h e g

Haute école de gestion
Genève

Pilotage d'un dispositif domotique depuis une application Android

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Douglas Michael SIFFERT

Conseiller au travail de Bachelor :

David BILLARD

Carouge, le 12 décembre 2014

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Hes·so GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre.

L'étudiant a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : http://www.urkund.fr/student_gorsahar.asp.

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Vernier, le 12 décembre 2014

Douglas SIFFERT

Résumé

Les dispositifs domotiques sont nombreux, mais la plupart du temps il s'agit de systèmes propriétaires.

Pour ce travail j'ai choisi de construire un système permettant de commuter des prises électriques afin d'allumer ou éteindre les objets s'y branchant. J'ai analysé les solutions existantes, les protocoles utilisés et finalement construit une maquette.

Le système est géré par un mini-ordinateur Raspberry Pi sur lequel sont connectés plusieurs circuits intégrés : une RTC, un capteur de température et un contrôleur pour relais. La partie logicielle comprend un agent WEB permettant de piloter les relais et lire la température de la pièce, ainsi qu'une API utilisée pour qu'une application Android, également développée pour l'occasion, offre les mêmes fonctionnalités que l'agent WEB.

Des tests précis sont effectués pour valider chaque étape du développement.

Table des matières

Liste des tableaux

Liste des figures

1. Introduction

La domotique ne date pas d'aujourd'hui, d'ailleurs c'est un terme plutôt vague, n'importe quel système permettant d'interagir avec le bâtiment pour en améliorer le confort est déjà un système domotique.

Il y a quelque temps, la domotique c'était les lumières qui s'allumaient lorsque la nuit tombait mais avec l'évolution des technologies informatiques l'utilisateur a de plus en plus de contrôle sur les fonctionnalités de son logement.

Lorsqu'on parles de systèmes libres tels que le système Android on se rends vite compte de la diversité des projets. C'est pourquoi je vais m'y intéresser moi aussi, étant autant passionné d'informatique que d'électronique.

2. Énoncé du problème

Nous souhaiterions utiliser une solution domotique afin de piloter un ou plusieurs dispositifs électriques, à distance, à l'aide d'une application Android.

Les dispositifs en question sont les produits électriques disposant d'un interrupteur mécanique. En effet, certains objets disposent d'un mode veille[1] et ne s'allumeront pas par simple apport de courant ; ces objets doivent être modifiés ou inclurent une commande à distance d'origine. Les autres, comme les lampes, ventilateurs ou moteurs de stores sont les produits visés par la solution à développer.

La solution imaginée sera composée d'un boîtier contenant un circuit intégré permettant de connecter plusieurs relais à un réseau sans fil. Un ordinateur miniature « Raspberry Pi » ou similaire fera normalement l'affaire. Le boîtier sera branché à une prise secteur murale standard et connectera plusieurs autres prises de type 13 par l'intermédiaire des relais. Depuis le périphérique Android, l'écran de l'application affichera la température de la pièce et un bouton par prise à piloter, presser sur un bouton changera l'état de la prise concernée. En gros, il s'agit d'un bloc multiprise intelligent pilotable à distance.

Bien entendu la solution pourra changer au cours de l'étude mais plus après commande des pièces.

Une recherche des solutions existantes sera effectuée ainsi qu'un tri entre les systèmes propriétaires et les systèmes libres, afin de poursuivre avec une analyse des protocoles dans le but de réaliser la solution matérielle et logicielle en utilisant des composants libres autant que possible. Une maquette fonctionnelle sera finalement présentée.

[1] Exemple : une télévision

3. Solutions existantes

Évidement de nombreuses solutions existent déjà, mais toutes ne sont pas libres.

Après recherches, les solutions peuvent être classées en quatre catégories :

- Les solutions complètes, personnalisables, le plus souvent multiplateformes, mais propriétaires et destinées aux architectes et monteurs électriciens.
- Les solutions architecturées autour d'un système Android et nécessitant l'achat d'un peu de matériel propriétaire[2] destinées aux particuliers.
- Les applications Android, dont certaines développées par des particuliers, mais pour des systèmes propriétaires.
- Les composants discrets pour lequel on développe ses propres applications.

3.1 Solutions pour professionnels

CS Domotic : propose un système PC, Mac, iPad, Android adapté aux besoins personnalisés du client et utilisant le système KNX et les composants EnOcean pour piloter la maison.

Hager propose le système Tebis[3] piloté par le logiciel Domovea sous iPad et Android.

Ces solutions ne sont pas adaptées à ce travail car tout est propriétaire, et elles se destinent à l'intégration dans un bâtiment complet ou à l'adaptation d'un système existant.

3.2 Solutions pour particuliers

Steffen propose une sélection de composants issus de plusieurs marques qu'un particulier peut acheter dans un magasin de bricolage[4] pour monter lui-même son système personnalisé. Il y a un présentoir dans les rayons des magasins, mais aucune information sur leur site. Ces composants peuvent être relativement coûteux.

Zodianet propose un boîtier, le Zibase Classic, multi-protocoles incluant une licence payante. Le Zibase Mini est composée d'un Raspberry Pi et propose les mêmes prestations. Finalement le Zibase Multi est l'alternative entièrement basée sur Android. Ces deux dernières solutions ont soit une licence gratuite à renouveler tous les trois mois, soit une licence unique à vie pour 100€.

Homidom est un logiciel Windows gratuit et open-source pour lequel une communauté développe des drivers supportant de très nombreux protocoles et composants matériels comme les kits Velleman ou les cartes Arduino. Il faut donc un simple PC

[2] Souvent disponible en magasin
[3] Repose sur KNX
[4] Jumbo, Hornbach, etc.

comme serveur mais les solutions ITX permettent de disposer d'un serveur compact et économique. Un client Android est disponible pour piloter et surveiller le système.

Ces solutions sont très intéressantes mais sont destinées à utiliser des composants domotiques existants issus de plusieurs systèmes éventuellement concurrents, d'où le support multi-protocole et il faut généralement faire appel à un électricien pour se procurer ces composants.

3.3 Les applications de Google Play

Les applications Android disponibles sont soit officielles soit développées par des particuliers, mais toujours destinées à utiliser des composants domotiques existants. Le but de cette étude étant de développer une solution, ces applications sont à proscrire dans ce travail mais restent éventuellement intéressantes pour comprendre certains protocoles.

Tableau 1 : Quelque exemples d'applications

Système	**Appli officielle**	**Appli non-officielle**
VeraLite / Vera 3 de Micasaverde	Home Buddy	AutoHTN Vera, AutHomation, Vera Mobile
Zibase de Zodianet	Zodianet	HCSa
eedomus de Connected Object	eedomus	eeBud Flower
Logiciel HomeSeer	HSTouch	DroidSeer
Logiciel Domogik	Domodroid	HSDroid
My Fox de Tag Technologies	My Fox	
Myxyty de M2M Solution	Myxyty	
logiciel AndFHEM, RFXCOM	AndFHEM	
Émetteur RFXCOM	Jmom NG 2	
box domotique Blyssbox de Castorama	Blyssbox	

(http://blog.domadoo.fr/2012/08/28/les-applications-domotiques-pour-android/)

3.4 Les composants discrets

Dans le but de réduire les coûts mais aussi de disposer de systèmes ouverts tant au niveau logiciel que matériel, plusieurs communautés ont lancés des projets domotiques visant à interfacer des mini-ordinateurs Raspberry Pi, Arduino, etc, à des composants existants ou des composants crées par l'utilisateur[5]. Dans ce cas un peu de développement matériel est requis mais pas forcément de grandes connaissances en électronique car les projets existants fournissent généralement une carte à connecter directement au mini-ordinateur, drivers, librairies et exemples de code compris, pour un faible coût.

Certains projets ne sont que des cartes à relais génériques pour microcontrôleurs et utilisent une interface standard série ou I2C, d'autres possèdent un connecteur pour se brancher directement sur le port GPIO du mini-ordinateur concerné. Ces dernier sont souvent disponibles avec des drivers et exemples de codes, ce qui est une bonne chose.

En recherchant des cartes à relais sur eBay, j'ai remarqué que l'on en trouvait de toutes sortes, certaines seules, d'autres avec tout ce qu'il faut pour les connecter à l'ordinateur. Pour certaines il peut y avoir jusqu'à trois connexions au choix : USB, GPIO série ou GPIO I2C. Le nombre de relais disponibles est généralement une puissance de 2 : 1, 2, 4, 8, 16 ou même chaînables deux à deux.

Quoi qu'il en soit, le choix se portera sur la documentation disponible, le nombre de relais, quatre étant parfait pour cette maquette, le prix de la carte et le pays de provenance, car le temps est limité pour ce travail et la livraison peut atteindre trois semaines suivant dans quel pays la carte est commandée. Une commande ça va, mais en cas de problème un second envoi poserait problème.

[5] Ce que les solutions précédentes permettent plus difficilement

4. Choix d'une solution

Afin de décider quelle solution conviendra à ce travail, voici un résumé :

Tableau 2 : Résumé des solutions

	Avantage	**Inconvénient**	**Prix**
CSDomotic	Complet	Propriétaire	Installateurs
Hager	Complet	Propriétaire	Installateurs
Zodianet	Complet	Pour matériel propriétaire	Gratuit ou 100 €
Homidom	Libre	Selon disponibilité des drivers	Gratuit
Applications	Choix matériel	Pour matériel propriétaire	Gratuit ou < 5 €
Cartes à relais	Générique	Support	~ 20 €

Les solutions CSDomotic et Hager sont retirées car il s'agit de matériel propriétaire à faire installer chez soi au prix d'une installation par monteur électricien qualifié ; ne convient pas à la construction d'une maquette sauf s'il s'agit d'une pièce entière.

La solution Zodianet et retirée car bien qu'intéressante dans ce cas elle se destine à du matériel standard, souvent propriétaire, que l'on trouve dans un logement, des composants domotiques souvent à 40 € pièce.

La solution Homidom est un projet très intéressant car entièrement libre et gratuit mais il se destine aussi au matériel domotique existant dont le support n'est possible qu'à condition que la communauté développe le driver concerné.

Les applications présentes sur Google Play sont semblables à Zodianet ; elles sont faites pour piloter du matériel existant trop coûteux pour ce travail. Les meilleures sont propriétaires et sont là pour que le client indécis puisse choisir un constructeur d'après le confort d'utilisation de l'application.

La solution retenue est l'utilisation de composants discrets tels qu'une carte à relais générique ce qui permet d'intégrer d'autres fonctionnalités comme la lecture de la température puisque tous ces composants peuvent être utilisés en même temps sur un mini-ordinateur[6]

[6] Avantage majeur du port série

5. Analyse des protocoles

Pour commencer je vais analyser les protocoles standards généralement utilisés par les composants domotiques afin de connaître la pertinence de chacun dans ce projet et de retirer les protocoles propriétaire. Une première recherche effectuée permet de trier lesquels sont propriétaires ; un protocole propriétaire nécessite généralement de pratiquer le reverse engineering pour comprendre son fonctionnement ce qui dépasserait le temps à disposition.

Voici la liste des protocoles courants, il en existe d'autres mais il s'agit souvent d'un système exclusif à un magasin de bricolage qui repose sur un de ces protocoles :

Tableau 3 : Liste des protocoles disponibles

Protocole	**Libre**	**Conçu pour**
Zwave	Non	Domotique
6LoWPAN	Oui	Réseaux faible consommation
EnOcean	Non	Domotique
ZigBee	Oui	Réseaux PAN
RFY	Non (Somfy)	Domotique (stores)
X2D / X3D	Non	Domotique haut de gamme
KNX	Non	Domotique (installateurs)
Chacon	Non	Domotique faible coûts
X10	Oui	Domotique par courant porteur
Domia	Non	Domotique (variante de Chacon)
Visonic	Non	Sécurité
Scientific Oregon	Non	Dispositifs météo d'intérieur
Bly	Non (Castorama)	Domotique
HTTP	Oui	Transfert de données

Les systèmes Zwave et EnOcean ne sont pas libres mais sont des middlewares destinés à être utilisés entre un dispositif existant[7] et les capteurs éventuellement fabriqués par d'autres constructeurs. Ils offrent donc une plus grande flexibilité qu'avec les autres systèmes propriétaires.

[7] Ordinateur ou réseau LAN

On remarque tout de suite qu'à part le protocole X10 les autres protocoles libres ne sont pas dédiés à la domotique. On trouve généralement de petits modules à faible coût comme une antenne que l'on peut intégrer dans son montage. Dans le cas de HTTP, la communication se fait entre logiciels pilotant le matériel[8]. D'une façon générale les solutions libres sont à adapter à son utilisation.

