

Christian Tavernier

Raspberry Pi

Prise en main
et premières réalisations

DUNOD

Du même auteur



Couverture : Rachid Maraï

Le pictogramme qui figure ci-contre mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du photocopillage. Le Code de la propriété intellectuelle du 1^{er} juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements		d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée. Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation de l'auteur, de son éditeur ou du Centre français d'exploitation du droit de copie (CFC, 20, rue des Grands-Augustins, 75006 Paris).
--	---	--

© Dunod, Paris, 2013

ISBN 978-2-10-60127-1

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2° et 3° a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

TABLE DES MATIÈRES

Avant-propos	IX
1 • À la découverte du Raspberry Pi	1
1.1 Raspberry Pi A ou B ?	1
1.2 Petite visite guidée	3
1.3 L'alimentation	5
1.3.1 Le chargeur pour smartphone	5
1.3.2 Le bloc secteur « prise de courant »	5
1.3.3 Le hub USB actif ou hub alimenté	6
1.4 Le clavier et la souris	7
1.5 L'écran	8
1.5.1 HDMI, DVI ou même VGA	8
1.5.2 La sortie vidéo composite via la prise Cinch ou RCA	9
1.6 Le réseau	10
1.7 Préparation de la carte SD	10
1.7.1 Téléchargement et vérification de la distribution Linux	11
1.7.2 Préparation de la carte SD	13
1.8 Mémoires de masse externes	15
2 • À la découverte de Raspbian	17
2.1 Première mise sous tension	17
2.1.1 Sélection du clavier Français	19
2.1.2 Localisation de votre distribution	21
2.2 LXDE, le bureau de Raspbian	23
2.2.1 Le menu Accessoires	24
2.2.2 Le menu Éducation	25
2.2.3 Le menu Internet	26
2.2.4 Le menu Programmation	27
2.2.5 Le menu Graphismes	27
2.2.6 Le menu Outils système	28

2.2.7	Le menu Préférences	28
2.3	Le mode ligne de commande	29
2.3.1	Accès au mode ligne de commande	29
2.3.2	Super-utilisateur, root et sudo	30
2.3.3	Utilisateurs et groupes	31
2.3.4	Organisation du système de fichiers	32
2.3.5	Navigation dans les fichiers et les répertoires	34
2.3.6	Droits associés aux fichiers	35
2.3.7	Une aide toujours présente	37
2.4	Ajout et suppression de programmes	38
2.4.1	À la recherche des logiciels	39
2.4.2	Installation d'un programme	40
2.4.3	Désinstallation d'un programme	41
2.4.4	Mise à jour d'un programme	42
3 •	Réseau, WiFi, partitions et paramétrages	43
3.1	Réseau câblé sans DHCP	43
3.1.1	Accès au réseau local	44
3.1.2	Accès à Internet	45
3.2	Le Raspberry Pi en WiFi	46
3.2.1	Configuration manuelle de la clé Wi-Pi	47
3.2.2	Configuration assistée de la clé Wi-Pi	49
3.2.3	Configuration des « autres » clés	51
3.3	Les partitions de la carte SD	54
3.3.1	Extension automatique de la partition principale	55
3.3.2	Extension manuelle de la partition principale	56
3.4	Sauvegardez votre carte SD	60
3.5	Le fichier config.txt	61
3.5.1	Modifier les paramètres d'affichage	62
3.5.2	Comment « overclocker » le Raspberry Pi ?	64
3.5.3	Comment survolter le Raspberry Pi ?	65
3.5.4	Augmenter les performances sans risque (ou presque)	66
3.5.5	Si le Raspberry Pi devient instable	67
4 •	Bureautique, media center et serveur web	69
4.1	Les différentes solutions et programmes bureautiques	69
4.1.1	Utilisation de programmes indépendants	70
4.1.2	Utilisation d'une suite bureautique	75
4.1.3	Une fois que le choix est fait	78
4.2	Partage de fichiers via le réseau	78

4.3	Un serveur avec votre Raspberry Pi	80
4.3.1	Installation des logiciels nécessaires	81
4.3.2	Utilisation du serveur	84
4.4	Le Raspberry Pi en tant que media center	84
4.4.1	De la musique en travaillant	85
4.4.2	Lorsque le son est récalcitrant	86
4.4.3	Un véritable media center (ou HTPC)	87
4.4.4	Quelques conseils pour votre <i>media center</i>	90
5 •	À la découverte de Python	93
5.1	L'environnement de travail	93
5.1.1	Interpréteur et compilateur	94
5.1.2	De la ligne au programme	95
5.1.3	Rendre un programme Python exécutable	98
5.2	Organisation générale d'un programme	99
5.2.1	Règles de syntaxe d'un programme	99
5.2.2	Importation de bibliothèques	100
5.3	Variables numériques, chaînes, listes et dictionnaires	101
5.3.1	Nommage des variables	101
5.3.2	Les quatre types de données et variables de base	101
5.3.3	Chaînes, listes, tuples et dictionnaires	102
5.4	Opérateurs, boucles et prise de décision	106
5.4.1	Opérateurs arithmétiques	107
5.4.2	Opérateurs de comparaison et opérateurs logiques	108
5.4.3	Prise de décision	109
5.4.4	Boucles itératives et conditionnelles	110
5.5	Entrées et sorties de données	112
5.5.1	Sorties de données avec print	112
5.5.2	Entrée de données avec input	114
5.5.3	Écriture dans un fichier	114
5.5.4	Lecture dans un fichier	115
5.6	Fonctions, modules, classes et méthodes	116
5.6.1	Fonctions	116
5.6.2	Modules	117
5.6.3	Classes et méthodes	119
6 •	Des accessoires indispensables	121
6.1	Un poussoir de reset pour votre Raspberry Pi	121
6.2	Des alimentations pour tous les besoins	123
6.2.1	Alimentations 5 volts de 1 à 3 ampères	123

6.2.2	Alimentation 5 volts 3 ampères à haut rendement	125
6.2.3	Utilisation à partir du secteur EDF	127
6.3	Les entrées/sorties du Raspberry Pi	128
6.3.1	Le port GPIO du Raspberry Pi	128
6.3.2	Attention à la tension !	130
6.4	Des cartes d'expérimentation bien utiles	130
6.4.1	Le Pi Cobbler : une excellente idée	131
6.4.2	La carte Pi Proto Plate	132
6.4.3	La carte Humble Pi	134
7 •	Poussoir, LED, relais et afficheurs	137
7.1	Une bibliothèque Python indispensable	138
7.1.1	La bibliothèque RPi.GPIO	138
7.1.2	Les principales fonctions de RPi.GPIO	139
7.2	Les sorties numériques parallèles	141
7.2.1	Commandes de LED et de relais	141
7.2.2	Commande directe de charges alimentées en continu	147
7.2.3	Commande directe de charges reliées au secteur	148
7.3	Les entrées numériques parallèles	149
7.3.1	Poussoirs, commutateurs et capteurs de type tout ou rien	149
7.3.2	Comment vaincre les rebondissements ?	151
7.3.3	Entrées haute tension et opto-isolées	152
7.4	Afficheurs à LED et à cristaux liquides (LCD)	153
7.4.1	Commande directe d'afficheurs à LED	154
7.4.2	Commande directe d'afficheurs à LED multiplexés	155
7.4.3	Utilisation d'afficheurs à cristaux liquides (LCD)	156
8 •	Servos, moteurs pas à pas, moteurs à courant continu et entrées/sorties analogiques	161
8.1	Les servos de radiocommande	161
8.1.1	Principe général	162
8.1.2	La bibliothèque RPIO et la commande de servos	164
8.2	Les moteurs pas à pas	166
8.2.1	Principes généraux	166
8.2.2	Circuits d'interface	167
8.2.3	Le logiciel	169
8.3	Gradation de puissance et moteurs à courant continu	170
8.3.1	Variation de tension et signaux PWM	170
8.3.2	Logiciel de gradation de tension	172
8.3.3	Variation de vitesse d'un moteur à courant continu	173

8.4	Manipulation de tensions analogiques	176
8.4.1	Génération de tensions analogiques	176
8.4.2	Acquisition de tensions analogiques	178
9	• Bus I2C, bus SPI, cartes PiFace et Gertboard	181
9.1	Le bus I2C	181
9.1.1	Généralités	182
9.1.2	Protocole du bus I2C	182
9.1.3	Exemple d'utilisation du bus I2C	186
9.2	L'interface ou bus SPI	190
9.2.1	Principe d'une liaison de type SPI	190
9.2.2	La carte à interface SPI PiFace	192
9.2.3	Les logiciels de la carte PiFace	194
9.3	La carte Gertboard	198
10	• Pour aller plus loin : associer Raspberry Pi et Arduino	201
10.1	Qu'est-ce que l'Arduino ?	202
10.2	Pourquoi relier Raspberry Pi et Arduino ?	203
	Annexe	205
	Index	211

LES COMPLÉMENTS EN LIGNE DE CET OUVRAGE

Vous trouverez sur *www.dunod.com*, sur la page dédiée à cet ouvrage, et sur le site de l'auteur à l'adresse *www.tavernier-c.com*, un certain nombre de compléments en ligne :

- les fiches techniques et assimilées de tous les circuits et composants électroniques actifs utilisés dans l'ouvrage ;
- les bibliothèques citées ou utilisées dans l'ouvrage, autres que celles fournies avec la distribution de base ou directement téléchargeables depuis cette dernière ;
- les logiciels tiers utilisés par certaines applications (création d'images sur carte SD, partitionnement de carte SD, vérification de somme SHA-1, etc.) ;
- les schémas du Raspberry Pi ainsi que diverses cartes d'extension utilisées, lorsque ces derniers sont publics.

Afin de faciliter l'utilisation de ces compléments, tous les fichiers correspondants sont groupés dans des répertoires qui portent les noms des chapitres de l'ouvrage dans lesquels ils apparaissent pour la première fois.

En lisant un chapitre, il suffit donc de vous positionner sur le répertoire correspondant pour avoir accès immédiatement à tous les documents et/ou programmes qui y sont utilisés ou évoqués.

AVANT-PROPOS

Conçu à l'origine pour initier les plus jeunes à la programmation et à la micro-informatique, le Raspberry Pi a très vite connu un tel succès planétaire, que même ses concepteurs n'auraient pas osé l'imaginer. Les quelques centaines d'exemplaires fabriqués au tout début ont ainsi été écoulés en quelques jours et, à la date de rédaction de cet ouvrage, plus d'un million de Raspberry Pi ont été vendus dans le monde entier.

Il faut dire que le produit a de quoi séduire puisque, sur un circuit imprimé à peine plus grand qu'une carte de crédit, vendu un peu moins de quarante euros, on trouve un micro-ordinateur complet qui, même s'il n'a pas la puissance d'un PC, est tout de même capable d'accomplir efficacement la majorité des tâches que l'on confie habituellement à ce dernier.

De plus, grâce à des choix judicieux, tel que l'intégration d'un processeur graphique performant ou bien encore l'accès à certaines lignes d'entrées/sorties du processeur, il est possible d'utiliser le Raspberry Pi aussi bien comme *media center* capable de diffuser des films en haute définition sur la télé du salon que comme automate programmable intelligent pour se constituer une maison domotique, par exemple.

Après vous avoir appris à configurer et à paramétrer correctement votre Raspberry Pi, cet ouvrage vous propose de découvrir ses nombreuses possibilités au travers d'utilisations qui vont de la bureautique aux automatismes en passant par les serveurs Internet et les applications multimédias.

Et pour que vous ne soyez pas prisonnier des seuls programmes écrits ou proposés par d'autres, vous y apprendrez aussi les bases du système d'exploitation Linux utilisé par le Raspberry Pi, ainsi que la programmation en Python, qui vous permettra de mettre en œuvre vos propres interfaces et applications.

1 • À LA DÉCOUVERTE DU RASPBERRY PI

Le Raspberry Pi appartient à ce que l'on appelle le monde du « libre » tant sur le plan logiciel que matériel, c'est-à-dire que tous les documents le concernant, que ce soient les schémas ou les programmes qu'il utilise, sont mis gracieusement à disposition de la communauté internationale par leurs auteurs respectifs.

En contrepartie de cette gratuité, il n'existe pas de structure commerciale, de vendeur si vous préférez, qui se charge de commercialiser et de centraliser tout ce qui concerne le produit et, en particulier, toutes les documentations le concernant. C'est à l'utilisateur de faire l'effort de glaner sur Internet les informations nécessaires en fonction de ce qu'il compte faire de son Raspberry Pi, et il faut bien reconnaître qu'après l'avoir reçu et lorsqu'on le sort de sa boîte, on est un peu perdu devant l'ampleur de la tâche car il est alors inutilisable...

Après une courte présentation du Raspberry Pi, nous allons voir ce qu'il est nécessaire de lui raccorder pour le rendre utilisable, puis nous vous expliquerons comment préparer sa carte mémoire avec le système d'exploitation à télécharger sur Internet afin que, d'un petit circuit imprimé inerte, vous passiez à un micro-ordinateur performant.

1.1 Raspberry Pi A ou B ?

Au moment où ces lignes sont écrites, deux versions de Raspberry Pi sont commercialisées ; la version A et la version B visibles figures 1.1 et 1.2. Si le processeur qui les équipe est identique, tout comme leur architecture interne, voici ce qui les différencie :

- l'interface réseau est absente sur le modèle A ;
- le modèle A ne comporte qu'un port USB contre deux pour le modèle B ;
- la taille de la mémoire du modèle A n'est que de 256 Mo contre 512 Mo pour le modèle B.

Compte tenu de la faible différence de prix entre les deux versions, nous vous recommandons vivement de choisir le modèle B. En effet, l'absence d'interface réseau est bien souvent rédhibitoire et, même s'il est possible d'en adjoindre une au modèle A grâce à un module USB – réseau, le coût de ce module ajouté à celui du modèle A, dépasse largement le prix du modèle B.

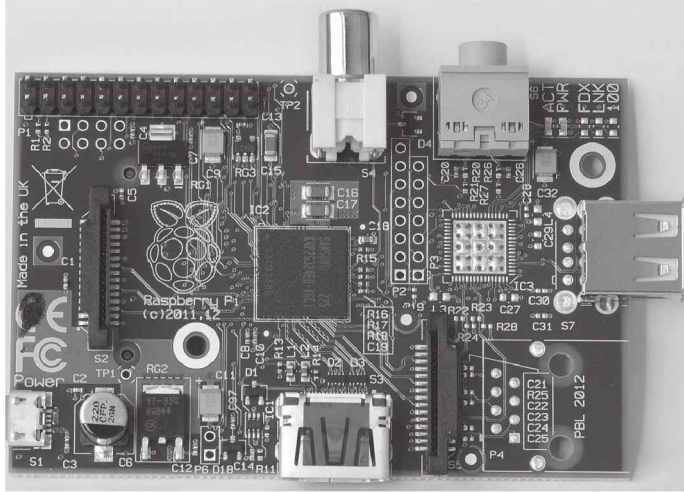


Figure 1.1 – Le Raspberry Pi modèle A. L'absence de contrôleur et de prise réseau est bien visible.

Et quand bien même le réseau ne vous serait pas utile, la présence d'un seul port USB est, elle aussi, handicapante et impose de recourir à un hub USB externe dès que l'on veut utiliser plus d'un périphérique. Là aussi, le prix du hub ajouté à celui du modèle A, dépasse bien souvent celui du modèle B.

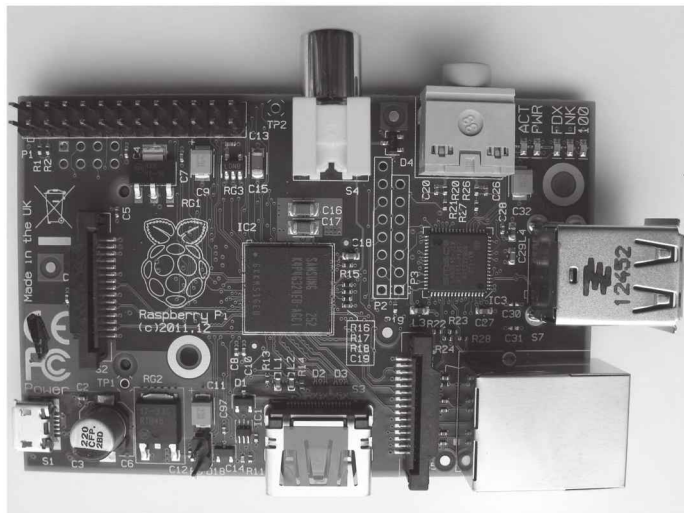


Figure 1.2 – Le Raspberry Pi modèle B avec sa prise réseau et ses deux prises USB.

La mémoire enfin, de par sa petite taille, réduit notablement les performances et interdit même l'exécution de certains logiciels.

1.2 Petite visite guidée

Afin de bien visualiser ce qu'il est possible d'adjoindre à la carte Raspberry Pi nue qui vous a été fournie, nous vous proposons de jeter un coup d'œil à la figure 1.3 qui présente tout à la fois son organisation générale et les différents connecteurs disponibles.

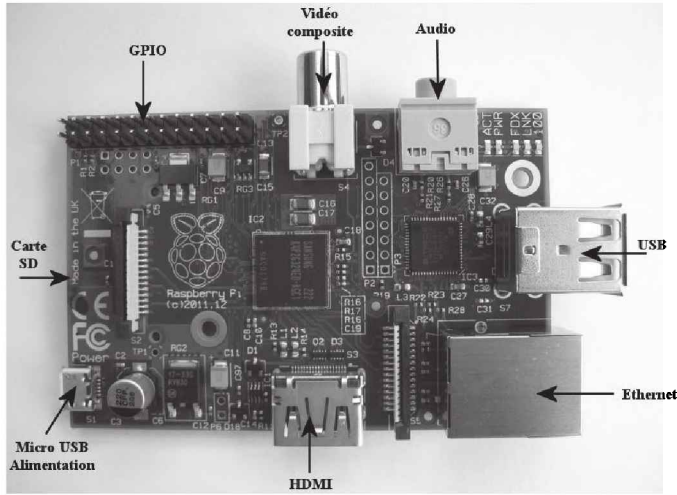


Figure 1.3 – Organisation générale du Raspberry Pi et de ses différents connecteurs.

Le centre de la carte est occupé par le processeur ou, plus exactement, par un empilement du processeur et de sa mémoire qui sont littéralement montés l'un sur l'autre. Nous reviendrons, dans la suite de ce chapitre, sur ce processeur quelque peu particulier. L'autre « gros » circuit intégré visible sur cette face du circuit imprimé n'est autre que le contrôleur réseau, absent bien évidemment sur le modèle A.

Paradoxalement, c'est la connectique qui occupe la majorité de la surface de la carte et non les composants actifs, et ce d'autant que des choix pour le moins curieux ont été faits au niveau des connecteurs, dont certains sont véritablement archaïques sur un produit aussi récent.

Nous vous proposons de partir de la prise réseau, située dans l'angle inférieur droit de la figure 1.3 et de tourner dans le sens inverse des aiguilles d'une montre. Ce connecteur réseau justement, présent sur le seul modèle B, accepte une prise normalisée RJ45 et tout réseau Ethernet classique fonctionnant à 10 Mbits ou 100 Mbits par seconde. Le Gigabit Ethernet n'est donc pas supporté par le Raspberry Pi mais, vu ses performances globales, cela n'est absolument pas contraignant. Le double connecteur USB voisin est quant à lui conforme à la norme USB 2.0, rétro-compatible USB 1.0 comme il se doit.

Dans l'angle situé à côté de ces connecteurs on trouve cinq LED d'état du Raspberry Pi :

- une LED verte d'activité du processeur ;

- une LED rouge de mise sous tension ;
- une LED verte signalant le mode full duplex de la connexion réseau ;
- une autre LED verte signalant l'activité réseau ;
- une LED jaune indiquant que le réseau fonctionne à 100 Mbits par seconde.

Viennent ensuite deux connecteurs véritablement « monstrueux » et dont la justification de l'un d'entre eux au moins nous semble discutable.

Le premier connecteur est un jack femelle stéréo de 3,5 mm de diamètre destiné à la connexion d'un casque, d'une paire de haut-parleurs amplifiés comme ceux que l'on utilise avec les PC, ou bien encore d'un amplificateur externe.

Le second connecteur, au format Cinch ou RCA, véhicule quant à lui le signal vidéo généré par le Raspberry Pi au format vidéo composite PAL ou NTSC et ne permet donc qu'un affichage en basse résolution, sur un vieux moniteur à tube cathodique de surcroît. Nous comprenons assez mal sa raison d'être, et ce d'autant que le Raspberry Pi dispose par ailleurs d'une sortie HDMI, compatible avec tous les moniteurs et téléviseurs récents, capable de surcroît de fournir de la vidéo au format full HD de 1080p. Ce connecteur RCA sera donc très rarement utilisé, sauf si vous voulez recycler de vieux téléviseurs mis au rebut depuis le passage à la TNT !

Le connecteur mâle, à deux rangées de 13 points chacune, qui suit est quant à lui bien plus intéressant. C'est en effet là que se trouvent les signaux d'interface du GPIO (General Purpose Input Output) c'est-à-dire de toutes les interfaces externes proposées par le microprocesseur qui équipe le Raspberry. On y trouve donc pélemêle des lignes de port parallèles, une interface série, un bus I2C, une liaison SPI, etc. Toutes choses que nous exploiterons à partir du chapitre 7 de cet ouvrage pour relier le Raspberry au monde extérieur et lui permettre ainsi de commander et de s'interfacer avec les appareils les plus divers.

Sur l'autre face du circuit imprimé se trouve le connecteur destiné à recevoir une carte SD. Le Raspberry Pi ne dispose en effet d'aucune interface pour mémoire de masse (disque dur ou assimilé) et c'est donc une carte SD qui est chargée de jouer son rôle.

Le retour sur la face supérieure nous permet de découvrir un connecteur micro USB qui ne véhicule par d'interface USB comme on aurait pu le penser, mais qui sert uniquement à alimenter le Raspberry Pi comme nous allons le voir dans un instant.

Le dernier connecteur enfin n'est pas l'un des moindres puisque c'est un modèle HDMI compatible de la norme 1.4, c'est-à-dire la plus récente à ce jour. Il délivre bien évidemment l'image générée par le Raspberry Pi, en haute définition full HD 1920 par 1080p, et véhicule également le son.

Deux autres connecteurs présents sur la carte sont destinés à recevoir des circuits imprimés souples et ne vous seront sans doute d'aucune utilité. Le premier permet en effet de raccorder un écran plat à interface de type DSI, réservé à l'heure actuelle aux seuls industriels et de prix prohibitif comparé à celui du Raspberry Pi. Le second est destiné à une caméra à interface de type CSI qui est en projet mais non encore commercialisée au moment où ces lignes sont écrites.

Ce petit tour du propriétaire étant terminé, nous vous proposons de découvrir sans plus tarder ce qui est nécessaire à la mise en œuvre de notre Raspberry Pi.

1.3 L'alimentation

Même si c'est là une fonction triviale, c'est pourtant le premier élément à prendre en compte lorsque l'on souhaite faire fonctionner son Raspberry Pi, et ce d'autant que cette alimentation peut être la cause de nombreux problèmes sur lesquels nous aurons l'occasion de revenir à plusieurs reprises.

Le Raspberry Pi s'alimente sous une tension unique de 5 volts ; tension sur laquelle il peut consommer jusqu'à 750 mA selon les tâches qu'il exécute. Cette alimentation doit être normalement fournie via le connecteur micro USB placé dans un angle de la carte, tel que nous l'avons vu en figure 1.3, même s'il existe d'autres possibilités décrites ci-après.

Théoriquement, toute alimentation capable de fournir cette tension et de délivrer au moins les 750 mA requis peut convenir, mais cela s'avère très souvent insuffisant. En effet, notre Raspberry Pi dispose également de deux ports USB sur lesquels nous allons être amenés à connecter divers périphériques tels que les classiques clavier et souris par exemple. Or, sauf à faire appel à un hub alimenté intermédiaire, les différents périphériques connectés sur ces ports vont devoir prélever leur alimentation à partir du Raspberry Pi puisque c'est là un des principes de la norme USB. Leur consommation va donc nécessairement s'ajouter aux 750 mA déjà absorbés par le Raspberry Pi seul.

Il est donc prudent de prévoir une alimentation capable de délivrer au moins un ampère si vous ne voulez pas que votre Raspberry Pi fonctionne de façon aléatoire, voire ne fonctionne pas du tout selon le bon vouloir de la consommation des périphériques qui lui seront raccordés.

1.3.1 Le chargeur pour smartphone

La majorité des smartphones actuels dispose d'un chargeur secteur équipé d'un câble micro USB qui délivre une tension de 5 volts. Il est donc tentant d'y faire appel comme l'ont d'ailleurs fait avec plus ou moins de bonheur nombre d'utilisateurs.

Pour simple et économique qu'elle soit, cette solution peut fonctionner... ou non. En effet, selon la marque et le modèle de smartphone auquel est destiné le chargeur, le courant de sortie qu'il peut fournir peut varier de 350 mA à 1 A. Dans le premier cas, tout fonctionnement sera donc impossible alors qu'un modèle délivrant 1 A conviendra parfaitement, même pour un Raspberry Pi relié à plusieurs périphériques.

Si donc vous envisagez cette solution, vérifiez sur le chargeur du smartphone le courant maximum que ce dernier peut délivrer ; il y est toujours indiqué.

1.3.2 Le bloc secteur « prise de courant »

En l'absence de chargeur de portable, la meilleure solution consiste à acheter un bloc secteur au format « prise de courant » disponible chez tous les revendeurs de composants électroniques ainsi que dans nombre de magasins de bricolage.

Comme pour le chargeur de portable, vous veillerez alors à ce qu'il puisse délivrer une tension de 5 volts sous un courant de 1 ampère, voire même un peu plus si vous envisagez de connecter de nombreux périphériques USB à votre Raspberry Pi.

Attention !

Choisissez bien un bloc secteur sur lequel figure la mention « régulé » ou « stabilisé », seul à même de vous garantir une tension de sortie stable de 5 volts, et fuyez comme la peste les blocs secteur bas de gamme, non stabilisés, dont la tension de sortie peut monter très au-delà de 5 volts avec, à la clé, la mort assurée pour votre Raspberry Pi.

1.3.3 Le hub USB actif ou hub alimenté

Dès lors que l'on souhaite connecter au Raspberry Pi autre chose qu'un clavier et une souris, il est nécessaire de faire appel à un hub USB. Celui-ci peut être passif, et il sert alors seulement à démultiplier les deux prises USB originales, mais l'alimentation de tous les périphériques USB continue dans ce cas à être prélevée sur le Raspberry Pi, ou il peut également être actif et disposer alors de sa propre alimentation.

Dans ce dernier cas, outre sa fonction de démultiplication des prises USB, il permet également d'alimenter les périphériques USB, ce qui est une excellente solution si vous envisagez d'en utiliser plusieurs ou de faire appel à des périphériques à forte consommation tel un disque dur par exemple.

Mais ce hub peut aussi servir à alimenter votre Raspberry Pi si ce dernier est un modèle assez récent. En effet, les alimentations de toutes les prises USB sont reliées entre elles sur le Raspberry Pi. Si donc l'on injecte une tension de 5 volts via une des prises USB « normales » du Raspberry Pi, en provenance d'un hub actif alimenté, cette tension l'alimentera normalement.

De ce fait, si l'alimentation du hub actif choisi est capable de fournir au moins 1 ampère, elle pourra alimenter le Raspberry Pi en plus des périphériques raccordés à ce hub, vous dispensant ainsi d'acquérir une alimentation spécifique pour le seul Raspberry Pi.

Cette possibilité n'est cependant offerte qu'aux Raspberry Pi les plus récents. En effet, sur les premiers modèles commercialisés, des fusibles réarmables de type Polyswitches, d'un courant de coupure de 160 mA, étaient placés au niveau des prises USB normales. Il était donc impossible de fournir plus de 160 mA au Raspberry Pi via ces prises. Ces fusibles ayant disparu sur les versions les plus récentes, cette restriction n'existe plus et l'alimentation depuis un hub USB devient donc possible.

Comme rien, au niveau de la référence, ne vient distinguer les Raspberry Pi avec fusibles de ceux qui n'en ont pas, nous vous proposons en figure 1.4 un gros plan sur la zone du circuit imprimé supportant ces derniers afin que vous puissiez déterminer de quel type de Raspberry Pi vous êtes en possession.

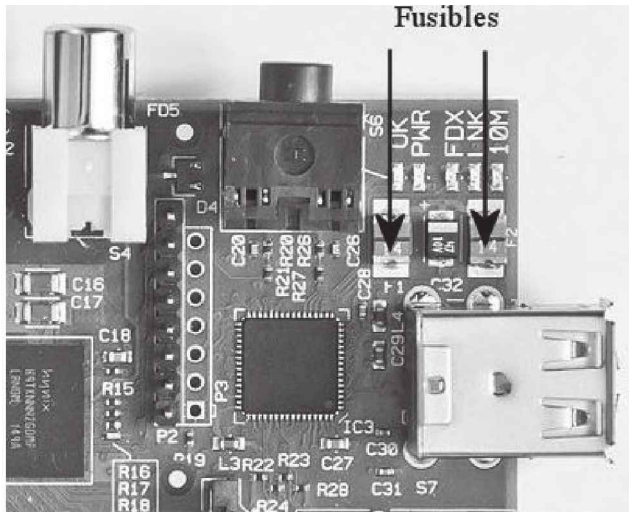


Figure 1.4 – Sur les Raspberry Pi équipés de fusibles, ceux-ci sont bien visibles.

1.4 Le clavier et la souris

Rien de bien particulier n'est à signaler en ce qui les concerne. N'importe quel clavier USB et n'importe quelle souris USB conviennent au Raspberry Pi et se connectent directement dans les deux ports USB prévus à cet effet sur le modèle B. Pour ce qui est du modèle A, il faut déjà faire appel à un hub pour pouvoir utiliser simultanément clavier et souris.

Si vous disposez d'anciens claviers et souris de PC munis d'une interface PS/2 (la prise ronde à 6 points), ils peuvent éventuellement être utilisés avec le Raspberry Pi au moyen d'un adaptateur spécial tel celui visible figure 1.5.



Figure 1.5 – Adaptateur double USB pour clavier et souris à interface PS/2.

Attention !

Les prix de tels adaptateurs font « le grand écart » puisque nous avons pu en trouver à des prix variant entre 8 et 30 euros pour une qualité quasiment identique !

1.5 L'écran

Deux solutions fort différentes vous sont proposées comme vous avez pu le découvrir lors de notre visite guidée ci-dessus : l'utilisation de la sortie HDMI, vivement conseillée en raison de ses possibilités, et l'utilisation de la prise Cinch ou RCA, à ne choisir que si vous ne pouvez vraiment pas faire autrement.

Voyons tout d'abord l'utilisation de la prise HDMI et de ses variantes que sont le DVI et, dans une moindre mesure, le VGA.

1.5.1 HDMI, DVI ou même VGA

Par défaut, le Raspberry Pi délivre des signaux à la norme HDMI 1.4 comportant vidéo et son. Il peut donc être raccordé directement à tout moniteur informatique ou à tout téléviseur à écran plat disposant d'une telle entrée.

Si l'écran choisi est équipé de haut-parleurs, ces derniers reproduiront le son fourni par le Raspberry Pi puisque celui-ci est véhiculé par la liaison HDMI. C'est bien sûr le cas des téléviseurs à écran plat mais pas nécessairement celui des moniteurs informatiques.

Dans une telle situation, vous pourrez alors récupérer le son depuis la prise jack stéréo de 3,5 mm que nous avons vue en figure 1.3. Si vous voulez pouvoir disposer d'un niveau sonore réglable et surtout confortable, le recours à des haut-parleurs amplifiés pour PC est vivement conseillé. Si aucun son n'est disponible au niveau de cette prise jack, nous vous invitons à prendre connaissance de la méthode à utiliser pour y remédier au chapitre 4.4.2 plus avant dans cet ouvrage.

Si vous ne disposez que d'un moniteur à interface DVI, son utilisation est également possible sans problème avec le Raspberry Pi. Il vous suffit juste d'acheter un câble HDMI – DVI, disponible pour quelques euros chez tout bon revendeur informatique ou accessoiriste pour Raspberry Pi, et le tour sera joué.

Attention !

Une prise DVI ne véhicule en aucun cas le son, que le moniteur qui en est équipé dispose ou non de haut-parleurs. Si donc vous utilisez un câble HDMI – DVI vous serez obligé de récupérer le son via la prise jack stéréo.

Si vous souhaitez utiliser un moniteur plus ancien, muni seulement d'une entrée de type VGA, ce n'est pas possible directement avec le Raspberry Pi, même en utilisant certains câbles ou adaptateurs passifs disponibles dans le commerce ou fournis très souvent avec les cartes graphiques de PC. En effet, la sortie HDMI du Raspberry Pi ne délivre pas les signaux analogiques nécessaires à ce type de liaison et il n'est pas possible de les fabriquer au moyen d'un adaptateur passif.

En revanche, on trouve depuis peu, chez Farnell notamment (<http://fr.farnell.com>), sous la dénomination Pi-View un adaptateur HDMI – VGA actif, que vous pouvez

découvrir en figure 1.6, et qui permet donc de relier sans problème un moniteur VGA à la sortie HDMI du Raspberry Pi.



Figure 1.6 – L'adaptateur HDMI – VGA Pi-View.

Cet adaptateur est auto alimenté par le port HDMI et ne nécessite donc pas d'alimentation externe. Par ailleurs, il supporte les affichages haute définition en modes 720p, 1080i et 1080p, pour peu bien sûr que le moniteur qui lui est raccordé en soit également capable.

1.5.2 La sortie vidéo composite via la prise Cinch ou RCA

Cette prise Cinch délivre, comme nous l'avons indiqué ci-dessus, un signal vidéo composite de qualité bien inférieure à celle permise par la sortie HDMI. Le recours à cette sortie ne sera donc qu'un pis-aller et ne permettra pas de dépasser une résolution de 640 par 480p, très en deçà de ce dont nous avons l'habitude aujourd'hui.

En contrepartie, cette sortie est compatible avec n'importe quel vieux téléviseur à tube cathodique, via sa prise péritélévision. Pour pouvoir l'utiliser, il suffit de faire appel à un adaptateur péritélévision – prises Cinch, tel celui de la figure 1.7 par exemple, sur lequel vous utiliserez bien évidemment la prise Cinch jaune qui véhicule la vidéo.



Figure 1.7 – Adaptateur prise péritélévision – prises Cinch.

1.6 Le réseau

Même s'il n'est pas absolument indispensable de connecter le Raspberry Pi à un réseau, puisque, par exemple, le modèle A est dépourvu d'une telle interface ; cela s'avère cependant très vite nécessaire du fait de l'accès à Internet que cela procure. En effet, tous les logiciels susceptibles d'être utilisés avec le Raspberry Pi sont téléchargeables sur Internet ou doivent être mis à jour par ce biais, ce qui rend bien vite cet accès indispensable.

Fort heureusement, si vous disposez d'un réseau filaire, sa connexion au Raspberry Pi est immédiate puisqu'il suffit de faire appel à n'importe quel câble RJ 45 standard. Rappelons que l'interface réseau ne fonctionne ici qu'à 10 ou 100 Mbits par seconde et qu'il n'est donc pas utile de recourir à du câble réseau de catégorie 6 destiné aux liaisons de type Gigabit Ethernet.

L'interface réseau du Raspberry Pi est évidemment de type auto MDI et accepte donc indifféremment les câbles réseau droits ou croisés.

Si vous ne disposez que d'un accès WiFi à votre réseau, il faudra faire l'acquisition d'une clé USB – WiFi car le Raspberry Pi n'intègre pas en standard un module WiFi. Même si de très nombreuses clés de ce type peuvent être utilisées avec succès avec le Raspberry Pi, nous vous conseillons cependant de consulter le site http://elinux.org/RPi_VerifiedPeripherals où figure une liste des références tenues à jour (en principe !) des clés dont le fonctionnement est assuré.

Si vous devez recourir à une interface réseau WiFi, nous vous proposons de poursuivre la mise en œuvre de votre Raspberry Pi en lisant ce chapitre, sans bénéficier dans un premier temps de la connectivité Internet, puis de consulter le chapitre 3 où sont détaillés les différents paramétrages réseau, que celui-ci soit filaire ou WiFi.

1.7 Préparation de la carte SD

Le Raspberry Pi ne dispose en standard d'aucune mémoire de masse. Ce rôle est donc dévolu à une carte mémoire de type SD qu'il faut enficher dans le connecteur prévu à cet effet. Cette carte mémoire, de type SD ou SD HC, peut avoir toute taille comprise entre 2 Go et 32 Go. Comme elle supporte le système d'exploitation ainsi que les éventuels logiciels supplémentaires que vous allez utiliser avec le Raspberry Pi, nous vous conseillons de choisir au moins une taille de 8 Go, voire même 16 Go puisque le prix de telles cartes est aujourd'hui très abordable.

Si vous n'êtes pas à un ou deux euros près, choisissez une carte de classe 8 ou 10, un peu plus rapide que les classiques cartes de classe 4 ou inférieure mais, avant de fixer votre choix, consultez le site http://elinux.org/RPi_VerifiedPeripherals qui dresse un tableau très complet de la majorité des cartes actuellement disponibles sur le marché qui fonctionnent sans problème avec le Raspberry Pi.

En effet, pour des raisons inexplicables, même par son concepteur, et alors que l'interface des cartes SD et SD HC est parfaitement normalisée, toutes ne fonctionnent pas nécessairement avec le Raspberry Pi.

Tout comme le disque dur d'un PC doit être préparé avec le système d'exploitation que l'on souhaite utiliser, la carte mémoire destinée au Raspberry Pi doit également être initialisée avec le système d'exploitation choisi.

Le Raspberry Pi n'utilisant pas un processeur disposant d'un jeu d'instructions compatibles de la famille x86 d'Intel, puisqu'il repose sur un processeur de type ARM 11, il n'est pas possible d'utiliser Windows, pas plus d'ailleurs que les différentes versions de Linux destinées à un PC. Fort heureusement, Linux appartenant au monde du logiciel libre, plusieurs versions (on dit habituellement « distributions ») ont été recompilées pour le Raspberry Pi et c'est donc avec ce système d'exploitation que vous allez devoir l'utiliser.

Rassurez-vous toutefois, si vous ne connaissez que l'environnement Windows, la transition ne sera pas trop difficile, et ce d'autant que nous vous présenterons les bases indispensables de Linux au chapitre suivant de cet ouvrage.

1.7.1 Téléchargement et vérification de la distribution Linux

La préparation de la carte SD va nécessiter un PC, fonctionnant sous Windows ou sous Linux, peu importe, équipé d'au moins un lecteur de carte SD bien évidemment. C'est sur ce PC que sera téléchargée en premier lieu la distribution Linux choisie depuis le site de la fondation Raspberry Pi à l'adresse : <http://www.raspberrypi.org/downloads>.

Depuis sa première commercialisation, et fort de son succès, le Raspberry Pi s'est vu doter de plusieurs distributions Linux présentes sur la page de téléchargement ci-dessus. Sauf si vous maîtrisez déjà bien Linux, nous vous conseillons de choisir la distribution **Raspbian** « **wheezy** » qui vous permettra d'accéder à toutes les possibilités du Raspberry Pi tout en étant d'un abord plus facile que certaines autres présentes sur cette même page.

Deux liens de téléchargement sont proposés : un lien direct avec des serveurs miroirs répartis sur toute la planète afin de vous garantir un débit correct, et un lien de type bit torrent pour ceux d'entre vous qui auraient des problèmes de débit ou d'accès aux liens directs.

Dans les deux cas, vous vous retrouverez au final avec un fichier compressé baptisé **20xx-xx-xx-wheezy-raspbian.zip** où les caractères **x** seront remplacés par la date de création de votre distribution.

Afin d'être certain de la qualité du téléchargement, nous vous conseillons de vérifier votre fichier téléchargé, et ce d'autant que le site fournit la somme de contrôle SHA-1 de celui-ci comme vous pouvez le constater figure 1.8.

Raspbian "wheezy"

If you're just starting out, **this is the image we recommend you use**. It's a reference root filesystem from Alex and Dom, based on the Raspbian optimised version of Debian, and containing LXDE, Midori, development tools and example source code for multimedia functions.

Torrent	2013-02-09-wheezy-raspbian.zip.torrent
Direct download	2013-02-09-wheezy-raspbian.zip
SHA-1	b4375dc9d140e6e46e0406f96dead3601fac6c81
Default login	Username: pi Password: raspberry

Figure 1.8 – La somme de contrôle SHA-1 permet de s'assurer de l'intégrité du fichier téléchargé.

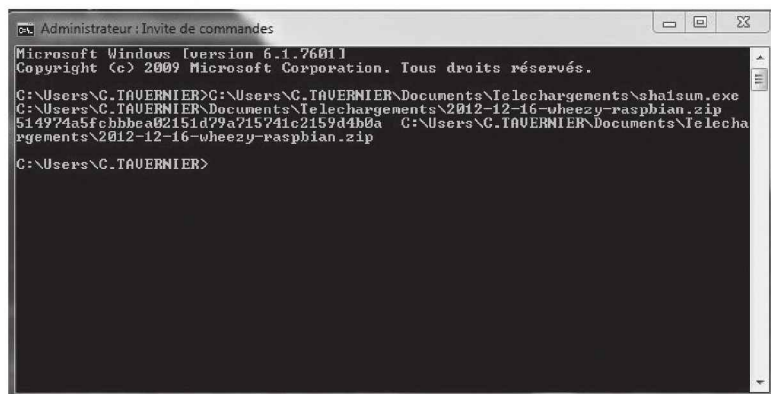
■ Sous Linux

Si votre PC fonctionne sous Linux, vous disposez en principe du programme `sha1sum` qui permet de calculer la somme de contrôle à partir de la distribution que vous avez téléchargée ; somme de contrôle qu'il ne reste plus ensuite qu'à comparer avec celle affichée sur la page du site visible figure 1.8.

■ Sous Windows

Si votre PC fonctionne sous Windows, aucun outil standard ne permet un tel calcul. Nous vous conseillons donc de télécharger `sha1sum` à cette adresse : <http://downloads.raspberrypi.org/tools/sha1sum.exe> ou depuis les compléments en ligne de cet ouvrage.

Ouvrez ensuite une fenêtre DOS et faites glisser dans celle-ci le fichier `sha1sum.exe` fraîchement téléchargé, puis saisissez un espace à la fin de la ligne correspondante et faites ensuite glisser le fichier compressé de la distribution que vous avez téléchargée, puis validez par **Entrée**. La fenêtre DOS doit alors ressembler à ce que vous pouvez découvrir figure 1.9.



```
Administrateur: Invite de commandes
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.

C:\Users\C.TAUERNIER>C:\Users\C.TAUERNIER\Documents\Telechargements\sha1sum.exe
C:\Users\C.TAUERNIER\Documents\Telechargements\2012-12-16-wheezy-raspbian.zip
514974a5fcbbea02151d79a715741c2159d4b0a C:\Users\C.TAUERNIER\Documents\Telech
rgements\2012-12-16-wheezy-raspbian.zip

C:\Users\C.TAUERNIER>
```

Figure 1.9 – Vérification de la distribution téléchargée au moyen de la somme SHA-1.

Il ne reste plus alors qu'à relever la somme de contrôle SHA-1 calculée à partir de votre téléchargement et à la comparer avec celle fournie sur la page de téléchargement de la figure 1.8 pour vous assurer de sa validité. Toute différence à ce niveau impose bien évidemment de télécharger à nouveau votre distribution.

Dans les deux cas, Linux ou Windows, décompressez le fichier téléchargé, ce qui a pour effet de générer un fichier portant le même nom que le fichier téléchargé mais au suffixe `img` cette fois-ci. Placez-le dans un répertoire facile d'accès dont vous noterez le nom.

1.7.2 Préparation de la carte SD

La préparation de la carte SD consiste tout simplement à écrire l'image téléchargée sur cette carte de façon à ce que le Raspberry Pi puisse « booter » correctement lors de sa mise sous tension. Ici encore, la procédure est différente selon que l'on utilise un PC sous Linux, qui dispose alors de la commande « qui va bien », ou d'un PC sous Windows pour lequel il sera nécessaire de télécharger un petit utilitaire spécialisé.

■ Sous Linux

Commencez par ouvrir une fenêtre console ou terminal et enfichez la carte destinée au Raspberry Pi dans le lecteur de carte du PC.

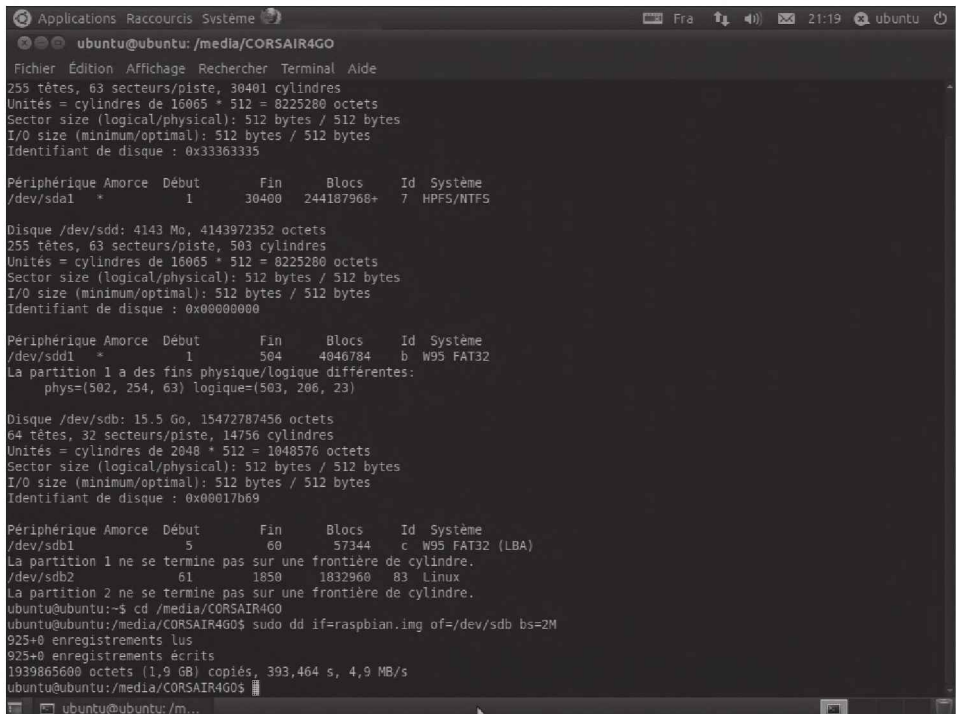
Saisissez la commande `sudo fdisk -l` afin de lister à l'écran toutes les mémoires de masse présentes sur votre machine et, en vous aidant de leurs noms et/ou de leurs tailles respectives, repérez l'appellation « Linux » de la carte mémoire. Elle doit être de la forme `/dev/sdx` où `x` est la lettre identifiant précisément la carte ou éventuellement `/dev/mmcblkx` sur certains systèmes.

Utilisez ensuite la commande `cd` afin de vous placer dans le répertoire contenant l'image issue de la décompression du fichier précédemment téléchargé.

Saisissez alors, sans faute de frappe, la commande `dd if=image.img of=/dev/sdx bs=2M` où `image.img` est le nom du fichier image décompressé et où `sdx` est le nom de la carte mémoire déterminé précédemment.

■ Attention !

Si vous avez mal identifié l'appellation Linux et que le `sdx` saisi dans cette commande ne correspond pas à votre carte mais à une autre mémoire de masse de votre PC, tout ce qu'elle contient sera détruit par la commande `dd`.



```
Applications Raccourcis Système
ubuntu@ubuntu: /media/CORSAIR4GO
Fichier Édition Affichage Rechercher Terminal Aide
255 têtes, 63 secteurs/piste, 30401 cylindres
Unités = cylindres de 16065 * 512 = 8225280 octets
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Identifiant de disque : 0x33363335

Périphérique Amorç Début Fin Blocs Id Système
/dev/sda1 * 1 30400 244187968+ 7 HPFS/NTFS

Disque /dev/sdd: 4143 Mo, 4143972352 octets
255 têtes, 63 secteurs/piste, 503 cylindres
Unités = cylindres de 16065 * 512 = 8225280 octets
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Identifiant de disque : 0x00000000

Périphérique Amorç Début Fin Blocs Id Système
/dev/sdd1 * 1 504 4046784 b W95 FAT32
La partition 1 a des fins physique/logique différentes:
phys=(502, 254, 63) logique=(563, 206, 23)

Disque /dev/sdb: 15.5 Go, 15472787456 octets
64 têtes, 32 secteurs/piste, 14756 cylindres
Unités = cylindres de 2048 * 512 = 1048576 octets
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Identifiant de disque : 0x00017b69

Périphérique Amorç Début Fin Blocs Id Système
/dev/sdb1 5 60 57344 c W95 FAT32 (LBA)
La partition 1 ne se termine pas sur une frontière de cylindre.
/dev/sdb2 61 1850 1832960 83 Linux
La partition 2 ne se termine pas sur une frontière de cylindre.
ubuntu@ubuntu:~$ cd /media/CORSAIR4GO
ubuntu@ubuntu:/media/CORSAIR4GO$ sudo dd if=raspbian.img of=/dev/sdb bs=2M
925+0 enregistrements lus
1939865600 octets (1,9 GB) copiés, 393,464 s, 4,9 MB/s
ubuntu@ubuntu:/media/CORSAIR4GO$
```

Figure 1.10 – Déroulement de la préparation de la carte sur un PC fonctionnant sous Linux.

La création de l'image sur la carte démarre immédiatement après cette commande et ne conduit à aucun affichage particulier pendant son déroulement qui peut nécessiter plusieurs minutes. Soyez donc patient et ne touchez à rien tant que n'apparaît pas à l'écran le message de bonne fin qui doit être de la forme :

```
xxx enregistrements lus
xxx enregistrements écrits
xxx octets copiés.
```

La carte mémoire est alors prête à l'emploi et peut être retirée du lecteur afin d'être insérée dans le connecteur prévu à cet effet sur le Raspberry Pi.

■ Sous Windows

Windows ne dispose d'origine d'aucune commande équivalente à la commande `dd` de Linux et la création de la carte mémoire va donc devoir faire appel à un programme externe appelé **Win32 Disk Imager** que vous pourrez par exemple télécharger à cette adresse : <http://sourceforge.net/projects/win32diskimager/> ou depuis les compléments en ligne de cet ouvrage.

Décompressez le fichier ainsi obtenu dans n'importe quel dossier de votre choix car il ne requiert aucune installation particulière.

Enfichez la carte mémoire dans le lecteur du PC et lancez Win32 Disk Imager en double cliquant sur le fichier `Win32DiskImager.exe` du dossier précité.

La lettre correspondant à la carte mémoire doit apparaître dans la boîte déroulante située en haut à droite sous l'appellation `Device` tandis que l'icône représentant un dossier va vous permettre de naviguer jusqu'à celui contenant l'image résultant de la décompression du fichier de la distribution précédemment téléchargée. Le tout devant au final se présenter comme indiqué figure 1.11.

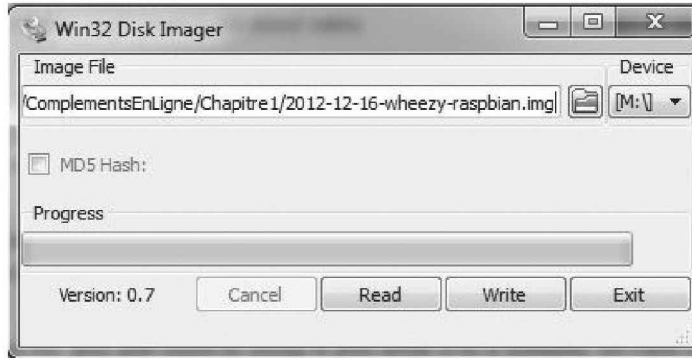


Figure 1.11 – Win32DiskImager est prêt à créer l'image de la distribution Linux sur la carte mémoire.

Il ne reste plus alors qu'à cliquer sur le bouton `Write` et à patienter jusqu'à ce que la barre de progression affiche 100 %.

Attention !

Le contenu de la carte mémoire est irrémédiablement détruit par cette opération. Veillez donc à bien choisir une carte vierge et, si plusieurs médias amovibles sont présents sur votre PC, à bien choisir le bon dans la boîte déroulante `Device`.

La carte mémoire est alors prête à l'emploi et peut être retirée du lecteur afin d'être insérée dans le connecteur prévu à cet effet sur le Raspberry Pi.

1.8 Mémoires de masse externes

Outre le fait de supporter le système d'exploitation du Raspberry Pi, la carte SD que nous venons de préparer lui sert aussi de mémoire de masse. C'est pour cette raison que nous vous avons conseillé de choisir une carte d'au moins 8 ou 16 Go octets de capacité ; le prix des 32 Go nous semblant encore prohibitif au moment où ces lignes sont écrites.

Si ces capacités sont suffisantes pour nombre d'applications, il peut s'avérer parfois nécessaire de bénéficier de tailles plus importantes, lorsque l'on veut pouvoir visionner des films en HD par exemple puisque, comme nous le verrons au chapitre 4, le Raspberry Pi peut constituer un *media center* très performant.

Dans une telle situation, il faut recourir à de la mémoire de masse externe via l'un de ses ports USB ce qui ne présente aucune difficulté si ce n'est, une fois encore, de prendre quelques précautions vis-à-vis de la consommation.

La consommation des clés USB étant relativement faible, elles ne nécessitent aucune précaution particulière et peuvent donc être utilisées sans restriction.

Pour ce qui est des disques durs externes, il ne faut recourir qu'à des disques durs disposant de leur propre alimentation. En effet, même si la consommation des disques durs auto-alimentés a beaucoup diminué ces temps derniers, elle est toujours de l'ordre de plusieurs centaines de mA qui sont donc à ajouter aux 750 mA déjà absorbés par le Raspberry Pi. De tels disques risquent donc de mettre à genoux l'alimentation de votre Raspberry Pi, sauf bien sûr s'il dispose de sa propre alimentation et que vous ayez en plus fait appel à un hub alimenté pour y connecter un tel disque dur.

Hormis cet éventuel problème de consommation, ces périphériques externes sont parfaitement utilisables avec le Raspberry Pi, après les avoir montés bien sûr, selon la terminologie propre à Linux que nous vous proposons de découvrir sans plus attendre avec le chapitre 2.

2 • À LA DÉCOUVERTE DE RASPBIAN

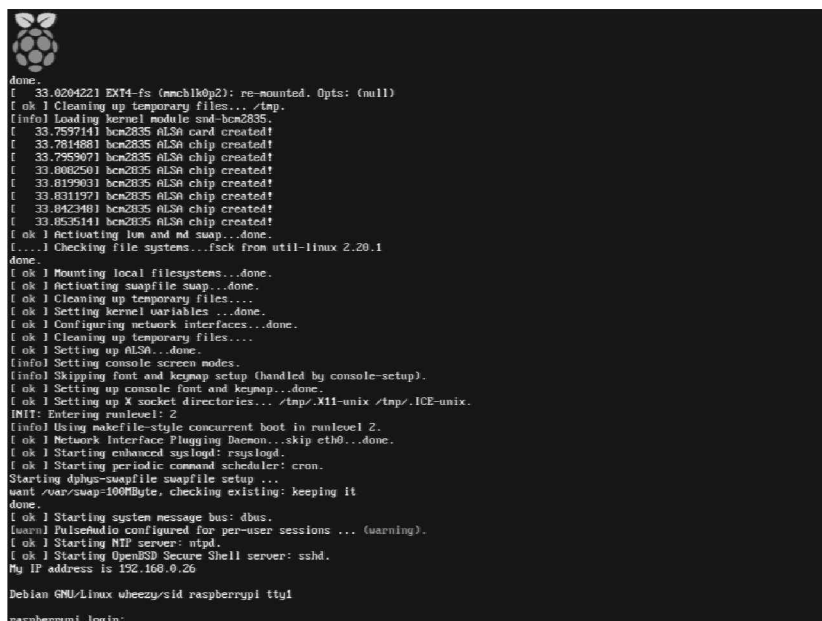
Raspbian, et plus précisément Raspbian Wheezy, est le système d'exploitation du Raspberry Pi par lequel il est conseillé de commencer, surtout si vous n'êtes pas familier du monde Linux et de son fonctionnement en mode ligne de commande. C'est d'ailleurs celui que nous vous avons fait installer sur la carte mémoire à la fin du chapitre précédent et que nous vous proposons de découvrir tout au long de ce chapitre.

Ne croyez pas cependant qu'il s'agisse d'un système d'exploitation « au rabais » et qu'il va vous falloir oublier les interfaces à fenêtres, les icônes et le travail à la souris auquel Windows vous a peut-être habitué. Certes, un certain nombre d'actions, essentiellement techniques et/ou de configuration, vont devoir être réalisées en mode ligne de commande mais Raspbian dispose aussi d'une interface graphique de très belle facture, propre à réconcilier les aficionados de Windows avec le monde Linux.

2.1 Première mise sous tension

Si vous avez déjà tenté de mettre le Raspberry Pi sous tension, sans avoir inséré de carte SD ou en ayant utilisé une carte non préparée, vous avez certainement pu constater que, contrairement à ce qui se passe avec un PC, rien ne s'affichait sur l'écran. En effet, il n'y a pas ici de BIOS qui, sur un PC et même en l'absence de système d'exploitation, est à même de gérer la machine à bas niveau et de la configurer. Cette configuration va donc avoir lieu lors du premier démarrage au moyen de la carte SD supportant le système d'exploitation.

Cette première mise sous tension se traduit, comme indiqué figure 2.1, par un long défilement de textes plus ou moins ésotériques dont vous pouvez ignorer la signification pour l'instant. Certaines lignes affichées contiendront même peut être des messages d'avertissement ou « warnings », repérés en orange, mais il n'y aura pas lieu d'en tenir compte.



```

done.
[ 33.020422] EXT4-fs (mmcblkp2): re-mounted. Opts: (null)
[ ok ] Cleaning up temporary files... /tmp.
[info] Loading kernel module snd-bcm2835.
[ 33.783714] bcm2835 ALSA card created!
[ 33.781488] bcm2835 ALSA chip created!
[ 33.795307] bcm2835 ALSA chip created!
[ 33.808250] bcm2835 ALSA chip created!
[ 33.819903] bcm2835 ALSA chip created!
[ 33.831197] bcm2835 ALSA chip created!
[ 33.842348] bcm2835 ALSA chip created!
[ 33.853514] bcm2835 ALSA chip created!
[ ok ] Activating lun and md swap...done.
[....] Checking file systems...fsck from util-linux 2.20.1
done.
[ ok ] Mounting local filesystems...done.
[ ok ] Activating swapfile swap...done.
[ ok ] Cleaning up temporary files....
[ ok ] Setting kernel variables ..done.
[ ok ] Configuring network interfaces...done.
[ ok ] Cleaning up temporary files....
[ ok ] Setting up ALSA...done.
[info] Setting console screen modes.
[info] Skipping font and keymap setup (handled by console-setup).
[ ok ] Setting up console font and keymap...done.
[ ok ] Setting up X socket directories... /tmp/.X11-unix /tmp/.ICE-unix.
INIT: Entering runlevel: 2
[info] Using makefile-style concurrent boot in runlevel 2.
[ ok ] Network Interface Plugging Daemon...skip eth0...done.
[ ok ] Starting enhanced syslogd: rsyslogd.
[ ok ] Starting periodic command scheduler: cron.
Starting dhup-swapfile swapfile setup ...
want /var/swap-100MByte, checking existing: keeping it
done.
[ ok ] Starting system message bus: dbus.
[warn] Pulseaudio configured for per-user sessions ... (warning).
[ ok ] Starting NTP server: ntpd.
[ ok ] Starting OpenSSH Secure Shell server: sshd.
My IP address is 192.168.0.26
Debian GNU/Linux wheezy/sid raspberrypi tty1
raspberrypi login:

```

Figure 2.1 – Extrait de l'écran de démarrage typique du Raspberry Pi.

Lorsque cette phase de démarrage est terminée, un écran de configuration similaire à celui visible figure 2.2 s'affiche. Même si vous êtes pressé d'aller plus loin, arrêtez-vous quelques instants à son niveau afin de configurer au moins quelques paramètres de base qui rendront l'utilisation ultérieure de Raspbian plus agréable.

Attention !

Si vous avez été trop rapide (ou trop pressé !) et que vous avez passé cet écran de configuration sans l'utiliser, sachez qu'il ne s'affiche que lors de la première mise sous tension. Pour y revenir ensuite il faut faire appel à une commande que nous présenterons dans la suite de ce chapitre.

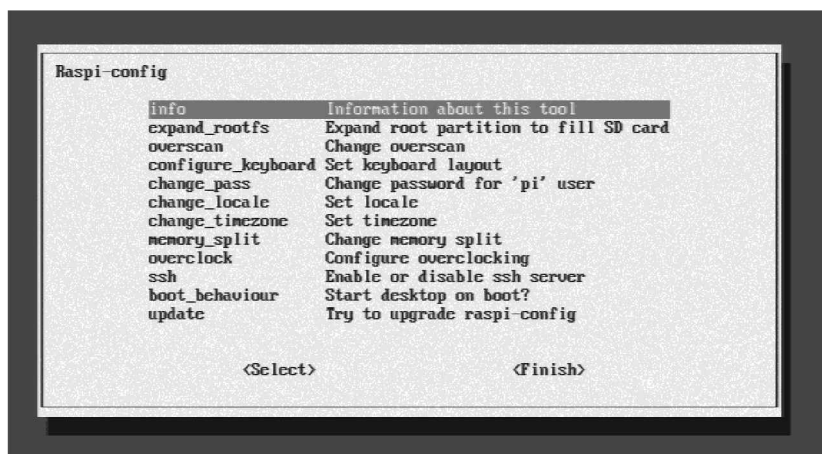


Figure 2.2 – L'écran de configuration visible lors de la première mise sous tension.

2.1.1 Sélection du clavier Français

Au moyen des flèches de déplacement du clavier, surlignez la ligne `configure_keyboard` de façon à paramétrer le Raspberry Pi pour fonctionner avec un clavier français, ou AZERTY si vous préférez, puis validez par `Entrée`. Après quelques secondes, un écran similaire à celui visible figure 2.3 est affiché.

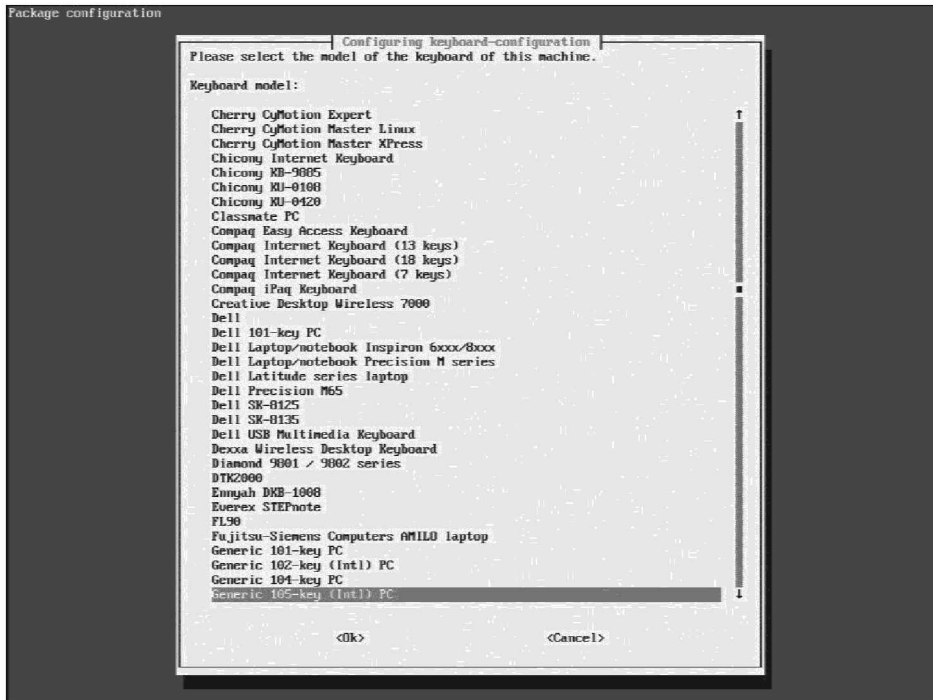


Figure 2.3 – L'écran de sélection de type de clavier.

Par défaut, la ligne `Generic 105-key (Intl) PC` est surlignée et correspond au clavier standard à 105 touches que l'on rencontre sur la majorité des PC. Validez par `ok` ou, si par extraordinaire vous utilisez un clavier spécifique et que celui-ci figure dans cette liste, fort longue, et que vous pouvez parcourir avec l'ascenseur situé sur sa droite, sélectionnez-le avant de valider par `ok`.

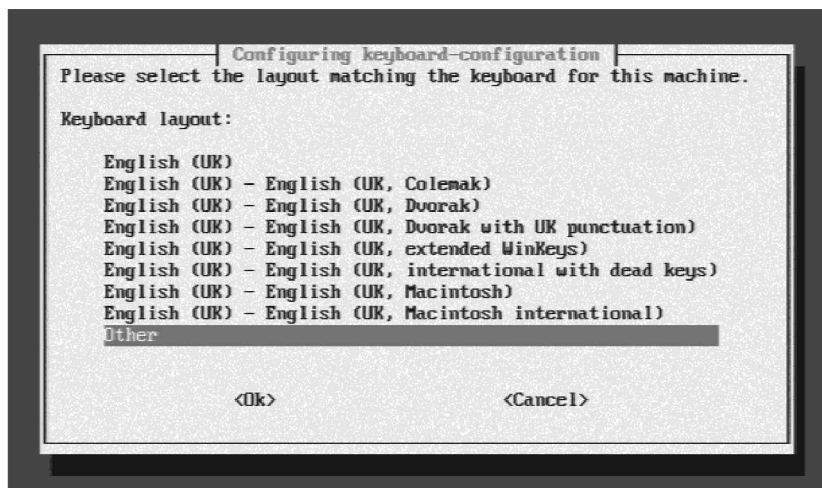


Figure 2.4 – Par défaut, le clavier est paramétré sur English.

L'écran visible figure 2.4 vous permet alors de sélectionner la langue du clavier. Par défaut, il doit théoriquement se positionner sur **English** comme indiqué sur cette même figure. Si vous utilisez un clavier QWERTY validez par **Ok**, dans le cas contraire, sélectionnez **Other** ce qui va avoir pour effet de faire afficher une longue liste de toutes les langues disponibles. Sélectionnez le français, puis validez par **Ok** ce qui fait afficher un nouvel écran avec « tous les français » disponibles comme indiqué figure 2.5.

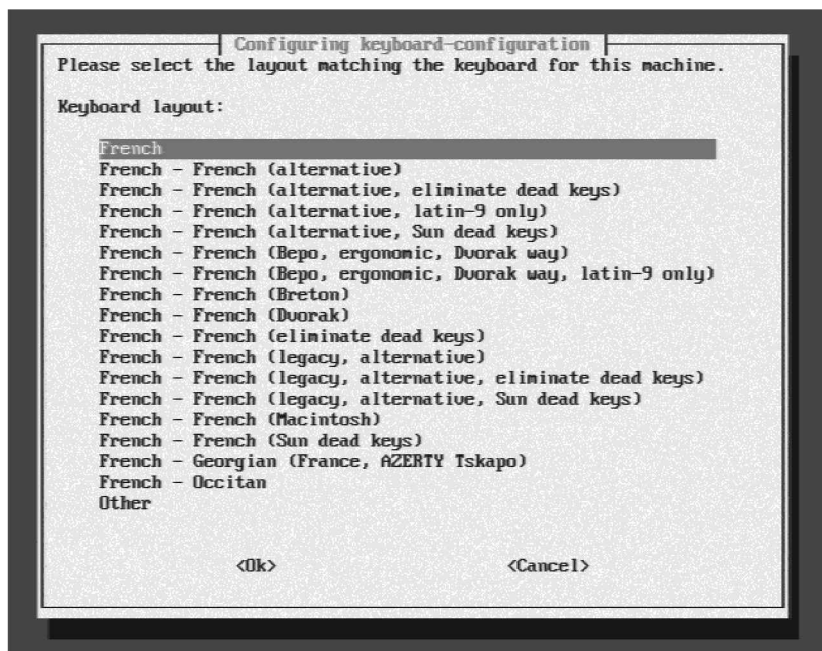


Figure 2.5 – Raspbian pousse le zèle jusqu'à proposer plusieurs types de français !

Sauf cas particulier, sélectionnez la ligne **French** et validez par **Ok** afin de bénéficier d'une prise en charge correcte de votre clavier français. Un nouvel écran s'affiche alors afin de configurer le fonctionnement de la touche **Alt Gr** du clavier. Sauf besoin légitime, laissez la sélection par défaut et validez à nouveau par **Ok**. Vient ensuite l'écran de définition du mode de saisie de la combinaison de touches permettant de générer des symboles non présents sur le clavier, ce que Raspbian appelle la « **compose key** ». Laissez pour l'instant la sélection par défaut (**No compose key**) et validez par **Ok**.

Enfin, il vous est proposé de programmer le fonctionnement de la combinaison **Control Alt Suppr** afin qu'elle termine ou non le serveur X. Laissez le choix par défaut sur **No** et validez par **Entrée**. Après quelques secondes nécessaires au rechargement de la nouvelle configuration du clavier, l'écran de paramétrage de la figure 2.2 devient à nouveau visible.

2.1.2 Localisation de votre distribution

Déplacez ensuite la surbrillance sur **change_locale** et validez par **Entrée** puis faites défiler la longue liste qui apparaît alors jusqu'à rencontrer la ligne commençant par un carré entre parenthèses. Frappez sur **Espace** pour effacer ce carré et faites à nouveau défiler cette liste jusqu'à trouver les deux lignes qui commencent par **fr_FR ISO-8859-1** et **fr_FR UTF-8 UTF-8**. Frappez à nouveau sur **Espace** à leur niveau pour faire apparaître un carré entre parenthèses devant chacune d'entre elles, comme indiqué figure 2.6, puis validez cet écran par **Ok**.



Figure 2.6 – Votre distribution Raspbian Wheezy est désormais en français.

Enfin, amenez la surbrillance sur `change_timezone` et, sur l'écran qui suit, sélectionnez **Europe** puis **Paris** afin de mettre votre Raspberry Pi à l'heure française. Même s'il est possible d'effectuer d'autres réglages à partir de cet écran, ils ne sont pas pour l'instant indispensables au bon fonctionnement de votre configuration et nous vous proposons donc de valider une fois pour toutes celui-ci en sélectionnant **Finish**.

Raspbian peut alors vraiment démarrer, ce qui se traduit par un affichage quelque peu laconique puisque ne sont visibles que les lignes suivantes :

```
My IP address is xxx.xxx.xxx.xx (si le Raspberry Pi est relié au
réseau)
Debian GNU/Linux wheezy/sid raspberry tty1
raspberrypi login:
```

Linux requiert en effet que vous saisissiez un nom d'utilisateur et son mot de passe associé afin que vous puissiez utiliser la machine. Dans le cas présent, cet utilisateur se nomme **pi** et le mot de passe est **raspberrypi**.

