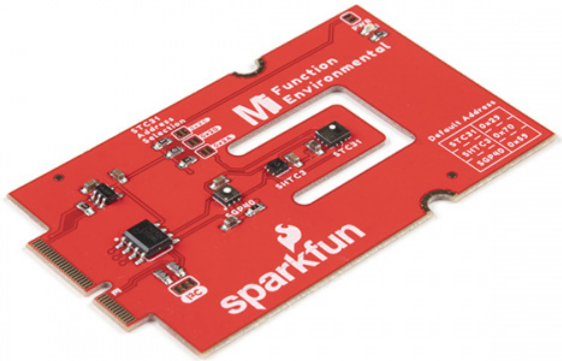
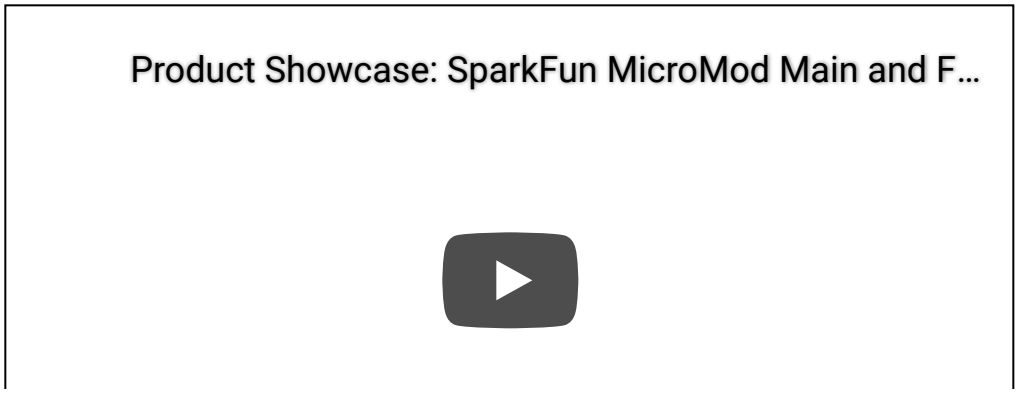
MicroMod Environmental Function Board Hookup Guide

Introduction

The SparkFun MicroMod Environmental Function Board adds additional sensing options to the MicroMod Processor Boards. This function board includes three sensors to monitor air quality (SGP40), humidity & temperature (SHTC3), and CO_2 concentrations (STC31) in your indoor environment. To make it even easier to use, all communication is over the MicroMod's I^2C bus! In this tutorial, we will go over how to connect the board and read the sensors.

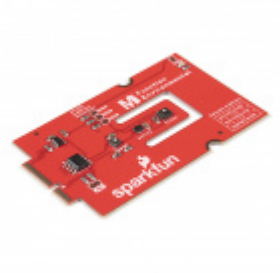


SparkFun MicroMod Environmental Function Board
SEN-18632



Required Materials

To follow along with this tutorial, you will need the following materials at a minimum. 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.



SparkFun MicroMod Environmental Function Board

SEN-18632

SparkFun MicroMod Main Board - Single

DEV-18575

Reversible USB A to C Cable - 2m

CAB-15424

SparkFun MicroMod Artemis Processor

DEV-16401

Pocket Screwdriver Set

TOL-12891

microSD Card - 1GB (Class 4)

COM-15107

MicroMod Main Board

To hold the processor and function boards, you will need one Main board. Depending on your application, you may choose to have either one or two function boards.

SparkFun MicroMod Main Board - Single

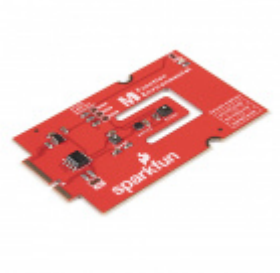
DEV-18575

SparkFun MicroMod Main Board - Double

DEV-18576

MicroMod Function Board

To add additional functionality to your Processor Board, you'll want to include one or two function boards when connecting them to the Main Board. Besides the MicroMod Environmental Function Board which this tutorial is focused on, you may descide to add the WiFi Function Board to the mix. Make sure to adjust the cart and include the MicroMod Main Board - Double as opposed to the Single when using two Function Boards. Check out the SparkFun catalog for other function boards.



SparkFun MicroMod Environmental Function Board

SEN-18632

SparkFun MicroMod WiFi Function Board - ESP32

WRL-18430

Tools

You will need a screw driver to secure the Processor and Function boards.

SparkFun Mini Screwdriver
TOL-09146

Suggested Reading

If you aren't familiar with the MicroMod ecosystem, we recommend reading here for an overview.

MicroMod

MicroMod Ecosystem

If you aren’t familiar with the following concepts, we also recommend checking out a few of these tutorials before continuing. Make sure to check the respective hookup guides for your processor board and function board to ensure that you are installing the correct USB-to-serial converter. You may also need to follow additional instructions that are not outlined in this tutorial to install the appropriate software.

What is an Arduino?

What is this 'Arduino' thing anyway? This tutorials dives into what an Arduino is and along with Arduino projects and widgets.

Installing Arduino IDE

A step-by-step guide to installing and testing the Arduino software on Windows, Mac, and Linux.

How to Install CH340 Drivers

How to install CH340 drivers (if you need them) on Windows, Mac OS X, and Linux.

SparkFun Humidity Sensor Breakout - SHTC3 (Qwiic) Hookup Guide

A Hookup Guide to get started using the SHTC3 breakout.

Getting Started with MicroMod

Dive into the world of MicroMod - a compact interface to connect a microcontroller to various peripherals via the M.2 Connector!

