Controle Wifi

Manual do programador

João Vianna (jvianna@gmail.com)

Versão 0.85

06/05/2024

Referência do Arquivo main.c

Controle de Periféricos através do Wifi.

Funções

void app_main (void)

Função principal do aplicativo.

Descrição detalhada

Este aplicativo cria um protocolo para controle de um módulo esp32 utilizando uma conexão Wifi e um servidor HTTP.

Autor

Joao Vianna (jvianna@gmail.com)

Versão

0.85

Base do código - Exemplos da biblioteca esp-idf

Veja também

app_web_server.c Protocolo HTTP para controlar os periféricos.

controle_gpio.c Interface com o micro-controlador.

Informações sobre mDNS:

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/mdns.html

https://docs.espressif.com/projects/esp-protocols/mdns/docs/latest/en/index.html

Atividades futuras:
Telas HTML embutidas na memória flash, para testar e configurar.
Melhorar documentação de instalação, e desenvolvimento do servidor e cliente.

Definições e Macros

#define USAR_MDNS 1

Funções (detalhes)

void app_main (void)

Ativa os diversos módulos para: Configurar o micro-controlador; Ler a configuração de memória permantente; Iniciar a comunicação via Wifi no modo configurado; Iniciar o servidor HTTP; Iniciar um temporizador para o módulo controle_gpio.c.

Referência do Arquivo app_config.c

Configuração do aplicativo em memória permanente.

Funções

enum modo_conexao app_config_modo_wifi (void)
Indica se modo Wifi deve ser Access Point (AP) ou Station (STA)

const char * app_config_wifi_ssid (void)
Obtém ssid da conexão Wifi.

const char * app_config_wifi_password (void)
Obtém senha da conexão Wifi.

const char * app_config_hostname (void)
Obtém nome do servidor para mDNS.

void app_config_set_modo_wifi (const char * modo)
Altera modo da conexão Wifi.

void app_config_set_wifi_ssid (const char *ssid)
Altera ssid da conexão Wifi.

void app_config_set_wifi_password (const char *pwd)
Altera a senha da conexão Wifi.

void app_config_set_hostname (const char *nome)
Altera o nome do servidor HTTP (mDNS).

void app_config_ler (void)
Ler configuração do aplicativo da memória FLASH.

esp_err_t app_config_gravar (void)
Gravar configuração do aplicativo na memória FLASH.

Descrição detalhada

Camada de abstração que esconde dos demais módulos a maneira como a configuração é armazenada.

Recupera e armazena informações de configuração gravadas na memória FLASH utilizando o sistema de arquivos LittleFS.

NOTA: Existe um pino no micro-controlador que força o retorno à configuração original. Se este pino estiver ativado (ON), a configuração armazenada é ignorada, e os valores 'de fábrica' são utilizados.

Veja também

https://github.com/littlefs-project/littlefs

Autor

Joao Vianna (jvianna@gmail.com)

Versão

0.82 03/05/2024

Pino do micro-controlador força retorno às configurações de fábrica.
Adicionado campo hostname e pequenas mudanças nos nomes dos parâmetros.

Código de armazenamento derivado de:

.../esp-idf/examples/storage/littlefs/main/esp_littlefs_example.c

---

Funções

esp_err_t app_config_gravar (void)

Nota: Se a configuração não foi alterada, nada será gravado.

void app_config_ler (void)

NOTA: Se o pino de voltar à configuração original estiver ligado (ON), ignora a configuração gravada e carrega um padrão 'de fábrica'.

void app_config_set_hostname (const char * *nome*)

Parâmetros

nome	Nome do servidor

Veja também

MAX_HOSTNAME_LEN

void app_config_set_modo_wifi (const char * *modo*)

Parâmetros

modo	"STA" para modo Station, ou "AP" para modo Access Point

void app_config_set_wifi_password (const char * *pwd*)

Parâmetros

pwd	senha a utilizar

Veja também

MAX_PASSWORD_LEN

NOTA: Se o tamanho do parâmetro exceder o limite, nada será alterado.

void app_config_set_wifi_ssid (const char * *ssid*)

Parâmetros

ssid	ssid a utilizar

Veja também

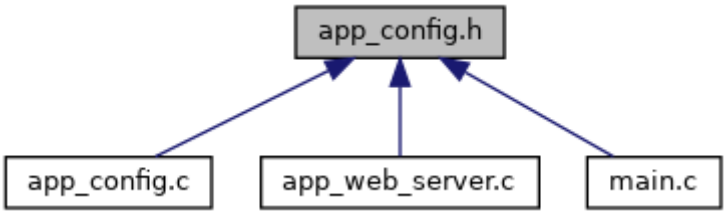
MAX_SSID_LEN

NOTA: Se o tamanho do parâmetro exceder o limite, nada será alterado.

