スマホでLED操作編

- ESP32におけるWi-Fi、Webサーバ機能の実装
- SPIFFSによるファイル操作

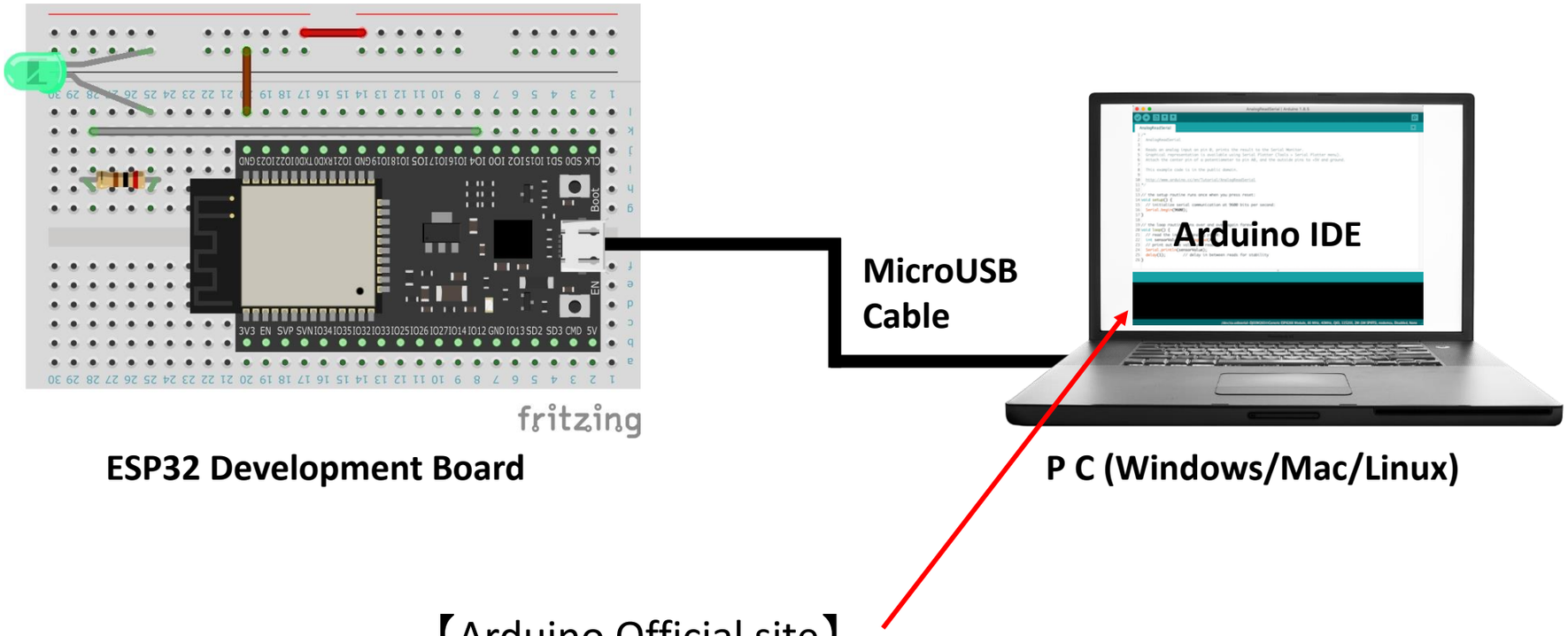
目　次　《スマホでLED操作編》

１－１．スマートリモコン製作全体の流れ

No	項目	内容	ハード	ソフト	記事
1	概要	全体の流れ、システム構成、利用物品、 選定理由、開発環境など	-	-	別動画で配信
2	LED	初めて電子工作される方向けの基本を行います。 LEDの点灯、点滅を行う「Lチカ」を製作します。	○	○	
3	赤外線受信センサ	赤外線受信センサーの説明 回路図から配線、ソフトウェア	○	○	
4	赤外線送信LED	赤外線送信LEDの説明 回路図から配線、ソフトウェア	○	○	
5	スマホでLED操作 （宅内）	工作したリモコンのLEDを屋内のスマホから操作する ソフトウェアを製作します。（Webサーバ機能、SPIFFS操作）	-	○	今回はこの動画
6	スマホでリモコン操作 （宅内）	工作したリモコンを屋内のスマホから操作する ソフトウェアを製作します。（ボタン名、信号保存・読出）	-	○	別動画で配信
7	屋外からスマホで操作 及び、AIスピーカ連携	工作したリモコンを屋外からスマホで操作したり AIスピーカ連携を実現するソフトウェアを製作します。	-	○	

１－２．開発環境Arduinoについて

開発環境はArduinoを利用していきます。



【Arduino Official site】
https://www.arduino.cc/
ダウンロード可能

2. Wi-Fi接続

ESP32において、Wi-Fi接続は以下のようにプログラミングできる。

```
const char *ssid          = "##### SSID #####";
const char *password  = "### PASSWORD ###";
IPAddress ip(192, 168, 1, 123);     // IP address (IP used by this machine)
IPAddress gateway(192, 168, 1, 1);  // default gateway
IPAddress subnet(255, 255, 255, 0); // sub-net mask
```

