## OTA über HTTPS

Geändert 2019-04-25 Erstellt 2019-04-25

Speziell: Zertifikat in Code einbinden

Espressif Doku OTA über HTTPS: https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/system/esp_https_ota.html

Espressif Doku http-client: https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/protocols/esp_http_client.html
Espressif Example: https://github.com/espressif/esp-idf/blob/a20d02b7f196c407bc9f39b781e31a0a4f665968/examples/protocols/esp_http_client/main/esp_http_client_example.c

Tutorial:

# ESP32 (37) – https OTA

LUCA 27/10/2018 14

In the previous post of this tutorial, I explained how it is possible to update your board *Over-The-Air* thanks to a feature of the Freshen IoT dashboard.

Today I'll show you how to update the *firmware* running on an **esp32** chip using only components included in the esp-idf framework, without the need of any external tools or platforms.

### OTA API

The esp-idf framework offers a set of *native* functions to implement, in your program, the ability to be updated over the air.

Those functions are grouped in the **app_update** component and to use them in your program you have to include the corresponding *header* file:

```
#include "esp_ota_ops.h"
```

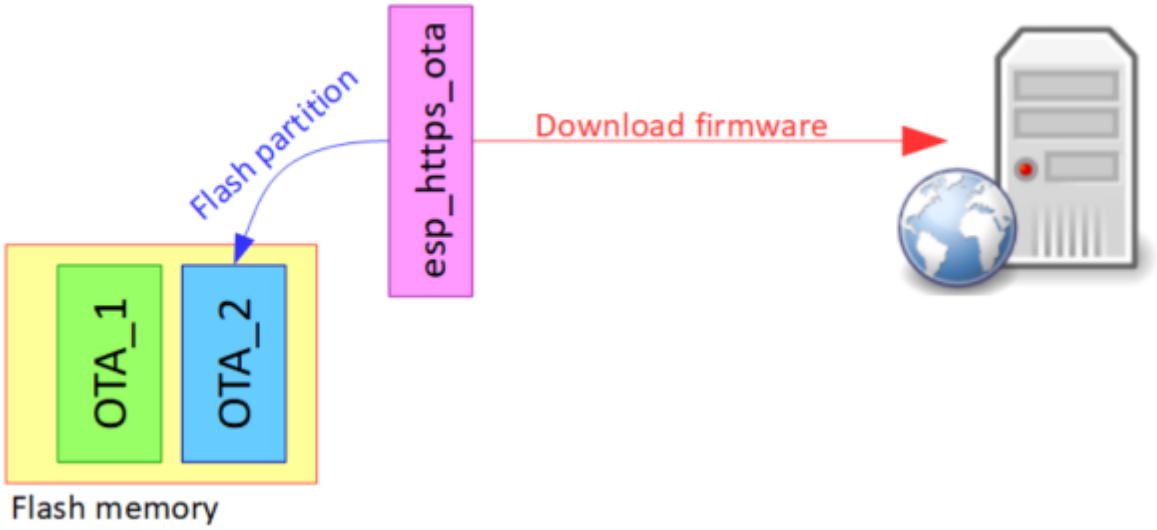
Altough the use of the **native** functions is not very difficult (on Github you can find an example program), Espressif developers have added a component to the framework that makes it even easier the *over the air* update if the new firmware is located on a web site.

The component is named esp_https_ota.

### esp_https_ota

The esp_https_ota component uses the OTA API to update the firmware of your board, downloading the binary file that contains the new firmware from a web site. As the name suggests, the only requirement (for security reason) is that the web site supports the **secure** version of the protocol (HTTPS).

The component is able to automatically identify an OTA partition in the flash memory that is **not in use** and to save the new firmware in that partition. It then configures the chip to boot from that partition:




The use is very simple. First create an **esp_http_client_config_t** *struct* to configure the URL of the file with the new firmware and the SSL **certificate** of the server (or the certificate of the CA that signed it):

```
esp_http_client_config_t ota_client_config = {
  .url = "https://mysite.com/newfirmware.bin",
  .cert_pem = server_cert_pem_start,
};
```

You have to provide the certificate in **PEM** format. To store the certificate in your program, you can leverage the *embedding binary files* functionality of the firmware, as I already explained in a previous tutorial.

Then you only have to call the function:

```
esp_err_t ret = esp_https_ota(&ota_client_config);
```

to start the update process. If – when the process is complete – the **ret** variable contains a positive result (**ESP_OK**), you can reboot the chip to run the new firmware:

```
esp_restart();
```

A real application would probably need to periodically check if a new firmware is available and, only in that case, to start the update process. How can it be done?

In the program I wrote for this post and that is explained in the video below, I'm going to show a way widely used also in commercial products… enjoy the show

as usual, the source code of the program is available in my Github repository


ESP32 OTA via HTTPS