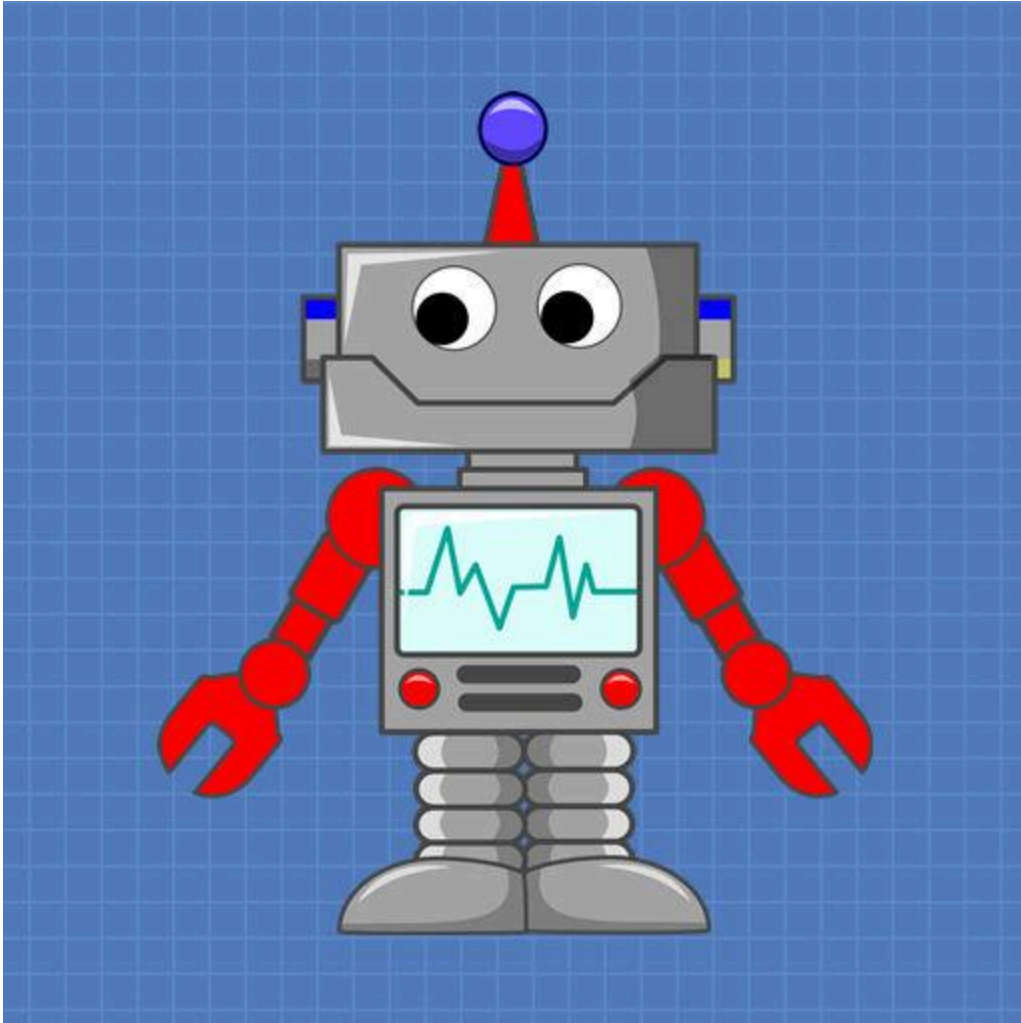


Air Pressure Sensors



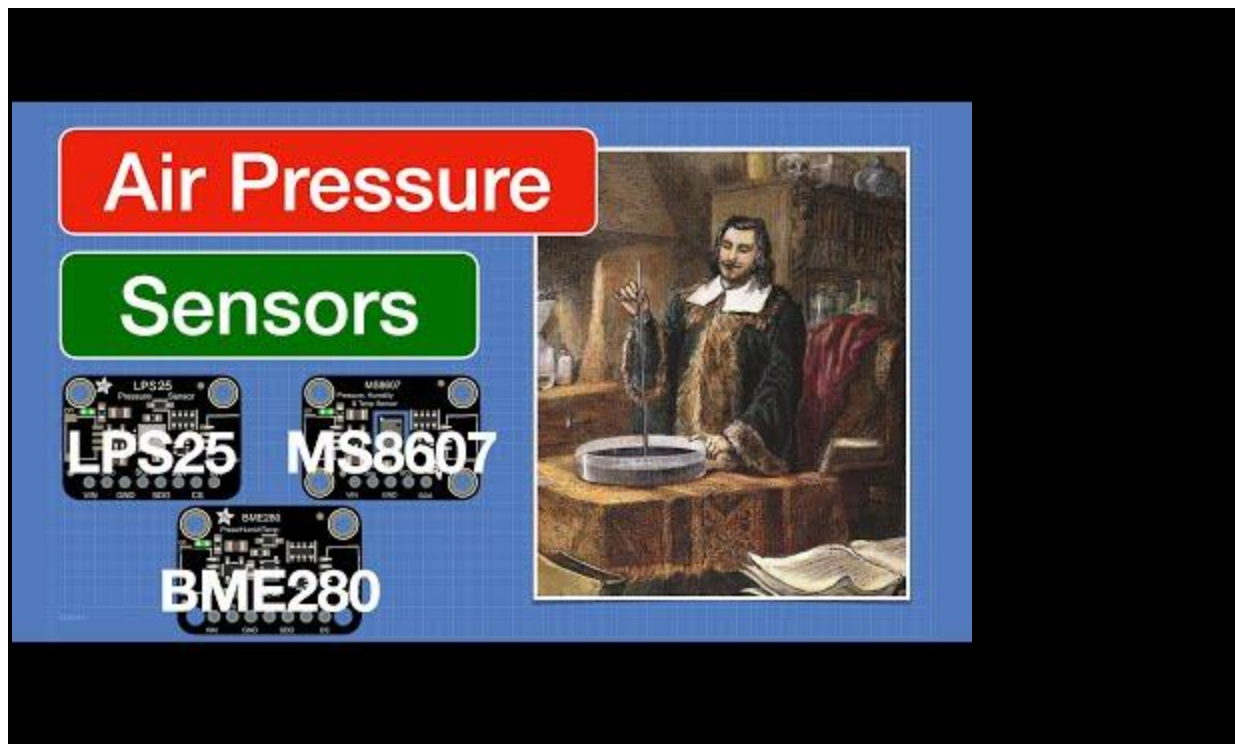
DroneBot Workshop Tutorial

<https://dronebotworkshop.com>

Table of Contents [\[show\]](#)



Air pressure is the weight of the air column above a point. It is a fundamental environmental measurement used for altitude estimation, weather forecasting, indoor climate monitoring, and aircraft and drone sensors. Air pressure can also be easily measured with a microcontroller and a suitable sensor. Today, we will examine three such sensors.



Introduction

Air pressure is (literally) all around us! It's the invisible force exerted by the weight of the atmosphere pressing down on everything on Earth. Measuring it helps us predict the weather, estimate altitude, and even stabilize drones and aircraft.



Today, we'll see how to sense air pressure using three high-quality Adafruit sensors:

- **MS8607** – A precision pressure, humidity, and temperature sensor from TE Connectivity.
- **BME280** – A versatile environmental sensor from Bosch that measures pressure, humidity, and temperature.
- **LPS25HB** – A compact barometric pressure sensor from STMicroelectronics.

We'll use them with several popular microcontrollers — including the Seeeduino XIAO ESP32-C3, Raspberry Pi Pico 2, ESP32-C6 DevKit, and Arduino Uno R4 WiFi — to see how each performs and how you can integrate them into your own IoT or weather-station projects.

