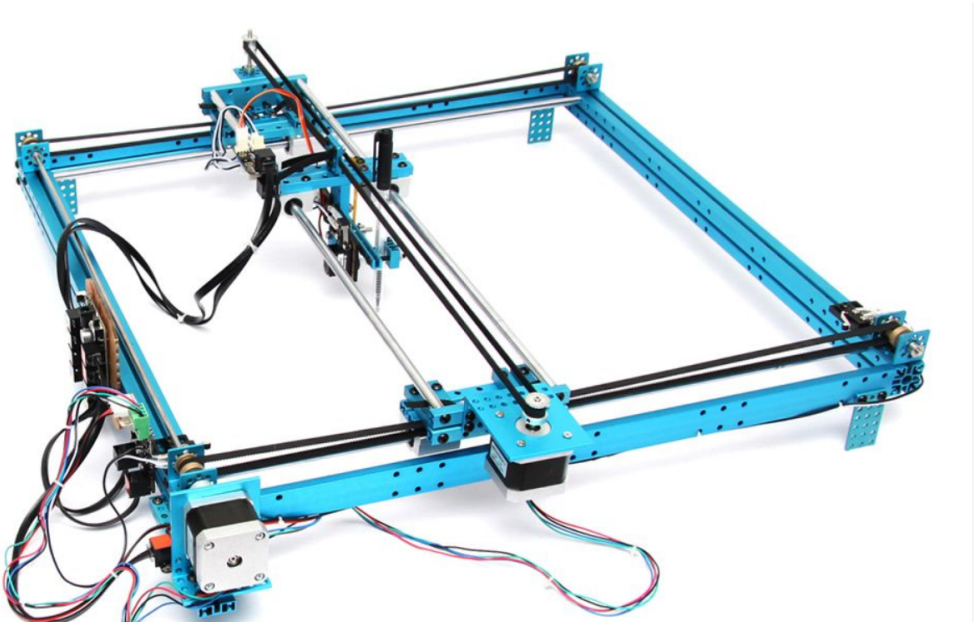
MARCEAU Dimitri
MONTEUX Gabriel
LECARME Gabriel
MEHMED Nesin
Institut d'Optique Graduate School

2020/2021

Rapport Technique Projet IETI



Introduction :

L'Automatique est la discipline qui développe les méthodes et les moyens relatifs à la commande des systèmes. Elle concerne une grande majorité de métiers aujourd'hui et tout ingénieur se doit d'en posséder une culture minimale. L'Automatique fait appel aux mathématiques, au traitement du signal, à l'informatique et aux connaissances des divers domaines d'application. Cette discipline est indispensable pour analyser, concevoir, simuler, optimiser, valider et vérifier les systèmes technologiques et socio-technologiques qui sont amenés dans la prochaine décennie à devenir de plus en plus interconnectés, avec un traitement d'énormes quantités de données et d'information, et avec de nouvelles formes de synergie entre les humains et les systèmes technologiques.

En cela, nous avons pris la décision de travailler sur le système deux axes, qui fait appel à beaucoup de compétences techniques essentielles pour nous, comme la programmation ou encore l'élaboration de montage électronique complexe pour le bon fonctionnement de

notre système. On s'appuie sur le langage C afin d'élaborer les différentes instructions qui permettront de réaliser nos objectifs.

On s'interrogera sur la question suivante :

Comment peut-on réaliser un système autonome qui permet de déplacer un objet d'un point à un autre suivant des instructions envoyées depuis l'ordinateur ?

Objectifs :

Notre objectif est de récupérer un objet aimanté, de le déplacer vers un autre point afin de le déposer. Tout cela devra être fait sans intervention extérieure.

On devra donc réaliser un système qui pourra se rendre à un point de la table à partir des commandes envoyées depuis l'ordinateur, puis de récupérer l'objet avec le servomoteur et l'électroaimant, qui permettent respectivement de se mettre à la bonne altitude et lorsque l'aimantation sera active d'attirer l'objet aimanté. Toutes ces commandes devront être envoyées depuis l'ordinateur vers la carte préalablement, afin que la procédure puisse se dérouler de manière totalement autonome.

