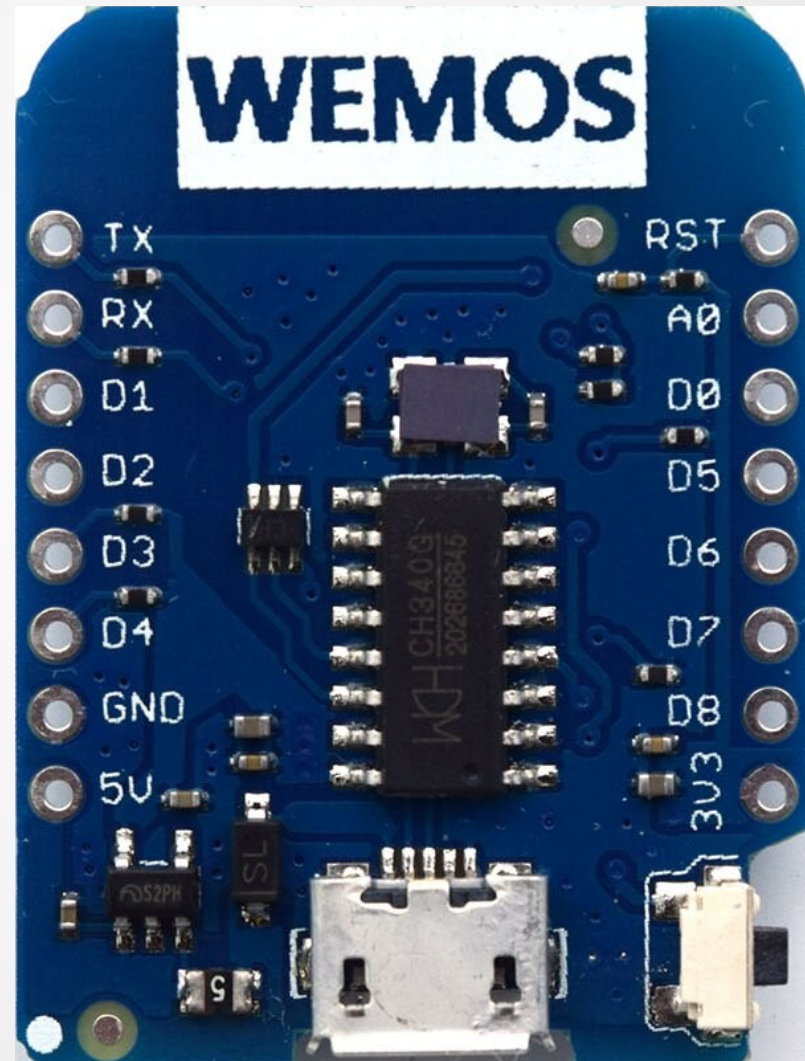
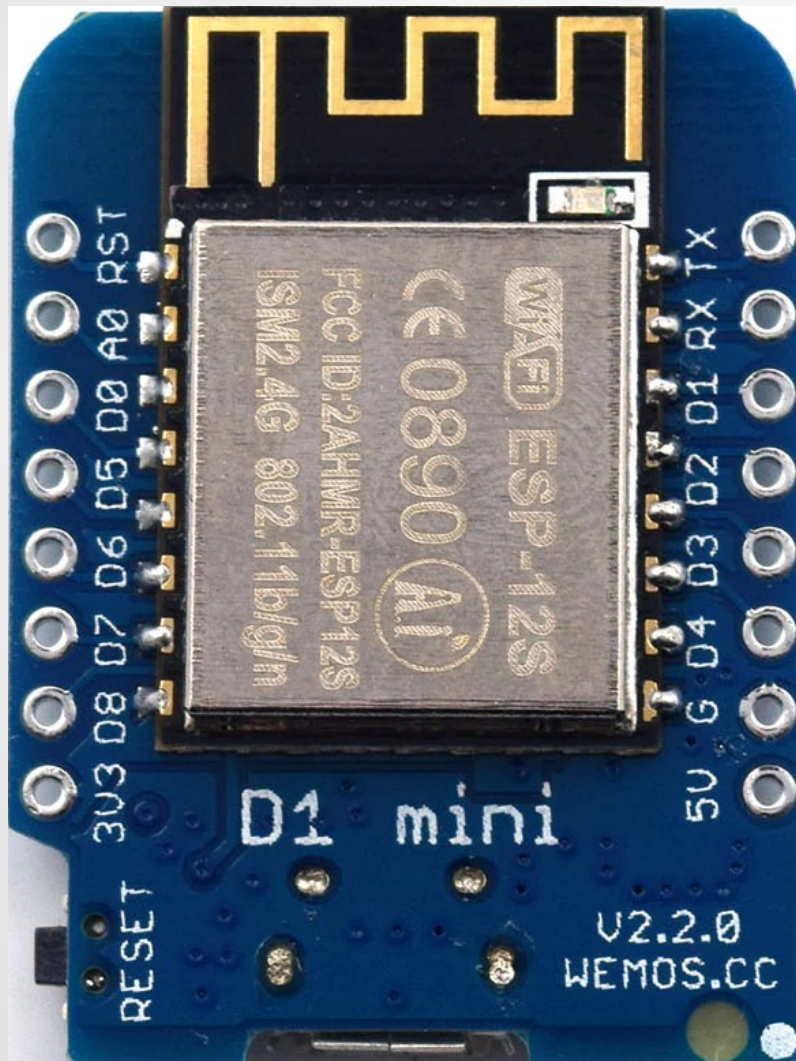
