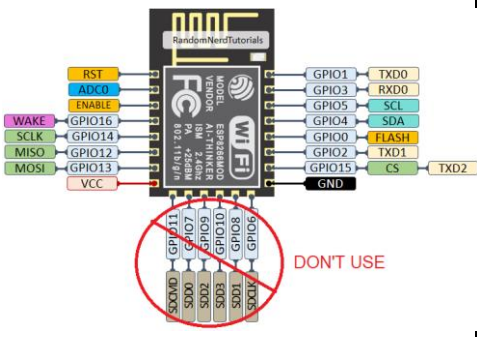
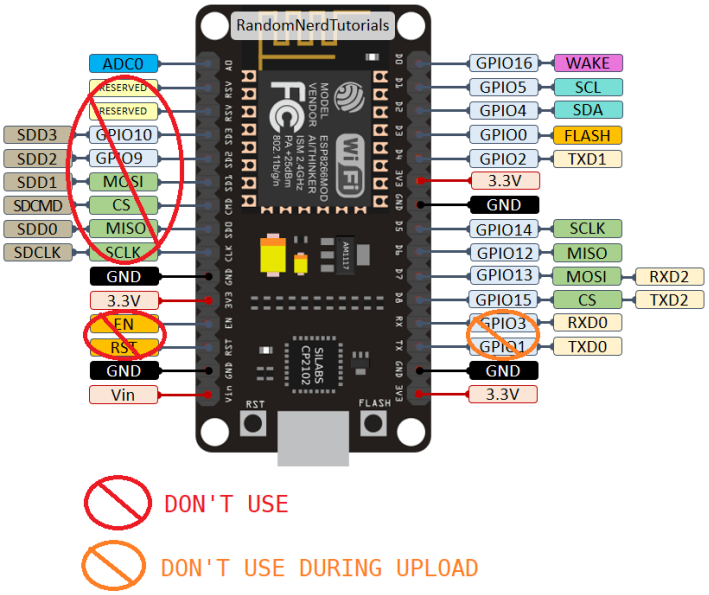
Fiche ESP8266

ESP8266 12-E Chip Pinout



ESP8266 12-E NodeMCU Kit



Périphériques de l'ESP8266 :

- 17 GPIOs GPIO (9 utilisables avec des restrictions)
- SPI SCLK MISO CS MOSI
- I2C (implémentation software) SDA SCL
- UART TXD0 RXD0 TXD2 RXD2 TXD1
- 10-bit ADC ADC0

Recommandation des Pins à utiliser

Une chose importante à comprendre est que le numéro du GPIO ne correspond pas au numéro du label. Par exemple le label D0 correspond au GPIO16.

- Les pins notées " OK " sont utilisables.
- Les pins notées " ! OK ! " sont utilisables avec précaution.

Label		GPIO	Input	Output	Notes
LED_BUILTIN	D0	GPIO16	! OK ! No `IT`	! OK ! No PWM, No I2C	HIGH at boot WAKE (from deep sleep)
	D1	GPIO5	OK	OK	often used as SCL (I2C)
	D2	GPIO4	OK	OK	often used as SDA (I2C)
BTN_FLASH	D3	GPIO0	! OK ! pulled up(10kΩ)	OK	connected to FLASH btn, **boot fail if pulled LOW**
LED_ANT	D4	GPIO2	! OK ! pulled up(10kΩ)	OK	HIGH at boot, **boot fail if pulled LOW**
	D5	GPIO14	OK	OK	(SPI) SCLK
	D6	GPIO12	OK	OK	(SPI) MISO
	D7	GPIO13	OK	OK	(SPI) MOSI RXD2
	D8	GPIO15	! OK ! pulled Down(10kΩ)	OK	(SPI) CS TXD2 **Boot fail if pulled HIGH**
RX	D9	GPIO3	! OK ! Not recommended	RX pin	HIGH at boot
TX	D10	GPIO1	TX pin	! OK ! Not recommended	HIGH at boot, **boot fail if pulled LOW**
	A0	ADC0	Analog Input		10 bits DAC 0V to 3.3V on DEV-Kit (0 ~ 1023)

Comportement des pins durant le démarrage

Pendant la phase de démarrage, l'ESP8266 génère des niveaux haut ou bas sur certaines pins pouvant produire des comportements erratiques des périphériques branchés sur ce dernier (claquement de relais, allumage de LEDs, …), voir https://rabbithole.wwwdotorg.org/2017/03/28/esp8266-gpio.html pour plus de détail.

- Pins au niveau HAUT durant le boot : GPIO16, GPIO14, GPIO12, GPIO13
- Pins avec alternance de niveaux logiques : GPIO0, GPIO2, GPIO3, GPIO1
- Pins au niveau BAS durant le boot : GPIO5, GPIO4, GPIO15

Les pins les plus sécurisées pour utiliser des relais sont les **GPIO5** et **GPIO4**.

Deep Sleep (GPIO16)

Le GPIO16 peut être utilisé pour réveiller l'ESP8266 depuis le mode "Deep Sleep". Pour cela, il faut connecter le GPIO16 à la pin RST. Dans le code arduino, on utilisera l'instruction `ESP.deepSleep(uS)` en passant en argument le temps de sommeil en µs, cette instruction passera le microcontrôleur en sommeil, enverra un signal par le GPIO16 sur RST réveillant l'ESP8266. Plus de détails sur l'utilisation https://randomnerdtutorials.com/esp8266-deep-sleep-with-arduino-ide/ .

Liaison I2C

Une liaison I2C est une liaison série maître/esclave(s) composée de 2 fils SDA (Datas) et SCL (Horloge). L'ESP8266 ne dispose pas de liaison I2C hardware. On peut néanmoins implémenter une liaison I2C software sur n'importe quelle GPIO (sauf GPIO16). Dans beaucoup d'exemples et de librairie, on utilise les GPIOs suivants :

- **GPIO4 : SDA**
- **GPIO5 : SCL**

Exemple code Arduino (master_reader.ino) :

```
  #include <Wire.h>
  #define SDA_PIN 4
  #define SCL_PIN 5
  const int16_t I2C_MASTER = 0x42; //adress I2C
...
  Wire.begin(SDA_PIN, SCL_PIN, I2C_MASTER);// (add option for master)
```

Liaison SPI

Une liaison SPI est une liaison série maître/esclave composée de 3 (+1 fil par esclave) fils SCLK (Horloge), MISO (Master In Slave Out), MOSI (Master Output Slave Input) et CS (Chip Select, 1 CS par esclave). L'ESP8266 comprend une liaison SPI hardware utilisable sur les pins :

- **GPIO12: MISO**
- **GPIO13: MOSI**
- **GPIO14: SCLK**
- **GPIO15: CS**

SPISlave > SPISlave_Master

```
#include <SPI.h>
SPI.begin();
SPI.……
```

SPISlave > SPISlave_Test

```
#include <SPISlave.h>
SPISlave.begin();
SPISlave.……
```

PWM 10 bits (tous les GPIO sauf GPIO16)

- **Tous les GPIO sauf GPIO16**

Exemple code Arduino :

```
analogWriteFreq(new_frequency); //def: 1kHz, Value in Hz
analogWrite(thisPin, value);    // value 0-1023
```

Interruption

- **Tous les GPIO sauf GPIO16**

Exemple code Arduino :

```
ICACHE_RAM_ATTR void MonIT() {  // fonction d'IT
  ...
attachInterrupt(digitalPinToInterrupt(GPIO), MonIT, RISING); //FALLING, CHANGE
```