Controle Wifi

Manual do programador

João Vianna (jvianna@gmail.com)

Versão 0.8

29/04/2024

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.

Author: Joao Vianna (jvianna@gmail.com)

Version: 0.8

Base do código - Exemplos da biblioteca esp-idf

Veja também

app_web_server.c Protocolo HTTP no arquivo.

controle_gpio.c Interface com o micro-controlador no arquivo.

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.

Referência do Arquivo app_config.c

Configuração do aplicativo em memória permanente.

Funções

bool str_startswith (const char *str, const char *substr)
Função auxiliar - Verifica se um string se inicia com determinada sequência.

bool app_config_softap (void)
Indica se modo Wifi deve ser Access Point (AP)

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

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

void app_config_set_softap (int 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 senha conexão Wifi.

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

Cria uma 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

Author: Joao Vianna (jvianna@gmail.com) Version: 0.5

Código de armazenamento derivado de:

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

---

Funções

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_softap (int *modo*)

Parâmetros

modo	0 para modo Station, outro 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.

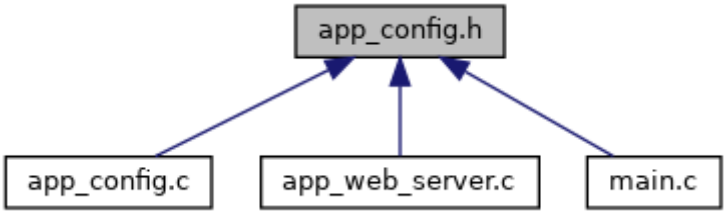
bool str_startswith (const char * *str*, const char * *substr*)

AFAZER: Mover para módulo de utilitarios.

Referência do Arquivo app_config.h

Configuração do aplicativo

Este grafo mostra quais arquivos estão 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.

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.

bool app_config_softap (void)
Indica se modo Wifi deve ser Access Point (AP)

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

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

void app_config_set_softap (int 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 senha conexão Wifi.

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_softap (int *modo*)

Parâmetros

modo	0 para modo Station, outro 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 /atuar?at=(id)&v=(valor) Para alterar o estado de um atuador (porta de saída do módulo), Onde at indica o atuador e v é 0 para desligar e 1 para ligar.

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

GET /config?ssid=(ssid)&pd=(password)&ap=(ap_mode?) 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 e Example: GPIO

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*)

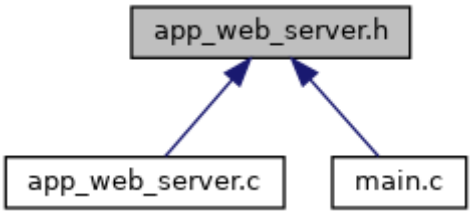
Parâmetros

server	Handle para o servidor

Referência do Arquivo app_web_server.h

Servidor HTTP

Este grafo mostra quais arquivos estão 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.

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

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

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.

