	Summer 2022-23	**Degree Program:**	BSc in CSE/EEE
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 and no part of it has been copied from any other student's work or from 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 clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared, and archived for the purpose of detecting plagiarism.
6. I/we give permission for 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 #

Sl No	**Name**	**ID**	**PROGRAM**	**SIGNATURE**

Faculty use only

FACULTY COMMENTS	**Marks Obtained**	
	Total Marks	

Table of Contents

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

Objectives:

The objectives of this experiment are to-

- Familiarize with the Micro-controller-based weather forecast system.
- Implement the environmental parameters, such as temperature, pressure and humidity.

Equipment List:

1) Arduino UNO board
2) BMP180/MPL115A
3) Inches96 inch OLED 128×64
4) Breadboard
5) Jumper Wires.

Circuit Diagram:

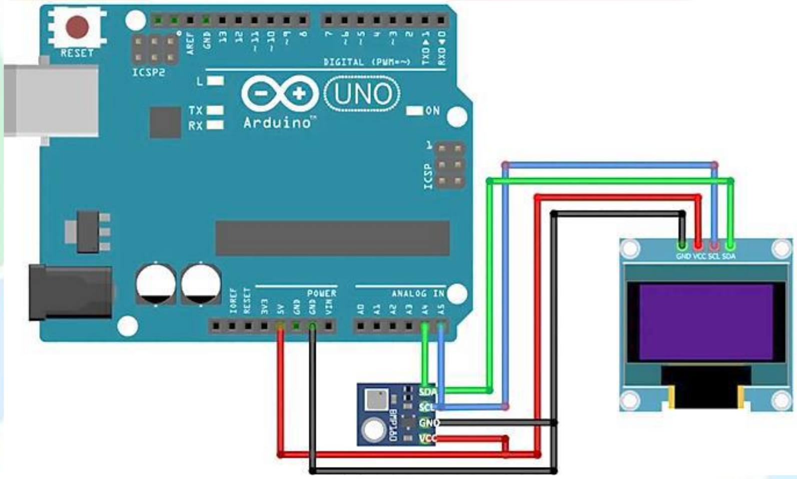


Fig. 1 Experimental setup of a weather forecast system using the ADC modules of an Arduino

Code/Program:

The following is the code for the implementation of a weather forecast system using the ADC modules of an Arduino with the necessary code explanation:

```
#include <SPI.h> //for communicating with devices using SPI protocol
#include <Wire.h> //for communicating with devices using I2C protocol
#include <Adafruit_GFX.h> //for drawing graphics on the OLED display
#include <Adafruit_SSD1306.h> //for controlling the OLED display
#include <Adafruit_BMP085.h> //for reading data from BMP085 sensor

//The following lines define the width and height of the OLED display in
pixels
```

```
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

//The following line creates an object display of type Adafruit_SSD1306 with
the specified screen width and height
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT);

//The following line creates an object bmp of type Adafruit_BMP085
Adafruit_BMP085 bmp;

// The following line defines a constant SEALEVELPRESSURE_HPA with a value of
101500 hPa, which is the average sea level pressure in atmospheric pressure
units
#define SEALEVELPRESSURE_HPA (101500)

//The following line declare four variables of type float which will be used
later in the code
float simpleweatherdifference, currentpressure, predictedweather,
currentaltitude;

//In the setup function, the microcontroller checks if the BMP sensor is
properly set or not. If not set properly, the program warns the user through
the serial monitor about it.
void setup() {
 display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
 if (!bmp.begin()) {
 Serial.println("Could not find a valid BMP085 sensor, check wiring!");
 while (1) {}
 }
}

void loop() {
display.clearDisplay(); // Clears the OLED display
display.setTextSize(1); // Sets the text size
display.setTextColor(SSD1306_WHITE); // Sets the text color

display.setCursor(0,5); // Sets the cursor position
display.print("BMP180"); // Prints the sensor name
display.setCursor(0,19); // Sets the cursor position for the next print
//The following lines prints temperature in Celsius units
display.print("T=");
display.print(bmp.readTemperature(),1);
display.println("*C");

//The following lines prints the pressure in hectopascal units
display.setCursor(0,30); // Sets the cursor position
display.print("P=");
display.print(bmp.readPressure()/100.0F,1);
display.println("hPa");

//The following lines prints the altitude in meters
display.setCursor(0,40); // Sets the cursor position
display.print("A=");
display.print(bmp.readAltitude(SEALEVELPRESSURE_HPA),1);
display.println("m");
delay(6000); //Delays for 6 seconds
display.display();

//The following line reads the current pressure in Pascals from the BMP180
sensor, converts it to hectopascals and stores it in the variable
"currentpressure"
currentpressure=bmp.readPressure()/100.0;

//The following line calculates the predicted weather based on the current
altitude and the standard atmospheric pressure at sea level (101.3 hPa). It
uses the barometric formula, which describes how atmospheric pressure
decreases with altitude. The resulting value is stored in the variable
"predictedweather"
predictedweather=(101.3*exp(((float)(currentaltitude))/(-7900)));

//The following line calculates the difference between the current pressure
and the predicted weather by subtracting the predicted weather from the current
pressure. The resulting value is stored in the variable
"simpleweatherdifference"
simpleweatherdifference=currentpressure-predictedweather;

display.setCursor(0,50); //// Sets the cursor position

//If the value of "simpleweatherdifference" is greater than 0.25, this line
prints the word "SUNNY" on the OLED screen
if (simpleweatherdifference>0.25)
 display.print("SUNNY");

//If the value of "simpleweatherdifference" is less than or equal to 0.25,
this line prints the words "SUNNY/CLOUDY" on the OLED screen
if (simpleweatherdifference<=0.25)
 display.print("SUNNY/CLOUDY");

//If the value of "simpleweatherdifference" is less than -0.25, this line
prints the word "RAINY" on the OLED screen
if (simpleweatherdifference<-0.25)
 display.print("RAINY");
display.display(); //This line updates the OLED screen with the text that was printed
delay(2000); //Delays for 2 seconds
}
```

Hardware Output Results:

Here is the hardware implementation of a weather forecast system using the ADC modules of an Arduino and the necessary explanation of the implementation:

