



24 JUIN 2022

SAE24

PROJET INTÉGRATIF

PARTIE COLLECTE

RAPPORT TECHNIQUE

GROUPE 1

BAHIR BOUDOUMA – ERNEST HALBOUT - JULES BRUTSCHY
Groupe C

Table des matières

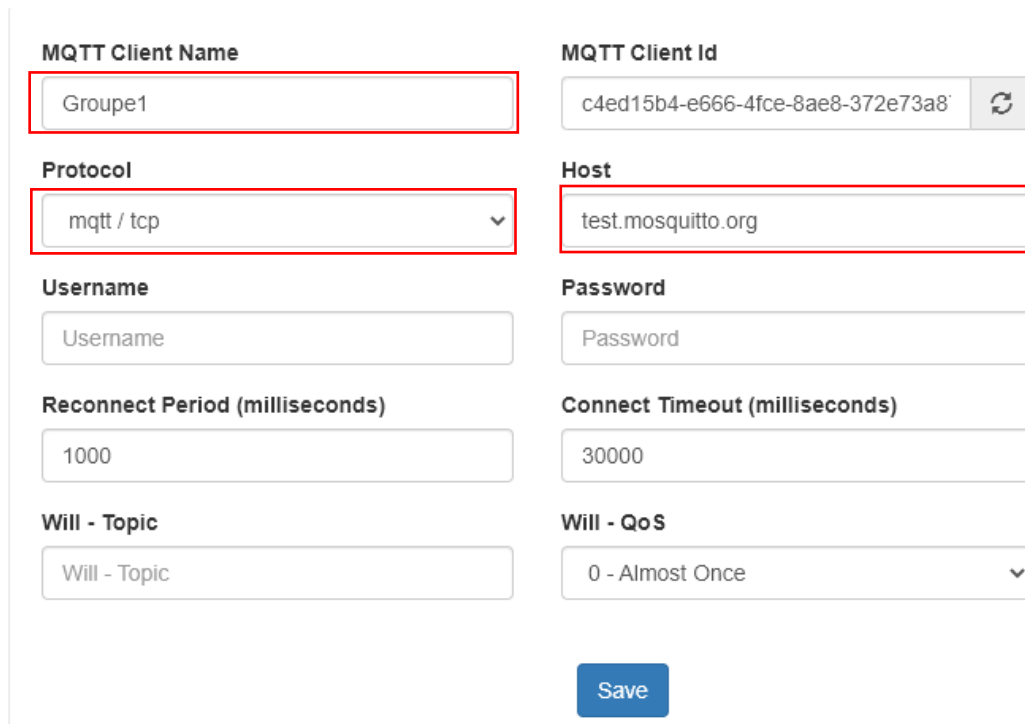
1. Introduction :.....	1
2. MQTTbox	2
3. Transfère des données dans la BDD	3

1. Introduction :

Dans le cadre de notre formation en BUT Réseaux et Télécommunications nous avons eu la possibilité de travailler sur un projet qui a eu pour but de mobiliser et valider toutes les compétences acquises au sein des ressources précédentes, cette SAÉ 24 nous a permis de renforcer nos différentes compétences techniques qui se traduisent par la création d'un site web dynamique avec un serveur local à l'aide différents éléments tel MQTT et python ainsi que la création d'une base de données externe vers qui on transfère des données recueillis sur MQTT. A l'issue de cette expérience nous avons acquis diverses aptitudes externes comme la recherche en autonomie de documentation des multiples procédures techniques requises pour l'avancée de notre projet.

