N°7 | JUILLET - AOÛT 2015

HACKABLE MAGAZINE

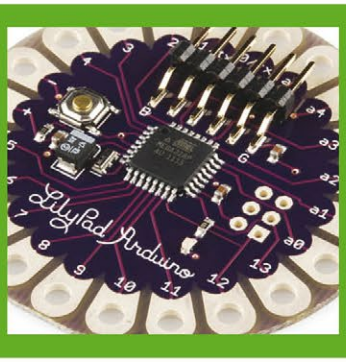
DÉMONTEZ | COMPRENEZ | ADAPTEZ | PARTAGEZ

France METRO : 7,90 € – CH : 13 CHF – BEL/PORT.CONT : 8,90 € – DOM TOM : 8,50 € – CAN : 14 $ cad – TUNISIE : 18 TND – MAR : 100 MAD

ARDUINO

Utilisez facilement différentes cartes avec le logiciel Arduino 1.6.5

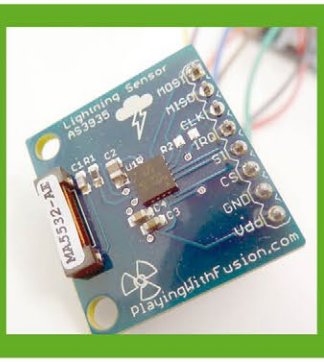
p. 08



MÉTÉO

Détectez les impacts de foudre et les orages à plusieurs kilomètres

p. 22



CLOUD PERSO

Stockez vos données avec une Raspberry Pi, un disque USB et Seafile

p. 70

PERFORMANCES

Qu'est-ce qui change vraiment avec la nouvelle Raspberry Pi 2 ?

p. 14

Un « Arduino » avec Wifi pour moins de 10 euros ?

WIFI ultra-économique !

p. 38

1 - Découvrez les modules ESP8266

2 - Ajoutez le support Arduino

3 - Programmez votre serveur web wireless

BONNES BASES

Découvrez 4 solutions et montages pour faire clignoter vos leds

p. 86

HAUTE TENSION

Apprenez comment contrôler en toute sécurité des appareils ~230V

p. 92

L 19338 - 7 - F: 7,90 € - RD

NE MANQUEZ PAS OPEN SILICIUM N°15 !

FOCUS :

INITIEZ-VOUS AU RÉSEAU TEMPS RÉEL

AVEC XENOMAI ET RTNET

TOUJOURS DISPONIBLE SUR :

www.ed-diamond.com

ÉDITO

Un an !

Souvenez-vous, l'été dernier arrivait en kiosque le tout premier numéro de *Hackable*. Depuis, un an et presque 600 pages plus tard, force est de constater que la liste d'idées et de sujets pour de futurs articles n'a absolument pas baissé, bien au contraire. Car voyez-vous c'est là précisément l'une des choses que j'apprécie tout particulièrement dès lors qu'on se met à créer quelque chose, qu'il s'agisse de matériels ou de logiciels. Chaque nouvelle idée est une occasion d'apprendre et de mettre en pratique un nouveau concept, ce qui découle forcément sur d'autres idées.

En une année également, les choses ont un peu changé dans l'écosystème de l'électronique de loisirs. Les plateformes ont gagné en maturité tout comme les environnements de développement. Prenons par exemple la toute récente version 1.6.5 de l'IDE Arduino (17/06) qui ajoute la possibilité de « replier » les lignes de code dans l'éditeur ou encore s'étoffe d'une fonction d'exportation du croquis compilé. Ce que vient compléter la gestion des cartes et des bibliothèques, dont nous parlerons dans ce numéro, introduite avec la 1.6.4.

Un autre exemple concerne Energia, l'environnement de développement « à la Arduino » pour les cartes Texas Instrument Launchpad. La récente et puissante MSP432P401R LaunchPad à cœur ARM Cortex M4F (48 Mhz) a permis, dans Energia 15, l'introduction d'une importante nouveauté : le multitâche. Avec cette plateforme, il devient donc possible d'exécuter plusieurs croquis en parallèle ! Chacun d'eux prend la forme d'une tâche distincte (avec son setup() et son loop()) fonctionnant sur un système d'exploitation appelé TI-RTOS (en logiciel libre sous licence BSD). Le tout avec la même logique et la même simplicité qu'un croquis classique.

C'est là, à mon avis, que réside la plus grande difficulté. La technologie offre de plus en plus de ressources pour un prix identique (ou inférieur). Pourtant, il faut arriver à maintenir un certain niveau d'accessibilité et de simplicité, tout en permettant l'utilisation de ces nouvelles ressources. Nous en reparlerons dans l'article sur la Raspberry Pi 2, mais le fait est que, sans évoluer dans nos habitudes, le changement n'a presque aucun intérêt. Mais ceci, après tout, est une vérité universelle...

Fort heureusement, nous sommes encore et toujours, les seules machines à auto-apprendre sur cette planète (n'en déplaise à Google, Boston Dynamics et feu Douglas Adams). Faisons donc simplement ce pour quoi nous sommes le plus adaptés : apprenons sans aucune limite !

Denis Bodor

Hackable Magazine
est édité par Les Éditions Diamond

LES ÉDITIONS DIAMOND
STIMULANT DE MATIÈRE GRISE

10, Place de la Cathédrale - 68000 Colmar
Tél. : 03 67 10 00 20 – Fax : 03 67 10 00 21
E-mail : lecteurs@hackable.fr
Service commercial : cial@ed-diamond.com
Sites : www.ed-diamond.com
Directeur de publication : Arnaud Metzler
Rédacteur en chef : Denis Bodor
Réalisation graphique : Kathrin Scali
Responsable publicité : Valérie Fréchard,
Tél. : 03 67 10 00 27 v.frechard@ed-diamond.com
Service abonnement : Tél. : 03 67 10 00 20

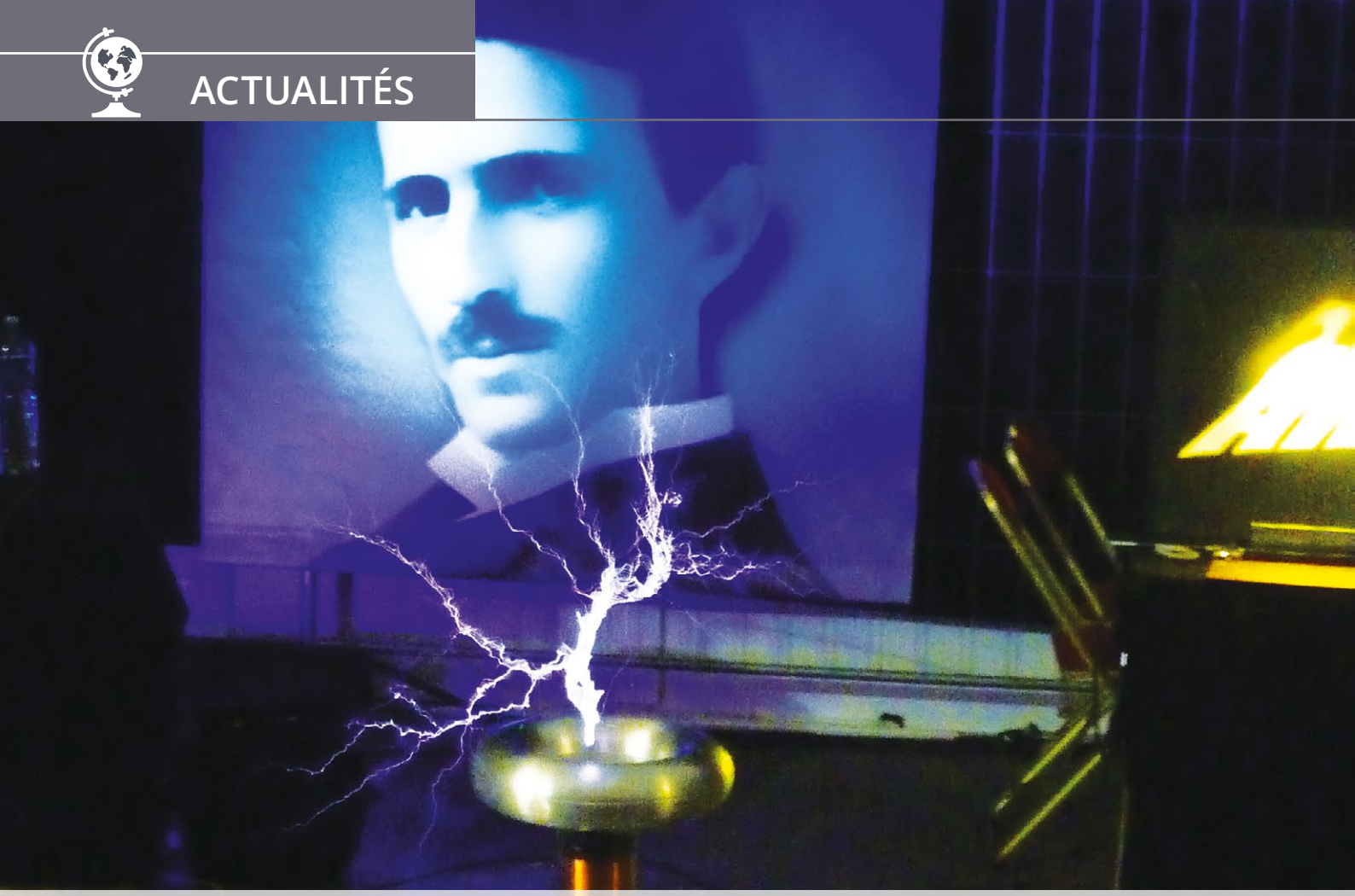
Impression : pva, Landau, Allemagne
Distribution France : (uniquement pour les dépositaires de presse)
MLP Réassort : Plate-forme de Saint-Barthélemy-d'Anjou. Tél. : 02 41 27 53 12

Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04
Service des ventes : Abomarque : 09 53 15 21 77
IMPRIMÉ en Allemagne - PRINTED in Germany
Dépôt légal : À parution,
N° ISSN : 2427-4631
Commission paritaire : K92470
Périodicité : bimestriel
Prix de vente : 7,90 €

MIXTE
Papier issu de sources responsables
FSC® C015136

Suivez-nous sur Twitter
@hackablemag

SOMMAIRE



MAKERFAIRE PARIS 2015 : PLUS GRANDE, PLUS OUVERTE, PLUS FAMILIALE !

Denis Bodor

Les 2 et 3 mai derniers s'est tenue la MakerFaire Paris, en plein milieu de la 111ème édition de la Foire de Paris. Cette seconde édition de la Faire a été une évolution importante depuis celle de l'année dernière, au 104. Nous étions, bien entendu, présents et vous proposons ici un petit retour sur cet événement mémorable.

coromonadalix

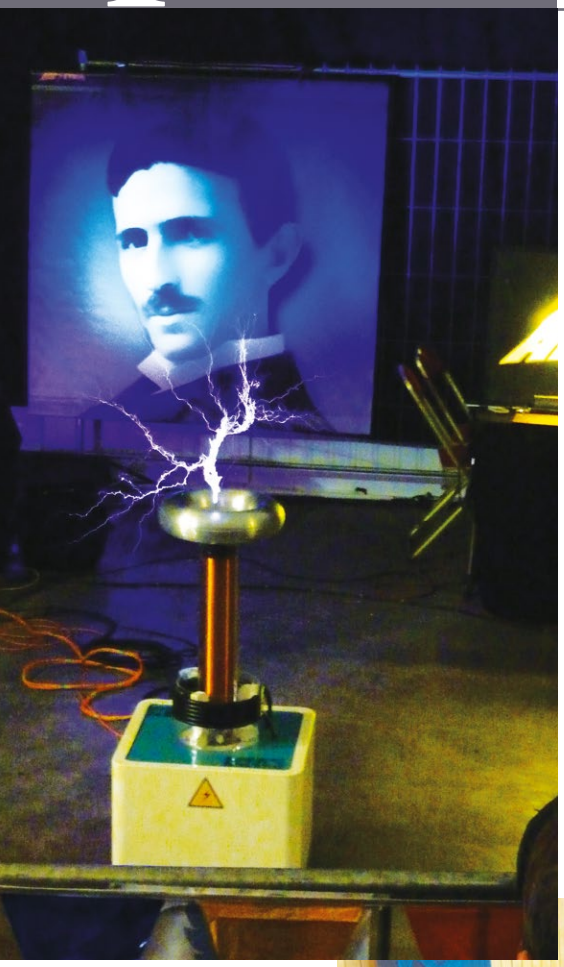
Électronique, robotique, bricolage, impression 3D, sculpture, tatouage, up-cycling, biologie... et bière bretonne. Le mélange parfait pour deux jours de découvertes et d'échanges avec son imagination comme seule limite.

La première MakerFaire parisienne, en 2014, a eu lieu au 104, Paris 19ème arrondissement et faisait suite, comme cette année, à la mini-MakerFaire bretonne de Saint-Malo (11/12 avril). Mais cette édition 2015 était différente sur bien des aspects.

Tout d'abord, nous avons le lieu et le moment. Paris bien sûr, mais plus précisément le parc des expositions porte de Versailles qui accueillait, en même temps, la Foire de Paris. Pour rappel, cette foire est née de l'Exposition Universelle de 1889 et de la volonté de poursuivre annuellement cette célébration de l'innovation et de l'invention à une échelle plus modeste. C'est donc tout naturellement qu'une MakerFaire, point de convergence des inventeurs à l'imagination débordante, trouve sa place lors d'une telle manifestation.

C'est ainsi tout un pavillon qui était dédié à cette seconde MakerFaire Paris, et le succès était, bien entendu au rendez-vous. Mieux encore, cette édition, nous l'avons clairement constaté durant tout le week-end, s'est avérée bien plus propice à la découverte par les visiteurs. En effet, la Foire de Paris étant une exposition regroupant bien des domaines, certains des visiteurs arrivaient à la MakerFaire sans vraiment savoir où ils mettaient les pieds. Nous avons pu lire sur les visages bien de la surprise, de l'émerveillement et de la fascination.

Cela peut paraître anodin, mais force est de constater que bien des personnes, devant l'avalanche de nouveautés et d'innovations technologiques, finissent par ne plus être étonnées de rien. Qu'il s'agisse de transporter sur soi un véritable ordinateur faisant aussi office de téléphone ou de montre, envoyer des messages instantanément d'un bout à l'autre de la planète ou régler la température dans son salon depuis n'importe où, tout ceci paraît souvent parfaitement normal, accepté, voire exigible. Si l'on ajoute à cela le fait que peu de monde porte un véritable intérêt à la compréhension de ce qui nous entoure, on se retrouve précisément dans une situation que Carl Sagan avait brillamment résumé dès 1997 : « Nous vivons dans une société totalement dépendante de la science et de la technologie et pourtant avons habilement organisé les choses de manière à ce que presque personne ne comprenne la science et la technologie. Ce sont là les prémices



L'une des attractions attirant le plus grand nombre de visiteurs : une bobine de Tesla de l'association Arc-ethic en pleine représentation musicale.

d'un désastre. On pourra continuer ainsi un temps, mais tôt ou tard ce mélange détonnant d'ignorance et de puissance va nous exploser au visage ». C'était vrai à l'époque et ce l'est d'autant plus aujourd'hui...

L'émerveillement est quelque chose qui se fait rare alors que c'est généralement le précurseur de l'intérêt, puis de la recherche, la découverte et l'apprentissage. Sans émerveillement, il n'y a pas de fascination et de passion. Voir ainsi les visiteurs éblouis des réalisations d'amateurs qu'ils découvraient était donc un constat rassurant et je ne peux m'empêcher d'espérer que parmi eux, beaucoup sont partis avec des idées plein la tête et une réelle motivation à s'essayer aux domaines.

Il faut avouer que côté étonnement et démonstrations les organisateurs avaient parfaitement dosé les choses. Les démonstrations de l'association Arc-ethic, par exemple attiraient énormément de monde. Il faut dire qu'une bobine de Tesla (ou plus exactement une DRSSTC (*Dual Resonant Solid State Tesla Coil*) possède une certaine aptitude à attirer l'attention tant au niveau sonore que visuel. De la même manière, les démonstrations techniques et scientifiques de Baptiste Mortier-Dumont, alias ExperimentBoy sur YouTube, ont su attirer un large public même si tout cela restait bien plus calme que ses vidéos (question de sécurité et d'assurance sans doute).

Beaucoup de choses sur le stand de l'Electrolab de Nanterre et, entre autres, une machine à coudre transformée pour l'occasion en machine à broder parfaitement fonctionnelle. Brillant !

Mais une MakerFaire n'est pas qu'un lieu de rencontre pour les amateurs de science, de technologie et d'électronique. Une bonne partie des stands étaient ceux de « crafters », un terme très américain désignant simplement les personnes qui, sur le vieux continent s'adonnent aux loisirs créatifs. C'est là ma seconde constatation importante matérialisée par cet événement : un renouveau de la création personnelle.

Beaucoup de domaines apparaissent pour certains comme nouveaux ou modernes alors que des communautés de passionnés ont toujours été

présentes, souvent de longue date. Ceci est vrai pour l'électronique, bien sûr, mais également pour la broderie, l'origami, le radio-amateurisme, la couture, l'aéromodélisme, la poterie, la mécanique, le travail du bois, le dessin... Mon sentiment sur ce renouveau est très positif car, inutile de se le cacher, la plupart de ces groupes de passionnés étaient soit vieillissant, soit implicitement constitué d'une catégorie de population précise. Qui sait, peut-être d'ici quelque temps, on ne pensera plus que je me suis perdu ou que je suis un extra-terrestre lorsque je cherche du fil ou du tissu dans une boutique de couture...

Comme l'année dernière, l'atelier soudure était l'occasion pour beaucoup de se découvrir de nouveaux talents et par la même occasion de constater que c'est bien plus facile qu'on le pense.

Peu importe en vérité qu'on appelle maintenant le loisir créatif du *crafting*, les bricoleurs des *makers*, les radio-amateurs des passionnés de SDR et l'accomplissement de projets personnels du *Do It Yourself* (DIY). L'important ce sont les actes et les créations. Si changer le terme associé à une activité suffit à lui donner une nouvelle jeunesse et à inviter de nouveaux passionnés, tant mieux !

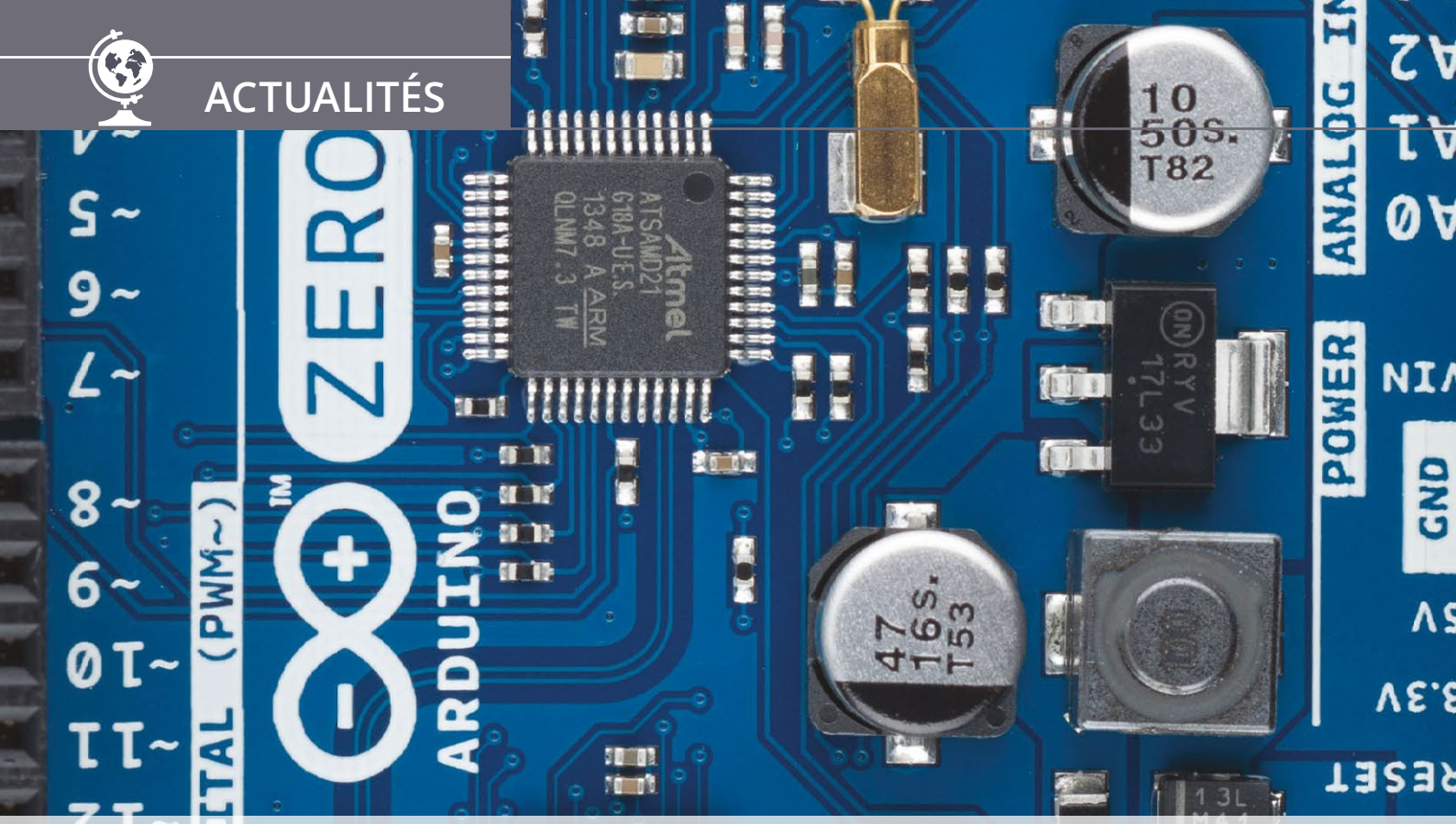
Cet « âge du faire » comme l'appellent certains se caractérise aussi par le mélange et l'accessibilité. Les domaines se recoupent de façon admirable et on peut aisément en venir, par exemple, à travailler le bois pour un projet électronique ou inversement, à découvrir l'électronique pour les besoins du travail du bois. Et ceci est bien entendu applicable à tous les domaines et, comme on l'a constaté lors de cette MakerFaire, est effectivement appliqué de façon merveilleuse et imaginative.

Le stand de PoopyCat présentait sa maison de jeux pour chats, une architecture modulaire en carton prédécoupé pour les tigres de salon. L'idée est originale même si cela reste parfaitement réalisable sans kit. Tout dépend du nombre de mains gauches que vous avez...

Dans ce tourbillon de créativité, rien d'étonnant donc que certains cherchent à prouver leurs affinités avec le domaine. C'est ainsi que parmi les principaux partenaires de cette MakerFaire parisienne on trouvait LeroyMerlin, Conrad mais aussi Microsoft et Intel. Ce dernier disposait d'un stand des plus massifs, mais étonnamment, en parlant avec les étudiants d'Epitech présents, on pouvait constater que les projets présentés faisaient une part importante aux plateformes Arduino et Raspberry Pi, les cartes Edison étant souvent simplement posées à côté.

En conclusion, cette seconde édition de la MakerFaire Paris a confirmé que le « mouvement » était engagé et n'était clairement pas sur le point de s'essouffler, bien au contraire. L'énergie qui émanait de ce pavillon, provenant à la fois des exposants et des visiteurs, laisse présager d'excellentes choses à venir et, pour ma part, confirme l'existence d'une réelle volonté et d'un réel besoin de réacquérir individuellement cette capacité qui fait de nous une espèce à part : celle de projeter une idée ou de transposer un concept... en d'autres termes, innover ! **DB**

coromonadalix



UTILISEZ DIFFÉRENTES CARTES AVEC ARDUINO 1.6.5

Denis Bodor

Arduino est un environnement de développement dédié aux cartes Arduino, mais il existe plusieurs déclinaisons de ce logiciel permettant d'utiliser des cartes autres que celles officiellement distribuées sous la marque Arduino. Jusqu'à l'arrivée de la version 1.6.4, l'utilisation d'autres cartes passait généralement par l'installation d'une copie modifiée de l'environnement. Mais tout ceci est en train de changer grâce au « Boards Manager »...

Précisons de suite que l'ensemble des projets clonant l'environnement Arduino n'a pas encore basculé sur le nouveau système de gestion des cartes et certains d'entre eux ne le feront sans doute jamais. Energia (rouge) ou Maple IDE (vert), par exemple, sont des environnements adaptés respectivement pour les plateformes Texas Instrument Launchpad (MSP430, MSP432, Tiva, C2000, Hercules) et Leaflabs Maple (STM32). Ces projets et plateformes sont sans doute trop « personnalisés » pour intégrer l'IDE Arduino officiel et/ou reposent sur une stratégie qui ne rend pas cela souhaitable (maintien d'un écosystème complet indépendant).

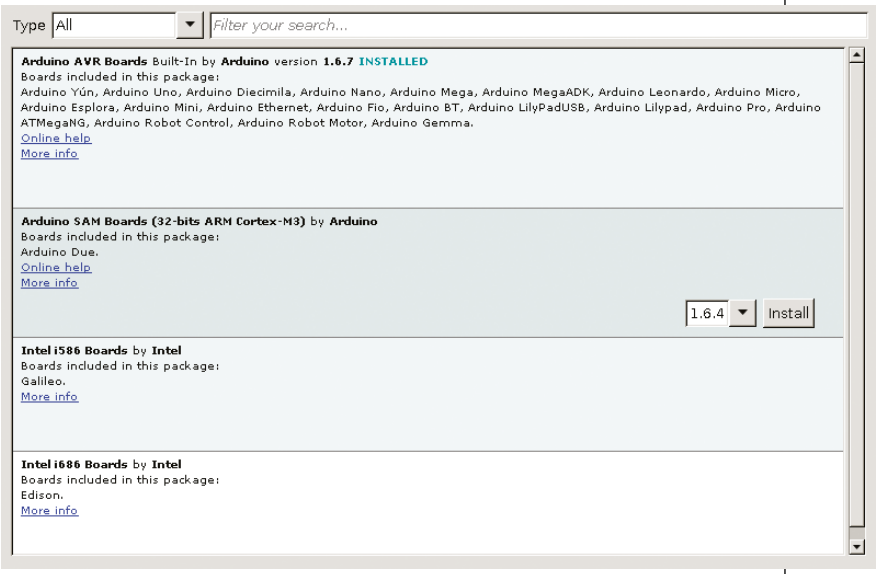
Cependant, un certain nombre de « dérivés » des cartes Arduino originales sont d'ores et déjà pris en charge. Le premier exemple qui vient naturellement à l'esprit est l'Arduino Due ainsi que les cartes Galileo et Edison d'Intel. Il faut savoir, en effet, que l'Arduino au sens premier du terme, sous la forme des cartes Uno, Mega, Leonardo, Nano, Micro, etc., repose sur des microcontrôleurs Atmel AVR. Ces cartes possèdent donc une base commune, car elles utilisent la même famille de composants et donc un ensemble d'outils logiciels (compilateur, bibliothèques « système » et programmeur) identiques (avr-gcc, avr-libc et avrdude).

Il n'en va pas de même pour l'Arduino Due construite autour du microcontrôleur AT91SAM3X8E d'Atmel qui est un ARM Cortex-M3 utilisant une architecture totalement différente, ainsi que Galileo/Edison utilisant des composants Intel compatibles i586 et i686 très proches de l'architecture Pentium ou de celle des processeurs actuels utilisés pas les PC et Mac. Ce type de différences notables implique l'utilisation en coulisse d'autres outils à la fois pour compiler les croquis et les charger dans la mémoire des composants.

Les possesseurs de cartes Arduino Due le savent, jusqu'à présent (Arduino 1.0.x), il était nécessaire d'installer une version bêta de l'environnement (versions 1.5.x) pour pouvoir utiliser et programmer la carte. Avec l'arrivée de l'environnement en version 1.6.4 et supérieur, ce n'est plus le cas, tout est géré au même endroit.

1. UTILISER LE BOARDS MANAGER

L'accès à ce nouveau système est très simple. Il vous suffit, dans l'environnement (IDE) Arduino, de passer par les menus ***Outils***, ***Type de carte*** et ***Boards Manager*** (tout en haut). Ceci a pour effet d'ouvrir une nouvelle fenêtre, présentant une liste.



Par défaut, on y trouve quatre entrées ou *packages* :

- « *Arduino AVR Boards* » : c'est l'environnement installé par défaut pour toutes les cartes Arduino à base d'Atmel AVR ;
- « *Arduino SAM Boards (32-bits ARM Cortex-M3)* » : l'environnement permettant l'utilisation l'Arduino Due, pour l'heure seule représentante de cette famille (en ce qui concerne l'Arduino Zero Pro, cf plus loin dans l'article) ;

- « *Intel i586 Boards* » : fourni par Intel, ceci assure le support des deux cartes Intel Galileo (la première et la Gen2) ;
- « *Intel i686 Boards* » : provenant également d'Intel, ceci est l'environnement pour la carte Edison.

L'installation d'un environnement supplémentaire se fera simplement en sélectionnant l'entrée et en cliquant sur le bouton ***Install*** apparaissant. L'interface procédera alors au téléchargement et à l'intégration des nouveaux éléments dans l'environnement. On constate qu'avec cette version 1.6.4, une différence est faite entre l'interface qui est l'environnement dans lequel vous programmez et l'environnement de compilation lui-même qui est composé d'un ensemble d'outils travaillant en arrière-plan.

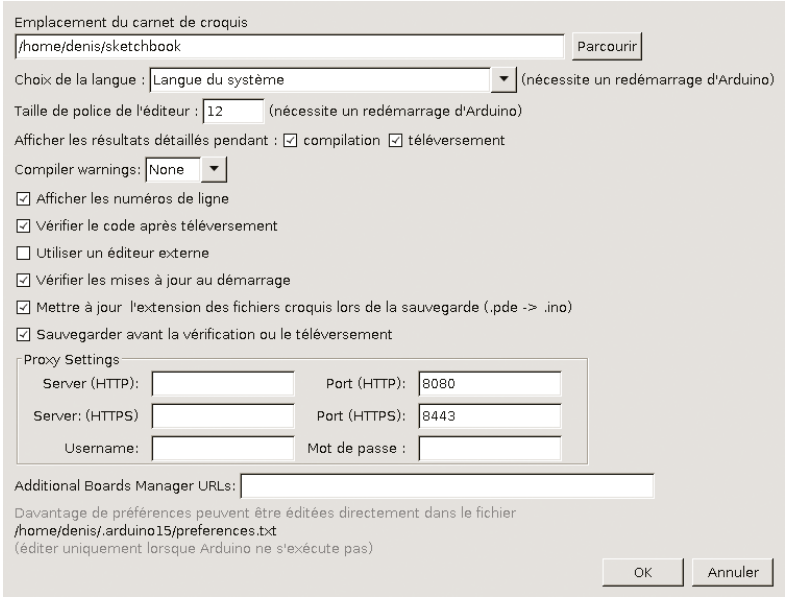
En installant le support pour l'Arduino Due par exemple, vous pourrez alors constater que dans la liste, à l'heure actuelle, nous avons donc un IDE/interface en version 1.6.4, un support AVR en 1.6.7 et un support SAM (Arduino Due) en 1.6.4. Les mises à jour des outils de développement se feront dans le « Boards Manager » et vous pouvez choisir et installer la version de votre choix, ou encore supprimer un support.

Une fois le support de la Due installé, vous devez immédiatement trouver deux nouvelles entrées dans le menu ***Outils***, ***Type de carte*** : « *Arduino Due (Programming Port)* » et « *Arduino Due (Native USB Port)* ». C'est aussi simple que cela.

2. AJOUTER D'AUTRES CARTES

La liste par défaut ne propose que quatre entrées, mais il vous est possible d'en ajouter plus. Le mécanisme utilisé par ce nouveau gestionnaire de cartes est relativement simple et est assez proche de ce qu'on peut trouver, par exemple, dans une distribution GNU/Linux. L'environnement graphique utilise des dépôts contenant les informations nécessaires au téléchargement et à l'installation d'éléments logiciels supplémentaires. En ajoutant des dépôts, on ajoute de nouveaux *packages* disponibles et donc de nouveaux supports pour des cartes. Une liste est disponible sur **http://playground.arduino.cc/Main/ArduinoOnOtherAtmelChips**.

L'ajout d'un dépôt se fait via les préférences de l'environnement graphique (***Fichier***, ***Préférences***) :



Tout en bas de la fenêtre se trouve un champ ***Additionnal Boards Manager URLs***. Celui-ci permet de spécifier une liste de liens Internet (URL), séparés par des virgules, désignant les endroits où l'on peut trouver les éléments. Voici deux exemples que nous avons testés :

- Le support pour les microcontrôleurs ATtiny45, ATtiny85, ATtiny44 et ATtiny84, de David Mellis : **https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json** ;
- Le support pour des microcontrôleurs ATmega168 et ATmega328p « nus » de Carlos Rodrigues : **https://raw.githubusercontent.com/carlosefr/atmega/master/package_carlosefr_atmega_index.json**

Pour ajouter le support, on précisera simplement ces deux URL, séparées avec une virgule, avant de retourner dans le gestionnaire de cartes. Là, après une brève mise à jour de la liste via le net, on pourra alors constater l'apparition de deux nouvelles entrées :

Il suffira alors de cliquer sur ***Install*** pour chacune d'elles afin d'ajouter le support correspondant. Notez que l'installation est bien plus rapide qu'avec le support pour l'Arduino Due. Ceci s'explique par le fait qu'il n'est pas nécessaire de télécharger et d'installer un nouveau compilateur. Il s'agit de deux supports pour des microcontrôleurs AVR et le nécessaire est déjà présent sur votre machine. La seule chose qui change dans ce cas est un ajout de quelques définitions spécifiques.

À propos d'ajout justement, en intégrant ces deux supports, vous trouverez de nouvelles entrées dans le menu ***Type de carte*** : « ATmega168 », « Atmega328p » et « ATtiny ». En choisissant l'une de ces nouvelles « cartes », le menu ***Outils*** se verra compléter de nouveaux paramètres dépendants de la plateforme cible. Ainsi, en choisissant la « carte » ATtiny, vous pourrez ensuite spécifier le modèle exact de microcontrôleur, ainsi que sa source d'horloge (*Clock*). Ces deux supports sont destinés à l'utilisation de microcontrôleurs seuls sur platine à essais ou sur un circuit « maison ». Ils peuvent être équipés d'un oscillateur à quartz comme sur les cartes Arduino, mais aussi fonctionner avec un oscillateur interne à 1 ou 8 Mhz, sensiblement moins précis, mais économisant des composants.

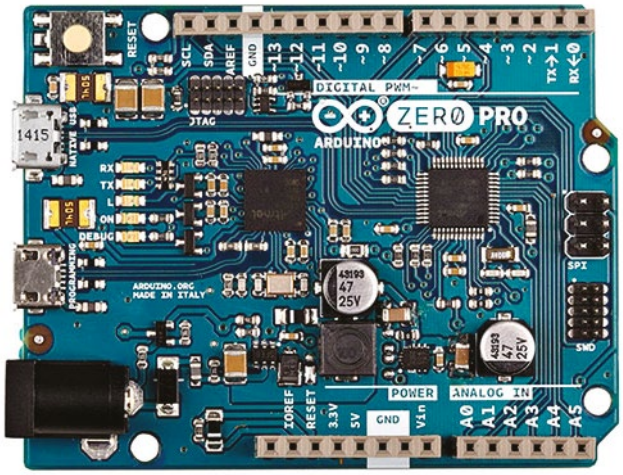
Une autre source intéressante est celle proposée par AdaFruit : **https://adafruit.github.io/arduino-board-index/package_adafruit_index.json**. Ce dépôt, à spécifier dans les préférences comme les autres, ajoute le support pour les cartes d'AdaFruit Industrie (Flora, Metro, Trinket, Gemma, etc.), mais également pour une plateforme totalement différente : l'ESP8266. Il s'agit d'une « carte » servant normalement de module Wifi, mais pouvant être également programmée comme une plateforme Arduino. Ce module tend à se populariser, aussi bien en utilisation « périphérique » où l'Arduino est alors le maître à bord, que comme plateforme de développement où l'on programme directement le microcontrôleur.

Notez cependant à propos du support ESP8266 que tous les systèmes ne sont pas supportés. Il vous faudra un Windows 32 ou 64 bits, un Mac OS X 64 bits ou un GNU/Linux 64 bits. Si vous utilisez un Mac OS X un peu ancien ou un Linux 32 bits, l'ESP8266 ne vous sera pas accessible.

3. ARDUINO 1.7.X ?!

Si vous suivez l'actualité autour d'Arduino, de la disponibilité de nouvelles cartes ou de nouvelles versions de l'environnement de développement, vous êtes peut-être tombé sur Arduino 1.7.3 ou la carte Arduino Zero Pro. Le site « officiel » pourtant ne propose que la version 1.6.4 et n'annonce pas de disponibilité pour la carte Arduino Zero. D'où vient le problème ? En réalité, il ne s'agit pas d'un problème de mise à jour ou de communication, mais du sens qu'on souhaitera donner au terme « officiel ».

Comme vous le savez, l'environnement de développement Arduino est open source, tout comme le design des cartes. Le nom « Arduino » et le logo associés, en revanche, ne répondent pas au même principe d'ouverture. Coupons de suite court à un débat inutile, la notion de marque n'a rien à voir avec celle d'open source ou de logiciel libre. Une marque est un nom, tantôt accompagné d'un logo, permettant de désigner un bien, un produit ou un service, et d'apporter la certitude à un utilisateur ou un client que ce nom représente bien ce qu'il est en droit d'attendre. Le Firefox de la fondation Mozilla est également une marque déposée. Ceci permet de protéger la fondation et les utilisateurs contre d'éventuelles « tromperies » (fortuites ou à dessin). N'importe qui peut prendre les sources de Firefox et les modifier pour les redistribuer, c'est le principe du logiciel libre. Il n'aura cependant pas le droit de dire qu'il s'agit de Firefox sans l'accord de la fondation Mozilla, car celle-ci souhaite associer ce nom



La carte Arduino Zero Pro produite par Arduino Srl et présentée uniquement sur Arduino.org. Le site Arduino.cc quant à lui, affiche la carte Arduino Zero (tout court) avec une mention « coming soon » (« arrive bientôt »). C'est l'un des nombreux aspects visibles de la bataille que se livrent les deux sociétés.

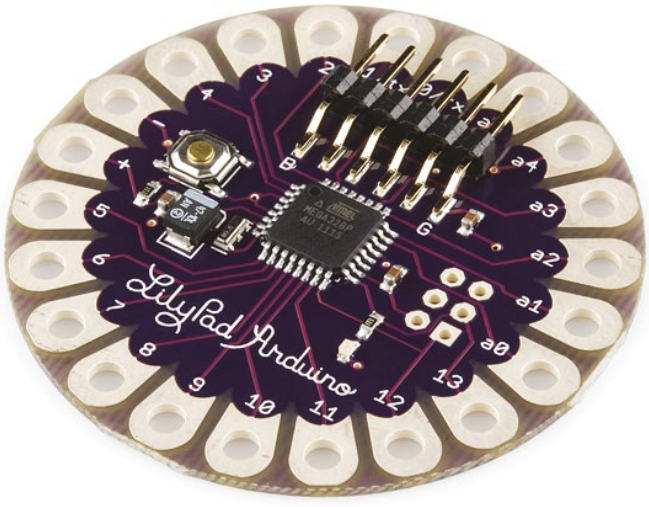
à sa réputation, ses efforts et son travail. C'est la raison pour laquelle le navigateur, dans la distribution Debian GNU/Linux, s'appelle Iceweasel.

Dans le cas d'Arduino, les choses se sont un peu envenimées ces derniers mois. Originellement, Massimo Banzi a initié le projet Arduino avec David Cuartielles, David Mellis, Tom Igoe et Gianluca Martino pour former en 2009 la société Arduino LLC et travailler sur le design des cartes, le logiciel et le support communautaire. La fabrication des cartes quant à elle a été détachée et confiée à une société tierce, Smart Projects Srl, créée par Gianluca Martino.

Fin 2014 début 2015 cependant, Gianluca Martino et Federico Musto ont décidé de renommer Smart Projects en Arduino Srl et de créer le site **arduino.org**. C'est ainsi qu'un conflit en justice oppose à présent Arduino LLC à Arduino Srl, que Arduino Srl propose une Arduino Zero Pro alors qu'il n'existe pas encore d'Arduino Zero (celle-ci est annoncée pour le 9 juin, mais sera très certainement déjà en rupture lorsque vous lirez ceci), que nous avons d'un côté un IDE Arduino 1.6.4 avec *Boards Manager* et de l'autre un 1.7.3 sans *Boards Manager*, mais avec un support Arduino Zero Pro ou encore que l'environnement Arduino 1.6.3 peut vous afficher un message « *This board comes from an uncertified manufacturer* » (« cette carte provient d'un fabricant non certifié »)...

Vous vous en doutez, ce genre de scission peut poser un problème, incitant d'une manière ou d'une autre les utilisateurs et les partenaires commerciaux à prendre parti pour l'un ou l'autre camp. En ce qui me concerne, je trouve que ce genre de choses n'a rien de bien étonnant, c'est arrivé à d'autres par le passé et arrivera à nouveau à l'avenir. Aujourd'hui, Arduino LLC et Arduino Srl revendiquent tous deux les droits sur la marque « Arduino » et se réclament être le « vrai Arduino ». Dernièrement, Massimo Banzi a d'ailleurs précisé, lors de sa conférence à la 10ème Maker Faire de San Francisco le 16 mai dernier, qu'une « marque sœur » avait été déposée mondialement, « Genuino ».

En tant qu'utilisateur les fonctionnalités forment le critère de sélection absolu et en cela, l'IDE Arduino 1.6.4 apporte des nouveautés intéressantes que la version 1.7.3 ne possède pas (ou pas encore). Exactement comme dans le cas d'un *fork* d'un projet open source, lorsque les tensions



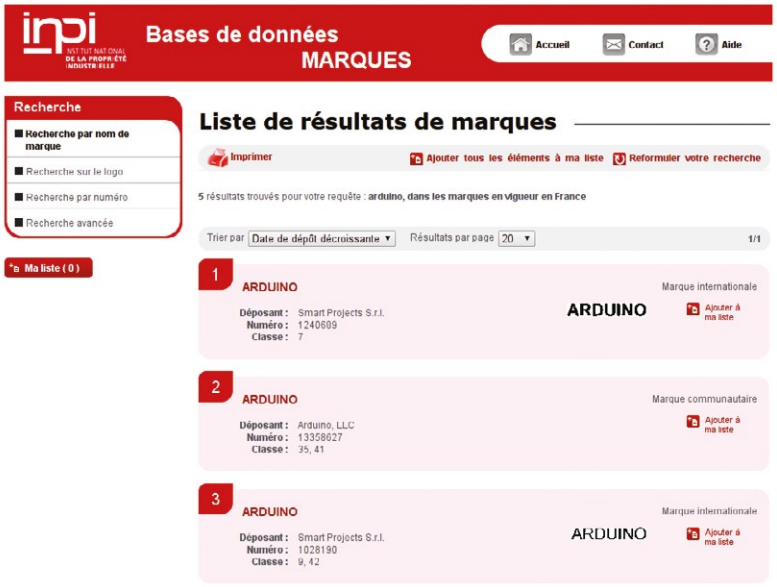
La carte LilyPad Arduino est conçue et produite par SparkFun Electronics, mais est listée sur le site Arduino.cc tout comme la Gemma d'Adafruit Industries. Ces cartes sont produites sous licence et des droits sont reversés à Arduino LLC pour l'utilisation de la marque, contrairement aux « clones » Arduino qui peuvent être parfaitement similaires mais, doivent être désignés différemment (Seeeduino, SunDuino, Freeduino, Freaduino, etc.).

coromonadalix

internes et les décisions prises sont davantage préjudiciables à l'ensemble que la division des équipes de développement. Les utilisateurs et les contributeurs vont alors naturellement au projet le plus ouvert, le plus dynamique et le plus prompt à intégrer des fonctionnalités intéressantes.

C'est du point de vue des partenaires commerciaux que les choses s'avèrent délicates. SparkFun par exemple précise sur son site qu'elle continuera de vendre des Arduino Uno fabriquées par Arduino Srl, mais que ceci ne l'empêchera pas de continuer de produire les cartes Arduino Pro, Pro Mini, Fio, et LilyPad en reversant des droits à Arduino LLC, et peut-être même à l'avenir les cartes Zero, Uno, Mega et Due.

On parle ici de marché, de partenaires, de marque et de droit, et plus exactement d'un partenariat entre plusieurs personnes qui touche à sa fin de façon mouvementée. La vérité c'est que vous ne saurez jamais intimement pourquoi et comment ces personnes sont devenues partenaires et donc pas davantage pourquoi et comment elles en arrivent à la confrontation. Un débouché légal destituant Arduino LLC de la marque pourrait changer la donne, mais pas vraiment pour l'utilisateur. Il s'agit ici surtout de business, car au final celui qui possède la marque décide automatiquement qui sont ceux qui peuvent fabriquer et vendre les cartes sous ce nom, et donc profiter de l'aura de la dénomination commerciale « Arduino » pour la convertir en chiffre d'affaires.

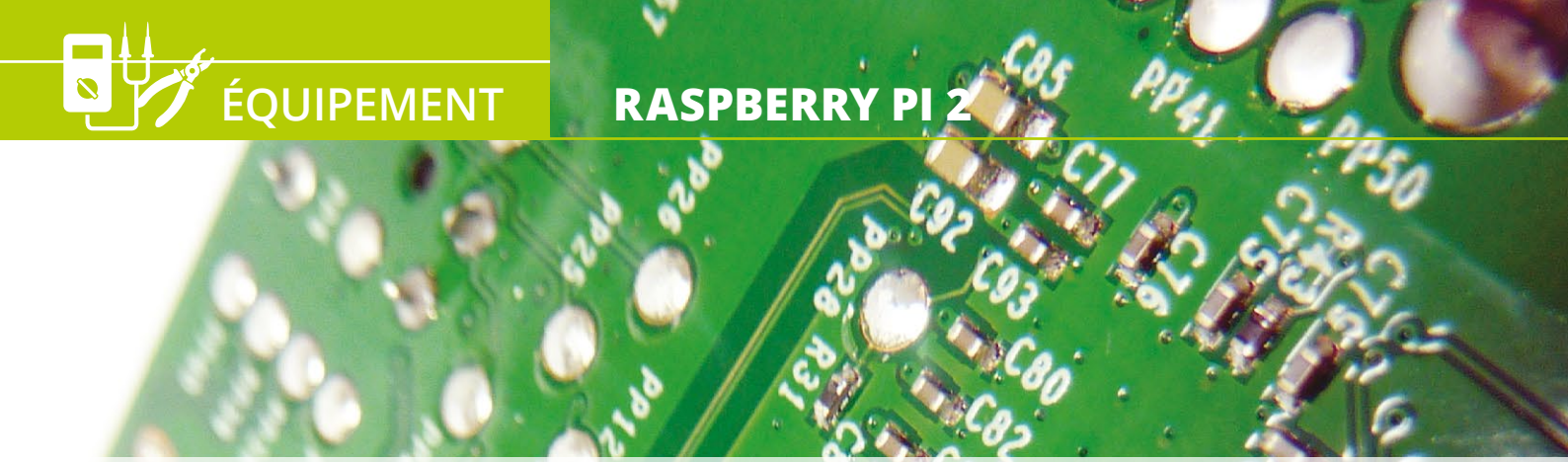


Le service de recherche des marques de l'INPI présente effectivement la marque internationale « Arduino » déposée à la fois par Arduino Srl et Arduino LLC, mais dans des classes différentes. 7 (Imprimantes 3D), 9 (Périphériques d'ordinateur, microcontrôleurs autonomes...) et 42 (Services scientifiques et technologiques) pour le premier et 35 (Services de vente au détail liés à l'électronique, au matériel informatique, aux circuits imprimés nus, aux microcontrôleurs, aux kits électroniques...) et 41 (Services d'enseignement) pour le second. On notera que le dépôt de Arduino Srl date de 2009 et celui de Arduino LLC de 2014. Ce dernier affiche comme statut « Opposition à l'encontre de la demande »...

