Getting Started with GY-BME280

The GY-BME280 is a high precision combined digital pressure, humidity and temperature sensor module with I2C and SPI interfaces.

PACKAGE INCLUDES:

- GY-BME280 Pressure Humidity Temperature Sensor Module
- 6-pin male header

KEY FEATURES OF GY-BME PRESSURE HUMIDITY TEMPERATURE SENSOR MODULE:

- Temperature: -40 to 85°C
- Humidity: 0 to 100%
- Pressure: 300 to 1100 hPa
- Altitude: 0 to 30,000 ft
- I2C and SPI Interface
- 3.3V operation

The BME280 is a precision Bosch sensor that can be used in many applications from weather monitoring to gaming controls to altitude measurements with enough precision to know if an object has been lifted a couple of feet.

The BME280 is a higher performance version of the BMP280 with much better noise performance and also includes humidity sensing the BMP280 does not have.

Compared to the **BME280 I2C module**, this one exposes the SPI interface along with I2C but it is 3.3V operation only while the BME280 I2C module has a built in regulator and logic level shifters and will operate at either 5V or 3.3V.

The temperature measurement is used internally to compensate the pressure and humidity sensors and is also available to estimate the ambient temperature as well.

The device has numerous sampling and filtering options that can be employed to optimize the device for specific applications.

Note that this device operates at 3.3V. If it is being used with a 5V MCU, it should be powered off the 3.3V output. The good news is that the I2C interface is an open-drain interface with 10K pull-up resistors to the modules Vcc (3.3V) and so it can be used without level shifting. If you want to use the SPI interface instead, it will need to be level shifted to avoid possible damage to the module.

Measuring Temperature

The BME280 can measure temperature over the range of -40 to 80°C.

Full accuracy of ±1.0°C is obtained over the range of 0 to 65°C. Outside that range, the accuracy can decrease to ±1.5°C

Measuring Humidity

The BME280 can measure humidity over the range of 0 to 100% with an accuracy of ±3%.

The sensor can measure up to 100% humidity over the temperature range of 0 to 60°C. At very high or low temperatures, the maximum measurable humidity decreases per the graph on page 10 of the datasheet.

Measuring Pressure

The BME280 can measure pressure over the range of 300 to 1100 hPa with excellent accuracy of ±1.0 hPa.

Full accuracy is obtained over the temperature range of 0 to 65°C. This gives an elevation measurement accuracy of approximately ±1 meter. Outside that range the accuracy can decrease to ±1.7 hPa

Calculating Altitude / Elevation

The BME280 does not measure altitude directly, but it can be calculated using the pressure reading. Most libraries for this device include altitude calculation routines.

Since the device does a very good job of measuring pressure, it can do a very good job of calculating relative altitude. If you have an altitude reading with the device sitting on a table and then move it to the floor, it will show a 2 foot decrease in altitude.

If on the other hand you are trying to measure absolute altitude, such as the altitude of your table relative to sea level, things get more complicated. Since altitude is relative to sea level the device needs to know the current air pressure corrected to sea level so that it has a reference by which to calculate the altitude given the air pressure that it is currently reading.

You can get somewhat close by finding the reported air pressure from a local airport or weather service on-line which are corrected for sea level. In our example program down below, you would enter this value in the SEA_LEVEL_PRESSURE constant. In our example, we have 1013.25 loaded which is a typical reading.

Since the air pressure is constantly changing based on time, location and weather conditions, unless you have an accurate barometer corrected to sea level with you to reference, it will be difficult to get closer than 20-30 feet.

If you know the altitude that the sensor is at, you can also back into a reading by modifying the SEA_LEVEL_PRESSURE constant to give you the correct altitude reading. This might be useful if you want to make absolute measurements based on your current elevation, such as sending a rocket up into the air.

Using the I2C Interface

Because the I2C interface does not require level shifting when used with a 5V MCU, it is typically the preferred interface to use.

The **CSB** (Chip Select Bus) pin determines which bus to use. A logic HIGH selects the I2C bus. A 10K pull-up resistor on the module pulls this pin HIGH and so the I2C bus is selected by default unless this pin is grounded.

