Smart Plantation

Generated by Doxygen 1.12.0

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 adc_sensor.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_adc/adc_oneshot.h"
#include "esp_adc/adc_cali.h"
#include "driver/adc.h"
#include "esp_adc_cal.h"
```

Functions

- float adcToPercentage (uint32_t adcValue)

  Wandelt einen ADC-Wert in einen Prozentwert um.
- void adc_init (adc_channel_t channel, adc_bits_width_t bitwidth)

  Initialisiert den ADC.
- float adc_read_sensor (adc_channel_t channel)

  Liest den Wert des angegebenen ADC-Kanals.
- void adc_cleanup ()

  Bereinigt die ADC-Ressourcen.

Variables

- adc_cali_handle_t adc1_cali_handle
- adc_oneshot_unit_handle_t adc1_handle

2.1.1 Function Documentation

2.1.1.1 adc_cleanup()

```
void adc_cleanup ()
```

Bereinigt die ADC-Ressourcen.

Diese Funktion löscht die Kalibrierungsschemata und gibt die Ressourcen des ADC frei.

2.1.1.2 adc_init()

```
void adc_init (
            adc_channel_t channel,
            adc_bits_width_t bitwidth)
```

Initialisiert den ADC.

Diese Funktion initialisiert den ADC mit der angegebenen Kanal- und Bitbreite und konfiguriert die Kalibrierung.

Parameters

channel	Der ADC-Kanal, der initialisiert werden soll.
bitwidth	Die Bitbreite für den ADC (z. B. ADC_WIDTH_BIT_12).

2.1.1.3 adc_read_sensor()

```
float adc_read_sensor (
            adc_channel_t channel)
```

Liest den Wert des angegebenen ADC-Kanals.

Diese Funktion liest den aktuellen ADC-Wert und gibt ihn als Prozentsatz zurück.

Parameters

channel	Der ADC-Kanal, von dem gelesen werden soll.

Returns

float Der aktuelle ADC-Wert in Prozent.

2.1.1.4 adcToPercentage()

```
float adcToPercentage (
            uint32_t adcValue)
```

Wandelt einen ADC-Wert in einen Prozentwert um.

Diese Funktion rechnet den Rohwert des ADC in einen Prozentsatz um.

Parameters

adcValue	Der ADC-Rohwert, der umgerechnet werden soll.

Returns

float Der umgerechnete Wert in Prozent.

2.1.2 Variable Documentation

2.1.2.1 adc1_cali_handle

```
adc_cali_handle_t adc1_cali_handle
```

2.1.2.2 adc1_handle

```
adc_oneshot_unit_handle_t adc1_handle
```

2.2 adc_sensor.h File Reference

```
#include <stdio.h>
#include "esp_err.h"
#include "driver/adc.h"
#include "esp_adc/adc_oneshot.h"
```

Macros

- #define ADC_CH_4 ADC1_CHANNEL_4

Functions

- void adc_init (adc_channel_t channel, adc_bits_width_t bit_width)
  Initialisiert den ADC.
- float adc_read_sensor (adc_channel_t channel)
  Liest den Wert des angegebenen ADC-Kanals.
- float adcToPercentage (uint32_t adcValue)
  Wandelt einen ADC-Wert in einen Prozentwert um.
- void adc_cleanup ()
  Bereinigt die ADC-Ressourcen.

2.2.1 Macro Definition Documentation

2.2.1.1 ADC_CH_4

```
#define ADC_CH_4 ADC1_CHANNEL_4
```

2.2.2 Function Documentation

2.2.2.1 adc_cleanup()

```
void adc_cleanup ()
```

Bereinigt die ADC-Ressourcen.

Diese Funktion löscht die Kalibrierungsschemata und gibt die Ressourcen des ADC frei.

2.2.2.2 adc_init()

```
void adc_init (
            adc_channel_t channel,
            adc_bits_width_t bitwidth)
```

Initialisiert den ADC.

Diese Funktion initialisiert den ADC mit der angegebenen Kanal- und Bitbreite und konfiguriert die Kalibrierung.

Parameters

channel	Der ADC-Kanal, der initialisiert werden soll.
bitwidth	Die Bitbreite für den ADC (z. B. ADC_WIDTH_BIT_12).

2.2.2.3 adc_read_sensor()