La reconnaissance du travail accompli pour créer l'environnement, la plateforme et l'écosystème ne peut se faire par la simple attribution d'une marque commerciale, et cette attribution ne prouvera rien du tout. La paternité de ce qu'on nomme aujourd'hui « Arduino » revient à un ensemble de personnes bien plus large que celles qui ont conçu les cartes, en ont fait la promotion ou les ont produites et commercialisées.

À mon sens, il est important de garder les considérations financières et légales bien distinctes du mérite communautaire de chacun. Dans le cas contraire, ceci reviendrait à oublier que ce qui était un projet d'étudiants en 2005 n'a été possible que parce que d'autres personnes avaient préalablement investi énormément d'énergie. Je pense principalement à GCC et à AVRdude qui existaient bien avant qu'on parle d'Arduino et sont encore aujourd'hui deux éléments qui forment la très grande majorité de l'environnement de développement.

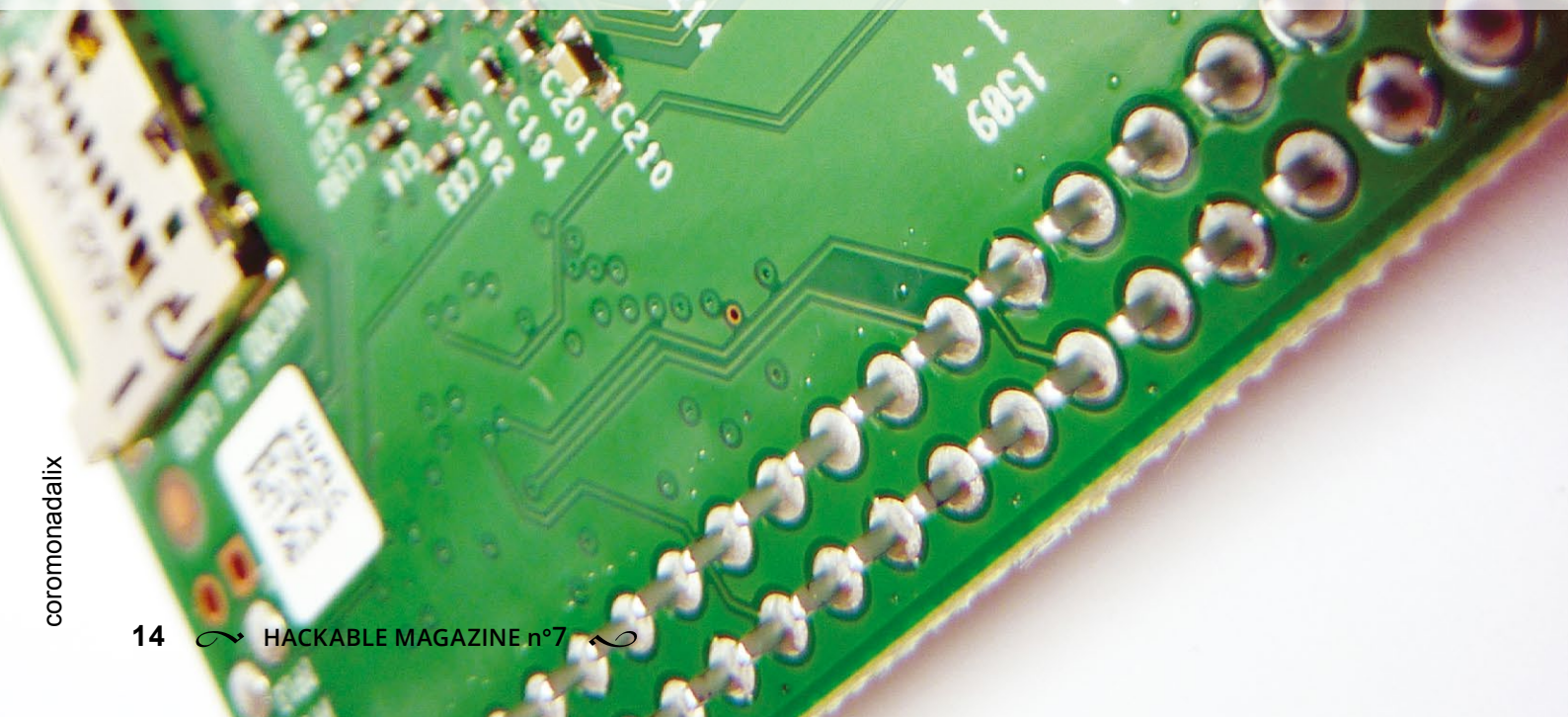
Les utilisateurs en gagnant en expérience le savent, ce n'est pas le nom qui fait la différence, mais les fonctionnalités et la qualité effective. Le nouveau venu en revanche n'aura aucun critère de comparaison et ne pourra reposer que sur une réputation et une marque. Nous n'avons pas ici affaire à un fork. Le conflit qui oppose Arduino LLC et Arduino Srl est un litige entre deux sociétés commerciales concernant l'attribution d'une marque, rien de plus. Deux compagnies initialement partenaires qui maintenant se disputent le marché. Mieux vaut éviter d'y voir, ou d'en faire, le combat du juste et du vertueux contre les forces du mal, car le marché, c'est vous et c'est moi. Plus exactement, c'est la probabilité que nous utilisions notre carte bancaire ou notre porte-monnaie pour celui qui interdira à d'autres d'utiliser le mot « Arduino » sur leurs produits... **DB**



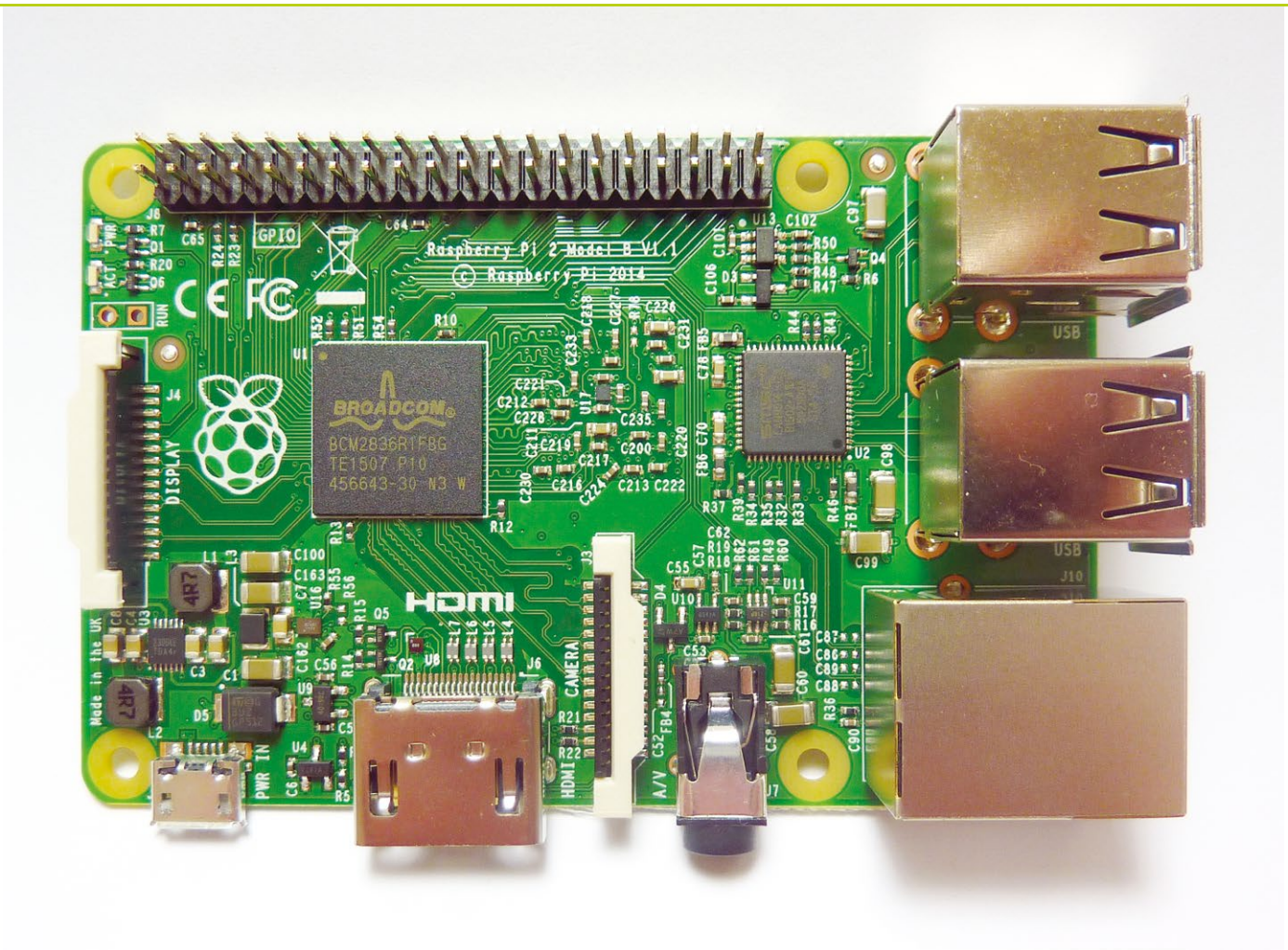
DOIS-JE ACHETER UNE RASPBERRY PI 2 ?

Denis Bodor

C'est un peu la même histoire pour tous les équipements ou appareils auxquels on s'attache. J'utilise le modèle actuel, il me convient, mais voici qu'une nouvelle version est mise en vente. Implacablement, la question qui se pose est : dois-je craquer ? Ça marche pour les smartphones, les ordinateurs portables, les équipements audio… et bien entendu aussi pour les cartes et nano-ordinateurs. Depuis peu est disponible une nouvelle version de la Raspberry Pi. La question est donc : est-ce que ça vaut le coup ?



coromonadalix



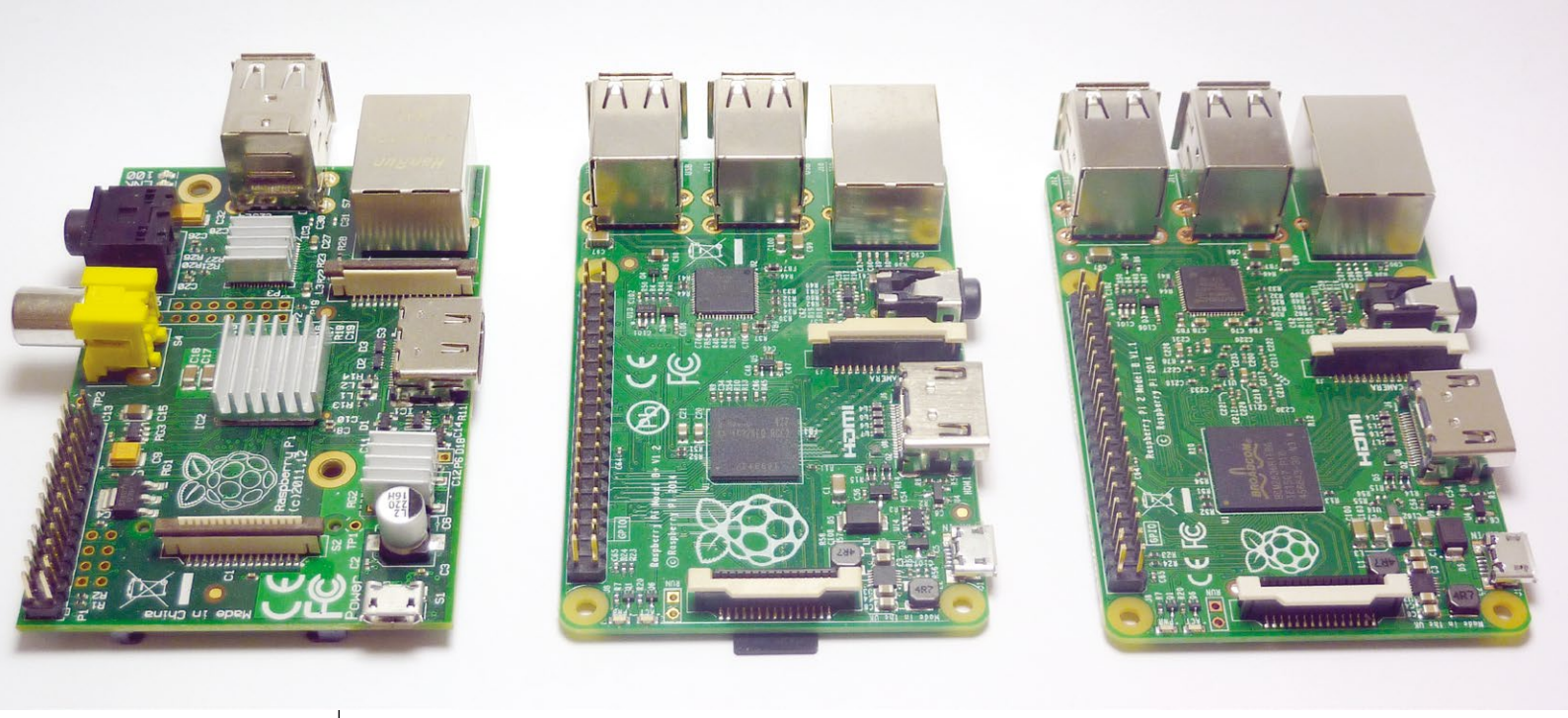
Au premier coup d'œil, on a bien du mal à reconnaître une Pi 2 d'une B+. La majorité du circuit est inchangée et seuls quelques composants ont été réagencés pour respecter les contraintes de fabrication et celles liées à l'augmentation de fréquence et de vitesse.

Le syndrome est toujours le même, tout va bien dans le meilleur des mondes et voici une nouveauté qui titille votre envie de posséder et d'utiliser la dernière génération disponible. Ça marche à tous les coups, sauf pour les films où généralement la suite est forcément moins bien que le premier. Souvent parce que nous ne ressentons plus le plaisir de la découverte et de la surprise, et parfois parce que c'est juste n'importe quoi (Highlander 2, Avengers 2, Une nuit en enfer 2, etc.).

Dans le cas de matériels et de technologies, si on élimine l'aspect purement ostentatoire du type smartphones à 800€ qui va avec la montre à 700€, la tablette à 500€ et le laptop doré à 1500€ (budget frime à 3500€, ouch !), les « suites » sont principalement des montées en puissance. Et c'est exactement le cas pour le nano-ordinateur Raspberry Pi qui a débarqué en version 2 en février dernier.

Au menu donc rien de radicalement nouveau concernant les fonctionnalités, mais simplement une augmentation massive de la puissance et des ressources :

- le processeur (ou plus exactement le SoC) Broadcom BCM2835 est remplacé par un BCM2836 reposant sur une architecture similaire, mais différente. On passe d'un ARM11 (ARMv6) à 700 MHz à un ARM Cortex-A7 (ARMv7) 900 MHz à quatre cœurs ;
- la mémoire passe de 512 Mo à 1 Go permettant ainsi d'attribuer le maximum de mémoire au processeur graphique tout en en ayant encore suffisamment pour le reste du système.



Trois générations de Raspberry Pi avec à gauche le modèle B et son branchement vidéo composite RCA, au centre le modèle B+ avec son connecteur 40 broches, ses deux ports USB supplémentaires et son alimentation sur le côté, et à droite la nouvelle génération avec le BCM2836 4 cœurs.

Le prix annoncé initialement est le même que celui de la Raspberry Pi B+ et est toujours celui constaté actuellement (environ 40€). Au milieu du mois de mai cependant, le blog de la fondation Raspberry Pi annonçait une baisse de prix sur le modèle B+, le ramenant à 30€. Ainsi, aujourd'hui la nouvelle Raspberry Pi 2 modèle B est en pratique moins chère que le précédent modèle B+, mais simplement parce que ce modèle B+ coûte maintenant moins cher à produire, non parce que le prix de la Raspberry Pi 2 est arbitrairement supérieur.

Comme le précise le billet sur le blog officiel, si vous recherchez un nano-ordinateur avec une sortie vidéo, une connectivité Ethernet et quelques ports USB sans nécessairement avoir besoin d'une puissance de calcul importante, le modèle B+ reste un très bon choix.

1. TOUT EST UNE QUESTION DE PERFORMANCE

En parlant de puissance de calcul justement, il est temps de vérifier exactement de quoi il est question en pratique. Il existe sous GNU/Linux un outil majoritairement utilisé pour tester les performances des trois principaux éléments de comparaison que sont la puissance du processeur, la vitesse des accès mémoire et les opérations de lecture/écriture des fichiers. Ce dernier point pour la Pi est totalement dépendant de la marque, du modèle et du type de carte microSD utilisée et ne fera pas l'objet de tests pour cet article.

Nous allons en revanche utiliser **sysbench** (installable avec le paquet du même nom) pour comparer les performances d'une Raspberry Pi modèle B+ avec la nouvelle Raspberry Pi 2 modèle B.

Commençons donc par un premier test consistant à trouver des nombres premiers avec une limite fixée par l'utilisateur. Voici ce qui se passe sur la B+ :

coromonadalix

```
% sysbench --test=cpu --cpu-max-prime=2000 run
[...]
Test execution summary:
    total time:                          54.4539s
    total number of events:              10000
    total time taken by event execution: 54.4171
    per-request statistics:
         min:                                  5.35ms
         avg:                                  5.44ms
         max:                                 15.84ms
         approx.  95 percentile:               5.49ms

Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   54.4171/0.00
```

La commande cherche les premiers jusqu'à 2000 tout en chronométrant les diverses opérations et en affichant ensuite un résumé. L'exécution complète aura duré quelques 54 secondes. À titre de comparaison, la même commande exécutée sur des machines de type PC affiche les résultats suivants :

- Intel Core2 Q6600 = 2,4259 s ;
- Intel Xeon E5520 = 1,6768 s ;
- Intel Core i7 = 0,9764 s ;
- Intel Celeron N2830 = 3,4586 s.

Précisons que cette dernière machine est un Intel NUC, un mini PC sans ventilateur orienté *desktop* faible consommation (pas un monstre de calcul donc). Si nous faisons de même avec la Raspberry Pi 2, nous obtenons :

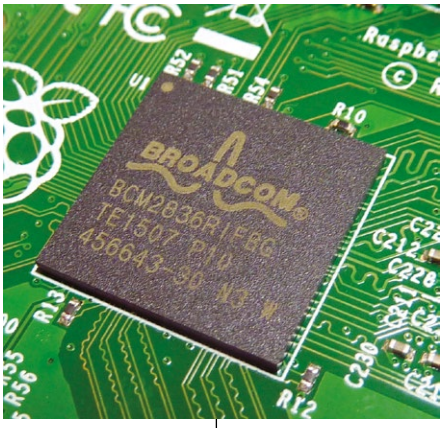
```
Test execution summary:
    total time:                          47.4590s
    total number of events:              10000
    total time taken by event execution: 47.4368
```

Voilà qui ne fait pas une grande différence, non ? Une telle conclusion serait oublier que nous avons là 4 cœurs et non un seul. Il est donc possible de faire fonctionner 4 *threads* (files d'exécution) en même temps pour ce test. En relançant la commande avec avec l'option **`--num-threads=4`**, le résultat est bien différent :

```
% sysbench --test=cpu --num-threads=4 \
--cpu-max-prime=2000 run
[...]
Test execution summary:
    total time:                          11.9082s
    total number of events:              10000
    total time taken by event execution: 47.5850
```

Nous avons littéralement divisé le temps d'exécution par 5 par rapport à la B+. En effet, la fréquence du processeur (900 contre 700 Mhz) ne fait pas tout, le nombre de cœurs est capital. Tout comme le fait de s'en servir. Bien entendu, utiliser cette option sur la B+ n'apporte rien :

```
Test execution summary:
    total time:                          54.3812s
    total number of events:              10000
    total time taken by event execution: 217.3081
```



Le changement important concernant la Raspberry Pi 2 concerne cette puce : le SoC Broadcom BCM2836. On parle de SoC pour « System On Chip » et non de processeurs, car il intègre également des périphériques, bus, contrôleurs, etc. Ce nouveau SoC intègre un processeur ARM Cortex-A7 900 Mhz à 4 cœurs.

Ceci est parfaitement normal, car même si on demande à l'outil d'utiliser 4 threads, ceux-ci ne peuvent pas être matériellement exécutés en même temps. Le résultat est donc assez similaire à l'utilisation d'un seul et unique thread.

2. ET LA MÉMOIRE ?

sysbench permet également de tester la vitesse d'accès à la mémoire. Avec la Raspberry Pi 2, nous en avons deux fois plus, ce qui est déjà un avantage certain, mais qu'en est-il de la rapidité ?

Commençons donc par la B+ :

```
% sysbench --test=memory --memory-block-size=1M \
--memory-total-size=2G run
[...]
Operations performed: 2048 (  360.77 ops/sec)
2048.00 MB transferred (360.77 MB/sec)

Test execution summary:
    total time:                          5.6767s
    total number of events:              2048
    total time taken by event execution: 5.6682
[...]
```

Cette commande va mesurer l'accès à 2 Go de mémoire par bloc de 1 Mo et nous informe du résultat. Un peu plus de 5 secondes et demi ont suffi pour ces 2 Go. Relativisons encore une fois en précisant les valeurs sur architecture PC :

- Intel Core2 Q6600 = 0,2990 s ;
- Intel Xeon E5520 = 0,4332 s ;
- Intel Core i7 = 0,3237 s ;
- Intel Celeron N2830 = 0,5555 s.

La comparaison est injuste, car on ne parle tout simplement pas des mêmes mondes. La consommation d'énergie et le coût seuls suffisent à discréditer cette mesure arbitraire. Ces chiffres ne sont là que pour relativiser nos mesures.

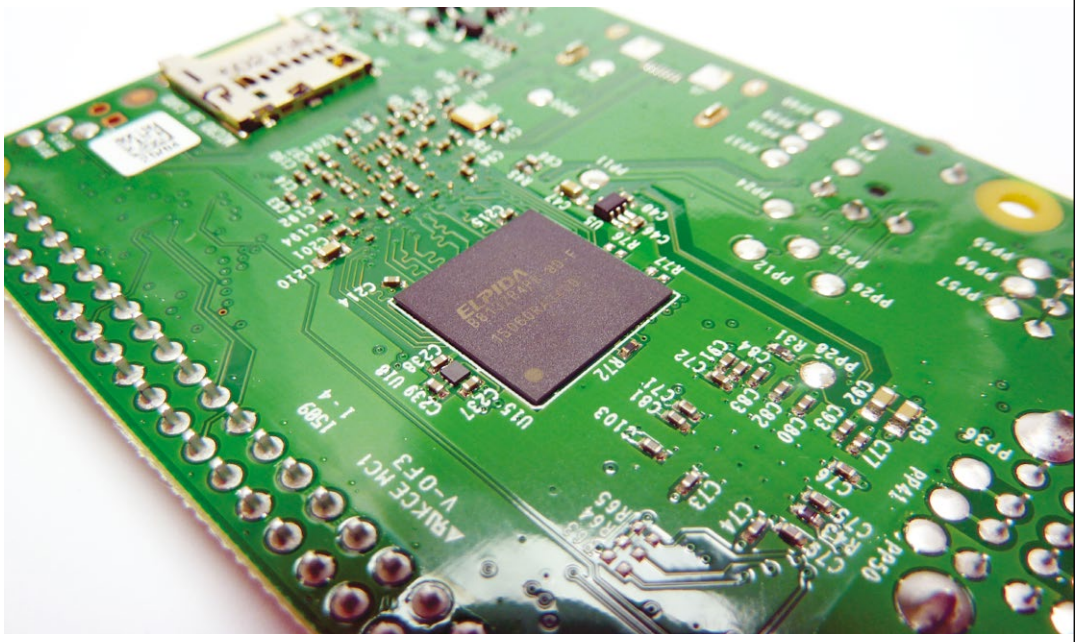
Testons à présent la commande sur la Raspberry Pi 2 :

```
Operations performed: 2048 (  754.96 ops/sec)
2048.00 MB transferred (754.96 MB/sec)

Test execution summary:
    total time:                          2.7127s
    total number of events:              2048
    total time taken by event execution: 2.7071
```

et avec `--num-threads=4` :

```
    total time:                          2.0838s
    total number of events:              2048
    total time taken by event execution: 8.3057
```

L'accès est clairement plus rapide, mais la différence n'est pas aussi flagrante qu'avec les ressources processeur. Mais les mesures de performance concernant la mémoire sont parfois trompeuses. En fonction de la manière dont on accède à cette dernière, les chiffres peuvent changer du tout au tout.



Sur la Pi modèle B, la mémoire était soudée sous le SoC, avec la B+ c'était au-dessus du SoC et avec la Raspberry Pi 2, elle est sous la carte. Remarquez les pistes du circuit qui serpentent apparemment sans raison. Ceci permet de s'assurer que les signaux soient transmis en un temps connu, déterminé par longueur de la piste.

En supprimant l'option `--memory-block-size=1M`, on utilisera pour le test des blocs de la taille par défaut, soit 1 Ko. Là les choses sont bien différentes, car ceci nécessite davantage de travail de la part du processeur. Sur la B+ tout d'abord :

```
Operations performed: 2097152 (102772.74 ops/sec)
2048.00 MB transferred (100.36 MB/sec)

Test execution summary:
    total time:                          20.4057s
    total number of events:              2097152
    total time taken by event execution: 16.6765
```

La durée de l'opération est plus importante. Ces 20 longues secondes découlent directement de l'exécution de 2 millions d'opérations pour arriver aux 2 Go transférés. Un test identique sur la Raspberry Pi 2 nous affiche des performances juste un peu différentes :

```
    total time:                          17.1518s
    total number of events:              2097152
    total time taken by event execution: 13.2057
```

17 secondes, ce n'est pas un gain phénoménal. Mais nous avons à nouveau omis d'utiliser la plateforme à sa pleine puissance. N'oubliez pas, nous avons maintenant 4 cœurs. Si nous ajoutons `--num-threads=4` :

```
    total time:                          4.4945s
    total number of events:              2097152
    total time taken by event execution: 13.6896
```

C'est presque 5 fois plus rapide qu'avec la B+ ! Dès que l'intervention du processeur est nécessaire et qu'on utilise judicieusement les ressources, les performances sont drastiquement meilleures. L'un dans l'autre, on arrive effectivement à ce qui est annoncé un peu partout, soit une carte environ 6 fois plus rapide que la B+.

3. CE QU'IL FAUT EN RETENIR

Résumer ceci en affirmant « *la Raspberry Pi 2 est 6 fois plus rapide, achetez-là* » serait stupide et malvenu. La réponse est plus complexe, et plus courte : « ça dépend ».

Deux choses sont à prendre en compte avant de dépenser vos 40€. La première est le fait qu'en termes de fonctionnalités les différences sont quasi inexistantes. Si vous n'utilisez pas d'applications gourmandes ou n'avez pas de projets nécessitant une forte puissance de calcul,

Modèle B+ (gauche) et Raspberry Pi 2 côte à côte. La nouvelle carte est ce qu'on peut appeler « a drop-in replacement » en anglais, une carte qu'il suffit de mettre à la place de l'ancienne. Ceci est valable mécaniquement, électriquement et de manière logicielle. Si votre système est à jour, il suffit de sortir la microSD de la B+ pour la glisser dans la nouvelle carte.

coromonadalix

autant économiser 10€ en optant pour le modèle B+. Ces 10€ pourront être dépensés pour une clé USB Wifi par exemple ou une microSD d'une taille un peu plus importante.

L'autre point à prendre en compte en ce qui concerne la puissance de calcul est la nécessité d'utiliser des applications capables de diviser leur travail sur différents cœurs. On parle généralement d'applications *multithreadées* par opposition aux processus mono-thread. Bien sûr, l'ensemble du système bénéficiera de la présence des 4 cœurs puisque le noyau se charge de répartir l'exécution des programmes. Mais si vous avez une application nécessitant des ressources processeurs et que celle-ci n'est pas *multithreadée* le gain sera insignifiant. Il en va de même pour les accès mémoires qui ne sont finalement boostés qu'en cas de multithreading et avec des petits blocs de données.

Certaines applications l'utilisent, d'autres non. Pire encore, certaines solutions ne supportent pas le multithreading par défaut, mais permettent d'utiliser des éléments qui en déclenchent l'usage. Un excellent exemple est le système de vision par ordinateur OpenCV, très gourmand en calcul. Il faudra utiliser la bibliothèque TTB pour tirer le maximum du nouveau processeur de la Raspberry Pi 2. Si le développeur ne le fait pas, la différence de performances sera minime.

Donc, « ça dépend ». Ça dépend de ce que vous avez à faire, de comment vous le faite et avec quoi. Si vous programmez, en Python par exemple, il faudra également envisager de gérer les *threads* dans vos scripts. Le module `thread` fait cela très bien, mais encore faut-il structurer votre programme de manière à vous en servir judicieusement. C'est un tout nouveau monde qui s'ouvre à vous. À vous de voir si vous préférez y voir une épreuve ou l'occasion d'apprendre quelque chose de nouveau...

4. ANECDOTE ET LEÇON DE PHYSIQUE

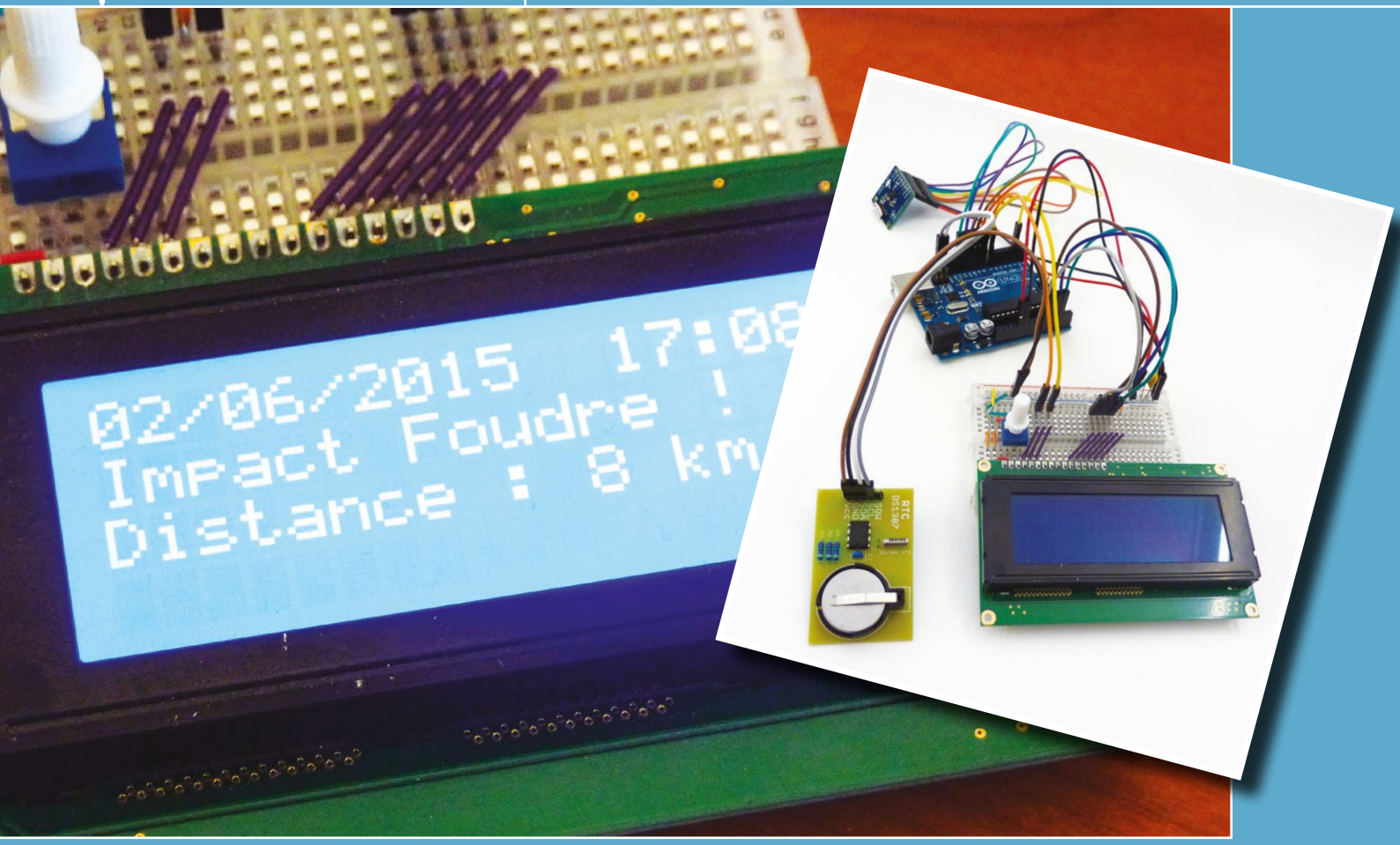
Peu après la mise en vente de la Raspberry Pi 2, un utilisateur sur le forum officiel a fait part d'une découverte intéressante : la carte n'aime pas être prise en photo et s'éteint toute seule. Ce n'est, bien entendu pas une question de timidité (puisque la carte a l'habitude d'être nue en public), mais un problème de flash et plus précisément de flash au xénon.

Le composant en cause est celui marqué U16 faisant partie du circuit de régulation de courant, placé entre le connecteur HDMI et le connecteur plat « display ». Il est facilement reconnaissable, car il est visuellement très différent des autres en raison de son aspect brillant métallisé. Et c'est justement là une part du problème.

La puce U16 utilise en effet une technologie appelée *Wafer Level Chip Scale Packages* (WL-CSP) où la puce de silicium est directement soudée à l'envers au circuit, sans boîtier plastique pour la protéger. En utilisant un appareil photo avec un flash au xénon (cela ne fonctionne pas avec les flash à led des smartphones), un effet photoélectrique apparaît. Les photons avec une énergie suffisante, lorsqu'ils rencontrent la surface en silicium, provoquent l'émission d'électrons. C'est le principe de fonctionnement des cellules photoélectriques, mais ici, les électrons générés perturbent le fonctionnement de la puce qui provoque une chute de tension du régulateur, qui éteint la carte.

Le problème n'a rien de nouveau ou d'exceptionnel et est connu de longue date avec ce genre de composants WL-CSP. Ceci n'endommage pas la carte et ne se produit que dans des conditions bien particulières. La nouvelle Raspberry Pi n'est pas susceptible de s'éteindre spontanément lorsqu'elle est simplement exposée à la lumière (soleil, éclairage, flash led, etc.). Cependant, selon l'environnement où vous placerez votre projet, il faudra prévoir un boîtier de protection ou du moins couvrir la puce avec un matériau opaque. **DB**

coromonadalix



CRÉEZ UN DÉTECTEUR D'IMPACT DE FOUDRE

Denis Bodor

Les orages et la foudre ont toujours eu, dans l'histoire de l'humanité, un impact très fort (c'est le cas de le dire). En dehors des catastrophes naturelles (tempêtes, séismes, tsunamis, etc.), c'est sans doute le rappel le plus fréquent que nous fait la nature à propos de l'humilité qui devrait être la nôtre en tant qu'espèce « dominante » sur la planète. La foudre est une manifestation électrique qu'il n'est généralement pas très intelligent d'ignorer ou de sous-estimer. Que peut faire une carte Arduino à ce propos pour nous aider ?

NIVEAU	TEMPS	BUDGET
	20 minutes	environ 60 € (Arduino inclus)

Le principal danger lié aux orages, en dehors des précipitations et des vents qui les accompagnent, concerne le risque de foudroiement. Les randonneurs le savent bien, un certain nombre de règles doivent être respectées dès lors qu'un orage se prépare, en particulier dans la montagne où le temps peut changer très rapidement et où il n'y a pas grand-chose dans l'environnement pour se protéger.

En ce qui nous concerne, bien qu'une version portable d'un montage puisse parfaitement être envisagée, nous serons ici davantage intéressés par les risques « urbains » et donc concernant principalement les installations techniques. De manière générale, un orage dans une agglomération présente peu de risque pour les personnes, car on est rarement en extérieur à ce moment-là, et un bâtiment tout comme un véhicule offrent une protection bien suffisante. Il n'en va pas de même pour un certain nombre d'équipements, à commencer par tout ce qui se trouve en hauteur et est susceptible de fournir un conducteur idéal pour la foudre. Les antennes arrivent donc typiquement parmi les points d'impact les plus probables en l'absence de parafoudre.

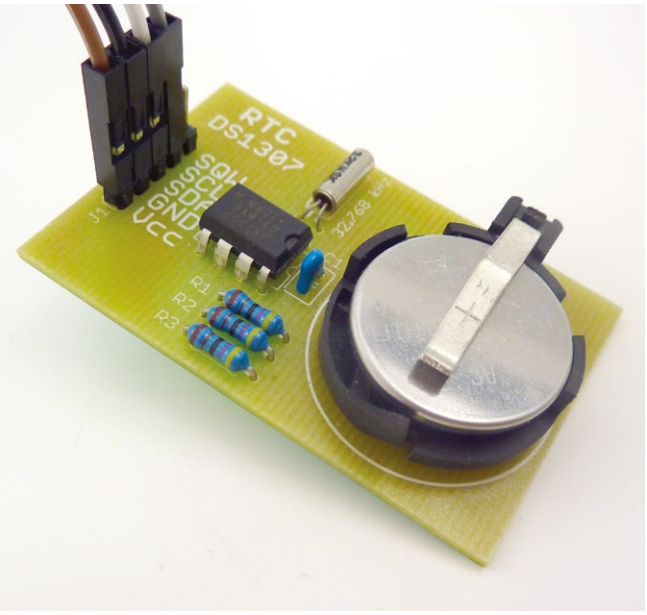
D'autres raisons, bien entendu, peuvent justifier le fait de vouloir détecter les impacts de foudre distants. Un tel dispositif peut venir compléter une installation météorologique amateur par exemple, s'ajouter à un réseau de capteurs (UV, ensoleillement, vibrations, températures, radiations, consommation d'énergie, etc.) remontant des informations sur un site de collecte de données ou encore simplement assouvir une certaine curiosité scientifique.

LE PRINCIPE

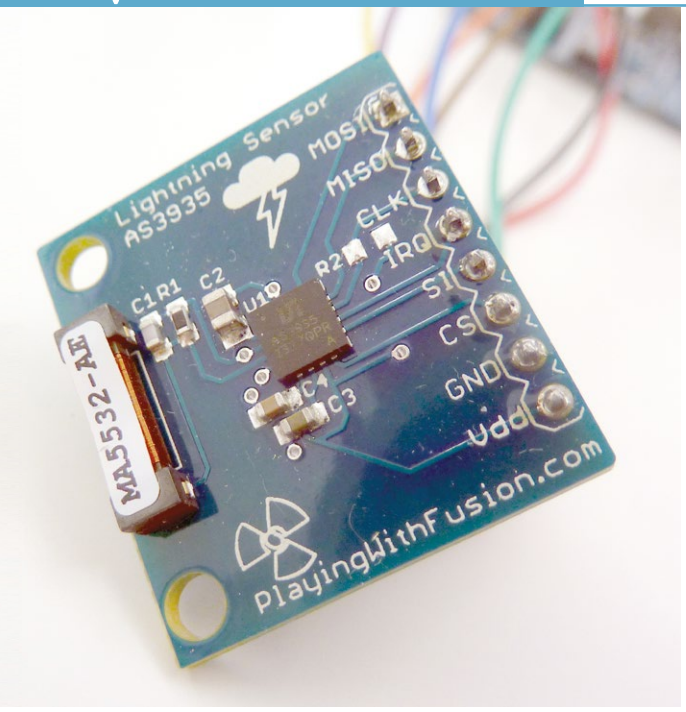
Il existe bien des équipements permettant de détecter les impacts de foudre (au sol ou entre nuages). Certains sont imposants et protègent des installations sensibles comme des centrales électriques, d'autres sont embarqués dans des véhicules et en particulier les aéronefs et enfin, il existe les produits portables pour les randonneurs, de la taille d'un mobile (les anciens, pas les tablettes géantes qui font téléphone).

Tous reposent sur un principe relativement simple si l'on sait qu'un éclair n'est pas une décharge unique, mais une série de 3 à 10 décharges. La décharge initiale du nuage vers le sol créé un canal ionisé qui facilite le passage d'un courant inverse du sol vers les nuages. C'est généralement cette série de décharges, espacées de quelques dizaines de millisecondes qui est perçue comme étant un éclair. Ce n'est toutefois pas une constante absolue puisqu'il peut y avoir des charges positives comme négatives s'accumulant dans les nuages qui provoquent des éclairs (tout dépend de quel endroit du nuage la décharge apparaît). Les règles exactes dictant la formation de la foudre sont encore un domaine largement inexploré, d'autant plus que la majorité des décharges a lieu hors de notre vue, en haute atmosphère.

Quoi qu'il en soit, l'énergie en œuvre est colossale. On parle ici de dizaines de millions de volts et jusqu'à 300000 ampères, pour un total d'énergie transféré pouvant atteindre des centaines de mégajoules. La nature particulière des décharges et la quantité d'énergie développée rendent le phénomène détectable sous forme d'interférences électromagnétiques à des kilomètres de la source. C'est sur cette caractéristique que se basent la plupart des détecteurs.

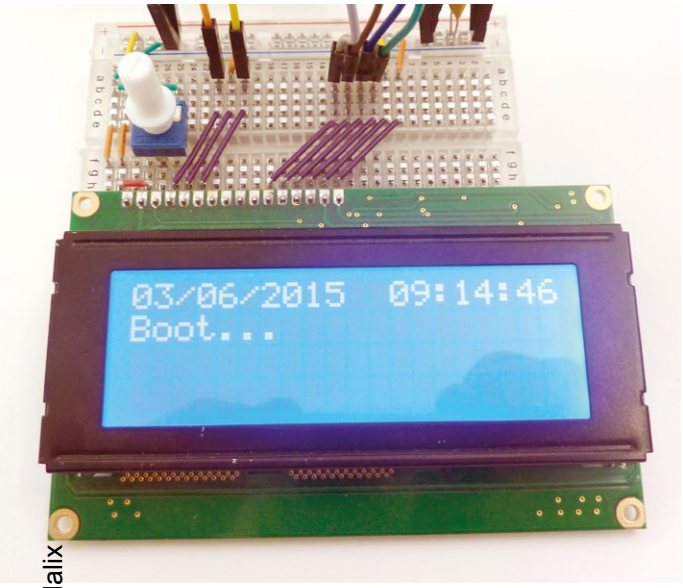


Un module RTC comme celui-ci, utilisant un composant DS1307 de Maxim, est un périphérique courant et peu coûteux qui complétera merveilleusement n'importe quel projet ayant besoin d'une référence de temps. Notez que le module intègre une pile bouton CR2032 permettant à l'horloge de fonctionner de façon autonome en l'absence d'alimentation.



Le module de détection de chez Playing With Fusion Inc avec en son centre le circuit intégré AMS AS3935 et sur la gauche l'antenne miniature.

N'importe quel afficheur compatible HD44780 fera l'affaire pour notre projet. Les modèles les plus courants sont sans le moindre doute les 2×16, mais disposer de 4 lignes de 20 caractères est un confort très appréciable. Notez qu'il existe également des modèles 4×40 utilisant deux contrôleurs et possédant deux broches de validation des données, une par contrôleur pilotant chacun une moitié de l'écran. On peut aussi envisager l'utilisation d'un écran graphique TFT (cf Hackable n°4) ou un afficheur à leds.



Il s'agit donc, dans les grandes lignes, de récepteurs basses fréquences (de l'ordre de 300 Khz), équipés de filtres et d'amplificateurs permettant de fournir une impulsion en sortie dès lors qu'une détection a lieu. Il existe une grande quantité de circuits de détection de foudre et c'est là un exercice très pédagogique dès lors qu'on souhaite toucher à l'électronique analogique. Quelques transistors, diodes, bobines, résistances et condensateurs, et vous voici dans le monde merveilleux de la détection de foudre... et de tout un tas d'autres perturbations électromagnétiques telles que les interrupteurs, les briquets électroniques, les tubes fluorescents ou encore les systèmes d'allumage de moteurs thermiques. L'électronique analogique pure n'étant pas notre domaine d'activité principal, je laisse le lecteur curieux tourner son attention vers le site de Charles Wenzel (**http://www.techlib.com/**) présentant plusieurs versions d'un détecteur de foudre très accessible en termes de coût et de réalisation (pour l'aspect théorique électronique c'est autre chose).

Nous nous intéresserons plutôt ici à un détecteur contenu dans un circuit intégré, l'AS3935 d'AMS (AustriaMicroSystems). Celui-ci peut être interfacé en i2c ou en SPI, cette dernière configuration étant celle par défaut dans le cas du module que nous avons utilisé. L'AS3935 n'est, en effet, qu'un circuit intégré qu'il faut compléter d'une antenne et de quelques composants passifs. L'ensemble est vendu sous forme de module complet par *Playing With Fusion Inc* (référence SEN-39001) via leur boutique en ligne (**http://playingwithfusion.com**) ou sur eBay (cherchez « *AS3935 Digital Lightning Sensor Breakout* »). Il vous en coûtera environ 34€ (26€ + 8€ de port depuis les USA).

C'est certes relativement cher, mais à ce prix vous avez un module directement utilisable, avec antenne intégrée et surtout pleinement calibrée. À ce propos, si vous achetez le module, ne jetez surtout pas l'emballage ! Une étiquette précise la valeur de calibration qu'il faut utiliser dans le croquis Arduino et celle-ci est spécifique à chaque module. Si nous ajoutons à cela que l'AS3935 est un circuit spécialisé capable de faire la différence entre un impact réel et une perturbation grâce à un algorithme fait main (dixit la documentation d'AMS, « *man-made disturber rejection algorithm* »), tout en fournissant une estimation de la distance, vous comprendrez que nous sommes dans une tout autre catégorie que le circuit analogique fait maison. Enfin, le module est capable de fonctionner avec une alimentation entre 2,4 et 5,5 volts avec un courant consommé de l'ordre de quelques centaines de microampères (oui, µA pas mA).

