Digital Barometer

Barometer with trend display

2021-12-13 / Document v.1.2 / Michael Willems (michael@willems.ca)

Basics

This Digital Barometer is a sophisticated electronic barometer, based on a BMP180-based GY68 Air pressure/temp meter chip on a small dedicated circuit board. An Arduino microcontroller provides the necessary logic.

This barometer is intended to give a quick overview of current barometric pressure (“QFE”), as well as 3-hour and 1-hour trend (rising/falling) and magnitude of trend, as per the British meteorological office Marine Forecast (see p.3).

The unit also provides a “not ready” indication and basic prediction of wind and precipitation, based on pressure change trend and magnitude.

Operation

When the unit is connected to power via the USB connector on the back panel, an LED test is done that briefly lights all LEDs. After that, the unit displays “done”, and finally “hPa” or “ºC” is displayed (as an indication of units), and operation starts. Barometric pressure in hPa is now displayed, unless switch 1 is in the “down” position, in which case temperature in ºC is displayed.

For the first three hours, the red “Not Ready” LED lights up, indicating the there is insufficient data to display trend direction and trend magnitude. After three hours, the red LED extinguishes and normal operation fully starts:

Rising/Falling:

The “Rising” and “Falling” LEDs normally display the **three-hour trend**, as follows:

- Rising by more than 0.5 hPa in three hours: “Rising” LED lights;
- Falling by more than 0.5 hPa in three hours: “Falling” LED lights .

When **switch 2** is in the “down” position, the “Rising” and “Falling” LEDs display the **one-hour** trend, as follows:

- Rising by more than 0.5 hPa in the past hour: “Rising” LED lights;
- Falling by more than 0.5 hPa in the past hour: “Falling” LED lights .

Magnitude of change:

The three horizontal LEDs display the magnitude of the change in barometric pressure. The divisions are those used by the British Met office:

- **Less than 1.5 hPa change in the past three hours:** no display[1]
- **1.5–3.5 hPa change in three hours:** “Falling”, or “Rising”. First LED lights. [2]
- **3.5–6 hPa change in three hours: “**Falling Quickly”, or “Rising Quickly”. Second LED lights.
- **>6 hPa change in three hours: “**Falling Very Rapidly", or “Rising Very Rapidly”. Third (red) LED lights.

Front Panel Switches:

SWITCH 1, when raised, causes the unit to display barometric pressure in hPa. When lowered, the unit displays temperature in ºC. Also, when the unit starts, the display momentarily displays “hPa” or ºC”, depending on switch position.

SWITCH 2, when raised, the two “rising”/“falling” LEDs display the three-hour trend. When the switch is down, the LEDs display the one-hour trend.

Pushbutton: When pushed, the unit momentarily displays:

- The actual three-hour increase or decrease, in hPa;
- The up/down one-hour trend, using the up/down LEDs.

---

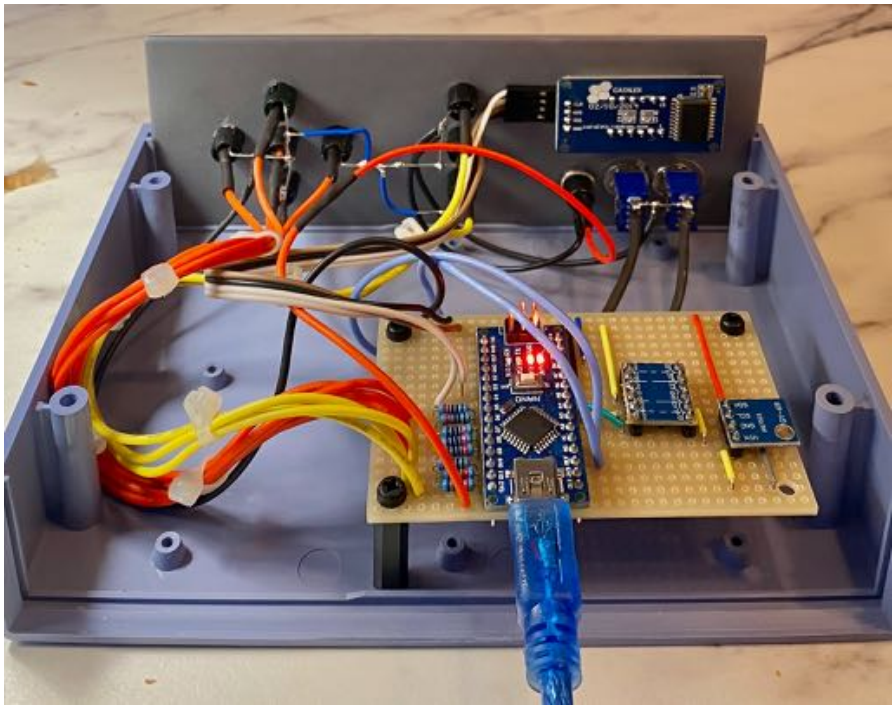
[1] This means that when the “Rising” or “Falling” LED **is** lit and the horizontal LEDs are **not** lit, the trend is “rising slowly” or “falling slowly”.

