CANARD PC HARDWARE

TESTS, COMPARATIFS, ANALYSES, CONSEILS

CANARD PC HARDWARE

TOUT SAVOIR POUR BIEN CHOISIR

N°07 - NOVEMBRE - DÉCEMBRE 2016

HORS-SÉRIE

COMPOSANTS

Des résistances aux transistors

Comment fonctionnent-ils ? Lesquels choisir ?

THÉORIE

Tension, courant, puissance

Les notions de base expliquées simplement

PLATEFORME

S'initier aux Arduino

De la théorie à la pratique

MICROCONTRÔLEURS

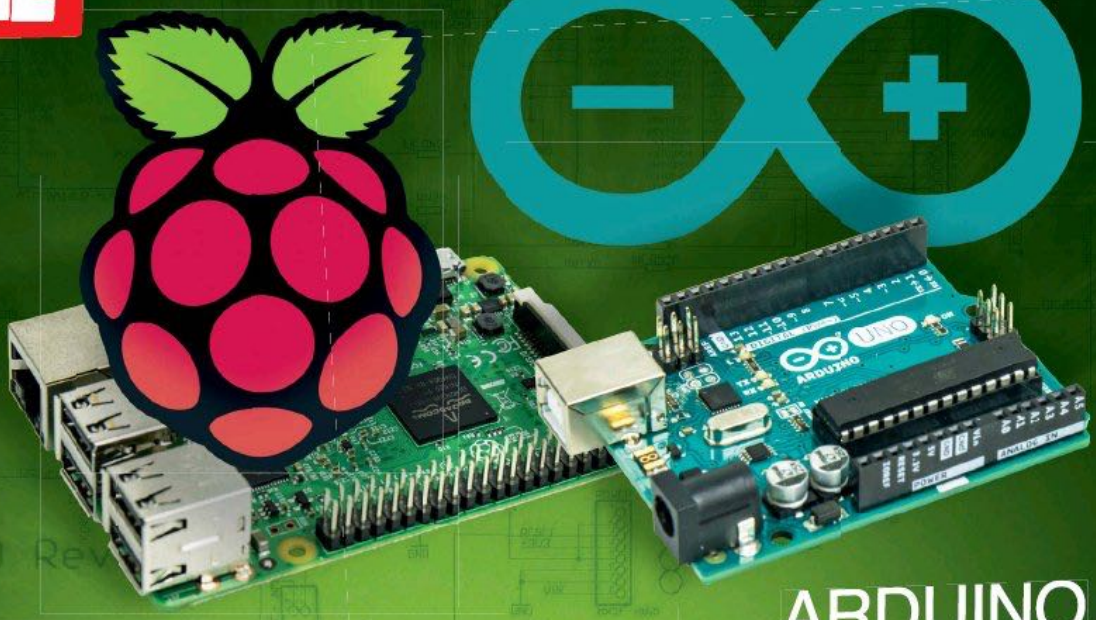
Entrées, sorties et GPIO

Analogique et numérique

PLATEFORME

Le Raspberry Pi en électronique

La carte à tout faire



ARDUINO & RASPBERRY PI

SPÉCIAL Électronique

Tout pour bien débuter

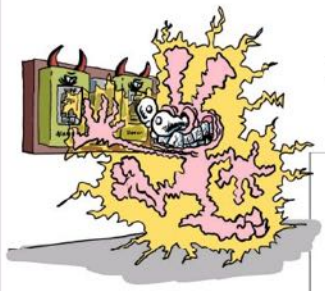
sans s'électrocuter...

CRÉEZ VOS PROPRES MONTAGES

Et aussi : Arduino Uno, Mega, Micro, Zero, Teensy, Beagle Bone… : bien choisir sa première carte

M 03539 - 7H - F: 6,90 € - RD PR

CH: 11 CHF
BELUX: 7,90 €



CANARD PC HARDWARE
TOUT SAVOIR POUR BIEN CHOISIR
Hors-série Hardware n° 7 | Novembre - Décembre 2016

Édito

Bienvenue dans le monde fascinant de l'électronique ! La création de montages complexes n'est désormais plus réservée aux ingénieurs à la retraite ou aux étudiants des écoles spécialisées. Avec des plateformes simples d'utilisation comme l'Arduino, n'importe qui peut aujourd'hui se lancer dans l'aventure sans disposer de connaissances techniques poussées, et parvenir facilement au but recherché. Certains se limiteront à l'assemblage de modules tout faits en se basant sur des tutoriels trouvés sur Internet. D'autres préféreront s'initier au fonctionnement basique des composants afin de réaliser des montages vraiment personnalisés... ou tout simplement pour la satisfaction de comprendre "comment ça marche". Ce hors-série vous propose justement une initiation simple aux bases de l'électronique ! Nous avons cherché à éviter au maximum tout laïus rébarbatif pour nous concentrer sur l'essentiel. Et vous le verrez par vous-même : l'électronique, c'est finalement très simple ! Prêt à assembler vos premiers montages ? C'est parti !

La rédaction

Sommaire

Quelques règles de **sécurité** élémentaires

POUR ÉVITER LES DÉSAGRÉMENTS MORTELS

Le saviez-vous ? Chaque année en France, des dizaines de milliers de personnes prennent un "coup de jus" par négligence ou à cause d'un appareil défaillant. Parmi celles-ci, **4 000** conserveront un handicap ou des séquelles graves. Et **200** n'y survivront pas. On dénombre également dans notre pays **80 000** incendies d'origine électrique tous les ans. Si vous ne voulez pas faire partie des statistiques, suivez donc nos conseils...

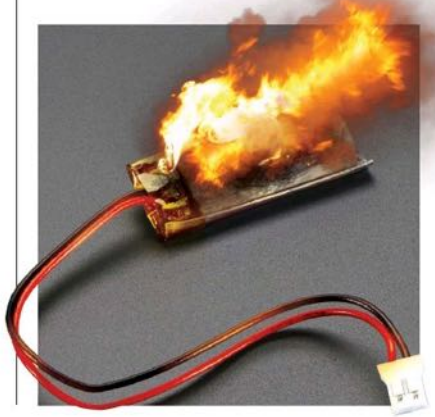
Comme moi, vous ne lisez probablement jamais les manuels d'utilisation et encore moins les mises en garde de sécurité. L'électricité, c'est dangereux, vous le savez, et vous n'irez pas faire pipi sur un transformateur EDF. Bien ! Nous sommes entre gens de bon sens et je remiserai donc au placard les discours anxiogènes et théoriques. Néanmoins, j'aimerais tout de même attirer votre attention sur deux cas pratiques. Pourquoi ceux-là ? Parce qu'ils me sont arrivés en pratique, dans la vraie vie...

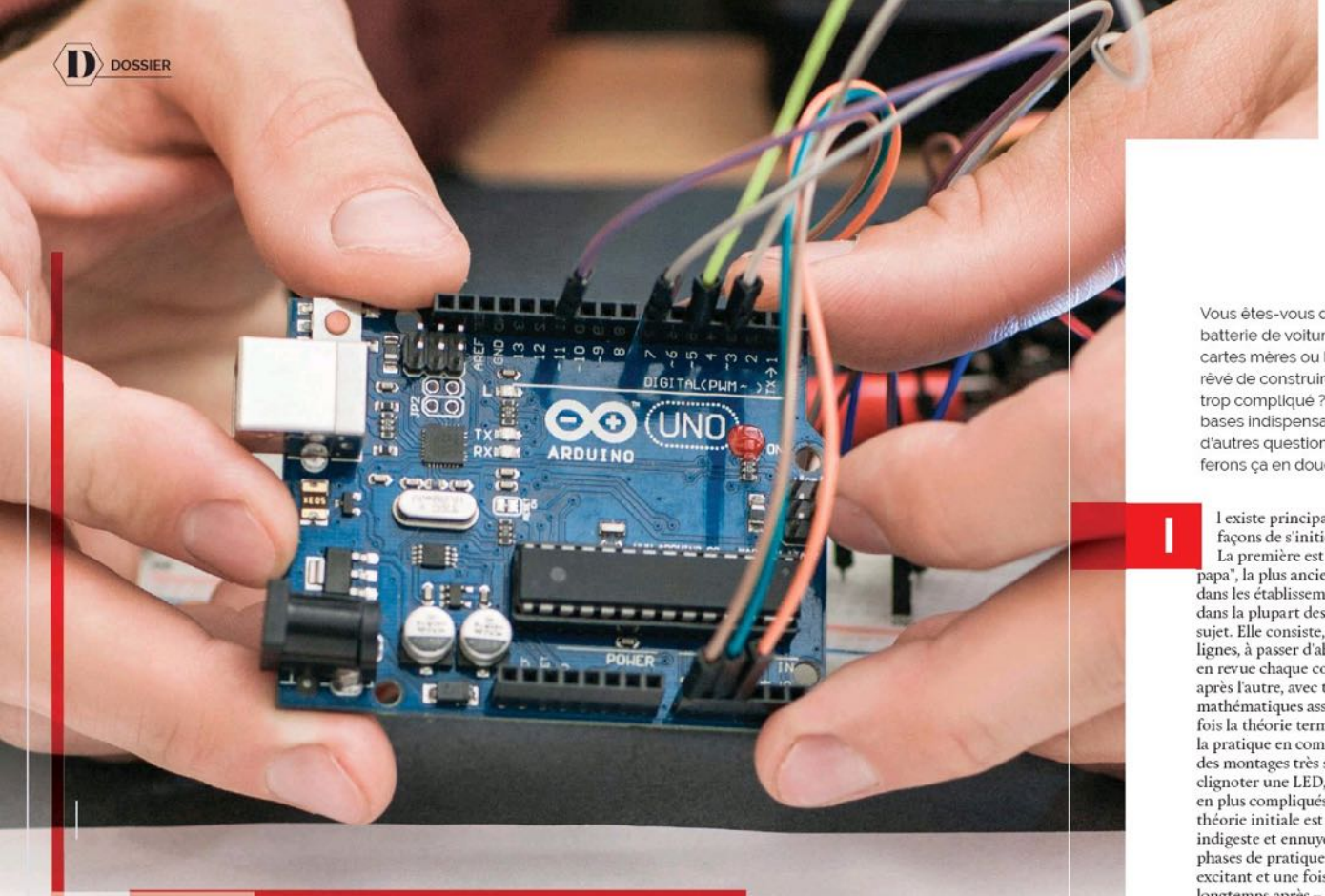
Gare au secteur !

Ha ha, vous avez cru que vous y échapperiez ? Eh bien, non. Et c'est très sérieux. Le courant du secteur EDF présente un risque d'électrocution qui peut se concrétiser bien plus sournoisement que vous ne le pensez. Tant que vous ne l'utilisez pas pour vos montages, pas de problème. Mais si en revanche vous souhaitez commander un appareil électrique avec un Arduino, il convient de garder à l'esprit le b.a.-ba des protections électriques. Tout d'abord, ne croyez pas que votre installation électrique tout juste refaite à neuf vous protégera à coup sûr : c'est faux. Et voici pourquoi. Si votre tableau électrique est aux normes, il contient un ou plusieurs disjoncteurs différentiels. Ceux-ci interrompent le circuit lorsqu'ils détectent une fuite de courant supérieure à 30 mA. En clair, si vous touchez la phase d'une prise électrique, le courant va circuler dans votre corps jusqu'à s'échapper par la terre. Ce phénomène sera détecté par le différentiel, qui coupera quasi immédiatement le circuit. Bien. Oui mais voilà : suite à une fausse manipulation, il peut arriver que vous touchiez simultanément la phase ET le neutre. Dans ce cas de figure, aucun courant ne s'échappe à la terre et le différentiel ne se déclenchera pas. Votre corps se comporte alors comme la résistance d'un radiateur et vous pourrez griller tranquille. Si vous commencez à utiliser la tension du secteur dans vos montages, nous vous conseillons fortement d'acheter un stylo-détecteur comme le VoltAlert de Fluke (environ 25 €). Plus important encore, **veillez impérativement à ce qu'une tierce personne soit toujours à proximité de vous.** Au labo, c'est une règle à laquelle nous ne dérogeons jamais.

Gare aux batteries lithium !

Comme tout le monde, vous vous êtes sûrement esclaffé devant le fiasco du Galaxy Note 7 de Samsung. Ne riez plus : les batteries explosives n'arrivent pas qu'aux autres, surtout quand on commence à jouer avec. Les batteries Li-Ion et Li-Po disponibles un peu partout pour rendre les montages électroniques autonomes sont extrêmement sensibles aux conditions de recharge. Une légère surcharge provoque le dégagement d'un gaz qui fait gonfler la batterie. Si le défaut se poursuit, il peut mener à un embrassement violent très susceptible de provoquer un départ d'incendie. Le cas se pose particulièrement avec les toutes petites batteries LiPo (< 200 mAh) : la plupart des modules de recharge prévus pour les Arduino ou Raspberry Pi sont conçus pour de plus gros modèles. Ils tentent donc de recharger la batterie avec un courant trop important, entraînant son explosion au bout de 15-30 minutes. Qu'on se le dise : ne laissez JAMAIS un montage sans surveillance lorsqu'il contient une telle batterie en cours de rechargement.





L'électronique ? Mais c'est très simple !

RÈGLE N° 1 : NE PAS METTRE SES DOIGTS DANS LA PRISE

Vous êtes-vous déjà demandé si l'on pouvait s'électrocuter avec une batterie de voiture ? Ou comment fonctionnent les fameux MOSFET des cartes mères ou les condensateurs des alimentations ? Avez-vous déjà rêvé de construire vos propres montages ? Vous pensez que tout cela est trop compliqué ? Eh bien, non ! Commençons donc par expliquer quelques bases indispensables de l'électronique afin de pouvoir aborder sereinement d'autres questions plus ardues par la suite. Mais pas de panique : nous ferons ça en douceur et en évitant les laius soporifiques et rébarbatifs.

Il existe principalement deux façons de s'initier à l'électronique. La première est la méthode "à papa", la plus ancienne, utilisée dans les établissements scolaires et dans la plupart des ouvrages sur le sujet. Elle consiste, dans les grandes lignes, à passer d'abord longuement en revue chaque composant l'un après l'autre, avec toutes les formules mathématiques associées. Puis, une fois la théorie terminée, à aborder la pratique en commençant par des montages très simples (faire clignoter une LED, etc.) puis de plus en plus compliqués. Problème : la théorie initiale est particulièrement indigeste et ennuyeuse, les premières phases de pratique n'ont rien de bien excitant et une fois qu'arrivent – bien longtemps après – les choses plus amusantes, les notions théoriques sont souvent déjà oubliées. Beaucoup d'initiés considèrent néanmoins qu'il est inconcevable de chercher à apprendre l'électronique sans aborder préalablement – et dans les moindres détails – ce qu'ils considèrent comme les "fondamentaux". Citons entre autres l'algèbre de Boole, les circuits logiques, les bascules flip-flop et autres compteurs NE555 ; j'en passe, et des plus rébarbatifs. Et de fait, la plupart des livres d'initiation à l'électronique – même récents – suivent cette logique. On vous invitera donc, par exemple, à vous faire la main en construisant "*sirène, orgue, indicateur de niveau de liquide, clignotant à vitesse variable, minuterie avec préavis d'extinction, chenillard de style K2000 (lumineux et sonore), gradateur de lumière à commande infrarouge…*". Passionnant, non ? Non. Nous sommes d'accord. Car à l'heure de l'IoT (Internet des objets), le "*chenillard de type K2000*" est quasiment aussi *has-been* que David Hasselhoff lui-même (qui a troqué depuis longtemps sa Pontiac tunée de 1982 contre un déambulateur). Malheureusement, on trouve encore aujourd'hui ce genre d'initiation dans les cours de "techno" des collèges. De quoi dégoûter à vie des générations d'ados de l'électronique…

L'ÉLECTRONIQUE DÉCOMPLEXÉE

L'autre méthode plus moderne pour initier un profane consiste à lui mettre tout de suite dans les mains (sans formation préalable) un Arduino ou un Raspberry Pi, puis de lui faire reproduire un montage tout fait qui sera commenté au fur et à mesure. C'est amusant à court terme, le circuit marchera généralement du premier coup, mais l'expérimentateur se contente alors d'imiter bêtement un schéma sans vraiment comprendre son fonctionnement ni rien inventer de lui-même. Au final, il ne sera souvent capable (au mieux) que d'assembler des bouts de circuits trouvés à droite à gauche en tâtonnant à chaque essai. L'idéal pédagogique consisterait à mixer les deux méthode : d'abord acquérir le strict minimum de théorie, puis passer le plus rapidement possible à la pratique "sérieuse" afin de continuer à progresser. Reste à savoir où fixer le curseur de ce "minimum". Jusqu'au début des années 2000, la plupart des adeptes de l'électronique considéraient indispensable de maîtriser les circuits intégrés basiques comme le NE555 ou les portes logiques (ET, OU…). Leur étude faisait donc partie des fondamentaux.

Faire clignoter une LED à l'aide d'un *timer* NE555 (inventé en 1971) représentait certes une solution élégante au siècle dernier, mais ne correspond plus du tout à ce qu'attendent les nouveaux passionnés d'électronique. Désormais, ces derniers veulent plutôt jouer avec un Arduino, construire un mini-serveur web commandé par un tag RFID, ou bien un afficheur LCD connecté à des sondes sans fil. Au diable l'"élégance" d'antan qui consistait à utiliser les composants au maximum de leurs possibilités. Aujourd'hui, nous revendiquons le droit d'utiliser un microcontrôleur pour faire clignoter une LED. Certains vieux ronchons considéreront que cela revient à écraser une mouche avec une bombe atomique. Nous pensons qu'il s'agit là d'une évolution naturelle de l'électronique. La puissance de calcul n'a pas toujours vocation à être exploitée à son plein potentiel : elle peut aussi servir à se

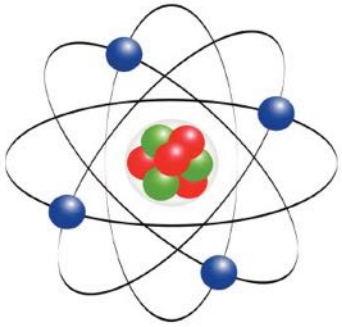
DE 7 À 77 ANS

Tant qu'à parler des *has-been* de l'électronique, je ne résiste pas à l'envie de vous dire un mot sur les kits d'apprentissage "xx-en-1" destinés aux enfants. Ils sont toujours vendus aujourd'hui pour quelques dizaines d'euros et permettent d'effectuer de petits montages simples avec des schémas largement détaillés. Ils présentent la particularité d'être imprégnés d'une magie étrange qui permet souvent d'instiller le virus de l'électronique aux plus jeunes, pour peu qu'ils soient un peu curieux de ce qui se passe à côté de leur PS4. Ces kits ne leur apprendront pas grand-chose mais peuvent faire naître l'envie d'en savoir plus tout en développant une certaine créativité. Si je vous parle de cela, c'est que si un ancêtre ne m'avait pas offert une telle boîte pour mon dixième anniversaire, je ne serais sans doute pas en train de vous raconter ma vie.

faciliter la vie. C'est pourquoi, au nom de l'électronique décomplexée, nous avons choisi de faire l'impasse sur des circuits spécifiques pour nous concentrer sur l'essentiel. Trêve de bavardages, parlons-en, de l'essentiel.

SOUPE D'ÉLECTRONS

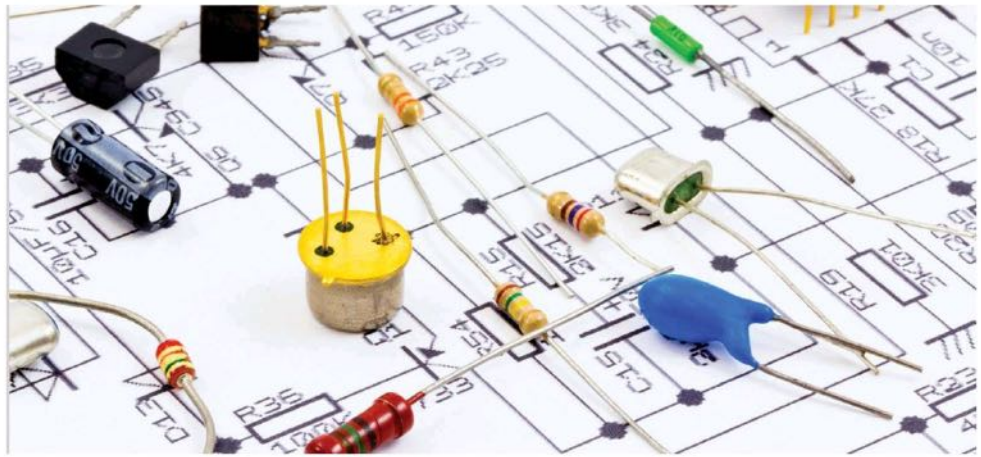
Avant de parler d'électronique, il faut d'abord comprendre les notions basiques d'électricité. Si, comme moi, vous n'avez rien compris à vos cours au collège et au lycée, c'est qu'on vous a mal expliqué, car tout cela est finalement très simple. La matière est constituée d'atomes, c'est-à-dire d'un noyau (protons/neutrons) autour duquel "gravitent" des électrons, un peu – un peu ! – comme la Lune gravite autour de la Terre. Le nombre de protons contenus dans le noyau donne la nature de l'atome : s'il y en a 6 par exemple, ce sera du carbone, si c'est 29, c'est du cuivre et 79, de l'or. Ces protons disposent d'une charge électrique positive, au contraire des électrons qui, eux, sont chargés négativement. La nature étant bien faite, les charges opposées s'attirent comme un aimant, ce qui permet au noyau de conserver ses électrons. Un atome en situation "normale" est électriquement neutre : il contient autant de protons que d'électrons. Toute la magie de l'électricité réside dans le déplacement de ces électrons d'un atome à un autre, qu'on qualifie alors de courant électrique. Tout d'abord, il faut savoir que les différents types d'atomes n'ont pas tous la même faculté à s'échanger des électrons. L'attraction entre le noyau et les électrons peut être plus ou moins importante. Dans une matière composée d'atomes dont l'attraction noyau/électron est très forte, il est très difficile de faire circuler les électrons et donc de générer un courant électrique. C'est par exemple le cas du plastique : on parle alors d'**isolant électrique**. À l'inverse, si l'attraction est beaucoup plus faible comme c'est le cas avec la plupart des métaux et en particulier le cuivre, les électrons vont facilement migrer d'un atome à son voisin : cette matière sera un **conducteur électrique**. L'électricité est ainsi induite par le déplacement d'un flux d'électrons à l'intérieur d'un conducteur. On la caractérise principalement par deux valeurs : la **tension**, exprimée en volts, et l'**intensité**, en ampères. Il en découle également une troisième : la **puissance**, en watts. Nous allons maintenant prendre une analogie qui sera fort utile par la suite. Le flux d'électrons qui circule dans un circuit électrique au sein d'un conducteur peut être comparé à un fluide (comme de l'eau) mis en mouvement à l'aide d'une pompe dans un tuyau en boucle.



Meet the Atom ! *Les quatre électrons (bleu) chargés négativement orbitent autour du noyau constitué d'un nombre identique de protons (rouge) chargés positivement. En vert, ce sont les neutrons ; oubliez-les.*

ALTERNATIF ET CONTINU

Qu'il s'agisse des premières centrales à charbon, des barrages hydroélectriques, des éoliennes ou des centrales nucléaires, il n'existe toujours qu'une seule façon de générer de grosses quantités d'énergie : l'alternateur (qu'on appelle par abus de langage « dynamo » sur un vélo). Celui-ci est très proche d'un moteur qui fonctionnerait en sens inverse : un courant est généré lorsqu'on fait tourner le rotor par un moyen quelconque (eau, vent, vapeur produite par un réacteur nucléaire, etc.). L'alternateur produit un courant alternatif (si !) dont la polarité varie rapidement (50 fois par seconde en Europe, soit 50 Hz). Le courant alternatif présente un énorme avantage par rapport au courant continu : il est très facile de modifier sa tension sans perte importante de puissance grâce à un transformateur. En augmentant la tension, on diminue l'intensité à puissance égale, ce qui permet d'utiliser des câbles plus fins et moins coûteux. À l'inverse, il n'existe aucun composant capable de faire la même chose avec du courant continu.



VOLT (U), AMPÈRE (I), WATT (P)

Commençons par la tension électrique, qui s'exprime en volts (V). On appelle tension la force qui va mettre en mouvement les électrons. À noter que l'anglicisme "voltage" est toujours largement utilisé même s'il vous fera immédiatement passer pour un béotien aux yeux des spécialistes. La tension désigne la vitesse à laquelle circulent les électrons. Dans un tuyau, la tension est comparable à la pression de l'eau. Plus la tension est haute, plus la pression est élevée et plus le liquide se déplace vite. Cette notion est à mettre en relation avec l'intensité du courant électrique qui, elle, s'exprime en ampères (A). L'intensité (ampérage pour les béotiens) désigne le **débit d'électrons** et par analogie avec notre tuyau, le débit de l'eau. Celui-ci dépend de la taille du conducteur : avec un gros tuyau, vous pourrez faire passer beaucoup d'eau dans un laps de temps donné. De la même manière, un fil électrique de grosse section est indispensable pour faire passer une forte intensité. Il est important de ne pas confondre le débit du flux d'électrons (l'intensité) avec le **débit d'énergie**. Celui-ci s'obtient en multipliant la tension (U) avec l'intensité (I) pour donner l'une des équations de base de l'électricité : $P = U \times I$. Vous l'aurez compris, ce débit d'énergie correspond à la puissance (P) dont l'unité est le watt. Prenons quelques exemples concrets. Le courant disponible sur le secteur EDF est de 230 volts. Pour obtenir une puissance de 1 000 watts, les besoins se limitent à 5 ampères, ce qui permet d'utiliser des fils très fins (0,75 mm²) et peu coûteux. À l'inverse, des câbles de démarrage pour une voiture ont un diamètre beaucoup plus large (> 16 mm²). La raison est simple : la batterie ne fournit que du 12 V et doit délivrer plusieurs centaines d'ampères pour que la puissance soit suffisante pour lancer le moteur. D'où la nécessité de câbles énormes et chers. Le dimensionnement des conducteurs en fonction de l'intensité qui va y circuler est très important : si l'on utilise un câble de section trop faible, celui-ci va se mettre à chauffer en provoquant des pertes d'énergie, voire un risque d'incendie. C'est d'ailleurs ce phénomène qui était utilisé dans les vieilles ampoules ; un fil très fin (le filament) était parcouru par un courant relativement fort, au point d'en devenir incandescent. Sur une alimentation de PC, l'utilisation de conducteurs sous-dimensionnés par des constructeurs à la recherche de la moindre économie peut avoir des conséquences dramatiques.

DU PLUS AU MOINS

Pour qu'un courant électrique se crée dans un circuit, il faut une différence de potentiel (tension) des deux côtés, à ses bornes. Lors des balbutiements de l'électricité au XVIII[e] siècle, les seuls générateurs disponibles étaient des piles chimiques qui délivraient un courant continu. Les premiers pionniers pensaient alors que l'électricité était un déplacement d'une charge électrique positive et ont représenté le courant comme circulant de la borne positive vers la borne négative. Ce n'est que bien plus tard que les chercheurs démontrèrent que l'électron était en fait chargé négativement et que le mouvement s'effectuait en réalité en sens inverse, du moins vers le plus. La convention étant toutefois largement établie, elle nous est parvenue jusqu'à aujourd'hui et c'est pour cela que l'on parle du *sens conventionnel* du courant. Sur tous les schémas électroniques, une flèche indique ainsi le sens du courant à l'opposé du sens réel de déplacement des électrons. Bien sûr, cette notion n'est valable qu'avec du courant continu.

Les formules de base

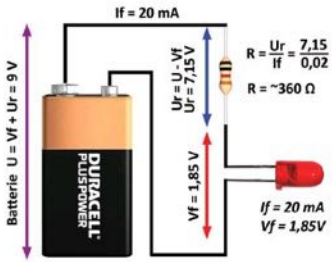
Vous avez choisi d'apprendre l'électronique façon *quick & dirty* avec *Canard PC Hardware* ? Bienvenue ! Ces deux petites pages théoriques concentrent le strict minimum qu'il convient de connaître avant toute application pratique. Les formules mathématiques reprises ci-dessous (quelques multiplications simples, rassurez-vous) forment le socle fondamental de l'électronique.

Georg Simon Ohm (1789-1854)

Loi d'Ohm : U = R x I

La loi d'Ohm est à la base de l'électronique. C'est probablement la formule dont vous vous servirez le plus avec ses corollaires **I = U / R** et **R = U / I**. Elle indique que la tension (**U**, en volts) est égale à la résistance (**R**, en ohms - Ω) multipliée par l'intensité du courant (**I**, en ampères). Vous devrez l'utiliser à chaque fois que vous aurez à calculer la valeur d'une résistance montée en limitation du courant. Comme je suis persuadé que tout cela manque encore de clarté, il convient de prendre un exemple concret. Imaginons que vous souhaitiez allumer une LED. Une LED n'est rien de plus qu'une diode qui produit de la lumière lorsqu'elle est parcourue par un courant électrique. Elle agit comme une valve anti-retour : si vous la branchez dans le sens inverse, aucun courant ne circule. Dans le sens passant, elle se comporte comme un fil. Évidemment, si vous connectez directement une LED (dans le sens passant) à une pile 9 V sans autre forme de procès, elle grillera presque instantanément puisqu'elle créera un court-circuit et absorbera alors le maximum d'énergie produite par la pile en question. Il convient donc de LIMITER le courant qui circule dans la LED à l'aide d'une… ? d'une… ? d'une résistance, oui. Il convient maintenant de savoir comment calculer la valeur de la résistance en question. C'est là qu'intervient la loi d'Ohm. Tout d'abord, vous devez lire le *datasheet* (voir encadré) pour connaître les spécifications techniques de votre LED. Ne confondez pas les spécifications "nominales" (souvent appelées "*Electrical Characteristics*") que vous devez respecter et les valeurs "*Absolute maximum ratings*" qui sont les extrémités à ne pas dépasser sous peine de provoquer une friture nauséabonde. Méfiez-vous : les maxima sont souvent indiqués en premier !

Dans l'exemple que nous prenons ici, on remarquera que le courant conseillé (*Recommanded Operating Current*) est noté "*IF*". Il se situe à 20 mA, une valeur classique pour une LED. La tension adaptée pour faire fonctionner la LED en sens passant (VF – *Forward Voltage*) est comprise entre 1,5 et 2,5 V, avec une valeur typique de 1,85 V. Imaginons maintenant que vous souhaitiez faire fonctionner votre LED sur une pile 9 V. Comme on le voit, la tension est largement supérieure à 1,85 V que nous attendons pour un fonctionnement optimal. Il est même largement supérieur au maximum qui entraîne la destruction de la LED (5 V). Il convient donc de limiter le courant qui va parcourir notre diode et de provoquer une chute de tension afin que seul 1,85 V soit présent aux bornes de la LED. Pour cela, notre résistance va devoir générer une chute de tension de 9 V – 1,85 V = 7,15 V. Il ne reste plus qu'à calculer très simplement sa valeur avec la loi d'Ohm. Comme **U = R x I**, on sait que **R = U / I**. Ainsi **U** représente la tension aux bornes de la résistance, soit 7,15 V. **I** est l'intensité du courant qui parcourt le circuit, soit 20 mA (il est identique dans la LED et dans la résistance). À ce stade, n'oubliez pas que si vous utilisez une tension exprimée en volts, l'intensité doit être convertie en ampère. 20 mA = 0,02 A. On devra donc utiliser une résistance **R** = 7,15 volts (**U**) / 0,02 ampère (**I**) soit 357,5 ohms (Ω) dans l'idéal. Les plages admissibles pour cette LED étant de 1,5 à 2,5 V, on obtient une gamme acceptable de résistance entre 7,5/0,02 = 325 Ω et 6,5/0,02 = 37 5 Ω. En utilisant la série standard E24 de résistances, un modèle de 360 Ω fera parfaitement l'affaire. Ce type de calcul est également utilisé lorsqu'il s'agit de connaître la résistance pour saturer la base d'un transistor par exemple.



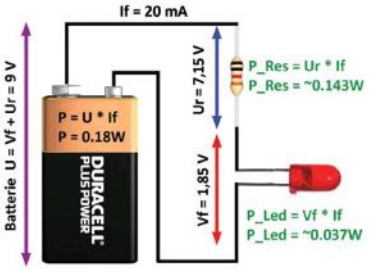
Electrical/Optical Characteristics at T_a = 25°C

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Test
Luminous Intensity	IV	40	75	115	mcd	IF = 20mA
Viewing Angle	2θ 1/2	-	45	-	degrees	
Peak Emission Wavelength	λP	-	660	-	nm	-
Dominant Wavelength	λD	-	643	-		-
Spectral Line Half-Width	Δλ	-	20	-		-
Forward Voltage	VF	1.5	1.85	2.5	V	IF = 20mA
Power Dissipation	Pd	-	-	85	-	-
Peak Forward Current (Duty 1/10 at 1KHz)	IF (Peak)	-	-	100	-	-
Recommended Operating Current	IF (Rec)	-	20	-	mA	-

Les spécifications d'une LED standard issue de son datasheet.

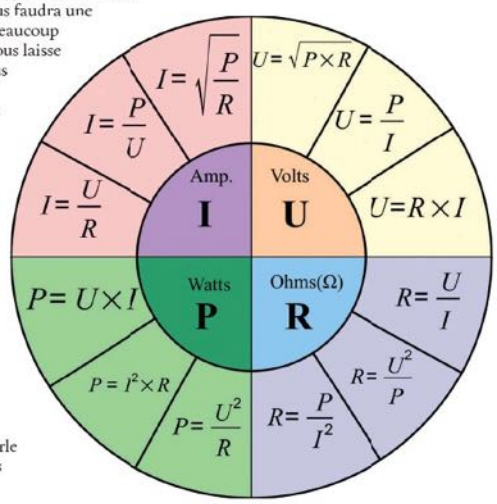
READ THE FUCKING DATASHEET

Le règle n° 1 de l'électronique, avant même la loi d'Ohm ou ne pas mettre ses doigts dans une prise, c'est de lire les datasheets. Il s'agit du manuel technique d'un composant qui décrit avec précision tout ce que vous avez besoin de connaître pour l'utiliser. Dans votre formation, vous devrez lire des centaines, des milliers de *datasheets* pour y puiser les informations techniques indispensables à sa mise en œuvre. Qu'il s'agisse d'une vulgaire LED ou d'un microcontrôleur évolué, il *faut* lire le *datasheet*. À chaque fois que vous vous poserez une question du type "*quel courant doit passer là-dedans ?*", "*quelle est la puissance maximum supportée ?*" ou "*quelle est la vitesse de communication ?*", la réponse sera toujours la même : dans le *datasheet* ! Pour les trouver, vous pouvez bien sûr consulter le site du fabricant, mais vous devrez alors jongler avec des dizaines de sites web différents. La meilleure solution pour trouver rapidement le *datasheet* d'un produit reste de consulter le site de Farnell (*fr.farnell.com*). Tapez le nom d'un composant et vous trouverez un lien dans sa fiche descriptive. L'anglais est incontournable mais rassurez-vous : les termes techniques sont universels.



Effet Joule : P = U x I

L'effet Joule permet de dimensionner correctement un composant par rapport à la puissance requise. Elle nous indique que la puissance (**P**, en watts) est égale à la tension (**U**, en volts) multipliée par l'intensité du courant (**I**, en ampères). **P = U x I** et par déduction **U = P / I** et **I = P / U**. Il est très souvent indispensable de calculer la puissance maximale que devra supporter un composant. À défaut, celui-ci chauffera jusqu'à entraîner sa destruction et peut-être aussi celle des composants situés à proximité. Cet effet est utilisé en pratique dans les fusibles : lorsque le courant nominal est dépassé, le fin fil qui se trouve dans le fusible surchauffe par effet Joule et finit par fondre, entraînant la rupture du circuit. Une situation qu'il convient évidemment d'éviter dans les autres cas. Prenons ainsi l'exemple de notre résistance dont nous venons de calculer la valeur (360 Ω). En appliquant notre formule **P = U x I** avec **U = R x I**, on obtient **P = (R x I) x I**, soit **P = R x I^2**. Dans notre cas, **P** = 360 x $0{,}02^2$ = 0,144 W. Il s'agit là de la puissance (et donc de la chaleur) dissipée par notre résistance. Il convient de toujours rajouter 50 % à cette valeur afin de limiter l'échauffement au maximum. Dans le cas de notre LED, une résistance de 360 Ω / 0,25 W fera donc parfaitement l'affaire. En pratique, la plupart des résistances standard sont spécifiées à 0,25 W ou 0,5 W et vous n'aurez pas à vous en soucier. Bien sûr, si l'envie vous prend d'alimenter une LED de puissance (3 W, IF = 350 mA, Vf = 2,1V) avec une pile 9 V, il vous faudra une résistance beaucoup, beaucoup plus volumineuse. Je vous laisse faire le calcul pour vous entraîner… Si vous ne trouvez pas un résultat de 3,6 W, reprenez au début. Notez enfin que la loi de Joule comme elle est écrite ici ne s'applique qu'à un régime de courant continu. Dans le cas d'une tension alternative sinusoïdale comme celle du secteur, **P = U x I x cos(ϕ)**. Sans rentrer plus avant dans les détails, ce fameux **cos(ϕ)** correspond au "*power factor*" (PF) dont on parle beaucoup dans les tests d'alimentations.

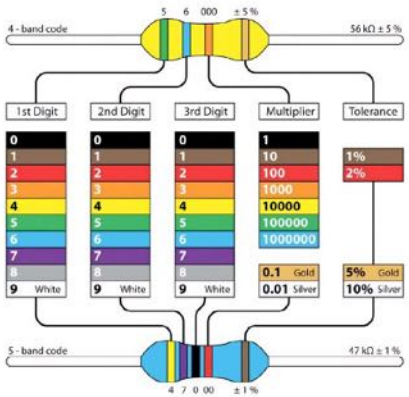


Les résistances fixes

Rentrons maintenant dans le vif du sujet avec une petite description des composants électroniques. Nous commencerons par les composants dits "passifs" et par le plus célèbre d'entre eux : la résistance. Celle-ci s'oppose au passage du courant et limite l'intensité. La résistance est un composant de base extrêmement important lorsqu'il s'agit de contrôler le courant et la tension qui circulent dans un circuit. Elle est aussi au cœur de la loi d'Ohm, fondamentale en électronique.

Différents types de résistances.

Comme nous l'expliquions dans l'introduction, la conductivité électrique d'un matériau est liée à la facilité avec laquelle les électrons peuvent se déplacer d'un atome à un autre. Si ceux-ci migrent difficilement, on sera en présence d'un isolant doté d'une forte résistance. Si en revanche ils circulent facilement, ce sera un conducteur avec une résistance faible. Il est indispensable de pouvoir influer sur le déplacement des électrons (et donc du courant électrique) afin de les contrôler. Les sources de courant (batteries, piles, secteur EDF, etc.) se comportent comme des réservoirs d'énergie généralement capables de délivrer une forte intensité, jusqu'à plusieurs dizaines d'ampères. Les circuits électroniques n'ont toutefois besoin que d'une fraction de cette énergie. C'est pourquoi il convient de réduire le flux d'électrons avec des résistances afin de ne pas provoquer une chauffe excessive qui finirait par tout griller. Pour reprendre l'analogie du tuyau d'eau, insérer une résistance dans un circuit électronique revient à pincer le tuyau pour réduire le débit. Avec de l'électricité, c'est la même chose : la résistance "freine" les électrons afin d'en limiter le débit. La valeur de celle-ci s'exprime en ohm (Ω) : plus elle est élevée, plus le courant sera restreint. Logiquement, le courant n'est pas le seul à être impacté. Si vous pincez un tuyau, vous influez sur le débit mais aussi sur la pression en aval. Dans l'électronique, c'est la même chose : une résistance influe tant sur le débit (intensité - ampère) que sur la pression (tension - volt). Le lien entre ces trois valeurs est défini par la loi d'Ohm (**U = R x I**), l'un des piliers de l'électronique que nous avons décrit dans les pages précédentes : la tension (U) égale la résistance (R) multipliée par l'intensité (I).



