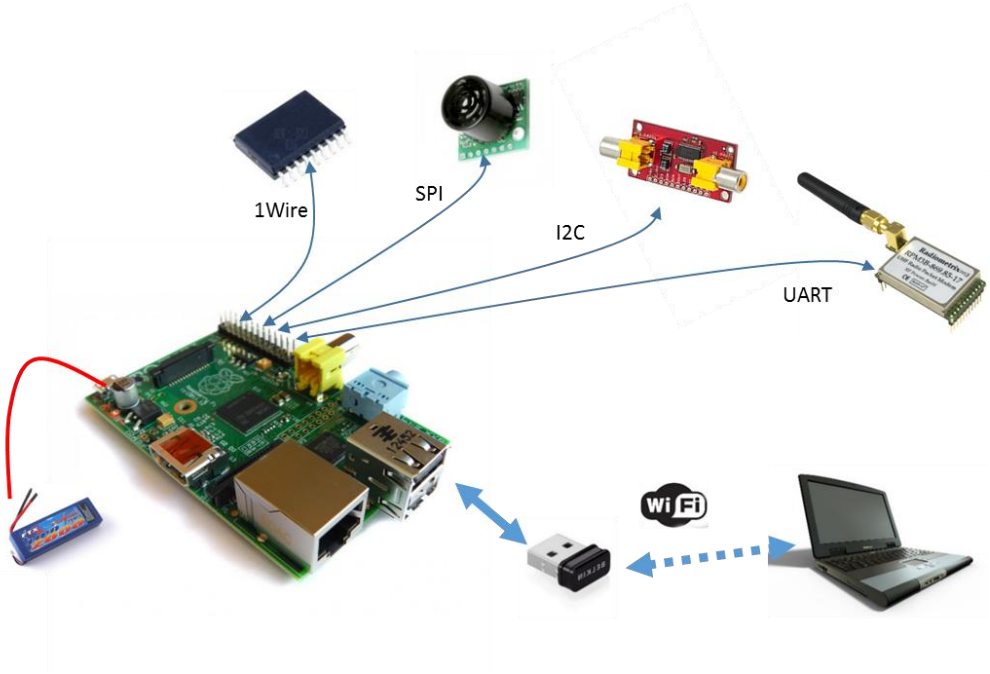
TP 2 : Prise main d'un système embarqué Rapsberry Pi



1 Introduction

Les ordinateurs embarqués sous le système d'exploitation Linux sont massivement présents dans les technologies modernes (transports, multimédia, téléphonie mobile, appareils photos …).

L'ordinateur Raspberry PI constitue un support d'apprentissage performant, très bon marché et disposant d'une forte communauté sur le net. Il possède des entrées/sorties puissantes permettant une connexion avec le monde physique par l'intermédiaire de capteurs et d'actionneurs.

L'objectif de ce TP est de réaliser une rapide prise en main d'un système embarqué au travers d'un ordinateur Raspberry PI et d'effectuer un tour d'horizon des pratiques de mise en œuvre et de développement.

Les connaissances acquises devront permettre la réalisation d'une application embarquée de supervision d'un système matériel simplifié.

2 Evaluation

L'évaluation portera sur :

- Un compte rendu détaillé explicitant les différentes opérations menées durant le TP.
  Pensez à prendre des copies d'écran pour illustrer ces opérations.
- la réalisation d'une application embarquée de supervision d'un système matériel simplifié.
  Le cahier des charges sera détaillé plus loin dans se document.

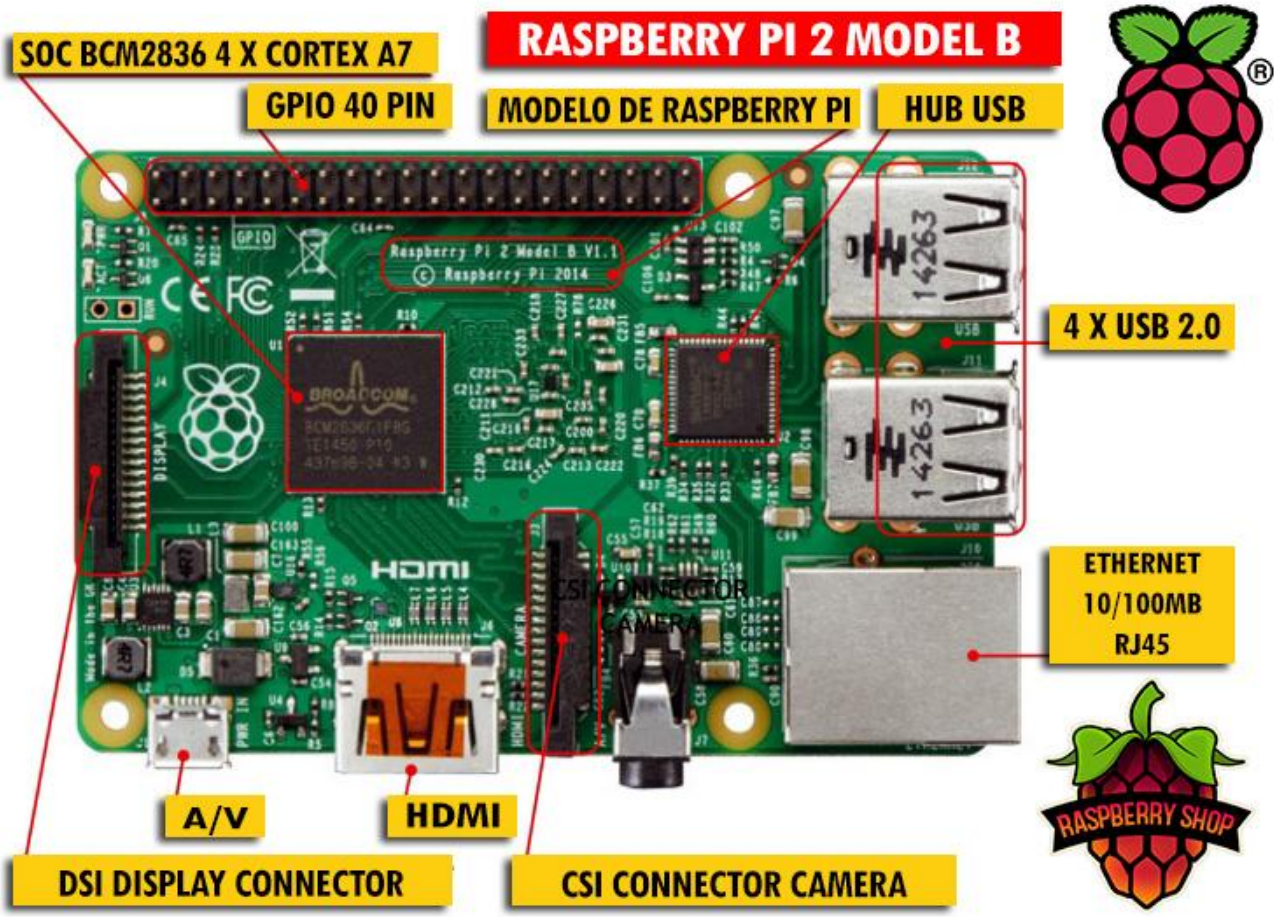
3 Description de l'ordinateur embarqué

3.1 Présentation

Raspberry Pi est un petit ordinateur sous le système d'exploitation Linux sur carte SD destiné à des applications d'informatique embarquée. Le cœur de l'ordinateur est un FPGA (Broadcom 2836) intégrant un processeur quad-core ARM Cortex A7 cadencé à 900MHz, 1Go de RAM et de nombreux périphériques.

