

## Formación Profesional en CePETel 2022

Desde la Secretaría Técnica del Sindicato CePETel convocamos a participar del siguiente curso de formación profesional:

### Introducción IoT (Internet of Things)

**Clases:** 5 de 3hs c/u de 18:30 a 21:30 hs.

**Días que se cursa:** los días lunes 2, 9, 16, 23 y 30 de mayo.

**Modalidad:** a distancia (requiere conectarse a la plataforma Zoom en los días y horarios indicados precedentemente).

**Docente:** Federico D'Angiolo

**La capacitación es:**

- Sin cargo para afiliados y su grupo familiar directo.
- Sin cargo para encuadrados con convenio CePETel.
- Con cargo al universo no contemplado en los anteriores.

**Informes:** enviar correo a [tecnico@cepetel.org.ar](mailto:tecnico@cepetel.org.ar)

**Inscripción (hasta el 27 de abril):** ingresar al formulario (se recomienda realizar el registro por medio de una cuenta de correo personal y **no utilizar dispositivos de la empresa para acceder al link**).

<https://forms.gle/E1mm1ft4jpmv2Y1X8>

### Objetivos

Que los participantes puedan:

- Comprender los conceptos, evolución, modelos y aplicaciones de sistemas basados en IoT.
- Conocer la programación de microcontroladores mediante sistemas basados en IoT.
- Adquirir conceptos básicos de sensores, actuadores y módulos de comunicaciones aplicados a IoT.
- Interpretar el vínculo entre Software, Firmware y Hardware.
- Adquirir las bases y herramientas para desempeñarse mejor y más activamente en sus respectivas áreas de trabajo y/o en la interacción con otras áreas de gestión de las telecomunicaciones y aplicaciones en segmentos verticales como industrias varias.

**Temario:**

**Unidad 1: Introducción a IoT (Internet of Things)**

**Ing. Daniel Herrero – Secretario Técnico – CDC**

- ¿Qué es IoT? Aplicaciones en la vida cotidiana y la industria.
- Protocolos utilizados en IoT.
- Introducción a Redes. Protocolos utilizados en internet.
- Introducción a módulos utilizados en IoT basados en IDE Arduino. ESP8266 y ESP32.
- Conceptos de Software y Hardware Libre.

## **Unidad 2: Introducción a la programación de módulos basados en ESP 8266 mediante Arduino.**

- Comparación de placas: Arduino UNO, ESP8266, ESP32. Pines de I/O digitales y analógicos.
- Instalación de IDE y módulos ESP 8266.
- Configuración y puesta en marcha. Primer programa en IDE de Arduino con módulos basados en ESP 8266. Funciones setup() y loop().
- Instrucciones pinMode() y digitalWrite(). Manejo del led on-board.
- Variables: tipos de variables, rango, signo, definición, asignación.

## **Unidad 3: Prácticas con módulos basados en ESP8266**

- Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.
- Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.
- Manejo de puertos digitales. Utilización de Relays.
- Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.
- Manejo de servo motor.
- Dispositivos I<sup>2</sup>C: Display LCD.

## **Unidad 4: Comunicaciones WiFi con placas basadas en ESP8266.**

- Primeros pasos en la conexión de WiFi mediante **Wemos D1**. (modo STA).
- Transmisión de datos de sensores a ThingsPeak. Estación meteorológica. Extracción de archivos .csv.
- ESP8266 como Web Server. HTML. Automatización de sistemas de potencia para el hogar y la industria.
- Automatización de sistemas para el hogar y la industria mediante protocolo MQTT. Análisis de Brokers.

## **Unidad 5: Integración de Conceptos**

Proyecto integrador basado en los conceptos adquiridos.

### **Requisitos:**

Conocimientos básicos de hardware, programación y Arduino.

**Ing. Daniel Herrero – Secretario Técnico – CDC**

## **Acerca del docente**

Federico D ´Angiolo es Ingeniero en Electrónica egresado de la Facultad de Ingeniería de la Universidad de Buenos Aires.

En la actividad independiente ha trabajado en el área de Sistemas Embebidos e IoT; análisis, desarrollo y asesoramiento en arquitecturas de 8 y 32 bits, especialmente en ARM; Desarrollo de Fuentes Conmutadas para aplicaciones específicas; Desarrollo de PCB para circuitos digitales y analógicos; Desarrollo de algoritmos de Machine Learning en Python. En la CONAE-FIUBA participó en el Diseño y modelado del bus de potencia del instrumento SAR para el satélite SAOCOM; efectuando el análisis de EMC y Diseño de un filtro para atenuar la interferencia conducida (trabajo que fue evaluado como Tesis de Grado). También se desempeñó en el diseño y simulación de un controlador para SMPS (Flyback), tarea que fuera solicitada exclusivamente por CONAE para el satélite SAOCOM.

En la docencia es Ayudante Primero en la FIUBA, Universidad de Buenos Aires en diseño e implementación de equipos de potencia en la asignatura Diseño de Circuitos Electrónicos. También es Profesor Asociado, UNDAV, Universidad Nacional de Avellaneda en Investigación y enseñanza en el área de Sistemas Embebidos utilizando arquitecturas ARM. Durante el 2021 dictó para nuestro sindicato y de manera virtual los cursos de Inteligencia Artificial, Robótica e Impresoras 3D y 4 D; e Introducción a Arduino.

**Ing. Daniel Herrero – Secretario Técnico – CDC**

<http://www.cepetel.org.ar> ✉ tecnico@cepetel.org.ar 📍 Rocamora 4029 (CABA) ☎ (+54 11)35323201

# Introducción a Sistemas Embebidos con Arduino

Coordinador CePETel: Ing. Daniel Herrero

Coordinadores Curso: Mg.Ing. Mayer, Roberto Osvaldo.

Ing. Paradiso, Juan Carlos.

Ing. Crivelli, Marcelo

Docente: Ing. D'Angiolo, Federico Gabriel [fgdangiolo@gmail.com](mailto:fgdangiolo@gmail.com)



# Planificación

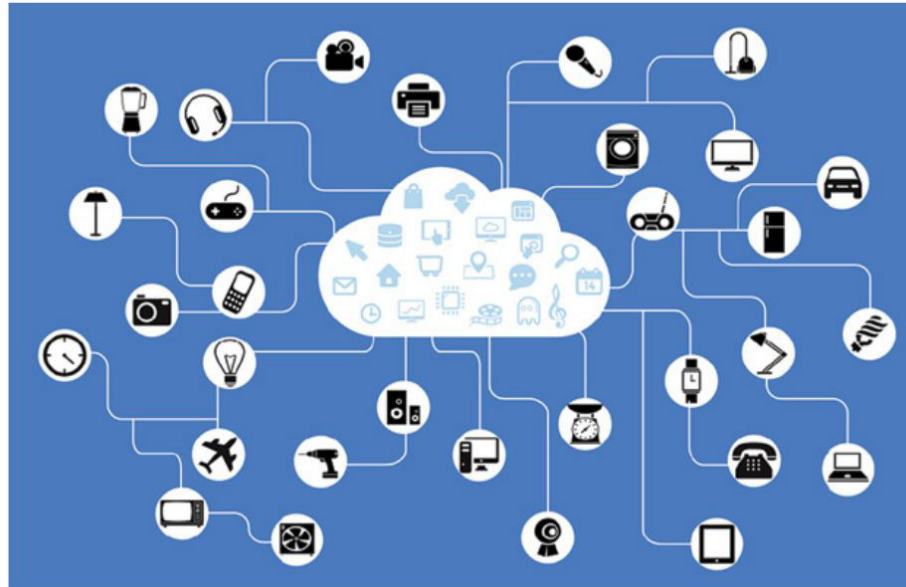
- **Introducción a IoT (Internet of Things).**
- Introducción a la programación de módulos basados en ESP 8266 mediante Arduino.
- Prácticas con módulos basados en ESP8266.
- Comunicaciones WiFi con placas basadas en ESP8266.
- Integración de Conceptos.

# Planificación

- - **¿Qué es IoT? Aplicaciones en la vida cotidiana y la industria.**
- - Protocolos utilizados en IoT.
- - Introducción a Redes. Protocolos utilizados en internet.
- - Introducción a módulos utilizados en IoT basados en IDE Arduino. ESP8266 y ESP32.
- - Conceptos de Software y Hardware Libre.

# ¿Qué es IoT?

El *Internet de las Cosas* (Internet of Things, en inglés), es una tecnología que permite **la conexión y comunicación entre múltiples dispositivos, utilizando internet**. Esta red tiene en cuenta dispositivos como **computadoras, teléfonos, tablets, sistemas embebidos**, etc.



# ¿Qué es IoT?

Cuando nos referimos a las **cosas** en **IoT**, lo hacemos a equipos como **electrodomésticos, máquinas, vehículos**, etc. De esta manera, todos los dispositivos se encuentran interconectados para intercambiar datos entre sí, sin la intervención humana y facilitando las conexiones remotas. [2]



# ¿Qué es IoT?

Para conocer cómo funciona la IoT, podemos ver algunos pasos:

- 1) Cada **cosa**, debe tener una identificación dentro de la red.
- 2) Cada **cosa**, debe poder comunicarse (WiFi, Bluetooth, ZigBee, etc)
- 3) Cada **cosa**, debe tener sensores para que se pueda obtener información del dispositivo. **Ejemplo:** sensores de temperatura, humedad, presión, etc.
- 4) Cada **cosa**, puede tener un microcontrolador que sea capaz de gestionar la toma de datos y la comunicación.
- 5) Se necesita un **servicio en la nube** para **almacenar, analizar y mostrar datos**, de forma que podamos estar al tanto.

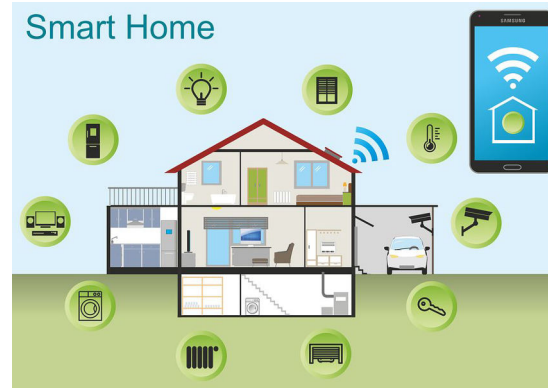
[2]

# Aplicaciones

## Algunas aplicaciones:

**Casas Inteligentes:** Probablemente esta sea una de las aplicaciones más intuitiva, lo que se llama **Smart Homes o Casas Inteligentes**, donde podemos **monitorizar y controlar**, por ejemplo, artefactos de **ventilación, aires acondicionados, estufas, luces, y sistemas de seguridad**.

Desde un celular podemos controlar distintos equipos que se pueden encontrar en nuestras casas y generar así, no sólo la **automatización del hogar** sino una **reducción en el consumo de energía**. [2]





# Aplicaciones

## Algunas aplicaciones:

**Transporte:** Mediante **IoT** los autos se pueden conectar para **planificar viajes, conocer el tráfico, encontrar lugares para estacionar y reducir accidentes**. Muchas empresas, tales como Tesla, Google, Uber, Cabify, Volkswagen, General Motors, y otras, se encuentran promoviendo esta tecnología. [2]

También se puede encontrar el **IoT** en el transporte público para conocer el estado y ubicación de trenes, colectivos, etc.





# Planificación

- - ¿Qué es IoT? Aplicaciones en la vida cotidiana y la industria.
- - **Protocolos utilizados en IoT.**
- - Introducción a Redes. Protocolos utilizados en internet.
- - Introducción a módulos utilizados en IoT basados en IDE Arduino. ESP8266 y ESP32.
- - Conceptos de Software y Hardware Libre.

# Protocolos utilizados en IoT

## ¿Qué es un protocolo?

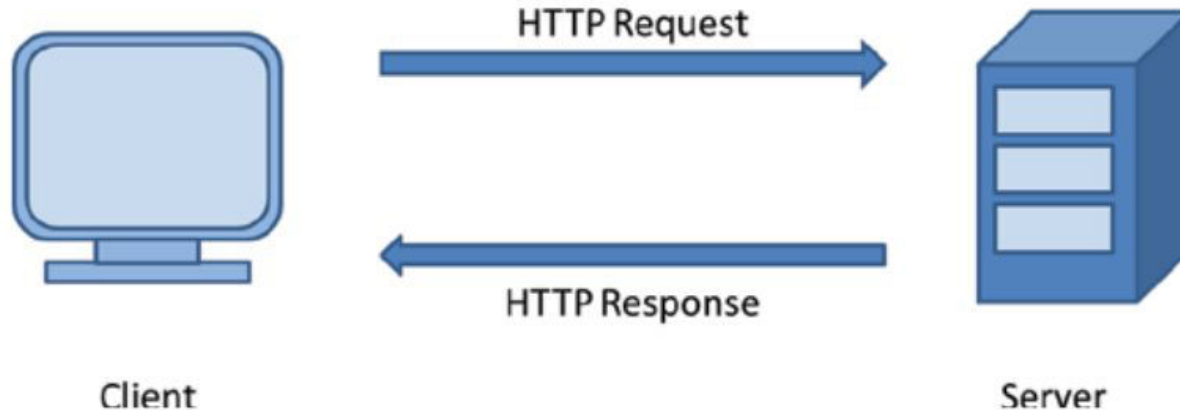
Los protocolos de comunicación son un **conjunto de reglas** que le permiten a los dispositivos, **comunicarse entre sí**. Estos permiten definir la sintaxis, la semántica y la sincronización, entre otras cosas.

Existen varios protocolos de comunicación para aplicaciones de IoT, algunos de ellos son:

- 1) HTTP
- 2) MQTT
- 3) CoAP

# Protocolos utilizados en IoT

**HTTP:** Este protocolo de **transferencia de hipertexto** (HTTP, Hypertext Transfer Protocol) es el utilizado en la **World Wide Web** (WWW). Se basa en la arquitectura **cliente-servidor** y opera en forma de **solicitud y respuesta** (request - response). Usa **TCP** (protocolo de control de transmisión) para proporcionar conexiones confiables.



# Protocolos utilizados en IoT

## Ejemplo de HTTP

### Request

```
GET /index.html HTTP/1.1
Host: www.mbed.com
Connection: keep-alive
User-agent: Mozilla/4.0
Accept-language: en
```

### Response

```
HTTP/1.1 200 OK
Server: nginx/1.7.10
Date: Sun, 12 Feb 2017 12:21:57 GMT
Content-Type: text/html
Content-Length: 185
Connection: close
Location: https://www.mbed.com/

<html>
<head><title>... ..</title></head>
<body>
... ..
</body>
</html>
```

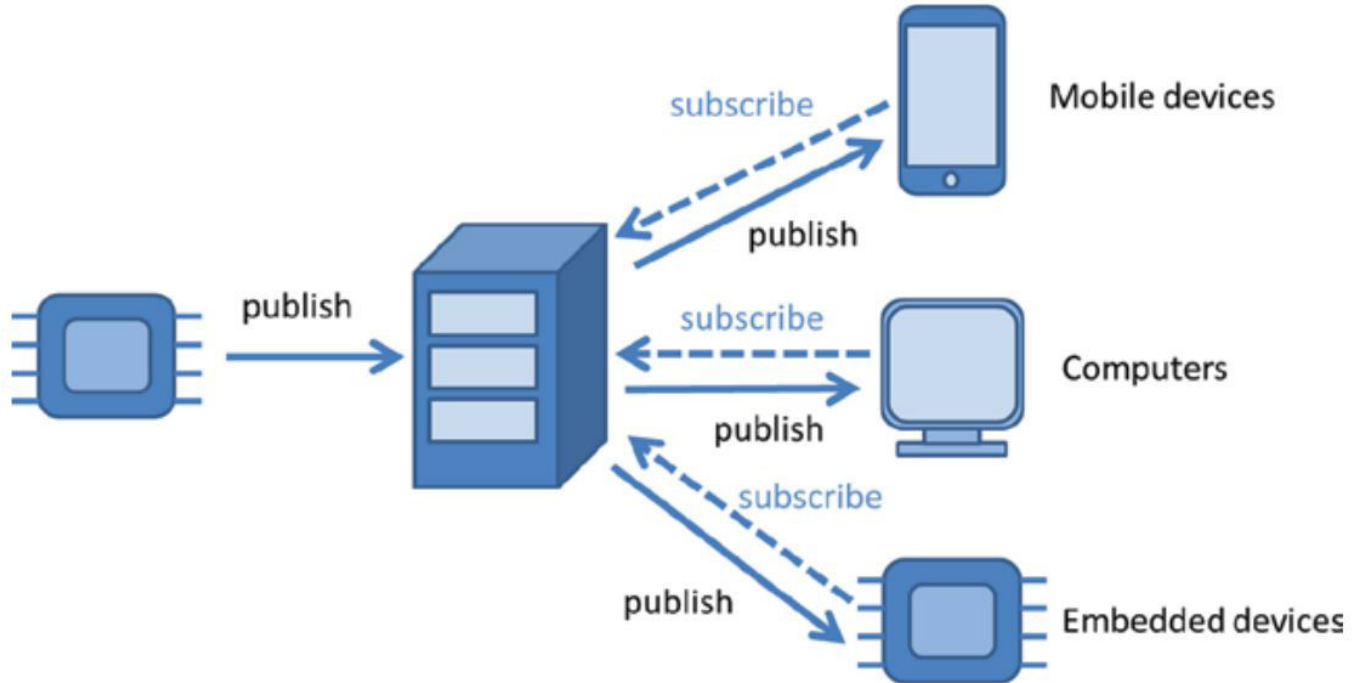
# Protocolos utilizados en IoT

**MQTT:** Este es un *protocolo de comunicación ligero*, diseñado para comunicaciones del tipo **M2M** (Máquina a Máquina). Este protocolo se encuentra basado en el modelo *Publicar-Suscribir* o (publisher - subscriber), en donde existen *dispositivos que publican datos y otros que se suscriben para recibir los datos*.

Para que se pueda establecer esta comunicación, existe un **broker** (server), el cual toma los datos del dispositivo que **publica** y los distribuye a aquellos dispositivos que se **suscriben**. A veces, el broker, *puede existir en la nube*.

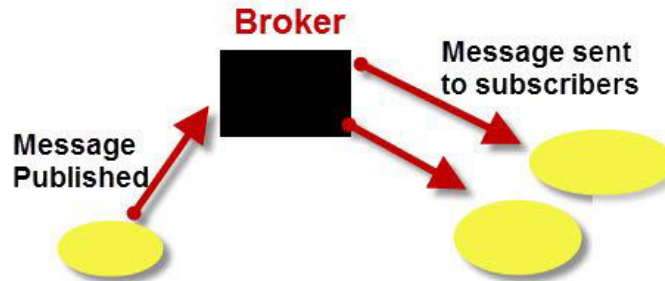
Esto se muestra a continuación.

# Protocolos utilizados en IoT



# Protocolos utilizados en IoT

La gran ventaja de **MQTT** sobre **HTTP** es la ligereza en la transmisión de datos, esto es sobre todo porque **HTTP** requiere de **mayor ancho de banda para transmitir los datos**. Hay que tener en cuenta que generalmente se usan **sistemas embebidos** para estas comunicaciones y, por lo general, son **sistemas de escasos recursos** por eso, para estas comunicaciones, resulta conveniente **MQTT**.



# Protocolos utilizados en IoT

**CoAP:** Este protocolo de aplicación restringida (CoAP, Constrained Application Protocol) se encuentra preparado para **dispositivos IoT restringidos**, es decir, **dispositivos con potencia, consumos y conectividad de red limitada**. Se basa en mensajes de **solicitud y respuesta, similar a HTTP, pero utiliza UDP**.

Aunque **UDP no proporciona transmisiones confiables**, es mucho más simple y tiene una sobrecarga mucho menor, por lo tanto, **es mucho más rápido**. [2]

Para más información: <http://coap.technology/>



# Planificación

- - ¿Qué es IoT? Aplicaciones en la vida cotidiana y la industria.
- - Protocolos utilizados en IoT.
- - **Introducción a Redes. Protocolos utilizados en internet.**
- - Introducción a módulos utilizados en IoT basados en IDE Arduino. ESP8266 y ESP32.
- - Conceptos de Software y Hardware Libre.

# Introducción a Redes

En base a lo comentado, podemos ver que **IoT** se trata de una **red de dispositivos conectados a través de Internet**. Por esta razón, resulta conveniente **estudiar el modelo que se utiliza en estas comunicaciones**.

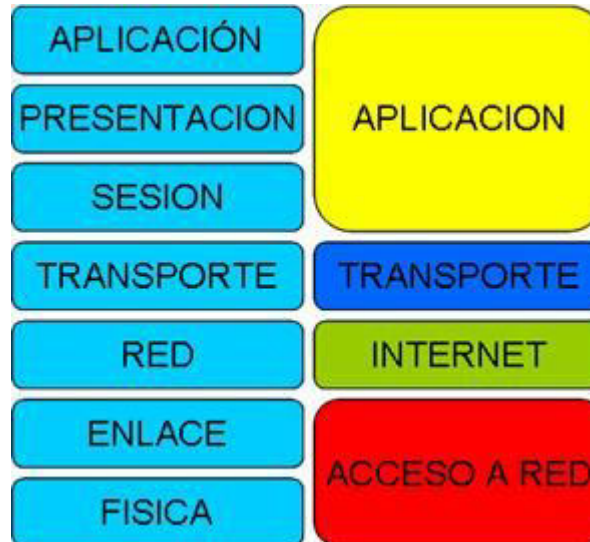
Este modelo comentado se denomina **TCP/IP**, el cual describe el **conjunto de reglas que permiten que un equipo pueda comunicarse en una red**.

Este modelo **especifica cuál es el formato** que deben tener los datos enviados.

# Introducción a Redes

Para poder comprender TCP/IP, se propone el siguiente modelo:

Capas OSI y TCP/IP



# Introducción a Redes

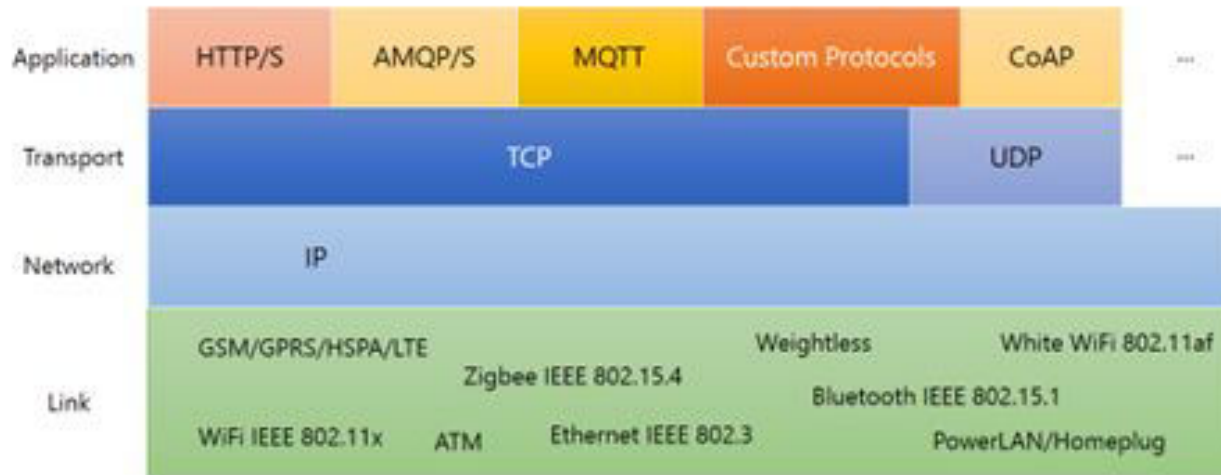
**Capa de Aplicación:** Utiliza los programas que *acceden servicios en la red*. Interactúan con uno o más protocolos de transporte para enviar o recibir datos. [17]

**Capa de Transporte:** Provee *comunicación extremo a extremo* desde un programa de aplicación a otro. *Puede proveer un transporte confiable asegurándose que los datos lleguen sin errores y en la secuencia correcta*. [17]

**Capa Internet:** Controla la comunicación entre un equipo y otro, *decide qué rutas deben seguir los paquetes de información para alcanzar su destino*. Conformar los **paquetes IP** que serán enviados por la capa inferior. [17]

**Capa de Acceso de Red:** Se ocupa de los recursos que utilizan los dispositivos para conectarse al medio de transmisión.

# Introducción a Redes



# Introducción a Redes

## Current Internet Protocols

## Expected IOT Protocols

HTTP FTP,SMTP,IMAP	Application	MQTT COAP,AMQP
TCP and UDP	Transport	UDP and TCP
IPv4 and IPv6	Networking	IPv6 and IPv4
Ethernet,Wi-Fi, GSM	Data Link	Ethernet,Wi-Fi, GSM, LTE-M, Lora, SigFox
Protocol Level TCP/IP Model		

[18]

IOT and Internet Protocols

# Introducción a Redes

## Nivel de Transporte: TCP y UDP:

El *protocolo UDP* es lo que se llama *un protocolo no orientado a conexión*. Por ejemplo, cuando una *máquina A envía paquetes* a una *máquina B*, *el flujo es unidireccional*.

Aquí, *se dice que la transferencia de datos se realiza sin haber hecho previamente una conexión con la máquina B*, de esta forma, el *destinatario recibirá los datos sin enviar una confirmación al emisor (la máquina A)*. [17]

# Introducción a Redes

## Nivel de Transporte: TCP y UDP:

*Al contrario que UDP, el protocolo TCP está orientado a conexión. Cuando una máquina A envía datos a una máquina B, esta última es informada de la llegada de datos, y confirma su buena recepción.*

TCP usa el concepto de *número de puerto para identificar a las aplicaciones emisoras y receptoras*. Cada lado de la conexión TCP tiene asociado un *número de puerto* (de 16 bits sin signo, con lo que existen 65536 puertos posibles) asignado por la aplicación emisora o receptora.



# Introducción a Redes

## Direccionamiento IP:

**TCP/IP** utiliza un identificador llamado **dirección IP**, cuya **longitud es de 32 bits**. Esta **IP identifica tanto a la red a la que pertenece la computadora como a ella misma dentro de dicha red**. Generalmente esta **IP** se representan **mediante cuatro octetos** (en formato decimal).

## Ejemplos:

**Clase A:** 10.0.0.0 a 10.255.255.255 (8 bits red, 24 bits hosts).

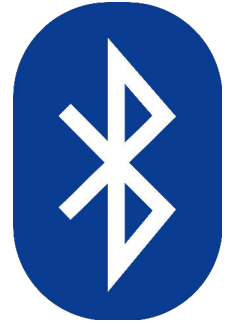
**Clase B:** 172.16.0.0 a 172.31.255.255 (16 bits red, 16 bits hosts). 16 redes clase B contiguas, uso en universidades y grandes compañías.

**Clase C:** 192.168.0.0 a 192.168.255.255 (24 bits red, 8 bits hosts). 256 redes clase contiguas, uso de compañías medianas y pequeñas además de pequeños proveedores de internet (ISP). **[17]**

# Introducción a Redes

**Acceso a Red:** Aquí podemos encontrar distintas tecnologías como:

- 1) **WiFi / 802.11.**
- 2) **Ethernet / 802.3** (Redes LAN).
- 3) **GSM** (Sistema Global de Comunicaciones Móviles, 2G).
- 4) **Bluetooth.**
- 5) **RFID** (Identificación por Radiofrecuencia).
- 6) **LoRa** (Long Range, tecnología inalámbrica, modulación en radiofrecuencia)



# Planificación

- - ¿Qué es IoT? Aplicaciones en la vida cotidiana y la industria.
- - Protocolos utilizados en IoT.
- - Introducción a Redes. Protocolos utilizados en internet.
- - **Introducción a módulos utilizados en IoT basados en IDE Arduino. ESP8266 y ESP32.**
- - Conceptos de Software y Hardware Libre.

# ESP8266 y ESP32.

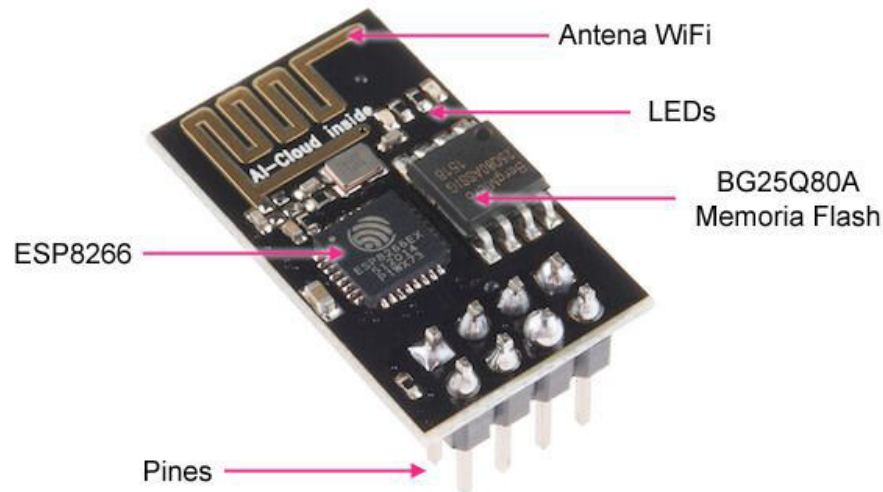
¿Qué es el ESP8266?



# ESP8266 y ESP32.

El **ESP8266** es un **SoC** (Sistem On Chip) que **contiene WiFi** y es compatible con el protocolo TCP/IP. El fabricante ofrece una serie de módulos con distintas capacidades:

## ESP-01



[4]

Tiene disponible **dos pines GPIO** digitales para controlar sensores y actuadores.

# ESP8266 y ESP32.

ESP-12

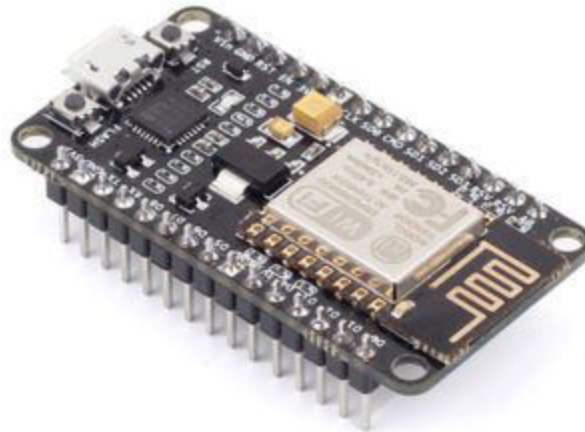


[4]

Tiene 11 puertos **GPIO** de los cuales uno, es **analógico** (10-bits).

# ESP8266 y ESP32.

NodeMCU

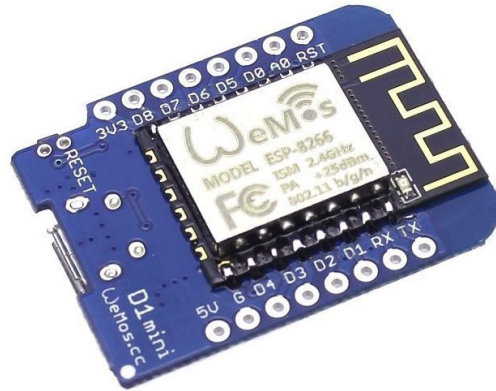


[4]

Tiene 11 puertos **GPIO** y uno **analógico** (10-bits).

# ESP8266 y ESP32.

## Mini Wemos D1



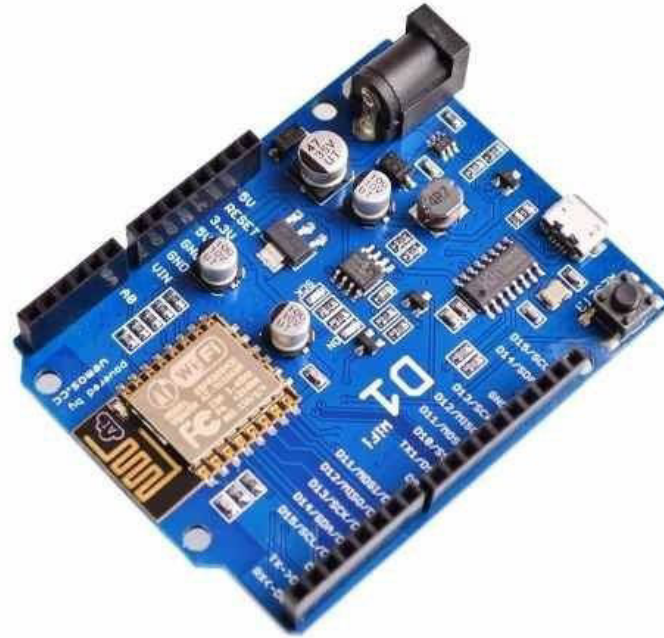
[4]

Tiene 11 puertos **GPIO** y uno **analógico** (10-bits).



# ESP8266 y ESP32.

Wemos D1

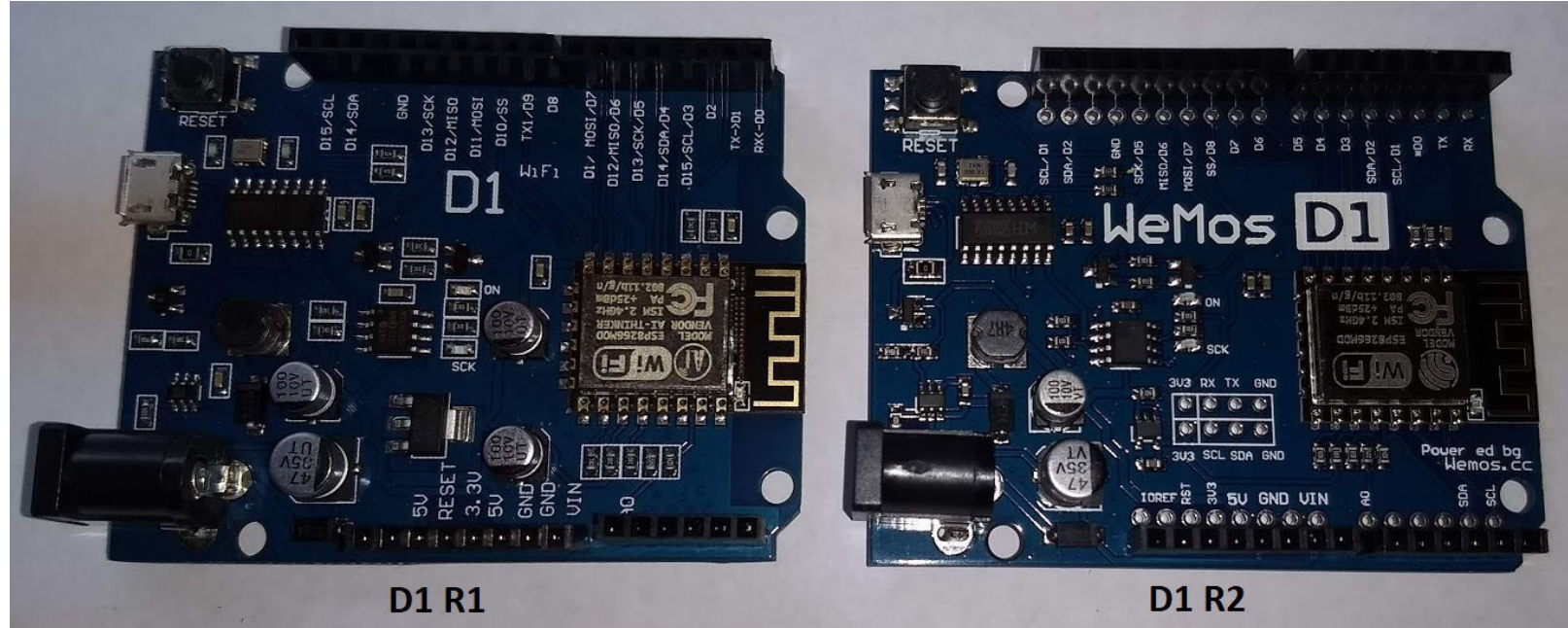


[23]

Tiene 11 puertos **GPIO** y uno **analógico** (10-bits).

# ESP8266 y ESP32.

Tener en cuenta que hay diferencias:



# ESP8266 y ESP32.

## Tener en cuenta que hay diferencias:

En la **D1 R1** las entradas/salidas digitales están numeradas desde **D0** hasta **D15**, pero en realidad, desde **D11** hasta **D15** están unidas a otros pines ya existentes.

En la **D1 R2** las entradas/salidas digitales están numeradas desde **D0** hasta **D8**

# ESP8266 y ESP32.

- ❖ Tensión de Alimentación: 5V DC.
- ❖ Tensión Entradas/Salidas: 3.3V DC.
- ❖ SoC: ESP8266 (Módulo ESP-12E).
- ❖ CPU: Tensilica Xtensa LX3 (32 bit).
- ❖ Frecuencia de Reloj: 80MHz/160MHz.
- ❖ Instruction RAM: 32KB.
- ❖ Data RAM: 96KB.
- ❖ Memoria Flash Externa: 4MB.
- ❖ Pines Digitales GPIO: 11 (pueden configurarse como PWM a 3.3V).
- ❖ Pin Analógico ADC: 1 (0-1V).
- ❖ UART: 1
- ❖ Chip USB: CH340G.
- ❖ Consumo corriente promedio: 70mA.

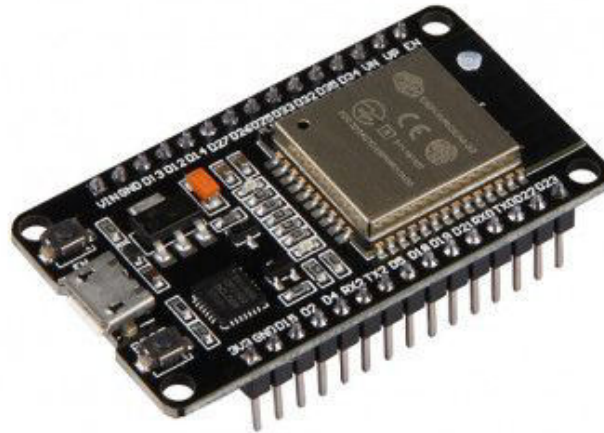
# ESP8266 y ESP32.

¿Qué es el ESP32?