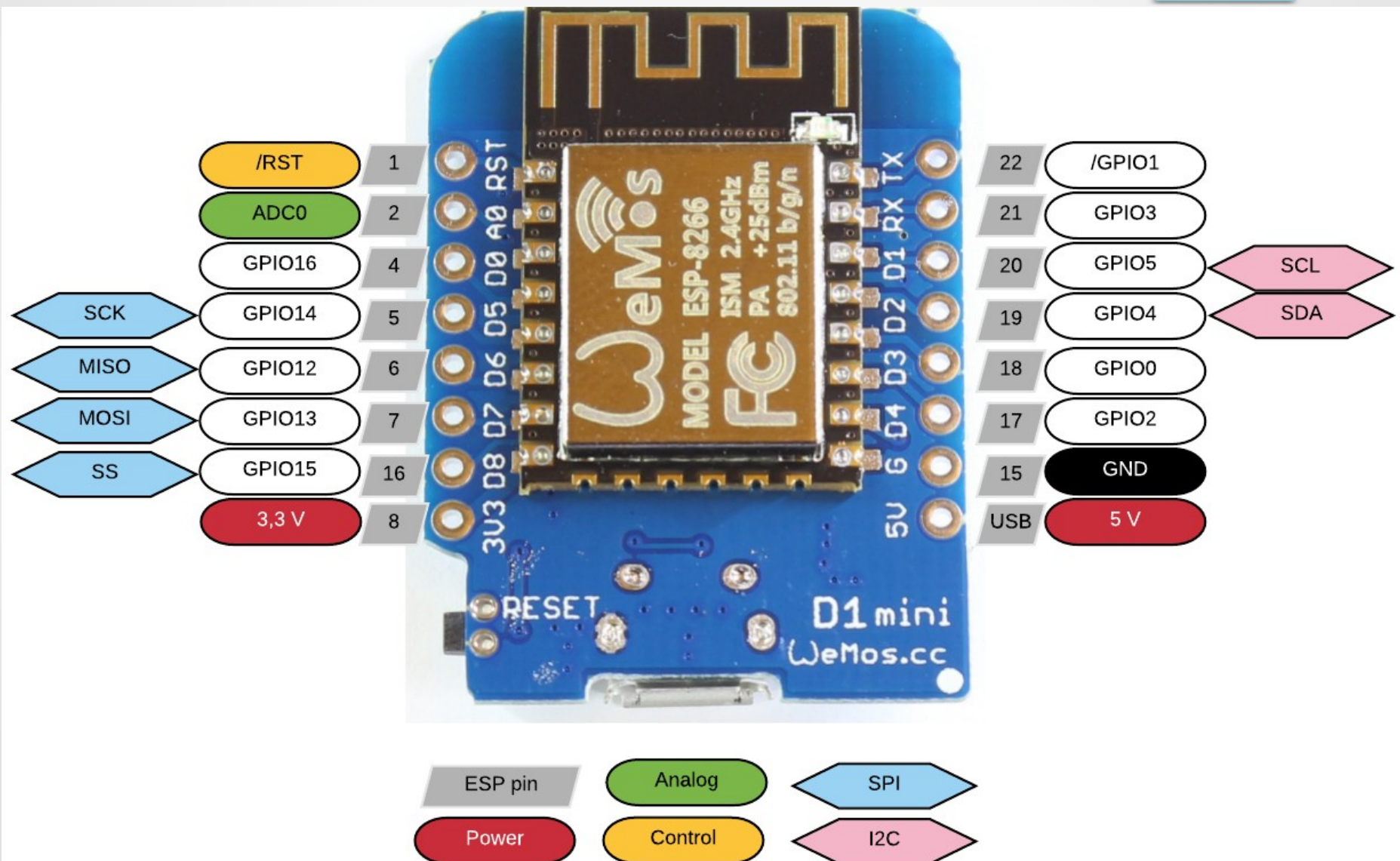


ESP8266 – Conexión a WiFi existente

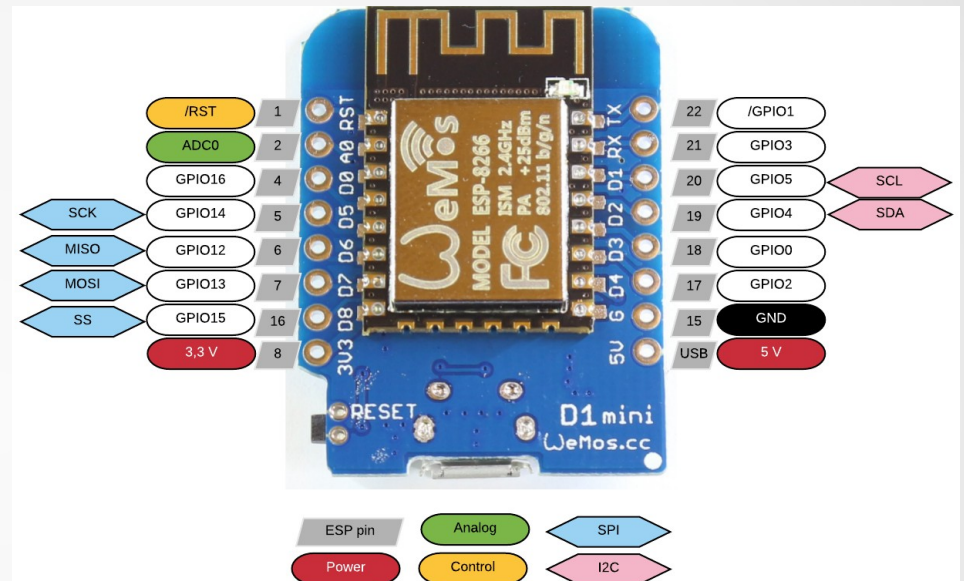


Pinout Wemos D1 mini



Pinout Wemos D1 mini

Pin	ESP-8266 Pin	Función
TX	TXD	TXD
RX	RXD	RXD
A0	A0	Analog input (max 3.2V)
D0	GPIO16	IO
D1	GPIO5	IO, PWM, Interrupt, I2C, SCL
D2	GPIO4	IO, PWM, Interrupt, I2C, SDA
D3	GPIO0	IO 10k Pull-up, PWM, Interrupt, I2C
D4	GPIO2	IO 10k Pull-up, PWM, Interrupt, I2C, BUILTIN_LED
D5	GPIO14	IO, PWM, Interrupt, I2C, SCK
D6	GPIO12	IO, PWM, Interrupt, I2C,, MISO
D7	GPIO13	IO, PWM, Interrupt, I2C,, MOSI
D8	GPIO15	IO 10k Pull-down, PWM, Interrupt, I2C,, SS
G	GND	Ground
5V	-	5V
3V3	3.3V	3.3V
RST	RST	Reset



ESP8266 – Conexión a WiFi existente

- A placa ESP8266 inclúe un módulo WiFi que nos permite comunicalo con outros elementos da rede.
- Existen diversas librerías que nos permiten usar con certa facilidade as funcións de rede do módulo WiFi. Nós empregaremos a librería:

<ESP8266WiFi.h>

- Podemos configurar a placa como:

WIFI_STA	Modo estación, conéctase a unha rede existente previamente configurada
WIFI_AP	Modo Access Point, xenera unha WiFi á que se poden conectar outros dispositivos
WIFI_AP_STA	Combinación de modo AP e STA
WIFI_OFF	Apagada



ESP8266 – Conexión a WiFi existente

- A librería ESP8266WiFi forma parte da definición de placas que descargamos co xestor de placas.
- Para usala, temos que empregar o obxecto WiFi, que representará un interface de rede. Á súa vez proporcionáanos diversas funcións (métodos), das que imos empregar polo de agora:
 - WiFi.mode(WIFI_STA) - configuramos o dispositivo para conectarse a unha WiFi preexistente
 - WiFi.disconnect() - desconectamos da rede actual
 - WiFi.begin(SSID, PASS) – conectamos a unha rede co nome SSID e chave PASS
 - WiFi.localIP() - devolve a IP configurada unha vez conectado á rede
 - WiFi.Status() - obtemos en que estado está a conexión á rede. Os estados posibles aparecen na diapositiva seguinte



ESP8266 – Conexión a WiFi existente

- Estados posibles da conexión de rede:

WL_IDLE_STATUS	A WiFi está mudando entre estados
WL_NO_SSID_AVAIL	Non se encontra rede co identificador SSID
WL_CONNECT	Conexión realizada con éxito
WL_CONNECT_FAILED	Chave de rede incorrecta
WL_DISCONNECTED	O ESP8266 non está configurado en modo STA



ESP8266 – Conexión a WiFi existente

- Imos empezar por conectar o Wemos D1 a unha WiFi creada para a ocasión nun móbil.
- O script que executará o Wemos tentará conectarse á rede (dispoñendo do nome e chave) durante un número determinado de intentos. Se ao finalizar o número de intentos non ten éxito, comunicará o feito polo porto serie. En caso de conectarse con éxito, mostrará polo porto serie a IP coa que se conecta á rede.
- Para aproveitar o código do anterior script, imos facer que o LED escintile rápido cando hai erro de rede e que escintile máis lento cando hai conexión de rede.



ESP8266 – Conexión a WiFi existente

```
conexion.wifi.sta.esp8266 | Arduino 1.8.12

conexion.wifi.sta.esp8266 $
19 #include <ESP8266WiFi.h>
20
21 // WiFi móvil
22 #define MAX_INTENTOS 50
23 #define WIFI_SSID "ssid"
24 #define WIFI_PASS "pass"
25
26 // Pins datos
27 // GPIO14: D5
28 #define LED 14
29
30 int tempo = 500;
31 bool conectado = false;
32
33 void setup() {
34   Serial.begin(9600);
35   pinMode(LED, OUTPUT);
36   conectarWiFi();
37 }
38
39 // Función que se encarga de xestionar a conexión á rede
40 void conectarWiFi() {
41   WiFi.mode(WIFI_STA); // Indica modo Station (conexión a outro dispositivo)
42   WiFi.disconnect(); // Desconecta unha posible conexión previa
43   WiFi.begin(WIFI_SSID, WIFI_PASS); // Inicia a conexión
44   Serial.print("\n\nAgardando pola WiFi ");
45   int contador = 0; // Comproba estado da conexión e fai varias tentativas
46   while(WiFi.status() != WL_CONNECTED and contador < MAX_INTENTOS) {
47     contador++;
48     delay(500);
49     Serial.print(".");
50   }
51   Serial.println("");
52   // Informa do estado da conexión e IP en caso de éxito
53   if(contador < MAX_INTENTOS) {
54     Serial.print("Conectado á WiFi coa IP: "); Serial.println(WiFi.localIP());
55   }
56   else {
57     Serial.println("Non se puido conectar á WiFi");
58   }
59 }
```

Guardado con Sucesso.

Leaving...

Hard resetting via RTS pin...

24 Generic ESP8266 Module em /dev/cu.wchusbserial1420

```
/dev/cu.wchusbser

sld??|?l?| ? l? c|?? ? ?r?c?HLX?KLKx? ?PHt??

Agardando pola WiFi .....
Non se puido conectar á WiFi

{d|??|?d?| ? l? c|?? ? ?{?c? b?nn?lgg??? c x??l{d{dx?o? ?d0I?@h?E?? hE?

Agardando pola WiFi .....
Conectado á WiFi coa IP: 192.168.43.48

☒ Avanzo automático de linha ☐ Mostrar marca de tempo
```

- O script completo está na carpeta correspondente dos scripts do repositorio, en Github.



ESP8266 – Conexión a WiFi existente

- Imos mellorar un pouco o script anterior, no sentido de crear un arquivo de configuración 'config.h', no que imos gardar o nome de rede e a chave, para que non sexa público nos repositorios e máis adiante outras variables necesarias para o noso script.
- A función conectarWiFi() tamén a imos pasar a outro arquivo 'toolsWiFi.hpp', que actuará como unha librería creada por nós e onde gardaremos as funcións relacionadas coa rede, para dispor dun código principal máis limpo.
- Para crear estes arquivos, podemos facelo polo método habitual ou creando arquivos de texto plano, ou mellor aínda premendo do triángulo co vértice cara abaixo do IDE de Arduíno. Aparece un menú contextual, no que podemos crear un novo 'separador', que non é máis que un novo arquivo de texto (coa extensión que nós queiramos) dentro da carpeta do noso proxecto.
- É interesante ademais que estes separadores van seguir unidos á xanela aínda que pechemos o IDE, formando parte do noso código.

