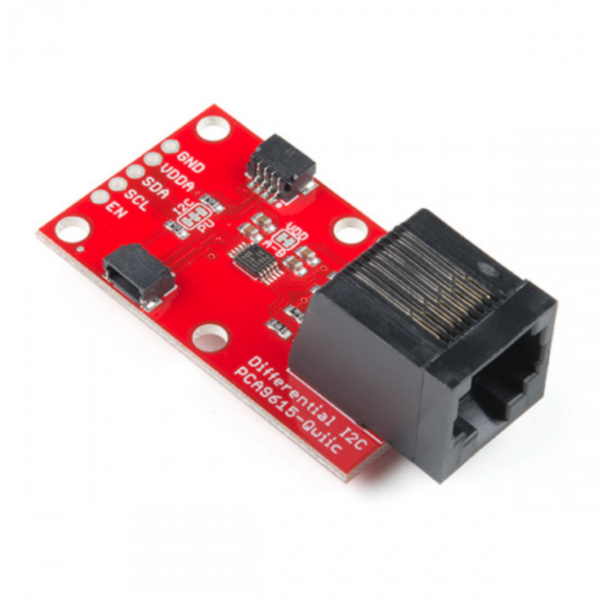
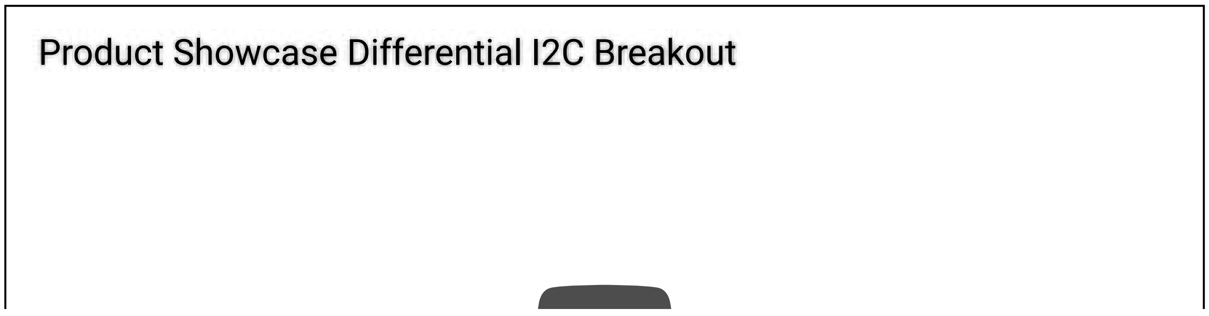
Qwiic Differential I2C Bus Extender (PCA9615) Hookup Guide

