SMARTAQNET-STATIONARY

Smart Air Quality Network Crowdsensor Firmware

Generated by Doxygen

Thu Mar 19 2020

1 Smart Air Quality Network (SMARTAQNET) 1

1.1 Introduction . . . 1

1.1.1 OLED Display Usage . . . 2

1.1.2 Error Decoding . . . 5

1.1.3 NVS Usage . . . 6

1.1.4 SDS011 Dust Sensor Usage . . . 6

1.1.5 BME280 Sensor Usage . . . 6

1.1.6 WiFi USage . . . 6

1.1.7 Configuration Page . . . 7

1.1.8 MDNS . . . 9

1.1.9 Daily Reboot . . . 9

1.1.10 Sensing and Sending Intervals . . . 9

1.1.11 Modelling and Creating Entities . . . 9

1.1.12 FlowCharts . . . 10

1.1.13 Important Constants . . . 10

1.2 Pre-requisites . . . 11

1.3 Hardware . . . 11

1.3.1 Peripherals . . . 11

1.3.2 Arduino IDE . . . 12

1.3.2.1 Pyserial . . . 12

1.3.2.2 ESP32 Arduino core . . . 12

1.3.2.3 Required Libraries . . . 12

1.3.3 Firmware Uploading . . . 13

1.4 Wirings . . . 13

1.5 Known Bugs . . . 14

1.5.1 Some Observations . . . 15

1.6 Change Log . . . 16

1.6.1 FIRMWARE_VERSION 0.8.6.1 onwards . . . 16

1.6.2 FIRMWARE_VERSION 0.8.6 and older . . . 18

1.7 Quick Debugging . . . 19

1.8 Author(s) . . . 20

1.9 License . . . 20

1.9.1 Acknowledgments . . . 20

1.9.2 MIT License . . . 21

1.9.3 U8g2lib License . . . 21

1.9.4 Espressif Systems License . . . 21

2 Hierarchical Index 23

2.1 Class Hierarchy . . . 23

3 Class Index 25

3.1 Class List . . . 25

4 File Index **27**

4.1 File List . . . 27

5 Class Documentation **29**

5.1 myAutoConnect Class Reference . . . 29

5.1.1 Detailed Description . . . 29

5.1.2 Constructor & Destructor Documentation . . . 29

5.1.2.1 myAutoConnect() . . . 30

5.1.3 Member Function Documentation . . . 30

5.1.3.1 _waitForConnect() . . . 30

5.1.3.2 begin() [1/2] . . . 30

5.1.3.3 begin() [2/2] . . . 30

6 File Documentation **33**

6.1 esp32-stationary/create_json.h File Reference . . . 33

6.1.1 Detailed Description . . . 33

6.1.2 Function Documentation . . . 34

6.1.2.1 json_madavi() . . . 34

6.1.2.2 json_RHT() . . . 34

6.1.2.3 json_sds() . . . 35

6.1.3 Variable Documentation . . . 35

6.1.3.1 use_bme280 . . . 35

6.1.3.2 use_dht22 . . . 35

6.2 esp32-stationary/display.h File Reference . . . 36

6.2.1 Detailed Description . . . 36

6.2.2 Function Documentation . . . 37

6.2.2.1 displayQRCode() . . . 37

6.2.2.2 generateQRCode() . . . 37

6.3 esp32-stationary/esp32-stationary.ino File Reference . . . 37

6.3.1 Detailed Description . . . 38

6.3.2 Function Documentation . . . 38

6.3.2.1 loop() . . . 38

6.3.2.2 setup() . . . 38

6.4 esp32-stationary/main.h File Reference . . . 39

6.4.1 Detailed Description . . . 53

6.4.2 Macro Definition Documentation . . . 53

6.4.2.1 BASE_URL . . . 53

6.4.3 Function Documentation . . . 53

6.4.3.1 createEntities() . . . 53

6.4.3.2 createEntity() . . . 53

6.4.3.3 createOrPatchEntities() . . . 54

6.4.3.4 createOrUpdateEntities() . . . 54

6.4.3.5 getEntity() . . . 55

6.4.3.6 getEntityResponse() . . . 55
6.4.3.7 patchEntity() . . . 55
6.4.3.8 postObservation() . . . 57
6.4.3.9 toHEXSHA() . . . 57
6.4.4 Variable Documentation . . . 58
6.4.4.1 esp32_chipid . . . 58
6.4.4.2 http_error . . . 58
6.4.4.3 license_type . . . 58
6.4.4.4 use_bme280 . . . 59
6.4.4.5 use_dht22 . . . 59
6.5 esp32-stationary/myAutoConnect.h File Reference . . . 59
6.5.1 Detailed Description . . . 60
6.6 esp32-stationary/nvs_storage.h File Reference . . . 60
6.6.1 Detailed Description . . . 61
6.6.2 Function Documentation . . . 61
6.6.2.1 get_integer() . . . 61
6.6.2.2 get_value() . . . 61
6.6.2.3 set_integer() . . . 62
6.6.2.4 set_value() . . . 62
6.7 esp32-stationary/send_data.h File Reference . . . 63
6.7.1 Detailed Description . . . 63
6.7.2 Function Documentation . . . 63
6.7.2.1 sendData() . . . 63
6.7.3 Variable Documentation . . . 64
6.7.3.1 esp32_chipid . . . 64
6.7.3.2 use_bme280 . . . 64
6.7.3.3 use_dht22 . . . 64
6.8 esp32-stationary/sensor.h File Reference . . . 64
6.8.1 Detailed Description . . . 65
6.9 esp32-stationary/timer_definitions.h File Reference . . . 65
6.9.1 Detailed Description . . . 66
6.9.2 Function Documentation . . . 66
6.9.2.1 resetModule() . . . 66

Index **67**

Chapter 1

Smart Air Quality Network (SMARTAQNET)

1.1 Introduction

SMARTAQNET project is aimed at development of a close-meshed network of local fine-dust data, which can be fed and used well by general public. Further details can be checked at: http://www.smartaq.net.
This script uses a ESP32 HELTEC Board, a SDS011 particulate matter sensor and a BME280 sensor or a DHT22 sensor. The Fig-1 shows the main board having different sensors connected together. The ESP32 is also refered to as crowdsensor. After power on, it opens a Wifi access point, default Password is 12345678. The Wifi access point can be searched by a device (Smartphone, Tablet, Laptop or similar) using the ID shown on the OLED display. A webpage (also called "Captive Portal") opens after connection to this AP with Smartphone. If no webpage is opening automatically, it can be opend by typing "http://192.168.244.1" into the browser. By using this webpage, the crowdsensor can be connected to any of the available Wifi networks. The crowdsensors configuration page can then be accessed using http://[localIP]/ or using http://[SSD-ID].local/ e.g., http://192.168.0.39 or http://181712.local. The local IP and the SSD-ID are both printed on the screen of the crowd sensor. The location address, the thermal sensor type, the ID of the dust sensor SDS011, the owner of the data and more can be configure using the configuration page. The firware version of the code is hard-coded by the constant (FIRMWARE_VERSION). The detailed debugging can be controlled for the firmware using the constant RELEASE. Following is the details for differnt features and steps to start:

- Install the arduino, board and required libraies as explained in Arduino IDE, ESP32 Arduino core, Required Libraries
- Connect the hardware as explained in Wirings, Hardware and Peripherals
- Configure the constants in the code as explained in Important Constants.
- Compile and upload the cdes as explained in Firmware Uploading and Pyserial
- Configure the WiFi and other parameters as explained in WiFi USage and Configuration Page
- If everything is installed and configured perfectly, there should be no error i.e., E0 as explained in Error Decoding
- If there is any errors please look into error or known bug section as explained in Error Decoding and Known Bugs
- For detailed debugging uncomment RELEASE constant. Some quick trick will also save the time as shown in Quick Debugging

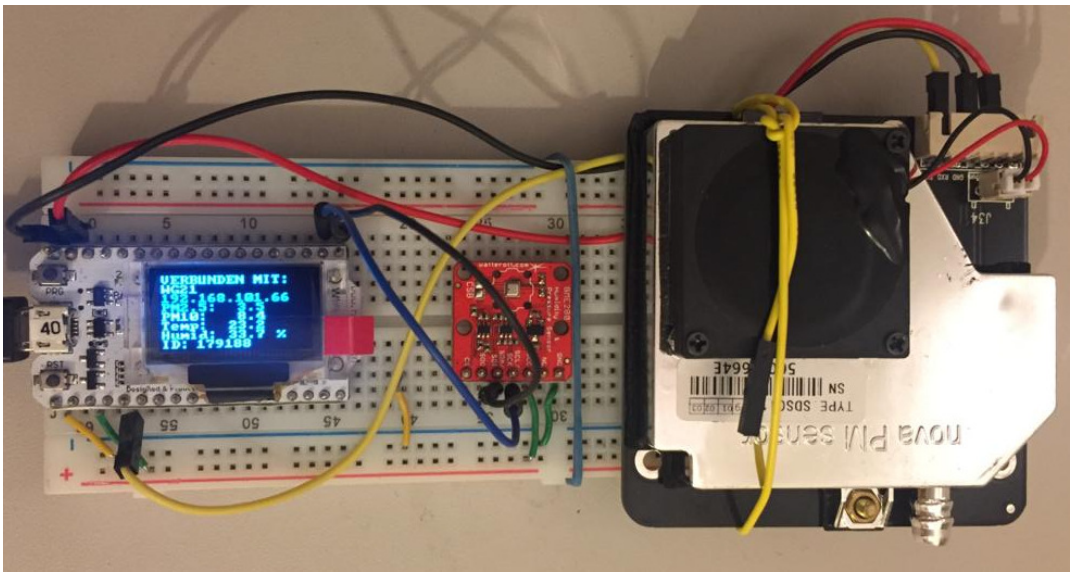


Figure 1.1 Sensor Board

1.1.1 OLED Display Usage

Built-in OLED, a SSD1306 based display, is used to display different type of information, messages, images and QR code. The display is connected using I2C interface using u8g2 library.

- Hardware PINs: RST (16) is reset, OLED_SDA (4) is SDA and OLED_SCL (15) is SCL pin of the display and OLED_ADDR (0x3c) is the I2C address of the display.
- ICONS and QR Codes: at the startup it displays logos and then a QR code
- Which Screens to display and their meaning: There are two main screens. The Fig-2 shows the main screen while the Fig-3 shows second screen. During the execution, different screens are:

1. When the firmware is started, OLED displays and it tries to search the previous configured WiFi as in Fig-2. The last line shows the hardware ID of the ESP32.

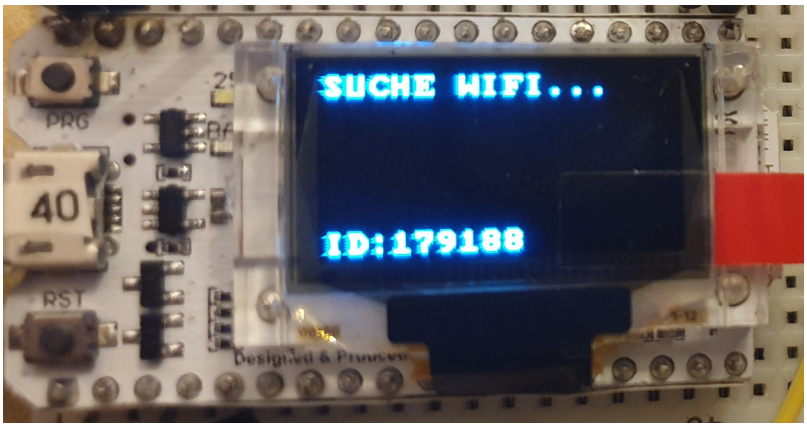


Figure 1.2 OLED Display: Searching Wifi

2. When the firmware could not find previous WiFi, it opens an AP and shows the screen as in Fig-3. It shows the SSID and Password for AP and its IP address.

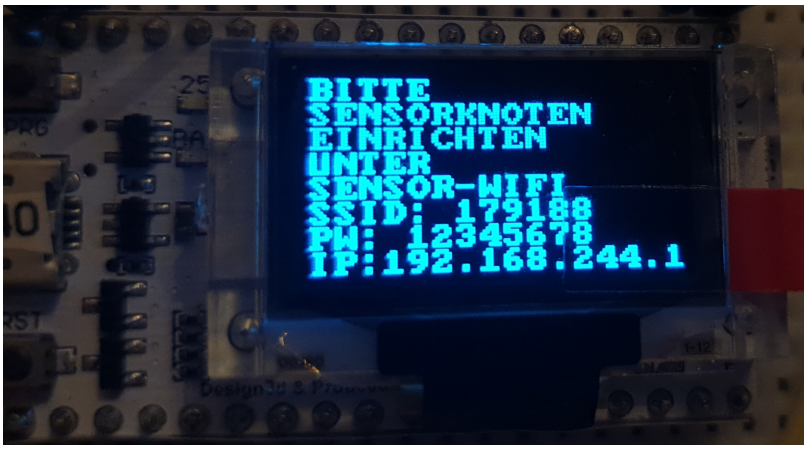


Figure 1.3 OLED Display: Access Point WiFi

3. When the WiFi is connected, it shows that it is waiting for the time to be synchronized, as in Fig-4.

Figure 1.4 OLED Display: NTP Time Sync

4. When the WiFi is connected, temperature and dust sensor are attached and configured, the screen in Fig-5 shows the sensed values, ID and static IP of the ESP32. ER shows the error code in hexadecimal. This is explained in the section below. ER of 0xC0 means the dust and temperature sensors are not connected.

Figure 1.5 OLED Display Screen-1

5. When the WiFi is connected, temperature and dust sensor are not attached, the screen in Fig-6 shows the messages (i.e., FEINSTAUBSENSOR:Bitte verbinden and TEMPERATURSENSOR:Bitte

verbinden, ID and static IP of the ESP32. When temperature and dust sensor are attached but temperature is not configured, screen shows the messages FEINSTAUBSENSOR:Bitte verbinden and TEM↩ PERATURSENSOR:Bitte auswaehlen.

Figure 1.6 OLED Display: Screen-1 Error

6. When the WiFi is not connected or it is disconnected or time is not synchronized during the second trial, Fig-7, shows the messages and ID of the ESP32. And it restarts.