Saisissez donc **pi** en face de **login** et validez par **Entrée** ce qui a pour effet de faire afficher :

```
Password:
```

Saisissez alors **raspberrypi** puis validez à nouveau par **Entrée**.

■ Attention !

Le nom d'utilisateur tout comme le mot de passe sont sensibles à la casse. Veillez donc à bien faire attention à les saisir en minuscules dans le cas présent. Notez également que le mot de passe ne s'affiche pas pendant sa saisie pour d'évidentes raisons de sécurité.

Un message indiquant la date et l'heure de la dernière connexion de l'utilisateur s'affiche, suivi d'un avertissement de licence puis vient enfin le « prompt » ou invite de commande en bon français qui doit se présenter comme suit :

```
pi@raspberrypi ~ $
```

À partir de cet instant, il est possible d'utiliser la distribution Linux Raspbian fonctionnant sur le Raspberry Pi en mode ligne de commande mais, si ce n'est pas votre tasse de thé, nous vous proposons tout d'abord de découvrir l'interface graphique qui ne devrait pas trop vous dépayser même si vous n'avez connu que les environnements Windows.

Avant d'aller plus loin, abordons cependant tout de suite une question fréquemment posée par nombre d'utilisateurs débutants du Raspberry Pi : comment arrêter ce dernier « proprement » ? En effet, il ne vous viendrait pas à l'idée d'arrêter votre PC fonctionnant sous Windows en appuyant brutalement sur le bouton marche/arrêt, et il ne faut donc pas non plus procéder de la sorte avec le Raspberry Pi. Aussi, avant de lui couper le courant, est-il indispensable d'exécuter la ligne de commande suivante :

```
pi@raspberrypi ~ $ sudo shutdown -h now
```

Lorsque les différents messages d'arrêt ont été affichés et que seule la LED rouge de mise sous tension est allumée, vous pouvez couper le courant en toute sécurité.

2.2 LXDE, le bureau de Raspbian

À partir de l'invite de commande précédente, tapez `startx` ce qui a pour effet de lancer un programme appelé LXDE pour *Lightweight X11 Desktop Environment* que l'on peut traduire en abrégé par environnement de bureau X11 allégé. L'écran, beaucoup plus sympathique, visible figure 2.7, s'affiche alors après quelques secondes d'attente.

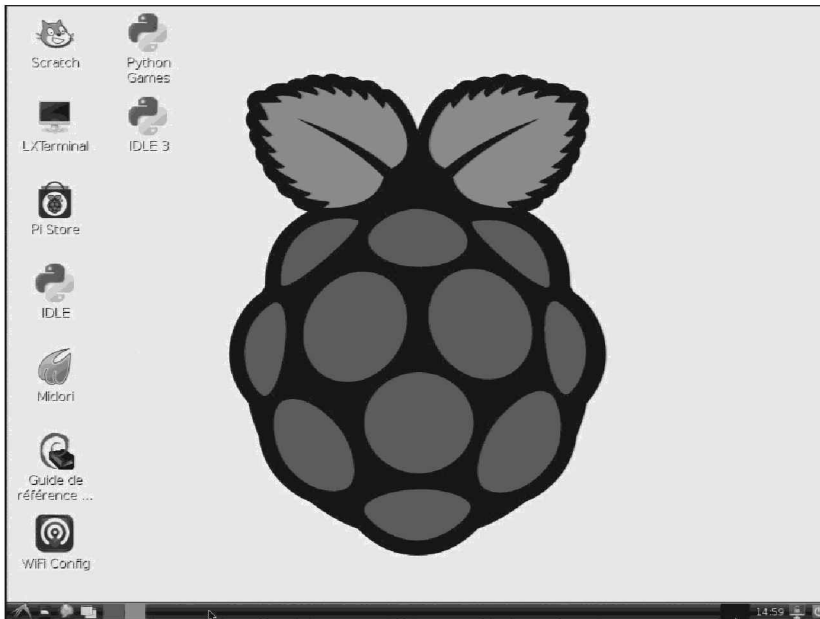


Figure 2.7 – Le bureau LXDE ne dépayse pas un habitué de Windows.

Ce bureau fonctionne bien évidemment comme n'importe quel bureau Windows et s'utilise de façon très classique avec la souris. Nous n'allons donc pas lui consacrer de longs développements mais seulement passer en revue les différents programmes proposés dont les noms, pour qui n'est pas habitué à Linux, sont parfois un peu déroutants.

Attention !

Si vous n'avez pas réalisé l'opération de localisation présentée au § 2.1.2, LXDE sera en anglais ainsi que tous les programmes associés.

Les programmes sont accessibles soit au moyen de leurs raccourcis, dont certains sont déjà présents sur le bureau, soit au moyen du menu auquel on accède grâce à l'icône située en bas et à gauche de l'écran, tout comme pour le menu démarrer de Windows. Ils sont classés en sept catégories dont nous allons détailler le contenu.

2.2.1 Le menu Accessoires

Ce menu donne accès aux huit programmes que voici :

- **Guide de référence pour Debian**, qui n'est pas à proprement parler un programme mais plutôt un manuel de référence de la distribution Linux Debian sur laquelle repose Raspbian, utilisée sur le Raspberry Pi. Si la syntaxe d'une commande Linux vous fait défaut, c'est donc ici que vous pourrez la trouver.
- **Ouvrir dans le répertoire de fichiers** est l'équivalent de l'explorateur de fichiers de Windows. Il permet de se déplacer dans les différents répertoires contenus sur la carte, de visualiser les fichiers qui y sont contenus, de les déplacer, les copier, les renommer, etc. Son utilisation est intuitive et ressemble fort à celle de l'explorateur de Windows. Si vous l'essayez, n'oubliez pas de faire un clic droit de la souris lorsque vous sélectionnez un dossier ou un fichier afin de faire ouvrir les menus contextuels.
- **Calculatrice** est, comme son nom l'indique (ou presque !), une calculatrice. Remarquez que c'est un modèle évolué car, pour peu que vous ouvriez ses différents menus vous découvrirez qu'elle supporte la notation polonaise inversée (comme les anciennes calculatrices programmables Helwett-Packard), l'entrée de formules, de nombreuses bases de numération dont le binaire et l'hexadécimal et que, si par défaut elle est en mode basique, elle constitue aussi une belle calculatrice scientifique comme indiqué figure 2.8.
- **Visionneur d'images** n'est autre... qu'une visionneuse d'images, assez simple mais permettant tout de même le zoom, la rotation et l'inversion haut-bas et gauche-droite de ces dernières.
- **Leafpad** est un éditeur de texte basique, qui fait indéniablement penser au bloc-notes de Windows. Il permet de saisir facilement des notes ou bien encore de modifier de courts fichiers textes.
- **LXTerminal** permet quant à lui d'ouvrir une fenêtre en mode ligne de commande et donc de saisir des commandes Linux sans quitter totalement l'environnement graphique.
- **Terminal administrateur** fonctionne comme **LXTerminal** mais place automatiquement l'utilisateur en mode **root**, c'est-à-dire en mode super-utilisateur en terminologie Linux. Il est alors possible d'exécuter des commandes qui ne sont normalement pas accessibles aux utilisateurs « classiques ».
- **Xarchiver** enfin est un programme de compression et de décompression de fichiers, au célèbre format zip par exemple.

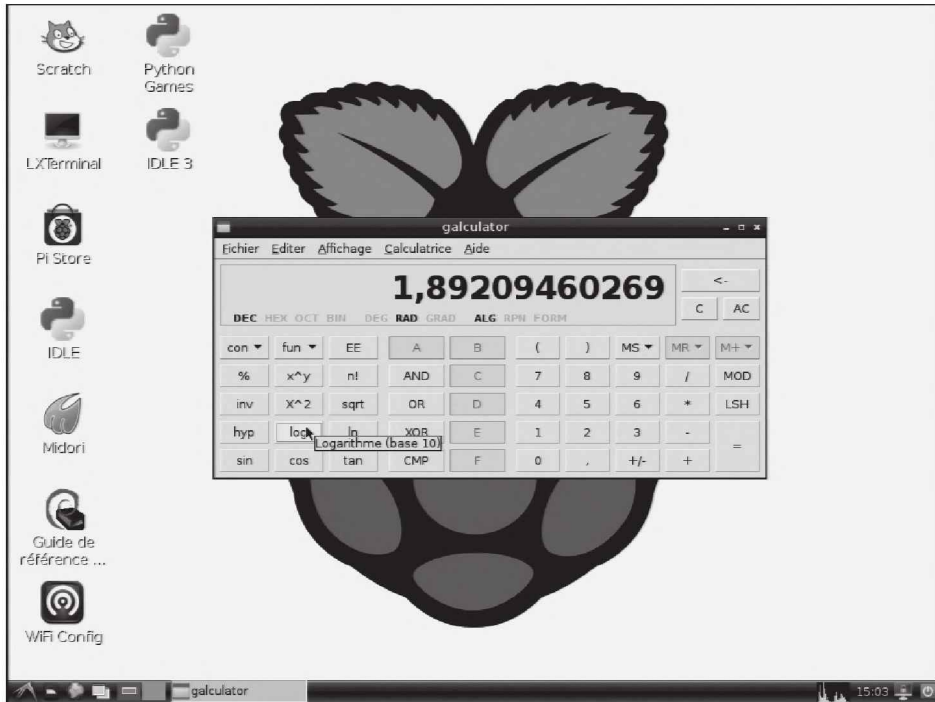


Figure 2.8 – La calculatrice n’a rien à envier à celle de Windows.

2.2.2 Le menu Éducation

Si l’on en croit les créateurs du Raspberry Pi, celui-ci a été conçu entre autres pour apprendre l’informatique aux plus jeunes et, à ce titre, la distribution standard conseillée avec le Raspberry Pi contient donc au moins un programme prévu à cet effet.

Il s’agit de **Scratch**, un langage de programmation graphique destiné à l’origine aux jeunes enfants mais que rien ne vous interdit d’expérimenter. La figure 2.9 montre, à titre d’exemple, son écran d’accueil sur lequel figurent des **sprites** que le programmeur en herbe peut animer et faire interagir au moyen des nombreux opérateurs de contrôle situés en partie gauche de cet écran.



Figure 2.9 – Scratch est un langage de programmation graphique destiné aux plus jeunes.

Le menu Éducation ne comporte donc que deux entrées :

- **Scratch** qui donne accès au programme que nous venons d'évoquer.
- **Squeak** qui donne accès à la plateforme sur laquelle fonctionne **Scratch** et que vous n'aurez en principe pas besoin d'utiliser.

2.2.3 Le menu Internet

La majorité des Raspberry Pi étant des modèles B, et donc disposant d'une interface réseau ; ils ont accès à Internet et il est donc normal qu'au moins un navigateur soit disponible au sein de l'environnement graphique LXDE.

En réalité, ce n'est pas un mais trois navigateurs qui sont proposés par LXDE et le menu Internet comporte donc... cinq entrées !

- **Dillo** qui est un projet issu du monde du logiciel libre. Ce navigateur n'en est cependant qu'à ses balbutiements et de nombreuses pages Internet, surtout lorsqu'elles sont très chargées en cadres, images et/ou vidéo, ont quelques difficultés à s'afficher correctement.
- **Midori** est un navigateur simple et rapide qui ne dépaysera pas un utilisateur d'Internet Explorer ou de Firefox même si, par défaut, nombre de modules d'extension auxquels ont fait souvent appel sur ces deux derniers sont absents. Son fonctionnement est satisfaisant sur la majorité des sites.

- **Midori Private Browsing** correspond aussi à **Midori**, mais lancé cette fois-ci en mode navigation privée, ce qui signifie que les sites visités ne sont pas mémorisés dans le cache du navigateur. Une option identique existe d'ailleurs sur la majorité des navigateurs plus connus.
- **Navigateur web NetSurf** est un autre navigateur, similaire à **Midori** mais pouvant parfois mieux se comporter sur certaines pages que **Midori** aurait du mal à afficher.
- **Wpa-gui** quant à lui n'est pas un navigateur Internet mais un programme de configuration de l'interface WiFi qu'il est possible d'utiliser avec le Raspberry Pi. Nous aurons l'occasion d'y revenir au chapitre suivant.

Quoi qu'il en soit, l'utilisation de l'un ou l'autre de ces navigateurs ne devrait pas vous poser de problème mais, si vous êtes un accro de Firefox, sachez que rien n'interdit de l'installer en complément de la distribution Raspbian de base.

2.2.4 Le menu Programmation

Ce menu est beaucoup plus original que le précédent et, puisque vous avez acheté ce livre, nous sommes prêts à parier qu'il vous intéressera beaucoup plus. Il donne en effet accès aux différents environnements de programmation proposés en standard par Raspbian. Il comporte pour cela les quatre entrées suivantes :

- **IDLE** donne accès à ce que les anglo-saxons appellent un IDE, pour *Integrated Development Environment*, ce qui signifie mot à mot « environnement de développement intégré ». Dans le cas présent, IDLE a été conçu pour vous permettre de programmer dans un langage qui s'appelle **Python** ; langage que nous vous proposerons de découvrir au chapitre 5 et avec lequel nous rédigerons la majorité des programmes d'interfaçage avec le monde extérieur présentés à partir du chapitre 7.
- **IDLE 3**, vu son nom, donne accès également à IDLE mais, alors que IDLE est adapté à la version 2.7 de **Python**, IDLE 3 est destiné à la version 3 de ce même langage.
- **Scratch**, que nous avons déjà vu dans le menu Éducation, apparaît à nouveau ici puisque, stricto sensu, c'est également un langage de programmation.
- **Squeak** apparaît également pour la même raison que **Scratch**.

2.2.5 Le menu Graphismes

Même si le Raspberry Pi n'est pas destiné a priori à constituer une machine bureautique, encore que cela soit possible comme nous le verrons au chapitre 4, Raspbian propose d'origine un logiciel permettant au moins de lire les fichiers pdf :

Xpdf permet, comme son nom le laisse supposer, de lire les fichiers à ce format sans devoir recourir aux logiciels d'Adobe.

2.2.6 Le menu Outils système

Ce menu ne contient lui aussi qu'une seule entrée :

Gestionnaire des tâches qui, à l'instar de celui dont vous avez très certainement l'habitude sous Windows, permet de mettre fin à l'exécution de programmes qui se seraient « plantés » ainsi que d'avoir une idée de la charge du processeur, globale ou tâche par tâche, de la quantité de mémoire disponible, etc.

2.2.7 Le menu Préférences

Comme son nom le laisse supposer, le menu préférence donne accès à un certain nombre de fonctions de configuration de LXDE et comporte pour ce faire six entrées :

- **Personnaliser l'apparence**, permet de définir ou de modifier l'apparence globale de l'environnement en modifiant par exemple l'aspect ou la couleur des fenêtres. L'onglet Widget donne ainsi accès à plusieurs environnements graphiques qui sont en outre éditables via cette fonction.
- **Paramètres de la session** permet de définir le comportement du système lors du démarrage du bureau et de choisir quel(s) programme(s) exécuter. À priori vous n'aurez pas à modifier ces informations sauf besoins très particuliers.
- **Clavier et souris** permet, comme son nom le laisse supposer, de modifier le comportement du clavier et de la souris pour ce qui est de la vitesse de répétition des touches, de celle de déplacement de la souris ou bien encore de la sensibilité de cette dernière.
- **Paramètres de l'écran** permet théoriquement de modifier la résolution du moniteur connecté au Raspberry Pi. Son utilisation reste cependant limitée et il est souvent nécessaire pour cela d'avoir recours aux fichiers de configuration comme nous le verrons par la suite.
- **Openbox Configuration Manager** permet de modifier le thème de l'environnement graphique utilisé par LXDE ; environnement graphique qui s'appelle justement Open Box, d'où le nom de cette entrée du menu.
- **Applications favorites** permet de définir ce que l'on appelle en « langage Windows » les associations de fichiers. En d'autres termes, c'est ici que vous définirez quelle application ouvre tel ou tel type de fichier. Dans un premier temps, nous vous recommandons de ne pas toucher aux définitions réalisées par défaut.

Ce tour d'horizon rapide du bureau LXDE n'avait bien sûr pas pour but de décrire en détail les fonctions de toutes les applications fournies d'origine avec la distribution Raspbian Wheezy mais de vous permettre de les découvrir car leurs noms, propres à l'environnement Linux, sont parfois assez « hermétiques » pour un habitué du mode Windows.

Nous aurons par contre l'occasion de revenir sur certaines d'entre elles dans la suite de cet ouvrage, notamment pour ce qui est de IDLE et de IDLE 3 lorsque nous découvrirons la programmation en Python.

Pour les applications plus classiques, telles que les navigateurs Internet, le bloc-notes ou le visionneur d'images par exemple, nous vous conseillons de les utiliser afin de découvrir leurs possibilités. Quoique vous fassiez comme « bêtise » en les utilisant, vous ne risquez pas de détériorer le système contenu sur la carte SD et si, par extraordinaire, c'était le cas, il ne vous faudrait que quelques minutes pour la recréer en suivant les indications du chapitre 1.7.2.

2.3 Le mode ligne de commande

Même si l'on peut théoriquement utiliser Linux presque exclusivement au travers de l'un ou l'autre de ses bureaux graphiques, cela n'est vrai que pour les distributions destinées aux PC et encore, si on ne cherche pas à les reconfigurer trop finement.

Dans le cas du Raspberry Pi, et comme nous allons le voir tout au long de cet ouvrage, un certain nombre d'opérations ne peuvent être réalisées qu'au moyen d'instructions, fonctions ou programmes exécutés en mode ligne de commande.

Rassurez-vous, il n'est cependant pas nécessaire d'être un pro du monde Linux pour cela car nous vous proposons de découvrir, dans les quelques lignes qui suivent, les commandes de base dont vous aurez besoin pour ce faire.

2.3.1 Accès au mode ligne de commande

Par défaut, et si vous n'avez pas modifié cette option au niveau du premier écran de configuration vu figure 2.1, la distribution Raspbian démarre en mode ligne de commande que l'on appelle aussi mode console. C'est d'ailleurs dans ce mode que nous vous avons fait saisir vos login et mot de passe ci-dessus.

Si par contre vous avez lancé LXDE comme nous vous avons proposé de le faire, vous pouvez également accéder à ce mode, sans toutefois quitter l'environnement graphique, en ouvrant une console grâce à l'application **LXTerminal** du menu **Accessoires**.

Dans ce dernier cas, vous disposerez alors de fonctions d'édition des commandes saisies, très utiles lorsque l'on débute et que l'on commet quelques fautes de frappe sur les commandes Linux ; possibilités d'édition qui vous sont évidemment inaccessibles si vous êtes en mode ligne de commande « brute ».

Nous vous conseillons donc, au moins pour faire vos premières armes avec les commandes Linux, de travailler au travers de l'application **LXTerminal** comme nous le ferons dans les exemples présentés ci-après et pour les recopies d'écran qui les accompagneront.

Précisons à ce propos que, afin que ce livre ne se transforme pas en album photo, nous ne réaliserons pas des recopies d'écran systématiques de toutes les commandes que nous présenterons. En revanche, pour que vous n'ayez aucun doute quant à ce que vous devrez saisir, nous reproduirons sous forme texte ce qui sera affiché à l'écran de votre Raspberry Pi.

Compte tenu du mode d'affichage des lignes de commande Linux, le « prompt » ou invite de commande en bon français, est systématiquement affiché au début de chaque ligne. Nous le reproduirons donc dans nos exemples mais il ne devra pas être saisi par vos soins. Ainsi par exemple, pour faire exécuter la commande `ls`, nous écrirons :

```
pi@raspberrypi: ~ $ ls
```

mais vous n'aurez évidemment qu'à saisir `ls`.

Et puisque nous y sommes, expliquons rapidement la signification de ce « prompt » propre à Linux.

Le nom qui précède l'arobase (@) est le nom de l'utilisateur, `pi` en l'occurrence, tandis que celui qui suit est le nom de l'ordinateur, `raspberrypi` dans le cas présent. Le caractère « tilda » (~) qui suit indique que nous nous trouvons dans le répertoire de travail, qui est par défaut le répertoire propre à l'utilisateur (`/home/pi` dans ce cas). Enfin, le dollar qui ferme la marche est le « prompt » proprement dit, c'est-à-dire le caractère après lequel apparaîtront les commandes saisies par l'utilisateur.

2.3.2 Super-utilisateur, root et sudo

Plus encore que sous Windows, où existe l'utilisateur nommé administrateur qui seul a pouvoir de tout faire sur une machine, les distributions Linux se protègent des « bêtises » que peuvent faire les utilisateurs en étant très strictes au niveau de la gestion de ces derniers et des privilèges dont ils disposent.

Le seul utilisateur qui peut « tout » faire sous Linux est le super-utilisateur ou `root` mais, afin que vous ne soyez pas tenté de vous connecter systématiquement en tant que `root`, surtout si vous êtes tout seul à utiliser la machine, certaines distributions ne vous le permettent pas, ce qui est le cas de Raspbian. Fort heureusement, il existe une solution de contournement de cette restriction car, si tel n'était pas le cas, nombre d'opérations de configuration ne vous seraient pas accessibles.

Par défaut, lorsque vous vous connectez au Raspberry Pi pour la première fois, votre login, ou nom d'utilisateur, est `pi` et le mot de passe associé est `raspberrypi`, comme nous l'avons vu précédemment, et vous disposez des mêmes prérogatives que celles de n'importe quel utilisateur lambda, c'est-à-dire ... rien !

Cela vous permet tout de même d'exécuter un certain nombre de commandes, que nous allons découvrir dans un instant, mais pas celles réservées à l'administration du système. Vous risquez fort alors d'être frustré lorsque vous constaterez que vous ne pourrez même pas régler la date machine de ce Raspberry Pi que vous avez pourtant payé de vos deniers ! Pour vous en convaincre, essayez donc de saisir la commande suivante :

```
pi@raspberrypi: ~ $ date 02271045
```

qui est censée régler la date machine au 27 février de l'année en cours et à 10 heures 45.

Vous recevrez en retour le message :

```
cannot set date : Operation not permitted
```

Pour passer outre cette restriction, ainsi que celles propres à toutes les commandes réservées au super-utilisateur, vous disposez de la commande `sudo` qui n'est autre que le résultat de l'abréviation de *SUper user DO*.

Si vous faites précéder une commande réservée au super-utilisateur par `sudo`, vous pourrez alors l'exécuter comme si vous vous étiez connecté en tant que super-utilisateur ou `root`. Pourquoi alors nous direz-vous avoir protégé l'accès au super-utilisateur si la simple saisie de `sudo` permet de passer outre ? Tout simplement parce que l'on estime que la contrainte imposée par cette saisie oblige l'utilisateur à bien réfléchir à ce qu'il fait, et donc minimise le risque de saisie d'une commande erronée par simple étourderie ou manque de réflexion.

Pour régler l'heure de votre système, vous saisissez donc la commande suivante :

```
pi@raspberrypi: ~ $ sudo date 02271045
```

qui sera immédiatement suivie d'effet, ce qui sera concrétisé par l'affichage des nouvelles date et heure.

Attention !

Selon l'enchaînement d'opérations qui auront eu lieu entre le démarrage de votre Raspberry Pi et la première utilisation de `sudo`, il se peut que votre login et votre mot de passe vous soient demandés. C'est normal et cela a pour but d'éviter qu'une personne qui ne serait pas un utilisateur autorisé de la machine puisse utiliser `sudo`.

2.3.3 Utilisateurs et groupes

Chaque utilisateur potentiel d'une machine Linux est identifié par son nom comme nous venons de le voir, mais ce n'est pas tout. En effet, chaque utilisateur peut également appartenir à un ou plusieurs groupes, ce qui lui confère des privilèges. Comme nous le verrons ci-dessous, les fichiers sont protégés au moyen de droits d'accès en lecture, écriture et exécution ; droits qui sont définis en fonction du propriétaire du fichier concerné bien sûr mais aussi en fonction du groupe auquel il appartient.

Deux commandes principales permettent donc fort logiquement de gérer les utilisateurs et les groupes.

Ainsi, pour savoir à quel(s) groupe(s) tel ou tel utilisateur appartient, il suffit de saisir :

```
groups nom d'utilisateur
```

Dans le cas de l'utilisateur standard `pi`, cela donne le résultat suivant :

```
pi@raspberrypi ~ $ groups pi
pi : pi adm dialout cdrom sudo audio video plugdev games users
netdev input
```

Pour créer un nouvel utilisateur, il faut évidemment être super-utilisateur du système et donc, dans notre cas, faire usage de la commande `sudo`. On procède alors avec la commande `useradd` prévue à cet effet de la façon suivante :

```
sudo useradd -m -G liste des groupes d'appartenance
```

L'option `-m` permet de créer un répertoire propre à ce nouvel utilisateur afin qu'il puisse y stocker ses fichiers, tandis que l'option `-G` permet de préciser les groupes auxquels cet utilisateur appartiendra.

Si, par exemple, vous voulez créer un deuxième utilisateur du Raspberry Pi, disposant des mêmes droits que vous-même ; il faudra le faire appartenir aux mêmes groupes que l'utilisateur `pi`. Ayant découvert ces droits grâce à la commande `groups` dans l'exemple ci-dessus, il suffira donc de saisir :

```
sudo useradd -m -G adm,dialout,cdrom,sudo,audio,video,plugdev,games,users,netdev,input
```

2.3.4 Organisation du système de fichiers

L'organisation générale des fichiers sous Linux ne diffère pas fondamentalement de ce à quoi vous êtes sans doute habitué sous Windows, mais quelques particularités méritent tout de même un peu d'attention.

Linux gère en effet toutes les mémoires de masse de la machine comme des répertoires. Ainsi, il n'existe pas comme sous Windows, un ou des disques avec des lettres pour les identifier, mais uniquement des répertoires contenus dans une arborescence globale. De ce fait, lorsque l'on connecte une mémoire de masse à la machine, on doit la « monter » – c'est le terme consacré – ce qui a pour effet de l'inclure dans cette arborescence.

Si vous travaillez avec l'environnement graphique LXDE vu précédemment, cette opération a lieu de manière automatique. Pour vous en convaincre, vous pouvez par exemple raccorder une clé USB au Raspberry Pi pour constater qu'elle devient visible dans l'explorateur de fichiers. Si vous êtes en mode console ou ligne de commande, il faut monter cette mémoire de masse « à la main » comme nous le verrons dans la suite de cet ouvrage.

Ceci étant précisé, les noms de répertoires sous Linux sont toujours précédés du caractère slash (/) étant entendu que le sommet de l'arborescence (le C : de Windows si l'on veut à tout prix faire un parallèle avec ce système d'exploitation) est repéré par un slash « tout seul ».

Pour descendre ensuite dans l'arborescence, on enchaîne tout simplement les noms des répertoires les uns après les autres en les séparant à chaque fois par un slash. Le répertoire `pi` contenu dans `home` est ainsi repéré par :

```
/home/pi
```

Par ailleurs, le répertoire courant, c'est-à-dire celui dans lequel on se trouve à un instant donné est repéré par un point (.) tandis que le répertoire de niveau immédiatement supérieur, ou répertoire parent, est repéré par deux points successifs (..).

Ceci étant vu, tout système Linux qui se respecte repose sur une arborescence dans laquelle un certain nombre de répertoires ont des noms et des fonctions bien précises. Même s'il n'est pas utile de les connaître tous, nous vous proposons de découvrir rapidement les principaux ci-dessous afin que vous sachiez ensuite plus facilement où chercher tel ou tel fichier en cas de besoin.

bin contient tous les exécutables nécessaires au fonctionnement de l'OS ainsi qu'à l'environnement graphique.

boot contient le noyau Linux et tout ce qui est nécessaire pour permettre le démarrage du Raspberry Pi.

dev est un répertoire virtuel qui n'existe donc pas en tant que tel sur la carte mémoire. C'est là que se trouvent placés tous les périphériques du système (dev est l'acronyme de *device*).

etc contient un certain nombre de fichiers de configuration ainsi que la liste des utilisateurs et leurs mots de passe, sous forme cryptée bien sûr.

home est un répertoire dans lequel chaque utilisateur dispose d'un sous-répertoire à son nom pour stocker ses fichiers personnels, par exemple /home/pi pour l'utilisateur pi.

lib est un répertoire de bibliothèques utilisées par différents programmes et applications.

lost+found stocke les fragments de fichiers en cas de plantage majeur du système.

media est un répertoire spécial dédié aux périphériques amovibles tels que clés USB ou bien encore lecteurs externes.

mnt n'est utilisé que pour « monter » les unités de stockage externes raccordées au système comme nous l'avons expliqué ci-dessus.

opt sert à stocker des logiciels optionnels ne faisant pas initialement partie de la distribution de base.

proc est un répertoire virtuel contenant des informations sur les programmes en cours d'exécution.

sbin stocke des fichiers réservés au super-utilisateur pour des opérations de maintenance.

sys contient des fichiers propres au système d'exploitation.

tmp stocke les fichiers temporaires de tous types en fonction des besoins.

usr permet aux programmes utilisateur de stocker des données.

var est un répertoire virtuel utilisé par les programmes pour stocker les valeurs de variables.

La figure 2.10, réalisée au moyen de la commande **ls** que nous découvrirons dans un instant, confirme bien la présence de tous ces répertoires suite au démarrage de notre Raspberry Pi.

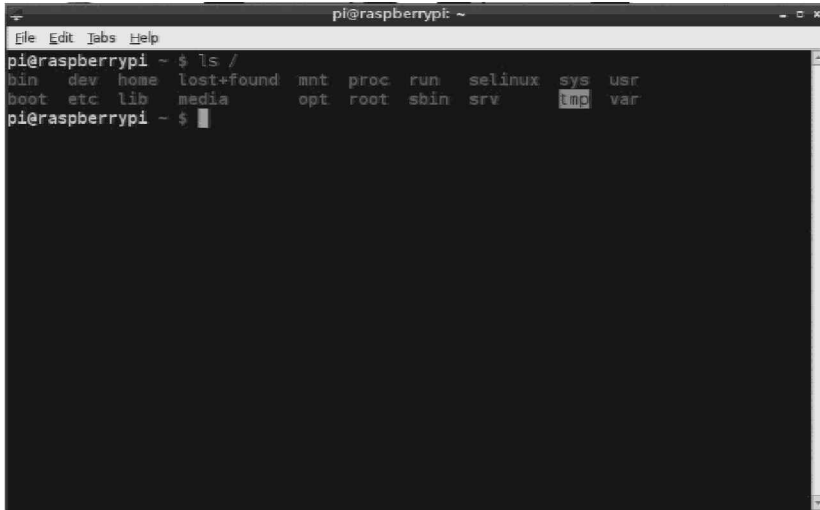


Figure 2.10 – Les répertoires présents sur la carte SD du Raspberry Pi.

2.3.5 Navigation dans les fichiers et les répertoires

Même avec les quelques explications qui précèdent, on est bien souvent perdu dans l'organisation de ces fichiers et répertoires lorsque l'on découvre un système d'exploitation, et les premières commandes auxquelles on fait appel sont alors celles qui permettent de naviguer parmi ces derniers.

Voici donc une liste, non exhaustive mais largement suffisante dans un premier temps, pour se déplacer au sein de l'arborescence de Raspbian et pour réaliser de la manipulation élémentaire de fichiers et de répertoires.

pwd est certainement la commande que vous utiliserez le plus souvent à vos débuts. C'est l'acronyme de *Print Working Directory* ce qui veut dire mot à mot « afficher le répertoire de travail ». Elle permet donc de savoir « où l'on se trouve » au sein de l'arborescence. Si vous l'utilisez après avoir lancé LXTerminal vous obtiendrez :

```
pi@raspberrypi: ~ $ pwd
/home/pi
```

cd est très semblable à son homologue sous DOS et permet de changer de répertoire puisque **cd** est l'acronyme de *Change Directory*. Elle s'utilise en la faisant suivre du nom du répertoire à atteindre pour y accéder directement, ou bien suivie de deux points successifs pour remonter d'un niveau dans l'arborescence puisque c'est au moyen de deux points que l'on désigne le répertoire parent comme nous l'avons vu au § 2.3.4. Ainsi, en partant de l'exemple précédent on obtient :

```
pi@raspberrypi: ~ $ cd ..
pi@raspberrypi: /home $
```

On est en effet remonté d'un cran dans l'arborescence et, comme on se trouvait dans `/home/pi` nous sommes maintenant dans `/home`.

ls permet quant à elle de lister le contenu d'un répertoire afin de savoir quels sont les fichiers et les sous-répertoires qui y sont contenus. Elle peut s'utiliser seule, auquel cas elle travaille sur le répertoire courant, ou suivie du nom d'un répertoire, auquel cas elle liste le contenu de celui-ci, quel que soit le répertoire courant. Elle dispose aussi de l'option très utile `-l` qui permet d'obtenir des informations détaillées sur le contenu du répertoire sur lesquelles nous reviendrons dans la suite de cet exposé. Utilisée seule dans le répertoire de `/home/pi`, elle fait ainsi afficher :

```
pi@raspberrypi ~ $ ls
Desktop      Documents    indiecity    python_games  Scratch
```

mv est l'acronyme de *move* et permet, d'une part de déplacer les fichiers d'un répertoire à un autre, mais aussi de renommer un fichier. En effet, pour Linux, renommer un fichier revient à le déplacer d'un nom à un autre.

rm, qui est l'acronyme de *remove*, permet comme son nom l'indique, d'enlever un fichier, c'est-à-dire de l'effacer. Tout comme la commande `del` du DOS, elle peut agir sur le seul fichier dont le nom doit alors suivre la commande, ou sur plusieurs fichiers pour peu que l'on utilise les caractères « jokers » habituels. Cette commande est donc à utiliser avec précaution.

rmdir est l'équivalent de **rm** mais pour effacer un répertoire complet. Par mesure de sécurité, le répertoire à effacer doit être vide avant de pouvoir être effacé au moyen de cette commande.

mkdir est en quelque sorte le pendant de **rmdir** puisqu'elle permet de créer un répertoire. Si la commande est suivie d'un simple nom de répertoire, celui-ci est créé dans le répertoire courant, mais il est aussi possible de créer un répertoire absolu en le désignant de façon complète depuis le répertoire racine.

2.3.6 Droits associés aux fichiers

Afin de sécuriser au maximum les manipulations sur les fichiers et les répertoires, Linux leur associe des droits en lecture, écriture ou encore exécution ; droits qui permettent une gestion très fine des autorisations données à tel ou tel utilisateur. Pour prendre connaissance de ces droits, il suffit d'utiliser la commande **ls**, vue ci-dessus, assortie de l'option `-l`. Voici par exemple ce que l'on obtient en saisissant cette commande depuis le répertoire `/home/pi/Documents` :

```
drwxr-xr-x 2 pi pi 4096 Feb 20 10:59 Scratch Projects
```

Le premier caractère de cette ligne indique de quel type d'objet (répertoire ou fichier) nous sommes en présence au moyen de la codification suivante :

- Un **d** indique un répertoire (d comme directory),
- Un **l** indique un lien,
- Un **-** (tiret) indique un fichier.

Les neuf caractères qui suivent précisent les droits associés à cet objet pour le propriétaire de l'objet, pour les membres d'un groupe d'utilisateurs spécifié, et enfin pour le « reste du monde », au moyen de la codification suivante :

- Un **r** signifie un droit d'accès en lecture (r comme read).
- Un **w** signifie un droit d'accès en écriture (w comme write).
- Un **x** signifie un droit d'exécution.
- Un **-** signifie l'absence de tout droit d'accès.

Ces neuf lettres sont suivies par le nom du propriétaire du fichier puis par celui de son groupe d'appartenance.

Dans le cas de notre exemple, nous constatons donc que l'utilisateur appelé **pi** (c'est-à-dire vous-même puisque c'est sous ce nom que vous vous êtes connecté au Raspberry Pi) possède les droits de lecture, d'écriture et d'exécution (premier triplet **rw****x**), que les membres du groupe d'utilisateurs baptisé **pi** possèdent seulement les droits de lecture et d'exécution (second triplet **r-x**) et enfin que tous les autres utilisateurs possèdent les mêmes droits (dernier triplet **r-x**).

Si nécessaire, ces droits peuvent évidemment être modifiés au moyen d'une commande appropriée qui s'appelle **chmod** pour laquelle plusieurs syntaxes peuvent être utilisées.

La syntaxe à notre avis la plus simple à utiliser est la suivante :

chmod catégorie+opération+droits fichier

catégorie correspond aux types d'utilisateurs sur lesquels on souhaite agir sachant que :

- Un **u** représente le propriétaire du fichier (u comme user).
- Un **g** représente le groupe ayant accès au fichier (g comme groupe).
- Un **a** représente « les autres ».

opération correspond à l'opération que l'on souhaite réaliser sur les droits à modifier avec la codification suivante :

- Un **+** ajoute le droit.
- Un **-** retire le droit.
- Un **=** définit le droit.

Le terme suivant enfin précise le droit que l'on souhaite modifier au moyen de l'une des trois lettres que vous connaissez déjà, à savoir **r** pour lecture, **w** pour écriture et **x** pour exécution.

Ainsi par exemple :

```
chmod a-rwx toto.txt
```

retire tous les droits (à cause de la présence de **r**, **w** et **x** précédés du signe **-**) à tous les utilisateurs autres que le propriétaire du fichier et les membres du groupe (à cause du **a**).

```
chmod u+w toto.txt
```

donne le droit en écriture (à cause du `w` et du signe `+`) au propriétaire du fichier (à cause du `u`).

L'autre syntaxe de `chmod` est purement numérique. Elle peut sembler plus difficile d'abord mais permet en contrepartie, et en une seule commande, de définir tous les droits pour tous les utilisateurs. Elle code pour cela les triplets définissant les droits somme indiqué par le tableau 2.1.

Tableau 2.1 – Codage des droits dans la syntaxe numérique de la commande `chmod`.

Triplet	Binaire	Décimal
---	000	0
--x	001	1
-w-	010	2
-wx	011	3
r--	100	4
r-x	101	5
rw-	110	6
rwX	111	7

Ceci étant précisé, la commande devient très simple d'utilisation puisque, par exemple :

```
chmod 751 toto.txt
```

Équivaut à donner au fichier `toto.txt` les droits suivant :

```
rwxr-x--x
```

2.3.7 Une aide toujours présente

Les commandes Linux sont évidemment beaucoup plus nombreuses que le court extrait que nous venons de découvrir ci-dessus pour la seule gestion des fichiers, répertoires et utilisateurs et, même pour ce qui est des commandes que nous vous avons présentées, de nombreuses options que nous n'avons pas nécessairement évoquées sont bien souvent disponibles.

Pour les découvrir, vous pouvez bien sûr vous aider d'un ouvrage de formation à Linux, ou bien encore consulter un des très nombreux sites Internet consacrés à ce système d'exploitation mais sachez que, pour un usage courant, vous disposez

en permanence et à portée de main d'une aide relative à la syntaxe de toutes les commandes et de leurs options respectives.

Pour y accéder, il vous suffit tout simplement de saisir `man` (comme l'abréviation de manuel) devant le nom d'une commande.

La figure 2.11 montre ainsi à titre d'exemple le résultat de l'exécution de la commande `man` avec la commande `ls`.

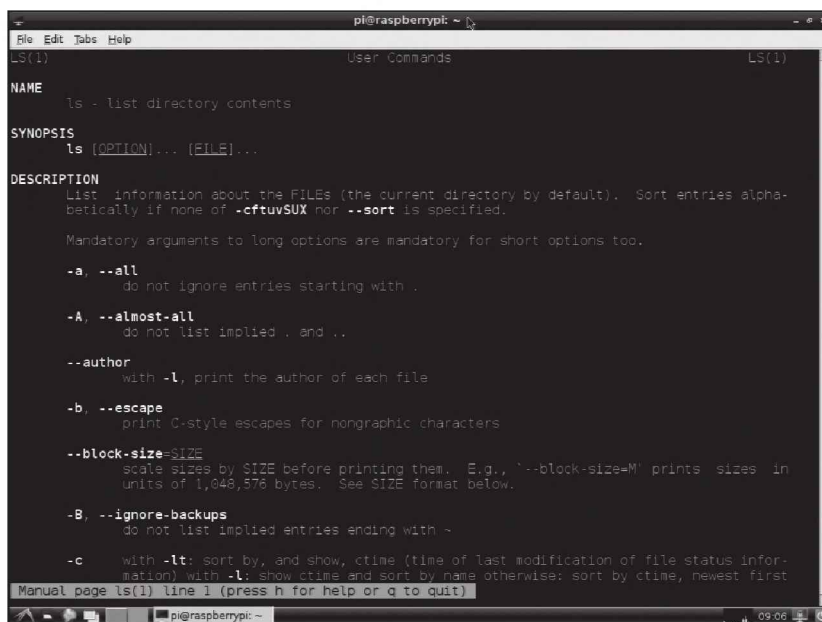
A screenshot of a terminal window on a Raspberry Pi. The window title is 'pi@raspberrypi: ~'. The terminal shows the output of the 'man ls' command. The output is formatted with sections: NAME, SYNOPSIS, DESCRIPTION, and a list of options. The DESCRIPTION section explains that 'ls' lists information about files, sorted alphabetically by default. The options listed include -a, -A, --author, -b, --block-size=SIZE, -B, --ignore-backups, and -c. At the bottom of the terminal, a status bar indicates 'Manual page ls(1) line 1 (press h for help or q to quit)'. The terminal window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'.

Figure 2.11 – L'aide offerte par la commande `man` dans le cas de la commande `ls`.

Si la description de la commande est trop longue pour tenir sur un seul écran, comme c'est le cas figure 2.11, il suffit d'utiliser les touches fléchées du clavier pour faire défiler cet écran vers le bas (ou vers le haut si vous désirez revenir en arrière). Une fois l'aide ainsi consultée, la frappe de `q`, pour `quit`, permet de revenir à la saisie de la commande désirée. C'est évidemment très pratique, d'autant que cette aide est généralement bien rédigée.

2.4 Ajout et suppression de programmes

Même si la distribution Raspbian Wheezy est relativement bien fournie en programmes et applications, vous souhaitez tôt ou tard en ajouter afin de faire accomplir à votre Raspberry Pi des tâches initialement non prévues. Si cet ajout passe parfois par le remplacement total de l'image système de Raspbian Wheezy

contenue sur la carte SD par une autre, comme c'est le cas par exemple pour transformer le Raspberry Pi en *media center*, comme nous le verrons au chapitre 4, des moyens aussi expéditifs ne sont pas toujours nécessaires et une simple installation de programme suffit parfois.

Cette installation peut être réalisée à partir de l'interface graphique mais, une fois n'est pas coutume, elle est beaucoup plus facile et surtout plus rapide à mettre en œuvre à partir de la ligne de commande grâce à une commande aux multiples facettes baptisée `apt`.

2.4.1 À la recherche des logiciels

Les logiciels disponibles sont contenus dans ce que Linux appelle des *packages*, que l'on peut traduire en français par paquets, et un logiciel d'installation et de désinstallation de ces packages est donc ce que l'on appelle un *package manager*. Outre ces fonctions d'installation et de désinstallation, il assure aussi la mise à jour des programmes lorsque cela s'avère nécessaire.

La liste des *packages*, et donc des programmes que l'on peut installer sur la machine est contenue dans une base de données appelée le cache dans le cas présent ; cache dont le contenu peut être mis à jour depuis des serveurs accessibles par Internet que l'on appelle des *repositories*.

La commande `apt` permet donc tout naturellement de gérer ce cache puisque c'est là qu'elle va devoir aller chercher les informations sur les programmes à installer. Pour ce faire, elle peut rechercher n'importe quel paquet précis dans ce cache ou, si vous n'en connaissez pas le nom exact, ce qui est souvent le cas, rechercher des paquets par thème.

Mais avant cela, il faut bien entendu s'assurer que les informations contenues dans ce cache soient à jour et pour cela, il convient donc de saisir la commande suivante :

```
pi@raspberrypi ~ $ sudo apt-get update
```

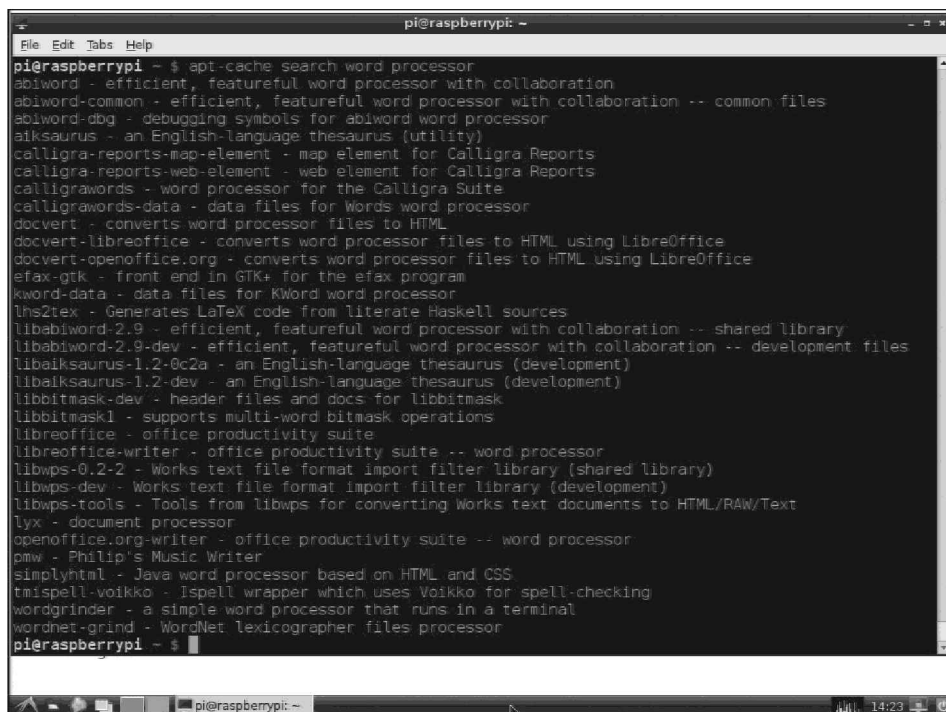
Notez que la commande `apt-get` doit être précédée de `sudo` car, comme elle a aussi pour fonction d'installer des programmes, elle n'est en principe accessible qu'au super-utilisateur.

Une fois le cache mis à jour et si, par exemple, nous désirons rechercher tous les packages contenant un logiciel de traitement de texte, il suffira de saisir :

```
pi@raspberrypi ~ $ apt-cache search word processor
```

La syntaxe de cette ligne est facile à analyser puisque `apt-cache` demande à la commande `apt` d'agir sur le contenu du cache et la fonction `search` lui enjoint de chercher tous les packages contenant les mots-clés qui suivent, `word processor` dans ce cas.

En fonction du mot-clé utilisé, les réponses peuvent être plus ou moins nombreuses comme le confirme la figure 2.12 qui présente l'écran faisant suite à la commande précédente.



```
pi@raspberrypi ~  
File Edit Tabs Help  
pi@raspberrypi ~$ apt-cache search word processor  
abiword - efficient, featureful word processor with collaboration  
abiword-common - efficient, featureful word processor with collaboration -- common files  
abiword-dbg - debugging symbols for abiword word processor  
aiksaurus - an English-language thesaurus (utility)  
calligra-reports-map-element - map element for Calligra Reports  
calligra-reports-web-element - web element for Calligra Reports  
calligrawords - word processor for the Calligra Suite  
calligrawords-data - data files for Words word processor  
docvert - converts word processor files to HTML  
docvert-libreoffice - converts word processor files to HTML using LibreOffice  
docvert-openoffice.org - converts word processor files to HTML using LibreOffice  
efax-gtk - front end in GTK+ for the efax program  
kword-data - data files for KWord word processor  
lhs2tex - Generates LaTeX code from literate Haskell sources  
libabiword-2.9 - efficient, featureful word processor with collaboration -- shared library  
libabiword-2.9-dev - efficient, featureful word processor with collaboration -- development files  
libaiksaurus-1.2-0c2a - an English-language thesaurus (development)  
libaiksaurus-1.2-dev - an English-language thesaurus (development)  
libbitmask-dev - header files and docs for libbitmask  
libbitmask1 - supports multi-word bitmask operations  
libreoffice - office productivity suite  
libreoffice-writer - office productivity suite -- word processor  
libwps-0.2-2 - Works text file format import filter library (shared library)  
libwps-dev - Works text file format import filter library (development)  
libwps-tools - Tools from libwps for converting Works text documents to HTML/RAW/Text  
lyx - document processor  
openoffice.org-writer - office productivity suite -- word processor  
pmw - Philip's Music Writer  
simplyhtml - Java word processor based on HTML and CSS  
tmspell-voikko - Ispell wrapper which uses Voikko for spell-checking  
wordgrinder - a simple word processor that runs in a terminal  
wordnet-grind - WordNet lexicographer files processor  
pi@raspberrypi ~$
```

Figure 2.12 – Les réponses à apt-cache search peuvent être très nombreuses.

Attention !

La recherche dans le cache ne comprend malheureusement que la langue anglaise. Ainsi la recherche précédente n'aurait ramené aucun résultat avec les mots clés traitement de texte.

2.4.2 Installation d'un programme

Une fois le ou les programmes de votre choix identifiés dans le cache, leur installation avec la commande `apt-get` est particulièrement simple. Supposons par exemple qu'à partir de la recherche précédente on souhaite installer le traitement de texte `abiword`. Il suffira tout simplement de saisir :

```
pi@raspberrypi ~$ sudo apt-get install abiword
```

Après un délai variable, dépendant de la vitesse de votre connexion Internet et correspondant essentiellement au temps de téléchargement des différents paquets constituant le programme, et après avoir accepté l'installation en répondant par o pour oui à la question posée, le logiciel sera installé sans nécessiter d'autre intervention manuelle de votre part.

Pour le vérifier il vous suffira, dans le cas de cet exemple, de quitter `LXTerminal` pour revenir au bureau afin de constater qu'une nouvelle rubrique, baptisée `Bureau-tique`, a fait son apparition, et qu'elle contient bien une entrée appelée `AbiWord` donnant accès au traitement de texte éponyme comme indiqué figure 2.13.

Attention !

Rappelons à nouveau que l'installation de ce logiciel n'aura lieu en français que si vous avez procédé à la localisation de votre distribution comme nous l'avons vu au § 2.1.2

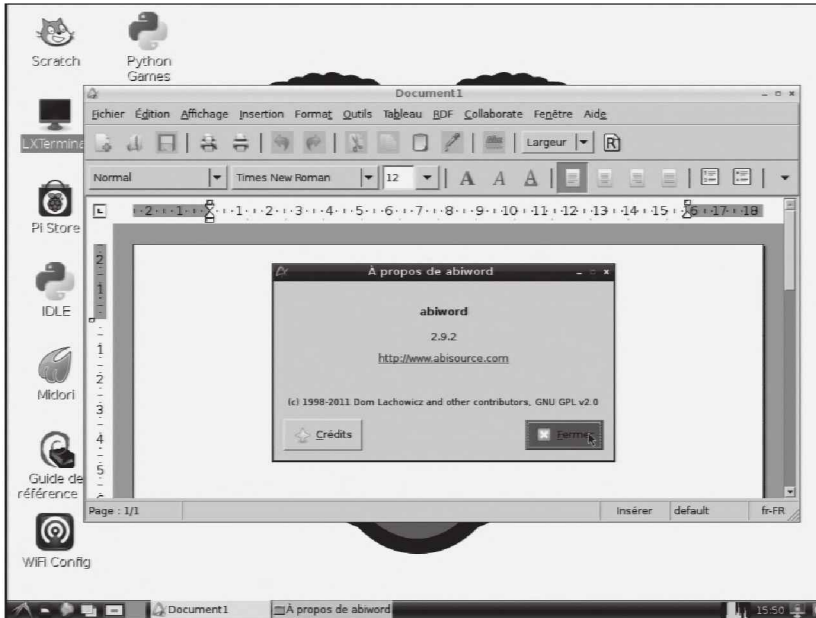


Figure 2.13 – Le traitement de texte Abiword a bien été installé comme prévu.

2.4.3 Désinstallation d'un programme

La désinstallation d'un programme est tout aussi facile à réaliser avec la commande `apt-get` mais propose deux variantes selon que vous souhaitez ou non conserver les paramètres de configuration du programme à désinstaller, en vue d'une nouvelle installation ultérieure par exemple.

Si donc vous souhaitez désinstaller un programme tout en conservant ses fichiers de configuration, il vous suffira de saisir :

```
sudo apt-get remove nom du programme
```

Si par contre vous souhaitez éliminer toute trace d'un programme, y compris ses fichiers de configuration, il suffira de remplacer la commande précédente par :

```
sudo apt-get purge nom du programme
```

2.4.4 Mise à jour d'un programme

Outre ses fonctions d'installation et de désinstallation de programmes, la commande `apt-get` permet de mettre à jour ces derniers mais, et c'est là que ça devient encore plus intéressant, elle peut aussi mettre à jour automatiquement tous les programmes installés sur votre système.

Toutefois, avant de lancer une telle mise à jour, il est indispensable de commencer par s'assurer que le cache est lui-même à jour en lançant une commande :

```
sudo apt-get update
```

Une fois que c'est fait, il est possible de mettre à jour tout le système en saisissant la commande :

```
sudo apt-get upgrade
```

■ Attention !

S'il y a longtemps que la distribution utilisée n'a pas été mise à jour, si ces mises à jour sont nombreuses et/ou si votre liaison Internet est lente, cette opération peut prendre beaucoup de temps.

Si par contre vous ne souhaitez mettre à jour qu'un programme précis, la syntaxe à utiliser devient tout simplement :

```
sudo apt-get install nom du programme
```

Il s'agit de la même ligne de commande que celle utilisée pour l'installation mais, `apt-get` détectant que le programme est déjà installé, elle ne procédera alors qu'à une mise à jour de ce dernier.

Arrivés à ce stade de l'ouvrage, nous n'avons pas la prétention d'avoir fait de vous des « pros » de Linux mais nous estimons vous avoir donné suffisamment d'informations pour que vous puissiez utiliser correctement votre Raspberry Pi et, surtout, pour que vous puissiez poursuivre la lecture de cet ouvrage avec profit.

3 • RÉSEAU, WIFI, PARTITIONS ET PARAMÉTRAGES

Dans la majorité des cas, la première mise en service du Raspberry Pi se déroule sans problème et conformément à ce que nous avons vu tout au long des deux chapitres précédents. Cependant, il peut parfois s'avérer nécessaire de « mettre les mains dans le cambouis », soit parce que votre routeur ne dispose pas du mode DHCP, soit parce que vous souhaitez faire fonctionner votre Raspberry Pi en WiFi, soit enfin parce que vous voulez utiliser au mieux l'espace disponible sur sa carte mémoire SD.

Et même si la mise en service s'est passée comme prévu, il se peut que vous ayez à retoucher l'affichage vidéo, parce qu'il déborde l'écran ou ne le remplit pas complètement par exemple, ou bien encore que vous ayez envie « d'overclocker » votre Raspberry Pi pour le rendre un peu plus performant.

Ce chapitre va donc être consacré à la mise en œuvre de toutes ces possibilités qui, sans être complexes, nécessitent tout de même des connaissances de Linux que ne possèdent pas forcément tous les utilisateurs potentiels du Raspberry Pi.

3.1 Réseau câblé sans DHCP

Si le routeur, ou la « box » Internet qui en fait fonction, auquel est relié le Raspberry Pi dispose d'un mode DHCP, c'est-à-dire s'il affecte automatiquement une adresse IP à tout nouvel abonné connecté au réseau qu'il gère ; la mise en service de l'interface réseau du Raspberry Pi a du se dérouler sans aucune difficulté et vous avez pu constater, comme nous l'avons vu au chapitre 2.2, que le Raspberry Pi vous indiquait même l'adresse IP que lui avait attribué le routeur.

Si vous n'avez pas fait attention à cette information, disponible à la fin des écrans de texte qui se déroulent lors du démarrage initial du Raspberry Pi, vous pouvez faire appel à la commande :

```
pi@raspberrypi ~ $ hostname -I
```

pour prendre connaissance de l'adresse IP affectée à votre machine.

3.1.1 Accès au réseau local

Si en revanche le routeur travaille en mode adresses IP fixes, la connexion au réseau du Raspberry Pi a normalement du échouer car il faut, dans une telle situation, lui donner toutes les informations relatives au réseau et notamment lui affecter une adresse IP et lui indiquer l'adresse de la passerelle. Pour ce faire, il faut éditer un fichier de configuration baptisé `interfaces` situé dans le répertoire `/etc/network/`.

Afin de réaliser cette opération dans les meilleures conditions possibles, nous vous conseillons de lancer l'interface graphique LXDE puis d'ouvrir `LXTerminal` car il va falloir travailler en ligne de commande.

L'édition du fichier `interfaces` étant fort simple à réaliser, un petit éditeur de texte aux fonctionnalités limitées est suffisant, tel que `nano`, disponible d'origine avec la distribution Raspbian Wheezy. Pour le lancer, vous saisissez donc tout simplement la ligne suivante :

```
pi@raspberrypi ~ $ sudo nano /etc/network/interfaces
```

ce qui provoquera l'affichage du contenu du fichier `interfaces` ; contenu particulièrement dépouillé dans sa version par défaut que vous pouvez découvrir figure 3.1.

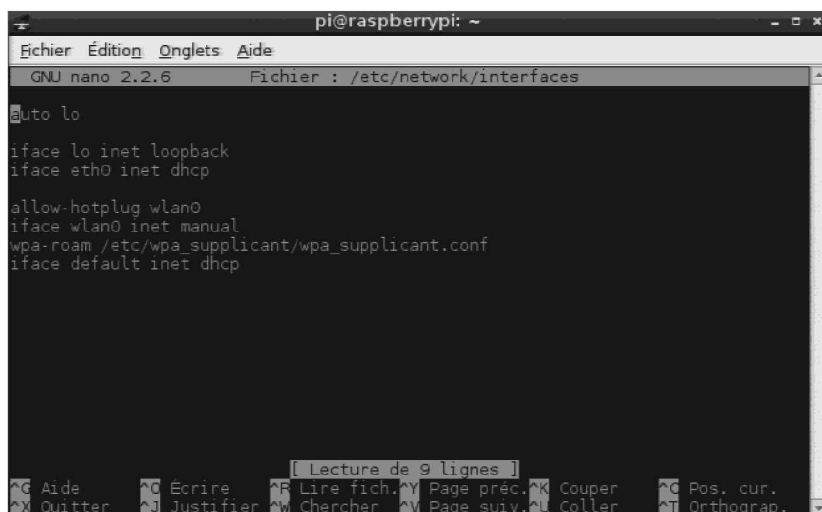


Figure 3.1 – Contenu initial du fichier `interfaces`.

La ligne à modifier est celle qui contient :

```
iface eth0 inet dhcp
```

Il faut tout d'abord remplacer `dhcp` par `static` puis, sur les trois lignes suivantes, saisir :

```
(tabulation)address xxx.xxx.xxx.xxx
(tabulation)netmask xxx.xxx.xxx.xxx
(tabulation)gateway xxx.xxx.xxx.xxx
```

où (tabulation) représente bien entendu un appui sur la touche **Tab** du clavier et où chacun des triplets **xxx** correspond aux adresses IP de votre réseau. Ainsi par exemple, si le Raspberry Pi doit être connecté à un réseau domestique dont les adresses sont de la forme **192.168.0.xxx**, dont la passerelle, une Freebox par exemple, est à l'adresse **192.168.0.254**, et que vous souhaitez lui affecter l'adresse **192.168.0.135**, il suffira de saisir :

```
(tabulation) address 192.168.0.135
(tabulation) netmask 255.255.255.0
(tabulation) gateway 192.168.0.254
```

Toute autre valeur est bien sûr possible et dépend de la plage d'adresses de votre réseau ainsi que de l'adresse de la passerelle (du routeur si vous préférez).

Lorsque cette saisie est terminée, sauvegardez le fichier avec un **CNTRL O**, validez le nom de fichier par **Entrée** et fermez l'éditeur **nano** avec un **CNTRL X**.

Pour faire prendre en compte ces nouveaux paramètres, il est nécessaire de redémarrer le Raspberry Pi mais, si vous ne voulez pas débrancher puis brancher à nouveau votre Raspberry Pi pour cela, sachez que vous pouvez le faire par logiciel grâce à la commande :

```
pi@raspberrypi ~ $ sudo reboot
```

Votre Raspberry Pi doit désormais indiquer, à la fin de ses écrans de chargement, l'adresse IP que vous avez saisie dans le fichier. Si par extraordinaire cela ne fonctionnait pas, il faudrait revoir le contenu de ce fichier dans lequel une faute de frappe a très certainement été commise.

Si nécessaire, le retour au fonctionnement en mode DHCP est possible à tout instant en éditant à nouveau ce même fichier et en remplaçant tout simplement **static** par **dhcp**.

3.1.2 Accès à Internet

Malgré son succès, la manipulation précédente vous a peut-être laissé sur votre faim. En effet, si elle a bien eu comme résultat de permettre au Raspberry Pi de se connecter correctement à votre réseau, il ne peut toujours pas accéder à Internet.

En effet, si un routeur fonctionnant en mode DHCP gère aussi automatiquement les adresses des serveurs DNS, ce n'est pas le cas en mode adresse IP fixe et il faut donc indiquer les adresses de ces serveurs au Raspberry Pi.

Rappelons que ces serveurs DNS sont ceux qui, sur Internet, traduisent les noms des sites en leurs véritables adresses IP. Ainsi, lorsque vous écrivez dans la barre d'adresse de votre navigateur *www.dunod.com*, cette adresse « texte » est traduite par les serveurs DNS d'Internet en **195.81.225.57** qui est l'adresse IP réelle du site de votre éditeur préféré.

Pour indiquer les adresses de ces serveurs au Raspberry Pi, il faut à nouveau éditer un fichier de configuration qui se nomme cette fois `resolv.conf` et qui se trouve dans le répertoire `/etc`.

Pour cela, il suffit de procéder comme ci-dessus, toujours au moyen de l'éditeur `nano` en saisissant au niveau console ou dans `LXTerminal` la ligne suivante :

```
pi@raspberrypi ~ $ sudo nano /etc/resolve.conf
```

Sauf cas particulier, ce fichier doit être vide et il suffit d'y insérer une ou plusieurs lignes successives de la forme :

```
nameserver xxx.xxx.xxx.xxx
```

où `xxx.xxx.xxx.xxx` est l'adresse IP d'un serveur de noms, ou DNS si vous préférez. Selon la configuration de votre réseau, cette adresse de DNS pourra être tout simplement l'adresse de votre routeur, ou bien encore celle(s) communiquée(s) par votre FAI (fournisseur d'accès Internet) ou enfin une ou des adresses de serveurs de noms publics telles celles d'OpenDNS par exemple qui sont particulièrement performantes et qui ont pour valeur : DNS Primaire `208.67.222.222` et DNS secondaire `208.67.220.220`.

Dans un tel cas, c'est-à-dire si plusieurs adresses DNS vous sont fournies pour le même serveur (en général il y en a deux, une primaire et une secondaire) ; les deux sont placées à la suite dans le fichier `resolve.conf` sous la forme :

```
nameserver 208.67.222.222
nameserver 208.67.220.220
```

Une fois cette saisie terminée, sauvegardez le fichier avec un `CNTRL O`, validez le nom de fichier par `Entrée` et fermez l'éditeur `nano` avec un `CNTRL X`.

Pour faire prendre en compte ces nouveaux paramètres, il est nécessaire là encore de redémarrer le Raspberry Pi qui doit désormais accéder à Internet sans problème. Un retour au mode DHCP et donc aux adresses DNS fournies par le routeur s'obtient tout simplement en effaçant les lignes `nameserver` que vous avez placées dans le fichier `resolve.conf`.

3.2 Le Raspberry Pi en WiFi

Par défaut, le Raspberry Pi ne dispose d'aucune interface WiFi mais il est bien sûr possible de lui en adjoindre une au moyen d'une des nombreuses « clés » WiFi que l'on trouve aujourd'hui sur le marché. Si vous ne disposez pas d'un accès réseau câblé à l'endroit d'installation de votre Raspberry Pi, c'est même la seule solution dont vous disposiez pour le raccorder au réseau et à Internet.

Sans être particulièrement complexe, cette opération nécessite tout de même quelques manipulations qui dépendent en outre de la clé que vous aurez choisie ; manipulations qui peuvent se compliquer quelque peu du fait que certaines informations sont à télécharger sur Internet sauf que, tant que la clé ne fonctionne pas, vous n'avez pas accès à Internet !

3.2.1 Configuration manuelle de la clé Wi-Pi

Commercialisée entre autres par Farnell (<http://fr.farnell.com>), qui semble s'être fait une spécialité dans la vente des accessoires pour Raspberry Pi, la clé Wi-Pi, visible figure 3.2, présente l'avantage d'être connue en mode natif par la distribution Raspbian Wheezy, ce qui signifie que ses pilotes sont déjà pré-chargés avec la distribution et que, de ce fait, sa configuration est particulièrement simple à réaliser.



Figure 3.2 – La clé USB WiFi, appelée Wi-Pi, est particulièrement simple à configurer.

Si vous ne possédez pas déjà une clé WiFi, et si donc vous devez en acheter une pour votre Raspberry Pi, autant acquérir celle-ci dont le prix est en outre tout à fait normal pour un produit de ce type.

Pour l'utiliser, il suffit de procéder comme indiqué au début du chapitre 2 mais en démarrant cette fois-ci le Raspberry Pi avec cette clé connectée dans un de ses ports USB et, bien entendu, en l'absence de tout câble réseau. La séquence de démarrage se déroule alors comme nous l'avons vu au début de ce même chapitre et, après avoir paramétré le clavier, il est possible d'accéder à l'environnement graphique LXDE tout à fait normalement.

Au moyen de l'éditeur nano que vous devez commencer à bien connaître, et via **LXTerminal**, éditez le fichier **interfaces** contenu dans **/etc/network/interfaces** au moyen de la commande :

```
pi@raspberrypi ~ $ sudo nano /etc/network/interfaces
```

Si vous venez juste de démarrer votre Raspberry Pi pour la première fois, et que ce fichier n'a pas été modifié, vous devez y lire le contenu suivant :

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp
```

```
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Si votre réseau WiFi utilise une clé WEP pour sa protection, éditez le fichier `interfaces` de façon à ce que son contenu devienne le suivant :

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

auto wlan0
iface wlan0 inet dhcp
wireless-essid SSID
wireless-key CLE
```

Si par contre votre réseau WiFi utilise une clé WPA ou WPA2, éditez le fichier `interfaces` pour que son contenu devienne le suivant :

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

auto wlan0
iface wlan0 inet dhcp
wireless-essid SSID
wireless-key CLE
```

Dans un cas comme dans l'autre, `SSID` est le SSID de votre réseau (son nom si vous préférez) et `CLE` est la clé de votre réseau WiFi. Ces deux informations doivent être saisies de façon parfaitement identique à ce qui a été défini au niveau de votre routeur ou point d'accès WiFi. En particulier, les majuscules et les minuscules doivent être scrupuleusement respectées. Une fois l'édition terminée, validez les modifications réalisées par un `CNTRL O` suivi de `Entrée` puis quittez nano par `CNTRL X`.

Afin que ces modifications soient prises en compte, il faut arrêter puis relancer l'interface réseau ce qui, même si on peut le réaliser au moyen de commandes Linux, est obtenu beaucoup plus rapidement en arrêtant puis en redémarrant le Raspberry Pi ou, encore plus simplement, en exécutant :

```
pi@raspberrypi ~ $ sudo reboot.
```

Vous devez alors constater que ce dernier est bien connecté au réseau, comme doit vous le confirmer l'écran de démarrage en affichant l'adresse IP allouée à votre machine. Il est également possible de lancer LXDE et de tenter d'accéder à Internet avec un des navigateurs fournis pour le vérifier.

Enfin, sachez que la commande `iwconfig`, que vous pouvez lancer sans aucun paramètre et sans même devoir recourir à `sudo`, est à même de vous donner toutes les informations utiles sur la configuration WiFi de votre Raspberry Pi comme indiqué figure 3.3.

```

pi@raspberrypi ~ $ iwconfig
lo          no wireless extensions.

wlan0      IEEE 802.11bgn  ESSID:"WIFIDESBUREAUX"
Mode:Managed  Frequency:2.462 GHz  Access Point: 00:09:5B:D0:36:42
Bit Rate=54 Mb/s   Tx-Power=20 dBm
  Retry  long limit:7   RTS thr:off   Fragment thr:off
Power Management:on
Link Quality=51/70  Signal level=-59 dBm
Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
Tx excessive retries:0 Invalid misc:4  Missed beacon:0

eth0       no wireless extensions.

pi@raspberrypi ~ $

```

Figure 3.3 – La commande iwconfig permet de tout savoir de la configuration WiFi.

3.2.2 Configuration assistée de la clé Wi-Pi

Depuis la dernière mouture de la distribution Raspbian Wheezy (au moment où ces lignes sont écrites), un utilitaire graphique de configuration est disponible via l'interface graphique LXDE. Il est accessible directement au moyen de l'icône **wifi config** présente sur le bureau. Par défaut, cet utilitaire ouvre une fenêtre dans laquelle la boîte déroulante **Network** est pour l'instant vide.

Un clic sur le bouton **scan** permet au logiciel de scanner tous les réseaux WiFi reçus par la clé comme indiqué figure 3.4 ; réseaux dont les noms et caractéristiques principales sont ensuite affichés sur l'écran.

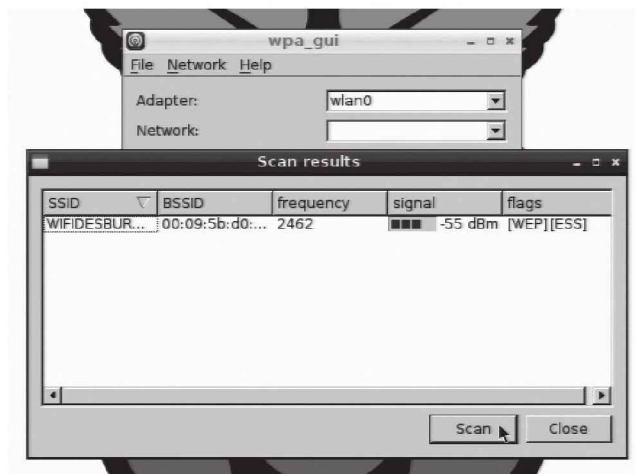


Figure 3.4 – Le logiciel de configuration scanne les réseaux WiFi reçus par la clé.

Un double-clic sur le réseau auquel vous souhaitez connecter le Raspberry Pi a pour effet de faire afficher sa fenêtre de paramétrage, visible figure 3.5 ; fenêtre dans laquelle il est nécessaire de renseigner la ou les clés de cryptage selon le cas. Dans l'exemple choisi ici, le réseau sans fil utilise une clé WEP qu'il suffit donc de renseigner à la rubrique **key 0** par exemple.

