American International University- Bangladesh

Department of Computer Science

<u>Lab Report Cover Sheet</u>

Course Name	MICROPROCESSOR AND EMBEDDED SYSTEMS
Lab Report No.	07
Lecturer Name	**MD. ALI NOOR**
Semester	SUMMER 2021-22
Submission Date	25/07/2022
Section	O
Group No.	03

Student Name	**Student ID**	**Contribution (out of 100%)**
1. EMON, MEHEDI HASAN	19-41118-2	
2. ASIF AHAMMED	19-40020-1	
3. EASINUL ABEDIN SAYEM	19-40291-1	
4. MD REDOY SHEIKH	19-40101-1	
5. MD. NAZIM HASAN	19-40118-1	
6. MOHI UDDIN ANANDO	20-43585-1	

Title: Using the ADC modules in Arduino for the Implementation of a weather forecast system

Abstract

The objective of this experiment is to get familiarized with Micro-controller based weather forecast system and environmental parameters (Temperature, Pressure and Humidity) measurement.

Theory and Methodology:

Weather Prediction

The BMP180 or MPL115A is an absolute device that can be used to predict and measure the barometric pressure to deduce weather patterns. Weather prediction requires a static location for the sensor and 2-3 hours to analyze a full weather pattern. Typically, the pressure changes due to weather are slow, requiring a few hours to determine the sloping of the pressure change. Vertical movement or a significant airflow can interfere with results due to only weather patterns in barometric pressure. The sensor should be kept in a relatively protected area from any strong air flows, and kept at that static location during analysis. Temperature effects can change the results of a normal pressure sensor especially if the measurement is done over several hours in varying temperature. Due to the nature of the calibration and temperature compensation, MBP180 meets these requirements, compensating for temperature swings over a large 0 to 85°C operating range. It will not require auto-zeroing for shifts in offset or span over temperature.

How Pressure Increases and Decreases with Weather

For weather pattern prediction, the BMP180 or MPL115A is a well-suited device with its pressure range and resolution. Barometric pressure changes can directly correlate to changes in the weather. Low pressure is typically seen as the precursor to worsening weather. High pressure increases can be interpreted as improving or clear weather. The typical reasoning can be seen in a comparison of molecular weights. If air is approximately 21% O2(g), 78% N2(g), O2(g) has a molecular mass of 32, N2(g) has a mass of 28. H2O (g) has a molecular mass of 18. So if there is a large amount of water vapor present in air, this air is going to be lighter than just regular dry air by itself. It’s an interesting fact that explains how weather patternslead to high or low pressure. If bad weather originates in an area in the formation of water-vapor clouds, this is falling pressure on a barometer. The vapor will reduce the barometric pressure as the H2O reduces the mass above that point on the earth. High pressure will signal the clearing of the water vapor as the air dries. Another quandary is how weather during severe hurricanes/cyclones with high 150 mph winds be defined as low pressure? This is due to the fact that hurricanes are low pressure conditions surrounded by higher pressure. The rush of air from higher to low pressure createsthe fast-moving winds. The lower the pressure in the center, the greater differential pressure between high and low areas. This leads to a stronger cyclone or hurricane. Some areas are harder to predict weather patterns. Cities located at the base of mountainous regions where condensation and fog are a daily occurrence is an example. An area like Hawaii where high colder mountains meet low warm sea regions can have harder to predict results. A network of sensors can give a more exact trend, but for a single sensor in a static location, there are a few ways to have a simple standalone weather station.

Local Weather Stations

When implementing a weather station, it is best to will check results with a local forecast. When researching local weather pressures, such as barometric pressure at the closest airport, remember that the weather is normalized for altitude. Normalization takes local barometric pressure and shifts it to reflect sea level altitude. Sea Level is 101.3 kPa, and by normalizing various points on a map, a meteorologist can see the weather pattern over a region. Without the normalization, the effect of altitude on the pressure reported by collection points will lead to useless data. A mountain data point will have pressure affected by altitude and as it leads to the valleys, the pressure point there will be higher, Airports are typical reporting stations to check barometric pressure. Some display only normalized pressure during a web search. This is such that a pilot landing at any airport can deduce the weather conditions by knowing the barometric pressure. If the airport is located at the beach, or on a mountainous region, normalization of this value removes the barometric variation due to the altitude. It standardizes pressure so that weather patterns can be mapped.