```
float adc_read_sensor (
            adc_channel_t channel)
```

Liest den Wert des angegebenen ADC-Kanals.

Diese Funktion liest den aktuellen ADC-Wert und gibt ihn als Prozentsatz zurück.

Parameters

channel	Der ADC-Kanal, von dem gelesen werden soll.

Returns

float Der aktuelle ADC-Wert in Prozent.

2.2.2.4 adcToPercentage()

```
float adcToPercentage (
            uint32_t adcValue)
```

Wandelt einen ADC-Wert in einen Prozentwert um.

Diese Funktion rechnet den Rohwert des ADC in einen Prozentsatz um.

Parameters

adcValue	Der ADC-Rohwert, der umgerechnet werden soll.

Returns

float Der umgerechnete Wert in Prozent.

2.3 adc_sensor.h

Go to the documentation of this file.

```
00001 #ifndef ADC_SENSOR_H
00002 #define ADC_SENSOR_H
00003
00004 #include <stdio.h>
00005 #include "esp_err.h"
00006 #include "driver/adc.h"
00007 #include "esp_adc/adc_oneshot.h"
00008
00009 // ADC-Kanal-Definition
00010 #define ADC_CH_4 ADC1_CHANNEL_4 // GPIO 32 auf dem ESP
00011
00012 // Funktion zur Kalibrierung und Initialisierung des ADC
00013 void adc_init(adc_channel_t channel, adc_bits_width_t bit_width);
00014
00015 // Funktion zum Lesen des ADC-Werts und Umwandlung in Prozent
00016 float adc_read_sensor(adc_channel_t channel);
00017
00018 // Funktion zur Umrechnung des ADC-Werts in Prozent
00019 float adcToPercentage(uint32_t adcValue);
00020
00021 // Funktion zur Bereinigung des ADC (optional)
00022 void adc_cleanup();
00023
00024 #endif // ADC_SENSOR_H
```

2.4 main.c File Reference

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/queue.h"
#include "task_common.h"
#include "adc_sensor.h"
#include "wifi-server.h"
#include "mdns_server.h"
#include "esp_log.h"
#include "nvs_flash.h"
```

Functions

- void app_main ()

  Hauptanwendung.

2.4.1 Function Documentation

2.4.1.1 app_main()

```
void app_main ()
```

Hauptanwendung.

Diese Funktion wird beim Start der Anwendung aufgerufen. Sie initialisiert den Non-Volatile Storage (NVS) Flash-Speicher, stellt eine WLAN-Verbindung her und erstellt die erforderlichen Tasks für den ADC-Sensor und den Webserver.

2.5 mdns_server.c File Reference

```
#include "mdns_server.h"
#include "esp_log.h"
```

Functions

- void start_mdns (const char *hostname, const char *instance_name, const char *service_type, const char *protocol, uint16_t port)

  Initialisiert den mDNS-Dienst für das Gerät.

2.5.1 Function Documentation

2.5.1.1 start_mdns()

```
void start_mdns (
            const char * hostname,
            const char * instance_name,
            const char * service_type,
            const char * protocol,
            uint16_t port)
```

Initialisiert den mDNS-Dienst für das Gerät.

Diese Funktion richtet den mDNS-Dienst ein, damit das Gerät im lokalen Netzwerk als Dienst angezeigt wird. Es verwendet den angegebenen Hostnamen, Dienstnamen und Netzwerkprotokoll.

Parameters

hostname	Der Hostname, der für mDNS verwendet werden soll.
instance_name	Der Name des Dienstes (z.B. "Smart Plantation").
service_type	Der Typ des Dienstes (z.B. "_http._tcp").
protocol	Das Netzwerkprotokoll (z.B. "tcp").
port	Die Portnummer, auf der der Dienst erreichbar ist.

Returns

void

2.6 mdns_server.h File Reference

```
#include "mdns.h"
```

Macros

- #define MDNS_NAME "Plantation"
- #define INSTANCE_NAME "Smart Plantation"
- #define SERVICE_TYPE "_http._tcp"
- #define PROTOCOL "tcp"
- #define PORT 80

Functions

- void start_mdns (const char ∗hostname, const char ∗instance_name, const char ∗service_type, const char ∗protocol, uint16_t port)

  Initialisiert den mDNS-Dienst für das Gerät.