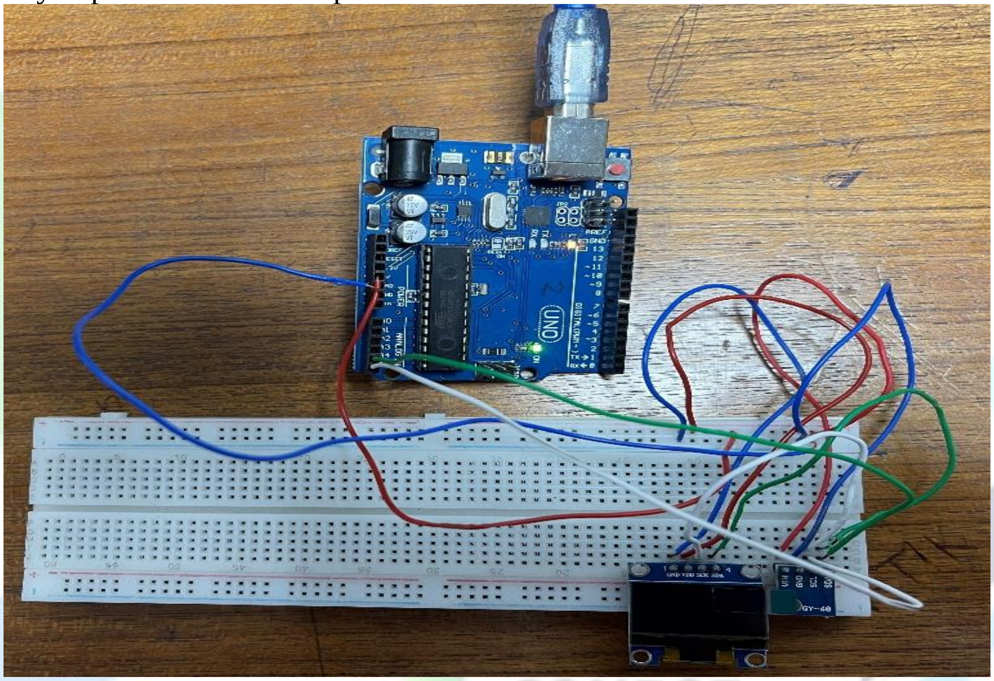


Fig. 2 Hardware implementation of a weather forecast system using the ADC modules of an Arduino

Explanation:

In the experiment, an Arduino Uno board was taken to perform the experiment. The OLED LED and the BMP180 sensor was set on the breadboard. Wires were connected with the 5V and GND port on the Arduino Uno with the following ports: V_{cc} and GND of the OLED and BMP180 Sensor. The SDA of the BMP180 sensor and the OLED was connected with A4 port of the Arduino Uno. The SCL of the BMP180 sensor and the OLED was connected with A5 port of the Arduino Uno.

Experimental Output Results:

Here are the results of a weather forecast system using the ADC modules of an Arduino and the necessary explanation of the results:

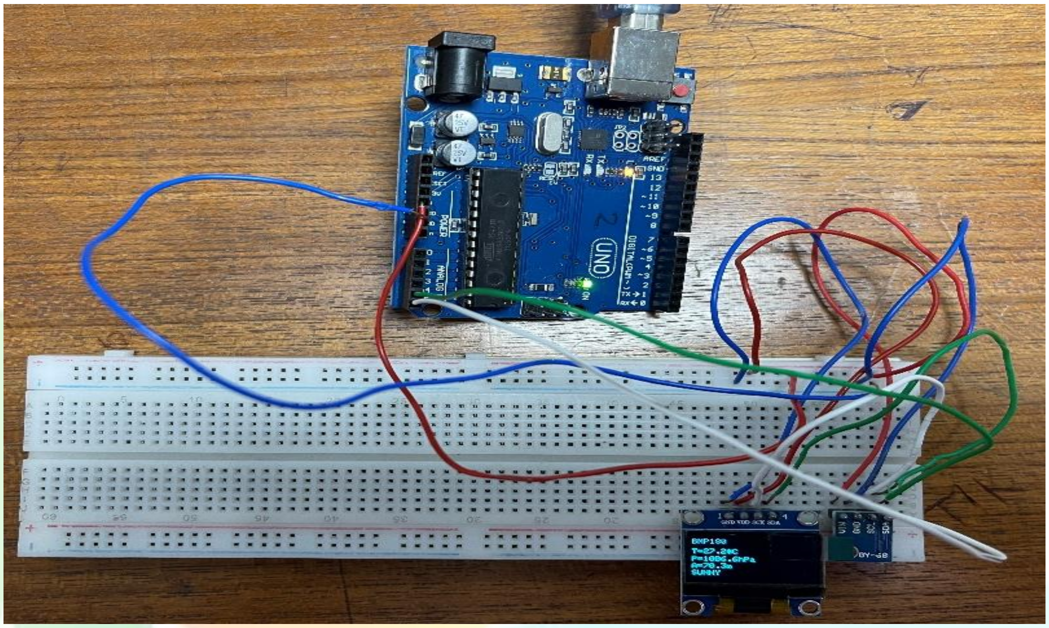


Fig.3 First Readings where Temperature was 27.2°C, Altitude was 70.3m and SUNNY condition

