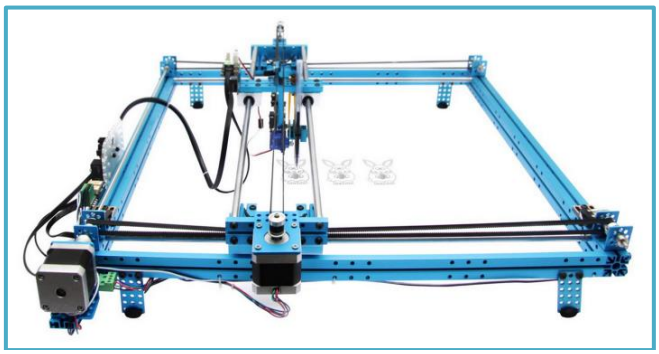
Projet Table Traçante

Rapport technique



Introduction :

Le projet de notre groupe était de concevoir une table traçante, capable de dessiner une forme aléatoire créée par l'utilisateur sur une application type 'InkSpace'. Les traits pouvaient être rectilignes ou courbes, et l'image finale pouvait être composée de plusieurs traits différents. Le système allait alors automatiquement vérifier que l'image était correctement dimensionnée, puis la tracer.

➔ Problématique :

Le but de notre projet était d'utiliser un cadre à deux dimensions, muni de deux moteurs pas à pas et d'un servomoteur et, à l'aide d'une carte microcontrôleur Nucléo, de tracer un dessin issu d'une image vectorielle du logiciel Inkscape. Pour cela, un programme Python, des codes MBed sont nécessaires pour transformer l'image et faire fonctionner les moteurs. Des capteurs de fin de course doivent aussi être ajoutés pour pouvoir réaliser toutes les fonctionnalités. Enfin, il faut alimenter tous les composants du système.

➔ Objectifs du système :

Notre système « table traçante » devait tracer une image donnée par l'utilisateur, en déplaçant le stylo à la fois verticalement et horizontalement. Pour cela, les points de coordonnées de l'image sont convertis en déplacement, et une fonction déplacement utilise un déplacement élémentaire en x ou en y pour faire bouger le stylo sur la feuille.

Les différents objectifs étaient :

- Traiter une **image vectorielle en entrée** (avec toutes les complexités, lignes droites puis courbes, plusieurs traits différents).
- Traiter un **fichier texte de déplacements** et le transmettre à la carte Nucléo.
- Ecrire des **fonctions de déplacements** pour les moteurs, les faire fonctionner en parallèle.
- Dessiner des **formes simples** (carré, triangle) et complexes (cercles, ovales).
- Ecrire **l'initialisation** de la table, qui permet d'assurer une sécurité pour le matériel et l'utilisateur (elle ne vient pas butter sur le cadre) et de faire une vérification automatique de la taille du dessin.
- **Lever du stylo**, pour deux tracés différents ou pour changer de couleur.

Schéma de principe des différents étages de notre système :

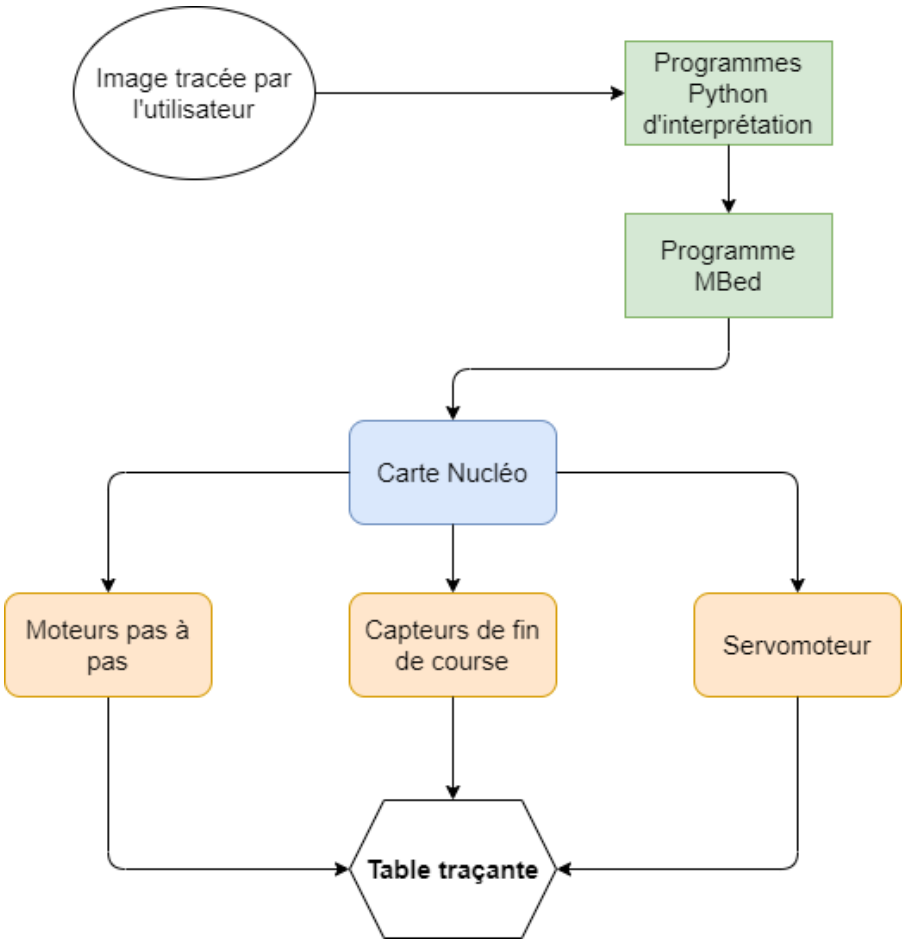


Figure 1 : schéma de principe du système de la table traçante.

Description détaillée des fonctionnalités :

Notre système est composé de deux moteurs pas-à-pas, fonctionnant de manière alternée, qui permettent le déplacement d'un module central selon les axes x et y dans le plan. Le module central est composé d'un système de maintien d'un stylo, et d'un servomoteur lié à une roue crantée, qui permet le déplacement vertical du stylo, pour l'abaisser ou le monter à des hauteurs déterminées.

