LABORATORIO 5 – CONFIGURACIÓN DE ESCANER Y CLIENTE WiFi

Material:

NodeMCU y cable USB

En este laboratorio se configurará el NODEMCU como un escaner de redes. El escaner de redes detecta las redes WiFi que hay alrededor y nos muestra información de ellas. La información que nos puede suministrar es el SSID o nombre de la red, RSSI que es el valor de potencia en dBm de la señal, el tipo de encriptación y otras más.

Para trabajar el NodeMCU en WiFi ya sea como escaner, acces point, cliente o servidor WEB, lo primero que hay que hacer es llamar la librería del ESP8266 para tabajar WiFi. Esto se hace a través de:

#include "ESP8266WiFi.h"
Ya para el caso del NodeMCU como escaner, el comando **WiFi.scanNetworks()** escanea las redes WIFI habilitadas**.** En este caso vamos a visualizar las redes WiFi escaneadas, por esta razón establecemos la velocidad de la transmisión con el comando ***Serial.begin(9600);***

WiFi.SSID() obtiene el nombre de la red WiFi en la que está conectado y **WiFi.RSSI()** obtiene el nivel en dBm de la señal WiFi.

Otro comando utilizado es **WiFi.encryptionType(),** que muestra el tipo de encriptación utilizado en la red WiFi. Para esto tenga en cuenta los siguientes valores:

TKIP (WPA) = 2
WEP = 5
CCMP (WPA) = 4
NONE = 7
AUTO = 8

Teniendo en cuenta los comandos vamos a escribir el código para configurar el escaner de redes.

Ejercicio 1: Configure su NodeMCU como escaner de redes WiFi.

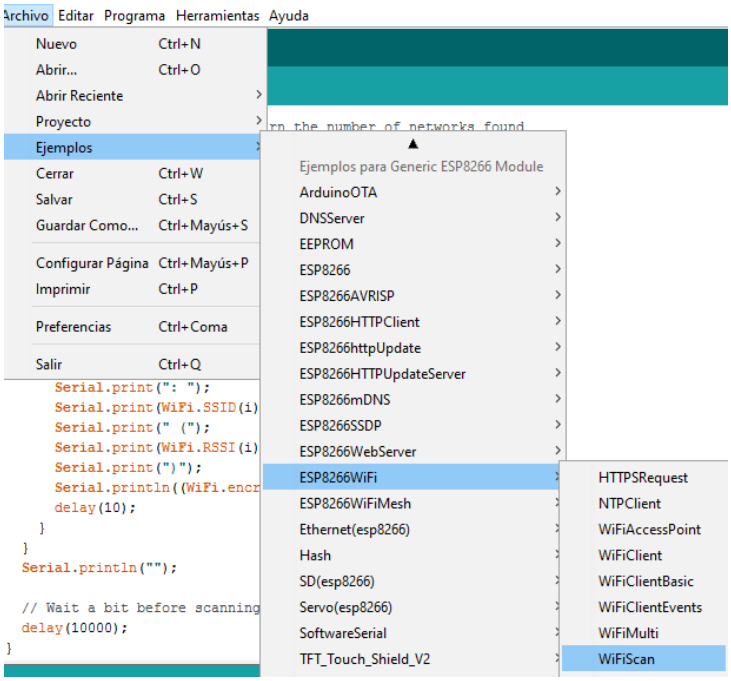
```
#include <ESP8266WiFi.h>
voidSetup(){
Serial.begin(9600);
Serial.println();
WiFi.mode(WIFI_STA);
WiFi.disconnect();
delay(100);
{

void loop(){
Serial.print("Iniciando escaneo ...");
int n = WiFi.scanNetworks();
Serial.println(n);
Serial.println("Redes encontradas:");
for (int i = 0; i < n; i++)
  {
    Serial.println(WiFi.SSID(i));
    Serial.println(WiFi.RSSI(i));
```

```
   delay(500);
  }
Serial.println();
delay(9000);
}
```

Analice el código utilizado, e incluya en el sketch que el NodeMCU los comandos necesarios para mostrar el tipo de seguridad utilizado por cada una una de las redes detectadas por el escáner de redes.

Compare ahora con el sketch o el código que acabó de utilizar con el encontrado en la librería en el IDE de arduino para el ESP8266. Para abrir este sketch sigamos la ruta de la siguiente figura:

Archivo/Ejemplos/ESP8266/WiFiScan.



¿Que diferencias encuentra entre los dos códigos?
¿Cuáles comandos se aplican en los dos códigos y cual es la función de estos?

Ejercicio 2: Configure el NodeMCU como un host. En este caso se debe utilizar el comando **WiFi.begin().** Otro comando util es **WiFi.localIP()** que averigua la dirección ip utilizada por el NodeMCU para conectarse a la red WiFi.

En este caso nos conectaremos a una red WiFi de la cual necesitaremos el mobre de la red SSID y el password, averíguelos y reeplázelos en el comando ***WiFi.begin("SSID","PASSWORD");*** del siguiente código.

```
#include <ESP8266WiFi.h>

voidSetup(){
Serial.begin(9600);
WiFi.mode(WIFI_STA);

delay(1000);
WiFi.disconnect();
Serial.println("INICIANDO");
WiFi.begin("SSID","PASSWORD");

while((!(WiFi.status() == WL_CONNECTED))){
delay(300);
Serial.println("...."); }

Serial.println("CONECTADO");
Serial.println((WiFi.localIP()));

{
voidloop(){
}
```

Analice el código anterior. ***WiFi.mode(WIFI_STA);*** es un comando opcional para indicar que el NodeMCU esta configurado como estación. El comando ***WiFi.disconnect();*** se utiliza para desconectar el NodeMCU a una red anterior a la que ya estuviera conectado.

Se iniciará la conexión WiFi y se repetirá a través del bucle **while ()** mientras que **WiFi.status ()** es distinto de **WL_CONNECTED**. El bucle saldrá sólo si el estado **WL_CONNECTED** cambia, es decir después de establecer una conexión satisfactoriamente.

ESP8266 entonces es capaz de operar tanto en estación como en el modo de punto de acceso. Esto proporciona la posibilidad de construir p.e. redes de malla.

Verifique conectividad con la placa de desarrollo a través del protocolo icmp. Tome el valor de la dirección IP y realice ping a esa dirección del NodeMCU.

Ejercicio 3: Deberá conectarse como estación pero utilizando el código WiFi.config(), el cual permite configurar una dirección IP estática, así como cambiar las direcciones DNS, puerta de enlace y subred. En este caso se asignará manualmente una dirección IP con máscara y Gateway, la IP debe encontrarse dentro de la red del access point al cual se conectará como cliente. Por favor veriffique le direccionamiento Ip donde se encuentra y asigna al NodeMCU una Ip no repetida, máscara y gateway.

TRABAJO PARA LA CASA:

Configurar el NodeMCU con WiFi Manager para evitar la codificación de las credenciales WiFi (SSID y password) en el código de Arduino.

Si tenemos valores codificados y cambia su router WiFi o desea llevar su dispositivo a otro lugar, va a tener que volver a programar su esp8266, lo cual no es ideal!

La configuración que hicimos hace que nuestro sketch sea menos compartible, ya que probablemente querrá reemplazar su SSID y contraseña antes de subir a algún sitio público como GitHub y cualquier persona que esté usando su código va a tener que actualizar el código antes de que puedan usarlo en su red.

Descargar la librería y configurar nuestro NodeMCU para que funcione correctamente el WiFi Manager.

https://github.com/tzapu/WiFiManager