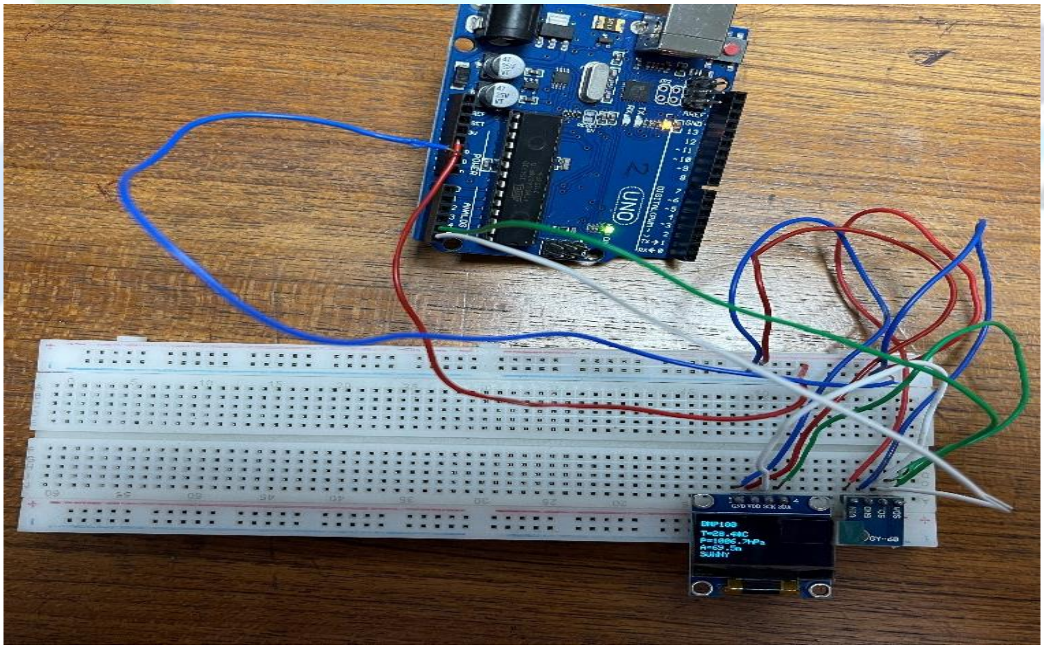


Fig.4 Second Reading where Temperature was 28.4°C, Altitude was 69.5m and SUNNY condition

