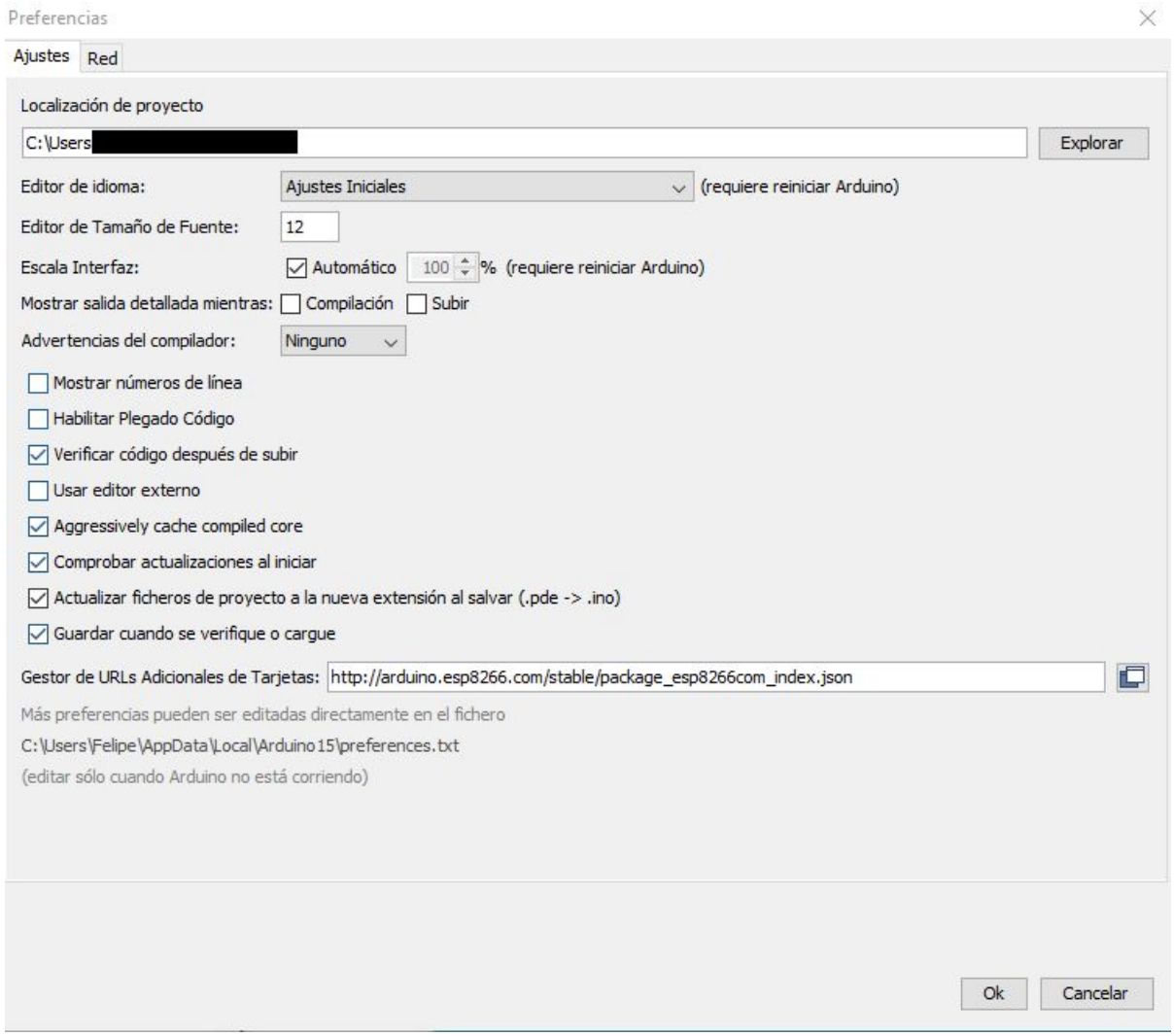
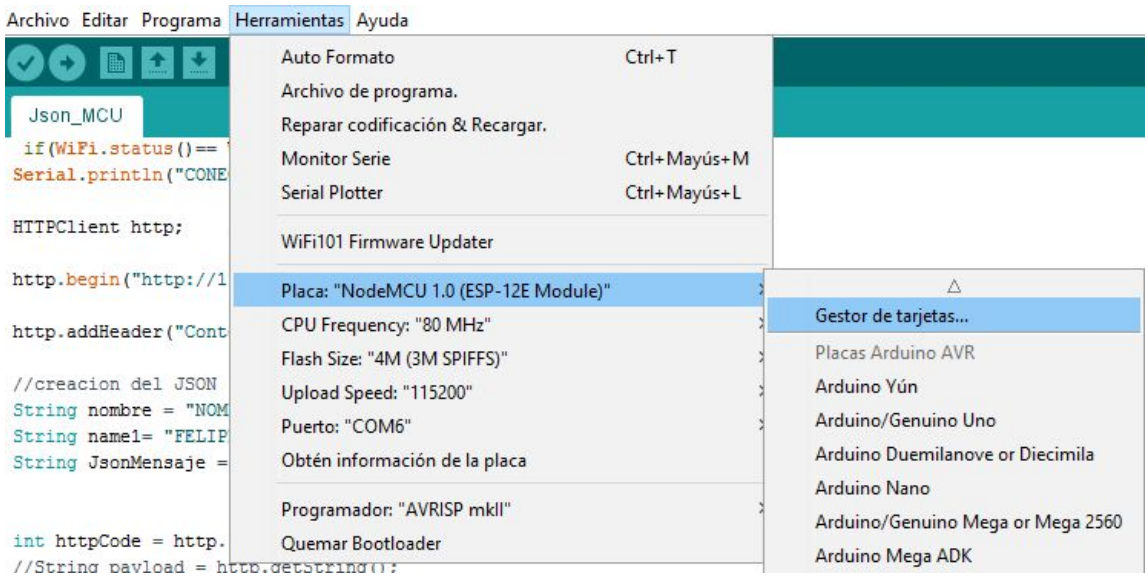
Configurar NodeMCU para enviar y recibir datos tipo JSON usando método POST y GET

