ACTIVITE 4
Serveur MQTT

Vous allez mettre en œuvre le protocole MQTT.
Nous utiliserons le broker "Mosquitto" qui a l'avantage d'être gratuit et performant.

Installation et utilisation du serveur MQTT de Mosquitto

☞ 1 – Vérifiez si le serveur MQTT de Mosquitto est déjà installé sur votre ordinateur.

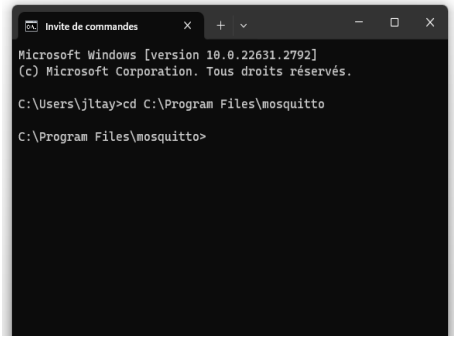
- Tapez dans l'invite de commande : "**cd C:\Program Files\mosquitto**".
- Si vous n'avez pas de message d'erreur, c'est que le serveur est déjà installé. Dans ce cas, sautez les questions 2 et 3.

☞ 2 – Télécharger la dernière version de Mosquitto sur le site : https://mosquitto.org/download/
(Version Windows 64 bits)

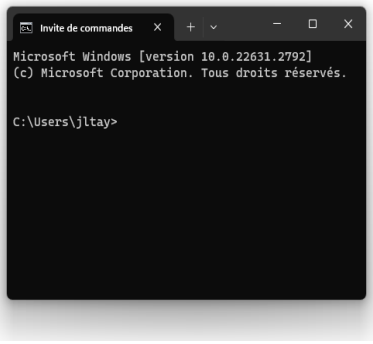
☞ 3 – Double cliquez sur le fichier exécutable et gardez les paramètres par défaut.
Retenez le chemin d'installation car vous en aurez besoin pour la suite.
Chemin du type "**C:\Program Files\mosquitto**"

✍ 4 – Vous allez tester la structure IoT. Ouvrir trois fenêtres "Invite de commande" et déplacez-vous dans le dossier d'installation de Mosquitto.
Commande :

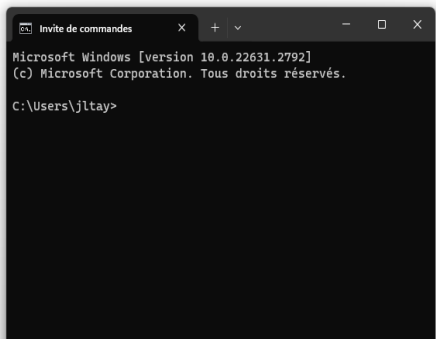
Le première fenêtre représentera le client 1 (publisher), la deuxième le broker (courtier) et la troisième pour le client 2 (subscriber).
Dans la réalité, ces trois fenêtres sont sur des machines différentes mais pour tester la structure, nous exécuterons les trois fenêtres sur la même machine.



Le client (publisher)
Il envoie
(Simule un objet connecté)



Le Broker
Il gère les topics



Le client (Subscriber)
Il reçoit
(Simule le tableau de bord qui reçoit la donnée)

☞ 5 – Activer le Broker. Tapez dans l'invite de commande "**mosquitto.exe**".

☞ 6 – Positionnez-vous dans la fenêtre du client 1 (publisher) pour l'envoie d'un topic vers le Broker.
Pour obtenir l'aide sur les commandes "Mosquitto" vous pouvez tapez cette commande : **mosquitto_pub –help**

Pour envoyer une valeur : **mosquitto_pub -h localhost -t temperature -m 25**

-h localhost --> -h indique que l'on va préciser l'adresse IP du broker où dans notre cas "localhost" puisque le broker est installé sur notre ordinateur.

-t temperature --> -t indique le nom du topic. Dans notre cas "temperature".

-m 25 --> -m indique la valeur à transmettre. Dans notre cas "25".

Faites un essai. Envoyez une valeur vers le serveur MQTT.

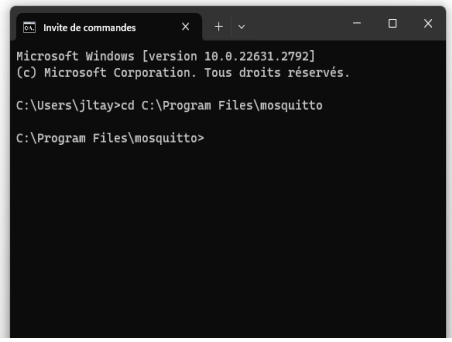
✍ 7 – Que constatez-vous sur la fenêtre du client 2 (Subscriber) ?