The module supports two different I2C addresses, either 0x76 or 0x77 which allows up to 2 sensors to be used on the same bus. The **SDO** pin determines which address to used. A weak pull-down resistor on the module pulls SDO low so 0x76 is the default address if the SDO pin is not connected. Connect the SDO to Vcc to select 0x77.

The **SCL** and **SDA** pins connect to the SCL and SDA pins on the MCU.

If more than 2 sensors are needed, then the SPI bus can be used.

Using the SPI Interface

To enable the SPI interface, connect **CSB** to ground.

The other connection are:

- Connect **SCL** pin to the SPI SCK on MCU
- Connect **SDA** pin to SPI SDI on MCU
- Connect **SDO** pin to SPI SDO on MCU

Note that if using this module with a 5V MCU, a logic level shifter should be added to these lines.

Module Connections

The module brings out the following connections.

1 x 6 Header

- **VCC** = 1.7 to 3.6V. Connect to 3.3V output of the MCU
- **GND** = Ground
- **SCL** =   Clock (SCL / SCK) for I2C and SPI
- **SDA** = Data (SDA / SDI) for I2C and SPI
- **CSB** = Chip Select Bus. Logic HIGH for I2C (default), logic LOW for SPI
- **SDO** = Data Out (SDO) for SPI. Sets I2C address for I2C

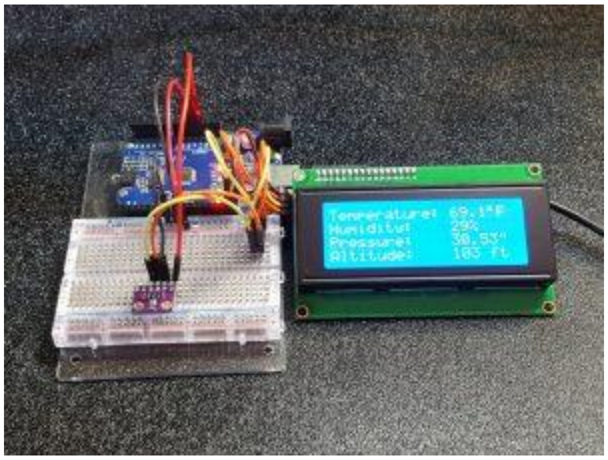
Module Assembly

The module ships with the male header strip loose. The header can be soldered to the top or bottom of the module depending on the planned use or wires can be used to make the connections.

For breadboard use, we put the headers on the bottom. Soldering is easiest if the header is inserted into a solderless breadboard to hold it in position during the soldering process.

OUR EVALUATION RESULTS:

These are nice little assemblies. The boards are high quality with gold plating.



The program below implements a simple weather station using the GY-BME280 module and a LCD2004 20 x 4 LCD to display the info.

A simpler version of the program that just dumps the information to the Serial Monitor window can be found on the **BME280 I2C page**.

There are many libraries for the BME280 sensor. In the example here, we are using the Adafruit libraries. Note that the Adafruit_sensor library will need to be manually downloaded from this link: **https://github.com/adafruit/Adafruit_Sensor**

The other libraries can be downloaded via the IDE library manager.