Raspberry Pi peut être directement connecté à une IHM classique, souris/clavier/ecran HDMI ou vidéo composite, cependant comme tout ordinateur Linux, Raspberry Pi peut intégrer ses propres outils de développement et une IHM reposant sur SSH contrôlable depuis un autre ordinateur par Ethernet ou WIFI.

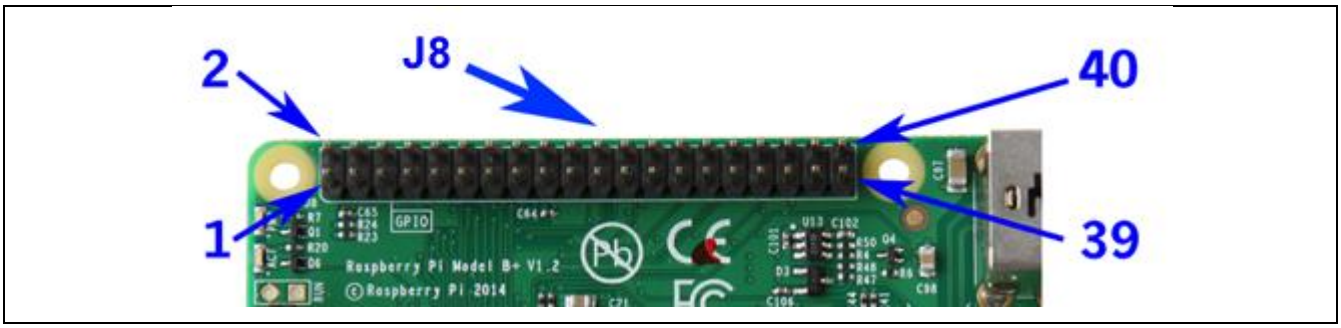
Le connecteur d'extension supporte les entrées/sorties parallèles ainsi que la plupart des bus de communication. C'est un support particulièrement économique et puissant qui peut être facilement mis en œuvre dans de petits systèmes nécessitant un accès au monde physique par des capteurs/actionneurs disposants d'interfaces numériques.



3.2 Les connecteurs d'extension

Le connecteur d'extension de la carte Raspberry PI est utilisé pour raccorder des périphériques de communication (UART, I2C, SPI, ...) ou TOR (Tout Ou Rien).

Les broches peuvent avoir des fonctions différentes suivant qu'elles sont activées en tant que GPIO (Global Purpose Input Output), périphérique de communication ou sorties PWM (Pulse Width Modulation).



Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I2C)		DC Power 5v	04
05	GPIO03 (SCL1 , I2C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)		(I2C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

http://www.element14.com/community/docs/DOC-73950/l/raspberry-pi-2-model-b-gpio-40-pin-block-pinout

3.3 Raspbian

Raspbian est un système d'exploitation libre basé sur la distribution GNU/Linux Debian, et optimisé pour le plus petit ordinateur du monde, la Raspberry Pi.

Raspbian ne fournit pas simplement un système d'exploitation basique, il est aussi livré avec plus de 35 000 paquets, c'est-à-dire des logiciels pré-compilés livrés dans un format optimisé, pour une installation facile sur votre Raspberry Pi via les gestionnaires de paquets.

La Raspberry Pi est une framboise merveilleuse, mais elle reste néanmoins dotée d'une puissance inférieure à celle d'un ordinateur moderne. Par conséquent, il est préférable d'installer un système optimisé pour la Raspberry.

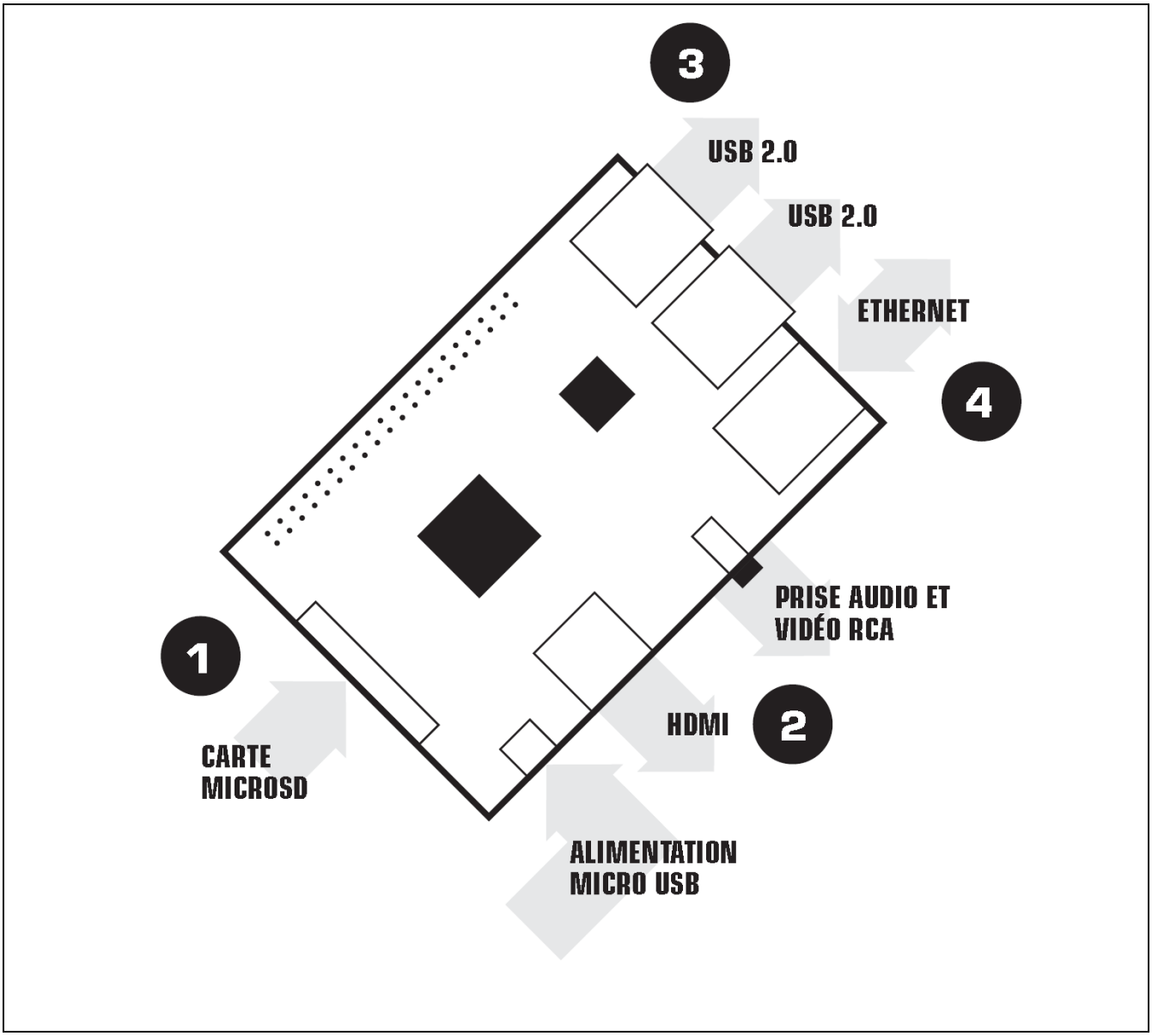
Raspbian a été créé dans cette optique, et il est donc tout particulièrement adapté à la Raspberry.

