Introduction aux microcontrolleurs

1 Définition et exemples

Un microcontrolleur est, au fond, un microprocesseur, mais ce qui le distingue du cœur de votre téléphone portable, tablette ou ordinateur, ce sont deux propriétés assez spécifiques :

- la présence de (plus ou moins) nombreuses entrées-sorties dédiées à recevoir des informations (capteurs, boutons) et à agir en conséquence ou à commander des actions externes
- une puissance très en retrait par rapport à un processeur récent (mais pas si ridicule) et adaptée aux problèmes posés, permettant également une consommation électrique faible, l'absence de refroidissement (actif comme, le plus souvent, passif)

Au fond, les microcontrolleurs sont partout, ou presque. Une utilisation fréquente est d'assister un microprocesseur dans certaines tâches, comme on va le voir, mais on en trouve aussi au cœur de nombreux dispositifs simples comme l'internet des objets (internet of things) ou, sans doute désormais, de nombreuses calculatrices graphiques et programmables.

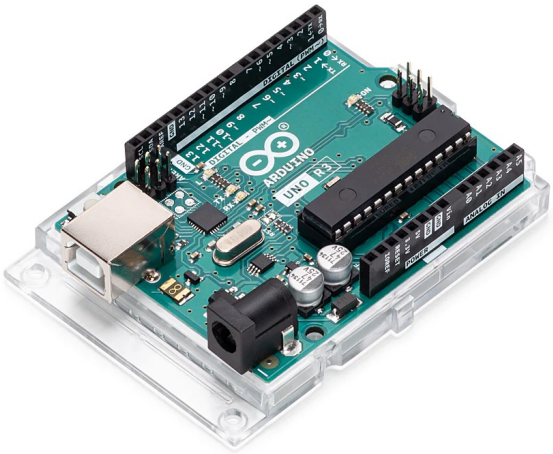
Si vous démontez (je ne vous le recommande pas pour autant, à moins que vous ne soyez très soigneux) le clavier de votre ordinateur (fixe, pour simplifier, et avec une connexion filaire) vous y trouverez sans le moindre doute un microcontrolleur, tout comme à l'intérieur d'une souris. On se l'explique aisément : dans un cable usb qui relie le clavier à l'ordinateur, il y a essentiellement quatre fils, deux pour l'alimentation électrique puisque, vous l'avez compris, un clavier contient en son sein un tout petit ordinateur, et deux autres pour l'échange de données. Or, un clavier d'ordinateur contient un nombre très important de touches (plus de 100 pour un clavier complet avec touches de fonction et pavé numérique). Bref, c'est au microcontrolleur présent dans le clavier que revient la tâche de scanner le clavier pour toute touche enfoncée ou relachée, et de transmettre l'information en respectant un certain protocole à l'ordinateur.

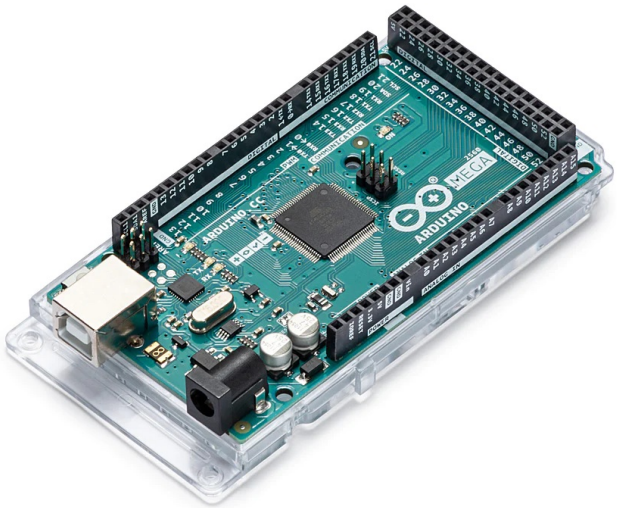
Nous verrons en activité le problème de scanner un clavier, problème qui n'est pas si évident à vrai dire, car il est illusoire sur un clavier de plus de 100 touches qu'un fil séparé soit dédié pour chacune des touches qui le composent. A supposer qu'un microcontrolleur dispose d'autant d'entrées, le cablage en serait un cauchemard et le coût prohibitif… On verra que la solution passe par un matriçage des touches du clavier, ce qui réduit très sensiblement la complexité du circuit électrique, mais pose alors des problèmes intéressant de programmation !