2.6.1 Macro Definition Documentation

2.6.1.1 INSTANCE_NAME

```
#define INSTANCE_NAME "Smart Plantation"
```

2.6.1.2 MDNS_NAME

```
#define MDNS_NAME "Plantation"
```

2.6.1.3 PORT

```
#define PORT 80
```

2.6.1.4 PROTOCOL

```
#define PROTOCOL "tcp"
```

2.6.1.5 SERVICE_TYPE

```
#define SERVICE_TYPE "_http._tcp"
```

2.6.2 Function Documentation

2.6.2.1 start_mdns()

```
void start_mdns (
            const char * hostname,
            const char * instance_name,
            const char * service_type,
            const char * protocol,
            uint16_t port)
```

Initialisiert den mDNS-Dienst für das Gerät.

Diese Funktion richtet den mDNS-Dienst ein, damit das Gerät im lokalen Netzwerk als Dienst angezeigt wird. Es verwendet den angegebenen Hostnamen, Dienstnamen und Netzwerkprotokoll.

Parameters

hostname	Der Hostname, der für mDNS verwendet werden soll.
instance_name	Der Name des Dienstes (z.B. "Smart Plantation").
service_type	Der Typ des Dienstes (z.B. "_http._tcp").
protocol	Das Netzwerkprotokoll (z.B. "tcp").
port	Die Portnummer, auf der der Dienst erreichbar ist.

Returns

void

2.7 mdns_server.h

Go to the documentation of this file.

```
00001 #ifndef MDNS_SERVER_H
00002 #define MDNS_SERVER_H
00003
00004 #include "mdns.h"
00005
00006 // MDNS Config Definitions
00007 #define MDNS_NAME "Plantation"
00008 #define INSTANCE_NAME "Smart Plantation"
00009 #define SERVICE_TYPE "_http._tcp"
00010 #define PROTOCOL "tcp"
00011 #define PORT 80
00012
00013 // Funktion zur Initialisierung des mDNS-Servers
00014 void start_mdns(const char *hostname, const char *instance_name, const char *service_type, const char
      *protocol, uint16_t port);
00015
00016 #endif // MDNS_SERVER_H
```

2.8 task_common.c File Reference

```
#include "task_common.h"
#include "adc_sensor.h"
#include "wifi-server.h"
#include "mdns_server.h"
#include "esp_log.h"
```

Macros

- #define QUEUE_LENGTH 1
- #define ITEM_SIZE sizeof(float)

Functions

- void adcSensorTask (void *pvParameters)

  ADC Sensor Task.
- void webServerTask (void *pvParameters)

  Webserver und mDNS Task.
- void init_queue ()

  Initialisiert die ADC-Daten-Queue.

Variables

- QueueHandle_t adcDataQueue

  Handle für die ADC-Daten-Queue.

2.8.1 Macro Definition Documentation

2.8.1.1 ITEM_SIZE

```
#define ITEM_SIZE sizeof(float)
```

2.8.1.2 QUEUE_LENGTH

```
#define QUEUE_LENGTH 1
```

2.8.2 Function Documentation

2.8.2.1 adcSensorTask()

```
void adcSensorTask (
            void * pvParameters)
```

ADC Sensor Task.

Diese Funktion liest kontinuierlich ADC-Werte von einem analogen Sensor (ADC_CH_4) und schreibt diese Werte in eine Queue. Es wird jede Sekunde ein neuer Wert gelesen.

Parameters

pvParameters	Pointer zu den übergebenen Parametern (nicht verwendet).

2.8.2.2 init_queue()

```
void init_queue ()
```

Initialisiert die ADC-Daten-Queue.

Diese Funktion erstellt eine Queue zur Kommunikation zwischen den Tasks. Die Queue hat eine definierte Länge und Größe für die darin enthaltenen Elemente.

2.8.2.3 webServerTask()

```
void webServerTask (
            void * pvParameters)
```

Webserver und mDNS Task.

Diese Funktion startet einen Webserver und einen mDNS-Dienst, um das Gerät im Netzwerk auffindbar zu machen. Der Task bleibt in einer Endlosschleife, um die Funktionalität des Servers aufrechtzuerhalten.