Figure 1.7 OLED Display: WiFi or NTP Error

7. The Fig-8 screen shows some statistical and informative messages, with the intervals of (SENSOR_INTERVAL) in non-release mode.

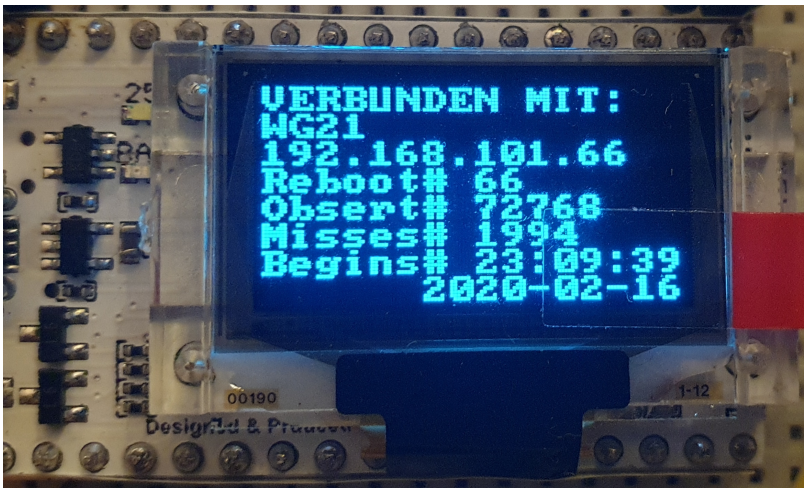


Figure 1.8 OLED Display: Screen-2

The statistical screen contains the following. These are tentative, we can change the words and information.

(a) **Reboot** is indicating the total number of reboots. This will help us how many normal boots (@4AM) and how many accidental boots happened? Saved in the eeprom.

(b) **Obsert** is the total number of the observations since the beginning. Saved in the eeprom.

(c) **Misses** is the total number of the observations that are not posted correctly i.e., negative HTTP status codes. Saved in the eeprom.

(d) **E** (i.e. Errors) is the encoding of the errors in hexadecimal. (see below). Not saved in the eeprom.

(e) **Begins** is the timestamp of the first observation at the time entities were created for the first time. Saved in the eeprom.

8. The QR screen is show at two occasions. First when there is no wifi configured and ESP32 opens an AP, it shows a screen (Fig-3) and a QR code alternatively with an interval of SHOW_QR_INTERVAL. Other screen after the WiFi is configured, is continuously with the text (Fig-5) of and QR code with an interval of SENSOR_INTERVAL.

1.1.2 Error Decoding

On the last line of the display, at the right most column indicated the different errors which are encoded in hexadecimal. If there is any error, corresponding bit is set to high. The error encoding is explained below in the Table 1.1:

Table 1.1 Error Enconding

Bit Location	7	6	5	4	3	2	1	0
Error Type	SDS HW Error	BME HW Error	BME Select Error	NTP First Error	mDNS Error	Create Entities Error	Location Missing Error	HTTP Server Error

1. **E00** indicates "NO ERROR".
2. **E01** indicates "HTTP Server Error", this happens when the HTTP Server is not responding.

3. **E02** indicates "Location Missing Error", this happens when the location is not configured on the configuration page.
4. **E04** indicates "Create Entities Error", this happens when the entities are not created properly.
5. **E08** indicates "mDNS Error", this happens when the mDNS has not started properly.
6. **E10** indicates "NTP First Error", this happens when the time is not sync for the first time.
7. **E20** indicates "BME Select ERROR", this happens when the Temperature sensor is not configured from the configuration page.
8. **E40** indicates "BME HW ERROR", this happens when the Temperature sensor is not connected.
9. **E80** indicates "SDS HW ERROR", this happens when the dust sensor is not given.
   There might b combination of above errors, for Example:
   (a) Error E03 means: The comobination of E01 and E02.
   (b) Error E21 means: The temperature sensor is not selected and the location is not configured.
   (c) Error E01 means: There is a HTTP Server error.

1.1.3 NVS Usage

NVS flash storage is used to save different information permanently. It saves latitude as "latitude", longitude as "longitude", SDS011 ID as "sds011id", human readable location as "address" license name as "data_license, license legal notice as "data_owner", license url as "data_url", error codes as "warning", starting timestamp as "tom", total number of observations posted as "tObservation", total number of errorneous observations as "nObservation", total number of reset counter as "rCounter". The configurational parameteres are updated in the nvs when there is change from HTML configuration page, while the statistical parameters are updated during different execution scenarios.

1.1.4 SDS011 Dust Sensor Usage

- Hardware Pins: it uses transmit SDS011_TXD (13) and recieve SDS011_RXD (17) pins
- SERIAL Port: it uses the HardwareSerial port 2 for SDS_SERIAL

1.1.5 BME280 Sensor Usage

- Hardware Pins: it uses SDA BME_SDA (21) and SCL BME_SCL (22) pins
- I2C Port: to avoid any mismatch it uses Wire1 (default is Wire0) in bme.begin(BME_ADDR, &Wire1)
- I2C address: BME_ADDR as 0x77 for Red, 0x76 for Purple (chinese)

1.1.6 WiFi USage

Firstly, on power up ESP32 tries to connect to earlier configured WiFi untill (WIFI_REBOOT_TIMEOUT), if not found it opens up a new access point. Any client can be connected using SSID and Passcode shown on the screen. If some connects to access point a configuration pages is automatically opened, if not, one can open it using http://[localIP]/ or using http://[SSD-ID].local/. Using this configuration page, any avaialable WiFi arround you can be connected with using respective credentials. The ESP32 waits for some client to be connected untill (APMODE_REBOOT_TIMEOUT), otherwise it will restart and repeate the process. If some client is connected within timeout, ESP32 tries to synchronised with the network time untill (NTP_TIMEOUT_WAIT), otherwise it skips this synchronization for the second trial later. Later, before creating server entities it tries to synchronised with the network time untill (NTP_TIMEOUT_WAIT) and restart if not seccessful.

1.1.7 Configuration Page

A HTML configuration page is prepared using AutoConnect library without using Pagebuilder library. Different elements are defined like: cfgtitle (ACText), cfgScript (ACElement), cfgBody (ACElement), cfgData (ACText), cfg↩ Body2 (ACElement) which are combined using AutoConnectAux. The function handleConfig() is used to handle the "/config" response of Webserver::on(...) after button submission, the function handleRoot() is used to handle the / response of AutoConnect::on(...), the function otaUpdateHandler() is used to handle the /otaUpdate HTTP_POST response of Webserver::on(...), the function updateHandler() is used to handle the /update HTTP_POST response of Webserver::on(...), the function updateHandlerResponse() is used to handle the /update HTTP_GET response of Webserver::on(...). HTML script otaUpdate defines OTA uploading webpage using AutoConnectAux. The HT↩ ML configuration element "inputAddress" defines to input the human readable address, "inputLon" defines to input location's longitude, "inputLat" defines to input location's latitude, "inputRhtsensor" defines to select the temperature sensor out of "default", "BME280" and "DHT22"; "inputSds011id" defines to input the printed SDS011 serial number, "inputLicense" defines to select license type between "CC0 1.0" or "CC BY 4.0" and "inputOwner" defines to input the legal_notice of the license owner. The license "CC0 1.0" and "CC BY 4.0" points to " #data1_url_cc0 " and " #data1_url_ccby " respectively.
The Fig-9 shows an example fo a configuration page. The configuration page has following information from top-bottom:

1. **Menus**: There are different menues which performs different actions
2. **Title**: It indicates the title of page and the firmware version.
3. **Map**: It indicates the google map of the location selected. Or you can choose any location from this map using the the cursor.
4. **Address**: One can input the human readable address for the location.
5. **Longitude**: One can input the longitude of the location.
6. **Latitude**: One can input the latitude of the location.
7. **Sensors**: One can chose the temperature sensor i.e., none, BME22, or DHT22.
8. **SDS011 ID**: One can input the hard-coded serial number from the sds011 dust sensor.
9. **Licese Name**: One can chose the license type here.
10. **Legal Notice**: One can chose the license owner of the data.
11. **Submit (Speichern)**: Button to submit the new configuration to ESP32, which will reset it and new entities will be created again.

The menu has the following items:

1. **HOME**: is the home of the configuration page. It shows the different parameters of the current connection.
2. **Update**: is used to update the firmware over OTA.
3. **Configure**: is the configuration page as shown in Fig-9.
4. **Reset...**: this will reset the board.
5. **Disconnect**: will disconnect the system from the current WiFi.
6. **Open SSIDs**: shows all the used/configured WiFi SSIDs.

7. **Configure new AP**: used to connect to a new WiFi. There it show all the available WiFi routers and one can connect to one having SSID and Password.
   Note: The config page sometimes takes quite long to load (esp. on a mobile phone). Patience needed!!!

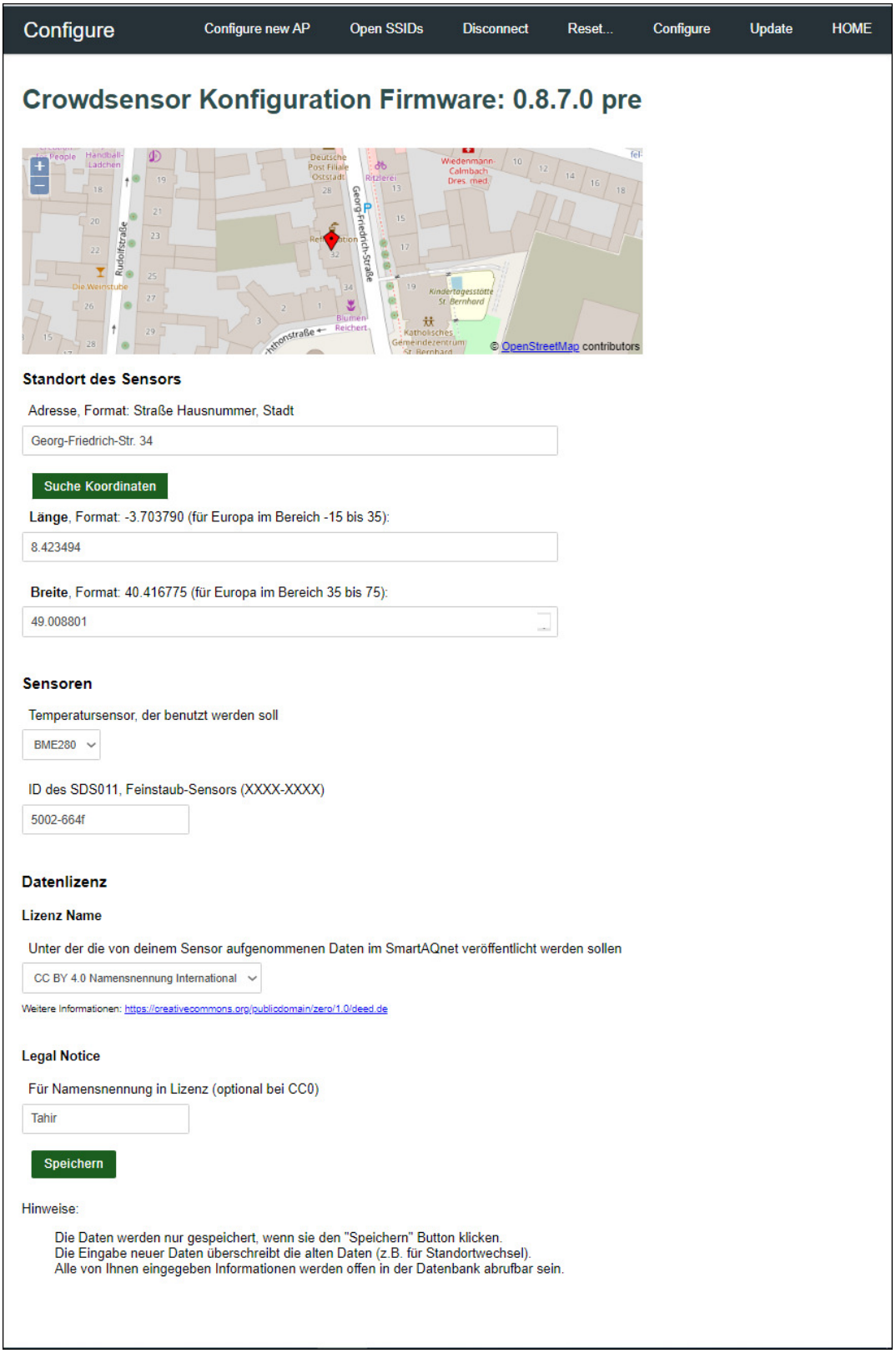


Figure 1.9 Configuration Page

1.1.8 MDNS

A domain name service is used instead of typing IP. It is configure according to hardware ID of ESP32. One can open configuration page using http://[ID].local/.

1.1.9 Daily Reboot

There is a daily reboot at time defined by (TIMETOREBOOT) if it is enabled by (ENABLE_DAILY_REBOOT). This is incoporated as a precaution.

1.1.10 Sensing and Sending Intervals

The temperature and dust values are sensed at an interval of (STATS_INTERVAL), and posted at the interval of (STATS_INTERVAL) if HTTP server is available, location data is given and if the server entities are created seccessfully. The sensed values are also sent to madavi and send2luftdaten servers at the interval of (SEN↩ D2MADAVI_INTERVAL) and the interval of (SEND2LUFTDATEN_INTERVAL), if (ENABLE_SEND2MADAVI) and (ENABLE_SEND2LUFTDATEN) is enabled respectively.

