Client MQTT sur Raspberry Pi en C++

Sommaire :

I - Présentation

Pour développer une application **cliente en C++**, on utilise la librairie **mosquitto-clients** et la librairie **libcurl**. La documentation des fonctions de la librairie est disponible aux adresses suivantes :

https://mosquitto.org/api/index/Functions.html
https://mosquitto.org/api/files/mosquitto-h.html.

On installe les librairies en exécutant les commandes suivantes :

```
apt-get install libmosquitto-dev
apt-get install libcurl-dev (ou apt-get install libcurl4-openssl-dev)
```

II - Abonnement MQTT

L'exemple de programme ci-dessous (fichier **main.cpp**), souscrit au topic **"sensor/temperature/1"** et affiche les valeurs publiées :

```
#include <iostream>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <mosquitto.h>

#define BROKER "193.168.147.50"
#define BROKER_PORT 1883
#define LOGIN "esp8266"
#define PASS "esp8266"
#define CLIENT_ID "client"

#define TOPIC_TEMPERATURE "sensor/temperature/#"
#define TOPIC_HUMIDITE "sensor/humidite/#"
#define TOPIC_ALL "sensor/#"
#define TOPIC_LED_ETAT "command/BC:DD:C2:2F:BD:13"
#define TOPIC_INFO "info"

using namespace std;

void publish(mosquitto *mosq, char const *topic, char const *payload) {
   mosquitto_publish(mosq, NULL, topic, strlen(payload), payload, 0, false);
}
```

```
void connect_callback(struct mosquitto *mosq, void *obj, int result) {
    // cout << "Resultat: " << result << endl << flush;
}

void message_callback(struct mosquitto *mosq, void *obj, const struct
mosquitto_message *message){
    cout << "Message: longueur " << message->payloadlen << " Valeur: " << (char*)
message->payload << " Topic: " << message->topic << endl << flush;
    if (strstr(message->topic, "temperature")) {
        char* clientID = strtok(message->topic, "/");
        cout << "clientId: " << clientID << endl << flush;
    }
}

int main(int argc, char *argv[]) {
    struct mosquitto *mosq;
    int rc = 0;
    char *lastWill = "Collecte KO!";
    mosq = mosquitto_new(CLIENT_ID, true, 0);

  if (mosq) {
        mosquitto_connect_callback_set(mosq, connect_callback);
        mosquitto_message_callback_set(mosq, message_callback);
        mosquitto_username_pw_set(mosq, LOGIN, PASS);
        mosquitto_will_set(mosq, TOPIC_INFO, strlen(lastWill), lastWill, 0, false);
        cout << "Attente connexion au broker: " <<  BROKER << endl << flush;
        while (mosquitto_connect(mosq, BROKER, BROKER_PORT, 60) !=
MOSQ_ERR_SUCCESS)                    {  };
        cout << "Connexion OK " <<  BROKER << endl << flush;
        mosquitto_subscribe(mosq, NULL, TOPIC_ALL, 0);
        mosquitto_subscribe(mosq, NULL, TOPIC_LED_ETAT, 0);

 while (1) {
            rc = mosquitto_loop(mosq, -1, 1);
            if (rc) {
                // cout << "Erreur de connexion " << endl << flush;
                sleep(10);
                mosquitto_reconnect(mosq);
            }
        }
        mosquitto_destroy(mosq);
    }
    mosquitto_lib_cleanup();
    return rc;
}
```

III - Publication MQTT

La fonction **mosquitto_loop()** permet de lancer l’application en boucle jusqu’à ce que le client se déconnecte correctement. Si l‘on veut **publier** cycliquement de l’information à l’aide du client on utilisera la fonction **mosquitto_loop_start()** puis on utilisera une boucle pour publier et enfin la fonction **mosquitto_loop_stop()** pour stopper le client. Donc on remplacera la partie de code :

```
rc = mosquitto_loop(mosq, -1, 1);
if (rc) {
        sleep(10);
        mosquitto_reconnect(mosq);
        }
```

par :

```
rc = mosquitto_loop_start(mosq);
if(rc != MOSQ_ERR_SUCCESS){
        cout << "Impossible de lancer la boucle" << endl << flush;
        exit(1);
        }
while (1) {
        cout << "Publication" << endl << flush;
        publish(mosq, TOPIC_INFO, "Collecte OK!");
        ……………
        usleep(10000000);    //10 s
}
```