

ESP32 station

0.1

Generated by Doxygen 1.9.1

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 measurements Struct Reference	5
3.2 statusStruct Struct Reference	6
4 File Documentation	7
4.1 src/communication.hpp File Reference	7
4.1.1 Detailed Description	8
4.1.2 Function Documentation	8
4.1.2.1 callback()	8
4.1.2.2 log()	9
4.1.2.3 uploadData()	10
4.2 src/config.hpp File Reference	11
4.2.1 Detailed Description	13
4.2.2 Macro Definition Documentation	13
4.2.2.1 DAYLIGHT_OFFSET_SEC	13
4.2.2.2 GMT_OFFSET_SEC	13
4.2.2.3 MEASUREMENTS_COUNTER	13
4.2.2.4 SEA_LEVEL_PRESSURE	14
4.2.2.5 SSID	14
4.2.2.6 uS_TO_S_FACTOR	14
4.3 src/datastruct.hpp File Reference	14
4.3.1 Detailed Description	15
4.4 src/jsonhelper.hpp File Reference	15
4.4.1 Detailed Description	16
4.4.2 Function Documentation	16
4.4.2.1 addEventToJSON() [1/2]	16
4.4.2.2 addEventToJSON() [2/2]	16
4.5 src/main.cpp File Reference	17
4.5.1 Detailed Description	18
4.5.2 Function Documentation	18
4.5.2.1 backup()	18
4.5.2.2 measure()	19
4.5.2.3 readBatteryLevel()	19
4.5.2.4 setSleepTimer()	20
4.5.2.5 setup()	21
4.5.2.6 sleep()	21
4.5.3 Variable Documentation	22

4.5.3.1 offlineCounter	22
4.6 src/rtcmodule.hpp File Reference	22
4.6.1 Detailed Description	23
4.6.2 Function Documentation	23
4.6.2.1 RTCGetString()	23
4.6.2.2 RTCSetTimeOnline()	23
4.7 src/sps30.hpp File Reference	24
4.7.1 Detailed Description	24
4.7.2 Function Documentation	25
4.7.2.1 sps30ModuleInfo()	25
4.7.2.2 sps30Prepare()	25
4.7.2.3 sps30ReadNewData()	26
4.8 src/statstruct.hpp File Reference	26
4.8.1 Detailed Description	27
4.9 src/storage.hpp File Reference	27
4.9.1 Detailed Description	28
4.9.2 Function Documentation	28
4.9.2.1 cardClearFile()	28
4.9.2.2 cardLoadJSONFromFile()	29
4.9.2.3 cardPrepare()	30
4.9.2.4 cardWriteJSONToFile()	30
Index	33

Chapter 1

Class Index

1.1 Class List

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

measurements	5
statusStruct	6

Chapter 2

File Index

2.1 File List

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

src/ communication.hpp	
This file contains function for communication with server	7
src/ config.hpp	
This file holds config values such as server ip, delay values, pin values etc	11
src/ datastruct.hpp	
Holds structure measurements	14
src/ jsonhelper.hpp	
Contains functions for working with JSON document	15
src/ main.cpp	
Main file	17
src/ rtcmodule.hpp	
Function for working with RTC module	22
src/ sps30.hpp	
Function for working with sps30 sensor	24
src/ statstruct.hpp	
Holds structure statusStruct	26
src/ storage.hpp	
Function for working with storage and JSON documents	27

Chapter 3

Class Documentation

3.1 measurements Struct Reference

Public Attributes

- float [humidity](#) = NAN
Humidity level in %.
- char * [time](#)
Time in "yyyy-MM-dd HH:mm:ss" format.
- float [batteryLevel](#) = NAN
Battery level in %.
- float [temperature](#) = NAN
Temperature in °C.
- float [pressure](#) = NAN
Pressure in hpa.
- float [altitude](#) = NAN
Altitude in meters.
- uint32_t [gasResistance](#) = 0
Gas resistance.
- sps30_measurement [spsData](#)
Contains data from sps30 sensor.

