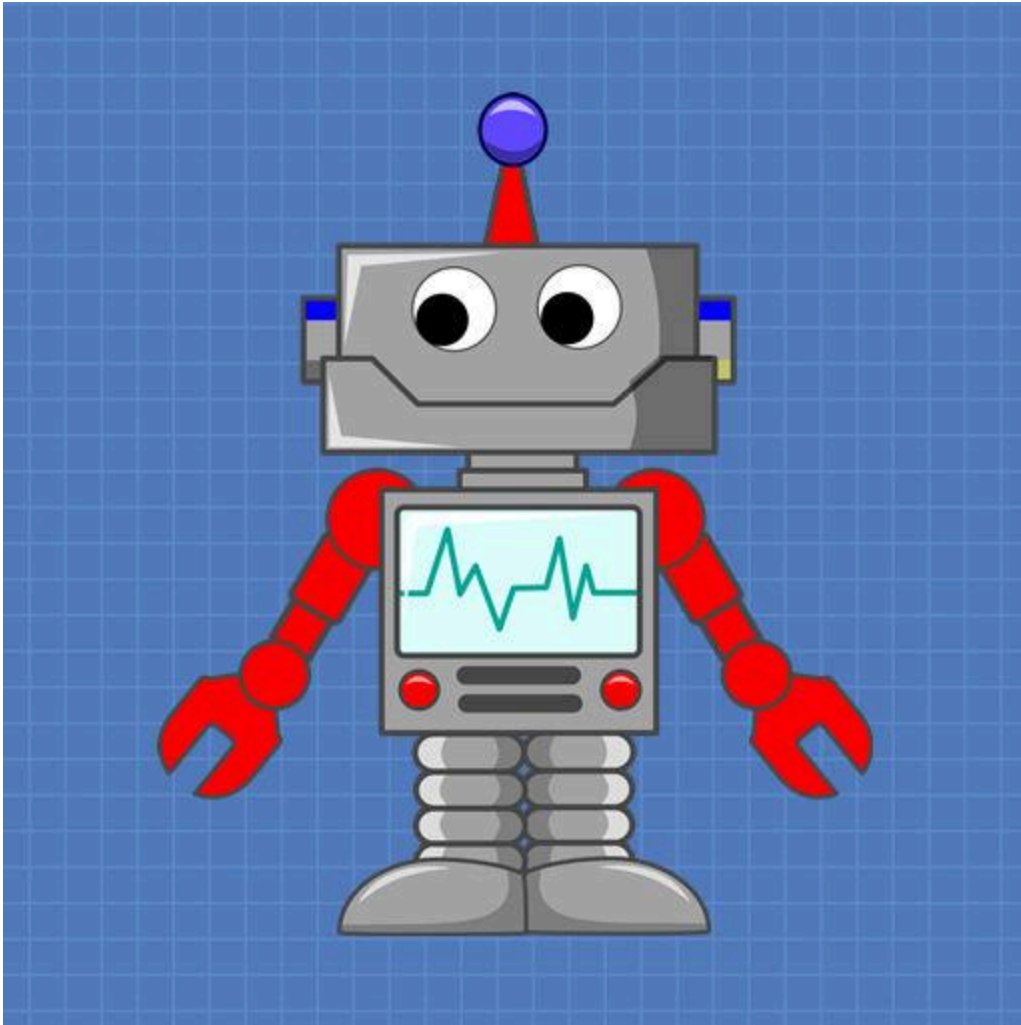


CO² Sensors for Microcontrollers

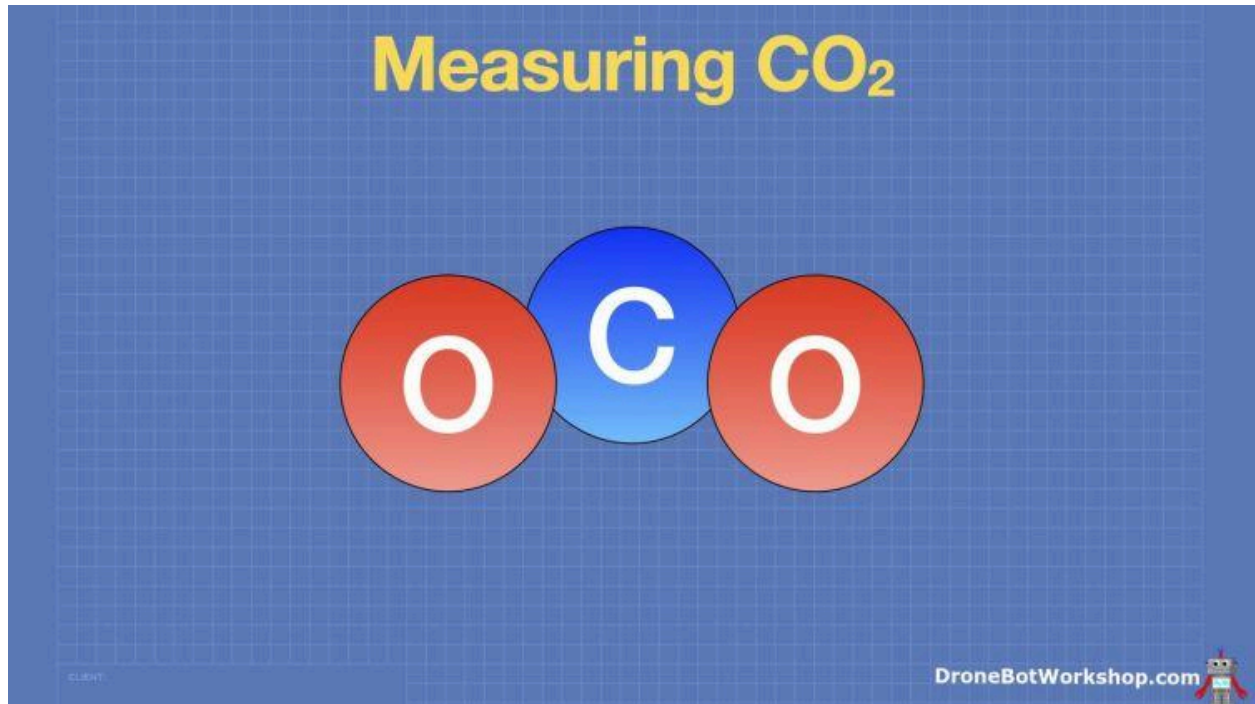


DroneBot Workshop Tutorial

<https://dronebotworkshop.com>

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

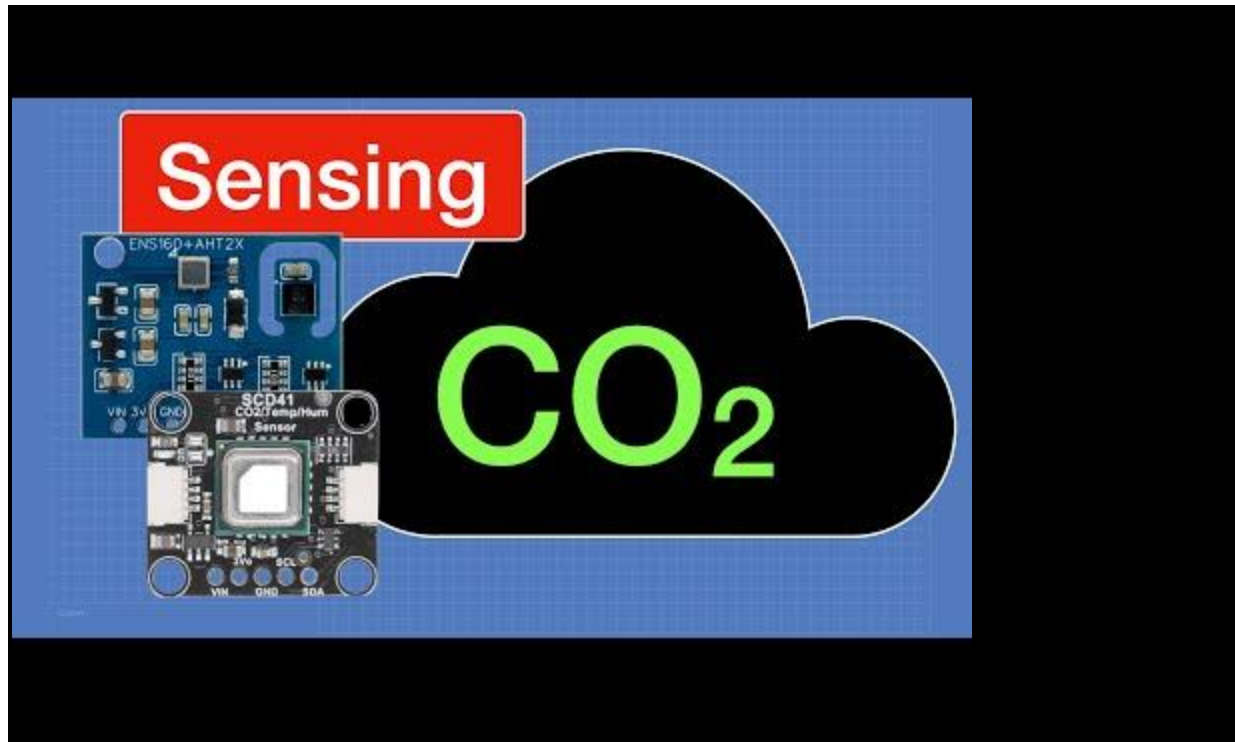
Today, we are going to check out two sensors that can measure the amount of Carbon Dioxide, or CO₂, in the air you are breathing. Actually, only one of them really measures Carbon Dioxide directly; the other measures “eCO₂” or “Estimated Carbon Dioxide”.



