www.aindustriosa.org

WORKSHOP

ESPRESSIF

ESP8266

centum
tech emotion

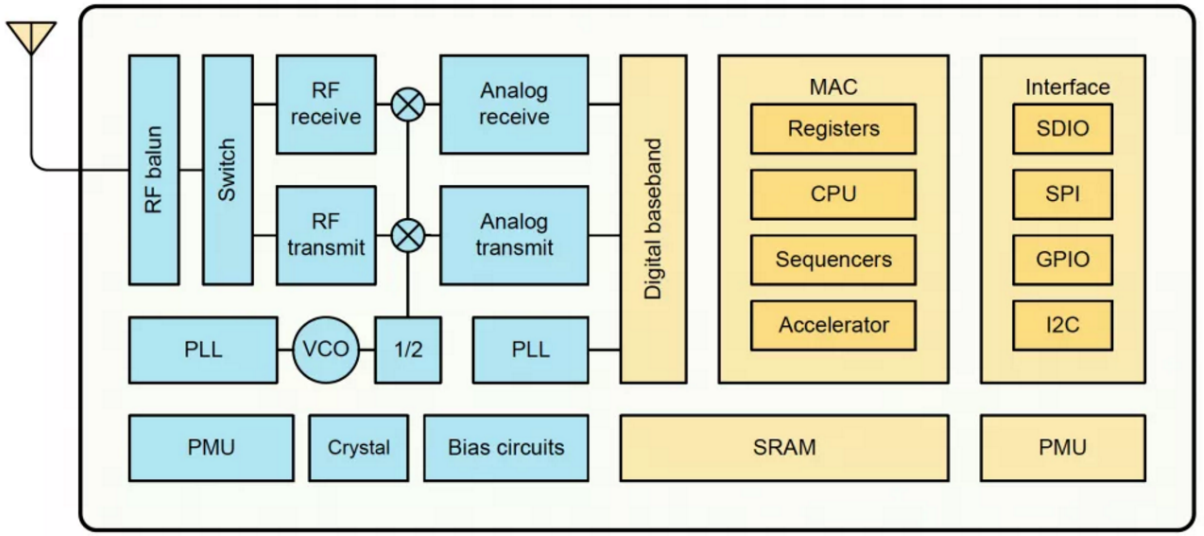
WORKSHOP

ESPRESSIF

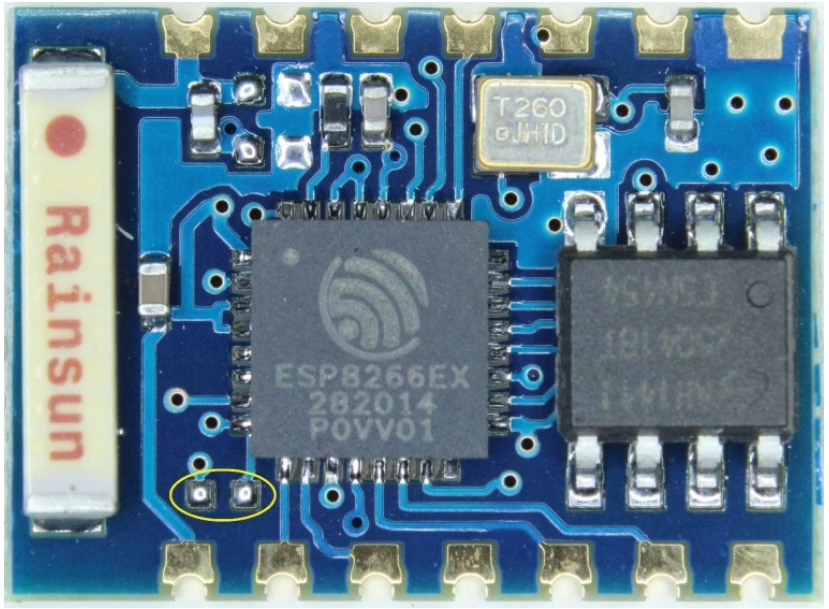
ESP8266

- Core ESP8266
- Tipos de Módulos
- Esquema de conexión PCB
- Placa NodeMCU
- SDK y Arduino Core
- Callbacks
- GPIO e Interrupciones
- Multiplexado
- PWM
- ADC
- Delays
- Serial Port
- “Timers”

- Flashear ESP8266
- ESP8266 STATION
- ESP8266 EVENT HANDLER
- ESP8266 como AP
- Wifi Manager
- Conexiones Http
- Cliente Http Json
- ESP8266 Web Server
- FSBrowser
- SPIFFSS



ESPRESSIF



Tensilica Xtensa L106

- 80 MHz* (overclokeable a 160MHz)
- 64 KiB of instruction RAM,
- 96 KiB of data RAM
- External QSPI flash: 512 KiB to 4 MiB* (up to 16 MiB is supported)

Tensilica is a company based in Silicon Valley
Tensilica is known for its customizable microprocessor core, the Xtensa configurable processor

ESP-01

ESP-02

ESP-03

ESP-04

ESP-05

ESP-06

ESP-07

ESP-08

ESP-09

ESP-10

ESP-11

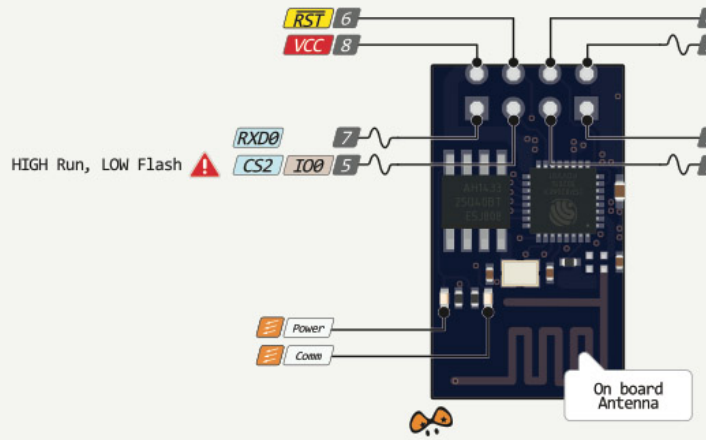
ESP8266 Antenna

ESP8266 PINOUT

All ESP8266 in/outs are NOT 5V tolerant!

