SGP30 Driver Integration (for Software I²C)

A Step-By-Step Guide

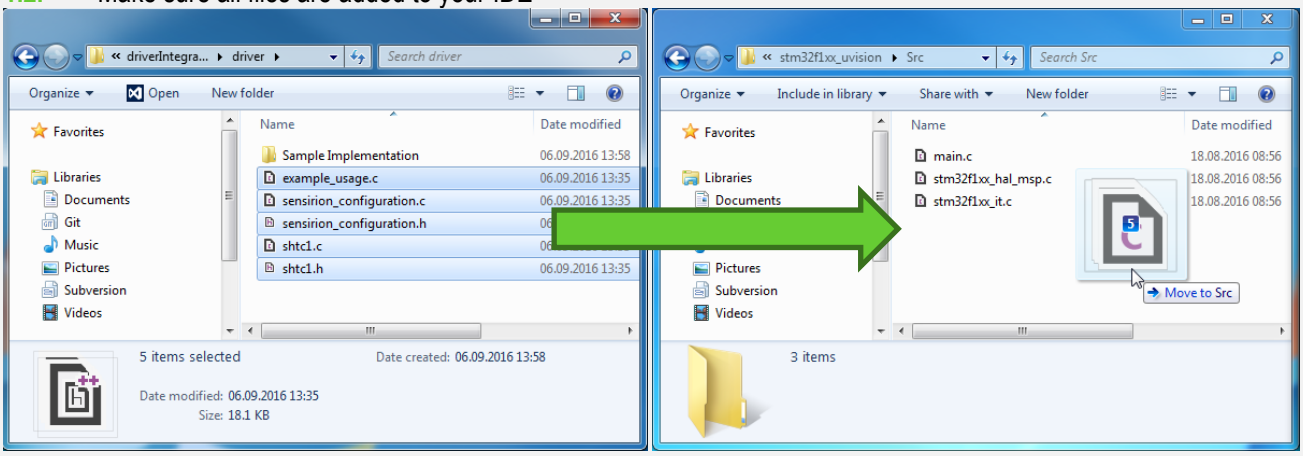
Preface

The easiest way to integrate the SGP30 sensor into a device is Sensirion's SGP30 driver. This document explains how to implement the hardware abstraction layer of the SGP driver and describes the provided API

COPY FILES TO YOUR PROJECT — STEP 1 | 2 | 3

1.1. Copy all SGP driver files (.c and .h) into your software project folder.

1.2. Make sure all files are added to your IDE



IMPLEMENT sensirion_configuration.c — 1 | STEP 2 | 3

To use software I²C the file **sensirion_configuration.c** (or *sensirion_configuration.cpp* for c++ projects) needs to be completed. All parts marked with "`// IMPLEMENT`" have to be replaced with the according setup.

2.1. Initialize the components needed to set SDA and SCL pins.

```
void sensirion_init_pins()
{
    // IMPLEMENT
}
```

2.2. Implement `sensirion_SDA_in()` and `sensirion_SCL_in()` to configure the SDA and SCL pins as input.

```
void sensirion_SDA_in()
{
    // IMPLEMENT
}

void sensirion_SCL_in()
{
    // IMPLEMENT
}
```

2.3. Implement `sensirion_SDA_out()` and `sensirion_SCL_out()` to configure the pins as output.

```
void sensirion_SDA_out()
{
    // IMPLEMENT
}

void sensirion_SCL_out()
{
    // IMPLEMENT
}
```

2.4. Implement `sensirion_SDA_read()` and `sensirion_SCL_read()` to read the values from the pins.

```
uint8_t sensirion_SDA_read()
{
    // IMPLEMENT
    return 1;
}

uint8_t sensirion_SCL_read()
{
    // IMPLEMENT
    return 1;
}
```

Return: 0 if the pin is low and 1 otherwise.

2.5. Implement `sensirion_sleep_usec()` to delay the execution for the given time in microseconds.

```
void sensirion_sleep_usec(uint32_t useconds) {
    // IMPLEMENT
}
```

MEASURE IAQ (tVOC / CO2eq) AND SIGNAL VALUES 1 2 STEP 3

The SGP driver provides functions to probe the sensor, to get the serial ID, and to measure/read tVOC and CO2-eq.

