SENSIRION
THE SENSOR COMPANY

SGP40 VOC Index Driver Integration

Sensirion AG

May 07, 2021

document version: 7.1.2

Contents

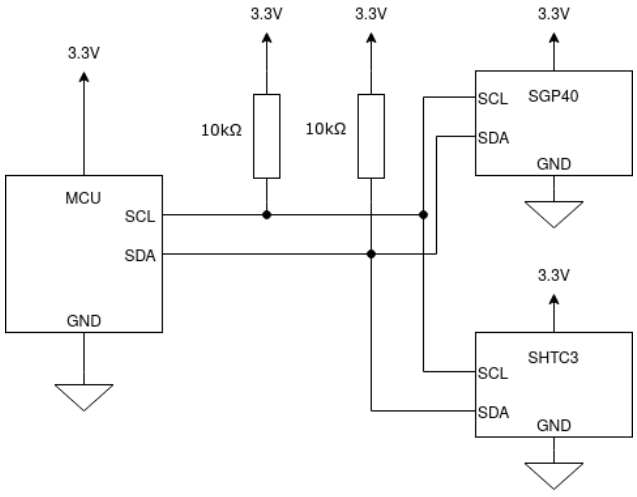
This application note will provide you with an introduction to the SGP40 VOC Index driver bundle.

1 Overview

This bundle combines Sensirion's SGP40 and SHTC3 sensors to provide a humidity compensated sensor signal. It further embeds a VOC algorithm that post-processes the raw sensor output into the resulting VOC Index.

2 Hardware setup

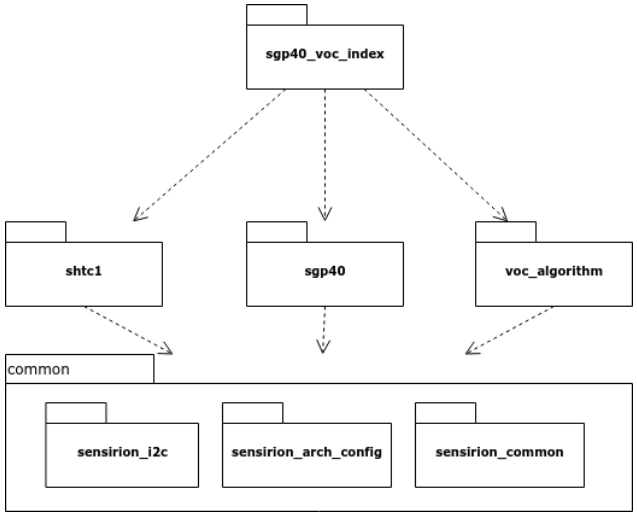
The hardware setup uses a single I2C bus where both sensors are connected to the micro controller unit (MCU). Connect the sensors according to the datasheet. Remember to add pull-up resistors for SCL and SDA (10kOhm).



3 Software Overview

The software is split into the following modules:

- *common*: Contains abstractions for I2C, the MCU architecture and some common helpers
- *shtc1* The driver for the SHTC3 sensor (also compatible with SHTC1, SHTW1 and SHTW2)
- *sgp40* The driver for the SGP40 sensor
- *voc_algorithm* The VOC algorithm to process the raw sensor signal and calculate the VOC index.
- *sgp40_voc_index* Combines the two drivers and the VOC algorithm to provide a simple API.



3.1 API

The example *sgp40_voc_index_example_usage.c* gives a starting point on how to use the sensors.

The SGP40 VOC Index driver provides the following functions:

int16_t **sensirion_init_sensors**()
Initialize the SGP40, SHT and VOC algorithm.

Return STATUS_OK on success, an error code otherwise

int16_t **sensirion_measure_voc_index**(int32_t **voc_index*)
Measure the humidity-compensated VOC Index.

The measurement triggers a humidity reading, sets the value on the SGP for humidity compensation and runs the gas signal through the VOC algorithm for the final result.

This command works like sensirion_measure_voc_index_with_rh_t() but does not return the measured ambient humidity and temperature used for compensation.

Return STATUS_OK on success, an error code otherwise

Parameters

- `voc_index`: Pointer to buffer for measured voc_index. Range 0..500.

int16_t **sensirion_measure_voc_index_with_rh_t** (int32_t **voc_index*, int32_t **relative_humidity*, int32_t **temperature*)

Measure the humidity-compensated VOC Index and ambient temperature and relative humidity.

This command works like sensirion_measure_voc_index() but also returns the measured ambient humidity and temperature used for compensation.

Return STATUS_OK on success, an error code otherwise

Parameters

- `voc_index`: Pointer to buffer for measured VOC index. Range 0..500.
- `relative_humidity`: Pointer to buffer for relative humidity in milli RH
- `temperature`: Pointer to buffer for measured temperature in milli degree Celsius.

