

Contrôler un arduino depuis une page web

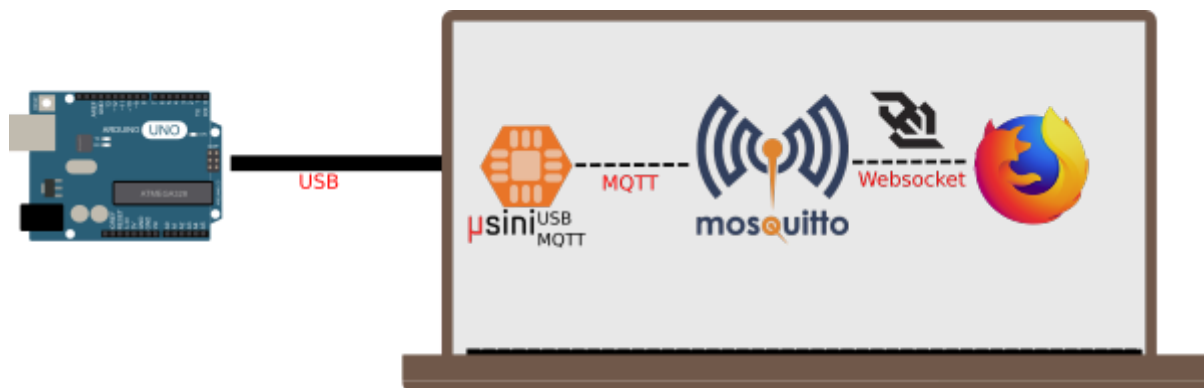
Nous allons apprendre comment connecter un arduino à un réseau MQTT et contrôler la led intégré depuis une page web.

Tout d'abord nous allons

- voir quels programmes va être utilisé pour créer notre réseau
- programmer un arduino afin de pouvoir le contrôler depuis le moniteur série
- installer et configurer mosquitto sous Windows
- créer une passerelle série vers MQTT
- utiliser une interface web basique pour contrôler une led.

Fonctionnement de notre réseau

Avant de se lancer, regardons un peu les programmes que nous avons besoin.



Pour contrôler la led de notre Arduino depuis une page web, 3 programmes sont nécessaires.

Chaque programme peut fonctionner sur des ordinateurs différents.

Vous n'avez pas besoin usb-mqtt si votre microcontrôleur est capable se connecter au Wi-Fi.

Usini usb-mqtt

Ce programme va transmettre les commandes envoyés depuis le port USB, au serveur mosquitto grâce au protocole MQTT.

<https://github.com/usini/usini-usbmqtt>

Mosquitto

Mosquitto est un serveur MQTT , il fait le lien entre les objets connectés (ici, notre arduino) et les interfaces clients (ici, une page web).

<https://mosquitto.org/>

Un navigateur web

Une fois nos objets connectés à Mosquitto, il est possible de communiquer avec lui de nombreuses façon.

J'ai choisi de vous montrer comment le connecter avec une simple page web (**led_example.html**).

On va pouvoir allumer et éteindre notre led et savoir son état actuel.

Cette méthode permet de pouvoir complètement customiser l'interface de contrôle.

Le navigateur web ne peut pas communiquer directement avec mosquitto pour cela le **protocole websocket** est nécessaire.

Programmer l'arduino

Commençons par le programme qui va nous permettre de contrôler notre arduino.

Ce programme utilise la connexion série fournie par le port USB.

il permet d'allumer et d'éteindre la led sur l'arduino avec quatre commandes.