2 Le précurseur : Arduino

Vous le devinez, avant qu'un microcontrolleur ne prenne sa place dans un clavier, quelque part dans une voiture, un avion, dans une imprimante 3d, un compteur linky ou un four à micro-ondes, de nombreux prototypes doivent être réalisés, ce qui est à la fois coûteux en temps et en argent, nécessitant, a priori, de concevoir nombre de circuits imprimés, de réaliser la soudure, puis la programmation, les tests, puis de remettre son travail sur l'ouvrage. Une équipe d'enseignants en Italie (à Ivrée) a au début des années 2000 conçu à la fois un langage simplifié pour développer des applications sur des microcontrolleurs (langage qu'on appelle encore arduino), lequel est une surcouche sur le langage C/C++, un environnement intégré de programmation, et des cartes électroniques comportant un microcontrolleur, un port usb et les ports d'entrée-sortie du microcontrolleur très accessibles, rendant l'apprentissage et le prototypage beaucoup plus faciles.

Les cartes électroniques développées par la fondation incluent la carte Arduino Uno et l'Arduino Mega :





La seconde comporte, on le voit, bien davantage de ports que la première, et le microcontrolleur dispose également de bien davantage de mémoire pour les scripts et les variables (même si cela reste assez dérisoire vis-à-vis des capacités d'un smartphone ou d'un ordinateur moderne). Cela étant dit, le microcontrolleur de la mega est de la même famille que la uno, cadencé à la même vitesse (peut-être devinerez-vous un oscillateur à quartz sur les cartes ci-dessus, à 16Mhz pour l'une comme pour l'autre).

3 Des cartes plus modernes

Si les cartes montrées ci-dessus, à base de microcontrolleurs 8 bits Atmel, existent toujours et peuvent être commandées à la fondation Arduino, comme sous forme de copies chinoises bon marché (rien d'illégal à celles-ci, sauf peut-être l'utilisation du nom Arduino sur ces cartes), la fondation Arduino elle-même développe désormais des cartes à base de microcontrolleurs 32 bits sous licence ARM (SAMD21, STM32, RP2040), et le choix de microcontrolleurs comme de cartes électroniques est désormais très large.

Les cartes arduino présentées ci-dessus restent intéressantes pour faire ses premiers pas avec les microcontrolleurs, à ceci près que les contraintes en sont assez restrictives : pour Arduino uno, 32ko de mémoire flash, c'est-à-dire de mémoire dédiée pour les programmes et les données statiques, et 2ko (2048 octets !) pour les variables, la pile d'exécution etc…

Les microcontrolleurs plus récents sont à la fois plus rapides (cœurs 32bits, fréquence de fonctionnement dépassant souvent les 100Mhz) et ont beaucoup plus de mémoire, autant pour la mémoire flash que pour la mémoire dite vive. Par exemple, le microcontrolleur RP2040 (pi pico) développé par la fondation Raspberry (connue pour ses ordinateurs Raspberry Pi) utilise le jeu d'instructions ARM (M0+), est cadencé à 133Mhz, a 2Mo de mémoire flash et 256Ko de mémoire vive.

Les nouvelles capacités de ces microcontrolleurs aidant, Damien George a cherché à démocratiser encore un peu la programmation de microcontrolleurs en permettant qu'ils puissent être programmés en python. Ce n'est en fait pas l'intégralité du langage python qui fut implémenté, mais une version bien spécifique aux microcontrolleurs, plus légère et adaptée donc aux ressources limitées de ceux-ci (mais qui demande tout de même beaucoup plus que ce dont dispose Arduino uno !), et Damien George a donné le nom de micropython à ce langage. (En fait, parallèlement à ce langage, il avait développé une carte électronique à base de STM32, portant le nom de pyboard, et vendue préprogrammée avec micropython installé. Désormais, micropython a été porté à bon nombre de cartes électroniques et n'est plus réservé à cette carte pyboard qu'on peut néanmoins toujours commander.)

