ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

TECNOLOGÍAS DE SOFTWARE PARA ELECTRÓNICA

COMUNICIÓN CON SERVICIOS WEB EN EL ESP8266 NODEMCU

DOCENTE: ING. DARWIN ALULEMA

ESTUDIANTES:

SALAZAR ISABEL

MORALES STEVE

REINOSO SANTIAGO

NRC: 4463

SANGOLQUÍ, 25 DE ABRIL DEL 2019

.

Índice general

0.1. PLANTEAMIENTO DEL PROBLEMA

El desarrollo de sistemas de tecnología basadas en el empleo de servicios web como herramienta principal sigue desarrollándose de manera acelerada, es así que en este tiempo la mayoría de aplicaciones de control o cualquier otro tipo de estas llevan consigo una implementación de un control software y este a su vez lleva un servicio web como apoyo para el desarrollo de su funcionalidad, en conjunto con ESP8266 NODEMCU el cual es un módulo wifi que permite enlazarce con el internet para así establecer una conección con los posibles servicios web a consumir.

0.2. OBJETIVOS

0.2.1. OBJETIVO GENERAL

Establecer una conección en tiempo real entre el módulo ESP8266 NODEMCU y los servicios web mediante la interfas de arduino y su correcta programación.

0.2.2. OBJETIVOS ESPECIFICOS

- Permitir la interacción entre clientes mediante el uso de un servidor.
- Analizar el funcionamiento y programación del módulo ESP8266 NODEMCU.

0.3. ESTADO DEL ARTE

En el campo de la comunicación con los servicios web empleando hardware adicional como lo son módulos wi-fi, Arduino, etc., se a dispuesto diferentes aplicaciones que motivan la innovación permitiendo al usuario obtener accesos directo a su objetivo de búsqueda como es mencionado en el proyecto de grado "Diseño e implementación de un sistema de vigilancia remota para una residencia usando ESP8266 NODEMCU", desarrollado por A. Soledad Zapata Y. y Andrés R. Vallejo P. en el año 2017 en el cual se plantea implementar un sistema que monitoree una vivienda a través del censo de 6 eventos intercambiando las muestras obtenidas mediante Wi-Fi usando el módulo ESP8266 NODEMCU hacia un servidor de base de datos. Por otra parte, podemos apreciar una aplicación más exterior en el trabajo de grado "Sistema de Medición Acústica usando NODEMCU ESP8266 para Determinar el Nivel de Ruido en Av. Víctor Larco cuadra 14 Trujillo 2018", desarrollado por Br. Otiniano López, Mercedes Francisco en el año 2018 que describe una implementación más electrónica del módulo ESP8266 NODEMCU para la medición acústica del nivel de ruido con lo cual también se busca prever enfermedades en los habitantes del sector. En comparación con nuestro objetivo los trabajos anteriormente mencionados presentan una misma vía de comunicación la cual es el módulo ESP8266 NODEMCU y a su vez el intercambio de información, sin embargo, podemos apreciar un enfoque más especifico el cual es la vigilancia de una vivienda o la medición acústica de niveles de ruido, en cambio, en el presente trabajo buscamos obtener un intercambio de información más básico.

Basados en la idea de comunicarse con servicios web se debe tener conocimiento de que un servidor Web es un software diseñado para la transferencia de hipertextos, o páginas en lenguaje HTML, este servidor se encuentra todo el tiempo en espera de algún tipo de petición por parte de un cliente, el servidor envía ante la petición el código y el cliente es el encargado de interpretarlo como lo afirma el trabajo de grado "Implantación de los servicios web 2.0 para la página del Departamento de Eléctrica y Electrónica de la Escuela Politécnica del Ejercito", desarrollado por Sr. Juan Felipe Calle Zhañay en el año 2011 en el cual se plante la configuración y el funcionamiento de un servidor web con aplicaciones Web 2.0 para de este modo potenciar la interacción existente entre los estudiantes del departamento y sus docentes. En comparación con nuestro proyecto podemos destacar los servicios web que se pretende ofrecer y hacer uso, aunque se mantiene una gran diferencia en el hardware de los mismos

0.4. MARCO TEÓRICO

0.4.1. INTERNET DE LAS COSAS (IOT)

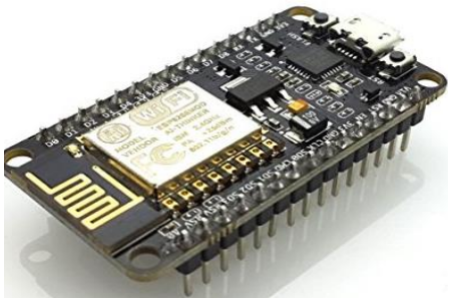
El internet de las cosas (IOT) consiste en que diferentes cosas u objetos tengan la capacidad de conectarse a internet en cualquier momento y en cualquier lugar. En un sentido más técnico, consiste en la integración de sensores y dispositivos en objetos cotidianos que estén conectados a internet a través de redes fijas e inalámbricas. De esta manera, cualquier objeto es susceptible de ser conectado y manifestarse en la red. Además, el IOT implica que todo objeto puede ser una fuente de información. (Tojeiro, 2014).
El internet de las cosas está presente en nuestro día a día. En nuestro entorno laboral, en nuestra casa, en la escuela, en el supermercado, etc. Un ejemplo de ello sería una máquina expendedora de refrescos que funciona con la publicación de un tweet, o con un hashtag en la página de la propia empresa o con un lique en Facebook.
En definitiva, el objetivo ideal del IOT sería lograr que cualquier objeto tenga vida propia a través de internet y con ello una identidad. Tojeiro, 2014)

Figura 1: INTERNET DE LAS COSAS

0.4.2. ESP8266 NODEMCU

Según (Naylampmechatronics, 2017)en su teoría indicó es un nuevo dispositivo orientado al internet de las cosas (IT) tiene un chip integrado llamado ESP8266 es muy pequeño siendo su ventaja principal y permite conectarme a internet vía Wi-Fi, teniendo un costo muy cómodo al alcance de todos, es parecido o similar al arduino que integra muchos componentes librerías, código, sensores, etc.

Figura 2: NODEMCU-ESP8266



Funcionamiento de Nodemcu Esp8266

Según (Naylampmechatronics, 2017)en su teoría indicó usando esta nueva tecnología que cuenta un chip integrado llamado ESP 8266 muy pequeño capaz de conectarme a internet vía Wi-Fi, usando datos del internet a través del celular, esta nueva tecnología permite registrar y guardar información en tiempo real siendo a una base de datos Mysql que está alojado en la nube conectándome con la página web (PHP) para ver los reportes de dicha medida, la unidad de medida se encuentra en decibeles.