L'objectif du projet est de créer, sur la base de l'AS3935, un système d'information complet. Nous utiliserons donc un système d'affichage signalant un impact de foudre accompagné de la date et l'heure de l'événement ainsi que l'estimation de la distance de l'impact.

coromonadalix

CE QU'IL VOUS FAUT

- **Une carte Arduino : n'importe quelle carte Arduino, clone ou similaire pourra ici être utilisée. Notez cependant que nous partons sur un montage très complet avec afficheur et référence de temps. Le module de détection de foudre est compatible 5, et 3,3V, ce qui n'est pas nécessairement le cas des autres composants. Si vous utilisez une plateforme en 3,3 V non tolérante au 5 V (comme l'Arduino Due ou une carte TI Launchpad par exemple) vous devrez vous assurer que l'ensemble des composants/modules soient en 3,3V ou utiliser des convertisseurs de tensions. Dans le projet présenté ici et les essais qui l'accompagnent, c'est une classique Arduino Uno qui est utilisée.**

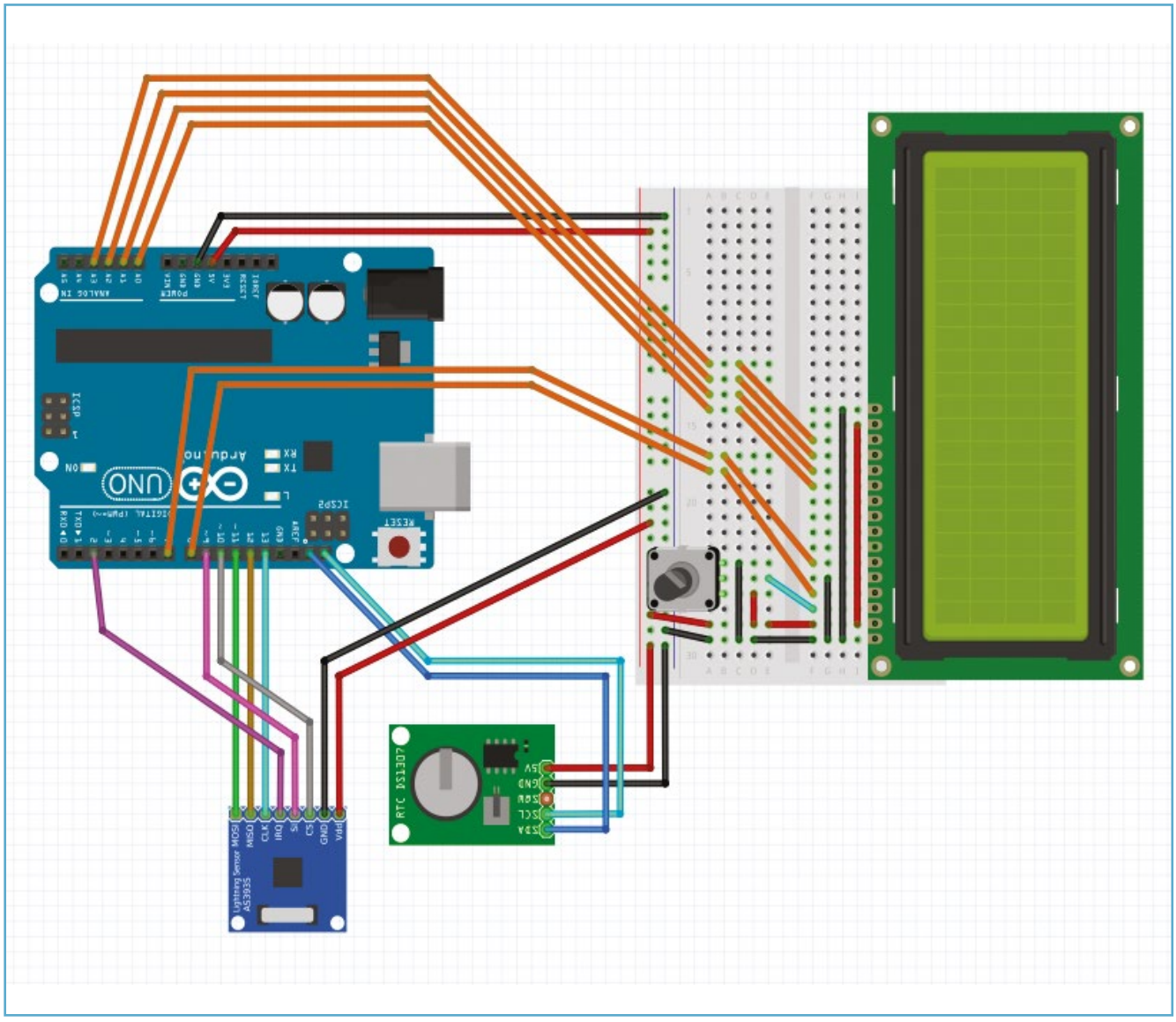
- **Une RTC DS1307 : nous souhaitons fixer les détections dans le temps, car il peut s'agir d'un orage ou simplement d'un événement isolé. L'utilisateur en regardant l'affichage doit savoir quand a eu lieu la dernière détection. Pour cela, nous avons besoin d'une base de temps et plus précisément une horloge temps réel (ou RTC pour *Real-Time Clock*). Ce type de module utilise un circuit intégré DS1307 interfacé en i2c et accompagné d'une pile bouton type CR2032. Attention, pour un montage en 3,3V, il faudra porter votre choix sur un module DS1338 (3V à 5,5V) et non un DS1307 (4,5 à 5,5V uniquement).**

- **Un afficheur LCD 4×20 : n'importe quel afficheur LCD compatible HD44780 fera l'affaire. Quatre lignes de 20 caractères permettent d'afficher à la fois la date et l'heure sur une ligne (« JJ/MM/AAAA HH:MM:SS »), le message concernant l'impact sur la seconde et le détail (distance) sur une troisième ligne. Un afficheur 2×16 pourra également faire l'affaire en jonglant un peu pour tout faire tenir en 32 caractères. Il est possible, selon le modèle d'afficheur LCD utilisé de le contrôler avec des signaux en 3,3 V tout en l'alimentant en 5 V. Ceci n'est pas vrai pour tous les afficheurs et il ne faut en aucun cas que celui-ci transmette des données (mettre impérativement la broche R/W à la masse). Dans le doute, optez pour un afficheur LCD 3,3 V. Ils sont rares, mais ils existent.**

- **Un module AMS AS3935 : le module conçu et vendu par *Playing With Fusion Inc* semble être le seul proposant un AS3935. Il ne semble étrangement exister aucun clone chinois peu cher pour cette solution. La seule autre option apparaît être le kit de développement d'AMS pour ce composant, vendu plus de 200 euros. La puce seule vous coûtera seulement 8€, mais n'est proposée que dans un format presque impossible à exploiter avec un équipement hobbyiste (MLPQ 16 broches).**

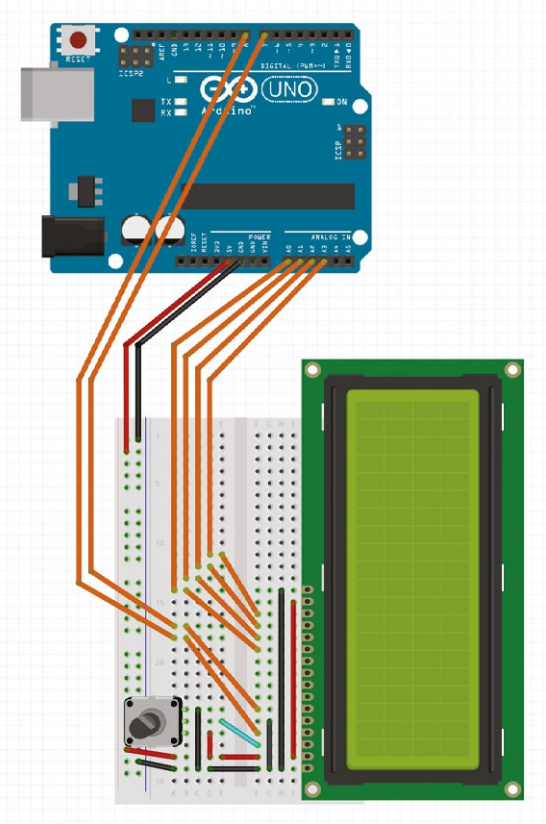
- **Une platine à essais et beaucoup de câbles : nous utiliserons une grande quantité des broches disponibles sur une carte Arduino courante comme la Uno et nous ferons usage de trois types d'interfaces : i2c pour la RTC, SPI pour le module AS3935 et un bus parallèle avec 4 bits de données pour l'afficheur LCD. Le montage proposé ici est présenté en version « didactique » ou « expérimentale ». L'idée est, par la suite, de pousser l'intégration plus avant et se débarrasser de la carte Arduino au profit d'un microcontrôleur Atmega328p (ou ATmega168) seul, soit sur plaque pastillée, soit avec un circuit imprimé.**

LE MONTAGE



Voici le montage complet du projet qui peut paraître impressionnant à première vue, mais finalement n'est que l'assemblage de trois éléments autour d'une carte Arduino Uno. Au bas de l'illustration, nous avons l'afficheur LCD 4×20 caractères sur platine à essais, sur la gauche en vert nous trouvons le module avec l'horloge temps réel DS1307 et sa pile bouton et à gauche en bleu nous avons le module AS3935 se chargeant de la détection. Pour comprendre l'interconnexion de l'ensemble, il suffit de prendre chaque élément séparément.

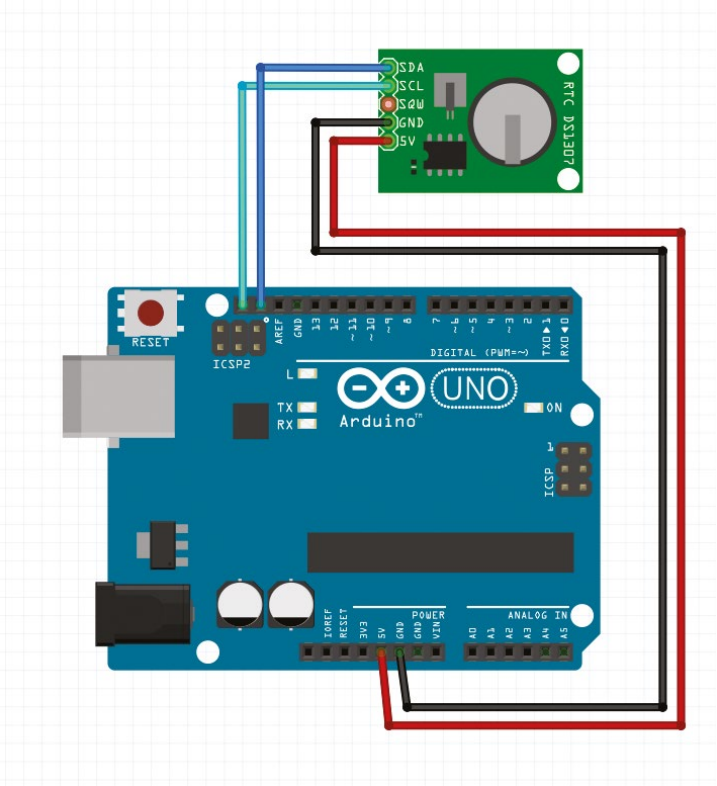
coromonadalix



1 L'afficheur LCD est ici connecté avec un minimum de lignes. Les afficheurs en mode texte HD44780 ou compatible utilisent presque tous le même brochage avec, de gauche à droite :

- GND : la masse ;
- Vcc : l'alimentation (généralement en +5V) ;
- Vo ou Vee : le contrôle du contraste qui se résume à l'application d'une tension sur cette broche comprise en 0 et Vcc. On utilise généralement un potentiomètre de 10 Kohms connecté à la fois à la masse et à Vcc et agissant comme un diviseur de tension. La résistance est de 10 Komhs entre les deux broches externes et celle au centre correspond à un point mobile contrôlé par la rotation du potentiomètre. De ce fait, en tournant le potentiomètre on fait varier la valeur des résistances d'un côté et de l'autre du point mobile et donc la tension ;
- RS : permet de préciser à l'afficheur si on lui envoie des commandes ou des données. Cette ligne est contrôlée par la broche 8 de la carte Arduino ;
- R/W : permet de spécifier si on écrit ou on lit des données. Ici cette broche est simplement mise à la masse pour spécifier une écriture sur l'afficheur. Cela nous empêche matériellement de lire des données, mais présente l'avantage de ne pas prendre le risque de laisser l'afficheur contrôler les lignes de données et donc d'utiliser une tension de 5V avec une carte Arduino (Due) ou autre, qui fonctionne en 3,3 V. Le fait de ne pas contrôler R/W implique qu'on ne peut pas, entre autres, vérifier si l'afficheur a fini de travailler. On laisse alors un délai arbitrairement choisi s'écouler entre les envois de données ou de commandes, l'affichage est donc sensiblement plus lent (mais on gagne une broche et c'est moins risqué) ;
- E : permet de valider l'envoi des données ou des commandes présentées sur les lignes qui suivent. Lorsque E est passé de l'état bas (0V) à l'état haut (5V) les données sont prises en compte. Cette ligne est contrôlée par la broche 7 de l'Arduino ;
- DB0 à DB3 : un afficheur HD44780 peut utiliser plusieurs types de liaisons ou largeurs de bus de données. Pour économiser des broches, nous n'utilisons pas les 8 lignes de données, mais uniquement 4. Comme pour R/W ceci ralentit l'affichage puisqu'il faut envoyer les 8 bits en deux fois, mais nous fait économiser 4 broches de l'Arduino ;
- DB4 à DB7 : en mode 4 fils on n'utilise que ces broches qui seront respectivement connectées, côté Arduino, aux broches A0 à A3, utilisées en sortie standard ;
- Anode et cathode led : c'est l'alimentation du rétro-éclairage souvent disponible sur les afficheurs LCD actuels. S'il s'agit d'un modèle de récupération un peu ancien et uniquement réflectif, ces broches sont tout simplement absentes. Il est également possible qu'il y ait 2 broches en plus, en particulier sur des afficheurs récents proposant un rétro-éclairage non pas avec une led (ou série de leds), mais trois : rouge, vert et bleu. On a alors une cathode (-) commune et des anodes (+) pour chaque couleur. On peut ainsi combiner les couleurs et pourquoi pas utiliser des sorties PWM pour régler leur intensité. Ne jouez pas à cela avec un afficheur avec un rétro-éclairage électro-luminescent, la PWM n'est pas utilisable et peut endommager le circuit d'alimentation haute tension (vécu).

coromonadalix



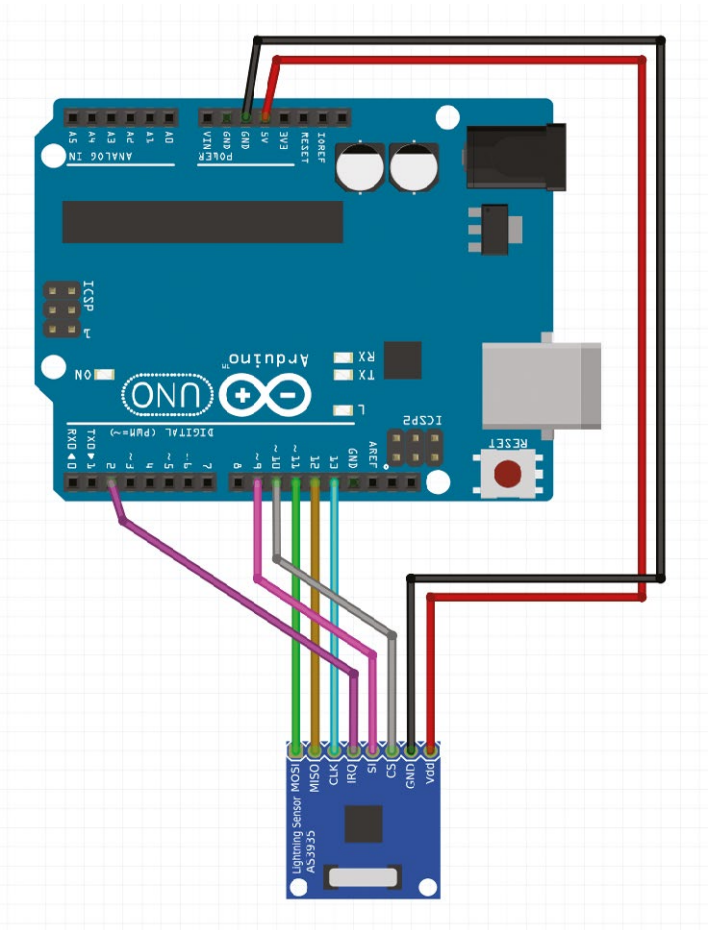
2 Une horloge temps réel comme la DS1307 de *Maxim Integrated* dialogue avec un protocole et un brochage particulier référencé sous le nom i2c ou TWI (*Two Wire Interface*, le nom « i2c » étant déposé par Philips/NXP). Heureusement pour nous, le microcontrôleur Atmel ATmega328p au cœur de l'Arduino Uno sait parfaitement « parler » le TWI lorsqu'on lui demande gentiment. Les versions actuelles des cartes Arduino possèdent deux connecteurs dédiés libellés SCL et SDA comme c'est le cas sur la Uno, près du bouton de reset. Sur les modèles plus anciens, le TWI était également possible, mais via les broches A4 et A5. En réalité, A5 est physiquement connecté à SCL et A4 à SDA. Il ne vous est pas possible d'utiliser les deux en même temps.

Il ne faut guère plus que ces deux broches ainsi qu'une alimentation +5V et une masse pour utiliser un composant en TWI. Il s'agit d'un bus et, bien que cela ne nous soit pas utile ici, il est possible de mettre plusieurs composants TWI sur les mêmes connexions SCL (horloge) et SDA (données). Chaque composant sur le bus possède une adresse unique permettant de l'identifier. Celle d'un DS1307 est `0x68` et ne peut être changée, mais d'autres composants possèdent parfois des broches permettant le choix de l'adresse. Ceci n'est pas un élément à prendre en considération ici puisque nous utiliserons une bibliothèque dédiée aux RTC qui gère tout cela pour nous, mais souvenez-vous de cette précision si vous souhaitez jouer avec le bus TWI.

---

3 Enfin, nous avons la connexion du module de détection d'impact de foudre. Celui-ci utilise un autre bus très utilisé : le SPI. Nous avons déjà évoqué ce bus dans les pages du magazine et résumerons simplement la chose en détaillant les broches du module lui-même :

- GND : la masse, tout simplement ;
- Vdd : la tension d'alimentation qui dans le cas du module AS3935 est entre 2.4V et 5.5V (attention, les tensions correspondant aux niveaux logiques sont à adapter en conséquence). Pourquoi Vdd et pas Vcc ? En pratique, ces termes désignent la même chose et sont hérités de l'histoire de l'électronique. « Vcc » fait référence à la broche « collecteur » des transistors bipolaires reliés côté anode, l'émetteur lui est relié à la masse, désignée par « Vee ». Avec l'arrivée de la technologie MOSFET, il n'y avait plus de raison de parler de collecteur et d'émetteur, mais plutôt de drain et de source, et donc de « Vdd » et de « Vss ». Avec le temps, les choses se sont un peu mélangées et il est courant de parler de « Vcc » et de « Vss » respectivement pour la tension d'alimentation et la masse,
- MISO : *Master In - Slave Out* concerne les données émises par le périphérique esclave (le module) et reçues par le maître (la carte Arduino) ;
- MOSI : *Master Out - Slace In* est le canal de communication inverse. À la différence du bus i2c/TWI, le SPI fournit une communication dite full-duplex puisque les échanges dans les deux sens peuvent



avoir lieu en même temps grâce aux deux canaux MISO et MOSI. Le TWI est dit half-duplex, chaque intervenant « parle » à tour de rôle ;

- CLK : c'est la ligne d'horloge cadençant les échanges (et est parfois appelée SCK), elle est contrôlée par le maître quel que soit le sens de la communication (contrairement au TWI où SCL est bidirectionnel comme SDA) ;
- CS : permet de « désigner » le périphérique ou l'activer. Il n'y a pas d'adresse sur un bus SPI, mais on peut tout de même utiliser plusieurs esclaves avec une seule liaison MISO/MOSI/CLK. Pour que le maître puisse préciser à qui il s'adresse, il utilise une ligne CS différente pour chaque esclave,
- IRQ : ceci n'est pas spécifique au bus SPI, mais est relativement courant. Pour que l'esclave puisse signifier qu'il a quelque chose à dire, il ne peut rien faire, car c'est le maître qui contrôle la communication. Il faut donc une broche spéciale pour signaler un événement sous la forme d'un « *hé ! Arrête ce que tu fais, il se passe quelque chose* » ou en d'autres termes une demande d'interruption de la part du périphérique : une *Interrupt ReQuest* ou IRQ en abrégé ;
- SI : là, il s'agit de quelque chose de spécifique. SI signifie *Select Interface* et permet au module d'être utilisé en SPI (SI à la masse) ou en i2c/TWI (SI à Vdd). Dans ce second cas, CLK devient SCL (horloge TWI) et MOSI devient SDA (données bidirectionnelles TWI), CS et MISO doivent alors être mis à la masse.

La connexion entre le module et l'Arduino se fait en utilisant les broches dédiées au bus SPI :

- MOSI : 11 ;
- MISO : 12 ;
- et CLK : 13.

CS est contrôlé par la broche 10 et SI par la broche 9. Notez sur ce point que SI pourrait être simplement relié à une masse, mais que la bibliothèque fournie par *Playing With Fusion Inc* intègre le contrôle de cette broche dans le code. Passer outre ce choix de la part du développeur (J. Steinlage) implique de modifier la bibliothèque. Ceci se résume à une poignée de lignes de code, mais l'article est déjà suffisamment complexe et long...

coromonadalix

LE CROQUIS

Fichier Édition Croquis Outils Aide

```
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
#include <LiquidCrystal.h>
#include <SPI.h>
#include <PWFusion_AS3935.h>

// définition des broches pour LCD
#define LCD_RS 8
#define LCD_EN 7
#define LCD_D4 A0
#define LCD_D5 A1
#define LCD_D6 A2
#define LCD_D7 A3

// déclaration afficheur LCD
LiquidCrystal lcd(LCD_RS, LCD_EN, LCD_D4, LCD_D5, LCD_D6, LCD_D7);

// configuration broches module AS3935
#define CS_PIN   10 // chip select
#define SI_PIN   9  // sélection interface
#define IRQ_PIN  2  // broche 2 et 3 avec interruption
#define IRQ_NUM  0  // interruption 0 ou 1 sur Uno (broches 2 ou 3)

// variable d'état pour interruption
volatile int8_t AS3935_ISR_Trig = 0;

// définition pour configuration AS3935
#define AS3935_INDOORS      0    // en intérieur
#define AS3935_OUTDOORS     1    // en extérieur
#define AS3935_DIST_DIS     0    // désactivation distance
#define AS3935_DIST_EN      1    // activation distance
#define AS3935_CAPACITANCE  104  // calibration / étiquette

// fonction interruption
void AS3935_ISR() {
  AS3935_ISR_Trig = 1;
}

// déclaration capteur AS3935
PWF_AS3935 lightning(CS_PIN, IRQ_PIN, SI_PIN);

// affichage valeur sur 2 chiffres LCD
void lcdprint2digits(int number) {
  if (number >= 0 && number < 10)
    lcd.print('0');
```

coromonadalix

```
  lcd.print(number);
}

// affichage heure LCD
void lcdprinttime(int ligne) {
  tmElements_t tm;

  if (RTC.read(tm)) {
    // effacement LCD
    lcd.clear();

    lcd.setCursor(0,ligne);
    lcdprint2digits(tm.Day);
    lcd.print("/");
    lcdprint2digits(tm.Month);
    lcd.print("/");
    lcd.print(tmYearToCalendar(tm.Year));
    lcd.print("  ");
    lcdprint2digits(tm.Hour);
    lcd.print(":");
    lcdprint2digits(tm.Minute);
    lcd.print(":");
    lcdprint2digits(tm.Second);
  }
}

// configuration
void setup() {
  // activation LCD 4*20
  lcd.begin(20, 4);
  // la date et heure
  lcdprinttime(0);
  // + message de démarrage
  lcd.setCursor(0,1);
  lcd.print("Boot...");

  // activation SPI
  SPI.begin();
  // horloge SPI /16
  SPI.setClockDivider(SPI_CLOCK_DIV16);
  // SPI mode1
  SPI.setDataMode(SPI_MODE1);
  // octet de poids fort en premier
  SPI.setBitOrder(MSBFIRST);

  // initialisation AS3935
  lightning.AS3935_DefInit();
  // calibration AS3935
  lightning.AS3935_ManualCal(AS3935_CAPACITANCE, AS3935_INDOORS, AS3935_DIST_EN);
  // ajout interruption sur front montant
  attachInterrupt(IRQ_NUM, AS3935_ISR, RISING);
}
```

coromonadalix

```
// boucle principale
void loop() {
  // boucle sans fin en attente d'interruption
  while(AS3935_ISR_Trig == 0){}

  delay(3);
  // interruption détectée, on repasse à 0 et on traite
  AS3935_ISR_Trig = 0;

  // vérification de la source d'interruption
  uint8_t int_src = lightning.AS3935_GetInterruptSrc();

  // on switch sur la valeur de la source
  switch(int_src) {
    case 0:
      // source non prise en charge
      // affichage de la date/heure
      lcdprinttime(0);
      // seconde ligne LCD
      lcd.setCursor(0,1);
      lcd.print("Autre interruption");
      break;
    case 1:
      // c'est une vraie détection
      lcdprinttime(0);
      lcd.setCursor(0,1);
      lcd.print("Impact Foudre !");
      // troisième ligne LCD
      lcd.setCursor(0,2);
      lcd.print("Distance : ");
      // on affiche la distance estimée par l'AS3935
      lcd.print(lightning.AS3935_GetLightningDistKm());
      lcd.print(" km");
      break;
    case 2:
      // c'est une interruption
      // mais pas un impact reconnu
      lcdprinttime(0);
      lcd.setCursor(0,1);
      lcd.print("Perturbation...");
      break;
    case 3:
      // l'AS3935 nous informe qu'il ne fait plus la
      // différence entre le bruit et les impacts
      lcdprinttime(0);
      lcd.setCursor(0,1);
      lcd.print("Trop de bruit...");
      break;
  }
}
```

Arduino

coromonadalix

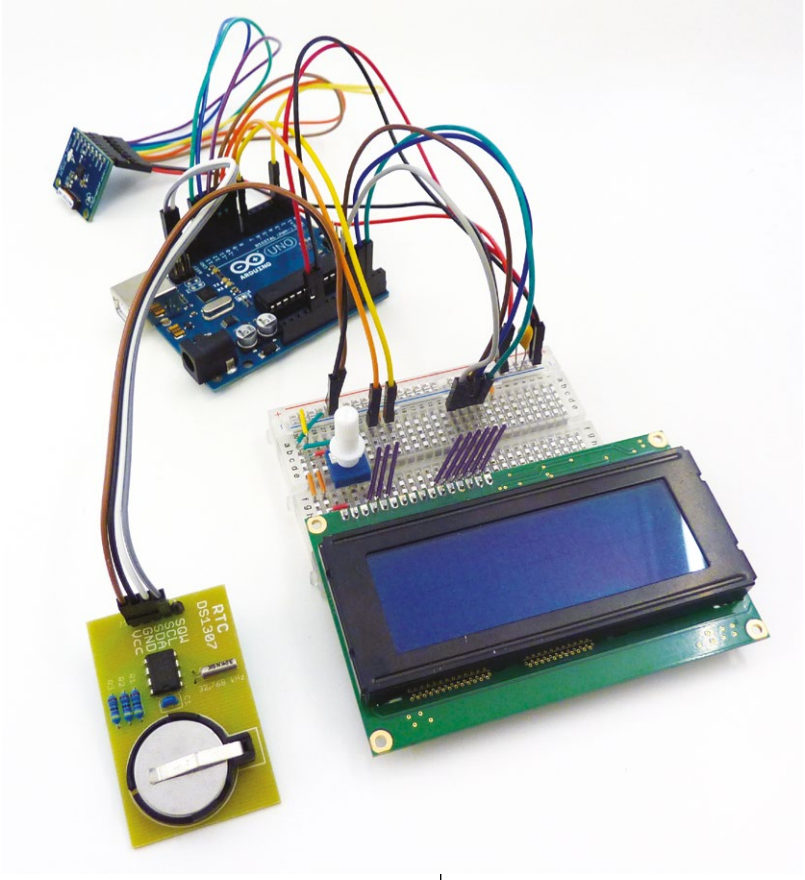
À PROPOS DU CROQUIS

Le croquis présenté ici est inspiré de l'exemple fourni avec la bibliothèque de *Playing With Fusion Inc* pour leur module. Celle-ci est téléchargeable sur la page produit de leur boutique (**http://playingwithfusion.com/productview.php?pdid=22**) sous la forme d'un fichier **`sen39001_arduino_r00.zip`**. Malheureusement, l'archive n'est pas structurée de façon à permettre une installation via l'environnement Arduino. Il faudra donc :

- récupérer le fichier **`sen39001_arduino_r00.zip`** ;
- le désarchiver dans un répertoire créé à cet effet et appelé **`PWFusion_AS3935`** (c'est capital) ;
- déplacer le répertoire en question dans le sous-répertoire **`libraries`** du répertoire correspondant à votre carnet de croquis (très probablement appelé **`sketchbook`**, vérifiez dans les préférences).

Après redémarrage de l'environnement Arduino, cette bibliothèque sera utilisable.

Il nous faudra également de quoi accéder au bus SPI, à l'afficheur LCD et au bus i2c/TWI. Ceci est déjà pris en charge par l'environnement respectivement sous la forme des bibliothèques *Wire*, *LiquidCrystal* et *SPI*. Il n'en va pas de même pour le support de la RTC DS1307 et de la gestion du format de date.



L'assemblage expérimental se transforme rapidement en salade de câbles en raison des différents bus utilisés. Il sera possible de simplifier cela en passant par exemple tout en TWI. L'AS3935 est connecté ici en SPI, mais est utilisable en TWI et des afficheurs LCD TWI/i2c existent (mais sont plus chers). L'étape finale du projet après essais et vérifications est bien entendu de réaliser un montage définitif sans platine à essais et en remplaçant l'Arduino par un simple Atmega328p (et pourquoi pas ajouter du Bluetooth ou du Wifi, et une signalisation sonore).

Comme vous avez très certainement mis à jour votre environnement avec la nouvelle version 1.6.4, il ne sera pas nécessaire de jongler une seconde fois avec des fichiers et des répertoires.

Rendez-vous simplement dans le menu ***Croquis***, ***Include Library*** et ***Manage Libraries*** (non, ce n'est pas encore traduit). Une nouvelle fenêtre s'affiche vous permettant le gérer les bibliothèques, d'en installer et de les mettre à jour. Dans la boite de recherche en haut à droite, cherchez simplement « time ». Dans la liste, installez les

bibliothèques « DS1307RTC » et « Time » de Michael Margolis. Cliquez simplement sur l'entrée correspondante puis sur le bouton ***Install*** qui apparaît alors sur la droite. Les bibliothèques sont directement utilisables sans redémarrage.

La première chose à faire suite à cela est de régler l'heure sur le module RTC s'il s'agit de sa première utilisation ou si sa pile a été temporairement retirée. Ceci peut être fait rapidement en ouvrant l'exemple **SetTime** livré avec la bibliothèque DS1307RTC. En chargeant cet exemple dans la carte Arduino, le module attaché sera automatiquement mis à la date et à l'heure de la compilation, elle-même tirée des paramètres actuels de votre ordinateur.

Concernant le croquis présenté ici, la majorité du code devrait être intelligible grâce aux différents commentaires ajoutés. Faisons simplement le tour des spécificités et des points les plus intéressants.

Notez tout d'abord l'utilisation massive de l'instruction **#define** permettant de déclarer des définitions. Il ne s'agit pas de variables, mais d'alias. Chaque fois que le compilateur (ou plutôt le préprocesseur) verra, par exemple **LCD_D4**, il utilisera en lieu et place **A0**. Tout ceci a lieu avant la compilation et consiste à un simple remplacement à la volée. C'est une technique courante pour rendre le code plus lisible sans utiliser la mémoire vive (SRAM).

Remarquez les définitions de **AS3935_INDOORS**, **AS3935_OUTDOORS**, **AS3935_DIST_DIS** et **AS3935_DIST_EN** qui ne sont que des « alias » pour **0** ou **1**. Ceci, selon moi, devrait plutôt trouver sa place dans **PWFusion_AS3935.h** faisant partie de la bibliothèque *PWFusion_AS3935* puisque c'est typiquement l'un des principaux intérêts d'un fichier d'en-tête **.h**.

Le croquis utilise une interruption selon le mécanisme suivant :

- le code configure qu'une interruption doit être déclenchée lorsque la broche 2 passe de l'état bas à l'état haut, grâce à **attachInterrupt()** ;
- lorsque cela arrive, le fonctionnement du croquis est interrompu pour appeler la fonction **AS3935_ISR()** qui est la routine d'interruption ou ISR (*Interrupt Service Routine*) ;
- cette fonction ne fait pas grand-chose (les ISR se doivent d'être concises) et change simplement la valeur de la variable globale **AS3935_ISR_Trig** à 1, signifiant qu'une interruption a eu lieu ;
- le code reprend alors son fonctionnement normal ;
- jusqu'à arriver au test **while(AS3935_ISR_Trig == 0)** qui est alors VRAI et on sort donc de la boucle **while** vide **{}** ;
- la première chose qu'on fait consiste à remettre **AS3935_ISR_Trig** à 0 pour l'interruption suivante ;
- puis on traite l'information en dialoguant avec le module AS3935.

Pour savoir ce qui vient de se passer, on interroge le module pour obtenir la source de l'interruption avec **AS3935_GetInterruptSrc()** qui nous retourne une valeur. Celle-ci provient de l'AS3935 et indique le type d'événement requérant notre attention.

Pour faire le tri, nous utilisons une construction particulière du C/C++ appelée un switch/case. Plutôt que de tester la valeur de **int_src** avec des **if**/**else** imbriqués comme c'est le cas dans l'exemple livré avec la bibliothèque, nous dispatchons nos actions en fonction des valeurs possibles :

```
switch(expression) {
  case valeur:
    // des trucs
    break;      // optionnel
  case autre_valeur:
    // autres trucs
    break;      // optionnel
  default:      // optionnel
    // truc à faire par défaut
}
```

Le **break** permet de stopper le processus et de sortir du **switch** et donc de sauter tous les autres **case**, y compris **default** qui est normalement utilisé lorsque rien d'autre ne correspond. L'utilisation du

coromonadalix

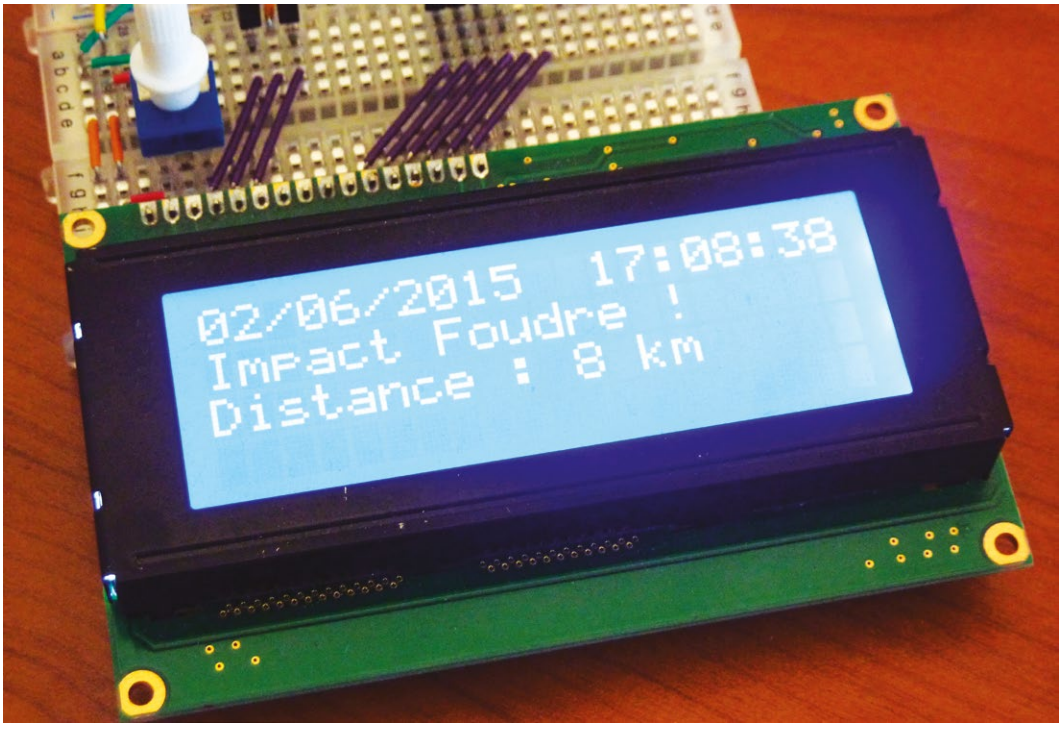
switch/case est ici plus lisible qu'une succession de `if`/`else`, en particulier si vous comptez désactiver certaines actions.

En effet, selon votre environnement et la proximité de sources de perturbations électrostatiques, il est probable que vous voyiez s'afficher régulièrement le message correspondant. Comme l'afficheur LCD ne renseigne que sur le dernier événement, vous ne saurez donc pas quand s'est produit la dernière détection valable. Avec le switch/case, il suffit de mettre en commentaire les lignes correspondantes sans avoir à se replonger dans le code. Ceci permet également d'intégrer plus facilement d'autres « notifications » selon le type d'événement.

CONCLUSION

Il y a bien de la place pour l'évolution de ce croquis. Avec seulement quelques 22% de la mémoire flash utilisée et quelques broches restant disponibles sur la carte Arduino on pourra envisager quelques améliorations comme :

- ajouter une notification sonore et/ou lumineuse ;
- garder une trace des détections en mémoire ou en EEPROM ;
- connecter un ou deux boutons au montage pour faire défiler les dernières informations importantes ;
- comptabiliser le nombre d'impacts sur diverses périodes en heures ou en jours ;
- remonter les informations à un PC ou une carte Raspberry Pi pour faire des statistiques ou des graphiques ;
- calculer un niveau d'alerte et déclencher une alarme quelconque en fonction de la fréquence des impacts et de la distance estimée ;
- etc.



Le résultat obtenu par le montage avec en direct une détection d'un impact isolé. Notez qu'on parle d'impact, mais la foudre peut se former également entre deux nuages. L'AS3935 est capable de différencier un véritable impact d'une interférence et d'estimer ses caractéristiques, dont la distance et l'énergie développée sous la forme d'une valeur sans unité.

coromonadalix

Le site lightningmaps.org affiche une carte des impacts de foudre détectés par un réseau de détecteurs installés par les utilisateurs. Le matériel utilisé est bien plus avancé qu'un simple détecteur en module comme celui présenté dans cet article et aussi plus cher. Mais si l'activité de détection vous passionne vraiment, c'est vers ce site et blitzortung.fr qu'il faudra orienter vos recherches.

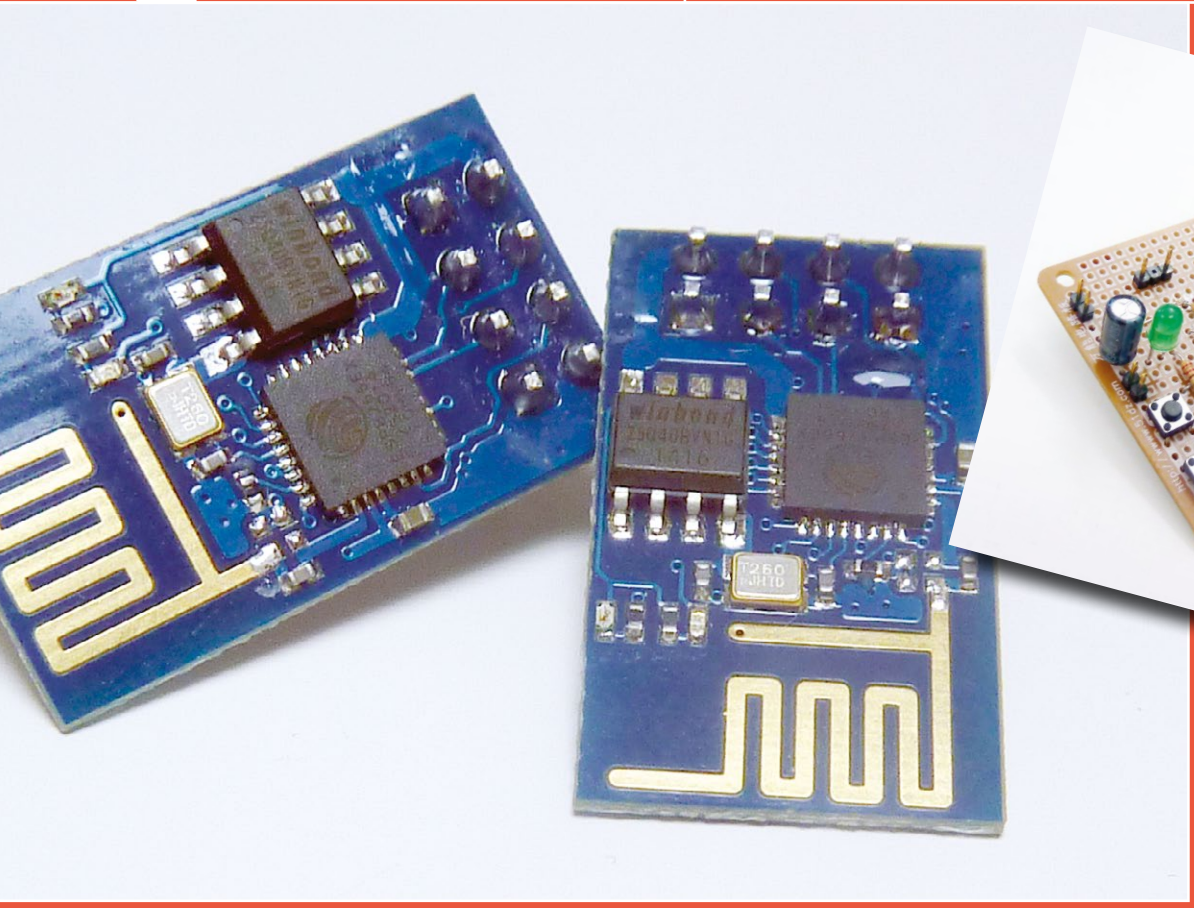
L'ensemble et en particulier le module AS3935 semblent extrêmement fiables bien que la documentation précise que ce circuit intégré ne doit pas être utilisé seul pour assurer la sécurité des personnes et des installations critiques. J'ai eu le plaisir à plusieurs reprises de m'assurer de la correspondance entre la proximité d'un orage et les informations du capteur. Bien qu'occasionnellement un impact isolé soit détecté, la présence d'une cellule orageuse provoque des détections massives et répétées, confirmées par des données visuelles, sonores et olfactives (si, si, on sent l'ozone) et les données collectées et diffusées par le site de l'Observatoire Français des Tornades et des Orages Violents (**http://www.keraunos.org/**).

Je recommande au passage fortement la visite du site KERAUNOS et en particulier des cartes en temps réel des impacts de foudre (monde, Europe, France, régions). C'est un vrai plaisir de voir un montage fonctionner de concert avec les données du site. Le seul regret est sans doute l'absence d'informations de direction concernant la source d'un impact. Il devrait être possible d'utiliser plusieurs modules AS3935 avec des antennes directionnelles ou des paraboles, mais le coût ainsi que le temps de mise au point sont malheureusement assez dissuasifs...

Le site communautaire **blitzortung.org** utilise un réseau de détecteurs différents et bien plus performants qu'un simple AS3935. Un kit complet existe intégrant une carte contrôleur (une STM32F4DISCOVERY de ST), un afficheur LCD graphique, une antenne GPS, deux cartes d'amplification et deux antennes ferrites. Oui, deux et ce n'est pas tout. L'ensemble se complète d'une carte d'amplification et une autre de préamplification pour une antenne VLF. Initialement vendu autour de 200€, le kit est actuellement épuisé, mais une nouvelle version est en préparation. L'ensemble des informations de détection provenant du réseau de participants permet la mise à jour d'une carte dynamique disponible sur **blitzortung.org** ainsi qu'une version Google Maps sur **lightningmaps.org**. Pour plus d'informations sur le projet, le kit et le réseau, pointez votre navigateur vers le forum officiel français : **blitzortung.fr**.

Pour une détection omnidirectionnelle cependant, l'AS3935 et ce module en particulier, pourront joyeusement compléter une installation météo amateur ou un système domotique. En ce qui me concerne, la conversion en circuit définitif est d'ores et déjà planifiée avec ajout d'un buzzer et son activation uniquement possible dans une tranche horaire précise. Si une RTC est présente, autant s'en servir pour éviter des bip-bip nocturnes intempestifs... **DB**

coromonadalix



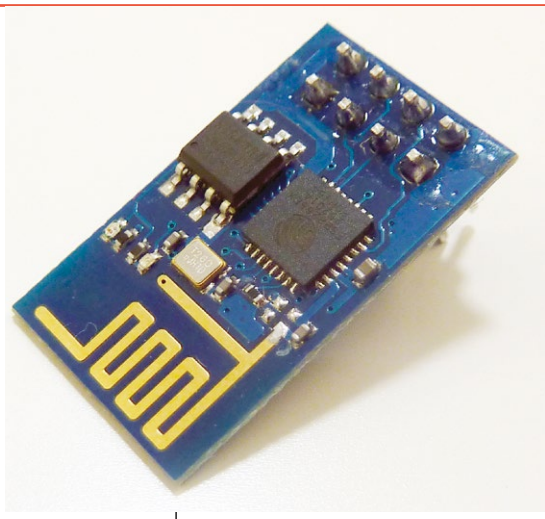
UN « ARDUINO » AVEC WIFI POUR MOINS DE 10 EUROS ?

Denis Bodor

Non, il ne s'agit pas de simplement ajouter des fonctionnalités Wifi à une carte Arduino avec un shield. Ce serait trop simple, trop cher et trop imposant. Nous allons plutôt utiliser ce qui est initialement un module Wifi pour en faire quelque chose d'assez similaire à une carte Arduino, ou du moins pouvant être programmé comme tel, avec l'IDE Arduino.

NIVEAU	TEMPS	BUDGET
	15 minutes	environ 10 €

coromonadalix



Le module Wifi chinois généralement désigné par ESP-01 est sans le moindre doute le modèle le plus courant, le moins cher, mais aussi celui offrant le moins d'entrées/sorties. Celui-ci provient de chez Electrodragon et intègre par défaut un firmware permettant de l'utiliser en tant que périphérique Wifi d'une carte Arduino par exemple.

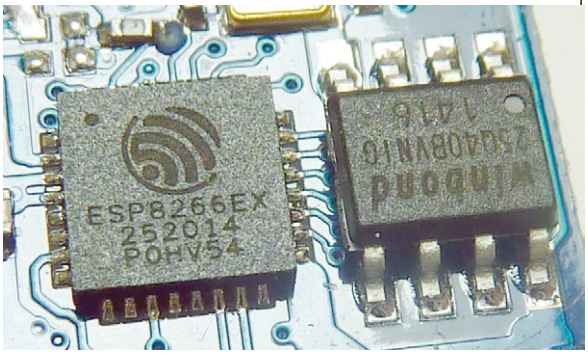
Deux principaux modules de communication sans fil emportent actuellement un énorme succès en dehors du Bluetooth. En réalité, il serait plus juste de parler de deux puces, car les modules et circuits existant sur cette base sont très diversifiés. Nous avons d'une part le nRF24L01+ de Nordic Semiconductor qui est un émetteur-récepteur à très faible consommation fonctionnant sur 2,4GHz (bande ISM). Il peut s'interfacer avec une carte Arduino ou une Raspberry Pi via une connexion SPI, comme avec n'importe quel nano-ordinateur/SBC (pour *Single Board Computer*) ou carte à microcontrôleur. Derrière ses apparences anodines, le nRF24L01+ est très complexe et dispose d'un support important en termes de bibliothèques Arduino ou de système GNU/Linux. En plus d'être très utilisé côté bidouille et réalisation électronique, il est également massivement présent dans les périphériques informatiques sans fil type claviers et souris.