Sensing Air Pressure

Air pressure is the weight of the air above a given area. At sea level under standard conditions, this pressure equals 1013.25 hPa (hectopascals), also written as 1013.25

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

mbar. As altitude increases, air pressure drops because there are fewer air molecules above you.

Air pressure decreases consistently with altitude, dropping by approximately 12 Pascals per meter near sea level. This characteristic enables altimeters to determine height above sea level by measuring atmospheric pressure. Pressure variations also arise from weather systems; high-pressure systems are typically associated with clear skies, whereas low-pressure systems often bring precipitation. Recognizing these patterns enables meteorologists to forecast weather days in advance.

Measurement Units and Conversions

You'll see pressure measured in several different units. The international standard (SI) unit is the *Pascal* (Pa). Because a single Pascal is a tiny amount of pressure, we almost always use *Hectopascals* (hPa), where 1 hPa equals 100 Pascals.

You may hear meteorologists talk about air pressure in *Millibars* (mbar). The good news is that they are exactly the same! **1 hPa = 1 mbar**.

There are other units for measuring air pressure as well. Inches of Mercury (inHg) is a common measurement in US weather reporting, where standard sea level pressure equals 29.92 inHg. This unit originated in mercury barometers, in which atmospheric pressure balanced a column of mercury.

This table summarizes Air pressure Measurement Units:

Unit	Equivalent	Typical Use
1 Pa (Pascal)	1 N/m ²	SI base unit
1 hPa = 1 mbar	100 Pa	Weather reports
1 atm	1013.25 hPa	Standard atmosphere
1 inHg	3386 Pa	Aviation, legacy units
1 psi	6895 Pa	Industrial measurements

For our projects, we'll primarily use hPa.

How Pressure Sensors Work

For centuries, people have known that air exerts pressure, but it wasn't until the 17th century that scientists learned how to measure it. Today, we have sensors small enough to fit on a fingernail, but they're all descendants of a few simple (and ingenious) early instruments.

Air Pressure

 Evangelista Torricelli Invented Mercury Barometer in 1643 <div style="background-color: black; color: white; padding: 5px; margin-top: 10px;">Glass tube inverted into a dish of Mercury. Air pressure on dish causes Mercury in tube to rise.</div>	 Lucien Vidi Invented Aneroid Barometer in 1844 <div style="background-color: black; color: white; padding: 5px; margin-top: 10px;">Sealed metal vacuum capsule that changes shape under air pressure.</div>
---	--

[DroneBotWorkshop.com](https://dronebotworkshop.com) 

Mercury Barometer

In 1643, Italian physicist Evangelista Torricelli invented the mercury barometer, the first true pressure-sensing device. He observed that when a long glass tube filled with mercury was inverted into a dish of the same liquid, the mercury column would consistently stop at approximately 760 mm (29.92 inches) above the surface. Torricelli accurately concluded that the surrounding air exerted pressure on the mercury in the dish, thereby supporting the column within the tube.

Aneroid Barometer

In 1843, French scientist Lucien Vidi invented the aneroid barometer, revolutionizing pressure measurement by eliminating mercury. The term “aneroid” comes from the Greek “a-neros,” meaning “without liquid.”

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

The core of an aneroid barometer is a flexible, sealed metal capsule, known as an aneroid cell, from which most of the air has been removed. This makes the cell highly sensitive to external pressure changes, causing it to either squeeze or expand. These minute movements are then amplified by a system of levers and springs, which are connected to a needle on a dial.

However, aneroid mechanisms are less accurate than mercury barometers, typically showing a deviation of $\pm 1\text{-}2$ hPa, and may require recalibration due to drift over time.

Piezoresistive Sensors

These use silicon crystals whose electrical resistance changes under mechanical stress. Tiny conductive paths, known as piezoresistors or strain gauges, are embedded directly into the surface of the silicon diaphragm. When the diaphragm flexes under pressure, it stretches or compresses these resistors, changing their electrical resistance.

These sensors are accurate, fast-responding, and can be miniaturized.

Capacitive Sensors

In a capacitive sensor, the flexible silicon diaphragm acts as one plate of a capacitor. A second, fixed plate is positioned just beneath it inside the vacuum-sealed cavity. When pressure causes the diaphragm to flex, the distance (the gap) between the two plates changes, altering the capacitance.

These sensors offer excellent sensitivity and stability, making them well-suited for high-precision applications.

<https://dronebotworkshop.com>

MEMS (Micro-Electro-Mechanical Systems) Sensors

MEMS sensors integrate mechanical and electrical components on a single silicon chip.

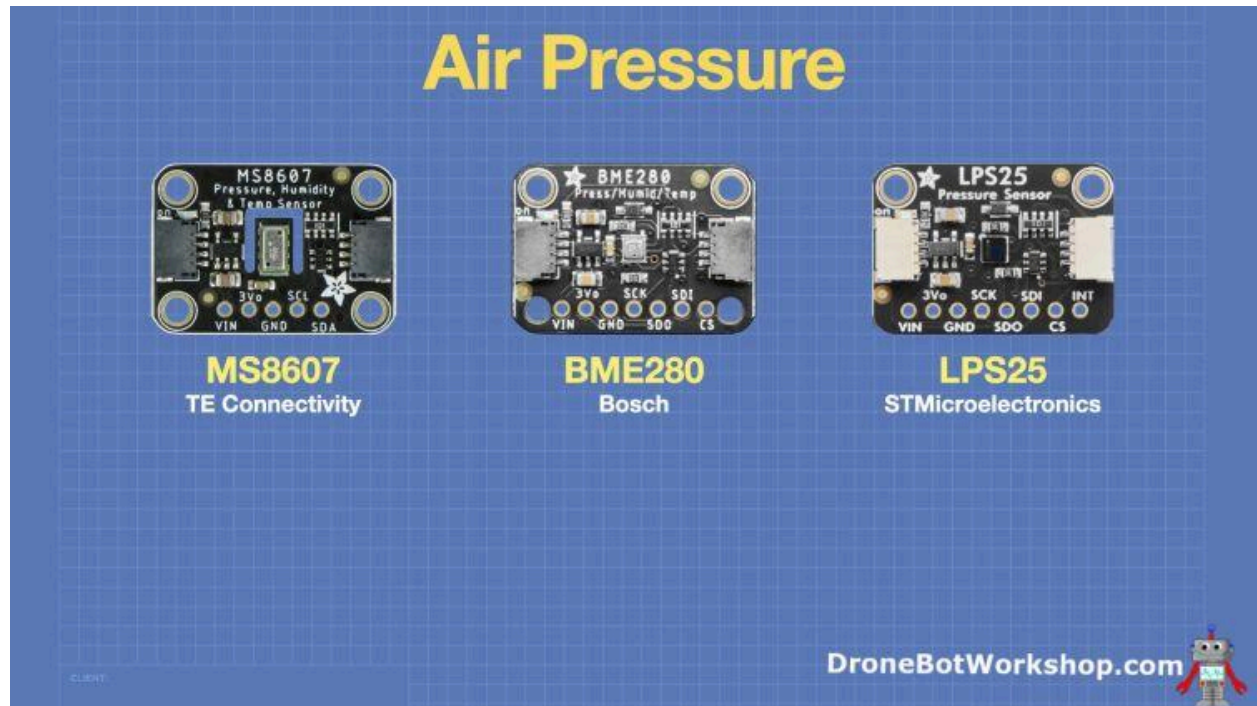
A tiny etched diaphragm (often just micrometers across) deflects under pressure, with the deflection measured by piezoresistive, capacitive, or piezoelectric elements.

MEMS sensors have many advantages:

- Extremely small (can fit in smartphones)
- Low power consumption
- Mass production keeps costs very low
- Digital output interfaces (I²C, SPI)
- Can include temperature compensation
- High reliability with no moving parts

The sensors we are using today are all MEMS sensors. These modules are more than just sensors; they are complete “laboratories on a chip.”