4 Getting Started on Raspberry Pi

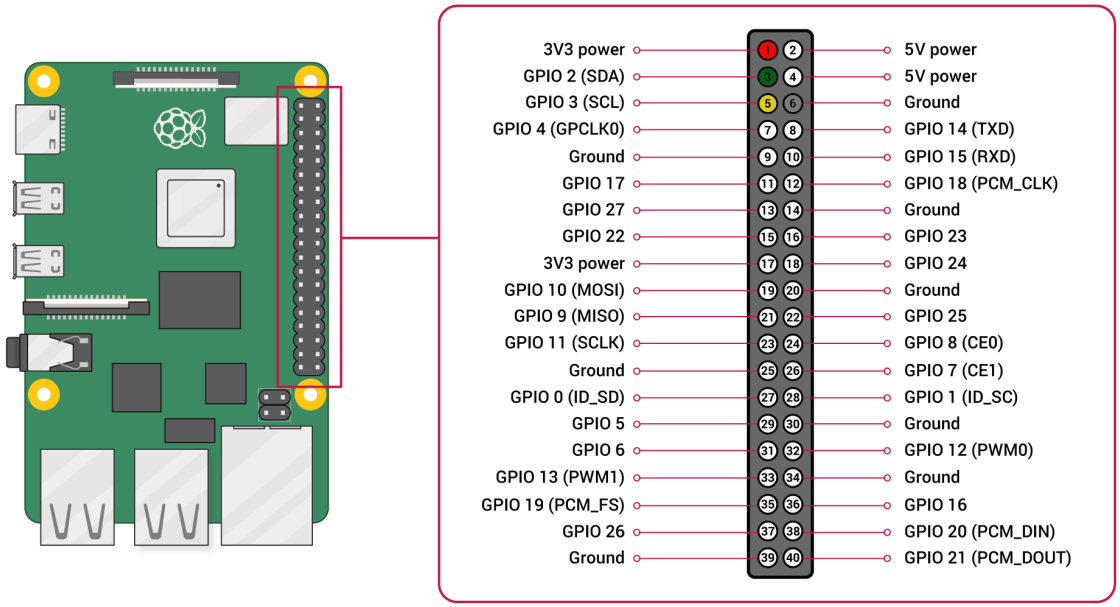
This document explains how to set up the SGP40 VOC Index driver bundle to run on a Raspberry Pi. Since the SGP40 and SHTC3 sensors run at 3.3V, which matches the Raspberry Pi's logic level, no level shifting is required on the I2C bus.

4.1 Setup Guide

Connecting the Sensor

Your sensor has the four different connectors: VCC, GND, SDA, SCL. Use the following pins to connect your SGP40:

SGP40 and SHTC3	*Raspberry Pi*
VCC	Pin 1 (3.3V source)
GND	Pin 6
SDA	Pin 3
SCL	Pin 5



Raspberry Pi

- **Install the Raspberry Pi OS on to your Raspberry Pi**
  - For help with this, kindly refer to the Raspberry Pi website: https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up
- **Enable the I2C interface in the Raspberry Pi config**
  - Open a terminal
  - Run `sudo raspi-config`
  - Select “5 Interfacing Options Configure connections to peripherals”
  - Select “P5 I2C Enable/Disable automatic loading of I2C kernel module”
  - Select “Yes” when questioned “Would you like the ARM I2C interface to be enabled?”
  - The Raspberry Pi should respond with “The ARM I2C interface is enabled”. Confirm with “Ok”
- **Download driver**
  - Go to the Sensirion SGP Driver Release page (https://github.com/Sensirion/embedded-sgp/releases) and download the latest `sgp40-VERSION.zip` file, whereas `VERSION` represents the latest version.
  - Unzip the file into the directory where you want to install the driver (Run: `unzip sgp40-voc-index-VERSION.zip`).
- **Adapt the driver to work with Raspberry Pi**
  - We use the `linux_user_space` implementation from `hw_i2c` (The Linux kernel provides an I2C controller, thus “hardware” I2C) to run the driver on the Raspberry Pi. To use the in the project provided sample implementation navigate to the `./hw_i2c/sample-implementations/linux_user_space` directory.

  - Copy the file `sensirion_hw_i2c_implementation.c` from there and replace the file named the same directly in `./hw_i2c` or just run this command in the root directory of your project: `cp ./hw_i2c/sample-implementations/linux_user_space/sensirion_hw_i2c_implementation.c ./hw_i2c/.`

- **Compile the driver**
  - Run `make` in the root directory of your project.