Introduction

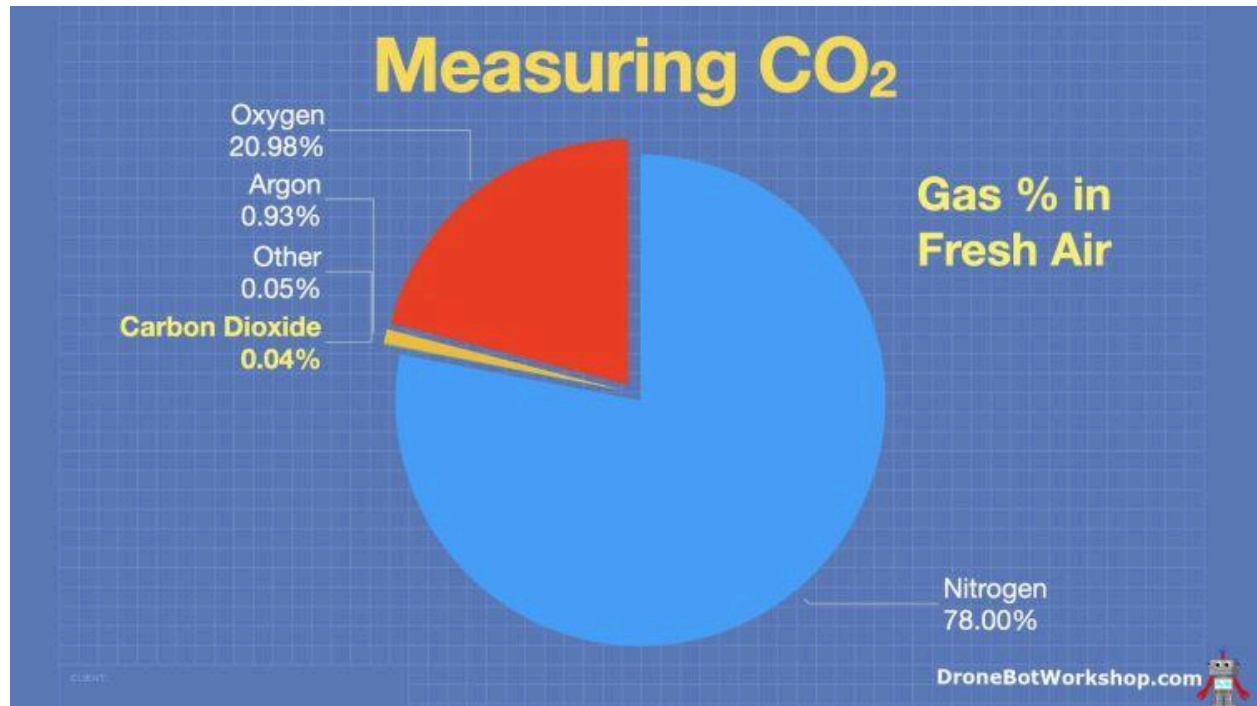
As concerns about indoor air quality and climate change grow, the ability to monitor carbon dioxide (CO₂) levels has become increasingly important. Whether you’re building a smart ventilation system, a classroom air monitor, or a personal environmental tracker, understanding how CO₂ behaves—and how to measure it accurately—is essential. This article explores two popular sensors: the **ENS160**, which estimates equivalent CO₂ (eCO₂), and the **SCD41**, a true CO₂ sensor. We’ll walk through their differences, how to connect them to a Seeeduino XIAO ESP32-S3, and how to use both in tandem for more robust air quality monitoring.

<https://dronebotworkshop.com>



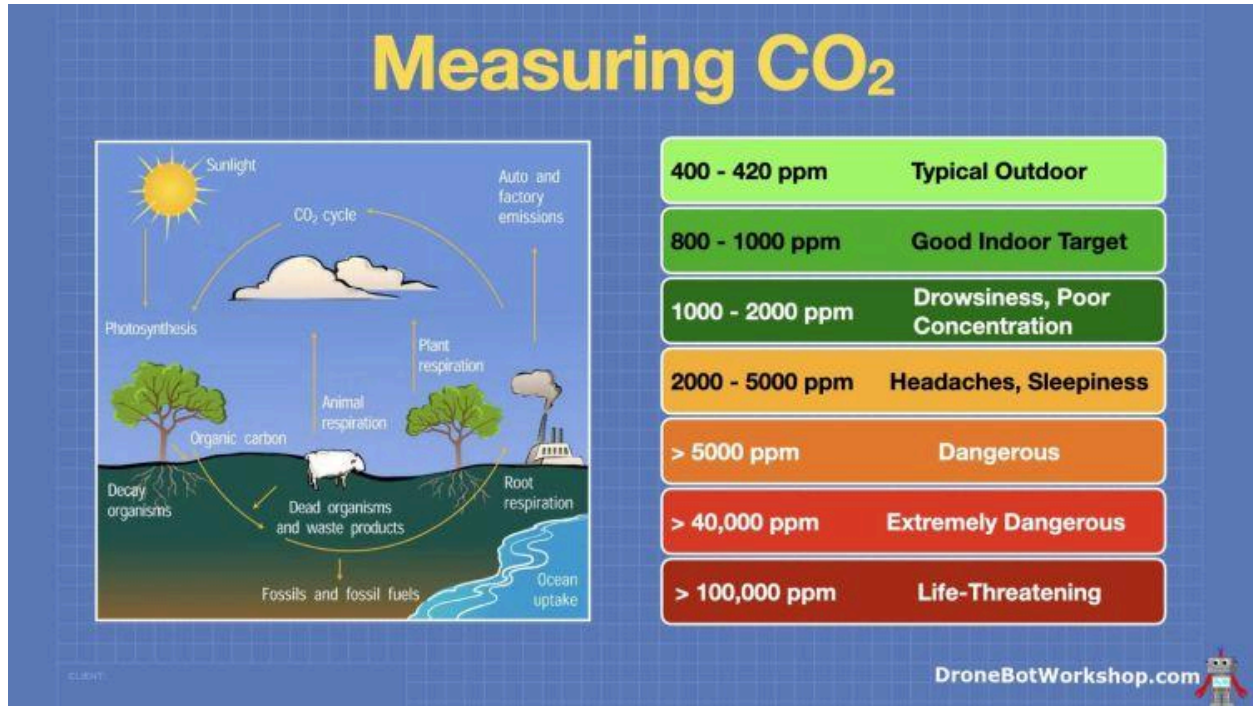
Carbon Dioxide

Carbon dioxide is a naturally occurring gas in Earth's atmosphere, playing a vital role in the **carbon cycle**—the process by which carbon moves through the biosphere, atmosphere, oceans, and geosphere. Plants absorb CO₂ during photosynthesis, while animals and humans release it through respiration. Volcanoes, forest fires, and decomposition are natural sources of CO₂, but **human activities**—especially fossil fuel combustion, deforestation, and industrial processes—have dramatically increased atmospheric CO₂ levels.



Elevated CO₂ indoors can lead to **poor air quality**, fatigue, headaches, and reduced cognitive performance. Outdoors, excess CO₂ contributes to the **greenhouse effect**, trapping heat and accelerating climate change.

Indoors, the primary source of pollution is simply **people breathing**. Combustion devices (such as gas stoves, heaters, and fireplaces), candles/incense, and idling vehicles in attached garages also contribute.



Outdoor air contains ~**420 ppm** of CO₂. Well-ventilated indoor spaces aim for < **800–1000 ppm**. As levels rise:

- **1000–2000 ppm:** drowsiness, reduced concentration.
- **2000–5000 ppm:** headaches, poorer cognition—ventilate now.
- **≥ 5000 ppm (8-hr exposure limit in many standards):** avoid sustained exposure.

Very high concentrations (several percent) can be dangerous.

Measuring CO₂ vs. eCO₂

There are two primary ways to measure carbon dioxide:

- **True CO₂ Sensors** (like the SCD41) utilize **NDIR (Non-Dispersive Infrared)** technology to measure CO₂ concentration directly. These sensors are highly accurate and ideal for scientific or regulatory applications.
- **eCO₂ Sensors** (like the ENS160) estimate CO₂ levels based on **volatile organic compounds (VOCs)** detected in the air. While not a direct measurement, eCO₂

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

readings are helpful in tracking trends in indoor air quality and are often more affordable and compact.