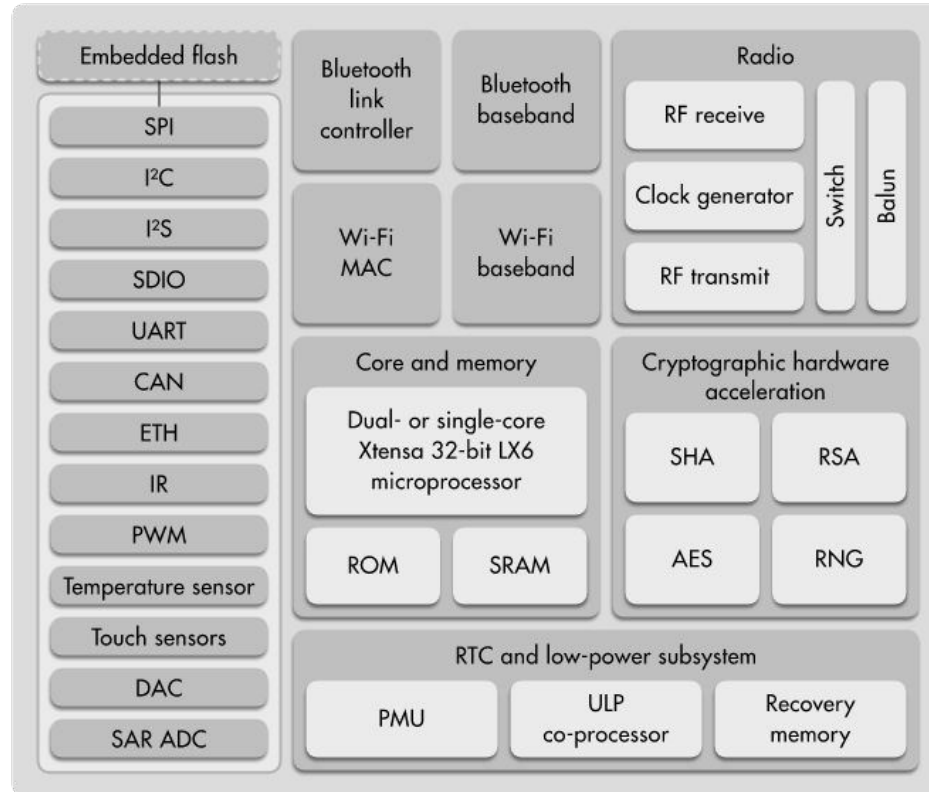
# ESP8266 y ESP32.

Creado por **Espressif Systems**, **ESP32** SoC (Sistem on Chip), de costos y consumos de energía reducidos, con la grandes capacidades de **Wi-Fi y Bluetooth**. [19]



# ESP8266 y ESP32.

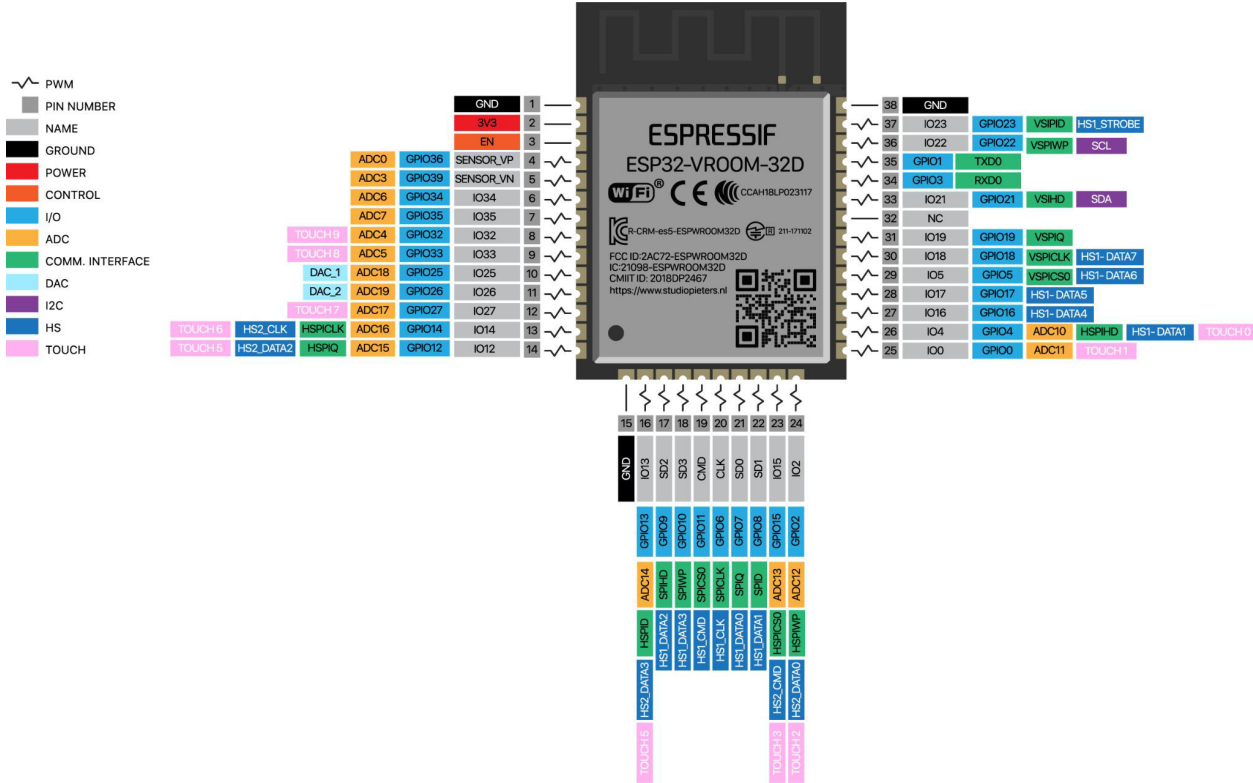
ESP32 FUNCTION BLOCK DIAGRAM



<http://esp32.net/>

<https://www.esp32.com/>

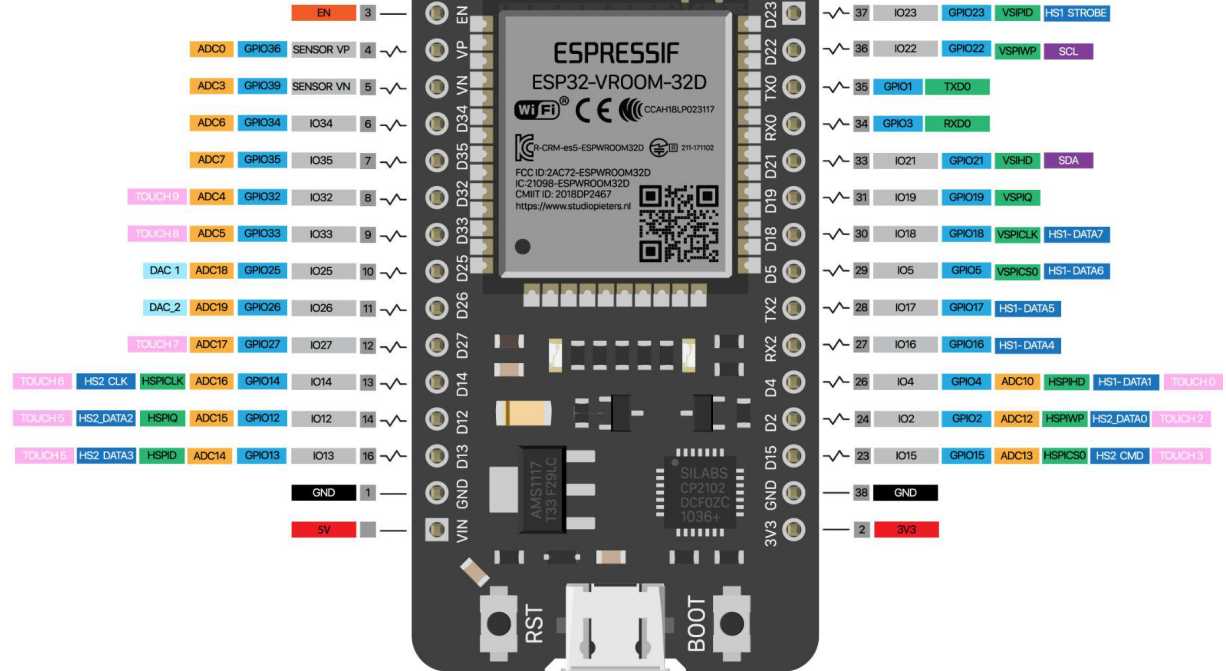
# ESP8266 y ESP32.





# ESP8266 y ESP32.

- PWM
- PIN NUMBER
- NAME
- GROUND
- POWER
- CONTROL
- I/O
- ADC
- COMM. INTERFACE
- DAC
- I2C
- HS
- TOUCH



# Planificación

- - ¿Qué es IoT? Aplicaciones en la vida cotidiana y la industria.
- - Protocolos utilizados en IoT.
- - Introducción a Redes. Protocolos utilizados en internet.
- - Introducción a módulos utilizados en IoT basados en IDE Arduino. ESP8266 y ESP32.
- - **Conceptos de Software y Hardware Libre.**

# Conceptos de Software y Hardware Libre.

Hoy en día, dado el gran avance en el Software y las grandes comunidades que se van formando, el Software libre se ve en ascenso.

El **software libre** ofrece al usuario cuatro libertades: **libertad de uso, de estudio y modificación, de distribución, y de redistribución de las versiones modificadas.**

Existen **licencias** que las garantizan y que dan una cobertura legal, como por ejemplo la licencia **GNU GPL**. El **hardware libre** toma estas mismas ideas del software libre para aplicarlas en su campo. [43]

Algunos ejemplos de Hardware Libre son: Arduino, Raspberry PI, e-puck, Reprap

# Planificación

- Introducción a IoT (Internet of Things)
- **Introducción a la programación de módulos basados en ESP 8266 mediante Arduino.**
- Prácticas con módulos basados en ESP8266.
- Comunicaciones WiFi con placas basadas en ESP8266.
- Integración de Conceptos.

# Introducción a los Sistemas Embebidos

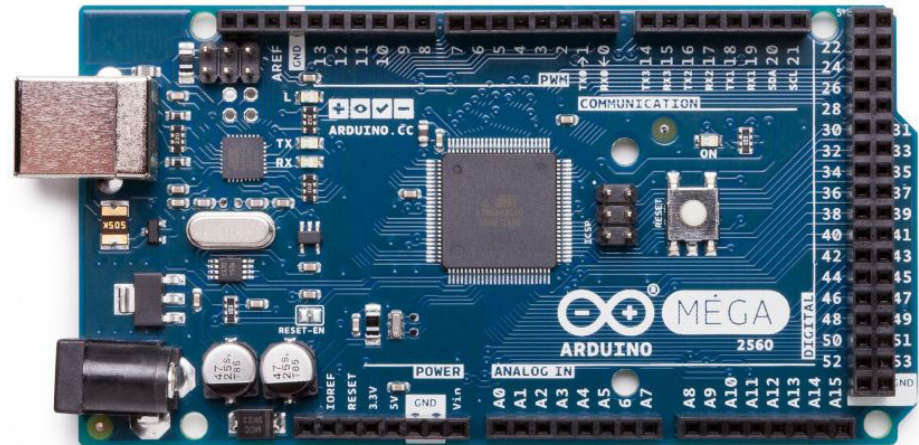
- - **Comparación de placas: Arduino UNO, ESP8266, ESP32. Pines de I/O digitales y analógicos.**
- - Instalación de IDE y módulos ESP 8266.
- - Configuración y puesta en marcha. Primer programa en IDE de Arduino con módulos basados en ESP 8266. Funciones setup() y loop().
- - Instrucciones pinMode() y digitalWrite(). Manejo del led on-board.
- - Variables: tipos de variables, rango, signo, definición, asignación.

# Plataformas de Desarrollo 8 bits

Arduino UNO

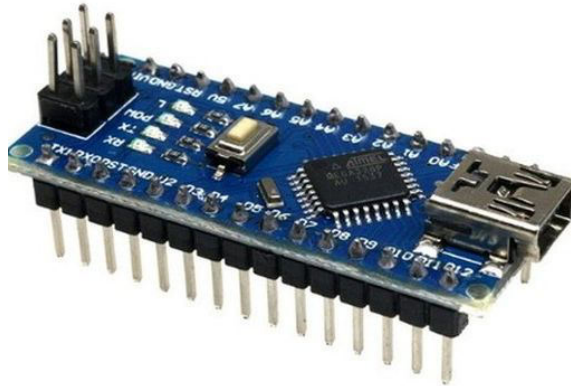


Arduino MEGA

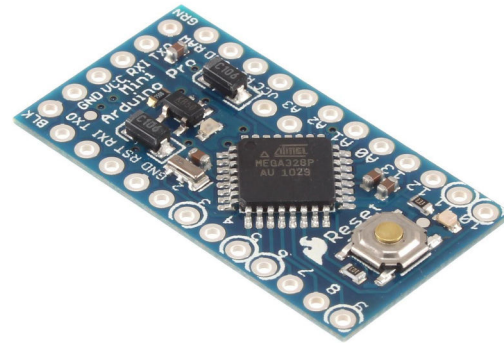


# Plataformas de Desarrollo 8 bits

Arduino NANO



Arduino Pro Mini



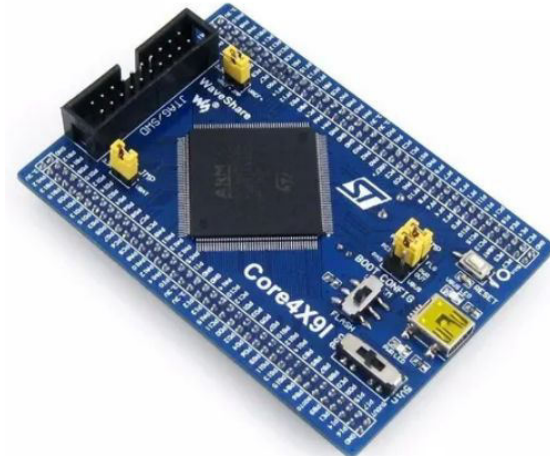
# Plataformas de Desarrollo 32 bits

STM32



STM32F103 - Cortex-M3

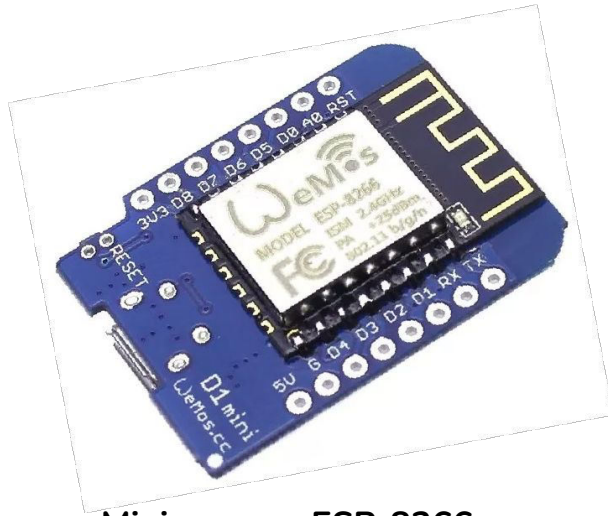
STM32



STM32f429 - Cortex M4



# Plataformas de Desarrollo con acceso WiFi



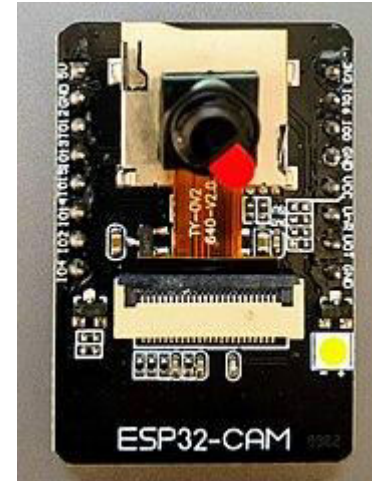
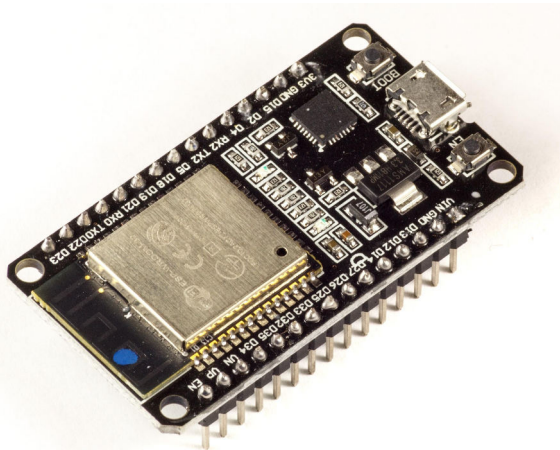
Mini wemos. ESP-8266



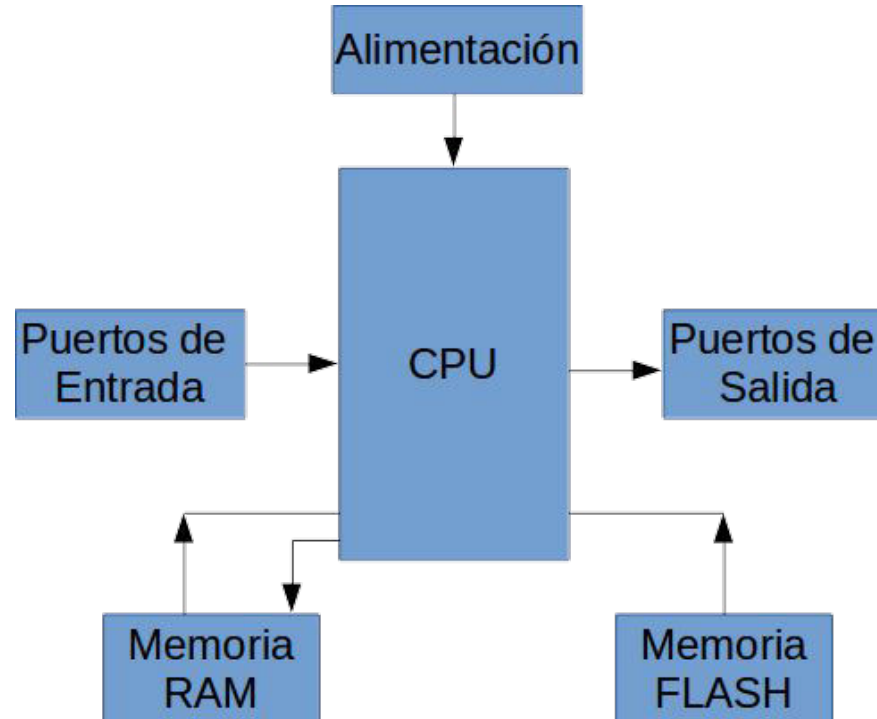
NodeMCU. ESP-8266

# Plataformas de Desarrollo con acceso WiFi

ESP32



# Arquitectura básica de una Placa de Desarrollo



# Introducción a los Sistemas Embebidos

- - Comparación de placas: Arduino UNO, ESP8266, ESP32. Pines de I/O digitales y analógicos.
- - **Instalación de IDE y módulos ESP 8266.**
- - Configuración y puesta en marcha. Primer programa en IDE de Arduino con módulos basados en ESP 8266. Funciones setup() y loop().
- - Instrucciones pinMode() y digitalWrite(). Manejo del led on-board.
- - Variables: tipos de variables, rango, signo, definición, asignación.

# Introducción a los Sistemas Embebidos

¿Qué es Arduino?



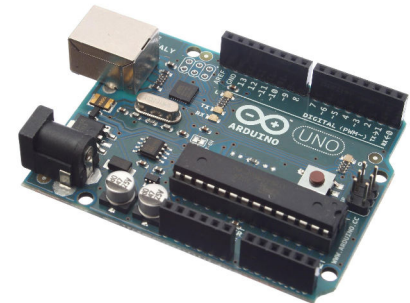
# Introducción a los Sistemas Embebidos

**Arduino** es una **plataforma electrónica** de **código abierto** (open source), basada en **hardware y software** que son sencillos de usar.

Las placas Arduino pueden leer entradas (sensores de luz, teclas, teclados táctiles), procesar estas entradas para luego, mediante sus salidas, activar algunos actuadores tales como motores u otro tipo de salidas como por ejemplo LEDs, balizas, pantallas LC

Para realizar esta lectura, procesamiento y activación, resulta importante hacerlo mediante el **lenguaje de programación Arduino** y el **Software Arduino (IDE)**.

<https://www.arduino.cc/en/Guide/Introduction>



# Introducción a los Sistemas Embebidos

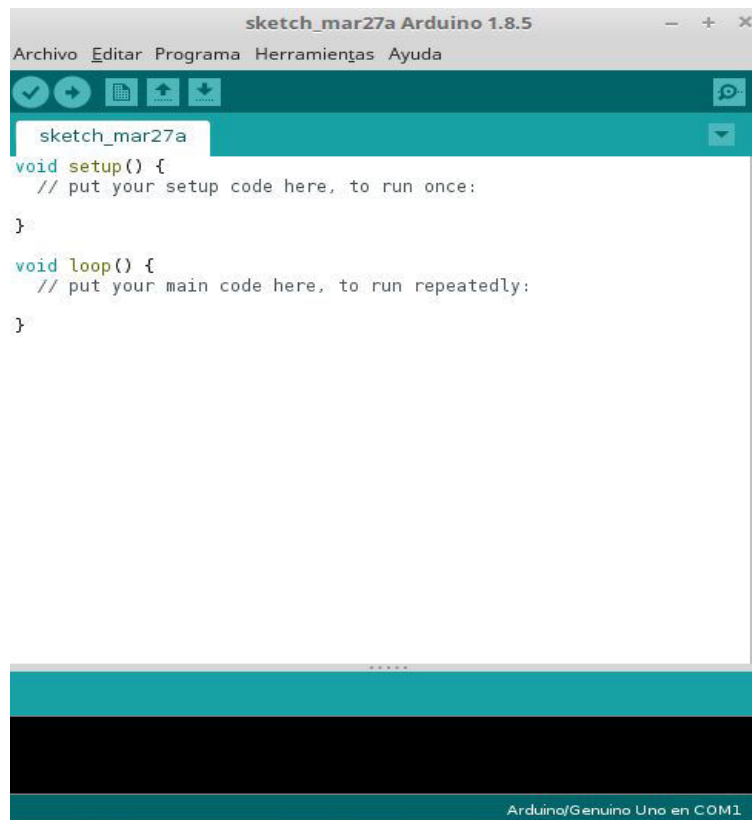
## Software de Arduino:

Para poder desarrollar código, generalmente se usa un **IDE** (Entorno de Desarrollo Integrado), el cual **contiene un editor de texto para escribir código, un área de mensajes, una consola de texto, una barra de herramientas con botones para funciones comunes y una serie de menús.**

Una vez escrito el código, se compila y se lo envía al **hardware Arduino** para **cargar programas y comunicarse con ellos.**

<https://www.arduino.cc/en/Guide/Environment>

# Introducción a los Sistemas Embebidos



```
sketch_mar27a Arduino 1.8.5
Archivo  _editar Programa  Herramientas  Ayuda
✓ → 📄 ⬆️ ⬇️ 🔍
sketch_mar27a
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}

Arduino/Genuino Uno en COM1
```



# Introducción a los Sistemas Embebidos

Algunas cosas importantes a la hora de trabajar con Arduino:

Los programas escritos con el **software Arduino** (IDE) se los llama **sketch**.

Estos **sketch** se escriben en el editor de texto y se los guarda con la extensión **.ino**.

La consola muestra la salida de texto del software Arduino (IDE), incluidos mensajes de error completos y otra información.

La esquina inferior derecha de la ventana muestra la placa configurada y el puerto serie. Los botones de la barra de herramientas le permiten verificar y cargar programas, crear, abrir y guardar **sketch** y abrir el monitor en serie.

<https://www.arduino.cc/en/Guide/Environment>

# Instalación IDE

Para poder instalar el IDE de Arduino conviene bajarlo de la página oficial:

<https://www.arduino.cc/en/software>

Como se puede ver de la página oficial, se puede usar tanto en **Windows, Linux** y **MAC OS**.

Luego de haberlo bajado, se puede comenzar con la instalación, la cual se puede consultar en:

<https://arduino-esp8266.readthedocs.io/en/latest/installing.html>

A continuación veremos paso a paso la instalación

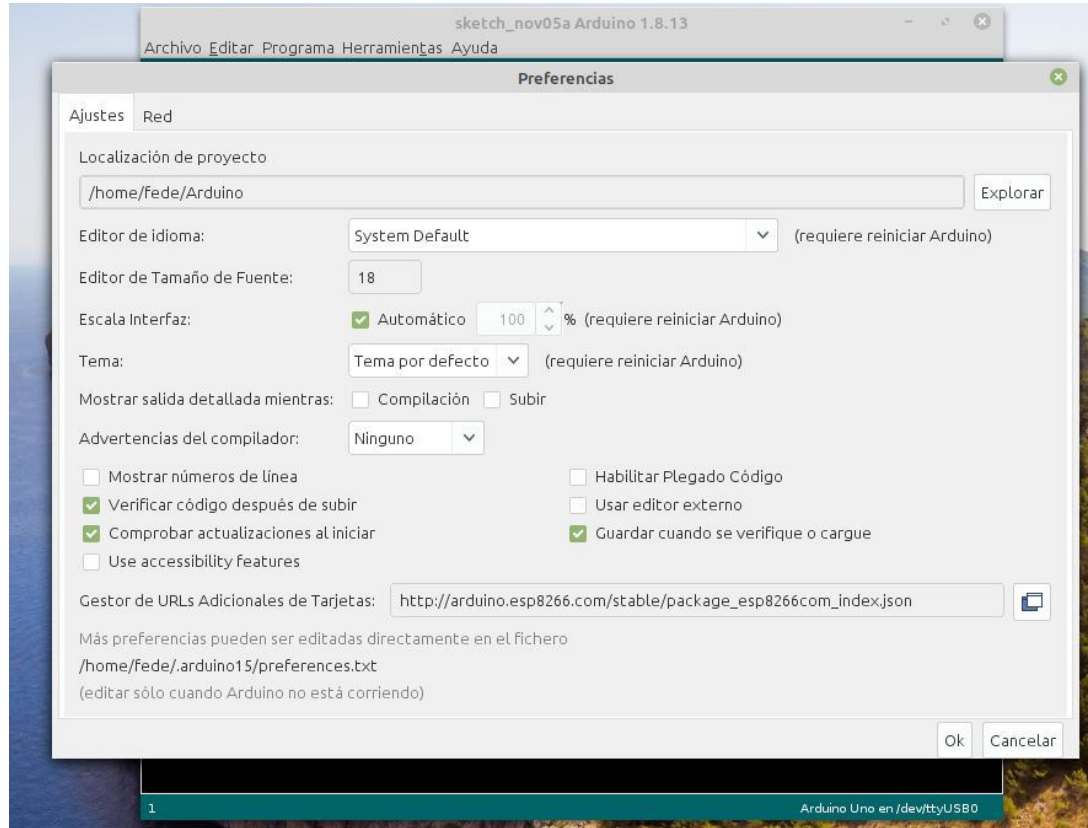
# Instalación IDE

En nuestro caso, para usar la placa **Wemos D1**, utilizaremos el **IDE de Arduino**. Para esto, a continuación se explican los pasos:

- 1) Abrir el IDE, ir a **Archivos**- > **Preferencias**
- 2) En el campo **Gestor de URLs Adicionales de Tarjetas**, ingresar la siguiente url:

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

# Instalación IDE



sketch\_nov05a Arduino 1.8.13  
Archivo Editar Programa Herramientas Ayuda

### Preferencias

Ajustes Red

Localización de proyecto  
/home/fede/Arduino Explorar

Editor de idioma: System Default (requiere reiniciar Arduino)

Editor de Tamaño de Fuente: 18

Escala Interfaz:  Automático 100 % (requiere reiniciar Arduino)

Tema: Tema por defecto (requiere reiniciar Arduino)

Mostrar salida detallada mientras:  Compilación  Subir

Advertencias del compilador: Ninguno

Mostrar números de línea  Habilitar Plegado Código  
 Verificar código después de subir  Usar editor externo  
 Comprobar actualizaciones al iniciar  Guardar cuando se verifique o cargue  
 Use accessibility features

Gestor de URLs Adicionales de Tarjetas: [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json) 📄

Más preferencias pueden ser editadas directamente en el fichero  
/home/fede/.arduino15/preferences.txt  
(editar sólo cuando Arduino no está corriendo)

OK Cancelar

1 Arduino Uno en /dev/ttyUSB0

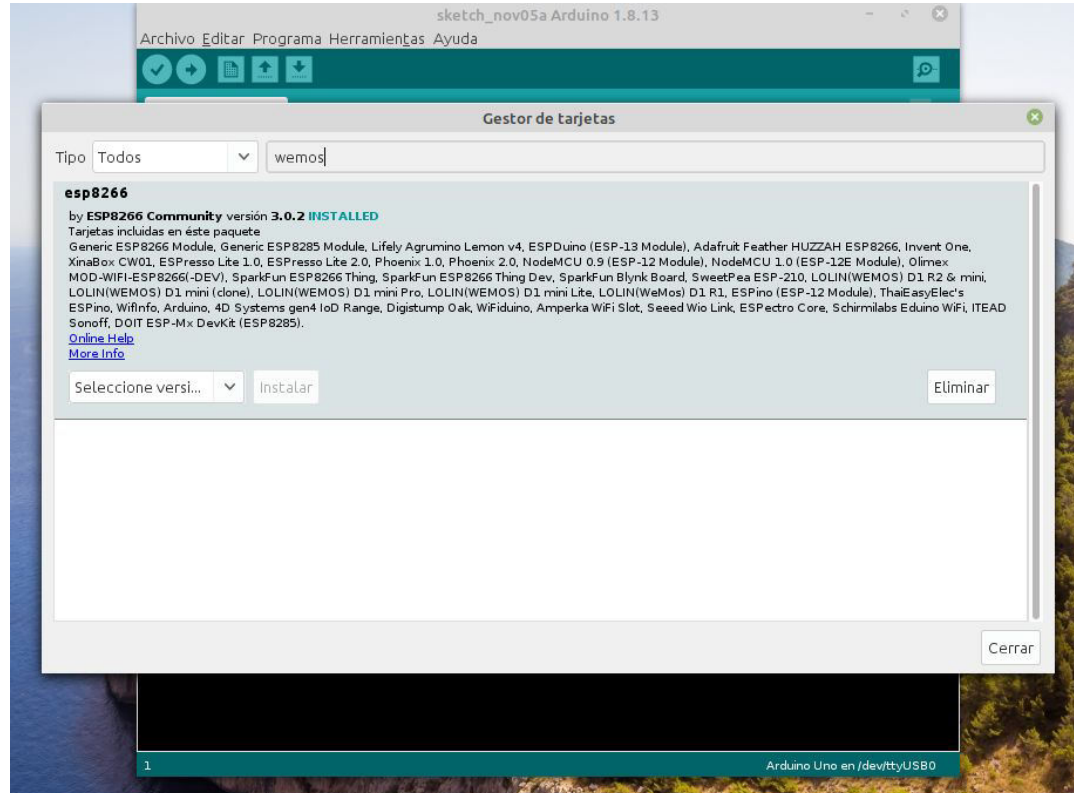
# Instalación IDE

3) Herramientas -> Placa -> Gestor de tarjetas.

En el buscador escribir: [wemos](#).

De allí, buscar la que dice: [esp8266 by ESP8266 Community](#).

# Instalación IDE



sketch\_nov05a Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

Gestor de tarjetas

Tipo Todos wemos

**esp8266**  
by **ESP8266 Community** versión **3.0.2** **INSTALLED**

Tarjetas incluidas en este paquete  
Generic ESP8266 Module, Generic ESP8285 Module, LilyFy Agrumino Lemon v4, ESPduino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invert One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SparkFun Blynk Board, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini (clone), LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, LOLIN(WeMos) D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifiInfo, Arduino, 4D Systems gen4 IoT Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core, Schirmilabs Eduino WiFi, ITEAD Sonoff, DOIT ESP-Mx DevKit (ESP8285).

[Online Help](#)  
[More Info](#)

Seleccione versi... Instalar Eliminar

Cerrar

1 Arduino Uno en /dev/tty/USB0

# Instalación IDE

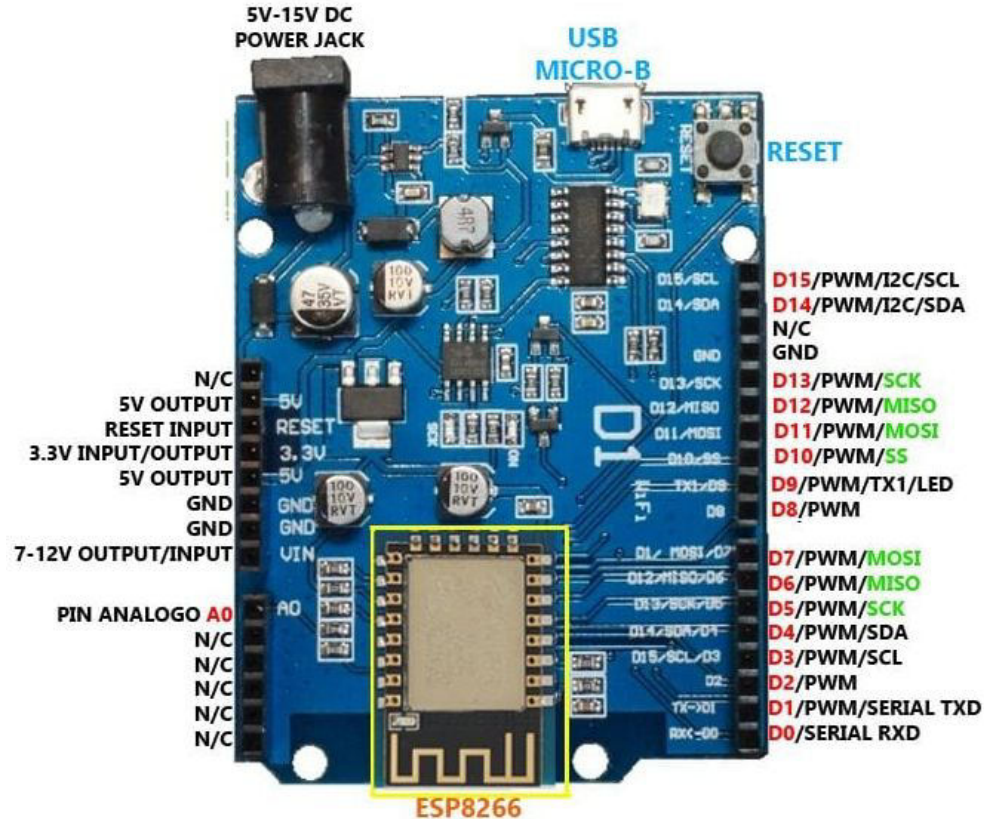
4) Por último, seleccionar la placa:

*Herramientas* -> *Placa* -> *Lolin(WEMOS) D1 R1*.

Previamente a realizar los primeros programas, conviene estudiar el **PINOUT**

# Pinout Arduino UNO

Pinout:





# Introducción a los Sistemas Embebidos

- - Comparación de placas: Arduino UNO, ESP8266, ESP32. Pines de I/O digitales y analógicos.
- - Instalación de IDE y módulos ESP 8266.
- - **Configuración y puesta en marcha. Primer programa en IDE de Arduino con módulos basados en ESP 8266. Funciones setup() y loop().**
- - Instrucciones pinMode() y digitalWrite(). Manejo del led on-board.
- - Variables: tipos de variables, rango, signo, definición, asignación.

# Ejemplo: Blinky LED

## Código

Ver: [Blink\\_Wemos.ino](#)

```
int led = 2;
```

```
void setup()  
{ pinMode(led, OUTPUT); }
```

```
void loop()  
{ digitalWrite(led, HIGH);  
  delay(1000);  
  digitalWrite(led, LOW);  
  delay(1000);  
}
```



# Introducción a los Sistemas Embebidos

- - Comparación de placas: Arduino UNO, ESP8266, ESP32. Pines de I/O digitales y analógicos.
- - Instalación de IDE y módulos ESP 8266.
- - Configuración y puesta en marcha. Primer programa en IDE de Arduino con módulos basados en ESP 8266. Funciones setup() y loop().
- - **Instrucciones pinMode() y digitalWrite(). Manejo del led on-board.**
- - Variables: tipos de variables, rango, signo, definición, asignación.

# Descripción de las funciones

## *pinMode():*

**Descripción:** Configura el pin especificado para que se comporte como una entrada o una salida.

**Sintaxis:** `pinMode(pin, mode)`

**Parámetros:**

**pin:** Número de pin con el que se desea trabajar

**mode:** INPUT o OUTPUT

**Devuelve:** Nada

# Descripción de las funciones

## *digitalWrite():*

**Descripción:** Escribe un valor ALTO o BAJO en la salida especificada

**Sintaxis:** digitalWrite(pin, value)

### **Parámetros:**

**pin:** Número de pin con el que se desea trabajar

**value:** HIGH o LOW

**Devuelve:** Nada

# Descripción de las funciones

## delay():

**Descripción:** Pausa el programa por la cantidad de tiempo (en milisegundos) especificado como parámetro

**Sintaxis:** delay(ms)

**Parámetros:**

**ms:** Número milisegundos que se quiere pausar

**Devuelve:** Nada

# Descripción de las funciones

*digitalRead()*:

**Descripción:** Lee el valor de un pin digital especificado, ya sea ALTO o BAJO.

**Sintaxis:** digitalRead(pin)

**Parámetros:**

**pin:** el número del pin digital que quieres leer

**Devuelve:** HIGH o LOW

# Introducción a los Sistemas Embebidos

- - Comparación de placas: Arduino UNO, ESP8266, ESP32. Pines de I/O digitales y analógicos.
- - Instalación de IDE y módulos ESP 8266.
- - Configuración y puesta en marcha. Primer programa en IDE de Arduino con módulos basados en ESP 8266. Funciones setup() y loop().
- - Instrucciones pinMode() y digitalWrite(). Manejo del led on-board.
- - **Variables: tipos de variables, rango, signo, definición, asignación.**



# Algunos conceptos previos...

Las computadoras trabajan en base a *números binarios*, esto es, utilizan *base dos*.

Por ejemplo, un número binario se puede representar como: **0101**

Cada *dígito* representa un *bit*.

Cada número binario se puede representar en otra base, por ejemplo, *en base diez o en base hexadecimal*.

**Ejemplo:** 10 (decimal) -> 1010 (binario) -> A (hexadecimal)

A los efectos de la programación que utilizaremos, podremos escribir valores en decimal ya que la computadora se podrá encargar de realizar la conversión.

# Algunos conceptos previos...

Para hacer conversiones entre números, usaremos la siguiente tabla

Decimal	Binario	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

# Variables

**Variable:** es una forma de *nombrar y almacenar un valor* para su uso posterior por el programa. Se puede guardar el número de pin a usar, datos de un sensor o el valor de un cálculo [10].

Antes de que se utilicen, todas las variables *deben declararse*. Declarar una variable significa *definir su tipo* y, *opcionalmente, establecer un valor inicial* (inicializar la variable). Las variables no tienen que inicializarse (asignar un valor) cuando se declaran, *pero a menudo es útil hacer esto*. [10].