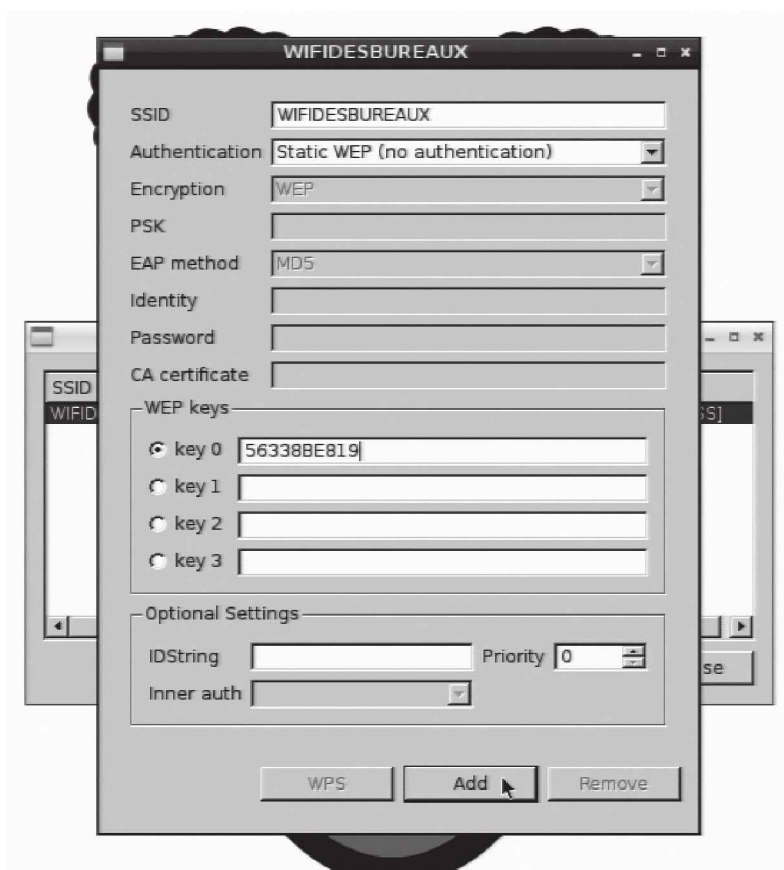


Figure 3.5 – Une fois le réseau sélectionné, il faut définir sa ou ses clés de cryptage.

Lorsque c'est fait, un clic sur le bouton **Add** permet d'ajouter ce réseau dans la liste de ceux auxquels le Raspberry Pi sera susceptible de se connecter. Remarquez que, comme c'est le cas aujourd'hui sur tous les smartphones, il est possible d'ajouter autant de réseaux qu'on le souhaite ; leur sélection ayant lieu ensuite par leurs noms respectifs au niveau de l'utilitaire de configuration **WiFi Config**.

La fenêtre qui s'était ouverte lors du lancement de **WiFi Config** est alors automatiquement complétée avec les paramètres du réseau ainsi défini comme vous pouvez le constater figure 3.6.

Il ne reste plus qu'à cliquer sur **Connect** pour que le Raspberry Pi se connecte au réseau sans-fil ainsi choisi.

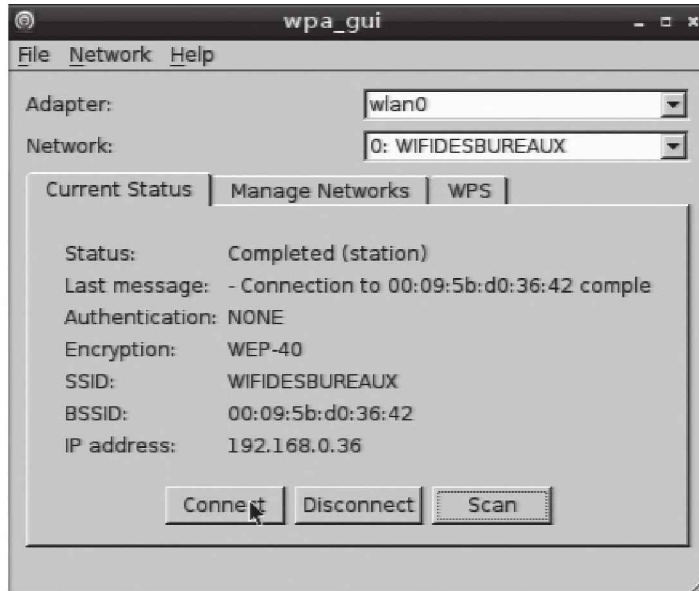


Figure 3.6 – Le réseau sélectionné apparaît dans la fenêtre de configuration de WiFi Config.

3.2.3 Configuration des « autres » clés

Il est évidemment possible d'utiliser d'autres clés USB WiFi que le modèle Wi-Pi présenté ci-dessus, mais leur configuration sera alors un petit peu plus délicate, ne serait-ce que parce qu'il faudra installer leurs pilotes.

Cependant, toutes les clés USB ne fonctionnent pas nécessairement avec le Raspberry Pi et il donc prudent, avant d'acheter ou d'utiliser telle ou telle clé, de vous rendre sur le site Internet dédié à la présentation des périphériques connus pour fonctionner correctement avec le Raspberry Pi à cette adresse : http://linux.org/RPi_VerifiedPeripherals, à la rubrique USB Wi-Fi Adapters. Si votre clé figure dans la liste, très bien fournie, de clés ayant été validées, vous pouvez vous lancer sans problème. Dans le cas contraire, ce sera à vos risques et périls.

Une fois cette clé choisie, la première opération à réaliser consiste à déterminer le *chipset* qui l'équipe. En effet, même si les fabricants de clés sont légion et si les modèles qu'ils proposent sont très nombreux, toutes ces clés reposent sur des *chipsets*, des jeux de circuits intégrés si vous préférez, provenant des grands noms du circuit intégrés tels qu'Atmel, Intel, etc. et ces *chipsets* ne sont pas si nombreux que cela, ce qui fait que de nombreuses clés en apparence différentes utilisent les mêmes *chipsets*, et donc les mêmes pilotes.

Le site Internet évoqué ci-dessus indique, pour la majorité des clés préconisées, le fabricant de son *chipset* mais, si celui de la clé que vous avez choisie n'y figure pas, ne vous inquiétez pas, le Raspberry Pi va vous dire lui-même de qui il s'agit.

Pour cela, démarrez votre Raspberry Pi comme nous l'avons fait au début du chapitre 2 en ayant pris soin de brancher au préalable votre clé sur un port USB. Comme aucun pilote n'est installé, un certain nombre de messages d'erreur vont être affichés dans le torrent de textes que l'on peut observer suite à la mise sous tension.

Comme il est très difficile de lire ces messages à la volée, une commande Linux est prévue pour cela. Lorsque votre Raspberry a fini de démarrer, lancez LXDE puis **LXTerminal** et saisissez la commande suivante :

```
pi@raspberrypi ~ $ dmesg | grep usb
```

La commande **dmesg** permet de lire ce que l'on appelle le tampon circulaire qui stocke tous les messages émis par le noyau Linux. Le trait vertical est la commande **pipe** en langage Linux, c'est-à-dire qu'elle dirige la sortie de **dmesg**, normalement prévue à l'écran, vers la commande suivante qui est **grep**. Enfin, le terme **usb** qui suit ordonne à cette commande **grep** de ne rechercher que les lignes contenant ce terme, c'est-à-dire les lignes relatives aux différents périphériques connectés sur l'interface USB.

```

pi@raspberrypi: ~
1.8750277] usb usb1: Product: DWC OTG Controller
1.884452] usb usb1: Manufacturer: Linux 3.2.27+ dwc_otg_hcd
1.894730] usb usb1: SerialNumber: bcm2708_usb
1.922350] usbcore: registered new interface driver uas
1.941725] usbcore: registered new interface driver usb-storage
1.961412] usbcore: registered new interface driver libusual
2.101722] usbcore: registered new interface driver usblid
2.111723] usblid: USB HID core driver
2.413127] usb 1-1: new high-speed USB device number 2 using dwc_otg
2.933971] usb 1-1: New USB device found, idVendor=0424, idProduct=5512
2.944631] usb 1-1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
2.953271] usb 1-1.1: new high-speed USB device number 3 using dwc_otg
3.063700] usb 1-1.1: New USB device found, idVendor=0424, idProduct=ec00
3.075145] usb 1-1.1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
3.158497] smc95xx 1-1.1.1.0: eth0: register 'smc95xx' at usb-bcm2708_usb-1.1, smc95xx USB 2.0 Ethernet, b8:27:eb:9d:12
:~$ dmesg | grep usb
3.273315] usb 1-1.2: new low-speed USB device number 4 using dwc_otg
3.393204] usb 1-1.2: New USB device found, idVendor=046a, idProduct=0011
3.405039] usb 1-1.2: New USB device strings: Mfr=0, Product=0, SerialNumber=0
3.431940] input: HID 046a:0011 as /devices/platform/bcm2708_usb/usb1/1-1.2/1-1.2.1.1.0/input/input0
3.448560] generic-usb 0003:046a:0011.0001: input: USB HID v1.11 Keyboard [HID 046a:0011] on usb-bcm2708_usb-1.2/input0
3.553238] usb 1-1.3: new high-speed USB device number 5 using dwc_otg
3.664970] usb 1-1.3: New USB device found, idVendor=05a5, idProduct=9008
3.677188] usb 1-1.3: New USB device strings: Mfr=0, Product=1, SerialNumber=0
3.689889] usb 1-1.3: Product: USB2.0 Hub
3.993444] usb 1-1.3.1: new low-speed USB device number 6 using dwc_otg
4.109684] usb 1-1.3.1: New USB device found, idVendor=046d, idProduct=0050
4.122115] usb 1-1.3.1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
4.134902] usb 1-1.3.1: Product: USB-PS/2 Optical Mouse
4.145479] usb 1-1.3.1: Manufacturer: Logitech
4.164016] input: Logitech USB-PS/2 Optical Mouse as /devices/platform/bcm2708_usb/usb1/1-1.3/1-1.3.1/1-1.3.1.1.0/inp
t/input1
4.187101] generic-usb 0003:046d:0050.0002: input: USB HID v1.10 Mouse [Logitech USB-PS/2 Optical Mouse] on usb-bcm2708_u
sb-1.3.1/input0
4.283905] usb 1-1.3.4: new high-speed USB device number 7 using dwc_otg
4.411119] usb 1-1.3.4: New USB device found, idVendor=14bf, idProduct=5370
4.424234] usb 1-1.3.4: New USB device strings: Mfr=1, Product=2, SerialNumber=3
4.437551] usb 1-1.3.4: Product: 802.11 n WLAN
4.447914] usb 1-1.3.4: Manufacturer: Ralink
4.458009] usb 1-1.3.4: SerialNumber: 1.0
11.473639] usb 1-1.3.4: reset high-speed USB device number 7 using dwc_otg
11.765657] Registered led device: rt2800usb-phy0:radio
11.765797] Registered led device: rt2800usb-phy0:assoc
11.765922] Registered led device: rt2800usb-phy0:quality
11.766043] usbcore: registered new interface driver rt2800usb
pi@raspberrypi ~ $

```

Figure 3.7 – À la recherche du chipset de la clé USB.

Le résultat doit ressembler à ce que vous pouvez observer figure 3.7, avec tous les termes **usb** trouvés affichés en rouge, mais ce n'est pas cela le plus important. Il suffit en effet de parcourir ces lignes jusqu'à trouver celle qui contient :

```
Product : 802.11 x WLAN
```

Elle doit être suivie par la ligne :

Manufacturer : xxxx

où **xxx** est le nom du fabricant du *chipset* de la clé.

Une fois le fabriquant trouvé, il ne reste plus qu'à installer les pilotes correspondants pour pouvoir utiliser la clé et, pour ce faire, nous allons faire appel à une de nos (déjà) vieilles connaissances, la commande `apt-cache`, afin de lui faire rechercher dans le cache ces fameux pilotes.

Attention !

Si, lors de la recherche du fabricant du *chipset* de votre clé, vous obtenez comme réponse Ralink, il est inutile de poursuivre la lecture de ces explications. C'est en effet le même *chipset* que celui de la clé Wi-Fi et tout ce que nous avons écrit précédemment à son sujet est donc valable pour votre clé.

Si par exemple nous sommes en présence d'une clé équipée d'un *chipset* Atmel, il faudra tout simplement saisir, dans une fenêtre de **LXTerminal** :

```
pi@raspberrypi ~ $ apt-cache search atm1
```

Atmel fabricant de nombreux produits susceptibles d'être utilisés avec le Raspberry Pi, la réponse comportera plusieurs lignes comme indiqué figure 3.8, mais une de ces réponses sera clairement identifiable comme correspondant aux pilotes recherchés. Dans le cas de la figure 3.8 c'est la ligne :

```
atmel-firmware - Firmware for Atmel at76c50x wireless networking chips.
```

```

pi@raspberrypi: ~
File Edit Tabs Help
4.424234] usb 1-1.3.4: New USB device strings: Mfr=1, Product=2, SerialNum
4.437581] usb 1-1.3.4: Product: 802.11 n WLAN
4.447914] usb 1-1.3.4: Manufacturer: Ralink
4.458009] usb 1-1.3.4: SerialNumber: 1.0
11.473539] usb 1-1.3.4: reset high-speed USB device number 7 using dwc_otg
11.765657] Registered led device: rt2800usb-phy0::radio
11.765797] Registered led device: rt2800usb-phy0::assoc
11.765922] Registered led device: rt2800usb-phy0::quality
11.766043] usbcore: registered new interface driver rt2800usb
pi@raspberrypi ~ $ apt-cache search atm1
ap-utils - Access Point SNMP Utils for Linux
avarice - use GDB with Atmel's JTAG ICE for the AVR
avr-libc - Standard C library for Atmel AVR development
avra - Assembler for Atmel AVR microcontrollers
avrdude - software for programming Atmel AVR microcontrollers
avrp - Programmer for Atmel AVR microcontrollers
binutils-avr - Binary utilities supporting Atmel's AVR targets
dfu-programmer - device firmware update (DFU) based USB programmer for Atmel chi
openocd - Open on-chip JTAG debug solution for ARM and MIPS systems
s51dude - In-System Programmer for 8051 MCUs using usbtiny
simulavr - Atmel AVR simulator
uisp - Micro In-System Programmer for Atmel's AVR MCUs
atmel-firmware - Firmware for Atmel at76c50x wireless networking chips.
pi@raspberrypi ~ $

```

Figure 3.8 – La commande `apt-cache search` permet de trouver les pilotes nécessaires à la clé WiFi.

Si votre clé est équipée d'un *chipset* d'une autre marque, il suffit de remplacer `atmel` par celle-ci dans la ligne de commande précédente pour parvenir au même résultat, mais avec les pilotes relatifs à la marque utilisée bien sûr.

Il ne reste plus maintenant qu'à installer ces pilotes en utilisant la syntaxe déjà vue au chapitre 2 soit :

```
pi@raspberrypi ~ $ apt-get install firmware
```

où `firmware` représente le nom exact donné aux pilotes tel que l'a indiqué la commande `apt-cache search` précédente.

Une fois cette installation réalisée, vous pouvez suivre les indications que nous avons données au chapitre 3.2.1 pour la configuration manuelle de la clé Wi-Pi puisque l'on se trouve, dès cet instant, dans la même situation que les utilisateurs de cette clé.

Arrivé à ce stade de l'ouvrage, et quelle que soit votre configuration réseau, vous devez donc désormais être en possession d'un Raspberry Pi connecté à ce réseau, que ce soit par câble ou en WiFi. Mais pour profiter pleinement de ses nombreuses possibilités, nous vous proposons d'aller un peu plus loin en nous intéressant à la gestion de sa carte SD.

3.3 Les partitions de la carte SD

Si la préparation de la carte SD du Raspberry Pi à partir d'une image téléchargée sur Internet décharge l'utilisateur de tout travail fastidieux, comme nous l'avons vu au chapitre 1, elle présente au moins un inconvénient, celui de ne pas tenir compte de la capacité de cette carte. Elle utilise en effet seulement la taille à partir de laquelle a été préparée cette image, c'est-à-dire généralement 2 Go dans le cas des images courantes que l'on trouve sur Internet pour le Raspberry Pi. Pour vous en persuader, saisissez par exemple en mode console ou dans `LXTerminal` la commande :

```
pi@raspberrypi ~ $ df -h
```

qui permet d'afficher le taux d'utilisation de la carte SD, et vous risquez fort d'être surpris... En effet, si vous avez acheté une carte de 8 Go, ou mieux de 16 Go comme nous vous l'avons conseillé au chapitre 1, vous vous retrouvez avec seulement 2 Go utilisés et utilisables et tout l'espace restant ne sert à rien.

Pour pouvoir l'exploiter, il est nécessaire de créer une ou plusieurs partitions dans cet espace libre afin de pouvoir ensuite les formater avec un système de fichiers reconnu par Linux. Pour ce faire, il existe trois approches distinctes.

La première consiste à créer de toutes pièces dans cet espace libre une ou plusieurs partitions, puis à les formater et enfin à les « monter » afin de les faire reconnaître par le système. Ce n'est pas très compliqué mais demande tout de même de nombreuses manipulations en ligne de commande, et ce n'est pas forcément utile dans le cas du Raspberry Pi. Nous n'y ferons donc pas appel dans le cadre de cet ouvrage

car, ce que l'on cherche généralement à faire, c'est tout simplement de pouvoir exploiter l'espace libre de la carte SD sans nécessairement qu'il se trouve dans une partition distincte de celle du système d'exploitation. Si tel est bien votre cas, et c'est celui de la majorité des utilisateurs du Raspberry Pi, les deux autres approches vous sont destinées.

La seconde approche consiste à étendre de façon quasi automatique la taille de la partition de départ, celle qui fait 2 Go (en réalité un peu moins), à tout l'espace libre restant sur la carte. Pour ce faire une fonction a été spécialement mise en place dans la distribution Raspbian Wheezy pour y parvenir directement à partir du Raspberry Pi.

La troisième approche consiste également à étendre la taille de la partition de départ mais pas nécessairement à toute la carte mémoire, pour le cas par exemple où vous souhaiteriez ensuite pouvoir créer d'autres partitions pour y installer d'autres choses. Dans ce cas, le recours à un PC, fonctionnant sous Linux ou sous Windows, et disposant d'un lecteur de cartes SD, s'avère nécessaire mais les possibilités de configuration disponibles sont évidemment plus nombreuses.

3.3.1 Extension automatique de la partition principale

Si vous souhaitez tout simplement pouvoir utiliser toute la capacité offerte par votre carte SD sans laisser de l'espace disponible pour d'éventuelles installations ultérieures, un outil automatique est à votre disposition.

Attention !

Pour efficace que soit cet outil, il redimensionne la partition système « à la volée » et il est donc impossible de garantir à 100 % qu'elle ne sera pas endommagée par cette opération. Si votre carte contient des données auxquelles vous tenez, il est donc préférable d'utiliser la méthode manuelle présentée au § 3.3.2 ou de faire une sauvegarde préalable de votre carte comme expliqué au § 3.4.

Cet outil, nous l'avons déjà découvert lors du lancement initial du Raspberry Pi (voir figure 2.2) mais nous ne l'avons utilisé à ce moment-là que pour configurer le clavier et localiser la distribution Raspbian. Le seul « problème » est qu'il ne se lance automatiquement que lors du premier démarrage du Raspberry Pi. Pour pouvoir y faire appel maintenant il faut donc recourir à une commande spécifique, depuis le mode console ou **LXTerminal** en saisissant :

```
pi@raspberrypi ~ $ sudo raspi-config
```

Comme indiqué figure 3.9, cela provoque à nouveau l'affichage de l'écran de configuration initial sur lequel il ne vous reste qu'à sélectionner la ligne **expand_rootfs**.

Après avoir validé par **Entrée**, quelques messages texte sont visibles, à la suite de quoi vous êtes informé que le redimensionnement de la partition aura lieu effectivement lors du prochain redémarrage du système ; redémarrage qui vous est proposé lorsque vous quittez **raspi-config** en sélectionnant **Finish**.

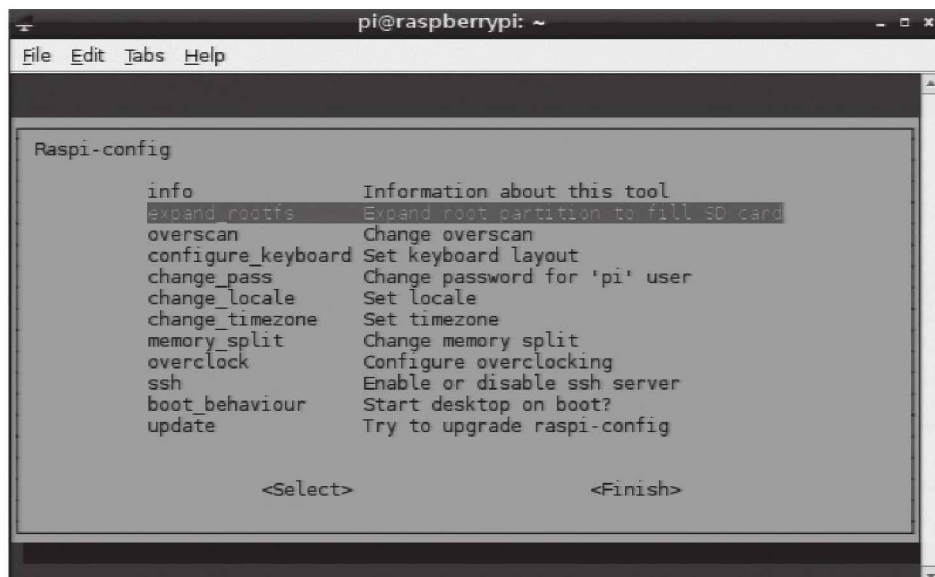


Figure 3.9 – La commande raspi-config permet de retrouver l'écran de configuration initial.

Attention !

Selon la capacité et la vitesse de votre carte mémoire, l'opération de redimensionnement peut nécessiter plusieurs minutes pendant lesquelles vous aurez l'impression que le système reste bloqué sur la ligne *Performing an on-line resize*. C'est normal et il faut attendre la fin de l'opération sans toucher à quoi que ce soit au risque d'endommager le contenu de la carte.

3.3.2 Extension manuelle de la partition principale

Si vous souhaitez pouvoir maîtriser au mieux la gestion de vos partitions et vous affranchir du risque, certes léger mais tout de même réel, de perdre le contenu de votre carte SD, une meilleure solution consiste à faire appel à un programme de gestion des partitions fonctionnant sur un PC tiers équipé d'un lecteur de carte SD.

Si ce PC fonctionne sous Linux, il doit disposer de l'outil baptisé **gparted** et, si tel n'est pas le cas, il suffit de l'installer au moyen du gestionnaire de paquets de la distribution Linux utilisée sur ce PC (`sudo apt-get install gparted` dans le cas d'une distribution Debian par exemple).

Si par contre ce PC fonctionne sous Windows, et même s'il dispose d'un outil de gestion des partitions, il est inutilisable car Windows ne reconnaît pas les formats des partitions Linux. Il faut donc faire appel à un outil spécifique, en l'occurrence le fameux **gparted** que nous venons d'évoquer, mais au travers d'un live CD puisque **gparted** ne fonctionne pas sous Windows.

Si donc votre PC fonctionne sous Linux, insérez la carte SD du Raspberry Pi dans son lecteur, lancez **gparted** et reportez-vous quelques lignes plus bas dans ce chapitre où nous décrivons son utilisation.

Si par contre votre PC fonctionne sous Windows, insérez aussi la carte SD du Raspberry Pi dans son lecteur de cartes mais réalisez ensuite les manipulations ci-dessous avant de pouvoir utiliser **gparted**.

À partir des compléments en ligne de cet ouvrage, ou directement à partir du site Internet de son auteur, à l'adresse <http://partedmagic.com>, téléchargez l'image ISO destinée à la fabrication du live CD **PartedMagic** puis, avec le logiciel de gravure de votre choix, gravez cette image sur un CD.

Attention !

Il faut impérativement mettre votre logiciel de gravure de CD en mode gravure d'image ISO et non en mode gravure de données, faute de quoi le disque serait inutilisable.

Une fois le disque terminé, redémarrez votre PC et accédez à son menu de boot (touche **F12** sur nombre de PC récents ou passage par le **setup** du BIOS pour les machines plus anciennes) afin de le faire démarrer sur le CD que vous venez de graver.

Cette opération est totalement sans danger pour votre PC. En effet, ce que vous venez de graver est un live CD, c'est-à-dire un CD supportant un système d'exploitation (Linux en l'occurrence) qui fonctionne entièrement en mémoire vive de votre machine et qui n'utilise en aucun cas son disque dur. Vous ne risquez donc pas d'altérer ni Windows, ni vos fichiers de données en réalisant une telle manipulation.

Au bout de quelques secondes, l'écran visible figure 3.10 s'affiche. Positionnez la surbrillance sur la première ligne, ce qui doit normalement être le cas par défaut, et validez en appuyant sur **Entrée**.

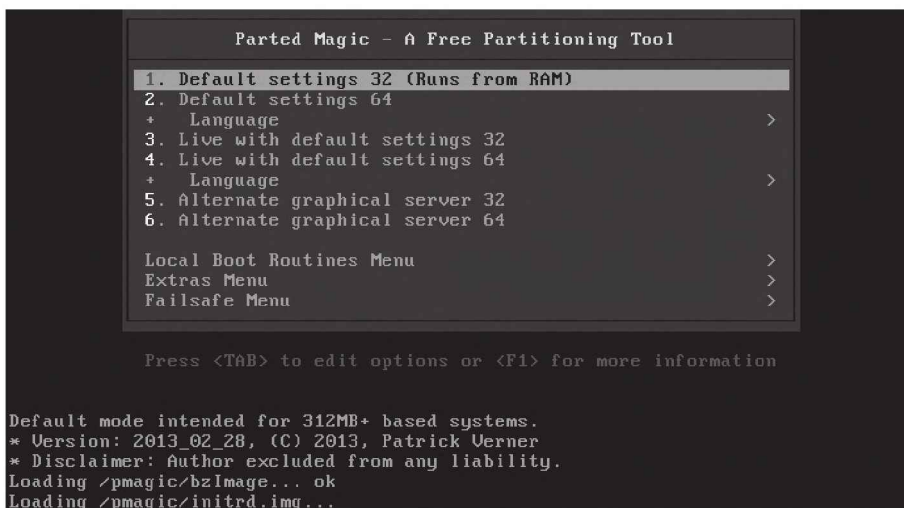


Figure 3.10 – L'écran de démarrage du CD PartedMagic.

Après une nouvelle attente, où vous verrez défiler sur l'écran des lignes de textes similaires à celles auxquelles vous a habitué votre Raspberry Pi, le bureau de **PartedMagic**, visible figure 3.11, doit s'afficher.



Figure 3.11 – Le bureau de PartedMagic.

Vous pouvez alors lancer **Partition Editor** en double cliquant sur l'icône prévue à cet effet sur ce bureau. Par défaut, cet éditeur de partitions qui n'est autre que **gparted**, s'ouvre sur la première mémoire de masse détectée sur le PC qui est généralement son disque dur principal. Il faut donc cliquer en haut et à droite de cet écran, sur les flèches situées au niveau de la boîte représentant les mémoires de masse (le curseur est positionné au bon endroit sur la figure 3.12), afin de sélectionner la carte SD du Raspberry Pi.

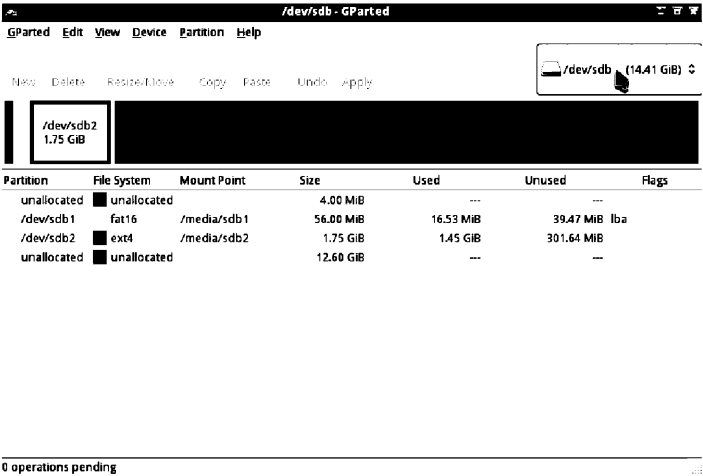
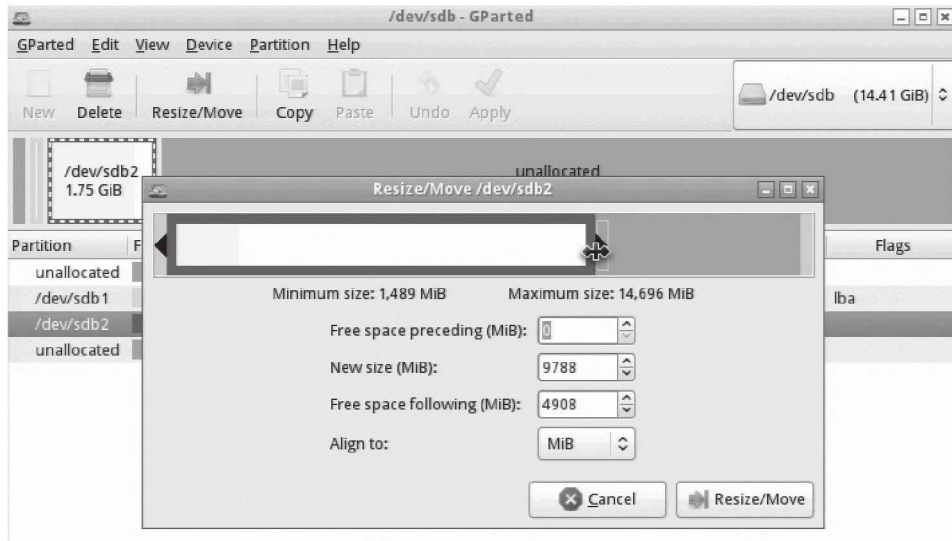


Figure 3.12 – Le contenu initial de la carte SD du Raspberry Pi affiché dans gparted.

Comme vous pouvez le constater, la plus grande partie de la mémoire de la carte est marquée comme **unallocated**, c'est-à-dire non allouée. Nous allons donc y étendre la partition principale du système qui est celle repérée `/dev/sdb2` dont la taille actuelle doit être de l'ordre de 1,7 Go (cela peut varier un peu en fonction de la date à laquelle vous lirez ces lignes).

Pour cela, cliquez sur le dessin de cette partition en partie haute de l'écran puis sélectionnez **Resize/Move** dans la barre d'outils. Cela conduit à un affichage similaire à celui visible figure 3.13 sur lequel il ne reste plus qu'à faire glisser avec la souris le bord droit du rectangle représentant la partition.



0 operations pending

Figure 3.13 – L'opération de redimensionnement de la partition est entièrement graphique.

Attention !

Lors de cette opération, veillez bien à ce que le curseur de la souris prenne la forme visible figure 3.13 (deux flèches de part et d'autre d'un trait vertical) et non celui de quatre flèches en croix car, dans ce dernier cas, vous déplacerez la partition au lieu de l'agrandir.

Vous pouvez bien évidemment amener le bord droit à la limite de l'écran, pour utiliser toute la place disponible sur la carte, ou vous arrêter avant la fin si vous souhaitez laisser de l'espace libre pour une utilisation ultérieure.

Validez votre choix en cliquant sur **Resize/Move** puis sur **OK** ce qui conduit à nouveau à l'écran principal de **gparted** sur lequel vous pouvez constater que la taille de la nouvelle partition principale correspond bien à ce que vous venez de définir.

Attention !

À ce stade des opérations, l'opération de redimensionnement n'a pas encore été exécutée et il reste possible de l'annuler en cas d'erreur en cliquant sur Undo dans la barre d'outils.

Si cela vous convient, cliquez sur l'icône **Apply** de la barre d'outils et patientez jusqu'à la fin de l'opération de redimensionnement, ce qui peut demander plusieurs minutes, fonction de la taille et de la vitesse de votre carte SD.

Celle-ci peut ensuite être retirée de son lecteur afin de regagner votre Raspberry Pi où vous bénéficierez alors de toute sa capacité, comme un nouveau recours à la commande `df -h` vue précédemment pourra vous le confirmer.

3.4 Sauvegardez votre carte SD

Bien que les manipulations proposées précédemment ne soient pas particulièrement risquées, elles posent tout naturellement la question de la sauvegarde du contenu de la carte SD. Si cette sauvegarde s'avère sans importance lors des premières utilisations du Raspberry Pi, puisque la carte ne contient alors guère plus que le système d'exploitation et les quelques paramètres que vous y avez réalisés, et qu'il est donc facile de la régénérer à partir de l'image téléchargée sur Internet, ce ne sera plus le cas lorsque vous aurez commencé à travailler plus sérieusement en écrivant des programmes par exemple. Nous vous proposons donc de découvrir comment sauvegarder la carte SD de votre Raspberry Pi ce qui, vous allez le voir, est d'une extrême simplicité, même si vous ne possédez qu'un PC fonctionnant sous Windows.

La solution la plus simple pour sauvegarder la carte SD du Raspberry Pi est tout simplement d'en faire une image que vous stockerez sur le support de votre choix, tel que le disque dur de votre PC par exemple.

Si votre PC fonctionne sous Linux, il vous suffira d'utiliser la commande `dd` bien connue après avoir placé la carte SD du Raspberry Pi dans son lecteur.

Si votre PC fonctionne sous Windows, ce sera un petit peu plus délicat puisque Windows ne sait pas lire les partitions Linux d'une part, et qu'il ne dispose pas d'un logiciel permettant de réaliser une image disque à partir de ces dernières d'autre part. Fort heureusement, si vous avez lu l'intégralité du chapitre 1, vous avez dû installer sur votre PC **win32 Disk Imager** afin de réaliser la carte SD destinée à votre Raspberry Pi à partir de l'image téléchargée sur Internet, et ce logiciel sait aussi fabriquer des images à partir du contenu d'un support externe.

Après avoir inséré la carte SD du Raspberry Pi dans le lecteur de votre PC, il suffit de lancer **win32 Disk Imager** et de renseigner, avec le nom de fichier de votre choix, la case **Image File** en n'oubliant pas de préciser le suffixe `.img` après le nom de ce dernier. Par défaut, ce fichier sera automatiquement placé dans le répertoire `C:\Users\Utilisateur\Downloads\` pour les machines fonctionnant sous Windows 7 et dans `C:\Documents and Settings\Utilisateur\Mes Documents\Downloads\` pour les machines fonctionnant sous Windows XP. Dans les expressions précédentes, **Utilisateur** représente évidemment le nom du répertoire de l'utilisateur de la machine.

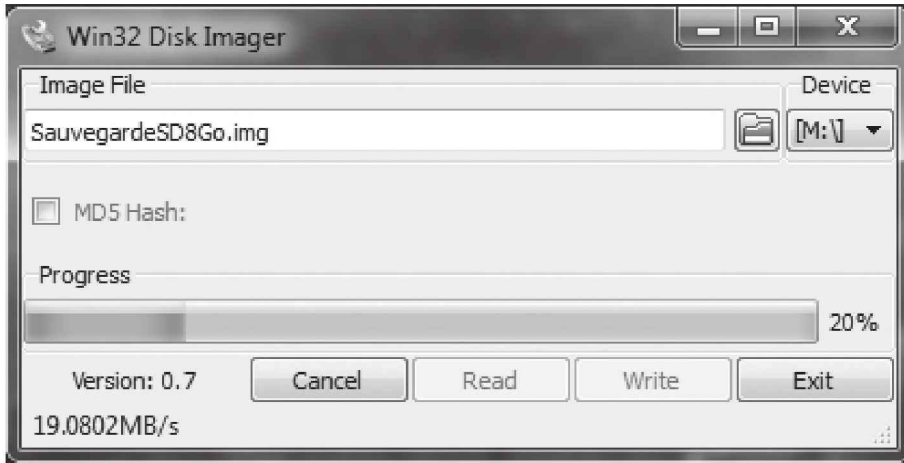


Figure 3.14 – Win32 Disk Imager permet de réaliser une image de la carte SD.

Un clic sur le bouton **write** lance alors le processus de création de l'image qui peut nécessiter plusieurs dizaines de minutes si la carte est de grande capacité.

Le fichier ainsi réalisé pourra ensuite être utilisé, si nécessaire, avec ce même **Win32 Disk Imager** pour recréer la carte SD du Raspberry Pi dans le cas où celle-ci viendrait à être corrompue pour une raison ou pour une autre.

3.5 Le fichier config.txt

Contrairement aux PC sur lesquels il est possible de définir de façon permanente un certain nombre d'informations de configuration matérielles, grâce notamment au BIOS et à sa mémoire EEPROM associée qui stocke ces dernières, les informations de configuration matérielle du Raspberry Pi sont découvertes par le système d'exploitation à chaque mise sous tension grâce à un certain nombre de fichiers spécifiques.

En éditant le contenu de certains de ces fichiers, il est donc possible, dans une certaine mesure, de modifier le comportement du Raspberry Pi et donc de l'adapter à tel ou tel besoin particulier. Nous vous proposons donc de découvrir, dans les lignes qui suivent, le plus important d'entre eux et, surtout, ce qu'il est possible d'y modifier.

Sachez tout d'abord que ces fichiers se trouvent dans le répertoire `/boot` de la carte SD, ou répertoire de démarrage (de « boot » si vous préférez) du Raspberry Pi et que le plus important d'entre eux est le fichier `config.txt` auquel nous allons consacrer les lignes qui suivent. D'autres fichiers tout aussi importants y sont également présents mais leur modification ne présente généralement pas d'intérêt pour la majorité des utilisateurs et risque en outre de rendre le fonctionnement du Raspberry Pi instable. Nous ne les étudierons donc pas dans le cadre de cet ouvrage.

Avant de voir les informations qu'il est possible d'ajouter et/ou de modifier dans le fichier `config.txt`, précisons que c'est un simple fichier texte dans lequel il suffit d'écrire, ou d'effacer ces informations afin de contrôler les paramètres qui leur correspondent.

■ Attention !

Ce fichier n'est lu qu'une fois lors du démarrage du Raspberry Pi. Les modifications que vous lui ferez subir ne pourront donc être prises en compte que lors du redémarrage suivant.

3.5.1 Modifier les paramètres d'affichage

La première information que vous serez susceptible de modifier concernera très probablement l'affichage. En effet, si tout se passe bien dans la majorité des cas, il peut arriver que celui-ci déborde de l'écran, n'occupe pas la totalité de sa surface ou bien encore, si vous utilisez la « vieille » sortie vidéo composite, que cette dernière ne soit pas au bon standard TV. De nombreux paramètres sont à votre disposition pour corriger tout cela que nous vous proposons de découvrir ci-dessous.

hdmi_drive permet de faire varier les niveaux électriques fournis par la sortie HDMI. Par défaut, ces derniers sont adaptés à un moniteur HDMI vrai et lui conviennent donc très bien. En revanche, si vous utilisez un adaptateur HDMI – DVI, il se peut que l'image reproduite soit fade ou, au contraire saturée. Si tel est le cas, vous pouvez rendre ce paramètre égal à 1 (niveaux adaptés à une interface DVI) ou à 2 (niveaux adaptés à une interface HDMI). Dans le cas où ce paramètre est égal à 1, la sortie HDMI est traitée comme une sortie DVI et ne véhicule donc plus le son.

hdmi_force_hotplug, lorsqu'il est mis à 1, oblige le Raspberry Pi à utiliser l'interface HDMI même s'il ne détecte aucun moniteur connecté sur cette dernière. Cela peut s'avérer utile avec certains câbles d'adaptation DVI et/ou VGA.

hdmi_mode permet de définir « de force » la résolution d'affichage utilisée par le moniteur connecté sur la sortie HDMI. En effet, cette résolution est fixée par défaut en fonction des informations transmises par le moniteur, mais il peut s'avérer nécessaire, si l'on veut rendre l'information plus lisible par exemple, de changer cette résolution. Les résolutions proposées étant très nombreuses, un code leur a été affecté ; code que vous trouverez en annexe A auquel il suffit de rendre égal ce paramètre.

config_hdmi_boost peut être rendu égal à n'importe quel chiffre entre 1 (minimum) et 7 (maximum) afin de faire varier les niveaux électriques produits sur l'interface HDMI, et ainsi à compenser dans une certaine mesure les pertes induites par de longs câbles de liaison. Si donc votre image est faiblement contrastée, vous pouvez essayer d'augmenter ce paramètre.

hdmi_group est complémentaire de **hdmi_mode** vu précédemment. Il permet de définir le groupe d'appartenance des paramètres supportés par le moniteur relié à la prise HDMI. La valeur 1 correspond aux normes CEA américaines tandis que la valeur 2 correspond aux normes VESA plus largement répandues chez nous. Selon que l'on se trouve dans un mode ou dans l'autre, les codes des résolutions d'affichage utilisés par **hdmi_mode** varient comme indiqué annexe A.

hdmi_safe, lorsqu'il est mis à 1, force un certain nombre de paramètres précédents sur des valeurs « sûres », c'est-à-dire sur des valeurs susceptibles de fonctionner dans la majorité des situations. Il impose donc d'un seul coup le paramétrage suivant : **hdmi_group=1**, **hdmi_mode=1**, **hdmi_force_hotplug=1**, **hdmi_boost=4** et **disable_overscan=0** (dont nous verrons le rôle ci-après).

overscan_left, **overscan_right**, **overscan_top** et **overscan_bottom** permettent de faire afficher l'image après un certain nombre de pixels vides, respectivement à partir de la gauche, de la droite, du haut et du bas de l'écran. Ces paramètres permettent donc de corriger les problèmes causés sur certains moniteurs par un débordement de l'image hors de l'écran, et donc de faire afficher correctement des informations qui, sans cela, seraient perdues ou au minimum tronquées. Il est possible d'utiliser un ou plusieurs de ces paramètres simultanément mais à raison de un par ligne, et ils doivent être rendus égaux au nombre pixels de décalage désiré (par exemple **overscan_left=30** pour décaler l'image de 30 pixels vers la gauche).

disable_overscan doit être rendu égal à 1 si, contrairement au cas que nous venons de voir, l'image ne remplit pas l'écran et se trouve de ce fait entourée de bordures noires.

sdtdv_mode permet de définir la norme des signaux vidéo composites générés sur la prise de même nom (la grosse prise Cinch jaune). Quatre valeurs sont autorisées : 1 pour la norme NTSC américaine, 2 pour la norme NTSC japonaise, 3 pour la norme PAL de la majorité de la planète et 4 pour la norme PAL du Brésil.

sdtdv_aspect permet quant à lui de définir, toujours pour la sortie vidéo composite, le format d'affichage. Trois valeurs sont autorisées : 1 pour un affichage 4/3, 2 pour un affichage 14/9 et enfin 3 pour un affichage en 16/9, soit le standard actuel le plus répandu.

Si, par exemple, vous souhaitez forcer la détection d'un moniteur HDMI sur lequel l'image est affectée de bords noirs et que vous utilisiez un câble HDMI long, vous éditez le fichier `config.txt` au moyen d'une commande :

```
pi@raspberrypi ~ $ sudo nano /boot/config.txt
```

saisie depuis le mode console ou `LXTerminal`. Et vous écrirez dans ce fichier les informations suivantes :

```
hdmi_force_hotplug=1
disable_overscan=1
config_hdmi_boost=6
```

Vous le validerez ensuite par `CNTRL O` puis `Entrée` et enfin `CNTRL X` pour quitter `nano`. Comme le fichier n'est pris en compte que lors du démarrage du Raspberry Pi, vous pourrez le provoquer par un simple :

```
pi@raspberrypi ~ $ sudo reboot
```

3.5.2 Comment « overclocker » le Raspberry Pi ?

Le processeur du Raspberry Pi est loin d'être un foudre de guerre, surtout si on le compare aux « monstres » dont peuvent disposer aujourd'hui nos PC. Même si cela n'a aucune importance dans la majorité des applications où il excelle, certains d'entre vous seront peut-être tentés d'essayer de le faire fonctionner plus vite ; de « l'overclocker » comme on dit couramment.

Une telle opération est parfaitement possible même si, comme nous avons pu le vérifier sur plusieurs échantillons de Raspberry Pi, la marge de manœuvre est relativement faible et le gain en performances modeste. Par ailleurs, n'oubliez pas que toute augmentation de la fréquence d'horloge se traduit par une augmentation de la dissipation de puissance, et donc de chaleur du processeur. Lors de ces manipulations, il conviendra donc de s'assurer que celui-ci ne s'échauffe pas trop, en évitant par exemple les boîtiers trop confinés.

En réalité, et même si par abus de langage on parle du processeur qui équipe le Raspberry Pi, il s'agit en fait de ce que l'on nomme aujourd'hui un SoC qui est l'acronyme de *System on a Chip* soit, mot à mot, système sur une puce. Si vous savez ce qu'est un microcontrôleur, vous pouvez très bien considérer qu'un SoC n'est rien d'autre qu'un « gros » microcontrôleur !

Toujours est-il que, de ce fait, il n'existe pas une horloge unique pour cadencer le fonctionnement de ce SoC mais trois : une première pour l'unité centrale proprement dite constituée du processeur ARM, la seconde pour définir la vitesse d'accès à la mémoire et la troisième pour le GPU ou processeur graphique. Mieux même, cette dernière se subdivise à son tour en quatre horloges distinctes affectées aux différents sous-ensembles de celui-ci.

Par le biais du fichier `config.txt`, il est possible d'agir séparément sur toutes ces horloges en rendant tout simplement le paramètre qui leur correspond égal à la fréquence d'horloge désirée exprimée en MHz. Voici donc quels sont ces paramètres ainsi que les valeurs par défaut qui leur sont affectées ; valeur qui sont tout de même, rappelons-le, celles préconisées par Broadcom, fabricant de la puce...

arm_freq est la fréquence du processeur ARM qui équipe le SoC. Son augmentation est censée améliorer quelque peu les performances dans tous les domaines puisque le processeur est sollicité, quoi que vous fassiez avec votre Raspberry Pi. Valeur par défaut 700 MHz.

sdram_freq est la fréquence d'accès à la RAM. Son augmentation est censée améliorer les performances globales puisque, tout comme le processeur, la RAM est sollicitée quoi qu'on fasse. Valeur par défaut 700 MHz.

init_emmc_clock permet de définir la fréquence de fonctionnement du contrôleur de la carte SD. Son augmentation permet d'accélérer les échanges avec la carte SD, pour peu que celle-ci le supporte bien sûr, mais augmente les risques de corruption de données. Valeur par défaut 80 MHz.

gpu_freq est la fréquence du processeur graphique. Son augmentation procure une augmentation de toutes les performances graphiques, contrairement à celle des sous-ensembles décrits ci-après qui n'agissent que dans un domaine bien particulier. Valeur par défaut 250 MHz.

core_freq est la fréquence du noyau du processeur graphique. Valeur par défaut 250 MHz.

h264_freq est la fréquence du décodeur matériel des flux vidéo haute définition. Compte tenu du fait que le GPU décode très bien la vidéo HD en 1080p, on ne voit pas très bien qu'est-ce qui pousserait à l'augmenter. Valeur par défaut 250 MHz.

isp_freq est la fréquence du sous-ensemble de capture vidéo utilisé par exemple si l'on raccorde une caméra au Raspberry Pi. Valeur par défaut 250 MHz.

v3d_freq est la fréquence de l'accélérateur matériel 3D. Valeur par défaut 250 MHz. Si vous souhaitez modifier l'une ou l'autre de ces fréquences, il suffit tout simplement d'ajouter les lignes correspondantes dans le fichier `config.txt` en procédant comme nous l'avons fait ci-dessus pour les paramètres d'affichage.

Ainsi, si vous voulez porter la fréquence du processeur à 800 MHz et celle du GPU à 300 MHz, vous écrirez tout simplement dans le fichier `config.txt` :

```
arm_freq=800
gpu_freq=300
```

Mais attention à la fumée qui risque de se dégager de votre Raspberry Pi !

3.5.3 Comment survolter le Raspberry Pi ?

Lorsque l'on augmente la fréquence d'horloge, que ce soit de l'unité centrale ou du GPU, on arrive bien évidemment à une limite au-delà de laquelle le circuit ne fonctionne plus du tout. Pour tenter de la dépasser, il existe alors une recette ultime qui consiste à augmenter légèrement la tension d'alimentation. Cette augmentation doit toutefois rester très mesurée car, outre le fait qu'elle risque de dépasser les limites admises par le circuit, elle fait aussi augmenter très rapidement la dissipation de puissance de la puce et donc son échauffement.

Malgré les risques encourus, les concepteurs du Raspberry Pi ont prévu de pouvoir augmenter la tension d'alimentation de divers sous-ensembles du Raspberry Pi au moyen de paramètres placés, comme les précédents, dans le fichier `config.txt`. Avant de voir leurs noms et fonctions, voici le principe de codage des données que vous pouvez leur affecter.

Par défaut, tous les éléments qu'ils contrôlent fonctionnent sous une tension nominale de 1,2 volt. Chaque chiffre entier positif augmente cette tension de 0,025 volt tandis que chaque chiffre entier négatif diminue cette tension de 0,025 volt. Ainsi, si dans le fichier `config.txt` vous écrivez `over_voltage=4`, vous augmenterez la tension d'alimentation de l'unité centrale du processeur de 0,1 volt (4 fois 0,025 volt) et vous la ferez donc monter à 1,3 volt. Les valeurs limites autorisées sont respectivement de -16, ce qui correspond à une diminution de tension de 0,4 volt à 8 ce qui correspond à une augmentation de 0,2 volt.

Ceci étant précisé, voici les paramètres sur lesquels vous pouvez agir.

over_voltage permet de définir la tension d'alimentation du cœur du processeur Broadcom qui équipe le Raspberry Pi.

over_voltage_sdram permet de définir la tension d'alimentation globale de la mémoire vive.

over_voltage_sdram_c permet d'intervenir sur la seule tension d'alimentation du contrôleur de la mémoire.

over_voltage_sdram_i permet quant à lui d'agir sur la seule tension d'alimentation des interfaces d'entrées/sorties de la mémoire.

Attention !

Plus encore que l'overclocking, l'augmentation de la tension d'alimentation est dangereuse pour les composants du Raspberry Pi. Si vous dépassez les limites permises pour les paramètres que nous venons de présenter, un indicateur interne au processeur Broadcom change d'état de manière irréversible et permet ensuite au fabricant de vous refuser toute application de la garantie en cas de panne.

3.5.4 Augmenter les performances sans risque (ou presque)

Les très (trop !) nombreux paramètres que nous venons de présenter, tant pour ce qui est des fréquences d'horloge que des tensions d'alimentation, ont de quoi vous laisser perplexe quant à l'augmentation des performances du Raspberry Pi. Sur lesquels faut-il agir ? Dans quelles proportions ? Est-ce dangereux pour le circuit ?

Afin de faciliter vos expérimentations en ce domaine, la dernière version de Raspbian Wheezy vous propose donc un certain nombre de valeurs prédéfinies permettant d'augmenter les performances de votre machine un peu, moyennement, beaucoup et même de la passer en mode « turbo ». Il n'est pas dit bien sûr que ces valeurs fonctionnent toutes sur votre Raspberry Pi, puisque l'overclocking n'est jamais une opération au résultat garanti, mais au moins les groupes de valeurs proposées sont-ils cohérents. Qui plus est, vous n'avez même pas à éditer `config.txt` pour les utiliser.

En mode console ou sous `LXTerminal`, il suffit de saisir à nouveau la commande :

```
pi@raspberrypi ~ $ sudo raspi-config
```

et de mettre en surbrillance la ligne **overclock**, de valider par **Entrée** et, après avoir accepté par un clic sur **OK** le message d'avertissement, de consulter le tableau visible figure 3.15.

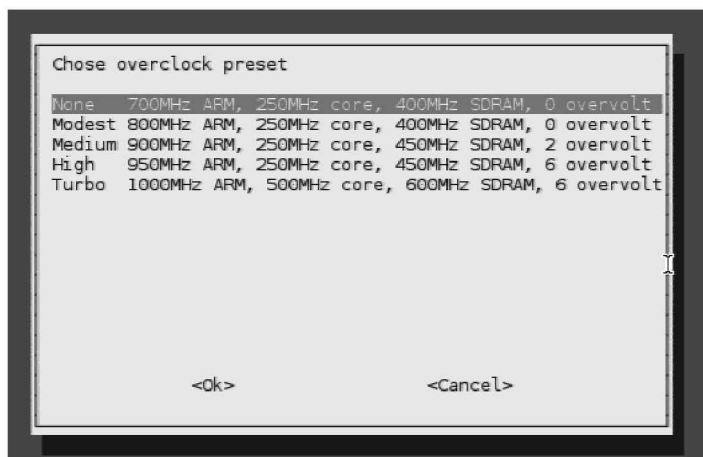


Figure 3.15 – Les différentes options d'overclocking proposées par `raspi-config`.

Celui-ci propose, outre la première ligne qui correspond à l'état normal du Raspberry Pi, quatre groupes de paramètres d'accélération matérielle baptisés : Modest, Medium, High et Turbo. Le premier ne fait pas usage de l'augmentation des tensions d'alimentation tandis que les trois suivants y font appel de manière de plus en plus agressive.

Il vous suffit donc de choisir le mode désiré, de valider par **Entrée** et de redémarrer le Raspberry Pi puisque les informations de configuration ne sont prises en compte que lors de son démarrage.

3.5.5 Si le Raspberry Pi devient instable

Si les valeurs choisies, que ce soit par édition directe de `config.txt` ou en utilisant `raspi-config`, conduisent à un fonctionnement instable, il suffit d'effacer les lignes correspondantes dans le fichier `config.txt` pour que les valeurs par défaut soient rétablies au démarrage suivant.

Cet effacement peut être réalisé avec `nano` si vous avez procédé à une édition directe du fichier, ou en relançant `raspi-config` en sélectionnant une valeur d'accélération moins forte, voire aucune (`none`), si votre Raspberry Pi est du genre paresseux !

Et si votre Raspberry Pi ne fonctionne plus du tout pour vous permettre d'éditer ce fichier ou de lancer `raspi-config`, rassurez-vous, la partition contenant le répertoire `/boot` est une partition FAT, parfaitement reconnue par Windows. Il suffit donc d'éditer le fichier `config.txt` sur n'importe quel PC muni d'un lecteur de carte SD pour revenir à la normale.

En revanche, si votre Raspberry Pi fonctionne ou semble fonctionner normalement après avoir choisi une des valeurs d'accélération, vous pouvez avoir envie de surveiller son fonctionnement en contrôlant, par exemple, sa température. C'est parfaitement possible avec l'environnement graphique LXDE en ajoutant des éléments à la barre des tâches en bas de l'écran. Pour cela, faites un clic droit dessus et sélectionnez **Ajouter/Enlever des éléments au tableau de bord**.

Cliquez alors sur le bouton **Ajouter** afin d'afficher la liste de tous les paramètres qu'il est possible d'afficher dans cette barre des tâches comme indiqué figure 3.16. Sélectionnez par exemple **Moniteur de la température** pour faire afficher la température du circuit Broadcom dans la barre des tâches. Notez également qu'un double-clic sur ce paramètre fait ouvrir la fenêtre présentée figure 3.17 qui propose deux valeurs d'alertes en cas de température excessive. Nous vous conseillons de conserver ces valeurs, quitte même à les diminuer de 5 °C par mesure de sécurité, mais en aucun cas de les augmenter car il y va de la vie du processeur Broadcom de votre Raspberry Pi.

Avec cet outil, et pour peu bien sûr que vous travaillez sous LXDE, vous pouvez donc vous assurer que malgré le taux d'accélération que vous avez choisi, votre Raspberry Pi ne risque pas la surchauffe.

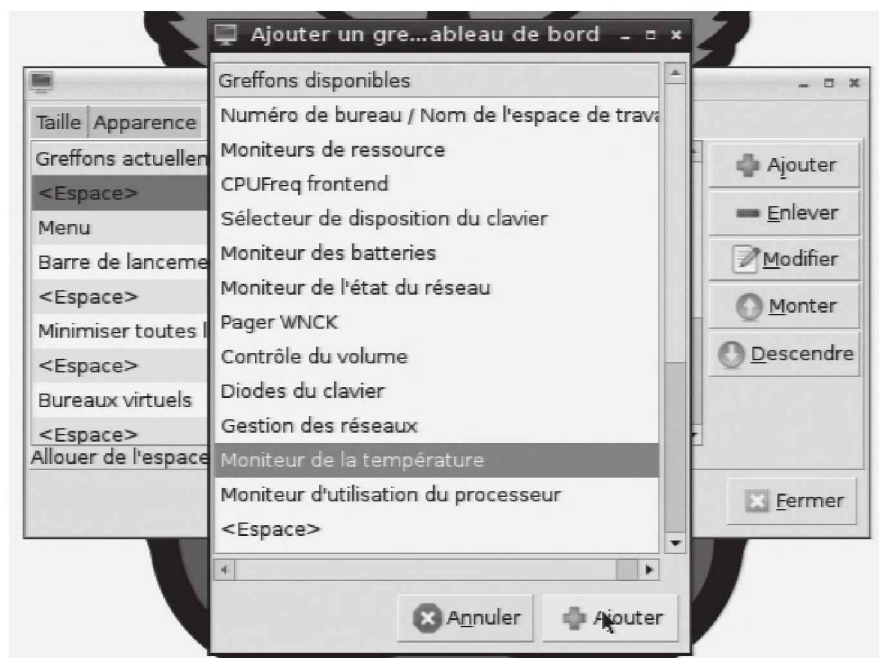


Figure 3.16 – Les nombreux paramètres que l'on peut faire afficher dans la barre des tâches.



Figure 3.17 – La surveillance de température dispose de deux seuils d'alerte réglables.

4 • BUREAUTIQUE, MEDIA CENTER ET SERVEUR WEB

Même si le Raspberry Pi n'est pas à proprement parler un micro PC, comme on le lit trop souvent dans la presse ou sur certains sites Internet, ne serait-ce que parce qu'il n'est pas conçu avec un processeur de la famille x86 d'Intel ou de la famille compatible de chez AMD, on peut néanmoins lui faire exécuter un certain nombre de tâches habituellement dévolues à un PC.

Il suffit pour cela que deux contraintes soient satisfaites : que le ou les programmes correspondants aient été portés sur la plateforme ARM qui équipe le Raspberry Pi, et que la puissance de calcul nécessaire ne soit pas trop importante au risque de décourager l'utilisateur par un excès de lenteur.

Nous vous proposons donc de découvrir dans ce chapitre trois domaines d'utilisation distincts du Raspberry Pi avec : la bureautique, la réalisation d'un *media center* ou HTPC, et enfin celle d'un serveur web.

4.1 Les différentes solutions et programmes bureautiques

La bureautique « de base » consiste généralement à pouvoir disposer d'un traitement de texte, d'un tableur, d'un logiciel de dessin ou de retouche d'images plus ou moins évolué et éventuellement d'un logiciel de présentation. Hormis l'application de retouche d'images, tous ces programmes se satisfont de ressources relativement modestes en termes de puissance de calcul et peuvent donc parfaitement être envisagées sur un Raspberry Pi pour peu que l'on choisisse les bons programmes. Pour y parvenir, nous vous proposons ci-après deux approches distinctes : le recours à des programmes indépendants d'une part et l'utilisation d'une « suite » bureautique d'autre part.

En ce qui nous concerne, nous n'avons pas de préférence pour l'une ou l'autre de ces solutions et nous vous conseillons donc de les essayer toutes les deux afin de choisir ensuite en fonction de votre ressenti. Ces essais sont d'autant plus faciles à réaliser que l'installation de ces programmes est véritablement un jeu d'enfant, qui ne demande que le temps nécessaire à leurs téléchargements respectifs, comme vous allez pouvoir le constater vous-même dès à présent.

Attention !

Avant de réaliser les manipulations présentées ci-dessous et afin de ne pas avoir à le répéter à chaque fois, veuillez dès à présent à mettre à jour la base de données des différents programmes disponibles au moyen de la commande `apt-get update` comme nous l'avons vu au § 2.4.1.

4.1.1 Utilisation de programmes indépendants

Le premier programme dont on a généralement besoin lorsque l'on fait de la bureautique est l'incontournable traitement de texte. Parmi les programmes indépendants existants, c'est-à-dire hors suite bureautique, nous vous proposons de faire appel à Abiword qui, même s'il n'est pas aussi complet que le mastodonte Word des dernières suites Office, dispose tout de même de la majorité des fonctions réellement utiles pour un tel programme.

Pour l'installer vous saisirez tout simplement, en mode ligne de commande depuis `lXTerminal` par exemple :

```
pi@raspberrypi ~ $ sudo apt-get install abiword
```

Ne partez pas boire un café en attendant que l'installation se déroule car, après quelques instants de recherche et l'affichage de l'emplacement que va occuper le programme sur la carte, il est nécessaire de donner votre accord pour cette installation en répondant `o` pour oui (ou `y` pour **y**es si vous n'avez pas localisé l'environnement comme nous l'avons expliqué au § 2.1.2) à la question posée, comme indiqué figure 4.1.

```

pi@raspberrypi: ~
Fichier  Édition  Onglets  Aide
pi@raspberrypi ~ $ sudo apt-get install abiword
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
abiword-common abiword-plugin-grammar abiword-plugin-mathview fonts-liberation
fonts-lyx libabiword-2.9 libcurl3-gnutls libfribidi0 libgdome2-0
libgdome2-cpp-smartOc2a libgsf-1-114 libgsf-1-common libgtkmathviewOc2a
liblink-grammar4 libloudmouth1-0 libmhash2 libots0 libraptor2-0 librasqal3 librdf0
libtl-5 libtelepathy-glib0 libtidy-0.99-0 libwmf0.2-7 libwpd-0.9-9 libwpg-0.2-2
libwps-0.2-2 libwv-1.2-4 libyajl2 link-grammar-dictionaries-en
Paquets suggérés :
raptor2-utils rasqal-utils librdf-storage-postgresql librdf-storage-mysql
librdf-storage-sqlite redland-utils
Les NOUVEAUX paquets suivants seront installés :
abiword abiword-common abiword-plugin-grammar abiword-plugin-mathview
fonts-liberation fonts-lyx libabiword-2.9 libcurl3-gnutls libfribidi0 libgdome2-0
libgdome2-cpp-smartOc2a libgsf-1-114 libgsf-1-common libgtkmathviewOc2a
liblink-grammar4 libloudmouth1-0 libmhash2 libots0 libraptor2-0 librasqal3 librdf0
libtl-5 libtelepathy-glib0 libtidy-0.99-0 libwmf0.2-7 libwpd-0.9-9 libwpg-0.2-2
libwps-0.2-2 libwv-1.2-4 libyajl2 link-grammar-dictionaries-en
0 mis à jour, 31 nouvellement installés, 0 à enlever et 210 non mis à jour.
Il est nécessaire de prendre 14,4 Mo dans les archives.
Après cette opération, 41,5 Mo d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer [O/n] ?

```

Figure 4.1 – L'installateur demande une confirmation quant à l'occupation du « disque » (en fait, la carte SD) qui va en résulter.

L'installation peut ensuite se poursuivre et peut nécessiter plusieurs minutes pendant lesquelles chaque étape de son déroulement est affichée à l'écran, pas toujours de façon très compréhensible pour le commun des mortels toutefois.

Une fois cette installation terminée, vous pouvez fermer **LXTerminal** et constater la présence, dans le menu des programmes, de la rubrique **Bureautique** contenant **Abiword**. Son lancement conduit à un écran similaire à celui présenté figure 4.2, écran qui ne doit pas vous dépayser si vous avez l'habitude d'utiliser un traitement de texte.

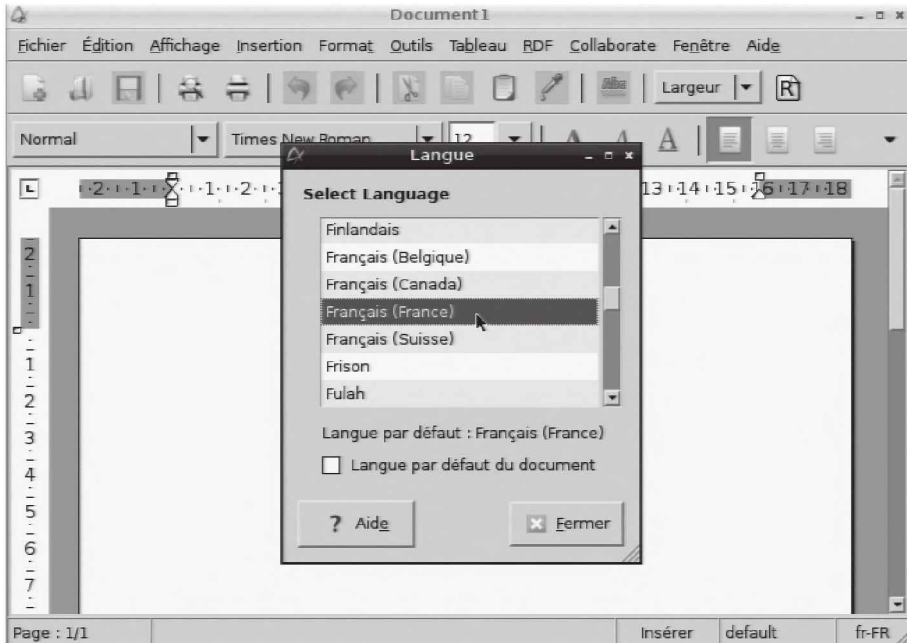


Figure 4.2 – Abiword ressemble à n'importe quel traitement de texte classique et connaît bien évidemment le français.

Si vous avez localisé correctement votre installation de Raspbian comme nous l'avons vu au § 2.1.2, Abiword doit naturellement être en français. Si ce n'est pas le cas, vous pouvez y parvenir en sélectionnant la rubrique **select Language** du menu **Tools**. Nous ne décrivons pas ici le mode d'emploi d'Abiword car d'une part, ce n'est pas le but de cet ouvrage et d'autre part, celui-ci ne présente aucune difficulté pour tout utilisateur déjà habitué à un traitement de texte.

Compagnon quasiment indispensable d'un traitement de texte, le tableur est le deuxième programme bureautique généralement installé ; celui qui est traditionnellement installé avec Abiword a pour nom **Gnumeric**.

Il s'installe aussi facilement qu'Abiword au moyen de la ligne de commande suivante saisie dans une fenêtre de **LXTerminal** :

```
pi@raspberrypi ~ $ sudo apt-get install gnumeric
```

La même confirmation relative à l'occupation mémoire que nous avons déjà vue pour Abiword est demandée avant que l'installation se poursuive sans encombre, en nécessitant ici encore, plusieurs minutes. Une fois celle-ci terminée, l'entrée **Gnumeric** est automatiquement ajoutée au menu **Bureautique** et permet de lancer ce tableur dont l'aspect, visible figure 4.3, est là aussi très classique.

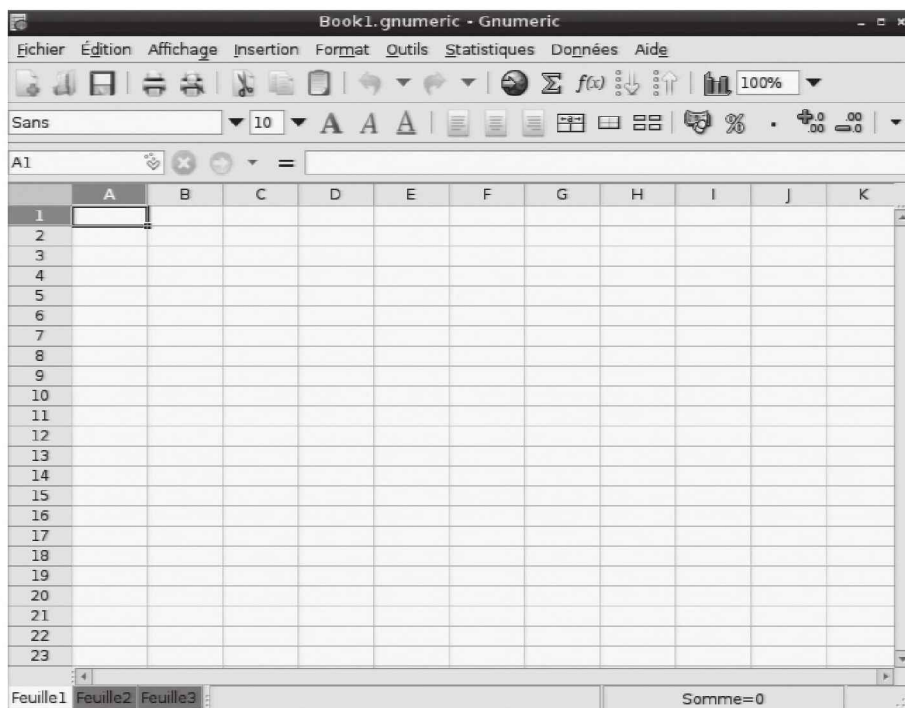


Figure 4.3 – Gnumeric n’a pas de quoi dépayser tout utilisateur habitué d’un tableur d’autant que son interface est, elle aussi, en français.

Indispensable complément de ce duo, le logiciel de dessin fait généralement partie du troisième programme à installer. Ici, nous vous proposons deux solutions selon les fonctionnalités dont vous voulez disposer.

Si de simples fonctions de dessin et de retouche, semblables à celles proposées par Microsoft Paint, vous suffisent, il est inutile d’installer une « usine à gaz », et un programme tel que Xpaint devrait vous satisfaire.

Comme ses homologues, il s’installe depuis une ligne de commande de **LXTerminal** avec un :

```
pi@raspberrypi ~ $ sudo apt-get install xpaint
```

Après la confirmation à laquelle vous êtes désormais habitué, le programme est installé et se trouve accessible au moyen de l’entrée **xpaint** du menu **Graphismes**. Son écran d’accueil, visible figure 4.4, confirme que son utilisation est au moins aussi simple que celle de son homologue de chez Microsoft.

