



Ce TP propose la réalisation d'un objet connecté (node) sur un nano-ordinateur Raspberry Pi. Le « node » dispose en entrée d'un bouton « on-off », en sortie d'une LED.

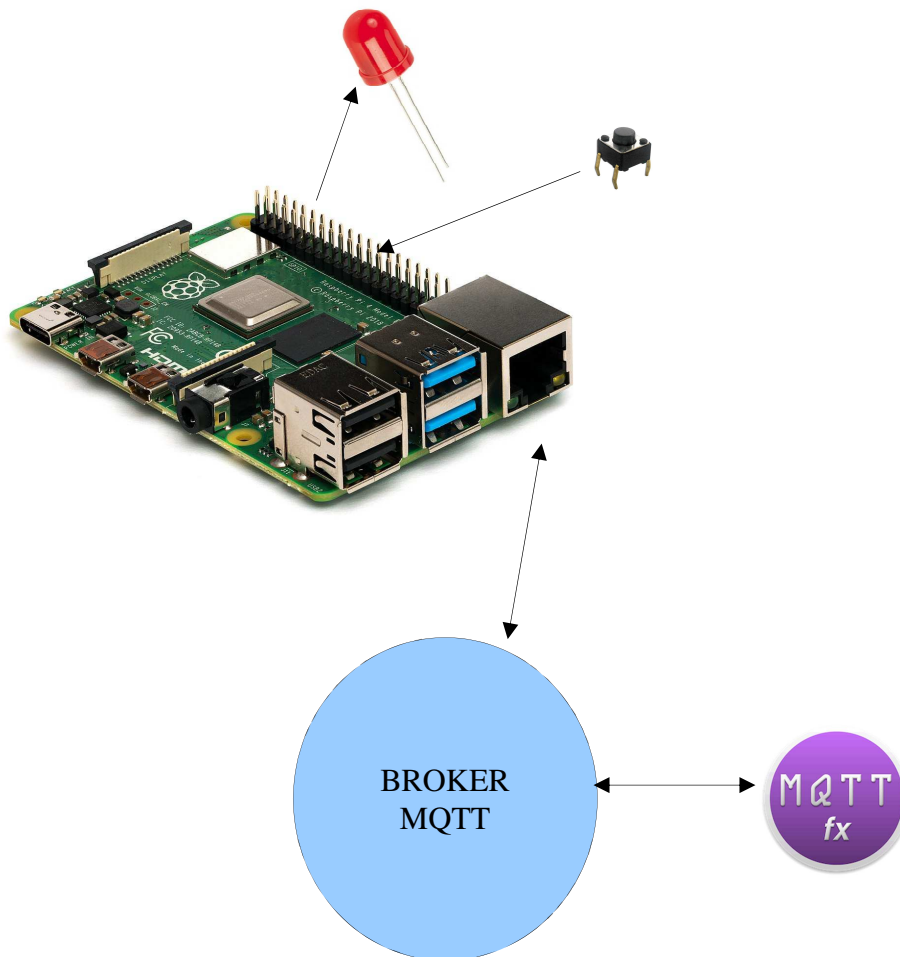
Les capteurs transmettent leur état vers un broker public avec le protocole MQTT. La liaison vers l'Internet est le WIFI.

Une liste non exhaustive des brokers publics est disponible ici :

https://github.com/mqtt/mqtt.github.io/wiki/public_brokers

La construction d'un client s'appuiera sur le projet eclipse-paho : <https://www.eclipse.org/paho/>.

La construction d'un broker s'appuiera sur le projet eclipse-mosquitto : <https://mosquitto.org/>.





Réalisation d'un client sur RASPBERRY PI

Installation du client MQTT:

Raspberry doit être connecté à Internet (WIFI ou Ethernet), la configuration peut se faire à distance via SSH, ou dans une console.