Air Quality Sensor - SGP40 (Qwiic) Hookup Guide

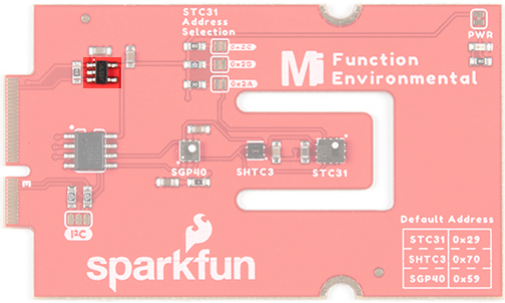
Get started measuring indoor air quality with the SparkFun Air Quality Sensor - SGP40 (Qwiic) Hookup Guide.

Hardware Overview

This section goes over the important features on the MicroMod Environmental Function Board. Of course, we recommend checking out the Resources and Going Further for more information on each sensor.

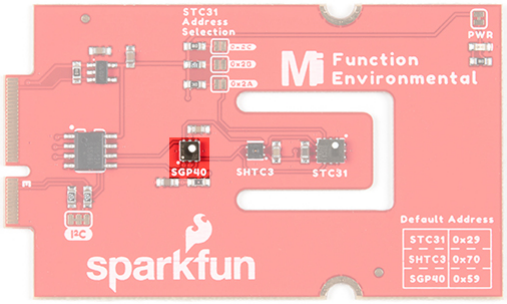
Power

To power the board, you will need to apply power to a SparkFun Main Board. Power applied will connect to the Function Board's VIN pin, which will be regulated down for the rest of the board with the AP2112 3.3V/600mA voltage regulator.



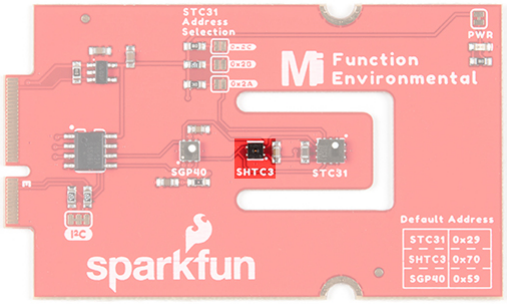
SGP40

The board includes the Sensirion SGP40 sensor IC which measures air quality. The reserved I^2C address for the SGP40 is **0x59**. For easy reference, the default address for the SGP40 is labeled on the board.



SHTC3

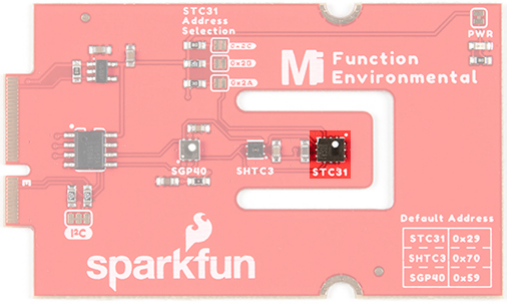
The board includes the Sensirion SHTC3 sensor IC which measures humidity and temperature. The reserved I^2C address for the SHTC3 is **0x70**. For easy reference, the default address for the SHTC3 is labeled on the board.



> **Note:** A multiplexer/Mux is required to communicate to multiple SHTC3 sensors on a single bus. The SHTC3 uses the same address as the Qwiic Mux (**0x70**). For advanced users that are using multiple SHTC3's with the Qwiic Mux, you will need to adjust the Qwiic Mux's default address.

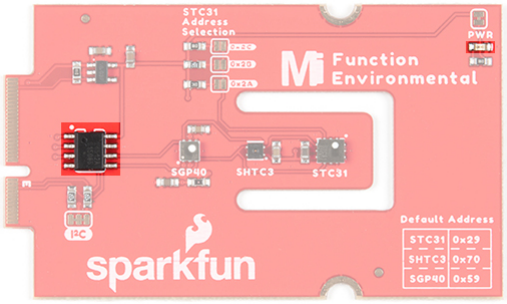
STC31

The board includes the Sensirion STC31 sensor IC which measures CO_2 concentrations in N_2 and CO_2 in air. The reserved I^2C address for the STC31 is **0x29**. For easy reference, the default address for the STC31 is labeled on the board.



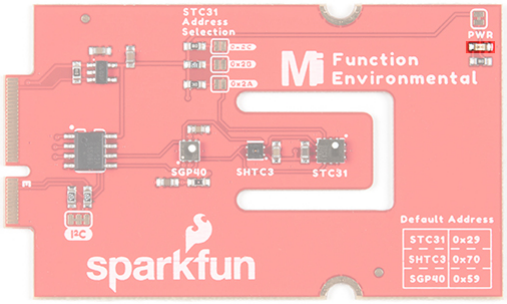
EEPROM

The board includes an I^2C EEPROM. Unfortunately, this is not available for the user and was meant to hold board specific information.



LED

There is one LED to indicate when there is power available. You can disable the LED with the PWR jumper

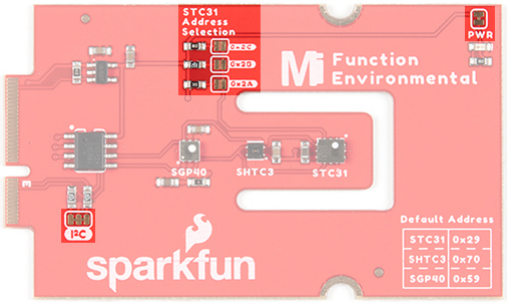


Jumpers

> **Note:** If this is your first time working with jumpers, check out the How to Work with Jumper Pads and PCB Traces tutorial for more information.