GY-BME280 Weather Station Example Program

```
/*
   BME280 Weather Station application

   Uses LCD2004 with I2C interface for display.
   Connect I2C interface of both LCD display and BME280
        SCL connects to A5 or dedicated SCL pin
        SDA connects to A4 or dedicated SDA pin
   Connect LCD Vcc to 5V and GND to ground
   Connect BME280 Vcc to 3.3V and GND to ground
   Need to install library LiquidCrystal_I2C
   Need to install library Adafruit_BME280
   Need to manually install library Adafruit_Sensor
*/
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <Wire.h>
```

```
#include <LiquidCrystal_I2C.h>

float temperature;
float humidity;
float pressure;
float altitude;

float const ALTITUDE = 62.0;               // Altitude at my location in meters
float const SEA_LEVEL_PRESSURE = 1013.25;  // Pressure at sea level

Adafruit_BME280 bme; // I2C

LiquidCrystal_I2C lcd(0x27, 20, 4);   // I2C address, 20 char x 4 lines
//==========================================================================
//  Initialization
//==========================================================================
void setup(void) {

  lcd.begin();
  lcd.clear();           // Clear display
  lcd.backlight();       // Make sure backlight is on
  lcd.print("Reading sensor");

  bool status;

  // default settings
  status = bme.begin(0x76);  // The I2C address of the sensor is 0x76
  if (!status) {             // Loop if sensor not found
    lcd.clear();
    lcd.print("Error. Check");
    lcd.setCursor(0, 1);
    lcd.print("connections");
    while (1);
  }
  // Print non-changing info on LCD once
  lcd.clear();           // Clear display
  lcd.setCursor(0, 0);  //Set cursor to character 0 on line 0
  lcd.print("Temperature: ");
  lcd.setCursor(0, 1);  //Set cursor to line 1
  lcd.print("Humidity: ");
  lcd.setCursor(0, 2); // Set cursor to line 2
  lcd.print("Pressure: ");
  lcd.setCursor(0, 3); // Set cursor to line 3
  lcd.print("Altitude:   ");
}
//=================================================================================
//  Main
//=================================================================================
void loop() {

  getPressure();   // Get sensor data and print to LCD
  getHumidity();
  getTemperature();
  getAltitude();

  delay(2000);     // Update readings every 2 seconds
}
//=================================================================================
//  getTemperature - Subroutine to get and print temperature
//=================================================================================
void getTemperature()
{
  temperature = bme.readTemperature();
  temperature = temperature * 9 / 5 + 32; // Convert C to F
  String temperatureString = String(temperature, 1); // One decimal position
  lcd.setCursor(13, 0);           // Move to start of reading
  lcd.print("       ");           // Clear old reading
  lcd.setCursor(13, 0);           // Reset cursor location
  lcd.print(temperatureString);   // Write new reading
  lcd.print((char)223);           // Degree symbol
  lcd.print("F ");
}
//=================================================================================
//  getHumidity - Subroutine to get and print humidity
```

```
//==============================================================================
void getHumidity()
{
  humidity = bme.readHumidity();
  String humidityString = String(humidity, 0);
  lcd.setCursor(13, 1);
  lcd.print("       ");
  lcd.setCursor(13, 1);
  lcd.print(humidityString);
  lcd.print("%");
}
//==============================================================================
//  getPressure - Subroutine to get and print pressure
//==============================================================================
void getPressure()
{
  pressure = bme.readPressure();
  pressure = bme.seaLevelForAltitude(ALTITUDE, pressure);
  pressure = pressure / 3386.39;    // Convert hPa to in/Hg
  lcd.setCursor(13, 2);
  lcd.print("       ");
  lcd.setCursor(13, 2);
  String pressureString = String(pressure, 2);
  lcd.print(pressureString);
  lcd.print(""");
}
//==============================================================================
//  getAltitude - Subroutine to get and print temperature
//==============================================================================
void getAltitude()
{
  altitude = bme.readAltitude(SEA_LEVEL_PRESSURE);
  altitude = altitude * 3.28084;  // Convert meters to feet
  lcd.setCursor(13, 3);
  lcd.print("      ");
  lcd.setCursor(13, 3);
  String altitudeString = String(altitude, 0);
  lcd.print(altitudeString);
  lcd.print(" ft");
}
```

TECHNICAL SPECIFICATIONS

Operating Ratings		
Vcc Range		1.7 – 3.6V (3.3V Typical)
Humidity	Range	0 – 100%
	Accuracy	±3%
Pressure	Range	300 – 1100hPa
	Accuracy	±1hPa
Temperature	Range	-40 – 80°C
	Accuracy	±1.0C
Dimensions	L x W (PCB)	15.5 x 11.5mm (0.61 x 0.45")
Datasheet	Bosch	**BME280**