Afin d'y voir plus clair, nous avons élaboré le schéma en bloc ci-dessous (figure 1), qui décrit de manière synthétique et précise tous nos objectifs intermédiaires, qui mis bout à bout, permettent de réaliser l'objectif final.

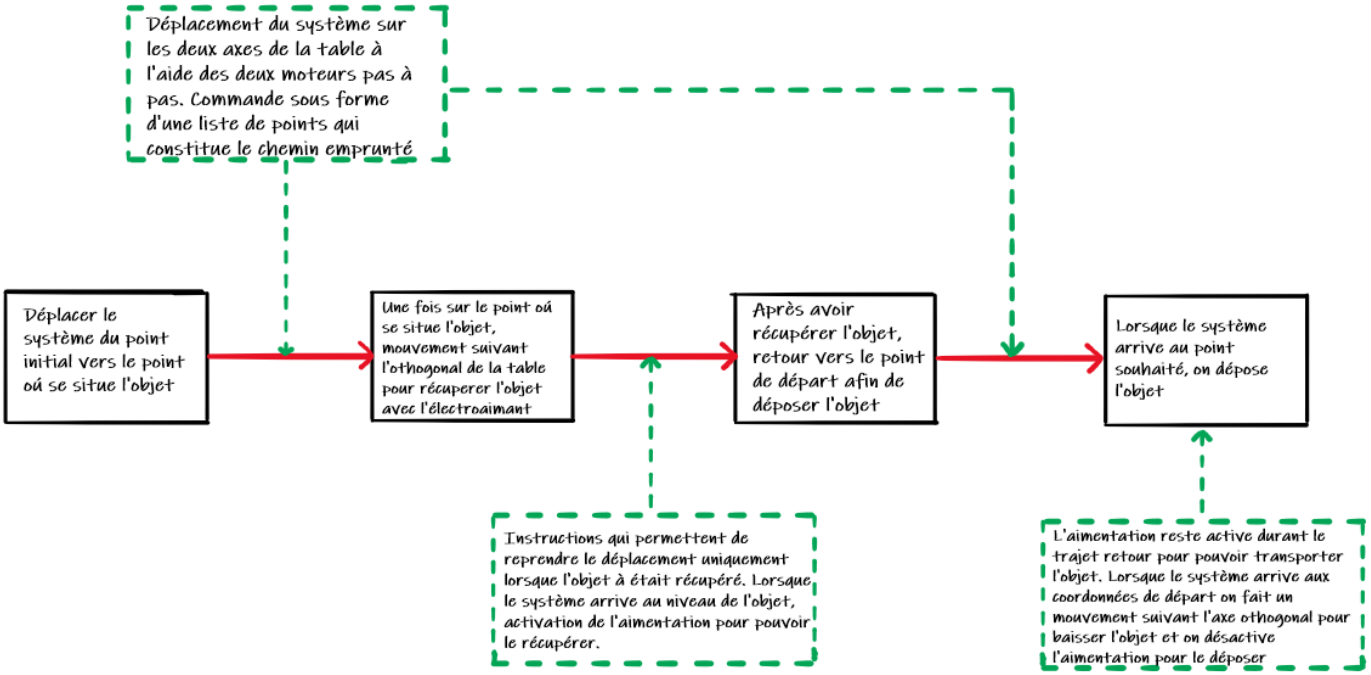


Figure 1 : Description du fonctionnement du système de positionnement

Réalisation du prototype

Pour que le système puisse réaliser les différents objectifs, il fallait également élaborer un schéma électronique, qui répond à chacune de nos demandes. Ainsi, pour le déplacement suivant le plan de la table on a pris deux moteurs pas à pas, que l'on pilote avec les deux cartes L297 et L298(explication en détail plus bas, section : **Déplacement suivant le plan de la table**). Puis, pour le déplacement suivant l'axe orthogonal à la table, on s'aide d'un servomoteur (explication en détail plus bas, section : **Déplacement suivant le plan orthogonal à la table**). Et enfin, on utilise l'électroaimant afin de récupérer notre objet, de le transporter, puis de le reposer à l'endroit souhaité (détail dans la section : **Utilisation de l'électroaimant afin de récupérer et déposer l'objet aimanté**). Tous ces composants sont reliés à la carte Nucléo (moteur par l'intermédiaire des cartes) qui va commander chacun d'entre eux, en envoyant les bonnes commandes au bon moment, que l'on aura préalablement entré sous forme de code sur MBED.

