IoT Environmental Station 1.0

Generated by Doxygen 1.14.0

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 dht_data_t Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Field Documentation	5
3.1.2.1 humidity	5
3.1.2.2 temperature	6
3.1.2.3 timestamp	6
4 File Documentation	7
4.1 includes/common.c File Reference	7
4.1.1 Function Documentation	7
4.1.1.1 create_json_payload()	7
4.1.1.2 get_current_date_time()	8
4.1.1.3 setup_sntp()	8
4.1.1.4 time_sync_notification_cb()	8
4.2 common.c	8
4.3 includes/common.h File Reference	9
4.3.1 Macro Definition Documentation	10
4.3.1.1 MAX_Q_SIZE	10
4.3.1.2 MEASURE_INTERVAL	10
4.3.1.3 STACK_SIZE	10
4.3.1.4 TIMER_ID	10
4.3.1.5 WIFI_RECONNECT_INTERVAL_MS	10
4.3.2 Function Documentation	10
4.3.2.1 create_json_payload()	10
4.3.2.2 get_current_date_time()	11
4.3.2.3 setup_sntp()	11
4.4 common.h	11
4.5 includes/dht_manager.c File Reference	11
4.5.1 Function Documentation	12
4.5.1.1 setup_dht()	12
4.6 dht_manager.c	
4.7 includes/dht_manager.h File Reference	12
4.7.1 Macro Definition Documentation	
4.7.1.1 CONFIG_EXAMPLE_INTERNAL_PULLUP	13
4.7.1.2 CONFIG_EXAMPLE_TYPE_DHT11	
4.7.1.3 DHT_SENSOR_TYPE	13

4.7.1.4 ISO8601_STR_LEN	13
4.7.2 Function Documentation	14
4.7.2.1 setup_dht()	14
4.8 dht_manager.h	14
4.9 includes/display_manager.c File Reference	14
4.9.1 Function Documentation	15
4.9.1.1 init_oled_display()	15
4.9.1.2 show_dht_data()	15
4.10 display_manager.c	16
4.11 includes/display_manager.h File Reference	16
4.11.1 Macro Definition Documentation	17
4.11.1.1 I2C_SCL	17
4.11.1.2 I2C_SDA	17
4.11.2 Function Documentation	17
4.11.2.1 init_oled_display()	17
4.11.2.2 show_dht_data()	17
4.12 display_manager.h	18
4.13 includes/mqtt_manager.c File Reference	18
4.13.1 Function Documentation	19
4.13.1.1 mqtt_event_handler()	19
4.13.1.2 setup_mqtt()	19
4.13.2 Variable Documentation	19
4.13.2.1 client	19
4.13.2.2 MQTT_CONNECTED	19
4.14 mqtt_manager.c	20
4.15 includes/mqtt_manager.h File Reference	20
4.15.1 Function Documentation	21
4.15.1.1 setup_mqtt()	21
4.16 mqtt_manager.h	21
4.17 includes/wifi_manager.c File Reference	21
4.17.1 Function Documentation	22
4.17.1.1 setup_wifi()	22
4.17.1.2 wifi_event_handler()	22
4.17.2 Variable Documentation	22
4.17.2.1 s_wifi_event_group	22
4.18 wifi_manager.c	23
4.19 includes/wifi_manager.h File Reference	24
4.19.1 Macro Definition Documentation	24
4.19.1.1 EXAMPLE_ESP_MAXIMUM_RETRY	24
4.19.1.2 EXAMPLE_ESP_WIFI_PASS	24
4.19.1.3 EXAMPLE_ESP_WIFI_SSID	24
4.19.1.4 WIFI_CONNECTED_BIT	24

35

4.19.2 Function Documentation	25
4.19.2.1 setup_wifi()	25
4.20 wifi_manager.h	25
4.21 main/main.c File Reference	26
4.21.1 Detailed Description	27
4.21.2 Function Documentation	27
4.21.2.1 app_main()	27
4.21.2.2 create_tasks()	28
4.21.2.3 measure_temp_hum()	28
4.21.2.4 setup_timer()	29
4.21.2.5 task_send_data_mqtt()	29
4.21.2.6 task_show_data_oled()	29
4.21.2.7 task_wifi()	30
4.21.3 Variable Documentation	30
4.21.3.1 client	30
4.21.3.2 displayQueue	30
4.21.3.3 MQTT_CONNECTED	31
4.21.3.4 mqttQueue	31
4.21.3.5 s_wifi_event_group	31
4.21.3.6 TAG	31
4.21.3.7 timerDHT	31
4.22 main.c	32

Index

# **Chapter 1**

# **Data Structure Index**

# 1.1 Data Structures

Н	lere	are	the	data	structures	with	brief	descriptions
---	------	-----	-----	------	------------	------	-------	--------------

dht_data_t										
Structure to hold DHT sensor data with timestamp				 				 		5

2 Data Structure Index

# **Chapter 2**

# **File Index**

# 2.1 File List

Here is a list of all files with brief descriptions:

includes/common.c
includes/common.h
includes/dht_manager.c
includes/dht_manager.h
includes/display_manager.c
includes/display_manager.h
includes/mqtt_manager.c
includes/mqtt_manager.h
includes/wifi_manager.c
includes/wifi_manager.h
main/main.c
IoT Environmental Station Main Application

File Index

# **Chapter 3**

# **Data Structure Documentation**

# 3.1 dht\_data\_t Struct Reference

Structure to hold DHT sensor data with timestamp.

```
#include <dht_manager.h>
```

#### **Data Fields**

- · float temperature
- float humidity
- char timestamp [ISO8601\_STR\_LEN]

# 3.1.1 Detailed Description

Structure to hold DHT sensor data with timestamp.

This structure contains temperature and humidity readings from a DHT sensor along with an ISO 8601 formatted timestamp indicating when the reading was taken.

Definition at line 17 of file dht\_manager.h.

# 3.1.2 Field Documentation

# 3.1.2.1 humidity

```
float dht_data_t::humidity
```

Relative humidity reading in percentage (0-100%)

Definition at line 20 of file dht\_manager.h.

# 3.1.2.2 temperature

```
\verb|float| dht_data_t:: temperature|
```

Temperature reading in degrees Celsius

Definition at line 19 of file dht\_manager.h.

# 3.1.2.3 timestamp

```
char dht_data_t::timestamp[ISO8601_STR_LEN]
```

ISO 8601 format timestamp (YYYY-MM-DDTHH:MM:SSZ)

Definition at line 21 of file dht\_manager.h.

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

• includes/dht\_manager.h

# **Chapter 4**

# **File Documentation**

# 4.1 includes/common.c File Reference

```
#include "common.h"
```

#### **Functions**

- static void time\_sync\_notification\_cb (struct timeval \*tv)
- void setup\_sntp (void)

Initializes and configures the SNTP (Simple Network Time Protocol) client.

void get\_current\_date\_time (char \*date\_time)

Retrieves the current date and time in ISO 8601 format.

char \* create\_json\_payload (const dht\_data\_t \*data)

Creates a JSON payload string from DHT sensor data.

# 4.1.1 Function Documentation

# 4.1.1.1 create\_json\_payload()

Creates a JSON payload string from DHT sensor data.

This function takes DHT sensor data (temperature, humidity, timestamp) and creates a JSON formatted string suitable for transmission via MQTT or other communication protocols.

### **Parameters**

```
data Pointer to a dht_data_t structure containing sensor readings and timestamp information
```

#### Returns

Pointer to a dynamically allocated string containing the JSON payload. The caller is responsible for freeing this memory. Returns NULL on error.

Definition at line 45 of file common.c.

# 4.1.1.2 get\_current\_date\_time()

Retrieves the current date and time in ISO 8601 format.

This function gets the current system time and formats it as an ISO 8601 timestamp string (YYYY-MM-DDTHH ←: MM:SSZ format).