- on → Allume la led
- off → Eteint la led
- status → Affiche l'état de la led
- name → affiche le nom du croquis

```
//Control the built-in led using serial commands

/* Command lists
ON      --> Turn on led
OFF     --> Turn off led
name    --> Sketch name
status  --> Tell if led is on or off
*/

const String sketch_name = "example_led.ino";
const int led = 13;

String command; //Serial string buffer
String state = "OFF"; //Led's state

void setup() {
  Serial.begin(9600); //Setup Serial
```

```

    Serial.println(sketch_name);    // Display name at startup
}

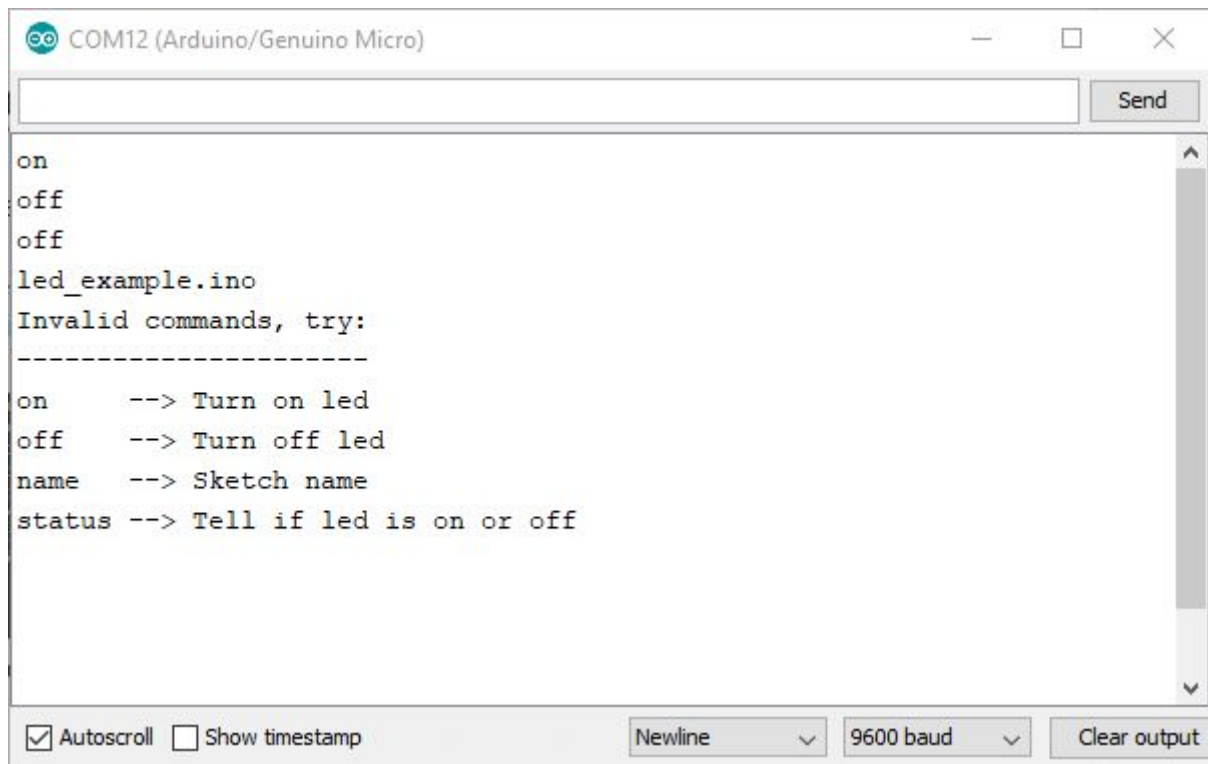
void loop() {
    /*
        When a command is sent, we save it as a string
    */
    if (Serial.available() > 0){
        command = Serial.readStringUntil('\n'); //
        command.toUpperCase(); //ignore lower/uppercase
    }
    //If a command is received do something
    if (command.length() > 0) {
        Serial.println(command);

        // Get sketch name
        if (command == "name") {
            Serial.println(sketch_name);
        }
        //Get current status of led
        else if (command == "status") {
            Serial.println(state);
        }
        //Turn led on
        else if (command == "on") {
            digitalWrite(led, HIGH);
            state = "ON";
            Serial.println(state);
        }
        //Turn led off
        else if (command == "off") {
            digitalWrite(led, LOW);
            state = "OFF";
            Serial.println(state);
        }
        else {
            Serial.println("Invalid commands, try:");
            Serial.println("ON      --> Turn on led");
            Serial.println("OFF     --> Turn off led");
            Serial.println("name   --> Sketch name");
            Serial.println("status --> Tell if led is on or off");
        }

        //We remove the command so it isn't reread
        command = "";
    }
}

```

Téléverser ce programme sur votre arduino puis ouvrez le moniteur série de l'arduino (Nouvelle Ligne / 9600) afin de l'essayer.



Préparer Mosquitto sous Windows

Nous avons vu comment contrôler notre arduino à l'aide du moniteur série, c'est amusant mais très limité.

Une seule personne peut contrôler la led et l'interface n'est pas pratique.

Pour pouvoir le contrôler à distance, il va nous falloir mettre en place un serveur Mosquitto.

Cette partie est un peu délicate, si vous animez un atelier, faites là par avance afin de fournir une version portable de mosquitto aux participants.

Installation de mosquitto

Installer Mosquitto sous Windows nécessite une étape supplémentaire, car il manque deux fichiers qu'il va falloir récupérer dans **openSSL**.

La bonne nouvelle c'est que l'on peut créer une version portable que l'on va pouvoir mettre sur une clé USB afin de pouvoir le faire fonctionner en un clic !