1.1.11 Modelling and Creating Entities

Before posting the observations of dust and temperature sensors, different type of entities need to be created on the server using HTTP methods. Usually, the the entities which are created consists of: Thing, Sensor, Location, Obervedproperty, Datastream and Observation. The nomenclature used for the creation can be seen in the presentation **Modelling the SmartAQnet Database**. Some of the entities are created from scratch, while some are only patched if already exist. For example, the "Thing" is patched with location and properties. The "Datastream" is alse needs to be patched with the properties. The sequence of the creation of different entities and how they are patched with required location and/or properties can be seen in the flowchart **Create Server Entities Flow**. Entities are created using particular JSON messages which are transmitted using HTTP methods. The seccessfull creation indicates only the status code of 200 and 201, and based on this a boolean response is returned. The observations are then posted at the interval of (STATS_INTERVAL) and if (ENABLE_SEND2FROST) is enabled. The Software property version can be controlled by the constant (sw_version) and timestamp is assign on the fly, while the harware revision date is controlled by using the constant (hw_date).
The Table 1.2 shows which entities are to be created and which entities are patched? The entities are created if they does not exist and patched with different properties if they already exist.

Table 1.2 Entities Creation & Patching

Entity	Type	Patched?	Patched With?
Thing	None	Yes	Location, Software Version
Location	None	No	None
Sensor	SDS011	No	None
	BME280	No	None
	DHT22	No	None
Observed Properties	PM2.5	No	None
	PM10	No	None
	Temperature	No	None
	Pressure	No	None
	Humidity	No	None
Observations	PM2.5	No	None
	PM10	No	None

Entity	Type	Patched?	Patched With?
	Temperature	No	None
	Pressure	No	None
	Humidity	No	None
Datastreams	PM2.5	Yes	Software Version, License
	PM10	Yes	Software Version, License
	Temperature	Yes	Software Version, License
	Pressure	Yes	Software Version, License
	Humidity	Yes	Software Version, License

1.1.12 FlowCharts

The project work-flow can be see in the **flowchart**. Also there is a another flowchart and explanation regarding "Inside AutoConnect::begin" function at https://hieromon.github.io/AutoConnect/lsbegin.html .
There is a another flowchart and explanation regarding creating & patching different SAQN Database Entities on the server using JSON messages, as shown in **Create Server Entities Flow**.
There creation of entities implements OGC like API i.e. https://developers.sensorup.com/docs/#sensorthingsAPISensing).

1.1.13 Important Constants

Have a look at the following constants to configure them accordingly. These constants needs to be configured before the firmware is uploaded.

- in the file myAutoConnect.h
  - WIFI_REBOOT_TIMEOUT set Timeout to search WiFi already configured
- in the file create_json.h
  - SOFTWARE_VERSION Software version of the crowdsensor
- in the file sensor.h
  - DHT_PIN 21, set DHT22 sensor communication pin
  - BME_ADDR 0x77, set BME280 I2C Address 0x77 for Red, 0x76 for Purple
  - BME_SDA, set BME280 I2C SDA pin
  - BME_SCL, set BME280 I2C SCL pin
  - SDS011_TXD, set SDS011 TXD pin is connected at RXD of Serial2 Object
  - SDS011_RXD, set SDS011 RXD pin is connected at TXD of Serial2 Object (no need to physically connect)
- in the file main.h
  - FIRMWARE_VERSION controls the hard-coded firmware version of the code
  - sw_version describes the software version of the Thing's software properties
  - hw_date describes the hardware revision timestamp of the Thing's hardware properties

- BASE_URL HTTP Server Address
- RELEASE controls the debugging prints and MY_DEBUG
- ENABLE_SEND2FROST, set switch for sending to FROST teco server
- SENSOR_INTERVAL, set intervall of reading sensor data in ms
- STATS_INTERVAL, set intervall of showing statistical/informative screen multiple of (SENSOR_INTERVAL)
- ENABLE_SEND2MADAVI, set switch for sending to madavi api
- ENABLE_SEND2LUFTDATEN, set switch for sending to luftdaten api
- SEND2MADAVI_INTERVAL, set intervall of sending to madavi api in ms
- SEND2LUFTDATEN_INTERVAL, set intervall of sending to madavi api in ms
- ENABLE_DAILY_REBOOT, set enable daily reboot at specific time
- TIMETOREBOOT, set time when to do daily reboot (hour of time of day for reboot goes here) //TODO set proper time
- APMODE_REBOOT_TIMEOUT, set intervall of auto reboot while in access point mode (config mode) in ms (6000000ms is 10min). autoreboot while in ap mode only occurs if no device is connected to the esp32
- NTP_TIMEOUT_WAIT, set Timeout for NTP timeout (s) in waitForSync()
- CREDENTIAL_OFFSET, Specified the NVS offset if the user data exists.
- data1_url_cc0 defines the Smart AQnet Licese for public domain
- data1_url_ccby defines the Smart AQnet Licese property

1.2 Pre-requisites

1. Arrange the required Hardware and carry out the Wirings.
2. Install the the Arduino IDE
3. Install the ESP32 Arduino core
4. Install the recommended Required Libraries

1.3 Hardware

It consists of ESP32 microcontrollers, which is "HelTec WiFi Kit 32" —> https://heltec.org/project/wifi-kit-32/ Following board, hardware and libraries are used to develop this project.

1.3.1 Peripherals

1. OLED Built-in OLED Display —> https://heltec.org/project/wifi-kit-32/
2. SDS011 Sensor —> https://learn.watterott.com/sensors/sds011/
3. BME280 Sensor for Humidity and Temperature —> https://www.ebay.de/itm/3-3V-Bosch-BME280-Barometer-Luftdruck-Luftfeuchte-Sensor-Modul-BMP280-SPI-I2C-/263548911232 or https://eckstein-shop.de/SparkFun-Barometric-Pressure-Sensor-Breakout-MPL115A1?curr=EUR&gclid=Cj0KCQjww7HsBRDkARIsAARsIT6QsuuszbyaQ4r1DLuYTOT5pgwMfpIhRTtwQKAA2vtHkuHGrBZa9JQaAo-gEALw_wcB
4. DHT22 Sensor for Temperature and Humidity —> https://www.sparkfun.com/products/10167

1.3.2 Arduino IDE

The Arduino IDE Version 1.8 or later is needed. Please install from the official Arduino IDE download page. This step is not required if you already have the most recent version.

1.3.2.1 Pyserial

If you use linux, you have to install the python pyserial package and add the current user to the dialout group:

```
sudo apt install python-pip
pip install pyserial
sudo /usr/sbin/usermod -a -G dialout <MY_USER_NAME>
```

1.3.2.2 ESP32 Arduino core

Also, to apply AutoConnect to ESP32, the arduino-esp32 core provided by Espressif is needed. Stable 1.0.3 or required and the latest release is recommended. Install third-party platform using the Boards Manager of Arduino IDE (for instructions see here: https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/boards_manager.md). You can add multiple URLs into Additional Board Manager URLs field, separating them with commas. Package URL is https://dl.espressif.com/dl/package_esp32_index.json for ESP32.

1.3.2.3 Required Libraries

Following are the commonly used libraries, and should be added in the Arduino IDE, before the compiling.

1. FIRMWARE_VERSION 0.8.6.1 onwards
    - ESP32 Board by Espressif Systems v1.0.3 (https://github.com/espressif/arduino-esp32)
    - Adafruit Unified Sensor by Adafruit v1.0.3 (https://github.com/adafruit/Adafruit_Sensor)
    - Adafruit BME280 Library by Adafruit v1.0.8 (https://github.com/adafruit/Adafruit_BME280_Library)
    - ArduinoJson by Benuit Blanchon v 6.12.0 (https://arduinojson.org/?utm_source=meta&utm_medium=library.properties) (IMPORTANT: ArduinoIDE installs the version (v6), but it doesnt work with this one.)
    - DHT sensor library by Adafruit v1.3.4 (https://github.com/adafruit/DHT-sensor-library)
    - PageBuilder by Hieromon Ikasamo v1.3.4 (https://github.com/Hieromon/PageBuilder)
    - eztime by Rop Gonggrijp v0.7.10 (https://github.com/ropg/ezTime)
    - u8g2 by oliver v2.26.14 (https://github.com/olikraus/u8g2)
    - QRCode by Richard Moore v0.0.1 (https://github.com/ricmoo/qrcode/)
    - AutoConnect by Hieromon Ikasamo v1.1.3 (https://github.com/Hieromon/AutoConnect.git)
    - name=SDS011 sensor Library by R. Zschiegner v0.0.6 (https://github.com/ricki-z/SDS011). This library is kept in src/lib/SDS011 folder. Even the library is updated on the github. But when tried to installed from library manager it is not installed properly for esp32.

2. FIRMWARE_VERSION 0.8.6 and older
    - Adafruit Unified Sensor by Adafruit v1.0.3 (https://github.com/adafruit/Adafruit_↩ Sensor)
    - Adafruit BME280 Library by Adafruit v1.0.8 (https://github.com/adafruit/Adafruit↩ _BME280_Library)
    - ArduinoJson by Benuit Blanchon v 5.13.5 (https://arduinojson.org/?utm_source=meta&utm↩ _medium=library.properties) (IMPORTANT: ArduinoIDE installs the version (v6), but it doesnt work with this one.)
    - DHT sensor library by Adafruit v1.3.4 (https://github.com/adafruit/DHT-sensor-library)
    - PageBuilder by Hieromon Ikasamo v1.3.4 (https://github.com/Hieromon/Page↩ Builder)
    - eztime by Rop Gonggrijp v0.7.10 (https://github.com/ropg/ezTime)
    - u8g2 by oliver v2.25.10 (https://github.com/olikraus/u8g2)
    - Autoconnect lib (is already in src folder, no need to install) [MODIFIED] [till FIRMWARE_VERSION "0.8.6 pre"]
    - SDS011 sensor Library by R. Zschiegner (already in src folder, no need to install) [MODIFIED] [till FIRMWARE_VERSION "0.8.6 pre"]

1.3.3 Firmware Uploading

Following steps are recommended to upload the firmware before handing over the new crowd-sensor node.

1. To erase flash completely before delivering the product: "C:\Users\Sajjad\Documents\Arduino↩ Data\packages\esp32\tools\esptool_py>esptool.py --chip esp32 erase_flash" from the command terminal.
2. Upload the firmware using the Ardiuno IDE

1.4 Wirings

1. OLED Display (built-in)
   The display is built-in already with the ESP32 board through I2C. The I2C address is 0x3c, SDA pin is 4, SCL pin is 15, and RST pin is 16. They are already connected. The custom fonts are generated using online Font Generatore from -—> http://oleddisplay.squix.ch/#/home. Don't need hardware wiring, just require pin definitions, as:
    - RST, set Reset PIN of OLED Module
    - OLED_ADDR, The I2C address of the SSD1306 OLDE
    - OLED_SDA, The SDA pin of the SSD1306 OLDE
    - OLED_SCL, The SCL pin of the SSD1306 OLDE
2. SDS011 Dust Sensor
   One has to:
    - connect TXD of Dust Sensor SDS011 to Pin 13 of ESP32 (ESP-RX)
    - connect 5V of Dust Sensor SDS011 to Pin 5V of ESP32
    - connect GND of Dust Sensor SDS011 to Pin GND of ESP32
3. BME280 Sensor for Humidity and Temperature

- connect SCK of BME280 sensor to Pin 22 of ESP32 (ESP-SCL)
- connect SDA of BME280 sensor to Pin 21 of ESP32 (ESP-SDA)
- connect VCC of BME280 sensor to 3V3 of ESP32
- connect GND of BME280 sensor to GND of ESP32
- the I2C address for different BME280 (purple/red) can be configure in sensor.h using BME_ADDR constant

4. DHT22 Sensor for Temperature and Humidity (if BME280 is not used)
   Look at the sensor from the front (grid shows in your direction), put the pins facing down, start counting the pins from left (pin1) to right (pin4). pin 3 is not connected.
   One has to:

   - Connect pin 1 of DHT22 sensor (on the left) to 3V3 of ESP32
   - Connect pin 2 of DHT22 sensor to Pin 21 of ESP32 (data)
   - Connect pin 4 of DHT22 sensor (on the right) to GND of ESP32
   - Connect a resistor in range 4.7k to 10k from pin 2 (data) to pin 1 (power) of the sensor

1.5 Known Bugs

There are following bugs and discrepensies that needs to be take care off.

1. The config page sometimes takes quite long to load (esp. on a mobile phone). Patience needed!!!
2. Sometimes, the AutoConnect does not recover the earlier configured credentials after the reset. This is happend in rare situations with some internet routers. Similar issure reported at: https://github.com/espressif/arduino-esp32/issues/1464
3. Sometimes, the EZTime Library can not synchronize with the internet time. This is happend rarely. Sometime, increasing NTP_TIMEOUT_WAIT constant's time, may solve the issue. Similar issue is reported at: https://github.com/ropg/ezTime/issues/40
4. Sometimes, while posting different observations to a server, there are some postings shows negative http status, like: HTTPC_ERROR_CONNECTION_REFUSED (-1) and HTTPC_ERROR_READ_TIMEOUT (-11) as traced in the HTTPClient library. This means, when there is -1 HTTP Status, then the observation is not being posted. Following are some tests that shows the percentage of these errors.