Sur les bords de la table, où sont fixés les moteurs pas-à-pas, se trouvent également deux capteurs de fin de course, un selon chaque axe x et y. Ces capteurs nous sont utiles pour détecter un impact imminent du module central avec un côté de la table.

L'ensemble de ces composants actifs sont alimentés en courant continu, et leurs parties de commande sont liées électriquement à une carte Nucléo qui fait le lien avec un programme écrit en C, ainsi qu'un autre programme écrit en Python. Ce dernier récupère en entrée une image vectorielle dessinée par un utilisateur, puis la convertit en un fichier texte contenant les informations de déplacements successifs selon les axes x et y. Premièrement, il réalise une lecture des coordonnées indiquées sur le fichier SVG selon sa structure (trait horizontal, vertical, déplacement, courbe …) avant d'enregistrer ces coordonnées dans un tableau 1D. Puis dans un second temps, une deuxième fonction permet de calculer les déplacements entre chaque coordonnée et les inscrire dans un fichier txt.

Schéma électrique de notre système :

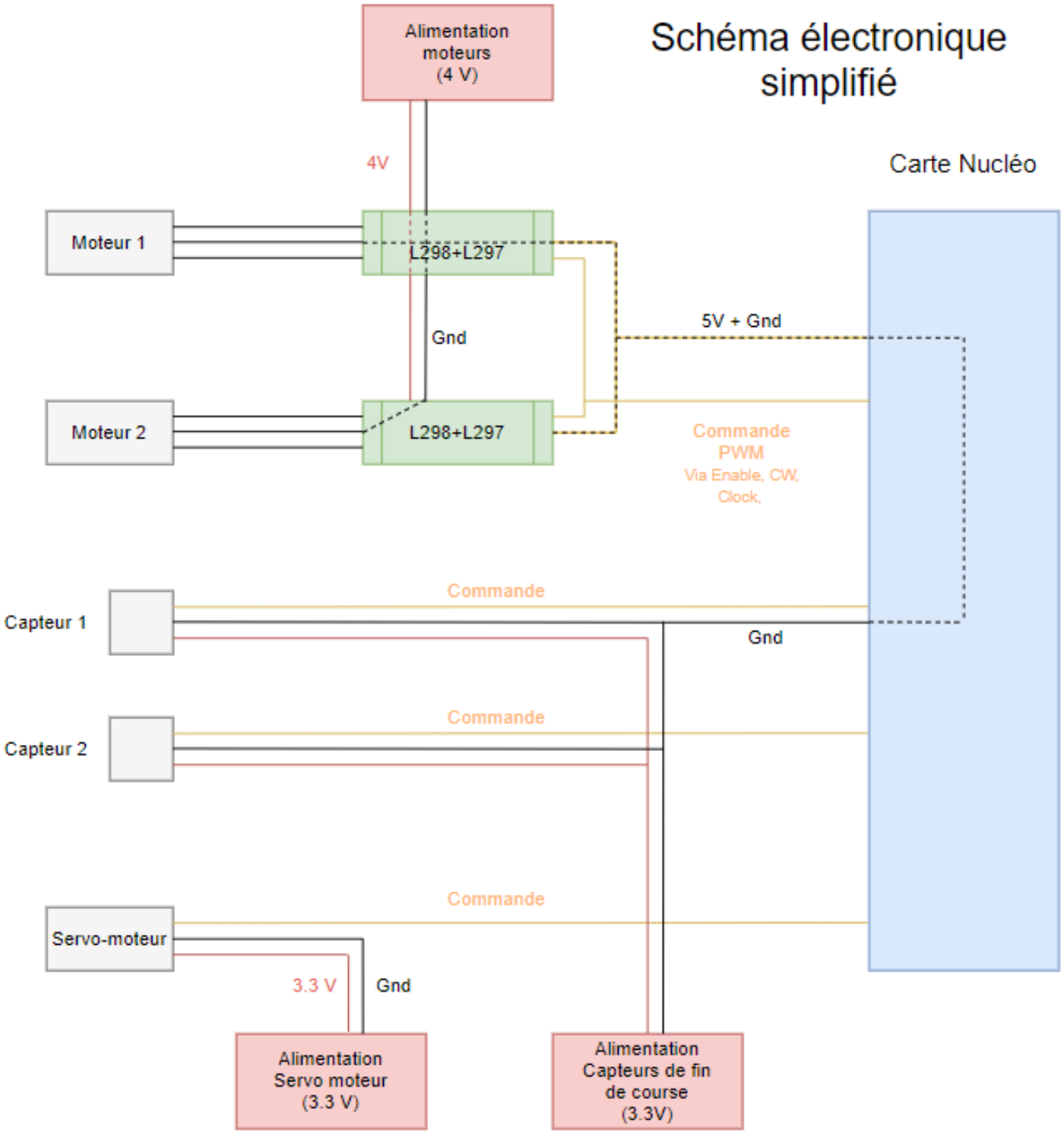


Figure 2 : schéma électrique simplifié du système de la table traçante.

Nos circuits électroniques dépendaient fortement de l'utilisation des cartes L297 et L298. Ces deux cartes, qui pouvaient s'emboîter l'une dans l'autre pour fonctionner ensemble, permettaient une gestion plus efficace et plus visuelle des moteurs. En effet, étaient directement implantés des entrées/sorties contrôlant le sens de rotation du moteur, le contrôle de la MLI envoyée à chaque moteur, l'activation ou non des moteurs avec les entrées 'Enable'… Ces cartes recevaient également la tension d'entrée des moteurs : elles centralisaient tous les fils nécessaires au bon fonctionnement des moteurs.

Algorithmes :

Algorithme de la fonction de déplacement élémentaire

Principe :

- Déclaration d'une sortie MLI, et d'une sortie « sens de rotation » (CW) moteur.
- Ecriture du sens de rotation du moteur désiré, des paramètres de la MLI désirée, du temps de rotation élémentaire.

Entrée :

Aucune (c'est une procédure).