Variantes

Como ya lo hemos dicho el ESP8266 es solo un procesador, pero su versión varia a la hora de construirlo sobre una placa impresa ya que sus características de construcción difieren en diferentes aspectos. Existen diferentes marcas fabricantes de estas excelentes variantes basadas en ESP8266. AI-Thinker la empresa China es una de las más importantes, con una extensa variedad de módulos de una excelente calidad a nivel global. Wemos (Compañía China) y Olimex (Europa) también aportan sus propias versiones. Las compañías Norte Americanas Adafruit y SparkFun no se pueden quedar atrás, fabricando dos modelos más de estas poderosas tarjetas Wi-fi. A continuación, revisaremos cada una de estas variantes de la global AI-Thinker.

Figura 3: VARIACIONES DE ESP

0.4.3. SERVIDOR WEB

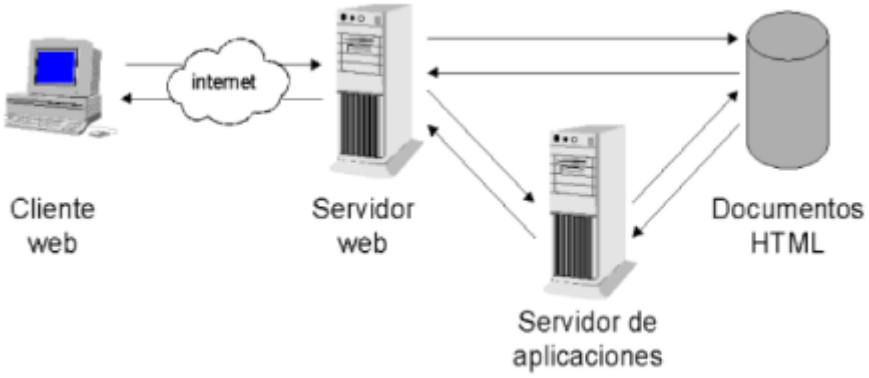
Un servidor Web es un software diseñado para la transferencia de hipertextos, o páginas en lenguaje HTML, este servidor se encuentra todo el tiempo en espera de algún tipo de petición por parte de un cliente, el servidor envía ante la petición el código y el cliente es el encargado de interpretarlo.

Los servidores Web no solo son encargados de la transferencia de código HTML, sino que también pueden presentar aplicaciones Web, estas pueden ejecutarse en el servidor o en el cliente, cuando estas se ejecutan en el cliente se requiere que este cuente con los programas para interpretar el código enviado por el servidor.

La principal desventaja de tener que ejecutar el código es que se tiene que instalar y en algunos casos la capacidad del navegador no puede interpretar el código del programa.

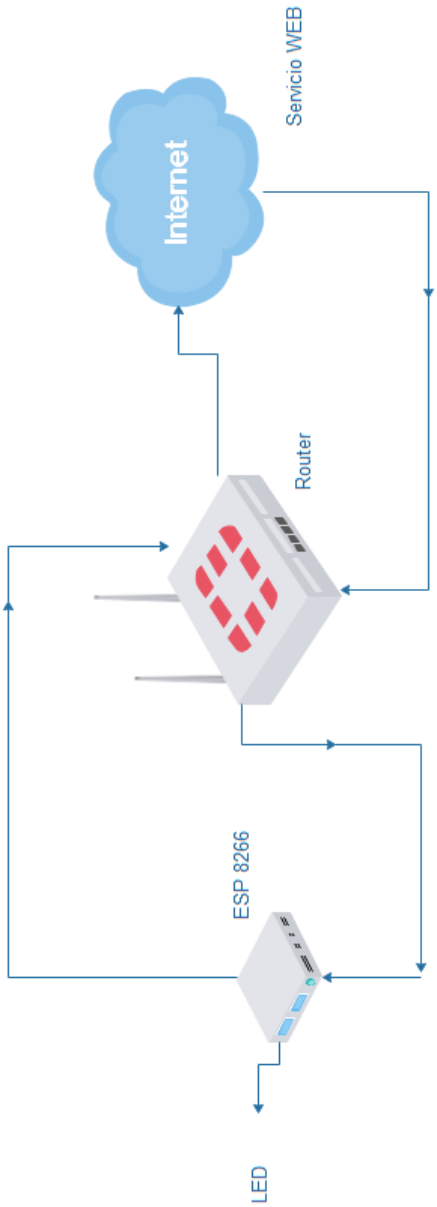
Para evitar que sea necesario un programa y con esto que el cliente utilice un navegador con pluginsextras, se ejecutan las aplicaciones web en el servidor, mostrando al final el resultado funcionando de la siguiente manera:

Figura 4: FUNCIONAMIENTO SERVICIOS WEB



0.5. DIAGRAMAS

Figura 5: DIAGRAMA DE BLOQUES



0.6. LISTA DE COMPONENTES

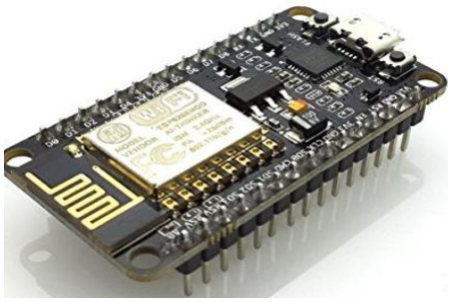
- **PROGRAMA ARDUINO IDE**

  Entorno de desarrollo integrado, llamado IDE (sigla en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

