SYNACORP TRADING & SERVICES
No.9, 1st Floor, Lorong 1/SS2, Bandar Tasek Mutiara, 14120 Simpang Ampat, S.Prai (S), Penang
Tel : +604.502.1726 Hunting Line : 012.403.3474 Fax : +604.502.1726
(Website) http://www.synacorp.my (Email) sales@synacorp.com.my

BMP280 3.3V Digital Barometric Pressure Sensor Module

Introduction

Bosch has stepped up their game with their new BMP280 sensor, an environmental sensor with temperature, barometric pressure that is the next generation upgrade to the BMP085/BMP180/BMP183. This sensor is great for all sorts of weather sensing and can even be used in both I2C and SPI

This precision sensor from Bosch is the best low-cost, precision sensing solution for measuring barometric pressure with ±1 hPa absolute accuraccy, and temperature with ±1.0°C accuracy. Because pressure changes with altitude, and the pressure measurements are so good, you can also use it as an altimeter with ±1 meter accuracy

The BME280 is the next-generation of sensors from Bosch, and is the upgrade to the BMP085/BMP180/BMP183 - with a low altitude noise of 0.25m and the same fast conversion time. It has the same specifications, but can use either I2C **or** SPI. For simple easy wiring, go with I2C. If you want to connect a bunch of sensors without worrying about I2C address collisions, go with SPI.

Specification

- 100% new and good quality
- Pressure Range: 300 - 1100 hPa
- Package: 8-Pin LGA metal-lid
- Relative Accuracy: +- 0.12 hPa, equiv. to +- 1m
- Absolute Accuracy: typ. +- 1 hPa
- Temperature Coefficient offset: 1.5Pa/K, equi. to 12.6 cm/K
- Digital Interfaces: I2C (up to 3.4 MHz), SPI (3 and 4 wire, up to 10MHZ)
- Current Comsumption: 2.7uA @ 1Hz sampling rate
- Temperature Range: -40C to +85C

Packing List

- BMP280 3.3V Digital Barometric Pressure Sensor Module

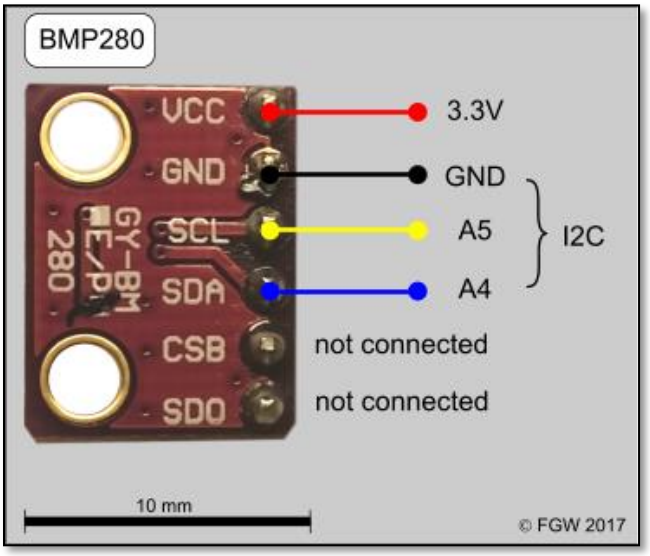
Requirements

It can be interface with any microcontroller such as PIC, SK40C, SK28A, SKds40A, Arduino series.

Necessary hardware to follow this guide:

- Arduino Uno
- BMP280 sensor Sensor module
- Male-Female/Female-Female jumper wire

Pin Assignment



Hardware Interface/Setup

Use this wiring if you want to connect via I2C interface

- Connect Vin to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
- Connect GND to common power/data ground
- Connect the SCK pin to the I2C clock SCL pin on your Arduino. On an UNO & '328 based Arduino, this is also known as A5,
- Connect the SDI pin to the I2C data SDA pin on your Arduino. On an UNO & '328 based Arduino, this is also known as A4