#### **Parameters**

date time Pointer to a character buffer where the formatted date/time string will be stored. Buffer must be at least ISO8601 S

Definition at line 36 of file common.c.

#### 4.1.1.3 setup\_sntp()

```
void setup_sntp (
     void )
```

Initializes and configures the SNTP (Simple Network Time Protocol) client.

This function sets up the SNTP client to synchronize the system time with internet time servers. It configures the timezone and starts the SNTP service.

Definition at line 9 of file common.c.

# 4.1.1.4 time\_sync\_notification\_cb()

```
void time_sync_notification_cb ( {\tt struct\ timeval\ *\ tv}) \quad [{\tt static}]
```

Definition at line 3 of file common.c.

# 4.2 common.c

#### Go to the documentation of this file.

```
00001 #include "common.h"
00002
00003 static void time_sync_notification_cb(struct timeval *tv)
00004 {
00005
           const char *TAG = "SNTP Notification";
          ESP_LOGI(TAG, "Notification of a time synchronization event");
00006
00007 }
80000
00009 void setup_sntp(void)
00010 {
          const char *TAG = "SNTP Initialization";
00011
          ESP_LOGI(TAG, "Initializing SNTP");
00012
00013
          esp_sntp_setoperatingmode(SNTP_OPMODE_POLL);
          esp_sntp_setservername(0, "pool.ntp.org");
sntp_set_time_sync_notification_cb(time_sync_notification_cb);
00014
00015
00016
          esp_sntp_init();
00017
00018
          // Set timezone to Spanish Peninsula Standard Time
00019
          setenv("TZ", "CET-1CEST, M3.5.0/2, M10.5.0/3", 1);
```

```
00020
           tzset();
00021
00022
           time_t now = 0;
00023
           struct tm timeinfo = {0};
00024
           int retry = 0;
00025
           const int retry_count = 10;
00027
            while (timeinfo.tm_year < (2020 - 1900) && ++retry < retry_count)</pre>
00028
                {\tt ESP\_LOGI\,(TAG,\ "Waiting\ for\ system\ time\ to\ be\ set...\ (%d)",\ retry);}
00029
                vTaskDelay(pdMS_TO_TICKS(2000));
00030
00031
                time(&now);
00032
                localtime_r(&now, &timeinfo);
00033
00034 }
00035
00036 void get_current_date_time(char *date_time)
00037 {
00038
           time_t now;
00039
           struct tm timeinfo;
00040
           time(&now);
00041
           localtime_r(&now, &timeinfo);
           strftime(date_time, ISO8601_STR_LEN, "%Y-%m-%dT%H:%M:%SZ", &timeinfo);
00042
00043 }
00044
00045 char *create_json_payload(const dht_data_t *data)
00046 {
           cJSON *root = cJSON_CreateObject();
char tempStr[7], humStr[7];
sprintf(tempStr, "%.2f", data->temperature);
sprintf(humStr, "%.2f", data->humidity);
00047
00048
00049
00050
00051
00052
           cJSON_AddStringToObject(root, "temperature", tempStr);
           cJSON_AddStringToObject(root, "humidity", humStr);
cJSON_AddStringToObject(root, "timestamp", data->timestamp);
00053
00054
00055
00056
           // Convert to string
00057
           char *json_str = cJSON_PrintUnformatted(root);
00058
           cJSON_Delete(root);
00059
00060
            return json_str;
00061 }
```

# 4.3 includes/common.h File Reference

```
#include "dht_manager.h"
#include "esp_sntp.h"
#include "cJSON.h"
#include "esp_log.h"
```

#### **Macros**

- #define MEASURE INTERVAL 60 \* 1000
- #define WIFI\_RECONNECT\_INTERVAL\_MS 60 \* 1000
- #define TIMER\_ID 1
- #define STACK\_SIZE 4 \* 1024
- #define MAX\_Q\_SIZE 10

#### **Functions**

void setup\_sntp (void)

Initializes and configures the SNTP (Simple Network Time Protocol) client.

void get current date time (char \*date time)

Retrieves the current date and time in ISO 8601 format.

char \* create\_json\_payload (const dht\_data\_t \*data)

Creates a JSON payload string from DHT sensor data.

#### 4.3.1 Macro Definition Documentation

# 4.3.1.1 MAX Q SIZE

```
#define MAX_Q_SIZE 10
```

Definition at line 13 of file common.h.

# 4.3.1.2 MEASURE\_INTERVAL

```
#define MEASURE_INTERVAL 60 * 1000
```

Definition at line 9 of file common.h.

# 4.3.1.3 STACK\_SIZE

```
#define STACK_SIZE 4 * 1024
```

Definition at line 12 of file common.h.

# 4.3.1.4 TIMER\_ID

```
#define TIMER_ID 1
```

Definition at line 11 of file common.h.

# 4.3.1.5 WIFI\_RECONNECT\_INTERVAL\_MS

```
#define WIFI_RECONNECT_INTERVAL_MS 60 * 1000
```

Definition at line 10 of file common.h.

#### 4.3.2 Function Documentation

# 4.3.2.1 create\_json\_payload()

Creates a JSON payload string from DHT sensor data.

This function takes DHT sensor data (temperature, humidity, timestamp) and creates a JSON formatted string suitable for transmission via MQTT or other communication protocols.

#### **Parameters**

data Pointer to a dht\_data\_t structure containing sensor readings and timestamp information

#### Returns

Pointer to a dynamically allocated string containing the JSON payload. The caller is responsible for freeing this memory. Returns NULL on error.

Definition at line 45 of file common.c.

4.4 common.h 11

# 4.3.2.2 get\_current\_date\_time()

Retrieves the current date and time in ISO 8601 format.

This function gets the current system time and formats it as an ISO 8601 timestamp string (YYYY-MM-DDTHH ∴ :MM:SSZ format).

#### **Parameters**

date\_time Pointer to a character buffer where the formatted date/time string will be stored. Buffer must be at least ISO8601\_S

Definition at line 36 of file common.c.

# 4.3.2.3 setup\_sntp()

```
void setup_sntp (
     void )
```

Initializes and configures the SNTP (Simple Network Time Protocol) client.

This function sets up the SNTP client to synchronize the system time with internet time servers. It configures the timezone and starts the SNTP service.

Definition at line 9 of file common.c.

# 4.4 common.h

Go to the documentation of this file.

```
00001 #ifndef COMMON_H
00002 #define COMMON_H
00003
00004 #include "dht_manager.h"
00005 #include "esp_sntp.h"
00006 #include "cJSON.h"
00007 #include "esp_log.h"
80000
00009 #define MEASURE_INTERVAL 60 * 1000
00010 #define WIFI_RECONNECT_INTERVAL_MS 60 * 1000
00011 #define TIMER_ID 1
00012 #define STACK_SIZE 4 * 1024
00013 #define MAX_Q_SIZE 10
00014
00022 void setup_sntp(void);
00035 void get_current_date_time(char *date_time);
00036
00050 char *create_json_payload(const dht_data_t *data);
00051
00052 #endif
```

# 4.5 includes/dht\_manager.c File Reference

```
#include "dht_manager.h"
```

#### **Functions**

esp\_err\_t setup\_dht (void)

Initializes the DHT sensor for temperature and humidity readings.

#### 4.5.1 Function Documentation

# 4.5.1.1 setup\_dht()

Initializes the DHT sensor for temperature and humidity readings.

This function configures the DHT sensor (DHT11, DHT22, AM2301, or SI7021) based on the compile-time configuration. It sets up the GPIO pin and prepares the sensor for data acquisition.

Returns

ESP\_OK on successful initialization, ESP\_FAIL or other ESP error codes on failure

Definition at line 4 of file dht\_manager.c.

