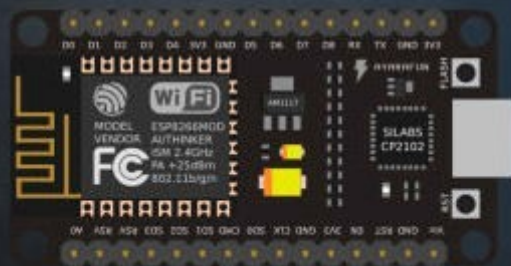


# Controle Dispositivos Remotamente com ESP8266



Fábio Souza



# Controle dispositivos remotamente con ESP8266

Fábio Souza

2018

# Aviso legal

Este eBook fue escrito con fines educativos, y con mucho esfuerzo para que quede lo más claro y didáctico posible.

El fin de este eBook es educar. El autor no garantiza que la información contenida en este eBook este completa en su totalidad y no se responsabiliza por errores u omisiones.

El autor no se responsabilizará ante ninguna persona o entidad por cualquier pérdida o daño causado o alegado a ser causado directa o indirectamente por este eBook.

Este eBook contiene ejemplos de código que puede utilizar en sus propios proyectos. Se puede acceder a todos los códigos en: <http://bit.ly/2zH7z7u>

Si encuentra algún error, o tiene sugerencias de temas o mejoras, envíeme un e-mail a: [fs.embarcados@gmail.com](mailto:fs.embarcados@gmail.com)



Esta obra se distribuye con una licencia [CC BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/)

## **Atribución-NoComercial-CompartirIgual 4.0 Internacional (CC BY-NC-SA 4.0)**

Traducido al español por:

Sergio Porporato (Técnico electrónico)

Nota: Las imágenes difieren de las originales, porque la versión del IDE y el sistema operativo son distintos a las versiones usadas en el libro. Específicamente utilice el IDE Arduino 1.8.7 y como sistema operativo, Linux Mint 19.3

Para capturar las imágenes, use la utilidad del escritorio Mate y en algunas ocasiones Shutter.

Para manipular, cortar y editar las imágenes, utilice Gthumb y Gimp

Toda la edición y posterior conversión a PDF se hizo con LibreOffice

Por favor, cualquier detalle o error en la traducción, comuníquelo al siguiente correo: [serpof@yahoo.com.ar](mailto:serpof@yahoo.com.ar)

Muchas gracias.

# Índice

<b><u>Índice</u></b>	<b><u>4</u></b>
<b><u>Sobre el Autor</u></b>	<b><u>5</u></b>
<b><u>Introducción</u></b>	<b><u>6</u></b>
<b><u>Internet de las cosas</u></b>	<b><u>7</u></b>
<b><u>ESP8266</u></b>	<b><u>9</u></b>
<b><u>ESP8266 con IDE de Arduino</u></b>	<b><u>14</u></b>
<u>IDE Arduino</u>	<u>14</u>
<u>Configuración del IDE Arduino para programar el ESP8266</u>	<u>18</u>
<u>Blink - Hello World</u>	<u>24</u>
<u>Conexión WIFI</u>	<u>26</u>
<b><u>ESP8266 como Web server</u></b>	<b><u>29</u></b>
<u>Web Server Hello</u>	<u>29</u>
<u>Accionando LEDs con webserver</u>	<u>32</u>
<b><u>MQTT (Message Queue Telemetry Transport)</u></b>	<b><u>35</u></b>
<u>Por que MQTT?</u>	<u>35</u>
<u>Broker</u>	<u>38</u>
<u>Creación de una instancia en CloudMQTT</u>	<u>39</u>
<u>MQTT DASH</u>	<u>41</u>
<b><u>Proyecto con MQTT Dash y CloudMQTT para accionamiento de una lámpara con ESP8266</u></b>	
<u>Materiales necesarios</u>	
<u>Circuito</u>	
<u>Código para tarjetas ESP8266 en Arduino</u>	
<u>Prueba de escritura en el tema con CloudMQTT</u>	
<u>Configuración de la aplicación en MQTT Dash</u>	
<u>Desafío</u>	
<b><u>Referencias</u></b>	

## Sobre el Autor



### Fábio Souza

Ingeniero con experiencia en el desarrollo de proyectos electrónicos embebidos. Hoy es director de operaciones del portal Embarcados, donde trabaja para desarrollar contenidos de electrónica, sistemas embebidos e IoT para Brasil. También se desempeña en la enseñanza electrónica y programación en Brasil. Es entusiasta del movimiento maker, de la cultura DIY y del intercambio de conocimientos, publica diversos artículos sobre electrónica y proyectos de hardware, como el proyecto Franzininho. Participó en la conferencia hacker 2018 en el Red Bull Basement. Cuando no está impartiendo conferencias, cursos o talleres, dedica su tiempo "barriendo bits" o proyectando tarjetas electrónicas.

Mis redes sociales: <https://about.me/fabio.souza>

# Introducción

Controlar dispositivos de manera remota ha sido un tema de mucho interés entre fabricantes y entusiastas.

Existen diversos proyectos con control de accionamiento remoto vía comunicación inalámbrica.

Con la facilidad de acceder a Internet a través de dispositivos móviles, junto a la disponibilidad de tarjetas con conexión WIFI, hoy es fácil controlar dispositivos remotamente, en cualquier lugar del mundo.

Este material pretende presentar los primeros pasos para trabajar con tarjetas basadas en ESP8266 para el accionamiento de dispositivos remotos, ya sea a través de un navegador web o a través de una aplicación móvil que publica en un broker MQTT.

También se presentarán los conceptos básicos de IoT, MQTT y aplicaciones prácticas. Al final podremos crear aplicaciones y controlar dispositivos en casa.

¡Manos a la obra!



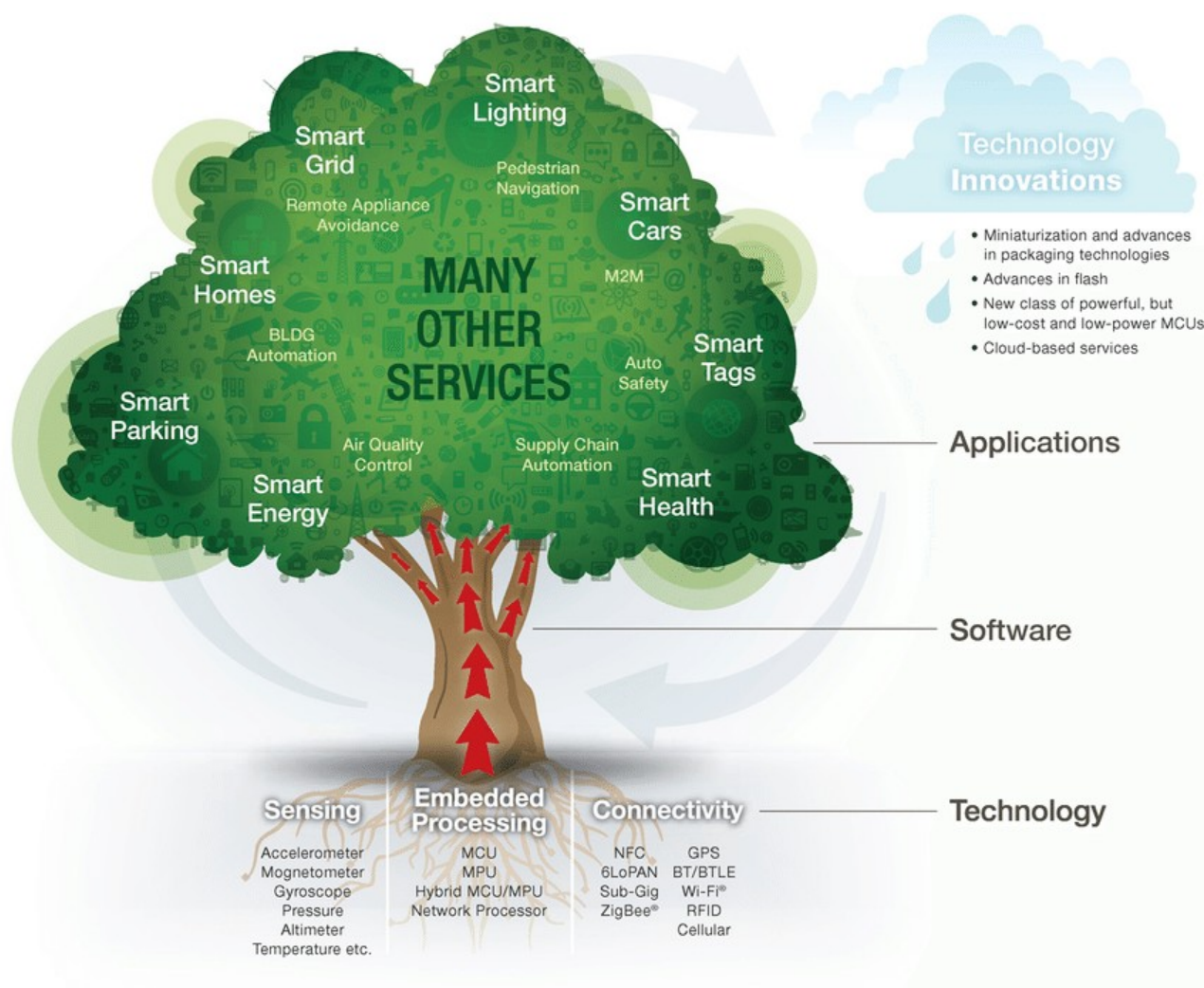
## Internet de las cosas

La Internet de las cosas, o en inglés Internet of Things, es un término que se ha popularizado en los últimos años. Pero ese término no es tan nuevo, Kevin Ashton lo usó en 1999 durante una presentación sobre una solución de RFID sobre la que estaba trabajando. Además, Mark Weiser, autor de “The computer of the 21 st Century”, trajo el concepto de ubicuidad a principios de los años 90.

La Internet de las cosas es una red de objetos físicos que poseen tecnología embebida para comunicar, captar señales e interactuar consigo mismo o con el ambiente externo.

Los sensores son llamados "cosas" y pueden ser cualquier objeto, poseyendo su propia IP y con capacidad de enviar y recibir datos por una red.

El gráfico siguiente da una noción de Internet de las cosas:



Fuente:

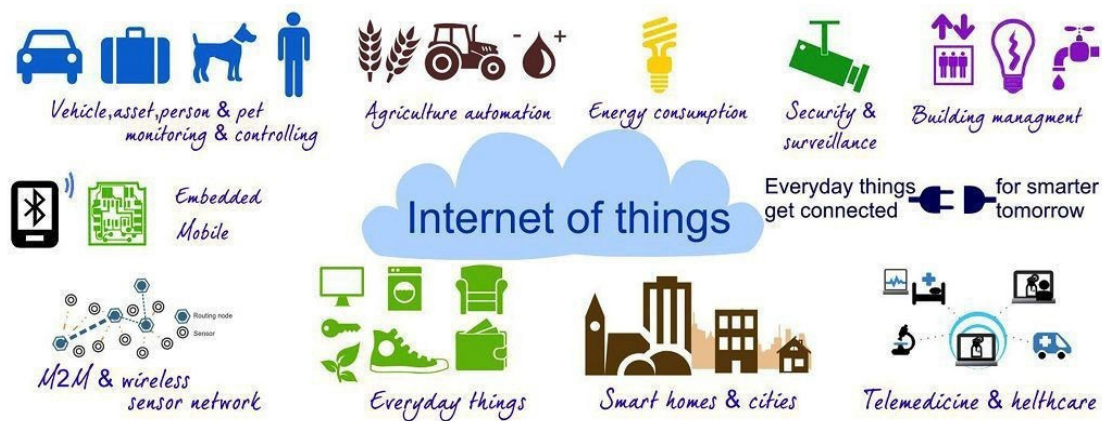
[https://www.researchgate.net/figure/The-IoT-Different-Services-Technologies-Meanings-for-Everyone-77\\_fig6\\_278798179](https://www.researchgate.net/figure/The-IoT-Different-Services-Technologies-Meanings-for-Everyone-77_fig6_278798179)

En las raíces tenemos los sistemas embebidos, o sea, la "cosa" que a través de sensores captura las magnitud del ambiente y las envía a través de algún tipo de comunicación (generalmente inalámbrica).

En el tronco tenemos el software que hace el tratamiento de datos y los envía a la nube.

En la nube tenemos las aplicaciones. El mismo dato puede utilizarse en diferentes servicios.

Hay una infinidad de aplicaciones para Internet de las cosas, en diversas áreas:



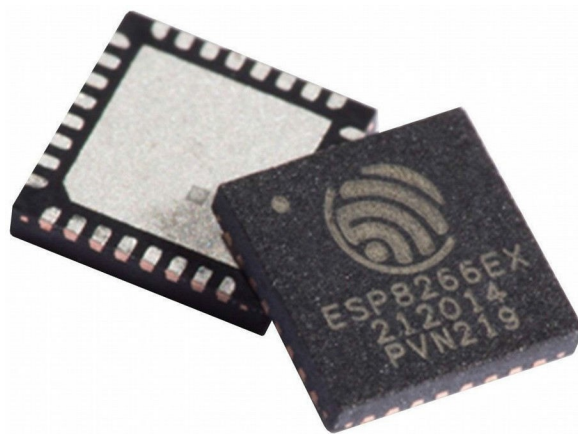
Ya se están creando soluciones para IoT por empresas conocidas y muchas otras nuevas.

Ver [The 2018 Internet of Things Landscape](#)



# ESP8266

El ESP8266 es un SoC<sup>1</sup> (System on a chip) de la empresa china Espressif. Construido alrededor del procesador Tensilica Xtensa LX3, incluye pila TCP / IP integrada que posibilita el acceso a la red WIFI. Creada originalmente para ser un adaptador WIFI, rápidamente se hizo independiente debido a sus recursos, desarrollo del firmware por la comunidad (a pesar de la falta de documentación inicial) y principalmente por el bajísimo costo.