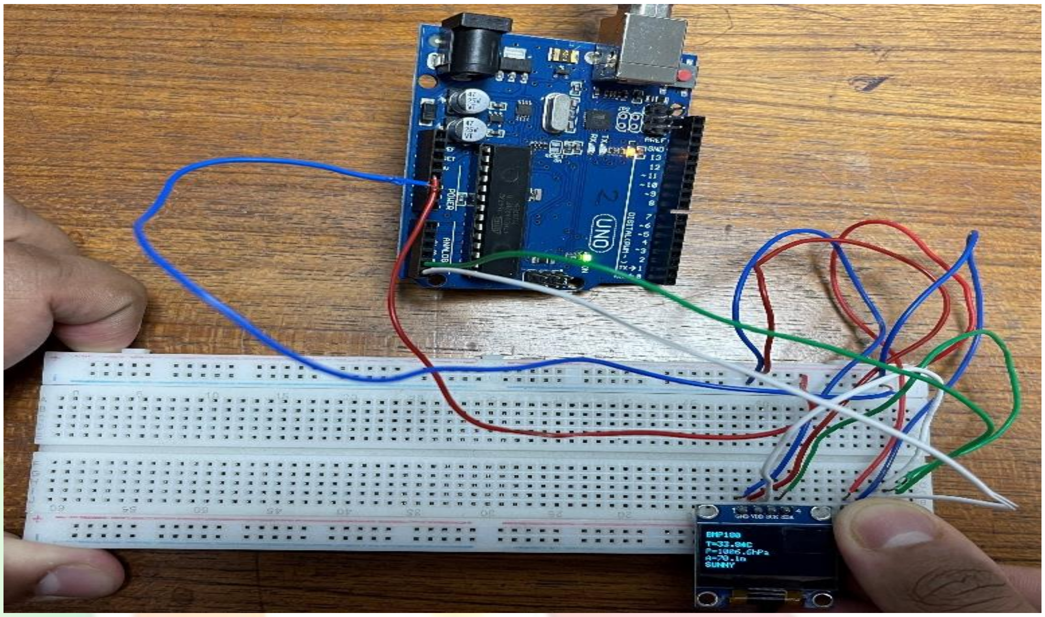


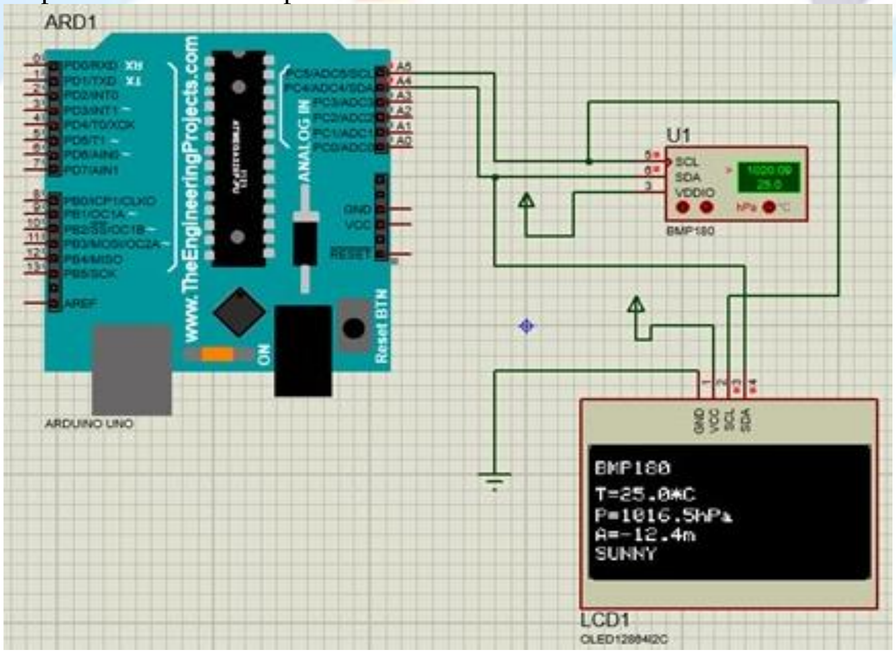
Fig.5 Third Reading where Temperature was 33.9°C, Altitude was 70.1m and SUNNY condition

Explanation:

In the experiment, the BMP180 sensor detected the temperature of the room and presented the data of the room accordingly. As the sensor was lifted upward, the value of the altitude changed accordingly. By increasing the temperature, the value of the temperature changed accordingly as well. Due to the unchanged weather conditions, the values of the weather could not be changed.

Simulation Output Results:

Here is the simulation implementation of a weather forecast system using the ADC modules of an Arduino and the necessary explanation of the implementation:



```
BMP180
T=25.0*C
P=1016.5hPa
A=-12.4m
SUNNY
```

Fig.6 First Simulation where Temperature was 25.0°C, Pressure was 1016.5 hPA, Altitude was 12.4m and SUNNY condition