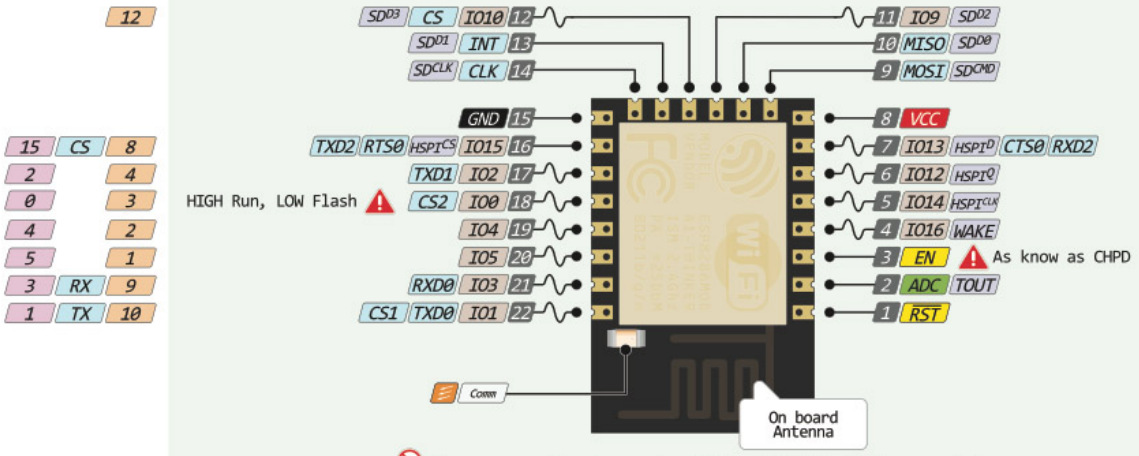
ESP-01

Absolute MAX per pin 12mA
recommended 6mA



ESP-12

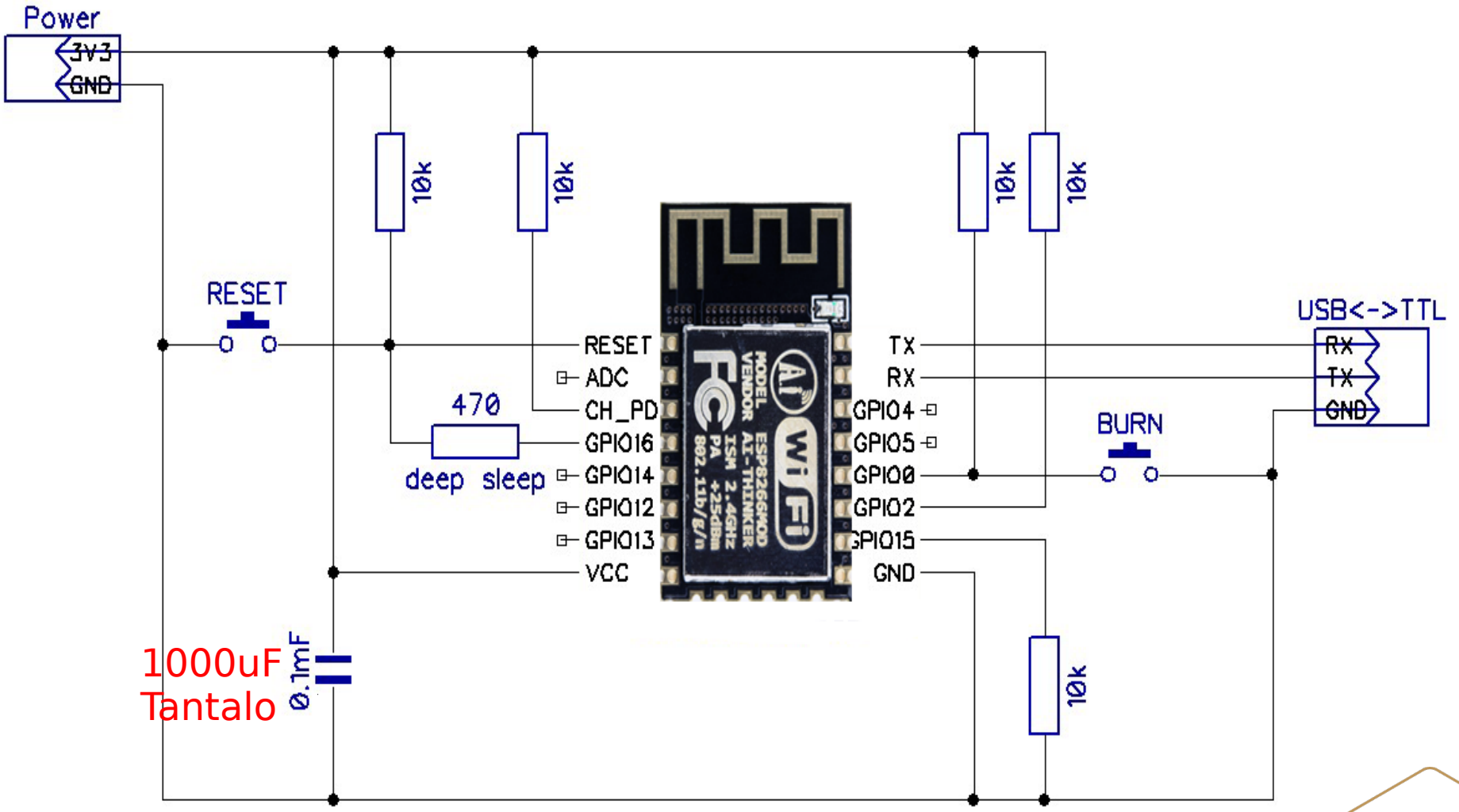
Absolute MAX per pin 12mA
recommended 6mA



When you use the sleep mode, IO16 and RST should be connected and IO16 will output LOW to reset the system at the time of wakeup.

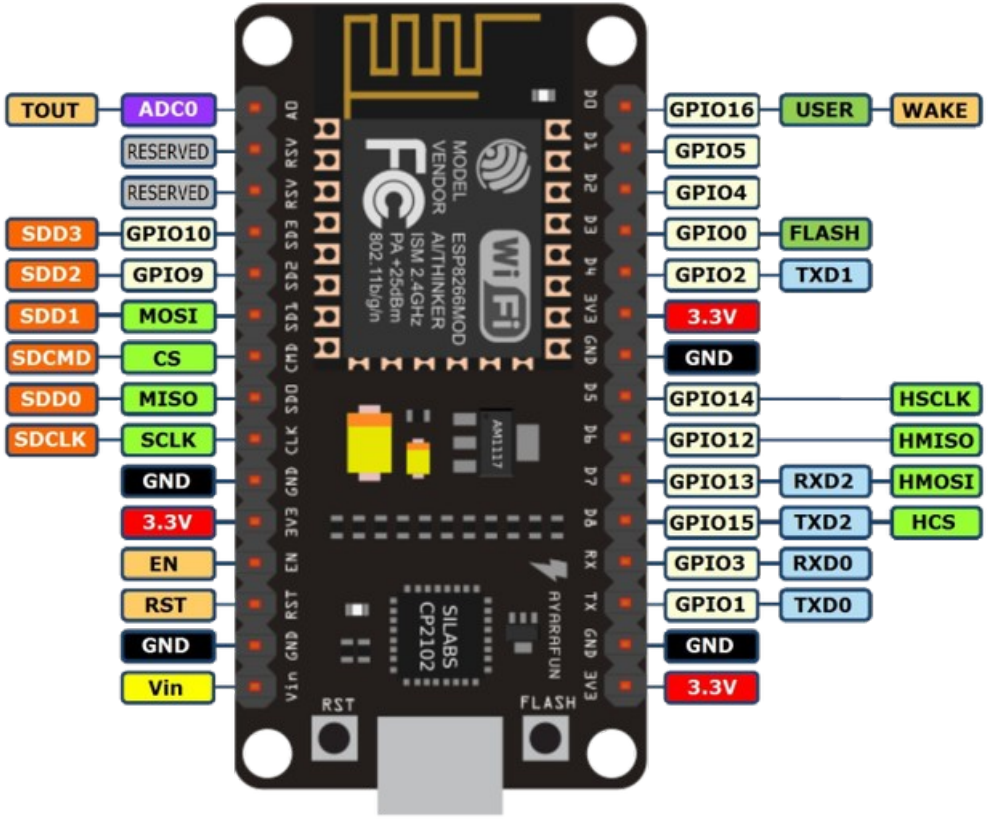
On every boot/reset/wakeup IO15 must keep LOW, IO2 must keep HIGH.

CH_PD → desctiva chip 3.3v
REST → resetea chip 5V



ESP8266

NodeMCU



- Alimentación **3.3V**
- La mayor parte del tiempo consume unos **70mA**, pero alcanza consumos de **250mA** regularmente.
- **Excepcionalmente puede alcanzar picos de 500mA**
- Los GPIO pins **NO SON 5V COMPATIBLES** excepto RX (incorpora un level shifter)
- Alcance en espacio abierto con antena integrada **90m**
- Alcance en espacio abierto con antena externa **150m**

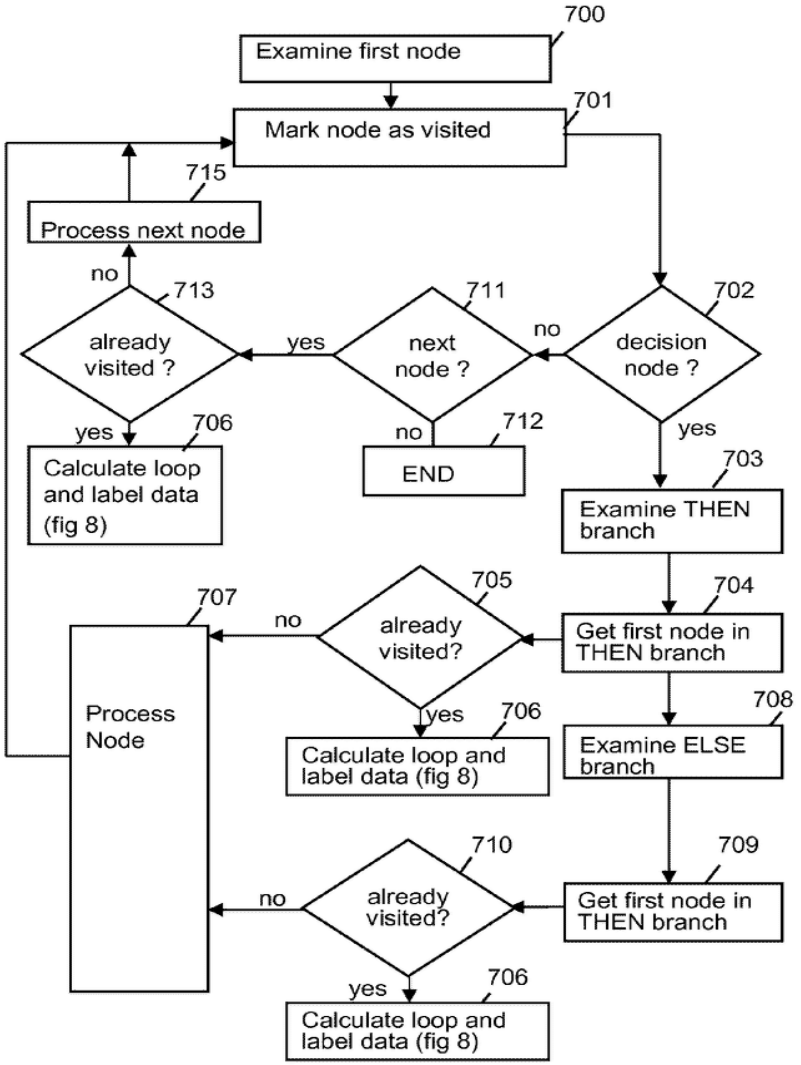
A INDUSTRIOSA
Laboratorio tecnolóxico cidadán

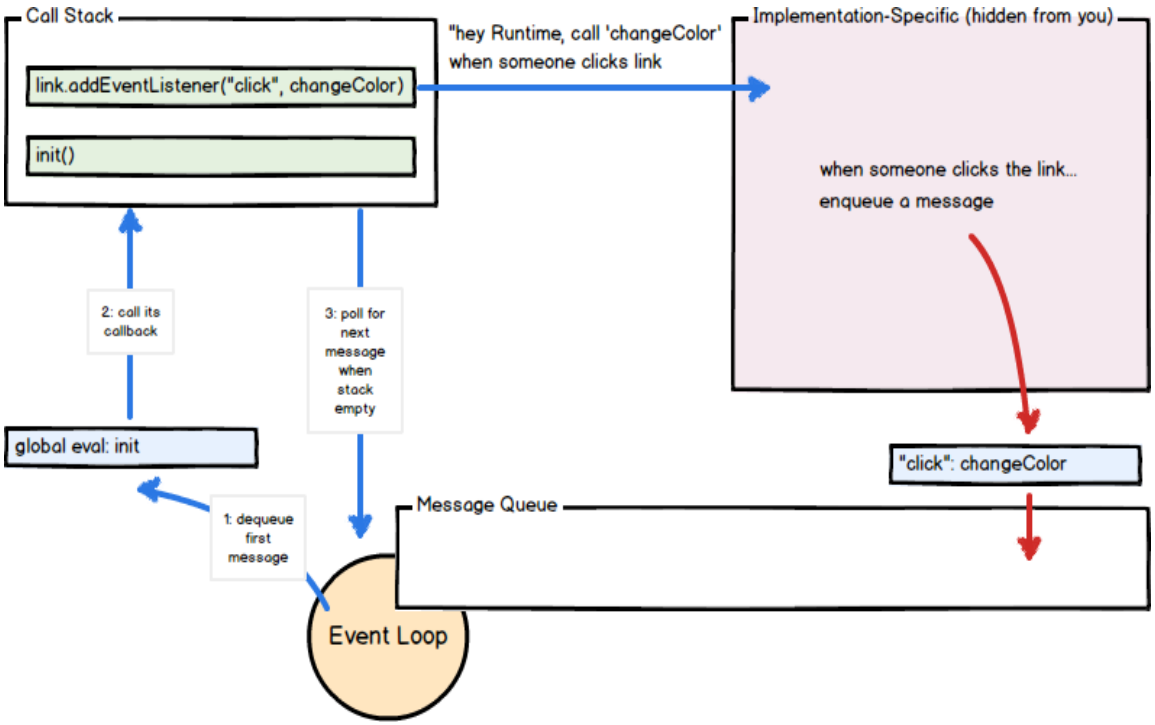
A finales de octubre de 2014, Espressif lanzó un kit de desarrollo de software (SDK) que permitió que el chip fuera programado, eliminando la necesidad de un microcontrolador aparte. Desde entonces, ha habido muchos lanzamientos oficiales de SDK de Espressif; Espressif mantiene dos versiones del SDK - una que se basa en FreeRTOS y la otra basada en callbacks.

