ZMOD4410 - Indoor Air Quality Sensor Platform

1. Introduction

The ZMOD4410 Gas Sensor Module is highly configurable to meet various application needs. This document describes the general program flow to set up ZMOD4410 Gas Sensor Modules for gas measurements in a customer environment. It also describes the function of example code provided as C code, which can be executed using the ZMOD4410 evaluation kit (EVK), Arduino®, and Raspberry Pi® hardware.

The corresponding firmware package is provided on the Renesas ZMOD4410 product page under the Software Downloads section. For various Renesas microcontrollers, ready-to-use code (ZMOD4xxx Sample application) is provided on the Sensor Software Modules for Renesas MCU Platforms product page.

For instructions on assembly, connection, and installation of the EVK hardware and software, see the document titled *Environmental Sensors Evaluation Kit Manual* on the ZMOD4410 EVK product page.

The ZMOD4410 has several modes of operation:

- IAQ 2nd Gen – The embedded artificial intelligence (AI) algorithm (“iaq_2nd_gen”) derived from machine learning outputs total volatile organic compounds (TVOC), equivalent ethanol (EtOH) concentration, estimated carbon dioxide level (eCO2), and an absolute and relative rating for the indoor air quality (IAQ). This operation mode is for highly accurate and consistent sensor readings. **This is the recommended operation mode for IAQ**.
- IAQ 2nd Gen Ultra-Low Power (ULP) – The embedded artificial intelligence (AI) algorithm (“iaq_2nd_gen_ulp”) derived from machine learning outputs total volatile organic compounds (TVOC), equivalent ethanol (EtOH) concentration, estimated carbon dioxide level (eCO2), and an absolute and relative rating for the indoor air quality (IAQ). This operation mode offers a much lower power consumption while keeping accurate and consistent sensor readings.
- Public Building AQ Standard (PBAQ) – The embedded artificial intelligence (AI) algorithm derived from machine learning outputs total volatile organic compounds (TVOC), equivalent ethanol (EtOH) concentration, and an absolute and relative rating for the indoor air quality (IAQ). This operation mode is for highly accurate and consistent sensor readings to fulfill several public building standards.
- Sulfur Odor – This semi-selective detection method for gas species allows discrimination between sulfur odors in the air. Odors are classified as “Acceptable” and “Sulfur” with an intensity level. For description of the firmware of this operation mode, see the *ZMOD4410 Programming Manual – Read Me* included in this Firmware package.

The previous Odor Firmware and output was replaced by the Relative IAQ output included in the other firmwares.

Note: This document is generic and valid for several firmware examples. Whenever the placeholder ”XXX” is used, replace the name (*iaq_2nd_gen, iaq_2nd_gen_ulp, pbaq*) corresponding to the chosen package.

Recommendation: Before using this document, read the *ZMOD4410 Datasheet* and corresponding documentation on the ZMOD4410 product page.

Contents

Figures

Tables

2. Hardware Requirements to Operate ZMOD4410

To operate the ZMOD4410, customer-specific hardware with a microcontroller unit (MCU) is needed. Depending on the sensor configuration and the hardware itself, the requirements differ. The following minimum requirements are provided as an orientation only:

- 20 to 40kB program flash for ZMOD4410-related firmware code (MCU architecture and compiler dependent), see Table 1.
- 1.5kB RAM for ZMOD4410-related operations (see Table 1).
- Capability to perform I²C communication, timing functions (5% precision), and floating-point instructions.
- The algorithm functions work with variables saved in background and require memory retention between each call.

Table 1. Exemplary Memory Footprint of ZMOD4410 Implementation on a Renesas RL78-G13 MCU [1]

	IAQ 2nd Gen	IAQ 2nd Gen ULP	PBAQ
Program flash usage in kB	19.6	17.2	18.2
RAM usage (required variables) in bytes	726	448	702
RAM usage (stack size for library functions, worst case) in bytes	560	256	576

1. This example does not contain hardware-specific I²C and delay functions. CCRL compiler used.

The ZMOD4410 firmware can be downloaded from the ZMOD4410 product page. To get access to the firmware, a Software License Agreement (SLA) has to be accepted. The firmware uses floating-point calculations with various integer and floating-point variables. A part of the firmware are precompiled libraries for many standard targets (microcontrollers), as listed in the following table.

Table 2. Targets and Compilers Supported by Default (Cont. on Next Page)

Target	Compiler
Arduino (Cortex-M0+)	arm-none-eabi-gcc (Arduino IDE)
Arm Cortex-A	arm-none-eabi-gcc (all others)
	iar-ew-arm (IAR Embedded Workbench)
	aarch64-linux-gnu-gcc
Arm Cortex-M	armcc (Keil MDK with Keil Legacy Arm Compiler 5 settings)
	armclang (Arm Developer Studio, Keil MDK)
	arm-none-eabi-gcc (all others)
	iar-ew-arm (IAR Embedded Workbench)
	iar-ew-synergy-arm (IAR Embedded Workbench)
Arm Cortex-R4	arm-none-eabi-gcc (all others)
	iar-ew-arm (IAR Embedded Workbench)
Arm Linux	arm-linux-gcc
Espressif ESP	xtensa-esp32-elf-gcc
	xtensa-esp32s2-elf-gcc
	xtensa-esp32s3-elf-gcc

Target	Compiler
	xtensa-lx106-elf-gcc
	riscv32-esp-elf-gcc
Intel 8051	iar-ew-8051 (IAR Embedded Workbench)
Microchip ATmega32 and AVR	avr-gcc (AVR-Studio, AVR-Eclipse, MPLAB, Atmel Studio)
Microchip PIC	xc8-cc (MPLAB)
	xc16-gcc (MPLAB)
	xc32-gcc (MPLAB)
Raspberry Pi	arm-linux-gnueabihf-gcc
	aarch64-linux-gnu-gcc
Renesas RL78	ccrl (e²studio, CS+)
	iar-ew-rl (IAR Embedded Workbench)
	rl78-elf-gcc
Renesas RX	ccrx (e²studio, CS+)
	iar-ew-rx (IAR Embedded Workbench)
	rx-elf-gcc
Texas Instruments MSP430	msp430-elf-gcc
Windows	mingw32
	mingw64

Note: For other platforms (e.g., other Linux platforms) and other Arduino boards, contact Renesas Technical Support.

3. Structure of ZMOD4410 Firmware

To operate the ZMOD4410 and use its full functionality, the following five code blocks are required (see Figure 1):

1. The “Target Specific I²C and Low-Level Functions” block is the hardware-specific implementation of the I²C interface. This block contains read and write functions to communicate with the ZMOD4410 and a delay function. For the use with Renesas EVKs, there are implementations for ESCom and HiCom boards provided with the ZMOD4410 firmware packages. Using the user’s own target hardware requires implementing the user’s target-specific I²C and low-level functions (this is highlighted in light blue in Figure 1).
2. The “Hardware Abstraction Layer (HAL)” block contains hardware-specific initialization and de-initialization functions. For the use with Renesas EVKs the Communication Board HAL is provided, which supports ESCom and HiCom boards. The HAL is described in the document *ZMOD4410-XXX-Documentation.pdf/.html*, which is included in the firmware packages.
3. The “Application Programming Interface (API)” block contains the functions needed to operate the ZMOD4410. The API should not be modified! A detailed description of the API is located in the document *ZMOD4410-XXX-Firmware-Documentation.pdf/.html*, which is included in the firmware packages.
4. The example.c file in the “Programming Example” block demonstrates the sensor initialization, sensor cleaning, raw sensor data readout, and algorithm results computation. For more information, see “Description of the Programming Examples”.
5. The “Gas Measurement Libraries” block contains one configuration file (*zmod4410_config_XXX.h*) that should not be modified! Furthermore, a library needed to calculate the firmware-specific results for the Indoor Air Quality related parameters is provided, such as IAQ, TVOC, EtOH, rel IAQ and eCO2 (the latter only for IAQ 2nd Gen and IAQ 2nd Gen ULP). The algorithms cannot be used together because the gas sensor must be configured differently for each algorithm. This block also contains the cleaning library. The libraries are described in more detail in the *ZMOD4410-XXX-Firmware-Documentation.pdf/.html*.

