Ayudantía 1 - NodeMCU **Redes**
INFO281 Javier Vallejos S.
Prof. Christian Lazo R. Fecha: 28/09/2018

Definición del escenario

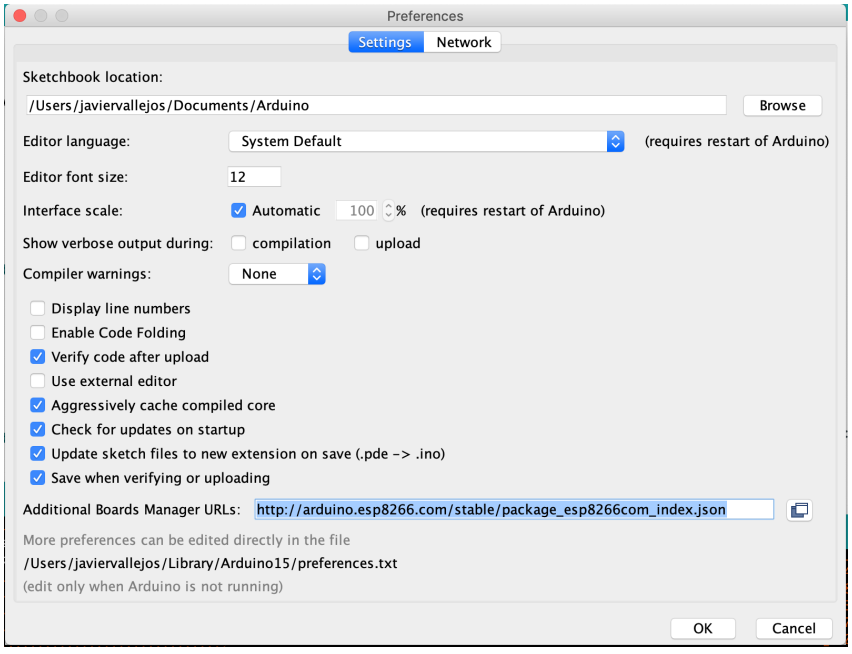
En un mundo globalizado es necesaria la comunicación entre dispositivos e infraestructuras, en la presente ayudantía abordaremos como conectar un microcontrolador con un servidor pasando desde su instalación y configuración en un pc hasta el envío de datos hacia un Servidor.

Configuración de la IDE de Arduino e instalación de Drivers.

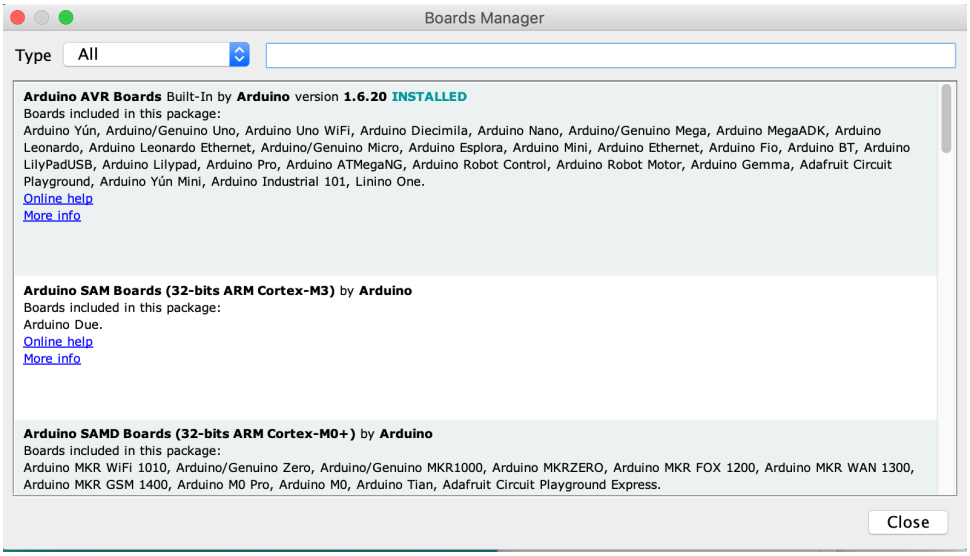
Para poder comenzar a programar en NodeMCU debemos ocupar la IDE de Arduino, esta placa nos hará los trabajos más fácil ya que trae incorporado un módulo WiFi ESP8266, esto nos permite conectarnos a cualquier *access point* o escanear redes con un par de lineas de código, pero como toda placa de desarrollo necesita Drivers para funcionar y un entorno de desarrollo, Para comenzar, pueden descargar el IDE de la página oficial: `https://www.arduino.cc/en/Main/Software`