Par ailleurs, en tant que distribution dérivée de Debian, il répond à la majeure partie de la très vaste documentation de Debian.

https://www.raspbian.org/

4 Prise en main directe

4.1 Connexion des périphériques

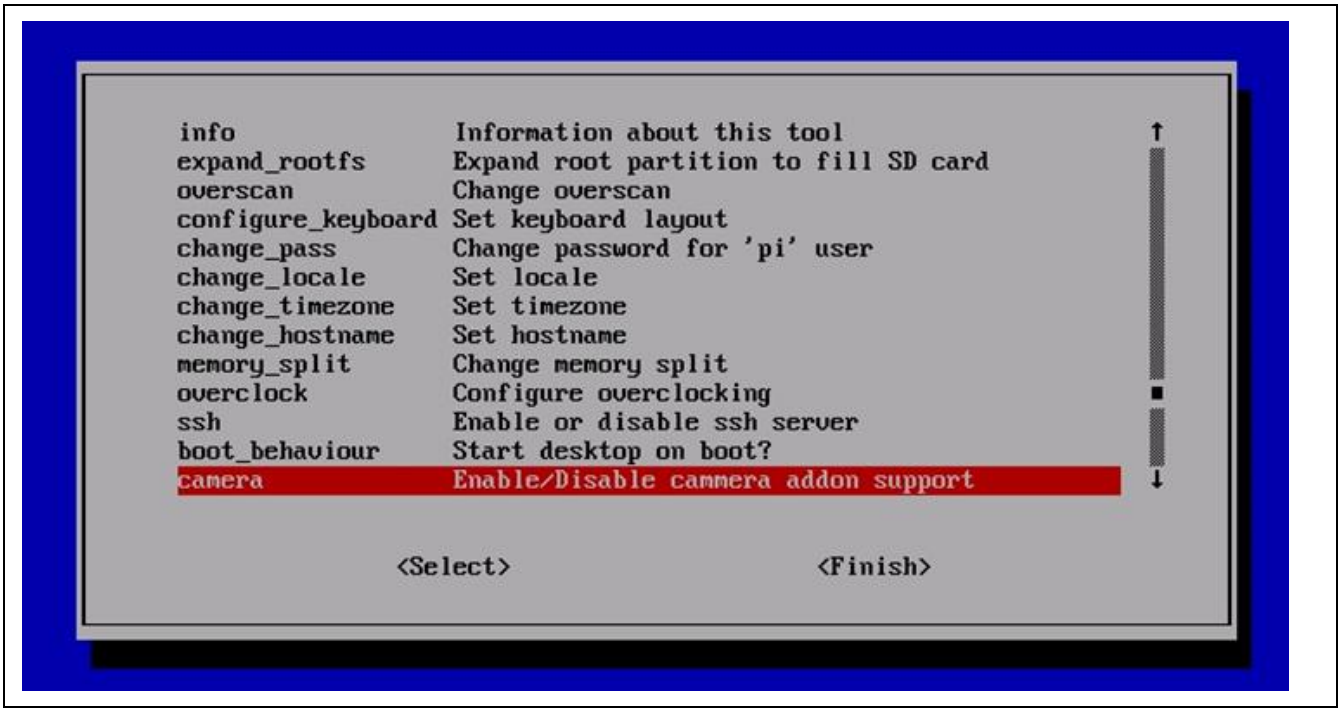


Avant de brancher quoi que ce soit à votre Raspberry Pi, assurez-vous de disposer de l'ensemble des éléments listés. Puis, suivez ces instructions :

- Commencez par insérer votre carte microSD dans le logement prévu sur le Raspberry Pi.
- Ensuite, connectez votre souris et votre clavier USB aux ports USB du Raspberry Pi.
- Assurez-vous que votre moniteur est sous tension et que vous avez sélectionné l'entrée appropriée (p. ex. HDMI 1, DVI, etc.).
- Ensuite, reliez votre Raspberry Pi à votre moniteur à l'aide du câble HDMI.
- Connectez votre Raspberry Pi au réseau à l'aide d'un câble Ethernet.
- Lorsque vous avez branché tous les câbles et inséré la carte microSD requise, branchez l'alimentation électrique micro USB. Cette action met sous tension et démarre votre Raspberry Pi.

4.2 Démarrage

Au premier démarrage, l'écran suivant apparait :



change pass : le mot de passe par défaut de l'utilisateur « pi » est « raspberry » (ne le changez pas)

- Indiquez votre emplacement géographique et heure, (locale et timezone).
- Configurez le clavier et activez le serveur SSH.

4.3 Connexion au réseau

Pour se connecter à un réseau, il faut renseigner le fichier **interfaces** : **/etc/network/interfaces**

- Configurez la connexion filaire pour recevoir automatiquement les paramètres TCP/IP d'un DHCP.
- Identifiez votre adresse IP.
- Vérifiez dans le navigateur l'accès à internet.
- Modifiez la configuration pour utiliser une adresse IP fixe (la même que celle que le DHCP vous a fourni).
- Vérifiez dans le navigateur l'accès à internet.

Pour la suite, reconfigurez la connexion en automatique.

4.4 Mise à jour

- Mise à jour de la liste des paquets
  apt-get update
- Mise à jour des paquets installés
  apt-get upgrade
- Mise à jour de la distribution Raspbian
  apt-get dist-upgrade

4.5 Compilation sur cible

C'est le mode de compilation le plus simple : on procède comme sur un PC fonctionnant sous une distribution classique le linux.