Equipment List

- Arduino Uno Board
- DHT11
- BMP180 / MPL115A
- 0.96 inch OLED 128X64
- Breadboard and Jump Wires

<u>Hardware implementation:</u>

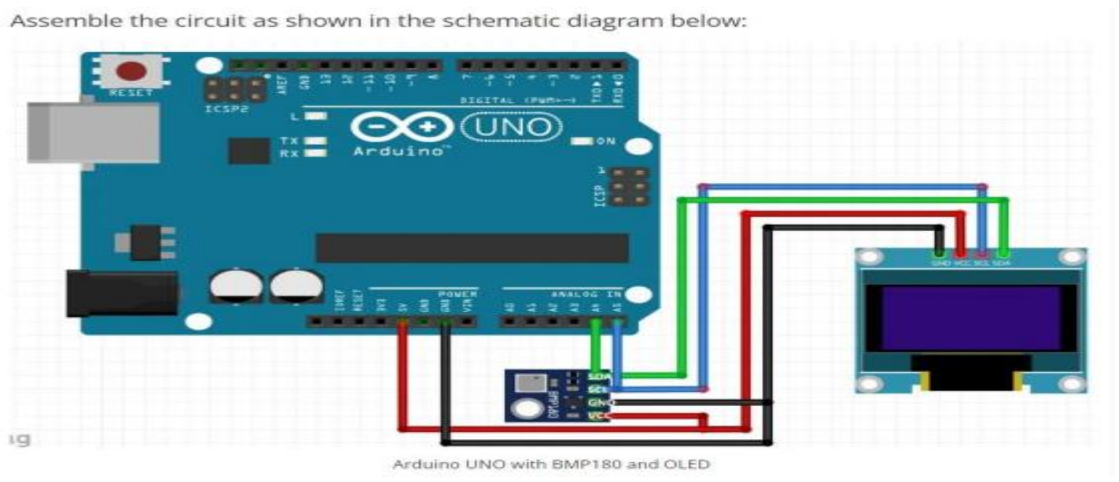

Assemble the circuit as shown in the schematic diagram below:

Figure 1: Aurdino UNO with BPM180 and OLED

Experimental Procedure:

Start by creating a new Visual Designer Project and then Add the BMP180 and DHT11 sensor. OLED display of 128X64 is added for visual presentation.

