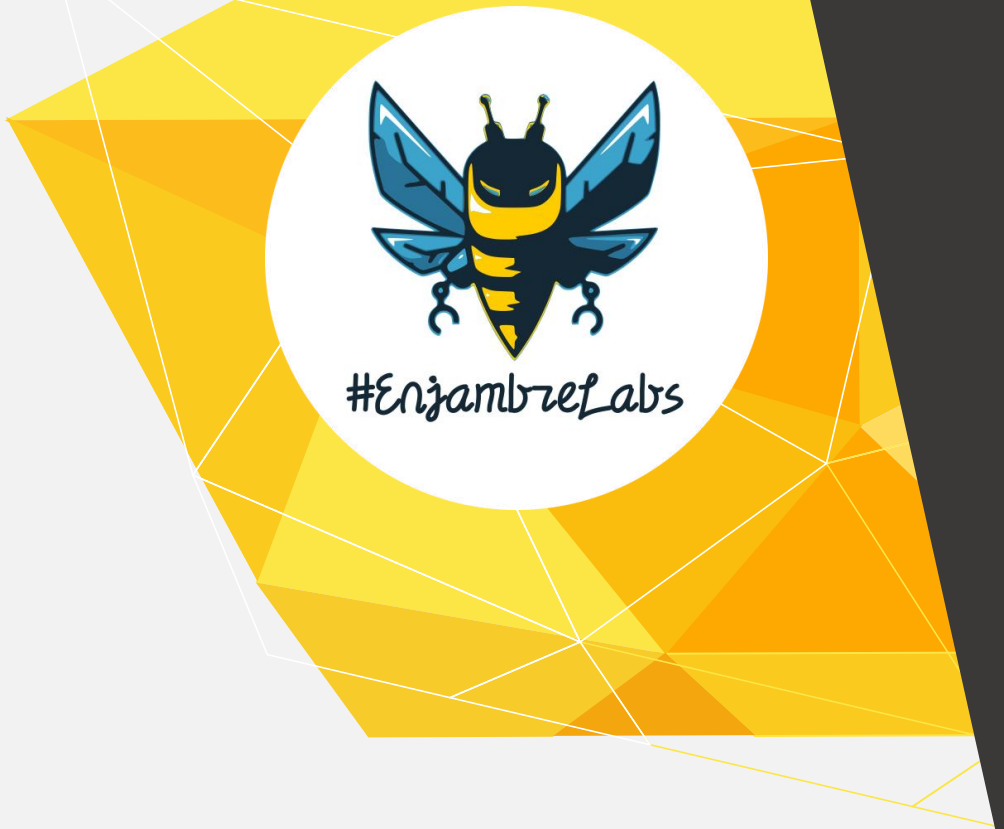


Juan Mite

@juanmisak

Amante del prototipado y de las hackatones, desarrollador de aplicaciones móviles, aplicaciones web y sistemas embebidos para la industria, ganador de varios premios locales y nacionales. Promotor de tecnologías libres tanto en hardware, software y aeromodelismo, obsesionado con el buen diseño, aficionado a la seguridad informática, inteligencia artificial y a la música.

Interfaces de comunicación

ESP32-µPython

PROTOCOLO

Es el que establece las reglas que gobernarán el intercambio de datos entre equipos.

INTERFACES

Preparan los datos para que puedan viajar por el medio de transmisión.