Tout d'abord il nous faut télécharger Mosquitto et Openssl

<https://mosquitto.org/download/>

<http://slproweb.com/products/Win32OpenSSL.html>

Sur les deux sites, plusieurs versions sont proposés,

Voici lesquelles téléchargés:

- **mosquitto-1.5.4-install-windows-x64.exe**
- **Win64 OpenSSL v1.1.1a Light**

Ensuite installer mosquitto et openssl

- Installer mosquitto dans un dossier facilement accessible (par ex: le bureau de windows).
- Installer openssl **dans le même dossier que mosquitto**

Seul les fichiers **libcrypto-1_1-x64.dll** et **libssl-1_1-x64.dll** sont nécessaires.

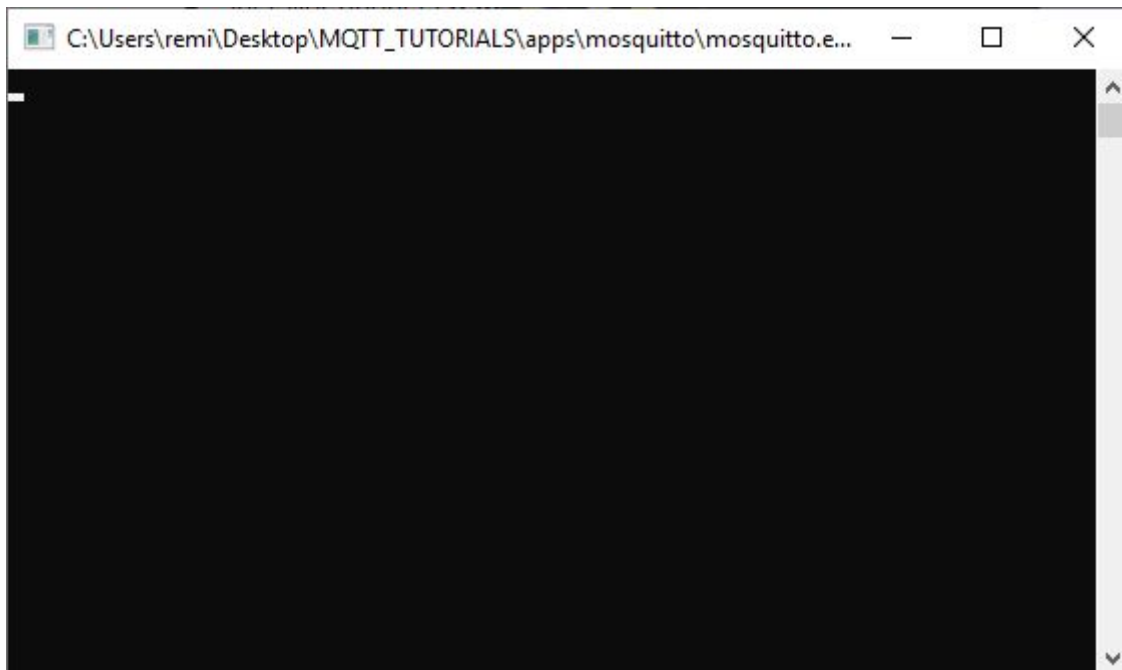
Vous pouvez aussi les copier manuellement dans le dossier où mosquitto est installé.

Mais cette technique permet de faire cela sans avoir à déplacer les fichiers.

Première lancement de mosquitto

Cliquez sur **mosquitto.exe**, normalement vous ne devriez pas avoir de message d'erreurs

Pour autant rien ne s'affiche, mosquitto fonctionne pour autant.



Configurer mosquitto

En effet, par défaut mosquitto n'affiche aucun message, et surtout le websocket n'est pas actif donc on ne pourra pas y accéder depuis une page web.

Pour changer cela, il va nous falloir modifier le fichier de configuration : **mosquitto.conf**

Notre modification va permettre plusieurs choses:

- Afficher ce qu'il se passe (les logs)
- Activer le protocole Websocket

Ouvrez le fichier **mosquitto.conf** et ajouter:

```
log_dest stdout
log_type error
log_type warning
log_type notice
log_type information

listener 1883
listener 9001
protocol websockets
```

On active les logs pour les erreurs / les avertissements / les informations puis on ajoute deux **listeners**:

Le premier, 1883 écoute les messages envoyés depuis le protocole **MQTT**

Le deuxième, 9001 écoute les messages envoyés par le protocole **websocket**

Préparer le script de lancement

Voilà notre serveur MQTT mosquitto est paramétré, mais si vous lancez **mosquitto.exe**, la configuration ne sera pas pris en compte.

Il faut lui dire d'ouvrir le fichier de configuration.