Referência do Arquivo app_config.h

Configuração do aplicativo

Arquivos direta ou indiretamente relacionados com esse arquivo:



Definições e Macros

#define MAX_SSID_LEN 32

Tamanho máximo do ssid.

#define MAX_PASSWORD_LEN 63

Tamanho máximo da senha.

Enumerações

enum modo_conexao_wifi { **MODO_WIFI_STA**, **MODO_WIFI_AP** }

Modo da conexão Wifi (STAtion ou Access Point).

Funções

void app_config_ler (void)
Ler configuração do aplicativo da memória FLASH.

esp_err_t app_config_gravar (void)
Gravar configuração do aplicativo na memória FLASH.

enum modo_conexao wifi app_config_modo_wifi (void)
Indica se modo Wifi deve ser Access Point (AP) ou Station (STA)

const char * app_config_wifi_ssid (void)
Obtém ssid da conexão Wifi.

const char * app_config_wifi_password (void)
Obtém senha da conexão Wifi.

const char * app_config_hostname (void)
Obtém nome do servidor para mDNS.

void app_config_set_wifi_ssid (const char *ssid)
Altera ssid da conexão Wifi.

void app_config_set_wifi_password (const char *pwd)
Altera senha conexão Wifi.

void app_config_set_hostname (const char *nome)
Altera o nome do servidor HTTP (mDNS).

Funções (detalhes)

esp_err_t app_config_gravar (void)

Se a configuração não foi alterada, nada será gravado.

void app_config_ler (void)

NOTA: Se o pino de voltar à configuração original estiver ligado (ON), ignora a configuração gravada e carrega um padrão 'de fábrica'.

void app_config_set_hostname (const char * *nome*)

Parâmetros

nome	Nome do servidor

Veja também

MAX_HOSTNAME_LEN

void app_config_set_modo_wifi (const char * *modo*)

Parâmetros

modo	"STA" para modo Station, ou "AP" para modo Access Point

void app_config_set_wifi_password (const char * *pwd*)

Parâmetros

pwd	senha a utilizar

Veja também

MAX_PASSWORD_LEN

NOTA: Se o tamanho do parâmetro exceder o limite, nada será alterado.

void app_config_set_wifi_ssid (const char * *ssid*)

Parâmetros

ssid	ssid a utilizar

Veja também

MAX_SSID_LEN

NOTA: Se o tamanho do parâmetro exceder o limite, nada será alterado.

Referência do Arquivo app_web_server.c

Servidor HTTP

Funções

httpd_handle_t start_webserver (void)
Iniciar servidor http.

esp_err_t stop_webserver (httpd_handle_t server)
Terminar servidor http.

Descrição detalhada

Um aplicativo cliente controla um módulo contendo atuadores, sensores, etc, utilizando um protocolo HTTP com o servidor.

As solicitações seguem o protocolo:

GET /status

Para obter um texto indicando o status do controlador.

GET /sensor?id=(identificador)

Para ler o estado de um sensor (porta de entrada do módulo), Onde id indica o sensor para o qual se deseja obter o valor.

GET /contador?id=(identificador)

Para ler a contagem de eventos de um contador (porta de entrada do módulo), Onde id indica o contador para o qual se deseja obter o valor.

POST /contador(id)

action=reset

Para reiniciar a contagem de um contador.

POST /atuador(id)

action=("off", "on", "toggle" ou "pulse")

[duration=(tempo em ms)]

Para alterar o estado de um atuador (porta de saída do módulo), Onde “off” desliga, “on” liga, “toggle” alterna o estado e “pulse” liga e desliga após duração de tempo determinada.

POST /config

ssid=(ssid do Wifi)

password=(senha do Wifi)

wifi_mode=(STA/AP)

hostname=(nome do servidor HTTP)

Para alterar a configuração do módulo.

Veja também

app_config.h

Base do código - Exemplos da biblioteca esp-idf

Derivado de Simple HTTPD Server Example

Funções (detalhes)

httpd_handle_t start_webserver (void)

Devolve o handle para o servidor, ou NULL, se houve erro.

esp_err_t stop_webserver (httpd_handle_t *server*)

Parâmetros

server	Handle para o servidor

Variáveis

Ações a que um atuador responde.

