Database - Air Quality

v0.1

Generated by Doxygen 1.9.6

Chapter 1

Designing a SQLite Database

This report presents the development of a C++ program that creates a SQLite database and stores measurement data in it. The program is designed to interact with home automation software, specifically Home Assistant, to display the stored data.

DTB documentation Embedded Systems Engineering

- Niels Urgert

1.1 General Information

Institution High School: Hogeschool van Arnhem en Nijmegen (HAN) Faculty: Academic Engineering and Automotive (AEA) Education: Embedded Systems Engineering (ESE)

1.2 Summary

The goal of this project was to design and implement a SQLite database for storing measurement data. The program utilizes the SQLite library to interact with the database and perform operations such as creating tables, inserting data, and querying the stored data. The stored measurement data can then be accessed and displayed in Home Assistant using the appropriate integrations.

1.3 Conceptual database design

The conceptual database design is represented by an Entity-Relationship (ER) diagram. The diagram illustrates the relationships between different entities and their attributes. In this case, the diagram depicts the relationships between sensors, entities, and metrics. Each sensor is associated with multiple entities, and each entity corresponds to multiple metrics.

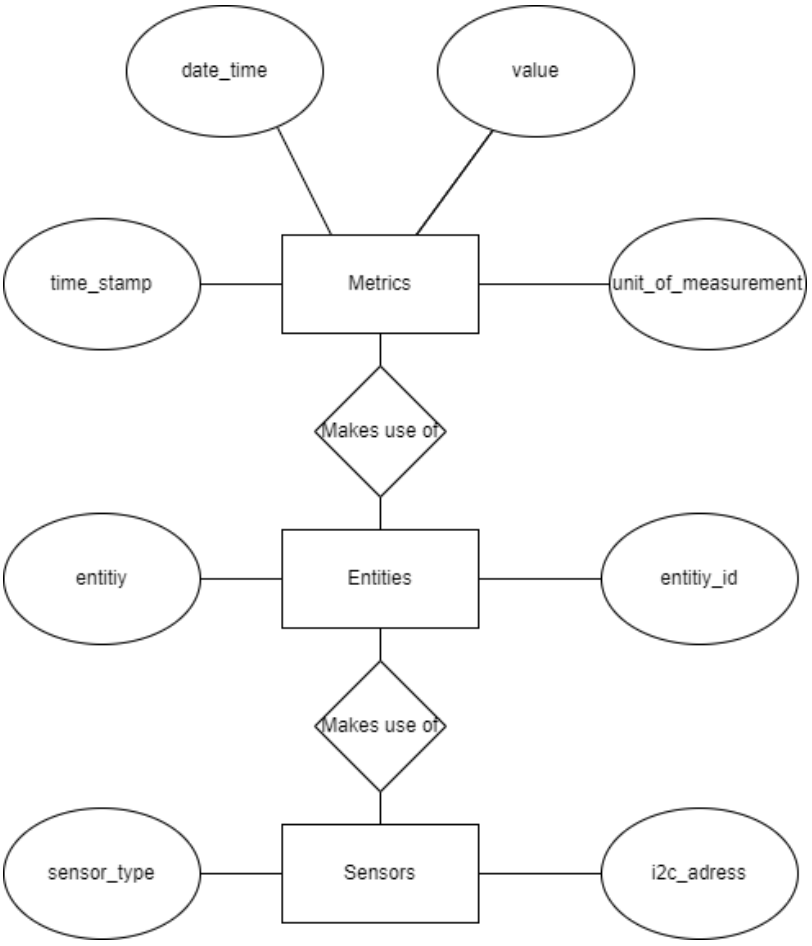


Figure 1.1 ER-Diagram

1.4 Logical database design

The logical database design defines the structure of the database tables and their attributes. The tables include 'sensors,' 'entities,' and 'metrics.' The 'sensors' table stores information about the sensors, such as their names and addresses. The 'entities' table represents the entities measured by the sensors, along with their corresponding sensor names. The 'metrics' table contains the actual measurement data, including the entity, timestamp, datetime, value, and unit.