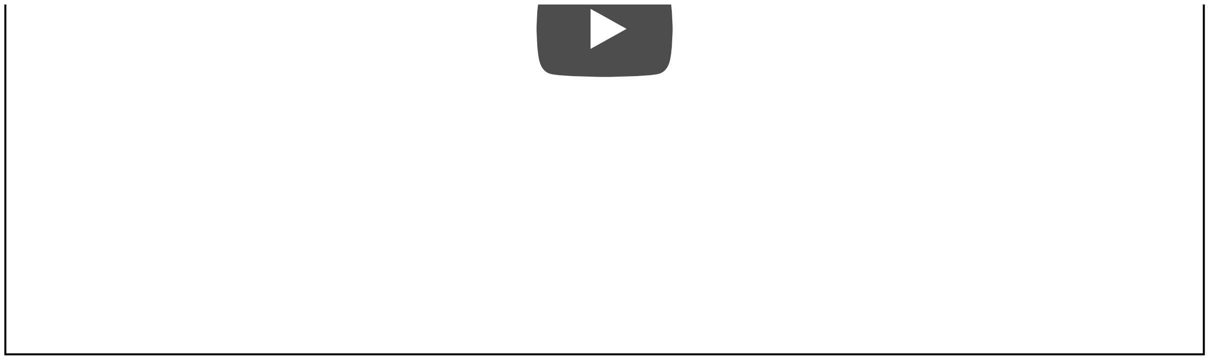
Introduction

The Qwiic Differential I^{2}C Breakout is the fastest and easiest way to extend the range of your I^{2}C communication bus. The breakout uses NXP's PCA9615 IC, which converts the two default I^{2}C signals into four differential signals: two for SCL and two for SDA. Coupled with the ease of SparkFun's Qwiic connection system, the differential I^{2}C breakout board makes it easier to connect it to the rest of your system. The differential signals are sent over an Ethernet cable, which attaches to the breakouts through the on-board RJ-45 connectors. The differential signaling allows the I^{2}C signals to reach distances of up to 100ft. while still maintaining their signal integrity!



SparkFun Differential I2C Breakout - PCA9615 (Qwiic)
◉ BOB-14589





Whether you need to extend the range of an I^{2}C sensor on an autonomous vehicle plagued with noise from motors or want to create a vast sensor network in your home or office, the Qwiic differential I^{2}C breakout is a great solution to extend distance and reduce noise susceptibility.

Required Materials

To follow along with this project tutorial, you will need the following materials. You may not need everything though depending on what you have. Add it to your cart, read through the guide, and adjust the cart as necessary.

- 2x Qwiic Differential I^{2}C breakouts.
- A straight-through Ethernet cable (up to 100ft in length).
- A microcontroller or single board computer capable of I^{2}C.
- An I^{2}C sensor to communicate with on the other end of the I^{2}C bus.
- If using the Qwiic connectors, two Qwiic cables.

Tools

You may need a soldering iron, solder, general soldering accessories, and a hobby knife depending on your setup.

Solder Lead Free - 100-gram Spool
TOL-09325

Weller WLC100 Soldering Station
TOL-14228

Hobby Knife
TOL-09200

Suggested Reading

If you aren’t familiar with our new Qwiic system, we recommend reading here for an overview. We would also recommend taking a look at the following tutorials if you aren’t familiar with them.

Logic Levels

Learn the difference between 3.3V and 5V devices and logic levels.

I2C

An introduction to I2C, one of the main embedded communications protocols in use today.

How to Work w/ Jumper Pads and PCB Traces

Handling PCB jumper pads and traces is an essential skill. In this tutorial, you will learn how to cut a PCB trace and add a solder jumper between pads to reroute connections. You will also learn how to repair a trace with the green wire method if a trace is damaged.

Qwiic Shield for Arduino & Photon Hookup Guide

Get started with our Qwiic ecosystem with the Qwiic shield for Arduino or Photon.

Qwiic HAT for Raspberry Pi Hookup Guide

Get started interfacing your Qwiic enabled boards with your Raspberry Pi. This Qwiic connects the I2C bus

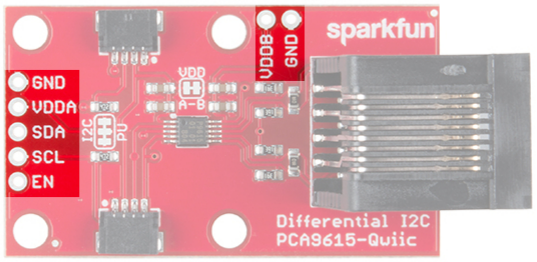
(GND, 3.3V, SDA, and SCL) on your Raspberry Pi to
an array of Qwiic connectors.

Hardware Overview

The simplicity of the Qwiic differential I^{2}C breakout is one of its biggest appeals. Other I^{2}C communication methods require packetizing I^{2}C communication into another protocol, be it RS-485 or 1-Wire. However, the PCA9615 keeps the I^{2}C protocol by utilizing a differential transceiver. In this section, we'll take a closer look at the board to better understand how it works.

Pinout

Below are the plated through hole pins that are broken out on the board. The I^{2}C pins are connected to the two Qwiic connectors on the sides.