Una alternativa al SDK oficial de Espressif es el open-source ESP-Open-SDK que se basa en la cadena de herramientas GCC. ESP8266 utiliza el microcontrolador Cadence Tensilica L106 y la cadena de herramientas GCC es de código abierto y mantenida por Max Filippov.

Otra alternativa es el "Unofficial Development Kit" de Mikhail Grigorev

Scuencial vs Callbacks



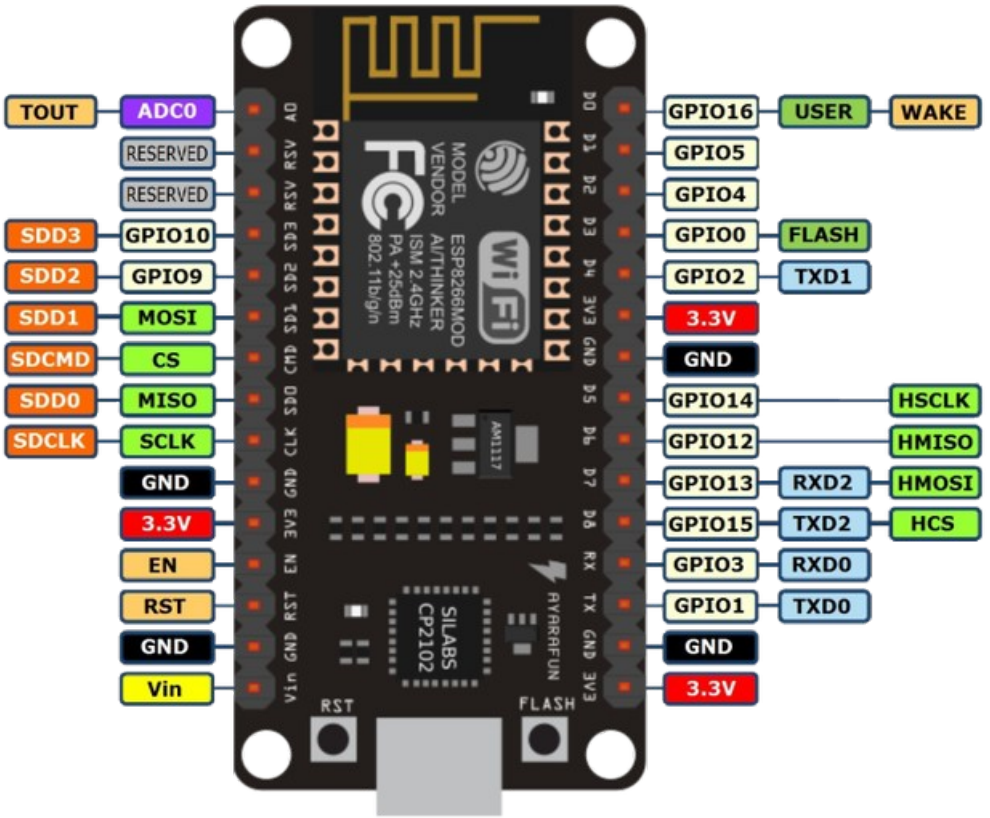


ESPRESSIF

ESP8266

PERIFERICOS

GPIO e Interrupciones



Los números de pin son los de GPIO directamente

- **PinMode (2, INPUT)**
  GPIO0-GPIO15 pueden ser:
  - INPUT,
  - OUTPUT,
  - INPUT_PULLUP
  - INPUT_PULLDOWN

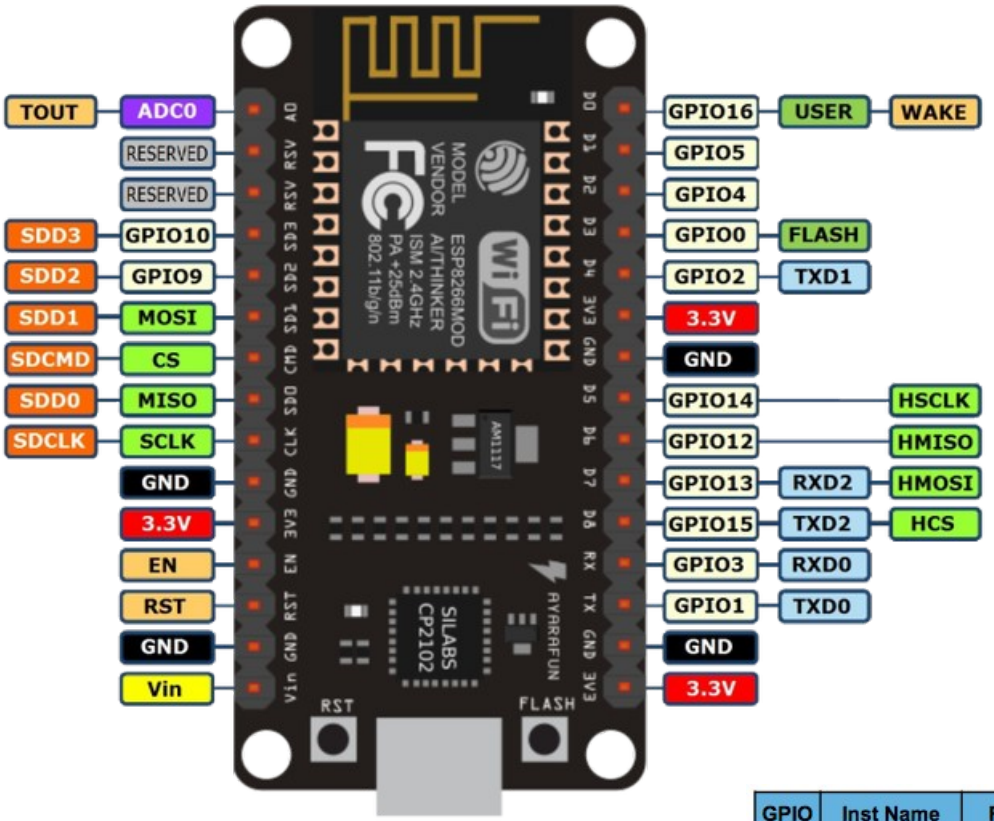
  GPIO16 puede ser:
  - INPUT
  - OUTPUT
  - INPUT_PULLDOWN
- **DigitalRead (2)**
- **DigitalWrite (2 , [HIGH / LOW])**

Interrupciones se pueden asocial a cualquier GPIO pin (INPUT) , excepto GPIO16

attachInterrupt(2, callbackFunction, RISING);
DetachInterrupt (2)

- CHANGE
- RISING
- FALLING

Multiplexado de Pin



Se utilizan a través de la macro **SPECIAL**

pinMode(pin, SPECIAL)

Asignará la función especial mapeada por defecto para ese pin.

Son:

- UART RX/TX en pines 1 - 3
- HSPI pines 12-15
- CLK pines 0, 4 y 5

GPIO	Inst Name	Function 0	Function 1	Function 2	Function 3	Function 4	At Reset	After Reset	Sleep
0	GPIO0_U	GPIO0	SPICS2			CLK_OUT	oe=0, wpu	wpu	oe=0
1	U0TXD_U	U0TXD	SPICS1		GPIO1	CLK_RTC	oe=0, wpu	wpu	oe=0
2	GPIO2_U	GPIO2	I2SO_WS	U1TXD		U0TXD	oe=0, wpu	wpu	oe=0
3	U0RXD_U	U0RXD	I2SO_DATA		GPIO3	CLK_XTAL	oe=0, wpu	wpu	oe=0
4	GPIO4_U	GPIO4	CLK_XTAL				oe=0		oe=0
5	GPIO5_U	GPIO5	CLK_RTC				oe=0		oe=0
6	*SD_CLK_U*	*SD_CLK*	*SPICLK*		*GPIO6*	*U1CTS*	*oe=0*		*oe=0*
7	*SD_DATA0_U*	*SD_DATA0*	*SPIQ*		*GPIO7*	*U1TXD*	*oe=0*		*oe=0*
8	*SD_DATA1_U*	*SD_DATA1*	*SPID*		*GPIO8*	*U1RXD*	*oe=0*		*oe=0*
9	*SD_DATA2_U*	*SD_DATA2*	*SPIHD*		*GPIO9*	*HSPIHD*	*oe=0*		*oe=0*
10	*SD_DATA3_U*	*SD_DATA3*	*SPIWP*		*GPIO10*	*HSPIWP*	*oe=0*		*oe=0*
11	*SD_CMD_U*	*SD_CMD*	*SPICS0*		*GPIO11*	*U1RTS*	*oe=0*		*oe=0*
12	MTDI_U	MTDI	I2SI_DATA	HSPIQ MISO	GPIO12	U0DTR	oe=0, wpu	wpu	oe=0
13	MTCK_U	MTCK	I2SI_BCK	HSPID MOSI	GPIO13	U0CTS	oe=0, wpu	wpu	oe=0
14	MTMS_U	MTMS	I2SI_WS	HSPICLK	GPIO14	U0DSR	oe=0, wpu	wpu	oe=0
15	MTDO_U	MTDO	I2SO_BCK	HSPICS	GPIO15	U0RTS	oe=0, wpu	wpu	oe=0
16	XPD_DCDC	XPD_DCDC	RTC_GPIO0	EXT_WAKEUP	DEEPSLEEP	BT_XTAL_EN	oe=1,wpd	oe=1,wpd	oe=1

