Técnico Superior en Telecomunicaciones

Electrónica Microcontrolada

Actividad 2

Profesores:
Jorge E. Morales.
Gonzalo Vera.

Grupo N° 7

Integrantes:
Godoy, Gustavo Alberto
Maldonado, Darío
Morales, Rebeca Ruth
Mulka, Sharon Micaela
Otero, Carlos Nahuel
Ruiz, Gloria

SOC ESP32s.

El chip ESP32 viene con 48 pines con múltiples funciones. No todos los pines están expuestos en todas las placas de desarrollo ESP32, y algunos pines no se pueden usar.

Características ESP32

Sensor de Pasillo

La placa de desarrollo ESP32 cuenta con un sensor de efecto Hall incorporado que detecta cambios en el campo magnético en su entorno. Se puede usar el sensor de efecto hall ESP32 con Arduino IDE y MicroPython.

Un sensor de efecto hall puede detectar variaciones en el campo magnético en su entorno. Cuanto mayor sea el campo magnético, mayor será el voltaje de salida del sensor.

El sensor de efecto Hall se puede combinar con una detección de umbral para actuar como un interruptor, por ejemplo. Además, los sensores de efecto Hall se utilizan principalmente para:

- Detectar proximidad;
- Calcular posicionamiento;
- Contar el número de revoluciones de una rueda;
- Detectar el cierre de una puerta;
- Y mucho más.
- El ESP32 cuenta con un sensor de efecto Hall incorporado
- El sensor de efecto Hall puede detectar cambios de campo magnético en su entorno.
- Las mediciones del sensor pueden aumentar o volverse negativas dependiendo del polo magnético que esté frente al sensor.

Sensor Táctil

Los pines táctiles ESP32 pueden detectar variaciones en cualquier cosa que tenga una carga eléctrica. A menudo se utilizan para despertar el ESP32 del sueño profundo

- El ESP32 tiene 10 GPIO táctiles capacitivos.
- Cuando toca un GPIO sensible al tacto, el valor leído por el sensor cae.
- Puede establecer un valor de umbral para hacer que suceda algo cuando detecta contacto.
- Los pines táctiles ESP32 se pueden usar para despertar el ESP32 del modo de suspensión profunda.

ESP32 I2C

El ESP32 tiene dos interfaces de bus I2C que pueden servir como maestro o esclavo I2C. Tenemos el protocolo de comunicación I2C con ESP32 usando Arduino IDE: cómo elegir pines I2C, conectar múltiples dispositivos I2C al mismo bus y cómo usar las dos interfaces de bus I2C.

Memoria Flash

Almacena datos permanentes (escritura y lectura)

Los datos guardados en la memoria flash permanecen allí incluso cuando el ESP32 se reinicia o cuando se desconecta la alimentación.

Para leer y escribir desde la memoria flash ESP32 usando Arduino IDE, usaremos la biblioteca EEPROM. Usar esta biblioteca con ESP32 es muy similar a usarla con Arduino

ESP32 de doble núcleo

El ESP32 viene con 2 microprocesadores Xtensa LX6 de 32 bits: core 0 y core 1. Entonces, es dual core. Cuando ejecutamos código en Arduino IDE, de forma predeterminada, se ejecuta en el núcleo 1. Puede ejecutar piezas de código simultáneamente en ambos núcleos y hacer que su ESP32 sea multitarea.

- **Nota:** no necesariamente necesita ejecutar doble núcleo para lograr la multitarea.
- Cuando subimos el código al ESP32 usando el IDE de Arduino, simplemente se ejecuta; no tenemos que preocuparnos de qué núcleo ejecuta el código.
- El ESP32 es de doble núcleo;
- Los bocetos de Arduino se ejecutan en el núcleo 1 de forma predeterminada;
- Para usar core 0 necesitas crear tareas;
- Puedes usar elxTaskCreatePinnedToCore()función para fijar una tarea específica a un núcleo específico;