L'autre puce est l'ESP8266 d'Espressif Systems. Contrairement au nRF24L01+, celle-ci forme la base pour des modules Wifi et est donc en mesure de dialoguer avec l'ensemble du matériel existant (point d'accès, PC, Mac, smartphone, etc.) alors que le nRF24L01+ ne communique qu'avec des modules compatibles construits autour du même composant ou d'autres puces Nordic Semiconductor. Les nRF24L01+ vont donc naturellement par deux ou plus, mais l'ESP8266 communique avec des équipements que vous possédez déjà, sans doute en quantité.

Les domaines d'application sont très différents entre l'un et l'autre composant, au point qu'il ne serait pas pertinent de les mettre en balance. Dans les deux cas cependant d'importantes communautés de développeurs, de bidouilleurs et d'utilisateurs se sont formées naturellement pour agréger des ressources, des documentations et des réalisations. De la même manière, pour le nRF24L01+ comme pour l'ESP8266, les fabricants, détaillants et revendeurs de modules sont à présent légion. On peut donc trouver facilement des modules utilisant l'un et l'autre, dans différentes déclinaisons, à des prix inférieurs à 5 euros pièce.

Dans le cadre du présent article, les modules à base d'ESP8266 présentent un petit plus : le composant lui-même n'est autre qu'un microcontrôleur à cœur Tensilica Xtensa LX106 (processeur RISC 32 bits) complété de fonctionnalités Wifi associées à une mémoire flash contenant le firmware. Celui proposé par défaut permet une configuration du composant par l'intermédiaire d'un ensemble de commandes AT (comme pour les modules Bluetooth ou les modems). Le principe consiste alors à configurer le module en client Wifi ou en point d'accès (ou les deux) et à communiquer avec lui via une liaison série (RX/TX). Ceci permet, par exemple, à une carte Arduino de communiquer avec n'importe quelle machine présente sur un réseau (filaire ou Wifi) sans avoir à se soucier de la partie gestion Wifi ou TCP/IP.

Mais quelques développeurs ont eu une bien meilleure idée. Si le cœur de l'ESP8266 n'est



Gros plan sur les deux composants au cœur des modules Wifi avec à gauche l'ESP8266 et à droite la mémoire flash de 512 Ko de chez Winbond. Ce duo est la configuration la plus fréquente pour l'ensemble des modules du marché. Seuls certains modèles spécifiques comme la version Olimex proposent une flash plus importante.

qu'un microcontrôleur (à 80 Mhz tout de même) qui ne fait qu'exécuter un programme, pourquoi ne pas lui faire exécuter un code compilé à partir d'un croquis Arduino ? Et c'est ainsi qu'a vu le jour l'« *Arduino-compatible IDE with ESP8266 support* », ou en d'autres termes, un environnement de développement Arduino ayant pour cible l'ESP8266. Initialement disponible comme un clone de l'environnement existant, le résultat de ces développements est maintenant pris en charge par le *Board Manager* de l'IDE Arduino 1.6.4. On peut donc très simplement intégrer le support ESP8266 dans une installation existante, en quelques clics de souris et programmer un module ESP8266 comme une carte Arduino.

LE PRINCIPE

Les modules à base d'ESP8266 reposent tous sur le même principe : ils sont vendus pour une utilisation via l'interface série et fournissent une connectivité Wifi en prenant en charge les protocoles réseaux TCP/IP, mais aussi des fonctionnalités de plus haut niveau comme HTTP. On peut donc, depuis une carte Arduino, simplement envoyer des requêtes HTTP et obtenir les réponses. Autrement dit, on spécifie des URL et on reçoit le contenu des pages.

Mais l'ESP8266 est un microcontrôleur, exactement comme l'Atmel AVR ATmega328p d'une carte Arduino Uno ou le AT91SAM3X8E (ARM Cortex-M3) d'une Arduino Due. C'est un composant programmable pouvant recevoir un code à exécuter, dès lors qu'on dispose du compilateur et des outils de programmation adaptés. Depuis peu, cette étape est grandement facilitée puisqu'il suffit d'aller dans les préférences de l'environnement Arduino et, à la section ***Additional Boards Manager URLs***, spécifier **http://arduino.esp8266.com/package_esp8266com_index.json**. Dès lors, un petit tour dans le menu ***Outil***, ***Type de carte***, ***Boards Manager*** vous permettra d'installer le support « esp8266 by ESP8266 Community ». Notez que le support ESP8266 est actuellement disponible pour Windows 32 et 64 bits, GNU/Linux 32 et 64 bits (le 32 bits est un ajout très récent (il y a 9 jours)) et Mac OS X 64 bits (10.6 et supérieur).

Ceci fait, une nouvelle « carte » est accessible dans le menu ***Outil***, ***Type de carte*** : « Generic ESP8266 Module ». Sa sélection fera apparaître de nouveaux menus dans ***Outil***, permettant de spécifier la vitesse (80 Mhz ou 160 Mhz), le débit pour la programmation et la taille de la mémoire flash pour le programme. Les valeurs par défaut (80 Mhz, 115200 et 512k) conviennent parfaitement pour mes modules provenant de chez Electrodragon, vendus sous la désignation « ESP-01 ESP8266 Wifi Module WI07c ». Il semblerait que ce soit actuellement le modèle le plus courant et massivement distribué » sous bien des noms.

CE QU'IL VOUS FAUT

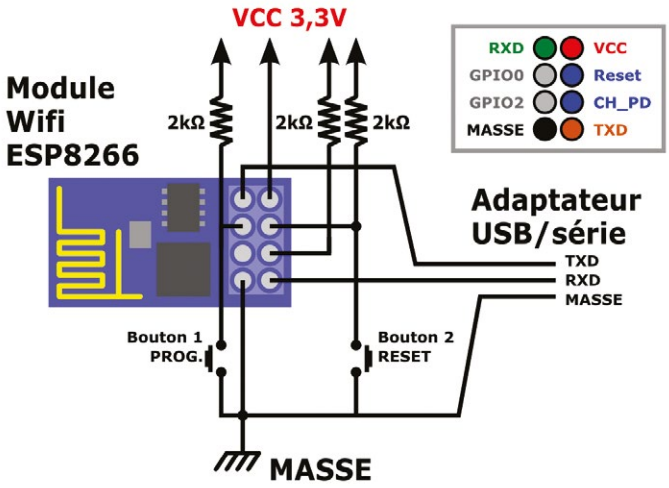
- **Un module ESP8266 : il existe bien des modèles de modules à base d'ESP8266. Le plus fréquent et le moins cher c'est l'ESP-01 (3,70€) également référencé sous WI07c. Celui-ci se présente sous la forme d'un circuit minuscule équipé d'un connecteur 8 broches en deux rangs de 4 broches (assez pénible pour une utilisation sur platine à essais). On peut considérer qu'il s'agit du modèle le plus basique qui soit, car l'ESP8266 lui-même propose bien plus de ports. Une autre version du module, l'ESP07 (parfois ESP-12E), sensiblement plus cher (~7€) propose 11 GPIO, une entrée analogique et un bus SPI en plus de la liaison série et de l'alimentation. Il est également équipé d'un blindage métallique destiné à réduire la sensibilité aux perturbations électromagnétiques. Enfin, nous avons la version d'Olimex, une société familiale bulgare produisant également les cartes OLinuXino, sous le nom « MOD-WIFI-ESP8266-DEV ». Cette déclinaison offre une grande quantité d'entrées/sorties ainsi que l'accès au convertisseur analogique/numérique, mais aussi un connecteur SDIO et surtout 2 Mo de flash en lieu et place des 512 Ko du module basique chinois, le tout pour un prix entre 6€ et 10€ suivant la source d'achat, la quantité et les frais de livraison. Notez qu'un kit**

de développement à 17€ est également disponible, intégrant un circuit d'alimentation, un relais et un bouton pour la programmation. La version Olimex est directement référencée dans la liste des cartes une fois le support ESP8266 installé.

- Un adaptateur USB/série 3,3V : ceci est un impératif non négociable. La communication avec n'importe quel module ESP8266 qu'il s'agisse d'échanges d'informations ou de sa programmation nécessite une liaison série avec des niveaux de tension 0/3,3V. Les modules ESP8266 ne sont pas, sauf exception, équipés de régulateurs 5V/3,3V et ne doivent EN AUCUN CAS être alimentés en 5 volts. Ceci est également valable pour les communications série, l'ESP8266 n'étant pas tolérant aux signaux 5 volts. Ne pas suivre cette règle pourra conduire à la destruction pure et simple du module. Si vous avez des doutes sur votre adaptateur USB/série, mieux vaut en acheter un autre, car entre détruire un module à 4€ et acheter un adaptateur à 4€, la dépense est la même et vous serez systématiquement gagnant.

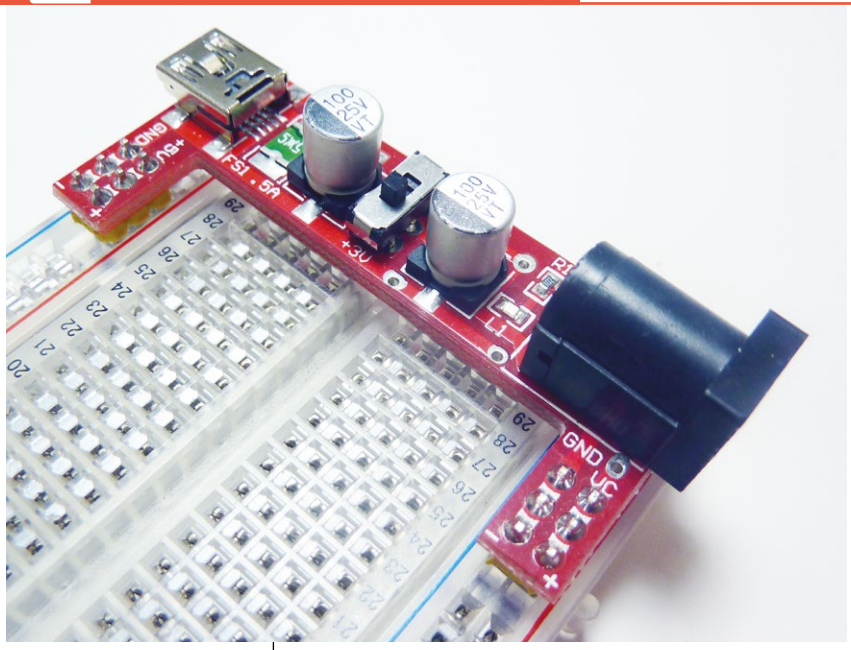
- Quelques résistances de rappel (entre 2 Kohms et 10 Kohms), deux boutons poussoir, une platine à essais et des connecteurs. Les boutons poussoir ne sont là que pour faciliter les choses, il existe d'autres solutions dont nous parlerons par la suite.

LE MONTAGE



Il existe plusieurs façons de connecter un module ESP8266 à un adaptateur série, mais celle, à mon sens, la plus simple et la plus fiable est présentée ici. Pour comprendre la connectique et la raison d'être des boutons et des résistances, il faut avant toutes choses connaître l'utilité de chaque broche :

- VCC : c'est la source d'alimentation en 3,3 volts. Le module utilisera au maximum, d'après les mesures effectuées par les utilisateurs et développeurs, 215 mA. Vous devez donc vous assurer de fournir au module suffisamment de courant. Selon le modèle d'adaptateur USB/série, il sera possible d'alimenter le module ESP8266 directement. Dans le cas contraire, une source d'alimentation dédiée sera nécessaire (cf légende de la photo de l'alimentation plus loin dans l'article).
- Masse : la classique et incontournable connexion à la masse.
- RXD : la ligne de réception de données, utilisée aussi bien pour la programmation que pour la communication avec votre futur croquis exécuté par l'ESP8266.
- TXD : la broche dédiée à l'envoi de données, la copine de la broche précédente.
- Reset : cette broche permet de déclencher une réinitialisation du module lorsqu'elle est connectée à la masse. Dans la plupart des cas, les modules commercialisés n'intègrent pas de résistance de rappel à Vcc permettant de « tirer » cette ligne à la tension d'alimentation lorsqu'elle n'est pas connectée. On utilisera ici une résistance entre 2 et 10 Kohms et ajouterons un bouton poussoir connecté à la masse. Ainsi une pression sur le bouton réinitialisera le module.



Pour alimenter le module Wifi en 3,3 volts tout en disposant d'une tension plus élevée, par exemple pour un relais 5V, il existe des modules d'alimentation très économiques comme celui-ci. Vendu quelques 3€ sous la désignation « Supply Module for MB102 Bread Board » sur DealExtreme (dx.com), il s'adapte sur une platine à essais et fournira 700mA en 5V et 3,3V à partir d'une connexion USB (5V) ou d'un bloc secteur 6 à 12 V.

- CH_PD : désigné sous le nom *CHip Power Down* cette broche permet de désactiver l'ESP8266. Comme la ligne Reset, celle-ci est active au niveau bas et doit être ramenée à la tension d'alimentation avec une résistance de rappel. Nous nous passons ici totalement de contrôle sur cette ligne qui sera donc en permanence à l'état haut et de ce fait, l'ESP8266 ne sera jamais désactivé. Cette broche peut être intéressante dans le cas d'une utilisation du module avec son firmware d'origine et donc piloté par un microcontrôleur. On peut ainsi réduire la consommation en mettant cette broche à la masse et uniquement la placer à l'état haut lorsque le module doit être utilisé.
- GPIO0 : cette broche possède une double fonction. Lorsque le module est démarré normalement, cette ligne est une entrée/sortie standard. En revanche, si celle-ci est connectée à la masse lorsque le module est mis sous tension ou réinitialisé, l'ESP8266 passe en mode programmation. Le code utilisateur n'est alors pas exécuté au bénéfice de celui du bootloader et la mémoire flash est accessible pour le chargement d'un autre code. Cette broche est connectée de la même manière que Reset avec une résistance de rappel et un bouton poussoir. Pour passer le module en mode programmation, il suffira donc de presser ce bouton, de le maintenir enfoncé tout en appuyant sur le bouton de réinitialisation. Le même résultat pourra être obtenu en maintenant enfoncé ce bouton tout en connectant l'alimentation au module.
- GPIO2 : il s'agit d'une entrée/sortie standard. Notez cependant que cette broche sert de second TXD en mode programmation, ne vous étonnez donc pas de voir une activité sur cette broche lorsque vous chargez un nouveau croquis (il s'agit des réponses et accusés-réception du module aux ordres de programmation).

Le site de développement du support Arduino pour les modules ESP8266 sur GitHub présente une configuration alternative. Celle-ci est assez similaire à celle présentée ici, mais ne fait pas usage des boutons poussoir et des résistances de rappel. En lieu et place ce sont les signaux DTR et RTS du port série qui sont utilisés, respectivement pour GPIO0 et Reset. Ceci permet théoriquement de laisser l'environnement de développement se charger du passage en mode programmation, mais impose que l'adaptateur USB/série possède effectivement les broches DTR et RTS (ce qui n'est pas toujours le cas avec les modèles les moins chers). En ajoutant à cela le fait que selon le système d'exploitation les signaux peuvent être contrôlés par ailleurs, vous comprendrez que je préfère largement la solution « manuelle ». Il suffit de ne pas oublier de passer le module en mode programmation avant de charger le croquis et le tour est joué à coup sûr.

Les boutons poussoir ne sont pas une absolue nécessité, une simple connexion/déconnexion sur platine à essais fera également parfaitement l'affaire. Attention cependant à la longueur des fils. Durant mes expérimentations,

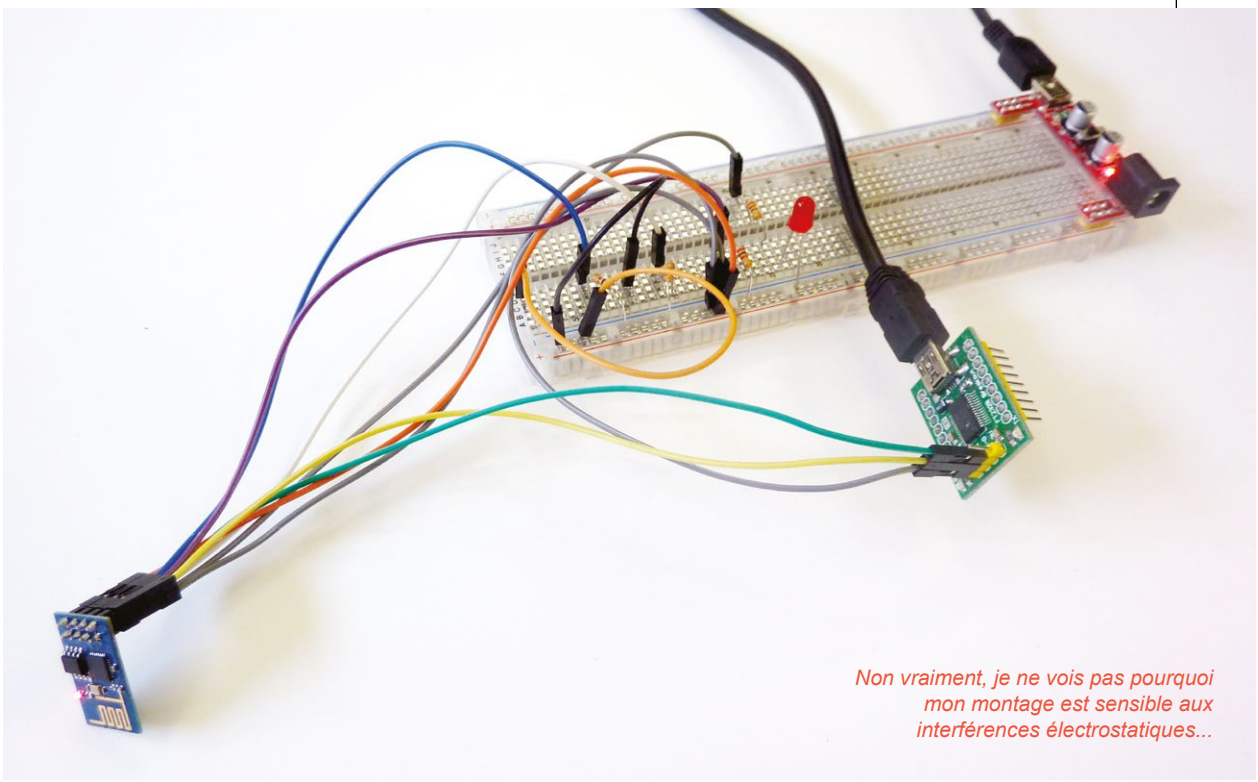
coromonadalix

j'ai eu la désagréable surprise de constater des réinitialisations intempestives d'un module dans diverses situations : extinction de la station de soudage, mise en marche de l'éclairage par tubes fluorescents et même passage de la position assise à debout avec un certain siège de bureau (qui est désormais officiellement classé dangereux pour tout équipement sensible). Le même module monté sur un circuit sur plaque pastillée, bien plus compact, ne posa plus aucun problème. Les platines à essai avec leurs contacts non francs et leurs longs connecteurs en boucle sont un véritable cauchemar en termes de perturbations électrostatiques...

La seule chose que le schéma présenté ici ne montre pas est la connexion de la broche GPIO2. C'est celle généralement la plus utilisée avec les modules chinois, car la seule disponible qui n'ait pas d'utilisation alternative. Quelle que soit son utilisation, deux choses doivent être gardées à l'esprit :

- nous sommes ici en 3,3 volts et il n'est pas question d'y connecter un signal en 5 volts ;
- en sortie, le port se comporte comme celui d'une carte Arduino. Il sera donc incapable de fournir un courant important, mais on pourra facilement tester le fonctionnement d'un croquis en ajoutant une led et une résistance de 220 ou 330 ohms.

Concernant GPIO0 et son utilisation alternative, la solution que j'ai choisie est de tout simplement utiliser un cavalier sur le circuit final. Dans une position, il permet l'utilisation du bouton avec la résistance de rappel et dans l'autre la connexion au reste du circuit. On peut ainsi disposer de deux entrées/sorties qui offrent un peu plus de perspectives qu'une seule (via utilisation d'un registre à décalage ou d'un démultiplexeur, type 74HCT139). Bien entendu, si vous avez dans l'idée de contrôler un afficheur LCD, à led 7 ou 16 segments ou encore un capteur quelconque, il faudra vous orienter vers un module ESP8266 fournissant davantage de GPIO et une connectivité SPI.



Non vraiment, je ne vois pas pourquoi mon montage est sensible aux interférences électrostatiques...

coromonadalix

LE CROQUIS

Fichier Édition Croquis Outils Aide

```
#include <ESP8266WiFi.h>

// Spécifier le SSID du point d'accès Wifi
const char* ssid = "your-ssid";

// Spécifier le mot de passe associé
const char* password = "your-password";

// Crée le serveur Web en spécifiant le port TCP/IP
// 80 est le port par défaut pour HTTP
WiFiServer server(80);

// Démarrage
void setup() {
  // Communication série 115200
  Serial.begin(115200);
  // Petite pause
  delay(10);

  // GPIO2 en sortie
  pinMode(2, OUTPUT);
  // GPIO2 à la masse
  digitalWrite(2, 0);

  // Deux sauts de ligne pour faire le ménage car
  // le module au démarrage envoie des caractères sur le port série
  Serial.println();
  Serial.println();
  Serial.print("Connexion a : ");
  Serial.println(ssid);

  // Connexion au point d'accès
  WiFi.begin(ssid, password);

  // On boucle en attendant une connexion
  // Si l'état est WL_CONNECTED la connexion est acceptée
  // et on a obtenu une adresse IP
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connecte");

  // Démarrage du serveur Web
  server.begin();
  Serial.println("Serveur demarre");

  // On affiche notre adresse IP
  Serial.println(WiFi.localIP());
}

// Boucle principale
void loop() {
  // Est-ce qu'un client Web est connecté ?
  WiFiClient client = server.available();
  if (!client) {
    // Non, on abandonne ici et on repart dans un tour de loop
    return;
```

coromonadalix

```
  }
  // Un client est connecté
  Serial.println("nouveau client");
  // On attend qu'il envoie des données
  while(!client.available()){
    delay(1);
  }
  // On récupère la ligne qu'il envoie jusqu'au premier retour chariot (CR)
  String req = client.readStringUntil('\r');
  // On affiche la ligne obtenue pour information
  Serial.println(req);
  // La ligne est récupérée, on purge
  client.flush();
  // Test de la requête
  int val;
  // Est-ce que le chemin dans la requête est "/gpio/0" ?
  if (req.indexOf("/gpio/0") != -1)
    // Oui, ceci correspond à "0" (OFF)
    val = 0;
  // Non, est-ce que le chemin est "/gpio/1" ?
  else if (req.indexOf("/gpio/1") != -1)
    // Oui, ceci correspond à "1" (ON)
    val = 1;
  // C'est un autre chemin qui est demandé et il ne correspond à rien
  else {
    // On signale que cette demande est invalide
    Serial.println("requete invalide");
    // et on déconnecte le client du serveur Web
    client.stop();
    // et on repart dans un tour de boucle
    return;
  }
  // Nous avons obtenu l'état demandé pour la sortie GPIO2
  // Nous pouvons écrire l'état de la sortie
  digitalWrite(2, val);

  // Peut-être le client a-t-il envoyé d'autres données,
  // mais nous purgeons la mémoire avant d'envoyer notre réponse
  client.flush();
  // Composition de la réponse à envoyer au client
  // On commence avec le texte d'une réponse HTTP 200 standard et un début de
  // page HTML
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n<html>\r\nLa sortie est a l'etat ";
  // puis on complète avec "haut" ou "bas" en fonction de la valeur de val
  s += (val)?"haut":"bas";
  // et on termine la page HTML
  s += "</html>\n";

  // On envoie la réponse au client
  client.print(s);
  // Petite pause
  delay(1);
  // et on le déconnecte du serveur Web
  Serial.println("Client deconnecte");

  // client.stop() n'est pas utilisé ici,
  // Le client est automatiquement déconnecté à la fin de la fonction loop(),
  // Lorsque l'objet "client" est détruit, car la déclaration est locale.
}
```

Arduino

coromonadalix

À PROPOS DU CROQUIS

Le croquis présenté ici n'a absolument rien d'original puisque c'est celui livré parmi les exemples du support ESP8266 sous le nom *WifiWebServer* (menu ***Exemples***, ***ESP8266WiFi***, ***WifiWebServer***). Il ne vous sera donc pas nécessaire de le saisir ou de le télécharger depuis notre dépôt GitHub. Nous avons cependant traduit et ajouté les commentaires pour en faciliter la compréhension.

Le fonctionnement du croquis se divise en deux parties. Nous avons en premier lieu la connexion au point d'accès Wifi (généralement votre box) et donc au réseau local, précédent le démarrage d'un serveur web. Tout ceci a lieu directement dans la fonction `setup()` et se résume à quelques lignes de code. L'ensemble des opérations techniques complexes se déroule en arrière-plan et fait partie des bibliothèques installées avec le support de l'ESP8266. C'est également dans `setup()` que le seul port GPIO facilement utilisable est initialisé, et ce exactement comme avec n'importe quel croquis pour une carte Arduino.

La seconde partie se résume au contenu de la fonction `loop()` et se charge de traiter les requêtes reçues par des clients web se connectant au serveur, typiquement un navigateur web. Vous avez ici un aperçu de la mécanique qui passe généralement inaperçue lorsque vous surfez.

Le serveur web embarqué dans le croquis se charge de la majeure partie du travail, ne laissant à votre charge que la gestion des messages reçus et envoyés. Ainsi lorsqu'un client se connecte au serveur web intégré, vous n'avez strictement rien à gérer. Ce n'est que lorsque celui-ci envoie une requête qu'il vous faut la tester pour en vérifier le contenu. Lorsqu'un utilisateur pointe son navigateur sur **http://adresse-ip-du-module/gpio/1**, par exemple, le module reçoit la chaîne `GET /gpio/1 HTTP/1.1`. En d'autres termes, le client demande (`GET`) la ressource se trouvant à l'emplacement `/gpio/1` sur le serveur, avec le protocole `HTTP/1.1`.

La partie qui nous intéresse est l'emplacement, ou chemin, et c'est sur cette information que nous déciderons s'il faut mettre la sortie GPIO2 à Vcc ou à la masse (ou simplement rejeter la connexion). Cette demande qui apparaît dans le moniteur série est exactement ce qu'envoie le navigateur lorsque vous y saisissez l'URL. Généralement, il l'accompagne d'une seconde demande, `GET /favicon.ico HTTP/1.1`, sans qu'on lui demande quoi que ce soit. Il s'agit de l'icône qui apparaît dans la barre d'adresse du navigateur. En utilisant un outil comme `curl` ou `wget` sous GNU/Linux en lieu et place d'un navigateur, la demande de la favicon n'a pas lieu.

La réponse envoyée par le croquis, après activation ou désactivation de la sortie, est également plus complexe que celle qui apparaît dans le navigateur et qui se résume à l'affichage de la phrase « *La sortie est a l'etat bas* » (oui, les accents sont absents, pour éviter les soucis d'encodage des caractères). En réalité, le navigateur affiche une page HTML formatée ainsi :

```
<!DOCTYPE HTML>
<html>
La sortie est a l'etat bas</html>
```

Ces données brutes peuvent être visualisées sur tous les navigateurs via un menu contextuel du type « afficher la source de la page ». Mais le navigateur vous cache des choses, la réponse réelle qu'il reçoit est :

```
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE HTML>
<html>
La sortie est a l'etat bas</html>
```

Les deux premières lignes, séparées du contenu par une ligne vide, constituent l'entête HTTP de la réponse et précisent

coromonadalix

le protocole utilisé, le code de la réponse et le type de données qu'elle contient. Nous avons ici affaire à du texte HTML, avec une réponse « OK » en protocole HTTP 1.1. Remarquez le 200 qui correspond grossièrement à « Ok, tout roule voilà la ressource ». Si vous vous demandiez d'où sort le « 404 » qui apparaît tantôt lorsque vous tentez d'accéder à une page qui n'existe pas, il s'agit tout simplement du code de réponse correspondant dans la norme. Je vous recommande vivement la lecture de la page Wikipédia dédiée aux codes de réponse HTTP (**http://fr.wikipedia.org/wiki/Liste_des_codes_HTTP**) et éventuellement, de fil en aiguille, la lecture de la RFC 2324 du 1er avril 1998 expliquant pourquoi il y a un code « *418 - I'm a teapot* » (« Je suis une théière »).

Une dernière remarque à propos du croquis concerne une ligne de code particulière utilisée dans la composition de la réponse :

```
s += (val)?"haut":"bas";
```

Cette chose cryptique est la concaténation d'une chaîne à une autre via un opérateur ternaire. Heu... quoi ça ? Ne bougez pas, j'explique.

Nous avons tout d'abord la concaténation. Nous avons juste au-dessus de cette ligne une déclaration et une affectation avec `String s = "blabla"`. On créé la variable `s` de type `String` et on lui affecte une valeur, la chaîne `blabla`. Avec `s +=` on ajoute simplement ce qui suit à la chaîne existante. On concatène la nouvelle chaîne et l'ancienne dans `s`.

Nous arrivons maintenant à l'opérateur ternaire, monstre pour certains, merveille pour d'autres (personnellement je le déteste, il est uniquement présent ici parce qu'il se trouve dans l'exemple original). La syntaxe est la suivante : `(test)?si_vrai:sinon`. Dans le cas présent, on teste si `(val)` est VRAI. Rappelez-vous, en C/C++ tout ce qui n'est pas 0 est VRAI. Ainsi, si `val` est 1 (VRAI) c'est `"haut"` qui est utilisé, et si `val` est 0 (FAUX), c'est `"bas"`.

L'opérateur ternaire permet d'éviter d'utiliser quelque chose comme ceci :

```
if(val)
  s += "haut";
else
  s += "bas";
```

Il est donc très intéressant pour gagner en concision, mais d'un autre côté, on perd en lisibilité, en particulier lorsqu'il se trouve en plein milieu des arguments d'une fonction ou si on a la brillante idée d'en imbriquer plusieurs :

```
String s=i%2==0?"a":i-3>0?"b":i+5!=0?"c":"d";
```

Vous voyez pourquoi je préfère ne pas trop utiliser l'opérateur ternaire ? C'est simplement pour éviter de me détester quelques semaines après, en cherchant à comprendre mon propre code.

PROGRAMMATION ET ESSAIS

Avant toutes choses, précisons que le fait de charger le croquis va supprimer le code présent dans la mémoire flash du module. Ceci signifie que l'utilisation « classique » du module avec le firmware d'origine et les commandes AT ne sera plus possible. Ce n'est pas une opération irréversible, mais la réinstallation du firmware d'origine est relativement pénible et loin d'être aussi aisée que la programmation via l'environnement Arduino. Vous êtes prévenu.

On n'oubliera pas, également, de relier une résistance et une led à GPIO2 afin de pouvoir vérifier que tout fonctionne.

La programmation du module ESP8266 se limite à une connexion correcte à l'adaptateur USB/série. On démarrera ensuite le module (par connexion de l'alimentation ou reset) tout en maintenant GPIO0 à la masse. Il suffira alors de sélectionner la bonne carte et le bon port dans l'environnement Arduino avant d'ouvrir le croquis exemple et le compléter des paramètres Wifi adéquats. Un « téléversement » du croquis provoquera la compilation et le chargement du code en flash.

En cas d'erreur, il faut vérifier :

- la bonne connexion des lignes RX et TX, certains adaptateurs possèdent un marquage inversé ;

coromonadalix

Chargement du croquis compilé dans le module ESP8266 démarré en mode programmation, avec affichage des « résultats détaillés » activé dans les préférences de l'environnement Arduino.

- que GPIO0 est effectivement connecté à la masse au démarrage du module ;
- que l'alimentation est suffisante en termes de courant fourni ;
- que GPIO15 (parfois appelé MTDO), s'il est présent sur votre module, est à la masse (ceci n'a pas été testé pour l'article, les modules autres que chinois à 8 broches n'étant pas arrivés à temps).

Le chargement du code en mémoire flash prend un certain temps en raison de sa taille (quelques 230 Ko) et se passe en deux fois. Si le chargement débute, mais débouche sur une erreur, c'est signe que les connexions sont correctes, mais que, soit l'alimentation ne suit pas, soit il y a un problème d'interférence. Sortez le multimètre, vérifiez, raccourcissez les fils et éventuellement ajoutez un condensateur de quelques dizaines de microfarads entre Vcc et la masse.

Une fois le code chargé, il ne vous reste plus qu'à démarrer le module sans entrer dans le mode programmation et de suivre le déroulement des opérations avec le moniteur série de l'IDE Arduino. Si vous voyez juste une série de points apparaître, c'est que le code reste coincé dans la boucle **`while`** dépendante du résultat retourné par **`WiFi.status()`**. Dans ce cas, vérifiez les paramètres Wifi (SSID et mot de passe) dans votre croquis et éventuellement la configuration de votre point d'accès. Selon la configuration en place, le module peut parfaitement se connecter en Wifi (WPA, WEP, etc.), mais ne pas recevoir d'adresse IP via DHCP, car son adresse matérielle (adresse MAC) n'est pas explicitement autorisée. Vous pouvez ajouter le code suivant juste après la ligne **`WiFi.begin(ssid, password)`** dans votre croquis pour afficher l'adresse MAC du module dans le moniteur série (oui, j'ai utilisé un opérateur ternaire) :

```
if(WiFi.macAddress(mac) != 0) {
  for(int i=0; i<WL_MAC_ADDR_LENGTH; i++) {
    Serial.print(mac[i],HEX);
    Serial.print((i < WL_MAC_ADDR_LENGTH-1) ? ":" : "\n");
  }
}
```

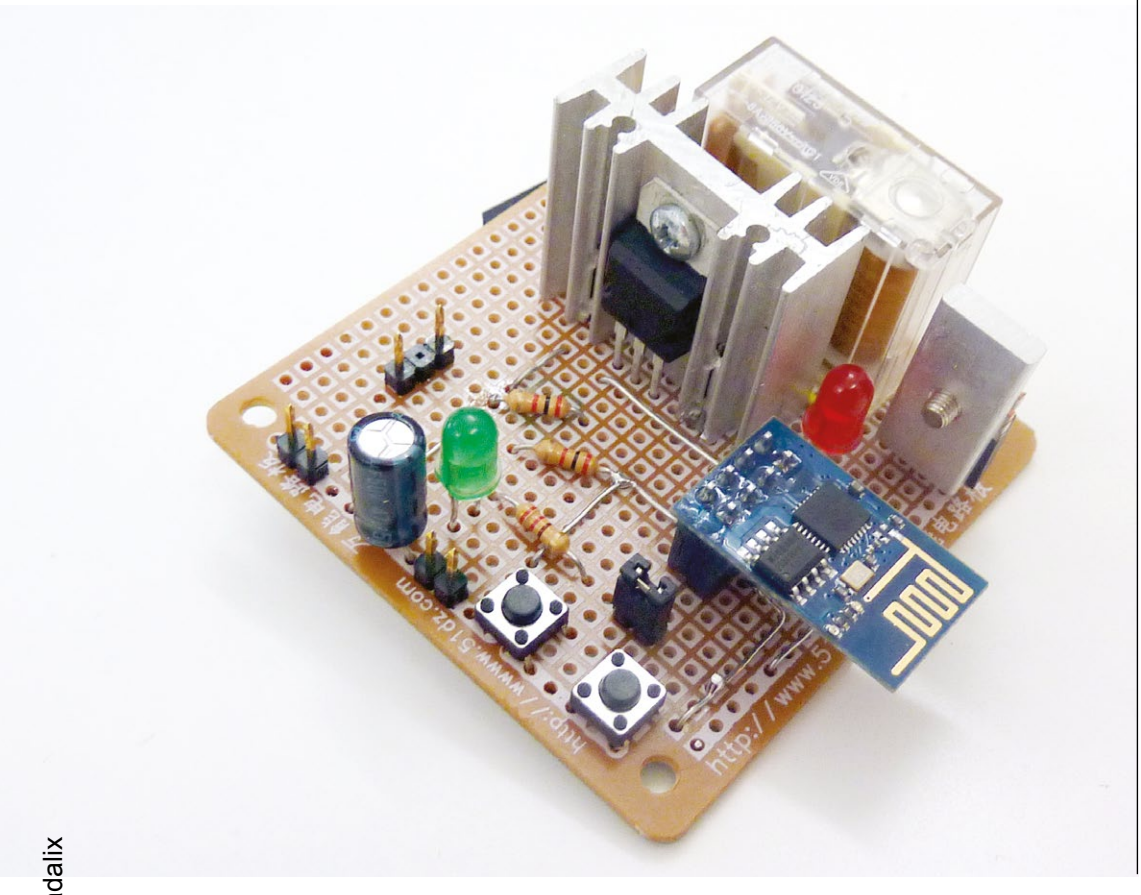
Si tout semble se passer correctement et que le moniteur série affiche une adresse IP, vous pouvez utiliser un navigateur quelconque pour visiter le serveur web avec, par exemple, **http://192.168.10.24/gpio/1**. Vous devez non seulement obtenir une réponse, mais, en plus, la led doit s'allumer. Inversement, elle doit s'éteindre en visitant **http://192.168.10.24/gpio/0**. Ça marche ? Bravo, vous avez créé un interrupteur Wifi !

Ce miraculeux et économique résultat conclut le présent article, mais nous reviendrons très certainement sur le sujet des modules ESP8266 (sans doute lorsque j'aurai reçu mes

modules avec plus de broches). Il est en effet possible de les utiliser de bien des façons, à commencer par le mode « standard » en compagnie d'une carte Arduino. Une autre voie intéressante est celle du projet NodeMCU, un autre firmware alternatif permettant une programmation dans le sympathique langage Lua.

En attendant, je vous recommande très fortement l'exploration des différents croquis exemples livrés avec le support Arduino ESP8266. Nous n'avons ici traité que de la partie accès client Wifi et serveur web, mais un module peut également faire office de point d'accès ou encore supporter bien d'autres protocoles comme mDNS, Telnet ou NTP. Sur la base de ces exemples relativement bien commentés, il est possible d'envisager de nombreux projets aussi bien en termes de contrôle à distance que de collecte d'informations ou d'affichage. **DB**

Lorsque le module démarre ou est réinitialisé alors que le moniteur est en marche, celui-ci envoie des données avant que notre croquis ne soit exécuté. Ceci peut être pris pour une erreur la première fois, mais c'est parfaitement normal. Notez que cette capture a été faite alors que le module était connecté sur une platine à essais avec des câbles un peu longs est des résistances de rappel de 10 Kohms. Là, je viens juste de m'asseoir. Résultat : 3 reset...



Voici un montage expérimental autour du module ESP8266. On retrouve les deux boutons pour le reset et le mode programmation, les résistances de rappel, un régulateur LD1117V33 sur un gros radiateur permettant d'obtenir 3,3V à partir du 12V en entrée, un relais 12V juste derrière contrôlé par le MOSFET à sa droite (avec un petit radiateur). L'ensemble forme un interrupteur 230V à relais commandé en Wifi. Notez que ceci est totalement expérimental, la version finale que j'utiliserai sur le terrain (mon appartement) utilisera un optocoupleur en plus, permettant de totalement isoler la partie haute tension de la partie Wifi.

CALIBREZ VOTRE RÉCEPTEUR RTL SDR

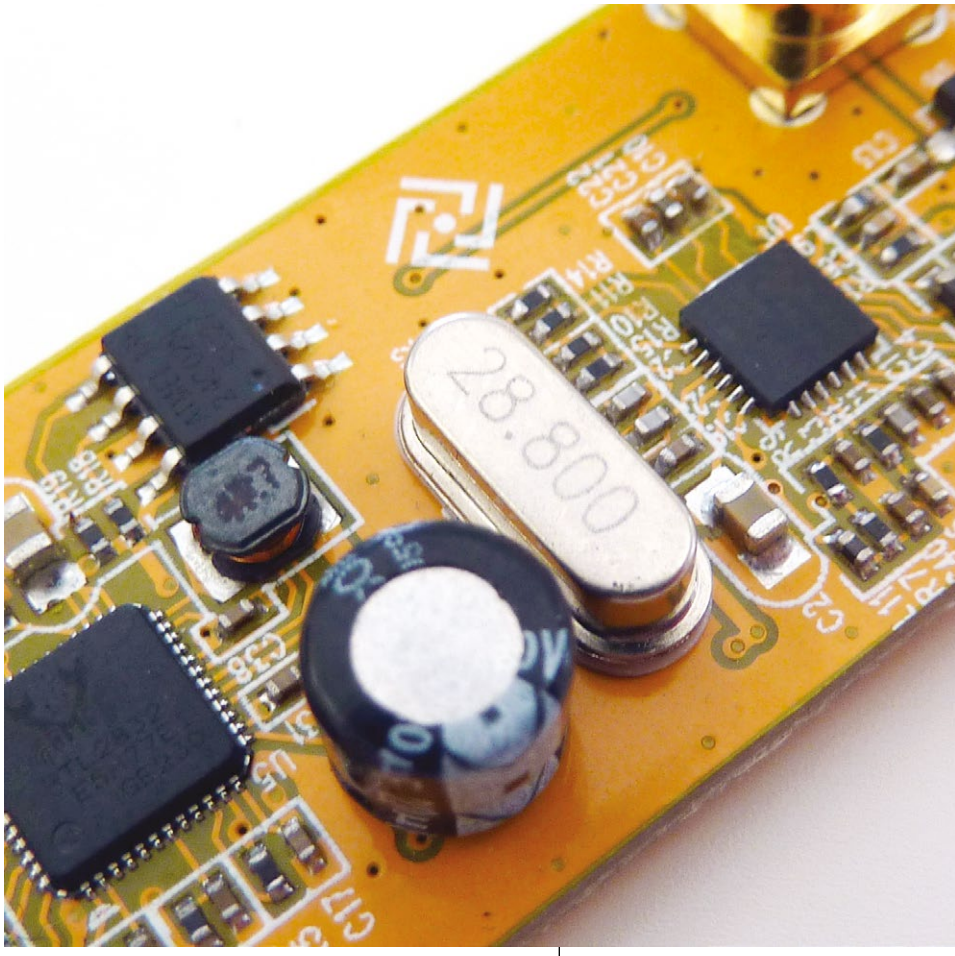
Denis Bodor

L'utilisation d'un récepteur DVB-T pour faire de la radio logicielle est une opportunité fantastique. On en perd presque de vue que techniquement le matériel n'est pas initialement destiné à ce genre d'utilisation. Ainsi certaines caractéristiques de ces clés USB et les conséquences d'une fabrication en série à très grande échelle sont tolérables pour la réception TNT, mais pas vraiment pour des utilisations plus poussées. Pour une utilisation en SDR, il est donc impératif de calibrer votre matériel.

Soyons honnêtes, une clé USB à 20 euros permettant normalement la réception des émissions TNT ne peut pas raisonnablement concurrencer un véritable récepteur SDR comme le HackRFOne ou encore le coûteux matériel vendu par Ettus Research. Pourtant ces constituants reposent sur la même architecture : un tuner, un oscillateur local et une puce de conversion analogique/numérique. Ce ne sera sans doute pas une surprise si j'affirme que ces matériels peu chers sont un assemblage de composants qui font leur travail, mais pas vraiment plus. Les composants ne sont pas de la plus grande qualité, juste les plus rentables pour le constructeur.

Le tuner (E4000, R820T, etc.) et le démodulateur/convertisseur RTL2832U ne posent généralement pas de problème, après tout il s'agit de circuits intégrés parfaitement normaux au comportement totalement prédictible (sauf bug ou composant défectueux). L'oscillateur local en revanche c'est une autre histoire. Dans votre récepteur RTL-SDR se trouve un oscillateur à quartz de 28,8 MHz duquel dérivent toutes les autres fréquences utilisées par le périphérique.

Le problème qui se pose alors est que ce quartz ne résonne pas exactement à 28800000 hertz, mais à une fréquence sensiblement différente. C'est vrai pour tous les quartz, de celui de votre montre à celui cadençant votre carte Arduino ou Raspberry Pi. Lors de sa fabrication, ce composant doit répondre à un certain nombre de critères pour passer le contrôle qualité et quitter l'usine. Parmi eux nous avons une tolérance dans la fréquence de résonance, exprimée en PPM pour « Parts Par Million ». Les quartz standards bas de gamme qu'on trouve un peu partout, ont généralement une tolérance de l'ordre de +/-100 PPM (+/- 10 PPM dans le meilleur des cas).



Le fameux oscillateur à quartz à 28,8 Mhz présent dans le récepteur DVB-T, ici à base de R820T. Remarquez juste à gauche, au-dessus du condensateur, on distingue la bobine faisant partie du circuit d'alimentation à découpage, source de bruit. Certains utilisateurs changent le quartz par un TCXO et en profitent pour remplacer le circuit d'alimentation par une série de diodes.

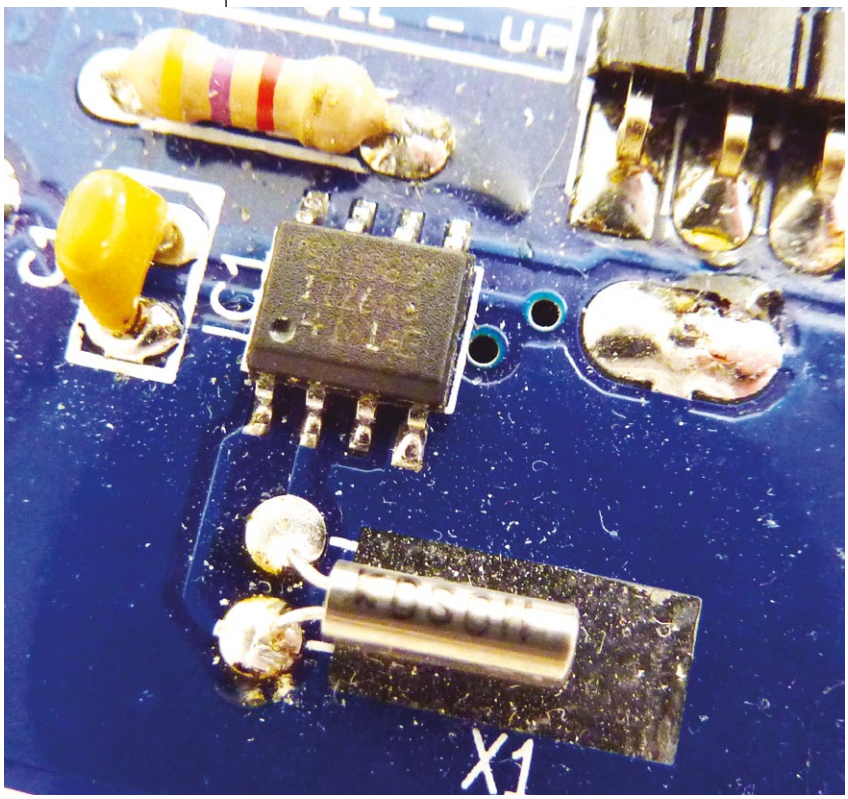
Les PPM fonctionnent comme les pourcentages à la différence qu'il s'agit d'une valeur sur un million et non sur cent. Avec un oscillateur à quartz peu coûteux à +/-100 PPM, ceci signifie qu'un écart par rapport à la fréquence nominale sera de 28800000/1000000×100, soit plus ou moins 2880 Hz. Ainsi, par définition, l'oscillateur

coromonadalix

soudé dans votre clé DVB-T peut avoir, dans les faits, une fréquence de résonance située n'importe où entre 28797120 Hz et 28802880 Hz. Pour certaines applications autres que de regarder des émissions lobotomisantes sur un petit écran, ceci est très problématique, on parle tout de même de presque 3 Khz (sur la fréquence de base, qui peut être multipliée en interne).