# 4.6 dht\_manager.c

Go to the documentation of this file.

```
00001
00002 #include "dht_manager.h"
00003
00004 esp_err_t setup_dht(void)
00005 {
00006
           esp_err_t res;
          // Enable internal pull-up resistor if specified in menuconfig
if (CONFIG_EXAMPLE_INTERNAL_PULLUP)
00007
80000
00009
00010
               res = gpio_pullup_en(CONFIG_ESP_TEMP_SENSOR_GPIO);
00011
00012
          else
00013
          {
00014
               res = gpio_pullup_dis(CONFIG_ESP_TEMP_SENSOR_GPIO);
00015
00016
           return res;
00017 }
```

# 4.7 includes/dht\_manager.h File Reference

```
#include "dht.h"
```

# **Data Structures**

struct dht\_data\_t

Structure to hold DHT sensor data with timestamp.

#### **Macros**

- #define CONFIG\_EXAMPLE\_INTERNAL\_PULLUP 0
- #define CONFIG\_EXAMPLE\_TYPE\_DHT11 0
- #define ISO8601 STR LEN 25
- #define DHT\_SENSOR\_TYPE (CONFIG\_EXAMPLE\_TYPE\_AM2301 ? DHT\_TYPE\_AM2301 ↔ : CONFIG\_EXAMPLE\_TYPE\_DHT11 ? DHT\_TYPE\_DHT11 : DHT\_TYPE\_SI7021)

#### **Functions**

esp\_err\_t setup\_dht (void)
 Initializes the DHT sensor for temperature and humidity readings.

# 4.7.1 Macro Definition Documentation

# 4.7.1.1 CONFIG\_EXAMPLE\_INTERNAL\_PULLUP

```
#define CONFIG_EXAMPLE_INTERNAL_PULLUP 0
```

Definition at line 6 of file dht\_manager.h.

# 4.7.1.2 CONFIG\_EXAMPLE\_TYPE\_DHT11

```
#define CONFIG_EXAMPLE_TYPE_DHT11 0
```

Definition at line 7 of file dht\_manager.h.

# 4.7.1.3 DHT\_SENSOR\_TYPE

```
#define DHT_SENSOR_TYPE (CONFIG_EXAMPLE_TYPE_AM2301 ? DHT_TYPE_AM2301 : CONFIG_EXAMPLE_TYPE_DHT11
? DHT_TYPE_DHT11 : DHT_TYPE_SI7021)
```

Definition at line 9 of file dht\_manager.h.

# 4.7.1.4 ISO8601\_STR\_LEN

```
#define ISO8601_STR_LEN 25
```

Definition at line 8 of file dht manager.h.

#### 4.7.2 Function Documentation

#### 4.7.2.1 setup dht()

Initializes the DHT sensor for temperature and humidity readings.

This function configures the DHT sensor (DHT11, DHT22, AM2301, or SI7021) based on the compile-time configuration. It sets up the GPIO pin and prepares the sensor for data acquisition.

#### Returns

ESP\_OK on successful initialization, ESP\_FAIL or other ESP error codes on failure

Definition at line 4 of file dht\_manager.c.

# 4.8 dht manager.h

Go to the documentation of this file.

```
00001 #ifndef DHT_SENSOR_H
00002 #define DHT_SENSOR_H
00003
00004 #include "dht.h"
00005
00006 #define CONFIG_EXAMPLE_INTERNAL_PULLUP 0
00007 #define CONFIG_EXAMPLE_TYPE_DHT11
00008 #define ISO8601_STR_LEN 25 // "YYYY-MM-DDTHH:MM:SSZ" + null
00009 #define DHT_SENSOR_TYPE (CONFIG_EXAMPLE_TYPE_AM2301 ? DHT_TYPE_AM2301 : CONFIG_EXAMPLE_TYPE_DHT11 ?
     DHT_TYPE_DHT11 : DHT_TYPE_S17021)
00010
00017 typedef struct
00018 {
00019
          float temperature;
00020
         float humidity;
         char timestamp[ISO8601_STR_LEN];
00021
00022 } dht_data_t;
00023
00034 esp_err_t setup_dht(void);
00035
00036 #endif
```

# 4.9 includes/display\_manager.c File Reference

```
#include "display_manager.h"
```

#### **Functions**

- u8g2\_t init\_oled\_display (void)
   Initializes the OLED display using the u8g2 graphics library.
- void show\_dht\_data (u8g2\_t u8g2, const char \*date, const char \*time, const char \*temp, const char \*hum)
   Displays DHT sensor data on the OLED screen.

# 4.9.1 Function Documentation

#### 4.9.1.1 init oled display()

Initializes the OLED display using the u8g2 graphics library.

This function sets up the I2C communication interface and initializes the OLED display using the u8g2 library. It configures the display for subsequent drawing operations.

#### Returns

u8g2\_t structure containing the display configuration and state. This structure is used for all subsequent display operations.

Definition at line 3 of file display\_manager.c.

# 4.9.1.2 show\_dht\_data()

Displays DHT sensor data on the OLED screen.

This function renders the date, time, temperature, and humidity values on the OLED display using the u8g2 graphics library. It formats and positions the text appropriately on the screen.

#### **Parameters**

u8g2	The u8g2 display structure initialized by init_oled_display()
date	Pointer to a null-terminated string containing the date information
time	Pointer to a null-terminated string containing the time information
temp	Pointer to a null-terminated string containing the temperature reading
hum	Pointer to a null-terminated string containing the humidity reading

Definition at line 34 of file display\_manager.c.

# 4.10 display manager.c

#### Go to the documentation of this file.

```
00001 #include "display_manager.h"
00002
00003 u8g2_t init_oled_display(void)
00004 {
00005
           static const char *TAG = "oled_display_module";
            /* OLED Display Setup*/
00006
00007
           u8g2_t u8g2;
80000
           u8g2_esp32_hal_t u8g2_esp32_hal = U8G2_ESP32_HAL_DEFAULT;
           u8g2_esp32_hal.bus.i2c.scl = I2C_SCL;
u8g2_esp32_hal.bus.i2c.sda = I2C_SDA;
00009
00010
00011
           u8g2_esp32_hal_init(u8g2_esp32_hal);
00012
00013
           u8g2_Setup_ssd1306_i2c_128x64_noname_f(&u8g2, U8G2_R0,
00014
                                                         // u8x8_byte_sw_i2c,
00015
                                                         u8g2_esp32_i2c_byte_cb,
00016
                                                         u8g2_esp32_gpio_and_delay_cb);
00017
           u8x8_SetI2CAddress(&u8q2.u8x8, 0x78);
00018
           ESP_LOGI(TAG, "Init Display");
00019
           u8g2_InitDisplay(&u8g2); // send init sequence to the display, display is in
00020
                                         // sleep mode after this,
00021
           ESP_LOGI(TAG, "WakeUp Display");
00022
           u8g2_SetPowerSave(&u8g2, 0); // wake up display ESP_LOGI(TAG, "Clear Buffer");
00023
00024
00025
00026
           \verb"u8g2_SetFont(\&u8g2, u8g2_font\_unifont_t\_symbols)";
00027
00028
           u8g2 ClearBuffer(&u8g2);
00029
           /* End of setup*/
00030
00031
           return u8g2;
00032 }
00033
00034 void show_dht_data(u8g2_t u8g2, const char *date, const char *time, const char *temp, const char *tum)
00035 {
00036
           u8g2 ClearBuffer(&u8g2);
           u8g2_DrawStr(&u8g2, 0, 12, date);
00038
           u8g2_DrawStr(&u8g2, 0, 28, time);
00039
           u8g2_DrawStr(&u8g2, 0, 46, temp);
           u8g2_DrawGlyph(&u8g2, 110, 46, 0x2600);

u8g2_DrawStr(&u8g2, 0, 62, hum);

u8g2_DrawGlyph(&u8g2, 110, 62, 0x2614);

u8g2_SendBuffer(&u8g2); // Send the buffer data to display
00040
00041
00042
00043
00044 }
```

# 4.11 includes/display manager.h File Reference

```
#include "driver/i2c_master.h"
#include "u8g2.h"
#include "u8g2_esp32_hal.h"
#include "esp_log.h"
```

# Macros

- #define I2C SDA 8
- #define I2C\_SCL 9

#### **Functions**

u8g2\_t init\_oled\_display (void)

Initializes the OLED display using the u8g2 graphics library.

