Système d'arrosage automatique
13 – Les processeurs dans la chaîne de traitement de l'information

Travail sur la partie "processeur" du système d'arrosage automatique.
Durée : entre 1h30 et 3 heures.

Le cours précédant à détaillé un des capteurs de notre système d'arrosage automatique, puis entamé la présentation de l'étape suivante dans la chaîne de traitement de l'information : la partie "processeur".

Nous allons aujourd'hui nous concentrer sur cette partie, et laisser le reste de côté, nous reviendrons sur l'énergie quand nous parlerons de la partie actionneur (la pompe) et de la partie panneau solaire et stockage sur batterie.

1 – Un peu d'histoire.

Les premier processeurs, et l'informatique "moderne", remontent au début des années 1970, même si les premier ordinateurs sont un peu plus vieux (quelques dizaines d'années, à partir de 1936) et les première machines à calculer mécaniques encore plus (quelques siècles, 1642 pour la Pascaline de Blaise Pascal).
Si cette partie de l'histoire des ordinateurs vous intéresse, vous pouvez lire un article assez complet sur ce sujet sur Wikipedia : https://fr.wikipedia.org/wiki/Histoire_des_ordinateurs

Pour notre part, nous commencerons donc autour des années 1970, parce que les techniques développées à cette époque sont toujours utilisées dans nos ordinateurs, téléphones, et autres appareils.
Parmi ces techniques, les plus importantes sont :

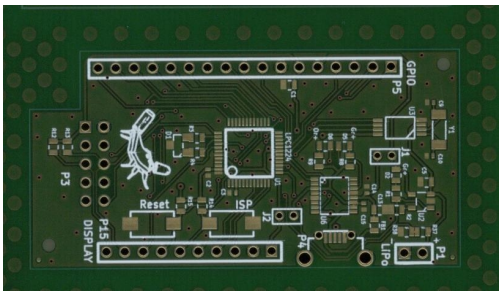
- l'utilisation des circuits imprimés en époxy
- l'utilisation de circuits intégrés (composants électroniques)
- l'utilisation de composants très similaires aux circuits intégrés pour la mémoire "vive"
- l'utilisation du système binaire
- l'utilisation de systèmes d'exploitation
- le développement des systèmes de stockage d'informations.

Je ne reviendrais pas sur le système binaire (qui a pris le dessus en informatique car il est plus simple à mettre en œuvre que le système décimal) puisque je vous en ai déjà parlé dans le premier cours sur la chaîne de traitement de l'information (document 03 – Traitement de l'information et énergie).

Je ne reviendrais pas non plus sur le développement des systèmes de stockage d'information, qui évoluent constamment (disquettes, disques durs, CD, clés USB, ...), et que vous connaissez relativement bien pour la plupart.
Si vous voulez trouver un peu plus d'informations sur le sujet par curiosité, vous pouvez commencer par l'article de Wikipédia sur ce sujet : https://fr.wikipedia.org/wiki/Stockage_d'information

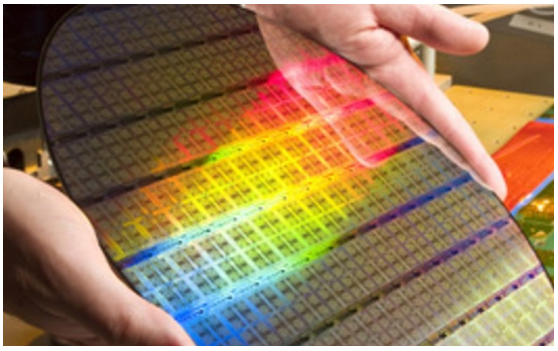
Nous allons donc évoquer les autres points : les circuits imprimés, les circuits intégrés et les systèmes d'exploitation.

2 – Les circuits imprimés



Attention, il ne faut pas confondre circuit imprimé (en photo ici) et circuit intégrés. Le **circuit imprimé** est une plaque de résine sur laquelle se trouvent des pistes de cuivre, le plus souvent protégées par un vernis, sauf aux endroits ou seront soudés les composants.
L'utilisation de l'époxy pour faire ces circuits imprimés a permis de créer des circuits de plus en plus complexes, plus fins, avec plus de couches, et donc d'obtenir des systèmes de plus en plus petit, mais le principe général de fabrication reste tout le temps le même.