Ci-dessous le schéma électronique de notre système :

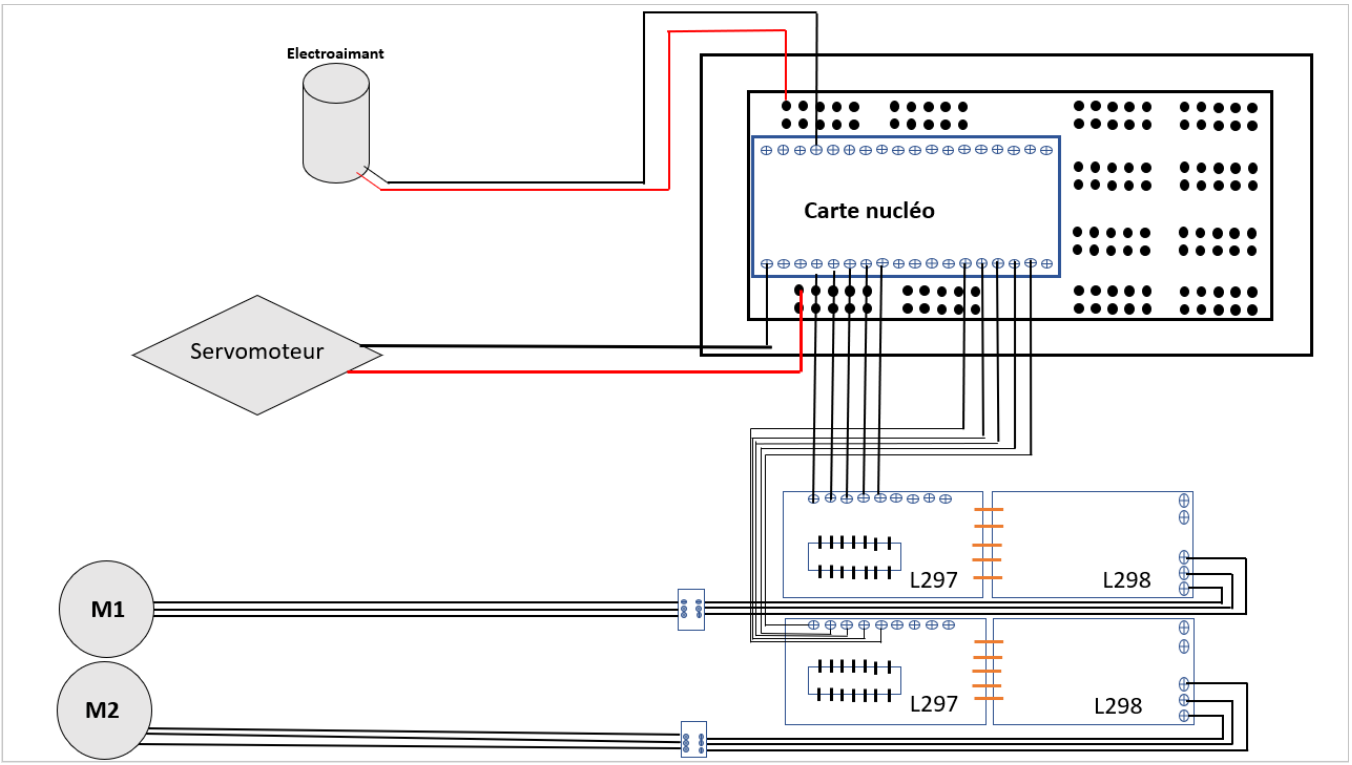


Figure 2 : Montage électronique de notre système

I. Déplacement suivant le plan de la table

Dans un premier temps nous avons commencé par piloter qu'un seul moteur pas à pas. Les déplacements que nous avons ordonnés au début était un déplacement continu et sans arrêt. Lorsque nous avons réussi à faire tourner le moteur nous avons pris les cartes L297 et L298 pour ainsi former un pont en H afin de piloter les moteurs. En effet, pour piloter un moteur il faut être capable d'inverser le sens du courant.

