La veilleuse enfant connectée

Rapport de projet IMA 4

Elève ingénieur :

Delatte Hugo

SOMMAIRE

INTRODUCTION

Les jeunes enfants ont souvent un sommeil agité ou n'arrivent tout simplement pas à le trouver. Soucieux du bien-être de nos enfants, nous créons depuis des années des petits accessoires de nuit leur permettant de passer de meilleures nuits. Le plus connus est surement la veilleuse diffusant une faible lumière combattant le noir, et permettant aux enfants de s'endormir l'esprit tranquille. Mais les enfants ne sont pas les seuls à profiter de ce genre d'accessoires qui évoluent constamment. Les parents passent en effet de meilleurs nuits grâce aux babyphones et autres gadgets permettant de faire un lien entre eux et leur enfant dormant à l'étage.

Ayant moi-même eu beaucoup de mal pour dormir étant jeune, je me suis senti concerné par le projet de Veilleuse connectée lancé par mes prédécesseurs dans le cadre des projets de 4eme année à Polytech Lille. J'ai donc décidé d'améliorer cette veilleuse qui rassemblait déjà une base solide de fonctions.

Dans une première partie je ferai une présentation générale du projet, puis j'expliquerai la conception des améliorations apportées à celui-ci. Je détaillerai pour finir les différentes difficultés rencontrées.

1. Présentation du projet

a. Objectif

L'objectif du projet était d'améliorer la veilleuse connectée existante en la rendant nomade, et de rajouter des fonctionnalités et des capteurs lui permettant de renforcer le lien entre l'enfant et ses parents

La veilleuse pourra en plus des fonctions de base :

- Etre nomade, c'est à dire fonctionner sur batterie
- S'allumer et s'éteindre à l'aide d'un bouton sans endommager le système
- Réceptionner des informations reçues par des capteurs.

Je devrai donc créer un circuit permettant de gérer l'alimentation délivrée à la Raspberry pi, son extinction et son allumage. Je procèderai à quelques configurations sur le code de la veilleuse lui permettant de faire un lien avec le circuit. Une boite devra être construite pour contenir l'ensemble des composants de la veilleuse.

b. Choix matériel

En ce qui concerne la gestion des différentes fonctionnalités, j'ai repris la Raspberry Pi 2B sur laquelle mes prédécesseurs avaient travaillé. Cette Raspberry Pi fonctionne sous la distribution Raspbian.

La projection d'image se fera toujours par le biais du picoprojecteur ASUS S1 qui sera branché sur le port HDMI de la Raspberry Pi. Ce projecteur possède une batterie permettant de ne pas le brancher à une prise. La diffusion de musique se fera également par le biais de la sortie HDMI car le projecteur dispose d'un haut-parleur intégré.

J'utilise, de plus, deux batteries pour faire fonctionner mon circuit d'alimentation et donc ma Raspberry Pi. Ce sont deux batteries 5V 1A.
J'ai pensé à économiser une batterie en utilisant l'alimentation que possède le pico projecteur mais lors des phases de test j'ai rencontré des problèmes d'extinctions incontrôlés. J'ai donc opté pour une alimentation séparée.

Le même dongle Wifi que l'année dernière sera utilisé pour permettre la connexion à celle-ci depuis un appareil.

Coté logiciel je n'ai rien touché, mais les mêmes paquets que l'année dernière sont installés. Le serveur apache est toujours actif et l'adresse de connexion est inchangée (192.168.100.1).