**Ejemplo:**

```
int Variable1;  
int Variable2 = 0; // Ambas son correctas
```

# Variables

## *Algunos tipos de variables:*

- 1) **char:** Es un tipo de datos que **ocupa 1 byte** de memoria que almacena un valor de caracter. **Con signo.**
- 2) **int:** Se utiliza principalmente para el almacenamiento de números enteros. En el Arduino Uno, un **int** almacena un valor de 16 bits (**2 bytes**). **Con signo**
- 3) **unsigned int:** Almacena números **sin signo** y **ocupa 2 bytes**.
- 4) **float:** Es un tipo de datos para números de coma flotante, un número que tiene un punto decimal. **Ocupa 4 bytes.**

# Variables

Algo **importante** a tener en cuenta es la **cantidad de combinaciones posibles** que podemos tener para cada **variable**. Por ejemplo, en el caso de **char**, dado que tenemos **8 bits**, podemos representar:

$$\text{combinaciones posibles} = 2^n$$

n = cantidad de bits.

**Ejemplo.** para char, tenemos:

$$\text{combinaciones posibles} = 2^8 = 256$$

Tenemos **256 combinaciones posibles**, desde **0 - 255**.

# Planificación

- Introducción a IoT (Internet of Things)
- Introducción a la programación de módulos basados en ESP 8266 mediante Arduino.
- **Prácticas con módulos basados en ESP8266.**
- Comunicaciones WiFi con placas basadas en ESP8266.
- Integración de Conceptos.

# Introducción a los Sistemas Embebidos

- - **Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.**
- - Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.
- - Manejo de puertos digitales. Utilización de Relays.
- - Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.
- - Manejo de servo motor.
- - Dispositivos I<sup>2</sup>C: Display LCD.

# UART

Una de las ventajas que tiene el **ESP8266** es la de poder comunicarse con el mundo mediante mediante puertos de entradas y salidas.

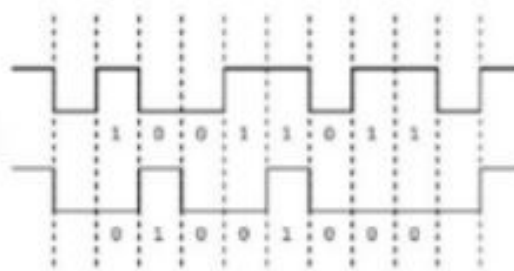
Uno de los puertos que permite intercambiar información mediante esta comunicación serie, es el **puerto Serie**. Cuando hablamos de una comunicación serie nos referimos a **transmitir** y **recibir** datos de **forma secuencial**.

Para realizar esta **comunicación**, el **ESP8266** dispone de un módulo llamado **UART** (Universal Asynchronous Receiver-Transmitter), para el cual se necesitan dos líneas: **Tx** y **Rx**.



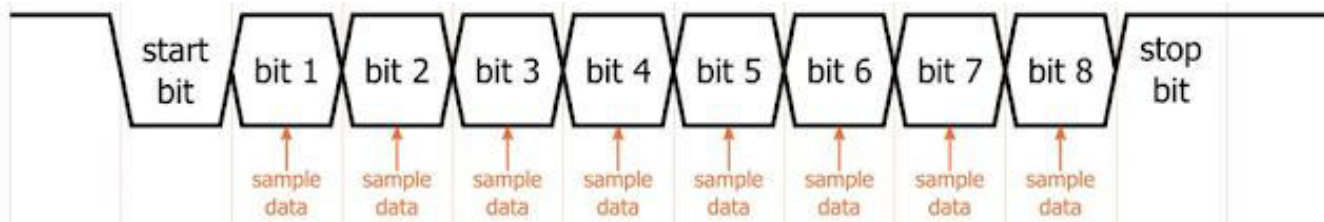
# UART

En nuestro caso vamos a utilizar con frecuencia a este puerto para comunicarnos con la PC y mostrar datos, por ejemplo, los que provienen de sensores



# UART

Esta comunicación se hace mediante una trama como la que se ve a continuación:



[12]

Siendo:

**bit Start**: inicia la comunicación, **bit 1-8**: datos, **bit stop**: finaliza la comunicación

# UART

## Configuración mediante el IDE:

Para la comunicación, hay que **inicializar el módulo**, el cual lo hacemos mediante:

```
Serial.begin();
```

En el argumento de esta función podemos setear dos parámetros:

**speed**: velocidad en bits por segundos (baudios). Generalmente se usa 9600 bits/s, aunque se pueden usar otras velocidades como 38400, 19200, etc.

**config**: nos permite setear la paridad, datos, bit de stop

# UART

## Configuración mediante el IDE - Envío de datos

Luego para enviar datos, se puede usar:

**`Serial.print()`**

Esta función permite imprimir datos en el puerto serie como texto ASCII.

En el argumento de la función se coloca el dato que se quiere enviar.

Devuelve la cantidad de bytes escritos

<https://www.arduino.cc/reference/en/language/functions/communication/serial/print/>

# UART

## Configuración mediante el IDE - Recepción de datos

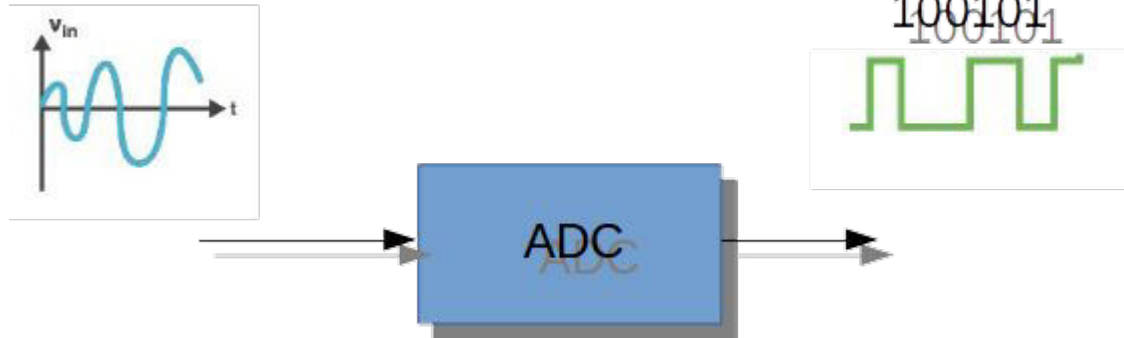
Para leer datos se puede usar `Serial.read()`, la cual es una función que **devuelve el primer byte de la comunicación.**

Es importante, antes de leer el dato, verificar que hay datos disponibles. Para eso se puede usar la función: `Serial.available()`, la cual obtiene la cantidad de bytes disponibles para leer.

# Convertor ADC

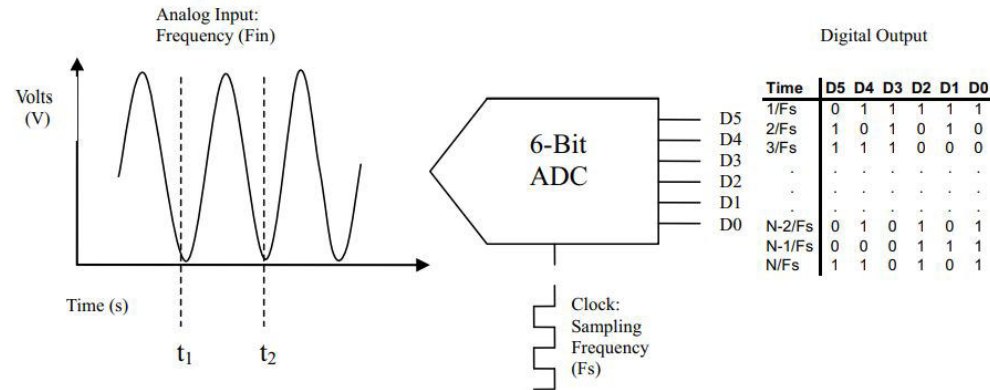
Convierte una **señal analógica** como por ejemplo, la señal de voz, en una **señal digital** compuesta por “unos” y “ceros”.

Cada **valor analógico** se corresponde con una **secuencia digital**.



# Conversor ADC

En la Figura se observa una **señal analógica** aplicada en la entrada de un **ADC**, la cual luego, **se convierte en palabras digitales mediante la frecuencia de muestreo ( $F_s$ )** aplicada al reloj ADC. Esta representación resulta ser en función del tiempo.



$F_{in}$ : Analog Input Frequency =  $1/(t_2 - t_1)$   
 $F_s$ : Clock Frequency  
 N: number of digital samples captured  
 n: number of output bits; in this 6-bit ADC example n = 6

# Convertor ADC

Dentro del ESP8266 se encuentra un **convertor ADC de 10 bits**. Dado que este **SOC** trabaja con **tensiones entre 0 y 1,1V**, cada uno de los **valores analógicos** que ingrese al convertor, se corresponderá con un **valor digital**, que se corresponderá a un valor en decimal.

## Ejemplo:

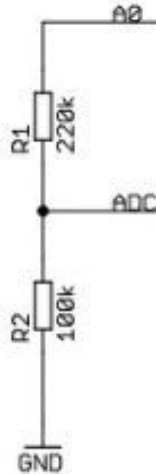
- 1) Si ingresan 1,1V al convertor ADC, este valor se corresponderá con el valor binario 1111111111, que pasandolo a decimal, corresponde al valor 1023.
- 2) Si ingresan 0,55V al convertor ADC, este valor se corresponderá con el valor binario 0111111111, que pasandolo a decimal, corresponde al valor 511.

Arduino podrá mostrar simplemente el valor en decimal, para hacer más simple el análisis



# Convertor ADC

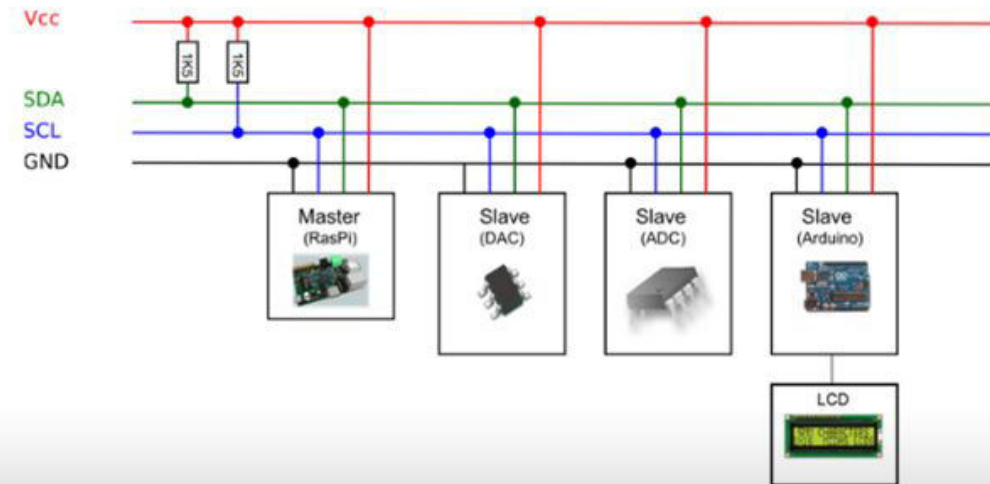
Es importante tener en cuenta que **el convertor ADC del ESP8266 soporta hasta 1,1V**. Sin embargo, en la placa, se encuentra un divisor resistivo de tensión el cual admite una **tensión de 3,3V a la entrada**.



Con esto, hay que tener en cuenta entonces que **la máxima tensión analógica de entrada que se le puede suministrar a la Wemos D1 es de 3,3V**.

# Interfaz I<sup>2</sup>C

El bus I<sup>2</sup>C es un protocolo de comunicación que utiliza dos “hilos” o cables para la comunicación. Esto es, utiliza una línea para los **datos (SDA)** y otra para **sincronización (SCL)**.



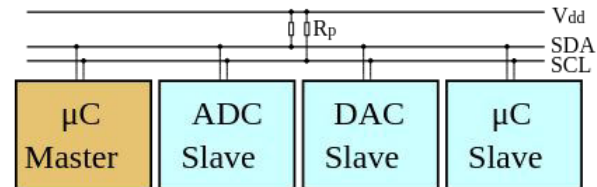
# Interfaz I<sup>2</sup>C

De la imagen anterior se ve que en el mismo bus, puede haber **varios dispositivos conectados**. Si se envía un mensaje a través del bus, éste tiene que ser dirigido a uno de los dispositivos.

Generalmente, quien **envía el dato** se llama **“Master”** y el/los que **reciben el dato** se llaman **“Slave”**.

El dispositivo **“Master”** es el que inicia la el envío de datos (conversación) y por ende quien envía las **señales de clock** (SCL).

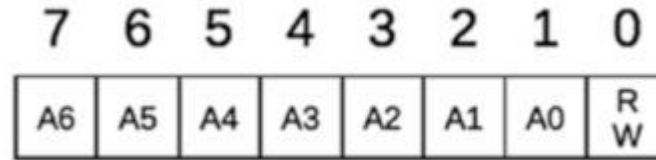
Cada **esclavo** tiene una **dirección de 7 bits** lo cual le permite al **Master**, saber a quién le va a enviar el dato/mensaje.



# Interfaz I<sup>2</sup>C

Siguiendo la idea anterior de la dirección, veamos cómo se direcciona

[5]



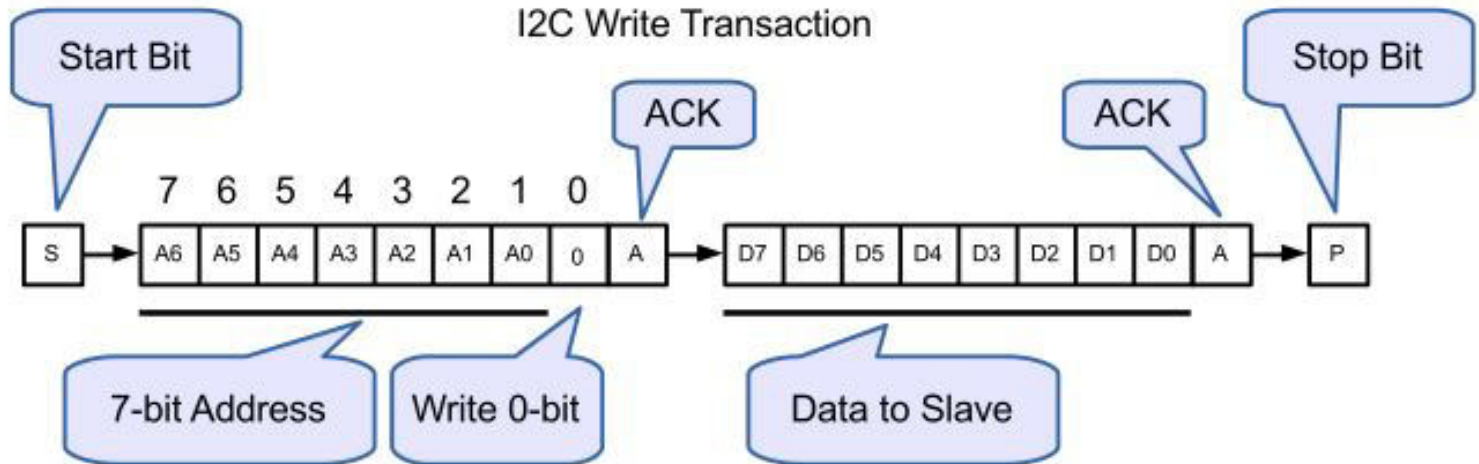
De la imagen anterior se puede ver que se usan los **7 bits más significativos para la dirección del esclavo** y el **bit "0"** se usa para **leer/escribir (R/W)**.

Si **R/W = 1**, indica operación de **lectura** desde el esclavo.

Si **R/W = 0**, indica operación de **escritura** sobre el esclavo.

# Interfaz I<sup>2</sup>C

En base a la idea anterior, se pueden definir los mensajes de *Escritura/Lectura*



# Interfaz I<sup>2</sup>C

De la imagen anterior se ve que, para el **proceso de escritura**, se necesitan los siguientes bits y bytes:

**Bit de Start:** Indica el comienzo de la comunicación.

**7-bits Address:** Indica la dirección del dispositivo a escribir (7-1).

**Write bit:** Tiene que estar en 0 para escribir (Write).

**ACK:** Bit de Acknowledge, bit de chequeo.

**Data to Slave:** Byte de dato a enviar.

**ACK:** Bit de Acknowledge, el dispositivo esclavo reconoce la solicitud

**Bit de Stop:** Fin de la comunicación.

# Interfaz I<sup>2</sup>C

En el protocolo de **lectura** se da una secuencia similar a la de **escritura**, simplemente que hay que **modificar** el **bit R/W** para que esté en “**1**” ya que el mismo indicará la **lectura** desde el dispositivo esclavo.

En arduino utilizaremos la biblioteca “**Wire.h**”

Las grandes **ventajas** de este protocolo es que necesita **pocos líneas** para la comunicación y **cada dispositivo tiene su propia dirección**.

**Mas información:** <https://www.i2c-bus.org>

# Referencias para funciones

Es importante saber que la **Referencia** para el manejo de puertos se puede encontrar en:

<https://arduino-esp8266.readthedocs.io/en/latest/reference.html#>

**Nota:** Algo importante a destacar es que la implementación de I<sup>2</sup>C en el ESP8266 se realiza mediante software. Para más información, ver:

[https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf)

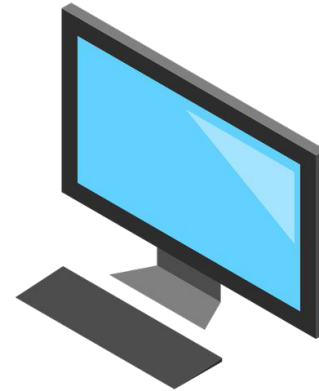
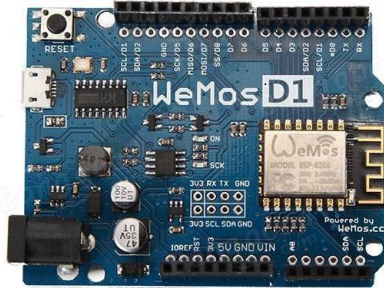


# Introducción a los Sistemas Embebidos

- - Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.
- - **Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.**
- - Manejo de puertos digitales. Utilización de Relays.
- - Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.
- - Manejo de servo motor.
- - Dispositivos I<sup>2</sup>C: Display LCD.

# Puerto Serie

En esta práctica se comunicarán la Wemos con la PC. De esta forma se podrán enviar y recibir datos.



# Puerto Serie

Para ver la aplicación, vamos a usar dos sketch:

- 1) UART\_envio.ino
- 2) UART\_recep.ino

Para esto, se debe conectar el cable desde Arduino hacia la PC

**Nota:** Algo importante a destacar es que el **ESP8266** tiene dos interfaces: **UART0** y **UART1**. Las transferencias de datos hacia / desde interfaces **UART** se pueden implementar a través de **hardware**. Para más información, ver:

[https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf)

# Introducción a los Sistemas Embebidos

- - Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.
- - Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.
- - **Manejo de puertos digitales. Utilización de Relays.**
- - Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.
- - Manejo de servo motor.
- - Dispositivos I<sup>2</sup>C: Display LCD.

# Relays

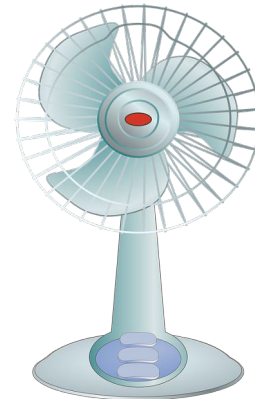
*¿Qué es un Relay?*



# Relays

Muchas veces cuando trabajamos con **Sistemas Embebidos**, resulta importante poder manejar **equipos eléctricos** como por ejemplo, **luminarias, ventiladores, persianas, etc.**

El ejemplo más claro se puede dar cuando queremos **encender luces de alguna habitación**. Aquí, hay que tener en cuenta que **las lámparas se conectan a los 220V**, con lo cual, con lo que conocemos hasta ahora, no se podría realizar.



# Relays

Para llevar a cabo esta tarea, podemos usar un **interruptor** que se **accione mediante los pulsos enviados desde la placa** (Arduino UNO, Wemos, PIC, etc). Al enviarle un **HIGH** ("1"), a este **interruptor se cierra** y, si enviamos un **LOW** ("0"), el **interruptor se abre**.

Lo interesante es **conectar este interruptor en serie con la lámpara** para que la misma se pueda encender. Sin olvidar de suministrar los 220V.

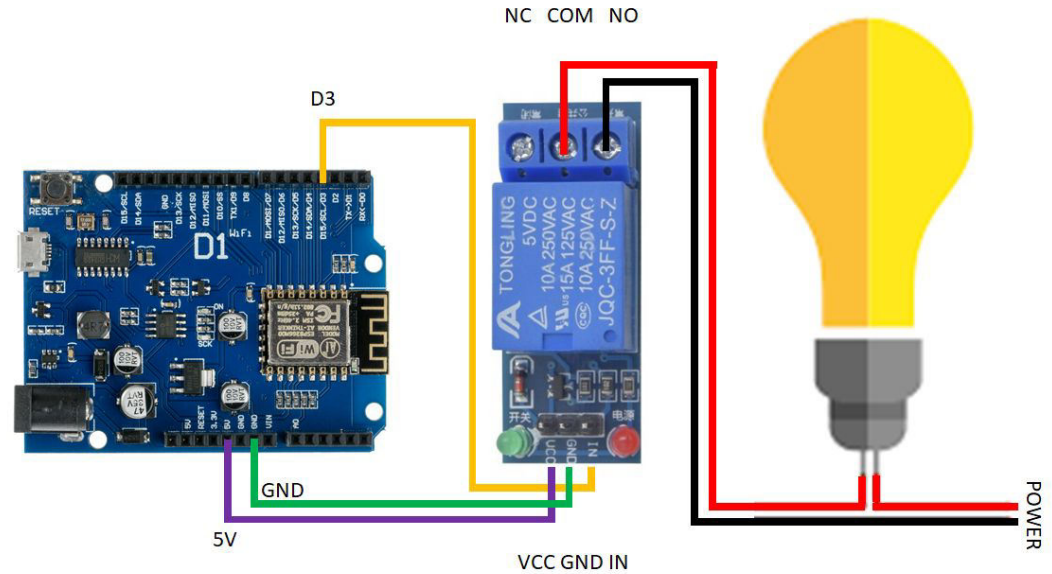
**Atención:** Puede haber riesgo eléctrico, manejar con precaución.



# Relays

El interruptor mencionado

es un **Relay**.





# Relays

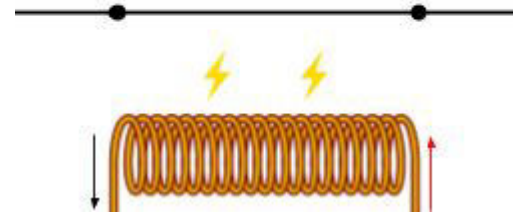
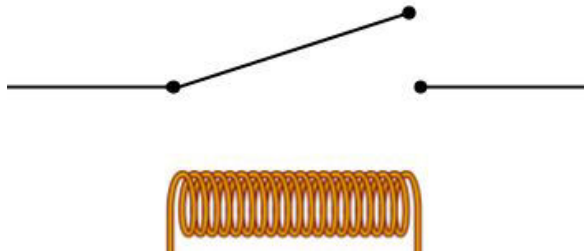
En la entrada del **Relay**, existe un **bobinado** tal que, al **energizarlo**, **acciona al interruptor**.



Estos módulos vienen con un **transistor** para activar a la bobina del **Relay**.

# Relays

Existen diversos modelos, algunos de ellos son los **Normalmente Abiertos (NO)**, los cuales hacen referencia a que, **cuando le enviamos un pulso, cierran sus contactos. De otra forma se mantienen abiertos.**



# Relays

También existen los **Normalmente Cerrados (NC)**, los cuales hacen referencia a que, **cuando le enviamos un pulso, abren sus contactos. De otra forma se mantienen cerrados.**



# Relays

A la hora de elegir el **Relay**, hay que tener en cuenta algunas **especificaciones técnicas**:

**Tensión de activación:** Hace referencia a la tensión con la que se debe activar a la bobina del Relay. Si bien la misma genera un campo magnético a través de una corriente, es necesario alimentarla. Para esto existen tensiones de 3V, 5V, 6V, 12V, 48V, etc

**Corriente y tensión máxima:** Estos parámetros hacen referencia a los contactos del mismo. Algunos niveles de corriente pueden ser de 10A o 20A. Para tensiones pueden ser de 220V.

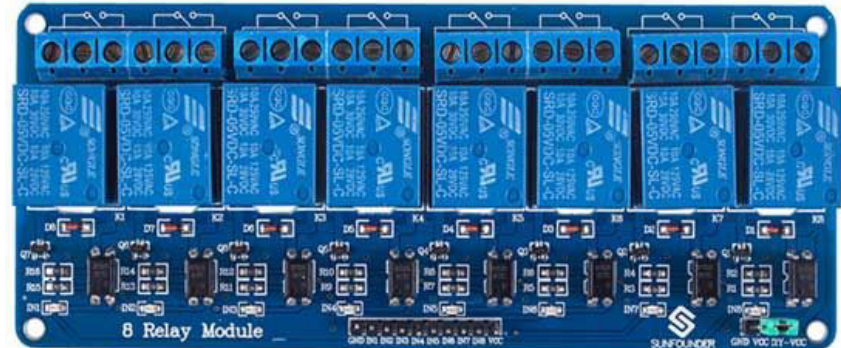
# Relays

Los Relays comentados son del tipo **electromecánicos**, existen también los de **estado sólido**.



# Relays

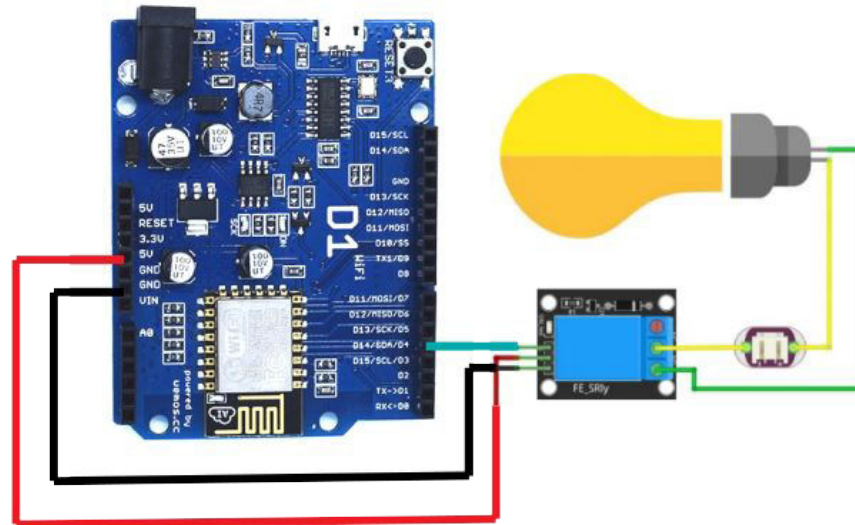
De los Relays electromecánicos, existen diversos módulos con distintas cantidades



# Relays

Para llevar estos conocimientos a la práctica, vamos a operar el **Relay** para que **apague y encienda un relay cada dos segundos**. El hardware es el siguiente:

**Ver:** [Relay\\_Wemos.ino](#)



# Introducción a los Sistemas Embebidos

- - Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.
- - Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.
- - Manejo de puertos digitales. Utilización de Relays.
- - **Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.**
- - Manejo de servo motor.
- - Dispositivos I<sup>2</sup>C: Display LCD.

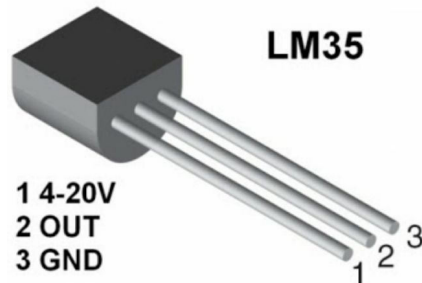


# Lectura con sensor LM35

El sensor de temperatura **LM35** convierte la temperatura en valores de tensión.

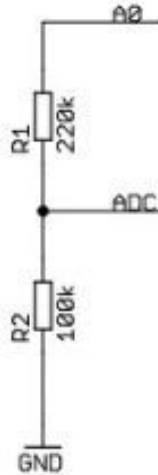
Por cada grado centígrado, entrega una tensión de 10mV. Por ejemplo, si en el interior de una habitación existen 20°C, el sensor entregará una tensión de 200mV.

**Rango de Temperatura:** -55°C a 150°C



# Lectura con sensor LM35

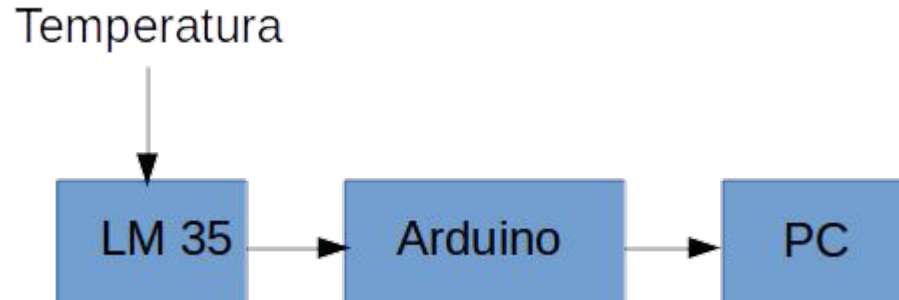
Es importante tener en cuenta que *el conversor ADC del ESP8266 soporta hasta 1,1V*. Sin embargo, en la placa, se encuentra un divisor resistivo de tensión el cual admite una tensión de 3,3V a la entrada.



Con esto, hay que tener en cuenta entonces que *la máxima tensión analógica de entrada que se le puede suministrar a la Wemos es de 3,3V*.

# Lectura con sensor LM35

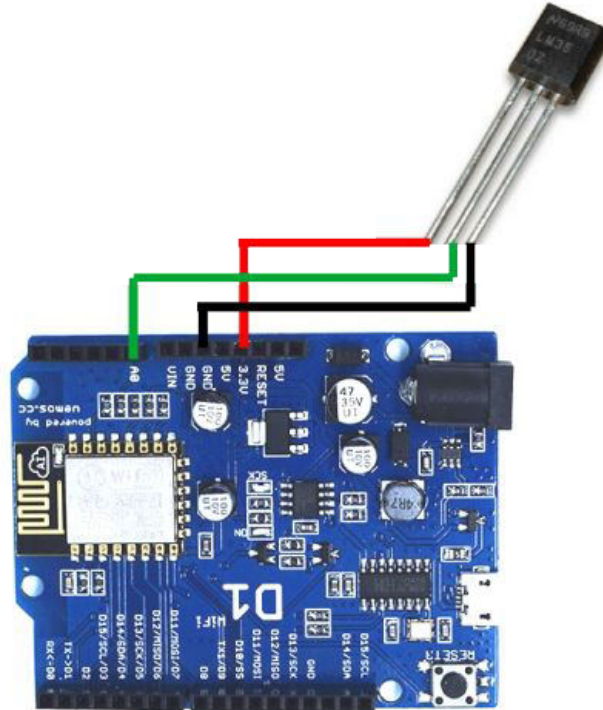
**Ejemplo:** El ejercicio consistirá en leer desde Arduino, la información suministrada por el sensor de temperatura y mostrarlo por pantalla.



Ver: [LM35\\_Wemos.ino](#)

# Lectura con sensor LM35

Hardware



# Lectura con sensor DHT 11

Sensor **digital** de Temperatura y Humedad.  
Dentro del dispositivo, se encuentra un  
sensor analógico. Luego, se convierte  
esta temperatura a un valor digital.

**Alimentación:** 3,5V a 5V

**Temperatura:** 0°C a 50°C

**Humedad:** 20%RH a 90%RH

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND

[7]

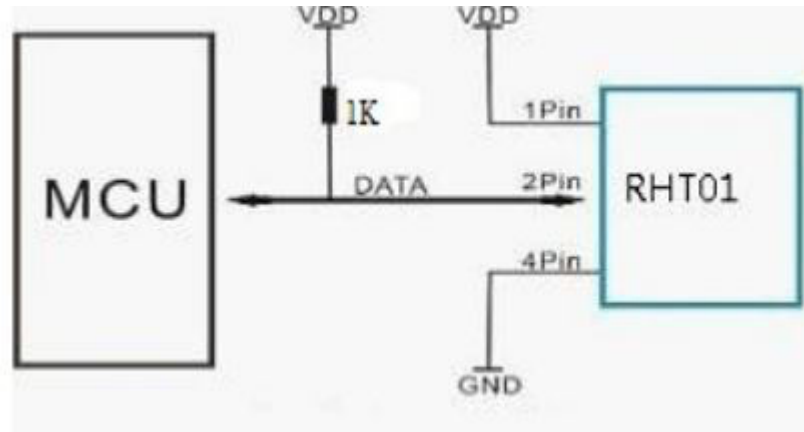


# Lectura con sensor DHT 11

Model	DHT11	
Power supply	3.3-5.5V DC	
Output signal	digital signal via Aosong 1-wire bus	
Sensing element	Polymer humidity resistor	
Operating range	humidity 20-90%RH;	temperature 0~50Celsius
Accuracy	<b>humidity +-5%RH;</b>	temperature +-2Celsius

[https://www.electronicoscaldas.com/datasheet/DHT11\\_Aosong.pdf](https://www.electronicoscaldas.com/datasheet/DHT11_Aosong.pdf)

# Lectura con sensor DHT 11



*El bus utiliza 1 cable para la comunicación entre MCU y DHT11*

# Lectura con sensor DHT 11

Este sensor dispone de una **biblioteca** llamada <DHT.h>. Dentro de ella, se pueden encontrar todas las funciones para el manejo de este sensor.

Esta biblioteca se puede instalar desde el IDE de Arduino de la siguiente forma:

Programa -> Incluir Librería -> Administrar biblioteca

Allí buscamos:

**DHT11**



# Lectura con sensor DHT 11

DHT sensor  
library



**Gestor de Librerías**

Tipo: Todos Tema: Todos

**EduIntro**  
by Arduino LLC  
**Library used for super-fast introduction workshops** Is intended to be used with Arduino UNO / MICRO / MEGA / NANO classic / NANO Every / MKR / WiFi REV2 and a set of basic components (led, button, piezo, LM35, thermistor, LDR, PIR, DHT11, and servo) as a way to introduce people to the basic aspects of Arduino during short workshops.  
[More info](#)

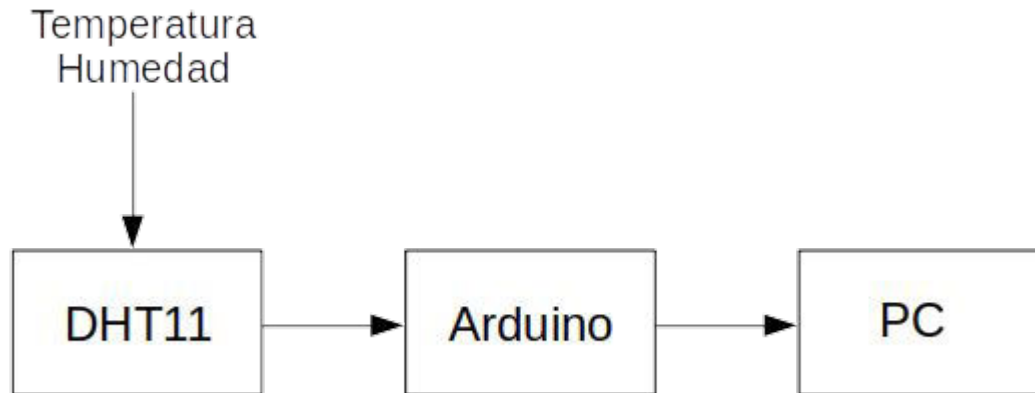
Versión 0.0.13

**DHT sensor library**  
by Adafruit Versión 1.4.2 **INSTALLED**  
**Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors** Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors  
[More info](#)

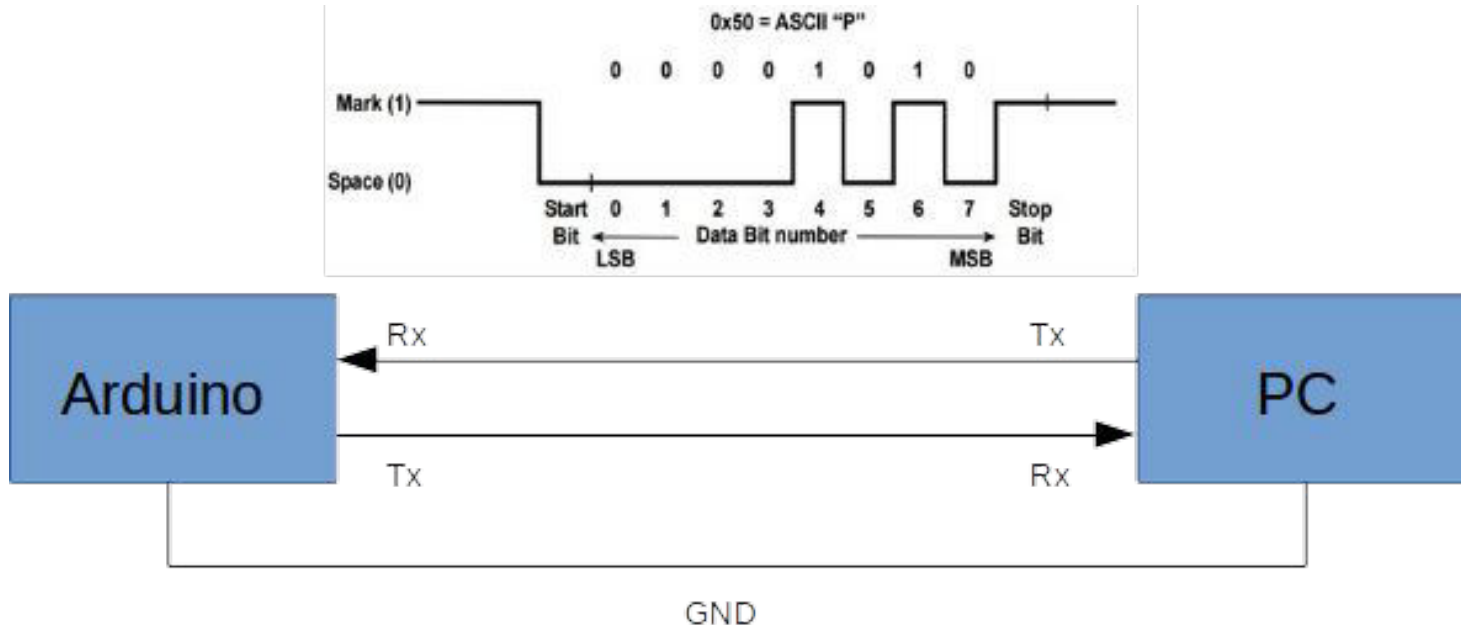
**DHT sensor library for ESPx**  
by beegee\_tokyo  
**Arduino ESP library for DHT11, DHT22, etc Temp & Humidity Sensors** Optimized library to match ESP32 requirements. Last changes: Fix negative temperature problem (credits @heljunky)  
[More info](#)

# Lectura con sensor DHT 11

**Ejemplo:** El ejercicio consistirá en leer desde Arduino, la información suministrada por el sensor de temperatura y mostrarlo por pantalla.