Ahora con somos capaces de programar archivos con extención .ino, pero aún no podemos compilarlos y subirlos a nuestra placa de desarrollo, para ello es necesario configurar el entorno e instalar drivers del microcontrolador. Acá dejaré el link de los drivers oficiales de los desarrolladores para todos los S.O. Link: `https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers`

Una vez tengamos instalados los drivers, y el IDE abierto vamos a **Archivo, Preferencias** y ahí copiamos el siguiente link `http://arduino.esp8266.com/stable/package_esp8266com_index.json` donde dice **Additional Boards Manager URLs:**



Luego para asignarla como placa de Desarrollo vamos a **Herramientas, Board:, Board Manager** y se nos abrirá una ventana como la que aparece en el cuadro siguiente:



Escribimos **ESP8266** en el buscador e instalamos el controlador. Con esto estamos listos para crear el primer código y subirlo a NodeMCU!!

Códigos de ejemplos:

En esta sección vamos a ver varios códigos de ejemplo, el más básico consiste en encender el led que trae la placa de desarrollo incorporado, para ello escribimos el siguiente codigo:

```
void setup() {
  // Inicializamos el pin 13 como Salida
  pinMode(13, OUTPUT);
}
// Funcion donde va nuestro codigo una y otra vez
void loop() {
  digitalWrite(13, HIGH);   // Prendemos el led (HIGH es el nivel de voltaje)
  delay(1000);              // Esperamos 1 segundo
  digitalWrite(13, LOW);    // apagamos el led bajando el voltaje
  delay(1000);              // Esperamos otro segundo
}
```

Luego de ver el ejemplo más básico que podemos programar con NodeMCU, vamos a implementar el código de un servidor web, para ello, ejecutamos el siguiente código:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

const char* ssid = "NombreWiFi";
const char* password = "Clave";
```

```
ESP8266WebServer server(80);
const int led = 13;
void handleRoot() {
  digitalWrite(led, 1);
  server.send(200, "text/plain", "Hola desde esp8266!");
  digitalWrite(led, 0);
}

void handleNotFound(){
  digitalWrite(led, 1);
  String message = "Archivo no encontrado\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET)?"GET":"POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i=0; i<server.args(); i++){
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);
  digitalWrite(led, 0);
}

void setup(void){
  pinMode(led, OUTPUT);
  digitalWrite(led, 0);
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");
  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Conectado a ");
  Serial.println(ssid);
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());

  if (MDNS.begin("esp8266")) {
    Serial.println("MDNS responder started");
  }

  server.on("/", handleRoot);
```

```
  server.on("/inline", [](){
    server.send(200, "text/plain", "Si funciona!!");
  });
  server.onNotFound(handleNotFound);
  server.begin();
  Serial.println("HTTP server started");
}

void loop(void){
  server.handleClient();
}
```

Luego veremos el código para convertir nuestro Node en un Scanner Wifi:

```
#include "ESP8266WiFi.h"

void setup() {
  Serial.begin(115200);
  // Set WiFi to station mode and disconnect from an AP if it was previously connected
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
  Serial.println("Setup done");
}

void loop() {
  Serial.println("scan start");