Ceci est possible à l'aide des ponts en H qui correspondent à la mise en série et en parallèle de quatre transistors du même type.

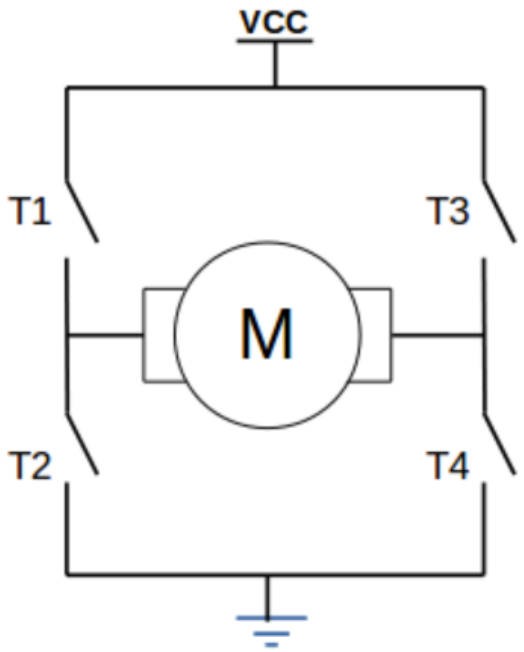


Figure 3 : Schéma d'un pont en H avec la mise en série et parallèle de quatre transistors

Les deux moteurs pas à pas étaient donc pilotés à l'aide des ponts en H formés par les cartes L297 et L298 que nous avions à disposition. La deuxième carte permettait aussi de générer automatiquement les signaux nécessaires au fonctionnement des moteurs. Nous avions alors seulement besoin d'un ticker qui permet des interruptions à temps fixe du programme; de deux signaux d'horloge (un par moteur) et de deux variables numériques pour le sens de rotation. Dans un premier temps, le code que nous avions établi permettait uniquement de faire des mouvements suivant l'axe Ox puis des déplacements suivant l'axe Oy. Nous avons calculé la vitesse de déplacement de notre système afin de pouvoir implémenter un temps de fonctionnement qui permet d'avoir le déplacement souhaité. Cette méthode est évidemment approximative de par: la mesure de la vitesse; le fait de considérer le profil de vitesses uniforme sur le déplacement… Néanmoins le codage du 2 axes en position n'étant pas absolu (on pilote le chariot sur des temps de déplacement et non d'une position à une autre) cela ne posait pas de problème pour l'usage qu'on faisait du système. Pour l'améliorer on pourrait imaginer asservir le système en position, le codage deviendrait absolu. Mais il faudrait pour cela mettre en place l'acquisition d'une valeur à mettre dans la boucle de retour, à l'aide d'une caméra par exemple et c'est beaucoup plus complexe.

On pouvait donc se permettre d'aller vers n'importe quel point en connaissant le nombre de points nécessaires à effectuer suivant Ox et suivant Oy. Mais, le chemin que nous empruntions n'était pas le plus court et le plus judicieux en termes d'économie d'énergie.

Nous avons fini par établir le dernier programme qui nous a permis de réaliser des déplacements suivant les deux axes mais également suivant la diagonale, pour permettre au système de se rendre le plus rapidement possible à l'endroit souhaité.

II. Déplacement suivant la perpendiculaire au plan de la table

Pour effectuer le mouvement suivant la perpendiculaire au plan de la table nous avons utilisé un servomoteur contenant un moteur à courant continu, un capteur de position angulaire ainsi qu'un contrôleur numérique. Avec la carte nucleo on envoi un signal de commande de type rectangulaire avec une période donnée ainsi qu'une durée, qui correspond à la durée au temps haut. C'est cette durée qui permet de modifier l'angle de sortie du servomoteur.