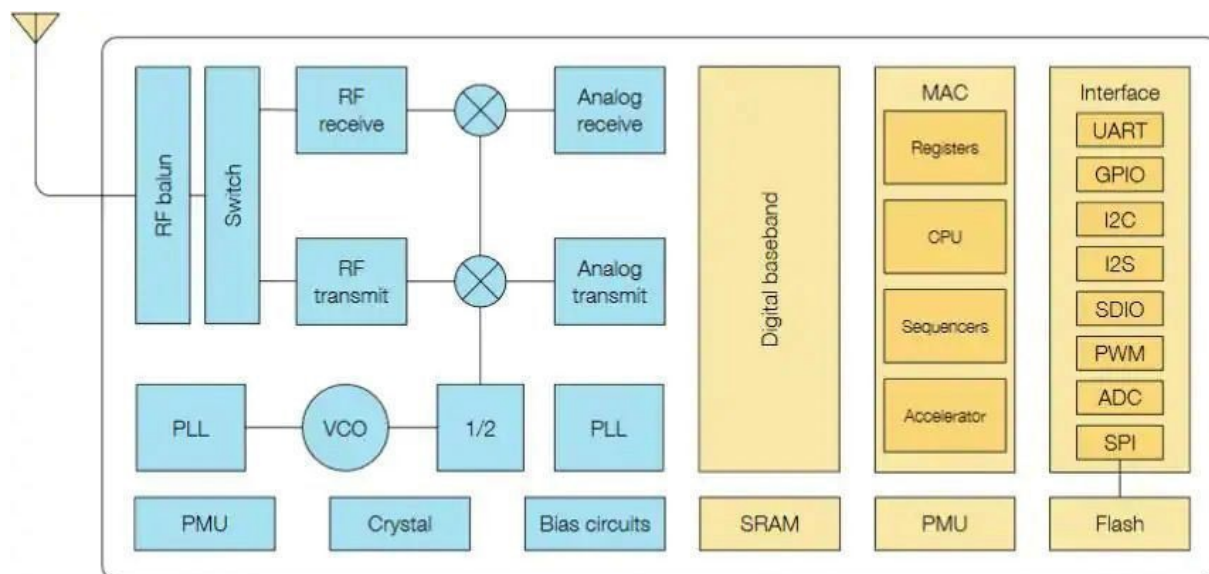


## Características del SoC ESP8266:

- Integrated low power Tensilica L106 32-bit MCU
- 802.11b/g/n
- Wi-Fi 2.4GHz, WPA/WPA2 support
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- Interfaces:SDIO 2.0, SPI, UART, I2C, I2S, PWM, GPIO
- FCC, CE, TELEC, Wi-Fi Alliance, and SRRC certified
- 32-pin QFN (5x5mm)

<sup>1</sup>Un sistema en chip o SoC, describe la tendencia cada vez más frecuente de usar tecnologías de fabricación que integran todos o gran parte de los módulos que componen un computador o cualquier otro sistema informático o electrónico en un único circuito integrado o chip. (Wikipedia)

Diagrama de bloques:



Hoy en día, el chip se utiliza en una infinidad de tarjetas y se ha convertido en el puerto de entrada para aplicaciones IoT:



Fuente: <https://pyliaorachel.github.io/tutorial/hardware/arduino/2017/04/13/esp8266-with-arduino-trials-and-errors.html>

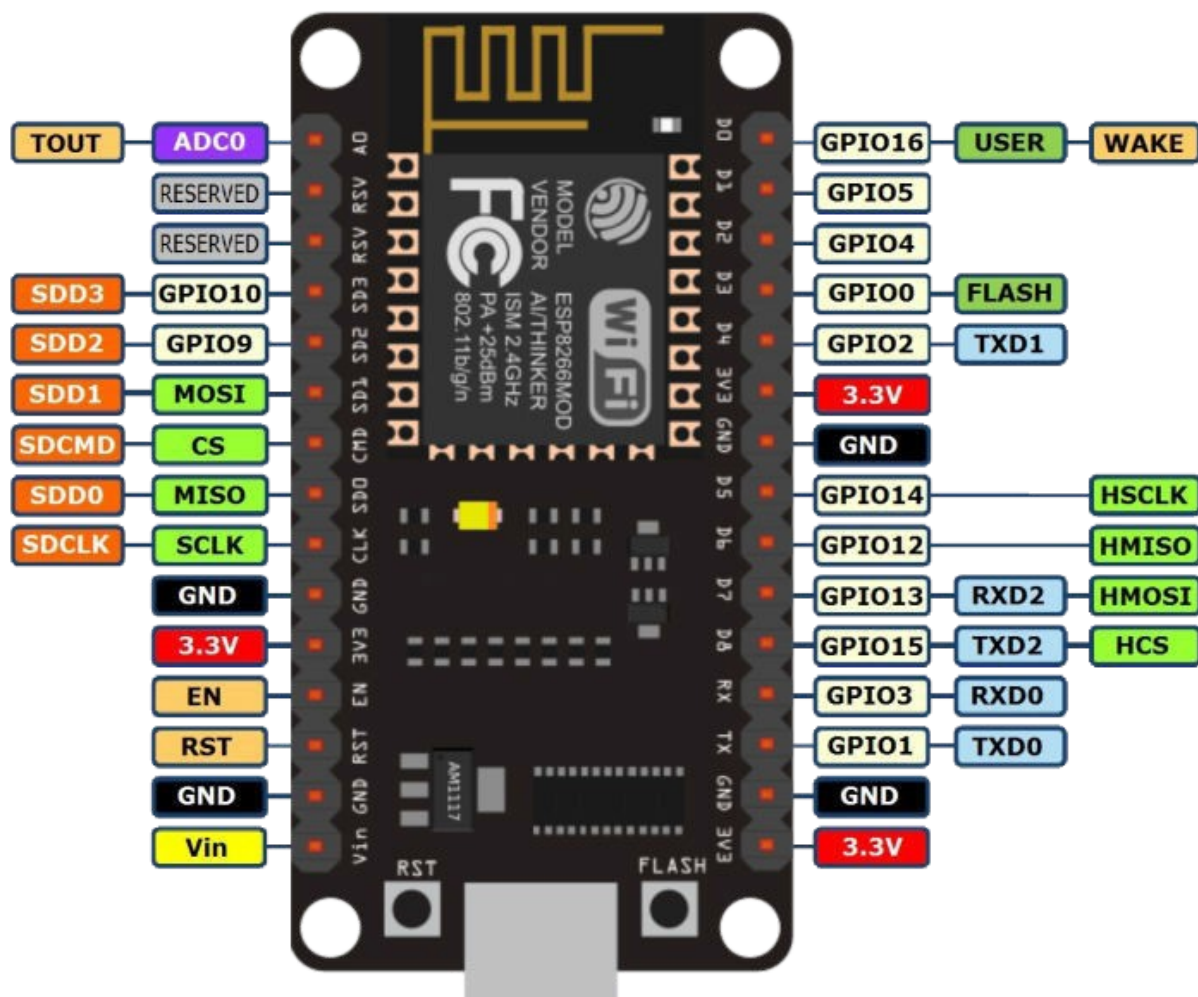
El módulo ESP-01 se hizo popular debido a su bajo costo, pero la dificultad de usarlo de manera autónoma, se convirtió en un obstáculo para los principiantes. Afortunadamente, otras tarjetas fueron creadas, como la famosa nodeMCU (que usaremos en este material de estudio)



Fuente: <https://en.wikipedia.org/wiki/NodeMCU>

El nodeMCU creado en base al módulo ESP 12E, facilita el proceso de programación del ESP8266 por poseer de manera integrada , el convertidor serie-USB , regulador de tensión y pines de E / S para su conexión en protoboard. Hoy se programa una tarjeta ESP8266 directamente en la IDE del Arduino, como si fuera un arduino más.

La siguiente figura muestra el pinout de la tarjeta:

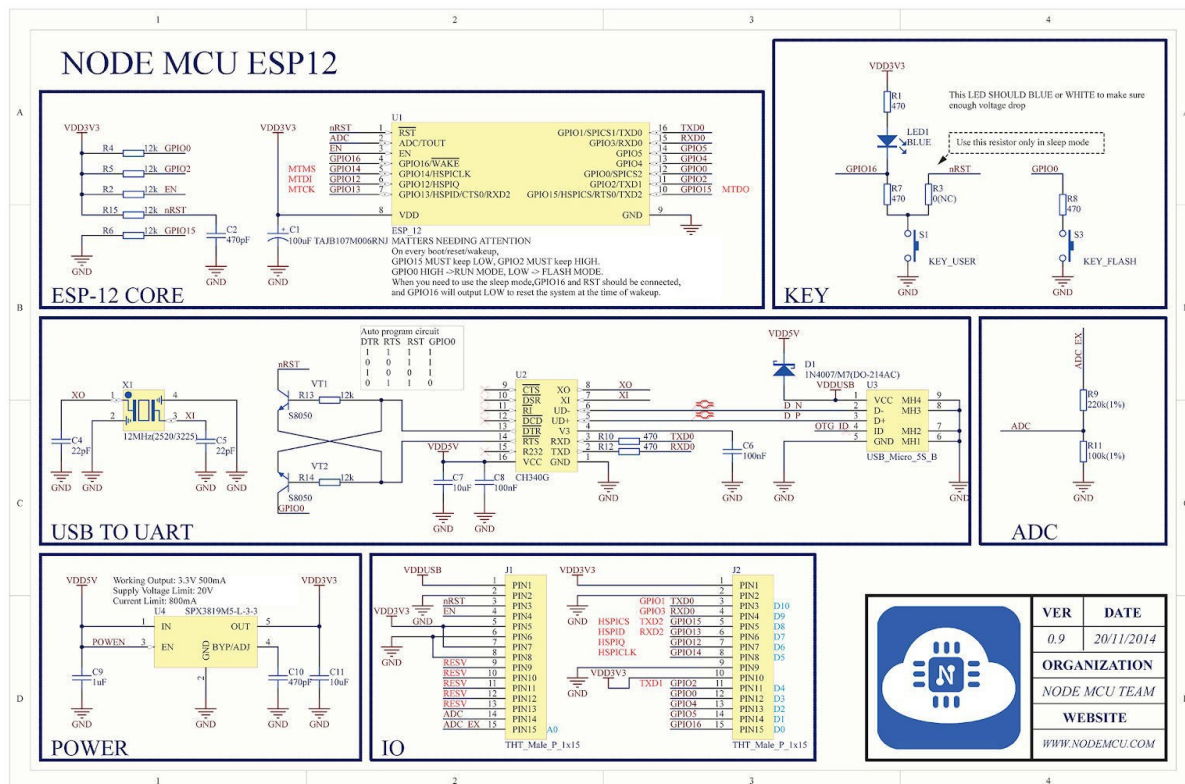


Guardar este pinout para recordar los pines cuando se programe la tarjeta.

La tarjeta tiene un LED de uso general conectado al GPIO\_16, un LED en el módulo ESP 12E (GPIO\_2), y botones RST (Reset) y FLASH (grabación del programa).



## Esquema de la tarjeta:



Hay versiones de la tarjetas que utilizan diferentes chips para comunicación serie. A continuación, los links para acceder a la descarga de los controladores de los dos chips más comunes:

- [CH340](#)
- [CP2102](#)

El proyecto es de hardware abierto y se puede acceder a los archivos en github: <https://github.com/nodemcu/nodemcu-devkit-v1.0>

# ESP8266 con el IDE de Arduino

En esta sección, se verá como descargar, instalar y preparar el IDE de Arduino para programar tarjetas con el ESP8266. Utilizaremos el IDE y las facilidades del Arduino para programar el ESP8266, junto con las bibliotecas existentes. Al final de este tema tendremos un entorno listo para trabajar en los proyectos.

## IDE de Arduino

El entorno de desarrollo Arduino es muy sencillo de usar. Basado en el procesamiento, facilita la programación de las tarjetas Arduino. Por ser una plataforma de código abierto, posibilita la integración de nuevas tarjetas, como el caso del ESP8266 que usaremos a continuación. Este es el repositorio de integración del ESP8266 en Arduino:

<https://github.com/esp8266/Arduino>

El IDE Arduino es multiplataforma y necesita JAVA instalado en el ordenador.

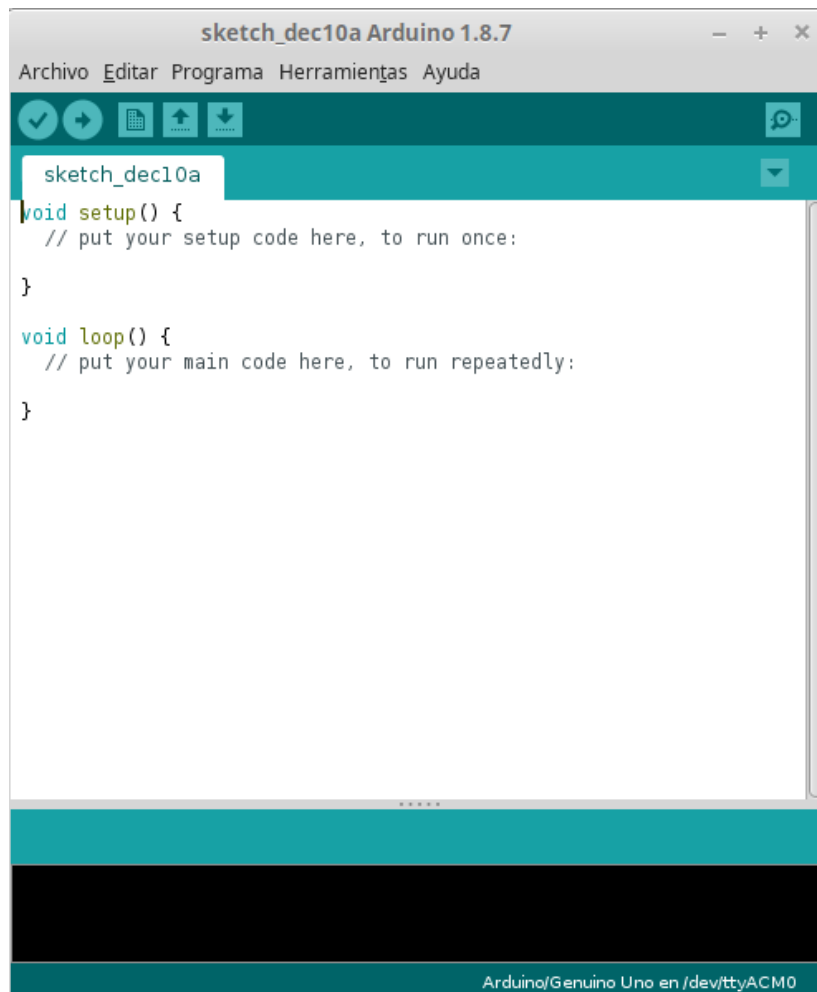


Se descarga gratuitamente del [sitio web de Arduino](https://www.arduino.cc/en/Main/Software) , donde podemos elegir la versión correcta para nuestro sistema operativo.

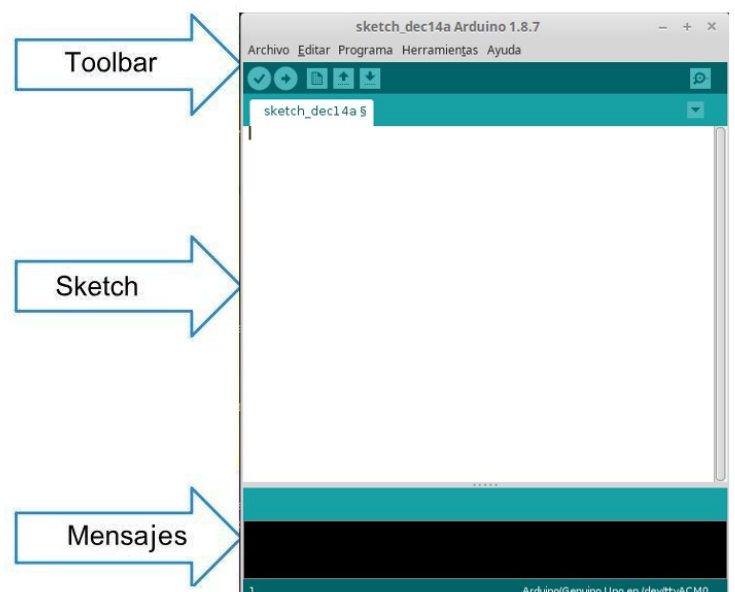
Después de instalar el IDE en su equipo, se ejecuta.