The following jumpers are included to configure the board.

- **PWR** - By default, the jumper with the label `PWR` is closed. This jumper connects the 3.3V line and LED. Cutting this jumper will disable the LED.
- **I²C Pull-up Resistors** - By default, this three way jumper labeled `I²C` is closed and connects two pull-up resistors to the I²C data lines. If you have many devices on your I²C data lines, then you may consider cutting these two jumpers.
- **STC31 Address Selection** - There are three jumpers available on the board to adjust the STC31's address. By default, the jumpers are open. The alternative addresses for the sensor are **0x2A**, **0x2B**, and **0x2C**. To select the address, you will need to close the jumper by adding a solder blob to one of the solder jumpers.



MicroMod Function Board Pinout

Depending on your window size, you may need to use the horizontal scroll bar at the bottom of the table to view the additional pin functions. Note that the M.2 connector pins on opposing sides are offset from each other as indicated by the bottom pins where it says (Not Connected)*. There is no connection to pins that have a "-" under the primary function.

MICROMOD ENVIRONMENT FUNCTION BOARD PINOUT TABLE

MICROMOD GENERAL PROCESSOR PINOUT TABLE

MICROMOD GENERAL PIN DESCRIPTIONS

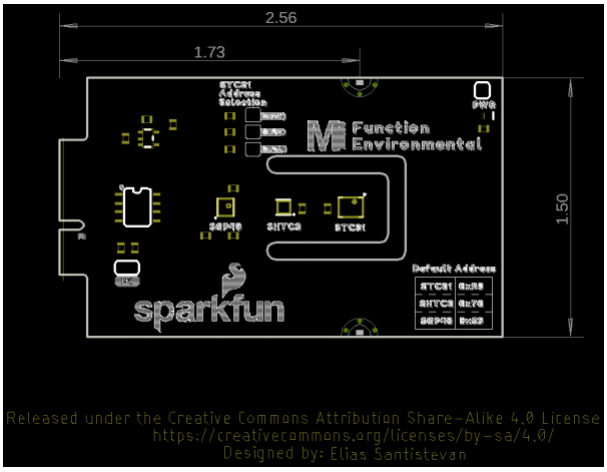
AUDIO	UART	GPIO/BUS	I²C	SDIO	SPI0	Dedicated

Alternative Function	Primary Function	Bottom Pin	Top Pin	Primary Function	Alternative Function
	(Not Connected)		**75**	GND	
	VIN	**74**	**73**	3.3V	
	VIN	**72**	**71**	Power EN	
	-	**70**	**69**	-	
	-	**66**	**65**	-	
	-	**64**	**63**	-	
	-	**62**	**61**	-	
	-	**60**	**59**	-	
	-	**58**	**57**	-	
	-	**56**	**55**	-	
	-	**54**	**53**	-	
	-	**52**	**51**	-	
	-	**50**	**49**	-	
	-	**48**	**47**	-	
	-	**46**	**45**	GND	
	-	**44**	**43**	-	

	-	**42**	**41**	-	
	-	**40**	**39**	GND	
	-	**38**	**37**	-	
	EEPROM_A0	**36**	**35**	-	
	EEPROM_A1	**34**	**33**	GND	
	EEPROM_A2	**32**	**31**	Module Key	
	Module Key	**30**	**29**	Module Key	
	Module Key	**28**	**27**	Module Key	
	Module Key	**26**	**25**	Module Key	
	Module Key	**24**	**23**	-	
	-	**22**	**21**	I2C_SCL	
	-	**20**	**19**	I2C_SDA	
	-	**18**	**17**	-	
	-	**16**	**15**	-	
	-	**14**	**13**	-	
	-	**12**	**11**	-	
	-	**10**	**9**	-	
	-	**8**	**7**	-	
	-	**6**	**5**	-	
	-	**4**	**3**	-	
	-	**2**	**1**	GND	

Board Dimensions

The board uses the standard MicroMod Function Board size which measures about 1.50"x2.56".



Hardware Hookup

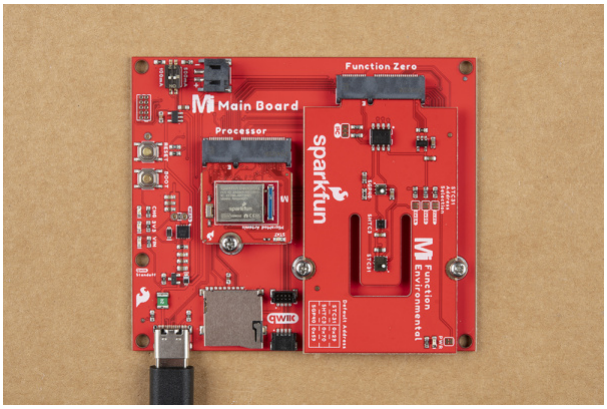
If you have not already, make sure to check out the Getting Started with MicroMod: Hardware Hookup for information on inserting your Processor and Function Boards to the Main Board.

Getting Started with MicroMod

OCTOBER 21, 2020

Dive into the world of MicroMod - a compact interface to connect a microcontroller to various peripherals via the M.2 Connector!

After securing the Processor and Function Board to the Main Board, your setup should look like the image below. Connect a USB Type C Cable to begin programming your Processor Board. In this case, we used the MicroMod Main Board - Single, MicroMod Artemis Processor, and MicroMod Environmental Function Board.



Software 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 the following tutorials.