To avoid naming conflicts, all API function names start with the prefix “zmod4xxx” in the ZMOD4410 code. The Arduino and Raspberry Pi examples have a similar structure but have some other features that facilitate operation with the corresponding hardware (see “Arduino Examples“ and “Raspberry Pi Example”).

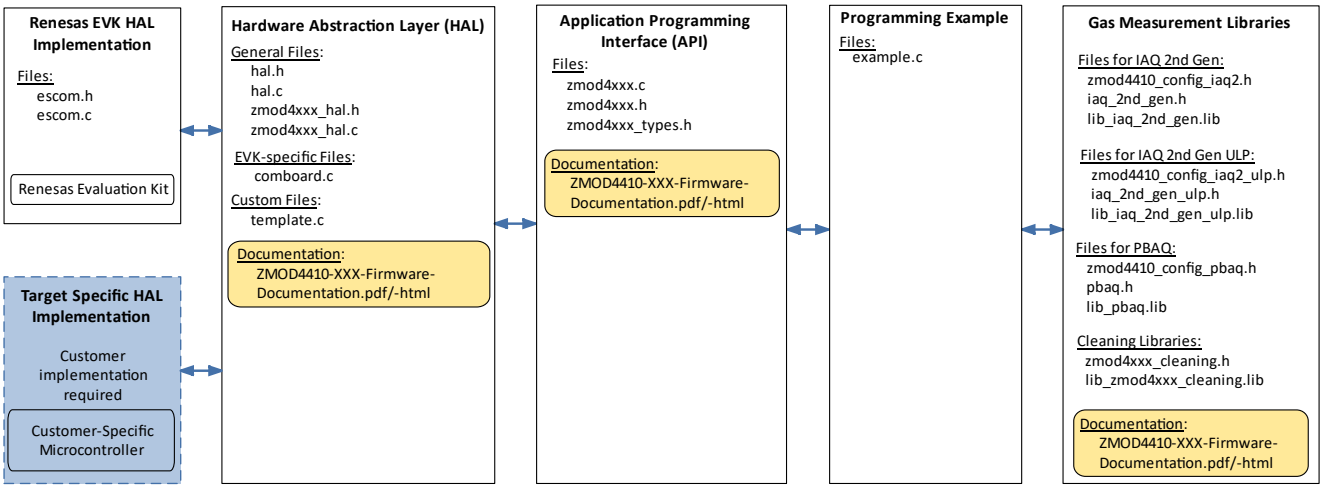


Figure 1. File Overview for ZMOD4410 Firmware

All files are part of a zipped firmware packages available on the ZMOD4410 product page under the Software Downloads section. Note that not all configurations and libraries are available for all operation modes; the individual library documentation will provide detailed insight on possible settings.

4. Description of the Programming Examples

This section describes the structure of the firmware examples and the steps needed to operate the sensor module. In the examples, the ZMOD4410 is initialized, the measurement is started, and measured values are outputted. They are intended to work on a Windows® computer in combination with the Renesas Environmental Sensor-EVK, however it can be easily adjusted to operate on other platforms (see “Adapting the Firmware Example for Target Hardware”). To run each example using the EVK without further configuration, start the file *XXX_example.exe,* which is included in the firmware packages. Examples for Arduino and Raspberry Pi hardware are also introduced (see “Arduino Examples“ and “Raspberry Pi Example”).

Note: Running an executable with legacy EVKs (HiCom) requires an FTDI driver to be available on the host computer.

4.1 IAQ 2nd Gen Example for EVK

The *example.c* file of the example contains the main program flow. First, the target-specific initializations are performed. The ZMOD4410 is configured by reading device parameters as well as Final Module Test parameters from the sensor’s non-volatile memory (NVM), and then initializing it. A measurement loop continuously checks the status and errors of the ZMOD4410 and reads its data. The raw data is subsequently processed. The TVOC, EtOH, IAQ, rel IAQ, and eCO2 algorithm results are calculated with the embedded neural net machine learning algorithm. All values are printed in the command line window. To stop the loop, press Ctrl+C, which releases the hardware and stops the program. For more information, refer to the example code. Table 3 shows the necessary function calls to operate the ZMOD4410. In Table 4, the combined use of the ZMOD4410 and ZMOD4510 is shown for compensation of oxidizing gases. This flow corresponds to the example provided in the firmware package.

Note: The blue colored lines in the following table can be run in an endless loop.

Table 3. IAQ 2nd Gen Program Flow using ZMOD4410

Line	Program Actions	Notes	Function
1	Reset the sensor.	Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin.	-
2	Detect and initialize the ZMOD4410 sensor.	This function must be used after each start-up.	detect_and_configure
3	Initialize the IAQ (TVOC, EtOH, rel IAQ, eCO2) algorithm.	Gas Algorithm Library function.	init_iaq_2nd_gen
4	Start the ZMOD4410 measurement.	One measurement is started every 3 seconds and takes 1010ms.	zmod4xxx_start_measurement
5	Delay (3000ms).	This delay is necessary to keep the right measurement timing and to call a measurement every 3 seconds with a maximum deviation of 5% to keep the algorithm accuracy.	-
6	Read the ZMOD4410 measurement.	Read every 3 seconds.	read_and_verify
7	Algorithm calculation and sensor self-check.	Calculate current MOx resistance Rmox, clean dry air resistance Rcda, IAQ, TVOC, EtOH, rel IAQ and eCO2. Relative humidity (in % RH) and temperature values (in °C) and ADC results need to be passed as algo_input. First 300 samples (15 minutes) are used for minimal, hard-coded sensor warm-up and output is frozen. Actual warm-up can take longer (up to 48 hours).	calc_iaq_2nd_gen