Parameters

pvParameters	Pointer zu den übergebenen Parametern (nicht verwendet).

2.8.3 Variable Documentation

2.8.3.1 adcDataQueue

```
QueueHandle_t adcDataQueue
```

Handle für die ADC-Daten-Queue.

2.9 task_common.h File Reference

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
```

Functions

- void adcSensorTask (void *pvParameters)
  ADC Sensor Task.
- void webServerTask (void *pvParameters)
  Webserver und mDNS Task.
- void init_queue ()
  Initialisiert die ADC-Daten-Queue.

Variables

- QueueHandle_t adcDataQueue

2.9.1 Function Documentation

2.9.1.1 adcSensorTask()

```
void adcSensorTask (
            void * pvParameters)
```

ADC Sensor Task.

Diese Funktion liest kontinuierlich ADC-Werte von einem analogen Sensor (ADC_CH_4) und schreibt diese Werte in eine Queue. Es wird jede Sekunde ein neuer Wert gelesen.

Parameters

pvParameters	Pointer zu den übergebenen Parametern (nicht verwendet).

2.9.1.2 init_queue()

```
void init_queue ()
```

Initialisiert die ADC-Daten-Queue.

Diese Funktion erstellt eine Queue zur Kommunikation zwischen den Tasks. Die Queue hat eine definierte Länge und Größe für die darin enthaltenen Elemente.

2.9.1.3 webServerTask()

```
void webServerTask (
            void * pvParameters)
```

Webserver und mDNS Task.

Diese Funktion startet einen Webserver und einen mDNS-Dienst, um das Gerät im Netzwerk auffindbar zu machen. Der Task bleibt in einer Endlosschleife, um die Funktionalität des Servers aufrechtzuerhalten.

Parameters

pvParameters	Pointer zu den übergebenen Parametern (nicht verwendet).

2.9.2 Variable Documentation

2.9.2.1 adcDataQueue

```
QueueHandle_t adcDataQueue  [extern]
```

2.10 task_common.h

Go to the documentation of this file.

```
00001 #ifndef TASK_COMMON_H
00002 #define TASK_COMMON_H
00003
00004 #include "freertos/FreeRTOS.h"
00005 #include "freertos/task.h"
00006 #include "freertos/queue.h"
00007
00008 // Deklaration der Queue für den ADC
00009 extern QueueHandle_t adcDataQueue;
00010
00011 // Funktion zur Initialisierung der Tasks
00012 void adcSensorTask(void *pvParameters);
00013 void webServerTask(void *pvParameters);
00014 void init_queue();
00015
00016 #endif // TASK_COMMON_H
```

2.11 wifi-server.c File Reference

```
#include "wifi-server.h"
#include "esp_log.h"
#include "esp_wifi.h"
#include "nvs_flash.h"
#include "sdkconfig.h"
```

Functions

- httpd_handle_t start_webserver (void)

  Prototyp für die Funktion zum Starten des Webservers.
- void stop_webserver (httpd_handle_t http_server)

  Prototyp für die Funktion zum Stoppen des Webservers.
- const uint8_t index_html_start[ ] asm ("_binary_index_html_start")

  Start der HTML-Datei.
- const uint8_t index_html_end[ ] asm ("_binary_index_html_end")

  Ende der HTML-Datei.
- const uint8_t app_css_start[ ] asm ("_binary_app_css_start")

  Start der CSS-Datei.
- const uint8_t app_css_end[ ] asm ("_binary_app_css_end")

  Ende der CSS-Datei.
- const uint8_t app_js_start[ ] asm ("_binary_app_js_start")

  Start der JS-Datei.
- const uint8_t app_js_end[ ] asm ("_binary_app_js_end")

  Ende der JS-Datei.
- const uint8_t favicon_ico_start[ ] asm ("_binary_favicon_ico_start")

  Start des Favicon.
- const uint8_t favicon_ico_end[ ] asm ("_binary_favicon_ico_end")

  Ende des Favicon.
- esp_err_t http_handler (httpd_req_t ∗req)

  Handler für HTML-Anfragen.
- esp_err_t css_handler (httpd_req_t ∗req)

  Handler für CSS-Anfragen.
- esp_err_t js_handler (httpd_req_t ∗req)

  Handler für JavaScript-Anfragen.
- esp_err_t favicon_handler (httpd_req_t ∗req)

  Handler für Favicon-Anfragen.
- esp_err_t adc_value_handler (httpd_req_t ∗req)

  Handler für ADC-Wert-Anfragen.
- void wifi_connection ()

  Stellt die Verbindung zum WLAN her.

Variables

- const char * ssid = CONFIG_WIFI_SSID

  SSID des WLANs.
- const char * pass = CONFIG_WIFI_PASSWORD

  Passwort für das WLAN.
- int retry_num = 0

  Zähler für Wiederholungsversuche bei der WLAN-Verbindung.
- QueueHandle_t adcDataQueue

  Handle für die ADC-Daten-Queue.
- float current_adc_value = 0.0

  Aktueller ADC-Wert.
- httpd_uri_t http_uri
- httpd_uri_t css_uri
- httpd_uri_t js_uri
- httpd_uri_t favicon_uri
- httpd_uri_t adc_uri

2.11.1 Function Documentation

2.11.1.1 adc_value_handler()