Cuando se abre el IDE del Arduino, aparecerá una imagen similar a la siguiente:



El IDE se divide en tres partes:  
La barra de herramientas en la parte superior, el código o la ventana Sketch en el centro, y la ventana de mensajes en la base, como se muestra en la figura.



En la barra de herramientas hay una guía, o un conjunto de guías, con el nombre del sketch. A la derecha hay un botón que habilita el monitor de serie. En la parte superior hay una barra de menús, con los elementos File, Edit, Sketch, Tools y Help. Los botones de la barra de herramientas proporcionan acceso rápido a las funciones más utilizadas dentro de estos menús.

A continuación se identifican los iconos de acceso directo del IDE:

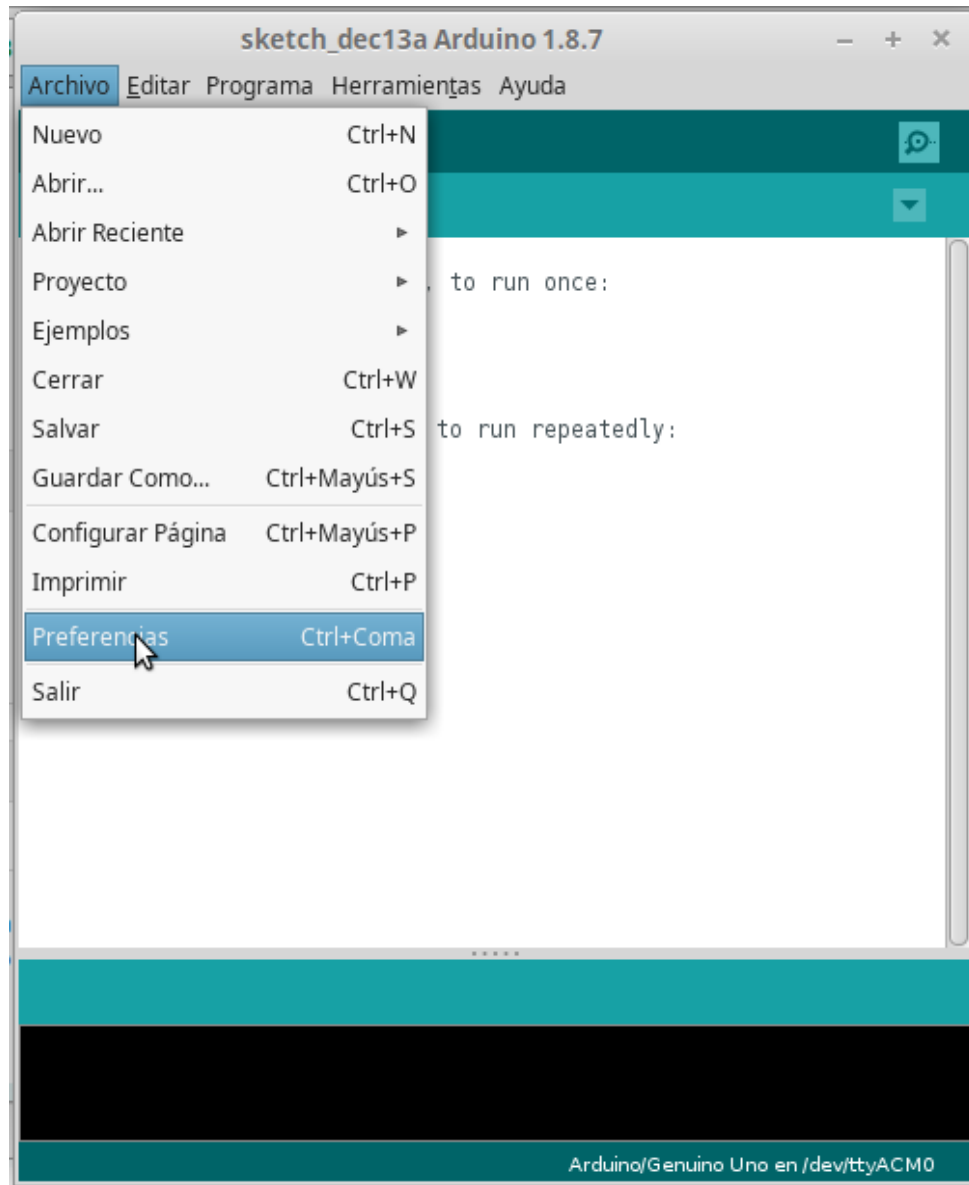
- **Verificar**
  - Comprueba si hay un error en el código introducido.
- **Subir**
  - Compila el código y graba en la tarjetas Arduino si esta conectada correctamente.
- **Nuevo**
  - Crea un nuevo sketch en blanco.
- **Abrir**
  - Abre un sketch, presente en el sketchbook.
- **Salvar**
  - Guarda el sketch activo
- **Monitor Serie**
  - Abre el monitor serie.

Los demás comandos presentes en la barra de menús se pueden consultar a



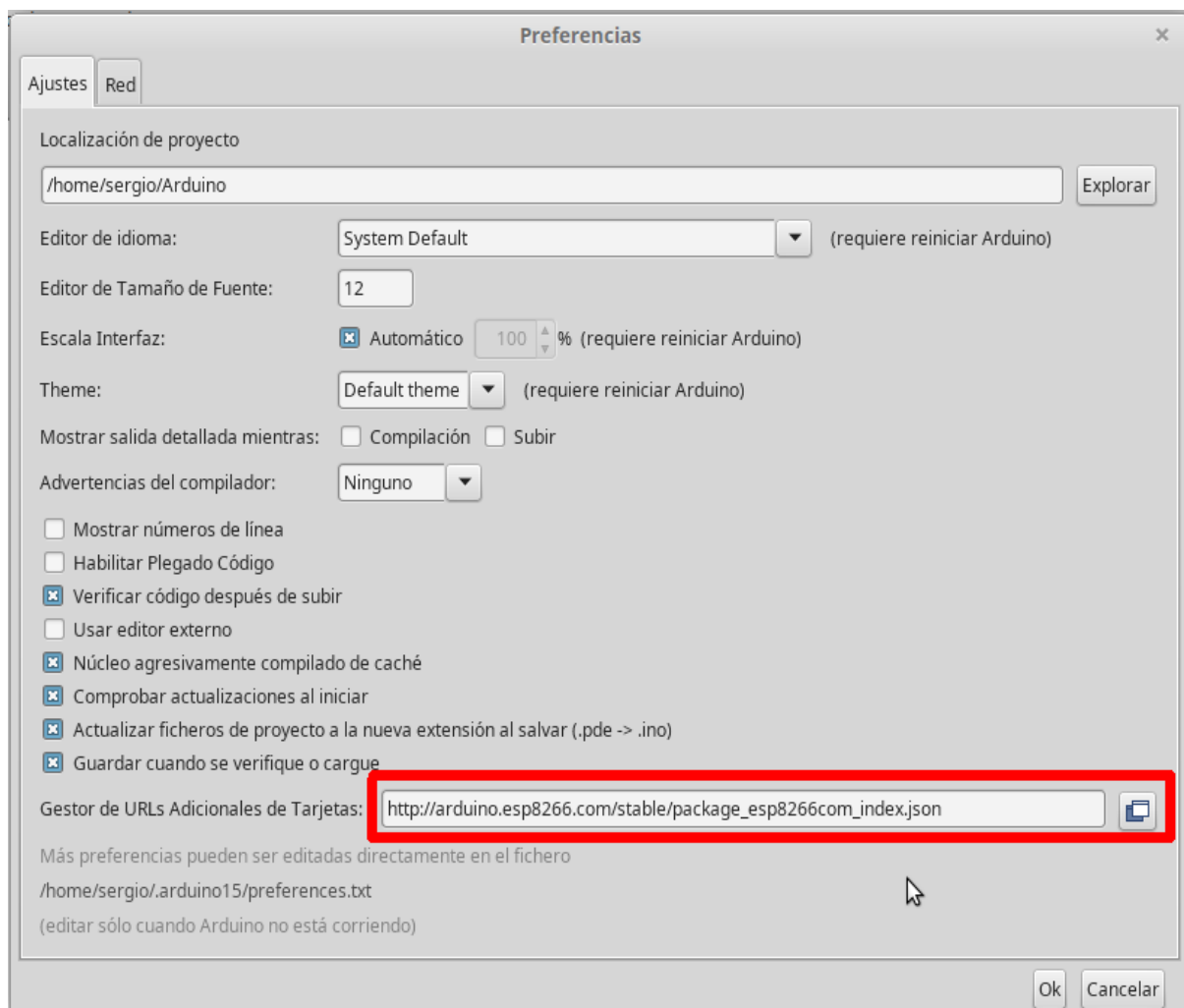
través del menú **Ayuda > Entorno**.

En el **IDE** acceder a **Archivo> Preferencias**



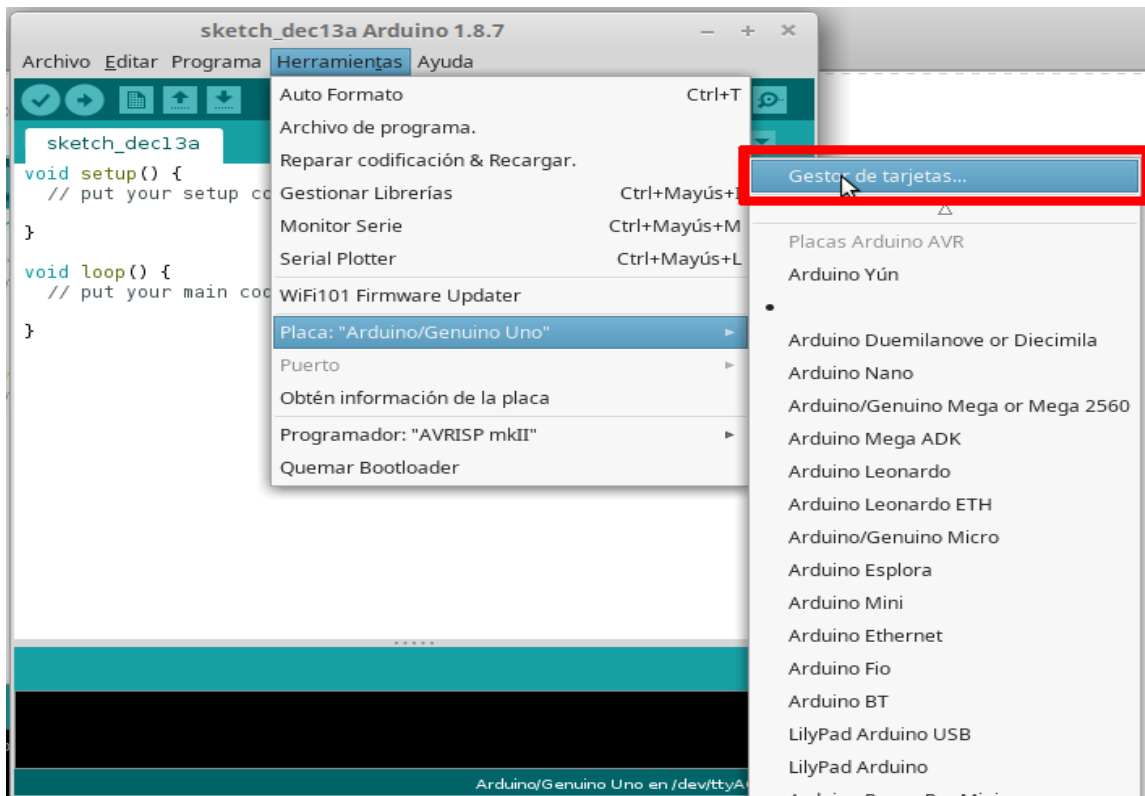
En la ventana de configuración de preferencias, introducir la siguiente URL en el campo "URL adicionales de los gestores de tarjetas":

[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)



Hacer clic en Aceptar.

Ahora acceder a **Herramientas> tarjetas> Gestores de tarjetas:**



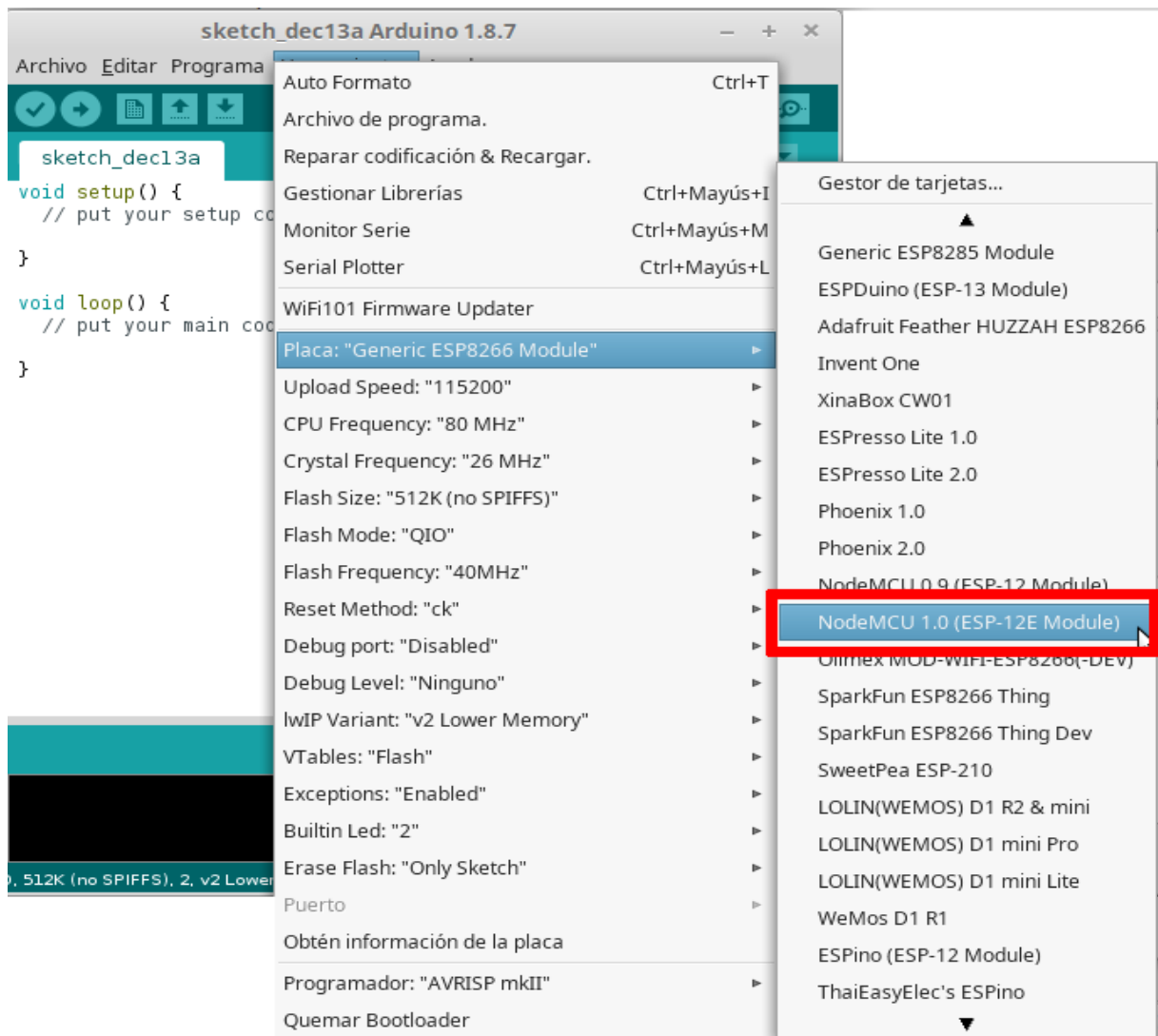
Esperar unos segundos para actualizar los índices de plataforma. Buscar 8266. Encontrar la opción "esp8266 by ESP8266 Community" y hacer clic en instalar:



Esperar a que finalice la instalación y, a continuación, hacer clic en Cerrar.