Procédure :

Ecriture du sens de rotation horaire sur la sortie dédiée de la carte Nucléo, reliée à une entrée contrôlant le sens de rotation sur la carte L298.

Ecriture du rapport cyclique et de la période de la MLI sur la sortie Nucléo correspondante.

Attente de 0,02 secondes : c'est le temps élémentaire du déplacement élémentaire, fixé arbitrairement.

Arrêt du moteur en redéfinissant le rapport cyclique à 0%.

Sortie :

Aucune (c'est une procédure).

Algorithme de la fonction de déplacement

Principe :

- Calcul de la pente du segment à tracer.
- Tracé du segment en différenciant les cas selon l'appartenance du segment à l'un des quatre quadrants de direction.

Entrée :

Scalaires x et y, représentant le déplacement sur l'axe x et sur l'axe y.

Procédure :

Calcul de la valeur de la pente du segment en calculant (double)(y*1.0/x*1.0).

Conditions 'if' pour déterminer la procédure de tracé du segment. Les différents cas possibles sont 'en haut à gauche' (x<0 et y>0), 'en haut à droite' (si x>0 et y>0), 'en bas à gauche' (si x<0 et y<0), et 'en bas à droite' (si x>0 et y<0).

Déplacement du moteur sur l'axe x de 1 déplacement élémentaire dans le sens considéré, tandis que le moteur sur l'axe y se déplace de a fois le déplacement élémentaire dans le sens considéré, et cela autant de fois que nécessaire pour atteindre la bonne longueur de segment. Les signes de x, y et a sont amenés à changer selon le quadrant, afin de rentrer dans des boucles inconditionnelles 'for' avec des paramètres positifs. Le cas d'un déplacement uniquement selon x ou uniquement selon y est également codé.

Sortie :

Aucune.

Algorithme de la fonction d'initialisation

Principe :

- Déplacements des moteurs 1 puis 2 automatiques jusqu'à détection d'un front montant en tension sur les capteurs de fin de course 1, puis 2.
- Déplacement automatisé des deux moteurs jusqu'au centre géométrique de la table.

Entrée :

Aucune.

Procédure :

Tant que le capteur de fin de course n°1 ne détecte pas de montée en tension à ses bornes, déplacements élémentaires du moteur 1 vers le capteur de fin de course. A chaque tour de boucle, le capteur mesure la tension à ses bornes et compare cette valeur à une valeur seuil de 2,5 V. On avance dans le programme une fois la condition réalisée.

Tant que le capteur de fin de course n°2 ne détecte pas de montée en tension à ses bornes, déplacements élémentaires du moteur 2 vers le capteur de fin de course. A chaque tour de boucle, le capteur mesure la tension à ses bornes et compare cette valeur à une valeur seuil de 2,5 V. On avance dans le programme une fois la condition réalisée.

Attente de deux secondes.

Déplacements des deux moteurs vers le centre de la table, avec des déplacements élémentaires alternés, calibrés manuellement.

Sortie :

Scalaire 0 confirmant le bon déroulement de l'algorithme.

Caractérisation des fonctionnalités réelles du système :

Nous avions créé un cahier des charges en début de projet reprenant différentes caractéristiques que notre système devait avoir en fin de projet. Parmi elles, le système devait avoir un temps d'abaissement du stylo inférieur à la seconde. Le servomoteur étant un moteur très réactif, les temps d'abaissement et de montée du stylo étaient très courts, de l'ordre du dixième de seconde.

Nous nous étions aussi fixés d'obtenir des écarts en taille de trait, pour les traits droits, inférieurs à 1% de la taille totale du trait. Sur la figure carrée, qui était en théorie quatre déplacements successifs de 100 déplacements élémentaires, on mesure pour les segments verticaux une longueur de 5,6 cm, et pour les segments horizontaux une longueur de 5,2 cm. Notre écart relatif vaut donc 7%. Nous ne sommes pas dans notre objectif, mais cet écart a plusieurs explications, dont notamment le fait que le support du stylo est assez peu solidaire du module de déplacement ; nos traits sont en partie "tremblotants" pour cette raison, et cela peut engendrer des écarts à la théorie de l'ordre du millimètre (nous ne pouvons pas réellement passer en-dessous de cette limite).

Un objectif d'écart en courbure accepté pour les traits courbes maximal de 5 % du rayon théorique avait été formulé. Cependant, jusqu'à la dernière séance, nous ne sommes pas parvenus à tracer un cercle complet ; nous ne pouvons pas valider cet objectif, faute de relevés convaincants.

Le temps maximal de tracé pour un trait de longueur 5 cm devait être inférieur à 5 secondes. Pour les segments horizontaux du carré, de longueur 5,2 cm, on mesure un temps de tracé de 1,3 secondes en moyenne. Cet objectif est donc validé.

Le temps de commutation entre les deux moteurs avait été fixé inférieur à 0,5 secondes. Par construction de notre programme, les deux moteurs fonctionnent alternativement en permanence ; le temps de commutation vaut le temps d'exécution d'une ligne de code, soit un temps bien inférieur à 0,5 secondes.

Enfin, la réactivité des capteurs de fin de course lorsqu'une pression leur est appliquée devait être inférieure à 0,1 secondes ; là encore, le programme d'initialisation est conçu pour entrer dans une boucle qui ne s'achève que lorsque le capteur détecte un front montant de tension. Le temps de réactivité vaut le temps d'exécution d'une boucle en C qui lit une variable extérieure et la compare à un seuil, soit un temps inférieur à 0,1 secondes.