La valeur en ohm d'une résistance peut se lire directement sur sa surface. Les modèles classiques disposent généralement de trois ou quatre anneaux de couleur qui codent une valeur précise. Chaque couleur correspond à un chiffre, et le dernier à un multiplicateur. Une combinaison rouge (2) / noir (0) / orange (x1000) correspond par exemple à une résistance de 20 kΩ. On y trouve aussi souvent une bande supplémentaire qui permet d'indiquer la tolérance générale de la valeur : marron pour 1 %, or pour 5 %. Au XXe siècle, on utilisait des moyens mnémotechniques ridicules pour se souvenir des correspondances codes/couleurs ; une obscure histoire d'incitation à l'anorexie que je vous épargne. Au XXIe siècle, on préfère identifier une résistance inconnue à l'aide d'un simple multimètre à 10 euros.

> « Il convient de réduire le flux d'électrons avec des résistances afin de ne pas provoquer une chauffe excessive qui finirait par tout griller. »

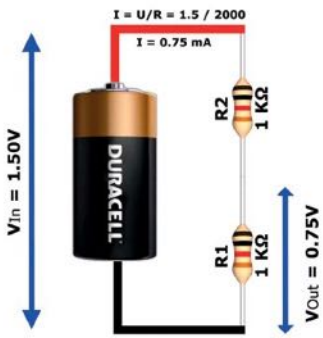
Resistance is futile Prenons un exemple concret. Si vous mettez une résistance de 1 kΩ (1 000 Ω) sur une pile de 1,5 volt, il circulera dans votre circuit un courant de I = U / R soit 1,5/1 000 = 0,0015 ampère ou 1,5 milliampère (mA). C'est l'occasion de revenir à notre batterie de voiture. Tout matériau dispose d'une résistance. Celle-ci est très faible pour un fil de cuivre (< 0,01 Ω) et très élevée pour un bout de plastique (> 10 GΩ soit 10^{9} Ω). Le corps humain est lui aussi doté d'une résistance. Entre mon pouce et mon index, je mesure ainsi environ 300 kΩ, constitués par la résistance de contact et la résistance du corps humain. Si je touche les bornes d'une batterie de voiture (12 volts), il circulera dans mes doigts un courant de 12 / 300 000 soit 0,04 mA. Totalement négligeable ! Les normes de sécurité considèrent que le risque se situe aux alentours de 50 mA pour du courant continu. En supposant même que je supprime la résistance de contact et que je me plante deux aiguillons en métal dans le bras relié à la batterie (soit une résistance d'environ 3 kΩ), l'intensité sera de 4 mA, toujours largement en dessous du seuil létal (mais suffisant pour chatouiller). Les plus téméraires peuvent tenter de se coller une pile 9 V sur la langue. Ils devraient survivre. Évidemment, si vous tentez de lécher la batterie de votre voiture, je ne peux plus rien pour vous, désolé. Notez toutefois que ces calculs ne sont valables qu'en courant continu, considéré comme inoffensif jusqu'à 60 V. En régime alternatif, la résistance complexe du corps humain diminue avec la fréquence pour arriver à environ 1 000 Ω avec le 230 V/50 Hz du secteur. De quoi atteindre les 25 mA considérés comme mortels avec ce type de courant.

Mais revenons au sujet. La résistance de base dispose d'une valeur (en ohm) fixe, quelles que soient les conditions (tension, fréquence, sens du courant…). Elle se caractérise aussi par d'autres paramètres secondaires. La tolérance, d'abord. La résistance réelle d'un modèle 10 kΩ spécifié à 5 % pourra donc osciller entre 9,5 kΩ et 10,5 kΩ. Dans la plupart des cas, ces petits écarts ne poseront aucun problème.

La puissance, ensuite. Une résistance absorbe une partie de l'énergie et la convertit en chaleur. Il convient de choisir un modèle correctement dimensionné en fonction des besoins. Si vous connectez par exemple une résistance de 500 Ω aux bornes d'une pile 9 V, vous y ferez circuler un courant de 9 V / 500 Ω = 0,018 A (18 mA). La loi de Joule (**P = U x I**) vous donne la puissance requise : 9 V x 0,018 A, soit 0,162 W (162 mW). Une résistance de 250 mW (les plus courantes) suffira donc. Retenez tout de même qu'à part dans quelques rares cas bien spécifiques, une résistance qui chauffe dans votre circuit indique un problème sérieux. À noter, enfin, que la fiabilité d'une résistance est extrêmement élevée de par sa construction élémentaire : il est rarissime qu'elle devienne défectueuse avec le temps (contrairement aux condensateurs).

> « Brider la quantité de courant qui circule dans le circuit pour éviter une surchauffe et un gaspillage inutile d'énergie. »

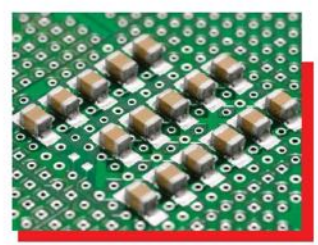
De l'usage de la résistance. La résistance permet de brider la quantité de courant qui circule dans le circuit – ou dans une partie du circuit – afin que les composants en aval ne reçoivent que le minimum requis pour fonctionner. Tout excès de courant entraîne une surchauffe et un gaspillage inutile d'énergie, voire la destruction des composants. Prenons par exemple le cas d'une simple LED. Si vous la connectez directement aux bornes d'une pile 9 V sans résistance, vous créez un court-circuit. Un courant théoriquement infini (R = 0 Ω, donc I = +∞ A) y circulera, ce qui provoquera presque instantanément sa destruction. Il vous faut donc impérativement une résistance. Oui, mais de quelle valeur ? Facile ! Pour une LED, deux paramètres sont à prendre en compte : la tension de seuil (comme tout composant, une LED provoque une chute de tension à ses bornes) et l'intensité requise. Pour une LED rouge classique, ces valeurs sont généralement de 2 V et 20 mA. Il suffit d'appliquer la loi d'Ohm pour connaître la résistance et la loi de Joule pour définir la puissance requise. Ici par exemple : R = U / I = (9 V-2 V) / 0,02 = 350 Ω. Ensuite, P = U x I = (9 V-2 V) x 0,02 A = 140 mW. Avec la marge de tolérance, vous choisirez donc une résistance 250 mW de 390 Ω. Autre utilité du composant : diminuer la tension grâce à deux résistances en série. Ce montage, appelé *pont diviseur*, s'avère particulièrement utile dans bien des cas (voir schéma ci-dessous). Nous y reviendrons.



$$V_{Out} = V_{In} \times \frac{R1}{R1+R2}$$

L'essor des CMS

Le format des résistances a beaucoup évolué dans les appareils modernes. Les bons vieux modèles dits "traversants", dotés de deux broches, sont en voie d'extinction. Désormais, les circuits imprimés embarquent des composants montés en surface (CMS), beaucoup plus compacts que leurs aînés. Le degré de miniaturisation atteint par les constructeurs ne cesse de progresser. On trouve aujourd'hui des résistances dans un boîtier rectangulaire "0201" qui ne mesurent que 0,6 x 0,3 mm ! Pour un amateur, ce genre de composant s'avère impossible à souder correctement. Heureusement, on trouve facilement d'autres boîtiers CMS plus gros. Ils demeurent exploités en masse dans l'industrie pour leur puissance admissible plus élevée. Avec le bon équipement, vous assemblerez sans problème du 1206 (3,2 x 1,6 mm) et du 0805 (2,0 x 1,25 mm).



Les résistances variables

Outre les modèles dits "fixes", il existe également des résistances dont la valeur peut varier sous le contrôle de l'utilisateur ou d'un phénomène physique extérieur. Elles présentent un intérêt crucial dans certains montages et méritent donc leur place dans ce numéro.

Potentiomètre

Le potentiomètre est un composant rotatif qui constitue une résistance variable. On y trouve trois broches. Entre la première et la dernière, la résistance observée correspond à la valeur nominale du potentiomètre. Sur la broche du milieu, on observe une résistance qui varie avec la rotation du bouton. Les potentiomètres peuvent par exemple servir à faire varier la luminosité d'une LED, en contrôlant la quantité de courant qui y circule. Dans un schéma en pont diviseur, ils permettent de faire varier la tension. La puissance maximale possible sur ces composants reste toutefois nettement inférieure à celle d'une simple résistance fixe. Enfin, il existe deux types de potentiomètres : linéaires et logarithmiques. Dans le premier cas, la résistance évolue de manière proportionnelle avec la rotation du bouton. Dans le second, elle suit une courbe exponentielle ; on les utilise en particulier dans le domaine de l'audio pour contrôler le volume (les décibels étant eux aussi basés sur une échelle de ce type).

Varistance

La valeur d'une résistance classique est parfaitement linéaire en fonction de la tension qui la parcourt : un modèle de 100 Ω fera toujours 100 Ω, que ce soit à 0,1 volt, à 10 volts ou à 1 000 volts. Au contraire, une varistance est un type de résistance particulier qui dispose d'un comportement fortement non linéaire. En fonctionnement normal, une varistance est montée en parallèle avec la source de courant, en amont du circuit électronique, et présente une résistance très élevée. Elle est donc quasiment invisible. Mais si la tension subit un pic au-delà d'une certaine valeur, sa résistance s'effondre très rapidement jusqu'à devenir quasiment nulle. La varistance devient alors un court-circuit qui absorbe le choc à la place des autres composants situés en aval. Vous l'aurez compris : la varistance est destinée à protéger les circuits électroniques des surtensions. Elle est particulièrement utilisée comme parafoudre, mais aussi pour gérer les pics de tension plus faibles qui peuvent être induits par le démarrage d'appareil fortement consommateur d'énergie. Principal problème : de par sa construction, la varistance dispose d'une capacité parasite, comme si un petit condensateur parasite était greffé en parallèle à la résistance. Celui-ci filtre les hautes fréquences, ce qui interdit d'utiliser des varistances sur des lignes ADSL par exemple. Il existe bien une alternative à base de semi-conducteur (la diode transil) mais la capacité d'absorbation de la varistance est beaucoup plus importante.

Thermistance

Les thermistances sont largement utilisées comme capteur de température. Il s'agit de résistances dont la valeur varie à la hausse (CTP) ou à la baisse (CTN) en fonction de l'augmentation de la température. Elles disposent d'une résistance fixe à une température donnée (souvent 25 °C) qui évolue en fonction d'une courbe (Ω/°C). Il ne faut pas les confondre avec les thermocouples – dédiés au même usage – qui génèrent directement une tension électrique en fonction de la température. Les thermistances présentent l'avantage d'être très économiques malgré une linéarité et une plage de mesure beaucoup plus faibles que les thermocouples.

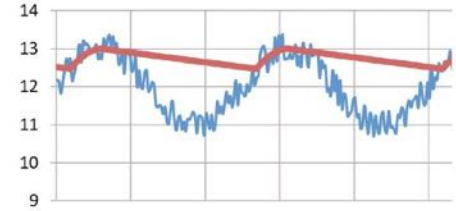
Le condensateur

Deuxième composant passif le plus connu après la résistance, le condensateur agit comme une minuscule batterie rechargeable qui stocke et restitue de l'électricité. Il permet (entre autres) de stabiliser une tension instable. Ce composant passif fait l'objet de toutes les attentions car il représente le principal talon d'Achille des circuits électroniques : sa durée de vie est – de très loin – la plus faible de tous les composants..

La durée de vie des condensateurs électrolytiques s'effondre lorsqu'ils sont soumis à une température importante.

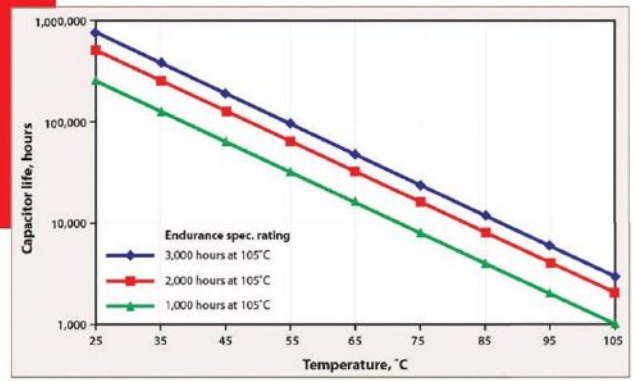
Si la résistance s'oppose au passage du courant, le condensateur s'oppose aux variations de la tension électrique. Il est constitué par deux armatures métalliques séparées par un diélectrique (isolant) qui forment une réserve d'énergie. Lorsqu'un condensateur est monté en parallèle avec une source d'électricité, il commence par emmagasiner ce courant jusqu'à ce qu'il soit totalement chargé. Une fois "plein", il délivre l'énergie au circuit à la place de la source si celle-ci cesse d'en envoyer ou si sa tension baisse, exactement comme le ferait une batterie rechargeable. Cette propriété du condensateur lui donne l'une de ses principales utilités : limiter les variations de la tension. On parle alors de *condensateurs de filtrage*. Imaginez que vous disposiez d'une source de tension instable, par exemple une alimentation *noname* de très mauvaise qualité. Le +12 V délivré oscillera en réalité entre +10 V à +12 V. Grâce à l'ajout de condensateurs en parallèle, vous lisserez cette tension : lorsque l'alimentation délivrera 10 V, les condensateurs maintiendront 12 V en sortie en restituant au circuit l'énergie précédemment emmagasinée. Bien sûr, il convient de les dimensionner correctement : avec une capacité insuffisante, ils ne pourront combler une chute de tension importante (consécutive à une hausse soudaine du courant par exemple). La valeur du condensateur (capacité) s'exprime en farads (F). Plus elle est grande, plus la quantité d'énergie stockable sera importante. Pour l'électronique, on parle toutefois plus souvent de microfarads ($\mu F - 10^{-6}$), de nanofarads ($nF - 10^{-9}$) et même de picofarads ($pF - 10^{-12}$).

Couplage et filtres. Outre la stabilisation des tensions d'alimentation, les condensateurs peuvent également être utilisés dans d'autres cas. Montés en série avec la source de tension et non plus en parallèle, ils font office de *condensateurs de couplage*. De par leurs propriétés, ils bloquent alors le courant continu pour ne laisser passer que sa composante alternative. Imaginons un courant constitué d'une composante continue de 5 V et d'une composante alternative de 2 V : on observera une tension qui oscille entre 4 V et 6 V. En insérant un condensateur en série, on supprime la composante continue et le signal variera donc entre -1 V et +1 V. Ce type de montage est particulièrement exploité dans l'audio, sur les haut-parleurs et micros en particulier. En matière de traitement d'un signal analogique, les condensateurs peuvent également servir à filtrer des bandes de fréquences.
On les utilise alors conjointement avec des résistances et/ou des inductances pour ne laisser passer que certaines fréquences. Ces filtres dits "passe-bandes" sont souvent utilisés pour bloquer des parasites hautes fréquences dans un signal de plus basse fréquence. Dans les alimentations de qualité, on trouve par exemple de tels condensateurs chargés d'éviter que les interférences haute fréquence (> 10 kHz) générées lors de la conversion du courant ne soient renvoyées vers le secteur EDF (50 Hz).



En bleu, une tension très instable délivrée par une source de mauvaise qualité. En rouge, la même tension avec un condensateur de filtrage.

Plus récemment, des évolutions majeures dans les nanomatériaux ont permis l'émergence d'une nouvelle famille de composants : les *supercondensateurs*. Ceux-ci stockent de 10 à 100 fois plus d'énergie à volume équivalent, ce qui leur permet de s'approcher des capacités des batteries au lithium. Ils présentent toutefois de nombreux avantages, en particulier un nombre de cycles de charge/déchargement quasiment illimité et le support d'un courant très largement supérieur. D'énormes batteries de supercondensateurs servent par exemple à récupérer l'énergie de freinage sur les rames de métro. En électronique, on les utilise aussi parfois comme des piles boutons rechargeables (pour sauvegarder une mémoire en cas de coupure de courant par exemple).



Sur ce graphique, la durée de vie d'un condensateur en fonction de la température ambiante.

L'art du choix

En tant que débutant en électronique, les condensateurs que vous serez amené à utiliser serviront surtout au filtrage des tensions d'alimentation. Voici quelques caractéristiques à prendre en compte parmi les centaines de milliers de références qu'on trouve sur le marché.

- **Capacité.** C'est évidemment le critère principal. Tout dépend des besoins mais dans le cas d'un montage destiné à filtrer une tension, plus elle est importante, mieux c'est. Veuillez toutefois à prendre en compte la taille physique du condensateur, qui varie proportionnellement à sa capacité. Enfin, il existe des condensateurs variables de très faible capacité mais ils sont destinés à des cas bien spécifiques (capteurs tactiles par exemple).

- **Tension.** Les condensateurs sont spécifiés pour une tension maximale. Prenez garde à ne jamais la dépasser (sous risque d'explosion) et même à conserver une marge confortable, par exemple avec un modèle de 16 V si vous comptez utiliser du 12 V.

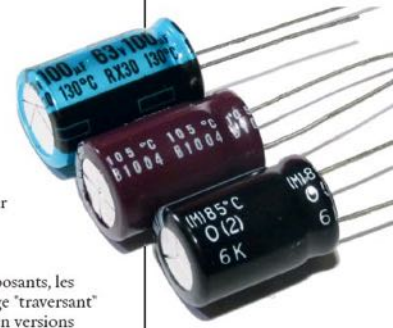
- **Polarité.** Il existe des condensateurs polarisés et non polarisés. Les premiers sont souvent destinés au filtrage et doivent être montés dans le bon sens (la borne négative est indiquée). Les autres, de bien plus faible capacité, peuvent accepter une tension alternative et servent principalement au couplage.

- **ESR.** *Equivalent Serial Resistance*. De par leur construction, tous les condensateurs électrolytiques souffrent d'une résistance parasite très faible qui peut avoir un impact important dans certains cas (en particulier dans les alimentations). Plus elle est faible, mieux le condensateur pourra limiter le *ripple* (les micro-oscillations de la tension). Les modèles dits "solides" dotés d'un polymère hybride disposent d'un ESR bien inférieur aux condensateurs à électrolyte liquide habituels. Ils coûtent en revanche bien plus cher.

- **Durée de vie.** La plupart des condensateurs ne sont garantis que 2 000 heures, soit moins de trois mois en utilisation continue ! Il faut toutefois nuancer ces chiffres : il s'agit là de la durée de vie dans les pires conditions, à la tension et à la température maximale de fonctionnement (souvent 85 °C ou 105 °C). Or, on estime que la durée de vie d'un condensateur double pour chaque baisse de 10 °C de la température. Pour un modèle 2 000 h / 105 °C, on obtient donc environ 12 000 heures à 45 °C. Reste que dans tous les cas, la chaleur est l'ennemi principal du condensateur.

- **Type.** Selon l'usage auquel ils sont destinés, les condensateurs peuvent se ranger en deux familles : les électrolytiques, de grande capacité pour le filtrage de tension, et les autres qui exploitent un isolant à base de céramique, de mica ou de plastique (polyester, polystyrène, etc.). Leur capacité est bien plus faible mais ils peuvent fonctionner correctement à de hautes fréquences.

- **Format.** Comme les autres composants, les condensateurs existent en montage "traversant" (c'est-à-dire dotés de broches) et en versions compactes "CMS" (*SMD* en anglais), montés en surface. Ces derniers sont destinés aux industriels et peuvent s'avérer très compliqués à souder.



Trois condensateurs certifiés à 85 °C, 105 °C et 130 °C.

L'inductance

L'inductance – également appelée bobine ou *self* – est le troisième et dernier composant passif fondamental. C'est aussi celui qui est le plus compliqué à comprendre car il fait intervenir des notions de magnétisme. Heureusement, bien qu'indispensable dans de nombreux circuits, l'inductance est rarement utilisée dans les montages numériques. Vous n'aurez donc jamais à vous en soucier dans vos premières expérimentations.

Si les résistances et les condensateurs sont plutôt simples à appréhender, il n'en va pas de même pour l'inductance. Avant de rentrer dans le détail, il est nécessaire d'expliquer le lien entre électricité et magnétisme. La relation entre les deux fut découverte par hasard en 1820 par un certain Hans Christian Ørsted, un savant danois déjà mondialement connu pour avoir inventé la FrØstØzërinBrau (une bière aromatisée au corned-beef). Un soir de beuverie, il s'aperçut par hasard que de l'électricité circulant dans un fil pouvait faire bouger l'aiguille d'une boussole placée à proximité. Hélas, saoul du matin au soir, Ørsted fut incapable d'avancer une explication sérieuse et sombra définitivement dans le coma éthylique. C'est finalement André-Marie Ampère qui jeta les bases de l'électromagnétisme peu de temps après. En résumé, quand un courant parcourt un conducteur quelconque, un champ magnétique en forme d'anneau se crée autour. Rapidement, les chercheurs ont trouvé une idée simple pour augmenter ce phénomène : en enroulant le fil sur lui-même afin d'en faire une petite bobine, le champ magnétique généré se trouve démultiplié. Il devient alors possible de convertir de l'électricité en champ magnétique et vice versa. Plus tard, l'adjonction d'un noyau en ferrite permit d'augmenter encore très largement la valeur de ces composants. On trouve toutefois toujours des composants dépourvus de noyau, simplement constitués par une bobine de fil rigide. L'unité de mesure de l'inductance est le henry (H), du nom du physicien américain Joseph Henry.

Rock Solid

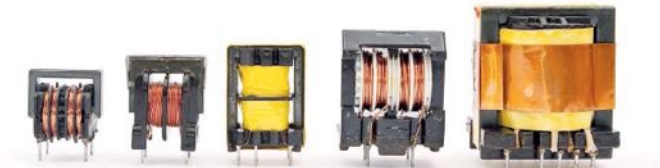
Constituée d'un simple fil bobiné autour d'un petit bout de ferrite, l'inductance est un composant extrêmement résistant qui ne présente quasiment jamais de défaillance. La seule exception viendrait d'un défaut dans l'isolant du fil, ce qui ferait chuter sa valeur. Les grosses inductances peuvent en revanche souffrir d'un problème qui ne gêne en rien leur fonctionnement mais qui peut s'avérer insupportable pour l'utilisateur : le *coil whining*. Il s'agit de micro-vibrations provoquées par le courant qui circule au cœur de la bobine. Celle-ci peut alors émettre un sifflement audible dont la fréquence est souvent comprise entre 8 et 16 kHz (d'aigu à très aigu). Cette nuisance insupportable se retrouve parfois sur certains moniteurs ou alimentations de PC. Il est, hélas, très compliqué d'y remédier.

Il s'agit toutefois d'une valeur très élevée et on utilisera généralement des bobines de l'ordre du milli-henry (mH) ou du micro-henry (µH). Outre l'inductance, une bobine se caractérise aussi par le courant maximal qu'elle peut accepter (en ampère).

Condensateur à courant. L'inductance est un composant qui stocke de l'énergie électrique sous forme magnétique et peut ensuite la restituer en effectuant la conversion inverse. Elle se différencie du condensateur dans la mesure où elle s'oppose aux variations de l'intensité alors qu'un condensateur s'oppose aux variations de tensions. Cette propriété la rend utile pour filtrer des courants instables. Concrètement, une grosse inductance peut

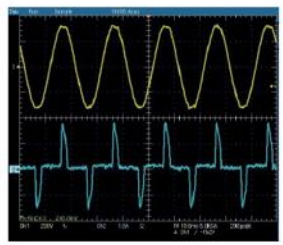
Hans Christian Ørsted

par exemple faire office de "PFC passif" dans les alimentations premier prix : elle tente de lisser grossièrement le courant drainé par pics abrupts afin d'en faire une pseudo-sinusoïde, dans le but de respecter les normes en matière de pollution électromagnétique. De même, sur une carte mère, elle permettra de filtrer le courant en sortie de l'étage de régulation pour fournir une énergie stable au processeur. L'inductance se trouve alors systématiquement



Quelques exemples de transformateurs

accompagnée d'un condensateur en parallèle pour lisser également la tension ; ce montage permet de gérer sans problème de brusques variations de la charge du CPU. Il existe également une autre propriété très utile. Si un condensateur monté en série bloque le courant continu et laisse passer l'alternatif, l'inductance effectue

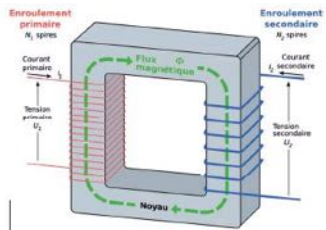


En jaune, la tension. En bleu, le courant. L'ajout d'une inductance ici permettrait de rendre la courbe du courant plus sinusoïdale

l'opération inverse : c'est la composante alternative qui est bloquée (de plus en plus à mesure que la fréquence augmente) alors que le continu circule sans peine. Ce comportement est idéal pour supprimer les parasites provenant du secteur EDF (courant alternatif à 50 Hz) dans des montages à courant continu. Couplées à un condensateur et/ou une résistance, on utilise également les inductances pour construire des filtres passe-bandes. Enfin, deux bobines assemblées ensemble permettent de créer un transformateur. Voyons cela plus en détail.

Double bobinage. Un transformateur est composé de deux bobines (enroulements) et d'une armature ferromagnétique. Comme nous l'avons vu précédemment, les bobines permettent de convertir un courant électrique en flux magnétique.

Dans un transformateur, le courant fourni est donc converti en un champ magnétique par le premier enroulement puis reconverti en énergie électrique par le second. Quel intérêt ? Fondamental ! En faisant varier le nombre de spires sur chaque bobine, on fait varier proportionnellement le ratio entre la tension envoyée dans la première bobine et celle récupérée dans la seconde. Cela permet de modifier la tension (quasiment) sans perte d'intensité. Prenons un exemple avec un transformateur doté d'un ratio 5:1, composé d'une bobine de 500 spires d'un côté et de 100 spires de l'autre (les deux étant enroulées autour du même noyau). Si l'on applique une tension de 50 volts et 1 ampère en entrée sur l'enroulement de 500 spires (soit 50 W), on récupérera 10 volts et 5 ampères en sortie sur la bobine de 100 spires (soit toujours 50 W). Et cela fonctionne dans les deux sens : si on applique le courant dans l'autre sens (sur la bobine de 100 spires), on récupère 250 volts et 0,2 ampère de l'autre côté ! Seul problème : le transformateur ne fonctionne qu'en régime alternatif. Pour abaisser ou augmenter une tension continue, il faut impérativement découper le signal pour le rendre alternatif puis le reconvertir en continu. C'est exactement le principe d'une alimentation à découpage de PC (voir ci-contre).



Le transfert d'énergie dans un transformateur.

CONVERSION HAUTE-FRÉQUENCE

Au siècle dernier, les blocs secteur qui servaient à alimenter les appareils électriques en basse tension étaient très volumineux. Il s'agissait d'alimentations linéaires, dotées d'un transformateur qui convertissait directement les 230 V du secteur en 12 V (par exemple). Un modèle de seulement 20 W pouvait peser un kilo. Aujourd'hui, les alimentations à découpage – comme celles des PC portables haut de gamme – peuvent délivrer 200 W tout en pesant trois fois moins. On y trouve pourtant toujours un (minuscule) transformateur. Par quel miracle ? Grâce au découpage. Les 230 V du secteur sont d'abord lissés en tension continue puis reconvertis en courant alternatif à haute fréquence (~50/100 kHz) à l'aide de transistors. Celle-ci permet de réduire drastiquement les pertes de conversion par rapport au 50 Hz du secteur et donc la taille des transformateurs.

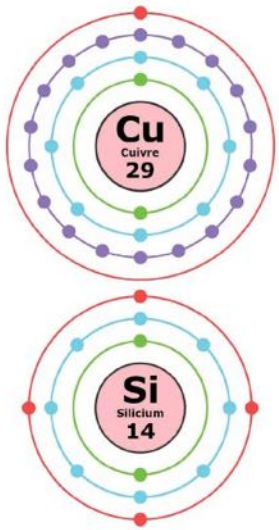
Les diodes

Parlons maintenant des composants dits "actifs" en commençant par le plus simple d'entre eux : la diode. Elle se comporte comme une valve anti-retour qui ne laisse circuler le courant électrique que dans un seul sens. Ses usages sont innombrables puisqu'ils s'étendent jusqu'à la production de lumière avec les fameuses LED. Comprendre le fonctionnement d'une diode permet de s'aventurer dans le monde des semi-conducteurs et des composants plus complexes. C'est pourquoi nous allons rentrer dans les détails.

Ça te Graetz ?

Les diodes, outre leur rôle de protection contre les inversions de polarité, servent aussi à de nombreux autres usages. Il existe ainsi des diodes électroluminescentes (LED) qui produisent de la lumière via un phénomène effroyablement compliqué (que je ne vous expliquerai pas ici, n'insistez pas). Une autre utilité des diodes est le redressement qui permet de transformer un signal alternatif en signal continu. Pour cela, on utilise 4 diodes interconnectées dans un montage dit "pont de Graetz" qui transforme (redresse) les alternances négatives en alternances positives. Au lieu d'un signal qui oscille de -100 V à +100 V, on récupère un signal toujours alternatif, mais entre 0 et +100 V. Il ne manque plus qu'un condensateur de lissage pour obtenir du 100 V (presque) continu. Toutes les alimentations à découpage comportent un tel pont de diode : avant d'être découpés, les 230 V du secteur EDF (qui oscille entre +320 V et -320 V) est converti en une tension continue de 320 V. Les diodes sont des composants extrêmement robustes qui peuvent supporter des tensions très élevées, couramment 1 000 V en polarisation inverse.

Dans l'introduction, nous avons parlé des propriétés des matériaux en les classant dans deux groupes : les conducteurs et les isolants. Il existe toutefois une troisième classe de matériaux dont la découverte est à l'origine de l'électronique moderne : les semi-conducteurs. Ceux-ci se distinguent par leur conductivité qui varie justement en fonction de divers paramètres, et en particulier de la tension. Cette notion étant fondamentale puisqu'à la base même de l'électronique, il convient de l'expliquer le plus précisément possible.
Dans un atome, les électrons qui orbitent autour du noyau sont positionnés en couches (ou niveaux). La dernière couche d'électrons, dite "périphérique" ou "de valence", ne peut en contenir plus de 8. Ce sont eux – les électrons de valence – qui peuvent s'affranchir de la force d'attraction de leur noyau lorsqu'ils sont attirés par un autre atome. En circulant d'un atome à un autre, ils créent un courant électrique. Pour qu'un électron puisse quitter son noyau, il faut que la dernière couche n'en contienne pas plus de 3. Un atome disposant de 8 électrons sur sa couche de valence (soit le maximum) s'avère en effet extrêmement stable. Il ne peut que très difficilement perdre un électron et devient en conséquence un excellent isolant électrique.
Prenons par exemple un atome d'oxygène. Son noyau contient 8 protons (N° atomique 8) et, comme tout atome dans son état normal, un nombre égal d'électrons. Ceux-ci sont répartis en deux couches : la première en contient 2 et la seconde – la couche de valence – en embarque 6. L'atome d'oxygène ne laissera donc pas partir ses électrons facilement, ce qui en fait un isolant électrique ; à l'inverse, il aura même fortement tendance à en attirer deux autres afin de combler sa couche de valence. Il deviendra un ion oxyde O^{2-}. À l'inverse, un atome de cuivre (29 protons) est composé de quatre couches comprenant respectivement (de la plus proche à la plus éloignée du noyau) 2, 8, 18 et 1 seul électron. L'unique électron présent sur sa couche de valence en fait un excellent conducteur ; il pourra être éjecté facilement et ainsi permettre à l'électricité de circuler. Les éléments qui constituent les meilleurs conducteurs (comme l'argent, l'or ou le platine) disposent tous d'un unique électron sur leur couche périphérique. L'aluminium, un conducteur correct mais moins efficace, en embarque 3.

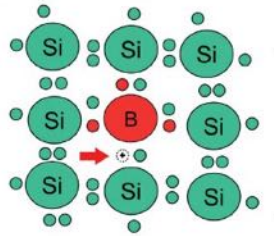


L'atome de cuivre est doté de 29 électrons : la couche de valence, en rouge, ne contient qu'un seul électron. L'atome de silicium dispose de 4 électrons périphériques qui peuvent créer des structures cristallines avec 4 autres atomes du même type.

Dopage pour tous

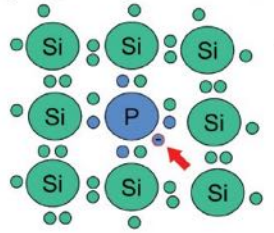
Revenons aux semi-conducteurs comme le germanium ou le silicium. Leurs atomes se distinguent justement par leur configuration : ils disposent de 4 électrons sur la couche périphérique, soit exactement la moitié d'une couche saturée. Chaque atome va naturellement mettre en commun chacun de ses 4 électrons libres avec ceux des atomes voisins afin que tous puissent remplir leurs couches périphériques (voir schéma). Évidemment, dans cette configuration cristalline, le silicium offre toutes les caractéristiques d'un excellent isolant puisqu'il n'a plus d'électrons libres. Pour obtenir le matériau semi-conducteur aux propriétés si intéressantes en électronique, il convient d'y injecter des atomes "parasites" qui vont rompre cet équilibre parfait en créant artificiellement des surplus ou des déficits d'électrons. Ce procédé s'appelle le dopage. Si l'on injecte des atomes dotés de 5 électrons sur leur couche de valence (comme l'arsenic ou le phosphore), un électron se retrouvera en surplus : il s'agira d'un dopage de type N (Négativement chargé, puisque les électrons ont une charge négative).
À l'inverse, si c'est un atome doté de seulement 3 électrons libres – comme le bore ou le gallium – qui se retrouve dans le cristal de silicium, il y aura déficit d'un électron. Dans ce cas, on dira que le semi-conducteur a subi un dopage de type P (Positivement chargé). Il en résulte un "trou" dans la bande de valence.

Dopage "P"
Injection d'impuretés de Bore dans une structure de Silicium



Silicium (4 électrons périphériques)
+ Bore (3 électrons périphériques)
= < 8 : Déficit d'un électron / Création d'un trou

Dopage "N"
Injection d'impuretés de Phosphore dans une structure de Silicium



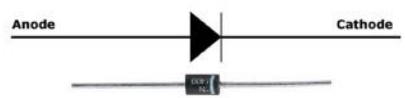
Silicium (4 électrons périphériques)
+ Phosphore (5 électrons périphériques)
= > 8 : Excès d'un électron

Diode : une jonction PN

Imaginons maintenant que l'on juxtapose un bout de silicium dopé P avec un autre dopé N. À l'endroit exact de la jonction, ce sera la fiesta électronique : les électrons côté N vont boucher les trous côté P dans une petite région intermédiaire qui deviendra neutre électriquement, c'est-à-dire un isolant. On obtiendra donc une zone isolante entourée de deux zones capables de porter une charge électrique : l'une en excès d'électrons (N), l'autre en déficit (P). Imaginons maintenant que nous appliquions une tension électrique (par exemple une pile) de part et d'autre de cette jonction P-N. Si l'on connecte le – de la pile sur la zone P et le + sur la zone N (polarisation inverse), aucun courant ne circulera : les électrons en provenance du – de la pile (vous n'avez pas oublié que les électrons circulent du – au + ?) iront boucher les trous de la zone P et n'auront aucun intérêt à tenter d'aller dans la zone N, déjà en excès d'électrons et en plus séparée par une zone isolante. Si, en revanche, on inverse la polarité et qu'on branche le – de la pile sur la zone N et le + sur la zone P (polarisation directe), les trous qu'aiment tant les électrons se retrouvent de l'autre côté de la barrière isolante et ils vont tenter d'y parvenir. Heureusement, celle-ci est suffisamment étroite pour qu'au-delà d'une certaine tension (environ 0,7 V sur du silicium), ils réussissent à traverser l'obstacle isolant. Le courant se met alors à circuler. Dans la pratique, une telle jonction PN constitue une diode. Celle-ci permet par exemple de ne pas griller vos appareils si vous branchez le chargeur à l'envers. Si la polarité est inversée, le courant ne circulera pas.

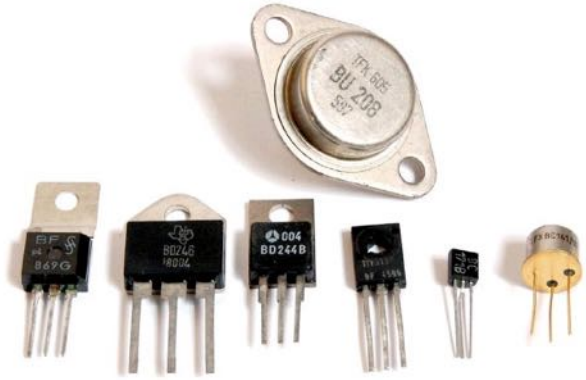
Diode en sens passant

Diode en sens inverse



Les transistors

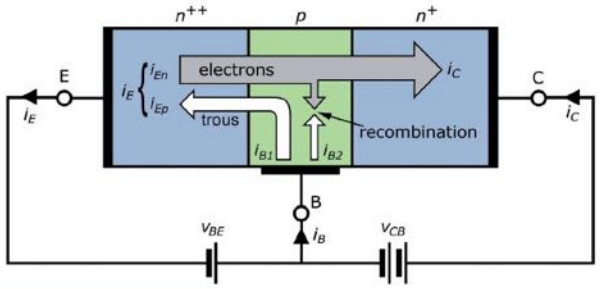
Il est maintenant temps de parler du composant fondamental qui a donné naissance à toute l'électronique moderne. En offrant la possibilité de commander ou d'amplifier un courant électrique, tout en consommant peu d'énergie lui-même et dans un volume minuscule, le transistor a révolutionné notre vie quotidienne.



Du point de vue de sa construction, le transistor n'est pas bien éloigné de la diode. On peut le voir comme deux diodes connectées en série tête-bêche, ce qui donne une juxtaposition de trois couches de semi-conducteurs dopés. Le transistor est constitué d'une jonction triple N-P-N (ou P-N-P) à trois broches alors que la diode se limite à un simple dipôle de type P-N (comme nous l'avons vu dans les pages précédentes). Dans son mode de fonctionnement le plus basique, il peut être comparé à une vanne de circuit d'eau : actionner la vanne nécessite peu de puissance mais permet de contrôler un fort débit d'eau. Sur un transistor, c'est le même principe : l'arrivée d'eau (de courant) est appelée *émetteur* (jonction de droite), la sortie *collecteur* (jonction de gauche) et la commande *base* (jonction du milieu). Si aucun courant ne circule dans la base, aucun courant ne circulera entre l'émetteur et le collecteur, comme avec un interrupteur fermé (OFF). Si vous appliquez un courant supérieur au seuil de *saturation* sur la base, le transistor agira comme un interrupteur ouvert (ON) entre l'émetteur et le collecteur ; un courant potentiellement très important pourra alors circuler. Dans ce mode de fonctionnement, le transistor agit en *commutation* et permet d'allumer ou d'éteindre une forte charge (un ventilateur par exemple) avec un très faible signal de commande, par exemple issu d'un microcontrôleur.