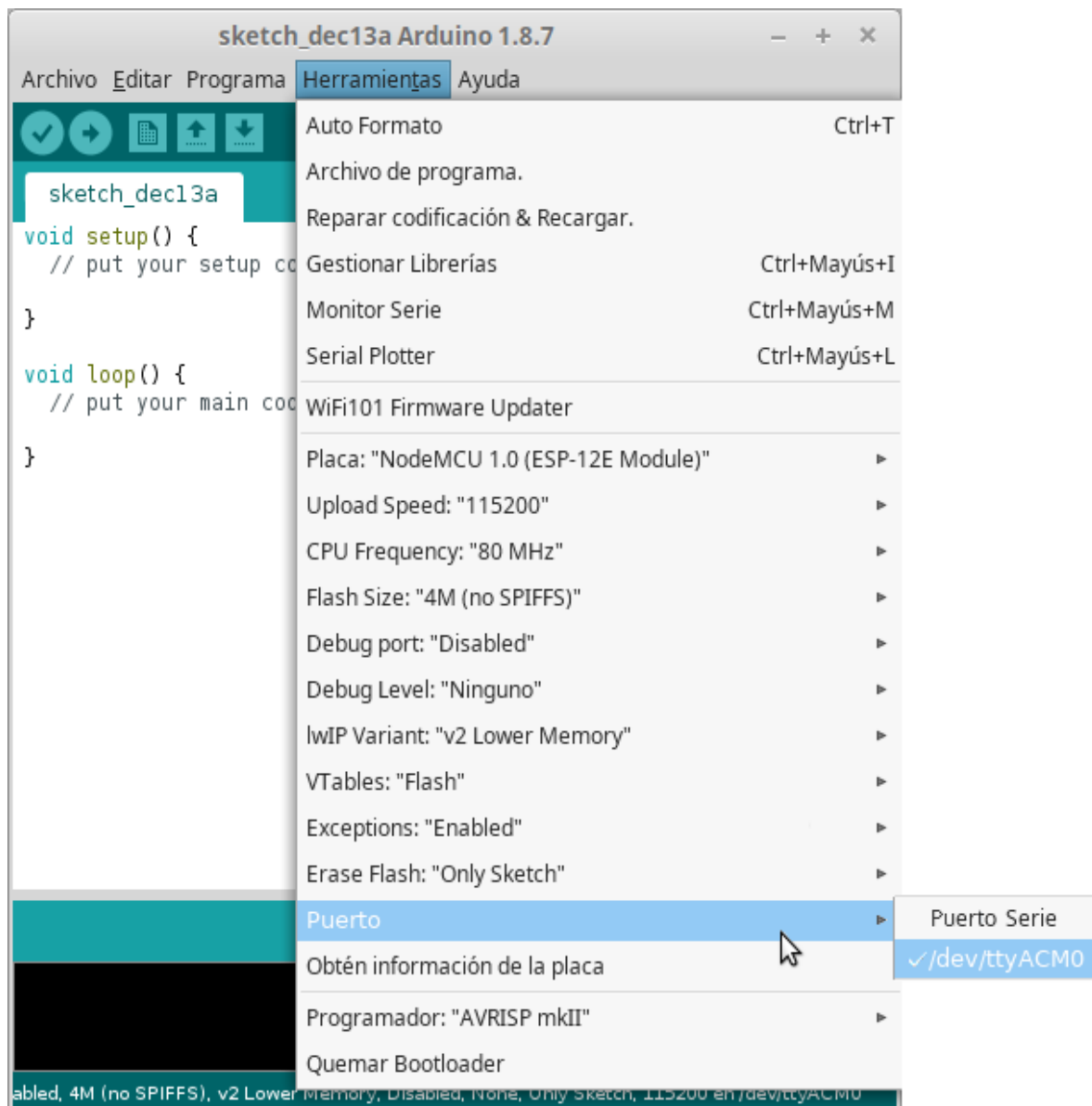
**Nota: Es necesario tener acceso a Internet**

Ahora vemos la lista de tarjetas ESP8266 instaladas en el IDE. Para nuestro tutorial, vamos a usar el NodeMCU 1.0 (ESP 12E Module):



Conectar la tarjeta al ordenador y seleccionar el puerto de comunicación:





Listo, el IDE está configurado para programar la tarjeta NodeMCU.

Ahora vamos al primer ejemplo.

## Blink - Hello World

Para comprobar si la configuración del IDE se ha realizado correctamente y si la tarjeta está funcionando, vamos a hacer un blink para parpadear el LED de la tarjetas en intervalos de 1 segundo.

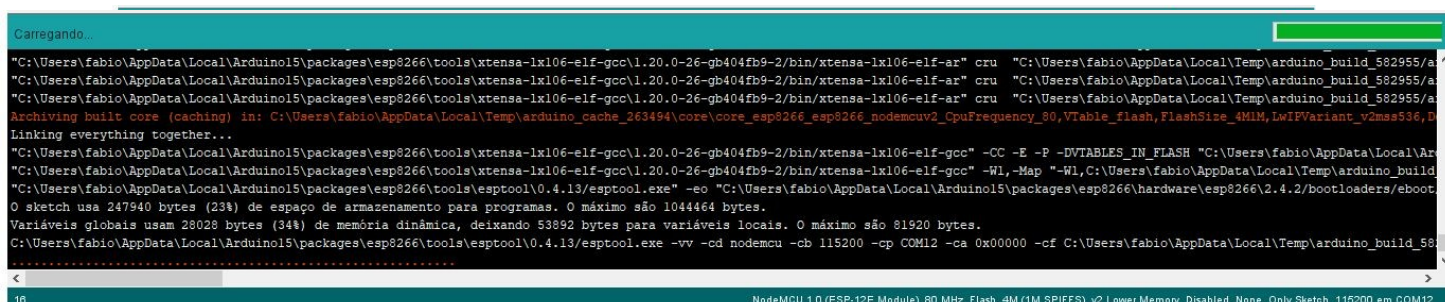
Escribi el siguiente código:

```
/*  
Workshop - Controle dispositivos remotamente con ESP8266  
Por: Fábio Souza  
*  
Ejemplo 1 - Blink LED  
Prende el LED onboard en intervalos de 1 segundo.  
Este ejemplo sirve para verificar la configuración del IDE y el funcionamiento  
de la placa  
*/  
  
const byte LED = 2;  
void setup () {  
  pinMode(LED, OUTPUT);      //Configura el pin del LED como salida  
  
}  
  
void loop() {  
  digitalWrite(LED, LOW);    // LED apagado  
  delay(1000);               // Espera un segundo  
  digitalWrite(LED, HIGH);   // LED encendido  
  delay(1000);               // Espera un segundo  
  
}
```

Después de escribir el código, hacer clic en el botón de carga  
Se iniciará el proceso de compilación y, a continuación, la carga.



Acompañe el proceso a través de la barra de carga y de los mensajes.  
Después de unos segundos el proceso se finalizará y aparecerá el mensaje  
**"Cargado"**:



```
Carregado.
..... [ 97% ]
..... [ 100% ]
starting app without reboot
  espcomm_send_command: sending command header
  espcomm_send_command: sending command payload
  espcomm_send_command: receiving 2 bytes of data
closing bootloader
  flush start
  setting serial port timeouts to 1 ms
  setting serial port timeouts to 1000 ms
  flush complete
```

10 NodeMCU 1.0 (ESP-12E Module), 80 MHz, Flash, 4M (1M SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 em COM12

Compruebe que el LED parpadee en la tarjetas.

Felicitaciones! Se configuro el IDE correctamente para programar las tarjetas. Ahora vamos al siguiente nivel!

## Conexión WIFI

Para conectar nuestra tarjeta al WIFI vamos a usar la biblioteca ESP8266Wifi.h.

Introducir el siguiente código. Hay que actualizar los campos "ssid" y "contraseña", con los valores de la red que desea conectar:

```
/*
 *Workshop - Controle dispositivos remotamente con ESP8266
 *Por: Fábio Souza
 *
 *Ejemplo 2 - WIFI
 *Conecta la tarjetas a una RED WIFI
 */

#include <ESP8266WiFi.h>

//configuración de red
const char* ssid = "ssid"; const

void setup() {
  Serial.begin(115200); //configura comunicación serie
  delay(10);
```

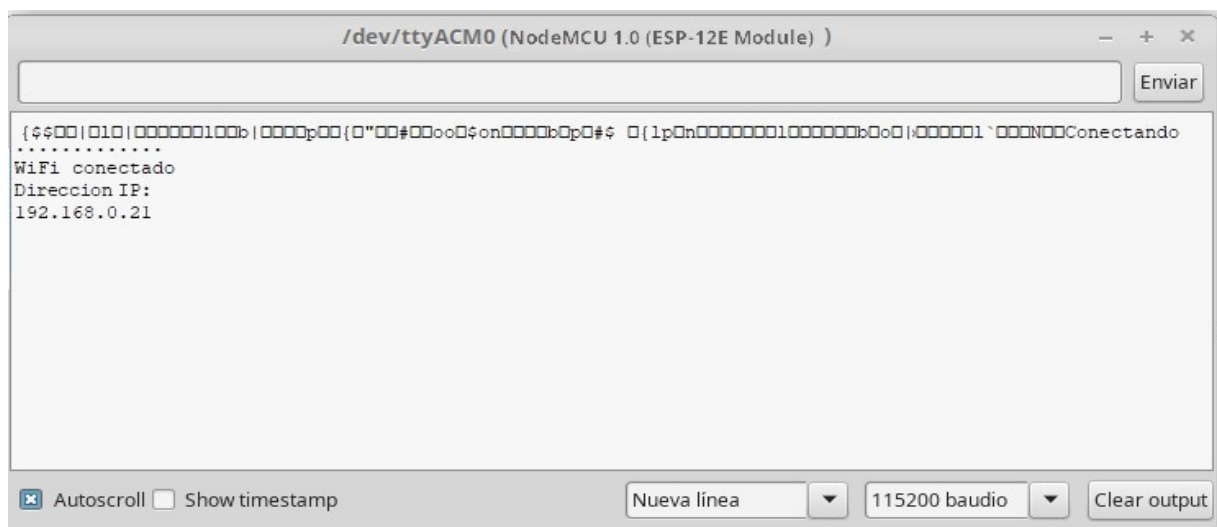
```
// mensaje de debug serial
Serial.print("Conectando a la red
                ")
; Serial.println(ssid);

// Iniciando la conexión WiFi
WiFi.begin(ssid, senha);

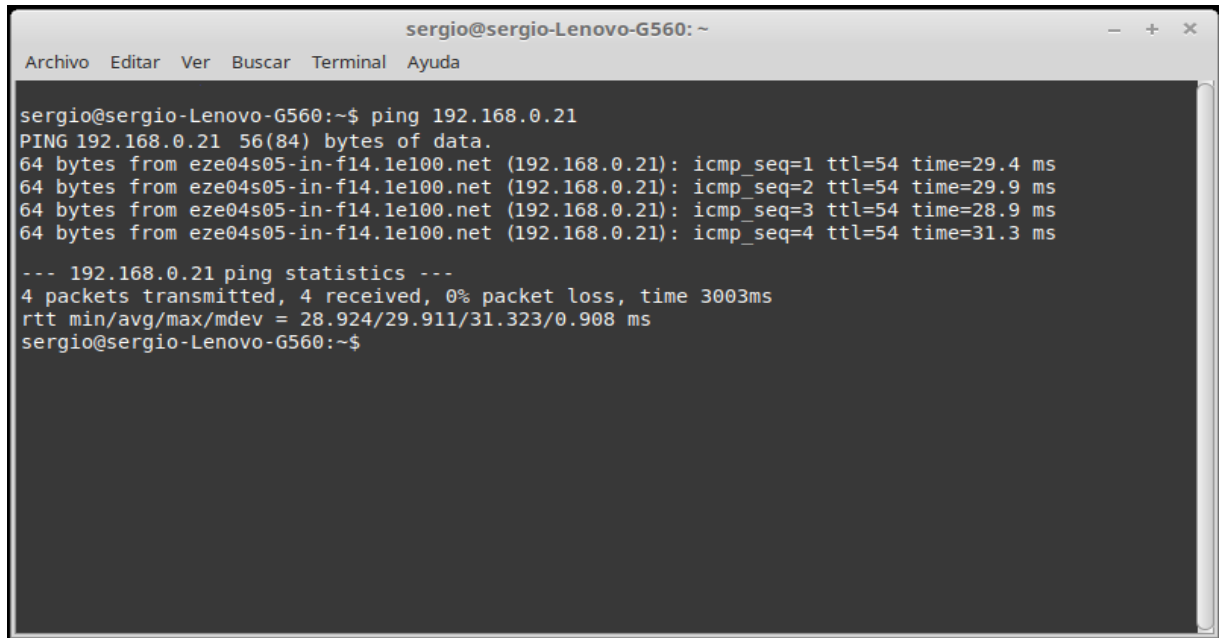
// espera la conexión WIFI
while (WiFi.status() !=
      WL_CONNECTED) { delay(500);
      Serial.print(".");
    }

//mensaje de conectado
Serial.println("");
Serial.println("WiFi
conectado");
Serial.println("Dirección IP:
");
Serial.println(WiFi.localIP())
;
}
```

Abrir el terminal Serie y ver si la tarjeta se conectó a la red Wifi:



Ahora se hace ping en la dirección IP a la que se ha conectado:



A screenshot of a terminal window titled "sergio@sergio-Lenovo-G560: ~". The window has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The terminal shows the execution of a ping command to 192.168.0.21. The output displays four successful ping responses with varying times and a summary of the statistics.

```
sergio@sergio-Lenovo-G560:~$ ping 192.168.0.21
PING 192.168.0.21 56(84) bytes of data.
64 bytes from eze04s05-in-f14.1e100.net (192.168.0.21): icmp_seq=1 ttl=54 time=29.4 ms
64 bytes from eze04s05-in-f14.1e100.net (192.168.0.21): icmp_seq=2 ttl=54 time=29.9 ms
64 bytes from eze04s05-in-f14.1e100.net (192.168.0.21): icmp_seq=3 ttl=54 time=28.9 ms
64 bytes from eze04s05-in-f14.1e100.net (192.168.0.21): icmp_seq=4 ttl=54 time=31.3 ms

--- 192.168.0.21 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 28.924/29.911/31.323/0.908 ms
sergio@sergio-Lenovo-G560:~$
```

Pronto aprendemos cómo conectar la tarjeta al WIFI, ahora vamos a crear una aplicación para accionar dispositivos.