Ce n'est pas tout. En plus de la tolérance, il faut prendre en compte une caractéristique physique de ces composants : leur fréquence change en fonction de la température, et bien sûr, votre clé USB chauffe ce qui n'arrange rien à l'affaire. Normalement, lorsqu'une grande précision est requise, on utilise un **TCXO** *ou Temperature Compensated X Oscillator* (le X désignant « *X-tal* » pou « *crystal* »), un oscillateur à quartz avec compensation en température. Plus coûteux, ils ne sont pas parfaits, mais simplement beaucoup plus stables que leur déclinaison standard. Vous vous en doutez, « coûteux » rime avec « pas utilisé dans un périphérique à 20 euros ».

Les RTC ou horloges temps réel qu'on utilise pour garder son projet Arduino ou sa Raspberry Pi à l'heure reposent sur un oscillateur à quartz de 32768 Hz. Pourquoi ? Parce que 32768 c'est 2 puissance 15 et que cela permet à un programme de facilement compter des intervalles d'une seconde avec un compteur 16 bits.



PETIT LEXIQUE DES OSCILLATEURS :

- **XO (*X-tal Oscillator*) : entre +/- 10 et 100 PPM ;**
- **TXCO (*Temperature Compensated X-tal Oscillator*) : +/- 1 PPM ;**
- **OCXO (*Oven Controlled X-tal Oscillator*) : +/- 0,1 à 0,01 PPM, intègrent un système de contrôle de la température ;**
- **DTCXO (*Digital Temperature Compensated X-tal Oscillator*) : +/- 0,1 à 0,01 PPM, embarquent un microcontrôleur et des données de calibration ;**
- **RbXO (*Rubidium X-tal Oscillator*) : +/- 0,001 PPM, coûteux, encombrants et gourmands en énergie, ils forment la base des horloges atomiques commerciales. Une déclinaison plus précise encore repose sur le césium 133 en lieu et place du Rubidium 87, ces horloges hors de prix sont utilisées comme références standards.**

Notez qu'on parle ici principalement de dérive en fonction de la température et non de la précision initiale fixée par le fabricant et de la dérive en fonction du vieillissement du composant.

coromonadalix

Notez que des versions spécifiques de récepteurs RTL-SDR existent, comme le *NooElec NESDR Mini 2+*. Celui-ci est proche d'un récepteur DVB-T standard (RTL2832 + tuner R820T2), mais intègre un TCXO à 0,5 PPM. Le matériel NooElec vous coûtera de ce fait deux fois plus cher qu'un récepteur bas de gamme. Enfin, précisons qu'un certain nombre d'utilisateurs apportent des modifications au matériel en remplaçant par exemple l'oscillateur par un TCXO ou en modifiant le circuit d'alimentation (remplaçant l'alimentation à découpage « bruyante » par une série de diodes). Ceci nécessite une certaine dextérité et implique bien entendu le risque de détruire le matériel.

Enfin, comme si cela ne suffisait pas, la position du quartz est également importante, car la gravité terrestre a un impact sur son fonctionnement. Plus exactement, l'accélération a un impact, ce qui vaut donc pour la gravité, mais aussi pour les vibrations.

La bonne nouvelle, il en faut bien une après ces explications, est le fait qu'une fois le matériel arrivé à une température stable et le décalage en fréquence connue, celui-ci est linéaire. En d'autres termes, une valeur unique permet de compenser ce décalage et dans la plupart des logiciels celui-ci peut être spécifié par l'utilisateur. Il sera cependant propre à chaque périphérique et devra avant tout être mesuré.

1. CALIBRER SON RÉCEPTEUR

Dans les faits, les symptômes de ces erreurs sur la fréquence de l'oscillateur local sont simples : en réglant votre logiciel sur une certaine fréquence, celle-ci ne correspond pas exactement à la fréquence sur laquelle est effectivement calé votre récepteur.

Les outils en ligne de commandes RTL-SDR d'Osmocom qui permettent d'utiliser les clés DVB-T comprennent un outil de test qui vous permet d'estimer le décalage et donc de connaître la valeur PPM à renseigner dans la plupart des logiciels SDR. Pour procéder au test, il suffit d'utiliser :

```
% rtl_test -p
Found 1 device(s):
  0:  Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 1.4 2.7
  3.7 7.7 8.7 12.5 14.4 15.7 16.6 19.7 20.7
  22.9 25.4 28.0 29.7 32.8 33.8 36.4 37.2
  38.6 40.2 42.1 43.4 43.9 44.5 48.0 49.6
Sampling at 2048000 S/s.
Reporting PPM error measurement every 10 seconds...
Press ^C after a few minutes.
Reading samples in async mode...
real sample rate: 2048390 current PPM: 191 cumulative PPM: 191
real sample rate: 2048247 current PPM: 121 cumulative PPM: 154
real sample rate: 2048299 current PPM: 146 cumulative PPM: 151
real sample rate: 2048366 current PPM: 179 cumulative PPM: 159
real sample rate: 2048277 current PPM: 136 cumulative PPM: 154
[...]
real sample rate: 2048347 current PPM: 169 cumulative PPM: 152
real sample rate: 2048313 current PPM: 153 cumulative PPM: 152
real sample rate: 2048349 current PPM: 170 cumulative PPM: 152
real sample rate: 2048321 current PPM: 157 cumulative PPM: 152
real sample rate: 2048266 current PPM: 130 cumulative PPM: 152
^CSignal caught, exiting!
```

coromonadalix

Exemple typique de dérive d'horloge entre un four micro-ondes (en haut) et un autre équipement électroménager comme un four standard (en bas). La plupart des fours micro-ondes n'intègrent pas de RTC ou d'oscillateur, mais utilisent la fréquence de l'alimentation secteur (50 Hz) comme base de temps. Cette fréquence n'étant pas absolument stable (+/- 0,5 Hz), ceci implique des dérives importantes sur plusieurs semaines.

Il sera nécessaire de laisser fonctionner le test pendant une période importante pour que le résultat, *cumulative PPM*, soit le plus juste possible. On le voit au cours du processus, la moyenne cumulée se stabilise finalement autour d'une valeur, ici 152, propre à chaque périphérique. Cependant, dans la plupart des cas, la valeur retournée ne sera pas totalement exacte et ceci ne constitue qu'une solution de fortune.

2. CALIBRER SUR LA BASE D'UNE FRÉQUENCE CONNUE

Une autre solution consiste à trouver une source ayant une fréquence précise, stable et définie puis de régler son récepteur sur cette dernière. Comme la source est connue et utilise une fréquence avec un oscillateur bien plus précis que celui embarqué dans un récepteur DVB-T, il est aisé d'en déduire le décalage.

Ceci est bien entendu possible manuellement en utilisant, par exemple, un répéteur APRS local. Il suffit de régler la réception sur 144,8000 Mhz et de mesurer le décalage. Nous sommes dans la bande des 2 mètres (144 à 146 MHz) et ceci suppose donc une antenne adaptée pour recevoir un signal clair. Antenne qui n'est généralement pas celle livrée avec le périphérique (on parle par exemple ici d'antennes quart d'onde d'environ 50 cm de long). Mais il y a une méthode bien plus facile et automatisée, reposant sur des émetteurs puissants et faciles à trouver puisqu'ils sont tout autour de nous : les « antennes » GSM.

La téléphonie est un monde complexe et riche en versions, en standards, en générations et en désignations commerciales. GSM, LTE, UMTS, 2G, 3G, 4G, HSPA, GPRS, EDGE, CMDA... les désignations ne manquent pas et sont généralement utilisées un peu aléatoirement par le grand public et les brochures commerciales des fournisseurs de téléphonie mobile. L'outil dont nous allons parler peut utiliser les bandes de fréquences suivantes :

- GSM-850 (869,2 à 893,8 Mhz) : inutilisable en Europe ;
- GSM-R ou R-GSM-900 (921,0 à 960,0 Mhz) : utilisée pour les applications ferroviaires ;
- GSM-900 ou P-GSM-900 (935,0 à 960,0 Mhz) : la bande la plus utilisée chez nous ;

coromonadalix

DÉCOUVREZ NOS NOUVELLES OFFRES D'ABONNEMENTS !

PRO OU PARTICULIER = CONNECTEZ-VOUS SUR :

www.ed-diamond.com

LES COUPLAGES PAR SUPPORT :

VERSION PAPIER

Retrouvez votre magazine favori en papier dans votre boîte à lettres !

VERSION PDF

Envie de lire votre magazine sur votre tablette ou votre ordinateur ?

Sélectionnez votre offre dans la grille au verso et renvoyez ce document complet à l'adresse ci-dessous !

Voici mes coordonnées postales :	
Société :	
Nom :	
Prénom :	
Adresse :	
Code Postal :	
Ville :	
Pays :	
Téléphone :	
E-mail :	

coromonadalix

❑ Je souhaite recevoir les offres promotionnelles et newsletters des Éditions Diamond.

❑ Je souhaite recevoir les offres promotionnelles des partenaires des Éditions Diamond.

En envoyant ce bon de commande, je reconnais avoir pris connaissance des conditions générales de vente des Éditions Diamond à l'adresse internet suivante : **boutique.ed-diamond.com/content/3-conditions-generales-de-ventes** et reconnais que ces conditions de vente me sont opposables.

HACKABLE
MAGAZINE

Édité par Les Éditions Diamond
Service des Abonnements
B.P. 20142 - 67603 Sélestat Cedex
Tél. : + 33 (0) 3 67 10 00 20
Fax : + 33 (0) 3 67 10 00 21

Vos remarques :

.....

.....

VOICI TOUTES LES OFFRES COUPLÉES AVEC HACKABLE !
POUR LE PARTICULIER ET LE PROFESSIONNEL ...

Prix TTC en Euros / France Métropolitaine

CHOISISSEZ VOTRE OFFRE !

Prix en Euros / France Métropolitaine

Offre	SUPPORT ABONNEMENT	PAPIER Réf	PAPIER Tarif TTC	PAPIER + PDF (PDF 1 lecteur) Réf	PAPIER + PDF Tarif TTC	PAPIER + BASE DOCUMENTAIRE (1 connexion BD) Réf	PAPIER + BASE DOCUMENTAIRE Tarif TTC	PAPIER + PDF + BASE DOCUMENTAIRE (PDF 1 lecteur + 1 connexion BD) Réf	PAPIER + PDF + BASE DOCUMENTAIRE Tarif TTC
HK	6n° HK*	☐ HK1	39,-	☐ HK12	58,-				
LES COUPLAGES « EMBARQUÉ »									
D	6n° HK* + 4n° OS	☐ D1	65,-	☐ D12	98,-	☐ D13	85,-*	☐ D123	118,-*
E	6n° HK* + 4n° OS + 6n° MISC	☐ E1	105,-	☐ E12	158,-	☐ E13	179,-*	☐ E123	232,-*
E+	6n° HK* + 4n° OS + 6n° MISC + 2n° HS	☐ E+1	119,-	☐ E+12	179,-	☐ E+13	193,-*	☐ E+123	253,-*
F	6n° HK* + 4n° OS + 11n° GLMF	☐ F1	125,-	☐ F12	188,-	☐ F13	229,-*	☐ F123	292,-*
F+	6n° HK* + 4n° OS + 11n° GLMF + 6n° HS	☐ F+1	183,-	☐ F+12	275,-	☐ F+13	287,-*	☐ F+123	379,-*
G	6n° HK* + 4n° OS + 6n° LP	☐ G1	100,-	☐ G12	150,-	☐ G13	164,- *	☐ G123	214,-*
G+	6n° HK* + 4n° OS + 6n° LP + 3n° HS	☐ G+1	129,-	☐ G+12	194,-	☐ G+13	193,-*	☐ G+123	258,-*
LES COUPLAGES « GÉNÉRAUX »									
H	6n° HK* + 4n° OS + 6n° LP + 6n° MISC + 11n° GLMF	☐ H1	200,-	☐ H12	300,-	☐ H13	402,-*	☐ H123	499,-*
H+	6n° HK* + 4n° OS + 6n° LP + 3n° HS + 6n° MISC + 2n° HS + 11n° GLMF + 6n° HS	☐ H+1	301,-	☐ H+12	452,-	☐ H+13	493,-*	☐ H+123	639,-*

Les abréviations des offres sont les suivantes : LM = GNU/Linux Magazine France | HS = Hors-Série | LP = Linux Pratique | OS = Open Silicium | HC = Hackable

* HK : Attention : La base Documentaire de Hackable n'est pas incluse dans l'offre.

N'hésitez pas à consulter les détails des offres ci-dessus sur : www.ed-diamond.com !

- EGSM (925,0 à 960,0 Mhz) : c'est une bande GSM-900 étendue ("E") ;
- DCS ou DCS-1800 (1805,2 à 1879,8 Mhz) : un peu utilisée en Europe ;
- PCS ou PCS-1900 (1930,2 à 1989,8 Mhz) : c'est une bande US.

DCS et PCS sont inutilisables avec les clés USB à base de tuners R820T, GSM-850 n'est pas présent en Europe, GSM-R est spécifique au ferroviaire... Il ne reste donc que GSM900 et EGSM (le second étant compris dans le premier), mais nous avons une profusion d'antennes sur tout le territoire. Les données en elles-mêmes ne nous intéressent pas, mais ces émetteurs utilisent une fréquence (*downlink* ou lien descendant) très précise dont nous pouvons nous servir.

C'est précisément l'objet et la raison d'être de Kalibrate, un petit outil en ligne de commandes pour GNU/Linux, maintenu par Steve Markgraf sur la base d'un code provenant de **http://thre.at/kalibrate** (qui apparaît inaccessible désormais). Il ne semble pas exister de versions directement utilisables (compilées) pour GNU/Linux, Mac OS X ou Windows. Il faudra donc vous rabattre sur les sources du programme, chose qui peut être assez délicate, en particulier si vous n'êtes pas coutumier du système GNU/Linux.

3. À LA PRATIQUE !

Pour que la compilation fonctionne, il faudra que votre système dispose de tous les outils permettant de compiler des programmes et de configurer automatiquement les sources. Avec une distribution Ubuntu ou Debian, ceci passe par l'installation des paquets **`build-essential`**, **`gawk`**, **`libtool`**, **`pkg-config`** et **`automake`**. Ce n'est pas tout, si vous utilisez RTL-SDR, vous avez déjà très certainement installé les paquets **`librtlsdr0`** et **`rtl-sdr`**, mais ce ne sera pas suffisant. Ces deux paquets ne permettent que d'utiliser le périphérique et non de développer des programmes. Il vous faudra donc également installer le paquet **`librtlsdr-dev`**, mais aussi **`libfftw3-dev`**, **`libusb-1.0-0-dev`**, ainsi que **`git`** si vous voulez suivre la procédure utilisée ici.

Normalement, avec tout cela, vous devez être paré pour la suite et ne rencontrer aucun problème particulier. La première étape consiste à récupérer les sources de l'outil. Celles-ci sont disponibles sur GitHub, vous pouvez donc soit récupérer un fichier Zip, soit directement utiliser la commande **`git`** :

```
% git clone https://github.com/steve-m/kalibrate-rtl.git
Clonage dans 'kalibrate-rtl'...
remote: Counting objects: 85, done.
remote: Total 85 (delta 0), reused 0 (delta 0), pack-reused 85
Unpacking objects: 100% (85/85), done.
Vérification de la connectivité... fait.
```

Dès lors, vous devez trouver un répertoire **`kalibrate-rtl`** dans le répertoire courant, contenant tout le nécessaire. Il faut ensuite générer les scripts permettant de configurer les sources :

```
% cd kalibrate-rtl/
% ./bootstrap
configure.ac:34: installing './config.guess'
configure.ac:34: installing './config.sub'
configure.ac:4: installing './install-sh'
configure.ac:4: installing './missing'
Makefile.am: installing './INSTALL'
src/Makefile.am: installing './depcomp'
```

RASPBERRY PI

Si vous comptez procéder à ces manipulations sur une Pi, vous devrez impérativement utiliser un système Raspbian en *testing* (Jessie) et non en *stable* (Wheezy) ou du moins préciser que vous souhaitez utiliser des paquets provenant de la version *testing*. Pour cela, vous devez ajouter une ligne dans `/etc/apt/sources.list` :

```
deb http://archive.raspbian.org/raspbian jessie main
```

puis procéder à une mise à jour de la liste des paquets avec `sudo apt-get update`. Ceci fait, vous pourrez installer les paquets nécessaires avec `sudo apt-get install rtl-sdr librtlsdr-dev libfftw3-dev libusb-1.0-0-dev libtool automake`. L'installation de `rtl-sdr` provoquera l'installation de `librtlsdr0` qui installe un fichier `/etc/modprobe.d/rtl-sdr-blacklist.conf`. Celui-ci interdit au noyau de charger le pilote pour le récepteur DVB-T, car les outils RTL-SDR accèdent directement au périphérique USB sans utiliser de pilote (au contraire, le pilote bloquerait l'accès pas les outils).

Ce n'est pas tout. Par défaut, seul le super-utilisateur peut accéder ainsi au matériel et il serait très pénible de devoir systématiquement utiliser sudo. Pour régler ce problème, connectez votre clé USB à la Raspberry Pi et consultez les listes des périphériques USB avec :

```
$ lsusb
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 006: ID 0bda:2838 Realtek Semiconductor Corp. RTL2838 DVB-T
```

La dernière ligne affiche ici le récepteur DVB-T. Ce qui nous intéresse en particulier ce sont les valeurs `0bda:2838` correspondant au numéro de constructeur et au numéro de produit (*VendorID* et *ProductID*). Ceux-ci varient d'un modèle de récepteur à l'autre et peuvent donc être différents pour vous. Pour autoriser les utilisateurs du groupe `adm`, dont l'utilisateur par défaut `pi` fait partie, à utiliser le périphérique, nous devons ajouter un fichier `99-rtlsdr.rules` dans le répertoire `/etc./udev/rules.d`. Celui-ci contenant sur une ligne :

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="0bda", ATTRS{idProduct}=="2838",
GROUP="adm", MODE="0666", SYMLINK+="rtl_sdr"
```

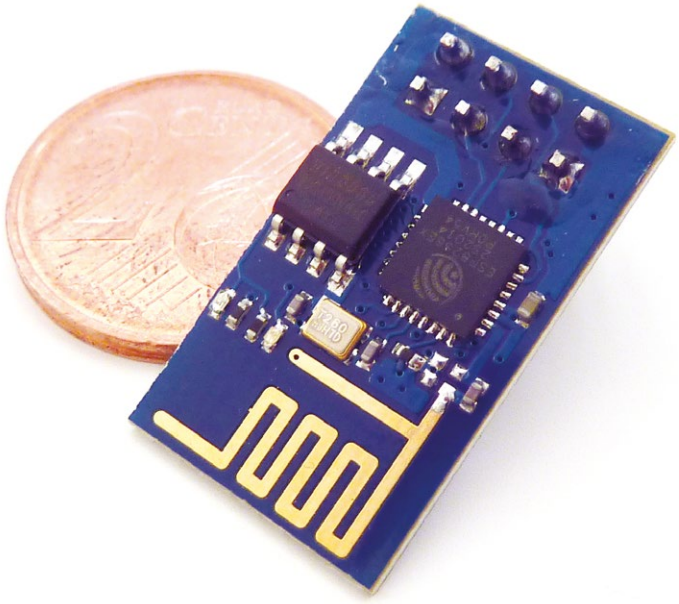
Notez les valeurs `0bda` et `2838` correspondant à la sortie affichée par `lsusb`. Il faudra enfin redémarrer le service chargé d'appliquer cette configuration avec `sudo service udev restart` et déconnecter/reconnecter le récepteur en USB. Vous êtes alors normalement prêt pour la suite. Pour vous assurer que tout fonctionne, vous pouvez utiliser `rtl_test` comme expliqué précédemment.

Précisons toutefois que même si l'utilisation d'outils simples comme Kalibrate est tout à fait possible, la SDR nécessite généralement des ressources importantes en termes de mémoire et de processeur que la Raspberry Pi peut ne pas être en mesure de fournir. Ceci ajouté aux petits problèmes concernant l'USB et l'utilisation de l'Ethernet sur USB fait de la Pi une carte qui n'est pas vraiment adaptée à certaines activités SDR. La situation est bien meilleure avec une Raspberry Pi 2 et son importante puissance de calcul, sans toutefois égaler un simple PC.

Ceci produit normalement, entre autres choses, un script **configure** qu'on lancera immédiatement. Ce script permet de vérifier le système et de trouver tous les éléments nécessaires à la construction de l'utilitaire :

```
% ./configure
[...]
checking pkg-config is at least version 0.9.0... yes
checking for FFTW3... yes
checking for LIBRTLSDR... yes
[...]
config.status: creating Makefile
config.status: creating src/Makefile
config.status: creating config.h
config.status: executing depfiles commands
```

Une longue série de lignes doit apparaître, chacune renseignant sur une vérification qui est faite. Normalement, le script devrait trouver tout ce qu'il cherche. Nous avons ici tronqué la sortie de la commande, car la majorité des éléments concerne le compilateur lui-même qui normalement ne pose aucun problème. Les deux éléments clés que le script cherche sont les fichiers de développement fournis par les paquets **librtlsdr-dev** et **libfftw3-dev**. S'il n'y a pas de message d'erreur, c'est très bon signe.



Les oscillateurs à quartz peuvent prendre bien des formes. Ici un module Wifi ESP8266 de chez ElectroDragon. Le quartz est le petit rectangle argenté sous les deux circuits intégrés (l'ESP8266 et l'EEPROM). Notez la taille du composant relativement à la pièce de 2 cts placée à l'arrière.

Dans le cas contraire, il faudra inspecter attentivement le message et essayer d'identifier ce qui manque. La méthode est simple, repérez de quoi parle le message d'erreur et vérifiez dans la liste des paquets que vous pouvez installer s'il y en a un correspondant et dont le nom se termine par **-dev** (comme paquet de développement). Si oui, installez-le et tentez à nouveau le **./configure**.

Il ne reste plus ensuite qu'à réellement compiler le tout pour obtenir la commande attendue. Là, un simple **make** suffira :

```
% make
[...]
make[2]: Rien à faire pour « all-am ».
make[2]: quittant le répertoire « /home/denis/RTL/0/kalibrate-rtl »
make[1]: quittant le répertoire « /home/denis/RTL/0/kalibrate-rtl »
```

coromonadalix

Là encore nous avons tronqué la sortie pour ne pas encombrer l'article de texte inintéressant. L'important est de ne pas avoir d'erreur et c'est très peu probable que passé l'étape du `./configure` vous en rencontriez. Après ce `make`, la commande `kal` est disponible dans le sous-répertoire `src/`. À ce stade, il vous est possible d'installer l'outil dans votre système avec `sudo make install`, mais je vous le déconseille fortement. En effet, ceci installera effectivement l'utilitaire, mais sans que le système de gestion de paquets n'en soit informé, ce qui peut conduire, par la suite à quelques problèmes.

Ici, nous avons un simple utilitaire qui fonctionnera parfaitement même sans être installé dans les répertoires dédiés. Il suffit de se placer dans le répertoire `src/` et de le lancer depuis cet endroit.

Notre première exécution consistera à trouver des émetteurs proches utilisant des fréquences dans la bande EGSM. On utilise alors la commande `kal` ainsi :

```
% cd src
% ./kal -e 152 -g 30 -s EGSM
Found 1 device(s):
  0:  Generic RTL2832U OEM
Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
Setting gain: 30.0 dB
kal: Scanning for E-GSM-900 base stations.
E-GSM-900:
  chan: 1 (935.2MHz + 14.457kHz)     power: 573600.49
  chan: 9 (936.8MHz + 14.707kHz)     power: 940341.77
  chan: 16 (938.2MHz + 14.360kHz)    power: 981102.79
  chan: 23 (939.6MHz + 14.034kHz)    power: 2724428.55
  chan: 35 (942.0MHz + 5.042kHz)     power: 1412362.51
  chan: 41 (943.2MHz - 5.264kHz)     power: 1902744.31
  chan: 86 (952.2MHz + 14.065kHz)    power: 639724.79
  chan: 102 (955.4MHz + 14.361kHz)   power: 545109.31
  chan: 975 (925.2MHz + 14.396kHz)   power: 1284105.72
  chan: 981 (926.4MHz + 14.368kHz)   power: 1280185.92
  chan: 990 (928.2MHz + 14.131kHz)   power: 933290.75
  chan: 994 (929.0MHz + 15.081kHz)   power: 1226449.46
  chan: 1005 (931.2MHz - 19.048kHz)  power: 306131.47
  chan: 1021 (934.4MHz + 5.085kHz)   power: 831216.39
  chan: 1022 (934.6MHz + 13.919kHz)  power: 1432177.49
```

Les options que nous utilisons permettent respectivement de spécifier (`-e 152`) la correction en PPM que nous avons obtenu avec `rtl_test`, le gain (`-g 30`) et la bande de fréquence dans laquelle chercher (`-s EGSM`). Le gain et la correction ne sont pas obligatoires. Ces options permettant simplement d'améliorer la réception et d'accélérer le processus. Après quelques minutes, nous voyons apparaître une liste de canaux avec leur numéro, leur fréquence et une indication de la puissance du signal.

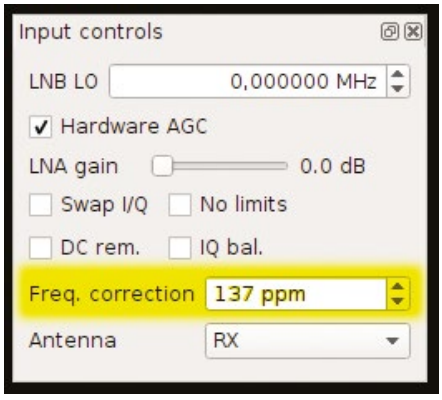
Pour utiliser une fréquence et obtenir notre valeur de décalage pour calibrer notre récepteur, il nous suffit de choisir un canal (avec **-c** suivi du numéro) de forte puissance et de laisser faire **kal** :

```
% ./kal -e 152 -g 30 -c 981
Found 1 device(s):
  0:  Generic RTL2832U OEM
Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
Setting gain: 30.0 dB
kal: Calculating clock frequency offset.
Using E-GSM-900 channel 981 (926.4MHz)
average         [min, max]   (range, stddev)
+ 14.384kHz  [14370, 14393]  (23, 5.819765)
overruns: 0
not found: 0
average absolute error: 136.474 ppm
```

La valeur attendue est donnée sur la dernière ligne et il est recommandé de réitérer la commande sur d'autres canaux pour s'assurer d'obtenir des valeurs similaires. Ici notre périphérique affiche en moyenne une déviation de +137 ppm (+ 0,0137 %). C'est moins que les +152 ppm rapportés par **rtl_test**, mais nous sommes bien au-delà des +/-100 PPM théoriques, sans doute en raison de la tolérance initiale s'additionnant au décalage en raison de la température (ou tout simplement parce que le quartz est vraiment de mauvaise qualité). Notez que cela fait tout de même, comme l'affiche **kal**, un écart de plus de 14 Khz sur 926,4 MHz !

4. QUE FAIRE AVEC CETTE VALEUR PPM ?

Une fois cette valeur connue, le reste dépend de votre logiciel SDR et quelque part dans ses préférences doit être fait mention d'une correction de fréquence, d'un *offset* ou d'une valeur à spécifier en PPM. Avec GQRX par exemple, ceci se trouve dans « *Input Controls* », sous le nom « *Freq. Correction* » :



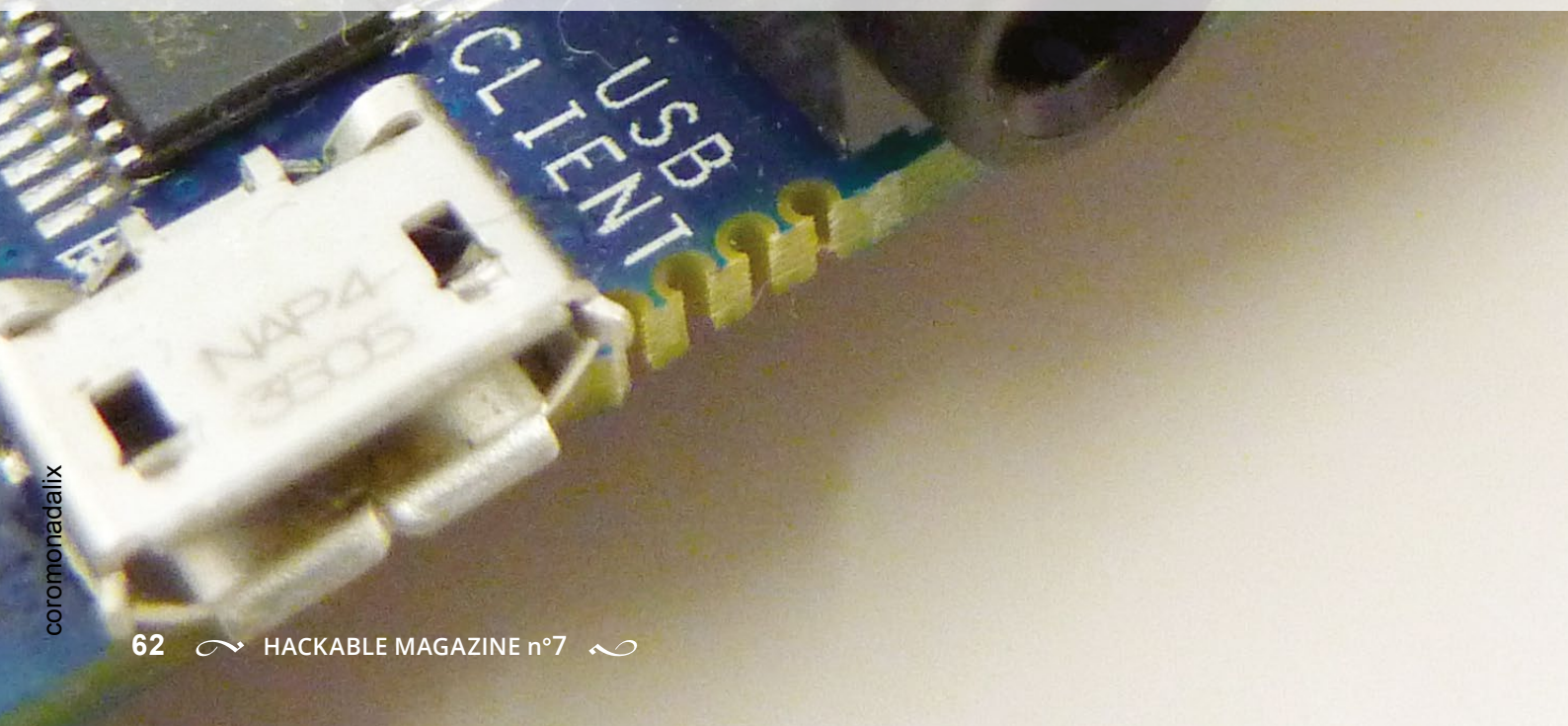
Les outils livrés avec le paquet **rtl-sdr** ainsi que la plupart des utilitaires en ligne de commandes disposent d'une option permettant de préciser la correction en PPM. Le plus souvent, il s'agit de **-p**, mais ce n'est pas une règle absolue. Mieux vaut consulter la documentation. **DB**

coromonadalix

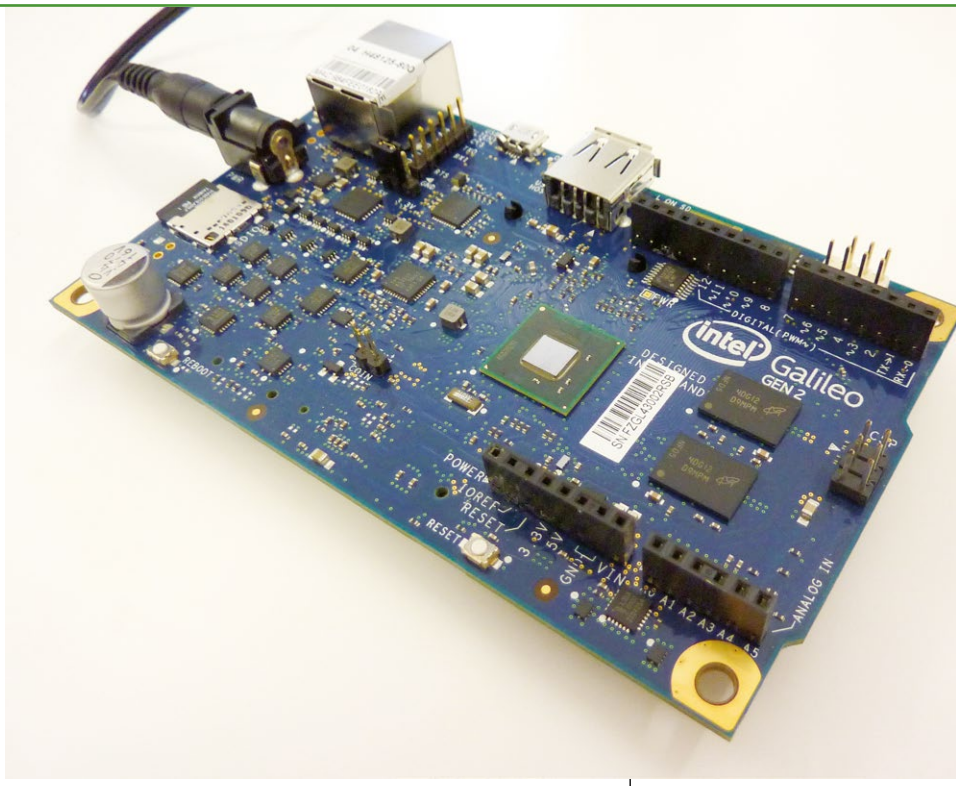
ESSAI DE WINDOWS SUR INTEL GALILEO

Denis Bodor

Windows 10 IoT Core Insider Preview est le fer de lance de la campagne Microsoft concernant l'utilité de l'entreprise dans le domaine des objets connectés (l'IoT). Les plateformes mises en avant sont la Raspberry Pi 2, la MinnowBoard MAX et la Intel Galileo. Microsoft semble bien décidé à obtenir une part du gâteau du marché des « makers » et autres bidouilleurs s'amusant à torturer d'innocentes cartes. Mais la société de Redmond peut-elle vraiment tirer son épingle du jeu ? Vérifions avec ce qui est actuellement disponible pour la carte Intel Galileo Gen 2.



coromonadalix



La carte Galileo Gen2 reprend le brochage standard d'une carte Arduino, mais la comparaison matérielle s'arrêtera là. Cette carte signée Intel intègre un processeur à 400 Mhz, 256 Mo de mémoire DDR3, un port Ethernet, un emplacement pour microSD, 8 Mo de flash, un port mini PCI Express et un port USB.

Le principe est toujours le même : des initiatives sont créées par des projets et ceci prend de l'ampleur jusqu'à arriver à un stade où les terminaisons nerveuses des poids lourds de l'industrie informatique commencent à s'agiter. Conscients de leur retard, ils mettent alors généralement les bouchées doubles pour se faire une place et montrer que, eux aussi, « en font partie ». Souvent cependant, lorsque la raison d'être du phénomène n'est pas pleinement comprise, ceci se traduit uniquement par la mise en œuvre de moyens considérables, en particulier en termes de communication. Un peu comme un voisin très riche qui se rend compte que vos soirées Doctor Who attirent du monde et qui tente de faire pareil en dépensant des milles et des cents, mais n'a finalement pas une once de fanboy en lui et ne fera jamais la différence entre les Cybermens et les Daleks. Au final, après beaucoup de bruit et de grosses dépenses, tout le monde se rend compte qu'il n'est « pas vraiment dans le truc » et n'a vraiment pas compris grand-chose...

Aujourd'hui, le « mouvement makers » a pris une ampleur attrayante et de plus en plus d'utilisateurs s'intéressent à ces plateformes que sont les Arduino, les Raspberry Pi, les BeagleBone, les cartes mbed ou encore les Launchpad de TI. Les voisins avec de gros moyens sont Intel et Microsoft. La question est donc de savoir s'ils ont ou non compris le sens profond du phénomène et sont ou non « dans le truc ».

1. EULA OU « ILS N'ONT TOUJOURS PAS COMPRIS »

Vouloir essayer une plateforme logicielle Microsoft est toujours une aventure épique, énervante et affligeante. Lorsqu'on utilise depuis des années des systèmes qui reposent sur une certaine philosophie et une grande ouverture, la « réalité Microsoft » vous arrive comme un mur de brique en pleine figure.

Pour télécharger « *Microsoft Windows for Intel Galileo* » et se faire une petite idée de la vision technique de la firme de Redmond il faut, avant toutes choses, posséder un compte Microsoft Connect et de ce fait fournir une masse d'informations personnelles conséquente. Certes, ceci n'est ni nouveau ni rare dans le monde du logiciel propriétaire, mais est très désagréable lorsqu'on est habitué aux mœurs des univers Raspberry Pi et Arduino (hé oui, on ne m'a jamais demandé qui je suis en téléchargeant sur ces sites).

Mais ce n'est pas tout, non content d'être en possession d'informations sur qui s'intéresse à leur petit bébé, Microsoft vous demande également d'accepter explicitement leurs conditions. Parmi lesquelles :

- « DURÉE. Ce contrat est conclu pour une durée de jusqu'au 30/09/2014 (jour/mois/année). » : de quoi ? En dehors du fait que ceci n'est absolument pas français (mais c'est la version anglaise qui fait

coromonadalix

foi), je suis invité à accepter un contrat d'ores et déjà caduc (mes manipulations sont effectuées ce jour, le 9 mai 2015) ;

- « RETOUR D'INFORMATIONS. Si vous faites part de vos observations concernant le logiciel à Microsoft, vous lui concédez gracieusement le droit d'utiliser, de partager et de commercialiser vos observations de quelque façon que ce soit et à toute fin » : ce que je traduis personnellement en « si vous trouvez un bug, merci, on se réserve le droit de faire de l'argent avec votre travail gracieux » ;
- « Vous n'êtes pas autorisé à [...] contourner les limitations techniques du logiciel, reconstituer la logique du logiciel [...], publier le logiciel pour que d'autres le copient [...] » : là nous avons très précisément l'inverse d'une licence open source ou de logiciel libre ;
- « Vous pouvez obtenir de Microsoft et de ses fournisseurs une indemnisation en cas de dommages directs limités uniquement à hauteur de 5,00 $ US » : celle-ci est carrément comique ;
- « *you may not[...]disclose the results of any benchmark tests of the software to any third party without Microsoft's prior written approval* » : Oui, vous avez bien compris, vous n'avez pas le droit de diffuser des résultats de tests de performance à quiconque sans l'accord de Microsoft !

Les deux cartes Intel Galileo côte à côte. En haut l'originale et en bas la Gen2 (pour génération 2) qui se différencie par un certain nombre de corrections et d'améliorations.

Bref, ce n'est qu'après avoir accepté ces règles drastiques que vous pourrez obtenir les fichiers `apply-BootMedia.cmd` et `9600.16384.x86fre.winblue_rtm_iotbuild.150309-0310_galileo_v2.wim` en passant par **https://dev.windows.com/en-us/iot.**

Comprenez bien que je n'ai strictement rien contre l'initiative de Microsoft, un peu plus d'ouverture sur les utilisateurs ne leur fera pas de mal (en principe). Je déplore simplement leur manque total de compréhension de l'écosystème malgré le fait que la page même du site principal parle de « *Maker community* », de partage (entre contributeurs) et de l'influence des utilisateurs sur les prochaines versions disponibles. Ce mouvement et cet intérêt des utilisateurs pour des plateformes comme la Raspberry Pi et l'Arduino découlent directement d'un sentiment d'ouverture et de liberté. L'ensemble s'est construit autour de cette philosophie, qui n'est pas étrangère à la popularité d'autres projets comme

coromonadalix

Firefox, GNU/Linux, LibreOffice ou Blender. J'ai clairement le sentiment que Microsoft, tout comme Intel, tente désespérément de s'accrocher à un train, qui est parti bien avant eux et qui est propulsé par des concepts qu'ils ne comprennent pas.

2. PRÉPARATION DE LA CARTE SD

Une fois les fichiers obtenus et placés dans votre répertoire de téléchargement, vous devrez, tenez-vous bien, invoquer l'interpréteur de commandes `cmd`. Non vous ne rêvez pas, alors même que GNU/Linux a la réputation d'être un système peu rassurant pour l'utilisateur débutant, la préparation d'une carte SD ou microSD pour la Raspberry Pi passe, sous Windows, dans le pire des cas, par l'application *Win32 Disk Imager*. Une autre solution consiste à tout bonnement utiliser NOOBS et le gestionnaire de fichiers.

Certes il s'agit d'une des premières versions disponibles et j'essaie d'être le plus tolérant possible, mais c'est le monde à l'envers ! LE système réputé simple et convivial est celui-là même qui nécessite l'utilisation d'une ligne de commandes alors que pour préparer son support pour Raspbian, un outil graphique existe. Est-ce une blague de mauvais goût ? N'était-il donc pas possible de tout simplement utiliser le même principe et format d'image et donc *Win32 Disk Imager* ? Était-il donc vraiment nécessaire de tout faire reposer sur le format WIM (*Windows Imaging Format*) qui n'est, dans les grandes lignes, qu'une archive 7zip un peu bricolée ? Microsoft n'a-t-il donc pensé ni aux actuels utilisateurs de Raspberry Pi, ni aux débutants ? Savez-vous que peu d'utilisateurs, même possédant une Raspberry Pi ou une carte Arduino, ont simplement connaissance du fait qu'une ligne de commandes est disponible dans Windows ?

Ce n'est pas fini, car une fois l'interpréteur lancé, il faut se rendre dans le bon répertoire après avoir bien repéré la lettre correspondant à la carte SD/microSD (de 16 Go ou plus !) et lancer le processus :

```
C:\Users\denis> cd /d %USERPROFILE%\Downloads

C:\Users\denis\Downloads> apply-BootMedia.
cmd -destination E: -image 9600.16384.x86fre.
winblue_rtm_iotbuild.150309-0310_galileo_
v2.wim -hostname mygalileo -password admin
```

Ce qui génère un affichage abscons que même un utilisateur GNU/Linux de longue date comme moi trouvera absolument ignoble et absolument pas rassurant pour un débutant, même bidouilleur :

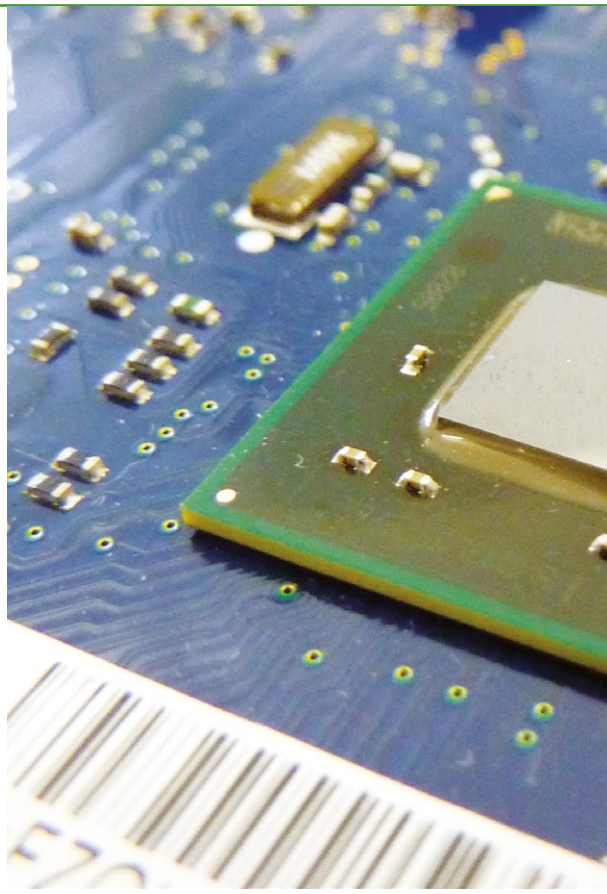
```
**** Temporarily changing time zone to 'Pacific
Standard Time'

**** Processing Arguments
**** Set-up work folder: C:\cygwin64\tmp\apply-
BootMedia-25581
**** Retrieveing C:\Users\denis\Downloads\9600.16384.
x86fre.winblue_rtm_iotbuild.150309-0310_galileo_v2.wim
****          to C:\Users\denis\AppData\Local\Temp\
apply-BootMedia-25581

Outil Gestion et maintenance des images de déploiement
Version : 6.3.9600.17031

Montage de l'image
L'opération a réussi.
**** Customizing image C:\Users\denis\AppData\Local\
Temp\apply-BootMedia-25581\9600.16384.x86fre.winblue_
rtm_iotbuild.150309-0310_galileo_v2.wim
****        mounted at C:\Users\denis\AppData\Local\
Temp\apply-BootMedia-25581\9600.16384.x86fre.winblue_
rtm_iotbuild.150309-0310_galileo_v2.wim.mount

Outil Gestion et maintenance des images de déploiement
Version : 6.3.9600.17031

Enregistrement de l'image
Démontage de l'image
L'opération a réussi.
```

```
**** Applying image C:\Users\denis\AppData\Local\Temp\
apply-BootMedia-25581\9600.16384.x86fre.winblue_rtm_
iotbuild.150309-0310_galileo_v2.wim
****             to E:\

Outil Gestion et maintenance des images de déploiement
Version : 6.3.9600.17031
Application de l'image