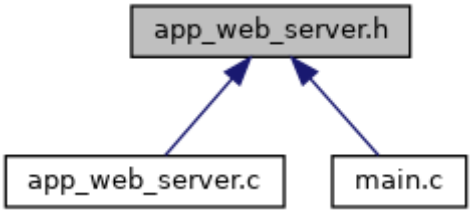
const char* acoes_conhecidas[]

Valor inicial:= {
"off",
"on",
"toggle",
"pulse",
"no action"
}

Referência do Arquivo app_web_server.h

Servidor HTTP

Arquivos direta ou indiretamente relacionados com esse arquivo:



Funções

httpd_handle_t start_webserver (void)
Iniciar servidor http.

esp_err_t stop_webserver (httpd_handle_t server)
Terminar servidor http.

Funções (detalhes)

httpd_handle_t start_webserver (void)

Devolve o handle para o servidor, ou NULL, se houve erro.

esp_err_t stop_webserver (httpd_handle_t *server*)

Parâmetros

server	Handle para o servidor

Referência do Arquivo controle_gpio.c

Controle dos atuadores e sensores.

Funções

void controle_gpio_iniciar (void)
Preparar a placa controladora para operar com o aplicativo.

const char * controle_gpio_status (void)
Obter status da placa controladora.

int controle_gpio_ler_sensor (int id)
Ler valor de um sensor.

int controle_gpio_ler_contador (int id)
Ler contagem de eventos de um contador.

void controle_gpio_reiniciar_contador (int id)
Reiniciar a contagem de um contador para zero.

void controle_gpio_mudar_atuador (int id, int valor)
Mudar o valor de um atuador.

void controle_gpio_alternar_atuador (int id)
Alternar o valor de um atuador entre 0 e 1 (toggle)

void controle_gpio_pulsar_atuador (int id, int duracao)
Cria um pulso positivo no valor de um atuador com certa duração.

bool controle_gpio_reconfig (void)
Indica se o sensor de reconfiguração está ativado.

bool controle_gpio_ativar_timer (void)
Ativa um timer de baixa frequencia para controlar contadores e a duração dos pulsos nos atuadores.

Descrição detalhada

Camada de abstração que esconde os detalhes das ligações das portas do micro-controlador.

Para o aplicativo, existem apenas atuadores, sensores, contadores... O código é específico para o ESP32, mas pode ser re-escrito para outros módulos, mantendo a mesma interface.

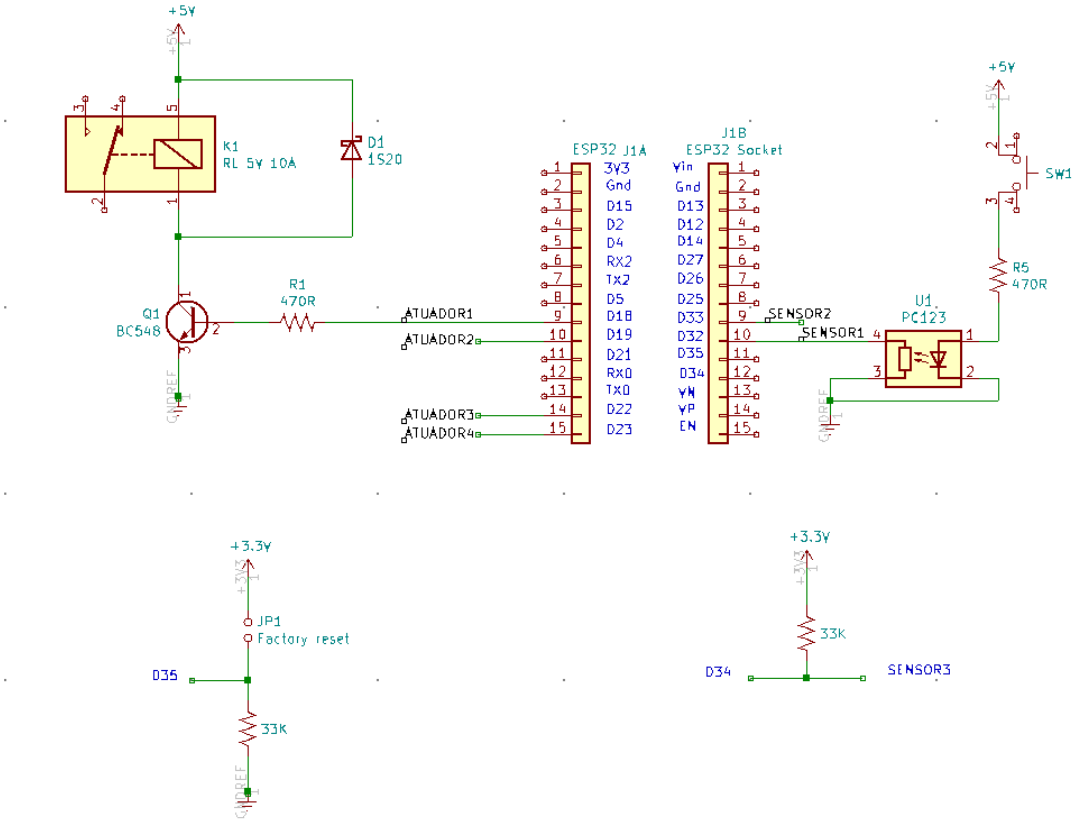
Na abstração, o identificador de cada pino vai de 1 até o limite da classe, independente do pino GPIO a que está conectado. Assim, temos atuador 1, atuador 2, ...; sensor 1, sensor 2, …; contador 1, ...