Table: sensors
Primary key
Primary key: sensor_name

sensor_name	i2c_adress
HTS221	0x5f
LPS25H	0x5c
SGP30	0x58

Table: entities
Primary key: entity
Foreign key: sensor_name (references sensors.sensor_name)

entity	entity_id	*sensor_name
Temperature	sensor.sensehat_temp	HTS221
Humidity	sensor.sensehat_hum	HTS221
Pressure	sensor.sensehat_press	LPS25H
CO2eq	sensor.sgp30_CO2eq	SGP30
TVOC	sensor.sgp30_TVOC	SGP30

Table: metrics
Primary key: entity, time_stamp
Foreign key: entity (references entities.entity)

*entity	time_stamp	date_time	value	unit_of_measurement
Temperature	168.340.052.083.244	2023-05-06 21:15:20.832436	20.54	°C
Humidity	168.340.052.083.244	2023-05-06 21:15:20.832436	60.32	%RH
Pressure	168.340.052.083.244	2023-05-06 21:15:20.832436	1024.78	Pa
CO2eq	168.340.052.083.244	2023-05-06 21:15:20.832436	400	ppm
TVOC	168.340.052.083.244	2023-05-06 21:15:20.832436	600	ppb

Figure 1.2 Logical Design

1.5 Dashboard

To visualize and interact with the stored data, a dashboard is implemented using Home Assistant.