```
esp_err_t adc_value_handler (
            httpd_req_t * req)
```

Handler für ADC-Wert-Anfragen.

Diese Funktion verarbeitet HTTP-Anfragen und gibt den letzten ADC-Wert als JSON zurück.

Parameters

req	Pointer auf die HTTP-Anfrage.

Returns

ESP_OK bei Erfolg.

2.11.1.2 asm() [1/8]

```
const uint8_t app_css_end[] asm (
            "_binary_app_css_end" )  [extern]
```

Ende der CSS-Datei.

2.11.1.3 asm() [2/8]

```
const uint8_t app_css_start[] asm (
            "_binary_app_css_start" )  [extern]
```

Start der CSS-Datei.

2.11.1.4 asm() [3/8]

```
const uint8_t app_js_end[] asm (
            "_binary_app_js_end" )  [extern]
```

Ende der JS-Datei.

2.11.1.5 asm() [4/8]

```
const uint8_t app_js_start[] asm (
            "_binary_app_js_start" )  [extern]
```

Start der JS-Datei.

2.11.1.6 asm() [5/8]

```
const uint8_t favicon_ico_end[] asm (
            "_binary_favicon_ico_end" )  [extern]
```

Ende des Favicon.

2.11.1.7 asm() [6/8]

```
const uint8_t favicon_ico_start[] asm (
            "_binary_favicon_ico_start" )  [extern]
```

Start des Favicon.

2.11.1.8 asm() [7/8]

```
const uint8_t index_html_end[] asm (
            "_binary_index_html_end" )  [extern]
```

Ende der HTML-Datei.

2.11.1.9 asm() [8/8]

```
const uint8_t index_html_start[] asm (
            "_binary_index_html_start" )  [extern]
```

Start der HTML-Datei.

2.11.1.10 css_handler()

```
esp_err_t css_handler (
            httpd_req_t * req)
```

Handler für CSS-Anfragen.

Diese Funktion verarbeitet HTTP-Anfragen für die CSS-Datei.

Parameters

req	Pointer auf die HTTP-Anfrage.

Returns

ESP_OK bei Erfolg.

2.11.1.11 favicon_handler()

```
esp_err_t favicon_handler (
            httpd_req_t * req)
```

Handler für Favicon-Anfragen.

Diese Funktion verarbeitet HTTP-Anfragen für das Favicon.

Parameters

req	Pointer auf die HTTP-Anfrage.

Returns

ESP_OK bei Erfolg.

2.11.1.12 http_handler()

```
esp_err_t http_handler (
            httpd_req_t * req)
```

Handler für HTML-Anfragen.

Diese Funktion verarbeitet HTTP-Anfragen für die HTML-Seite.

Parameters

req	Pointer auf die HTTP-Anfrage.

Returns

ESP_OK bei Erfolg.

2.11.1.13 js_handler()

```
esp_err_t js_handler (
            httpd_req_t * req)
```

Handler für JavaScript-Anfragen.

Diese Funktion verarbeitet HTTP-Anfragen für die JavaScript-Datei.

Parameters

req	Pointer auf die HTTP-Anfrage.

Returns

ESP_OK bei Erfolg.

2.11.1.14 start_webserver()