# ESP8266 como Web server

Ahora vamos a hacer algo más agradable con el ESP8266.

Un servidor web es un dispositivo conectado a la red que almacena y entrega (sirve) archivos. Los clientes pueden solicitar tal archivo u otro dato, y el servidor enviará los datos / archivos correctos de nuevo al cliente. Las solicitudes se realizan mediante HTTP.

HTTP o Protocolo de transferencia de hipertexto es un protocolo basado en texto que se utiliza para comunicarse con servidores (web). Hay varios métodos de solicitud HTTP, los más comunes son GET y POST.

## Web Server Hello

Ahora vamos a configurar nuestra tarjeta con ESP8266 como web server y controlar salidas digitales a través de un navegador web.

Como primer ejemplo vamos a crear una página web simple para mostrar información en el navegador:

```
/*
 * Workshop - Controle dispositivos remotamente con ESP8266
 * Por: Fábio Souza
 *
 * Ejemplo 3 - webserver hello
 * Crea un servidor web en la tarjeta
 */

#include

<ESP8266WiFi.h> int x

= 0;
```

```

const char* ssid    = "ssid";
const char* senha = "senha";

WiFiServer server(80);//instancia el servidor en el puerto 80

void setup() {
    Serial.begin(115200); //configura comunicación serie
    delay(10);

    // mensaje de debug serial
    Serial.print("Conectando a la red
                  ")
    ; Serial.println(ssid);

    // Iniciando de la conexión WiFi
    WiFi.begin(ssid, senha);

    // espera la conexión WIFI
    while (WiFi.status() != WL_CONNECTED)
        { delay(500);
          Serial.print(".");
        }

    server.begin();
    Serial.println("Servidor inicializado!");

    //mensaje de conectado
    Serial.println("");
    Serial.println("WiFi conectado");
    Serial.println("Dirección IP: ");
    Serial.println(WiFi.localIP());
}

void loop() {
    // Comprueba si el cliente está
conectado    WiFiClient client =
server.available(); if (!client) {
    return;
}

    // Espera datos del cliente
    Serial.println("cliente");
    while (!client.available())
    {
        delay(1);
    }

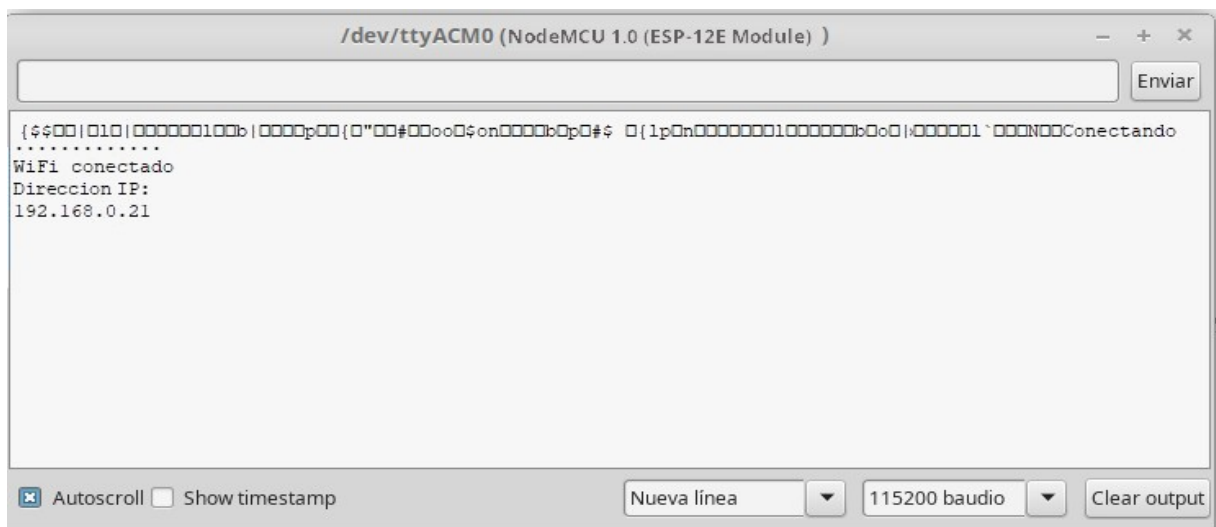
    String txt = ""; //string para montar el html

    txt += "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n\r\n<html><meta http-equiv='refresh' content='3'>\r\n\r\n";
    txt += "<h1>Hola Makers!</h1>";
}

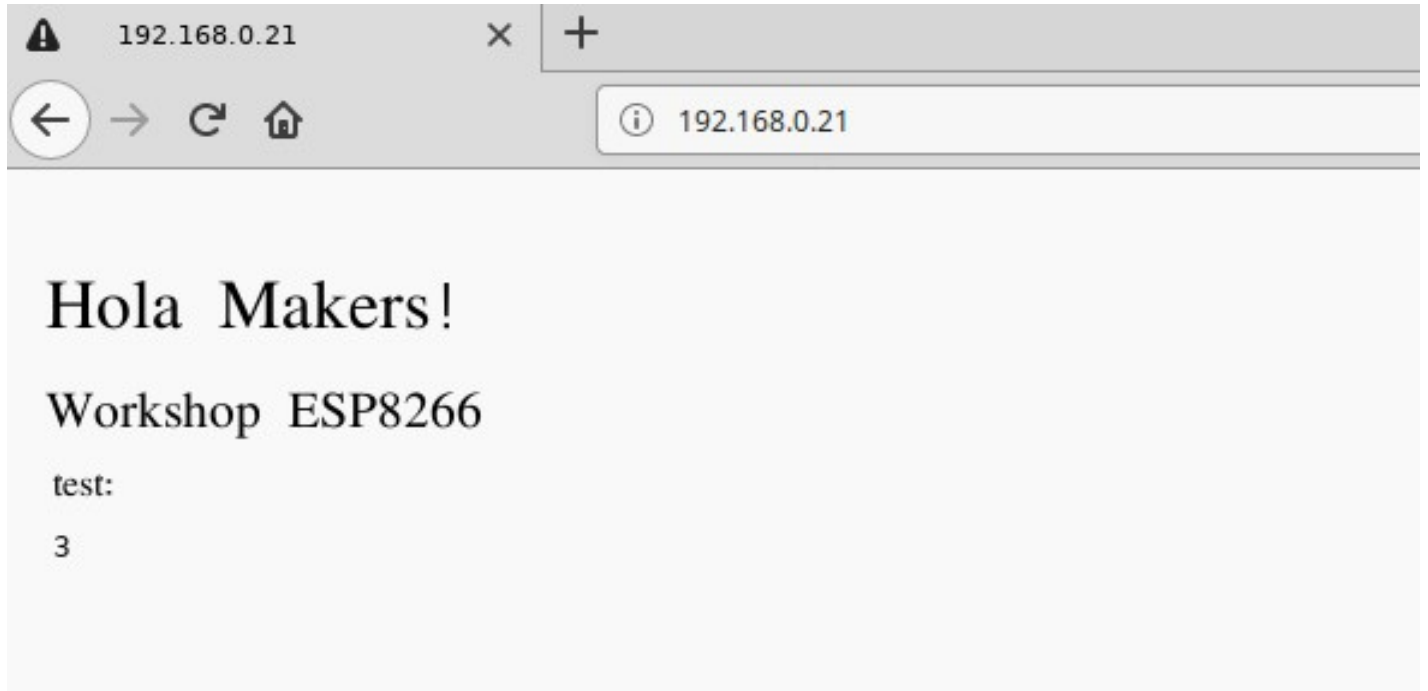
```

```
txt += "<h2>Workshop
ESP8266</h2>"; txt +=
"<h3>test:</h3>";
txt += "</html>\n";
client.print(txt)
;
client.print(x); x+=
+;
```

Abrir el terminal serie y ver que IP asignó a la tarjeta:



Entonces, introducir esa IP en el navegador del ordenador (conectado a la misma red que la tarjeta).



## Accionando los LED con webserver

Ahora vamos a crear un botón para encender y apagar el LED de la tarjeta:

```
/*  
 *Workshop - Controle dispositivos remotamente com ESP8266  
 *Por: Fábio Souza  
 *  
 *ejemplo 4 - webserver IO  
 *Acciona I/O a través del browser  
 */  
  
#include <ESP8266WiFi.h>  
  
#define OUT1 2  
  
boolean statusOUT1 = LOW;  
  
//configuración de red  const  
  
WiFiServer server(80); //instancia el servidor en el puerto 80  
  
void setup() {
```

```

Serial.begin(115200); //configura comunicación serie
delay(10);

//configura pin de salida
pinMode(OUT1, OUTPUT);
digitalWrite(OUT1, LOW);

// mensaje de debug serial
Serial.print("Conectando a la red
                ")
; Serial.println(ssid);

// Inicio de la conexión WiFi
WiFi.begin(ssid, senha);

// espera la conexión WIFI
while (WiFi.status() !=
      WL_CONNECTED) { delay(500);

}

server.begin();
Serial.println("Servidor inicializado!");

//mensaje de conectado
Serial.println("");
Serial.println("WiFi conectado");
Serial.println("Dirección IP: ");
Serial.println(WiFi.localIP());
}

void loop() {
  // Comprueba si el cliente está
  // conectado
  WiFiClient client =
  server.available(); if (!client) {
    return;
  }

  // Espera datos del cliente
  Serial.println("connect");
  while (!client.available())
  {
    delay(1);
  }

  String req = client.readStringUntil('\
r'); Serial.println(req);
  client.flush();

  if (req.indexOf("ioon") != -1)
  {
    digitalWrite(OUT1, HIGH);
    statusOUT1 = HIGH;
  }
}

```

```

}
else if (req.indexOf("iooff") != -1)
{
    digitalWrite(OUT1, LOW); statusOUT1
    = LOW;
}

// Si el servidor, logró entender la llamada que hicimos arriba, devuelve el
valor leído y muestra en el navegador
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/
html"); client.println("");
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<h1>Workshop ESP8266</h1>");

if(!statusOUT1)
client.println("<p>OUT1 <a
href=\"ioon\"><button>CONECTAR</button></a></p>"); else
client.println("<p>OUT1 <a href=\"iooff\"><button>DESCONECTAR</button></a></p>");

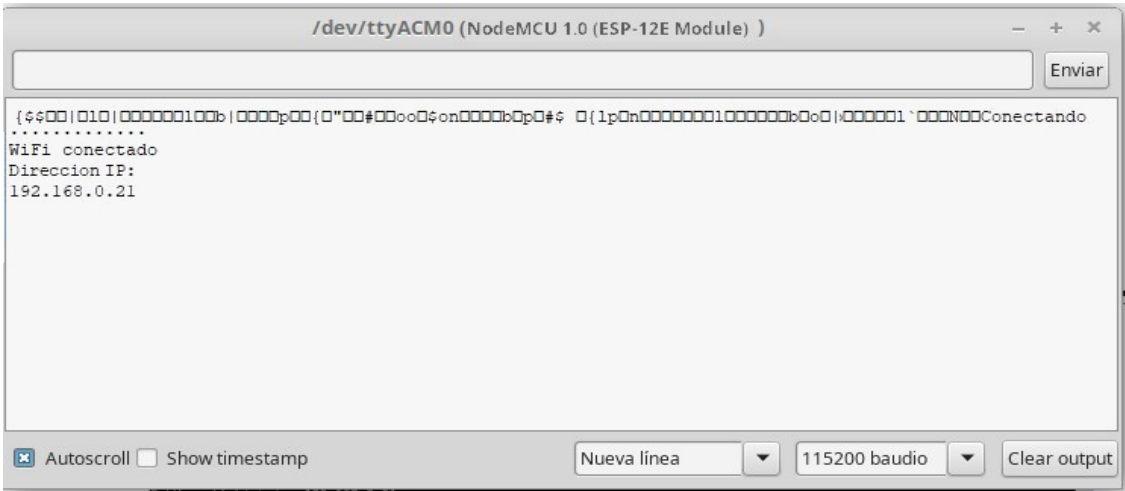
client.println("</

html>"); delay(10);
}

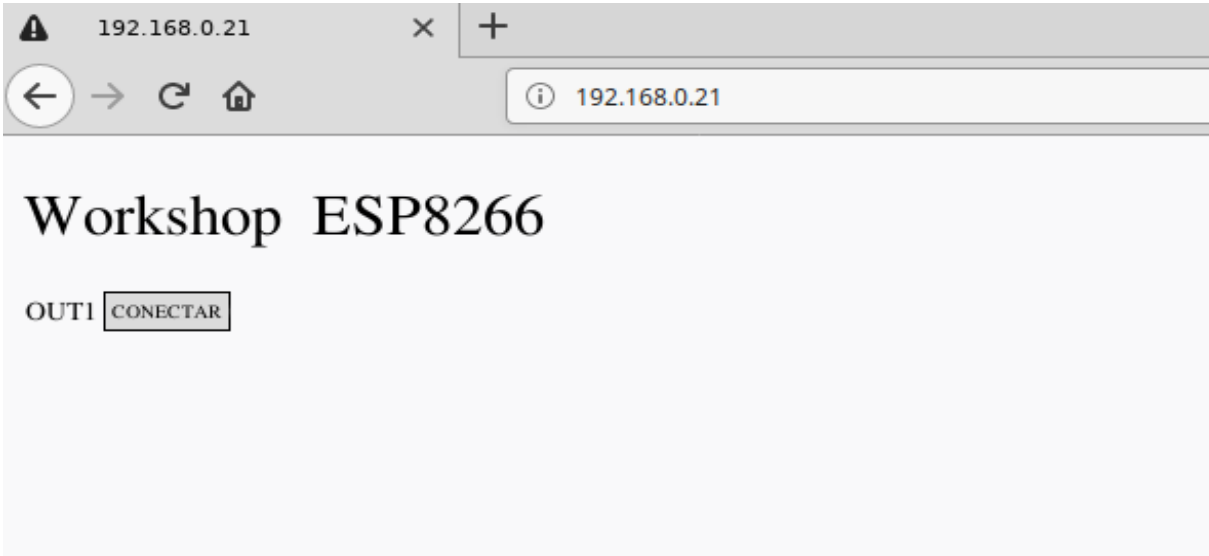
```



Abrir el terminal y ver que IP se ha asignado a la tarjeta:



Introducir esta dirección IP en el navegador:



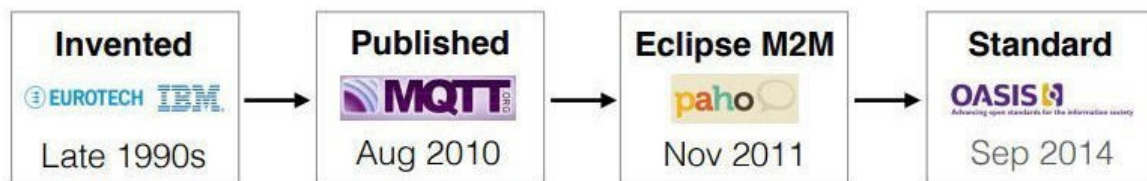
Hacer clic en el botón y ver si el estado del LED cambia en la tarjetas.