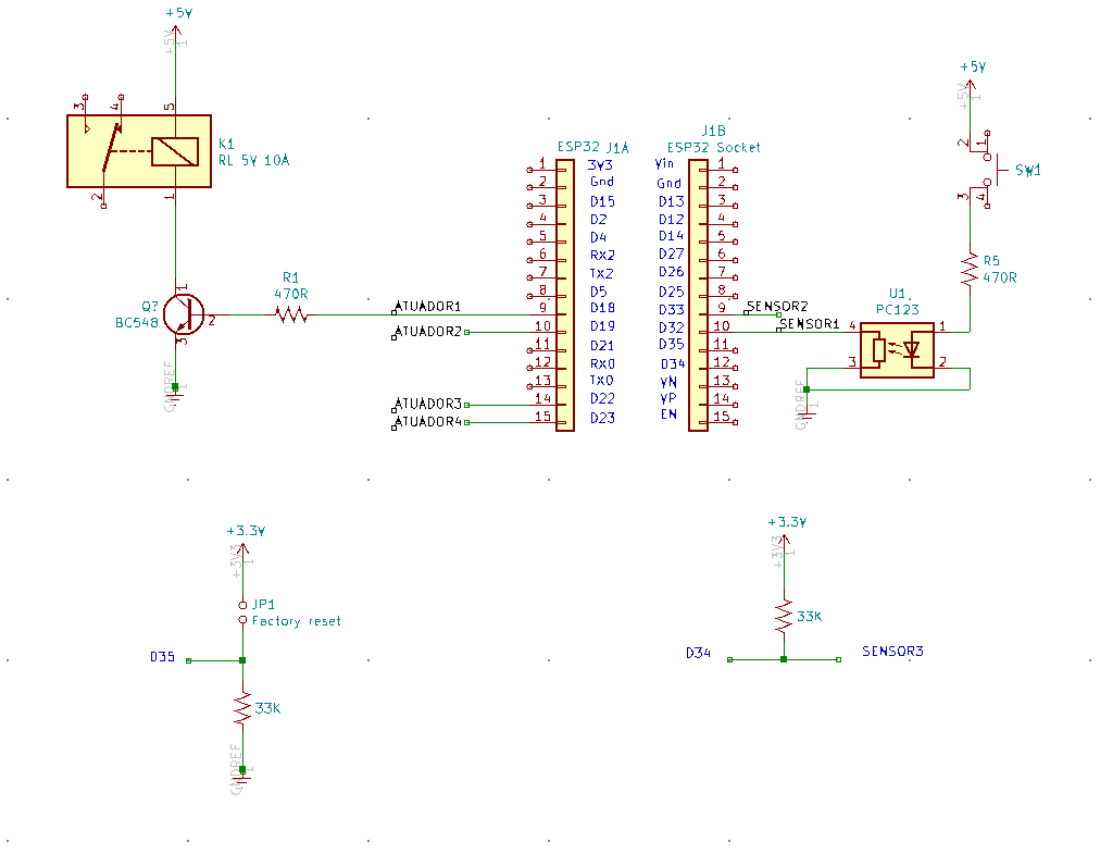
Na abstração, o identificador de cada pino vai de 1 até o limite da classe, independente do pino GPIO a que está conectado. Assim, temos atuador 1, atuador 2, ...; sensor 1, sensor 2, ...

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:



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.

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

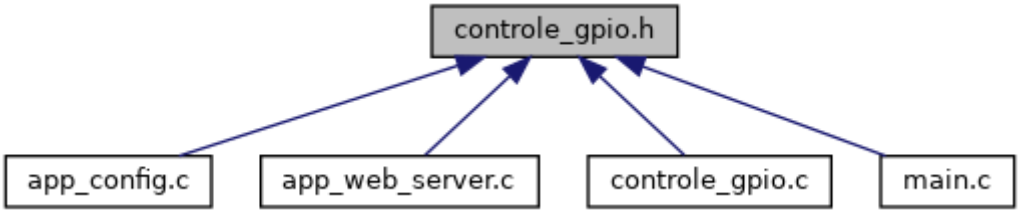
const char* controle_gpio_status (void)

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

Referência do Arquivo controle_gpio.h

Controle dos atuadores e sensores.

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Definições e Macros

#define MAX_ALARMES 1

Máximo de alarmes no módulo.

#define MAX_ATUADORES 4

Máximo de atuadores no módulo.

#define MAX_SENSORES 2

Máximo de sensores no módulo.

Enumerações

enum periferico { PRF_NDEF, PRF_ATUADOR, PRF_ALARME, 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.

int controle_gpio_ler_sensor (int id)

Ler valor de um sensor.

void controle_gpio_mudar_atuador (int id, int valor)

Mudar o valor de um atuador.

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_ALARME	Alarme (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.

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

const char* controle_gpio_status (void)

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

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.

Funções (detalhes)

void wifi_init_softap (const char * *ssid*)

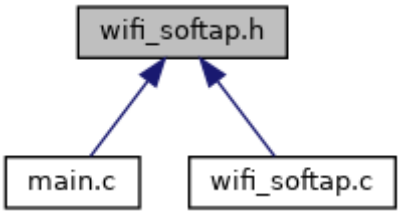
Parâmetros

ssid	Identificador do Access Point

Referência do Arquivo wifi_softap.h

Inicia Wifi no modo soft Access Point.

Este grafo mostra quais arquivos estão 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*)

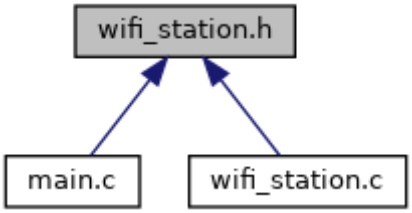
Parâmetros

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.

Este grafo mostra quais arquivos estão 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.

Índice de Arquivos Fonte