CNAM 2020
Initiation Arduino sur ESP8266

Documentation :
Le site de référence pour les fonctions et les classes de l'environnement Arduino
https://www.arduino.cc/reference/en
Le site de référence pour les spécificités du portage Arduino sur l'ESP8266
https://arduino-esp8266.readthedocs.io

1. Principe de programmation

L'ESP8266 est un *microcontrôleur*. C'est à dire un système complet (microprocesseur, mémoires, périphériques) minimaliste qui permet de supporter une application complète. Les ressources sont limités par rapport à un ordinateur de bureau mais le prix et la consommation sont très faibles.
La présence du WiFi et de 4Mo de mémoire Flash le rend particulièrement adapté aux applications IOT.
Caractéristiques principales :
- CPU 32 bits cadencé à 80 MHz
- 112ko de RAM
- jusqu'à 16Mo de Flash (4Mo sur l'ESP12E)
- 16 entrées/sorties (GPIO) dont une analogique (10bits)
- Wi-Fi IEEE 802.11 b/g/n authentification WEP/WPA/WPA2

Ce module peut se programmer de différentes manières :

- En C/C++ avec le SDK fourni par la société ESPRESSIF : complexe mais permet d'exploiter le maximum des possibilités.

- En C/C++ dans l'environnement Arduino : beaucoup plus simple pour démarrer qu'avec le SDK. Présence de nombreux exemples et librairies en open source. Possibilité d'appeler les fonctions natives du SDK si nécessaire.

- En Python avec le projet MicroPython : simplicité et puissance du langage Python, performance légèrement réduite. Manque encore de nombreuses librairies.

Nous allons utiliser l'environnement Arduino.
Le code se présente toujours sous la forme d'une fonction `setup()` qui est exécutée une seule fois au lancement de l'application. Puis d'une fonction `loop()` qui est appelée dans une boucle infinie.

```
void setup() {

      }
void loop() {

 }
```

Dans le setup on effectue l'initialisation de l'application.
Dans la boucle infinie on va exécuter en séquence toutes les fonctionnalités de l'application[1].
Par exemple une application de régulation de température qui a un serveur Web, un afficheur lcd, un module d'archivage de la température :

```
void loop() {
            lireTemperature() ;
            regulationTemperature() ;
            archivageTemperature() ;
            serveurWeb() ;
            affichageLCD() ;
      }
```

Il faut bien comprendre que chaque fonction ne s'exécute que lorsque la précédente est terminée (multitâche coopératif élémentaire). Le temps de réaction est égal au temps de parcours de la boucle. Un fonction lente retarde toutes les autres.

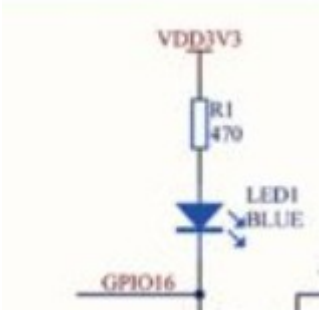
2. Entrées et sorties logiques

GPIO : general purpose input output.

La plupart des broches sont utilisables comme entrée ou sortie (broches dites bidirectionnelles) mais sans matériel supplémentaire sur la carte ESP12E seulement 3 broches sont utilisables pour cette initiation.
En sortie nous pouvons utiliser les broches D0 et D4 sur lesquels sont câblés deux leds bleues.
En entrée nous avons la broche D3 sur laquelle est câblée le bouton FLASH.

2.1 Sortie

Un sortie est un générateur de tension (ici 0V / 3.3V) limité en courant (qq mA) qui est commandable par programme. État 0 → 0V / État 1 → 3.3V
Les leds bleues sont câblée selon le schéma ci-dessous :



Ainsi la LED s'allumera lorsque la sortie vaudra 0V et s'éteindra lorsque la sortie vaudra 3.3V. Logique dite inversée.