- Installing the Arduino IDE
- Installing Board Definitions in the Arduino IDE
- Installing an Arduino Library

Arduino Board Definitions and Driver

We'll assume that you installed the necessary board files and drivers for your Processor Board. In this case, we used the MicroMod Artemis Processor Board which uses the CH340 USB-to-serial converter. If you are using a Processor Board, make sure to check out its hookup guide for your Processor Board.

Installing Board Definitions in the Arduino IDE

SEPTEMBER 9, 2020

How do I install a custom Arduino board/core? It's easy! This tutorial will go over how to install an Arduino board definition using the Arduino Board Manager. We will also go over manually installing third-party cores, such as the board definitions required for many of the SparkFun development boards.

MicroMod Artemis Processor Board Hookup Guide

OCTOBER 21, 2020

Get started with the Artemis MicroMod Processor Board in this tutorial!

How to Install CH340 Drivers

AUGUST 6, 2019

How to install CH340 drivers (if you need them) on Windows, Mac OS X, and Linux.

Arduino Library

The SparkFun SGP40, SHTC3, and STC3X Arduino libraries can be downloaded with the Arduino library manager by searching '**SparkFun SGP40**,' '**SHTC3**,' and '**STC3X**'. Or you can grab the zip here from each respective GitHub repository (SGP40, SHTC3, STC3X) to manually install:

SPARKFUN SGP40 ARDUINO LIBRARY (ZIP)

SPARKFUN SHTC3 ARDUINO LIBRARY (ZIP)

SPARKFUN STC3X ARDUINO LIBRARY (ZIP)