[2] The terms here are also the terms that the British Meteorological Office uses.
See https://www.metoffice.gov.uk/weather/guides/coast-and-sea/glossary

Technical and Construction Details

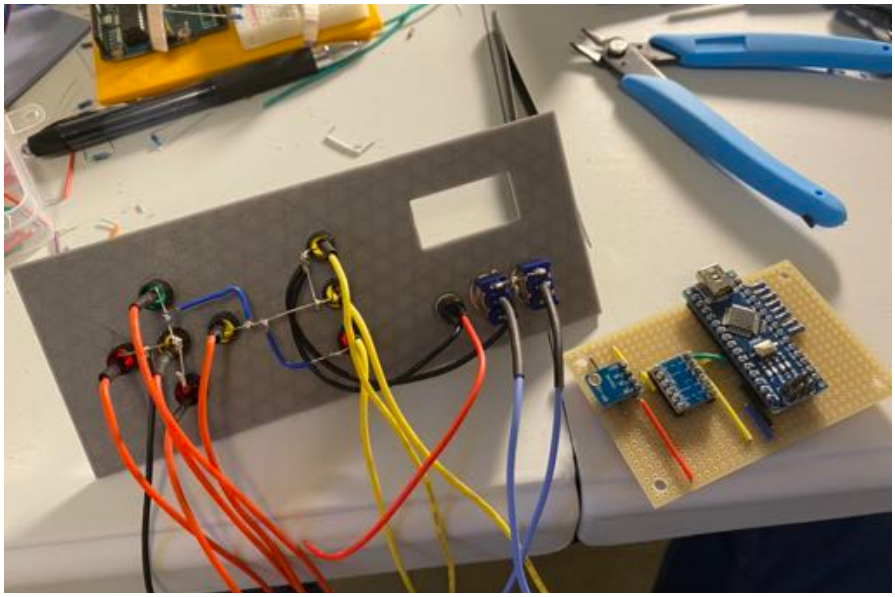
The unit uses an Arduino Nano microcontroller as its controller, and LEDs and a 4-digit 7-segment LED display for display of status, trend, predictive and numerical information.