☞ 8 – Pour recevoir des données sur le client 2 (Sbscriber) il faut au préalable s'abonner.
Abonnement à un topic : **mosquitto_sub -h localhost -t temperature**
Abonnez-vous au topic "temperature" depuis le client (subscriver) puis envoyer une nouvelle valeur au serveur MQTT depuis le client (publisher)

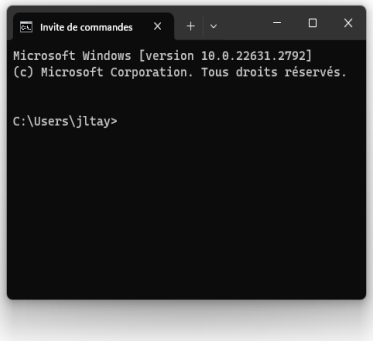
✍ 9 – Que constatez-vous sur la fenêtre du client 2 (Subscriber) ?

On constate que la communication fonctionne localement, lorsque le publisher, le broker et le subscriber sont sur la même machine.
Vous allez déployer cette solution dans un cas plus classique :
- Votre machine hébergera le client 1 (publisher) et servira aussi pour le client 2 (subscriber) pour des topics générés par d'autres machines de la salle de TP.
- Le broker est hébergé sur un serveur (déjà configuré) à l'adresse : **10.194.196.24:1883**

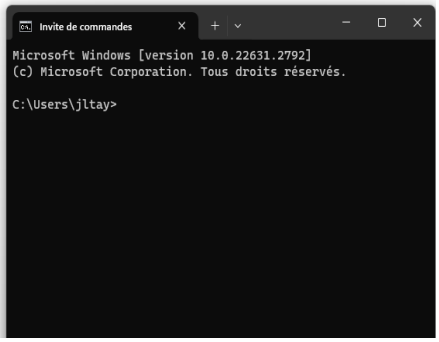
☞ 10 – Associez-vous avec un autre étudiant du groupe de TP. Echangez-vous des topics via le broker de la salle ET132 ... même si vous êtes en ET130.
Lorsque la solution sera opérationnelle, vous passerez à la suite du TP.



Le client (publisher)
Carte Arduino + Node-Red
Votre machine



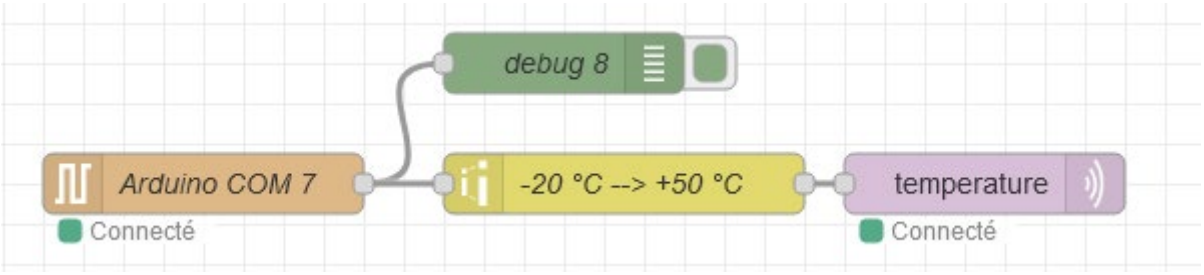
Le Broker
Il gère les topic
10.194.196.24 port 1883



Le client (Subscriber)
Votre machine ou une autre machine de la salle

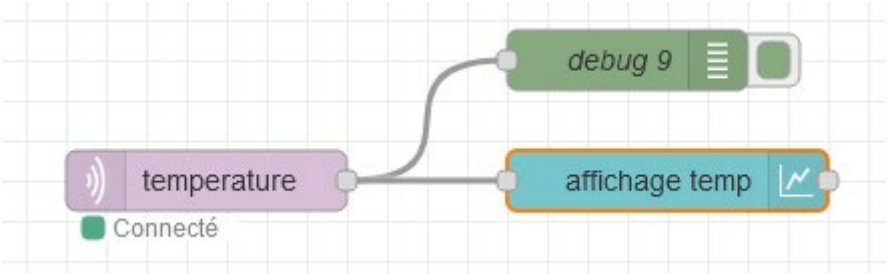
☞ 11 – Vous allez remplacer le client 1 (Invite de commande) par une solution plus performante. Remplacez le client "Publisher" par votre carte Arduino associée à Node-Red.
- Reprenez la structure que vous avez mise en place lors de l'activité 2.