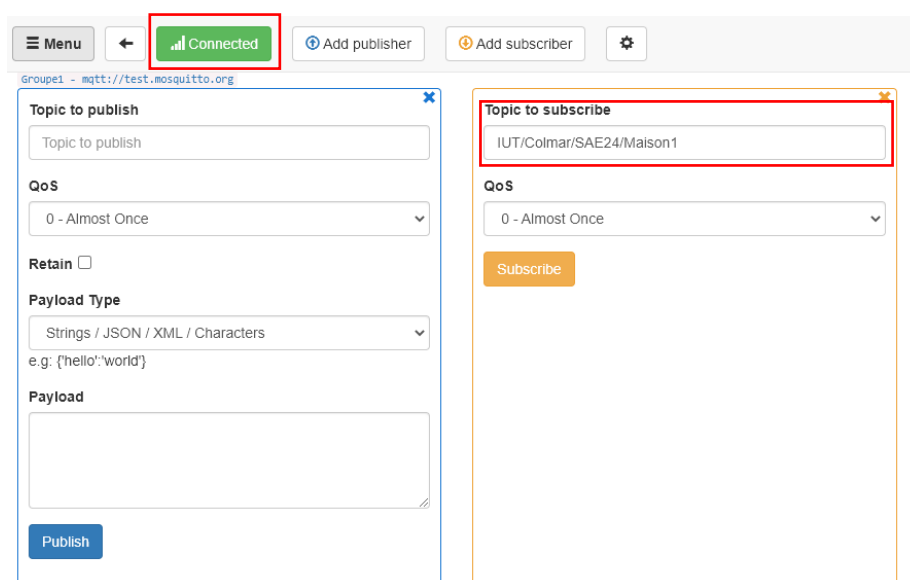
2. MQTTbox

On commence par l'installation de MQTTbox depuis navigateur web, une fois celui-ci installé nous mettons en place la configuration afin d'être sur le broker et le topic demandé pour pouvoir recevoir les données.



The image shows a web-based configuration form for an MQTT client. It is organized into two columns. The left column contains fields for 'MQTT Client Name' (with the value 'Groupe1'), 'Protocol' (a dropdown menu showing 'mqtt / tcp'), 'Username' (with the placeholder 'Username'), 'Reconnect Period (milliseconds)' (with the value '1000'), and 'Will - Topic' (with the placeholder 'Will - Topic'). The right column contains fields for 'MQTT Client Id' (with the value 'c4ed15b4-e666-4fce-8ae8-372e73a8' and a refresh icon), 'Host' (with the value 'test.mosquitto.org'), 'Password' (with the placeholder 'Password'), 'Connect Timeout (milliseconds)' (with the value '30000'), and 'Will - QoS' (a dropdown menu showing '0 - Almost Once'). At the bottom center of the form is a blue 'Save' button. Several input fields are highlighted with red rectangular boxes.

Une fois ces paramètres mis en place, nous sauvegardons les changements et on peut voir qu'on est bien connecté et on aura plus qu'à mettre en place le topic afin de recevoir les données.



The image shows the MQTT client interface after configuration. At the top, there is a status bar with a 'Menu' button, a 'Connected' status indicator (highlighted with a red box), and buttons for 'Add publisher', 'Add subscriber', and a settings gear icon. Below the status bar, the interface is split into two main panels. The left panel, titled 'Topic to publish', contains fields for 'Topic to publish', 'QoS' (dropdown showing '0 - Almost Once'), a 'Retain' checkbox, 'Payload Type' (dropdown showing 'Strings / JSON / XML / Characters'), and a 'Payload' text area. The right panel, titled 'Topic to subscribe', contains a 'Topic to subscribe' field (highlighted with a red box), a 'QoS' dropdown showing '0 - Almost Once', and a 'Subscribe' button. The address bar at the top shows 'Groupe1 - mqtt://test.mosquitto.org'.

3. Transfère des données dans la BDD

Ensuite on reprend le programme python mis à disposition par le prof afin de pouvoir se connecter au topic et recevoir les données sur la console python et les mettre directement dans la base de données.

Tout d'abord on se connecte à la base de données avec l'adresse ip de celle-ci ainsi que les identifiants :

```
try:
    db=_mysql.connect("localhost","root","toto", "groupe1B")
except OperationalError:
    db=_mysql.connect("localhost","root","toto")
    db.query("CREATE DATABASE groupe1B")
    db.query("USE groupe1B")
```

On modifie certains lignes afin de bien être connecté au broker demandé et au topic correspondant :

```
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    client.subscribe("IUT/Colmar/SAE24/Maison1")

client = mqtt.Client(client_id=f"client-grpl-{random.randint(1, 99999)}")
client.on_connect = on_connect
client.on_message = on_message

client.connect("test.mosquitto.org", port=1883, keepalive=60)

client.loop_forever()
```

