# Weather station

1

Generated by Doxygen 1.8.20

# Chapter 1

# Data Structure Index

## 1.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all files with brief descriptions:

# Chapter 3

# Data Structure Documentation

## 3.1 data Struct Reference

### Data Fields

- float humidity
- float temperature

### 3.1.1 Detailed Description

The structure that will hold the data of the made measurements.

### 3.1.2 Field Documentation

#### 3.1.2.1 humidity

```
float data::humidity
```

#### 3.1.2.2 temperature

```
float data::temperature
```

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

- main/si7021.c

## 3.2 ldr Struct Reference

### Data Fields

- uint32_t lux

### 3.2.1 Detailed Description

The structure that will hold the data of the made measurement.

### 3.2.2 Field Documentation

#### 3.2.2.1 lux

```
uint32_t ldr::lux
```

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

- main/ldr.c

# Chapter 4

# File Documentation

## 4.1 main/CMakeLists.txt File Reference

## 4.2 main/ldr.c File Reference

Reads the voltage difference across the ldr and a 100K resistor.

```
#include "ldr.h"
#include <string.h>
#include <stdio.h>
#include <FreeRTOS.h>
#include <freertos/task.h>
#include <freertos/queue.h>
```

### Data Structures

- struct ldr

### Functions

- void ldr_task (void ∗pvParameters)
  *A FreeRTOS task for measuring lux using ADC.*

### Variables

- static const char ∗ JSON_TEMPLATE = "{\"type\":\"%s\",\"value\":\"%s\"}"

### 4.2.1 Detailed Description

Reads the voltage difference across the ldr and a 100K resistor.

**Author**

Bedirhan Dincer

### 4.2.2 Function Documentation

#### 4.2.2.1 ldr_task()

```
void ldr_task (
            void * pvParameters )
```

A FreeRTOS task for measuring lux using ADC.

**Parameters**

| | |
|---|---|
| *pvParameters* | contains a reference to the queue. |

### 4.2.3 Variable Documentation

#### 4.2.3.1 JSON_TEMPLATE

```
const char* JSON_TEMPLATE = "{\"type\":\"%s\",\"value\":\"%s\"}"  [static]
```

A reusable JSON template to send a JSON message onto the queue.

## 4.3 main/ldr.h File Reference

```
#include "esp_system.h"
```

### Functions

- void ldr_task (void *pvParameters)
  *A FreeRTOS task for measuring lux using ADC.*

### 4.3.1 Function Documentation

#### 4.3.1.1 ldr_task()

```
void ldr_task (
            void * pvParameters )
```

A FreeRTOS task for measuring lux using ADC.

**Parameters**

| | |
|---|---|
| *pvParameters* | contains a reference to the queue. |

## 4.4 main/main.c File Reference

```
#include "sender.h"
#include "si7021.h"
#include "ldr.h"
#include "nvs_flash.h"
#include "esp_log.h"
```

### Functions

- void app_main (void)

  *The main entry point for creating the queue and the tasks.*

### Variables

- static xQueueHandle queue

### 4.4.1 Function Documentation

#### 4.4.1.1 app_main()

```
void app_main (
            void  )
```

The main entry point for creating the queue and the tasks.

### 4.4.2 Variable Documentation

#### 4.4.2.1 queue

```
xQueueHandle queue  [static]
```

Create the queue for sending and receiving message between FreeRTOS tasks

## 4.5 main/sender.c File Reference

Sends measurement data to the server and sleeps for 60 seconds.

```
#include "sender.h"
#include <string.h>
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "freertos/queue.h"
#include "esp_wifi.h"
#include "esp_log.h"
#include "esp_sleep.h"
#include "nvs_flash.h"
#include <netdb.h>
```

### Macros

- #define WIFI_SSID CONFIG_WIFI_SSID

  *SSID of Wi-Fi AP.*
- #define WIFI_PASS CONFIG_WIFI_PASSWORD

  *Password of Wi-Fi AP.*
- #define WEB_SERVER "europe-west1-tactical-crow-272620.cloudfunctions.net"

  *The root server.*
- #define WEB_PORT "80"

  *Port number for TCP connection.*
- #define WEB_URL "http://europe-west1-tactical-crow-272620.cloudfunctions.net/measurement"

  *HTTP API endpoint.*
- #define SA struct sockaddr

  *Structure for socket connection.*
- #define MAX_LINE 1024

  *Max bits allowed to send in a HTTP POST request.*
- #define MAX_REQUESTS 2

  *Max amount of requests possible.*
- #define SLEEP_DURATION 60

  *The deep sleep duration.*

### Functions

- static esp_err_t event_handler (void ∗ctx, system_event_t ∗event)