Le transistor peut également fonctionner en amplification, toujours comme un robinet qui modifie le débit d'eau à mesure qu'on le tourne (= qu'on applique un courant plus ou moins important sur la base). Si l'on fait circuler sur la base un courant compris entre 0 et le seuil de saturation, celui-ci se répercutera de façon démultipliée sur le courant qui circule entre l'émetteur et le collecteur. Le pouvoir d'amplification des transistors communs est souvent compris entre 100 et 200, ce qui signifie qu'avec un courant de commande (sur la base) de 10 mA, on fera passer entre 1 et 2 A maximum entre l'émetteur et le collecteur. C'est ainsi qu'on amplifie un signal audio par exemple. Le transistor original *bipolaire*, constitué de deux jonctions P-N, date des années 1950 et reste très largement utilisé aujourd'hui. Il présente des caractéristiques idéales pour l'électronique analogique.

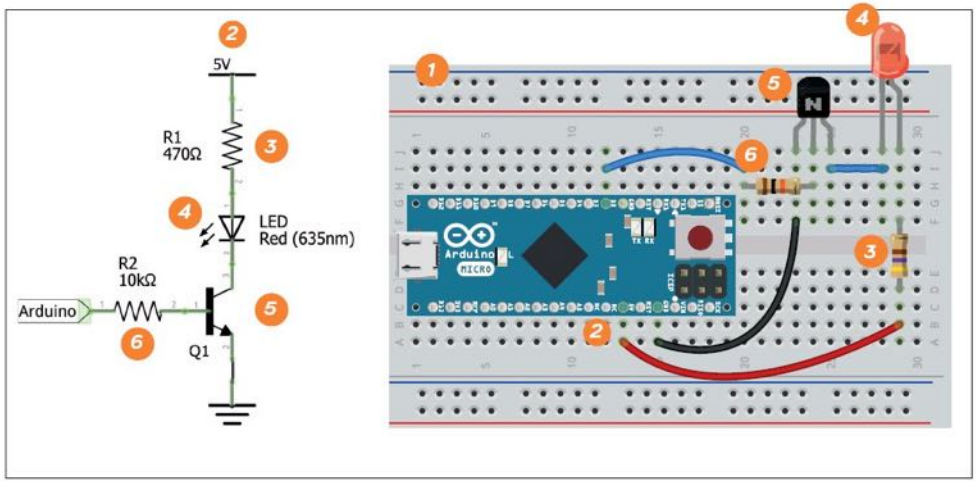
MOSFET. Toutefois, avec l'arrivée de l'électronique numérique, un nouveau type de transistor s'est démocratisé : le MOSFET, ou *metal-oxide semiconductor field-effect transistor*. Comme son nom l'indique, celui-ci est basé sur une jonction métal/semi-conducteur et non plus semi-conducteur/semi-conducteur. Le MOSFET s'avère particulièrement adapté pour fonctionner en commutation simple et cela pour deux raisons. D'abord, les pertes sur le circuit de puissance (entre le collecteur et l'émetteur) sont beaucoup plus faibles que sur un transistor bipolaire : il chauffera donc nettement moins à courant égal. Ensuite, la base se commande avec une tension et non plus avec un courant, ce qui permet d'utiliser une énergie extrêmement faible pour faire commuter le transistor. Il est aussi facile de chaîner un grand nombre de MOSFET en parallèle (en passant par leur base) pour multiplier encore le courant commandé. Les fabricants utilisent largement les MOSFET pour contrôler de très larges courants comme ceux qui circulent dans l'étage d'alimentation d'une carte mère (plusieurs dizaines d'ampères).



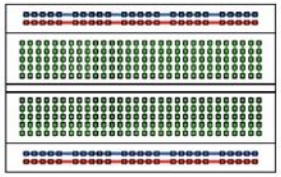
Bien utiliser un transistor

AU MOINS DANS LE CAS LE PLUS SIMPLE…

Le transistor étant le composant le plus important en électronique, il convient de bien comprendre son fonctionnement. Pour les débutants, nous avons donc conçu le montage le plus simple possible : allumer une LED en utilisant un transistor en commutation, commandé par un Arduino. Nous tenterons d'expliquer toutes les difficultés qui peuvent survenir dans ce genre de cas.



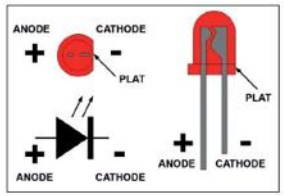
1 La platine d'essai (plus communément appelée par son nom anglais *breadboard*) permet de relier des composants entre eux sans soudure. Elle est constituée de rangées de six trous électriquement connectés en interne. Chaque rangée reste isolée des autres. On utilise des cavaliers constitués de fils rigides pour effectuer les liaisons entre deux rangées. Ici, nous avons utilisé le bleu, le noir (pour la masse) et le rouge (pour la tension d'alimentation).



2 Pour des raisons de simplification, nous considérerons que l'Arduino Micro est alimenté par son port micro-USB. Notre LED sera alimentée par le +5 V produit par la carte. Il reste bien sûr possible d'utiliser une autre source de tension plus importante, comme du +12 V pour alimenter un ventilateur (à la place de la LED) par exemple. Seul impératif : la masse doit être commune avec celle de l'Arduino.

3 Pour limiter le courant qui circulera dans la LED (et dans le transistor), une résistance est obligatoire, sans quoi vous créeriez un court-circuit. Nous utilisons ici une résistance de 470 ohms. À noter que la résistance pourrait très bien être positionnée après la LED. L'important reste que le courant soit limité à un moment ou un autre dans son cheminement.

4 L'anode de la LED (diode) est reliée à la borne positive de l'alimentation (ici, +5 V) par l'intermédiaire de la résistance. La cathode est connectée au collecteur du transistor. Les deux broches se distinguent facilement : la broche de l'anode est plus longue et le côté "cathode" est marqué par un plat sur le boîtier.

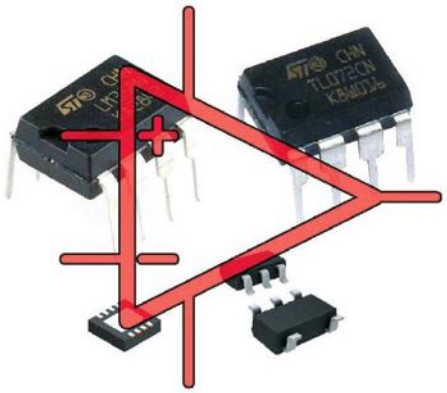


5 Nous utilisons ici un transistor bipolaire NPN de type BC548, adapté au sens du courant de notre montage (les modèles PNP fonctionnent avec une polarité inverse). On y trouve trois broches : le collecteur, connecté à la LED ; l'émetteur, relié à la masse ; et la base, commandée par l'Arduino. Un courant sur la base – en configurant la sortie de l'Arduino à "1" – entraîne le passage du courant entre le collecteur et l'émetteur (maximum 30 V et 500 mA). Et la LED s'éclaire. Attention : contrairement aux LED, les broches n'ont pas de disposition "standard".

6 Cette résistance s'intercale entre la sortie de l'Arduino et la base du transistor. Elle limite très fortement le courant de commande nécessaire à la commutation du transistor. Nous avons choisi ici un modèle de 10 kohms. En utilisant un MOSFET plutôt qu'un transistor bipolaire, cette résistance devient facultative.

Les amplificateurs opérationnels

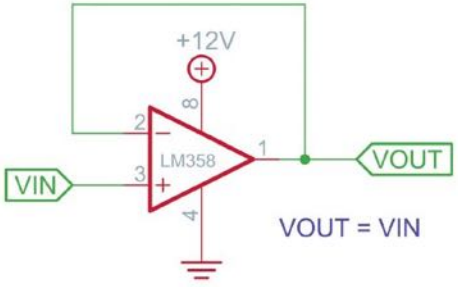
Les amplificateurs opérationnels (AOP) font partie des circuits intégrés les plus simples. Ils sont principalement constitués de transistors montés en chaîne et permettent d'amplifier très fortement un signal analogique. Comme les transistors, on peut également les utiliser pour booster un courant, par exemple lorsque les sorties d'un microcontrôleur s'avèrent trop limitées.



Un AOP dispose de deux entrées et d'une sortie : il amplifie la différence de tension entre les deux entrées en supprimant au passage plusieurs limitations des transistors. Pour faire simple, l'AOP s'utilise principalement en régime linéaire (donc en électronique analogique) et permet d'atteindre un gain entre l'entrée et la sortie de plusieurs millions, là où le transistor bipolaire seul est limité à environ 100-200. Un AOP peut sans problème amplifier une tension de quelques microvolts pour en faire des volts en sortie. Pour faire hurler les puristes, disons que l'AOP se comporte comme un super-transistor spécialisé dans le régime linéaire (analogique) alors que le MOSFET peut être vu comme un super-transistor dédié au régime de commutation (numérique). L'AOP est particulièrement utilisé lorsqu'il s'agit d'amplifier largement un signal faible tout en gardant une grande précision "analogique". C'est le cas des amplis audio et des cartes son par exemple. Voyons deux cas courants.

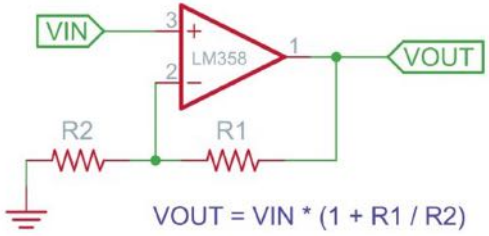
MONTAGE EN SUIVEUR

C'est la configuration la plus simple. La sortie de l'AOP est directement connectée à l'entrée négative dite "inverseuse". Dans cette configuration, la tension de sortie sera toujours identique à la différence de potentiel entre les deux entrées. On retrouvera donc en sortie (VOUT) la tension appliquée sur la broche positive (VIN, dite "non inverseuse"). Vous pensez probablement que ce genre de montage ne sert à rien. Détrompez-vous. Le principal intérêt vient du courant quasiment nul requis en entrée de l'AOP – quelques microampères – alors qu'en sortie, il devient possible de consommer plusieurs centaines de mA (voire d'ampères pour les plus gros modèles). Il fait ainsi office de *buffer*.



MONTAGE EN AMPLIFICATEUR

Le fonctionnement en amplification simple reste l'un des montages les plus courants. On applique toujours la tension à amplifier VIN sur l'entrée positive non inverseuse, mais on diminue artificiellement le retour – entre la broche négative inverseuse et la sortie – à l'aide de deux résistances qui forment un pont diviseur (voir page 12). La valeur de ce dernier conditionne directement le facteur d'amplification. Par exemple, si l'on utilise deux résistances de 100 kΩ (R1) et 10 kΩ (R2), on obtient un pont diviseur qui, en son milieu, délivrera une tension 10 fois inférieure à celle appliquée en amont. L'AOP compensera en fournissant une tension 10 fois supérieure pour maintenir l'équilibre entre ses entrées et sa sortie. Une tension VIN de 0,1 V donnera donc une tension VOUT de 1 V.



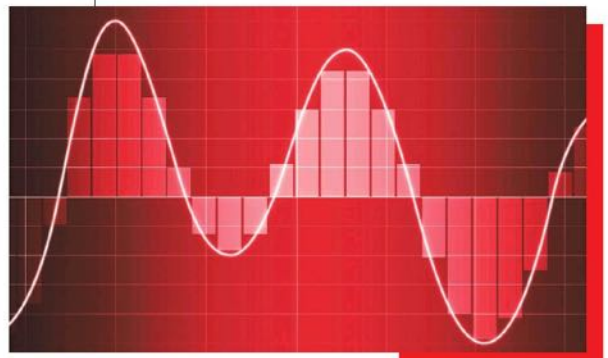
Les convertisseurs analogique / numérique

Indispensables pour lire une tension issue d'un capteur ou pour convertir un fichier numérique en son, les convertisseurs analogique et numérique permettent aux microcontrôleurs de communiquer avec l'extérieur. Un DAC (*digital-to-analog converter*) transforme un nombre binaire (numérique) en une tension analogique alors qu'un ADC (*analog-to-digital converter*) effectue l'opération inverse.

Depuis plus de 30 ans, l'électronique est majoritairement numérique. Qu'il s'agisse de votre PC, d'un Arduino ou d'un Raspberry Pi, toutes les opérations internes traitées ne sont qu'une suite de 0 et de 1. Or, nous autres humains réagissons principalement à des stimuli analogiques, comme le son de la voix par exemple. Les grandeurs physiques de notre monde (température, lumière…) sont également analogiques. Pour les convertir en valeurs compréhensibles par un microprocesseur, il convient donc de les numériser. On utilise pour cela des ADC. On en trouve intégrés d'office dans certains microcontrôleurs (comme ceux des Arduino) mais pas dans d'autres : les SoC des Raspberry Pi en sont dépourvus par exemple ; il faut alors passer par un composant externe pour effectuer cette opération.

LES ADC. Ils convertissent une tension en valeur binaire. La communication s'effectue avec le microcontrôleur par le biais d'un bus série comme l'I2C ou le SPI. Les principaux paramètres d'un ADC sont la tension d'entrée admissible, la fréquence maximale de numérisation et surtout la résolution, exprimée en bits. Prenons le cas d'un ADC 8 bits. Dans ces 8 bits, on peut stocker 256 valeurs différentes, comprises entre 00000000 (0) et 11111111 (255). Si l'on alimente l'ADC avec une référence de tension de 5 V, celui-ci sera capable de distinguer un écart de tension maximum sur ses entrées de 5 V/256 = ~19,5 mV. En connectant n'importe quelle tension comprise entre 0 et 5 V sur ses entrées, on obtiendra en sortie une valeur numérique proportionnelle, avec une précision de 19,5 mV. Prenons quelques exemples. Si nous appliquons une tension de 5 V, l'ADC fournira une valeur de 255 (le maximum) au microcontrôleur. Avec une tension de 2,5 V (soit 5 V / 2), on obtiendra 256 / 2, soit 128. Et ainsi de suite. Prenons le calcul dans le sens inverse en poussant un peu la difficulté. L'ADC intégré à votre carte dispose d'une résolution de 10 bits et vous lui fournissez une référence de tension maximale de 1,1 V. Dans votre programme, vous lisez le résultat : 598. À quelle tension d'entrée correspond-il ? Facile ! Votre ADC 10 bits encode une valeur comprise entre 0 (pour 0 V) et 1024 (pour son maximum, soit ici 1,1 V). 598 correspond donc à 1,1 V / 1024 * 598, soit ~642,38 mV. S'il s'agit d'un capteur de température calibré à 0 °C = 500 mV puis 1 °C/10 mV, la température mesurée sera de (642,38 – 500) / 10 = 14,2 °C.

LES DAC. Dans la plupart de vos montages, vous ne rencontrez que peu de DAC. Bien plus complexes que les ADC, ils servent très majoritairement à créer un signal analogique sonore à l'aide de données numériques (comme dans les cartes son). À noter qu'il est possible de créer un ersatz de DAC avec une simple sortie numérique à l'aide d'une technique appelée PWM (*pulse width modulation* - modulation de largeur d'impulsions). Elle consiste à créer un signal alternatif binaire à fréquence fixe, mais doté d'un rapport cyclique (durée des phases "0" et "1") variant de 0 à 100 %. À l'aide d'un filtre, ou plus simplement d'un condensateur, on obtient ensuite une tension correspondant à la moyenne du signal, liée à son rapport cyclique.

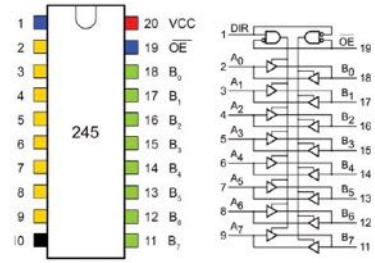


Les régulateurs

SURVEILLEZ VOTRE TENSION !

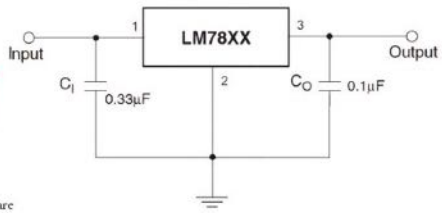
L'adaptation de tension reste souvent l'un des premiers problèmes que tout amateur d'électronique rencontrera. Qu'il s'agisse de l'alimentation de la carte en elle-même ou des tensions qui circuleront sur les broches entrées/sorties, il convient de toujours veiller à ne pas dépasser les maximums spécifiés par les constructeurs. Voici quelques conseils rapides pour éviter que la fumée magique ne se dégage trop vite de vos composants…

Avant de mettre en service pour la première fois un montage électronique, vous devrez vérifier minutieusement que la tension et le courant qui circulent dans les différentes parties de votre circuit sont bien corrects. En cas de différences, vous devrez adapter les tensions, souvent en les diminuant.



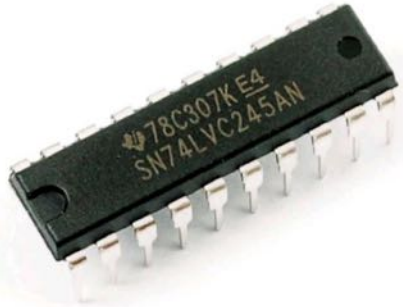
RÉGULATION D'ALIMENTATION

Imaginons que vous disposiez d'une tension d'alimentation de 12 V, mais que vous ayez besoin de 9 V pour un moteur par exemple. Il vous faudra un régulateur pour abaisser cette tension. Les plus basiques font partie de la série 78xx. Il s'agit de régulateurs dits "linéaires" qui se contentent de dissiper en chaleur l'excès de tension. Ces composants sont très simples à utiliser puisqu'ils ne comportent que trois broches : l'entrée (que vous connecterez au 12 V dans notre exemple), la masse et la sortie (où vous récupérerez le 9 V). Pour du 9 V, il conviendra de choisir un régulateur de type 7809. La plupart des cartes comme l'Arduino comportent un composant de ce type pour convertir le 7-12 V de leur alimentation en +5 V utilisable par le microcontrôleur. Seul problème : ils chauffent beaucoup, ce qui limite le courant utilisable. Ne comptez pas utiliser le 5 V fourni par un Arduino pour alimenter un composant gourmand comme un ventilateur ou même un gros relais par exemple ! Le régulateur intégré ne pourra fournir plus de 200 mA dans le meilleur des cas. Avec un second régulateur externe, vous pourrez consommer jusqu'à 500 mA et même parfois jusqu'à 1 A si vous le surmontez d'un dissipateur. Pour aller plus haut, il existe des régulateurs DC-DC à découpage comme on en trouve dans les alimentations. Nous vous conseillons les modèles tout faits de Traco Power ou de XP Power. Leur efficacité atteint 90 %, ils sont aussi simples à utiliser que les régulateurs linéaires et peuvent délivrer un courant conséquent (plusieurs ampères selon les modèles). Seul bémol : ils ne sont pas donnés (environ 10 €).



RÉGULATION GPIO

Si vous optez pour une carte de développement qui n'accepte que du 3,3 V sur ses entrées/sorties (Arduino Zero, Raspberry Pi…) et que vous souhaitez utiliser un capteur travaillant en 5 V (ou vice-versa), il vous faudra aussi adapter la tension. Cette fois, vous n'aurez pas besoin d'un courant important mais plutôt de travailler à haute fréquence. Un vulgaire transistor pourrait parfaitement faire l'affaire, mais il existe des composants (*level shifter*) plus pratiques quand il s'agit de convertir plusieurs lignes numériques simultanément. Le 74LVC245 (maximum 5 V) ou le 74HC4050 (maximum 15 V) en font partie. Vous leur fournissez une tension d'alimentation (3,3 V) et ils convertiront automatiquement le +5 V appliqué sur l'une de ses huit entrées en 3,3 V sur la sortie correspondante. Ce type de composant reste toutefois unidirectionnel, mais s'avère parfait pour commander un afficheur OLED en 3,3 V avec un Arduino Uno (5 V) par exemple. À noter qu'il existe également des versions bidirectionnelles pour la communication avec un capteur complexe.



> INITIATION À L'ÉLECTRONIQUE

Le b.a.-ba du matériel

Réjouissez-vous, le blabla purement théorique est déjà terminé. Dès les prochaines pages, nous passerons à la pratique afin de poursuivre notre initiation. Il vous faudra pour l'occasion vous procurer le petit matériel indispensable que nous allons décrire ici. Inutile de dépenser des fortunes pour un oscilloscope haut de gamme. Le strict minimum ne vous coûtera pas plus que quelques dizaines d'euros. Vous pourrez toujours opter pour du matériel plus sérieux par la suite.

TOURNEVIS ET PINCES

Les outils de base que quiconque se doit d'avoir à portée de main sont les tournevis et quelques pinces. Attention : il s'agit de modèles miniatures adaptés à l'électronique et pas de matériel issu d'une caisse à outils de tractopelle. Pour ce qui est des tournevis, plus ils sont petits, plus ils sont fragiles. On trouve partout des kits de "tournevis de précision" pour quelques euros. Soyez assuré que leurs embouts casseront dès les premières utilisations. Les marques Bost (Casto) et Roebuck (Farnell) offrent un bon rapport qualité/prix. Évitez Facom en revanche : depuis son rachat par Stanley et la délocalisation généralisée en Chine, ses outils ne valent guère mieux que le bas de gamme. Procurez-vous deux ou trois (petites) tailles en plats et en cruciformes. Pour les pinces, vous devez disposer d'une plate à mâchoire longue et d'une coupante, adaptées à l'électronique. La marque allemande Knipex représente le top, mais comptez minimum 30 euros la pince. Des marques à prix plus raisonnables existent (Toolcraft chez Conrad par exemple) mais ne dépensez pas moins de 10 euros, sinon elles finiront à la poubelle en deux semaines. Vous devriez vous en sortir pour 40 euros l'ensemble tournevis + pinces.

MULTIMÈTRE

Le multimètre permet de mesurer très rapidement la tension, le courant, la résistance et parfois bien d'autres paramètres comme la capacité. La précision des modèles d'entrée de gamme reste largement suffisante pour une initiation en basse tension continue. La plupart des revendeurs proposent des multimètres numériques aux alentours de 30 euros qui feront parfaitement l'affaire. Ne descendez pas en dessous car les "premiers prix" à 10-15 euros souffrent souvent de graves problèmes de sécurité. Un modèle "*autorange*" (inutile de sélectionner manuellement le calibre) est fortement conseillé. Le VC175 de Voltcraft (Conrad) semble un bon choix. Et si vous comptez vous lancer dans l'électronique sur le long terme, rien ne vaut un multimètre Fluke que vous garderez des années… tout comme le souvenir de la facture.

PLAQUETTE D'ESSAI ET CAVALIERS

La plaquette d'essai ou platine enfichable (*breadboard* en anglais) est indispensable pour effectuer des expérimentations sans soudure. Elle est composée de nombreux contacts disposés en rangées interconnectées dans lesquels vous enfichez vos composants qui peuvent être modifiés en permanence ensuite. Les rangées sont connectées entre elles par le biais de cavaliers (*jumpers*), des petits fils conducteurs pré-pliés et réutilisables. Achetez donc une platine composée d'environ 800 contacts (10 euros), une boîte de cavaliers rigides (10 euros) et quelques cavaliers flexibles (5 euros).

ALIMENTATION DE LABORATOIRE

Une alimentation de laboratoire vous offrira une sécurité et une flexibilité autrement supérieures que des piles, batteries et autres adaptateurs secteur. C'est de loin l'appareil le plus cher pour un débutant, mais il est difficile d'y couper tant les services rendus facilitent la vie. Son but est d'offrir une tension stabilisée et réglable, avec souvent une limitation de courant pour éviter de tout griller en cas de court-circuit ou de placement malheureux d'un composant. Comptez environ 60-70 euros pour un premier prix. Si votre budget est limité, il reste toujours la possibilité d'utiliser les régulateurs des plateformes de développement et le port USB de votre PC. Nous tenterons dans la mesure du possible de privilégier cette approche pour la suite, mais cela ne sera pas toujours possible.

FER À SOUDER

Il est parfaitement possible de débuter en électronique sans rien souder, en utilisant une platine d'essai. C'est d'ailleurs ce que nous ferons dans le prochain numéro. L'usage du fer à souder devient toutefois quasi obligatoire à partir d'un certain niveau, ne serait-ce que pour souder un connecteur. Les modèles les plus pratiques sont à température variable mais leurs prix débutent à 70 euros. Plus bon marché, un fer à souder à température fixe fera l'affaire pour un début. Nous ne pouvons que vous conseiller le WM20L de Weller, de loin la meilleure marque dans ce domaine. Pour 25 euros, difficile de faire mieux. Prévoyez également un peu de tresse à dessouder (5 euros), éventuellement une "troisième main" (5-10 euros), et surtout une bobine de fil de soudure (5-10 euros). Choisissez un fil de 1 mm et un alliage étain/plomb de type Sn60Pb40, beaucoup plus facile à souder. À l'inverse, la soudure à l'argent est plus chère et dispose d'un point de fusion plus élevé. Le plomb étant encore autorisé pour les réparations et le hobby (mais interdit dans l'industrie), autant l'utiliser ; ces quelques points de soudure ne mettront pas la planète en péril.

CARTE DE DÉVELOPPEMENT

L'électronique moderne ne se contente pas uniquement d'une platine d'essai mais s'accompagne de plus en plus souvent par un microcontrôleur. La plateforme la plus adaptée aux premiers pas est sans conteste l'Arduino. C'est d'ailleurs vers elle que nous nous tournerons lorsqu'il s'agira de mettre le pied à l'étrier. La version la plus courante et la moins chère est pour l'heure l'Arduino Uno R3, que l'on trouve un peu partout aux environs de 25 euros. Elle est largement suffisante pour démarrer et même se familiariser de manière approfondie avec les microcontrôleurs. Côté Raspberry Pi, bien que la plateforme n'ait initialement pas été conçue pour l'électronique, elle bénéficie d'un tel engouement que les concepteurs ont rajouté des fonctionnalités intéressantes dans les dernières révisions (2 et 3). Si l'Arduino reste idéal pour débuter, le Rapsberry Pi offrira des possibilités de programmation bien supérieures malgré des interfaces beaucoup moins simples à utiliser.

COMPOSANTS DIVERS

Pour vous lancer dans le bain, vous aurez besoin d'un petit stock de composants de base, à commencer par des résistances et des condensateurs. Ceux-ci ne coûtent que quelques centimes ou dizaines de centimes à l'unité, mais il est rageant de devoir faire une commande et payer des frais de port pour une résistance à 2 cts. La plupart des revendeurs proposent des "kits" de résistance incluant 15 ou 20 pièces des 20 ou 30 valeurs les plus courantes pour environ 15 euros. La même chose existe pour les condensateurs électrolytiques. En plus de ces composants passifs, quelques autres vous seront très souvent utiles. Citons par exemple quelques LED 3 et 5 mm (0,05 €/pièce), deux-trois régulateurs linéaires de tension (type 7805 et 7812, 0,50 €/pc) et surtout une bonne dizaine de transistors bipolaires NPN (type 2N2222, 2N3904 ou équivalent) et PNP (2N3906…). Bien d'autres composants utiles pour l'expérimentation comme des boutons poussoirs, des interrupteurs, des relais, des petits moteurs, etc. peuvent être récupérés dans de vieux appareils mis au rebut. Pensez-y !

Où acheter son matos ?

QUELQUES BONNES ADRESSES

Les plateformes de développement électronique et leurs écosystèmes (composants, connecteurs, etc.) étaient autrefois disponibles quasi uniquement sur les sites professionnels. Si certains vendeurs – comme Conrad ou Selectronic – s'adressaient déjà aux quelques particuliers amateurs, ils n'ont pas su capitaliser sur la récente démocratisation des Arduino et autres Raspberry Pi. Voici quelques adresses testées par nos soins pour trouver tout ce dont vous pourriez avoir besoin.

Semageek

BOUTIQUE.SEMAGEEK.COM

À l'origine simple blog dédié à l'électronique créé en 2009, Semageek a depuis ouvert une boutique dédiée qui propose un large choix de cartes de développement et de composants électroniques. Les prix sont très raisonnables, tout comme les frais de port et les délais d'expédition. Si vous avez besoin d'un Arduino Uno ou d'un modèle compatible plus rare comme les Teensy ou les Feather, allez donc y faire un tour.

Semageek

Lextronic

WWW.LEXTRONIC.FR

Parmi les "vieux" sites français dédiés à l'électronique grand-public (Conrad, Selectronic, GoTronic, etc.), nous choisissons Lextronic. La société existe depuis 1969 (!) et propose un choix très large de composants dans de nombreux domaines. On y trouve de nombreux types d'afficheurs, des outils, des capteurs, des composants ainsi que les cartes habituelles et même de plus rares comme les BeagleBone ou les plateformes de développement avancées de MikroElektronika. Les tarifs restent toutefois assez élevés dès qu'on sort du basique.


Lextronic

Farnell

FR.FARNELL.COM

Farnell est un site professionnel qui ne joue clairement pas dans la même cour que les sites grand public. Avec un catalogue pléthorique de plusieurs centaines de milliers de composants électroniques, vous y trouverez probablement tout ce que vous ne pourriez jamais chercher. Le service, digne des pros, mérite aussi le détour : commande avant 19 h livrée le lendemain avant 10 h. Le tarif des composants reste toutefois à la mesure du service. Et n'oubliez pas que tous les prix sont hors taxe !

Farnell

Amazon

WWW.AMAZON.FR

Le Marketplace d'Amazon France reste un excellent endroit pour trouver des clones d'Arduino à très bas prix. Nous vous conseillons toutefois de ne choisir que les produits livrés par Amazon et de sélectionner les vendeurs tiers en fonction de leurs avis. À l'heure où j'écris ces lignes, on y trouve des copies conformes d'Arduino Uno pour moins de 10 euros.


Amazon

Adafruit

WWW.ADAFRUIT.COM

Le site américain Adafruit – et son égérie LadyAda – ont très largement contribué à populariser l'électronique auprès du grand public. Nous leur devons par exemple l'Arduino Micro, développé en interne puis cédé à Arduino. L'excellente boutique s'accompagne de très nombreuses explications détaillées sur le fonctionnement des composants, avec des exemples pratiques. Si vous comprenez l'anglais, il s'agit là d'une ressource de grande qualité. Seul bémol : les frais de port vers la France ne sont pas donnés. Dans le même genre, nous vous conseillons également le très bon Sparkfun (*www.sparkfun.com*).

SeeedStudio

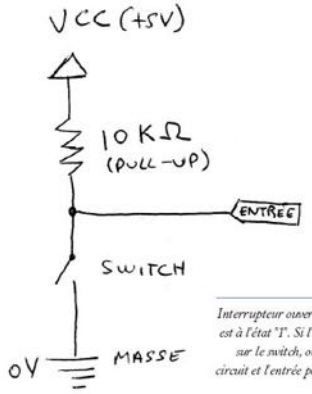
WWW.SEEEDSTUDIO.COM

Basé en Chine à Shenzhen, Seeed-Studio demeure probablement le vendeur le plus complet et le plus sérieux lorsqu'il s'agit d'électronique. Les tarifs sont inférieurs à ceux qu'on trouve sur Adafruit ou Sparkfun, tout comme les frais de port, mais gare aux délais de livraison qui peuvent prendre plus de deux semaines. Néanmoins, si comme nous vous finissez par craquer régulièrement sur de nouveaux gadgets, vous ne pourrez que trouver votre bonheur dans ce joyeux bazar.

Les entrées

POUR TOUT BIEN CAPTER...

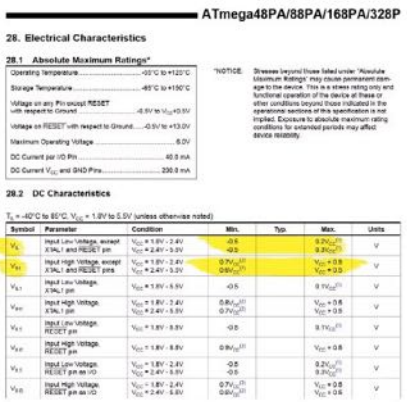
Les entrées d'un microcontrôleur permettent d'y connecter des capteurs afin d'acquérir une information. Elle peut être de deux types : numérique ou analogique. Dans le premier cas, le signal détecté se traduira par un "0" ou un "1" logique alors que dans le second, il pourra prendre l'une des nombreuses valeurs d'une large gamme. Comprendre le fonctionnement des entrées est particulièrement important pour bien débuter en électronique. Nous allons d'abord voir les caractéristiques de ces deux principales familles avant de passer à des exemples pratiques.



Interrupteur ouvert, l'entrée est à l'état "1". Si l'on appuie sur le switch, on ferme le circuit et l'entrée passe à "0".

Entrées numériques

Les entrées numériques servent à connecter des capteurs simples comme des interrupteurs, des boutons poussoirs, mais également des capteurs de fermeture ou d'intrusion. Ceux-ci renvoient un signal binaire (on/off). En clair, le programme qui s'exécute sur le microcontrôleur va "lire" la valeur d'une entrée sous la forme d'un unique bit de donnée. Si l'entrée est connectée à la masse (0 V), la valeur correspondante sera un "0" logique. Au contraire, si elle est reliée à la tension d'alimentation (VCC), on récupérera un "1" logique. Lors de l'utilisation d'une entrée numérique, il convient de ne jamais dépasser cette tension VCC sous peine d'endommager le microcontrôleur. Si celui-ci est alimenté en +3,3 V et que vous appliquez sur l'une de ses entrées numériques une tension de +5 V, vous le détruirez presque instantanément. L'inverse est en revanche possible puisque ces entrées disposent d'une tension de seuil indiquée clairement dans le datasheet du composant, généralement sous les appellations VIL (*voltage input low*) et VIH (*voltage input high*). Sur l'ATmega328 (qui équipe l'Arduino Uno), on trouve par exemple une valeur VIL de -0,5 V à 0,3 * VCC et VIH de 0,6 * VCC à VCC+0,5. Vu que la puce est alimentée en +5 V, une tension comprise entre -0,5 V et 1,5 V sera considérée comme un "0" logique alors qu'une tension entre 3 V et 5,5 V sera interprétée comme un "1". Entre ces deux valeurs, c'est le hasard et il convient de ne pas s'y aventurer.



ATmega48PA/88PA/168PA/328P

28. Electrical Characteristics

28.1 Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except RESET with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on RESET with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

28.2 DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 1.8V to 5.5V (unless otherwise noted)

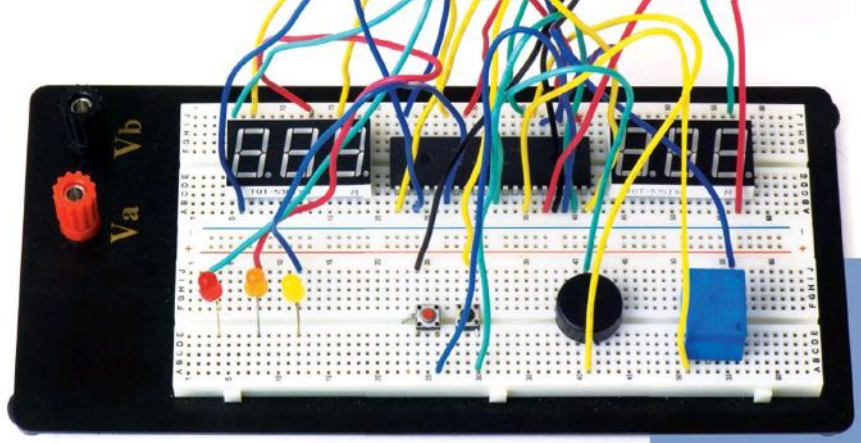
Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage, except XTAL1 and RESET pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	-0.5 -0.5		0.2V_{CC}[1] 0.3V_{CC}[1]	V
V_{IH}	Input High Voltage, except XTAL1 and RESET pins	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	0.7V_{CC}[2] 0.6V_{CC}[2]		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL1}	Input Low Voltage, XTAL1 pin	V_{CC} = 1.8V - 5.5V	-0.5		0.1V_{CC}[1]	V
V_{IH1}	Input High Voltage, XTAL1 pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	0.8V_{CC}[2] 0.7V_{CC}[2]		V_{CC} + 0.5 V_{CC} + 0.5	V
V_{IL2}	Input Low Voltage, RESET pin	V_{CC} = 1.8V - 5.5V	-0.5		0.1V_{CC}[1]	V
V_{IH2}	Input High Voltage, RESET pin	V_{CC} = 1.8V - 5.5V	0.9V_{CC}[2]		V_{CC} + 0.5	V
V_{IL3}	Input Low Voltage, RESET pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	-0.5 -0.5		0.2V_{CC}[1] 0.3V_{CC}[1]	V
V_{IH3}	Input High Voltage, RESET pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	0.7V_{CC}[2] 0.6V_{CC}[2]		V_{CC} + 0.5 V_{CC} + 0.5	V

PULL'EM ALL

Les entrées d'un microcontrôleur sont de type "haute impédance", ce qui signifie que leur résistance interne est très élevée, comme si une résistance de très forte valeur était connectée aux broches. Elles ne prélèveront donc qu'une infime quantité de courant sur le signal d'entrée et il ne sera pas nécessaire de rajouter une autre résistance en série. Il y a toutefois un effet pervers induit : lorsque l'entrée n'est connectée à rien (à un interrupteur ouvert par exemple), le bruit électrique ambiant est suffisant pour générer des valeurs aléatoires. Il est donc indispensable de "forcer" la valeur d'une entrée au repos à un état logique connu par l'intermédiaire d'une résistance dite "*pull-up*" (si elle est connectée à VCC) ou plus rarement "*pull-down*" (si elle est reliée à la masse). De cette façon, l'état au repos de l'entrée est connu et ne fluctuera pas. Une valeur de 10 kΩ est le plus souvent parfaitement adaptée. Maintenant que vous connaissez la base des résistances de rappel, j'en viens au meilleur : la plupart des microcontrôleurs (dont l'ATmega328) disposent de pull-up internes activables logiciellement.

PROTECTION++

Je vous expliquais ci-dessus que vu les caractéristiques haute impédance des entrées, il n'était pas nécessaire d'intercaler une autre résistance juste avant. Mais si vous êtes très maladroit, en installer tout de même une de 100 Ω peut être judicieux. Pourquoi ? Imaginons que vous vous trompiez dans le code du microcontrôleur, que vous configuriez l'entrée en question comme une sortie, que vous lui appliquiez l'état "1" (soit +5 V) et que vous pressiez le bouton. Dans ce cas, vous créez un court-circuit franc qui risque d'entraîner la mort prématurée de votre montage. À vous de voir si vous êtes assez maladroit pour enchainer ces trois erreurs ou pas...



Entrées analogiques

Si les entrées numériques sont parfaites pour traiter un signal binaire, il est souvent nécessaire d'acquérir une information nettement plus complexe. Pour cela, on utilise une entrée analogique, capable de convertir une tension électrique variable en une valeur numérique précise à l'aide d'un ADC (convertisseur analogique/numérique). La tension maximale mesurable est constituée par une référence fixe de grande stabilité et de grande précision que l'on applique sur l'une des broches du microcontrôleur (VRef). Évidemment, cette tension ne peut en aucun cas être supérieure à la tension d'alimentation (VCC) de la puce sous peine – une fois de plus – de tout griller. Lorsque les exigences en termes de précision ne sont pas très importantes, il est possible de configurer l'ADC interne pour utiliser la tension d'alimentation comme référence (VRef = VCC). Le nombre de "paliers" de tension, c'est-à-dire le nombre de valeurs numériques possibles, est défini par le nombre de bits de l'ADC. On parle ainsi d'ADC 8 bits, 10 bits, 12 bits et plus si affinités. Vous n'avez rien compris à ce que je viens d'écrire ? Bon. Prenons un exemple clair.

FAISONS SIMPLE...

