Bioloid Commande par Arduino

GRUSON QUENTIN / RAZAFINDRAIBE JORDAN

Projet IMA4 / 2015-2016 / Encadrant : Blaise Conrard

Table des matières

Introduction

Les servos-moteurs pour Bioloid font partie de la gamme Dynamixel, une gamme de servos-moteurs programmables parmi les plus efficaces du marché. Ils sont à la fois petits, légers, puissants et très robustes, ce qui permet de développer des applications robotiques complexes.

Nous avions, en IMA3, construit des robots Bioloid en nous basant sur l'utilisation du logiciel Robotis. Ce logiciel permettait, via un système de programmation simple type grafcet, de programmer l'interface logique CM5 qui contrôlait les servos-moteurs.

Ce projet propose d'aller plus loin en programmant directement les servos-moteurs via une carte Arduino.

Nous commencerons par vous présenter le projet, ensuite nous parlerons des différentes étapes qui nous ont permis d'arriver à une solution et enfin nous exprimerons notre ressenti par rapport au travail effectué.

1. Présentation du projet

1.1. L'objectif du projet

L'objectif de ce projet est de développer une application en nous servant d'une carte Arduino pour contrôler les servos-moteurs Bioloid.

L'interface habituelle de programmation de ces moteurs via le logiciel Robotis est certes simple mais assez limité. En passant par une carte Arduino, nous pourrons développer une application avec des possibilités de programmation plus étendues.

En effet, nous pourrons inclure des fonctions électroniques complexes, mieux gérer le microcontrôleur, programmer directement en C/C++... Ce qui permettra de développer une application plus complète.

1.2. Description du projet

L'utilisation de la carte Arduino en remplacement du CM5 demande une adaptation électronique et informatique de l'Arduino. Il sera donc nécessaire de concevoir un circuit électronique et une méthode de programmation informatique s'inspirant du comportement du CM5.

Nous avons également dû réfléchir à une application démonstrative de notre travail. Après avoir discuté de plusieurs idées avec notre tuteur : détection d'objets, surveillance d'une salle... Nous avons finalement décidé de développer une horloge robotique.

De ce fait, nous nous sommes basés sur l'idée de la création d'un bras robotique qui, toutes les minutes écrira l'heure puis l'effacera.

1.3. Matériel utilisé

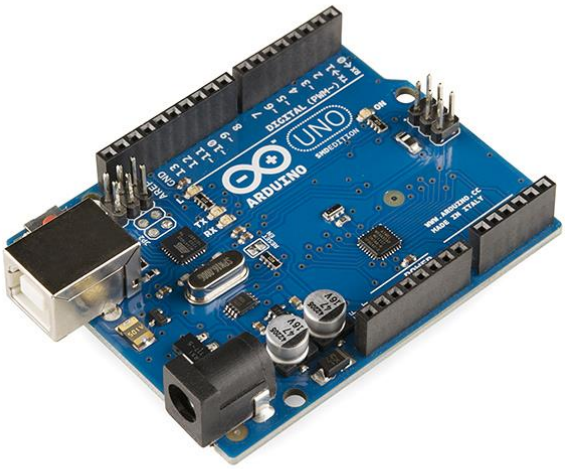
Nous avons évidemment eu besoin d'une carte arduino, le tuteur nous a ainsi fournit une carte type Arduino Uno.

Nous avons initialement eu besoin d'un buffer trois états SN74LS241N pour adapter électroniquement l'arduino aux servos-moteurs. Nous avons pour cela commandé ce composant qui nous a été remis deux semaines après le début du projet.

Nous avons utilisé 6 servos-moteurs pour notre application d'horloge, ces servos-moteurs étaient inclus dans le kit Bioloid remis par Mr Conrard au début du projet.

Ensuite nous avons conçu notre application sur une ardoise magique facilement effaçable.

Enfin il a été nécessaire d'utiliser deux boutons pour gérer l'effacement de l'heure, ces deux boutons ont étés récupérés en démontant une souris inutilisée.