Arduino Examples

Example 1: Reading SHTC3, STC31, and SGP40

Below is the combined example to read SHTC3, STC31, and SGP40. If you have not already, select your Board (in this case the **MicroMod Artemis**), and associated COM port. Copy and paste the code below in your Arduino IDE. Hit the upload button and set the serial monitor to **115200** baud.

```
/*************************************************************************

  WRITTEN BY: Ho Yun "Bobby" Chan
  @ SparkFun Electronics
  DATE: 10/19/2021
  GITHUB REPO: https://github.com/sparkfun/MicroMod_Environmental_Sensor_Function_Board
  DEVELOPMENT ENVIRONMENT SPECIFICS:
    Firmware developed using Arduino IDE v1.8.12

  ========== DESCRIPTION==========
  This example code combines example codes from the SHTC3, STC31, and SGP40 libraries.
  Most of the steps to obtain the measurements are the same as the example code.
  Generic object names were renamed (e.g. mySensor => mySGP40 and mySTC3x).

     Example 1: Basic Relative Humidity and Temperature Readings  w/ SHTC3; Written by Owen Lyke
     Example 2: PHT (SHTC3) Compensated CO2 Readings w/ STC31; Written by Paul Clark and based o
n earlier code by Nathan Seidle
     Example 1: Basic VOC Index w/ SGP40; Written by Paul Clark

  Open a Serial Monitor at 115200 baud to view the readings!

  Note: You may need to wait about ~5 minutes after starting up the code before VOC index
  has any values.

  ========== HARDWARE CONNECTIONS ==========
  MicroMod Artemis Processor Board => MicroMod Main Board => MicroMod Environmental Function Boa
rd (with SHTC3, STC31, and SGP40)

  Feel like supporting open source hardware?
  Buy a board from SparkFun!
       MicroMod MicroMod Artemis Processor   | https://www.sparkfun.com/products/16401
       MicroMod Main Board - Single          | https://www.sparkfun.com/products/18575
       MicroMod Environmental Function Board | https://www.sparkfun.com/products/18632

  You can also get the sensors individually.

       Qwiic SHTC3 | https://www.sparkfun.com/products/16467
       Qwiic STC31 | https://www.sparkfun.com/products/18385
       Qwiic SGP40 | https://www.sparkfun.com/products/17729

  LICENSE: This code is released under the MIT License (http://opensource.org/licenses/MIT)

*************************************************************************/



#include <Wire.h>

#include "SparkFun_SHTC3.h" //Click here to get the library: http://librarymanager/All#SparkFun_
SHTC3
SHTC3 mySHTC3; // Create an object of the SHTC3 class

#include "SparkFun_STC3x_Arduino_Library.h" //Click here to get the library: http://librarymanag
```

```
er/All#SparkFun_STC3x
STC3x mySTC3x; // Create an object of the mySTC3x class

#include "SparkFun_SGP40_Arduino_Library.h" // Click here to get the library: http://librarymana
ger/All#SparkFun_SGP40
SGP40 mySGP40; //Create an object of the SGP40 class

float RH = 0.00; // Variable to keep track of SHTC3 temperature compensation for the STC31
float temperature = 0.00; // Variable to keep track of SHTC3 relative humidity compensation for
 the STC31




void setup() {

  Serial.begin(115200);
  //while (!Serial) ; // Wait for Serial Monitor/Plotter to open for Processors with Native USB
 (i.e. SAMD51)
  Serial.println(F("Initializing Combined Example w/ SGP40, SHTC3, and STC31."));
  Wire.begin();

  //mySTC3x.enableDebugging(); // Uncomment this line to get helpful debug messages on Serial
  //mySGP40.enableDebugging(); // Uncomment this line to print useful debug messages to Serial



  if (mySHTC3.begin() != SHTC3_Status_Nominal)
  {
    Serial.println(F("SHTC3 not detected. Please check wiring. Freezing..."));
    while (1)
      ; // Do nothing more
  }

  if (mySTC3x.begin() == false)
  {
    Serial.println(F("STC3x not detected. Please check wiring. Freezing..."));
    while (1)
      ; // Do nothing more
  }

  if (mySGP40.begin() == false)
  {
    Serial.println(F("SGP40 not detected. Check connections. Freezing..."));
    while (1)
      ; // Do nothing more
  }



  //We need to tell the STC3x what binary gas and full range we are using
  //Possible values are:
```

```
  //  STC3X_BINARY_GAS_CO2_N2_100   : Set binary gas to CO2 in N2.  Range: 0 to 100 vol%
  //  STC3X_BINARY_GAS_CO2_AIR_100  : Set binary gas to CO2 in Air. Range: 0 to 100 vol%
  //  STC3X_BINARY_GAS_CO2_N2_25    : Set binary gas to CO2 in N2.  Range: 0 to 25 vol%
  //  STC3X_BINARY_GAS_CO2_AIR_25   : Set binary gas to CO2 in Air. Range: 0 to 25 vol%
  if (mySTC3x.setBinaryGas(STC3X_BINARY_GAS_CO2_AIR_25) == false)
  {
    Serial.println(F("Could not set the binary gas! Freezing..."));
    while (1)
      ; // Do nothing more
  }



  //We can compensate for temperature and relative humidity using the readings from the SHTC3

  if (mySHTC3.update() != SHTC3_Status_Nominal) // Request a measurement
  {
    Serial.println(F("Could not read the RH and T from the SHTC3! Freezing..."));
    while (1)
      ; // Do nothing more
  }

  //In case the ‘Set temperature command’ has been used prior to the measurement command,
  //the temperature value given out by the STC31 will be that one of the ‘Set temperature comman
d’.
  //When the ‘Set temperature command’ has not been used, the internal temperature value can be
 read out.
  temperature = mySHTC3.toDegC(); // "toDegC" returns the temperature as a floating point number
in deg C
  Serial.print(F("Setting STC3x temperature to "));
  Serial.print(temperature, 2);
  Serial.print(F("C was "));
  if (mySTC3x.setTemperature(temperature) == false)
    Serial.print(F("not "));
  Serial.println(F("successful"));

  RH = mySHTC3.toPercent(); // "toPercent" returns the percent humidity as a floating point numb
er
  Serial.print(F("Setting STC3x RH to "));
  Serial.print(RH, 2);
  Serial.print(F("% was "));
  if (mySTC3x.setRelativeHumidity(RH) == false)
    Serial.print(F("not "));
  Serial.println(F("successful"));

  //If we have a pressure sensor available, we can compensate for ambient pressure too.
  //As an example, let's set the pressure to 840 mbar (== SF Headquarters)
  uint16_t pressure = 840;
  Serial.print(F("Setting STC3x pressure to "));
  Serial.print(pressure);
  Serial.print(F("mbar was "));
  if (mySTC3x.setPressure(pressure) == false)
    Serial.print(F("not "));
  Serial.println(F("successful"));
```

```
  Serial.println(F("Note: Relative humidity and temperature compensation for the STC31 will be u
pdated frequently in the main loop() function."));

} //end setup()




void loop() {

  //==============================
  //==========READ SHTC3==========
  //==============================
  //minimum update rate = ~100Hz

  SHTC3_Status_TypeDef result = mySHTC3.update();           // Call "update()" to command a meas
urement, wait for measurement to complete, and update the RH and T members of the object

  RH = mySHTC3.toPercent();                                 // "toPercent" returns the percent h
umidity as a floating point number
  Serial.print(F("RH = "));
  Serial.print(RH);

  Serial.print(F("%, T = "));
  Serial.print(mySHTC3.toDegF());                           // "toDegF" return the temperature a
s a flaoting point number in deg F
  Serial.print(F(" deg F, "));

  temperature = mySHTC3.toDegC();                           // "toDegC" returns the temperature
 as a floating point number in deg C
  Serial.print(temperature);
  Serial.print(F(" deg C"));

  if (mySHTC3.lastStatus == SHTC3_Status_Nominal)           // You can also assess the status of
the last command by checking the ".lastStatus" member of the object
  {
    Serial.println("");                                         //Sample data good, no need to o
utput a message
  }
  else {
    Serial.print(F(",     Update failed, error: "));        //notify user if there is an error
    errorDecoder(mySHTC3.lastStatus);
    Serial.println("");
  }



  //==============================
  //==========READ STC31==========
  //==============================
  //minimum update rate = 1Hz
```

```
  if (mySTC3x.setRelativeHumidity(RH) == false)
    Serial.print(F("Unable to set STC31 Relative Humidity with SHTC3."));

  if (mySTC3x.setTemperature(temperature) == false)
    Serial.println(F("Unable to set STC31 Temperature with SHTC3."));


  Serial.print(F("CO2(%): "));

  if (mySTC3x.measureGasConcentration())                  // measureGasConcentration will retur
n true when fresh data is available
  {
    Serial.println(mySTC3x.getCO2(), 2);
  }
  else
  {
    Serial.print(mySTC3x.getCO2(), 2);
    Serial.println(F(",     (old STC3 sample reading, STC31 was not able to get fresh data yet)"
));  //output this note to indicate  when we are not able to obtain a new measurement
  }



  //==============================
  //==========READ SGP40==========
  //==============================
  //minimum update rate = 1Hz

  Serial.print(F("VOC Index is: "));
  Serial.println(mySGP40.getVOCindex()); //Get the VOC Index using the default RH (50%) and T (2
5C)



  //================================
  //=========SPACE & DELAY==========
  //================================
  //Serial.println("");// Uncomment this line to add some space between readings for the Serial
 Monitor
  delay(1000); //Wait 1 second - the Sensirion VOC and CO2 algorithms expects a sample rate of 1
Hz

}//end loop()




void errorDecoder(SHTC3_Status_TypeDef message)                             // The errorDecoder
 function prints "SHTC3_Status_TypeDef" results in a human-friendly way
{
  switch (message)
```

```
  {
    case SHTC3_Status_Nominal : Serial.print("Nominal"); break;
    case SHTC3_Status_Error : Serial.print("Error"); break;
    case SHTC3_Status_CRC_Fail : Serial.print("CRC Fail"); break;
    default : Serial.print("Unknown return code"); break;
  }
}
```