Example Code

This is example code for moisture sensor module. The full code can be download at https://learn.adafruit.com/adafruit-bmp280-barometric-pressure-plus-temperature-sensor-breakout/arduino-test

```
#include <Wire.h>

#define BME280_ADDRESS 0x76
unsigned long int hum_raw,temp_raw,pres_raw;
signed long int t_fine;

uint16_t dig_T1;
 int16_t dig_T2;
 int16_t dig_T3;
uint16_t dig_P1;
 int16_t dig_P2;
 int16_t dig_P3;
 int16_t dig_P4;
 int16_t dig_P5;
 int16_t dig_P6;
 int16_t dig_P7;
 int16_t dig_P8;
 int16_t dig_P9;
 int8_t  dig_H1;
 int16_t dig_H2;
 int8_t  dig_H3;
 int16_t dig_H4;
 int16_t dig_H5;
 int8_t  dig_H6;

void setup()
{
   uint8_t osrs_t = 1;           //Temperature oversampling x 1
   uint8_t osrs_p = 1;           //Pressure oversampling x 1
   uint8_t osrs_h = 1;           //Humidity oversampling x 1
   uint8_t mode = 3;             //Normal mode
   uint8_t t_sb = 5;            //Tstandby 1000ms
   uint8_t filter = 0;          //Filter off
   uint8_t spi3w_en = 0;         //3-wire SPI Disable

   uint8_t ctrl_meas_reg = (osrs_t << 5) | (osrs_p << 2) | mode;
   uint8_t config_reg    = (t_sb << 5) | (filter << 2) | spi3w_en;
   uint8_t ctrl_hum_reg  = osrs_h;
```

