

PROTOCOLE MQTT sur Raspberry Pi

Intentions pédagogiques :

Ce TP propose de réaliser un thermomètre sur Raspberry Pi avec émission vers un broker disposant d'une API graphique.

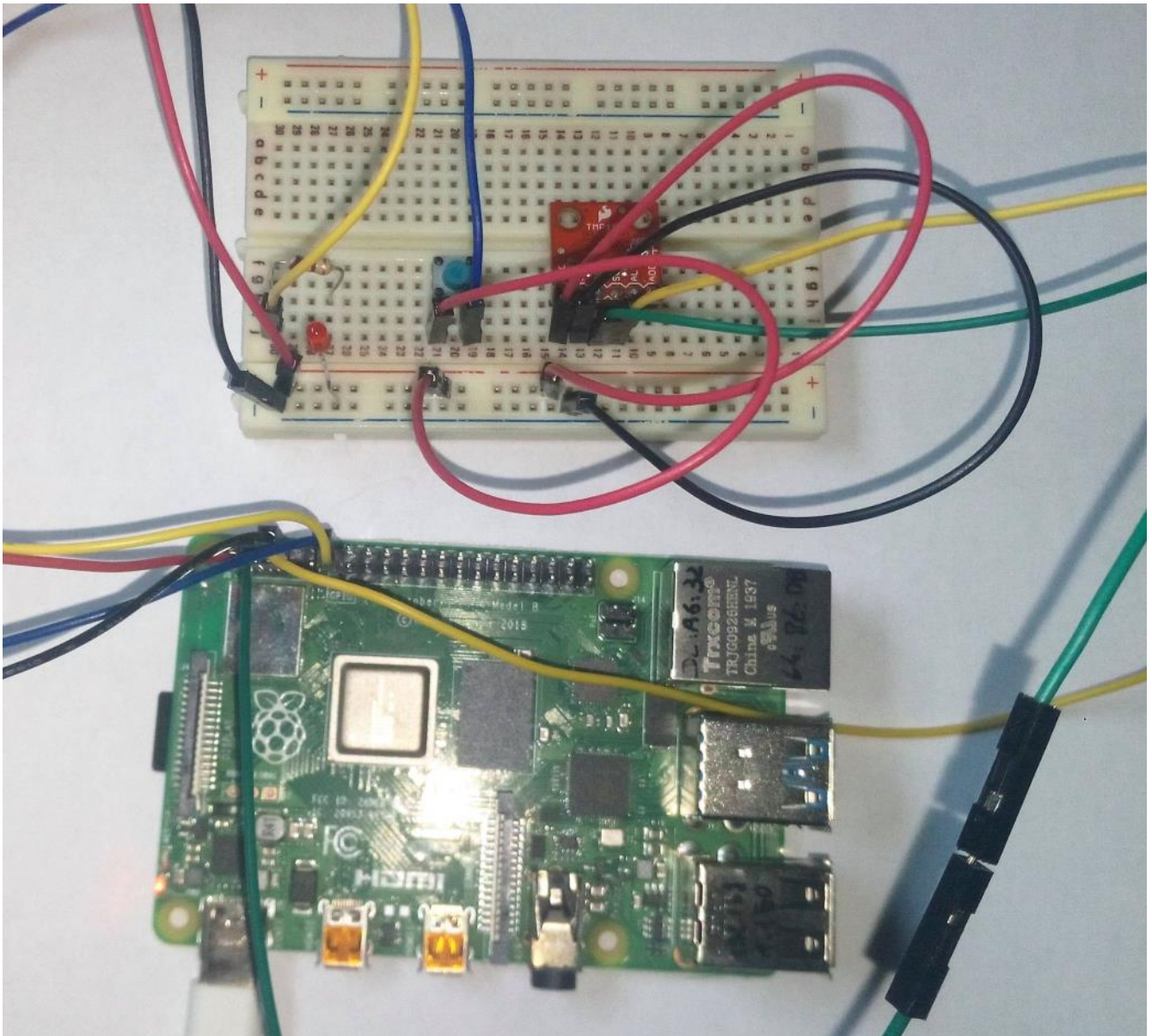
La température est acquise avec un script Python dont la sortie est redirigée vers une programme exécutable créée en C++ qui transmet les données vers un broker MQTT.