Aller dans le répertoire utilisateur et créer un répertoire de travail (ici cpp) :

```
cd ~
mkdir cpp
cd cpp
```

- MQTT utilise SSL pour le cryptage, installation de la bibliothèque SSL :

```
sudo apt install libssl-dev
```

- La bibliothèque C curl est utilisée par le client paho

```
sudo apt install libcurl4-openssl-dev
```

-Récupération de la bibliothèque client eclipse-paho

```
git clone https://github.com/eclipse/paho.mqtt.c.git
cd paho.mqtt.c.git
make
sudo make install
```

- Récupération de l'exemple client maqiatto

```
git clone https://github.com/cagdasdoner/maqiatto.git
```

L'exemple de client se trouve dans le répertoire maqiatto/linux-based, afin de limiter l'arborescence, on copie répertoire dans ~/cpp

```
cp -r ~/cpp/paho.mqtt.c/maqiatto/linux-based ~/cpp/mqttdemo
cd ~/cpp/mqttdemo
```

le répertoire contient :

credentials.h : header permettant de définir l'identifiant, le mot de passe , l'adresse du broker etc ;
 main.c : programme principal du node ;
 Makefile : le Makefile ;
 mqtt.c : bibliothèque MQTT ;
 mqtt.h : son header ;
 README.md.

Le broker utilisé sera test.mosquitto.org, pas d'identification ni de mot de passe.

Avec nano, modifier credentials.h comme suit :

```
#ifndef LINUX_CREDENTIALS
#define LINUX_CREDENTIALS

/* Provide MQTT broker credentials */
#define CLIENTID      "votreNom"
#define BROKER_ADDR   "test.mosquitto.org"
/* define USERNAME & PASSWORD as NULL for mosquitto brocker*/
#define USERNAME      NULL
#define PASSWORD      NULL

#define TOPIC         "leNomDeVotreTOPIC"

#endif /* LINUX_CREDENTIALS */
```



Personnaliser le TOPIC comme suit , éditer main.c :

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

#include "mqtt.h"
#include "credentials.h"

#define TOUT_TO_PUBLISH 5000

/* This program connects to https://www.maqiatto.com/
 * Periodically publishes test messages with your credentials.
 */

int main(int argc, char* argv[])
{
    MQTTBegin();

    MQTTSubscribe(TOPIC);

    while(1)
    {
        MQTTPublish(TOPIC, "mon premier TOPIC sur RPi");
        sleep(TOUT_TO_PUBLISH / 1000);
    };

    MQTTDisconnect();

    return 0;
}
```

Compilation du projet :

```
make          pour compiler ;
./mqttConnect pour exécuter.
```

Vérifier avec MQTT.fx par exemple que le programme envoie un message toutes les 5 secondes.



Réalisation du logiciel du "node"

À la fin du TP le node transmettra l'état du bouton. La LED sera contrôlée depuis un autre client.

Dans la suite du TP la bibliothèque wiringPi doit être installée.

Pour le vérifier entrer la commande `gpio -v`.

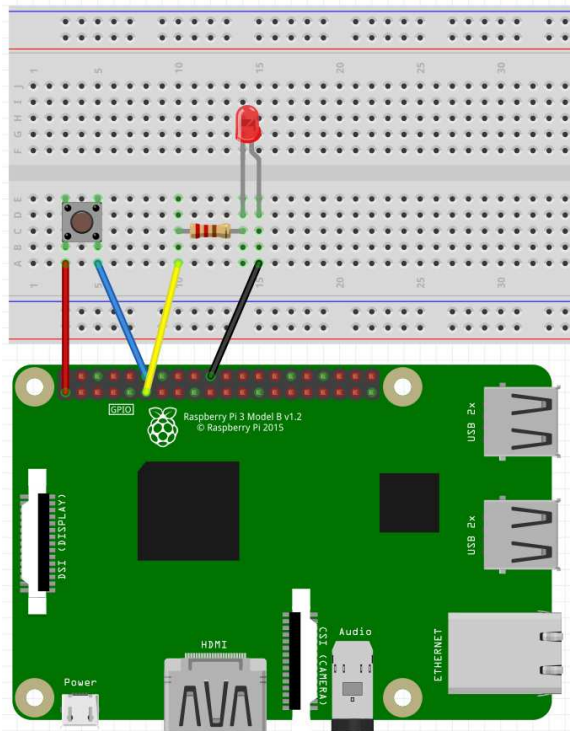
```
pi@raspberrypi:~/cpp/mqttdemo $ gpio -v
gpio version: 2.50
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
Type: Unknown17, Revision: 01, Memory: 0MB, Maker: Sony
* Device tree is enabled.
*--> Raspberry Pi 4 Model B Rev 1.1
* This Raspberry Pi supports user-level GPIO access.
```

En cas d'échec, installer wiringPi (<http://wiringpi.com>).

Le programme ci dessous émet sur le topic lors de l'appui sur un bouton, et allume une LED lors de la réception d'un message commençant par le code ASCII '1'.

Le bouton est câblé sur GPIO1 et la LED sur GPIO0 (Anode sur R220 et cathode sur GND).



Éditer makefile et ajouter la bibliothèque wiringPi