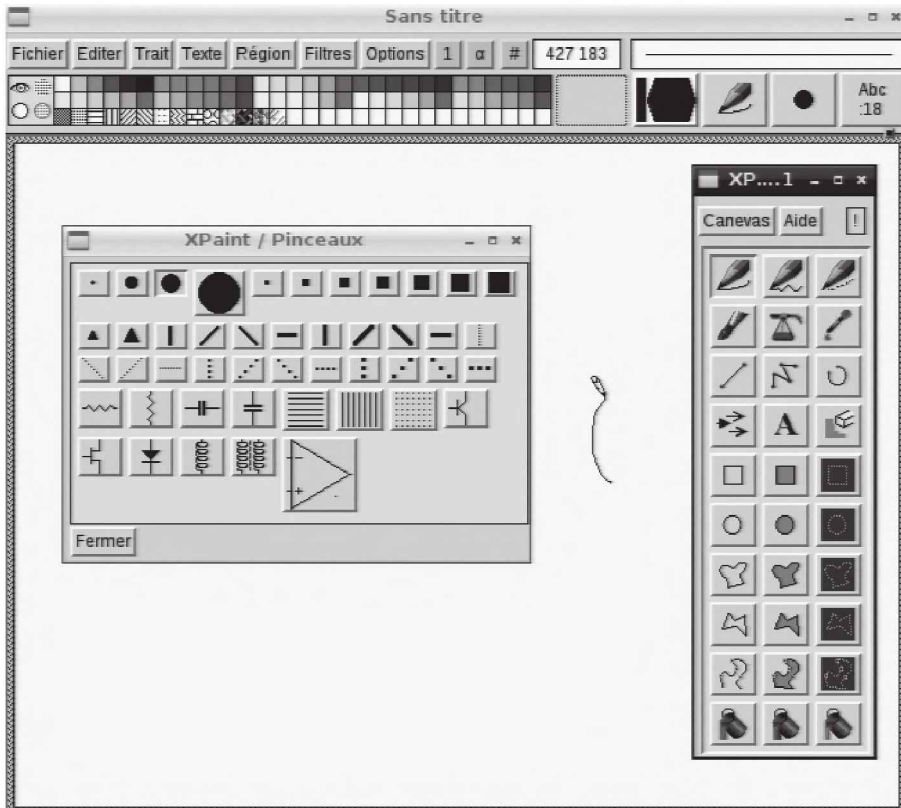


Figure 4.4 – Xpaint fait indubitablement penser à Microsoft Paint.

Notez toutefois la présence de quelques fonctions inconnues sous Paint avec, par exemple, l'existence de symboles prédéfinis tels par exemple ceux de composants électroniques visibles figure 4.4.

Bien qu'il soit suffisant pour réaliser ou retoucher des dessins simples dans le cas de nombre d'utilisations bureautiques, Xpaint est incapable par exemple de faire de la retouche de photos digne de ce nom. Si donc tel est votre besoin, sachez que vous pouvez parfaitement installer sur le Raspberry Pi le poids lourd du monde Linux en ce domaine : The Gimp.

Comme pour ce qui est des logiciels précédents, l'installation a lieu grâce au désormais célèbre :

```
pi@raspberrypi ~ $ sudo apt-get install gimp
```

Attention !

Bien que le programme s'appelle The Gimp, veillez à saisir seulement gimp dans la ligne de commande ci-dessus.

Plus encore que pour les programmes précédents, et après la demande de confirmation habituelle, cette installation prend du temps car le volume de données à

télécharger et le nombre d'éléments à configurer sont nettement plus importants que pour ces derniers. Cette attente en vaut toutefois la peine car cela permet ensuite de disposer sur le Raspberry Pi d'un programme qui, sans être aussi puissant que le célèbre Photoshop, offre tout de même des possibilités dignes de professionnels.

Une fois l'installation terminée, c'est fort logiquement dans le menu **Graphismes** qu'il faut aller chercher The Gimp sous l'appellation **Editeur d'image GIMP** alors que dans le menu **Autre** il s'appelle **The GIMP**.

Quoi qu'il en soit, son lancement conduit à l'affichage de pas moins de quatre fenêtres que nous avons fait en sorte de rendre toutes visibles figure 4.5.

Attention !

Lors de son premier lancement, certaines fenêtres de The Gimp peuvent ne pas être très visibles car elles se trouvent en bordure d'écran. Il suffit alors de les faire glisser avec la souris et/ou de les redimensionner afin de faciliter cette opération.

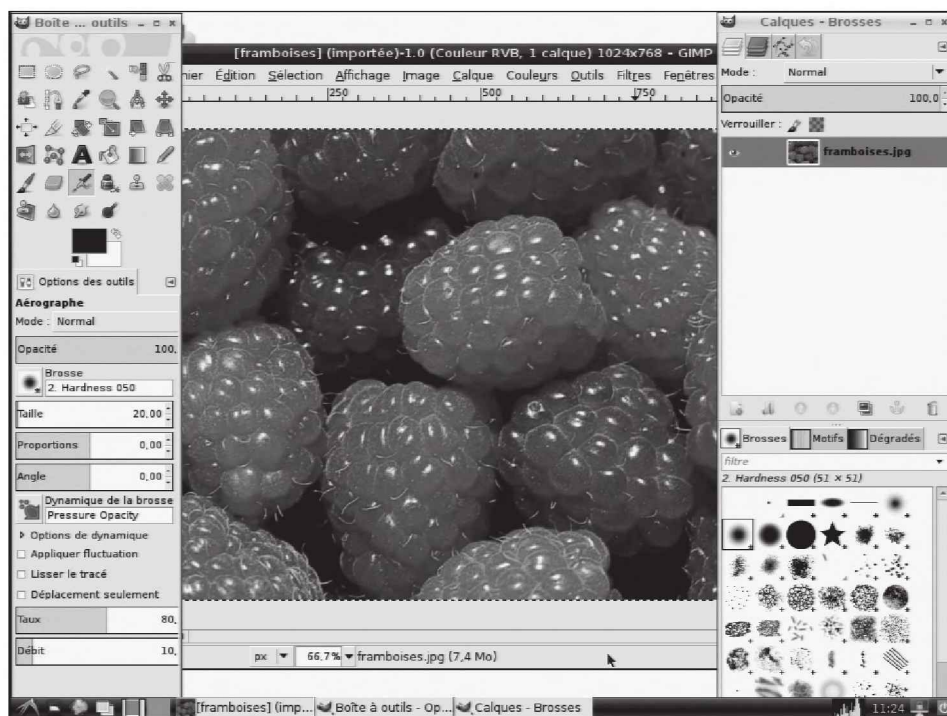


Figure 4.5 – Les nombreuses fenêtres de The Gimp lors de la retouche d'une magnifique photo de... framboises évidemment !

Cette version de The Gimp est une version complète du logiciel mais, afin d'économiser un peu d'espace sur la carte mémoire du Raspberry Pi, son manuel n'est ni téléchargé, ni installé. De ce fait, l'accès à l'aide, que ce soit par **Aide** ou via la touche **F1** fait afficher un message vous proposant de consulter ce manuel en ligne, ce qui est bien entendu parfaitement possible depuis le Raspberry Pi.

Notez aussi que, vu ses possibilités, l'utilisation de The Gimp est relativement complexe mais que de très nombreux tutoriels sont disponibles à son sujet sur Internet. Une petite recherche avec votre Google préféré devrait vous donner l'embarras du choix.

Attention !

La manipulation et le traitement d'images restent des opérations très gourmandes en ressources machine. Ne vous attendez donc pas à ce que votre « petit » Raspberry Pi soit aussi véloce que votre PC de bureau survitaminé à coup de processeur double ou quadruple cœur.

4.1.2 Utilisation d'une suite bureautique

Même si les programmes que nous venons de présenter donnent satisfaction, chacun dans leurs domaines, ils restent des programmes indépendants et on peut donc avoir envie de leur préférer une suite, à la manière du célèbre Office de Microsoft ; suite au sein de laquelle les différents programmes qui la composent sont donc parfaitement intégrés.

De plus, si cette suite est compatible avec les formats de fichiers utilisés par la suite Office, ce n'est que mieux puisque cela facilite ensuite les échanges sans nécessiter les fastidieuses opérations de conversions de formats, pas toujours couronnées de succès d'ailleurs. Et bien cette suite existe dans le monde Linux et peut en outre être installée sur le Raspberry Pi sous réserve de disposer d'au moins 350 Mo d'espace disponible sur la carte SD.

Si donc vous possédez une carte SD de taille supérieure aux 2 Go minimum demandés par Raspbian Wheezy, mais que vous n'avez pas encore procédé à l'augmentation de la taille des partitions de base, rendez-vous vite au chapitre 3.3 pour y procéder, faute de quoi l'installation de cette suite serait impossible.

Dérivée de l'ancienne suite OpenOffice dont le développement est aujourd'hui abandonné, la suite que nous vous proposons d'installer a pour nom LibreOffice et elle s'installe donc tout naturellement avec un désormais classique :

```
pi@raspberrypi ~ $ sudo apt-get install libreoffice
```

Attention !

Cette installation est très longue car, outre le fait qu'il faille télécharger près de 300 Mo de données, la mise en place et la configuration des fichiers constitutifs des programmes de la suite sollicite beaucoup le Raspberry Pi. Comptez donc près de vingt minutes pour la mener à bien.

Une fois l'installation terminée, LibreOffice est opérationnelle mais, si vous la lancez telle quelle, son interface sera en anglais. Si cela vous convient, tant mieux mais si vous préférez la langue de Molière, il faut lui ajouter le paquet français au moyen de la commande suivante :

```
pi@raspberrypi ~ $ sudo apt-get install libreoffice-l10n-fr
```

Si vous avez correctement localisé votre version de Raspbian Wheezy comme nous l'avons vu au § 2.1.2, le fait d'installer ce paquet doit passer automatiquement LibreOffice en français comme vous pouvez le constater à l'examen de la figure 4.7.

Si vous n'avez pas localisé votre version de Raspbian Wheezy, et une fois ce complément d'installation réalisé, il faut mettre la suite en français manuellement. Pour ce faire, lancez l'accueil de LibreOffice en cliquant sur **LibreOffice** dans le menu **Bureautique**.

Sur l'écran qui s'affiche alors, ouvrez le menu **Tools** et choisissez la rubrique **Options** puis, dans cette dernière, sélectionnez dans le volet gauche de la fenêtre qui est alors affichée la ligne **Language Settings** et faites déployer la branche correspondante en cliquant sur le signe **+**. Vous pouvez alors cliquer sur **Languages**, ce qui conduit à l'affichage visible figure 4.6 sur lequel il ne vous reste plus qu'à faire apparaître **Français** dans toutes les boîtes où figure l'anglais. Validez ensuite par **OK** puis, comme vous en informe l'écran d'avertissement, fermez LibreOffice pour que ces modifications soient prises en compte.

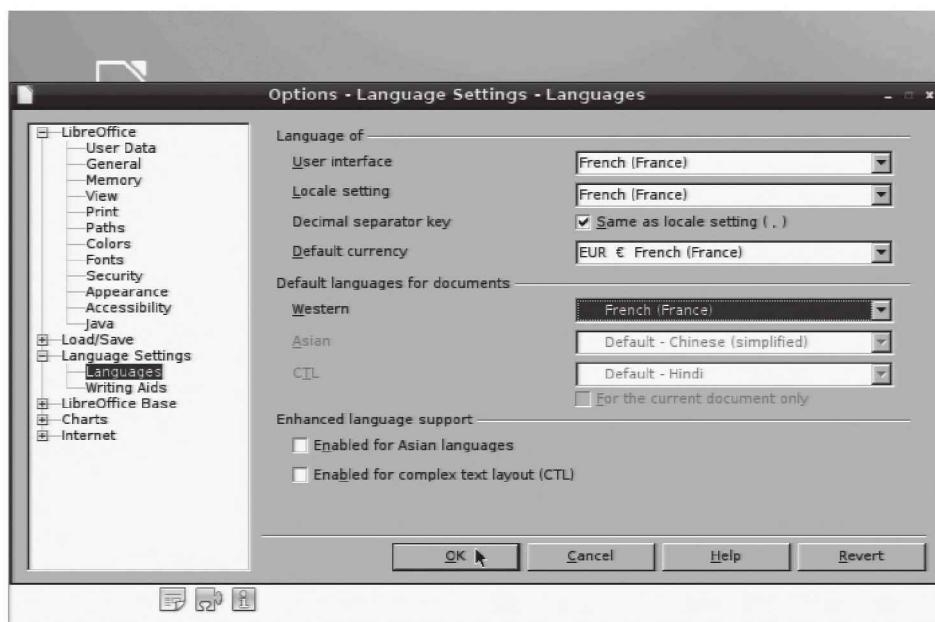


Figure 4.6 – Une fois le paquet « langue française » installé, et si votre version de Raspbian n'a pas été localisée, il faut mettre LibreOffice en français manuellement.

Si vous relancez LibreOffice, vous constaterez alors que son interface est bien en français comme le confirme la figure 4.7.



Figure 4.7 – LibreOffice est désormais en français.

Comme vous pouvez le constater sur cette figure, LibreOffice est bien une suite bureautique complète similaire dans son principe à Microsoft Office. On y trouve en effet par défaut un traitement de texte, un tableur, un logiciel de dessin, un gestionnaire de base de données, un logiciel de présentation et même un éditeur de formules mathématiques.

Comme le suggèrent les autres icônes présentes sur cet écran, il est possible de télécharger des modèles pour tous ces programmes (icône **Modèles**) ou bien encore d'ajouter des modules complémentaires (icône en forme de pièce de puzzle en bas de l'écran).

Une aide très complète et en français est également accessible via le site Internet français consacré à LibreOffice auquel vous pouvez accéder directement en cliquant sur l'icône contenant un **i** située également en bas de l'écran.

Arrivés à ce stade de notre installation, nous vous laissons le soin de découvrir cette suite qui ne démerite absolument pas vis-à-vis de Microsoft Office, pour un prix défiant toute concurrence puisque LibreOffice est totalement gratuite.

Et si vous prenez goût à LibreOffice, sachez qu'il existe une version identique pour les PC fonctionnant sous Windows. C'est un bon moyen de ne pas être dépaycé en passant du PC au Raspberry Pi, à moins que ce ne soit l'inverse.

LibreOffice ne comportant pas de programme de traitement d'image digne de ce nom, n'hésitez pas à installer The Gimp, comme nous l'avons vu au § 4.1.1 afin de compléter utilement cette suite.

4.1.3 Une fois que le choix est fait

Bien que nous ayons déjà évoqué la procédure à utiliser au chapitre 2, il nous semble utile de rappeler ici comment procéder pour désinstaller un programme et supprimer tous les fichiers qui lui sont associés car, une fois que vous aurez fait votre choix parmi les solutions précédentes, ce besoin se fera certainement sentir afin de ne pas occuper inutilement de la place sur votre carte SD.

Rappelons donc que, pour désinstaller un programme, il suffit de saisir la commande :

```
pi@raspberrypi ~ $ sudo apt-get remove nom du programme
```

Cette façon de procéder laisse toutefois en place les éventuels fichiers de configuration de ce dernier, bien utiles en cas de ré-installation ultérieure, alors que, si vous voulez vraiment tout effacer, il faut remplacer la ligne précédente par :

```
pi@raspberrypi ~ $ sudo apt-get purge nom du programme
```

4.2 Partage de fichiers via le réseau

Pouvoir utiliser son Raspberry Pi comme machine bureautique, avec un format de fichier compatible avec Microsoft Office de surcroît, c'est intéressant ; encore faut-il que les fichiers traités par ce dernier puissent être facilement partagés ou échangés avec d'autres machines.

Bien sûr, il est toujours possible de connecter une clé USB ou un disque dur externe au Raspberry Pi afin et de les utiliser pour déplacer les fichiers d'une machine à une autre, mais c'est une solution assez peu pratique, surtout lorsque l'on sait que le Raspberry Pi est raccordé au réseau.

Nous vous proposons donc, dans les quelques lignes qui suivent, de vous montrer comment installer sur votre Raspberry Pi un programme de partage de fichiers afin que vous puissiez, au travers de votre réseau et depuis n'importe quelle machine qui lui est raccordée, échanger des fichiers sans restriction avec votre Raspberry Pi.

Le programme de partage utilisé est universellement connu dans le monde Linux puisqu'il s'agit de Samba. Son installation est aussi simple que celle des programmes bureautiques précédents et a lieu comme d'habitude, en ligne de commande sous **LXTerminal** grâce à :

```
pi@raspberrypi ~ $ sudo apt-get install samba samba-common-bin
```

Après l'habituelle acceptation de la poursuite de l'installation à laquelle vous êtes habitué, il ne faut que quelques minutes pour que le programme de partage de fichiers soit installé.

Lorsque c'est terminé, il faut éditer le fichier `smb.conf` contenu dans `/etc/samba` afin de fournir à samba quelques informations relatives au réseau auquel le Raspberry Pi est raccordé, ainsi que concernant les utilisateurs autorisés à se connecter à distance au Raspberry Pi.

Pour cela nous allons utiliser l'éditeur nano et saisir la ligne suivante sous **LXTerminal** :

```
pi@raspberrypi ~ $ sudo nano /etc/samba/smb.conf
```

Au début du fichier, dans la rubrique **## Browsing/Identification ###**, corrigez la ligne :

```
workgroup = WORKGROUP
```

par :

```
workgroup = le nom du groupe de travail donné à votre réseau.
```

Si vous voulez vous connecter à un réseau constitué de machines sous Windows, notez que ce dernier utilise aussi **workgroup** comme nom par défaut. Cette ligne n'est donc à corriger que si vous avez modifié le nom de ce groupe de travail Windows.

Continuez à parcourir ce fichier vers le bas jusqu'à arriver à la rubrique **##### Authentication #####** et enlevez le symbole dièse (#) au début de la ligne :

```
# security = user
```

Cela permet d'imposer à tous les utilisateurs potentiels désireux de se connecter à votre Raspberry Pi via le réseau de saisir le nom d'utilisateur (**pi**) et le mot de passe (**raspberrypi**) avant de pouvoir y parvenir.

Si l'on en reste à ce stade des modifications du fichier de configuration, vous pourrez lire les fichiers du Raspberry Pi depuis un autre ordinateur du réseau et les y copier, mais il vous sera impossible d'envoyer des fichiers au Raspberry Pi. Pour cela, il faut descendre un peu plus bas dans le fichier en cours d'édition et remplacer :

```
read only = yes
```

par :

```
read only = no
```

Validez ensuite l'édition de ce fichier par un **CNTRL O** suivi de **Entrée** puis de **CNTRL X** pour fermer nano. Puis, afin de faire prendre en compte ce fichier par le Raspberry Pi, il faut arrêter puis redémarrer samba, ce qui s'obtient tout simplement avec la ligne de commande suivante :

```
pi@raspberrypi ~ $ sudo /etc/init.d/samba restart
```

Il faut ensuite informer samba que l'utilisateur **pi** est autorisé à y faire appel, ce qui s'obtient au moyen de :

```
pi@raspberrypi ~ $ sudo smbpasswd -a pi
```

À la suite de quoi Samba va vous demander de saisir à deux reprises le mot de passe de l'utilisateur `pi` (`raspberrypi` en l'occurrence). Dès cet instant, votre Raspberry Pi doit être visible sur le réseau et il est alors possible d'y accéder depuis un PC sous Windows par exemple, comme le confirme bien la figure 4.8 sur laquelle on peut voir le contenu du répertoire `pi` à partir de l'explorateur Windows ouvert sur un PC connecté sur le même réseau.

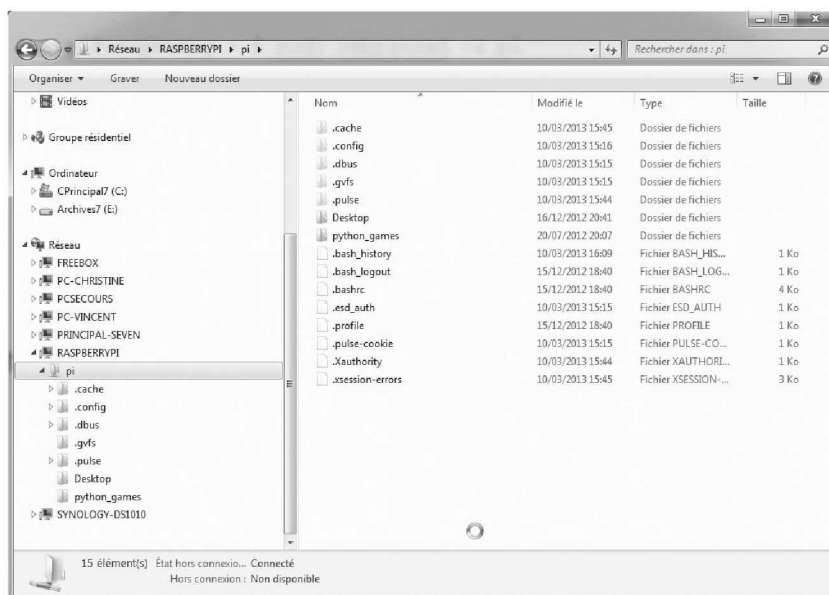


Figure 4.8 – Le contenu du répertoire `pi` du Raspberry Pi vu depuis un PC.

Attention !

Si, lors de votre première tentative de connexion au Raspberry Pi depuis un PC il s'avérait impossible d'y accéder, utilisez le navigateur Internet du PC avec l'adresse suivante `\\RASPBERRY\pi`. Vos nom et mot de passe de connexion vous seront alors demandés (`pi` et `raspberrypi`) et l'accès au Raspberry Pi sera ensuite possible tout à fait normalement grâce à l'explorateur de fichiers de Windows.

Votre Raspberry Pi fait désormais partie intégrante de votre réseau et vous pouvez librement échanger des fichiers dans les deux sens sans restriction, et récupérer ainsi par exemple le travail bureautique que vous avez réalisé au moyen des logiciels installés précédemment.

4.3 Un serveur avec votre Raspberry Pi

Dès lors que le Raspberry Pi est connecté au réseau, et même si son processeur n'est pas très puissant, il est parfaitement possible de le transformer en serveur, surtout si vous n'envisagez pas de trop nombreuses connexions simultanées sur ce dernier. Outre le fait de pouvoir constituer ainsi un serveur Internet « classique », il lui sera

également possible, par exemple, de transmettre à distance l'état de capteurs ou bien encore de commander à distance des actionneurs que vous lui aurez connectés.

Cette transformation du Raspberry Pi en serveur est d'autant plus facile que la majorité des serveurs Internet actuels fonctionnent avec Apache qui est un programme tournant naturellement sous Linux. Cette seule installation d'Apache, si elle permet d'afficher des pages html statiques, est toutefois insuffisante si vous souhaitez faire des sites Internet interactifs, aussi allons-nous également installer sur notre Raspberry Pi de quoi traiter le langage php, avec php dans sa version 5, ainsi que le classique gestionnaire de bases de données de l'environnement Linux : mysql.

4.3.1 Installation des logiciels nécessaires

Après la traditionnelle mise à jour du cache par un `apt-get update` que vous commencez à bien connaître, vous pouvez procéder à l'installation de tous ces programmes au moyen de la seule et unique ligne de commande ci-dessous saisie sous **LXTerminal** :

```
pi@raspberrypi ~ $ sudo apt-get install apache2 php5 mysql-server
phpmyadmin
```

Une fois la validation désormais classique acceptée en répondant `o` à la question posée, ou `y` si votre installeur est en anglais parce que vous n'avez pas localisé votre distribution, l'installation se poursuit jusqu'à ce qu'apparaisse la fenêtre demandant un mot de passe pour l'administrateur du gestionnaire de bases de données. Même s'il est possible de passer outre, nous vous recommandons de saisir un tel mot de passe par mesure de sécurité.

La fenêtre visible figure 4.9 est affichée peu de temps après afin de vous demander quel serveur reconfigurer pour utiliser phpMyAdmin. Sélectionnez **apache2** en frappant sur la barre d'espace lorsque la surbrillance est placée sur la ligne correspondante et validez par `ok`.

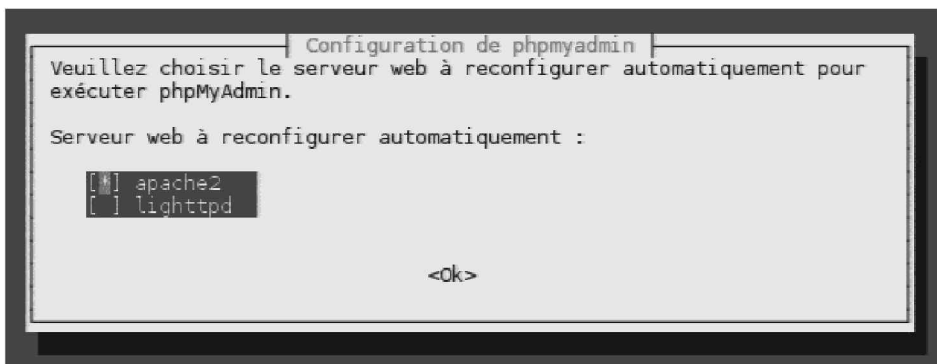


Figure 4.9 – Validation du serveur à utiliser pour phpMyAdmin.

L'installation peut alors se poursuivre, ce qui peut nécessiter plusieurs dizaines de minutes jusqu'à l'apparition de la fenêtre visible figure 4.10 que nous vous conseillons d'accepter en validant le oui présélectionné, sauf si vous maîtrisez la gestion de bases de données au moyen de phpMyAdmin.

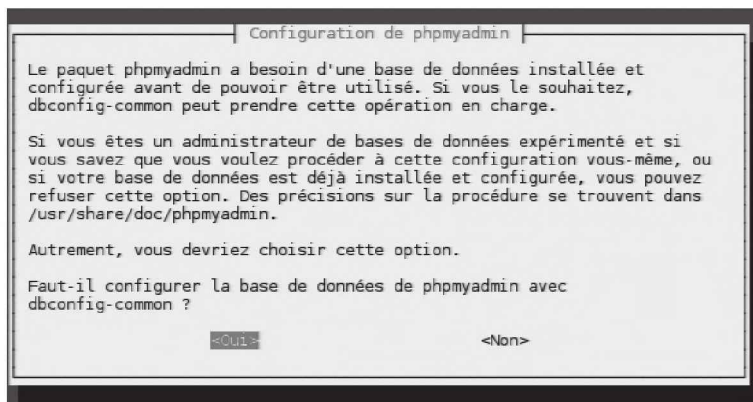


Figure 4.10 – Validez la proposition par défaut, sauf si vous maîtrisez phpMyAdmin.

Saisissez ensuite le mot de passe de l'administrateur de la base de données et laissez la configuration de cette dernière se poursuivre jusqu'à la fin de l'installation complète. Le serveur Apache fonctionnant en tâche de fond dès la fin de son installation, il vous est possible de le tester très facilement depuis n'importe quelle autre machine connectée à votre réseau. Saisissez pour cela dans la barre d'adresse de son navigateur l'adresse IP de votre Raspberry Pi, ce qui doit avoir pour effet de faire afficher la page presque vide visible figure 4.11.

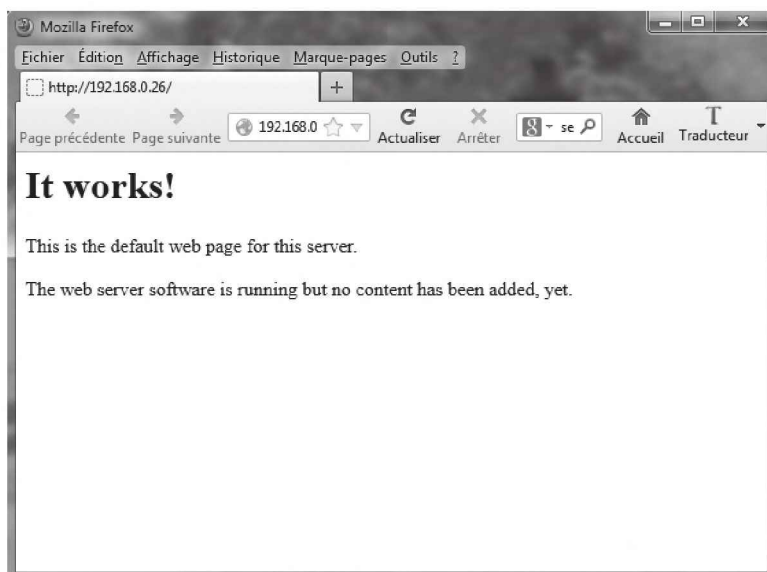


Figure 4.11 – Le serveur Apache fonctionne. Il ne reste plus qu'à créer des pages html.

Si vous voulez aussi vous assurer du bon fonctionnement de php et du gestionnaire de bases de données, il vous suffit de renouveler la manipulation précédente en ajoutant tout simplement `/phpmyadmin` après l'adresse IP du Raspberry Pi, ce qui doit conduire à l'affichage visible figure 4.12.

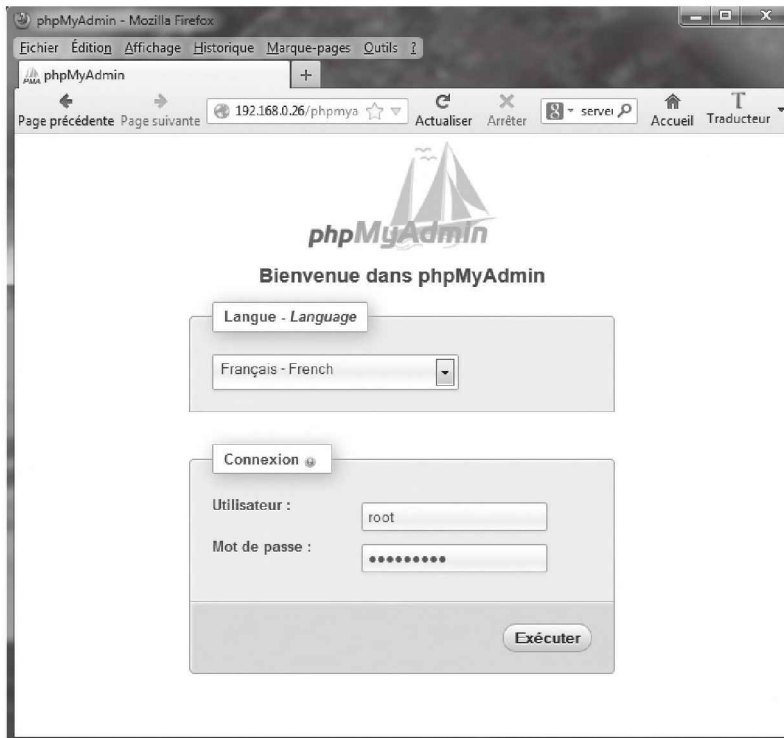


Figure 4.12 – phpmyadmin fonctionne également sans problème.

Pour vous connecter à l'interface d'administration, il suffit de saisir comme nom d'utilisateur `root` et comme mot de passe celui que vous avez défini lors de la procédure d'installation vue précédemment. Dès lors, l'accès à la page d'administration visible figure 4.13 doit avoir lieu sans difficulté.

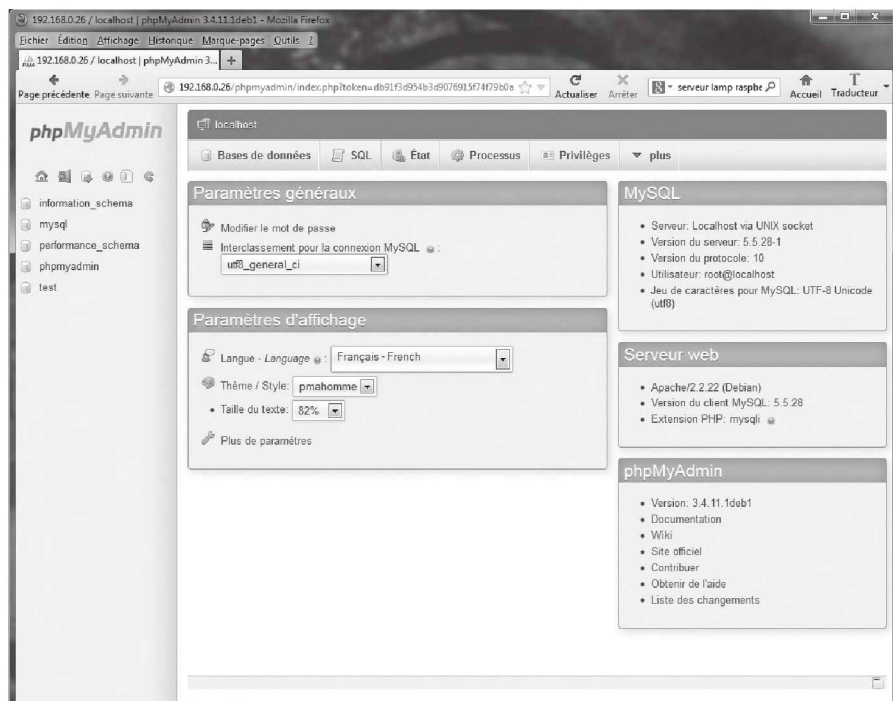


Figure 4.13 – La page d’administration de phpmyadmin.

4.3.2 Utilisation du serveur

Si les tests précédents sont concluants, et il n’y a aucune raison pour qu’ils ne le soient pas, votre serveur est utilisable.

Les pages constituant votre site sont à placer dans le répertoire `/var/www` dans lequel vous pouvez d’ailleurs trouver par défaut la page `index.html` qui correspond à l’affichage que nous avons vu figure 4.11.

4.4 Le Raspberry Pi en tant que media center

Même si le processeur Broadcom qui anime le Raspberry Pi n’est pas un foudre de guerre, ses concepteurs ont eu la bonne idée de le doter d’un GPU (processeur graphique) particulièrement musclé, capable de lire de la vidéo haute définition en 1080p sans aucune difficulté. De là à ce que germe dans la tête de certaines personnes l’idée de transformer le Raspberry Pi en *media center*, il n’y avait qu’un pas qui a aujourd’hui été franchi, et même plutôt deux fois qu’une.

Nous allons donc découvrir maintenant comment procéder pour, dans un premier temps, lire tout simplement des fichiers audio sur votre Raspberry Pi, mais sans le transformer en *media center* pour autant et, dans un second temps, comment le transformer réellement en *media center*, ou HTPC si vous préférez, ce qui est l’acronyme de *Home Theater PC*, même si le Raspberry Pi n’est bien sûr pas un PC !

4.4.1 De la musique en travaillant

Si vous voulez simplement pouvoir lire quelques fichiers MP3 avec votre Raspberry Pi pendant que vous travaillez par exemple, point n'est besoin de le transformer en HTPC pour autant. Un simple lecteur très basique tel que `moc` suffit pour cela, et ce d'autant qu'il consomme peu de ressources.

Son installation a lieu avec un désormais classique :

```
pi@raspberrypi ~ $ sudo apt-get install moc
```

Une fois l'installation menée à bien, et comme le lecteur est vraiment basique, il n'est accessible qu'en mode ligne de commande au moyen de :

```
pi@raspberrypi ~ $ mocp
```

Attention !

Même si le logiciel s'appelle `moc`, il faut bel et bien saisir `mocp` pour le lancer. Il ne s'agit pas d'une faute de frappe !

Ceci a pour effet d'ouvrir l'interface très simple visible figure 4.14. La partie gauche de la fenêtre est un explorateur de fichiers dans lequel vous rechercherez les fichiers MP3 à lui faire lire, tandis que la partie droite de cette fenêtre permet, si vous le souhaitez, de constituer une liste de lecture en y déplaçant les fichiers de votre choix.



Figure 4.14 – L'interface de `moc`, présentée ici en train de lire un fichier MP3, est des plus dépouillées.

Pour lancer la lecture d'un fichier, il suffit d'amener la surbrillance sur son nom et de valider par **Entrée**. Par ailleurs, toutes les commandes de `moc` ont lieu au clavier et, pour en découvrir la liste et la syntaxe, fort simple au demeurant, il suffit de saisir `h` (comme `help`) depuis l'écran principal.

N'hésitez pas à parcourir cet écran, comme nous l'avons fait figure 4.15, car vous y découvrirez que ces commandes sont très nombreuses et permettent finalement de lire des fichiers MP3 dans d'excellentes conditions.



Figure 4.15 – Les nombreuses commandes de moc.

Ceci est d'autant plus intéressant que, même si vous fermez moc (commande `q` pour quitter) alors qu'il est en train de lire des fichiers, et que vous faites autre chose, que ce soit en mode console ou sous l'environnement graphique, la lecture des fichiers se poursuivra. Quand on vous disait qu'il était fait pour écouter de la musique en travaillant...

À tout instant, il est possible de relancer l'interface en saisissant à nouveau `mocp` depuis la ligne de commande.

4.4.2 Lorsque le son est récalcitrant

Par défaut, c'est-à-dire lorsque la distribution Raspbian Wheezy est utilisée sans modification, le son issu des différents logiciels, dont moc par exemple, est envoyé sur la prise HDMI. Cela fonctionne très bien si vous utilisez un téléviseur ou un moniteur équipé de haut-parleurs mais, si vous utilisez un moniteur informatique, généralement sans haut-parleur, et que vous désirez récupérer le son depuis le jack stéréo prévu à cet effet, vous n'obtiendrez rien.

Pour y parvenir, il faut désinstaller le module qui gère le son via la sortie HDMI et valider ensuite celui qui permet la sortie son en analogique via le jack stéréo. Pour cela, depuis l'interface en ligne de commande, saisissez la suite de commandes que voici :

```
pi@raspberrypi ~ $ sudo apt-get autoremove pulseaudio
pi@raspberrypi ~ $ sudo apt-get install alsa
pi@raspberrypi ~ $ sudo amixer cset numid=3 1
```

Le son devrait alors être disponible via la sortie jack stéréo, mais plus via la sortie HDMI. Si vous désirez ultérieurement valider à nouveau le son sur la sortie HDMI, mais ne plus en disposer à ce moment-là sur le jack stéréo, il conviendra de saisir en ligne de commande :

```
pi@raspberrypi ~ $ sudo apt-get install pulseaudio
pi@raspberrypi ~ $ sudo amixer cset numid=3 2
```

4.4.3 Un véritable media center (ou HTPC)

Un *media center*, ou HTPC puisque les deux appellations coexistent, est à l'origine un PC capable de gérer, avec un maximum de souplesse et de convivialité, tous vos médias, qu'ils soient musicaux ou vidéo. Le plus connu a longtemps été Windows Media Center, disponible depuis longtemps déjà sur les PC fonctionnant sous Windows, mais il montre aujourd'hui ses limites, tant en termes de performances que d'ergonomie.

Le monde du logiciel libre a en effet développé entre temps un produit baptisé XBMC, qui est un logiciel *media center* aux performances tout à fait remarquables. Le but de cet ouvrage n'étant pas d'en faire la promotion ni d'en décrire le mode d'emploi, très fourni eu égard à ses innombrables possibilités, nous vous renvoyons par exemple au site <http://www.passion-xbmc.org> pour découvrir toutes les informations le concernant.

Outre le fait d'appartenir au monde du logiciel libre, et donc d'être totalement gratuit et modifiable par l'utilisateur, XBMC est également multi plateforme et fonctionne ainsi, entre autres systèmes, sous Android, iOS, Windows, Linux et bien sûr sur notre Raspberry Pi.

Même s'il est possible d'installer XBMC sous la distribution Raspbian Wheezy, ce n'est pas à notre avis la meilleure des approches car son paramétrage, pour obtenir un fonctionnement irréprochable de XBMC, reste assez délicat et conduit parfois à des instabilités.

Il vaut beaucoup mieux faire appel à une des deux principales distributions Linux prévues à cet effet, intégrant nativement XBMC, qui permettent de disposer en quelques minutes d'un *media center* parfaitement opérationnel et tout aussi performant que ses homologues fonctionnant sur PC.

Ces deux distributions sont d'une part, **Raspbmc** disponible depuis le site de son auteur à l'adresse <http://www.raspbmc.com/>, auteur qui, cela mérite d'être salué, a fait un travail de portage remarquable alors qu'il a tout juste 19 ans au moment où ces lignes sont écrites et d'autre part, **XBMC openelec** disponible en téléchargement à cette adresse <http://openelec.tv/get-openelec/download>.

■ Installation de Raspbmc

Cette installation est un tout petit peu plus simple que celle de **XBMC openelec** car son auteur fournit avec l'image de la carte SD un installateur prêt à l'emploi fonctionnant sous Windows. Il suffit donc de télécharger le fichier prévu à cet effet,

disponible à cette adresse : <http://www.raspbmc.com/wiki/user/windows-installation/> et de le décompresser dans le répertoire temporaire de votre choix.

Exécutez ensuite le fichier `setup.exe`, non sans avoir au préalable placé la carte SD destinée au Raspberry Pi dans le lecteur de carte de votre PC. Cochez la case spécifiant que vous acceptez la licence comme indiqué figure 4.16, ainsi que celle correspondant à votre lecteur de carte, et validez par **Install**. L'installateur va alors formater partiellement votre carte SD et télécharger le logiciel nécessaire à la poursuite de l'installation depuis le Raspberry Pi à partir d'Internet, ce qui peut nécessiter quelques minutes selon la vitesse de votre connexion.

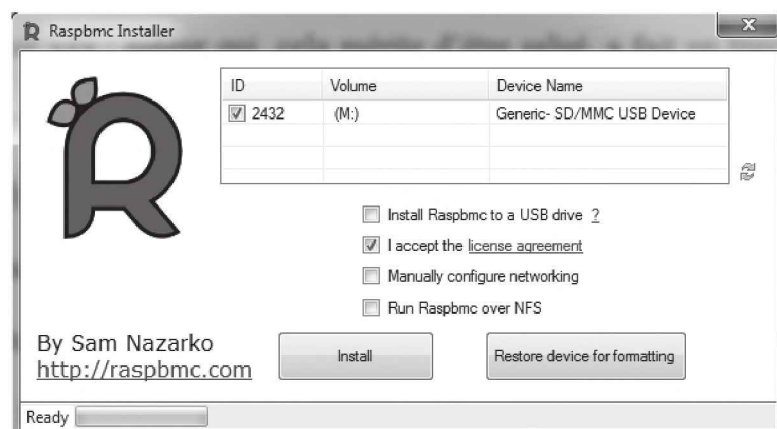


Figure 4.16 – L'installateur de Raspbmc se charge de la préparation de la carte SD.

Une fois l'opération terminée, insérez la carte dans votre Raspberry Pi et redémarrez-le en vous assurant qu'il est bien connecté à un réseau lui donnant accès à Internet. Le processus d'installation et de configuration de Raspbmc va alors se poursuivre de manière automatique et va nécessiter de 10 à 30 minutes, car c'est en fait à ce moment-là que la carte SD sera effectivement intégralement formatée et que sera téléchargée réellement l'image de Raspbmc.

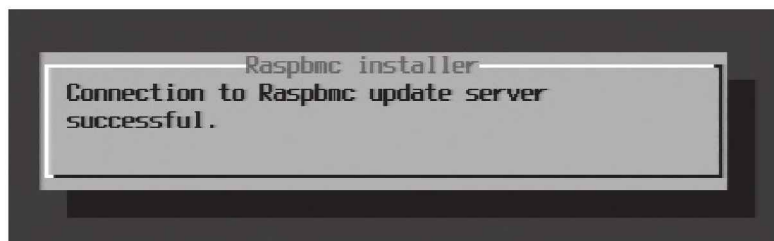


Figure 4.17 – La connexion au serveur de mise à jour de Raspbmc s'est bien passée.

Après le message de confirmation de bon établissement de la connexion au serveur de Raspbmc, visible figure 4.17, le téléchargement de l'image complète prend place avec une barre de progression... et une invitation à prendre un café vu le temps nécessaire.

Raspbmc est ensuite automatiquement installé et mis à jour sans nécessiter la moindre intervention manuelle de votre part. Plusieurs redémarrages ont lieu mais ils sont entièrement automatiques et, si votre écran semble se figer avec un R rouge fixe au centre de ce dernier, n'en croyez rien et patientez quelques dizaines de secondes.

XBMC sera ensuite mis à jour automatiquement et finira par être lancé avec, comme vous pouvez le constater figure 4.18, la fenêtre de changement de langue automatiquement ouverte sur laquelle il ne vous restera plus qu'à cliquer sur **français**.



Figure 4.18 – XBMC démarre en vous proposant de choisir la langue de son interface.

Remarquez que, lors du redémarrage ultérieur de XBMC une fenêtre s'affiche un court instant qui, outre le fait de vous dire de ne pas paniquer (sic !), vous propose d'accéder au mode ligne de commande en frappant sur la touche **Échap** à ce moment-là. Raspbmc est tellement bien configuré que vous n'aurez, en principe, pas besoin de faire appel à cette possibilité.

Arrivé à ce stade des opérations, XBMC est pleinement fonctionnel sur votre Raspberry Pi et un écran similaire à celui visible figure 4.19, doit apparaître.

■ Installation de XBMC Openelec

L'installation de **XBMC Openelec** adopte une philosophie différente mais n'est pas plus compliquée pour autant. Il vous faut juste faire appel à un logiciel tiers, que vous possédez déjà depuis le chapitre 1 puisqu'il s'agit de **Win32 Disk Imager**.

Le procédé va en effet consister à réaliser directement depuis un PC une image complète de la carte SD qui sera ensuite immédiatement opérationnelle dans le Raspberry Pi. Pour cela, il faut télécharger la dernière version de l'image à cette adresse : <http://openelec.thestateofme.com/>.

Décompressez-là dans le répertoire de votre choix et lancez ensuite **Win32 Disk Imager**, comme nous l'avons fait au chapitre 1 pour créer l'image de Raspbian

Wheezy, afin de créer cette fois-ci l'image de **XBMC Openelec** sur la carte SD que vous aurez pris soin de placer dans le lecteur de votre PC.

■ Attention !

Au moment où ces lignes sont écrites, et pour une raison indéterminée, il ne faut pas utiliser de carte SD de capacité supérieure à 8 Go sinon XBMC Openelec refusera de démarrer une fois la carte mise en place sur le Raspberry Pi.

L'interface sera bien évidemment mise en français comme ce fut le cas pour **Raspbmc**, ce qui conduira à l'affichage visible figure 4.19.

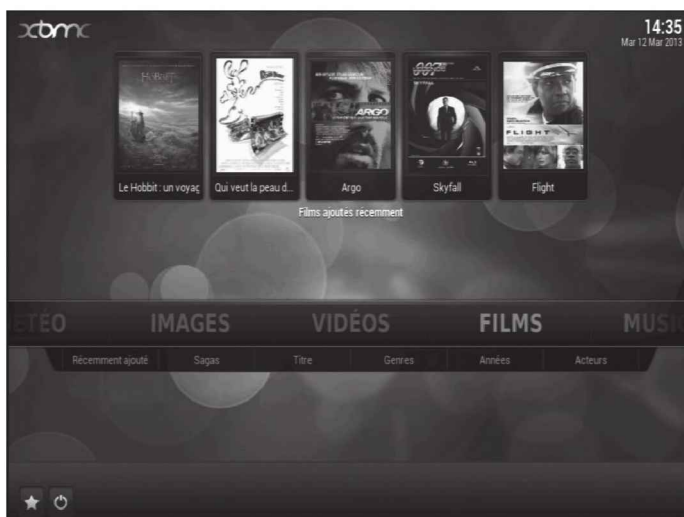


Figure 4.19 – Que ce soit avec Raspbmc ou avec XBMC Openelec, XBMC fonctionne parfaitement sur le Raspberry Pi.

4.4.4 Quelques conseils pour votre *media center*

Que ce soit avec **XBMC Openelec** ou **Raspbmc**, c'est toujours le *media center* XBMC qui fonctionne et vous pouvez donc utilement prendre connaissance des très nombreuses informations disponibles sur le site Passion XBMC, d'adresse <http://www.passion-xbmc.org>, cité précédemment. Il nous semble cependant utile de vous fournir quelques petites informations à propos de ce qui nous semble être la meilleure configuration pour un fonctionnement optimum ; configuration que d'ailleurs nous utilisons avec succès depuis que **Raspbmc** existe !

■ Sources audio et vidéo

La mémoire du Raspberry Pi étant très limitée, ce n'est évidemment pas là que vous allez stocker vos fichiers audio et vidéo, surtout si vous voulez visionner des films en HD dont la taille dépasse couramment les 7 à 10 Go. Une unité de stockage externe est donc indispensable.

Si un disque dur USB est une solution envisageable, cette interface montre ses limites, en termes de vitesse, lors du visionnage de films HD en 1080p. Nous vous conseillons donc de stocker vos fichiers audio, mais surtout vidéo sur un NAS (*Network Attached Storage*), c'est-à-dire sur un disque dur réseau. Il en existe d'excellents et très performants chez Synology par exemple. Non content de rendre ainsi ces fichiers accessibles à toutes les machines connectées à votre réseau, cela permettra à l'XBMC de votre Raspberry Pi de les lire sans aucun problème, même dans la résolution maximale de 1080p.

■ À propos du son

XBMC est capable de gérer les deux sorties son disponibles sur le Raspberry Pi : la sortie via la prise HDMI et celle via le jack audio. Si vous possédez un téléviseur récent ou, mieux, un amplificateur home-cinéma, nous vous conseillons de faire sortir le son par la prise HDMI car vous bénéficierez ainsi du Dolby Digital, ou bien encore du DTS pour peu que la piste son de votre film soit à ce format, tandis que la sortie via le jack ne pourra donner que de la stéréo, de piètre qualité de surcroît. Ce n'est pas XBMC qui est en cause mais bien le Raspberry Pi hélas.

■ Télécommandez votre media center

XBMC se commande très bien depuis la souris ou le clavier de votre Raspberry Pi mais, dans le salon à côté de la télé, cela ne fait pas très chic ! Il est fort heureusement possible de se passer de tout cela en faisant appel à une télécommande compatible Windows Media Center. Ces télécommandes, telle celle visible figure 4.20 par exemple, se trouvent très facilement sur Internet pour deux dizaines d'euros environ et il suffit juste d'en choisir une munie d'un récepteur infrarouge à connecter sur un port USB. Elle sera alors automatiquement reconnue par XBMC que vous pourrez piloter sans faire appel ni au clavier, ni à la souris.



Figure 4.20 – Une télécommande compatible Media Center et le récepteur infrarouge – USB IR 605.

Si vous disposez d'une télécommande universelle, telle les excellentes Logitech Harmony par exemple, ou tout autre modèle équivalent, vous pouvez vous contenter d'acheter le récepteur infrarouge – USB seul. Un modèle qui fonctionne très bien avec XBMC est l'IR 605 visible lui aussi figure 4.20.

Et si vous avez un smartphone, que ce soit un i « quelque chose » ou un modèle sous Android, sachez qu'il existe des applications qui le transforment en télécommande XBMC fonctionnant également ici sans problème.

■ Les deux licences à acheter

Tel qu'il est installé sur le Raspberry Pi, XBMC peut lire tous les formats de fichiers audio et vidéo... ou presque. En effet, pour faire quelques économies sur le prix de vente du Raspberry Pi, ses concepteurs n'ont pas acheté deux licences indispensables permettant de lire les vidéos aux formats VC1 et MPEG2 ce qui est très ennuyeux car, sans elles, adieu la lecture des fichiers de DVD ou de SVCD.

Pour y remédier, il existe fort heureusement une solution très simple qui consiste à acheter ces licences depuis la boutique Raspberry Store à l'adresse <http://www.raspberrypi.com/>. Rassurez-vous, elles coûtent à elles deux moins de 5 euros et vous sont livrées très rapidement par e-mail puisque ce sont en fait deux codes alphanumériques à placer dans le fichier `config.txt` qui se trouve à la racine de la carte SD du Raspberry Pi.

Éditez ce fichier `config.txt`, qui se trouve dans le répertoire racine de la carte SD, avec n'importe quel éditeur de texte tournant sur PC et ajoutez en fin de fichier les lignes suivantes :

```
decode_MPG2=xxxxxxx  
decode_WVC1=xxxxxxx
```

où `xxxxxxx` sont les codes alphanumériques des licences que vous avez achetées. Certaines versions du fichier `config.txt` contiennent même déjà ces lignes sous forme de commentaires, et il suffit juste d'effacer le symbole dièse (#) qui se trouve au début des lignes correspondantes et de remplacer les `xxxx` par les numéros « qui vont bien ».

5 • À LA DÉCOUVERTE DE PYTHON

Même si le Raspberry Pi a déjà de quoi vous satisfaire avec l'une des nombreuses applications étudiées dans les précédents chapitres, il peut également s'interfacer très facilement avec le monde extérieur grâce au connecteur mâle à 26 points qui équipe son circuit imprimé ; connecteur qui donne accès à certaines lignes d'entrées/sorties du processeur.

Nous verrons dans les chapitres qui suivent qu'il est possible de relier ces lignes à une multitude de composants externes, LED, poussoirs, claviers, afficheurs, etc., mais pour faire fonctionner tous ces éléments, il faudra écrire des programmes spécifiques et ces programmes, nous vous proposons de les écrire en Python, langage de programmation que vous allez découvrir dans ce chapitre.

Pourquoi Python et pas Basic, C ou C++, tous plus connus au moins du grand public ? Pour au moins quatre raisons : la première est que Python est un langage facile à apprendre par qui n'a jamais programmé, la seconde est que Python est un langage interprété et qu'il est donc très facile d'essayer immédiatement une ou plusieurs instructions dont on n'est pas sûr du comportement ; la troisième est que Raspbian Wheezy est livrée d'origine avec un environnement de développement Python et la quatrième enfin est que de nombreuses bibliothèques sont dès à présent disponibles sur Internet pour piloter les lignes d'entrées/sorties du Raspberry Pi en Python.

Python étant un langage très riche, surtout si l'on prend en compte les nombreuses bibliothèques complémentaires disponibles, nous n'allons pas, en un seul chapitre au nombre de pages nécessairement limité, pouvoir en faire une présentation exhaustive.

Nous avons cependant rédigé celle-ci de façon à ce que vous disposiez des connaissances de base nécessaires afin de vous permettre d'écrire vos premiers programmes Python et, surtout, de comprendre les quelques exemples des chapitres suivants.

5.1 L'environnement de travail

Afin de pouvoir écrire avec un maximum de confort des programmes en Python, il est souhaitable de disposer de ce que l'on appelle en anglais un IDE, pour *Integrated Development Environment* qui, dans le cas de Python, s'appelle IDLE ou IDLE 3 selon qu'il est fait référence à la version 2 de Python ou à la version 3. Dans la

mesure du possible, nous ferons référence à la version 3, sauf lorsque l'absence de certains modules nous obligera à revenir à Python en version 2. Cet environnement se lance tout simplement en double cliquant sur l'icône prévue à cet effet sur le bureau de Raspbian Wheezy ce qui conduit à l'affichage visible figure 5.1.

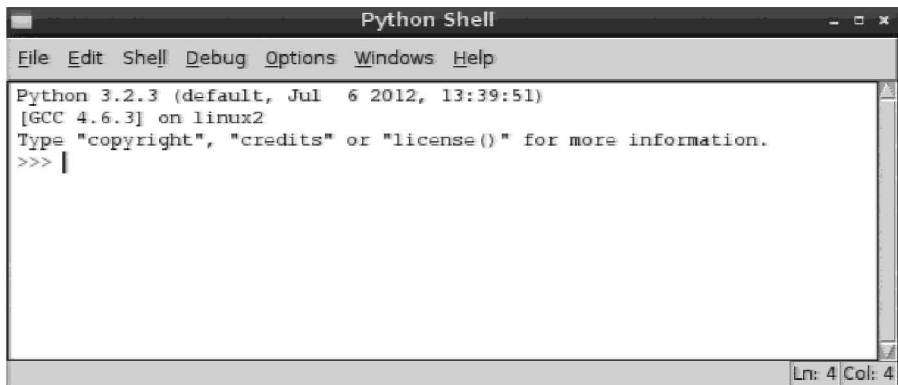


Figure 5.1 – L'écran d'accueil, ou shell, ou bien encore fenêtre de terminal de IDLE 3.

5.1.1 Interpréteur et compilateur

Si vous avez l'habitude de langages de programmation plus classiques, cet écran a de quoi vous surprendre quelque peu. En effet, Python est un langage interprété contrairement à de nombreux autres langages, le C notamment, qui sont des langages compilés. Quelle est la différence nous direz-vous ?

Dans un langage compilé, l'utilisateur écrit un programme et, lorsqu'il a fini, ou lorsqu'il veut l'essayer en cours de développement, il fournit ce programme à un compilateur qui le traduit alors dans son intégralité en langage « machine », qui est le seul langage compris par le processeur. Une fois ce programme traduit, et même si la traduction a pris du temps, le programme machine obtenu peut être très rapide à exécuter puisqu'il n'est constitué que d'instructions directement comprises par le processeur. En outre, si on l'exécute de multiples fois, il ne sera pas nécessaire de le compiler à nouveau puisque cela aura déjà été fait.

Dans un langage interprété, chaque ligne de programme est traduite au fur et à mesure en langage machine avant d'être exécutée, ce qui signifie que l'exécution d'un tel programme sera nécessairement beaucoup plus longue que celle d'un programme compilé puisqu'on fera, en quelque sorte, une compilation ligne par ligne à chaque fois. Et si le programme interprété doit être exécuté de nombreuses fois, et bien à chaque fois les lignes devront être traduites à nouveau.

En contrepartie, comme chaque ligne est traduite au fur et à mesure, il est possible d'essayer son programme pas à pas, ce qui s'avère très confortable surtout au début, lorsque l'on ne sait pas bien ce que va faire telle ou telle instruction par exemple.

Lorsque l'environnement IDLE 3 est lancé par défaut, on se trouve en mode interpréteur interactif, c'est-à-dire que l'on peut faire exécuter des lignes de codes

Python immédiatement. Pour vous en convaincre, saisissez à côté du prompt (`>>>`) la ligne suivante que vous validerez par **Entrée** :

```
>>> print ('Bonjour')
```

Ceci aura pour effet de faire afficher immédiatement en dessous le mot `Bonjour` comme vous pouvez le constater figure 5.2. L'instruction `print` du langage Python, dont le but est de faire afficher l'information qui la suit, a en effet été immédiatement interprétée par l'environnement de développement et donc a pu ensuite être exécutée par le processeur du Raspberry Pi.

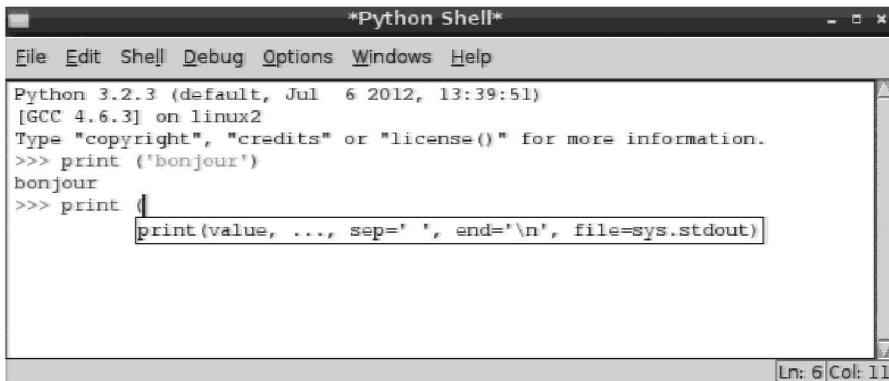


Figure 5.2 – Une aide syntaxique est disponible lors de la saisie des instructions.

Remarquez que, dès la saisie d'une instruction ou mot-clé du langage, une aide syntaxique est apparue dans un info-bulle comme on le constate figure 5.2. Dès que l'on possède les bases du langage, c'est là quelque chose de très confortable qui évite de devoir se plonger dans un manuel lorsque l'on a un doute sur la syntaxe d'une instruction.

Sans même connaître d'autres instructions du langage vous pouvez continuer à expérimenter en saisissant par exemple :

```
>>> 2+3
```

Pour constater que, dès la validation par **Entrée**, vous obtenez le résultat de l'opération soit 5 bien entendu.

5.1.2 De la ligne au programme

Si cette façon de procéder est très pratique pour essayer des instructions ou des petits morceaux de programmes, elle ne permet pas de réaliser des programmes à proprement parler. En effet, tout ce que vous saisissez après le prompt (`>>>`) est perdu dès que vous quittez l'environnement IDLE.

Pour écrire un programme, il va donc falloir placer la suite des instructions qu'il devra exécuter dans un fichier et dire ensuite au système que ce fichier contient un

programme Python de façon à lancer automatiquement l'interpréteur chaque fois que l'on y fera appel. Rassurez-vous, tout cela se fait très facilement avec l'environnement de développement comme nous allons le voir maintenant avec un exemple élémentaire. Voici en effet comment transformer notre première ligne de Python en programme.

Depuis le shell de IDLE 3, sélectionnez **New Window** dans le menu **File**, ce qui provoque l'ouverture d'une nouvelle fenêtre comme vous pouvez le constater figure 5.3. Remarquez que, dans celle-ci, ne figure plus le prompt (`>>>`) précédent car elle n'est plus destinée à une interprétation immédiate de vos instructions mais à la saisie de votre futur programme.

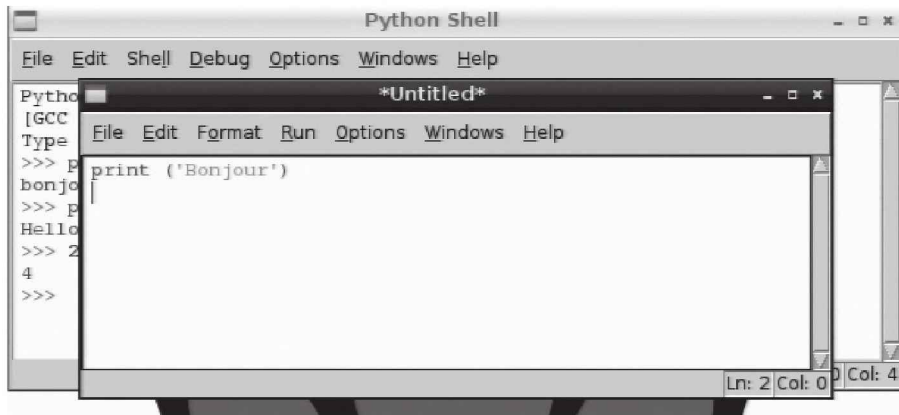


Figure 5.3 – La fenêtre permettant de saisir un programme.

Écrivez donc à nouveau dans cette fenêtre la ligne :

```
print ('Bonjour')
```

Vous constatez que, même après avoir frappé **Entrée**, rien ne se passe. Nous ne sommes plus en effet en mode interpréteur mais en mode saisie de programme. Ce programme, nous allons l'enregistrer sur la carte SD afin de pouvoir l'exécuter ultérieurement.

Pour cela, et afin que ce ne soit pas la pagaille sur notre carte, nous vous proposons de créer un répertoire baptisé par exemple `programmes_python` que vous placerez dans `/home/pi`. Pour cela, ouvrez l'explorateur de fichiers depuis l'environnement graphique (menu **Accessoires** de LXDE) et, au beau milieu de l'espace vide du répertoire `/home/pi` dont il affiche le contenu, faites un clic droit puis, dans le menu contextuel, choisissez **Dossier** comme indiqué figure 5.4. Donnez au dossier le nom `programmes_python` puis cliquez sur **Valider** et quittez l'explorateur de fichiers.

Vous pouvez dès lors sauvegarder le programme contenu dans la fenêtre d'IDLE 3 au moyen de la commande **Save As** du menu **File**. Sélectionnez bien entendu le répertoire `programmes_python` que nous venons de créer et donnez au programme un nom explicite ; `bonjour` par exemple.

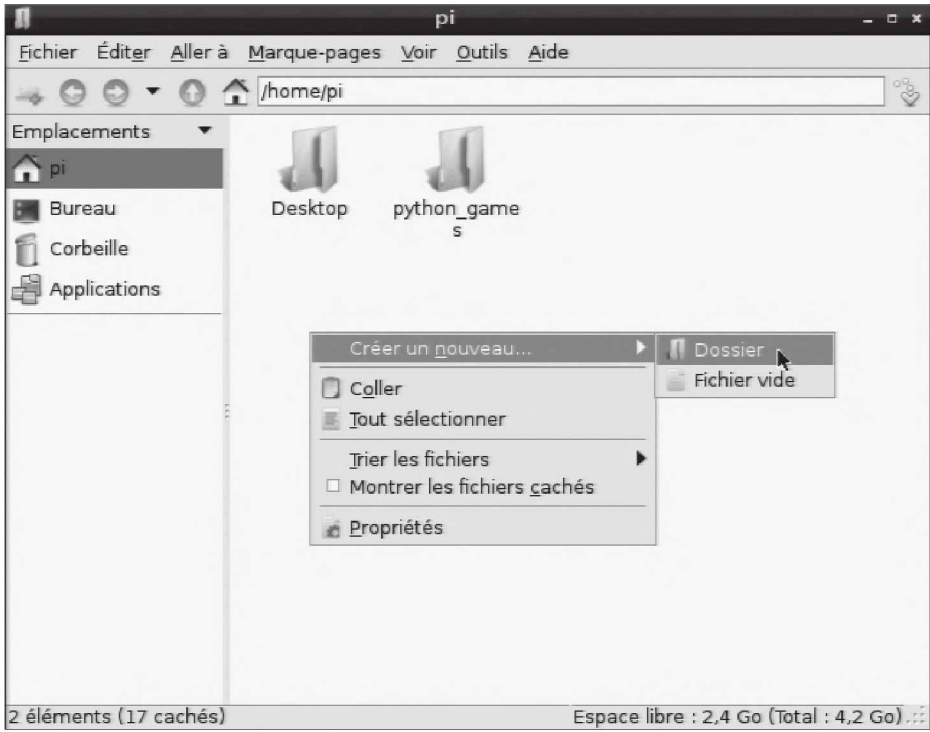


Figure 5.4 – Création d'un dossier pour ranger nos futurs programmes.

Remarquez qu'il n'est pas nécessaire de fournir un suffixe à ce nom car, comme nous sommes dans IDLE 3, ce dernier lui ajoute automatiquement le suffixe `.py` puisque c'est un programme Python.

Vous pouvez alors faire exécuter votre programme en sélectionnant **Run Module** depuis le menu **Run** ou bien encore en utilisant la touche de raccourci **F5**. Le programme s'exécute et, comme vous pouvez le constater figure 5.5, il affiche **Bonjour**. Ce n'est pas très original mais ça fonctionne.



Figure 5.5 – Notre premier programme s'exécute dans l'environnement IDLE 3.

En revanche, si vous quittez IDLE 3, il sera impossible de faire quoi que ce soit de votre fichier `bonjour.py` et c'est bien dommage. Aussi allons-nous voir sans plus tarder comment le transformer en un programme exécutable pour que, à la manière de toutes les commandes Linux que nous avons utilisées jusqu'à présent, vous puissiez le faire exécuter en tapant seulement son nom.

5.1.3 Rendre un programme Python exécutable

La procédure que nous allons décrire maintenant sera à appliquer à tous les programmes Python que vous voudrez rendre exécutables par la seule saisie de leur nom. Si vous voulez seulement les faire exécuter au travers de IDLE 3, comme nous l'avons fait précédemment, elle est évidemment inutile.

La première opération consiste à dire au programme où se trouve l'interpréteur Python sur la machine, ce qui s'obtient en ajoutant au tout début du programme la ligne suivante :

```
#!/usr/bin/env python
```

Ajoutez donc cette ligne au début du programme `bonjour` et sauvegardez-le à nouveau sous le même nom, comme nous venons de le faire. Il faut ensuite rendre ce programme exécutable, ce qui s'obtient avec une commande `chmod` que vous avez pu découvrir au chapitre 2.

Après avoir quitté IDLE 3 et être passé en mode ligne de commande dans une fenêtre de `LXTerminal` par exemple, saisissez :

```
pi@raspberrypi ~ $ cd programmes_python
```

pour passer dans le répertoire `programmes_python` puis :

```
pi@raspberrypi ~ /programmes_python $ chmod +x bonjour.py
```

pour donner l'attribut exécutable à `bonjour.py`.

Dès cet instant, le programme est exécutable comme vous pouvez le constater en saisissant :

```
./bonjour.py
```

C'est déjà un peu mieux mais, si nous laissons les choses en l'état, le programme ne peut être exécuté que s'il est appelé avec son chemin complet d'où le `./` qui a précédé son nom dans la ligne précédente.

Si vous voulez pouvoir appeler ce programme sans avoir à vous soucier de son chemin d'accès, il faut le copier dans le répertoire `/usr/local/bin` destiné à cet effet, comme nous l'avons vu au chapitre 2. Pour ce faire il faut utiliser la commande :

```
pi@raspberrypi ~ /programmes_python $ sudo cp bonjour.py /usr/local/bin
```

Vous pouvez alors revenir dans le répertoire par défaut au moyen de `cd ..` par exemple et essayer `bonjour.py` pour constater que, cette fois-ci, votre programme s'exécute bien sans être positionné au préalable dans le répertoire qui le contient.

Et si la frappe du suffixe `.py` vous ennuie, il suffit d'en débarrasser le nom du programme grâce à la commande suivante :

```
pi@raspberrypi ~ $ sudo mv /usr/local/bin/bonjour.py /usr/local/bin/
bonjour
```

Dès lors, la simple frappe de `bonjour`, depuis n'importe quel répertoire machine, conduira à l'exécution de votre programme Python.

Cette procédure peut bien sûr être appliquée à n'importe quel programme Python que vous serez amené à écrire et peut être synthétisée par les étapes suivantes :

- Placer en début de programme : `#!/usr/bin/env python`
- Rendre le programme exécutable par un `chmod +x nom.py` où `nom` est le nom du programme.
- Copier ou déplacer le programme dans le répertoire `/usr/local/bin`.
- Supprimer éventuellement le suffixe `.py` en renommant le programme sans ce suffixe.

Ceci étant, et puisque vous savez maintenant tester un programme en direct ou le transformer en programme exécutable, nous pouvons partir à la découverte du langage Python.

5.2 Organisation générale d'un programme

Comme tout langage de programmation, Python doit respecter certaines règles de syntaxe afin de permettre à l'interpréteur de bien comprendre et traduire les instructions. Dans le cas de Python, ces règles sont cependant relativement simples et peu nombreuses.

5.2.1 Règles de syntaxe d'un programme

Toute ligne de programme qui commence par un dièse (`#`) est considérée par Python comme du commentaire et n'est donc ni traduite, ni bien sûr exécutée. La seule exception à cette règle est la ligne `#!/usr/bin/env python` que nous vous avons demandé ci-avant de placer en début de programme, et qui a une signification particulière.

Toute ligne ne commençant pas par un dièse est donc considérée par Python comme une ligne de programme exécutable. En outre, il est possible, et même conseillé pour une meilleure lisibilité de vos programmes, de mettre des commentaires après une instruction exécutable. Il suffit de les faire précéder par un dièse.

```
# ligne de commentaires
instruction exécutable
instruction exécutable # suivie d'un commentaire
```

Contrairement à ce qui a lieu dans d'autres langages, Python utilise comme séparateur d'instruction l'espace. Ainsi, `print ('bonjour')` est-il correct alors que `print('bonjour')` fait générer une erreur. Sauf dans les blocs d'instructions présentés ci-dessous, le nombre d'espaces n'a aucune importance dès lors qu'il y en a au moins un.

Notez également qu'il est possible de placer plusieurs instructions sur une même ligne en les séparant les unes des autres par le caractère point-virgule. Cette pratique est toutefois à déconseiller car elle nuit à la bonne lisibilité des programmes. Python utilise également les espaces pour identifier ce que l'on appelle les blocs d'instructions. Ainsi pour réaliser une boucle, c'est-à-dire une suite d'instructions qui se répètent comme nous le verrons dans un instant, il faut procéder de la sorte :

```
Instruction de début de boucle :
    Instruction numéro 1 de la boucle
    Instruction numéro 2 de la boucle
    ...
    Dernière instruction de la boucle
Instruction suivant la boucle
```

Remarquez tout d'abord que la ligne contenant l'instruction de début de boucle est terminée par le caractère `deux points`, ce qui indique à Python qu'il doit rencontrer ensuite un bloc d'instructions.