ENS160 (with AHT21)

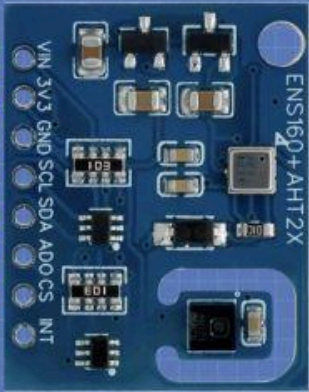


The **ScioSense ENS160** is a digital MOX air-quality sensor that reports **TVOC**, **eCO₂**, and a **5-step AQI**. The breakout board we will be using also includes an **AHT21** for temperature and humidity sensing, which improves ENS160's internal compensation.

These modules are very inexpensive and are ideal for use in automated ventilation systems.

ENS160 Pinout

The sensor communicates using I²C, and has the following pinouts:



ENS160

VIN - Power Input (3.3 or 5 V) *

3V3 - 3.3 Volt Output (Low Current)

GND - Ground

SCL - I²C Clock

SDA - I²C Data

AD0 - I²C Address Select **

CS - I²C / SPI

INT - Interrupt Out

* Some models have pull-ups connected to VIN

** Open or HIGH = 0x53, LOW (Ground) = 0x52

DroneBotWorkshop.com

Note the use of the AD0 connection; it can set the I²C address as follows:

- **OPEN** – Address 0x53 (default)
- **GROUNDED** – Address 0x52

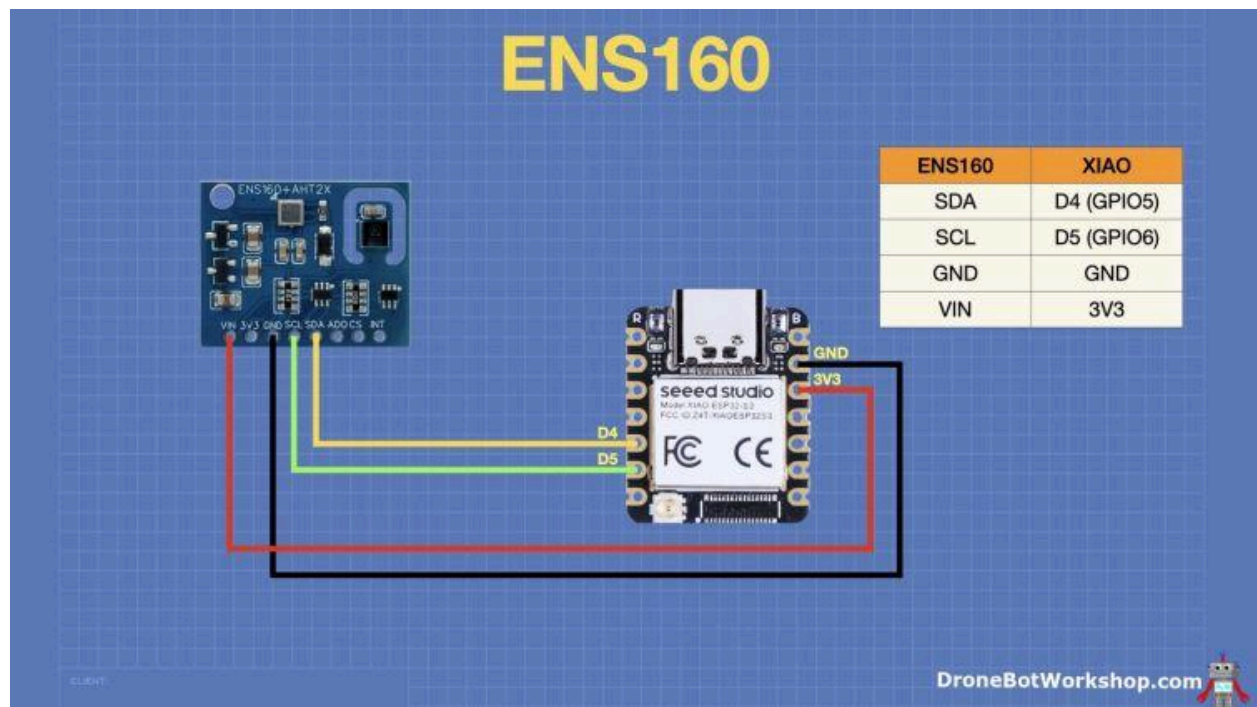
The sensor should be powered by 3.3 volts on the VIN pin. **The 3V3 pin is an OUTPUT**, and it only provides a reference voltage – don't use this to power anything!

The included AHT21 temperature and humidity sensor has an I²C address of 0x38. Bear in mind that its proximity to the ENS160, which uses an internal heating element, causes the temperature to read slightly higher.

ENS160 Hookup

We will conduct our experiments using the ENS160 with an ESP32. I used a Seeedstudio XIAO ESP32-S3 board, but any ESP32 (or just about any 3.3-volt microcontroller) will work.

Here is how we will hook everything up:



ENS160 Code

We will be using libraries from both Sparkfun and Adafruit to program our ESP32. This will allow us to retrieve data from both the ENS160 and the AHT21 temperature and humidity sensors.

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

Here is some code that will display the parameters from all the sensors, output in a convenient table format:

```
/*  
  ENS160 eCO2 Sensor with AHT21 T/H Sensor  
  ens160.ino  
  Reads TVOC/eCO2/AQI and compensates with AHT21 T/RH  
  Requires SparkFun_ENS160 Library  
  Requires Adafruit_AHTX0 Library  
  Uses Seeeduino XIAO ESP32-S3, adjust pins for other ESP32  
  
  DroneBot Workshop 2025  
  https://dronebotworkshop.com  
*/  
  
// Include required libraries  
#include <Wire.h>  
#include "SparkFun_ENS160.h"  
#include <Adafruit_AHTX0.h>  
  
// I2C pins (adjust for different ESP32)  
#define SDA_PIN 5 // XIAO D4  
#define SCL_PIN 6 // XIAO D5  
  
// Objects for ENS160 & AHT21  
SparkFun_ENS160 ens;  
Adafruit_AHTX0 aht;  
  
// Output formatting  
const uint8_t HEADER_EVERY = 12; // reprint table header every N lines  
uint8_t lineCount = 0;
```

```
// AQI Rating Function

const char* aqiText(uint8_t aqi) {

    switch (aqi) {

        case 1: return "Excellent";

        case 2: return "Good";

        case 3: return "Moderate";

        case 4: return "Poor";

        case 5: return "Unhealthy";

        default: return "?";

    }

}

// Ventilation Hint Function

const char* ventHint(uint16_t eco2_ppm) {

    if (eco2_ppm >= 1500) return "Ventilate now";

    if (eco2_ppm >= 1000) return "Consider ventilation";

    return "OK";

}

// Header Print Function

void printHeader() {

    Serial.println();

    Serial.println(F("+-----+-----+-----+-----+-----+-----+-----+-----+
-----+"));

    Serial.println(F("| Time (s) | Temp C | RH % | TVOC ppb | eCO2 ppm | AQI | AQI
(text) |"));

}
```

```
Serial.println(F("+-----+-----+-----+-----+-----+-----+-----+-----+
-----+"));
}

void setup() {
    // Start Serial Monitor
    Serial.begin(115200);
    delay(200);

    // Start I2C
    Wire.begin(SDA_PIN, SCL_PIN);
    Wire.setClock(400000);

    // Start ENS160 measurements
    if (!ens.begin()) {
        Serial.println("ENS160 not found (0x53/0x52). Check wiring/ADD pin.");
        while (1) delay(100);
    }
    ens.setOperatingMode(SFE_ENS160_STANDARD);

    // Start AHT21
    if (!aht.begin()) {
        Serial.println("AHT21 not found (0x38). Check wiring.");
        while (1) delay(100);
    }

    Serial.println(F("ENS160 + AHT21 ready. ENS160 needs a short warm-up for stable
readings."));
    printHeader();
}
```

```
}

void loop() {

    // Read AHT21 T/RH for display (and to show alongside ENS160 data)

    sensors_event_t humidity, temp;
    aht.getEvent(&humidity, &temp);

    float tC = temp.temperature;
    float rh = humidity.relative_humidity;

    // When ENS160 data is ready (about once per second), print a row
    if (ens.checkDataStatus()) {

        uint8_t aqi = ens.getAQI();    // 1..5
        uint16_t tvoc = ens.getTVOC(); // ppb
        uint16_t eco2 = ens.getECO2(); // ppm (equivalent; VOC-derived)

        if (lineCount % HEADER_EVERY == 0) printHeader();

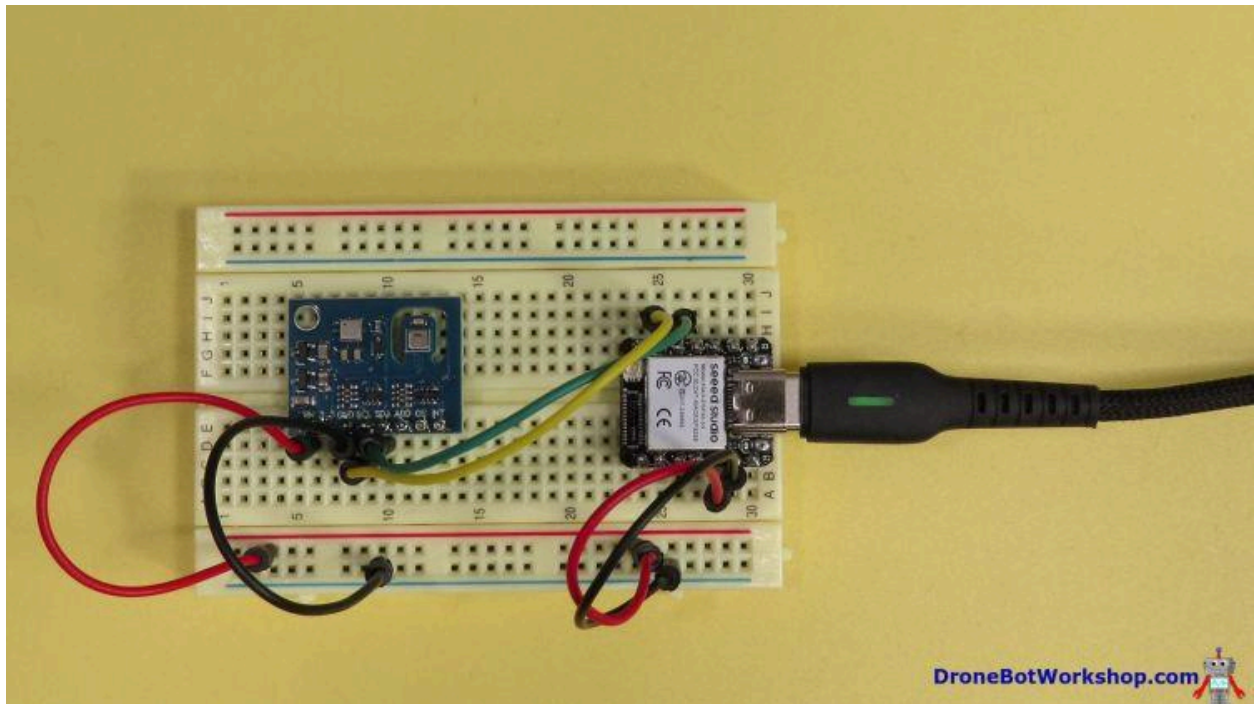
        // Row
        Serial.printf("| %8lu | %7.2f | %6.1f | %9u | %9u | %u | %-12s |\n",
                      millis() / 1000UL, tC, rh, tvoc, eco2, aqi, aqiText(aqi));

        // One-line hint (eCO2 is an estimate)
        Serial.printf("  Note: AQI=%u (%s), eCO2=%u ppm → %s (eCO2 is estimated)\n\n",
                      aqi, aqiText(aqi), eco2, ventHint(eco2));

        lineCount++;
    }
}
```