   (a) **Duration**:
   From 2020-01-19T03:02:38.000Z to 2020-01-20T04:14:02.000Z
   Result: 1 day, 1 hour, 11 minutes and 24 seconds= 90,684 seconds
   Analysis:
   Reset occurrences: 3 i.e. routine resets at 4:00
   [API] Post Observation... counts 43429 i.e. 8685.8 observation set of five
   [API] HTTP Code: -1x counts 339
   i.e. 339/43429x100=0.78%

   (b) **Duration**:
   From 2020-01-22T09:34:19.000Z to 2020-01-25T00:03:57.000Z
   Result: 2 days, 14 hours, 29 minutes and 38 seconds = 224,978 seconds
   Analysis:
   Reset occurrences: 2 i.e. routine resets at 4:00
   [API] Post Observation... counts 111655
   [API] HTTP Code: -1x counts 198
   i.e. 198/111655x100 = 0.17%

(c) **Duration**:
From 2020-01-26T22:17:08.000Z to 2020-01-27T20:01:32.000Z
Result: 21 hours, 44 minutes and 24 seconds=78,264 seconds
Analysis:
Reset occurrences: 1 i.e. routine resets at 4:00
[API] Post Observation... counts 38645
[API] HTTP Code: -1x counts 149
i.e. 149/38645x100 = 0.38%

(d) **Duration**:
From 2020-01-28T21:01:21.000Z to 2020-01-30T11:58:11.000Z
Result: 1 day, 14 hours, 56 minutes and 50 seconds=140,210 seconds
Analysis:
Reset occurrences: 2 i.e. routine resets at 4:00
[API] Post Observation... counts 69315
[API] HTTP Code: -1x counts 225
i.e. 225 / 69315 x 100 = 0.32%

(e) **Duration**:
From 2020-02-13T18:45:43.000Z to 2020-02-15T16:52:03.000Z
Result: 1 day, 22 hours, 6 minutes and 20 seconds = 165,980 seconds
Analysis:
Reset occurrences: 2 i.e. routine resets at 4:00
[API] Post Observation... counts 80725
[API] HTTP Code: -1x counts 191
i.e. 191 / 80725 x 100 = 0.22%

1.5.1 Some Observations

1. I tested it with long testing. And had the following observations (Ticket#2157).
   - From "phenomenonTime":"2019-12-31T01:29:44.000Z" to "phenomenonTime":"2019-12-31T17:37↩ :14.000Z". There has been only 27 counts for "[API] HTTP Code: -"
   - Then, there is an interruption on the internet. I noticed that my browser was not responding and there continuous -1 or -11 error.
   - There has been 64 counts for "[API] HTTP Code: -" i.e. another 37 counts just between "phenomenon↩ Time":"2019-12-31T17:37:24.000Z" and "phenomenonTime":"2019-12-31T17:43:13.000Z".
   - The queries [BASE_URL]/Datastreams('saqn%3Ads%3Ace4fe57')/Observations?$count=True&$filter=phenomenon↩ Time%20ge%202019-12-31T01:29:44.000Z%20and%20phenomenonTime%20le%202019-12-31T02↩ :03:53.000Z&$orderby=phenomenonTime%20desc gives 205 count.
   - The queries [BASE_URL]/Datastreams('saqn%3Ads%3A6d3c4e7')/Observations?$count=True&$filter=phenomenon↩ Time%20ge%202019-12-31T01:29:44.000Z%20and%20phenomenonTime%20le%202019-12-31T02↩ :03:53.000Z&$orderby=phenomenonTime%20desc gives 204 count, there was [API] HTTP Code: -1.
   - This means, when there is -1 HTTP Status, then the observation is not being posted. When traced the HTTPClient library for -1 or -11, it shows:
     (a) HTTPC_ERROR_CONNECTION_REFUSED (-1)
     (b) HTTPC_ERROR_READ_TIMEOUT (-11)
2. Just carried the test and judged how many times there is HTTP CODE: -1 (Ticket#2185).
   - Duration:
     – Start: 2020-01-19T03:02:38 and End: 2020-01-20T04:14:02
     – Alternative time units: 1 day, 1 hour, 11 minutes and 24 seconds can be converted to one of these units: 90,684 seconds

- Analysis:
  - Reset occurrences: 3 i.e. routine resets at 4:00
  - ideally it should be around 9068 observation set of five, searchig the following in an Editor shows:
    (a) "[API] Post Observation..." counts 43429 i.e. 8685.8 observation set of five
    (b) "[API] HTTP Code: -1" counts 339 i.e. 339/43429x100=0.0078%

1.6 Change Log

1.6.1 FIRMWARE_VERSION 0.8.6.1 onwards

- [0.8.7.0] Mar. 11, 2020
  - Showing error as E00. Also separate Error Decoding section is added explaining different errors.
  - software version is not now saved in eeprom.
  - license patching bug fix. now it is implemented as: if any of "name" and "legal_notice" is changed, the license is patched with new "name", "legal_notice" and "url".
  - get_value("user_data", "rht_sensor") is used to read rht_sensor values frequently, just dit this at the begining and save it to "rhtSensor"
  - more introduction text, added logo to Latex title page by adding custom headers, removal of "Generated by Doxygen" from refman.tex manually
- [0.8.7.0] Mar. 07, 2020
  - SOFTWARE_VERSION in create_json.h is updated and sync with FIRMWARE_VERSION
  - updated the doxygen with Known bugs
  - license patching is implemented now. If the License is not there or if it is in the old form, it is patched.
  - use the current timestamp (=date of installation or date of first booting) for software.version
  - adjust error code and ID on the same line
  - bug fix to show QR while creating entities and whether it is successful or not
- [0.8.7.0] Mar. 01, 2020
  - handle client is bring out of the statemachine, Alternative QR code screens for ID (in AP mode) and for IP (in Wifi connected), interval can be controlled using different constants. Contrast is controlled.
- [0.8.7.0] Feb. 24, 2020
  - changing bit position of errors according to harwdare/software types, use multiple screens for release version only,
  - cleaning code and removing extra variables and expressions etc. update the flowchart and documentation
- [0.8.7.0] Feb. 20, 2020
  - adding the recent features into SM Flowchart and into create entities flowchart.
  - avoid using long paths, and setting it to use relative path in doxygen html/latex (STRIP_FROM_PATH).
  - License type=>name, owner=>legal_notice, metadata=>url in the json messages, and Lizenz=> Lizenz Name, Name=>Legal Notice in the html configuration page
  - fixing "Lizenz Name", "Legal Notice" split on different lines. bold, italic, strong
- [0.8.7.0] Feb. 18, 2020
  - cleaned extra constant and variables from timer_definition.h, made interrupt based inetervals to balance screen timings

  - added images into doxygen html, generated doxygen latex and then pdf

- [0.8.7.0] Feb. 14, 2020
  - ESP32 board version 1.0.4 downgraded to 1.0.3, as the earlier was having problem with mDNS.
  - added integer save and read function from eeprom
  - There are two screens that are toggled with the interval of 5sec: 1.Normal Screen 2. Statistical Screen

- [0.8.7.0] Jan. 19, 2020
  - modeling new nomenclature, blinking HTTP server error but only at the begining, patching things properties, posting observations with /Datastreams('iotId')/Observations

- [0.8.7.0] Jan. 12, 2020
  - adding 3rd parameter to createOrUpdateEntities(), if true will patch the created entities otherwise just create it. only datastreams are patched.
  - instead of using repeated checks for 200 & 201 code, just copy it inside individiual funstions.
  - Changes the QRCode=ID, Changes Frost Manager Initialising, Updating AutoConnect 1.1.3

- [0.8.7.0] Jan. 3, 2020
  - integrating u8g2 and u8x8 unseccessful, using u8g2 for suitable text FONT and QR code generation, including logos KIT,TECO, showing QR code.
  - using alternative oled and QR code libraries, u8g2: (https://github.com/olikraus/u8g2), QRcode: (https://github.com/ricmoo/qrcode/).

- [0.8.7.0] Dec. 29, 2019
  - introducing the common DEBUG controll using MY_DBG()
  - for HTTP Error Code 404, making consistent check for different "Nothing found." response from different servers

- [0.8.7.0] Dec. 24, 2019
  - first version of deFrosting the esp32-thingsapi, looking into API code and catching up all JSON message and converting into strings without using ArduinoJSON.
  - fixing http error for different entities while creating and posting and gaurentying no error, test different Things and Locations and how they are creating/patched.
  - added flow chart for creating entities, removed all esp32-thingsapi calls/libraries

- [0.8.7.0] Dec. 2, 2019
  - added RELEASE_THINGSAPI1 & RELEASE_THINGSAPI2 in esp32_thingsapi/src/workshop_↩ instances/Release-ThingsAPI.h
  - in esp32-thingsapi, all prints are precded by [API] to differtiate from other debugging statements
  - updated eztime library to 0.8.2, used eztime setInterval(10) function just before waitsync().

- [0.8.7.0] Nov. 1, 2019
  - Updated Espressif from 1.0.2 to 1.0.4 and AutoConnect from 1.0.2 to 1.1.2 (Reuires changes in myAutoConnect.h)
  - added initial screens around FROST initialisation
  - used "master" branch instead of "development" of submodule esp32_thingsapi

- [0.8.6.1] Sept. 19, 2019
  - Adding configuration & OTA pages using AutoConnectAux without Pagebuilder and fixing different issues (Sajjad)
  - cleaning all the unnessary comments, defining HW pins constants, addresses etc and documenting them. (Sajjad)

  - moved to JSON6 (by Jan)
  - fixed graphic OLED overlapping bugs because it don't have clearline() function and it clears whole display (Sajjad)
  - using BME280 RED/PURPLE by just changing address in bme(address, sda, scl) (Sajjad)
  - fixed BME280 not found problem by using another TwoWire instance Wire1 (Sajjad)
  - started doxygen documentations (Sajjad).
  - graphic based oled with qr support (Sajjad).
  - converted to state machine (Sajjad).
  - inherited AutoConnect to use custom code (Sajjad).
  - added NTP sync timeout (Sajjad).

1.6.2 FIRMWARE_VERSION 0.8.6 and older

change history (since version 0.8)

- [0.8.1]
  - bugfix for coordinates: switched to coordinates[]={long, lat...}
- [0.8.2]
  - added hack for HTTP patch for location
  - removed all libs from src folder that are used in its original state and included the libs directly in arduino
  - updated SDS011 lib to newest version
  - changed display font
  - added over the air update support
  - added mDNS: now esp32.local can be typed into browser to connect to esp32
- [0.8.3]
  - bugfix for error that no reboot occured after timeout in ap mode
  - bugfix for wrong display message if no RH T sensor is configured
  - bugfix for wrong display message if Wifi is searched
  - improved display messages
  - bugfix for not working geocode button
- [0.8.4]
  - added properties and webinterface inputs for data license
  - geocoding button does not work again, previous bugfix did not work
- [0.8.5]
  - (hopefully) permanent bugfix for not working geocode button
  - added license url to webinterface
  - added RELEASE macro that can be used to switch between development and release versions (switches server addresses and supressses some serial output)
  - display shows FW version for 2 sec on startup
  - webinterface shows map to pick location
  - added reboot after saving configuration, so that thing and datastreams are patched in new setup()

1.7 Quick Debugging

The detailed debugging can be controlled for the firmware using the constant RELEASE.

1. ESP32-THINGSAPI DEBUGGING For quick debugging of different functionality and parts of the code, we can use following quick tricks for debugging. Defined a constant i.e. RELEAS_THINGSAPI1 & RELEAS_T↩ HINGSAPI2 in the file main.h, for more detailed or more important respectively.
   uncomment MY_DEBUG to supress debugg messages (otherwise will print more detailed)
   All prints are precded by [API] to differtiate from other debugging statements
2. Use Short Timeouts
   - APMODE_REBOOT_TIMEOUT 3600000=10min (default). This is intervall of auto reboot while in access point mode. Try with 1min for debugging
   - NTP_TIMEOUT_WAIT=60s. Timeout for NTP timeout in waitForSync(), try 20s
   - WIFI_REBOOT_TIMEOUT=150000=2.5min, Timeout to search WiFi already configured. Try with 1min for debugging
3. Sending data without JSON error: To test some functionality, like AutoConnect, Wifi, EEPROM, Timeouts etc, uploading is not required, one can skip it.

```
case check_location_data:
   // only send if position data is given
   //if (enable_send2frost &&  (get_value("user_data", "latitude") != ""))
   //{
   //  Serial.println("position data is given, start sending");
   //  state = check_timesync;
   //}else
     state = handle_client;
   break;
```

4. ByPassing Save Credentials: One can use the manual WiFi to quickly run the code. Put the following in the setup()

```
WiFi.begin("WG21", "Pakistan1");//Sajjad Debugging
```

5. Testing Webpage & Avoid changing data: One can fix the eeprom data by putting the following code in the setup()

```
set_value("user_data", "address", "");       //inputAddress
set_value("user_data", "latitude", "");       //inputLon
set_value("user_data", "longitude", "");      //inputLat
set_value("user_data", "rht_sensor", "default");  //inputRhtsensor
set_value("user_data", "sds011id", "");       //inputSds011id
set_value("user_data", "data_license", "CC BY 4.0");//inputLicense
set_value("user_data", "data_owner", "");     //inputOwner
```

6. If the storage space is too small when compiling, change it in arduino IDE:
   Tools -> board -> select ESP32 DEV Module
   Tools -> partition scheme -> No OTA (large APP)

1.8 Author(s)

- Jan Formanek,
- Sajjad Hussain,
- Matthias Budde,

The team at TECO (www.teco.edu) and at Chair for Pervasive Computing Systems (https://pcs.tm.↩kit.edu).

1.9 License

This code belongs to TECO www.teco.edu and must be reffered when used. BSD license, all text here must be included in any redistribution.

1.9.1 Acknowledgments

The following libraries were used:

- Espressif Systems: Licenses and Copyright 2015-2016 Espressif Systems (Shanghai) PTE LTD
- ArduinoJson is licensed under the MIT License. https://arduinojson.org/
- Adafruit BME280 is licensed under Adafruit. https://adafruit.com/
- DHT sensor is licensed under Adafruit. https://adafruit.com/
- AutoConnect is licensed under the MIT License. https://github.com/Hieromon/Auto↩Connect
- QRCode by Richard Moore is licensed under the MIT License https://github.com/ricmoo/QR↩Code/blob/master/LICENSE.txt
- ezTime is licensed under the MIT License. https://github.com/ropg/ezTime
- PageBuilder is licensed under the MIT License. https://github.com/Hieromon/Page↩Builder
- U8g2 is licensed under the MIT License. https://github.com/olikraus/u8g2
- SDS011 sensor Library is licensed under the MIT License. (https://github.com/ricki-z/SD↩S011)

1.9.2 MIT License

```
Copyright (c) 2017-2019 Hieromon Ikasamo

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

1.9.3 U8g2lib License

```
The U8g2lib code (http://code.google.com/p/u8g2/) is licensed under the terms of
the new-bsd license (two-clause bsd license).
See also: http://www.opensource.org/licenses/bsd-license.php

The repository and optionally the releases contain icons, which are
derived from the WPZOOM Developer Icon Set:
http://www.wpzoom.com/wpzoom/new-freebie-wpzoom-developer-icon-set-154-free-icons/
WPZOOM Developer Icon Set by WPZOOM is licensed under a Creative Commons
Attribution-ShareAlike 3.0 Unported License.