SPI

Del inglés Serial Peripheral Interface

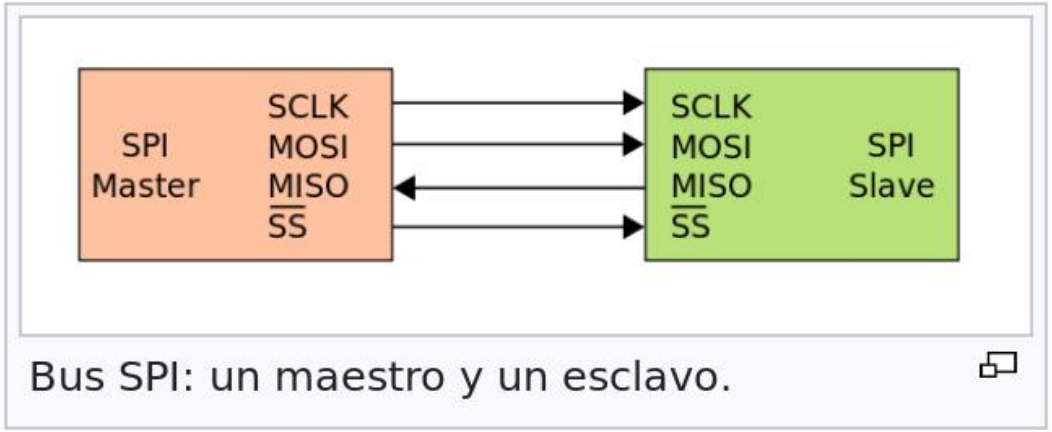
Ventajas

- Comunicación Full Duplex.

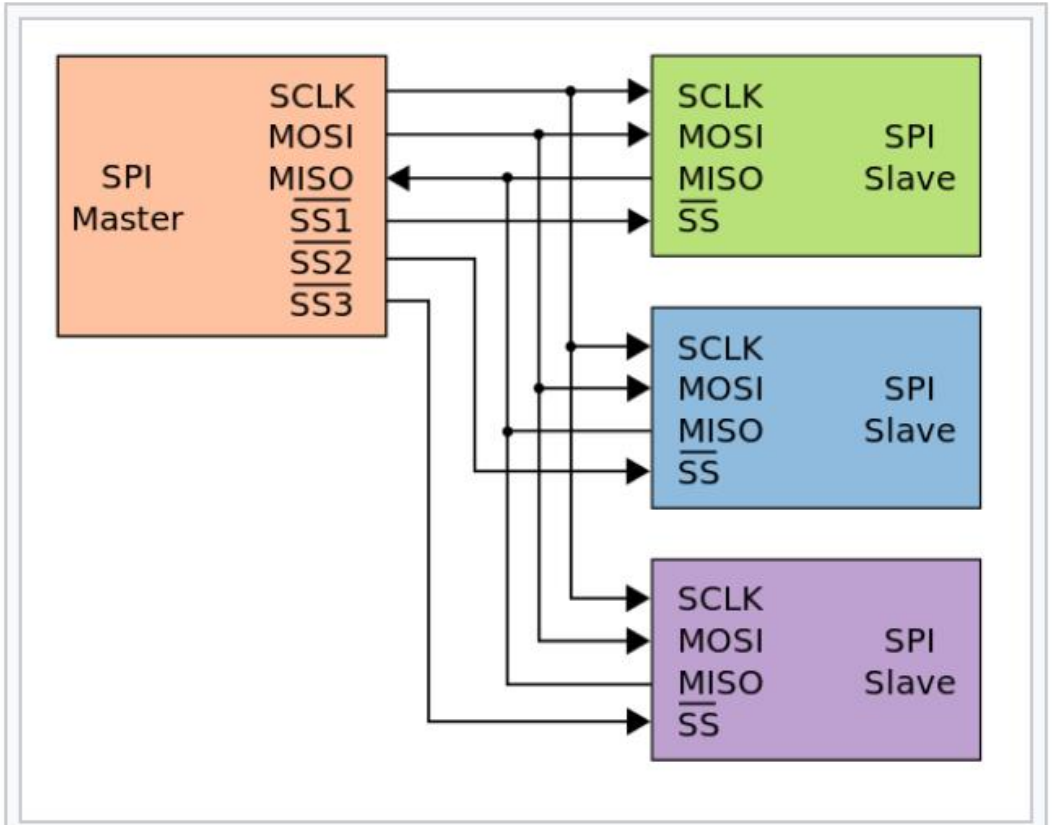
- Mayor velocidad de transmisión que con I²C o SMBus.

- Protocolo flexible en que se puede tener un control absoluto sobre los bits transmitidos.

- Los dispositivos clientes usan el reloj que envía el servidor, no necesitan por tanto su propio reloj.



Bus SPI: un maestro y un esclavo.



