CH-1092 Belmont, Switzerland
Tel +41 21 728-6156, Fax 728-6157
Email info@didel.com, www.didel.com

C/Arduino pour comprendre les bases de l'informatique

1 Introduction

Le fonctionnement électrique d'un ordinateur se base sur les lois de l'électricité, enseignées au collège. Le transistor commutateur et les circuits logiques sont les éléments de base des ordinateurs, intéressants mais pas essentiels comme base de la culture informatique.

Trouver une architecture efficace à partir de ces éléments a pris du temps. Il a fallu se convaincre que notre système décimal était inapproprié pour les calculs, définir le codage des informations, développer les notions de macros, de fonctions, de pile (stack), de FIFO, maîtriser la récursivité, les interruptions, et enfin le multitâche. Comprendre ces notions aide à naviguer dans la complexité des systèmes informatiques actuels.

Ces concepts font-ils partie de la culture informatique? Si la réponse est oui, ce papier présente un scénario compatible avec les contraintes d'une formation informatique obligatoire. Ce qui est proposé ici n'est qu'une partie de la base culturelle à mettre en place, qui doit tenir compte des mêmes contraintes:

1. Il faut être efficace, il y a aussi d'autres concepts à expliquer
2. Il faut que cela soit concret et motivant
3. Il faut que le coût soit faible
4. Il faut que les enseignants soient faciles à former

2 Quel support utiliser?

Faut-il du matériel spécialisé pour enseigner plus efficacement les concepts de base de l'informatique? La physique et la chimie peut-elle ne s'apprendre que dans des livres/tablettes? Actuellement, si on réduit le temps passé dans les labos, est-ce à cause de leur inutilité pédagogique? Les contraintes doivent pousser à optimiser, et non pas sacrifier,

En informatique, s'il faut du matériel, quel matériel choisir? Pour expliquer quels concepts, pour développer quelles structures mentales? Un seul matériel ou plusieurs comme en physique?

Le domaine de l'informatique continue à s'élargir et les outils se multiplient et évoluent pour être utilisés sans connaissances préalables. Si on se préoccupe de culture informatique, il faut, comme avec l'histoire, dégager les phases d'évolution, répertorier les concepts, créer la documentation, ouvrir des musées et des sites internet. Comme en physique, musique, gymnastique, il faut des équipements spécialisés. Le PC, la tablette n'est que le sommet de l'iceberg informatique.

2.1 Le microcontrôleur comme système didactique

Le cœur de tout système informatique est un processeur avec sa mémoire et des entrées-sorties pour communiquer. Les systèmes ont évolué pour interagir d'abord avec des automates, puis avec l'homme, avec ses gadgets, et avec la techno-sphère qui se développe actuellement. On ne peut comprendre et expliquer au niveau du collège et gymnase que quelques principes à la base de cette récente espèce technologique — en s'intéressant comme en biologie à des structures simples.

Le simulateur du Dauphin (Epsitec, 2007) a été utilisé avec succès dans plusieurs classes. Il ne remplace pas la proposition qui suit, mais plusieurs éléments de cette proposition peuvent être économisés si le Dauphin a été pratiqué avant (dès 10-12 ans).

Le microcontrôleur est l'association du processeur, de suffisamment de mémoire "morte" et "vive" et de circuits d'entrée-sortie facilitant la commande de moteurs, la lecture de capteurs et les communications. Il s'est popularisé depuis 10 sans sous forme de cartes électroniques "Arduino" qui vont guider notre réflexion. Arduino intéresse les bricodeurs (makers) et est à la portée des jeunes; on l'a vu dans quantités de travaux de maturité. Arduino n'est pas vendu pour comprendre la programmation (vous achetez, vous câblez comme sur la photo, vous chargez le programme). Il utilise un C simplifié qui a fait son succès. Mais il a le potentiel qu'il faut pour aller en profondeur quand c'est nécessaire. Une recherche sur internet documente plusieurs exemples d'utilisation dans l'enseignement. La focalisation y est toutefois faite sur le matériel, sur le plaisir des élèves, plus que sur les objectifs pédagogiques.

2.2 Objectifs

Ce qui nous intéresse c'est le langage C, proche de l'assembleur et de son codage hexa que seul le processeur digère. Le but est de comprendre comment l'information est codée et est manipulée, et de se former à la discipline implacable de la programmation. Citons quelques notions à connaître et assimiler:

- Nombres binaires, hexadécimaux, BCD
- Mots de 8, 16, 32 bits, logiques et arithmétiques
- Dépassement de capacité, gestion des erreurs
- Constantes et variables, suite d'instructions traversantes ou bloquantes
- Organigrammes, structogrammes, machines d'état
- Monotâche, interruptions, système temps réel
- Structures ajoutées par les langages de "haut niveau"

La découverte des ces notions se fait en exécutant et analysant des programmes écrits en C qui utilisent quelques capteurs et affichages simples. Les programmes doivent être modifiés pas l'élève pour vérifier qu'ils ont été assimilés.

3 Scenario "Arduino"

L'environnement logiciel Arduino/C convient et est bien connu des "hackers", un enseignant trouvera facilement de l'aide dans son entourage. Quantités de jeunes s'y sont accrochés avec l'aide d'un vendeur. Le succès et la fabrication en Chine fait que l'offre est très large et les prix très bas. Un conditionnement pédagogique est toutefois nécessaire pour garantir l'utilisation sur plusieurs années. Le matériel consiste en une carte connectée à un PC via un câble USB. Une carte toute simple avec un petit affichage alphanumérique et graphique, des poussoirs et des LEDs permet de satisfaire quantités d'objectifs.
Il y a dans Arduino un aspect "IKEA" qui a fait son succès, mais qu'il ne faut pas suivre. Le C permet de coder simplement ce qui est simple, d'écrire des fonctions bien structurées et lisibles. Il permet plus, mais il faut se limiter à quelques concepts clés (listés plus haut et à mettre en forme), tester leur présentation avec des enseignants, documenter pour les profs et pour les élèves.