Imaginons que vous souhaitiez mesurer la température d'une pièce pour allumer un appareil lorsqu'elle passe sous une valeur prédéterminée. Il vous faut donc un capteur capable de fournir l'information voulue (la température) au microcontrôleur sous la forme d'une tension. Imaginons que les caractéristiques dudit capteur soient de 100 mV (0.1V) par degré Celsius (100 mV/°C). Si l'air ambiant est à 25 °C, celui-ci délivrera donc une tension de 2,5 V (100 mV x 25 °C). Et à 30 °C, de 3 V. Facile ! Vous connectez ensuite la sortie de ce capteur à une entrée analogique et vous configurez en software le convertisseur A/D pour que la référence de tension (VRef) soit connectée aux 5 V de l'alimentation VCC de l'ensemble. La puce que vous avez choisie dispose d'une précision de 10 bits. Cela nous fait donc 2^10= 1 024 valeurs possibles entre la masse et VRef, soit entre 0 V et 5 V. Les "paliers" de tension mesurables seront alors d'environ 5 mV (5/1024). Si la température ambiante est de 25 °C et que la tension appliquée à l'entrée est de 2,5 V, vous récupérez une valeur brute de 1024/5 V * 2,5 V = 512. Il est d'ailleurs logique que vous récupériez une valeur à mi-chemin entre 0 et 1 024 puisque votre tension se situe exactement à mi-chemin entre 0 et 5 V. Imaginons maintenant que le capteur baigne dans un air à 30 °C. Il renverra donc une tension de 3 V qui sera lue comme 1024/5,0 * 3,0 = 614. En fait, la tension exacte se situera quelque part entre 614 et 615, soit entre 2,998 (5/1024 * 614) et 3,003 (5/1024 * 615). Une précision excellente. Par contre, si votre microcontrôleur dispose d'ADC à 8 bits (soit 2^8 = 256 possibilités), alors la précision sera un peu moindre : entre 153 (2,988 V – 29,88 °C) et 154 (3,008 V – 30,08 °C). Dans cet exemple, on comprend difficilement l'intérêt d'un ADC 10 ou 12 bits vu le faible écart au final. En pratique toutefois, il est bien présent puisque la valeur réelle des capteurs est souvent plus proche de 10 mV (voire de 1 mV) par unité que de 100 mV. Or, si l'on reprend les calculs précédents avec 10 mV/ °C, on obtient une précision de 0,5 °C maximum en 10 bits et seulement 2 °C en 8 bits. Et si le capteur offre une sortie de 1 mV/° C, on peut encore diviser la précision par 10, ce qui rend obligatoire l'utilisation d'un ADC d'au moins 12 bits. Heureusement, la plupart des microcontrôleurs récents disposent d'ADC à 10 ou 12 bits. Les précisions supérieures (16, 18 et jusqu'à 24 bits) sont accessibles grâce à des composants externes mais ceux-ci sont particulièrement complexes à exploiter. À noter enfin qu'il n'est généralement pas problématique de dépasser la tension VRef sur une entrée pour peu que l'on ne dépasse jamais VCC.

PRÉCISION++

Même si la précision de base des convertisseurs ADC s'avérera souvent de base bien suffisante pour ce que vous aurez à en faire, les références de tension sont des puces simples à utiliser qu'il peut parfois être intéressant d'utiliser. Leur stabilité est très largement supérieure à celle du +5 V ou du +3.3 V des alimentations classiques et leurs valeurs nominales sont pensées pour simplifier les calculs. Les références de tension les plus courantes utilisent par exemple 1,024 V ou 4,096 V, ce qui les rend parfaitement adaptées aux ADC de 10 et 12 bits avec des paliers de 1 mV tout rond. Toutefois, avant de considérer l'utilisation d'une telle référence, vérifiez bien que la précision du capteur est capable d'en tirer parti.

Les capteurs d'entrée

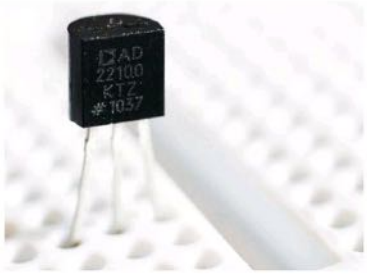
QUELQUES MODÈLES COURANTS DÉTAILLÉS

Il existe une infinité de capteurs différents pour mesurer à peu près toutes les grandeurs physiques possibles et imaginables. Le choix du modèle se fera en fonction des contraintes techniques voulues, en particulier de la précision. À titre d'exemple pratique, nous allons passer en revue quelques modèles classiques afin que vous compreniez mieux la façon de les utiliser. Tous ces capteurs utilisent une entrée analogique et s'interfacent donc très facilement. Ils ne requièrent généralement que l'alimentation électrique, rarement un ou deux composants annexes supplémentaires. Beaucoup de capteurs sont disponibles soit "nus" sur les sites de vente de composants, soit sous la forme d'une minuscule carte d'interface "toute prête"(*breadboard*) sur les sites comme Sparkfun ou Adafruit.

CAPTEUR DE TEMPÉRATURE

Analog Devices TMP36

Le TMP36 est un capteur de température très répandu, fiable, précis, peu coûteux (1,50 euro) et extrêmement simple à mettre en œuvre. Il se présente sous la forme d'un petit boîtier TO-92 à 3 broches. Fournissez-lui une alimentation comprise entre 2,7 et 5,5 V sur les deux broches situées aux extrémités – gare à la polarité ! – et il délivrera automatiquement une tension proportionnelle à la température ambiante sur la broche du milieu. Une fois reliée à une entrée analogique, vous pourrez alors lire très facilement la valeur reçue. Un petit calcul mathématique s'imposera toutefois dans votre programme. Le TMP36 offre une plage de mesure de -40 °C à +125 °C et une tension à 0 °C de 500 mV. Sa sensibilité est de 10 mV /°C. À une température de 25 °C, vous obtiendrez donc 750 mV sur la sortie. Lue avec un microcontrôleur disposant d'un ADC 10 bits et d'une tension de référence de 5 V, on obtiendra une valeur brute de 1 024 / 5 * 0,75 = ~154. Pour convertir cette valeur en température, il suffit d'effectuer l'opération inverse : Valeur_lue * VRef/1024 = 0,75. On soustrait ensuite le décalage au zéro (500 mV) et il ne reste plus qu'à diviser le tout par la sensibilité (10 mV/° C) pour obtenir des degrés. On obtient donc : ((154 * 5/1 024) – 0.5) / 0,01 = 25 °C. Si on lit 214 en donnée brute, le capteur mesure une température de 54,5 °C. Facile, non ?



CAPTEUR D'HUMIDITÉ

Honeywell HIH-4000

Prenons un autre exemple, en l'occurrence un capteur d'humanité très connu, le HIH-4000 d'Honeywell. Celui-ci se comporte exactement comme le capteur de température dans les grandes lignes malgré un coût beaucoup plus élevé (15 euros). Il dispose de deux broches d'alimentation et d'une troisième destinée à transmettre le signal à une entrée analogique. Son datasheet (*cpc.cx/9aE*) nous apprend qu'il doit être alimenté entre 4 et 5,8 V et que sa sortie répond à une équation mathématique nettement plus compliquée : à 25 °C, V_Out =VCC*((0,0062*humidité) +0,16). Avec VCC = 5 V, la tension mesurée à 10 % d'humidité sera donc 5*((0,0062*10)+0,16) = 1,11 V et lue comme "227" par notre ADC 10 bits. À 80 % d'humidité, on obtiendra 3,28 V (672) dans les mêmes conditions. En reprenant l'équation à l'envers dans le programme du microcontrôleur, on peut lire aisément l'humidité ambiante.

CAPTEUR D'ACCÉLÉRATION

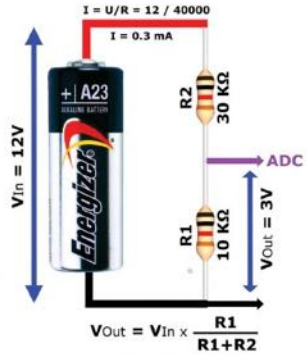
Analog Devices ADXL335

L'ADXL335 est un composant de type LFCSP ultra-miniature qu'il est impossible d'utiliser tel quel pour le commun des mortels. Il faudra donc se le procurer sous la forme d'une *breadboard* où il sera prémonté pour une vingtaine d'euros. Il s'agit d'un accéléromètre à trois dimensions comme ceux qui équipent tous les appareils mobiles de nos jours. Son fonctionnement de base est ici encore identique à celui des précédents (sortie en tension à connecter sur une entrée analogique) mais ses caractéristiques électriques diffèrent : il fonctionne avec une tension d'alimentation de 3,6 V maximum et dispose de trois sorties, une par dimension (X, Y et Z). Heureusement, la plupart des microcontrôleurs sont équipés d'une bonne dizaine d'entrées analogiques, ce qui ne posera donc aucun problème. Le reste des spécifications permet un fonctionnement très simple : le point d'inertie ("0 g") se situe à 1,5 V et le capteur a une sensibilité de 300 mV/g. Il peut donc mesurer de -3 g à +3 g, soit des valeurs de 0,6 V à 2,4 V. Je vous fais grâce des étapes de conversion.

CAPTEUR DE PROXIMITÉ

Maxbotix EZx

Juste pour le plaisir et pour vous montrer l'étendue des possibilités, je vous présente également ce capteur de proximité par ultrason (environ 25 euros). Il s'alimente avec une tension comprise entre 2,5 et 5 V, consomme très peu d'énergie et permet de mesurer la distance qui le sépare d'un objet avec une grande précision : environ 2,5 cm jusqu'à 6 mètres ! Sa sortie est extrêmement simple à exploiter puisqu'elle est étalonnée à 10 mV/pouce (2,54 cm). Comme beaucoup de capteurs évolués, la série EZ de Maxbotix dispose également d'une sortie sous la forme d'un bus de communication RS232 (voir encadré) mais l'utilisation d'une simple entrée analogique permet de ne pas surcharger le microcontrôleur en dialogues inutiles.



CAPTEUR DE TENSION

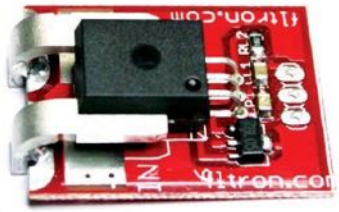
Pont diviseur

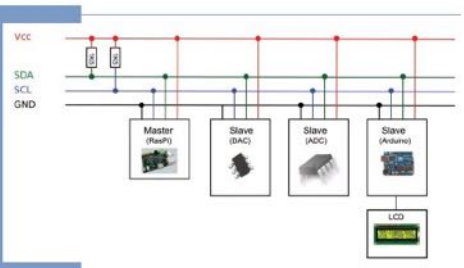
À première vue, mesurer une tension brute avec une entrée analogique pour s'en servir de voltmètre est un jeu d'enfants puisqu'il suffit d'y connecter le signal directement. Certes. Mais il faut tout de même prendre certaines précautions. Tout d'abord, bien vérifier que la masse est commune entre la source à mesurer et celle du microcontrôleur. Sans quoi des boucles de masse apparaîtront et ruineront l'ensemble. Ensuite, il faut évidemment veiller à ce que la tension appliquée au microcontrôleur ne dépasse pas VCC, soit généralement 5 V. Comment faire pour mesurer une tension supérieure malgré tout, par exemple une batterie 12 V ? Simple ! Le pont diviseur est là pour ça. Il consiste à faire transiter une fraction du courant à mesurer par deux résistances (R1 et R2) puis de connecter l'entrée analogique au point milieu (voir schéma). La relation entre la tension d'entrée (VIN) et celle qui sera mesurée en sortie (VOUT) par l'entrée analogique s'écrit ainsi : VOUT = VIN * (R1 /(R1+R2)). Si vous souhaitez mesurer une valeur d'environ 12 V avec une marge de sécurité suffisante sur un microcontrôleur alimenté en 5 V, il faudra donc utiliser un pont diviseur par 4. Une résistance R1 de 10 kΩ et R2 de 30 kΩ feront parfaitement l'affaire. Avec une tension de 12 V appliquée à l'entrée du pont diviseur, on obtiendra 12*(10/(30+10)) = 12*0,25 = 3 V en sortie. Une valeur parfaitement compatible avec l'entrée analogique.

CAPTEUR DE COURANT

Shunts

Le courant électrique (pour la charge ou décharge d'une batterie par exemple) est l'une des grandeurs physiques les plus compliquées à mesurer correctement. Il n'existe pas beaucoup de possibilités : soit un capteur à effet Hall qui souffre de nombreux désavantages (cher, peu précis…) soit une bonne vieille résistance de très faible valeur. Au passage d'un courant, une tension va apparaître logique à ses bornes selon la loi d'Ohm, U = R*I. Faites passer un courant de 7 ampères dans une résistance de 0,1 Ω et vous obtiendrez une tension de 0,7 V *à ses bornes*. Facile ? En fait pas du tout car il convient de surmonter deux problèmes majeurs. Tout d'abord, cette tension est à ses bornes, c'est-à-dire qu'elle n'est pas référencée à la masse et que pour la mesurer avec un ADC, il faut la référencer de nouveau à la masse en utilisant un AOP monté en comparateur. Ensuite, l'échauffement de la résistance sera rapide et très important : la loi de Joule (voir page 11) nous dit que P=U*I donc notre résistance devra dissiper 0,7 * 7 = 4,9 watts ! Bonne nouvelle : il existe des solutions toutes faites avec AOP intégré pour moins de 10 euros. Si vous devez mesurez un courant, utilisez-les sans hésiter.





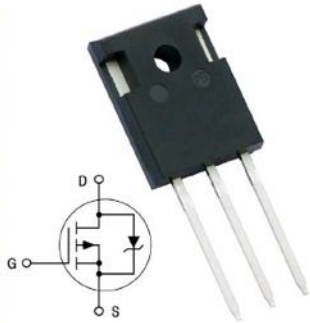
L'heure du bus

Outre les entrées numériques et analogiques que nous venons de décrire, il existe une autre méthode d'échange de données entre un capteur et un microcontrôleur : le bus de communication. On trouve par exemple les protocoles 1-Wire, I2C ou SPI pour les plus courants, qui ne sont pas sans rappeler le bon vieux RS232 des antiques ports série. Ces bus de communication sont utilisés pour transmettre rapidement un grand flux de données et souvent capables de gérer plusieurs dizaines de périphériques simultanément. Leur mise en œuvre s'annonce toutefois nettement plus complexe et nous en parlerons éventuellement plus tard dans une partie "avancée", une fois que l'initiation sera terminée.

Les sorties

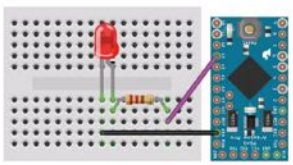
OU COMMENT COMMANDER AUX ÉLÉMENTS…

Capter des informations, c'est bien, mais pour qu'un montage ait un sens, il faut également qu'il puisse influer physiquement sur des éléments extérieurs. Un microcontrôleur est un minuscule composant qui ne dispose que de très faibles capacités en termes de commande. Heureusement, grâce aux interfaces adaptées dont nous allons vous présenter le principe ici, il est possible de contrôler un voyant, un écran LCD, une ampoule, un moteur ou même un tractopelle à l'aide d'une simple sortie numérique. Si.

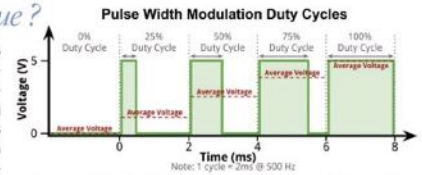


Connexion directe

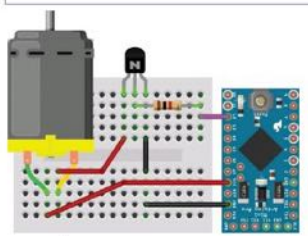
Il n'existe que très peu de composants qui peuvent être contrôlés directement par la sortie d'un microcontrôleur sans autre forme de procès. Le courant disponible ne dépasse pas les quelques dizaines de milliampères et toute surcharge entraîne la destruction de la sortie, voire de toute la puce. Les modèles d'ATMEL ou de Microchip (entre autres) disposent de sorties généralement spécifiées à 40 mA, mais nous vous déconseillons fortement de les charger à plus de 20 mA. L'ensemble de la puce est également limité : même si chaque sortie peut fournir jusqu'à 40 mA, l'ensemble des sorties ne peut débiter plus de 200 mA et c'est déjà beaucoup. Le seul composant que l'on peut raisonnablement connecter directement reste une vulgaire LED qui fera office de voyant ; sa consommation se situe aux alentours de 20 mA. Évidemment, comme vous avez scrupuleusement suivi avec attention les parties précédentes de cette initiation, vous savez qu'il est impératif de limiter le courant qui circulera dans la LED (et accessoirement dans la sortie du microcontrôleur) à l'aide d'une résistance. Pour calculer sa valeur, vous devez connaître la tension disponible sur la sortie à l'état "1" (VCC), la tension Vf de la diode (généralement 1,9 V pour une LED rouge) et son courant maximum (If, disons 20 mA). Toutes ces données sont disponibles dans le datasheet. Comme R = U / I, la valeur de la résistance R se calcule ainsi : R = (VCC-Vf) / If = 155 Ω. Comme vous êtes quelqu'un de raisonnable, vous choisirez une résistance de 180 ou 200 Ω pour limiter le courant qui circulera dans la LED.



Sortie analogique ?



Vous noterez que nous ne parlons ici que de sorties numériques. Théoriquement, il existe aussi des sorties analogiques obtenues à l'aide d'un DAC (convertisseur numérique/analogique). Leur usage est très limité en pratique et il est rarissime qu'un microcontrôleur dispose de telles sorties. Les modèles récents embarquent toutefois un ersatz de sorties analogiques nommée "PWM" (*pulse width modulation*) : il s'agit en fait de faire varier très rapidement (et automatiquement) une sortie numérique de 0 à 1 et inversement à une fréquence variable. À l'aide de quelques composants externes (condensateurs, résistances), on peut alors simuler grossièrement une sortie analogique. Reste que concrètement, les circuits qui nécessitent un DAC sont très peu nombreux. Les sorties PWM utilisées nativement permettent cependant quelques montages intéressants, par exemple pour faire varier la vitesse d'un ventilateur.

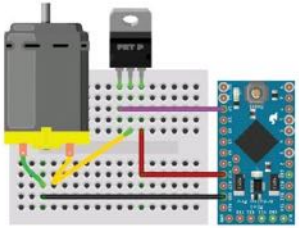


Transistor

Le transistor est le composant le plus courant pour permettre à une sortie numérique de commander un courant beaucoup plus important. Nous l'utiliserons ici en régime de commutation : en saturant sa *base* avec quelques mA, les broches collecteur et *émetteur* se mettent à conduire le courant, jusqu'à plusieurs ampères selon le modèle. On peut donc utiliser le transistor comme un interrupteur électrique dont l'état (*On* ou *Off*) dépend de celui de la sortie numérique. La base du transistor se commande par un courant qui circule librement entre la base et l'émetteur. Comme pour la LED, il est donc indispensable d'utiliser une résistance qui va limiter ce courant. Problème : le calcul du courant (et donc de la valeur de la résistance) nécessaire pour saturer complètement la base d'un transistor et ainsi le rendre complètement passant est assez compliqué à réaliser. Non seulement il dépend de multiples facteurs fixes liés à la construction, mais l'intensité du courant à commander rentre aussi en ligne de compte. Je vous offre donc un protip : pour les transistors NPN les plus courants de type 2N2222, utilisez une résistance de 1 kΩ ; elle fera l'affaire dans la plupart des cas et ne nécessitera que 5 mA sur la sortie numérique (5/1000). Toutefois, j'ai encore mieux à vous proposer : oubliez le transistor bipolaire de grand papa et passez au MOSFET !

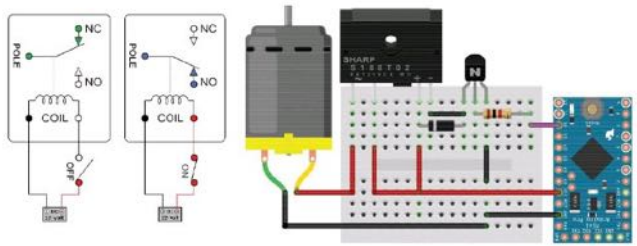
MOSFET

Le MOSFET (*metal-oxide-semiconductor field-effect transistor*) a tout d'un transistor classique et coûte à peine quelques centimes plus cher. Il dispose toutefois d'un immense avantage : sa base (*gate* - grille) n'est plus commandée avec un courant mais avec une tension. En clair, vous n'avez pas à calculer ni à vous soucier du courant (extrêmement faible) qui va circuler par sa grille. Pour peu que vous lui appliquiez une tension suffisante, le MOSFET basculera à l'état passant. Cette tension est notée dans le datasheet sous l'appellation VGS (*gate-source voltage*) et se situe le plus souvent aux alentours de 2-4 V maximum. On ne peut donc pas faire plus simple : connectez directement la grille du MOSFET à la sortie numérique et vous pourrez commander une forte charge très facilement. Évidemment, certains esprits chagrins vous recommanderont d'intercaler tout de même une résistance, pour protéger le microcontrôleur au cas où le MOSFET viendrait à défaillir ou bien pour éviter une surcharge temporaire à la commutation due à la présence d'une capacitance parasite au niveau de la grille. Baste ! Selon nous, elle n'est nécessaire qu'aux plus paranoïaques ou pour certains types bien particuliers de MOSFET. Si vous le souhaitez vraiment, une valeur de 1 kΩ fera l'affaire.

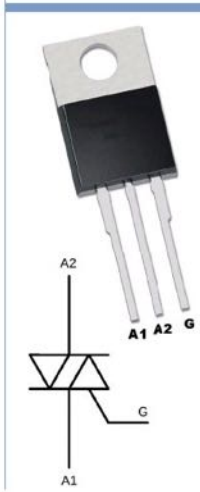


Relais

Certaines charges électriques ne peuvent être commandées par des transistors classiques ou des MOSFET. Soit parce qu'elles utilisent un courant alternatif, soit parce qu'elles ne partagent pas une masse commune avec le microcontrôleur. Pour commander l'allumage d'une ampoule électrique connecté au secteur EDF par exemple, il est impossible d'utiliser les techniques décrites précédemment. Un bon vieux relais – qui garantit une isolation totale avec le reste du circuit – est alors une solution idéale. Il s'agit en fait d'un interrupteur mécanique commandé par un électroaimant. Celui-ci se commande grâce àa un courant électrique relativement faible. "Relativement" car il dépasse souvent les 100 mA et il est donc impossible de le relier directement à une sortie numérique. Pour utiliser un relais, il faudra donc impérativement le commander par l'intermédiaire d'un transistor (ou mieux, d'un MOSFET). L'électroaimant commandera alors à son tour l'activation du relais. À noter – sans paranoïa cette fois – qu'il est souvent indispensable de connecter une diode aux bornes de l'électroaimant pour éviter que des surtensions parasites ne viennent endommager le microcontrôleur. Certains relais embarquent toutefois une telle diode en standard.



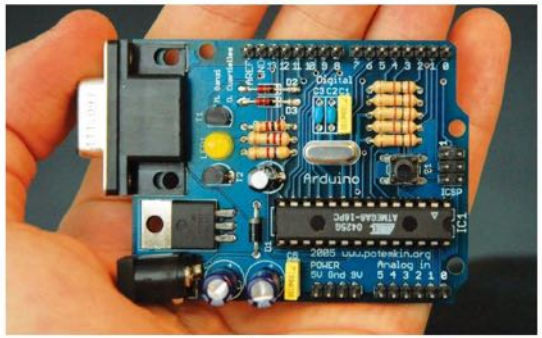
Triac



Le triac est la version électronique du relais. Il est beaucoup plus rapide et permet lui aussi de commander une charge alternative. C'est le composant idéal pour faire un variateur de lumière par exemple. En commutant très rapidement des cycles d'allumage/extinction, il peut faire varier l'intensité lumineuse très efficacement. Vu qu'il est généralement prévu pour être connecté au secteur EDF, il convient impérativement de respecter des règles de sécurité fondamentales. Tout d'abord, contrairement au relais, il ne fournit pas l'isolation électrique obligatoire entre la haute tension et le microcontrôleur. Il faut donc le commander avec un optocoupleur, qui "casse" la liaison électrique et isole les deux parties en transférant l'information On/Off grâce à un flux lumineux. Il se gère en fait comme une LED (il en intègre une) côté commande et comme un transistor côté sortie. Je ne vous en dirai pas plus ici : en tant que débutant, il n'est pas raisonnable de jouer avec un triac et avec de la haute tension. Si vous devez contrôler un appareil haute tension, familiarisez-vous d'abord avec les relais.

Les Arduino

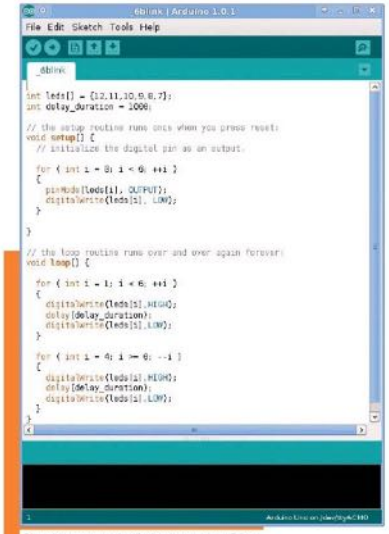
Qui aurait imaginé qu'une vulgaire carte *open source* conçue par des Italiens et nommée en hommage au bistrot du coin allait révolutionner l'électronique amateur ? C'est pourtant bien la prouesse réalisée en quelques années par les cartes Arduino. Plus que de leur matériel, elles doivent surtout leur immense succès à leur écosystème logiciel accessible à tous.



Une des premières cartes Arduino.

Un Arduino basique se compose principalement d'un microcontrôleur 8-bit de la famille AVR (Atmel), de broches d'entrée/sortie et d'un environnement de développement. Le choix de la puce s'explique d'abord par son coût, bien plus faible que celui d'un CPU généraliste. Il offre aussi l'avantage de ne pas nécessiter d'OS pour fonctionner : dans un Arduino, le microcontrôleur exécute le programme conçu par l'utilisateur… et rien d'autre. La carte Arduino Uno originelle – toujours, de loin, la plus répandue aujourd'hui – se base sur l'ATmega328, une puce capable d'atteindre 20 MHz qui dispose de 2 Ko de RAM et de 32 Ko de mémoire flash pour stocker le programme. Ces valeurs peuvent sembler ridicules à l'époque des Core i7 à 4 GHz et des SSD de 4 To, mais restent suffisantes pour de l'électronique embarquée destinée à effectuer des tâches simples en temps réel. Le vrai secret de l'Arduino, celui qui a fait son succès face aux innombrables plateformes de développement qui existaient auparavant, c'est son environnement de programmation très simple. Tous les microcontrôleurs des cartes Arduino sont préprogrammés avec un *bootloader* qui permet à l'utilisateur de télécharger du code directement grâce à une connexion USB, sans passer par un programmateur externe de composant. Le développement s'effectue grâce à un IDE (*integrated development environment*) qui combine un éditeur de texte, le compilateur C et l'utilitaire d'upload vers la carte. De très nombreuses librairies intégrées permettent de simplifier largement l'utilisation de fonctionnalités avancées comme les afficheurs LCD ou les moteurs.

Enfin, l'IDE Arduino est disponible sur Windows, Linux et macOS.



L'environnement de développement sous Linux.

OPEN SOURCE

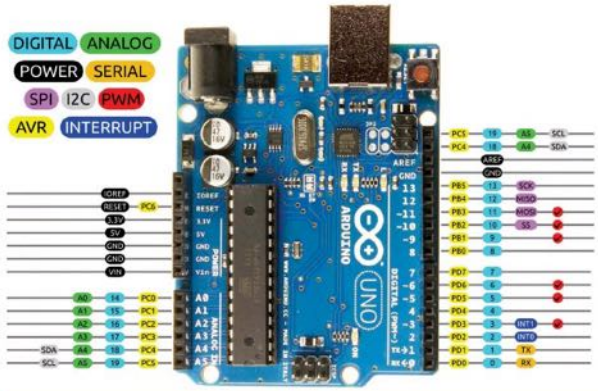
Côté hardware, l'énorme succès de l'Arduino vient également de sa licence "libre". Le design du matériel est en effet disponible sous licence *Creative Commons*, alors que la partie logicielle est distribuée sous licence LGPL. Des clones d'Arduino existent donc, ce qui a permis l'émergence de nombreuses déclinaisons compatibles avec l'écosystème mais dotées de fonctionnalités spécifiques. Si les schémas de la carte sont libres, ce n'est pas le cas du microcontrôleur d'Atmel, mais le *bootloader* et le compilateur peuvent facilement être adaptés à d'autres architectures (les Cortex d'ARM et même le x86 d'Intel par exemple). Seul le nom "Arduino" est une marque déposée dans certains pays (voir encadré ci-contre). Les créateurs préfèrent le restreindre aux produits *officiels*, en laissant les clones indiquer "*compatible avec Arduino*" sur leurs périphériques.

GENUINO VS ARDUINO

Suite à une sombre histoire de brevet provoquée en 2015 par la gourmandise d'un des cinq cofondateurs originaux du projet, la marque "Arduino" s'est retrouvée scindée en deux. D'un côté, le "faux" Arduino (Arduino SRL – arduino.org) qui s'est approprié arbitrairement la marque en Europe. De l'autre, le "vrai" Arduino (Arduino LLC – arduino.cc) qui a été forcé de créer une autre marque (Genuino) pour continuer à vendre ses produits dans l'UE. La communauté est toutefois restée largement en faveur d'Arduino LLC, gérée par 4 des 5 cofondateurs originaux. Le 1er octobre dernier, les deux sociétés ont annoncé avoir mis fin à leur guérilla juridique et comptent désormais se réunifier d'ici la fin de l'année sous le nom de "Arduino Holding".

Arduino Uno (20 €)

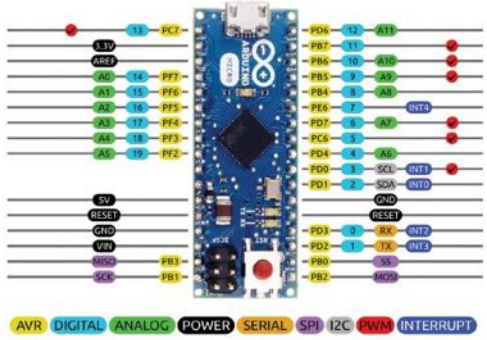
LA RÉFÉRENCE



L'Arduino Uno (Genuino dans certains cas) reste de loin le modèle le plus populaire. Il intègre un ATmega328P à 16 MHz (2 Ko de RAM, 32 Ko de flash, 1 Ko d'EEPROM), une interface USB pour la programmation ainsi que 20 entrées/sorties. On y trouve 6 entrées analogiques capables de convertir une tension sur 10 bits de résolution (soit 1 024 valeurs différentes), 14 entrées/sorties numériques (dont 6 peuvent fonctionner en mode PWM pour générer des impulsions) et des bus de communication SPI, I2C et série. Les entrées analogiques peuvent être configurées comme des entrées/sorties numériques si besoin et une LED est présente par défaut sur la sortie 13. La carte s'alimente soit en USB, soit grâce à une source externe (de 7 à 12 V). Un régulateur de tension intégré convertit alors cette tension en 5 V. Dans tous les cas, il convient de ne jamais appliquer sur ses broches une tension supérieure à 5 V, sous peine de destruction immédiate. Et attention : certaines déclinaisons comme l'Arduino Zero ou l'Arduino MKR1000 se limitent même à 3,3 V. Enfin, le courant disponible sur chaque broche est limité à 20 mA (grand) maximum.

Arduino Micro (25 €)

L'Arduino Micro dispose de fonctionnalités quasi identiques à l'Arduino Uno mais dans un format bien plus compact (48 x 18 mm). Il embarque un microcontrôleur très proche de l'ATmega328P (l'ATmega32U4) qui présente l'avantage d'intégrer directement la gestion de l'USB. Il devient donc possible de se passer de la puce dédiée présente sur le Uno et d'effectuer des opérations plus complexes en USB (comme l'émulation d'un clavier ou d'une souris). Autre différence : l'Arduino Micro supporte jusqu'à 12 entrées analogiques et embarque un connecteur MicroUSB. Il fonctionne toujours en 5 V.



Arduino Mega (35 €)

Ce modèle plus haut de gamme vise ceux qui en veulent plus. Plus de broches : 54 entrées/sorties numériques (dont 15 avec PWM) et 16 entrées analogiques. Plus de mémoire : 8 ko de RAM, 256 ko de flash, 4 ko d'EEPROM. Mais aussi plus d'encombrement (100 x 54 mm) et pour plus cher (environ 35 €). La carte intègre un ATmega2560 à 16 MHz ainsi que quatre UART (des ports série) pour communiquer avec d'autres périphériques. Le modèle le plus intéressant pour ceux qui ont de gros besoins.

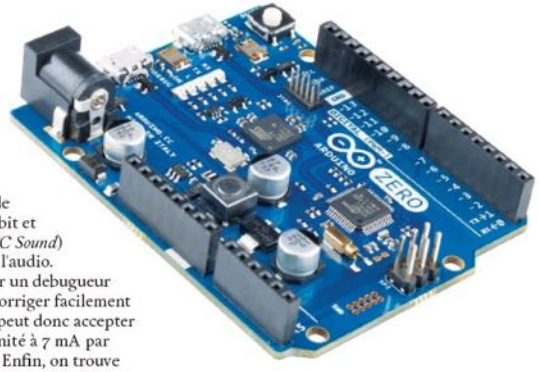
Les Arduino alternatifs

ORIGINAUX ET COMPATIBLES

Open source, la plateforme Arduino a rapidement été clonée en dizaines de cartes compatibles qui présentent plus ou moins d'intérêt. Nous avons sélectionné ici quelques déclinaisons que nous avons particulièrement appréciées. Certaines proviennent directement d'Arduino LLC, d'autres sont conçues et assemblées par des tierces parties.

Arduino Zero (40 €)

SAUCE ARM



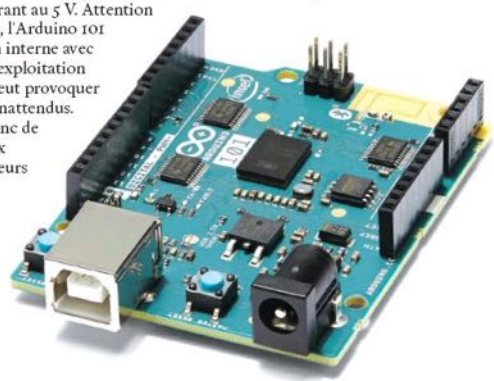
L'Arduino Zero ressemble physiquement à un Uno, mais son microcontrôleur n'en demeure pas moins très différent. Alors que la majorité des Arduino exploitent un modèle AVR 8-bit, le Zero intègre un cœur ARM 32-bit Cortex M0+ à 48 MHz (un Atmel SAMD21) bien plus puissant. On y trouve aussi bien plus de mémoire (32 ko de RAM, 256 ko de flash), une horloge RTC intégrée, des entrées analogiques sur 12-bit et même une sortie analogique sur 10-bit. Une interface I2S (*Inter IC Sound*) est également présente afin de supporter des accessoires dédiés à l'audio. Arduino a aussi travaillé en partenariat avec Atmel pour proposer un debugueur interne au microcontrôleur, ce qui permet aux développeurs de corriger facilement leur code. En pratique, l'Arduino Zero fonctionne en 3,3 V et ne peut donc accepter aucune tension supérieure sur ses entrées. Le courant est aussi limité à 7 mA par broche. Impossible donc de commander une LED sans transistor. Enfin, on trouve également l'Arduino Zero sous le nom de "Arduino M0 Pro" (produit par Arduino SRL).

Arduino MKR1000 (35 €)

L'Arduino MKR1000 est destiné à des projets IoT (Internet des objets). Très compact, il intègre un SoC ATSAMW25 composé d'un microcontrôleur 32-bit ARM Cortex-M0 identique à celui de l'Arduino Zero et d'un contrôleur Wi-Fi basse consommation 802.11 b/g/n doté d'une antenne. La plupart des fonctionnalités présentes sur le Zero en termes d'entrées/sorties existent également sur le MKR1000. On y trouve en plus un connecteur pour une petite batterie Li-Po et un circuit de recharge, le tout dans un format de 62 x 25 mm.

Arduino 101 (40 €)

La particularité de l'Arduino 101 vient de son microcontrôleur. Cette fois, il ne s'agit ni d'un ARM ni d'un AVR, mais d'un SoC Intel Curie. Celui-ci intègre deux processeurs à 32 MHz : un Quark SE basé sur l'architecture x86 et un ARC RISC. La carte intègre aussi des fonctions inédites : un contrôleur Bluetooth LE et un accéléromètre. L'Arduino 101, comme le Zero et le MKR1000, travaille par défaut avec une tension de 3,3 V mais reste en revanche tolérant au 5 V. Attention tout de même, l'Arduino 101 fonctionne en interne avec un système d'exploitation (RTOS) qui peut provoquer des résultats inattendus. Il convient donc de le réserver aux expérimentateurs avisés.



GARE AUX CLONES ULTRA CHEAP !

Vous cherchez depuis des heures d'où peut bien provenir le comportement incohérent de votre programme ? Posez-vous la question : avez-vous acheté un clone d'Arduino Uno à 4€ (frais de ports compris) chez Aliexpress ? Si oui, ne cherchez plus. S'il existe bien de nombreuses copies conformes de très bonne qualité, on trouve également sur le marché d'infâmes ersatz mal soudés qui provoquent des faux-contacts à répétition. Nous considérons qu'en deçà de 7€, soit trois fois moins que le tarif d'un "vrai" Arduino Uno, la fiabilité risque de souffrir. Ne sombrez pas dans les économies de bouts de chandelle !

ESP8266 (3 €)

L'ARDUINO WI-FI À BAS PRIX

Les cartes à base de SoC ESP8266 (qui intègre un microcontrôleur et un chip Wi-Fi) rencontrent un grand succès sur eBay. Et pour cause : elles offrent le Wi-Fi et une compatibilité Arduino pour moins de 3 €, frais de port compris ! Les ESP8266 peuvent s'utiliser soit comme contrôleur Wi-Fi externe pour un Arduino (avec de simples commandes AT), soit exécuter directement des programmes à l'aide d'un firmware adapté. La version de base dispose de seulement deux GPIO, mais il existe des déclinaisons plus évoluées (comme l'ESP-012) qui en offrent 12, ainsi qu'une entrée analogique. L'ESP32, plus moderne et plus rapide, intègre en plus du Bluetooth LE un processeur dual-core à 160 MHz. La programmation des ESP demande un peu de travail (elle peut se faire à travers un Arduino) mais le prix très faible des cartes et le format compact permettent une intégration simple pour commander des appareils à distance en Wi-Fi. Le jouet parfait pour les bidouilleurs fauchés.

Teensy (25 €)

Les cartes de la famille Teensy offrent une compatibilité Arduino complète dans un format ultra-compact (encore plus que l'Arduino Micro). Différentes versions coexistent, mais la plus aboutie – Teensy 3.2 – embarque un microcontrôleur 32-bit ARM Cortex M4 à 72 MHz dont les performances surpassent largement tous les Arduino de taille comparable. On y trouve également un nombre pharamineux d'entrées/sorties (plus de 30 !) accessibles soit sur des broches, soit directement sur le PCB. Elles sont de plus tolérantes au 5 V. Bref, un Arduino Micro boosté aux hormones.