Fonts are licensed under different conditions.
See  https://github.com/olikraus/u8g2/wiki/fntgrp
for detailed information on the licensing conditions for each font.
```

1.9.4 Espressif Systems License

```
Copyright 2015-2016 Espressif Systems (Shanghai) PTE LTD

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

Date

19.08.2019

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AutoConnect
myAutoConnect 29

Chapter 3

Class Index

3.1 Class List

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

myAutoConnect
MyAutoConnect Class inherited from AutoConnect 29

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

esp32-stationary/create_json.h
Create JSON string out of given values . . . 33
esp32-stationary/display.h
Create logs images & QR . . . 36
esp32-stationary/esp32-stationary.ino
This is the Arduino .ino file . . . 37
esp32-stationary/main.h
This the main page for the SMARTAQNET which is the core of the project . . . 39
esp32-stationary/myAutoConnect.h
Inherits the AutoConnect class and overrides few functions . . . 59
esp32-stationary/nvs_storage.h
Function to store and read nvs flash variables . . . 60
esp32-stationary/send_data.h
Function to send json data to an API . . . 63
esp32-stationary/sensor.h
Variable and definitions for sensors . . . 64
esp32-stationary/timer_definitions.h
Variable and function definitions for timings . . . 65

Chapter 5

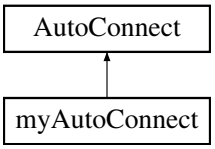
Class Documentation

5.1 myAutoConnect Class Reference

myAutoConnect Class inherited from AutoConnect

`#include <myAutoConnect.h>`

Inheritance diagram for myAutoConnect:



Public Member Functions

- myAutoConnect (WebServerClass &webServer)
- wl_status_t _waitForConnect (unsigned long timeout)
- bool begin (void)
- bool begin (const char *ssid, const char *passphrase, unsigned long timeout)

5.1.1 Detailed Description

myAutoConnect Class inherited from AutoConnect

Definition at line 63 of file myAutoConnect.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 myAutoConnect()

```
myAutoConnect::myAutoConnect (
            WebServerClass & webServer )  [inline]
```

myAutoConnect default constructor. This entry activates WebServer internally and the web server is allocated internal.

Definition at line 71 of file myAutoConnect.h.

5.1.3 Member Function Documentation

5.1.3.1 _waitForConnect()

```
wl_status_t myAutoConnect::_waitForConnect (
            unsigned long timeout )  [inline]
```

Wait for establishment of the connection until the specified time expires.

Parameters

timeout	Expiration time by millisecond unit.

Returns

wl_status_t

Definition at line 78 of file myAutoConnect.h.

5.1.3.2 begin() [1/2]

```
bool myAutoConnect::begin (
            void  )  [inline]
```

Starts establishing WiFi connection without SSID and password.

Definition at line 104 of file myAutoConnect.h.

5.1.3.3 begin() [2/2]

```
bool myAutoConnect::begin (
            const char * ssid,
            const char * passphrase,
            unsigned long timeout )  [inline]
```

Starts establishing WiFi connection. Before establishing, start the Web server and DNS server for the captive portal. Then begins connection establishment in WIFI_STA mode. If connection can not established with the specified SSID and password, switch to WIFI_AP_STA mode and activate SoftAP.

Parameters

ssid	SSID to be connected.
passphrase	Password for connection.
timeout	A time out value in milliseconds for waiting connection.

Returns

true Connection established, AutoConnect service started with WIFI_STA mode.
false Could not connected, Captive portal started with WIFI_AP_STA mode.

Definition at line 123 of file myAutoConnect.h.

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

- esp32-stationary/myAutoConnect.h

Chapter 6

File Documentation

6.1 esp32-stationary/create_json.h File Reference

Create JSON string out of given values.

Functions

- String json_madavi (bool status_sds, float pm10, float pm25, float temp, float hum, float atm)
  create a json string
- String json_sds (bool status_sds, float pm10, float pm25)
  create a json string for SDS values
- String json_RHT (float temp, float hum, float atm)
  create a json string for temperature & Humidity

Variables

- bool use_bme280
  Externally define variable to check if using bme280.
- bool use_dht22
  Externally define variable to check if using dht22.
- const String SOFTWARE_VERSION = "esp32-crowdsensor-v"+String(FIRMWARE_VERSION)
  Software version of the crowdsensor for json messages.
- const int bufferLength = 1024
  The JSON Buffer length.

6.1.1 Detailed Description

Create JSON string out of given values.

This file is used to create JSON messages for different sensors, according to their destination and send to differnt servers.

Date

19.08.2019

6.1.2 Function Documentation

6.1.2.1 json_madavi()

```
String json_madavi (
            bool status_sds,
            float pm10,
            float pm25,
            float temp,
            float hum,
            float atm )
```

create a json string

Parameters

status_sds	SDS sensor status
pm10	sensor's pm10 float value
pm25	sensor's pm25 float value
temp	temperature float value
hum	humidity float value
atm	pressure float value

Returns

output String response

Definition at line 32 of file create_json.h.

6.1.2.2 json_RHT()

```
String json_RHT (
            float temp,
            float hum,
            float atm )
```

create a json string for temperature & Humidity

Parameters

temp	temperature value
hum	humidity value
atm	pressure value

Returns

JSON String

Definition at line 131 of file create_json.h.

6.1.2.3 json_sds()

```
String json_sds (
            bool status_sds,
            float pm10,
            float pm25 )
```

create a json string for SDS values

Parameters

status_sds	SDS sensor status
pm10	pm10 float value
pm25	pm25 float value

Returns

output String response

Definition at line 92 of file create_json.h.

6.1.3 Variable Documentation

6.1.3.1 use_bme280

```
bool use_bme280
```

Externally define variable to check if using bme280.

Externally define variable to check if using bme280.

Definition at line 1149 of file main.h.

6.1.3.2 use_dht22

```
bool use_dht22
```

Externally define variable to check if using dht22.

Externally define variable to check if using dht22.

Definition at line 1151 of file main.h.

6.2 esp32-stationary/display.h File Reference

Create logs images & QR.

```
#include <U8g2lib.h>
#include <U8x8lib.h>
#include <qrcode.h>
```

Macros

- #define _QR_doubleSize
  define for double sized pixel QR code

Functions

- void setPixelsHigh ()
  Highlight the OLED pixels.
- void displayQRCode (unsigned int xinit, unsigned int yinit)
  Display the created QR code at given x,y coordinate.
- void generateQRCode (unsigned int xinit, unsigned int yinit, const char ∗txt)
  Generate the QR code.

Variables

- const int QRcode_Version = 3
  version 3 code with double sized code and starting at y0 = 2 is good
- const int QRcode_ECC = 0
  version 3 with ECC_LOW gives 53 "bytes".
- const uint8_t PROGMEM teco [ ]
  TECO Icon bits of size 64x128.
- const uint8_t PROGMEM kit [ ]
  KIT Icon bits of size 64x128.
- QRCode qrcode
  Create the QR code variable.
- U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2
  U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /∗ reset=∗/ 16, /∗ clock=∗/ 15, /∗ data=∗/ 4);.

6.2.1 Detailed Description

Create logs images & QR.

This file is used to create ICON images and creating QR-code for OLED display of size 64x128.

Date

3.1.2020

6.2.2 Function Documentation

6.2.2.1 displayQRCode()

```
void displayQRCode (
            unsigned int xinit,
            unsigned int yinit )
```

Display the created QR code at given x,y coordinate.

Parameters

xinit	int value for x coordinate
yinit	int value for y coordinate

Definition at line 235 of file display.h.

6.2.2.2 generateQRCode()

```
void generateQRCode (
            unsigned int xinit,
            unsigned int yinit,
            const char * txt )
```

Generate the QR code.

Parameters

xinit	int value for x coordinate
yinit	int value for y coordinate
txt	String value for generating the QR Code

Definition at line 289 of file display.h.

6.3 esp32-stationary/esp32-stationary.ino File Reference

This is the Arduino .ino file.

```
#include "main.h"
```

Functions

- void setup ()

  initialization of peripherals attached to ESP32 board
- void loop ()

  reading the Microphone with timer function

6.3.1 Detailed Description

This is the Arduino .ino file.

This is the ESP32 implementation file, which implement the overall flow of the project by calling different files/functions. It implements the state-machine like flow as described in the `flowchart`.

Date

19.08.2019

6.3.2 Function Documentation

6.3.2.1 loop()

```
void loop ( )
```

reading the Microphone with timer function

Returns

void

Definition at line 171 of file esp32-stationary.ino.

6.3.2.2 setup()

```
void setup ( )
```

initialization of peripherals attached to ESP32 board

Returns

void

Definition at line 23 of file esp32-stationary.ino.

6.4 esp32-stationary/main.h File Reference

This the main page for the SMARTAQNET which is the core of the project.

```
#include "src/lib/SDS011/SDS011.h"
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <DHT.h>
#include <ArduinoJson.h>
#include <Preferences.h>
#include <WiFi.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <Wire.h>
#include <U8g2lib.h>
#include <U8x8lib.h>
#include <ezTime.h>
#include <HTTPClient.h>
#include "nvs_storage.h"
#include "sensor.h"
#include "timer_definitions.h"
#include "send_data.h"
#include "create_json.h"
#include "myAutoConnect.h"
```

Macros

- #define FIRMWARE_VERSION "0.8.7.0 pre"

  firmware version, (for intermediate versions being actively worked on, append "pre" to number, indicating that it is "not yet" the specified number)
- #define sw_version FIRMWARE_VERSION

  Software.version Name property for Thing and Datastreams.
- #define hw_date "2020-01-17T12:00:00.000Z"

  Hardware.Revision Date property for Thing and Datastreams.
- #define BASE_URL "http://193.196.38.108:8080/FROST-Server/v1.0"

  Debugging, uncomment to switch servers to release version, enable sending to madavi and luftdaten.info, and supress some debug output.
- #define MY_DEBUG

  Define MYDEBUG for controlling debugging.
- #define MY_DBG(...) do {Serial.print(__VA_ARGS__);} while (0)

  Define MY_DEB() for controlling serial.print debugging.
- #define MY_DBGln(...) do {Serial.println(__VA_ARGS__);} while (0)

  Define MY_DEBln() for controlling serial.println debugging.
- #define ENABLE_SEND2FROST true

  switch for sending to FROST teco server
- #define SENSOR_INTERVAL 10000

  intervall of reading sensor data in ms
- #define STATS_INTERVAL 10

  intervall of showing statistics informations, it is multiple of SENSOR_INTERVAL i.e. 10 mean 100s
- #define ENABLE_SEND2MADAVI false

  switch for sending to madavi api
- #define ENABLE_SEND2LUFTDATEN false

switch for sending to luftdaten api

- #define SEND2MADAVI_INTERVAL 145000

  intervall of sending to madavi api in ms
- #define SEND2LUFTDATEN_INTERVAL 145000

  intervall of sending to madavi api in ms
- #define ENABLE_DAILY_REBOOT true

  enable daily reboot at specific time
- #define TIMETOREBOOT "2245"

  set time when to do daily reboot (hour of time of day for reboot goes here) //TODO set proper time
- #define APMODE_REBOOT_TIMEOUT 600000

  intervall of auto reboot while in access point mode (config mode) in ms (600000ms is 10min). autoreboot while in ap mode only occurs if no device is connected to the esp32
- #define NTP_TIMEOUT_WAIT 120

  Timeout for NTP timeout (s) in waitForSync()
- #define CREDENTIAL_OFFSET 0

  Specified the offset if the user data exists.
- #define data1_url_cc0 "https://creativecommons.org/publicdomain/zero/1.0/deed.de"

  url for public domain
- #define data1_url_ccby "https://creativecommons.org/licenses/by/4.0/deed.de"

  url for license property
- #define RST 16

  Reset PIN of OLED Module.
- #define OLED_ADDR 0x3c

  The I2C address of the SSD1306 OLDE.
- #define OLED_SDA 4

  The SDA pin of the SSD1306 OLDE.
- #define OLED_SCL 15

  The SCL pin of the SSD1306 OLDE.
- #define dsPM25 1

  Post Observation Selector for SDS011 PM25.
- #define dsPM10 2

  Post Observation Selector for SDS011 PM10.
- #define dsHmBme 3

  Post Observation Selector for BME280 Humidity.
- #define dsTmBme 4

  Post Observation Selector for BME280 Temperature.
- #define dsPrBme 5

  Post Observation Selector for BME280 Pressure.
- #define dsHmDht 6

  Post Observation Selector for DHT22 Humidity.
- #define dsTmDht 7

  Post Observation Selector for DHT22 Temperature.
- #define init_state 1

  state machine initial state
- #define check_first_sync 2

  state machine check NTP sync state
- #define idle_state 3

  state machine idle state
- #define wifi_found 4

  state machine WiFi found state
- #define start_measurement_sds 5

state machine start measurement state

- #define start_measurement_bme 6

  state machine check sensor selection state

- #define show_sds_values 7

  state machine show valid SDS values state

- #define show_bme_values 8

  state machine show valid BME values state

- #define check_location_data 9

  state machine check if position data is given state

- #define check_timesync 10

  state machine check NTP sync again state

- #define create_frost_entties 11

  state machine check if frost is initialized state

- #define send_sds_data 12

  state machine send sds data state

- #define send_bme_data 13

  state machine send bme data state

- #define send_dht_data 14

  state machine send dht data state

- #define send_to_madavi 15

  state machine send madavi data state

- #define send_to_luftdaten 16

  state machine send luftdaten data state

- #define wifi_error 17

  state machine WiFi error state

- #define handle_client 18

  state machine handle client state

- #define check_server_status 19

  state machine check server response state

- #define show_2nd_screen 20

  state to show the 2nd screen

- #define start_showing 21

  state to start showing screens

Functions

- myAutoConnect Portal (Server)

  initialise AutoConnect

- U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2 (U8G2_R0, 16, 15, 4)