```
gcc -o mqttConnect main.c mqtt.c -l. -lcurl -lpaho-mqtt3c -lpthread -lwiringPi
```

Une fonction de décodage du message reçu est maintenant nécessaire.

Modifier mqtt.c comme suit

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

#include "MQTTClient.h"
#include "mqtt.h"
#include "credentials.h"
```



```
#define RET 0

void decode(char *mess);

MQTTClient client;

/* Subscribed MQTT topic listener function. */
int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message)
{
    if(message) {
        printf("Message arrived\n");
        printf("  -> topic: %s\n", topicName);
        printf("  -> message: ");
        printf("%s\n", (char*)message->payload);
        decode(message->payload);
    }
    MQTTClient_freeMessage(&message);
    MQTTClient_free(topicName);
    return 1;
}

void connlost(void *context, char *cause)
{
    printf("Connection lost\n");
    if (cause) printf("Reason is : %s\n", cause);
    MQTTDisconnect();
    /* Force to reconnect! */
    MQTTBegin();
}

void MQTTSubscribe(const char* topic)
{
    printf("Subscribing to topic <%s> for client <%s> using QoS%d\n\n",
        topic, CLIENTID, QOS);
    MQTTClient_subscribe(client, topic, QOS);
}

void MQTTPublish(const char* topic, char* message)
{
    MQTTClient_message pubmsg = MQTTClient_message_initializer;
    MQTTClient_deliveryToken token;
    pubmsg.payload = message;
    pubmsg.payloadlen = (int)strlen(message);
    pubmsg.qos = QOS;
    pubmsg.retained = RET;
    MQTTClient_publishMessage(client, topic, &pubmsg, &token);
    printf("Waiting for publication of message: %s\n\r",message);
    printf("  -> topic: %s  QOS: %d  RET: %d for client: %s\n\r", topic, QOS,RET,CLIENTID);
    int rc = MQTTClient_waitForCompletion(client, token, 1000);
    printf("Message with delivery token %d delivered\n", token);
}

void MQTTDisconnect()
{
    MQTTClient_disconnect(client, TIMEOUT);
    MQTTClient_destroy(&client);
}

void MQTTBegin()
{
    int rc = -1;
    printf("Initializing MQTT for <%s> broker\n",BROKER_ADDR);
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    conn_opts.keepAliveInterval = KEEP_ALIVE;
    conn_opts.cleansession = 1;
    conn_opts.username = USERNAME;
    conn_opts.password = PASSWORD;
    MQTTClient_create(&client, BROKER_ADDR, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);

    /* Set connection, subscribe and publish callbacks. */
    MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, NULL);

    while ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS)
    {
        printf("Failed to connect, return code %d\n", rc);
        sleep(TIMEOUT / 1000);
    }
    printf("Connexion established\n");
}
```



```
}
```

et main.c

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <wiringPi.h>
#include "mqtt.h"
#include "credentials.h"