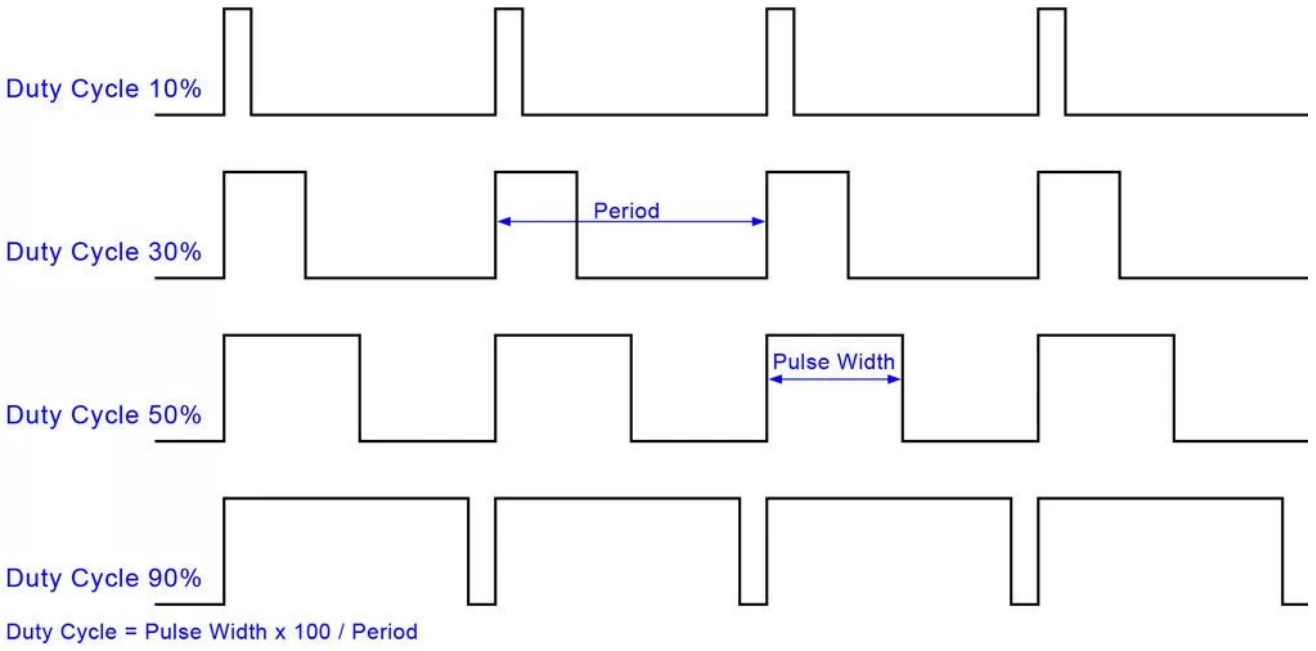
PWM pinMode Output

DUTY MAXIMO 90%

PWM es por Software

- AnalogWriteRange(2) Duty cycle only 0, 50, 90%
- AnalogWrite (2, 1023)
- AnalogWriteFreq([4000 / 5000]) default 1KHz
- analogWrite(pin, 0) to disable PWM If we don’t do it, the next calls to the digitalWrite function on that pin will not work

```
#define  qap      2      //  Quick As Possible ... Duty cycle only 0, 50, 100%
#define  HFreq    5150
#define  pPulse   D2     // a NodeMCU/ESP8266 GPIO PWM pin
analogWriteRange(qap);    analogWriteFreq( HFreq );  analogWrite(pPulse, 1);     // start PWM
```



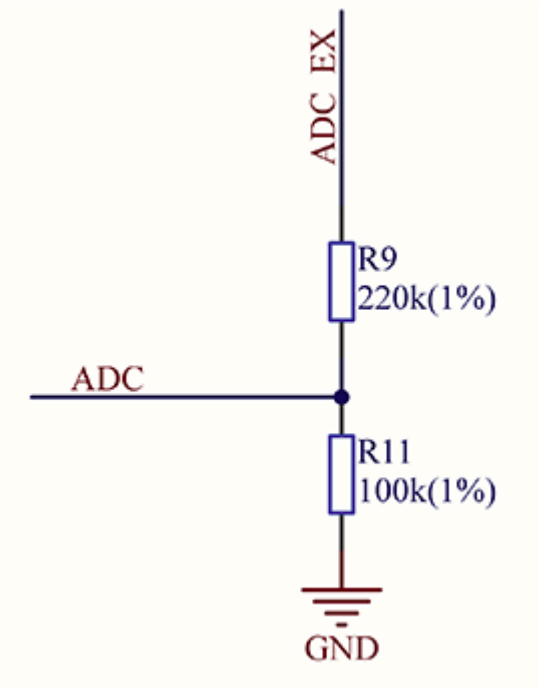
INDUSTRIOSA
Laboratorio tecnolóxico cidadán

ADC

ADC_MODE(ADC_VCC);

Esta línea tiene que aparecer fuera de cualquier función,
por ejemplo inmediatamente después de las líneas #include

analogRead(A0) Lee el valor del canal ADC conectado al pin TOUT.



https://github.com/esp8266/Arduino/issues/338

DELAYS / LOOP /Yield

delay (xx) detiene la ejecución del programa por un número dado de milisegundos **y permite que las tareas WiFi y TCP / IP se ejecuten**

DelayMicrosseconds (xx) hace una pausa para un número dado de microsegundos.
La función delayMicroseconds, por otro lado, no cede a otras tareas, por lo que no se recomienda su uso para retrasos de más de 20 milisegundos.

También existe una función **yield ()** que es equivalente a delay (0).

Recuerde que hay un montón de código que necesita ejecutarse en el chip además de programa cuando WiFi está conectado.

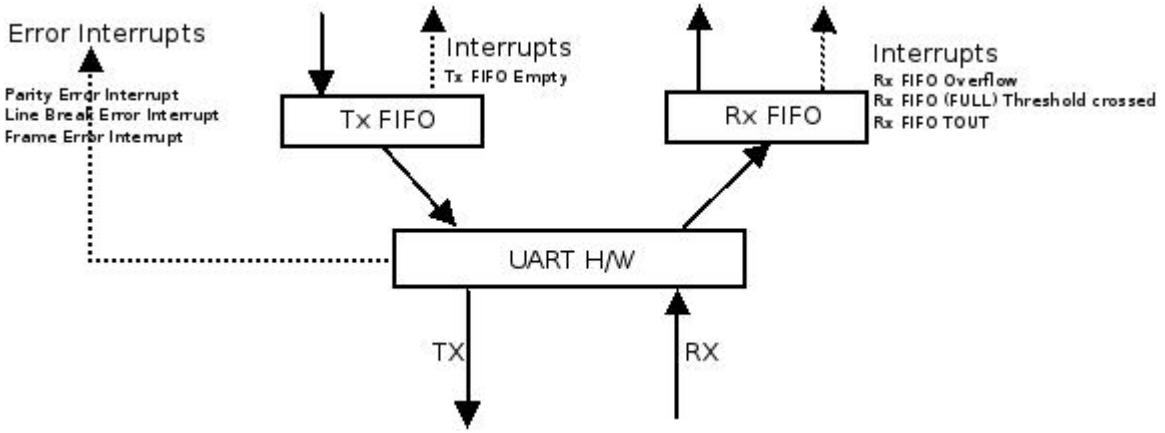
Las bibliotecas WiFi y TCP / IP tienen la oportunidad de manejar cualquier evento pendiente cada vez que se completa la función loop (), o cuando se llama a delay (...).

Si tienes un bucle en algún lugar del programa que tarda mucho tiempo (> 50ms) podrías considerar agregar una llamada a Yield() en cada ciclo del bucle para mantener la pila WiFi funcionando sin problemas.

PUERTO SERIE

- Hardware FIFO (128 bytes for TX and RX)
- HardwareSerial 256-byte TX y RX buffers adicionales

.



UART0 → pines GPIO1 (TX) y GPIO3 (RX).
Puede remapearse a GPIO15 (TX) y GPIO13 (RX) llamando a Serial.swap();

UART1 **TX ONLY** pin GPIO2. → Serial1.begin.

De forma predeterminada, la salida de diagnóstico WiFi está deshabilitada cuando llama a Serial.begin. Para habilitar la salida de diagnóstico → **Serial.setDebugOutput (true)**
Para redirigir la salida de depuración a Serial1 en su lugar, llame a **Serial1**.setDebugOutput (true)

Soportan 5, 6, 7, 8 data bits, odd (O), even (E), and no (N) parity, y 1 or 2 stop bits.
Serial.begin(baudrate, SERIAL_8N1)

INTERRUPCIONES

https://gist.github.com/igrr/5c93c37e8f062e544ed7

TIMER0

Up Counter

```
#include <ESP8266WiFi.h>

volatile unsigned long next;

void inline servoISR(void){

  //timer0_write(next);

}

void setup()
{

  noInterrupts();
  timer0_isr_init();
  timer0_attachInterrupt(servoISR);
  next=ESP.getCycleCount()+ 80000000;
  timer0_write(next);
  interrupts();

}

void loop()
{
  delay(50);

}
```

2 ms tick will crash the ESP6288 if you are using WiFi.