Um pino especial oferece ao aplicativo um indicador de reconfiguração, para que o sistema possa ser retornado à configuração de fábrica.

Código criado a partir do exemplo:

.../esp-idf/examples/peripherals/gpio/generic_gpio/main/gpio_example_main.c

A implementação corrente é compatível com o circuito no diagrama abaixo:



Nota: Para simplificar o diagrama, apenas os circuitos ligados ao atuador 1 e ao sensor 1 foram representados. Os circuitos para os demais atuadores e sensores é o mesmo.

Funções (detalhes)

void controle_gpio_iniciar (void)

Deve ser ativada no início da lógica do aplicativo, antes da lógica que age sobre os diversos periféricos.

int controle_gpio_ler_sensor (int *id*)

Parâmetros

id	Identificador do sensor (de 1 a MAX_SENSORES)

Retorna 1 se o sensor está em nível ligado e 0 se está desligado.

int controle_gpio_ler_contador (int *id*)

Os eventos são disparados por uma transição de nível alto para nível baixo no pino do contador, que é iniciado como entrada com PULL_UP interno, para simplificar os circuitos eletrônicos.

Parâmetros

id	Identificador do contador (de 1 a MAX_CONTADORES)

Retorna

a contagem dos eventos

void controle_gpio_alternar_atuador (int *id*)

Parâmetros

id	Identificador do atuador (de 1 a MAX_ATUADORES)

void controle_gpio_mudar_atuador (int *id*, int *valor*)

Parâmetros

id	Identificador do atuador (de 1 a MAX_ATUADORES)
valor	0 para desligado; 1 para ligado

void controle_gpio_pulsar_atuador (int *id*, int *duracao*)

O atuador é ligado por um tempo determinado, e depois desligado.

Parâmetros

id	Identificador do atuador (de 1 a MAX_ATUADORES)
duracao	Duração do pulso em milissegundos

Atividades futuras:

Completar implementação utilizando um temporizador.

void controle_gpio_reiniciar_contador (int *id*)

Parâmetros

id	Identificador do contador (de 1 a MAX_CONTADORES)

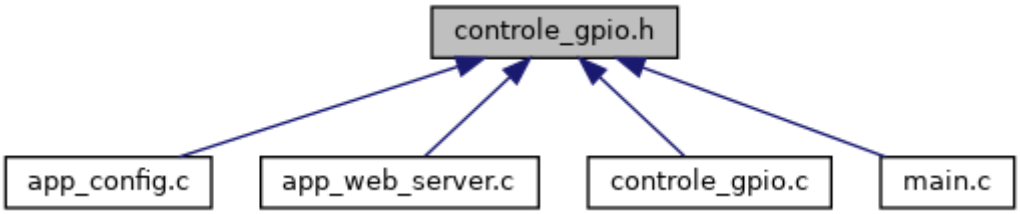
const char* controle_gpio_status (void)

Devolve texto indicando o estado do módulo (número de dispositivos, etc.)

Referência do Arquivo controle_gpio.h

Controle dos atuadores e sensores.

Arquivos direta ou indiretamente relacionados com esse arquivo:



Definições e Macros

#define MAX_ATUADORES 4
Máximo de atuadores no módulo.

#define MAX_CONTADORES 1
Máximo de contadores no módulo.

#define MAX_SENSORES 2
Máximo de sensores no módulo.