Ce bloc d'instructions, contenues dans la boucle dans le cas présent, est identifié uniquement par son indentation par rapport aux autres lignes du programme. Cette indentation peut être réalisée à votre choix avec la touche `Tab` du clavier ou en saisissant plusieurs `espaces`, mais vous devez être homogène et ne pas mélanger les deux. Une pratique courante en Python veut que, si l'on utilise des espaces, on en utilise quatre.

Attention !

Il est très facile de créer des erreurs de programmes difficiles à identifier en mélangeant tabulations et espaces. Choisissez donc une façon de réaliser ces indentations et tenez-vous-y une fois pour toutes.

5.2.2 Importation de bibliothèques

Dans un certain nombre de programmes, il est nécessaire de faire appel à ce que les anglo-saxons appellent des *librairies*, ce que l'on peut traduire en français par bibliothèques. Ces dernières sont en fait des recueils de fonctions, généralement dans des domaines déterminés, qui viennent ainsi s'ajouter aux fonctions de base dont dispose le langage Python.

Certaines de ces bibliothèques sont installées en même temps que Python, d'autres sont écrites par des tierces personnes et doivent être installées manuellement avant de pouvoir être utilisées. Ce sera le cas par exemple de certaines bibliothèques de pilotages des entrées/sorties que nous verrons dans les chapitres suivants.

■ Import

Quoi qu'il en soit, lorsqu'un programme souhaite faire appel à au moins une fonction contenue dans une bibliothèque externe, il faut inclure celle-ci au début du programme avec une simple ligne du type :

```
import nom de la bibliothèque
```

Le nombre de ligne de ce type n'est pas limité et il est de ce fait possible d'importer plusieurs bibliothèques distinctes dans un même programme.

5.3 Variables numériques, chaînes, listes et dictionnaires

Comme dans tout langage de programmation qui se respecte, une variable est une zone de la mémoire où une valeur est stockée ; zone de mémoire à laquelle on donne un nom au moyen duquel le programme y fait ensuite référence.

Par contre, alors que dans d'autres langages de programmation il faut déclarer le type et le nom d'une variable avant de pouvoir l'utiliser, c'est inutile en Python et ces deux opérations ont lieu lors de la première utilisation de la variable.

5.3.1 Nommage des variables

Python est très souple quant au nommage des variables puisque le nom de ces dernières peut comporter des lettres de A à Z, en majuscules ou en minuscules, des chiffres de 0 à 9 et le caractère souligné (`_`) ou *underscore* en anglais.

Les seules contraintes à respecter sont les suivantes :

- Un nom de variable ne doit pas commencer par un chiffre.
- Un nom de variable ne doit contenir ni espace ni caractère accentué.
- Un nom de variable ne doit pas être un mot-clé du langage (une instruction si vous préférez).

■ Attention !

Python est sensible à la casse c'est-à-dire que Toto, toto et toTo seront considérées comme étant des variables différentes.

5.3.2 Les quatre types de données et variables de base

■ Entiers et flottants

Pour ce qui est des données numériques, Python connaît deux types de base qui sont automatiquement déterminés lors de l'initialisation de la variable qui doit les contenir. Ainsi, si vous écrivez :

```
A = 2
```

Vous déclarerez automatiquement une variable appelée `A` et, comme vous lui imposez d'être égale à 2 ce sera une variable de type entier.

Si vous aviez écrit :

```
B = 2.
```

Vous auriez fait de même avec la variable `B` mais, cette fois-ci, la variable aurait été du type flottant, ou décimal ou bien encore réel si vous préférez. Le chiffre 2 est en effet suivi d'un point qui, en notation internationale, correspond à notre virgule décimale. Si donc vous faites suivre un chiffre ou un nombre par un point, il est considéré comme étant représenté en virgule flottante.

Si vous avez un doute sur le type d'une variable (du point de vue de Python bien sûr) vous disposez de la fonction `type` (nom de la variable) pour vous indiquer celui-ci.

■ Chaînes de caractères

Python connaît évidemment aussi les données de type chaînes de caractères qui doivent être comprises entre deux délimiteurs identiques, qui sont généralement des apostrophes (') mais qui peuvent aussi être des guillemets (« ») si par exemple une apostrophe doit être contenue dans la chaîne. Nous avons déjà vu un tel type de chaîne ci-dessus avec notre programme `bonjour`.

Ces chaînes de caractères peuvent contenir des caractères « spéciaux » :

- `\n` permet ainsi d'insérer un saut à la ligne au sein d'une chaîne de caractères ;
- `\'` permet d'insérer une apostrophe dans une chaîne qui serait déjà délimitée par des apostrophes.
- `\` employé seul permet d'écrire sur plusieurs lignes une commande qui serait trop longue pour tenir sur une seule.

■ Variables logiques

Enfin, Python connaît aussi les deux types de variables logiques que sont `True` pour vrai et `False` pour faux, que l'on utilise par exemple lors des opérations de prise de décision ou dans certaines boucles.

5.3.3 Chaînes, listes, tuples et dictionnaires

■ Chaînes de caractères

En Python, une chaîne de caractères n'est en fait qu'une suite ordonnée de caractères et il est possible d'accéder à n'importe lequel d'entre eux en précisant sa position. Soit par exemple la chaîne suivante :

```
>>> ordinateur = 'Raspberry Pi'
```

Si nous essayons `>>> (ordinateur)` suivi de `Entrée` nous obtiendrons bien évidemment l'intégralité de la chaîne `Raspberry Pi`, mais si nous essayons maintenant :

```
>>> (ordinateur[2])
```

Nous obtiendrons comme résultat la seule lettre `s`.

En effet, Python considère toute chaîne de caractères comme ce que l'on appelle une variable indexée dans d'autres langages, et permet d'accéder à n'importe quel caractère en utilisant son index, sachant que le premier caractère de la chaîne est affecté de l'index 0 et que ce dernier doit être exprimé entre crochets derrière le nom de la chaîne. Ainsi, `chaîne[n]` fait afficher le (n+1)^e caractère de la chaîne.

Cette façon de faire va plus loin puisqu'il est ainsi possible d'extraire une sous-chaîne d'une chaîne existante en procédant de la sorte :

```
souschaîne = chaîne[n1:n2]
```

Dans ces conditions, `souschaîne` commence au caractère de rang `n1` et se termine au caractère de rang `n2+1`.

■ Attention !

Ne vous trompez pas en décomptant les caractères. N'oubliez pas que le 1^{er} caractère est celui d'indice 0. Par ailleurs, faites attention à `n2` qui ne repère pas le dernier caractère de la sous-chaîne mais le dernier + 1 !

Pour éviter toute confusion, essayez dans l'interpréteur IDLE 3 les quelques lignes suivantes :

```
>>> ordinateur = 'Raspberry Pi'
>>> (ordinateur[2])
's'
>>> souschaîne = ordinateur[10 :12]
>>> souschaîne
'pi'
```

Vous obtenez bien au final `pi` et non `p` puisque le `i` se trouve à la position d'index 11 !

Notez que ce mode d'extraction d'une sous-chaîne connaît deux variantes fort utiles :

```
souschaîne=chaîne[n1:]
souschaîne=chaîne[:n2]
```

Dans le premier cas on extrait tous les caractères depuis `n1` jusqu'à la fin de la chaîne et dans le second tous les caractères depuis le début de la chaîne jusqu'à celui d'indice `n2-1`.

Signalons aussi l'existence de la fonction `len(chaîne)` qui renvoie comme résultat la longueur de la chaîne exprimée en nombre de caractères.

■ Listes

Une fois que l'on a compris ces différents principes, celui concernant les listes est tout aussi simple. En effet, en Python une liste est une suite ordonnée de données, étant entendu que ces données peuvent être de n'importe quels types : numériques, chaînes de caractères, logiques et même listes.

Une liste se définit tout simplement en lui donnant un nom et en plaçant ensuite entre crochets tous les éléments qu'elle contient, séparés les uns des autres par des virgules comme dans l'exemple ci-dessous :

```
maliste = [2, 55, 'Raspberry Pi', True, 345]
```

On le voit bien sur cet exemple, tous les types de données peuvent y être présents sans que cela pose le moindre problème au langage Python.

Pour accéder à n'importe quel élément de la liste, il suffit de le spécifier par son index entre crochets comme nous l'avons fait pour les chaînes des caractères.

■ Attention !

Comme pour les chaînes de caractères, l'index commence à 0.

Essayez l'exemple ci-dessus avec IDLE 3 et différentes valeurs d'index afin de bien appréhender cette notion.

■ Attention !

Chaque élément de la liste correspond à un index quel que soit son type et sa taille. Ainsi, la chaîne Raspberry Pi de la liste précédente n'occupe-t-elle que le seul indice 2 bien qu'elle comporte 12 caractères.

La fonction `len`, que nous avons vue ci-dessus pour les chaînes de caractères, fonctionne aussi pour les listes et renvoie bien évidemment le nombre d'éléments contenus dans la liste. Mais il est possible de faire d'autres choses avec ces listes.

Ainsi, il est possible de remplacer n'importe quel élément d'une liste en écrivant tout simplement :

```
liste[n] = nouvel élément
```

Ce qui aura pour effet de remplacer l'élément d'index `n` par le nouvel élément ainsi désigné. Par exemple :

```
>>> maliste[1] = 4
```

transforme la liste de l'exemple précédent en :

```
>>> maliste  
[2, 4, 'Raspberry Pi', True, 345]
```

Si vous souhaitez retirer un élément de la liste, vous pouvez utiliser `pop` sous la forme suivante :

```
nomdelaliste.pop(n)
```

Où `nomdelaliste` est évidemment le nom donné à la liste tandis que `n` représente l'index de l'élément à enlever. Si `n` est absent, le dernier élément est automatiquement sélectionné. Essayez par exemple :

```
>>> maliste = [2, 4, 'Raspberry Pi', True, 345]  
>>> malitse.pop(2)  
'Raspberry Pi'
```

```
>>> maliste
[2, 4, True, 345]
```

Le troisième élément de la liste, d'index 2 (donc la chaîne `Raspberry Pi`), a bien été enlevé de la liste comme le confirme le test réalisé au moyen des quelques instructions précédentes.

Il est également possible de faire l'inverse, c'est-à-dire d'ajouter un élément au sein d'une liste au moyen cette fois-ci de `insert` qui s'utilise de la façon suivante :

```
nomdelaliste.insert(position, donnée)
```

Où `nomdelaliste` est évidemment le nom de la liste, `position` est l'indice indiquant la donnée avant laquelle l'insertion sera faite tandis que `donnée` est la donnée à insérer. Si nous reprenons l'exemple précédent afin de remettre `Raspberry Pi` à sa place initiale nous écrirons ainsi :

```
>>> maliste.insert(2, 'Raspberry Pi')
>>> maliste
[2, 4, 'Raspberry Pi', True, 345]
```

Le mode de notation que nous avons utilisé ici, à savoir le nom de la liste suivi d'un point puis de la fonction à exécuter, est propre aux langages supportant ce que l'on appelle la programmation orientée objet, auxquels Python appartient. La liste est ce que l'on appelle dans ce cas un objet et la fonction qui s'applique à cet objet, dont elle est séparée par un point, s'appelle une méthode. Nous aurons l'occasion de revenir sur ces notions lors de la présentation de certains exemples de commandes d'interfaces dans les chapitres suivants.

Attention !

Même si les chaînes des caractères partagent un certain nombre de caractéristiques et de fonctions communes avec les listes, il est impossible de modifier, d'ajouter ou de retirer un caractère d'une chaîne comme nous l'avons fait avec une liste.

■ Tuples

Les tuples (pas de traduction connue !) ressemblent aux listes mais ne sont pas modifiables. Ils se définissent comme les listes mais cette fois-ci sans nécessiter de crochets ni de parenthèses. Par contre, comme cela peut parfois conduire à une certaine confusion lors de la lecture des programmes, il est possible de définir un tuple en encadrant les valeurs entre parenthèses sans que Python ne génère une erreur pour autant.

Comme on ne peut pas modifier les tuples, il est impossible de faire appel à `pop` ou à `insert` par exemple mais on peut parfaitement extraire des « sous-tuples » comme nous l'avons fait précédemment pour les chaînes. Les index fonctionnent évidemment de la même façon. Voici d'ailleurs un petit exemple :

```
>>> montuple = (2, 4, 'Raspberry Pi', True, 345)
>>> montuple[2:4]
['Raspberry Pi', True]
```

Notez que dans cet exemple, nous avons repris comme contenu du tuple, baptisé `montuple` (quelle imagination !) les mêmes éléments que ceux de `maliste` utilisée précédemment.

■ Dictionnaires

Dernier type de liste étudié ici, le dictionnaire est également une suite de données qui peuvent être hétéroclites mais, au lieu d'être référencées par un index comme dans les listes ou les tuples, chaque donnée est référencée par une clé qui lui est associée lors de son inscription dans le dictionnaire. Afin de le distinguer des listes et des tuples, les éléments d'un dictionnaire sont contenus entre des accolades (`{}`).

Voici un exemple très simple avec un dictionnaire de traduction anglais – français de quelques termes informatiques baptisé `traducteur`.

```
>>> traducteur = {'memory' : 'mémoire', 'mouse' : 'souris', 'computer' : 'ordinateur'}
>>> traducteur['mouse']
souris
```

Le dictionnaire contient des couples clés – données séparés l'un de l'autre par le caractère deux points, chaque couple étant séparé du suivant par une virgule. Dès lors, pour extraire une donnée du dictionnaire, il suffit de l'appeler par sa clé comme nous l'avons fait par exemple avec `mouse`.

■ Attention !

Données et clés ne sont pas interchangeables et, dans le couple clé – donnée, c'est la clé qui est en premier.

Il est possible d'ajouter à tout instant des couples à un dictionnaire existant, de même qu'il est possible de remplacer une valeur par une autre comme nous l'avons fait pour les listes mais en référençant ici la donnée par sa clé. Voici un petit exemple avec le dictionnaire précédent :

```
>>> traducteur
{'memory' : 'mémoire', 'mouse' : 'souris', 'computer' : 'ordinateur'}
>>> traducteur['mouse'] = 'mulot'
>>> traducteur
{'memory' : 'mémoire', 'mouse' : 'mulot', 'computer' : 'ordinateur'}
```

5.4 Opérateurs, boucles et prise de décision

Maintenant que nous connaissons un peu mieux les types de données pouvant être manipulées par Python, voyons quels sont les opérateurs qu'il est possible de leur appliquer, étant entendu bien sûr que certains d'entre eux ne seront utilisables qu'avec certains types de variables.

5.4.1 Opérateurs arithmétiques

■ Égalité, addition, soustraction, multiplication, division

Le premier opérateur est évidemment l'opérateur « égal » (=) ou opérateur d'assignation, à ne pas confondre avec l'opérateur « égal à » (=) qui est un opérateur de comparaison que nous découvrirons ci-après même si, dans certains langages de programmation tels que le Basic par exemple, ces deux opérateurs sont confondus en un seul symbole.

L'opérateur égal sert donc à assigner une valeur à une constante ou à une variable. Ainsi `A = 2` signifie-t-il que l'on donne la valeur 2 à la variable `A`. Contrairement à l'égal dont on a l'habitude en algèbre, il n'est pas commutatif et `2 = A` conduirait à une erreur car cela n'aurait aucun sens. On ne peut pas en effet donner la valeur `A` au chiffre 2 !

Les quatre opérateurs suivants sont l'addition (+), la soustraction (-), la multiplication (*) et la division (/). Ils se comportent bien évidemment comme vous l'avez appris depuis l'école primaire mais notez toutefois que si les deux opérandes sont de type différent, entier et flottant par exemple, le résultat est de type flottant.

Ces précisions étant vues, ces opérateurs s'utilisent de la façon suivante :

```
résultat = variable1 opérateur variable2
```

■ Division entière et reste

Deux opérateurs particuliers doivent aussi être mentionnés. Le premier, noté //, est une division entière, tandis que le second, appelé modulo et noté %, permet d'accéder justement au reste de cette division entière. Il s'utilise de la façon suivante :

```
reste = dividende % diviseur
```

Ainsi par exemple, `reste` sera égal à 1 suite à l'écriture de :

```
reste = 9 % 4
```

puisque 9 divisé par 4 est égal à 2 avec un reste de 1.

■ Élévation à une puissance

Dernier opérateur à mentionner, l'élévation à une puissance qui se note comme c'est souvent le cas en programmation : **. Ainsi, `x` à la puissance 3 s'écrit-il `x**3`.

■ Concaténation

Avant d'en terminer avec ces opérateurs arithmétiques, précisons que l'opérateur d'addition, noté +, change de nom et s'appelle alors opérateur de concaténation mais conserve le même symbole lorsqu'il est utilisé avec des chaînes de caractères.

Ainsi par exemple : `'bonjour' + 'monsieur'` conduit-il à créer la chaîne `'bonjour monsieur'`.

C'est évidemment la seule opération que l'on peut réaliser avec des chaînes de caractères car toutes les autres n'auraient aucun sens.

5.4.2 Opérateurs de comparaison et opérateurs logiques

Les opérateurs de comparaison sont utilisés toutes les fois où l'on doit prendre une décision au moyen des instructions que nous verrons ci-après. Cette prise de décision résulte toujours de l'état d'une condition qui peut être vraie (`True`) ou fausse (`False`).

■ Opérateurs de comparaison

Les opérateurs disponibles en Python sont classiques et se retrouvent dans quasiment tous les langages de programmation, avec les mêmes symboles de surcroît. Le tableau 5.1 en dresse la liste complète.

Tableau 5.1 – Opérateurs de comparaison.

Notation	Fonction
<code>x == y</code>	x est égal à y
<code>x != x</code>	x est différent de y
<code>x < y</code>	x est inférieur à y
<code>x > y</code>	x est supérieur à y
<code>x <= y</code>	x est inférieur ou égal à y
<code>x >= y</code>	x est supérieur ou égal à y

Ces conditions ont souvent besoin d'être combinées lors des prises de décision ; combinaisons qui ont lieu au moyen d'opérateurs logiques bien connus puisqu'issus du langage courant. Ils sont au nombre de trois en Python.

■ and

L'opérateur `and` (pour et en français) autorise la condition résultant de la combinaison de deux conditions à être vraie si et seulement si les deux conditions sont vraies. Ainsi par exemple :

`a > 2 and b < 3`

ne sera satisfaite que si, simultanément, `a` est supérieur à 2 ET `b` est inférieur à 3.

■ **or**

Le **or** (pour ou en français) autorise la condition résultant de la combinaison de deux conditions à être vraie si une au moins des deux conditions est vraie. Ainsi par exemple :

```
a > 2 or b < 3
```

sera satisfaite si *a* est supérieur à 2 ou si *b* est inférieur à 3. Comme il s'agit d'un ou logique inclusif, le fait que les deux conditions soient satisfaites est également valide.

■ **Attention !**

Le ou inclusif utilisé ici diffère du ou du langage parlé qui est généralement un ou exclusif (j'ai pris le train ou l'avion mais pas les deux !)

■ **not**

Le dernier opérateur logique est baptisé **not** (pour non en français) et a pour effet d'inverser l'état logique d'une variable ou d'une condition. **not x** est **False** (fausse) si *x* était **True** (vraie) et vice versa.

5.4.3 Prise de décision

Ces opérateurs de comparaison sont généralement utilisés dans des instructions de prise de décision dont il n'existe en Python qu'une version, mais avec plusieurs variantes.

■ **if**

La seule instruction de prise de décision n'est autre que **if** qui s'utilise sous la forme suivante :

```
if expression conditionnelle :
    Bloc d'instructions
Instruction suivante
```

Où **expression conditionnelle** est une expression faisant intervenir un ou plusieurs opérateurs de comparaison, associés éventuellement avec des opérateurs logiques. Si cette expression est vraie, le bloc d'instructions qui suit (repéré par son indentation comme nous l'avons vu au § 5.2.1 ci-dessus) est exécuté. Dans le cas contraire, il est ignoré et l'exécution se poursuit au niveau de **instruction suivante**.

■ **else**

Si un bloc d'instructions spécifique doit être exécuté seulement lorsque la condition n'est pas vraie, il est possible de recourir à **else** en écrivant cette fois-ci :

```
if expression conditionnelle :
    Bloc d'instructions si condition vraie
else :
    Bloc d'instructions si condition fausse
Instruction suivante
```

■ elif

Il est également possible d'écrire plusieurs instructions conditionnelles les unes à la suite des autres afin de réaliser des sélections multiples. On peut alors recourir à `elif` qui est la contraction en un seul « mot » de `else if`. On obtient alors la syntaxe suivante :

```
if expression conditionnelle 1:
    Bloc d'instructions si condition 1 vraie
elif expression conditionnelle 2:
    Bloc d'instructions si condition 2 vraie
elif expression conditionnelle 3:
    Bloc d'instructions si condition 3 vraie
else :
    Bloc d'instructions si condition 3 fausse
Instruction suivante
```

Il n'existe pas, en Python, d'autres instructions de prise de décision contrairement à ce que l'on trouve parfois dans d'autres langages.

5.4.4 Boucles itératives et conditionnelles

Alors qu'un programme se déroule normalement en séquence, c'est-à-dire que les instructions sont exécutées les unes après les autres, il peut être nécessaire de répéter certaines portions de programme et on utilise alors pour ce faire des boucles. Il existe en Python deux types de boucles, les boucles itératives, où l'on exécute les instructions un certain nombre de fois, défini à l'avance, et les boucles conditionnelles où l'on exécute les instructions tant qu'une condition est réalisée ou non.

■ for

La boucle itérative ou boucle `for` utilise la syntaxe suivante :

```
for variable in sequence :
    Bloc d'instructions de la boucle
Instruction suivante
```

`variable` est ce que l'on appelle la variable de boucle tandis que `sequence` est une liste (voir ci-dessus) dont toutes les valeurs vont être parcourues par `variable`. Pour chaque valeur de cette liste égale à `variable`, le bloc d'instructions de la boucle va être exécuté une fois. Ainsi, si nous voulons épeler `Raspberry` nous pourrions écrire :

```
for x in 'Raspberry' :
    print (x)
print 'la boucle est finie'
```

Ce qui aura pour effet de faire afficher chaque lettre de la « liste » `Raspberry`, qui est aussi bien sûr une chaîne de caractères, sur une ligne distincte.

■ range

Très souvent, ces boucles doivent être réalisées conjointement à une progression numérique de la variable de boucle. Afin de ne pas devoir créer une liste spécifique contenant tous les chiffres nécessaires, ce qui serait fastidieux si le nombre de tours de boucle était important, on peut utiliser le mot-clé `range`, de la façon suivante :

```
for x in range (1,11) :
    print (x)
print 'j\'ai compté de 1 à 10'
```

`range(n1, n2)` génère en effet une liste d'entiers compris entre `n1` et `n2-1`.

Il peut aussi être utilisé sous la forme `range(n)`, auquel cas il génère une liste d'entiers compris entre 0 et `n-1`, mais aussi sous la forme `range (n1, n2, n3)`, auquel cas il génère une liste d'entiers compris entre `n1` et `n2` mais avec le pas de progression `n3`. Ainsi par exemple `range (1, 14, 2)` génère-t-il la liste : 1, 3, 5, 7, 9, 11, 13.

■ while

Contrairement à la boucle `for`, la boucle `while` est une boucle conditionnelle, c'est-à-dire qu'elle n'est exécutée que tant qu'une condition est vraie. Sa syntaxe est la suivante :

```
while condition :
    Bloc d'instructions exécutées tant que la condition reste
    vraie
Instruction suivante
```

Le bloc d'instructions placé après la ligne contenant `while` sera exécuté autant de fois que nécessaire jusqu'à ce que la condition qui suit `while` devienne fausse. Le programme continuera alors son déroulement avec `instruction suivante`. Si la condition n'est jamais vraie, la boucle n'est jamais exécutée (même pas une seule fois).

L'exemple très simple ci-dessous affiche ainsi automatiquement les chiffres de 1 à 10 :

```
i=1
while i<=10 :
    print (i)
    i=i+1
print ('la boucle est terminée')
```

■ Attention !

Si la condition ne devient jamais fausse, la boucle s'exécute alors indéfiniment et le programme se trouve donc bloqué à son niveau, sauf si l'on utilise `break` (voir ci-dessous) pour sortir « de force » de cette dernière.

■ break

Pour de multiples raisons il peut être nécessaire de sortir prématurément d'une boucle, que ce soit une boucle `for` ou une boucle `while`. Il suffit pour cela d'utili-

ser `break` qui provoque alors une sortie immédiate de la boucle et la poursuite du programme au niveau de la première instruction qui suit le bloc d'instructions de cette dernière.

`break` s'utilise seule et sans paramètre.

■ continue

Cette instruction permet également de modifier le comportement d'une boucle, que ce soit une boucle `for` ou une boucle `while`. Dès que le programme arrive sur la ligne contenant l'instruction `continue`, il recommence la boucle à son début sans exécuter les instructions qui suivent dans le bloc d'instructions.

Comme `break`, `continue` s'utilise seule et sans paramètre.

5.5 Entrées et sorties de données

Un programme communique généralement avec le monde extérieur, que ce soit directement avec son utilisateur au moyen de l'écran et du clavier, ou via des fichiers dans lesquels sont lues ou écrites des données. Un certain nombre de fonctions sont donc à notre disposition pour cela.

5.5.1 Sorties de données avec print

La fonction `print` a pour effet de faire afficher sur le terminal utilisé, l'écran du Raspberry Pi dans le cas présent, la ou les variables qui suivent le mot-clé. Ces variables doivent être contenues entre parenthèses en Python 3 alors qu'elles sont par contre inutiles en Python 2.

S'il y en a plusieurs, ces variables doivent être séparées les unes des autres par une virgule, et elles sont alors affichées sur la même ligne, en étant séparées automatiquement par un espace, comme dans l'exemple ci-dessous que vous pouvez essayer sous IDLE 3 :

```
>>> a = 2
>>> b = 3
>>> print (a,b)
2 3
```

Si les variables doivent être affichées à raison d'une variable par ligne, il faut utiliser une fonction `print` par variable.

Lors de l'affichage de nombres, et en l'absence de directive particulière, ceux-ci sont affichés dans leur intégralité, ce qui est parfois gênant pour les nombres réels ou décimaux qui apparaissent ainsi avec tous les chiffres après la virgule que Python est capable d'utiliser, et ils sont nombreux ! Pour vous en convaincre, essayez par exemple :

```
>>> a = 7/3
>>> print (a)
```

Ce qui va donner comme affichage :

```
2.3333333333333335
```

Très souvent, on souhaite limiter le nombre de décimales affichées, ce qui est possible en Python grâce à l'opérateur de formatage noté % qui s'utilise, entre autres modes, sous la forme %.nt où n indique le nombre de décimales désirées et t le type de donnée. Ce type est égal à f pour un nombre décimal (ou flottant) et à i pour un nombre entier (i comme integer).

Si le nombre à afficher doit apparaître au beau milieu d'une chaîne de caractères, il est remplacé au sein de la chaîne par cet opérateur qui est ensuite répété à la suite de la chaîne, à côté de la variable à laquelle il s'applique. Un exemple est plus parlant qu'un long discours, voici un exemple que vous pouvez essayer sous IDLE 3.

```
>>> a = 7/3
>>> print ('7/3 est égal à %.2f si on arrondit à 2 décimales' % a)
```

Ce qui doit vous donner comme résultat :

```
7/3 est égal à 2.33 si on arrondit à 2 décimales
```

Cet opérateur permet aussi d'indiquer sur combien d'emplacements (pas nécessairement de chiffres donc) un nombre doit être écrit, ce qui permet de réaliser des affichages parfaitement alignés. Voici à nouveau un exemple à essayer sous IDLE 3 permettant d'en comprendre le principe, avec des nombres entiers cette fois-ci.

```
>>> print (10) ; print (100) ; print (1000)
10
100
1000
```

L'affichage obtenu correspond à ce que l'on est en droit d'attendre de la fonction print ne recourant à aucun formatage particulier. Si maintenant nous souhaitons aligner les nombres par la droite, il suffit d'ordonner à print d'afficher les entiers sur quatre « chiffres » (en fait sur quatre positions lorsque les chiffres n'existent pas) ce que l'on note :

```
>>> print ('%4i' % 10) ; print ('%4i' % 100) ; print ('%4i' % 1000)
10
100
1000
```

D'autres possibilités de formatage sont offertes par la fonction print mais, dans le cadre de cette découverte de Python, nécessairement limitée en temps et en place, il n'est pas utile d'en faire état.

5.5.2 Entrée de données avec input

Pour qu'un opérateur puisse fournir des données à un programme, il dispose de la fonction `input` qui s'utilise sous l'une des deux formes suivantes :

```
variable = input
variable = input ('texte')
```

Dans les deux cas, l'exécution du programme est interrompue et attend que l'utilisateur saisisse une donnée au clavier et la valide par **Entrée**. Cette donnée est affectée à `variable` dont le type est automatiquement adapté à la donnée saisie (chaîne de caractères pour du texte, entier pour un nombre entier, etc.)

La seconde forme d'utilisation de `input` permet d'afficher du texte au moment de la saisie afin, par exemple, d'indiquer à l'utilisateur ce qu'il doit fournir comme information.

■ Attention !

Dans le cas de la deuxième forme, pensez à prévoir un espace à la fin de la chaîne de caractères si vous ne voulez pas que ce qui sera saisi au clavier soit collé à celle-ci.

5.5.3 Écriture dans un fichier

De nombreux programmes ont besoin d'écrire des données, non pas sur l'écran destiné à l'utilisateur, mais dans un fichier pour un traitement ultérieur par exemple. Un programme qui enregistre des températures fournies par un capteur peut ainsi, par exemple, créer un fichier contenant ces enregistrements afin de pouvoir les traiter ensuite avec un tableur dans le but de réaliser diverses opérations statistiques.

■ open

En Python, l'écriture et la lecture dans un fichier sont des opérations particulièrement simples mais, que l'on veuille lire ou écrire, il faut commencer par « ouvrir » le fichier, ce qui s'obtient grâce à la ligne suivante :

```
nom = open ('nom complet du fichier', 'm')
```

`nom` représente le « nom de variable » qui va être donné au fichier dans la suite du programme. Il est purement formel et n'a rien à voir avec le nom réel du fichier, même s'il peut parfois être similaire afin de faciliter la compréhension du programme.

`nom complet du fichier` est, comme son nom l'indique, le nom réel du fichier sur le disque (ou sur la carte SD dans le cas du Raspberry Pi). Par défaut, ce fichier est considéré comme étant dans le répertoire courant mais il est possible de spécifier tout autre emplacement sous réserve d'indiquer le chemin d'accès complet.

`m` est le mode dans lequel le fichier doit être ouvert et ne peut être que l'une des trois lettres : `r` pour read ou lecture, `w` pour write ou écriture, `a` pour append ou ajout.

Les modes `a` et `w` permettent tous deux d'écrire dans un fichier mais se comportent de façon différente. Le mode `a` permet d'ajouter des données à la suite de celles déjà présentes dans un fichier existant, tandis que le mode `w` permet de créer un nouveau fichier afin d'y placer des données. Si `w` est utilisé avec un fichier existant, ce dernier est effacé afin d'y placer les données à écrire.

■ write

Une fois le fichier ouvert, il est possible d'y écrire au moyen de la ligne suivante :

```
nom.write ('donnée à écrire')
```

`nom` est le « nom de variable » qui a été donné au fichier lors de son ouverture par `open`.

Si vous voulez réaliser quelques essais avec IDLE 3, vous pouvez par exemple saisir les lignes suivantes :

```
>>> essai = open ('test.txt', 'w')
>>> essai.write ('première donnée')
>>> essai.write ('deuxième donnée')
>>> essai.close ()
```

Ceci a pour effet de créer un fichier baptisé `test.txt`, qui est placé dans le répertoire courant, puis d'y écrire deux chaînes de caractères.

■ close

Notez dans cet exemple l'apparition d'une nouvelle fonction permettant de fermer le fichier, ce qui est obligatoire lorsque l'on a fini de l'utiliser. La syntaxe est évidemment de la forme :

```
nom.close ()
```

`nom` est ici encore le « nom de variable » donné au fichier lors de son ouverture.

Nous ne pouvons rien faire pour l'instant avec ce nouveau fichier tant que nous n'avons pas vu comment lire dans un fichier, ce qui ne saurait tarder.

5.5.4 Lecture dans un fichier

Tout comme pour l'écriture, il faut évidemment commencer par ouvrir le fichier avec la même syntaxe que celle présentée ci-dessus, mais en utilisant cette fois-ci le mode `r` bien évidemment.

■ read

La lecture proprement dite peut alors avoir lieu au moyen de l'une des deux lignes suivantes :

```
variable = nom.read ()
variable = nom.read(n)
```

`variable` est la variable qui contiendra la donnée lue dans le fichier, tandis que `nom` est le « nom de variable » donné au fichier lors de son ouverture. Comme dans le cas précédent, ce nom est totalement indépendant du nom du fichier, de même qu'il est indépendant du « nom de variable » qui a été utilisé lors de l'écriture dans le fichier.

Si aucune donnée n'est spécifiée entre les parenthèses, l'intégralité du fichier est lue. Si `n`, qui est une valeur numérique entière est spécifiée, la lecture ne comportera que les `n` caractères à partir de la position déjà atteinte dans le fichier.

S'il ne reste pas assez de caractères entre la position courante et la fin du fichier pour satisfaire cette valeur de `n`, le fichier est lu jusqu'à sa fin mais aucune erreur n'est générée. Si par contre le fichier a déjà été lu jusqu'à la fin, `read` renvoie une chaîne vide.

Concrétisons cela avec le fichier que nous avons créé sous IDLE 3 à l'étape précédente :

```
>>> lecture = open ('test.txt', 'r')
>>> contenu_fichier = lecture.read ()
première donnée deuxième donnée
>>> lecture.close ()
```

Nous avons appelé cette fois-ci notre fichier `lecture` et nous avons bien pu lire l'intégralité de son contenu comme prévu.

■ Attention !

Comme cela a été sous-entendu dans tout ce qui précède, les fichiers ainsi manipulés ne contiennent que des données de types chaînes de caractères. S'ils doivent contenir des données numériques, il est donc nécessaire de convertir ces dernières de nombres en chaînes et vice-versa afin de pouvoir les utiliser.

5.6 Fonctions, modules, classes et méthodes

5.6.1 Fonctions

Lorsque l'on doit réaliser dans un programme des tâches répétitives mais utilisant des données différentes, il est préférable de créer une fonction plutôt que de répéter à de multiples reprises les mêmes instructions. Une fois écrite, celle-ci pourra être ensuite appelée autant de fois que cela sera nécessaire, avec des paramètres différents si besoin est.

■ def

La syntaxe de définition d'une fonction en Python est très simple et n'utilise que le mot-clé `def` de la façon suivante :

```
def nom de la fonction (liste de paramètres) :
    Bloc d'instructions
    return résultat ou none
```

nom de la fonction est évidemment le nom que vous souhaitez donner à la fonction. Il doit respecter les règles de nommage vues pour les variables en début de chapitre et, en général, ne doit pas commencer par une majuscule car cela est usuellement employé pour définir les classes (mais ce n'est pas une erreur). Il ne doit pas non plus être un des mots-clés du langage, mais cela tombe sous le sens sinon comment voudriez-vous que Python fasse la différence entre ses fonctions ou instructions propres et votre fonction ?

liste de paramètres est la suite des paramètres à passer à la fonction, séparés les uns des autres par des virgules. Si aucun paramètre n'est nécessaire, les deux parenthèses sont accolées.

Bloc d'instructions est l'ensemble des instructions qui constituent la fonction. Il doit être indenté avec des **espaces** ou le caractère **Tab** comme nous l'avons vu précédemment pour les boucles **for** ou **while** par exemple.

return enfin marque la fin de la fonction et doit être suivi, soit du nom de la variable qui contient la donnée que la fonction fournit en retour, soit du mot **none** si la fonction ne fournit aucune donnée en retour.

Soit par exemple à réaliser une fonction qui calcule la somme du carré et du cube de deux nombres (résultat = $a^2 + b^3$) ; cette fonction sera définie de la façon suivante que vous pouvez parfaitement essayer sous IDLE 3 :

```
>>> def mafonction (a, b) :
>>>     res = a**2 + b**3
>>>     return res
```

Pour l'essayer, il suffira ensuite de saisir :

```
>>> mafonction (4, 5)
141
```

Notez que toutes les variables utilisées dans une fonction sont internes à cette dernière et n'ont rien à voir avec celles du programme appelant ou avec celles d'autres fonctions. Ce sont ce que l'on appelle des variables locales. Par contre, les variables utilisées dans le programme principal, ou programme appelant la fonction, sont des variables dites globales. Elles sont visibles de la fonction, qui peut donc les lire si nécessaire, mais ne peut en aucun cas les modifier.

5.6.2 Modules

Si vous avez rédigé plusieurs fonctions dans le même domaine, il peut être intéressant de les regrouper dans ce que l'on appelle alors un module ; module auquel vous donnez alors un nom de fichier correspondant au domaine d'utilisation des fonctions.

De très nombreux modules Python existent déjà, soit parce qu'ils font partie de la bibliothèque standard, soit parce qu'ils ont été écrits par la communauté des développeurs Python qui est très nombreuse.

Une liste aussi complète que possible de ces modules est disponible à l'adresse suivante : <http://docs.python.org/release/3.1.5/library/index.html>. Chaque module y est en outre parfaitement documenté avec la présentation des fonctions qu'il contient ainsi que leur syntaxe.

■ import

Si vous avez besoin d'un ou plusieurs modules dans un de vos programmes, il suffit de les importer au début de celui-ci en écrivant :

```
import nom du module
```

Par défaut, ce mode d'écriture importe toutes les fonctions du module qu'il est ensuite possible d'utiliser directement dans votre programme sous la forme :

```
nom du module.nom de la fonction
```

Considérons par exemple le module `random` qui gère tout ce qui a trait aux nombres aléatoires et qui contient, entre autres choses, la fonction `randint` qui génère un nombre aléatoire entier compris entre deux bornes. Nous vous proposons de l'essayer sous IDLE 3 en écrivant :

```
>>> import random
>>> random.randint (1,10)
4
```

Il est également possible, quoique parfois dangereux à notre avis, de changer le nom d'un module dans le programme qui l'importe en écrivant lors de son importation :

```
import nom du module as nouveau nom
```

Pour appeler ensuite les fonctions de ce module, il faudra bien évidemment écrire :

```
nouveau nom.nom de la fonction
```

L'exemple précédent pourrait ainsi s'écrire :

```
>>> import random as aleatoire
>>> aleatoire.randint (1,10)
6
```

Dans le cas où vous n'avez besoin que d'une fonction d'un module et que son nom n'interfère pas avec le nom des fonctions standards de Python, il est également possible de réaliser l'importation de cette seule fonction de la façon suivante :

```
from nom du module import nom de la fonction
```

La fonction ainsi importée pourra ensuite être appelée dans le programme sous son seul nom, sans la faire précéder par le nom du module.

Toujours avec l'exemple précédent nous pourrions ainsi écrire :

```
>>> from random import randint
>>> randint (1,10)
8
```

Il est même possible de faire, bien que ce soit déconseillé car sujet à conflits et erreurs ultérieures :

```
from nom du module import *
```

Ceci a pour effet d'importer toutes les fonctions du module qui peuvent donc ensuite être appelées par leurs seuls noms, sans les faire précéder par le nom du module d'origine.

5.6.3 Classes et méthodes

Nous abordons ici la partie la plus délicate des notions relatives à ce que l'on appelle la programmation orientée objet, supportée par nombre de langages de programmation modernes dont bien sûr Python. Si l'on veut être exhaustif dans ce domaine, il faut y consacrer de très longs développements qui sortent largement du cadre de cette simple découverte de Python et surtout des besoins que nous allons avoir pour la suite de cet ouvrage.

Si vous ne connaissez pas du tout la programmation orientée objet, cela devrait cependant vous suffire dans un premier temps et, si vous savez tout de cette programmation, ne lisez pas ce qui suit ou pardonnez-nous ce que vous pourrez considérer comme de grossières approximations, mais qui permettent à un débutant en ce domaine d'y voir un peu plus clair.

En première approximation, on peut considérer qu'une classe est un module agissant sur des objets au moyen de méthodes, qui sont peu ou prou l'équivalent des fonctions des modules que nous venons de présenter.

Les objets peuvent être à peu près tout ce que Python est capable de manipuler. Les chaînes de caractères sont ainsi des objets qui appartiennent à la classe `str` (début de *string* pour chaîne en anglais).

Les nombres réels, ou flottants si vous préférez, sont aussi des objets qui appartiennent quant à eux à la classe `float` (comme flottant en anglais) et ainsi de suite. Et si vous avez un doute quant à l'appartenance d'un objet à une classe, il suffit de saisir la ligne suivante :

```
objet.__class__
```

Essayez sous IDLE 3 par exemple :

```
>>> 'bonjour'.__class__
<class 'str'>
>>> 1.25.__class__
<class 'float'>
```

En fait, cette notion de programmation orientée objet va un peu plus loin que cela en associant, au sein d'un même objet, sa classe, ses instances qui sont les variables du type de l'objet (la chaîne de caractère 'bonjour' par exemple), et ses méthodes qui sont les fonctions que l'on peut appliquer à l'objet.

Dans la classe chaîne de caractères par exemple, il existe la méthode `capitalize` qui permet de mettre automatiquement en majuscule le premier caractère d'une chaîne et lui seul. Nous pourrons dès lors l'utiliser sous la forme `'bonjour'.capitalize()` pour mettre un B majuscule au début de `bonjour`.

`'bonjour'` est ce que l'on appelle une instance de la classe `str` tandis que `capitalize()` est une méthode de la classe `str`.

Nous en resterons là de cette très rapide présentation de ces notions car nous aurons l'occasion de les utiliser en partie dans les chapitres qui suivent, lorsque nous allons agir sur les entrées/sorties du Raspberry Pi.

6 • DES ACCESSOIRES INDISPENSABLES

Dans ce chapitre, qui préfigure les suivants où nous allons apprendre à utiliser les ports d'entrées/sorties du Raspberry Pi pour le connecter au monde extérieur, nous allons voir comment réaliser quelques accessoires indispensables.

Comme vous n'êtes pas forcément des électroniciens aguerris et que vous ne possédez pas nécessairement un outillage très complet en ce domaine, nous n'avons fait appel qu'à des solutions très simples à mettre en œuvre pour lesquelles le matériel nécessaire se résume au minimum.

Un fer à souder à panne fine de 25 à 45 watts environ, de la soudure d'électronicien avec ou sans plomb peu importe (celle avec plomb, de plus en plus difficile à trouver, soude mieux et nous ne sommes pas convaincus de sa toxicité...), des pinces coupantes à becs fins bien aiguisées et une paire de pinces plates à bec fins ; c'est à peu près tout ce qu'il va vous falloir pour mener à bien nos premières réalisations.

6.1 Un poussoir de reset pour votre Raspberry Pi

Même si Raspbian Wheezy, comme d'ailleurs la majorité des distributions Linux pour le Raspberry Pi, est très stable et se « plante » donc très rarement, il peut tout de même être nécessaire de pouvoir faire un reset, ne serait-ce que parce que l'un de vos programmes, lui, se sera planté.

Bien sûr, en cas de blocage du Raspberry Pi il est toujours possible de couper l'alimentation, mais c'est tout de même assez peu pratique car, par manque d'interrupteur, il faut débrancher puis brancher à nouveau le bloc secteur.

Par ailleurs, lorsque vous avez arrêté « proprement » le Raspberry Pi, ce qui s'obtient rappelons-le au moyen de la commande :

```
pi@raspberrypi ~ $ sudo shutdown -h now
```

ce dernier reste alimenté mais devient totalement passif, et il n'est possible de le réveiller ni au moyen du clavier ni avec la souris puisque les ports USB correspondants sont inactifs. Dans une telle situation, un reset lui permet de redémarrer proprement, comme après une nouvelle mise sous tension.

Si cette possibilité de reset a été complètement oubliée, n'ayons pas peur de le dire, sur les premières versions de Raspberry Pi, ce n'est plus le cas aujourd'hui et toutes les cartes, depuis la révision 2.0, disposent de deux pastilles permettant d'y souder des fils ou un connecteur à deux broches pour y brancher un poussoir de reset.

Pour connaître la version de votre Raspberry Pi, il existe une commande très simple :

```
pi@raspberrypi ~ $ cat /proc/cpuinfo
```

L'exécution de cette commande doit conduire à l'affichage d'un écran semblable à celui que vous pouvez découvrir figure 6.1, sur lequel il suffit de lire la ligne appelée **Revision**, avant et de se reporter au tableau 6.1.

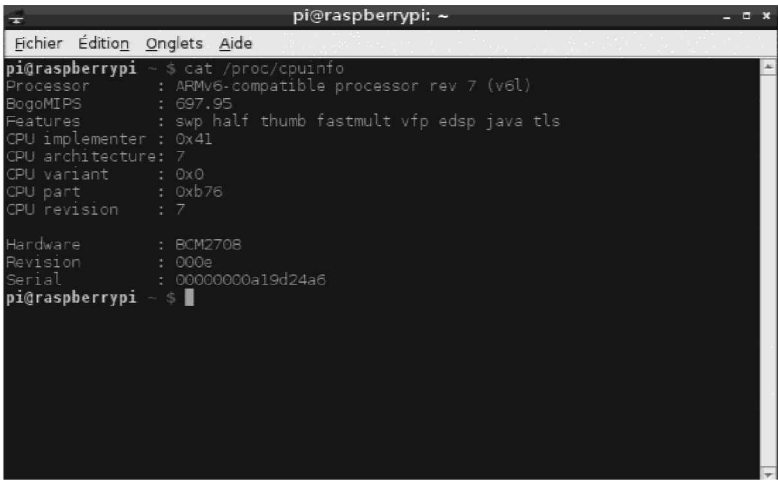


Figure 6.1 – Toutes les informations sur les versions et révisions de votre Raspberry Pi.

Tableau 6.1 – Correspondance entre révision indiquée et révision réelle.

Révision indiquée	Révision réelle
0002	Modèle B Révision 1.0
0003	Modèle B Révision 1.0 sans fusibles au niveau de l'USB
0004 et au-delà	Modèle B Révision 2.0

Notez que, si votre carte n'est pas au moins en révision 2.0, il est impossible d'ajouter ce poussoir de reset. Ce n'est pas que ce soit techniquement compliqué, puisqu'il suffirait de souder sur la carte un condensateur et deux diodes mais, en raison de l'utilisation pour le Raspberry Pi de composants CMS (Composants à Montage en Surface), et compte tenu du fait que le processeur Broadcom est contenu dans un boîtier BGA placé sous le boîtier mémoire, l'accès à la patte du processeur sur laquelle il faudrait ajouter ces éléments est quasiment impossible avec des moyens d'amateur.

Si par contre vous disposez d'une carte en version 2.0 ou au-delà, il suffit tout simplement de souder deux picots mâles au pas de 2,54 mm dans les deux trous prévus à cet effet, à l'emplacement repéré P6, entre le régulateur de tension et la prise HDMI comme indiqué figure 6.2.

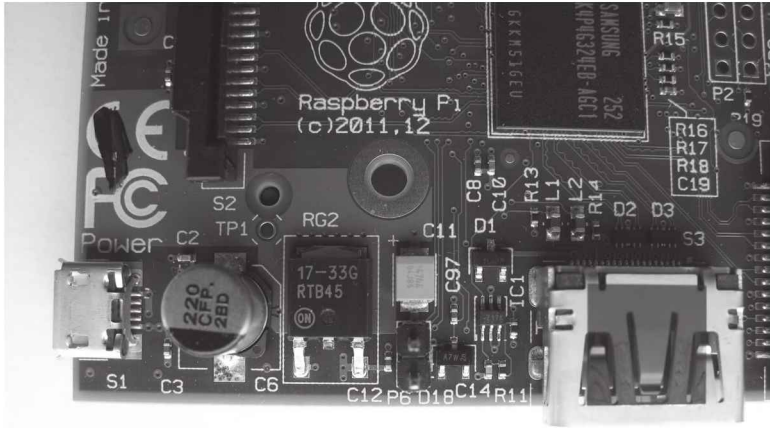


Figure 6.2 – Ajout des picots permettant de brancher un poussoir de reset.

Il ne reste plus qu'à raccorder à ces picots, lorsque c'est nécessaire, un simple poussoir à un contact travail, c'est-à-dire établissant le contact lorsque l'on appuie dessus, pour disposer d'un véritable poussoir de reset.

Pour un usage occasionnel, le poussoir n'est même pas nécessaire car, bien que ce ne soit pas très élégant, il suffit de court-circuiter les picots avec n'importe quel objet métallique pour provoquer le reset.

6.2 Des alimentations pour tous les besoins

Si l'alimentation à partir d'un bloc secteur ou d'un chargeur USB convient tant que le Raspberry Pi est utilisé sur table sans trop de périphériques USB connectés ; il existe un certain nombre de situations où elle ne donne plus satisfaction.

Pour certaines applications, il est nécessaire de pouvoir alimenter le Raspberry Pi sur batteries par exemple, ou bien encore si vous raccordez à son port d'entrées/sorties des interfaces qui consomment beaucoup de courant, telles que des commandes de moteurs, comme nous le verrons au chapitre suivant. Dans ce cas, le chargeur ou le bloc secteur USB ne convient pas et il faut se tourner vers d'autres solutions.

Plutôt que de payer fort cher une alimentation pas forcément bien adaptée, nous vous proposons ci-dessous de réaliser deux modèles différents avec plusieurs variantes, ce qui vous permettra au total de disposer de cinq solutions distinctes, à même de couvrir tous les besoins tout en étant d'un prix de revient dérisoire.

6.2.1 Alimentations 5 volts de 1 à 3 ampères

La solution la plus simple pour réaliser une alimentation pour le Raspberry Pi consiste à faire appel à des régulateurs de tension intégrés de type linéaire. Ces régulateurs sont en effet très simples à mettre en œuvre et ne nécessitent que très peu de composants externes. En contrepartie, ils ont un faible rendement et, si la tension à

partir de laquelle vous voulez alimenter le Raspberry Pi est élevée (batterie 12 volts de voiture par exemple), ils dissipent pas mal de puissance sous forme de chaleur et il faut alors les monter sur un radiateur.

Le schéma d'une telle alimentation, fonctionnant à partir d'une tension continue telle celle délivrée par une batterie de voiture, ou bien encore par un pack de batteries Ni-MH par exemple, est présenté figure 6.3. La diode D1 placée à l'entrée protège le montage d'une inversion de polarité accidentelle qui serait destructrice, tandis que les condensateurs C1, C2 et C3 filtrent la tension appliquée et délivrée par le régulateur. La LED et sa résistance associée, placées en sortie, permettent de s'assurer de la présence de la tension de sortie et donc du bon fonctionnement de l'alimentation. Elles sont bien évidemment facultatives.

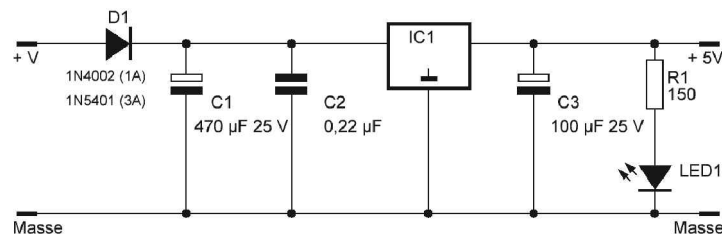


Figure 6.3 – Schéma d'une alimentation à base de régulateur linéaire.

Ce montage peut être réalisé avec trois types de régulateur différents qui conditionnent les paramètres suivants :

- La tension d'entrée minimum nécessaire pour assurer un fonctionnement correct de la régulation et être ainsi certain d'avoir toujours 5 volts en sortie ;
- La tension maximum d'entrée au-delà de laquelle le régulateur risque d'être détruit ;
- Le courant pouvant être délivré par l'alimentation au Raspberry Pi et à ses périphériques.

Tableau 6.2 – Choix du régulateur en fonction des paramètres désirés.

Référence régulateur	Tension d'entrée minimum	Tension d'entrée maximum	Courant de sortie maximum
L7805CP	7,5 V	25 V	1 A
LM2940CT-5.0	6 V	26 V	1 A
LT1085CT-5	5,5 V	20 V	3 A

Le tableau 6.2 synthétise ces informations et vous permet donc de choisir le régulateur correspondant exactement à vos besoins. Les trois références indiquées dans ce

tableau sont très répandues et disponibles chez tout bon revendeur de composants électroniques, ainsi bien sûr que la diode et les condensateurs. Si vous commandez votre Raspberry Pi chez Farnell (<http://www.fr.farnell.com>) qui dispose également d'un certain nombre d'accessoires vus au chapitre 1 (Pi View et clé Wi-Pi par exemple), sachez que vous y trouverez aussi ces composants.

Comme vous ne disposez sans doute pas du matériel nécessaire pour réaliser un circuit imprimé, nous vous proposons de câbler cette alimentation sur une plaquette isolante à bandes cuivrées au pas de 2,54 mm, disponible elle aussi chez tout bon revendeur de composants, en suivant le plan d'implantation de la figure 6.4.

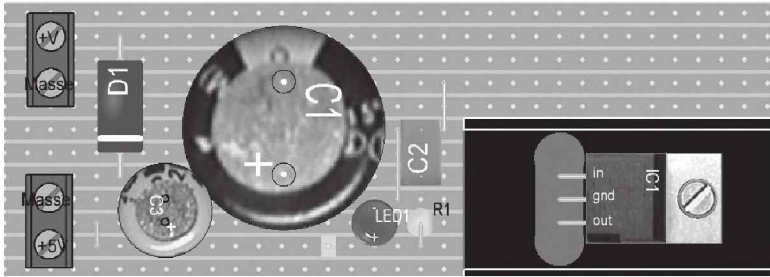


Figure 6.4 – Réalisation de l'alimentation linéaire sur une plaquette à bandes cuivrées.

Selon la tension d'entrée appliquée au montage, il pourra s'avérer nécessaire de visser le régulateur sur un radiateur destiné à assurer son refroidissement ; radiateur qui a d'ailleurs été prévu d'origine sur le plan d'implantation de la figure 6.4. Celui-ci pourra être un simple morceau d'aluminium plié en U ou bien un radiateur spécifique acheté chez un revendeur de composants électroniques.

Attention !

La semelle métallique du régulateur, et donc le radiateur sur lequel elle sera vissée, est reliée à la masse du montage. Veillez à ce qu'elle ne touche pas de parties sous tension du Raspberry Pi.

Quoi qu'il en soit, n'ayez aucune crainte, les trois régulateurs proposés sont protégés contre les échauffements excessifs et s'arrêtent automatiquement lorsqu'ils ont trop chaud. Si cela se produit, c'est donc que votre radiateur est de taille insuffisante.

6.2.2 Alimentation 5 volts 3 ampères à haut rendement

Même si les alimentations précédentes donnent toute satisfaction, ce sont des alimentations dites linéaires et elles ont donc un rendement d'autant plus mauvais que leur tension d'entrée est importante. En effet, elles dissipent cet excès de tension sous forme de chaleur dans le régulateur. Par exemple, si vous utilisez une batterie de voiture qui délivre 12 volts, pour alimenter votre Raspberry Pi sous 5 volts, et que celui-ci consomme 2,5 ampères parce que vous utilisez de nombreux périphériques gourmands, vous allez perdre en chaleur dans le régulateur $(12-5) \times 2,5$ soit 17,5 watts. Presque autant, sinon plus, que ce que consomme l'ampoule à

économie d'énergie qui vous éclaire peut-être... Le rendement global est évidemment désastreux puisque vous absorbez 30 watts sur la batterie ($12 \times 2,5$) pour en restituer seulement 12,5 au Raspberry Pi ($5 \times 2,5$) ce qui nous donne 41,5 % !

Cela n'est pas catastrophique en soi sauf si vous souhaitez bénéficier d'une grande autonomie, auquel cas il faut éviter de gaspiller de la sorte l'énergie fournie par votre batterie. Pour y remédier, il existe fort heureusement une solution, à peine plus compliquée que la précédente, qui passe par la réalisation de ce que l'on appelle une alimentation à découpage. Une telle alimentation affiche un rendement de l'ordre de 80 % au minimum et jusqu'à 90 % dans certains cas, soit nettement mieux que nos alimentations linéaires.

Si ces alimentations étaient compliquées à réaliser il y a encore quelques années, ce n'est plus le cas aujourd'hui grâce aux circuits intégrés de la famille Simple Switchers de Texas Instruments (anciennement National Semiconductor) et le schéma d'une telle alimentation devient alors ce que vous pouvez découvrir figure 6.5.

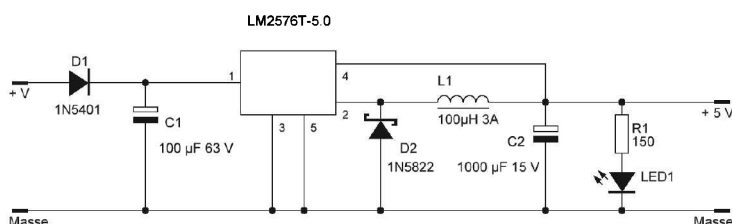


Figure 6.5 – Schéma d'une alimentation à découpage haut rendement 5 volts 3 ampères.

Ce schéma est à peine plus compliqué que le précédent et tous les composants sont, là encore, disponibles chez tous les bons revendeurs de composants.

Attention !

La référence de la diode D2 doit impérativement être respectée ainsi que les caractéristiques de la self L1, faute de quoi l'alimentation risque de mal fonctionner, voire de ne pas fonctionner du tout. Si votre revendeur habituel ne dispose pas de ces deux éléments, vous les trouverez chez Farnell (<http://www.fr.farnell.com>) ou chez Radiospares (<http://radiospares-fr.rs-online.com/web/>).

Comme pour l'alimentation précédente, nous vous proposons de câbler celle-ci sur une plaquette isolante à bandes cuivrées au pas de 2,54 mm en suivant le plan d'implantation de la figure 6.6.

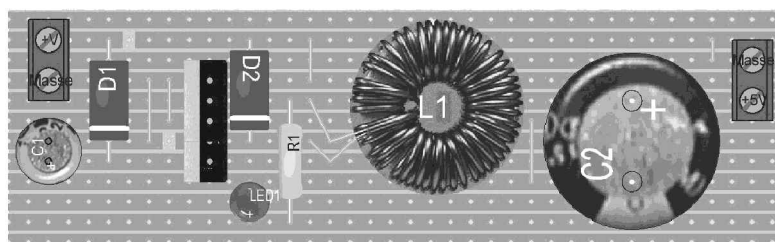


Figure 6.6 – Réalisation de l'alimentation à découpage sur une plaquette à bandes cuivrées.

Si la tension d'entrée appliquée à cette alimentation n'est pas trop élevée, le régulateur IC1 pourra se passer de radiateur car, contrairement à l'alimentation précédente, il ne dissipe que très peu de calories. Dans le cas contraire, un simple rectangle d'aluminium ou bien un radiateur spécifique acheté chez un revendeur de composants électroniques fera l'affaire.

Attention !

Comme pour les régulateurs linéaires, la semelle métallique de ce régulateur, et donc le radiateur sur lequel elle sera éventuellement vissée, est reliée à la masse du montage. Veillez à ce qu'elle ne touche pas de parties sous tension du Raspberry Pi.

Quoi qu'il en soit, n'ayez aucune crainte, ce régulateur est également protégé contre les échauffements excessifs et il s'arrête donc tout seul lorsqu'il a trop chaud. Enfin, précisons que cette alimentation peut admettre en entrée toute tension continue comprise entre 7 et 40 volts ce qui, outre son fort courant de sortie, la rend vraiment polyvalente.

6.2.3 Utilisation à partir du secteur EDF

Les deux schémas précédents sont destinés à être utilisés à partir de tensions d'entrées continues, telles celles produites par des batteries par exemple. Si vous souhaitez les utiliser en remplacement définitif du bloc secteur ou du chargeur USB traditionnel, c'est-à-dire à partir du secteur EDF, il faut les faire précéder d'un transformateur, d'un redresseur et d'un condensateur de filtrage comme indiqué figure 6.7.

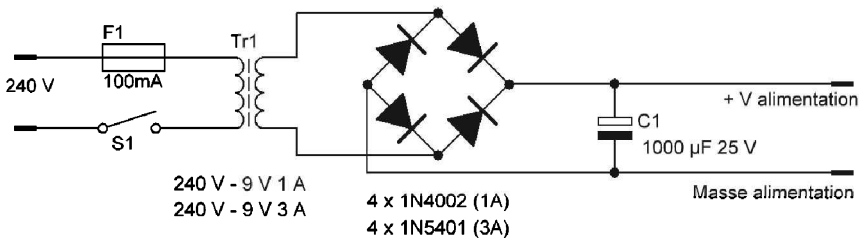


Figure 6.7 – Utilisation des alimentations précédentes à partir du secteur EDF.

La puissance du transformateur est conditionnée par le courant que vous souhaitez pouvoir faire délivrer à votre alimentation. Ce sera donc un modèle 240 V – 9 V 1 ampère pour une alimentation délivrant 1 ampère et un modèle 240 V – 9 V 3 ampères si votre alimentation doit délivrer 3 ampères (peu importe que ce soit le modèle linéaire ou celui à découpage). La même remarque prévaut pour les diodes dont les références sont indiquées figure 6.7 pour les deux cas. Évidemment, « qui peut le plus peut le moins » et vous pouvez donc très bien choisir des composants prévus pour 3 ampères avec un régulateur 1 ampère.

Attention !

Cette partie de la réalisation met en œuvre le secteur EDF et il importe donc d'être très prudent lors de son câblage. Évitez de vous y lancer si vous n'avez aucune expérience des montages électroniques reliés au secteur.

Si vous reliez directement et de façon permanente au moyen de fils soudés l'une des deux alimentations précédentes à la sortie du montage de la figure 6.7, il n'y a plus de risque d'inversion de polarité et la diode D1 qui précède les régulateurs des figures 6.3 et 6.5 est inutile et peut être remplacée par un court-circuit.

6.3 Les entrées/sorties du Raspberry Pi

Contrairement à un ordinateur conventionnel, un des points forts du Raspberry Pi est de permettre d'accéder facilement à un certain nombre de lignes d'entrées/sorties du processeur, lignes sur lesquelles on va en outre pouvoir agir par programme pour commander des éléments extérieurs ou pour recevoir leurs informations.

Ces entrées/sorties ont lieu via ce que l'on a coutume d'appeler le port GPIO, ce qui signifie tout simplement *General Purpose Input Output* ou, en français, entrées sorties à usage général.

6.3.1 Le port GPIO du Raspberry Pi

Ce port GPIO est accessible via le connecteur mâle à 26 points présent sur la carte du Raspberry Pi, connecteur dont le brochage vous est présenté figure 6.8.

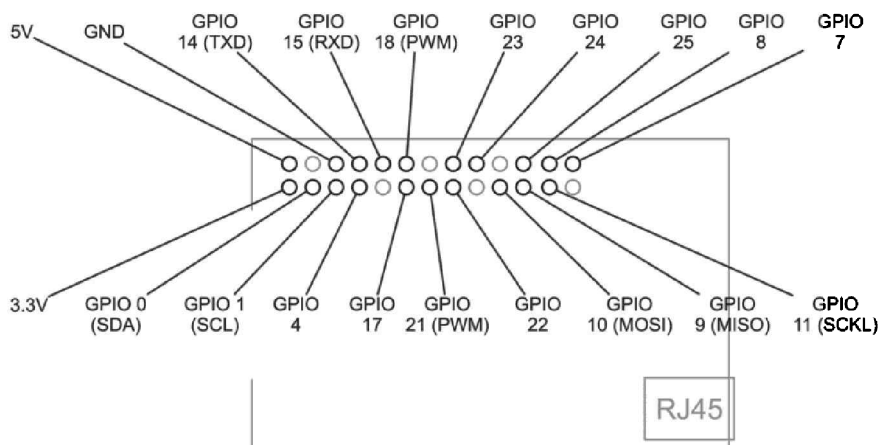


Figure 6.8 – Brochage du port GPIO du Raspberry Pi.

De légères modifications ont été faites au niveau du brochage de ce connecteur entre les versions 1.0 et 2.0 du Raspberry Pi. Elles concernent essentiellement la permutation des ports I2C 0 et I2C 1 du Raspberry Pi au niveau des pattes 3 et 5 de ce connecteur.

Sur les Raspberry Pi version 1.0, ce sont respectivement SDA0 et SCL0 qui arrivent sur 3 et 5 tandis que sur les versions 2.0 et au-delà ce sont maintenant SDA1 et SCL1 qui arrivent sur ces mêmes pattes 3 et 5. Concrètement, cela signifie que si vous voulez utiliser un logiciel qui exploite le bus I2C du Raspberry Pi, il faudra vérifier dans son code source s'il utilise le bus I2C 0 ou le bus I2C 1 et corriger en conséquence en fonction de la révision de la carte Raspberry Pi utilisée.

Ceci étant précisé, on dispose sur ce port GPIO :

- De sept lignes qui peuvent fonctionner en entrées ou en sorties parallèles sans être partagées avec d'autres ressources, ce sont les lignes GPIO 4, 17, 21, 22, 23, 24 et 25.
- D'un UART, ou port série asynchrone, avec sa ligne d'émission de données, ou TXD, en GPIO 14 et sa ligne de réception de données, ou RXD, en GPIO 15.
- D'un bus I2C, appelé aussi bus SMB même s'ils ne sont pas strictement compatibles, avec sa ligne SDA en GPIO 0 et sa ligne SCL en GPIO 1.
- D'un bus SPI avec sa ligne MOSI en GPIO 10, MISO en GPIO 9, CLK en GPIO 11, CE0 en GPIO 8 et CE1 en GPIO 7.
- D'une sortie de signaux PWM (modulation de largeur d'impulsions) en GPIO 18.
- Des deux tensions d'alimentation présentes sur la carte du Raspberry Pi avec le + 5 volts en 2 du connecteur et le + 3,3 volts en 1. La masse correspondante est quant à elle disponible sur la patte 6.

En complément à la figure 6.8, nous vous proposons également la figure 6.9 qui reprend les informations précédentes mais en y faisant figurer les noms des ports GPIO du processeur Broadcom (repère BCM GPIOxx sur cette figure) conjointement aux numéros des broches du connecteur. Ceci vous évitera de fastidieuses recherches en présence de certains programmes que vous trouverez sur Internet qui, tantôt référencent les lignes de port par les numéros de broches du connecteur, tantôt par les numéros de ports du processeur.

			P1			
<50mA	3V3		1	2	5V	
BCM GPIO00/02	SDA0/1	8	3	4	5V	
BCM GPIO01/03	SCL0/1	9	5	6	GND	
BCM GPIO04		7	7	8	15	TX BCM GPIO14
	GND		9	10	16	RX BCM GPIO15
BCM GPIO17		0	11	12	1	PWM0 BCM GPIO18
BCM GPIO21/27		2	13	14		GND
BCM GPIO22		3	15	16	4	BCM GPIO23
<50mA	3v3		17	18	5	BCM GPIO24
BCM GPIO10	SPIMOSI	12	19	20		GND
BCM GPIO9	SPIMOSO	13	21	22	6	BCM GPIO25
BCM GPIO11	SPI SCLK	14	23	24	10	SPI CE0 N BCM GPIO08
	GND		25	26	11	SPI CE1 N BCM GPIO07

Figure 6.9 – Correspondance entre les lignes de ports GPIO du connecteur et les numéros de ports GPIO du processeur Broadcom.

Comme nous le verrons à partir du chapitre suivant, il va donc être possible de relier à ce connecteur GPIO des circuits à interface parallèle, à interface série asynchrone, à bus I2C ou bien encore à bus SPI ce qui va ouvrir de nouveaux horizons à notre Raspberry Pi.

6.3.2 Attention à la tension !

Pour pratique qu'il soit, ce connecteur doit être utilisé avec un minimum de précaution et ce n'est pas ceux qui ont détruit leur Raspberry Pi suite à une fausse manœuvre à son niveau qui nous contrediront ! Voici donc quelles sont les précautions à prendre si vous ne voulez pas détruire le processeur Broadcom, et donc votre Raspberry Pi, en manipulant le port GPIO.

En tout premier lieu, notez que toutes les lignes d'entrées/sorties présentes sur ce connecteur sont reliées directement au processeur, sans aucun élément de limitation de courant ou de protection. Un court-circuit d'une de ces lignes à la masse alors qu'elle est à un niveau de 3,3 volts par exemple, ou l'inverse, ou bien encore l'application d'une tension sur une de ces lignes alors qu'elle n'est pas configurée en entrée mais en sortie peut donc avoir un effet dévastateur certain. Et une fois que le processeur Broadcom du Raspberry Pi est mort, c'est toute la carte qui part à la poubelle !

Notez ensuite que, bien que la carte Raspberry Pi s'alimente sous 5 volts, le processeur fonctionne sous une tension de 3,3 volts pour ce qui est de ses lignes d'entrées/sorties. Cela implique que les niveaux logiques qu'il fournit sur le connecteur GPIO sont des niveaux logiques compris entre 0 et 3,3 volts mais, réciproquement, qu'il ne peut accepter en entrée que des niveaux logiques de la même amplitude. Si vous envisagez d'utiliser des circuits logiques classiques, TTL ou compatibles, fonctionnant sous 5 volts, il faudra impérativement prévoir des adaptateurs de niveaux, au moins dans le sens des entrées du Raspberry Pi afin de ne pas appliquer à celles-ci des tensions supérieures à 3,3 volts.

Notez aussi que les deux broches du connecteur repérées 5 volts et 3,3 volts proviennent des régulateurs intégrés à la carte du Raspberry Pi. Elles peuvent être utilisées pour alimenter un circuit d'interface pour peu que sa consommation reste raisonnable. Aucune indication n'est fournie à ce sujet par la fondation Raspberry Pi mais on peut valablement tabler sur un maximum de 100 mA.

6.4 Des cartes d'expérimentation bien utiles

Dès le chapitre suivant nous allons voir comment utiliser notre Raspberry Pi pour commander des LED, des afficheurs, des moteurs, ou bien encore pour lire l'état d'interrupteurs et de capteurs. Si vous voulez réaliser les expériences correspondantes, tester nos programmes et surtout, réaliser les vôtres, il va vous falloir câbler les composants correspondants.

Nous vous proposons donc dans les lignes qui suivent de découvrir trois des solutions que nous préconisons pour que vous puissiez faire tout cela avec un

maximum de confort et un minimum d'investissement. Et même si les trois produits présentés sont des produits du commerce (vendus en kit toutefois), une réalisation personnelle de leurs équivalents ne présente aucun intérêt car le prix des composants nécessaires, dans le commerce de détail, est supérieur au prix de vente du kit !

6.4.1 Le Pi Cobbler : une excellente idée

Proposé initialement par la société américaine Adafruit (<http://www.adafruit.com>) mais disponible sous nos latitudes, par exemple chez Lextronic (<http://www.lextronnic.fr>) ou bien encore Snootlab (<http://www.snootlab.com>) pour un prix dérisoire, inférieur à 10 euros au moment où ces lignes sont écrites, le Pi Cobbler permet de relier le connecteur GPIO du Raspberry Pi aux plaques de câblage rapides sans soudeuse bien connues qui existent dans de nombreuses dimensions.