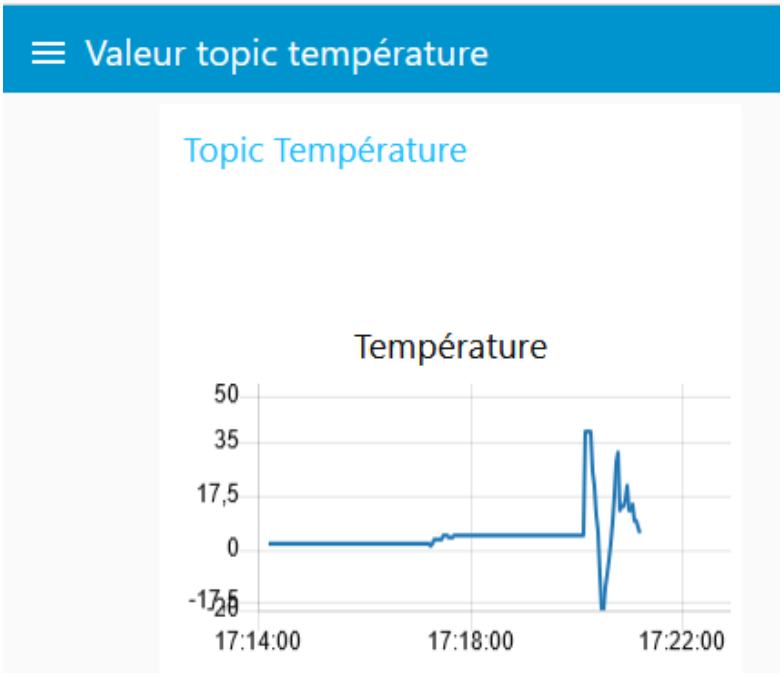
- Ajoutez et configurez un nœud pour envoyer au serveur MQTT les données récupérées par Node-Red.
    - Exécutez **Node-red** dans l'invite de commande en tant administrateur.
    - Connectez-vous à Node-red à partir du navigateur : **localhost:1880**
    - Ajoutez le nœud "**mqtt out**".
    - Configurez-le.
    - Pour que cela soit plus réaliste, convertissez les valeurs comprises entre 0 et 1023 en des valeurs de température comprises entre -20 et 50°C.



☞ 12 – La dernière étape consiste à remplacer le client 2 (subsciber) par un beau tableau de bord Node-Red.

- Ouvrir un nouvel onglet sous Node-Red.
- Ajoutez un nœud "MQTT IN" pour récupérer les données du Broker.
- Configurez le nouveau nœud.
- Construire le tableau de bord.
- Ajoutez dans le tableau de bord des topics provenant de machines différentes.
-

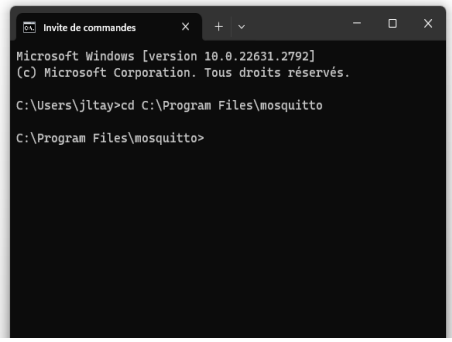




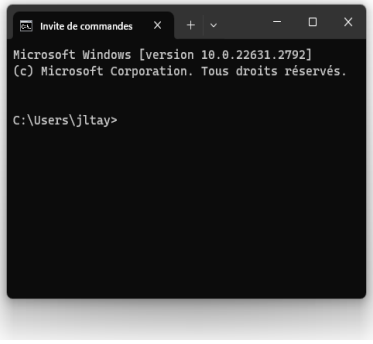
Configuration et sécurisation du Broker MQTT

Dans cette partie vous n'allez pas utiliser le broker de la salle ET132 mais vous allez configurer et sécuriser votre propre serveur.