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

- [src/datastruct.hpp](#)

3.2 statusStruct Struct Reference

Public Attributes

- bool [cardAvailable](#) = true
Whenever or not SD card is available.
- bool [bmeAvailable](#) = true
Whenever or not BME680 is available.
- bool [rtcAvailable](#) = true
Whenever or not RTC module is available.
- bool [spsAvailable](#) = true
Whenever or not is sps30 sensor is available.
- bool [problemOccured](#) = false
Whenever or not problem occurred.

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

- [src/statstruct.hpp](#)

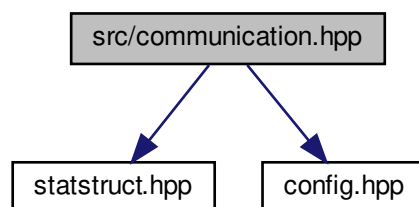
Chapter 4

File Documentation

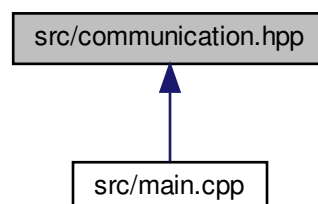
4.1 src/communication.hpp File Reference

This file contains function for communication with server.

```
#include "statstruct.hpp"
#include "config.hpp"
Include dependency graph for communication.hpp:
```



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



Functions

- void `callback` (String &topic, String &payload)
Callback function for MQTT client.
- bool `uploadData` (DynamicJsonDocument &doc, char *topic)
Function for uploading measurment data to MQTT topic on server.
- void `log` (const char *message, bool newline=true, const char *topic=LOG_TOPIC)
Is used to print log output to serial communication.

Variables

- MQTTClient `mqttClient`

4.1.1 Detailed Description

This file contains function for communication with server.

4.1.2 Function Documentation

4.1.2.1 `callback()`

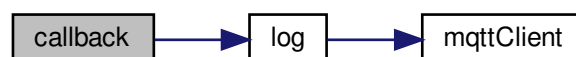
```
void callback (  
    String & topic,  
    String & payload )
```

Callback function for MQTT client.

Parameters

<i>topic</i>	MQTT topic name
<i>payload</i>	content

Here is the call graph for this function:



4.1.2.2 log()

```
void log (
    const char * message,
    bool newLine = true,
    const char * topic = LOG_TOPIC )
```

Is used to print log output to serial communication.

If connection to server is available log is also sent to MQTT topic.

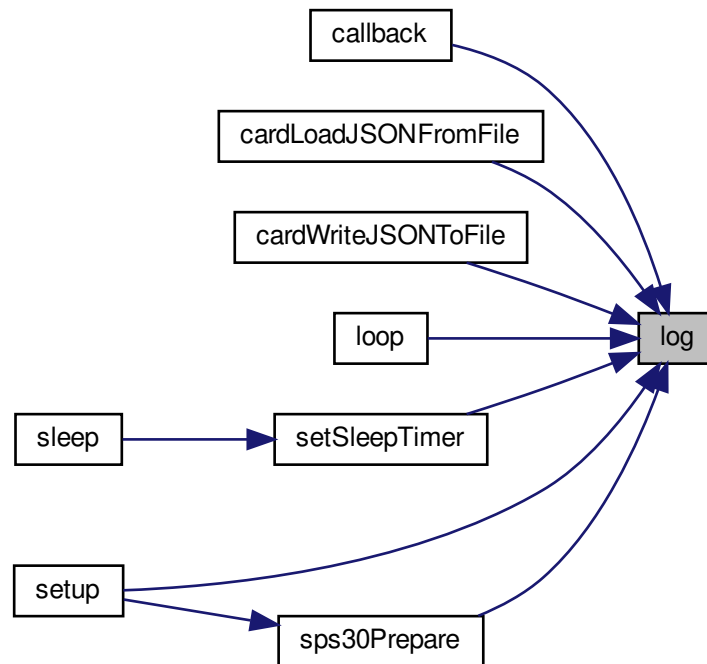
Parameters

<i>message</i>	Content of log
<i>newLine</i>	Whether or not print log to new line in serial communication DEFAULT VALUE = true.
<i>topic</i>	MQTT topic name DEFAULT VALUE = LOG_TOPIC.

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.2.3 uploadData()

```

bool uploadData (
    DynamicJsonDocument & doc,
    char * topic )

```

Function for uploading measurment data to MQTT topic on server.