• void show\_dht\_data (u8g2\_t u8g2, const char \*date, const char \*time, const char \*temp, const char \*hum)

Displays DHT sensor data on the OLED screen.

# 4.11.1 Macro Definition Documentation

# 4.11.1.1 I2C\_SCL

```
#define I2C_SCL 9
```

I2C SCL (Serial Clock) pin number

Definition at line 10 of file display\_manager.h.

#### 4.11.1.2 I2C SDA

```
#define I2C_SDA 8
```

I2C SDA (Serial Data) pin number

Definition at line 9 of file display\_manager.h.

# 4.11.2 Function Documentation

#### 4.11.2.1 init oled display()

Initializes the OLED display using the u8g2 graphics library.

This function sets up the I2C communication interface and initializes the OLED display using the u8g2 library. It configures the display for subsequent drawing operations.

#### Returns

u8g2\_t structure containing the display configuration and state. This structure is used for all subsequent display operations.

Definition at line 3 of file display manager.c.

#### 4.11.2.2 show\_dht\_data()

Displays DHT sensor data on the OLED screen.

This function renders the date, time, temperature, and humidity values on the OLED display using the u8g2 graphics library. It formats and positions the text appropriately on the screen.

#### **Parameters**

u8g2	The u8g2 display structure initialized by init_oled_display()
date	Pointer to a null-terminated string containing the date information
time	Pointer to a null-terminated string containing the time information
temp	Pointer to a null-terminated string containing the temperature reading
hum	Pointer to a null-terminated string containing the humidity reading

Definition at line 34 of file display\_manager.c.

# 4.12 display\_manager.h

#### Go to the documentation of this file.

```
00001 #ifndef DISPLAY_H
00002 #define DISPLAY_H
00003
00004 #include "driver/i2c_master.h"
00005 #include "u8g2.h"
00006 #include "u8g2_esp32_hal.h"
00007 #include "esp_log.h"
00008
00009 #define I2C_SDA 8
00010 #define I2C_SCL 9
00011
00023 u8g2_t init_oled_display(void);
00024
00039 void show_dht_data(u8g2_t u8g2, const char *date, const char *time, const char *temp, const char *hum);
00040
00041 #endif
```

# 4.13 includes/mqtt\_manager.c File Reference

```
#include "mqtt_manager.h"
```

#### **Functions**

- static void mqtt\_event\_handler (void \*handler\_args, esp\_event\_base\_t base, int32\_t event\_id, void \*event←
   \_data)
- void setup\_mqtt (void)

Initializes and configures the MQTT client for IoT communication.

#### **Variables**

- bool MQTT\_CONNECTED = false
- esp\_mqtt\_client\_handle\_t client = NULL

# 4.13.1 Function Documentation

# 4.13.1.1 mqtt\_event\_handler()

Definition at line 7 of file mqtt manager.c.

# 4.13.1.2 setup\_mqtt()

Initializes and configures the MQTT client for IoT communication.

This function sets up the MQTT client with the necessary configuration including broker connection details and event handlers. It establishes the connection to the MQTT broker and prepares the client for publishing sensor data and receiving commands. The configuration parameters are typically read from the ESP-IDF configuration system (menuconfig).

Definition at line 49 of file mqtt\_manager.c.

# 4.13.2 Variable Documentation

# 4.13.2.1 client

```
esp_mqtt_client_handle_t client = NULL
```

MQTT client handle

Definition at line 5 of file mqtt\_manager.c.

# 4.13.2.2 MQTT\_CONNECTED

```
bool MQTT_CONNECTED = false
```

MQTT connection status flag

Definition at line 4 of file mqtt\_manager.c.

# 4.14 mqtt manager.c

#### Go to the documentation of this file.

```
00001 #include "mqtt_manager.h"
00002
00003
00004 bool MOTT CONNECTED = false:
00005 esp_mqtt_client_handle_t client = NULL;
00006
00007 static void mqtt_event_handler(void *handler_args, esp_event_base_t base, int32_t event_id, void
      *event_data)
} 80000
           const char *TAG = "MQTT_EVENT_HANDLER";
00009
           esp_mqtt_event_handle_t event = event_data;
00010
           esp_mqtt_client_handle_t client = event->client;
00011
00012
           int msg_id;
00013
           switch ((esp_mqtt_event_id_t)event_id)
00014
           case MQTT_EVENT_CONNECTED:
00015
               ESP_LOGI(TAG, "MQTT_EVENT_CONNECTED");
MQTT_CONNECTED = true;
00016
00017
00018
00019
               msg_id = esp_mqtt_client_subscribe(client, "/home/office/dht", 0);
00020
               ESP_LOGI(TAG, "sent subscribe successful, msg_id=%d", msg_id);
00021
          case MOTT_EVENT_DISCONNECTED:
    ESP_LOGW(TAG, "MQTT_EVENT_DISCONNECTED");
    MQTT_CONNECTED = false;
00022
00023
00024
00025
               break;
00026
00027
          case MQTT_EVENT_SUBSCRIBED:
00028
             ESP_LOGI(TAG, "MQTT_EVENT_SUBSCRIBED, msg_id=%d", event->msg_id);
00029
               break:
00030
          case MQTT_EVENT_UNSUBSCRIBED:
00031
             ESP_LOGW(TAG, "MQTT_EVENT_UNSUBSCRIBED, msg_id=%d", event->msg_id);
00032
          case MQTT_EVENT_PUBLISHED:
    ESP_LOGI(TAG, "MQTT_EVENT_PUBLISHED, msg_id=%d", event->msg_id);
00033
00034
00035
               break;
          case MQTT_EVENT_DATA:
00036
              ESP_LOGI(TAG, "TOPIC=%.*s", event->topic_len, event->topic);
ESP_LOGI(TAG, "DATA=%.*s", event->data_len, event->data);
00038
00039
00040
          case MQTT_EVENT_ERROR:
             ESP_LOGI(TAG, "MQTT_EVENT_ERROR");
00041
00042
               break;
00043
          default:
00044
              ESP_LOGI(TAG, "Other event id:%d", event->event_id);
00045
00046
           }
00047 }
00048
00049 void setup_mqtt(void)
00050 {
           const char *TAG = "Setup MQTT";
00051
00052
           ESP_LOGI(TAG, "STARTING MQTT");
          esp_mqtt_client_config_t mqttConfig = {
   .broker.address.uri = CONFIG_BROKER_URI,
00053
00054
00055
00056
00057
          client = esp_mqtt_client_init(&mqttConfig);
00058
           esp_mqtt_client_register_event(client, ESP_EVENT_ANY_ID, mqtt_event_handler, client);
00059
           esp_mqtt_client_start(client);
00060 }
```

# 4.15 includes/mqtt\_manager.h File Reference

```
#include "mqtt_client.h"
#include "esp_log.h"
```

# **Functions**

void setup\_mqtt (void)

Initializes and configures the MQTT client for IoT communication.

4.16 mqtt\_manager.h

# 4.15.1 Function Documentation

#### 4.15.1.1 setup\_mqtt()

```
void setup_mqtt (
     void )
```

Initializes and configures the MQTT client for IoT communication.

This function sets up the MQTT client with the necessary configuration including broker connection details and event handlers. It establishes the connection to the MQTT broker and prepares the client for publishing sensor data and receiving commands. The configuration parameters are typically read from the ESP-IDF configuration system (menuconfig).

Definition at line 49 of file mqtt\_manager.c.