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

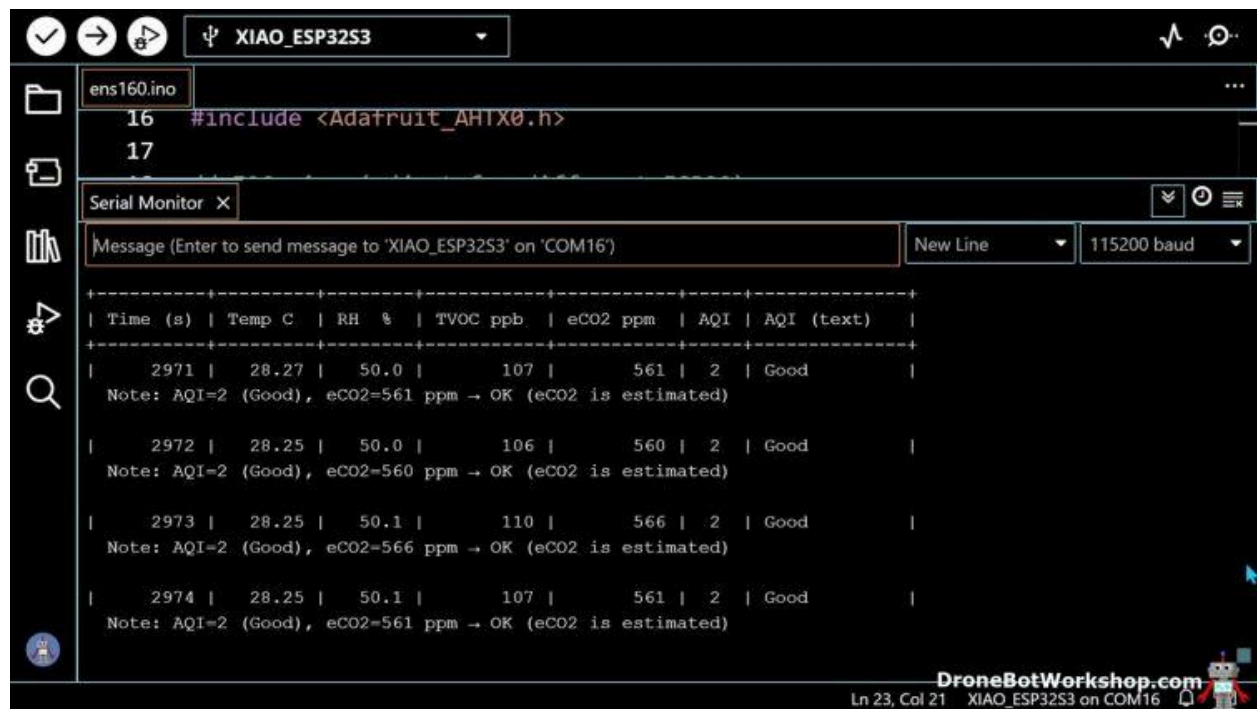
```
delay(200);  
}
```



Load the code onto the XIAO and let it run for at least three minutes to give the ENS160 time to stabilize. You should see the results on your Serial Monitor.

<https://dronebotworkshop.com>

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



```
ens160.ino
16 #include <Adafruit_AHTX0.h>
17

Serial Monitor X
Message (Enter to send message to 'XIAO_ESP32S3' on 'COM16') New Line 115200 baud

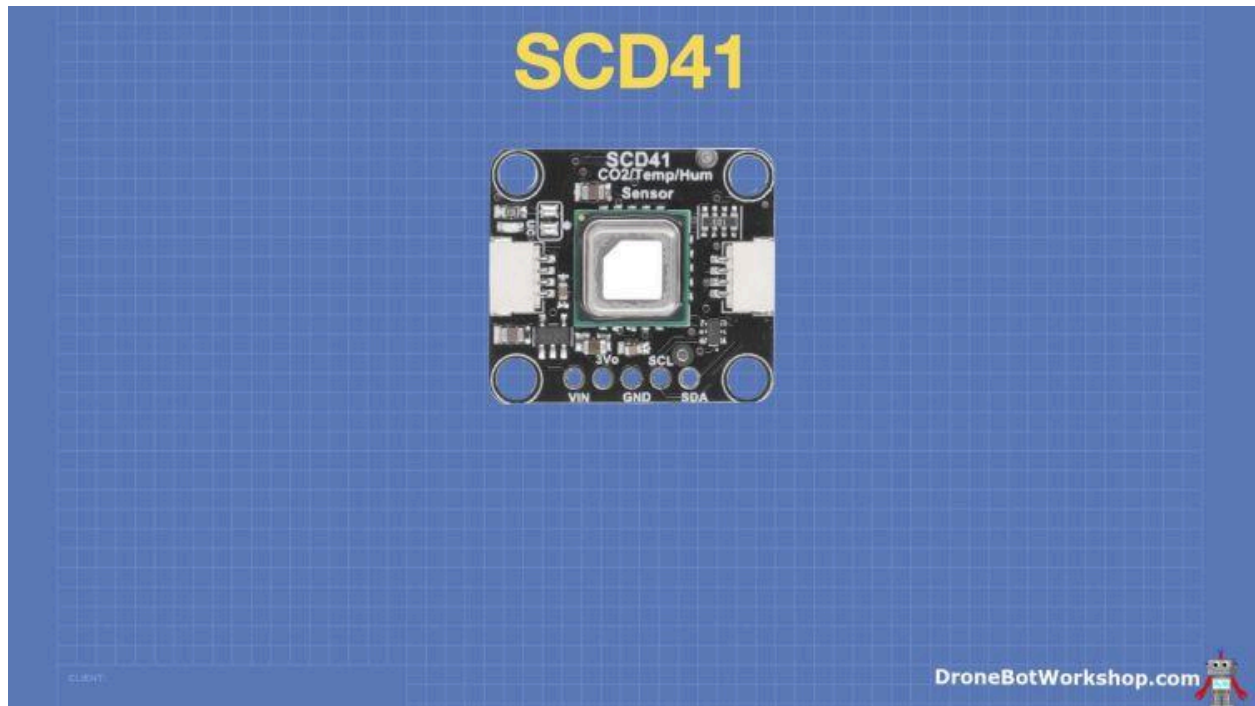
+-----+-----+-----+-----+-----+-----+-----+-----+
| Time (s) | Temp C | RH % | TVOC ppb | eCO2 ppm | AQI | AQI (text) |
+-----+-----+-----+-----+-----+-----+-----+
| 2971 | 28.27 | 50.0 | 107 | 561 | 2 | Good |
| Note: AQI=2 (Good), eCO2=561 ppm → OK (eCO2 is estimated) |
+-----+-----+-----+-----+-----+-----+-----+
| 2972 | 28.25 | 50.0 | 106 | 560 | 2 | Good |
| Note: AQI=2 (Good), eCO2=560 ppm → OK (eCO2 is estimated) |
+-----+-----+-----+-----+-----+-----+-----+
| 2973 | 28.25 | 50.1 | 110 | 566 | 2 | Good |
| Note: AQI=2 (Good), eCO2=566 ppm → OK (eCO2 is estimated) |
+-----+-----+-----+-----+-----+-----+-----+
| 2974 | 28.25 | 50.1 | 107 | 561 | 2 | Good |
| Note: AQI=2 (Good), eCO2=561 ppm → OK (eCO2 is estimated) |
+-----+-----+-----+-----+-----+-----+-----+

Ln 23, Col 21 XIAO_ESP32S3 on COM16
```