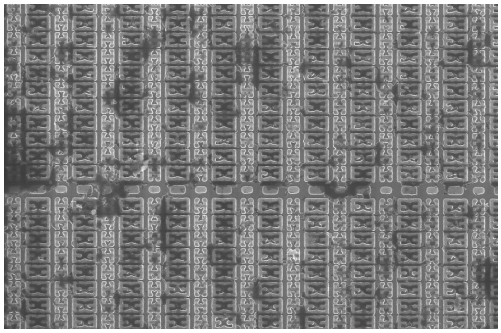
3 – Les circuits intégrés



Les **circuits intégrés** (et les mémoires) sont des composants électroniques créés à partir de silicium (le composant des roches et du sable)
Les fabricants de processeurs utilisent un procédé de gravure sur ce que l'on appelle un "Waffer" (le grand disque sur la photo) pour fabriquer les processeurs et autres composants "actifs".

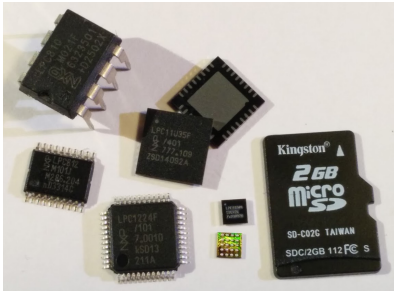
Un Waffer comme celui-ci comporte plusieurs dizaines de processeurs qui seront découpés avant d'être placés dans un bloc de résine protecteur, et disposant de pattes (ou de petites billes sur le dessous) qui permettront de le souder. Certains composants sont si petits (moins de 1mm de côté) qu'il y en a des milliers sur un Waffer, mais ce n'est rien devant la taille des transistors qui les composent : quelques nanomètres.

Sur chaque Waffer sont gravés des milliers, puis des millions et enfin des milliards de transistors avec la progression des techniques de gravure.



Rappelez vous la vidéo de la deuxième séance qui parlait de gravure de transistors.
Je vous remet l'image prise par un microscope électronique à balayage que je vous avais montré : cette image montre une toute petite partie d'un Waffer, avec quelques colonnes de transistors.

La façon dont sont organisés ces transistors permet de créer différentes fonctions, comme la mémoire, ou des opérations simples comme l'addition et la soustraction.
Cela forme des blocs, qui sont ensuite connectés entre eux, pour former des blocs plus complexes, jusqu'à obtenir un composant capable d'exécuter le code des programmes de l'utilisateur.



Exemples de petits micro-contrôleurs récents (Avec une carte micro-SD pour l'échelle)

4 – Les systèmes d'exploitation

La dernière grande avancée des années 1970 est la création des systèmes d'exploitation, dont certains sont toujours utilisés, heureusement avec quelques améliorations pour s'adapter aux nouveaux usages et aux nouveaux composants.

Un système d'exploitation est un ensemble de logiciels qui permet de faire fonctionner un ordinateur, un téléphone, ou tout autre matériel évolué. Vous en connaissez au moins trois normalement : Windows, IOS et Android, mais il y en a beaucoup d'autres !

Android (le système développé principalement par Google) est un dérivé des systèmes "GNU/Linux", qui sont eux même dérivés des premiers systèmes UNIX, créés au début des années 1970.

IOS (le système d'Apple) est un dérivé des systèmes BSD, qui sont encore une fois dérivés des premiers systèmes UNIX.

Les systèmes **GNU/Linux** sont très très largement utilisés, mais invisibles pour l'utilisateur, par exemple sur les serveurs, ou les robots et autres systèmes embarqués, comme les télévisions, ou les box d'accès à Internet !

Les systèmes **UNIX** et leurs dérivés sont toujours utilisés dans certains domaines spécifiques.

Et enfin, **Windows**, développé par Microsoft, est un peu à part.
Il est très connu et utilisé car souvent vendu automatiquement avec un nouvel ordinateur, ce qui a aidé à sa très large diffusion.

L'avantage des systèmes d'exploitation est d'être communs à différentes machines, ce qui facilite l'écriture des programmes pour l'utilisateur, qui n'a plus à se soucier de la marque du matériel utilisé sur chaque machine, ni de comment fonctionne le matériel.
Les premier systèmes d'exploitation ont été codés en langage C, plus pratique et surtout plus universel que l'assembleur (ce sont des langages de programmation, je reviendrais dessus plus tard).