PTH Connections

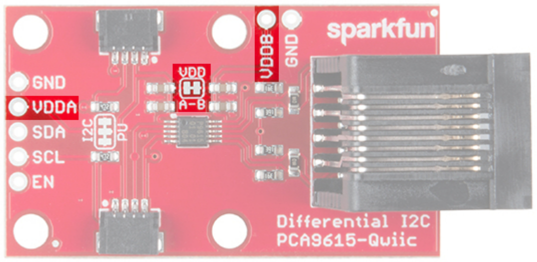
- **GND** - Ground
- **VDDA** - **2.3VDC to 5.5VDC**. I^{2}C-bus side power supply.
- **VDDB** - **3.0VDC to 5.5VDC**. Differential side power supply. If jumper "**VDD A-B**" is not shorted, then VDDB will need to be powered for the differential I^{2}C bus to operate.
- **SDA** - I^{2}C data signal.
- **SCL** - I^{2}C clock signal.
- **EN (optional)** - PCA9615 enable (active high, internally pulled up). This is used to disable the bus buffer, and is useful for fault finding, power-up sequencing, or reconfiguration of a large bus system by isolating sections not needed at all times.

Qwiic Connections

- **GND** - Ground.
- **VDDA** - **2.3VDC to 5.5VDC**. I^{2}C-bus side power supply. If "**VDD A-B**" is not shorted, VDDB will need to powered separately for the differential I^{2}C bus to operate.
- **SDA** - I^{2}C data signal.
- **SCL** - I^{2}C clock signal.

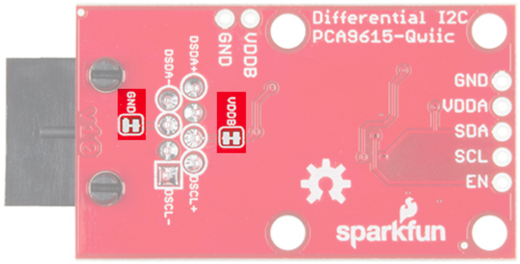
VDDA vs VDDB

To power the board, VDDA must be present and be the same logic voltage as the SDA/SCL lines, while VDDB is used to power the differential I^{2}C bus. By default, the jumper labeled "**VDD A-B**" is closed, which connects the VDDA rail to the VDDB rail. By cutting the jumper you can separate the two rails which would allow for one rail to operate at 3.3V while the other can operate at 5V.



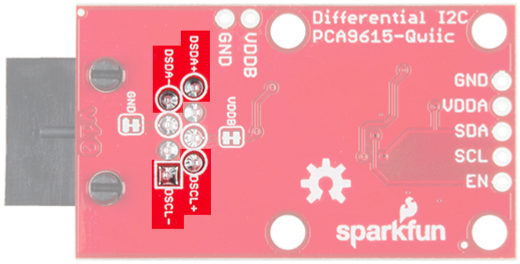
VDDB and Ground Jumpers

If the number of sensors connected on the other side of the extended I^{2}C bus is minimal, you can power them over the Ethernet cable. However, if there are numerous sensors connected, it is advised that both ends be powered separately. To isolate the power supplies at both ends of the Ethernet cable, use a sharp blade to cut the small traces betweens the pads of the jumpers labeled “**VDDB**” and “**GND**”. VDDB will still be present on each board, but the Ethernet cable will not carry any current to power a device at the other end of the cable.



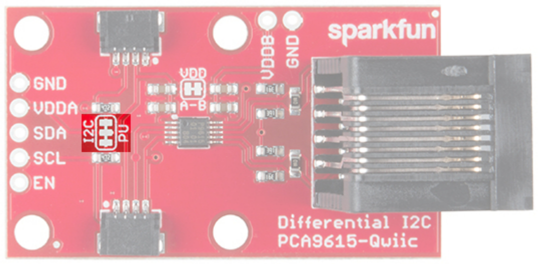
Differential Signals

From the bottom of the board, we can see which pins the RJ-45 connector uses for the differential signaling. The board has been designed to use a standard Ethernet cable. If a custom cable is made, make sure to connect a twisted pair for pins 1 and 2 (**DSCL-**, **DSCL+**) and pins 7 and 8 (**DSDA-**, **DSDA+**).