- A MEMS pressure element measures deflection with sub-micron precision.
- An onboard temperature sensor provides compensation data.
- A digital signal processor (DSP) converts and linearizes the output.
- Data is sent directly over I²C or SPI, eliminating the need for analog circuitry.



We will be using sensor modules from Adafruit, but the sensors themselves are from three different manufacturers:

- **Adafruit LPS25HB** – From STMicroelectronics – MEMS sensor with ± 0.2 hPa accuracy.
- **Adafruit BME280** – From Bosch – A combined pressure, humidity, and temperature sensor.
- **Adafruit MS8607** – From TE Connectivity – Like the BME280, with built-in calibration constants.

These devices can measure pressure changes as small as 0.01 hPa, corresponding to altitude changes of less than 10 cm — far beyond what early scientists could have imagined, and much easier (and safer) to carry around than a large tube of Mercury!

Adafruit MS8607



The MS8607 is a calibrated PHT module: pressure, humidity, temperature. This is an I²C device with the following specifications:

Pressure Sensing:

- Range: 300 to 1200 hPa
- Resolution: 0.016 hPa
- Accuracy: ± 2 hPa
- Response time: 1ms

Temperature Sensing:

- Range: -40°C to $+85^{\circ}\text{C}$
- Resolution: 0.01°C
- Accuracy: $\pm 2^{\circ}\text{C}$

Humidity Sensing:

- Range: 0 to 100% RH
- Resolution: 0.04% RH
- Accuracy: $\pm 3\%$ RH

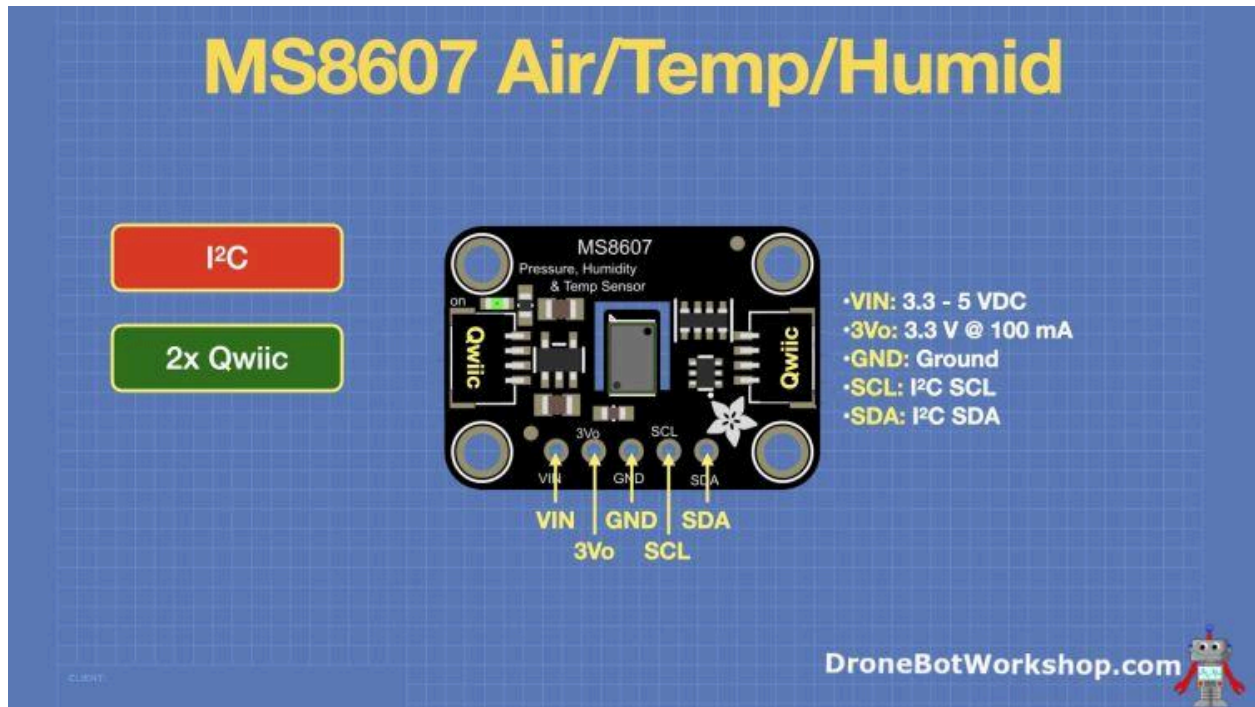
Electrical Characteristics:

- Supply Voltage: 1.5V to 3.6V
- Current Consumption: 1.4 μ A at 1 Hz sampling
- Interface: I²C (up to 400 kHz)
- Default I²C Address: 0x76
- Dimensions: 17.8mm x 15.3mm x 2.9mm

The MS8607 is an excellent choice when you need a complete environmental snapshot. It combines a pressure/temperature sensor and a separate humidity sensor in one compact, factory-calibrated package.

Adafruit MS8607 Module Pinout

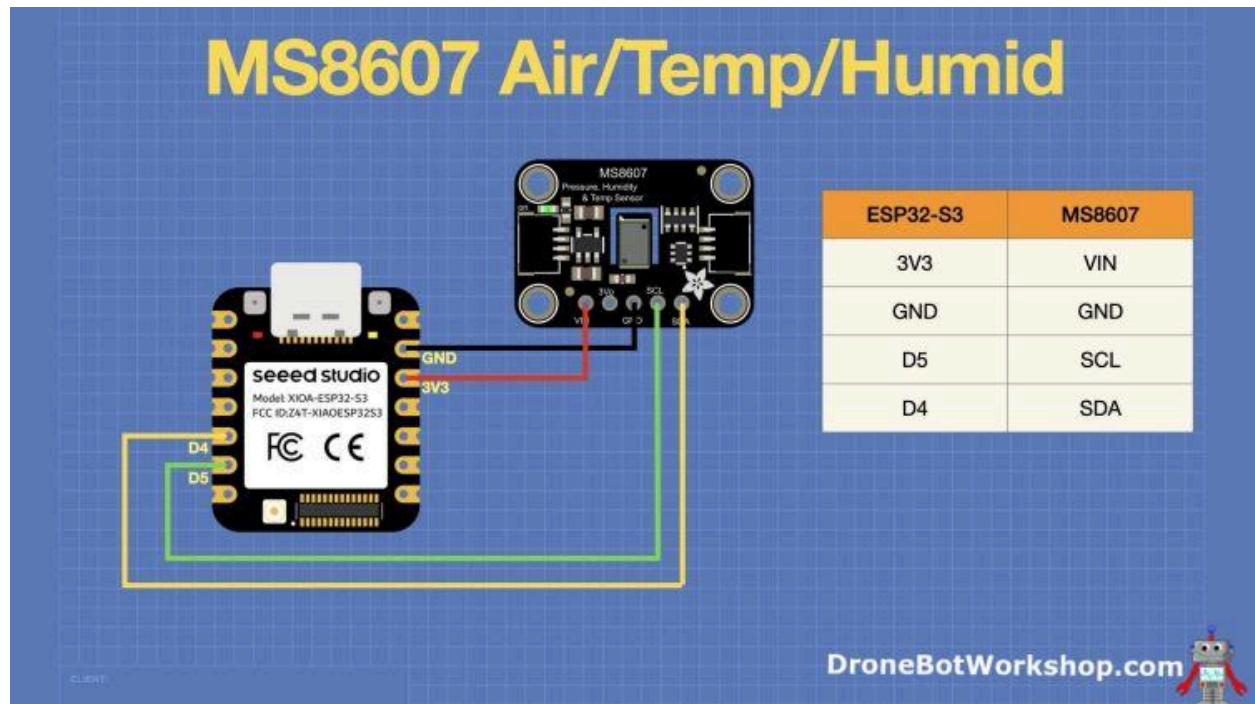
Here is the pinout of the Adafruit MS8607 module:



Note that you can connect via I²C using either the pins or the Qwiic connectors. The *VIN* should match your logic voltage.