  // WiFi.scanNetworks will return the number of networks found
  int n = WiFi.scanNetworks();
  Serial.println("scan done");
  if (n == 0)
    Serial.println("no networks found");
  else
  {
    Serial.print(n);
    Serial.println(" networks found");
    for (int i = 0; i < n; ++i)
    {
      // Print SSID and RSSI for each network found
      Serial.print(i + 1);
      Serial.print(": ");
      Serial.print(WiFi.SSID(i));
      Serial.print(" (");
      Serial.print(WiFi.RSSI(i));
      Serial.print(")");
      Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)?" ":"*");
      delay(10);
    }
  }
```

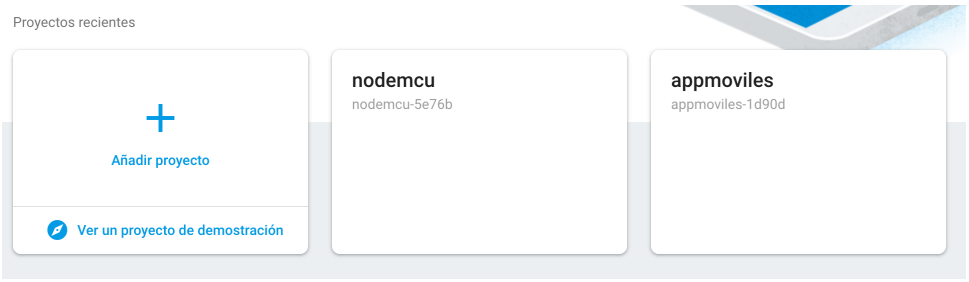
```
Serial.println("");

// Wait a bit before scanning again
delay(5000);
```

Crear una cuenta en Firebase

- Qué es Firebase:
  Firebase es una plataforma de backend para construir aplicaciones web, se encarga del manejo de la infraestructura permitiendo que el desarrollador se enfoque en otros aspectos de la aplicación. Entre sus características se incluye **base de datos de tiempo real**, autenticación de usuarios y almacenamiento(hosting) estático. La idea de usar Firebase es no tener que escribir código del lado del servidor y aprovechar al máximo las características que nos provee la plataforma.

- Registrarse en Firebase Entramos al link de Firebase `https://firebase.google.com/` la ventaja es que pueden iniciar sesión con su cuenta google. hay que tener en cuenta que esta plataforma es gratuita los primeros 10 GB, personalmente he dejado el NodeMCU enviando datos toda una tarde y nunca he pasado los 100kb.

- Crear un proyecto en Firebase Una vez que iniciamos sesión vamos a crear una aplicación, que en realidad es una base de datos donde almacenaremos los datos recolectados pudiendo consultarlos en tiempo real, es por ello que no usamos base de datos estructuradas, lo que quiere decir esto es que para consultar los datos el servidor avisa cuando varían y el usuario no tiene que estar preguntando los nuevos valores para actualizarlos.



  Luego de esto, lo que nos interesa es poder obtener una URL del host y un tocken de seguridad para enviar / recibir información desde cualquier dispositivo o sitio web hacía nuestra base de datos Firebase.

- Url del host y Tocken de seguridad
  Para poder encontrar el url y el tocken debemos ir a `https://console.firebase.google.com/project/`

y luego encontramos el tocken y el url como en la siguiente imágen en la url: `https://console.firebase.google.com/project/redes-180617/overview`

Enviar datos a Firebase con NodeMCU

Estructura de datos en Firebase para la geolocalización WiFi con Arduino y NodeMCU

La estructura para envíar datos a una base de datos NoSQL es en formato de un archivo JSON como el siguiente:

```
{
 "dispositivo" : {
   "MAC" : {
     "lat" : 38.3685,
     "lon" : -0.4219,
     "prec" : 100999,
     "nombre" : "Grupo 3"
   }
 }
}
```

Esta es la información que enviaremos desde Arduino o NodeMCU a Firebase, ahora veremos cómo hacerlo.

El equivalente al archivo JSON anterior con la sintaxis de NodeMCU sería lo siguiente:

```
// Petición PUT JSON
String toSend = "PUT /dispositivo/";
toSend += macStr;
toSend += ".json HTTP/1.1\r\n";
toSend += "Host:";
toSend += HOSTFIREBASE;
toSend += "\r\n" ;
toSend += "Content-Type: application/json\r\n";
String payload = "{\"lat\":";
payload += String(loc.lat, LOC_PRECISION);
payload += ",";
payload += "\"lon\":";
payload += String(loc.lon, LOC_PRECISION);
payload += ",";
payload += "\"prec\":";
payload += String(loc.accuracy);
payload += ",";
payload += "\"nombre\": \"";
payload += nombreComun;
payload += "\"}";
payload += "\r\n";
toSend += "Content-Length: " + String(payload.length()) + "\r\n";
toSend += "\r\n";
toSend += payload;
Serial.println(toSend);
client.println(toSend);
client.println();
client.flush();
client.stop();
```

Ejemplo: Código para obtener geolocalización a partir de las redes wifi cercanas:

```
#include <ESP8266WiFi.h>
#include "WifiLocation.h"

#define GOOGLE_KEY "AIzaSyDP4SOZgPpJCEYormxf53cA9tnFIPoxArk" // Clave API Google Geolocation
#define SSID "SSID" // SSID de tu red WiFi
#define PASSWD "PASSWD" // Clave de tu red WiFi
#define HOSTFIREBASE "HOST-FIREBASE" // Host o url de Firebase
#define LOC_PRECISION 7 // Precisión de latitud y longitud

// Llamada a la API de Google
WifiLocation location(GOOGLE_KEY);
location_t loc; // Estructura de datos que devuelve la librería WifiLocation

// Variables
```

```
byte mac[6];
String macStr = "";
String nombreComun = "NodeMCU";

// Cliente WiFi
WiFiClientSecure client;

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

  // Conexión con la red WiFi
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(SSID);
    // Connect to WPA/WPA2 network:
    WiFi.begin(SSID, PASSWD);
    // wait 5 seconds for connection:
    delay(5000);
    Serial.print("Status = ");
    Serial.println(WiFi.status());
  }

  // Obtenemos la MAC como cadena de texto
  macStr = obtenerMac();

  Serial.print("MAC NodeMCU: ");
  Serial.println(macStr);
}

void loop() {
  // Obtenemos la geolocalización WiFi
  loc = location.getGeoFromWiFi();

  // Mostramos la información en el monitor serie
  Serial.println("Location request data");
  Serial.println(location.getSurroundingWiFiJson());
  Serial.println("Latitude: " + String(loc.lat, 7));
  Serial.println("Longitude: " + String(loc.lon, 7));
  Serial.println("Accuracy: " + String(loc.accuracy));

  // Hacemos la petición HTTP mediante el método PUT
  peticionPut();

  // Esperamos 15 segundos
  delay(15000);
}

/********** FUNCIÓN PARA OBTENER MAC COMO STRING **********/
String obtenerMac()
```

```
{
  // Obtenemos la MAC del dispositivo
  WiFi.macAddress(mac);

  // Convertimos la MAC a String
  String keyMac = "";
  for (int i = 0; i < 6; i++)
  {
    String pos = String((uint8_t)mac[i], HEX);
    if (mac[i] <= 0xF)
      pos = "0" + pos;
    pos.toUpperCase();
    keyMac += pos;
    if (i < 5)
      keyMac += ":";
  }

  // Devolvemos la MAC en String
  return keyMac;
}

/********** FUNCIÓN QUE REALIZA LA PETICIÓN PUT **************/
void peticionPut()
{
  // Cerramos cualquier conexión antes de enviar una nueva petición
  client.stop();
  client.flush();
  // Enviamos una petición por SSL
  if (client.connect(HOSTFIREBASE, 443)) {
    // Petición PUT JSON
    String toSend = "PUT /dispositivo/";
    toSend += macStr;
    toSend += ".json HTTP/1.1\r\n";
    toSend += "Host:";
    toSend += HOSTFIREBASE;
    toSend += "\r\n" ;
    toSend += "Content-Type: application/json\r\n";
    String payload = "{\"lat\":";
    payload += String(loc.lat, LOC_PRECISION);
    payload += ",";
    payload += "\"lon\":";
    payload += String(loc.lon, LOC_PRECISION);
    payload += ",";
    payload += "\"prec\":";
    payload += String(loc.accuracy);
    payload += ",";
    payload += "\"nombre\": \"";
    payload += nombreComun;
    payload += "\"}";
```

```
    payload += "\r\n";
    toSend += "Content-Length: " + String(payload.length()) + "\r\n";
    toSend += "\r\n";
    toSend += payload;
    Serial.println(toSend);
    client.println(toSend);
    client.println();
    client.flush();
    client.stop();
    Serial.println("Todo OK");
  } else {
    // Si no podemos conectar
    client.flush();
    client.stop();
    Serial.println("Algo ha ido mal");
  }
}
```