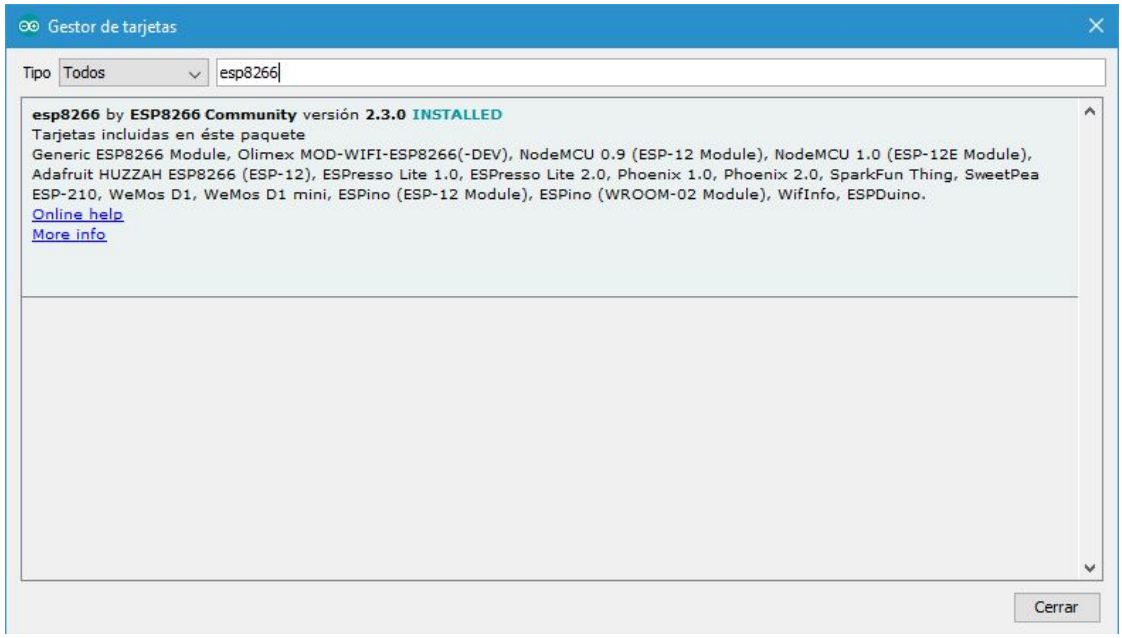
1. **Configurar nodeMCU:**
   a. **Descargar e instalar el plugin ESP8266**
   Para instalar el plugin primero debemos abrir el IDE de arduino y dirigirnos a \Archivo\preferencias.