Voici un schéma de principe avec une période de 20ms et la durée temps haut qui prend différentes valeurs.

On voit que la durée du temps haut correspond à une position angulaire particulière. Par exemple, on observe que la position angulaire de 0 degré correspond à un temps haut de 1,5 ms.

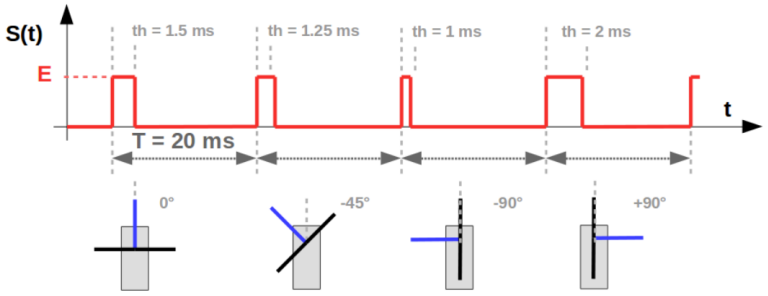
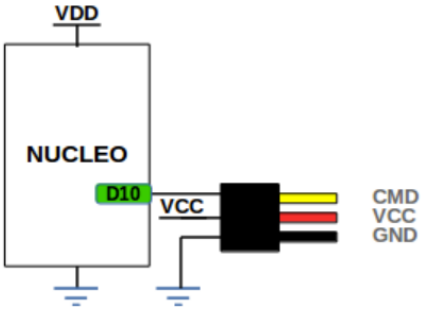


Figure 4 : Principe de fonctionnement d'un servomoteur



Le montage reste assez simple à effectuer car il n'y a pas beaucoup de branchement (seulement trois). En témoigne le schéma de la figure 5 :

Figure 5 : Câblage du servomoteur

Le programme de base n'était pas très convenable pour l'application que nous voulions faire du servomoteur. En effet, dans un premier temps nous avions réussi à faire des rotations suivant les deux sens, avec des angles assez importants pour permettre à l'électroaimant de descendre à un hauteur suffisamment faible pour récupérer l'objet aimanté. Le problème était sur le fait que le programme ne s'arrêtait pas et que les commandes étaient réalisées en boucle. Or, nous voulions que le servomoteur fasse descendre l'électroaimant prendre le

temps afin que l'électroaimant puisse récupérer l'objet et ensuite qu'il fasse remonter l'électroaimant et le garder en position haute durant toute la durée du trajet retour.

Pour ça nous avons introduit les instructions du servo dans les boucles conditionnelles déjà existantes du code et ajouté un compteur pour savoir à quelle étape de la manœuvre nous en étions.

III. Utilisation de l'électroaimant afin de récupérer et déposer l'objet aimanté

Pour le bon fonctionnement de l'électroaimant nous avons besoin de l'alimenter avec une tension de 12V. Or, la carte nucléo délivrant uniquement 5V au maximum, on s'aide d'un transistor MOSFET, commandé en tension, afin d'amplifier la tension que va recevoir notre électroaimant. Voir figure ci-dessous :

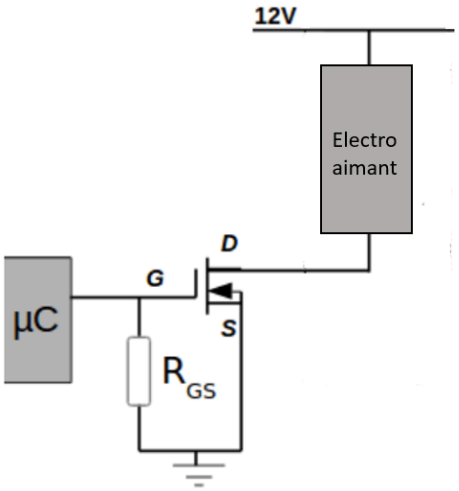


Figure 6 : Montage électronique de l'électroaimant

Valider et caractériser le système final

Le premier test à réaliser est de s'assurer que la table traçante se déplace bien d'un point A à un point B déterminé préalablement pour pouvoir récupérer un objet et ensuite puisse revenir à son emplacement initial. L'emplacement initial étant défini au départ de la table traçante. On code les coordonnées de notre objet et on se place à un endroit bien particulier dans le plan de la table où nous serons certain de récupérer l'objet. Il faut également être capable de déplacer la table transversalement, longitudinalement et diagonalement.

