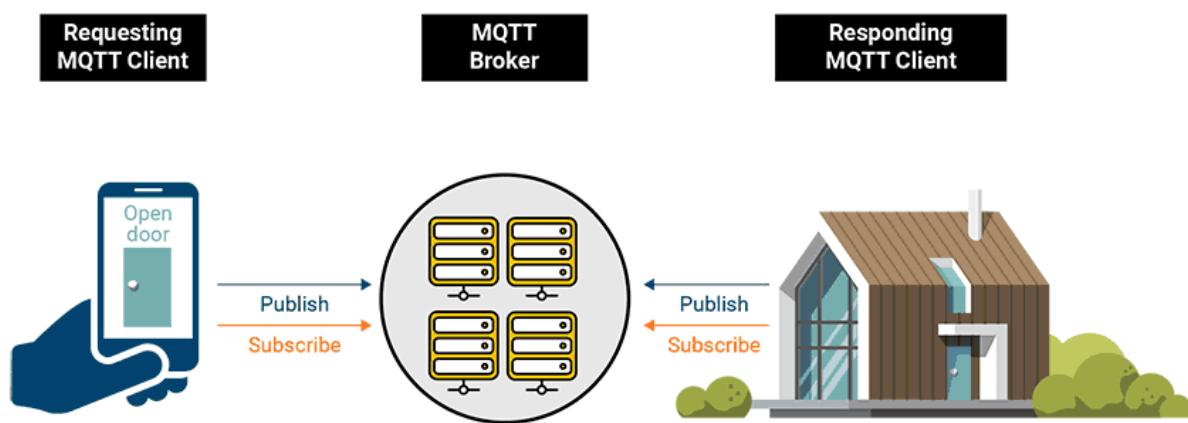




Communication Arduino - Android

Protocole MQTT



Si le gif ne s'anime pas, cliquer [ici](#).

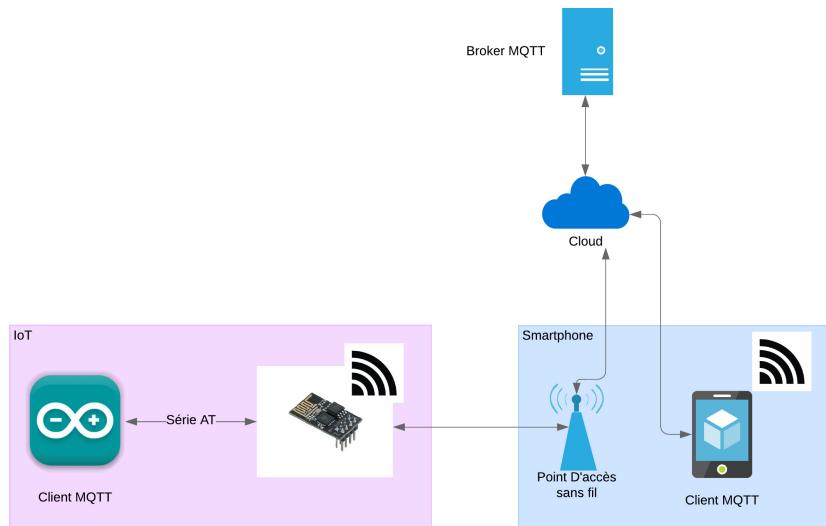
Avantages :

- Protocole de messagerie publish/subscribe basé sur TCP/IP
- Léger : prévu pour l'IoT
- Sécurité : prend en charge TLS, système d'authentification
- Fonctionnalités : QoS, persistence de session cliente, persistence de message sur serveur

Exemples d'utilisations :

- L'Android publie une action, l'Arduino est abonné à une action et la réalise
- L'Arduino publie l'état d'un capteur, l'Android permet une visualisation de son état

Exemple d'infra :



Liens :

- <https://mqtt.org/>
- Exemple de [client ESP-01 avec Arduino](#)
- [Bibliothèque C# qui fait client/broker MQTT](#)
 - Prendre en compte cette [note](#) si on utilise l'android comme broker
- Serveur dédié : [mosquitto](#) (disponible avec docker)
- Liste des [commandes AT, exemples](#)
 - [Support des commandes pour ESP8266](#)
 - AT : test si le module est fonctionnel (renvoie OK)
 - AT+RESTORE : restaure le module à ses paramètres par défaut
 - AT+UART_DEF : configure le port série, exemple ⇒ AT+UART_DEF=115200,8,1,0,3

