Logiciel Pyramidion

Pierre FICHEUX (pierre.ficheux@smile.fr)

29/01/2019

MODIFICATIONS

VERSION	***DATE***	***AUTEUR***	***DESCRIPTION***
1.0	29/01/2019	P. Ficheux	Création

Table des matières

1.Introduction.....4
2.Description technique.....4
2.1.Composants pour le maître.....4
2.2.Composants pour l’esclave.....5
3.Configuration de la carte.....5
3.1.Installation du maître.....5
3.2.Installation de l’esclave.....6
4.Reste à faire.....7
5.Références et bibliographie.....7

1.Introduction

Ce document décrit le fonctionnement du système de contrôle du Pyramidion (projet PHARES) [1]. Le logiciel fonctionne actuellement sur une Raspberry Pi (ou 0, B+, 3) mais tout autre système sous Linux capable de contrôler des GPIO peut convenir.

Le but du système est de contrôler le clignotement des leds dans deux cas :

- avec une fréquence de 30 bpm (comportement par défaut « au repos »)
- en reproduisant la fréquence cardiaque d'un utilisateur à l'aide d'un capteur adapté (Easy Pulse [2])

La système utilise une GPIO en sortie (le pilotage des leds) et deux en entrée (le capteur Easy Pulse et le commutateur manuel/automatique).

Le commutateur manuel/automatique permettant de choisir le mode de fonctionnement :

- en mode manuel le système fonctionne en permanence (clignotement des leds)
- en mode automatique le système démarre/arrête le service à heures fixes (exemple : début à 19:00, fin à 02:00) → utilisation du service « cron »

Un système « maître » (disposant du capteur Easy Pulse) peut également communiquer la période à un système distant « esclave » en transmettant la fréquence de clignotement par un message MQTT. Les deux sites clignotent alors à la même fréquence.

2.Description technique

Le fonctionnement est basé sur plusieurs composants. Les programmes installés sont différents pour le maître et l'esclave.

2.1.Composants pour le maître

- un programme `gpioIrq` (écrit en langage C) et permettant le pilotage direct des GPIO (entrée et sortie)
- un script `pyramidion-gpio.sh` dont le seul but est d'exécuter le programme précédent avec les options correctes (numéros des GPIO en entrée et sortie, adresse du serveur MQTT, et « topic » MQTT associé)

  ```
  gpioIrq -i $GPIO_IN -o $GPIO_OUT -h $MQTT_SERVER -t $MQTT_TOPIC
  ```
- un script `pyramidion-button.sh` qui démarre le cron, initialise les GPIO et teste l'état manuel/automatique.
  - Si le bouton est basculé vers « automatique » on arrête le service GPIO
  - Si le bouton est basculé vers « manuel » on démarre le service GPIO
- un script `pyramidion-start.sh` appelé par « cron » → démarrage service GPIO

- un script `pyramidion-stop.sh` appelé par « cron » → arrêt service GPIO
- un fichier `cron.tab` définissant les heures de démarrage / arrêt
- les fichiers `.service` associés (pour systemd)

REMARQUES :

- Le basculement en mode manuel n'est pas possible durant les horaires du mode automatique si le clignotement est actif
- Si le mode est manuel, le démarrage par « cron » ne change rien mais si l'on relâche le bouton le service est arrêté même si l'on est dans la plage horaire du « cron » → ne PAS oublier de basculer le bouton en mode en automatique après usage !!
- Le capteur Easy Pulse n'est pas très « performant ». De ce fait le programme `gpioIrq` attends 10 secondes avant d'utiliser la valeur de bpm obtenue (stabilisation) ! Cette valeur est modifiable avec l'option `-w` du programme `gpioIrq` .