Example 2: Reading SHTC3, STC31, and SGP40 in CSV

Below is the same combined code but formatted for CSV. If you have not already, select your Board (in this case the **MicroMod Artemis**), and associated COM port. Copy and paste the code below in your Arduino IDE. Hit the upload button and set the serial monitor to **115200** baud.

```
/*************************************************************************

  WRITTEN BY: Ho Yun "Bobby" Chan
  @ SparkFun Electronics
  DATE: 10/19/2021
  GITHUB REPO: https://github.com/sparkfun/MicroMod_Environmental_Sensor_Function_Board
  DEVELOPMENT ENVIRONMENT SPECIFICS:
    Firmware developed using Arduino IDE v1.8.12

  ========== DESCRIPTION==========
  This example code combines example codes from the SHTC3, STC31, and SGP40 libraries.
  Most of the steps to obtain the measurements are the same as the example code.
  Generic object names were renamed (e.g. mySensor => mySGP40 and mySTC3x).

     Example 1: Basic Relative Humidity and Temperature Readings  w/ SHTC3; Written by Owen Lyke
     Example 2: PHT (SHTC3) Compensated CO2 Readings w/ STC31; Written by Paul Clark and based o
n earlier code by Nathan Seidle
     Example 1: Basic VOC Index w/ SGP40; Written by Paul Clark

  Open a Serial Monitor/Plotter at 115200 baud to view the readings!

  Note: You may need to wait about ~5 minutes after starting up the code before VOC index
  has any values.

  ========== HARDWARE CONNECTIONS ==========
  MicroMod Artemis Processor Board => MicroMod Main Board => MicroMod Environmental Function Boa
rd (with SHTC3, STC31, and SGP40)

  Feel like supporting open source hardware?
  Buy a board from SparkFun!
       MicroMod MicroMod Artemis Processor   | https://www.sparkfun.com/products/16401
       MicroMod Main Board - Single          | https://www.sparkfun.com/products/18575
       MicroMod Environmental Function Board | https://www.sparkfun.com/products/18632

  You can also get the sensors individually.
       SHTC3 | https://www.sparkfun.com/products/16467
       STC31 | https://www.sparkfun.com/products/18385
       SGP40 | https://www.sparkfun.com/products/17729

  LICENSE: This code is released under the MIT License (http://opensource.org/licenses/MIT)

*************************************************************************/



#include <Wire.h>

#include "SparkFun_SHTC3.h" //Click here to get the library: http://librarymanager/All#SparkFun_
SHTC3
SHTC3 mySHTC3; // Create an object of the SHTC3 class

#include "SparkFun_STC3x_Arduino_Library.h" //Click here to get the library: http://librarymanag
er/All#SparkFun_STC3x
```

```
STC3x mySTC3x; // Create an object of the STC3x class

#include "SparkFun_SGP40_Arduino_Library.h" // Click here to get the library: http://librarymana
ger/All#SparkFun_SGP40
SGP40 mySGP40; //Create an object of the SGP40 class

float RH = 0.00; // Variable to keep track of SHTC3 temperature compensation for the STC31
float temperature = 0.00; // Variable to keep track of SHTC3 relative humidity compensation for
 the STC31

//Debug mode, comment one of these lines out using a syntax
//for a single line comment ("//"):
#define DEBUG 0     //0 = Output for Serial Plotter, CSV
//#define DEBUG 1     //1 = Output for Serial Monitor




void setup() {

  Serial.begin(115200);
  //while (!Serial) ; // Wait for Serial Monitor/Plotter to open for Processors with Native USB
 (i.e. SAMD51)


#if DEBUG
  Serial.println(F("Initializing Combined Example w/ SGP40, SHTC3, and STC31."));
#else
  Serial.println(F("RH,degF,degC,SHTC3_Valid,RH_Compensate_Valid,degC_Compensate_Valid,CO2%,STC3
1_Valid,VOC_Index"));
#endif

  Wire.begin();

  //mySTC3x.enableDebugging(); // Uncomment this line to get helpful debug messages on Serial
  //mySGP40.enableDebugging(); // Uncomment this line to print useful debug messages to Serial



  if (mySHTC3.begin() != SHTC3_Status_Nominal)
  {
#if DEBUG
    Serial.println(F("SHTC3 not detected. Please check wiring. Freezing..."));
#endif
    while (1)
      ; // Do nothing more
  }

  if (mySTC3x.begin() == false)
  {
#if DEBUG
    Serial.println(F("STC3x not detected. Please check wiring. Freezing..."));
#endif
```

```
    while (1)
      ; // Do nothing more
  }

  if (mySGP40.begin() == false)
  {
#if DEBUG
    Serial.println(F("SGP40 not detected. Check connections. Freezing..."));
#endif
    while (1)
      ; // Do nothing more
  }



  //We need to tell the STC3x what binary gas and full range we are using
  //Possible values are:
  //  STC3X_BINARY_GAS_CO2_N2_100   : Set binary gas to CO2 in N2.  Range: 0 to 100 vol%
  //  STC3X_BINARY_GAS_CO2_AIR_100  : Set binary gas to CO2 in Air. Range: 0 to 100 vol%
  //  STC3X_BINARY_GAS_CO2_N2_25    : Set binary gas to CO2 in N2.  Range: 0 to 25 vol%
  //  STC3X_BINARY_GAS_CO2_AIR_25   : Set binary gas to CO2 in Air. Range: 0 to 25 vol%
  if (mySTC3x.setBinaryGas(STC3X_BINARY_GAS_CO2_AIR_25) == false)
  {
#if DEBUG
    Serial.println(F("Could not set the binary gas! Freezing..."));
#endif
    while (1)
      ; // Do nothing more
  }



  //We can compensate for temperature and relative humidity using the readings from the SHTC3

  if (mySHTC3.update() != SHTC3_Status_Nominal) // Request a measurement
  {
#if DEBUG
    Serial.println(F("Could not read the RH and T from the SHTC3! Freezing..."));
#endif
    while (1)
      ; // Do nothing more
  }

  //In case the ‘Set temperature command’ has been used prior to the measurement command,
  //the temperature value given out by the STC31 will be that one of the ‘Set temperature comman
d’.
  //When the ‘Set temperature command’ has not been used, the internal temperature value can be
 read out.
  temperature = mySHTC3.toDegC(); // "toDegC" returns the temperature as a floating point number
in deg C
#if DEBUG
  Serial.print(F("Setting STC3x temperature to "));
  Serial.print(temperature, 2);
  Serial.print(",");
```

```
  Serial.print(F("C was "));
#endif

  if (mySTC3x.setTemperature(temperature) == false) {
#if DEBUG
    Serial.print(F("not "));
#endif
  }
#if DEBUG
  Serial.println(F("successful"));
#endif

  RH = mySHTC3.toPercent(); // "toPercent" returns the percent humidity as a floating point numb
er

#if DEBUG
  Serial.print(F("Setting STC3x RH to "));
  Serial.print(RH, 2);
  Serial.print(",");
  Serial.print(F("% was "));
#endif

  if (mySTC3x.setRelativeHumidity(RH) == false) {
#if DEBUG
    Serial.print(F("not "));
#endif
  }
#if DEBUG
  Serial.println(F("successful"));
#endif



  //If we have a pressure sensor available, we can compensate for ambient pressure too.