On retrouve ici les avantages du langage python : la possibilité d'exécuter des commandes en direct dans une console et d'en voir l'action immédiatement, sans passer par la case compilation. Bien entendu, si on veut profiter de toute la puissance du microcontrolleur, on pourra préférer un langage compilé, comme C/C++ avec arduino ou d'autres langages (pourquoi pas l'assembleur, pour les plus courageux !).

A noter que sur vos calculatrices, de plus en plus nombreuses, qui permettent de programmer en python, c'est en fait de micropython qu'il s'agit.

Un fork de micropython (nommé circuitpython) a également été développé par Adafruit. Les différences en sont assez tenues et on se contentera de l'original dans nos expérimentations.

4 La simulation

On peut se contenter dans un premier temps d'expérimenter avec la simulation. Deux sites très intéressants le permettent en effet, sans qu'il soit d'ailleurs nécessaire d'installer quoi que ce soit sur son ordinateur, puisque tout s'exécute dans le navigateur. L'un s'appelle Tinkercad circuits (https://tinkercad.com) et est développé par Autodesk (qui n'est pas du tout une association à but non lucratif, mais qui propose il faut le reconnaître nombre d'outils à but pédagogique et qui sont proposés gratuitement).

Tinkercad permet la conception visuelle de circuits imprimés et de tester leur fonctionnement. On peut pour ce faire insérer ici ou là un multimètre, un oscilloscope, et bien sûr des microcontrolleurs, même si on est limité à des cartes Arduino Uno et à des cartes micro:bit. La programmation est le point faible de Tinkercad, car est limité avec Arduino à un langage par blocs (genre scratch) ou Arduino, et avec micro:bit à une programmation également par blocs ou bien par une sorte de micropython. En fait, les différences sont suffisamment importantes avec micropython pour que, en termes de programmation, je lui préfère Wokwi.

Il n'empêche, si on veut réfléchir un peu à la conception de circuits électroniques, à l'utilisation de composants passifs (résistances, inducteurs et condensateurs, diodes, transistors) et actifs (amplificateurs opérationnels) sans risquer de détruire des composants (alimenter une led sans rajouter de résistance par exemple, ou tout court-circuit), voire de se mettre en danger (ne jamais brancher à l'envers un condensateur polarisé par exemple !), Tinkercad garde tout son intérêt, et je vous invite à vous créer un compte.

Wokwi (https://wokwi.com)quant à lui est développé par un passionné, dispose de l'émulation de bien davantage de microcontrolleurs que Tinkercad (Arduino, micro:bit, raspberry pi pico mais aussi ESP32, un des plus puissants micontrolleurs tout public) et permet la programmation en Arduino ou micropython, mais en revanche ne dispose pas des fonctionnalités analogique des circuits imprimés : vous ne pourrez ici mesurer un courant traversant un point du circuit, ou une tension entre deux points, ni mettre en évidence un oubli de résistance ici ou là qui conduirait à la destruction de composants…

Bref, on a là deux outils en ligne très complémentaires. Comme nous nous concentrerons sur la programmation, nous utiliserons plutôt le second, mais je ne peux que vous conseiller d'aller essayer également Tinkercad.

5 Acheter un microcontrolleur ?

Si vous en avez les moyens, ou que votre anniversaire arrive bientôt, disposer d'un microcontrolleur, de quelques composants, jumpers et autres peut être un outil motivant et ludique pour l'apprentissage d'un petit peu d'électronique et, à la fois, de programmation. Il existe des boitiers complets comprenant composants et microcontrolleurs, mais on peut aussi se constituer, petit à petit, un panel de composants utiles pour expérimenter.

Certains cartes électroniques sont munies de nombre de capteurs et, au moins dans un premier temps, peuvent donc se suffire à elles-mêmes.