3.1. Call `sgp_get_feature_set_version()` to readout the feature set version and product type of the SGP sensor. If product_type is 0 it is a SGP30 gas sensor, if it is 1 it is an SGPC3 gas sensor.

```
int16_t sgp_probe(void);
```

Return: 0 if the sensor is detected, else an error code.

3.2. Call `sgp_get_serial_id()` to readout the serial ID of the SGP sensor.

```
int16_t sgp_get_feature_set_version (u16 *feature_set_version, u8 *product_type);
```

3.3. Call `sgp_measure_iaq_blocking_read()` to start a tVOC and CO2-eq measurement and to readout the values.

```
int16_t sgp_measure_iaq_blocking_read(uint16_t *tvoc_ppb, uint16_t *co2_eq_ppm);
```

Note: This function blocks the processor while the measurement is in progress.
Return: 0 if the command is successful, else an error code.

3.4. For non-blocking measurement and readout of tVOC and CO2-eq use the two functions `sgp_measure_iaq()` and `sgp_read_iaq()`.

```
int16_t sgp_measure_iaq(void);
```

Return: 0 if the command is successful, else an error code.

Be sure to wait until the SGP sensor finishes the measurement. Read the measurement result using `sgp_read_iaq()`.

```
int16_t sgp_read_iaq(uint16_t *tvoc_ppb, uint16_t *co2_eq_ppm);
```

Note: If the measurement is still in progress, this function returns an error code.
Return: 0 if the command is successful, else an error code.

3.5. For best performance and faster startup times, the current baseline needs to be persistently stored on the device before shutting it down and set again accordingly after boot up.

Use `sgp_get_iaq_baseline()` to get the baseline.

```
int16_t sgp_get_iaq_baseline (uint32_t *baseline);
```

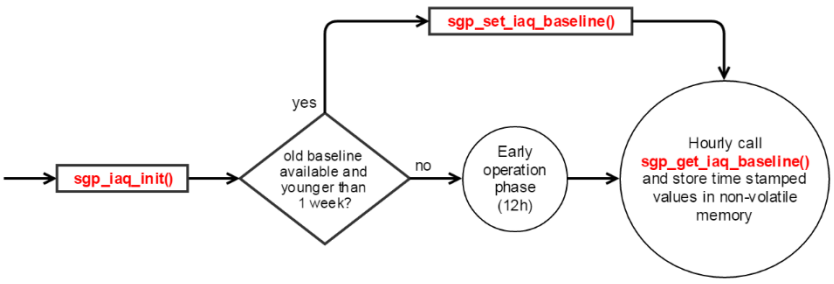
Return: 0 if the command is successful, else an error code.
Note: If the call is not successful, the `baseline` value must be discarded. Approximately in the first 60min of operation after `sgp_probe` or `sgp_iaq_init` the call will fail unless a previous baseline was restored.

3.6. Use `sgp_set_iaq_baseline()` to set the baseline.

```
int16_t sgp_set_iaq_baseline (uint32_t baseline);
```

Return: 0 if the command is successful, else an error code.
Note: The baseline value must be *exactly* as returned by `sgp_get_iaq_baseline()` and should only be set if it's less than one week old.

3.7. SGP baseline states



Call `sgp_iaq_init()` to reset all SGP baselines. The initialization takes up to around 15 seconds, during which `sgp_iaq_measure()` output will not change.

If no stored baseline is available after initializing the baseline algorithm, the sensor has to run for 12 hours until the baseline can be stored. This will ensure an optimal behavior for preceding startups. Reading out the baseline prior should be avoided unless a valid baseline is restored first. Once the baseline is properly initialized or restored, the current baseline value should be stored approximately once per hour. While the sensor is off, baseline values are valid for a maximum of seven days.

3.8. Call `sgp_iaq_init()` to initialize or re-initialize the indoor air quality algorithm.

```
int16_t sgp_iaq_init(void);
```

Return: 0 if the command is successful, else an error code.