MS8607 Hookup (SPI Bus)

We will use this sensor with a Seeedstudio XIAO ESP32-S3 board, a small, inexpensive microcontroller. Here is how we hook everything up:



Pull-up resistors should not be required, as the Adafruit module already has them.

MS8607 Code

We are just going to write a simple sketch to read the sensor values and print them on the Serial Monitor.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

```
/*  
  MS8607 Air Pressure Sensor Demo  
  air_ms8607-demo.ino  
  Uses Adafruit MS8607 Library  
  Uses Seeeduino XIAO ESP32-S3  
  
  DroneBot Workshop 2025  
  https://dronebotworkshop.com  
*/  
  
// Include Required Libraries  
#include <Wire.h>  
#include <Adafruit_MS8607.h>  
  
// Create object for sensor  
Adafruit_MS8607 ms;  
  
void setup() {  
  Serial.begin(115200);  
  // Default I2C on XIAO ESP32-C3: SDA=GPIO6 (D4), SCL=GPIO7 (D5)  
  Wire.begin();  
  
  Serial.println("MS8607 Demo - XIAO ESP32-C3");  
  if (!ms.begin(&Wire)) {  
    Serial.println("MS8607 not found! Check wiring and I2C address (0x76).");  
    while (1) delay(10);  
  }  
  Serial.println("MS8607 found!");  
}
```

<https://dronebotworkshop.com>

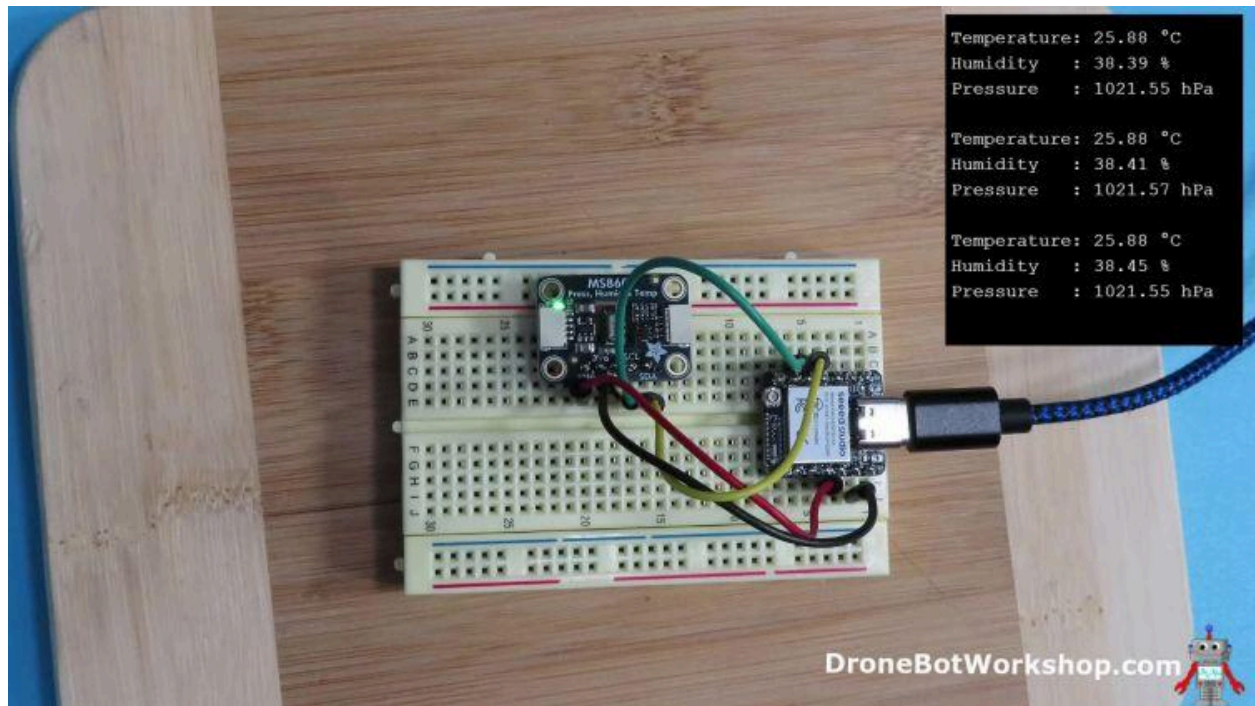
For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

```
}  
  
void loop() {  
    // Define sensor event  
    sensors_event_t p, t, h; // pressure, temperature, humidity  
  
    if (ms.getEvent(&p, &t, &h)) {  
        // On ESP32, Serial.printf is supported. Use print/println if you prefer.  
        Serial.printf("Temperature: %.2f °C\n", t.temperature);  
        Serial.printf("Humidity    : %.2f %%\n", h.relative_humidity);  
        Serial.printf("Pressure   : %.2f hPa\n\n", p.pressure);  
    } else {  
        Serial.println("MS8607 read failed");  
    }  
  
    delay(1000);  
}
```

This sketch requires the Adafruit MS8607 Library, which you can install via the Arduino IDE Library Manager.

Run the code and observe the serial monitor.

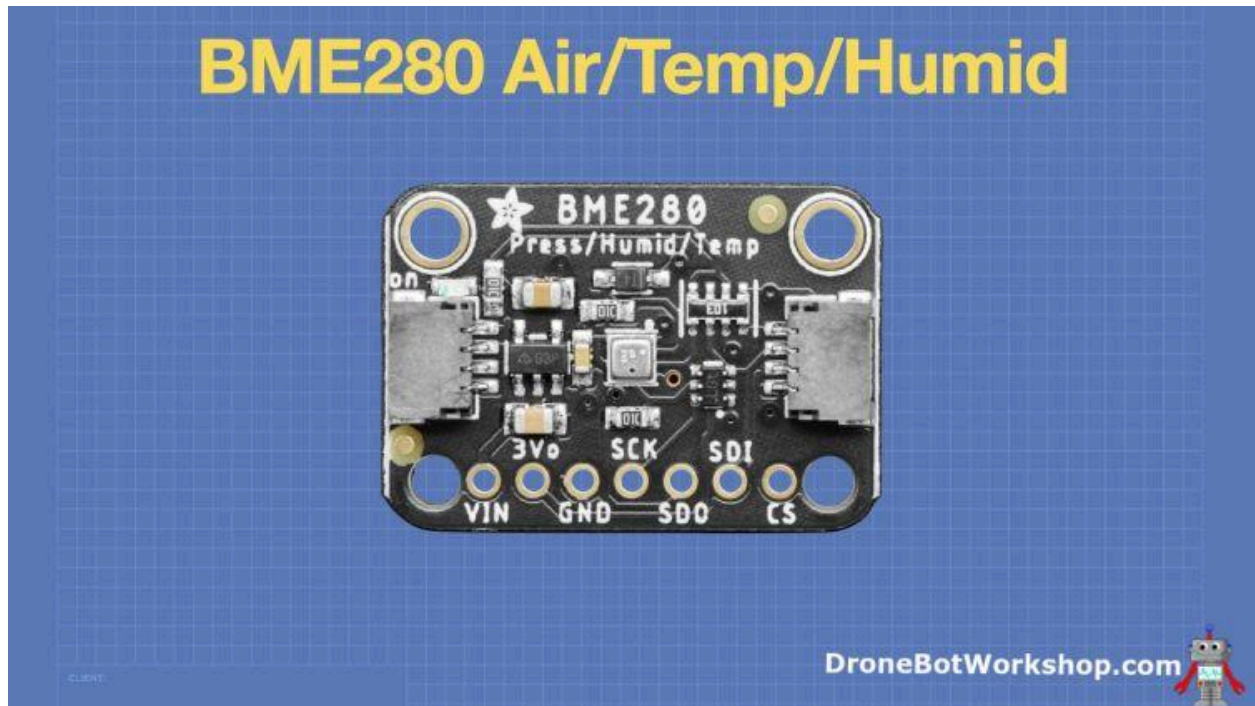
For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>



You should see the sensor values displayed.