Planning des séances :

26/01	Mécanique	Découverte du transistor L293.	Octave, Justin, Cassandre, Kyliann
		Câblage du pont en H avec le L293.	Justin, Cassandre
		Vérification du fonctionnement d'un moteur.	Octave, Justin, Cassandre, Kyliann
	Informatique	Début du code Mbed : rotation simple d'un moteur.	Octave, Kyliann
	Organisation	Création du Cahier des Charges et du Planning.	Octave, Kyliann
02/02	Mécanique	Câblage des deux moteurs principaux.	Cassandre, Justin
	Informatique	Code MBed	Octave, Kyliann
	Organisation	Livrables intermédiaires terminés et rendus.	Octave, Justin, Cassandre, Kyliann
09/02/2021	Mécanique	Câblage du demi-pas des bobines.	
		Caractérisation et câblage du troisième moteur.	Justin, Cassandre
		Calcul du déplacement élémentaire des moteurs.	Octave, Kyliann
	Informatique	Code Mbed : codage du déplacement dans toutes les directions des moteurs.	Octave, Kyliann
		Code Mbed : codage du demi-pas des moteurs.	Octave, Kyliann
23/02	Mécanique	Câblage complet des deux moteurs réels de translation selon les axes x et y, plus le troisième.	Justin, Cassandre
		Vérification des fonctionnements globaux, avec le code MBed complet.	Octave, Justin, Cassandre, Kyliann
	Informatique	Fin du code Mbed : vérification du fonctionnement, tracé de formes complexes (carré, triangle).	Octave, Kyliann
31/03/2021	Mécanique	Câblage des capteurs de fin de course.	Octave, Justin, Cassandre, Kyliann
	Informatique	Codage Mbed du traitement de l'image 1 : déplacements successifs avec un tableau de vecteurs en entrée.	Justin, Cassandre
		Codage du tracé du cercle, implémentation des bordures physiques de la table dans le code.	Octave, Kyliann
		Codage du lever et baisse de crayon, avec le troisième moteur.	Justin
07/04	Mécanique	Recâblage général	Octave, Justin, Cassandre, Kyliann
	Informatique	Codage Mbed du traitement de l'image 2 : transformation d'une image vectorielle en tableau compréhensible pour le code principal.	
	Organisation	Audit séance 6	Octave, Justin, Cassandre, Kyliann
12/05	Mécanique	Recâblage général	Octave, Justin, Cassandre, Kyliann
	Informatique	Codage Mbed : tentative de mise en place du lien entre la transformation de l'image vectorielle et le code principal.	Octave, Justin, Cassandre, Kyliann
		Codage Mbed : tentative de réussir à tracer le cercle en entier.	Octave, Kyliann
	Organisation	Début de création de l'affiche du projet	Octave, Kyliann
25/05	Mécanique	Recâblage général	Octave, Justin, Cassandre, Kyliann
	Informatique	Test du tracé de cercle complet et du code amélioré.	Kyliann
	Organisation	Séance de restitution et de présentation.	Octave, Justin, Cassandre, Kyliann

Difficultés rencontrées :

Le problème le plus contraignant que nous avons rencontré était de devoir rebrancher les fils à chaque séance, nous faisant perdre beaucoup de temps : non seulement rebrancher les nombreux fils des alimentations, des capteurs, les cartes L297 que nous ne pouvions pas garder sur notre montage, mais surtout vérifier à chaque séance à quoi correspondait chaque fil étant donné qu'un autre groupe les changeait à chaque fois, y compris leur couleur.

Nous avons rencontré plusieurs problèmes techniques mineurs en essayant de maîtriser la Nucléo et les moteurs :

- Des vibrations du moteur, qui n'avançait pas ou mal : il fallait augmenter le courant à 280mA, mais surtout ne pas brancher des fils du moteurs sur les broches D0 et D1 de la Nucléo : celles-ci ne sont pas soudées au reste de la carte, nous étions donc reliés au vide !
- Ne pas oublier de relier toutes les masses entre elles, sinon la carte Nucléo ne fonctionne plus correctement, elle se déconnecte et se reconnecte, sans effectuer le programme.

Nous n'avons reçu les cartes L297 fabriquées au LEnsE qu'une fois le projet bien entamé, ce qui nous a conduit à revoir tous les branchements et le code pour les adapter à cette carte. Cette opération nous a pris beaucoup de temps.

Nous n'avons pas réussi à faire fonctionner les deux moteurs en parallèle malgré nos essais.

La principale difficulté que nous avons finalement rencontrée est la complexité des images vectorielles (.svg). Le programme transformant les images en déplacements du stylo fonctionne seulement pour des images sans courbes (uniquement des traits droits). Il nous aurait fallu beaucoup plus de temps pour traiter des images plus complexes.

Afin de réaliser la lecture du fichier SVG, nous avons travaillé au départ à la réalisation d'un programme en C. Cependant, nous avons eu beaucoup de difficultés dans la réalisation de ce code. Après discussion sur ce sujet avec Etienne Minnaert, celui-ci nous a proposé de nous fournir une ébauche de code Python qui réalise cette fonction. Nous avons retravaillé cette fonction lors d'une séance pour réaliser ce que nous souhaitions. Etant donné que nous avons juste travaillé à partir de ce code sans réaliser de changements majeurs dans l'algorithme, nous préférons ne pas mettre en avant ce code dans ce compte-rendu.

Le dernier problème technique que nous avons rencontré est la lecture du fichier texte contenant ces déplacements par MBed : malgré des recherches sur Internet, nous n'avons pas réussi. La solution se situe certainement avec une autre version de MBed, mais nous n'avons pas pu chercher davantage.

Analyse du travail d'équipe :

Le travail d'équipe s'est parfaitement déroulé dès les premières séances. Naturellement, Justin et Cassandre se sont occupés de la partie électrique du projet, en procédant au montage dans un premier temps des moteurs, puis du montage entier avec servomoteur et capteurs. Quant à eux, Octave et Kyliann se sont intéressés à la partie codage du projet, en écrivant la fonction de déplacement des moteurs et en codant l'initialisation, et dans un second temps le dessin des formes. Justin a codé le lever du crayon, Cassandre s'est intéressée aux structures des images vectorielles, et ils ont écrit le code Python pour les transformer en déplacement ; Kyliann et Octave sont donc passés à la partie montage électrique pour tester leurs codes. Chaque membre du groupe maîtrise donc le montage électrique, et a codé une partie du projet. Cette organisation nous permet donc d'avoir une vue d'ensemble du projet et d'acquérir des compétences variées.