Table 4. IAQ 2nd Gen Program Flow using ZMOD4410 and ZMOD4510

Line	Program Actions	Notes	Function
1	Reset the sensor.	Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin.	-
2	Detect and initialize the ZMOD4410 sensor.	This function must be used after each start-up.	detect_and_configure
3	Detect and initialize the ZMOD4510 sensor.	This function must be used after each start-up.	detect_and_configure
4	Initialize the IAQ (TVOC, EtOH, rel IAQ, eCO2) algorithm.	Gas Algorithm Library function.	init_iaq_2nd_gen
5	Start the ZMOD4510 measurement.	One measurement is started for compensation every 6 seconds and takes 4020ms.	zmod4xxx_start_measurement
	Start the ZMOD4410 measurement.	One measurement is started every 3 seconds and takes 1010ms.	zmod4xxx_start_measurement
6	Delay ().	This delay is necessary to keep the right measurement timing and to call this loop every 3 seconds with a maximum deviation of 5% to keep the algorithm accuracy.	-
7	Read the ZMOD4510 measurement.	Read every 6 seconds for compensation.	read_and_verify
	Read the ZMOD4410 measurement.	Read every 3 seconds after the measurement finished.	read_and_verify
8	Algorithm calculation and sensor self-check.	If measurement was read, calculate current MOx resistance Rmox, clean dry air resistance Rcda, IAQ, TVOC, EtOH, rel IAQ and eCO2. adc_rmox3_4510 value, relative humidity (in % RH) and temperature values (in °C) and ADC results need to be passed as algo_input. First 300 samples (15 minutes) are used for minimal, hard-coded sensor warm-up and output is frozen. Actual warm-up can take longer (up to 48 hours).	calc_iaq_2nd_gen

4.2 ULP Example for EVK (IAQ 2nd Gen ULP)

The *example.c* file of the example contains the main program flow. First, the target-specific initializations are performed. The ZMOD4410 is configured by reading device parameters as well as Final Module Test parameters from the sensor's non-volatile memory (NVM) and initializing it. An endless measurement loop continuously checks the status of the ZMOD4410 and reads its data. The raw data is subsequently processed. The TVOC, EtOH, IAQ, rel IAQ, and eCO2 algorithm results are calculated with the embedded neural net machine learning algorithm. All values are printed in the command line window. To stop the loop, press Ctrl+C, which releases the hardware and stops the program. For more information, refer to the example code.

Note: The blue colored lines in the following table can be run in an endless loop.

Table 5. ULP Program Flow (IAQ 2nd Gen ULP)

Line	Program Actions	Notes	Function
1	Reset the sensor.	Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin.	-
2	Detect and initialize the ZMOD4410 sensor.	This function must be used after each start-up.	detect_and_configure
3	Initialize the IAQ (TVOC, EtOH, eCO2) algorithm.	Gas Algorithm Library function.	init_iaq_2nd_gen_ulp
4	Start the measurement.	One measurement is started.	zmod4xxx_start_measurement
5	Delay ().	Wait until the measurement is done. This is the first delay. It should be longer than 1010 ms.	-
6	Read the ZMOD4410 measurement.	Read and verify the validity of the results.	read_and_verify
7	Algorithm calculation and sensor self-check.	Calculate current MOx resistance Rmox, clean dry air resistance Rcda, IAQ, TVOC, EtOH, rel IAQ and eCO2. Relative humidity (in % RH) and temperature values (in °C) need to be passed as arguments. First 10 samples (15 minutes) are used for minimal, hard-coded sensor warm-up. Actual warm-up can take longer (up to 48 hours).	calc_iaq_2nd_gen_ulp
8	Delay ().	This second delay is necessary to keep the right measurement timing. The sum of the first and second delay should amount 90 seconds to call a measurement every 90 seconds with a maximum deviation of 5% to keep the algorithm accuracy.	-

4.3 PBAQ Example for EVK

The *example.c* file of the example contains the main program flow. First, the target-specific initializations are performed. The ZMOD4410 is configured by reading device parameters as well as Final Module Test parameters from the sensor's non-volatile memory (NVM), and then initializing it. A measurement loop continuously checks the status of the ZMOD4410 and reads its data. The raw data is subsequently processed. The TVOC, EtOH, IAQ, and rel IAQ algorithm results are calculated with the embedded neural net machine learning algorithm. All values are printed in the command line window. To stop the loop, press Ctrl+C, which releases the hardware and stops the program. For more information, see the example code. Table 6 shows the necessary function calls to operate the ZMOD4410. In Table 7, the combined use of the ZMOD4410 and ZMOD4510 is shown for compensation of oxidizing gases. This flow corresponds to the example provided in the firmware package.

Note: The blue colored lines in the following table can be run in an endless loop.

Table 6. PBAQ Program Flow using ZMOD4410

Line	Program Actions	Notes	Function
1	Reset the sensor.	Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin.	-
2	Detect and initialize the ZMOD4410 sensor.	This function must be used after each start-up.	detect_and_configure
3	Initialize the PBAQ (TVOC, EtOH, IAQ, rel IAQ) algorithm.	Gas Algorithm Library function.	init_pbaq
4	Start the ZMOD4410 measurement.	One measurement is started every 5 seconds and takes 256ms.	zmod4xxx_start_measurement
5	Delay ().	This delay is necessary to keep the right measurement timing and to call a measurement every 5 seconds with a maximum deviation of 5% to keep the algorithm accuracy.	-
6	Read the ZMOD4410 measurement.	Read every 5 seconds.	read_and_verify
7	Algorithm calculation and sensor self-check.	Calculate current MOx resistance Rmox, clean dry air resistance Rcda, IAQ, TVOC, EtOH, and rel IAQ. Relative humidity (in % RH) and temperature values (in °C) and ADC results need to be passed as algo_input. First 300 samples (25 minutes) are used for minimal, hard-coded sensor warm-up and output is frozen. Actual warm-up can take longer (up to 48 hours).	calc_pbaq

Table 7. PBAQ Program Flow using ZMOD4410 and ZMOD4510 (Cont. on Next Page)

Line	Program Actions	Notes	Function
1	Reset the sensor.	Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin.	-
2	Detect and initialize the ZMOD4410 sensor.	This function must be used after each start-up.	detect_and_configure

Line	Program Actions	Notes	Function
3	Detect and initialize the ZMOD4510 sensor.	This function must be used after each start-up.	detect_and_configure
4	Initialize the PBAQ (TVOC, EtOH, IAQ, rel IAQ) algorithm.	Gas Algorithm Library function.	init_pbaq
5	Start the ZMOD4510 measurement.	One measurement is started for compensation every 6 seconds and takes 4020ms.	zmod4xxx_start_measurement
	Read the ZMOD4510 measurement.	Read 5 seconds after measurement start.	read_and_verify
6	Start the ZMOD4410 measurement.	One measurement is started every 5 seconds and takes 256ms.	zmod4xxx_start_measurement
	Read the ZMOD4410 measurement.	Read 4 seconds after measurement start.	read_and_verify
7	Algorithm calculation and sensor self-check.	If measurement was read, calculate current MOx resistance Rmox, clean dry air resistance Rcda, IAQ, TVOC, EtOH, and rel IAQ. adc_rmox3_4510 value, relative humidity (in % RH) and temperature values (in °C) and ADC results need to be passed as algo_input. First 300 samples (25 minutes) are used for minimal, hard-coded sensor warm-up and output (IAQ, TVOC, EtOH) is frozen. Actual warm-up can take longer (up to 48 hours).	calc_pbaq
8	Delay ().	This delay is necessary to keep the right measurement timing and to call this loop every 1 second with a maximum deviation of 5% to keep the algorithm accuracy.	-

4.4 Compile for EVK Hardware

The EVK firmware example is designed to work with the EVK hardware. To evaluate the impact of code changes on sensor performance, it is possible to use the EVK as reference. This section describes how to compile the adapted source code into an executable file, which can then be used with the EVK on a Windows platform. The firmware folder structure should be identical to that in the download package.