Figura 6: PROGRAMA ARDUINO IDE

- **ESP8266 NODEMCU**

Figura 7: NODEMCU-ESP8266



0.7. MAPA DE VARIABLES

	Variables Globales			Variable Temporal	Monitor Serial
	ssid	password	host	line	
	ESPE	****	www.vermiip.es		
1	ESPE	****	www.vermiip.es		..... connected
2	ESPE	****	www.vermiip.es		[Conectando a www.vermiip.es ... conectado]
3	ESPE	****	www.vermiip.es		[Enviando peticion]
4	ESPE	****	www.vermiip.es		[Respuesta:]
5-717	ESPE	****	www.vermiip.es		
718	ESPE	****	www.vermiip.es	HTTP/1.1 200 OK	HTTP/1.1 200 OK
719	ESPE	****	www.vermiip.es	Location: https://ufasiqpntise.espe.edu.ec:8443/portal/gateway?sessionid=fa000010a001e7b5462f1c15c&portal=a=a43af030-5a11-11e8-ac7e-42cbee303e0d8&action=cwa&token=02f4b44a2d4043339dec8643a8bc61d0f8&redirect=www.vermiip.es/	Location: https://ufasiqpntise.espe.edu.ec:8443/portal/gateway?sessionid=fa000010a001e7b5462f1c15c&portal=a=a43af030-5a11-11e8-ac7e-42cbee303e0d8&action=cwa&token=02f4b44a2d4043339dec8643a8bc61d0f8&redirect=www.vermiip.es/
720	ESPE	****	www.vermiip.es	Content-Type: text/html	Content-Type: text/html
721	ESPE	****	www.vermiip.es	Content-Length: 459	Content-Length: 459
722	ESPE	****	www.vermiip.es		
723	ESPE	****	www.vermiip.es	<HTML><HEAD><TITLE> Web Authentication Redirect</TITLE><META http-equiv="Cache-control" content="no-cache"><META http-equiv="Pragma" content="no-cache"><META http-equiv="Expires" content="-1"><META http-equiv="refresh" content="1; URL=https://ufasiqpntise.espe.edu.ec:8443/portal/gateway?sessionid=fa000010a001e7b5462f1c15c&portal=a43af030-5a11-11e8-ac7e-42cbee303e0d8&action=cwa&token=02f4b44a2d4043339dec8643a8bc61d0f8&redirect=www.vermiip.es/"></HEAD></HTML>	<HTML><HEAD><TITLE> Web Authentication Redirect</TITLE><META http-equiv="Cache-control" content="no-cache"><META http-equiv="Pragma" content="no-cache"><META http-equiv="Expires" content="-1"><META http-equiv="refresh" content="1; URL=https://ufasiqpntise.espe.edu.ec:8443/portal/gateway?sessionid=fa000010a001e7b5462f1c15c&portal=a43af030-5a11-11e8-ac7e-42cbee303e0d8&action=cwa&token=02f4b44a2d4043339dec8643a8bc61d0f8&redirect=www.vermiip.es/"></HEAD></HTML>
724	ESPE	****	www.vermiip.es		
725	ESPE	****	www.vermiip.es		[Desconectado]

0.8. EXPLICACIÓN DE CÓDIGO FUENTE

Incluimos la librería ESP8266Wifi.h también declaramos contantes de tipo char que vamos a usar a lo largo de todo el programa.

Figura 8: LIBRERIAS-ESP8266wifi.h

```
#include <ESP8266WiFi.h>                    //Incluye la librería ESP8266WiFi

const char* ssid = "TP-Link_F04A";           //Indicamos el nombre de la red WiFi (SSID) a la que queremos conectarnos.
const char* password = "27081419";           //Indicamos la contraseña de de red WiFi

const char* host = "www.vermiip.es";        //Declaramos el servidor de conexión
```

Abrimos la comunicación serial a 115200 baudios porque vamos a trabajar a con wifi.
Declaramos al pin 2 (internamente está conectado con el led que existe en la placa) como pin de salida.
Mostramos un mensaje con la SSID que nos queramos conectar y empezamos la conexión con la función Wifi.begin(SSID,contraseña);
Creamos un while con la condicion WiFi.status()! = WL_CONNECTED.
WiFi.status() nos puede devolver:

- WL_CONNECTED: cuando se conecta a una red WiFi
- WL_NO_SHIELD: cuando no hay ningún escudo WiFi presente
- WL_IDLE_STATUS: es un estado temporal asignado cuando se llama a $WiFi.begin()$ y permanece activo hasta que el número de intentos caduque (lo que resulta en WL_CONNECT_FAILED) o se establece una conexión (que resulta en WL_CONNECTED).
- WL_NO_SSID_AVAIL: cuando no hay SSID disponible.
- WL_SCAN_COMPLETED: la red scan se completa.
- WL_CONNECT_FAILED: cuando falla la conexión para todos los intentos.
- WL_CONNECTION_LOST: a cuando se pierde la conexión
- WL_DISCONNECTED: cuando se desconecta de una red

El objetivo de este while es conectarse con la red por eso mientras no devuelva WL_CONNECTED va a seguir intentado y nos mostrara en consola un punto.
Desde de establer la conexión es decir salio del bucle nos mostrara un mensaje de çonnected”

El objetivo de este while es conectarse con la red por eso mientras no devuelva WL_CONNECTED va a seguir intentado y nos mostrara en consola un punto

Desde de establer la conexión es decir salio del bucle nos mostrara un mensaje de çonnected"

Figura 9: FUNCIÓN VOID SETUP

```
void setup()
{
  Serial.begin(115200);                         //Inicializamos el Puerto Serie
  Serial.println();
  pinMode(2,OUTPUT);
  Serial.printf("Connecting to %s ", ssid);     //Inicializamos la conexión Wi-Fi en modo Station
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" connected");
}
```

Creamos un objeto WiFiClient de nombre client.

Mostramos en consola a que servidor(host) nos hemos conectado.

Creamos un if con la condicion client.connect(host, 80), con esto estamos confirmando si el servidor(host) es una pagina web porque 80 es el codigo para ello.

Si la condicon es verdadera mostramos çonectado]z prendemos el led interno.

Mostramos el mensaje "[Enviando peticion]z enviamos un apeticion al servidor(host)

$client.print...$ con esta peticion obtendremos el codigo html de la pagina, para el ejercio usamos un servidor que nos informara cual es nuestra IP.

Notificamos que vamos a recibir la respuesta "[Respuesta:]"

Y la recibimos mediante estemos conectados al servidor client.connected()

Luego si tenemos datos de llegada client.available()

Los almacenamos en un variable temporal line línea por línea client.readStringUntil($'$ n'); y las mostramos Cuando termine de revivir todos los datos saldrá del bucle y nos desconectaremos del servidor $client.stop()$ notificamos " $n[Desconectado]$" y apagamos el led.

Si nuestra primera condicIÓn if (client.connect(host, 80)) no se cumplió es decir no es un servidor web.

Mostramos "conexion con el servidor no lograda" y nos desconectamos.

Ponemos un retardo de 10 segundos y finalizamos el código

Figura 10: FUNCION VOID LOOP

```
void loop()
{
  WiFiClient client;                                    //Inicializamos el cliente (client)

  Serial.printf("\n[Conectando a %s ... ", host);       //Establecemos la conexión con el servidor
  if (client.connect(host, 80))
  {
    Serial.println("conectado]");
    digitalWrite(2,HIGH);
    Serial.println("[Enviando peticion]");               //Enviamos la petición de datos
    client.print(String("GET /") + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n" +
                 "\r\n"
                );
```

Figura 11: CÓDIGO RESPUESTA

```
    Serial.println("[Respuesta:]");                     //Leemos la respuesta del servidor
    while (client.connected())
    {
      if (client.available())
      {
        String line = client.readStringUntil('\n');
        Serial.println(line);
        }
      }
    }
    client.stop();                                      //Finalizamos la conexión con el servidor
    Serial.println("\n[Desconectado]");
    digitalWrite(2,LOW);
  }
```

Figura 12: PRIMERA CONDICIÓN DEL IF NO CUMPLE ENTONCES CONEXION NO LOGRADA

```
    else
    {
      Serial.println("conexion con el servidor no lograda!]");
      client.stop();
    }
    delay(10000);
}
```

0.9. DESCRIPCIÓN DE PRERREQUISITOS Y CONFIGURACIÓN

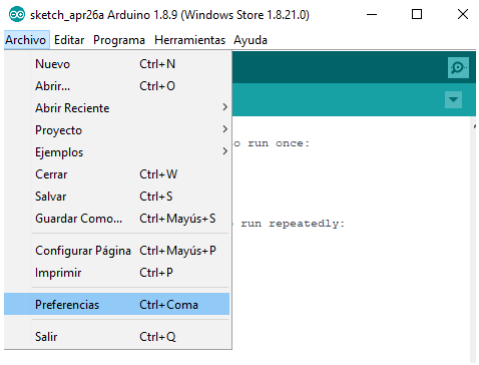
Uno de los prerequisistos importantes para el presente trabajo es la configuración del entorno de trabajo **Arduino IDE** para poder hacer uso de la placa ESP8266 NodeMCU, para lo cual se debe seguir los siguientes pasos:

- Abrir el programa Arduino IDE

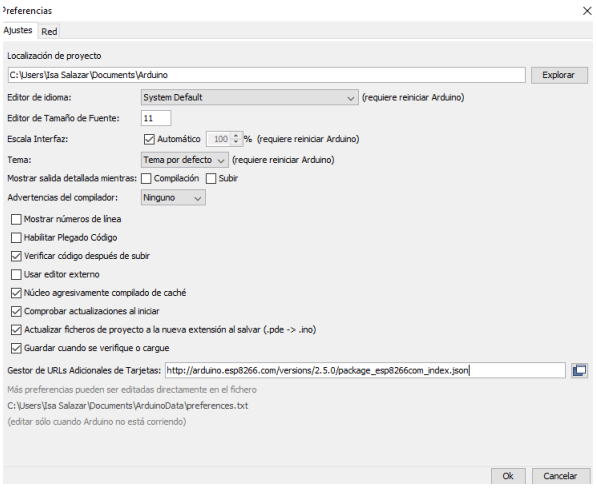
  Fuente de Descarga: `https://www.arduino.cc/en/Main/Software`

Figura 13: INICIO DE ARDUINO IDE

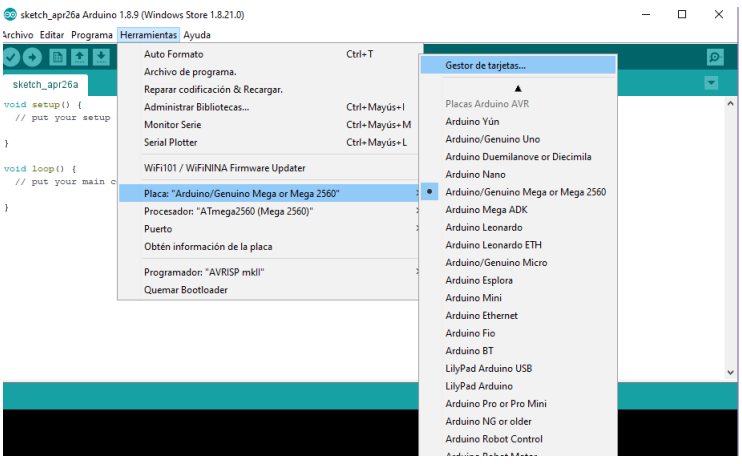
- Una vez en la hoja de trabajo se procede a dar click en **Archivo**, ubicado en la parte superior derecha de la hoja de trabajo, se despliega un menú y procedemos a dar click en **Preferencias**.



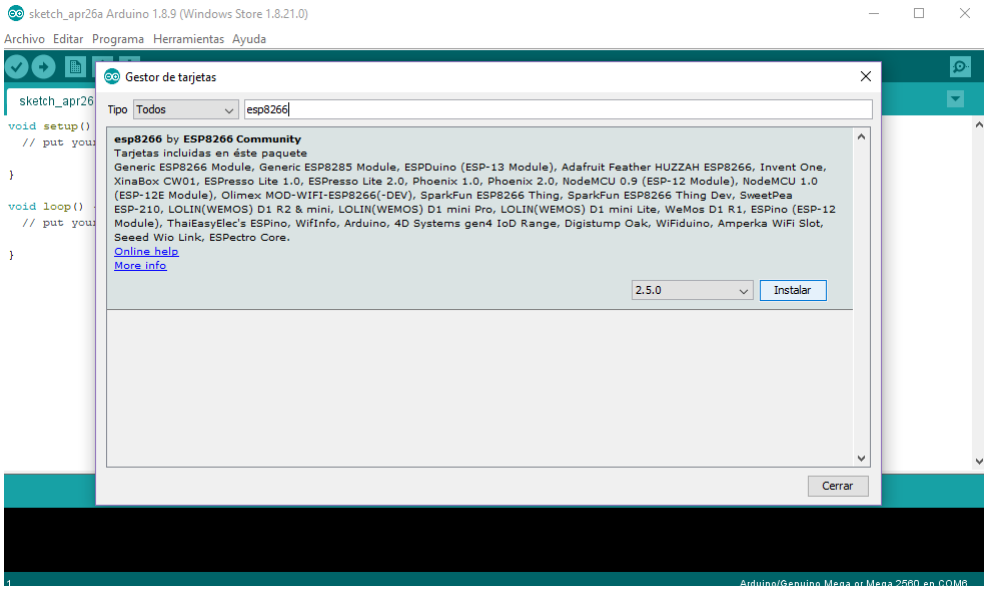
- En **Gestor de URLs Adicionales de Tarjetas** es necesario poner la siguiente url: `http://arduino.esp8266.com/stable/package_esp8266com_index.json` y camos click en .ºk". Este nos va a permitir tener acceso a diferentes modelos de placas.