Conclusions et perspectives générales :

Ce projet nous a permis de nous initier à la gestion de projet technique. Nous avons dû prendre connaissance du matériel sur lequel nous avons travaillé puis nous avons déterminé un cahier des charges avant d'essayer de trouver des solutions techniques aux problèmes que nous avons pu rencontrer.
Durant les séances, nous avons pu comprendre le fonctionnement détaillé de deux types de moteurs dont le pilotage est désormais fonctionnel et efficace. L'initialisation de la position du stylo via les capteurs de fin de course, ainsi que le contrôle de la hauteur du stylo, sont eux aussi fonctionnels. Le programme permettant la transcription d'une image de vecteur vers un tableau de déplacement est achevé.
Nous avons cependant manqué de temps pour réunir ces éléments et réaliser une démonstration complète fonctionnelle.
A remarquer que nos moteurs ne fonctionnent pas en parallèle. Il aurait été intéressant de faire fonctionner les deux moteurs simultanément, grâce par exemple à un pont en H, pour obtenir des courbes plus fluides lors des tracés.

Conclusions et perspectives personnelles :

On était parti sur le projet de table traçante sans vraiment savoir ce qu'on allait y faire, sans avoir aucune idée de ce que le système était capable de faire… Et on n'a pas été déçu ! La première séance de prise en main était intéressante, on arrivait devant un système dont on ne connaissait aucune caractéristique, et c'était à nous de faire fonctionner les moteurs, les programmes, sans énoncé pour nous guider ! J'ai personnellement beaucoup aimé créer des petits programmes pour faire tourner les moteurs pas à pas, en commandant la tension envoyée dans chacune des bobines successivement : là, on était vraiment en train de partir de zéro, et de comprendre réellement comment fonctionne ce genre de moteurs ! Petit à petit, le travail à chaque séance s'est complexifié, codes de déplacement plus fournis, ajout des capteurs de fin de course, fonctions annexes pour l'initialisation de la table… Malgré une partie « mécanique » (électrique) elle aussi complexe, avec beaucoup de câbles, c'est bien la première fois qu'on avait autant de contrôle sur ce que l'on faisait, que nos codes écrits à partir de rien pouvaient avoir un impact sur un système réel câblé de nos mains… Je retire beaucoup de positif de ce projet ! Kyliann

La seule partie négative du projet, c'est le montage très redondant et chronophage. C'est dommage, puisque nous aurions pu utiliser ce temps pour avancer plus loin ! Pour moi, il y a deux améliorations possibles :

- D'une part, avoir accès aux cartes L297 plus tôt dans le déroulé des séances. C'était très instructeur de devoir faire fonctionner les moteurs pas à pas pour comprendre l'utilité des bobines, mais le montage intermédiaire avec les cartes L298 mais sans les L297 n'a pas apporté beaucoup.

- D'autre part, la communication entre équipes de différents groupes aurait dû être meilleure : beaucoup de changements de fils, de couleur, qui ont entraîné une perte de temps à tester la nature des signaux sortants ; alors qu'il aurait suffi de se mettre d'accord. Cassandre

Le projet était dans l'ensemble très plaisant, l'ambiance d'un travail en groupe sur un temps prolongé m'a fait beaucoup de bien. L'objectif du projet n'était pas en soi le plus intéressant à mes yeux ; une fois entrés dans les problématiques liées à la conception, l'intérêt naissait à mesure que les défis se présentaient. J'ai eu la chance d'avoir des camarades très performants et entraînés pour le codage notamment, qui m'ont aidé à comprendre et à trouver des idées de programmation. Je tiens ici à les remercier pour leur patience, qualité déterminante dans un travail en groupe.

L'idée générale du projet en une petite dizaine de séances m'a beaucoup plu. Trouver des idées, s'apercevoir que d'autres méthodes auraient pu aussi fonctionner a donné à cette expérience une dimension très réaliste de la résolution d'une tâche complexe. Nous étions encadrés, mais uniquement pour sortir des difficultés dans lesquelles nous nous étions engagés, et j'ai beaucoup apprécié cette autonomie. Quand les professeurs nous ont demandé de faire un retour oral à la fin de la dernière séance, la seule chose que j'aurais pu dire était que ce projet était quasiment parfait dans ce format-là. Octave

Ce projet fut idéal dans son format car nous avons été mis en grande autonomie sur la gestion du projet. Cela nous a permis de comprendre le fonctionnement du matériel pas à pas. Nous avons pu de cette manière comprendre le fonctionnement de deux types de moteurs différents ainsi que réaliser un montage électronique bien plus complexe que ce que nous avons pu voir lors du premier semestre. Je regrette le manque de communication entre les groupes partageant le même matériel. C'est d'ailleurs finalement de là que provient la majorité des difficultés que nous avons pu rencontrer (montages récurrents, disfonctionnements ...).

Il serait intéressant de voir comment contourner la limitation de la carte Nucléo qui est incapable de lire un fichier annexe au code qui contiendrait de l'information. Dans notre cas, nous aurions été intéressés que la carte puisse lire les coordonnées de l'image, pour en déduire les déplacements à réaliser lors de l'exécution sans devoir entrer directement à la main les déplacements dans le code Mbed. Justin