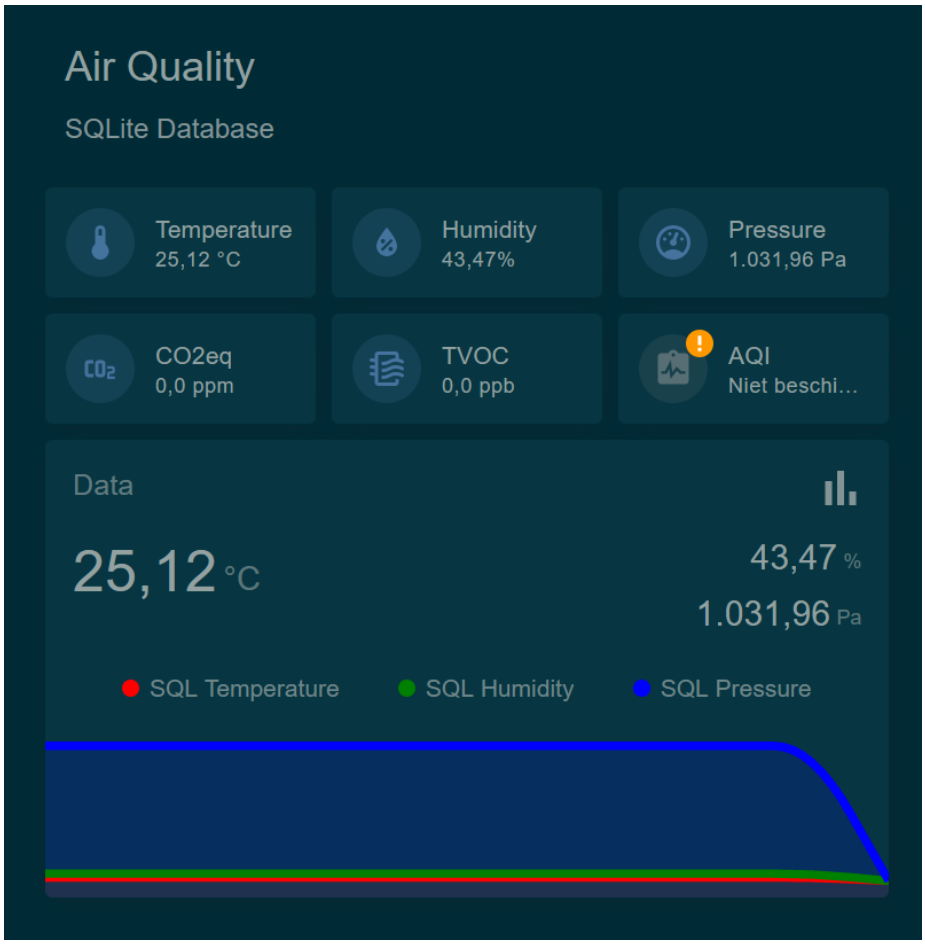


Figure 1.3 Dashboard

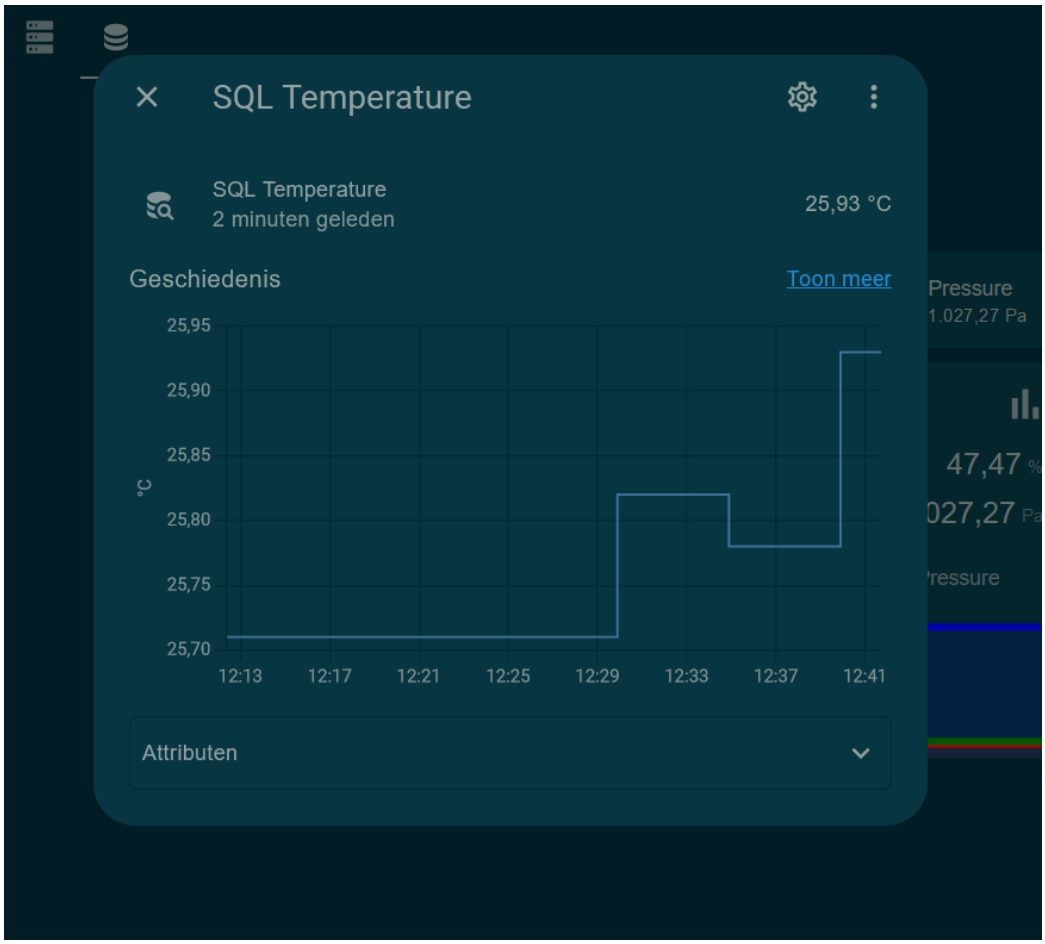


Figure 1.4 Dashboard Temperature Graph

1.6 Implementation

The database is implemented in Home Assistant. It uses the SQL database integration, which makes it possible to read out databases using queries.