5 – L'exécution des "programmes utilisateurs"

Un système d'exploitation seul n'a pas beaucoup d'intérêt. Il est là uniquement pour permettre à l'utilisateur d'exécuter ses propres **programmes**, ou **logiciels**, comme par exemple un navigateur Web, un logiciel de dessin, un lecteur de vidéos, un logiciel de traitement de texte, un jeu ... ou le programme de notre système d'arrosage automatique !

Un programme est un ensemble composé de **données** d'un côté, et de **code** de l'autre.

Les **données** sont les **valeurs** que l'on utilise dans le programme. Par exemple si on veut que notre programme attende 10 minutes, alors "10" sera une donnée dans notre programme. Ou encore, si on veut qu'il affiche "Coucou", alors "Coucou" sera aussi une donnée dans notre programme.

Ces données peuvent aussi provenir d'éléments extérieurs, comme un capteur d'humidité dans notre cas. Il faudra alors utiliser une "**variable**" pour enregistrer cette donnée d'entrée temporairement dans notre programme.
Cet enregistrement n'est que temporaire, c'est à dire que si on éteins la machine, ou qu'on quitte le programme, alors la donnée est perdue, et le programme recommencera avec une variable vide la prochaine fois. Il faudra à nouveau lire la donnée en provenance du capteur et la stocker dans la variable.

Une donnée peut bien entendue être enregistrée, par exemple dans un fichier sur une clé USB, ce qui permettra de la retrouver, soit depuis notre programme, soit en utilisant d'autres programmes.
Ces données de sortie peuvent aussi être envoyées à d'autre programmes, ou servir à piloter des éléments externes, comme une pompe dans le cas de notre système d'arrosage.

Il existe un autre type de variables, utilisées par exemple pour compter de 1 à 5 dans un programme. Une telle variable est "**initialisée**" à "1", puis **incrémentée** jusqu'à "5", c'est à dire que l'on ajoute "1", 5 fois de suite.
Il est ainsi possible de compter de 2 en deux, en partant de 5, jusqu'à 25. La variable sera initialisée à 5, puis on ajoutera 2 à chaque fois, jusqu'à 25, donc 10 fois.

Le **code** est une **succession d'opérations élémentaires**, qui utilisent les données pour exécuter une tâche.

Le code est la représentation dans un **langage** donné de l'**algorithme** d'un programme. Je vous ai déjà parlé d'algorithmes dans le cours précédant.
Le mot **algorithme** vient du nom d'un mathématicien perse du IX^{e} siècle, Al-Khwârizmî.
Un algorithme est une suite d'opérations à exécuter **dans un ordre déterminé** permettant de résoudre un problème.

Il existe énormément de langages de programmation permettant d'écrire du code. Le langage C, le langage python et le langage assembleur en sont des exemples, tout comme un que vous avez déjà utilisé : **Scratch** !
C'est ce dernier que nous utiliserons cette année !

Les langages de programmations comme Scratch, python, ou le langage C sont des langage que le processeur d'un ordinateur ne sait pas exécuter.
Il est alors nécessaire de passer soit par un **interpréteur** (on parle alors de **script** plutôt que de programme), soit par un **compilateur**.
L'interpréteur est un programme qui lit le script à chaque exécution, pour exécuter les opérations demandées les unes à la suite des autres.
A l'inverse, **le compilateur** est un programme qui n'est utilisé qu'une fois, et dont le travail est de traduire notre code en une suite d'instructions que le processeur sait exécuter. Il devient alors possible d'exécuter le programme sans avoir le "**code source**", c'est à dire le code écrit dans un langage que les humains peuvent facilement comprendre.
Les langages de programmation permettent d'écrire les algorithmes sous une forme qui pourra être soit lue par un "interpréteur", soit traduite en "langage machine" par un compilateur.

Chaque langage définit au moins un certain nombre de types de **variables** (texte, nombres, ...), d'opérateurs (addition, multiplication, ...), et des **structures de contrôle** permettant de réaliser des **tests** et des **boucles**.

Travail personnel :
Vérifiez que vous avez la possibilité d'utiliser Scratch (https://scratch.mit.edu/) et créez un programme simple qui compte de deux en deux, en partant de 7, jusqu'à 31, et en attendant 10 secondes entre chaque incrémentation.
Commencez par écrire sur papier l'organigramme du programme correspondant avant de passer sur l'ordinateur.

Prévenez moi si vous avez des problèmes pour l'utilisation de Scratch.
Si besoin, vous pouvez me faire valider votre organigramme.