AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH (AIUB)
Faculty of Engineering
Department of EEE and CoE
Undergraduate Program

Course: Microprocessor and Embedded Systems **Summer 2021-22, Final**

Experiment 7: Using the ADC modules in Arduino for the Implementation of a weather forecast system

Submitted By:

Name	ID
MD SHAMIM SIDDIKY	20-42649-1
WASIUDDIN	15-30843-3
TANJIM HOSSAIN	18-36964-1
MD. SAKIB HOSSAIN RIJON	19-39460-1
SADIA AFRIN ETY	19-39659-1
FERDOUS SUNY	19-40485-1
DIPONKAR SUTRA DHAR	19-41004-2

Section: O
Group: 6

Submitted To

Md Ali Noor
Lecturer
Faculty of Computer Engineering
American International University-Bangladesh

Theory and methodology:

The BMP180 or MPL115A is an absolute device that can be used to predict and measure 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 at varying temperatures. 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, and 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 the air, this air is going to be lighter than just regular dry air by itself. It's an interesting fact that explains how weather patterns lead 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? because hurricanes are low-pressure conditions surrounded by higher pressure. The rush of air from higher to low pressure creates fast-moving winds. The lower the pressure in the center, the greater the differential pressure between high and low areas. This leads to a stronger cyclone or hurricane. Some areas are harder to predict weather patterns. An example is cities located at the base of mountainous regions where condensation and fog are a daily occurrence. 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 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, telling nothing about the weather without the normalization.

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 in 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.

Objectives:

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

Equipment List:

1)Arduino Uno Board
2)BMP180 / MPL115A
3) 0.96-inch OLED 128X64
4) Breadboard and Jump Wires

Circuit Diagram:

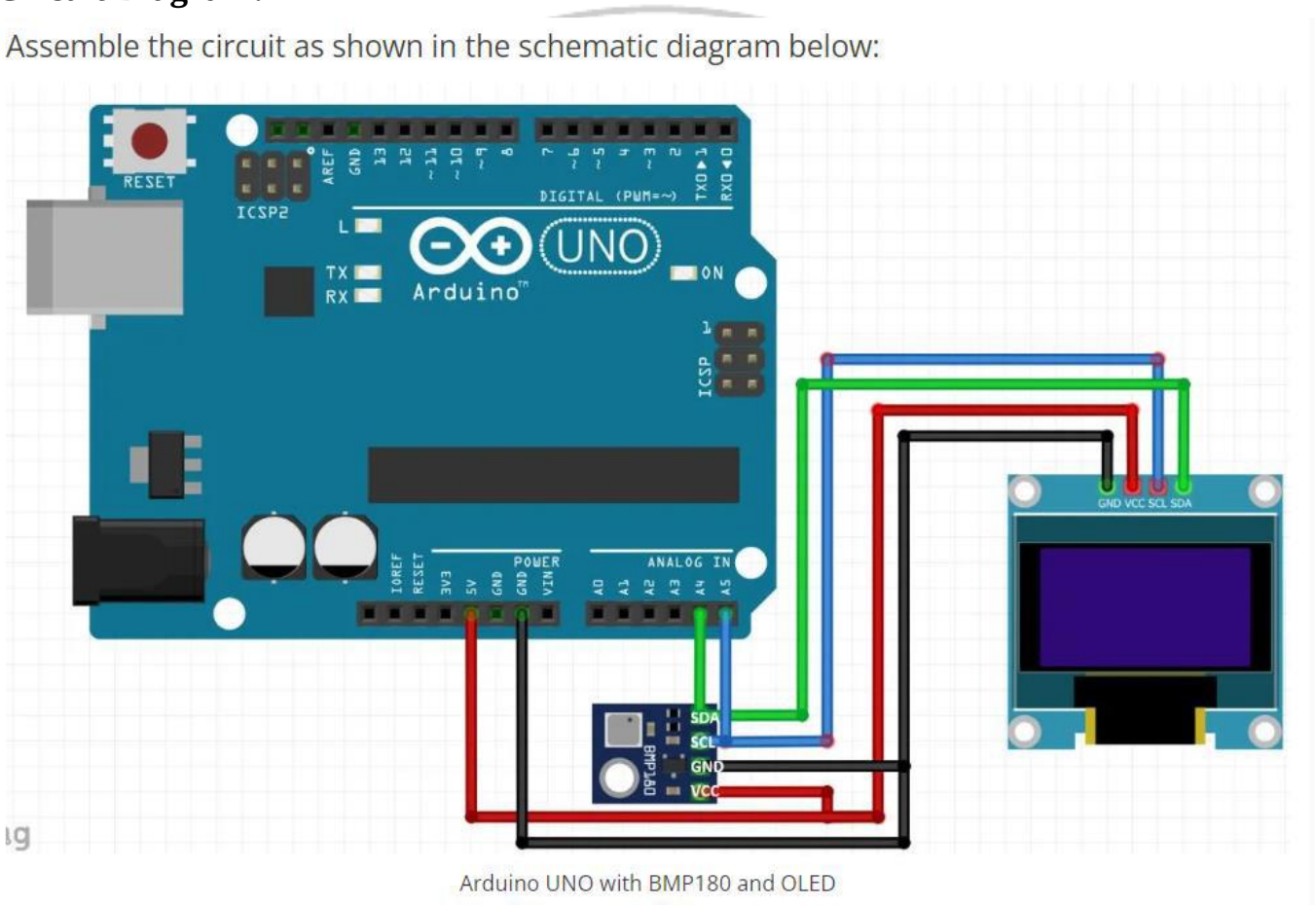

Fig 1: Circuit diagram for weather forecast system

Hardware Setup:

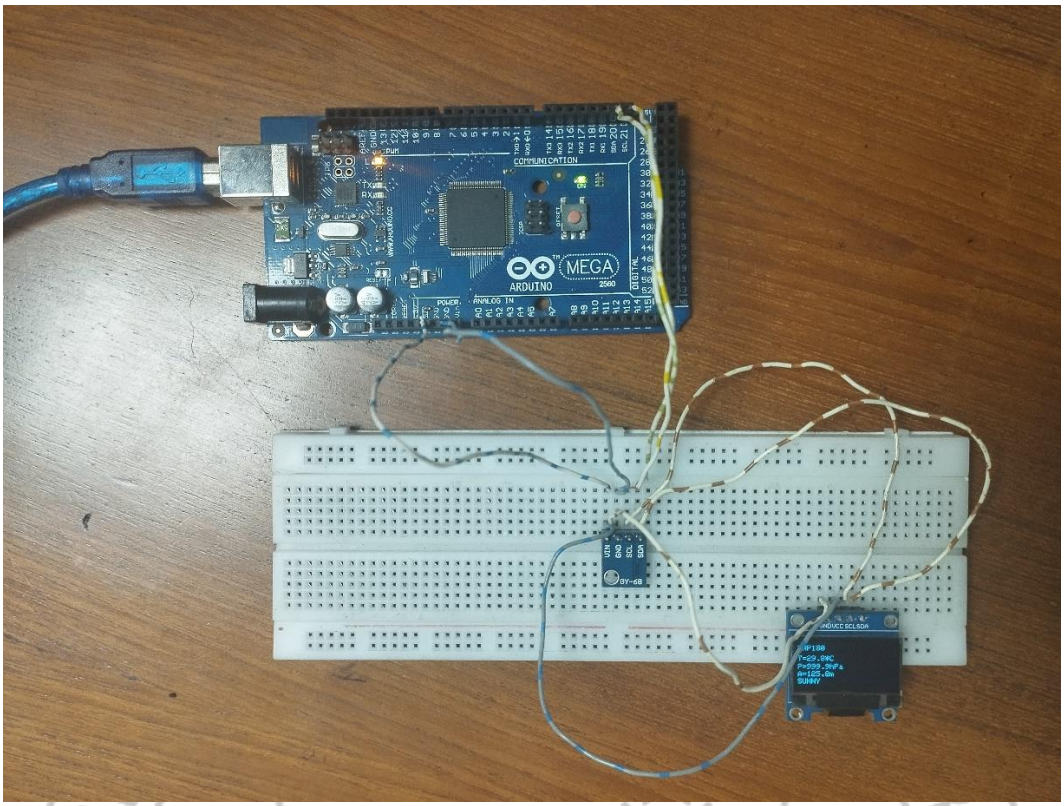

Figure 2: Hardware Set-up for weather forecast system

Experiment result:

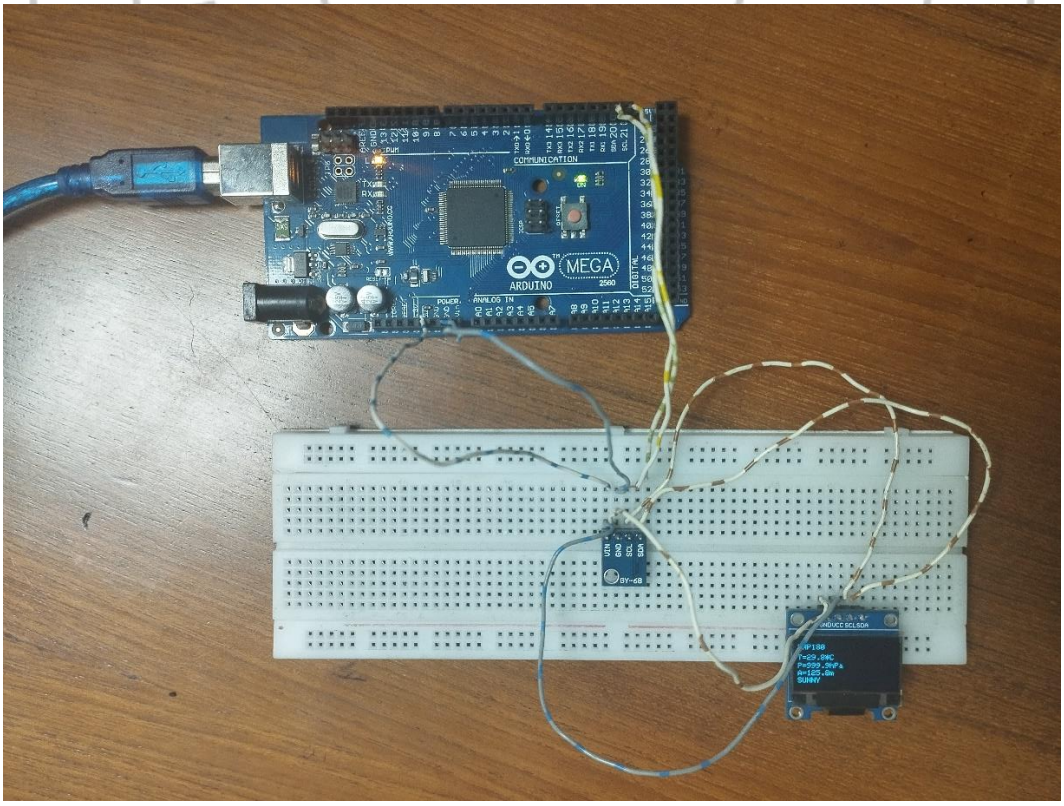

Fig 3: Temperature, pressure, altitude showing in display

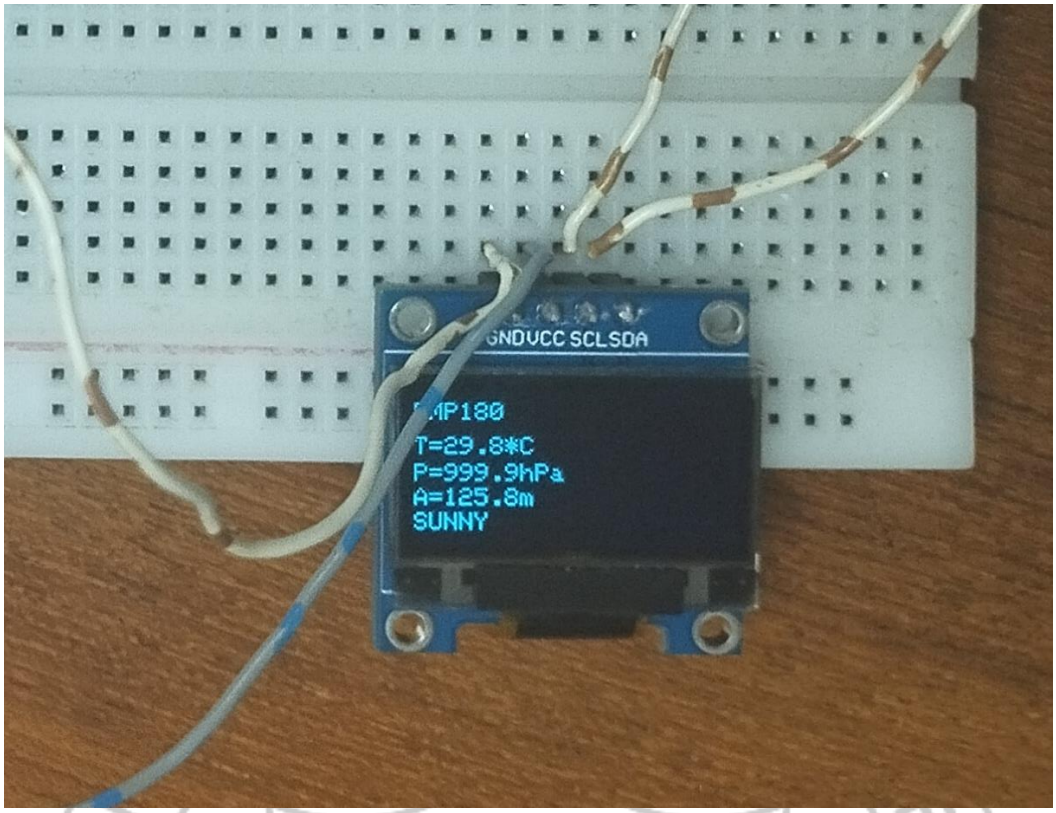

Fig 4: Temperature, pressure, altitude showing in display (zoomed view)

Code Analysis:

First of all, we imported some header files like SPI.h ,Wire.h ,Adafruit_GFX, Adafruit_SSD1306, and Adafruit_BMP085. Then we set the screen width, and height and set the sea level pressure to 101500. In the setup function, we passed SSD1306_SWITCHCAPVCC to generate the display voltage. then we check is bpm. begins, If BMP begin is not available then print "Could not find a valid BMP085 sensor, check to the wire! ". Then in the loop function, it just takes input through the sensor and shows the output in OLED. And the end of the code we check the simpleweatherdifference and based on that we print the SUNNY (simpleweatherdifference>0.25) , SUNNY/CLOUDY (simpleweatherdifference<=0.25) , and RAIN (simpleweatherdifference<-0.25).

```
#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);
}
```

Discussion:

By conducting this experiment, we learned about the timers in Arduino Uno microcontroller and how it works to study of blink test using and implementing of a traffic control system. This experiment was carried out in two ways. First, by using Arduino board, three colored animated LED lights (red, yellow, and green), three resistors breadboard and connecting wires, we created a traffic control circuit. The LED lights (red, yellow, and green) were connected to ports 8, 10 and 12. Then we wrote some code for the traffic control system in Arduino IDE. After that, we connected the Arduino board to the computer and ran the code for the blink test. Another way we did this experiment with the help of Proteus software to study was the blink test. At the time of using Proteus software, we faced some problems. But we solved these problems with the help of the internet. And, when we did this in Lab, there are some mistaken in pin configuration. And we solve this issue in lab. In both ways, we obtained the same result. Finally, we can say that our experiment was successful.