Ensuite on met en place une fonction qui nous permet de transférer directement dans la base de données les informations du topic :

```
def on_message(client, userdata, msg):
    # On vérifie un peu si qqun ne s'amusse pas à envoyer n'importe quoi sur le broker
    try:
        correct = len(str(msg.payload)[2:-1].split(",")) == 5
    except Exception:
        correct = False

    if correct:
        payload = str(msg.payload)[2:-1].split(",")

        mac_addr= payload[0].split("=")[1]
        piece= payload[1].split("=")[1]
        dt = datetime.strptime(payload[2].split("=")[1] + " " + payload[3].split("=")[1],
                                '%d/%m/%Y %H:%M:%S')
        temp= payload[4].split("=")[1]

        try:
            db.query(f"INSERT INTO sensors (macaddr, piece) VALUES ('{mac_addr}', '{piece}')"
        except Exception:
            pass

        db.query(f"SELECT id FROM sensors WHERE macaddr='{mac_addr}'")
        id = int(db.store_result().fetch_row()[0][0])

        sql_data = f"INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES ({id},
        '{dt.strftime('%Y-%m-%d %H:%M:%S')}', {temp})"
        print(sql_data)

        reachable = True
        try:
            db.query(sql_data)
            reachable = True
        except Exception:
            reachable = False
            bak = open("backup.sql", "a")
            bak.write(f"{sql_data}\n")
            bak.close()

        if exists("backup.sql") and reachable:
            with open('backup.sql') as f:
                for line in f:
                    db.query(line)
                    remove('backup.sql')
```

Le programme vérifie le message afin qu'il ne prenne pas des messages faux du topic, ensuite on récupère le message envoyé par le topic, on vise les différents éléments et on formate l'élément afin de garder une chaîne de caractère cohérent à rentrer dans la base de données. Pour la première table on récupère l'ID et l'adresse mac que l'on insère dans la table sensor et dans la table sensor data on insère les éléments l'id du sensor, datetime et température.

On retrouve en sortie de console les données du topic et dans la base de données les informations recueillies du topic :

```
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1188, '2022-06-24 13:44:42', 3.21)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1187, '2022-06-24 13:44:47', 15.94)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1188, '2022-06-24 13:44:47', 13.3)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1187, '2022-06-24 13:44:52', 23.72)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1188, '2022-06-24 13:44:52', 14.12)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1187, '2022-06-24 13:44:57', 21.99)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1188, '2022-06-24 13:44:57', 15.92)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1187, '2022-06-24 13:45:02', 4.38)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1188, '2022-06-24 13:45:02', 10.84)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1187, '2022-06-24 13:45:07', 21.3)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1188, '2022-06-24 13:45:07', 20.2)
```

Unnamed

groupe1b

416,0 KiB

auth_group

32,0 KiB

auth_group_permission...

48,0 KiB

auth_permission

32,0 KiB

auth_user

32,0 KiB

auth_user_groups

48,0 KiB

auth_user_user_permi...

48,0 KiB

django_admin_log

48,0 KiB

django_content_type

32,0 KiB

django_migrations

16,0 KiB

django_session

16,0 KiB

sensors

32,0 KiB

sensors_data

32,0 KiB

information_schema

mysql

performance_schema

sys

groupe1b.sensors_data: 188

limite: 500 total (sensors)

Suivant

Tout montrer

Tri

Colonnes (4/4)

Filtre

id	sensor_id	datetime	temp
1 350	1 180	2022-06-24 13:44:27	17,56
1 357	1 187	2022-06-24 13:44:27	17,56
1 358	1 188	2022-06-24 13:44:27	11,49
1 359	1 187	2022-06-24 13:44:32	18,81
1 360	1 188	2022-06-24 13:44:32	21,06
1 361	1 187	2022-06-24 13:44:37	7,14
1 362	1 188	2022-06-24 13:44:37	5,3
1 363	1 187	2022-06-24 13:44:42	25,33
1 364	1 188	2022-06-24 13:44:42	3,21
1 365	1 187	2022-06-24 13:44:47	15,94
1 366	1 188	2022-06-24 13:44:47	13,3
1 367	1 187	2022-06-24 13:44:52	23,72
1 368	1 188	2022-06-24 13:44:52	14,12
1 369	1 187	2022-06-24 13:44:57	21,99
1 370	1 188	2022-06-24 13:44:57	15,92
1 371	1 187	2022-06-24 13:45:02	4,38
1 372	1 188	2022-06-24 13:45:02	10,84
1 373	1 187	2022-06-24 13:45:07	21,3
1 374	1 188	2022-06-24 13:45:07	20,2