最初に利用する環境にあった設定を定義します。

Setup関数内

```
// Wireless Wi-Fi connection
WiFi.config( ip, gateway, subnet );
WiFi.begin ( ssid, password );

// Wi-Fi connection processing (infinite loop until connected)
while ( WiFi.status() != WL_CONNECTED ) {
    // Wait for 1 second
    delay ( 1000 );
    Serial.print ( "." );
}
Serial.print ( "Wi-Fi Connected! IP address: " );
Serial.println ( WiFi.localIP() );
```

定義した設定値をセットします。
定義したSSIDとパスワードで接続を開始します。

Wi-Fiが接続状態になるまで1秒毎に状態を確認します。
接続状態になるまで繰り返します。

接続状態になるとIPアドレスをシリアルモニタに表示します。

３．Webサーバ機能

Webサーバの機能はライブラリを利用して、実現していきます。

- ライブラリとは、「特定の機能を部品化しているもの」です。

今回利用するライブラリ　（概要欄にリンクがあります）

①ESPAsyncWebServer
https://github.com/me-no-dev/ESPAsyncWebServer

②AsyncTCP
https://github.com/me-no-dev/AsyncTCP

3．Webサーバ機能

ライブラリを追加後にWebサーバのプログラミングを実施する。

```
#include <ESPAsyncWebServer.h>
```

ライブラリを利用できるように取り込みます。

```
AsyncWebServer webServer ( 80 );
```

Webサーバの利用とポート番号を定義
（規定されているHTTPポートの80を利用）

Setup関数

```
// Set WebServer reception process "/"
 webServer.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
   sendHtml(request);          // Send web page content
   Serial.println ( "TOP page" );
 });
 // Set WebServer reception processing "/on"
 webServer.on("/on", HTTP_GET, [](AsyncWebServerRequest *request){
   digitalWrite(LED_PIN, HIGH);  // LED is lit by setting the LED pin to HIGH
   Serial.println ( "LED ON" );
   sendHtml(request);          // Send web page content
 });
～　省略　～
 // WebServer startup processing
 webServer.begin();
```

“/”つまりサイトのTOPにアクセスされた場合に
処理する内容をプログラムします。

サイトURLの“/on” にアクセスされた場合に
処理する内容をプログラムします。
アクセスがあれば処理する内容を同様に追加
していきます。

プログラムした内容で、Webサーバを起動します。

5. SPIFFS

SPIFFS機能を利用し、Flashメモリをファイルシステムとして利用します。

SPIFFSとは

SPIFFS (SPI Flash File System) の略で、接続されているFlashメモリをファイルシステムとして利用する方法
SPI（Serial Peripheral Interface）は、マイクロコントローラとその周辺ICの間で利用されるインターフェースで、
ESP32ではFlashメモリの接続に使われています。

- ESP32データシート（公式）より

2 Block Diagram

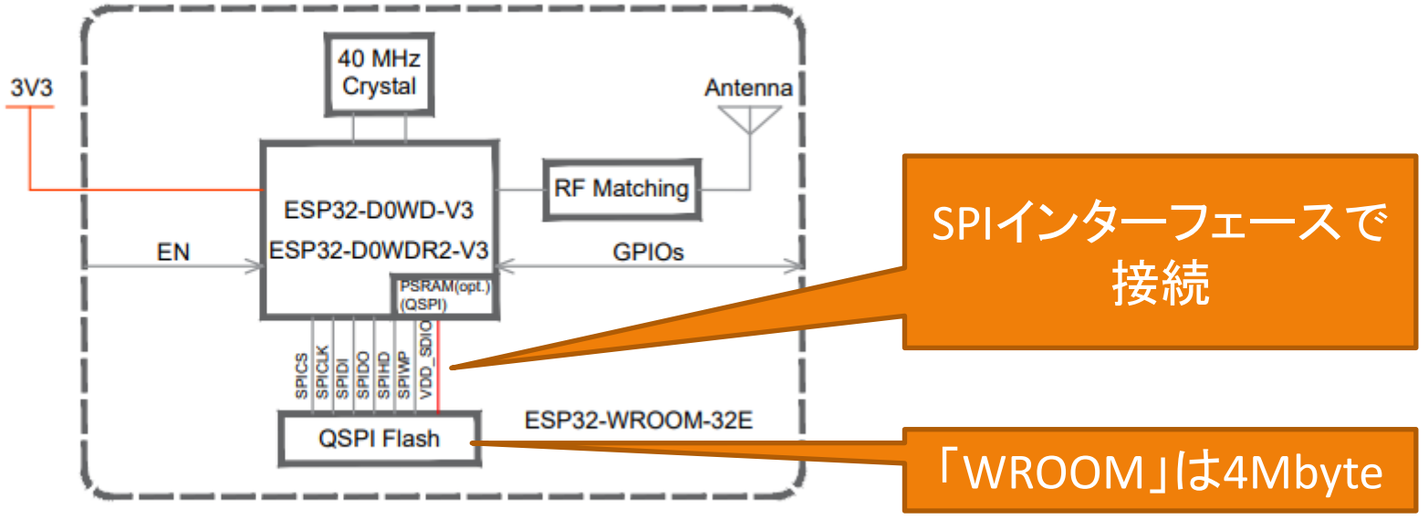


Figure 1: ESP32-WROOM-32E Block Diagram

SPIインターフェースで接続

「WROOM」は4Mbyte

【SPIFFS】
この接続されているFlashメモリの一部をファイルシステムとして利用する

https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf