Como usar esp32

1. **Conectarlo a una red wi fi**: Debemos asignarle el modo station al esp32. Este se asigna de la siguiente manera: WiFi.mode**(**WIFI\_STA**)**;

El modo station perimite conectar el módulo a una red wi fi. Debemos asignarle un nombre de red y su contraseña para conectarnos. Simplemente usamos el método 'Wifi.begin(…)', que tiene la siguiente definición:

WiFi.begin**(**ssid, password = null, channel = 0, bssid = null, connect = true**)**

Siendo:

* **ssid**, nombre de la red WiFi a la que nos queremos conectar (max 32 char)
* **password**, opcional, contraseña (mínimo 8 char y máximo 64)
* **channel**, opcional, el canal de WiFi a emplear
* **bssid**, opcional, dirección MAC del punto de acceso
* **connect**, indica si queremos conectar inmediatamente (si es false, sólo guarda los parámetros)

Las funciones WiFi.begin(...) devuelven el estado del ESP32 para determinar cuándo hemos establecido la conexión con la red. También podemos obtenerlo en otro momento a lo largo de nuestro programa con la función WiFi.status(). Esta variable de estado del ESP32 puede ser:

|  |  |
| --- | --- |
| **WL\_CONNECTED** | **Conexión establecida** |
| WL\_NO\_SSID\_AVAIL | El SSID no se encuentra |
| WL\_CONNECT\_FAILED | Contraseña incorrecta |
| WL\_IDLE\_STATUS | Wi-Fi está cambiando entre estados |
| WL\_DISCONNECTED | El ESP8266 no está configurado en modo STA |

1. While para esperar conexión

Como vimos, el status nos devuelve la situación de nuestra conexión. En un while preguntaremos si el status es distinto a WL\_CONNECTED. Mientras lo sea, entrará en un bucle infinito que aguardará a que el status sea WL\_CONNECTED. El código es el siguiente:

while **(**WiFi.status**()** != WL\_CONNECTED**)**

**{**

delay**(**100**)**;

Serial.print**(**'.'**)**;

**}**

El código final nos queda así:

#include <Wifi.h>

const char\* ssid = "ssid";

const char\* password = "password";

void ConnectWiFi\_STA**()**

**{**

Serial.println**(**""**)**;

WiFi.mode**(**WIFI\_STA**)**;

WiFi.begin**(**ssid, password**)**;

while **(**WiFi.status**()** != WL\_CONNECTED**)**

**{**

delay**(**100**)**;

Serial.print**(**'.'**)**;

**}**

Serial.println**(**""**)**;

Serial.print**(**"Iniciado STA:\t"**)**;

Serial.println**(**ssid**)**;

Serial.print**(**"IP address:\t"**)**;

Serial.println**(**WiFi.localIP**())**;

**}**

void setup**()**

**{**

Serial.begin**(**115200**)**;

ConnectWiFi\_STA**()**;

**}**

void loop**()**

**{**

**}**

1. IP Estática

El siguiente ejemplo se conectaría a la red, y fijaría la IP a 192.168.1.200:

const char \*ssid = "SSID";

const char \*password = "PASSWORD";

IPAddress ip**(**192,168,1,200**)**;

IPAddress gateway**(**192,168,1,1**)**;

IPAddress subnet**(**255,255,255,0**)**;

void setup**()**

**{**

Serial.begin**(**115200**)**;

delay**(**10**)**;

Serial.println**()**;

WiFi.mode**(**WIFI\_STA**)**;

WiFi.config**(**ip, gateway, subnet**)**;

WiFi.begin**(**ssid, password**)**;

Hay que tener en cuenta que el router tiene que estar configurado para asignarnos la IP deseada. Típicamente, esto significa que debe estar fuera del rango de IP designadas para DHCP.

ESP32 como servidor y con contenido JSON:

1. Incluir la librería WebServer.h. De ella usaremos las siguientes funciones:
   1. WebServer server**(**80**)**; Genera un objeto de la clase WebServer, y recibe como parámetro el puerto. El 80 referencia al Protocolo de transferencia de hipertexto (HTTP)
   2. server.begin**()**;Inicializa el servidor.
   3. server.handleClient**()**; Se llama en el void.loop, y escucha las peticiones http.
   4. server.on**(**"ruta", funcion**)** Este método realiza una funcion cuando el cliente hace una petición http a la ruta. Por ejemplo:

server.on**(**"/saludar", funcion**)** Si el cliente esta en la siguiente url: 192.168.1.200/saludar, se realizará la funcion puesta como argumento.

* 1. server.send**(**200, "text/plain", "Hola mundo!"**)**; Esta funcion recibe como primer parámetro el código de respuesta HTTP. Luego, el tipo de contenido a trasnferir, y por último, el contenido de la respuesta. Se usa como función del método on. Por ejemplo:

server.on**(**"/", **[]()** **{**

server.send**(**200, "text/plain", "Hello World!"**)**;

**})**;

Si el cliente está en la ruta ip/ se ejecuta la funcion lambda. Este tipo de funcion es una funcion anónima, que se ejecuta al mismo tiempo de ser creada. La funcion dice: si el código es 200 (es decir que la solicitud tuvo éxito, refiriéndose a que hay un cliente en la url especificada en la funcion server.on) se va a volcar contenido en formato de texto plano, que será un Hello World!

* 1. server.onNotFound**(**handleNotFound**)**; Ejecuta la funcion de su argumento cuando el cliente está en una url que no conocemos. El handler pregunta a todos los server.on si tiene la dirección de url especificada en dicha funcion. Si ninguno lo tiene, entra al onNotFound.