- Saisissez le programme suivant en enregistrez dans le dossier personnel de l'utilisateur **pi** sous le nom **hello1.cpp** :

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
        string prenom;
        cout << "Test d'ecriture et de compilation d'un programme C++ sur Raspbery PI " << endl ;
        cout << "- Ecriture sur cible  " << endl ;
        cout << "- Enregistrement local  " << endl ;
        cout << "- Compilation sur cible  " << endl ;
        cout << "Quel est ton prenom ? " ;
        cin >> prenom;
        cout << "Bonjour " << prenom << endl;
}
```

- Compilez et exécutez le programme. Sauvegardez le rapport de compilation.

  g++ -ftime-report hello1.cpp –o hello1

5 Prise en main à distance

5.1 ssh / scp

En général, les systèmes embarqués n'ont pas de moniteur, de clavier et de souris, ils sont conçus pour fonctionner à l'intérieur d'un système matériel et donc inaccessible. Leur gestion se fait alors à distance, par l'intermédiaire de liaisons de communications séries (souvent RS232) ou par réseau filaire ou wifi. On accède alors au système en mode console grâce au protocole SSH (Secure Shell) et on gère les fichiers grâce à l'utilisation de protocoles d'échange comme FTP, SFTP ou encore SCP. On peut également gérer l'interface graphique en mode distant (« terminal distant ») mais l'utilisation d'une interface graphique dans la plupart des systèmes embarqués est une hérésie. Elle consomme des ressources inutilement.

5.2 Hôte linux

Sur un hôte linux standard, les protocoles SSH et SCP sont généralement déjà installé.

- Depuis un hôte linux connecté au réseau, ouvrez une session SSH avec la Raspberry PI dans un terminal. Explicitez les différentes étapes.

  ssh <IP_Raspberry_PI>

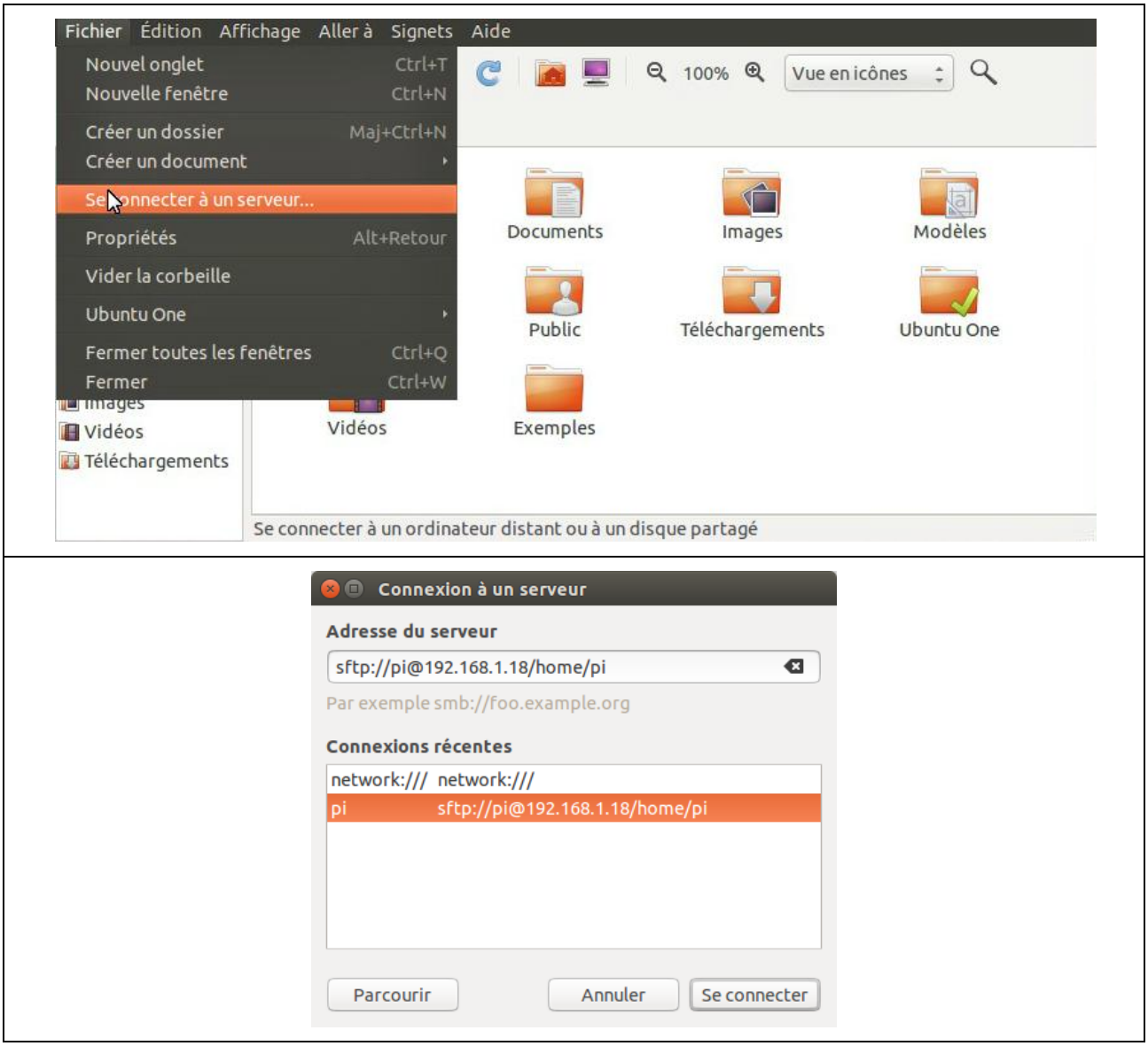
- La commande **scp** permet de copier un fichier ou un répertoire (**-r**) d'un client vers un serveur ou d'un serveur vers un client. Depuis votre hôte linux, copiez le fichier **hello1.cpp** présent sur la Raspberry PI dans votre dossier personnel.

  scp <Fichier_local> <login@IP_client_distant:Chemin>

  scp <login @IP_serveur_distant:Chemin/Fichier>

  Explicitez les différentes étapes et donnez la syntaxe précise utilisée.

- Le gestionnaire de fichiers de l'hôte local est Nautilus. Ce gestionnaire de fichiers permet de se connecter à un serveur distant et de monter le dossier indiqué dans le système de fichiers local, se qui permet d'y naviguer graphiquement. Depuis l'hôte linux, ouvrez Nautilus :



Remarque : Utilisez l'adresse IP de votre carte Raspberry PI.

- Ouvrez l'éditeur de texte de votre hôte linux et saisissez et enregistrez localement le programme suivant sous le nom **hello2.cpp** :

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
        string prenom;
        cout << "Test d'ecriture et de compilation d'un programme C++ sur Raspbery PI " << endl ;
        cout << "- Ecriture sur hote distant linux  " << endl ;
        cout << "- Enregistrement sur hote distant puis transfert sur cible  " << endl ;
        cout << "- Compilation sur cible depuis hote distant  " << endl ;
        cout << "Quel est ton prenom ? " ;
        cin >> prenom;
        cout << "Bonjour " << prenom << endl;
}
```

- Transférez le fichier sur la carte Raspberry PI dans le dossier personnel de l'utilisateur pi.