SPI bus: un maestro y tres esclavos.

SPI

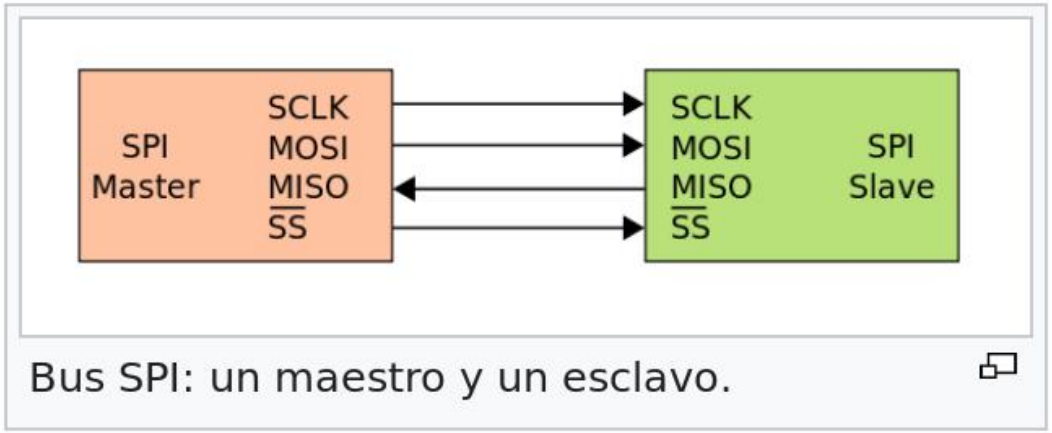
Del inglés Serial Peripheral Interface

Desventajas

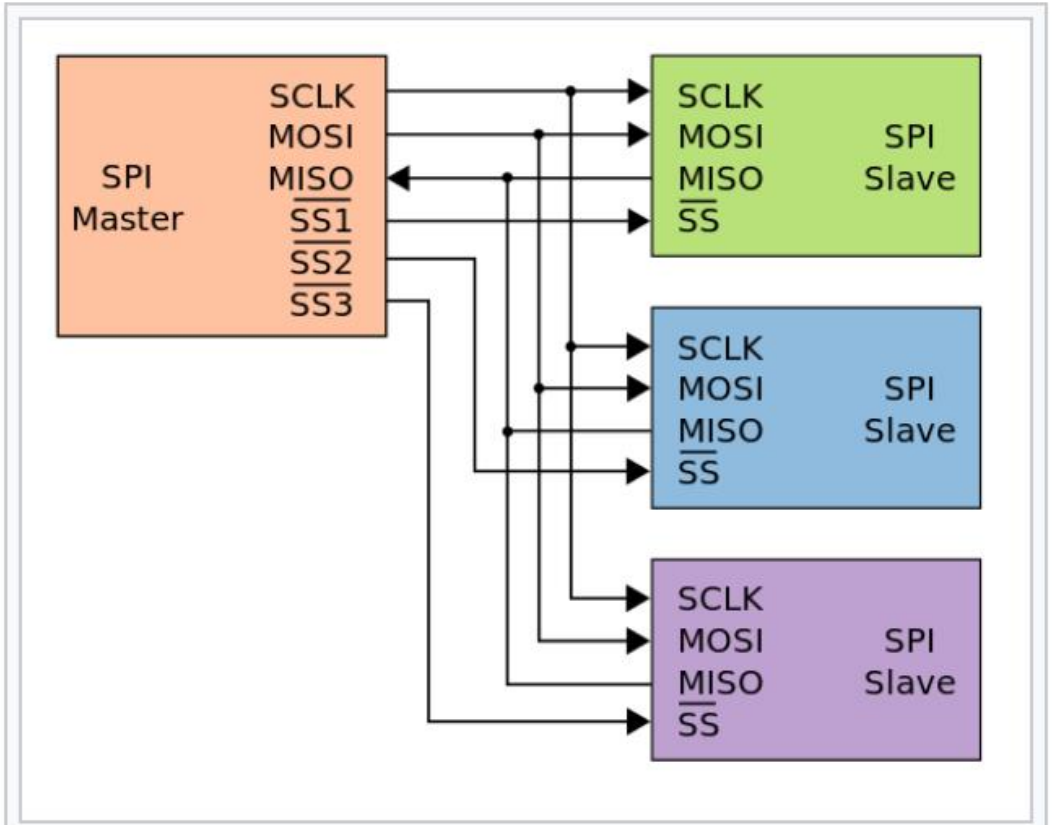
- No hay señal de asentimiento. El servidor podría estar enviando información sin que estuviese conectado ningún cliente y no se daría cuenta de nada.