TICKER

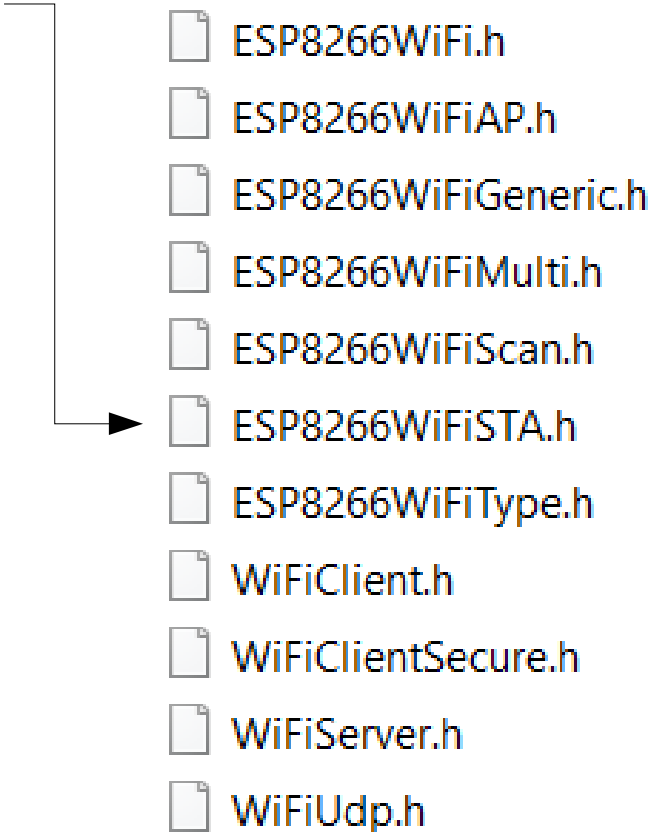
```
#include <Ticker.h>

Ticker flipper;

int count = 0;

void flip()
{
  int state = digitalRead(1);
  digitalWrite(1, !state);

  ++count;
  // when the counter reaches
  if (count == 20)
  {
    flipper.attach(0.1, flip);
  }
  // when the counter reac
  else if (count == 120)
  {
    flipper.detach();
  }
}

void setup() {
  pinMode(1, OUTPUT);
  digitalWrite(1, LOW);

  // flip the pin every 0.3s
  flipper.attach(0.3, flip);
}

void loop() {
}
```

```
#include <Ticker.h>

Ticker tickerSetHigh;
Ticker tickerSetLow;

void setPin(int state) {
  digitalWrite(1, state);
}

void setup() {
  pinMode(1, OUTPUT);
  digitalWrite(1, LOW);

  // every 25 ms, call setPin(0)
  tickerSetLow.attach_ms(25, setPin, 0);

  // every 26 ms, call setPin(1)
  tickerSetHigh.attach_ms(26, setPin, 1);
}

void loop() {
}
```

LIBRERIAS

arduino-1.8.3\hardware\esp8266com\esp8266\libraries

- ESP8266WiFi\src



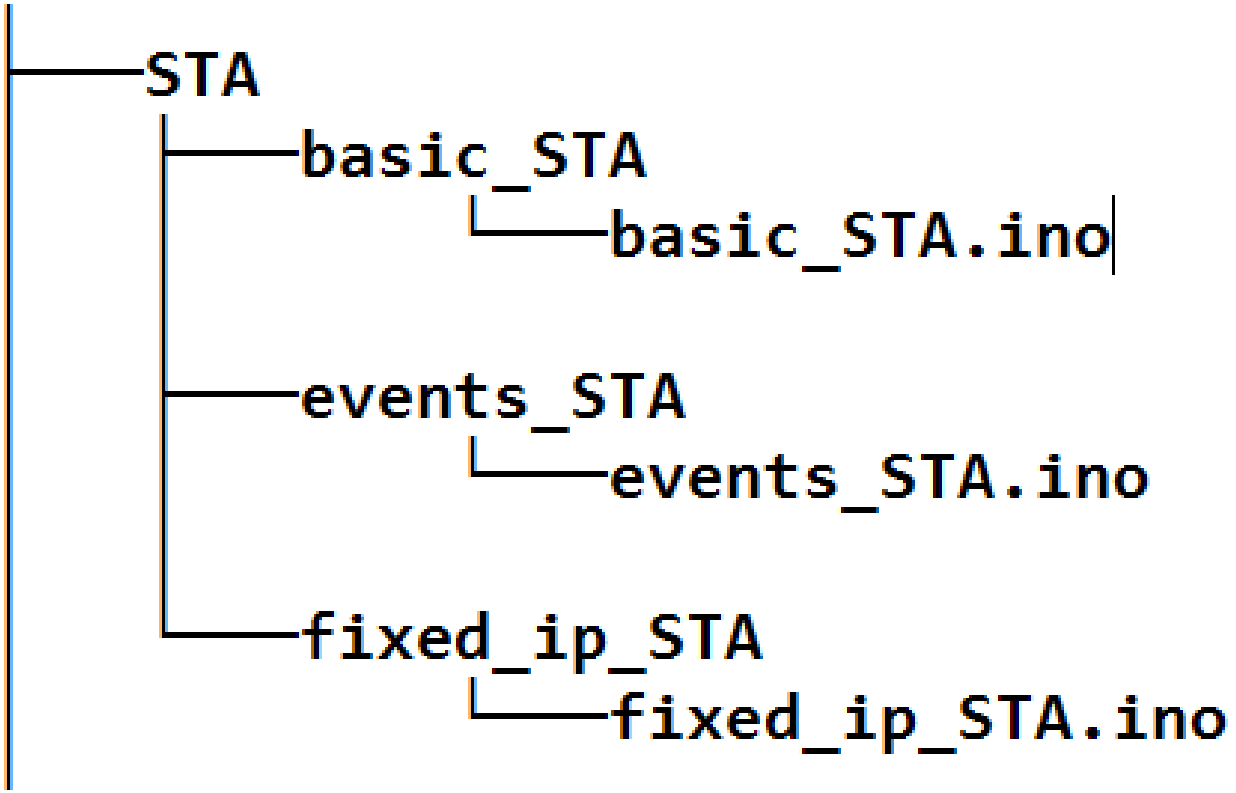
- ESP8266HTTPClient\src
- ESP8266WebServer\src

INDUSTRIOSA
Laboratorio tecnolóxico cidadán

ESPRESSIF

ESP8266

ESP8266 - STATION

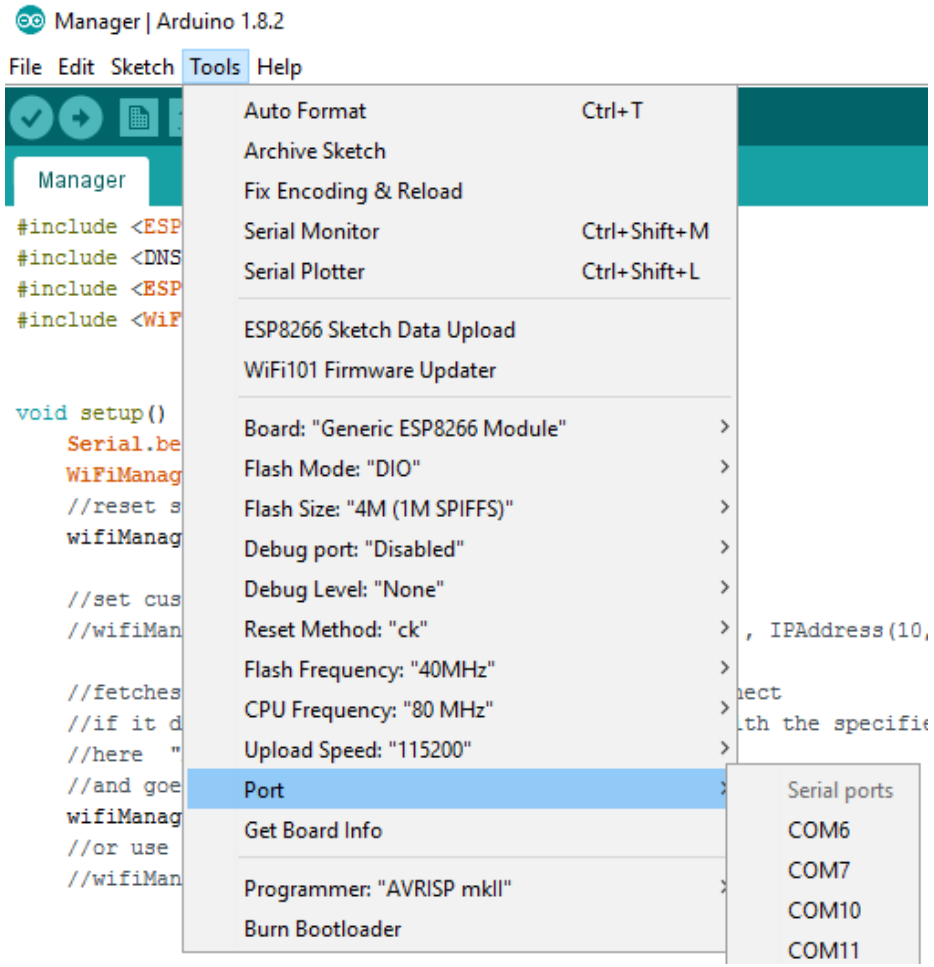


WIFI ST

```ino
#include <ESP8266WiFi.h>

const char* ssid     = "AIndustriosa";
const char* password = "workshop";

uint8_t wifiCNT =0;

void setup() {
  Serial.begin(115200);
  delay(10);

  Serial.print("Connecting to ");

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED && wifiCNT < 20) {
    delay(500);
    Serial.print(".");
    wifiCNT++;
  }
  Serial.println("WiFi connected");

}
void loop() {
  delay(1000);
  Serial.println(WiFi.status());
  Serial.println(WiFi.macAddress());
  Serial.println(WiFi.localIP());
  Serial.println(WiFi.subnetMask());
  Serial.println(WiFi.gatewayIP());
  Serial.println(WiFi.dnsIP(0));
  Serial.println(WiFi.RSSI());

  /*if (WiFi.status() != WL_CONNECTED && wifiCNT < 10) {
    delay(500);
    Serial.print(".");
    wifiCNT++;
  }else{
    Serial.println(WiFi.status());
  }*/
}
```