```
httpd_handle_t start_webserver (
            void )
```

Prototyp für die Funktion zum Starten des Webservers.

Startet den Webserver.

Diese Funktion initialisiert und startet den HTTP-Server und registriert die URI-Handler für HTML, CSS, JS und Favicon.

Returns

httpd_handle_t Handle des gestarteten HTTP-Servers oder NULL bei Fehler.

2.11.1.15 stop_webserver()

```
void stop_webserver (
            httpd_handle_t http_server)
```

Prototyp für die Funktion zum Stoppen des Webservers.

2.11.1.16 wifi_connection()

```
void wifi_connection (
            void )
```

Stellt die Verbindung zum WLAN her.

Diese Funktion initialisiert das WLAN, registriert die Ereignis-Handler und stellt die Verbindung zum definierten WLAN her.

2.11.2 Variable Documentation

2.11.2.1 adc_uri

```
httpd_uri_t adc_uri
```

Initial value:

```
= {
    .uri = "/adc",
    .method = HTTP_GET,
    .handler = adc_value_handler,
    .user_ctx = NULL}
```

2.11.2.2 adcDataQueue

```
QueueHandle_t adcDataQueue  [extern]
```

Handle für die ADC-Daten-Queue.

2.11.2.3 css_uri

```
httpd_uri_t css_uri
```

Initial value:

```
= {
    .uri = "/app.css",
    .method = HTTP_GET,
    .handler = css_handler,
    .user_ctx = NULL}
```

2.11.2.4 current_adc_value

```
float current_adc_value = 0.0
```

Aktueller ADC-Wert.

2.11.2.5 favicon_uri

```
httpd_uri_t favicon_uri
```

Initial value:

```
= {
    .uri = "/favicon.ico",
    .method = HTTP_GET,
    .handler = favicon_handler,
    .user_ctx = NULL}
```

2.11.2.6 http_uri

```
httpd_uri_t http_uri
```

Initial value:

```
= {
    .uri = "/",
    .method = HTTP_GET,
    .handler = http_handler,
    .user_ctx = NULL}
```

2.11.2.7 js_uri

```
httpd_uri_t js_uri
```

Initial value:

```
= {
    .uri = "/app.js",
    .method = HTTP_GET,
    .handler = js_handler,
    .user_ctx = NULL}
```

2.11.2.8 pass

```
const char* pass = CONFIG_WIFI_PASSWORD
```

Passwort für das WLAN.

2.11.2.9 retry_num

```
int retry_num = 0
```

Zähler für Wiederholungsversuche bei der WLAN-Verbindung.

2.11.2.10 ssid

```
const char* ssid = CONFIG_WIFI_SSID
```

SSID des WLANs.

2.12 wifi-server.h File Reference

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/semphr.h"
#include "freertos/queue.h"
#include "esp_event.h"
#include "esp_http_server.h"
```

Functions

- void wifi_connection (void)
  Stellt die Verbindung zum WLAN her.
- httpd_handle_t start_webserver (void)
  Prototyp für die Funktion zum Starten des Webservers.

2.12.1 Function Documentation

2.12.1.1 start_webserver()

```
httpd_handle_t start_webserver (
            void )
```

Prototyp für die Funktion zum Starten des Webservers.

Startet den Webserver.

Diese Funktion initialisiert und startet den HTTP-Server und registriert die URI-Handler für HTML, CSS, JS und Favicon.

Returns

httpd_handle_t Handle des gestarteten HTTP-Servers oder NULL bei Fehler.

2.12.1.2 wifi_connection()

```
void wifi_connection (
             void )
```

Stellt die Verbindung zum WLAN her.

Diese Funktion initialisiert das WLAN, registriert die Ereignis-Handler und stellt die Verbindung zum definierten WLAN her.

2.13 wifi-server.h

Go to the documentation of this file.

```
00001 #ifndef WIFI_WEB_H
00002 #define WIFI_WEB_H
00003
00004 #include "freertos/FreeRTOS.h"
00005 #include "freertos/task.h"
00006 #include "freertos/semphr.h"
00007 #include "freertos/queue.h"
00008 #include "esp_event.h"
00009 #include "esp_http_server.h"
00010
00011 void wifi_connection(void);
00012 httpd_handle_t start_webserver(void);
00013
00014 #endif  // WIFI_WEB_H
```

Index