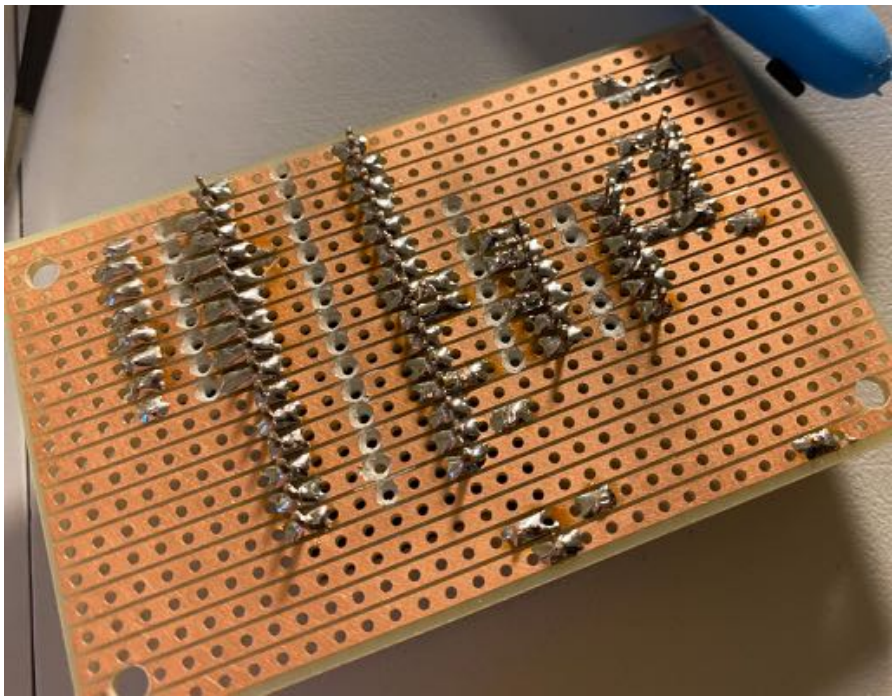
Construction:

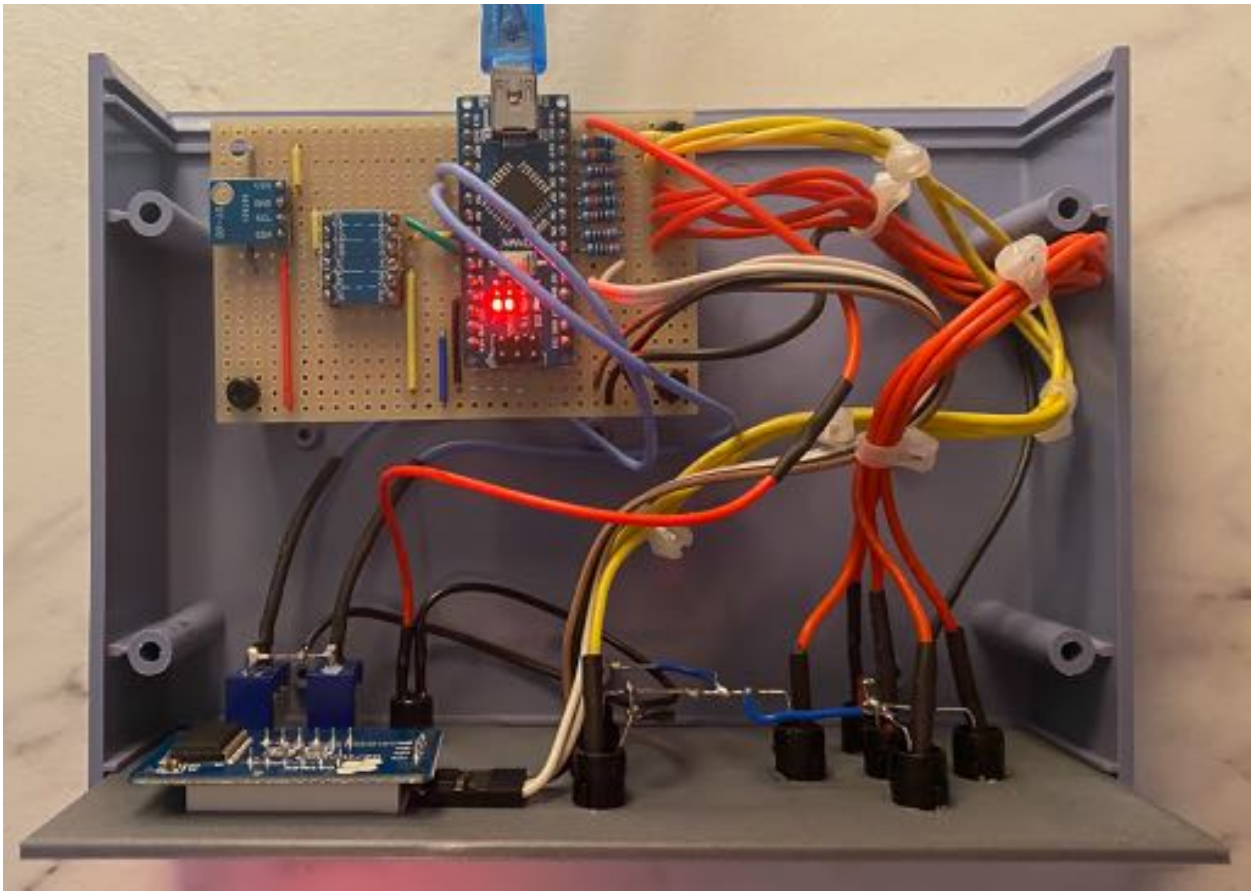


The unit is housed in a box with a custom-printed front and back panel. It is intended for continuous indoors operation.

Construction details:





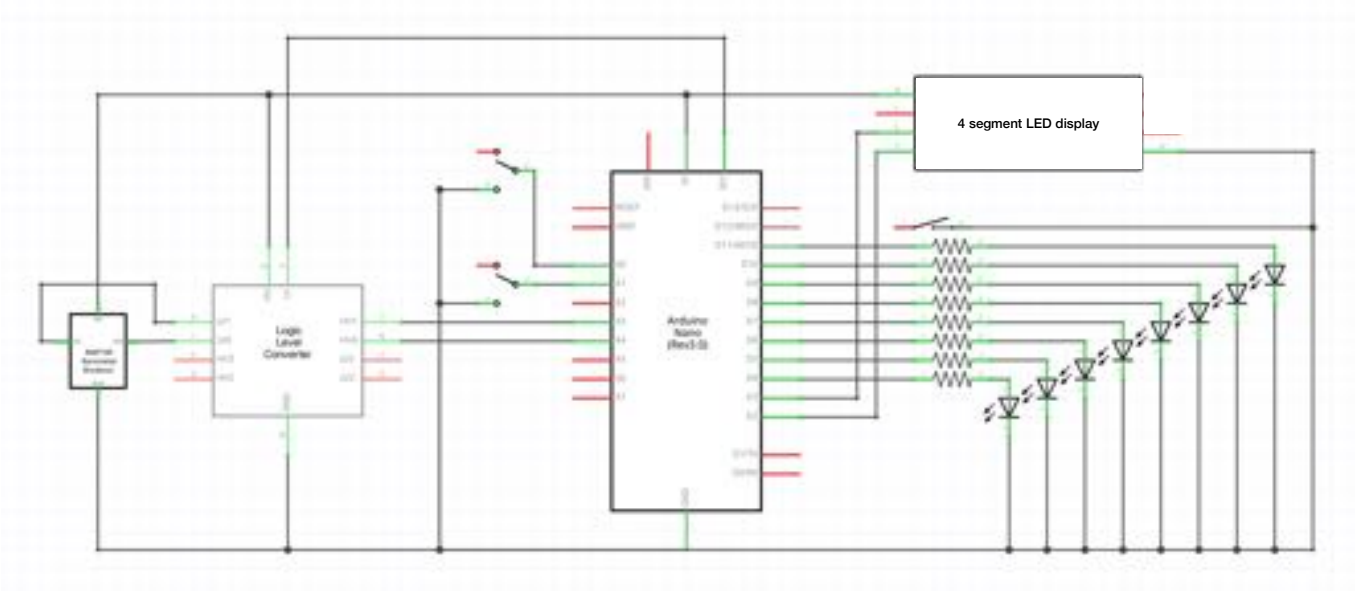


The Arduino Nano microcontroller is equipped with a custom made stripboard that contains the Arduino and the GY-68 sensor board (containing the chip and support circuitry), and a 3.3-5V logic voltage converter to enable the 3.3V-powered chip to talk to the 5V-powered Arduino I2C ports.

