Formation IoT

PAR ALAIN CARIOU, MAI 2019

I – Les protocoles de communications

Le protocole Wi-Fi

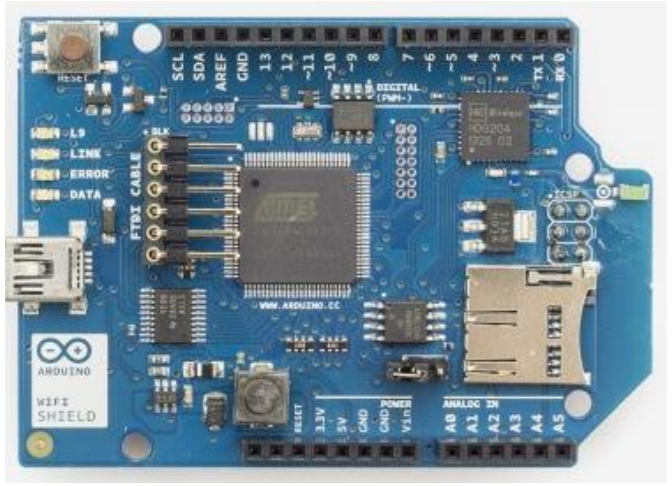
- Le protocole de communication sans fil Wi-Fi permet de relier, grâce à des ondes radio, différents appareils informatiques à l'intérieur d'un même réseau afin de leur permettre d'échanger des données entre eux.

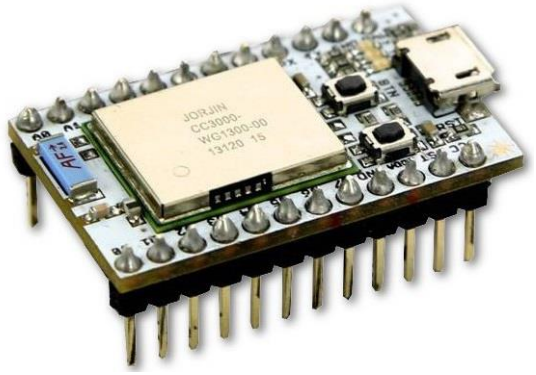
- Il est régit par les normes **IEEE 802.11**

- Différentes versions de ces normes existent et ont permis d'augmenter les débits ou encore d'améliorer la sécurité : 802.11a, 802.11b, 802.11g, 802.11n ...

Les différentes solutions Wi-Fi

- A l'origine les premières cartes Arduino n'avait pas de Wi-Fi, il fallait utiliser un shield, un module Wi-Fi ou encore une carte spécifique





- *1. Shield Wi-Fi pour Arduino*

2. Module ESP8266

3. Spark Core

Présentation du module ESP8266

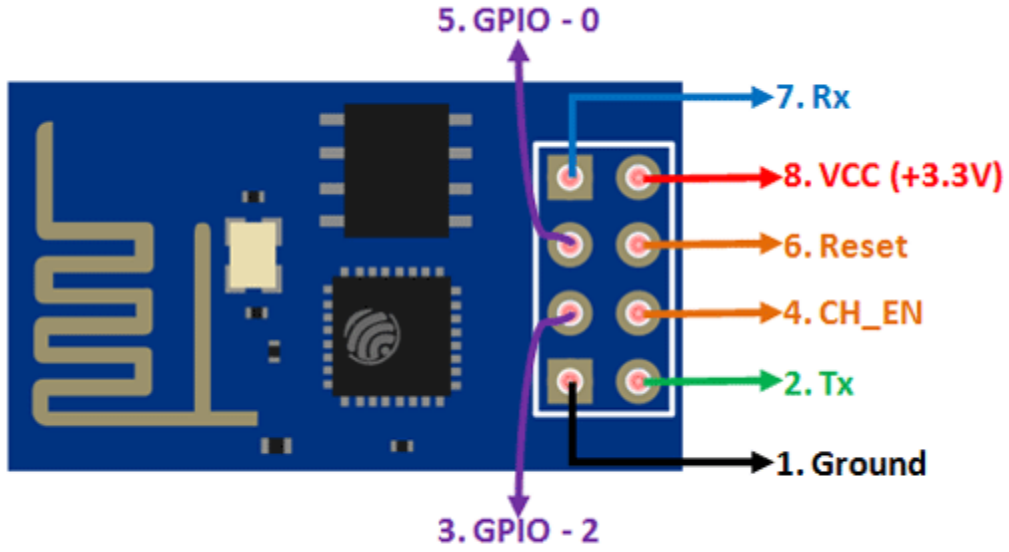
- Le module **ESP8266** est un composant permettant de gérer la communication avec le Wi-Fi, que ce soit de manière autonome ou bien à partir d'un autre composant