Il est ainsi possible de réaliser sur ces plaquettes tous les montages expérimentaux de son choix en les reliant, de manière fiable et sans aucun risque de court-circuit ou de mauvais contact au Raspberry Pi.

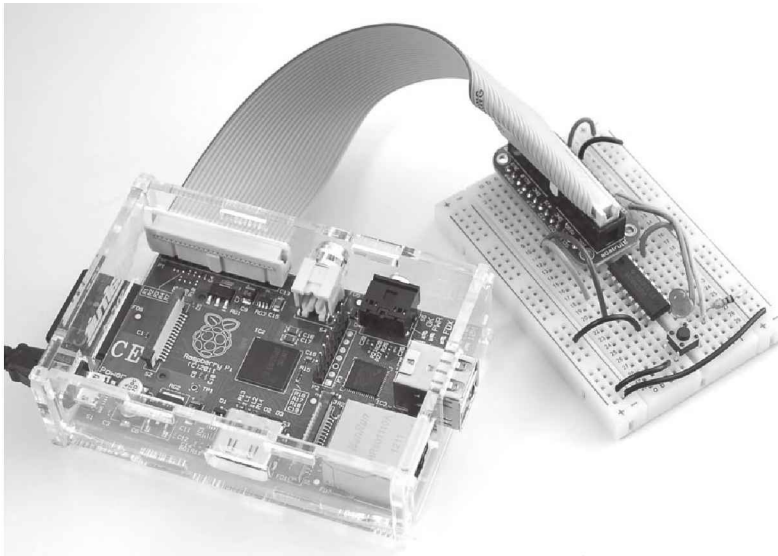


Figure 6.10 – Exemple d'utilisation d'un Pi Cobbler (doc. Adafruit).

En effet, comme le montre la figure 6.10 qui présente un Pi Cobbler en situation, il est composé d'un câble plat avec ses deux connecteurs sertis, dont une extrémité vient s'enficher dans le connecteur GPIO du Raspberry Pi et dont l'autre extrémité s'enfiche dans un petit circuit imprimé, sur un connecteur du même type.

Ce petit circuit imprimé comporte, sur son envers, des pinoches mâles au pas de 2,54 mm qui viennent s'enficher parfaitement dans les plaques de câblage sans

soudures normalisées. Et pour faciliter un peu plus son utilisation, la sérigraphie présente sur ce circuit imprimé rappelle le brochage du connecteur avec le nom des principaux signaux.

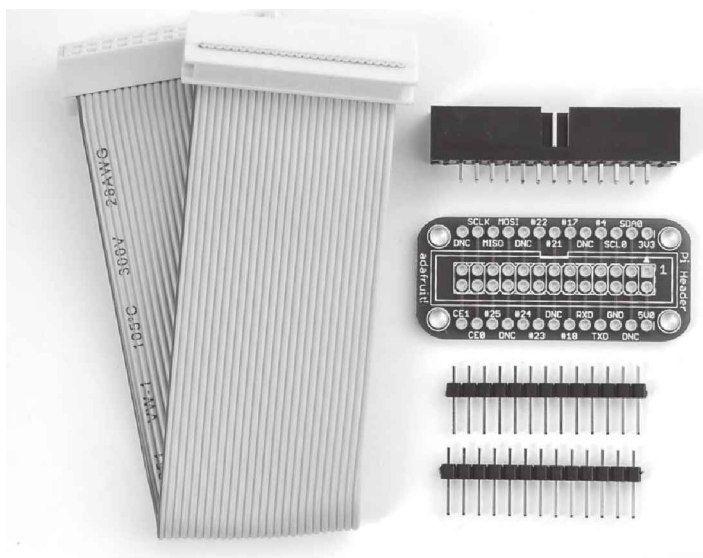


Figure 6.11 – Le kit Pi Cobbler (doc. Adafruit).

Comme le montre la figure 6.11, le Pi Cobbler est fourni en kit mais celui-ci contient tous les éléments nécessaires à sa réalisation qui ne demande que quelques minutes avec les outils évoqués en début de chapitre.

Seule la plaque de câblage sans soudure est à acheter en plus, à moins que vous n'en possédiez déjà au moins une. Si tel n'est pas le cas, elles figurent à la rubrique « Plaques d'expérimentation sans soudure » chez Farnell et « Plaques d'essai sans soudure » chez Radiospares. N'hésitez pas à choisir un modèle de grande taille puisque, grâce au Pi Cobbler, vous pouvez la déporter d'une dizaine de centimètres de votre Raspberry Pi.

6.4.2 La carte Pi Proto Plate

Si vous ne voulez pas déporter vos montages expérimentaux sur une plaque de câblage rapide ou, éventuellement, si vous avez mis un montage au point et que vous souhaitez l'intégrer de manière définitive à votre Raspberry Pi, la carte Pi Proto Plate constitue une alternative intéressante.

Tout comme le Pi Cobbler, elle est proposée par Adafruit (<http://www.adafruit.com>) et est également disponible en France chez Lextronic (<http://www.lextronic.fr>) ou chez Snootlab (<http://www.snootlab.com>) pour moins de 20 euros au moment où ces lignes sont écrites, soit à peine le prix des composants qu'elle supporte.

Cette carte est destinée à s'enficher directement sur le connecteur GPIO du Raspberry Pi comme on peut le constater figure 6.12, mais elle conserve cependant son aspect carte d'expérimentation car elle est pourvue, à sa périphérie, de borniers miniatures à vis sur lesquels sont ramenées toutes les lignes d'entrées/sorties du port GPIO. On peut donc quasiment faire des essais simples sans sortir le fer à souder et c'est parfois très appréciable.

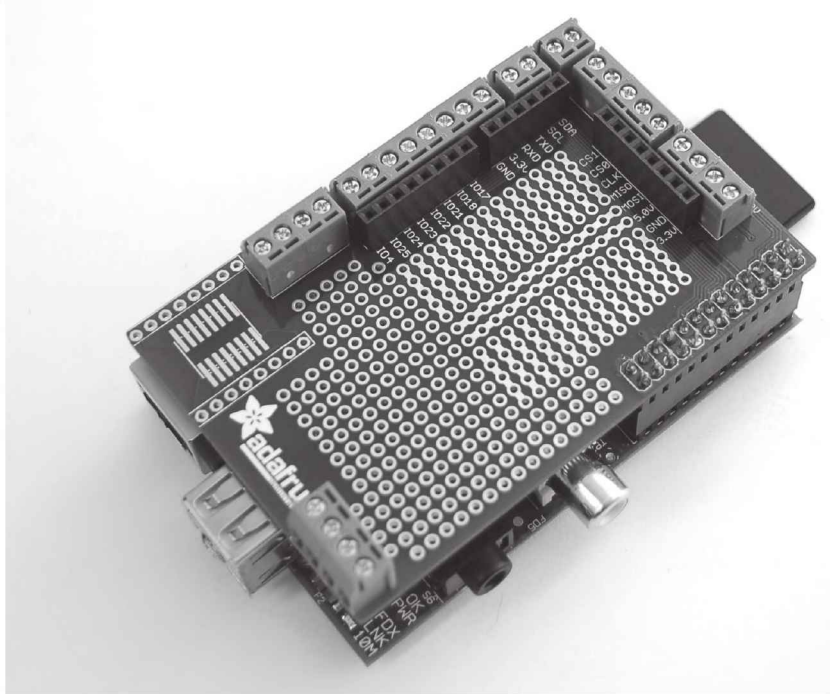


Figure 6.12 – La carte Pi Proto Plate enfichée sur le Raspberry Pi (doc. Adafruit).

Bien qu'elle comporte sur la majorité de sa surface une zone pastillée propre à recevoir des composants soudés, nous vous conseillons de ne pas l'utiliser pour faire vos essais. En effet, le propre de tels essais et de monter et démonter à plusieurs reprises et, après quelques séances de soudage et de dessoudage, il ne restera pas grand-chose des pastilles et des pistes. Les composants soudés sont donc à réserver à un éventuel montage déjà mis au point et devenu, de ce fait, définitif.

Pour utiliser cette carte à titre expérimental, nous vous conseillons de coller, sur la partie centrale pastillée, une plaque de câblage rapide telles celles évoquées précédemment pour le Pi Cobbler, comme cela a été fait par exemple figure 6.13. Il en existe de dimensions suffisamment réduites pour prendre place sur la carte sans difficulté.

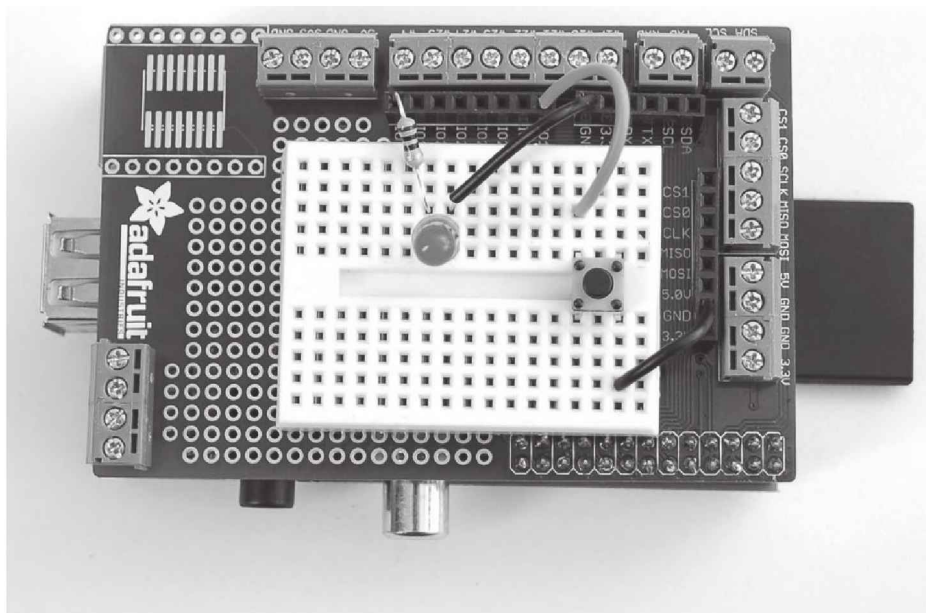


Figure 6.13 – La carte Pi Proto Plate est plus intéressante équipée d'une plaque de câblage rapide (doc. Adafruit).

Par ailleurs, des connecteurs femelles au pas de 2,54 mm sont implantés contre les borniers à vis et sont reliés eux aussi à toutes les lignes du port GPIO. Ils permettent ainsi, au moyen de petits straps en fil rigide, de relier très facilement les lignes de port de votre choix à cette plaque de câblage rapide comme cela a été fait à titre d'exemple sur la figure 6.13.

Comme le Pi Cobbler, cette carte est vendue en kit complet (sans la plaque de câblage rapide toutefois) et son montage ne demande que quelques minutes de fer à souder pour aboutir à un résultat quasiment professionnel.

6.4.3 La carte Humble Pi

Alors que la carte Pi Proto Plate est plus une carte de prototypage, destinée à réaliser des expérimentations sur les interfaces du Raspberry Pi, la carte Humble Pi est plus destinée, à notre avis, à réaliser de façon définitive un montage que vous avez préalablement mis au point, et ce sans avoir à réaliser de circuit imprimé. Contrairement à la carte Pi Proto Plate, elle ne supporte pas en effet de borniers ou de connecteurs femelles donnant accès aux ports du Raspberry Pi. Elle est aussi plus compacte et prend moins de place en hauteur une fois enfichée sur le Raspberry Pi.

Elle a été conçue à l'origine par la société anglaise Ciseco et peut être commandée directement par Internet à cette société à l'adresse <http://shop.ciseco.co.uk/k001-humble-pi/>.

Le prix là encore est dérisoire puisqu'il est inférieur à 5 euros mais il faut dire que, pour ce prix-là, on ne dispose que du minimum vital, à savoir la carte et son connecteur.

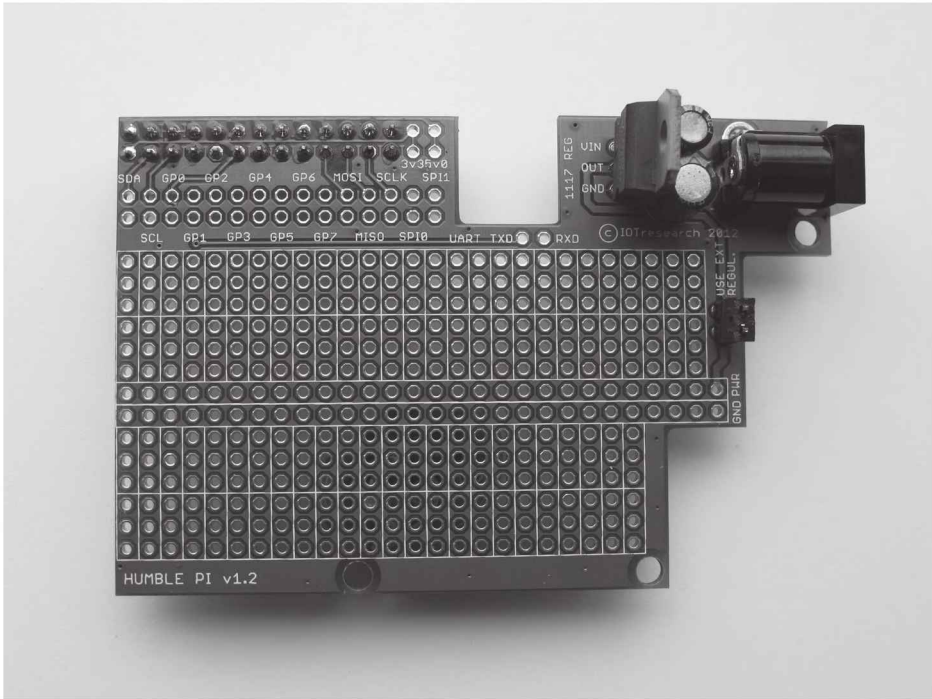


Figure 6.14 – La carte Humble Pi équipée de son régulateur de tension optionnel.

Le connecteur est évidemment à souder « sous » la carte de façon à ce qu'elle puisse s'enficher directement dans le connecteur GPIO du Raspberry Pi qui assure son maintien conjointement à une vis en plastique, également fournie, dont nous cherchons encore le trou de montage...

Cette carte peut également recevoir en option un régulateur de tension 3,3 volts permettant d'alimenter vos créations en vous abstenant de consommer du courant sur celui du Raspberry Pi. Un emplacement est prévu sur le circuit imprimé pour le régulateur, ses deux condensateurs associés, un jack femelle pour brancher la source d'alimentation externe et même un strap à mettre en place sur deux pinoches mâles, pour pouvoir couper ou non cette alimentation sans avoir à souder ou dessouder quoi que ce soit.

Si vous envisagez d'utiliser cette alimentation, nous vous conseillons d'acheter le kit d'alimentation en même temps que la carte car, outre son prix très correct, ses concepteurs ont choisi un régulateur au brochage assez peu répandu, que vous risquez donc d'avoir un peu de mal à trouver.

Armé de l'un ou l'autre de ces trois modules, mais notre préférence va au Pi Cobbler, vous êtes prêt à essayer de connecter votre Raspberry Pi au monde extérieur, non sans avoir bien sûr, lu les chapitres suivants.

7 • POUSSOIR, LED, RELAIS ET AFFICHEURS

Dès que l'on souhaite automatiser une application avec un micro-ordinateur ou un microcontrôleur on s'aperçoit bien vite qu'il passe le plus clair de son temps à communiquer avec le monde extérieur par le biais de ce que l'on désigne sous le nom générique d'entrées/sorties. Bien sûr, ces dernières varient fortement d'une application à une autre. Ainsi, si votre microcontrôleur est l'automate de gestion d'un lave-vaisselle, ses entrées seront des contacts de capteurs divers, et ses sorties des commandes de moteurs et d'électrovannes. Par contre, s'il s'agit d'un thermostat programmable, il utilisera un clavier et un afficheur destinés à communiquer avec l'utilisateur, ainsi qu'une sonde de température et un relais de commande de la chaudière.

Cette diversité des entrées/sorties n'a cependant rien d'inquiétant ni de difficile à maîtriser, en effet, la majorité d'entre elles fait appel à des schémas de base classiques et à leurs programmes associés que nous avons regroupés dans la suite de ces chapitres.

Les solutions que nous allons vous présenter n'ont cependant pas la prétention d'être les seules utilisables ou d'être les meilleures. En effet, ce qui fait la force d'un microcontrôleur c'est qu'il existe bien souvent une multitude de moyens différents pour parvenir au même résultat. C'est à vous de déterminer celui qui vous convient le mieux en utilisant les exemples présentés ici comme tremplins.

Ce chapitre, consacré au dialogue avec le monde extérieur du Raspberry Pi, traite des entrées/sorties parallèles c'est-à-dire des ports GPIO à usage général ou, si vous préférez, non affectés à des interfaces particulières telles que les bus I2C ou SPI par exemple. Les entrées parallèles sont utilisées pour connaître l'état de divers éléments : commutateurs, poussoirs, etc., pour vérifier la présence ou l'absence de tension sur certains dispositifs au moyen, par exemple, de photocoupleurs d'isolement, ou bien encore pour scruter un clavier de commande mis à disposition de l'utilisateur de l'application.

Les sorties parallèles quant à elles sont utilisées pour commander des LED, des relais, des afficheurs, qu'ils soient à base de diodes électroluminescentes ou bien encore de cristaux liquides.

Dans les deux cas, ce sont les interfaces externes les plus faciles à gérer ; il est donc logique que vous commenciez avec elles vos essais. Précisons à ce propos que la majorité des expérimentations proposées dans ce chapitre peut être réalisée avec très peu de câblage, en utilisant par exemple le Pi Cobbler branché sur une plaque de câblage rapide ou bien encore la Pi Proto Plate équipée elle aussi d'une plaque de câblage rapide.

7.1 Une bibliothèque Python indispensable

Puisque Python n'a plus de secrets pour vous, ou presque, c'est avec ce langage que nous allons travailler pour commander nos entrées/sorties, et ce d'autant que de nombreuses bibliothèques sont à notre disposition pour ce faire sur Internet. En outre, comme le Raspberry Pi appartient au monde « du libre » ainsi que la plupart des logiciels qu'il utilise, la majorité de ces bibliothèques sont également mises gracieusement à disposition des utilisateurs par leurs auteurs respectifs.

7.1.1 La bibliothèque RPi.GPIO

Pour gérer les entrées/sorties via les lignes de ports parallèles du Raspberry Pi, la bibliothèque Python RPi.GPIO fait l'unanimité au point qu'elle fait désormais partie intégrante de la distribution Raspbian Wheezy que nous vous avons fait installer au chapitre 1. Si vous utilisez cette distribution, vous n'aurez donc rien à faire de particulier pour pouvoir l'employer, si ce n'est bien sûr de l'importer dans vos programmes.

Attention !

Cette bibliothèque ne peut être utilisée qu'en mode super-utilisateur. Il faudra donc faire précéder tous les appels des programmes Python qui y ont recours de `sudo`. Par ailleurs, cette bibliothèque qui a été écrite à l'origine pour Python 2, fonctionne désormais aussi sous Python 3 contrairement à ce que l'on peut encore lire sur de nombreux sites Internet.

Pour le vérifier, lancez l'interpréteur Python, en mode super-utilisateur donc, et en version 3 puisque c'est celle que nous avons utilisée au chapitre 5, en saisissant :

```
pi@raspberrypi ~ $ sudo python3
```

puis, lorsque le prompt de Python s'est affiché :

```
>>> import RPi.GPIO as GPIO
```

Si aucun message d'erreur ne s'affiche mais que, au contraire, vous obtenez le prompt de Python en retour, la bibliothèque est bien installée. Si vous recevez un message d'erreur, c'est qu'elle n'est pas ou mal installée. Il faut donc réaliser une nouvelle installation en procédant comme suit.

Quittez l'interpréteur Python par un `CNTRL D` ou au moyen de `exit()` et saisissez ensuite les deux lignes de commandes suivantes que nous n'avons plus besoin de commenter à ce stade de l'ouvrage :

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get install python3-rpi.gpio
```

Lorsque c'est fait, procédez à la vérification précédente qui doit cette fois-ci bien se terminer.

Ce module étant installé, vous êtes prêt à faire toutes les expériences que nous allons vous proposer au niveau des ports parallèles et, puisque Python est un langage interprété comme nous l'avons expliqué au chapitre 5, un certain nombre

d'entre elles vont pouvoir être réalisées en « ligne de commande » sans avoir à écrire le moindre programme.

7.1.2 Les principales fonctions de RPi.GPIO

Cette bibliothèque étant incontournable pour la gestion des ports parallèles et quasiment personne n'ayant pris la peine de la documenter sur Internet, nous estimons utile de présenter ici rapidement ses principales fonctions.

■ Attention !

Bien que nous ayons utilisé la version la plus récente de RPi.GPIO au moment de la rédaction de cet ouvrage, il se peut que de nouvelles fonctions soient apparues lorsque vous lirez ces lignes car cette bibliothèque est en constante évolution.

■ **GPIO.setmode(GPIO.BOARD) et GPIO.setmode(GPIO.BCM)**

La bibliothèque GPIO est capable de repérer les lignes d'entrées/sorties du port GPIO par le numéro de broche du connecteur GPIO, si l'on écrit :

```
GPIO.setmode(GPIO.BOARD)
```

ou au moyen du numéro de la ligne d'entrée/sortie telle qu'elle est définie par le processeur Broadcom si l'on écrit :

```
GPIO.setmode(GPIO.BCM)
```

La figure 6.9, vue au chapitre précédent, permet très facilement de passer d'un mode à l'autre mais l'utilisation de l'une ou l'autre de ces deux fonctions évite toute gymnastique intellectuelle.

■ **GPIO.setup(numéro de port, sens)**

Cette fonction permet de préciser le sens de fonctionnement du numéro de port spécifié en entrée si `sens` est remplacé par `GPIO.IN`, ou en sortie si c'est par `GPIO.OUT`. Par exemple :

```
GPIO.setup(23, GPIO.OUT)
GPIO.setup(25, GPIO.IN)
```

■ **GPIO.setup(numéro de port, sens, état initial)**

Cette fonction est une extension de celle permettant de définir le fonctionnement d'une ligne de port en sortie. Elle permet de fixer l'état initial dans lequel elle va se trouver. `sens` est nécessairement égal à `GPIO.OUT` tandis que `état initial` peut être remplacé par `initial=GPIO.LOW` ou `initial=GPIO.HIGH` selon que l'on souhaite un état bas ou un état haut. Par exemple :

```
GPIO.setup(23, GPIO.OUT, initial=GPIO.HIGH)
```

■ **GPIO.setup(numéro de port, sens, résistances de tirage)**

Cette fonction est une extension de celle permettant de définir le fonctionnement d'une ligne de port en entrée. Elle permet en effet, tout en définissant ce sens, de préciser si les résistances de tirage intégrées dans le processeur Broadcom vont ramener par défaut la ligne de port correspondante au niveau haut ou au niveau bas lorsqu'elle n'est pas sollicitée, ou bien encore si elles doivent être désactivées. **sens** est nécessairement égal à **GPIO.IN** tandis que **résistances de tirage** est remplacé par **pull_up_down=GPIO.PUD_UP** pour un rappel au niveau haut, par **pull_up_down=GPIO.PUD_DOWN** pour un rappel au niveau bas et par **pull_up_down=GPIO.PUD_OFF** pour désactiver toute résistance de tirage. Par exemple :

```
GPIO.setup(25, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

■ **GPIO.output(numéro de port, état)**

Une fois qu'une ligne de port a été placée en sortie, cette instruction permet de la mettre au niveau haut si **état** vaut **GPIO.HIGH** ou au niveau bas si **état** vaut **GPIO.LOW**. Par exemple :

```
GPIO.output(23, GPIO.HIGH)
```

Il est également possible de remplacer **GPIO.HIGH** par **True** et **GPIO.LOW** par **False**.

■ **GPIO.input(numéro de port)**

Une fois qu'une ligne de port a été placée en entrée, cette fonction permet de lire l'état de cette ligne dans une variable en écrivant par exemple, pour lire l'entrée numéro 25 dans la variable **A** :

```
A = GPIO.input(25)
```

■ **GPIO.add_event_detect(numéro de port, type)**

Cette fonction, relativement récente, permet de valider une détection automatique de changement d'état sur une ligne qui a été préalablement mise en entrée. Il suffit pour cela de remplacer **type** par **GPIO.RISING** pour détecter un front montant, **GPIO.FALLING** pour un front descendant et **GPIO.BOTH** pour l'un ou l'autre. Par exemple :

```
GPIO.add_event_detect(25, GPIO.RISING)
```

■ **GPIO.remove_event_detect(numéro de port)**

Cette fonction est le complément de la précédente et permet d'arrêter une détection automatique de changement d'état qui aurait préalablement été validée sur une ligne de port. Par exemple :

```
GPIO.remove_event_detect(25)
```

■ GPIO.event_detected(numéro de port)

Cette fonction ne peut être utilisée qu'après avoir validé la détection de changement d'état sur une ligne de port. Elle s'utilise dans une structure de prise de décision sous la forme suivante :

```
if GPIO.event_detect(numéro de port) :
    Bloc d'instructions exécutées
    si un changement d'état a été détecté
Instructions suivantes exécutées dans tous les cas
```

■ GPIO.cleanup()

Comme son nom le laisse supposer, cette instruction réinitialise toutes les entrées/sorties qui ont été préalablement définies, à l'état par défaut suivant : entrée sans résistance de tirage et sans détection de changement d'état. Elle s'utilise tout simplement sous la forme :

```
GPIO.cleanup()
```

Comme vous allez pouvoir le constater avec la présentation des différents éléments que l'on peut relier à un port parallèle dans la suite de ce chapitre, ces fonctions vont nous permettre d'assurer très simplement la majorité du travail nécessaire pour les commander.

Attention !

Compte tenu du mode de fonctionnement du noyau Linux et de l'interpréteur Python, ces fonctions ne sont absolument pas temps réel et ne peuvent donc être employées pour des applications critiques en ce domaine ou devant respecter un timing très précis.

7.2 Les sorties numériques parallèles

Les sorties parallèles sont très fréquemment utilisées dans les applications à base de microcontrôleurs. Elles sont ainsi mises à contribution pour allumer des LED, actionner des relais ou piloter des afficheurs, et encore si l'on s'en tient à leurs fonctions de base car, comme on peut les piloter individuellement de manière très souple, elles peuvent même être utilisées pour échanger des données sous forme série ! Elles sont généralement simples à mettre en œuvre car elles ne requièrent bien souvent que très peu de composants externes.

7.2.1 Commandes de LED et de relais

Nous commençons volontairement par le plus facile avec la commande de LED et de relais qui varie toutefois quelque peu compte tenu du courant que peuvent consommer de tels composants.

En effet, les ports numériques du Raspberry Pi sont des ports dits à fort courant (pour un microcontrôleur s'entend) car ils peuvent délivrer ou absorber jusqu'à 15 mA chacun. Des LED ordinaires peuvent alors être directement connectées sur ces ports avec interposition d'une simple résistance série de limitation de courant.

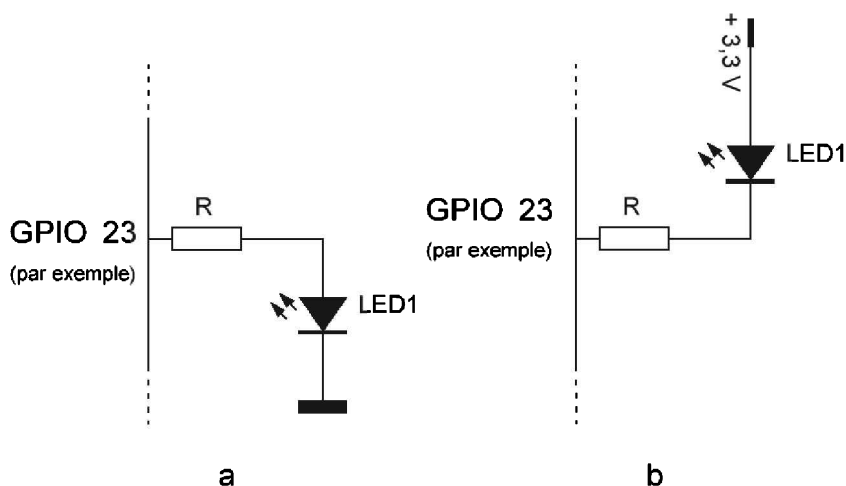


Figure 7.1 – Commande directe de LED par les ports GPIO du Raspberry Pi.

La figure 7.1 montre ainsi à titre d'exemple la commande d'une LED par la sortie 23 du Raspberry Pi ; commande qui peut être réalisée comme indiqué en 7.1a, auquel cas la LED s'allumera pour un niveau logique haut en sortie, ou en 7.1b, auquel cas elle s'allumera pour un niveau logique bas en sortie.

Même si les ports du Raspberry Pi sont limités en courant et si l'on voit parfois sur Internet de telles LED câblées sans résistance série de limitation de courant, c'est une hérésie et cela fait courir un risque inutile au processeur Broadcom du Raspberry Pi.

Une résistance série est donc obligatoire et se calcule très facilement de la façon suivante :

$$R = (3,3 - V_{LED}) / I_{LED} \text{ où}$$

- V_{LED} est la tension de seuil de la LED (1,7 V pour le rouge et l'orange, 2 V pour le vert) ;
- I_{LED} est le courant traversant la LED (exprimé en mA) ; courant qui ne doit en aucun cas excéder 15 mA pour ne pas surcharger les sorties du Raspberry Pi. La majorité des LED ordinaires se satisfait d'ailleurs d'un courant de l'ordre de 10 mA.

Si un affichage très lumineux est nécessaire, ou si l'on souhaite faire appel par exemple à des LED de puissance qui demandent nettement plus de 15 mA pour

fonctionner correctement, il faudra utiliser le montage suivant qui consiste à intercaler sur les sorties du Raspberry Pi, un amplificateur à transistor, même rudimentaire. La figure 7.2 en donne un exemple.

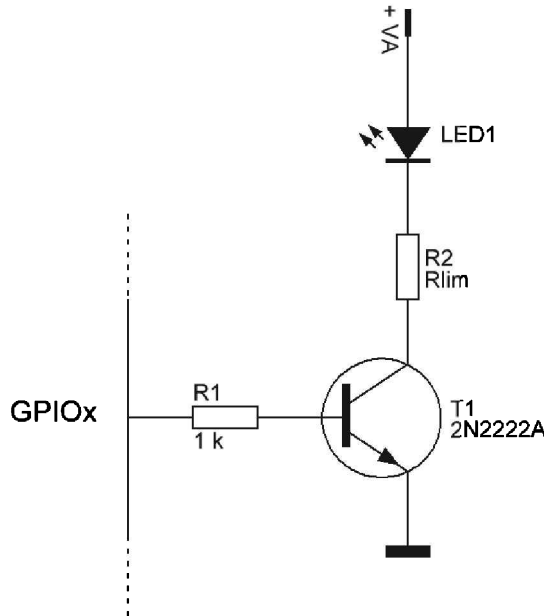


Figure 7.2 – Utilisation d'un transistor amplificateur pour disposer d'un courant de sortie plus important lors de la commande de LED de puissance.

La résistance R_{Lim} de limitation de courant de la LED est à calculer en fonction du courant désiré, comme indiqué ci-dessus, en remplaçant 3,3 par VA qui est la valeur de la tension d'alimentation du montage ; tension qui n'est plus nécessairement égale à celle du Raspberry Pi compte tenu de la présence du transistor.

Compte tenu de la valeur de la résistance de base de ce transistor et de son gain, un courant pouvant atteindre 250 mA peut être fourni par ce montage, ce qui est généralement plus que suffisant pour des LED. Si tel n'est pas le cas, vous découvrirez dans la suite de ce chapitre comment commander des charges consommant plusieurs ampères à partir des sorties du Raspberry Pi.

La commande de photocoupleurs relève bien évidemment des mêmes méthodes que la commande de LED puisque l'organe d'entrée de ces composants est justement... une LED. Tout ce que nous venons de dire reste donc parfaitement valable étant entendu que les résistances de limitation de courant seront calculées compte tenu du courant maximum d'entrée des photocoupleurs utilisés.

Le principe de la commande de relais est très proche de ce que nous venons de voir pour les LED mais, compte tenu du courant important consommé même par les plus petits relais, il faut toujours faire appel à un transistor « de puissance » externe comme indiqué par exemple figure 7.3.

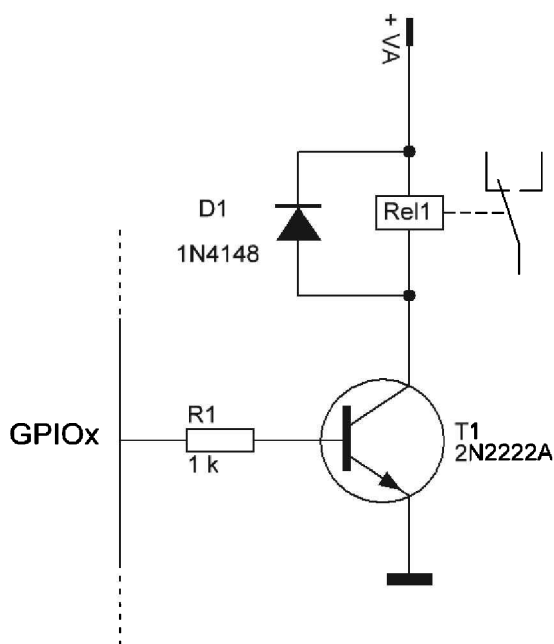


Figure 7.3 – Commande d'un relais au moyen d'un transistor externe.

Comme les relais sont des composants inductifs, il ne faut surtout pas oublier la diode de protection montée en inverse sur la bobine comme indiqué sur cette figure. La tension d'alimentation de ce montage sera bien évidemment égale à la tension de collage du relais. Avec le transistor indiqué ici, tout relais nécessitant un courant inférieur ou égal à 200 mA pourra être utilisé, soit quasiment tous les relais habituellement rencontrés en électronique.

Pour moins de quatre relais, on peut reproduire ce schéma en autant d'exemplaires que nécessaire mais, au-delà, il peut être plus intéressant d'utiliser un ULN 2803 qui, avec ses 500 mA de courant de sortie, peut commander jusqu'à 8 exemplaires de n'importe quel type de relais classique selon le schéma présenté figure 7.4.

Notez que la diode de protection visible figure 7.3 n'a pas besoin d'être prévue car elle est déjà intégrée dans le boîtier de l'ULN 2803 ce qui simplifie encore l'utilisation.

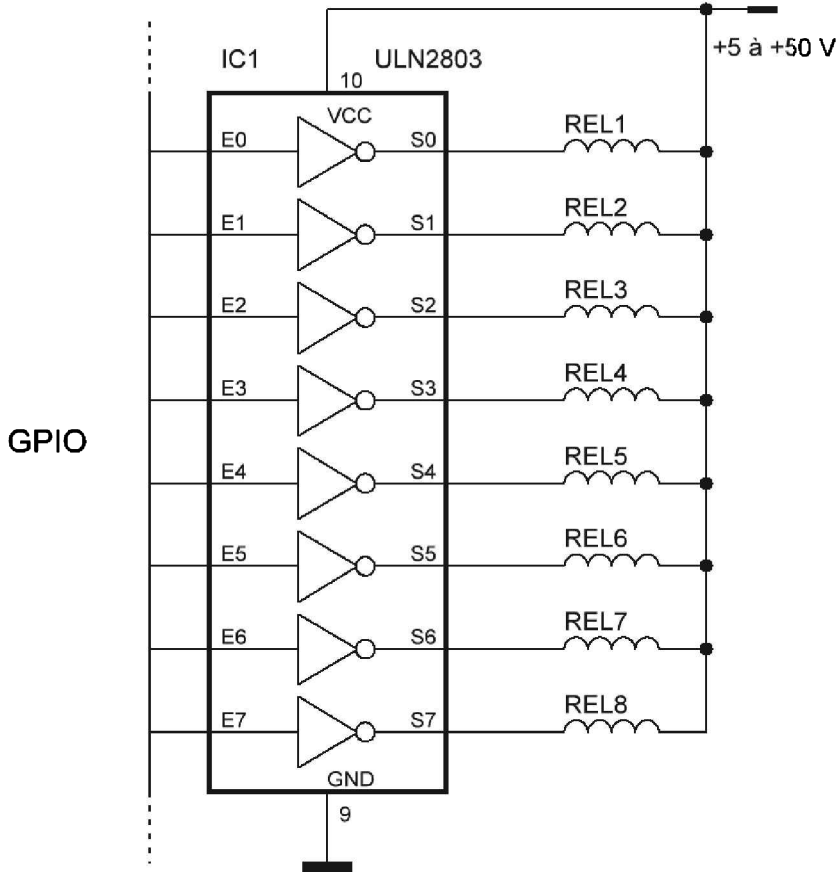


Figure 7.4 – Commande de plusieurs relais au moyen d'un ULN 2803.

Le « logiciel », si l'on peut dire, nécessaire à la commande de toutes ces charges de sortie se résume à bien peu de choses et nous vous proposons de l'essayer en câblant par exemple une LED sur le port 23 du Raspberry Pi conformément au schéma de la figure 7.1b. Choisissez par exemple une LED rouge et donc une résistance de limitation de courant de 150 ohms.

Lancez ensuite l'interpréteur Python en mode super-utilisateur avec un :

```
pi@raspberrypi ~ $ sudo python3
```

Comme le Pi Cobbler utilise la numérotation Broadcom pour sa sérigraphie, nous choisirons de fonctionner dans ce mode dans toute la suite de ce chapitre. Nous saisissons donc, au prompt de l'interpréteur Python :

```
>>> import RPi.GPIO as GPIO
>>> GPIO.setmode(GPIO.BCM)
```

Il faut évidemment mettre la ligne de port correspondante en sortie et on écrira donc :

```
>>> GPIO.setup(23, GPIO.OUT)
```

Ce qui signifie que l'on place la ligne GPIO 23 du processeur Broadcom (celle où nous avons connecté notre LED), c'est-à-dire encore la pinoche numéro 16 du connecteur GPIO (revoyez la figure 6.9 si nécessaire), en sortie.

Si maintenant nous saisissons :

```
>>> GPIO.output(23, GPIO.LOW)
```

Nous mettrons la ligne de port GPIO 23 à l'état bas et allumerons donc la LED tandis que :

```
>>> GPIO.output(23, GPIO.HIGH)
```

mettra cette même ligne à l'état haut et éteindra donc la LED.

C'est aussi simple que cela et cela fonctionne de la même façon pour tous les éléments que nous venons de voir sur les figures précédentes : LED et relais entre autres bien sûr, mais aussi pour des charges beaucoup plus puissantes que nous allons découvrir tout de suite.

Et puisque traditionnellement, le premier programme que l'on écrit lorsque l'on connecte une LED à un microcontrôleur consiste à la faire clignoter, nous vous proposons ci-dessous les quelques lignes de code nécessaires pour ce faire.

Ce code ne pourra pas être exécuté en mode direct sous l'interpréteur Python aussi devrez-vous en faire un programme que vous ferez ensuite exécuter comme nous l'avons vu en détail au chapitre 5.1.2.

```
import RPi.GPIO as GPIO
import time

# Utilisation des numéros de ports du processeur
GPIO.setmode(GPIO.BCM)
# Mise en sortie du port GPIO 23 (broche 16 du connecteur)
GPIO.setup(23, GPIO.OUT)

while True : # Réalisation d'une boucle infinie
    GPIO.output(23, GPIO.LOW)
    time.sleep(1)
    GPIO.output(23, GPIO.HIGH)
    time.sleep(1)
```

Ce programme comportant une boucle sans fin, vous ne pourrez l'interrompre qu'en frappant **CTRL C** au clavier du Raspberry Pi.

Notez par ailleurs l'apparition de la bibliothèque `time`, que nous aurons l'occasion d'utiliser à d'autres reprises dans ce chapitre, et qui permet de disposer d'un certain nombre de fonctions de gestion du temps. Nous utilisons ici sa fonction `sleep` qui permet de mettre en sommeil le programme pendant le nombre de secondes passé en paramètre.

7.2.2 Commande directe de charges alimentées en continu

Même si c'est une pratique que nous déconseillons, tant en raison des problèmes d'isolement qui peuvent se poser, qu'en raison des parasites que cela peut induire sur l'alimentation du Raspberry Pi s'il s'agit de moteurs par exemple, voici néanmoins le principe à utiliser pour actionner directement des charges de forte puissance alimentées en courant continu.

Il suffit tout simplement de faire appel à un transistor Darlington de puissance, ou à un assemblage de transistors réalisant l'équivalent d'un Darlington. Ainsi, comme le montre la figure 7.5 réalisée avec un banal MJ 3001 (mais n'importe quel autre Darlington identique, plus ou moins puissant, peut être utilisé de la même façon), on peut commander, avec n'importe quelle sortie numérique du Raspberry Pi, des courants pouvant aller jusqu'à 10 ampères.

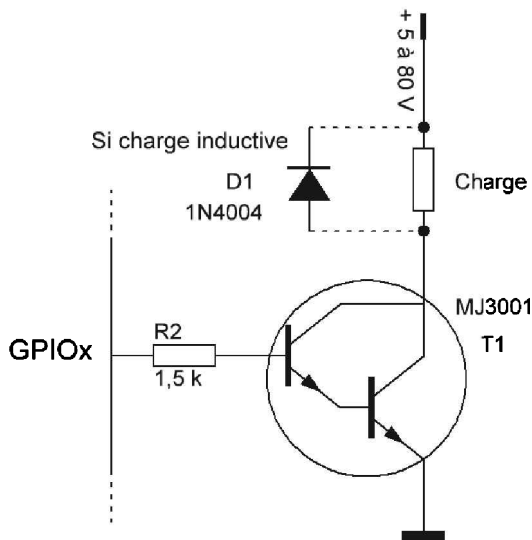


Figure 7.5 – Commande directe d'une charge de puissance alimentée en continu.

Si la ou les charges commandées sont inductives (relais et moteurs par exemple) il est primordial de ne pas oublier la diode de protection évoquée ci-dessus. De plus, pour les moteurs, il peut être nécessaire de prévoir une circuiterie typique d'antiparasitage avec selfs en série et condensateurs de découplage. Chaque cas étant un cas particulier dépendant des caractéristiques précises du moteur, il n'est pas possible de donner ici de schéma général.

Un tel circuit se commande bien évidemment comme les précédents par simple mise au niveau logique haut ou bas de la sortie correspondante du Raspberry Pi.

7.2.3 Commande directe de charges reliées au secteur

Les charges reliées au secteur peuvent évidemment être commandées par des relais qui, contrairement à ce que l'on lit encore trop souvent aujourd'hui, n'ont rien d'anachronique à côté d'un microcontrôleur. Les relais modernes sont des composants fiables qu'il ne faut pas hésiter à utiliser lorsque qu'aucune contrainte particulière ne s'y oppose (fortes vibrations pas exemple).

Lorsque l'on ne peut pas faire appel à des relais classiques ou relais électromécaniques, la seule solution à notre disposition est celle du relais statique, fort heureusement très facile à commander à partir des sorties du Raspberry Pi. En effet, que vous achetiez un relais statique du commerce ou que vous le réalisiez vous-même, il respectera, à de légères variantes près, le schéma de la figure 7.6 qui montre la commande d'une charge pouvant consommer jusqu'à 8 ampères grâce à un MOC 3040 ou 3041 ; photo triac très répandu aujourd'hui.

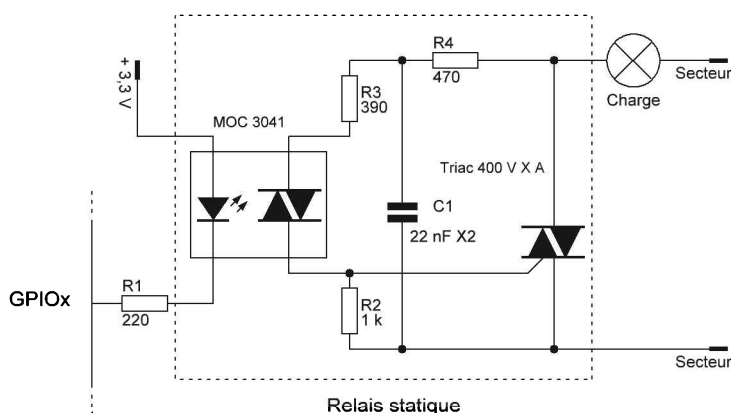


Figure 7.6 – Schéma d'un relais statique commandé par une sortie du Raspberry Pi.

Ce circuit est un photo triac à détection de passage par zéro qui assure ainsi la mise en conduction du triac sans parasite, quel que soit le moment auquel le Raspberry Pi déclenche la commande par rapport aux alternances du secteur. Bien sûr, vous pourriez réaliser la même chose avec un MOC 3020 ou 3021, qui ne possède pas de détection de passage par zéro, et réaliser celle-ci par logiciel. Dans le cas présent, la faible différence de prix entre un MOC 3021 et un MOC 3041 ne justifie toutefois pas à nos yeux le surcroît de travail nécessaire pour réaliser la détection de passage par zéro du secteur par logiciel.

Si vous préférez recourir à un relais statique du commerce, toute la partie encadrée de pointillés sur la figure 7.6 sera contenue dans ce dernier et son mode de connexion au Raspberry Pi ne changera donc pas.

7.3 Les entrées numériques parallèles

Tout comme les sorties parallèles, les entrées parallèles sont parmi les plus utilisées sur un microcontrôleur. En effet, que ce soit pour lire l'état de poussoirs, d'interrupteurs ou de claviers, ou bien encore pour vérifier la présence de tension en tel ou tel point d'une application, ce sont les plus simples à mettre en œuvre, d'autant qu'elles ne requièrent souvent elles aussi que très peu de composants externes.

7.3.1 Poussoirs, commutateurs et capteurs de type tout ou rien

La lecture de l'état de poussoirs, de commutateurs, de roues codeuses (qui ne sont que des commutateurs un peu particuliers) ou de capteurs de type tout ou rien, c'est-à-dire de capteurs qui se contentent de fermer ou d'ouvrir un contact (ILS, ampoule au mercure, etc.) est d'une extrême simplicité, que ce soit avec un Raspberry Pi ou avec n'importe quel autre type de microcontrôleur d'ailleurs.

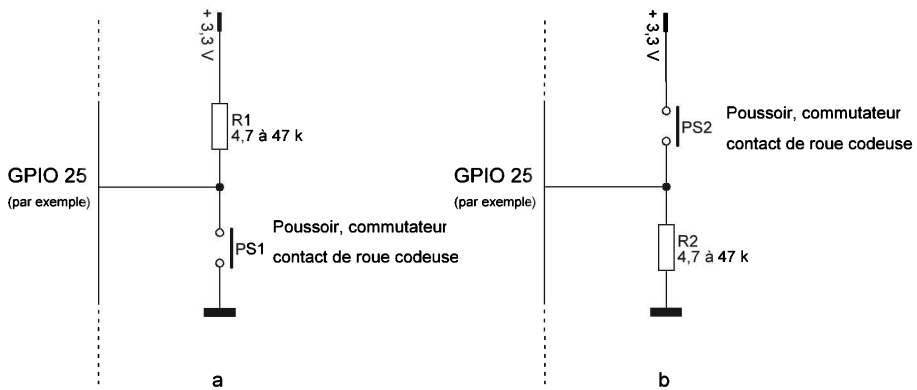


Figure 7.7 – Interface avec un poussoir, un commutateur ou plus généralement tout contact.

Il suffit en effet de relier ce dernier entre une entrée parallèle et la masse comme schématisé figure 7.7a ou entre une entrée parallèle et le + 3,3 volts comme indiqué figure 7.7b. Dans le cas de la figure 7.7a, l'entrée est par ailleurs ramenée au niveau logique haut par une résistance de tirage ou « pull up » pouvant aller jusqu'à 47 ou 100 k Ω si vous avez des impératifs de faible consommation.

Si tel n'est pas le cas, et si vous travaillez dans un milieu susceptible d'être perturbé (voisinage de moteurs par exemple), il est préférable d'utiliser une résistance de valeur plus faible (typiquement 4,7 ou 10 k Ω). En effet, il est plus facile de parasiter un niveau logique haut maintenu par une résistance de 100 k Ω qu'un niveau du même type maintenu par une résistance de 10 k Ω ou de 4,7 k Ω .

Tant que le commutateur est ouvert, l'entrée est vue comme étant au niveau logique haut. Lorsqu'il est fermé, l'entrée est vue comme étant au niveau logique bas. Il

suffit donc de lire l'état de la ou des lignes du port parallèle concerné pour connaître celui de l'interrupteur ou du commutateur, ce qui se fait très facilement comme nous allons le voir dans un instant.

Dans le cas de la figure 7.7b tout ce que nous venons d'écrire reste valable mais c'est évidemment le contraire qui se produit, à savoir un niveau logique bas lorsque l'interrupteur est ouvert et un niveau haut lorsqu'il est fermé.

Ceci étant, et si vous avez le choix, il est toujours préférable de choisir le schéma a plutôt que le b. En effet, si l'interrupteur ou le capteur doit être déporté hors de la carte d'interface du Raspberry Pi, il vaut mieux véhiculer la masse, comme ce sera le cas dans le schéma a, plutôt que l'alimentation + 3,3 volts, comme ce sera le cas dans le schéma b. Un court-circuit éventuel entre les fils de liaison et la masse (boîtier métallique de l'appareil par exemple) est en effet sans danger dans le cas a alors qu'il peut conduire à la destruction de l'alimentation, et donc du Raspberry Pi si c'est lui qui la fournit, dans le cas b.

Enfin, notez que vous pouvez également vous passer totalement de résistance, que vous adoptiez un schéma ou l'autre, en validant tout simplement les résistances de tirage intégrées au niveau des entrées/sorties du processeur Broadcom au moyen de la fonction `GPIO.setup(numéro de port, sens, résistances de tirage)` que nous avons vue précédemment. Pour réaliser le schéma de la figure 7.7a vous validerez les résistances de tirage au niveau haut (`pull_up_down=GPIO.PUD_UP`) et pour le schéma de la figure 7.7b les résistances de tirage au niveau bas (`pull_up_down=GPIO.PUD_DOWN`).

La lecture de l'état de notre poussoir pourra dès lors être réalisée au moyen du programme très simple ci-dessous :

```
import RPi.GPIO as GPIO
import time
# Utilisation des numéros de ports du processeur
GPIO.setmode(GPIO.BCM)
# Mise en entrée du port GPIO 25 (broche 22 du connecteur)
GPIO.setup(25, GPIO.IN)
while True: # Réalisation d'une boucle infinie
    entree = GPIO.input(25)
    if (entree == False):
        print ('Le bouton a été actionné')
        time.sleep(.01)
```

Si vous essayez ce programme, vous constaterez qu'il fonctionne mais fort mal. En effet, chaque appui sur le poussoir fait afficher de multiples lignes indiquant que le poussoir a été actionné quand bien même vous n'appuyez dessus qu'une fois brièvement. Deux phénomènes, normaux lorsque l'on cherche à lire un contact mécanique, viennent en effet perturber ce programme.

Le premier est que le programme s'exécute très rapidement, à notre échelle de temps de pauvre être humain, et que donc la boucle de test du poussoir est exécutée une centaine de fois par seconde, et encore parce que nous l'avons volontairement ralenti au moyen du `sleep(.01)`. Pour vous en convaincre, essayez de réduire ou d'augmenter cette valeur pour voir son effet.

Le second phénomène est causé par ce que l'on appelle les rebondissements.

7.3.2 Comment vaincre les rebondissements ?

Tous les commutateurs sont sujets à ce que l'on appelle des rebondissements, c'est-à-dire que la fermeture ou l'ouverture du contact ne se fait pas immédiatement mais qu'elle est assortie d'une multitude d'oscillations parasites dont la durée peut être de quelques millisecondes. Au lieu de disposer aux bornes d'un tel commutateur qui s'ouvre ou qui se ferme du beau créneau visible figure 7.8a, on dispose alors du train d'impulsions parasites visible figure 7.8b.

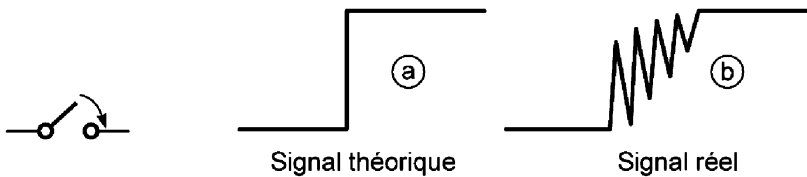


Figure 7.8 – Mise en évidence du phénomène de rebondissement.

Selon l'interprétation de l'état du commutateur qui est faite par le logiciel, cela peut ou non avoir des conséquences fâcheuses et c'est le cas dans le programme précédent qui, à chaque appui sur le poussoir par exemple, voit en réalité plusieurs appuis puisqu'il tourne suffisamment vite pour prendre en compte certains rebondissements.

En logique classique, on élimine les rebondissements des contacts avec des bascules RS, des monostables, ou des cellules RC devant des triggers de Schmitt. En micro-informatique, on peut tenter, avec plus ou moins de bonheur, de se débarrasser des rebondissements par logiciel.

Une première approche simpliste du problème peut consister à accroître de façon importante la durée de sommeil au niveau de l'instruction `sleep` qui termine la boucle mais, dans ce cas, on court le risque de manquer des appuis sur le bouton qui seraient alors trop brefs par rapport à cette durée.

Une deuxième solution, même si elle n'est pas parfaite en l'état actuel de la bibliothèque `RPi.GPIO` (du moins dans la version utilisée au moment où ces lignes sont écrites), consiste à faire appel à la fonction de détection de changement d'état au moyen par exemple du programme suivant.

```
import RPi.GPIO as GPIO
import time
# Utilisation des numéros de ports du processeur
GPIO.setmode(GPIO.BCM)
# Mise en entrée du port GPIO 25 (broche 22 du connecteur)
GPIO.setup(25, GPIO.IN)
# Validation de la détection de changement d'état
GPIO.add_event_detect(25, GPIO.FALLING)
while True: # Réalisation d'une boucle infinie
    if GPIO.event_detected():
        print ('Le bouton a été actionné')
        time.sleep(1)
```

Le fonctionnement de ce programme est déjà nettement plus satisfaisant que le précédent et, sous réserve que vous ne conserviez pas le doigt sur le poussoir pendant un temps trop long, la détection se passe correctement.

Cependant, la meilleure solution et de loin, est celle qui consiste à venir lire la position du bouton au moyen de deux lectures successives séparées de quelques millisecondes afin de s'affranchir de ce phénomène. On réalise ainsi un anti-rebondissement par logiciel. Voici à titre d'exemple un programme qui procède de la sorte. Il est évidemment un peu plus long que la version précédente mais donne toute satisfaction dans tous les cas.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Etat courant du bouton
# Au départ le bouton est supposé au repos
ETAT_BOUTON = False

GPIO.setup(25, GPIO.IN)

while True:
    Lecture1 = GPIO.input(25)
    ETAT1 = not(Lecture1)
    # Si changement d'état, nouvelle lecture du bouton
    if( ETAT1 != ETAT_BOUTON):
        time.sleep(.01)
        # Seconde lecture du bouton
        Lecture2 = GPIO.input(25)
        ETAT2 = not(Lecture2)
        # Si les deux lectures sont identiques
        # le bouton a bien changé d'état
        if( ETAT1 == ETAT2):
            ETAT_BOUTON = ETAT2

# -- Programme principal --
if( ETAT_BOUTON == True):
    print('Bouton appuyé')
else:
    print('Bouton relâché')
```

7.3.3 Entrées haute tension et opto-isolées

Lorsque l'application à base de Raspberry Pi doit prendre en compte des informations disponibles sous haute tension ou des informations liées au secteur EDF par exemple, l'interfaçage direct est vivement déconseillé en raison des dangers d'électrocution qu'il comporte, et la meilleure solution consiste alors à faire appel à un photocoupleur classique à diode ou transistor. Il suffit de le câbler comme indiqué figure 7.9.

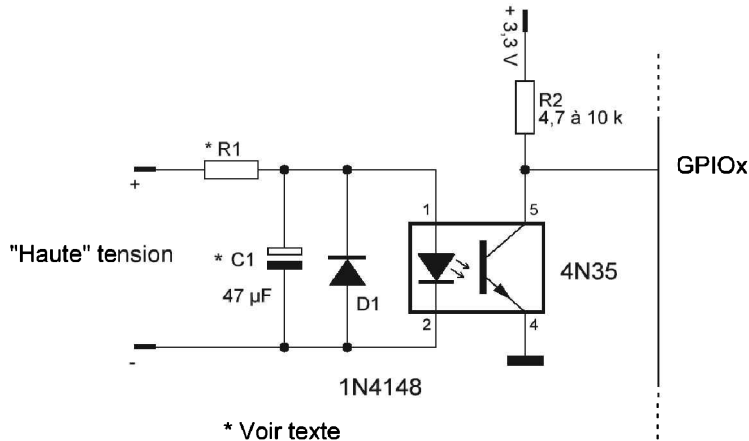


Figure 7.9 – Prise en compte d'information sous haute tension au moyen d'un photocoupleur.

Lorsque le circuit externe est sous tension, ce qui revient à dire qu'il passe du courant du côté LED du photocoupleur, le transistor qu'il contient est saturé et l'entrée du Raspberry Pi est vue comme étant au niveau logique bas (LOW en Python). Lorsque le circuit externe n'est pas alimenté, le transistor est bloqué et l'entrée est vue comme étant au niveau haut (HIGH en Python).

Le photocoupleur peut être n'importe quel modèle classique tel que, par exemple 4N25 ou 4N35. Le condensateur C1 est à mettre en place si l'on utilise ce montage pour détecter la présence ou l'absence de tension de manière statique ; pour voir si un appareil est alimenté par le secteur EDF par exemple.

Par contre, si l'on souhaite utiliser ce montage pour synchroniser le Raspberry Pi avec la fréquence du secteur EDF, il ne faut pas mettre en place C1. On dispose alors aux bornes du transistor d'un signal qui est majoritairement au niveau logique bas et qui passe au niveau haut à chaque passage par zéro de la sinusoïde du secteur, soit exactement 100 fois par seconde.

La résistance R1 est à calculer de façon à ce que le courant traversant la LED du photocoupleur ne soit pas excessif. On peut utiliser pour cela la relation suivante :

$$R1 = (VHT - 1,7) / 0,02$$

où VHT est la valeur efficace de la haute tension.

Le processus de lecture de l'entrée du Raspberry Pi est identique à celui vu précédemment pour un simple interrupteur. Seule la fonction anti-rebondissement peut être absente selon le circuit qui agit sur la LED du photocoupleur.

7.4 Afficheurs à LED et à cristaux liquides (LCD)

On rencontre aujourd'hui sur les applications à base de microcontrôleurs des afficheurs à LED ou plus majoritairement, des afficheurs à cristaux liquides. Ces derniers sont en effet plus intéressants que leurs homologues à LED car ils

consommant nettement moins, ils sont très souvent à même d'afficher des caractères alphanumériques et donc du texte et, à nombre de caractères affichés égaux, ils coûtent nettement moins cher que les modèles à LED.

En outre, dans le cas particulier du Raspberry Pi où le nombre de lignes d'entrées/sorties parallèles est relativement faible, il faut fréquemment recourir à un circuit externe dès que l'on veut commander plus d'un ou deux afficheurs à LED alors que ce n'est pas forcément nécessaire avec les afficheurs à cristaux liquides.

7.4.1 Commande directe d'afficheurs à LED

La commande d'afficheurs à LED utilise à priori les mêmes principes que ceux vus précédemment pour les LED individuelles puisque des afficheurs de ce type ne sont rien d'autre que des groupements de LED de formes adéquates dans un même boîtier. Il faut cependant considérer deux cas selon le nombre d'afficheurs que l'on souhaite commander.

Si l'on n'a qu'un afficheur à piloter et si l'on dispose d'un nombre de lignes d'entrées/sorties suffisant, on peut faire appel au schéma de la figure 7.10. Les segments des afficheurs sont commandés directement par les sorties du Raspberry Pi, puisque ce ne sont rien d'autre que des LED et que le courant maximum qui peut être fourni (15 mA rappelons-le) est suffisant pour peu que l'on utilise des afficheurs à haute luminosité.

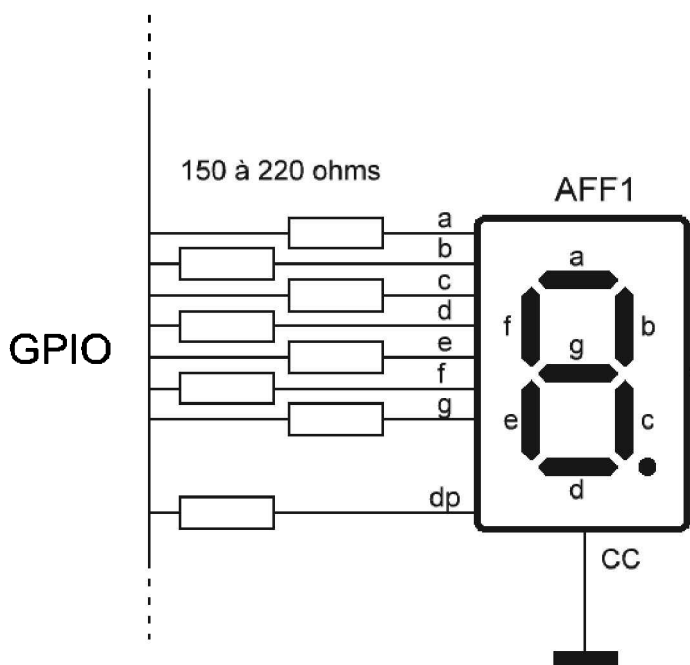


Figure 7.10 – Commande directe d'un afficheur 7 segments à LED.

Un tel afficheur n'étant constitué que de LED, il suffit d'allumer ou éteindre ses segments un à un pour dessiner les chiffres ou symboles à afficher. Il suffit donc de dresser un tableau regroupant tout à la fois les pattes du port GPIO reliées aux segments de l'afficheur et l'état que doivent avoir ces derniers pour dessiner tel ou tel chiffre.

Une fois ce tableau réalisé, il ne reste plus qu'à définir une liste par chiffre à afficher et de faire balayer cette dernière au moyen d'une boucle `for`. Cela reste cependant relativement lourd, surtout que l'on ne gère de la sorte qu'un seul afficheur.

7.4.2 Commande directe d'afficheurs à LED multiplexés

La méthode précédente fonctionne mais présente l'inconvénient de consommer 7 ou 8 sorties du Raspberry Pi (7 si l'on n'utilise que les chiffres et 8 si l'on utilise les chiffres et le point décimal) par afficheur. S'il n'y en a qu'un, cela reste acceptable mais, dès que l'on veut en commander deux, et à plus forte raison plus de deux, cela n'est plus envisageable.

Dans une telle situation, il faut recourir à la technique du multiplexage qui fonctionne comme indiqué figure 7.11, dessinée dans le cas de deux afficheurs seulement mais extensible à l'infini, du moins en théorie.

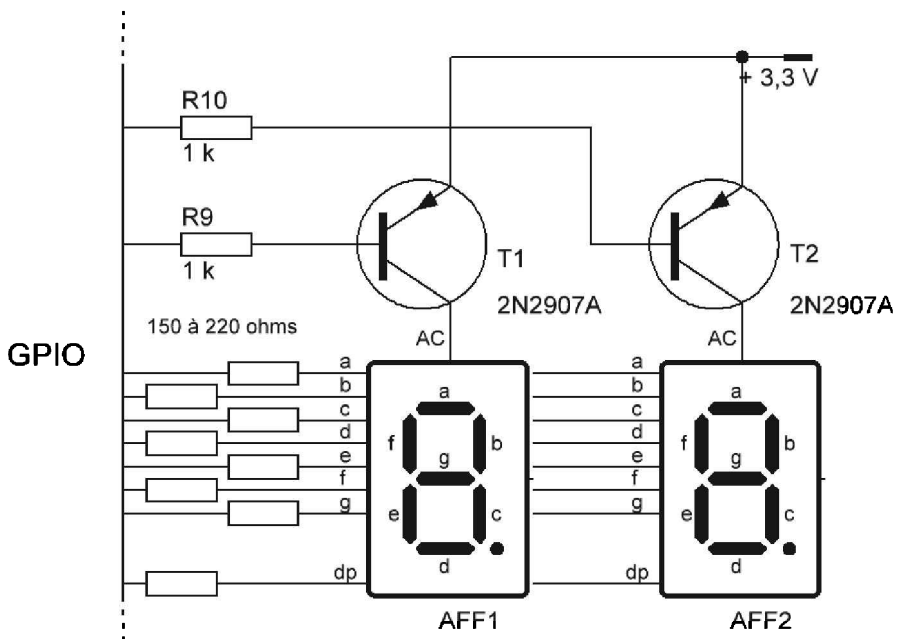


Figure 7.11 – Schéma de principe d'un affichage 7 segments à LED multiplexé.

Les segments des afficheurs sont toujours commandés par les 7 ou 8 sorties du Raspberry Pi comme nous venons de le voir, tandis que les anodes communes des afficheurs (dans cet exemple) sont validées à tour de rôle par des transistors commandés par deux autres sorties. Des afficheurs à cathodes communes peuvent évidemment être utilisés en remplaçant les transistors PNP par des NPN et en reliant leurs émetteurs communs non plus au positif de l'alimentation mais à la masse.

Le principe de fonctionnement du programme de pilotage d'un tel affichage est fort simple. On présente sur les sorties segments le mot binaire correspondant aux segments à allumer comme cela a été fait ci-dessus pour un seul afficheur et on valide ensuite la sortie correspondant au premier chiffre. Après une temporisation dépendant de la fréquence d'appel de ce programme, on invalide le premier chiffre et on reprend la même démarche pour le chiffre suivant.

Les afficheurs sont donc allumés l'un après l'autre mais donnent l'illusion d'un affichage simultané en raison de la persistance des impressions rétinienne. Cette dernière ayant une durée de l'ordre de 20 ms, il faut évidemment que l'allumage de chaque afficheur se répète suffisamment vite pour que l'œil n'ait pas une impression de papillotement.

Le problème de ce mode de fonctionnement est justement lié à cette durée minimum de rafraîchissement qui, si elle est assez facile à respecter pour deux afficheurs et un programme pas trop complexe, tourne vite au cauchemar sur les programmes les plus compliqués. Cela devient même inextricable si le programme doit attendre une action de l'utilisateur, comme l'appui sur un poussoir par exemple. Une bien meilleure solution consiste alors à faire appel à un circuit spécialisé externe, piloté par exemple par le bus I2C ou SPI du port GPIO du Raspberry Pi.

7.4.3 Utilisation d'afficheurs à cristaux liquides (LCD)

Lorsque l'application a des besoins conversationnels qui ne peuvent plus se satisfaire d'un banal affichage de type chiffres ou symboles simples, on fait très souvent appel aux afficheurs alphanumériques à cristaux liquides. On trouve en effet aujourd'hui à bas prix sur le marché des modèles autorisant l'affichage de deux ou quatre lignes de seize à quarante caractères. Qui plus est, ces caractères ne sont plus seulement limités à des chiffres mais à tous les caractères alphanumériques que l'on rencontre sur un clavier de micro-ordinateur.

Ces afficheurs sont toujours fournis sous forme d'un petit circuit imprimé support comprenant tout à la fois l'afficheur lui-même et l'électronique de gestion. Cette électronique facilite la commande de l'afficheur par le microcontrôleur en le déchargeant de toute la partie gestion de l'afficheur et surtout de la génération, tout de même assez complexe, des signaux nécessaires.

Les miracles de la « standardisation automatique » dont sont coutumiers les fabricants d'Extrême Orient font que quasiment tous ces afficheurs utilisent les mêmes contrôleurs, ou des contrôleurs compatibles (contrôleurs souvent appelés de la famille HD44780 du nom du premier à avoir été utilisé). Ils s'interfaçent donc

tous avec les mêmes signaux, qui se gèrent de la même façon, et ils comprennent tous un noyau d'ordres de base communs, même si certains modèles plus évolués que d'autres disposent de quelques commandes ou fonctions supplémentaires. Interfacer et gérer un tel afficheur avec un Raspberry Pi reste une opération très simple comme nous allons le découvrir maintenant grâce aux schémas des figures 7.12 et 7.13.

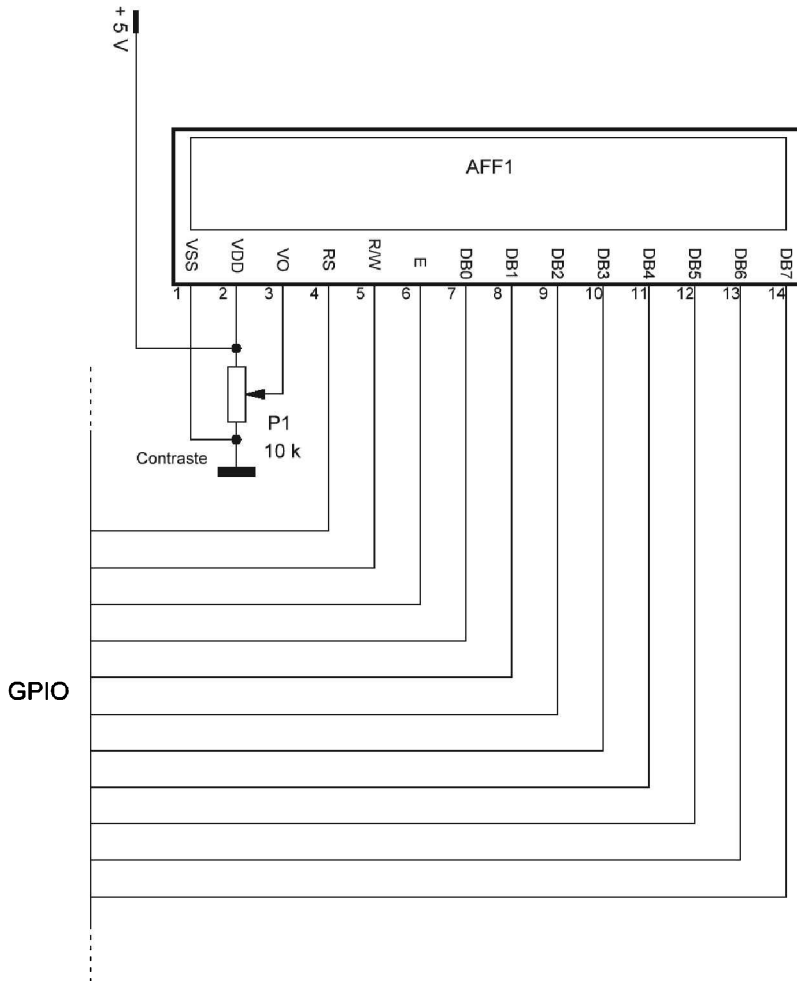


Figure 7.12 – Commande d'un afficheur LCD en mode 8 bits.

Comme vous pouvez le constater, les données échangées avec l'afficheur peuvent être codées sur 8 bits, via les lignes appelées DB0 à DB7, que l'on pilote alors à partir de huit sorties numériques du Raspberry Pi dans le cas de la figure 7.12. On dit alors tout naturellement que l'afficheur fonctionne en mode 8 bits.

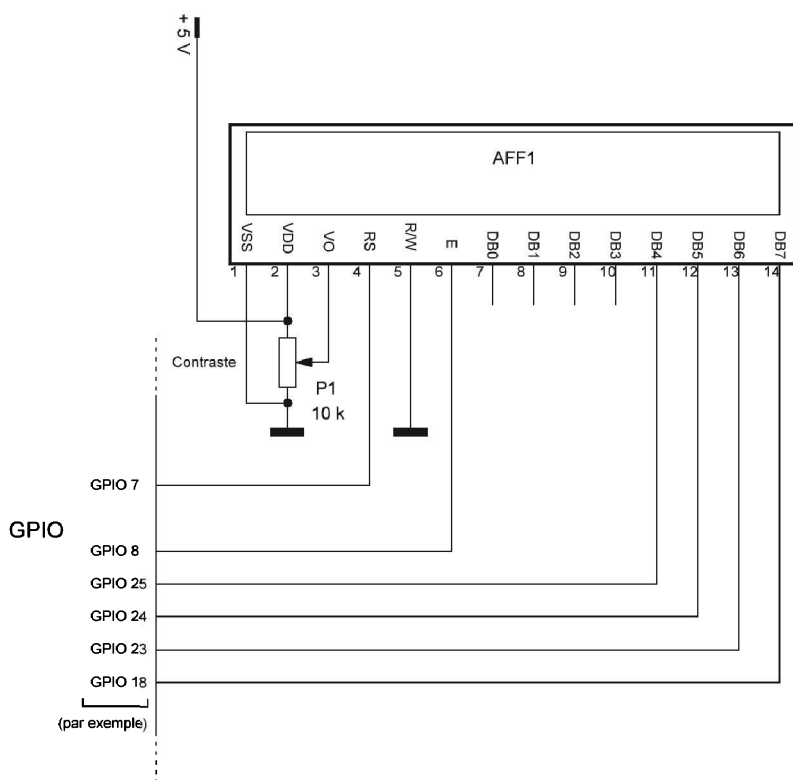


Figure 7.13 – Commande d'un afficheur LCD en mode 4 bits.

Mais tous ces afficheurs peuvent aussi fonctionner comme indiqué figure 7.13, qui correspond à ce que l'on appelle alors le mode 4 bits. Dans un tel mode, les données sont transmises en deux fois sur 4 bits, ce qui consomme bien évidemment moitié moins des précieuses sorties numériques du Raspberry Pi.

Outre ces lignes de données, qui véhiculent tout à la fois les informations affichées et les commandes envoyées à l'afficheur, trois lignes de contrôle sont également nécessaires à son bon fonctionnement.

Ces lignes de contrôle ont les fonctions suivantes, valables pour tous les afficheurs de ce type :

- **E ou Enable** valide l'afficheur lorsqu'elle est au niveau haut. Il est alors à même de recevoir des commandes ou des caractères à afficher via ses lignes de données. Il reste insensible à leur état dans le cas contraire.
- **R/W barre pour Read/Write barre** indique si l'on veut écrire une donnée dans l'afficheur (R/W à zéro) ou lire ses informations (R/W à un). Il dispose en effet d'un registre interne capable de fournir certaines indications d'état. Lorsque l'on souhaite économiser des entrées/sorties, cette ligne est fréquemment mise à la masse, bloquant alors l'afficheur en mode écriture qui est son mode d'utilisation

principal. Il devient alors impossible, dans ce cas, de lire ses registres d'état et sa mémoire de caractères mais cela n'est pas nécessairement contraignant dans la majorité des modes d'utilisation.

- **RS pour Register Select** indique si les lignes de données véhiculent des informations à afficher (RS à 1) ou des commandes de gestion de l'afficheur (RS à 0).

En mode informations à afficher, c'est-à-dire lorsque RS est au niveau logique 1, l'afficheur affiche successivement les caractères de code ASCII correspondant aux données reçues les uns à la suite des autres ; l'avancement de son curseur étant automatique.

Par ailleurs, un certain nombre de commandes permettent une gestion performante de cet affichage : déplacement du curseur de droite à gauche ou l'inverse, avec ou sans effacement de caractère, effacement de tout l'affichage, etc. Ces commandes sont en fait des codes qui doivent être envoyés sur les lignes de données de l'afficheur après avoir mis celui-ci en mode commande (RS à 0). Certains modèles disposent même d'une mémoire de génération de caractères interne dans laquelle vous pouvez stocker les formes de caractères de votre choix.

Le dialogue avec un tel afficheur est fort simple et se passe de la façon suivante dans le cas d'une écriture par exemple :

- mise à zéro de la ligne R/W ;
- positionnement de la ligne RS au niveau désiré selon que l'on souhaite envoyer un caractère à afficher ou une commande ;
- positionnement du code du caractère ou de la commande sur DB0 à DB7 (mode 8 bits) ou DB4 à DB7 (mode 4 bits) ;
- mise à un de la ligne E permettant la prise en compte de ces informations ;
- mise à zéro de la ligne E pour rendre à nouveau l'afficheur insensible à l'état de DB0 à DB7.

Ce processus peut se répéter autant de fois que nécessaire mais, compte tenu de la relative lenteur des afficheurs LCD, un délai d'attente doit être respecté entre l'envoi de deux données ou commandes successives. Sa valeur typique peut aller de 100 µs pour un simple affichage de caractère jusqu'à 2 millisecondes pour les opérations les plus complexes telles que l'effacement complet de l'afficheur par exemple. Une commande de lecture d'état est disponible si nécessaire mais n'est que rarement utilisée.

La réalisation du dialogue avec un tel afficheur ne présente pas de difficultés particulières puisqu'elle se résume à lire ou écrire des données sur les lignes de ports parallèles reliés à l'afficheur, comme nous l'avons fait jusqu'à présent pour commander des relais ou des LED. Cela se révèle toutefois assez fastidieux à la longue, surtout si l'on veut pouvoir exploiter toutes les possibilités offertes par l'afficheur. Fort heureusement, plusieurs auteurs ont mis à disposition, sur leurs sites Internet respectifs, une collection de fonctions permettant de réaliser ces opérations avec un maximum de confort.

Nous vous conseillons par exemple celles de Matt Hawkins, disponibles sur son site Internet d'adresse <http://www.raspberrypi-spy.co.uk/2012/07/16x2-lcd-module-controlling-python/> ainsi bien sûr que via les compléments en ligne de cet ouvrage.

Ces fonctions, à extraire au besoin du programme qu'il fournit en exemple, permettent d'utiliser l'afficheur en mode 4 bits, conformément au câblage que nous avons réalisé figure 7.13. Ce câblage peut cependant être modifié puisque la correspondance entre les lignes de l'afficheur et celles du port GPIO sont définies en début de programme.

Nous vous laissons le soin de prendre connaissance des différentes fonctions présentes dans le programme fourni par Matt à la lumière des explications que nous vous avons données ci-dessus quant au dialogue avec l'afficheur.

Le programme « principal » de son exemple se réduit quant à lui à :

