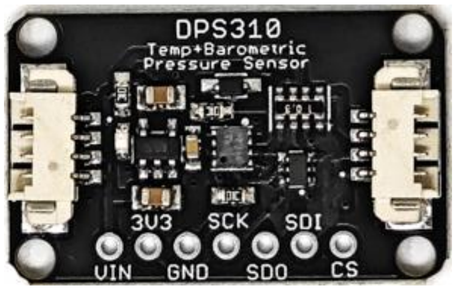
SmartElex DPS310 - Precision Barometric Pressure Altitude Sensor



The DPS310 sensor from Infineon a high precision barometric sensor, perfect for measuring altitude changes with a up to ±0.002 hPa (or ±0.02 m) precision high precision mode and ± 1 hPa absolute accuracy. That means you can know your absolute altitude with 1 meter accuracy when you set the sea-level pressure, and measure changes in altitude with up to 2 cm precision. This makes it a great sensor for use in drones or other altitude-sensitive robots. This sensor would also do well in any environmental sensing kit, you can use it to predict weather system changes

You can use this sensor with either I2C or SPI, so it's easy to integrate into projects. It also has a temperature sensor built in, with ± 0.5°C accuracy. For the lowest noise readings, set it up to take multiple measurements and perform a low-pass filter, that capability is built in! You can use it from 300 to 1200 hPa and in ambient temperature ranges from -40 to 85 °C.

Pinouts

Power Pins

- **Vin** - this is the power pin. Since the sensor chip uses 3 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino, use 5V

- **3Vo** - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like
- **GND** - common ground for power and logic

I2C Logic Pins:

- **SCK** - This is *also* the I2C **clock pin SCL**, connect to your microcontroller's I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- **SDI** - This is *also* the I2C **data pin SDA**, connect to your microcontroller's I2C data line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- **SDO** - This is *also* the I2C address pin **ADR**. Pulling this pin **low to GND** or bridging the solder jumper on the back will change the I2C address from **0x77** to **0x76**

SPI Logic pins:

All pins going into the breakout have level shifting circuitry to make them 3-5V logic level safe. Use whatever logic level is on **Vin!**

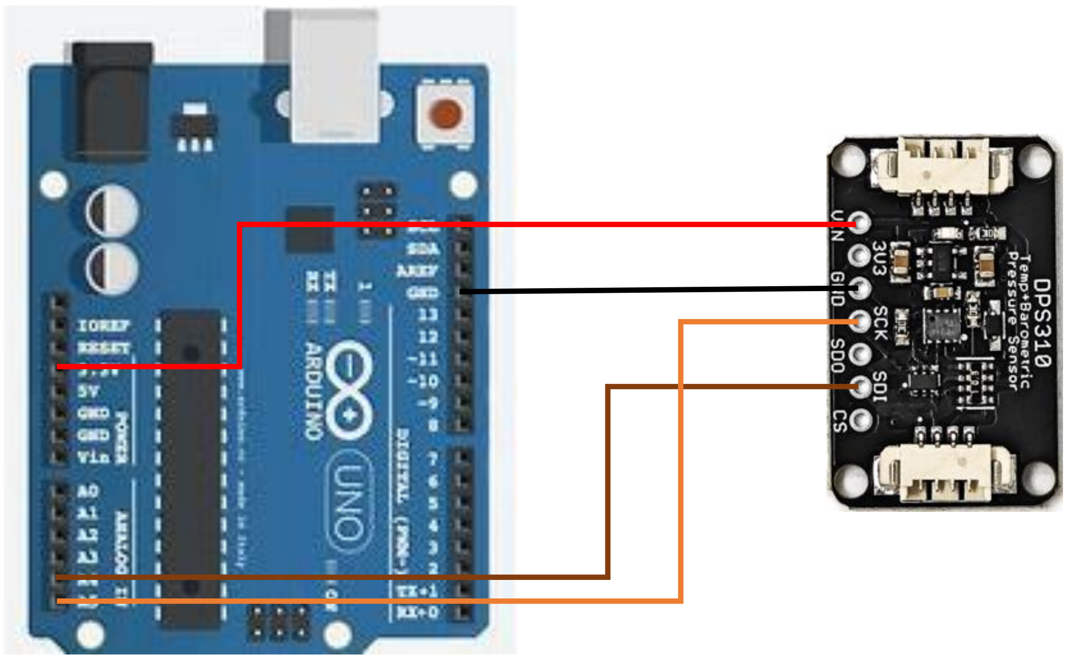
- **SCK** - The **S**PI **C**loc**k** pin, it's an input to the chip
- **SDO** - The **S**erial **D**ata **O**ut / **M**icrocontroller **I**n **S**ensor **O**ut, for data sent from the DPS310 to your processor.
- **SDI** - The **S**erial **D**ata **I**n / **M**icrocontroller **O**ut **S**ensor **I**n pin, for data sent from your processor to the DPS310
- **CS** - The **C**hip **S**elect pin, drop it low to start an SPI transaction. Its an input to the chip

If you want to connect multiple DPS310's to one microcontroller, have them share the **SDI**, **SDO** and **SCK** pins. Then assign each one a unique **CS** pin.

