SENSIRION
THE SENSOR COMPANY

SGP30 Linux Driver Documentation

A Step-by-Step Guide

Preface

The easiest way to integrate the SGP30 sensor into a Linux device is Sensirion's *sgp30* Linux driver. This document explains how to cross compile the driver for Linux targets.

PREREQUISITES — STEP 1 — 2 — 3

1.1. You will need the sources to the exact Linux kernel running on your target device.

To use the provided build scripts, the following environment variables must be defined:
1. The source directory must be exported as `$KERNELDIR`,
2. The target architecture must be exported as `$ARCH`,
3. If needed, the cross compiler must be defined as `$CROSS_COMPILE`.

Example:

```
$ export KERNELDIR=$HOME/linux-4.4.21
$ export ARCH=arm
$ export CROSS_COMPILE=$HOME/toolchain/bin/gcc-linaro-arm-linux-gnueabi-
```

1.1.1. The kernel must be configured with CRC8 (`CONFIG_CRC8`) and iio (`CONFIG_IIO`) support.
Run these commands in your kernel directory to enable those options:

```
$ cd $KERNELDIR
$ ./scripts/config --enable CRC8 --enable IIO
```

Note: You will need to rebuild your kernel with these options enabled (also possible as kernel modules).

BUILD, TEST AND INSTALL KERNEL MODULE — 1 — STEP 2 — 3

Note: Root access is needed for any command prefixed by the pound sign (#)
Run `sudo -s` to obtain a root shell.

```
$ sudo -s
```

2.1. To build the kernel modules for the prepared kernel (see Step 1), run `make modules` in the SGP driver directory:

```
$ cd $HOME/sgp30
$ make modules
```

The resulting kernel module *sgpxx.ko* is placed in the same directory.

2.2. To test the module on the target device without installing it, load it ("insert") by running `insmod sgp30.ko` as root.

```
# insmod sgp30.ko
```

Note: on error `insmod: ERROR: could not insert module sgp30.ko: Unknown symbol in module`, try loading the module dependencies `crc8` and/or `industrialio` first, if compiled as module:

```
# modprobe industrialio crc8
```

2.3. Once inserted, a new driver instance must be created on the appropriate I²C bus with the sensor's address (0x58)

```
# I2CBUS=1
# echo sgp30 0x58 > /sys/bus/i2c/devices/i2c-$I2CBUS/new_device
# ls /sys/bus/i2c/devices/i2c-$I2CBUS/$I2CBUS-0058/
```

Note: You will have to determine the correct bus id on your device (`$I2CBUS` in the example above)
On Android, the device path may be slightly different:

```
# echo sgp30 0x58 > /sys/devices/i2c-$I2CBUS/new_device
# ls /sys/devices/i2c-$I2CBUS/$I2CBUS-0058/
```

If the sensor is correctly detected, the device's folder will contain a folder `ii:device0`. If this folder is absent, no sensor was detected. Please consult `dmesg` output as it provides useful kernel debug information on whether the sensor was detected or not.

```
# dmesg
[ 1465.835588] i2c i2c-0: new_device: Instantiated device sgp30 at 0x58
```

Note: The result above shows a successful sensor detection on bus 0, while the output below shows a failed instantiation due to a missing sensor on bus 1.

```
# dmesg
[ 1542.015653] sgp30: probe of 1-0058 failed with error -5
[ 1542.015700] i2c i2c-0: new_device: Instantiated device sgp30 at 0x58
```

2.4. To remove the driver instance (before unloading the kernel module) or re-instantiate the driver to cause a new sensor probing, execute the following command:

```
# echo 0x58 > /sys/bus/i2c/devices/i2c-$I2CBUS/delete_device
```

Note: On Android, the device path may be slightly different:

```
# echo 0x58 > /sys/devices/i2c-$I2CBUS/delete_device
```

2.5. To unload the driver, remove all driver instances as described in 2.4. and then run

```
# rmmod sgp30
```

Note: Alternatively, `modprobe -r sgp30` can be used

2.6. To permanently install the driver module on the target device, run `make modules_install` from the driver directory. To unload the driver, remove all driver instances as described in 2.4. and then run

```
$ cd $HOME/sgp30
$ sudo -s
# make modules_install
```

Note: `sudo –s` step above may not be necessary if the install path, as defined by the kernel, is user writable.

2.7. To retrieve the software version of the kernel module, run `modinfo sgp30` when the module is loaded or `modinfo sgp30.ko` on the compiled module.

```
# modinfo sgp30
```

Note: This only works if the driver is compiled as module.

MEASURE IAQ (tVOC / CO_2eq) AND GAS SIGNALS 1 2 STEP 3

The SGP30 driver provides functions to get the serial ID and to measure / read tVOC.
The following commands must all be run from the sensor instance's directory.

```
$ cd /sys/bus/i2c/devices/$I2CBUS-0058/iio:device0
```

In the following section, the return value of the command is displayed as part of the example commands. The line prefixed with the dollar sign ($) represents the command, while the following non-prefixed lines represent the printed response.

Note: The sensor initialization happens automatically at module load-time and takes up to around 15 seconds, during which `in_concentration_co2_input and in_concentration_voc_input` will report a busy error.
If a previously retrieved baseline exists, it may be set to speed up the initialization (see step 3.6.)
After loading the sensor is periodically polled in the background by the driver to keep the internal baseline updated.