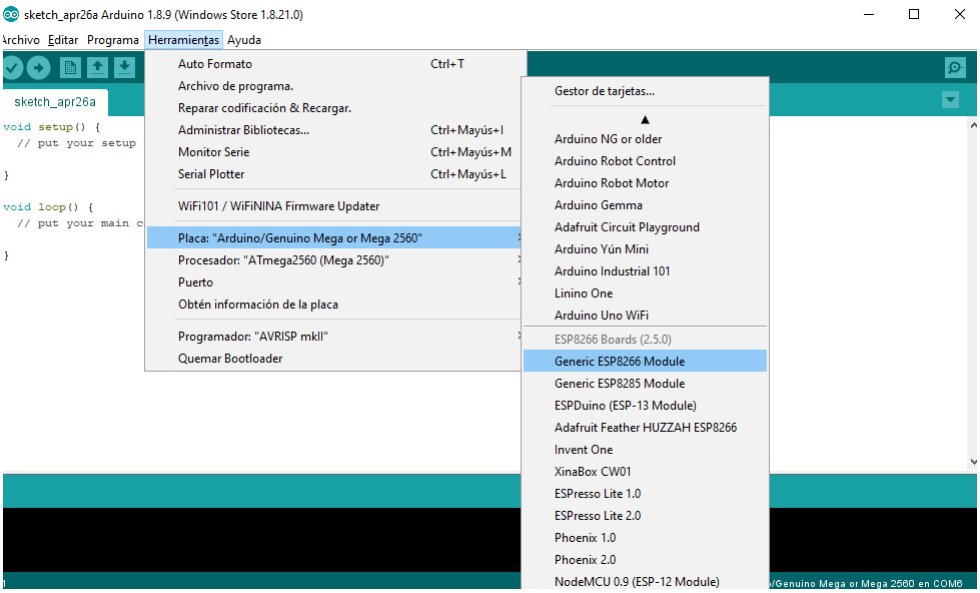
- De regreso en la hoja de trabajo damos click en **Herramientas**, en el menú que se despliega damos click en **Placa: xxxxxxxxxxx** y se vuelve a desplegar otro menú en el cual damos click en **Gestor de Tarjetas**.



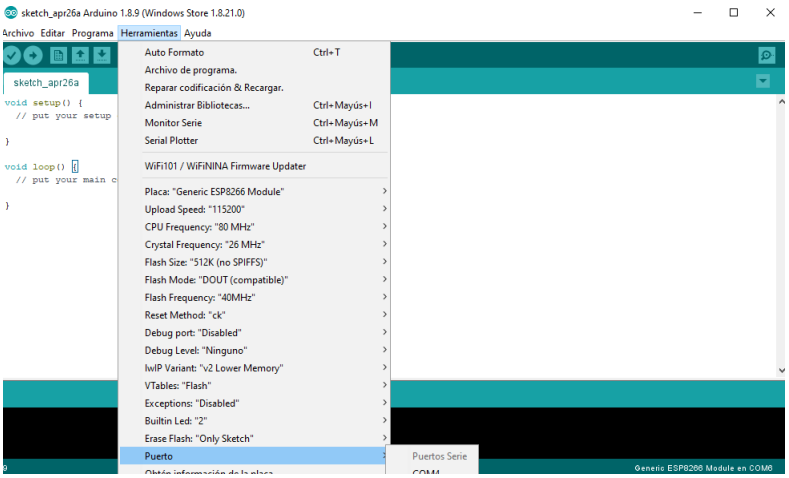
- En la ventana que se despliega buscamos **esp8266** y procedemos a instalar la versión **Community** que esté disponible para usted.



- En el mismo menú **Herramientas** damos click en **Placa: xxxxxxxxxxx** y ibservamos que se nos despliega un menú más extenso con diferentes tios de placas **esp**. Escogemos **Generic ESP8266 Module**.



- Continuando en el menú **Herramientas** al conectar la placa podemos observar que se muestra mas opciones de configuración e incluso el puerto en el que se creo la placa.



0.10. CONCLUSIONES

- Para el presente trabajo de investigación procedemos a concluir que en la actualidad la comunicación con los servicios web se produce de manera mas espontanea y con mucha mas amplitud de búsqueda lo que permite al usuario ser parte de una conexión eficiente.
- El módulo ESP8266 nodecun es parte de una familia de módulos que permiten la conexión wi-fi entre usuarios y servidores de manera practica y con un hardware mínimo, es decir, que es aplicable para diferentes proyectos de aprendizaje e implementación.
- La programación realizada para el presente trabajo se desarrolló de manera fluida debido a que se utilizó el entorno de programación de Arduino el cual permite el desarrollo del programa con mayor entendimiento y cuyo lenguaje se interpreta de mejor manera.

0.11. RECOMENDACIONES

- En el entorno de programación utilizado se recomienda comentar el código debido a que si en algún momento se necesita editar el mismo se pueda identificar la parte por cambiar y ubicar correctamente el funcionamiento.
- Tener muy en cuenta que que el modulo establezca correctamente la conexion con los servicios web, de ese modo evitar no lograr captar correctamente los datos.
- La programacion del modulo es la misma para el arduino, de ese modo se debe tener en cuenta que se realice la configuracion correcta en el IDE de Arduino para tener un correcto uso.