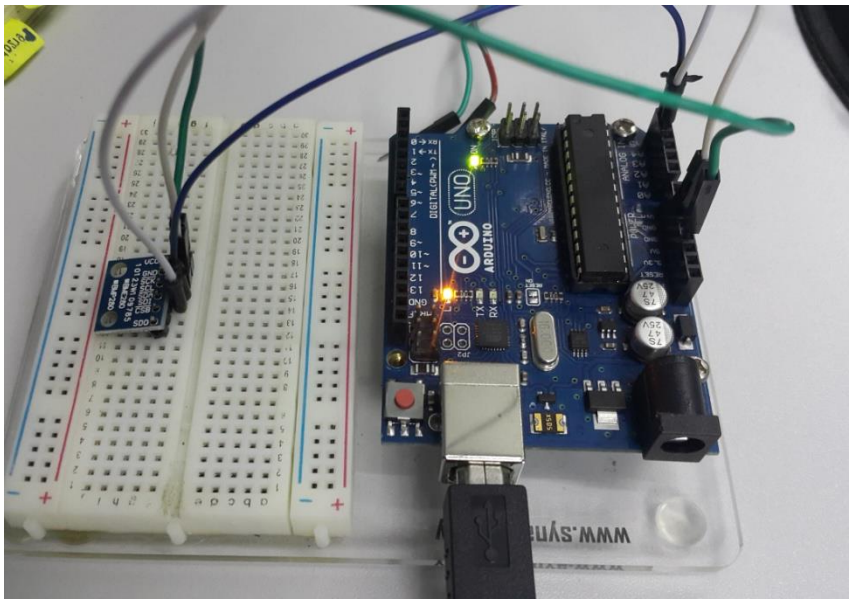
```
  Serial.begin(9600);
  Wire.begin();

  writeReg(0xF2,ctrl_hum_reg);
  writeReg(0xF4,ctrl_meas_reg);
  writeReg(0xF5,config_reg);
  readTrim();                    //
}


void loop()
{
  double temp_act = 0.0, press_act = 0.0,hum_act=0.0;
  signed long int temp_cal;
  unsigned long int press_cal,hum_cal;

  readData();

  temp_cal = calibration_T(temp_raw);
  press_cal = calibration_P(pres_raw);
  hum_cal = calibration_H(hum_raw);
  temp_act = (double)temp_cal / 100.0;
  press_act = (double)press_cal / 100.0;
  hum_act = (double)hum_cal / 1024.0;
  Serial.print("TEMP : ");
  Serial.print(temp_act);
  Serial.print(" DegC  PRESS : ");
  Serial.print(press_act);
  Serial.print(" hPa  HUM : ");
  Serial.print(hum_act);
  Serial.println(" %");

  delay(1000);
}
void readTrim()
{
  uint8_t data[32],i=0;                    // Fix 2014/04/06
  Wire.beginTransmission(BME280_ADDRESS);
  Wire.write(0x88);
  Wire.endTransmission();
  Wire.requestFrom(BME280_ADDRESS,24);      // Fix 2014/04/06
  while(Wire.available()){
    data[i] = Wire.read();
    i++;
  }
```

```
  Wire.beginTransmission(BME280_ADDRESS);    // Add 2014/04/06
  Wire.write(0xA1);                          // Add 2014/04/06
  Wire.endTransmission();                    // Add 2014/04/06
  Wire.requestFrom(BME280_ADDRESS,1);        // Add 2014/04/06
  data[i] = Wire.read();                     // Add 2014/04/06
  i++;                                       // Add 2014/04/06

  Wire.beginTransmission(BME280_ADDRESS);
  Wire.write(0xE1);
  Wire.endTransmission();
  Wire.requestFrom(BME280_ADDRESS,7);        // Fix 2014/04/06
  while(Wire.available()){
    data[i] = Wire.read();
    i++;
  }
  dig_T1 = (data[1] << 8) | data[0];
  dig_T2 = (data[3] << 8) | data[2];
  dig_T3 = (data[5] << 8) | data[4];
  dig_P1 = (data[7] << 8) | data[6];
  dig_P2 = (data[9] << 8) | data[8];
  dig_P3 = (data[11]<< 8) | data[10];
  dig_P4 = (data[13]<< 8) | data[12];
  dig_P5 = (data[15]<< 8) | data[14];
  dig_P6 = (data[17]<< 8) | data[16];
  dig_P7 = (data[19]<< 8) | data[18];
  dig_P8 = (data[21]<< 8) | data[20];
  dig_P9 = (data[23]<< 8) | data[22];
  dig_H1 = data[24];
  dig_H2 = (data[26]<< 8) | data[25];
  dig_H3 = data[27];
  dig_H4 = (data[28]<< 4) | (0x0F & data[29]);
  dig_H5 = (data[30] << 4) | ((data[29] >> 4) & 0x0F); // Fix 2014/04/06
  dig_H6 = data[31];                                  // Fix 2014/04/06
}
void writeReg(uint8_t reg_address, uint8_t data)
{
  Wire.beginTransmission(BME280_ADDRESS);
  Wire.write(reg_address);
  Wire.write(data);
  Wire.endTransmission();
}


void readData()
{
```

```
  int i = 0;
  uint32_t data[8];
  Wire.beginTransmission(BME280_ADDRESS);
  Wire.write(0xF7);
  Wire.endTransmission();
  Wire.requestFrom(BME280_ADDRESS,8);
  while(Wire.available()){
    data[i] = Wire.read();
    i++;
  }
  pres_raw = (data[0] << 12) | (data[1] << 4) | (data[2] >> 4);
  temp_raw = (data[3] << 12) | (data[4] << 4) | (data[5] >> 4);
  hum_raw  = (data[6] << 8) | data[7];
}


signed long int calibration_T(signed long int adc_T)
{

  signed long int var1, var2, T;
  var1 = ((((adc_T >> 3) - ((signed long int)dig_T1<<1))) * ((signed long int)dig_T2)) >> 11;
  var2 = (((((adc_T >> 4) - ((signed long int)dig_T1)) * ((adc_T>>4) - ((signed long
int)dig_T1))) >> 12) * ((signed long int)dig_T3)) >> 14;

  t_fine = var1 + var2;
  T = (t_fine * 5 + 128) >> 8;
  return T;
}

unsigned long int calibration_P(signed long int adc_P)
{
  signed long int var1, var2;
  unsigned long int P;
  var1 = (((signed long int)t_fine)>>1) - (signed long int)64000;
  var2 = (((var1>>2) * (var1>>2)) >> 11) * ((signed long int)dig_P6);
  var2 = var2 + ((var1*((signed long int)dig_P5))<<1);
  var2 = (var2>>2)+(((signed long int)dig_P4)<<16);
  var1 = (((dig_P3 * (((var1>>2)*(var1>>2)) >> 13)) >>3) + ((((signed long int)dig_P2) *
var1)>>1))>>18;
  var1 = ((((32768+var1))*((signed long int)dig_P1))>>15);
  if (var1 == 0)
  {
    return 0;
  }
  P = (((unsigned long int)(((signed long int)1048576)-adc_P)-(var2>>12)))*3125;
  if(P<0x80000000)
```