3.1. Read `in_serial_id` to access the Serial ID of the SGP30 sensor

```
$ cat in_serial_id
56789012
```

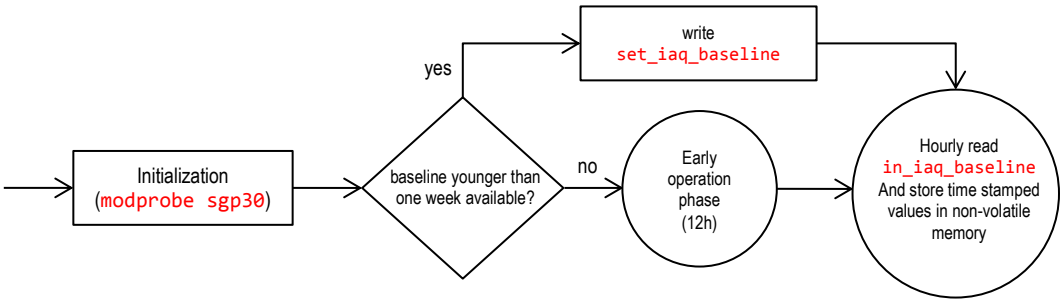
Note: The Serial ID is read only once when the driver is instantiated and remains cached for future calls. To re-read the Serial ID, the driver instance must first be deleted (See step 2.4.) and re-instantiated (See step 2.3.).

3.2. Read `in_concentration_voc_input` or `in_concentration_co2_input` to perform a single IAQ measurement

```
$ cat in_concentration_co2_input
0.000400
$ cat in_concentration_voc_input
0.000000005
```

Note: The call returns the last cached value of the periodic polling that happens in the background.
`tvoc` is returned in **ppb** precision units (1e-6), while `co2_eq` is returned in **ppm** precision units (1e-9).

3.3. SGP30 baseline states



After the initialization, when loading the driver module, the sensor has to be operated for **12 hours** until the baseline can be stored. This will ensure an optimal behavior for preceding startups. Reading out the baseline before those 12 hours is possible but should be avoided unless a valid baseline is set with `set_iaq_baseline`. Once the baseline is properly initialized or restored, the current baseline value should be stored approximately once per hour. When the sensor is switched off, baseline values are valid for a maximum of seven days.

Baseline persistence logic in pseudo code:

```
BASELINE_VALID_S = 7 * 24 * 3600;      // 1 week
EARLY_OPERATION_PHASE_S = 12 * 3600;   // 12 hours
PERSISTANCE_INTERVAL_S = 3600;         // 1 hour

// [Module initialization]

load_baseline_from_disk(&baseline, &timestamp);

if (baseline && ((now() - timestamp) < BASELINE_VALID_S))
    // Baseline is available
    set_iaq_baseline(baseline);
    goto regular_operation_phase;

early_operation_phase:
for (counter = 0; counter < EARLY_OPERATION_PHASE_S; counter++)
    sleep(1s);
    iaq_measure(&tvoc_ppb, &co2_eq_ppm);

regular_operation_phase:
counter = 0;
while (1)
    sleep(1s)
    counter++;
    iaq_measure(&tvoc_ppb, &co2_eq_ppm);
    if (counter % PERSISTANCE_INTERVAL_S == 0)
        baseline = get_iaq_baseline();
        store_iaq_baseline_to_disk(baseline, now());
```

3.4. Read `in_iaq_baseline` to retrieve the current IAQ baseline. For the best performance and faster startup times, the current baseline needs to be persistently stored on the device. We recommend to save it once per hour and before shut-down. After boot-up it can be set again (see 3.3.)

```
$ cat in_iaq_baseline
1a2b3c4d
```

Note: The baseline value is returned without trailing newline and must be written back accordingly.
When the returned baseline value is all zeroes (`00000000`) the sensor's baseline is not initialized yet and its value must be discarded. Without restoring a baseline first (see 3.3.), a valid baseline is returned approximately 12 hours after initialization.

3.5. Write `set_iaq_baseline` to restore a previously retrieved IAQ baseline. The baseline value must be *exactly* as read from `in_iaq_baseline` (see 3.4.) and should only be set if it's less than one week old.

```
$ echo -n "1a2b3c4d" > set_iaq_baseline
```

Note: The baseline value is returned without trailing newline and must be written back accordingly, thus `echo -n` is used to suppress the trailing newline.

3.6. Read `gas_signals` to readout the gas signals

```
$ cat in_concentration_ethanol_raw
17477
$ cat in_concentration_h2_raw
13314
```

Note: The signals are returned as unitless *ticks* without baseline compensation. They are derived from the SGP multipixel architecture.

3.7. Read `in_selftest` 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 of the string *OK*.

```
$ cat in_selftest
OK
```

3.8. Write `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

On the kernel's sysfs interface, 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 driver does not handle floating point numbers.
Example usage to set the absolute humidity to 13.000 g/m³:

```
$ echo -n "13000" > set_absolute_humidity
```

Note: The humidity compensation is disabled by setting the value to 0.

REVISION HISTORY

Date	Version	Page(s)	Changes
February 2017	0.1.0	all	Initial Linux documentation
March 2017	0.2.0	3,4	Add baseline states documentation
March 2017	0.3.0	4	Add measure_test documentation
March 2017	1.4.0	4	Add pseudo code for baseline persistence
April 2017	1.5.0	all	SGP30
May 2017	1.6.0	5	Document gas_signals
August 2017	1.7.0	5	Add humidity compensation
October 2017	1.8.0	all	Capital T in AH formula, Add modinfo
April 2018	1.9.0	5	Remove sigal scaling
April 2018	2.0.0	all	Sync with updated v2 driver

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