Une fois la table traçante capable d'aller à notre objet, il faut s'assurer qu'elle puisse récupérer l'objet pour le ramener.

Le second test à réaliser est alors de s'assurer du fonctionnement du servomoteur, qu'il descende et remonte à la bonne hauteur pour pouvoir coller l'aimant à l'objet. On rappelle qu'on ne peut pas prendre n'importe quelle taille d'objet pour que l'aimant puisse bien se fixer. Un objet assez large mais pas très lourd pour que sa fixation à l'aimant soit optimale. Lors du trajet retour, on s'assurera que l'aimantation reste effective et que le servomoteur ne tourne plus. On s'assurera d'un temps d'arrêt entre la descente/ la remontée du servomoteur et le démarrage des moteurs pour effectuer le trajet retour.

Le dernier test à réaliser est de s'assurer que l'objet soit déposé lors du retour à la position initiale. Encore une fois, si l'objet est trop léger, il est très probable que l'aimantation ne disparaisse pas d'un coup et que l'objet ne soit pas déposé, même pendant un temps d'arrêt de 5s. Une fois l'objet déposer, si l'on souhaite récupérer un autre objet il faudra définir un autre objectif de coordonnées finale.

Planning des séances

Séance 1	-Piloter un moteur seul (réussi) -Trouver des idées de ce qu'on voulait faire réaliser à la table traçante -Réalisation du cahier des charges
Séance 2	-Réalisation du planning -Réalisation du schéma fonctionnel et du plan de formation -Répartition du travail pour les prochaines séances -Utilisation de la carte nucléo pour générer les créneaux
Séance 3	- Réaliser un circuit électronique permettant de faire fonctionner les deux moteurs pas à pas ensemble. -Comprendre le fonctionnement des deux cartes L297 et L298 pour les utiliser dans notre circuit avec les deux moteurs. -Réaliser un code sous MBED pour avoir des déplacements suivants les deux axes de la table (Déplacement non diagonal pour le moment) Il faudra en particulier essayer d'avoir des instructions qui permettent de stopper le programme après un temps donné. Et donc s'affranchir du contrainte du moteur qui tourne en continu.
Séance 4	- Réaliser un code qui permet d'effectuer un déplacement en diagonal. - S'occuper du pilotage de servomoteur, avec le code et le câblage. - Commencer à réfléchir sur le code qui pourra permettre au stylo de faire un déplacement précis. C'est-à-dire d'un point quelconque vers un point que l'on aura préalablement fixé.
Séance 5	- Finir le code commencé au séance 4 pour permettre au stylo de pouvoir se déplacer vers n'importe quel point du plan avec précision. Donc pouvoir réaliser un dessin qui possède la forme d'une figure géométrique.

	- Code permettant au servo-moteur de réaliser des mouvements au stylo pour pouvoir dessiner dans un premier temps une forme géométrique bien précise.
Séance 6	- Améliorer le code pour qu'on fasse bien plus que des figures géométriques, on peut essayer de faire une phrase.
Séance 7	Nous avons finalement changé notre objectif. Désormais, nous récupérerons un objet aimanté (une pièce par exemple) à l'aide d'un électro-aimant et nous le déplacerons vers un autre point afin de le déposer. -Rentrer les bonnes coordonnées de l'objet à récupérer
Séance 8	-Réalisation du programme pour l'électroaimant -Réalisation du programme pour le servomoteur -Présentation orale du programme

Difficultés rencontrées:

La première difficulté a été de trouver un trouver un objectif pour notre projet. Nous étions partis tout d'abord pour tracer des figures géométriques ou écrire une phrase mais au fur et à mesure des séances nous nous sommes rendus compte de la difficulté de l'objectif et nous avons changé l'objectif final. Nous avons finalement décidé de nous focaliser sur la manière de rendre notre table traçante capable de pouvoir aller d'un point A à un point B définis préalablement.