- **WL_IDLE_STATUS = 0**
- **WL_NO_SSID_AVAIL = 1**
- **WL_SCAN_COMPLETED = 2**
- **WL_CONNECTED = 3**
- **WL_CONNECT_FAILED = 4**
- **WL_CONNECTION_LOST = 5**
- **WL_DISCONNECTED = 6**

FLASH



Dejar pulsados
RESET + FLASH

Cuando compile (Blanco)
SOLTAR RESET
DEJAR FLASH PULSADO 5sec

```
Archiving built core (caching) in: C:\Users\s1\AppData\Local\Temp\arduino_cache_208068\core\core_esp8266com_esp8266_generic_
Sketch uses 332414 bytes (31%) of program storage space. Maximum is 1044464 bytes.
Global variables use 37712 bytes (46%) of dynamic memory, leaving 44208 bytes for local variables. Maximum is 81920 bytes.
Uploading 336560 bytes from C:\Users\s1\AppData\Local\Temp\arduino_build_123903/Manager.ino.bin to flash at 0x00000000
.................................................................................. [ 24% ]
...........
```

WIFI ST EventHandler

```ino
#include <ESP8266WiFi.h>
//ESP8266WIFISTA.h

const char* ssid     = "AIndustriosa";
const char* password = "workshop";

bool wifiConnected =false;

WiFiEventHandler stationConnectedHandler;

void setup() {
  Serial.begin(115200);
  delay(10);

  Serial.print("Connecting to ");

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  stationConnectedHandler = WiFi.onStationModeConnected(&onStationConnected);
}

void onStationConnected(const WiFiEventStationModeConnected& evt) {
  Serial.print("Station connected: ");
  wifiConnected=true;
}

void loop() {
  delay(1000);
  if(wifiConnected){
    Serial.println(WiFi.status());
    Serial.println(WiFi.macAddress());
    Serial.println(WiFi.localIP());
    Serial.println(WiFi.subnetMask());
    Serial.println(WiFi.gatewayIP());
    Serial.println(WiFi.dnsIP(0));
    Serial.println(WiFi.RSSI());
  }

}
```

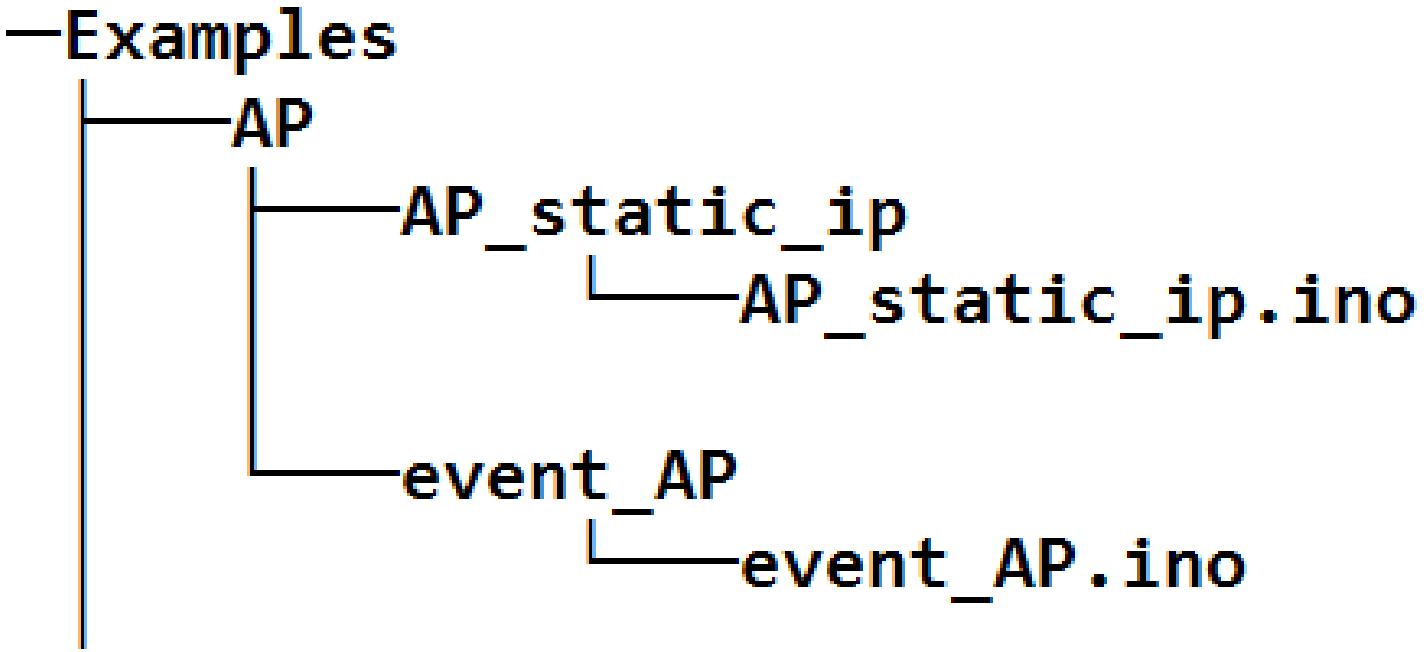
- WiFiEventHandler **onStationModeConnected**
  (std::function<void(const WiFiEventStationModeConnected&)>);
- WiFiEventHandler **onStationModeDisconnected**
  (std::function<void(const WiFiEventStationModeDisconnected&)>);
- WiFiEventHandler **onStationModeAuthModeChanged**
  (std::function<void(const WiFiEventStationModeAuthModeChanged&)>);
- WiFiEventHandler **onStationModeGotIP**
  (std::function<void(const WiFiEventStationModeGotIP&)>);
- WiFiEventHandler **onStationModeDHCPTimeout**
  (std::function<void(void)>);

WIFI ST
FIXED IP

```
#include <ESP8266WiFi.h>

const char* ssid     = "AIndustriosa";
const char* password = "workshop";

uint8_t wifiCNT =0;

IPAddress ip(192, 168, 1, 20);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress dns(8, 8, 8, 8);

void setup() {
  Serial.begin(115200);
  delay(10);

  Serial.print("Connecting to ");

  WiFi.config(ip,gateway, subnet,dns);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED && wifiCNT < 20) {
    delay(500);
    Serial.print(".");
    wifiCNT++;
  }
  Serial.println("WiFi connected");

}
void loop() {
  delay(1000);
  Serial.println(WiFi.status());
```