# 4.16 mqtt\_manager.h

#### Go to the documentation of this file.

```
00001 #ifndef MQTT_MANAGER_H
00002 #define MQTT_MANAGER_H
00003
00004 #include "mqtt_client.h"
00005 #include "esp_log.h"
00006
00017 void setup_mqtt(void);
00018
00019 #endif
```

# 4.17 includes/wifi\_manager.c File Reference

```
#include "wifi_manager.h"
```

#### **Functions**

- static void wifi\_event\_handler (void \*arg, esp\_event\_base\_t event\_base, int32\_t event\_id, void \*event\_data)
- void setup\_wifi (void)

Initializes and configures WiFi connectivity for the ESP32.

#### **Variables**

EventGroupHandle\_t s\_wifi\_event\_group

# 4.17.1 Function Documentation

#### 4.17.1.1 setup\_wifi()

```
void setup_wifi (
     void )
```

Initializes and configures WiFi connectivity for the ESP32.

This function sets up the WiFi subsystem including:

- · WiFi driver initialization
- · Event loop configuration
- · Access Point connection with credentials from configuration
- · Retry mechanism for failed connections
- · Event handling for connection status

The function uses configuration parameters (SSID, password, security mode) defined through the ESP-IDF configuration system. It will attempt to connect to the configured access point and handle reconnection attempts according to the maximum retry settings.

Definition at line 28 of file wifi\_manager.c.

# 4.17.1.2 wifi\_event\_handler()

Definition at line 6 of file wifi\_manager.c.

#### 4.17.2 Variable Documentation

# 4.17.2.1 s\_wifi\_event\_group

```
EventGroupHandle_t s_wifi_event_group
```

WiFi event group handle

Definition at line 4 of file wifi\_manager.c.

4.18 wifi\_manager.c 23

# 4.18 wifi manager.c

#### Go to the documentation of this file.

```
00001 #include "wifi_manager.h"
00002
00003 \slash \star FreeRTOS event group to signal when we are connected \slash \star
00004 EventGroupHandle_t s_wifi_event_group;
00005
00006 static void wifi_event_handler(void *arg, esp_event_base_t event_base,
00007
                                        int32_t event_id, void *event_data)
00008 {
           const char *TAG = "Wifi Event Handler";
00009
           if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START)
00010
00011
               esp wifi connect();
00012
00013
           else if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_DISCONNECTED)
00014
           {
00015
               ESP_LOGI(TAG, "Disconnected from AP, will try to reconnect in 60 seconds");
               xEventGroupClearBits(s_wifi_event_group, WIFI_CONNECTED_BIT);
00016
00017
00018
           else if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP)
00019
               ip_event_got_ip_t *event = (ip_event_got_ip_t *)event_data;
ESP_LOGI(TAG, "got ip:" IPSTR, IP2STR(&event->ip_info.ip));
00020
00021
               // initialize_sntp();
// mqtt_app_start();
00022
00023
00024
               xEventGroupSetBits(s_wifi_event_group, WIFI_CONNECTED_BIT);
00025
           }
00026 }
00027
00028 void setup_wifi(void)
00029 {
           const char *TAG = "Setup Wifi";
00030
00031
           ESP_LOGI(TAG, "Setting up WiFi");
00032
           s_wifi_event_group = xEventGroupCreate();
00033
00034
           ESP_ERROR_CHECK(esp_netif_init());
00035
00036
           ESP_ERROR_CHECK(esp_event_loop_create_default());
esp_netif_create_default_wifi_sta();
00037
00038
00039
           wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
00040
           ESP_ERROR_CHECK(esp_wifi_init(&cfg));
00041
00042
           esp_event_handler_instance_t instance_any_id;
00043
           esp_event_handler_instance_t instance_got_ip;
           ESP_ERROR_CHECK(esp_event_handler_instance_register(WIFI_EVENT,
00044
00045
                                                                    ESP_EVENT_ANY_ID,
00046
                                                                    &wifi_event_handler,
00047
                                                                    NULL,
00048
                                                                    &instance_any_id));
00049
           ESP ERROR CHECK(esp event handler instance register(IP EVENT,
00050
                                                                    IP_EVENT_STA_GOT_IP,
                                                                    &wifi_event_handler,
00051
00052
                                                                    NULL,
00053
                                                                    &instance_got_ip));
00054
00055
           wifi_config_t wifi_config = {
00056
               .sta = {
00057
                   .ssid = EXAMPLE_ESP_WIFI_SSID,
00058
                    .password = EXAMPLE_ESP_WIFI_PASS,
00059
                    /\star Authmode threshold resets to WPA2 as default if password matches WPA2 standards
       (password len \Rightarrow 8).
                    \star If you want to connect the device to deprecated WEP/WPA networks, Please set the
00060
      threshold value
00061
                    * to WIFI_AUTH_WEP/WIFI_AUTH_WPA_PSK and set the password with length and format matching
00062
                    * WIFI_AUTH_WEP/WIFI_AUTH_WPA_PSK standards.
00063
                   .threshold.authmode = ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD,
00064
00065
                    .sae_pwe_h2e = ESP_WIFI_SAE_MODE,
00066
                    .sae_h2e_identifier = EXAMPLE_H2E_IDENTIFIER,
00067
               },
00068
00069
           {\tt ESP\_ERROR\_CHECK\,(esp\_wifi\_set\_mode\,(WIFI\_MODE\_STA)\,)\,;}
00070
           ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_STA, &wifi_config));
00071
           ESP_ERROR_CHECK(esp_wifi_start());
00072
00073
           ESP_LOGI(TAG, "wifi_task started and wifi driver started");
00074 }
```

# 4.19 includes/wifi manager.h File Reference

```
#include "esp_log.h"
#include "esp_wifi.h"
#include "freertos/event_groups.h"
#include "freertos/FreeRTOS.h"
```

#### **Macros**

- #define EXAMPLE\_ESP\_WIFI\_SSID CONFIG\_ESP\_WIFI\_SSID
- #define EXAMPLE\_ESP\_WIFI\_PASS CONFIG\_ESP\_WIFI\_PASSWORD
- #define EXAMPLE\_ESP\_MAXIMUM\_RETRY CONFIG\_ESP\_MAXIMUM\_RETRY
- #define WIFI CONNECTED BIT BIT0

#### **Functions**

void setup\_wifi (void)

Initializes and configures WiFi connectivity for the ESP32.

#### 4.19.1 Macro Definition Documentation

#### 4.19.1.1 EXAMPLE ESP MAXIMUM RETRY

```
#define EXAMPLE_ESP_MAXIMUM_RETRY CONFIG_ESP_MAXIMUM_RETRY
```

Definition at line 11 of file wifi\_manager.h.

# 4.19.1.2 EXAMPLE\_ESP\_WIFI\_PASS

```
#define EXAMPLE_ESP_WIFI_PASS CONFIG_ESP_WIFI_PASSWORD
```

Definition at line 10 of file wifi\_manager.h.

# 4.19.1.3 EXAMPLE ESP WIFI SSID

```
#define EXAMPLE_ESP_WIFI_SSID CONFIG_ESP_WIFI_SSID
```

Definition at line 9 of file wifi\_manager.h.

# 4.19.1.4 WIFI\_CONNECTED\_BIT

```
#define WIFI_CONNECTED_BIT BIT0
```

Event bit indicating successful WiFi connection

Definition at line 44 of file wifi\_manager.h.

4.20 wifi\_manager.h

#### 4.19.2 Function Documentation

# 4.19.2.1 setup\_wifi()

```
void setup_wifi (
     void )
```

Initializes and configures WiFi connectivity for the ESP32.

This function sets up the WiFi subsystem including:

- · WiFi driver initialization
- · Event loop configuration
- · Access Point connection with credentials from configuration
- · Retry mechanism for failed connections
- · Event handling for connection status

The function uses configuration parameters (SSID, password, security mode) defined through the ESP-IDF configuration system. It will attempt to connect to the configured access point and handle reconnection attempts according to the maximum retry settings.

Definition at line 28 of file wifi\_manager.c.