Voici une comparaison plus détaillée dans le but de savoir si les protocoles libres permettent d'atteindre les objectifs fixés pour ce travail pour lequel il faudra développer un minimum :

Tableau 4 : Protocoles libres

Protocole	**Inconvénient**	**Manque**
6LoWPAN	Juste un réseau	Composants domotiques
ZigBee	Juste un type d'antenne	Composants domotiques
X10	Réseau par courant porteur	Interface avec Android
HTTP	Juste un protocole de transfert	Interface avec Android

Le protocole HTTP se prête particulièrement bien à l'utilisation d'un mini-ordinateur car il n'impose pas de matériel. Les protocoles 6LoWPAN et ZigBee sont conçus pour permettre le fonctionnement de composants à très faible énergie comme les capteurs et actionneurs fonctionnant sur piles. Les mini-ordinateurs ont une alimentation constante et le problème de la longévité des piles ne se pose pas. Les protocoles 6LoWPAN et X10 sont des types de réseau qui pourraient tout à fait être interfacés par le port USB d'un mini-ordinateur mais le problème se pose lors de la communication avec le périphérique Android qui utilise le Wi-Fi et/ou la 3G/4G, le mini-ordinateur ne devient plus qu'un pont entre les deux réseaux ce qui impose que l'application Android intègre directement la gestion de ces protocoles.

Le protocole HTTP nous laisse choisir le matériel, donc la possibilité d'utiliser le Wi-Fi pour connecter les périphériques Android. Il ne restera qu'à développer une API pour communiquer avec l'application Android et éventuellement un agent Web pour utiliser des ordinateurs de bureau en plus des téléphones/tablettes Android. Avec le choix du protocole HTTP, on utilisera une architecture très répandue pour communiquer avec les périphériques mobiles. Le prix du matériel est faible puisqu'il suffit de se procurer des projets matériels de carte à relais, capteurs divers compatibles avec le Raspberry Pi ou autre plus une poignée de composant courants (prises, câbles, boîtier) que l'on

[8] Application client – API serveur – interface physique – composant domotique

peut acheter partout. La création des composants domotiques pour les autres protocoles peut poser un problème de disponibilité des composants et de respect des délais.

6. Le protocole HTTP

C'est un protocole de transfert de données entre un client et un serveur créé à l'origine pour le Web et la navigation par liens hypertexte[9].

Un client, le plus connu étant le navigateur Web, envoie une requête HTTP au serveur, généralement un programme tel qu'Apache. La requête est constituée d'une commande (méthode), de l'URL du document et de la version du protocole HTTP, le tout suivi d'une en-tête comprenant des informations sur la requête et d'un corps comprenant des données devant être envoyées au serveur. L'en-tête et le corps sont généralement facultatifs mais cela dépend de la méthode.

6.1 Format

Formellement une requête HTTP a la syntaxe suivante :

(<crlf> signifie retour chariot ou saut de ligne) :

```
METHODE URL VERSION<crlf>

EN-TETE : Valeur<crlf>

Valeur<crlf>

Ligne vide<crlf>

CORPS DE LA REQUETE
```

Voici donc un exemple de requête HTTP :

```
GET http://www.commentcamarche.net HTTP/1.0

Accept : text/html

If-Modified-Since : Saturday, 15-January-2000 14:37:11 GMT

User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
```

[9] HTTP = HyperText Transfert Protocol

Voici la liste des commandes :

Tableau 5 : Les commandes HTTP

Commande	**Description**
GET	Requête de la ressource située à l'URL spécifiée
HEAD	Requête de l'en-tête de la ressource située à l'URL spécifiée
PUT	Envoi de données au programme situé à l'URL spécifiée
POST	Envoi de données à l'URL spécifiée
DELETE	Suppression de la ressource située à l'URL spécifiée

La Réponse est constituée d'une ligne de statut indiquant la version du protocole, d'un code de statut et de sa signification, le tout suivi d'un en-tête et d'un corps qui contient le document voulu.

Une réponse HTTP a la syntaxe suivante :

(<crlf> signifie retour chariot ou saut de ligne) :

VERSION-HTTP CODE EXPLICATION<crlf>

EN-TETE : Valeur<crlf>

Valeur<crlf>

Ligne vide<crlf>

CORPS DE LA REPONSE

Voici donc un exemple de réponse HTTP :

HTTP/1.0 200 OK

Date : Sat, 15 Jan 2000 14:37:12 GMT

Server : Microsoft-IIS/2.0

Content-Type : text/HTML

Content-Length : 1245

Last-Modified : Fri, 14 Jan 2000 08:25:13 GMT

Document voulu

6.2 Versions

HTTP 0.9 : version initiale, avant que HTTP ne soit décrit dans une RFC, très simple car la seule commande est GET et le serveur revoie le fichier tel quel sans métadonnées. Cela permettait déjà de naviguer dans le Web

HTTP 1.0 : (RFC 1945) ajoute un en-tête (RFC 822) pour transférer des métadonnées et ajoute les commandes HEAD et POST.

HTTP 1.1 : (RFC 2616) améliore la gestion du cache et n'utilise plus que des connexions permanentes en TCP, tout ceci dans le but de réduire le nombre de connexions pour chaque documents ; la complexité du WEB augmentant. Toutes les autres commandes sont ajoutées.

HTTP 1.1b : (RFC 7230 à 7237) publication de 8 nouveaux documents en juin 2014 pour pallier aux imprécisions de la RFC 2616.

HTTP 2.0 : en cours de développement par l'IETF suite à l'émergence de SPDY de Google.

6.3 Utilité de HTTP dans ce projet

Plusieurs périphériques peuvent utiliser le protocole HTTP pour demander des données, à savoir :

Un ordinateur de bureau qui contrôle le produit à l'aide d'un navigateur WEB, ce qui permet la compatibilité de tous les systèmes avec la solution.

Un téléphone ou une tablette Android contrôlant le produit par l'intermédiaire de l'application[10]. L'application effectuera des requêtes HTTP pour demander des informations ou pour commander une API présente au niveau du serveur, dans ce cas on utilisera une requête HTTP simple et le pilotage se fera par l'URL vers le serveur PHP, par exemple.

[10] Évidement le contrôle par navigateur est possible pour iPhone, iPad, Windows Phone, etc.

7. Description de la solution

Le produit à concevoir doit comporter une partie logicielle à développer et être accessible financièrement pour le présent travail de Bachelor. La solution retenue est la construction d'un dispositif matériel architecturé autour d'un mini-ordinateur Raspberry Pi complété de quelques composants électroniques. Pour la partie logicielle à développer, il y aura une application Android.

Le Raspberry Pi est retenu car il est moins cher que ses concurrents et qu'il y a une énorme communauté l'utilisant pour des projets très divers. Un Arduino aurait pu faire l'affaire mais je trouve le concept ultra-modulaire un peu fragile.

La maquette sera composée de plusieurs composants placés dans un boîtier sur lequel le Raspberry Pi viendra se brancher par son câble GPIO. Le boîtier comporte une prise de courant pour se brancher au réseau électrique et alimente les quatre prises Type 13 interrompues par les relais. Un circuit fournit une tension de 5V pour le Raspberry Pi et les relais mais les circuits connectés au port GPIO[11] fonctionnent avec la tension de 3.3V fournie par le Raspberry Pi. Le contact 5V du port GPIO est un équipotentiel avec le connecteur mini-USB du Raspberry Pi qui n'utilise ce dernier que pour s'alimenter en courant ; les connecteurs à picots espacés de 2.54mm pouvant supporter jusqu'à 3A environ, on peut donc alimenter le Raspberry Pi par le port GPIO et il démarrera au moment de l'allumage du boîtier. Quatre LED sur l'avant du boîtier indiquent l'état des relais.

7.1 Recherche des composants

J'ai déjà commandé des composants électroniques dans plusieurs magasins et sur plusieurs sites (Conrad, Distrelec, eBay, etc.) et pour cette maquette je pense que je vais choisir eBay pour plusieurs raisons. Premièrement, depuis qu'eBay a mis en place le système d'évaluation à cinq étoiles, je n'ai constaté aucun manque de sérieux d'un quelconque vendeur. Ensuite, tous les vendeurs acceptent PayPal ce qui permet d'effectuer un paiement immédiat et de gagner du temps. Pour finir les délais de livraisons sont relativement constants et les commandes sont envoyées de suite. Commander dans les autres magasins par correspondance est un peu plus sûr, mais les délais de livraisons peuvent être variables et certains facturent en plus des frais de gestion de la commande.

[11] RTC, Capteur de température et contrôleur des relais

Par expérience, les délais des commandes eBay, lorsque le vendeur utilise la poste locale, ce qui est le moins cher, sont les suivants :

- Europe centrale : 1 semaine à 10 jours
- USA : 1 semaine à 10 jours
- Taïwan, Thaïlande : 10 jours
- Australie : 2 semaines
- Chine : 3 semaines

De nombreux vendeurs chinois ne facturent pas de frais de port sur eBay.

En cherchant des composants pour le Raspberry Pi, j'ai trouvé le magasin en ligne « The Pi Hut » qui vends du matériel pour mini-ordinateurs.

Les prises doubles Type13 sont difficiles à trouver, il y en a dans les magasins de bricolage mais ce n'est pas la bonne version car il faut la que deux prises soient séparées électriquement. S'il n'est pas disponible en magasin, il faut passer par une entreprise d'électricité pour changer le matériel, ce qui n'est clairement pas concevable pour la construction d'une petite maquette, mais j'ai trouvé un autre magasin en ligne qui vends du matériel électrique aux particuliers, avec un choix intéressant et qui ne facture pas de frais de port : www.elektrobedarf.ch

J'ai donc commandé le Raspberry Pi et ses accessoires sur The Pi Hut et eBay pour les recevoir rapidement, les composants électroniques (connecteurs, LED, etc.) chez des vendeurs asiatiques car ces composants sont plutôt pour la finition donc ils sont moins urgents et pour finir les prises et le boîtier chez Elektrobedarf.

7.2 La partie matérielle

Voici la liste des composants :

Tableau 6 : Liste des composants

N°	Description
01	Raspberry Pi
02	RTC pour Raspberry Pi
03	Câble GPIO
04	Connecteur GPIO pour implantation sur circuit imprimé
05	Pieds pour boîtier
06	Boîtier pour Raspberry Pi
07	Carte SD avec image de NOOBS + adaptateur Wi-Fi
08	Carte à relais avec contrôleur BV4111 et câble de liaison
09	Capteur de température Adafruit MCP9808
10	Circuit imprimé prototype 50x50mm
11	Connecteur réseau électrique + interrupteur + porte fusible
12	Circuit d'alimentation AC-DC 5V 1A
13	Plusieurs LED 3.3V
14	Barrettes de connecteurs sécables M
15	Barrettes de connecteurs sécables F
16	Boîtier pour installations
17	Prises 2x T13 séparées

Cette liste est complétée de quelques pièces me restant d'anciens projets.

Le Raspberry Pi (01) est un modèle B, équipé d'un port Ethernet et de deux USB.

La RTC (02) est une horloge équipée d'une pile qui conserve les paramètres temporels au cas où le serveur NTP est absent, car le Raspberry Pi est conçu pour utiliser un serveur NTP pour des raisons de coûts. La maquette étant faite pour fonctionner sans être systématiquement connectée à Internet les paramètres temporels seront conservés.

L'adaptateur Wi-Fi (07) est utilisé pour connecter les périphériques sans fil au Raspberry Pi, ce dernier n'ayant d'origine qu'une prise Ethernet RJ-45.

La carte SD (07) est livrée avec plusieurs images de systèmes d'exploitation pour le Raspberry Pi et on choisit celui que l'on a besoin, il s'agit de NOOBS : New Out Of the Box Software. J'utiliserai Raspbian Wheezy, une version de Linux Debian pour le Raspberry Pi.

Le boîtier (06) sert à protéger le Raspberry Pi, le boîtier (16) à recevoir les autres composants et les prises Type13 (17).

Le Raspberry Pi (01) sera connecté au boîtier de commutation (16) par le câble GPIO (03). Le boîtier de commutation (16) contiendra quatre prises branchées sur relais. Les relais sont connectés au connecteur GPIO (04) par le contrôleur série BV4111 (08).

Le branchement au réseau électrique se fait par le connecteur (11) qui contient aussi l'interrupteur principal et un porte-fusible pour des raisons de sécurité.

Le petit circuit d'alimentation (12) fournit la tension de 5V requise pour le Raspberry Pi et les relais (08). Cette solution permet d'éviter que ce soit le Raspberry Pi (01) qui fournisse le 5V à la carte à relais (08).

Le boîtier de commutation fournira l'alimentation au Raspberry Pi qui démarrera au moment de l'allumage ce qui est moins brusque et fournira la tension de 3.3V requise pour les circuits intégrés RTC (02), MCP9808 (09), BV4111 (08) et les LED (13). Une intensité de 1A devrait suffire car les relais ne consomment que 20mA et le Raspberry Pi peut fonctionner avec un chargeur de téléphone qui dépasse rarement les 750mA. En effet je n'ai pas besoin de beaucoup de performance pour ce projet : un point d'accès et un serveur WEB + PHP.

La platine d'expérimentation (10) recevra les connecteurs qui relieront les circuits intégrés entre eux plus les LED et le Raspberry Pi. La platine prendra sa place dans un emplacement similaire à celui des prises Type13.

Les LED (13) seront branchées à la carte à relais, cette dernière fonctionnant en logique négative on peut relier la cathode de chaque LED au contact concerné de la carte à relais. Chaque LED indiquera l'état d'un relais, donc d'une prise ; cette solution permet de diagnostiquer d'éventuels problèmes.