0.12. CRONOGRAMA

Figura 14: CRONOGRAMA DE TRABAJO INVESTIGATIVO

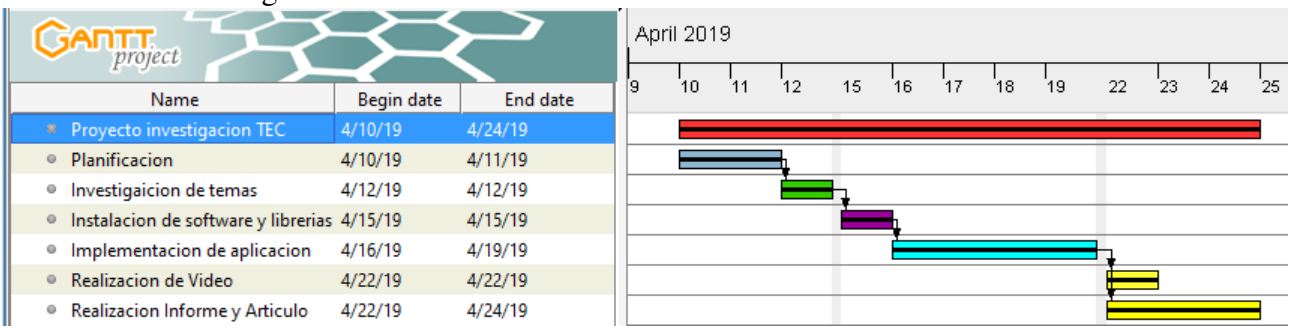


Figura 15: DIAGRAMA DE PROCESOS DEL TRABAJO INVESTIGATIVO

0.13. BIBLIOGRAFIA

- Rodrigues, C. M., & Castro, B. S. A Vision of Internet of Things in Industry 4.0 with ESP8266. International Journal of Electronics and Communication Engineering and Technology, 9(1), 20.
- Otiniano López, M. F. (2018). Sistema de Medición Acústica usando NODEMCU ESP8266 para Determinar el Nivel de Ruido en Av. Víctor Larco cuadra 14 Trujillo 2018.
- Candelario Elías, J. (2016). Implementación de WPS en el firmware NodeMCU para el ESP8266.
- Irigoyen Gallego, R. (2018). Internet de las cosas. Sistema electrónico de control basado en Arduino (Doctoral dissertation).
- Calle Zhañay, Juan Felipe (2017). Implantación de los servicios web 2.0 para la página del Departamento de Eléctrica y Electrónica de la Escuela Politécnica del Ejército. Carrera de Ingeniería en Electrónica, Redes y Comunicación de Datos. ESPE. Sede Sangolquí.
- Aguirre Rojas, Marco Esteban (2018). Emulador a escala de un sistema remoto de conducción vehicular terrestre mediante la transferencia de su dinámica. Carrera de Ingeniería en Electrónica y Telecomunicaciones. Universidad de las Fuerzas Armadas ESPE. Matriz Sangolquí.

0.14. MANUAL DE USUARIO

Configuración del Arduino IDE

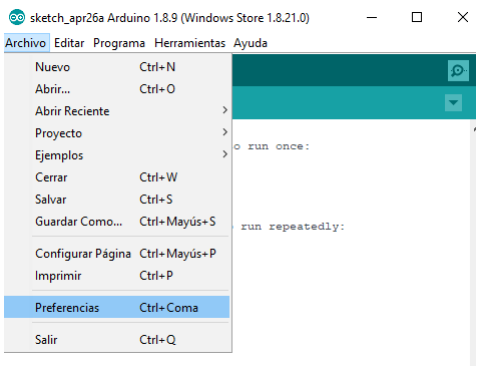
La configuración del entorno de trabajo **Arduino IDE** para poder hacer uso de la placa ESP8266 NodeMCU, para lo cual se debe seguir los siguientes pasos:

- Abrir el programa Arduino IDE

  Fuente de Descarga: `https://www.arduino.cc/en/Main/Software`

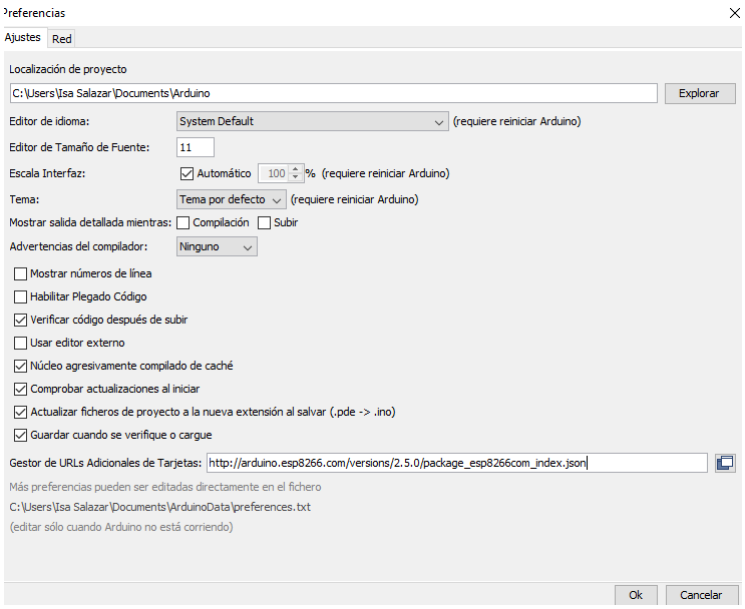
Figura 16: INICIO DE ARDUINO IDE

- Una vez en la hoja de trabajo se procede a dar click en **Archivo**, ubicado en la parte superior derecha de la hoja de trabajo, se despliega un menú y procedemos a dar click en **Preferencias**.

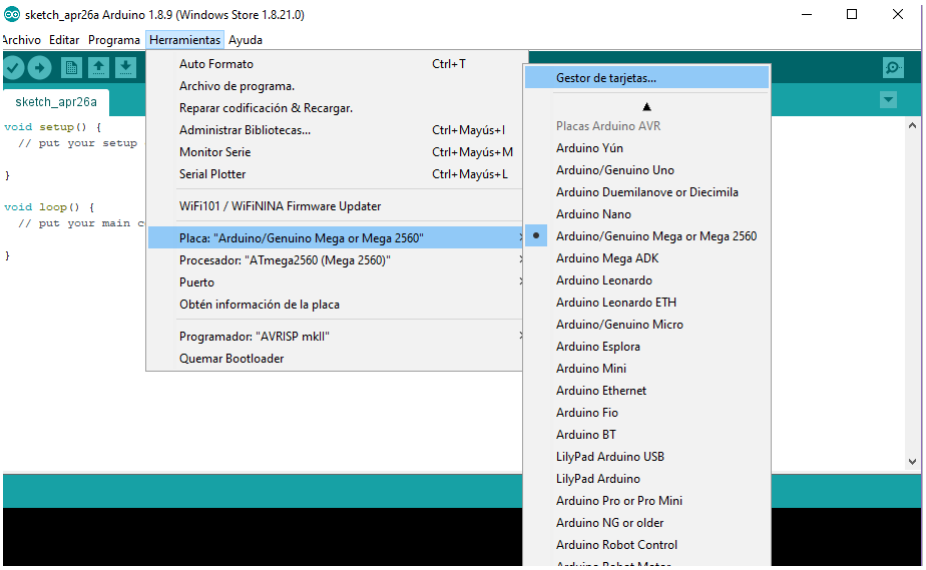


- En **Gestor de URLs Adicionales de Tarjetas** es necesario poner la siguiente url: `http://`