  *An ESP event handler that communicates with the Wi-Fi driver.*
- static void initialise_wifi (void)

  *Setup a Wi-Fi connection with a AP.*
- void http_post_task (void ∗pvParameters)

  *A FreeRTOS task for sending the measurements to the server.*

## Variables

- static const char ∗ JSON_HEAD = "{\"measurements\":["

  *The JSON formatted message header.*
- static const char ∗ JSON_TAIL = "]}"

  *The JSON formatted message tail.*
- static const int32_t CONNECTED_BIT = BIT0

  *A 32-bit long with only the first bit set.*
- static xSemaphoreHandle mutex_bus

  *The mutex bus keeps the shared function protected for a moment.*
- static const char ∗ TAG = "sender"

  *The TAG that is meant as a tag for who is writing to stdout.*
- static EventGroupHandle_t wifi_event_group

  *The event group handle that is responsible for receiving events from the Wi-Fi driver.*

### 4.5.1 Detailed Description

Sends measurement data to the server and sleeps for 60 seconds.

**Author**

Bedirhan Dincer

**Socket error code:**

1. See `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/lwip.`
`html#socket-error-reason-code`

### 4.5.2 Macro Definition Documentation

#### 4.5.2.1 MAX_LINE

```
#define MAX_LINE 1024
```

Max bits allowed to send in a HTTP POST request.

#### 4.5.2.2 MAX_REQUESTS

```
#define MAX_REQUESTS 2
```

Max amount of requests possible.

60 seconds a time.

#### 4.5.2.3 SA

```
#define SA struct sockaddr
```

Structure for socket connection.

#### 4.5.2.4 SLEEP_DURATION

```
#define SLEEP_DURATION 60
```

The deep sleep duration.

#### 4.5.2.5 WEB_PORT

```
#define WEB_PORT "80"
```

Port number for TCP connection.

#### 4.5.2.6 WEB_SERVER

```
#define WEB_SERVER "europe-west1-tactical-crow-272620.cloudfunctions.net"
```

The root server.

#### 4.5.2.7 WEB_URL

```
#define WEB_URL "http://europe-west1-tactical-crow-272620.cloudfunctions.net/measurement"
```

HTTP API endpoint.

#### 4.5.2.8 WIFI_PASS

```
#define WIFI_PASS CONFIG_WIFI_PASSWORD
```

Password of Wi-Fi AP.

#### 4.5.2.9 WIFI_SSID

```
#define WIFI_SSID CONFIG_WIFI_SSID
```

SSID of Wi-Fi AP.

### 4.5.3 Function Documentation

#### 4.5.3.1 event_handler()

```
static esp_err_t event_handler (
             void * ctx,
             system_event_t * event )  [static]
```

An ESP event handler that communicates with the Wi-Fi driver.

**Parameters**

| | |
|---|---|
| *ctx* | reserver for user. |
| *event* | application specified event callback. |

**Returns**

The operation was succesfull or failure.

#### 4.5.3.2 http_post_task()

```
void http_post_task (
             void * pvParameters )
```

A FreeRTOS task for sending the measurements to the server.

**Parameters**

| | |
|---|---|
| *pvParameters* | contains a reference to the queue. |

#### 4.5.3.3 initialise_wifi()

```
static void initialise_wifi (
             void  )  [static]
```

Setup a Wi-Fi connection with a AP.

### 4.5.4 Variable Documentation

#### 4.5.4.1 CONNECTED_BIT

```
const int32_t CONNECTED_BIT = BIT0  [static]
```

A 32-bit long with only the first bit set.

#### 4.5.4.2 JSON_HEAD

```
const char* JSON_HEAD = "{\"measurements\":["  [static]
```

The JSON formatted message header.

#### 4.5.4.3 JSON_TAIL

```
const char* JSON_TAIL = "]}"  [static]
```

The JSON formatted message tail.

#### 4.5.4.4 mutex_bus

```
xSemaphoreHandle mutex_bus  [static]
```

The mutex bus keeps the shared function protected for a moment.

#### 4.5.4.5 TAG

```
const char* TAG = "sender"  [static]
```

The TAG that is meant as a tag for who is writing to stdout.

#### 4.5.4.6 wifi_event_group

```
EventGroupHandle_t wifi_event_group  [static]
```

The event group handle that is responsible for receiving events from the Wi-Fi driver.