To compile for 64-bit Windows, “skeeto/w64devkit” must be downloaded and unzipped:

1. Install skeeto/w64devkit:
   a. Download the latest version of w64devkit-<VERSION>.zip and unzip it.
2. Compiling:
   a. Go to the Command Prompt and add skeeto/w64devkit to the system path in command line by using:
      `set PATH=<PATH_TO_W64DEVKIT>\bin;%PATH%`
   b. Change to the following directory of the firmware folder:
      [...]\Renesas-ZMOD4410-XXX-Firmware\src
   c. Execute the following command:
      `make`
   d. An executable file called *XXX-example.exe* will be created in folder *build*.

To compile for 32-bit Windows, “MinGW” must be installed:

1. Install MinGW:
   a. MinGW (32 bit) must be used.
   b. Download *mingw-get-setup.exe* from https://osdn.net/projects/mingw/releases/.
   c. The downloaded executable file installs “Install MinGW Installation Manager”.
   d. Select the required packages:
      i. mingw-developer-toolkit-bin
      ii. mingw32-base-bin
      iii. mingw32-gcc-g++-bin
      iv. msys-base-bin.
   e. Click “Installation” from the top-left corner and select “Update Catalogue”.
   f. Finish installation.
2. Compiling:
   a. Go to the Command Prompt and add MinGW to the system path in command line by using:
      `set PATH=<PATH_TO_MinGW>\bin;<PATH_TO_MinGW>\msys\1.0\bin;%PATH%`
   b. Change to the following directory of the example folder: *[...]\Renesas-ZMOD4410-XXX-Firmware\src*
   c. Execute the following command:
      `make ARCH=32 GCC=mingw32-gcc`
      An executable file called *XXX-example.exe* will be created in folder *build*.

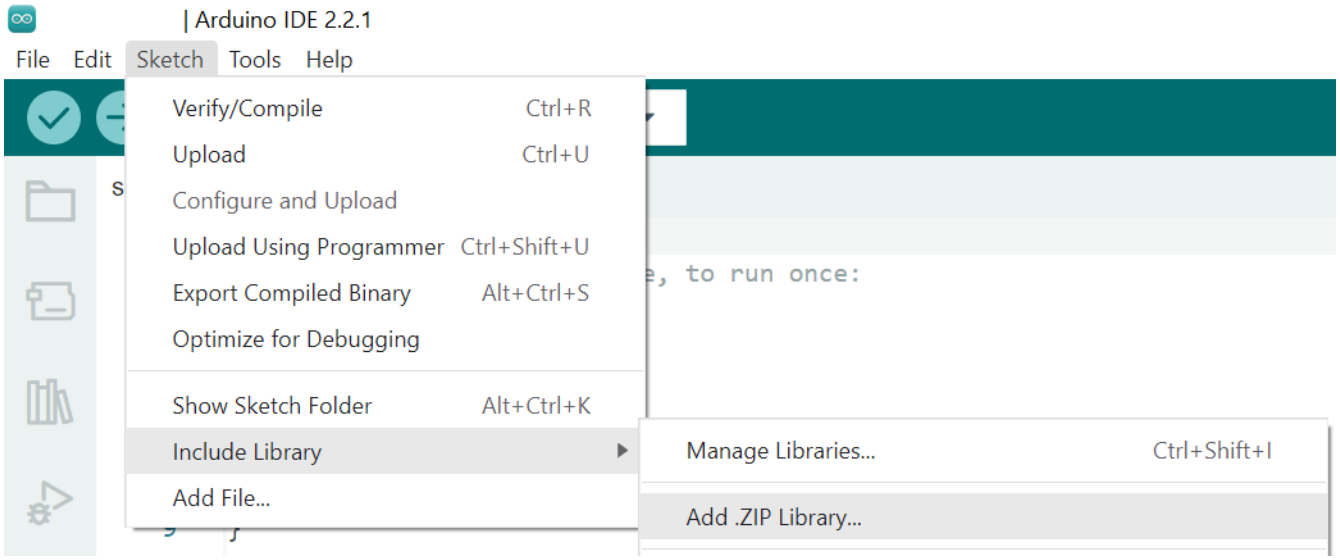
Note: Running compiled executable with legacy EVKs (HiCom) requires an FTDI driver to be available on the host computer.

4.5 Arduino Examples

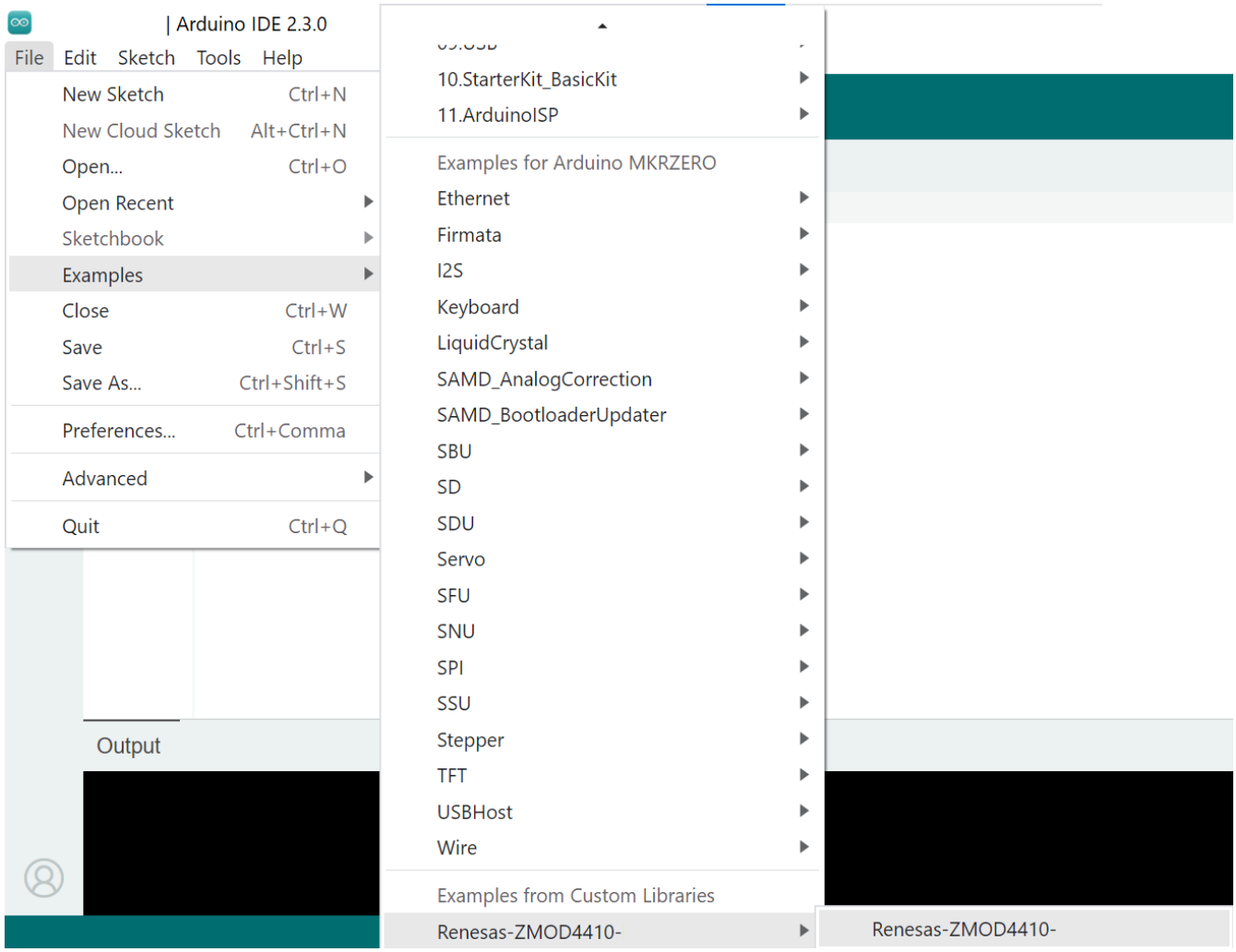
To set up firmware for an Arduino target, Renesas provides the above-mentioned EVK examples also as an Arduino example. These example are high-level Arduino libraries and have a similar structure as shown in Figure 1 but with a HAL dedicated for Arduino, an Arduino-compatible structure, and Arduino-specific files. An Arduino IDE with version 2.0.0 or higher is needed. The Program Flows correspond to those depicted in the EVK examples. To get the Arduino example started, complete the following steps (example shown for SAMD 32-bit ARM Cortex-M0+ based Arduino-hardware):