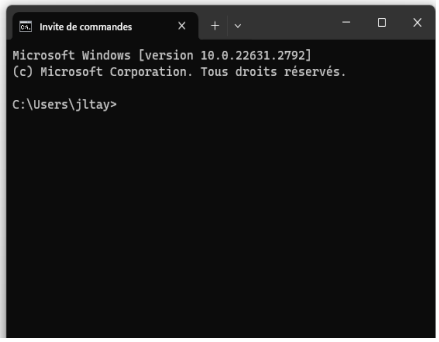
Vous n'utiliserez pas Node-Red, ni de carte Arduino pour la suite des opérations.
Vous utiliserez la structure mise en place à la question 4 : trois fenêtres "Invite de commande" **en tant qu'administrateur.**



Le client (publisher)
Il envoie
(Simule un objet connecté)



Le Broker
Il gère les topic



Le client (Subscriber)
Il reçoit les publications

Configuration du broker Mosquitto

☞ 13 – Vous pouvez modifier le fichier de configuration *par* défaut ***mosquitto.conf*** à partir du répertoire d'installation.
Le fichier ***mosquitto.conf*** par défaut dans le dossier d'installation contient des lignes d'exclusion (lignes qui commencent par le **#**) avec des options par défaut. Pour les changer, vous devez ne pas commenter les lignes que vous voulez (en enlevant le **#**) et spécifiez des valeurs différentes.

☞ 14 – Il faut préciser le port que le broker doit écouter : 1883 (port généralement utilisé par Mosquitto)
Retrouver dans le fichier de configuration la section :

```
# ================================================================
# Listeners
# ================================================================

# Listen on a port/ip address combination. By using this variable
# multiple times, mosquitto can listen on more than one port. If
# this variable is used and neither bind_address nor port given,
# then the default listener will not be started.
# The port number to listen on must be given. Optionally, an ip
# address or host name may be supplied as a second argument. In
```

Ligne à modifier : **listener 1883 @IPdu Broker** (enlevez le # devant listener)
Noter l'adresse IP du broker

☞ 15 – Vous pouvez autoriser des connexions anonymes à partir de n'importe quel hôte (c'est-à-dire des connexions sans nom d'utilisateur ni mot de passe) en ajoutant les options suivantes à votre fichier de configuration : **allow_anonymous true.**

☞ 16 – Relancez Mosquitto mais en lui précisant de lire le fichier de configuration pour appliquer les modifications effectuées.

Dans l'invite de commande broker, tapez : **mosquitto -v -c mosquitto.conf**

-v permet de connaitre la version et le port d'écoute.
-c précise le nom du fichier de configuration.

Si Windows demande une autorisation (firewall) --> accepter

☞ 17 – Pour vérifier si Mosquitto fonctionne sur le port 1883, lancez la commande suivante :

netstat -an | findstr 1883

Si le serveur Mosquitto MQTT a ouvert un socket d'écoute IPv4 et IPv6 sur le port 1883, la sortie de cette commande sera du type :

```
C:\Program Files\mosquitto>netstat -an | findstr 1883
  TCP    0.0.0.0:1883           0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1883         0.0.0.0:0              LISTENING
  TCP    192.168.1.27:1883      192.168.1.27:7897      ESTABLISHED
  TCP    192.168.1.27:7897      192.168.1.27:1883      ESTABLISHED
  TCP    [::1]:1883             [::]:0                 LISTENING
```

Pensez à modifier la config "@IP" des nœuds mqtt_in et mqtt_out de node-red.

☞ 18 – Testez la communication Arduino-Broker – Node Red :

Sécurisation par mot de passe

☞ 19 – Mise en place de l'authentification Utilisateur / Mot de passe.

Vous utiliserez la commande **mosquitto-passwd** pour ajouter ou supprimer un utilisateur (client) de Mosquitto.

Commande : **mosquitto_passwd -c pwdfile utilisateur1**

Pwdfile représente le fichier dans lequel vous allez sauvegarder les identifiants des utilisateurs.
utilisateur1 est le nom de l'utilisateur que l'on souhaite créer.
Entrez le mot de passe lorsque l'invite de commande le demande.

Mot de passe : **MotDePasse1**

Autres commandes :

mosquitto_passwd -b pwdfile utilisateur2 MotDePasse2

mosquitto_passwd -D utilisateur1

- c : pour créer le fichier et ajouter un utilisateur
- b : pour ajouter un utilisateur lorsque le fichier existe
- D : pour supprimer un utilisateur

Créez deux utilisateurs.

Utilisateur 1 : **util1** Mot de passe : **mdputil1**
Utilisateur 2 : **util2** Mot de passe : **mdputil2**

✍ 20 – Vérifiez le contenu du fichier pwdfile (dossier d'installation de Mosquitto). Que constatez-vous ?

Pour la suite de l'activité, vous n'activerez pas les connexions anonymes pour des raisons de sécurité.