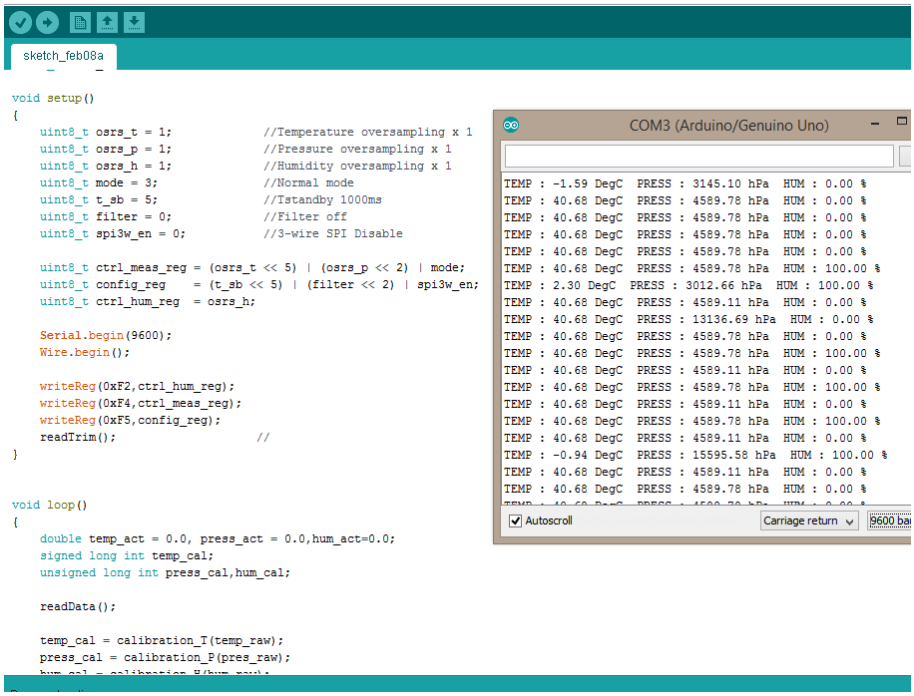
```
  {
    P = (P << 1) / ((unsigned long int) var1);
  }
  else
  {
     P = (P / (unsigned long int)var1) * 2;
  }
  var1 = (((signed long int)dig_P9) * ((signed long int)(((P>>3) * (P>>3))>>13)))>>12;
  var2 = (((signed long int)(P>>2)) * ((signed long int)dig_P8))>>13;
  P = (unsigned long int)((signed long int)P + ((var1 + var2 + dig_P7) >> 4));
  return P;
}

unsigned long int calibration_H(signed long int adc_H)
{
  signed long int v_x1;

  v_x1 = (t_fine - ((signed long int)76800));
  v_x1 = (((((adc_H << 14) -(((signed long int)dig_H4) << 20) - (((signed long int)dig_H5) *
v_x1)) +
          ((signed long int)16384)) >> 15) * (((((((v_x1 * ((signed long int)dig_H6)) >> 10) *
          (((v_x1 * ((signed long int)dig_H3)) >> 11) + ((signed long int) 32768))) >> 10) + ((
signed long int)2097152)) *
          ((signed long int) dig_H2) + 8192) >> 14));
  v_x1 = (v_x1 - (((((v_x1 >> 15) * (v_x1 >> 15)) >> 7) * ((signed long int)dig_H1)) >> 4));
  v_x1 = (v_x1 < 0 ? 0 : v_x1);
  v_x1 = (v_x1 > 419430400 ? 419430400 : v_x1);
  return (unsigned long int)(v_x1 >> 12);
}
```