Qualité de service

Voir <https://hivemq.com> pour plus de détails :

QoS 0 :

- Au plus 1 fois ⇒ donnée envoyée 1 fois, sans garantie de livraison
- Pour connexion stable et donnée non critique
- Pas de fil d'attente des message

QoS 1 :

- Au moins 1 fois ⇒ renvoie la donnée jusqu'à réception d'un PUBACK
- Les abonnés peuvent recevoir plusieurs fois la même donnée (normalement invisible côté applicatif ⇒ flag DUP)

QoS 2 :

- Exactement 1 fois
- Plus lent que QoS 1

Tester le module

Objectif : Faire des commandes AT directement depuis le moniteur série

Éléments nécessaires :

- Arduino Mega 2560
- Module wifi ESP-01
- Adaptateur 5V vers 3V pour ESP-01

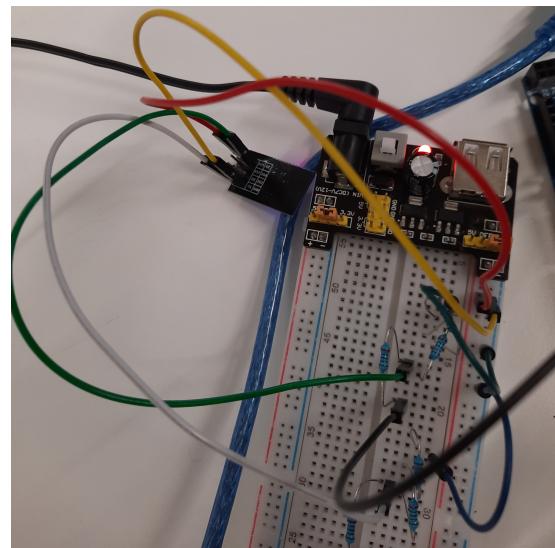
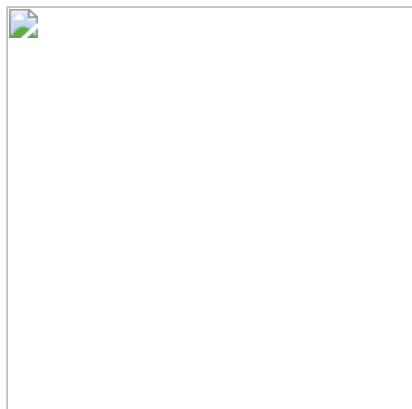
Documentation module wifi ESP-01 www.microchip.ua ou simba-os.readthedocs.io

Problématique : programmer le module wifi

Schéma

Connexion entre Arduino Mega 2560 et ESP-01

Montage



Lancement de commande AT

1. Ouvrir un moniteur série sur /dev/ttyACM0
 - 115200/8/1/N
 - CR+LF ("Both NL&CR" sur Arduino IDE)
2. Écrire "AT" et appuyer sur entrée
 - résultat attendu : OK

3. Se connecter à un wifi (malheureusement, seul l'ESP32 supporte la connexion à un AP entreprise)

- AT+CWJAP=[<ssid>],[<pwd>]
- Réponse attendue : WIFI CONNECTED

Remarques

Une LED rouge et une LED bleue sont allumées sur le module wifi.

Lors de l'envoi des données via le moniteur série, les LED TX et RX de l'Arduino s'illuminent.

Les données en sortie du moniteur série sont illisibles.

Commandes MQTT

Malheureusement, la version installée du firmware de l'ESP ne prend pas en charge MQTT.

Version actuelle :

- AT version:1.1.0.0(May 11 2016 18:09:56)
- SDK version:1.5.4

Il faut upgrade vers la 2.2.1

Pour upgrade, voir https://docs.espressif.com/projects/esp-at/en/release-v2.2.0.0_esp8266/Get_Started/Downloading_guide.html

Tentative de flash du firmware sans adaptateur TTL vers USB

1. Même branchement que vu précédemment
2. Installer [esptool](#)
3. Lancer la commande `esptool.py --chip auto --port /dev/ttyACM0 --before default_reset --after hard_reset write_flash -z download.config` pour flasher le firmware
 - a. Échec
4. Tentatives avec l'argument `--baud 9600` pour tester avec un débit plus faible, configuration de l'esp avec la commande AT : `AT+UART_CUR=9600,8,1,0,3`
 - a. Échec

Raisons potentielles d'échec

- Utiliser des fils rend les branchements sensibles au bruit. Il faut éviter d'utiliser une breadboard.
- L'arduino n'est pas assez puissante pour alimenter le module. Utiliser une alim externe rend la connexion série inutilisable.