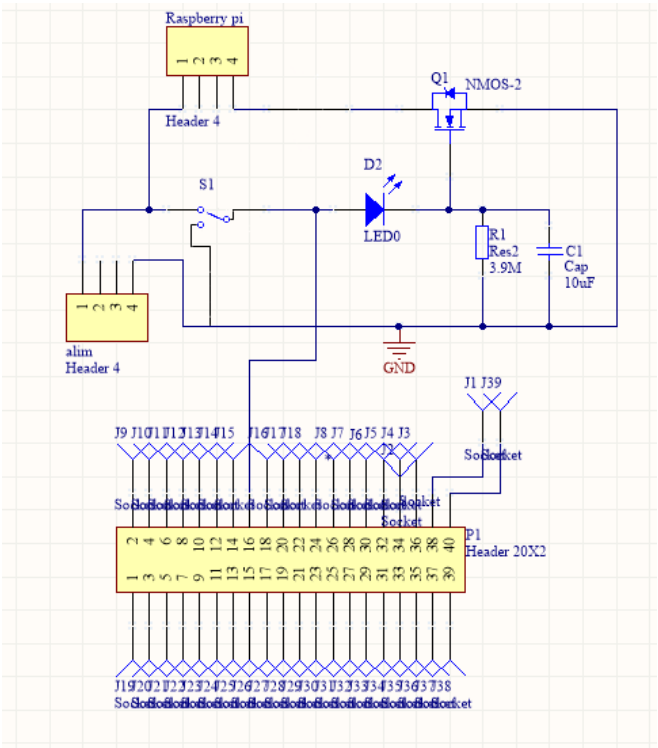
2. Conception

a. Circuit de démarrage

Pour un système embarqué, la gestion de l'énergie est très importante. En effet si le système est portable mais consomme énormément d'énergie, nous serons obligés de le recharger constamment. Ce n'est évidemment pas ce que nous voulons.

La Raspberry Pi restant constamment allumée tant qu'elle est branchée à une source de tension, il me fallait mettre un interrupteur permettant de couper cette source. Toutefois il est important d'arrêter le système grâce à une commande spécifique « sudo halt » avant de le couper électriquement parlant.

Je me suis donc lancé dans un circuit de démarrage qui se trouvera entre la Raspberry Pi et la source d'alimentation (mes batteries). Ce circuit, sur l'enclenchement d'un interrupteur, enverra un signal à la Raspberry Pi qui lancera d'elle-même la commande d'extinction. L'alimentation sera coupée quelques secondes après l'enclenchement de l'interrupteur permettant au système de s'éteindre « proprement ».



Le circuit électrique final retenu

L'idée du circuit est de contrôler l'alimentation délivré grâce à un transistor. Pendant que la Raspberry fonctionne, une capacité se charge (C1). Lorsque nous voulons arrêter la Raspberry Pi il nous suffit d'enclencher l'interrupteur S1. La capacité prend donc le relais de la batterie pour tenir le transistor saturé et donc garder la Raspberry Pi allumée un certain temps.

De plus, l'enclenchement du bouton provoque une chute de tension de la pin 16 du Header P1. Cet Header représente les broche GPIO de la Raspberry Pi. Je me sers de cette broche comme capteur. Lorsqu'une chute de tension se produit au niveau de cette broche, le système de la Raspberry Pi s'éteint grâce à un petit script que j'ai ajouté dedans. (Voir wiki ou annexes)

Ce Script python nommé ShutDown.py est assez simple. On configure la ou les, broches qui nous intéresse. Ici on mais la GPIO7 en entré pour pouvoir recevoir quelque chose.
On met cette broche en écoute et on demande de lancer une fonction « extinction » lors de la capture d'une chute de tension (traduit par un 0 logique).

La fonction « extinction » permet juste l'exécution de la commande « sudo halt ». Etant donné que ce programme tourne en boucle en attendant l'extinction, on ajoute une ligne demandant au programme de se mettre en veille régulièrement pour éviter d'occuper toute la mémoire du CPU :

While 1
Time.Sleep(0.02)