Le circuit MCP9808 (09) est un capteur de température qui sera installé à l'extérieur du boîtier, ainsi les mesures ne seront pas influencée par l'alimentation qui pourrait chauffer un peu.

Le MCP9808 et la RTC sont connectés sur le même bus I2C car plusieurs circuits peuvent cohabiter.

7.3 La partie logicielle

Le dispositif doit pouvoir être commandé depuis une application Android, donc le protocole HTTP sera utilisé pour transférer les requêtes, ce qui permettra en plus de commander le dispositif depuis n'importe quel périphérique utilisant un navigateur WEB. La liaison entre la commande HTTP (l'URL) et l'action exécutée par le Raspberry Pi se fera par une API également à développer. Je vais devoir décider du protocole de communication entre l'API et les clients. Après quelques recherches j'apprends que de nombreux Framework PHP sont disponibles et qu'il est fortement conseillé de les utiliser.

J'en ai retenu deux qui possèdent une bonne réputation et pour lesquels le code à écrire est clair : Phalcon PHP et Slim. Le problème de Phalcon est qu'il est relativement lourd pour le Raspberry Pi bien qu'il s'agisse du Framework PHP le plus rapide. En effet il faut le compiler et les 512Mb du Raspberry Pi sont insuffisants, car 2Gb sont conseillés. De plus Phalcon permet de développer des « vraies » applications complètes ce qui en fait un Framework recommandé pour les grands projets. Compiler Phalcon sur un autre ordinateur et le transférer sur le Raspberry Pi pour n'utiliser qu'une infime partie de ses capacités ne vaut pas le coup. En effet une API telle que celle dont j'ai besoin est considérée comme une micro-application et justement Slim est un micro-Framework qui est fait pour les petits projets. Il suffit de le télécharger dans le dossier du projet et quatre lignes de code plus les routes suffisent à son utilisation.

Un agent Web de base est disponible pour administrer le Raspberry Pi, je pensais rajouter un onglet pour ajouter mes fonctionnalités et y intégrer l'API mais le « pi-web-agent » ne s'installe pas dans le dossier publique du serveur WEB donc je ne modifierai pas pi-web-agent. Je développerai ma page servant d'agent WEB pour piloter les relais et placerai l'API à proximité. Le pi-web-agent m'est toutefois utile car il possède un outil permettant de visualiser l'état du connecteur GPIO et un outil permettant de voir les services activés. J'utiliserai donc le pi-web-agent pour diagnostiquer des problèmes s'il y en a.

7.4 Le protocole de communication client-serveur

L'API fournira les réponses aux requêtes sous forme de chaîne Json. Il n'est pas obligatoire d'utiliser le format Json, la commande « echo » de PHP suffit à retourner du texte, mais l'application Android sera faite pour récupérer un objet Json en utilisant un couple « clé - valeur ». L'API appellera des fonctions PHP qui exécuteront des commandes Linux, il n'y a besoin que de requêtes « GET », l'application ne faisant que consulter des données. Les commandes nécessaires sont les suivantes :

- Connaître le nombre de relais afin d'activer le nombre de boutons voulu pour l'outil[12] car la solution pourra fonctionner pour 8 relais maximum, le nombre étant stocké dans une base de donnée.
- Lire la température de la pièce.
- Commuter un relai. Un bouton n'allume pas ou n'éteint pas un relai, mais change son état.
- Allumer tous les relais.
- Eteindre tous les relais.

Je voulais implémenter une fonction de réaffichage de la température, mais après réflexion, ce n'est pas utile car la température ne change pas de façon significative dans un laps de temps court.

J'installerai une base de données car j'ai pensé à quelques fonctionnalités future, et notamment à la centralisation d'informations entre le Raspberry Pi et les périphérique s'y connectant. Je l'utiliserai pour stocker le nombre de relais et choisirai SQLite pour sa légèreté. De plus il vient avec son propre interpréteur de commande ce qui permet de taper du code SQL dans le Shell de Linux.

[12] Agent WEB ou application Android

8. Installation du système

8.1 Installation du Raspberry Pi

Il n'y a rien de particulier à faire si ce n'est l'assemblage du mini-ordinateur et l'installation d'une distribution Linux. Lors du premier démarrage de NOOBS une liste de différents systèmes s'affiche, on en choisit un et l'image est installée. Il y a Debian pour Raspberry Pi, Raspbian, XBMC et différents modes de démarrages. Une icône permet de choisir entre l'installation depuis la carte SD et l'installation depuis internet, si une connexion réseau est établie. J'ai installé Raspbian car c'est très rependu et les fabricants du matériel que j'ai acquis l'utilisent dans les tutoriels. En cas de problème, on peut afficher à nouveau la liste en appuyant sur « MAJ » lors du démarrage. L'installation en elle-même ne pose aucun problème car tous les paramètres sont auto-détectés. Pour le premier démarrage de Raspbian un écran s'affiche pour changer quelques options comme le mot de passe. Après cela, le démarrage se termine et on peut commencer à travailler.

8.2 Préparation du bus I2C

Avant toute chose il faut activer le bus I2C sur le connecteur GPIO. Le bus I2C est utilisé tel quel par les circuits intégrés pour communiquer entre eux. Seuls deux lignes sont nécessaires : SDA (Serial Data) et SCL (Serial Clock), aucun autre composant n'est utile, les adresses des circuits sont établies à l'initialisation de ceux-ci et ensuite la communication se fait de tel à tel circuit par programmation. Du fait qu'il est léger, le bus I2C est principalement utilisé à l'intérieur de produits complexes comme les ordinateurs et les télévisions ou dans des systèmes compacts comme les cartes à puces.

Pour utiliser le bus I2C du Raspberry Pi il faut commencer par charger les modules. Pour ce faire, saisir dans un terminal :

```
sudo nano /etc/modules
```

Et ajouter ces lignes à la fin :

```
i2c-bcm2708

i2c-dev
```

Puis redémarrer :

```
sudo reboot
```

Pour s'assurer du fonctionnement des circuits connectés, il est utile d'installer les outils I2C :

```
sudo apt-get install python-smbus

sudo apt-get install i2c-tools
```

En fonction de sa distribution on peut avoir un fichier « /etc/modprobe.d/raspi-blacklist.conf » et c'est mon cas, donc ouverture :

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Ajouter des « # » pour placer les lignes suivantes en commentaires :

```
blacklist spi-bcm2708

blacklist i2c-bcm2708
```

Et redémarrage :

```
sudo reboot
```

Pour finir lancer la détection des composants I2C pour s'assurer de leur fonctionnement :

```
sudo i2cdetect -y 1
```

Le « 1 » est valable pour un Raspberry Pi révision 2, le modèle B, version 512MB. Pour les modèles A, remplacer « 1 » par « 0 » car les deux ports sont intervertis d'une version à l'autre.

Si tout fonctionne, un tableau s'affiche avec quelques nombres correspondants à l'adresse sur le bus I2C des circuits concernés.

Dans mon cas deux nombres apparaissent : 0x18 qui correspond au capteur de température et 0x68 qui correspond à la RTC.

8.3 Installation de la RTC

Ma RTC est un circuit DS3231. Le vendeur me dirige vers la page de « learn.adafruit.com » pour l'utilisation du circuit DS1307, mais du moment que les deux circuits utilisent les même commandes il ne devrait pas y avoir de problème, d'ailleurs un avertissement précise qu'il est obligatoire que la distribution de Raspbian embarque le module DS1307 pour pouvoir installer une RTC. De nombreux circuits

sont sûrement compatibles. Le module équipé du DS3231 est fait pour se brancher directement sur le port GPIO du Raspberry Pi mais je le brancherai autrement.

Lorsque le circuit est détecté avec i2cdetect, on peut charger le module RTC :

```
sudo modprobe rtc-ds1307
```

Ensuite, en tant que root :

```
sudo bash

echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device

exit
```

Là aussi, valable pour un Raspberry Pi modèle B (i2c-0 sinon).

Lire le temps dans la RTC :

```
sudo hwclock -r
```

Donnée lue : « sam. 01 janv. 2000 01:07:48 UTC -0.977223 seconds », ce qui est normal pour un circuit vierge.

Mon Raspberry Pi étant connecté à Internet à ce moment-là, le temps du système est obtenu par un serveur NTP, confirmé par l'horloge à l'heure dans la barre des tâches.

Pour écrire les données temporelles du système dans la RTC :

```
sudo hwclock -w
```

Confirmation en relisant le temps depuis la RTC :

```
sudo hwclock -r
```

Il faut ensuite ajouter le module RTC à la liste des modules à charger au démarrage :

```
sudo nano /etc/modules
```

Et ajouter à la fin du fichier :

```
rtc-ds1307
```

Enfin, la création du composant DS1307 au démarrage :

```
sudo nano /etc/rc.local
```

Et ajouter avant la fin du fichier (exit 0) :

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device

sudo hwclock -s
```

Lors du prochain démarrage sans connexions Internet, les paramètres temporels seront conservés.

8.4 Installation du capteur de température

Il faut installer les librairies Python pour MCP9808, pour commencer les dépendances :

```
sudo apt-get update

sudo apt-get install build-essential python-dev python-pip python-smbus git
```

Puis la librairie RPi.GPIO :

```
sudo apt-get install RPi.GPIO
```

Enfin les librairies pour le capteur de température :

```
cd ~

git clone https://github.com/adafruit/Adafruit_Python_MCP9808.git

cd Adafruit_Python_MCP9808

sudo python setup.py install
```

Pour confirmer le fonctionnement, un programme de test est fourni avec les librairies, lancer :

```
cd examples

sudo python simpletest.py
```

Si tout fonctionne correctement, la température est affichée chaque secondes en Celsius et en fahrenheit, il suffit d'appuyer sur « Ctrl + C » pour quitter.

Le programme simpletest.py utilise le MCP9808 de la façon suivante :

```
# Chargement du module MCP9808.
import Adafruit_MCP9808.MCP9808 as MCP9808

# Une instance de MCP9808 est crée.
sensor = MCP9808.MCP9808()

# L'adresse par défaut est 0x18 et le bus est le seul disponible sur GPIO
# mais on peut changer celà :
# sensor = MCP9808.MCP9808(address=0x20, busnum=0).

# Initialise la communication avec le capteur.
sensor.begin()

# Boucle affichant les mesures chaque seconde.
print 'Press Ctrl-C to quit.'
while True:
        temp = sensor.readTempC()
        print 'Temperature: {0:0.3F}*C / {1:0.3F}*F'.format(temp, c_to_f(temp))
        time.sleep(1.0)
```

Les méthodes propres à l'utilisation du capteur sont donc, après import du module : MCP9808() (le constructeur), begin() et readTempC() qui retourne la mesure en Celsius.

8.5 Installation du contrôleur BV4111

Sur le Raspberry Pi, une interface sérielle appelée ttyAMA0 peut être utilisée comme un port série générique, mais l'interface est à la disposition du système par défaut et il faut la libérer pour l'attribuer à l'utilisateur sur le port GPIO, puis installer les librairies python.

Désactiver la sortie du log sur console :

```
sudo nano /boot/cmdline.txt
```

Retirer les termes :

```
console=ttyAMA0,115200 kgdboc=ttyAMA0,115200
```

Ensuite ouvrir :

```
sudo nano /etc/inittab
```

Et mettre en commentaires la ligne à la fin du fichier :

```
#Spawn a getty on Raspberry Pi serial line

#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Maintenant le système n'utilise plus le port série, ce qui évite les conflits.

Il ne reste plus qu'à installer les librairies :

```
sudo apt-get install python-serial
```

Le port est désormai « /dev/ttyAMA0 »

Quelques programmes sont disponibles pour tester le protocole SV3[13].

```
wget http://pin1.org/download/py/sv3.py

wget http://pin1.org/download/py/sv3test.py
```

On connecte juste le circuit mais pas les relais et on exécute :

```
python sv3test.py
```

Ou il y a le terminal Minicom :

[13] Protocole série alégé

```
sudo apt-get install minicom
```

Exécution en choisissant le nombre de bauds et le composant :

```
minicom -b 115200 -D /dev/ttyAMA0
```

Normalement, une fois le circuit connecté, à chaque fois que l'on appuie sur « entrée » une « * » apparait. On peut tester différents débits (bauds) et pour écrire la valeur :

```
dW4,3
```

La datasheet du circuit donne le « d » (l'adresse) et le « 3 » (le débit, 3 = 9600 bauds).

Sinon il y a une simulation propre au BV4111 et c'est ce que j'ai utilisé :

```
wget http://pin1.org/download/py/bv4111/sv3clV2_2.py

wget http://pin1.org/download/py/bv4111/BV4111.py

wget http://pin1.org/download/py/bv4111/bv4111_demo.py
```

Exécuter bv4111_demo.py après avoir connecté la carte à relais au circuit :

```
python bv4111_demo.py "/dev/ttyAMA0"
```

Le programme va commuter le relai 1, puis le relai 2.