- Depuis l'hôte linux, compilez et exécutez le programme sur la cible. Sauvegardez le rapport de compilation.

  g++ -ftime-report hello2.cpp –o hello2

- Explicitez les différentes étapes.

5.3 Hôte Windows

Sous Windows, on utilisera le logiciel (PC) WinSCP et/ou le client SSH (Secure Shell) Putty.

- Téléchargez Putty ici : http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe
- Téléchargez WinSCP ici et installez-le : https://winscp.net/eng/docs/lang:fr
- Ouvrez l'éditeur de texte de votre hôte Windows (notepad++ ?) et saisissez et enregistrez localement le programme suivant sous le nom **hello3.cpp** :

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
        string prenom;
        cout << "Test d'ecriture et de compilation d'un programme C++ sur Raspbery PI " << endl ;
        cout << "- Ecriture sur hote distant Windows  " << endl ;
        cout << "- Enregistrement sur hote distant puis transfert sur cible  " << endl ;
        cout << "- Compilation sur cible depuis hote distant  " << endl ;
        cout << "Quel est ton prenom ? " ;
        cin >> prenom;
        cout << "Bonjour " << prenom << endl;
}
```

- Transférez le fichier sur la carte Raspberry PI dans le dossier personnel de l'utilisateur pi.
- Depuis l'hôte Windows, compilez et exécutez le programme sur la cible. Sauvegardez le rapport de compilation.

  g++ -ftime-report hello3.cpp –o hello3

- Explicitez les différentes étapes.

5.4 Cross-compilation

Le processus de compilation mobilise les ressources du système (temps processeur et mémoire). Il nécessite également un espace de stockage suffisant pour l'enregistrement des données temporaires de compilation.

Sur un système embarqué, ces caractéristiques matérielles sont naturellement limitées et il est souvent préférable de procéder à la compilation des sources sur un système plus performant puis de transférer les exécutables obtenus sur la cible.

Ce type de compilation s'appelle cross-compilation ou compilation croisé en français. Il s'agit de compiler sur une machine (PC) pour une autre (Raspberry PI). Le cross-compilateur doit être construit sur mesure, pour une cible donnée et sur un système hôte donné. Il faut donc le compiler en utilisant le compilateur de l'hôte, des outils pour la cible (**binutils**) et des bibliothèques C adaptées (**libC**). Cette préparation peut être très complexe. Il faut faire les bons choix et résoudre les problèmes de dépendances. Heureusement, il existe des solutions « pré-faites » comme **crosstool-NG**. Il permet de fabriquer sur mesure les outils nécessaires pour faire de la compilation croisée, en l'occurrence, les outils nécessaires pour compiler sur un PC x86-64 pour une plate-forme ARM. Nous allons donc voir comment utiliser **crosstool-NG** pour faire de la compilation croisée facile pour Raspberry Pi.

5.4.1 Installer crosstool-ng

- Nous aurons besoin des paquets **gawk, bison, flex, gperf, cvs, texinfo, automake, libtool ncurses-dev, g++, subversion, python-dev, libexpat1-dev et cmake**pour ce qui suit. Procédez à leur installation.
- Installez la dernière version de crosstool-NG : http://crosstool-ng.org/

```
$ export VERSION=1.21.0
$ wget http://crosstool-ng.org/download/crosstool-ng/crosstool-ng-${VERSION}.tar.bz2
$ tar xjf crosstool-ng-${VERSION}.tar.bz2
$ cd crosstool-ng-${VERSION}
$ ./configure --prefix=/opt/crosstool-ng
$ make
$ sudo make install
```

- Réglez les éventuels problèmes de dépendances révélés par **./configure**

5.4.2 Configurer crosstool-NG

Nous avons installé sur notre machine les scripts nécessaires pour fabriquer les outils de compilation croisée. L'étape suivante va consister à expliquer à crosstool-NG de quelle toolchain – c'est-à-dire de quels outils – nous aurons besoin. Nous préciserons entre autre l'architecture et l'OS cible et le type de compilateur qui nous intéresse.

- Ajoutez le chemin des exécutable de crosstool-NG au PATH

```
$ cd
$ export PATH= "${PATH} :/opt/crosstool-ng/bin"
```

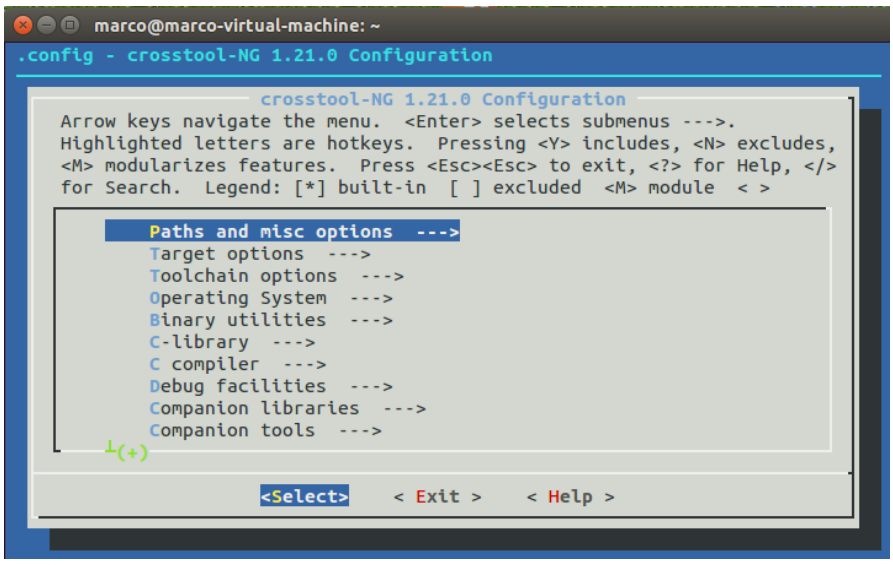
- Créez un dossier pour la configuration des outils à construire.
  Attention : ce n'est pas là que les outils seront installés à la fin !

```
$ mkdir my-dev
$ cd my-dev
```

- Configurez la toolchain à fabriquer.