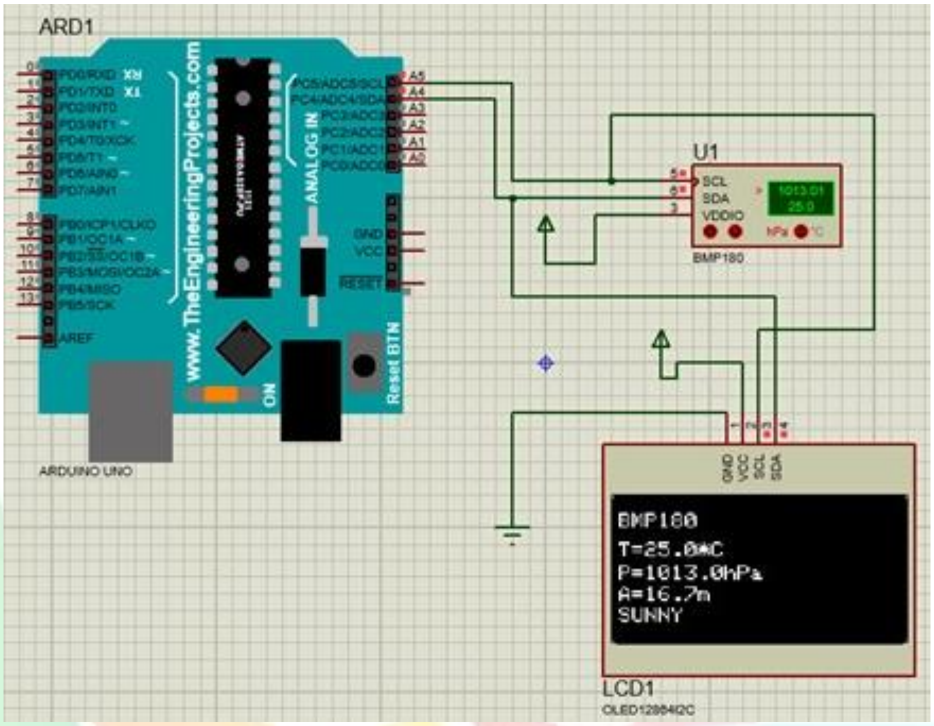


Fig.7 Second Simulation where Temperature was x°C, Pressure was xx hPA, Altitude was ym and SUNNY/CLOUDY condition

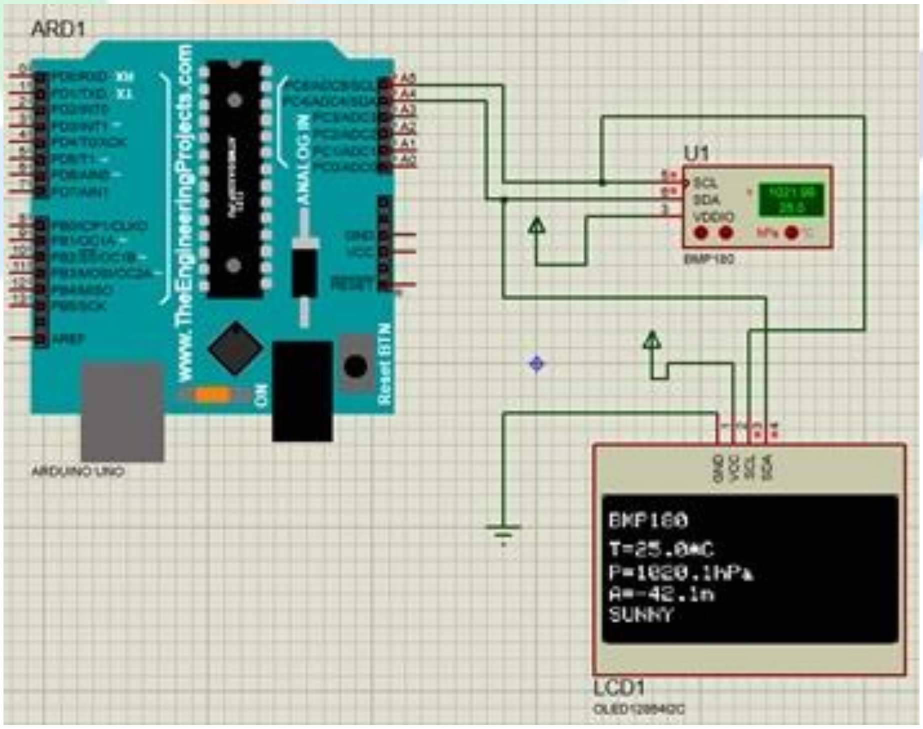


Fig.8 Third Simulation where Temperature was x°C, Pressure was xx hPA, Altitude was ym and RAINY condition