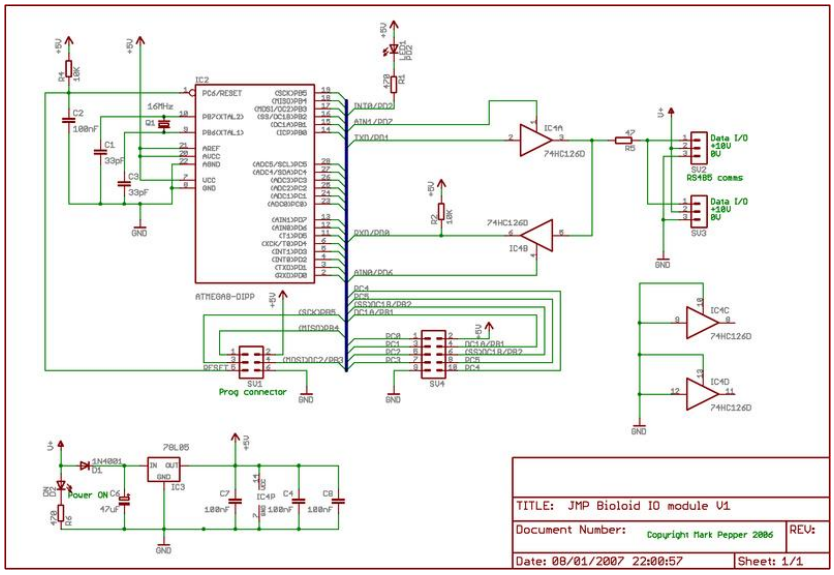
2. Déroulement du projet

2.1. L'adaptation de l'Arduino aux servos

2.1.1. L'adaptation électronique

La première étape de notre projet a été d'étudier minutieusement le comportement électronique de l'unité CM5 avec les servos-moteurs. Ceci dans le but d'adapter électroniquement l'Arduino aux servos.

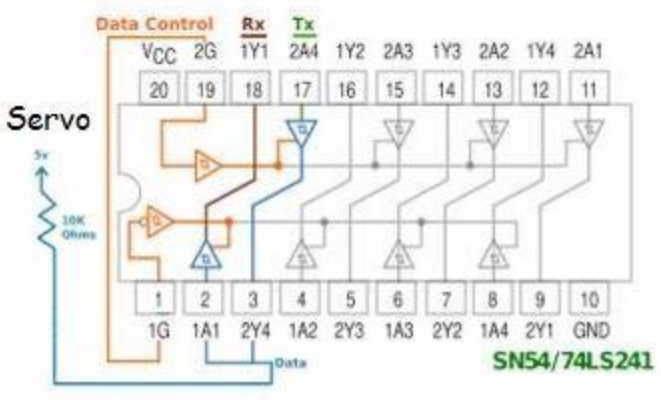
Nous avons pour cela étudié la datasheet suivante :

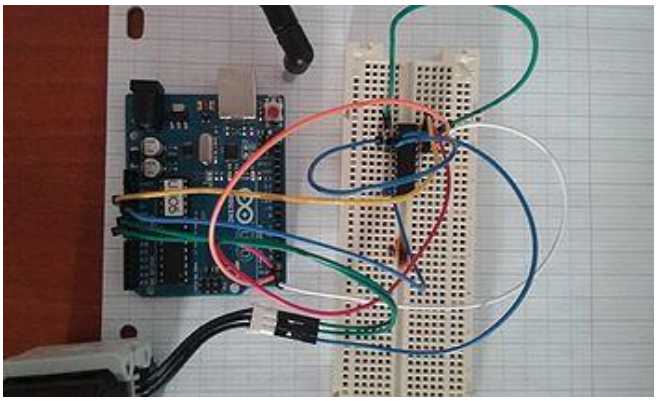


Nous avons remarqué que la communication avec les servos se faisaient via un système de transmission/réception des données avec les servos (TX/RX).

Nous avons compris que cette méthode de communication était également possible via les ports TX/RX de l'arduino. Le seul souci étant de gérer les deux informations sur un seul bus de données (comme on peut le voir ci-dessus, le servo a trois fils : deux pour l'alimentation et un unique pour les données).

Nous avons donc, comme le préconisait la datasheet, utilisé un buffer trois états pour contourner ce problème. En lieu et place du 74HC126D du CM5, nous avons préféré utilisé un SN74LS241N (celui-ci étant disponible plus facilement) dont le comportement est le même.





2.1.2. L'adaptation informatique

Une fois cette adaptation électronique réalisée, nous avons dû réfléchir à une façon de programmer les servos-moteurs en vitesse et en position.

Avec l'arduino, il est possible de programmer en C ou C++. Nous avons choisi de développer en C++ afin de mettre en pratique les notions acquises en cours de « conception logicielle à objets ». Nous avons ainsi créé une classe de servos-moteurs avec ses différents attributs.

Cette méthode de programmation nous a permis de modifier directement les attributs de la classe servomoteur en différentes fonctions.

Pour l'écriture des fonctions de commande, nous avons trouvé des informations intéressantes sur le site dédié à l'Eurocup 2012 :
https://github.com/7Robot/Eurobot-2012/tree/master/Arduino

Nous avons également trouvé des informations sur le contrôle des servos-moteurs en CLO sur le site de Savage Electronics :

http://savageelectronics.blogspot.fr/2011/08/actualizacion-biblioteca-dynamixel.html

Nous avons appliqué ces méthodes de développement à notre cas afin de contrôler aisément les servos-moteurs en vitesse et en position.

Enfin les fonctions de traitement des fonctions ont été développées directement sur l'IDE de l'arduino.

Cette méthode nous a permis d'utiliser simplement les méthodes setup() et loop() de l'IDE pour développer notre application d'horloge robotique, à l'aide des différentes fonctions d'écritures, de gestion de l'heure … Que nous verrons par la suite.

2.2. La gestion du temps

Notre application étant une horloge, il est indispensable de connaître l'heure en temps réel afin de pouvoir l'écrire ensuite.

Pour cela, nous avions initialement songé à utiliser un module RTC (Real Time Clock). Ce type de module calcule l'heure en temps réel, il suffit juste de la récupérer puis de l'afficher. Malheureusement, notre tuteur n'a pas été en mesure de nous en remettre un et, les commandes étant très longue à arriver, nous avons préféré développer une autre solution pour connaitre l'heure.

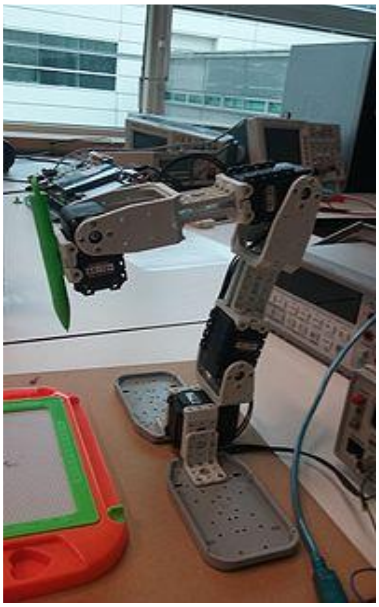
Nous avons donc utilisé la fonction millis() qui compte une fois l'application lancée. Le principal problème de cette fonction et qu'elle utilise le processeur constamment, ce qui pourrait être problématique. Cependant, cela ne constitue pas un problème pour notre application étant donné que notre arduino ne fait que compter l'heure et gérer les mouvements des servos.

A partir de ce comptage, on décompose l'heure toutes les 60 secondes avec l'utilisation d'un tant que. On connaît donc directement l'heure à partir de celle spécifiée au départ.

Le principal inconvénient de cette méthode est qu'il faut à chaque fois spécifier l'heure de départ dans le programme si l'on éteint l'horloge (l'arduino compte à partir de la valeur de départ), ce qui nécessite un nouveau téléversement du programme vers l'arduino.

2.3. La construction du robot

Après la gestion du temps, nous avons pu songer à la construction du robot. Pour ce faire, nous nous sommes tout simplement inspirés des articulations autour d'un bras humain.



En plaçant un moteur à l'horizontale (le moteur le plus en bas), et les autres à la verticale, nous pouvions atteindre chaque point situé à une distance inférieur au bras tendu. Afin d'écrire sur l'ardoise magique, nous avons collé le stylet sur une pièce Bioloid, elle-même vissée sur le dernier moteur (moteur le plus haut).

Cependant lors de mouvement complexe et trop rapide, le robot basculait, le centre de gravité étant trop haut. Pour résoudre ce problème, nous avons cloué le robot sur une planche en bois. Cette planche en bois de 50 par 40 a été coupée à la découpeuse laser au FabLab de Polytech. L'ardoise ainsi que les moteurs servant à effacer, que nous verrons plus tard, ont également été fixés sur la planche.

Une fois le robot stable et l'ardoise immobile, nous avons programmé les moteurs afin de pouvoir écrire des chiffres.

2.4. Programmation des moteurs

Pour la programmation des moteurs et notamment l'écriture des chiffres, nous avons créé une position de base qui nous servirait de repère. Cette position initiale est exécutée au lancement du programme arduino, puisque les commandes sont situées dans le *setup()*.

Ensuite nous sommes partis du fait que l'ardoise devait être divisée en quatre cadrans : un pour l'écriture des unités des minutes, le second pour les dizaines de minute, le troisième pour les unités des heures et le dernier pour les dizaines d'heure. En utilisant principalement le moteur du bas, il était aisé de distinguer ces 4 cadrans.

Pour l'écriture des chiffres nous avons choisi de suivre le même modèle qu'un afficheur 7 segments. De ce fait, nous avons programmé chaque mouvement possible, c'est-à-dire aussi bien les mouvements verticaux de bas en haut et de haut en bas, que les mouvements horizontaux de gauche à droite et de droite à gauche. Une fois ces mouvements créés nous les avons utilisés à la suite afin de réaliser chaque chiffre de 0 à 9. Après l'écriture du chiffre du dernier cadran, nous avons programmé les moteurs pour qu'ils puissent, comme dans une vraie horloge digitale, afficher deux points afin de séparer les minutes des heures.

La synchronisation des mouvements a notamment été réalisé grâce à une fonction *attente_ecrire()* qui permettait de laisser du temps au moteur pour réaliser leur rotation.

Voici un exemple de l'affichage :

Sur cette image, nous lisons qu'il est 15h41.

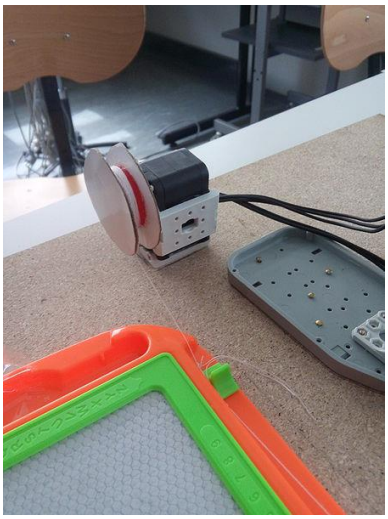
2.5. Gestion des effaceurs

L'effacement de l'heure n'a pas été aussi simple que prévu. Plusieurs idées ont étés mise en place. Tout d'abord, nous voulions équiper notre robot d'un deuxième bras pour gérer indépendamment l'effacement. Ensuite est venue l'idée d'utiliser le bras existant pour pousser l'effaceur à l'aide du crayon. Cette perspective n'a pas été retenue car elle risquait d'endommager le matériel, et plus particulièrement de décoller le stylet.

Nous avons alors émis la possibilité d'utiliser un fil, attaché à l'effaceur. Il suffisait de fixer un fil sur un moteur et un effaceur, puis par un mouvement de rotation, le fil tirerait l'effaceur.

Nous avons donc opté pour cette solution et de ce fait, placé deux moteurs supplémentaires de part et d'autre de l'ardoise. Pour enrouler et dérouler le fil, nous avons fabriqué des bobines. Ces dernières ont été réalisées avec deux cartons et un bouchon.

Pour vous faire une image plus précise de la construction, vous trouverez ci-dessous un des effaceurs.



Ensuite nous nous sommes posé la question suivante : Comment gérer la marche et l'arrêt du moteur afin que l'ardoise s'efface entièrement ?

Dans un premier temps, nous avons commandé les moteurs temporellement. C'est-à-dire que nous les faisions tournés pendant une durée précise. Pendant que l'un tirait le fil, le second donnait du mou. Toutefois, cette méthode temporelle n'était pas adaptée à un système comme le nôtre car la position de départ des moteurs ainsi que la raideur du fil, ne sont jamais initialisés pareillement.

Dans un second temps, nous avons utilisé des fins de courses. Malheureusement, aucun n'était disponible dans l'école. Nous avons alors démonté une souris afin de récupérer ses boutons qui ferait office de fins de courses.

Ces boutons, que nous avons par la suite soudés et branchés sur les ports de l'arduino, permettaient de détecter la fin de l'effacement. En effet, nous avons placé une tige sur l'effaceur et disposés les boutons de part et d'autre de l'ardoise.

De cette manière, lorsque l'effaceur arrivera au bout de son trajet, il appuiera sur le bouton qui enverra un signal à l'arduino. Malheureusement, les fins de courses n'ont pas pu être fixées à temps donc nous devons appuyer manuellement sur les boutons poussoirs pour arrêter l'effacement.

3. Impressions personnelles sur le projet

3.1. Gestion du projet

Pour ce projet, nous avions un semestre pour concevoir complétement notre solution afin qu'elle corresponde complétement au cahier des charges.