Au Reset, les broches bidirectionnelles sont toujours en entrée. Il faut une action volontaire pour les configurer en sorties. C'est le rôle de la fonction pinMode() :

```
#define LED D0
#define LED_ANT D4

void setup() {
 pinMode(LED,OUTPUT); // met la broche LED en sortie
```

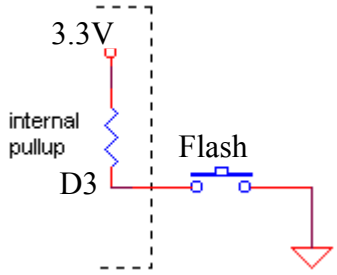
1 La gestion du WiFi est réalisée de manière quasi transparente pour le programmeur dans l'environnement Arduino. En fait la fonction loop() est une tâche d'un OS temps réel.

```
 digitalWrite(LED,0); // allume la LED
}

void loop() {

}
```

Faite de même avec la LED LED_ANT.
Faite clignoter les deux leds (voir exemple Blink pour l'instant utilisez la fonction delay())

2.2 Entrée

Une entrée logique est une broche dont on peut connaître l'état logique par programme.
0V → État logique 0 , 3.3V → État logique 1
Au Reset les broches bidirectionnelles sont déjà en entrée. L'initialisation n'est pas nécessaire.
pinMode() permet de revenir à une configuration en entrée : `pinMode(LED,INPUT);`
Une entrée ne doit jamais être laissé en l'air (non branchée) car sinon on risque de lire une valeur aberrante. Souvent on utilise un résistance interne au microcontrôleur qui fixe le potentiel de l'entrée à 1. C'est le mode pullup : `pinMode(LED,INPUT_PULLUP);`
Le bouton FLASH est câblé de la manière suivante :



Si on appuie sur le bouton, l'entrée D3 est à 0 et quand on relâche elle passe à 1. Encore une logique inversée.
Ainsi le programme ci-dessous permet d'allumer la led lors de l'appuie sur le bouton.

```
#define LED D0
#define LED_ANT D4
#define BTN D3

void setup() {
 pinMode(LED,OUTPUT);
 pinMode(BTN,INPUT_PULLUP);
 digitalWrite(LED,1);
}

void loop() {
      int b = digitalRead(BTN);
      if (b==0) {
            digitalWrite(LED,0);
      }
      else {
            digitalWrite(LED,1);
      }
}
```

On aurait pu simplement écrire : `digitalWrite(LED,!digitalRead(BTN));`

Programmer le fonctionnement suivant :
  bouton enfoncé : LED_ANT allumée et LED éteinte.
  bouton relâché : LED_ANT éteint et LED allumée.

3. Gestion du temps

On a souvent besoin de cadencer des actions. Cela peut se faire facilement avec les fonctions `delay()` et `delayMicroseconds()` qui retardent l'exécution d'un nombre de millisecondes ou de microsecondes passé en paramètre. Voir par exemple le programme exemple Blink.
Mais ces fonctions bloquent la suite du programme. Il ne faut donc pas les utiliser sauf pour des retards très faibles devant le temps de réaction souhaité (quelques millisecondes au maximum).
Un autre mécanisme est proposé par la fonction `millis()` qui donne le nombre de millisecondes écoulées depuis le Reset. Pour exécuter une action à intervalle régulier, on va stocker la valeur de millis() dans une variable et "attendre" en relisant millis() en permanence que la durée souhaitée soit écoulé.
Cette méthode est implémentée dans l'exemple `02.Digital → BlinkWithoutDelay` .

Bien comprendre cette exemple et en vous en inspirant réalisez le fonctionnement suivant :

La LED clignote avec une période de 2 secondes
La LED_ANT est éteinte pendant 3 secondes et allumées pendant 500ms.
Dans le terminal l'état relâché/appuyé du bouton est affiché chaque seconde.
Si on appuie sur le bouton la période d'allumage de la LED et de 400ms (au lieu de 2 secondes).