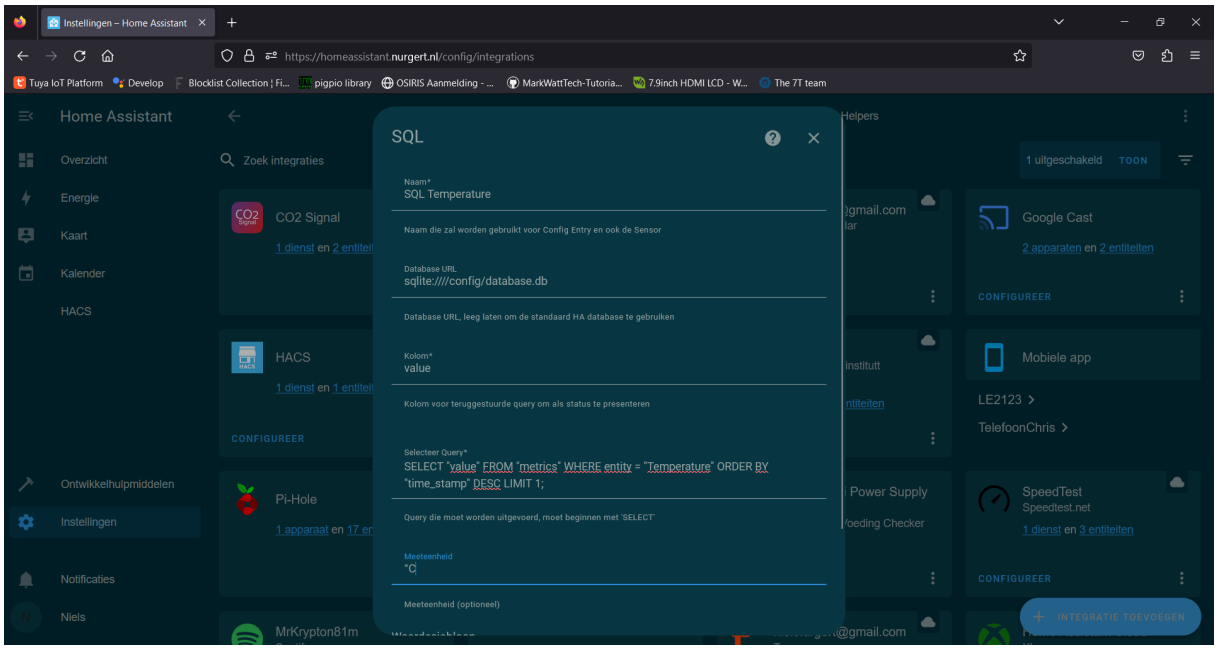


Figure 1.5 Dashboard

The figure below shows the working implementation. The most recent data is displayed in Home Assistant and matches the correct value in the database.

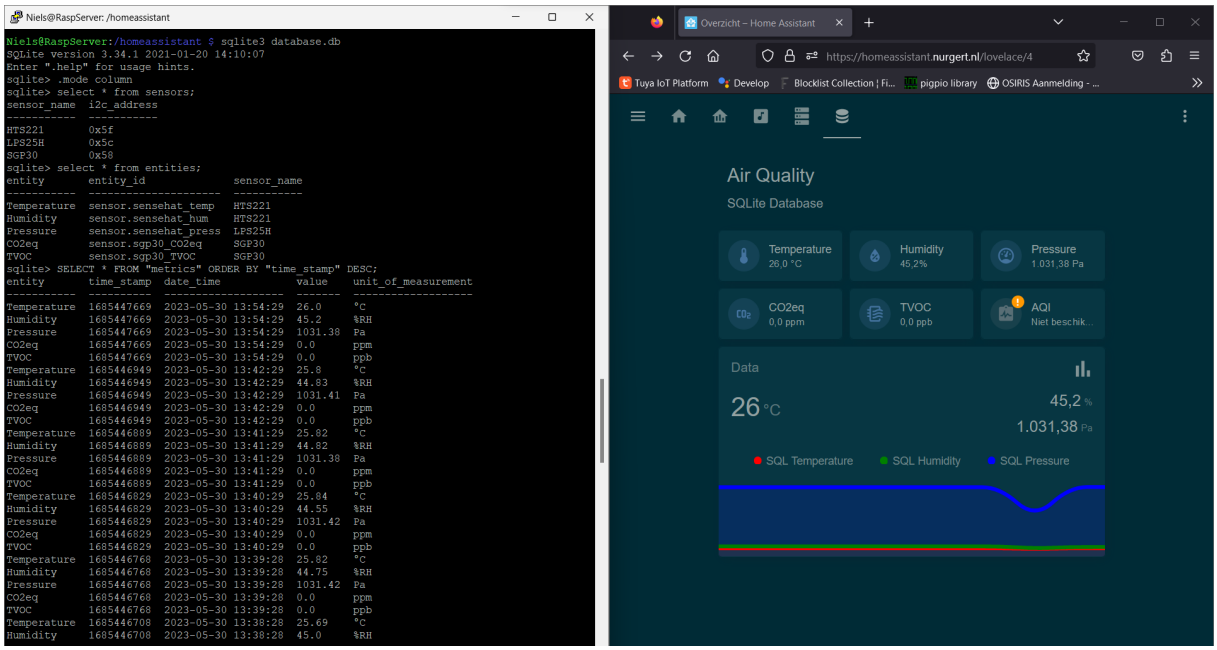


Figure 1.6 Dashboard

1.7 Appendix A: Configuration Dashboard

name: SQL Temperature database: sqlite:////config/database.db query: SELECT "value" FROM "metrics" WHERE entity = "Temperature" ORDER BY "time_stamp" DESC LIMIT 1; column: value Unit: ℃

name: SQL Humidity database: sqlite:////config/database.db query: SELECT "value" FROM "metrics" WHERE entity = "Humidity" ORDER BY "time_stamp" DESC LIMIT 1; column: value Unit: %

name: SQL Pressure database: sqlite:////config/database.db query: SELECT "value" FROM "metrics" WHERE entity = "Pressure" ORDER BY "time_stamp" DESC LIMIT 1; column: value Unit: Pa

name: SQL CO2eq database: sqlite:////config/database.db query: SELECT "value" FROM "metrics" WHERE entity = "CO2eq" ORDER BY "time_stamp" DESC LIMIT 1; column: value Unit: ppm

name: SQL TVOC database: sqlite:////config/database.db query: SELECT "value" FROM "metrics" WHERE entity = "TVOC" ORDER BY "time_stamp" DESC LIMIT 1; column: value Unit: ppb

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 air_quality Namespace Reference