En la parte inferior de la ventana, en "Gestor de URLs adicionales de tarjeta copiamos lo siguiente:
http://arduino.esp8266.com/stable/package_esp8266com_index.json
después de esto le damos clic en OK e ingresamos a \Herramientas\placa\Gestor de tarjetas.

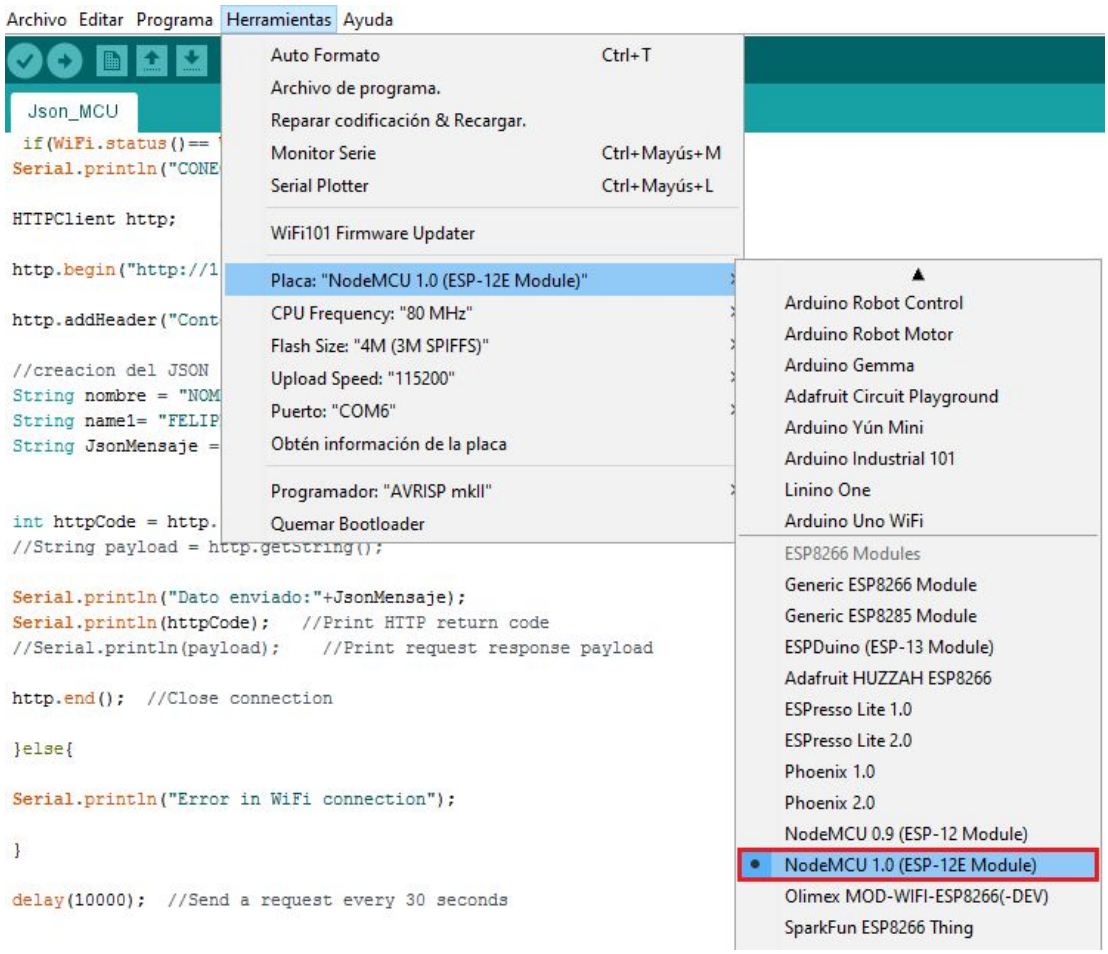


Buscaremos la linea que dice “esp8266 by ESP8266 community version” y la instalaremos. Esto puede tardar unos minutos dependiendo de la coneccion a internet.