L'affichage graphique se fera avec [thethinspeak](http://thethinspeak.com).

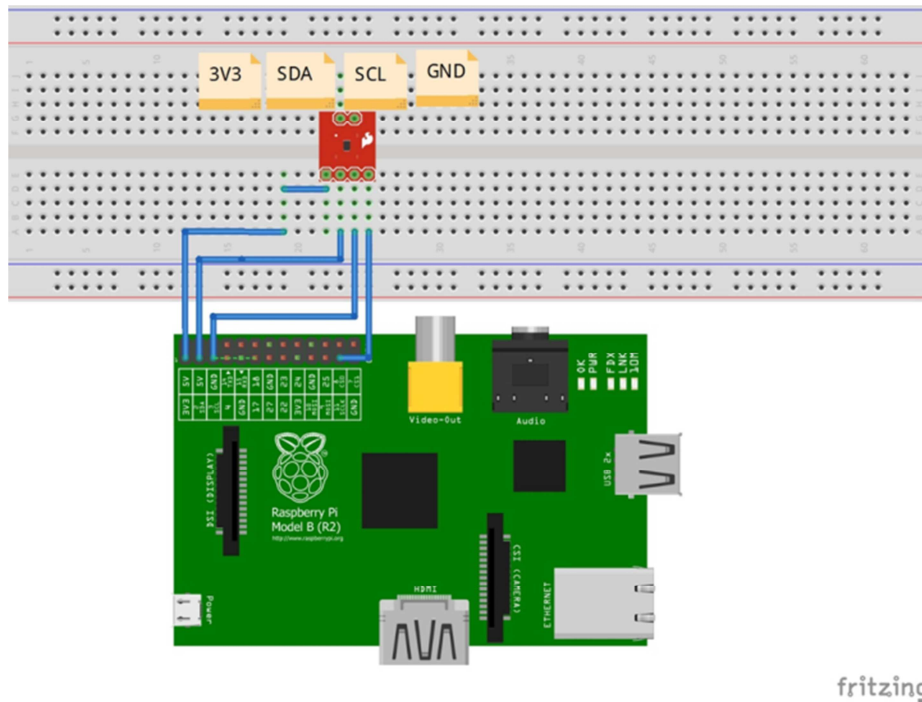
Le capteur sera un TMP102 sur breakout Sparkfun :

<https://www.ti.com/product/TMP102>

<https://www.sparkfun.com/products/13314>



1 Montage



La bibliothèque i2ctools est préinstallée sur RPi, la doc est ici : <https://www.mankier.com/package/i2c-tools>
Le bus I2C doit être activé: `sudo raspi-config`.

Vérification de la connexion : `sudo i2cdetect -y 1`. Le périphérique i2c tmp102 apparaît à l'adresse 0x48.

```

pi@raspberrypi: ~/python
pi@raspberrypi ~/python $ sudo i2cdetect -y 1
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- 48 -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- --
pi@raspberrypi ~/python $
    
```

`sudo i2cget -y 1 0x48 0x00` retourne la température en hexadécimal.

Exemple : Si la commande précédente retourne 0x2017 :

Le tmp102 retourne la valeur de son ADC avec en premier les poids faibles puis les poids forts , après permutation on obtient 0x1720.

Le 0 des unités n'est pas significatif, après décimation on obtient 0x172 soit 370 en décimal.

La résolution du TMP102 est de 0,0625 degrés, la température est donc de $370 \times 0,0625 = 23,125^{\circ}\text{C}$.

Remarque : si la résolution est de $0,0625^{\circ}\text{C}$, la précision absolue est de $\pm 0,5^{\circ}\text{C}$ sans calibration.


2 The thingspeak

The thingspeak est un broker LORA / MQTT permettant l'affichage graphique des données reçues.

Sur <https://thingspeak.com> créer un compte et l'activer.

Éditer votre profil, et générer en particulier une clé "MQTT API key".

