```
// Adding code to calculate + or - average barometric pressure (Yearly KC bp = 30.5 in)

// Added f() macro to all strings

//*****************Arduino anemometer sketch*****************************
// Make sure to set serial printer to 57600 baud rate
// Using floats instead of integers
// Needs a hardware debounce circuit!
// Add a comma and a one to limit to one decimal EX:(wSpeedMPH,1);

// What the F() macro Does (Strings only - saves the string  )
// The F() macro tells the compiler to leave this particular array in PROGMEM. Then when it is time to access it, one byte of the data is copied to RAM at a time.
// There's a small performance overhead for this extra work. However, printing strings over Serial or to a LCD is a really slow process, so a few extra clock cycles really won't matter.
// Added IF statement if aFreq > 500 the aFreq = 0  ----- for testing

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Adafruit_BME280.h>

LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 16 chars and 2 line display

float MaxSpeed;
float MaxAvg;

int Z=0;   // counter for how many times aFreq=1000
int humidity;
int setTime=50;
int dt=5000;
int tempF;
int rh;
float bp;
float mb;
float adjusted_pressure;
float adjustedbp;
int rhadjust;
float x=0;

Adafruit_BME280 bme; // use I2C interface
Adafruit_Sensor *bme_temp = bme.getTemperatureSensor();
Adafruit_Sensor *bme_pressure = bme.getPressureSensor();
Adafruit_Sensor *bme_humidity = bme.getHumiditySensor();

const byte interruptPin = 2; //anemomter input to digital pin                    ----------  Connect wind anemometer to PIN#2 on UNO R3 board
volatile unsigned long sTime = 0; //stores start time for wind speed calculation
unsigned long dataTimer = 0; //used to track how often to communicate data
```

```
volatile float pulseTime = 0; //stores time between one anemomter relay closing and the next
volatile float culPulseTime = 0; //stores cumulative pulsetimes for averaging
volatile bool start = true; //tracks when a new anemometer measurement starts
volatile unsigned int avgWindCount = 0; //stores anemometer relay counts for doing average wind
speed
float aSetting = 60.0; //wind speed setting to signal alarm

// -------------------------------- VOID SETUP ----------------------------------------------------------------------------------
---------------

void setup() {
  pinMode(13, OUTPUT); //setup LED pin to signal high wind alarm condition
  pinMode(interruptPin, INPUT_PULLUP); //set interrupt pin to input pullup
  attachInterrupt(interruptPin, anemometerISR, RISING); //setup interrupt on anemometer input pin,
interrupt will occur whenever falling edge is detected
  dataTimer = millis(); //reset loop timer

  MaxSpeed = 0;  // Set Maximum Speed to 0
  MaxAvg = 0;

  lcd.init();                  // initialize the lcd
  // Print a message to the LCD.
  lcd.backlight();
  lcd.print(F("ForceTronics 39"));
  delay(1000);

  Serial.begin(57600);  //start serial monitor to communicate wind data
  Serial.println(F("BME280 Sensor event test"));
  if (!bme.begin()) {
  Serial.println(F("Could not find a valid BME280 sensor, check wiring!"));
  while (1) delay(10);
  }

  bme_temp->printSensorDetails();
  bme_pressure->printSensorDetails();
  bme_humidity->printSensorDetails();

}

// -------------------------------- VOID LOOP ----------------------------------------------------------------------------------
---------------

void loop() {

  unsigned long rTime = millis();
  if((rTime - sTime) > 2400) pulseTime = 0; //if the wind speed has dropped below 1MPH than set it to
zero
```

```
  if((rTime - dataTimer) > 1800){ //See if it is time to transmit

    detachInterrupt(interruptPin); //shut off wind speed measurement interrupt until done
communication
    int aWSpeed = getAvgWindSpeed(culPulseTime,avgWindCount); //calculate average wind speed
    if(aWSpeed > MaxAvg){MaxAvg = aWSpeed;}                  // Statement that checks for Maximum
Average Wind Speed
    if (MaxAvg > 150) { Serial.println( "MaxAvg over 150 ");MaxAvg =0;}                      // Reset MaxAvg to
zero when the number is too large due to very low wind speed
    if(aWSpeed >= aSetting) digitalWrite(13, HIGH);   // high speed wind detected so turn the LED on
    else digitalWrite(13, LOW);   //no alarm so ensure LED is off
    culPulseTime = 0; //reset cumulative pulse counter
    avgWindCount = 0; //reset average wind count