While this is just an eCO₂ sensor, it remains a valuable tool, as it can also measure other parameters, such as air quality and TVOC.

<https://dronebotworkshop.com>

SCD41 CO₂ Sensor

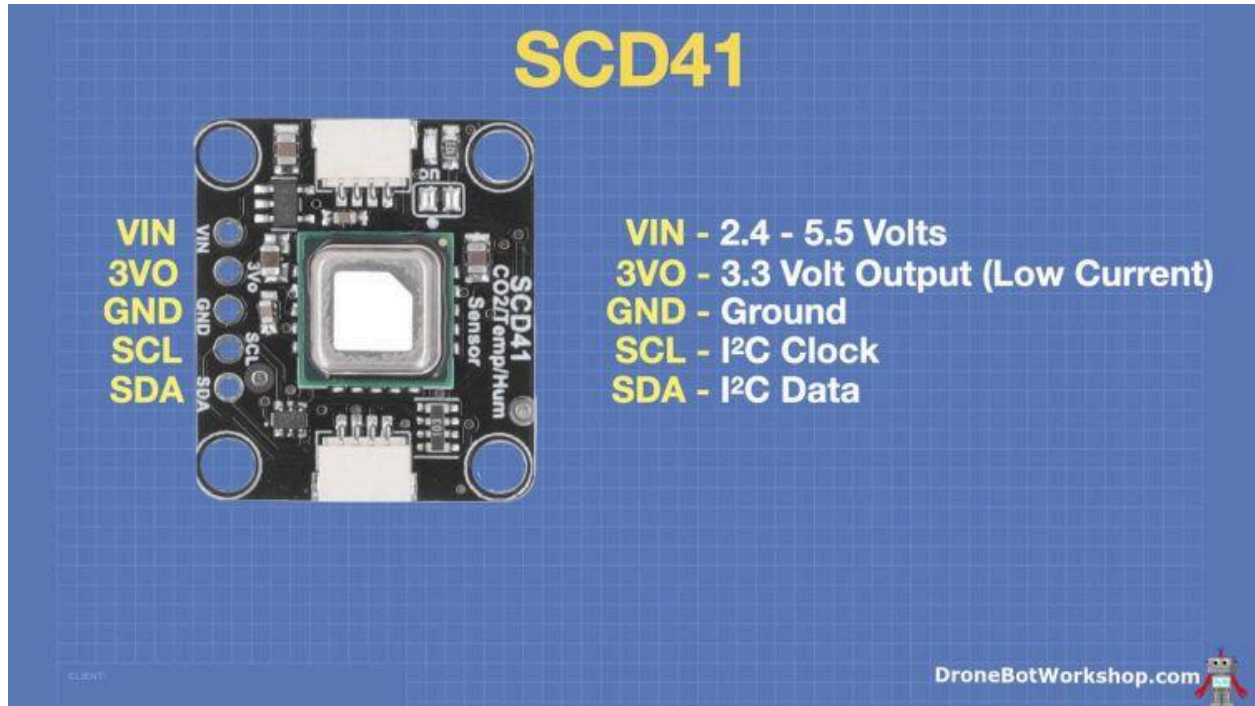


The **SCD41** by Sensirion is a compact, high-precision **NDIR CO₂ sensor** capable of measuring up to 5,000 ppm. It also includes temperature and humidity sensing, making it ideal for indoor air quality applications.

This sensor is more expensive than the ENS160, but it is very accurate. It features a self-calibration function that ensures its accuracy.

SCD41 Pinout

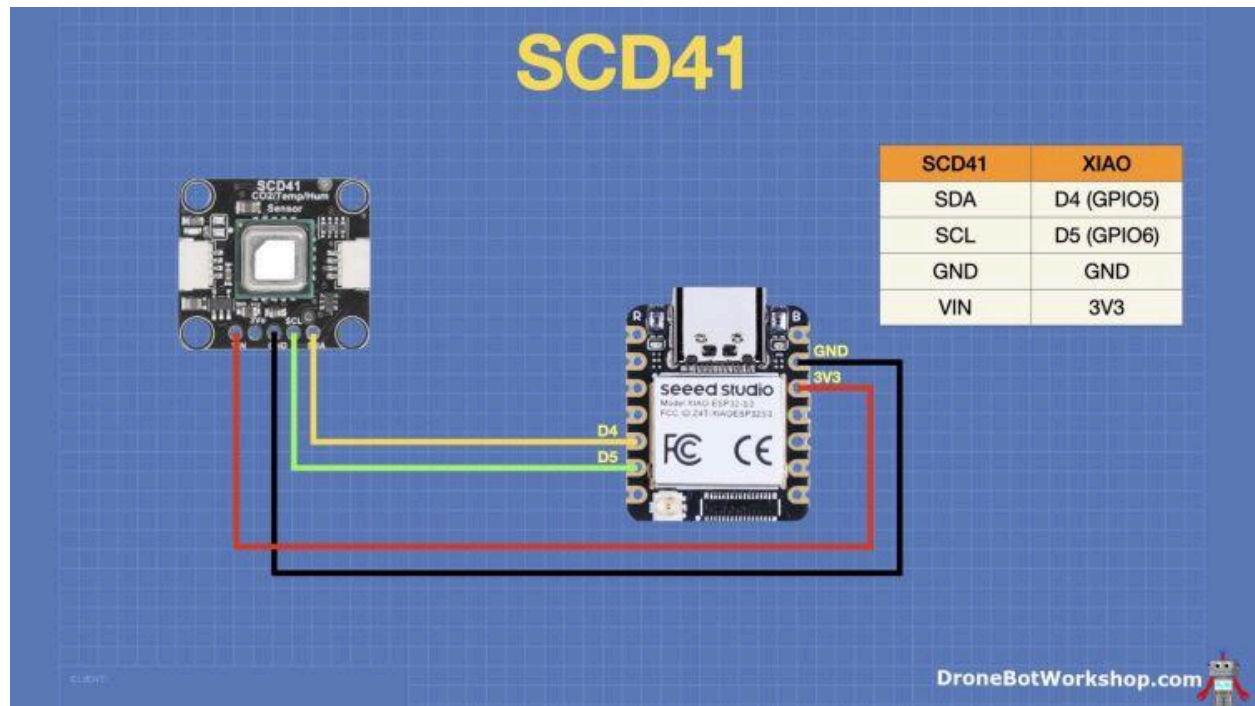
The interface on the SCD41 is nearly identical to the ENS160. This is another I²C sensor that also includes a low-current 3.3-volt reference output.



Once again, you will want to power this with a regulated 3.3 volts to the VIN pin. The device has internal pull-ups on the SDA and SCL lines. It has an I²C address of 0x62.

SCD41 Hookup

The hookup is quite similar to the arrangement with the ENS160. Again, I'm using a Seeedstudio XIAO ESP32-S3 as the microcontroller.



Try to keep the I²C lines as short as possible, as they are susceptible to electrical noise, which can affect the readings.

SCD41 Code

Sensirion manufactures the SCD41, and they have provided a library for using it with the Arduino IDE. Look for the Sensirion Scd4x Library in your Library Manager.