<https://dronebotworkshop.com>

Adafruit BME280



The Adafruit BME280 is based on a popular Bosch sensor that combines three essential environmental measurements—temperature, humidity, and air pressure. It is similar in function and shape to the MS8607; however, it has both I²C and SPI interfaces.

The Adafruit BME280 has the following specifications:

Pressure Sensing:

- Range: 300 to 1100 hPa
- Resolution: 0.16 Pa
- Accuracy: ± 1 hPa (absolute), ± 0.12 hPa (relative)

Temperature Sensing:

- Range: -40°C to $+85^{\circ}\text{C}$

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

- Resolution: 0.01°C
- Accuracy: $\pm 1^\circ\text{C}$

Humidity Sensing:

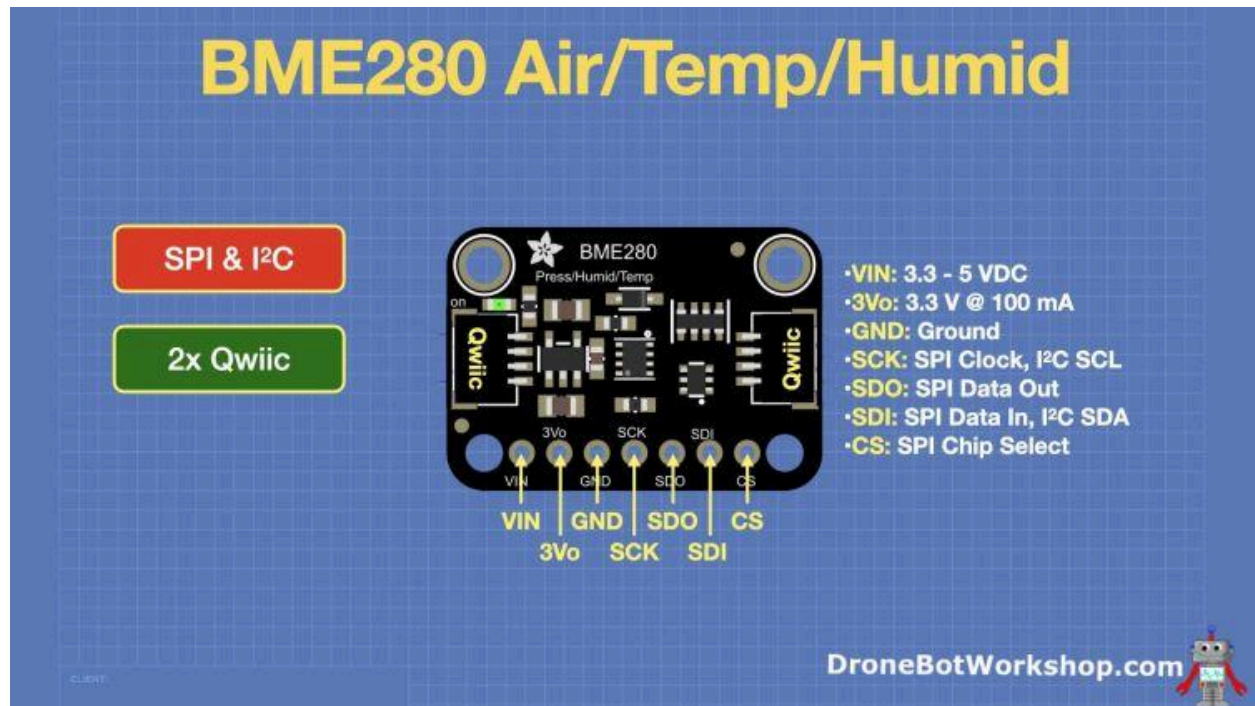
- Range: 0 to 100% RH
- Resolution: 0.008% RH
- Accuracy: $\pm 3\%$ RH

Electrical Characteristics:

- Supply Voltage: 1.71V to 3.6V (sensor), 3.3V to 5V (breakout)
- Current Consumption: 3.6 μA at 1 Hz
- Interface: I²C (up to 3.4 MHz) and SPI
- Default I²C Address: 0x77 (or 0x76 with SDO to GND)

Adafruit BME280 Module Pinout

This sensor supports both SPI and I²C, so some of the pins are multi-purpose. There is no jumper or strap to select the interface; connecting the CS and SDO pins will let the module know you want to use SPI mode.

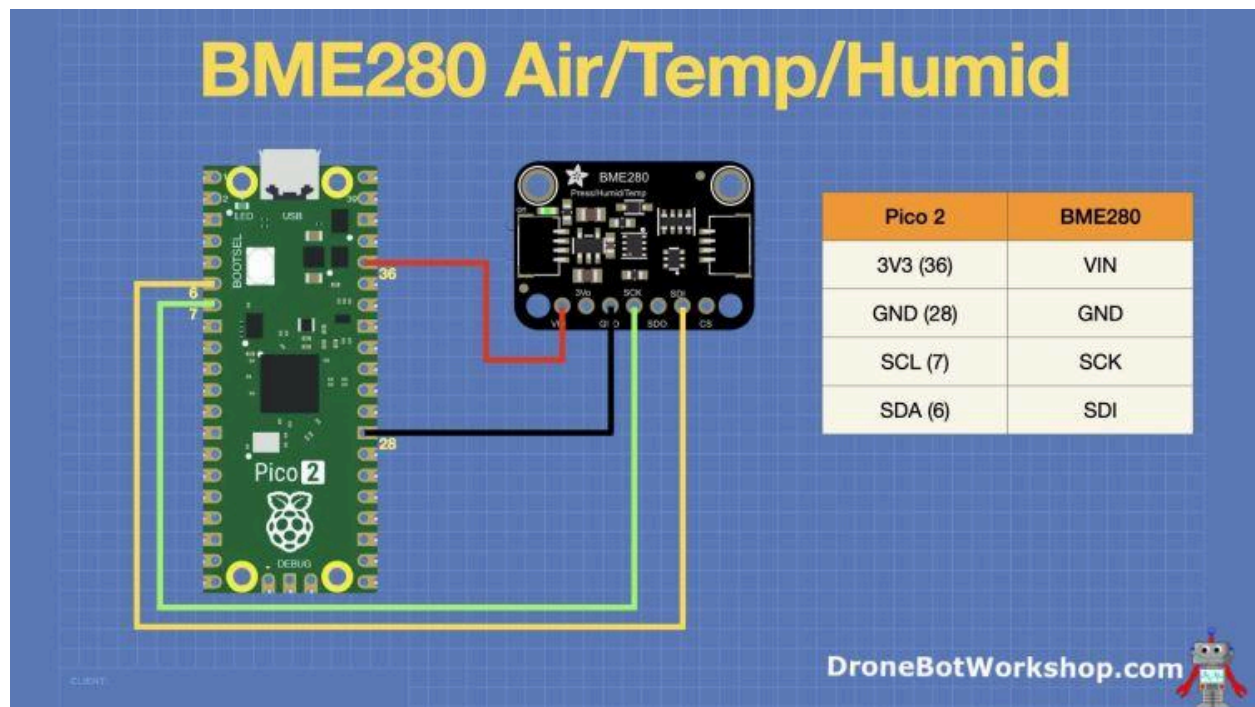


Otherwise, the pinout is similar to the previous module, including the two Qwiic connectors and the 3.3-volt output.