1. Connect the ZMOD4410 to the Arduino board. To connect the EVK Sensor Board, check the pin configuration on connector “X1” in the *Environmental Sensors Evaluation Kit Manual* on ZMOD4410 EVK product page.
2. Go to the Arduino example path (for example […]\Documents\Arduino\libraries) and check if a ZMOD4410 example exists. Old example folders must be deleted.

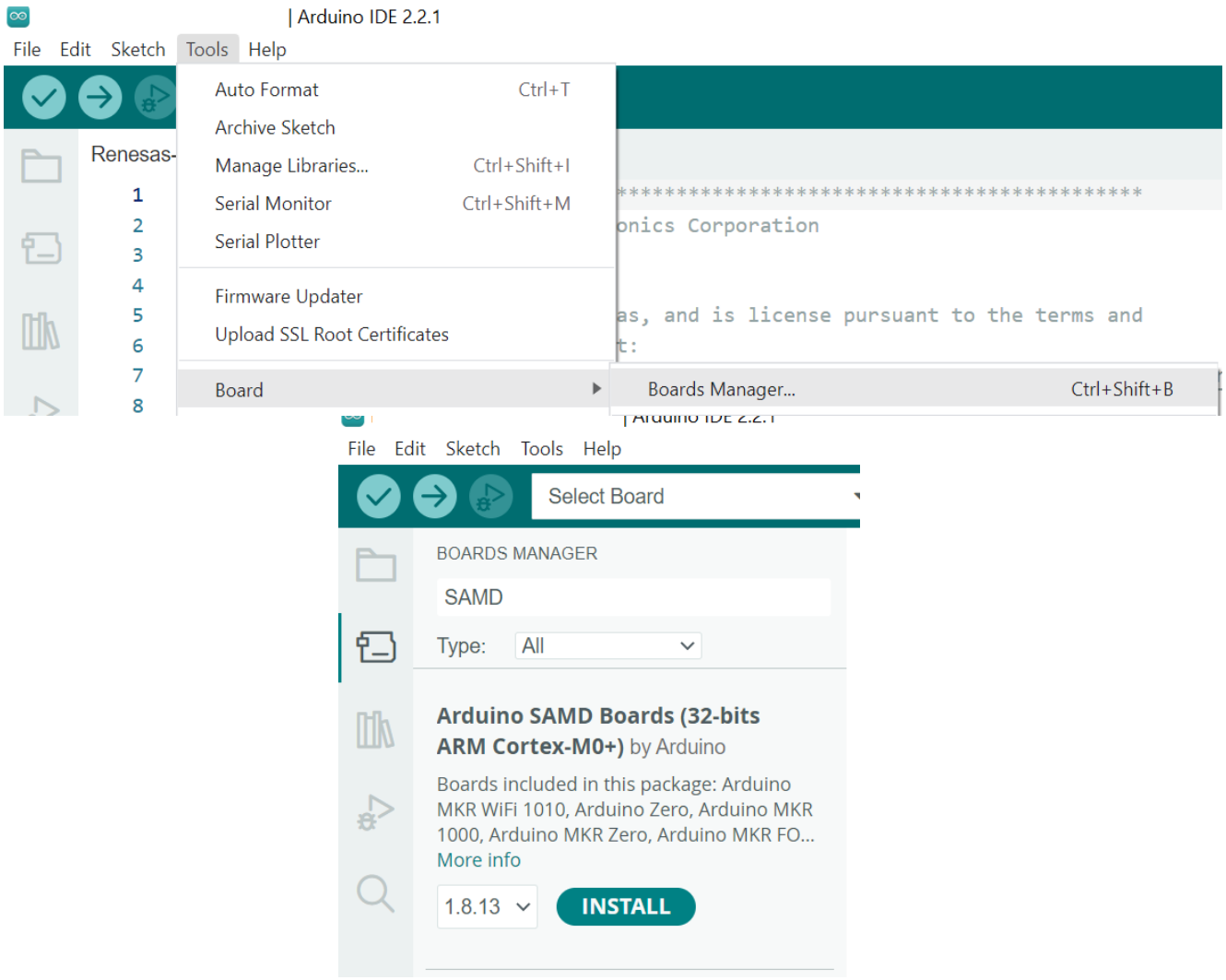
3. Open Arduino IDE. Select “Sketch > Include Library > Add .ZIP Library”.



4. Select the Renesas-ZMOD4410-XXX-Firmware-Arduino.zip file.
5. Select “File > Examples > Renesas- ZMOD4410-XXX.” A new Arduino IDE window opens automatically with examples main file.