- Peu cher (environ 2€), il permet de gérer les protocoles **802.11 b/g/n** et doit être alimenté en **3,3V**

- Il existe plusieurs versions du module qui font toutes à peu près la même chose et différent au niveau du nombre de pins disponibles ainsi que sur la taille de la mémoire flash incluse

Composition du module

- La pun VCC sert pour l'alimentation
- Les pins GPIO 0 et 2 servent pour connecter des éléments
- La pin CH_EN permet d'activer le module
- La pin Tx (GPIO 1) gère la transmission
- La pin Rx (GPIO 3) gère la réception
- La pin reset permet de réinitialiser le module



Configurer le module

- Dans un premier temps, vérifier dans le moniteur en série que le nombre de bauds est à **115200** et que les retours à la ligne sont spécifiés en **nouvelle ligne et retour chariot**

- Dans l'IDE d'Arduino, allez dans « **Fichier** », « **Préférences** » puis ajoutez dans le champ « **URL de gestionnaire de cartes supplémentaires** » cette URL :

http://arduino.esp8266.com/stable/package_esp8266com_index.json

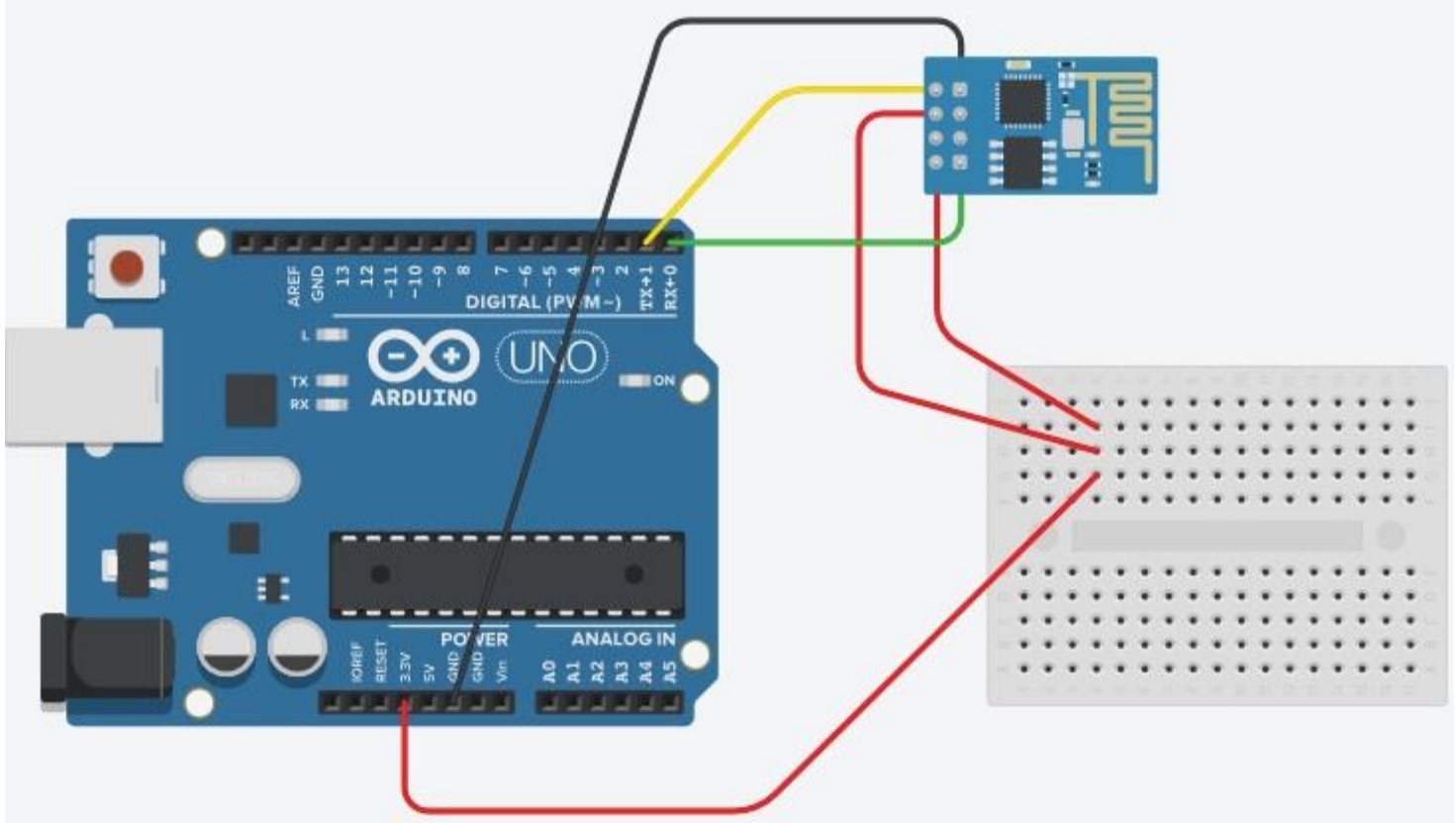
- N'utilisez pas la communication en série sur votre programme Arduino, elle sera utilisée pour communiquer avec le module ! Mettez le programme **blink** sur l'Arduino si besoin