- No permite fácilmente tener varios servidores conectados al bus.

- Sólo funciona en las distancias cortas.



Bus SPI: un maestro y un esclavo.



SPI bus: un maestro y tres esclavos.

I2S

Del inglés Integrated Interchip Sound

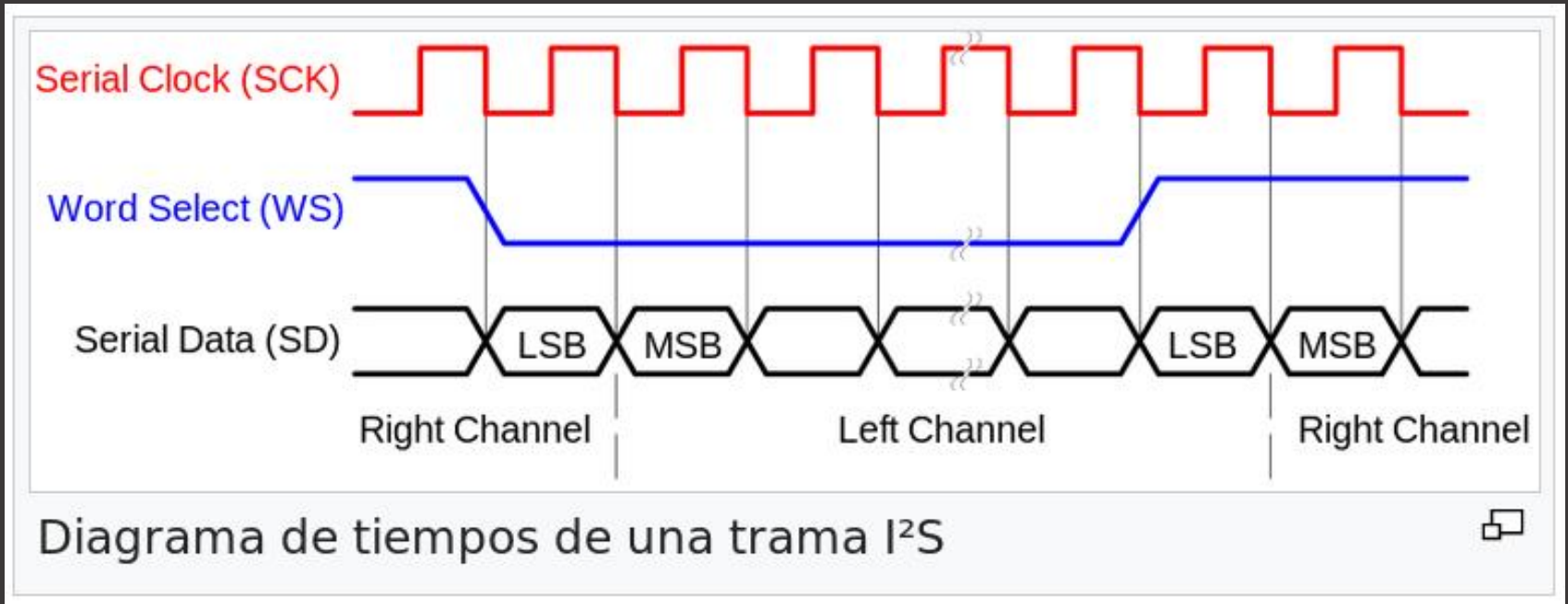
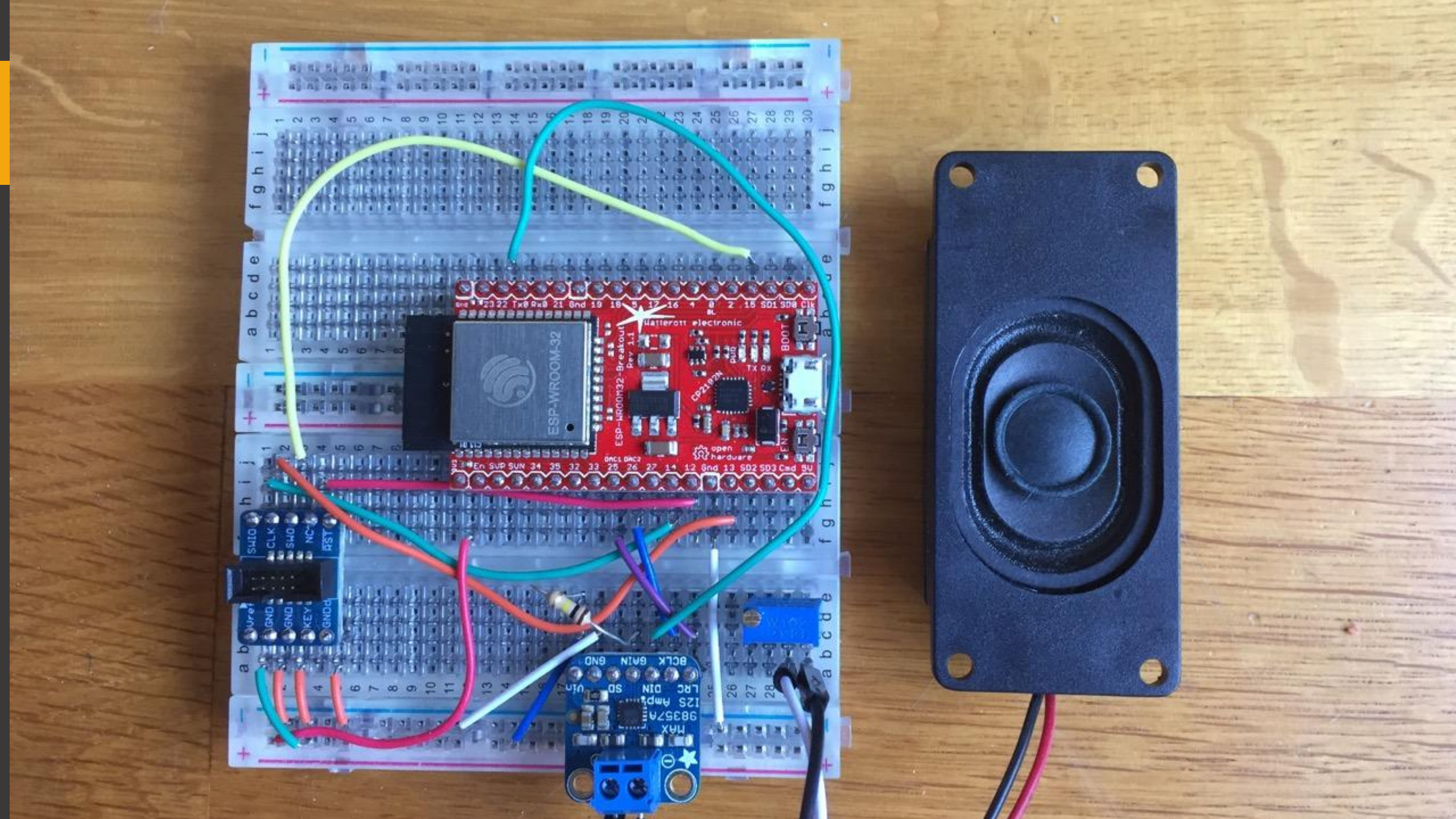