  //As an example, let's set the pressure to 840 mbar (== SF Headquarters)
  uint16_t pressure = 840;

#if DEBUG
  Serial.print(F("Setting STC3x pressure to "));
  Serial.print(pressure);
  Serial.print(F("mbar was "));
#endif

  if (mySTC3x.setPressure(pressure) == false) {
#if DEBUG
    Serial.print(F("not "));
#endif
  }
#if DEBUG
  Serial.println(F("successful"));

  Serial.println(F("Note: Relative humidity and temperature compensation for the STC31 will be u
pdated frequently in the main loop() function."));
#endif
```

```
} //end setup()




void loop() {


  //==============================
  //======DEBUG TURNED ON=========
  //==============================
#if DEBUG
  //==============================
  //==========READ SHTC3==========
  //==============================
  //minimum update rate = ~100Hz

  SHTC3_Status_TypeDef result = mySHTC3.update();           // Call "update()" to command a meas
urement, wait for measurement to complete, and update the RH and T members of the object

  RH = mySHTC3.toPercent();                                 // "toPercent" returns the percent h
umidity as a floating point number
  Serial.print(F("RH = "));
  Serial.print(RH);

  Serial.print(F("%, T = "));
  Serial.print(mySHTC3.toDegF());                           // "toDegF" return the temperature a
s a flaoting point number in deg F
  Serial.print(F(" deg F, "));

  temperature = mySHTC3.toDegC();                           // "toDegC" returns the temperature
 as a floating point number in deg C
  Serial.print(temperature);
  Serial.print(F(" deg C"));

  if (mySHTC3.lastStatus == SHTC3_Status_Nominal)           // You can also assess the status of
the last command by checking the ".lastStatus" member of the object
  {
    Serial.println("");                                         //Sample data good, no need to o
utput a message
  }
  else {
    Serial.print(F(",     Update failed, error: "));        //notify user if there is an error
    errorDecoder(mySHTC3.lastStatus);
    Serial.println("");
  }



  //==============================
  //==========READ STC31==========
  //==============================
```

```
  //minimum update rate = 1Hz


  if (mySTC3x.setRelativeHumidity(RH) == false)
    Serial.print(F("Unable to set STC31 Relative Humidity with SHTC3."));

  if (mySTC3x.setTemperature(temperature) == false)
    Serial.println(F("Unable to set STC31 Temperature with SHTC3."));


  Serial.print(F("CO2(%): "));

  if (mySTC3x.measureGasConcentration())                  // measureGasConcentration will retur
n true when fresh data is available
  {
    Serial.println(mySTC3x.getCO2(), 2);
  }
  else
  {
    Serial.print(mySTC3x.getCO2(), 2);
    Serial.println(F(",     (old STC3 sample reading, STC31 was not able to get fresh data yet)"
));  //output this note to indicate  when we are not able to obtain a new measurement
  }



  //==============================
  //==========READ SGP40==========
  //==============================
  //minimum update rate = 1Hz

  Serial.print(F("VOC Index is: "));
  Serial.println(mySGP40.getVOCindex()); //Get the VOC Index using the default RH (50%) and T (2
5C)






  //==============================
  //=====DEBUG TURNED OFF=========
  //==============================
#else
  //==============================
  //==========READ SHTC3==========
  //==============================
  //minimum update rate = ~100Hz

  SHTC3_Status_TypeDef result = mySHTC3.update();           // Call "update()" to command a meas
urement, wait for measurement to complete, and update the RH and T members of the object

  RH = mySHTC3.toPercent();
  Serial.print(RH);
  Serial.print(",");
```

```
  Serial.print(mySHTC3.toDegF());
  Serial.print(",");
  temperature = mySHTC3.toDegC();                           // "toDegC" returns the temperature
 as a floating point number in deg C
  Serial.print(temperature);
  Serial.print(",");

  if (mySHTC3.lastStatus == SHTC3_Status_Nominal)           // You can also assess the status of
the last command by checking the ".lastStatus" member of the object
  {
    Serial.print("1");                                       //Sample data good, no need to ou
tput a message
    Serial.print(",");
  }
  else
  {
    Serial.print("0");                                       //Sample data bad, no need to out
put a message
    Serial.print(",");
  }



  //==============================
  //==========READ STC31==========
  //==============================
  //minimum update rate = 1Hz


  if (mySTC3x.setRelativeHumidity(RH) == false)
  {
    //Serial.print(F("Unable to set STC31 Relative Humidity with SHTC3."));
    Serial.print("0");
    Serial.print(",");
  }
  else
  {
    Serial.print("1");
    Serial.print(",");
  }

  if (mySTC3x.setTemperature(temperature) == false)
  {
    //Serial.println(F("Unable to set STC31 Temperature with SHTC3."));
    Serial.print("0");
    Serial.print(",");
  }
  else
  {
    Serial.print("1");
    Serial.print(",");
  }

  if (mySTC3x.measureGasConcentration())                    // measureGasConcentration will retur
```

```
n true when fresh data is available
  {
    Serial.print(mySTC3x.getCO2(), 2);
    Serial.print(",");
    Serial.print("1");                                        //Fresh Data
    Serial.print(",");
  }
  else
  {
    Serial.print(mySTC3x.getCO2(), 2);
    Serial.print(",");
    Serial.print("0");                                        //Data not fresh
    Serial.print(",");
  }



  //==============================
  //==========READ SGP40==========
  //==============================
  //minimum update rate = 1Hz

  Serial.println(mySGP40.getVOCindex()); //Get the VOC Index using the default RH (50%) and T (2
5C)

#endif



  //================================
  //=========SPACE & DELAY==========
  //================================
  //Serial.println("");// Uncomment this line to add some space between readings for the Serial
 Monitor
  delay(1000); //Wait 1 second - the Sensirion VOC algorithm expects a sample rate of 1Hz

}//end loop()




void errorDecoder(SHTC3_Status_TypeDef message)                             // The errorDecoder
 function prints "SHTC3_Status_TypeDef" resultsin a human-friendly way
{
  switch (message)
  {
    case SHTC3_Status_Nominal : Serial.print("Nominal"); break;
    case SHTC3_Status_Error : Serial.print("Error"); break;
    case SHTC3_Status_CRC_Fail : Serial.print("CRC Fail"); break;
    default : Serial.print("Unknown return code"); break;
  }
}
```