<https://thingspeak.com/account/profile>


[Channels ▾](#)
[Apps ▾](#)
[Support ▾](#)

My Profile




MathWorks Account Settings

MathWorks Account Email	christian.dupaty@ac-aix-marseille.fr
User ID	christian.dupaty
Password	*****

[Edit MathWorks Account Settings](#)

[Edit MathWorks Community Information](#)

ThingSpeak Settings

Time Zone	UTC	Edit
User API Key		↻
MQTT API Key		↻
Alerts API Key		↻

PROTOCOLE MQTT sur Raspberry Pi

Dans "channels" créer un nouveau "chanel".

<https://thingspeak.com/channels/new>

ThingSpeak™ Channels Apps Support

New Channel

Name

Description

Field 1 ☒

Field 2 ☐

Cliquer "save channel" en bas de la page.

Dans l'onglet "sharing" partager le graph avec tout le monde.

ThingSpeak™ Channels Apps Support

temperature

Channel ID: **1076599**
Author: christian.dupaty
Access: Private

temperature de mon bure.
TMP102 associé à Raspber
MQTT

Private View Public View Channel Settings Sharing API Keys

Channel Sharing Settings

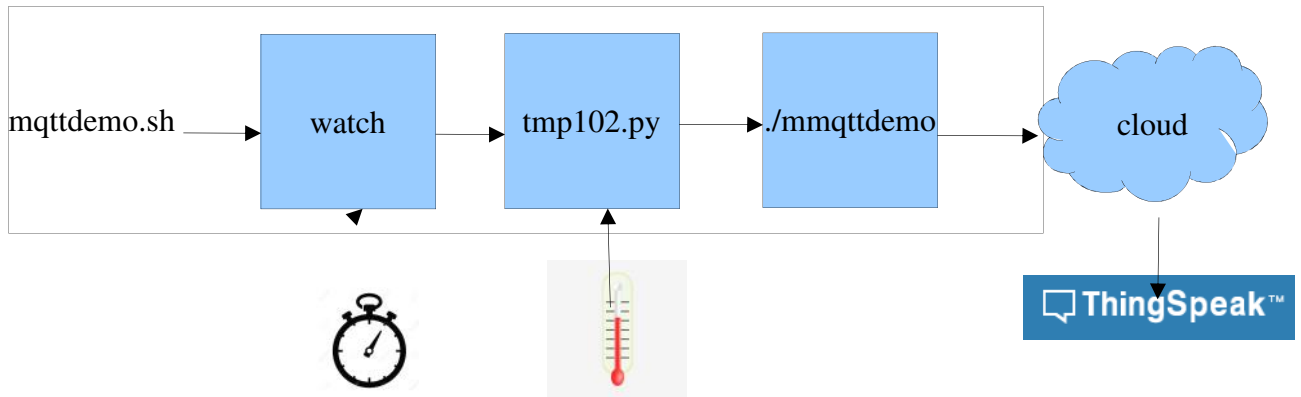
☐ Keep channel view private
☒ Share channel view with everyone
☐ Share channel view only with the following users:

Email Address

C'est terminé...

3 Acquisition et transmission de la température

Un script bash lance l'utilitaire Linux watch qui appelle périodiquement un script python qui récupère la température du TMP102. La sortie de ce script est redirigée vers un exécutable qui publie la température avec MQTT sur le broker thethingspeak. Ce dernier dispose de capacités graphiques et affiche l'évolution de la température.



Le programme tmp102.py affiche la température mesurée par le capteur

```
#!/usr/bin/env python
# pour vérifier la présence d'un périphérique i2c :
#sudo i2cdetect -y 1
import smbus
import time
bus_pi = smbus.SMBus(1)
# I2C address for TMP102
addr = 0x48
try:
    x = bus_pi.read_word_data(addr,0)
    msb=x>>8
    lsb=x&0x00FF
    wtemp= ((lsb<<8) | msb)>>4
    print (wtemp*0.0625)
except:
    print ('...erreur...')
```

```
pi@raspberrypi: ~/cpp/mqttdemo
pi@raspberrypi:~/cpp/mqttdemo $ python3 tmp102.py
26.125
pi@raspberrypi:~/cpp/mqttdemo $
```

Transmission de la température au broker avec le protocole MQTT.

La réalisation en C++ d'un client MQTT sur Raspberry Pi est décrite dans le TP "3 client MQTT sur Rpi.pdf" paragraphe "**Réalisation d'un client sur RASPBERRY PI**".