Connection



VCC	3.3V
GND	GND
SCK	A5
SDI	A4

Results



```
void setup()
{
    uint8_t osrs_t = 1;             //Temperature oversampling x 1
    uint8_t osrs_p = 1;             //Pressure oversampling x 1
    uint8_t osrs_h = 1;             //Humidity oversampling x 1
    uint8_t mode = 3;               //Normal mode
    uint8_t t_sb = 5;               //Tstandby 1000ms
    uint8_t filter = 0;             //Filter off
    uint8_t spi3w_en = 0;           //3-wire SPI Disable

    uint8_t ctrl_meas_reg = (osrs_t << 5) | (osrs_p << 2) | mode;
    uint8_t config_reg    = (t_sb << 5) | (filter << 2) | spi3w_en;
    uint8_t ctrl_hum_reg  = osrs_h;

    Serial.begin(9600);
    Wire.begin();

    writeReg(0xF2,ctrl_hum_reg);
    writeReg(0xF4,ctrl_meas_reg);
    writeReg(0xF5,config_reg);
    readTrim();                     //
}


void loop()
{
    double temp_act = 0.0, press_act = 0.0,hum_act=0.0;
    signed long int temp_cal;
    unsigned long int press_cal,hum_cal;

    readData();

    temp_cal = calibration_T(temp_raw);
    press_cal = calibration_P(pres_raw);
```

```
COM3 (Arduino/Genuino Uno)
TEMP : -1.59 DegC  PRESS : 3145.10 hPa  HUM : 0.00 %
TEMP : 40.68 DegC  PRESS : 4589.78 hPa  HUM : 0.00 %
TEMP : 40.68 DegC  PRESS : 4589.78 hPa  HUM : 0.00 %
TEMP : 40.68 DegC  PRESS : 4589.78 hPa  HUM : 0.00 %
TEMP : 40.68 DegC  PRESS : 4589.78 hPa  HUM : 0.00 %
TEMP : 40.68 DegC  PRESS : 4589.78 hPa  HUM : 100.00 %
TEMP : 2.30 DegC  PRESS : 3012.66 hPa  HUM : 100.00 %
TEMP : 40.68 DegC  PRESS : 4589.11 hPa  HUM : 0.00 %
TEMP : 40.68 DegC  PRESS : 13136.69 hPa  HUM : 0.00 %
TEMP : 40.68 DegC  PRESS : 4589.78 hPa  HUM : 0.00 %
TEMP : 40.68 DegC  PRESS : 4589.78 hPa  HUM : 100.00 %
TEMP : 40.68 DegC  PRESS : 4589.11 hPa  HUM : 0.00 %
TEMP : 40.68 DegC  PRESS : 4589.78 hPa  HUM : 100.00 %
TEMP : 40.68 DegC  PRESS : 4589.11 hPa  HUM : 0.00 %
TEMP : 40.68 DegC  PRESS : 4589.78 hPa  HUM : 100.00 %
TEMP : 40.68 DegC  PRESS : 4589.11 hPa  HUM : 0.00 %
TEMP : -0.94 DegC  PRESS : 15595.58 hPa  HUM : 100.00 %
TEMP : 40.68 DegC  PRESS : 4589.11 hPa  HUM : 0.00 %
TEMP : 40.68 DegC  PRESS : 4589.78 hPa  HUM : 0.00 %
Autoscroll  Carriage return  9600 ba
```

Applications

- Enhancement of GPS navigation (e.g. time-tofirst-fix improvement, dead-reckoning, slope detection)
- Indoor navigation (floor detection, elevator detection)
- Outdoor navigation, leisure and sports applications
- Weather forecast
- Health care applications (e.g. spirometry)
- Vertical velocity indication (e.g. rise/sink speed)