Explanation:

In this experiment, the simulations are done using Proteus software. A new project in Proteus was created. Arduino UNO board was added to the schematic design window from the components library. Then, the OLED was placed in the design from the component library. Also, the BMP180 component was added in the design from the component library. Next, the ground pins of the Arduino board, BMP180 sensor and OLED were connected together. The V_{cc} pin of the BMP180 sensor and OLED were connected to the %V pin of the Arduino Uno board. Analog input A4 of the Arduino board was connected to the output pin of the BMP180 sensor and OLED. Analog input A5 of the Arduino board was connected to the output pin of the BMP180 and OLED. The program was written on the Arduino IDE and the HEX file was generated while compiling. The HEX file was imported in the Arduino Uno board in the Proteus simulation. Lastly, the simulation was observed and the results were obtained accordingly. The temperature, pressure and altitude were changed accordingly to observe the changed results as well.

Answers to the Questions in the Lab Manual:

3) Configure the port numbers for outputs and inputs according to your ID. Consider the matched two digits from your ID (if your ID is XY-PQABC-Z then consider An ports from your ID, where n must match with any digits from 0-6). Include all the programs and results within your lab report.

Solution: The following university ID was used:

2	0	-	4	2	5	5	7	-	1
X	Y	-	P	Q	A	B	C	-	Z

SDA Pin = 4
SCL Pin = 5

The following is the code for an obstacle detection system using Arduino with the necessary code explanation:

The Modified Code:

```
#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);

#define CUSTOM_SDA_PIN 4
#define CUSTOM_SCL_PIN 5

Adafruit_BMP085 bmp;
```

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

void setup() {
  Wire.begin(CUSTOM_SDA_PIN, CUSTOM_SCL_PIN);
  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);
 }
```

Note:

The simulation of the above code could not be generated as a custom ‘wire’ library file was required for to compile the code. Due to lack of time, the ideal library could not be found from the internet. The results of the could would be similar to the previous results. The only change could have been that the SDA and SCL of BMP180 sensor and OLED would have connected to pin 4 and 6 accordingly.

Discussion:

In this experiment, a BMP180 sensor was used to detect the temperature, pressure and altitude and the information collected from the sensor were represented on an OLED display which were connected through ADC connection. The BMP180 sensor that was used in this experiment were studied carefully before using it. The pin operations and how the BMP180 works were observed and carefully understood. After that, the BMP180 sensor was set accordingly with the Arduino Uno board accordingly. The OLED display was set accordingly as well. After that, the system was operated and the systems operations were observed. The method of how the weather were determined by detecting the temperature and the pressure were observed. The results were turning showing accordingly based on the formulas that were set on the code. All the results that were observed were carefully noted down for further evaluation. The similar system was developed on the simulation softwares like Proteus. The results that were obtained on the physical operation were evaluated with the simulated outcomes. There were some minor discrepancies that were observed. The distance that were generated on the simulation’s serial monitor were a bit different compared to the ones that were observed on the physical testing. This might be caused due to minor system and human errors. This caused the inconsistencies in the values of the serial monitor. Moreover, the detecting rate of the system in the physical environment and simulation virtual environment were a bit different. This was ruled as normal as human error was general in the physical world. From the observation it can be said that after both hardware and software implementation showed the expected outcomes and the experimental objectives was achieved.

References:

1) BMP180 Datasheet, [Online] [Cited: July 29, 2023] Available: https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf

2) Arduino CC Website, [Online] [Cited: July 29, 2023] Available: https://docs.arduino.cc/hardware/uno-rev3

3) *Interface BMP180 Barometric Pressure & Temperature Sensor with Arduino*, Last Minute Engineers, [Cited: July 29, 2023] Available: https://lastminuteengineers.com/bmp180-arduino-tutorial/

4) AIUB Microprocessor and Embedded Systems Lab Manual 8