Parameters

<i>doc</i>	JSON document with data.
<i>topic</i>	MQTT topic name.

Returns

Wherever or not upload was successful.

Here is the call graph for this function:



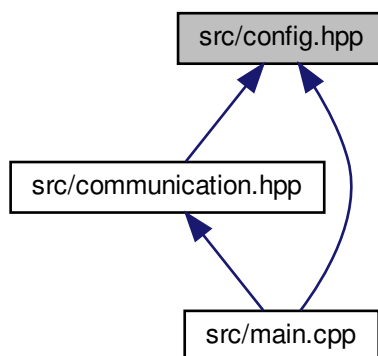
Here is the caller graph for this function:



4.2 src/config.hpp File Reference

This file holds config values such as server ip, delay values, pin values etc.

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



Macros

- `#define DEBUG 1`
Turns on or off debug logs in serial communication.
- `#define SSID "Damian"`
< Wi-Fi SSID.
- `#define WIFI_PASSWD "Ahoj123123"`
NTP server address.
- `#define NTP_SERVER "0.cz.pool.ntp.org"`
MQTT client ID.
- `#define MQTT_ID "esp32"`
MQTT client name.
- `#define MQTT_NAME "esp32"`
MQTT server IP address.
- `#define MQTT_SERVER "192.168.0.38"`
MQTT server port.
- `#define MQTT_PORT 1883`
Station hostname in local network.
- `#define HOSTNAME "esp32"`
MQTT packet size in bytes.
- `#define MQTT_PACKET_SIZE 512`
Topic name for logs.
- `#define LOG_TOPIC "esp32/log"`
Topic name for measured data.
- `#define DATA_TOPIC "esp32/data"`
Topic name for error reports.
- `#define REPORT_TOPIC "esp32/report"`
Quality of service (QoS) for MQTT publish communication.
- `#define MQTT_PUB_QOS 1`
Quality of service (QoS) for MQTT subscribe communication.
- `#define MQTT_SUB_QOS 1`
Connection timeout.
- `#define MQTT_TIMEOUT 5000`
- `#define uS_TO_S_FACTOR (uint64_t)1000000`
< Holds amount of uS in second.
- `#define TIME_TO_SLEEP_DEFAULT (uint64_t)60`
Amount of seconds for station to sleep when battery level is in level 1.
- `#define TIME_TO_SLEEP_ONE (uint64_t)60`
Amount of seconds for station to sleep when battery level is in level 2.
- `#define TIME_TO_SLEEP_TWO (uint64_t)300`
Amount of seconds for station to sleep when battery level is in level 3.
- `#define TIME_TO_SLEEP_THREE (uint64_t)600`
Define upper level of battery.
- `#define UPPER_LEVEL 90`
Define middle level of battery.
- `#define LOWER_LEVEL 80`
- `#define MEASUREMENTS_COUNTER 1`
< Specifies how many measurments to make before uploading data.
- `#define BAUD_RATE 115200`
Battery analog pin value.
- `#define BATTERY_PIN 32`

- SD card CS pin.*
 - #define `SD_CS` 5
- JSON document buffer size in bytes for measured data.*
 - #define `JSON_DOC_SIZE_MEASUREMENTS` 512
- JSON document buffer size in bytes for error report.*
 - #define `JSON_DOC_SIZE_STATUS` 192
- Full path of storage location.*
 - #define `STORAGE_LOCATION` "/station"
- #define `SEA_LEVEL_PRESSURE` 1034
- Sea level of current location of station.*
 - #define `GMT_OFFSET_SEC` 3600
- GMT offset in seconds.*
 - #define `DAYLIGHT_OFFSET_SEC` 3600
- UTC daylight offset.*
 - #define `CPU_SPEED` 80
- CPU speed in MHz Recommended speed is 240, minimum is 80.*

4.2.1 Detailed Description

This file holds config values such as server ip, delay values, pin values etc.

4.2.2 Macro Definition Documentation

4.2.2.1 DAYLIGHT_OFFSET_SEC

```
#define DAYLIGHT_OFFSET_SEC 3600
```

UTC daylight offset.

Is used to set time from NTP server.