```
$ ct-ng menuconfig
```

La fenêtre suivante s'ouvre :



- **Configurer Path Options**
  - **Paths and misc options.**
    - Validez **Try features marked as EXPERIMENTAL**.
    - Validez **Use a mirror**.
    - Renseignez **Number of parallel jobs** (ex : 4-core CPU avec HyperThreading, saisissez 16).
- **Configurer Target Options**
  - **Target options.**
    - Choisissez **arm** pour **Target Architecture**.
    - Sélectionnez **Endianness** à **Little endian** and **Bitness** à **32-bit**.
    - Sélectionnez **hardware (FPU)** pour **Floating point**.
    - Sélectionnez **append 'hf' to the tuple (EXPERIMENTAL).**
- **Configurer Operating System Options**
  - **Operating System.**
    - Choisissez **Linux** pour **Target OS**.
- **Configurer Binary Utility Options**
  - **Navigate to Binary utilities.**
    - Sélectionnez la dernière version de **Binutils**.
- **Configurer Lib C Options**
  - **C-library.**
    - Choisissez **glibC**.
- **Configurer Compiler Options**
  - **C-compiler.**
    - Sélectionnez la dernière version de **gcc**
    - Sélectionnez **Show Linaro versions**.
    - Sélectionnez **C++**.
- **Configurer Debug Options (Ne pas faire pour réduire le temps de compilation)**
  - **Debug facilities.**
    - Sélectionnez **gdb** puis **Build a static cross gdb**.
    - Saisissez **--with-expat** dans **Cross-gdb extra config**
    - Sélectionnez ltrace.
    - Sélectionnez strace.

5.4.3 Construire crosstool-NG

- Lancez le processus de fabrication de la toolchain.

```
$ ct-ng build
```

Remarque : Ce processus peut être très long en fonction des ressources de votre machine. A titre d'exemple, il m'a fallu attendre plus de 20 minutes sur une machine avec processeur i7 4 cores, 8 Go de RAM et 8Go de mémoire virtuelle (Swap). Il aura fallu plus de 3h30 sur une machine virtuelle avec processeur double cores, 2 Go de RAM et 2Go de mémoire virtuelle. Je recommande fortement de passer en mode console (Ctrl + Alt + F1 à F6) pour cette opération.

```
File  Edit  View  Search  Terminal  Help
marco@marco-virtual-machine:~/my-dev$ ct-ng menuconfig
  IN     config.gen/arch.in
  IN     config.gen/kernel.in
  IN     config.gen/cc.in
  IN     config.gen/libc.in
  CONF   config/config.in
#
# configuration saved
#
marco@marco-virtual-machine:~/my-dev$ ct-ng build
[INFO ]  Performing some trivial sanity checks
[INFO ]  Build started 20121223.153535
[INFO ]  Building environment variables
[INFO ]  ======================================================================
[INFO ]  Retrieving needed toolchain components' tarballs
[INFO ]  Retrieving needed toolchain components' tarballs: done in 0.08s (at 00:02)
[INFO ]  ======================================================================
[INFO ]  Extracting and patching toolchain components
[INFO ]  Extracting and patching toolchain components: done in 2.12s (at 00:04)
[INFO ]  ======================================================================
[INFO ]  Installing GMP for host
[INFO ]  Installing GMP for host: done in 27.93s (at 00:32)
[INFO ]  ======================================================================
[INFO ]  Installing MPFR for host
[INFO ]  Installing MPFR for host: done in 10.99s (at 00:43)
```

A la fin du processus de fabrication, si tout c'est bien passer, un nouveau dossier x-tools a été créé. Il contient le compilateur et les éléments nécessaires à la compilation croisée de programmes pour linux sur une architecture ARM.

- Pour pouvoir compiler de n'importe où dans le sytème de fichier, renseignez la variable PATH avec le chemin d'accès au compilateur : **/x-tools/arm-unknown-linux-gnueabihf/bin**

5.4.4 Compilation et test

- Ouvrez l'éditeur de texte de votre hôte linux et saisissez et enregistrez localement le programme suivant sous le nom **hello4.cpp** :

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
        string prenom;
        cout << "Test d'ecriture et de crosscompilation d'un programme C++ pour Raspbery PI" << endl ;
        cout << "- Ecriture sur hote distant Ubuntu  " << endl ;
        cout << "- Transfert par ssh / serveur monte dans FS Ubuntu  " << endl ;
        cout << "- Compilation sur hote  " << endl ;
        cout << "Quel est ton prenom ? " ;
        cin >> prenom;
        cout << "Bonjour " << prenom << endl;
}
```

- Depuis l'hôte linux, compilez, transférez l'exécutable généré et exécutez le sur la cible. Sauvegardez le rapport de compilation.

  arm-unknown-linux-gnueabihf-g++ -ftime-report hello4.cpp -o hello4 -static-libstdc++

- Explicitez les différentes étapes.

6 Utilisation des GPIO

Le Raspberry Pi offre quelques possibilités d'entrées-sorties directes en utilisant les broches GPIO présentes sur son connecteur P1. Elles ne sont pas très nombreuses (une dizaine) mais cela peut suffire pour des petits projets interactifs nécessitant d'interroger des capteurs tout-ou-rien ou de valider des actionneurs.

Nous pouvons utiliser ces GPIO de différentes façons, depuis l'espace utilisateur ou depuis le noyau. Le but ici est de balayer les principaux aspects.

6.1 L'interface sysfs

Sysfs est un système de fichiers virtuel introduit par le noyau Linux 2.6. Sysfs permet d'exporter depuis l'espace noyau vers l'espace utilisateur des informations sur les périphériques du système et leurs pilotes, et est également utilisé pour configurer certaines fonctionnalités du noyau.

Pour chaque objet ajouté dans l'arbre des modèles de pilote (pilotes, périphériques, classes de périphériques), un répertoire est créé dans sysfs.

- La relation parent/enfant est représentée sous la forme de sous-répertoires dans **/sys/devices/** (représentant la couche physique).
- Le sous-répertoire **/sys/bus/** est peuplé de liens symboliques, représentant la manière dont chaque périphérique appartient aux différents bus.
- **/sys/class/** montre les périphériques regroupés en classes, comme les périphériques réseau par exemple.

Pour les périphériques et leurs pilotes, des attributs peuvent être créés. Ce sont de simples fichiers, la seule contrainte est qu'ils ne peuvent contenir chacun qu'une seule valeur et/ou n'autoriser le renseignement que d'une valeur. Ces fichiers sont placés dans le sous-répertoire du pilote correspondant au périphérique.

6.2 Lire/Ecrire sur une broche TOR

6.2.1 Accès depuis l'espace utilisateur

L'accès simple, depuis le shell peut se faire très aisément grâce au système de fichiers **/sys**.

```
/ # cd /sys/class/gpio/
/sys/class/gpio # ls
export     gpiochip0  unexport
```

Accédons au GPIO 24 (broche 18) :

```
/sys/class/gpio # echo 24 > export
/sys/class/gpio # ls
export     gpio24     gpiochip0  unexport
/sys/class/gpio # cd gpio24/
/sys/class/gpio/gpio24 # ls
active_low   direction   edge   subsystem   uevent   value
```

6.2.2 Ecriture

La broche doit être configurée en sortie pour écrire :

```
/sys/class/gpio/gpio24 # echo out > direction
/sys/class/gpio/gpio24 # echo 1 > value
/sys/class/gpio/gpio24 # echo 0 > value
```

6.2.3 Lecture

La broche doit être configurée en entrée pour lire :

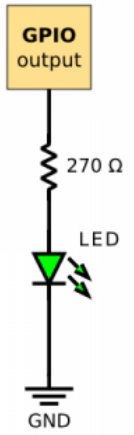
```
/sys/class/gpio/gpio24 # echo in > direction
/sys/class/gpio/gpio24 # cat value
0
```

6.2.4 Programmation des GPIO

L'utilisation des ports GPIO est liée à l'accès à plusieurs fichiers. Les opérations d'export, de configuration de la direction, de lecture ou d'écriture peuvent être réalisées dans n'importe quel langage de programmation.