8.6 Installation de l'agent WEB

Il suffit d'installer « pi-web-agent » par le « pistore » depuis le Raspberry Pi et de le lancer :

```
sudo /etc/init.d/pi-web-agent start
```

L'agent est accessible par un navigateur, login=admin, password=admin, il est nécessaire d'ajouter l'agent comme exception aux filtres de sécurité car il n'a pas encore de certificat valide, il s'agit d'une bêta :

https://raspberrypi:8003

En navigant dans l'interface on peut valider les services activés ou la planification des tâches, etc.

Pour lancer le service au démarrage, ajouter la ligne suivante au fichier « /etc/rc.local » :

```
sudo service pi-web-agent start
```

8.7 Installation du serveur WEB avec PHP

Apache 2 est déjà installé avec pi-web-agent, mais je lance quand même les commandes pour m'assurer que tous les paquets soient là :

```
sudo apt-get update

sudo apt-get install apache2 php5 libapache2-mod-php5
```

Effectivement, des paquets sont ajoutés. Exécuter les commandes suivantes en cas d'erreurs :

```
sudo groupadd www-data

sudo usermod -g www-data www-data
```

J'obtiens des commentaires indiquant que le groupe existe déjà et que rien n'est exécuté, c'est sûrement dû à l'installation du pi-web-agent qui a déjà fait une partie du travail.

Il peut être utile pour l'API d'activer le module « Rewrite » d'Apache qui permet de rediriger le trafic vers une autre page :

```
sudo a2enmod rewrite
```

Pour forcer les pages à consulter les .htaccess il faut changer « AllowOverride None » en « AllowOverride ALL » dans :

```
sudo nano /etc/apache2/sites-enabled/000-default
```

Afin que le serveur WEB ait la possibilité d'exécuter des commandes « sudo », ce qui est obligatoire pour lancer des commandes Linux depuis une page WEB, il faut saisir :

```
sudo visudo
```

Et rajouter cette ligne à la fin (on remarque que l'utilisateur « pi » est enregistré ici) :

```
www-data ALL=(ALL) NOPASSWD: ALL
```

Terminer avec le redémarrage d'Apache :

```
sudo service apache2 restart
```

Lancement au démarrage :

```
sudo update-rc.d apache2 enable
```

8.8 Changement du Hostname

Le Hostname est le nom de la machine, par défaut c'est « raspberrypi » et c'est le cas avec n'importe quel autre Raspberry Pi, en tout cas sous Raspbian. Ce peut être une bonne idée de changer ne serait-ce que pour différentier le présent Raspberry Pi d'un autre.

Pour commencer éditer le fichier :

```
sudo nano /etc/hosts
```

Repérer la ligne « 127.0.0.1 raspberrypi » indiquant que l'hôte local s'appelle « raspberrypi ». Ne changer que « raspberrypi », dans mon cas je le renomme en « domotic01 ».

Ensuite éditer le fichier :

```
sudo nano /etc/hostname
```

Ce fichier ne contient que le hostname qu'il suffit de modifier,il doit évidemment être identique à celui entré dans le fichier « /etc/hosts ».

Pour finir il suffit de confirmer les changements et redémarrer le système :

```
sudo /etc/init.d/hostname.sh

sudo reboot
```

8.9 Changement du mot de passe

Par défaut un utilisateur « pi » est créé et son mot de passe est « raspberry » pour le changer il y a deux méthodes : comme un utilisateur normal et comme un utilisateur privilégié.

Pour changer son mot de passe en utilisateur normal :

```
passwd
```

Cette méthode est à utiliser une fois logué et elle demande l'ancien mot de passe.

Pour changer son mot de passe en utilisateur privilégié :

```
sudo passwd pi
```

Puisque l'utilisateur pi est un utilisateur privilégié il peut changer les mots de passe des autres utilisateurs, il suffit de remplacer « pi » par le nom d'utilisateur. Cette méthode ne demande pas l'ancien mot de passe ce qui peut être pratique en cas d'oubli. Dans le cas de l'utilisateur « pi » il vaut mieux avoir choisi le login automatique avant d'oublier le mot de passe[14].

8.10 Installation du point d'accès sans fil

En lisant un article sur Adafruit j'apprends que seulement certains adaptateurs Wi-Fi peuvent fonctionner en point d'accès et que j'avais négligé ceci mais dans la description du produit le vendeur indiquait qu'il peut fonctionner en infrastructure. Après quelques recherches je trouve un article confirmant que les adaptateurs Wi-Fi basés sur le chipset Ralink RT5370 supportent le mode point d'accès.

Tout d'abord installer Hostapd et le DHCP :

```
sudo apt-get update

sudo apt-get install hostapd udhcpd
```

Ensuite configurer le DHCP :

```
sudo nano /etc/udhcpd.conf
```

Modifier les lignes suivantes :

```
start    192.168.42.2

end      192.168.42.20

interface        wlan0

remaining        yes

opt      dns     8.8.8.8 8.8.4.4

opt      subnet  255.255.255.0

opt      router  192.168.42.1

opt      lease   864000
```

Puis éditer :

[14] Le login automatique ne concerne que l'accès physique au Raspberry Pi, utilisé avec clavier, souris et connecté à la télévision. Les accès distants comme avec SSH sont toujours protégés.

```
sudo nano /etc/default/udhcpd
```

Mettre en commentaire la ligne suivante :

```
#DHCPD_ENABLED="no"
```

Attribuer une adresse IP statique à l'interface wlan0 :

```
sudo ifconfig wlan0 192.168.42.1
```

Editer le fichier :

```
sudo nano /etc/network/interfaces
```

Ajouter les lignes suivantes à la fin du fichier pour que l'adresse IP statique soit permanente :

```
iface wlan0 inet static
  address 192.168.42.1
  netmask 255.255.255.0
  gateway 192.168.1.1
```

Mettre les lignes suivantes en commentaires :

```
#wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
#iface default inet manual
```

Maintenant pour configurer Hostapd, créer le fichier :

```
sudo nano /etc/hostapd/hostapd.conf
```

Ajouter ce contenu pour un point d'accès sécurisé :

```
interface=wlan0

driver=nl80211

ssid=raspi-domotic01

hw_mode=g

channel=1

macaddr_acl=0

auth_algs=1

ignore_broadcast_ssid=0

wpa=1

wpa_passphrase=Unlock4Me

wpa_key_mgmt=WPA-PSK

wpa_pairwise=TKIP CCMP

rsn_pairwise=CCMP
```

Ou, pour un point d'accès non sécurisé :

```
interface=wlan0

driver=nl80211

ssid=raspi-config

channel=1
```

Poursuivre avec l'édition de :

```
sudo nano /etc/default/hostapd
```

Activer et modifier la ligne :

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Activer le NAT :

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

Pour que ce soit lancé au démarrage, éditer :

```
sudo nano /etc/sysctl.conf
```

Et ajouter à la fin :

```
net.ipv4.ip_forward=1
```

Ensuite activer le NAT dans le kernel :

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

sudo iptables -A forward -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT

sudo iptables -A forward -i wlan0 -o eth0 -j ACCEPT
```

Pour sauvegarder ces commandes :

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Et éditer :

```
sudo nano /etc/network/interfaces
```

Pour ajouter à la fin :

```
up iptables-restore < /etc/iptables.ipv4.nat
```

Finalement, lancer les services :

```
sudo service hostapd start

sudo service udhcpd start
```

Lancer au démarrage :

```
sudo update-rc.d hostapd enable
```

Retirer le fichier suivant s'il existe :

```
sudo mv /usr/share/dbus-1/system-services/fi.epitest.hostap.WPASupplicant.service ~/
```

Je n’ai pas besoin de service NAT puisque le point d’accès n’est pas censé servir à connecter tous les utilisateurs à Internet, c’est un petit composant de faible puissance mais suffisant pour connecter quelques utilisateurs à quelques mètres. Je supprimerai la fonction NAT plus tard lorsque la maquette fonctionnera correctement.

8.11 Installation de SQLite3

Il suffit d'installer les paquets suivants :