  initialise the OLED which is used. OLED Module Object with I2C

- ACElement (cfgScript, " <script src=\"http://www.openlayers.org/api/OpenLayers.js\"></script>\n" " <script>\n" " //global vars\n" " var latitude;\n" " var longitude;\n" " //var elevation;\n" " var address = \"\";\n" " var wgsProjection = new OpenLayers.Projection(\"EPSG:4326\"); // WGS 1984 Projection\n" " var mercatorProjection = new OpenLayers.Projection(\"EPSG:900913\"); // Spherical Mercator Projection\n" " var initialposition = new OpenLayers.LonLat(8.424171,49.013034).transform(wgsProjection, mercator↩ Projection);\n" " var initialzoom = 3;\n" " var position = initialposition;\n" " var zoom = 18;\n" " var mapnik = new OpenLayers.Layer.OSM();\n" " var markers = new OpenLayers.Layer.Markers(\"Markers\");\n" " var marker = new OpenLayers.Marker(position);\n" " var touchNavigation = new OpenLayers.Control.Touch↩ Navigation({\n" " defaultClickOptions:{\n" " pixelTolerance: 10\n" " }\n" " });\n" "\n " " // function to geocode: look up coordinates from address text field, populate lon and lat fields and update map \n" " function

geolookup() {\n" " address = document.getElementById('inputAddress').value;\n" " var request = new X↩ MLHttpRequest();\n" " var url = \"https://nominatim.openstreetmap.org/search/\" + encodeURI(address) + \"?format=json&limit=1\";\n" " request.open('GET', url, true);\n" " request.onload = function() {\n" " if (this.↩ status >= 200 && this.status < 300) {\n" " var response = JSON.parse(this.responseText);\n" " if (response[0]) {\n" " latitude = Number(response[0].lat).toFixed(6);\n" " longitude = Number(response[0].lon).toFixed(6);\n" " document.getElementById(\"inputLon\").value = longitude;\n" " document.getElementById(\"inputLat\").value = latitude;\n" " updateMarker();\n" " } else {\n" " alert(\"Keine Koordinaten gefunden, bitte Adress-Anfrage ändern.\");\n" " }\n" " } \n" " };\n" " request.send();\n" " }\n" "\n " " // function to \"move\" marker (avoiding marker.↩ moveTo(), as it is an \"unofficial\" function)\n" " function replaceMarker() {\n" " markers.clearMarkers();\n" " position = new OpenLayers.LonLat(longitude,latitude).transform(wgsProjection, mercatorProjection);\n" " marker = new OpenLayers.Marker(position);\n" " markers.addMarker(marker);\n" " }\n" "\n " "\n " " // function to update marker based on inputs and center on map\n" " function updateMarker() {\n" " longitude = document.↩ getElementById(\"inputLon\").value;\n" " latitude = document.getElementById(\"inputLat\").value; \n" "\n " " if (typeof map !== 'undefined') {\n" " if (latitude && longitude) {\n" " replaceMarker();\n" " map.setCenter(position, zoom);\n" " } else {\n" " markers.clearMarkers();\n" " map.setCenter(initialposition, initialzoom);\n" " }\n" " }\n" " }\n" "\n " " // function to \"move\" marker on click in map\n" " OpenLayers.Util.extend(touchNavigation, {\n" " defaultClick: function(evnt) {\n" " var lonlat = map.getLonLatFromViewPortPx(evnt.xy);\n" "\n" " latitude = lonlat.transform(mercatorProjection, wgsProjection).lat.toFixed(6);\n" " longitude = lonlat.lon.toFixed(6);\n" "\n" " document.getElementById(\"inputLon\").value = longitude;\n" " document.getElementById(\"input↩ Lat\").value = latitude;\n" "\n" " replaceMarker();\n" " }\n" " });\n" "\n" " // function to unfocus current form element when enter key was pressed\n" " function unfocusForm() {\n" " if (event.key == 'Enter') {\n" " document.↩ activeElement.blur();\n" " return false;\n" " } else {\n" " return true;\n" " }\n" " }\n" "\n " " // execute the following once all DOM elements are loaded\n" " document.addEventListener('DOMContentLoaded', function(){\n" "\n " " // add geocoding button\n" " var findbutton = document.createElement(\"button\");\n" " findbutton.↩ setAttribute(\"id\", \"geocode\");\n" " findbutton.setAttribute(\"type\", \"button\");\n" " findbutton.innerHTML = \"Suche Koordinaten\";\n" " findbutton.addEventListener (\"click\", function() {\n" " geolookup();\n" " });\n" " document.getElementById(\"pgeocode\").appendChild(findbutton);\n" "\n " " // init OpenLayers map\n" " map = new OpenLayers.Map(\"mapDiv\");\n" " map.addLayer(mapnik);\n" " map.addLayer(markers);\n" "\n " " updateMarker();\n" "\n " " // register function to \"move\" marker\n" " map.addControl(touchNavigation); \n" " touchNavigation.activate();\n" "\n " " document.getElementById(\"inputLon\").addEventListener(\"change\", updateMarker);\n" " document.getElementById(\"inputLat\").addEventListener(\"change\", updateMarker);\n" "\n " "\n " " });\n" " </script>\n")

Crowdsensor html page script.

- ACText (cfgData, " ")

  Crowdsensor temporary ACText Element.

- ACElement (cfgBody, "<style type=\"text/css\"> \n" " input[type=number], input#owner { font-weight: 300; width: calc(100% - 124px); background-color: #fff; border: 1px solid #ccc; border-radius: 2px; color↩ : #444; margin: 8px 0 8px 0; padding: 10px; }\n" " button[type=button] { font-weight: normal; padding: 8px 14px; margin: 12px; width: auto; outline: none; text-decoration: none; background-color: #1b5e20; border-color: #1b5e20; letter-spacing: 0.8px; color: #fff; border: 1px solid; border-radius: 2px; font-size: 0.9em; }\n" " div#OpenLayers_Control_Attribution_16 { bottom: .2em; }\n" "</style>\n" "\n" " <div id=\"mapDiv\" style=\"height:250px\">

\n" "\n" " <h3>Standort des Sensors</h3>\n" " <label for=\"inputAddress\">Adresse, Format: Straße Hausnummer, Stadt </label>\n" "
\n" " <input type=\"text\" name=\"inputAddress\" id=\"inputAddress\" maxlength=\"160\" placeholder=\"Adresse\" value=\"\" style=\"width: calc(100% - 124px);\">\n" "
\n" " <span id=\"pgeocode\">\n" "
\n" " <label for=\"inputLon\">Länge, Format: -3.703790 (für Europa im Bereich -15 bis 35):</label>\n" "
\n" " <input type=\"number\" name=\"inputLon\" id=\"inputLon\" min=\"-180\" max=\"180\" step=\"0.000001\" placeholder=\"z.B. -3.703790\" value=\"\" required onchange=\"update↩ Marker()\">\n" "
\n" "
\n" " <label for=\"inputLat\">Breite, Format: 40.416775 (für Europa im Bereich 35 bis 75):</label>\n" "
\n" " <input type=\"number\" name=\"inputLat\" id=\"inputLat\" min=\"-90\" max=\"90\" step=\"0.000001\" placeholder=\"z.B. 40.416775\" value=\"\" required onchange=\"updateMarker()\"> \n" "
\n" "
\n" " <h3>Sensoren</h3>\n" " <label for=\"inputRhtsensor\">Temperatursensor, der benutzt werden soll</label>\n" "
\n" " <select name=\"inputRhtsensor\" id=\"inputRhtsensor\">\n" " <option value=\"default\">(keiner)</option>\n" " <option value=\"BME280\">BME280</option>\n" " <option value=\"DHT22\">DHT22</option>\n" " </select>\n" "
\n" "
\n" " <label for=\"inputSds011id\">ID des SDS011, Feinstaub-↩ Sensors (XXXX-XXXX)</label>\n" "
\n" " <input type=\"text\" name=\"inputSds011id\" id=\"input↩ Sds011id\" maxlength=\"10\" placeholder=\"XXXX-XXXX\" value=\"\">\n" "
\n" "
\n" " <h3>↩

Datenlizenz</h3>\n" " <h4>Lizenz Name</h4>\n" " <label for=\"inputLicense\">Unter der die von deinem Sensor aufgenommenen Daten im SmartAQnet veröffentlicht werden sollen</label>\n" "
\n" " <select name=\"inputLicense\" id=\"inputLicense\" onchange=\"set_required()\">\n" " <option value=\"CC0 1.0\">CC0 1.0 Universell</option>\n" " <option value=\"CC BY 4.0\">CC BY 4.0 Namensnennung International</option>\n" " </select>\n" "
\n" " <span style=\"font-size: 75%\"> Weitere Informationen: " data1_url_cc0 "\n" " <script>\n" " function set_required() {\n" " var lic = document.getElementById('inputLicense').value;\n" " if (lic == \"CC BY 4.0\") {\n" " document.getElementById(\"inputOwner\").required = true;\n" " document.getElementById(\"license_url\").innerHTML = \"" data1_url_ccby "\";\n" " document.getElementById(\"license_url\").href = \"" data1_url_ccby "\";\n" " } else {\n" " document.getElementById(\"inputOwner\").required = false;\n" " document.getElementById(\"license_url\").innerHTML = \"" data1_url_cc0 "\";\n" " document.getElementById(\"license_url\").href = \"" data1_url_cc0 "\";\n" " }\n" " }\n" " </script>\n" "
\n" "
\n" " <h4>Legal Notice</h4>\n" " <label for=\"inputOwner\">Für Namensnennung in Lizenz (optional bei CC0)</label>\n" "
\n" " <input type=\"text\" name=\"inputOwner\" id=\"inputOwner\" maxlength=\"40\" placeholder=\"Name (optional bei CC0)\" value=\"\">\n" "
\n" " <input type=\"button\" value=\"Speichern\" name=\"cfgButton\" onclick = \"if (confirm('Daten speichern?')) _sa('/config')\" method=\"post\">\n" "<script>\n" "function ConfirmBox1() {\n" " if (confirm('Daten speichern1?')) window.location.href = \"/config\";\n" "}\n" "</script>\n" "
\n")

Crowdsensor main page html body.

- ACElement (cfgBody2, " Hinweise:
\n" "
\n" " Die Daten werden nur gespeichert, wenn sie den \"Speichern\" Button klicken.
\n" " Die Eingabe neuer Daten überschreibt die alten Daten (z.B. für Standortwechsel).
\n" " Alle von Ihnen eingegeben Informationen werden offen in der Datenbank abrufbar sein.
\n" "
\n" " \n" "<script>\n" " window.addEventListener('load', function(){\n" " document.getElementById(\"_aux\").onkeydown = \"return unfocusForm()\"\n" " });\n" "</script>\n" "")

  Crowdsensor main page html body remaining part.
- ACText (cfgtitle, "", "text-align:center;color:#2f4f4f;")

  Crowdsensor main page html main title as ACText element.
- AutoConnectAux cfgSensorPageAux ("/", "Configure", true, { cfgtitle, cfgScript, cfgBody, cfgData, cfgBody2 })

  Crowdsensor main page cfgSensorPageAux based on different AutoConnect Elements.
- ACElement (otaScript, otaUpdate)

  Crowdsensor OTA page html script.
- AutoConnectAux cfgUpdatePageAux ("/otaUpdate", "Update", true, { otaScript })

  Crowdsensor OTA page cfgSensorPageAux based on different AutoConnect Elements.
- void myDebugging (void)

  Just used for debugging, i.e. to save dummy values on behalf of the webpage.
- String getIsoTime ()

  Get Standard time.
- String getHourMinute ()

  Get hours from the time.
- String getChipId ()

  get chip id from MAC only 3-bytes
- void handleConfig ()

  Function to handle the `http://`[IPAddress]/config response of Webserver::on(...) after button submission.
- String handleRoot (AutoConnectAux &aux, PageArgument &args)

  Function to handle the `http://`[IPAddress]/ response of AutoConnect::on(...)
- void otaUpdateHandler (void)

  Function to handle the `http://`[IPAddress]/otaUpdate HTTP_POST response of Webserver::on(...)
- void updateHandler (void)

  Function to handle the `http://`[IPAddress]/update HTTP_POST response of Webserver::on(...)
- void updateHandlerResponse (void)

  Function to handle the `http://`[IPAddress]/update HTTP_GET response of Webserver::on(...)
- String toHEXSHA (String serial)

create a HASH using SHA-1

- boolean createEntity (String url, String contents)

  Create (HTTP Post) entities.
- boolean patchEntity (String url, String contents)

  Patch (HTTP Patch) entities.
- int getEntity (String url)

  Get the HTTP code (HTTP Get) from a server and print the response.
- String getEntityResponse (String url)

  Get the HTTP response (HTTP Get) from a server and print the response.
- boolean createOrUpdateEntities (String url, String contents, String id, boolean doPatch)

  Create (HTTP Post) if entities does exist and patch (HTTP Patch) if it already exists.
- boolean createOrPatchEntities (String entity, String json, String iotId)

  Create (HTTP Post) if entities does exist and patch (HTTP Patch) the properties if it already exists.
- bool createEntities ()

  create different entities like Thing, Sensor, Datastream, ObservedProperties, & Location and Patch if not already linked
- void postObservation (int datastream, String phenomenonTime, String resultTime, double result)

  Post Observation for sensed vaules of PM25, PM10, humidity, temperature and Pressure (HTTP Post)

Variables

- const int precision = 6

  Precision for location coordinates.
- String licType

  License Type.
- String licOwner

  License Owner.
- String licUrl

  License Url.
- String locName

  Location Name.
- String locDesc

  Location Description.
- String locEnc

  Location Encoding Type.
- String locTyp

  Location Type.
- String locIotId

  Location IOT ID.
- String locCord

  Location Coordinates.
- String locJson

  Location JSON message.
- String thName

  Thing Name.
- String thDesc

  Thing Description.
- String thHwDate

  Thing Hardware Revision Date.
- String thHwVersion

Thing Hardware Revision Version.