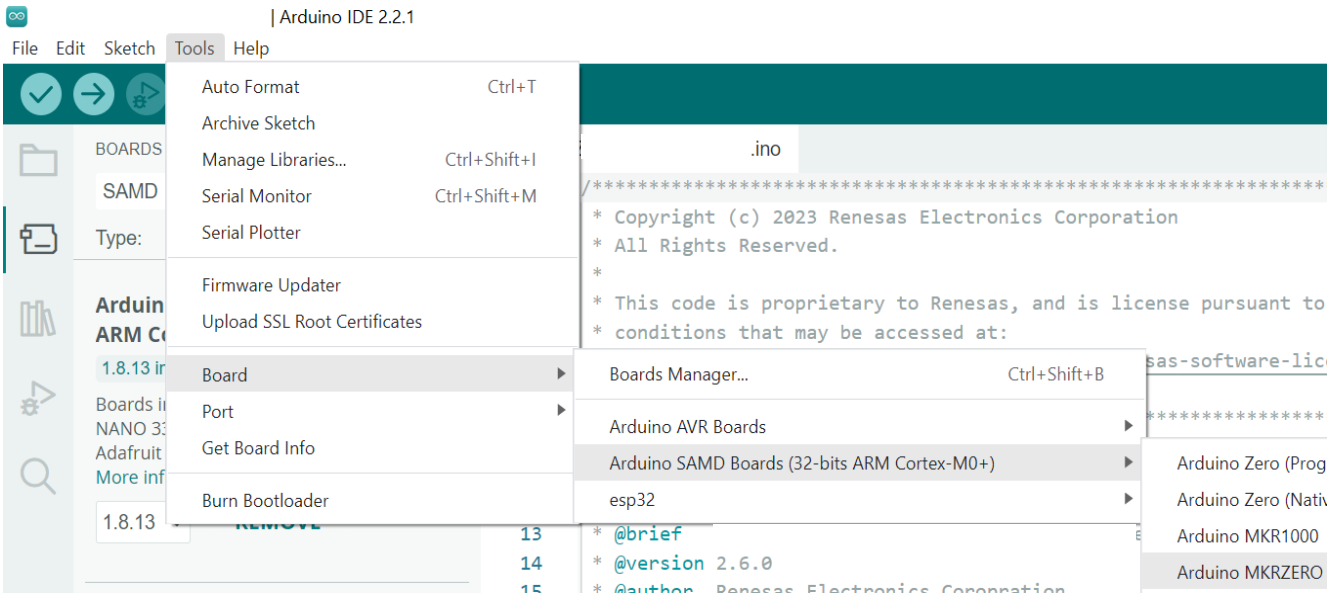


6. (This step may not be required for other Arduino hardware.) Install “Arduino SAMD (32-bit ARM Cortex-M0+)” Boards library under “Tools > Board > Board Manager”.

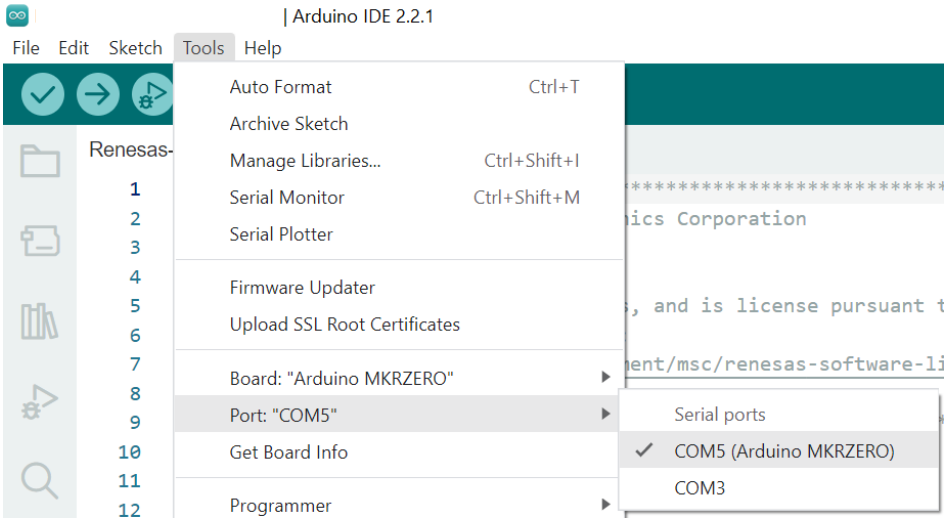


If it already exists, skip this step. Type “Arduino SAMD Boards” in search field and click the “Install” button in “Arduino SAMD (32-bits ARM Cortex-M0+)” field.

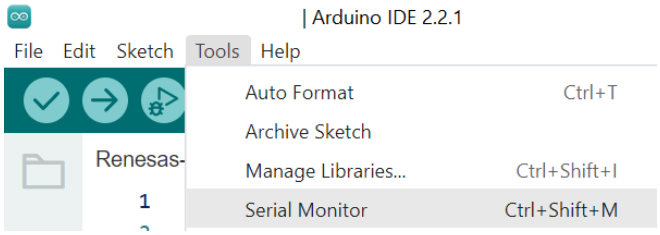
7. Select the target board under “Tools > Board > Arduino SAMD (32-bits ARM Cortex-M0+) > Arduino MKRZERO”.



8. Compile the example with the “Verify” icon.
9. Select the connected port with “Tools > Port > (Connected Port)”. The correct COM-Port should show your Arduinos board name.



10. Load the program into the target hardware with the “Upload” icon.
11. Check the results with the Serial Monitor (Tools > Serial Monitor).



Note: The sensor may execute the cleaning procedure during the first operation, which takes one minute to complete. Afterwards, the results are shown.

4.6 Raspberry Pi Examples

To set up a firmware for a Raspberry Pi based target, Renesas provides the above-mentioned EVK examples also as Raspberry Pi examples. These examples have a similar structure as shown in Figure 1 but with a HAL dedicated for Raspberry Pi and a Makefile to easily compile the code. The example is based on the pigpio library and Raspberry Pi OS (previously called Raspbian).

The example is tested on the following Raspberry Pi models on Raspberry Pi OS:

- Raspberry Pi 3 B, B+
- Raspberry Pi 4 B

The following table describes the connection of the Sensor Board connector “X1” and the Raspberry Pi GPIO Connector. Documentation of X1 can be found in the *Environmental Sensor Evaluation Kit Manual*. Documentation of Raspberry Pi GPIO can be found on command line typing “pinout” or online.

Table 8. Connection of Sensor Board to Raspberry Pi

Sensor Board Pin (X1)	Sensor Board Description	Raspberry Pi Pin	Raspberry Pi Description
1	VDD	1, 17	3V3 power
7	SDA	3	GPIO 2 (SDA)
5	SCL	5	GPIO 3 (SCL)
14	GND	6, 9, 14, 20, 25, 30, 34, 39	Ground

The Program flows correspond to those displayed in the EVK examples. To get the Raspberry Pi example started, complete the following steps:

1. Install the Raspberry Pi operating system on the Raspberry Pi. An imager tool is available to easily flash the operating system to SD card.
2. To configure the I²C interface go to */boot* directory and open the configuration file:
   `sudo nano config.txt`
3. Enable the I²C interface and change the baud rate by uncommenting the line `#dtparam=i2c_arm=on` and changing it to:
   `dtparam=i2c_arm=on,i2c_arm_baudrate=200000`
4. Reboot the Raspberry Pi to complete the initial setup. Once done, the example code can be started.
5. Copy the whole Renesas firmware package to your Raspberry Pi and extract it to your preferred location (e.g., “Downloads”).
6. Open the Terminal and go to the directory containing the executable.:
   `cd ~/Downloads/Renesas-ZMOD4410-XXX-Firmware/raspberrypi/`
7. Start the example with the following command (sudo is required for pigpio package):
   `sudo ./XXX_example`

   You may have to assign yourself execute permissions with `chmod 544 XXX-example`. If you get an error “Can't lock /var/run/pigpio.pid”, run the following command:
   `sudo killall pigpiod`.

   You can also try to establish an internet connection via Wi-Fi or LAN and install updates on the Raspberry Pi using the command `sudo apt update && sudo apt upgrade -y` on the Terminal. The updates may take some time to finish.

4.7 Compile for Raspberry Pi Hardware

This section provides guidelines for compiling the adapted source code into an executable file. This executable can be used on the Raspberry Pi like the original provided executable file. The compile method is working for the following Raspberry Pi models 3 and 4. For compiling, “make” must be installed, which is a standard package in Raspberry Pi OS. The folder structure should be identical to that in the downloaded package.