## 4.6 main/sender.h File Reference

```
#include "esp_event_loop.h"
```

### Functions

- void http_post_task (void *pvParameters)

  *A FreeRTOS task for sending the measurements to the server.*

### 4.6.1 Function Documentation

#### 4.6.1.1 http_post_task()

```
void http_post_task (
            void * pvParameters )
```

A FreeRTOS task for sending the measurements to the server.

**Parameters**

| | |
|---|---|
| *pvParameters* | contains a reference to the queue. |

## 4.7 main/si7021.c File Reference

Reads the temperature and humidity of the SI7021 sensor.

```
#include "si7021.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "esp_log.h"
#include "esp_system.h"
#include "nvs_flash.h"
```

### Data Structures

- struct data

## Macros

- #define I2C_MASTER_SCL_IO 5

  *GPIO5 is the I2C master clock line.*
- #define I2C_MASTER_SDA_IO 4

  *GPIO4 for i2c data line.*
- #define I2C_MASTER_NUM I2C_NUM_0

  *I2C port number for master dev.*
- #define I2C_MASTER_TX_BUF_DISABLE 0

  *I2C master do not need buffer.*
- #define I2C_MASTER_RX_BUF_DISABLE 0

  *I2C master do not need buffer.*
- #define WRITE_BIT I2C_MASTER_WRITE

  *I2C master write bit.*
- #define READ_BIT I2C_MASTER_READ

  *I2C master read bit.*
- #define ACK_CHECK_ENABLE 0x1

  *I2C master will check ack from slave.*
- #define ACK_CHECK_DISABLE 0x0

  *I2C master will not check ack from slave.*
- #define ACK_VAL 0x0

  *I2C ack value.*
- #define NACK_VAL 0x1

  *I2C nack value.*
- #define LAST_NACK_VAL 0x2

  *I2C last_nack value.*
- #define SI7021_SENSOR_ADDRESS 0x40

  *SI7021 register address definitions.*
- #define TEMP_MEASURE_HOLD 0xE3

  *Measure temperature address.*
- #define HUMD_MEASURE_HOLD 0xE5

  *Measure relative humidity address.*
- #define TEMP_PREV 0xE0

  *Measure from previous measurement address.*
- #define I2C_TIMEOUT 998

  *I2C error codes.*
- #define BAD_CRC 999

## Functions

- static esp_err_t i2c_master_init ()

  *I2C master initialization/configuation settings.*
- static esp_err_t i2c_master_measure_relative_humidity (i2c_port_t i2c_num, uint8_t ∗data)

  *I2C master measure and reading relative humidity.*
- static esp_err_t i2c_master_read_temperature_from_relative_humidity (i2c_port_t i2c_num, uint8_t ∗data)

  *Sequence to read temperature value from previous RH measurement.*
- static float get_relative_humidity ()

  *Measures the relative humidity.*
- static float get_temp_from_prev_hr_measurement ()

  *Reads temperature from previous humidity measurement.*
- void i2c_task (void ∗pvParameters)

  *A FreeRTOS task for measuring temperature and humidity using I2C.*

## Variables

- static const char ∗ JSON_TEMPLATE = "{\"type\":\"%s\",\"value\":\"%s\"}"

### 4.7.1 Detailed Description

Reads the temperature and humidity of the SI7021 sensor.

**Author**

Bedirhan Dincer

**Datasheet:**

The datasheet for the sensor is available at `https://www.silabs.com/documents/public/data-sheets/↩ Si7021-A20.pdf` last checked on: 16-09-2020

**Pin assignment:**

1. GPIO4 is assigned as a data signal of i2c master port.
2. GPIO5 is assigned as a clock signal of i2c master port.

**Connection:**

1. Connect sda/scl of sensor with GPIO4/GPIO5.
2. No need to add external pull-up resistors, driver will enable internal pull-up resistors.

**Test cases:**

1. Measuring relative humidity.
2. Reading temperature from previous measurement.

### 4.7.2 Macro Definition Documentation

#### 4.7.2.1 ACK_CHECK_DISABLE

```
#define ACK_CHECK_DISABLE 0x0
```

I2C master will not check ack from slave.

#### 4.7.2.2 ACK_CHECK_ENABLE

```
#define ACK_CHECK_ENABLE 0x1
```

I2C master will check ack from slave.

#### 4.7.2.3 ACK_VAL

```
#define ACK_VAL 0x0
```

I2C ack value.

#### 4.7.2.4 BAD_CRC

```
#define BAD_CRC 999
```

#### 4.7.2.5 HUMD_MEASURE_HOLD

```
#define HUMD_MEASURE_HOLD 0xE5
```

Measure relative humidity address.

#### 4.7.2.6 I2C_MASTER_NUM

```
#define I2C_MASTER_NUM I2C_NUM_0
```

I2C port number for master dev.