AINDUSTRIOSA
Laboratorio tecnolóxico cidadán

ESP8266 - AP

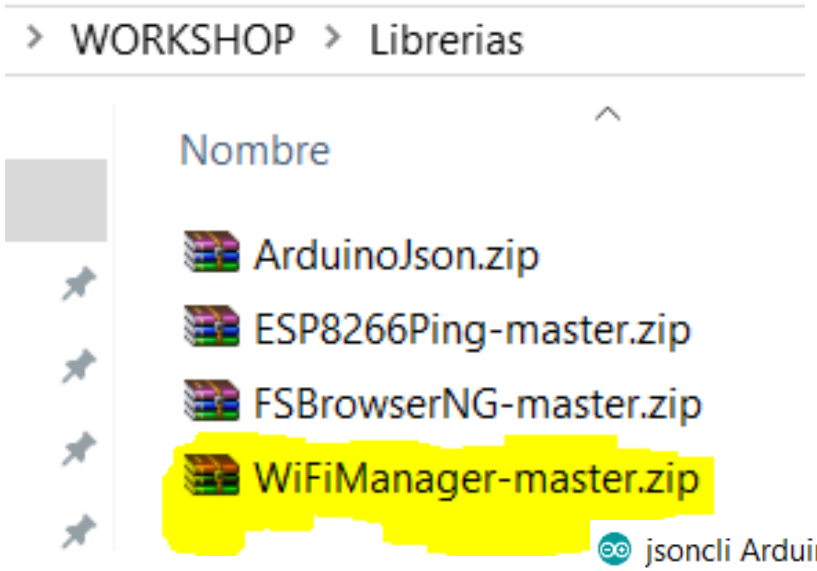

Examples
AP
AP_static_ip
AP_static_ip.ino
event_AP
event_AP.ino

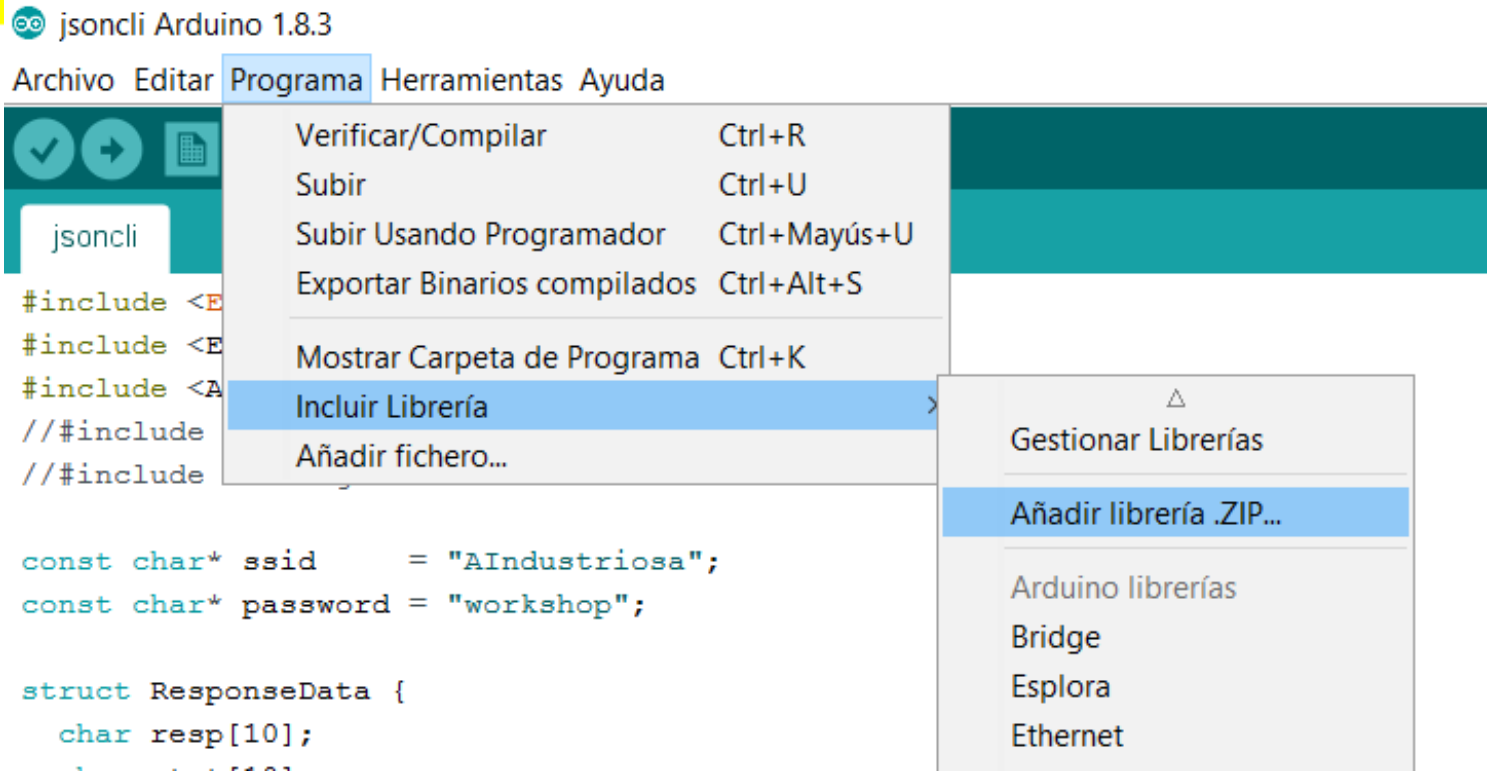
```ino
#include <ESP8266WiFi.h>

const char* ssid     = "MiAP";
const char* password = "workshop";

WiFiEventHandler stationConnectedHandler;
WiFiEventHandler stationDisconnectedHandler;
WiFiEventHandler probeRequestPrintHandler;
WiFiEventHandler probeRequestBlinkHandler;


void setup() {
  Serial.begin(115200);

  // Don't save WiFi configuration in flash - optional
  WiFi.persistent(false);

  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid, password);

  stationConnectedHandler = WiFi.onSoftAPModeStationConnected(&onStationConnected);
  stationDisconnectedHandler = WiFi.onSoftAPModeStationDisconnected(&onStationDisconnected);
  probeRequestPrintHandler = WiFi.onSoftAPModeProbeRequestReceived(&onProbeRequestPrint);

}

void onStationConnected(const WiFiEventSoftAPModeStationConnected& evt) {
  Serial.print("Station connected: ");
  Serial.println(macToString(evt.mac));
}

void onStationDisconnected(const WiFiEventSoftAPModeStationDisconnected& evt) {
  Serial.print("Station disconnected: ");
  Serial.println(macToString(evt.mac));
}

void onProbeRequestPrint(const WiFiEventSoftAPModeProbeRequestReceived& evt) {
  Serial.print("Probe request from: ");
  Serial.print(macToString(evt.mac));
  Serial.print(" RSSI: ");
  Serial.println(evt.rssi);
}
```

- WiFiEventHandler **onSoftAPModeStationConnected**
  (std::function<void(const WiFiEventSoftAPModeStationConnected&)>);
- WiFiEventHandler **onSoftAPModeStationDisconnected**
  (std::function<void(const WiFiEventSoftAPModeStationDisconnected&)>);
- WiFiEventHandler **onSoftAPModeProbeRequestReceived**
  (std::function<void(const WiFiEventSoftAPModeProbeRequestReceived&)>);