4.2.2.2 GMT_OFFSET_SEC

```
#define GMT_OFFSET_SEC 3600
```

GMT offset in seconds.

Is used to set time from NTP seWho Will Save Us Nowrver.

4.2.2.3 MEASUREMENTS_COUNTER

```
#define MEASUREMENTS_COUNTER 1
```

< Specifies how many measurments to make before uploading data.

Baud rate for serial communication.

4.2.2.4 SEA_LEVEL_PRESSURE

```
#define SEA_LEVEL_PRESSURE 1034
```

Sea level of current location of station.

Is used to calibrate sensors.

4.2.2.5 SSID

```
#define SSID "Damian"
```

< Wi-Fi SSID.

Wi-Fi password.

4.2.2.6 uS_TO_S_FACTOR

```
#define uS_TO_S_FACTOR (uint64_t)1000000
```

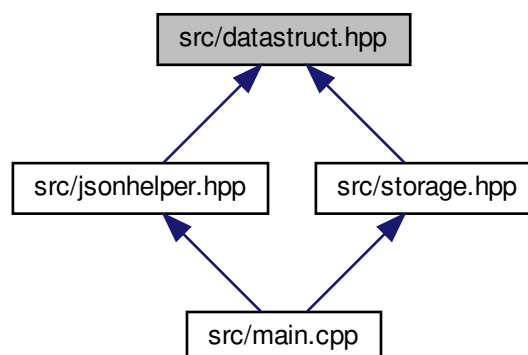
< Holds amount of uS in second.

Default amount of seconds for station to sleep.

4.3 src/datastruct.hpp File Reference

Holds structure measurements.

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



Classes

- struct [measurements](#)

4.3.1 Detailed Description

Holds structure measurements.

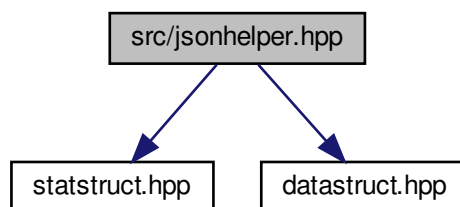
4.4 src/jsonhelper.hpp File Reference

Contains functions for working with JSON document.

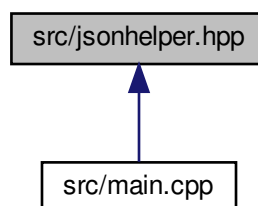
```
#include "statstruct.hpp"
```

```
#include "datastruct.hpp"
```

Include dependency graph for jsonhelper.hpp:



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



Functions

- void [addEventToJSON](#) (DynamicJsonDocument &doc, [measurements](#) &event)
Append measurements to JSON document.
- void [addEventToJSON](#) (DynamicJsonDocument &doc, [statusStruct](#) &status)
Append status to JSON document.

4.4.1 Detailed Description

Contains functions for working with JSON document.

4.4.2 Function Documentation

4.4.2.1 addEventToJSON() [1/2]

```
void addEventToJSON (
    DynamicJsonDocument & doc,
    measurements & event )
```

Append measurements to JSON document.

Parameters

<i>doc</i>	JSON document with data
<i>event</i>	struct with measurement data

Here is the caller graph for this function:



4.4.2.2 addEventToJSON() [2/2]

```
void addEventToJSON (
    DynamicJsonDocument & doc,
    statusStruct & status )
```

Append status to JSON document.

Parameters

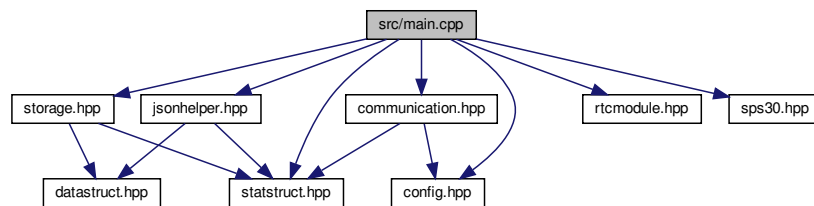
<i>doc</i>	JSON document with data
<i>status</i>	struct with status data