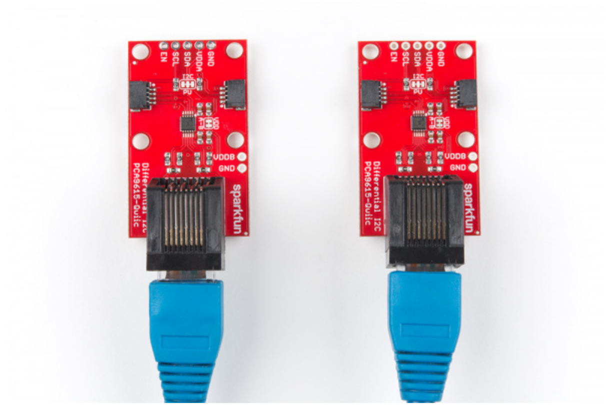
I^{2}C Pull-Up Resistors

As with most SparkFun I^{2}C products, there is a jumper for the pull-up resistors on the I^{2}C bus. If multiple sensors are connected to the bus with the pull-up resistors enabled, the parallel equivalent resistance will create too strong of a pull-up for the bus to operate correctly. As a general rule of thumb, disable all but **one** pair of pull-up resistors if multiple devices are connected to the bus.



Hardware Assembly

Ethernet cables used must be straight-through (i.e. Pin 1 on one side of the cable is connect to pin 1 on the other side. The same for pin 2 and so on.).



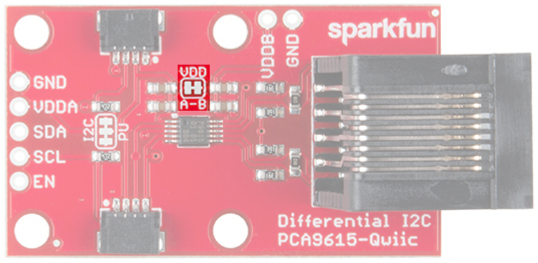
I²C Pull-Up Resistors

Remember, each individual non-differential I²C bus needs at least on set of pull-up resistors enabled. Make sure you keep track of which devices have their I²C pull-ups enabled and which do not.

Power Schemes

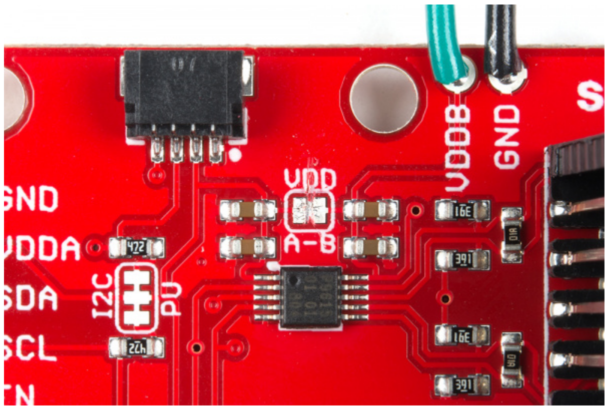
With two power supply rails and quite a few jumpers, it's easy to feel confused about how to power the differential I²C bus extender. In this section, we'll go over the different ways you can power your project.

VDD_A == VDD, Power Whole Bus (Default)



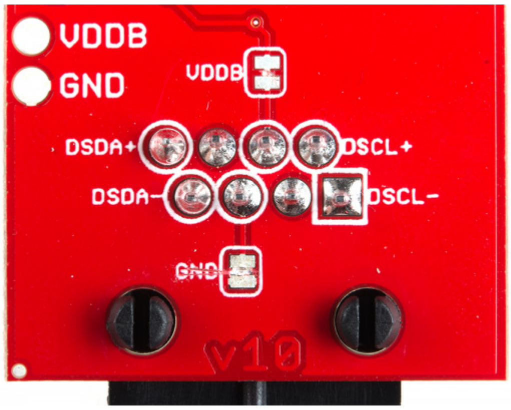
- VDD_A and VDD_B are connected.
- Both are powered at 3.3V.
- Power is connected to a twisted pair of the Ethernet cable and sent to the slave nodes.
- No power is required at the slave nodes.