Enumerações

enum periferico {
PRF_NDEF,
PRF_ATUADOR,
PRF_CONTADOR,
PRF_SENSOR }
Tipos de perifericos controlados.

Funções

void controle_gpio_iniciar (void)

Preparar a placa controladora para operar com o aplicativo.

const char * controle_gpio_status (void)

Obter status da placa controladora.

bool controle_gpio_ativar_timer (void)

Ativa um timer de baixa frequencia para controlar contadores e a duração dos pulsos nos atuadores.

int controle_gpio_ler_sensor (int id)

Ler valor de um sensor.

int controle_gpio_ler_contador (int id)

Ler contagem de eventos de um contador.

void controle_gpio_mudar_atuador (int id, int valor)

Mudar o valor de um atuador.

void controle_gpio_reiniciar_contador (int id)

Reiniciar a contagem de um contador para zero.

void controle_gpio_alternar_atuador (int id)

Alternar o valor de um atuador entre 0 e 1 (toggle)

void controle_gpio_pulsar_atuador (int id, int duracao)

Cria um pulso positivo no valor de um atuador com certa duração.

bool controle_gpio_reconfig (void)

Indica se o sensor de reconfiguração está ativado.

Enumerações (detalhes)

enum periferico

Enumeradores:

PRF_NDEF	Periférico não definido.
PRF_ATUADOR	Atuador binário (0 - Off, 1 - On)
PRF_CONTADOR	Contador de eventos
PRF_SENSOR	Sensor binário (0 - Off, 1 - On)

Funções (detalhes)

void controle_gpio_iniciar (void)

Deve ser ativada no início da lógica do aplicativo, antes da lógica que age sobre os diversos periféricos.

int controle_gpio_ler_sensor (int *id*)

Parâmetros

id	Identificador do sensor (de 1 a MAX_SENSORES)

Retorna 1 se o sensor está em nível ligado e 0 se está desligado.

int controle_gpio_ler_contador (int *id*)

Os eventos são disparados por uma transição de nível alto para nível baixo no pino do contador, que é iniciado como entrada com PULL_UP interno, para simplificar os circuitos eletrônicos.

Parâmetros

id	Identificador do contador (de 1 a MAX_CONTADORES)

Retorna

a contagem dos eventos

void controle_gpio_reiniciar_contador (int *id*)

Parâmetros

id	Identificador do contador (de 1 a MAX_CONTADORES)

void controle_gpio_alternar_atuador (int *id*)

Parâmetros

id	Identificador do atuador (de 1 a MAX_ATUADORES)

void controle_gpio_mudar_atuador (int *id*, int *valor*)

Parâmetros

id	Identificador do atuador (de 1 a MAX_ATUADORES)
valor	0 para desligado; 1 para ligado

void controle_gpio_pulsar_atuador (int *id*, int *duracao*)

O atuador é ligado por um tempo determinado, e depois desligado.

Parâmetros

id	Identificador do atuador (de 1 a MAX_ATUADORES)
duracao	Duração do pulso em milissegundos

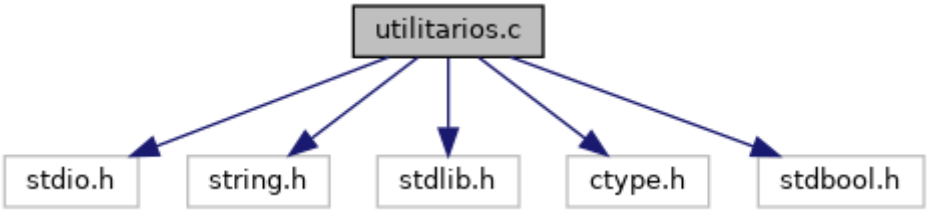
const char* controle_gpio_status (void)

Devolve texto indicando o estado do módulo (número de dispositivos, etc.)

Referência do Arquivo utilitarios.c

Diversos utilitários para lidar com strings, etc.

Dependência de inclusões para utilitarios.c:



Funções

bool str_startswith (const char *str, const char *substr)

Função auxiliar que verifica se um string se inicia com determinada sequência.

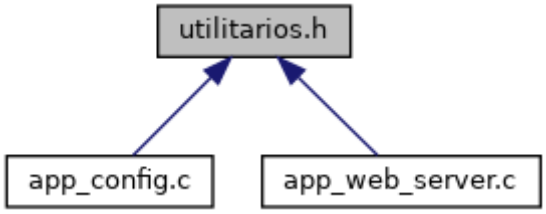
void urldecode2 (char *dst, const char *src)

Função auxiliar para decodificar uma URI.