Hardware Set-Up

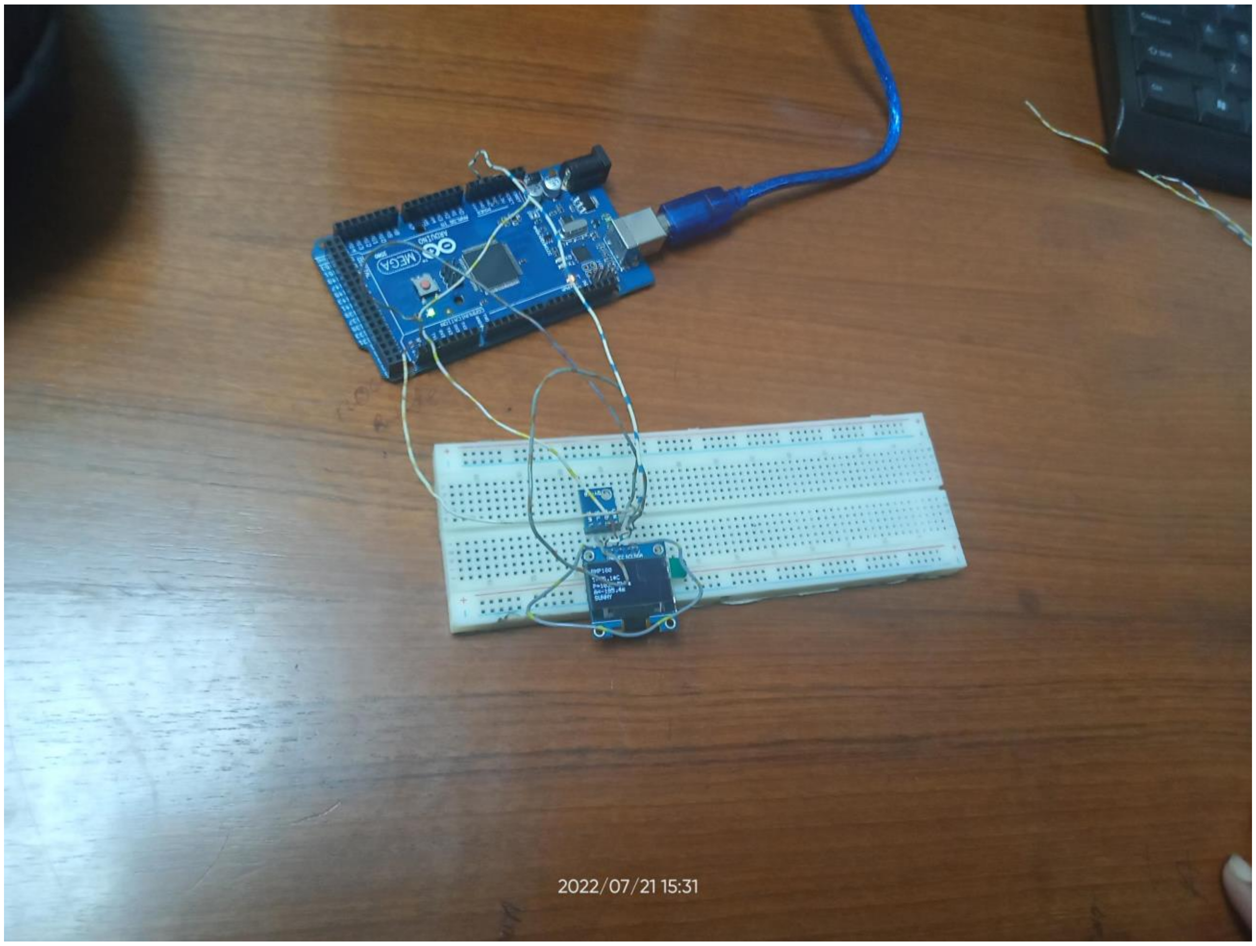


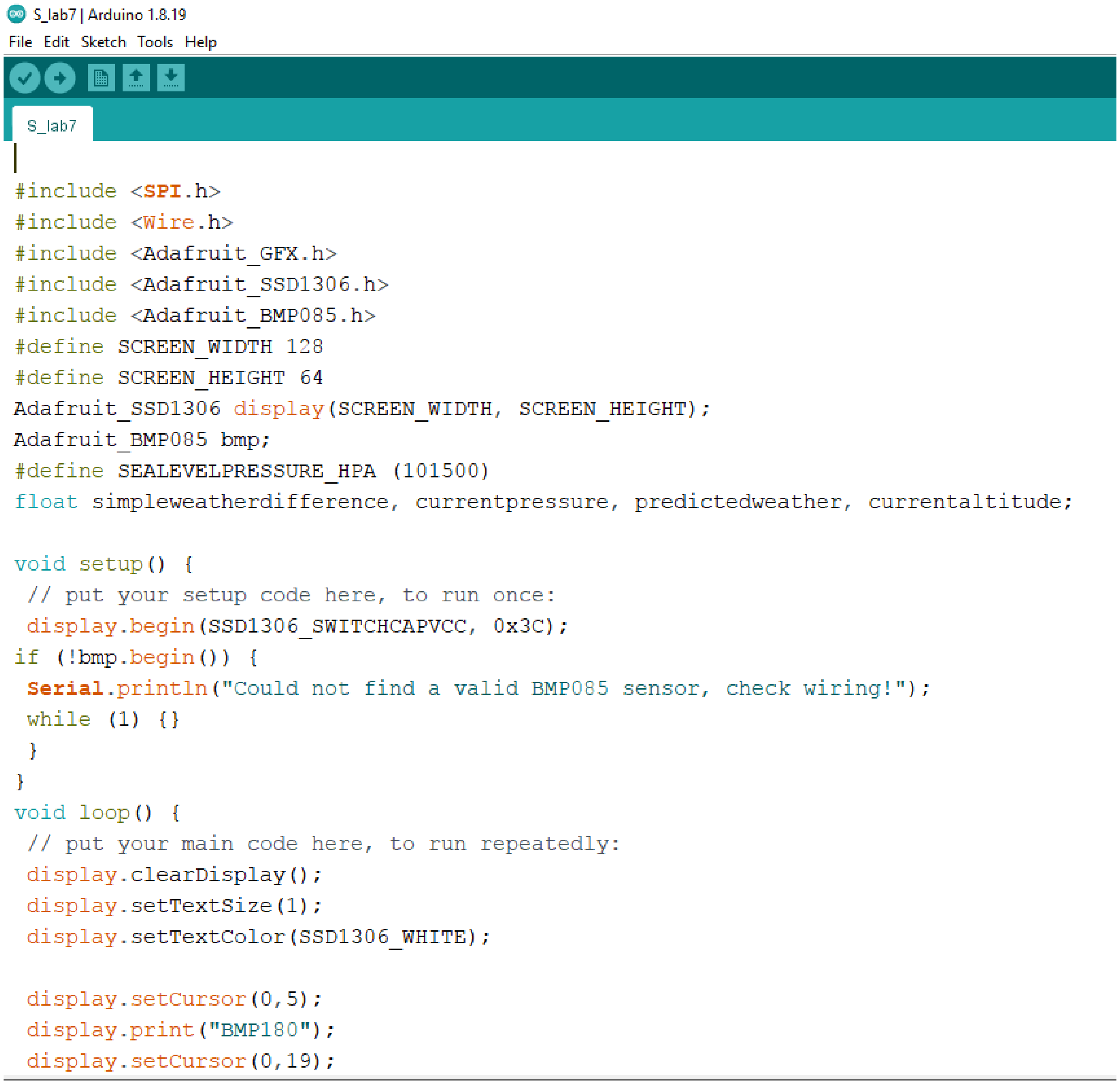
Figure : Lab Work

Program/Lab Code:

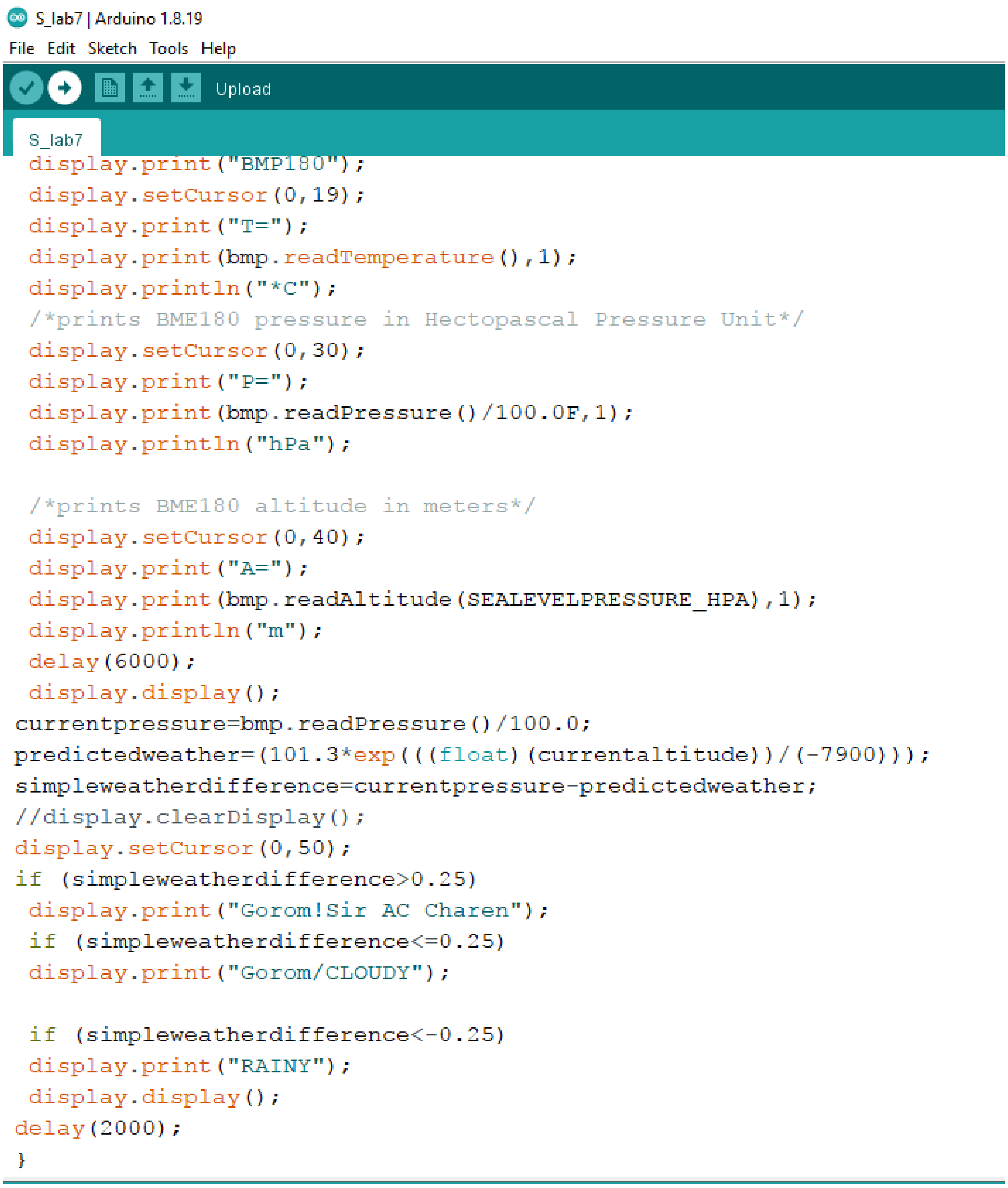
```
#include <SPI.h>

#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);
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(20,19);
 display.print("T=");
 display.print(bmp.readTemperature(),2);
 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(3000);
 display.display();
currentpressure=bmp.readPressure()/100.0;
predictedweather=(101.3*exp(((float)(currentaltitude))/(-7900)));
simpleweatherdifference=currentpressure-simpleweatherdifference;
//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);
```