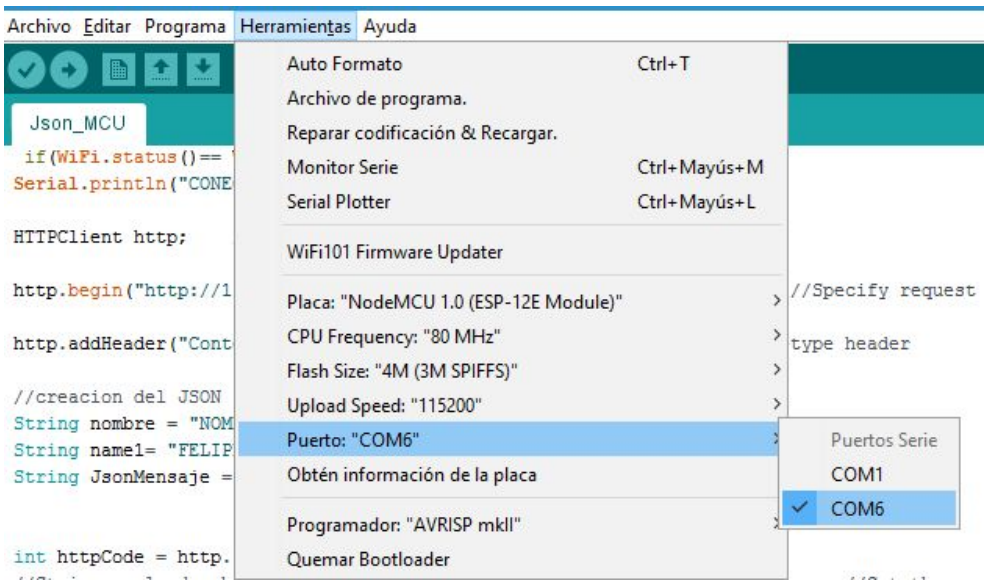


b. Configurar NodeMCU en arduino IDE

Ahora nos dirigimos a \Herramientas\Placa y ya veremos que podemos seleccionar el **NodeMCU 1.0 (ESP-12 Module).**



Verificamos que tengamos conectado nuestro NodeMCU y nos dirigimos a \Herramientas\Puerto y seleccionamos el puerto COM correspondiente a la coneccion de la placa.



Con esto ya finalizamos la configuración del NodeMCU en el arduino IDE.

2. **Solicitud HTTP POST:**
   a. **Incluimos las librerías:**
      Utilizaremos las librerías del ESP8266 para la configuración del método POST en el arduino IDE.

```
#include <ESP8266HTTPClient.h>
#include <ESP8266WiFi.h>
```

   b. **Método setup():**
      En el método setup solo inicializamos la conexión en serie para imprimir los resultados de la aplicación. Se establece conexión a un AP y el ESP8266 (NodeMCU) creará las solicitudes HTTP.

```
void setup() {
  Serial.begin(115200); //Serial connection
  WiFi.begin("yourSSID", "yourPASS");  //WiFi connection
  while (WiFi.status() != WL_CONNECTED) {  //Wait for the WiFI connection completion
    delay(500);
    Serial.println("Waiting for connection");
  }
}
```

      Para realizar la coneccion wifi es necesario ingresar “yourSSID” y “yourPASS” los cuales pertenecen a el SSID del wifi al que se conectara y a la contraseña que este tenga,respectivamente.

   c. **Método loop():**
      En el método loop se ingresaran todo lo relacionado a la solicitud HTTP con el método POST utilizando la clase HTTPClient.
      Primero declaramos un objeto el cual llamaremos “http” de la clase HTTPClient:

```
HTTPClient http;
```

      Después de esto debemos utilizar el método “begin” para realizar la coneccion con el URL del servicio web:

```
http.begin("http://192.168.1.88:9999/hello");
```

      Ahora utilizamos el método addHeader para definir el encabezado de nuestro webservice, que en este caso es “application/json”:

```
http.addHeader("Content-Type", "application/json");
```

      Enviaremos a traves del metodo POST, el JSON que hayamos creado, pero lo asignaremos a una variable tipo Int ya que este método retorna un código de respuesta HTTP importante para el manejo de errores.

```
int httpCode = http.POST(JsonMensaje);
```

      Al final llamamos al método end() del objeto para garantizar que la coneccion TCP esté cerrada.

```
http.end();
```