VDD_A != VDD_B, Power Whole Bus



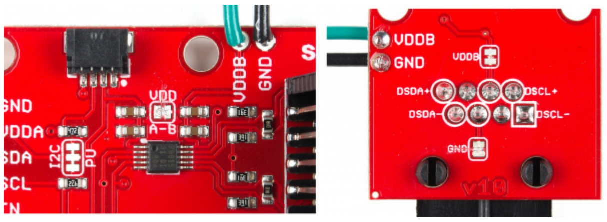
- VDD A-B jumper trace has been cut.
- Separate power supply (3.0-5.5V) is supplied to VDD_B, while VDD_A remains at 3.3V.
- VDD_B voltage is connected to a twisted pair of the Ethernet cable and sent to the slave nodes.
- No power is required at the slave nodes.

VDD_A == VDD_ B, Power Each Node Separately



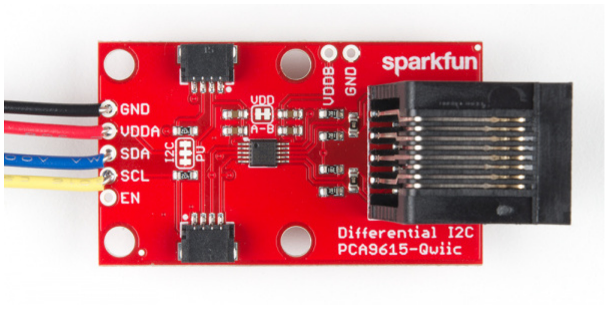
- VDD_A and VDD_B are connected.
- Both are powered at 3.3V.
- Jumpers underneath board (VDDB and GND) are cut and power is **NOT** connected to the Ethernet cable.
- Each slave node is powered individually with 3.3V only. Differential I^{2}C signals are the only connections on the bus.

VDD_A != VDD_ B, Power Each Node Separately



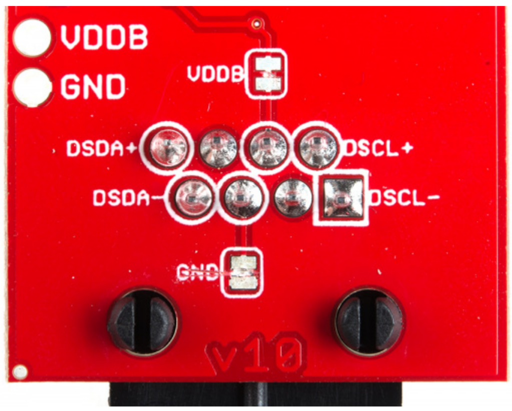
- VDD A-B jumper trace has been cut on the top of the board.
- Separate power supply (3.0-5.5V) is supplied to VDD_B, while VDD_A remains at 3.3V.
- Jumpers underneath board (VDDB and GND) are cut and power is **NOT** connected to the Ethernet cable.
- Each slave node is powered individually with 3.3V only. Differential I^{2}C signals are the only connections on the bus.

VDD_A == VDD_ B, Power Whole Bus (Non-Qwiic Option)



- VDD_A and VDD_B are connected.
- Both are powered at 5V.
- Power is connected to a twisted pair of the Ethernet cable and sent to the slave nodes.
- No power is required at the slave nodes.

VDD_A == VDD_ B, Power Each Node Separately (Non-Qwiic Option)



- VDD_A and VDD_B are connected.
- Both are powered at 5V.
- Jumpers underneath board (VDDB and GND) are cut and power is **NOT** connected to the Ethernet cable.
- Each slave node is powered individually with 5V only. Differential I^{2}C signals are the only connections on the bus.