```
sudo apt-get install sqlite3 libsqlite3-0 lisqlite3-dev php5-sqlite
```

8.12 Installation de Slim

Télécharger le fichier .zip à l'adresse :

https://github.com/codeguy/Slim/zipball/master

Décompresser l'archive et placer le contenu du dossier « codeguy-Slim-XYZ » dans la racine du projet (/var/www).

J’ai préféré récupérer le .ZIP avec la commande « wget », décompresser avec « unzip » et placer le contenu du dossier concerner au bon endroit avec « mv ».

9. Construction de la maquette

Voici le détail de la construction de la maquette avec des images et des plans. Les photos ont été prises par mes soins et j'ai dessiné les plans en utilisant la version étudiant d'un vieux programme sur lequel j'ai été formé dans une précédente école. Il s'agit de Me10 8.50, ce programme a été intégré dans une suite et n'existe plus tel quel. Au départ il est conçu pour le dessin de plans cotés pour la mécanique, mais puisqu'il s'agit de dessin vectoriel, je m'en sert pour tout et n'importe quoi. La version étudiant n'est pas limitée mais il est interdit de s'en servir à des fins commerciales.

9.1 Le Raspberry Pi

Je n'ai qu'à placer le circuit avec le câble GPIO branché et une carte SD insérée dans le boîtier[15] et visser. J'ai collé les pieds en caoutchouc pour donner de l'adhérence au tout.

Figure 1 : Le Raspberry Pi assemblé

[15] Ce boîtier contient un couvercle pour éviter de retirer la carte SD

9.2 Le câble prototype

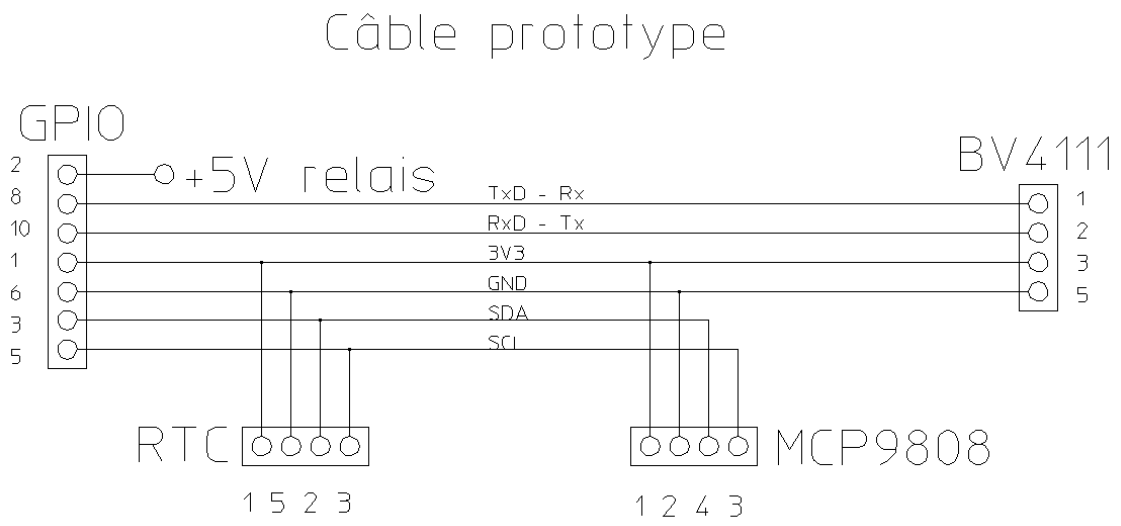
Avant de commencer il est utile de tester le fonctionnement d'une version prototype du montage, un câble sera fabriqué afin de permettre la connexion de tous les composants supportant les fonctionnalités du système. Le câble comportera quatre connecteurs, à savoir :

- GPIO
- RTC
- MCP9808
- BV4111 + alimentation des relais

Ce câble connectera les trois circuits intégrés en même temps afin d'utiliser une version prototype de la solution finale avant la construction de la maquette. La carte à relais est reliée au contrôleur BV4111 par l'intermédiaire d'un câble en nappe fourni à la livraison. Ce câble possède un connecteur séparé pour chaque fil ce qui permet de connecter l'alimentation au connecteur GPIO pour la phase de prototype.

Voici le plan du câble prototype :

Figure 2 : Câble prototype



9.3 Le boîtier de la maquette

Il doit comporter certaines découpes afin d'assembler le cadre de montage des prises Type13, la platine GPIO et le bloc de connexion secteur. Le boîtier est assez épais, dans les 2.5 mm, pour rendre la découpe au cutter presque impossible. J'ai donc utilisé une perceuse pour effectuer les découpes, des petits perçages sont effectués au niveau de tout le contour des découpes, le plastique restant est retiré et la finition est

faite à la lime. Je n'ai pas de matériel de traçage comme un trusquin alors j'utilise une équerre graduée, une règle et essaie de faire au mieux, voici les plans des découpes :

Figure 3 : Découpe du couvercle du boîtier

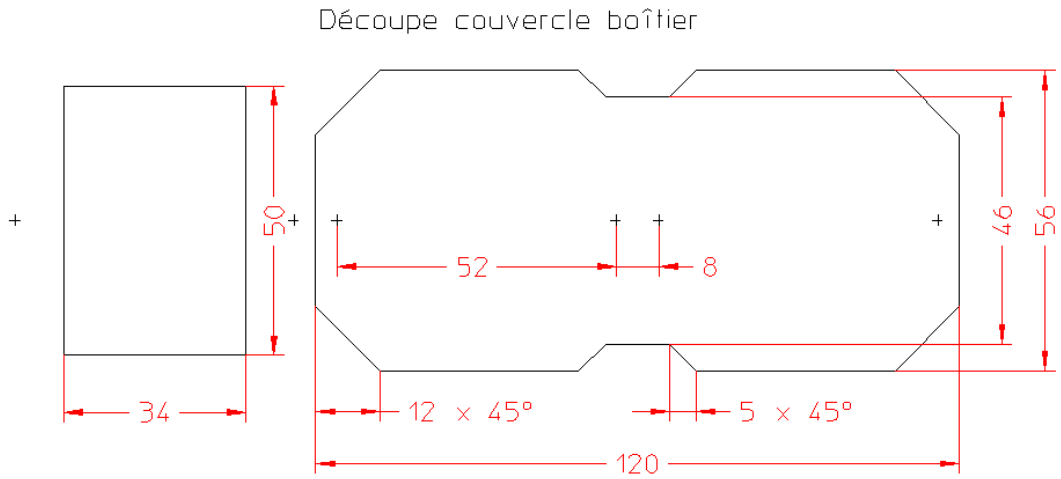


Figure 4 : Découpe de l’arrière du boîtier

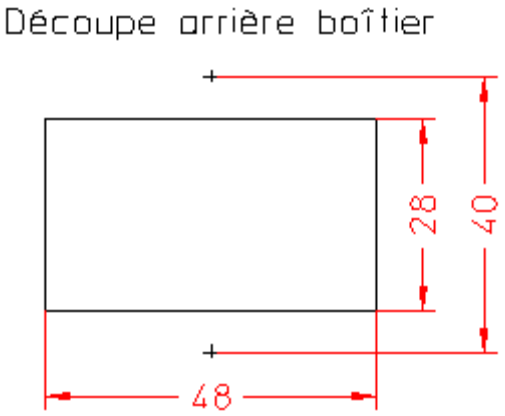
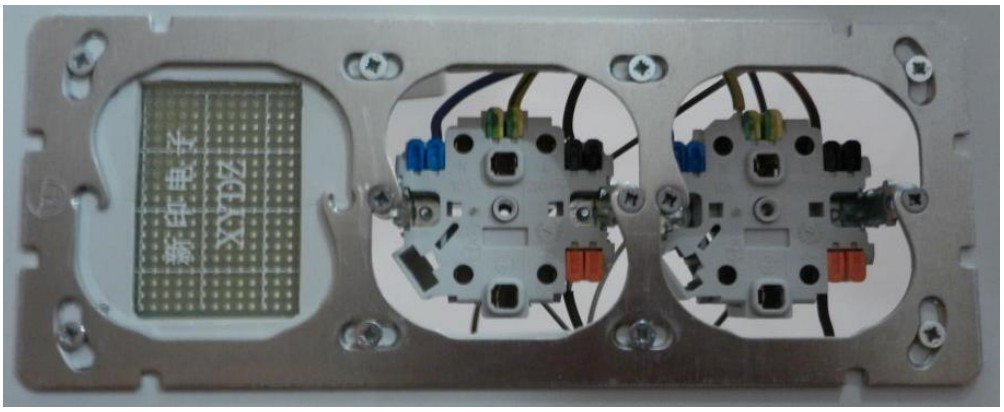


Figure 5 : Prises montées

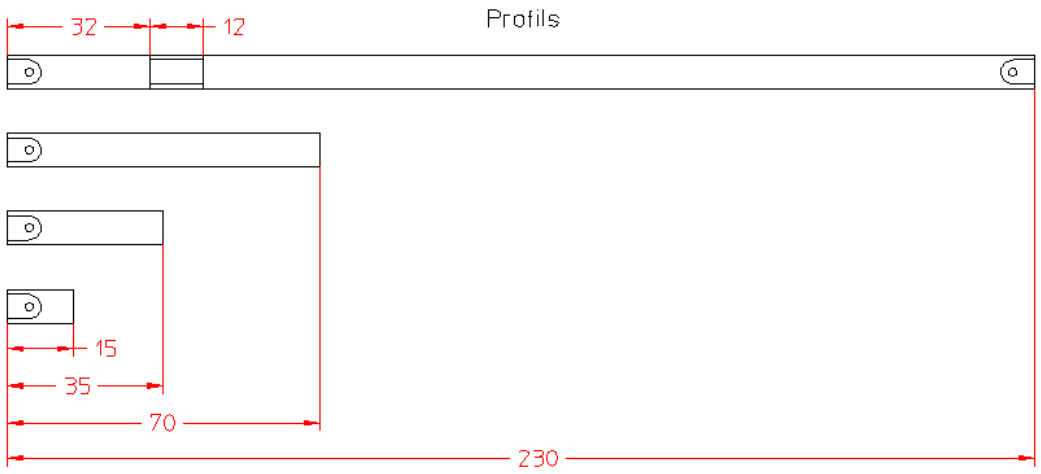


Pour le couvercle, j'ai coté les découpes par rapport aux trous de fixations des prises car ils sont tous visibles sur le cadre de fixation et une finition du boîtier permet de trouver le centre du couvercle.

9.4 Les profils de fixation

Le bas du boîtier comporte des trous et des rainures afin de visser directement des composants à l'intérieur mais comme ils ne correspondent pas aux trous de fixation de mes composants j'ai utilisé un profil en plastique 7.5 x 7.5 mm coupé en plusieurs pièces puis vissés dans les trous du boîtier. Les composants viennent directement se visser dessus, il suffit de pointer l'endroit et d'y placer une vis filière. Un dégagement doit être fait pour les branchements de l'alimentation.

Figure 6 : Profils de fixation



Pour les dégagements des vis, percer à 5 mm du bord un trou de 2 mm, agrandir la partie supérieure à 6.5 mm de diamètre et ébavurer à la pince coupante ou au cutter. Pour le dégagement de l'alimentation, scier le profil sur la moitié de la section (une ligne est présente au milieu) et retirer le reste au cutter. Placer l'alimentation et la carte à relais, pointer les trous et visser directement.

9.5 La platine GPIO

La platine GPIO fonctionne comme le câble prototype. Il s'agit d'un circuit composé de pastille rondes sur lesquelles on soude les composants et pour lesquelles il faut utiliser du fil pour relier les pastilles entre elles.

La face composants comporte le connecteur GPIO, le connecteur du capteur MCP9808 et les LED. La face soudures comporte le connecteur d'alimentation, le connecteur de la RTC, le câble des LED, le câble d'alimentation des relais et le câble

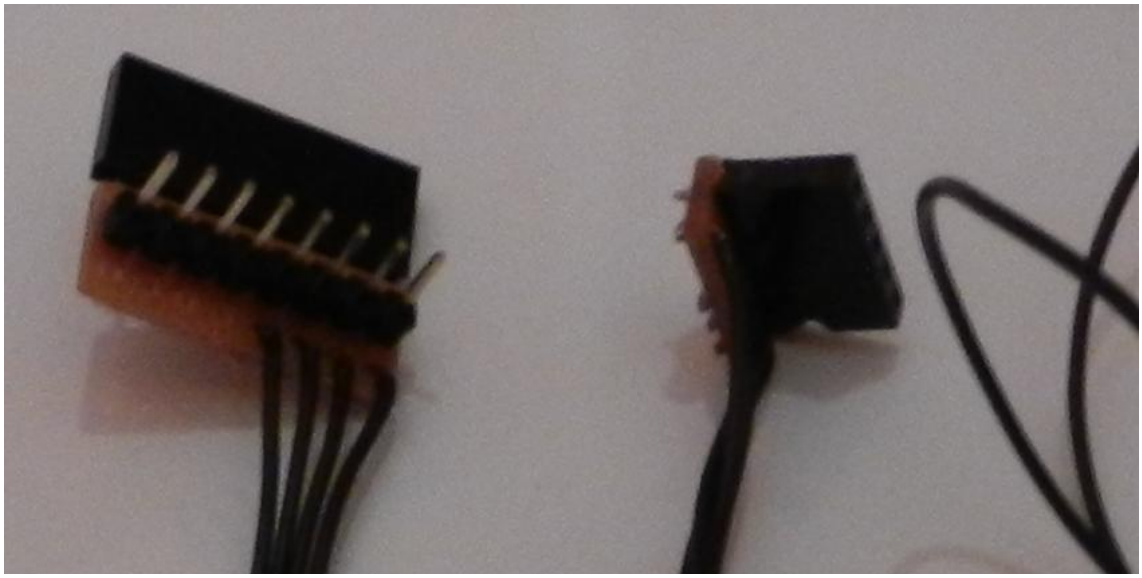
du circuit BV4111. Le connecteur d'alimentation et le connecteur de la RTC sont placés sur un côté de la platine et à plat. De cette façon il n'est possible de brancher l'alimentation et la RTC que dans un seul sens.

Le brochage des composants est le même que celui du câble prototype, les LED en plus.

9.6 Les connecteurs internes

La platine GPIO et les prises Type13 sont montées sous le couvercle du boîtier et les autres composants dans le fond. Afin de rendre d'éventuels ajustements plus simples en cas de problème et de permettre de séparer le couvercle du fond du boîtier, j'ai décidé d'utiliser des connecteurs amovibles. J'ai utilisé les barrettes de connecteurs restantes et les ai raccordées aux câbles à l'aide de petits morceaux de platine universelle cuivrée en ligne restante d'un autre montage, afin d'avoir de meilleures soudures.

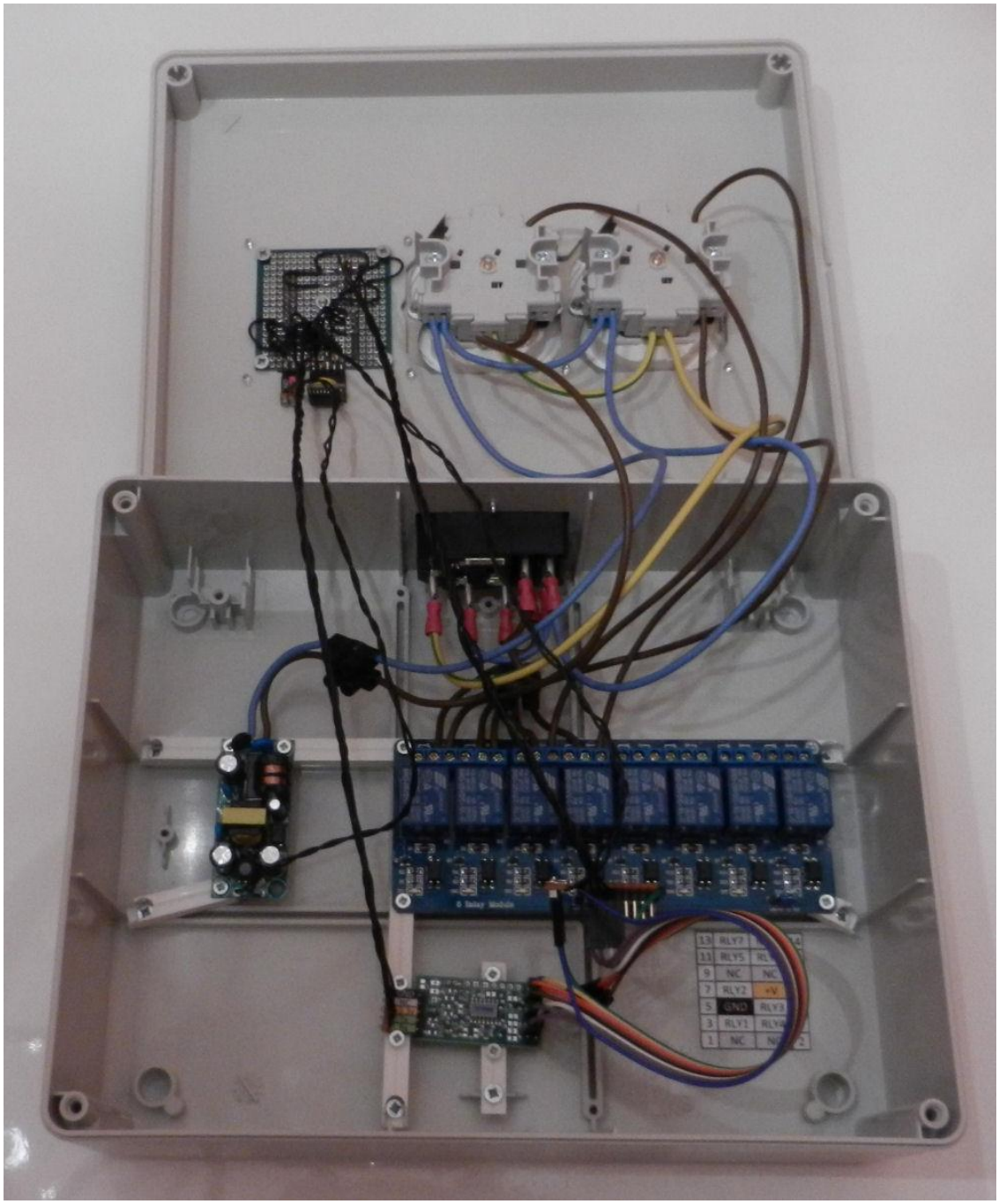
Figure 7 : Le connecteur des LED et le connecteur BV4111



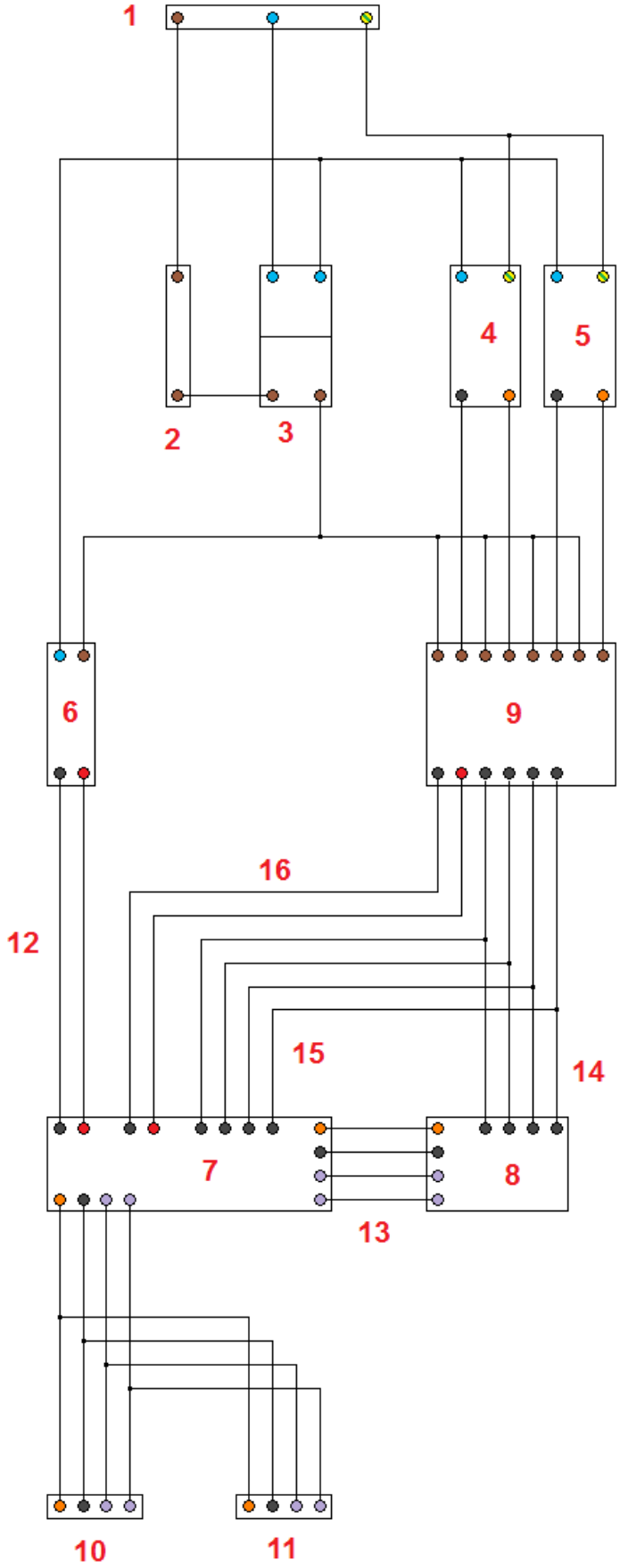
9.7 Le câblage

Voici le câblage de la maquette, la maquette réelle suivie du plan :

Figure 8 : Boîtier ouvert



Effectivement il y a huit relais, le vendeur m'a envoyé celle-ci au même prix car il n'y avait plus de cartes à quatre relais en stock. Le fonctionnement est le même que celui prévu, je ne m'occupe pas des relais supplémentaires.


1
2
3
4
5
6
9
16
12
15
14
7
8
13
10
11

Composant N°	Description
1	Prise secteur
2	Porte-fusible
3	Interrupteur principal
4	Prises 1 et 3
5	Prises 2 et 4
6	Alimentation
7	Platine GPIO
8	BV4111
9	Carte à relais
10	RTC
11	MCP9808
12	Câble d'alimentation 5V
13	Câble série BV4111
14	Câble en de commande des relais
15	Câble des LED
16	Câble d'alimentation des relais

Pour la partie « courant fort », le brochage est indiqué sur les composants.

Le brochage de la carte à relais est marqué sous forme de sérigraphie, au niveau du bornier il suffit de connecter les prises sur le contact ouvert de l'inverseur.

Le brochage de la platine GPIO, du BV4111, de la RTC et du MCP9808 est le même que le câble prototype.

La Maquette m’a couté dans les 300 francs dont voici le résumé :

Tableau 7 : Prix des commandes

Vendeur	**Total**	**Shipping**	**CHF**
The Pi Hut	44.00 £	3.00 £	75.26
jimeer	13.80 £	3.30 £	27.36
eshop4u05	1.85 $	Free	1.84
whymind	1.57 $	Free	1.57
new_it	21.45 £	9.95 £	50.24
inikis	1.99 $	4.1 $	6.06
digitalmeans	6.65 £	3.2 £	15.76
zeenelectronic	4.95 $	Free	4.91
kingfull-electronic-company	4.63 $	Free	4.60
gc_supermarket	1.83 $	Free	1.83
Elektrobedarf Troller	84.80 CHF	Free	84.80
Total			274.23

Il faut encore rajouter quelques pièces de dernière minutes et cela arrondi à ~300 francs.

10. Scripts

Les seuls composants qui requièrent des scripts sont le capteur de température et le contrôleur des relais.

10.1 MCP9808

Pour ce montage j'ai besoin de pouvoir recevoir une seule fois l'information de température lors de l'appel du script python, de cette façon la sortie sera passée à PHP. L'utilisation d'une boucle est exclue.

J'ai placé la fin du code de l'exemple (simpletest.py) en commentaire et ai ajouté l'affichage d'une seule information, sur une seule ligne :