L'opération a réussi.
**** Mounting E:\\Windows\System32\config\SYSTEM
****       to HKEY_USERS\Galileo-25581-SYSTEM
**** Setting hostname to mygalileo
**** Restoring time zone to 'W. Europe Standard Time'
****
****   Successfully applied C:\Users\denis\
Downloads\9600.16384.x86fre.winblue_rtm_
iotbuild.150309-0310_galileo_v2.wim
****                    to E:\
****
****             hostname: mygalileo
****             timezone: Pacific Standard Time
****             Username: Administrator
****             Password: admin
****
**** Done.
```

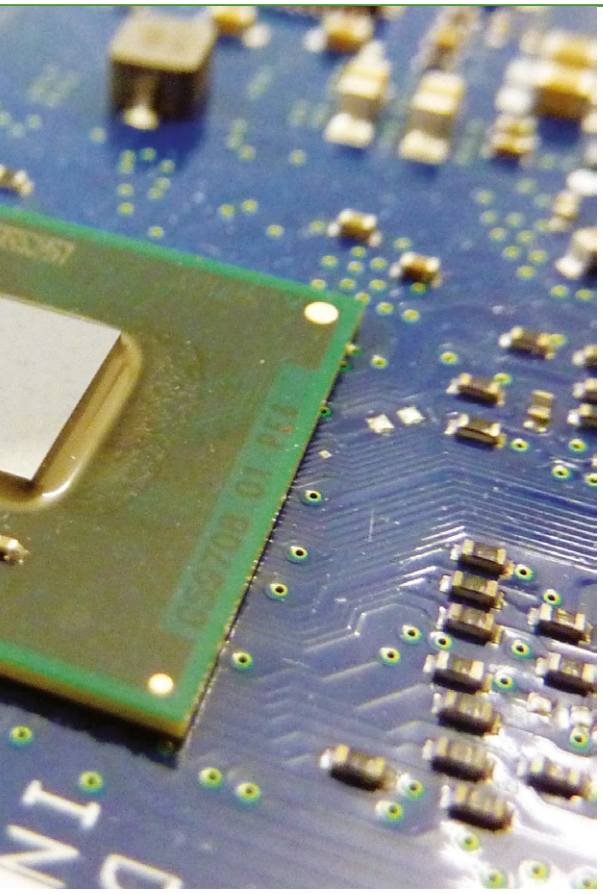
3. PREMIER DÉMARRAGE

En suivant la documentation officielle (**http://ms-iot.github.io/content/win8/SetupGalileo.htm**), l'étape suivante consiste à éjecter la carte de la machine Windows et à la placer dans la Galileo. Il faut ensuite relier cette dernière au PC à l'aide d'un câble réseau Ethernet. Vous n'avez qu'un port Ethernet sur votre machine et l'utilisez déjà pour l'accès Internet ? Achetez-vous un adaptateur USB/Ethernet, sinon tant pis pour vous. Ce n'est pas moi qui le dit c'est en gros la documentation officielle, en précisant « *Your computer will still have internet connectivity* » (« votre ordinateur aura encore de la connectivité Internet »), je suis curieux de savoir par quel miracle (ah oui, pour Microsoft le Wifi c'est la vie, désolé, mais pas pour moi).

Vous le remarquez, ma patience commence sérieusement à être entamée, mais je ne jette pas encore l'éponge. On met la carte sous tension (en connectant le bloc d'alimentation) et on attend. La documentation précise que Windows sur Galileo peut mettre jusqu'à 2 minutes à démarrer (sans commentaire), et qu'il faut surveiller la led d'activité de la microSD. Dès qu'elle arrête de clignoter pendant quelques secondes, la carte Galileo a démarré...



La documentation détaille ensuite qu'il faut utiliser Telnet pour accéder à la carte, en passant par « Start -> Run » (Démarrer, Exécuter) et en tapant `telnet mygalileo`. On respire un grand coup et on reste zen (hurler « mais quelle bande de débiles » fait peur aux collègues). Dans quel univers vivent les personnes de Microsoft (parce qu'il y a bien un « © 2015 Microsoft » au bas de la page) ayant produit cette documentation ? Non seulement le client Telnet n'est pas installé par défaut dans Windows 8 mais, en plus, quelqu'un a du oublier qu'il n'y a plus de menu « Démarrer » dans cette version du système (vous savez celui qui est installé en standard sur les PC neufs et qui fait suite à Windows 7 dont le support standard a pris fin le 13 janvier 2015) !



Le cœur de la Galileo est un processeur (ou SoC) Intel Quark X1000, monocœur, monothread, cadencé à 400 Mhz, utilisant un jeu d'instructions Pentium. Ce composant se rapproche plus du Pentium de votre jeunesse que du processeur ARM présent dans votre smartphone. On remarquera avec amusement qu'Intel, maintenant en guerre ouverte avec ARM a pourtant, fut un temps, fait partie du club des « fondeurs » ARM avec sa famille XScale, revendue en 2006 à Marvell.

Pour continuer l'exploration, il faut donc passer par le panneau de configuration, choisir ***Programmes***, puis ***Activer ou désactiver des fonctionnalités de Windows*** et enfin cocher ***Telnet*** dans la liste. Ah oui, zut, il faut Internet pour installer cet élément, si vous n'êtes pas connecté en Wifi, amusez-vous avec les câbles réseau. Enfin, depuis l'ignoble interface Metro, on cherche « cmd » (l'interpréteur de commandes) et dans la fenêtre noire on utilise `telnet mygalileo`. Un nom d'utilisateur est demandé, `Administrator`, puis le mot de passe `admin`. Et voilà ! Vous avez en main la carte Galileo et avez accès à Windows, enfin... à l'interpréteur de commandes Windows. En utilisant `dir` pour lister les fichiers, vous déchantez immédiatement devant la lenteur de la chose...
On remarquera également que la commande `ver` retourne `Version 6.3.9600` qui correspond à Windows 8 et non 6.4 ou 10.0 comme on peut s'y attendre pour un Windows 10. Une petite mention le précise sur la page « *Get Started* » : « *Galileo supports only the previous versions of Windows, not Windows 10 IoT Core* » (« Galileo ne supporte que la précédente version de Windows, non Windows 10 IoT Core »).

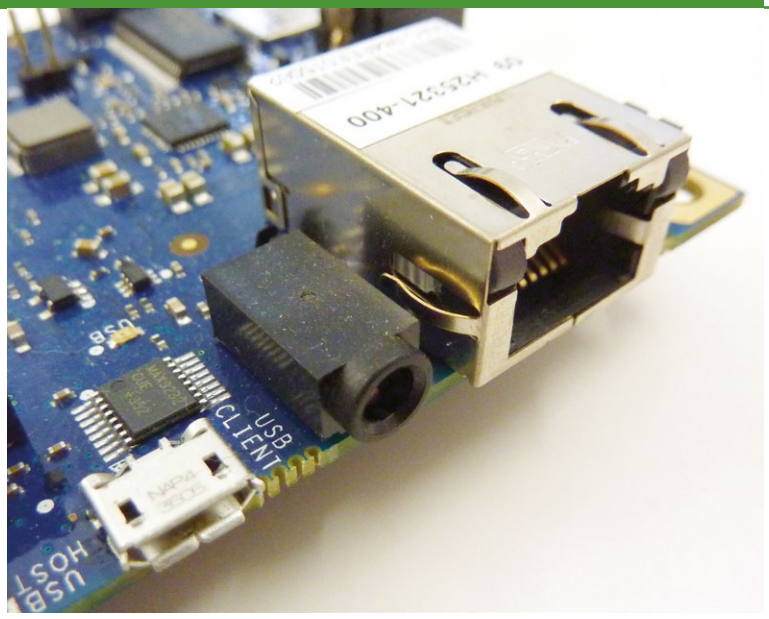
Rappelons au passage que Telnet n'est pas non plus activé sur le système Raspbian de la Raspberry Pi... parce qu'il est remplacé par SSH, une version sécurisée utilisant un mode de communication chiffré. Telnet lui, ne devrait même plus être utilisé de nos jours.

4. LA SUITE ? PAS DE SUITE POUR MOI, DÉSOLÉ

La documentation poursuit ses explications concernant l'après-démarrage. Il est question d'installer Visual Studio 2013 Express (ou une version payante) ainsi que le gestionnaire de paquets Nuget et le fichier `WindowsDeveloperProgramforIOT.msi`. Tout ce processus nécessite un compte Microsoft Connect et quantité de manipulations pour simplement développer son premier code (Hello Blinky). Chose qu'on fera, sans rien installer, en python sur une Raspberry Pi et en quelques minutes avec l'IDE Arduino.

La documentation, par moment, fait preuve d'une impressionnante clairvoyance : « *The main reason you'll want to telnet into your Galileo is so that you can interact with your Galileo and gracefully shut it down* » (« la principale raison pour laquelle vous voulez vous connecter en telnet à votre Galileo, c'est pour interagir avec elle et l'éteindre proprement »). En effet, arrivé à ce stade, la seule chose qui me vient à l'esprit c'est d'éteindre cette carte... et oublier que j'ai dépensé environ 90 euros pour l'avoir (le prix de trois Raspberry Pi). Du moins le temps que, soit j'évacue ma frustration et ma colère, soit que la Team Rocket (alias Microsoft et Intel) arrive à comprendre que ce n'est absolument pas comme cela que les choses fonctionnent. En attendant, la Galileo va gentiment s'envoler vers d'autres cieux (soit une distribution GNU/Linux si j'en ai le courage, soit le fond obscur d'un tiroir).

coromonadalix



Voici, à mon sens, LA caractéristique très révélatrice de la mauvaise approche d'Intel dans le domaine. La première génération de Galileo (fin 2013) proposait un connecteur pour la console série sous la forme d'un jack en RS-232 (+12v/-12v). Un choix incompréhensible alors que l'ensemble des cartes existantes proposait naturellement un connecteur standard au pas de 2,54 mm en 0/+5V ou 0/+3,3V.

J'ai cependant encore accordé un peu de temps à cet essai, mais cette fois avec un environnement adapté à l'exploration et à la bidouille. La Galileo ainsi équipée de Windows est configurée avec l'adresse 169.254.199.224. Sachant cela, il est possible d'utiliser une machine GNU/Linux pour s'y connecter en configurant une interface Ethernet avec une adresse sur le même réseau (disons 169.254.199.10). L'accès Telnet fonctionne sans problème, mais on peut maintenant utiliser des outils pour en savoir plus, comme **nmap** :

```
% nmap 169.254.199.224

Starting Nmap 6.25 ( http://nmap.org )
  at 2015-05-05 16:02 CEST
Nmap scan report for 169.254.199.224
Host is up (0.021s latency).
Not shown: 992 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
23/tcp    open  telnet
80/tcp    open  http
135/tcp   open  msrpc
445/tcp   open  microsoft-ds
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
MAC Address: 98:4F:EE:01:82:AE (Unknown)
```

Non seulement, bien sûr, le port Telnet est ouvert, mais ce n'est pas le seul. On constate que FTP, un autre protocole peu sûr, est actif (et accessible avec **Administrator**/**admin**), ainsi qu'un serveur web (HTTP). En pointant son navigateur sur l'adresse de la Galileo, on tombe sur une page affichant la liste des tâches, des fichiers et des statistiques mémoire. Un service de partage est également en place (port 445) ainsi que MSRPC (135). Nous avons donc affaire à un Windows assez classique et assez ouvert (dans le mauvais sens du terme). Je ne peux m'empêcher de penser aux conséquences d'une prolifération de ce type de plate-formes en termes de sécurité.

La conclusion de ce bref article est assez simple : la Galileo sous Windows est plus cher, moins puissante, moins pratique, plus difficile à prendre en main, moins ouverte, plus restrictive en termes de droits de l'utilisateur... D'un point de vue technique comme philosophique, c'est pour moi un très mauvais choix non seulement pour débuter, mais également pour l'utilisateur confirmé. Je ne vois même pas l'intérêt de lister les points positifs ou négatifs car, même en faisant de mon mieux, je suis incapable de trouver une unique raison de recommander cette carte.

En ce qui me concerne, ce matériel, ainsi que le système testé ici, ne sont que la confirmation que Intel et Microsoft ont raté ce train comme ils ont raté celui de la téléphonie mobile. Toute la communication, le marketing et la promotion au monde ne feront pas le poids face à des valeurs évidentes pour l'utilisateur : ouverture, sincérité, respect et assistance aux communautés de développeurs. Et ceci commence par le fait de ne pas demander ses papiers à quiconque veut télécharger un environnement de développement et à ne pas lui imposer des règles qui s'opposent à l'évolution naturelle d'une plateforme... **DB**

coromonadalix

PROFESSIONNELS !

DÉCOUVREZ NOS NOUVELLES OFFRES D'ABONNEMENTS ...

PDF COLLECTIFS

		PROFESSIONNELS					
		1 - 5 lecteurs		6 - 10 lecteurs		11 - 25 lecteurs	
OFFRE	ABONNEMENT	Réf	Tarif TTC	Réf	Tarif TTC	Réf	Tarif TTC
PROHK2	6ⁿ° HK	☐ PRO HK2/5	156,-	☐ PRO HK2/10	312,-	☐ PRO HK2/25	624,-

Prix TTC en Euros / France Métropolitaine

PROFESSIONNELS : N'HÉSITEZ PAS À NOUS CONTACTER POUR UN DEVIS PERSONNALISÉ PAR E-MAIL : abopro@ed-diamond.com OU PAR TÉLÉPHONE : 03 67 10 00 20

ACCÈS COLLECTIFS BASE DOCU

		PROFESSIONNELS					
		1 - 5 connexion(s)		6 - 10 connexions		11 - 25 connexions	
OFFRE	ABONNEMENT	Réf	Tarif TTC	Réf	Tarif TTC	Réf	Tarif TTC
PROOS+3	OS	☐ PRO OS+3/5	90,-	☐ PRO OS+3/10	180,-	☐ PRO OS+3/25	360,-
PROH+3	GLMF + HS + LP + HS + MISC + HS + OS	☐ PRO H+3/5	447,-	☐ PRO H+3/10	894,-	☐ PRO H+3/25	1788,-

Prix TTC en Euros / France Métropolitaine

OS = Open Silicium
LP = Linux Pratique
HS = Hors-Série
LM = GNU/Linux Magazine France

coromonadalix

...EN VOUS CONNECTANT À L'ESPACE DÉDIÉ AUX PROFESSIONNELS SUR :

www.ed-diamond.com

STOCKEZ VOS DONNÉES EN LIGNE AVEC RASPBERRY PI ET SEAFILE

Pierre-André BILLAT

Pour pouvoir accéder à vos données de partout, vous pouvez utiliser les services de Google, Apple ou encore Dropbox. C'est gratuit selon la quantité que vous y déposez, mais en échange ils peuvent accéder à vos documents (« Nous pouvons divulguer vos informations à des tiers si nous estimons que cela est raisonnablement nécessaire » - CGU Dropbox).Vous avez également l'option de tout héberger chez vous sur vos disques durs ! L'option la plus utilisée par les petites et moyennes entreprises c'est le NAS (serveur de stockage en réseau). Son prix est très variable selon le nombre de disques que vous allez pouvoir lui donner (environ 100 euros pour 2 baies jusqu'à 400 euros pour 4 baies bien évidemment en plus des disques durs).

coromonadalix

Aujourd'hui, je vous propose de réaliser votre cloud avec un Raspberry Pi et un disque dur. Soyons clairs dès le départ, la capacité du Raspberry Pi est très largement en deçà des NAS et leurs options de cloud actuellement sur le marché, mais si comme moi vous avez peu de données (<2To) vous n'avez pas trop intérêt à acheter un NAS.

1. LE PRINCIPE

Dans ce document, nous allons apprendre comment faire son propre cloud en utilisant un Raspberry Pi et un logiciel appelé Seafile. Il existe évidemment plein d'autres logiciels faciles d'installation à l'instar de ownCloud. Pour ma part, j'ai très nettement préféré Seafile à ce dernier pour plusieurs raisons : ownCloud est trop lourd pour fonctionner sur notre Raspberry Pi (à moins de l'overclocker...), il n'est plus actualisé pour le Raspberry Pi (il faut prendre une vieille version) et est moins sécurisé (Seafile chiffre la bibliothèque du serveur avec l'algorithme AES-128). De plus, l'application smartphone est gratuite ! Le hardware est composé d'un ou plusieurs disques durs externes permettant le stockage des fichiers et d'un Raspberry Pi assurant l'interface entre ces disques et l'utilisateur.

On va avoir besoin d'une carte SD formatée sous Raspbian (ou équivalent) avec quelques petites modifications.

Mais au fait une carte SD c'est quoi ? La carte *secure digital* est un petit support de stockage utilisant de la mémoire flash. Elle va donc pouvoir stocker rapidement beaucoup d'informations (la vitesse est fonction de la classe), mais elle a une durée de vie relativement limitée par le nombre de cycles d'écriture/réécriture. Ce nombre de cycles est défini plus ou moins lors de la fabrication de la carte par la qualité des matériaux. Pensez-y lorsque vous hésiterez à acheter une carte SD de marque inconnue à moitié prix. Ce qui est convenable pour un appareil photo peut devenir très vite limitant pour un mini-ordi.

Il faudra donc régulièrement sauvegarder ce qui est présent sur votre carte SD. Sachez toutefois qu'en cas de gros pépin vous avez le petit programme `fsck` qui peut vous être d'un grand secours.

On reprend notre config de Raspbian.

```
sudo raspi-config
```

Si ce n'est déjà fait, on étend la mémoire au maximum, « *Expand Filesystem* ». Et on redémarre la bête.

2. LE WIFI

Tout d'abord, il nous faut configurer le WiFi.

Vous avez le choix entre deux possibilités : soit le laisser connecté en Ethernet, soit brancher une petite clé WiFi. Avec une clé WiFi, vous pourrez planquer votre installation où vous voulez chez vous, mais vous perdrez du débit, à vous de voir. Personnellement, j'ai opté pour cette solution avec une clé compatible à 5 euros (EDUP EP-N8508GS). Dans les deux cas, il vous faudra configurer une adresse IP fixe.

Bon avant de commencer, on vérifie et installe les mises à jour :

```
sudo -i
```

Pour passer une bonne fois pour toutes en root (j'espère que vous me faites confiance !!!).

Maintenant, on configure le réseau :

```
root@raspberrypi: nano /etc/network/interfaces
```

Les plus barbus d'entre nous pourront bien sûr utiliser vi.

coromonadalix

On remplace les paramètres par défaut par :

```
auto lo
iface lo inet loopback

allow-hotplug wlan0
auto wlan0
iface wlan0 inet static
address 192.168.1.100
netmask 255.255.255.0
gateway 192.168.1.1
wpa-ssid "Mabox123"
wpa-psk "monsupermotdepasse"
```

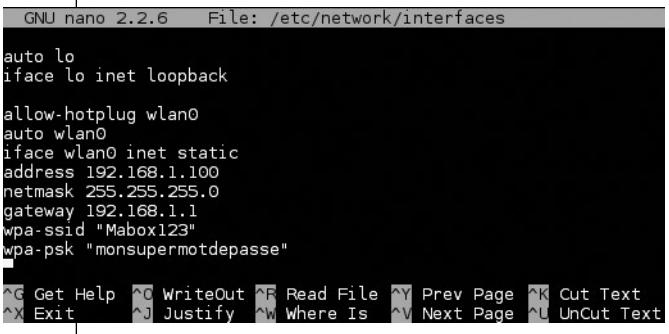


Fig. 1 : Édition du fichier de configuration réseau.

N.B. Si vous êtes en Ethernet, il faut remplacer **wlan0** par **eth0**, de plus les lignes relatives au WPA sont inutiles.

Address correspond à l'adresse réseau de votre Raspberry Pi, vous pouvez affecter la valeur que vous souhaitez au dernier nombre. Notez-le bien toutefois, car nous allons l'utiliser fréquemment.

Ensuite, un petit **ifdown wlan0**, puis **ifup wlan0** et tout sera en place. Vous pouvez le vérifier avec un **ifconfig**.

On peut également vérifier l'état de la connexion avec un **ping 192.168.1.100**.

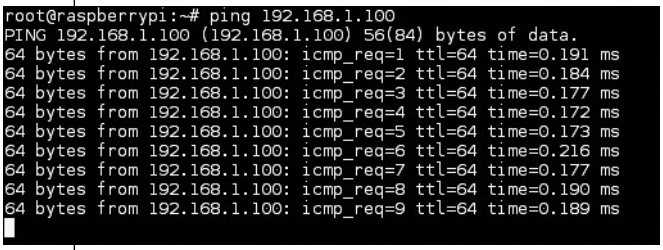


Fig. 2 : Sortie de la commande ping.

3. LE DISQUE DUR

Passons à présent au dur avec le disque.

Au niveau du choix des disques durs, bannissez les disques durs improprement appelés autoalimentés. Ceux-ci pompent en effet leur énergie depuis le port USB (et ici votre Raspberry Pi). Préférez donc les disques durs à brancher sur le secteur à 5400 tours par minute, moins énergivores.

De plus, le plus souvent ils se mettent en veille lorsque l'on ne les utilise pas (d'où l'intérêt d'avoir son Raspberry Pi branché sur le secteur).

Il est temps de formater votre disque dur, j'ai choisi le format de formatage ext4 avec **gparted**, mais il existe bien d'autres formats en fonction de l'utilisation que vous souhaitez en faire. Si vous souhaitez créer des liens symboliques par exemple, je ne peux que vous conseiller ext4 !

Si vous optez toutefois pour un formatage type FAT ou NTFS, il faut installer le pilote :

```
root@raspberrypi:apt-get
install exfat-fuse
root@raspberrypi:reboot
```

4. INSTALLATION DE SEAFILE

4.1 Installation des dépendances logicielles

On installe les packages qui vont permettre à Seafile de tourner :

```
root@raspberrypi:apt-get
install python2.7 python-
setuptools python-simplejson
python-imaging sqlite3
```

Puis **mysql** :

```
root@raspberrypi:apt-get
install mysql-server mysql-
client php5-mysql python-
mysqldb
```

Suivez le guide avec les options par défaut. Pensez à choisir un mot de passe solide et ne l'oubliez pas...

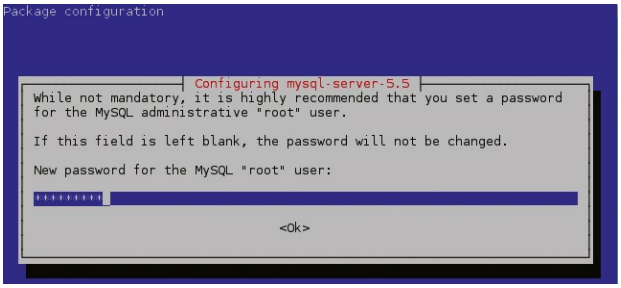


Fig. 3 : Installation de MySQL.

MySQL prend pas mal de place, on va donc préférer installer sa bibliothèque sur notre disque dur. On se place à l'intérieur de celui-ci et on crée notre dossier de bibliothèque.

```
root@raspberrypi:/# cd /media/votre disque dur externe/
root@raspberrypi:/media/ votre disque dur externe # mkdir lib
root@raspberrypi:/media/ votre disque dur externe # cd ./lib
root@raspberrypi:/media/ votre disque dur externe /lib# mkdir mysql
```

Maintenant, on crée un lien symbolique :

```
root@raspberrypi: ln -s /var/lib/mysql /media/ votre disque dur externe/lib/mysql
```

Ce qui signifie que lorsque le Raspberry Pi va chercher le dossier **mysql**, il ne le cherchera plus dans **/var/lib**, mais directement dans votre disque dur !

4.2 Installation de Seafile

Enfin, on peut télécharger Seafile, comme celui-ci n'est pas (encore) dans les dépôts, on le télécharge depuis les serveurs de dépôt :

```
root@raspberrypi: wget https://bitbucket.org/haiwen/seafile/downloads/seafile-server_4.0.6_pi.tar.gz
```

Notez l'extension en **tar.gz**, celle-ci signifie que le document n'est pas utilisable en tant que tel et demande quelques manipulations.

On crée le dossier d'installation :

```
mkdir ma_box
mv seafile-server_4.0.6_pi.tar.gz ma_box
cd ma_box
tar -xzf seafile-server_4.0.6_pi.tar.gz
mkdir installed
mv seafile-server_4.0.6_pi.tar.gz installed
```

Maintenant, on lance l'installation depuis le dossier :

```
cd seafile-server-4.0.6_pi
./setup-seafile-mysql.sh
```

Attention ! **seafile-server-4.0.6_pi** possède ici un trait d'union au lieu du tiret bas tout à l'heure (ça vous évitera de vous arracher les cheveux).

```
Need to get 84.9 kB of archives.
After this operation, 279 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main python-mysqldb ar
mhf 1.2.3-2 [84.9 kB]
Fetched 84.9 kB in 2s (30.8 kB/s)
Selecting previously unselected package python-mysqldb.
(Reading database ... 84560 files and directories currently installed.)
Unpacking python-mysqldb (from .../python-mysqldb_1.2.3-2_armhf.deb) ...
Setting up python-mysqldb (1.2.3-2) ...
root@raspberrypi:~/ma_box/seafile-server-4.0.6# sudo ./setup-seafile-mysql.sh
Checking python on this machine ...
  Checking python module: setuptools ... Done.
  Checking python module: python imaging ... Done.
  Checking python module: python-mysqldb ... Done.

-----------------------------------------------------------------
This script will guide you to setup your seafile server using MySQL.
Make sure you have read seafile server manual at

        https://github.com/haiwen/seafile/wiki

Press ENTER to continue
-----------------------------------------------------------------
```

Fig. 4 : Installation de Seafile.

```
For example: www.mycompany.com, 192.168.1.101
[ this server's ip or domain ] 192.168.1.100

Which port do you want to use for the ccnet server?
[ default "10001" ] 10001

Where do you want to put your seafile data?
Please use a volume with enough free space
[ default "/root/ma_box/seafile-data" ] /media/063C-4FFB/seafile-data

Which port do you want to use for the seafile server?
[ default "12001" ] 12001

Which port do you want to use for the seafile fileserver?
[ default "8082" ] 8082

----------------------------------------------------------
Please choose a way to initialize seafile databases:
----------------------------------------------------------

[1] Create new ccnet/seafile/seahub databases
[2] Use existing ccnet/seafile/seahub databases

[ 1 or 2 ]
```

Fig. 5 : Configuration de Seafile.

Vous vous laissez ensuite guider par le logiciel, vous créez votre compte **admin**, donnez une adresse mail (qui vous servira à vous connecter par la suite) et votre mot de passe sécurisé (pas 123456 !).

Nous créons alors une nouvelle base de données. On laisse les paramètres par défaut.

4.3 On démarre Seafile !

Toujours dans le dossier **seafile-server_4.0.6**, on lance le serveur :

```
./seafile.sh start
./seahub.sh start
```

De la même façon, si vous souhaitez arrêter le serveur :

```
./seafile.sh stop
./seahub.sh stop
```

Et maintenant... dans votre navigateur préféré et connecté à votre réseau local http://192.168.1.100:8000/.

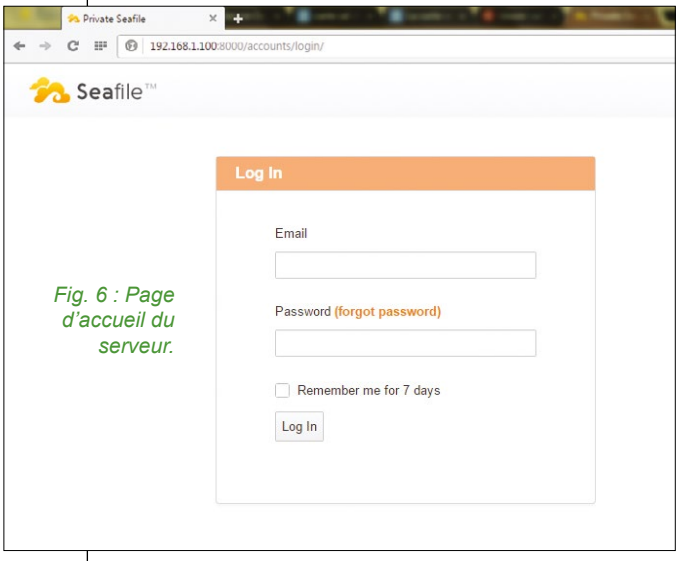


Fig. 6 : Page d'accueil du serveur.

Ici, on se connecte sur l'appareil du réseau portant le numéro 100 via le port numéro 8000.

Si vous n'accédez pas à cette page, vérifiez que vous avez correctement démarré le serveur (voir « On démarre Seafile »).

Après identification, vous arrivez sur une belle interface graphique, vous pouvez alors créer des dossiers et les protéger par un mot de passe, faire glisser-déposer des documents directement à l'intérieur de celui-ci...

Bon vous me direz c'est cool, mais si je veux y accéder depuis mon smartphone dans le train ?

Alors on repart.

5. CONFIGURATION HORS RÉSEAU LOCAL

On va ouvrir notre réseau au monde. Bien entendu, cela présente des risques de sécurité importants, c'est pourquoi, à moins d'être un crack de la sécurité réseau, évitez d'exposer votre réseau professionnel.

Pour limiter l'exposition, nous n'allons ouvrir que les ports utilisés par Seafile, on va faire ce que l'on appelle du *port forwarding*.

Allez sur la page d'accueil de votre box, dans votre navigateur : 192.168.1.1.

Après identification, vous devez trouver quelque part selon votre FAI une section appelée NAT.

Dans cette section, configurez comme dans le tableau ci-contre les ports.

Maintenant les ports sont ouverts !

Vous pouvez le vérifier sur le site **canyouseeme.org**. Tapez votre adresse IP puis le port à vérifier.

Maintenant autre chose, cette adresse IP vous est donnée par votre FAI et il se peut qu'elle change. Du coup vous ne pourrez plus vous connecter à votre cloud en tapant votre IP. Comment faire pour qu'elle reste toujours la même ? La solution est le pourvoyeur de DNS. Le *domain name server* fait correspondre à votre adresse IP un nom de serveur. Vous avez de nombreux sites sur lesquels vous pouvez acheter des noms de domaine. Personnellement, je suis sur no-ip qui présente l'avantage de fournir gratuitement 3 noms de domaine avec cependant la contrainte de devoir les renouveler (gratuitement) tous les mois.

	Nom	Protocole	Port de destination	IP de votre Raspberry Pi
1	Seahub	TCP	8000	192.168.1.100
2	FileServer	TCP	8082	192.168.1.100
3	Ccnet_Daemon	TCP	10001	192.168.1.100
4	Seafile_Daemon	TCP	12001	192.168.1.100
5	HTTPS	TCP	443	192.168.1.100

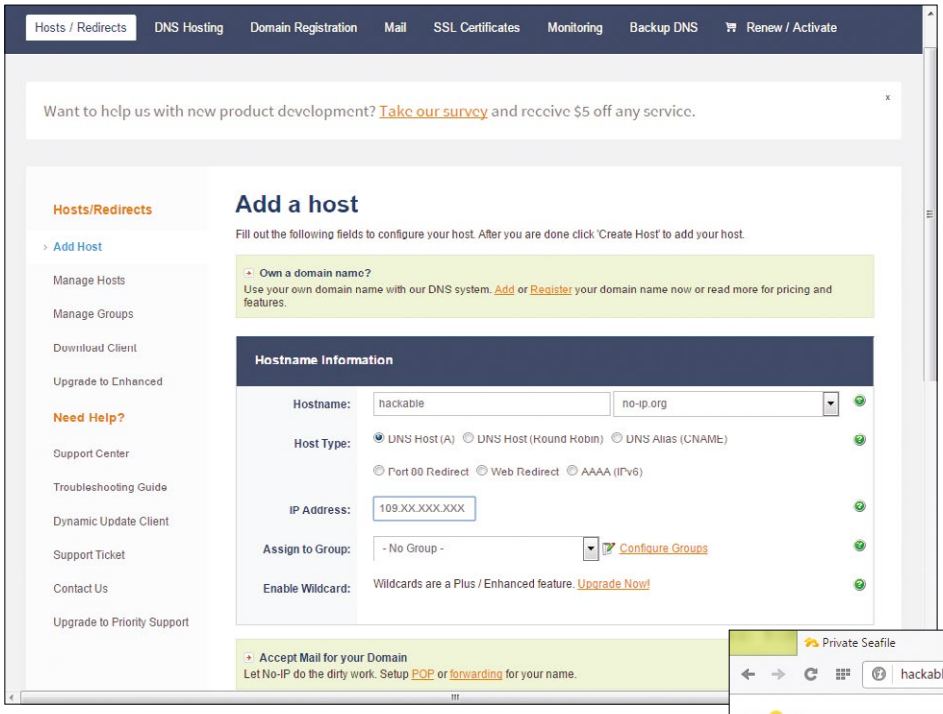


Fig. 7 : Obtenir son DNS via no-ip.

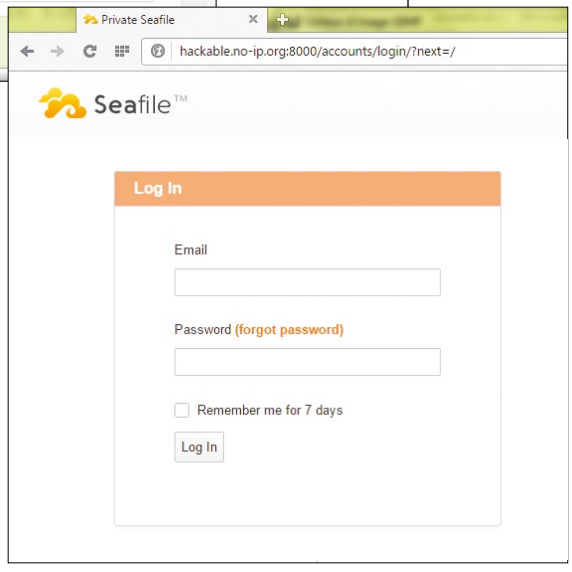


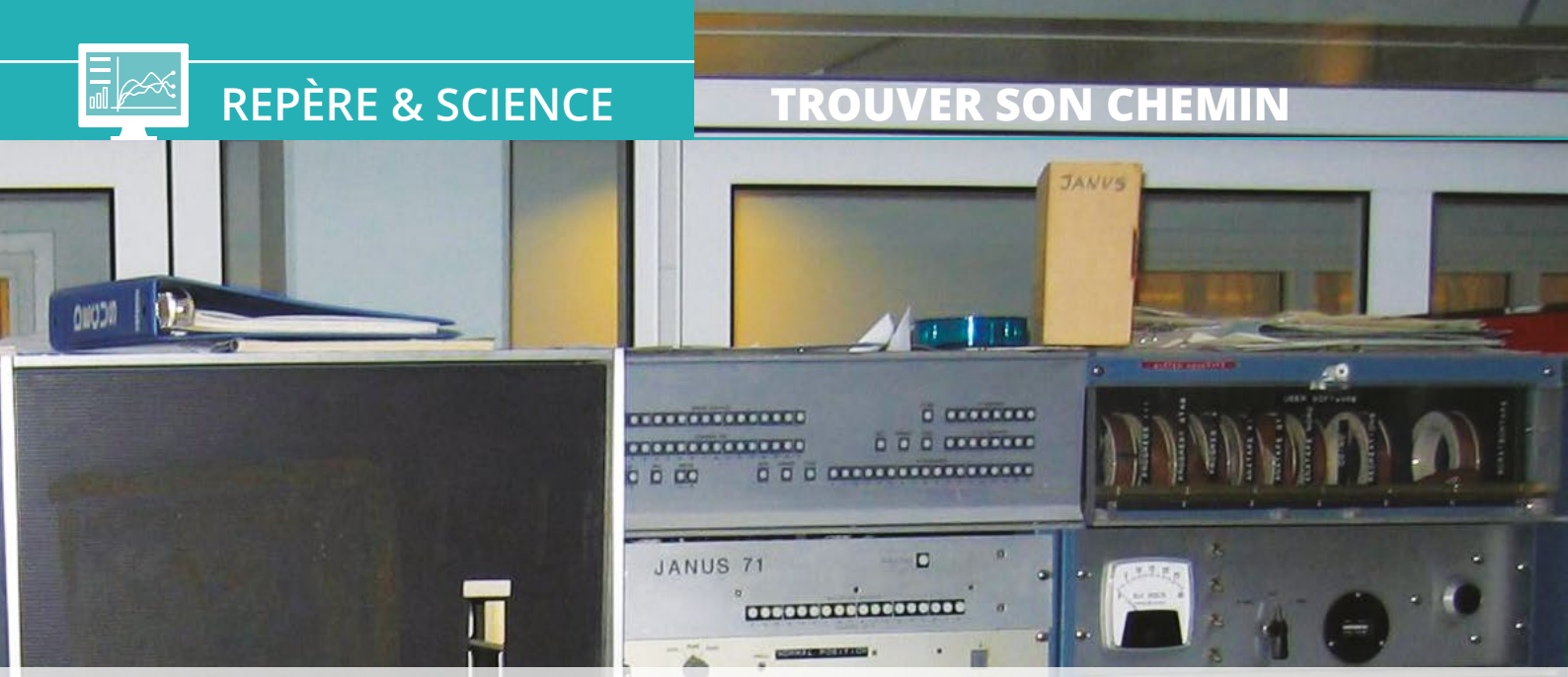
Fig. 8 : Page d'accueil du cloud.

Vous pouvez peut-être même configurer dans la page d'accueil de votre box votre DNS et donner directement votre nom de domaine (en **xxxx.noip.org** par exemple).

Maintenant, essayez de taper votre tout nouveau nom de domaine sur Internet...

Au final, nous savons comment stocker nos données sur un cloud personnel basé chez nous. Ce cloud est par définition accessible de partout soit via un client lourd (à télécharger sur le site officiel), soit via votre navigateur internet. Vos données personnelles le restent. Cependant, vous l'aurez certainement remarqué, la vitesse est réduite (chez moi 80kB/s en moyenne) et surtout, votre réseau est exposé à Internet (et aux méchants pirates...). Après à vous de voir ! **P-AB**

coromonadalix



S'Y RETROUVER DANS LES RÉPERTOIRES DE SA RASPBERRY PI

Denis Bodor

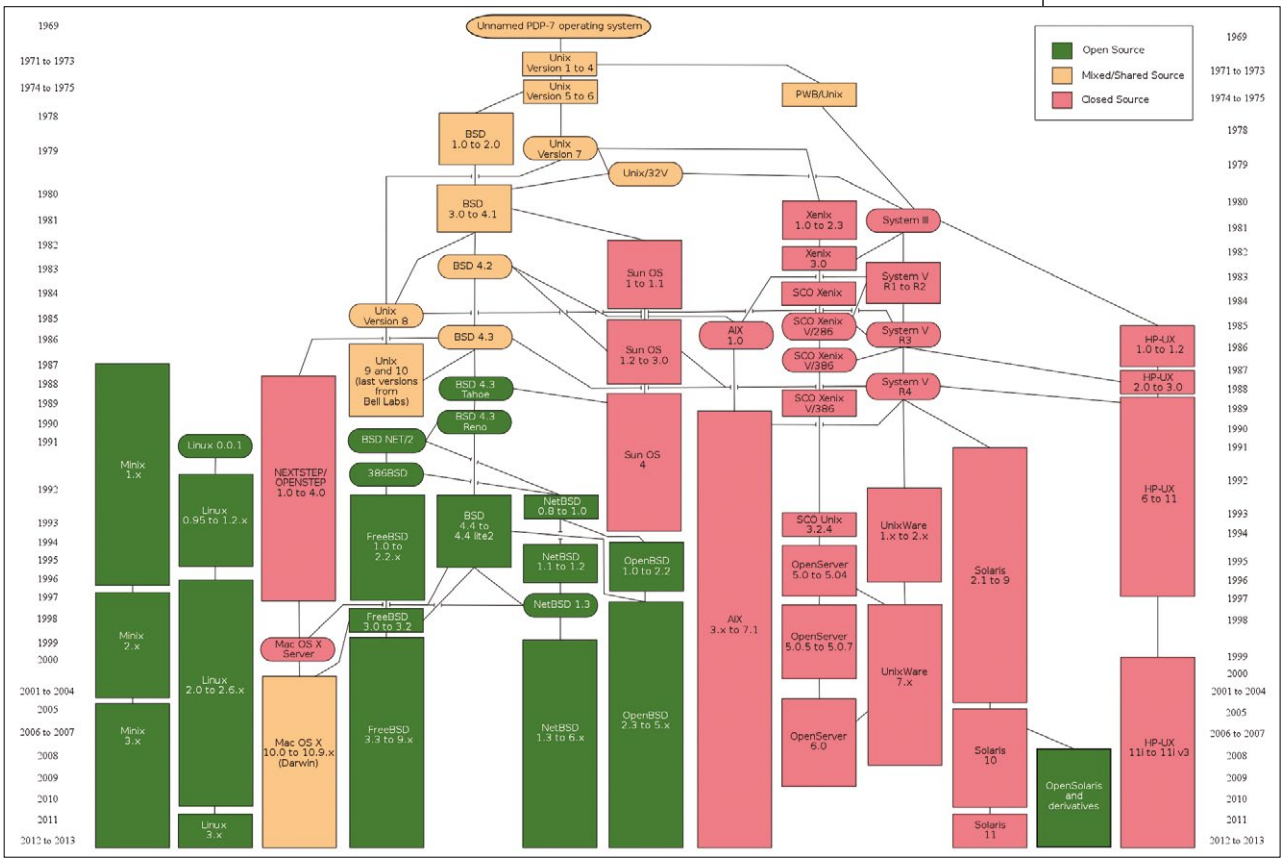
Les cartes Raspberry Pi comme bien d'autres matériels du même type, généralement qualifiés de nano-ordinateurs ou de systèmes mono-cartes sont pour beaucoup le moment de faire connaissance avec un nouvel univers bien différent de celui du PC Windows. Découvrir la Raspberry Pi est ainsi l'occasion de se frotter à un système bien particulier appelé GNU/Linux, mais aussi parfois, de s'y perdre. Pourtant, GNU/Linux, digne héritier d'une grande famille de systèmes UNIX, est avant tout le royaume de l'ordre et de la droiture. Voyons donc ensemble comment est organisé tout cela…

coromonadalix

Le système utilisé sur la Raspberry Pi et d'autres cartes comme la BeagleBone Black, la BananaPi ou encore l'Odroid, s'appelle GNU/Linux (même si cette désignation est généralement écourtée en « Linux »). Ce système n'est ni récent et ni limité aux seuls nano-ordinateurs. La plupart des serveurs sur Internet utilisent GNU/Linux tout comme bon nombre de postes de travail chez le particulier ou en entreprise. Il est également présent dans beaucoup d'équipements comme les caméras réseau, les routeurs, les box, les TV... ou encore, en version légèrement modifiée, dans près de 1,6 milliard de smartphones Android dans le monde.

Mais GNU/Linux n'est pas le seul de son espèce. Tout comme il existe plusieurs Windows, il existe plusieurs systèmes partageant la même structure et la même philosophie que GNU/Linux. On appelle cette famille **les systèmes UNIX**. Contrairement à Windows et son prédécesseur MS/DOS, la famille UNIX ne se limite pas à une seule société ou un seul éditeur de logiciel. Il faut plutôt voir cela comme un ensemble de bonnes idées et de choix qui forment une manière de concevoir la façon dont un système doit fonctionner : c'est la **philosophie UNIX**, et ses bases ont été jetées dès le milieu des années 70 par Ken Thompson, Dennis Ritchie et Brian Kernighan (entre autres). Ces idées et cette philosophie sont tellement robustes qu'elles perdurent et prospèrent encore de nos jours sous bien des formes et dans bien des systèmes. Mac OS X, par exemple, est un système UNIX.

Voici une version résumée de l'histoire des systèmes UNIX découlant du prodigieux travail de recherche et de synthèse d'Éric Lévénez. Visitez son site, levenez.com/unix, pour trouver la chronologie complète de 1969 à 2015 en PDF sur 32 pages à assembler soi-même. Idéal pour décorer son salon ! (source Wikipédia / Creative Commons BY-SA)



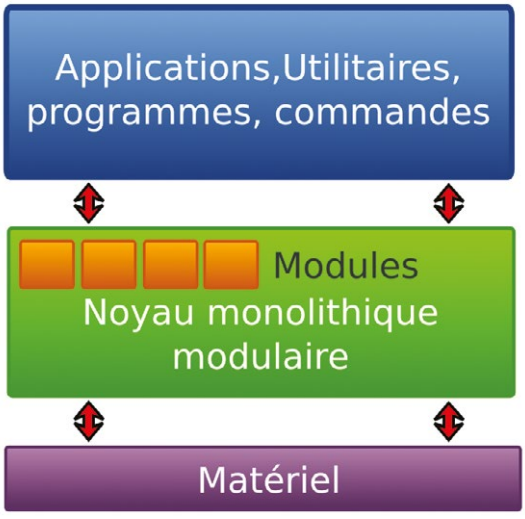
Cette philosophie peut être résumée de bien des manières, mais la plus courante est la suivante : cette famille se caractérise par une conception modulaire où chaque élément se limite à une tâche très précise, mais l'accomplit pleinement et parfaitement. C'est l'ensemble de ces « briques » qui, utilisées de concert, apportent la robustesse à l'ensemble. Implicitement, qui dit « un outil pour chaque tâche » dit « une place pour chaque chose et chaque chose à sa place ». Ceci est précisément l'objet du présent article.

1. AVANT TOUTES CHOSES, PLANTER LE DÉCOR

Ce qui nous intéresse ici est l'organisation des répertoires du système GNU/Linux. Il est important de savoir précisément de quoi on parle, car l'ordre est partout dans le système et les termes utilisés sont cruciaux. GNU/Linux est un **système d'exploitation**. Son travail est de vous permettre d'utiliser le matériel de la manière la plus simple et la plus efficace possible.

Pour accomplir cette tâche, il se compose de deux parties. Nous avons d'une part **le noyau**, chef d'orchestre du système et autorité toute puissante. C'est le noyau qui utilise directement le matériel, supervise l'exécution des programmes, se charge des communications et de la gestion de la sécurité. Dans un système GNU/Linux, ce noyau s'appelle **Linux**. C'est un gros morceau de code, toujours présent en mémoire et lancé dès les premières millisecondes du démarrage. Avec la Raspberry Pi, celui-ci prend la forme du fichier `kernel.img` sur la première partition de la carte SD ou microSD (celle que vous pouvez voir en glissant la carte dans un lecteur et en la parcourant sous Windows). Le noyau ou *kernel* en anglais, c'est ce fichier et rien d'autre (oui, il y a aussi des modules, mais dans l'absolu ce sont des morceaux de noyau, inutile de compliquer les choses pour l'instant). Je répète, le noyau, c'est un gros morceau de code et rien de plus.

Un système comme GNU/Linux sépare clairement le programme qui est en relation directe avec le matériel et les différents outils, utilitaires et applicatifs qui forment le reste de l'installation. Le premier est le noyau qui peut tout faire, tout gérer et tout contrôler et le reste, l'espace utilisateur, ayant des capacités réduites et fonctionnant sous l'autorité du noyau. Lorsqu'on parle du noyau Linux, on ne désigne que la partie verte de ce schéma.



Tout le reste des programmes qui composent le système est appelé **espace utilisateur** et fonctionne sous l'autorité du noyau. Dans cet espace utilisateur peuvent se trouver des milliers de programmes. Mais ce n'est pas tout, car chacun de ces programmes, outils, utilitaires ou applications ne se limite pas simplement à un ensemble d'instructions à exécuter. Ils peuvent avoir besoin de données, de configurations, d'images ou d'autres morceaux de programmes. Tout ce petit monde doit être stocké quelque part pour être facilement accessible, aussi bien par les programmes, mais aussi par l'utilisateur.

Bien entendu, avec des milliers de programmes qui ont besoin de données, il faut organiser tout cela. Pour structurer et ranger ces informations, on détermine des règles que l'ensemble du système doit suivre. On appelle cela un **système de fichiers**, et il en existe bien des sortes. Windows par exemple utilise généralement les systèmes de fichiers FAT ou NTFS. Mac OS X utilise HFS+, mais peut également utiliser FAT. Et GNU/Linux utilise le plus souvent ext2, ext3 ou ext4, mais peut accéder à presque tous les systèmes de fichiers existants (FAT, NTFS, HFS, ReiserFS, etc.).

coromonadalix

On confond souvent système de fichiers et partitions. Le fait est que les partitions ne sont qu'une façon de découper un support (comme un disque dur ou une carte SD) en plusieurs zones. C'est dans ces zones ou **partitions** que se trouvent les systèmes de fichiers. Comme on créé généralement une partition Windows pour y placer un système de fichiers FAT ou NTFS, ou encore une partition Linux pour de l'ext2, le raccourci consistant à parler de partition Windows pour désigner en réalité le système de fichiers FAT qui s'y trouve est vite fait. Dans l'absolu pourtant, il s'agit de choses bien différentes (rien ne vous empêche de créer un système de fichiers ext2 dans une partition Windows, si ce n'est le bon sens).

La quasi-totalité de ces systèmes de fichiers utilise les notions de **fichiers** et de **répertoires**. Les fichiers représentent des paquets de données auxquels on donne un nom pour pouvoir les retrouver, et les répertoires permettent de regrouper les fichiers dans des « dossiers ». Les répertoires peuvent contenir d'autres répertoires et l'ensemble prend alors la forme d'une structure faisant penser à celle d'un arbre où les feuilles/fichiers sont attachées à des petites branches, elles-mêmes attachées à des plus grosses, etc. On parle donc d'une structure arborescente ou, plus souvent, d'une **arborescence**.

Et nous en venons précisément au sujet de cet article : comprendre le sens de l'organisation de l'arborescence du système GNU/Linux utilisé sur une Raspberry Pi. Notez que les répertoires dont nous allons parler se trouvent non seulement dans GNU/Linux, quelle que soit la distribution utilisée (Debian, Ubuntu, Fedora, SuSE, etc.), mais également dans tous les systèmes UNIX, plus ou moins certaines variations. Leur utilité est restée inchangée depuis les toutes premières versions d'UNIX, n'en déplaise à certains qui semblent leur préférer des usages imaginaires et étalent leur manque d'expérience en vidéo **[1]** (oui, ça m'a énervé).

2. LES RÉPERTOIRES IMPORTANTS

Nous n'allons pas détailler l'intégralité de l'arborescence, car non seulement ceci occuperait l'ensemble du magazine, mais, de plus, seule l'architecture générale est importante et réellement fixe. Le contenu des répertoires et les sous-arborescences sont relativement dépendants des éléments installés et configurés, mais aussi de la distribution choisie. Nous nous contenterons ici de conserver un caractère générique tout en apportant des précisions uniquement sur le système le plus utilisé avec Raspberry Pi : Raspbian.

Notez que tous les systèmes UNIX n'utilisent pas ces répertoires de la même manière, mais que, par souci de compatibilité, la fondation Linux, depuis 1994, maintient un document décrivant un standard généralement respecté par tous les systèmes GNU/Linux. C'est le FHS pour *Filesystem Hierarchy Standard* (standard de la hiérarchie des systèmes de fichiers). Ainsi, ce qui est valable pour un système Raspberry Pi doit également l'être pour une Ubuntu installée sur PC, par exemple.

Les répertoires dont nous allons parler se trouvent à la racine du système de fichiers, qui est symbolisée par le caractère `/`. Vous pouvez voir cela comme le `C:\` des systèmes Microsoft, le point de départ de l'ensemble de l'arborescence. Vous pouvez très simplement lister les répertoires à la racine en faisant un simple `ls /` (le `-F` est là pour faire plus joli) :