6.3 Application

- Réaliser une application dans le langage de votre choix permettant de faire clignoter une LED connectée sur la broche 18.
- Explicitez dans votre compte-rendu la méthodologie appliquée (langage, librairies éventuellement utilisées, outils de développement, compilation éventuelle, sur cible ou sur hôte, exécution, ...).



7 Installation de logiciel

7.1 Installation d'un serveur web

L'implantation d'un serveur web dans un système embarqué permet un accès simplifié à ce dernier à travers une interface Web. On en trouve par exemple dans des équipements de domotique, matériels électro-ménagers ou Hi-Fi, équipement réseaux, caméra IP et de nombreuses applications professionnelles.

Les serveurs web embarqués doivent prendre en compte de nombreuses problématiques liées à leurs caractéristiques. En effet, la taille réduite, la consommation énergétique, l'architecture flexible ainsi que le coût de ces serveurs sont autant de problématiques qu'il est nécessaire de soulever pour amener internet vers les systèmes embarqués.

De nombreuses solutions sont disponibles, la plus utilisée lorsque les performances du système le permet est Apache (environ 60% de part de marché).

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

7.1.1 Installation

```
# apt-get install apache2 php5
```

7.1.2 Test

- Dans un navigateur web, saisissez l'adresse IP de la carte Raspberry PI.
- Explicitez les différentes étapes de l'installation est décrivez votre procédure de test pour vérifier le fonctionnement du serveur pour l'affichage de pages html et php.

7.2 Installation d'un SGBD

Bien que les SGBD offrent de nombreux avantages par rapport à un enregistrement sur fichier, ces derniers sont souvent préférés aux SGBD dans les systèmes embarqués. En effet, les SGBD ont la réputation d'être des logiciels lourds, encombrants, compliqués à installer et à manipuler.

On devrait les utiliser dès lors que l'on souhaite stocker et manipuler des données issues de capteurs ou de l'activité du système embarqué, en particulier lorsqu'il ne peut être connecté en permanence à un réseau de données, excluant ainsi la possibilité de transmettre ses données au fil de l'eau.

Divers solutions existent sur le marché, la plus connue étant certainement MySQL. Dans le monde de l'embarqué, on lui préfère souvent des solutions plus compacts comme SQLite, mais les performances du Raspberry PI nous permettent de faire le choix de MySQL.

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

7.2.1 Installation

```
# apt-get install mysql-server php5-mysql
```

7.2.2 Test

- Dans un terminal, saisissez la commande suivante :

```
mysql --user=root --password=votrepasswd
```

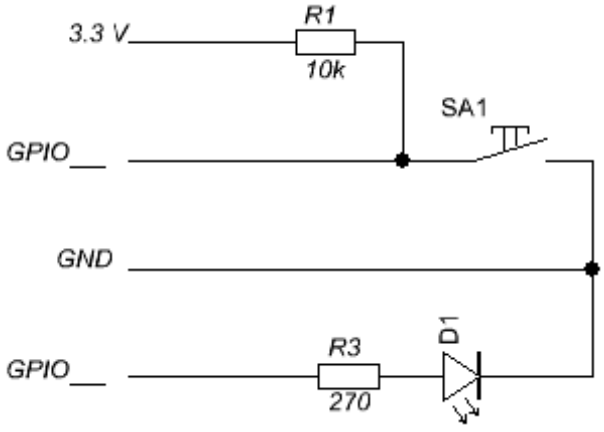
- Explicitez les différentes étapes de l'installation et proposez une solution permettant de gérer vos bases de données par une interface graphique et à distance.

8 Mini-projet

Le but du mini projet est de réaliser une interface web de supervision distante d'un système matériel simplifié utilisant les entrées/sorties TOR disponibles sur le Raspberry PI.

L'application doit permettre de :

- Contrôler la led D1
- Visualiser l'état du switch SA1
- Tracer les activités (base de données ou fichiers log)
- Afficher les données d'activités selon plusieurs critères (date, switch, led, ...)



Les technologies de programmation sont librement choisies et explicitées dans votre compte-rendu.

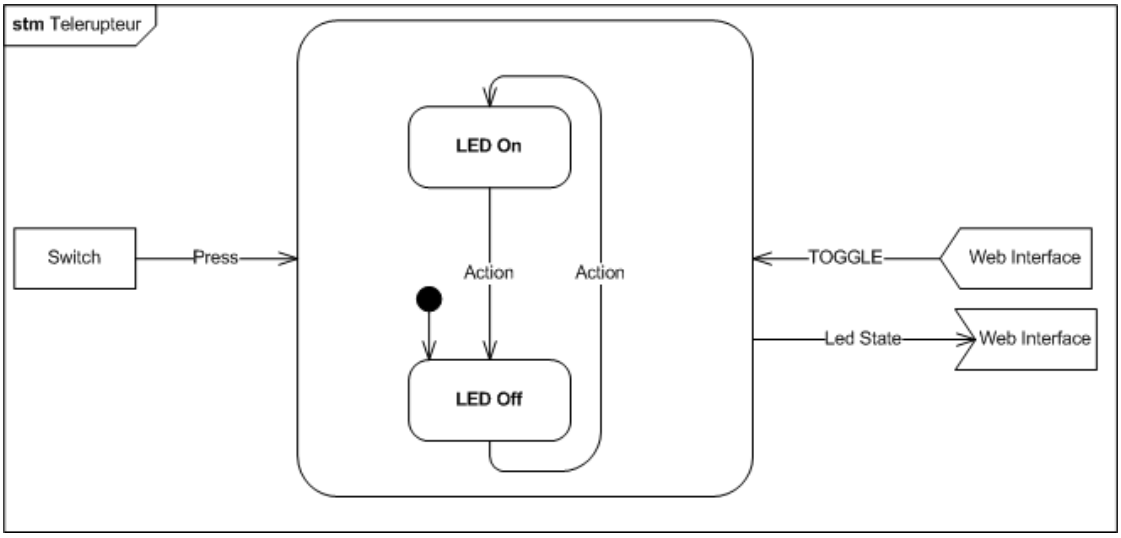
Le fonctionnement de l'application fait l'objet d'une **discussion avec le professeur**.

9 Proposition de réalisation

9.1 Principe général

L'interface web doit pouvoir allumer ou éteindre la LED et afficher en temps réel l'état du bouton poussoir.