Troubleshooting

Not working as expected and need help?

If you need technical assistance and more information on a product that is not working as you expected, we recommend heading on over to the SparkFun Technical Assistance page for some initial troubleshooting.

SPARKFUN TECHNICAL ASSISTANCE PAGE

If you don't find what you need there, the SparkFun Forums: MicroMod are a great place to find and ask for help. If this is your first visit, you'll need to create a Forum Account to search product forums and post questions.

SPARKFUN FORUMS: MICROMOD

Resources and Going Further

Now that you've successfully got your MicroMod Environmental Function Board up and running, it's time to incorporate it into your own project! For more information, check out the resources below:

- Schematic (PDF)
- Eagle Files (ZIP)
- Board Dimensions (PNG)
- SGP40
    - SGP40 Datasheet (PDF)
    - VOC Index for Experts (PDF)
    - SGP40 Design In Guide (PDF)
    - SGP40 Quick Testing Guide (PDF)
- SHTC3
    - SHTC3 Datasheet (PDF)
- STC31
    - STC31 Datasheet (PDF)
    - STC Field Calibration Guide (PDF)
    - STC Design-In Guide (PDF)
- Arduino Libraries
    - SGP40
    - SHTC3
    - STC3X
- GitHub Hardware Repo
- SFE Product Showcase

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

Getting Started with MicroMod

Dive into the world of MicroMod - a compact interface to connect a microcontroller to various peripherals via the M.2 Connector!

MicroMod Artemis Processor Board Hookup Guide

Get started with the Artemis MicroMod Processor Board in this tutorial!

New!

MicroMod WiFi Function Board - ESP32 Hookup Guide

The MicroMod ESP32 Function Board adds additional wireless options to MicroMod Processor Boards that do not have that capability. This special function board acts as a coprocessor that takes advantage of Espressif's ESP32 WROOM to add WiFi and Bluetooth® to your applications.

New!

MicroMod WiFi Function Board - DA16200 Hookup Guide

Add IoT functionality to any MicroMod project with the MicroMod WiFi function Board - DA16200!