    float aFreq = 0; //set to zero initially
    if(pulseTime > 0.0) aFreq = getAnemometerFreq(pulseTime); //calculate frequency in Hz of
anemometer, only if pulsetime is non-zero
    if(aFreq > 500) {aFreq = 0; Serial.println( "Stopped a 1000 aFreq!");Z = Z + 1;}
    float wSpeedMPH = getWindMPH(aFreq); //calculate wind speed in MPH, note that the 2.5 comes
from anemometer data sheet
     if(wSpeedMPH > MaxSpeed){MaxSpeed = wSpeedMPH;}       // Statement that checks for Maximum
Wind Speed
     if(MaxSpeed > 200) {MaxSpeed = 0;}

    // ------------- Timer for checking bp ---------------------------------------

  sensors_event_t temp_event, pressure_event, humidity_event;
  bme_temp->getEvent(&temp_event);
  bme_pressure->getEvent(&pressure_event);
  bme_humidity->getEvent(&humidity_event);

  tempF={(temp_event.temperature*1.8)+32};
  rh=(humidity_event.relative_humidity);
  rhadjust=(humidity_event.relative_humidity -22);
  bp={(pressure_event.pressure*0.02953)};
  adjustedbp={(bp + 1.03)};                          //  Adjustment to bring my bp sensor reading up to KC
Weather data bp
  adjusted_pressure={(pressure_event.pressure + 32.16)};   //  31.45   Adjustment to bring my bp sensor
reading up to KC Weather data bp


  {
  const unsigned long fiveMinutes = .5 * 60 * 1000UL;
  static unsigned long lastSampleTime = 0 - fiveMinutes;  // initialize such that a reading is due the first
time through loop()

  unsigned long now = millis();
  if (now - lastSampleTime >= fiveMinutes)
```

```
  {
    lastSampleTime += fiveMinutes;
  }
 lcd.setCursor(0,3);

}

     // ---------------------------------------------------------------------------------------------------------------------------


  Serial.print("  ");
  Serial.print(wSpeedMPH,1); Serial.print(F(" mph    "));
  Serial.print(MaxSpeed,1);Serial.print(F(" max mph   -"));


  Serial.print(F("   Temp "));Serial.print(tempF,1);Serial.print(F(" F    -"));
  Serial.print(F("    Humidity = "));

  Serial.print( rhadjust);
  Serial.print(F(" %    -    "));

  Serial.print(F("Pressure = "));
  Serial.print(adjustedbp,2);

  //Serial.print(bp);
  Serial.print(F(" Hg"));
  Serial.println(F("    "));


  // Serial.print(Z); Serial.println(" = 1000 Counter     ");
  // Serial.end(); //serial uses interrupts so we want to turn it off before we turn the wind measurement
interrupts back on

    lcd.setCursor(0,0);
    lcd.print(wSpeedMPH,1);
    lcd.print(F(" mph  "));
    lcd.print(MaxSpeed,1);
    lcd.print(F(" max   "));

    lcd.setCursor(0,1);
    lcd.print(tempF);
    lcd.print(F(" F   "));

    lcd.print( rhadjust);
    lcd.print(F("%"));
    lcd.print(F(" RH   "));

    lcd.setCursor(0,2);
```

```
    lcd.print(adjustedbp);
    lcd.print(F(" hg  "));

    lcd.setCursor(0,3);
    lcd.print(F("              [39]"));

    // delay(1000);


   start = true; //reset start variable in case we missed wind data while communicating current data out
   attachInterrupt(digitalPinToInterrupt(interruptPin), anemometerISR, RISING); //turn interrupt back on
   dataTimer = millis(); //reset loop timer
  }
}

//using time between anemometer pulses calculate frequency of anemometer
float getAnemometerFreq(float pTime) { return (1/pTime); }
//Use anemometer frequency to calculate wind speed in MPH, note 2.5 comes from anemometer data
sheet
float getWindMPH(float freq) { return (freq*2.5); }
//uses wind MPH value to calculate KPH
float getWindKPH(float wMPH) { return (wMPH*1.61); }
//Calculates average wind speed over given time period
float getAvgWindSpeed(float cPulse,int per) {
  if(per) return getWindMPH(getAnemometerFreq((float)(cPulse/per)));
  else return 0; //average wind speed is zero and we can't divide by zero
  }

//This is the interrupt service routine (ISR) for the anemometer input pin
//it is called whenever a falling edge is detected
void anemometerISR() {
  unsigned long cTime = millis(); //get current time
  if(!start) { //This is not the first pulse and we are not at 0 MPH so calculate time between pulses
   // test = cTime - sTime;
    pulseTime = (float)(cTime - sTime)/1000;
    culPulseTime += pulseTime; //add up pulse time measurements for averaging
    avgWindCount++; //anemomter went around so record for calculating average wind speed
  }
  sTime = cTime; //store current time for next pulse time calculation
  start = false; //we have our starting point for a wind speed measurement
}
```