}

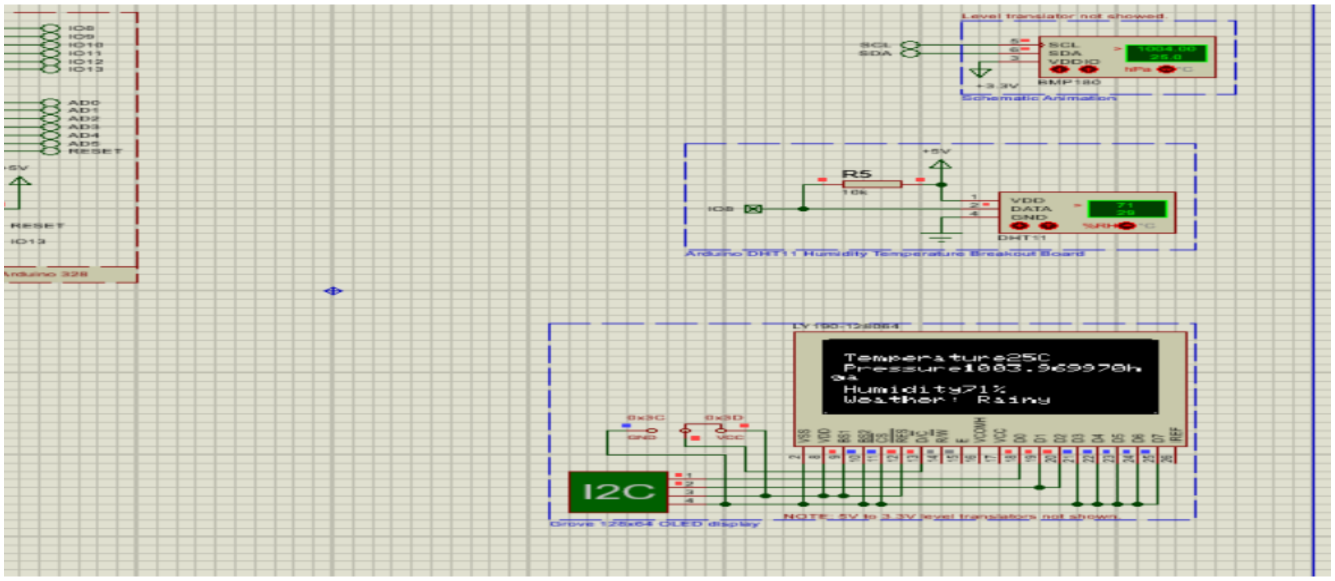


```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_BMP085.h>
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT);
Adafruit_BMP085 bmp;
#define SEALEVELPRESSURE_HPA (101500)
float simpleweatherdifference, 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("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("Gorom!Sir AC Charen");
 if (simpleweatherdifference<=0.25)
 display.print("Gorom/CLOUDY");

 if (simpleweatherdifference<-0.25)
 display.print("RAINY");
 display.display();
delay(2000);
}
```