BME280 Hookup

We will be using a Raspberry Pi Pico 2 with the BME280 module. Of course, you could use any microcontroller, but I'm using the Pico 2 for its MicroPython support.

Here is how we hook all of this up. The sensor is connected to the Pico 2 default I²C bus.



Note that you can use any of the ground pins on the Pico 2; I just showed pin 28 to simplify the schematic drawing. Ground pins on the Pico 2 are identified by square trace edges rather than round ones.

BME280 Code (MicroPython)

We will code this in MicroPython. Unlike most Adafruit sensors, which only support CircuitPython, the BME280 has a MicroPython library available from a third-party on GitHub.

You will need to install MicroPython on the Pico 2 before you can use it. Here is what you need to do:

- Go to micropython.org/download/RPI_PICO2.
- Download the latest `.uf2` firmware file.
- Hold the **BOOTSEL** button on your Pico 2 and connect it to your computer with USB.
- A drive called **RPI-RP2** will appear.
- Drag and drop the `.uf2` file onto that drive.
- The board will reboot as a MicroPython device, ready for use in the **Thonny IDE**.

You can [download the library in a ZIP file from GitHub](#). There are two libraries in the package, `bme280_float.py` and `bme280_int.py`. You need to select one (I used the float) and rename it to `bme280.py`. You will then need to use your MicroPython IDE (I used the Thonny IDE) to copy the file to the Pico 2.

To make your life easier, I have included the renamed file in the code downloads for this article.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

Now create another MicroPython file (or copy the one provided in the article's downloads). I called mine *bme280_test.py*. Copy the following code into the file:

```
from machine import Pin, I2C
from bme280 import BME280
import time

# Pico 2 default I2C0 pins: GP5=SCL, GP4=SDA
i2c = I2C(0, scl=Pin(5), sda=Pin(4), freq=100000)

print("Scan:", [hex(a) for a in i2c.scan()])

# Explicitly specify the detected address
bme = BME280(i2c=i2c, address=0x77)

while True:
    try:
        t, p, h = bme.values
        print("Temperature:", t)
        print("Pressure:", p)
        print("Humidity:", h)
        print()
    except Exception as e:
        print("Read error:", e)

    time.sleep(2)
```

Run the file, and you should see the sensor values displayed in the Shell.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>



```
7
8 print("Scan:", [hex(a) for a in i2c.scan()])
9
10 # Explicitly specify the detected address.
11 bme = BME280(i2c=i2c, address=0x77)
12
13 while True:
14     try:
15         t, p, h = bme.values
16         print("Temperature:", t)
17         print("Pressure:", p)
18         print("Humidity:", h)
19         print()
20     except Exception as e:
21         print("Read error:", e)
22         time.sleep(2)
23
```

Shell

```
Temperature: 25.50C
Pressure: 1020.58hPa
Humidity: 34.91%

Temperature: 25.50C
Pressure: 1020.60hPa
Humidity: 34.92%
```

DroneBot Workshop

MicroPython (Raspberry Pi Pico) • Board: CDC

<https://dronebotworkshop.com>

Adafruit LPS25



If accuracy is what you need, the Adafruit LPS25 delivers exceptional pressure measurement accuracy. This is an air-pressure-only sensor, though it includes a temperature sensor used for calibration. It can work with both I²C and SPI connections, and has the following specifications:

Pressure Sensing:

- Range: 260 to 1260 hPa
- Resolution: 0.01 hPa RMS
- Accuracy: ± 0.2 hPa at 25°C
- Noise: 0.01 hPa RMS

Temperature Sensing:

- Range: -30°C to +105°C

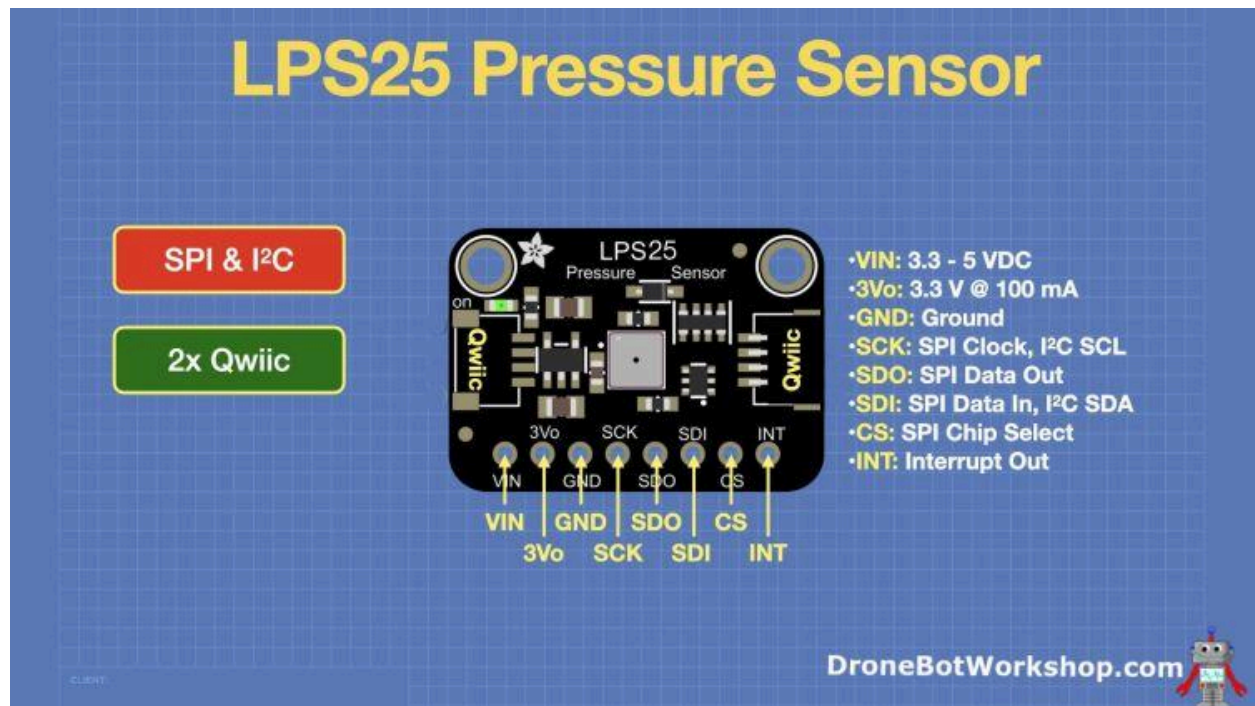
- Accuracy: $\pm 2^{\circ}\text{C}$

Electrical Characteristics:

- Supply Voltage: 3.0V to 5.5V
- Current Consumption: 25 μA in low-power mode
- Interface: I²C and SPI
- SPI Max Clock: 10 MHz
- FIFO: 32 levels
- Dimensions: 17.8mm x 15.2mm x 2.8mm

Adafruit LPS25 Module Pinout

The pinout of this module is very similar to the BME280:

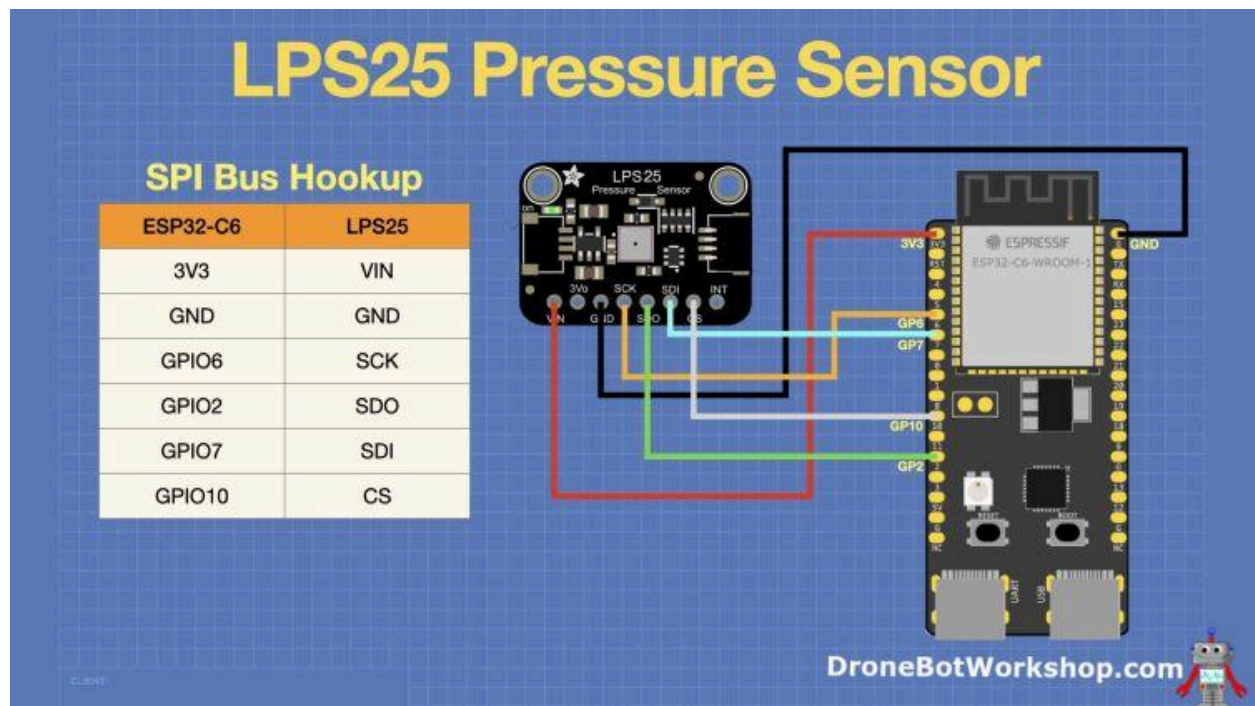


Once again, the simplest way of using this sensor is with a Qwiic connector.

LPS25 Hookup (SPI Bus)

We're going to break with tradition and use the LPS25 with the SPI bus, instead of I²C.

Here is how we will connect it to an ESP32-C6 DevKit:



This hookup uses the ESP32-C6's default SPI pins. The CS pin can be any GPIO pin; I chose GPIO10. If you use a different ESP32, you may use a different pin; just edit the code as required.

LPS25 Code

Here is some simple code for displaying the air pressure (and temperature) values from the LPS25. Note that it is written for the SPI bus.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

```
/*  
  LPS25 Air Pressure Sensor Demo  
  
  air_lps25_spi_demo.ino  
  
  Uses Adafruit Adafruit_LPS2X Library  
  
  Uses ESP32-C6 DevKit  
  
  Connected via SPI Bus  
  
  
  DroneBot Workshop 2025  
  
  https://dronebotworkshop.com  
*/
```

```
// Include Required Libraries  
  
#include <SPI.h>  
  
#include <Adafruit_LPS2X.h>  
  
  
// SPI pin definitions for ESP32-C6  
  
#define LPS25_CS    10  
  
#define LPS25_SCK   6  
  
#define LPS25_MISO  2  
  
#define LPS25_MOSI  7
```

```
// Create sensor object  
  
Adafruit_LPS25 lps;  
  
  
  
void setup() {  
  
  Serial.begin(115200);  
  
  while (!Serial) delay(10);  
  
  Serial.println("LPS25 SPI Demo");  
  
}
```

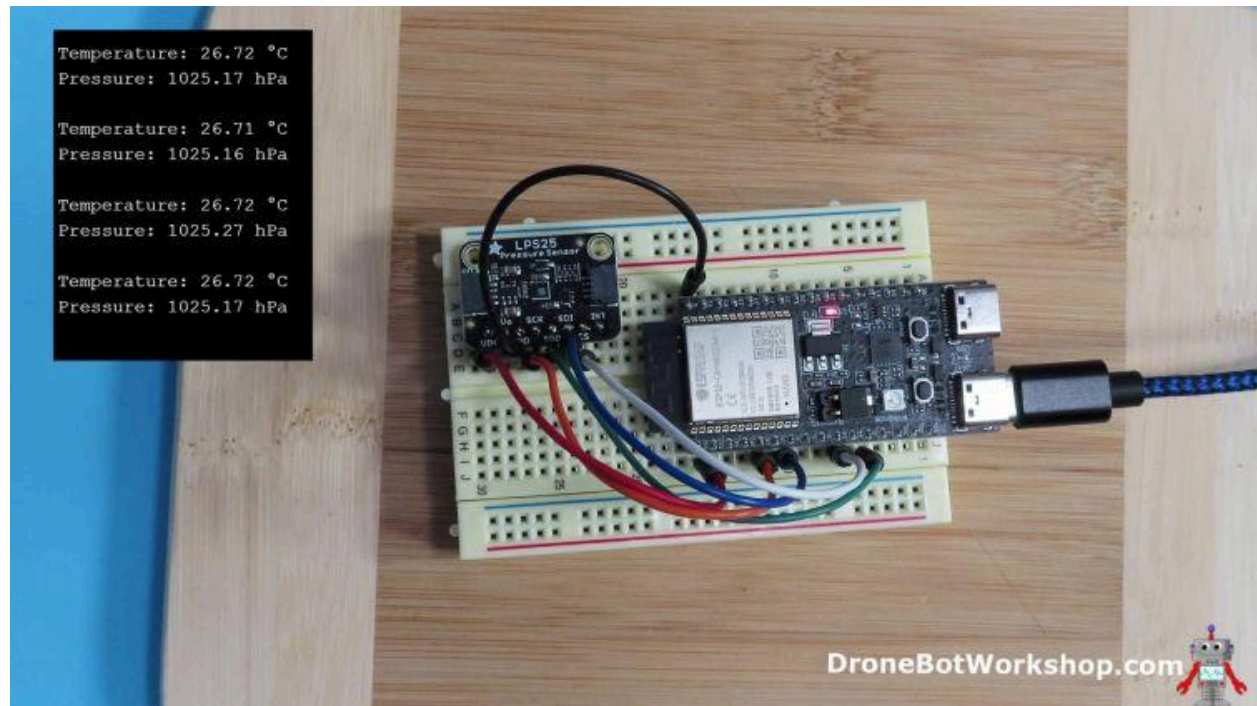
<https://dronebotworkshop.com>

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

```
if (!lps.begin_SPI(LPS25_CS, LPS25_SCK, LPS25_MISO, LPS25_MOSI)) {  
    Serial.println("Could not find LPS25 sensor!");  
    while (1) delay(10);  
}  
}  
  
void loop() {  
    // Sensors Event  
    sensors_event_t temp, pressure;  
    lps.getEvent(&pressure, &temp);  
    Serial.printf("Temperature: %.2f °C\n", temp.temperature);  
    Serial.printf("Pressure: %.2f hPa\n\n", pressure.pressure);  
    delay(1000);  
}
```

Load the code, run it, and observe the serial monitor. You should see the air pressure and the sensor calibration temperature.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>



<https://dronebotworkshop.com>

Comparing Sensors

As all of these sensor modules are I²C devices using Qwiic connectors, it is a pretty simple matter to hook them all together and compare their outputs.