Après importation de la bibliothèque MQTT et de la démo du client maquiatto.

Récupérer les clés d'écriture et de lecture des topics.

https://thingspeak.com/channels/1076599/api_keys

Channels ▾ Apps ▾ Support ▾

temperature

Channel ID: 1076599
Author: christian.dupaty
Access: Public

temperature de mon bure:
TMP102 associé à Raspber
MQTT

Private View Public View Channel Settings Sharing API Keys

Write API Key

Key

Generate New Write API Key

Read API Keys

Key

La password MQTT est dans le profil de l'utilisateur.

MQTT API Key

Compléter credentials.h comme suit :

```
#ifndef LINUX_CREDENTIALS
#define LINUX_CREDENTIALS

/**/
#define CLIENTID      "christian.dupaty"
#define BROKER_ADDR   "mqtt.thingspeak.com"
#define USERNAME      "christian.dupaty"
#define PASSWORD      "XXXXXXXXXXXXXXXXXXXX"

/* Provide topic as it is denoted in your topic list.
 * For example mine is : cadominna@gmail.com/topic1
 * To add topics, see https://www.maquiatto.com/configure
 */
#define TOPICP        "channels/344721/publish/fields/field1/XXXXXXXXXXXXXXXXXXXX"
#define TOPICS         "channels/344721/subscribe/json/XXXXXXXXXXXXXXXXXXXX"

#endif /* LINUX_CREDENTIALS */
```

modifier main.c comme suit :

Le programme se connecte au broker, attend sur le flux d'entrée par défaut (stdin), un nombre, le publie et ferme la connexion.

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

void decode(char *mess)
{
    printf("Reception message, aucune action\n\n");
}

int main(int argc, char* argv[])
{
    char buff[20];
    MQTTBegin();
    MQTTSubscribe(TOPICS);
    printf("Attente de mesure ...\n\n");
    scanf("%s",buff); // lecture temperature depuis sortie tmp102.py
    printf("Publication de la temperature -> %s°C\n\n",buff);
    MQTTPublish(TOPICP, buff);
    MQTTDisconnect();
    printf("Deconnexion OK\n\n");
    return 0;
}
```

Exécuter make pour compiler le programme.

Liaison sortie python vers exécutable

Linux permet de transmettre la sortie d'un programme à l'entrée d'un autre par un pipe (alt-gr + 6).

```
python3 tmp102.py | ./mqttConnect
```

Automatisation de la tâche

La commande Linux watch permet d'exécuter un programme périodiquement.

<http://www.man-linux-magique.net/man1/watch.html>

Essayer -> `watch -d -n 1 "python3 tmp102.py | ./mqttConnect"`

Exécutera la commande entre "" toutes les secondes.

Plaçons l'ensemble dans un fichier mqttdemo.sh :