I2C Wiring

Use this wiring if you want to connect via I2C interface

By default, the I2C address is **0x77**. If you add a jumper from D**DO** to **GND** the address will change to **0x76**



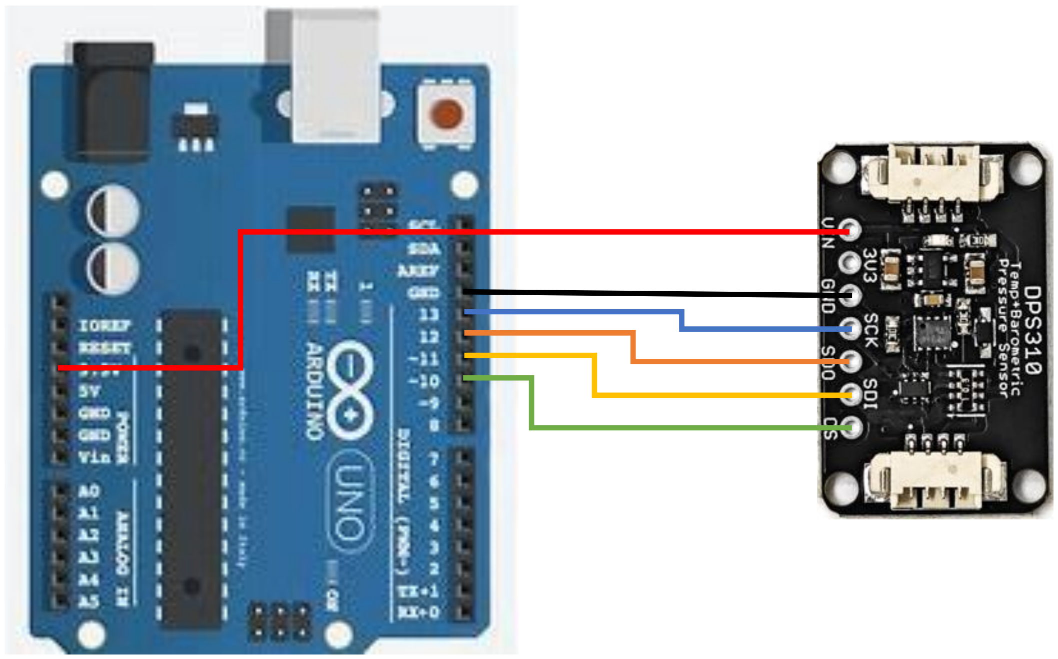
Arduino	DPS310
SCL(A5)	SCK
SDA(A4)	SDI
5v OR 3.3v	VIN
GND	GND

- Connect **board VIN** to **Arduino 5V** if you are running a **5V** board Arduino (Uno, etc.). If your board is **3V,** connect to that instead.
- Connect **board GND** to **Arduino GND**
- Connect **board SCL** to **Arduino SCL**
- Connect **board SDA** to **Arduino SDA**

The final results should resemble the illustration above, showing an Adafruit Metro development board.

SPI Wiring

Since this is a SPI-capable sensor, we can use hardware or 'software' SPI. To make wiring identical on all microcontrollers, we'll begin with 'software' SPI. The following pins should be used:



Arduino	DPS310
D13	SCK
D12	SDO
D11	SDI
D10	CS
5v OR 3.3v	VIN
GND	GND

- Connect **Vin** to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of
- Connect **GND** to common power/data ground
- Connect the **SCK** pin to **Digital #13** but any pin can be used later
- Connect the S**DO** pin to **Digital #12** but any pin can be used later
- Connect the **SDI** pin to **Digital #11** but any pin can be used later
- Connect the **CS** pin **Digital #10** but any pin can be used later

Later on, once we get it working, we can adjust the library to use hardware SPI if you desire, or change the pins to others.

Library Installation

You can install the **Adafruit DPS310 Library** for Arduino using the Library Manager in the Arduino IDE.

Click the **Manage Libraries.** menu item, search for **Adafruit DPS310** and select the **Adafruit DPS310** library

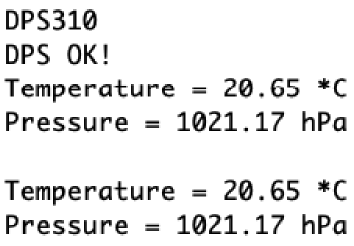
Load Example

Open up **File -> Examples -> Adafruit DPS310 -> dps310_simpletest** and upload to your Arduino wired up to the sensor.

Depending on whether you are using I2C or SPI, change the pin names and comment or uncomment the following lines.

```
if (! dps.begin_I2C()) {             // Can pass in I2C address here
  //if (! dps.begin_SPI(DPS310_CS)) {  // If you want to use SPI
```

Once you upload the code and open the Serial Monitor (**Tools->Serial Monitor**) at **115200** baud, you will see the current configuration printed, followed by the pressure, and temperature measurements. You should see something similar to this:



```
DPS310
DPS OK!
Temperature = 20.65 *C
Pressure = 1021.17 hPa

Temperature = 20.65 *C
Pressure = 1021.17 hPa
```

Carefully pressing on the small port on the top of the sensor will change the pressure and temperature readings.

Example Code:

```
// This example shows how to read temperature/pressure



#include <Adafruit_DPS310.h>



Adafruit_DPS310 dps;



// Can also use SPI!

#define DPS310_CS 10



void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);

  Serial.println("DPS310");
  if (! dps.begin_I2C()) {             // Can pass in I2C address here
  //if (! dps.begin_SPI(DPS310_CS)) {  // If you want to use SPI
    Serial.println("Failed to find DPS");
    while (1) yield();
  }
  Serial.println("DPS OK!");

  dps.configurePressure(DPS310_64HZ, DPS310_64SAMPLES);
  dps.configureTemperature(DPS310_64HZ, DPS310_64SAMPLES);
}
void loop() {
  sensors_event_t temp_event, pressure_event;

  while (!dps.temperatureAvailable() || !dps.pressureAvailable()) {
    return; // wait until there's something to read
  }
  dps.getEvents(&temp_event, &pressure_event);
  Serial.print(F("Temperature = "));
  Serial.print(temp_event.temperature);
  Serial.println(" *C");
  Serial.print(F("Pressure = "));
  Serial.print(pressure_event.pressure);
  Serial.println(" hPa");

  Serial.println();
}
```