```
# Loop printing measurements every second.

#print 'Press Ctrl-C to quit.'

#while True:

#       temp = sensor.readTempC()

#       print 'Temperature: {0:0.3F}*C / {1:0.3F}*F'.format(temp, c_to_f(temp))

#       time.sleep(1.0)



# Print temperature once.

temp = sensor.readTempC()

print 'T = {0:0.3F} [C]'.format(temp)
```

Sauvegardé sous « simpletest_v2.py »

10.2 BV4111

J'ai besoin des fonctions suivantes pour utiliser les relais :

- allumer un relai
- éteindre un relai
- allumer tous les relais
- éteindre tous les relais
- lire l'état des relais

La démo allume / éteint le relai 1, puis le 2, elle ne comporte que l'appel à la fonction de commande d'un relai, mais importe la librairie « BV4111 ».

Cette dernière inclut la fonction de commande d'un relai « Rly() », la fonction renvoyant le compteur d'un relai « Val() » et la fonction pour éteindre tous les relais « Alloff() ». Il manquait le paramètre « self » dans « Alloff() », je l'ai rajouté et ai ajouté une fonction pour lire l'état des relais (Stat()) en accord avec la datasheet du circuit bv4111 :

```
# --------------------------------------------------------------------$
# all relays off
# --------------------------------------------------------------------$
def Alloff(self):
    self.Send("o\r")
    return self.Wack()


# --------------------------------------------------------------------$
# returns relay status
# --------------------------------------------------------------------$
def Stat(self):
    self.Send("i\r")
    return self.Read()
```

« Self.Send() » envoie la commande au BV4111, commande décrite dans la datasheet (bas de la page 4).

Tableau 8 : Commandes du BV4111

Commande	**Description**
a à h	Allumer / éteindre un relai
r	Lire le compteur d'un relai
o	Éteindre tous les relais
i	Lire l'état de tous les relais

(datasheet du BV4111)

Le « \r » est obligatoire car c'est le retour de chariot.

Sauvegardé sous « BV4111_v2.py »

J'ai donc modifié l'exemple « bv4111_demo.py » pour tester toutes les fonctions dont j'ai besoin :

```
print "Discover() reports ",d
        if Devd.Active() !=0:
            Devd.Alloff()
            sleep(1)
            print "Switching relay 1"
            Devd.Rly(1,1,0)
            sleep(1)
            Devd.Rly(1,0,0)
            sleep(1)
            print "Switching relay 2"
            Devd.Rly(2,1,0)
            sleep(1)
            Devd.Rly(2,0,0)
            sleep(1)
            print "Switching relay 3"
            Devd.Rly(3,1,0)
            sleep(1)
            Devd.Rly(3,0,0)
            sleep(1)
            print "Switching relay 4"
            Devd.Rly(4,1,0)
            sleep(1)
            Devd.Rly(4,0,0)
            sleep(1)
            print "Switching relay 5"
            Devd.Rly(5,1,0)
            sleep(1)
            Devd.Rly(5,0,0)
            sleep(1)
            print "Switching relay 6"
            Devd.Rly(6,1,0)
            sleep(1)
            Devd.Rly(6,0,0)
```