```
#!/bin/bash
if [ $# -eq 0 ]
then set $1 '1'
echo "sans argument la tempo est de 1.0s, syntaxe ./demomqtt.sh 1.1, tapez entrer pour continuer"
read a
fi
watch -d -n $1 "python3 tmp102.py | ./mqttConnect"
```

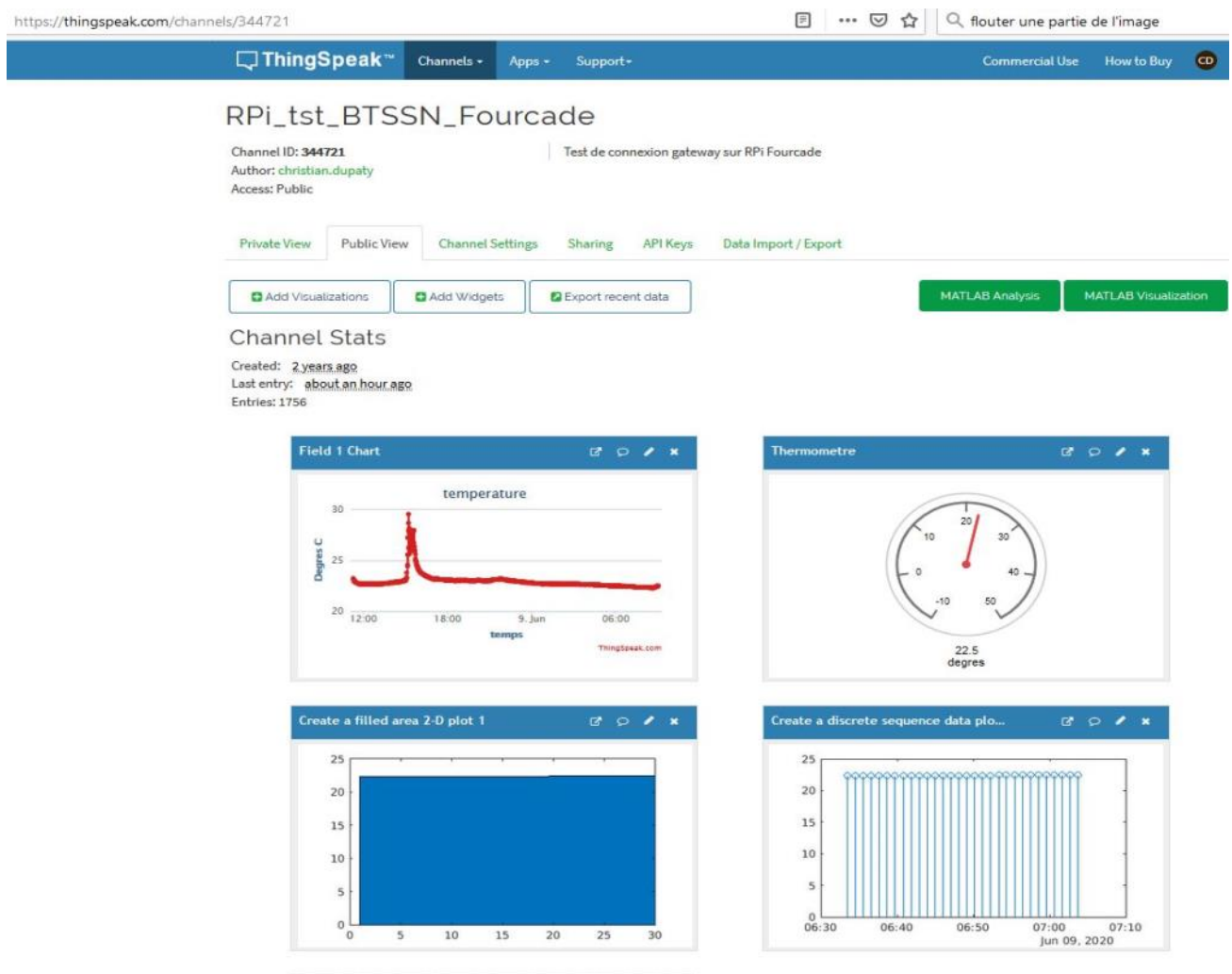
Pour le lancer : `./mqttdemo.sh` (pour le rendre exécutable : `chmod +x mqttdemo.sh`)

PROTOCOLE MQTT sur Raspberry Pi

L'exécution produit :

```
pi@raspberrypi: ~/cpp/mqttdemo
Every 5,0s: python3 tmp102.py | ./mqttConnect
Initializing MQTT for <mqtt.thingspeak.com> broker
Connexion established
Subscribing to topic <channels/344721/subscribe/json/<redacted>> for client <christian.dupaty> using QoS0
Attente de mesure ...
Publication de la temperature -> 27.1875°C
Waiting for publication of message: 27.1875
-> topic: channels/344721/publish/fields/field1/<redacted> QOS: 0 RET: 0 for client: christian.dupaty
Message with delivery token 68984 delivered
Deconnexion OK
```

Sur thethingspeak le graphique évolue en fonction de la température :



Exercice :

Le TMP102 est un simple capteur de température, le DHT11 est un capteur de température et d'humidité souvent utilisé dans les stations météo.

<https://components101.com/dht11-temperature-sensor>

Vous trouvez facilement sur Internet de nombreux exemples d'implantation du DHT11 sur Raspberry Pi en C++ et en Python.

Remplacer le TMP102 par un DHT11 et visualiser sur thethingspeak la température et l'humidité.