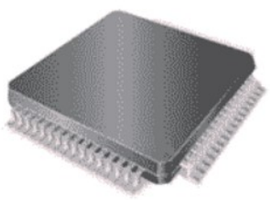
ESP8266/ESP32 LUIS LLAMAS *(VERSION 25-5-19)*

DETALLES DE HARDWARE DEL ESP8266 Y DIFERENCIAS CON ARDUINO

15 MAYO, 2018



Continuamos con el SoC ESP8266 a comentar algunos de los **detalles de hardware del ESP8266 y del ESP12E** (en la entrada anterior vimos su pinout y que son esencialmente equivalentes), y sus diferencias respecto a una placa Arduino.

Lo primero a destacar es que **el ESP8266 carece de muchas funciones por hardware que incluyen los procesadores Atmel**. Unidos a la gestión del Wifi, el SoC tiene que dividir sus recursos entre todas estas funciones. Aunque es más rápido que un Arduino, es una sobrecarga de trabajo, lo cuál en algunas aplicaciones puede suponer un problema.

Lo siguiente a destacar es que, como vimos en la entrada de presentación, **el ESP8266 carece de memoria Flash interna**. La memoria donde almacenamos el programa se incluye en el módulo, pero fuera del SoC, y se comunican entre si por SPI. Eso significa que hay módulos con distinta cantidad de memoria, y que no podremos usar los pines GPIO que comunican con la memoria (lo veremos a continuación).

ALIMENTACIÓN

La alimentación del ESP8266 es a 3.3V, alimentar un voltaje superior a 3.6V destruirá el SoC. Aunque muchas placas de desarrollo incluyen reguladores de voltaje para poder alimentar la placa a 5V, o desde el USB.

Respecto a si el resto de pines del ESP8266 son tolerantes a 5V, ha habido mucho debate. Según el Datasheet (que cambiaba según versiones) no quedaba muy claro, pero se entendía que no lo eran. Sin embargo, en la actualidad, declaraciones de Espressif y experimentos de usuarios nos permiten decir que **los pines GPIO del ESP8266 sí son tolerantes a 5V cuando funcionan como entrada digital.**.

La corriente máxima que pueden proporcionar o absorber los pines digitales es de 12mA. En comparación, la mayoría de modelos de Arduino pueden suministrar 20-40mA.

Respecto al **conversor analógico digital (ADC) la tensión máxima que pueden registrar es de 0-1V**. Suministrar más de 1V al ADC lo dañará. No obstante, algunas placas tienen de desarrollo incluyen divisores para ampliar el rango de 0-3.3V. Tendréis que comprobar en cada placa el rango de medición del ADC.

El pin CHIP_EN controla el encendido o apagado del SoC, estando encendido cuando está en HIGH. El pin EXT_RSTB controla el Reset del ESP8266, activándose cuando está en LOW.

ARRANQUE (BOOT)

El ESP8266 tiene 3 modos de arranque:

- UART Bootloader, para subir un programa por UART a la memoria flash.
- Boot Sketch, ejecuta el último programa subido en la memoria flash.
- SDIO, que no se usa con al programar con Arduino.

Para arrancar en un modo u otro se deben configurar adecuadamente los pines GPIO15, GPIO0 y GPIO2 según la siguiente tabla:

Modo	GPIO15	GPIO0	GPIO2
Uart Bootloader	0V	0V	3.3V
Boot normal	0V	3.3V	3.3V
SDIO	3.3V	x	x

En la mayoría de las placas de desarrollo habrá sistemas que gestionarán el estado de estos pines por nosotros. No obstante, tanto si queremos usar el ESP8266 (o el ESP12E) de forma independiente, como en el caso de una placa de desarrollo, debemos tener en cuenta lo siguiente:

- El GPIO15 está siempre pulled down, así que no podemos usar su resistencia Pull-Up.
- El GPIO0 está puesto a HIGH durante el funcionamiento.
- EL GPIO2 no puede estar puesto a LOW durante el arranque.

ENTRADAS Y SALIDAS DIGITALES (GPIO)

El ESP8266 tiene 17 pines I/O (GPIO, General Purpose Input/Output pins). Estos pueden actuar de salida proporcionando tensión de 0V o 3.3V (LOW y HIGH). Actuando como entrada pueden reconocer una tensión de 0V o 3.3V suministrada al GPIO.