```
sleep(1)
print "Switching relay 7"
Devd.Rly(7,1,0)
sleep(1)
Devd.Rly(7,0,0)
sleep(1)
print "Switching relay 8"
Devd.Rly(8,1,0)
sleep(1)
Devd.Rly(8,0,0)
sleep(1)
print "Relay status: "+str(Devd.Stat())
Devd.Rly(1,1,0)
Devd.Rly(2,1,0)
Devd.Rly(3,1,0)
Devd.Rly(4,1,0)
Devd.Rly(5,1,0)
Devd.Rly(6,1,0)
Devd.Rly(7,1,0)
Devd.Rly(8,1,0)
sleep(1)
print "Relay status: "+str(Devd.Stat())
Devd.Alloff()
```

Chaque relai est allumé pendant une seconde puis éteint, ensuite > affichage de l'état des relais > tous les relais allumés (l'un après l'autre, mais ça à l'air instantané) > affichage de l'état des relais > tous les relais éteints.

Ceci n'est qu'un script de test destiné à confirmer le fonctionnement du système. J'ai fait un script pour trois des cinq fonctions dont j'ai besoin afin de pouvoir lancer une fonction par ligne de commande :

- BV4111_Rlysw.py avec deux paramètres pour allumer / éteindre un relai
- BV4111_Stat.py pour connaître l'état de tous les relais
- BV4111_Alloff.py pour éteindre tous les relais

L'agent WEB ou l'API, connaissant le nombre de relais depuis la base de donnée exécute plus ou moins de fois BV4111_Rlysw.py pour implémenter la fonction « tous allumés ».

L'état des relais est un nombre binaire renvoyé en décimal, le bit de poids fort indique l'état du relai 8, le bit de poids faible indique l'état du relai 1. Lorsque tous les relais sont éteint « 0 » est renvoyé, pour « 00000000 » et lorsque tous les relais sont allumés « 255 » est renvoyé, pour « 11111111 ». Si les quatre premiers relais sont allumés, on aura « 15 », pour « 00001111 ».

10.3 L’application Android

Pour faire cette application, je suis parti d’un travail pratique réalisé dans le cadre du cours de programmation Android (module 535). Ce travail consistait en une application de météo qui interrogeait des serveurs et recevait le résultat sous forme de Json. Ayant terminé ce travail pratique, j’ai décidé d’en réutiliser une partie dans mon application domotique.

11. Tests

J'ai décidé d'utiliser des lettres pour classifier les tests : « M » pour matériel, « F » pour fonctionnement et « L » pour logiciel.

Un test matériel est un test réalisé sur un composant indépendamment du système, si j'ai les moyens techniques de le faire mais qui peut diagnostiquer un problème avant connexion au Raspberry Pi. Généralement, il s'agit des tests effectués à la réception du matériel pour s'assurer du bon fonctionnement.

Un test de fonctionnement, comme son nom l'indique, valide le fonctionnement d'un composant après sa connexion et son installation.

Un test logiciel est effectué sur le code que je modifie ou développe.

11.1 Test de la carte à relais [M1]

La carte à relais reçue est une carte à huit relais car le vendeur m'a indiqué que la carte à quatre relais n'était plus en stock, mais le prix reste le même. Cette carte comporte déjà des LED mais sur le circuit, on ne les verra pas à l'intérieur du boîtier. Les huit relais seront testés même si seulement quatre seront utilisés.

Épreuve : la platine est alimentée avec une tension de 5V, une tension de 5V est appliquée à tour de rôle sur chaque broche IN1 à IN8.

Comportement attendu : lorsque la tension est appliquée à une broche IN, le relais concerné passe en mode de travail (un « clic » peut retentir) et la LED située sur la carte à proximité de ce relais s'allume.

La résistance du contact inverseur de chaque relais sera mesurée pour confirmer son état, une résistance nulle indique un contact fermé alors qu'une résistance infinie indique un contact ouvert.

Tableau 9 : Mesures du test [M1]

5V	Bruit	LED	R	Conclusion
IN1	Clic	S’allume	Max 1.4 Ohm	Relai 1 OK
IN2	Clic	S’allume	Max 1.4 Ohm	Relai 2 OK
IN3	Clic	S’allume	Max 1.4 Ohm	Relai 3 OK
IN4	Clic	S’allume	Max 1.4 Ohm	Relai 4 OK
IN5	Clic	S’allume	Max 1.4 Ohm	Relai 5 OK
IN6	Clic	S’allume	Max 1.4 Ohm	Relai 6 OK
IN7	Clic	S’allume	Max 1.4 Ohm	Relai 7 OK
IN8	Clic	S’allume	Max 1.4 Ohm	Relai 8 OK

Résultat du test : la carte à relais fonctionne.

11.2 Test des LED [M2]

Épreuve : quatre LED sont connectée à tour de rôle à une alimentation de PC fournissant 3.3V (fil orange) et à la masse (fil noir). Une LED comporte un plat sur le côté du boîtier pour signaler où est la cathode qui doit être connectée à la masse, dans ce cas.

Comportement attendu : chaque LED s'illumine en bleu.

Mesure : chaque LED s'illumine en bleu.

Résultat du test : les quatre LED fonctionnent.

11.3 Test de la RTC [F1]

Épreuve : la RTC est connectée au Raspberry Pi par l'intermédiaire du câble prototype, la procédure d'installation est suivie avec succès, le Raspberry Pi est éteint, débranché et rebranché sauf le réseau (pour éviter les serveurs NTP).

Comportement attendu : le Raspberry Pi démarre avec l'heure et la date à jour.

Mesure : la date et l'heure sont à jour.

Résultat du test : le fonctionnement de la RTC correspond à celui attendu.

11.4 Test du capteur de température [F2]

Épreuve : le MCP9808 est connecté au Raspberry Pi par l'intermédiaire du câble prototype, la procédure d'installation est suivie avec succès, le script simpletest.py est exécuté et un thermomètre classique est à proximité pour comparaison.

Comportement attendu : la température de la pièce est affichée dans le terminal.

Mesure : la température est affichée dans le terminal et correpond à celle affichée par un thermomètre classique.

Résultat du test : le fonctionnement du MCP9808 correspond à celui attendu.

11.5 Test du contrôleur BV4111 [F3]

Épreuve : le BV4111 est connecté au Raspberry Pi par l'intermédiaire du câble prototype, la procédure d'installation est suivie avec succès, les huit relais sont raccordés au BV4111 et le script bv4111_demo.py est exécuté.

Comportement attendu : le relai 1 commute cinq fois puis le relai 2 commute cinq fois.

Mesure : le relai 1 a commuté cinq fois puis le relais 2 a commuté cinq fois.

Résultat du test : le fonctionnement du BV4111 correspond à celui attendu.

11.6 Test du câble prototype [F4]

Les résultats positifs aux tests [F1], [F2] et [F3] valident le fonctionnement du câble prototype.

11.7 Test de l'agent WEB [L1]

Je teste couramment le fonctionnement d'un programme lors du développement, ce test est celui effectué au moment où j'ai considéré l'agent WEB comme terminé.

Épreuve : Le Raspberry Pi est connecté à la maquette et au réseau câblé. L'adresse IP du port Ethernet du Raspberry Pi est saisie dans le navigateur d'un autre ordinateur du réseau. La confirmation de l'état d'un relai est indiquée par les LED bleues et mesurées dans les prises avec un tournevis tâteur.

Tableau 10 : Mesures du test [L1]

Bouton	**Comportement attendu**	**Mesure**
/	Affichage de la température	Température affichée
Commuter prise 1	La prise 1 change d'état	La prise 1 a changé d'état
Commuter prise 2	La prise 2 change d'état	La prise 2 a changé d'état
Commuter prise 3	La prise 3 change d'état	La prise 3 a changé d'état
Commuter prise 4	La prise 4 change d'état	La prise 4 a changé d'état
Tous allumés	Les 4 prises s'allument	Les 4 prises sont allumées
Tous éteints	Les 4 prises s'éteignent	Les 4 prises sont éteintes

Résultat du test : le fonctionnement de l'agent WEB correspond à celui attendu.

11.8 Test de l'API [L2]

Ce test est celui effectué au moment où j'ai considéré l'API comme terminée.

Épreuve : Le Raspberry Pi est connecté à la maquette et au réseau câblé.

Les résultats du test [L1] sont positifs.

L'adresse IP du port Ethernet du Raspberry Pi suivie de la route voulue est saisie dans le navigateur d'un autre ordinateur du réseau.

Tableau 11 : Mesures du test [L2]

Route	**Retour**	**Mesure**
/api/api.php/	Working	/
/api/api.php/nbrel	{"nbrel":4}	/
/api/api.php/temp	{"temp":"T = 23.650 C"}	/
/api/api.php/rly/1	{"result":"OK"}	La prise 1 a changé d'état
/api/api.php/rly/2	{"result":"OK"}	La prise 2 a changé d'état
/api/api.php/rly/3	{"result":"OK"}	La prise 3 a changé d'état
/api/api.php/rly/4	{"result":"OK"}	La prise 4 a changé d'état
/api/api.php/allon	{"result":"OK"}	Toutes les prises s'allument
/api/api.php/alloff	{"result":"OK"}	Toutes les prises s'éteignent

Résultat du test : le fonctionnement de l'API correspond à celui attendu.

11.9 Test de l'application Android [L3]

Ce test est celui effectué au moment où j'ai considéré l'application comme terminée.

Épreuve : Le Raspberry Pi est connecté à la maquette et au réseau câblé.

Les résultats des tests [L1] et [L2] sont positifs.

L'application est lancée sur l'émulateur Android depuis un autre ordinateur du réseau.

Tableau 12 : Mesures du test [L3]

Bouton	**Comportement attendu**	**Mesure**
OK	6 boutons s'affichent	6 boutons s'affichent
/	Affichage de la température	Température affichée
Commuter prise 1	La prise 1 change d'état	La prise 1 a changé d'état
Commuter prise 2	La prise 2 change d'état	La prise 2 a changé d'état
Commuter prise 3	La prise 3 change d'état	La prise 3 a changé d'état
Commuter prise 4	La prise 4 change d'état	La prise 4 a changé d'état
Tous allumés	Les 4 prises s'allument	Les 4 prises sont allumées
Tous éteints	Les 4 prises s'éteignent	Les 4 prises sont éteintes

Résultat du test : le fonctionnement de l'application Android correspond à celui attendu.

11.10 Test du point d'accès sans fil [F5]

Épreuve : Le service Hostapd est actif et un dispositif mobile Android est utilisé pour se connecter au SSID concerné (ici raspi-domotic).

Comportement attendu : le dispositif mobile affiche « connecté » ou similaire et la page de l'agent WEB s'affiche dans le navigateur d'Android après saisie de l'adresse IP du point d'accès (http://192.168.42.1)

Mesure : le dispositif mobile affiche « connecté » et la page de l'agent WEB est affichée.

Résultat du test : le fonctionnement du point d'accès correspond à celui attendu.

Note : le résultat est le même avec mes deux dispositifs Android.

11.11Test de la solution en condition réelle [F6]

Épreuve : La maquette est branchée au réseau électrique, le Raspberry Pi est connecté au port GPIO et le câble Ethernet n'est pas branché.

Aucune alimentation externe n'est utilisée, en effet le boîtier connecté au réseau électrique fournit l'alimentation au Raspberry Pi par le câble GPIO.

Une lampe est connectée à chaque prise à tour de rôle et l'application est installée sur un périphérique Android et l'interrupteur de la maquette est placé sur la position « 1 ».

Comportement attendu : le Raspberry Pi démarre (la LED « ACT » signale un accès à la carte SD), le ssid du point d'accès « raspi-domotic » apparaît dans les réseaux disponibles du dispositif Android, ce dernier s'y connecte. L'application est lancée, l'appui sur le bouton « OK » provoque l'affichage de la température et des autres boutons. L'appui sur chaque bouton provoque le comportement souhaité.

Tableau 13 : Mesures du test [F6]

Action	**Mesure**
Connexion à « raspi-domotic »	« Connecté » est affiché
Appui sur le bouton « OK »	La température et les boutons s'affichent
Appui sur « Commuter prise 1 »	La lampe branchée sur la prise 1 s'allume
Appui sur « Commuter prise 2 »	La lampe branchée sur la prise 2 s'allume
Appui sur « Commuter prise 3 »	La lampe branchée sur la prise 3 s'allume
Appui sur « Commuter prise 4 »	La lampe branchée sur la prise 4 s'allume
Appui sur « Tous allumés »	Toutes les prises fournissent du courant
Appui sur « Tous éteints »	Aucune prise ne fournit de courant

Résultat du test : le fonctionnement de la maquette correspond à celui attendu, tous les objectifs sont atteints.

Note : le résultat est le même avec mes deux dispositifs Android.

12. Conclusion

Tous les objectifs ont été atteints, j'ai pu suivre le planning, installer tout le matériel, construire la maquette, développer un agent WEB, une API et une application Android.

Des difficultés ont été rencontrées mais cela m'a permis d'en apprendre plus concernant PHP en général mais surtout Linux. En principe, l'agent WEB et le pi-web-agent ne servent à rien pour piloter les relais depuis Android mais ils ont permis de valider le fonctionnement de la maquette avant que je me lance dans le développement de l'API qui fait partie des éléments nouveaux auxquels je ne m'était pas intéressé avant. En utilisant du temps pour développer l'agent WEB, j'en ai économisé sur le dépannage et cela a rendu la maquette compatible avec du matériel non-Android.

Je m'intéressais déjà à l'électronique mais n'avais pas encore touché à un mini-ordinateur comme le Raspberry Pi. Lors de l'avancement du projet, on remarque rapidement les avantages de Linux pour ce genre d'utilisation. Je pense que je réutiliserais des Raspberry Pi dans d'autres projets.

Bibliographie

Solutions domotiques :

CSDomotic, 2014. Site officiel [en ligne]. [Consulté le 21.10.2014]. Disponible à l'adresse : http://csdomotic.ch/

