Controler sa maison à l'aide d'un Raspberry: Présentation générale

Essig Meryll Rocacher Tamara

17 mars 2016

Introduction

Au cours de cette dernière semaine nous avons appliqué le circuit de la photrésistance à l'Arduino, avec un programme qui envoie la valeur sur le port usb du Raspberry Pi. Par la suite nous avons intégré la réception de données par USB et le contrôle des différents éléments dans notre code. Pour finir nous avons fait un récapitultif de l'avancement, rajouté quelques fonctionalités pour commencer une utilisation possible à tester et définit les prochaines étapes attendues.

Travail réalisé

Noyau

Nous avons basé notre application sur un système client-serveur, permettant au client d'avoir par la suite une interface graphique pour une meilleure utilisation. D'autre part cela permet a l'utilisateur de contrôler à distance avec une accessibilité simplifiée. Cela nous permettra notament de faire nos demonstrations aux réunions. Le serveur contient tout le programme noyau, tandisque le client intervient pour donner des ordres prioritaires ou des configurations. Le serveur utilise Netty.io, une librairie réseau évenementielle asynchrone. Chaque type de composant est instancié et un pin GPIO y est associé dans un ordre prédéfini. Par la suite les numeros de pin seront configurables.

Photorésistance

Un code (en PJ et sur git) a été televersé sur l'Arduino permettant de récupérer la valeur de la photrésistance et de l'envoyer dans un flux série. Coté serveur (Raspberry Pi) on lit le flux de données série et on récupère la valeur, a un intervalle de 1 seconde. Cette information est ensuite utilisée pour comparer la luminosité à un seuil. Nous avons choisit un seuil qui permet des tests, mais par la suite il correspondra a la luminosité déclenchant le mode jour. Lorsque le seuil de luminosité est atteint, une LED s'éteint (ie les lumières en journée). D'autre part, une seconde LED est quand à elle contrôlable, permettant à l'utilisateur d'avoir la main en plus des modes actif. Pour permettre une demonstration, nous nous sommes servis de cette seconde LED pour pouvoir simulé le changement Jour/Nuit.

Buzzer

Un buzzer à été mit en place pour permettre de programmer un reveil ou un minuteur de cuisine, ou tout autre alarme (incendie, intrusion,..). Le buzzer utilise généralement un signal

sinusoïdale permettant d'obtenir un son d'après une fréquence. Le Raspberry Pi n'étant pas analogique, nous avons réalisé un algorithme qui permet de réaliser en boucle un cycle activé-désactivé du buzzer selon une fréquence et une durée. Pour le moment le buzzer est actionné par la commande BUZZ.

Client

Le client est une interface permettant à l'utilisateur de communiquer avec le serveu, et donc avec le programme. Sous forme d'un chat, l'utilisateur envoie une chaine de caractères permettant une commande. En fonction de la commande, le client recoit en reponse l'état (action réussie ou non). Lorsque le client se déconnecte, le programme du serveur continue ses traitements automatique (analyses pour les scénarios,etc).

Difficultés rencontrées

Nous avons été ralentis en grande partie par des problèmes de compilation liés aux bibliothèques et au paramétrage de notre IDE. Nous avons utilisé intelliJ idea, auquel nous avons ajouté les différentes librairies necessaires en .jar.
D'autre part le traitement des données USB à necessité beaucoup de mises au point car il y a des interruptions régulières qu'il faut pouvoir filtrer pour éviter une données corrompue ou une interruption du programme.

A venir

configuration

Nous utiliserons un fichier de type .ini pour faciliter les paramétrages récurrents en évitant de devoir recompiler le programme. Seront concernés par exemple, l'IP du serveur, le port, des données utilisateurs ou encore l'utilisation des pins GPIO.

buzzer

La commande de contrôle permettra d'utiliser une heure de déclenchement. Nous utiliserons la librairie Quartz, pemettant de manipuler le temps et la gestion de tâches. *client On pourra prévoir une fonctionnalité se basant sur un client principal (necessité de s'identifier). Lorsque ce client est deconnecté du serveur, le scénario "Congés" est activé, avec eventuellement un delai d'erreur. Ainsi la commande "Deconnexion" (par la suite un bouton), permettra de mettre toute la maison en hibernation.

scénarios

La structure se base sur un fichier contenant les règles associées à des faits(états) et des actions. Nous comptons utiliser de nouveau la librairie Quartz pour la planification de tâches, coté gestion des règles. Des tests nous permettrons mieux par la suite de determiner les librairies que nous devront utiliser (Jess,..) ou une inmplémentation personnel du minimum necessaire pour nos besoins.