```
// Arduino Weather Station by Glen Popiel - KW5GP
//
// DHT Relative Humidity and Termperature Code is based on DHT Library
// from Arduino Playground, http://playground.arduino.cc/Main/DHTLib
// Released under Creative Commons Attribution ShareAlike 3.0. license
//
// BMP085 Barometric Pressure and temperature Sensore Code is based on
// Code provided by http://bildr.org/2011/06/bmp085-arduino/
// Released under Creative Commons Attribution ShareAlike 3.0 Unported
(CC BY-SA 3.0)
// Based largely on code by  Jim Lindblom

/*
 Get Relative Humidity and Temperature from the RHT03
 Get pressure, altitude, and temperature from the BMP085.
 Display it on a Nokia display
 */

#include <Wire.h>  // Use the internal I2C Library
#include <dht.h>  // Use the DHT Relative Humidity Library
#include <LCD5110_Basic.h>    // Use the Nokia 5110 LCD Library

#define BMP085_ADDRESS 0x77  // I2C address of BMP085

dht DHT;  // Define the DHT object
#define DHT22_PIN 2  // Set the I/O pin used for the RHT03 Sensor

LCD5110 glcd(12,11,10,8,9);  // Set the I/O pins used by the Nokia
display
extern uint8_t SmallFont[];  // Define the Small Font for the Nokika
display

const unsigned char OSS = 0;  // Oversampling Setting for the BMP085

// Calibration values for the BMP085
int ac1, ac2, ac3, b1, b2, mb, mc, md;
unsigned int ac4, ac5, ac6;

long b5;  // b5 is calculated in bmp085GetTemperature(...), this variable
is also used in bmp085GetPressure(...)
          // so ...Temperature(...) must be called before
...Pressure(...).

int chk ;   // Status Check variable for RHT03

float centigrade, fahrenheit, inHg;

void setup()
{
  Wire.begin();  // Start the I2C Interface

  glcd.InitLCD(65);  // Initialize the Nokia 5110 Display, set the
Contrast to 65
  glcd.setFont(SmallFont);  // Set the Font to Small Font
```

```
  glcd.print("KW5GP", CENTER, 0);  // Display the Startup screen
  glcd.print("Weather", CENTER, 8);
  glcd.print("Station", CENTER,16);
  glcd.print("Initializing", CENTER,32);
  delay(3000);
  glcd.clrScr();  // Clear the LCD screen

  bmp085Calibration();  // Run the BMP085 Calibration Function

}  // End Setup

void loop()
{
  glcd.print("Current Wx", CENTER, 0);
  // Read the RHT03 RH/Temp Sensor
  glcd.print("R/H : ",0,16);
  glcd.print("Temp: ",0,24);

  chk = DHT.read22(DHT22_PIN);    // Read the RHT03 RH/Temp Sensor
  switch (chk)
  {
    case DHTLIB_OK:
      // Display the RH Data if it's a valid read
      glcd.printNumF(DHT.humidity,1,30,16);
      glcd.print("%     ",55,16);
      centigrade = DHT.temperature;
      fahrenheit = (centigrade * 1.8) + 32;   // convert to Fahrenheit
      glcd.printNumF(fahrenheit,1,30,24);
      glcd.print("F",55,24);
      break;

    case DHTLIB_ERROR_CHECKSUM:
      glcd.print("CK Error",25,16);
      break;

    case DHTLIB_ERROR_TIMEOUT:
      glcd.print("T/O Error",25,16);
      break;

    default:
      glcd.print("Unk Error",25,16);
      break;
  }

  // Now Read and Display the Pressure
  glcd.print("Press: ",0,40);

  float temperature = bmp085GetTemperature(bmp085ReadUT()); //MUST be
called first
  float pressure = bmp085GetPressure(bmp085ReadUP());
  float atm = pressure / 101325; // "standard atmosphere"
  float altitude = calcAltitude(pressure); //Uncompensated caculation -
in Meters
```

```
  pressure  = pressure / 1000;  // Convert to KiloPascals
  inHg = pressure * 0.2952998016471232;  // Convert KiloPascals to Inches
of Mercury

  glcd.printNumF(inHg,2,40,40);  // Display the Barometric Pressure in
Inches of Mercury

  delay(5000); //wait 5 seconds and get values again.