Pour cela, nous allons créer un fichier .bat pour le lancer sans passer par la ligne de commande.

Appellons le **start_mosquitto.bat**

Dans ce fichier mettez ceci:

```
mosquitto -c mosquitto.conf
pause
```

Cela lancera mosquitto avec le fichier de configuration.

Par précaution on met une **pause** afin que si le programme ne démarre pas, on puisse voir le message d'erreur.

Cliquer sur **start_mosquitto.bat**

Vous devriez voir ceci:

```
1547568848: mosquitto version 1.5.4 starting
1547568848: Config loaded from mosquitto.conf.
1547568848: Opening ipv6 listen socket on port 1883.
1547568848: Opening ipv4 listen socket on port 1883.
1547568848: Opening websockets listen socket on port 9001.
```

Voilà mosquitto est prêt à fonctionner !

Connexion MQTT

Il faut maintenant que nous fassions le lien entre notre arduino et mosquitto.

Il a plusieurs façon de créer une passerelle série vers MQTT.

La plus connu est d'utiliser **node-red**, mais à ma connaissance, il n'existe pas un programme pour connecter un périphérique USB série à un réseau en un clic sans installation.

<https://nodered.org/>

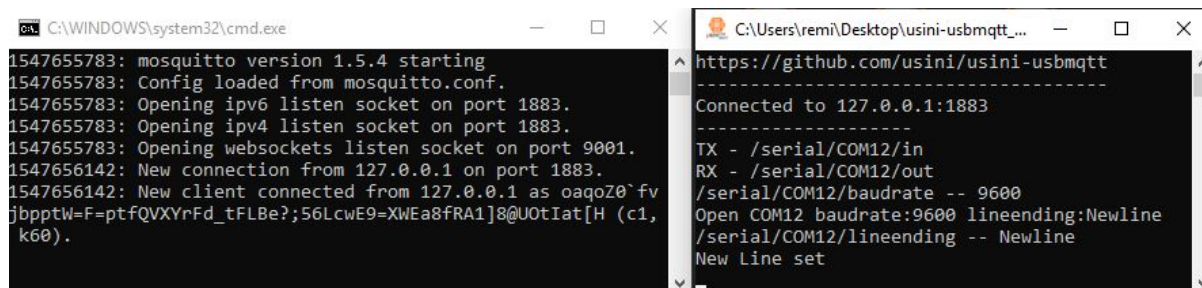
C'est pour cela que j'ai créer un programme: **usini-usbmqtt** avec python et que je l'ai compilé à l'aide de **pyinstaller** afin qu'il puisse fonctionner sous Windows/Linux et un raspberry pi sans avoir à installer python.

Vous pouvez le télécharger ici : <https://github.com/usini/usini-usbmqtt>

Utilisation

Afin d'utiliser **usini-usbmqtt** il suffit de cliquer sur **usini-usbmqtt_win.exe**

N'oubliez pas d'arrêter le moniteur série arduino avant de lancer usini-usbmqtt, les ports série ne peut être connectés qu'à un programme à la fois!



```
C:\WINDOWS\system32\cmd.exe
1547655783: mosquitto version 1.5.4 starting
1547655783: Config loaded from mosquitto.conf.
1547655783: Opening ipv6 listen socket on port 1883.
1547655783: Opening ipv4 listen socket on port 1883.
1547655783: Opening websockets listen socket on port 9001.
1547656142: New connection from 127.0.0.1 on port 1883.
1547656142: New client connected from 127.0.0.1 as oaqoZ0`fv
jbpptW=F=ptfQVXYrFd_tFLBe?;56LcwE9=XWEa8fRA1]8@UOtIat[H (c1,
k60).

C:\Users\remi\Desktop\usini-usbmqtt_...
https://github.com/usini/usini-usbmqtt
-----
Connected to 127.0.0.1:1883
-----
TX - /serial/COM12/in
RX - /serial/COM12/out
/serial/COM12/baudrate -- 9600
Open COM12 baudrate:9600 lineending:Newline
/serial/COM12/lineending -- Newline
New Line set
```

usini-usbmqtt va chercher le premier port série disponible.

Utilisation avec choix du port

Si le port n'est pas le même que celui dans le logiciel Arduino, il se peut qu'il existe d'autres ports séries, dans ce cas il va falloir préciser le port USB.

Vous pouvez créer un fichier bat comme nous l'avons vu précédemment avec mosquitto, soit lancer usini-usbmqtt depuis la ligne de commande avec l'argument **--port**

usini-usbmqtt --port LE PORT DE L'ARDUINO

Sous windows ce port commence toujours par COM et finit par un nombre.

