# American International University- Bangladesh

**Department of Electrical and Electronic Engineering**

EEE 4130: Microprocessor and Embedded Systems Laboratory

## Title:

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

## Objectives:

The objectives of this experiment are to-

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

## Theory and Methodology:

### Weather Prediction

The BMP085/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, BMP085/BMP180 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 BMP085/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 clearing weather. The typical reasoning can be seen in a comparison of molecular weights. If air is approximately 21% $O_2(g)$, 78% $N_2(g)$, $O_2(g)$ has a molecular mass of 32, and $N_2(g)$ has a mass of 28. Water vapor, $H_2O(g)$ 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 is 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 $H_2O$ 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 lower-pressure regions 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.

In some areas, it is 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.

**Example:**

An airport located at 600 m elevation would have a pressure of 93.97 kPa according to our pressure to altitude equation. If the weather was sunny and mostly clear, it would most probably have a published pressure of 101.3 kPa for weather conditions. It may not be extremely clear skies as this would be a high-pressure weather system. It would be a stable pressure with neither extreme low nor high pressure.

Remember to discern this information when trying to see if the MPL115A value matches the local weather barometric pressure. Sometimes a disparity in the value occurs due to normalization.

**Algorithms for weather Simple Approach**

How is weather predicted using the barometric sensor?

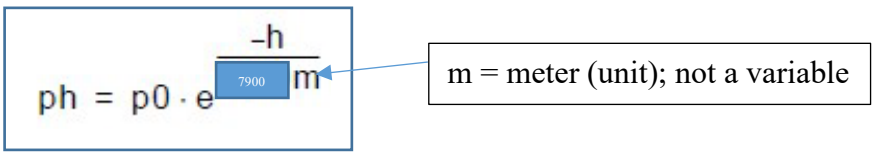
Approach-1:

There is a simple approach looking at increasing or decreasing pressure. Simply an increase over time is a trend that approaches "sunny" or "clear" days. Dropping pressure can signal a worsening "cloudy" or "rainy" day. This can be seen typically as a rising or falling bar on many simple solution weather stations. It can be interpreted as an increase/decrease gradient for the user to interpret, but the time interval is not used extensively to reach weather predictions. The user can look at the results for a 12-hour time frame to predict the weather trend.

This table is typically used: (dP = difference in pressure)

| Analysis | Output |
|---|---|
| dP > +0.25 kPa | Sun Symbol |
| -0.25 kPa ≤ dP ≤ 0.25 kPa | Sun/Cloud Symbol |
| dP < -0.25 kPa | Rain Symbol |

Approach-2: (Best Solution)
Another approach that is more direct and quicker in calculating the weather is to know the current altitude. This cuts the need to wait and see a "trend".




where p0 = 101.3 kPa and h = current altitude. The pressure (ph in KPa) for the local barometric can be calculated using the formula above. This is the pressure for good sunny weather at the current altitude location.

By using the pressure equation and knowing the normalized good weather pressure for the current location (best for a static weather station), the weather can be deduced by the difference. As in the table for the weather symbols, the ideal pressure is compared to the value from the MPL115A and the appropriate symbol of Sun/Cloud/Rain is selected.

Below simple C code from the DEMOAPEXSENSOR demo kit that calculates which weather symbol to display on the LCD screen.

- CurrentAltitude - (h in meter) Altitude in meters that is entered into the system by the user for that current static location.
- Pweather - (ph in kPa) Pressure at the current altitude. It is calculated using Pressure (ph in kPa) exponential equation, inputting CurrentAltitude (h) in meters. This is the ideal pressure for the current location on a stable relatively sunny day.
- decPcomp - (kPa) Value of sea-level-compensated pressure from BMP085/BMP180/MPL115A, lying in the range of 101.3 kPa±10%.

## Simple Weather Station Code

```
Pweather = (101.3 * exp(((float)(CurrentAltitude))/(-7900)));

Simpleweatherdiff = decPcomp-Pweather;
if (Simpleweatherdiff >0.25)
  Simpleweatherstatus = 0; //Sun Symbol
if ((Simpleweatherdiff <=0.25) || (Simpleweatherdiff >=(-0.25)))
  Simpleweatherstatus = 1; //Sun/Cloud Symbol
if (Simpleweatherdiff <(-0.25))
Simpleweatherstatus = 2; //Rain Symbol
```

Let us look at some data:

| decPcomp (kPa) (MPL115A Pressure) | PWeather (kPa) (Ideal weather) | Simpleweatherdiff (kPa) | Weather Type |
|---|---|---|---|
| 96.6 | 96.85 | -0.25 | Sun/Cloud |
| 96.4 | 96.85 | -0.45 | Rain |
| 97.4 | 96.85 | 0.55 | Sun |
| 96.92 | 96.85 | 0.07 | Sun/Cloud |

For example, Altitude for Tempe, AZ = 359 m, thus PWeather = 101.3 x $e^{-359/7900m}$ = 96.8 kPa.