- String thSwDate

  Thing Software Version Date.
- String thSwVersion

  Thing Software Version.
- String thHwId

  Thing Hardware ID.
- String thShortName

  Thing Short Name.
- String thOpDomain

  Thing Operater Name.
- String thId

  Thing Simple ID.
- String thIotId

  Thing IOT ID.
- String thJson

  Thing JSON Message.
- String prJson

  Thing Properties JSON message.
- String sdsSerial

  SDS011 Serial No.
- String sdsName

  SDS011 Name.
- String sdsDesc

  SDS011 Description.
- String sdsEnc

  SDS011 Encoding Type.
- String sdsDataUrl

  SDS011 DataSheet URL.
- String sdsShortName

  SDS011 Short Name.
- String sdsManDomain

  SDS011 Menufacture Name.
- String sdsId

  SDS011 Simple ID.
- String sdsIotId

  SDS011 IOT ID.
- String sdsMeta

  SDS011 Meta DATA.
- String sdsJson

  SDS011 JSON Message.
- String bmeSerial

  BME280 Serial No.
- String bmeName

  BME280 Name.
- String bmeDesc

  BME280 Description.
- String bmeEnc

  BME280 Encoding Type.
- String bmeDataUrl

  BME280 Datasheet Url.

- String bmeShortName
  BME280 Short Name.
- String bmeManDomain
  BME280 Manufacture Domain.
- String bmeId
  BME280 Simple ID.
- String bmeIotId
  BME280 IOT ID.
- String bmeMeta
  BME280 MetaData.
- String bmeJson
  BME280 JSON Message.
- String dhtSerial
  DHT22 Serial No.
- String dhtName
  DHT22 Name.
- String dhtDesc
  DHT22 Description.
- String dhtEnc
  DHT22 Encoding Type.
- String dhtDataUrl
  DHT22 Datasheet Url.
- String dhtShortName
  DHT22 Short Name.
- String dhtManDomain
  DHT22 Manufacture Domain.
- String dhtId
  DHT22 SImple ID.
- String dhtIotId
  DHT22 IOT ID.
- String dhtMeta
  DHT22 Metadata.
- String dhtJson
  DHT22 JSON Message.
- String opPM10Name
  ObservedProperty PM10 Name.
- String opPM10Desc
  ObservedProperty PM10 Description.
- String opPM10Defi
  ObservedProperty PM10 Definition.
- String opPM10ShortDefi
  ObservedProperty PM10 Short Definition.
- String opPM10ShortName
  ObservedProperty PM10 Short Name.
- String opPM10UoM
  ObservedProperty PM10 Unit of Measurement.
- String opPM10Sym
  ObservedProperty PM10 UoM Symbol.
- String opPM10ConDef
  ObservedProperty PM10 UoM Convention Definition.
- String opPM10Id

ObservedProperty PM10 Simple ID.

- String opPM10IotId

  ObservedProperty PM10 IoT ID.
- String opPM10Json

  ObservedProperty PM10 JSON message.
- String opPM25Name

  ObservedProperty PM2.5 Name.
- String opPM25Desc

  ObservedProperty PM2.5 Description.
- String opPM25Defi

  ObservedProperty PM2.5 Definition.
- String opPM25ShortDefi

  ObservedProperty PM2.5 Short Definition.
- String opPM25ShortName

  ObservedProperty PM2.5 Short Name.
- String opPM25UoM

  ObservedProperty PM2.5 Unit of Measurement.
- String opPM25Sym

  ObservedProperty PM2.5 UoM Symbol.
- String opPM25ConDef

  ObservedProperty PM2.5 UoM Convention Definition.
- String opPM25Id

  ObservedProperty PM2.5 Simple ID.
- String opPM25IotId

  ObservedProperty PM2.5 IoT ID.
- String opPM25Json

  ObservedProperty PM2.5 JSON Message.
- String opHmName

  ObservedProperty Humidity Name.
- String opHmDesc

  ObservedProperty Humidity Description.
- String opHmDefi

  ObservedProperty Humidity Definition.
- String opHmShortDefi

  ObservedProperty Humidity Short Definition.
- String opHmShortName

  ObservedProperty Humidity Short Name.
- String opHmUoM

  ObservedProperty Humidity Unit of Measurement.
- String opHmSym

  ObservedProperty Humidity UoM Symbol.
- String opHmConDef

  ObservedProperty Humidity UoM Convention Definition.
- String opHmId

  ObservedProperty Humidity Simple ID.
- String opHmIotId

  ObservedProperty Humidity IoT ID.
- String opHmJson

  ObservedProperty Humidity JSON Message.
- String opTmName

  ObservedProperty Temperature Name.

- String opTmDesc

  ObservedProperty Temperature Description.
- String opTmDefi

  ObservedProperty Temperature Definition.
- String opTmShortDefi

  ObservedProperty Temperature Short Definition.
- String opTmShortName

  ObservedProperty Temperature Short Name.
- String opTmUoM

  ObservedProperty Temperature Unit of Measurement.
- String opTmSym

  ObservedProperty Temperature UoM Symbol.
- String opTmConDef

  ObservedProperty Temperature UoM Convention Definition.
- String opTmId

  ObservedProperty Temperature Simple ID.
- String opTmIotId

  ObservedProperty Temperature IoT ID.
- String opTmJson

  ObservedProperty Temperature JSON Message.
- String opPrName

  ObservedProperty Pressure Name.
- String opPrDesc

  ObservedProperty Pressure Description.
- String opPrDefi

  ObservedProperty Pressure Definition.
- String opPrShortDefi

  ObservedProperty Pressure Short Definition.
- String opPrShortName

  ObservedProperty Pressure Short Name.
- String opPrUoM

  ObservedProperty Pressure Unit of Measurement.
- String opPrSym

  ObservedProperty Pressure UoM Symbol.
- String opPrConDef

  ObservedProperty Pressure UoM Convention Definition.
- String opPrId

  ObservedProperty Pressure Simple ID.
- String opPrIotId

  ObservedProperty Pressure IoT ID.
- String opPrJson

  ObservedProperty Pressure JSON Message.
- String dsPM10Name

  Datastream BME280 PM10 Name.
- String dsPM10Desc

  Datastream BME280 PM10 Description.
- String dsPM10ObType

  Datastream BME280 PM10 Type.
- String dsPM10UoMName

  Datastream BME280 PM10 Unit of Measurement.
- String dsPM10UoMSym

Datastream BME280 PM10 UoM Symbol.

- String dsPM10UoMDef

  Datastream BME280 PM10 UoM Definition.
- String dsPM10Id

  Datastream BME280 PM10 Simple ID.
- String dsPM10IotId

  Datastream BME280 PM10 IoT ID.
- String dsPM10Json

  Datastream BME280 PM10 JSON Message.
- String dsPM25Name

  Datastream BME280 PM2.5 Name.
- String dsPM25Desc

  Datastream BME280 PM2.5 Description.
- String dsPM25ObType

  Datastream BME280 PM2.5 Type.
- String dsPM25UoMName

  Datastream BME280 PM2.5 Unit of Measurement.
- String dsPM25UoMSym

  Datastream BME280 PM2.5 UoM Symbol.
- String dsPM25UoMDef

  Datastream BME280 PM2.5 UoM Definition.
- String dsPM25Id

  Datastream BME280 PM2.5 Simple ID.
- String dsPM25IotId

  Datastream BME280 PM2.5 IoT ID.
- String dsPM25Json

  Datastream BME280 PM2.5 JSON Message.
- String dsHmBmeName

  Datastream BME280 Humidity Name.
- String dsHmBmeDesc

  Datastream BME280 Humidity Description.
- String dsHmBmeObType

  Datastream BME280 Humidity Type.
- String dsHmBmeUoMName

  Datastream BME280 Humidity Unit of Measurement Name.
- String dsHmBmeUoMSym

  Datastream BME280 Humidity UoM Symbol.
- String dsHmBmeUoMDef

  Datastream BME280 Humidity UoM Definition.
- String dsHmBmeId

  Datastream BME280 Humidity Simple ID.
- String dsHmBmeIotId

  Datastream BME280 Humidity IoT ID.
- String dsHmBmeJson

  Datastream BME280 Humidity JSON MEssage.
- String dsTmBmeName

  Datastream BME280 Temperature Name.
- String dsTmBmeDesc

  Datastream BME280 Temperature Description.
- String dsTmBmeObType

  Datastream BME280 Temperature Type.

- String dsTmBmeUoMName

  Datastream BME280 Temperature Unit of Measurement Name.
- String dsTmBmeUoMSym

  Datastream BME280 Temperature UoM Symbol.
- String dsTmBmeUoMDef

  Datastream BME280 Temperature UoM Definition.
- String dsTmBmeId

  Datastream BME280 Temperature Simple ID.
- String dsTmBmeIotId

  Datastream BME280 Temperature IoT ID.
- String dsTmBmeJson

  Datastream BME280 Temperature JSON Message.
- String dsPrBmeName

  Datastream BME280 Pressure Name.
- String dsPrBmeDesc

  Datastream BME280 Pressure Description.
- String dsPrBmeObType

  Datastream BME280 Pressure Type.
- String dsPrBmeUoMName

  Datastream BME280 Pressure Unit of MEasurement Name.
- String dsPrBmeUoMSym

  Datastream BME280 Pressure UoM Symbol.
- String dsPrBmeUoMDef

  Datastream BME280 Pressure UoM Definition.
- String dsPrBmeId

  Datastream BME280 Pressure Simple ID.
- String dsPrBmeIotId

  Datastream BME280 Pressure IoT ID.
- String dsPrBmeJson

  Datastream BME280 Pressure JSON Message.
- String dsHmDhtName

  Datastream DHT22 Humidity Name.
- String dsHmDhtDesc

  Datastream DHT22 Humidity Description.
- String dsHmDhtObType

  Datastream DHT22 Humidity Type.
- String dsHmDhtUoMName

  Datastream DHT22 Humidity Unit of Measurement Name.
- String dsHmDhtUoMSym

  Datastream DHT22 Humidity UoM Symbol.
- String dsHmDhtUoMDef

  Datastream DHT22 Humidity UoM Definition.
- String dsHmDhtId

  Datastream DHT22 Humidity Simple ID.
- String dsHmDhtIotId

  Datastream DHT22 Humidity IoT ID.
- String dsHmDhtJson

  Datastream DHT22 Humidity JSON Message.
- String dsTmDhtName

  Datastream DHT22 Temperature Name.
- String dsTmDhtDesc

Datastream DHT22 Temperature Description.

- String dsTmDhtObType

  Datastream DHT22 Temperature Type.
- String dsTmDhtUoMName

  Datastream DHT22 Temperature Unit of Measurement Name.
- String dsTmDhtUoMSym

  Datastream DHT22 Temperature UoM Symbol.
- String dsTmDhtUoMDef

  Datastream DHT22 Temperature UoM Definition.
- String dsTmDhtId

  Datastream DHT22 Temperature Simple ID.
- String dsTmDhtIotId

  Datastream DHT22 Temperature IoT ID.
- String dsTmDhtJson

  Datastream DHT22 Temperature JSON Message.
- bool use_bme280 = false

  tracking if BME280 sensor is selected
- bool use_dht22 = false

  tracking if DHT22 sensor is selected
- bool initialised_frost = false

  if frost manager is initialized, i.e. entities are createdd on the server seccessfully
- bool busy

  indicated the busy status while entities are being created on the server
- bool initialised_http = false

  if HTTP Server manager is initialized
- bool wifi_connected

  if wifi is connected
- bool time_updated

  if NTP time is synced
- unsigned int tObservations

  total number of observations posted
- unsigned int nObservations

  total number of observations not posted successfully
- unsigned int resetCounter

  No. of reboots.
- String timeOfMeasure

  the time of the measurement of the first observation
- unsigned int errorCode

  the error encodings
- unsigned int oldErrCode

  the error encodings last saved
- unsigned int countSensorInterval

  count the intervals of sensor measurement
- float pm10

  vars to store sensor results, sensor results for dust sensor pm10
- float pm25

  sensor results for dust sensor pm25
- float temp

  temperature values
- float hum

  humidity values

- float atm

  pressure values
- String timeofmeas_sds

  time of measurement for sds reading
- String timeofmeas_rht

  time of measurement for rht reading
- int status_sds

  sds status
- boolean http_error

  set display contrast
- unsigned char state

  state machine tracking variable
- float latitude

  Location Latitude Point.
- float longitude

  Location longitude Point.
- float abovenn

  Location abovenn Point.
- float abovennEstimated

  Location Estimated abovenn Point.
- String humanReadableLocation

  Human Readable Name of the street location.
- String rhtSensor

  rht sensor global variable
- String license_type

  Possible additional sensors other than SDS011.
- String license_owner

  License Owner for License property.
- String license_url

  License Meta for License property.
- String device_Serial

  Chip ID for the ESp32.
- String SDS011_Serial

  ID written of SDS011 Sensor.
- byte mac [6]

  used to store mac address
- String esp32_chipid

  Chip ID based on MAC.
- char destination_madavi [ ] = "https://api-rrd.madavi.de/data.php"

  host to send data madavi API
- char destination_luftdaten [ ] = "https://api.luftdaten.info/v1/push-sensor-data/"

  host to send data luftdaten API
- HTTPClient gHttp

  HTTP Client Variable Global.
- int gHttpCode

  HTTP Client Response Global.
- WiFiUDP ntpUDP

  initialise udp for network time
- WebServer Server

  initialise Webserver
- AutoConnectConfig Config

initialise AutoConnectConfig

- String cfgTemp

  String to hold temporary html scripts for AutoConnect.

- const char * otaUpdate

  Page for OTA Programming html.

6.4.1 Detailed Description

This the main page for the SMARTAQNET which is the core of the project.

6.4.2 Macro Definition Documentation

6.4.2.1 BASE_URL

```
#define BASE_URL "http://193.196.38.108:8080/FROST-Server/v1.0"
```

Debugging, uncomment to switch servers to release version, enable sending to madavi and luftdaten.info, and supress some debug output.

Non-Release Version: Frost Server Base address used in Frost_Server.cpp

Definition at line 598 of file main.h.

6.4.3 Function Documentation

6.4.3.1 createEntities()

```
bool createEntities ( )
```

create different entities like Thing, Sensor, Datastream, ObservedProperties, & Location and Patch if not already linked

Returns

created bool variable to indicated if the all the entities are seccessfully created

Definition at line 2134 of file main.h.

6.4.3.2 createEntity()

```
boolean createEntity (
            String url,
            String contents )
```

Create (HTTP Post) entities.

Parameters

url	String that contains the address of entities to be created/patched
contents	String that contains the JSON message for the server

Returns

httpCode int indicating HTTPClient Code Error

Definition at line 1830 of file main.h.