I2C Bus Extender Example

For this example, we're going to use our Environmental Combo Breakout using the Differential I^{2}C Bus Extender. To follow along in this example, you'll need the following:

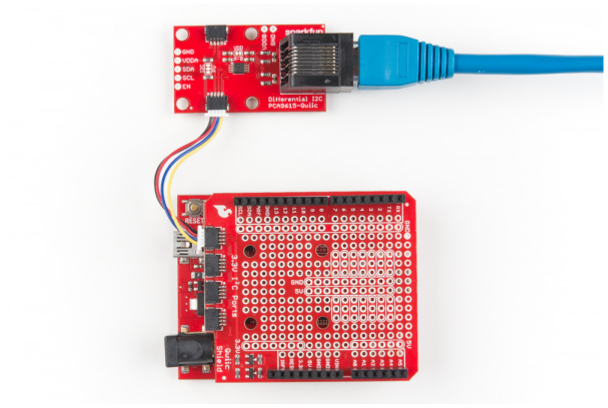
Differential I2C Breakout Hookup Guide Wishlist SparkFun Wish List

	(2) SparkFun Differential I2C Breakout - PCA9615 (Qwiic) BOB-14589
	SparkFun Environmental Combo Breakout - CCS811/BME280 (Qwiic) SEN-14348 The SparkFun CCS811/BME280 Environmental Combo Breakout takes care of all your atmospheric-quality sensin…
	SparkFun Qwiic Shield for Arduino DEV-14352
	SparkFun RedBoard - Programmed with Arduino DEV-13975 At SparkFun we use many Arduinos and we're always looking for the simplest, most stable one. Each board is a bit…
	(2) Qwiic Cable - 50mm PRT-14426

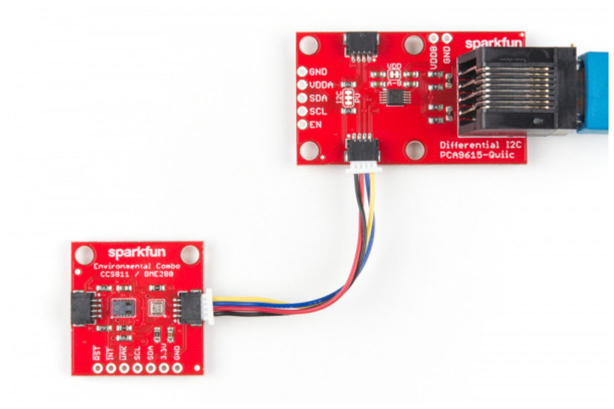
You'll also need a straight-through Ethernet cable (up to 100ft).

Hardware Hookup

You'll first need to solder the headers on to the Qwiic shield and then connect the shield to the Redboard. Once that's done, you can connect one of the differential I^{2}C to your Redboard as shown below using one of the Qwiic cables.



On the other end, all that's needed is to connect the environmental combo sensor to the other differential I^{2}C using another Qwiic cable as shown below.



Arduino Library Installation

Note: This example assumes you are using the latest version of the Arduino IDE on your desktop. If this is your first time using Arduino, please review our tutorial on installing the Arduino IDE. If you have not previously installed an Arduino library, please check out our installation guide.

We’re going to use the code from the environmental combo sensor’s hookup guide.

CCS811/BME280 (Qwiic) Environmental Combo Breakout Hookup Guide

SEPTEMBER 8, 2017

Sense various environmental conditions such as temperature, humidity, barometric pressure, eCO2 and tVOCs with the CCS811 and BME280 combo breakout board.

Since we are using the environmental combo sensor, the BME280 and CCS811 libraries need to be installed. SparkFun has written libraries to control both the CCS811 and the BME280. You can obtain these libraries through the Arduino Library Manager. Search for **SparkFun BME280** and **SparkFun CCS811** to install the latest version. If you prefer downloading the libraries, you can grab them here to manually install:

DOWNLOAD THE SPARKFUN BME280 LIBRARY (ZIP)