Hager SAS, 2014. Site officiel [en ligne]. [Consulté le 21.10.2014]. Disponible à l'adresse : http://www.hager.fr/particuliers/solutions-hager/domotique-maison/563.htm

Zodianet, 2014. Site officiel [en ligne]. [Consulté le 21.10.2014]. Disponible à l'adresse : http://www.zodianet.com/

HoMIDoM, 2011. Site officiel [en ligne]. [Consulté le 21.10.2014]. Disponible à l'adresse : http://www.homidom.com/

Laurent, 2012. Les applications domotiques pour Android [en ligne]. 28 août 2012. [Consulté le 21.10.2014]. Disponible à l'adresse : http://blog.domadoo.fr/2012/08/28/les-applications-domotiques-pour-android/

Cartes à relais :

EBay, 2014. Article 251547347030 [en ligne]. [Consulté le 23.10.2014]. Disponible à l'adresse : http://www.ebay.com/itm/5V-4-Channel-Relay-Module-for-Arduino-PIC-ARM-DSP-AVR-Raspberry-Pi-/251547347030?pt=LH_DefaultDomain_0&hash=item3a9163e856

EBay, 2014. Article 301320112765 [en ligne]. [Consulté le 23.10.2014]. Disponible à l'adresse : http://www.ebay.com/itm/JBtek-4-Channel-DC-5V-Relay-Module-for-Arduino-Raspberry-Pi-DSP-AVR-PIC-ARM-/301320112765?pt=LH_DefaultDomain_0&hash=item4628140a7d

EBay, 2014. Article 180870781816 [en ligne]. [Consulté le 23.10.2014]. Disponible à l'adresse : http://www.ebay.com/itm/IACS-8-Channel-Relay-Board-5V-9V-12V-24V-DC-8CH-TTL-10A-Raspberry-Pi-Arduino-AVR-/180870781816?pt=UK_BOI_Electrical_Components_Supplies_ET&var=&hash=item2a1cbd1b78

EBay, 2014. Article 160927333012 [en ligne]. [Consulté le 23.10.2014]. Disponible à l'adresse : http://www.ebay.com/itm/SPI-breakout-board-with-6-relays-for-AVR-arduino-PIC-Raspberry-Pi-/160927333012?pt=LH_DefaultDomain_0&hash=item2578043a94

EBay, 2014. Article 261619547472 [en ligne]. [Consulté le 23.10.2014]. Disponible à l'adresse : http://www.ebay.com/itm/Raspberry-Pi-Digital-and-Relay-I-O-Expander-DIY-

kit-PCB-component-sample-code-/261619547472?pt=LH_DefaultDomain_0&hash=item3ce9bd7d50

EBay, 2014. Article 310657229076 [en ligne]. [Consulté le 23.10.2014]. Disponible à l'adresse : http://www.ebay.com/itm/Serial-Relay-with-Timer-4-way-10A-for-PC-Microcontroller-Raspberry-Pi-/310657229076?pt=UK_BOI_Electrical_Components_Supplies_ET&hash=item48549d2114

EBay, 2014. Article 111175881635 [en ligne]. [Consulté le 23.10.2014]. Disponible à l'adresse : http://www.ebay.com/itm/Serial-I2C-Twin-Stackable-Relay-with-ADC-for-the-Raspberry-Pi-Arduino-/111175881635?pt=UK_BOI_Electrical_Components_Supplies_ET&hash=item19e29953a3

EBay, 2014. Article 110989499360 [en ligne]. [Consulté le 23.10.2014]. Disponible à l'adresse : http://www.ebay.com/itm/Serial-Relay-Controller-with-Timer-for-PC-Microcontroller-Raspberry-Pi-/110989499360?pt=UK_BOI_Electrical_Components_Supplies_ET&hash=item19d77d5be0

EBay, 2014. Article 201184644307 [en ligne]. [Consulté le 23.10.2014]. Disponible à l'adresse : http://www.ebay.com/itm/PIFACE-Digital-2-fully-assembled-for-the-RASPBERRY-Pi-model-B-B-plus-/201184644307?pt=UK_Computing_Other_Computing_Networking&hash=item2ed78a0cd3

ByVac, 2013. Pi Chips [en ligne]. [Consulté le 23.10.2014]. Disponible à l'adresse : http://www.pichips.co.uk/index.php/Twin_Relay

ByVac, 2014. BV4111 [en ligne]. [Consulté le 23.10.2014]. Disponible à l'adresse : http://www.byvac.com/index.php/BV4111

Protocoles :

Wikipedia, 2014. Z-Wave [en ligne]. [Consulté le 27.10.2014]. Disponible à l'adresse : http://fr.wikipedia.org/wiki/Z-Wave

Wikipedia, 2014. 6LoWPAN [en ligne]. [Consulté le 27.10.2014]. Disponible à l'adresse : http://fr.wikipedia.org/wiki/6LoWPAN

Wikipedia, 2014. Enocean [en ligne]. [Consulté le 27.10.2014]. Disponible à l'adresse : http://fr.wikipedia.org/wiki/Enocean

Wikipedia, 2014. ZigBee [en ligne]. [Consulté le 27.10.2014]. Disponible à l'adresse : http://fr.wikipedia.org/wiki/ZigBee

TODESCHINI, Vincent, 2014. Le protocole RFY [en ligne]. 22 mai 2014. [Consulté le 27.10.2014]. Disponible à l'adresse : https://plus.google.com/115226308120596292694/posts/Kz2b8JMu4ML

Pascal, 2014. Le protocole X2D / X3D, comment ça marche? Pour qui? [en ligne]. [Consulté le 27.10.2014]. Disponible à l'adresse : http://domotique.comprendrechoisir.com/astuce/voir/155607/le-protocole-x2d-x3d-comment-ca-marche-pour-qui

Swiss-Domotique, 2014. Présentation du protocole HomeEasy (Chacon) [en ligne]. [Consulté le 27.10.2014]. Disponible à l'adresse : http://www.swiss-domotique.ch/index.php/Guides/Domotique/Les-technologies/HomeEasy-Chacon/presentationhomeeasy.html

Wikipedia, 2014. X10 (informatique) [en ligne]. [Consulté le 27.10.2014]. Disponible à l'adresse : http://fr.wikipedia.org/wiki/X10_(informatique)

Visonic, 2014. Site officiel [en ligne]. [Consulté le 27.10.2014]. Disponible à l'adresse : http://www.visonic.com/france/fr/Visonic-France/index.html?setRegion=true

Olivier, 2012. Décodage des protocoles Oregon Scientific sur Arduino (1/3) [en ligne]. 6 septembre 2012. [Consulté le 27.10.2014]. Disponible à l'adresse : http://connectingstuff.net/blog/decodage-protocole-oregon-arduino-1/

Skywodd, 2012. Interrupteurs domotique Blyss de castorama [en ligne]. 18 juin 2012. [Consulté le 27.10.2014]. Disponible à l'adresse : http://forum.arduino.cc/index.php?topic=109892.0

Wikipedia, 2014. Hypertext Transfer Protocol [en ligne]. [Consulté le 27.10.2014]. Disponible à l'adresse : http://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Comment ça marche, 2014. Le protocole HTTP [en ligne]. Novembre 2014. [Consulté le 27.10.2014]. Disponible à l'adresse : http://www.commentcamarche.net/contents/520-le-protocole-http

PIDEIL, Claude, 2001. Le protocole HTTP [en ligne]. 3 décembre 2001. [Consulté le 27.10.2014]. Disponible à l'adresse : http://www.lirmm.fr/~ajm/Cours/01-02/DESS_TNI/TER13/dochttp.htm

Zodianet, 2014. Comparaison des protocoles domotiques [en ligne]. [Consulté le 27.10.2014]. Disponible à l'adresse : http://www.zodianet.com/la-toolbox/protocols-comparison.html

Domotics, 2013. Par où commencer ? Etape 1 : Quel protocole choisir ? [en ligne]. 15 octobre 2013. [Consulté le 27.10.2014]. Disponible à l'adresse : http://www.touteladomotique.com/index.php?option=com_content&view=article&id=1030:par-ou-commencer-nd1-quel-protocole-choisir-&catid=5:domotique&Itemid=89#.VIp4wdKG8U1

Installation :

MONK, Simon, 2014. Configuring I2C [en ligne]. [Consulté le 13.11.2014]. Disponible à l'adresse : https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c

Lady Ada, 2014. Adding a Real Time Clock to Raspberry Pi [en ligne]. [Consulté le 13.11.2014]. Disponible à l'adresse : https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi?view=all

DICOLA Tony, 2014. MCP9808 Temperature Sensor Python Library [en ligne]. [Consulté le 13.11.2014]. Disponible à l'adresse : https://learn.adafruit.com/mcp9808-temperature-sensor-python-library/software

ByVac, 2014. RPI Serial [en ligne]. 23 septembre 2013 [Consulté le 13.11.2014]. Disponible à l'adresse : http://doc.byvac.com/index.php5?title=RPI_Serial

ByVac, 2014. RPi Serial Troubleshooting Guide [en ligne]. [Consulté le 13.11.2014]. Disponible à l'adresse : http://www.byvac.com/index.php/RPi_Serial_Troubleshooting_Guide

Vaslabs, 2013. Pi-web-agent [en ligne]. [Consulté le 13.11.2014]. Disponible à l'adresse : https://github.com/vaslabs/pi-web-agent

Drcurzon, 2014. Raspberry Pi Web Server [en ligne]. [Consulté le 13.11.2014]. Disponible à l'adresse : http://www.instructables.com/id/Raspberry-Pi-Web-Server/step7/Install-Apache-with-PHP/

IBEX. PHP/Apache [en ligne]. [Consulté le 13.11.2014]. Disponible à l'adresse : http://www.raspberry-projects.com/pi/software_utilities/phpapache

How-to Geek, 2014. How to Change Your Raspberry Pi (or Other Linux Device's) Hostname [en ligne]. [Consulté le 13.11.2014]. Disponible à l'adresse : http://www.howtogeek.com/167195/how-to-change-your-raspberry-pi-or-other-linux-devices-hostname/

Paulmckinnie, 2012. Reset OS password [en ligne]. 19 octobre 2012 [Consulté le 13.11.2014]. Disponible à l'adresse : http://www.raspberrypi.org/forums/viewtopic.php?f=47&t=20397

Lady Ada, 2014. Setting up a Raspberry Pi as a WiFi access point [en ligne]. [Consulté le 13.11.2014]. Disponible à l'adresse : https://learn.adafruit.com/setting-up-a-raspberry-pi-as-a-wifi-access-point/install-software

Elinux.org, 2014. RPI-Wireless-Hotspot [en ligne]. 5 août 2014 [Consulté le 13.11.2014]. Disponible à l'adresse : http://elinux.org/RPI-Wireless-Hotspot

The Rantings and Ravings of a Madman, 2011. How To : Use The Raspberry Pi As A Wireless Access Point/Router Part 1 [en ligne]. [Consulté le 13.11.2014]. Disponible à l'adresse : http://sirlagz.net/2012/08/09/how-to-use-the-raspberry-pi-as-a-wireless-access-pointrouter-part-1/

Poing, 2012. [Tutorial] Using the RasPi as a WiFi hostspot (8 Nov 12) [en ligne]. [Consulté le 13.11.2014]. Disponible à l'adresse : http://www.raspberrypi.org/forums/viewtopic.php?f=36&t=19120&start=100

SMITH, Andy, 2013. Raspberry Pi Wi-Fi Honeypot [en ligne]. 19 août 2013 [Consulté le 13.11.2014]. Disponible à l'adresse : http://andrewmichaelsmith.com/2013/08/raspberry-pi-wi-fi-honeypot/

SIMS, Gary, 2014. How to Set up a Raspberry Pi as a Wireless Access Point [en ligne]. 3 février 2014 [Consulté le 13.11.2014]. Disponible à l'adresse : http://www.maketecheasier.com/set-up-raspberry-pi-as-wireless-access-point/

KRÖMER, Guido, 2013. RESTful API with Slim PHP and SQLite [en ligne]. 28 avril 2013 [Consulté le 27.11.2014]. Disponible à l'adresse : http://cacodaemon.de/index.php?id=48

LOCKHART, Josh, 2012. SLIM Framework Documentation [en ligne]. [Consulté le 27.11.2014]. Disponible à l'adresse : http://docs.slimframework.com/

Chris C, 2013. How to manually install the Slim Framework for PHP [en ligne]. 10 octobre 2013 [Consulté le 27.11.2014]. Disponible à l'adresse : https://www.a2hosting.com/kb/installable-applications/manual-installations/slim-framework-for-php

Développement :

Stack Overflow, 2012. Putting a simple if-then statement on one line [en ligne]. 12 octobre 2012 [Consulté le 24.11.2014]. Disponible à l'adresse : http://stackoverflow.com/questions/2802726/putting-a-simple-if-then-statement-on-one-line

Tutorialspoint, 2014. Python Basic Operators [en ligne]. [Consulté le 24.11.2014]. Disponible à l'adresse : http://www.tutorialspoint.com/python/python_basic_operators.htm

MEUS, Frank, 2013. Running it all on the Raspberry Pi (Part 1) [en ligne]. 22 juillet 2013 [Consulté le 27.11.2014]. Disponible à l'adresse : http://sharedmemorydump.net/post/2013-07-22-running-it-all-on-the-raspberry-pi-part-1

Phalcon, 2014. Micro Applications [en ligne]. [Consulté le 27.11.2014]. Disponible à l'adresse : http://docs.phalconphp.com/fr/latest/reference/micro.html

PHP, 2014. passthru [en ligne]. [Consulté le 25.11.2014]. Disponible à l'adresse : http://php.net/manual/fr/function.passthru.php

GAIC, Dragan, 2014. BEST AVAILABLE PHP RESTFUL MICRO FRAMEWORKS [en ligne]. 7 février 2014 [Consulté le 27.11.2014]. Disponible à l'adresse : http://www.gajotres.net/best-available-php-restful-micro-frameworks/

Jquery Foundation, 2014. Downloading jQuery [en ligne]. [Consulté le 25.11.2014]. Disponible à l'adresse : http://jquery.com/download/

RILEY, Adam, 2012. Getting started with Databases on the Pi with SQLite [en ligne]. 26 novembre 2012 [Consulté le 25.11.2014]. Disponible à l'adresse : http://www.raspberrypiblog.com/2012/11/getting-started-with-databases-on-pi.html

Stack Overflow, 2013. How to Call a PHP Function on the Click of a Button [en ligne]. 23 décembre 2013 [Consulté le 25.11.2014]. Disponible à l'adresse : http://stackoverflow.com/questions/20738329/how-to-call-a-php-function-on-the-click-of-a-button

THEOBALD, Lewis, 2014. Creating a RESTful API with PHP using Slim [en ligne]. 3 avril 2014 [Consulté le 2711.2014]. Disponible à l'adresse : http://www.aljtmedia.com/blog/creating-a-restful-api-with-php-using-slim/

YAGUDAEV, Michael,, 2010. Resolving PHP Relative Path Problem [en ligne]. 18 mars 2010 [Consulté le 25.11.2014]. Disponible à l'adresse : http://yagudaev.com/posts/resolving-php-relative-path-problem/

PHP, 2014. SQLite3::query [en ligne]. [Consulté le 25.11.2014]. Disponible à l'adresse : http://php.net/manual/fr/sqlite3.query.php

PHP, 2014. Système d'exécution de programme [en ligne]. [Consulté le 25.11.2014]. Disponible à l'adresse : http://php.net/manual/fr/book.exec.php

Adafruit. DICOLA, Tony, 2014. Script « simpletest.py ». Disponible à l'adresse : https://learn.adafruit.com/mcp9808-temperature-sensor-python-library/software

ByVac, 2014. Script « sv3clV2.py ». Disponible à l'adresse : http://www.byvac.com/index.php/BV4111

ByVac, 2014. Script « BV4111.py ». Disponible à l'adresse : http://www.byvac.com/index.php/BV4111

ByVac, 2014. Script « bv4111_demo.py ». Disponible à l'adresse : http://www.byvac.com/index.php/BV4111

DAEHNE, Peter, 2013 - 2014. Travaux pratique série 10. Disponible à l'adresse : http://campus.hesge.ch/Daehne/2013-2014/Module635.1/Prog/Module635.1.htm