Referência do Arquivo utilitarios.h

Diversos utilitários para lidar com strings, etc.

Arquivos direta ou indiretamente relacionados com esse arquivo:



Funções

bool str_startswith (const char *str, const char *substr)

Função auxiliar - Verifica se um string se inicia com determinada sequência.

void urldecode2 (char *dst, const char *src)

Função auxiliar para decodificar uma URI.

Funções (detalhes)

bool str_startswith (const char * *str*, const char * *substr*)

Parâmetros

str	string que vai ser verificada.
substr	substring que deveria aparecer no início de str.

Retorna

true se os strings são iguais até o fim do segundo.

void urldecode2 (char * *dst*, const char * *src*)

Parâmetros

dst	Destino do resultado (pode ser o mesmo que a origem).
src	Começa com a URL original, a ser transformada;

Código obtido na Internet

Veja também

https://gist.github.com/jmsaavedra/7964251

Referência do Arquivo wifi_softap.c

Inicia Wifi no modo soft Access Point.

Funções

void wifi_init_softap (const char *ssid)

Iniciar serviço de Wifi no modo Soft AP (Access Point)

Descrição detalhada

Código original em:

.../esp-idf/examples/wifi/getting_started/softAP/

NOTA: Houve modificações do código original do exemplo conforme outros exemplos encontrados na Internet para permitir o uso juntamente com um servidor HTTP.

Atividades futuras:

Usar mDNS para publicar nome do servidor. Até esta versão, em modo AP, o IP do servidor é fixo (192.168.0.10).

Funções (detalhes)

void wifi_init_softap (const char * *ssid*)

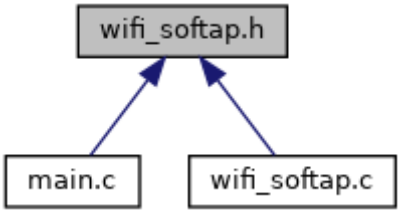
Parâmetros

ssid	Identificador do Access Point

Referência do Arquivo wifi_softap.h

Inicia Wifi no modo soft Access Point.

Arquivos direta ou indiretamente relacionados com esse arquivo:



Funções

void wifi_init_softap (const char *ssid)

Iniciar serviço de Wifi no modo Soft AP (Access Point)

Funções (detalhes)

void wifi_init_softap (const char * *ssid*)

Parâmetros

ssid	Identificador do Access Point

Referência do Arquivo wifi_station.c

Inicia Wifi no modo Station.

Funções

void wifi_init_sta (const char *ssid, const char *pwd)
Iniciar serviço de Wifi no modo Station.

Descrição detalhada

Código original em:

.../esp-idf/examples/wifi/getting_started/station/main/station_example_main.c

NOTA: Foram realizadas modificações mínimas, passando o ssid e senha como parâmetros para a função de iniciar.

Funções (detalhes)

void wifi_init_sta (const char * *ssid*, const char * *pwd*)

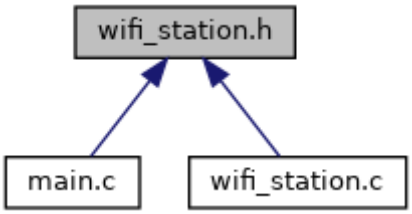
Parâmetros

ssid	Identificador do serviço Wifi com que conectar.
pwd	Senha a utilizar para se conectar.

Referência do Arquivo wifi_station.h

Inicia Wifi no modo Station.

Arquivos direta ou indiretamente relacionados com esse arquivo:



Funções

void wifi_init_sta (const char *ssid, const char *pwd)

Iniciar serviço de Wifi no modo Station.

Funções (detalhes)

void wifi_init_sta (const char * *ssid*, const char * *pwd*)

Parâmetros

ssid	Identificador do serviço Wifi com que conectar.
pwd	Senha a utilizar para se conectar.

Lista de atividades futuras

Global controle_gpio_ler_sensor (int id)

Acrescentar sensores do tipo ADC, com valores inteiros.

Arquivo main.c

Telas HTML embutidas na memória flash, para testar e configurar.
Melhorar documentação de instalação, e desenvolvimento do servidor e cliente.

Arquivo wifi_softap.c

Usar mDNS para publicar nome do servidor no modo Soft AP.
Até esta versão, em modo AP, o IP do servidor é fixo (192.168.0.10).

Índice de Arquivos Fonte