WIFI AP EventHandler

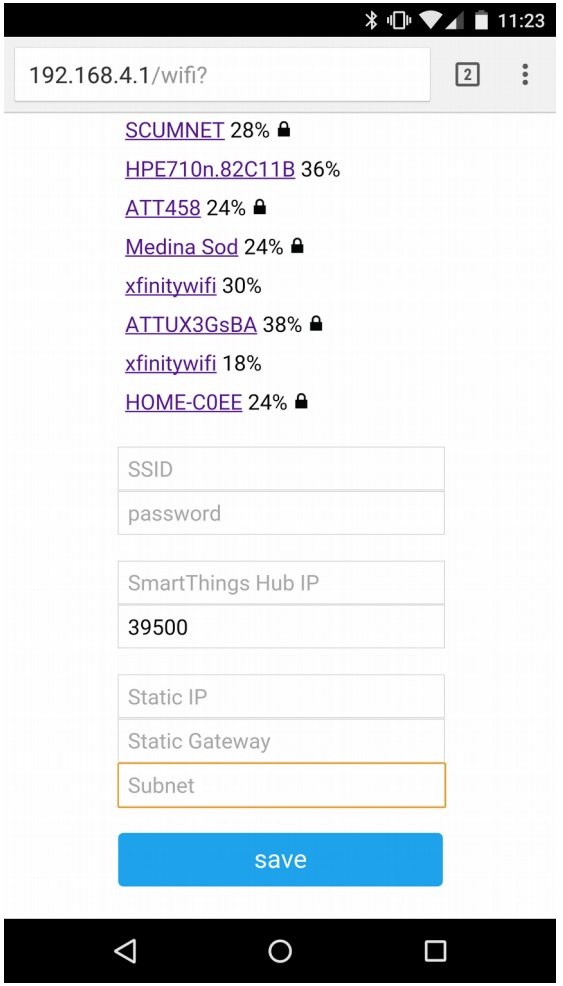
WIFI MANAGER



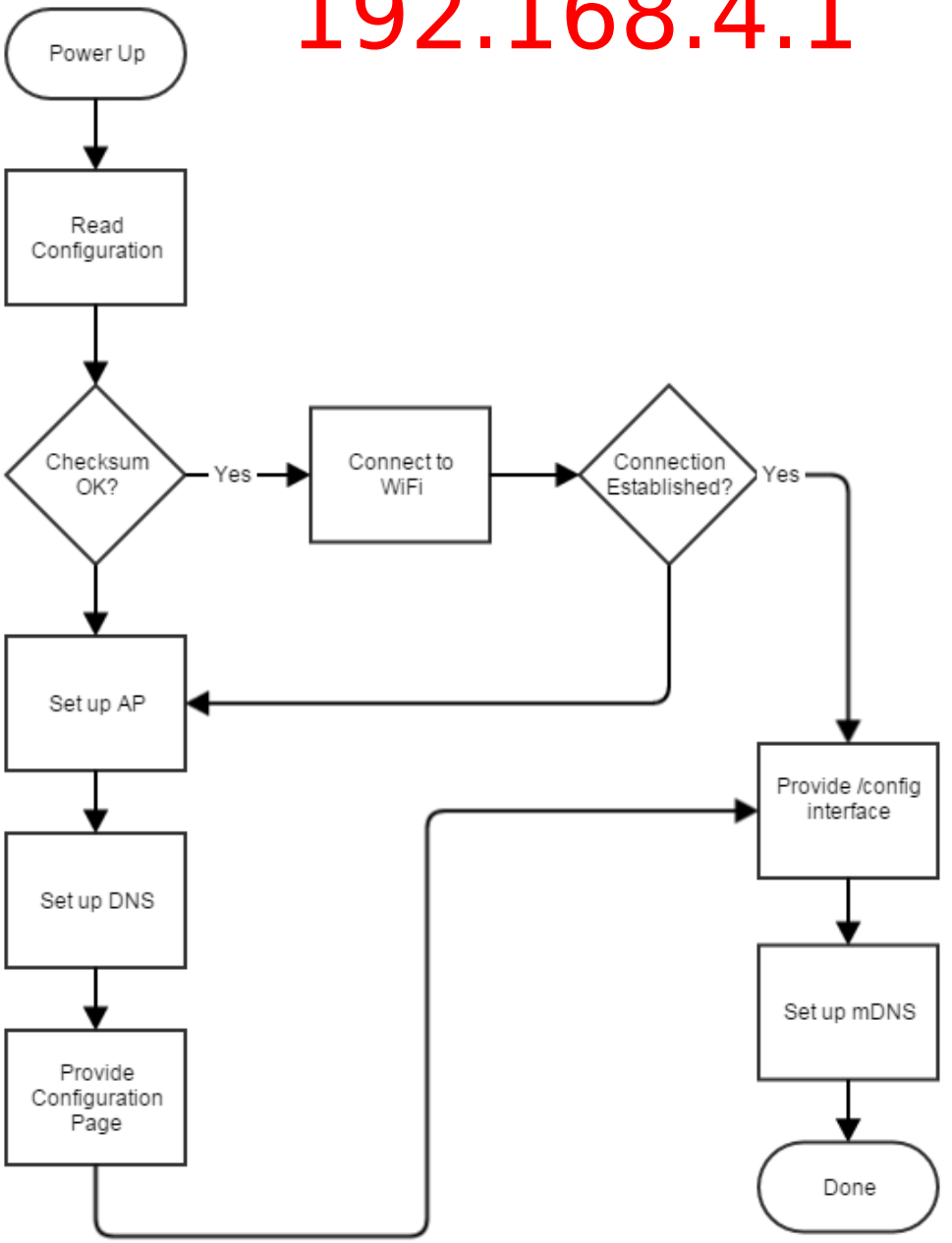


WIFI MANAGER

https://github.com/tzapu/WiFiManager.git



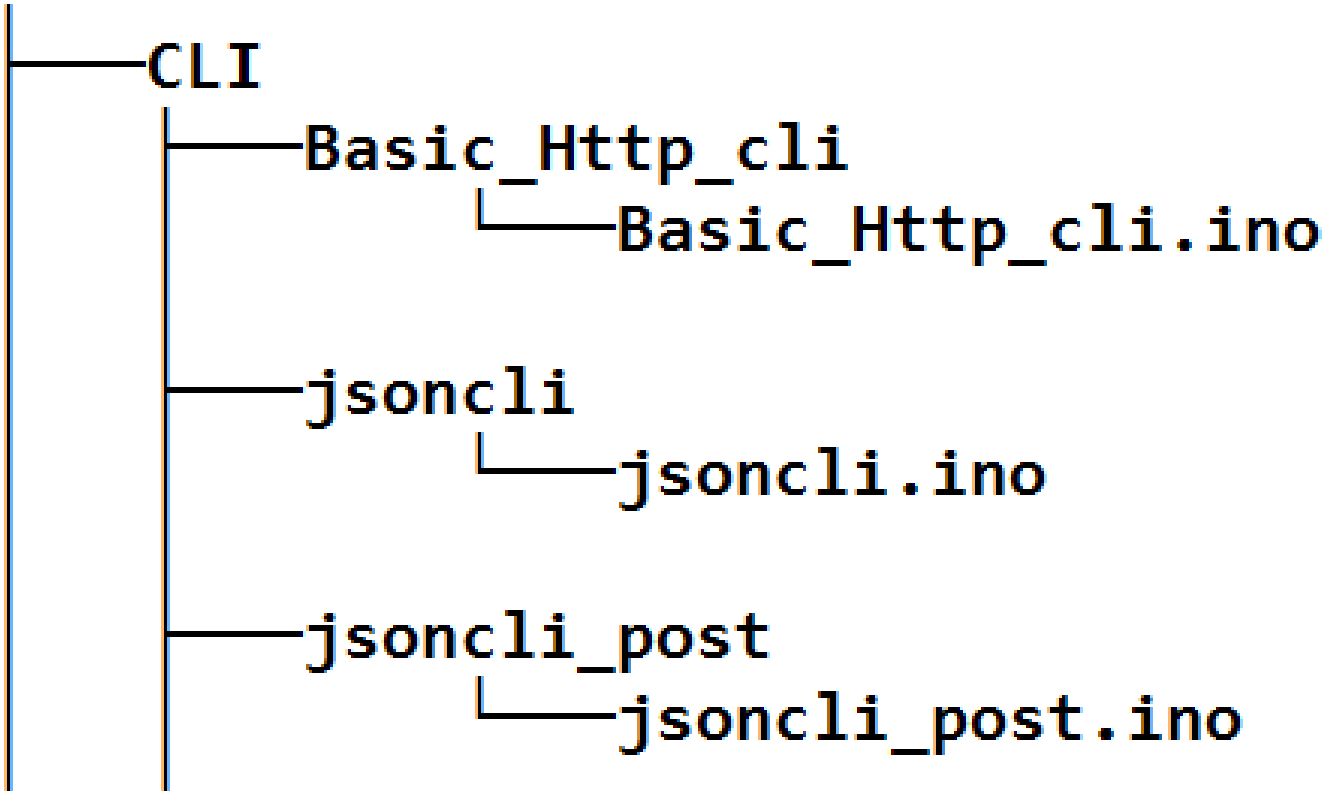
192.168.4.1



WIFI MANAGER

```
#include <ESP8266WiFi.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>

void setup() {
    Serial.begin(115200);
    WiFiManager wifiManager;
    //reset saved settings
    wifiManager.resetSettings();

    //wifiManager.setAPStaticIPConfig(IPAddress(10,0,1,1), IPAddress(10,0,1,1), IPAddress(255,255,255,0));

    //and goes into a blocking loop awaiting configuration
    wifiManager.autoConnect("AMyAP", "workshop");

    Serial.println("connected... :)");
}

void loop() {
    delay(100),
    Serial.println("OK");

}
```

ESP8266 – HTTP CLI



```
void loop() {

    if((WiFi.status() == WL_CONNECTED)) {

        HTTPClient http;

        Serial.print("[HTTP] begin...\n");
        http.begin("http://192.168.1.47/"); //HTTP
        http.addHeader("Content-Type", "application/json");

        Serial.print("[HTTP] GET...\n");
        int httpCode = http.GET();

        // httpCode will be negative on error
        if(httpCode > 0) {
            // HTTP header has been send and Server response header has been handled
            Serial.printf("[HTTP] GET... code: %d\n", httpCode);

            // file found at server
            if(httpCode == HTTP_CODE_OK) {
                String payload = http.getString();
                Serial.println(payload);
            }
        } else {
            Serial.printf("[HTTP] GET... code: %d\n", httpCode);
            Serial.println();
            Serial.printf("[HTTP] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());
        }

        http.end();
    }

    delay(10000);
}
```

Http Error Code

CONNECTION_REFUSED	(-1)
SEND_HEADER_FAILED	(-2)
SEND_PAYLOAD_FAILED	(-3)
NOT_CONNECTED	(-4)
CONNECTION_LOST	(-5)
NO_STREAM	(-6)
NO_HTTP_SERVER	(-7)
TOO_LESS_RAM	(-8)
ENCODING	(-9)
STREAM_WRITE	(-10)
READ_TIMEOUT	(-11)

```
CODE_OK = 200,
CREATED = 201,
ACCEPTED = 202,
…

BAD_REQUEST = 400,
UNAUTHORIZED = 401,
…..
```

ESP8266WebServer

FSBrowser

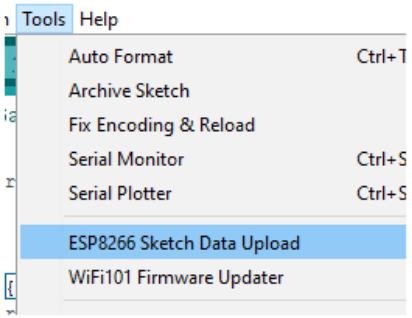
FSBrowserNG

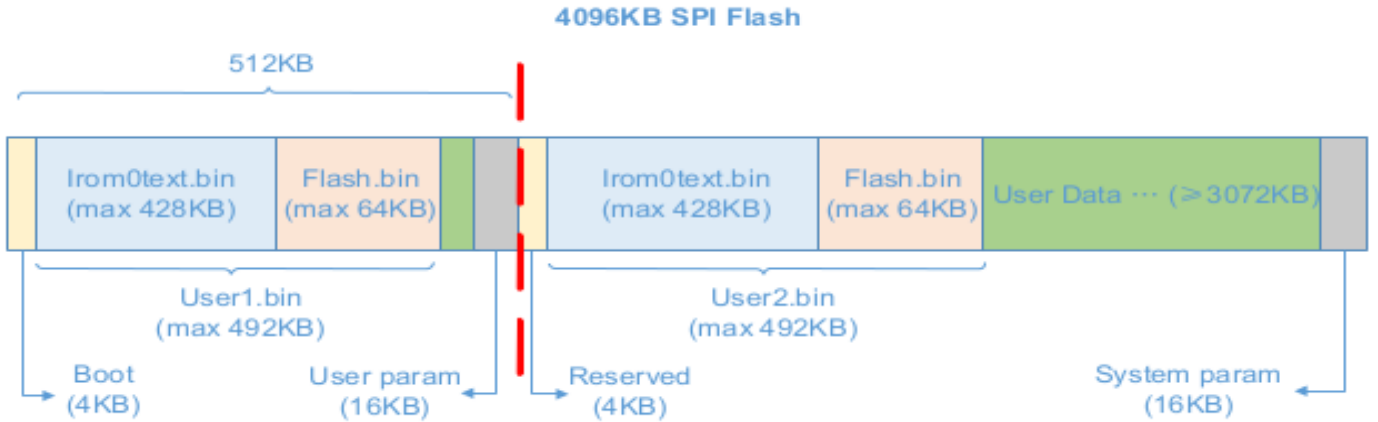
https://github.com/gmag11/FSBrowserNG.git

SPIFFS data uploader
https://github.com/esp8266/arduino-esp8266fs-plugin/releases/download/0.2.0/ESP8266FS-0.2.0.zip

Bajar version v2 ..... la v3 da problemas

Descomprimir Tols/ESP8266FS-0.2.0.zip en:
C:\Program Files (x86)\Arduino\tools → Reiniciar Arduino





Board	Flash chip size, bytes	File system size, bytes
Generic module	512k	64k
Generic module	1M	64k, 128k, 256k, 512k
Generic module	2M	1M
Generic module	4M	3M
Adafruit HUZZAH	4M	1M, 3M
NodeMCU 0.9	4M	1M, 3M
NodeMCU 1.0	4M	1M, 3M

PRAGMA

PSTR(x)==> static const char __c[] PROGMEM =xxx

FPSTR(pstr_pointer) ==> const __FlashStringHelper * (pstr_pointer)
Hace un cast de PSTR al tipo __FlashStringHelper

F(string_literal) ==> (FPSTR(PSTR(string_literal)))