Dossier d'Alexander Strauch

Projet de SI: RUBiK'Solver

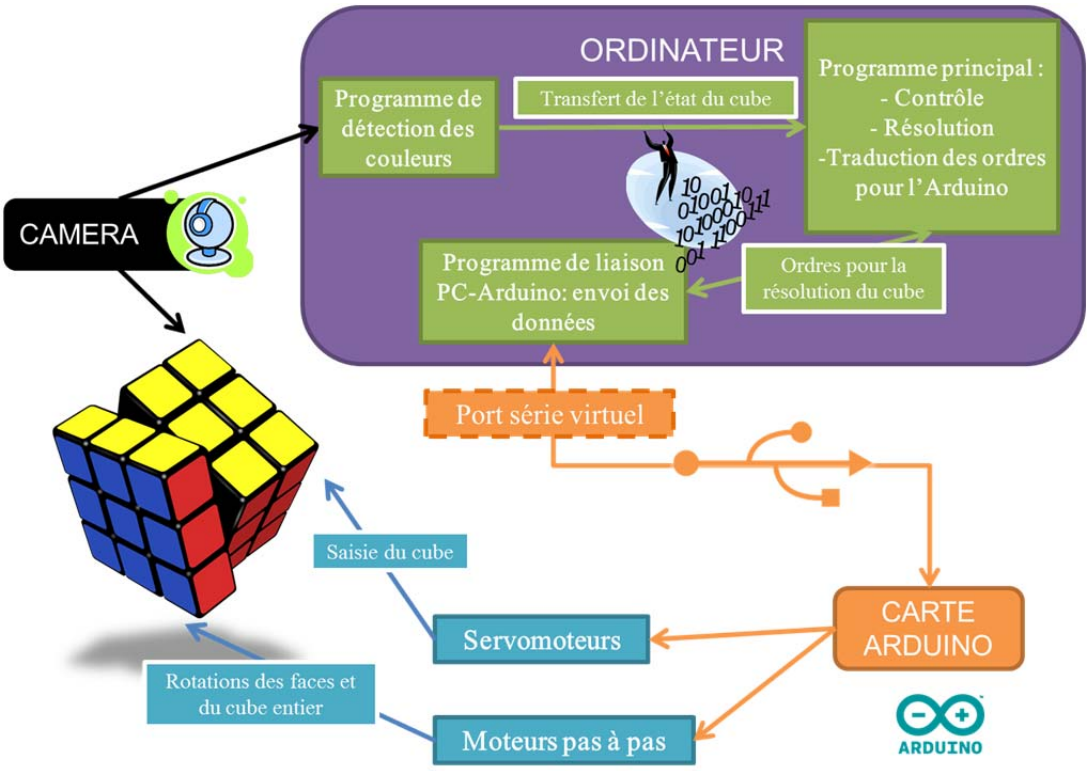
Résolution automatisé d'un Rubik's Cube

I) Présentation du projet

Pour le projet de SI de terminale, nous avons conçu et réalisé dans le cadre de ce dernier un robot capable de résoudre de façon totalement automatisée le célèbre Rubik's Cube. La machine devait remplir les objectifs principaux suivants :

- Etre capable de détecter les couleurs du Rubik's Cube une fois ce dernier inséré dans le système mécanique,
- Etre capable de résoudre le Rubik's Cube virtuellement,
- Etre capable d'effectuer les mouvements résolus précédemment.

Notre conception nous a alors permis d'aboutir à l'organisation globale de la machine suivante :



On constate que la machine possède son cœur à la fois dans la mécanique/électronique, mais aussi dans la partie informatique, où une grande partie de la résolution en elle-même du cube est gérée par un ensemble de programmes qui travaillent en synergie. Grâce à cette organisation, on constate que les objectifs précédemment définis sont remplis, les mouvements du cube par des moteurs spécifiques, la résolution par des programmes, et des transferts matériels pour la détection et les codes électroniques. La partie informatique possède donc une série de tâches importantes, et fait le lien entre réalité et monde de l'électronique.

II) Choix du langage et des librairies

Est vite venu le besoin de choisir quels outils utiliser pour réaliser toutes ces tâches. Connaissant relativement bien les langages C et C++, il paraissait tout naturel de chercher si l'ensemble de nos capacités dans ces domaines nous permettait l'utilisation d'outils adaptés. Pour l'analyse d'images, nous avons finalement opté pour la librairie OpenCV (Open Source Computer Vision), qui permet la récupération de flux vidéo branchés à un ordinateur (comme une caméra USB), librairie très adaptée au langage C.

Quant à la résolution du Rubik's Cube, elle découle plus de la logique que de fonctions complexes. Elle ne nécessite que de la manipulation de variables, de tableaux, de chaînes, etc. Bref, des fonctions simples incluses dans les librairies de base du langage C, avec stdio.h et stdlib.h. Pour le visuel, la manipulation d'images et l'organisation d'une fenêtre visuellement agréable, le choix s'est orienté vers la SDL, relativement lourde et peu optimisée mais simple de reprise en main.

Pour transfert les mouvements que doit réaliser la machine de l'ordinateur à la machine il a été nécessaire de passer par une carte électronique programmable. Nous avons choisi une carte Arduinio Mega puisqu'elle permet de commander facilement les moteurs que nous avons choisi (cf. paragraphe III)) mais également elle possède un port une sortie de type USB ce qui facilite la communication avec l'ordinateur cependant la communication est virtuellement identique à une communication série

Pour le transfert série, notre choix initial était l'inclusion de librairies spéciales en C, mais dont l'utilisation s'est rapidement avérée trop complexe. Nous avons finalement choisi le C++, possédant grâce à Visual Studio et le header stdafx.h, où la classe IO :: Ports :: Serial donnait des fonctions d'ouverture, de lecture et d'écriture très confortables pour la programmation. Côté carte électronique, elle se programme à l'aide d'un langage très proche du C, et contrôle grâce à certaines librairies incluses dans le logiciel de l'IDE les moteurs décrits dans la première partie.

III) Choix des moteurs

Un autre choix que nous avons dû faire très tôt est celui des moteurs que nous allions utiliser. En effet deux types de mouvements sont nécessaires à la résolution du Cube : il faut d'une part pouvoir saisir le Cube et d'autre par faire tourner les différentes faces ainsi que le Cube en entier. Pour cela nous avons choisi des moteurs pas à pas ainsi que des servomoteurs : les moteurs pas à pas assurent la précision nécessaire à la rotation des

faces alors que les servomoteurs permettent grâce à leur puissance de saisir le Cube mais aussi de le maintenir sous pression afin qu'il ne glisse pas des pinces.

IV) Les différentes liaisons électriques de la machine

La machine possède deux « circuit » différents : un premier qui permet d'assurer l'alimentation des différents composants et un autre qui permet de faire transiter les informations entre les différents ensembles.

Le circuit qui alimente la machine en électricité est divisé en deux parties :

-Une alimentation externe qui alimente uniquement la carte Arduino, nous avons fait ce choix afin de garantir que celle-ci dispose toujours de la tension nécessaire afin qu'elle ne se réinitialise pas ce qui aurait pour conséquence de stop la résolution sans que l'on puisse la reprendre.

-Un second circuit qui alimente l'ensemble des moteurs de la machine. Il est directement relié aux servomoteurs alors que pour les moteurs pas à pas il est nécessaire d'utiliser une interface de puissance.

Les liaisons d'information sont de trois différents types :

-La première liaison celle qui relie la carte à l'ordinateur, c'est un câble USB cependant la carte aussi bien que le PC reconnaissent cette liaison comme étant une liaison série du type RS232. Cette liaison permet de »synchroniser » les deux éléments au début de la résolution puis de transmettre du PC vers la carte les mouvements à réaliser et de la carte vers le PC d'une part la confirmation de l'exécution du mouvement précèdent mais aussi l'état des boutons.

-La seconde liaison est celle entre la carte et les moteurs. Ces liaisons permettent d'actionner les moteurs, chez les pas à pas on indique de combien de pas ils doivent avancer/reculer et chez les servomoteurs la carte indique quelle angle ils doivent atteindre puis maintenir.

-La dernière liaison va de la carte vers les DEL ce qui permet d'informer l'utilisateur si la machine peut commencer une résolution et d'autre part des boutons vers la carte pour pouvoir démarrer la résolution, la mettre en pause ou l'arrêter.

V) La carte électronique

La carte Arduino Mega possède trois types de pin mais seul deux nous intéressent sur ce projet. Les pins numériques sont utilisés pour actionner les moteurs pas à pas, allumer/éteindre les DEL et à capter l'état des boutons poussoirs. Le second type de pin

que nous utilisons est un cas particulier de pin numérique : ceux sont les PWM (Pulse Width Modulation ce qui signifie modulation de largeur d'impulsion). Ces pins sont nécessaires pour actionner les servomoteurs.

La carte possède également la possibilité de brancher un câble USB directement sur la carte mais utiliser cette possibilité bloque l'utilisation des pins 0 et 1.

VI) Programmation de la carte Arduino

Ce programme est programme est codé en Arduino un langage dérivé du C mais comporte les même caractéristiques (points-virgules à la fin des lignes, syntaxe de différentes fonctions...), les différences avec le C viennent des librairies qui ne sont pas compatibles avec la carte puisque celle-ci ne possède par exemple pas d'écran. Ce programme n'est pas « visible » de façon graphique, il est visible uniquement par les actions des moteurs/DEL/réaction de PC par rapport au boutons pressoir.

Une autre particularité de ce langage est la structure, celle-ci se décompose en 3 parties :

-L'initialisation : c'est ici que l'on inclue les librairies, défini les constantes et déclare les variables, ces 3 actions sont réalisé avec les même lignes qu'en C. On trouve ici une autre particularité de l'Arduino, en effet on déclare aussi des pseudos variables qui prennent différents arguments mais par la suite sont utilisé de façon semble aux autres variables.

-La « void setup() {} » : elle n'existe pas dans le langage C, elle n'est exécuté qu'une fois à savoir au démarrage de la carte. Cette fonction ne prend pas d'arguments et ne renvoie rien. Elle a pour but de paramétrer la carte : c'est ici qu'est défini si une broche est une sortie ou une entrée. Cette boucle permet aussi de réalisé toutes les instructions nécessaire afin de réaliser les actions qui permet de mettre la machine dans l'état initial.

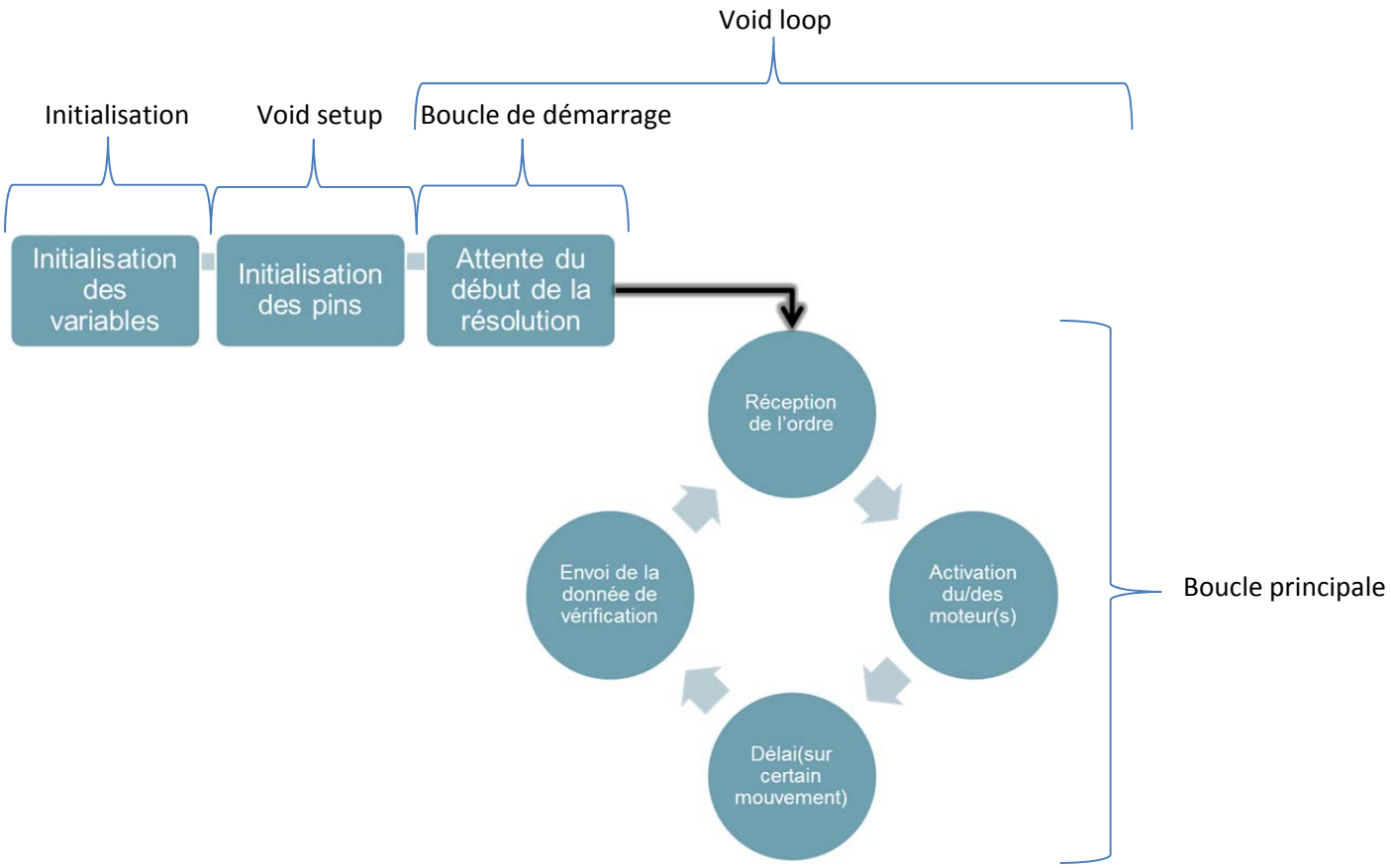
-La « void loop() {} » : cette boucle est exécuté en continu par le programme. En C cette boucle correspond au « int main () ; » et ne prend pas d'argument et ne renvoie rien. On y trouve l'ensemble du programme.

Le programme que j'ai créé pour la machine se trouve dans la « void loop(){} » et comporte différente partie :

-La boucle de démarrage de résolution qui attend d'avoir reçu du programme principale du PC un caractère de validation ainsi que « Start » soit enfoncé. Cette boucle relaie alors l'information que le bouton a été pressé pour lancer la résolution sur le PC. Puis elle referme les pinces sur le cube et éteint la DEL qui indique que la machine peut démarrer une résolution.

-La boucle principale se constitue d'une boucle « tant que » qui s'exécute tant que le caractère de fin de résolution n'a pas été reçu.

Le programme peut être résumé par l'organigramme suivant :



VII) Modélisation SolidWorks

Nous avons réalisé une modélisation complète de la machine (visible en annexe en orientation portrait) afin de définir facilement les dimensions des différentes pièces mobile de la machine mais aussi pour pouvoir modifier plusieurs fois la base de la machine. De plus la modélisation est nécessaire à l'usinage (sous Charly Robot) des pièces en PVC (polychlorure de vinyle) et à l'impression (avec l'imprimante 3D) des engrenages en ABS (acrylonitrile butadiène styrène).

VIII) Conclusion

Ce projet au-delà de l'épreuve qu'il constitue m'a permis d'avoir un avant-gout de ce qu'est le travail d'ingénieur et de vivre une grande aventure de groupe qui a permis de développer les compétences de chacun dans tous les domaines.

IX) Annexe