-

Connecter le module à un Arduino

- Connecter le module comme suit :

- En théorie le module ESP8266 possède un circuit de protection pouvant aller jusqu'à 5,8V. On n'a donc pas besoin de rajouter des résistances



Les commandes AT - 1

- En ouvrant le moniteur en série, on va pouvoir envoyer des commandes particulières au module : les commandes AT. Elles permettent de le contrôler.

  - **AT** : renvoie OK
  - **AT+RST** : relance le module
  - **AT+GMR** : retourne la version du firmware
  - **AT+CWMODE** : permet de choisir le mode Wi-Fi à utiliser. AT+CWMODE=1 pour se connecter à un réseau. 2 pour le mode point d'accès. 3 pour les deux modes combinés.

Les commandes AT - 2

- **AT+CWLAP** : renvoie la liste des réseaux Wi-Fi à proximité
- **AT+CWJAP= « nom du réseau », « mot de passe du réseau »** : permet de se connecter à un réseau
- **AT+CIFSR** : permet de récupérer l'adresse IP du module
- **AT+CIPMUX=<mode>** : permet d'indiquer si on gère une seule connexion avec mode = **0** ou de multiples connexions avec mode = **1**

Les commandes AT - 3

- D'autres commandes sont utiles lorsque l'on souhaite établir une communication avec un serveur distant :

  - **AT+CIPSTART** : permet d'initialiser une connexion TCP ou UDP en fonction du type de connexion (unique ou multiple) indiqué par **CIPMUX**
    - Connexion unique : **AT+CIPSTART=<type>, « <adresse IP> », <port>**
    - Connexion multiple : **AT+CIPSTART=<id>, « <type> », »<adresse IP> », <port>**
    - L'id peut être compris entre 0 et 4
    - Le type est soit « **TCP** », soit « **UDP** »

Les commandes AT - 4

- **AT+CIPSEND** : permet d'envoyer des données selon une syntaxe différente suivant le type de connexion (unique ou multiple) indiqué par **CIPMUX**
  - Connexion unique : **AT+CIPSEND=<taille des données>**
  - Connexion multiple : **AT+CIPSEND=<id>, <taille des données>**
- **AT+CIPCLOSE** : permet de terminer la connexion
- **AT+CIPSERVER=<mode>[, <port>]** : lance le module en tant que serveur. Le serveur pouvant être en mode fermé (mode = 0) ou ouvert (mode = 1)

- Lien vers la documentation : *https://www.itead.cc/wiki/ESP8266_Serial_WIFI_Module#AT_Commands*

La librairie SoftwareSerial

- Afin que notre programme Arduino puisse envoyer de lui-même ces commandes, on utilisera la librairie **SoftwareSerial** qui s'utilise comme **Serial**

- Elle permet d'utiliser d'autres pins que **Rx** et **Tx** afin de simuler une connexion en série avec d'autres composants

- On retrouve donc des fonctions tel que **begin()**, **available()**, **println()** ou encore **read()**

- *https://www.arduino.cc/en/Reference/SoftwareSerial*

Petit TP

- Commencez par configurer votre ESP8266 à partir d'Arduino
- Vous mettrez ensuite en place un petit serveur NodeJS
- A partir de votre Arduino, envoyez une requête à votre serveur Nodejs
- Une fois que c'est fonctionnel, ajoutez un capteur à l'Arduino puis envoyez régulièrement les informations de ce capteur à votre serveur