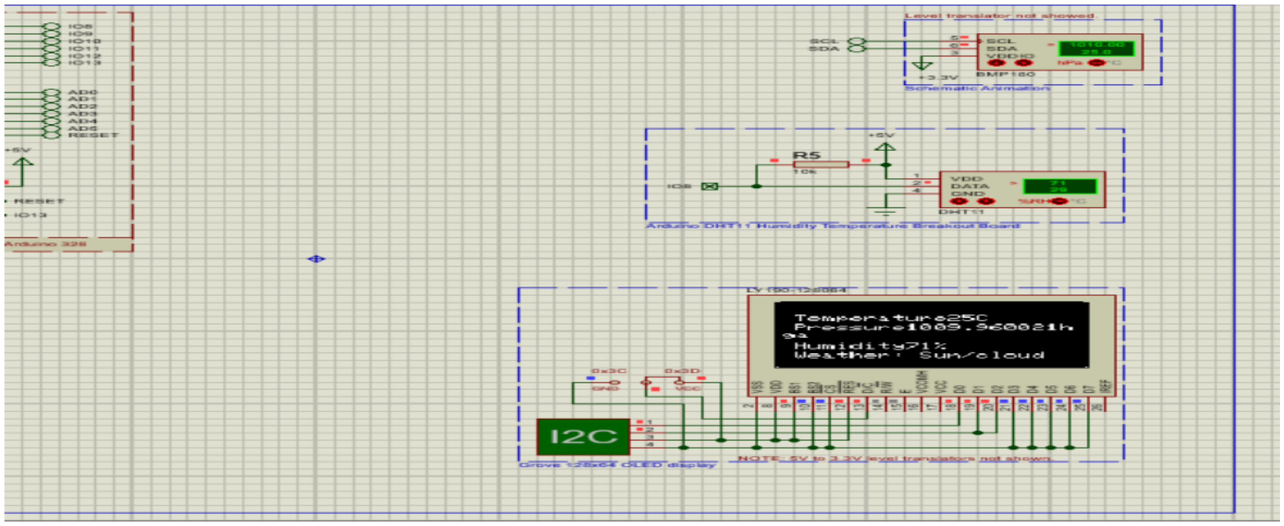
Simulations:



```
Temperature25C
Pressure1003.969970h
Pa
Humidity71%
Weather: Rainy
```

Level translator not showed
SCL
SDA
BMP180
+3.3V
Schematic Animation
R5
10k
+5V
IO8
VDD
DATA
GND
DHT11
Arduino DHT11 Humidity Temperature Breakout Board
LY190-128064
I2C
0x3C
0x3D
GND
VCC
NOTE: 5V to 3.3V level translators not shown.
Grove 128x64 OLED display
RESET
IO13
Arduino 328

Figure 3: Rainy weather



```
Temperature25C
Pressure1009.960021h
Pa
Humidity71%
Weather: Sun/cloud
```

Level translator not showed
SCL
SDA
BMP180
+3.3V
Schematic Animation
R5
10k
+5V
IO8
VDD
DATA
GND
DHT11
Arduino DHT11 Humidity Temperature Breakout Board
LY190-128064
I2C
0x3C
0x3D
GND
VCC
NOTE: 5V to 3.3V level translators not shown.
Grove 128x64 OLED display
RESET
IO13
Arduino 328