Diagrama de tiempos de una trama I²S

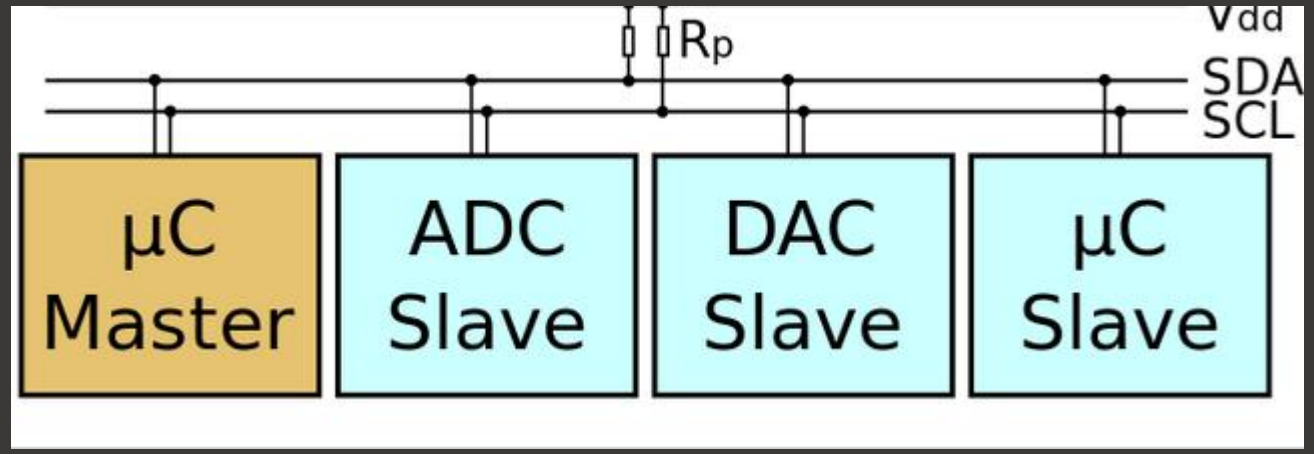

ESP-WROOM-32
ESP-WROOM32-Breakout
Rev 1.1
Mattairott electronic
open hardware
BOOT
TX RX
SWIO
CLK
SWO
NC
RST
GND
KEY
BCLK
GAIN
LRC
DIN
MAX
98357A
I2S Amp

GITHUB:

https://github.com/MrBuddyCasino/ESP32_MP3_Decoder

I2C

Inter-Integrated Circuit

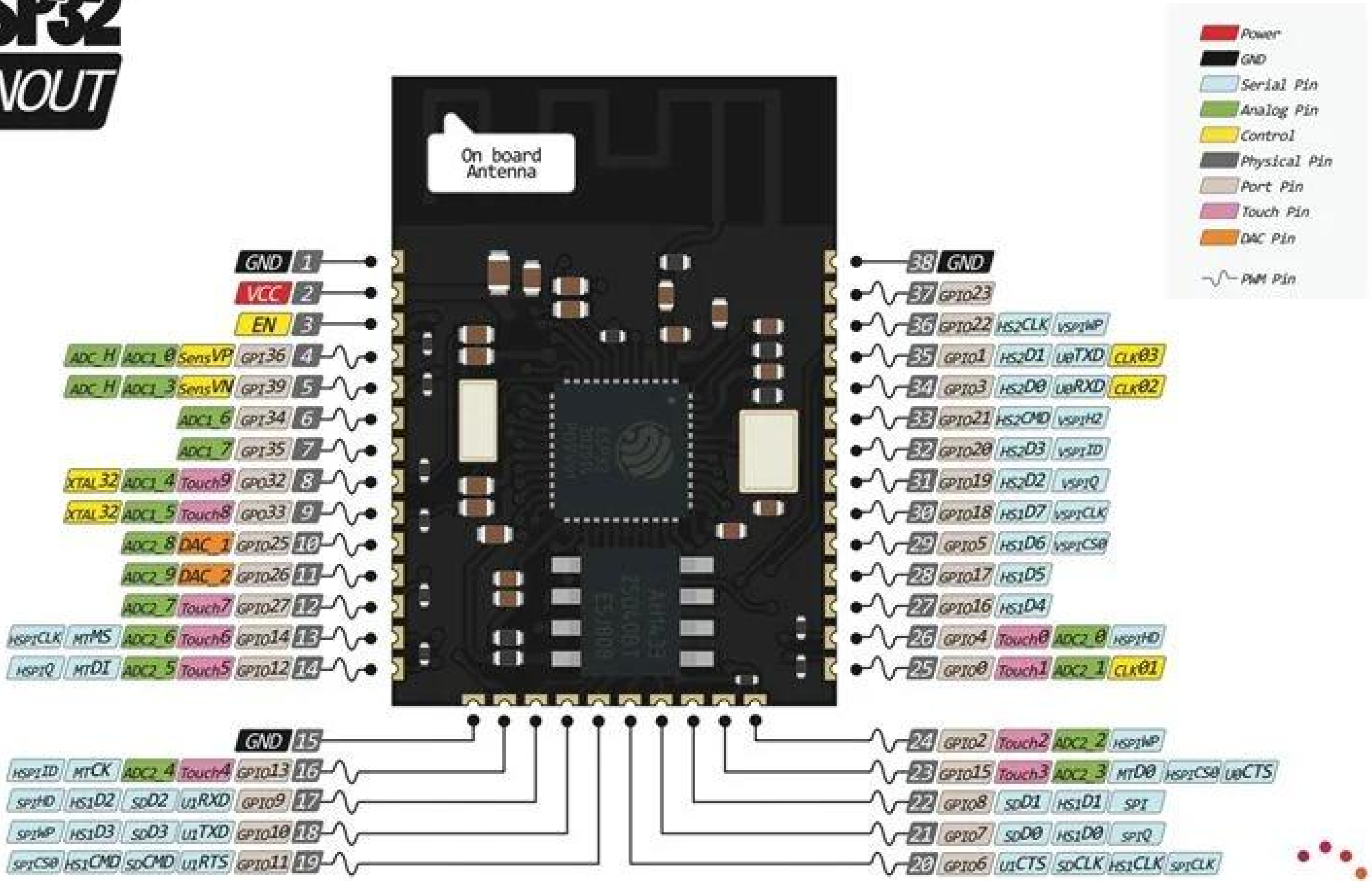


En el diagrama de la derecha se encuentran representados tres dispositivos. El I²C precisa de dos líneas de señal: reloj (CLK, Serial Clock) y la línea de datos (SDA, Serial Data)

I2C vs SPI

Permite multi master -	- No permite multi master
Comunicación half-duplex -	- Comunicación full-duplex
Usa 2 cables, para el reloj y para la trama -	- Usa 4 cables mosi, miso, SCL, SS
I2C es mas lento SPI -	- SPI es mas rápido que I2C.
I2C concume mas energía que SPI -	- SPIconcume menos energía que I2C.
I2C es menos suceptible al ruido que SPI -	- SPI es mas suceptible al ruido que I2C.
Se envía la dirección de destino -	- Se seleciona el dispositivo
I2C es mejor para distancias largas -	- I2C es mejor para distancias cortas.

ESP32 PINOUT



BLUETOOTH

REDES

ESP32 como cliente

```
def do_connect(wifi_ssid,wifi_passwd):          #Función útil para conectar con una red WiFi local
    import network, time
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print('\nConnecting to network', end='')
        wlan.connect(wifi_ssid, wifi_passwd)
        while not wlan.isconnected():
            print('.', end='')
            time.sleep(0.5)
            pass

    import ubinascii
    print()
    print("Interface's MAC: ", ubinascii.hexlify(network.WLAN().config('mac'),':').decode())  # Imprime la dirección MAC
    print("Interface's IP/netmask/gw/DNS: ", wlan.ifconfig(),"\n") # Imprime las direcciones IP/netmask/gw/DNS

do_connect("NOMBRE_DE_RED_WIFI","CLAVE_DE_RED_WIFI")
#Ejecuta la función - Se deben sustituir NOMBRE_DE_RED_WIFI y CLAVE_DE_RED_WIFI
#con los datos de la red WiFi local a la que queremos conectar el dispositivo
```