Variables

- str database = "/homeassistant/database.db"
- sqlite3 con = sqlite3.connect(database)
- sqlite3 cur = con.cursor()
- sqlite3 checkForTableSensors = cur.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='sensors'").fetchone()
- list sensors
- sqlite3 checkForTableEntities = cur.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='entities'").fetchone()
- list entities
- sqlite3 checkForTableMetrics = cur.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='metrics'").fetchone()
- SenseHat sense = SenseHat()
- SenseHat temp = sense.get_temperature()
- SenseHat hum = sense.get_humidity()
- SenseHat press = sense.get_pressure()
- datetime date_time = datetime.now()
- datetime time_stamp = datetime.timestamp(date_time)
- list data

5.1.1 Variable Documentation

5.1.1.1 checkForTableEntities

```
sqlite3 air_quality.checkForTableEntities = cur.execute("SELECT name FROM sqlite_master WHERE
type='table' AND name='entities'").fetchone()
```

5.1.1.2 checkForTableMetrics

```
sqlite3 air_quality.checkForTableMetrics = cur.execute("SELECT name FROM sqlite_master WHERE
type='table' AND name='metrics'").fetchone()
```

5.1.1.3 checkForTableSensors

```
sqlite3 air_quality.checkForTableSensors = cur.execute("SELECT name FROM sqlite_master WHERE
type='table' AND name='sensors'").fetchone()
```

5.1.1.4 con

```
sqlite3 air_quality.con = sqlite3.connect(database)
```

5.1.1.5 cur

```
sqlite3 air_quality.cur = con.cursor()
```

5.1.1.6 data

```
list air_quality.data
```

Initial value:

```
00001 =  [
00002         ("Temperature", time_stamp, date_time, temp, "°C"),
00003         ("Humidity", time_stamp, date_time, hum, "%RH"),
00004         ("Pressure", time_stamp, date_time, press, "Pa"),
00005         #("CO2eq", time_stamp, date_time, co2, "ppm"),
00006         #("TVOC", time_stamp, date_time, tvoc, "ppb"),
00007     ]
```

5.1.1.7 database

```
str air_quality.database = "/homeassistant/database.db"
```

5.1.1.8 date_time

```
datetime air_quality.date_time = datetime.now()
```

5.1.1.9 entities

```
list air_quality.entities
```

Initial value:

```
00001 =  [
00002         ("Temperature", "sensor.sensehat_temp", "HTS221"),
00003         ("Humidity", "sensor.sensehat_hum", "HTS221"),
00004         ("Pressure", "sensor.sensehat_press", "LPS25H"),
00005         ("CO2eq", "sensor.sgp30_CO2eq", "SGP30"),
00006         ("TVOC", "sensor.sgp30_TVOC", "SGP30"),
00007     ]
```

5.1.1.10 hum

```
round air_quality.hum = sense.get_humidity()
```

5.1.1.11 press

```
round air_quality.press = sense.get_pressure()
```

5.1.1.12 sense

```
SenseHat air_quality.sense = SenseHat()
```

5.1.1.13 sensors

```
list air_quality.sensors
```

Initial value:

```
00001 =  [
00002         ("HTS221", "0x5f"),
00003         ("LPS25H", "0x5c"),
00004         ("SGP30", "0x58"),
00005     ]
```

5.1.1.14 temp

```
round air_quality.temp = sense.get_temperature()
```

5.1.1.15 time_stamp

```
datetime air_quality.time_stamp = datetime.timestamp(date_time)
```

Chapter 6

Class Documentation

6.1 Sensors Class Reference

Public Member Functions

- Sensors ()
- void initialize ()
- void readSenseHATData ()
- void readSGP30Data ()
- void cleanup ()

Public Attributes

- double temp
- double hum
- double press
- uint16_t eCO2
- uint16_t TVOC

6.1.1 Constructor & Destructor Documentation

6.1.1.1 Sensors()

```
Sensors::Sensors ( )  [inline]
```

6.1.2 Member Function Documentation

6.1.2.1 cleanup()

```
void Sensors::cleanup ( )  [inline]
```

6.1.2.2 initialize()

```
void Sensors::initialize ( )  [inline]
```

6.1.2.3 readSenseHATData()

```
void Sensors::readSenseHATData ( )  [inline]
```

6.1.2.4 readSGP30Data()

```
void Sensors::readSGP30Data ( )  [inline]
```

6.1.3 Member Data Documentation

6.1.3.1 eCO2

```
uint16_t Sensors::eCO2
```

6.1.3.2 hum

```
double Sensors::hum
```

6.1.3.3 press

```
double Sensors::press
```

6.1.3.4 temp

```
double Sensors::temp
```

6.1.3.5 TVOC

```
uint16_t Sensors::TVOC
```

The documentation for this class was generated from the following file:

- air_quality.cpp

6.2 SQLManager Class Reference

Public Member Functions

- SQLManager (const std::string &dbPath)
- ~SQLManager ()
- void createSensorsTable ()
- void insertSensors (const std::string &sensor_name, const std::string &i2c_address)
- void createEntitiesTable ()
- void insertEntities (const std::string &entity, const std::string &entity_id, const std::string &sensor_name)
- void createMetricsTable ()
- void insertMetrics (const std::string &entity, int time_stamp, const std::string &date_time, double value, const std::string &unit)
- void deleteMetrics (int limit)

6.2.1 Constructor & Destructor Documentation

6.2.1.1 SQLManager()

```
SQLManager::SQLManager (
            const std::string & dbPath )  [inline]
```

6.2.1.2 ~SQLManager()

```
SQLManager::~SQLManager ( )  [inline]
```

6.2.2 Member Function Documentation

6.2.2.1 createEntitiesTable()

```
void SQLManager::createEntitiesTable ( )  [inline]
```

6.2.2.2 createMetricsTable()

```
void SQLManager::createMetricsTable ( )  [inline]
```

6.2.2.3 createSensorsTable()

```
void SQLManager::createSensorsTable ( )  [inline]
```

6.2.2.4 deleteMetrics()

```
void SQLManager::deleteMetrics (
            int limit )  [inline]
```

6.2.2.5 insertEntities()

```
void SQLManager::insertEntities (
            const std::string & entity,
            const std::string & entity_id,
            const std::string & sensor_name )  [inline]
```

6.2.2.6 insertMetrics()

```
void SQLManager::insertMetrics (
            const std::string & entity,
            int time_stamp,
            const std::string & date_time,
            double value,
            const std::string & unit )  [inline]
```

6.2.2.7 insertSensors()

```
void SQLManager::insertSensors (
            const std::string & sensor_name,
            const std::string & i2c_address )  [inline]
```

The documentation for this class was generated from the following file:

- air_quality.cpp

6.3 Time Class Reference

Public Member Functions

- Time ()
- void update ()
- int getTimestamp () const
- std::string getDateTime () const

6.3.1 Constructor & Destructor Documentation

6.3.1.1 Time()

```
Time::Time ( )  [inline]
```

6.3.2 Member Function Documentation

6.3.2.1 getDateTime()

```
std::string Time::getDateTime ( ) const  [inline]
```

6.3.2.2 getTimestamp()

```
int Time::getTimestamp ( ) const  [inline]
```

6.3.2.3 update()

```
void Time::update ( )  [inline]
```

The documentation for this class was generated from the following file:

- air_quality.cpp

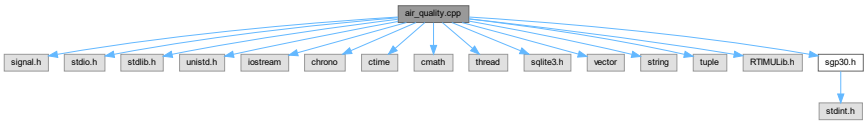
Chapter 7

File Documentation

7.1 air_quality.cpp File Reference

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <iostream>
#include <chrono>
#include <ctime>
#include <cmath>
#include <thread>
#include <sqlite3.h>
#include <vector>
#include <string>
#include <tuple>
#include "RTIMULib.h"
#include "sgp30.h"
```

Include dependency graph for air_quality.cpp:



Classes

- class Sensors
- class Time
- class SQLManager

Functions

- void signalHandler (int signal)
- int main ()

Variables

- const std::string database = "/homeassistant/database.db"
- bool running = true

7.1.1 Function Documentation

7.1.1.1 main()

```
int main ( )
```

7.1.1.2 signalHandler()

```
void signalHandler (
            int signal )
```

7.1.2 Variable Documentation

7.1.2.1 database

```
const std::string database = "/homeassistant/database.db"
```

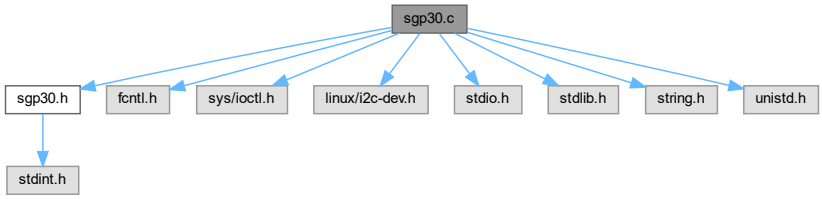
7.1.2.2 running