Figure 4: Sun/Cloud weather

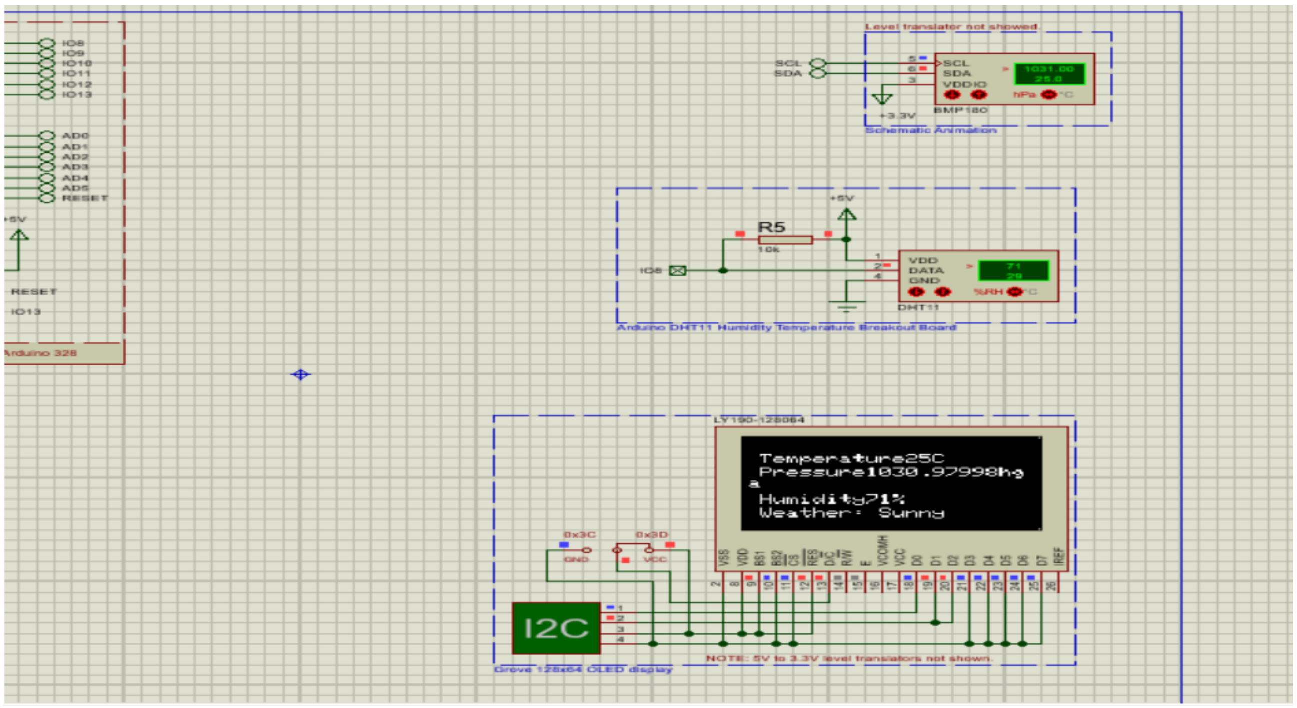
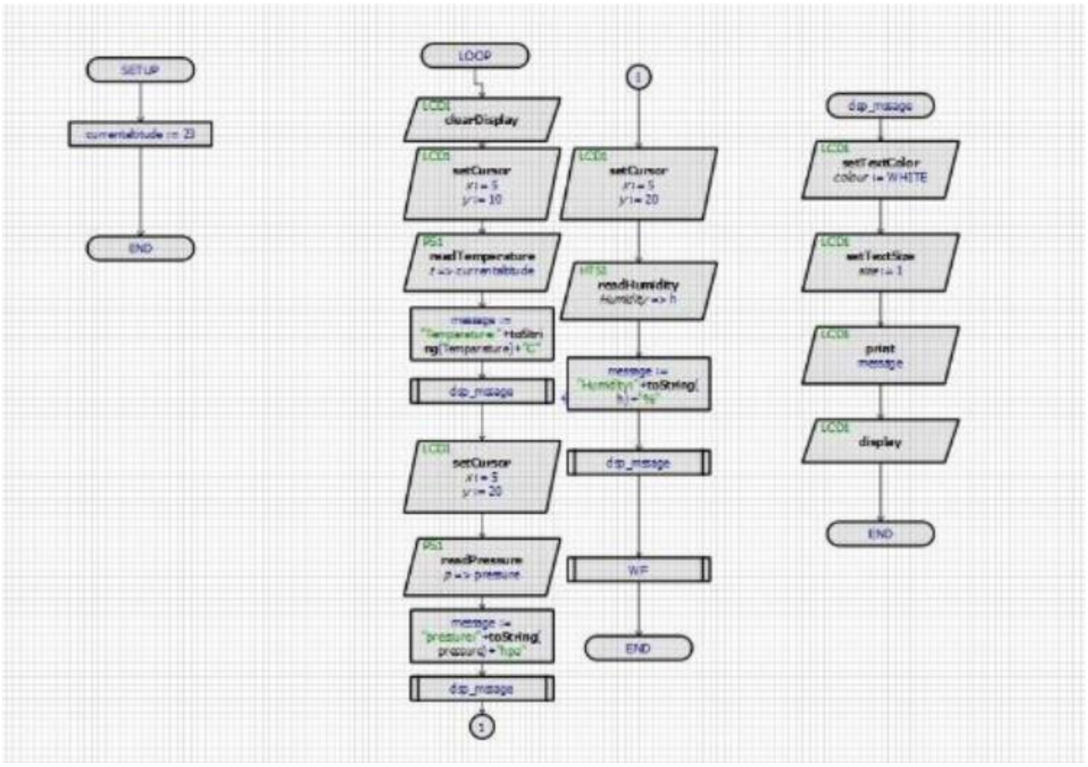


Figure 5: Sunny weather

Flowchart for Proteus Visual Designer:



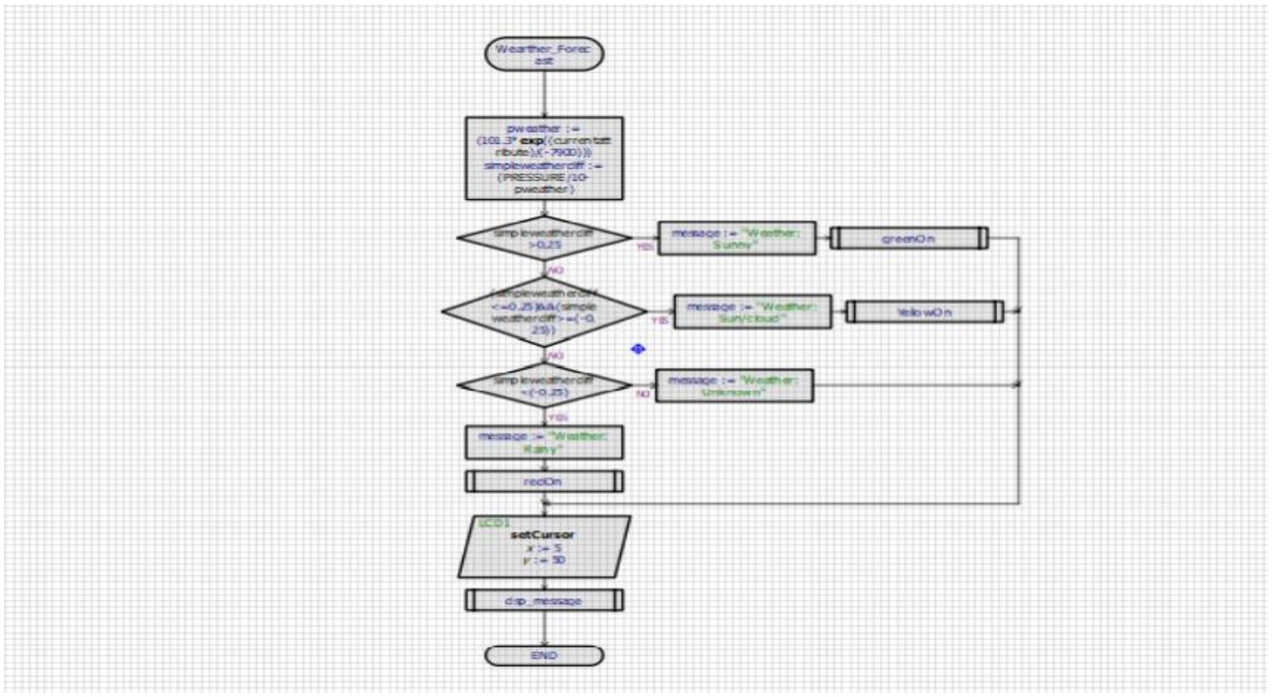

Weather_Forecast
pweather := (101.3* exp((currentattribute)/(-7900)))
simpleweatherdiff := (PRESSURE/10-pweather)
simpleweatherdiff >0.25
YES
message := "Weather: Sunny"
greenOn
NO
(simpleweatherdiff <=0.25)&&(simpleweatherdiff>=(-0.25))
YES
message := "Weather: Sun/cloud"
YellowOn
NO
simpleweatherdiff <(-0.25)
NO
message := "Weather: Unknown"
YES
message := "Weather: Rainy"
redOn
LCD1
setCursor
x := 5
y := 50
disp_message
END

Figure: Flow Chart

Discussion: Using the pressure and heat in Proteus, this experiment enabled us to develop a weather forecasting system. The weather will be bright if the pressure is larger than 0.25 kPa, wet if the pressure is greater than 0.25 kPa, and so forth, according to our instructor's discussion of the weather analysis system.It will be bright or overcast if the pressure is between -0.25 kPa and +0.25 kPa. Afterward, the required apparatus from the apparatus portion was inserted in the schematic capture. We implement the method in proteus in visual designer using flowchart and used necessary attributes to create the design.The simulation was ran, and Fig. 1 shows the outcome. However, another issue was that executing the simulation flowchart automatically erased it, making it more difficult to recover since proteus doesn't support it in version 8.6.

Conclusion: Despite various setbacks, we were able to obtain the desired outcome, and the experiment was successfully carried out inside the proteus. Due to the success of the experiment, the weather prediction system was created utilizing an Arduino UNO, Weather Station Shield, DHT11 Sensor, and flowchart.