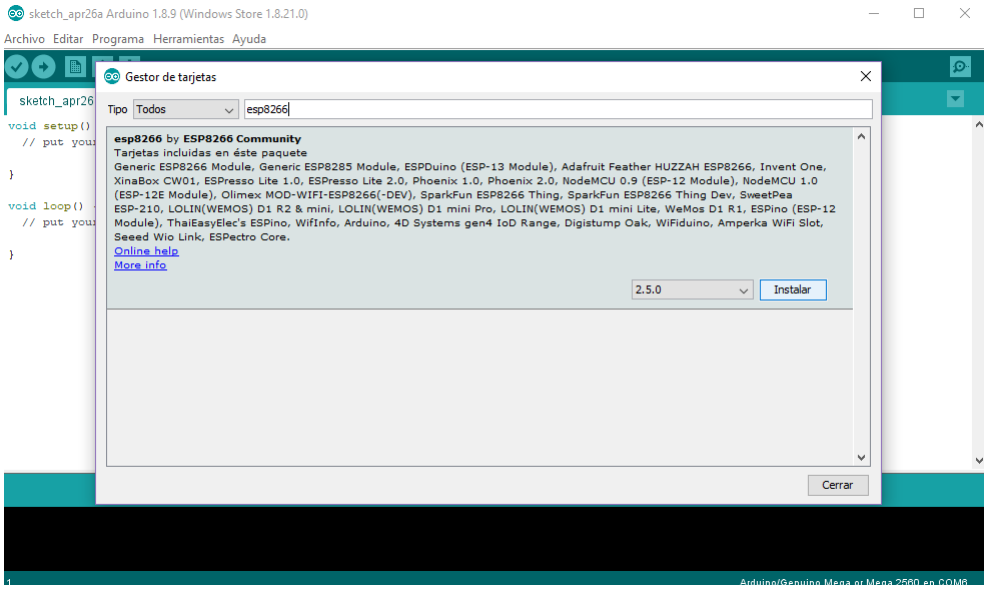
`arduino.esp8266.com/stable/package_esp8266com_index.json` y camos click en .ºk". Este nos va a permitir tener acceso a diferentes modelos de placas.



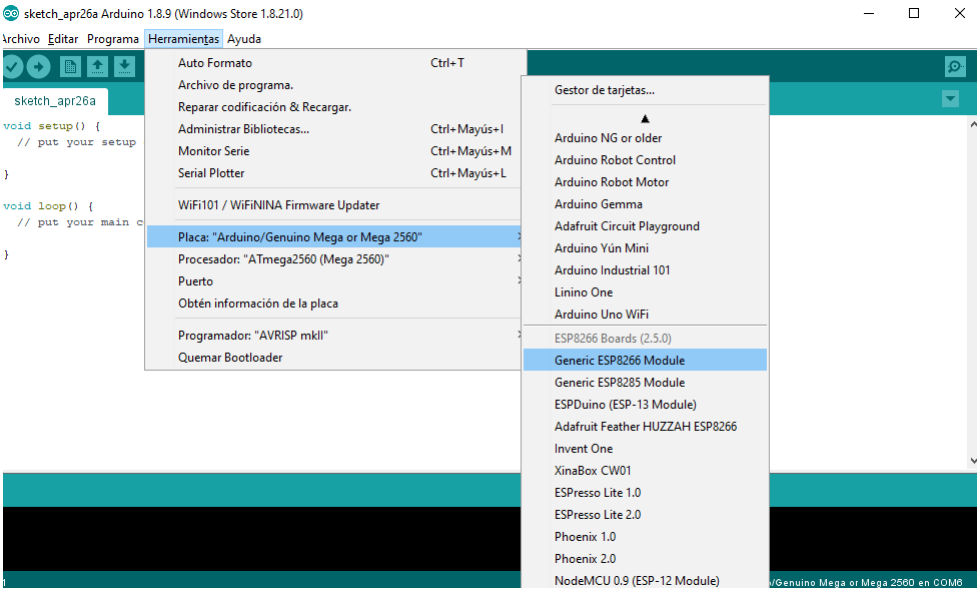
- De regreso en la hoja de trabajo damos click en **Herramientas**, en el menú que se despliega damos click en **Placa: xxxxxxxxxxx** y se vuelve a desplegar otro menú en el cual damos click en **Gestor de Tarjetas**.



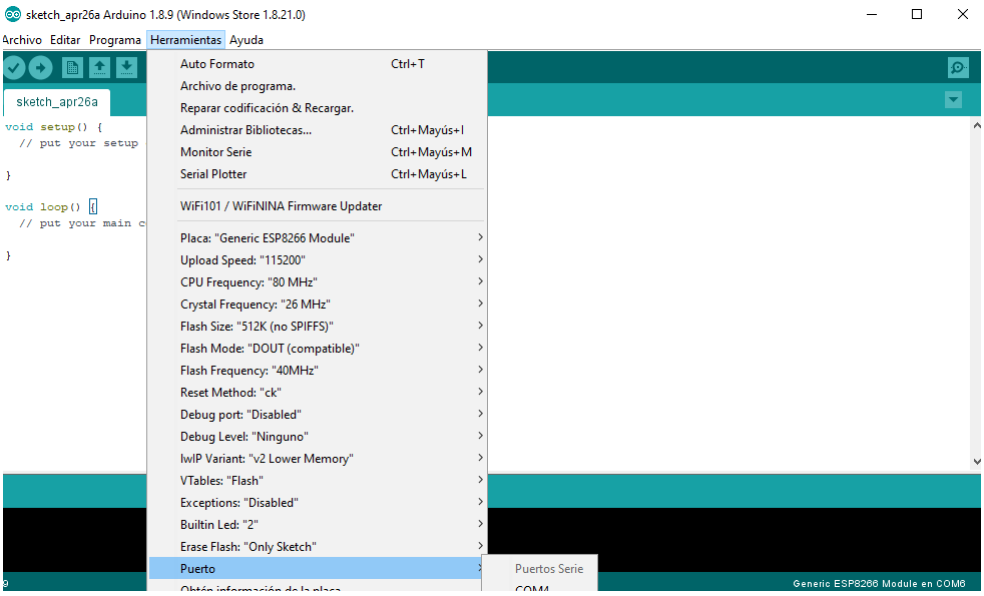
- En la ventana que se despliega buscamos **esp8266** y procedemos a instalar la versión **Community** que esté disponible para usted.