```
# Initialisation de l'affichage
lcd_init()

# Affichage de Raspberry Pi Modèle B
lcd_byte(LCD_LINE_1, LCD_CMD)
lcd_string("Rasperry Pi")
lcd_byte(LCD_LINE_2, LCD_CMD)
lcd_string("Modèle B")

time.sleep(3) # Attente pendant 3 secondes

# Affichage du nom du site Internet de Matt
# dans le cadre de cet exemple
lcd_byte(LCD_LINE_1, LCD_CMD)
lcd_string("Rasperrypi-spy")
lcd_byte(LCD_LINE_2, LCD_CMD)
lcd_string(".co.uk")

time.sleep(20) # Attente pendant 20 secondes
```

Toute la partie « pénible » de la gestion du dialogue avec l'afficheur est réalisée avec les trois fonctions présentes par ailleurs dans la suite du listing fourni sur son site ; fonctions qui comprennent l'essentiel de ce qui est nécessaire avec :

- **Lcd_init ()** qui assure l'initialisation de l'afficheur ; initialisation qu'il est possible d'adapter à vos besoins puisque, dans le cas présent, cette fonction ne fait qu'envoyer à l'afficheur la suite de codes : 33, 32, 28, 0C, 06 et 01 dont nous vous laissons découvrir les fonctions dans la fiche technique de ce dernier, que vous trouverez dans les compléments en ligne.
- **Lcd_byte(bits, mode)** qui envoie à l'afficheur l'octet contenu dans la variable **bits** soit en mode données si **mode** vaut **True**, soit en mode commande si **mode** vaut **False**.
- **Lcd_string(message)** qui envoie à l'afficheur le contenu de la variable chaîne de caractères **message**.

8 • SERVOS, MOTEURS PAS À PAS, MOTEURS À COURANT CONTINU ET ENTRÉES/SORTIES ANALOGIQUES

Après avoir vu, au chapitre précédent, comment s'interfacer avec les incontournables éléments numériques que sont les LED, les afficheurs et les contacts de tous poils, nous abordons maintenant des périphériques moins répandus mais tout aussi utiles, surtout dans des applications telles que la robotique par exemple, puisqu'il s'agit des moteurs.

En effet, même s'il est possible de commander des moteurs avec des relais, eux-mêmes pilotés par des sorties numériques comme nous l'avons vu au chapitre précédent, une telle commande reste rudimentaire et se borne bien souvent à de simples fonctions marche/arrêt et inversion de sens. La diversité des moteurs que l'on rencontre aujourd'hui en électronique et en robotique permet de faire bien mieux en utilisant, selon le cas, des moteurs classiques à courant continu, des servomoteurs initialement conçus pour la radiocommande ou bien encore des moteurs pas à pas. Ce sont ces différentes familles de moteurs que nous vous proposons maintenant d'apprendre à faire tourner avec votre Raspberry Pi.

Enfin, et bien que ce ne soit pas là le point fort du Raspberry Pi, nous aborderons également dans ce chapitre la manipulation de tensions analogiques, ou considérées comme telles, que ce soit en entrée ou en sortie.

8.1 Les servos de radiocommande

Les servos de radiocommande nécessitent un mode d'interfaçage un peu particulier, pas tellement au plan électrique mais plutôt au plan logiciel. Voici tout à la fois le problème à résoudre et sa solution !

Un servo de radiocommande se présente sous un aspect physique similaire à ce que vous pouvez découvrir figure 8.1. Il est généralement fourni avec de nombreux accessoires pouvant être montés sur son arbre de sortie (disques perforés, bras de diverses tailles et formes, etc.) facilitant ainsi sa liaison avec les organes mécaniques environnants.

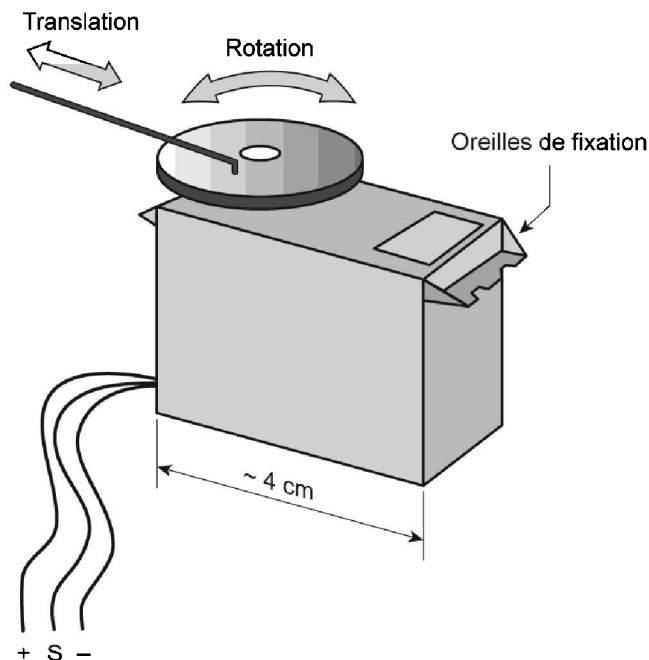


Figure 8.1 – Allure physique d'un servo de radiocommande.

8.1.1 Principe général

Côté électrique, un tel servo ne dispose que de trois fils codés par des couleurs. Les fils rouge et noir sont destinés à son alimentation qui peut être comprise entre 4,8 et 6 volts.

Attention !

Compte tenu de la puissance des moteurs qui équipent les servos, un courant de plusieurs centaines de mA doit pouvoir être fourni par cette alimentation lorsque le servo tourne.

Le troisième fil, de couleur jaune ou blanche (ou de toute autre couleur que rouge et noir en fait), sert à transmettre les ordres au servo sous forme d'impulsions codées en largeur. La figure 8.2 montre le principe de codage de ces impulsions ; principe au demeurant fort simple.

Notez tout d'abord que ces dernières doivent se répéter à un rythme tel que l'on ne doit pas avoir plus de 10 à 20 ms d'écart entre deux impulsions successives.

- une impulsion de 1,5 ms de large place le servo dans sa position dite centrée ou de repos ;

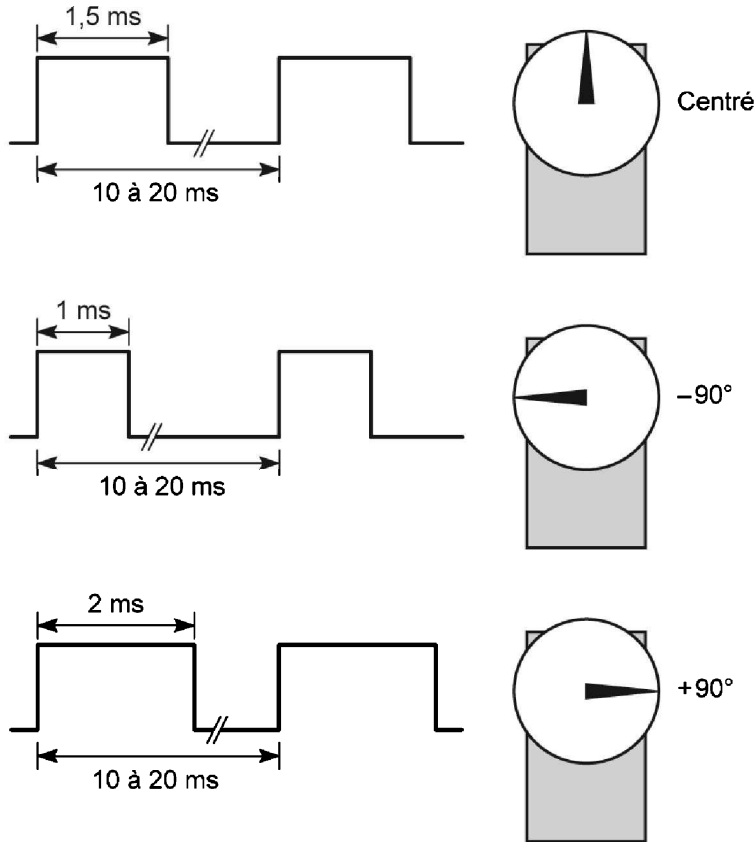


Figure 8.2 – Chronogramme de commande d'un servo de radiocommande.

- une impulsion de 1 ms de large fait tourner le servo dans sa position maximum en sens inverse des aiguilles d'une montre, ce qui représente généralement un angle de 90° par rapport à sa position de repos ;
- une impulsion de 2 ms de large fait tourner le servo dans sa position maximum dans le sens des aiguilles d'une montre ce qui représente, là aussi, un angle de 90° par rapport à sa position de repos ;
- toute autre largeur d'impulsion, comprise entre 1 et 2 ms, permet d'obtenir les innombrables positions intermédiaires.

La connexion d'un servo au Raspberry Pi est évidemment fort simple comme le montre la figure 8.3. La seule précaution à prendre se situe au niveau de l'alimentation du servo dont seule la masse doit être commune avec celle du Raspberry Pi. Par contre, la tension positive ne doit pas être prélevée sur le + 5 volts destiné au Raspberry Pi car, d'une part la consommation du servo est trop importante pour la majorité des blocs secteurs ou chargeurs USB qui alimentent le Raspberry Pi, d'autre part chacun de ses mouvements provoque des appels de courant susceptibles de perturber le Raspberry Pi en causant une baisse anormale de sa tension d'alimentation.

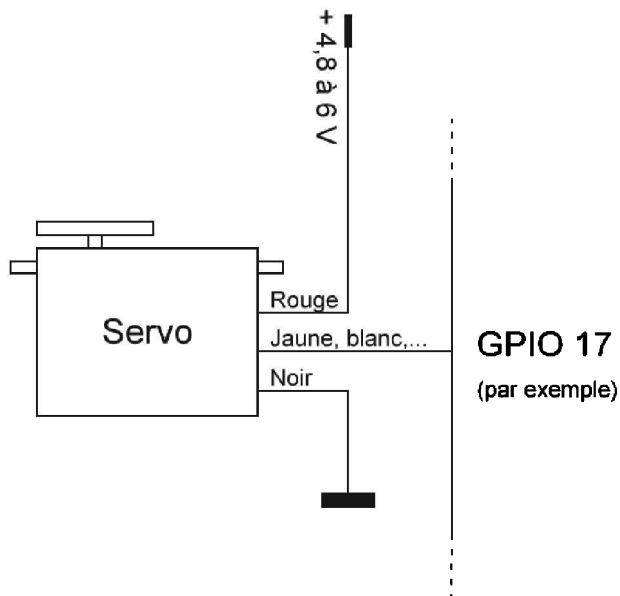


Figure 8.3 – Connexion d'un servo au Raspberry Pi.

Ceci étant précisé, la commande d'un servo avec un Raspberry Pi est fort simple pour peu que l'on fasse appel à une bibliothèque qui facilite la gestion des ports PWM du Raspberry Pi. Ces ports en effet, dont l'appellation PWM signifie *Pulse Width Modulation* ou modulation de largeur d'impulsions en français, permettent de générer des impulsions de largeurs variables tout en déchargeant le processeur du Raspberry Pi de leur gestion une fois que les caractéristiques du signal à produire ont été initialisées.

8.1.2 La bibliothèque RPIO et la commande de servos

La bibliothèque `RPi.GPIO`, que nous avons utilisée jusqu'à présent, ignore totalement la génération de signaux PWM dans l'état où elle se trouve au moment où ces lignes sont écrites. Fort heureusement, on peut trouver sur le site <http://pythonhosted.org> une autre bibliothèque, baptisée `RPIO` qui, outre le fait d'être compatible à 95 % avec `RPi.GPIO`, est également capable de gérer les ports PWM. Nous vous proposons donc d'y faire appel pour réaliser la commande de servos à partir du Raspberry Pi.

La première étape à accomplir consiste à installer cette bibliothèque au moyen des deux commandes suivantes :

```
pi@raspberrypi ~ $ sudo apt-get install python-setuptools
pi@raspberrypi ~ $ sudo easy_install -U RPIO
```

La documentation complète de cette bibliothèque est disponible depuis le site qui l'héberge d'adresse <http://pythonhosted.org/RPIO/#examples> mais, pour ce qui est de la seule commande de servos qui nous occupe ici, nous n'aurons besoin que des fonctions que voici.

■ **set_servo(numéro de port, largeur d'impulsion)**

C'est évidemment la fonction principale qui permet de définir tout à la fois le numéro de port GPIO PWM raccordé au servo, ainsi que la durée de l'impulsion à générer exprimée en microsecondes. Par défaut, la période de répétition de ces impulsions est de 20 ms, soit exactement ce qui convient aux servos de radiocommande standards. Enfin, la résolution de cette fonction est de 10 μ s, ce qui signifie que le paramètre largeur d'impulsion doit évoluer de 10 en 10 uniquement.

■ **stop_servo(numéro de port)**

Cette fonction permet quant à elle, et comme son nom le laisse supposer, d'arrêter toute génération d'impulsion sur le port spécifié.

■ **cleanup()**

Le recours à cette fonction n'est en principe pas nécessaire sauf s'il faut arrêter simultanément plusieurs générations de signaux PWM sur différents ports. Elle a en effet pour but de stopper toute activité PWM sur tous les ports GPIO.

L'utilisation de ces fonctions s'avère donc être d'une grande simplicité comme le montre l'exemple ci-dessous que vous pouvez mettre en œuvre avec le schéma de câblage de la figure 8.3.

```
from RPIO import PWM
import time

servo = PWM.Servo()

# Positionnement du servo raccordé sur GPIO17
# à sa position centrale (1.5ms)
servo.set_servo(17, 1500)
time.sleep(5)

# Positionnement du servo complètement en sens
# inverse des aiguilles d'une montre (1.0ms)
servo.set_servo(17, 1000)
time.sleep(5)
# Positionnement du servo complètement dans le sens
# des aiguilles d'une montre (2.0ms)
servo.set_servo(17, 2000)
time.sleep(5)

# Fin de la génération des signaux
servo.stop_servo(17)
```

Il a pour but de positionner un servo dans sa position centrale pendant cinq secondes, puis sur chacune de ses positions extrêmes pendant également cinq secondes. Il se passe de tout commentaire puisque l'on y retrouve tout simplement les fonctions que nous venons de décrire, utilisées de manière strictement conforme à leur syntaxe. Sous réserve d'utiliser cette bibliothèque `RPIO`, la commande de servos à partir du Raspberry Pi ne présente donc aucune difficulté.

Attention !

Au moment où ces lignes sont écrites, la bibliothèque RPIO n'est disponible que pour la version 2 de Python. Si vous désirez essayer le programme ci-dessus, il faudra donc le faire avec Python et non avec Python3. Par ailleurs, tout comme RPi.GPIO, cette bibliothèque ne fonctionne qu'en mode super-utilisateur, il faudra donc utiliser `sudo python` suivi du nom de votre programme pour en lancer l'exécution.

8.2 Les moteurs pas à pas

Un autre mode d'action mécanique de mise en œuvre très courante en électronique et en robotique consiste à faire appel à des moteurs pas à pas.

8.2.1 Principes généraux

Contrairement aux moteurs à courant continu ou alternatif classiques qui tournent tant qu'ils sont alimentés, les moteurs pas à pas ne tournent que lorsqu'ils reçoivent des impulsions sur leurs bobines ; impulsions qui doivent être présentées dans un ordre bien précis pour faire tourner le moteur dans un sens ou dans l'autre.

Du fait de cette commande impulsionnelle, les moteurs ne tournent pas de manière continue mais avancent en fait à chaque fois d'un pas élémentaire, qui varie selon le type de moteur entre $1,8^\circ$ et $7,5^\circ$ le plus souvent. Cette avance par pas permet au programme qui commande le moteur de connaître très précisément sa position, sous réserve bien sûr que la charge maximum du moteur ne soit pas dépassée, ce qui le conduirait alors à ne pas avancer, même en ayant reçu les impulsions adéquates.

En outre, si le moteur est alimenté mais ne reçoit plus d'impulsions, il reste verrouillé sur la position qu'il a atteinte. On dispose ainsi d'une sorte de frein électrique, toujours sous réserve de ne pas dépasser les possibilités de charge du moteur bien sûr.

Les moteurs pas à pas les plus simples sont les moteurs dits unipolaires à quatre bobines. Pour les commander, il suffit en effet d'appliquer ou non une tension à leurs bobines. Dans l'autre variété de moteurs, appelés moteurs bipolaires, il faut inverser régulièrement la polarité de la tension appliquée aux bobines ce qui complique un peu le schéma à mettre en œuvre.

Tableau 8.1 – Ordre d'alimentation des bobines d'un moteur pas à pas unipolaire.

	Numéro de pas				
	1	2	3	4	5
Bobine 1	1	1	0	0	1
Bobine 2	0	0	1	1	0
Bobine 3	1	0	0	1	1
Bobine 4	0	1	1	0	0

Le tableau 8.1 présente l'ordre dans lequel il faut alimenter les bobines d'un moteur unipolaire pour le faire tourner. Si l'on parcourt les séquences de ce tableau de 1 vers 5, on fait tourner le moteur dans le sens des aiguilles d'une montre, alors qu'un parcours inverse change son sens de rotation. En outre, chaque pas de ce tableau correspond à un pas mécanique du moteur.

8.2.2 Circuits d'interface

Il existe sur le marché divers circuits spécialisés de commande de moteurs pas à pas mais, dans le cas du Raspberry Pi, ils n'induisent ni économie de programme ni économie de composants externes. Nous vous proposons donc une solution beaucoup plus simple, faisant appel à un banal ULN 2803 déjà utilisé au chapitre 7 lors de la commande de plusieurs relais. Le coût de ce circuit est en effet nettement inférieur à celui des circuits spécialisés pour la commande des moteurs pas à pas et le logiciel nécessaire pour le faire fonctionner n'est pas plus complexe.

La méthode de commande la plus simple qui vient à l'esprit en présence d'un moteur pas à pas unipolaire à quatre bobines est celle schématisée figure 8.4.

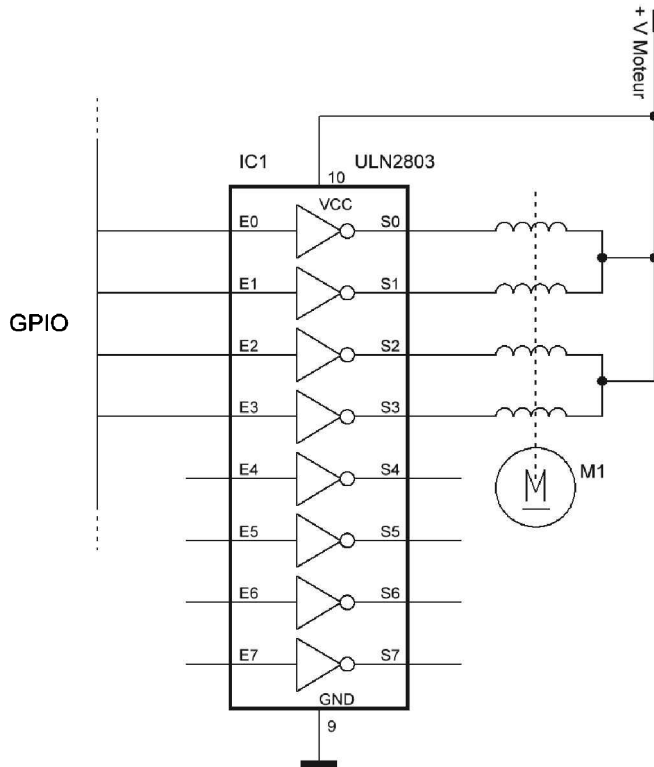


Figure 8.4 – Commande d'un moteur pas à pas unipolaire à quatre bobines au moyen de quatre sorties du Raspberry Pi.

Comme on le voit sur cet exemple, chaque bobine est commandée par une sortie du Raspberry Pi ; l'ULN 2803 faisant office d'amplificateur de courant permettant ainsi de commander des moteurs consommant jusqu'à 500 mA par bobine et dont la tension de fonctionnement peut atteindre 48 volts.

Le logiciel de commande se borne à générer sur les sorties les niveaux indiqués par le tableau 8.1, en le parcourant de gauche à droite ou de droite à gauche selon que l'on souhaite faire tourner le moteur dans un sens ou dans l'autre.

Ce schéma fonctionne mais présente l'inconvénient de consommer quatre sorties du Raspberry Pi alors que l'examen détaillé du tableau 8.1 permet de constater que les états des bobines 1 et 4 d'une part et 2 et 3 d'autre part sont toujours contraires. Ceci permet de commander le moteur pas à pas avec seulement deux sorties du Raspberry Pi comme le montre le schéma fort simple de la figure 8.5. Les amplificateurs libres contenus dans l'ULN 2803 servent en effet d'inverseurs pour commander les bobines 2 et 3 à partir de l'état des bobines 1 et 4.

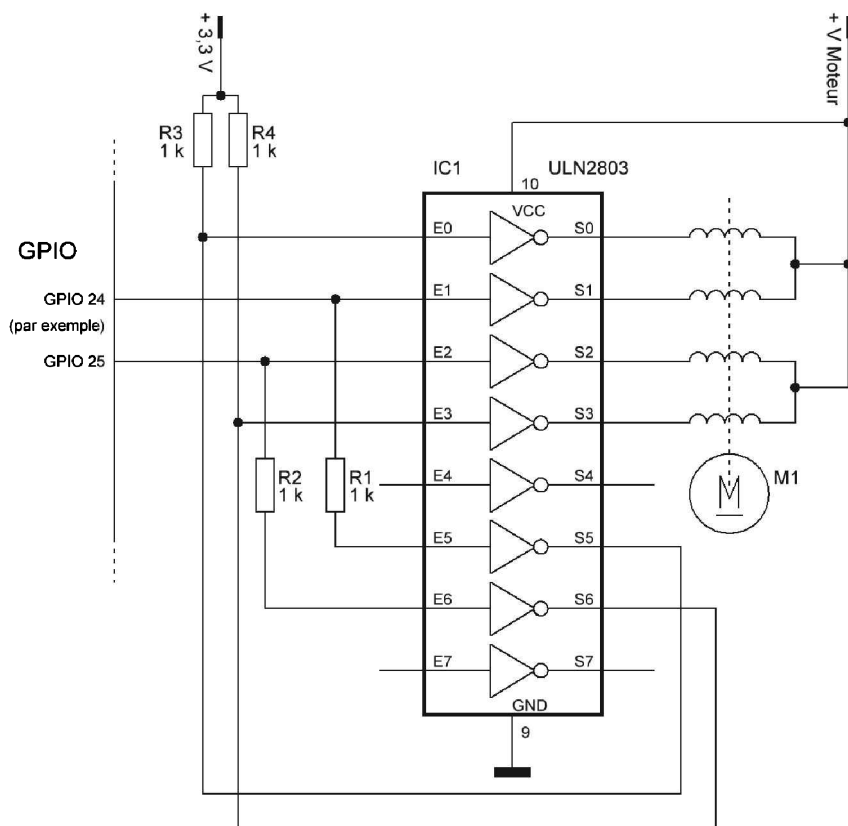


Figure 8.5 – Commande d'un moteur pas à pas unipolaire à quatre bobines avec seulement deux sorties du Raspberry Pi.

Ici aussi, il ne reste plus qu'à générer sur ces sorties les niveaux logiques obtenus en parcourant de gauche à droite ou de droite à gauche le tableau 8.1 selon que l'on souhaite faire tourner le moteur dans un sens ou dans l'autre.

8.2.3 Le logiciel

Aucune bibliothèque spécifique n'est nécessaire pour commander un tel moteur puisqu'il suffit tout simplement de faire sortir sur les lignes de ports parallèles du Raspberry Pi des niveaux logiques conformes à ce qui figure dans le tableau 8.1, au rythme auquel l'on souhaite faire tourner le moteur ; rythme qui dépend tout à la fois de la vitesse de rotation désirée mais aussi des caractéristiques limites du moteur. En effet, si l'on fait générer les pas trop vite, le moteur ne peut plus suivre et il perd des pas.

Le court exemple ci-dessous montre comment commander un moteur pas à pas avec le schéma de la figure 8.5 sans la moindre difficulté.

```
import RPi.GPIO as GPIO
import time

# Utilisation des numéros de ports du processeur

GPIO.setmode(GPIO.BCM)

# Interface moteur sur GPIO 24 et GPIO 25

GPIO.setup(25, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)

# Définition des séquences à générer (tableau 8.1)

seqbob1 = [1, 1, 0, 0]
seqbob2 = [1, 0, 0, 1]

while True :
    for i in range (0,4) :
        GPIO.output(25, seqbob1[i])
        GPIO.output(24, seqbob2[i])
        time.sleep(.01) # Un pas toutes les 10 ms
```

Le circuit d'interface de la figure 8.5 est relié aux ports GPIO 24 et 25 du processeur. Les séquences de signaux, extraites du tableau 8.1, sont définies respectivement dans `seqbob1` pour les bobines 1 et 2 et dans `seqbob2` pour les bobines 3 et 4. La boucle `for` qui suit se charge ensuite de balayer ces séquences grâce à l'index `i` qui évolue de 0 à 3 et, comme on impose un délai d'attente de 10 ms grâce au `time.sleep(.01)` la vitesse de génération des pas est elle aussi fixée à un nouveau pas toutes les 10 ms.

Pour faire tourner le moteur dans l'autre sens, il suffit tout simplement, soit de définir deux autres séquences au contenu inverse de `seqbob1` et `seqbob2` soit, ce qui est encore plus simple, de faire tourner la boucle `for` dans l'autre sens en faisant évoluer `i` de 3 à 0.

8.3 Gradation de puissance et moteurs à courant continu

Lorsque, ni les servos de radiocommande, ni les moteurs pas à pas ne conviennent et qu'il faut donc faire appel à un moteur à courant continu, sa commande avec un Raspberry Pi reste possible mais peut nécessiter, selon le cas, un peu de circuiterie externe.

Pour faire varier la vitesse de rotation d'un moteur à courant continu, la seule solution disponible consiste à faire varier sa tension d'alimentation, ce dont les ports GPIO du Raspberry Pi sont parfaitement incapables.

En revanche, comme nous l'avons déjà brièvement évoqué pour les servos de radiocommande, le Raspberry Pi est capable de générer des signaux PWM c'est-à-dire des impulsions dont la durée est modulable. Sous réserve de savoir exploiter ces signaux, nous allons voir qu'il va être possible de régler de façon très souple la vitesse de rotation d'un moteur à courant continu ou même, pourquoi pas, la gradation de l'intensité lumineuse d'une ampoule.

8.3.1 Variation de tension et signaux PWM

Toutes les charges alimentées sous une tension continue voient leur puissance varier lorsque l'on fait varier leur tension d'alimentation ou, plus exactement, la valeur moyenne de cette dernière. C'est le cas des moteurs électriques à courant continu mais aussi des ampoules à incandescence par exemple.

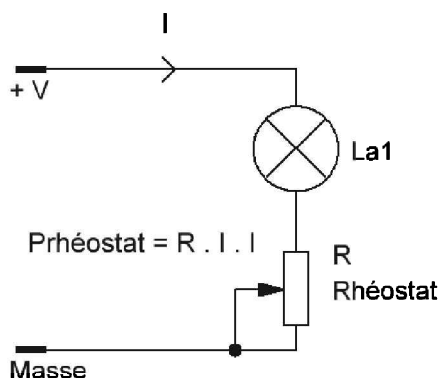


Figure 8.6 – Une méthode de gradation très peu performante.

La figure 8.6 montre comment est réalisée habituellement la gradation de l'éclairage du tableau de bord sur une voiture, choisie à titre d'exemple car elle est très représentative du problème. Un rhéostat, c'est-à-dire un potentiomètre de puissance bobiné, est placé tout simplement en série avec l'alimentation des ampoules.

Cela fonctionne mais équivaut parfois à dissiper en chaleur dans le rhéostat autant de puissance que dans les ampoules. Ainsi par exemple, si le tableau de bord est éclairé par deux ampoules de 5 watts et que vous avez réglé la luminosité à la moitié de sa valeur maximale, les deux ampoules dissiperont à peu près 5 watts (10 watts divisés par 2) mais le rhéostat dissipera aussi 5 watts en pure perte sous forme de chaleur.

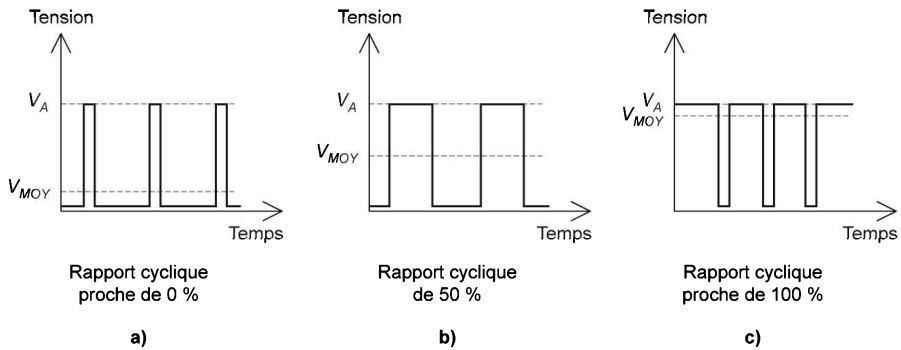


Figure 8.7 – Relation entre valeur moyenne et rapport cyclique de signaux PWM.

Une solution beaucoup plus performante consiste à faire appel à des signaux PWM qui sont constitués par des impulsions à fréquence fixe, d'amplitude égale à la tension d'alimentation totale, mais de rapport cyclique réglable en fonction de la luminosité désirée.

Lorsque les signaux ont un rapport cyclique très faible, comme schématisé en 8.7 a, la valeur moyenne de la tension correspondante est aussi très faible et les ampoules rougeoient tout juste. Lorsque ce rapport cyclique est égal à 50 % comme schématisé en 8.7 b, c'est-à-dire encore lorsque les états hauts et bas des impulsions ont la même durée, la valeur moyenne de la tension est égale à la moitié de l'amplitude totale des impulsions. Les ampoules ont alors une luminosité moyenne. Enfin, lorsque le rapport cyclique des impulsions est proche de 100 % comme schématisé en 8.7 c, la valeur moyenne de la tension est proche de la tension d'alimentation totale et la luminosité des ampoules est maximale.

En première approximation on peut estimer que la tension moyenne est donnée par la relation :

$$V_{MOY} = V_A \times RCYC$$

Où V_A représente l'amplitude maximum des impulsions, $RCYC$ le rapport cyclique exprimé en pour cent, et V_{MOY} la tension moyenne ainsi obtenue.

Comparée à la solution faisant appel au rhéostat, la méthode PWM s'avère beaucoup plus performante car, pour produire des impulsions, on fait travailler un transistor en commutation et sa dissipation de puissance est alors quasi nulle. En effet, lorsque le transistor est bloqué, ce qui correspond aux états hauts des impulsions, il n'est traversé par aucun courant et ne dissipe donc aucune puissance. Lorsque ce même transistor est saturé, ce qui correspond aux états bas des

impulsions, il est traversé par un fort courant (celui d'alimentation des ampoules) mais la tension à ses bornes est très faible car c'est sa tension de saturation qui est en général inférieure au volt, et il ne dissipe donc qu'une puissance très faible.

8.3.2 Logiciel de gradation de tension

Afin que vous puissiez constater de visu l'efficacité de ce procédé, nous vous proposons de réaliser le montage extrêmement simple de la figure 8.8.

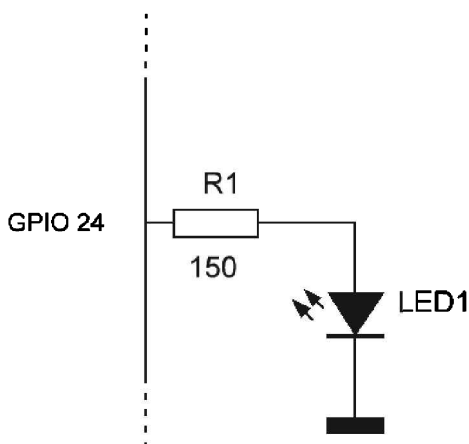


Figure 8.8 – Expérimentation la gradation de tension PWM au moyen d'un LED.

Il permet de régler la luminosité d'une LED, simulant ici une charge plus puissante dont nous verrons dans un instant comment elle peut être commandée, en fonction de l'évolution d'une variable d'un programme, et donc en fonction des désirs de l'utilisateur.

Le programme nécessaire pour parvenir à un tel résultat est d'une grande simplicité grâce à la bibliothèque RPIO que nous avons déjà utilisée précédemment pour la commande de servos.

```
from RPIO import PWM
import time
```

```

servo = PWM.Servo()

# Evolution de la gradation du minimum au maximum
# par pas de 5 % à raison d'un changement toutes les 100 ms

for i in range (0, 20000, 1000):
    servo.set_servo(24, i)
    time.sleep(.1)

# Fin de la génération des signaux
servo.stop_servo(24)

```

Nous utilisons dans ce programme la fonction de la bibliothèque `RPIO` qui permet de générer un signal PWM de période égale à 20 ms et de durée fixée par le paramètre passé à la fonction ; paramètre que nous faisons évoluer ici de son minimum (0) à son maximum 20 000 puisque la période est de 20 ms soit 20 000 μ s. Comme le pas choisi est de 1000, nous obtenons donc une gradation en 20 pas, soit par pas de 5 %. Il serait bien sûr possible de faire moins, sans toutefois descendre en dessous de 10 pour ce pas, puisque la fonction de la bibliothèque `RPIO` le limite à 10 μ s.

Et si la gradation de puissance d'une simple LED ne vous suffit pas, mais que vous souhaitez au contraire commander des charges plus puissantes, il suffit de faire appel à un amplificateur à transistor tels ceux que nous avons décrits au chapitre 7 lors de la présentation des sorties numériques. En effet, les signaux PWM étant des signaux logiques, les schémas que nous y avons présentés restent parfaitement utilisables ici. Ainsi, si vous voulez assurer la gradation de puissance d'une charge pouvant consommer jusqu'à 10 ampères, vous pourrez reprendre sans hésiter le schéma de la figure 7.5.

8.3.3 Variation de vitesse d'un moteur à courant continu

L'exposé que nous venons de réaliser quant à la gradation de tension est intégralement applicable à un moteur à courant continu. En effet, la vitesse de ces derniers ne dépend que de la tension qui leur est appliquée et ils s'accommodent fort bien des signaux PWM. Tout au plus émettent-ils un grognement aux plus basses vitesses dû aux légères vibrations mécaniques que cela engendre s'ils sont très réactifs.

Si le moteur ne doit tourner que dans un seul sens, et que l'on ne souhaite donc pouvoir doser à partir du Raspberry Pi que sa vitesse de rotation, l'utilisation de la gradation PWM vue précédemment convient parfaitement.

Il ne reste qu'à mettre en place sur la sortie du Raspberry Pi qui commande le moteur un amplificateur de courant, tels ceux que nous avons déjà vus à plusieurs reprises et dont les deux principaux schémas d'utilisation sont rappelés figure 8.9.

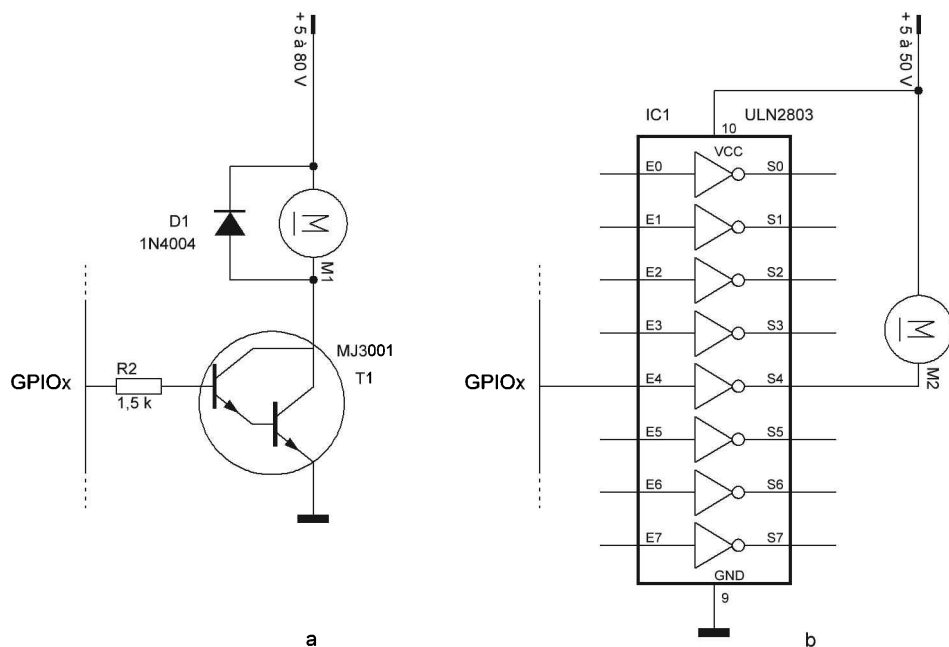


Figure 8.9 – Commande d'un moteur à courant continu avec un seul sens de rotation.

La figure 8.9 b fait appel à un ULN2803, désormais bien connu, dont nous rappelons qu'il peut fournir au maximum 500 mA sous une tension maximum absolue de 50 volts. La figure 8.9 a quant à elle utilise un transistor Darlington de puissance très répandu qui peut commander jusqu'à 10 ampères sous une tension maximum de 80 volts.

Par contre, si le sens de rotation du moteur doit pouvoir être modifié par logiciel, ces schémas ne sont plus utilisables tels quels. Il est bien sûr possible de leur adjoindre des relais pour assurer l'inversion de sens mais, compte tenu de la complexité, de l'encombrement et du surcoût que cela engendre, il est alors préférable de faire appel à un circuit spécialisé au premier rang desquels se classe le L298 de ST Microelectronics.

Ce circuit, dont vous trouverez la fiche technique complète dans les compléments en ligne de cet ouvrage, est un double pilote de moteurs à courant continu capable de commander tout moteur consommant jusqu'à un maximum de 2 ampères sous une tension pouvant atteindre 48 volts.

Il se commande par des signaux logiques et peut donc être relié directement aux sorties du Raspberry Pi et ne nécessite, comme composants externes, que quatre diodes de protection par moteur. La figure 8.10 montre son utilisation typique dans le cas où l'on souhaite pouvoir inverser le sens de rotation d'un moteur et où l'on veut pouvoir doser sa vitesse au moyen de signaux PWM.

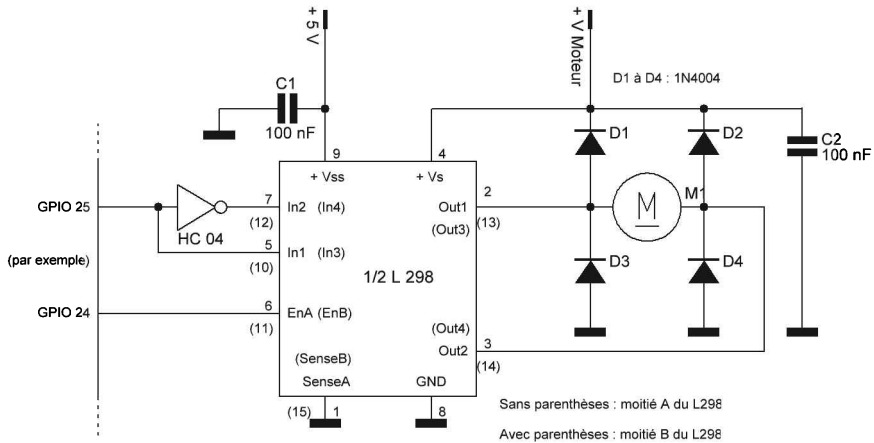


Figure 8.10 – Commande bidirectionnelle d'un moteur à courant continu au moyen d'un L298.

Comme le montre cette figure, on dispose de deux signaux de commande. Celui appliqué sur **In1** permet de déterminer le sens de rotation du moteur selon qu'il est à un niveau logique haut ou bas. Celui appliqué sur **Ena** commande quant à lui la vitesse de rotation du moteur au moyen d'un signal PWM.

Notez dès à présent que le L298 est double et comporte donc, dans un seul boîtier, deux fois le circuit de la figure 8.10. Il peut donc commander simultanément et indépendamment l'un de l'autre deux moteurs à courant continu. C'est donc un circuit idéal pour exploiter, par exemple, une base robotique à deux moteurs telles celles que l'on trouve aujourd'hui couramment sur le marché.

Nous vous proposons ci-dessous un exemple de logiciel de commande d'un tel circuit mais c'est presque inutile car tous les éléments qui le composent vous sont déjà connus. En effet, il se borne à générer un signal PWM sur le port GPIO relié à **Ena** ou **Enb** après avoir mis au préalable à 1 ou à 0 le port GPIO relié à **In1** ou **In3** afin de déterminer le sens de rotation.

```
from RPIO import PWM
import RPIO
import time

servo = RPIO.PWM.Servo()

# Initialisation du sens du port 25 et mise à zéro

RPIO.setup (25, RPIO.OUT, initial=RPIO.LOW)

# Accélération progressive du moteur dans un sens
# par pas de 5 % à raison d'un changement toutes les 100 ms

for i in range (0, 20000, 1000):
```

```
servo.set_servo(24, i)
time.sleep(.1)

# Arrêt du moteur
servo.stop_servo(24)

# Changement de sens de rotation

RPIO.output (25, True)

# Accélération progressive du moteur dans l'autre sens
# par pas de 5 % à raison d'un changement toutes les 100 ms

for i in range (0, 20000, 1000):
    servo.set_servo(24, i)
    time.sleep(.1)

# Arrêt du moteur
servo.stop_servo(24)
```

Remarquez que, bien que nous utilisions la bibliothèque `RPIO` et non `Rpi.GPIO`, nous utilisons les mêmes fonctions pour agir sur la ligne de port GPIO 25 car, comme nous l'avons indiqué lors de sa présentation, la bibliothèque `RPIO` est compatible à 95 % avec la bibliothèque `Rpi.GPIO`.

8.4 Manipulation de tensions analogiques

Le Raspberry Pi est très pauvre en matière de manipulation de tensions analogiques, que ce soit pour leur génération, puisqu'il ne possède pas de convertisseur digital/analogique, ou pour leur lecture puisqu'il ne possède pas non plus de convertisseur analogique/digital.

Comme nous le verrons au chapitre suivant, il est bien sûr possible de relier de tels convertisseurs au Raspberry Pi en exploitant les bus SPI ou I2C mis à notre disposition via son port GPIO mais, si l'on n'a pas besoin d'une grande précision dans la manipulation de ces tensions analogiques, nous allons voir maintenant deux solutions simples.

8.4.1 Génération de tensions analogiques

Nous avons vu, lorsque nous avons parlé ci-dessus de la gradation de tension, que les signaux PWM pouvaient être assimilés, dans une certaine mesure, à une tension continue dont la valeur moyenne dépendait de leur rapport cyclique.

Dans les cas précédents, et que ce soit avec une ampoule à incandescence ou avec un moteur, c'est l'organe commandé qui assure tout seul la conversion du signal PWM en un équivalent de la tension continue qu'il faudrait lui appliquer pour obtenir le même résultat. Pour parler à la façon des électroniciens, il réalise un filtrage passe-bas afin de disposer de la valeur moyenne du signal PWM.

Lorsque l'on ne souhaite pas commander un tel organe mais seulement disposer d'une tension continue variable à destination de tel ou tel équipement. Cela reste possible à partir d'une sortie PWM mais il faut alors réaliser nous-même ce filtrage passe-bas.

Une simple cellule RC, telle celle représentée figure 8.11 a, suffit pour cet usage étant entendu que les valeurs de R et de C doivent être calculées en fonction de la charge connectée aux bornes de C et de la fréquence du signal PWM qui, dans le cas de la bibliothèque `RPi0` est de 50 Hz (période de 20 ms rappelons-le).

Afin d'améliorer la qualité de ce filtrage il est d'ailleurs vivement conseillé de faire appel à un amplificateur opérationnel monté en suiveur de tension comme indiqué figure 8.11 b, ce qui permet de rendre les caractéristiques de la cellule RC indépendantes de la charge.

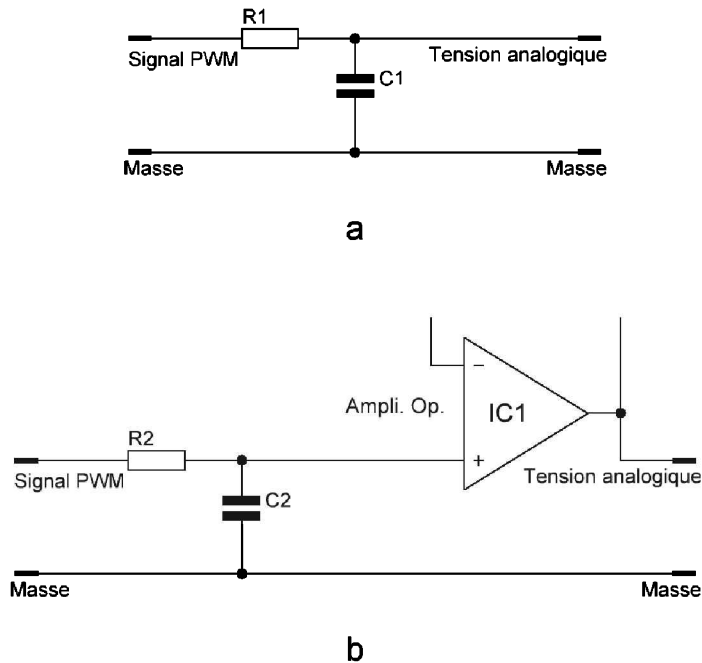


Figure 8.11 – Conversion d'un signal PWM en tension analogique.

Cependant, et même en recourant à l'amplificateur suiveur, une telle conversion ne peut être qualifiée de précise. Si donc votre application nécessite une conversion digitale/analogique rigoureuse, le recours à un convertisseur externe s'avère nécessaire. Vous choisirez donc dans les nombreuses références proposées par les fabricants spécialisés un modèle qui dispose d'une interface par bus I2C ou SPI par exemple, et le problème de conversion sera alors ramené à un simple dialogue sur l'un de ces deux bus qui sont présentés au chapitre suivant.

8.4.2 Acquisition de tensions analogiques

Contrairement à de nombreux microcontrôleurs, dont le célèbre Arduino par exemple, le Raspberry Pi ne dispose d'aucun convertisseur analogique/digital ce qui lui interdit toute lecture précise d'une tension sans l'ajout d'un convertisseur externe.

Dans certaines applications cependant, on n'a pas besoin de la mesure précise d'une tension analogique mais seulement d'une estimation de sa valeur, ou bien encore de pouvoir détecter si elle varie dans un sens ou dans un autre. Cela se produit lorsque l'on fait appel à des capteurs résistifs, dont la résistance varie en fonction d'un paramètre physique extérieur. Les plus connus sont les LDR pour *Light Dependent Resistors* ou résistances dépendant de la lumière, ou bien encore les CTN ou les CTP qui sont respectivement des résistances à Coefficient de Température Négatif ou à Coefficient de Température Positif dont la valeur dépend donc de la température.

La relation liant le phénomène physique à la valeur de la résistance dans de tels capteurs n'est pas linéaire, loin s'en faut, et ils sont donc rarement utilisés pour faire des mesures précises de ces derniers. Par contre, ils font merveille pour détecter la présence ou l'absence de lumière dans le cas de la LDR ou le fait qu'il fasse chaud ou froid, éventuellement par rapport à un seuil déterminé, pour la CTN ou la CTP.

Même si ce n'est pas très élégant techniquement, on peut alors faire appel au montage proposé figure 8.12, que ce soit avec une LDR, une CTN, une CTP ou n'importe quel autre capteur résistif.

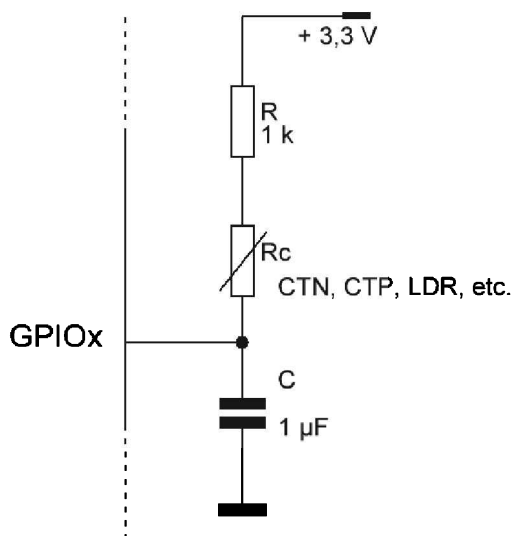


Figure 8.12 – Détection de la variation de valeur d'un capteur analogique résistif.

Son principe est relativement simple et repose sur le temps nécessaire pour charger un condensateur au travers d'une résistance. Si v_a est la tension d'alimentation du

montage (3,3 volts dans le cas du Raspberry Pi), R la résistance placée en série avec le condensateur ($R_1 + R_c$ dans le cas de la figure 8.12) et C la valeur du condensateur, la tension aux bornes du condensateur est donnée par la relation :

$$V_c = V_a \cdot (1 - e^{-(t/RC)})$$

Si vous tracez la courbe correspondante, ou si vous avez quelques réminiscences de vos études, vous constaterez que pour la valeur de t égale à RC , ce que l'on appelle la constante de temps du montage, V_c vaut 63 % de V_a .

Or il se trouve que les ports d'entrées/sorties numériques du Raspberry Pi considèrent qu'ils sont au niveau logique bas lorsque la tension qui leur est appliquée est inférieure à environ 2 volts, tandis qu'ils se considèrent comme au niveau logique haut lorsqu'elle dépasse ces 2 volts. Et il se trouve aussi que 63 % de 3,3 volts, c'est quasiment égal à 2 volts.

Dès lors le principe du montage de la figure 8.12, et donc du programme qui va le gérer, est simple à comprendre : il va suffire de mesurer le temps mis par la patte du port GPIO choisie pour passer de 0 à 1 pour déterminer le temps de charge du condensateur et, par là même, la valeur du produit RC . Et comme C est fixe, la variation de ce temps sera à l'image de la variation de la valeur de R et donc du paramètre physique qu'il représente.

```
import RPi.GPIO as GPIO
import time

# Utilisation des numéros de ports du processeur
GPIO.setmode(GPIO.BCM)

# Définition d'une fonction de mesure
# du temps de charge

def tempscharge (Patte):
    temps = 0

# Décharge du condensateur

    GPIO.setup(Patte, GPIO.OUT)
    GPIO.output(Patte, GPIO.LOW)
    time.sleep(0.1)

# Mise en entrée de Patte

    GPIO.setup(Patte, GPIO.IN)
# Comptage du nombre de tours de boucle nécessaires
# pour que Patte passe au niveau haut

    while (GPIO.input(Patte) == GPIO.LOW):
        temps = temps +1
    return temps

# Exemple d'appel de la fonction

while True:
    print tempscharge(25) # Si utilisation du port GPIO 25
```

L'analyse de ce programme, qui recourt à une fonction de façon à pouvoir appeler celle-ci en plusieurs endroits de son déroulement si nécessaire, est très simple à la lueur des explications qui précèdent.

La fonction commence par décharger le condensateur en mettant le port GPIO qui lui est relié à 0. La mesure pourrait en effet être faussée si le condensateur avait une charge résiduelle provenant d'une mesure précédente. Une boucle `while` est ensuite initialisée et tourne tant que le port GPIO ne passe pas de 0 à 1, c'est-à-dire tant que le temps de charge n'a pas atteint la valeur du produit RC .

Il est même possible, avec ce principe, de réaliser quelques mesures grossières. Il suffit pour cela d'étalonner le montage en remplaçant le capteur résistif par des résistances fixes de valeurs connues et de dresser un tableau des valeurs de temps que cela produit. Ces valeurs pourront ensuite être placées dans une liste ou dans un dictionnaire (voir chapitre 5 si nécessaire) pour pouvoir être exploitées dans un programme.

9 • BUS I2C, BUS SPI, CARTES PIFACE ET GERTBOARD

Même si l'utilisation des ports parallèles permet comme nous l'avons vu d'interfacer le Raspberry Pi avec de nombreux éléments externes, elle est loin de résoudre tous les problèmes que l'on peut rencontrer au sein d'une application. En effet, de nombreux circuits, capteurs ou équipements informatiques utilisent pour communiquer ce que l'on peut regrouper sous le vocable commun de **liaisons séries** ou de bus séries selon le cas. Fort heureusement, ces bus ou liaisons sont aujourd'hui parfaitement normalisés pour ce qui est des plus importants d'entre eux et il est donc relativement facile d'y connecter un Raspberry Pi.

Ce chapitre va donc s'intéresser à ces ressources et nous allons y étudier comment communiquer avec « le monde extérieur » via un bus I2C tout d'abord puis une interface SPI ensuite, ce qui nous permettra de vous présenter la carte PiFace, propre à réconcilier tous ceux d'entre vous qui souhaitent interfacer leur Raspberry Pi au monde extérieur sans sortir le fer à souder.

Enfin, nous évoquerons la carte Gertboard, qui est aussi une alternative intéressante pour ceux d'entre vous qui veulent interfacer le Raspberry Pi avec le monde extérieur sans fer à souder, avec des possibilités plus nombreuses que celles offertes par la carte PiFace, mais au prix d'un encombrement et d'un coût plus importants.

9.1 Le bus I2C

Le succès remporté par le bus I2C, très utilisé depuis déjà de nombreuses années dans les appareils audio et vidéo, a comme conséquence que de nombreux boîtiers sont disponibles avec une telle interface. On trouve ainsi des mémoires bien sûr mais aussi des convertisseurs analogiques/digitaux et digitaux/analogiques, des horloges temps réel, des interfaces parallèles ainsi que toute une gamme de circuits aux fonctions beaucoup plus complexes puisqu'il existe même des synthétiseurs de fréquence intégrés pour récepteurs radio ou TV.

Même s'il est possible de l'utiliser sans savoir vraiment comment il fonctionne, comme le font allègrement de nombreux sites Internet, la connaissance des principes de base de son protocole s'avère utile à sa bonne mise en œuvre.

9.1.1 Généralités

Le bus I2C, qui n'utilise que deux lignes de signal (et les masses correspondantes bien sûr), permet d'échanger des informations sous forme série avec un débit pouvant atteindre 100 kilobits par seconde ou 400 kilobits par seconde pour les versions les plus récentes. Ses points forts sont les suivants :

- c'est un bus série bifilaire utilisant une ligne de données appelée SDA (Serial DAta) et une ligne d'horloge appelée SCL (Serial CLock).
- Les données peuvent être échangées dans les deux sens sans restriction.
- Le bus est multi-maître.
- Chaque abonné dispose d'une adresse codée sur 7 bits. On peut donc connecter simultanément 128 abonnés d'adresses différentes sur le même bus (sous réserve de ne pas le surcharger électriquement bien sûr).
- Un acquittement est généré pour chaque octet de donnée transféré.
- Le bus peut travailler à une vitesse maximum de 100 kilobits par seconde ou 400 kilobits par seconde pour les versions les plus récentes, étant entendu qu'un procédé automatique permet de ralentir l'équipement le plus rapide pour s'adapter à la vitesse de l'élément le plus lent lors d'un transfert.
- Le nombre maximum d'abonnés n'est limité que par la charge capacitive maximale du bus qui peut être de 400 pF. Ce nombre ne dépend donc que de la technologie des circuits et du mode câblage employés.
- Les niveaux électriques permettent l'utilisation de circuits en technologies CMOS, NMOS ou TTL.

9.1.2 Protocole du bus I2C

La figure 9.1 montre le principe adopté au niveau des étages d'entrée/sortie des circuits d'interface au bus I2C. Si la partie entrée n'appelle aucune remarque particulière, on constate que la partie sortie fait appel à une configuration à drain ouvert (l'équivalent en MOS du classique collecteur ouvert) ce qui permet de réaliser des ET câblés par simple connexion, sur la ligne SDA ou SCL, des sorties de tous les circuits.

Aucune charge n'étant prévue dans ces derniers, une résistance de rappel à une tension positive doit être mise en place. Le niveau électrique n'est pas précisé pour l'instant car il dépend de cette tension. Nous parlerons donc de niveaux logiques hauts ou « 1 » ou bien encore de niveaux logiques bas ou « 0 » étant entendu que l'on travaille en logique positive c'est-à-dire qu'un niveau haut correspond à une tension plus élevée qu'un niveau bas.

Compte tenu de ce mode de connexion en ET câblé, lorsqu'aucun abonné n'émet sur le bus, les lignes SDA et SCL sont au niveau haut qui est leur état de repos.

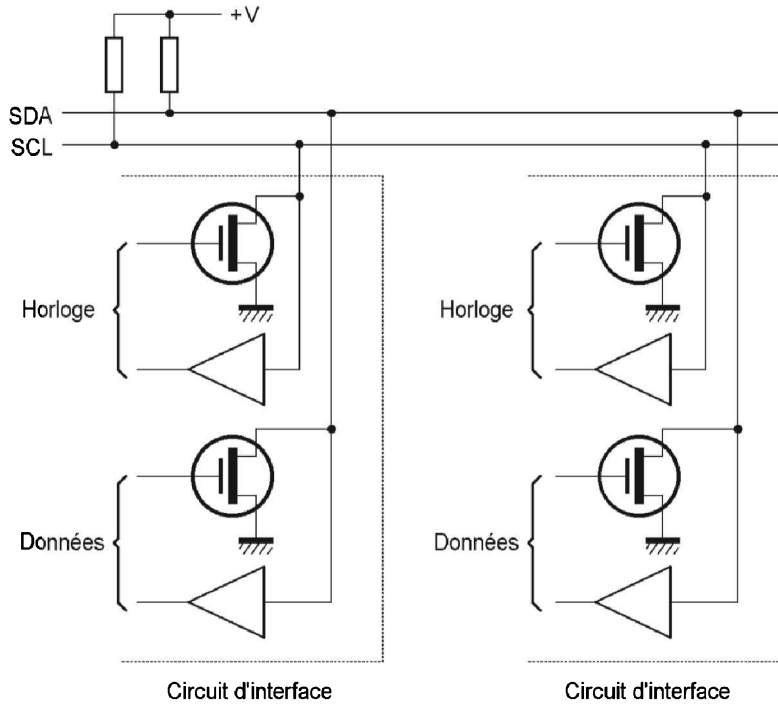


Figure 9.1 – Schéma de l'interface matérielle des circuits compatibles du bus I2C.

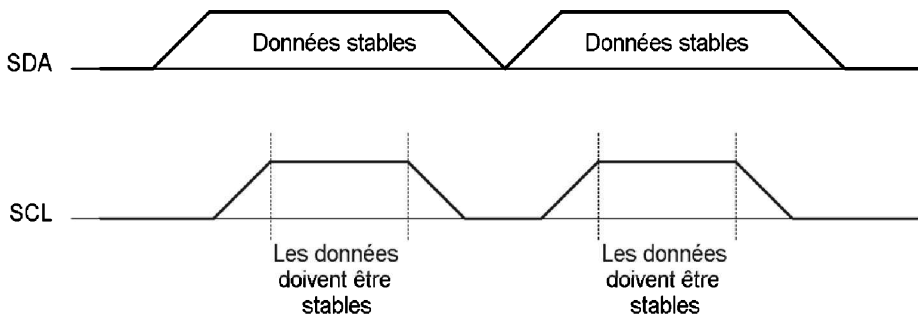


Figure 9.2 – Chronogramme fondamental du bus I2C.

Ceci étant vu, la figure 9.2 résume le principe fondamental d'un transfert, à savoir : une donnée n'est considérée comme valide sur le bus que lorsque le signal SCL est à l'état haut. L'émetteur doit donc positionner la donnée à émettre lorsque SCL est à l'état bas et la maintenir tant que SCL reste à l'état haut.

Comme la transmission s'effectue sous forme série, une information de début et de fin doit être prévue. L'information de début s'appelle ici condition de départ et l'information de fin s'appelle condition d'arrêt.

Une condition de départ est réalisée lorsque la ligne SDA passe du niveau haut au niveau bas alors que SCL est au niveau haut. Réciproquement, une condition d'arrêt est réalisée lorsque SDA passe du niveau bas au niveau haut alors que SCL est au niveau haut.

En outre, bien que nous soyons en présence d'un bus série, les données sont envoyées par paquets de huit, ou octets, même si un octet peut parfois regrouper huit bits indépendants. Le bit de poids fort est envoyé le premier. Chaque octet est suivi par un bit d'acquiescement de la part du destinataire. L'ensemble du processus fonctionnant de la façon présentée figure 9.3.

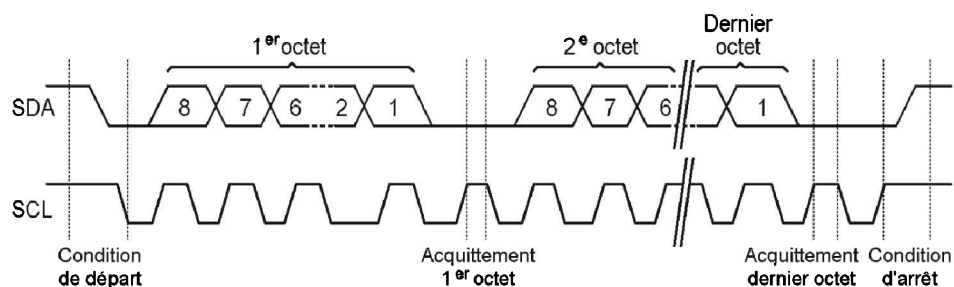


Figure 9.3 – Chronogramme d'un échange complet sur le bus I2C.

Tout d'abord, précisons que, lors d'un échange de ce type, la ligne SCL est pilotée par l'initiateur de l'échange ou maître, quitte à ce que l'esclave agisse également dessus dans certains cas particuliers.

La figure 9.3 nous montre tout d'abord une condition de départ, générée par le maître du bus à cet instant. Elle est suivie par le premier octet de données, poids forts en tête. Après le huitième bit, l'émetteur qui est aussi le maître dans ce cas, met sa ligne SDA au niveau haut c'est-à-dire au repos mais continue à générer l'horloge sur SCL. Pour acquiescer l'octet, le récepteur doit alors forcer la ligne SDA au niveau bas pendant l'état haut de SCL qui correspond à cet acquiescement, prenant en quelque sorte la place d'un neuvième bit.

Le processus peut alors continuer avec l'octet suivant et se répéter autant de fois que nécessaire pour réaliser un échange d'informations complet. Lorsque cet échange est terminé, le maître génère une condition d'arrêt.

Maintenant que nous savons comment se déroulent les échanges ; il nous reste à examiner le format des données transmises afin de comprendre comment fonctionne l'adressage, mais aussi la définition du sens de transferts des données.

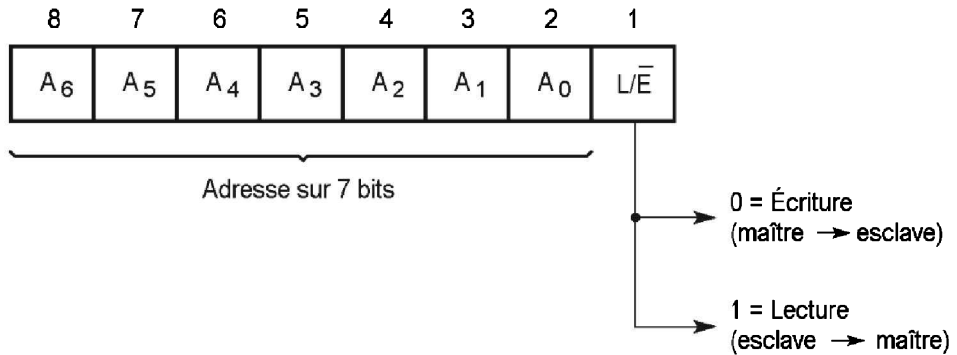


Figure 9.4 – Contenu de l’octet d’en-tête d’un échange sur le bus I2C.

La figure 9.4 montre le contenu du premier octet qui est toujours présent en début d’échange. Ses sept bits de poids forts contiennent l’adresse du destinataire du message, ce qui autorise 128 combinaisons différentes. Le bit de poids faible quant à lui indique si le maître va réaliser une lecture ou une écriture. En d’autres termes, si ce bit est à zéro le maître va écrire dans l’esclave ou lui envoyer des données. Si ce bit est à un, le maître va lire dans l’esclave c’est-à-dire que le maître va recevoir des données de l’esclave.

Lorsqu’un maître désire effectuer plusieurs échanges à destination d’esclaves d’adresses différentes, il n’est pas obligé de terminer le premier échange par une condition d’arrêt mais peut les enchaîner en générant une condition de départ dès la fin d’un échange.

Enfin il existe une procédure dite d’appel général où l’adresse envoyée par le maître, c’est-à-dire rappelons-le, les sept bits de poids forts du premier octet, est nulle. Tous les circuits connectés sur le bus capables de répondre à un tel appel général doivent alors le faire et prendre en compte les données qui suivent. Leur attitude dépend du bit de lecture/écriture de ce premier octet. En effet, si ce bit est à zéro, le deuxième octet revêt une signification particulière, que nous ne détaillerons pas ici, mais qui permet, au niveau de chaque esclave, la programmation de son adresse par logiciel ou matériel.

Lorsque ce bit est à un, on est en présence d’un appel général matériel qui est transmis par un maître du bus matériel, tel qu’un circuit d’interface de clavier par exemple. Un tel circuit est incapable de générer l’adresse du destinataire de l’information qu’il a à envoyer. Dans ces conditions, l’octet suivant contient l’adresse de ce maître matériel afin qu’un esclave « intelligent », généralement un microcontrôleur, le reconnaisse et prenne en compte l’information qu’il émet ensuite.

Il y aurait encore beaucoup à dire à propos du bus I2C mais ce n’est pas le but de cet ouvrage aussi avons-nous restreint l’exposé précédent au strict minimum nécessaire afin que vous puissiez interfacer au mieux des circuits I2C avec le Raspberry Pi.

9.1.3. Exemple d'utilisation du bus I2C

Si vous n'avez pas l'habitude d'utiliser le bus I2C, tout ceci vous laisse peut-être un peu perplexe aussi allons-nous voir maintenant un exemple particulièrement utile de surcroît puisqu'il va nous permettre de doter notre Raspberry Pi d'un convertisseur analogique/digital qui lui fait cruellement défaut, comme nous avons eu l'occasion de le remarquer au chapitre précédent.

Avant de voir le schéma et les programmes correspondants, précisons toutefois comment l'on doit travailler lorsque l'on veut interfacer un circuit I2C avec le Raspberry Pi. La méthodologie est en effet toujours la même.

La première opération, qui est bien souvent la plus longue et la plus délicate, consiste à lire en détail la fiche technique du circuit afin de prendre connaissance de sa ou de ses adresses I2C, des registres qu'il contient et de ce qu'il faut y écrire pour réaliser les fonctions désirées. Une fois que c'est fait, et éventuellement résumé noir sur blanc sur une feuille de papier, il ne reste plus qu'à enchaîner les fonctions nécessaires pour envoyer et/ou recevoir les données correspondantes.

Ceci étant précisé, nous allons utiliser un convertisseur analogique/digital de type ADS1015 de Texas Instruments, dont vous trouverez la fiche technique dans les compléments en ligne de cet ouvrage. Il s'agit d'un convertisseur 12 bits disposant de quatre entrées qui peuvent être utilisées sous forme « normale » ou unipolaire (*single ended* en bon anglais) ou sous forme de deux entrées différentielles. Il s'alimente sous une tension unique de 3,3 volts, parfaitement adaptée au Raspberry Pi donc, et s'interface bien évidemment avec un bus I2C.

Son schéma de mise en œuvre est d'une extrême simplicité comme vous pouvez le constater figure 9.5, et encore la résistance R3 n'est-elle pas utile dans le cadre de l'usage que nous allons faire du circuit.

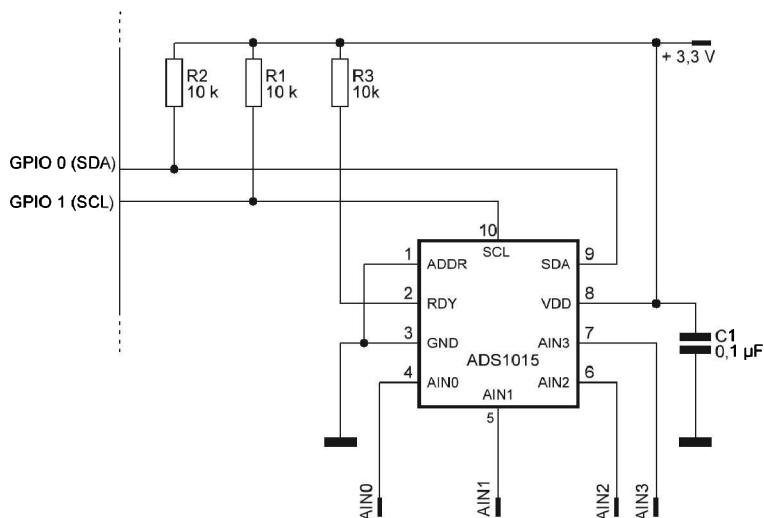


Figure 9.5 – Schéma de mise en œuvre du convertisseur analogique/digital ADS1015.

Le seul « défaut » de ce convertisseur est qu'il n'est disponible qu'en boîtier CMS (Composant à Montage en Surface) minuscule et qu'il n'est donc pas envisageable que vous le soudiez vous-même sur quoi que ce soit. Fort heureusement, la société Adafruit (<http://www.adafruit.com>) qui nous a déjà livré le Pi Cobbler et le Pi Proto Plate du chapitre 6, propose un « *breakout board* » équipé de ce circuit, dont l'aspect est visible figure 9.6.

Ce circuit est en outre disponible en France chez Snootlab (<http://www.snootlab.com>) à un prix du même ordre de grandeur que ce que vous paieriez au détail pour les composants qui l'équipent. Alors pourquoi s'en priver ?

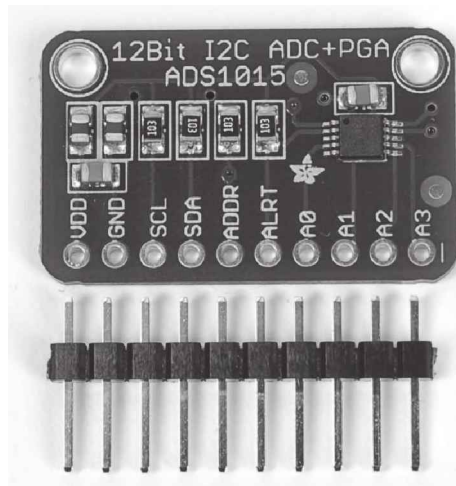


Figure 9.6 – Le circuit imprimé support de l'ADS1015 proposé par Adafruit (doc. Adafruit).

Une fois que le connecteur mâle – mâle fourni avec le kit est soudé sur le circuit imprimé, ce dernier peut s'enficher sans problème sur une carte de câblage rapide, telles celles que nous avons présentées au chapitre 6, et vous permettre ainsi toutes les expérimentations le concernant.

Une fois en possession de ce circuit, ou de tout autre circuit à interface I2C bien sûr, il faut se livrer à un petit travail de préparation avant de le relier au Raspberry Pi, faute de quoi l'interface I2C disponible sur le port GPIO restera désespérément muette. À partir de l'interface ligne de commande ou depuis *LXTerminal*, éditez le fichier `modules` au moyen de `nano` par exemple grâce à la commande suivante :