```
bool running = true
```

7.2 sgp30.c File Reference

```
#include "sgp30.h"
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/i2c-dev.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

Include dependency graph for sgp30.c:



Functions

- int SGP30_init (const char *i2c_adaptor)
- int SGP30_deinit (void)
- int SGP30_measure_air_quality (uint16_t *eCO2_out, uint16_t *TVOC_out)
- int SGP30_measure_raw_signals (uint16_t *h2_out, uint16_t *ethanol_out)
- int SGP30_get_baseline (uint16_t *eCO2_out, uint16_t *TVOC_out)
- int SGP30_set_baseline (const uint16_t eCO2_in, const uint16_t TVOC_in)
- int SGP30_set_humidity (uint16_t humidity_in)
- int SGP30_measure_test (void)
- int SGP30_get_feature_set_version (uint16_t *version_out)
- int SGP30_get_serial_id (uint64_t *serial_id)

7.2.1 Function Documentation

7.2.1.1 SGP30_deinit()

```
int SGP30_deinit (
            void  )
```

7.2.1.2 SGP30_get_baseline()

```
int SGP30_get_baseline (
            uint16_t * eCO2_out,
            uint16_t * TVOC_out )
```

7.2.1.3 SGP30_get_feature_set_version()

```
int SGP30_get_feature_set_version (
            uint16_t * version_out )
```

7.2.1.4 SGP30_get_serial_id()

```
int SGP30_get_serial_id (
            uint64_t * serial_id )
```

7.2.1.5 SGP30_init()

```
int SGP30_init (
            const char * i2c_adaptor )
```

7.2.1.6 SGP30_measure_air_quality()

```
int SGP30_measure_air_quality (
            uint16_t * eCO2_out,
            uint16_t * TVOC_out )
```

7.2.1.7 SGP30_measure_raw_signals()

```
int SGP30_measure_raw_signals (
            uint16_t * h2_out,
            uint16_t * ethanol_out )
```

7.2.1.8 SGP30_measure_test()

```
int SGP30_measure_test (
            void  )
```

7.2.1.9 SGP30_set_baseline()

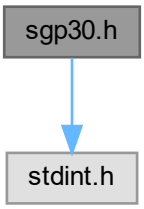
```
int SGP30_set_baseline (
            const uint16_t eCO2_in,
            const uint16_t TVOC_in )
```

7.2.1.10 SGP30_set_humidity()

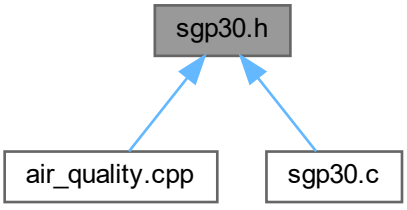
```
int SGP30_set_humidity (
            uint16_t humidity_in )
```

7.3 sgp30.h File Reference

`#include <stdint.h>`
Include dependency graph for sgp30.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define DEBUG_PRINT 0
- #define debug_print(fd, fmt, ...) do { if (DEBUG_PRINT) fprintf(fd, fmt, __VA_ARGS__); } while (0)
- #define SGP30_ADDRESS 0x58

Enumerations

- enum Error {
  NO_ERROR , ERROR_TIME , ERROR_DRIVER , ERROR_CRC ,
  ERROR_INVAL , ERROR_I2C }

Functions

- int SGP30_init (const char *i2c_adaptor)
- int SGP30_deinit (void)
- int SGP30_measure_air_quality (uint16_t *eCO2_out, uint16_t *TVOC_out)
- int SGP30_measure_raw_signals (uint16_t *h2_out, uint16_t *ethanol_out)
- int SGP30_get_baseline (uint16_t *eCO2_out, uint16_t *TVOC_out)
- int SGP30_set_baseline (const uint16_t eCO2_in, const uint16_t TVOC_in)
- int SGP30_set_humidity (const uint16_t humidity_in)
- int SGP30_measure_test (void)
- int SGP30_get_feature_set_version (uint16_t *version_out)
- int SGP30_get_serial_id (uint64_t *serial_id)

7.3.1 Macro Definition Documentation

7.3.1.1 DEBUG_PRINT

```
#define DEBUG_PRINT 0
```

7.3.1.2 debug_print

```
#define debug_print(
              fd,
              fmt,
              ...  )   do { if (DEBUG_PRINT) fprintf(fd, fmt, __VA_ARGS__); } while (0)
```