Circuit board and display are glued in place using superglue.

The unit can be powered by a standard USB port, e.g. of an external power supply or of a computer. If using a power supply, but is probably good to use a regulated supply, as noise the 5V power line could negatively affect unit functionality.

Circuit Diagram:



Arduino Code:

```
/*-------------------------------------------------------------------------------
 New Sketch with:
 - 4-digit 7-segment LED display
 - GY68  Air pressure/temp meter  (BMP180)
 - LEDs to indicate trend, etc
 - (For future use: Orleans is 87m / 285ft elevation for QNH calc)
 Code by Michael Willems
 version 1.0
 2021-12-12
-------------------------------------------------------------------------------*/


// ---------------------------------------------------------------------------------------------
// INCLUDES AND DECLARATIONS:
// ---------------------------------------------------------------------------------------------
#include <Wire.h>                 // for I2C communication (Arduino IDE built-in library)
#include <Adafruit_BMP085.h>      // for air pressure/temp meter (Arduino IDE built-in library)
Adafruit_BMP085 bmp;
int temperature;
int barometer;
int oldtemperature = 0;
int oldbarometer = 0;

/* Code for TM1637 4 digit 7 segment display: */
#include <TM1637Display.h>
#define CLK 2  // Define the connections pins for display 1
#define DIO 3
// Create display object of type TM1637Display:
TM1637Display display = TM1637Display(CLK, DIO);

// Create array that turns all segments on:
const uint8_t data[] = {0xff, 0xff, 0xff, 0xff};

// Create array that turns all segments off:
const uint8_t blank[] = {0x00, 0x00, 0x00, 0x00};

// Set the individual segments per digit to spell words or create other symbols:
const uint8_t done[] = {
  SEG_B | SEG_C | SEG_D | SEG_E | SEG_G,           // d
  SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F,   // O
  SEG_C | SEG_E | SEG_G,                           // n
  SEG_A | SEG_D | SEG_E | SEG_F | SEG_G            // E
};
const uint8_t hPA[] = {
  0x00,                                            //
  SEG_C | SEG_E | SEG_F | SEG_G,                   // h
  SEG_A | SEG_B | SEG_E | SEG_F | SEG_G,           // P
  SEG_A | SEG_B | SEG_C | SEG_E | SEG_F | SEG_G    // A
};

const uint8_t dC[] = {
  0x00,                                            //
  0x00,                                            //
  SEG_A | SEG_B | SEG_F | SEG_G,                   // [deg]
  SEG_A | SEG_D | SEG_E | SEG_F                    // C
};
```

```
const uint8_t zero[] = {
  SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F,    // 0
  SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F,    // 0
  0x00,
  0x00
};

// Misc variables etc:
unsigned long counter;            // for flashing on-board LED 13 with 1s frequency
unsigned long halfhourcounter;    // for the "once every half hour" stuff
unsigned long cyclecounter = 0;   // to see how often the loop does its thing
int onboardled = 13;              // for heartbeat LED 13
int risingled = 4;                // currently rising
int fallingled = 5;               // currently falling
int delta1led = 6;                // >1.5 change
int delta2led = 7;                // >3.5 change
int delta3led = 8;                // >6 change
int predictwindled = 9;           // wind predicted
int predictrainled = 10;          // precipitation predicted
int notreadyled = 11;             // prediction and trend not reADY, IT HASN'T BEEN 3 HOURS YET
int pushbutton = 12;
long baromem[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} ;   // for tracking barometer trend
long barodiff1h = 0;
long barodiff3h = 0;
long uptime = 0;



// ----------------------------------------------------------------------------------------