1. Complete your code changes in the source code of the firmware package.
2. Open the Terminal and go to the directory containing the example code. For example,
   `cd ~/Downloads/Renesas-ZMOD4410-XXX-Firmware/src`
3. Type `make`, and a file called *XXX-example* will be generated in the folder named *build*.
4. Start the example with the following command (sudo is required for pigpio package). Make sure to have the I²C interface enabled (for instructions, see “Raspberry Pi Examples”.
   `sudo build/XXX-example`

4.8 Error Codes

All API functions return a code to indicate the success of the operation. If no error occurred, the return code is zero. If an error occurs, a negative number is returned. The API has predefined symbols *zmod4xxx_err* for the error codes defined in *zmod4xxx_types.h*. If an error occurs, check the following table for solutions. Note that the ZMOD API cannot detect an incorrect I²C implementation. Each error may occur also with an incorrect I²C implementation.

Table 9. Error Codes (Cont. on Next Page)

Error Code	Error	Description	Solution
1	XXX_STABILIZATION	Sensor is in stabilization.	Algorithm results obtained during this period should not be considered as valid outputs. Ignore the outputs.
0	ZMOD4XXX_OK	No error.	
-1	ERROR_INIT_OUT_OF_RANGE	The initialization value is out of range.	Not used.
-2	ERROR_GAS_TIMEOUT	A previous measurement is running that could not be stopped or sensor does not respond.	1. Try to reset the sensor by powering it off/on or toggling the reset pin. Then, start the usual Program Flow as shown in “Description of the Programming Examples”. 2. Check your I²C wrapper functions for I²C read and write. It is best to analyze the voltage levels of the SDA/SCL line and check if they match the pattern described in figure “I2C Data Transmission Protocol” in the datasheet. Do a register check as requested in “I2C Interface and Data Transmission Protocol” in the datasheet. Also check multiple register write and read out.
-3	ERROR_I2C	I²C communication was not successful.	1. If available, check the error code of your parent I²C functions used in the ZMOD HAL for I2C_write/I2C_read implementation. 2. Check if the sensor correctly wired and if the voltage levels are appropriate. 3. Check your I²C wrapper functions for I²C read and write. It is best to analyze the voltage levels of the SDA/SCL line and check if they match the pattern described in figure “I2C Data Transmission Protocol” in the datasheet. Do a register check as requested in “I2C Interface and Data Transmission Protocol” in the datasheet. Also check multiple register write and read out.

Error Code	Error	Description	Solution
-4	ERROR_SENSOR_UNSUPPORTED	The Firmware configuration used does not match the sensor module.	1. Check the part number of your device. Go to the product page at www.renesas.com/zmod4410. Under the “Downloads”, you will find the right firmware for ZMOD4410. Replace it. 2. Check your I²C wrapper functions for I²C read and write. It is best to analyze the voltage levels of the SDA/SCL line and check if they match the pattern described in figure “I2C Data Transmission Protocol” in the datasheet. Do a register check as requested in “I2C Interface and Data Transmission Protocol” in the datasheet. Also check multiple register write and read out.
-5	ERROR_CONFIG_MISSING	There is no pointer to a valid configuration.	Not used.
-6	ERROR_ACCESS_CONFLICT	Invalid ADC results due to a still running measurement while results readout.	1. Check if the delay function is correctly implemented. You can use a scope plot of a GPIO pin that is switched on and off. The delay function must introduce delays in milliseconds. 2. Check measurement timing by comparing your flow with the Program Flow as shown in “Description of the Programming Examples”. Figure 3 shows graphically the correct timing. Make sure to start a result readout after active measurement phase finished. See hints on measurement timing in “Interrupt Usage and Measurement Timing”. 3. Check your I²C wrapper functions for I²C read and write. It is best to analyze the voltage levels of the SDA/SCL line and check if they match the pattern described in figure “I2C Data Transmission Protocol” in the datasheet. Do a register check as requested in “I2C Interface and Data Transmission Protocol” in the datasheet. Also check multiple register write and read out.
-7	ERROR_POR_EVENT	An unexpected reset of the sensor occurred.	Check stability of power supply and power/reset lines (e.g., for crosstalk). After a Power-On reset the sensor lost its configuration and must be reconfigured. If the host-controller did not lose its memory due to a restart, start with *zmod4xxx_prepare_sensor* function and continue in the Program flow as shown in “Description of the Programming Examples”.
-8	ERROR_CLEANING	The maximum numbers of cleaning cycles ran on this sensor. *zmod4xxx_cleaning_run* function has no effect anymore.	Using cleaning too often can harm the sensor module. The cleaning cannot be used anymore on this sensor module. Comment out the function *zmod4xxx_cleaning_run* if not needed.

-9	ERROR_NULL_PTR	The *hal* structure did not receive the pointers for I²C read, write, delay and/or reset.	The HAL_Init function contains assigning the variable of hal i2cRead, i2cWrite and msSleep function pointers. These three functions have to be generated for the corresponding hardware and assigned in the init_hardware function. This is exemplary shown in template.c:
			`hal -> msSleep = _Sleep;`
			`hal -> i2cRead = _I2CRead;`
			`hal -> i2cWrite = _I2CWrite;`
			Check if the assignment was done.

-102	XXX_DAMAGE	Sensor is probably damaged.	If an error occurs at the beginning of a sensor’s lifetime, wait for 60 minutes and check if the error disappears. For more information, see “General Characteristics” in the *ZMOD4410 Datasheet*.

5. Adapting the Firmware Example for Target Hardware

5.1 System Hierarchy and Implementation Steps

The Renesas ZMOD4410 C API is located between the application and the hardware level.

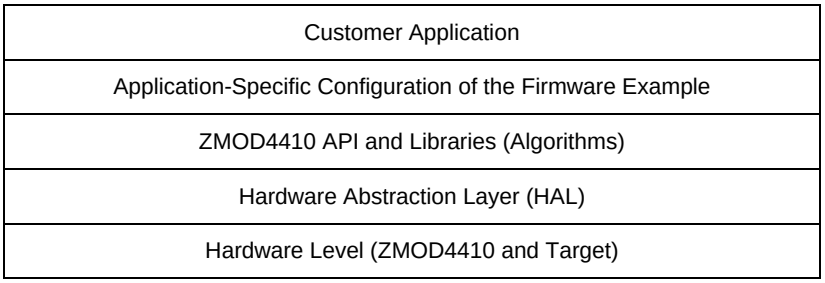


Figure 2. System Hierarchy

The ZMOD4410 example code uses a Hardware Abstraction Layer to separate target hardware implementation details from the actual sensor interface. To transfer the example to a different hardware platform, the following steps are recommended:

1. Recursively copy everything from *src* folder into your project.
2. Remove all subdirectories from *src/hal* except *custom*.
3. Adapt the file *src/hal/custom/template.c* to work with your hardware. This requires customized implementations of the `_I2CRead`, `_I2CWrite`, and `_Sleep` functions. For more information, see the "I^2C Interface and Data Transmission Protocol" section of the *ZMOD4410 Datasheet* and the *ZMOD4410-XXX-Firmware-Documentation.pdf* and *.html*. If possible, try to verify your implementation with a scope or logic analyzer. Conduct a register test as described in the datasheet.
4. Modify the file *src/example.c* as required or copy its functionality to your own application code. Make sure to leave the order of operations and timing unchanged.
   Before continuing to the next step, you can also try compiling and running the application without the algorithm libraries. Comment out calls to zmod4xxx_cleaning_run, init_* and calc_* functions and check if zmod4xxx_read_adc_results() function outputs changing ADC values in main measurement loop.
5. Link the application code with the algorithm and cleaning libraries matching your microcontroller and compiler. A list of supported MCUs and compilers is provided in Table 2.