#define led 0
#define btn 1

void decode(char *mess)
{
    if (*mess=='1') {
        digitalWrite(led,HIGH);
        printf("LED allumee\n\r");
    }
    else {
        digitalWrite(led,LOW);
        printf("LED eteinte\n\r");
    }
}

int main(int argc, char* argv[])
{
    wiringPiSetup();
    pinMode(led,OUTPUT);
    pinMode(btn,INPUT);

    MQTTBegin();
    MQTTSubscribe(TOPIC);

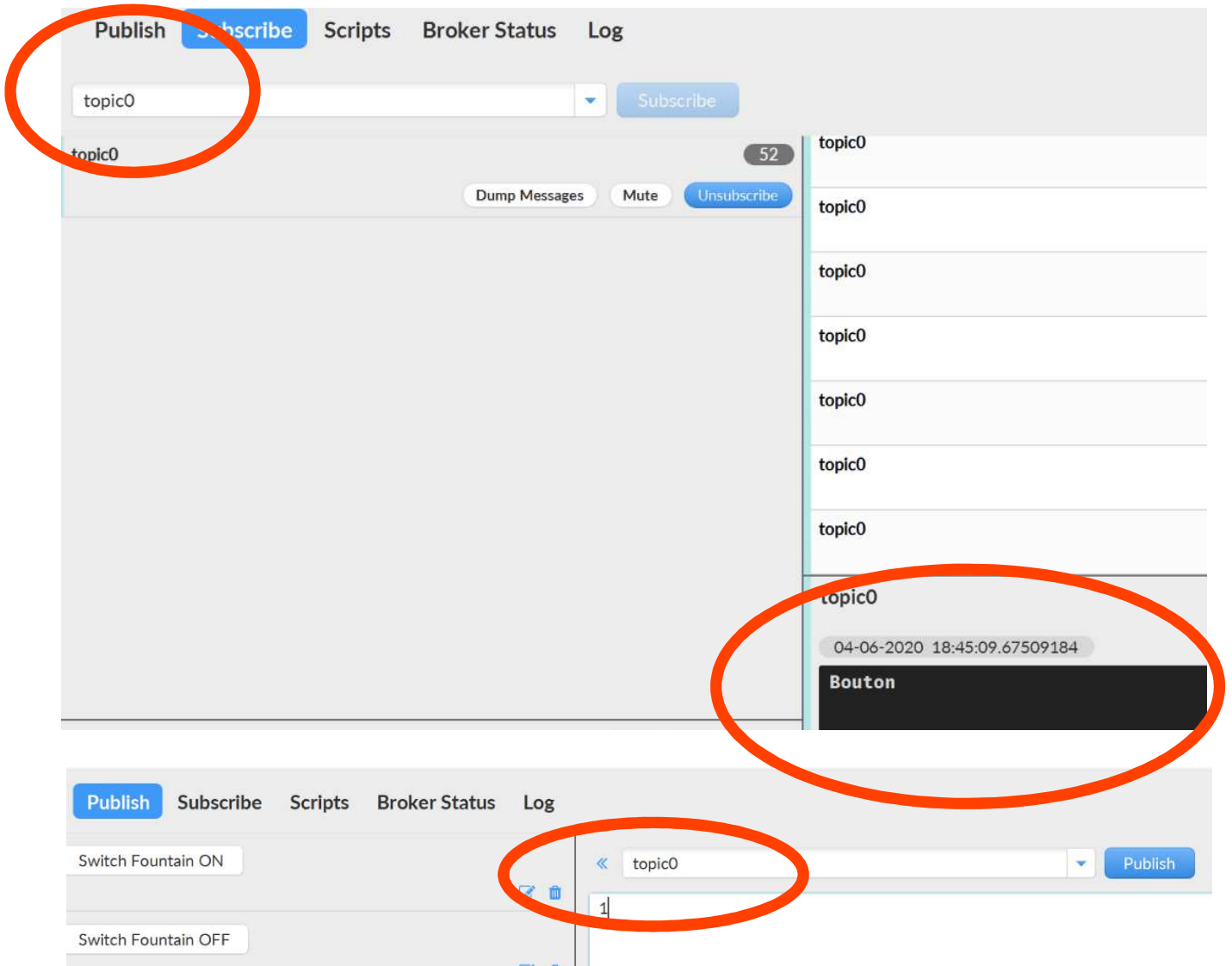
    printf("Le bouton sur GPIO%d provoque l'emission du topic -> %s\n",btn,TOPIC);
    while(1)
    {
        while(digitalRead(btn)==1);
        MQTTPublish(TOPIC, "Bouton");
        while(digitalRead(btn)==0);
    }