Approach:1- Observing Pressure over time will yield similar results over 12 hours. In this case, the changes in the pressure that takes place over an extended time will be enough for the simple method to figure out the weather patterns. This negates the need for user input of the approximate location/altitude. The weather algorithm will be more accurate at the end of the 12-hour interval as the trend is visible versus at initialization.

Approach:2- Knowing the altitude can also be useful for a dynamically changing location to predict simple weather. Take for example a GPS unit: a GPS unit can give an approximate altitude measurement. Measuring the difference from the MPL115A pressure sensor and calculated pressure from the GPS altitude gives a close approximation of weather patterns quickly at that dynamically changing point. Weather approximation can be deduced in the symbol style as above.

## Advanced Version of Weather Station

Approach:3- A more complex approach is to measure the P/t and see how the gradient is changing over time. As in the simple approach, this does need to be kept in a static location during measurement. Essentially as time progresses, the weather can be broken into more exact categories than the simple approach of basic symbols.

This can also use less time than waiting for a full 12 hours to see the pattern of pressure change. In Table 3, the ranges of pressure change over time leading to the definition of the weather patterns shown. It is a change in the pressure per hour. 2-3 hours are needed to deduce how the pressure is migrating. r

**Table 3. Advanced Weather Determination**

| Analysis | Output |
|---|---|
| dP/dt > 0.25 kPa/h | Quickly rising High-Pressure System, unstable and hottest weather |
| 0.05 kPa/h < dP/dt < 0.25 kPa/h | Slowly rising High-Pressure System, stable and sunny weather |
| -0.05 kPa/h < dP/dt < 0.05 kPa/h | Stable weather condition (conducive) |
| -0.25 kPa/h < dP/dt < -0.05 kPa/h | Slowly falling Low-Pressure System, stable and rainy weather |
| dP/dt < -0.25 kPa/h | Quickly falling Low Pressure, unstable weather, and Thunderstorm |

### Hardware implementation:

Connect the circuit diagram as follows, upload the program into the board, and then run the program. Observe the weather parameters on the Organic light-emitting diodes (OLED) display.