# 4.20 wifi\_manager.h

#### Go to the documentation of this file.

```
00001 #ifndef WIFI MANAGER H
00002 #define WIFI_MANAGER_H
00003
00004 #include "esp log.h'
00005 #include "esp_wifi.h"
00006 #include "freertos/event_groups.h"
00007 #include "freertos/FreeRTOS.h"
80000
00009 #define EXAMPLE_ESP_WIFI_SSID CONFIG_ESP_WIFI_SSID 00010 #define EXAMPLE_ESP_WIFI_PASS CONFIG_ESP_WIFI_PASSWORD
00011 #define EXAMPLE_ESP_MAXIMUM_RETRY CONFIG_ESP_MAXIMUM_RETRY
00012
00013 #if CONFIG_ESP_WPA3_SAE_PWE_HUNT_AND_PECK
00014 #define ESP_WIFI_SAE_MODE WPA3_SAE_PWE_HUNT_AND_PECK
00015 #define EXAMPLE_H2E_IDENTIFIER ""
00016 #elif CONFIG_ESP_WPA3_SAE_PWE_HASH_TO_ELEMENT
00017 #define ESP_WIFI_SAE_MODE WPA3_SAE_PWE_HASH_TO_ELEMENT
00018 #define EXAMPLE_H2E_IDENTIFIER CONFIG_ESP_WIFI_PW_ID
00019 #elif CONFIG_ESP_WPA3_SAE_PWE_BOTH
00020 #define ESP_WIFI_SAE_MODE WPA3_SAE_PWE_BOTH
00021 #define EXAMPLE_H2E_IDENTIFIER CONFIG_ESP_WIFI_PW_ID
00022 #endif
00023 #if CONFIG_ESP_WIFI_AUTH_OPEN
00024 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_OPEN
00025 #elif CONFIG_ESP_WIFI_AUTH_WEP
00026 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WEP
00027 #elif CONFIG_ESP_WIFI_AUTH_WPA_PSK
00028 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WPA_PSK
00029 #edificONFIG_ESP_WIFI_AUTH_WPA2_PSK
00030 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WPA2_PSK
00031 #elif CONFIG_ESP_WIFI_AUTH_WPA_WPA2_PSK
00032 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WPA_WPA2_PSK
00033 #elif CONFIG_ESP_WIFI_AUTH_WPA3_PSK
00034 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WPA3_PSK 00035 #elif CONFIG_ESP_WIFI_AUTH_WPA2_WPA3_PSK
00036 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WPA2_WPA3_PSK
00037 #elif CONFIG_ESP_WIFI_AUTH_WAPI_PSK
```

```
00038 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_WAPI_PSK
00039 #endif
00040
00041 /* The event group allows multiple bits for each event, but we only care about two events:
00042 * - we are connected to the AP with an IP
00043 * - we failed to connect after the maximum amount of retries */
00044 #define WIFI_CONNECTED_BIT BITO
00045
00062 void setup_wifi(void);
00063
00064 #endif
```

# 4.21 main/main.c File Reference

IoT Environmental Station Main Application.

```
#include <stdio.h>
#include <string.h>
#include "dht_manager.h"
#include "display_manager.h"
#include "wifi_manager.h"
#include "mqtt_manager.h"
#include "common.h"
#include "freertos/task.h"
#include "freertos/timers.h"
#include "sp_system.h"
#include "esp_event.h"
#include "nvs_flash.h"
```

#### **Functions**

• esp\_err\_t setup\_timer (void)

Creates and starts the DHT sensor measurement timer.

esp\_err\_t create\_tasks (void)

Creates and starts all FreeRTOS tasks for the application.

void measure\_temp\_hum (TimerHandle\_t timer)

Timer callback function for reading DHT sensor data.

void task\_show\_data\_oled (void \*args)

FreeRTOS task for displaying sensor data on OLED screen.

void task\_send\_data\_mqtt (void \*args)

FreeRTOS task for transmitting sensor data via MQTT.

void task\_wifi (void \*args)

FreeRTOS task for WiFi connection management.

void app\_main (void)

Main application entry point.

#### **Variables**

- EventGroupHandle\_t s\_wifi\_event\_group
- bool MQTT\_CONNECTED
- esp\_mqtt\_client\_handle\_t client
- TimerHandle t timerDHT
- QueueHandle t displayQueue
- QueueHandle\_t mqttQueue
- static const char \* TAG = "iot\_env\_station"

# 4.21.1 Detailed Description

IoT Environmental Station Main Application.

This file contains the main application logic for an IoT environmental monitoring station that reads temperature and humidity data from a DHT sensor, displays the information on an OLED screen, and transmits the data via MQTT over WiFi.

The application uses FreeRTOS tasks for concurrent operation:

- · Timer-based sensor reading
- · OLED display updates
- · MQTT data transmission
- · WiFi connection management

Definition in file main.c.

# 4.21.2 Function Documentation

#### 4.21.2.1 app\_main()

```
void app_main (
     void )
```

Main application entry point.

This function initializes the IoT environmental station by:

- Setting up NVS (Non-Volatile Storage) for configuration data
- · Creating FreeRTOS queues for inter-task communication
- · Initializing the DHT sensor
- · Setting up the measurement timer
- · Creating and starting all application tasks

The function handles NVS initialization errors by erasing and reinitializing the flash if necessary.

Definition at line 67 of file main.c.

#### 4.21.2.2 create\_tasks()

Creates and starts all FreeRTOS tasks for the application.

This function creates three main tasks:

- 1. Display task (priority 1): Updates OLED display with sensor data
- 2. WiFi task (priority 3): Manages WiFi connection and reconnection
- 3. MQTT task (priority 2): Handles MQTT communication and data transmission

Each task is created with a stack size defined by STACK\_SIZE and assigned appropriate priorities for proper system operation.

Returns

ESP\_OK if all tasks are created successfully, ESP\_FAIL if any task creation fails

Definition at line 133 of file main.c.

# 4.21.2.3 measure\_temp\_hum()

Timer callback function for reading DHT sensor data.

This function is called periodically by the FreeRTOS timer to:

- Read temperature and humidity data from the DHT sensor
- · Capture the current timestamp in ISO8601 format
- · Package the data into a dht\_data\_t structure
- · Send the data to both display and MQTT queues for processing

The function handles read errors by logging appropriate error messages and only sends data to queues when the sensor reading is successful. Queue send operations use a timeout to prevent blocking if queues are full.

#### **Parameters**

timer	Handle to the timer that triggered this callback (unused)

Definition at line 318 of file main.c.

# 4.21.2.4 setup\_timer()

Creates and starts the DHT sensor measurement timer.

This function creates a FreeRTOS software timer that triggers periodic temperature and humidity measurements from the DHT sensor. The timer is configured to repeat at intervals defined by MEASURE\_INTERVAL and calls the measure\_temp\_hum() callback function.

Returns

ESP\_OK on successful timer creation and start, ESP\_FAIL on error

Definition at line 96 of file main.c.

#### 4.21.2.5 task\_send\_data\_mqtt()

FreeRTOS task for transmitting sensor data via MQTT.

This task handles MQTT communication by:

- · Waiting for WiFi connection establishment
- · Initializing MQTT client and SNTP time synchronization
- · Continuously monitoring the MQTT queue for sensor data
- Converting sensor data to JSON format
- · Publishing data to the configured MQTT topic

The task only attempts to send data when both the queue has data and the MQTT client is connected to the broker.

#### **Parameters**

args Pointer to task parameters (unused in this implementation)

Definition at line 234 of file main.c.

# 4.21.2.6 task\_show\_data\_oled()

FreeRTOS task for displaying sensor data on OLED screen.

This task continuously monitors the display queue for new sensor data and updates the OLED display with formatted temperature, humidity, date, and time information. The task:

- · Waits for data from the display queue
- · Parses the ISO8601 timestamp from sensor data
- · Formats date and time strings for display
- Updates the OLED screen with current readings

#### **Parameters**

args Pointer to task parameters (unused in this implementation)

Definition at line 187 of file main.c.