4.5 src/main.cpp File Reference

Main file.

```
#include "communication.hpp"
#include "statstruct.hpp"
#include "jsonhelper.hpp"
#include "rtcmodule.hpp"
#include "storage.hpp"
#include "config.hpp"
#include "sps30.hpp"
```

Include dependency graph for main.cpp:



Functions

- SDFS [card](#) (SD)
SD card module.
- MQTTClient [mqttClient](#) (MQTT_PACKET_SIZE)
MQTT client.
- void [setupWifi](#) ()
Is used to attempt to establish Wi-Fi connection.
- void [measure](#) ([measurements](#) &data, RTC_DS3231 &rtc, Adafruit_BME680 &bme)
Get reading from all available sensors and modules.
- double [readBatteryLevel](#) ()
Read battery level.
- int [setSleepTimer](#) (float batteryLevel)
Determine and set the time to put station to sleep.
- void [backup](#) (DynamicJsonDocument &doc)
Save JSON document on SD card.
- void [sleep](#) (float batteryLevel)
Prepare MCU for sleep, set deep sleep interval according to battery level.
- void [setup](#) ()
Setup function Is used to setup pin modes, MCU, and initialize sensors and modules.
- void [loop](#) ()
Main loop function.

Variables

- RTC_DATA_ATTR unsigned int [offlineCounter](#) = 0
Offline counter stored in internal RTC memory of ESP32, counts how many times station measured.
- WiFiClient [wifiClient](#)
Wi-Fi client.
- RTC_DS3231 [rtc](#)
RTC module.
- Adafruit_BME680 [bme](#)
BME680 sensor.
- [statusStruct](#) [status](#)
Struct that contains availability status for each sensor or module.

4.5.1 Detailed Description

Main file.

4.5.2 Function Documentation

4.5.2.1 backup()

```
void backup (
    DynamicJsonDocument & doc )
```

Save JSON document on SD card.

Parameters

<i>doc</i>	
------------	--

Here is the caller graph for this function:



4.5.2.2 measure()

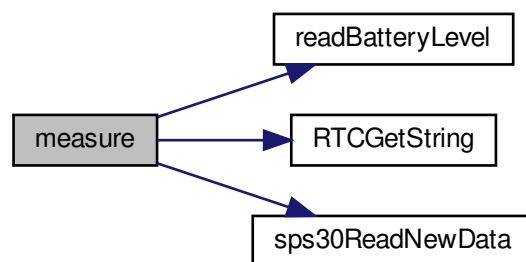
```
void measure (
    measurements & data,
    RTC_DS3231 & rtc,
    Adafruit_BME680 & bme )
```

Get reading from all available sensors and modules.

Parameters

<i>data</i>	Data Struct in which measured data is stored
<i>rtc</i>	rtc module
<i>bme</i>	BME680 sensor

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.2.3 readBatteryLevel()

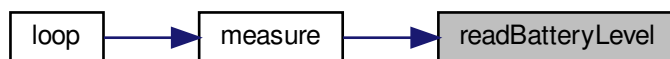
```
double readBatteryLevel ( )
```

Read battery level.

Returns

battery level in %

Here is the caller graph for this function:

**4.5.2.4 setSleepTimer()**

```
int setSleepTimer (  
    float batteryLevel )
```

Determine and set the time to put station to sleep.

Parameters

<i>batteryLevel</i>	battery level in %
---------------------	--------------------

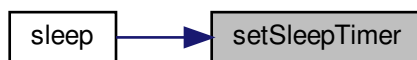
Returns

Sleep mode

Here is the call graph for this function:



Here is the caller graph for this function:

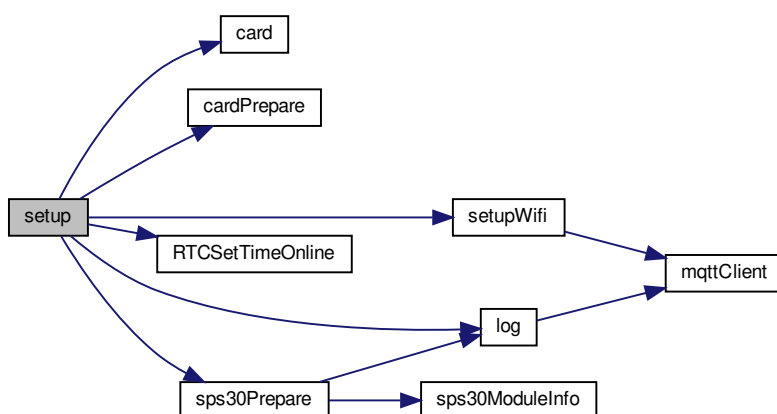


4.5.2.5 setup()

```
void setup ( )
```

Setup function is used to setup pin modes, MCU, and initialize sensors and modules.

