Université Tunis El-Manar	Année Universitaire : 2021-2022
Faculté des Sciences de Tunis	Module : Conception des objets connectés
Section : LIOT3	Enseignants : C.A. ABID & I. Benabdallah

TP1 : Prise en main de la Raspberry Pi

Partie I : Installation de Raspberry OS Lite et connexion distante SSH à partir d'un PC Linux

On se propose d'installer la dernière version de la distribution officielle Raspberry OS. La système sera installé sur une carte SD.

- Télécharger le fichier image du système d'exploitation à installer sur la Raspberry Pi à partir de **https://www.raspberrypi.org/software/operating-systems/#raspberry-pi-os-32-bit**
Veuillez choisir Raspberry OS Lite.

- Placer la carte SD dans le lecteur de cartes, puis à l'aide de la commande `lsblk`, déterminer le nom des partitions de la carte SD

- Il faut démonter les partitions présentes sur la carte en utilisant la commande `umount`.
Par exemple : `umount /dev/sdX1` avec `/dev/sdX1` est une partition de la carte `/dev/sdX`.

- Maintenant, nous copions les fichiers de la distribution pour le Raspberry Pi sur la carte SD. Nous utilisons la commande `dd` pour reproduire des zones de disque qui ne font pas partie d'un système de fichier : secteur de démarrage (le MBR), tables de partition, etc.

```
dd bs=4M if=2021-03-04-raspios-buster-armhf-lite.img
of=/dev/sdX status=progress conv=fsync
```

L'option bs permet de spécifier la taille de chaque bloc. L'option if permet de préciser le chemin du fichier image du système d'exploitation à installer . L'option of spécifie le nom de la carte SD sur laquelle le système d'exploitation va être créé. Il faut bien rentrer le nom de la carte et pas d'une partition, c'est à dire enlever le chiffre au bout du nom des partitions (par exemple `sdc`, pas `sdc1` ou `mmcblk0` mais pas `mmcblk0p1`).

- Une fois la création du système sur la carte est effectuée, nous continuons par configurer la connexion au réseau Wifi et l'activation du serveur SSH. Cela nous permettra de se connecter à la carte à travers le protocole SSH.

Dans la première partition, dans le répertoire `/boot` :

- Créer une fichier `wpa_supplicant.conf` contenant le code suivant :

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=FR

network={
ssid="SSID"
psk="password"
key_mgmt=WPA-PSK
```

```
}
```

Veuillez remplacer le SSID et password par les paramètres du réseau Wifi auquel vous êtes connectés.

- Créer un fichier vide appelé ssh. Ce fichier permet d'indiquer à l'OS que le protocole SSH doit être activé.

- Lancez la commande ci-après afin de vider le cache d'écriture et permettre l'éjection sans risque de la carte SD :

```
sudo sync
```

- Quand la commande vous rend la main, sortez la carte du lecteur de votre ordinateur et l'insérez la dans la Raspberry Pi.

Normalement, si la configuration est effectuée correctement, après être connectée à l'alimentation, la carte Raspberry PI démarrera avec la distribution Raspberry OS, tout en autorisant la connexion au système via le protocole SSH avec le login pi et le mot de passe par défaut raspberry. Nous devons pour cela retrouver l'adresse IP de la carte sur le réseau. À partir de Linux, il est possible d'utiliser la commande `nmap` :

```
sudo nmap -sS -p 22 x.x.x.0/24
```

Sinon, on peut aussi consulter le contenu du fichier `/var/log/daemon.log` sur la carte SD de la RPi (bien sûr, il faut la retirer de la carte RPi et l'insérer dans le lecteur de PC) pour retrouver la dernière adresse IP utilisée.

Une fois l'adresse IP est identifiée , vous pouvez vous connecter à la carte à travers le protocole SSH :

```
$ ssh pi@<adresse IP de la carte>
```

Partie II : Compilation d'un programme C++

On se propose d'écrire des programmes C++, les compiler sur la carte RPi. On utilise principalement la bibliothèque C pigpi.

- Installation de la bibliothèque pigpio

Taper les commandes suivantes à partir du shell de la RPi :

```
wget https://github.com/joan2937/pigpio/archive/master.zip
unzip master.zip
cd pigpio-master
make
sudo make install
```

S'il y a un problème à cause de python, il faut installer les paquets suivants :

```
sudo apt install python-setuptools python3-setuptools
```

- Édition, Compilation et exécution d'un programme C++

- Utiliser l'éditeur nano pour éditer un programme C++. À partir de la ligne de commande, lancer la commande suivante :

```
$ nano HelloRaspberry.cpp
```

- Saisir le programme suivant :

```
#include <iostream>


int main() {
     std::cout << "!!!Hello Raspberry Pi!!!" << std::endl;
     return 0;
```