# Lectura con sensor DHT 11



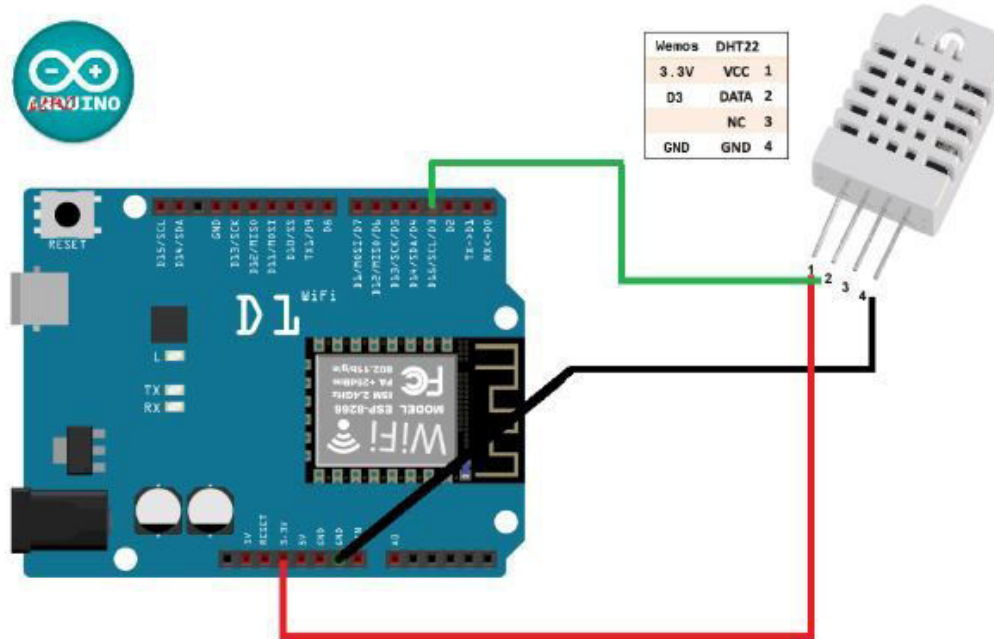
# Lectura con sensor DHT 11

Se propone el siguiente ejercicio:

Leer los datos de **Temperatura y Humedad** provistos por el sensor y **mostrarlos por el monitor serial** de Arduino.

Ver: DHT11\_Wemos.ino

# Lectura con sensor DHT 11



[26]

**Importante:** Agregar una resistencia de 10K ohms entre 5V y D3

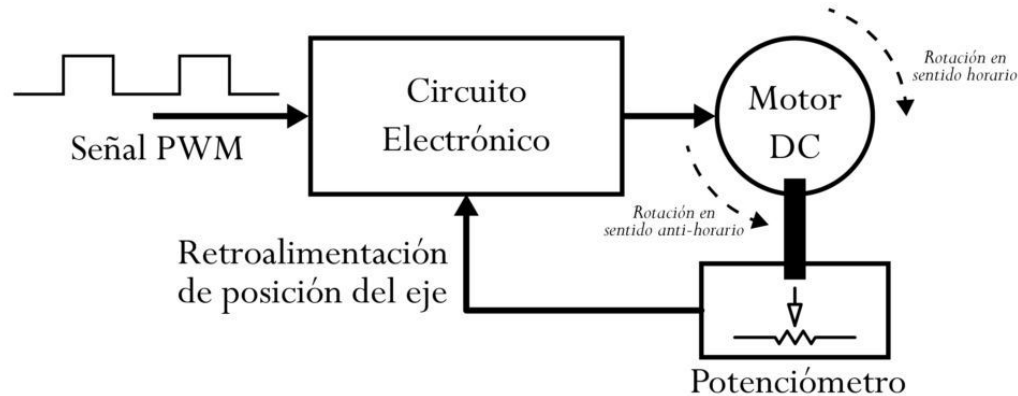
# Introducción a los Sistemas Embebidos

- - Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.
- - Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.
- - Manejo de puertos digitales. Utilización de Relays.
- - Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.
- - **Manejo de servo motor.**
- - Dispositivos I<sup>2</sup>C: Display LCD.

# Introducción a los Sistemas Embebidos

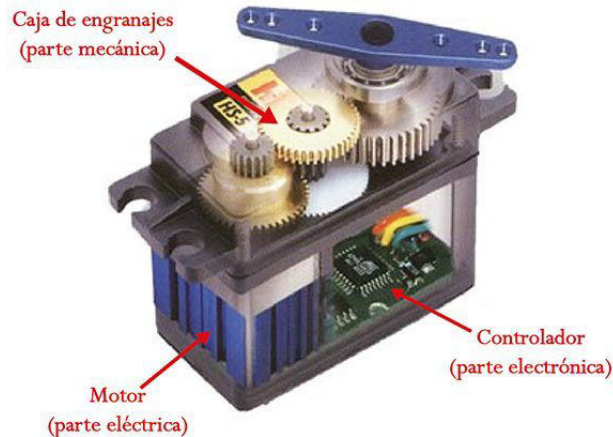
Un **servomotor** es un sistema que consta de un motor para controlar, por ejemplo, la **posición del eje**. Estos sistemas se mueven en una **determinada cantidad de grados** y luego quedan fijos allí.

## DIAGRAMA DE BLOQUE DEL SERVOMOTOR



# Introducción a los Sistemas Embebidos

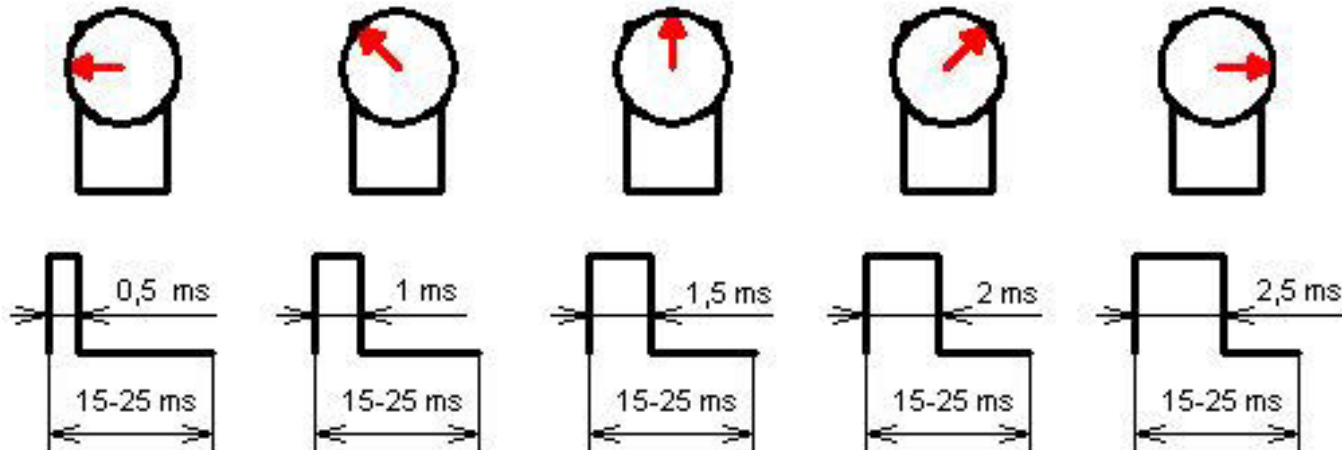
En particular estaremos usando el **Sg90**, el cual tiene una estructura como la que se ve a continuación:





# Introducción a los Sistemas Embebidos

**Funcionamiento:** Para el funcionamiento, se suele usar una señal **PWM** y, dependiendo del ancho del pulso, el motor se detendrá en una posición fija.



# Introducción a los Sistemas Embebidos

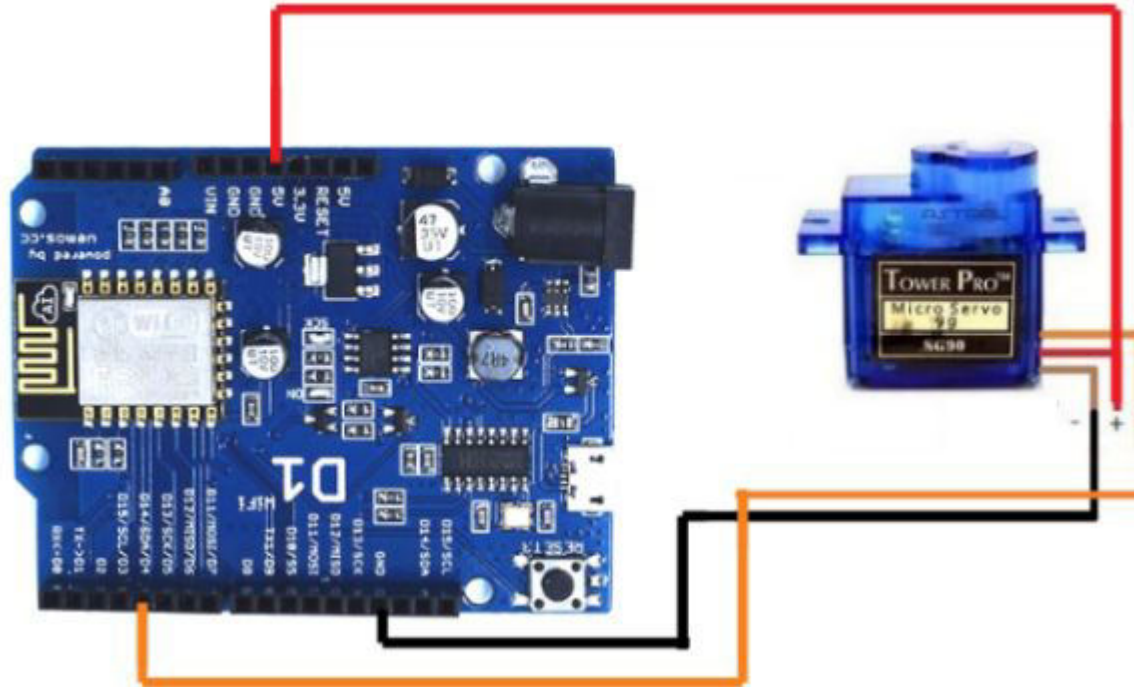
Vamos a usar la biblioteca **“Servo”**. Por esto, conviene instalar dicha biblioteca y luego estudiar el código:

**Sweep\_Wemos.ino:** Código básico para el manejo del servo

**Nota:** Algo importante a destacar es que las funcionalidades del **PWM** se pueden implementar mediante software. Para más información, ver:

[https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf)

# Introducción a los Sistemas Embebidos

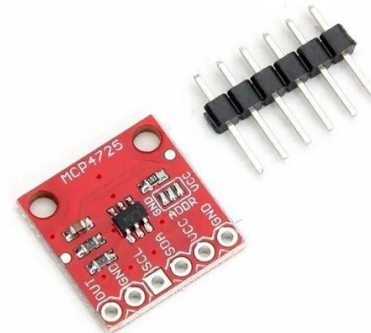
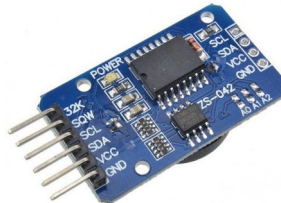
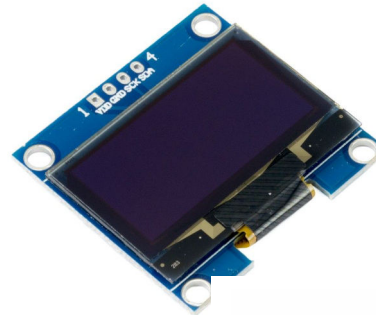
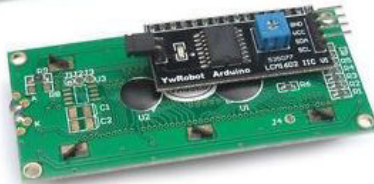


# Introducción a los Sistemas Embebidos

- - Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.
- - Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.
- - Manejo de puertos digitales. Utilización de Relays.
- - Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.
- - Manejo de servo motor.
- - **Dispositivos I<sup>2</sup>C: Display LCD.**

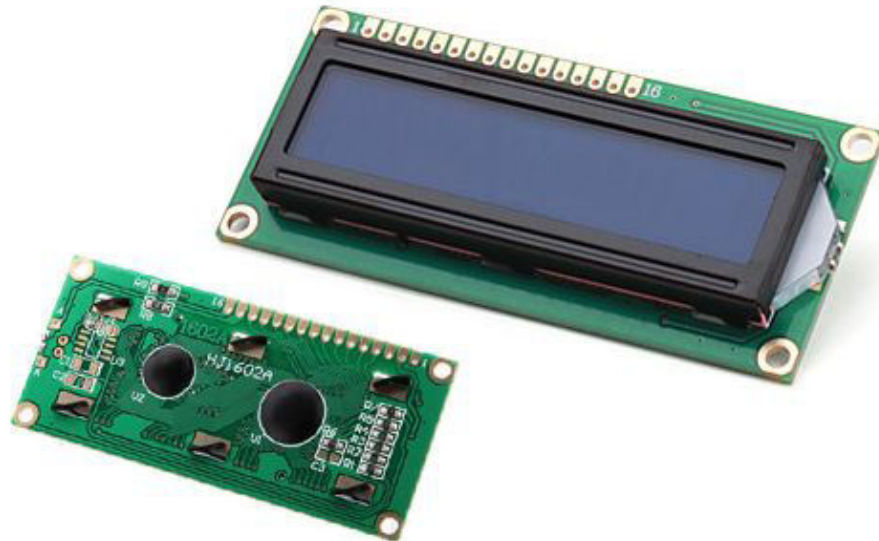
# Dispositivos I<sup>2</sup>C: Display LCD

¿cuáles dispositivos puedo conectar?



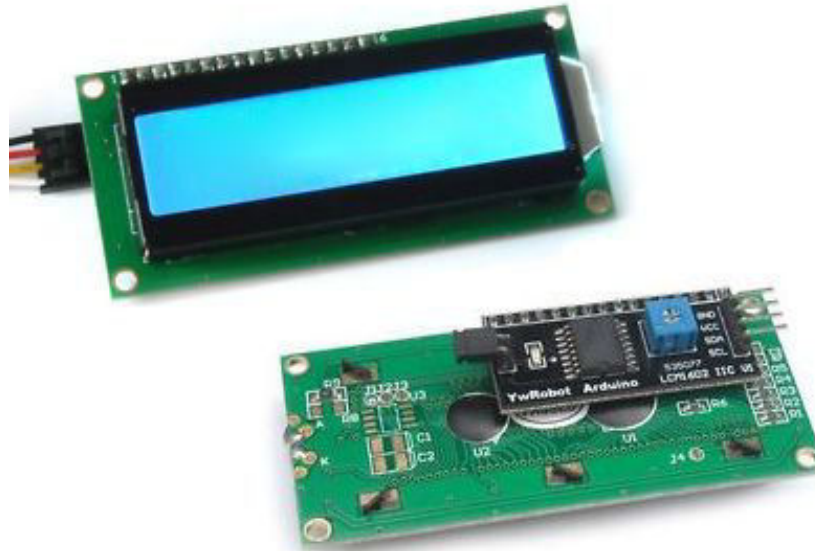
# Dispositivos I<sup>2</sup>C: Display LCD

Los **displays LCD** permiten mostrar la información en forma de matriz, algunos formatos pueden ser en **2x16**, tal como el que se muestra a continuación:



# Dispositivos I<sup>2</sup>C: Display LCD

Los que usaremos aquí contienen un módulo expensor. No todos los módulos LCD vienen con este expensor.



# Dispositivos I<sup>2</sup>C: Display LCD

El circuito integrado **PCF8574** se lo puede ver a continuación:

[5]



GND	—	GND
Vcc	—	5V
SDA	—	SDA (A4)
SCL	—	SCL (A5)

Este circuito integrado es un **expansor de entrada / salida (E / S) de 8 bits para el bus bidireccional de dos líneas (I<sup>2</sup>C)**

[https://www.ti.com/lit/ds/symlink/pcf8574.pdf?ts=1611935644461&ref\\_url=https%253A%252F%252Fwww.google.com%252F](https://www.ti.com/lit/ds/symlink/pcf8574.pdf?ts=1611935644461&ref_url=https%253A%252F%252Fwww.google.com%252F)



# Dispositivos I<sup>2</sup>C: Display LCD

En base a esto y, por lo que vimos sobre protocolo I<sup>2</sup>C, el dispositivo tiene una **dirección** que depende de **A0-A1-A2**. Esto depende de cómo esté configurado el módulo que contiene al **PCF8574**. Puede ser el **0x3F** o el **0x27**. Si esta dirección no está bien configurada en el código desarrollado, no se podrá establecer la comunicación.

Es decir, cuando se adquiere el módulo, muchas veces no hay información de cuál es la dirección del dispositivo por eso, para averiguar esto, se puede usar el siguiente sketch:

[i2c\\_scanner\\_arduino\\_code. ino](#)

**Más información:** <https://playground.arduino.cc/Main/I2cScanner/>

# Dispositivos I<sup>2</sup>C: Display LCD

Una vez determinado el Hardware, ahora podemos ver la **Biblioteca** a utilizar. En particular estaremos usando **LiquidCrystal\_I2C**, la cual la podemos instalar de la siguiente forma:

**Herramientas -> Administrar Bibliotecas**

Luego, se puede buscar: LiquidCrystal\_I2C e instalar la que dice:

**LiquidCrystal I2C by Marco Schwartz**

# Dispositivos I<sup>2</sup>C: Display LCD

Dentro de **LiquidCrystal\_I2C**, podemos encontrar funciones como:

**LiquidCrystal\_I2C(lcd\_Addr, lcd\_cols, lcd\_rows)**: Esta función permite crear un objeto de la clase LiquidCrystal\_I2C, siendo:

**lcd\_Addr**: dirección del dispositivo.

**lcd\_cols**: cantidad de columnas

**lcd\_rows**: cantidad de filas

# Dispositivos I<sup>2</sup>C: Display LCD

**init():** Permite Inicializar el módulo.

**clear():** Permite borrar la pantalla LCD y posicionar el cursor en la esquina superior izquierda.

**setCursor(col, row):** Esta función permite **posicionar el cursor del LCD** según lo **indicado** en **col** y **row**. Con esto se puede determinar la ubicación donde se comenzará a mostrar el texto.

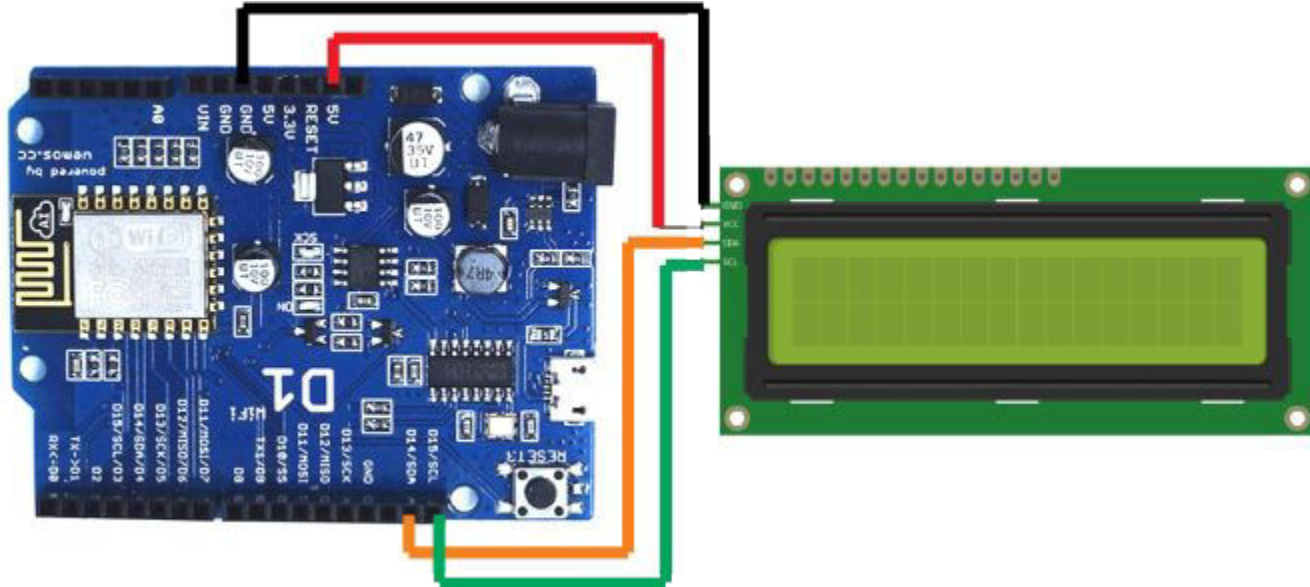
**print():** Esta función permite escribir texto o un mensaje en el LCD. Haciendo la similitud con lo practicado con el puerto serie de Arduino, sería como usar `Serial.print`

# Dispositivos I<sup>2</sup>C: Display LCD

Para llevar a la práctica estos conocimientos, vamos a mostrar la información “Hola mundo” en el display.

**Ver:** [LCD\\_2x16\\_Hola\\_Mundo.ino](#)

# Dispositivos I<sup>2</sup>C: Display LCD



# Planificación

- Introducción a IoT (Internet of Things)
- Introducción a la programación de módulos basados en ESP 8266 mediante Arduino.
- Prácticas con módulos basados en ESP8266.
- **Comunicaciones WiFi con placas basadas en ESP8266.**
- Integración de Conceptos.

# Planificación

- **Primeros pasos en la conexión de WiFi mediante Wemos D1. (modo STA).**
- Transmisión de datos de sensores a ThingsPeak. Estación meteorológica. Extracción de archivos .csv.
- ESP8266 como Web Server. HTML. Automatización de sistemas de potencia para el hogar y la industria.
- Automatización de sistemas para el hogar y la industria mediante protocolo MQTT. Análisis de Brokers.



# Primeros pasos con WiFi

Previamente a conectar la **Wemos**, veamos algunos conceptos:

Todos los **dispositivos** que se **conectan a redes Wi-Fi** se denominan **estaciones (STA, stations)**. La **conexión a Wi-Fi** se realiza mediante un **Punto de Acceso (AP, Access Point)**, que **actúa como un concentrador** para una o más **estaciones**.

El **punto de acceso** generalmente se **integra con un enrutador** (router) para proporcionar acceso desde una **red Wi-Fi a Internet**. Cada **punto de acceso** se reconoce por un **SSID (Service Set Identifier)**, que **indica el nombre de la red**.

<https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>

# Primeros pasos con WiFi



# Primeros pasos con WiFi

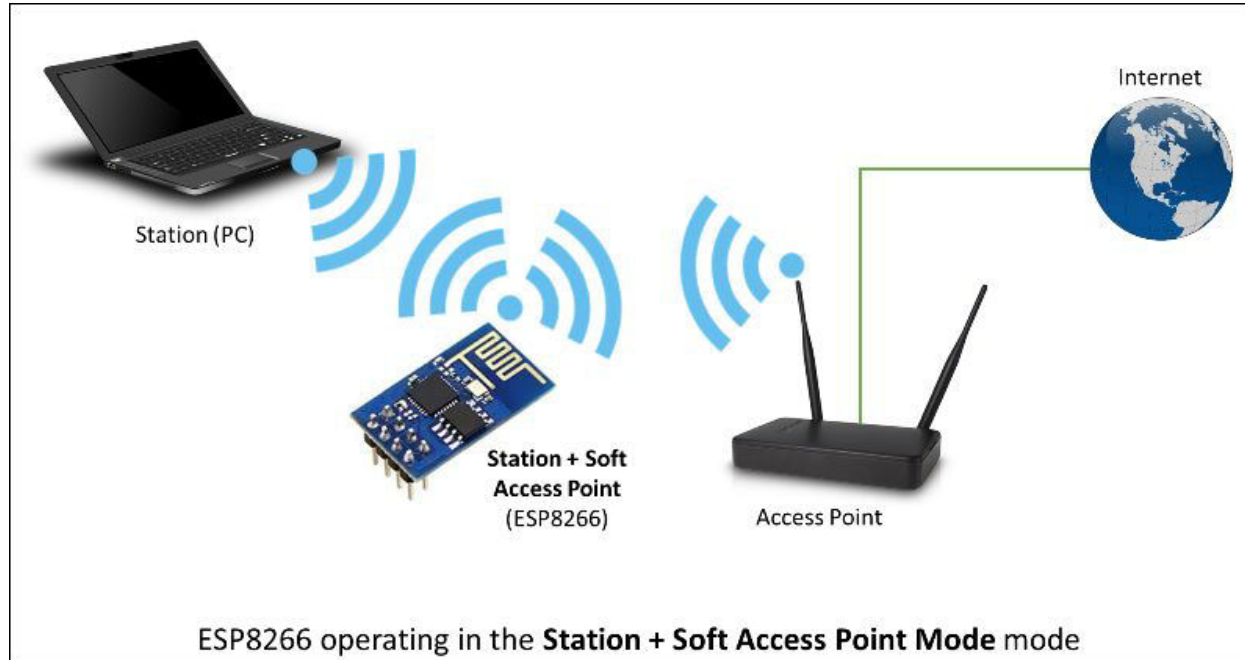
De acuerdo a la imagen anterior, vamos a usar la placa con **ESP8266** en modo **estación** o **STA**.

Sin embargo, es importante tener en cuenta que **los módulos ESP8266**, aparte de funcionar en modo **estación (STA)**, pueden funcionar como **punto de acceso suave** (Soft-AP), **para establecer su propia red Wi-Fi**.

Cuando el módulo **ESP8266** está funcionando como un **punto de acceso suave**, podemos **conectar otras estaciones al módulo ESP**. Es decir, el módulo ESP8266 **también puede funcionar como estación y como modo de punto de acceso suave**. Esto proporciona la posibilidad de construir, por ejemplo, redes de malla (mesh networks).

<https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>

# Primeros pasos con WiFi



# Primeros pasos con WiFi

En nuestro caso para operar mediante WiFi, vamos a usar la biblioteca:

**ESP8266WiFi**

cuya documentación se puede ver en :

<https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi>

**Nota:** Algo importante a tener en cuenta es que, *para que funcione correctamente la biblioteca ESP8266WiFi*, tiene que estar seleccionada la placa a utilizar (Wemos R1 D1)

<https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>

# Primeros pasos con WiFi

Con estos conocimientos, pasaremos a realizar la primera conexión a **WiFi**. Para esto, conectar la placa **Wemos D1** a la PC.

**Ver:** [Wemos\\_WiFi.ino](#)

# Primeros pasos con WiFi

El código utiliza el método **status**, el cual devuelve el **estado de la conexión**.

WL\_CONNECTED: assigned when connected to a WiFi network;

WL\_NO\_SHIELD: assigned when no WiFi shield is present;

WL\_IDLE\_STATUS: it is a temporary status assigned when WiFi.begin() is called and remains active until the number of attempts expires (resulting in WL\_CONNECT\_FAILED) or a connection is established (resulting in WL\_CONNECTED);

# Primeros pasos con WiFi

WL\_NO\_SSID\_AVAIL: assigned when no SSID are available;

WL\_SCAN\_COMPLETED: assigned when the scan networks is completed;

WL\_CONNECT\_FAILED: assigned when the connection fails for all the attempts;

WL\_CONNECTION\_LOST: assigned when the connection is lost;

WL\_DISCONNECTED: assigned when disconnected from a network

<https://www.arduino.cc/en/Reference/WiFiStatus>;



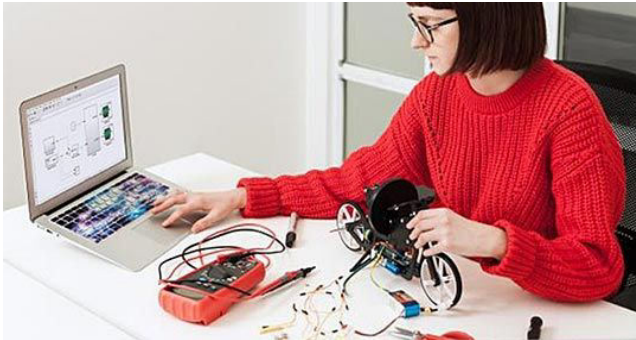
# Planificación

- Primeros pasos en la conexión de WiFi mediante Wemos D1. (modo STA).
- **Transmisión de datos de sensores a ThingsPeak. Estación meteorológica. Extracción de archivos .csv.**
- ESP8266 como Web Server. HTML. Automatización de sistemas de potencia para el hogar y la industria.
- Automatización de sistemas para el hogar y la industria mediante protocolo MQTT. Análisis de Brokers.

# ThingSpeak

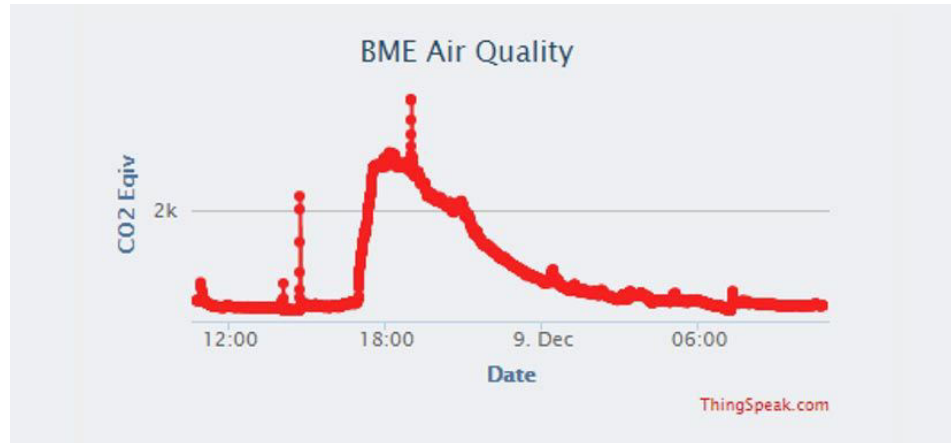
Muchas veces resulta necesario poder enviar datos a la **nube** para **visualizarlos y analizarlos**. Para esto, una posible herramienta resulta ser **ThingSpeak**:

<https://thingspeak.com>



# ThingSpeak

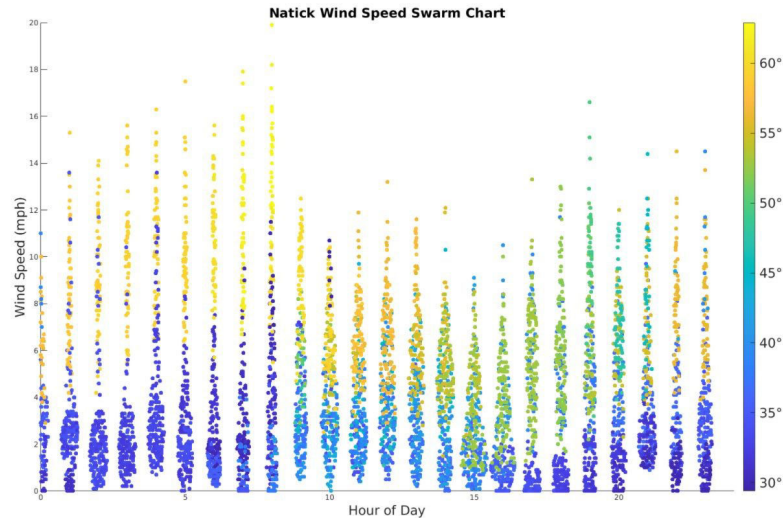
**ThingSpeak** es una *plataforma de análisis de IoT* que permite *agregar, visualizar y analizar flujos de datos en vivo en la nube*. Se puede enviar datos a ThingSpeak desde distintos dispositivos, como por ejemplo, desde la **Wemos** que estaremos usando.



<https://thingspeak.com/>

# ThingSpeak

Además, este servicio provee herramientas de análisis mediante **MATLAB**.



<https://thingspeak.com/>

# ThingSpeak

Para poder enviar los datos a ThingSpeak y observar los datos, resulta necesario crear un **canal**.

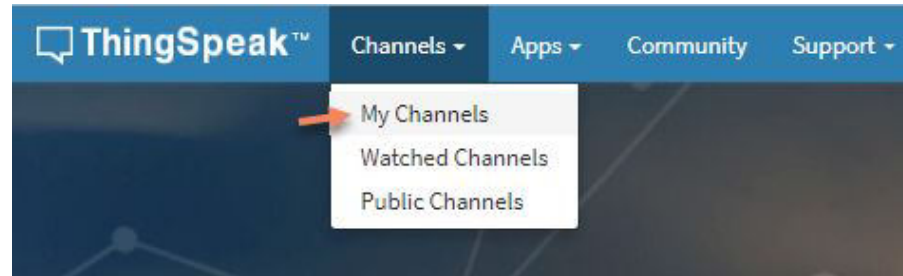
Para esto, previamente hay que hacer una **cuenta** propia en **ThingSpeak**.

Estos pasos se pueden realizar mediante la **documentación** que se encuentra en ThingSpeak.

<https://www.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html>

# ThingSpeak

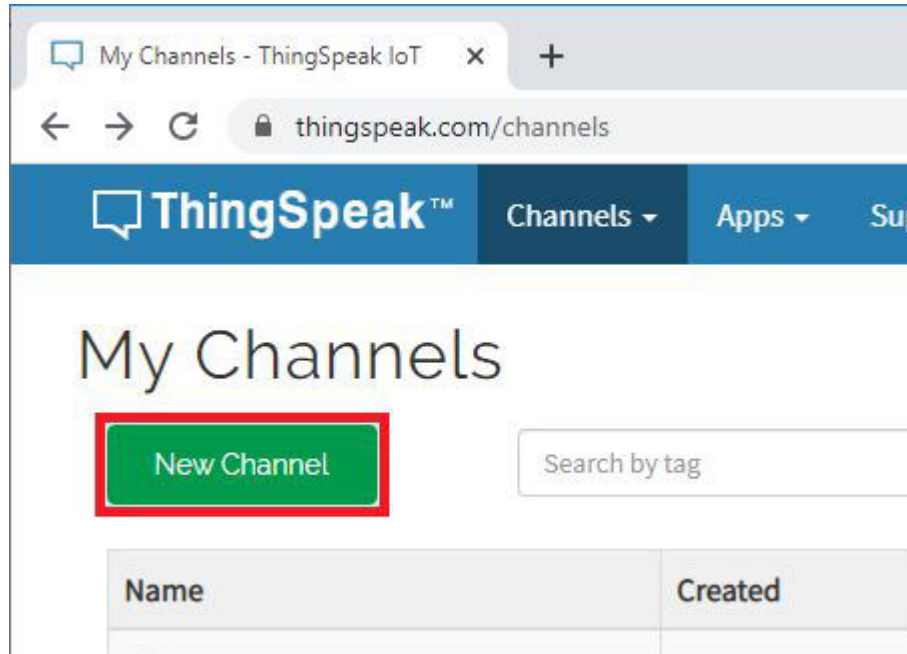
- 1) Iniciar sesión en ThingSpeak™ .
- 2) Click en Channels > MyChannels.



- 3) En la página Channels, hacer clic en **Canal Nuevo** (New Channel).

<https://www.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html>

# ThingSpeak



The screenshot shows a web browser window with the address bar displaying 'thingspeak.com/channels'. The page title is 'My Channels - ThingSpeak IoT'. The navigation bar includes the ThingSpeak logo and menu items for 'Channels', 'Apps', and 'Sup'. The main content area is titled 'My Channels' and features a green 'New Channel' button, a search box labeled 'Search by tag', and a table with columns 'Name' and 'Created'.

Name	Created
-	

# ThingSpeak

4) Marcar las casillas junto al campo 1 (si hay más datos se tildan más campos). Se puede elegir un nombre para el campo (field), por ejemplo, **Temperatura DHT**

ThingSpeak™ Channels Apps Community Support

### New Channel

Name

Description

Field 1

Field 2

Field 3

Field 4

Show Video

YouTube

Vimeo

Video URL

Show Status

Save Channel



# ThingSpeak

5) Hacer clic en **Guardar** canal en la parte inferior de la configuración. Ahora se podrán ver las siguientes pestañas:

**Private View:** esta pestaña muestra información sobre el canal que solo nosotros podemos ver.

**Public View:** si se elige que el canal esté disponible públicamente, se usa esta pestaña para mostrar los campos seleccionados y las visualizaciones del canal.

**Channel Settings:** esta pestaña muestra todas las opciones de canal que estableció en la creación. Puede editar, borrar o eliminar el canal de esta pestaña.

**Sharing:** esta pestaña muestra las opciones para compartir canales. Puede configurar un canal como privado, compartido con todos (público) o compartido con usuarios específicos.

# ThingSpeak

**API Keys:** esta pestaña muestra las claves API del canal. Utilizar las teclas para leer y escribir en su canal.

**Data Import/Export:** esta pestaña permite importar y exportar datos del canal.

<https://www.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html>

# ThingSpeak

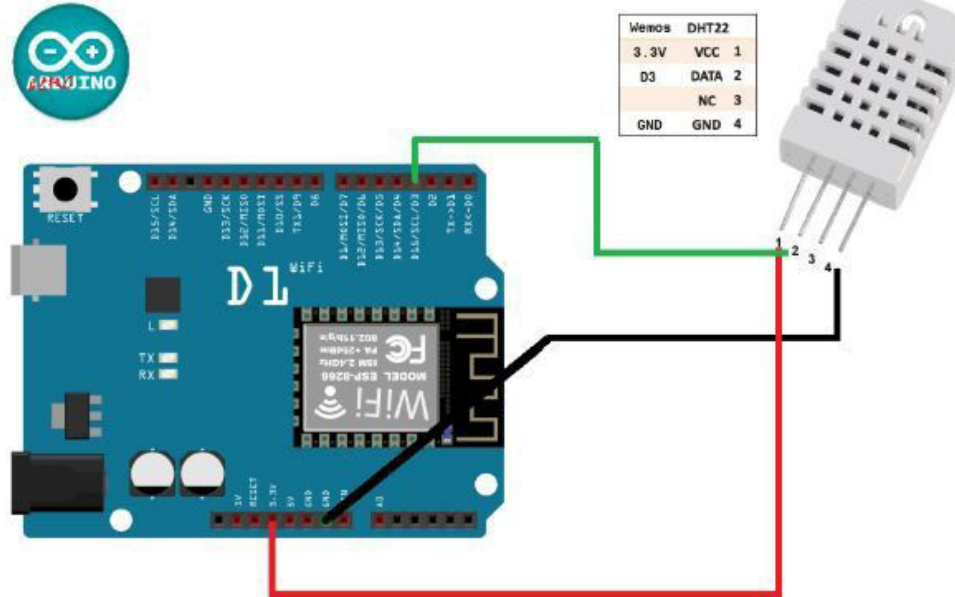
En particular, la práctica que sigue a continuación, consiste en **tomar datos de sensores** como por ejemplo, el **DHT11**, para enviarlos a **ThingSpeak** y poder visualizarlos en tiempo real.

Luego se podrá descargar un archivo .csv para poder analizarlo.

**Ver:** [WeMos\\_ThingSpeak\\_DHT\\_11.ino](#)

# ThingSpeak

## Hardware



[26]

**Importante:** Agregar una resistencia de 10K ohms entre 5V y D3

# ThingSpeak

Una vez determinado el Hardware, ahora podemos ver la **Biblioteca** a utilizar. En particular estaremos usando **ThingSpeak**, la cual proporciona funciones o métodos para publicar los datos tomados por los sensores en un solo campo o en varios.

La podemos instalar de la siguiente forma:

**Herramientas -> Administrar Bibliotecas**

Luego, se puede buscar: **ThingSpeak** e instalar la que dice:

**by MathWorks.**

# ThingSpeak



Gestor de Librerías

Tipo Todos Tema Todos thingspeak

**ThingSpeak**  
by MathWorks Versión 2.0.1 **INSTALLED**  
**ThingSpeak Communication Library for Arduino, ESP8266 & ESP32** ThingSpeak ( <https://www.thingspeak.com> ) is an analytic IoT platform service that allows you to aggregate, visualize and analyze live data streams in the cloud.  
[More info](#)

**ThingSpeak\_asukiaaa**  
by Asuki Kono  
**An API manager for ThingSpeak** It writes field values for ThinkgSpeak.  
[More info](#)

Cerrar

# ThingSpeak

Conviene tener en cuenta dos cosas importantes:

- 1) En el desarrollo del código, son importantes las **API Keys** para el envío de datos desde **Wemos** (ESP8266) a ThingSpeak.

Para esto, abrir la pestaña **Claves de API** (en ThingSpeak) y copiar la clave de **API de escritura** y copiarla en el IDE de Arduino.

# ThingSpeak

2) Para el envío de datos se utilizará el método `writeField()`, el cual se puede estudiar así:

Argumentos: **Número del canal**, **Número del campo**, **Dato a publicar**, **API Key**

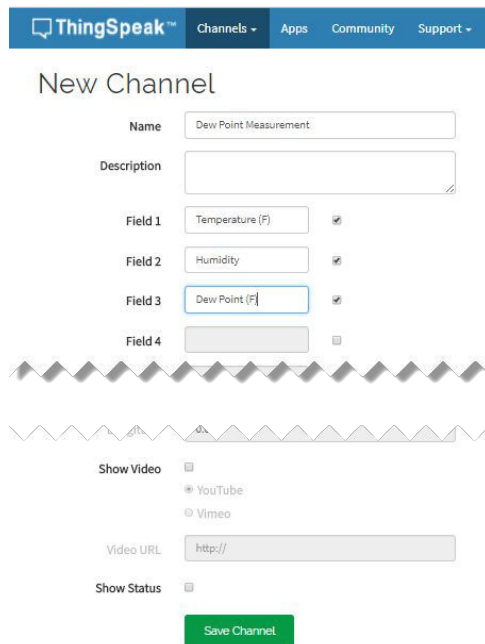
Devuelve: un entero que es el **código de éxito** (200)

<https://github.com/mathworks/thingspeak-arduino/blob/master/src/ThingSpeak.h>



# Múltiples datos a ThingSpeak

Para escribir en **múltiples campos**, previamente hay que setear los Fields (o campos) en **ThingSpeak**



The screenshot shows the 'New Channel' form in the ThingSpeak interface. The form includes the following fields and options:

- Name:** Dew Point Measurement
- Description:** (Empty text area)
- Field 1:** Temperature (F)
- Field 2:** Humidity
- Field 3:** Dew Point (F)
- Field 4:** (Empty)

Below the field configuration, there are two decorative wave patterns. Under the second wave, there are additional options:

- Show Video:** 
  - YouTube
  - Vimeo
- Video URL:** http://
- Show Status:**

A green 'Save Channel' button is located at the bottom of the form.

# ThingSpeak

Luego, en el código de Arduino, el siguiente método **setField ()** asigna los valores correspondientes a cada campo. Por ejemplo:

```
ThingSpeak.setField(1, temperature_C);
```

```
ThingSpeak.setField(2, humedad_rel);
```

```
ThingSpeak.setField(3, presion);
```

# ThingSpeak

Luego, se usa el método `writeFields()` para enviar los datos

```
ThingSpeak.writeField(myChannelNumber, myWriteAPIKey);
```

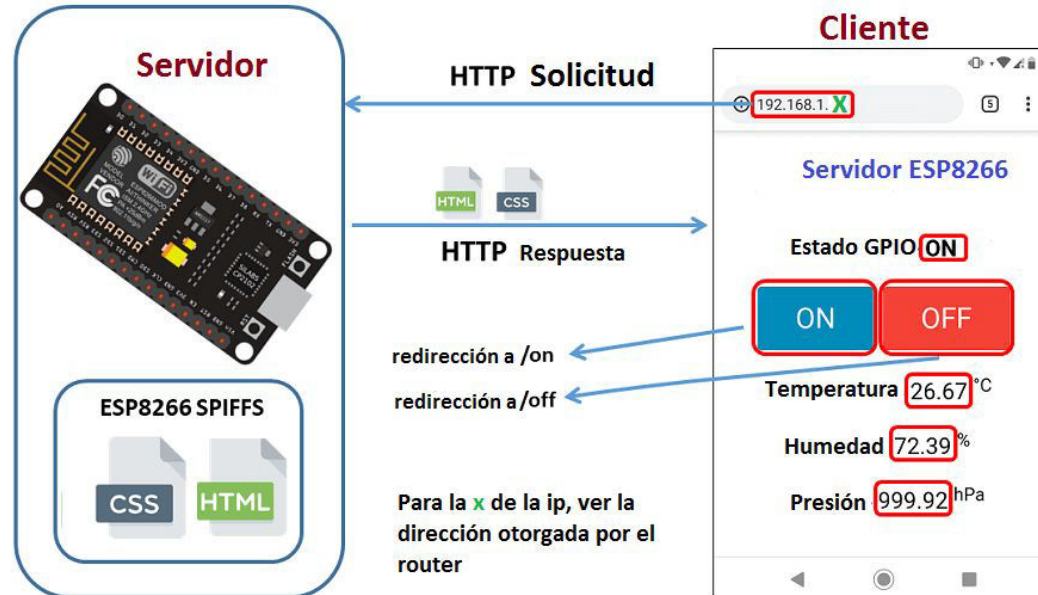
**Ver:** `WeMos_ThingSpeak_DHT_11_Mult.ino`

# Planificación

- Primeros pasos en la conexión de WiFi mediante Wemos D1. (modo STA).
- Transmisión de datos de sensores a ThingsPeak. Estación meteorológica. Extracción de archivos .csv.
- **ESP8266 como Web Server. HTML. Automatización de sistemas de potencia para el hogar y la industria.**
- Automatización de sistemas para el hogar y la industria mediante protocolo MQTT. Análisis de Brokers.

# ESP8266 como Web Server

**Concepto:** Crear un **servidor web** dentro de la placa Wemos para controlar dispositivos a distancia.



# ESP8266 como Web Server

De la imagen anterior se puede ver que la comunicación se hace mediante *una petición http* que se realiza desde un **cliente** (PC, celular, etc), lo cual produce que la **Wemos muestre la información que contiene en su código**.

Del lado del **cliente** se puede ver el texto con botones, información de sensores, etc. Esto resulta así gracias a los archivos **html y css** que se *incluyen en la memoria flash del ESP8266*.

# ESP8266 como Web Server

## Visualizando la página en HTML:

El concepto de **HTML** genera un curso aparte, sin embargo, veremos los conceptos básicos.

**HMTL** o **HyperText Markup Language** ('lenguaje de marcado de hipertexto'), hace referencia al *lenguaje para la elaboración de páginas web*. Define una estructura básica y un código para la *definición de contenido de una página web*, como *texto*, *imágenes*, *videos*, *juegos*, etc.

# ESP8266 como Web Server

## Práctica:

Para llevar estos conceptos a la práctica, se propone realizar un **servidor web** dentro de la placa **Wemos** para que, desde un **cliente**, se puedan **prender o apagar dos LEDs**. **Cada uno de estos LEDs** se podrán comandar, cada uno, mediante **dos botones en pantalla**, tal como se muestra en la siguiente imagen:



# ESP8266 como Web Server

  192.168.43.21

## ESP8266 Web Server

GPIO 5 - State off

ON

Led Wemos - State off

ON

# ESP8266 como Web Server

Para lograr esto, cuando se cargue el código en el ESP8266, el mismo **mostrará la IP en el Serial monitor, la cual tendrá que ser cargada en nuestro browser, para observar el estado de los botones y de los LEDs**

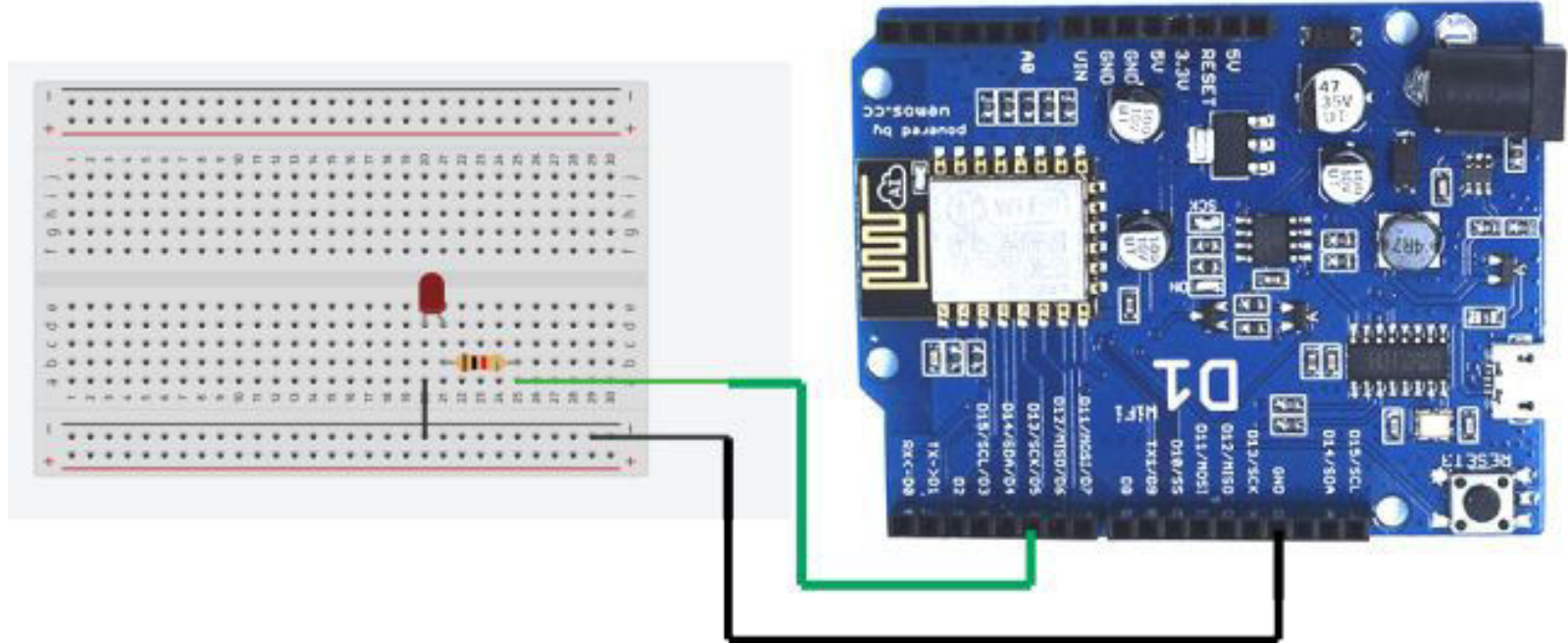
Por ejemplo, si en el Serial monitor vemos la IP: 192.168.27.43, **ésta es la que debe ser cargada en nuestro browser**. Conviene que tanto el cliente como el server, estén en la misma red.

Para esto, cuando se cargue el siguiente código en Arduino

**Ver:** ESP8266\_Web\_Server.ino

# ESP8266 como Web Server

Hardware.



# ESP8266 como Web Server

## Ítems a tener en cuenta en Arduino

- 1) La **página web** se envía al **cliente** mediante la función ***client.println()***. Esta función recibe como argumento lo que se ***desea enviar al cliente***.
- 2) ***El primer texto que siempre debe enviar es***

**<!DOCTYPE html> <html>**

la cual indica que estamos **trabajando con HTML**.

# ESP8266 como Web Server

3) Luego, la siguiente línea hace que la página web responda en cualquier navegador web.

```
client.println ("<head><meta      name=\"viewport\"      content=\"width=device-width,  
initial-scale=1\">");
```

4) Definido esto, a continuación veremos el **estilo de la página**, es decir se utiliza algo de **CSS** para diseñar los botones y la apariencia de la página web. Se elige la **fuentes Helvética**, el **contenido a mostrar como un bloque** y **alineado en el centro**.

```
client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto;  
text-align: center;}");
```

# ESP8266 como Web Server

5) Luego, se define el **estilo para un segundo botón**, con las mismas propiedades del botón definido anteriormente pero con un **color diferente**. Este será el **estilo del botón de apagado**.

```
client.println( ".button2 {background-color: #77878A;}</style></head>");
```

6) En la siguiente línea, se configura el primer **encabezado de la página web**, puede cambiar este texto a lo que se quiera.

```
client.println("<h1>ESP8266 Web Server</h1>");
```

7) Luego, vemos cómo actuar para mostrar el estado del LED en el botón

```
client.println("<p>GPIO 5 - State " + output5State + "</p>");
```

# Planificación

- Primeros pasos en la conexión de WiFi mediante Wemos D1. (modo STA).
- Transmisión de datos de sensores a ThingsPeak. Estación meteorológica. Extracción de archivos .csv.
- ESP8266 como Web Server. HTML. Automatización de sistemas de potencia para el hogar y la industria.
- **Automatización de sistemas para el hogar y la industria mediante protocolo MQTT. Análisis de Brokers.**

# Protocolo MQTT

El proyecto que estaremos armando, se basa en la idea de una **red de área doméstica** utilizando la placa **Wemos D1**.

La idea de este proyecto es que la red de área doméstica se puede controlar desde una **PC a través de Internet**.

Esta red tiene como objetivo **controlar un LED** a través de Internet. Esto resulta ser a los efectos de la práctica pero, en vez de un LED, se puede conectar un **Relay** u otro actuador que resulte de interés.

A su vez, mientras se controla el led, la placa **Wemos** enviará mensajes hacia la PC.



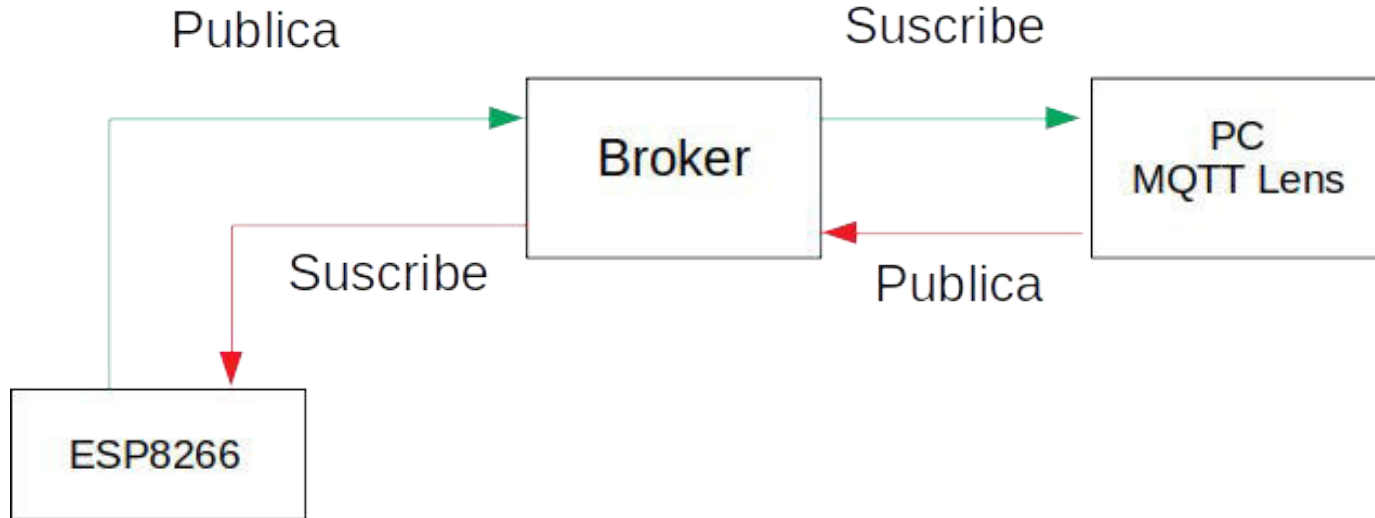
# Protocolo MQTT

En este caso la **PC remota actúa como otro dispositivo IoT**. La PC **se conecta con el dispositivo módulo Wemos a través del broker MQTT**.

En este caso se utiliza el **EMQ X** como **broker** aunque se puede usar cualquier otro.

Para que la PC se pueda conectar, se utilizará un **complemento de Google Chrome: MQTTLens**

# Protocolo MQTT



# Protocolo MQTT

Del lado de la PC, *instalar MQTTLens* en el navegador *Chrome*. *MQTTlens admite el protocolo MQTT y se puede utilizar para la publicación y suscripción de mensajes.*

Al *cliente PC se le debe asignar un ID* para que el *broker de MQTT* pueda identificar cuál cliente está *publicando y suscribiendo el tema* y la *carga útil* (payload).

# Protocolo MQTT

## Instalación MQTT Lens

- 1) En el buscador de Google, agregar: **MQTT Lens**.
- 2) En el link ubicado, al hacer click, se verá la siguiente imagen:

[Inicio](#) > [Aplicaciones](#) > MQTTLens



### MQTTLens

Ofrecido por: MQTTLens

★★★★★ 153 | [Extensiones](#) | 100.000+ usuarios

Añadir a Chrome

- 3) **Añadir a Chrome**

# Protocolo MQTT

¿Cómo funciona todo el proyecto?



# Protocolo MQTT

El **cliente ESP8266** se configura como **publicador** (publisher) para el tópico “**outTopic\_1**” y como **suscriptor** para el tópico “**inTopic\_1**”.

El **cliente de PC se configura** como **suscriptor** para el tópico “**outTopic\_1**” y como **publicador** (publisher) para el tópico “**inTopic\_1**”.

El cliente **ESP8266** inicia la conexión publicando el mensaje “**me conecté!**” al **broker** de MQTT.

# Protocolo MQTT

El **cliente PC** puede **publicar mensajes** como "1" y "0" sobre el tópicó "inTopic\_1".

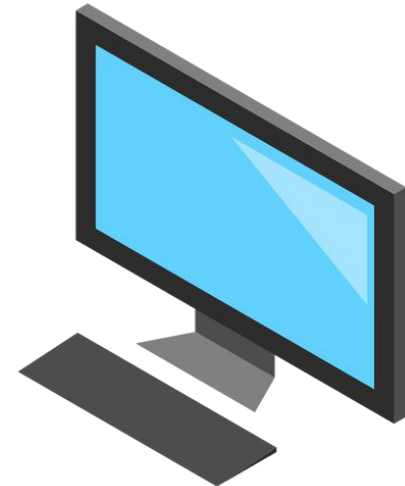
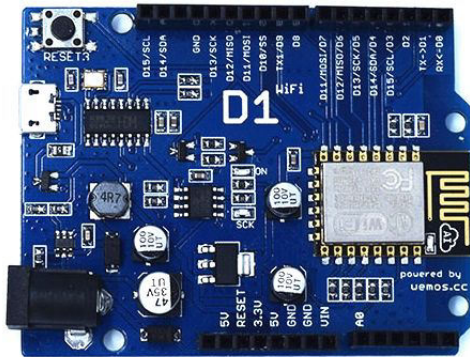
Si el **mensaje "1"** es **publicado** por el **cliente PC**, será **recibido por el cliente ESP8266** y su firmware interpreta el mensaje para **encender el LED**.

Si el **mensaje "0"** es **publicado** por el **cliente PC**, será **recibido por el cliente ESP8266** y su firmware interpreta el mensaje para **apagar el LED**.

**Ver:** MQTT\_LED.ino

# Protocolo MQTT

## Hardware





# Protocolo MQTT

Una vez determinado el Hardware, ahora podemos ver la **Biblioteca** a utilizar. En particular estaremos usando **pubsubclient**, la cual permite que la placa se comporte como un **cliente MQTT** para lograr la **suscripción** y **publicación** de mensajes mediante **MQTT**.

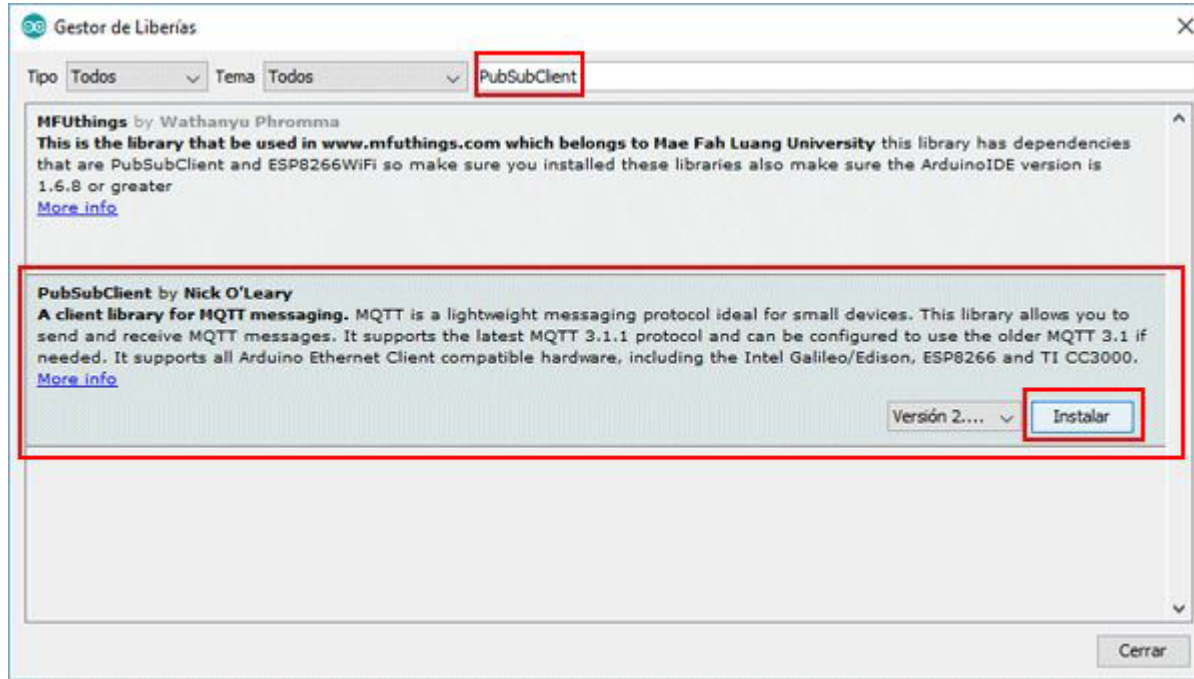
La podemos instalar de la siguiente forma:

**Herramientas -> Administrar Bibliotecas**

Luego, se puede buscar: PubSubClient e instalar la que dice:

**PubSubClient by Nick O'Leary**

# Protocolo MQTT



**CePETel**

Sindicato de los Profesionales  
de las Telecomunicaciones

**SECRETARÍA TÉCNICA**



*Instituto Profesional de  
Estudios e Investigación*

**Mayo 2022**

# **Protocolo MQTT**

Para más información sobre la biblioteca:

<https://pubsubclient.knolleary.net/>

# Protocolo MQTT

Por último, resta la configuración de **MQTT Lens**. Para esto, al iniciar la aplicación se puede ver un cuadro como el siguiente:



- 1) Hacer click en el ícono "+", para agregar una nueva conexión

# Protocolo MQTT

Luego, se abrirá una ventana como la siguiente:

Add a new Connection ✕

---

### Connection Details

<b>Connection name</b>	<input type="text" value="Eclipse MQTT"/>	<b>Connection color scheme</b>	<input type="color" value="#00FF00"/>
<b>Hostname</b>	<input data-bbox="510 743 633 790" type="text" value="tcp://"/> <input type="text" value="e.g. iot.eclipse.org"/>	<b>Port</b>	<input type="text" value="1883"/>
<b>Client ID</b>	<input type="text" value="lens_M3ouummyMG4TMFMFdCftJBdAPcnM"/> <input type="button" value="Generate a random ID"/>		
<b>Session</b>	<input checked="" type="checkbox"/> Clean Session	<b>Automatic Connection</b>	<input checked="" type="checkbox"/> Automatic Connection
		<b>Keep Alive</b>	<input type="text" value="120"/> seconds

# Protocolo MQTT

Luego, se abrirá una ventana como la siguiente:

**Connection Name:** PC\_MQTT

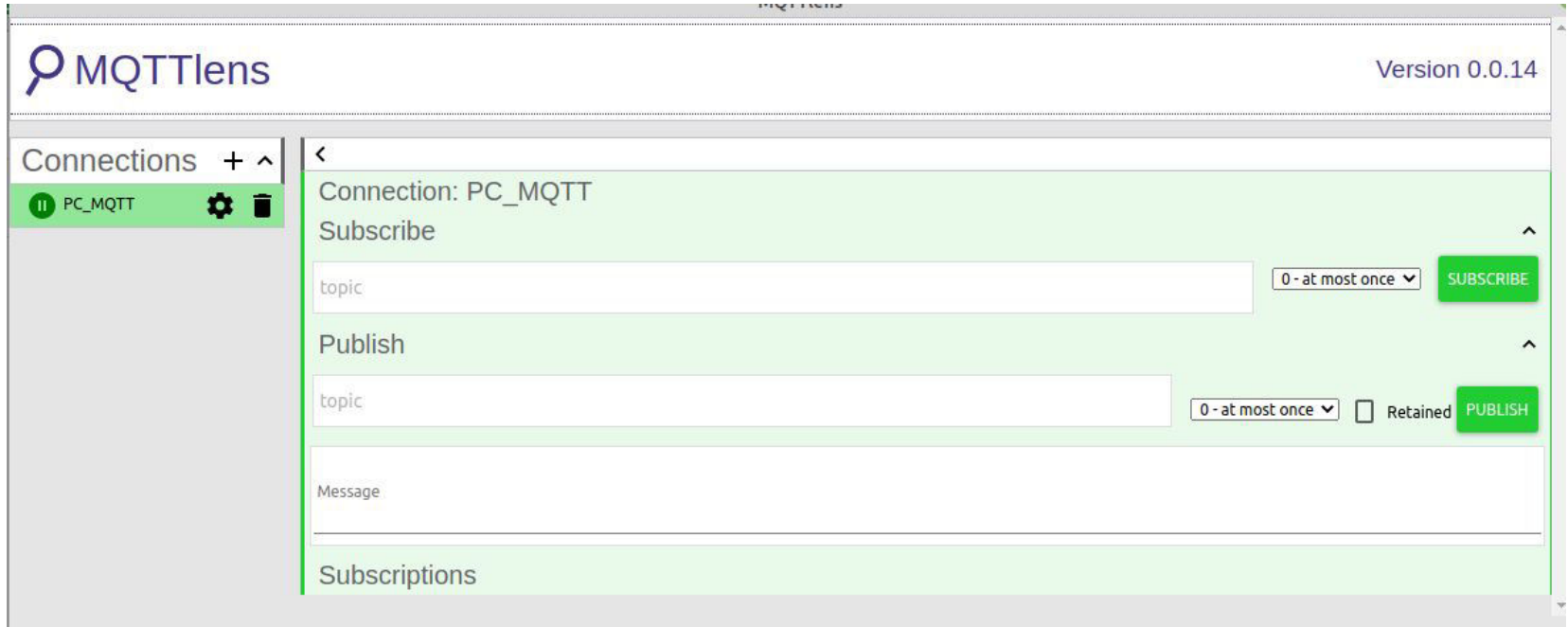
**Hostname: Nombre del Broker:** broker.emqx.io

**Port:** 1883

Por último, darle click a **CREATE CONNECTION**

# Protocolo MQTT

Luego, se abrirá una ventana como la siguiente. Aquí podemos **Publicar** y **Suscribir**



# Protocolo MQTT

## Suscripción:

Completar de la siguiente forma:

**Subscribe:** outTopic\_1

Darle click al botón **Subscribe**. Luego de esto, se verá más abajo cómo la **aplicación** (MQTT LENS) se suscribe al cliente y muestra los mensajes publicados



# Protocolo MQTT

## Publicación:

Completar de la siguiente forma:

**Publish:** inTopic\_1

Luego, debajo, escribir el mensaje deseado. En este caso será **1** o, de otra forma, **0**, y darle click al botón **Publish**.

Luego de esto, se verá más abajo cómo el **LED** de la placa **Wemos**, se prende (o apaga).

# Protocolo MQTT

En base a la práctica realizada, se puede estudiar y comprobar el funcionamiento de otros **brokers**. A continuación se comenta una posible lista de brokers:

[https://github.com/mqtt/mqtt.org/wiki/public\\_brokers](https://github.com/mqtt/mqtt.org/wiki/public_brokers)

# Planificación

- Introducción a IoT (Internet of Things)
- Introducción a la programación de módulos basados en ESP 8266 mediante Arduino.
- Prácticas con módulos basados en ESP8266.
- Comunicaciones WiFi con placas basadas en ESP8266.
- **Integración de Conceptos.**

# Integración de Conceptos.

- 1) a) Diseñar un sistema, utilizando **MQTT**, que permita suscribirse a un tópico y **activar un Relay**. A la vez, se debe mostrar la información de activación del relay mediante un display LCD 2x16.
- b) Agregar un sensor de temperatura al sistema y publicar los datos de temperatura.
- c) Si los datos del sensor de temperatura superan un determinado umbral (por ejemplo, 20 °C), activar o desactivar el Relay (Opcional)

# Integración de Conceptos.

2) Diseñar un sistema, mediante **MQTT**, que permita suscribirse a un tópico y **activar un Servomotor**. El ángulo del servo puede variar de acuerdo a le mensaje que envíe el publicador.

# Integración de Conceptos.

3) Diseñar un sistema que envíe datos de temperatura a **ThingSpeak** y, a la vez, muestre los datos mediante un display LCD de 2x16.

El envío de datos debe ser múltiple, teniendo en cuenta datos de distintos sensores. Por ejemplo, LM 35, DHT, etc.

# Integración de Conceptos.

4) Diseñar un sistema, mediante un servidor, para activar un servomotor y un relay. Tener en cuenta el diseño del servidor web para facilitar el uso de estos dispositivos.

# Referencias

- [1] <https://ros.2021discountoutlets.ru/category?name=wemos%20d1%20mini%20pinout%20led>
- [2] Designing Embedded Systems and the Internet of Things with ARM. Perry Xiao
- [3] <http://www.steves-internet-guide.com/mqtt-works/>
- [4] <https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/>
- [5] <https://aprendiendoarduino.wordpress.com/2018/10/17/pantalla-lcd-i2c-en-arduino/>
- [6] <https://www.luisllamas.es/arduino-lcd-hitachi-hd44780/>
- [7] <http://panamahitek.com/que-es-y-como-funciona-un-servomotor/>
- [8] <https://forum.arduino.cc/t/controling-5v-relay-with-wemos-d1-r1/671433>
- [9] <https://programarfacil.com/blog/arduino-blog/rele-con-arduino-lampara/>
- [10] <https://www.tecnoajudes.com/es/control-por-voz-de-dispositivos-con-esp8266/>



# Referencias

- [11] <http://diymakers.es/usando-el-puerto-serie-del-arduino/>
- [12] <http://programaciondeavr.blogspot.com/2018/12/usart-o-uart.html>
- [13] <https://www.murkyrobot.com/guias/comunicacion/i2c>
- [14] <https://aprendiendoarduino.wordpress.com/2017/07/09/i2c/>
- [15] <https://www.engineersgarage.com/controlling-an-led-light-with-pc-using-esp8266-based-han-and-hivemq-broker-iot-part-20/>
- [16] <https://biql.es/tooling-tuesday-wemos-d1-mini-micropython/>
- [17] <https://aprendiendoarduino.wordpress.com/tag/tcpip/>
- [18] <https://aprendiendoarduino.wordpress.com/tag/tcpip/>
- [19] <http://esp32.net/>
- [20] <https://www.studiopieters.nl/esp32-pinout/>

# Referencias

- [21] <https://www.tecnologia-informatica.es/Servidor-web-con-nodemcu-esp8266/>
- [22] <https://randomnerdtutorials.com/esp8266-web-server/>
- [23] <https://www.monarcaelectronica.com.ar/productos/wemos-d1-wifi-uno-shield-esp8266-arduino-uno-mona/>
- [24] <https://www.profetolocka.com.ar/2020/05/15/utilizando-la-placa-wemos-d1/>
- [25] <https://www.electronicclinic.com/wemos-d1-esp8266-arduino-compatible-its-specs-and-how-to-use-it/>
- [26] <http://arubia45.blogspot.com/2019/04/arduino-wemos-sensor-mqtt-home-assistant.html>
- [27] <https://elosciloscopio.com/tutorial-pantalla-lcd-arduino-esp8266-esp32/>
- [28] <https://uelectronics.com/producto/wemos-d1-wifi-esp8266-esp-12f-compatible-con-arduino/>
- [29] <https://aprendiendoarduino.wordpress.com/2021/02/20/mqtt-y-esp8266/>
- [30] <https://randomnerdtutorials.com/esp8266-nodemcu-thingspeak-publish-arduino/>

# Planificación

- Introducción a IoT (Internet of Things)
- Introducción a la programación de módulos basados en ESP 8266 mediante Arduino.
- **Prácticas con módulos basados en ESP8266.**
- Comunicaciones WiFi con placas basadas en ESP8266.
- Integración de Conceptos.

# Introducción a los Sistemas Embebidos

- - **Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.**
- - Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.
- - Manejo de puertos digitales. Utilización de Relays.
- - Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.
- - Manejo de servo motor.
- - Dispositivos I<sup>2</sup>C: Display LCD.

# UART

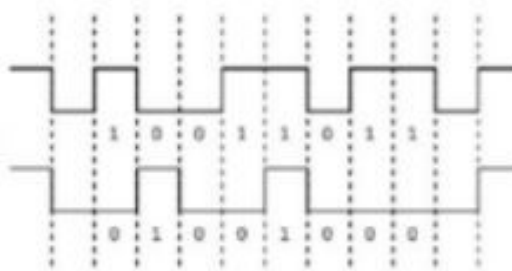
Una de las ventajas que tiene el **ESP8266** es la de poder comunicarse con el mundo mediante mediante puertos de entradas y salidas.

Uno de los puertos que permite intercambiar información mediante esta comunicación serie, es el **puerto Serie**. Cuando hablamos de una comunicación serie nos referimos a **transmitir** y **recibir** datos de **forma secuencial**.

Para realizar esta **comunicación**, el **ESP8266** dispone de un módulo llamado **UART** (Universal Asynchronous Receiver-Transmitter), para el cual se necesitan dos líneas: **Tx** y **Rx**.

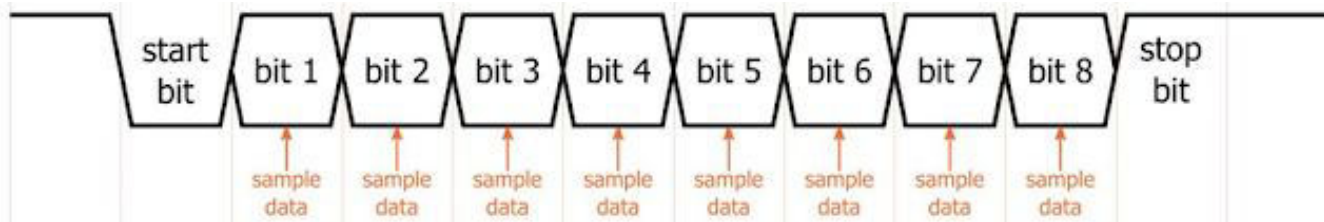
# UART

En nuestro caso vamos a utilizar con frecuencia a este puerto para comunicarnos con la PC y mostrar datos, por ejemplo, los que provienen de sensores



# UART

Esta comunicación se hace mediante una trama como la que se ve a continuación:



[12]

Siendo:

**bit Start**: inicia la comunicación, **bit 1-8**: datos, **bit stop**: finaliza la comunicación

# UART

## Configuración mediante el IDE:

Para la comunicación, hay que **inicializar el módulo**, el cual lo hacemos mediante:

```
Serial.begin();
```

En el argumento de esta función podemos setear dos parámetros:

**speed**: velocidad en bits por segundos (baudios). Generalmente se usa 9600 bits/s, aunque se pueden usar otras velocidades como 38400, 19200, etc.

**config**: nos permite setear la paridad, datos, bit de stop



# UART

## Configuración mediante el IDE - Envío de datos

Luego para enviar datos, se puede usar:

**Serial.print()**

Esta función permite imprimir datos en el puerto serie como texto ASCII.

En el argumento de la función se coloca el dato que se quiere enviar.

Devuelve la cantidad de bytes escritos

<https://www.arduino.cc/reference/en/language/functions/communication/serial/print/>

# UART

## Configuración mediante el IDE - Recepción de datos

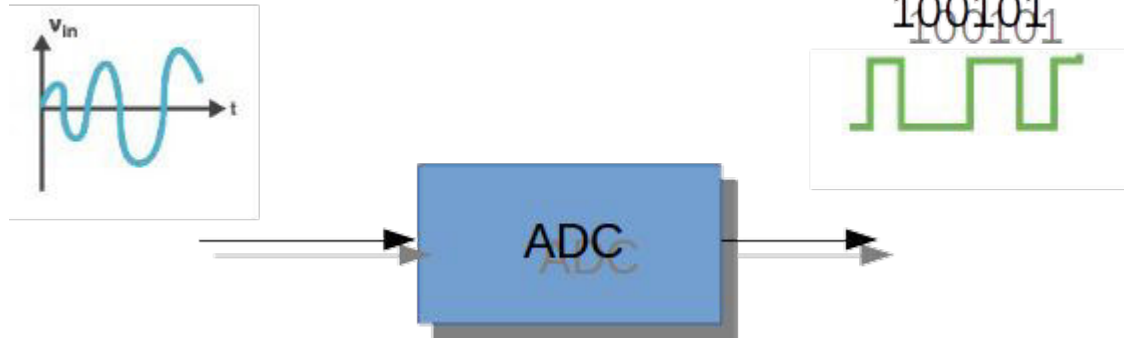
Para leer datos se puede usar `Serial.read()`, la cual es una función que **devuelve el primer byte de la comunicación.**

Es importante, antes de leer el dato, verificar que hay datos disponibles. Para eso se puede usar la función: `Serial.available()`, la cual obtiene la cantidad de bytes disponibles para leer.

# Convertor ADC

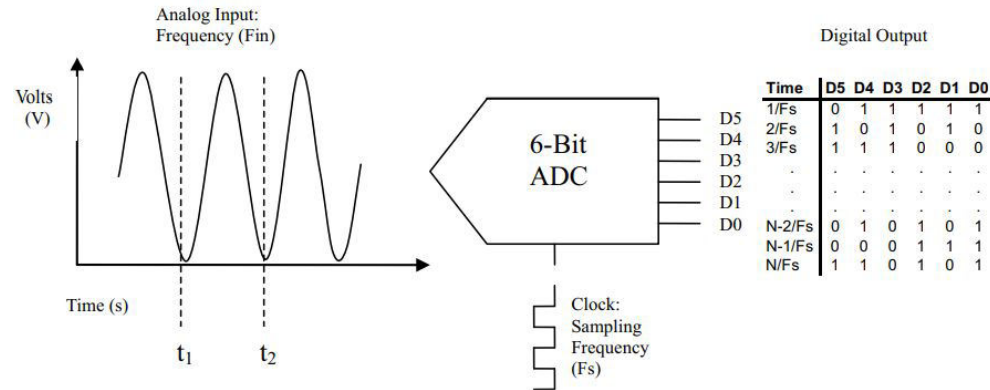
Convierte una **señal analógica** como por ejemplo, la señal de voz, en una **señal digital** compuesta por “unos” y “ceros”.

Cada **valor analógico** se corresponde con una **secuencia digital**.



# Conversor ADC

En la Figura se observa una **señal analógica** aplicada en la entrada de un **ADC**, la cual luego, **se convierte en palabras digitales mediante la frecuencia de muestreo ( $F_s$ )** aplicada al reloj ADC. Esta representación resulta ser en función del tiempo.



$F_{in}$ : Analog Input Frequency =  $1/(t_2 - t_1)$   
 $F_s$ : Clock Frequency  
 N: number of digital samples captured  
 n: number of output bits; in this 6-bit ADC example n = 6

# Conversor ADC

Dentro del ESP8266 se encuentra un **conversor ADC de 10 bits**. Dado que este **SOC** trabaja con **tensiones entre 0 y 1,1V**, cada uno de los **valores analógicos** que ingrese al conversor, se corresponderá con un **valor digital**, que se corresponderá a un valor en decimal.

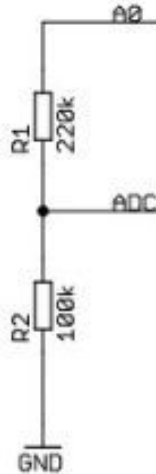
## Ejemplo:

- 1) Si ingresan 1,1V al conversor ADC, este valor se corresponderá con el valor binario 1111111111, que pasandolo a decimal, corresponde al valor 1023.
- 2) Si ingresan 0,55V al conversor ADC, este valor se corresponderá con el valor binario 0111111111, que pasandolo a decimal, corresponde al valor 511.

Arduino podrá mostrar simplemente el valor en decimal, para hacer más simple el análisis

# Convertor ADC

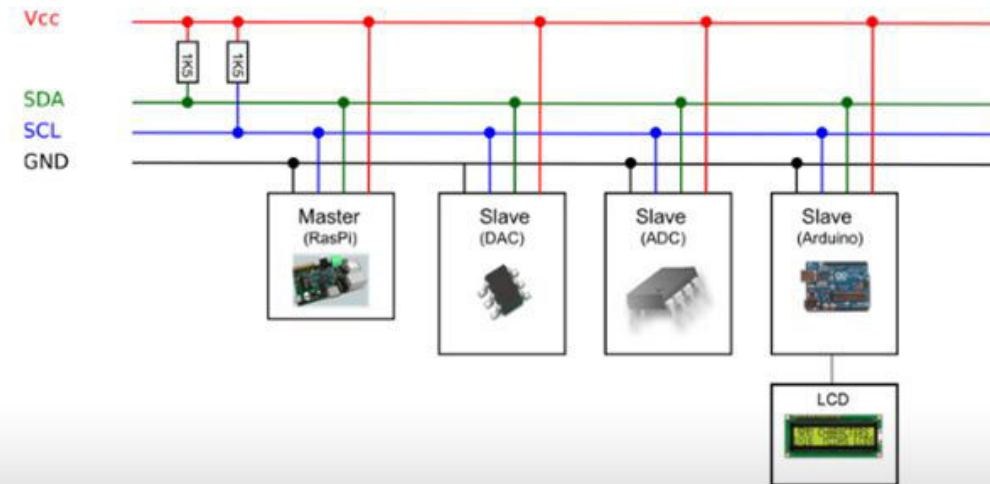
Es importante tener en cuenta que *el convertor ADC del ESP8266 soporta hasta 1,1V*. Sin embargo, en la placa, se encuentra un divisor resistivo de tensión el cual admite una *tensión de 3,3V a la entrada*.



Con esto, hay que tener en cuenta entonces que *la máxima tensión analógica de entrada que se le puede suministrar a la Wemos D1 es de 3,3V*.

# Interfaz I<sup>2</sup>C

El bus I<sup>2</sup>C es un protocolo de comunicación que utiliza dos “hilos” o cables para la comunicación. Esto es, utiliza una línea para los **datos (SDA)** y otra para **sincronización (SCL)**.



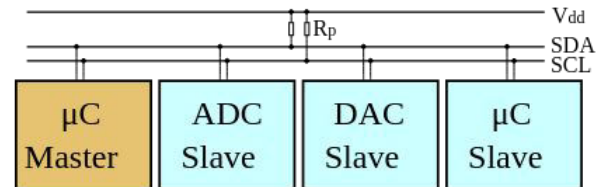
# Interfaz I<sup>2</sup>C

De la imagen anterior se ve que en el mismo bus, puede haber **varios dispositivos conectados**. Si se envía un mensaje a través del bus, éste tiene que ser dirigido a uno de los dispositivos.

Generalmente, quien **envía el dato** se llama **“Master”** y el/los que **reciben el dato** se llaman **“Slave”**.

El dispositivo **“Master”** es el que inicia la el envío de datos (conversación) y por ende quien envía las **señales de clock** (SCL).

Cada **esclavo** tiene una **dirección de 7 bits** lo cual le permite al **Master**, saber a quién le va a enviar el dato/mensaje.

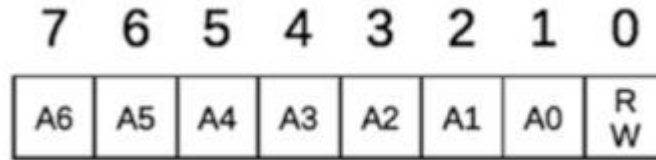




# Interfaz I<sup>2</sup>C

Siguiendo la idea anterior de la dirección, veamos cómo se direcciona

[5]



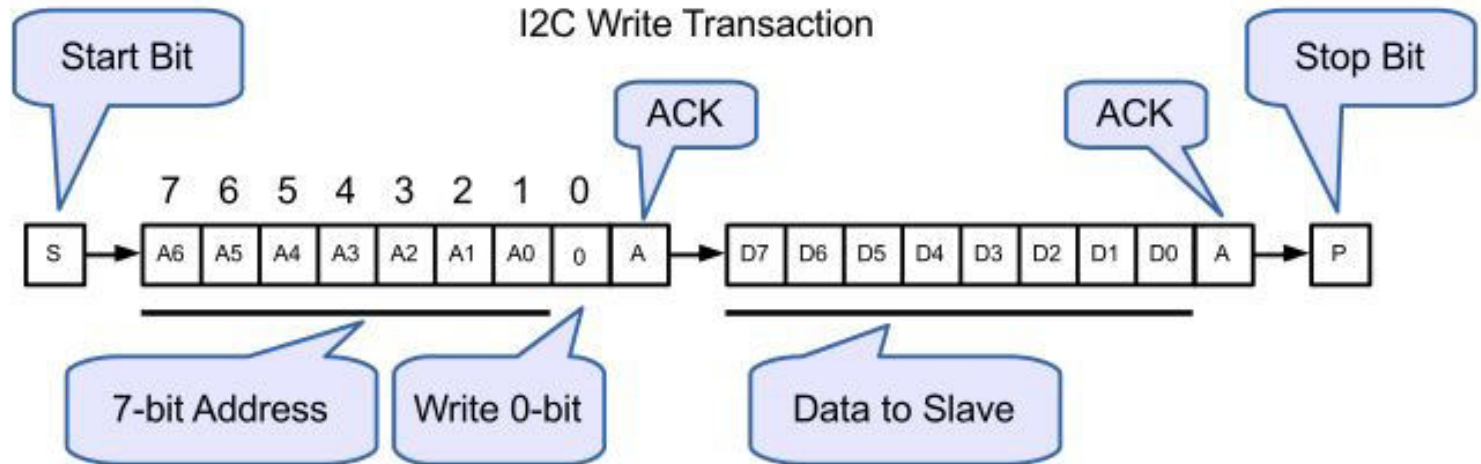
De la imagen anterior se puede ver que se usan los **7 bits más significativos para la dirección del esclavo** y el **bit "0"** se usa para **leer/escribir (R/W)**.

Si **R/W = 1**, indica operación de **lectura** desde el esclavo.

Si **R/W = 0**, indica operación de **escritura** sobre el esclavo.

# Interfaz I<sup>2</sup>C

En base a la idea anterior, se pueden definir los mensajes de *Escritura/Lectura*



# Interfaz I<sup>2</sup>C

De la imagen anterior se ve que, para el **proceso de escritura**, se necesitan los siguientes bits y bytes:

**Bit de Start:** Indica el comienzo de la comunicación.

**7-bits Address:** Indica la dirección del dispositivo a escribir (7-1).

**Write bit:** Tiene que estar en 0 para escribir (Write).

**ACK:** Bit de Acknowledge, bit de chequeo.

**Data to Slave:** Byte de dato a enviar.

**ACK:** Bit de Acknowledge, el dispositivo esclavo reconoce la solicitud

**Bit de Stop:** Fin de la comunicación.

# Interfaz I<sup>2</sup>C

En el protocolo de **lectura** se da una secuencia similar a la de **escritura**, simplemente que hay que **modificar** el **bit R/W** para que esté en “**1**” ya que el mismo indicará la **lectura** desde el dispositivo esclavo.

En arduino utilizaremos la biblioteca “**Wire.h**”

Las grandes **ventajas** de este protocolo es que necesita **pocos líneas** para la comunicación y **cada dispositivo tiene su propia dirección**.

**Mas información:** <https://www.i2c-bus.org>

# Referencias para funciones

Es importante saber que la **Referencia** para el manejo de puertos se puede encontrar en:

<https://arduino-esp8266.readthedocs.io/en/latest/reference.html#>

**Nota:** Algo importante a destacar es que la implementación de I<sup>2</sup>C en el ESP8266 se realiza mediante software. Para más información, ver:

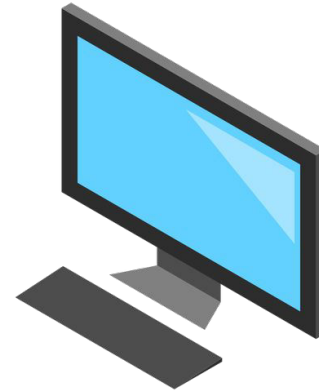
[https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf)

# Introducción a los Sistemas Embebidos

- - Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.
- - **Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.**
- - Manejo de puertos digitales. Utilización de Relays.
- - Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.
- - Manejo de servo motor.
- - Dispositivos I<sup>2</sup>C: Display LCD.

# Puerto Serie

En esta práctica se comunicarán la Wemos con la PC. De esta forma se podrán enviar y recibir datos.



# Puerto Serie

Para ver la aplicación, vamos a usar dos sketch:

- 1) UART\_envio.ino
- 2) UART\_recep.ino

Para esto, se debe conectar el cable desde Arduino hacia la PC

**Nota:** Algo importante a destacar es que el **ESP8266** tiene dos interfaces: **UART0** y **UART1**. Las transferencias de datos hacia / desde interfaces **UART** se pueden implementar a través de **hardware**. Para más información, ver:

[https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf)



# Introducción a los Sistemas Embebidos

- - Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.
- - Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.
- - **Manejo de puertos digitales. Utilización de Relays.**
- - Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.
- - Manejo de servo motor.
- - Dispositivos I<sup>2</sup>C: Display LCD.

# Relays

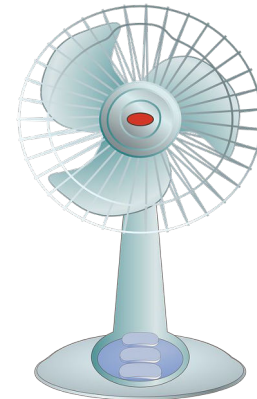
*¿Qué es un Relay?*



# Relays

Muchas veces cuando trabajamos con **Sistemas Embebidos**, resulta importante poder manejar **equipos eléctricos** como por ejemplo, **luminarias, ventiladores, persianas, etc.**

El ejemplo más claro se puede dar cuando queremos **encender luces de alguna habitación**. Aquí, hay que tener en cuenta que **las lámparas se conectan a los 220V**, con lo cual, con lo que conocemos hasta ahora, no se podría realizar.



# Relays

Para llevar a cabo esta tarea, podemos usar un **interruptor** que se **accione mediante los pulsos enviados desde la placa** (Arduino UNO, Wemos, PIC, etc). Al enviarle un **HIGH** ("1"), a este **interruptor se cierra** y, si enviamos un **LOW** ("0"), el **interruptor se abre**.

Lo interesante es **conectar este interruptor en serie con la lámpara** para que la misma se pueda encender. Sin olvidar de suministrar los 220V.

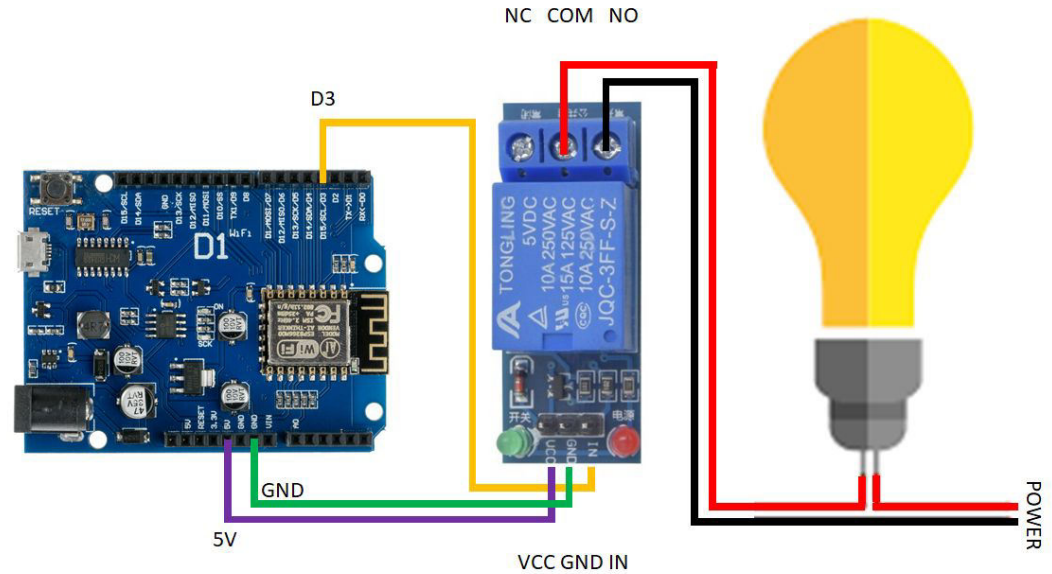
**Atención:** Puede haber riesgo eléctrico, manejar con precaución.



# Relays

El interruptor mencionado

es un **Relay**.



# Relays

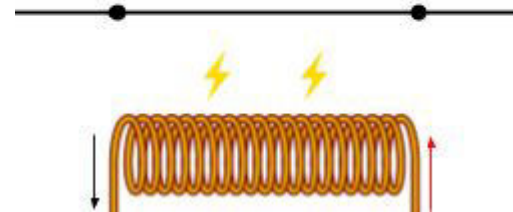
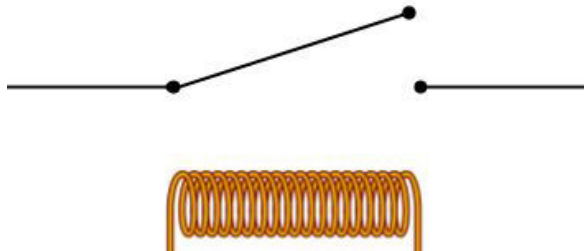
En la entrada del **Relay**, existe un **bobinado** tal que, al **energizarlo**, **acciona al interruptor**.



Estos módulos vienen con un **transistor** para activar a la bobina del **Relay**.

# Relays

Existen diversos modelos, algunos de ellos son los **Normalmente Abiertos (NO)**, los cuales hacen referencia a que, **cuando le enviamos un pulso, cierran sus contactos. De otra forma se mantienen abiertos.**



# Relays

También existen los **Normalmente Cerrados (NC)**, los cuales hacen referencia a que, **cuando le enviamos un pulso, abren sus contactos. De otra forma se mantienen cerrados.**





# Relays

A la hora de elegir el **Relay**, hay que tener en cuenta algunas **especificaciones técnicas**:

**Tensión de activación:** Hace referencia a la tensión con la que se debe activar a la bobina del Relay. Si bien la misma genera un campo magnético a través de una corriente, es necesario alimentarla. Para esto existen tensiones de 3V, 5V, 6V, 12V, 48V, etc

**Corriente y tensión máxima:** Estos parámetros hacen referencia a los contactos del mismo. Algunos niveles de corriente pueden ser de 10A o 20A. Para tensiones pueden ser de 220V.

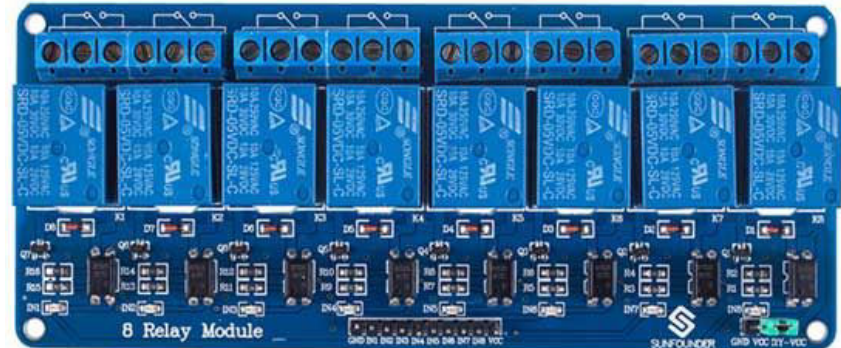
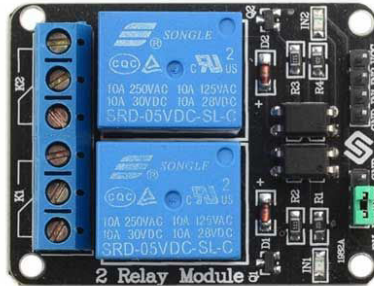
# Relays

Los Relays comentados son del tipo **electromecánicos**, existen también los de **estado sólido**.



# Relays

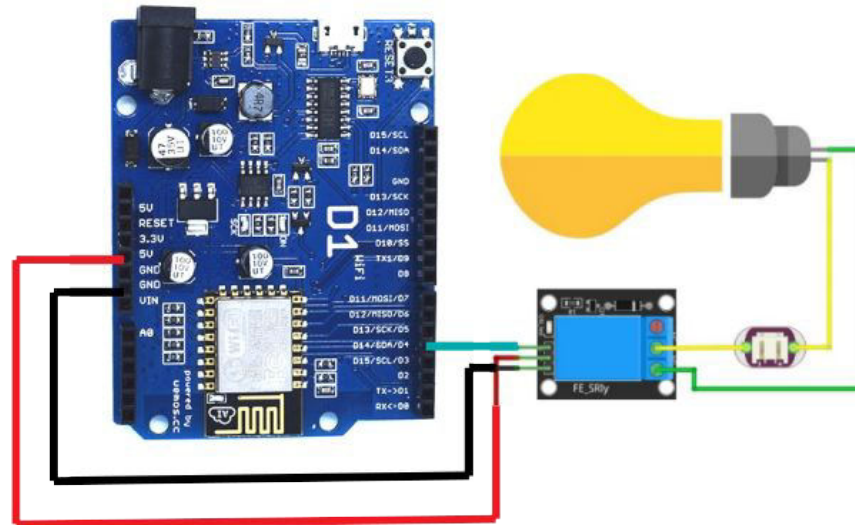
De los Relays electromecánicos, existen diversos módulos con distintas cantidades



# Relays

Para llevar estos conocimientos a la práctica, vamos a operar el **Relay** para que **apague y encienda un relay cada dos segundos**. El hardware es el siguiente:

**Ver:** [Relay\\_Wemos.ino](#)



# Introducción a los Sistemas Embebidos

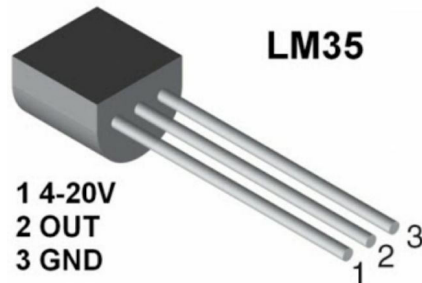
- - Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.
- - Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.
- - Manejo de puertos digitales. Utilización de Relays.
- - **Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.**
- - Manejo de servo motor.
- - Dispositivos I<sup>2</sup>C: Display LCD.

# Lectura con sensor LM35

El sensor de temperatura **LM35** convierte la temperatura en valores de tensión.

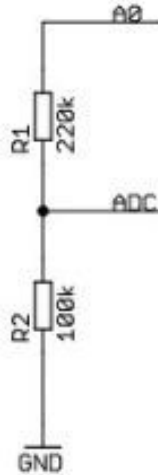
Por cada grado centígrado, entrega una tensión de 10mV. Por ejemplo, si en el interior de una habitación existen 20°C, el sensor entregará una tensión de 200mV.

**Rango de Temperatura:** -55°C a 150°C



# Lectura con sensor LM35

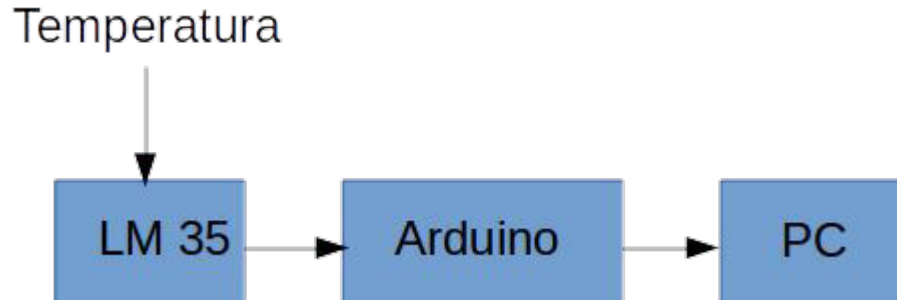
Es importante tener en cuenta que *el conversor ADC del ESP8266 soporta hasta 1,1V*. Sin embargo, en la placa, se encuentra un divisor resistivo de tensión el cual admite una tensión de 3,3V a la entrada.



Con esto, hay que tener en cuenta entonces que *la máxima tensión analógica de entrada que se le puede suministrar a la Wemos es de 3,3V*.

# Lectura con sensor LM35

**Ejemplo:** El ejercicio consistirá en leer desde Arduino, la información suministrada por el sensor de temperatura y mostrarlo por pantalla.

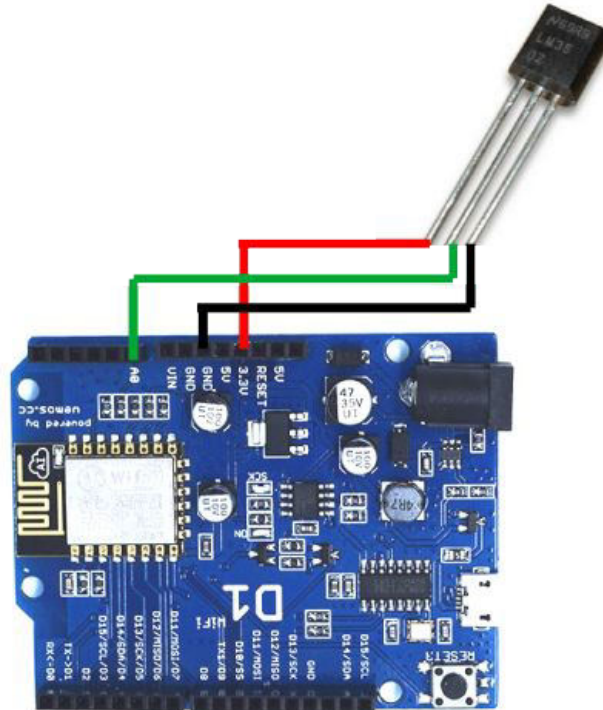


Ver: [LM35\\_Wemos.ino](#)



# Lectura con sensor LM35

Hardware



# Lectura con sensor DHT 11

Sensor **digital** de Temperatura y Humedad.  
Dentro del dispositivo, se encuentra un sensor analógico. Luego, se convierte esta temperatura a un valor digital.

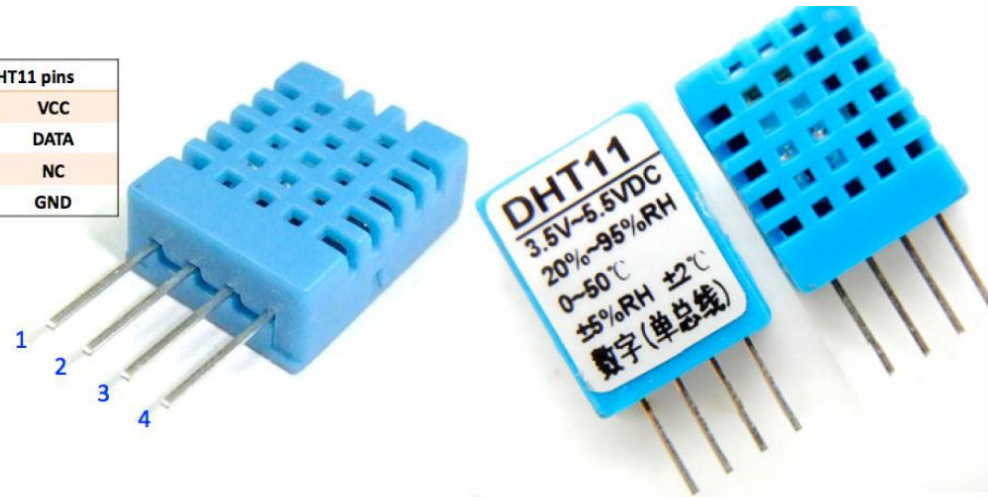
**Alimentación:** 3,5V a 5V

**Temperatura:** 0°C a 50°C

**Humedad:** 20%RH a 90%RH

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND

[7]

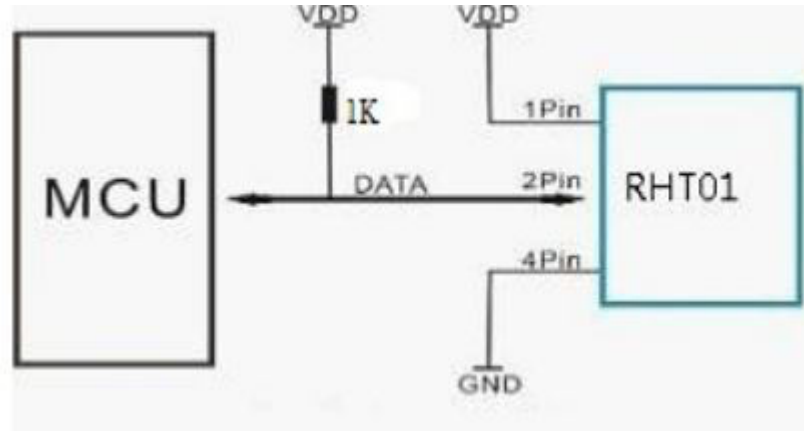


# Lectura con sensor DHT 11

Model	DHT11	
Power supply	3.3-5.5V DC	
Output signal	digital signal via Aosong 1-wire bus	
Sensing element	Polymer humidity resistor	
Operating range	humidity 20-90%RH;	temperature 0~50Celsius
Accuracy	<b>humidity +-5%RH;</b>	temperature +-2Celsius

[https://www.electronicoscaldas.com/datasheet/DHT11\\_Aosong.pdf](https://www.electronicoscaldas.com/datasheet/DHT11_Aosong.pdf)

# Lectura con sensor DHT 11



*El bus utiliza 1 cable para la comunicación entre MCU y DHT11*

# Lectura con sensor DHT 11

Este sensor dispone de una **biblioteca** llamada <DHT.h>. Dentro de ella, se pueden encontrar todas las funciones para el manejo de este sensor.

Esta biblioteca se puede instalar desde el IDE de Arduino de la siguiente forma:

Programa -> Incluir Librería -> Administrar biblioteca

Allí buscamos:

**DHT11**

# Lectura con sensor DHT 11

DHT sensor

library



**Gestor de Librerías**

Tipo: Todos Tema: Todos

**EduIntro**  
by Arduino LLC  
**Library used for super-fast introduction workshops** Is intended to be used with Arduino UNO / MICRO / MEGA / NANO classic / NANO Every / MKR / WiFi REV2 and a set of basic components (led, button, piezo, LM35, thermistor, LDR, PIR, DHT11, and servo) as a way to introduce people to the basic aspects of Arduino during short workshops.  
[More info](#)

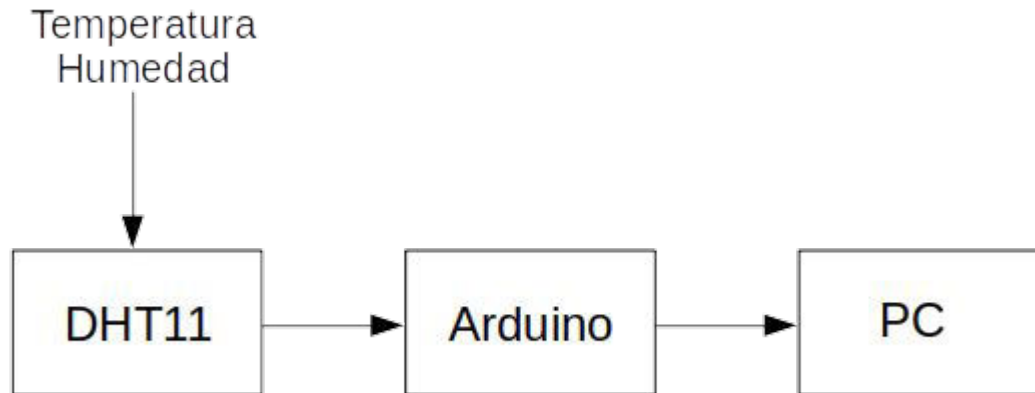
Versión 0.0.13

**DHT sensor library**  
by Adafruit Versión 1.4.2 **INSTALLED**  
**Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors** Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors  
[More info](#)

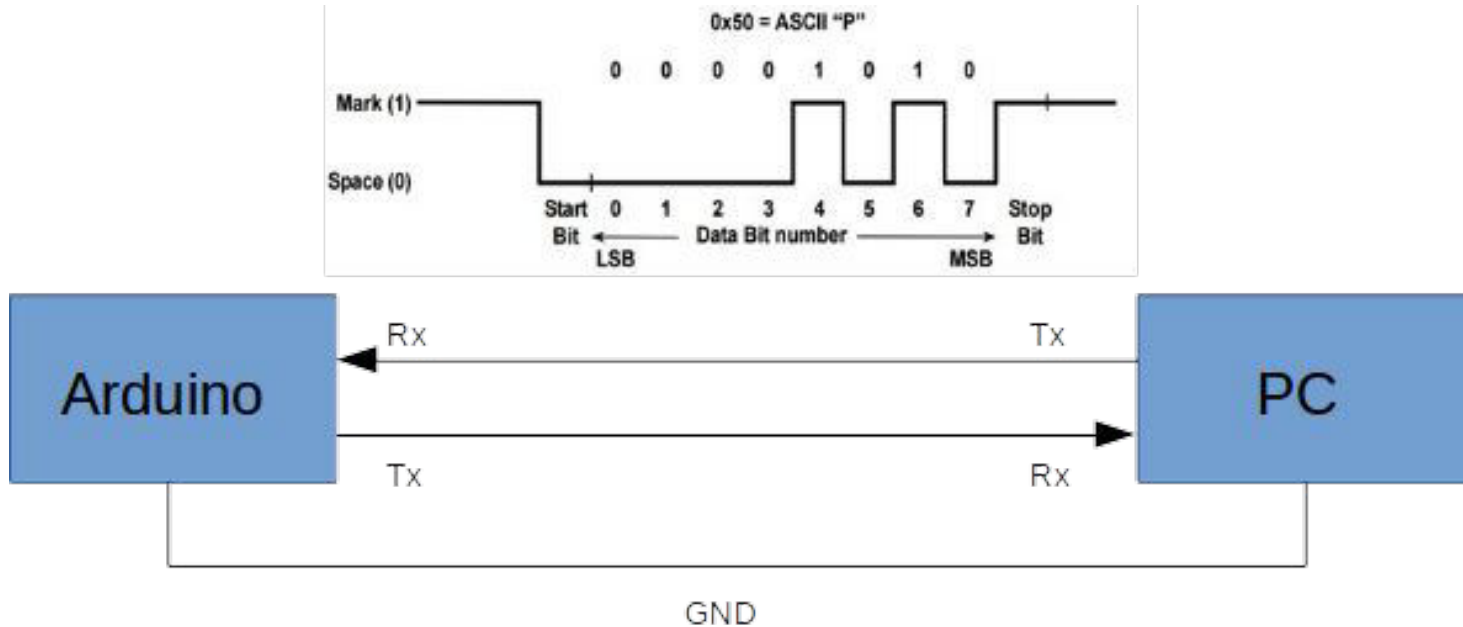
**DHT sensor library for ESPx**  
by beegee\_tokyo  
**Arduino ESP library for DHT11, DHT22, etc Temp & Humidity Sensors** Optimized library to match ESP32 requirements. Last changes: Fix negative temperature problem (credits @heljunky)  
[More info](#)

# Lectura con sensor DHT 11

**Ejemplo:** El ejercicio consistirá en leer desde Arduino, la información suministrada por el sensor de temperatura y mostrarlo por pantalla.



# Lectura con sensor DHT 11





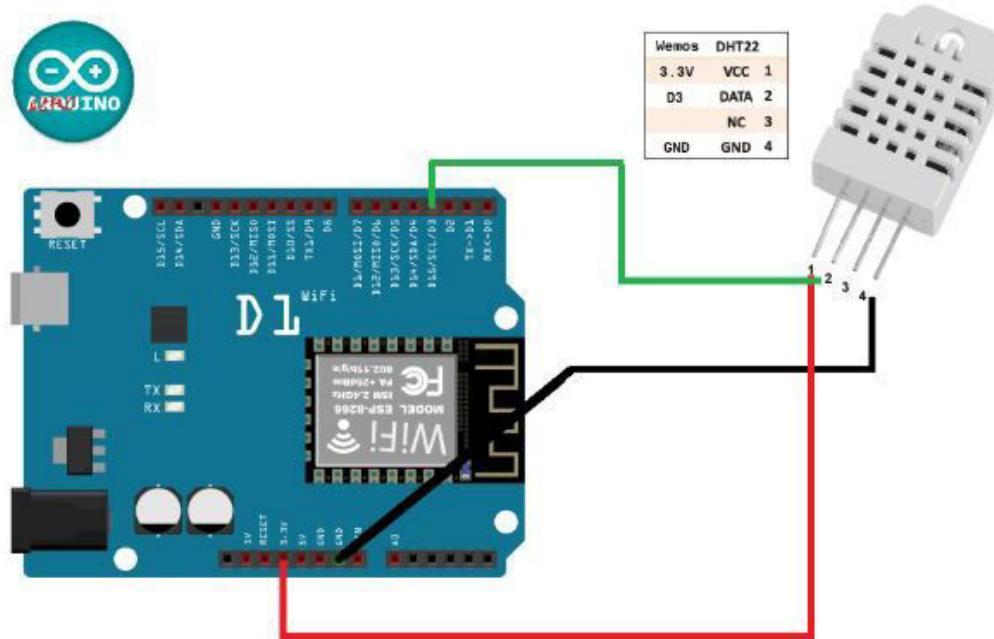
# Lectura con sensor DHT 11

Se propone el siguiente ejercicio:

Leer los datos de **Temperatura y Humedad** provistos por el sensor y **mostrarlos por el monitor serial** de Arduino.

Ver: DHT11\_Wemos.ino

# Lectura con sensor DHT 11



[26]

**Importante:** Agregar una resistencia de 10K ohms entre 5V y D3

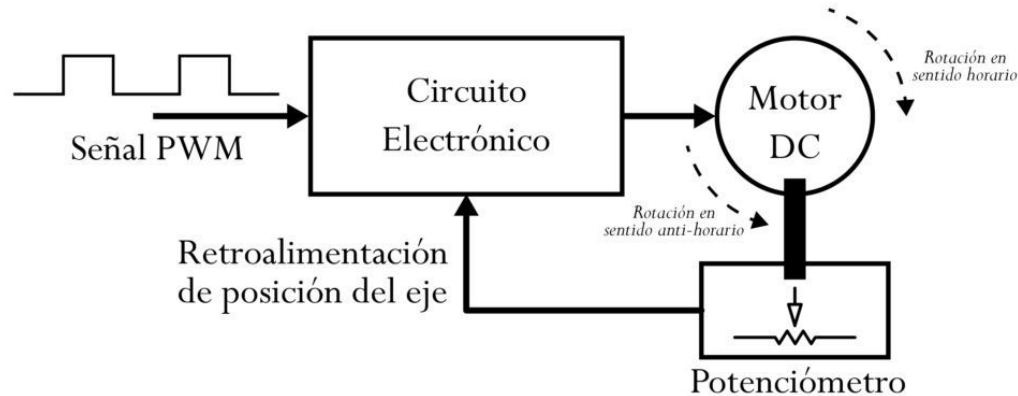
# Introducción a los Sistemas Embebidos

- - Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.
- - Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.
- - Manejo de puertos digitales. Utilización de Relays.
- - Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.
- - **Manejo de servo motor.**
- - Dispositivos I<sup>2</sup>C: Display LCD.