Adafruit Flora (20 €)

La liste des clones et autres cartes compatibles avec les Arduino pourrait être encore bien plus longue, mais nous avons choisi le Flora d'Adafruit. Ce module très compact et rond est prévu pour être incorporé dans des vêtements ; les Ricains parleront de "*wearable*". Le Flora dispose d'une LED RGB, d'une prise USB pour la programmation et de quelques entrées/sorties. Un bon exemple d'intégration et d'adaptation à un usage précis tout en gardant la compatibilité et la simplicité d'utilisation des Arduino. Pour faire clignoter votre slip, il n'y a pas mieux !

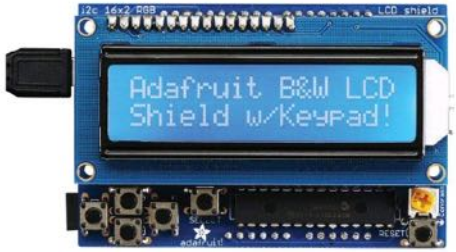
Les Shield Arduino

LE PARADIS DES FEIGNASSES

La plateforme Arduino offre une option intéressante : les Shield. Il s'agit de petites cartes qui se placent sur les connecteurs d'un Arduino et ajoutent des fonctionnalités, en répliquant ensuite les broches pour un éventuel autre Shield. Ces cartes d'extension modulaires peuvent intégrer une prise réseau, un module GPS, un contrôleur RFID, etc.

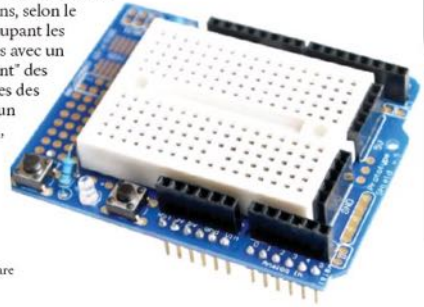
Shield Afficheur (20 €)

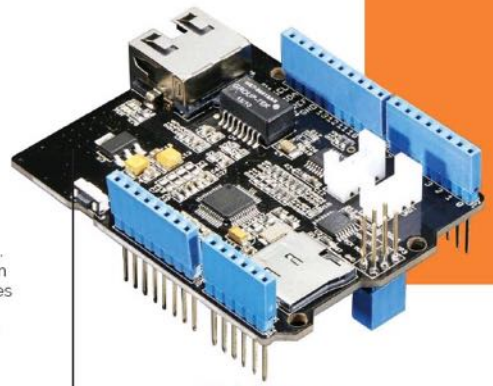
Il existe énormément de Shield qui ajoutent une fonction pratique à un Arduino : un afficheur. Le choix est très large, en fonction de votre budget et de vos envies. Vous trouverez des écrans LCD monochromes, des modèles OLED très fins, des dalles eINK, etc. Certains affichent uniquement des caractères alors que d'autres proposent des modes graphiques avancés. La seule limite demeure votre imagination et bien évidemment la mémoire des Arduino. Attention : certains Shield consomment plus que d'autres, ce qui peut poser des soucis au moment de l'intégration.



ProtoShield (10 €)

Une fois vos montages testés sur une *breadboard*, vous chercherez sûrement un moyen de les pérenniser "en dur". C'est justement le rôle des Shield vierges de prototypage. Ils disposent généralement d'une grille de trous permettant de relier ensemble vos composants. Les connexions peuvent s'effectuer de différentes façons, selon le modèle : soit en coupant les pistes préexistantes avec un cutter, soit en "tirant" des fils entre les broches des composants. Avec un peu d'organisation, on peut parvenir à un résultat correct et fiable.





Shield Ethernet (30 €)

Un grand classique dans les Shield : l'Ethernet. Un module avec une prise RJ45 permet de connecter facilement un Arduino à un réseau local (ou même à Internet) et cette technologie éprouvée a l'avantage d'être stable et efficace. Une bonne partie des variantes modernes ajoute généralement aussi un emplacement pour une carte microSD. La configuration ne pose pas de souci tant que vous restez en IPv4. N'espérez pas du Gigabit Ethernet ni même des débits décents : les puces utilisent généralement un bus série pour la connexion à l'Arduino et se destinent à l'envoi de quelques données.

Shield GSM (35 €)

Pour communiquer avec un Arduino éloigné, le réseau cellulaire GSM des téléphones portables s'impose vite. Les Shield de ce type intègrent généralement une puce compatible avec la 2G et exigent évidemment une carte SIM. Ils permettent d'envoyer et recevoir des SMS, de gérer des appels vocaux avec des prises jack dédiées et de communiquer vers l'extérieur en IP. Attention, ils se limitent souvent à la norme GPRS (~56 kbits/s) et visent donc en priorité les applications qui envoient peu de données.

Shield Wi-Fi (50 €)

Si l'Ethernet reste stable et simple, le Wi-Fi a l'avantage d'éviter les câbles. Les dernières générations de Shield Wi-Fi supportent le 802.11n dans la bande des 2,4 GHz et les solutions de chiffrement classiques (WPA2), avec une configuration qui consiste simplement à fournir le SSID et le mot de passe dans le programme. Pour cacher un Arduino au fond d'une armoire sans accès direct à une prise RJ45, le Wi-Fi demeure la solution la plus pratique actuellement. De plus, les puces modernes consomment peu et peuvent même s'activer uniquement à la demande. Si l'encombrement du Shield représente un problème, tournez-vous vers les cartes avec Wi-Fi intégré.

Shield RFID (40 €)

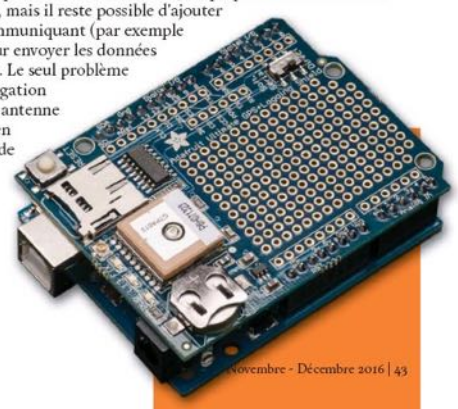
Les puces RFID deviennent très courantes dans la vie de tous les jours. On en trouve du NFC des smartphones aux cartes de paiement dans les distributeurs, en passant par les passeports ou les "clés" ultra-sécurisées – ou pas – des bâtiments. Un Shield compatible RFID permet de lire le contenu d'une puce et même de le modifier dans certains cas. Avec des cartes vierges, il devient possible d'imaginer énormément d'applications. Nous vous conseillons les modèles à base de PN532, un contrôleur classique et éprouvé, facilement programmable. Certains modèles peuvent même fonctionner sans Arduino, directement connecté à un PC sous Linux.

Shield Bluetooth (20 €)

Le Bluetooth va souvent trouver sa place dans les Shield, mais avec deux usages bien différents. Une partie des modèles vise surtout à permettre une communication sans fil entre l'Arduino et un smartphone ou un ordinateur, en émulant une connexion série à travers un lien Bluetooth (un des profils de base). Ce lien autorise la programmation d'un Arduino à distance ou – tout simplement – la récupération de données facilement. Les autres modèles, à la norme Bluetooth LE, peuvent remplacer un *beacon* et émettre en permanence des informations vers un smartphone ou une tablette, pour faire réagir une application. Les *beacon* s'utilisent par exemple en tant que capteurs de température.

Shield GPS (45 €)

Pour suivre un Arduino à la trace, quoi de mieux qu'un GPS ? Des Shield intégrant un récepteur GPS (parfois aussi compatibles avec GLONASS) existent, pour récupérer les coordonnées géographiques de l'Arduino. Les modèles évolués offrent un emplacement pour une carte microSD, ce qui permet de stocker les informations, mais il reste possible d'ajouter un Shield communiquant (par exemple un GSM) pour envoyer les données en temps réel. Le seul problème vient de l'obligation d'utiliser une antenne externe et bien évidemment de l'absence de localisation rapide (type aGPS).



Les Raspberry Pi

Lancée en 2012, la carte Raspberry Pi n'est pas *stricto sensu* un produit dédié à l'électronique. Pourtant, le prix et les nombreuses broches GPIO la rendent parfaitement utilisable pour de nombreux montages. Sans compter que la présence d'un CPU relativement puissant – surtout dans les dernières déclinaisons – offre de l'intérêt indéniable pour certains projets.

La première version de la carte, en 2012.

Le Raspberry Pi a dépassé les dix millions d'exemplaires vendus récemment. Au fil du temps, la fondation à l'origine de la carte a proposé diverses évolutions, mais le concept de base n'a pas bougé : il s'agit d'un outil de développement qui vise à l'origine le monde de l'éducation (bien qu'il ait largement trouvé sa place chez les bidouilleurs de tout poil).

Électronique et Linux. La carte combine deux choses qui intéressent beaucoup les geeks et ceux qui aiment mettre les mains dans le cambouis : un système d'exploitation libre (souvent une distribution Linux) et des broches GPIO programmables. Les différents modèles se basent sur une plateforme ARM avec des CPU offrant nettement plus de fonctions et de puissance que les microcontrôleurs présents dans les Arduino. Les Raspberry Pi utilisent des cores généralistes, capables de faire tourner n'importe quel OS compatible avec les puces d'ARM (ARMv6 à ARMv8, selon la génération). Au niveau des OS, Linux reste la solution la plus courante, notamment avec la distribution dédiée aux Raspberry Pi (Raspbian), mais il en existe d'autres comme RISC OS, NetBSD ou même Windows 10 (en version IoT, dédiée aux objets connectés). Si certains préfèrent les montages à base d'Arduino pour l'électronique grâce à la facilité de programmation, les Raspberry Pi offrent pourtant de l'intérêt dans certains cas. Premièrement, le Raspberry Pi bénéficie de la possibilité d'utiliser des langages standard, comme le Python, alors que les Arduino se basent uniquement sur un langage simple à apprendre, dérivé du C. Deuxièmement, disposer de la puissance (toute relative) d'un CPU ARM, d'une connectivité réseau facile à mettre en place (Ethernet, Wi-Fi ou Bluetooth) et de la possibilité de traiter (et éventuellement stocker) les données simplifie grandement les choses et évite de multiplier les périphériques.

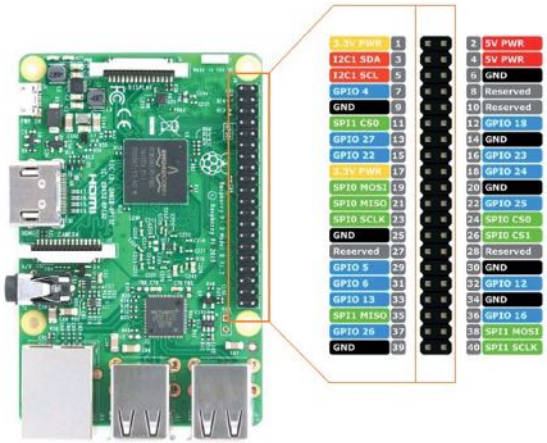
Une communauté active. En bref, si les Arduino conservent leurs avantages en termes d'interfaces électroniques pures (plus efficaces, plus nombreuses et plus robustes), les Raspberry Pi affichent une indéniable supériorité technique sur la partie logicielle. Face à leurs concurrentes, ces deux plateformes disposent également d'un autre avantage de taille : leur communauté très active. Trouver rapidement une réponse face à un problème rencontré avec un programme récalcitrant ne pose généralement pas de problème sur Raspberry Pi. Avec des millions de cartes dans la nature, des forums dédiés et des dizaines de livres, les chances que personne n'ait été touché par le même bug restent très faibles. *A contrario*, les autres cartes ARM (nous en parlons page 46) donnent souvent l'impression de se retrouver dans le désert, en slip et sans eau.

Un prototype du Raspberry Pi, avant la commercialisation.

ATTENTION AUX VIEUX TUTORIAUX ET LIVRES

Si vous voulez jouer avec les GPIO des Raspberry Pi, méfiez-vous d'une chose : les vieux tutoriaux utilisent parfois le brochage de la première génération de Raspberry Pi, qui offre moins de possibilités. De plus, il existe deux variantes dotées d'une numérotation différente pour les entrées/sorties : les broches 00 et 01 deviennent 02 et 03 et la broche 21 devient la 27. Si les programmes que vous testez utilisent ces trois GPIO, pensez à changer les valeurs dans votre code.

Raspberry Pi 3 (40 €)



Nous vous conseillons d'opter directement pour la carte la plus complète, le Raspberry Pi 3, qui intègre du Wi-Fi et du Bluetooth ainsi qu'un SoC ARMv8 (64 bits). Elle dispose d'un connecteur doté de deux rangées de 20 broches pour les GPIO (*General Purpose Input/Output*). On y trouve une vingtaine d'entrées/sorties numériques, une seule sortie PWM hardware, un bus I2C, un bus SPI et un UART (série). Tous sont limités à 3,3 V. Grosse lacune face aux Arduino : le SoC intégré dans le Raspberry Pi ne dispose d'aucune entrée analogique. Impossible donc de lire une tension ou un capteur de température basique ; il vous faudra impérativement passer par un Shield ADC externe qui s'interface en SPI ou en I2C. Pour de nombreux montages, ce manquement s'avérera rédhibitoire. Pensez-y ! Et n'oubliez pas que le Raspberry Pi 3 consomme beaucoup d'énergie : presque 100 fois plus qu'un Arduino ! Oubliez les petites batteries Li-Po : une alimentation secteur de qualité (5 V/2,1 A idéalement) s'impose.

Raspberry Pi A+ (25 €)

Le Raspberry Pi A+ est souvent oublié et sous-estimé, mais cette version propose pourtant quelques avantages, spécialement pour les amateurs d'électronique. Il s'agit du modèle qui consomme le moins (grâce à un CPU qui tourne à seulement 700 MHz) tout en gardant les broches GPIO accessibles. La carte se place entre le Raspberry Pi 3 et le Zero au niveau de la taille et l'unique port USB permet d'installer du Wi-Fi ou de l'Ethernet en cas de besoin.

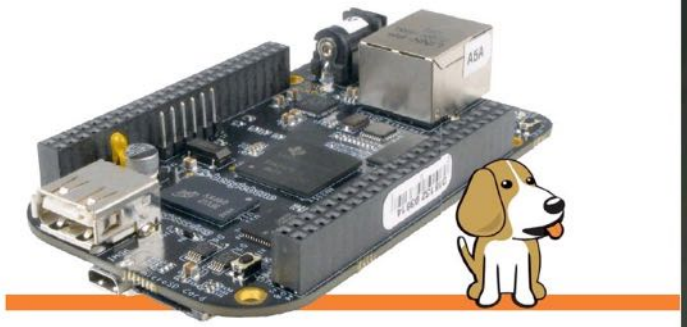
Raspberry Pi Zero (5 €)

Le Raspberry Pi Zero se distingue par un prix très bas (5 dollars) et un format compact. De plus, la carte consomme peu (mais tout de même plus que le modèle A+). Elle souffre toutefois de deux lacunes pour les débutants. D'abord, il vous faudra souder des broches (*header*) pour les GPIO ; par défaut, on n'y trouve que des trous. Ensuite, le connecteur micro-USB *host* s'avère peu pratique pour ajouter des périphériques comme un dongle Wi-Fi (souvent au format USB classique).

Les alternatives au Raspberry Pi

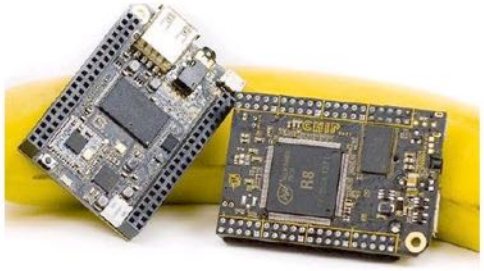
Le succès des Raspberry Pi a mené à la création d'une multitude de clones plus ou moins aboutis. Comme dans le monde Arduino, les millions de cartes vendues aiguisent les appétits et chacun a grappillé une part du gâteau. Nous en avons sélectionné trois, ceux qui nous semblent les plus dignes d'intérêt.

BeagleBone (50 €)



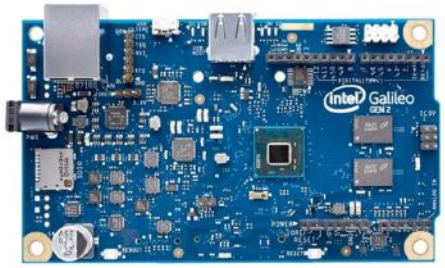
Les cartes BeagleBone représentent la meilleure tentative de marier les Raspberry Pi et les Arduino. Des premiers, elles reprennent le SoC ARM puissant et le support d'un système d'exploitation complet (Linux / Debian). Notre préférée, la BeagleBone Black classique, intègre ainsi un cœur ARM Cortex-A8 à 1 GHz épaulé par 512 Mo de DDR3 et 4 Go de eMMC. Mais les concepteurs lui ont également greffé les fonctionnalités dédiées à l'électronique qui font le succès des Arduino. À l'aide de deux processeurs additionnels dédiés aux GPIO, la BeagleBone Black peut accéder extrêmement rapidement aux dizaines d'entrées/sorties numériques et fournir des fonctions bien utiles : 4x UART, 8x PWM, 2x SPI, des timers, et surtout… des entrées analogiques ! Celles-ci restent certes limitées à 1,8 V maximum (contre 3,3 V pour toutes les autres broches), mais elles ouvrent de nombreuses possibilités. Pour maintenir un tarif raisonnable (50 €), la BeagleBone Black fait toutefois l'impasse sur la sortie HDMI. Un Shield ("Cape") spécifique optionnel rend toutefois l'affichage externe possible. Cette carte représente une alternative intéressante pour l'électronique et pour ne rien gâcher, sa communauté est assez importante.

C.H.I.P. (10 €)



Testée dans *Canard PC Hardware* n° 28, cette micro-carte issue d'un projet Kickstarter offre des performances correctes – elle est animée par un SoC ARM Cortex A8 à 1 GHz –, une intégration excellente (Wi-Fi, USB, GPIO, etc.) et quelques bonnes idées comme un connecteur standard pour une batterie Li-Ion ou un bouton d'allumage directement sur le PCB. Son plus gros avantage demeure son prix : 9 dollars dans sa version de base. Point important : une nouvelle déclinaison (C.H.I.P. Pro) devrait voir le jour d'ici la fin de l'année.

Galileo v2 (80 €)



Pour les allergiques aux puces ARM, Intel propose toujours une carte de développement basée sur un processeur x86, la Galileo. Elle intègre 256 Mo de RAM et un CPU "Quark" (en fait, un antique 486 gravé avec les techniques modernes) à 400 MHz, mais offre surtout une compatibilité avec l'Arduino au niveau des accessoires et de la programmation. Malgré tout, elle peut fonctionner de façon indépendante – comme les Raspberry Pi – avec une distribution Linux comme système d'exploitation. Seule ombre au tableau : Intel semble s'en désintéresser de plus en plus.

Débuter sur Arduino

LE "HELLO WORLD !" ÉLECTRONIQUE

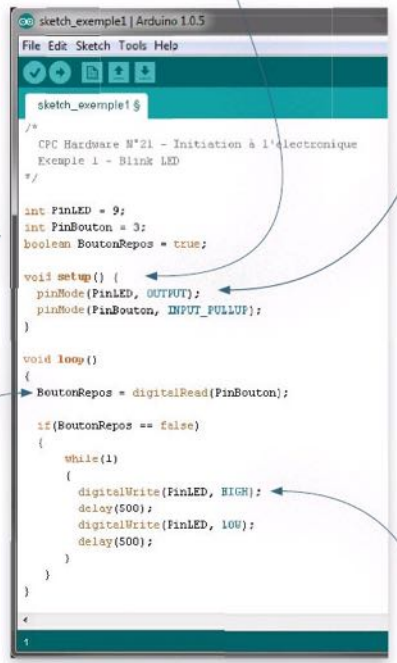
Première étape : nous allons programmer le clignotement d'une LED en appuyant sur un bouton. Cela nous permettra de nous familiariser avec l'IDE Arduino et les entrées/sorties numériques. Pour ce premier test, vous aurez besoin d'une résistance, d'une LED, d'un bouton-poussoir quelconque et d'une carte Arduino Uno. Ah j'oubliais : si vous vous demandez d'où proviennent les schémas que nous utilisons, rendez-vous sur *fritzing.org* : ces gens méritent une visite !

Software

Comme tout programme en C, il est possible de définir des variables. Pour cela, il faut spécifier le type, le nom et éventuellement la valeur d'initialisation. Ici, nous définissons les variables `PinLED` et `PinBouton` qui serviront à stocker la broche utilisée : 9 pour la LED et 3 pour le bouton. Elles sont de type int, c'est-à-dire un nombre entier compris entre -32,768 et -32,767. Nous définissons également une variable `BoutonRepos` de type boolean (binaire ; vrai ou faux) qui stockera l'état du bouton. Nous la définissons à `true` car nous sommes en logique inversée (voir plus bas). Protip : si vous êtes fainéant, utilisez directement la LED intégrée à la carte sur la broche 13.

On trouve ici la structure de base. Un programme Arduino doit impérativement contenir au moins deux fonctions : **`setup()`** et **`loop()`**. La première est exécutée une seule fois au démarrage du microcontrôleur. Elle sert à configurer le type des entrées/sorties et globalement tout ce qui doit l'être avant que le programme principal s'exécute. Celui-ci est constitué par une boucle infinie appelée `loop()`.

Nous sommes dans la boucle principale `loop()`. La première fonction que nous utilisons est **`digitalRead()`** qui permet de lire l'état d'une broche numérique. La fonction renvoie un boolean (vrai ou faux – HIGH "1" ou LOW "0") que nous stockons dans la variable `BoutonRepos`. Cette commande est exécutée en boucle indéfiniment.



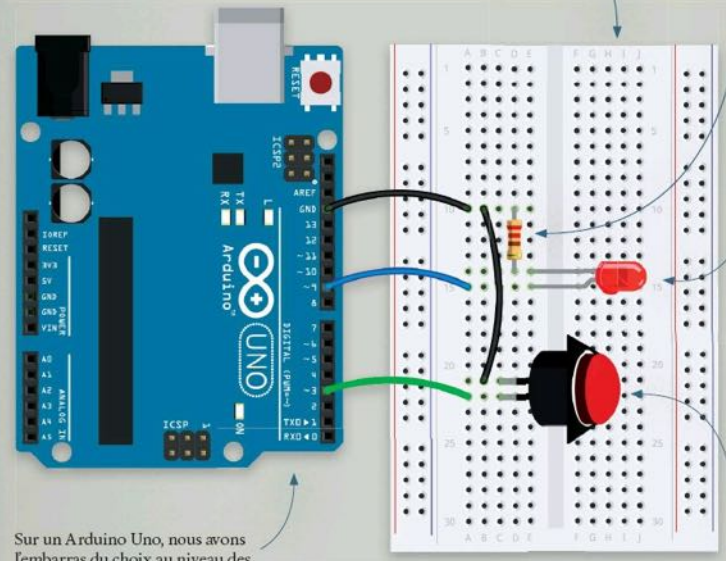
Nous sommes dans la fonction `setup()`. La fonction `pinMode()` sert à définir le type de la broche : entrée ou sortie. Elle accepte en paramètre le numéro de la broche et son sens de communication. Dans notre exemple, la broche 9 appelée `PinLED` est une sortie (***OUTPUT***) et la broche 3 `PinBouton` est une entrée (***INPUT***). À noter que nous utilisons ici le troisième état possible (***INPUT_PULLUP***), identique à INPUT mais qui active en sus la résistance interne de rappel à l'état haut ("*pullup*"). La valeur par défaut – à vide – de l'entrée est donc "1" et l'action du bouton entraîne son passage à "0" ; c'est la logique inversée dont nous parlions plus haut. Cela permet d'économiser une résistance de rappel externe.

Si le bouton est pressé, l'état de l'entrée `PinBouton` passe à 0. Une fois la valeur lue, elle est stockée dans la variable `BoutonRepos` qui passe aussi à 0 (= "LOW", "false"). En conséquence, la fonction conditionnelle **`if()`** est exécutée. Celle-ci est constituée d'une boucle infinie (**`while(1)`**") qui va générer le clignotement. Pour cela, on utilise les fonctions **`digitalWrite()`** qui permet de commander l'état d'une broche. Avec HIGH, elle passe à 5 V et la LED s'allume ; LOW à 0 V et la LED s'éteint. On place des fonctions **`delay()`** pour faire une pause entre chaque état et générer le clignotement. La valeur s'exprime en millisecondes ; 500 correspond donc à 0,5 seconde par état.

Hardware

Un mot sur le fonctionnement de la platine d'essai : les trous situés sur les deux colonnes centrales sont connectés en interne ligne par ligne. Vous obtenez donc deux colonnes indépendantes constituées de dizaines de rangées de cinq connexions reliées électriquement. À l'inverse, les deux petites colonnes situées aux deux extrémités sont destinées à l'alimentation : bleu pour la masse, rouge pour la tension (+5 V généralement). Sur celles-ci, toutes les broches de chaque colonne sont interconnectées pour faciliter le dispatch sur toute la platine. En résumé : au centre, les composants sur des connexions en ligne, aux extrémités l'alimentation sur des connexions en colonne.

Une LED en conduction se comporte comme un fil électrique. Pour ne pas provoquer un court-circuit, il convient donc absolument de placer une résistance en amont ou en aval pour limiter le courant. Comme vous avez parfaitement retenu les leçons précédentes, vous connaissez la valeur à utiliser. Non ? Lisez donc l'encadré en bas de page.



La LED est une diode électroluminescente qui consomme relativement peu de courant (généralement pas plus de 20 mA). Elle peut donc être commandée directement par une sortie de l'Arduino (40 mA maximum). Attention toutefois à ne jamais dépasser cette valeur. À noter également que l'ensemble du microcontrôleur ne peut pas fournir plus de 200 mA. Pour en revenir à la LED, attention au sens ! La cathode (patte la plus courte) doit être reliée à la masse sinon la diode ne sera jamais passante.

Sur un Arduino Uno, nous avons l'embarras du choix au niveau des broches. Pour cet exemple, nous aurons besoin de deux entrées/sorties numériques. Arbitrairement, nous utiliserons la broche 9 en sortie pour le contrôle de la LED et la broche 3 pour le signal en provenance du bouton.

L'interrupteur est relié d'un côté à la masse, de l'autre à l'entrée. Une résistance interne au microcontrôleur ("*pullup*") maintient en effet la broche à un état neutre "haut", c'est-à-dire 5 V ou 1 logique. En la court-circuitant à la masse grâce à un appui sur le bouton, l'entrée change d'état et passe à 0. On aurait également pu faire l'inverse en réactivant la résistance *pullup* et en connectant le bouton au 5 V par l'intermédiaire d'une résistance externe (10 kΩ).

LES SPÉCIFICATIONS D'UNE LED

Vous devez impérativement toujours limiter le courant qui circule dans une LED par une résistance. Sa valeur dépend des spécifications de la LED. Dans la fiche technique, recherchez les caractéristiques ***If*** *(Forward Current)* et ***Vf*** *(Forward Voltage)*. Sur cet exemple d'une LED standard, If = 0.02 A (20 mA) et Vf = 1.85 V. Sachant que les sorties de l'Arduino délivrent du 5 V et que U = R * I, vous aurez besoin d'une résistance théorique de (5-1.85)/0,02 = 157.5 Ω . Pour maintenir une marge de sécurité, utilisez une résistance de 180 Ω (marron-gris-marron) ou de 220 Ω (rouge-rouge-marron).

Electrical/Optical Characteristics at T_a = 25°C

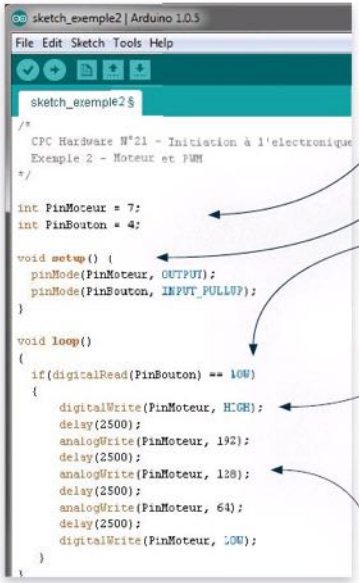
Parameter	Symbol	Minimum	Typical	Maximum	Unit	Test
Luminous Intensity	IV	10	20	30	mcd	IF = 20mA
Viewing Angle	2θ 1/2	-	150	-	degrees	
Peak Emission Wavelength	λP	-	660	-	nm	-
Dominant Wavelength	λD	-	643	-		-
Spectral Line Half-Width	Δλ	-	20	-		-
Forward Voltage	VF	1.5	1.85	2.5	V	IF = 20mA
Power Dissipation	Pd	-	-	85	-	-
Peak Forward Current (Duty 1/10 at 1KHz)	IF (Peak)	-	-	100	-	-
Recommended Operating Current	IF (Rec)	-	20	-	mA	-

Commander une charge en PWM

TRANSISTORS ET MOSFET

L'exemple précédent était indispensable pour expliquer le b.a.-ba. bien qu'il n'ait pas grand intérêt en pratique. Utiliser un microcontrôleur pour faire clignoter une LED n'est pas exactement la killer-app du moment. Nous allons maintenant voir comment rendre l'Arduino autonome (sans besoin d'un PC connecté en USB pour l'alimentation) et comment commander une charge bien plus importante qu'une simple LED. Nous en profiterons également pour parler des sorties "analogiques" PWM.

Software



On commence par attribuer des noms intelligibles aux entrées/sorties comme dans l'exemple précédent. Nous choisissons au hasard la 4 pour le bouton et la 7 pour le moteur. Attention toutefois, toutes les sorties ne sont pas capables de fonctionner en PWM ; seules celles accompagnées d'un "~" le peuvent soit, sur l'Arduino Uno, la 2, 5, 6, 9, 10 et 11.

La partie `setup()` est également très simple : les deux broches sont configurées et la résistance *pullup* est activée sur celle reliée au bouton pour maintenir un état au repos "HIGH" / +5 V.

La boucle principale a été simplifiée. L'appui sur le bouton est détecté directement dans la fonction `if()` : si quelqu'un appuie dessus, l'entrée passe à l'état bas (LOW) et le code est exécuté. Sinon, le programme boucle indéfiniment.

Une broche configurée en sortie peut être commandée en mode numérique ou analogique. Dans notre boucle, nous commençons par activer la charge normalement avec la fonction `digitalWrite()` comme nous l'aurions fait d'une simple LED connectée directement à la sortie. Nous rajoutons ensuite une temporisation de 2 500 ms (2,5 secondes).

La modulation PWM commence ensuite avec la fonction `analogWrite()`. Celle-ci va faire commuter la sortie extrêmement rapidement avec un certain ratio entre les périodes "ON" et les périodes "OFF". La valeur s'exprime de 0 à 255. Si l'on choisit une valeur de 128 par exemple, le temps passé en état "ON" sera identique à celui passé en "OFF". Si en revanche on choisit 64, la sortie sera à l'état haut pendant ¼ du temps et à l'état bas pendant les ¾ du temps. Avec une LED, celle-ci clignotera extrêmement rapidement et fournira au final 25 % de sa luminosité. L'œil humain ne distinguera évidemment pas le clignotement vu la fréquence employée. Sur ce test, nous faisons varier par étape la charge à 75 % (255/192), 50 % (255/128) et 25 % (255/64) du maximum puis nous la coupons jusqu'au cycle suivant (nouvel appui sur le bouton).

CHOISIR UN MOSFET

Les MOSFET existent sous toutes les formes : des minuscules composants SOT23 à monter en surface aux énormes boîtiers TO3. Les plus courants sont les boîtiers TO92 (petits) et TO220 (moyen). Généralement, plus le boîtier est gros et plus la charge à commander peut être importante. Pour choisir un MOSFET, plusieurs valeurs sont à prendre en compte. Vérifiez d'abord le V_{GS} : c'est la tension d'activation. Pour un Arduino, vérifiez que la valeur "max" garantie soit inférieure à 5 V. La valeur V_{DSS} est également importante : c'est la tension maximum que vous pourrez commander. En principe, elle est largement supérieure à 20 V sur les boîtiers TO92 et plus ; elle atteint souvent la centaine de volts. Enfin, I_S indique le courant maximum que le MOSFET peut commuter de manière continue. Ici, l'IRF540N atteint les 33 A mais pour cela, il aura besoin d'un gros dissipateur. Si votre charge ne dépasse pas quelques ampères (ce qui est déjà énorme), vous pouvez vous en passer.

Electrical Characteristics @ T_J = 25°C (unless otherwise specified)

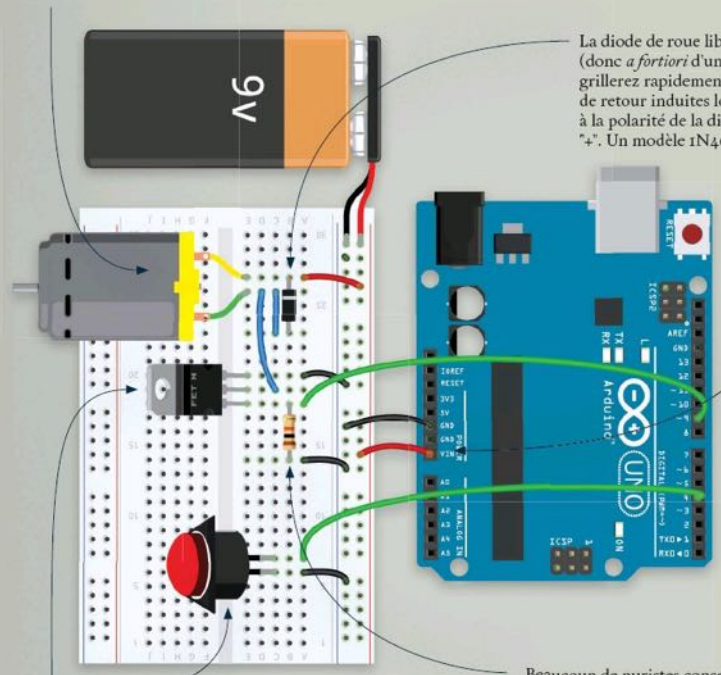
	Parameter	Min.	Typ.	Max.	Units	Conditions
$V_{(BR)DSS}$	Drain-to-Source Breakdown Voltage	100	—	—	V	V_{GS} = 0V, I_D = 250µA
$\Delta V_{(BR)DSS}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	—	0.12	—	V/°C	Reference to 25°C, I_D = 1mA
$R_{DS(on)}$	Static Drain-to-Source On-Resistance	—	—	44	mΩ	V_{GS} = 10V, I_D = 16A ④
$V_{GS(th)}$	Gate Threshold Voltage	2.0	—	4.0	V	V_{DS} = V_{GS}, I_D = 250µA
g_{fs}	Forward Transconductance	21	—	—	S	V_{DS} = 50V, I_D = 16A④

Source-Drain Ratings and Characteristics

	Parameter	Min.	Typ.	Max.	Units	Conditions
I_S	Continuous Source Current (Body Diode)	—	—	33	A	MOSFET symbol showing the integral reverse p-n junction diode.
I_{SM}	Pulsed Source Current (Body Diode)①	—	—	110		
V_{SD}	Diode Forward Voltage	—	—	1.2	V	T_J = 25°C, I_S = 16A, V_{GS} = 0V ④
t_{rr}	Reverse Recovery Time	—	115	170	ns	T_J = 25°C, I_F = 16A

Hardware

On trouve ici la charge à commander. Pour cet exemple, nous avons utilisé un petit moteur classique mais nous aurions également pu connecter une rangée de LED, une LED de forte puissance ou bien encore un ventilateur. L'une des broches de la charge est à relier à sa tension d'alimentation (ici 9 V), l'autre à la masse grâce à la broche Drain du MOSFET qui fait office d'interrupteur. Il est possible de faire varier l'intensité d'une LED ou la vitesse de rotation d'un moteur classique en PWM. Mais attention : ne tentez pas d'utiliser du PWM sur un moteur type *brushless* comme les ventilateurs de PC ! Leur fonctionnement interne ne le supporte pas. Mais si vous souhaitez tout de même commander la vitesse de rotation d'un ventilateur, procurez-vous un modèle "4-pin PWM" et commandez la broche "PWM" à l'aide d'un petit MOSFET. Les broches "+" et "-" du ventilateur sont alors à connecter directement à la tension d'alimentation.



La diode de roue libre entre les bornes d'une bobine (donc *a fortiori* d'un moteur) est indispensable. Sans elle, vous grillerez rapidement le MOSFET à cause des fortes tensions de retour induites lors de l'interruption du moteur. Attention à la polarité de la diode : la cathode (bague) doit être vers le "+". Un modèle 1N400x fera parfaitement l'affaire.

Nous allons alimenter l'Arduino par une batterie externe afin de le rendre autonome. La carte Uno peut accepter une tension de 7 à 12 V par le biais du jack 2,1 mm ou par la broche **VIN**. C'est celle-ci que nous allons utiliser dans le cas présent en la connectant à une batterie 9 V classique. Nous vous déconseillons d'utiliser une tension de 12 V à moins de ne charger qu'*a minima* les sorties du microcontrôleur. Faites bien attention à ne pas confondre VIN et l'entrée 5 V, sous peine de griller instantanément la carte !

Le bouton utilisé ici est monté de manière identique à celui de l'exemple précédent : il est connecté à la masse et court-circuite la résistance *pullup* interne lors de son activation.

Beaucoup de puristes conseillent d'utiliser une résistance "*pulldown*" d'une dizaine de kΩ entre la broche **Gate** du MOSFET et la masse afin de s'assurer d'un état connu "à vide". Ne le répétez à personne, mais je n'en ai jamais utilisé sur aucun de mes montages et tout a toujours fonctionné sans problème…

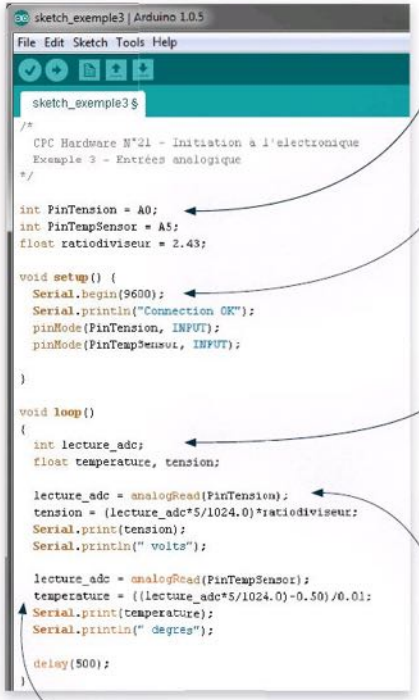
Pour commander une forte charge, il faut utiliser un transistor. Seulement voilà : le bon vieux transistor bipolaire à papa est désormais dépassé. Lorsqu'il s'agit d'utiliser un régime de commutation (interrupteur commandé), un MOSFET sera bien plus efficace et plus simple à mettre en œuvre dans la plupart des cas. Il existe deux types de MOSFET – N et P – qui se différencient par leur polarité. Vous trouverez une pléthore de modèles différents sur le marché. Nous utiliserons ici un MOSFET de type N et plus précisément un IRF540, très courant et peu cher. Il permet de commander une charge de plusieurs dizaines d'ampères avec une tension maximale de 100 V entre sa broche **Drain** (à connecter à la charge) et sa broche **Source** (à relier à la masse). La commutation est commandée par une tension comprise entre 4 V et 20 V sur la broche **Gate**. Les 5 V d'une sortie de l'Arduino feront donc parfaitement l'affaire. La vitesse de commutation d'un MOSFET est extrêmement rapide, ce qui permet de le faire fonctionner en PWM pour faire varier la vitesse de rotation d'un moteur par exemple.