Note: `sgp_iaq_init()` is already called as part of `sgp_probe()`.

3.9. Call `sgp_measure_tvoc_blocking_read()` to start a tVOC measurement and to readout the value in ppb.

```
int16_t sgp_measure_tvoc_blocking_read(uint16_t *tvoc_ppb);
```

Note: This function blocks the processor while the measurement is in progress.

Return: 0 if the command is successful, else an error code.

3.10. For non-blocking measurement and readout of tVOC use the two functions `sgp_measure_tvoc()` and `sgp_read_tvoc()`

```
int16_t sgp_measure_tvoc(void);
```

Return: 0 if the command is successful, else an error code.

Be sure to wait until the SGP sensor finishes the measurement. Read the measurement result using `sgp_read_tvoc()`.

```
int16_t sgp_read_tvoc(uint16_t *tvoc_ppb);
```

Note: If the measurement is still in progress, this function returns an error code.

Return: 0 if the command is successful, else an error code.

3.11. Call `sgp_measure_co2_eq_blocking_read()` to start a CO2-eq measurement and to readout the value in ppm.

```
int16_t sgp_measure_co2_eq_blocking_read(uint16_t *co2_eq_ppm);
```

Note: This function blocks the processor while the measurement is in progress.
Return: 0 if the command is successful, else an error code.

3.12. For non-blocking measurement and readout of CO2-eq use the two functions `sgp_measure_co2_eq()` and `sgp_read_co2_eq()`.

```
int16_t sgp_measure_co2_eq(void);
```

Return: 0 if the command is successful, else an error code.

Be sure to wait until the SGP sensor finishes the measurement. Read the measurement result using `sgp_read_co2_eq()`.

```
int16_t sgp_read_co2_eq(uint16_t *co2_eq_ppm);
```

Note: If the measurement is still in progress, this function returns an error code.
Return: 0 if the command is successful, else an error code.

3.13. Call `sgp_measure_signals_blocking_read()` to start signal measurements and to readout the values.

```
int16_t sgp_measure_signals_blocking_read(uint16_t *scaled_ethanol_signal,
                                          uint16_t *scaled_h2_signal);
```

Return: 0 if the command is successful, else an error code.
Note: The returned values are scaled signals. To get the resulting signals, the values must first be divided by 512.
Note: This function blocks the processor while the measurement is in progress. This function modifies the IAQ baseline, thus when using IAQ functions `sgp_measure_iaq()`, `sgp_measure_tvoc()`, `sgp_measure_co2_eq()` and their blocking_read equivalents, the baseline must be saved prior to running this call, and restored after:

```
int16_t ret;
uint16_t scaled_ethanol_signal, scaled_h2_signal;
```

```
// save current baseline
uint32_t baseline;
ret = sgp_get_iaq_baseline(&baseline);

// run signals measurement
ret = sgp_measure_signals_blocking_read(&scaled_ethanol_signal, &scaled_h2_signal);

// wait and read signals, then restore baseline
ret = sgp_set_iaq_baseline(baseline);
```

3.14. For non-blocking measurement and readout of signals values use the two functions `sgp_measure_signals()` and `sgp_read_signals()`.

```
int16_t sgp_measure_signals(void);
```

Return: 0 if the command is successful, else an error code.

Note: This function modifies the IAQ baseline, thus when using IAQ functions `sgp_measure_iaq()`, `sgp_measure_tvoc()`, `sgp_measure_co2_eq()` and their blocking_read equivalents, the baseline must be saved prior to running this call, and restored after:

```
int16_t ret;
uint16_t scaled_ethanol_signal, scaled_h2_signal;

// save current baseline
uint32_t baseline;
ret = sgp_get_iaq_baseline(&baseline);

// run signals measurement
ret = sgp_measure_signals();
usleep(200000);
sgp_read_signals(&scaled_ethanol_signal, &scaled_h2_signal)

// wait and read signals result, then restore baseline
ret = sgp_set_iaq_baseline(baseline);
```

Be sure to wait until the SGP sensor finishes the measurement. Read the measurement results using `sgp_read_signals()`.

```
int16_t sgp_read_signals(uint16_t *scaled_ethanol_signal,
                         uint16_t *scaled_h2_signal);
```

Note: If the measurement is still in progress, this function returns an error code.
Note: The returned values are scaled signals. To get the resulting signals, the values must first be divided by 512.
Return: 0 if the command is successful, else an error code.