# Introducción a los Sistemas Embebidos

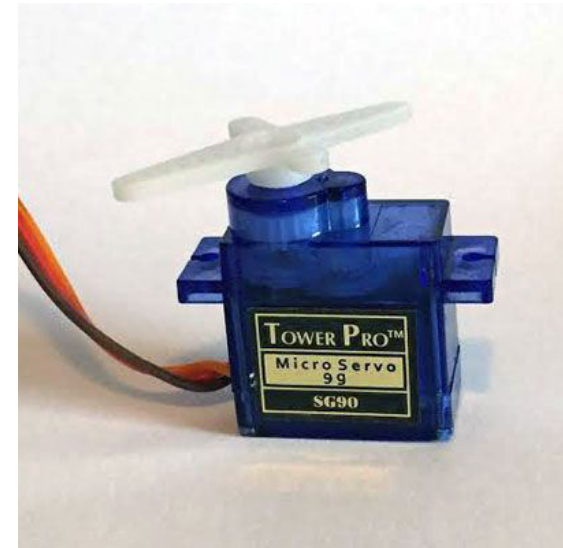
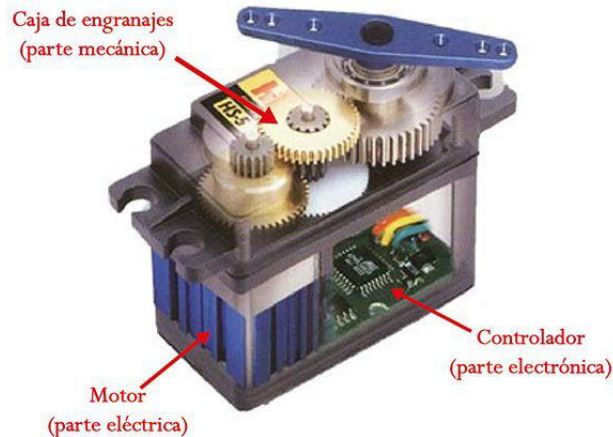
Un **servomotor** es un sistema que consta de un motor para controlar, por ejemplo, la **posición del eje**. Estos sistemas se mueven en una **determinada cantidad de grados** y luego quedan fijos allí.

## DIAGRAMA DE BLOQUE DEL SERVOMOTOR



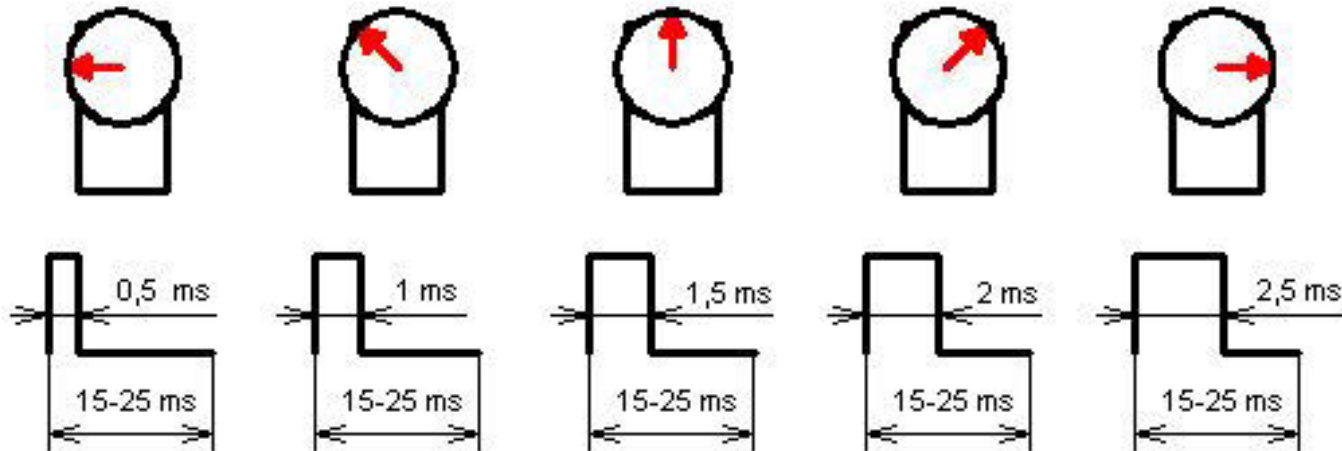
# Introducción a los Sistemas Embebidos

En particular estaremos usando el **Sg90**, el cual tiene una estructura como la que se ve a continuación:



# Introducción a los Sistemas Embebidos

**Funcionamiento:** Para el funcionamiento, se suele usar una señal **PWM** y, dependiendo del ancho del pulso, el motor se detendrá en una posición fija.



# Introducción a los Sistemas Embebidos

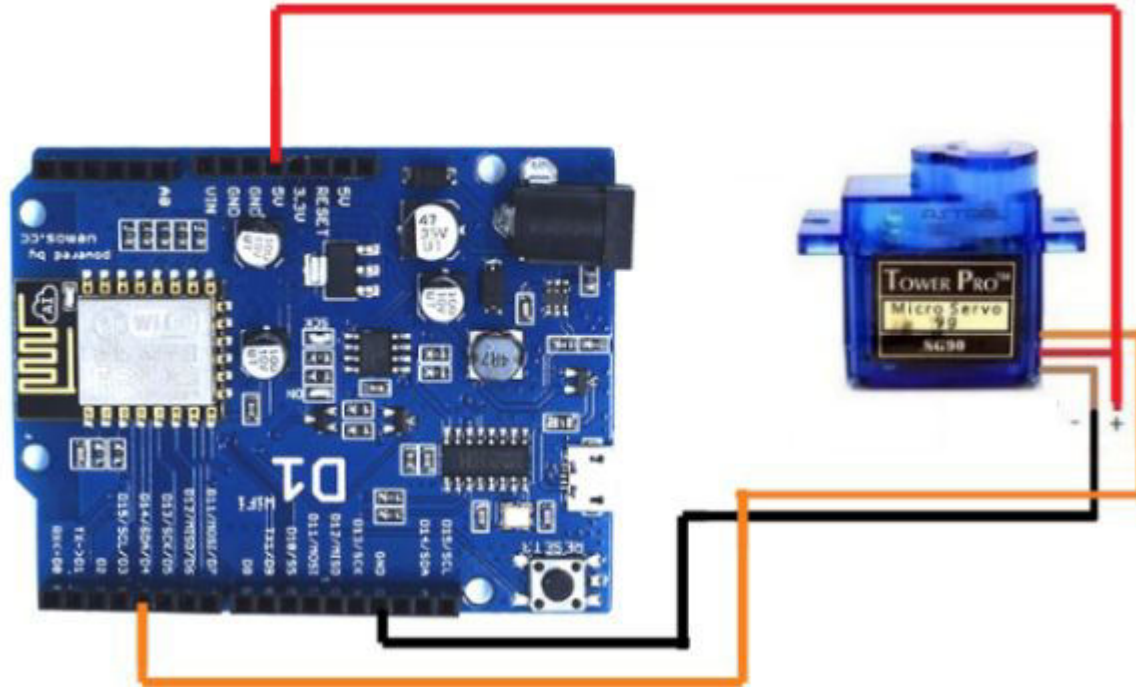
Vamos a usar la biblioteca **“Servo”**. Por esto, conviene instalar dicha biblioteca y luego estudiar el código:

**Sweep\_Wemos.ino:** Código básico para el manejo del servo

**Nota:** Algo importante a destacar es que las funcionalidades del **PWM** se pueden implementar mediante software. Para más información, ver:

[https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf)

# Introducción a los Sistemas Embebidos



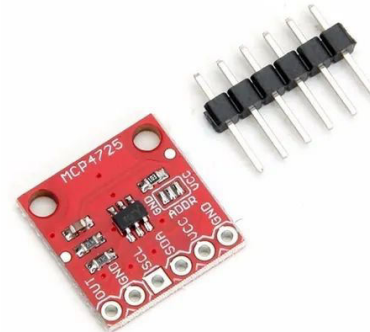
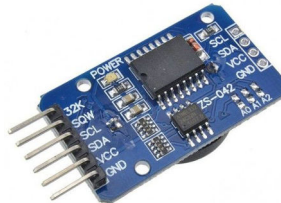
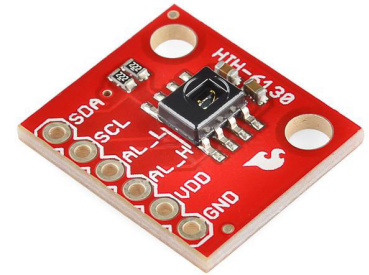
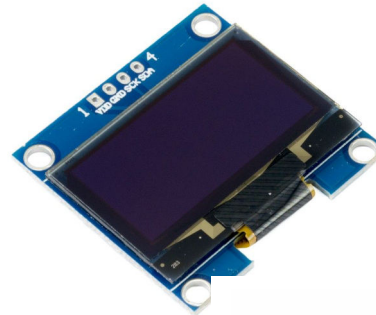
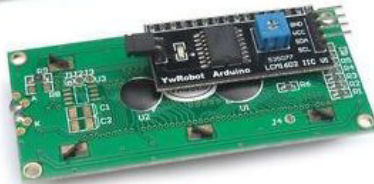


# Introducción a los Sistemas Embebidos

- - Estudio y utilización de módulos del ESP8266: UART, ADC, I<sup>2</sup>C.
- - Utilización del monitor por puerto serie como herramienta de depuración de código (debugging), función Serial.print(), velocidad de comunicación.
- - Manejo de puertos digitales. Utilización de Relays.
- - Lectura de temperatura mediante sensor analógico (LM35). Sensor DHT11.
- - Manejo de servo motor.
- - **Dispositivos I<sup>2</sup>C: Display LCD.**

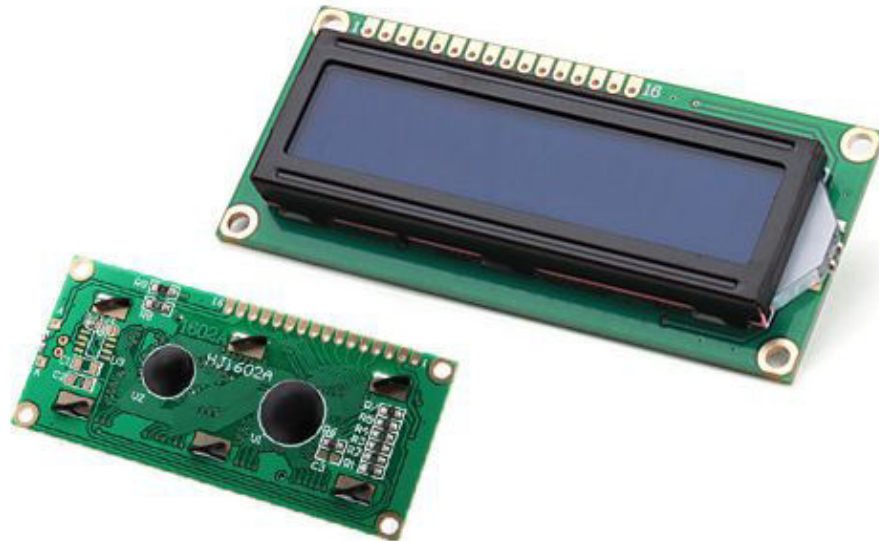
# Dispositivos I<sup>2</sup>C: Display LCD

¿cuáles dispositivos puedo conectar?



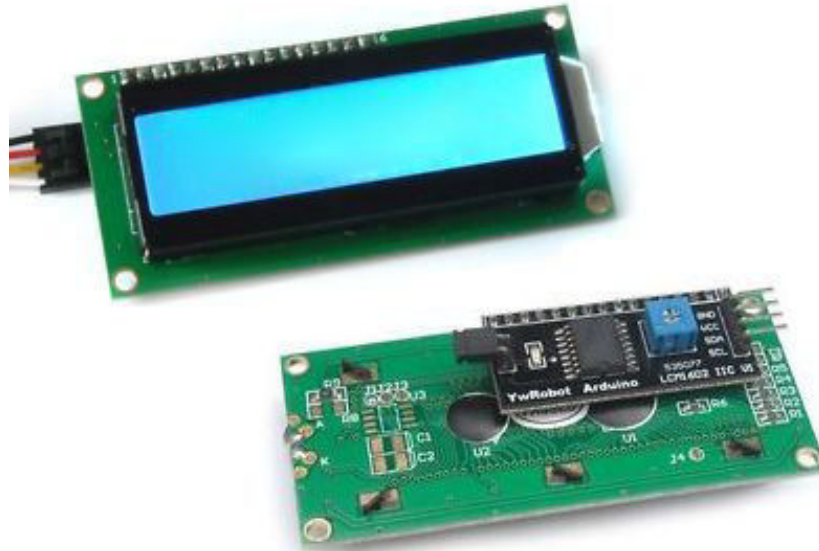
# Dispositivos I<sup>2</sup>C: Display LCD

Los **displays LCD** permiten mostrar la información en forma de matriz, algunos formatos pueden ser en **2x16**, tal como el que se muestra a continuación:



# Dispositivos I<sup>2</sup>C: Display LCD

Los que usaremos aquí contienen un módulo expensor. No todos los módulos LCD vienen con este expensor.



# Dispositivos I<sup>2</sup>C: Display LCD

El circuito integrado **PCF8574** se lo puede ver a continuación:

[5]



GND	—	GND
Vcc	—	5V
SDA	—	SDA (A4)
SCL	—	SCL (A5)

Este circuito integrado es un **expansor de entrada / salida (E / S) de 8 bits para el bus bidireccional de dos líneas (I<sup>2</sup>C)**

[https://www.ti.com/lit/ds/symlink/pcf8574.pdf?ts=1611935644461&ref\\_url=https%253A%252F%252Fwww.google.com%252F](https://www.ti.com/lit/ds/symlink/pcf8574.pdf?ts=1611935644461&ref_url=https%253A%252F%252Fwww.google.com%252F)

# Dispositivos I<sup>2</sup>C: Display LCD

En base a esto y, por lo que vimos sobre protocolo **I<sup>2</sup>C**, el dispositivo tiene una **dirección** que depende de **A0-A1-A2**. Esto depende de cómo esté configurado el módulo que contiene al **PCF8574**. Puede ser el **0x3F** o el **0x27**. Si esta dirección no está bien configurada en el código desarrollado, no se podrá establecer la comunicación.

Es decir, cuando se adquiere el módulo, muchas veces no hay información de cuál es la dirección del dispositivo por eso, para averiguar esto, se puede usar el siguiente sketch:

[i2c\\_scanner\\_arduino\\_code. ino](#)

**Más información:** <https://playground.arduino.cc/Main/I2cScanner/>

# Dispositivos I<sup>2</sup>C: Display LCD

Una vez determinado el Hardware, ahora podemos ver la [Biblioteca](#) a utilizar. En particular estaremos usando [LiquidCrystal\\_I2C](#), la cual la podemos instalar de la siguiente forma:

[Herramientas -> Administrar Bibliotecas](#)

Luego, se puede buscar: LiquidCrystal\_I2C e instalar la que dice:

[LiquidCrystal I2C by Marco Schwartz](#)

# Dispositivos I<sup>2</sup>C: Display LCD

Dentro de [LiquidCrystal\\_I2C](#), podemos encontrar funciones como:

[LiquidCrystal\\_I2C\(lcd\\_Addr, lcd\\_cols, lcd\\_rows\)](#): Esta función permite crear un objeto de la clase `LiquidCrystal_I2C`, siendo:

[lcd\\_Addr](#): dirección del dispositivo.

[lcd\\_cols](#): cantidad de columnas

[lcd\\_rows](#): cantidad de filas



# Dispositivos I<sup>2</sup>C: Display LCD

**init():** Permite Inicializar el módulo.

**clear():** Permite borrar la pantalla LCD y posicionar el cursor en la esquina superior izquierda.

**setCursor(col, row):** Esta función permite **posicionar el cursor del LCD** según lo **indicado** en **col** y **row**. Con esto se puede determinar la ubicación donde se comenzará a mostrar el texto.

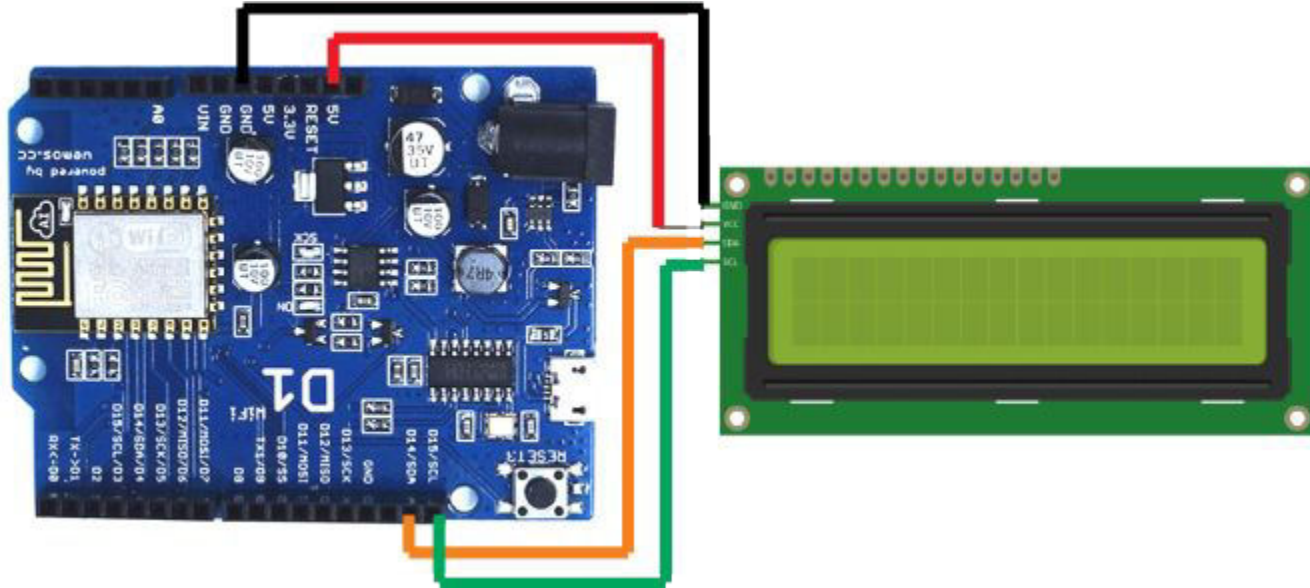
**print():** Esta función permite escribir texto o un mensaje en el LCD. Haciendo la similitud con lo practicado con el puerto serie de Arduino, sería como usar `Serial.print`

# Dispositivos I<sup>2</sup>C: Display LCD

Para llevar a la práctica estos conocimientos, vamos a mostrar la información “Hola mundo” en el display.

**Ver:** [LCD\\_2x16\\_Hola\\_Mundo.ino](#)

# Dispositivos I<sup>2</sup>C: Display LCD



# Planificación

- Introducción a IoT (Internet of Things)
- Introducción a la programación de módulos basados en ESP 8266 mediante Arduino.
- Prácticas con módulos basados en ESP8266.
- **Comunicaciones WiFi con placas basadas en ESP8266.**
- Integración de Conceptos.

# Planificación

- **Primeros pasos en la conexión de WiFi mediante Wemos D1. (modo STA).**
- Transmisión de datos de sensores a ThingsPeak. Estación meteorológica. Extracción de archivos .csv.
- ESP8266 como Web Server. HTML. Automatización de sistemas de potencia para el hogar y la industria.
- Automatización de sistemas para el hogar y la industria mediante protocolo MQTT. Análisis de Brokers.

# Primeros pasos con WiFi

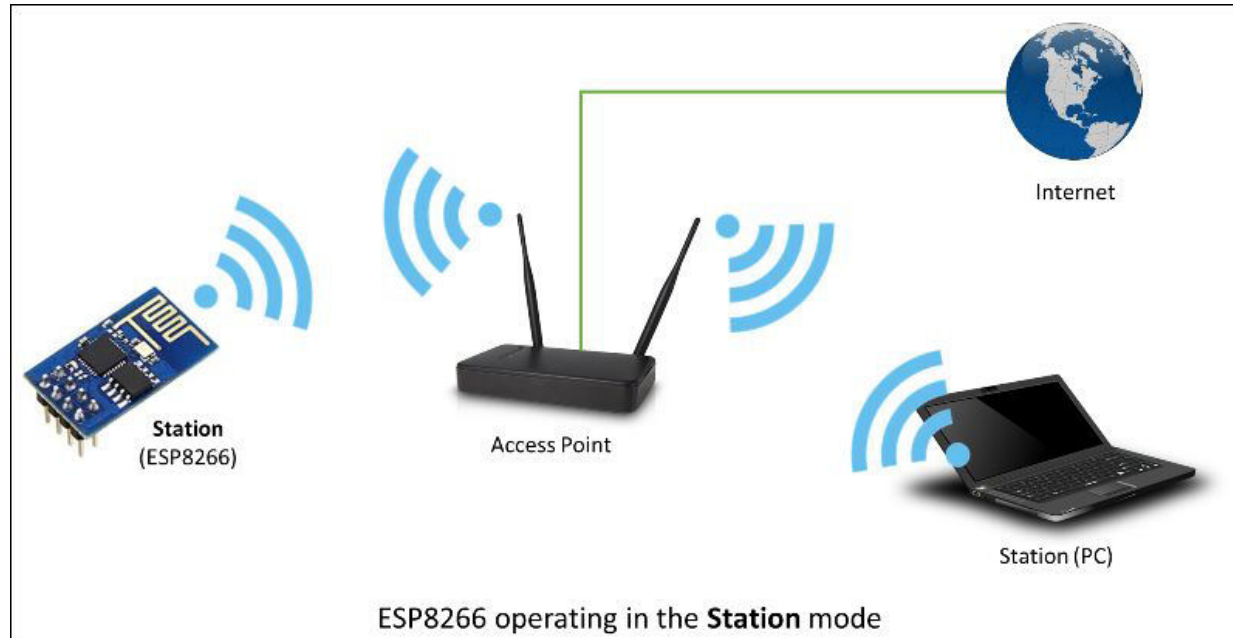
Previamente a conectar la **Wemos**, veamos algunos conceptos:

Todos los **dispositivos** que se **conectan a redes Wi-Fi** se denominan **estaciones (STA, stations)**. La **conexión a Wi-Fi** se realiza mediante un **Punto de Acceso (AP, Access Point)**, que **actúa como un concentrador** para una o más **estaciones**.

El **punto de acceso** generalmente se **integra con un enrutador** (router) para proporcionar acceso desde una **red Wi-Fi a Internet**. Cada **punto de acceso** se reconoce por un **SSID (Service Set Identifier)**, que **indica el nombre de la red**.

<https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>

# Primeros pasos con WiFi



# Primeros pasos con WiFi

De acuerdo a la imagen anterior, vamos a usar la placa con **ESP8266** en modo **estación** o **STA**.

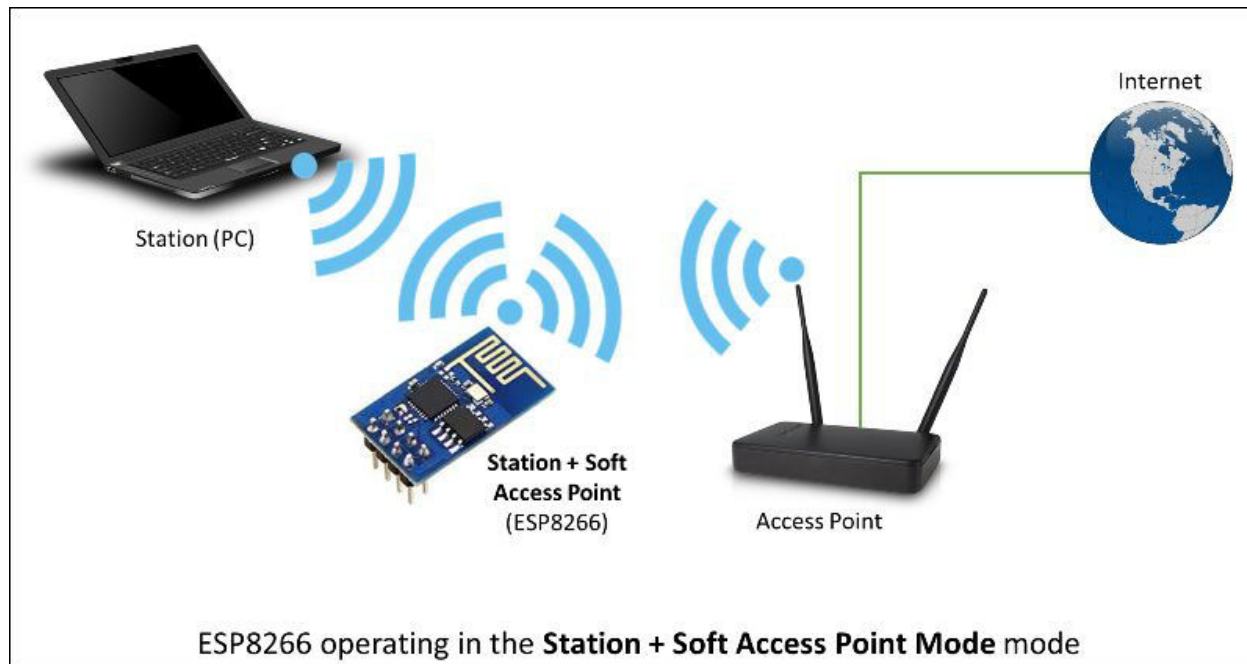
Sin embargo, es importante tener en cuenta que **los módulos ESP8266**, aparte de funcionar en modo **estación (STA)**, pueden funcionar como **punto de acceso suave** (Soft-AP), **para establecer su propia red Wi-Fi**.

Cuando el módulo **ESP8266** está funcionando como un **punto de acceso suave**, podemos **conectar otras estaciones al módulo ESP**. Es decir, el módulo ESP8266 **también puede funcionar como estación y como modo de punto de acceso suave**. Esto proporciona la posibilidad de construir, por ejemplo, redes de malla (mesh networks).

<https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>



# Primeros pasos con WiFi



# Primeros pasos con WiFi

En nuestro caso para operar mediante WiFi, vamos a usar la biblioteca:

**ESP8266WiFi**

cuya documentación se puede ver en :

<https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi>

**Nota:** Algo importante a tener en cuenta es que, ***para que funcione correctamente la biblioteca ESP8266WiFi***, tiene que estar seleccionada la placa a utilizar (Wemos R1 D1)

<https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>

# Primeros pasos con WiFi

Con estos conocimientos, pasaremos a realizar la primera conexión a **WiFi**. Para esto, conectar la placa **Wemos D1** a la PC.

**Ver:** [Wemos\\_WiFi.ino](#)

# Primeros pasos con WiFi

El código utiliza el método **status**, el cual devuelve el **estado de la conexión**.

WL\_CONNECTED: assigned when connected to a WiFi network;

WL\_NO\_SHIELD: assigned when no WiFi shield is present;

WL\_IDLE\_STATUS: it is a temporary status assigned when WiFi.begin() is called and remains active until the number of attempts expires (resulting in WL\_CONNECT\_FAILED) or a connection is established (resulting in WL\_CONNECTED);

# Primeros pasos con WiFi

WL\_NO\_SSID\_AVAIL: assigned when no SSID are available;

WL\_SCAN\_COMPLETED: assigned when the scan networks is completed;

WL\_CONNECT\_FAILED: assigned when the connection fails for all the attempts;

WL\_CONNECTION\_LOST: assigned when the connection is lost;

WL\_DISCONNECTED: assigned when disconnected from a network

<https://www.arduino.cc/en/Reference/WiFiStatus>;

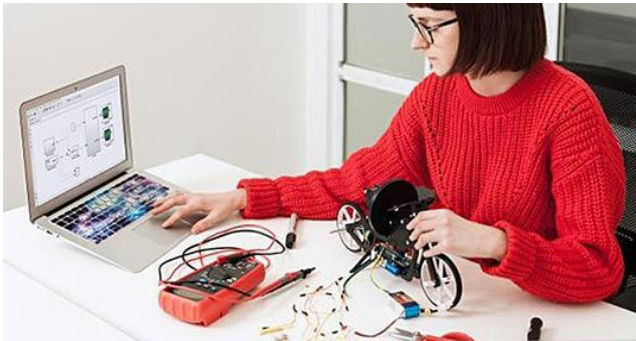
# Planificación

- Primeros pasos en la conexión de WiFi mediante Wemos D1. (modo STA).
- **Transmisión de datos de sensores a ThingsPeak. Estación meteorológica. Extracción de archivos .csv.**
- ESP8266 como Web Server. HTML. Automatización de sistemas de potencia para el hogar y la industria.
- Automatización de sistemas para el hogar y la industria mediante protocolo MQTT. Análisis de Brokers.

# ThingSpeak

Muchas veces resulta necesario poder enviar datos a la **nube** para **visualizarlos y analizarlos**. Para esto, una posible herramienta resulta ser **ThingSpeak**:

<https://thingspeak.com>



# ThingSpeak

**ThingSpeak** es una *plataforma de análisis de IoT* que permite *agregar, visualizar y analizar flujos de datos en vivo en la nube*. Se puede enviar datos a ThingSpeak desde distintos dispositivos, como por ejemplo, desde la **Wemos** que estaremos usando.

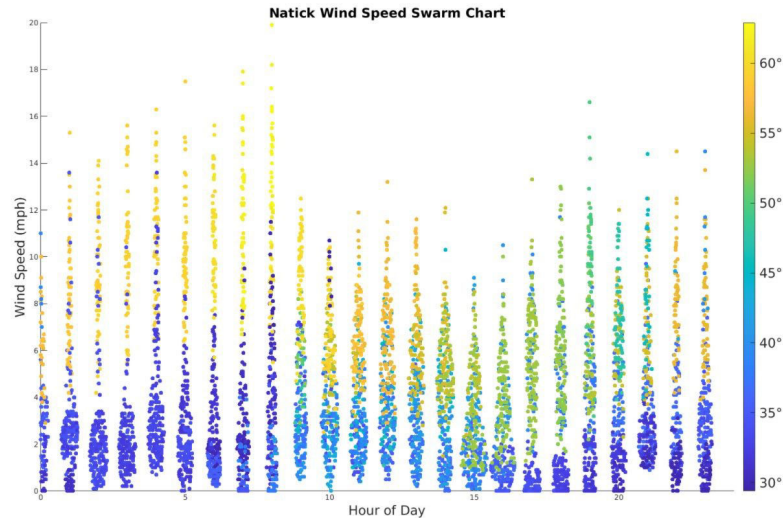


<https://thingspeak.com/>



# ThingSpeak

Además, este servicio provee herramientas de análisis mediante **MATLAB**.



<https://thingspeak.com/>

# ThingSpeak

Para poder enviar los datos a ThingSpeak y observar los datos, resulta necesario crear un **canal**.

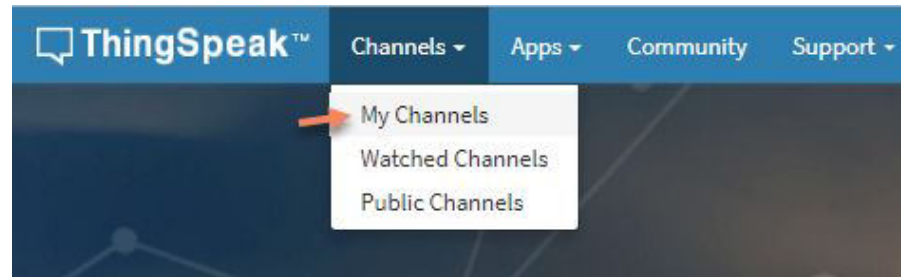
Para esto, previamente hay que hacer una **cuenta** propia en **ThingSpeak**.

Estos pasos se pueden realizar mediante la **documentación** que se encuentra en ThingSpeak.

<https://www.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html>

# ThingSpeak

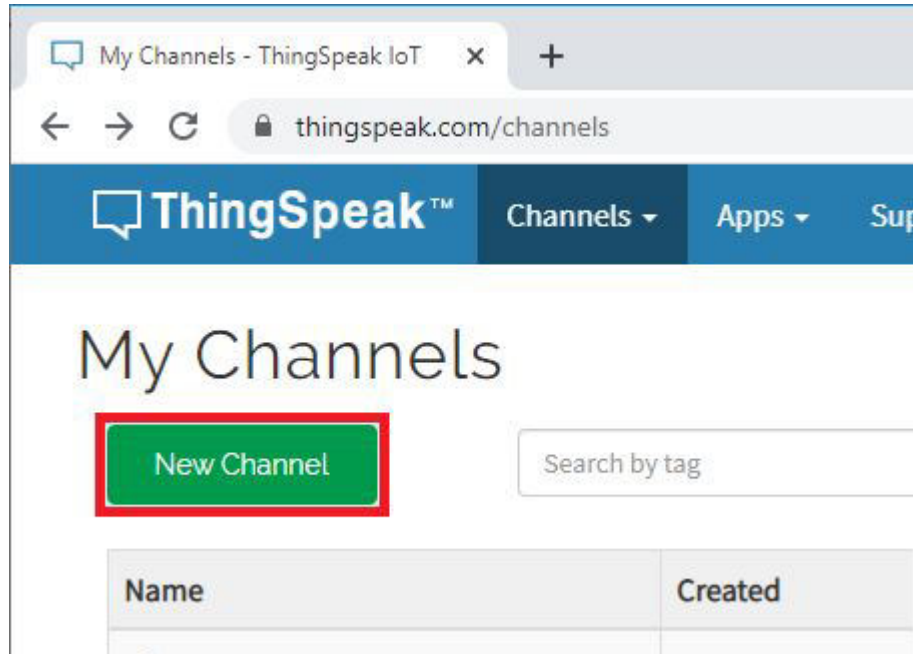
- 1) Iniciar sesión en ThingSpeak™ .
- 2) Click en Channels > MyChannels.



- 3) En la página Channels, hacer clic en **Canal Nuevo** (New Channel).

<https://www.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html>

# ThingSpeak



The screenshot shows a web browser window with the address bar displaying 'thingspeak.com/channels'. The page title is 'My Channels - ThingSpeak IoT'. The navigation bar includes the ThingSpeak logo and menu items for 'Channels', 'Apps', and 'Sup'. The main content area is titled 'My Channels' and features a green 'New Channel' button, a search box labeled 'Search by tag', and a table with columns for 'Name' and 'Created'.

Name	Created
-	

# ThingSpeak

4) Marcar las casillas junto al campo 1 (si hay más datos se tildan más campos). Se puede elegir un nombre para el campo (field), por ejemplo, **Temperatura DHT**

ThingSpeak™ Channels Apps Community Support

### New Channel

Name

Description

Field 1

Field 2

Field 3

Field 4

Show Video

YouTube

Vimeo

Video URL

Show Status

# ThingSpeak

5) Hacer clic en **Guardar** canal en la parte inferior de la configuración. Ahora se podrán ver las siguientes pestañas:

**Private View:** esta pestaña muestra información sobre el canal que solo nosotros podemos ver.

**Public View:** si se elige que el canal esté disponible públicamente, se usa esta pestaña para mostrar los campos seleccionados y las visualizaciones del canal.

**Channel Settings:** esta pestaña muestra todas las opciones de canal que estableció en la creación. Puede editar, borrar o eliminar el canal de esta pestaña.

**Sharing:** esta pestaña muestra las opciones para compartir canales. Puede configurar un canal como privado, compartido con todos (público) o compartido con usuarios específicos.

# ThingSpeak

**API Keys:** esta pestaña muestra las claves API del canal. Utilizar las teclas para leer y escribir en su canal.

**Data Import/Export:** esta pestaña permite importar y exportar datos del canal.

<https://www.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html>

# ThingSpeak

En particular, la práctica que sigue a continuación, consiste en **tomar datos de sensores** como por ejemplo, el **DHT11**, para enviarlos a **ThingSpeak** y poder visualizarlos en tiempo real.

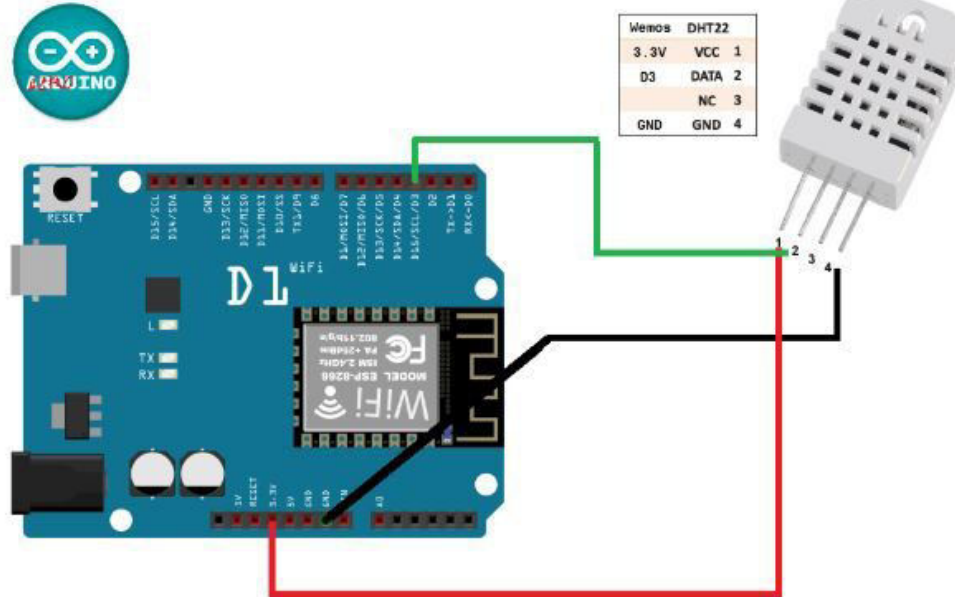
Luego se podrá descargar un archivo .csv para poder analizarlo.

**Ver:** [WeMos\\_ThingSpeak\\_DHT\\_11.ino](#)



# ThingSpeak

## Hardware



[26]

Importante: Agregar una resistencia de 10K ohms entre 5V y D3

# ThingSpeak

Una vez determinado el Hardware, ahora podemos ver la **Biblioteca** a utilizar. En particular estaremos usando **ThingSpeak**, la cual proporciona funciones o métodos para publicar los datos tomados por los sensores en un solo campo o en varios.