```
}
```

Pour compiler le programme, taper dans la ligne de commande suivante qui va produire un fichier exécutable nommé MonProgramme :

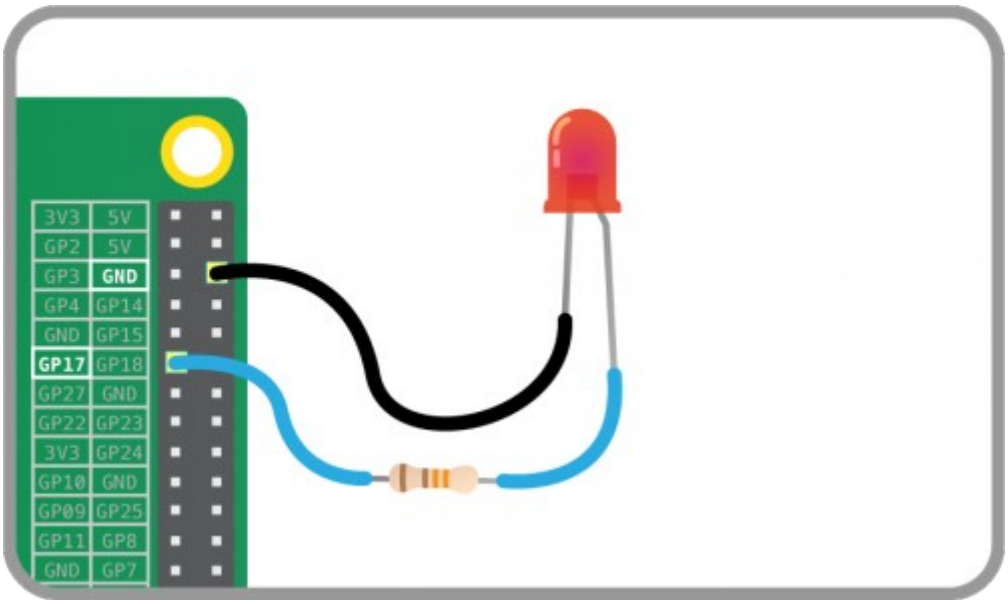
```
$ g++ MonProgramme.cpp -o MonProgramme
```

Pour lancer le programme, il suffit de taper son nom dans ligne de commande :

```
$ ./MonProgramme
```

Maintenant, on écrira un programme permettant de flasher une diode LED en utilisant la bibliothèque `pigpio`.

Réaliser le montage suivant :



Utiliser l'éditeur `nano` pour saisir le programme C++ `FlasherLED.cpp` :

```
#include <iostream>
#include <unistd.h>
#include <pigpio.h>

#define O_LED 17    // Using GPIO17

int main(int argc, char *argv[]) {
     if (gpioInitialise()<0) {
          std::cout<<"Échec d'initialisation...\n";
          exit(-1);
     }
     gpioSetMode(O_LED, PI_OUTPUT);
     while (1) {
          gpioWrite(O_LED, 1);
```

```
            sleep(1);
            gpioWrite(O_LED, 0);
            sleep(1);
        }
        gpioTerminate();
        return 0;
}
```

Réaliser la compilation du programme pour produire le fichier exécutable FlasherLED. Ici, on utilisera le l'option -lpigpio pour indiquer à l'éditeur des liensl'utilisation de la bibliohtèque pigpio.

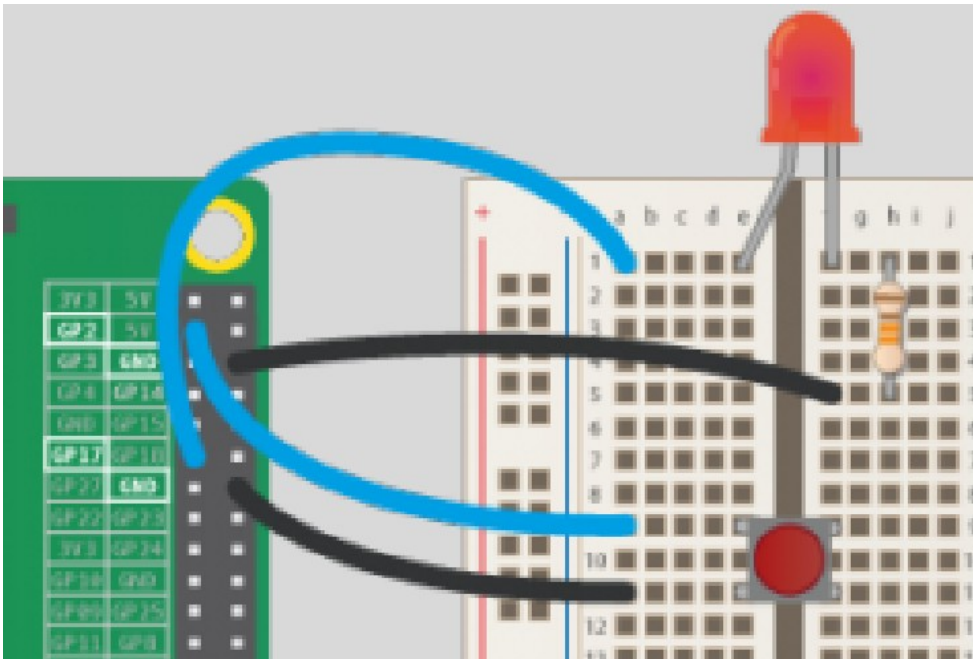
```
$ g++ FlasherLED.cpp -o FlasherLED -lpigpio
```

Un programme utilisant la bibliothèque `pigpio` doit être exécuté avec les privilèges du superutilisateur root :

```
$ sudo ./FlasherLED
```

Partie III : Exercice

Soit le montage suivant :



1. Écrire un programme C++ permettant de changer l'état de la diode LED à chaque appuie du bouton poussoir.

2. Écrire un programme C++ qui permet d'activer ou de désactiver le flashage d'une diode LED chaque fois qu'on appuie sur le bouton poussoir.
Expliquer pourquoi dans certains cas le programme ne réagit pas à l'appuie du bouton poussoir ?