Once you have the library installed, you can run the following code to display the CO₂ levels in your environment:

```
/*
  SCD41 CO2 Sensor
  scd41.ino
  Periodic 5 s measurements (CO2 ppm, Temp C, RH%)
  Uses Sensirion Scd4x Library
  Uses Seeeduino XIAO ESP32-S3, adjust pins for other ESP32

  DroneBot Workshop 2025
  https://dronebotworkshop.com
*/

// Include required libraries
#include <Wire.h>
#include <SensirionI2cScd4x.h>

// I2C Pins and Parameters
#define SDA_PIN 5 // XIAO D4
#define SCL_PIN 6 // XIAO D5
#define SCD41_ADDR 0x62

// Create object for sensor
SensirionI2cScd4x scd4x;

// Output formatting
const uint8_t HEADER_EVERY = 12;
uint8_t rowCount = 0;

// Ventilation hint function
```



```
const char* ventHint(uint16_t co2) {  
    if (co2 >= 1500) return "VENTILATE NOW";  
    if (co2 >= 1000) return "Add fresh air";  
    if (co2 >= 800) return "Okay";  
    return "Good";  
}  
  
// Header Print Function  
void printHeader() {  
    Serial.println();  
    Serial.println(F("+-----+-----+-----+-----+-----+"));  
    Serial.println(F("| Time (s) | CO2 ppm | Temp C | RH % | Ventilation |"));  
    Serial.println(F("+-----+-----+-----+-----+-----+"));  
}  
  
void setup() {  
    // Start Serial Monitor  
    Serial.begin(115200);  
    delay(200);  
  
    // Start I2C  
    Wire.begin(SDA_PIN, SCL_PIN);  
    Wire.setClock(100000);  
  
    // Start SCD41  
    scd4x.begin(Wire, SCD41_ADDR);  
    scd4x.stopPeriodicMeasurement();  
    delay(500);  
}
```

```
// Start Periodic Measurement

if (scd4x.startPeriodicMeasurement() != 0) {

    Serial.println("Failed to start SCD41 periodic measurement.");

    while (1) delay(10);

}

Serial.println(F("SCD41 measuring... first valid sample in ~5-10 s.));

printHeader();

}

void loop() {

    // Only read when data-ready to avoid I2C NACKs

    bool ready = false;

    if (scd4x.getDataReadyStatus(ready) == 0 && ready) {

        uint16_t co2 = 0;

        float tC = NAN, rh = NAN;

        if (scd4x.readMeasurement(co2, tC, rh) == 0 && co2 != 0) {

            if (rowCount % HEADER_EVERY == 0) printHeader();

            Serial.printf("| %8lu | %8u | %7.2f | %6.1f | %-13s |\n",

                           millis() / 1000UL, co2, tC, rh, ventHint(co2));

            Serial.printf("  Note: CO2=%u ppm → %s\n\n", co2, ventHint(co2));

            rowCount++;

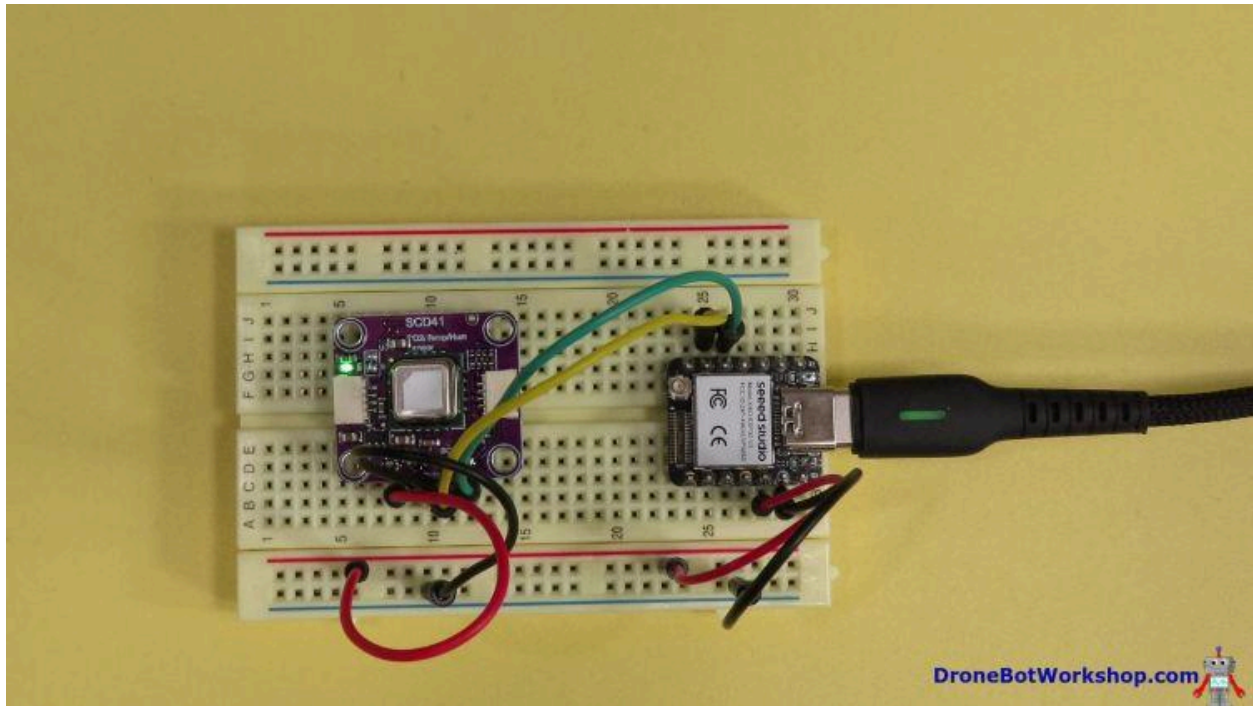
        }

    }

    delay(200); // poll lightly; new SCD41 sample about every 5 s

}
```

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



Load the code to the XIAO and watch the serial monitor. Unlike the previous sketch, this only outputs data approximately once every 5 seconds. As with all these sensors, give the readings time to stabilize.

Here is what the output should look like:

<https://dronebotworkshop.com>

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



The screenshot shows the Arduino IDE interface with the file 'scd41.ino' open. The Serial Monitor is active, displaying data from the 'XIAO_ESP32S3' board at 115200 baud. The data is presented in a table format with columns for Time (s), CO2 ppm, Temp C, RH %, and Ventilation. The CO2 levels are steadily increasing from 844 ppm to 859 ppm. The temperature and humidity readings are also visible.

Time (s)	CO2 ppm	Temp C	RH %	Ventilation
685	844	24.64	68.2	Okay
690	846	24.65	68.3	Okay
694	853	24.66	68.3	Okay
699	859	24.66	68.3	Okay

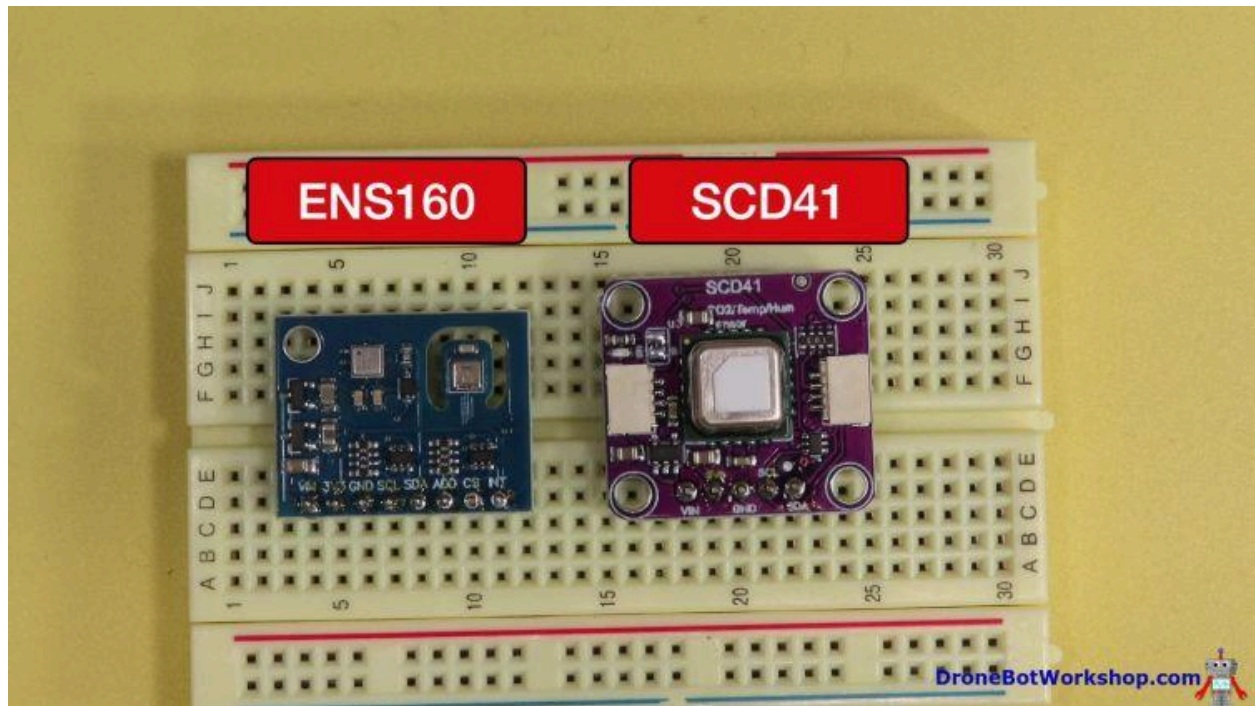
Ln 73, Col 30 XIAO_ESP32S3 on COM7

The readings made more sense to me than the ENS160 readings. I also noted that the temperature and humidity were more in line with my ambient conditions.