2.2.Composants pour l'esclave

- le programme `rpi_gpio_ns` (écrit en langage C) qui fait clignoter une GPIO avec une période donnée
- le script `pyramidion-receive.sh` qui reçoit la valeur de fréquence du maître (par MQTT) et démarre le programme `rpi_gpio_ns`
- `le script pyramidion-30bpm.sh` qui démarre le programme rpi_gpio_ns avec une fréquence de 30 bpm au démarrage de la carte. Pour une raison obscure le démarrage à 30 bpm du programme `rpio_gpio_ns` au début du script `pyramidion-receive.sh` ne fonctionne pas !
- les fichiers `.service` associé (pour systemd)

3.Configuration de la carte

Actuellement la configuration de la carte est manuelle à partir d'un Raspbian 9 (Stretch). La carte maître nécessite une horloge sauvegardée (RTC) car la Pi n'en dispose pas par défaut. Le paquets nécessaires pour le maître et l'esclave sont différents mais pour simplifier on installera les mêmes sur les deux cibles.

```
$ sudo apt-get install mosquitto-clients libmosquitto-dev i2c-tools
```

3.1.Installation du maître

Compiler et installer le programme `gpioIrq` :

```
$ cd gpioIrq
$ make
$ sudo make install
```

Copier les scripts suivants dans `/home/pi` :

- `pyramidion-button.service`
- `pyramidion-button.sh`
- `pyramidion-gpio.service`
- `pyramidion-gpio.sh`
- `pyramidion-gpio-start.sh`
- `pyramidion-gpio-stop.sh`

Installer et activer les services systemd :

```
$ sudo cp *.service /etc/systemd/system
$ sudo systemctl enable pyramidion-button.service
```

Configurer le fichier « cron » (fichier `/home/pi/cron.tab`) et valider la configuration.

```
$ crontab cron.tab
```

Le contenu du fichier est le suivant :

```
# Format = min heure jour-du-mois mois jour-semaine
#
# exemples:
#
# 0 8 * * * -> tous les jours a 8h
# 0 * * * * -> tous les heures
#
# Début à 19h, fin à 2h
00 19 * * * sudo ~pi/pyramidion-gpio-start.sh
00 02 * * * sudo ~pi/pyramidion-gpio-stop.sh
```

Il faut également configurer la RTC [3] en suivant le lien [4]. Par rapport au lieu précédent nous avons parfois du ajouter une attente (commande `sleep`) au fichier `/etc/rc.local`.

```
# RTC
echo "Setting RTC."
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
sleep 2
hwclock -s
```

3.2.Installation de l'esclave

La configuration est plus simple car l'esclave reçoit la fréquence BPM par MQTT.

Compiler et installer le programme `rpi_gpio_ns` :

```
$ cd rpi_gpio_ns
$ make
$ sudo make install
```

Copier les scripts suivants dans `/home/pi` :

- `pyramidion-receive.service`
- `pyramidion-receive.sh`
- `pyramidion-30bpm.service`
- `pyramidion-30bpm.sh`

Installer et activer les services systemd :

```
$ sudo cp *.service /etc/systemd/system
$ sudo systemctl enable pyramidion-30bpm.service
$ sudo systemctl enable pyramidion-receive.service
```

4.Reste à faire

Les spécifications du fonctionnement n'ont pas été vraiment écrites et le logiciel a évolué peu à peu. Le logiciel est fonctionnel mais a été peu testé (à peine quelques heure en simulation). De plus il n'existe *personne* dans l'environnement du projet ayant des compétences en Linux et en programmation.

Il est *fondamental* d'écrire un jeu de test afin de détecter les conditions aux limites. Pour cela il faudra tester le logiciel sur une longue durée (plusieurs jours) et provoquer des changements d'état aléatoires ainsi que des erreurs.

Dans une deuxième phase il est important de pouvoir configurer le système sans avoir de connaissances Linux :

- mise à jour des logiciels
- configuration de l'accès réseau Wi-Fi
- etc.

Cette configuration pourrait s'effectuer depuis un terminal quelconque en mettant à jour des données de configuration (que la Raspberry Pi irait télécharger). Dans le cas d'un système non connecté, une mise à jour locale pourrait également être définie en utilisant une clé USB (ou bien la Raspberry Pi pourrait définir un point d'accès Wi-Fi utilisable par un terminal).

5.Références et bibliographie

[1] Projet PHARES http://www.mileneguermont.com/fr-fr/oeuvres/phares.html
[2] Capteur Easy Pulse https://www.tindie.com/products/rajbex/easy-pulse-sensor-based-on-photoplethysmography/
[3] Carte RTC https://www.amazon.fr/gp/product/B06XZYPNMG/ref=oh_aui_detailpage_o00_s00?ie=UTF8&psc=1

[4] Configuration RTC http://hardware-libre.fr/2013/08/raspberry-pi-ajouter-une-horloge-rtc-en-i%C2%B2c/