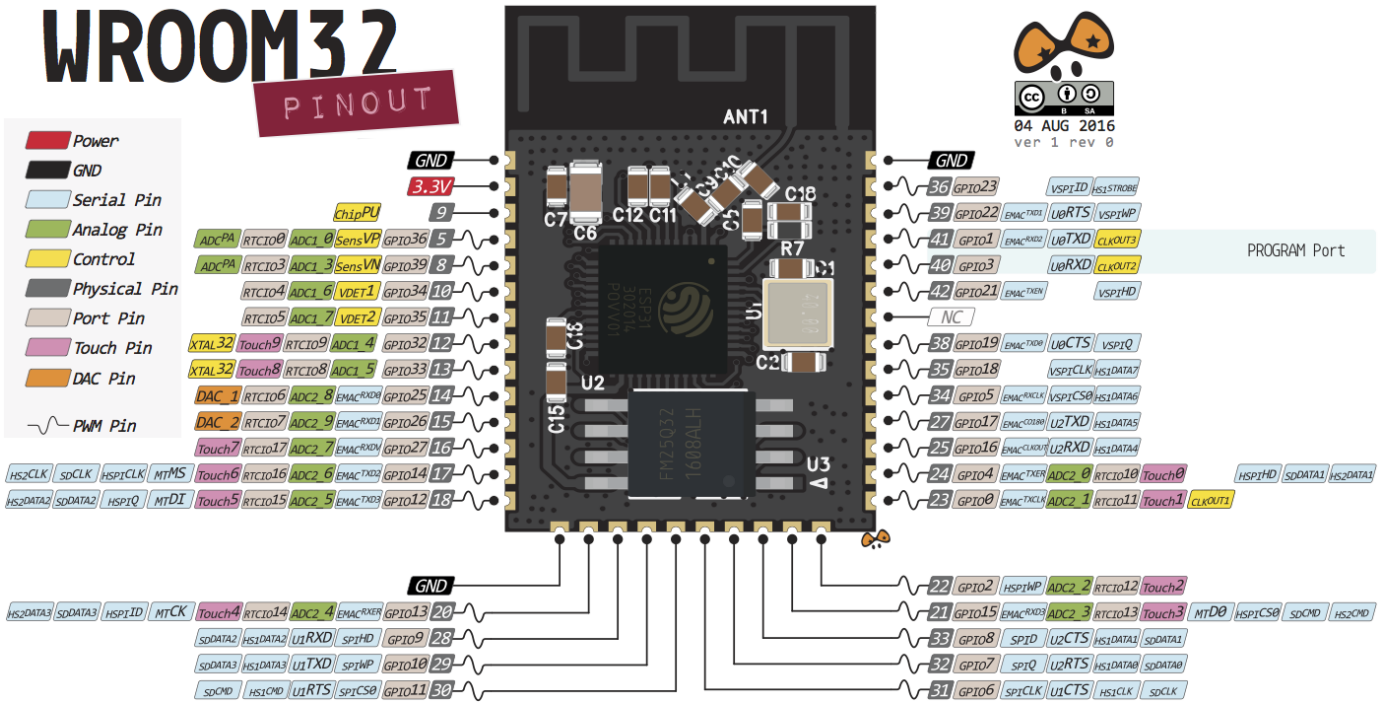
Con este método, puede ejecutar dos tareas diferentes de forma independiente y simultánea utilizando los dos núcleos.

Algunos ejemplos de esp32:

Cuando se utiliza el **ESP32** con el IDE de Arduino, se deben utilizar los **pines** por defecto del **ESP32** I2C (soportados por la biblioteca Wire): GPIO 21 (SDA) GPIO 22 (SCL)

Esp32 de 38 **Pines** totales. Modelo con gran cantidad de **Pines**, ideal para realizar proyectos IoT. Se puede programar facilmente con IDE de arduino.

La siguiente figura ilustra el pinout ESP-WROOM-32. Puede usarlo como referencia si está utilizando un chip desnudo ESP32 para construir una placa personalizada:

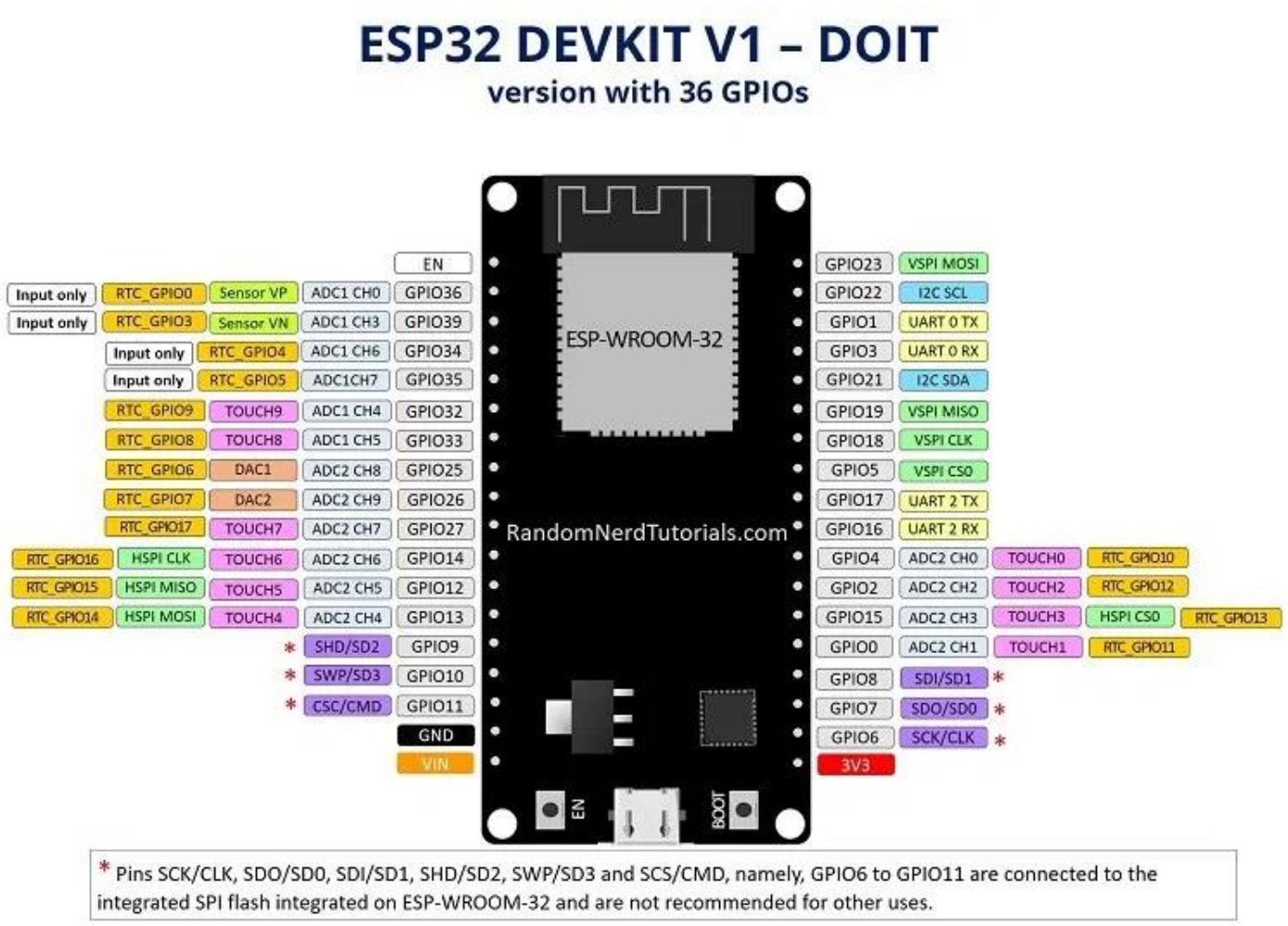


Los periféricos ESP32 incluyen:

- 18 canales de convertidor de analógico a digital (ADC)
- 3 interfaces SPI
- 3 interfaces UART
- 2 interfaces I2C
- 16 canales de salida PWM
- 2 convertidores de digital a analógico (DAC)
- 2 interfaces I2S
- 10 GPIO de detección capacitiva

Las características ADC (convertidor analógico a digital) y DAC (convertidor digital a analógico) se asignan a pines estáticos específicos. Sin embargo, puede decidir qué pines son UART, I2C, SPI, PWM, etc., solo necesita asignarlos en el código. Esto es posible gracias a la función de multiplexación del chip ESP32.

Aunque puede definir las propiedades de los pines en el software, hay pines asignados de forma predeterminada como se muestra en la siguiente figura (este es un ejemplo para la placa DOIT ESP32 DEVKIT V1 con 36 pines: la ubicación del pin puede cambiar según el fabricante).



Además, hay pines con características específicas que los hacen adecuados o no para un proyecto en particular. La siguiente tabla muestra qué pines son los mejores para usar como entradas, salidas y cuáles debe tener cuidado.

Entornos de programación

El ESP32 se puede programar en diferentes entornos de programación. Puede utilizar:

- Arduino IDE
- Espressif IDF (Marco de desarrollo de IoT)
- Micropython
- JavaScript
- LUA
- ...

Lenguajes

MicroPython es uno de los **lenguajes** para sistemas embebidos que ha empezado a ganar popularidad desde hace unos años gracias a su versatilidad y gran potencia. En lo particular, es muy útil para programar tarjetas de desarrollo basadas en **ESP32**, ya que permite obtener toda la capacidad de la tarjeta.