// THE SETUP (RUNS ONCE):
// ----------------------------------------------------------------------------------------
void setup() {
  pinMode(onboardled, OUTPUT);
  pinMode(pushbutton, INPUT_PULLUP);
  pinMode(risingled, OUTPUT);
  pinMode(fallingled, OUTPUT);
  pinMode(delta1led, OUTPUT);
  pinMode(delta2led, OUTPUT);
  pinMode(delta3led, OUTPUT);
  pinMode(predictwindled, OUTPUT);
  pinMode(predictrainled, OUTPUT);
  pinMode(notreadyled, OUTPUT);
  pinMode (A0, INPUT_PULLUP);  //switch 1: on/off
  pinMode (A1, INPUT_PULLUP);  //switch 2: 3 hour or 1 hour up/down trend?
  Serial.begin(9600);
  Wire.begin(); // Initiate the Wire library
  delay(10);
  counter = millis();
  halfhourcounter = millis();
  cyclecounter = 0;  //for testing loops/second
  //
  // now set up, and start, the air pressure detector:
  if (!bmp.begin()) {
         Serial.println("Could not find a valid BMP085 sensor, check wiring!");
         while (1) {}
  }
  // Set the brightness:
  display.setBrightness(6);
  // All segments on:
```

```
  display.setSegments(data);
  delay(1000);
  // flash all LEDs:
  for (int test = 4; test<12; test ++){
    digitalWrite (test, HIGH);
    delay (500);
    digitalWrite (test, LOW);
  }
  // display "done":
  display.setSegments(done);
  delay(1000);

  // clear and display the units
  display.clear();

  if (digitalRead (A0) == HIGH){
    display.setSegments(hPA);
  }
  else {
    display.setSegments(dC);
  }
  delay(2500);
  display.clear();

  // now we set the barometer memories all the same to start:
    baromem[0] = (bmp.readPressure())-1;
    for (int mem = 1; mem<16; mem++){
      baromem[mem] = (baromem[0]);
    }
}

// ------------------------------------------------------------------------------------------
// THE LOOP:
// ------------------------------------------------------------------------------------------
void loop() {

  // --------------------------------
  // Now all the once-a-second stuff:
  // --------------------------------
  if ((millis() - counter) > 1000) {   //yup, it's time
    counter = millis();
    digitalWrite (onboardled,!(digitalRead(onboardled)));   //the on-board heartbeat LED
    //--------------------------------
    // get the pressure and temp:
    temperature = ((bmp.readTemperature())-0);   //read temp from the chip & adjust
    barometer = ((bmp.readPressure()/100)+9);    //read pressure from the chip & adjust
    //--------------------------------
    // now display the values (+re-display if they have changed):
    if ((barometer != oldbarometer) or (1 != 2)) {
     if (digitalRead(A0) == HIGH) {
       display.showNumberDec (barometer, false, 4, 0);
     }
     else {
       display.showNumberDec (temperature, false, 4, 0);
     }
      oldbarometer = barometer;
    }
    cyclecounter = 0;
    //--------------------------------
```

```
  // check how long we have been up (8 hours is the maximum):
  uptime = (millis()/3600000);
  if (uptime > 8) {
    uptime = 8;
  }
  if (uptime <3){
    digitalWrite(notreadyled, HIGH);
  }
  else {
    digitalWrite(notreadyled, LOW);
  }
} //ok, end of once-per-second stuff.

// ----------------------------------
// Now the once per half hour stuff:
// ----------------------------------
if (millis() - halfhourcounter > 1800000) {

  //-----
  // update the 16 last half-hourly readings:
  //-----
  halfhourcounter = millis();
  for (int i = 15; i>0; i--) {
    baromem[i] = baromem[i-1];
  }
  baromem[0] = (bmp.readPressure());

  //-----
  // calculate if it's dropping or climbing in 1 AND 3 hours, and how fast:
  //-----
  barodiff1h = (baromem[0] - baromem[2]);
  barodiff3h = (baromem[0] - baromem[6]);

  //------
  // Display 3 hour "rising now" or "falling now" led:
  //------
  if (uptime <3){                          // turn on the "not ready" led
    digitalWrite (notreadyled, HIGH);
    digitalWrite (delta1led, LOW);
    digitalWrite (delta2led, LOW);
    digitalWrite (delta3led, LOW);
  }
  else {
    digitalWrite (notreadyled, LOW);      // turn off the "not ready" led
    if (digitalRead(A1) == HIGH) {
      show3hourtrend();
    }
    else {
      show1hourtrend();
    }
    // ...and now display the 3 hour trend:
    show3hourchange();
  }  // end of 3-hour trend display

  //predict wind:
  if (barodiff3h < -600) {
    digitalWrite (predictwindled, HIGH);
  }
  else {
```

```
      digitalWrite (predictwindled, LOW);
    }

    //predict precipitation:
    if (barodiff3h < -350) {
      digitalWrite (predictrainled, HIGH);
    }
    else {
      digitalWrite (predictrainled, LOW);
    }

  }  //end of once-per-half-hour stuff

  // ----------------------------------------
  // Now, any code that has to run every loop:
  // ----------------------------------------
  if (digitalRead(pushbutton) == LOW) {
    // display 3h air pressure difference instead of current air pressure
    // &display 1hr trend LEDS also
    // maybe somehow display.showNumberDec (uptime, false, 2, 2);  //show number of hours logged
    show1hourtrend();
    display.showNumberDec (int(barodiff3h/100), false, 4, 0);
    delay(1000);
    //now back to normal display:
    show3hourtrend();   //back to normal settings
    show3hourchange();
    display.showNumberDec (barometer, false, 4, 0);
  }

  cyclecounter = cyclecounter +1;  //to facilitate counting the cycles/second

} //end of loop


// ---------------------------------------------------------------------------------------------
// THE FUNCTIONS:
// ---------------------------------------------------------------------------------------------

void show1hourtrend() {
    if (barodiff1h > 50) {
      digitalWrite (risingled, HIGH);
      digitalWrite (fallingled, LOW);
    }
    else if (barodiff1h < -50) {
      digitalWrite (risingled, LOW);
      digitalWrite (fallingled, HIGH);
    }
    else {
      digitalWrite (risingled, LOW);
      digitalWrite (fallingled, LOW);
    }
}


void show3hourtrend() {
    if (barodiff3h > 50) {
      digitalWrite (risingled, HIGH);
      digitalWrite (fallingled, LOW);
    }
```

```
    else if (barodiff3h < -50) {
      digitalWrite (risingled, LOW);
      digitalWrite (fallingled, HIGH);
    }
    else {
      digitalWrite (risingled, LOW);
      digitalWrite (fallingled, LOW);
    }
}


void show3hourchange() {
  if (barodiff3h > 600) {
    //RISING VERY RAPIDLY
    digitalWrite (delta1led, LOW);
    digitalWrite (delta2led, LOW);
    digitalWrite (delta3led, HIGH);
  }
  else if (barodiff3h > 350) {
    //RISING QUICKLY
    digitalWrite (delta1led, LOW);
    digitalWrite (delta2led, HIGH);
    digitalWrite (delta3led, LOW);
  }
  else if (barodiff3h > 150) {
    //RISING
    digitalWrite (delta1led, HIGH);
    digitalWrite (delta2led, LOW);
    digitalWrite (delta3led, LOW);
  }
  else if (barodiff3h < -600) {
    //FALLING VERY RAPIDLY, >600
    digitalWrite (delta1led, LOW);
    digitalWrite (delta2led, LOW);
    digitalWrite (delta3led, HIGH);
  }
  else if (barodiff3h < -350) {
    //FALLING QUICKLY, 350-600
    digitalWrite (delta1led, LOW);
    digitalWrite (delta2led, HIGH);
    digitalWrite (delta3led, LOW);
  }
  else if (barodiff3h < -150) {
    //FALLING, 150-350
    digitalWrite (delta1led, HIGH);
    digitalWrite (delta2led, LOW);
    digitalWrite (delta3led, LOW);
  }
  else {
    //NC
    //must be green led, 0-150
    digitalWrite (delta1led, LOW);
    digitalWrite (delta2led, LOW);
    digitalWrite (delta3led, LOW);
  }
}
```