Annexe 1 : Codes MBed

```
#include "mbed.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#define TAILLE_MAX 1000


////////// DECLARATION DES ENTREES ET SORTIES NECESSAIRES ////////////////////

DigitalOut  led(D13);
//Entrées analogiques des capteurs de fin de course
AnalogIn     Capt1_NO(A2);
AnalogIn     Capt2_NO(A3);
//Entrées du moteur 1
PwmOut       M1Clock(PB_11);                                    //Clock
DigitalOut   M1CW(PB_1);                                        //CW
//Entrées du moteur 2
PwmOut       M2Clock(PA_11);                                    //Clock
DigitalOut   M2CW(PB_2);                                        //CW
//Enables pour la carte L297
DigitalOut   En1(D6);                                           //Tous les fils à 3.3 V
DigitalOut   En2(D7);                                           //Tous les fils à 3.3 V
//Contrôle du servomoteur
PwmOut       servo_mot(D14);

DigitalOut   myled(LED1);
Serial       pc(USBTX, USBRX);
double       compteurX=0.0;             //Pour connaitre les bordures de la table selon l'axe x
double       compteurY=0.0;             //Pour connaitre les bordures de la table selon l'axe y


////////// DECLARATIONS DE FONCTIONS ////////////////////////////////////////

int initialisation(void);

void deplacement(int x, int y);

void deplacement_elementaire_moteur1(void);
void deplacement_elementaire_moteur2(void);
void deplacement_elementaire_inverse_moteur1(void);
void deplacement_elementaire_inverse_moteur2(void);

void triangle(void);
void carre(void);
void cercle(void);


////////// FONCTION PRINCIPALE ////////////////////////////////////////////////

int main() {
    led = 0;
    pc.baud(115200);
```

```
    En1.write(1);
    En2.write(1);            //Initialisation des enable à 1 pour pouvoir utiliser les moteurs

    servo_mot.period_ms(20);   // Initialisation de la période de fonctionnement du servomoteur
    servo_mot.pulsewidth_us(2000) ;    //  Initialisation  en  position  0  (position  haute  du
servomoteur)
    wait_us(2000000);

    fin=initialisation() ;      //Recherche du coin de la table puis positionnement au centre
de la table

    while(1) {
        servo_mot.pulsewidth_us(1800);              //Position 1 (position basse du servomoteur)
        wait_us(4000000);                           //On attend 4 secondes dans la position 1
        triangle();
        wait_us(10000);
        carre();
        wait_us(10000);
        myled = 1;
        wait_us(500000);
        myled = 0;
        wait_us(500000);                    //Pour savoir où on en est, et ne pas sortir du code
        cercle();
        servo_mot.pulsewidth_us(1500);                //Position 0 (position haute)
        wait_us(2000000);                             //On attend 2 secondes dans la position 0
        }
    return 0;
    }


////////// EXPRESSIONS DES FONCTIONS ////////////////////////////////////////

void deplacement_elementaire_moteur1(void) {
    M1CW.write(1);
    double rc = 0.5;                                    //On initie un rapport cyclique à 50%
    M1Clock.period_ms(3);                               //Sur une période de 3 ms
    M1Clock.write(rc);
    wait_us(20000);                           //On fait tourner le moteur pendant 0,02 secondes
    rc = 0;
    M1Clock.write(rc);                                          //On l'arrête
    compteurX+=1.0;                                             //Incrémentation de compteurX
    }

void deplacement_elementaire_moteur2(void) {
    M2CW.write(1);
    double rc = 0.5;                                    //On initie un rapport cyclique à 50%
    M2Clock.period_ms(3);                               //Sur une période de 3 ms
    M2Clock.write(rc);
    wait_us(20000);                           //On fait tourner le moteur pendant 0,02 secondes
    rc = 0;
    M2Clock.write(rc);                                          //On l'arrête
    compteurY+=1.0;                                             //Incrémentation de compteurY
    }
```

```
void deplacement_elementaire_inverse_moteur1(void) {
    M1CW.write(0);
    double rc = 0.5;                                        //On initie un rapport cyclique à 50%
    M1Clock.period_ms(3);                                   //Sur une période de 3 ms
    M1Clock.write(rc);
    wait_us(20000);                             //On fait tourner le moteur pendant 0,02 secondes
    rc = 0;
    M1Clock.write(rc);                                          //On l'arrête
    compteurX-=1.0;                                         //Décrémentation de compteurX
    }

void deplacement_elementaire_inverse_moteur2(void) {
    M2CW.write(0);
    double rc = 0.5;                                        //On initie un rapport cyclique à 50%
    M2Clock.period_ms(3);                                   //Sur une période de 3 ms
    M2Clock.write(rc);
    wait_us(20000);                             //On fait tourner le moteur pendant 0,02 secondes
    rc = 0;
    M2Clock.write(rc);                                            //On l'arrête
    compteurY-=1.0;                                         //Décrémentation de compteurY
    }

void deplacement(int x, int y) {
    int i=0;
    int j=0;

    if((x!=0) && (y!=0)){
        double a=(double)((y*1.0)/(x*1.0));                 //Calcul de la pente
        if((x>0)&&(y>0)){                                   //Dans le quadrant en haut à droite
            for(i=0;i<x;i++){