Debido a que el desarrollo de programas para ESP32 se hace comúnmente en la IDE de Arduino, consideramos que sería de gran relevancia comparar las ventajas que tiene empezar a adoptar la programación en MicroPython comparada con el lenguaje C++. Revisemos pues dichas ventajas:

MicroPython	Arduino
Se instala una sola vez, y para acceder al código de y hacer modificaciones sólo se accesa a un sistema de archivos	El proceso de compilado y enlace del programa se hace cada que se cambia el código, así como el proceso de flasheado
Se pueden agregar tantas librerías o scripts como uno desee. El número sólo está limitado por la memoria flash del dispositivo	La compilación del programa puede demorar más entre más librerías se incluyan
La ejecución del archivo principal main.py va después del archivo boot.py	El archivo de programa se compila a lenguaje máquina, lo que lo hace más eficiente, pero menos portable

Adoptar el uso de MicroPython puede vislumbrarse como el desarrollo de programas más portátiles y la homologación de los códigos, permitiendo a los desarrolladores aportar a la comunidad una cantidad más sustancial de código para nuevas librerías y módulos. Su uso en tarjetas basadas en ESP32 presenta una ventaja sustancial respecto a los programas desarrollados en la IDE de Arduino

Lenguajes de programación, marcos, plataformas y entornos utilizados para la programación ESP32:

- Arduino IDE con ESP32 Arduino Core
- MicroPython Una implementación ajustada de Python 3 para microcontroladores
- Marco de desarrollo de malla Espressif
- Espruino : SDK de JavaScript y firmware que emulan de cerca a Node.js
- Kit de herramientas de red Lua / IoT para ESP32-Wrover
- Mongoose OS : un sistema operativo para productos conectados en microcontroladores; programable con JavaScript o C. Una plataforma recomendada por Espressif Systems, AWS IoT, y Google Cloud IoT.
- mruby para el ESP32
- NodeMCU : firmware basado en Lua
- Zerynth : Python para IoT y microcontroladores, incluido el ESP32

Terminado y para tener en cuenta:

Antes de seleccionar una placa de desarrollo **ESP32** es necesario considerar ciertos aspectos:

- **Números de pines y configuración**: es importante tener acceso a la distribución de pines de la placa para poder hacer un uso correcto de ella.
- Interfaz **Serie-USB** y **regulador de voltaje**: estas dos características las tienen prácticamente todas las **placas de desarrollo**. Estas son las que permiten conectar la placa directamente al ordenador para ser energizada y programada.
- **Conector para batería**: si estás pensando en incursionar en sistemas de bajo consumo con baterías puedes optar por placas que ya incluyan conectores para baterías.
- **Funciones extras**: muchas placas de desarrollo para ESP32 traen características extras como **cámaras, pantallas OLED, módulos LoRa**, etc.

SoC ESP8266

Es un microcontrolador diseñado por Espressif Systems, una empresa china con sede en Shanghai. La producción en masa de estos microcontroladores comenzó hasta principios de 2014. El ESP8266 se ofrece como una solución WiFi independiente.

Si se fuera a usar un ESP8266 salido de fábrica probablemente no se sabría qué hacer, y es gracias a que los fabricantes los construyen encima de circuitos impresos y placas prefabricadas, que estos quedan listos para nuestro uso. Por eso se da lugar a varias versiones de ESP8266 (Ej: ESP-01, ESPO? ...) pero todas con el mismo procesador, lo que las diferencian son el número de pines GPIO expuestos, la cantidad de memoria flash, las dimensiones, la forma de exponer los pines, y otras consideraciones varias relativas a su construcción. Pero desde una perspectiva de programación todas son iguales. El ESP8266 es ideal para aplicaciones IOT por su bajo costo, sus características, sus variantes y aplicaciones.