#### 4.7.2.7 I2C_MASTER_RX_BUF_DISABLE

```
#define I2C_MASTER_RX_BUF_DISABLE 0
```

I2C master do not need buffer.

#### 4.7.2.8 I2C_MASTER_SCL_IO

```
#define I2C_MASTER_SCL_IO 5
```

GPIO5 is the I2C master clock line.

#### 4.7.2.9 I2C_MASTER_SDA_IO

```
#define I2C_MASTER_SDA_IO 4
```

GPIO4 for i2c data line.

#### 4.7.2.10 I2C_MASTER_TX_BUF_DISABLE

```
#define I2C_MASTER_TX_BUF_DISABLE 0
```

I2C master do not need buffer.

#### 4.7.2.11 I2C_TIMEOUT

```
#define I2C_TIMEOUT 998
```

I2C error codes.

#### 4.7.2.12 LAST_NACK_VAL

```
#define LAST_NACK_VAL 0x2
```

I2C last_nack value.

#### 4.7.2.13 NACK_VAL

```
#define NACK_VAL 0x1
```

I2C nack value.

#### 4.7.2.14 READ_BIT

```
#define READ_BIT I2C_MASTER_READ
```

I2C master read bit.

#### 4.7.2.15 SI7021_SENSOR_ADDRESS

```
#define SI7021_SENSOR_ADDRESS 0x40
```

SI7021 register address definitions.

Master address

#### 4.7.2.16 TEMP_MEASURE_HOLD

```
#define TEMP_MEASURE_HOLD 0xE3
```

Measure temperature address.

#### 4.7.2.17 TEMP_PREV

```
#define TEMP_PREV 0xE0
```

Measure from previous measurement address.

#### 4.7.2.18 WRITE_BIT

```
#define WRITE_BIT I2C_MASTER_WRITE
```

I2C master write bit.

### 4.7.3 Function Documentation

#### 4.7.3.1 get_relative_humidity()

```
static float get_relative_humidity ( )  [static]
```

Measures the relative humidity.

**Returns**

The measured relative humdity in percentage

#### 4.7.3.2 get_temp_from_prev_hr_measurement()

```
static float get_temp_from_prev_hr_measurement ( )  [static]
```

Reads temperature from previous humidity measurement.

**Returns**

The measured temperature in celcius degrees

**Note**

This is however not a new measurement. It gets the value from the registers.

#### 4.7.3.3 i2c_master_init()

```
static esp_err_t i2c_master_init ( )  [static]
```

I2C master initialization/configuation settings.

**Returns**

The initialization has been succesfully made.

#### 4.7.3.4 i2c_master_measure_relative_humidity()

```
static esp_err_t i2c_master_measure_relative_humidity (
            i2c_port_t i2c_num,
            uint8_t * data )  [static]
```

I2C master measure and reading relative humidity.

**Parameters**

| | |
|---|---|
| *i2c_num* | i2c port number. |
| *data* | buffer contains the relative humidity value. |

**Returns**

The operation was succesfull or failure.

#### 4.7.3.5 i2c_master_read_temperature_from_relative_humidity()

```
static esp_err_t i2c_master_read_temperature_from_relative_humidity (
            i2c_port_t i2c_num,
            uint8_t * data )  [static]
```

Sequence to read temperature value from previous RH measurement.

**Parameters**

| | |
|---|---|
| *i2c_num* | i2c port number. |
| *data* | buffer contains the temperature value. |

**Returns**

The operation was succesfull or failure

#### 4.7.3.6 i2c_task()

```
void i2c_task (
            void * pvParameters )
```

A FreeRTOS task for measuring temperature and humidity using I2C.

**Parameters**

| | |
|---|---|
| *pvParameters* | contains a reference to the queue. |

### 4.7.4 Variable Documentation

#### 4.7.4.1 JSON_TEMPLATE

```
const char* JSON_TEMPLATE = "{\"type\":\"%s\",\"value\":\"%s\"}"  [static]
```

A reusable JSON template to send a JSON message onto the queue.

## 4.8 main/si7021.h File Reference

```
#include "esp_err.h"
#include "driver/i2c.h"
```

### Functions

- void i2c_task (void *pvParameters)

  *A FreeRTOS task for measuring temperature and humidity using I2C.*

### 4.8.1 Function Documentation

#### 4.8.1.1 i2c_task()

```
void i2c_task (
            void * pvParameters )
```

A FreeRTOS task for measuring temperature and humidity using I2C.

**Parameters**

| | |
|---|---|
| *pvParameters* | contains a reference to the queue. |

# Index