I also noted that the CO₂ levels steadily increased as I was filming the video associated with this article. This makes perfect sense – I have a tiny workshop, I close the doors when I record videos, and I’m a considerable source of CO₂!

<https://dronebotworkshop.com>

Using Multiple Sensors



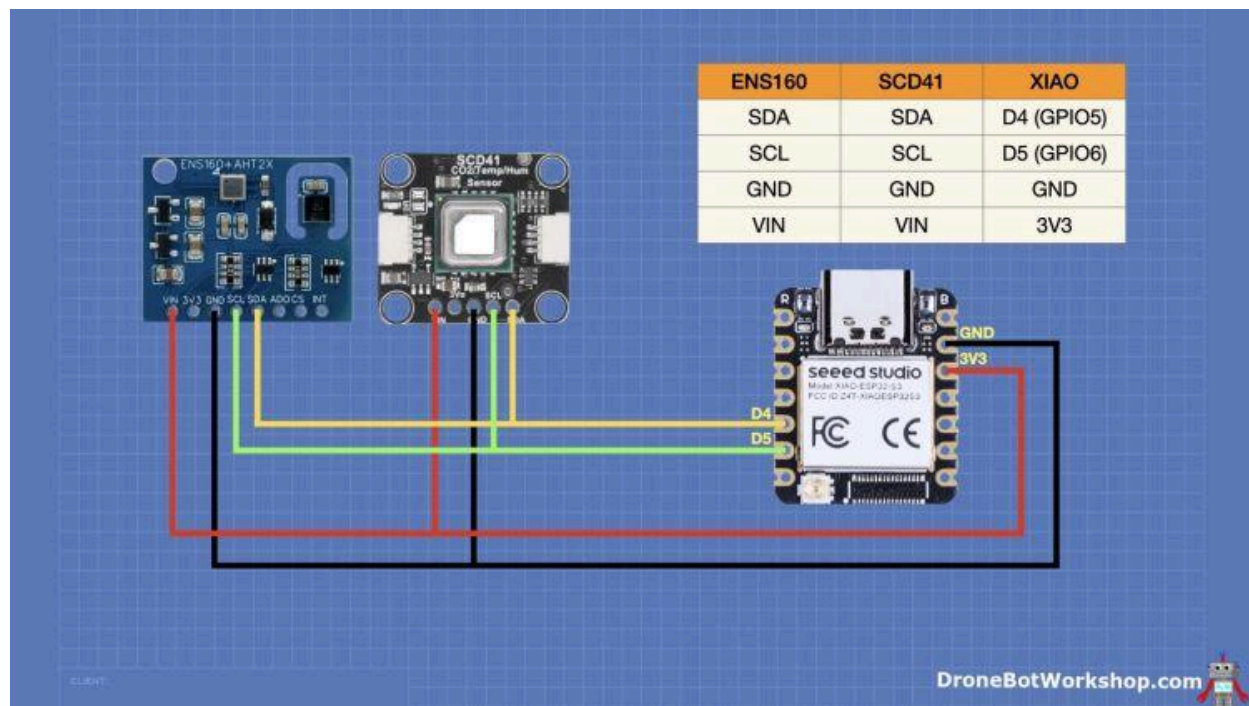
Combining the ENS160 and SCD41 in a single circuit offers several advantages:

- **Redundancy:** eCO₂ readings can be cross-validated with true CO₂ data.
- **Trend Analysis:** VOC spikes (from cooking, cleaning, etc.) can be tracked alongside CO₂ levels.
- **Environmental Context:** Temperature and humidity from both sensors improve compensation and calibration.
- **Smart Control:** More data enables smarter decisions for ventilation, alerts, or automation.

The sensors work well together as they have different I²C addresses.

Multiple Sensor Hookup

As both devices have essentially identical I2C interfaces, hooking them up to a microcontroller like the Seeeduino XIAO is very easy. Here is a diagram of the connections required:



Try to keep the sensors fairly close together to reduce the length of the wires on the I²C bus. Both sensors have internal pull-up resistors, so no additional ones are required.