Mesurer une tension / un capteur

OU COMMENT CONSTRUIRE UNE STATION MÉTÉO

Il existe d'innombrables types de capteurs différents. Les plus simples délivrent une tension qu'il est possible de lire grâce aux entrées analogiques des microcontrôleurs. L'Arduino Uno en contient six, connectées à un convertisseur analogique/numérique (ADC) d'une précision de 10 bits. Dans cet exemple, nous allons vous montrer comment lire une tension quelconque et comment utiliser un capteur (de température par exemple). Il devient alors possible de commander un ventilateur, un moteur ou tout autre appareil en fonction d'une grandeur physique.

Software



La définition des variables sort des règles habituelles puisque le nom des entrées analogiques n'est pas un chiffre mais une chaîne : de A0 à A5. Il s'agit en fait d'un alias interne, ce qui permet de les utiliser dans un *int*. Nous définissons également le ratio du diviseur de tension dans un type **float** (qui peut accueillir un nombre à virgule).

Le `setup()` comporte, en plus de la définition des broches analogiques en sortie sans PULLUP ("INPUT"), l'initialisation de l'interface série virtuelle avec la commande **`Serial.begin()`**. Le chiffre passé en paramètre indique la vitesse de communication en baud. Nous utilisons 9600 ici mais il est possible de monter à 115 200… et plus ! Pour vérifier le fonctionnement de l'interface, nous envoyons une chaîne *"Connection OK"* à l'interface série par la commande **`Serial.println()`**. Celle-ci est identique à la commande **`Serial.print()`** mais envoie également un retour à la ligne automatique.

Nous définissons ici quelques variables supplémentaires. Contrairement à celles définies en début de programme, elles sont temporaires. Nous utilisons `lecture_adc` (int) pour stocker la valeur brute (de 0 à 1024) du convertisseur analogique/numérique et deux *float* pour la température et la tension réelle.

Pour lire la tension de la batterie 9 V, on commence par récupérer la valeur brute de la broche A0 par le biais de la fonction `analogRead()`. Une fois celle-ci récupérée, il faut la multiplier par la tension de référence (5 V) et la diviser par le nombre de "pas" du convertisseur ADC (10 bits donc 1024 pas). On obtient alors la tension réelle au niveau de la broche A0. Il ne reste plus qu'à la multiplier par le ratio du diviseur pour retrouver la tension d'origine en amont de celui-ci. Une petite subtilité à prendre en compte : comme la variable qui sert au calcul est de type *float*, il convient de toujours mettre un point sur les nombres fixes utilisés : "1024.0" et non "1024". La valeur est ensuite envoyée à l'interface série par une commande `Serial.print()`.

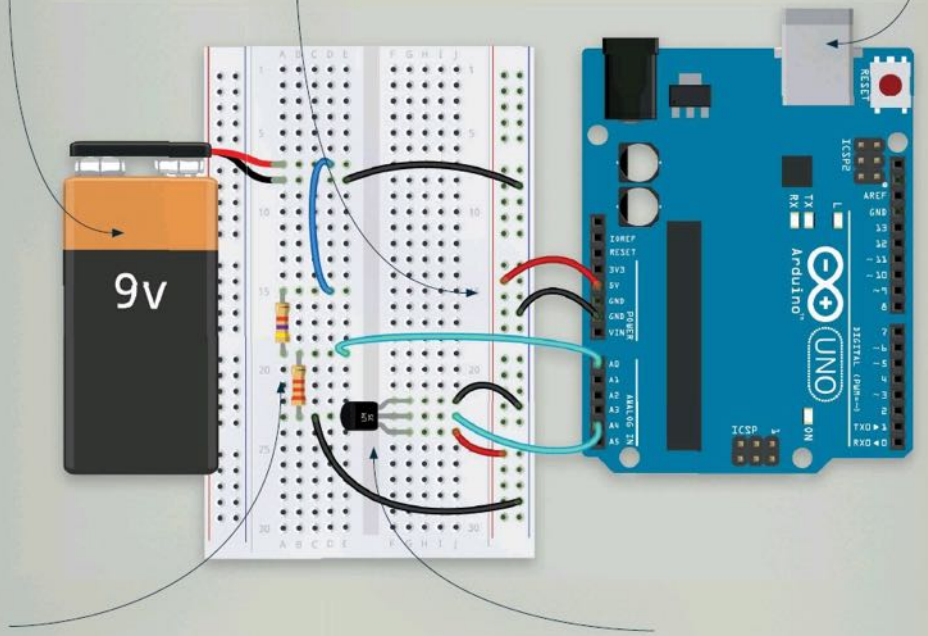
Pour la température, on garde le même principe mais le calcul est un peu différent. Le datasheet du MCP9700 nous indique d'abord qu'à 0 °C, la sortie est à 500 mV (0,5 V). Il convient donc de retrancher cette valeur après la première conversion. Ensuite, le même datasheet nous dit que la tension retournée est proportionnelle à la température à raison de 10 mV (0,01 V) par degré. C'est la raison de la division finale par 0,01. On obtient alors une valeur en degrés réels qu'il suffit d'envoyer à la sortie série. Une fois le programme envoyé à l'Arduino, il suffit de se rendre dans le menu "Tools" puis "Serial Monitor" de l'IDE pour obtenir les données renvoyées. Vous devriez alors voir s'afficher les informations de tension et de température renvoyées par la carte au rythme de 2 fois par seconde (`delay(500)`).

Hardware

À titre d'exemple, nous utilisons une batterie 9 V comme source de courant mais nous pourrions utiliser n'importe quelle autre tension. Le but est ici de montrer comment mesurer une tension supérieure à 5 V. Prenez tout de même garde à ne pas dépasser 50 V !

À noter que pour simplifier le schéma, nous avons décidé de ne pas utiliser la pile (ou toute source de courant entre 7 et 12 V) pour alimenter l'Arduino. Cela aurait bien sûr été possible en respectant les règles précédemment décrites ; l'important reste que l'Arduino partage une masse avec la source de "haute" tension à mesurer.

Lire une tension ou une température, c'est bien, mais encore faut-il pouvoir l'utiliser. Il serait bien sûr possible de créer facilement un thermostat en utilisant les sorties numériques et en commandant un ventilateur ou une résistance chauffante mais nous avons choisi de reporter cette tension/température au PC par l'intermédiaire du port USB. L'occasion de faire un essai sur la liaison série !



Le pont diviseur est la clé de voûte du dispositif. Il permet de diviser la tension pour la rendre compatible avec les limitations à 5 V des entrées de l'Arduino. Il se compose de deux résistances en série qui relient la "haute" tension à mesurer à la masse. La tension divisée est disponible entre ces deux résistances. Le calcul d'un pont diviseur est simple $Vout = Vin * (R2 / (R1 + R2))$. Dans notre exemple nous avons Vin = 9 V, il convient donc de diviser la tension au moins par deux. Nous avons utilisé une résistance R1 (en haut) de 47 kΩ et une résistance R2 (en bas) de 33 kΩ. Cela nous donne une résistance totale de 80 kΩ, soit un peu plus de 0,1 mA consommé sur la batterie. Le facteur de division est de 80/33=~2,42. Une tension de 9 V sera donc abaissée à 3,71 V. Tant que la valeur ne dépasse pas les 5 V, aucun problème pour l'Arduino. Sinon, il suffit d'utiliser un ratio de division plus élevé.

Le capteur de température utilisé est un MCP9700A de Microchip. Il en existe bien d'autres comme le fameux LM35, mais celui-ci est fiable, peu coûteux, assez précis (2 °C) et il dispose d'une large gamme de mesures (-40 / +150 °C). Il suffit de lui fournir une tension comprise entre 2,3 et 5,5 V pour qu'il délivre une tension proportionnelle à la température sur sa sortie. Le ratio est de 10 mV/°C avec un 0 °C à 500 mV.

Utiliser les interruptions

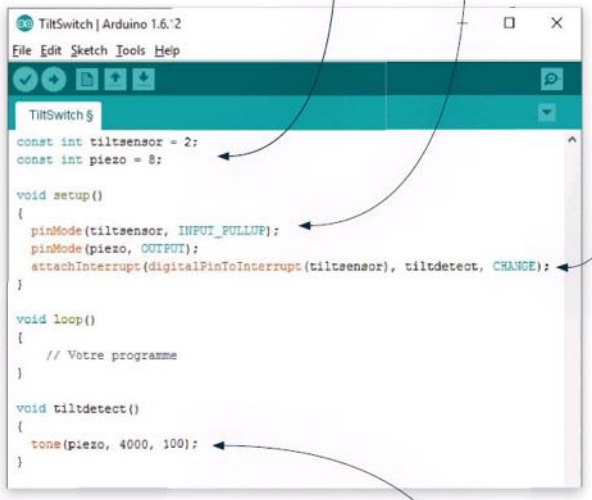
UNE QUESTION DE PRIORITÉS

Nous allons maintenant voir un concept très important dans la conception de certains montages Arduino : les interruptions. Comme nous l'avons vu précédemment, le programme principal s'exécute dans une boucle `loop()`. Les interruptions permettent de réagir immédiatement à un stimulus sur une des entrées en quittant temporairement cette boucle principale, le temps d'exécuter une fonction prédéfinie. Dans notre exemple, nous utiliserons cette fonctionnalité pour concevoir un détecteur de mouvement basique.

Software

Nous commençons par définir deux variables qui correspondent aux broches de l'Arduino que nous allons utiliser : la 8e pour le buzzer piézoélectrique et la 2e pour le capteur de tilt, qui sera attachée à une interruption.

La configuration des broches avec `pinMode()` n'appelle aucun commentaire particulier : celle dédiée au capteur de mouvement est simplement paramétrée comme une entrée (avec *pullup* active) alors que l'autre, qui accueille le buzzer, se voit définie en tant que sortie.



La définition d'une interruption se fait dès la fonction initiale `setup{}`. On utilise pour cela la fonction `attachInterrupt()` qui exige trois paramètres. En premier, le numéro de l'interruption. Attention : il ne s'agit pas du numéro de la broche. Pour effectuer le lien, la librairie Arduino dispose toutefois d'une fonction dédiée – `digitalPinToInterrupt()` – que nous allons utiliser en lui fournissant comme paramètre le numéro de la broche physique (soit la variable `tiltsensor`). En deuxième, on trouve le nom de la fonction qui sera appelée en cas de déclenchement d'une interruption. Celle-ci doit impérativement se trouver plus loin dans votre programme. Ici, nous utiliserons le nom `tiltdetect`. Enfin, le troisième argument concerne le mode de déclenchement. Les valeurs possibles sont `LOW` (quand la broche est à l'état logique 0), `CHANGE` (lors d'un changement d'état), `RISING` (passage de 0 à 1) et `FALLING` (transition de 1 à 0).

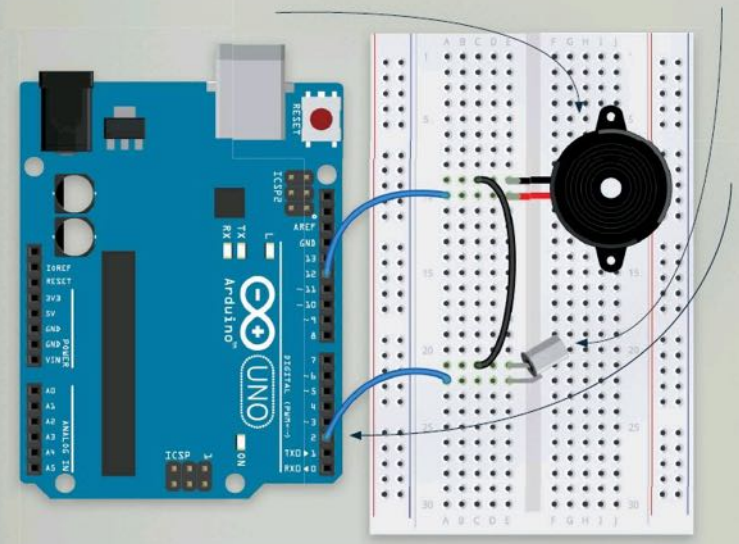
Le programme principal qui se trouve dans la boucle `loop()` peut contenir toute instruction à votre choix. Le flux de traitement sera toutefois interrompu lorsqu'une interruption se déclenchera. Le microcontrôleur exécutera alors la fonction prédéfinie (`tiltdetect` dans notre exemple) avant de poursuivre la boucle principale là où elle s'était arrêtée.

Cette fonction est appelée lorsqu'une interruption est détectée (mais rien ne vous empêche de l'appeler également à partir de la boucle principale). Dans notre exemple, nous l'utilisons pour exécuter `tone()`, qui permet de générer un son via un signal carré sur notre buzzer. Celle-ci accepte également trois paramètres : le numéro de la broche (`piezo`, soit 8 ici), la fréquence du son (4 000 Hz) et la durée d'émission (100 ms).

Hardware

Le buzzer piézoélectrique présente l'avantage de faire beaucoup de bruit tout en consommant très peu d'énergie (moins de 20 mA). Il se connecte directement à la masse et sur une broche de l'Arduino, qui devra le commander à l'aide d'un signal alternatif. La fréquence de ce signal donnera la fréquence du son. La plupart des buzzers sont spécifiés à une fréquence de résonance (généralement 2 kHz ou 4 kHz) qui permet d'obtenir le volume le plus élevé.

Le capteur de *tilt* agit comme un détecteur de mouvement simpliste. Il est constitué d'une petite bille métallique qui, par défaut, repose sur de minuscules contacts. À la moindre vibration, celle-ci saute et rompt la connexion électrique entre les broches du composant. Il se comporte donc comme un interrupteur commandé par un (faible) mouvement.
À noter que nous n'utilisons pas une résistance *pullup* externe vu que le microcontrôleur de l'Arduino en contient déjà une.



Nous choisissons la broche n° 2 de l'Arduino pour y connecter le capteur de *tilt*. Celle-ci sera configurée en software pour gérer une interruption. Il convient de noter que toutes les broches ne le permettent pas. Ainsi, sur les Arduino Uno, Mini et Nano, seules la 2e et la 3e peuvent être reliées à une interruption. Les Arduino Leonardo et Micro disposent de 5 broches compatibles (0, 1, 2, 3, 7) contre 6 pour les Arduino Mega (2, 3, 18 à 21). Enfin, sur les modèles Zero, Due et 101, toutes les broches fonctionneront.

■ *Un interrupteur Reed et plusieurs modules accéléromètres montés sur des* breadboards.

D'AUTRES DÉTECTEURS DE MOUVEMENTS

Si le capteur mécanique de tilt reste très correct pour détecter de petits (ou gros) chocs, il n'en demeure pas moins très imprécis. Il existe d'autres types de composants dédiés à des usages plus particuliers, ou offrant une bien meilleure précision. On trouve par exemple les interrupteurs dits Reed. Ils sont constitués de deux lames souples métalliques séparées de moins d'un millimètre. En présence d'un aimant, ils se touchent et créent un contact électrique entre les broches du composant. Ces composants sont largement utilisés pour détecter l'ouverture d'une fenêtre ou d'une porte par exemple : il suffit de coller un petit aimant sur le battant à proximité immédiate de l'interrupteur Reed qui, lui, se trouvera sur le montant. En cas d'ouverture de la porte, le circuit fermé s'ouvrira et pourra générer une alarme. Côté montage, rien ne change : il suffit d'en connecter un à la place du capteur de tilt. Les accéléromètres et autres capteurs électroniques de tilt s'avèrent en revanche plus compliqués à mettre en œuvre. D'abord parce qu'ils ne sont disponibles que sous la forme de microscopiques composants BGA quasiment impossibles à souder pour un amateur. Heureusement, il existe de nombreux *breadboards* (petits circuits imprimés pré-soudés) qui permettent de les utiliser de manière plus conventionnelle. Ensuite, ces composants exploitent souvent un bus de communication numérique I2C ou SPI et nécessitent l'utilisation de librairies propriétaires. On trouve toutefois encore quelques accéléromètres (comme les ADXL3xx d'Analog Devices) dotés de sorties analogiques.

Connecter un afficheur LCD alphanumérique

MAIS… QUEL TEMPS FAIT-IL ?

Faire clignoter une LED, c'est bien, mais parfois, le besoin se fait sentir d'avoir un retour plus précis et plus complet. C'est à ce moment qu'interviennent les afficheurs LCD. Il en existe de toutes tailles et à tous les prix, mais les moins chers restent toutefois les modèles alphanumériques. On en trouve principalement en version 2x16 et 4x20 caractères. Dans cet exemple, nous allons en utiliser un pour afficher la température ambiante.

Software

L'IDE Arduino dispose d'une librairie spécialisée dans la gestion des afficheurs LCD alphanumériques basés sur le chipset Hitachi HD44780 (de loin le plus courant). Il convient de l'inclure dès le début du code.

La fonction `lcd.begin()`, appelée ici dans la boucle d'initialisation `setup()`, permet de spécifier à la librairie le type d'afficheur. Elle accepte en paramètre le nombre de caractères par ligne, puis le nombre de lignes. Nous utilisons ici un modèle à 16x2 caractères, donc nous le configurons comme tel. À noter l'ajout d'un délai de 250 ms, généralement facultatif, mais qui permet de s'assurer que le contrôleur de l'écran est bien initialisé.

LCD | Arduino 1.6.12

File Edit Sketch Tools Help

LCD

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(2,5,8,9,10,11);

void setup() {

  lcd.begin(16,2);
  delay(250);

}

void loop() {

  lcd.clear();

  int temp_brute = analogRead(A0);

  float temperature = ((temp_brute*5.0/1024.0)-0.50)/0.01;

  lcd.setCursor(0, 0);
  lcd.print("Temperature :");
  lcd.setCursor(0, 1);
  lcd.print(temperature);
  lcd.print(" degres Celsius");

  delay(1000);
}
```

Ici, nous configurons le lien physique entre les broches de l'Arduino et celles de l'afficheur. Ce dernier fonctionne généralement en mode 4 et 8 bits. Nous utiliserons le plus simple, le mode 4 bits pour limiter le câblage. La configuration initiale exige au minimum 6 paramètres, qui correspondent aux broches RS, ENABLE, D4, D5, D6 et D7 de l'afficheur. Nous allons les connecter aux pins 2, 5, 8, 9, 10 et 11 de l'Arduino, respectivement. Mais vous pouvez aussi utiliser n'importe quelle autre sortie numérique de la carte. Pensez tout de même à consulter le datasheet de l'afficheur pour vous assurer de la correspondance.

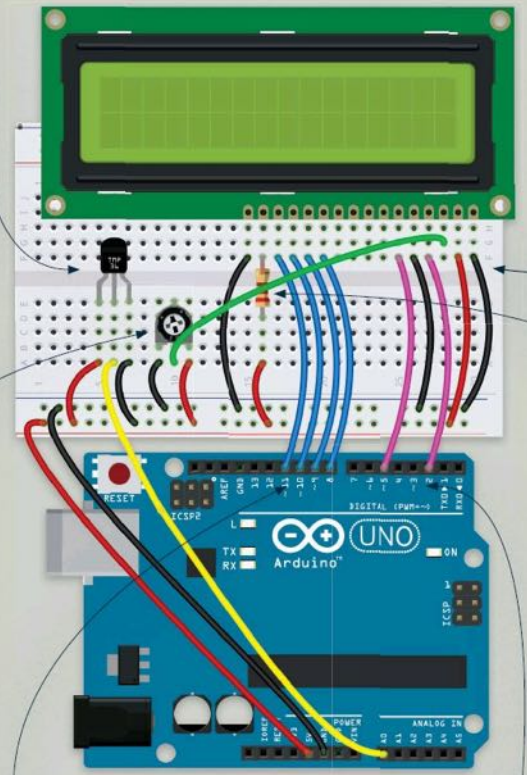
La boucle principale `loop()` commence par l'effacement complet de l'afficheur avec la fonction `lcd.clear()`. Sans elle, les nouveaux caractères seraient écrits par-dessus les anciens, ce qui mènerait vite à un texte illisible.

Il ne reste plus qu'à afficher les informations sur l'écran. La commande `lcd.setCursor()` permet de définir l'emplacement initial du curseur, où sera écrit le premier caractère. Elle accepte deux paramètres : la colonne et la ligne (qui commencent toutes deux par 0). Enfin, `lcd.print()` envoie le texte (ou la valeur) à l'afficheur.

Nous lisons d'abord la valeur brute de la broche A0, sur laquelle est connectée le capteur de température. La valeur récupérée (un entier entre 0 et 1023) est ensuite stockée dans la variable `temp_brute`. L'étape suivante consiste à la convertir en un nombre flottant correspondant à la température en degrés Celsius. Pour cela, nous réutilisons la formule déjà expliquée précédemment, qui prend en compte les spécificités du capteur de température (ici un LM35) ainsi que celles de l'ADC 10 bits de l'Arduino.

Hardware

Le capteur de température TMP36. Celui-ci renvoie une tension qui varie en fonction de la température à raison de 10 mV par °C, avec un zéro degré calibré à 500 mV. À noter que nous aurions pu utiliser d'autres types de capteur, comme un capteur d'humidité. Certains fonctionnent de la même manière (en analogique), d'autres utilisent le bus de communication SPI ou I2C. Le capteur est connecté à l'entrée analogique A0 de l'Arduino.



L'afficheur LCD exige évidemment une alimentation (ici 5 V), que nous dérivons de celle de l'Arduino.

Outre le rétroéclairage, les afficheurs nécessitent également une autre tension pour configurer le contraste des caractères (par rapport au fond). On utilise pour cela un petit potentiomètre de 10 kΩ. Le point central dont la résistance varie en fonction de la rotation est connecté à une broche spécifique.

Le rétroéclairage de l'afficheur LCD utilise généralement des LED qui doivent en principe être alimentées, souvent indépendamment du reste. Attention : une résistance s'impose généralement, sous peine de griller le rétroéclairage. La valeur de celle-ci dépend des spécifications techniques. Ici, nous utilisons un modèle de 220 Ω.

Les afficheurs LCD alphanumériques classiques communiquent normalement via une interface parallèle 8 bits. Toutefois, les contrôleurs Hitachi HD44780 peuvent aussi fonctionner en mode 4 bits. Le transfert s'avère théoriquement un peu plus lent, mais vous ne percevrez pas la différence en pratique. Gros avantage : le mode 4 bits se contente de 4 fils, ce qui limite la complexité du câblage (et le nombre de broches utilisées sur l'Arduino).

Outre les broches de données, les afficheurs exigent aussi au moins deux broches de contrôle. La première sert à définir si les données envoyées correspondent à une commande ou s'il s'agit d'une donnée. Elle se nomme *"Register Select"* ou simplement "RS". L'autre, baptisée *"Enable"*, "EN" ou simplement "E", indique à l'afficheur que de nouvelles données sont prêtes à être traitées. Enfin, notez la présence d'une broche *"Read/Write"* (ou "RW") qui ne sert généralement pas : elle doit tout de même être connectée à la masse.

Afficher du texte sur un écran LCD graphique

POUR NE PLUS VOIR LA VIE EN VERT

La plupart des afficheurs LCD/OLED graphiques évolués embarquent un lecteur de cartes microSD. Celui-ci peut servir pour enregistrer des données récupérées par des capteurs, mais aussi – et surtout – pour stocker des images destinées à l'écran. La mémoire embarquée par l'Arduino s'avère en effet très limitée et ne permettrait pas de gérer beaucoup de graphiques complexes ou d'images bitmap. La documentation du site Arduino.cc propose quelques exemples qui permettent d'aller beaucoup plus loin avec cet écran, comme la possibilité de lire une image BMP et de l'afficher *(cpc.cx/htf)*.

Software

Le programme importe les bibliothèques qui permettent de commander directement la dalle LCD et le lecteur de cartes SD. `TFT.H` s'occupe de l'affichage, `SD.H` des cartes SD et `SPI.H` du bus utilisé pour la communication.

La fonction `begin()` active la dalle, et les suivantes permettent d'afficher du texte. `Background()` remplit l'arrière-plan, avec des couleurs codées en RGB (rouge, vert, bleu). La valeur maximale est 255 pour chaque couleur : (255,255,255) représente du blanc et (0,0,0) du noir. `stroke` utilise la même technique pour la couleur du texte et `setTextSize` règle la taille du texte, de 1 à 10 (10 à 100 pixels de haut). Enfin, la commande `text` affiche du texte aux coordonnées indiquées. Il s'agit du point de départ du coin supérieur gauche du texte, en sachant que celui-ci possède les coordonnées (0,0) alors que le coin inférieur droit se trouve en (159,127). Dans notre cas, avec une police en taille 2 (20 pixels), la seconde ligne commence donc en 20,40. Si nous avions utilisé une police "3", il aurait fallu passer en (20,30), etc.

Cette partie ouvre le fichier en écriture et écrit un message à l'intérieur avec `myFile.print`.

L'étape suivante ouvre le fichier en lecture. La boucle va permettre de récupérer le contenu. `TFTscreen.write(myFile.read());` lit le premier octet, l'affiche sur l'écran et avance d'un caractère dans le fichier, jusqu'à ce qu'elle arrive à la fin.

SDnew | Arduino 1.6.12

File Edit Sketch Tools Help

SDnew

```
#include <SD.h>
#include <TFT.h>
#include <SPI.h>

File myFile;
TFT TFTscreen = TFT(10, 9, 8);

void setup()
  {
  TFTscreen.begin();
  TFTscreen.background(255, 0, 0);
  TFTscreen.stroke(255, 255, 255);
  TFTscreen.setTextSize(2);
  TFTscreen.text("Canard PC", 20, 20);
  TFTscreen.setTextSize(3);
  TFTscreen.stroke(0, 255, 0);
  TFTscreen.text("Hardware", 20, 40);
  delay(10000);

  TFTscreen.background(255, 0, 0);
  TFTscreen.stroke(255, 255, 255);
  TFTscreen.setTextSize(1);
  SD.begin(4);

  myFile = SD.open("exemple.txt", FILE_WRITE);
  myFile.close();

  if (SD.exists("exemple.txt"))
    {
    Serial.print("Le fichier existe\n");
    }

  myFile = SD.open("exemple.txt", FILE_WRITE);
  myFile.print("Vive Canard PC Hardware\n");
  myFile.close();
  myFile = SD.open("exemple.txt");
  while (myFile.available())
    {
    TFTscreen.write(myFile.read());
    }
  myFile.close();

  SD.remove("exemple.txt");
  }

void loop()
  {
  }
```

L'écran LCD communique grâce au bus SPI, mais nécessite aussi quelques broches supplémentaires (CS, DC et RESET). La correspondance avec les pins Arduino (dans notre exemple les 8, 9 et 10) est configurée ici.

Après une petite attente, l'écran est effacé en remettant le fond en rouge (255,0,0).

La commande `SD.begin(4);` active la SD via la broche 4.

Avec `myFile = SD.open("exemple.txt", FILE_WRITE);`, nous ouvrons un fichier `exemple.txt` sur la carte SD, en écriture. S'il n'existe pas (c'est le cas), le programme va le créer. `myFile.close();` le ferme ensuite.

`SD.exists` vérifie si le fichier a été ajouté (un message sur l'écran le confirme).

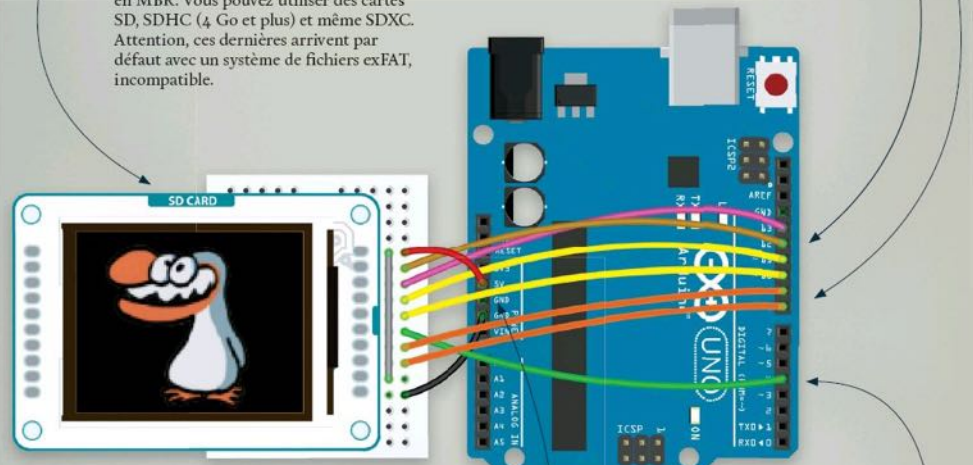
La dernière commande, `SD.remove("exemple.txt");`, efface le fichier.

Hardware

Dans notre cas, l'écran dispose d'une définition de 160 x 128 pixels et affiche des couleurs 18 bits (~262 000). Pour la carte SD, les Arduino passent par un mode précis de la norme, qui permet un accès en SPI (un bus série). Le débit n'atteint évidemment pas celui d'une carte UHS, mais pour écrire quelques données ou lire une image, il offre des performances suffisantes. La microSD doit être formatée en FAT16 ou en FAT32 avec une partition en MBR. Vous pouvez utiliser des cartes SD, SDHC (4 Go et plus) et même SDXC. Attention, ces dernières arrivent par défaut avec un système de fichiers exFAT, incompatible.

La septième et la huitième broche du module afficheur se relient aux broches 9 et 8 de l'Arduino, elles servent à commander l'écran.

En partant du haut : la broche 2 de l'écran à la 12 de l'Arduino (MISO), la 3 à la 13 (SCK), les 4 et 5 aux broches 11 (MOSI) et 10 (SS). Il s'agit des broches utilisées pour la connexion SPI, un bus série utilisé par l'écran et le lecteur de cartes SD.



Placez le périphérique face à vous, la zone notée SD Card vers l'avant et celle équipée d'une flèche verte sur la *breadboard*.

La broche du haut doit être reliée à la source d'alimentation en 5 V de l'Arduino, celle du bas à la masse.

La sixième broche de l'écran commande la carte SD et se connecte à la broche 4 de l'Arduino. Il est possible d'utiliser une autre sortie numérique en la configurant ensuite dans le code.

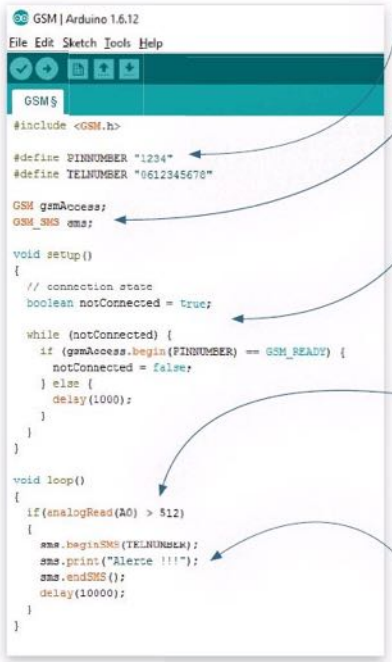
Communiquer à distance

LES JOIES DU RÉSEAU GSM

Les cartes Arduino peuvent facilement être embarquées dans des périphériques mobiles. Elles présentent l'avantage de n'exiger que peu d'énergie (surtout comparées aux Raspberry Pi) et offrent une autonomie décente sur batterie. La communication, en utilisant le réseau de téléphone mobile (GSM), ouvre un champ d'application très large. Si le Shield GSM original d'Arduino n'est plus fabriqué, on trouve toujours sur Internet des dizaines de modèles compatibles pour environ 30 euros. Armé d'une carte SIM jetable, concevons une alarme qui nous envoie un SMS quand une personne ouvre la porte d'une armoire fermée. Une photorésistance détecte la présence de lumière (et donc l'ouverture) et active l'émission du message.

Software

Nous commençons par importer la bibliothèque qui prend en charge les modules GSM. Pour la suite, deux informations sont nécessaires : le code PIN de la carte SIM pour l'activation du module (vous pouvez éventuellement passer cette étape en supprimant cette sécurité depuis un téléphone) et le numéro auquel vous voulez envoyer un SMS.



Les deux variables permettent de définir des noms compréhensibles pour les commandes que nous utilisons.

La boucle `setup` active le module GSM et attend qu'il indique que la connexion au réseau est effective. Tant que ce n'est pas le cas, la commande d'activation se relance toutes les secondes. `gsmAccess.begin` nécessite le code PIN de la carte SIM et renvoie `GSM_READY` en cas de succès.

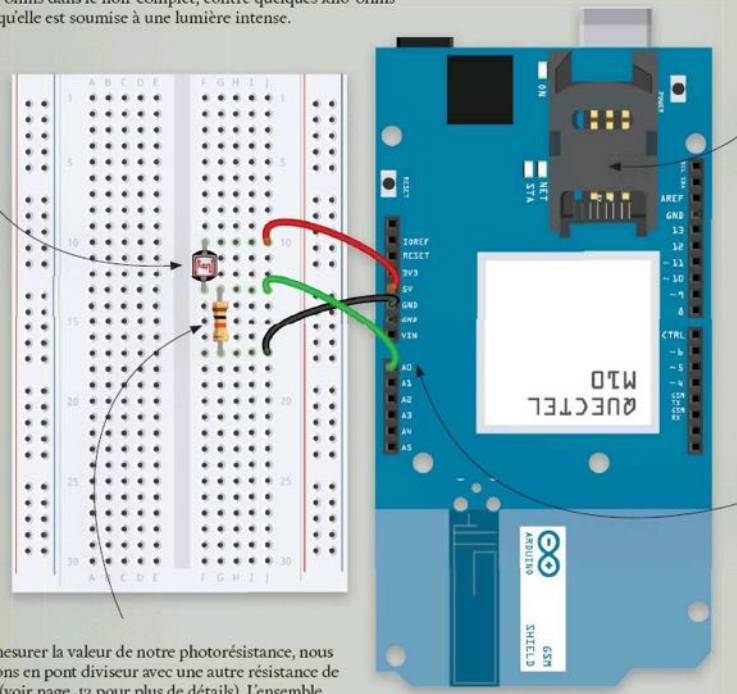
La condition lit la tension aux bornes de la photorésistance par l'entrée analogique. Cette dernière renvoie une valeur située entre 0 et 1023, en fonction de la tension (0 indique une tension de 0 V, 1023 une de 5 V). Nous avons choisi arbitrairement de déclencher la condition quand elle dépasse 512, mais nous vous conseillons d'effectuer quelques essais en fonction de votre environnement. En testant la photorésistance dans le noir complet et en illuminant peu à peu la scène, vous pourrez définir un seuil à partir duquel vous considérez qu'une alerte doit être envoyée.

L'envoi d'un SMS demande le numéro de téléphone de votre appareil et le message, 140 caractères en ASCII strict, sans accents. `sms.beginSMS()` lance la création d'un SMS, `sms.print()` permet de construire le message et `sms.endSMS()` l'envoie. Un délai de 10 secondes laisse le temps au texto de partir et empêche le programme de vous noyer (complètement) sous les messages. Toutefois, une fonction plus élaborée – avec un système de reset par exemple – reste indispensable pour un usage pratique.

Hardware

La photorésistance est un composant qui modifie sa résistivité en fonction de la luminosité ambiante. Plus elle reçoit de lumière, plus elle diminue. Un modèle classique offre par exemple une résistance de plusieurs centaines de kilo-ohms dans le noir complet, contre quelques kilo-ohms lorsqu'elle est soumise à une lumière intense.

Le Shield GSM se place sur l'Arduino, en se connectant directement aux broches. Il réplique celles-ci pour un usage avec d'autres composants. Il nécessite bien évidemment une carte SIM active, avec un forfait classique (pour envoyer des SMS) ou un abonnement avec de la data, si vous comptez vous connecter à Internet en 2G (GPRS).



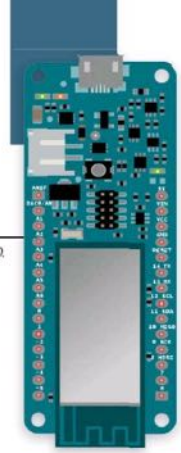
La photorésistance se relie à la roche A0 de l'Arduino, une entrée analogique. Elle va permettre de lire la tension aux bornes du composant, qui varie donc en fonction de la lumière. Une valeur faible indique qu'elle se trouve dans le noir complet, une valeur élevée qu'elle reçoit de la lumière (et, dans notre montage, que la porte a été ouverte).

Pour mesurer la valeur de notre photorésistance, nous l'utilisons en pont diviseur avec une autre résistance de 10 kΩ (voir page 13 pour plus de détails). L'ensemble est alimenté en 5 V. Dans le noir complet, on obtiendra donc une tension de 5 V * (10 kΩ / (10 kΩ + 200 kΩ)) soit ~ 0,23 V. À l'inverse, une fois éclairée, on obtient 5 V * (10 kΩ / (10 kΩ + 5 kΩ)) soit ~ 3,33 V.

LES LIMITES DES SHIELD GSM

Si l'idée de connecter un Arduino sans fil à Internet semble séduisante, rappelons que les Shield GSM posent énormément de limites. La majorité des modèles se contentent d'offrir une compatibilité avec les réseaux de deuxième génération : ils peuvent donc envoyer et recevoir des SMS, effectuer des appels (sur les modèles équipés de prises jack pour connecter un micro et un haut-parleur) et transmettre des données.

Sur ce point, la norme classique reste le GPRS – 85 kilobits/s en théorie, souvent nettement moins en pratique –, une connexion proche des antiques modems 56K. Point important pour les fans de Xavier Niel, Free Mobile ne dispose pas de son propre réseau 2G et utilise actuellement celui d'Orange, mais l'itinérance se termine en 2018 : ne comptez donc pas profiter d'un forfait à 2 euros pendant des années avec un Shield GSM.



Le schéma de l'Arduino MKR1000.

Le MKR1000 est beaucoup plus compact que le UNO.

L'Arduino MKR1000

WI-FI INSIDE

L'Arduino MKR1000 offre les fonctions d'un Arduino UNO dans un format compact, avec du Wi-Fi et – surtout – quelques différences pratiques. Penchons-nous sur ce modèle à succès.

Premièrement, l'Arduino MKR1000 contient un SoC basé sur un processeur ARM (Cortex M0+) mais ce changement n'a pas réellement d'impact visible pour l'utilisateur, les programmes fonctionnent de la même façon. Les deux principales différences viennent de l'intégration du Wi-Fi, avec une puce compatible 802.11n dans la bande des 2,4 GHz, et de la tension de fonctionnement des broches, qui passe à 3,3 V au lieu de 5 V. Ce changement nécessite quelques précautions, notamment si vous comptez utiliser des LED : les valeurs des résistances ne sont pas les mêmes qu'avec un Arduino UNO (par exemple) et une erreur peut être fatale pour le MKR1000. La carte possède aussi la possibilité d'être alimentée par une batterie externe de type Li-Po (attention à la capacité).

Connecter un MKR1000 au Wi-Fi. La fonctionnalité la plus intéressante du MKR1000 provient de sa puce Wi-Fi. Nous allons détailler rapidement la connexion à un réseau, en vous laissant ensuite le soin de trouver des usages innovants à un Arduino connecté.

Dans l'environnement de développement, la première étape consiste à installer le support du MKR1000. Allez dans **Outils → Type de carte → Gestionnaire de carte** et sélectionnez la prise en charge des cartes à base de Cortex M0. La seconde va être de récupérer les bibliothèques Wi-Fi. Le gestionnaire se trouve dans **Croquis → Inclure une bibliothèque → Gérer les bibliothèques**. L'Arduino MKR1000 utilise la bibliothèque WiFi101, présente aussi avec certains Shield.