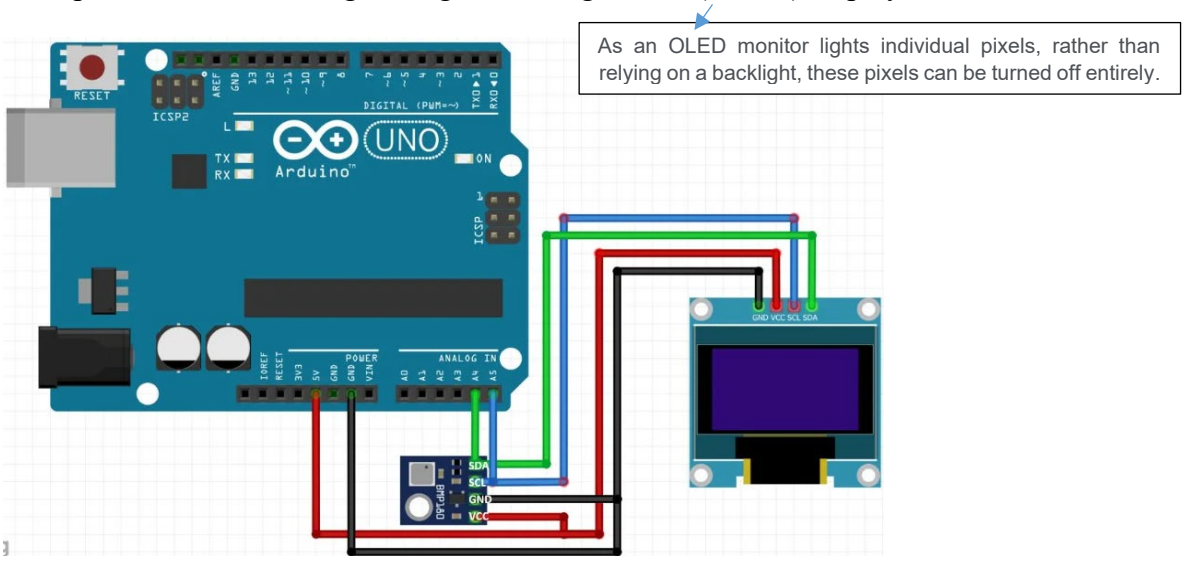


Figure 2. Arduino Uno with BMP180 and OLED

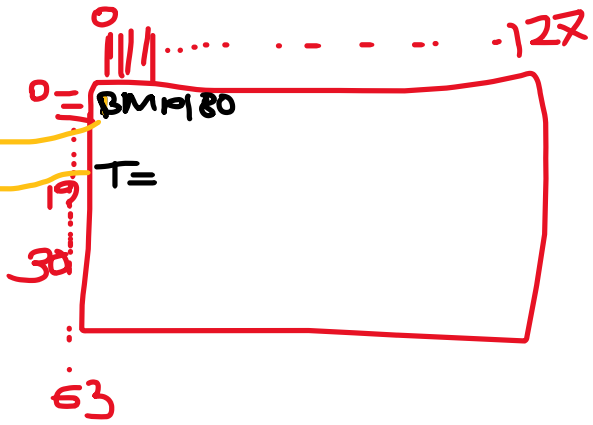
## Components List

- Arduino Uno Board
- BMP085/BMP180 / MPL115A
- inches96 inch OLED 128X64
- Breadboard and Jump Wires

## Program:

```
// For SPI serial communication
#include <SPI.h>
// For I2C serial communication
#include <Wire.h>

// providing a common syntax and set of graphics functions (like lines, circles, and text) for all of our LCD and OLED displays and LED matrices
#include <Adafruit_GFX.h>
// handling the low-level communication with the hardware like OLEDs.
// 128 pixels wide and 64 pixels deep (128x64)
#include <Adafruit_SSD1306.h>

// a library for the Adafruit BMP085/BMP180 Barometric Pressure + Temp sensor
#include <Adafruit_BMP085.h>

// 128 pixels wide and 64 pixels deep (128x64)
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display (SCREEN_WIDTH, SCREEN_HEIGHT);
// defining SEALEVELPRESSURE_HPA variable needed to calculate the altitude.
// changing it to current sea level pressure (101.5 kPa = 10.15hPa = 10.15*1000*10 Pa) at your location.
#define SEALEVELPRESSURE_HPA (101500)
// creating a "bmp" object so that we can access functions of "Adafruit_BMP085" related to it; I2C interface
Adafruit_BMP085 bmp;

float simpleweatherdifference, currentpressure, predictedweather, currentaltitude;

void setup () {
// SSD1306_SWITCHCAPVCC = generating display voltage from 3.3V internally
  display.begin (SSD1306_SWITCHCAPVCC, 0x3C); // Address 0x3C for 128x64 OLED display
        if (!bmp.begin()) {
  Serial.println ("Could not find a valid BMP085 sensor, check wiring!");
  while (1) {}
  }
}

void loop () {
// turning off all pixels; Clear display buffer
  display.clearDisplay();
// setting the font size, supports sizes from 1 to 8; 1 = Normal 1:1 pixel scale; 2 = Draw 2X-scale text
  display.setTextSize(1);
//setting white font and black background.
```

```
  display.setTextColor(SSD1306_WHITE);

// set up the OLED's number of columns and rows
display.setCursor(0,5);
  display.print("BMP180");
  display.setCursor(0,19);
  display.print("T=");
// 1 is optional here
  display.print(bmp.readTemperature(),1);
  display.println("*C");

  /*prints BME180 pressure in Hectopascal Pressure Unit*/
  display.setCursor(0,30);
  display.print("P=");
// F and 1 are optional here
  display.print(bmp.readPressure()/100.0F,1);
  display.println("hPa");

   /*prints BME180 altitude in meters*/
  display.setCursor(0,40);
  display.print("A=");
// 1 is optional here
  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);
}
```




**Questions for Report Writing:**

- Include all codes and scripts in the lab report following the writing template.
- Now implement the same system in the Proteus simulation tool by practicing from the link: (https://www.youtube.com/watch?v=M4f4ntzgkv4)