Par exemple:

usini-usbmqtt --port COM5

Il existe d'autres paramètres, vous pouvez les voir en tapant **usini-usbmqtt --help**

Ou sur la page github : <https://github.com/usini/usini-usbmqtt>

Gardez à l'esprit que usini-usbmqtt n'est qu'en version beta, il se peut que certains fonctionnalités ne marchent pas correctement.

Interface web

Utilisation

Notre arduino est connecté à mosquitto, on va pouvoir lui envoyer des commandes, usini-usbmqtt créer plusieurs **topics**.

Les plus importants sont:

/serial/PORT/in → Envoi un message à l'arduino

/serial/PORT/out → Réception d'un message depuis l'arduino

/serial/PORT/status → Etat de l'arduino (online/offline/serialerror)

Par ex si votre arduino est connecté au port COM5

/serial/COM5/in

/serial/COM5/out

*usini-usbmqtt envoie les messages de l'arduino avec **retain** ce qui permet de garder le dernier message envoyés par celui-ci en mémoire.*

Vous pouvez essayer de le contrôler avec un client mqtt (comme **mqtt.fx** sous windows : <https://mqttfx.jensd.de/>).

Mais pour commencer le plus simple est d'utiliser mon client web qui va fonctionner directement.

Cliquer sur **led_example.html**.

Si vous cliquer sur le bouton on, la led s'allume , le bouton off éteint la led.

Au dessus dans le rectangle noir (ou vert si la led est allumé) vous pouvez voir l'état de la led.

Au dessous dans les rectangles jaunes vous pouvez voir si la connexion au serveur mosquitto fonctionne et si vous êtes bien connecté à votre arduino.



Explication du code

Nous allons voir quelques passages du code source de **led_example.html**.

Connexion

La page web utilise le client javascript de paho-mqtt afin de gérer la connexion à notre arduino: <https://www.eclipse.org/paho/clients/js/>

Au début du code vous pouvez modifier l'adresse du serveur (**hostname**) et le port du listener websocket (**port**).

La connexion se fait en deux étapes:

D'abord nous créons le client.

```
client = new Paho.MQTT.Client(hostname, port, random_client_id);
```

puis nous affectons des fonctions qui vont être appelés en cas de perte de connexion et si un message est reçu.

```
client.onConnectionLost = onConnectionLost;
```

```
client.onMessageArrived = onMessageArrived;
```

Ensuite nous connectons le client.

Encore une fois nous affectons des fonctions en cas de succès ou d'échec.

Si on veut utiliser des certificats SSL (pour chiffrer la connexion) ou utiliser un mot de passe c'est ici que la configuration se fait.

```
//Connect client
```

```
client.connect({
```

```
    //called when client connect
```

```
    onSuccess:onConnect,
```

```
    //called when client failed to connect
```

```

onFailure:onFailure
//useSSL:true,
//userName: "",
//password: ""
});

```

la fonction **onConnect**, est active lorsque la connexion à mosquitto est faite..

On s'abonne à **/serial/#** afin d'avoir accès à tout les topics liés aux arduinos connectés avec usini-usbmqtt.

```
client.subscribe("/serial/#");
```

Réception de message

la fonction **onMessageArrived** est active lorsque l'on reçoit un message provenant de **/serial/#**

Lorsque la page web est ouverte, on ne connaît pas le port de notre arduino, c'est pour cela que l'on va chercher le premier topic : **/serial/PORT/status** où le status n'est pas offline afin de savoir dans quel topic envoyer et recevoir les messages.

```

if(serial_port = "")
    topic_splited = message._getDestinationName().split("/")
    if(topic_splited[3] == "status"){

```

Si un arduino est trouvé c'est la partie **else** qui sera active

Si le message on ou off provient de notre arduino on modifie la couleur de l'élément led (le bloc au début de la page).

```

if(message._getDestinationName() == serial_rx){
    if(message_received == "off"){
        //Turn led box black
        document.getElementById("led").style.backgroundColor = "black";
    }
}

```

Envoi de message

La fonction send permet d'envoyer un message à notre arduino

Cela permet de créer facilement des boutons pour le contrôler:

```
<button style="color:lightgreen" onclick="send('on')">ON</button>
```

Conclusion

Nous avons vu comment connecter un arduino à une page web, pour cela nous avons utilisé la connexion USB série pour la transmettre à un serveur MQTT puis à notre navigateur web.

Il peut être compliqué d'utiliser un microcontrôleur avec le wifi, car il faut gérer à la fois la connexion et les composants branchés dessus.

Cette méthode permet de tester simplement une idée.