Cuando actúan como salida la corriente máxima que puede proporcionar (o absorber) cada GPIO es de 12mA.

Como entrada, como hemos dicho anteriormente en la actualidad podemos decir que son tolerantes a 5V (aunque no me responsabilizo de que reventéis vuestra placa), por lo que emplear una entrada para medir un voltaje de hasta 5.8V no debería dañarla.

De los 17 GPIO (0 a 16) **no podemos usarlos todos**. De hecho podemos usar bastante pocos, lo cuál es otra limitación importante respecto a Arduino.

- 6 GPIO (GPIO6 a GPIO11) son usados para conectar por SPI con la memoria flash, por lo que no podemos usarlos.
- Los GPIO0, GPIO2 y GIPO15 intervienen en el arranque, con las consideraciones que hemos visto en el apartado anterior.
- Los GPIO1 y GPIO3 son usados para comunicación Serial (UART)

En cuanto a resistencias internas de Pull, igual que Arduino los pines GPIO0 a GPIO15 tienen resistencias de Pull-Up. Mientras que el GPIO16 tiene una resistencia de Pull-Down.

PWM (SALIDAS ANALÓGICAS)

A diferencia de Arduino **el ESP8266 no tiene PWM por hardware**, en su lugar tiene que emularlo por software. Esto tiene la ventaja de que podemos usar PWM en todos los GPIO, pero también que supone una carga de cálculo para el ESP8266. La frecuencia por defecto es de 1kH, pero puede ser cambiada.

ENTRADAS ANALÓGICAS (ADC)

El ESP8266 tiene un ADC de 10bits de resolución. El ADC tiene su propio pin, independiente de los GPIO. El rango de entrada del ADC es de 0-1V, intentar medir una tensión superior a 1V dañará el ADC. No obstante, ya hemos visto que muchas placas tienen divisores que amplían el rango para poder medir de 0-3.3V.

COMUNICACIÓN

SERIAL (UART)

El ESP8266 tiene 2 UARTs por hardware:

- UART0 en pines 1 y 3 (TX0 y RX0).
- UART1 en pines 2 y 8 (TX1 y RX1).

Sin embargo, el pin 8 es empleado para conectar con la memoria flash por lo que, en la práctica, el puerto UART1 sólo puede emplear el pin TX1 (sólo puede enviar, no recibir). Por otro lado, el UART0 también tiene acceso desde los pines 15 y 13 (RTS0 y CTS0).

I2C

El ESP8266 **no tiene hardware para I2C** por lo cual tiene que emularlo por software. Eso significa que podemos emplear I2C con casi cualquier pin GPIO, pero nuevamente supone una carga para el procesador. Por defecto, la librería usa el GPIO4 y GPIO5 (SDA y SCL). La velocidad máxima es de 450kHz.

SPI

El ESP8266 tiene un 2 SPI por hardware, pero uno es utilizado para conectar con la memoria flash. Por tanto, queda **el ESP8266 tiene 1 SPI disponible** (HSPI) que puede actuar tanto como Master como Slave. Los pines usados son el GPIO14 (CLK), GPIO12 (MISO), GPIO13 (MOSI) y GPIO15 (SS).

RESUMEN DE LOS PINES

Aquí tenemos un resumen en una tabla para su consulta:

GPIO	Función	Estado	Condiciones
0	Boot	3.3V	No entrada
1	TX0	–	No usable con UART
2	Boot y TX1	3.3V (solo en boot)	No a GND al arrancar

GPIO	Función	Estado	Condiciones
3	RX0	–	No usable con UART
4	SDA (I²C)	–	–
5	SCL (I²C)	–	–
6 – 11	Memoria Flash	x	No usables
12	MISO (SPI)	–	–
13	MOSI (SPI)	–	–
14	SCK (SPI)	–	–
15	SS (SPI)	0V	No Pull-Up
16	Wake up desde sleep	–	Resistencia Pull-Down Conectar a RST para Wake-Up

En las próximas entradas iremos viendo distintas placas de placas de desarrollo como laNodeMCU que integran el ESP8266 y empezaremos su programación.