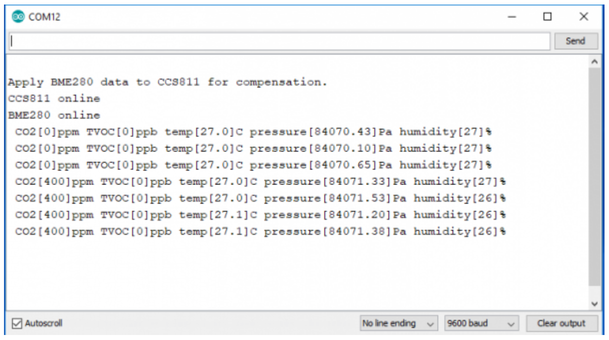
DOWNLOAD THE SPARKFUN CCS811 LIBRARY (ZIP)

Example Code

With the sensor connected via the I^{2}C bus extender, we will use example 2 of the environmental combo breakout. Assuming that you have installed the library for both sensors, copy the code below and paste it into your Arduino IDE and upload the sketch to your board.

```
#include <SparkFunBME280.h>  //https://github.com/sparkfun/SparkFun_BME280_Arduino_Library/
#include <SparkFunCCS811.h>  //https://github.com/sparkfun/SparkFun_CCS811_Arduino_Library/

#define CCS811_ADDR 0x5B //Default I2C Address
//#define CCS811_ADDR 0x5A //Alternate I2C Address

//Global sensor objects
CCS811 myCCS811(CCS811_ADDR);
BME280 myBME280;

void printData()
{
  Serial.print(" CO2[");
  Serial.print(myCCS811.getCO2());
  Serial.print("]ppm");

  Serial.print(" TVOC[");
  Serial.print(myCCS811.getTVOC());
  Serial.print("]ppb");

  Serial.print(" temp[");
  Serial.print(myBME280.readTempC(), 1);
  Serial.print("]C");

  //Serial.print(" temp[");
  //Serial.print(myBME280.readTempF(), 1);
  //Serial.print("]F");

  Serial.print(" pressure[");
  Serial.print(myBME280.readFloatPressure(), 2);
  Serial.print("]Pa");

  //Serial.print(" pressure[");
  //Serial.print((myBME280.readFloatPressure() * 0.0002953), 2);
  //Serial.print("]InHg");

  //Serial.print("altitude[");
  //Serial.print(myBME280.readFloatAltitudeMeters(), 2);
  //Serial.print("]m");

  //Serial.print("altitude[");
  //Serial.print(myBME280.readFloatAltitudeFeet(), 2);
  //Serial.print("]ft");

  Serial.print(" humidity[");
  Serial.print(myBME280.readFloatHumidity(), 0);
  Serial.print("]%");

  Serial.println();
}

void printDriverError( CCS811Core::status errorCode )
{
  switch ( errorCode )
  {
    case CCS811Core::SENSOR_SUCCESS:
      Serial.print("SUCCESS");
      break;
    case CCS811Core::SENSOR_ID_ERROR:
      Serial.print("ID_ERROR");
      break;
    case CCS811Core::SENSOR_I2C_ERROR:
      Serial.print("I2C_ERROR");
      break;
    case CCS811Core::SENSOR_INTERNAL_ERROR:
      Serial.print("INTERNAL_ERROR");
      break;
    case CCS811Core::SENSOR_GENERIC_ERROR:
      Serial.print("GENERIC_ERROR");
      break;
    default:
      Serial.print("Unspecified error.");
  }
}

void setup()
{
  Serial.begin(9600);
  Serial.println();
  Serial.println("Apply BME280 data to CCS811 for compensation.");

  //This begins the CCS811 sensor and prints error status of .begin()
  CCS811Core::status returnCode = myCCS811.begin();
  if (returnCode != CCS811Core::SENSOR_SUCCESS)
  {
    Serial.println("Problem with CCS811");
    printDriverError(returnCode);
  }
  else
  {
    Serial.println("CCS811 online");
  }

  //Initialize BME280
  //For I2C, enable the following and disable the SPI section
  myBME280.settings.commInterface = I2C_MODE;
  myBME280.settings.I2CAddress = 0x77;
  myBME280.settings.runMode = 3; //Normal mode
  myBME280.settings.tStandby = 0;
  myBME280.settings.filter = 4;
  myBME280.settings.tempOverSample = 5;
  myBME280.settings.pressOverSample = 5;
  myBME280.settings.humidOverSample = 5;

  //Calling .begin() causes the settings to be loaded
  delay(10);  //Make sure sensor had enough time to turn on. BME280 requires 2ms to start up.
  byte id = myBME280.begin(); //Returns ID of 0x60 if successful
  if (id != 0x60)
  {
    Serial.println("Problem with BME280");
  }
  else
  {
    Serial.println("BME280 online");
  }
}

void loop()
{
  if (myCCS811.dataAvailable()) //Check to see if CCS811 has new data (it's the slowest sensor)
  {
    myCCS811.readAlgorithmResults(); //Read latest from CCS811 and update tVOC and CO2 variables
    //getWeather(); //Get latest humidity/pressure/temp data from BME280
    printData(); //Pretty print all the data
  }
  else if (myCCS811.checkForStatusError()) //Check to see if CCS811 has thrown an error
  {
    Serial.println(myCCS811.getErrorRegister()); //Prints whatever CSS811 error flags are detect
ed
  }

  delay(2000); //Wait for next reading
}
```