3.1 Autres solutions

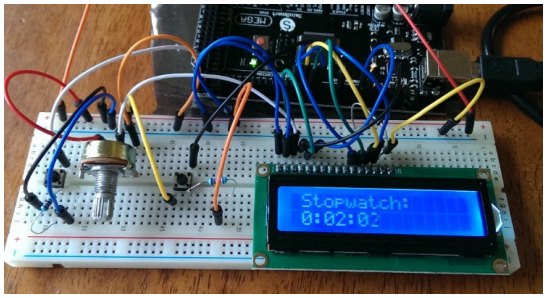
Pourquoi Arduino? Avec Raspberry on pourrait faire du Python. L'environnement est très lourd. On peut en cacher une bonne partie aux élèves, mais pas aux enseignants.
On a déjà un Thymio, on ne pourrait pas l'utiliser? Aseba cherche à cacher les structures simples qui nous intéressent, et il manque des capteurs qui font transition avec l'électronique.
Notre approche ici n'est pas de se dire "j'ai ce matériel, comment l'utiliser", mais de ce dire "j'aimerais enseigner tels concepts, quels outils utiliser". Et notre scénario s'imagine dans un contexte avec Thymio utilisé avant, et Python après pour des notions et algorithmes plus riches.

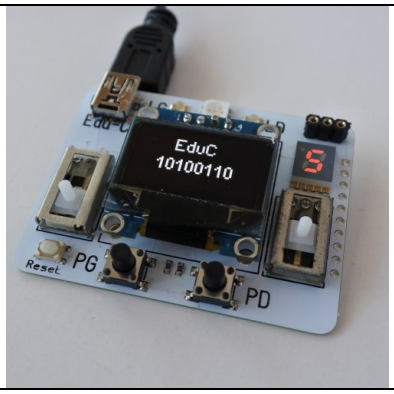
3.2 Éléments du scénario

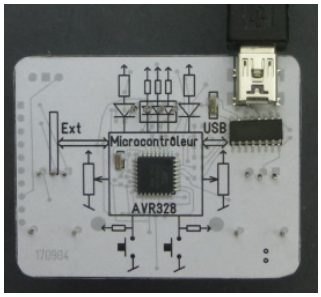
Les câblages Arduino sont incompatibles avec une utilisation en classe. Les modules peuvent se perdre, se dégrader si les élèves doivent les insérer. Un fil dans un mauvais trou est difficile à localiser. Développer un matériel optimisé est facile, et le coût avec conditionnement pédagogique est très faible (20-50CHF, pour 10-20 heures d'enseignement).

3.2.1 A quoi peut ressembler ce matériel?

La solution que l'on voit partout avec une carte à trous sur laquelle on insère des composants à programmer n'est pas acceptable. Une LED, un poussoir, un potentiomètre, consolident les notions de courant, tension, résistance, marges de bon fonctionnement, traitement par le processeur. Il faut un affichage pour "jouer" avec des nombres et des pixels.
Cette approche, excellente dans un atelier avec un petit groupe, n'est pas possible en classe. Arduino avec des "shields" est trop fragile. La carte compatible Arduino Edu-C permet les exercices traditionnels, sans perte de temps à câbler et à trouver les connexions fausses. Sur la face dessous, un voit un schéma-bloc des éléments.







3.2.2 Environnement logiciel

L'environnement de développement (IDE) Arduino est simple. mais il faudra améliorer les messages d'erreur. Chaque programme distingue bien le "setup", pour configurer l'application, et le "loop", qui code la tâche à exécuter. Les instructions for, while sont bloquantes, ce n'est pas le cas du if et du case. Avec si peu à apprendre, on a beaucoup à comprendre et exercer:

- Lire un poussoir, copier sur une LED. Statique et dynamique. Lire 2 poussoirs, ET, OU, NON.
- Signal physique (0-5V, rebonds) et logique (0-1). Temps de réponse et de traitement.
- Signaux analogiques: lire un potentiomètre, allumer une LED en intensité variable. LED couleur.
- Nombres 8bits, 16 bits. Positifs, signés, en virgule flottante. Dépassement de capacité.
- Ordre des bits et des octets. Problème des dates et des adresses.
- Fonctions, bibliothèques (librairies), structuration et documentation.
- Multitâche par balayage ou interruptions.
- Redondance, codage et transmission en série

3.2.3 Formation des enseignants et vérification de l'acquis des élèves

Arduino a été accepté par des personnes de toute formation et de tout âge. Le travail avec les élèves est essentiellement pratique, et ne suppose pas une connaissance des derniers outils informatique.
Les concepts à faire ressortir des manipulations doivent être clairement documentés pour les maîtres.
La vérification des connaissances se fait avec des "quiz" et la programmation d'algorithmes simples. Comme pour toute matière, une équipe de professeurs motivés devra intervenir.

4 Conclusion

La nécessité d'une formation proche du matériel est généralement acceptée. A quel âge, avec quel matériel? Le C avec l'environnement Arduino est facile à mettre en œuvre et apporte une bonne vision de ce qui se passe dans la machine. Beaucoup préféreraient du Python qui évidemment est plus "propre" et permet d'aller plus loin. Un environnement Python aussi facile d'accès que le C/Arduino sera probablement développé à terme. Il ressemblera au BBC-micro:bit? Les objectifs sont toutefois très différents.

jdn 171025