                deplacement_elementaire_moteur1();                  //Déplacement de 1 sur x...
                compteurX+=1.0;
                for(j=0;j<int(a);j++){
                    deplacement_elementaire_moteur2();              //...et de a sur y
                    compteurY+=a;
                    }
                }
            }
        if((x<0)&&(y>0)){                                   //Dans le quadrant en haut à gauche
            x=-x;
            a=-a;     //Pour faire des tours de boucle positifs, on change les signes de x et a
            for(i=0;i<x;i++){
                deplacement_elementaire_inverse_moteur1();   //Déplacement inverse de 1 sur x...
                compteurX-=1.0;
                for(j=0;j<int(a);j++){
                    deplacement_elementaire_moteur2();       //...et de a sur y
                    compteurY+=a;
                    }
                }
            }
        if((x>0)&&(y<0)){                                   //Dans le quadrant en bas à droite
            y=-y;
            a=-a;     //Pour faire des tours de boucle positifs, on change les signes de y et a
            for(i=0;i<x;i++){
                deplacement_elementaire_moteur1();                  //Déplacement de 1 sur x...
```

```
                compteurX+=1.0;
                for(j=0;j<int(a);j++){
                    deplacement_elementaire_inverse_moteur2();      //...et déplacement inverse
de a sur y
                    compteurY-=a;
                    }
                }
            }
         if((x<0)&&(y<0)){                                  //Dans le quadrant en bas à gauche
            x=-x;
            y=-y;     //Pour faire des tours de boucle positifs, on change les signes de x et y
            for(i=0;i<x;i++){
                deplacement_elementaire_inverse_moteur1();   //Déplacement inverse de 1 sur x...
                compteurX-=1.0;
                for(j=0;j<int(1/a);j++){
                    deplacement_elementaire_inverse_moteur2();     //...et déplacement inverse
de a sur y
                    compteurY-=a;
                    }
                }
            }
        }
    if(x==0) {                                                  //Pour une ligne verticale
        if(y>0){
            for(i=0;i<y;i++){
                deplacement_elementaire_moteur2();                  //Soit on monte...
                compteurY+=1.0;
                }
            }
        if(y<0){
            y=-y;
            for(i=0;i<y;i++){
                deplacement_elementaire_inverse_moteur2();            //... soit on descend
                compteurY-=1.0;
                }
            }
        }
    if(y==0){                                               //Pour une ligne horizontale
        if(x>0){
            for(i=0;i<x;i++){
                deplacement_elementaire_moteur1();              //Soit on va vers la droite...
                compteurX+=1.0;
                }
            }
        if(x<0){
            x=-x;
            for(i=0;i<x;i++){
                deplacement_elementaire_inverse_moteur1();      //... soit on va vers la gauche
                compteurX-=1.0;
                }
            }
        }
    }
```

```
void triangle(void){
    deplacement(0,100);
    deplacement(-50,-50);
    deplacement(50,-50);
    }

void carre(void) {
    deplacement(-100,0);
    deplacement(0,-100);
    deplacement(100,0);
    deplacement(0,100);
    }

void cercle(void) {
    double t[124];
    int i;
    for(i=0;i<124;i++) {
        t[i]=0.05*i;
    }
    double X[124];
    double Y[124];
    for(i=0;i<124;i++) {
        X[i]=100*cos(t[i]);
        Y[i]=100*sin(t[i]);
        }
    for(i=0;i<123;i++){
        deplacement(X[i+1]-X[i],Y[i+1]-Y[i]);
        }
    }

int initialisation(void){
    int k=0;
    int l=0;
    int m=0;
    int n=0;
    double tension1;
    double tension2;                                        //Initialisations
    while(1) {
        deplacement_elementaire_moteur1();                  //Tant qu'on n'atteint pas le capteur
de fin de course n°1, on avance vers lui
        k=Capt1_NO.read_u16();
        tension1 = k / 65535.0 * 3.3;               //On lit et convertit la tension du capteur
        if(tension1>2.5) {                  //Lorsqu'on détecte un pic de tension (déclenchement
du capteur), on passe à l'autre axe...
            for(n=0;n<11;n++){
                deplacement_elementaire_inverse_moteur1();    //... après avoir décalé le stylo
un peu pour ne pas laisser le capteur n°1 enclenché
                }
            while(1){
                deplacement_elementaire_inverse_moteur2();
                l=Capt2_NO.read_u16();
                led = 1;
                tension2 = l / 65535.0 * 3.3;
                if(tension2>2.5){
                        for(n=0;n<11;n++){
```

```
                        deplacement_elementaire_moteur2();  //On applique la même procédure
pour le capteur n°2
                    }
                    wait_us(2000000);                       //Attente de deux secondes
                    for(m=0;m<280;m++){
                        deplacement_elementaire_inverse_moteur1();
                        deplacement_elementaire_moteur2();  //Déplacement du stylo vers le
centre de la table.
                        }
                    wait_us(3000000);
                    return 0;                   //On sort de la procédure d'initialisation.
                }
            }
        }
    }
}
```