Once uploaded, open your favorite serial monitor to the port assigned to the Arduino. You should see something like similar to the output below:



```
Apply BME280 data to CCS811 for compensation.
CCS811 online
BME280 online
 CO2[0]ppm TVOC[0]ppb temp[27.0]C pressure[84070.43]Pa humidity[27]%
 CO2[0]ppm TVOC[0]ppb temp[27.0]C pressure[84070.10]Pa humidity[27]%
 CO2[0]ppm TVOC[0]ppb temp[27.0]C pressure[84070.65]Pa humidity[27]%
 CO2[400]ppm TVOC[0]ppb temp[27.0]C pressure[84071.33]Pa humidity[27]%
 CO2[400]ppm TVOC[0]ppb temp[27.0]C pressure[84071.53]Pa humidity[26]%
 CO2[400]ppm TVOC[0]ppb temp[27.1]C pressure[84071.20]Pa humidity[26]%
 CO2[400]ppm TVOC[0]ppb temp[27.1]C pressure[84071.38]Pa humidity[26]%
```

Resources and Going Further

Now that you've successfully got your differential I^{2}C breakout up and running, it's time to incorporate it into your own project!

For more on the differential I^{2}C breakout, check out the links below:

- Schematic (PDF)
- Eagle Files (ZIP)
- PCA9615 Datasheet (PDF)
- Qwiic Landing Page
- GitHub Repo
- SFE Product Showcase: Differential I2C Breakout

For more information related to I^{2}C over long distances, check out the resources below:

- An Introduction to I^{2}C Over Long Wires
- An Introduction to Differential I^{2}C
- Differential I^{2}C Application Note - AN10710 (PDF)
- One mile long I^{2}C communication (PDF)

Need some inspiration for your next project? Check out some of these related tutorials:

Qwiic Shield for Arduino & Photon Hookup Guide

Get started with our Qwiic ecosystem with the Qwiic shield for Arduino or Photon.

Qwiic Adapter Hookup Guide

Get started with your Qwiic adapter board. This adapter breaks out the I2C pins from the Qwiic connectors to pins that you can easily solder with your favorite I2C enabled device.

SparkFun GPS Breakout - XA1110 (Qwiic) Hookup Guide

Figure out where in the world you are with the Qwiic SparkFun GPS Breakout - XA1110.

Qwiic Accelerometer (MMA8452Q) Hookup Guide

Freescale’s MMA8452Q is a smart, low-power, three-axis, capacitive micro-machined accelerometer with 12-bits of resolution. It’s perfect for any project that needs to sense orientation or motion. We’ve taken that accelerometer and stuck it on a Qwiic-Enabled breakout board to make interfacing with the tiny, QFN package a bit easier.