- En el mismo menú **Herramientas** damos click en **Placa: xxxxxxxxxxxx** y ibservamos que se nos despliega un menú más extenso con diferentes tios de placas **esp**. Escogemos **Generic ESP8266 Module**.



- Continuando en el menú **Herramientas** al conectar la placa podemos observar que se muestra mas opciones de configuración e incluso el puerto en el que se creo la placa.



Programacion paso a paso

Incluimos la librería ESP8266Wifi.h también declaramos contantes de tipo char que vamos a usar a lo largo de todo el programa.

Figura 17: LIBRERIAS-ESP8266wifi.h

```
#include <ESP8266WiFi.h>                    //Incluye la librería ESP8266WiFi

const char* ssid = "TP-Link_F04A";           //Indicamos el nombre de la red WiFi (SSID) a la que queremos conectarnos.
const char* password = "27081419";           //Indicamos la contraseña de de red WiFi

const char* host = "www.vermiip.es";        //Declaramos el servidor de conexión
```

Abrimos la comunicación serial a 115200 baudios porque vamos a trabajar a con wifi.
Declaramos al pin 2 (internamente está conectado con el led que existe en la placa) como pin de salida.
Mostramos un mensaje con la SSID que nos queramos conectar y empezamos la conexión con la función Wifi.begin(SSID,contraseña);
Creamos un while con la condicion WiFi.status()! = WL_CONNECTED.
WiFi.status() nos puede devolver:

- WL_CONNECTED: cuando se conecta a una red WiFi
- WL_NO_SHIELD: cuando no hay ningún escudo WiFi presente
- WL_IDLE_STATUS: es un estado temporal asignado cuando se llama a $WiFi.begin()$ y permanece activo hasta que el número de intentos caduque (lo que resulta en WL_CONNECT_FAILED) o se establece una conexión (que resulta en WL_CONNECTED).

- WL_NO_SSID_AVAIL: cuando no hay SSID disponible.
- WL_SCAN_COMPLETED: la red scan se completa.
- WL_CONNECT_FAILED: cuando falla la conexión para todos los intentos.
- WL_CONNECTION_LOST: a cuando se pierde la conexión
- WL_DISCONNECTED: cuando se desconecta de una red

El objetivo de este while es conectarse con la red por eso mientras no devuelva WL_CONNECTED va a seguir intentado y nos mostrara en consola un punto.
Desde de establer la conexión es decir salio del bucle nos mostrara un mensaje de çonnected"
El objetivo de este while es conectarse con la red por eso mientras no devuelva WL_CONNECTED va a seguir intentado y nos mostrara en consola un punto
Desde de establer la conexión es decir salio del bucle nos mostrara un mensaje de çonnected"

Figura 18: FUNCIÓN VOID SETUP

```
void setup()
{
  Serial.begin(115200);                        //Inicializamos el Puerto Serie
  Serial.println();
  pinMode(2,OUTPUT);
  Serial.printf("Connecting to %s ", ssid);     //Inicializamos la conexión Wi-Fi en modo Station
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" connected");
}
```

Creamos un objeto WiFiClient de nombre client.
Mostramos en consola a que servidor(host) nos hemos conectado.
Creamos un if con la condicion client.connect(host, 80), con esto estamos confirmando si el servidor(host) es una pagina web porque 80 es el codigo para ello.
Si la condicon es verdadera mostramos çonectado]z prendemos el led interno.
Mostramos el mensaje "[Enviando peticion]z enviamos un apeticion al servidor(host)
$client.print...$ con esta peticion obtendremos el codigo html de la pagina, para el ejercio usamos un servidor que nos informara cual es nuestra IP.

Figura 19: FUNCION VOID LOOP

```
void loop()
{
  WiFiClient client;                                    //Inicializamos el cliente (client)

  Serial.printf("\n[Conectando a %s ... ", host);      //Establecemos la conexión con el servidor
  if (client.connect(host, 80))
  {
    Serial.println("conectado]");
    digitalWrite(2,HIGH);
    Serial.println("[Enviando peticion]");             //Enviamos la petición de datos
    client.print(String("GET /") + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n" +
                 "\r\n"
                );
```

Notificamos que vamos a recibir la respuesta "[Respuesta:]"

Y la recibimos mediante estemos conectados al servidor client.connected()

Luego si tenemos datos de llegada client.available()

Los almacenamos en un variable temporal line línea por línea client.readStringUntil(′ n′); y las mostramos Cuando termine de revivir todos los datos saldrá del bucle y nos desconectaremos del servidor $client.stop()$ notificamos " $n[Desconectado]$" y apagamos el led.

Figura 20: CÓDIGO RESPUESTA

```
    Serial.println("[Respuesta:]");                   //Leemos la respuesta del servidor
    while (client.connected())
    {
      if (client.available())
      {
        String line = client.readStringUntil('\n');
        Serial.println(line);
        }
      }
    }
    client.stop();                                    //Finalizamos la conexión con el servidor
    Serial.println("\n[Desconectado]");
    digitalWrite(2,LOW);
  }
```

Si nuestra primera condicIÓn if (client.connect(host, 80)) no se cumplió es decir no es un servidor web.

Mostramos "conexion con el servidor no lograda" y nos desconectamos.

Ponemos un retardo de 10 segundos y finalizamos el código

Figura 21: PRIMERA CONDICIÓN DEL IF NO CUMPLE ENTONCES CONEXION NO LOGRADA

```
  else
  {
    Serial.println("conexion con el servidor no lograda!]");
    client.stop();
  }
  delay(10000);
}
```

0.15. HOJAS TÉCNICAS

Figura 22: ESPECIFICACIONES ESP8266

1.2. Specifications

Table 1-1. Specifications

Categories	Items	Parameters
Wi-Fi	Certification	Wi-Fi Alliance
	Protocols	802.11 b/g/n (HT20)
	Frequency Range	2.4G ~ 2.5G (2400M ~ 2483.5M)
	TX Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: -91 dbm (11 Mbps)
		802.11 g: -75 dbm (54 Mbps)
		802.11 n: -72 dbm (MCS7)
	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware	CPU	Tensilica L106 32-bit processor
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5V ~ 3.6V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	-40°C ~ 125°C
	Package Size	QFN32-pin (5 mm x 5 mm)
	External Interface	-
Software	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

Note:

The TX power can be configured based on the actual user scenarios.

Figura 23: DIAGRAMA PLACA ESP8266