# MQTT (Message Queue Telemetry Transport)

Creada por IBM a finales de la década de los 90, es un protocolo de mensajes ligero, utilizado para la comunicación Machine to Machine (M2M). Es un protocolo de mensajes asíncrono, que facilita la aplicación en diversos escenarios en IoT.

Utiliza un modelo de comunicación de publicación y firma, lo que facilita la expansión de dispositivos en el sistema.

En 2014 se convirtió oficialmente en el estándar abierto OASIS.



Hoy el MQTT es muy utilizado en aplicaciones IoT ya pesar de su simplicidad, presenta características como seguridad, calidad de servicio y flexibilidad.

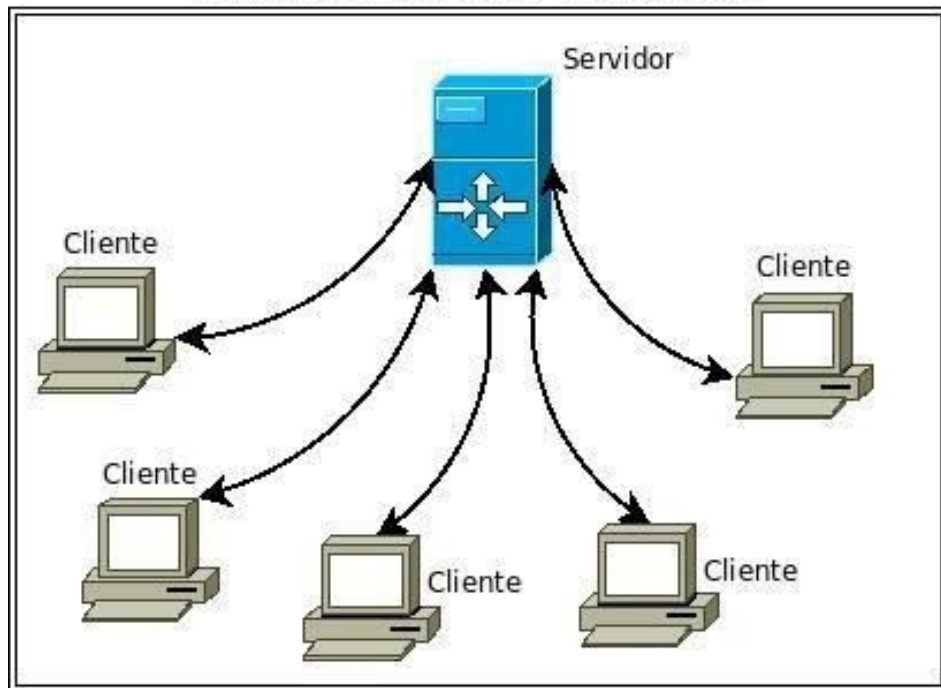
Necesita poca banda y procesamiento, pudiendo ser utilizado por hardware más simplificados, con microcontroladores. Se basa en TCP / IP tiene carga más pequeña que el HTTP.

## Por que MQTT?

A pesar de las diversas ventajas presentadas arriba, vamos a analizar la estructura de comunicación del MQTT.

Es importante recordar el modelo cliente-servidor:

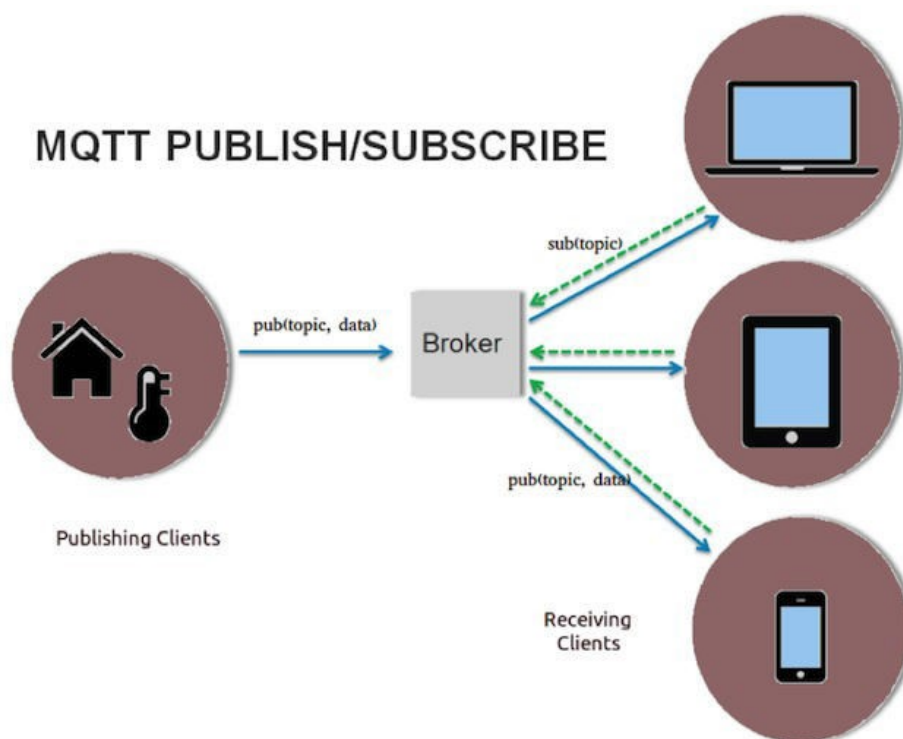
## Modelo Cliente-Servidor



La característica del modelo cliente-servidor, describe la relación de programas en una aplicación. El componente de servidor proporciona una función o servicio a uno o más clientes que inician las solicitudes de servicio. Es una comunicación sincrónica, el cliente inicia la comunicación y espera la respuesta del servidor.

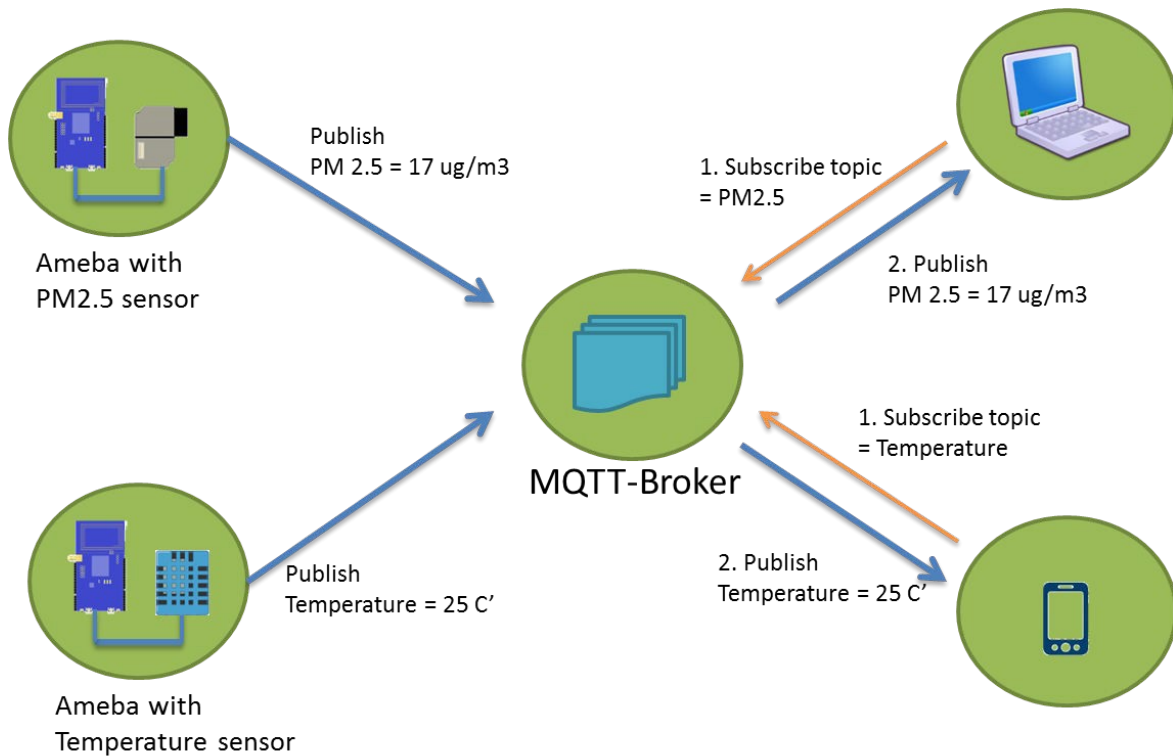
Este modelo es muy utilizado en aplicaciones WEB, pero no es muy eficiente para aplicaciones IoT.

El MQTT tiene la arquitectura adecuada para aplicaciones con diversos sensores. Tiene dos elementos en la red: Un broker y numerosos clientes:



El broker es un servidor que recibe los mensajes de los clientes que envían a los clientes que firman los temas. El patrón de intercambio de mensajes utilizado es el publish / subscriber (publicador / suscriptor), que facilita el intercambio de mensajes entre los elementos de forma asincrónica. La comunicación se realiza de la siguiente manera:

1. El cliente se conecta al broker a través de una conexión TCP / IP, firmando un "tema" de mensaje;
2. El cliente publica los mensajes en un asunto en el broker;
3. El broker encamina el mensaje a todos los clientes que firman este tema.



ejemplo de tópicos:

casa/temperatura  
casa/humedad  
casa/L1

## Broker

Hay varias implementaciones de [brokers MQTT](#), de código abierto o cerrado. Una de ellas, es [Eclipse Paho](#) que proporciona SDKs y bibliotecas de MQTT en varios lenguajes de programación.

Puede instalar el broker en un equipo local e interactuar a través de la línea de comandos o conectar dispositivos a la red local. Para descargar e instalar el módulo mosquitto acceda al [sitio del mosquitto](#).

Para este tutorial vamos a utilizar el CloudMQTT, un servicio en nube que facilitará el control de dispositivos a través de Internet.

CloudMQTT tiene un plan gratuito que permite 10 conexiones a una velocidad de 10 Kbit / s.

## Creación de una cuenta en CloudMQTT

Acceder al [sitio web de CloudMQTT](https://customer.cloudmqtt.com) y realizar el registro.

Después de iniciar sesión, hacer clic en “+Crear nueva instancia”. Se abrirá la siguiente página:

Instances - CloudMQTT

https://customer.cloudmqtt.com/instance/create

Plan Region Configure (Dedicated plans only) Confirm

### Select a plan and name - Step 1 of 4

Name: OficinaloT

Plan: Cute Cat (Free)

Tags:

Tags are used to separate your instances between projects. This is primarily used in the project listing view for easier navigation and access control.

Tags allow admins to [manage team members access](#) to different groups of instances.

Plan

Cute Cat

See the [plan page](#) to learn about the different plans.

Cancel Select Region

Ingresar el nombre, elegir el plan (Cute Cat) y hacer clic en Select Region.

Instances - CloudMQTT

https://customer.cloudmqtt.com/instance/region?name=OficinaIoT&plan=cat&tags=

## Create new instance

No credit card Please [add a credit card](#) if you want to subscribe to a paid plan

Plan Region Configure (Dedicated plans only) Confirm

### Select a region and data center - Step 2 of 4

Data center US-East-1 (Northern Virginia)

aws


« Back Cancel Review

Continuar dando clic en “Review”

Plan Region Configure (Dedicated plans only) Confirm

### Confirm new instance - Step 4 of 4

Plan



Cute Cat

Total: \$0/month

Name: OficinaIoT

Provider: Amazon Web Services

Region: US-East-1 (Northern Virginia)

Tags:

« Back Cancel Create instance ✓


Finalizar dando clic en “Create Instance”

SETTINGS  
CERTIFICATES  
USERS & ACL  
BRIDGES  
AMAZON KINESIS STREAM  
WEBSOCKET UI  
STATISTICS  
CONNECTIONS  
LOG

### Instance info

Server	m11.cloudmqtt.com	
User	[REDACTED]	<button>Restart</button>
Password	[REDACTED]	<button>Rotate</button>
Port	13403	
SSL Port	23403	
Websockets Port (TLS only)	33403	
Connection limit	5	

#### Active Plan



Cute Cat

Upgrade Instance

### Reset DB

This will erase all stored messages and sessions. The instance will be restarted.

Reset DB

### API

API Key [REDACTED]90

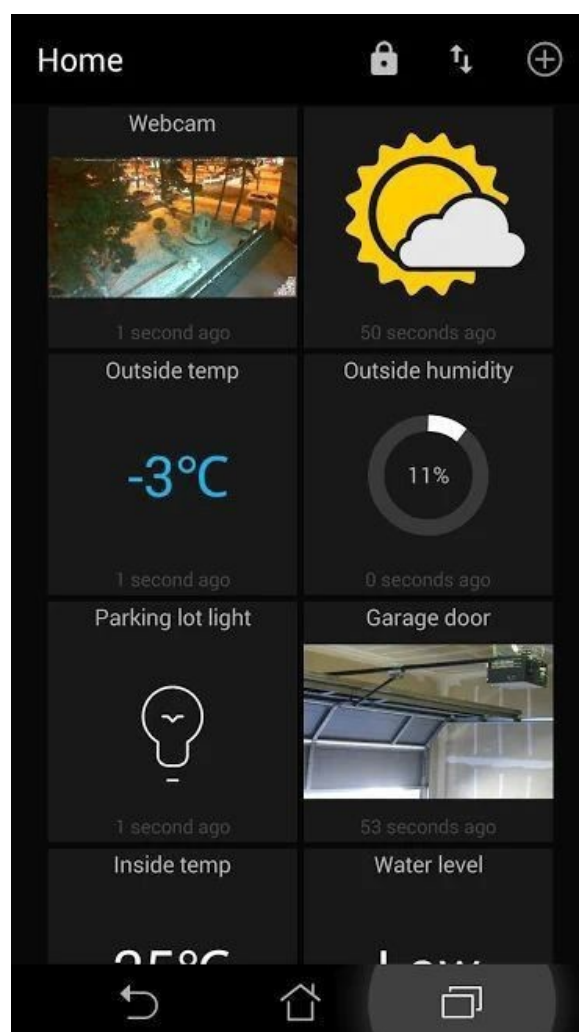
API documentation can be found at [https://docs.cloudmqtt.com/cloudmqtt\\_api.html](https://docs.cloudmqtt.com/cloudmqtt_api.html).