6.4.3.3 createOrPatchEntities()

```
boolean createOrPatchEntities (
            String entity,
            String json,
            String iotId )
```

Create (HTTP Post) if entities does exist and patch (HTTP Patch) the properties if it already exists.

Parameters

entity	String that contains the address of entities to be created/patched
json	String that contains the JSON message for the server
iotId	String that indicate the iot.id

Returns

httpCode boolean indicating HTTPClient Code Error

Definition at line 2017 of file main.h.

6.4.3.4 createOrUpdateEntities()

```
boolean createOrUpdateEntities (
            String url,
            String contents,
            String id,
            boolean doPatch )
```

Create (HTTP Post) if entities does exist and patch (HTTP Patch) if it already exists.

Parameters

url	String that contains the address of entities to be created/patched
contents	String that contains the JSON message for the server
id	String that indicate the iot.id
doPatch	bool if true then create & patch, if false then only create.

Returns

httpCode int indicating HTTPClient Code Error

Definition at line 1983 of file main.h.

6.4.3.5 getEntity()

```
int getEntity (
            String url )
```

Get the HTTP code (HTTP Get) from a server and print the response.

Parameters

url	String that contains the address of entities to be created/patched

Returns

response String indicating HTTPClient Code response

Definition at line 1909 of file main.h.

6.4.3.6 getEntityResponse()

```
String getEntityResponse (
            String url )
```

Get the HTTP response (HTTP Get) from a server and print the response.

Parameters

url	String that contains the address of entities to be created/patched

Returns

response String indicating HTTPClient Code response

Definition at line 1945 of file main.h.

6.4.3.7 patchEntity()

```
boolean patchEntity (
            String url,
            String contents )
```

Patch (HTTP Patch) entities.

Parameters

url	String that contains the address of entities to be created/patched
contents	String that contains the JSON message for the server

Returns

httpCode int indicating HTTPClient Code Error

Definition at line 1872 of file main.h.

6.4.3.8 postObservation()

```
void postObservation (
            int datastream,
            String phenomenonTime,
            String resultTime,
            double result )
```

Post Observation for sensed vaules of PM25, PM10, humidity, temperature and Pressure (HTTP Post)

Parameters

datastream	int selector for PM25, PM10, humidity, temperature and Pressure etc.
phenomenonTime	String that contains phenomenon/occurance Time
resultTime	String that contains result/finish Time
result	double that contains values for PM25, PM10, humidity, temperature and Pressure etc. from SDS011, BME280, DHT22 etc.

Returns

void none value

Definition at line 2522 of file main.h.

6.4.3.9 toHEXSHA()

```
String toHEXSHA (
            String serial )
```

create a HASH using SHA-1

Parameters

serial	String values from where HASH to be calculated

Returns

sha1hex String HASH output

Definition at line 1795 of file main.h.

6.4.4 Variable Documentation

6.4.4.1 esp32_chipid

```
String esp32_chipid
```

Chip ID based on MAC.

Externally define variable for Chip ID.

Definition at line 1227 of file main.h.

6.4.4.2 http_error

```
boolean http_error
```

set display contrast

HTTP Server error

Definition at line 1195 of file main.h.

6.4.4.3 license_type

```
String license_type
```

Possible additional sensors other than SDS011.

additional sensors tracking variable License Type for License property

Definition at line 1215 of file main.h.

6.4.4.4 use_bme280

```
bool use_bme280 = false
```

tracking if BME280 sensor is selected

Externally define variable to check if using bme280.

Definition at line 1149 of file main.h.

6.4.4.5 use_dht22

```
bool use_dht22 = false
```

tracking if DHT22 sensor is selected

Externally define variable to check if using dht22.

Definition at line 1151 of file main.h.

6.5 esp32-stationary/myAutoConnect.h File Reference

Inherits the AutoConnect class and overrides few functions.

```
#include "display.h"
#include <AutoConnect.h>
```

Classes

- class myAutoConnect

  myAutoConnect Class inherited from AutoConnect

Macros

- #define AC_DEBUG

  enable debug output.
- #define AC_DEBUG_PORT Serial

  Debug output destination can be defined externally with AC_DEBUG_PORT.
- #define AC_DBG_DUMB(...) do {AC_DEBUG_PORT.printf(__VA_ARGS__);} while (0)

  Debug output destination can be defined externally with AC_DEBUG_PORT.
- #define AC_DBG(...) do {AC_DEBUG_PORT.print("[AC] "); AC_DEBUG_PORT.printf(__VA_ARGS__);} while (0)

  Debug output destination can be defined externally with AC_DEBUG_PORT.
- #define WIFI_REBOOT_TIMEOUT 150000

  Timeout to search WiFi already configured. (150000ms=2.5min)
- #define SHOW_WAITING_DOTS_WIFI 300

  The interval to show dots while waiting for the searching Wifi (in ms)
- #define SHOW_WAITING_DOTS_AP 500

  The interval to show dots while waiting for the access point (in ms)
- #define SHOW_QR_INTERVAL 10000

  The interval to show QR or AP waiting screen (in ms)

Variables

- U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2

  Externally defined OLED Display object.
- unsigned char cntr

  counter to count 500ms
- boolean show_QR

  show QR or Text for AP
- unsigned long previousMillis

  Just calculate 500ms interval while waiting the AP is configure to connect a WiFi.

6.5.1 Detailed Description

Inherits the AutoConnect class and overrides few functions.

This is the files that contains the functions inherited from AutoConnect Class which needs some modfications to add OLED display code. These functions are then over ridden here. Basically, when there is no wifi connected, it shows a message on serial port and on oled display that is searching for a Wifi. Similarly, when no wifi is connected in sub period of time, it opens an access-point and shows another message with default SSID and password.

Date

19.08.2019

6.6 esp32-stationary/nvs_storage.h File Reference

function to store and read nvs flash variables

```
#include <Preferences.h>
```

Functions

- void set_value (const char used_namespace[14], const char key_nvs[14], String value)

  store given value of a key in a namespace
- String get_value (const char used_namespace[14], const char key_nvs[14])

  retrieve given value of a key in a namespace
- void set_integer (const char used_namespace[14], const char key_nvs[14], unsigned int value)

  store given int value of a key in a namespace
- unsigned int get_integer (const char used_namespace[14], const char key_nvs[14])

  retrieve given int value of a key in a namespace

Variables

- Preferences preferences

  Initialize a Preferences Object to store the data into ESP32 flash.

6.6.1 Detailed Description

function to store and read nvs flash variables

This is the files that contains the functions to save and read data from the nvs flash.

Date

19.08.2019

6.6.2 Function Documentation

6.6.2.1 get_integer()

```
unsigned int get_integer (
            const char used_namespace[14],
            const char key_nvs[14] )
```

retrieve given int value of a key in a namespace

Parameters

used_namespace	the intended namespace
key_nvs	the key to be retrieved

Returns

int retrieved integer value

Definition at line 102 of file nvs_storage.h.

6.6.2.2 get_value()

```
String get_value (
            const char used_namespace[14],
            const char key_nvs[14] )
```

retrieve given value of a key in a namespace

Parameters

used_namespace	the intended namespace to get values
key_nvs	the String key to be retrieved

Returns

String retrieved string value

Definition at line 51 of file nvs_storage.h.

6.6.2.3 set_integer()

```
void set_integer (
            const char used_namespace[14],
            const char key_nvs[14],
            unsigned int value )
```

store given int value of a key in a namespace

Parameters

used_namespace	the intended namespace to be used
key_nvs	the key to be stored
value	the integer key to be stored

Returns

void

Definition at line 80 of file nvs_storage.h.

6.6.2.4 set_value()

```
void set_value (
            const char used_namespace[14],
            const char key_nvs[14],
            String value )
```

store given value of a key in a namespace

Parameters

used_namespace	the intended namespace to store the values
key_nvs	the key to be stored at
value	the String key to be stored

Returns

void

Definition at line 27 of file nvs_storage.h.

6.7 esp32-stationary/send_data.h File Reference

function to send json data to an API

Functions

- void sendData (const char *data_json, char destination[ ], String pin)
  Send JSON data to API.

Variables

- String esp32_chipid
  Externally define variable for Chip ID.
- bool use_bme280
  Externally define variable to check if using bme280.
- bool use_dht22
  Externally define variable to check if using dht22.

6.7.1 Detailed Description

function to send json data to an API

This is the files that contains the functions to send json data to an API.

Date

19.08.2019

6.7.2 Function Documentation

6.7.2.1 sendData()

```
void sendData (
            const char * data_json,
            char destination[],
            String pin )
```

Send JSON data to API.

Parameters

data_json	the JSON Message to be sent
destination	the destination API madavi or luftdaten
pin	the pin-code values

Returns

void

Definition at line 29 of file send_data.h.

6.7.3 Variable Documentation

6.7.3.1 esp32_chipid

```
String esp32_chipid
```

Externally define variable for Chip ID.

Externally define variable for Chip ID.

Definition at line 1227 of file main.h.

6.7.3.2 use_bme280

```
bool use_bme280
```

Externally define variable to check if using bme280.

Externally define variable to check if using bme280.

Definition at line 1149 of file main.h.

6.7.3.3 use_dht22

```
bool use_dht22
```

Externally define variable to check if using dht22.

Externally define variable to check if using dht22.

Definition at line 1151 of file main.h.

6.8 esp32-stationary/sensor.h File Reference

variable and definitions for sensors

Macros

- #define DHT_PIN 21

  DHT22 sensor communication pin.
- #define BME_ADDR 0x77

  BME280 I2C Address 0x77 for Red, 0x76 for Purple (chinese)
- #define BME_SDA 21

  BME280 I2C SDA pin.
- #define BME_SCL 22

  BME280 I2C SCL pin.
- #define SDS011_TXD 13

  SDS011 TXD pin is connected at RXD of Serial2 Object.
- #define SDS011_RXD 17

  SDS011 RXD pin is connected at TXD of Serial2 Object (no need to physically connect)

Functions

- HardwareSerial SDS_SERIAL (2)

  use Serial 2 Object
- DHT dht (DHT_PIN, DHT22)

  DHT22 Temperature & Humidity Sensor Object.

Variables

- SDS011 sds

  Dust sensor object.
- Adafruit_BME280 bme

  BME280 Pressure & Humidity Sensor Object.

6.8.1 Detailed Description

variable and definitions for sensors

This is the files that contains the variables, pins definitions and functions for sensors to be connected.

Date

19.08.2019

6.9 esp32-stationary/timer_definitions.h File Reference

variable and function definitions for timings

Functions

- void IRAM_ATTR resetModule ()

  ISR functions called for Reset ESP32.

Variables

- unsigned long previousMillis_sensor = 0

  will store the last time the event (in loop) was triggered for sending sensor values to FrostManager
- unsigned long previousMillis_send2madavi = 0

  will store the last time the event (in loop) was triggered sending sensor values to madavi
- unsigned long previousMillis_send2luftdaten = 0

  will store the last time the event (in loop) was triggered sending sensor values to luftdaten
- bool sensorInterval

  time interval to send values to Frostmanager
- bool send2madaviInterval

  time interval to send values to MADAVI
- bool send2luftdatenInterval

  time interval to send values to LUFTDATEN

6.9.1 Detailed Description

variable and function definitions for timings

This is the files that contains the variables, macros, pins definitions and functions for timing related routines.

Date

19.08.2019

6.9.2 Function Documentation

6.9.2.1 resetModule()

```
void IRAM_ATTR resetModule ( )
```

ISR functions called for Reset ESP32.

Returns

void

Definition at line 30 of file timer_definitions.h.

Index

_waitForConnect
    myAutoConnect, 30

BASE_URL
    main.h, 53
begin
    myAutoConnect, 30

create_json.h
    json_madavi, 34
    json_RHT, 34
    json_sds, 35
    use_bme280, 35
    use_dht22, 35
createEntities
    main.h, 53
createEntity
    main.h, 53
createOrPatchEntities
    main.h, 54
createOrUpdateEntities
    main.h, 54

display.h
    displayQRCode, 37
    generateQRCode, 37
displayQRCode
    display.h, 37

esp32-stationary.ino
    loop, 38
    setup, 38
esp32-stationary/create_json.h, 33
esp32-stationary/display.h, 36
esp32-stationary/esp32-stationary.ino, 37
esp32-stationary/main.h, 39
esp32-stationary/myAutoConnect.h, 59
esp32-stationary/nvs_storage.h, 60
esp32-stationary/send_data.h, 63
esp32-stationary/sensor.h, 64
esp32-stationary/timer_definitions.h, 65
esp32_chipid
    main.h, 58
    send_data.h, 64

generateQRCode
    display.h, 37
get_integer
    nvs_storage.h, 61
get_value
    nvs_storage.h, 61
getEntity
    main.h, 55
getEntityResponse
    main.h, 55

http_error
    main.h, 58

json_madavi
    create_json.h, 34
json_RHT
    create_json.h, 34
json_sds
    create_json.h, 35

license_type
    main.h, 58
loop
    esp32-stationary.ino, 38

main.h
    BASE_URL, 53
    createEntities, 53
    createEntity, 53
    createOrPatchEntities, 54
    createOrUpdateEntities, 54
    esp32_chipid, 58
    getEntity, 55
    getEntityResponse, 55
    http_error, 58
    license_type, 58
    patchEntity, 55
    postObservation, 57
    toHEXSHA, 57
    use_bme280, 58
    use_dht22, 59
myAutoConnect, 29
    _waitForConnect, 30
    begin, 30
    myAutoConnect, 29

nvs_storage.h
    get_integer, 61
    get_value, 61
    set_integer, 62
    set_value, 62

patchEntity
    main.h, 55
postObservation
    main.h, 57

resetModule
timer_definitions.h, 66

send_data.h
esp32_chipid, 64
sendData, 63
use_bme280, 64
use_dht22, 64
sendData
send_data.h, 63
set_integer
nvs_storage.h, 62
set_value
nvs_storage.h, 62
setup
esp32-stationary.ino, 38

timer_definitions.h
resetModule, 66
toHEXSHA
main.h, 57

use_bme280
create_json.h, 35
main.h, 58
send_data.h, 64
use_dht22
create_json.h, 35
main.h, 59
send_data.h, 64