3.15. Call `sgp_set_absolute_humidity()` to a value greater than 0 and smaller than 256000 mg/m³ to enable the humidity compensation feature, or write 0 to disable it.
The absolute humidity in g/m³ can be retrieved by measuring the relative humidity and temperature using a Sensirion SHT sensor and converting the value to absolute humidity with the formula

$$AH = 216.7 \cdot \frac{\frac{RH}{100.0} \cdot 6.112 \cdot \exp\frac{17.62 \cdot t}{243.12 + t}}{273.15 + t}$$

With *AH* in g/m³, *RH* in 0-100%, and *t* in °C

Note: the value in g/m³ has to be multiplied by 1000 to convert to mg/m³ and any remaining decimal places have to be rounded and removed since the interface does not support floating point numbers.

```
int16_t sgp_set_absolute_humidity(uint32_t absolute_humidity);
```

Return: 0 if the command is successful, else an error code.

Note: The humidity compensation is disabled by setting the value to 0.
Example: To set the absolute humidity to 13.000 g/m³:

```
// Set absolute humidity to 13.000 g/m^3
uint32_t ah = 13000;
sgp_set_absolute_humidity(ah);
```

3.16. Call `sgp_measure_test()` to run the on-chip self-test. This command can be used during production to ensure the SGP30 is not damaged. A success is indicted by a return code of 0, in that case the value of `test_result` will be *0xd400* .

```
// Run the on-chip self-test
uint16_t test_result;
int16_t ret = sgp_measure_test(&test_result);
if (ret != STATUS_OK) {
   // The sensor is likely damaged
}
```

Note: `sgp_measure_test()` must not be executed after `sgp_iaq_init()`. If this is needed, the baseline must be retrieved prior to running `sgp_measure_test`. After `sgp_measure_test`, `sgp_iaq_init` followed by setting the baseline again is needed to resume IAQ operations.

3.17. Call `sgp_get_driver_version()` to retrieve the driver version.

```
const char *sgp_get_driver_version(void);
```

Return: The driver version string is returned in the form "major.minor.patchset" e.g. "2.2.1"

REVISION HISTORY

Date	Version	Page(s)	Changes
October 2016	1.0.0	all	Initial release
January 2017	1.0.1	all	Add CO2-eq output to the driver
January 2017	1.0.2	all	Fixing layout
January 2017	1.1.0	all	Add IAQ functions
January 2017	1.1.1	3	Document how long a baseline value is valid
March 2017	1.2.0	3-5	Update baseline documentation
March 2017	1.4.0	3	Update baseline persistence documentation
April 2017	1.5.0	1, 3	SGP30
May 2017	2.0.0	all	Change interfaces from resistance to ethanol and h2 signals
May 2017	2.0.1	all	Document signal scaling
August 2017	2.1.0	6	Add humidity compensation, measure_test
August 2017	2.1.1	6	Document driver version

Headquarters and Subsidiaries

Sensirion AG
Laubisruetistr. 50
CH-8712 Staefa ZH
Switzerland

Phone: +41 44 306 40 00
Fax: +41 44 306 40 30
info@sensirion.com
www.sensirion.com

Sensirion Inc., USA
Phone: +1 805 409 4900
info_us@sensirion.com
www.sensirion.com

Sensirion Japan Co. Ltd.
Phone: +81 3 3444 4940
info@sensirion.co.jp
www.sensirion.co.jp

Sensirion Korea Co. Ltd.
Phone: +82 31 345 0031 3
info@sensirion.co.kr
www.sensirion.co.kr

Sensirion China Co. Ltd.
Phone: +86 755 8252 1501
info@sensirion.com.cn
www.sensirion.com.cn

Sensirion AG (Germany)
Phone: +41 44 927 11 66
info@sensirion.com
www.sensirion.com

To find your local representative, please visit www.sensirion.com/contact