Multiple Sensor Code

As you might expect, the code we are using to read data from both sensors is essentially an amalgamation of the two sketches we have already examined. We'll format the output to include values from both sensors.


```
/*  
  XIAO ESP32-S3 Integrated Air-Quality Logger  
  multisensor.ino  
  Uses SCD41, ENS160 & AHT21  
  True CO2 from SCD41  
  VOC/eCO2/AQI from ENS160  
  T/RH from AHT21 (also feeds ENS160 compensation)  
  
  DroneBot Workshop 2025  
  https://dronebotworkshop.com  
*/  
  
// Include required libraries  
#include <Wire.h>  
#include <SensirionI2cScd4x.h>  
#include "SparkFun_ENS160.h"  
#include <Adafruit_AHTX0.h>  
  
// I2C  
#define SDA_PIN    5  
#define SCL_PIN    6  
#define SCD41_ADDR 0x62  
  
// Create Objects  
SensirionI2cScd4x scd4x;  
SparkFun_ENS160   ens;  
Adafruit_AHTX0    aht;
```

```
// Header format

const uint8_t HEADER_EVERY = 12;

uint8_t rowCount = 0;

// Latest values

uint16_t co2_ppm = 0;

float tC_scd = NAN, rh_scd = NAN;

float tC_aht = NAN, rh_aht = NAN;

uint16_t tvoc_ppb = 0, eco2_ppm = 0;

uint8_t aqi = 0;

// AQI Function

const char* aqiText(uint8_t v){

    switch (v){ case 1:return "Excellent"; case 2:return "Good"; case 3:return
    "Moderate";

                case 4:return "Poor"; case 5:return "Unhealthy"; default:return "?"; }

}

// Ventilation Hint Function

const char* ventHint(uint16_t co2){

    if (co2 >= 1500) return "VENTILATE NOW";

    if (co2 >= 1000) return "Add fresh air";

    if (co2 >= 800)  return "Okay";

    return "Good";

}

// Formatting

void printFloatOrDash(float v, uint8_t width, uint8_t prec){

    if (isnan(v)) { for (uint8_t i=0;i<width;i++) Serial.print(i==width/2?'-':' '); }

}
```

```
    else { char buf[20]; dtostrf(v, width, prec, buf); Serial.print(buf); }
}

// Print Header Function
void printHeader() {
    Serial.println();

    Serial.println(F("+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+"));

    Serial.println(F(" | Time (s) | CO2 ppm | T_SCD C | RH_SCD | T_AHT C | RH_AHT | TVOC
ppb | eCO2 ppm | AQI | AQI (text) |"));

    Serial.println(F("+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+"));
}

void setup() {
    // Start Serial Monitor
    Serial.begin(115200);

    delay(200);

    // Start I2C
    Wire.begin(SDA_PIN, SCL_PIN);

    Wire.setClock(400000);

    // ENS160
    if (!ens.begin()) { Serial.println("ENS160 not found (0x53/0x52)."); while (1)
delay(100); }

    ens.setOperatingMode(SFE_ENS160_STANDARD);

    // AHT21
```

```
if (!aht.begin()) { Serial.println("AHT21 not found (0x38)."); while (1) delay(100);
}

// SCD41

scd4x.begin(Wire, SCD41_ADDR);
scd4x.stopPeriodicMeasurement();
delay(500);
scd4x.startPeriodicMeasurement();

Serial.println(F("Integrated air-quality monitor running... (SCD41 updates ~5 s)"));
printHeader();
}

void loop() {

// --- SCD41: read only when ready ---

bool ready = false;

if (scd4x.getDataReadyStatus(ready) == 0 && ready) {

    uint16_t co2; float tC; float rh;

    if (scd4x.readMeasurement(co2, tC, rh) == 0 && co2 != 0) {

        co2_ppm = co2; tC_scd = tC; rh_scd = rh;

    }

}

// --- AHT21 ---

sensors_event_t hum, temp;

if (aht.getEvent(&hum, &temp)) {

    tC_aht = temp.temperature;

    rh_aht = hum.relative_humidity;

}
```

```
// --- ENS160 (~1 Hz) ---

if (ens.checkDataStatus()) {
    tvoc_ppb = ens.getTVOC();
    eco2_ppm = ens.getECO2();
    aqi      = ens.getAQI();
}

// --- Print once per second ---

static uint32_t lastPrint = 0;

if (millis() - lastPrint >= 1000) {
    lastPrint = millis();

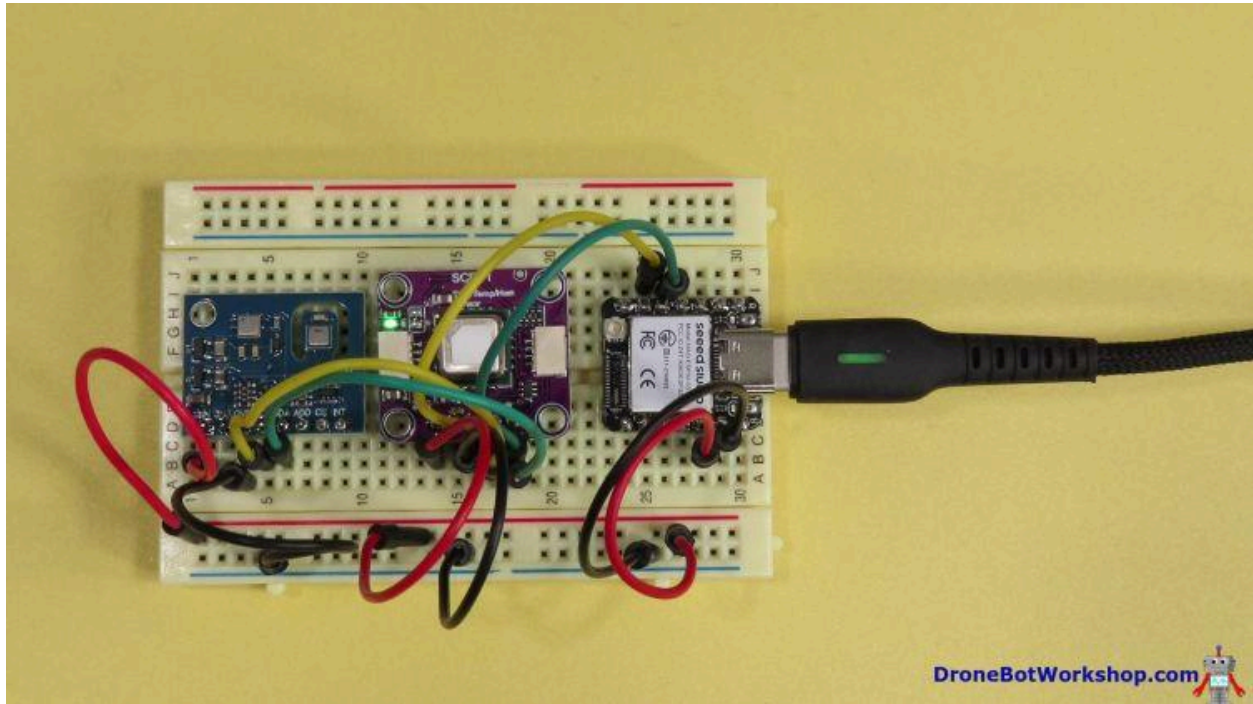
    if (rowCount % HEADER_EVERY == 0) printHeader();

    Serial.print("| ");
    Serial.printf("%8lu | %8u | ", millis() / 1000UL, co2_ppm);
    printFloatOrDash(tC_scd, 7, 2); Serial.print(" | ");
    printFloatOrDash(rh_scd, 6, 1); Serial.print(" | ");
    printFloatOrDash(tC_aht, 7, 2); Serial.print(" | ");
    printFloatOrDash(rh_aht, 6, 1); Serial.print(" | ");
    Serial.printf("%9u | %9u | %u | %-12s |\n",
                  tvoc_ppb, eco2_ppm, aqi, aqiText(aqi));

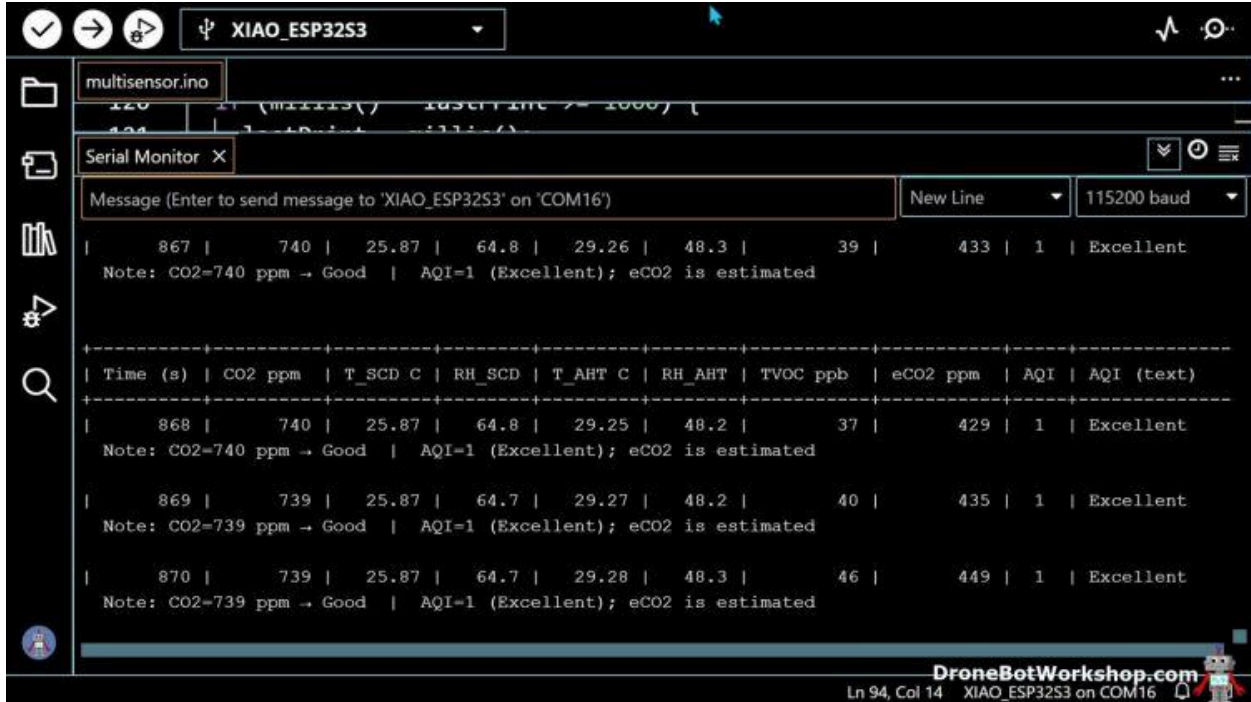
    Serial.printf("  Note: CO2=%u ppm → %s | AQI=%u (%s); eCO2 is estimated\n\n",
                  co2_ppm, ventHint(co2_ppm), aqi, aqiText(aqi));

    rowCount++;
}
```

```
delay(50);  
}
```



Load the code onto the XIAO and give both sensors time to warm up. While you will start getting readings instantly, please wait at least three minutes for them to become accurate. In my experience, I found that it took nearly an hour for them to stabilize fully.



The screenshot shows the Arduino IDE interface with the 'Serial Monitor' window open. The monitor displays a table of air quality data collected from an ESP32. The data includes time, CO2 concentration, temperature, humidity, TVOC, eCO2, and AQI. The AQI is consistently 1, indicating 'Excellent' air quality. The status bar at the bottom indicates 'Ln 94, Col 14' and 'XIAO_ESP32S3 on COM16'.

Time (s)	CO2 ppm	T_SCD C	RH_SCD	T_AHT C	RH_AHT	TVOC ppb	eCO2 ppm	AQI	AQI (text)
867	740	25.87	64.8	29.26	48.3	39	433	1	Excellent
Note: CO2=740 ppm → Good AQI=1 (Excellent); eCO2 is estimated									
868	740	25.87	64.8	29.25	48.2	37	429	1	Excellent
Note: CO2=740 ppm → Good AQI=1 (Excellent); eCO2 is estimated									
869	739	25.87	64.7	29.27	48.2	40	435	1	Excellent
Note: CO2=739 ppm → Good AQI=1 (Excellent); eCO2 is estimated									
870	739	25.87	64.7	29.28	48.3	46	449	1	Excellent
Note: CO2=739 ppm → Good AQI=1 (Excellent); eCO2 is estimated									

The combination of both sensors will provide you with a great deal of insight into the quality of the air you are breathing. By combining the ENS160 and SCD41, you gain a more comprehensive understanding of indoor air quality and the tools to respond intelligently.

Conclusion

CO₂ and eCO₂ sensors, such as the ENS160 and SCD41, enable us to effectively monitor and improve indoor air quality.

For future projects, consider building a portable CO₂ meter with OLED display for real-time feedback, or an automated ventilation fan that activates above 1,000 ppm using relays and ESP32's Wi-Fi for IoT integration (e.g., Home Assistant). Expand to data logging on SD cards or cloud services for long-term analysis, or incorporate

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

machine learning to predict air quality trends. With climate awareness growing, these sensors open doors to innovative solutions—start experimenting today!