Output:

```
rm -f sgp40_voc_index_example_usage
cc -Os -Wall -fstrict-aliasing -Wstrict-aliasing=1 -fPIC -I. -I. -I. -I. -I. -I. -I./
↪hw_i2c -o sgp40_voc_index_example_usage ./sensirion_arch_config.h ./sensirion_i2c.h .
↪/sensirion_common.h ./sensirion_common.c ./sgp_git_version.h ./sgp_git_version.c ./
↪sht_git_version.h ./sht_git_version.c ./sgp40.h ./sgp40.c ./shtc1.h ./shtc1.c ./
↪sensirion_voc_algorithm.h ./sensirion_voc_algorithm.c ./sgp40_voc_index.h ./sgp40_
↪voc_index.c ./hw_i2c/sensirion_hw_i2c_implementation.c ./sgp40_voc_index_example_
↪usage.c -lm
```

- **To test the setup run the example code which was just compiled with** `./sgp40_voc_index_example_usage`

Output:

```
initialization successful
VOC Index: 0
Temperature: 39.128degC
Relative Humidity: 29.695%RH
VOC Index: 0
Temperature: 39.086degC
Relative Humidity: 29.687%RH
VOC Index: 0
Temperature: 39.106degC
Relative Humidity: 29.647%RH
VOC Index: 0
Temperature: 39.076degC
Relative Humidity: 29.714%RH
VOC Index: 0
Temperature: 39.068degC
Relative Humidity: 29.615%RH
VOC Index: 0
Temperature: 39.082degC
Relative Humidity: 29.674%RH
...
```

4.2 Troubleshooting

Initialization failed

- Ensure that you connected the sensor correctly: all cables are fully so all cables are fully plugged in and connected to the correct header.
- Ensure that I2C is enabled on the Raspberry Pi. For this redo the steps on "Enable the I2C interface in the Raspberry Pi config" in the guide above.
- Ensure that your user account has read and write access to the I2C device. If it only works with user root (`sudo ./sgp40_example_usage`), it's typically due to wrong permission settings.

5 Getting Started On a Custom Platform

5.1 Copy files to your project

1. Copy all top level source files (.c and .h) into your software project folder.
2. Copy *hw_i2c/sensirion_hw_i2c_implementation.c* to your project as well. You'll need to adapt it to your platform. See *Implement sensirion_hw_i2c_implementation.c*.
3. Make sure all files are added to your IDE.

5.2 Adapt *sensirion_arch_config.h* for your platform

You may need to adapt *sensirion_arch_config.h* if your compiler doesn't support C99 and thus does not provide *stdint.h* and *stdlib.h*.

5.3 Implement *sensirion_hw_i2c_implementation.c*

To use your I2C hardware the file *hw_i2c/sensirion_hw_i2c_implementation.c* needs to be completed. In it you find the following functions where all parts marked with "// IMPLEMENT" have to be replaced with code performing the necessary setup described here.

Alternatively you can use a sample implementation from *hw_i2c/sample-implementations/* and override it.

I2C functions to implement

void **sensirion_i2c_init**(void)
Initialize all hard- and software components that are needed for the I2C communication.

void **sensirion_i2c_release**(void)
Release all resources initialized by sensirion_i2c_init().

int8_t **sensirion_i2c_read**(uint8_t *address*, uint8_t **data*, uint16_t *count*)
Execute one read transaction on the I2C bus, reading a given number of bytes. If the device does not acknowledge the read command, an error shall be returned.

Return 0 on success, error code otherwise

Parameters

- `address`: 7-bit I2C address to read from
- `data`: pointer to the buffer where the data is to be stored
- `count`: number of bytes to read from I2C and store in the buffer

int8_t **sensirion_i2c_write**(uint8_t *address*, **const** uint8_t **data*, uint16_t *count*)
Execute one write transaction on the I2C bus, sending a given number of bytes. The bytes in the supplied buffer must be sent to the given address. If the slave device does not acknowledge any of the bytes, an error shall be returned.

Return 0 on success, error code otherwise

Parameters

- `address`: 7-bit I2C address to write to

- `data`: pointer to the buffer containing the data to write
- `count`: number of bytes to read from the buffer and send over I2C

void **`sensirion_sleep_usec`**(uint32_t *useconds*)

Sleep for a given number of microseconds. The function should delay the execution approximately, but no less than, the given time.

When using hardware i2c: Despite the unit, a <10 millisecond precision is sufficient.

When using software i2c: The precision needed depends on the desired i2c frequency, i.e. should be exact to about half a clock cycle (defined in `SENSIRION_I2C_CLOCK_PERIOD_USEC` in `sensirion_arch_config.h`).

Example with 400kHz requires a precision of 1 / (2 * 400kHz) == 1.25usec.

Parameters

- `useconds`: the sleep time in microseconds

int16_t **`sensirion_i2c_select_bus`**(uint8_t *bus_idx*)

Select the current i2c bus by index. All following i2c operations will be directed at that bus.

THE IMPLEMENTATION IS OPTIONAL ON SINGLE-BUS SETUPS (all sensors on the same bus)

Return 0 on success, an error code otherwise

Parameters

- `bus_idx`: Bus index to select

5.4 Run *sgp40_voc_index_example_usage.c*

If your platform supports the `printf` function just run the examples and you should see the following messages:

```
initialization successful
VOC Index: 0
Temperature: 39.128degC
Relative Humidity: 29.695%RH
VOC Index: 0
Temperature: 39.086degC
Relative Humidity: 29.687%RH
VOC Index: 0
Temperature: 39.106degC
Relative Humidity: 29.647%RH
...
```

If your platform doesn't support `printf` remove the print statements and observe the measurement values in your debugger.

Index

S