Aquí está la información necesaria para conectarse a Broker: Server, User, Password y Port. Vamos a utilizar esta información en el código del ESP8266 y en la aplicación MQTT Dash, más adelante.



# MQTT DASH

MQTT Dash es una de las mejores aplicaciones para interfaz gráfica en el teléfono inteligente. Tiene una interfaz agradable, de fácil personalización y configuración, siendo una de las mejores aplicaciones que he utilizado para ese fin.



Se puede descargar de [Google Play](https://play.google.com/store/apps/details?id=com.mqtt.dash) e instalar en un smartphone.

Luego vamos a configurar la aplicación.

# Diseño con MQTT Dash y CloudMQTT para Accionamiento de lámpara con ESP8266

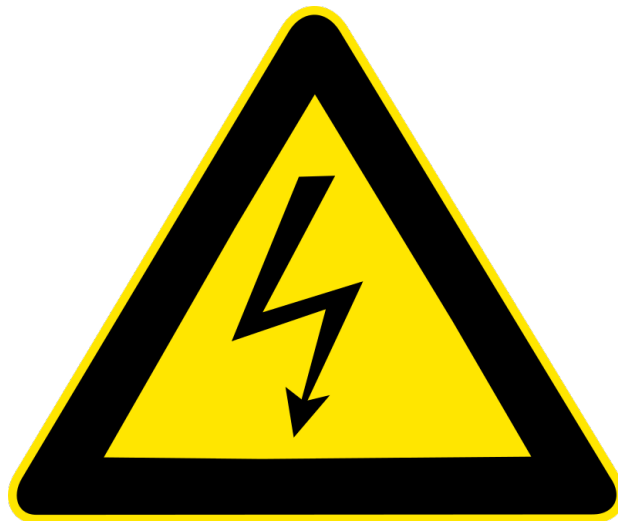
A continuación veremos cómo utilizar esta aplicación en conjunto con el ESP8266 y un broker MQTT configurado en el CloudMQTT, para controlar una lámpara remotamente. ¡Mira que fácil!

## Materiales necesarios

1. tarjetas con ESP8266 (nodeMCU, Wemos D1 o mini, etc);
2. Módulo RELÉ;
3. Lámpara;
4. Cables y Jumpers;

También se puede probar el ejemplo presentado aquí, sin usar el relé y la lámpara, sólo con una tarjeta ESP8266 y LEDs en un protoboard.

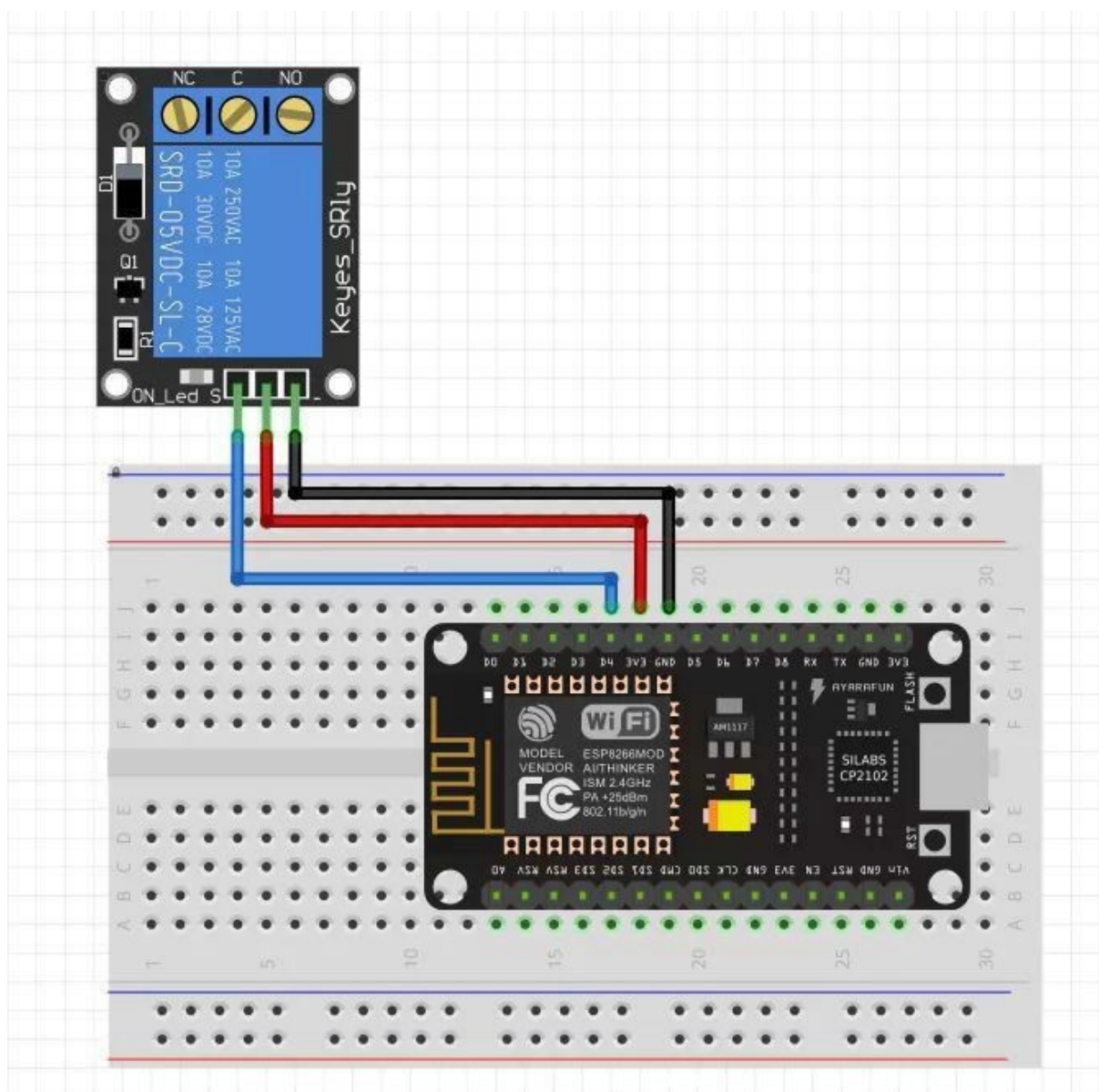
1



**TENER MUCHO CUIDADO ACTIVANDO CARGAS DE CORRIENTE ALTERNA Y VOLTAJES PELIGROSOS.**

<sup>1</sup>Made with [Sodipodi](#) by [Duesentrieb](#) - Trabajo propio (Wikipedia)

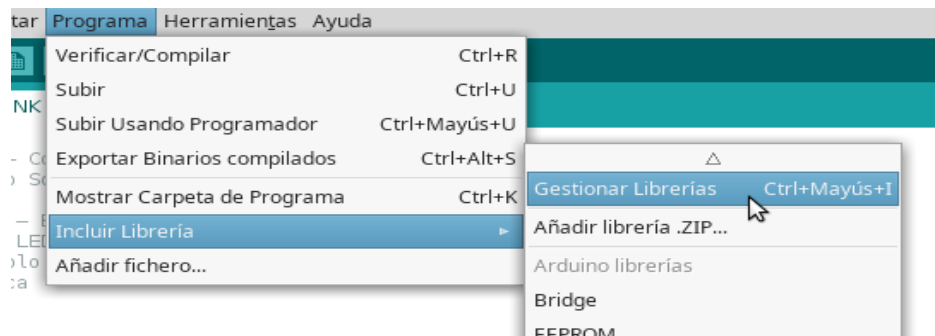
## Circuito



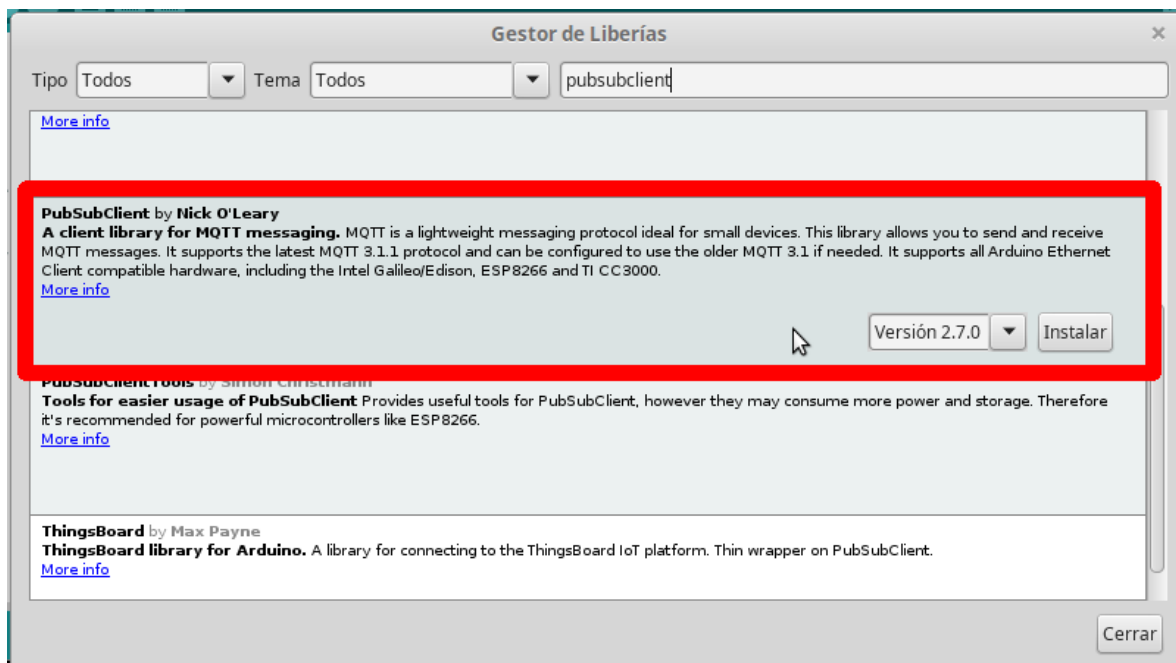
# Código para tarjetas ESP8266 en Arduino

Como dijimos vamos a utilizar una tarjetas nodeMCU que activará una lámpara. Realizar la configuración del IDE, tal como se muestra en la sección correspondiente.

Se debe instalar la biblioteca **PubSubClient.h**. Para ello, accede a **Sketch> Incluir Biblioteca> Administrar bibliotecas**:



Buscar PubSubClient e instalar la siguiente opción:



En el siguiente código, reemplazar la información de la red WIFI (ssid, password) y del Broker MQTT (mqttServer, mqttUser, mqttPassword, mqttPort), indicados con x. Compilar y cargar a la tarjeta con ESP8266:.

```
/*
 * Workshop - Controle dispositivos remotamente con ESP8266
 * Por: Fábio Souza
 *
 * ejemplo 5 - MQTT
 * Asigna tema en el servidor MQTT
 */

#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#define DEBUG

#define L1 2 //pin de salida para Accionamiento de la Lámpara L1

//informações da rede WIFI
const char* ssid = "ssid"; //SSID de la red WIFI
const char* password = "contraseña"; //contraseña de red wifi

//información del broker MQTT - Compruebe la información generada por CloudMQTT
const char* mqttServer = "xxxxxxxxxxxxxxxx";
//
server const char* mqttUser = "xxxxxxx"; //user
const char* mqttPassword = "xxxxxxx";

//password const int mqttPort = xxxxx; //port

const char* mqttTopicSub = "casa/L1"; //tema que sera asignado

WiFiClient espClient;
PubSubClient
client(espClient);

void setup() {

  Serial.begin(115200);
  pinMode(L1, OUTPUT);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED)
  { delay(500);
    #ifdef DEBUG
```

```

Serial.println("Conectando al WiFi..");

    #endif
}
#ifdef DEBUG
Serial.println("Conectado a la red WiFi");
#endif

client.setServer(mqttServer, mqttPort);
client.setCallback(callback);

while (!client.connected()) {
    #ifdef DEBUG
    Serial.println("Conectando al Broker MQTT...");
    #endif

    if (client.connect("ESP8266Client", mqttUser, mqttPassword )) {
        #ifdef DEBUG
        Serial.println("Conectado");
        #endif

    } else {
        #ifdef DEBUG
        Serial.print("error estado
                        ")
        ;
        Serial.print(client.state())
        ;
        #endif
        delay(2000);
    }
}

//suscribe en el tema
client.subscribe(mqttTopicSub);

}

void callback(char* topic, byte* payload, unsigned int length) {

    //almacena el msg recibido en una string
    payload[length] = '\0';
    String strMSG = String((char*)payload);

    #ifdef DEBUG
    Serial.print("Mensaje llega del tema:
    "); Serial.println(topic);
    Serial.print("Mensaje:");
    Serial.print(strMSG);
    Serial.println();
    Serial.println("-----");
    #endif
}