```
$ ls -F /
bin/   boot/    dev/   etc/      home/   lib/
lost+found/  media/  mnt/       opt/   proc/
root/   run/    sbin/  selinux/  srv/    sys/
tmp/   usr/     var/
```

2.1 /bin et /sbin

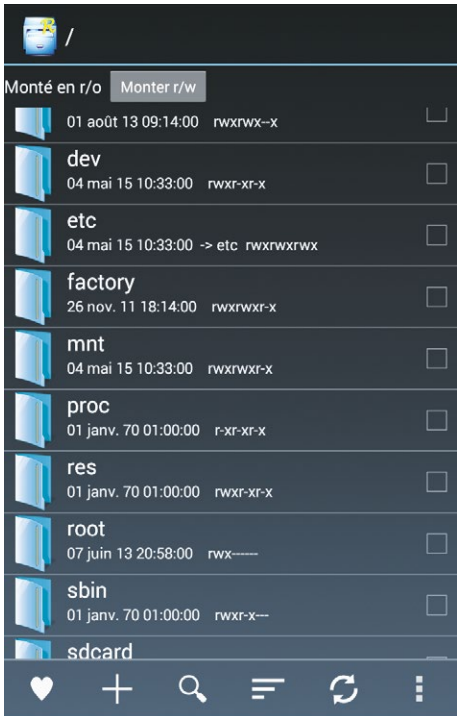
Ces deux répertoires contiennent les programmes (commandes, outils, utilitaires) importants pour le fonctionnement et l'utilisation du système. `/sbin` se différencie de `/bin` par le fait qu'il regroupe les programmes indispensables au fonctionnement du système alors

que **/bin** concerne les programmes qui doivent être disponibles pour l'utilisateur. N'entrons pas trop dans le détail, mais sachez que lorsque vous utilisez GNU/Linux, vous vous trouvez dans un niveau d'exécution (*runlevel*) particulier. En temps normal, vous êtes en niveau 2, le 6 étant le redémarrage du système et le 1 le mode maintenance. Les programmes dans **/bin** sont ceux qui doivent être accessibles en niveau 1. Ce mode est généralement utilisé en cas de gros problème et n'accepte qu'un seul utilisateur, généralement le super-utilisateur (il est également appelé mode mono-utilisateur).

S'il faut retenir quelque chose de ces deux répertoires, c'est ceci : ne vous en mêlez pas et n'y mettez pas votre petit grain de sel. Leur contenu est l'affaire du système lui-même. Il existe un autre endroit pour vous et vos petites affaires :

2.2 /usr

Ce répertoire comme son nom l'indique est fait pour les utilisateurs, les *users* en anglais (**usr** en abrégé). Ce répertoire contient une hiérarchie complète assez similaire à cette de la racine. Attention, par « utilisateurs » il faut comprendre « destiné à être utilisé par les utilisateurs », et non géré par ces derniers. Seul le responsable humain (ou vaguement humain) de tout le système, le super-utilisateur **root**, peut y écrire. Les autres se contentent d'y lire et d'utiliser les programmes qui s'y trouvent.



Voici une capture d'écran du gestionnaire de fichiers RootExplorer sous Android. À la racine, on reconnaît clairement les différents répertoires typiques d'un système GNU/Linux : /dev, /root, /sbin, /etc, /mnt...

À propos d'exécution justement, c'est dans **/usr/bin** que se trouvent les programmes pour tous les utilisateurs. La différence avec **/bin** est le fait qu'ils n'ont pas à être disponibles en mode maintenance. **/bin**, par exemple, contient la commande **ls**, mais c'est dans **/usr/bin** que se trouve **raspi-config** ou le navigateur web **epiphany** ainsi que toutes les commandes de gestion de paquets (**apt**, **dpkg**, **aptitude**, etc.).

Certains programmes sont un peu plus que des commandes et des outils pour les utilisateurs. Ils font partie intégrante des mécanismes qui, en coulisse, font fonctionner le système. Pourtant, il ne sont pas essentiels et leur absence n'empêchera pas le système de fonctionner. Ceux-ci se trouvent, alors non pas dans **/usr/bin** mais, dans **/usr/sbin**. La plupart des services fonctionnant en arrière-plan (démons) se trouvent dans ce répertoire.

Les programmes simples se suffisent à eux-mêmes (ou presque), mais d'autres dépendent de morceaux de code qui se trouvent ailleurs, dans des bibliothèques. Le principe est simple si un logiciel A et un logiciel B ont besoin de savoir lire des images, il est stupide d'avoir cette fonctionnalité en double dans le système. Il est bien plus efficace que A et B reposent sur un élément commun charger de lire et traiter les données d'une image. C'est une bibliothèque partagée chargée dynamiquement. C'est un peu comme les bibliothèques Arduino à la différence qu'ici tout se passe au moment du lancement du programme.

Il existe quantité de bibliothèques dans le système. N'oubliez pas la philosophie UNIX : ne faire qu'une chose, mais le faire bien. Dans **/usr/lib**, on trouvera donc quantité de fichiers dont le nom se termine par **.so** et des chiffres. **libgif.so.4.1.6** par exemple est une bibliothèque qui regroupe plein de fonctionnalités liées à la manipulation des fichiers GIF et sera utilisée par tous les programmes en ayant besoin. On dit que les programmes et la bibliothèque sont liés.

Mais ce n'est pas tout. Certains programmes ont besoin d'autres fichiers pour fonctionner. Un cas

typique est celui des applications en mode graphique qui nécessitent des icônes à afficher sur des boutons ou encore des thèmes que l'utilisateur peut choisir. Ce ne sont ni des programmes, ni des bibliothèques et ces éléments se trouvent alors sous forme de fichiers dans **/usr/share** (*share* comme « partage », ce sont des éléments partagés). **/usr/share/icons** par exemple contient une quantité phénoménale d'icônes utilisées par presque toutes les applications graphiques. De la même manière, **/usr/share/doc** regroupe tous les fichiers de documentation installés avec les programmes et applications. Enfin, **/usr/share/man** contient les pages de manuels, une forme de documentation consultable avec la commande **man** suivie du nom du programme ou de la fonctionnalité recherchée (oui **man man** fonctionne, forcément).

Enfin, nous devons parler de **/usr/local** qui est relativement spécial. De la même manière que **/usr** contient une arborescence similaire à **/**, **/usr/local** contient un troisième niveau avec son **/usr/local/bin**, **/usr/local/lib** ou encore son **/usr/local/share**. Historiquement, **/usr/local** est destiné à regrouper les éléments spécifiques au système local, à la machine en cours d'utilisation. Deux installations de Raspbian sur deux Raspberry Pi différentes auront des **/usr** relativement similaires, voire totalement identiques si les logiciels installés sont les mêmes. Les systèmes UNIX cependant partent du principe que le propriétaire d'un ordinateur est aussi un programmeur et qu'il aime à compléter son système d'outils qu'il développe, installe et gère lui-même. **/usr/local** est l'emplacement pour ces créations locales qui n'existent en principe pas sur d'autres ordinateurs, même équipés du même système.

2.3 /etc

Le répertoire **/etc** est celui où sont stockées les configurations pour l'ensemble des éléments du système. Ceci concerne tout autant la configuration du système lui-même, du réseau, des scripts de démarrage, des paramètres par défaut, mais aussi les configurations des applications. Notez que généralement, un programme installé dans **/usr/local** verra sa configuration stockée dans **/usr/local/etc** et non **/etc**. Il est important de le relever, car de nombreux tutoriels expliquent qu'il faut installer les nouveaux programmes en utilisant les commandes **./configure**, **make** et **make install** qui par défaut placent tout dans **/usr/local**. Généralement, c'est un bon moyen de semer le désordre dans le système, mais parfois il n'y a pas le choix.

Le nom **/etc** peut paraître étonnant, car on s'attendrait à ce que **/conf** soit un choix plus judicieux. Historiquement, **/etc** a été choisi comme le diminutif de « et cetera » et était destiné, littéralement, à contenir « tout ce qui n'allait pas ailleurs ». Avec le temps, la majorité des fichiers dans ce genre de situation étaient ceux de configuration et le FHS interprète maintenant ces trois lettres comme *Editable Text Configuration* (configuration en fichiers textes éditables). Le standard précise également très explicitement que ce répertoire ne doit pas contenir de programmes exécutables.

Notez que **/etc** contient des configurations valables pour l'ensemble du système. L'éditeur **nano** par exemple, possède un **/etc/nanorc** contenant la configuration par défaut. Elle sera appliquée pour tous les utilisateurs... sauf si un **.nanorc** existe dans le répertoire utilisateur. Dans ce cas, c'est la configuration de l'utilisateur qui prévaut sur celle générique.

2.4 /home

Lorsque vous vous connectez au système de votre Raspberry Pi, vous vous retrouvez, en ligne de commandes, dans votre répertoire personnel symbolisé par un caractère tilde **~** (ou **$HOME**). En réalité, vous vous trouvez dans un sous-répertoire de **/home** :