I'm doing precisely that with an Arduino Uno R4 WiFi board.

Comparison Setup

The setup for this comparison is straightforward, thanks to the Qwiic connectors. I simply "daisy-chained" the three sensors off the Qwiic connector on the Uno R4 WiFi board. No hookup diagram required!

The one thing about this arrangement is that the Uno R4 WiFi board, the Qwiic connector is attached to the secondary I2C bus, not the (default) primary one. You need to specify that in the code to make everything work.

Otherwise, I just created a sketch to read the air pressure values from all three sensors. Only the air pressure is measured, even on the two sensors that also read temperature and humidity.

Here is the sketch I used to display the air pressure readings from all three sensors:

```
/*  
  Multi-Sensor Air Pressure  
  multi-air-pressure.ino  
  Uses Adafruit MS8607 Library  
  Uses Adafruit BME280 Library  
  Uses Adafruit LPS2X Library  
  Displays air pressure readings from all sensors  
  Uses Arduino Nano R4, sensors connectd with Qwiic  
  
  DroneBot Workshop 2025  
  https://dronebotworkshop.com  
*/  
  
// Include Required Libraries  
#include <Wire.h>  
#include <Adafruit_MS8607.h>  
#include <Adafruit_BME280.h>  
#include <Adafruit_LPS2X.h>  
  
// Create objects for each pressure sensor  
Adafruit_MS8607 ms;      // MS8607 (PHT)  
Adafruit_BME280 bme;     // BME280 (PHT)  
Adafruit_LPS25 lps;      // LPS25HB (pressure)  
  
void setup() {  
  Serial.begin(115200);  
  while (!Serial) { }
```

```
// Qwiic on UNO R4 WiFi uses the SECONDARY I2C bus:

Wire1.begin();

// MS8607: begin() takes TwoWire*
if (!ms.begin(&Wire1)) {
    Serial.println("MS8607 not found!");
}

// BME280: try 0x76 then 0x77 on Wire1
if (!bme.begin(0x76, &Wire1)) {
    Serial.println("BME280 not found at 0x76 (trying 0x77)...");
    if (!bme.begin(0x77, &Wire1)) {
        Serial.println("BME280 not found!");
    }
}

// LPS25HB: begin_I2C(addr, wire)
if (!lps.begin_I2C(LPS2X_I2CADDR_DEFAULT, &Wire1)) { // default 0x5C
    Serial.println("LPS25HB not found!");
}

Serial.println("Starting pressure comparison...\n");
}

void loop() {

    // --- MS8607 (Unified Sensor API) ---

    sensors_event_t p_ms, t_ms, h_ms;

    float ms_hpa = NAN;
```

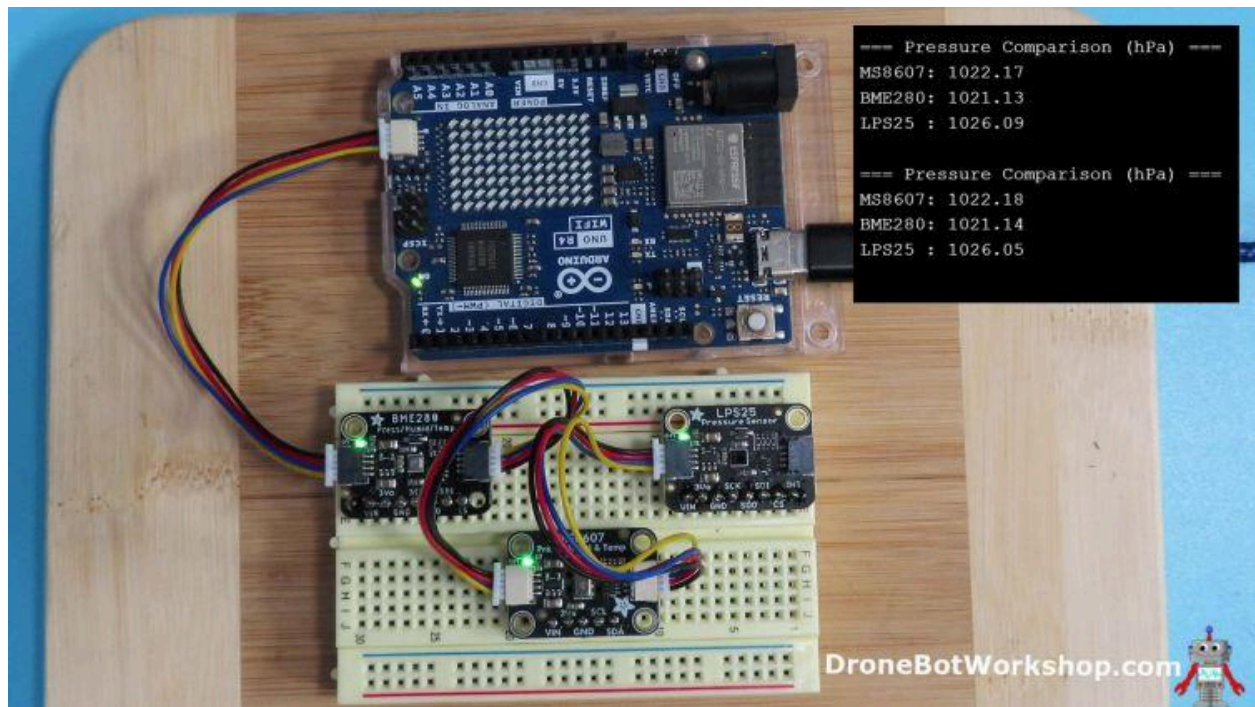


```
if (ms.getEvent(&p_ms, &t_ms, &h_ms)) {  
    ms_hpa = p_ms.pressure; // hPa  
}  
  
// --- BME280 ---  
  
float bme_hpa = NAN;  
  
if (bme.sensorID()) { // crude check that object is alive  
    bme_hpa = bme.readPressure() / 100.0f; // Pa -> hPa  
}  
  
// --- LPS25HB (Unified Sensor API) ---  
  
sensors_event_t p_lps, t_lps;  
  
float lps_hpa = NAN;  
  
if (lps.getEvent(&p_lps, &t_lps)) {  
    lps_hpa = p_lps.pressure; // hPa  
}  
  
Serial.println("=== Pressure Comparison (hPa) ===");  
  
Serial.print("MS8607: ");  
  
if (isnan(ms_hpa)) Serial.println("N/A"); else Serial.println(ms_hpa, 2);  
  
Serial.print("BME280: ");  
  
if (isnan(bme_hpa)) Serial.println("N/A"); else Serial.println(bme_hpa, 2);  
  
Serial.print("LPS25 : ");  
  
if (isnan(lps_hpa)) Serial.println("N/A"); else Serial.println(lps_hpa, 2);  
  
Serial.println();
```

```
    delay(2000);  
}
```

Results

The values were very close, as you can see from this screen grab:



I think that this illustrates that any one of these sensors can be trusted to give accurate air pressure readings.

Conclusion

In this project, we learned how air pressure is measured and explored three excellent sensors from Adafruit:

- **MS8607** — precision triple sensor for professional-grade applications.

For more projects and tutorials visit the DroneBot Workshop - <https://dronebotworkshop.com>

- **BME280** — versatile, easy-to-use environmental sensor ideal for IoT projects.
- **LPS25HB** — small, accurate barometric sensor perfect for drones or altimeters.

We experimented with different microcontrollers and programming environments — **Arduino C++**, **MicroPython**, and **ESP-based web servers** — showing that pressure sensing is accessible no matter which platform you use.

Keep experimenting — try logging the data to an SD card, sending it to **Home Assistant**, or displaying it on a dashboard.

As always, **build something amazing — and I'll see you next time!**