```
#include <WiFi101.h>

char ssid[] = "Votre SSID";
char pass[] = "Votre mot de
passe";
int status = WL_IDLE_STATUS
const int LED=6;

void setup()
  {
  pinMode(LED, OUTPUT);

  while ( status != WL_CONNECTED)
    {
    status = WiFi.begin(ssid,
pass);
    delay(10000);
    }
  }

void loop()
  {
  if (status == WL_CONNECTED)
    {
    digitalWrite(LED,HIGH);
    delay(250);
    digitalWrite(LED,LOW);
    delay(250);
    }
  }
```

Le support des Arduino à base d'ARM doit être ajouté manuellement.

Le programme se connecte à un réseau et la LED de l'Arduino clignote en cas de succès. La première ligne intègre la bibliothèque Wi-Fi. Les deux suivantes contiennent les paramètres du réseau, la troisième définit une constante qui affiche les retours du pilote Wi-Fi et la dernière sélectionne la broche 6 comme LED, car le MKR1000 n'utilise pas la broche 13 comme les autres modèles. Dans le `setup`, la broche est activée en sortie et une boucle va connecter l'Arduino. Tant que le pilote ne renvoie pas `WL_CONNECTED` (qui indique que l'Arduino est connecté), la commande `WiFi.begin(ssid, pass)` va être exécutée toutes les 10 secondes. Cette dernière lance la connexion avec les paramètres fournis. Enfin, dans la boucle principale (`loop`), un bloc de code fait clignoter la LED tant que l'Arduino reste connecté au réseau Wi-Fi. Parmi les idées envisageables, la création d'un scanner Wi-Fi : la LED de l'Arduino clignoterait à une fréquence variant en fonction de la proximité du point d'accès.

Raspberry Pi

JETONS LES BASES !

Face à l'Arduino, le Raspberry Pi souffre de fonctionnalités électroniques limitées – en particulier à cause de l'absence d'ADC – mais se rattrape grâce à sa puissance de calcul bien plus importante. Ici, pas de C ni d'environnement logiciel tout intégré (IDE) : il vous faudra connaître les bases de Linux et programmer en Python. Rassurez-vous, ce n'est pas très compliqué ! Nous allons commencer par détailler deux opérations simples.

Un kit de démarrage de Raspberry Pi vendu sur Amazon.

D'abord, le Raspberry Pi seul ne suffit pas. Vous aurez besoin de quelques accessoires, souvent présents dans les kits vendus par différents magasins. Le minimum vital comprend une carte microSD de 8 Go rapide (*classe 10* si possible), un chargeur de qualité capable de fournir 2 A et un câble micro-USB. Évitez les cartes récupérées dans un vieux smartphone – elles sont souvent lentes – et ne lésinez pas sur le chargeur et le câble. Un modèle d'entrée de gamme ou mal conçu risque de faire planter votre carte sans raison visible, spécialement sur le Raspberry Pi 3, bien plus sensible car consommant nettement plus en charge. Petite astuce si vous utilisez un écran : le Raspberry Pi affiche un carré contenant les couleurs de l'arc-en-ciel dans un coin quand l'alimentation fournit une tension trop faible. La création de la carte microSD demande un lecteur (souvent présent sur les PC portables), les instructions se trouvent sur le site de la fondation (*www.raspberrypi.org*). Installez Raspbian directement ou en utilisant NOOBS.

LOCAL OU REMOTE ?

Pour écrire vos programmes, deux solutions : soit brancher directement un écran (en HDMI) et un clavier USB, soit passer par le réseau en vous connectant en SSH depuis une autre machine. Nous préférons la seconde solution, car elle permet de facilement copier/coller les lignes nécessaires depuis un appareil réactif. Toutefois, rien ne vous empêche de tout faire depuis la distribution Linux d'origine.
Nous vous conseillons de le brancher en Ethernet : régler un problème de connexion sera bien plus simple qu'en utilisant le Wi-Fi intégré dans certains modèles. Dans la suite, nous considérerons que vous êtes capable d'allumer le Raspberry et de vous connecter, ainsi que de taper quelques commandes. Nos montages ne nécessitent pas des connaissances pointues sous Linux et – dans le pire des cas – vous trouverez facilement des informations sur Internet.

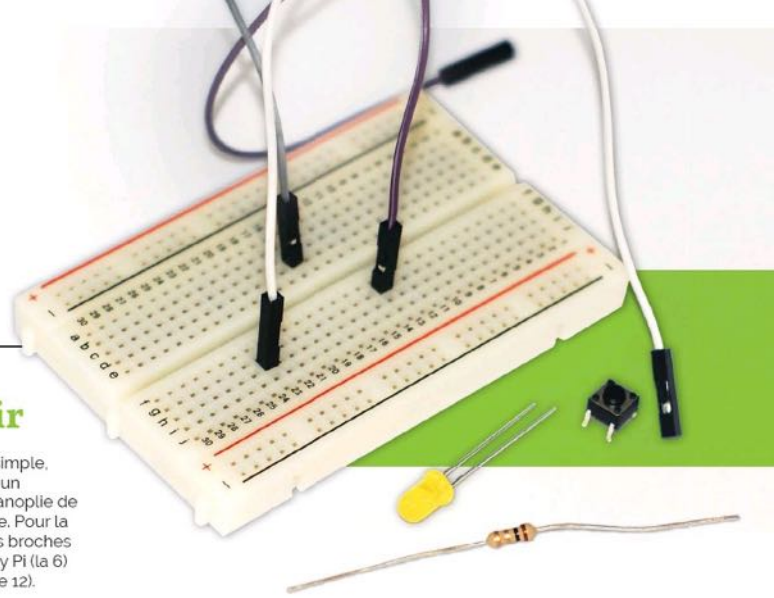
Si votre PC fonctionne sous Linux ou Mac OS X, le système propose un terminal qui offre la possibilité de vous connecter directement au Raspberry Pi. Seule l'adresse IP sera nécessaire : elle est affichée au démarrage (si vous avez branché un écran) et de nombreux programmes permettent de scanner le réseau pour détecter les appareils présents. La commande à taper est `ssh pi@192.168.1.2` (avec la bonne IP), le mot de passe par défaut `raspberry`. Sous Windows, vous pouvez utiliser Putty (*www.putty.org*) en utilisant l'adresse IP, le nom d'utilisateur (`pi`) et le mot de passe. La distribution Raspbian intègre le nécessaire pour programmer en Python.

HELLO WORLD

Pour éditer le fichier, nous utiliserons Nano. La commande `nano programme.py` crée un fichier vide, dans lequel vous pouvez écrire le code (l'extension .py indique que c'est du Python). Cet éditeur en mode texte reste plus simple que l'antique `vi` et les raccourcis permettant de sauver, quitter, etc. s'activent en pressant la touche Ctrl et une lettre, comme indiqué en bas de l'écran. Ctrl+x quitte, Ctrl + o sauvegarde, etc. Pour lancer le programme une fois le code terminé, tapez `python programme.py`. Pour le quitter, la méthode la plus simple (et expéditive) consiste à presser Ctrl + c ou à prévoir une condition de sortie dans votre code. Dans nos exemples, nous utilisons des boucles infinies dans certains cas, ce qui impose la première méthode.

Une fenêtre de connexion à un Raspberry Pi.

```
pi@NabzPi: ~ — -ssh -p 22 pi@NabzPi.local. — 80×24
pi@nabzpi.local.'s password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Oct 20 14:41:46 2016 from fe80::c5a:16a2:9beb:f215%eth0
pi@NabzPi ~ $
```



Brancher un bouton poussoir

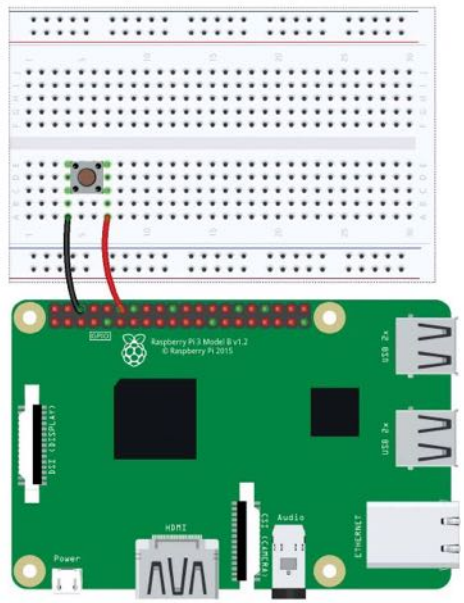
Pour notre premier exemple, très simple, vous allez brancher un bouton sur un Raspberry Pi. Ils font partie de la panoplie de base de tous les kits d'électronique. Pour la connexion, il suffit de relier une des broches du bouton à la masse du Raspberry Pi (la 6) et la seconde au GPIO 18 (la broche 12).

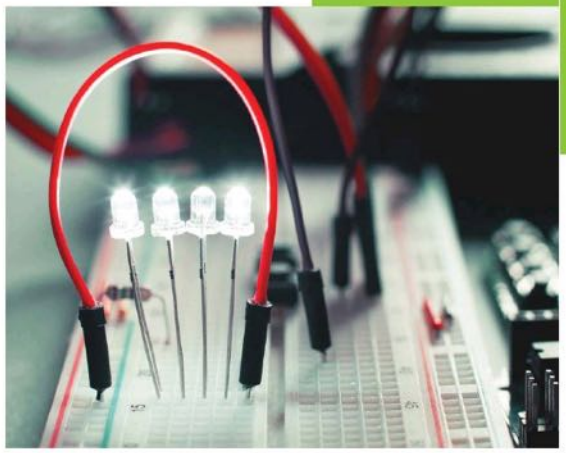
Le fonctionnement d'un bouton poussoir est simple : toutes les pattes ne sont reliées ensemble que quand il est pressé. Pour vérifier si c'est le cas, nous allons développer un petit programme en Python, un langage assez facile à appréhender.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    etat = GPIO.input(18)
    if etat == False:
        print('Bouton appuyé !')
        time.sleep(0.2)
```

Les deux premières lignes importent les bibliothèques qui contiennent les fonctions que nous allons utiliser. La bibliothèque `time` intervient sur les commandes liées à la gestion du temps et `RPi.GPIO` permet de gérer les GPIO. La troisième ligne définit le nom des GPIO : la valeur `GPIO.BCM` se base sur la numérotation interne des GPIO, alors que `GPIO.BOARD` utilise la numérotation physique. Nous vous conseillons le premier choix étant donné que la seconde peut changer en fonction des cartes. La ligne suivante configure le GPIO 18 comme une entrée et active la résistance *pullup* interne. Sans elle, vous ne pourrez rien mesurer vu qu'aucune tension ne serait présente sur la broche.

La suite est spécifique au Python. `while True:` lance une boucle infinie, que vous pourrez arrêter avec la commande Ctrl + c. Le langage se base sur l'indentation pour définir les blocs de code qui font partie de la boucle : tout ce qui sera décalé sera exécuté avant de revenir à la ligne de départ. Une variable `etat` va lire la valeur renvoyée par le GPIO 18 : quand le bouton n'est pas pressé, la masse et le GPIO 18 ne sont pas reliés. La tension sur l'entrée est de 3,3 V, due à la résistance *pullup* interne. Cet état est considéré comme *True*. Quand le bouton est pressé, à l'inverse, la tension sur la broche passe à 0 V et la valeur lue devient *False*. Le bloc qui suit le `if` (qui indique une condition) est indenté (décalé) et ne s'exécute que si la broche passe à *False*, donc que le bouton est activé. La commande `print` affiche un message et `time.sleep` met le programme en pause pendant 0,2 seconde, pour éviter que la pression ne soit détectée plusieurs fois.

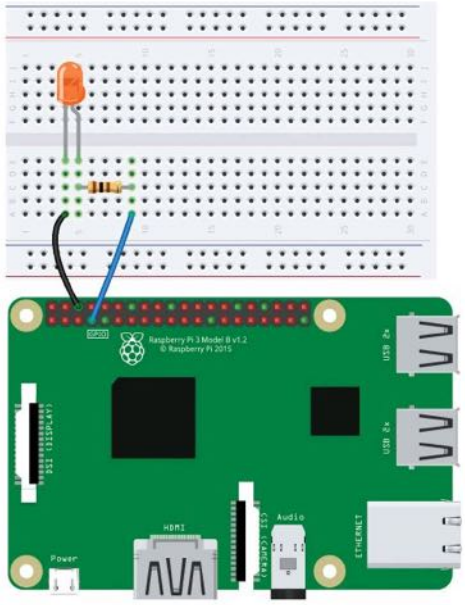




Faire clignoter une LED

Alimenter quelques vulgaires LED ne pose pas de problème sur un Arduino et ne nécessite aucun composant supplémentaire. Sur un Raspberry Pi en revanche, c'est un peu plus compliqué. La faute aux limitations inhérentes au SoC ARM, qui ne peut fournir autant de courant qu'un ATMega.

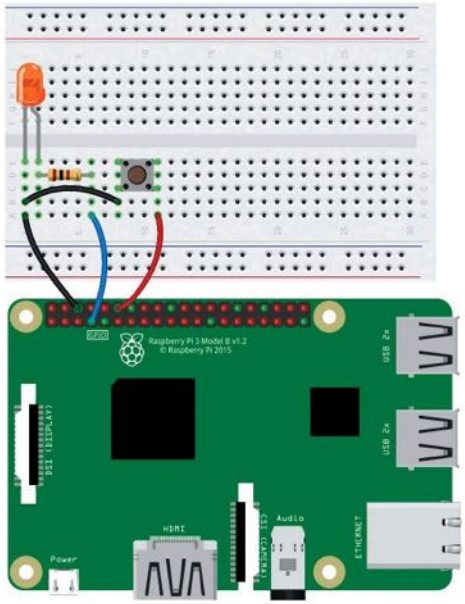
Si nous insistons sur ce point, c'est parce que les Raspberry peuvent facilement être détruits par un courant trop important. Ils ne bénéficient d'aucune protection interne et le moindre défaut sera immédiatement fatal. Le maximum que vous ne devriez jamais dépasser se situe à seulement 16 mA par broche, et 50 mA pour l'ensemble des GPIO de la carte. Tel quel, c'est insuffisant pour alimenter une simple LED rouge (qui exige généralement 20 mA) à sa pleine puissance. Il vous reste néanmoins plusieurs solutions. La plus élégante consiste à utiliser un transistor comme dans le montage décrit page 23. Vous pouvez également choisir une LED basse consommation : certaines se contentent de 10 mA, d'autres de 5 mA et parfois même de 2 mA. Si vous n'avez que des LED classiques sous la main, il convient d'utiliser une résistance correctement dimensionnée pour limiter le courant. Prenons l'exemple d'une LED orange : elle possède une tension de fonctionnement minimale de 2,1 V (V_f) et une intensité nominale de 20 mA (I_f). Heureusement, elle s'illuminera bien avant d'atteindre les 20 mA. Nous la limiterons donc à 15 mA. Pour connaître la résistance à appliquer, nous utilisons la loi d'Ohm (U = R * I), soit $(3{,}3\text{-}V_f) / 0{,}015 = 80\ \Omega$. Pour des questions de sécurité, nous utiliserons un modèle de 100 ohms, soit une charge sur la broche GPIO de 12 mA. Ne descendez pas sous la valeur calculée, sous peine de griller votre Raspberry Pi, mais n'allez pas non plus trop haut : si la résistance est trop élevée, la LED ne s'allumera pas.



Pour notre branchement, nous avons utilisé le GPIO 4. La patte la plus courte de la LED (la cathode, –) se relie à la masse, la patte la plus longue (anode, +) au GPIO qui lui fournira sa tension d'alimentation. La résistance peut se placer soit entre la broche du Raspberry Pi et l'anode, soit entre la masse et la cathode..

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(04, GPIO.OUT)

while (True):
    GPIO.output(04, True)
    time.sleep(0.5)
    GPIO.output(0, False)
    time.sleep(0.5)
```

Comme précédemment, le programme importe les bibliothèques nécessaires et paramètre les GPIO. Ici, la commande `GPIO.setup(04, GPIO.OUT)` configure le GPIO 4 comme une sortie. Dans la boucle, très simple, la commande `GPIO.output(04, True)` allume la LED et `GPIO.output(0, False)` l'éteint. Entre deux, une temporisation de 0,5 seconde crée un clignotement à une fréquence de 2 Hz.



Raspberry Pi 3 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO2 (SDA1 , I²C)		DC Power 5v	04
05	GPIO3 (SCL1 , I²C)		Ground	06
07	GPIO4 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO9 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO8	24
25	Ground		(SPI_CE1_N) GPIO7	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO5		Ground	30
31	GPIO6		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Le schéma des GPIO d'un Raspberry Pi 3

Allumer la LED quand le bouton est pressé

Nous allons maintenant combiner les deux programmes que nous venons de tester. Le montage ne pose aucun problème particulier : la masse du Raspberry Pi est reliée à la cathode de la LED ainsi qu'à une broche du bouton, le GPIO 18 (configuré en entrée) se connecte sur la seconde broche du bouton et le GPIO 4 (en sortie) à l'anode de la LED, avec bien évidemment la résistance entre les deux.

Vous trouverez ci-contre la liste de correspondances entre les broches de la carte et les GPIO internes. Le Raspberry Pi 3 comporte 26 GPIO mais seuls 17 ne sont pas multiplexés avec d'autres fonctions importantes (comme le bus SPI). Nous vous conseillons de les utiliser en priorité (en vert sur le schéma). À noter que les Raspberry Pi Zero utilisent le même *header*.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.
PUD_UP)
GPIO.setup(04, GPIO.OUT)

while True:
    etat = GPIO.input(18)
    if etat == False:
        GPIO.output(04, True)
        time.sleep(1)
        GPIO.output(04, False)
```

Le code n'a rien de compliqué : il contient l'importation des bibliothèques, la configuration des GPIO dans le bon mode et une boucle. Dans cette dernière, au lieu d'afficher un message à l'écran, nous allons allumer la LED pendant une seconde avant de l'éteindre. Cette étape est importante : la commande `time.sleep` place le programme en pause, elle n'éteint pas la LED automatiquement. Ce programme fournit les bases nécessaires pour comprendre une structure simple et effectuer des montages en utilisant les GPIO d'un Raspberry Pi. Dans les pages suivantes, nous allons nous intéresser à des périphériques un peu plus complets qu'une LED ou un bouton.

Sense HAT

THE SHIELD FROM OUTER SPACE

La fondation à l'origine de la carte propose un accessoire très complet pour ceux qui veulent jouer avec des capteurs sur le Raspberry PI. À l'origine, le Sense HAT est un module pensé pour une mission dans la Station spatiale internationale (ISS), sous le nom d'Astro Pi. Cette déclinaison en est simplement une version commerciale, qui s'adapte sur la majorité des Raspberry Pi à l'exception des premières générations.

Le Sense HAT dans l'ISS.

La carte mesure exactement la même taille que le Raspberry Pi A+ et offre beaucoup de fonctions. Au niveau pratique, elle intègre 64 LED RGB programmables, ce qui permet d'afficher des images de 8 x 8 pixels, ainsi qu'un petit joystick composé de cinq boutons (les quatre directions plus un interrupteur). Mais le Sense Hat embarque surtout plusieurs capteurs. Premièrement, on y trouve de quoi se positionner dans l'espace : un gyroscope, un accéléromètre et un magnétomètre.

Le Sense HAT utilise uniquement les GPIO 3 et 5 (cpc.cx/hpn).

En récupérant les informations de ces trois périphériques, il devient possible de définir assez précisément l'orientation du Raspberry Pi ainsi que ses mouvements. Deuxièmement, elle intègre aussi le nécessaire pour obtenir des données sur l'environnement direct : la température, la pression atmosphérique avec un baromètre numérique et l'humidité. Avec un peu de temps, se monter une pseudo-station météo devient possible facilement.

Une installation simple

Dans les pages suivantes, nous allons montrer comment utiliser l'écran et récupérer les informations des capteurs, mais la première chose à faire reste évidemment d'installer la carte. Bonne nouvelle, il s'agit d'une opération assez simple : il suffit de la placer sur le connecteur GPIO de seconde génération (à partir du B+) puis de la fixer avec les supports fournis. Le Sense HAT permet de continuer à utiliser les GPIO et se contente des broches 3 et 5 (bus I2C), ainsi que l'alimentation sur les broches 1 et 2. Si vos périphériques se basent sur d'autres GPIO, il ne devrait donc pas y avoir de problèmes. Le langage de programmation le plus utilisé sur le Raspberry Pi demeure le Python et la fondation fournit les bibliothèques compatibles pour exploiter la carte. La commande `sudo apt-get install sense-hat` (suivie d'un redémarrage) installe tout le nécessaire pour les exemples que nous allons détailler.

ET SUR LE RASPBERRY PI ZERO ?

Le Sense HAT fonctionne sur les Raspberry Pi équipés de la seconde génération de GPIO (à partir du B+) mais aussi sur le Raspberry Pi Zero, avec quelques modifications. En effet, la version la plus compacte de la carte n'offre pas les broches pour connecter des périphériques sur les GPIO, mais il demeure parfaitement possible de souder ces dernières directement sur le PCB, ce que nous avons fait pour nos tests.

EN RÉSUMÉ

Prix : ~40 €
Compatibilité : Raspberry Pi B+, A+, 2, 3.
Capteurs : magnétomètre, accéléromètre, gyroscope, température, pression atmosphérique, humidité.
Écran : LED RGB, 8x8.
Interface E/S : joystick

Gérer l'écran du Sense HAT

La fonction la plus visible du Sense HAT reste évidemment sa matrice de LED, placée sur le dessus de la carte. Nous allons donc passer en revue les possibilités de ce dernier.

Les 64 pixels de l'écran du Sense HAT.

L'écran du Sense HAT dispose de 64 LED dans un carré de 8 LED de côté. Elles peuvent afficher 32 nuances de rouge et de bleu ainsi que 64 nuances de vert (RGB 565) mais les API fournies utilisent des variables sur 8 bits (256 valeurs) converties ensuite en interne. Deux fonctions en Python permettent de manipuler du texte facilement sur l'écran : la première est `sense.show_message`, la seconde `sense.show_letter`.

Voici un exemple avec `sense.show_message` (sur une seule ligne).

```
sense.show_message("Canard PC Hardware", scroll_speed=0.05, text_colour=[255,255,0], back_colour=[0,0,255])
```

La valeur entre guillemets va s'afficher sur l'écran, mais attention : elle ne doit contenir que des caractères ASCII stricts, c'est-à-dire sans accents. La variable `scroll_speed` permet de gérer la vitesse du défilement, avec une valeur par défaut de 0,1. Plus elle est élevée, plus le texte bouge lentement. Les deux suivantes (`text_colour` et `back_colour`) définissent la couleur du texte et celle de l'arrière-plan. Les trois valeurs permettent de régler le rouge, le vert et le bleu : 255 représente l'intensité maximale, 0 la minimale. Un fond noir doit donc être indiqué par 0, 0, 0 et un blanc par 255, 255, 255. Passons maintenant à `sense.show_letter`. La commande affiche une lettre (toujours de l'ASCII strict), avec les mêmes paramètres pour les couleur (toujours sur une seule ligne).

```
sense.show_letter("C",text_colour=[0, 0, 0], back_colour=[255, 255, 255])
```

Vous trouverez en ligne (cpc.cx/hv3) un exemple de code qui utilise ces deux fonctions pour afficher du texte en Python et le faire défiler, avec une commande qui permet de choisir un nombre aléatoirement et définir l'intensité des couleurs.

Travailler en mode graphique.

La seconde manière de contrôler l'écran consiste à l'adresser directement en mode graphique, pixel par pixel. Cette méthode permet d'afficher n'importe quelle image, mais la définition – 8 x 8, rappelons-le – limite évidemment les possibilités. Pour sélectionner un pixel, il suffit d'utiliser la fonction `sense.set_pixel`, avec cinq paramètres :

```
sense.set_pixel(5, 5, [255, 0, 0])
```

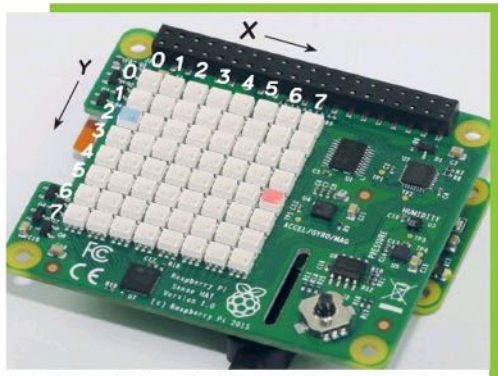
Les deux premiers chiffres indiquent la position absolue du pixel. Le couple 0,0 donne le pixel en haut à gauche, 7,7 celui en bas à droite. Les trois valeurs suivantes définissent la couleur, de la même façon que pour le texte. L'exemple ici affichera donc le sixième pixel de la sixième ligne en rouge.

Une dernière fonction, `sense.set_rotation`, modifie l'orientation de l'écran pour l'adapter à vos montages. Elle accepte quatre valeurs : 0, 90, 180 et 270.

Vous trouverez en ligne (cpc.cx/hv4) un autre exemple qui permet d'afficher facilement une image sans devoir adresser individuellement chaque pixel. Elle consiste à utiliser une matrice qui fait référence à une couleur définie précédemment. Le code qui suit utilise cette technique : a représente du noir, b du blanc et la matrice de 8 x 8 permet de charger les variables rapidement.

```
a = [0,0,0]
b = [255,255,255]

image = [
a,b,a,b,a,b,a,b,
b,a,b,a,b,a,b,a,
a,b,a,b,a,b,a,b,
b,a,b,a,b,a,b,a,
a,b,a,b,a,b,a,b,
b,a,b,a,b,a,b,a,
a,b,a,b,a,b,a,b,
b,a,b,a,b,a,b,a
]

sense.set_pixels(image)
```

Maintenant que nous avons appris à afficher du texte ou une image, intéressons-nous aux capteurs présents dans le Sense HAT.



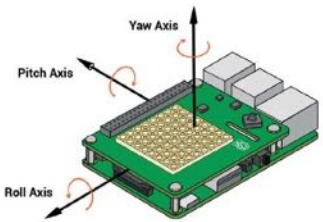
Le pixel bleu possède les coordonnées (0,2), le pixel rouge se place en (7,4).

Gérer les capteurs du Sense HAT

Le Sense HAT dispose de plusieurs capteurs et la lecture, toujours en Python, se trouve facilitée par les librairies bien conçues proposées par la fondation Raspberry Pi.

Les capteurs sont bien identifiés sur la carte.

Commençons par la température, la pression et l'humidité. La récupération des trois informations suit en effet la même logique et il existe une fonction dédiée dans chaque cas. La pression s'obtient avec la commande `sense.get_pressure()`, qui renvoie une valeur en millibar (typiquement aux alentours de 1 000 en temps normal). L'humidité passe par une commande similaire, `sense.get_humidity()`, avec un résultat exprimé en pourcentage. Enfin, le cas de la température est intéressant, à cause de plusieurs points. Premièrement, le Sense HAT ne possède pas un capteur dédié pour la température, mais utilise soit le capteur d'humidité, soit le capteur de pression. La commande par défaut (`sense.get_temperature()`) passe en fait par le premier, alors qu'une alternative (`sense.get_temperature_from_pressure()`) utilise le second. Pourquoi en parler ? Parce qu'ils n'indiquent pas le même résultat. Lors de nos tests, le capteur de pression indique une température systématiquement plus faible d'environ 1 °C. Plus problématique : le SoC peut polluer le résultat. Un Raspberry Pi 3 en pleine charge chauffe beaucoup et peut facilement fausser les mesures après quelques minutes.
Le petit programme que nous vous proposons va lire la température et l'afficher en rouge sur l'écran si elle dépasse une valeur fixée ici à 30 °C. La partie qui commence par `while` indique qu'il s'agit d'une boucle infinie, dans laquelle nous allons récupérer la température dans une variable t. Si cette dernière (arrondie) dépasse 30, un message avec un fond rouge (voir page précédente) s'affiche sur l'écran.



Les trois axes du Sense HAT.

```
from sense_hat import SenseHat
sense = SenseHat()

while True:
    t = sense.get_temperature()
    t = round(t, 1)
    if t > 30.0:
        msg = "Temperature = {0}".format(t)
        sense.show_message(msg, scroll_speed=0.05, back_colour=(100,0,0))
```

Les capteurs d'orientation.
Le Sense HAT propose trois capteurs d'orientation, un gyroscope, un accéléromètre et un magnétomètre. La manière la plus simple de les utiliser consiste à passer par la commande `sense.get_orientation()` qui se base sur les données des capteurs pour obtenir des informations lisibles par un humain. Elle affiche trois résultats, par rapport à trois axes. Le premier, le *pitch*, affiche l'angle d'orientation de la carte (en degrés) en fonction d'un axe virtuel qui la traverse dans le sens de la longueur. Le second, le *roll*, dans le sens de la largeur. Les deux affichent une valeur nulle quand elle se trouve parfaitement à plat. Le troisième, le *yaw*, représente l'orientation, sur un axe perpendiculaire à la carte. La seconde commande, `sense.get_accelerometer_raw()`, donne l'accélération (en g) de chaque axe, ce qui permet de détecter un mouvement.

Le petit programme suivant va afficher le message *Tilt !* quand un des capteurs détecte un mouvement brusque (une accélération de plus de 1 g), comme celui d'un lecteur qui secoue son Raspberry Pi. Nous n'utilisons pas l'axe z pour une raison simple : si la carte est placée à plat, elle mesure l'attraction terrestre.

```
from sense_hat import SenseHat

sense = SenseHat()

while True:
    acceleration = sense.get_accelerometer_raw()
    x = acceleration['x']
    y = acceleration['y']

    x = abs(x)
    y = abs(y)

    if x > 1 or y > 1:
        sense.show_message("Tilt !", scroll_speed=0.05, back_colour=(100,0,0))
    else:
        sense.clear()
```

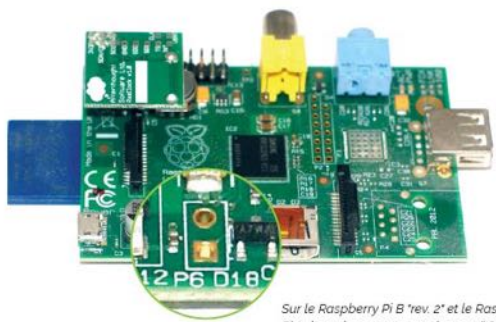
Avec les informations que nous vous avons proposées, vous devriez pouvoir tirer parti du Sense HAT et arriver à imaginer des programmes qui utilisent les différents capteurs de la carte. À noter que le joystick fonctionne par défaut comme un clavier classique avec les directions reliées aux flèches et le bouton central *mappé* comme une touche enter.

Installer un bouton reset sur le Raspberry Pi

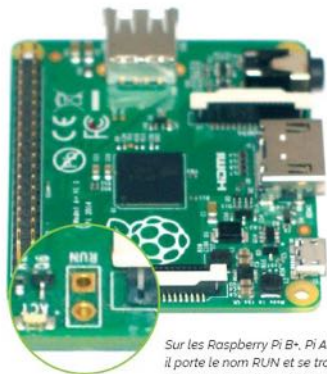
Depuis la sortie du Raspberry Pi, une fonction fait cruellement défaut : un bouton pour éteindre ou relancer la carte sans débrancher l'alimentation. En réalité, elle existe, mais n'est pas câblée par défaut.

Tous les modèles de Raspberry Pi, à l'exception de la première génération (le B avec 256 Mo de RAM) disposent d'un emplacement sur la carte mère pour installer un bouton *reset*. Il change de place selon les cartes, mais sa fonction reste la même : réinitialiser le SoC en cas de souci.
Le branchement d'un bouton *reset* consiste simplement à connecter un bouton poussoir, par exemple récupéré dans une carcasse de PC, à deux trous préexistants. Il suffit souvent de souder deux broches sur celles-ci. Attention : le *reset* redémarre totalement le Raspberry Pi, sans éteindre correctement le système d'exploitation. Nous vous déconseillons donc de l'utiliser sur un appareil en fonctionnement, sous peine de corrompre les données présentes sur la carte SD. Ce bouton trouvera surtout son sens pour redémarrer un Raspberry Pi planté ou arrêté manuellement sans devoir débrancher l'alimentation.

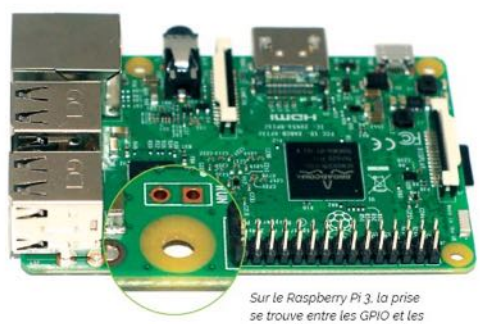
Un bouton reset issu d'un PC monté sur le Raspberry Pi 3.



Sur le Raspberry Pi B "rev. 2" et le Raspberry Pi A, l'emplacement porte le nom P6 et se trouve à gauche de la prise HDMI.



Sur les Raspberry Pi B+, Pi A+ et Pi 2, il porte le nom RUN et se trouve entre les GPIO et la prise pour la caméra.



Sur le Raspberry Pi 3, la prise se trouve entre les GPIO et les ports USB, sous le nom RUN.

Sur le Raspberry Pi Zero, le header RUN se trouve au niveau des GPIO.

PiFace Control and Display 2

L'ÉCRAN À TOUT FAIRE

Le PiFace *Control and Display 2* offre trois choses intéressantes pour les utilisateurs de Raspberry Pi : un écran, un récepteur infrarouge et quelques boutons.

Le PiFace Control and Display sur un Raspberry Pi 3.

La société PiFace propose plusieurs cartes différentes (Digital, Rack, etc.), mais nous allons nous attarder sur la *Control and Display* (en version 2). Imaginée dès les débuts du Raspberry Pi, elle ne reprend pas le design des HAT habituels. Alors que ces derniers peuvent s'intégrer facilement et mesurent exactement la même taille qu'un Raspberry Pi B+, le Control and Display dépasse un peu sur les côtés et ne dispose pas de trous pour se fixer à la carte.

Un écran, des boutons. La carte offre trois fonctions principales. La première, la plus visible, demeure évidemment l'écran LCD. Il s'agit d'un modèle alphanumérique monochrome capable d'afficher deux lignes de seize caractères. Il dispose d'un rétroéclairage et, bien qu'il ne propose pas d'accès direct en mode graphique, les API permettent tout de même de définir des caractères personnalisés. Le PiFace C&D intègre également cinq boutons poussoirs ainsi qu'une sorte de sélecteur à trois positions. Son fonctionnement est un peu particulier : il doit être pressé physiquement au centre (avec un retour sensible) alors que les deux directions (gauche et droite) s'activent avec le mouvement, sans pression. Enfin, le périphérique intègre aussi un récepteur infrarouge compatible avec LIRC (un logiciel souvent utilisé sous Linux pour prendre en charge les télécommandes). Nous allons expliquer dans la suite comment gérer facilement les trois fonctions dans vos programmes.

L'activation du SPI devient nécessaire.

Mise en route. L'installation du PiFace *Control and Display 2* ne devrait pas poser de souci : la carte se branche sur les connecteurs GPIO de n'importe quel Raspberry Pi, du premier modèle de 2012 au Raspberry Pi 3. Attention, il existe une version "1" qui ne s'adapte pas bien aux Raspberry Pi B+ (et suivants) mais qui offre exactement les mêmes fonctions. Le PiFace C&D se pilote lui aussi en Python. L'installation logicielle se limite à entrer une seule ligne de commande : `sudo apt-get install python{,3}-pifacecad`. Une fois les bibliothèques installées, l'activation du lien SPI avec la commande `sudo raspi-config` s'imposera dans certains cas. Une fois l'utilitaire lancé, rendez-vous dans **Advanced Options** puis dans **SPI** pour vérifier que la technologie est activée. Enfin, attention à une chose : le PiFace C&D ne réplique pas les connecteurs GPIO et ne permet donc pas l'utilisation d'un autre périphérique.

La vis de réglage du contraste.

LE RÉGLAGE DU CONTRASTE

Une minuscule vis placée au-dessus de l'écran LCD permet de régler le contraste. Un tournevis Philips #0 ou #00 est évidemment fortement recommandé au vu de sa taille. N'espérez pas de miracles : l'écran reste assez moyen, du niveau de celui des téléphones portables du début des années 2000.

EN RÉSUMÉ

Prix : ~35 €
Compatibilité : tous les modèles
Capteurs : récepteur IR
Écran : LCD monochrome, 16 caractères, 2 lignes
Interface E/S : cinq boutons, switch 3 positions

L'écran avec un smiley personnalisé.

Gérer l'écran du Control and Display

La carte de PiFace dispose d'un écran LCD monochrome, suffisant pour donner des informations à l'utilisateur sur ce qui se passe dans vos programmes.

Commençons par la partie alphanumérique. L'écran peut afficher deux lignes de 16 caractères, en monochrome et avec un rétroéclairage. Première chose à prendre en compte : oubliez les accents. L'API ne supporte que l'ASCII strict, composé des caractères suivants.

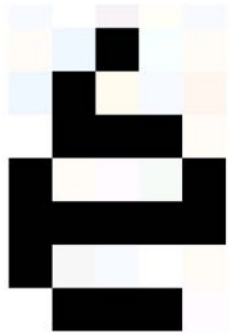
```
ABCDEFGHIJKLMNOPQRSTUVWXYZabc
defghijklmnopqrstuvwxyz0123456789
!"#$%&'()*+,-./:;>=<?@[\]^_`{}|~
```

Dans un programme en Python, il suffit d'importer les bibliothèques nécessaires (`import pifacecad`), d'activer l'écran (`cad = pifacecad.PiFaceCAD()`) et d'écrire du texte (`cad.lcd.write("Bonjour monde")`). Dans l'exemple qui suit, nous activons le rétroéclairage de l'écran (`cad.lcd.backlight_on()`) et coupons l'affichage du curseur (`cad.lcd.blink_off()` et `cad.lcd.cursor_off()`), après avoir effacé l'écran (`cad.lcd.clear()`).

```
import pifacecad

cad = pifacecad.PiFaceCAD()
cad.lcd.clear()
cad.lcd.backlight_on()
cad.lcd.cursor_off()
cad.lcd.blink_off()
cad.lcd.write("Bonjour
Monde !")
```

Pour passer à la ligne, vous disposez de deux possibilités. Soit utiliser les caractères dédiés en ASCII (`\n`) soit positionner le curseur manuellement. La commande `cad.lcd.set_cursor(4, 1)` le place dans la cinquième colonne de la seconde ligne. N'oubliez pas qu'en informatique les valeurs commencent à 0 : la position `(0,0)` représente le coin supérieur gauche, la position `(7,1)` l'inférieur droit. Les deux dernières commandes pratiques pour du texte sont `cad.lcd.move_left()` et `cad.lcd.move_left()`. Elles déplacent le texte à gauche ou à droite d'un caractère. En utilisant Python, il devient donc possible de faire défiler du texte.

Les caractères personnalisés. Si l'écran ne propose pas de mode graphique à proprement parler, une solution alternative existe : la création de caractères personnalisés. Notre exemple montre la création du "e" avec un accent aigu. Un caractère est composé de 40 pixels (5 x 8), chaque pixel ayant une valeur 1 (noir) ou 0 (blanc) et la structure du programme montre bien la création de la lettre. La commande `cad.lcd.store_custom_bitmap(0,eaigu)` va stocker le caractère en position 0 et la commande `cad.lcd.write_custom_bitmap(0)` va l'afficher. Le PiFace peut contenir 8 caractères personnalisés (numérotés de 0 à 7) et il les garde en mémoire tant que la carte est alimentée.



```
import pifacecad

cad = pifacecad.PiFaceCAD()
eaigu = pifacecad.LCDBitmap([
0b00000,
0b00100,
0b01000,
0b01110,
0b10001,
0b11111,
0b10000,
0b01110])
cad.lcd.store_custom_bitmap(0,
eaigu)
cad.lcd.write_custom_bitmap(0)
```

Un message sur deux lignes

Les boutons du PiFace Control and Display

Comme expliqué précédemment, la carte dispose de huit boutons, utilisables pour interagir avec vos programmes.