# 4.21.2.7 task\_wifi()

```
void task_wifi (
     void * args)
```

FreeRTOS task for WiFi connection management.

This task manages the WiFi connection by:

- · Initializing the WiFi subsystem and connecting to the configured network
- · Continuously monitoring the connection status using event groups
- · Automatically attempting reconnection if the connection is lost
- · Implementing a watchdog mechanism with configurable reconnect intervals

The task uses the WIFI\_RECONNECT\_INTERVAL\_MS timeout to check connection status and triggers reconnection attempts when necessary.

#### **Parameters**

args Pointer to task parameters (unused in this implementation)

Definition at line 282 of file main.c.

# 4.21.3 Variable Documentation

#### 4.21.3.1 client

```
esp_mqtt_client_handle_t client [extern]
```

MQTT client handle

Definition at line 5 of file mqtt\_manager.c.

#### 4.21.3.2 displayQueue

QueueHandle\_t displayQueue

Queue for sensor data to be displayed on OLED

Definition at line 39 of file main.c.

# 4.21.3.3 MQTT\_CONNECTED

bool MQTT\_CONNECTED [extern]

MQTT connection status flag

Definition at line 4 of file mqtt\_manager.c.

# 4.21.3.4 mqttQueue

QueueHandle\_t mqttQueue

Queue for sensor data to be sent via MQTT

Definition at line 40 of file main.c.

# 4.21.3.5 s\_wifi\_event\_group

EventGroupHandle\_t s\_wifi\_event\_group [extern]

WiFi event group handle

Definition at line 4 of file wifi\_manager.c.

# 4.21.3.6 TAG

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

Log tag for this module

Definition at line 42 of file main.c.

# 4.21.3.7 timerDHT

TimerHandle\_t timerDHT

Timer handle for periodic DHT sensor readings

Definition at line 38 of file main.c.

# 4.22 main.c

#### Go to the documentation of this file.

```
00001
00015
00016 #include <stdio.h>
00017 #include <string.h>
00018 #include <time.h>
00019
00020 #include "dht_manager.h"
00021 #include "display_manager.h"
00022 #include "wifi_manager.h"
00023 #include "mqtt_manager.h"
00024 #include "common.h"
00025
00026 #include "freertos/task.h"
00027 #include "freertos/timers.h"
00028 #include "freertos/queue.h"
00029
00030 #include "esp_system.h"
00031 #include "esp_event.h"
00032 #include "nvs_flash.h"
00033
00034 extern EventGroupHandle_t s_wifi_event_group;
00035 extern bool MOTT_CONNECTED;
00036 extern esp_mqtt_client_handle_t client;
00037
00038 TimerHandle_t timerDHT;
00039 QueueHandle_t displayQueue;
00040 QueueHandle_t mqttQueue;
00041
00042 static const char *TAG = "iot env station";
00043
00044 /* Function prototypes */
00045 esp_err_t setup_timer(void);
00046 esp_err_t create_tasks(void);
00047
00048 void measure_temp_hum(TimerHandle_t timer);
00049 void task_show_data_oled(void *args);
00050 void task_send_data_mqtt(void *args);
00051 void task_wifi(void *args);
00052
00067 void app_main(void)
00068 {
00069
            // Initialize NVS
            esp_err_t ret = nvs_flash_init();
00070
00071
            if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND)
00072
00073
                 ESP_ERROR_CHECK(nvs_flash_erase());
00074
                ret = nvs_flash_init();
00075
00076
           ESP_ERROR_CHECK(ret);
00077
00078
            displayQueue = xQueueCreate(MAX_Q_SIZE, sizeof(dht_data_t));
00079
            mqttQueue = xQueueCreate(MAX_Q_SIZE, sizeof(dht_data_t));
            ESP_ERROR_CHECK(setup_dht());
08000
           ESP_ERROR_CHECK(setup_timer());
ESP_ERROR_CHECK(create_tasks());
00081
00082
00083 }
00084
00096 esp_err_t setup_timer(void)
00097 {
00098
            timerDHT = xTimerCreate("Timer DHT",
                                         pdMS_TO_TICKS (MEASURE_INTERVAL),
00099
00100
                                         pdTRUE,
                                         (void *)TIMER_ID,
00101
00102
                                         measure_temp_hum);
00103
            if (timerDHT == NULL)
00104
                 ESP_LOGE(TAG, "Timer could not be created");
00105
00106
                return ESP_FAIL;
00107
           }
00108
           else
00109
           {
                 if (xTimerStart(timerDHT, 0) != pdPASS)
00110
00111
                {
                     ESP_LOGE(TAG, "The timer could not be set into the Active state");
00112
00113
                     return ESP_FAIL;
00114
                }
00115
00116
            return ESP_OK;
00117 }
00118
00133 esp_err_t create_tasks(void)
00134 {
```

4.22 main.c 33

```
static uint8_t ucParameterToPass;
00136
           TaskHandle_t displayHandle = NULL;
00137
          TaskHandle_t mqttHandle = NULL;
00138
          TaskHandle_t wifiHandle = NULL;
00139
00140
          if (xTaskCreate(task show data oled,
                            "Show read data on oled display",
00141
00142
                            STACK_SIZE,
00143
                            &ucParameterToPass,
00144
                            &displayHandle) != pdPASS)
00145
00146
          {
00147
              return ESP_FAIL;
00148
00149
          00150
00151
                            STACK_SIZE,
00152
00153
                            &ucParameterToPass,
00154
00155
                            &wifiHandle) != pdPASS)
00156
00157
               return ESP FAIL;
00158
          }
00159
          if (xTaskCreate(task_send_data_mqtt,
00160
00161
                            "Send data to MQTT broker",
00162
                            STACK SIZE.
00163
                            &ucParameterToPass,
00164
                            2.
00165
                            &mattHandle) != pdPASS)
00166
          {
00167
              return ESP_FAIL;
00168
00169
          return ESP OK:
00170
00171 }
00172
00187 void task_show_data_oled(void *args)
00188 {
00189
          dht_data_t sensorData = {0};
00190
          u8g2_t u8g2;
00191
          // OLED Display Setup
00192
          u8g2 = init_oled_display();
00193
00194
           while (true)
00195
00196
               if (xQueueReceive(displayQueue, &sensorData, portMAX_DELAY))
00197
00198
                   struct tm timeinfo:
00199
                   char temp_str[20], hum_str[20], date_str[17], time_str[12];
                   // Parse ISO8601 string back to time
00200
                   strptime(sensorData.timestamp, "%Y-%m-%dT%H:%M:%SZ", &timeinfo);
00201
00202
                   // Format the date as "\mbox{dd/mm/YYYY"}
                   strftime(date_str, sizeof(date_str), "Date: %d/%m/%Y", &timeinfo);
00203
00204
                   // Format the time as "HH:mm"
                   strftime(time_str, sizeof(time_str), "Time: %H:%M", &timeinfo); sprintf(temp_str, "Temp: %.2fC", sensorData.temperature); sprintf(hum_str, "Hum: %.2f %%", sensorData.humidity);
00205
00206
00207
00208
                   show_dht_data(u8g2, date_str, time_str, temp_str, hum_str);
00209
               }
00210
               else
00211
               {
00212
                   ESP_LOGE(TAG, "Error receiving data or no data in buffer");
00213
00214
               vTaskDelay(pdMS_TO_TICKS(100));
00215
          }
00216 }
00217
00234 void task_send_data_mqtt(void *args)
00235 {
00236
          EventBits_t bits = xEventGroupWaitBits(s_wifi_event_group,
                                                    WIFI_CONNECTED_BIT, pdFALSE,
00237
00238
00239
                                                     pdTRUE,
00240
                                                    pdMS_TO_TICKS(portMAX_DELAY));
00241
00242
           if ((bits & WIFI_CONNECTED_BIT) == 1)
00243
00244
               ESP LOGI(TAG, "WiFi connected, starting MOTT task and SNTP");
00245
               setup_mqtt();
00246
               setup_sntp();
00247
00248
00249
          dht_data_t sensorData = {0};
00250
          while (true)
00251
```

```
if ((xQueueReceive(mqttQueue, &sensorData, portMAX_DELAY)) && MQTT_CONNECTED)
00253
00254
                   // Convert to JSON string
                   char *json_str = create_json_payload(&sensorData);
esp_mqtt_client_publish(client, "/home/office/dht", json_str, 0, 0, 0);
00255
00256
00257
                   free(json_str);
00258
00259
00260
                   ESP_LOGE(TAG, "Error receiving data or no data in buffer");
00261
00262
00263
               vTaskDelay(pdMS_TO_TICKS(100));
00264
          }
00265 }
00266
00282 void task_wifi(void *args)
00283 {
00284
          setup wifi();
00285
00286
          while (true)
00287
00288
               EventBits_t bits = xEventGroupWaitBits(s_wifi_event_group,
                                                        WIFI_CONNECTED_BIT,
00289
                                                        pdFALSE.
00290
00291
                                                        pdFALSE,
00292
                                                        pdMs_TO_TICKS(WIFI_RECONNECT_INTERVAL_MS));
00293
00294
               if ((bits & WIFI_CONNECTED_BIT) == 0)
00295
               {
00296
                   esp_wifi_connect();
00297
00298
               vTaskDelay(pdMS_TO_TICKS(1000));
00299
00300 }
00301
00318 void measure_temp_hum(TimerHandle_t timer)
00319 {
          esp_err_t res;
00321
          dht_data_t dhtData;
00322
          res = dht_read_float_data(DHT_SENSOR_TYPE, CONFIG_ESP_TEMP_SENSOR_GPIO, &dhtData.humidity,
      &dhtData.temperature);
00323
          get_current_date_time(dhtData.timestamp);
00324
           if (res == ESP OK)
00325
00326
               if (xQueueSend(displayQueue, &dhtData, pdMS_TO_TICKS(100)) != pdPASS)
00327
00328
                   ESP_LOGE(TAG, "Error sending data to display queue");
00329
00330
               if (xQueueSend(mqttQueue, &dhtData, pdMS_TO_TICKS(100)) != pdPASS)
00331
               {
00332
                   ESP_LOGE(TAG, "Error sending data to MQTT queue");
00333
00334
00335
          else
00336
          {
00337
               ESP_LOGE(TAG, "Error reading data");
00338
00339 }
```