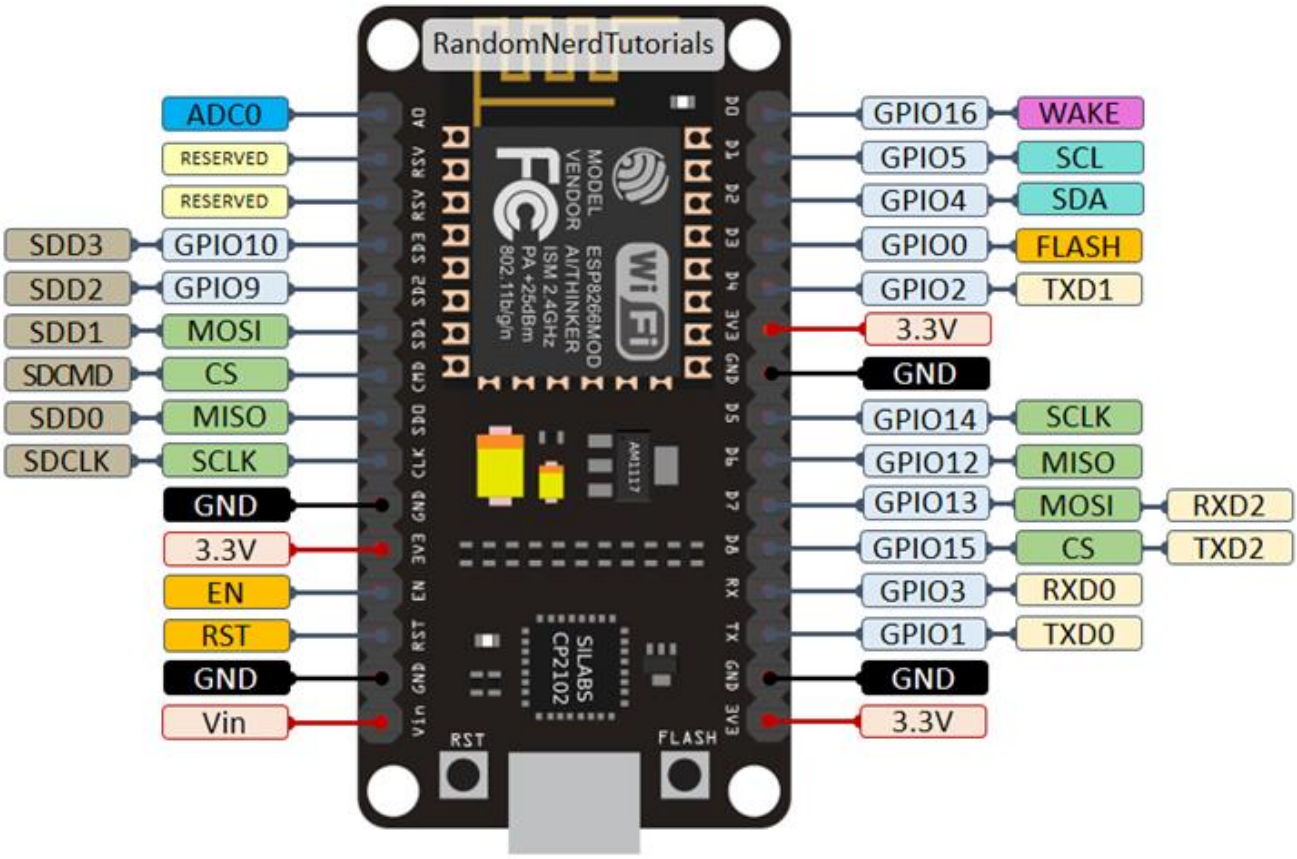
ESP8266 especificaciones:

- Protocolo 11 b/g/n
- Wi-Fi Direct (P2P), soft-AP (punto de acceso habilitado por software)
- Pila de protocolo TCP/IP integrado
- CPU integrado de 32 bits de bajo consumo
- SDIO 2.0, SPI, UART