5.2 Interrupt Usage and Measurement Timing

The firmware examples are written with delays. The microcontroller is blocked during these time periods. Depending on target hardware and the application, this can be avoided by using interrupts. The measurement sequences for each algorithm are displayed in the following figure.

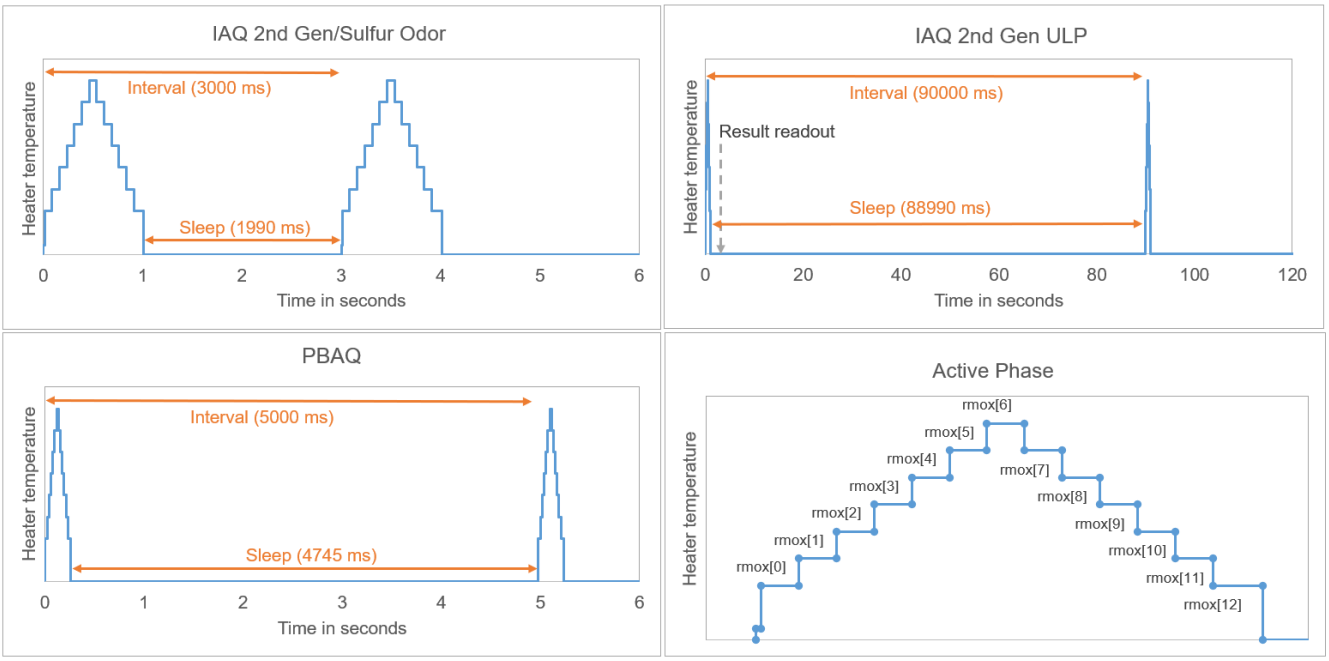


Figure 3. Measurement Sequences

An active measurement is indicated with a heater target temperature greater than zero. To lower power consumption a Sleep phase follows. When using ZMOD4410 together with ZMOD4510 for compensation of oxidizing gases, the ZMOD4510 measurement interval (sample rate) of 6 seconds should be considered. The timing must be kept exactly with a maximum deviation of 5% to keep the algorithm accuracy. Each measurement must be restarted with the API command *zmod4xxx_start_measurement*.

A timer interrupt can be used to start each measurement sequence. Note that an ADC read-out just before the end of the active measurement phase when results are written to the registers will lead to an error. When replacing the delay make sure that the measurement is completed before the ADC readout by checking with API function *zmod4xxx_read_status* and compare the output variable *zmod4xxx_status* with *STATUS_SEQUENCER_RUNNING_MASK*. An AND link of both should give zero, otherwise the measurement was not completed.

Instead of reading out the results directly after the measurement, another option is to use just one timer interrupt with the measurement interval. Then, the ADC result read-out, error check, and algorithm calculation is done just before starting the next measurement (default for IAQ 2nd Gen and PBAQ). This option will increase the sensor response time with the measurement interval length.

6. Revision History

Revision	Date	Description
1.16	June 20, 2024	▪ Updated ZMOD4510 sequence for IAQ 2nd Gen. ▪ Added ozone compensation mode for PBAQ. ▪ Completed other minor changes.
1.15	Mar 12, 2024	▪ Updated IAQ 2nd Gen and ULP description. ▪ Removed Sulfur Odor description (to be maintained in firmware package). ▪ Moved cleaning and self-check description to the *ZMOD4410 Datasheet*. ▪ Updated compiling strategies. ▪ Restructured various sections and completed minor edits.
1.14	Jul 13, 2023	▪ Completed several changes to support new EVK hardware ▪ Updated target and compiler list
1.13	Mar 7, 2023	▪ Updated to include a broader definition of Public Building Air Quality (PBAQ)
1.12	Jan 30, 2023	▪ Added PBAQ example description ▪ Removed Arduino ATmega32 support ▪ Completed other minor changes
1.11	Nov 7, 2022	▪ Completed minor edits.
1.10	Oct 12, 2022	▪ Added Rel IAQ and Rel IAQ ULP description ▪ Added Raspberry Pi description ▪ Updated cleaning functionality ▪ Added sensor damage self-check functionality ▪ Added RH/T compensation functionality for IAQ 2nd Gen ▪ Removed C90 support ▪ Removed Odor descriptions (legacy) ▪ Completed other minor changes
1.09	Dec 2, 2021	▪ Added IAQ 2nd Gen ULP Operation Mode description ▪ Added error code description ▪ Removed IAQ 1st Gen descriptions (legacy) ▪ Completed other minor changes
1.08	Aug 19, 2021	▪ Added Arduino description and updated target and compiler list. ▪ Added stack RAM usage. ▪ Corrected and reworked Program Flows. ▪ Extended “Interrupt Usage” description. ▪ Completed other minor changes
1.07	Sep 24, 2020	▪ Add sections “Interrupt Usage”, “Adaptions to Follow C90 Standard”, “How to Compile for EVK Hardware”. ▪ Add example for memory footprint and update target and compiler list. ▪ Refined implementation steps. ▪ Minor edits in text.
1.06	May 27, 2020	▪ Completed many changes throughout the document.
1.05	Nov 14, 2019	▪ Cleaning procedure added and explained. ▪ Figure for file overview updated.

Revision	Date	Description
1.04	Feb 12, 2019	▪ Update for change in the program flow for Continuous (skip the first 10 samples) and Low Power (skip the first 5 samples) Operation Modes. ▪ Implementation of plain trim value calibration. ▪ Minor edits in text.
1.03	Dec 5, 2018	▪ Update for Low Power Operation. ▪ Minor edits.
1.02	Sep 27, 2018	▪ Revision of document title from ZMOD44xx Programming Manual with ZMOD4410 Example to ZMOD4410 Programming Manual – Read Me. ▪ Full update for Odor Operation Mode 2. ▪ Minor edits.
1.00	Jun 11, 2018	Initial release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01 Jan 2024)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.