On pourra, pour illustrer la problématique, réaliser un télérupteur commadé par le switch ou par l'interface web.

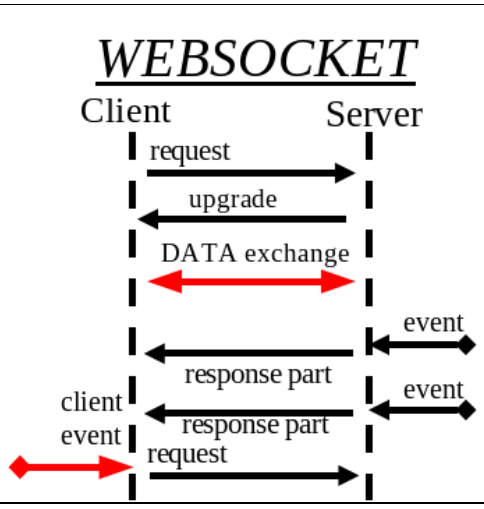


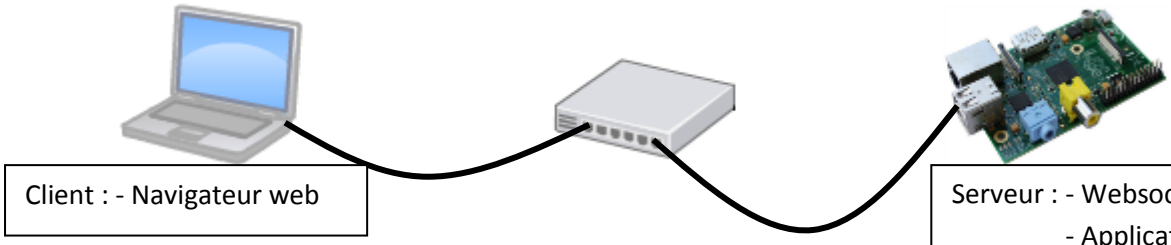
- La led change d'état à chaque front montant du switch (à la pression mais pas au relachement).
- L'interface web permet de voir en temps réel l'état de la led et de changer son état. On conserve l'historique des actions menées.
- La led est connectée au GPIO21 et le switch au GPIO20
- Interruption sur front montant sur le port GPIO20 (switch) :

```
#/sys/class/gpio/gpio20 # echo rising > edge
```

9.2 Technologies

La difficulté réside dans l'affichage temps réelle de l'interface web sans action à partir du client. La technologie des Websockets est ici parfaitement adaptée.

WebSocket est une spécification d'un protocole permettant une communication bidirectionnelle et full duplex sur une seule socket TCP entre un client et un serveur. Initialement développé pour HTML 5, WebSocket a été normalisé par l'IETF et le W3C. Tous les navigateurs récents implémentent et supportent les WebSockets. Ce protocole permet notamment d'implémenter facilement et de manière standard l'envoi de données en mode Push à l'initiative du serveur. http://tools.ietf.org/html/rfc6455	



- **Websocket serveur** : http://websocketd.com/
- **Application telerupteur** : A développer dans le langage de votre choix.
- **Page web** : HTML et ou php / CSS / javascript

9.3 Application telerupteur

Elle doit changer d'état la led à chaque front montant délivré par le switch et à la demande du client. Le serveur websocket se substitue aux flux standard d'entrée et de sortie. Dans une exécution hors websocket, l'entrée standard est le clavier et la sortie est la console.

- Il y a deux types d'évènements à surveiller => réalisation de deux threads.
- Dans chaque thread, il faudra agir sur la led.

```
int toggle(string source)
{
	// inverser la valeur dans gpio21/value et afficher un log
}

void bySwitch()
{
	// Ouvrir le fichier qui contient l'etat du switch
	while (1)
	{
		// Attente passive sur le switch (pas de timeout, donc infinie...)
		// Lire la valeur actuelle du switch
		// Verrouiller l'acces au fichier qui contient l'etat de la LED
		// Lire l'état de la LED
		// Inverser l'état de la LED
		// Deverrouiller l'acces au fichier qui contient l'etat de la LED
	}
	close(fd);
}// fin du thread bySwitch

void byApp()
{
	while(1)
	{
		// Lire la commande sur le flux d'entrée
		// Verrouiller l'acces au fichier qui contient l'etat de la LED
		// Inverser l'état de la LED
		// Deverrouiller l'acces au fichier qui contient l'etat de la LED
	}
}// fin du thread byApp

int main()
{
	thread th1 (bySwitch);
	thread th2 (byApp);

	cout << "Gestion des GPIO 20 et 21" << endl;

	th1.join();
	th2.join();
}
```

Compilation : `g++ telerupteur.cpp -o telerupteur -std=c++11 -lpthread`

9.4 Page web

Elle doit contenir la connexion au websocket :

```
<!DOCTYPE html>
<script>
  var msg;
  function log(msg)
  {
       // Afficher l'image de la LED correspondant à son état
       // Ajouter une ligne de log
  } ;
  // Ouverture du websocket
  var ws = new WebSocket('ws://192.168.1.31:8080/');
  ws.onopen = function()
  {
    log('CONNECT');
  };
  ws.onclose = function()
  {
    log('DISCONNECT');
  };
  ws.onmessage = function(event)
  {
    msg = event.data;
    log(msg);
  };
  function sendLED()
  {
       if(ws)
       {
              try { ws.send('TOGGLE'); }
              catch (ex){alert("Erreur : "+ex);}
       }
  } ;
</script>
<HTML>
<HEAD>
       <style>
           /* Mes styles */
       </style>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
       <TITLE>Supervision GPIO</TITLE>
</HEAD>

<BODY>
<center><h1>Controler la LED sur GPIO21 </h1>
       <a href="#" onclick="sendLED()";><img id="LED" src ="leds/led-off.png"></a>
</center>
<h2>   Historique :</h2>
<div id='log'></div>
</BODY>

</HTML>
```

Ce script est à placer dans le dossier indiqué par l'option –staticdir lors de l'exécution du serveur. Le client devra spécifier explicitement le port, ici : http://192.168.1.31:8080

Le script peut aussi être placé dans le dossier /var/www/html si apache ou lighttpd est installé.

9.5 Lancement du serveur websocket

```
$ ./websocketd --port=8080 --staticdir=. ./telerupteur
```


Fichier Édition Affichage Historique Marque-pages Outils ?
Supervision GPIO
192.168.1.31/supervise.html
raspberry pi
http://www.hosteur.c...
Les plus visités
Débuter avec Firefox
À la une
Contrôler la LED sur GPIO21
Historique :
CONNECT
Gestion des GPIO 20 et 21Surveillance App ...OK
Surveillance switch ...OK
21:12:39 : Led allumee par switch
21:12:43 : Led eteinte par App
21:12:46 : Led allumee par App
21:12:48 : Led eteinte par App
21:12:49 : Led allumee par App
21:12:51 : Led eteinte par App