}

// This funtion stores all of the bmp085's calibration values into global
variables
// Calibration values are required to calculate temp and pressure
// This function should be called at the beginning of the program
void bmp085Calibration()
{
  ac1 = bmp085ReadInt(0xAA);
  ac2 = bmp085ReadInt(0xAC);
  ac3 = bmp085ReadInt(0xAE);
  ac4 = bmp085ReadInt(0xB0);
  ac5 = bmp085ReadInt(0xB2);
  ac6 = bmp085ReadInt(0xB4);
  b1 = bmp085ReadInt(0xB6);
  b2 = bmp085ReadInt(0xB8);
  mb = bmp085ReadInt(0xBA);
  mc = bmp085ReadInt(0xBC);
  md = bmp085ReadInt(0xBE);
}

// This function calculates the temperature in deg C
float bmp085GetTemperature(unsigned int ut){
  long x1, x2;

  x1 = (((long)ut - (long)ac6)*(long)ac5) >> 15;
  x2 = ((long)mc << 11)/(x1 + md);
  b5 = x1 + x2;

  float temp = ((b5 + 8)>>4);
  temp = temp /10;

  return temp;
}

// This function calculates pressure
// calibration values must be known
// b5 is also required so bmp085GetTemperature(...) must be called first.
// Value returned will be pressure in units of Pa.
long bmp085GetPressure(unsigned long up)
{
  long x1, x2, x3, b3, b6, p;
  unsigned long b4, b7;

  b6 = b5 - 4000;
  // Calculate B3
```

```
  x1 = (b2 * (b6 * b6)>>12)>>11;
  x2 = (ac2 * b6)>>11;
  x3 = x1 + x2;
  b3 = (((((long)ac1)*4 + x3)<<OSS) + 2)>>2;

  // Calculate B4
  x1 = (ac3 * b6)>>13;
  x2 = (b1 * ((b6 * b6)>>12))>>16;
  x3 = ((x1 + x2) + 2)>>2;
  b4 = (ac4 * (unsigned long)(x3 + 32768))>>15;

  b7 = ((unsigned long)(up - b3) * (50000>>OSS));
  if (b7 < 0x80000000)
    p = (b7<<1)/b4;
  else
    p = (b7/b4)<<1;

  x1 = (p>>8) * (p>>8);
  x1 = (x1 * 3038)>>16;
  x2 = (-7357 * p)>>16;
  p += (x1 + x2 + 3791)>>4;

  long temp = p;
  return temp;
}

// This function reads 1 byte from the BMP085 at 'address'
char bmp085Read(unsigned char address)
{
  unsigned char data;

  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(address);
  Wire.endTransmission();

  Wire.requestFrom(BMP085_ADDRESS, 1);
  while(!Wire.available())
    ;

  return Wire.read();
}

// This function reads 2 bytes from the BMP085
// First byte will be from 'address'
// Second byte will be from 'address'+1
int bmp085ReadInt(unsigned char address)
{
  unsigned char msb, lsb;

  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(address);
  Wire.endTransmission();

  Wire.requestFrom(BMP085_ADDRESS, 2);
```

```
  while(Wire.available()<2)
    ;
  msb = Wire.read();
  lsb = Wire.read();

  return (int) msb<<8 | lsb;
}

// This functon reads the uncompensated temperature value
unsigned int bmp085ReadUT()
{
  unsigned int ut;

  // Write 0x2E into Register 0xF4
  // This requests a temperature reading
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(0xF4);
  Wire.write(0x2E);
  Wire.endTransmission();

  // Wait at least 4.5ms
  delay(5);

  // Read two bytes from registers 0xF6 and 0xF7
  ut = bmp085ReadInt(0xF6);
  return ut;
}

// This function reads the uncompensated pressure value
unsigned long bmp085ReadUP()
{
  unsigned char msb, lsb, xlsb;
  unsigned long up = 0;

  // Write 0x34+(OSS<<6) into register 0xF4
  // Request a pressure reading w/ oversampling setting
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(0xF4);
  Wire.write(0x34 + (OSS<<6));
  Wire.endTransmission();

  // Wait for conversion, delay time dependent on OSS
  delay(2 + (3<<OSS));

  // Read register 0xF6 (MSB), 0xF7 (LSB), and 0xF8 (XLSB)
  msb = bmp085Read(0xF6);
  lsb = bmp085Read(0xF7);
  xlsb = bmp085Read(0xF8);

  up = (((unsigned long) msb << 16) | ((unsigned long) lsb << 8) |
(unsigned long) xlsb) >> (8-OSS);

  return up;
}
```

```
// This function writes a value to the selected register
void writeRegister(int deviceAddress, byte address, byte val)
{
  Wire.beginTransmission(deviceAddress); // start transmission to device
  Wire.write(address);       // send register address
  Wire.write(val);         // send value to write
  Wire.endTransmission();     // end transmission
}

// This function reads the selected register
int readRegister(int deviceAddress, byte address)
{
  int v;
  Wire.beginTransmission(deviceAddress);
  Wire.write(address); // register to read
  Wire.endTransmission();

  Wire.requestFrom(deviceAddress, 1); // read a byte

  while(!Wire.available()) {
    // waiting
  }

  v = Wire.read();
  return v;
}

// This function calculates altitude
float calcAltitude(float pressure)
{
  float A = pressure/101325;
  float B = 1/5.25588;
  float C = pow(A,B);
  C = 1 - C;
  C = C /0.0000225577;

  return C;
}
```