CFPT

Documentation technique

Partie Raspberry

Groupe Bugs Destroyer
Octobre 2021

Table des matières

I. Introduction

Ceci est un projet annexe des bornes d'arcades initialement prévu pour la cité des métiers 2021. Il a été réalisé par l'Ecole Entreprise 2021-2022. En reprenant le développement du menu de sélection de jeux installés sur les bornes d'arcades, nous avons prévu une solution de minuteur afin de ne plus pouvoir jouer indéfiniment sans laisser sa place à d'autres joueurs.

Afin de visualiser ce minuteur et voir combien de temps de jeu il reste, un Raspberry Pi contrôle un affichage 7 segments. Ce Raspberry communique avec le PC de la borne d'arcade via une connexion Ethernet (et grâce au protocole HTTP).

Ce document est destiné aux futurs développeurs qui devront reprendre le projet.

II. Informations générales

Le code source est disponible publiquement sur GitHub. Afin de mettre en place votre environnement de développement, vous pouvez suivre les étapes 1 à 3 et la partie "Câblage" du manuel d'installation. Pour indiquer au server Flask qu'il s'agit d'un environnement de test et non plus de production, définissez le paramètre "debug" sur True dans la fonction app.run().

```
app.run(host='0.0.0.0', port=80, debug=True)
```

N'hésitez pas à vous référer au site de Adafruit, qui donne de nombreux détails sur la library utilisée pour commander l'affichage.

III. Développement en python

Le script repose sur 2 sortes de boucles qui tournent indéfiniment :

1. Le listener Flask.

   C'est le serveur web. Il est chargé d'attendre et de répondre aux requêtes GET entrantes. Il est paramétré de sorte à répondre sur l'URL `http://<IP raspberry>:80/timerStatus`

   Il répond un texte formaté en JSON sous forme de `{"info":"ok"}`. Il répond également un code selon les standards du protocole HTTP.

2. La boucle while infinie

   Elle ne tourne pour rien tant que le serveur web ne lui a pas dit de commencer. Lorsque c'est le moment, elle va s'exécuter une fois par seconde et gérer l'actualisation de l'affichage et le nombre de secondes restantes.

Le script repose sur le concept de multi-processing. Le but de cette méthode est de pouvoir faire tourner les 2 tâches simultanément.

Une fonction récursive est utilisée pour réduire de 1 seconde le temps. Ce temps est stocké sous forme de 4 chiffres dans un array.

IV. Configuration du script

Dans le manuel d'installation, on explique comment faire des modifications essentielles pour faire marcher l'application. Ici nous allons voir 2 autres paramètres moins importants.

1. Server Web :

   Par défaut, le server écoute sur le port 80 (donc en HTTP seulement). Il répond aux requêtes GET de n'importe quels clients, à moins que vous n'ayez installé un firewall sur votre Raspberry.
   Vous pouvez changer le port en modifiant les paramètres de la fonctions "app.run()".
   Vous pouvez également sélectionner le mode de débogage.

```
app.run(host='0.0.0.0', port=80, debug=True)
```

2. Luminosité de l'affichage :

   La luminosité des leds est variable de 0 à 1 (1 étant le maximum). Vous pouvez le modifier simplement dans le code avec cette instruction :

```
display.brightness = 0.5
```

V. Informations supplémentaires :

Pour récupérer le code : GitHub

Product Owner : Katia MOTA & Francisco GARCIA

Si vous avez besoin de plus d'informations, n'hésitez pas à contacter jeremie.arcdc@eduge.ch.