Ce fût très enrichissant de concevoir des solutions afin d'adapter l'arduino aux servos-moteurs Bioloid. De plus, la liberté que nous avions pour l'application nous a permis d'établir une solution ludique et assez originale.

Ensuite, la répartition du travail en fonction des différentes compétences que nous avions acquises (Jordan en SA et Quentin en SC) nous a permis de développer des solutions complètes que ça soit pour la partie « communicante » (programmation de l'arduino, conception du circuit électronique…) que pour la partie « autonome » (construction des moteurs, gestion des sources d'énergie…).

Le seul bémol que nous pouvons admettre et la mauvaise gestion du matériel. Nous n'avons pas réussi à récupérer de RTC, ni de fin de courses correcte… Ce qui est vraiment dommage car cela nous aurait permis de terminer complétement notre projet.

Mais ce n'est qu'expérience pour la suite, nous réfléchirons mieux à la gestion du matériel plutôt que de nous lancer tout de suite dans le code ou la construction du robot.

3.2. Bilan

Nous sommes satisfaits de notre travail. Nous avons respecté le cahier des charges, et développer une application qui fonctionne quasiment parfaitement.

Cependant, nous pouvons noter une certaine déception vis-à-vis du fait que nous n'avons pas réussi à rendre complétement autonome notre application. Une meilleure gestion du temps et du matériel nous aurait permis d'aboutir à une solution complète.

Pour faire fonctionner l'application, il suffit de brancher la carte Arduino à un adaptateur secteur 12V pour alimenter les servos-moteurs. L'horloge robotique va ainsi écrire l'heure à partir de l'heure spécifiée dans le programme.

Conclusion

Pour conclure, ce projet de fin de quatrième année nous a été très bénéfique. Il nous a permis d'approfondir nos connaissances sur les systèmes embarqués et de mettre en pratique directement ce que nous avons appris en cours.

Ce projet nous a également appris à suivre le cycle d'un projet, c'est-à-dire de la conception à la réalisation. Nous devions respecter un planning d'environ 6 mois pour tout réaliser. Il fallait donc être organisé et bien réfléchir avant la mise en place d'une solution.

Ce projet nous a permis de développer des valeurs essentielles pour notre avenir professionnel. En effet, le travail en équipe, le partage des idées et des tâches a été fondamental pour la concrétisation de notre projet.

Nous tenons à remercier notre encadrant Mr Blaise Conrard pour nous avoir proposé le sujet. Nous tenons également à remercier Xavier Redon, Alexandre Boé, et enfin particulièrement Thierry Flamen pour leurs aide et conseils durant ce projet.

Annexe

Principales fonctions utilisées

```
void setup()
{
  pinMode (bouton1, INPUT) ;
  pinMode (bouton2, INPUT) ;

  Dynamixel.setEndless(4,OFF);
  Dynamixel.setEndless(3,OFF);
  Dynamixel.begin(1000000,2);

  position_initiale (position_2,position_5,position_6,position_13) ;
}

/*
Le lancement du programme exécute en tout premier la boucle setup().
Nous définissons donc, dans cette boucle, l'entrée des pins 8 et 12.

Ces pins servent à gérer l'effacement vu précédemment. Nous définissons
également la communication avec les servomoteurs par le biais de la
fonction Dynamixel.begin(1000000,2).

A la fin du setup(), nous plaçons le robot dans une position initiale
qui sera la position dans laquelle le robot se placera lorsqu'il ne
fera rien.
*/
```

```
void loop()
{

  unsigned long dateCourante = millis();
  unsigned long intervalle = dateCourante - dateDernierChangement;

  gestion_heure () ;    //Gere le passage

    if (intervalle > SECONDE)
    {

       effacer () ;

       ecrire(1,u_min);
       ecrire(2,d_min);
       ecrire(3,u_heu);
       ecrire(4,d_heu);

        dateDernierChangement = dateCourante ;
        minute++;
        u_min++;
        if(u_min>9)
        {
          u_min=0;
          d_min++;
        }
    }
}

/*
Cette fonction loop(), est le cœur du programme.
Grâce à la variable intervalle, nous pouvons connaître précisément, de
l'ordre du millième de seconde, le temps écoulé.

En comparant cette durée à une minute nous exécutons la boucle if.
Cette boucle efface, puis écrit l'heure.

DateDernierChangement étant défini comme une variable globale et
intervalle remis à zéro lorsque nous passons dans la boucle if, nous
pouvons dire que nous écrivons bien en temps réel.
*/
```