Establish Wi-Fi connection and connection to MQTT broker. Here is the call graph for this function:



4.5.2.6 sleep()

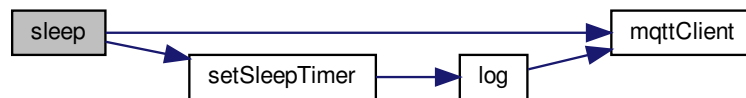
```
void sleep (
    float batteryLevel )
```

Prepare MCU for sleep, set deep sleep interval according to battery level.

Parameters

<i>batteryLevel</i>	Battery voltage level in %.
---------------------	-----------------------------

Here is the call graph for this function:



4.5.3 Variable Documentation

4.5.3.1 offlineCounter

```
RTC_DATA_ATTR unsigned int offlineCounter = 0
```

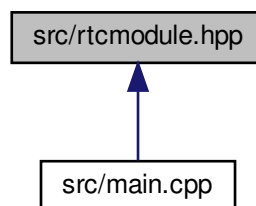
Offline counter stored in internal RTC memory of ESP32, counts how many times station measured.

Maximum amount of measurements is specified by MEASUREMENTS_COUNTER macro.

4.6 src/rtcmodule.hpp File Reference

Function for working with RTC module.

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



Functions

- void [RTCSetTimeOnline](#) (tm &timeStr)
Set time in RTC module from BTP server.
- char * [RTCGetString](#) ()
Get string representation of time in "yyyy-MM-dd HH:mm:ss" format.
- char * [RTCGetTimestamp](#) ()

Variables

- RTC_DS3231 [rtc](#)
RTC module.

4.6.1 Detailed Description

Function for working with RTC module.

4.6.2 Function Documentation

4.6.2.1 RTCGetString()

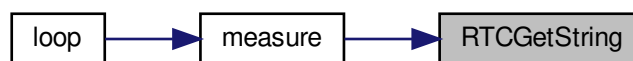
```
char* RTCGetString ( )
```

Get string representation of time in "yyyy-MM-dd HH:mm:ss" format.

Returns

string representation of time

Here is the caller graph for this function:



4.6.2.2 RTCSetTimeOnline()

```
void RTCSetTimeOnline (
    tm & timeStr )
```

Set time in RTC module from BTP server.

Parameters

<i>timeStr</i>	time struct
----------------	-------------

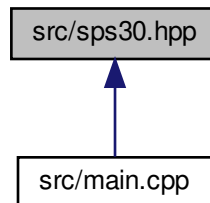
Here is the caller graph for this function:



4.7 src/sps30.hpp File Reference

Function for working with sps30 sensor.

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



Functions

- bool [sps30Prepare](#) ()
Initialize sps30 sensor.
- bool [sps30ReadNewData](#) (sps30_measurement &data)
Read new data from sps30 sensor.
- const char * [sps30ModuleInfo](#) ()
Get string representation of serial number and firmware version of sps30 sensor.

4.7.1 Detailed Description

Function for working with sps30 sensor.

4.7.2 Function Documentation

4.7.2.1 sps30ModuleInfo()

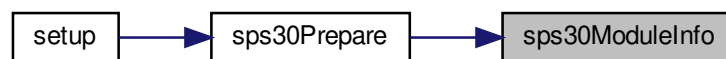
```
const char* sps30ModuleInfo ( )
```

Get string representation of serial number and firmware version of sps30 sensor.

Returns

string representation of serial number and firmware version of sps30 sensor.