Je ne recommande donc l'achat d'une carte arduino (plus probablement chinoise d'ailleurs, vu les prix) qu'accompagnée de composants, donc sous la forme d'un kit. A choisir, un kit autour d'un raspberry pi pico (RP2040) ou d'un microcontrolleur ESP32 peut d'ailleurs être préférable (et pas forcément plus cher, car une carte microcontrolleur basée sur ESP32 ou RP2040 a un prix de vente de quelques euros)

En revanche, peuvent se suffire à eux-même :

- la micro:bit, commandée par la BBC, et distribuée gratuitement à toute une tranche d'âge de collégiens anglais. La version 2 de cette carte se programme par blocs dans un navigateur, en javascript, à l'aide d'Arduino mais aussi, et c'est ce qui fait je trouve son intérêt, en micropython. On peut l'acheter autour de 23 euros (sur kubii.fr, ou mouser.fr). Présents sur la carte sont : un accéléromètre, un gyroscope, un capteur de température, un microphone, un buzzer, une matrice de 5x5 leds qui sert d'écran, le bluetooth… Cette carte est très commode à utiliser en micropython, mais a un défaut pénible selon moi, je n'ai su trouver comment accéder au microphone pour obtenir en micropython un enregistrement, ou la mesure en temps direct de la pression acoustique. L'accès au microphone est limité à la mesure d'un niveau sonore pour réagir à un bruit. (Ce n'est bien sûr pas tant un défaut de la carte elle-même que des bibliothèques micropython qui l'accompagnent, et peut-être qu'un jour celles-ci seront un peu plus développées.)

- Certaines cartes développées par M5Stack à base de ESP32 : M5Stick-C-Plus qui comportent elles aussi tout ce que la bbc:micro a mais aussi le wifi, un écran lcd. On peut la programmer, à tout le moins, à l'aide du langage Arduino (complété de fonctions spécifiques à ESP32), ainsi qu'avec UIFlow, qui est une version modifiée de micropython (et qui permet aussi la programmation par blocs). Petit souci, la programmation sous micropython est assez pénible et s'interface difficilement avec les éditeurs appropriés tels que Thonny, mu-editor. (Les choses peuvent ici évoluer)

  Dans la même veine, les boitiers M5Stack Core et Core2 proposent à peu près la même chose que le M5Stick-C, avec un écran plus grand, plus de mémoire flash, et un emplacement pour micro-sd. (Compter entre 25 et 30 euros pour le M5Stick-C plus, et autour de 50 euros pour un Core2)

  Je recommande également mouser.fr pour ce genre de microcontrolleur. Attention : les prix y sont indiqués hors taxes et hors frais de livraison.

6 Quelques conseils

Si vous vous lancez dans l'aventure et vous équipez d'une carte microcontrolleur, voici quelques points dont il vaut mieux être conscient :

- Un court-circuit ou une surcharge peut avoir des conséquences fâcheuses (destruction de tout ou partie de la carte microcontrolleur, de composants externes voire, ce n'est pas complètement exclu, destruction du port usb de l'ordinateur s'il ne dispose pas d'une protection suffisante : sur un ordinateur récent, je pense que c'est à peu près impossible, mais on n'est jamais trop prudent !)

- Un micro-controlleur n'est pas fait pour alimenter des dispositifs électriques nécessitant un tant soit peu de puissance : alimenter une led ou deux, quelques néopixels, tel ou tel capteur oui, mais un moteur, une rangée de plus que quelques leds non : et dans ce genre de circuit, une alimentation séparée pour les leds ou le ou les moteurs sera nécessaire.

- Les broches (pin en anglais) d'un microcontrolleur peuvent avoir plusieurs fonctions : entrée digitale, sortie digitale, entrée analogique, sortie analogique, masse (GND), Vcc. Bien lire et relire, configurer si nécessaire les broches.

  - Une sortie numérique a, a priori, deux état : bas (même niveau que la masse) et haut : selon le microcontrolleur, il peut s'agir de +5V, +3.3V. (+5V pour arduino, +3.3V pour ESP32, RP2040 et beaucoup d'autres microcontrolleurs. On notera par la suite Vcc cette tension nomiale). Lorsqu'une broche est configurée comme une sortie numérique, le courant pouvant être délivré ne doit pas dépasser 20mA pour Arduino uno, et cet ordre de grandeur doit être à peu près le même pour nombre d'autres microcontrolleurs.