7.3.1.3 SGP30_ADDRESS

```
#define SGP30_ADDRESS 0x58
```

7.3.2 Enumeration Type Documentation

7.3.2.1 Error

```
enum Error
```

Enumerator

NO_ERROR	
ERROR_TIME	
ERROR_DRIVER	
ERROR_CRC	
ERROR_INVAL	
ERROR_I2C	

7.3.3 Function Documentation

7.3.3.1 SGP30_deinit()

```
int SGP30_deinit (
            void  )
```

7.3.3.2 SGP30_get_baseline()

```
int SGP30_get_baseline (
            uint16_t * eCO2_out,
            uint16_t * TVOC_out )
```

7.3.3.3 SGP30_get_feature_set_version()

```
int SGP30_get_feature_set_version (
            uint16_t * version_out )
```

7.3.3.4 SGP30_get_serial_id()

```
int SGP30_get_serial_id (
            uint64_t * serial_id )
```

7.3.3.5 SGP30_init()

```
int SGP30_init (
            const char * i2c_adaptor )
```

7.3.3.6 SGP30_measure_air_quality()

```
int SGP30_measure_air_quality (
            uint16_t * eCO2_out,
            uint16_t * TVOC_out )
```

7.3.3.7 SGP30_measure_raw_signals()

```
int SGP30_measure_raw_signals (
            uint16_t * h2_out,
            uint16_t * ethanol_out )
```

7.3.3.8 SGP30_measure_test()

```
int SGP30_measure_test (
            void  )
```

7.3.3.9 SGP30_set_baseline()

```
int SGP30_set_baseline (
            const uint16_t eCO2_in,
            const uint16_t TVOC_in )
```

7.3.3.10 SGP30_set_humidity()

```
int SGP30_set_humidity (
            const uint16_t humidity_in )
```

7.4 sgp30.h

Go to the documentation of this file.

```
00001 #ifndef SGP30
00002 #define SGP30
00003
00004 #include <stdint.h>
00005
00006 // Print function that only prints if DEBUG is defined
00007 #ifdef DEBUG
00008 #define DEBUG_PRINT 1
00009 #else
00010 #define DEBUG_PRINT 0
00011 #endif
00012 #define debug_print(fd, fmt, ...) \
00013             do { if (DEBUG_PRINT) fprintf(fd, fmt, __VA_ARGS__); } while (0)
00014
00015 enum Error {
00016   NO_ERROR,
00017   ERROR_TIME,           // Timeout (in I2C, due to clock stretching)
00018   ERROR_DRIVER,         // Driver failed to init
00019   ERROR_CRC,            // Checksum failed
00020   ERROR_INVAL,          // Invalid argument
00021   ERROR_I2C             // I2C driver failed to read or write data
00022 };
00023
00024 // Chip defines
00025 #define SGP30_ADDRESS 0x58
00026
00027 // Set up and tear down SGP30 I2C interface
00028 int SGP30_init(const char *i2c_adaptor);
00029 int SGP30_deinit(void);
00030
00031 // Fetch data from SGP30
00032 int SGP30_measure_air_quality(uint16_t *eCO2_out,
00033                               uint16_t *TVOC_out);
00034 int SGP30_measure_raw_signals(uint16_t *h2_out,
00035                               uint16_t *ethanol_out);
00036 int SGP30_get_baseline(uint16_t *eCO2_out,
00037                        uint16_t *TVOC_out);
00038
00039 // Set data in SGP30
00040 int SGP30_set_baseline(const uint16_t eCO2_in,
00041                        const uint16_t TVOC_in);
00042 int SGP30_set_humidity(const uint16_t humidity_in);
00043
00044 // Data about SGP30
00045 int SGP30_measure_test(void);
00046 int SGP30_get_feature_set_version(uint16_t *version_out);
00047 int SGP30_get_serial_id(uint64_t *serial_id);
00048 #endif
```

7.5 mainpage.md File Reference

7.6 air_quality.py File Reference

Namespaces

- namespace air_quality

Variables

- str air_quality.database = "/homeassistant/database.db"
- sqlite3 air_quality.con = sqlite3.connect(database)
- sqlite3 air_quality.cur = con.cursor()
- sqlite3 air_quality.checkForTableSensors = cur.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='sensors'").fetchone()
- list air_quality.sensors

- sqlite3 air_quality.checkForTableEntities = cur.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='entities'").fetchone()
- list air_quality.entities
- sqlite3 air_quality.checkForTableMetrics = cur.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='metrics'").fetchone()
- SenseHat air_quality.sense = SenseHat()
- SenseHat air_quality.temp = sense.get_temperature()
- SenseHat air_quality.hum = sense.get_humidity()
- SenseHat air_quality.press = sense.get_pressure()
- datetime air_quality.date_time = datetime.now()
- datetime air_quality.time_stamp = datetime.timestamp(date_time)
- list air_quality.data

Index