```
pi@raspberrypi ~ $ sudo nano /etc/modules
```

Ajoutez, à la fin de ce fichier, les deux lignes suivantes :

```
i2c-bcm2708
i2c-dev
```

Sauvegardez le fichier par un **CNTRL O** suivi de **Entrée** et quittez nano par un **CNTRL X**. Installez ensuite les outils nécessaires à la manipulation du bus I2C au moyen des commandes désormais bien connues :

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get install python-smbus
pi@raspberrypi ~ $ sudo apt-get install i2c-tools
```

Vérifiez si, dans votre distribution, vous disposez du fichier `raspi-blacklist.conf` dans le répertoire `/etc/modprobe.d/`. Si tel est le cas, éditez ce fichier avec nano au moyen de la ligne suivante :

```
pi@raspberrypi ~ $ sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Ajoutez ensuite un dièse (#) devant chacune des deux lignes suivantes :

```
spi-bcm2708
i2c-bcm2708
```

À ce stade des opérations, le dièse devant la ligne commençant par `spi` n'est pas utile mais, comme nous supposons que vous essaieriez tôt ou tard le bus SPI du Raspberry Pi, autant le débloquent dès à présent.

Sauvegardez le fichier par un **CNTRL O** suivi de **Entrée** et quittez nano par un **CNTRL X**. Redémarrez alors votre Raspberry Pi pour que ces fichiers de configuration soient pris en compte au moyen d'un désormais classique `sudo reboot`.

Une fois toutes ces opérations réalisées, vous pouvez relier le circuit imprimé supportant le convertisseur à votre Raspberry Pi et vérifier qu'il est bien vu comme étant connecté au bus I2C au moyen de la commande suivante :

```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
```

Celle-ci doit conduire à l'affichage sur votre écran d'une grille similaire à ce qui est reproduit figure 9.7. Cette grille représente en fait toute l'étendue d'adressage permise par le bus I2C avec, au seul endroit où ne figure pas un tiret, l'adresse de votre module. Si cette détection est positive, et elle pourrait ne pas l'être qu'en cas de grosse erreur de câblage de votre part, vous pouvez passer à l'écriture du programme.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00:				--	--	--	--	--	--	--	--	--	--	--	--	--
10:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40:	--	--	--	--	--	--	--	--	48	--	--	--	--	--	--	--
50:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Figure 9.7 – Le convertisseur analogique/digital a bien été détecté sur le bus I2C.

Pour ce faire, il faut utiliser la bibliothèque `smbus` qui, bien que théoriquement destinée au bus SMB, fonctionne aussi avec l'I2C puisque ces deux bus ne diffèrent en fait que par leur appellation.

Cette bibliothèque comporte toutes les fonctions utiles pour lire et écrire sur le bus I2C telles que, par exemple : `read_byte(adresse)` qui lit un octet dans le circuit I2C d'adresse spécifiée ou bien encore `write_byte(adresse, donnée)` qui écrit une donnée dans le circuit I2C d'adresse spécifiée. Cette bibliothèque est hélas assez mal documentée et, si vous voulez découvrir l'essentiel de ses fonctions et de sa syntaxe, nous vous proposons de consulter le site d'adresse <http://wiki.erazor-zone.de/wiki:linux:python:smbus:doc#usage> qui en fait une synthèse utilisable.

Pour ce qui est de notre convertisseur analogique/digital, et afin de ne pas accaparer trop de pages uniquement consacrées à son sujet dans le cadre de cet ouvrage, nous vous proposons de télécharger les deux programmes mis à votre disposition par Adafruit pour piloter sa carte à partir de l'adresse : https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code/tree/master/Adafruit_ADS1x15.

`Adafruit_ADS1x15.py` est le programme qui gère les échanges sur le bus I2C avec l'ADS1015 tandis que `ads1015_example.py` est un exemple d'utilisation de ce programme.

Nous vous laissons le soin de découvrir leur contenu, non sans avoir au préalable pris connaissance de la fiche technique de l'ADS1015 et des registres internes qu'il convient d'initialiser, et dans lesquels on vient ensuite lire le résultat de la conversion.

L'utilisation du fichier `ads1015_example.py` doit se faire en mode super-utilisateur sous la forme :

```
pi@raspberrypi ~ $ sudo python ads1015_example.py
```

ce qui a pour effet de faire afficher les tensions mesurées sur les entrées `AIN0` et `AIN1` du convertisseur.

Attention !

La tension d'entrée du convertisseur doit toujours être positive ou nulle et inférieure à sa tension d'alimentation, soit 3,3 volts, sous peine de détruire ce dernier.

Et si le programme exemple d'Adafruit ne vous plaît pas, n'hésitez pas à créer le vôtre, ce qui est très simple sous réserve que vous y importiez `Adafruit_ADS1x15` comme dans l'exemple que voici :

```
from Adafruit_ADS1x15 import ADS1x15
import time

adc = ADS1x15()

while True:
    tension_brute = adc.readADCSingleEnded(0)
    print tension_brute
    sleep(.2)
```

Ce programme vient tout simplement lire la tension présente sur l'entrée **AIN0** du convertisseur, utilisée en mode normal (non différentielle donc), et affiche le résultat obtenu sous forme brute, c'est-à-dire directement le résultat fourni par le convertisseur.

9.2 L'interface ou bus SPI

Au royaume des circuits d'interface le bus I2C n'est pas seul et se trouve talonné depuis de nombreuses années par les circuits dotés d'une liaison série synchrone de type SPI. Cette appellation recouvre cependant des signaux et des chronogrammes qui sont beaucoup moins bien normalisés que ceux du bus I2C, tout simplement parce que la liaison SPI ne fait l'objet d'aucune norme officielle.

Certains circuits indiquent donc qu'ils sont munis d'une interface SPI, mais vous pourrez aussi trouver comme appellation Microwire (qui est une marque déposée de National Semiconductor) ou bien encore bus série trois fils. En effet, un bus de ce type permet la connexion, sous forme série maître - esclave, de plusieurs circuits disposant d'interfaces compatibles, avec seulement trois fils de liaison actifs, complétés si nécessaire par un fil de validation.

9.2.1 Principe d'une liaison de type SPI

La figure 9.8 présente le principe général d'interconnexion retenu. Ce schéma ne doit cependant pas être considéré comme unique. On peut en effet relier le SPI à de très nombreux autres circuits : registres à décalage, afficheurs disposant de leur propre logique, etc. Il suffit de s'assurer qu'ils sont à même de dialoguer avec l'interface, ce qui est une contrainte relativement simple à satisfaire vu la souplesse de programmation permise.

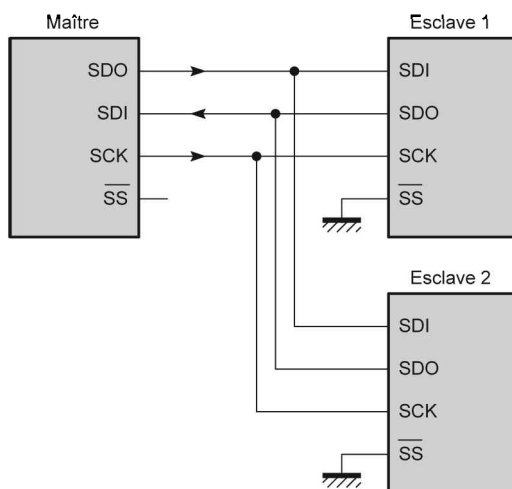


Figure 9.8 – Principe général d'une liaison de type SPI.

Si l'on reprend la figure 9.8 comme exemple, on y voit trois lignes « utiles » que sont SDO, SDI et SCK et une ligne de sélection de mode de fonctionnement du circuit concerné appelée /SS.

- La ligne SDO, pour *Serial Data Out*, est la sortie de données séries. Elle est aussi appelée MOSI ce qui signifie *Master Out Slave In*.
- La ligne SDI, pour *Serial Data In*, est comme son nom le laisse supposer, l'entrée de données série. Elle est aussi appelée MISO ce qui signifie *Master In Slave Out*.
- La ligne SCK pour *Serial ClocK* est utilisée pour synchroniser les échanges de données puisque nous sommes ici en mode série synchrone ne l'oublions pas. Cette ligne est nécessairement une sortie sur le maître et une entrée sur le ou les esclaves. La figure 9.9 montre les relations de phase qui existent entre les données et cette horloge étant entendu que celles-ci peuvent varier d'un circuit à un autre mais sont généralement programmables au niveau de l'élément « intelligent » qui gère le bus.
- La ligne /SS est la sélection du mode maître ou esclave. Elle est au niveau haut sur un maître et au niveau bas sur un esclave. Pour éviter toute erreur de transmission, cette ligne doit être au niveau bas sur le ou les esclaves avant toute transmission et y rester pendant toute la durée de celle-ci. En outre, il est évident qu'à un instant donné, n'importe quel circuit peut être maître et n'importe quel autre esclave, mais qu'il ne doit y avoir qu'un maître par transaction.

Toutes les liaisons SPI respectent le chronogramme de principe présenté figure 9.9 mais quatre variantes existent au niveau de la prise en compte des données qui sont échangées.

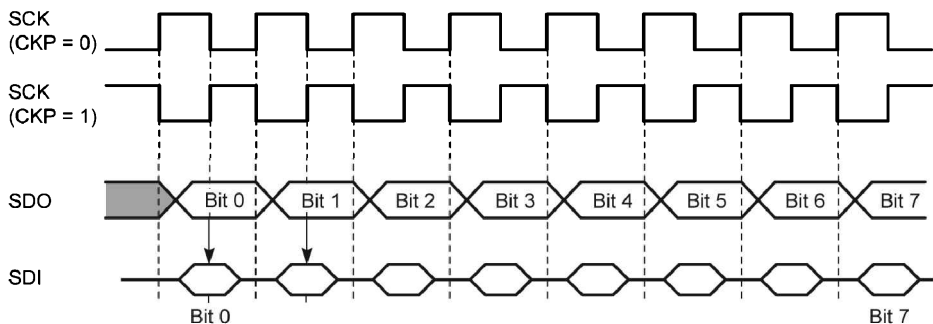


Figure 9.9 – Chronogramme général d'un échange sur une liaison SPI.

Ces variantes sont parfois appelées modes SPI et sont alors numérotées de 0 à 3. Nous préférons quant à nous dessiner un petit chronogramme permettant de bien préciser ce qu'il en est. En présence d'un circuit à interface SPI, il vous suffira donc de comparer sa fiche technique avec ce chronogramme pour savoir dans quel « mode » il fonctionne.

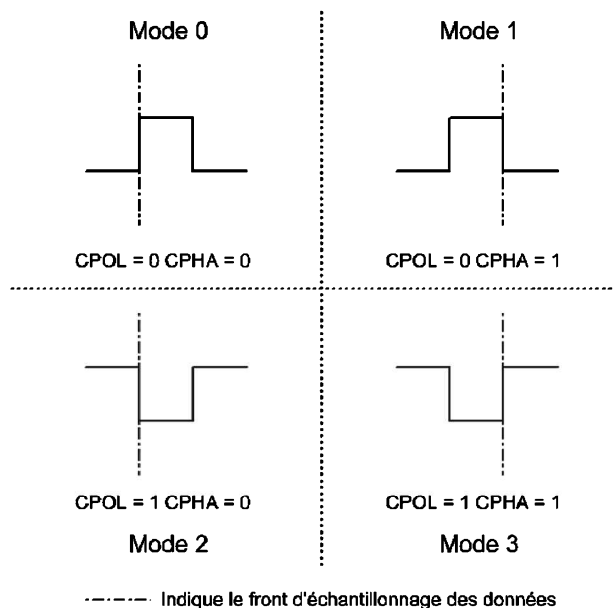


Figure 9.10 – Les quatre modes de prise en compte des données de la liaison SPI.

En fait, ces quatre modes sont définis par la polarité de l'horloge, appelée CPOL, qui peut être égale à 0 ou à 1, et par la phase de cette même horloge, appelée CPHA, par rapport aux données, qui peut aussi être égale à 0 ou à 1. Cela conduit aux quatre possibilités visibles figure 9.10 qui indiquent en fait sur quel front du signal d'horloge les données sont échantillonnées et peuvent donc être considérées comme valides. Par défaut, le mode le plus courant est le mode appelé mode 0 c'est-à-dire celui où CPOL et CPHA sont tous deux nuls.

9.2.2 La carte à interface SPI PiFace

Arrivés à ce stade de l'ouvrage, nous n'allons pas vous proposer d'interfacer un circuit choisi à titre d'exemple au bus SPI du Raspberry Pi, mais plutôt de joindre l'utile à l'agréable en vous présentant un module qui présente tout à la fois l'avantage de faire appel au bus SPI pour son interface avec le Raspberry Pi, mais également de mettre à votre disposition des LED, des poussoirs et des relais que vous pouvez commander tout à loisir.

Qui plus est, ce module est vendu prêt à l'emploi ce qui vous permettra, si vous n'avez pas eu le courage de sortir le fer à souder et les outils qui vont avec au cours des chapitres précédents, de pouvoir tout de même relier votre Raspberry Pi au monde extérieur sans faire la moindre soudure.

Baptisée PiFace, cette petite carte vous est présentée figure 9.11 et s'enfiche directement sur le connecteur GPIO du Raspberry Pi avec lequel elle forme ensuite un bloc relativement compact. Elle met à votre disposition les éléments suivants :

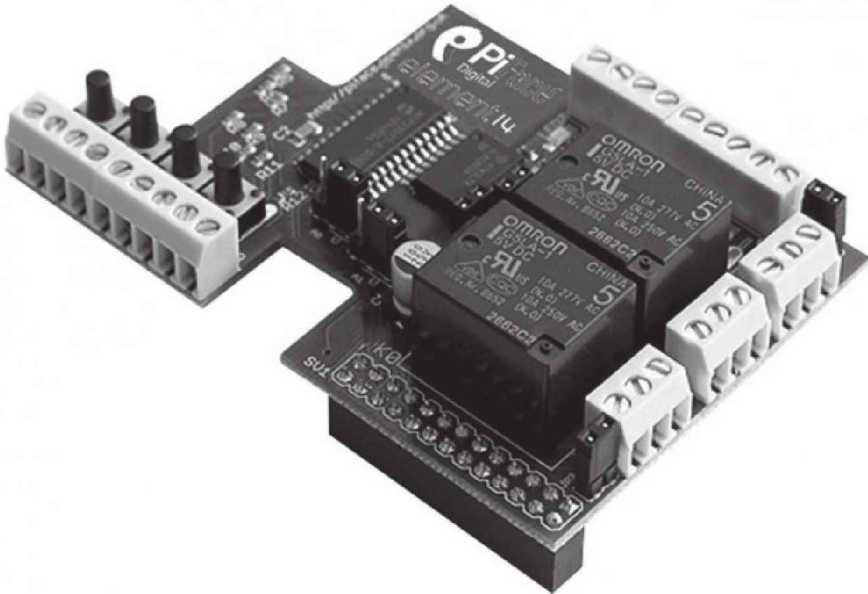


Figure 9.11 – La carte PiFace.

- Deux relais, capables de commander des charges reliées au secteur ; charges qui peuvent être puissantes puisqu'ils peuvent commuter chacun jusqu'à 10 ampères.
- Quatre poussoirs.
- Huit entrées numériques.
- Huit sorties numériques à collecteurs ouverts, partagées cependant avec la commande des relais si ceux-ci sont utilisés, ce qui n'est pas obligatoire.
- Huit LED indicatrices de l'état de ces sorties.

En outre, tous les éléments présents sur la carte, et donc toutes les entrées/sorties, sont accessibles au moyen de borniers à vis que l'on distingue très bien tout autour de la carte sur la figure 9.11, tandis que des straps amovibles permettent certaines modifications de sa configuration initiale.

Au moment où ces lignes sont écrites, cette carte n'est disponible en France que chez Farnell (<http://fr.farnell.com>) pour un peu moins de 30 euros ce qui, là encore, reste très raisonnable compte tenu des composants qui l'équipent.

Malgré cette profusion d'entrées/sorties qui, si elles étaient réalisées sous forme parallèle comme nous l'avons vu au chapitre 7, consommeraient quasiment toutes les lignes du port GPIO du Raspberry Pi, ce n'est pas le cas ici puisque la carte utilise un circuit intégré MCP23S17. Ce circuit, dont vous trouverez la fiche technique dans les compléments en ligne de cet ouvrage, est en fait une interface parallèle seize bits sur bus SPI. PiFace ne consomme donc que quatre lignes sur le port GPIO du Raspberry Pi : MISO, MOSI, SCLK et la ligne de sélection CE0.

9.2.3 Les logiciels de la carte PiFace

L'utilisation de cette carte est possible sous deux formes différentes. Avec un émulateur graphique tout d'abord, qui permet de visualiser la carte à l'écran et d'agir directement sur ses entrées/sorties sous forme très visuelle, et une bibliothèque spécifique, à importer dans vos propres programmes. Cette bibliothèque vous décharge de toute tâche de gestion du bus SPI qui devient ainsi totalement transparent depuis votre programme.

Ces deux logiciels, émulateur et bibliothèque, sont installés en une seule fois au moyen d'un paquet prévu à cet effet mais, avant de procéder à cette installation, il faut valider l'interface SPI du Raspberry Pi en éditant le fichier `raspi-blacklist.conf` comme nous l'avons fait au 9.1.3 ci-dessus pour l'I2C en ajoutant un dièse devant `spi-bcm2708`.

Lorsque c'est fait, à partir de la ligne de commande ou depuis **LXTerminal**, exécutez les commandes suivantes :

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ wget
http://pi.cs.man.ac.uk/download/install.txt
pi@raspberrypi ~ $ bash install.txt
```

Cela va avoir pour effet de télécharger tous les logiciels nécessaires à la carte PiFace et de les installer ensuite correctement dans les répertoires appropriés.

■ Attention !

Soyez patient car cette opération peut prendre jusqu'à une bonne demi-heure.

Une fois que c'est terminé, faites redémarrer votre Raspberry Pi par un désormais bien connu `sudo reboot`.

Même si le fonctionnement avec l'émulateur n'est pas le plus intéressant puisqu'il ne permet aucune intégration dans vos programmes, nous vous conseillons de l'essayer au moins une fois car, outre son aspect graphique très plaisant, il permet de tester la carte PiFace et, si vous êtes un peu curieux, de suivre les échanges qui ont lieu sur le bus SPI.

Pour lancer cet émulateur, placez-vous sous l'environnement graphique LXDE et ouvrez une fenêtre **LXTerminal** dans laquelle vous saisissez la ligne suivante :

```
pi@raspberrypi ~ $ piface/scripts/piface-emulator
```

Vous devriez alors observer un écran semblable à celui visible figure 9.12 sur lequel il vous faudra cliquer sur **Override Enable** afin de rendre cet émulateur prioritaire sur un éventuel autre programme qui piloterait la carte PiFace.

Vous pouvez alors cliquer sur le bouton de votre choix pour valider telle ou telle sortie et constater que c'est confirmé sous forme visuelle sur le dessin de la carte mais aussi, évidemment que cela se traduit dans la réalité sur la carte elle-même.

Si vous voulez vérifier que l'appui sur les poussoirs est bien pris en compte, cochez la case **Keep inputs updated** et, si vous êtes un tant soit peu curieux, cochez aussi **SPI visualiser**. La fenêtre prend alors l'aspect visible figure 9.13 et l'on peut prendre connaissance, dans sa partie basse, des trames échangées sur le bus SPI.

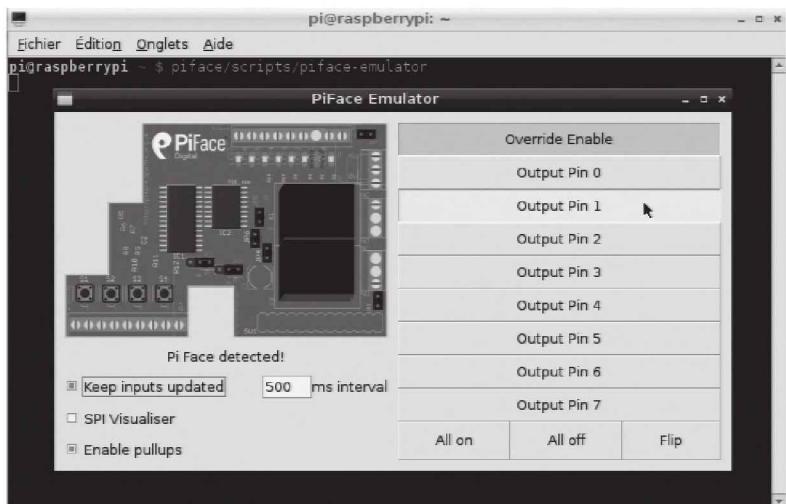


Figure 9.12 – L'émulateur permet de gérer sous forme graphique toutes les entrées/sorties de la carte PiFace.

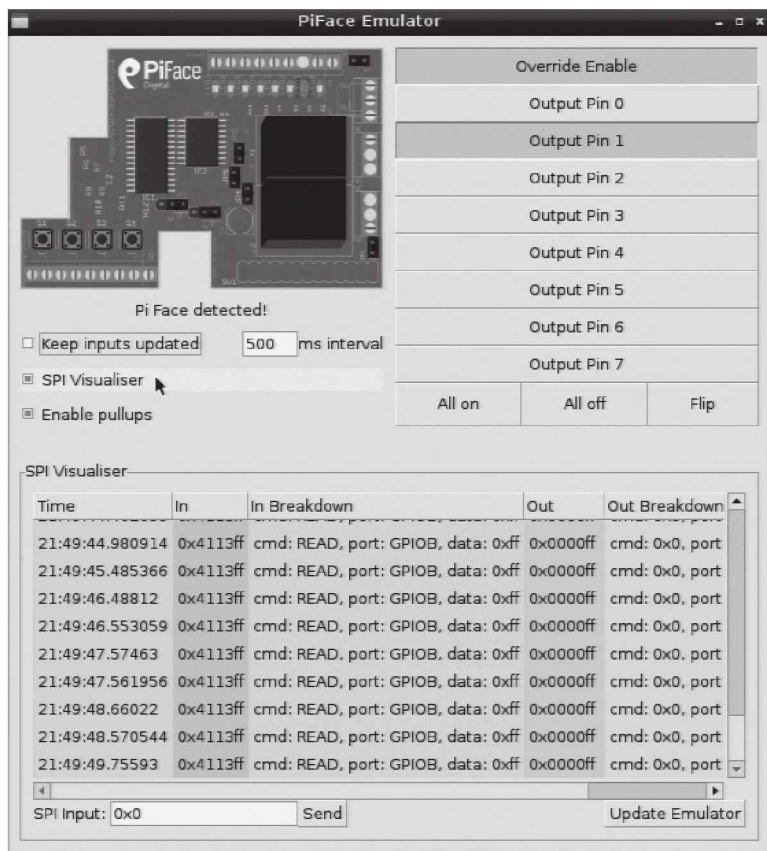


Figure 9.13 – L'émulateur peut afficher les trames SPI échangées avec la carte PiFace.

Aussi agréable que puisse être cet émulateur, ce n'est pas avec lui que vous pourrez réellement profiter de la carte PiFace mais en réalisant vos propres programmes. Pour cela, il faut faire appel à la bibliothèque `piface.pfio` qui a été installée automatiquement si vous avez réalisé les manipulations précédentes. Cette bibliothèque dispose des quatre fonctions que voici.

■ `init()`

Cette fonction doit être appelée une fois avant toute utilisation de la carte. Elle n'utilise aucun paramètre et ne sert qu'à initialiser PiFace.

■ `digital_read(numéro d'entrée)`

Cette fonction lit l'état de l'entrée dont le numéro est spécifié en paramètre. Ce numéro peut évoluer de 0 à 7 étant entendu que les poussoirs S1, S2, S3 et S4 sont câblés sur les entrées 0 à 3. Elle fournit comme résultat un 1 si l'entrée est au niveau logique haut et un 0 si l'entrée est reliée à la masse.

■ `digital_write(numéro de sortie, état)`

Cette fonction écrit, sur la sortie dont le numéro est spécifié, l'état désiré qui peut être un 1 pour la mettre à l'état haut ou un 0 pour la mettre à l'état bas. Les sorties sont également numérotées de 0 à 7 étant entendu que les deux relais sont reliés par défaut aux sorties 0 et 1.

■ `digital_write_pullup(numéro d'entrée, état)`

Afin de faciliter le câblage des interrupteurs, poussoirs ou capteurs de type contact, reliés aux entrées de la carte, il est possible d'activer, entrée par entrée, une résistance de « *pull up* » ou de tirage au niveau haut si vous préférez, de 10 kOhms (revoyez si nécessaire à ce propos la figure 7.7). `numéro d'entrée` spécifie bien évidemment l'entrée concernée tandis que `état` est mis à 0 pour désactiver la résistance tandis qu'un 1 l'active.

Une fois ces fonctions connues, l'écriture d'un programme exploitant la carte PiFace est un jeu d'enfant. Il est même possible de le tester directement dans IDLE en mode direct afin d'en vérifier le comportement. Après avoir lancé IDLE, vous écrirez par exemple :

```
>>> import piface.pfio as pfio
>>> pfio.init()
>>> pfio.digital_write(0,1)
```

pour faire coller le relais numéro 0 et allumer la LED correspondante.

Pour lire une entrée, c'est tout aussi simple puisqu'il suffit par exemple d'écrire :

```
>>> import piface.pfio as pfio
>>> pfio.init()
>>> while True:
>>> if pfio.digital_read(0):
>>>     print 'Poussoir S1 appuyé'
```

Pour voir afficher le message aussi longtemps que le poussoir S1 sera appuyé.

Attention !

Au moment où ces lignes sont écrites, la bibliothèque `piface.pfio` n'est disponible que pour Python 2. Il n'est donc pas possible de l'utiliser avec Python 3 et, bien évidemment, il est impossible de faire fonctionner les exemples ci-dessus sous IDLE 3.

La carte PiFace n'est évidemment pas limitée aux seuls poussoirs et relais qui y sont installés d'origine et toutes ses entrées/sorties sont accessibles via les borniers qui l'entourent.

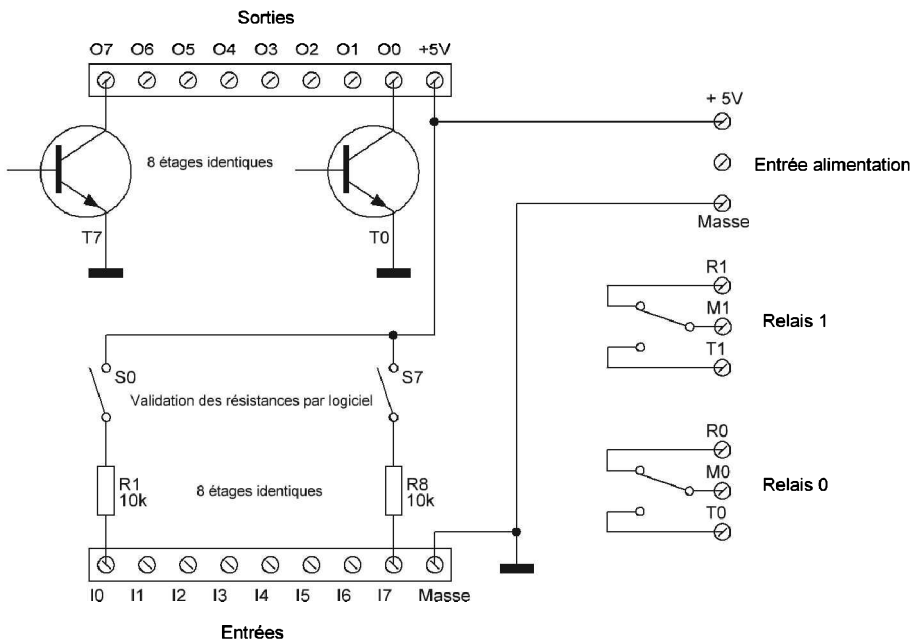


Figure 9.14 – Schémas équivalents des entrées/sorties de la carte PiFace.

Afin de pouvoir y raccorder vos propres éléments, sachez que la circuiterie interne de la carte respecte le schéma visible figure 9.14, que vous pourrez utilement comparer à ceux que nous vous avons présentés au chapitre 7, selon que vous souhaitez lire des interrupteurs, commander des relais, des LED, des moteurs ou tout autre élément s'accommodant d'entrées/sorties sous forme parallèle.

Notez qu'au niveau des sorties, les transistors qui sont représentés figure 9.14 sont en fait contenus dans un circuit intégré ULN 2803, similaire à celui que nous avons utilisé à de multiples reprises, et qu'ils peuvent donc commuter un courant de 500 mA sous une tension maximum absolue de 50 volts.

Attention !

Par défaut, les diodes de protection contenues dans cet ULN 2803 sont ramenées au + 5 volts qui alimente la carte PiFace via le strap JP4. Si vous désirez commander avec ce circuit des éléments alimentés sous une tension plus élevée que 5 volts, il faut désactiver ces diodes en enlevant ce strap.

Les autres cavaliers présents sur la carte PiFace permettent de modifier son adresse pour ce qui est de JP1 et JP2 mais, comme il est impossible de monter plusieurs PiFace sur un même Raspberry Pi, nous n'en voyons pas bien l'intérêt. Quant à JP5 et JP6, ils permettent de déconnecter les relais des sorties 0 et 1 si vous souhaitez utiliser ces dernières pour d'autres fonctions.

Enfin, JP3 permet de relier l'alimentation 5 volts de la carte PiFace à celle du Raspberry Pi et ainsi, soit d'alimenter PiFace depuis l'alimentation USB du Raspberry Pi, soit de faire le contraire, c'est-à-dire d'alimenter le Raspberry Pi à partir de l'alimentation 5 volts de la carte PiFace, qui doit être appliquée dans ce cas aux deux bornes prévues à cet effet à côté de JP7. Si JP3 n'est pas mis en place, les alimentations 5 volts du Raspberry Pi et de la carte PiFace sont indépendantes.

9.3 La carte Gertboard

Cette carte, conçue par Gert Van Loo qui est un des pères du Raspberry Pi, est une sorte de « super » PiFace mais avec des possibilités considérablement plus étendues. Il faut dire qu'elle dispose pour cela d'une surface conséquente puisqu'elle est presque deux fois plus grande que le Raspberry Pi lui-même, comme vous pouvez le constater figure 9.15.

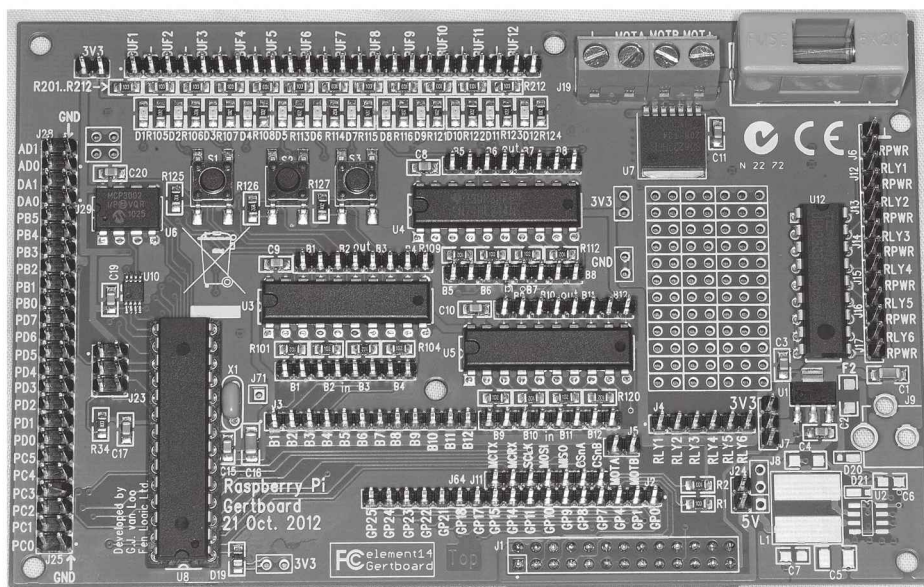


Figure 9.15 – La carte Gertboard.

Elle supporte les éléments suivants :

- Douze entrées/sorties amplifiées.
- Trois poussoirs.
- Six sorties à collecteur ouvert capable de commander 500 mA sous 50 volts.
- Un contrôleur de moteur à courant continu 18 volts sous 2 ampères maximum.
- Un convertisseur digital/analogique 8, 10 ou 12 bits à deux canaux.
- Un convertisseur analogique/digital 10 bits à deux canaux.
- Un microcontrôleur Atmel ATmega328.

Comme le montre le schéma synoptique de la figure 9.16, ces éléments ne sont pas interconnectés entre eux par défaut et de très nombreux picots mâles au pas de 2,54 mm permettent donc de les relier comme bon vous semble aux entrées/sorties du port GPIO du Raspberry Pi dans lequel bien entendu cette carte vient s'enficher. Des straps fournis avec la carte, constitués de fils souples munis de pinoches femelles, permettent de réaliser ces liaisons sans nécessiter aucune soudure.

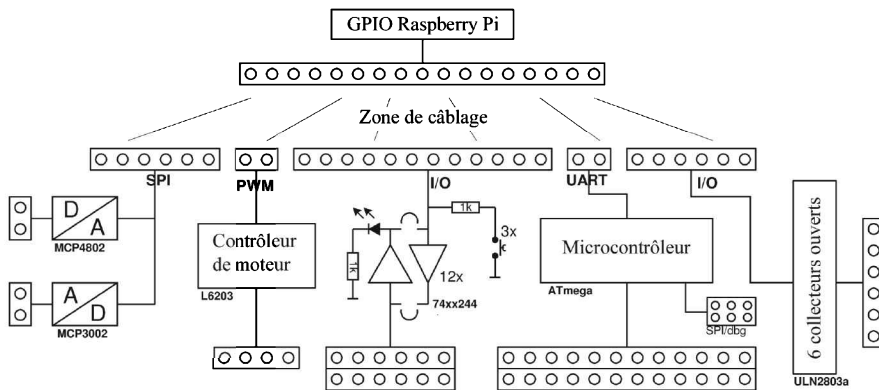


Figure 9.16 – Schéma synoptique de la carte Gertboard.

L'examen du schéma de cette carte, que vous trouverez dans les compléments en ligne de cet ouvrage, permet de constater que, pour ce qui est des entrées/sorties amplifiées, des poussoirs et du contrôleur de moteur, on retrouve à peu de choses près les schémas que nous vous avons présentés aux chapitres 7 et 8. Vous pouvez d'ailleurs, si vous le désirez et si son prix ne vous rebute pas, utiliser cette carte pour faire les manipulations que nous vous avons proposées dans ces mêmes chapitres. Les deux convertisseurs analogique/digital et digital/analogique quant à eux peuvent être reliés au bus SPI du port GPIO tandis que le microcontrôleur Atmel ATmega est livré vierge afin que vous puissiez le programmer en fonction de vos besoins. Il est prévu qu'il communique avec le Raspberry Pi via son UART au moyen donc d'une liaison série asynchrone. Bien sûr, cela nécessite de maîtriser la programmation des microcontrôleurs Atmel de la famille AVR mais, si ce n'est pas

votre cas, nous vous proposons de découvrir au chapitre 10 une autre approche particulièrement intéressante.

Cette carte est fournie avec une notice en anglais de près de cinquante pages et de nombreux exemples de programmes destinés à chacun des sous-ensembles qu'elle contient. Ces derniers ne sont toutefois que des versions à peine plus élaborées des exemples que nous vous avons présentés aux chapitres 7 et 8. Par contre, ils sont disponibles en Python mais aussi en C, ce qui peut être une alternative intéressante pour ceux d'entre vous qui maîtrisent déjà ce langage.

Malgré une conception irréprochable, l'intérêt de cette carte nous semble toutefois assez limité hors du domaine de la formation pour lequel, répétons-le, elle permet de faire quasiment toutes les expériences que nous avons présentées dans les chapitres précédents. Pour ce qui est de l'intégrer dans une application, notre avis est nettement moins positif car, outre le fait qu'elle coûte, au moment où ces lignes sont écrites, près d'une fois et demi le prix du Raspberry Pi, elle est aussi particulièrement encombrante, et ce d'autant qu'elle s'enfiche dans le connecteur du Raspberry Pi « vers l'extérieur » comme vous pouvez le constater figure 9.17. L'ensemble devient alors vraiment encombrant.

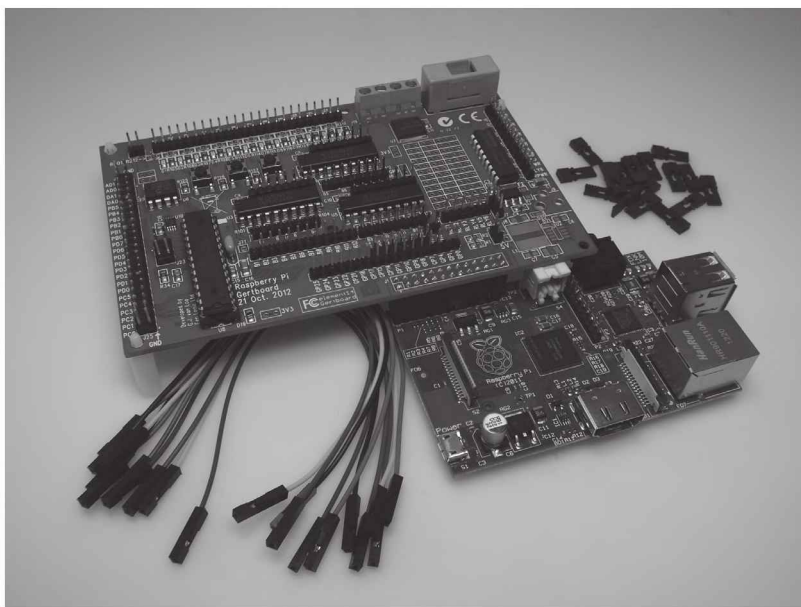


Figure 9.17 – La carte Gertboard s'enfiche sur le Raspberry Pi « vers l'extérieur » (doc. Gertboard).

Enfin, si vous voulez exploiter le microcontrôleur ATmega 328 qu'elle supporte, il faut vous initier à son utilisation et surtout à sa programmation, ce qui n'est pas forcément très facile, et ce d'autant que nous allons vous proposer maintenant une toute autre approche.

10 • POUR ALLER PLUS LOIN : ASSOCIER RASPBERRY PI ET ARDUINO

Arrivé à ce stade de l'ouvrage vous devez être à même d'utiliser la majorité des applications pouvant fonctionner sous Raspbian Wheezy et donc sur votre Raspberry Pi, et même si nous ne vous les avons pas toutes présentées, vous savez désormais manipuler suffisamment bien `apt-get` pour rechercher, télécharger et installer celles qui vous feraient défaut.

Pour ce qui est de concevoir des circuits d'interface et des programmes aptes à répondre à vos besoins d'automatisation par exemple nous avons vu, depuis le chapitre 7, comment utiliser les principaux « périphériques » qu'il est possible de connecter au Raspberry Pi : poussoirs, claviers, afficheurs, relais, moteurs, capteurs, circuits à interface série, I2C ou SPI, etc. Il ne vous reste donc plus qu'à utiliser nos exemples, que ce soient les schémas ou les exemples de programmes, et à les assembler comme des briques de Lego© pour parvenir à vos fins.

Il est cependant bien évident que nous n'avons pas pu traiter toutes les interfaces possibles tant les usages du Raspberry Pi sont aujourd'hui variés. Ainsi, par exemple nous n'avons pas vu comment envoyer par Internet les résultats de mesures effectuées par des capteurs, comment télécommander, par ce même Internet, des relais connectés en sortie du Raspberry Pi ou bien encore comment utiliser un GPS, un accéléromètre, un gyroscope et bien d'autres systèmes tous plus évolués les uns que les autres.

Il faut dire que, même si le connecteur GPIO du Raspberry Pi lui permet déjà de s'interfacer avec nombre d'éléments externes, il lui manque tout de même un certain nombre de possibilités pour lui permettre de piloter facilement des circuits évolués ou disposant d'interfaces spécifiques.

En outre, le langage Python, pour simple d'utilisation qu'il soit, est loin d'être un langage temps réel, ou même tout simplement d'exécution rapide, et cela peut s'avérer rédhibitoire dans certaines situations.

Pour remédier à tout cela, et pour aller plus loin dans les possibilités d'interfaçage de votre Raspberry Pi avec le monde extérieur, il existe une solution : celle de l'associer avec un Arduino.

10.1 Qu'est-ce que l'Arduino ?

Si vous ne le connaissez pas encore, ce qui serait surprenant car il est antérieur au Raspberry Pi et au moins aussi célèbre que lui sinon plus, sachez donc que l'Arduino, ou plutôt les Arduino devrions nous écrire car ils ont fait des petits, est une simple carte électronique de format comparable à celui du Raspberry Pi, supportant un microcontrôleur, entouré du minimum de composants nécessaires à son fonctionnement de base. La figure 10.1 présente ainsi à titre d'exemple le plus connu d'entre eux : l'Arduino Uno.

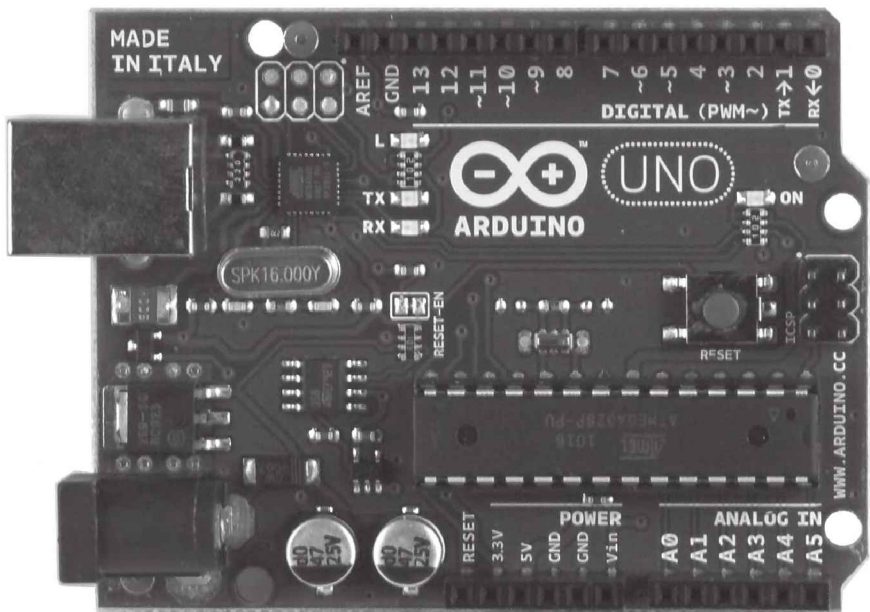


Figure 10.1 – L'Arduino Uno.

En fait, cette carte supporte également, sur sa périphérie, une rangée de connecteurs dans lesquels peuvent venir s'enficher directement des cartes d'interface, appelées les « shields ». Ces cartes d'interface sont aujourd'hui très nombreuses et couvrent quasiment tous les besoins des applications, parfois même des plus « exotiques ». Mais ce n'est pas tout. L'Arduino est également équipé d'un connecteur USB permettant de le raccorder à un micro-ordinateur, PC ou Mac à l'origine mais aussi, bien évidemment aujourd'hui, Raspberry Pi. Ce micro-ordinateur associé est utilisé pour écrire le programme destiné à piloter l'Arduino, mais également pour programmer le microcontrôleur qui équipe l'Arduino sans aucune intervention matérielle de votre part.

Cerise sur le gâteau, le projet Arduino dans son ensemble, c'est-à-dire les cartes mais aussi les logiciels associés, appartiennent au monde du libre, tout comme le Raspberry Pi.

10.2 Pourquoi relier Raspberry Pi et Arduino ?

Puisque le point faible du Raspberry Pi se situe au niveau de son port GPIO, qui ne lui permet pas de raccorder tous les types de circuits d'interface souhaitables, tandis que l'Arduino ne souffre pas de telles limitations, que ce soit au travers des shields commerciaux ou des réalisations personnelles, et comme l'Arduino et le Raspberry Pi peuvent être reliés via leurs ports USB respectifs, l'idée de les faire communiquer via cette liaison a germé dans la tête de certains auteurs de logiciels.

Si donc vous avez aujourd'hui des besoins d'entrées/sorties qui ne peuvent être satisfaits par le seul Raspberry Pi, une solution possible consiste à traiter ces derniers avec un Arduino et à le relier au Raspberry Pi, sur lequel bien sûr devra s'exécuter un programme approprié.

Pour ce qui est du développement de l'application côté Arduino, nous vous renvoyons aux deux livres que nous lui avons consacré : « Arduino – Maîtrisez sa programmation et ses cartes d'interface » qui contient une initiation complète à l'Arduino, à son logiciel de développement et aux interfaces les plus courantes, et « Arduino – Applications avancées » qui aborde des interfaces plus complexes telles que GPS, liaison radio XBee, reconnaissance vocale ou encore reconnaissance d'image. Ces deux livres sont disponibles, tout comme le présent ouvrage, aux éditions Dunod.

Pour ce qui est du côté Raspberry Pi, la meilleure solution actuelle consiste à faire appel à **Nanpy**. C'est un ensemble de logiciels comprenant une bibliothèque Python, permettant d'appeler les interfaces connectées à l'Arduino de la même façon que si on programmait directement l'Arduino lui-même, ce qui est très confortable. Par ailleurs **Nanpy** comprend également un programme serveur, à placer dans l'Arduino, afin de gérer les échanges avec cette bibliothèque.

Au moment où ces lignes sont écrites, **Nanpy** est en plein développement par son auteur et ne supporte pas encore toutes les méthodes du langage de programmation de l'Arduino, mais cela ne cesse d'évoluer et il est probable que, lorsque vous lirez ce livre, cela ait beaucoup progressé.

Au stade où nous en sommes de cet ouvrage, nous n'aborderons pas l'utilisation de **Nanpy** car, d'une part comme nous venons de le dire le programme n'est pas finalisé, d'autre part cela nécessiterait de très longs développements qui n'intéresseraient que ceux d'entre vous qui souhaitent coupler à leur Raspberry Pi des interfaces évoluées.

Si vous souhaitez suivre le développement de **Nanpy**, sachez cependant que vous pourrez trouver les informations le concernant à cette adresse : <https://pypi.python.org/pypi/nanpy>.

Ceci étant, **Nanpy** n'est pas la seule solution apte à faire communiquer un Arduino et un Raspberry Pi via leurs ports USB respectifs et on peut aussi, par exemple, utiliser **Firmata** côté Arduino, avec un programme Python de communication sous forme série asynchrone via l'USB côté Raspberry Pi.

Toutes ces possibilités, ainsi que d'autres utilisations plus évoluées du Raspberry Pi, feront très certainement l'objet d'un autre livre que vous pourrez alors aborder en toute confiance grâce à toutes les connaissances de base que vous avez acquises dans le présent ouvrage.

Codes du paramètre `hdmi_mode` contenu dans le fichier `config.txt` en fonction du format d’affichage désiré.

Le premier tableau correspond au paramètre `hdmi_group` égal à un pour la norme CEA tandis que le second tableau correspond au paramètre `hdmi_group` égal à deux pour la norme VESA.

hdmi_group = 1 – norme CEA

Code	Format d’affichage
1	640 x 480 (VGA)
2	480p 60 Hz
3	480p 60 Hz 16/9
4	720p 60 Hz
5	1080i 60 Hz
6	480i 60 Hz
7	480i 60 Hz 16/9
8	240p 60 Hz
9	240p 60 Hz 16/9
10	480i 60 Hz quadruplement de pixels
11	480i 60 Hz quadruplement de pixels 16/9
12	240p 60 Hz quadruplement de pixels
13	240p 60 Hz quadruplement de pixels 16/9
14	480p 60 Hz doublement de pixels
15	480p 60 Hz doublement de pixels 16/9

Code	Format d'affichage
16	1080p 60 Hz
17	576p 50 Hz
18	576p 50 Hz 16/9
19	720p 50 Hz
20	1080i 50 Hz
21	576i 50 Hz
22	576i 50 Hz 16/9
23	288p 50 Hz
24	288p 50 Hz 16/9
25	576i 50 Hz quadruplement de pixels
26	576i 50 Hz quadruplement de pixels 16/9
27	288p 50 Hz quadruplement de pixels
28	288p 50 Hz quadruplement de pixels 16/9
29	576p 50 Hz doublement de pixels
30	576p 50 Hz doublement de pixels 16/9
31	1080p 50 Hz
32	1080p 24 Hz
33	1080p 25 Hz
34	1080p 30 Hz
35	480p 60 Hz quadruplement de pixels
36	480p 60 Hz quadruplement de pixels 16/9
37	576p 50 Hz quadruplement de pixels
38	576p 50 Hz quadruplement de pixels 16/9
39	1080i 50 Hz
40	1080i 100 Hz
41	720p 100 Hz

Code	Format d'affichage
42	576p 100 Hz
43	576p 100 Hz 16/9
44	576i 100 Hz
45	576i 100 Hz 16/9
46	1080i 120 Hz
47	720p 120 Hz
48	480p 120 Hz
49	480p 120 Hz 16/9
50	480i 120 Hz
51	480i 120 Hz 16/9
52	576p 200 Hz
53	576p 200 Hz 16/9
54	576i 200 Hz
55	576i 200 Hz 16/9
56	480p 240 Hz
57	480p 240 Hz 16/9
58	480i 240 Hz
59	480i 240 Hz 16/9

hdmi_group = 2 – norme VESA

Code	Format d'affichage
1	640 x 350 85 Hz
2	640 x 400 85 Hz
3	720 x 400 85 Hz
4	640 x 480 60 Hz
5	640 x 480 72 Hz
6	640 x 480 75 Hz

Code	Format d'affichage
7	640 x 480 85 Hz
8	800 x 600 56 Hz
9	800 x 600 60 Hz
10	800 x 600 72 Hz
11	800 x 600 75 Hz
12	800 x 600 85 Hz
13	800 x 600 120 Hz
14	848 x 480 60 Hz
15	1024 x 768 43 Hz
16	1024 x 768 60 Hz
17	1024 x 768 70 Hz
18	1024 x 768 75 Hz
19	1024 x 768 85 Hz
20	1024 x 768 120 Hz
21	1152 x 864 75 Hz
22	1280 x 768
23	1280 x 768 60 Hz
24	1280 x 768 75 Hz
25	1280 x 768 85 Hz
26	1280 x 768 120 Hz
27	1280 x 800
28	1280 x 800 60 Hz
29	1280 x 800 75 Hz
30	1280 x 800 85 Hz
31	1280 x 800 120 Hz
32	1280 x 960 60 Hz
33	1280 x 960 85 Hz

Code	Format d'affichage
34	1280 x 960 120 Hz
35	1280 x 1024 60 Hz
36	1280 x 1024 75 Hz
37	1280 x 1024 85 Hz
38	1280 x 1024 120 Hz
39	1360 x 768 60 Hz
40	1360 x 768 120 Hz
41	1400 x 1050
42	1400 x 1050 60 Hz
43	1400 x 1050 75 Hz
44	1400 x 1050 85 Hz
45	1400 x 1050 120 Hz
46	1440 x 900
47	1440 x 900 60 Hz
48	1440 x 900 75 Hz
49	1440 x 900 85 Hz
50	1440 x 900 120 Hz
51	1600 x 1200 60 Hz
52	1600 x 1200 65 Hz
53	1600 x 1200 70 Hz
54	1600 x 1200 75 Hz
55	1600 x 1200 85 Hz
56	1600 x 1200 120 Hz
57	1680 x 1050
58	1680 x 1050 60 Hz
59	1680 x 1050 75 Hz
60	1680 x 1050 85 Hz

Code	Format d'affichage
61	1680 x 1050 120 Hz
62	1792 x 1344 60 Hz
63	1792 x 1344 75 Hz
64	1792 x 1344 120 Hz
65	1856 x 1392 60 Hz
66	1856 x 1392 75 Hz
67	1856 x 1392 120 Hz
68	1920 x 1200
69	1920 x 1200 60 Hz
70	1920 x 1200 75 Hz
71	1920 x 1200 85 Hz
72	1920 x 1200 120 Hz
73	1920 x 1440 60 Hz
74	1920 x 1440 75 Hz
75	1920 x 1440 120 Hz
76	2560 x 1600
77	2560 x 1600 60 Hz
78	2560 x 1600 75 Hz
79	2560 x 1600 85 Hz
80	2560 x 1600 120 Hz
81	1368 x 768 60 Hz
82	1920 x 1080 (1080p) 60 Hz
83	1600 x 900
84	2048 x 1152
85	1280 x 720 (720p) 60 Hz
86	1366 x 768

4/3 63
4N35 153
14/9 63
16/9 63
1080p 91
/SS 191

A

Abiword 70
Accéléromètre 201
Accessoires 24
Addition 107
Adobe 27
Adresse 185
Adresse IP 43
ADS1015 186
Affichage 62
Afficheurs à cristaux liquides 153
Afficheurs à LED 153
Afficheurs alphanumériques à cristaux liquides 156
Aide syntaxique 95
Alimentation 5, 123
Alimentation à découpage 126
Alimentation linéaire 125
Alsa 86
Amixer 86
Amplificateur à transistor 143
Ampoule au mercure 149
And 108
Anodes communes 156
Apache 81
Apostrophes 102
Appel général 185
Applications favorites 28
Apt 39
Apt-cache 39
Apt-get 39
Arborescence 32, 34

Arduino 202
ARM 11, 64
Arm_freq 64
Arrêter le Raspberry Pi 22
Atmel ATmega328 199
Auto MDI 10
AVR 199
AZERTY 19

B

Batterie 124
BCM 129
Bibliothèques 100
Bin 33
BIOS 17, 61
Bit d'acquittement 184
Blocs d'instructions 100
Bloc secteur 5
Boot 33
Boucle conditionnelle 111
Boucle itérative 110
« Box » Internet 43
Break 112
Bureautique 69
Bus I2C 181
Bus série trois fils 190

C

Cache 39
Capteur analogique résistif 178
Carte SD 4, 10, 64
Cathodes communes 156
Cd 34
CEA 62
Chaîne de caractères 102, 113
Charges alimentées en continu 147
Charges reliées au secteur 148
Chargeur 5
Chipset 52

Chmod 36
Cinch 4, 9, 63
Classe 119
Clavier 7, 28
Clavier Français 19
Clé 106
Cleanup 165
Clé USB 32
Clé USB – WiFi 10
Clé WEP 48
Clé WiFi 47
Clé Wi-Pi 47
Clé WPA 48
Close 115
Commentaire 99
Commutateurs 149
Compilateur 94
Concaténation 107
Condition d'arrêt 184
Condition de départ 184
Config_hdmi_boost 62
Config.txt 61, 92
Console 29
Constante de temps 179
Continue 112
Convertisseur analogique/digital 176, 186
Convertisseur digital/analogique 176
Core_freq 65
CPHA 192
CPOL 192
CSI 4
CTN 178
CTP 178

D

Darlington 147, 174
Debian 24
Décimal 102

Def 116
 Deux points 100
 Dev 33
 DHCP 43
 Dictionnaire 106
 Dièse (#) 99
 Différent 108
 Digital_read 196
 Digital_write 196
 Digital_write_pullup 196
 Dillo 26
 Diode de protection 144, 147
 Disable_overscan 63
 Disques durs externes 16
 Distribution Linux 11
 Division 107
 Division entière 107
 Dmesg 52
 DNS 45
 Dolby Digital 91
 Droit d'accès en écriture 36
 Droit d'accès en lecture 36
 Droit d'exécution 36
 Droits 35
 DSI 4
 DTS 91
 DVD 92
 DVI 8, 62

E

Écran 28
 Écriture dans un fichier 114
 Éditeur de texte 44
 Éducation 25
 Égal 108
 Élévation à une puissance 107
 Elif 110
 Else 109
 Else if 110
 Enable 158
 Entier 102, 113
 Esclave 184
 Espaces 100
 Et 108
 Etc 33
 Ethernet 3
 Expression conditionnelle 109

F

False 102
 Faux 102

Films HD 91
 Firmata 204
 Flottant 102, 113
 Fonction 116
 For 110
 Full HD 4
 Fusibles 6

G

Calculator 24
 Gertboard 198
 Gestionnaire de bases de données 81
 Gestionnaire des tâches 28
 Gigabit Ethernet 3, 10
 Gnumeric 71
 Gparted 56
 GPIO 4, 128
 GPIO.add_event_detect 140
 GPIO.cleanup 141
 GPIO.event_detected 141
 GPIO.input 140
 GPIO.output 140
 GPIO.remove_event_detect 140
 GPIO.setmode 139
 GPIO.setup 139
 GPS 201
 GPU 64, 84
 Gpu_freq 64
 Gradation de puissance 170
 Gradation de tension 172
 Graphismes 27
 Grep 52
 Groupe de travail 79
 Groupes 31
 Groups 31
 Guillemets 102
 Gyroscope 201

H

H264_freq 65
 Haute tension 152
 HD44780 156
 HDMI 4, 8, 62, 86, 91
 Hdmi_drive 62
 Hdmi_force_hotplug 62
 Hdmi_group 62
 Hdmi_mode 62
 Hdmi_safe 63
 Home 33

Hostname 43
 Html 81
 HTPC 84, 87
 Hub actif 6
 Hub USB 6
 Humble Pi 134

I

I2C 4
 I2c-bcm2708 188
 I2cdetect 188
 I2c-dev 187
 I2c-tools 188
 IDE 93
 IDLE 27, 93
 IDLE 3 27, 93
 If 109
 ILS 149
 Image 60
 Import 101, 118
 Indentation 100
 Index 103
 Inférieur 108
 Inférieur ou égal 108
 Init 196
 Init_emmc_clock 64
 Input 114
 Insert 105
 Instable 67
 Install 40
 Interfaces 44
 Interface série 4
 Internet 26, 45
 Interpréteur 94
 Invite de commande 30
 Isp_freq 65

J

Jack audio 91
 Jack femelle stéréo 4
 Jack stéréo 87

L

L298 174
 LCD 156
 LDR 178
 Leafpad 24
 Lecture dans un fichier 115
 LED 141
 LED de puissance 142
 LED d'état 3

Len 104
 Lib 33
 LibreOffice 75
 Licences 92
 Ligne de commande 29
 Liste 103
 Localisation 21, 41
 Logiciel de dessin 72
 Login 22, 30
 Lost+found 33
 Ls 35
 LXDE 23
 LXTerminal 24, 29

M

Maître 184
 Man 38
 MCP23S17 193
 Media 33
Media center 84, 87
 Mémoire de masse 15, 32
 Méthode 105, 119
 Micro USB 4
 Microwire 190
 Midori 26
 Midori Private Browsing 27
 Mise à jour 42
 MISO 191
 Mkdir 35
 Mnt 33
 Moc 85
 MOC 3041 148
 Mocp 85
 Mode 4 bits 158
 Mode 8 bits 157
 Mode « turbo » 66
 Modulation de largeur
 d'impulsions 164
 Module 117
 Modulo 107
 Moniteur de la température 67
 Monter 33, 54
 MOSI 191
 Mot de passe 22, 30
 Moteur à courant continu 170,
 173
 Moteur pas à pas unipolaire
 166
 Moteurs pas à pas 166
 MP3 85
 MPEG2 92

Multiplexage 155
 Multiplication 107
 Mv 35
 Mysql 81

N

Nameserver 46
 Nano 44
 Nanpy 203
 NAS 91
 NetSurf 27
 Nom de l'ordinateur 30
 Nom de variable 101
 Nom d'utilisateur 22, 30
 Non 109
 Not 109
 NTSC 4, 63

O

Objet 105, 119
 Open 114
 Openbox Configuration
 Manager 28
 OpenDNS 46
 OpenOffice 75
 Opérateur de formatage 113
 Opérateurs arithmétiques 107
 Opérateurs de comparaison 108
 Opérateurs logiques 108
 Opt 33
 Or 109
 Ou 109
 Ou exclusif 109
 Ou inclusif 109
 Outils système 28
 Overclocker 64
 Over_voltage 65
 Over_voltage_sdram 65
 Over_voltage_sdram_c 66
 Over_voltage_sdrma_i 66

P

Packages 39
 PAL 63
 Paquets 39
 Partage de fichiers 78
 PartedMagic 57
 Partitions 54
 Passerelle 45
 Password 22

Péritélévision 9
 Photocoupleur 143, 152
 Photoshop 74
 Photo triac 148
 Php 81
 PhpMyAdmin 81
 Pi Cobbler 131
 PiFace 192
 Pipe 52
 Pi Proto Plate 132
 Pi-View 8
 Pop 104
 Poussoir de reset 121
 Poussoirs 149
 Préférences 28
 Print 95, 112
 Prise de décision 109
 Proc 33
 Programmation 27
 Programmation orientée objet
 119
 Programme exécutable 98
 Prompt 30
 PS/2 7
 Pull up 149
 Pulseaudio 86
 Purge 41
 Pwd 34
 PWM 164, 171, 177
 Python 93
 Python-smbus 188

Q

QWERTY 20

R

RAM 64
 Range 111
 Rapport cyclique 171
 Raspbian Wheezy 11, 17
 Raspbmc 87
 Raspi-blacklist.conf 188
 Raspi-config 55, 66
 RCA 4
 Read 115
 Read_byte 189
 Rebondissements 151
 Reboot 45
 Récepteur infrarouge 92
 Reconnaissance d'image 203
 Reconnaissance vocale 203

Redimensionnement de la
partition 55
Réel 102
Register Select 159
Règles de syntaxe 99
Régulateurs de tension 123
Relais 141
Relais statique 148
Remove 41
Répertoire courant 33
Répertoire de travail 30
Répertoire parent 33
Repositories 39
Reset 121
Résistance de tirage 149
Resolv.conf 46
Reste 107
Return 117
Révisions 122
Rm 35
Rmdir 35
Root 30
Roues codeuses 149
Routeur 43
RPi.GPIO 138
RPiO 164, 173
RS 159
R/W barre 158

S

Samba 78
Saut à la ligne 102
Sauvegarde 60
Sbin 33
SCK 191
SCL 182
Scratch 26
SD 10
SDA 182
SD HC 10
SDI 191
SDO 191
Sdram_freq 64
Sdtv_aspect 63
Sdtv_mode 63
Search 39

Secteur EDF 127, 152
Sequence 110
Serveur 80
Servo de radiocommande 161
Set_servo 165
SHA-1 11
Shasum 12
Shields 202
Shutdown 121
Slash 32
Smartphones 5
Smb.conf 78
SoC 64
Son 86
Souris 7, 28
Soustraction 107
SPI 4, 190
Spi-bcm2708 188
Squeak 26
SSID 48
Startx 23
Stop_servo 165
Sudo 31
Suite bureautique 75
Supérieur 108
Supérieur ou égal 108
Super-utilisateur 30
SVCD 92
Sys 33
Système de fichiers 32
System on a Chip 64

T

Tab 100
Tableur 71
Tabulations 100
Télécommandes 91
Tensions analogiques 176
Terminal administrateur 24
The Gimp 73
Tmp 33
Traitement de texte 70
True 102
Tuple 105
Type 102

U

ULN2803 144, 167, 174, 197
Update 42
Upgrade 42
USB 1.0 3
USB 2.0 3
Useradd 32
Usrc 33

V

V3d_freq 65
Valeur moyenne 171
Var 33
Variable 101
Variable indexée 103
Variables locales 117
VC1 92
Version A 1
Version B 1
Versions 122
VESA 62
VGA 8, 62
Vidéo composite 9
Vidéo composite PAL 4
Visionneur d'images 24
Vrai 102

W

While 111
WiFi 46
WiFi Config 49
Win32 Disk Imager 14, 60, 89
Workgroup 79
Wpa-gui 27
Write 115
Write_byte 189

X

Xarchiver 24
XBee 203
XBMC 87
XBMC Openelec 87
Xpaint 72
Xpdf 27