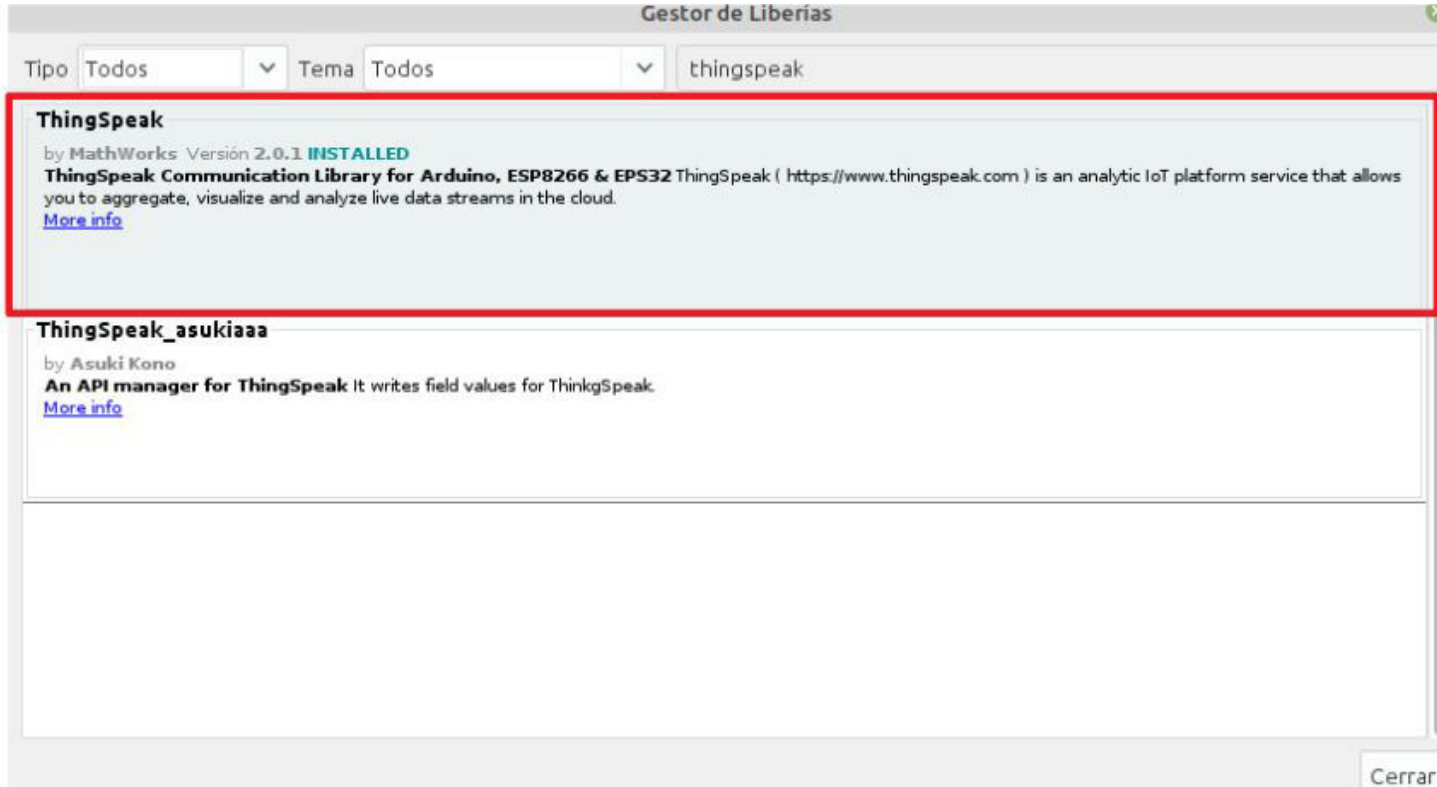
La podemos instalar de la siguiente forma:

**Herramientas -> Administrar Bibliotecas**

Luego, se puede buscar: **ThingSpeak** e instalar la que dice:

**by MathWorks.**

# ThingSpeak



Gestor de Librerías

Tipo Todos Tema Todos thingspeak

**ThingSpeak**  
by MathWorks Versión 2.0.1 **INSTALLED**  
**ThingSpeak Communication Library for Arduino, ESP8266 & ESP32** ThingSpeak ( <https://www.thingspeak.com> ) is an analytic IoT platform service that allows you to aggregate, visualize and analyze live data streams in the cloud.  
[More info](#)

**ThingSpeak\_asukiaaa**  
by Asuki Kono  
**An API manager for ThingSpeak** It writes field values for ThinkgSpeak.  
[More info](#)

Cerrar

# ThingSpeak

Conviene tener en cuenta dos cosas importantes:

- 1) En el desarrollo del código, son importantes las **API Keys** para el envío de datos desde **Wemos** (ESP8266) a ThingSpeak.

Para esto, abrir la pestaña **Claves de API** (en ThingSpeak) y copiar la clave de **API de escritura** y copiarla en el IDE de Arduino.

# ThingSpeak

2) Para el envío de datos se utilizará el método `writeField()`, el cual se puede estudiar así:

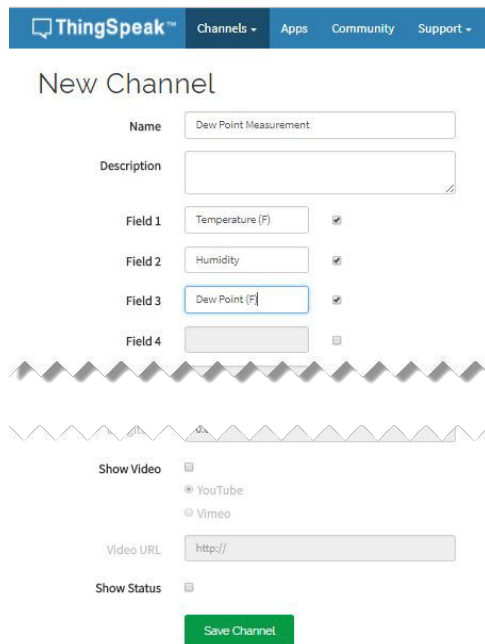
Argumentos: **Número del canal**, **Número del campo**, **Dato a publicar**, **API Key**

Devuelve: un entero que es el **código de éxito** (200)

<https://github.com/mathworks/thingspeak-arduino/blob/master/src/ThingSpeak.h>

# Múltiples datos a ThingSpeak

Para escribir en **múltiples campos**, previamente hay que setear los Fields (o campos) en **ThingSpeak**



The screenshot shows the 'New Channel' form in the ThingSpeak interface. The form includes the following fields and options:

- Name:** Dew Point Measurement
- Description:** (empty text area)
- Field 1:** Temperature (F)
- Field 2:** Humidity
- Field 3:** Dew Point (F)
- Field 4:** (empty)

Below the field configuration, there are two decorative wave patterns. Under the first, the 'Show Video' section is visible with radio buttons for YouTube (selected) and Vimeo. Below that is a 'Video URL' field containing 'http://'. At the bottom, the 'Show Status' section is visible with an unchecked checkbox. A green 'Save Channel' button is located at the very bottom of the form.

# ThingSpeak

Luego, en el código de Arduino, el siguiente método **setField ()** asigna los valores correspondientes a cada campo. Por ejemplo:

```
ThingSpeak.setField(1, temperature_C);
```

```
ThingSpeak.setField(2, humedad_rel);
```

```
ThingSpeak.setField(3, presion);
```

# ThingSpeak

Luego, se usa el método `writeFields()` para enviar los datos

```
ThingSpeak.writeField(myChannelNumber, myWriteAPIKey);
```

**Ver:** `WeMos_ThingSpeak_DHT_11_Mult.ino`

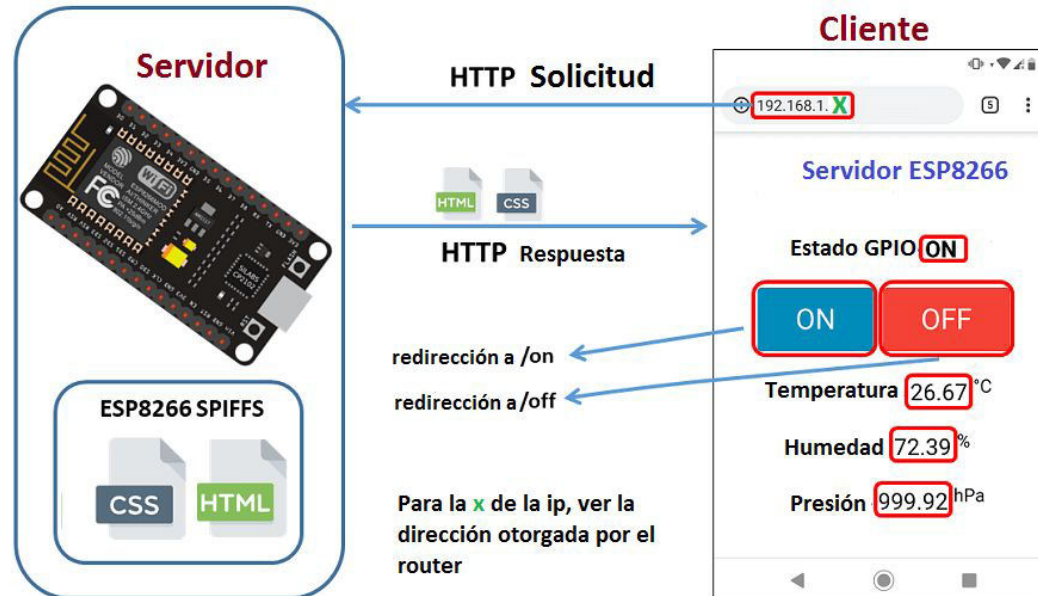


# Planificación

- Primeros pasos en la conexión de WiFi mediante Wemos D1. (modo STA).
- Transmisión de datos de sensores a ThingsPeak. Estación meteorológica. Extracción de archivos .csv.
- **ESP8266 como Web Server. HTML. Automatización de sistemas de potencia para el hogar y la industria.**
- Automatización de sistemas para el hogar y la industria mediante protocolo MQTT. Análisis de Brokers.

# ESP8266 como Web Server

**Concepto:** Crear un **servidor web** dentro de la placa Wemos para controlar dispositivos a distancia.



# ESP8266 como Web Server

De la imagen anterior se puede ver que la comunicación se hace mediante *una petición http* que se realiza desde un **cliente** (PC, celular, etc), lo cual produce que la **Wemos muestre la información que contiene en su código**.

Del lado del **cliente** se puede ver el texto con botones, información de sensores, etc. Esto resulta así gracias a los archivos **html y css** que se *incluyen en la memoria flash del ESP8266*.

# ESP8266 como Web Server

## Visualizando la página en HTML:

El concepto de **HTML** genera un curso aparte, sin embargo, veremos los conceptos básicos.

**HMTL** o **HyperText Markup Language** ('lenguaje de marcado de hipertexto'), hace referencia al *lenguaje para la elaboración de páginas web*. Define una estructura básica y un código para la *definición de contenido de una página web*, como *texto*, *imágenes*, *videos*, *juegos*, etc.

# ESP8266 como Web Server

## Práctica:

Para llevar estos conceptos a la práctica, se propone realizar un **servidor web** dentro de la placa **Wemos** para que, desde un **cliente**, se puedan **prender o apagar dos LEDs**. **Cada uno de estos LEDs** se podrán comandar, cada uno, mediante **dos botones en pantalla**, tal como se muestra en la siguiente imagen:

# ESP8266 como Web Server

  192.168.43.21

## ESP8266 Web Server

GPIO 5 - State off

ON

Led Wemos - State off

ON

# ESP8266 como Web Server

Para lograr esto, cuando se cargue el código en el ESP8266, el mismo **mostrará la IP en el Serial monitor, la cual tendrá que ser cargada en nuestro browser, para observar el estado de los botones y de los LEDs**

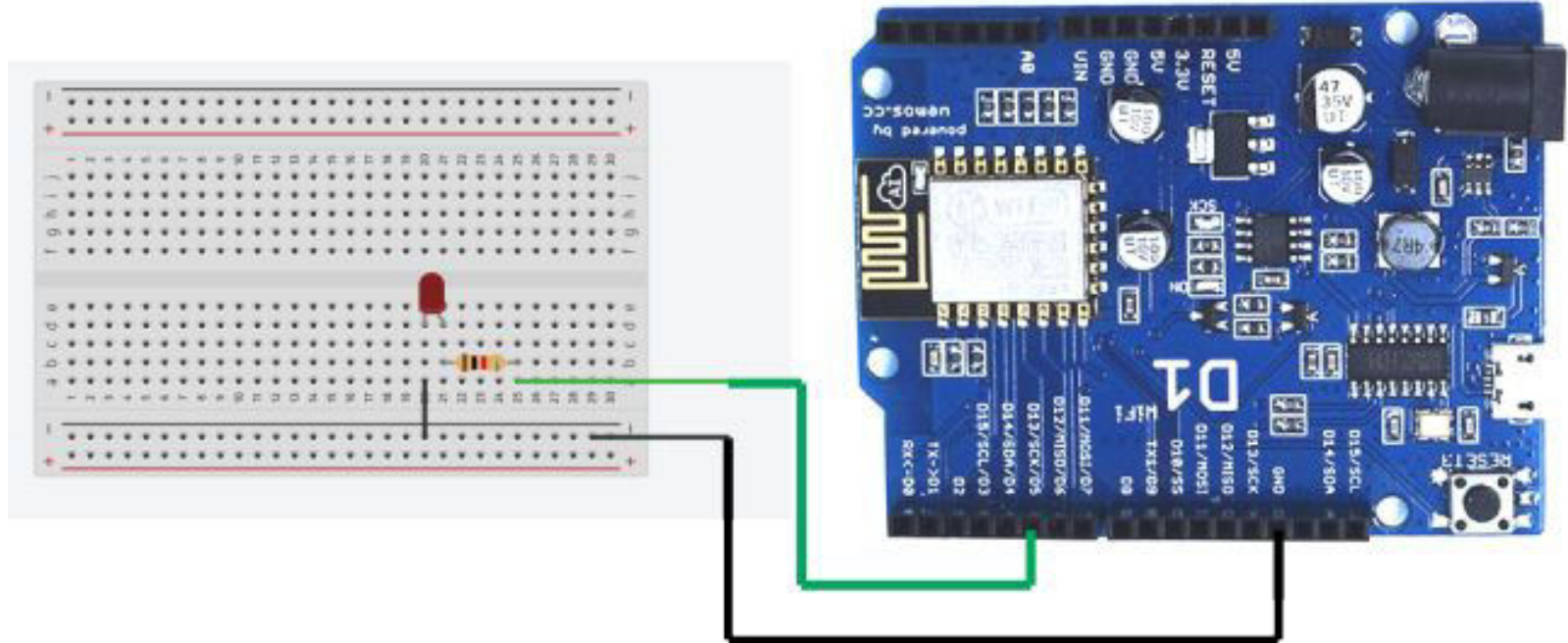
Por ejemplo, si en el Serial monitor vemos la IP: 192.168.27.43, **ésta es la que debe ser cargada en nuestro browser**. Conviene que tanto el cliente como el server, estén en la misma red.

Para esto, cuando se cargue el siguiente código en Arduino

**Ver:** ESP8266\_Web\_Server.ino

# ESP8266 como Web Server

Hardware.





# ESP8266 como Web Server

## Ítems a tener en cuenta en Arduino

- 1) La **página web** se envía al **cliente** mediante la función ***client.println()***. Esta función recibe como argumento lo que se ***desea enviar al cliente***.
- 2) ***El primer texto que siempre debe enviar es***

**<!DOCTYPE html> <html>**

la cual indica que estamos **trabajando con HTML**.

# ESP8266 como Web Server

3) Luego, la siguiente línea hace que la página web responda en cualquier navegador web.

```
client.println ("<head><meta      name=\"viewport\"      content=\"width=device-width,  
initial-scale=1\">");
```

4) Definido esto, a continuación veremos el **estilo de la página**, es decir se utiliza algo de **CSS** para diseñar los botones y la apariencia de la página web. Se elige la **fuentes Helvética**, el **contenido a mostrar como un bloque** y **alineado en el centro**.

```
client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto;  
text-align: center;}");
```

# ESP8266 como Web Server

5) Luego, se define el **estilo para un segundo botón**, con las mismas propiedades del botón definido anteriormente pero con un **color diferente**. Este será el **estilo del botón de apagado**.

```
client.println(".button2 {background-color: #77878A;}</style></head>");
```

6) En la siguiente línea, se configura el primer **encabezado de la página web**, puede cambiar este texto a lo que se quiera.

```
client.println("<h1>ESP8266 Web Server</h1>");
```

7) Luego, vemos cómo actuar para mostrar el estado del LED en el botón

```
client.println("<p>GPIO 5 - State " + output5State + "</p>");
```

# Planificación

- Primeros pasos en la conexión de WiFi mediante Wemos D1. (modo STA).
- Transmisión de datos de sensores a ThingsPeak. Estación meteorológica. Extracción de archivos .csv.
- ESP8266 como Web Server. HTML. Automatización de sistemas de potencia para el hogar y la industria.
- **Automatización de sistemas para el hogar y la industria mediante protocolo MQTT. Análisis de Brokers.**

# Protocolo MQTT

El proyecto que estaremos armando, se basa en la idea de una **red de área doméstica** utilizando la placa **Wemos D1**.

La idea de este proyecto es que la red de área doméstica se puede controlar desde una **PC a través de Internet**.

Esta red tiene como objetivo **controlar un LED** a través de Internet. Esto resulta ser a los efectos de la práctica pero, en vez de un LED, se puede conectar un **Relay** u otro actuador que resulte de interés.

A su vez, mientras se controla el led, la placa **Wemos** enviará mensajes hacia la PC.

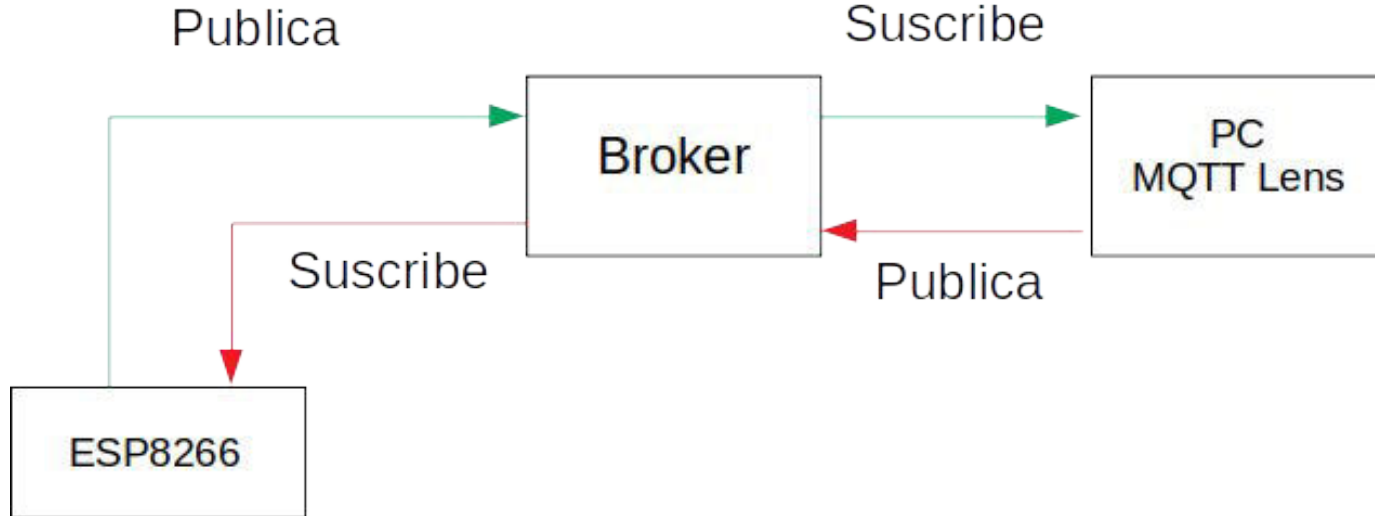
# Protocolo MQTT

En este caso la **PC remota actúa como otro dispositivo IoT**. La PC **se conecta con el dispositivo módulo Wemos a través del broker MQTT**.

En este caso se utiliza el **EMQ X** como **broker** aunque se puede usar cualquier otro.

Para que la PC se pueda conectar, se utilizará un **complemento de Google Chrome: MQTTLens**

# Protocolo MQTT



# Protocolo MQTT

Del lado de la PC, *instalar MQTTLens* en el navegador *Chrome*. *MQTTlens admite el protocolo MQTT y se puede utilizar para la publicación y suscripción de mensajes.*

Al *cliente PC se le debe asignar un ID* para que el *broker de MQTT* pueda identificar cuál cliente está *publicando y suscribiendo el tema* y la *carga útil* (payload).



# Protocolo MQTT

## Instalación MQTT Lens

- 1) En el buscador de Google, agregar: **MQTT Lens**.
- 2) En el link ubicado, al hacer click, se verá la siguiente imagen:

[Inicio](#) > [Aplicaciones](#) > MQTTLens



### MQTTLens

Ofrecido por: MQTTLens

★★★★★ 153 | [Extensiones](#) |  100.000+ usuarios

Añadir a Chrome

- 3) **Añadir a Chrome**

# Protocolo MQTT

¿Cómo funciona todo el proyecto?



# Protocolo MQTT

El **cliente ESP8266** se configura como **publicador** (publisher) para el tópico “**outTopic\_1**” y como **suscriptor** para el tópico “**inTopic\_1**”.

El **cliente de PC se configura** como **suscriptor** para el tópico “**outTopic\_1**” y como **publicador** (publisher) para el tópico “**inTopic\_1**”.

El cliente **ESP8266** inicia la conexión publicando el mensaje “**me conecté!**” al **broker** de MQTT.

# Protocolo MQTT

El **cliente PC** puede **publicar mensajes** como "1" y "0" sobre el tópicó "inTopic\_1".

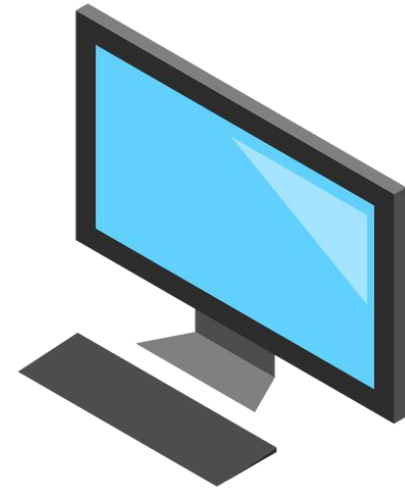
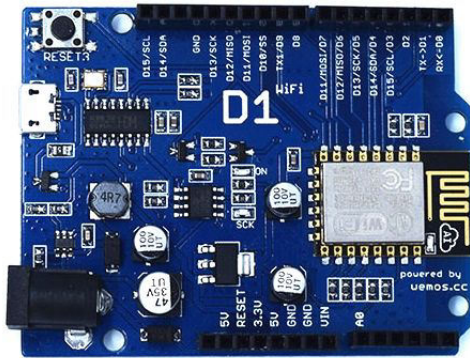
Si el **mensaje "1"** es **publicado** por el **cliente PC**, será **recibido por el cliente ESP8266** y su firmware interpreta el mensaje para **encender el LED**.

Si el **mensaje "0"** es **publicado** por el **cliente PC**, será **recibido por el cliente ESP8266** y su firmware interpreta el mensaje para **apagar el LED**.

**Ver:** MQTT\_LED.ino

# Protocolo MQTT

## Hardware



# Protocolo MQTT

Una vez determinado el Hardware, ahora podemos ver la **Biblioteca** a utilizar. En particular estaremos usando **pubsubclient**, la cual permite que la placa se comporte como un **cliente MQTT** para lograr la **suscripción** y **publicación** de mensajes mediante **MQTT**.

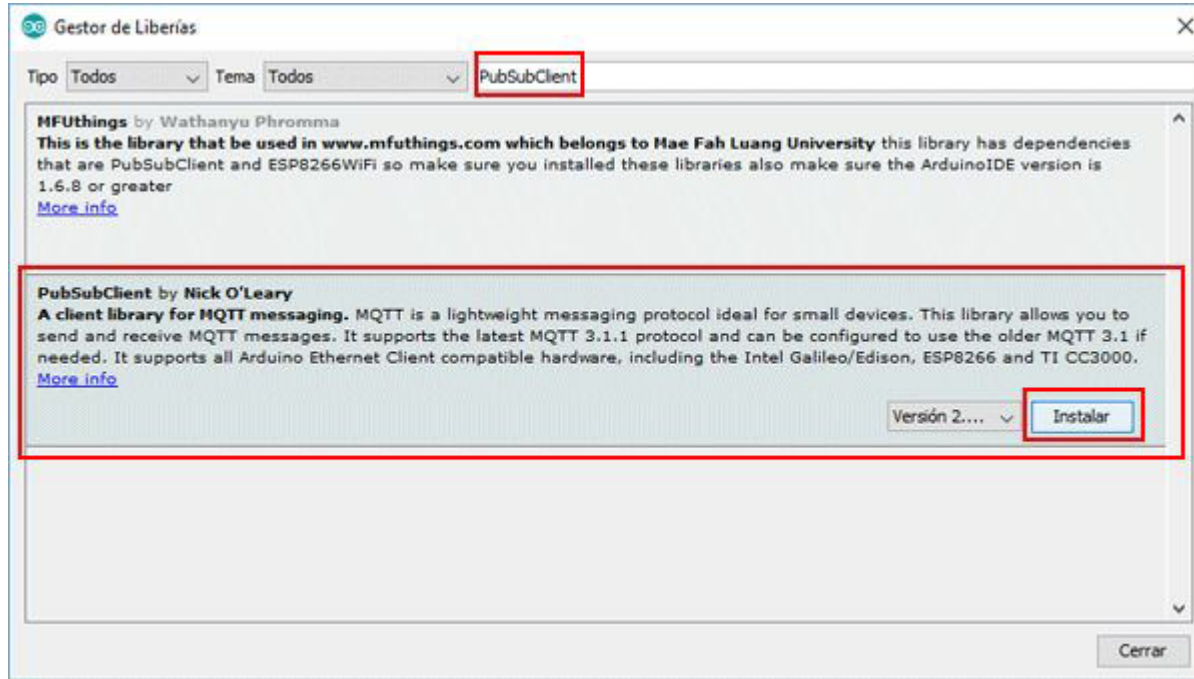
La podemos instalar de la siguiente forma:

**Herramientas -> Administrar Bibliotecas**

Luego, se puede buscar: PubSubClient e instalar la que dice:

**PubSubClient by Nick O'Leary**

# Protocolo MQTT



**CePETel**

Sindicato de los Profesionales  
de las Telecomunicaciones

**SECRETARÍA TÉCNICA**



*Instituto Profesional de  
Estudios e Investigación*

**Mayo 2022**

# **Protocolo MQTT**

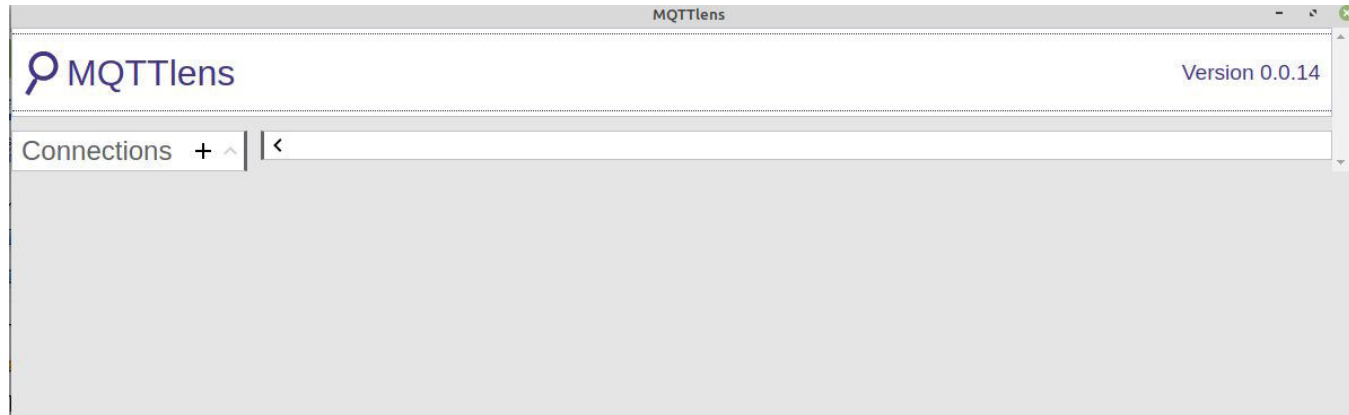
Para más información sobre la biblioteca:

<https://pubsubclient.knolleary.net/>



# Protocolo MQTT

Por último, resta la configuración de **MQTT Lens**. Para esto, al iniciar la aplicación se puede ver un cuadro como el siguiente:



- 1) Hacer click en el ícono "+", para agregar una nueva conexión

# Protocolo MQTT

Luego, se abrirá una ventana como la siguiente:

Add a new Connection ✕

---

### Connection Details

<b>Connection name</b>	<input type="text" value="Eclipse MQTT"/>	<b>Connection color scheme</b>	<input type="color" value="#00FF00"/>
<b>Hostname</b>	<input data-bbox="511 742 633 791" type="text" value="tcp://"/> <input type="text" value="e.g. iot.eclipse.org"/>	<b>Port</b>	<input type="text" value="1883"/>
<b>Client ID</b>	<input type="text" value="lens_M3ouummyMG4TMFMFdCftJBdAPcnM"/> <input type="button" value="Generate a random ID"/>		
<b>Session</b>	<input checked="" type="checkbox"/> Clean Session	<b>Automatic Connection</b>	<input checked="" type="checkbox"/> Automatic Connection
		<b>Keep Alive</b>	<input type="text" value="120"/> seconds

# Protocolo MQTT

Luego, se abrirá una ventana como la siguiente:

**Connection Name:** PC\_MQTT

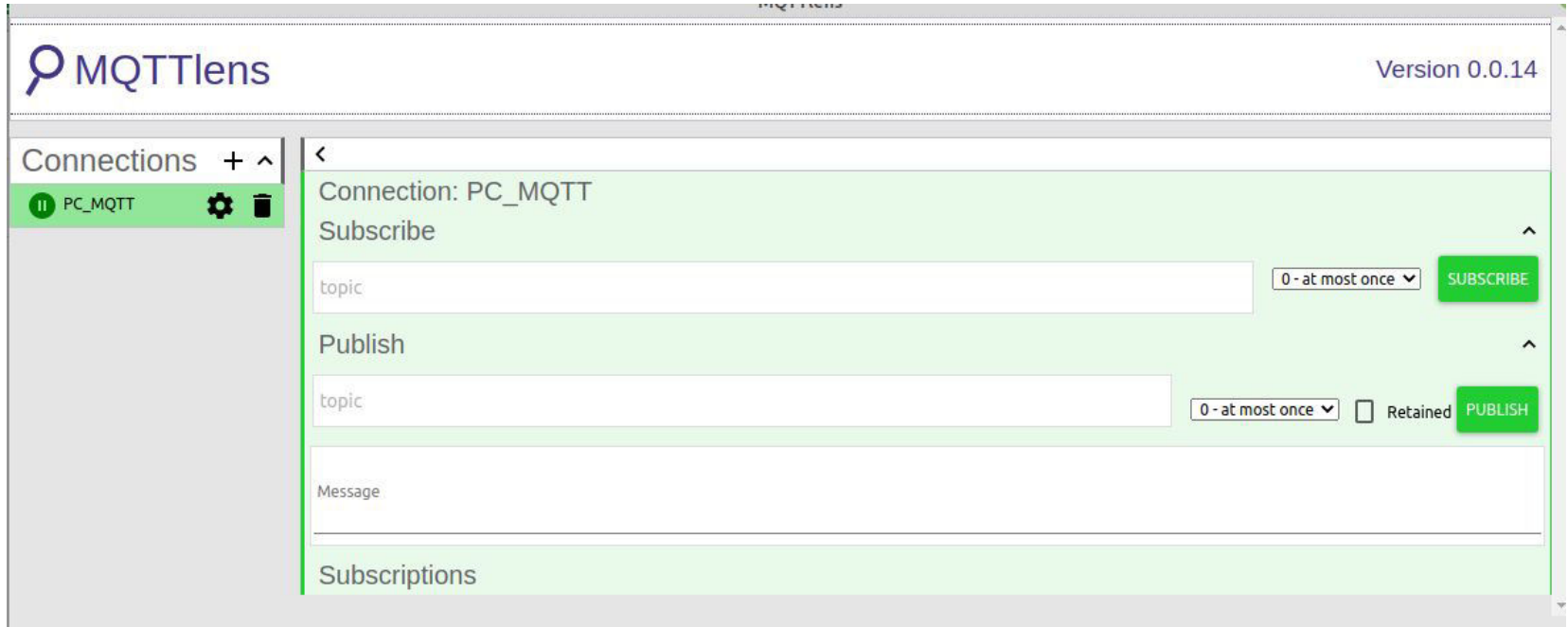
**Hostname: Nombre del Broker:** broker.emqx.io

**Port:** 1883

Por último, darle click a **CREATE CONNECTION**

# Protocolo MQTT

Luego, se abrirá una ventana como la siguiente. Aquí podemos **Publicar** y **Suscribir**



# Protocolo MQTT

## Suscripción:

Completar de la siguiente forma:

**Subscribe:** outTopic\_1

Darle click al botón **Subscribe**. Luego de esto, se verá más abajo cómo la **aplicación** (MQTT LENS) se suscribe al cliente y muestra los mensajes publicados

# Protocolo MQTT

## Publicación:

Completar de la siguiente forma:

**Publish:** inTopic\_1

Luego, debajo, escribir el mensaje deseado. En este caso será **1** o, de otra forma, **0**, y darle click al botón **Publish**.

Luego de esto, se verá más abajo cómo el **LED** de la placa **Wemos**, se prende (o apaga).

# Protocolo MQTT

En base a la práctica realizada, se puede estudiar y comprobar el funcionamiento de otros **brokers**. A continuación se comenta una posible lista de brokers:

[https://github.com/mqtt/mqtt.org/wiki/public\\_brokers](https://github.com/mqtt/mqtt.org/wiki/public_brokers)

# Planificación

- Introducción a IoT (Internet of Things)
- Introducción a la programación de módulos basados en ESP 8266 mediante Arduino.
- Prácticas con módulos basados en ESP8266.
- Comunicaciones WiFi con placas basadas en ESP8266.
- **Integración de Conceptos.**



# Integración de Conceptos.

- 1) a) Diseñar un sistema, utilizando **MQTT**, que permita suscribirse a un tópico y **activar un Relay**. A la vez, se debe mostrar la información de activación del relay mediante un display LCD 2x16.  
  
b) Agregar un sensor de temperatura al sistema y publicar los datos de temperatura.  
  
c) Si los datos del sensor de temperatura superan un determinado umbral (por ejemplo, 20 °C), activar o desactivar el Relay (Opcional)

# Integración de Conceptos.

2) Diseñar un sistema, mediante **MQTT**, que permita suscribirse a un tópico y **activar un Servomotor**. El ángulo del servo puede variar de acuerdo a le mensaje que envíe el publicador.

# Integración de Conceptos.

3) Diseñar un sistema que envíe datos de temperatura a **ThingSpeak** y, a la vez, muestre los datos mediante un display LCD de 2x16.

El envío de datos debe ser múltiple, teniendo en cuenta datos de distintos sensores. Por ejemplo, LM 35, DHT, etc.

# Integración de Conceptos.

4) Diseñar un sistema, mediante un servidor, para activar un servomotor y un relay. Tener en cuenta el diseño del servidor web para facilitar el uso de estos dispositivos.

# Referencias

- [1] <https://ros.2021discountoutlets.ru/category?name=wemos%20d1%20mini%20pinout%20led>
- [2] Designing Embedded Systems and the Internet of Things with ARM. Perry Xiao
- [3] <http://www.steves-internet-guide.com/mqtt-works/>
- [4] <https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/>
- [5] <https://aprendiendoarduino.wordpress.com/2018/10/17/pantalla-lcd-i2c-en-arduino/>
- [6] <https://www.luisllamas.es/arduino-lcd-hitachi-hd44780/>
- [7] <http://panamahitek.com/que-es-y-como-funciona-un-servomotor/>
- [8] <https://forum.arduino.cc/t/controling-5v-relay-with-wemos-d1-r1/671433>
- [9] <https://programarfacil.com/blog/arduino-blog/rele-con-arduino-lampara/>
- [10] <https://www.tecnoajudes.com/es/control-por-voz-de-dispositivos-con-esp8266/>

# Referencias

- [11] <http://diymakers.es/usando-el-puerto-serie-del-arduino/>
- [12] <http://programaciondeavr.blogspot.com/2018/12/usart-o-uart.html>
- [13] <https://www.murkyrobot.com/guias/comunicacion/i2c>
- [14] <https://aprendiendoarduino.wordpress.com/2017/07/09/i2c/>
- [15] <https://www.engineersgarage.com/controlling-an-led-light-with-pc-using-esp8266-based-han-and-hivemq-broker-iot-part-20/>
- [16] <https://biql.es/tooling-tuesday-wemos-d1-mini-micropython/>
- [17] <https://aprendiendoarduino.wordpress.com/tag/tcpip/>
- [18] <https://aprendiendoarduino.wordpress.com/tag/tcpip/>
- [19] <http://esp32.net/>
- [20] <https://www.studiopieters.nl/esp32-pinout/>

# Referencias

- [21] <https://www.tecnologia-informatica.es/Servidor-web-con-nodemcu-esp8266/>
- [22] <https://randomnerdtutorials.com/esp8266-web-server/>
- [23] <https://www.monarcaelectronica.com.ar/productos/wemos-d1-wifi-uno-shield-esp8266-arduino-uno-mona/>
- [24] <https://www.profetolocka.com.ar/2020/05/15/utilizando-la-placa-wemos-d1/>
- [25] <https://www.electronicclinic.com/wemos-d1-esp8266-arduino-compatible-its-specs-and-how-to-use-it/>
- [26] <http://arubia45.blogspot.com/2019/04/arduino-wemos-sensor-mqtt-home-assistant.html>
- [27] <https://elosciloscopio.com/tutorial-pantalla-lcd-arduino-esp8266-esp32/>
- [28] <https://uelectronics.com/producto/wemos-d1-wifi-esp8266-esp-12f-compatible-con-arduino/>
- [29] <https://aprendiendoarduino.wordpress.com/2021/02/20/mqtt-y-esp8266/>
- [30] <https://randomnerdtutorials.com/esp8266-nodemcu-thingspeak-publish-arduino/>