```
$ pwd
/home/pi
```

Il est fort probable que vous n'utilisez votre carte qu'avec un seul compte utilisateur et très certainement le compte **pi** configuré initialement. Il faut cependant savoir que pour chaque utilisateur existe un répertoire personnel (ou *home*, maison), qu'il soit humain ou non. Vous utilisez le compte **pi**, mais les programmes fonctionnant en tâche de fond dans GNU/Linux possèdent également une identité et donc un compte dans le

système. Ainsi, ils possèdent aussi un répertoire personnel, mais celui-ci est généralement défini ailleurs que dans `/home`.

C'est un « privilège » réservé aux humains et ceci permet, pour chaque utilisateur, d'avoir un petit espace personnel dans lequel placer ses fichiers et ses petites affaires. Seul un utilisateur possède un emplacement différent. Le super-utilisateur `root` dispose d'un répertoire `/root`. Ceci découle de l'architecture même du système qui part du principe que cet utilisateur est le responsable du système et donc celui aux commandes en mode mono-utilisateur en cas de maintenance ou de problème. Le répertoire `/home` était (et parfois l'est encore) un système de fichiers différent du reste du système, voire un disque différent. En cas de démarrage en niveau d'exécution 1, le disque n'est alors pas accessible et `/home` reste vide.

2.5 /var

`/var` est destiné à contenir les éléments variables du système. Tout ce qui est changeant se trouve ici. Il peut s'agir de journaux d'activité, d'éléments temporaires comme les informations sur l'exécution des services ou encore de caches permettant d'accélérer le fonctionnement de certains programmes. On y trouve également les boîtes mail locales, les files d'attente pour les impressions ou encore les fichiers d'un serveur web ou ceux d'une base de données.

Tout ce qui a une durée de vie limitée ou qui est susceptible de changer au cours du fonctionnement du système se trouve ici. Attention, « variable » n'est absolument pas synonyme de « temporaire ». Les fichiers temporaires trouvent leur place dans `/tmp` et n'ont pas de persistance entre deux démarrages du système.

Notez qu'avec l'évolution des versions, quelques changements ont été apportés au contenu de `/var`. Ainsi les systèmes actuels utilisent `/var/run`, mais ce n'est qu'un lien symbolique (un raccourci) vers `/run`. Ce répertoire historiquement dans `/var` sert à stocker des fichiers contenant des informations sur les programmes en cours d'exécution.

2.6 /mnt et /media

Un système UNIX comme GNU/Linux est relativement différent dans son utilisation courante d'un système Windows. Du moins, ceci était le cas jusqu'à l'arrivée des périphériques de stockage USB. Typiquement, lorsque vous voulez utiliser une clé USB, vous la connectez au port adéquat et quelques secondes après, les fichiers qu'elle contient sont accessibles par le gestionnaire de fichiers. Lorsque vous ne l'utilisez plus, vous éjectez la clé puis la déconnectez pour la ranger.

Cette étape d'éjection n'était pas dans les mœurs avec d'autres supports amovibles comme les CD-ROM ou les disquettes. Dans le premier cas, retirer le CD en pleine lecture n'était pas vraiment en problème, aucune

Voici un mini-ordinateur DEC PDP-7. Le même modèle qu'utilisait Ken Thompson en 1969 lorsqu'il développa son propre système. Dès l'année suivante, en compagnie de Dennis Ritchie et Brian Kernighan, ces travaux découlaient sur un nouveau système appelé UNICS. Le nom fut ensuite réduit en UNIX et plus tard le système réécrit dans un tout nouveau langage créé par Dennis et Brian, le C. Dès le début des années 70, on retrouve la plupart des répertoires qui sont décrits dans cet article (photo Wikipédia - Matias Fjeld, licence CC BY SA 3.0).

donnée ne pouvait être perdue puisque le support est en lecture seule. Dans le cas des disquettes, soyons honnêtes, qui aurait eu l'idée d'y toucher tant que le voyant était actif et que le moteur s'excitait à l'intérieur du lecteur ? L'attente n'était jamais très importante étant donné l'espace disponible.

Une clé USB en revanche est très différente et l'étape d'éjection est absolument nécessaire. L'espace disponible et le caractère non mécanique permettent de facilement déconnecter le périphérique alors même que le système est en train d'y écrire des données. Données qui seront alors corrompues. Il faut éjecter le support pour signifier au système qu'il doit terminer son opération et ne plus accéder au périphérique.

Avec un système GNU/Linux c'est légèrement différent. Windows détecte la connexion de la clé USB et la rend immédiatement accessible sans demander l'intervention de l'utilisateur. Ce n'est qu'ensuite qu'il demande si vous avez une préférence concernant l'application à utiliser. GNU/Linux, lui, détecte également la connexion, mais n'accédera pas par défaut au contenu. Le support est considéré comme un périphérique non utilisé par le système. Pour accéder aux fichiers, vous devez **monter** le système de fichiers, c'est-à-dire le rendre accessible. Là encore, contrairement à Windows, vous avez le choix quant à l'endroit qui constituera le point d'accès. On **monte** le système de fichiers dans un répertoire et, dès lors, le contenu du répertoire correspond au contenu de la clé. Inversement, une fois l'utilisation terminée, on **démonte** le système de fichiers. Toutes les écritures en attente sont appliquées et la clé est laissée en paix. Pour autant, celle-ci n'est pas déconnectée du système. Pour la faire réellement disparaître, on peut l'éjecter pour la considérer comme déconnectée.

Les répertoires servant de points d'accès, on parle de **points de montage**, se trouvent dans `/mnt` lorsqu'il s'agit de montages temporaires et dans `/media` en ce qui concerne les périphériques amovibles (dans le sens CD/DVD du terme).

Notez que, pour un système UNIX, l'intégralité des systèmes de fichiers est ainsi montée. Ceci est bien entendu également valable pour votre `/` qui n'est autre que le système de fichiers ext2 présent dans la seconde partition de la carte SD ou microSD de la Raspberry Pi. De la même manière, le système de fichiers FAT de la

coromonadalix

première partition est également monté, cette fois dans `/boot` qui historiquement contient les fichiers nécessaires au démarrage du système. Vous pouvez lister les systèmes de fichiers montés en utilisant la commande `mount`.

2.7 /dev

En faisant abstraction de `/tmp` et `/run`, `/dev` est le premier répertoire spécial que nous rencontrons. Il est particulier dans le sens où il semble contenir des fichiers existant physiquement, mais qui ne sont en réalité qu'une illusion. Ainsi, si vous éteignez votre Raspberry Pi, retirez la carte SD/microSD et la placez dans un PC GNU/Linux, vous constaterez que ce répertoire ne contient rien. Pourtant, lorsque le système est en marche sur la Pi, vous y verrez plus de 150 fichiers !

Il n'en a pas toujours été ainsi, mais les fichiers présents dans `/dev` ont toujours été spéciaux puisqu'il ne s'agit pas de réels fichiers, mais de nœuds. Une autre caractéristique de la philosophie veut que la représentation idéale de tous les composants d'un système soit le fichier. Selon ce principe, ce que vous voyez, par exemple, s'afficher sur l'écran en utilisant une commande est en réalité, pour le système, une écriture dans un fichier qui représente cette sortie (STDOUT). De la même manière, les données acceptées par une commande sont aussi, en interne, des lectures d'un fichier représentant le périphérique d'entrée (STDIN). Cette uniformité permet de faciliter grandement le travail des programmeurs, mais aussi celui des utilisateurs, car pour envoyer des données d'un programme à un autre il suffit de connecter la sortie de l'un à l'entrée de l'autre. Mieux encore, on peut tout aussi bien diriger la sortie vers un fichier, un vrai, pour ensuite lister ce contenu et l'envoyer en entrée à un autre programme.

Cette approche « tout est fichier » va jusqu'à l'accès aux périphériques eux-mêmes. Cependant, pour pouvoir faire le lien, il faut que l'arborescence liste des fichiers représentant ces périphériques. C'est précisément le contenu de `/dev`.

Historiquement, ces fichiers ou nœuds étaient créés de manière statique ou manuellement avec la commande `MAKEDEV`. Avec l'arrivée des périphériques connectables à chaud (comme l'USB) cela n'était plus gérable. Un mécanisme a donc été mis en place et s'est bonifié avec le temps pour permettre au système d'automatiquement créer des fichiers dans `/dev` en fonction de l'apparition des périphériques.

Cette architecture couvre presque l'ensemble des périphériques. `/dev/fb0` par exemple est la mémoire de l'affichage sur le port HDMI de la Raspberry Pi. `/dev/mmcblk0` représente l'intégralité de la carte SD/MicroSD et `/dev/mmcblk0p2` est une représentation de la seconde partition. `/dev/bus/usb/001` est un répertoire représentant le bus USB de la Raspberry Pi, à l'intérieur se trouve une arborescence complète représentant les périphériques connectés.

Le fait que les pseudo-fichiers de `/dev/` soient gérés dynamiquement rend l'utilisation d'un vrai système de fichiers désuet et pénalisant. On parle donc de système de fichiers virtuel. Il n'est pas réel, mais est une représentation créée par le système, c'est pourquoi, lorsque le système n'est pas en marche, le point de montage apparaît vide.

2.8 /proc et /sys

Ces deux répertoires sont également des points de montage pour des systèmes de fichiers virtuels. `/proc` contient une représentation de l'ensemble des processus (programmes en exécution) dans le système ainsi qu'un grand nombre d'informations sur le système lui-même. On y trouve par exemple des données sur le processeur (`/proc/cpuinfo`), sur la mémoire (`/proc/meminfo`), sur le temps écoulé depuis le démarrage (`/proc/uptime`), sur les paramètres utilisés au démarrage du noyau (`/proc/cmdline`), etc. On y trouve également un ensemble de répertoires nommés par des chiffres. Ceux-ci représentent les programmes en cours d'exécution, identifiés par leur identifiant de processus (PID). Dans ces répertoires se trouvent, sous la forme de fichiers, toutes les informations utiles sur le processus concerné.

`/sys` est différent, car il apporte plus que de simples informations. Il permet en effet un lien entre l'espace utilisateur et le noyau. Dans `/sys`, on retrouve l'ensemble du système tel que vu par le noyau. Il est alors possible de consulter les données, en lisant les fichiers, mais également d'indiquer au noyau des paramètres quant à la gestion des périphériques ou des fonctionnalités internes, en écrivant dans les fichiers.

Il est important de faire clairement la différence entre `/sys` et `/dev`. Le premier permet de dialoguer avec le noyau, parfois en rapport avec des périphériques, alors que le second permet un accès aux périphériques. Historiquement, les systèmes UNIX reposaient sur la commande `sysctl` pour dialoguer d'un espace à l'autre, mais l'arrivée de `/sys` a uniformisé l'ensemble à la façon de `/dev` et `/proc` (`sysctl` cependant est toujours utilisable et utilisé). Rappelez-vous, tout est fichier...

3. OK, MAIS AVEC QUOI JE PEUX JOUER ALORS ?

Avec tout et avec rien, ça dépend. Vous pouvez donc toucher, tritouiller, explorer et personnaliser, mais cela dépend :

- des permissions dont vous disposez. Par défaut, un utilisateur standard a tout pouvoir sur ses propres fichiers, mais aucun sur ceux d'autres utilisateurs et encore moins sur ceux du système. Le super-utilisateur `root` lui, par l'entremise de la commande `sudo` pour un utilisateur autorisé, peut agir sur presque tous les fichiers du système ;
- des risques que vous voulez prendre et du chaos que vous êtes prêt à assumer. Tout comme avec d'autres systèmes, rien ne vous empêche de faire tout et n'importe quoi un peu partout (avec `sudo`), mais c'est un peu comme bricoler son vélo... en roulant dans une descente... et à contresens, il faut savoir ce qu'on fait ;
- du respect vis-à-vis de l'organisation déjà en place, en particulier celle utilisée par la distribution installée. Raspbian sur Raspberry Pi est un dérivé de Debian GNU/Linux. Ce système utilise une gestion de paquets pour installer et désinstaller des programmes. En installant à la main, même dans `/usr/local`, vous ne reposez plus sur le système de gestion de paquets qui, entre autres choses, garde une trace de ce qui est installé ou non. Sans lui, à vous de vous souvenir de ce que vous avez bricolé, comment et où.

De manière générale, en commençant à jouer avec les fichiers et les éléments installés vous devez impérativement partir du principe que quelque chose est sûrement déjà prévu pour vous faciliter la vie. Il peut s'agir d'une version en paquet du logiciel que vous souhaitez installer, mais également d'un emplacement plus adéquat qu'un autre pour configurer une fonctionnalité.

Nous l'avons vu dans l'article sur l'utilisation d'une horloge temps réel permettant de garder la Raspberry Pi à l'heure. Il existe bien des manières de prendre en charge cette fonctionnalité, mais une solution est meilleure que les autres, tout simplement parce que le système est déjà prévu pour accepter facilement ce type de configuration.

Ne vous présumez jamais plus malin que les développeurs qui travaillent à la conception et à l'amélioration de GNU/Linux ou de la distribution. Et je le dis pour vous comme pour moi. Vous ferez cette erreur comme je l'ai faite en débutant et comme tantôt je la fais encore. Mais c'est en faisant des erreurs qu'on apprend et c'est en crashant son système bêtement qu'on finit par comprendre.

Cet article a pour but de vous donner, dans les grandes lignes, une idée de l'organisation générale, mais ce n'est qu'une carte sommaire du territoire que seul vous pouvez explorer. Souvent, il faut se perdre pour découvrir de nouveaux territoires et échouer avant de trouver la bonne solution. Ceci s'appelle l'expérience et si j'y suis arrivé, vous le pouvez aussi. **DB**

[1] https://www.youtube.com/watch?v=NTmExWohkAQ

coromonadalix

4 SOLUTIONS POUR FAIRE CLIGNOTER UNE LED

Denis Bodor

Il faut l'avouer, depuis l'arrivée de l'Arduino le milieu de l'électronique de loisirs s'en est trouvé tout chamboulé. Il est vrai que c'est une solution parfaite si l'on veut rapidement obtenir un résultat après s'être jeté à l'eau. Le premier croquis de tout utilisateur est la classique led qui clignote, mais les électroniciens font clignoter des leds depuis des décennies. Bien avant l'Arduino, les microcontrôleurs ou les circuits intégrés. Faisons donc ensemble un petit retour aux sources…

Notre objectif ici est de simplement faire en sorte qu'une led s'allume puis s'éteigne en boucle environ une fois par seconde. Nous allons explorer ici quelques solutions pour cette tâche simple et apprendre au passage qu'il y a bien plus d'une manière de faire et d'envisager la chose. Bien sûr, il ne s'agit que d'une minuscule sélection de techniques, de montages et de composants utilisables, mais c'est à mon sens un ensemble assez représentatif pour s'ouvrir à de nouvelles perspectives.

1. ARDUINO ET COMPAGNIE

Si vous avez une carte Arduino, c'est sans doute la façon la plus simple pour un programmeur, un informaticien ou un utilisateur d'ordinateur, de faire clignoter une simple led. Ceci se résume, comme vous devez le savoir, au chargement du croquis Blink fourni parmi les exemples basiques de l'environnement :

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

La led intégrée à la carte Arduino se mettra à clignoter lentement avec un intervalle d'une seconde. Le microcontrôleur présent sur la carte Arduino (un Atmel AVR ATmega328p dans le cas d'une Uno) exécutera

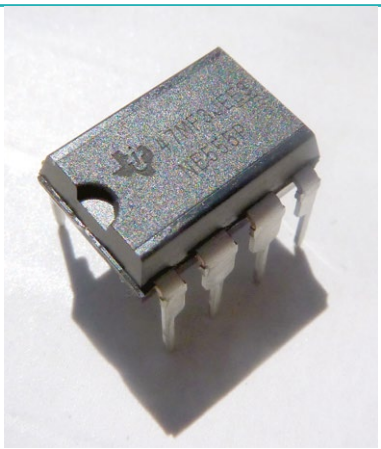


Figure 1 : Ceci est un composant légendaire depuis 45 ans, le NE555.

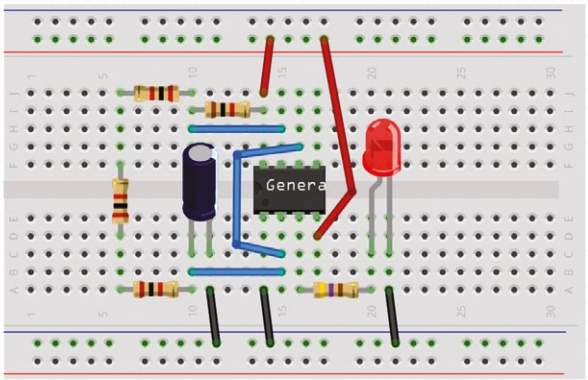


Figure 2 : Le NE555 sur platine à essais monté en configuration astable.

le programme que vous y avez placé, lui demandant de changer l'état de la sortie **13** en boucle. Cette solution est très proche de la programmation classique sur un ordinateur et c'est là toute la magie de cette plateforme.

2. NE 555

Oublions à présent le monde des microcontrôleurs, car avant que ceux-ci n'existent il y avait déjà des circuits intégrés. Tout comme un microcontrôleur, ce type de puce rassemble un groupe de transistors, de diodes, de résistances et de condensateurs dans un boîtier minuscule. Ces composants peuvent également contenir des parties électromécaniques (ce sont les MEMS pour *MicroElectroMechanical systems*, l'accéléromètre de votre smartphone est un très bon exemple). Comme le nom l'indique, il s'agit littéralement de circuits, intégrés sur une puce (*die*). Leur inventeur, Jack Kilby en 1958, s'est d'ailleurs vu décerner le prix Nobel de physique en 2000 pour ses travaux qui ont révolutionné l'industrie électronique, et accessoirement permis le perfectionnement du système de guidage des missiles nucléaires américains *Minuteman* dès 1962 (LGM-30F), le premier « produit » fabriqué en série comprenant des circuits intégrés.

À partir de la fin des années 1960, les circuits intégrés trouvent leur chemin dans bien des domaines et pour bien des utilisations. L'un d'entre eux fut le NE555, créé par Hans R. Camenzind en 1970. Ce composant est toujours utilisé de nos jours tant il est pratique et polyvalent. Il regroupe 23 transistors, 2 diodes et 16 résistances pour former deux comparateurs, un inverseur et une bascule. Le NE555 tantôt simplement appelé 555 est ce qu'on pourrait appeler un circuit de légende. Tout électronicien ou bidouilleur qui se respecte, se doit d'utiliser au moins une fois dans sa vie un NE555.

Ce composant permet énormément d'applications, aussi bien en temporisateur, en générateur d'impulsion ou en oscillateur. C'est presque le couteau suisse de l'électronique (Hans R. Camenzind est d'ailleurs né à Zurich, cela ne peut pas être un hasard). Cette polyvalence rend le NE555 utile pour bien des situations, mais celle qui nous intéresse ici est le fonctionnement en oscillateur ou en configuration astable. Les broches du composant sont référencées ainsi :

- GND : la masse ;
- TRIG : la gâchette qui déclenche la temporisation si la tension est inférieure à 1/3 de Vcc ;

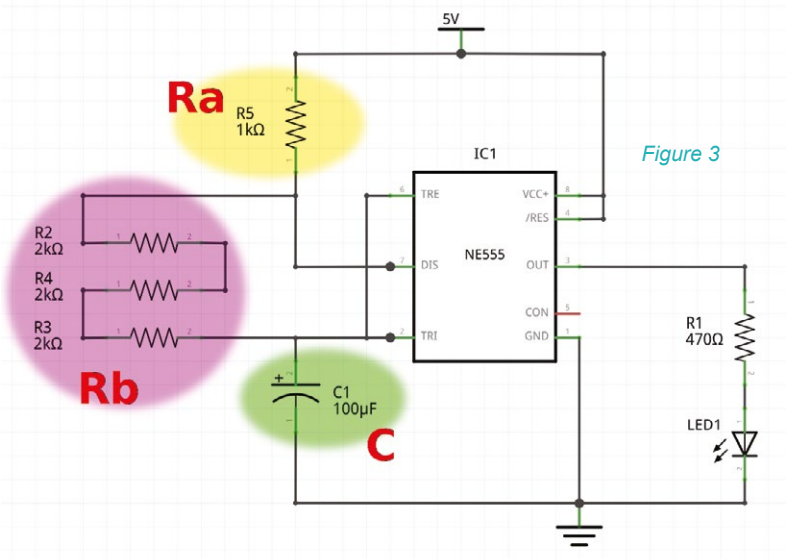


Figure 3

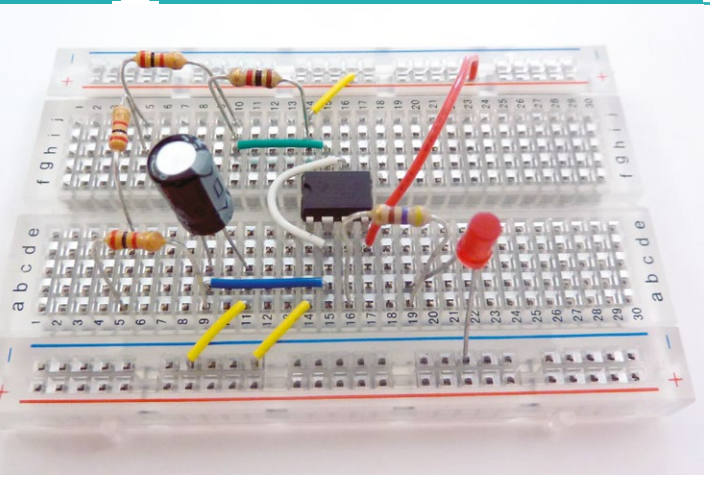


Figure 4 : Notre montage du NE555 pour test. Faute de résistance à la bonne valeur, pas le choix, on en utilise plusieurs en série.

- OUT : le signal en sortie ;
- RESET : la remise à zéro qui stoppe la temporisation ;
- CONT : accès à la tension de référence interne (à 2/3 de Vcc) ;
- THRES : seuil qui détermine la fin de la temporisation lorsque la tension est supérieure à 2/3 de Vcc ;
- DISCH : connexion du condensateur à décharger ;
- VCC : tension d'alimentation.

Le montage en configuration astable sur platine à essais est présenté en figure 2, mais il est plus simple de comprendre la connexion avec le schéma en figure 3.

Notez que j'utilise ici trois résistances de 2000 ohms en série pour un total de 6000 ohms, tout simplement parce que je n'avais pas sous la main d'autres résistances pour arriver à cette valeur (et dans *Hackable* on ne parle que de montages qu'on a effectivement fait fonctionner). Cette utilisation du NE555 permet de précisément régler la fréquence d'oscillation ainsi que le rapport cyclique (la relation entre la durée à l'état haut et à l'état bas de la sortie). C'est la relation entre la résistance de reset (en jaune), celle de décharge (en rose) et le condensateur (en vert) qui détermine ces deux paramètres.

Les formules à utiliser sont les suivantes, pour la fréquence nous avons :

Fréquence = 1,44/((Ra+2*Rb)*C)

Dans notre cas, nous pouvons calculer :

1.44/((1000+2*6000)*0,0001) = 1,107 Hz

On changera donc d'état en sortie presque toutes les secondes, mais la durée à l'état haut sera légèrement plus importante. Pour connaître ce rapport, il nous suffit de calculer :

Rapport cyclique = 1-(Rb/(Ra+2*Rb))

Ce qui nous donne ici :

1-(6000/(1000+2*6000)) = 0.5384 ~= 54 %

Le NE555 en compagnie des résistances et d'un condensateur aux bonnes valeurs permet donc de réaliser la tâche qu'on s'est fixé et, comme le microcontrôleur de la carte Arduino, il permet bien plus encore. Il pourra service de détecteur d'impulsion, de générateur PWM (variation de vitesse de moteur ou d'intensité d'une led), de générateur PPM (contrôle de servomoteur), de temporisateur séquentiel, etc.

3. CIRCUITS LOGIQUES

Le NE555 est un circuit intégré complexe et surtout capable de bien plus que ce qu'on lui demande ici. Descendons donc encore d'un cran avec un montage oscillant plus spartiate. Pourquoi ne pas utiliser de simples inverseurs comme celui présent dans le NE555 ? Un inverseur est ce qu'on appelle une porte logique et plus exactement la porte NON (*NOT*) qui complète le ET (*AND*) et le OU (*OR*). Avec ces trois portes plus la simple connexion directe (le OUI), il est possible de créer toutes les autres portes : NON-ET, NON-OU, OU exclusif (*XOR*) et NON-OU exclusif (*XNOR*).

L'inverseur peut être simple ou à bascule (*trigger*) de Schmitt. Dans ce dernier cas, le signal est inversé si la tension entrée dépasse un certain seuil et non inversé s'il descend sous un autre seuil. Sur cette base, on peut imaginer simplement une tension qui augmente de 0 volt à +5 volts. Lorsque la tension dépasse par exemple +3 volts la sortie est activée et inversée, on a 0 volt. La tension continue de monter jusqu'à +5 volts, rien ne change.

À présent, on ramène la tension doucement de +5 volts à 0 volt. Arrivant sous 2 volts par exemple, l'entrée bascule la sortie qui est inversée : nous avons +5 volts en sortie.

Comment alors arriver à utiliser cette logique pour que l'opération se répète sans fin et que nous puissions utiliser cette sortie pour commander notre led ?
Il suffit d'utiliser un condensateur et une résistance. Le condensateur fonctionne comme un élément de stockage d'énergie, il accepte du courant jusqu'à ce qu'il soit plein. En ajoutant une résistance, qui est un composant qui limite le passage du courant, on obtient un circuit RC (Résistance/Condensateur) qui possède

coromonadalix

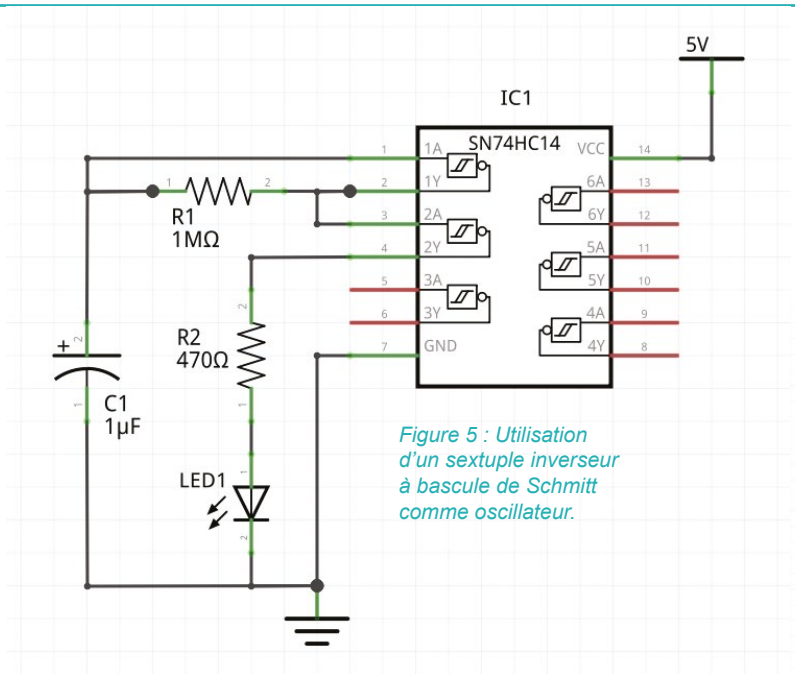


Figure 5 : Utilisation d'un sextuple inverseur à bascule de Schmitt comme oscillateur.

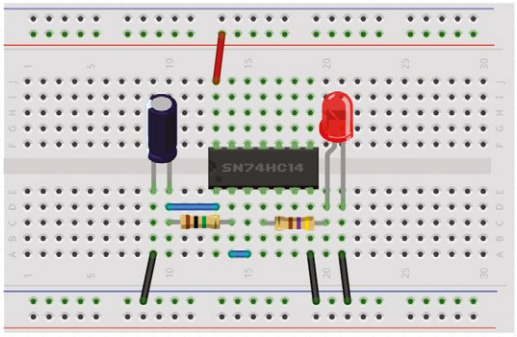


Figure 6 : Le montage sur platine à essais est plus aisé que le circuit à base de NE555, mais là encore une grande partie des fonctionnalités proposées ne sont pas utilisées.

comme caractéristique calculable le temps de charge/décharge du condensateur.

Sautons directement au montage reposant sur un circuit logique 74HC14 contenant 6 inverseurs à bascule de Schmitt que nous montons comme sur le schéma en figure 5. Ce qui correspond à l'assemblage sur platine à essais présenté en figure 6.

Les six inverseurs à bascule de Schmitt sont représentés sur le schéma ce qui facilite sa compréhension. Partons de l'état suivant lequel le condensateur n'est pas chargé. L'entrée 1A du premier inverseur est donc à 0 volt ce qui a pour effet de mettre sa sortie 1Y à +5 volts. De ce fait, le condensateur C1 va se charger via R1 jusqu'à présenter à ses bornes une différence de potentiel suffisante pour activer l'inverseur. Mais si c'est le cas, 1Y passe à 0 volt et le condensateur va se décharger dans 2A qui est l'entrée du second inverseur. Ceci jusqu'à ce que le condensateur présente une tension suffisamment basse pour que 1A soit à nouveau considéré comme non active, et le cycle se répète.

Il nous suffit d'utiliser la sortie du second inverseur, 2Y, pour brancher une led et sa résistance. Celle-ci va alors s'activer et se désactiver au rythme imposé par la charge/décharge du condensateur.

Le calcul de la vitesse de clignotement de la led est complexe, car dépendant de la valeur de R1 et C1, mais également des niveaux de tension utilisés par le modèle de circuit logique et donc les tensions de seuil de la bascule de Schmitt. Ajoutez à cela que la charge et la décharge du condensateur ne sont pas linéaires. Avec un 74AC14 de Texas Instrument (déclinaison haute vitesse du 74HC14), un condensateur de 1µF Jamicon (bonne qualité) de récupération (donc usé) et une résistance standard de 1 Mohm, on arrive grossièrement à 1 Hz (normal puisque T=R*C avec T en secondes, R en ohms et C en farads).

J'apprécie tout particulièrement ce montage pour sa simplicité. Il fait le travail qu'on lui demande même si 4 des 6 inverseurs sont inutilisés et que la précision est difficile à obtenir.

4. TRANSISTORS & MOSFET

Ce montage est un classique de l'apprentissage de l'électronique. Généralement proposé sur la base d'un circuit à deux transistors avec deux leds,

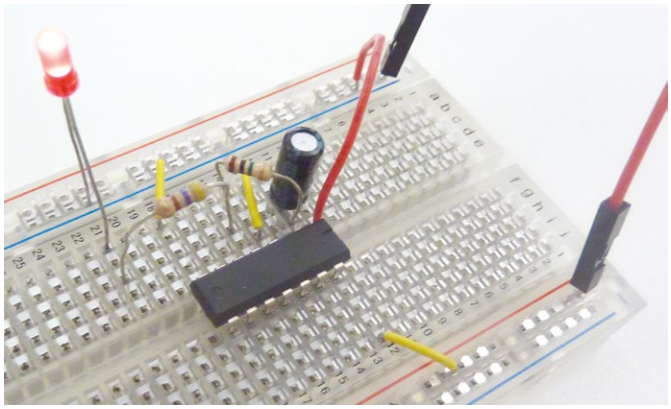


Figure 7 : L'oscillateur à base de 74ac14 en marche. Bien entendu, ça clignote...

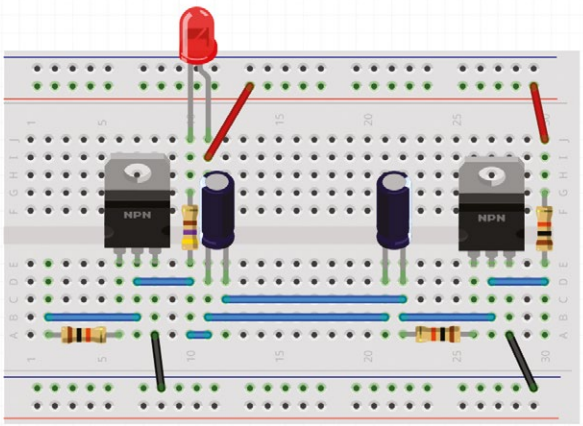


Figure 8 : De tous les circuits présentés ici, le multivibrateur astable à base de MOSFET est sans contexte le plus complexe.

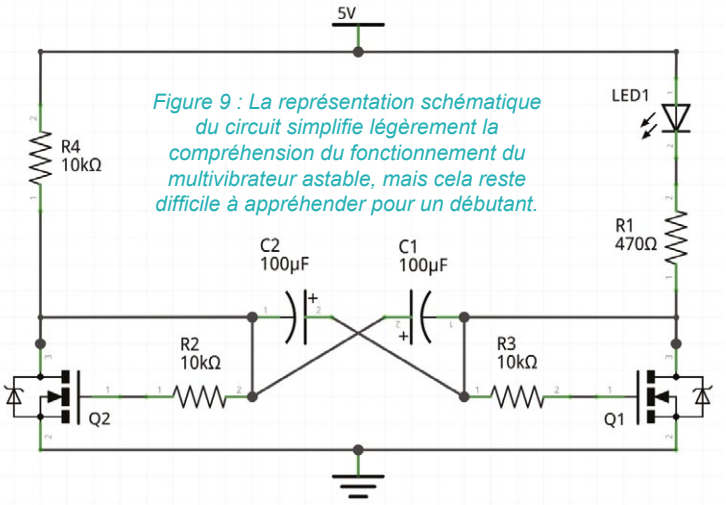


Figure 9 : La représentation schématique du circuit simplifie légèrement la compréhension du fonctionnement du multivibrateur astable, mais cela reste difficile à appréhender pour un débutant.

celui-ci est ici décrit dans sa version MOSFET-N avec une seule led pour les besoins de la démonstration. Un MOSFET est un transistor à effet de champ à grille isolée (en anglais *Metal Oxide Semiconductor Field Effect Transistor*). La raison pour laquelle je préfère ici utiliser un MOSFET-N et non un transistor bipolaire (NPN) est à la fois en raison de la simplicité de son fonctionnement et de son utilité générale.

Un MOSFET-N, comme l'IRL520, est généralement le composant utilisé, à partir d'une carte Arduino, pour contrôler un circuit utilisant un courant ou une tension plus importante (moteur, éclairage, ventilateur, etc.). Le transistor bipolaire est un composant contrôlé en courant alors que le MOSFET est piloté en tension. Sa conception permet de facilement le contrôler en appliquant une tension supérieure à sa tension de seuil, il commence alors à devenir conducteur. Piloté par une carte Arduino en 5 volts, l'IRL520 agit presque comme un interrupteur (attention, ce n'est pas le cas du IRF520 qui avec une tension de +5 volts entre la grille et la source ne laisse pas passer son maximum de courant). De ce fait lorsqu'on parle d'électronique numérique il est plus intéressant, dans un premier temps, de posséder en quantité des MOSFET comme l'IRL520 que des transistors bipolaires. Ainsi, au-delà de la démonstration qui suit, ces MOSFET pourront plus certainement vous servir à autre chose.

Le circuit permettant de faire clignoter une led avec des MOSFET ou des transistors bipolaires est appelé un multivibrateur astable. Ce n'est bien entendu pas la seule solution, mais c'est certainement la plus connue. L'assemblage sur platine à essais est présenté en figure 8, mais il est bien plus aisé d'en comprendre le fonctionnement en étudiant le schéma électronique correspondant, figure 9.

Pas de panique, c'est bien plus simple qu'il n'y paraît. Il suffit de partir d'un des deux états dans lequel peut se trouver le circuit.

Nous décrétons donc que le MOSFET Q1 est conducteur et que le condensateur C2 est chargé. De ce fait, C2 se vide graduellement de son énergie par la grille du MOSFET au travers de la résistance R3. Pendant ce temps, le MOSFET Q1 laisse passer le courant entre +5V et la masse et donc au travers de la led et de la résistance R1. En effet, la tension entre la grille Q1 et la source reliée à la masse est suffisante pour rendre le MOSFET conducteur.

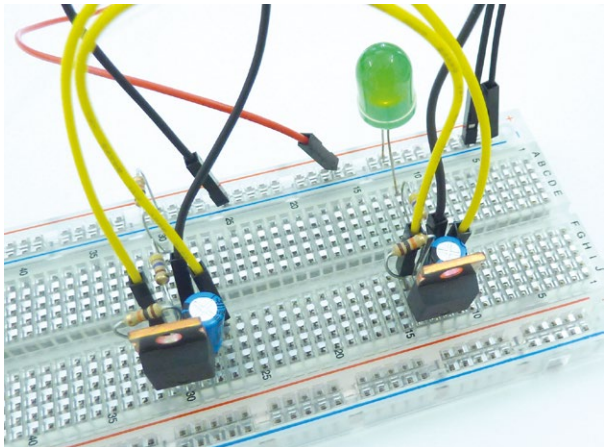
Q2 à ce moment ne conduit pas entre R4 et la masse. Le courant arrive donc au condensateur C1 qu'il charge petit à petit.

Une fois que C2 est presque vide, il n'y a plus une tension suffisante entre la grille de Q1 et la masse, le MOSFET cesse de conduire le courant. En conséquence, la led s'éteint, C2 commence à se recharger et C1 se décharge doucement via R2 dans la grille de Q2. Ce MOSFET devient conducteur et le courant passe entre R4 et la masse jusqu'au moment où C1 est suffisamment vide pour que la tension ne soit plus suffisante entre la grille et la source reliée à la masse. On se retrouve avec un C2 chargé et tout recommence...

5. MORALITÉ ET CONCLUSION

Voici venue l'heure du bilan concernant les différentes solutions pour notre hypothétique projet de clignoteur à led. Bien sûr, tout ceci

Figure 10 : La mise en pratique du montage sur platine est presque aussi compliquée que son principe de fonctionnement.



coromonadalix

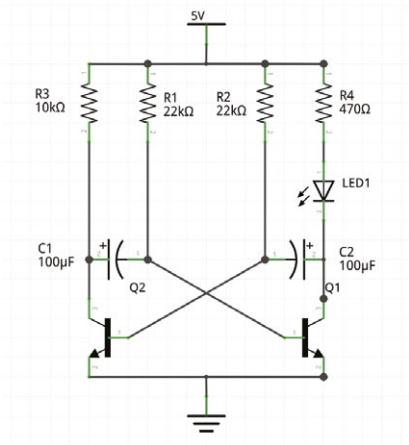


Figure 11 : Un circuit assez similaire et qu'on retrouve un peu partout sur le web, utilise des transistors NPN, c'est un classique du genre.

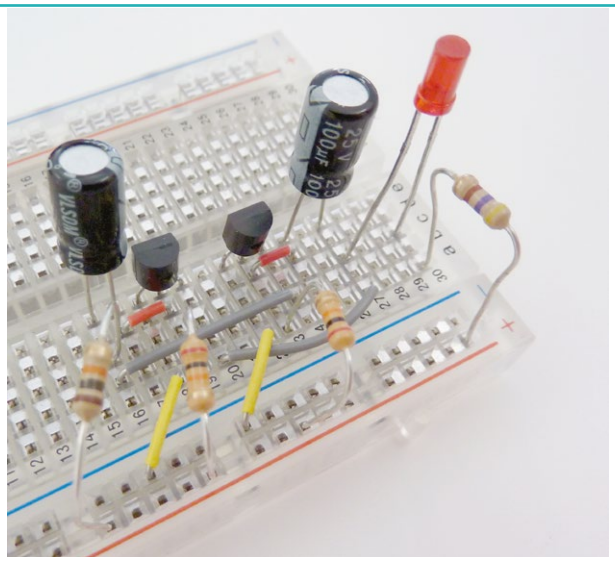


Figure 12 : Le montage à base de transistors bipolaires NPN, ici des BC547, est un classique du genre et une étape presque incontournable si l'on souhaite explorer le monde de l'électronique.

n'est qu'une excuse pour apprendre l'électronique et retenir ce qui est sans doute la leçon la plus importante pour tout bidouilleur : il y a toujours plusieurs solutions pour arriver à ses fins. Dans la « vraie vie » si on veut qu'une led clignote bêtement, sans contrôle, on opte pour... une led clignotante.

Ceci cependant n'empêche pas de comparer les solutions, en particulier au niveau de leur coût. La méthode la plus chère est sans le moindre doute la carte Arduino. Une simple Uno implique une dépense de quelques 25 euros. Ce prix chutera en optant pour un clone et d'autant plus encore en se concentrant sur le microcontrôleur seul (voir article sur la programmation ISP). Un ATmega328p revient à environ 3,70€, mais on peut revoir les caractéristiques à la baisse (32 Ko de flash ce n'est pas vraiment dans le cahier des charges) et se diriger vers un ATTiny85 (1,80€) ou un ATTiny45 (1,50€).

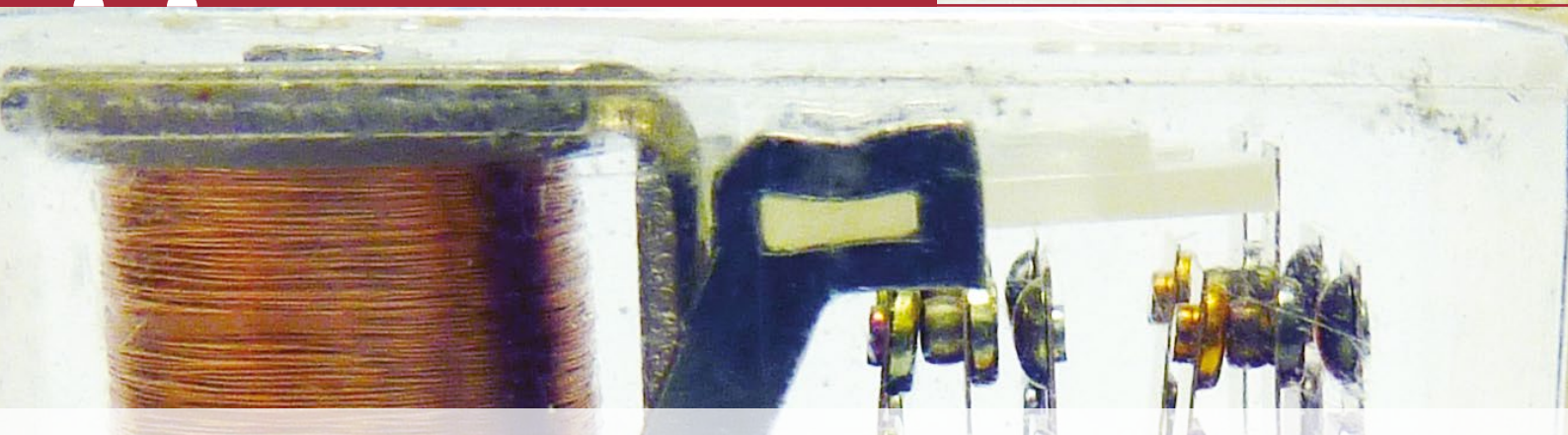
En quittant le monde des microcontrôleurs et en utilisant un NE555, on passera directement sous la barre de l'euro avec un prix généralement constaté pour un format PDIP (utilisable sur platine à essais) autour de 0,65€. Ce à quoi il faut cependant ajouter le prix des composants passifs (résistances + condensateur).

Le choix du circuit logique ne change pas grand-chose avec ses 0,60€ et une résistance en moins. Et enfin, l'utilisation de MOSFET-N IRL520 à environ 1 euro pièce, finalement, inverse la tendance en faisant à nouveau augmenter le coût. Bien entendu, troquer les MOSFET pour de simples transistors NPN comme des BC547 divise par quatre le coût à l'unité et on retrouve alors un montage sous la barre fatidique de l'euro.

Que faut-il finalement retenir de tout cela ? Simplement que la solution à un problème est totalement dépendante de ce dernier et des conditions autour de ce dernier. Utiliser une carte comme l'Arduino Uno revient ici à littéralement déployer une architecture de 16 MIPS (Million d'Instructions Par Seconde) pour une led. Mais attention, si la led doit clignoter selon le résultat d'un calcul découlant d'une communication avec un capteur sur une liaison SPI, le tout en fonction d'une date obtenue d'une RTC en i2c, ça change tout. Pour un projet donné, il n'est pas possible de séparer les éléments de l'ensemble et de les traiter individuellement. Sinon, la solution consisterait à ajouter un montage à base de 74hc14 en plus de l'Arduino, juste pour la led. Ce qui n'a aucun sens.

Il faut donc correctement estimer ses besoins, borner et quantifier les ressources nécessaires et prévoir une marge de manœuvre. L'exercice vous paraît difficile ? C'est normal, il l'est. Et même les professionnels vous le diront, il n'est pas rare qu'en plein milieu d'un projet il soit nécessaire de revoir sa copie et d'envisager purement et simplement un changement de plateforme. Pire encore, de mauvais choix peuvent rendre un produit obsolète très rapidement après sa mise sur le marché. Aujourd'hui, nous sommes en plein boum de l'internet des objets et dans l'effervescence ambiante, nombreux sont ceux qui ne prévoient pas que dans quelques années (peut être moins) HTTPS sera la norme et que de nombreuses plateformes ne disposent ni de la puissance ni de la mémoire nécessaire au chiffrement des communications avec SSL/TLS. Tous ces objets seront alors officiellement obsolètes, voire dangereux et devront être remplacés. S'il s'agit de votre montage « maison », vous basculerez simplement sur une nouvelle plateforme puisque vous avez la maîtrise du fonctionnement. S'il s'agit d'un produit « fermé », vous n'aurez d'autre choix que d'en acheter un nouveau. À condition, bien sûr, que le nouveau soit effectivement plus sûr que l'ancien... **DB**

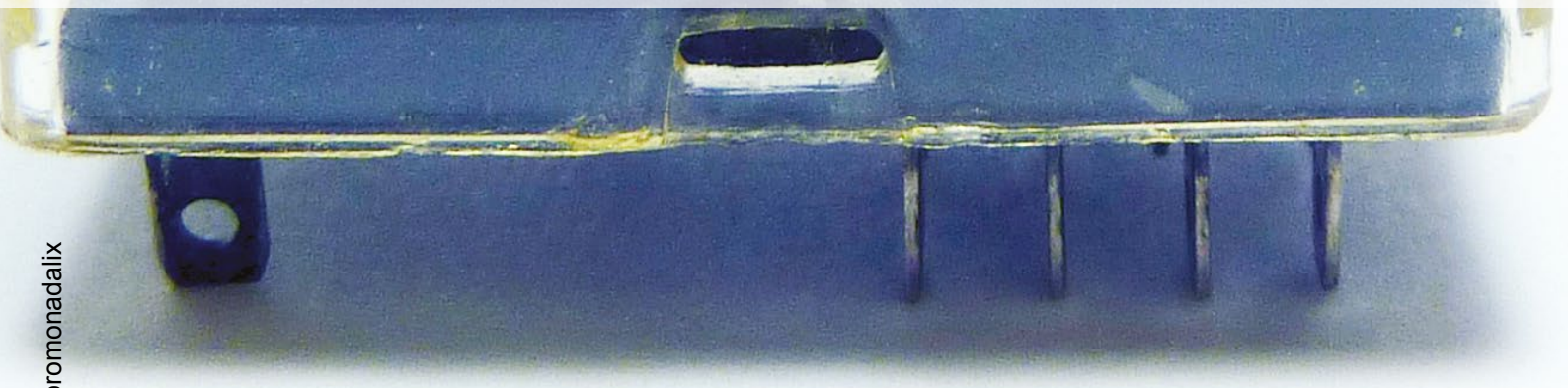
coromonadalix



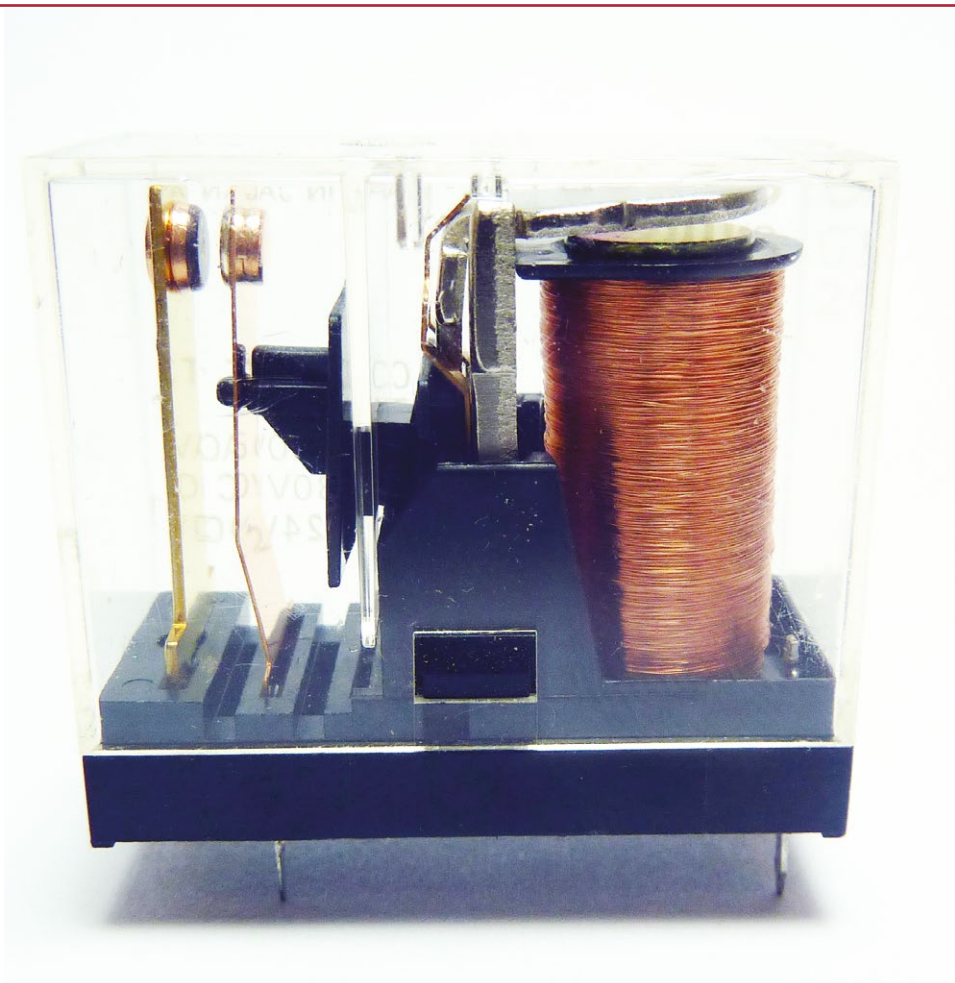
CONTRÔLER UN APPAREIL DOMESTIQUE : 230 VOLTS !

Denis Bodor

Dès lors qu'il s'agit de domotique, il est question du contrôle d'appareils et d'équipements fonctionnant avec un courant et une tension dangereux et potentiellement mortels. Il n'est pas question de faire n'importe quoi avec les quelques 230 volts circulant par nos prises murales et alimentant tous nos objets du quotidien. Au-delà du bon sens, un certain nombre de règles de sécurité s'appliquent et certains composants doivent être utilisés. L'un d'entre eux est le relais. Voyons cela ensemble...



coromonadalix



Les relais en boîtiers transparents permettent de facilement comprendre leur fonctionnement (en plus d'être vraiment jolis). Sur la droite, la bobine avec juste au-dessus une partie mobile en métal ferreux. Celle-ci, lors de l'activation, est attirée par la bobine et déplace un support plastique (noir au centre). Lui-même pousse sur une lamelle du contacteur en platine à gauche qui agit comme un interrupteur.

Avant toutes choses, rappelons que l'électricité disponible dans nos lieux de vie utilise un courant alternatif avec une tension de 230 volts (oui, ça fait plus de 30 ans que ce n'est plus 220V). Ce courant est distribué dans une installation répartie en différents circuits, tous ayant pour origine un tableau de répartition se trouvant juste « derrière » le compteur électrique de votre fournisseur d'énergie. Ces circuits sont installés par un technicien spécialisé en suivant les règles imposées par les normes d'usage. Ces normes limitent le nombre de prises par circuit, de lampes (éclairage plafond) et de circuits spécialisés. Chaque circuit et/ou prise spéciale (four, sèche-linge, etc.) est protégé par un disjoncteur (16A pour les prises, 10A pour les luminaires, 20A ou 32A pour les prises spéciales).

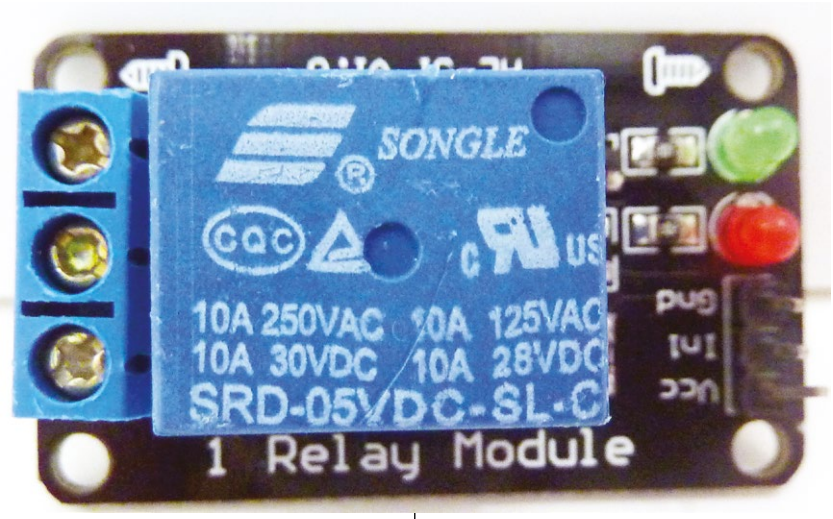
Le disjoncteur a pour objectif de couper le courant dès lors que celui-ci dépasse une certaine valeur. Cette fonction, anciennement assurée par les fusibles, permet de protéger des surcharges (trop d'appareils sur un circuit) et des courts-circuits (connexion phase/neutre ou phase/terre). À cela s'ajoutent un ou plusieurs interrupteurs différentiels (ou disjoncteurs différentiels) qui eux aussi assurent une protection contre les surcharges et les courts-circuits, mais ajoutent une détection d'une fuite de courant. Dans un circuit « normal », la quantité de courant qui arrive sur un conducteur (la phase) doit forcément revenir sur l'autre (le neutre). Si ce n'est pas le cas, cela signifie que quelque part dans le circuit une partie du courant passe ailleurs et donc à la terre, potentiellement à travers une personne. Si une différence est détectée, de plus de 30 mA selon la norme française, l'interrupteur différentiel coupe le courant.

Pourquoi tant de protections ? La réponse est évidente : l'électricité domestique est dangereuse et peut tuer. On notera au passage qu'il y a souvent une mauvaise utilisation des termes concernant ce risque. Techniquement, vous ne pouvez pas dire que vous vous êtes électrocuté. En effet, l'électrocution désigne exclusivement une électrisation entraînant un décès. Après une électrisation vous pouvez peut-être encore parler, avec une électrocution c'est assez improbable...

Ce n'est pas tout, une autre « légende urbaine » véhicule une fausse vérité : « *ce n'est pas la tension qui tue, c'est le courant* ». Les choses sont plus complexes que cela, il suffit de se souvenir de la loi d'Ohm. Le corps humain est constitué majoritairement de fluides conducteurs, mais la

coromonadalix

HAUTE TENSION



Les relais possèdent généralement des inscriptions permettant de rapidement connaître leurs caractéristiques. Ici nous apprenons que ce modèle peut être utilisé pour contrôler un circuit en 250V/10A alternatif (AC) ou 30V/10A en courant continu (DC). Une consultation de la documentation constructeur est cependant fortement recommandée.

peau présente une résistance importante. Pour qu'un courant suffisant pour être dangereux puisse passer dans un corps humain, il faut donc une tension suffisamment importante pour traverser la peau, sinon il ne se passe rien. Inversement, et c'est sans doute la source de la fausse vérité, l'électricité statique, produite par exemple par des chaussettes en laine sur une moquette en polyester (matériaux triboélectriques), génère des décharges de haute tension, mais en très peu de temps. L'électricité passe effectivement au travers de la peau, mais trop peu de charges circulent pour causer des dommages. En conclusion, ce n'est pas QUE le courant qui est fatal, mais le courant, la tension et la durée d'exposition. Quoi qu'il en soit, 230V et 10A sont largement plus qu'il n'en faut pour vous tuer, vous blesser ou déclencher un incendie.

J'espère vous avoir fait suffisamment peur pour attirer votre attention sur ce qui va suivre.

1. LE MAÎTRE MOT : ISOLATION

Si vous comptez contrôler un appareil fonctionnant en 230V, comme une lampe ou une machine à café, à partir d'une carte Arduino ou Raspberry Pi, il est évident que vous devrez travailler sur le côté haute tension du montage. Avant toutes choses, une règle est à respecter (et elle est également valable pour les montages en basse tension) : NE MANIPULEZ **JAMAIS** UN CIRCUIT SOUS TENSION ! JAMAIS !

Il y a des situations exceptionnelles où on n'a pas le choix, bien sûr, mais avant d'en arriver à ces extrémités dangereuses, il faut avoir exploré toutes les autres solutions, et le faire avec la plus grande prudence. Généralement, la règle de « *une main dans la poche* » est un bon début (ceci évite de toucher les deux pôles par mégarde, avec le cœur au milieu). Il va sans dire également qu'un montage contrôlant l'électricité domestique, lorsqu'il est en marche, se doit d'être placé dans un boîtier et non laissé à l'air libre.

Le reste tourne autour d'un simple mot : isolation. L'idée consiste à séparer électriquement la partie basse tension de la partie haute tension. Ceci protège à la fois le matériel et limite le risque pour les personnes. Pour cela, on évite simplement de fournir un chemin, pour le courant côté haute tension, qui lui permettrait de passer côté basse tension et faire des ravages. Le composant le plus accessible et le plus pratique pour une telle isolation est le relais.

coromonadalix

Il s'agit d'un composant électromécanique composé d'une bobine et d'un interrupteur. La bobine, lorsqu'elle est traversée par un courant, crée un champ électromagnétique capable d'attirer les métaux ferreux. Cet électroaimant permet de déplacer un dispositif qui agit sur un interrupteur. Il n'y a pas de liaison électrique entre la partie bobine et la partie interrupteur.

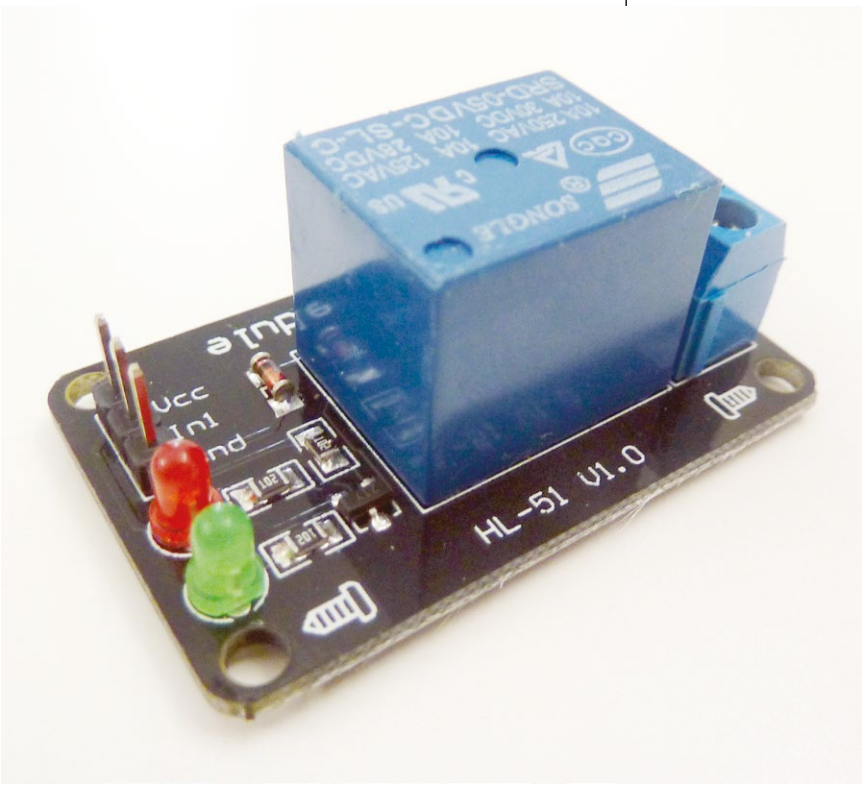
Généralement, un relais se pilote avec un circuit prévu à cet effet. La bobine d'un relais courant utilisant 5 volts présente souvent une résistance de l'ordre de 70 ohms. Un rapide calcul (U=R.I, donc I=U/R) montre qu'on parle d'un courant de quelques 70mA, largement au-delà de ce que peut fournir ou accepter une carte Arduino ou Raspberry Pi. On doit donc utiliser un transistor ou un MOSFET pour contrôler l'alimentation de la bobine du relais.

Nous disposons avec un relais d'une isolation entre les deux « mondes », mais ceci est souvent complété d'un autre type d'isolation. En effet, en cas de problème avec le relais qui reste un composant mécanique pouvant s'user, si le courant haute tension passe dans la bobine, il risque de remonter vers le circuit de contrôle puis dans le reste du montage (Arduino, etc.). On décide donc d'isoler le circuit alimentant la bobine du relais du reste du montage avec un optocoupleur.

Ce composant est constitué d'une led infrarouge d'un côté et d'un phototransistor de l'autre, le tout dans un boîtier. Il n'y a aucun contact électrique entre les deux éléments. Lorsqu'un courant passe dans la led, celle-ci s'allume et éclaire le phototransistor. Ce dernier se comporte exactement comme un transistor, mais au lieu d'être contrôlé par un courant à sa base, il l'est par la réception des infrarouges. Dès que la led est allumée, le phototransistor laisse passer le courant. Là, il n'y a aucune partie mobile, tout est entièrement optique et contenu dans un seul composant.

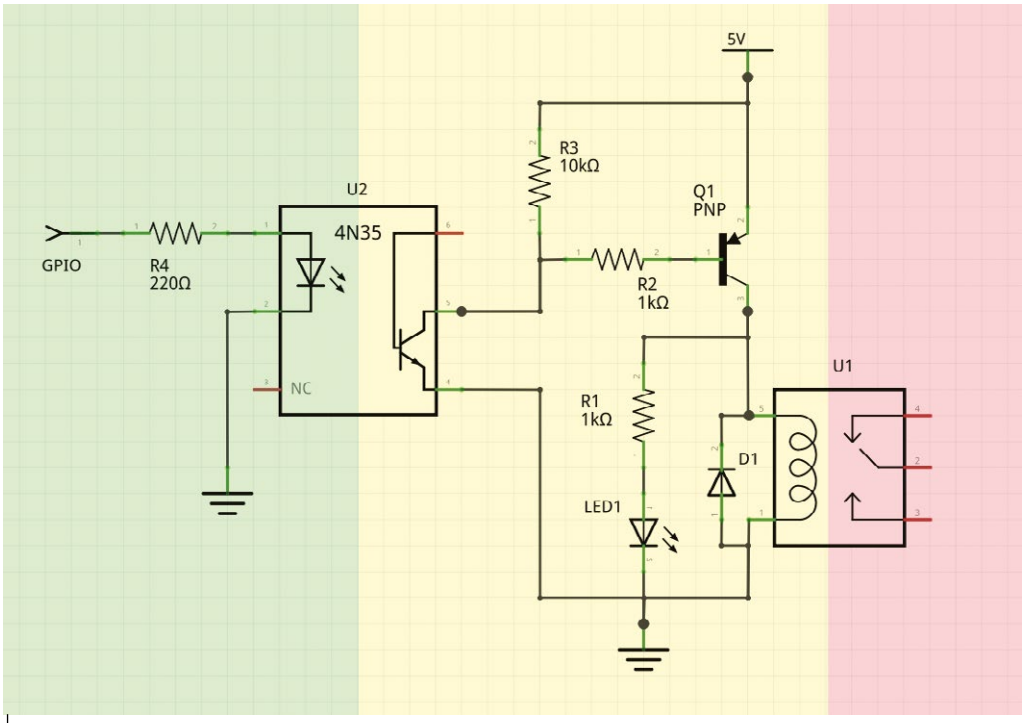
On utilise ainsi l'optocoupleur comme une led classique côté Arduino/Pi pour contrôler le circuit d'alimentation du relais qui lui-même contrôle la haute tension. Il faut, bien entendu, n'établir, par ailleurs aucune autre liaison électrique entre les deux côtés du montage. Je pense en particulier à l'alimentation du relais qui ne peut être un courant fourni par l'Arduino ou la Raspberry Pi. Ceci fonctionnera, bien sûr, mais vous n'avez plus d'isolation et l'optocoupleur ne sert alors à rien.

Nous pouvons résumer ce montage par le schéma présenté ici. Il repose sur un module chinois très courant comprenant un relais 5V, un transistor PNP, quelques



Les modules utilisés ici, vendus à un prix dérisoire, intègrent un relais, un transistor, deux leds, une diode, quelques résistances, un bornier et un connecteur permettant l'alimentation et le contrôle. Il manque l'optocoupleur, mais à 1€ le module on ne va pas faire la fine bouche...

coromonadalix

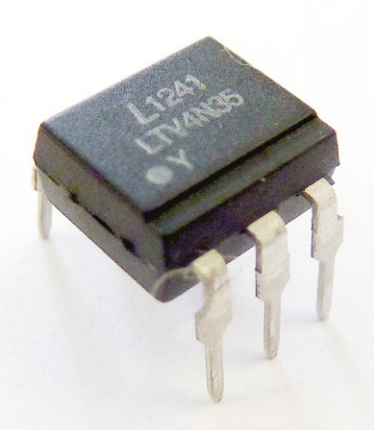


Le montage typique permettant de contrôler un appareil 230V (zone rouge) avec un relais 5V (zone jaune) et une carte utilisant, par exemple 3,3V (zone verte). On voit clairement que les trois espaces sont électriquement isolés les uns des autres.

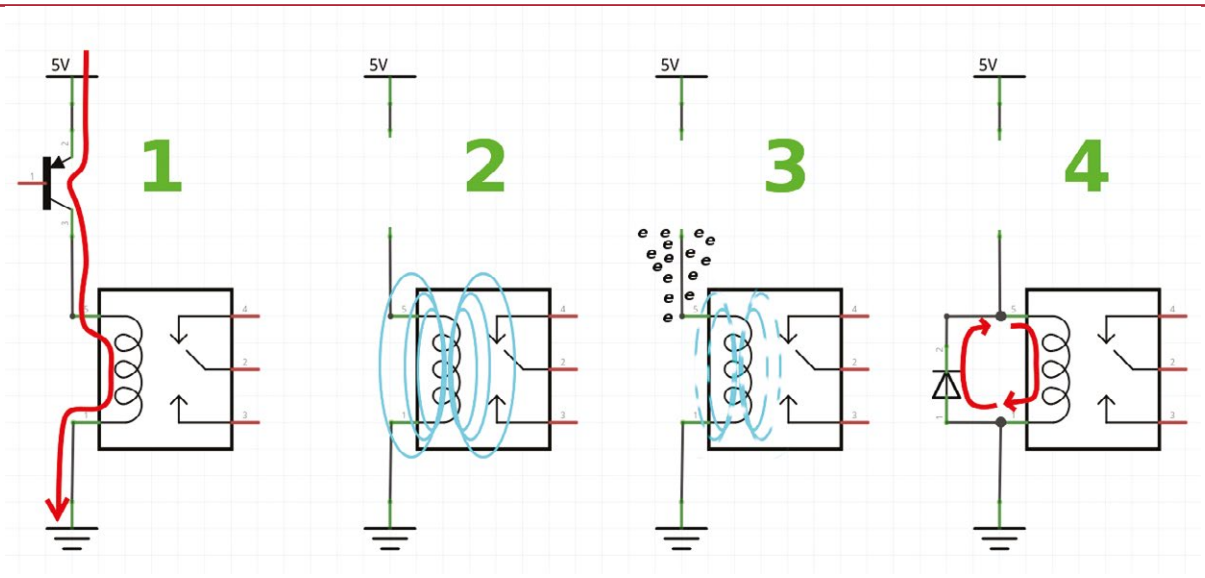
résistances, une diode et deux leds. Une seule est présentée ici pour simplifier le schéma, la seconde est simplement le témoin d'alimentation relié entre +5V et la masse via une autre résistance de 1 Kohm. Ces modules sont commercialisés à 1 euro pièce (port offert) par un vendeur appelé *hohfashion* et expédié depuis Hong Kong.

U2 sur la gauche est un optocoupleur 4N35 très courant qui ne fait pas partie du module. Il en va de même pour la résistance R3 ajoutée pour inverser le signal. En effet, nous avons ici un transistor PNP qui contrôle le passage du courant entre l'émetteur (haut) et le collecteur (bas) en fonction du courant qui circule entre la base (gauche) et le collecteur. Dans les grandes lignes, lorsqu'il n'y a pas de courant à la base, le transistor conduit, et inversement est bloquant lorsqu'un courant circule à la base. Ceci signifie qu'en mettant R2 à +5V le transistor, et donc le relais, est ouvert. Pour l'activer, il faut mettre R2 à la masse.

Ceci est assez contre-intuitif pour un croquis Arduino par exemple. Le fait d'ajouter l'optocoupleur avec la résistance R3 inverse donc le signal en plus d'isoler les deux côtés. Lorsque le phototransistor n'est pas conducteur, tout se passe comme s'il était absent et un courant circule à la base du transistor



L'optocoupleur intègre une diode infrarouge et un phototransistor réunis dans un boîtier DIP miniature. Ceci est un classique 4N35, mais il existe également des versions intégrant plusieurs composants pour piloter plusieurs circuits indépendamment.



1) Lorsque le relais est alimenté via le transistor, il circule au travers de la bobine dans le sens conventionnel du courant, du (+) vers le (-), et un champ magnétique se forme autour de la bobine. Ceci n'est pas instantané, une bobine, comme un condensateur, s'oppose aux variations du courant, car le champ magnétique se construit et une tension apparaît aux bornes de la bobine.

2) Une fois le champ magnétique présent, il permet de faire fonctionner le mécanisme du relais. Mais si l'on coupe l'alimentation, ce champ magnétique est toujours présent. Ici, j'ai simplement supprimé le transistor du circuit pour illustrer la rupture d'alimentation, mais le principe est le même avec un transistor non conducteur.

3) Le champ magnétique s'effondre progressivement et provoque l'apparition d'un courant dans la bobine. Rappelez-vous que le sens de circulation des électrons est inverse de celui du sens conventionnel. Ce sont des porteurs de charge négative. Ces électrons vont alors s'accumuler à une borne de la bobine sans pouvoir s'échapper. Le courant veut circuler et la tension va alors drastiquement augmenter. Le transistor voit alors entre le collecteur et l'émetteur une tension qui peut dépasser celle qu'il peut tolérer (tension de claquage) et a toutes les chances d'être détruit.

4) En ajoutant une diode adaptée à la bobine et aux tensions qui apparaissent, nous offrons au courant induit un chemin pour circuler et l'énergie sera rapidement dissipée entre la bobine et la diode. Il n'y a plus de surtension apparaissant aux bornes de la bobine et le transistor ne risque plus rien.

Q1. Mais quand le phototransistor conduit parce qu'il est éclairé par la led interne au 4N35, il n'y a plus assez de courant à la base Q1, qui devient conducteur et le relais est activé. Nous retombons sur nos pieds, un niveau haut côté Arduino correspond à l'activation du relais.

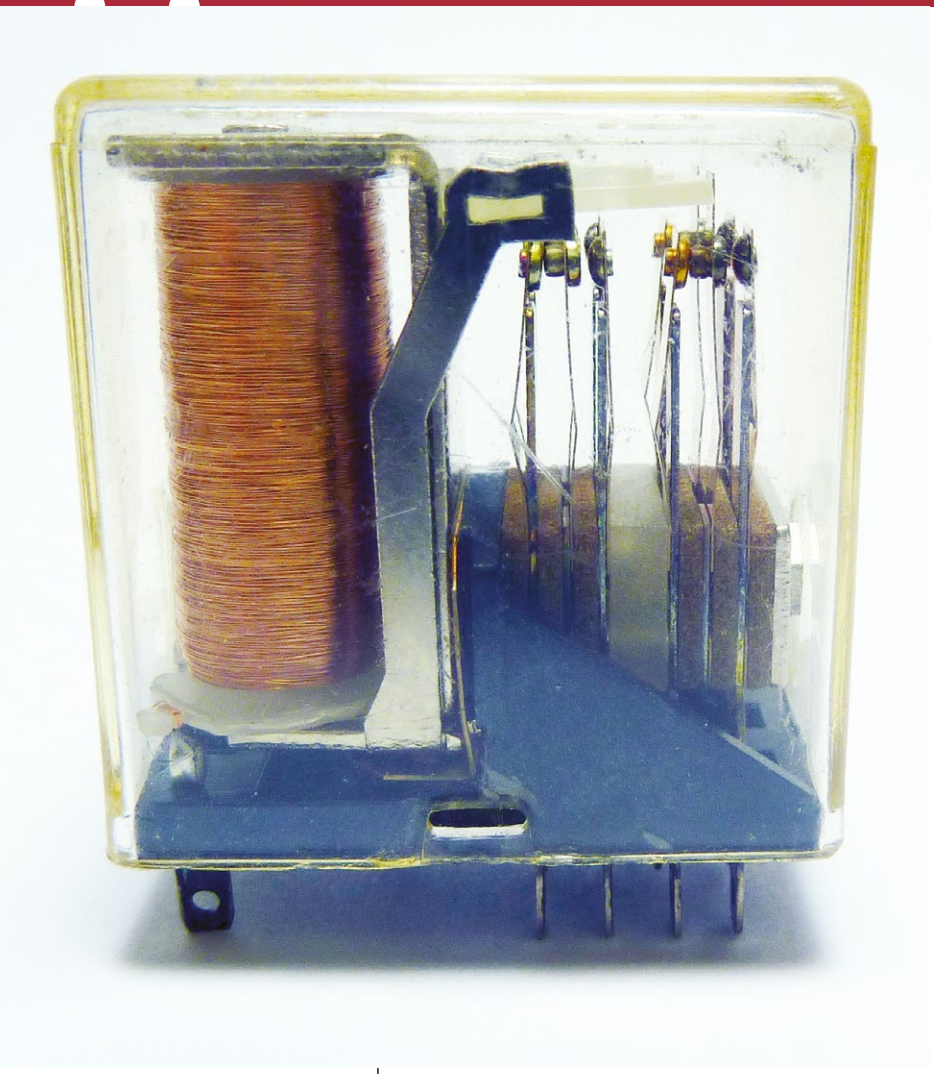
Ces modules très peu chers peuvent en principe être utilisés sans optocoupleur, mais avec le temps et l'expérience, on finit par respecter les règles de sécurité à la lettre pour éviter les problèmes. Mère nature ne fait pas de cadeau et lorsqu'il s'agit de hautes tensions, on se montre prudent. Notez qu'il existe bien d'autres modèles de modules, intégrant parfois directement un ou des optocoupleurs. Il est en effet courant de trouver des cartes complètes avec 4 ou 8 relais, toutes équipées, ne nécessitant donc aucun ajout.

2. À PROPOS DE LA DIODE

Peut-être avez-vous remarqué la diode D1 juste à côté du relais U1 sur le schéma. Celle-ci est désignée sous les termes « *diode de roue libre* » et vous la retrouverez dans de nombreux montages incluant une bobine ou self (relais, moteur, etc.). Cette diode permet de se protéger des surtensions apparaissant à la rupture d'alimentation d'une bobine.

Il faut en effet savoir que n'importe quel conducteur métallique, lorsqu'il fait circuler un courant électrique, fait apparaître un champ magnétique autour de lui. C'est le principe même de la bobine faisant office d'électroaimant dans le relais. L'effet est simplement démultiplié, car le conducteur en question est enroulé autour d'un cœur en fer de multiple fois.

coromonadalix



Certains relais disposent de plusieurs contacteurs. Il est alors possible avec une seule bobine de contrôler plusieurs circuits indépendants. Remarquez l'ingéniosité et la beauté du dispositif qui méritent clairement d'être mises en valeur dans un montage dans la mesure du possible (sécurité avant tout).

L'apparition du champ magnétique n'est pas instantanée, il se construit progressivement jusqu'à son maximum au fur et à mesure de la circulation du courant. Que se passe-t-il lorsqu'on cesse d'alimenter la bobine ? Tout simplement l'opération inverse. Le champ magnétique s'effondre et un courant circule dans le conducteur. Aux bornes de la bobine apparaît alors une différence de potentielle importante dépassant largement la tension initiale appliquée.

Ce courant doit aller quelque part et on parle ici d'une tension pouvant atteindre des dizaines de volts. En plaçant une diode dans le sens opposé du flux de courant, on crée un chemin pour que cette énergie résiduelle puisse circuler et se dissiper entre la bobine et la diode. La surtension est éliminée et il n'y a plus de risque de détruire les autres composants du circuit (et le transistor en particulier).

3. POUR FINIR

Je pense qu'on ne le répétera jamais assez : faites attention, soyez prudent, ne manipulez pas un circuit sous tension, isolez les différents niveaux de tension, ne laissez pas un montage à l'air libre, n'utilisez pas de 230V sur une platine à essais et respectez les valeurs spécifiées sur le relais. Les modules utilisés ici intègrent un relais SONGLE sur lequel est marqué :

- 10A 250VAC : 10 ampères maximum en 250 volts courant alternatif ;
- 10A 30VDC : 10 ampères maximum en 30 volts courant continu.

Ce sont des valeurs absolues à ne pas dépasser, ni le courant, ni la tension. Pas question donc de contrôler un four, une plaque de cuisson ou tout autre appareil en 230V susceptible d'utiliser plus de 10A avec ce type de relais. Ceci signifie également que la tension utilisable en courant alternatif est bien plus élevée qu'en courant continu (30 volts ici). Dépasser la tension c'est risquer la formation d'un arc électrique. Dépasser le courant maximum implique la possibilité que le relais chauffe, que les lamelles fondent et que le tout prenne littéralement feu.

Le relais est également marqué de la tension d'utilisation, 05VDC, correspondant à 5V. Attention cependant, le marquage des composants n'est pas toujours explicite et mieux vaut se référer à la documentation constructeur en PDF avec une petite recherche de la marque et du modèle sur le web. **DB**

coromonadalix

NE MANQUEZ PAS LINUX PRATIQUE HORS-SÉRIE N°33 !

DOSSIER SPÉCIAL :

PHP

7 JOURS POUR APPRENDRE À PROGRAMMER EN PHP !

coromonadalix

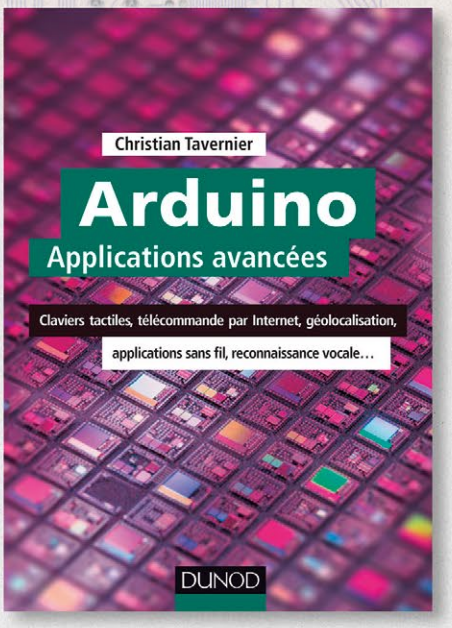
DISPONIBLE SUR :

www.ed-diamond.com

LES OUVRAGES PRATIQUES POUR LES MAKERS

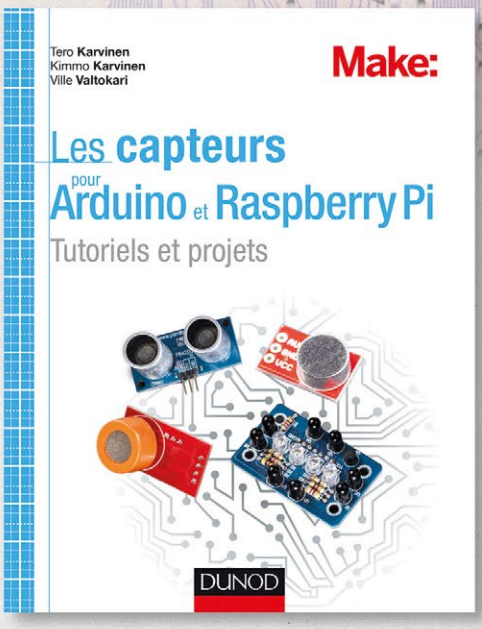
9782100710409, 232 p., 24,90 €
Christian TAVERNIER

Tous les éléments nécessaires à la conception et à la mise en œuvre de nombreuses applications performantes avec Arduino.



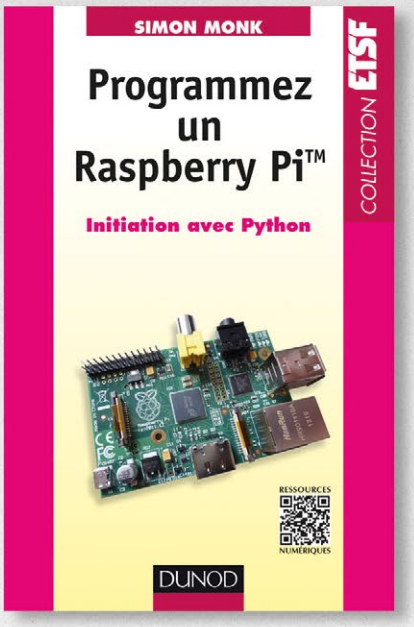
9782100582051, 224 p., 27 €
Christian TAVERNIER

Pour ceux qui souhaitent développer des applications évoluées à base d'Arduino.



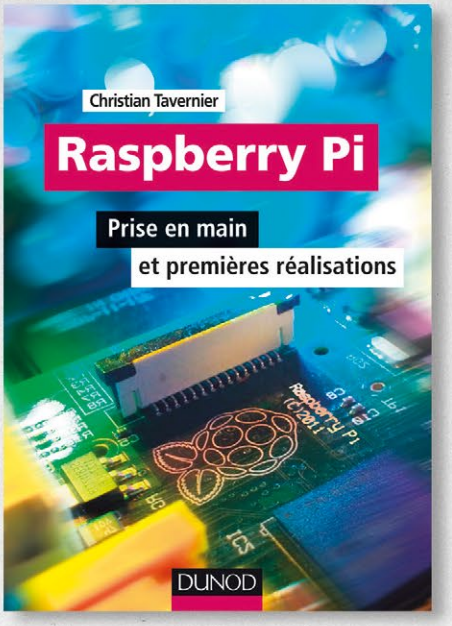
9782100717934, 304 p., 29,90 €
Tero KARVINEN et al.

Bien utiliser les capteurs pour concevoir des montages avec Arduino ou Raspberry Pi qui interagissent avec leur environnement.



9782100706594, 192 p., 16,90 €
Simon MONK

Cet ouvrage s'adresse à ceux qui découvrent le Raspberry Pi et leur explique en termes simples comment écrire des programmes à l'aide du langage Python.



9782100598915, 224 p., 19,90 €
Christian TAVERNIER

Configurez, paramétrez et découvrez les nombreuses possibilités du « micro-ordinateur » Raspberry Pi.

TOUT LE CATALOGUE SUR **DUNOD.COM**

DUNOD
ÉDITEUR DE SAVOIRS