ESP32 como servidor

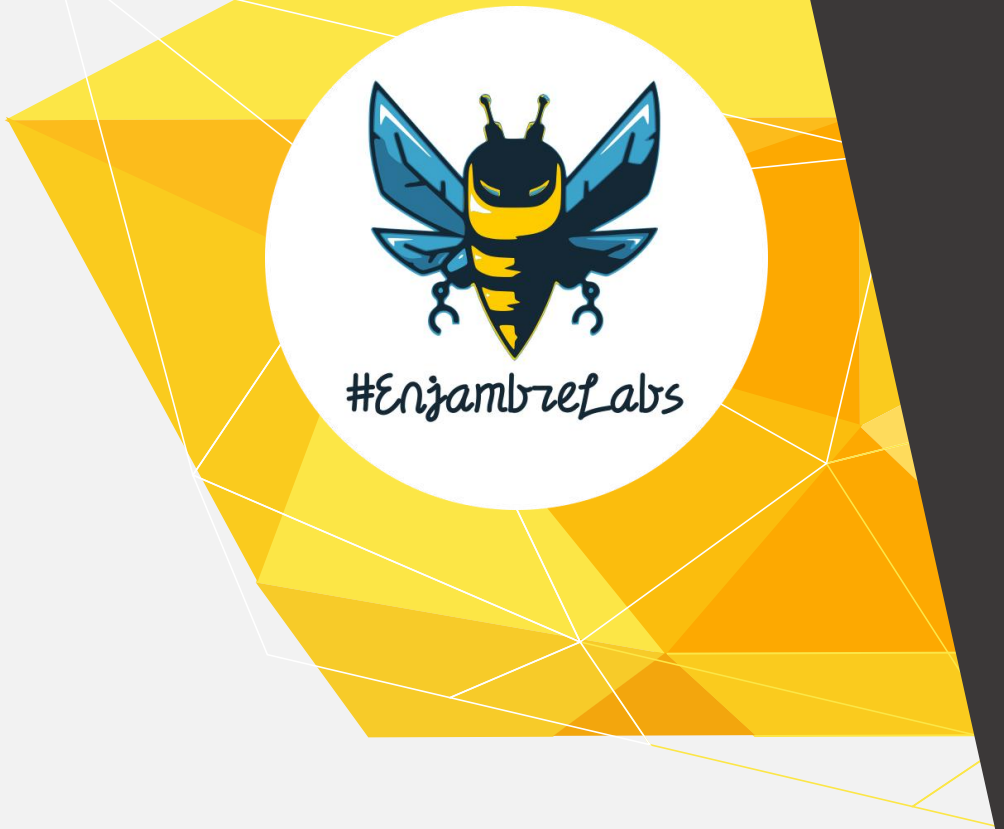
```
def do_connect(wifi_ssid,wifi_passwd):          #Función útil para conectar con una red WiFi local
    import network, time
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print('\nConnecting to network', end='')
        wlan.connect(wifi_ssid, wifi_passwd)
        while not wlan.isconnected():
            print('.', end='')
            time.sleep(0.5)
            pass

    import ubinascii
    print()
    print("Interface's MAC: ", ubinascii.hexlify(network.WLAN().config('mac'),':').decode())  # Imprime la dirección MAC
    print("Interface's IP/netmask/gw/DNS: ", wlan.ifconfig(),"\n") # Imprime las direcciones IP/netmask/gw/DNS

do_connect("NOMBRE_DE_RED_WIFI","CLAVE_DE_RED_WIFI")
#Ejecuta la función - Se deben sustituir NOMBRE_DE_RED_WIFI y CLAVE_DE_RED_WIFI
#con los datos de la red WiFi local a la que queremos conectar el dispositivo
```

MQTT

Protocolo por excelencia para IoT



Juan Mite

@juanmisak

Amante del prototipado y de las hackatones, desarrollador de aplicaciones móviles, aplicaciones web y sistemas embebidos para la industria, ganador de varios premios locales y nacionales. Promotor de tecnologías libres tanto en hardware, software y aeromodelismo, obsesionado con el buen diseño, aficionado a la seguridad informática, inteligencia artificial y a la música.