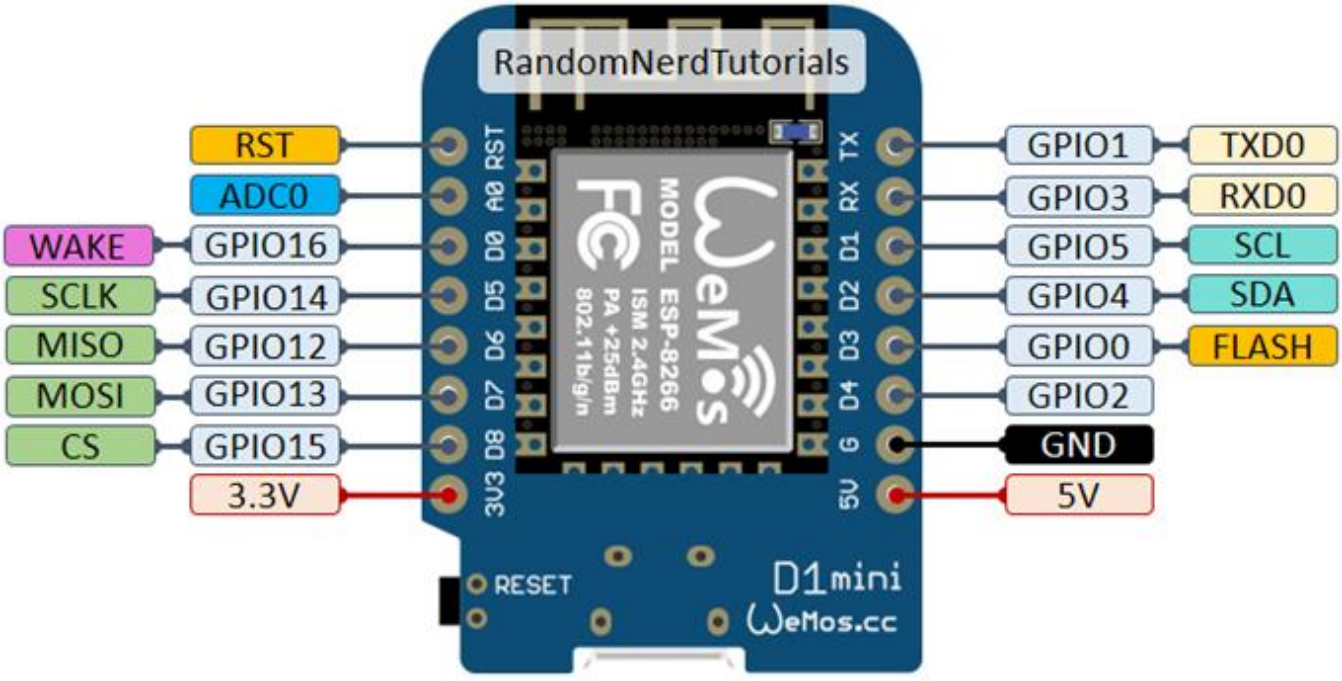
Variedad de versiones:

Distribución de PINES:

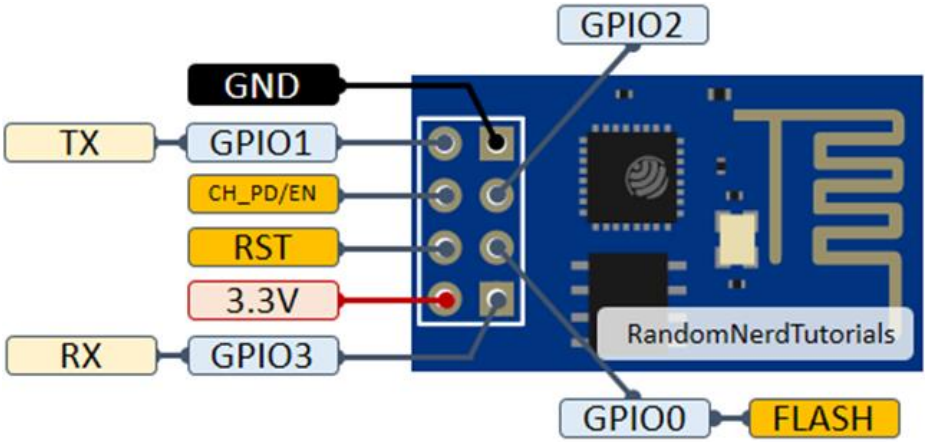
Asignación de pines del kit ESP-12E NodeMCU:



Pinout WeMos D1 Mini:



ESP8266-01 Pinout:



Label	GPIO	Input	Output	Notes
D0	GPIO16	no interrupt	no PWM or I2C support	HIGH at boot used to wake up from deep sleep
D1	GPIO5	OK	OK	often used as SCL (I2C)
D2	GPIO4	OK	OK	often used as SDA (I2C)
D3	GPIO0	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
D4	GPIO2	pulled up	OK	HIGH at boot connected to on-board LED, boot fails if pulled LOW
D5	GPIO14	OK	OK	SPI (SCLK)
D6	GPIO12	OK	OK	SPI (MISO)
D7	GPIO13	OK	OK	SPI (MOSI)
D8	GPIO15	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH
RX	GPIO3	OK	RX pin	HIGH at boot
TX	GPIO1	TX pin	OK	HIGH at boot debug output at boot, boot fails if pulled LOW
A0	ADC0	Analog Input	X	

Programación del ESP8266:

Hay varias formas de programar el ESP8266. A menudo usamos Arduino IDE o MicroPython.