Here is the caller graph for this function:



4.7.2.2 sps30Prepare()

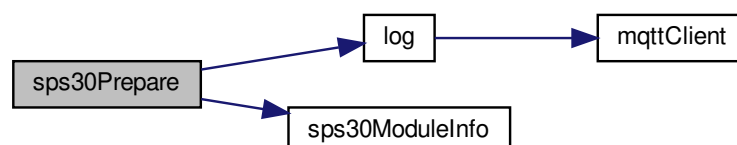
```
bool sps30Prepare ( )
```

Initialize sps30 sensor.

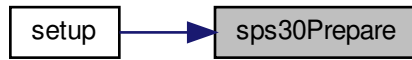
Returns

Whatever or not initialization was successful.

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.2.3 sps30ReadNewData()

```
bool sps30ReadNewData (
    sps30_measurement & data )
```

Read new data from sps30 sensor.

Parameters

<i>data</i>	sps30 data struct.
-------------	--------------------

Returns

Whatever or not measurment was successful.

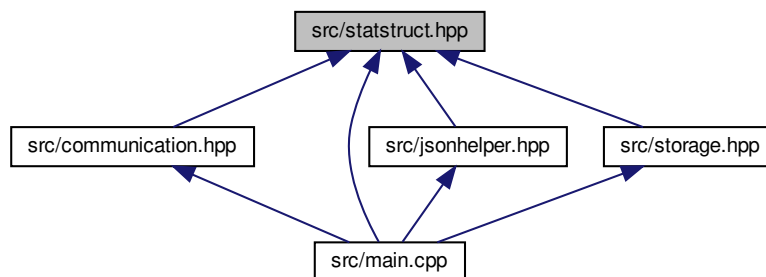
Here is the caller graph for this function:



4.8 src/statstruct.hpp File Reference

Holds structure [statusStruct](#).

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



Classes

- struct [statusStruct](#)

4.8.1 Detailed Description

Holds structure [statusStruct](#).

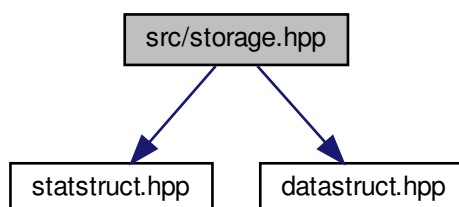
4.9 src/storage.hpp File Reference

Function for working with storage and JSON documents.

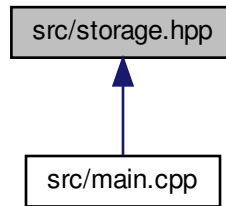
```
#include "statstruct.hpp"
```

```
#include "datastruct.hpp"
```

Include dependency graph for `storage.hpp`:



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



Functions

- bool `cardPrepare` ()
Initialize and prepare SD card.
- bool `cardWriteJSONToFile` (DynamicJsonDocument &doc, const char *fileName)
Store measured data into file on SD card.
- bool `cardLoadJSONFromFile` (DynamicJsonDocument &doc, char *fileName)
Load measured data from file on SD card.
- bool `cardClearFile` (const char *fileName)
Delete file on SD card.

Variables

- SDFS `card`

4.9.1 Detailed Description

Function for working with storage and JSON documents.

4.9.2 Function Documentation

4.9.2.1 `cardClearFile()`

```
bool cardClearFile (  
    const char * fileName )
```

Delete file on SD card.

Parameters

<i>fileName</i>	Full path to file
-----------------	-------------------

Returns

Whatever or not operation was successful

Here is the call graph for this function:



4.9.2.2 cardLoadJSONFromFile()

```
bool cardLoadJSONFromFile (
    DynamicJsonDocument & doc,
    char * fileName )
```

Load measured data from file on SD card.

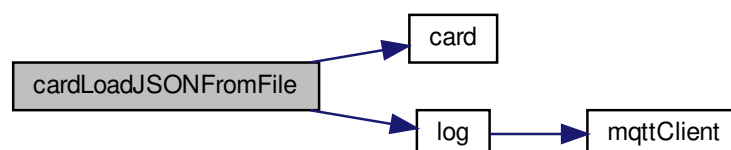
Parameters

<i>doc</i>	JSON document into which data is stored
<i>fileName</i>	Full path to file

Returns

Whatever or not operation was successful

Here is the call graph for this function:



4.9.2.3 cardPrepare()

```
bool cardPrepare ( )
```

Initialize and prepare SD card.