Les boutons de la carte, cinq devant et un switch à l'arrière.

La façon la plus simple de lire les données des boutons consiste simplement à aller vérifier leur valeur. La commande `cad.switches[3].value` renvoie 1 quand le quatrième bouton est pressé et 0 le reste du temps. Les cinq premiers interrupteurs sont numérotés de 0 à 4, le centre du switch porte le numéro 5, et les deux directions (gauche et droite) sont nommées 6 et 7.

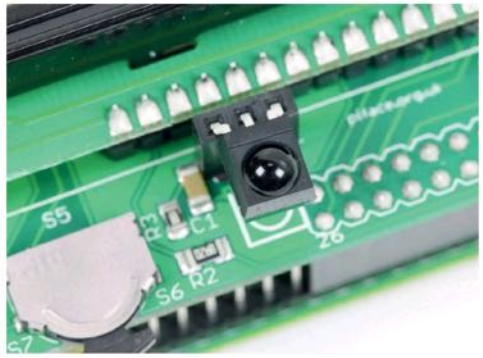
Le code suivant effectue une boucle qui attend une pression sur un bouton :

```
import pifacecad
from time import sleep

cad = pifacecad.PiFaceCAD()
cad.lcd.backlight_on()
cad.lcd.write("Canard PC Hardware !")
cad.lcd.cursor_off()
cad.lcd.blink_off()

while cad.switches[5].value == 0:
    if cad.switches[6].value == 1:
        cad.lcd.move_left()
        sleep(0.5)
    if cad.switches[7].value == 1:
        cad.lcd.move_right()
        sleep(0.5)
cad.lcd.clear()
```

Une méthode à base d'interruptions, plus propre, existe (elle est détaillée sur ce site : *cpc.cx/hqh*). La ligne `while cad.switches[5].value == 0:` exécute le programme tant que le bouton 5 (le switch central) n'est pas pressé. Les deux autres lignes font défiler le texte à gauche ou à droite en fonction des boutons. Le `sleep(0.5)` sert à induire une pause pour un défilement lisible. Attention à l'indentation (les espaces avant les lignes) ; le Python se base dessus pour séparer les différents blocs de code.

Le récepteur infrarouge

La carte intègre un récepteur infrarouge qui va permettre d'utiliser une télécommande comme source d'entrée.

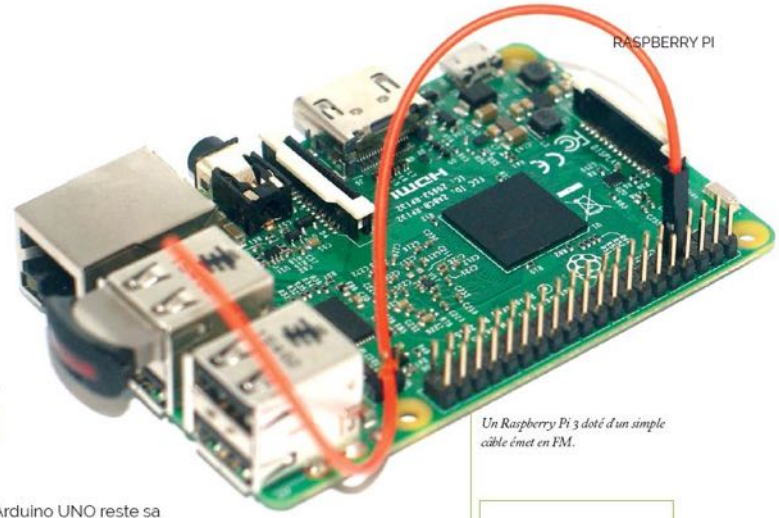


Le récepteur infrarouge permet l'utilisation d'une télécommande.

S'il est possible d'utiliser une télécommande dans un programme en Python, le récepteur sert surtout dans un cas précis : commander un Media Center. Bonne nouvelle, le PiFace C&D fonctionne avec LIRC, la référence sous Linux. Nous vous proposons ici l'installation de base pour utiliser une télécommande, ensuite libre à vous de chercher des informations pour une télécommande en particulier. Première étape, installer LIRC et activer le récepteur. Un script effectue toute les manipulations.

```
wget https://raw.github.com/piface/pifacecad/
master/bin/setup_pifacecad_lirc.sh
chmod +x setup_pifacecad_lirc.sh
sudo ./setup_pifacecad_lirc.sh
```

Après un redémarrage, vous pourrez installer un fichier de configuration pour la télécommande (*cpc.cx/hqk*). Normalement, les principaux Media Center devraient reconnaître directement la télécommande, et au pire les instructions détaillées restent disponibles en ligne (*cpc.cx/hql*).



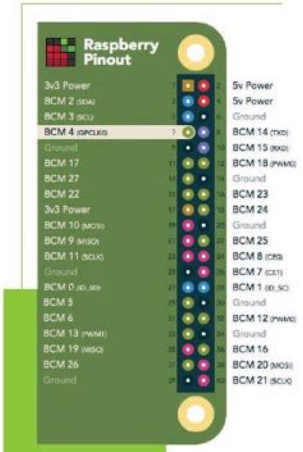
Un Raspberry Pi 3 doté d'un simple câble émet en FM.

Émettre en FM

avec un Raspberry Pi

L'intérêt du Raspberry Pi face à l'Arduino UNO reste sa puissance de calcul bien supérieure. Le SoC ARM est tellement performant qu'il peut même transformer un GPIO en émetteur FM ! Cette fonction alternative nécessite peu de composants et offre un résultat très audible.

Sur le Raspberry Pi, le GPIO 4 permet d'émettre un signal d'horloge (GPCLK0) et des bidouilleurs ont rapidement découvert qu'il était possible d'utiliser cette fréquence comme source d'émission. Si par défaut, les GPIO se basent sur une horloge à 19,2 MHz, la carte propose d'autres sources : 216 MHz, 500 MHz et même 1 GHz (qui dépend de la fréquence du CPU, donc qui peut varier en cas d'overclocking). En utilisant la fréquence de 500 MHz, une simple division permet d'émettre un signal à 100 MHz, au milieu de la bande FM. Ensuite, un vulgaire fil de cuivre de la bonne longueur placé sur le GPIO 4 permet d'émettre un signal FM avec une portée de quelques dizaines de mètres. Pour la longueur du câble, il faut prendre en compte la fréquence et effectuer le calcul suivant pour obtenir la taille idéale en cm : `((300/fréquence)/16)*100`. Pour une fréquence de 107,9 MHz, un câble de ~17,5 cm aura donc la longueur idéale.



La seule broche nécessaire est le GPIO 4.

L'installation. Au niveau matériel, un câble isolé de la bonne longueur suffit pour émettre vers un récepteur radio classique, comme un radioréveil ou un smartphone. Au niveau du logiciel, plusieurs programmes existent, mais nous avons une préférence : Pi-FM-RDS (*cpc.cx/hra*). Ce *fork* français ajoute en effet le support du RDS, ce qui permet d'afficher quelques informations sur les récepteurs compatibles. L'installation ne demande que quelques lignes pour commencer à émettre.

```
git clone https://github.com/
ChristopheJacquet/PiFmRds.git
cd PiFmRds/src
make clean
make
```

Une fois le programme compilé, il suffit de le lancer avec un fichier audio en paramètres et votre émission de radio personnalisée peut commencer.

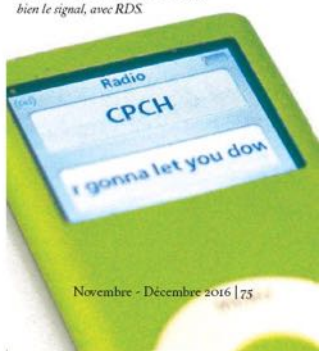
```
sudo ./pi_fm_rds -audio sound.wav
```

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

LA LÉGISLATION

Si la loi française a longtemps interdit l'émission personnelle dans la bande FM, la Décision n° 2014-1263 du 6 novembre 2014 (*cpc.cx/hrg*) l'autorise tant que la puissance d'émission ne dépasse pas 50 mW. Nous vous conseillons tout de même d'utiliser la fréquence 107,9 MHz, car elle ne correspond à aucune station officielle.

Un iPod équipé d'un tuner FM reçoit bien le signal, avec RDS.



Installer un RTC dans un Raspberry Pi

Une RTC (*Real Time Clock*) fournit une base de temps fiable et permet de conserver l'heure et la date dans une mémoire distincte, même en cas de coupure de l'alimentation. Les cartes Raspberry Pi n'intègrent pas ce composant qui semble pourtant standard dans le monde de l'informatique depuis les années 1980. Et parfois, il faut pallier ce manque.

Les cartes Raspberry Pi n'embarquent pas de RTC par défaut pour plusieurs raisons. La première demeure extrêmement pragmatique : le circuit coûte environ un dollar ; or, le but du Raspberry Pi est de diminuer autant que possible le prix du matériel. La seconde vient d'une contrainte physique : une puce RTC doit être alimentée en permanence pour continuer à mesurer le temps, ce qui nécessite une source d'énergie dédiée. Dans les PC, il s'agit généralement d'une pile bouton type CR2032, qui permet aussi de garder en mémoire les paramètres du BIOS. L'intégrer dans le Raspberry Pi était inenvisageable, autant pour des questions de volume que de coût. De plus, la présence d'une pile au lithium complique la logistique liée au transport aérien.

Le PiFace RTC sur un Raspberry Pi 3.

La gestion de l'heure et ses contraintes. Vous vous demandez peut-être comment le Raspberry Pi reste tout de même généralement à l'heure ? Simplement en utilisant un serveur NTP (*Network Time Protocol*) qui fournit l'heure à travers une connexion à Internet. L'OS de la carte, à chaque démarrage, va donc récupérer la date et l'heure et les mettre à jour périodiquement. Un Raspberry Pi qui n'est pas connecté n'a pas la possibilité de le faire et c'est là que les puces RTC entrent en jeu. Les accessoires se branchent sur les GPIO et quelques manipulations permettent ensuite de définir l'heure et – surtout – de la garder même quand le Raspberry Pi s'éteint. Certains montages nécessitent une activation à une heure précise et le RTC devient vite indispensable pour éviter les surprises suite à un redémarrage inattendu par exemple.

Le RasClock se connecte sur les broches GPIO.

La précision et le choix du module. Il existe énormément de modèles de RTC : des montages à base de DS1302 (à éviter, surtout s'ils viennent de Chine ; voir encadré) à des choses plus complexes qui reposent sur des puces très précises. La documentation vous indiquera parfois une valeur en ppm (*parts per million*), qui peut être convertie pour obtenir la dérive moyenne (la différence entre la base de temps et la réalité) sur une durée définie. Une précision de 100 ppm provoque un décalage moyen de 10^{-4} seconde (ou 0,0001 seconde). Sur une journée (86 400 secondes), la différence peut donc atteindre 8,64 secondes, ou 4 minutes et 20 secondes sur un mois. Pour nos tests, nous avons utilisé un PiFace *Real Time Clock* et un *RasClock*. Le premier intègre une puce MCP7940N de Microchip dont la précision moyenne n'est pas annoncée, alors que le second se base sur un NXP PCF212x (selon la version), avec une précision sous les 3 ppm (moins de 2 minutes de décalage par an). Dans les deux cas, les cartes valent environ 10 euros mais le RasClock est livré avec une pile CR1220, absente chez PiFace. Les modules moins onéreux (environ 5 euros) utilisent généralement des puces avec une précision de l'ordre de 50 ppm, par exemple un DS13024. Sous ce prix, ne craquez pas.

ATTENTION À LA QUALITÉ

Petite astuce qui vous évitera bien des prises de tête : n'achetez pas un module à bas prix sur eBay, Aliexpress ou d'autres. Les quartz utilisés dans ces derniers manquent de précision et dérivent nettement plus qu'un modèle un peu plus cher vendu par une société qui a pignon sur rue. Un décalage de plusieurs minutes par jour peut devenir handicapant dans certains cas, surtout s'il n'est pas possible de resynchroniser l'heure. Dans les pages qui suivent, un des montages se base par exemple sur le lancement d'un script deux minutes avant un événement : avec une puce RTC peu précise, le risque de rater ce moment sera élevé.

EN RÉSUMÉ

Prix : ~10 €
Compatibilité : tous les modèles
Connexion : GPIO
Alimentation : pile bouton

L'installation du PiFace RTC

Le PiFace RTC s'installe sur les broches GPIO et utilise le bus I2C. Sa conception lui permet de s'intercaler entre le Raspberry Pi et un éventuel périphérique. Elle dispose en effet de simples trous et s'insère "à travers" les broches d'un Raspberry Pi (sauf évidemment sur le Zero qui n'en offre pas).

Le RasClock sur un Raspberry Pi 3.

S'ils ne sont pas alignés correctement, c'est normal : le décalage de certains emplacements permet de forcer le contact sur les GPIO 3 et 5, utilisés par la carte. Cette façon de connecter le RTC peut sembler ingénieuse, mais elle a le défaut de placer la pile au niveau du SoC du Raspberry Pi et d'empêcher l'utilisation d'un radiateur. Sur un Raspberry Pi 3 qui chauffe énormément, nous vous déconseillons d'installer cette carte si vos programmes effectuent beaucoup de calculs.

Une fois la carte installée, vous devrez activer le bus I2C avec la commande `sudo raspi-config` (dans **Advanced Options -> I2C**), récupérer les outils nécessaires avec `sudo apt-get install i2c-tools` et enfin obtenir les scripts qui permettent l'utilisation de la puce RTC.

```
wget https://raw.github.com/piface/PiFace-Real-Time-Clock/master/install-piface-real-time-clock.sh
chmod +x install-piface-real-time-clock.sh
sudo ./install-piface-real-time-clock.sh
```

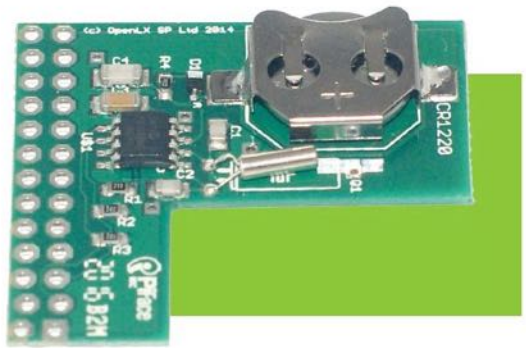
Après un redémarrage, la puce RTC fonctionnera. La dernière étape consiste à sauver la date. Un Raspberry Pi connecté à Internet au moment de la configuration devrait proposer une valeur correcte, mais vous pouvez la fixer manuellement avec la commande `date`. La suivante (`sudo hwclock -systohc`) copie la date système dans la puce, alors que le script lancé précédemment récupère automatiquement ce que contient le composant RTC au démarrage.

Alternative : Le RasClock

Le RasClock utilise un connecteur plus classique : il se branche sur les six premières broches de n'importe quel Raspberry Pi – sauf le Zero – et bloque par conséquent les autres cartes d'extension (HAT, écrans, etc.). Il demande plus de place en hauteur, mais a l'avantage de ne pas se positionner directement au-dessus du SoC. Avec une distribution Raspbian à jour, les manipulations seront peu nombreuses malgré une installation assez artisanale. Si vous devez utiliser une version plus ancienne, les instructions nécessaires existent sur le site des développeurs de la carte (*cpc.cx/hqK*). Première étape : ajouter le support du RTC en éditant le fichier de configuration (`sudo nano /boot/config.txt`) puis ajouter `dtoverlay=i2c-rtc,pcf2127` à la fin du fichier. Modifiez ensuite quelques lignes dans le fichier `/lib/udev/hwclock-set`. Faites précéder les lignes suivantes d'un #, pour les commenter :

```
if [ -e /run/systemd/system ] ; then
    exit 0
fi

/sbin/hwclock --rtc=$dev --systz -badyear

/sbin/hwclock --rtc=$dev -systz
```

Après un redémarrage, la commande `sudo hwclock` vous permettra de lire (avec `-r`) ou d'écrire (`-w`) dans la mémoire du RTC.



Les trous ne sont pas alignés pour forcer le contact.

LA DURÉE DE VIE

Les modules RTC que nous avons choisis utilisent une pile CR1220 avec une durée de vie annoncée de plus de deux ans quand le Raspberry Pi n'est pas alimenté. Si la carte est sous tension – *a priori* la majorité du temps –, elle ne sera donc pas utilisée.

Lire les données d'un compteur électrique

HACKEZ VOTRE LINKY !

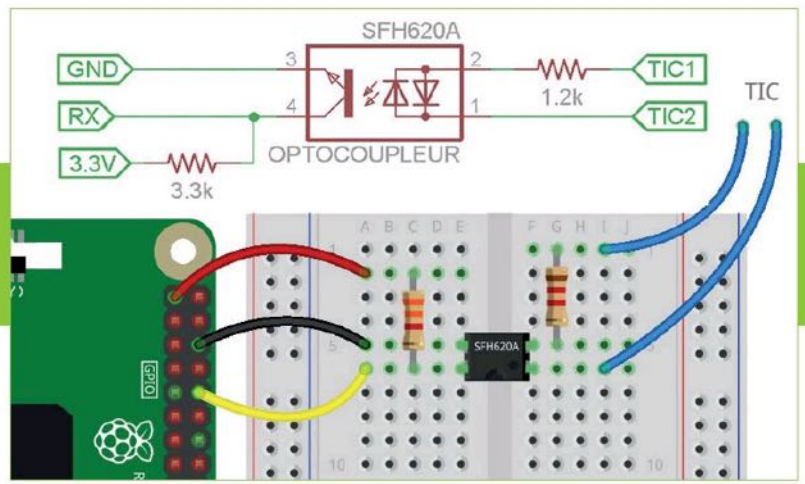
Si vous êtes bricoleur – avec ce magazine en main, nous n'en doutons pas –, la possibilité de lire votre consommation électrique en temps réel depuis votre compteur devrait vous intéresser. Et ce n'est pas très compliqué.

Depuis environ vingt ans, tous les nouveaux compteurs électriques installés chez les particuliers permettent de lire certaines informations comme la consommation, le type d'abonnement, le type d'appareil, etc. La téléinformation (TIC) sert à l'origine aux partenaires d'EDF pour offrir des boîtiers capables de mettre en forme les données et de les afficher, ce que nous allons faire ici. Les sorties sont librement accessibles sur le compteur. Attention : celles-ci ne doivent pas être confondues avec les broches liées au télé-relevé (qui permet à un agent de la société de récupérer votre consommation sans entrer dans votre domicile). Heureusement, vous ne devriez pas faire d'erreur : ces dernières se situent sous un cache plombé, elles.

Un hic dans la TIC. D'un point de vue technique, la TIC fournit un signal numérique encodé en modulation d'amplitude sur une porteuse à 50 kHz. Rassurez-vous, ce n'est pas aussi compliqué que ça en a l'air. Imaginez un signal sinusoïdal à 50 kHz. Si l'amplitude à un moment *t* est supérieure à 1,5 V, vous lirez un "1" logique. Et si, à moment *t+1*, elle est inférieure, alors ce sera un "0". Ce mode de fonctionnement original a été choisi par ERDF/Enedis pour permettre l'utilisation de fils jusqu'à 500 mètres entre le compteur et le récepteur. Coup de bol : la fréquence est suffisamment élevée pour que les GPIO d'un microcontrôleur interprètent directement cette modulation comme un simple signal numérique classique.

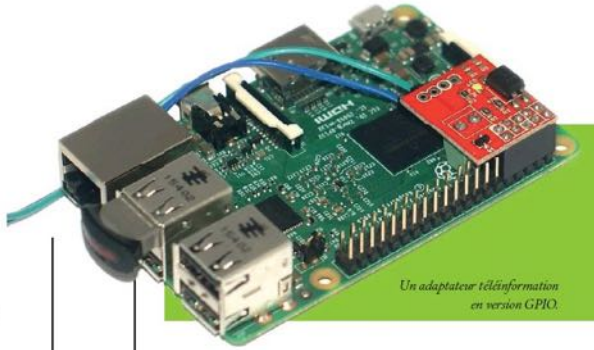
Les prises I1 et I2 du compteur qui permettent de récupérer la téléinformation.

Pour relier le compteur à votre Raspberry Pi, peu de composants sont nécessaires : un optocoupleur (SFH620A par exemple), deux résistances (1,2 kΩ et 3,3 kΩ) et quelques fils. Pour rappel, un optocoupleur sert à isoler deux circuits tout en permettant une communication entre eux. Il est constitué d'un phototransistor d'un côté et d'une ou deux LED de l'autre. Pour des raisons de sécurité, l'optocoupleur est incontournable pour un branchement sur un compteur électrique. Et attention lors du branchement : prenez un luxe de précaution car les fils sous tension ne sont pas bien loin ! En sortie, le flux d'impulsions est transmis directement à la broche RX (réception) de l'UART (port série) du Raspberry Pi. Les anciens compteurs n'offrent un débit que de 1 200 bps alors que les Linky peuvent atteindre 9 600 bps. Un gros progrès…



Les solutions alternatives

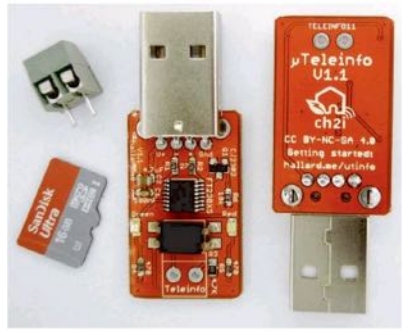
Si vous n'avez pas envie de jouer avec des résistances et des optocoupleurs, il existe quelques alternatives grâce à des adaptateurs prêts à l'emploi, disponibles sur Internet.


Un adaptateur téléinformation en version GPIO.

La première, que nous avons utilisée avec succès pendant plusieurs mois, est le PiTInfo (*cpc.cx/hru*). Ce module reprend le design de l'adaptateur présenté à la page précédente, mais dans un format compact qui trouve directement sa place sur les broches GPIO d'un Raspberry Pi. Une fois branché, l'OS devrait le détecter comme un port série – sous le nom `/dev/ttyAMA0` dans notre cas – et fournir exactement les mêmes informations qu'un module maison. Pour environ 15 euros, il s'agit d'une bonne solution toute faite pour récupérer les données de téléinformation sur un Raspberry Pi.

En USB ou en série. La seconde solution, du même concepteur que la première, y ajoute un adaptateur série vers USB. L'avantage de ce modèle vendu 25 euros en ligne (*cpc.cx/hrv*) vient du fait que l'USB offre évidemment plus de souplesse dans le choix de la machine que les broches GPIO. Le Micro TeleInfo utilise un composant FT230X signé FTDI, bien supporté en général sous Linux. Il fonctionne sur un Raspberry Pi mais aussi sur n'importe quelle carte de développement qui propose un port USB et un système d'exploitation basé sur une distribution Linux (la majorité). Vous trouverez dans le commerce d'autres modèles équipés d'un port USB, généralement pour un prix compris entre 20 et 50 euros environ. Attention à un point : certains vieux périphériques utilisent une prise série de type RS-232, ce qui nécessite un adaptateur dédié pour un branchement sur un Raspberry Pi ou un PC. De même, certains montages un peu anciens visent les PC des années 1990 (souvent équipés d'une prise série RS-232), et intègrent donc des composants pour gérer les tensions élevées attendues (-12 V et +12 V). Ils sont inutiles avec un Raspberry Pi ou un Arduino, ces deux appareils travaillent généralement en 3,3 V (Raspberry Pi) ou en 5 V (Arduino).

Le cas du Linky. Le compteur Linky a fait abondamment parler de lui, comme vous avez pu le lire dans *Canard PC Hardware* n° 28. Mais le compteur moderne a l'avantage de proposer exactement la même technologie que les anciens (avec toutefois un mode amélioré en supplément). Sous le capot, la boîte jaune offre un connecteur destiné à recevoir un module sans fil en technologie ZigBee, mais il peut parfaitement s'utiliser pour récupérer les données de téléinformation. Enfin, si vous avez l'un des prototypes de Linky équipés d'un port USB en façade, attention : le brochage n'est pas standard. Il s'agit en fait du connecteur téléinformation classique modifié et camouflé, comme l'indique la documentation de l'époque (*cpc.cx/hrw*). Le branchement à ce type de compteur demande donc un câble USB spécifique qui permet de récupérer les signaux nécessaires sur deux broches.



La version USB vendue sur Internet intègre un contrôleur série interfacé en USB et tous les composants nécessaires dans un périphérique très compact.

ET SI ÇA NE FONCTIONNE PAS ?

Dans certains cas, la téléinformation ne renvoie pas de données en dehors de l'identifiant du compteur, mais il existe une solution : demander à votre fournisseur d'énergie de l'activer. Le cas reste assez rare étant donné qu'une partie des appareils de chauffage modernes utilise la technologie pour gérer correctement le délestage, mais il existe. Elle nécessite généralement le déplacement d'un technicien, qui sera bien évidemment facturé. Les Linky sont en revanche toujours activés par défaut.

Lire les données recueillies

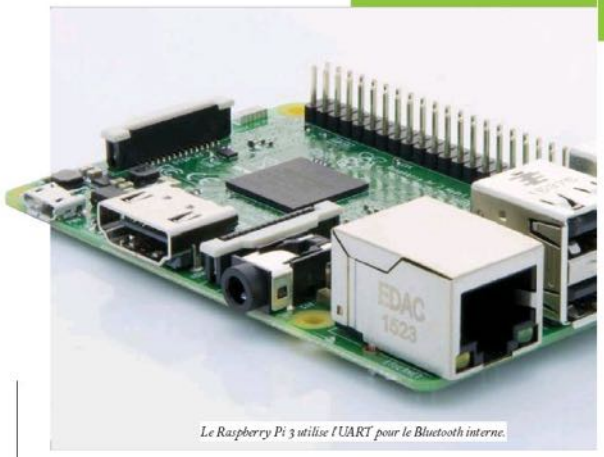
Une fois le Raspberry Pi allumé, la première étape consiste à vérifier que les données arrivent correctement. En supposant que vous utilisiez un Raspberry Pi 3, voici la marche à suivre (elle peut différer avec d'autres modèles). Par défaut, l'UART principal de ce modèle de Raspberry Pi sert en effet pour le contrôleur Bluetooth. Éditez le fichier `/boot/cmdline.txt`, supprimez la ligne `console=serial0,115200` et ajoutez `enable_uart=1` en fin de fichier. Après un redémarrage, entrez la commande qui suit pour configurer le bus série.

```
stty -F /dev/ttyS0 1200 sane evenp parenb cs7 -crtscts
```

Vous pouvez maintenant essayer de lire les données, avec la commande `cat /dev/ttyS0`. Le résultat devrait ressembler à ceci.

```
ADCO 01234567890 @
OPTARIF HC.. <
ISOUSC 45 ?
HCHC 010956910 %
HCHP 016779643 >
PTEC HP..
IINST 021 Z
IMAX 047 J
PAPP 04860 3
HHPHC A ,
MOTDETAT 000000 B
```

Des compteurs Linky, la nouvelle génération.



Le Raspberry Pi 3 utilise l'UART pour le Bluetooth interne.

Si seule la première ligne s'affiche (`ADCO`), la téléinformation n'a pas été activée sur votre ligne (voir l'encadré page précédente). Une trame basique commence par `ADCO` et termine par `MOTDETAT`. Entre les deux, vous trouverez des informations utiles. Chaque ligne contient une étiquette (le type de données), une valeur et enfin un caractère qui sert à vérifier l'intégrité des informations. Dans l'exemple ici, `ADCO` donne le numéro de série du compteur, `OPTARIF` le type d'abonnement, etc. Les premières données intéressantes sont `HCHC` et `HCHP`. Elles indiquent le nombre de Wh consommés en heures creuses et en heures pleines (en cas d'abonnement compatible) au total, depuis l'installation du compteur. En effectuant des comparaisons après avoir stocké les valeurs, il devient possible de mesurer la consommation journalière par exemple. Les deux autres données intéressantes pour nos calculs sont `IINST` et `PAPP`. La première indique l'intensité instantanée en ampères, l'autre la puissance apparente en voltampères (VA). Cette dernière s'obtient justement en multipliant l'intensité instantanée par la tension du secteur (soit ~230 V). Attention : le facteur de puissance n'est pas pris en compte ni indiqué. En conséquence, il vous sera impossible de connaître la consommation instantanée (puissance active) en watts. Au mieux, vous pourrez la déduire en moyenne sur une plage de temps donnée (10 minutes par exemple) à l'aide de l'évolution des index en Wh. Seul Linky permet d'obtenir cette information directement, grâce à la courbe de charge.

Que faire des données ?

Dans la suite, nous expliquerons comment récupérer et analyser les données. La solution la plus simple consiste à afficher quelques informations de façon intelligible, comme la consommation calculée à partir de l'intensité instantanée. Avec les écrans que nous avons présentés dans les pages précédentes, un petit programme en Python qui récupère les données en temps réel et les affiche reste assez simple. Mais, et c'est tout l'intérêt du Raspberry Pi, vous pouvez aussi les stocker dans une base de données et calculer des graphiques qui montrent l'évolution de la consommation.

Récupérer les informations en direct

Lire les données en direct est une chose, les enregistrer et les traiter en est une autre. Avec un Raspberry Pi, vous pouvez parfaitement stocker les informations et les afficher dans un navigateur.

La préparation du système demande quelques commandes. Comme les fichiers PHP à placer sur votre Raspberry Pi sont un peu longs, nous les avons publiés sur le site de *Canard PC Hardware*. Première étape, mettre à jour le Raspberry Pi et installer les logiciels nécessaires.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install apache2 php5 php5-sqlite
libapache2-mod-php5
```

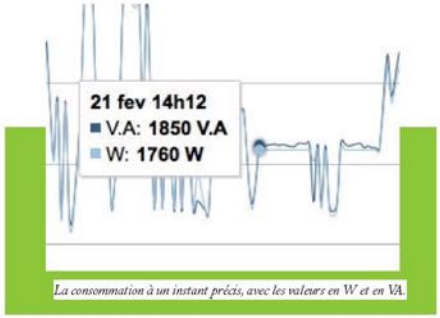
Pensez aussi à configurer l'UART pour qu'il reçoive correctement les données. Vous pouvez ajouter la ligne qui suit dans le fichier `/etc/rc.local`, juste avant le `exit 0` : elle sera exécutée à chaque démarrage. La valeur `/dev/ttyAMA0` dépend de votre adaptateur.

```
stty -F /dev/ttyAMA0 1200 sane evenp parenb cs7 -crtscts
```

Apache a besoin de petits réglages, avec les lignes suivantes.

```
sudo a2enmod rewrite
sudo /etc/init.d/apache2 restart
cd /var/www/html
sudo rm index.html
```

Pensez aussi à mettre le Raspberry Pi à l'heure et dans le bon fuseau horaire avec un `sudo raspi-config`. Vous pouvez éventuellement installer un module RTC (voir page 76). Le code utilisé provient du site MagdiBlog (*cpc.cx/hu6*) avec quelques modifications. Les quatre fichiers nécessaires se trouvent dans un ZIP à cette adresse : *cpc.cx/huO*. Vous devez les télécharger et les placer dans le dossier `/var/www/html`. Attention, ils contiennent des références à une interface série qui porte le nom `/dev/ttyAMA0` : si la vôtre en porte un autre, pensez à modifier toutes les occurrences. La dernière étape consiste à rendre les scripts PHP exécutables et à les lancer à intervalles réguliers, ce qui implique – rappelons-le – un Raspberry Pi à l'heure et dans le bon fuseau horaire.



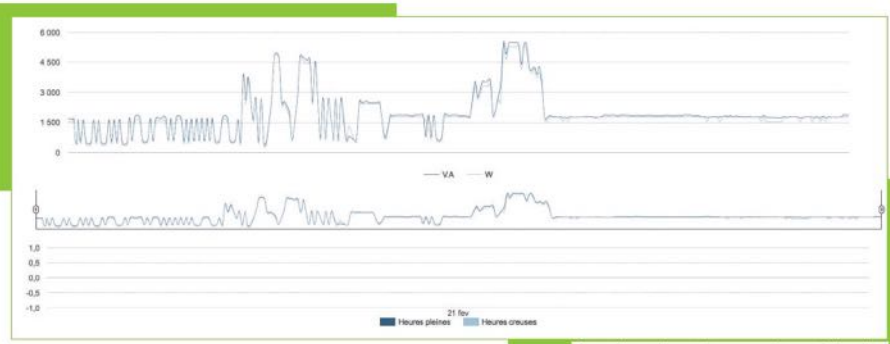
La consommation à un instant précis, avec les valeurs en W et en VA.

```
sudo chmod +x teleinfo_puissance.php
sudo chmod +x teleinfo_conso.php
```

Pour le lancement automatique, tapez la commande `crontab -e` et ajoutez ces deux lignes à la fin du fichier. La première démarre la récupération des données toutes les minutes, la seconde la consommation journalière tous les jours à 23 h 58.

```
* * * * * php /var/www/html/teleinfo_puissance.php
58 23 * * * php /var/www/html/teleinfo_conso.php
```

Après avoir redémarré le Raspberry Pi et l'avoir laissé quelques jours en place pour récupérer les données dans la base appropriée, vous obtiendrez de superbes graphiques qui affichent la consommation pratiquement en temps réel. Pour vous connecter au programme, il suffit d'entrer l'adresse IP de la carte dans un navigateur quelconque.



Un graphique qui montre la consommation sur plusieurs jours.

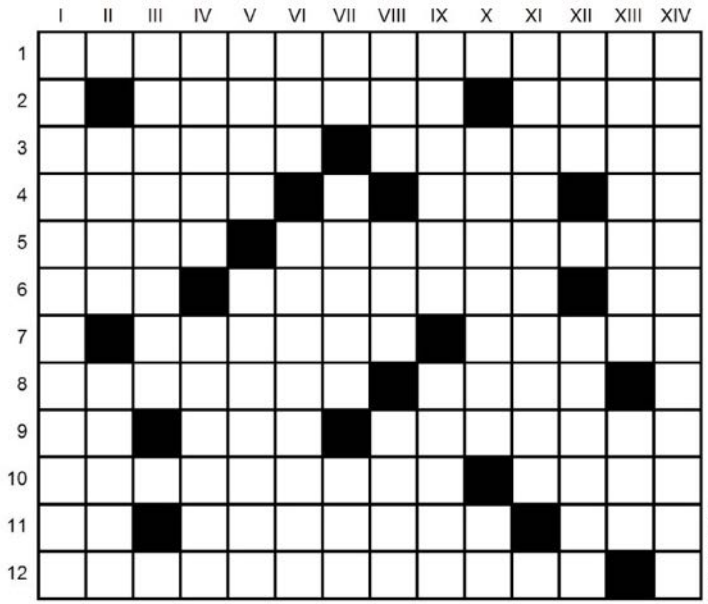
La grille de Kate Mosfet et Lady Diode *(mise au point : Jackie Quartz*)*

Horizontalement

1. Base de montage électronique. 2. Donne son signal au circuit. Resident sur console. 3. Connexions électriques. Remise à jour comme la mémoire. 4. Composant électronique ou LED. Créature japonaise. Pour l'information non disponible. 5. Crochet de boucher. Pièce électronique 6. Réseau des réseaux. Fichier partagé. Fin d'infinitif qui n'est pas if. 7. Chargés du montage et câblage en électronique. Au milieu des canassons. 8. Récepteur. Acteur fétiche de Besson. 9. Possessif. L'Agence spatiale européenne des intimes. Métal qui devrait être absent des circuits électroniques. 10. Odes grecques. Jardin merveilleux. 11. Fait remonter le post du forum. Complètes. Comme un PC d'Asus. 12. Récupère l'énergie thermique.

Verticalement

I. Composant électronique. II. Département des Hauts-de-France. Connecteur flexible de circuit. III. Résistance électrique. IV. À angle droit. Démesurée. V. Cochon basque. Champ électrique. VI. Pronom pluriel. Raccordé. VII. Unité informatique. Logique sur les circuits. Finit le caviar. VIII. Inspection générale des finances. Messieurs anglais. Mélangea. IX. Prononce comme une mélopée. Remise à zéro. X. Énoncer bêtement. Pronom réfléchi. XI. Composant électronique. XII. Forme de sida. Raccorder à l'étain. XIII. Possessives. Crié comme un cerf. XIV. Produisent de l'électricité.



* désolé, c'était trop tentant

CANARD PC HARDWARE

Canard PC Hardware Hors-série
Est édité par Presse Non-Stop SAS au capital de 86 400 euros.

Immatriculée au RCS de Paris sous le n° 450 482 872.

Président : Jérôme Darnaudet

Associés : Jérôme Darnaudet, Domisys, Gandi, Ivan Gaudé, Pascal Hendrickx, Olivier Peron et Michael Sarfati

Siège social : Presse Non-Stop - 14 rue Soleillet - BAL 62 - 75020 Paris

Administration

Tél : 01 43 49 42 27

Secrétariat : Pauline Carmet pauline@pressenonstop.fr

Abonnements : abonnements@canardpc.com

Rédaction

Directeur de la publication : Jérôme Darnaudet

Rédaction en chef : Samuel Demeulemeester

Rédacteur en chef online : Ivan Gaudé

Ont participé à ce numéro : Pierre Dandumont et Samuel Demeulemeester

Premier rédacteur graphique : Jean-Ludovic Vignon

Rédacteurs graphiques : Katell Chabin, Marie Lemaire et Thomas Rainfroy

Secrétaire de rédaction : Sonia Jensen

Dessinateur : Didier Couly

Publicité

Denis
denis@canardpc.com
Tél : 09 66 88 42 27

Impression

Imprimé en France par : CPI Aubin Imprimeur

Diffusion : PRESSTALIS

Commission paritaire : 0620 T 90441

ISSN : N° 2264-4202

Hors-série numéro 7
prix unitaire : 6,90 €

Date de parution : 10 novembre 2016

Dépôt légal à parution

SOLUTIONS : Horizontalement : 1. Circuit imprimé. 2. Horloge. Evil. 3. Nœuds. Flashée. 4. Diode. Oni. NC. 5. Esse. Composant. 6. Net. Torrent. Er. 7. Agents. Nasso. 8. Antenne. Reno. 9. Ta. Ase. Mercure. 10. Épinicies. Eden. 11. Up. Totales. EEE. 12. Régénérateur.
Verticalement : I. Condensateur. II. Oise. Nappe. III. Rhéostat. IV. Coudé. Géante. V. Urde. Tension. VI. Ils. Connecté. VII. To. Porte. Iar. VIII. IGF. Mrs. Mêla. IX. Mélope. Reset. X. Anonner. Se. XI. Résistance. XII. IVH. Souder. XIII. Miennes. Réé. XIV. Électrogènes.

ASUS®
IN SEARCH OF INCREDIBLE

CARTES MÈRES ASUS X99

FAITES PARLER VOS CŒURS. CHOISISSEZ LE MEILLEUR

N°1 DES VENTES	Plus de 500 millions de cartes mères vendues depuis 1989.
FACILES D'UTILISATION	La simplicité est l'élément clé des cartes mères ASUS.
FIABLE	ASUS vise à fournir une stabilité et une fiabilité sans pareille.
RÉCOMPENSÉES	Partagez la passion des experts.

ASUS - No.1 des ventes de cartes mères depuis plus de 10 ans