    MQTTDisconnect();
    return 0;
}
```

Compilation :

```
make
./mqttConnect
```

Vérifier le fonctionnement avec MQTT.fx.
L'émission d'un message contenant '1' sur le topic0 allume la LED.
L'émission d'un message contenant '0' sur le topic0 éteint la LED.



Exercice :

Afficheur 7 segments :

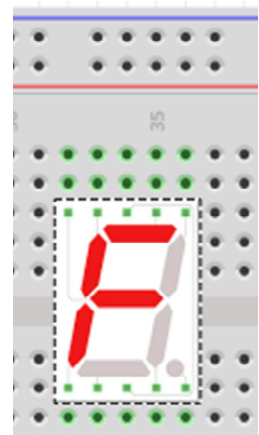
https://fr.wikipedia.org/wiki/Afficheur_7_segments

<https://sitelec.org/cours/abati/aff7seg.htm>

Câbler l'afficheur 7 segments proposé sur les GPIO de votre choix.

Le programme attend un message MQTT contenant un chiffre compris entre 0 et 9 qui sera alors visible sur l'afficheur 7 segments.

La page suivante propose un exemple sur Rpi.





```
// test AFF7SEG cathode commune sur RPi
#include <stdio.h>
#include <wiringPi.h>
#define anoA 17
#define anoB 22
#define anoC 19
#define anoD 6
#define anoE 5
#define anoF 27
#define anoG 26
#define anoDP 13
#define cathol 12

const int code7seg[]={0xFC,0x60,0xDA,0xF2,0x66,0xB6,0xBE,0xE0,0xFE,0xF6};

void aff7seg(int aff,int n)
{
    digitalWrite (cathol, HIGH) ;
    digitalWrite (anoA, n&0x80) ;
    digitalWrite (anoB, n&0x40) ;
    digitalWrite (anoC, n&0x20) ;
    digitalWrite (anoD, n&0x10) ;
    digitalWrite (anoE, n&0x08) ;
    digitalWrite (anoF, n&0x04) ;
    digitalWrite (anoG, n&0x02) ;
    digitalWrite (anoDP, n&0x01) ;
    digitalWrite (cathol, LOW) ;
}

void init7seg(void)
{
    pinMode (anoA, OUTPUT) ; // segment A
    pinMode (anoB, OUTPUT) ; // segment A
    pinMode (anoC, OUTPUT) ; // segment A
    pinMode (anoD, OUTPUT) ; // segment A
    pinMode (anoE, OUTPUT) ; // segment A
    pinMode (anoF, OUTPUT) ; // segment A
    pinMode (anoG, OUTPUT) ; // segment A
    pinMode (anoDP, OUTPUT) ; // segment A
    pinMode (cathol, OUTPUT) ; // afficheur 4
    digitalWrite (cathol, HIGH) ;
}

int main (void)
{
    int i;
    wiringPiSetupGpio () ;
    init7seg();
    while(1)
    {
        for(i=0;i<10;i++)
        {
            aff7seg(1,code7seg[i]);
            printf("%d \n",i);
            delay (500) ;
        }
    }
    return 0 ;
}
```