Returns

Whatever or not initialization was successful

Here is the caller graph for this function:



4.9.2.4 cardWriteJSONToFile()

```
bool cardWriteJSONToFile (
    DynamicJsonDocument & doc,
    const char * fileName )
```

Store measured data into file on SD card.

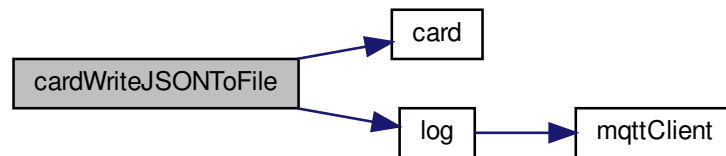
Parameters

<i>doc</i>	JSON document with data
<i>fileName</i>	Full path to file

Returns

Whatever or not operation was successful

Here is the call graph for this function:



Index

- addEventToJSON
 - jsonhelper.hpp, [16](#)
- backup
 - main.cpp, [18](#)
- callback
 - communication.hpp, [8](#)
- cardClearFile
 - storage.hpp, [28](#)
- cardLoadJSONFromFile
 - storage.hpp, [29](#)
- cardPrepare
 - storage.hpp, [30](#)
- cardWriteJSONToFile
 - storage.hpp, [30](#)
- communication.hpp
 - callback, [8](#)
 - log, [8](#)
 - uploadData, [10](#)
- config.hpp
 - DAYLIGHT_OFFSET_SEC, [13](#)
 - GMT_OFFSET_SEC, [13](#)
 - MEASUREMENTS_COUNTER, [13](#)
 - SEA_LEVEL_PRESSURE, [13](#)
 - SSID, [14](#)
 - uS_TO_S_FACTOR, [14](#)
- DAYLIGHT_OFFSET_SEC
 - config.hpp, [13](#)
- GMT_OFFSET_SEC
 - config.hpp, [13](#)
- jsonhelper.hpp
 - addEventToJSON, [16](#)
- log
 - communication.hpp, [8](#)
- main.cpp
 - backup, [18](#)
 - measure, [18](#)
 - offlineCounter, [22](#)
 - readBatteryLevel, [19](#)
 - setSleepTimer, [20](#)
 - setup, [21](#)
 - sleep, [21](#)
- measure
 - main.cpp, [18](#)
- measurements, [5](#)
- MEASUREMENTS_COUNTER
 - config.hpp, [13](#)
- offlineCounter
 - main.cpp, [22](#)
- readBatteryLevel
 - main.cpp, [19](#)
- RTCGetString
 - rtcmodule.hpp, [23](#)
- rtcmodule.hpp
 - RTCGetString, [23](#)
 - RTCSetTimeOnline, [23](#)
- RTCSetTimeOnline
 - rtcmodule.hpp, [23](#)
- SEA_LEVEL_PRESSURE
 - config.hpp, [13](#)
- setSleepTimer
 - main.cpp, [20](#)
- setup
 - main.cpp, [21](#)
- sleep
 - main.cpp, [21](#)
- sps30.hpp
 - sps30ModuleInfo, [25](#)
 - sps30Prepare, [25](#)
 - sps30ReadNewData, [26](#)
- sps30ModuleInfo
 - sps30.hpp, [25](#)
- sps30Prepare
 - sps30.hpp, [25](#)
- sps30ReadNewData
 - sps30.hpp, [26](#)
- src/communication.hpp, [7](#)
- src/config.hpp, [11](#)
- src/datastruct.hpp, [14](#)
- src/jsonhelper.hpp, [15](#)
- src/main.cpp, [17](#)
- src/rtcmodule.hpp, [22](#)
- src/sps30.hpp, [24](#)
- src/statstruct.hpp, [26](#)
- src/storage.hpp, [27](#)
- SSID
 - config.hpp, [14](#)
- statusStruct, [6](#)
- storage.hpp
 - cardClearFile, [28](#)
 - cardLoadJSONFromFile, [29](#)
 - cardPrepare, [30](#)

cardWriteJSONToFile, [30](#)

uploadData

communication.hpp, [10](#)

uS_TO_S_FACTOR

config.hpp, [14](#)