Nouvelle tentative avec un adaptateur USB :

1. Commande : `esptool.py --chip auto --port /dev/ttyUSB0 --baud 115200 --before default_reset --after hard_reset write_flash -z --flash_mode dio --flash_freq 80m --flash_size detect 0x8000 partition_table/partition-table.bin 0x9000 ota_data_initial.bin 0x0 bootloader/bootloader.bin 0x10000`

```
esp-at.bin 0xF0000 at_customize.bin 0xFC000 customized_partitions/client_ca.bin 0x106000  
customized_partitions/mqtt_key.bin 0x104000 customized_partitions/mqtt_cert.bin 0x108000  
customized_partitions/mqtt_ca.bin 0xF1000 customized_partitions/factory_param.bin 0xF8000  
customized_partitions/client_cert.bin 0xFA000 customized_partitions/client_key.bin
```

a. Succès !

2. Tester une commande AT

a. Échec : le binaire fournit par espressif fait 2Mo, cependant la flash de l'esp fait 1Mo

En compilant les sources :

1. Faire un fork de <https://github.com/espressif/esp-at>
2. Lancer github action sur la branche v2.2.0.0_esp8266
3. Télécharger esp8285-1MB-at dans les artefacts
4. Commande :

```
esptool.py --chip auto --port /dev/ttyUSB0 --baud 115200 --before default_reset --  
after hard_reset write_flash -z --flash_mode dout --flash_freq 26m --flash_size detect 0x8000  
partition_table/partition-table.bin 0x9000 ota_data_initial.bin 0x0 bootloader/bootloader.bin 0x20000  
esp-at.bin 0x18000 at_customize.bin 0x1A000 customized_partitions/client_cert.bin 0x1B000  
customized_partitions/client_key.bin 0x1C000 customized_partitions/client_ca.bin 0x1D000  
customized_partitions/mqtt_cert.bin 0x1E000 customized_partitions/mqtt_key.bin 0x1F000  
customized_partitions/mqtt_ca.bin 0x19000 customized_partitions/factory_param.bin
```

5. Tester une commande AT

a. Échec

b. Problème potentiel : la nouvelle version map les ports vers le GPIO au lieu de TX et RX

En remapant les broches :

1. Issue sur Github relevant le pb : <https://github.com/espressif/esp-at/issues/606>
2. Télécharger la version compilée : <https://github.com/espressif/esp-at/issues/606#issuecomment-1033569789> (un grand merci à JAndrassy ❤)
3. Téléverser avec esptool.py
4. Tester les commandes AT
 - a. Succès !
 - b. Les commandes AT pour MQTT fonctionnent !
5. Se connecter à un broker MQTT
 - a. AT+MQTTUSERCFG=0, 6, "user", "mdp", "id", 0, 0, ""
 - b. AT+MQTTCNN=0, "IP", PORT, 0
 - i. Erreur : l'ESP crash et reboot, pas assez de mémoire ?
<https://github.com/espressif/esp-at/issues/566>

Logiciel

Lib MQTT : <https://github.com/arduino-libraries/ArduinoMqttClient>

Programmer le module ESP-01

Prérequis :

- Arduino
 - File > preferences > Additional boards manager URLs :
https://arduino.esp8266.com/stable/package_esp8266com_index.json
 - Board manager : Chercher ESP8266 et l'installer
 - Library manager : Chercher PubSubClient et l'installer
 - Communication série : 115200bps (à vérifier)

Liens utiles :

- <https://www.princetronics.com/arduino-uno-as-usb-to-serial-ttl-converter>
- <https://www.instructables.com/Using-ESP-01-and-Arduino-UNO/>

Blink

Plateforme de cloud IoT en low code : <https://blynk.io>

Arduino Mega 2560 with ESP8266 (ESP-01) Wifi, AT Commands and Blynk

In this video, I show how to connect an ESP8266 to the Arduino Mega 2560, and I send data to my phone which is running the Blynk app. I can also control an LED from the Blynk app. This tutorial assumes that you already know how to create

▶ <https://www.youtube.com/watch?v=YLKEZtLhfZo>