A continuación todo el código referente al método loop:

```
void loop() {

  if (WiFi.status() == WL_CONNECTED) { //Check WiFi connection status
    Serial.println("CONECTADO !!!!");

    HTTPClient http;    //Declare object of class HTTPClient
    http.begin("http://192.168.0.15:8080/ws-prueba/webresources/ws");      //Specify request destination
    http.addHeader("Content-Type", "application/json");  //Specify content-type header

    //creacion del JSON
    String nombre = "NOMBRE";
    String name1 = "Julanito Perez";
    String JsonMensaje = "{\"" + nombre + "\":\"" + name1 + "\"}";
    int httpCode = http.POST(JsonMensaje);   //Send the request
    //String payload = http.getString();                                        //Get the response payload

    Serial.println("Dato enviado:" + JsonMensaje);
    Serial.println(httpCode);   //Print HTTP return code
    //Serial.println(payload);    //Print request response payload

    http.end();  //Close connection

  } else {
    Serial.println("Error in WiFi connection");
  }
  delay(10000);  //Send a request every 30 seconds
}
```

3. **<u>Solicitud HTTP GET:</u>**
   a. **Incluimos las librerías:**
   Al igual que en el uso del método POST, debemos incluir las librerías del ESP8266 para la configuración del GET.

```
#include <ESP8266HTTPClient.h>
#include <ESP8266WiFi.h>
```

   b. **Método setup():**
   El método setup() que se utiliza para la configuración del método GET es el mismo que se usó para el POST, ya que es necesario establecer la conexión a un AP para que el ESP8266 (NodeMCU) cree las solicitudes HTTP.

```
void setup () {
  Serial.begin(115200);
  WiFi.begin("yourSSID", "YourPASS");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print("Connecting..");
  }
}
```

Nuevamente los parámetros “yourSSID” y “yourPASS” corresponden a el SSID del wifi al que se conectara y a la contraseña que este tenga, respectivamente.

c. Método loop():

Para llevar a cabo la solicitud HTTP con el método GET debemos declarar un objeto de clase HTTPClient, el cual llamaremos “http”.

```
HTTPClient http;
```

Lo siguiente es llamar al método begin en el objeto “http” el cual recibirá como parámetro la URL del web service a la que queremos conectarnos y realizaremos la petición GET.

```
http.begin("http://192.168.1.88:9999/hello");
```

En el caso de no tener un webservice montado, podemos utilizar la siguiente URL con la finalidad de probar la petición GET a esta.

URL: http://jsonplaceholder.typicode.com/users/1

A continuación enviaremos la solicitud llamando al método GET con el objeto “http”. Este método retorna el estado de la operación el cual es importante para el manejo de errores. Si el valor es mayor a 0, entonces es un código HTTP estándar. Si es menor a 0 entonces hay un error de cliente relacionado con la conexión.

```
int httpCode = http.GET();
```

Los códigos de error disponibles para este método puedes verlos en el siguiente enlace:

https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266HTTPClient/src/ESP8266HTTPClient.h#L45

Una vez conocido el valor que tiene el código, si este es mayor a 0 podremos imprimir la respuesta llamando al método getString en el objeto “http”.

```
String payload = http.getString();
Serial.println(payload);
```

Finalmente llamamos el método final para cerrar la conexión TCP.

```
http.end();
```

A continuación todo el código referente al método loop:

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

void setup () {
  Serial.begin(115200);
  WiFi.begin("yourSSID", "YourPASS");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print("Connecting..");
  }
}
void loop() {
  if (WiFi.status() == WL_CONNECTED) { //Check WiFi connection status
    HTTPClient http;  //Declare an object of class HTTPClient
    http.begin("http://jsonplaceholder.typicode.com/users/1");  //Specify request destination}
    int httpCode = http.GET();    //Send the request
    if (httpCode > 0) { //Check the returning code
      String payload = http.getString();   //Get the request response payload
      Serial.println(payload);       //Print the response payload
    }
    http.end();   //Close connection
  }
  delay(30000);    //Send a request every 30 seconds
}
```

Guia hecha por : Felipe Garaycochea, Universidad Autónoma de Occidente.

Referencias:

- Techtutorialsx. ESP8266: HTTP POST Requests. Julio 21 2016. [Online] Disponible en: <https://techtutorialsx.com/2016/07/21/esp8266-post-requests/>
- Techtutorialsx. ESP8266: HTTP GET Requests. Julio 17 2016. [Online] Disponible en: <https://techtutorialsx.com/2016/07/17/esp8266-http-get-requests/>
- Prometec. PROGRAMANDO NODEMCU CON ARDUINO IDE. Abril 2017. [Online] Disponible en: <http://www.prometec.net/nodemcu-arduino-ide/>