Nous avons également rencontré des difficultés pour contrôler les 2 moteurs en même temps et ainsi réaliser un mouvement diagonal. L'un des deux moteurs se bloquait sans cesse.

Nous avions aussi du mal à planifier nos séances car nous ne nous rendions pas compte de la difficulté du projet.

Également, au début de chaque séance nous devions re-câbler ce qu'on avait fait la séance d'avant ce qui prenait du temps. Le matériel était parfois endommagé, ce qui nous a fait perdre pas mal de temps au cours d'une séance, le temps qu'on puisse trouver l'origine du problème

Annexe:

Code:

```
#include "mbed.h"
#define N 2
Serial pc(USBTX, USBRX);

Ticker ticker;

PwmOut servo_mot(D10);
DigitalOut aimant(D4);
PwmOut horlogey(D7);
DigitalOut sensy(D6);
PwmOut horlogex(D9);
DigitalOut sensx(D8);

void interruption(void);
void CalculDeplacement(int* coordonnees, int* deplacement, int dim);
int absolue(int x);

int compteur = 10;
int coordonneesX[N] = {2000, 0};
int coordonneesY[N] = {2500, 0};
int vecteursX[N];
int vecteursY[N];

int main() {
   horlogex.period_us(10000);
```

```
horlogey.period_us(10000);

horlogex.write(0);
horlogey.write(0);

servo_mot.period_ms(20);        // Initialisation période
servo_mot.pulsewidth_us(1500);   // Initialisation en position 0

ticker.attach(&interruption, 0.01);
CalculDeplacement(coordonneesX, vecteursX, N);
CalculDeplacement(coordonneesY, vecteursY, N);
int i = -1;
sensx = 1;
sensy = 0;

while(1){
   if(compteur == 0){
      horlogex.write(0);
      horlogey.write(0);
      i = i+1;
      if(i==1) {
         servo_mot.pulsewidth_us(2500);   // Angle négatif
         wait(1);
         aimant = 1;
         wait(2);
         servo_mot.pulsewidth_us(1500);
         }
      if(i==2) {
         servo_mot.pulsewidth_us(2500);
```

```
    wait(1);
    aimant = 0;
}
wait(1);
if (i < N){
    if (((double)vecteursX[i]/vecteursX[i+1])<0) {
        sensx = 1 - sensx;
    }
    if (((double)vecteursY[i]/vecteursY[i+1])<0) {
        sensy= 1 - sensy;
    }

    if (absolue(vecteursX[i]) < absolue(vecteursY[i])){
        compteur = absolue(vecteursY[i]);
        horlogey.period_us(10000);
        horlogey.write(0.5);
        int k = 10000*((double)vecteursY[i]/vecteursX[i]);
        k = absolue(k);
        if (k !=0){
            horlogex.period_us(k);
            horlogex.write(0.5);
        }
    }
    else {
        compteur = absolue(vecteursX[i]);
        horlogex.period_us(10000);
        horlogex.write(0.5);
        int k = 10000*((double)vecteursX[i]/vecteursY[i]);
        k = absolue(k);
```

```
                if(k != 0){
                    horlogey.period_us(k);
                    horlogey.write(0.5);
                }
            }
        }
        else{
            horlogex.write(0);
            horlogey.write(0);
        }
    }
  }

}

void interruption () {
  if(compteur > 0) compteur = compteur - 1 ;

}

void CalculDeplacement(int* coordonnees, int* deplacement, int dim) {
int i;

*deplacement = *coordonnees;
for (i=1; i < dim; i++){
  *(deplacement+i) = coordonnees[i] - coordonnees[i-1];
  }
}
```

```
int absolue(int x){
    if(x > 0){
        return x;
        }
    else {
        return -x;
        }
}
```