- Une led branchée sur une sortie numérique doit toujours être accompagnée d'une résistance en série. En effet, la tension aux bornes d'une led en fonctionnement est, pour une led rouge, verte ou jaune de l'ordre de 1.6V de sorte que si on impose une tension de 3.3 ou de 5V, alors il est certain que la led, et sans doute le microcontrolleur vont en souffrir puisque le courant qui sera délivré dépassera les limites tant de la led que de la broche du microcontrolleur. Par exemple pour une tension délivrée de 3.3V, si l'on souhaite un courant d'environ 10mA, on mettra en série une résistance de $\frac{1.7}{10e^{-3}} = 170\Omega$ au moins. Pour une tension de 5V à la broche de sortie et pour le même courant, on mettra en série une résistance de $\frac{3.4}{10e^{-3}} = 340\Omega$ au moins. (Je vous laisse méditer sur ce point !)

- Une entrée numérique distingue elle aussi deux états. Il faut quelquefois adapter des capteurs dont la logique utilise une tension nominale de 5v ou de 3.3v (il existe des dispositifs pour cela, mais nous n'aborderons pas ce point).

- Une sortie dite analogique peut être de deux natures :

  - L'une n'est pas tant une sortie analogique qu'une modulation de sortie digitale : certaines broches peuvent être capables d'émettre un signal rectangulaire, alternant entre le signal haut et le signal bas, à une fréquence relativement importante (500Hz environ pour Arduino). L'intensité qui en découle est fonction de la proportion de temps passé sur un signal haut par rapport au tout (de 0% à 100%). Le vocabulaire anglais est PWM (pulse width modulation) et la proportion de temps passé sur un signal haut est le « duty cycle ». Voir cet extrait de la documentation arduino : https://docs.arduino.cc/tutorials/generic/secrets-of-arduino-pwm pour plus de détails. Sur un arduino uno, les broches capables d'une sortie PWM sont marqués d'un tilde ~ (voir photo de la première page). Même si la sortie n'est donc pas à proprement parler une sortie continue variable entre 0 et Vcc, l'effet peut en être le même, comme on peut en faire l'expérience avec des led que l'on allume et éteint très rapidement (ce qu'il est impossible de voir à l'oeil nu) et qui donnent l'impression de briller plus ou moins fort, selon la proportion du temps où elles sont allumées.

  - certains microcontrolleurs (rares, on trouve cela sur certains STM32, comme certaines cartes nucléo) disposent d'un véritable convertisseur digital-analogique et peuvent sur certaines broches délivrer une tension variable entre 0 et Vcc (entre 0V et 3.3V donc la plupart du temps).

- Des entrées analogiques (ADC : analog to digital converter) sont en revanche présentes sur à peu près tous les microcontrolleurs. Elles peuvent distinguer le plus souvent 1024 (sur 10 bits) états, voire dans certains cas 4096 états (12 bits), depuis la plus faible tension 0V (=GND) jusqu'à la tension nominale du microcontrolleur (5v ou 3.3V). A noter que la méthode utilisée pour mesurer un signal analogique conduit à une mesure plus lente que la mesure d'un signal digital : plusieurs comparateurs en cascade sont utilisés pour déterminer la tension imposée à la broche mesurée.

- Lorsqu'une broche est configurée en entrée, on peut considérer qu'aucun courant (ou presque) ne traversera ladite broche, pourvu toutefois que celle-ci ne soit pas soumise à des tensions inférieures à 0v ou supérieures à Vcc. Il n'y a donc a priori pas de protection à considérer, mais il n'empêche qu'il est courant de rajouter une résistance de rappel (pull-down resistor) ou de tirage (pull-up resistor) à une broche configurée en entrée. La raison en est que si une broche est laissée « flottante » la valeur de lecture de celle-ci est indéterminée, et il suffit d'un rien pour qu'on passe d'un état à un autre (de manière à peu près aléatoire). Voici une vidéo (en anglais) sur youtube qui montre bien cet effet : https://www.youtube.com/watch?v=wxjerCHCEMg

  Bref, lorsqu'une broche est utilisée pour lire l'état d'un bouton, une résistance de rappel ou de tirage est indispensable.