ESP8266 – Conexión a WiFi existente



```
7 * serie e fará escintilar
8 * non conseguir conectarse
9 * de veces, superados est
10 * fai escintilar o LED co
11 *
12 * Non fai control de posi
13 * delay() pode ocasionar
14 * imposibilidade de execu
15 * a espera. Haberá que te
16 *
17 */
18
19 #include <ESP8266WiFi.h>
20 #include "config.h" // Arquivo local para configuración de rede
21 #include "toolsWiFi.hpp" // Biblioteca local para funcións de rede
22
23 // Pins datos
24 // GPIO14: D5
25 #define LED 14
26
27 int tempo = 500;
28 bool conectado = false;
29
30 void setup() {
31   Serial.begin(9600);
32   pinMode(LED, OUTPUT);
33   conectarWiFi();
34 }
35
36 void loop() {
37   if(conectado) escintila(tempo);
38   else escintila(tempo/10);
39 }
40
41 // Función que encende e apaga o LED cunha
42 // frecuencia determianda pola variable 'espera'
43 void escintila(int espera) {
44   digitalWrite(LED, HIGH);
45   delay(espera);
46   digitalWrite(LED, LOW);
47   delay(espera);
48 }
```

Carregamento completo
Leaving...
Hard resetting via RTS pin...

20 Generic ESP8266 Module em /dev/cu.wchusbserial1420



```
1 /*
2 * Biblioteca local para gardar as nosas funcións e
3 * utilidades de rede.
4 * |
5 */
6
7
8 // Función que se encarga de xestionar a conexión á rede
9 void conectarWiFi() {
10   WiFi.mode(WIFI_STA); // Indica modo Station (conexión a outro di
11   WiFi.disconnect(); // Desconecta unha posible conexión previa
12   WiFi.begin(WIFI_SSID, WIFI_PASS); // Inicia a conexión
13   Serial.print("\n\nAgardando pola WiFi ");
14   int contador = 0; // Comproba estado da conexión e fai varias te
15   while(WiFi.status() != WL_CONNECTED and contador < MAX_INTENTOS)
16     contador++;
17     delay(500);
18     Serial.print(".");
19   }
20   Serial.println("");
21   // Informa do estado da conexión e IP en caso de éxito
22   if(contador < MAX_INTENTOS) {
23     Serial.print("Conectado á WiFi coa IP: "); Serial.println(WiFi.
24   }
```



```
1 /*
2 * Arquivo para almacenar a configuración da conexión
3 * á rede WiFi.
4 *
5 */
6
7
8 // WiFi móbil
9 #define MAX_INTENTOS 50
10 #define WIFI_SSID "ssid"
11 #define WIFI_PASS "pass"
```

- O código principal queda máis claro e compacto.
- O código secundario queda nos outros dous 'separadores'.
- Impórtase o código secundario co nome do arquivo entre comillas dobles (" e "), para indicar que son locais.



ESP8266 – Conexión a WiFi existente

- Outras funcións que nos poden ser útiles en proxectos que impliquen WiFi co ESP8266 son:
 - `WiFi.reconnect()` - Tenta a reconexión cando se cancela por sinal baixo ou tempo de inactividade
 - `WiFi.disconnect(true)` – Desconecta a rede cando sexa necesario aforrar batería
 - `WiFi.isConnected()` - Para empregar en esquemas condicionais (if-else, switch-case)
 - `WiFi.setAutoConnect(autoConnect)` – Tena a reconexión sempre que se desconecte
 - `WiFi.status()` - Informa do estado de conexión da red
- Se precisamos coñecer datos da nosa conexión: nome de rede (en caso de multirrede), IP, porta de enlace, etc, as funcións que nos dan esa información son:
 - `WiFi.SSID()`
 - `WiFi.hostname()`
 - `WiFi.localIP()`
 - `WiFi.subnetMask()`
 - `WiFi.gatewayIP()`
 - `WiFi.dnsIP(dns_no)`
 - `WiFi.macAddress()`



ESP8266 – Conexión a WiFi existente

- Nesta unidade aprendemos a:
 - recoñecer os modos de rede (STA ou AP)
 - recoñecer os posibles estados da conexión
 - conectarnos a unha rede existente e consultar parámetros da mesma
 - mellorar o código separándoo en librerías definidas polo usuario (locais), ou eliminar información sensible trasladándoa a arquivos de configuración