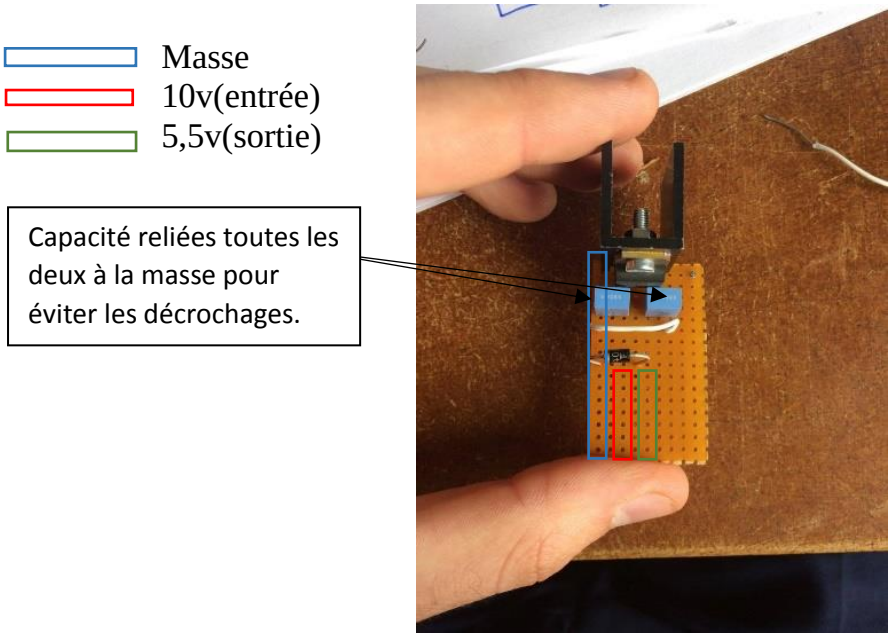
Maintenant que le script est prêt, je dois faire en sorte qu'il se lance au démarrage. Je vais donc dans le dossier /etc/init.d et j'ajoute un nouveau script « ShutDown » appelant ShutDown.py au démarrage.
En effet les scripts dans ce dossier peuvent être lancés au démarrage en tapant juste une commande :

update-rc.d ShutDown defaults

Le dossier rc.d représentant des liens vers les ficher init.d qui seront lancés au démarrage. Nous voilà parés pour éteindre correctement notre système.

b. Circuit du régulateur de tension

Un problème c'est posé lorsque j'ai réalisé le montage décris au-dessus. Ma Raspberry Pi ne s'allumait pas. Après quelques recherches j'ai constaté qu'elle n'avait pas assez de tension à ses bornes (4,5V) alors que la tension de la batterie était de 5V. Certains éléments du circuit consommaient du courant. Je devais donc rajouter une batterie en série de la précédente. J'avais donc 10V. Il m'a fallu réaliser un régulateur ayant en sortie une tension de 5,5V permettant l'allumage de ma Raspberry Pi.



Montage de mon régulateur

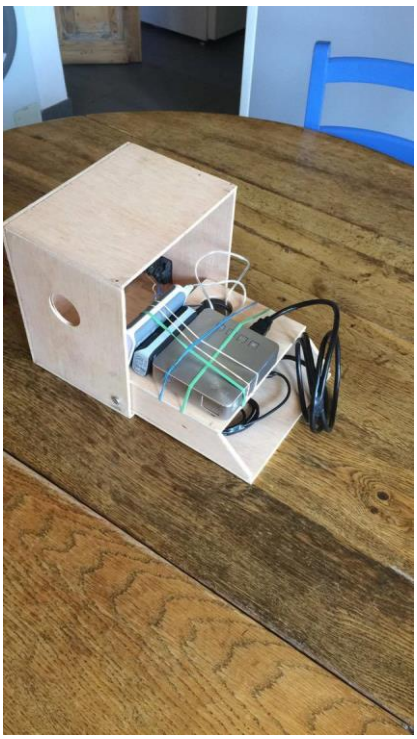
Ce régulateur 7805C peut recevoir une tension entre 7 et 25V et délivre une tension de 5v en sortie si la broche commune est reliée à la masse.
Le régulateur a trois broches : une entrée, une sortie et une commune permettant de fixer la masse. Ici j'ai rajouté une diode qui grâce à sa tension de seuil fixe le potentiel de la broche commun à 0,6v. Ce qui permet de récupérer 5,6v en sortie.

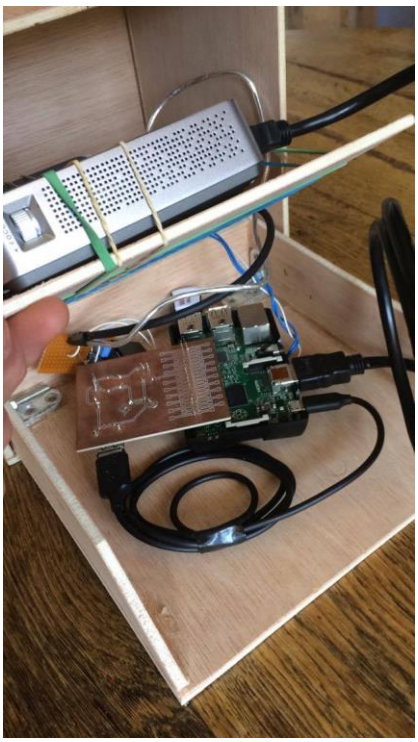
c. La Boite

Pour la boite, j'ai choisi de travailler du bois moi-même, ayant les outils nécessaires chez moi. Elle comporte un interrupteur extérieur pour l'allumage de la Raspberry Pi. Je n'ai pas pu relié l'allumage du rétroprojecteur au bouton car je n'avais pas le droit de le démonter. Il faut donc ouvrir la boite pour le lancer.

Elle est composée en deux étages. Celui du dessus avec le rétroprojecteur et les batteries, et celui du dessous comportant les circuits et la Raspberry Pi. Les câbles passent à travers les étages grâce à un trou.

J'ai fixé les éléments avec des élastiques. Evidemment il est préférable de les fixer définitivement mais par soucis pratique du recyclage du matériel, j'ai préféré laisser ça comme cela. Pour que le rétroprojecteur puisse projeter hors de la boite il était évident de devoir faire un percement au bon endroit pour laisser l'image se diffuser à l'extérieur.





3. Limites du Projet

J'ai réussi dans ce projet à optimiser une base déjà existante pour la rendre plus confortable dans son utilisation. Toutefois je n'ai pas pu exploiter toutes les idées que j'avais.

Comme vous avez pu le constater je n'ai pas eu le temps d'exploiter la partie sur les capteurs. Il aurait été très intéressant de pouvoir contrôler l'environnement dans lequel se trouve l'enfant. Par exemple connaitre la température, l'humidité ou encore capter les pleurs d'un bébé pour prévenir les parents.

J'avais de plus en tête le rajout d'une fonction « veilleuse classique » qui consistait en l'allumage et l'extinction d'une lumière tamisé pour l'enfant. Mon idée était de créer un circuit à l'aide d'un transistor contrôlé par une des broches GPIO configurée en sortie. L'alimentation de cette lumière aurait été géré par les mêmes batteries déjà présentes.

Des modifications sont donc toujours possibles et cela en ferait de mon point de vu un bon projet pour les années ultérieures.

Conclusion

En conclusion, ce projet a été pour moi le moyen d'enrichir mes connaissances sur la Raspberry Pi ainsi que sur la création de PCB. J'ai pu de plus me servir de mes simples connaissances de travail du bois pour construire une protection sobre mais efficace.

Ce fût mon premier projet autonome. Au cours de ces douze semaines, j'ai donc pu mettre en pratique les connaissances apprises durant mes deux premières années au sein de la section IMA à Polytech.

Cette expérience m'a permis de me mettre dans des situations que je pourrai rencontrer dans un avenir proche. J'ai dû apprendre à respecter les échéances fixées, à tenir un compte-rendu d'activités hebdomadaire, à prendre mes responsabilités, et à trouver des alternatives faisant suite à certains problèmes.

Au final je suis assez fier d'avoir pu m'attaquer, seul, à un projet déjà en cours et d'avoir réussi à apporter ma pierre à cet édifice.

ANNEXES

Raspberry pi 2B

Asus S1

```
#!/usr/bin/env python2.7

# on importe les modules necessaires
import time
import os
import RPi.GPIO as GPIO

# on met RPi.GPIO en mode notation BCM (numero des pins)
GPIO.setmode(GPIO.BCM)

# on initialise le GPIO 23 en mode ecoute
GPIO.setup(7, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# on definit notre fonction qui sera appelee quand on appuiera sur le bouton
def extinction(channel):
    # on affiche un petit message pour confirmer
    print("Appui detecter le GPIO 7")

    # on reinitialise les GPIO
    GPIO.cleanup()
    # on lance la commande d extinction
    os.system('sudo halt')

# on met le bouton en ecoute
GPIO.add_event_detect(7, GPIO.FALLING, callback=extinction)

# on lance une boucle infinie, pour garder le script actif
while 1:
    # une petite pause entre chaque boucle, afin de reduire la charge sur le C$
    time.sleep(0.02)

# on reinitialise les ports GPIO en sortie de script
GPIO.cleanup()
```

Script d'extinction du system