# Index

```
app_main
                                                         init_oled_display, 17
    main.c, 27
                                                         show dht data, 17
                                                    displayQueue
client
                                                         main.c, 30
    main.c, 30
                                                    EXAMPLE_ESP_MAXIMUM_RETRY
    mgtt manager.c, 19
                                                         wifi manager.h, 24
common.c
                                                    EXAMPLE_ESP_WIFI_PASS
    create_json_payload, 7
    get_current_date_time, 7
                                                         wifi_manager.h, 24
    setup sntp, 8
                                                    EXAMPLE_ESP_WIFI_SSID
                                                         wifi_manager.h, 24
    time_sync_notification_cb, 8
common.h
                                                    get_current_date_time
    create json payload, 10
                                                         common.c, 7
    get current date time, 10
                                                         common.h, 10
    MAX Q SIZE, 10
    MEASURE INTERVAL, 10
                                                    humidity
    setup sntp, 11
                                                         dht data t, 5
    STACK SIZE, 10
    TIMER_ID, 10
                                                    I2C SCL
    WIFI_RECONNECT_INTERVAL_MS, 10
                                                         display manager.h, 17
CONFIG_EXAMPLE_INTERNAL_PULLUP
                                                    I2C SDA
    dht manager.h, 13
                                                         display_manager.h, 17
CONFIG_EXAMPLE_TYPE_DHT11
                                                    includes/common.c, 7, 8
    dht manager.h, 13
                                                    includes/common.h, 9, 11
create ison payload
                                                    includes/dht manager.c, 11, 12
    common.c, 7
                                                    includes/dht_manager.h, 12, 14
    common.h, 10
                                                    includes/display_manager.c, 14, 16
create tasks
                                                    includes/display_manager.h, 16, 18
    main.c, 27
                                                    includes/mqtt_manager.c, 18, 20
                                                    includes/mqtt_manager.h, 20, 21
dht_data_t, 5
                                                    includes/wifi manager.c, 21, 23
    humidity, 5
                                                    includes/wifi manager.h, 24, 25
    temperature, 5
                                                    init oled display
    timestamp, 6
                                                         display_manager.c, 15
dht_manager.c
                                                         display_manager.h, 17
    setup dht, 12
                                                    ISO8601 STR LEN
dht manager.h
                                                         dht manager.h, 13
    CONFIG EXAMPLE INTERNAL PULLUP, 13
    CONFIG EXAMPLE TYPE DHT11, 13
                                                    main.c
    DHT SENSOR TYPE, 13
                                                         app_main, 27
    ISO8601_STR_LEN, 13
                                                         client, 30
    setup_dht, 14
                                                         create_tasks, 27
DHT SENSOR TYPE
                                                         displayQueue, 30
    dht manager.h, 13
                                                         measure temp hum, 28
display_manager.c
                                                         MQTT CONNECTED, 30
    init oled display, 15
                                                         mqttQueue, 31
    show dht data, 15
                                                         s wifi event group, 31
display manager.h
                                                         setup timer, 28
    12C SCL, 17
                                                         TAG, 31
    12C SDA, 17
                                                         task_send_data_mqtt, 29
```

36 INDEX

task_show_data_oled, 29	common.c, 8
task wifi, 30	TIMER ID
timerDHT, 31	common.h, 10
main/main.c, 26, 32	timerDHT
MAX_Q_SIZE	main.c, 31
common.h, 10	timestamp
•	•
MEASURE_INTERVAL	dht_data_t, 6
common.h, 10	WIFI_CONNECTED_BIT
measure_temp_hum	
main.c, 28	wifi_manager.h, 24
MQTT_CONNECTED	wifi_event_handler
main.c, 30	wifi_manager.c, 22
mqtt_manager.c, 19	wifi_manager.c
mqtt_event_handler	s_wifi_event_group, 22
mqtt_manager.c, 19	setup_wifi, 22
mqtt_manager.c	wifi_event_handler, 22
client, 19	wifi_manager.h
MQTT CONNECTED, 19	EXAMPLE_ESP_MAXIMUM_RETRY, 24
mqtt_event_handler, 19	EXAMPLE_ESP_WIFI_PASS, 24
setup_mqtt, 19	EXAMPLE_ESP_WIFI_SSID, 24
mqtt_manager.h	setup_wifi, 25
setup_mqtt, 21	WIFI CONNECTED BIT, 24
mqttQueue	WIFI_RECONNECT_INTERVAL_MS
•	common.h, 10
main.c, 31	
s_wifi_event_group	
main.c, 31	
wifi_manager.c, 22	
setup_dht	
dht_manager.c, 12	
dht_manager.h, 14	
setup_mqtt	
mqtt_manager.c, 19	
mqtt_manager.h, 21	
setup_sntp	
common.c, 8	
common.h, 11	
setup_timer	
main.c, 28	
setup_wifi	
wifi_manager.c, 22	
wifi manager.h, 25	
show_dht_data	
display_manager.c, 15	
display_manager.h, 17	
STACK SIZE	
<del>_</del>	
common.h, 10	
TAG	
main.c, 31	
task_send_data_mqtt	
main.c, 29	
task_show_data_oled	
main.c, 29	
task_wifi	
main.c, 30	
temperature	
dht_data_t, 5	
time_sync_notification_cb	