☞ 21 – Modifier le fichier mosquitto.conf et mettre le paramètre **allow_anonymous** à **false.**

☞ 22 – Il faut aussi préciser l'emplacement du fichier contenant les mots de passe.

password_file C:\Program Files\mosquitto\pwdfile

☞ 23 – Redémarrez le broker et modifiez la configuration de node-red pour que les utilisateurs "util1" et "util2" soient pris en compte.

- Le client 1 (publisher) utilisera l'identifiant "util1" plus le mot de passe pour d'identifier sur le broker.
- Abonnez le client 2 au topic "temperature" avec l'utilisateur "util2".

Pour info, si on n'utilise pas Node-Red voici les lignes de commande qu'il faudrait utiliser pour envoyer ou lire un topic sur le Broker.

Publisher : **mosquitto_pub -h 192.168.1.27 -u util1 -P mdputil1 -t temperature -m 21**
Subscriber : **mosquitto_sub -h 192.168.1.27 -u util2 -P mdputil2 -t temperature**

- -u : nom utilisateur
- -P : mot de passe

✍ 24 – Testez votre solution - Conclusion

Chiffrement de la communication avec TLS

Après avoir mis en place l'authentification par mot de passe, il faudrait ajouter une couche supplémentaire en chiffrant les communications.
Nous ne disposons pas de certificat d'autorité de certification, donc vous ne mettrez pas en place le chiffrement de la communication.

Pour information, voici la procédure qu'il faudrait appliquer :

1 - Changer le port : le port habituellement utiliser pour le chiffrement est 8883.

2 – Modifier le fichier de configuration :
listener 8883
cafile <chemin d'accès du certificat d'autorité de certification>
certfile < chemin d'accès du certificat client pour l'authentification auprès du serveur >
keyfile < chemin d'accès vers la clé privée >
tls_version tlsv1.2

3 - Après avoir apporté des modifications au fichier de configuration, redémarrez Mosquitto.

Exemple de commande pour publier un message

mosquitto_pub -h 192.168.1.27 **-p** 8883 **-u** util1 -P mdputil1 **-t** temperature **-m** "21" **--cafile** /path/to/ca.crt **--cert** /path/to/client.crt **--key** /path/to/client.key **--tls-version** tlsv1.2

Exemple de commande pour s'abonner à un topic

mosquitto_sub -h 192.168.1.27 **-p** 8883 **-u** util1 -P mdputil1 **-t** temperature **--cafile** /path/to/ca.crt **--cert** /path/to/client.crt **--key** /path/to/client.key **--tls-version** tlsv1.2

Signification des options utilisées dans ces commandes

-p : port à utiliser pour la connexion. Dans ce cas, il s'agit de 8883 (port par défaut pour MQTT sur TLS).

--cafile : spécifie le chemin vers le fichier de certificat d'autorité de certification (CA) à utiliser pour vérifier le certificat du serveur.

- Il s'agit d'un certificat délivré par une autorité de certification reconnue qui atteste de l'identité d'une entité (généralement un serveur).
- Le certificat CA contient la clé publique de l'autorité de certification, qui est utilisée pour vérifier l'authenticité du certificat du serveur.
- Lorsqu'un client se connecte à un serveur, le serveur présente son certificat au client. Le client vérifie alors le certificat du serveur en utilisant le certificat CA pour s'assurer qu'il a été délivré par une autorité de confiance.

--cert : spécifie le chemin vers le fichier de certificat client à utiliser pour l'authentification auprès du serveur.

- Il s'agit d'un certificat délivré par une autorité de certification qui atteste de l'identité d'un client (généralement un utilisateur ou un appareil).
- Le certificat client contient la clé publique du client, qui est utilisée pour vérifier l'authenticité du client auprès du serveur.
- Lorsqu'un client se connecte à un serveur, le serveur peut demander au client de présenter son certificat. Le serveur vérifie alors le certificat du client en utilisant le certificat CA pour s'assurer qu'il a été délivré par une autorité de confiance.

En résumé, le certificat CA est utilisé pour vérifier l'authenticité du serveur, tandis que le certificat client est utilisé pour vérifier l'authenticité du client. Les deux types de certificats sont utilisés conjointement pour établir une connexion sécurisée et authentifiée entre un client et un serveur.

--key : spécifie le chemin vers le fichier de clé privée client à utiliser pour l'authentification auprès du serveur. Dans ce cas, il s'agit de /path/to/client.key.

--tls-version : spécifie la version de TLS à utiliser pour la connexion. Dans notre cas, il s'agit de tlsv1.2.