```

```

//Acciona salida según el mensaje recibido
if (strMSG == "1"){           //si msg "1"
    digitalWrite(L1, LOW); //coloca salida en LOW para encender la lámpara -> el módulo
    RELE usado tiene Accionamiento invertido. Si es necesario ajuste para su módulo
}else if (strMSG == "0"){ //se msg "0"
    digitalWrite(L1, HIGH); //coloca salida en HIGH para apagar la lámpara -> el
    módulo RELE usado tiene Accionamiento invertido. Si es necesario ajuste para su
    módulo
}

}

//función para volver a conectar al servicio MQTT
void
reconnect()
{
    //Mientras esté desconectado
    while (!client.connected()) {
        #ifdef DEBUG
        Serial.print("Intentar conectarse al servidor MQTT");
        #endif

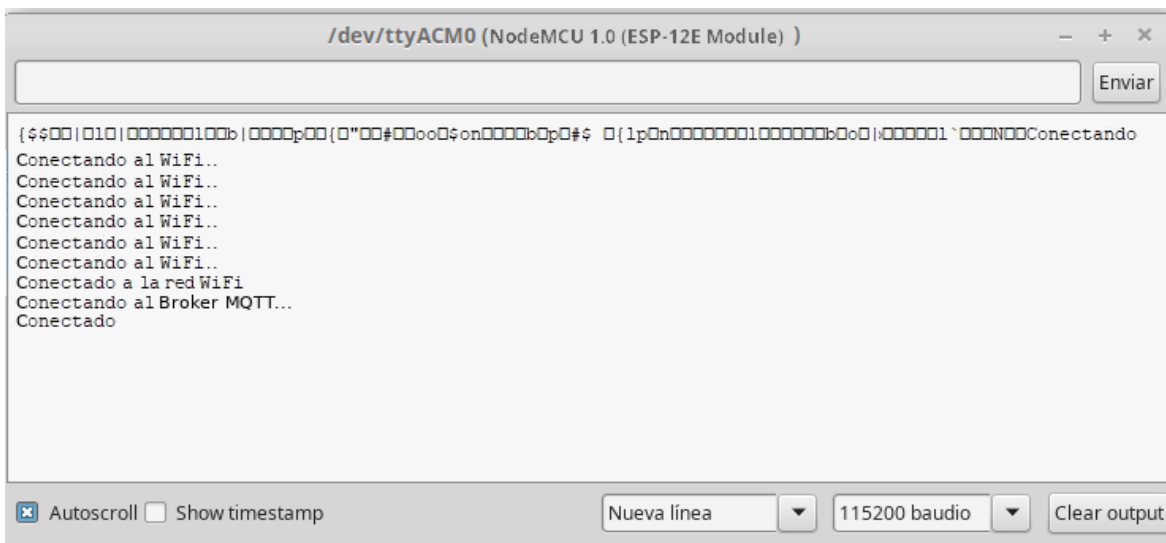
        bool conectado = strlen(mqttUser) > 0 ?
            client.connect("ESP8266Client", mqttUser, mqttPassword) :
            client.connect("ESP8266Client");

        if(conectado) {
            #ifdef DEBUG
            Serial.println("Conectado!");
            #endif
            //suscribe en el tema
            client.subscribe(mqttTopicSub, 1); //nivel de calidad: QoS 1
        } else {
            #ifdef DEBUG
            Serial.println("Error durante la
            conexión.Code: "); Serial.println(
            String(client.state()).c_str());
            Serial.println("Intentando de nuevo en 10
            s");
            #endif
            //Aguarda 10 segundos
            delay(10000);
        }
    }
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}

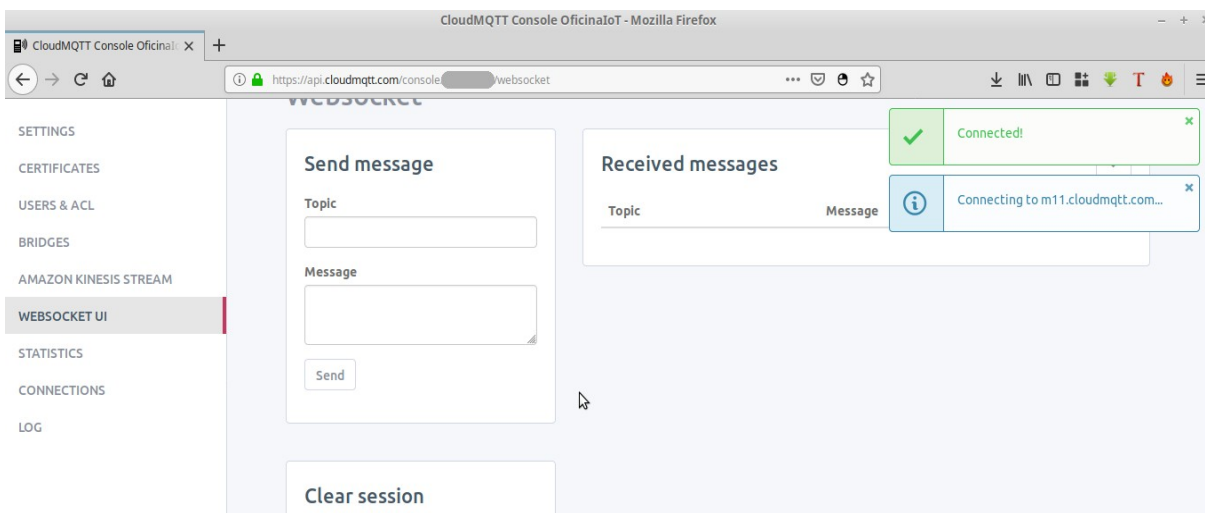
```

Abrir el terminal y ver si la tarjeta se conectó a la red WIFI y al broker:



Prueba de escritura en el tema con CloudMQTT

Abrir la ficha Websocket de la interfaz de usuario en CloudMQTT:



Enviar un mensaje en el tema casa / L1 por el Websocket UI:

- 0 - Salida a nivel LOW
- 1 - Salida a nivel HIGH



**Websocket**

**Send message**

Topic:

Message:

**Send**

**Received messages**

Topic	Message
casa/L1	1
casa/L1	0
casa/L1	1
casa/L1	0

Comprobar que el estado de la salida muda en la aplicación ha cambiado.

También se puede ver si la tarjeta recibió el mensaje a través del terminal de serie:

```

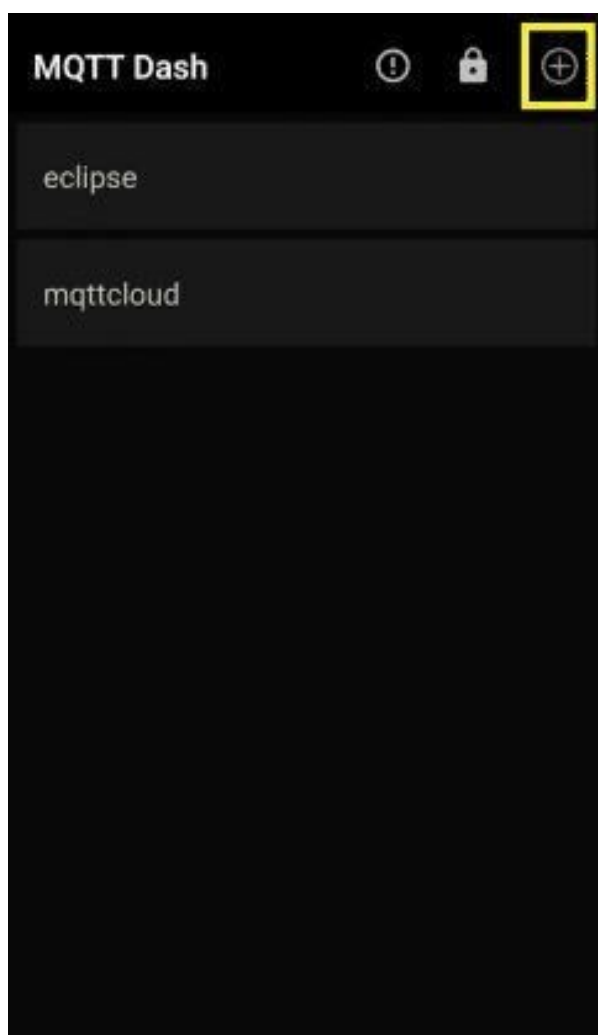
/dev/ttyACM0 (NodeMCU 1.0 (ESP-12E Module) )
{ $ $ 0 0 | 0 1 0 | 0 0 0 0 0 0 1 0 0 b | 0 0 0 0 p 0 0 { 0 " 0 0 # 0 0 o o 0 $ o n 0 0 0 0 b 0 p 0 # $ 0 { 1 p 0 n 0 0 0 0 0 0 0 1 0 0 0 0 0 0 b 0 o 0 | 0 0 0 0 0 1 ` 0 0 0 0 0 0 0 Conectando
Conectando al WiFi...
Conectando al WiFi...
Conectando al WiFi...
Conectando al WiFi...
Conectando al WiFi...
Conectando al WiFi...
Conectado a la red WiFi
Conectando al Broker MQTT...
Conectado
Mensaje llega del tema: casa/L1
Mensaje:1
-----
Mensaje llega del tema: casa/L1
Mensaje:0
-----

```

Autoscroll ☒ Show timestamp ☐ Nueva línea  115200 baudio  Clear output

## Configuración de la aplicación en el MQTT Dash

Después de instalar, hacer clic en el signo "+" en su pantalla de inicio:



Se abrirá la configuración de una nueva conexión. Introducir la siguiente información:

- Name
- Address
- Port
- User Name
- User Password

**MQTT Dash**

Name  
casa

Address  
m13.cloudmqtt.com

Port  
12857

☐ Enable connection encryption (SSL/TLS).  
Note: if server certificate is self-signed, you need to install it to your device or enable option below, otherwise connection will fail. If server certificate issued by a known Certificate Authority (CA), it will work out of box, without installing to you device. Also don't forget, that MQTT servers have different ports for plain and SSL/TLS connections.

☐ This broker uses self-signed SSL/TLS certificate. I trust this certificate at my own risk.

User name  
dreqdghr

User password  
.....

Client ID (must be unique)  
\_\_\_\_\_

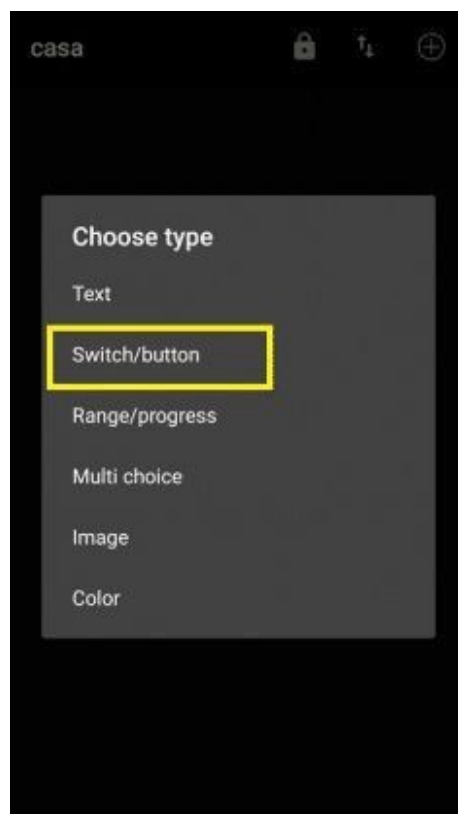
Y, por último, guardar tocando el disquete en la parte superior derecha.

Abrir la conexión creada. Si la configuración es correcta, no aparecerá ningún mensaje. En caso contrario, aparecerá el mensaje de error de conexión. Si esto ocurre, rehacer la configuración.

Con la conexión configurada correctamente, hacer clic en el signo "+" dentro del tablero de instrumentos creado:



Insertar un Switch / button:



El botón se utilizará para Accionamiento de la lámpara. Se puede dar el nombre L1 para él, u otro que encontremos mejor. Para el tema, configurar "casa / L1":



The image shows a configuration screen for an MQTT Dash metric. At the top, it says "MQTT Dash" with a save icon. Below is a description: "This metric is intended for state displaying and switching (e.g. light on/off). Or it can behave as a simple static button. Payload is expected to be string." The "Name" field is set to "L1". The "Topic (sub)" field is set to "casa/L1". There is a link to JSON path documentation: <https://github.com/jayway/JsonPath/blob/master/README.md>. The "Enable publishing" checkbox is checked. The "Topic (pub)" field is empty, with a note "keep empty if the same as sub". The "Update metric on publish immediately (do not wait for incoming message to update visual state)" checkbox is also checked. At the bottom, there is a note about payload and icons: "Payload and icons. If you need not a switch, but a simple button, just set the same payload values and the same icons for On and Off. This way the switch will never change icon and always send the same."

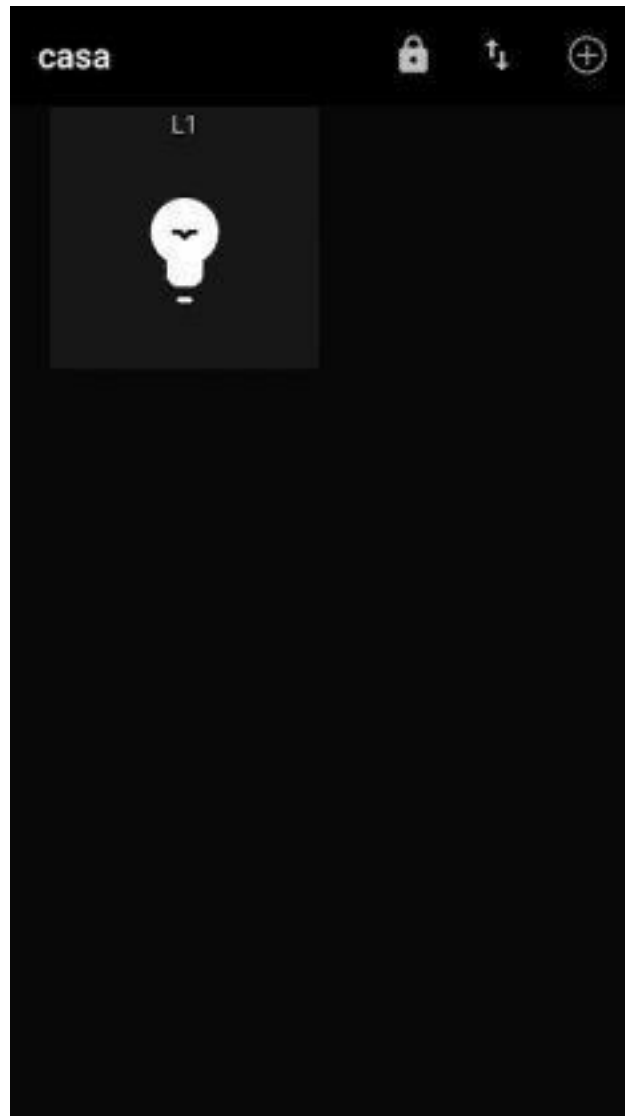
Para la parte visual de este botón, configurar para que muestre iconos de una lámpara encendida y otra apagada.

Para On, enviaremos el valor "1" y Off, el valor "0":



Por último, seleccionar el nivel de calidad para QoS(1).

Listo, la aplicación está conectada y configurada:



Abrir la guía Websocket UI en el CloudMQTT y ver si los mensajes están llegando al presionar el botón:

### Websocket

#### Send message

Topic

Message

Send

#### Received messages

Topic	Message
casa/L1	1
casa/L1	0
casa/L1	1
casa/L1	0
casa/L1	1
casa/L1	0
casa/L1	1
casa/L1	0
casa/L1	1

#### Clear session

Clear any data for a client id

Si la tarjeta está conectada la salida también cambiará su estado.

Ahora enviar el comando vía Websocket UI en el CloudMQTT y ver lo que sucede en la aplicación.

## Desafío

Añadir más salidas y otros temas al proyecto



# Referencias

<https://www.embarcados.com.br/mqtt-dash/> \_

<https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html> <https://www.embarcados.com.br/mqtt-protocolos-para-iot/>