Annexe 2 : Code Python d'interprétation

```
# -*- coding: utf-8 -*-
"""

Script qui permet d'obtenir les listes des points entiers à partir des données de Inskape
        ne fonctionne que pour des segments de droite -> retourne des listes en cas d'echecs

Attention il ne fonctionne pas avec toutes les courbes de Bezier

ressource image SVG:
    https://www.w3.org/TR/SVG/paths.html

ressource courbes de Bezier:
    http://www-ihm.lri.fr/~mbl/ENS/IG1/COURS/04bis-Bezier.pdf

"""

def supp_str(l:list, chaine:str='')->list:
    """
    Supprime dans une liste quelconque les occurences d'un str et renvoie une nouvelle liste
    par defaut la fonction supprime le str vide ''
    """
    liste = []
    for elem in l:
        if elem !=chaine: liste.append(elem)
    return liste


def separation_type(chaine:str)-> list:
    """
    A partir d'une chaine de caractère "d=" obtenu dans un fichier .svg retourne une
    liste avec les informations séparées
    """
    liste = []

    chaine += ' ' #condition de fin supprimée à la fin

    info = ''
    for k in range(len(chaine)):
        if chaine[k] in ' ,':
            liste.append(info)
            info = ''
            continue
        else:

            info += chaine[k]
            if chaine[k] in "LlMmZzVvHhCcSsQqTtAa": #lettres courbes de Bezier
                if info[0] in "-0123456789":
                    liste.append(info[:-1])

                liste.append(info[-1])
                info = ''
```

```
    liste = supp_str(liste)
    return liste

def conversion(l:list, fact:float =1)-> list:
    """
    Convertie les nombres en str en int et réalise une dilatation de facteur fact
    """

    liste = []

    for elem in l:
        if elem[0] in "-0123456789": #on verifie que c'est un chiffre
            valeur = round(fact*float(elem)) #typage et dilatation
            liste.append(valeur)
        else :
            liste.append(elem)

    return liste



def coordo_absolu(liste:list)-> (list,list):

    #on ne sait pas gerer les courbes et les arcs
    if len(set(liste)&set('CcSsQqTtAa')) > 0: #intersection d'ensemble
        print('\n\nCourbes non gérées par le programme CcSsQqTtAa',  \
              '*coordo_absolu* \n\n')
        return None

    else:
            if len(set(liste)&set('zZ')) == 0: liste += ["FIN"] #eviter les dépacements de
boucle
            liste_x,liste_y = [],[]
            posXY = 'x' #permet de différencier un

            compteur, limit, sauter = 0, 0, 0
            liste_x,liste_y = [0],[0]
            liste_xk,liste_yk = [],[]
            for indice, elem in enumerate(liste):

                if sauter == 1:
                    compteur +=1
                    if compteur >= limit : compteur, limit, sauter = 0, 0, 0
                    continue

#       -----------  Initialisation des listes : premiers points  --------------
                if str(elem) in 'mM':
                    dim=len(liste_xk)
                    liste_x
                    liste_x.append(0)
                    liste_x.append(111111)
                    liste_x.append(0)
                    dimg=len(liste_x)
                    liste_x.append(liste_x[dimg-4])
```

```
                    liste_y.append(0)
                    liste_y.append(111111)
                    liste_y.append(0)
                    liste_y.append(liste_y[dimg-4])
                    for i in range (0,dim):
                        liste_x.append(liste_xk[i])
                        liste_y.append(liste_yk[i])
                    liste_xk,liste_yk=[],[]
                    liste_xk.append(liste[indice+1])
                    liste_yk.append(liste[indice+2])
                    limit = 2
                    sauter = 1

#       --------------    Derniers points et condition de fin    ---------------
                elif str(elem) in 'zZ': #si la  courbe est fermée
                    liste_xk.append(liste_xk[0])
                    liste_yk.append(liste_yk[0])
                    return liste_x,liste_y

                elif str(elem) == "FIN":
                    dim=len(liste_xk)
                    for i in range (0,dim):
                        liste_x.append(liste_xk[i])
                        liste_y.append(liste_yk[i])
                    return liste_x,liste_y

#       --------------           Segments de droites         -------------------
                elif str(elem) in 'H':
                    indice = indice + 1
                    elem = liste[indice]
                    while str(elem)[0] in "-0123456789":
                        liste_xk.append(liste[indice])
                        liste_yk.append(liste_yk[-1])

                        indice += 1
                        elem = liste[indice]
                        limit += 1

                    sauter =1


                elif str(elem) in 'V':
                    indice = indice + 1
                    elem = liste[indice]
                    while str(elem)[0] in "-0123456789":
                        liste_xk.append(liste_xk[-1])
                        liste_yk.append(liste[indice])

                        indice += 1
                        elem = liste[indice]
                        limit += 1

                    sauter =1

                elif str(elem) == "v":
```

```
                    indice = indice + 1
                    elem = liste[indice]

                    while str(elem)[0] in "-0123456789":
                        liste_xk.append(liste_x[-1])
                        liste_yk.append(liste_y[-1]+elem)

                        indice += 1
                        elem = liste[indice]
                        limit += 1

                    sauter =1



                elif str(elem) == "h":
                    indice = indice + 1
                    elem = liste[indice]

                    while str(elem)[0] in "-0123456789":
                        liste_xk.append(liste_x[-1]+elem)
                        liste_yk.append(liste_y[-1])

                        indice += 1

                        elem = liste[indice]
                        limit += 1

                    sauter =1

                elif str(elem) == "l":

                    indice = indice + 1
                    elem = liste[indice]
                    while str(elem)[0] in "-0123456789":
                        if posXY == "x":
                            liste_x.append(liste_x[-1]+elem)
                            posXY = "y"

                        elif posXY == "y":
                            liste_y.append(liste_y[-1]+elem)
                            posXY = "x"

                        indice += 1
                        elem = liste[indice]
                        limit += 1
                    sauter =1

                elif str(elem) == "L":
                    indice = indice + 1
                    elem = liste[indice]
                    while str(elem)[0] in "-0123456789":
                        if posXY == "x":
                            liste_xk.append(elem)
                            posXY = "y"
```

```
                        elif posXY == "y":
                            liste_yk.append(elem)
                            posXY = "x"

                        indice += 1
                        elem = liste[indice]
                        limit += 1
                    sauter =1

#       ----------      Segments de droites (par defaut)         ---------------
                else:
                    if liste[0] == "M": ## deplacement absolu  "L"
                        if posXY == "x":
                            liste_x.append(elem)
                            posXY = "y"
                        elif posXY == "y":
                            liste_y.append(elem)
                            posXY = "x"

                    elif liste[0] == "m": # deplacement relatif "l"
                            if posXY == "x":
                                liste_x.append(liste_x[-1]+elem)
                                posXY = "y"

                            elif posXY == "y":
                                liste_y.append(liste_y[-1]+elem)
                                posXY = "x"



def listes_pointsSVG(info:str, fact:float = 1)-> (list,list):
    """
    Renvoie deux listes respectivement des coordonnées X et Y à partir d'un
    str trouver dans une image SVG : d=
    """
    # separation de toutes les informations du str
    info = separation_type(info)

    #print("\n {}".format(info))

    #conversion des points en entier avec un dilation de facteur fact
    l = conversion(info, fact)

    #print("\n {}".format(l))

    #creation de deux listes donnant l'abscisse (resp l'ordonnée) de chaque point
    try:
        listeX, listeY = coordo_absolu(l)
    except TypeError:
        print(TypeError)
        listeX, listeY = [], []

    return listeX, listeY
```

```
if  __name__=='__main__':
    info = "m 57.132569,55.055021 20.77548,167.24261 m 86.737628,215.02622 155.8161,155.8161 l 1.55816,-47.26422   -36.35709,4.67448   -8.82958,11.94591   -12.465286,-58.171347   46.225446,-11.426515 17.13977,7.790805 23.37241,37.915247"
    # info = "m 101.06583,114.87855 94,149 -140.999996,37 -27,-133 9,-142.000007 92.999996,-7 v 0"


    X,Y = listes_pointsSVG(info,2)


    print("-"*50, "\n\n")

    print("\t{}\t\t{}".format('X','Y'))
    for u,k in zip(X,Y):
        print("\t{}\t\t{}".format(u,k))

    print()
    print()
    print("X={} \t {}".format(X, len(X)))
    print("Y={} \t {}".format(Y, len(Y)))
    print()

    print()


#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: Étienne Minnaert - avril 2021
"""


def ecriture_trace(liste_x :list, liste_y :list, fichier)-> None:
    """
    Ecriture dans un ficher d'une trace
        1er  ligne : nombre de points
        2eme ligne : déplacement moteur selon x (m2)
        3eme ligne : déplacement moteur selon y (m1)
    """

    fichier.write(str(len(liste_x))+'\n')

    for elem in liste_x:
        info = str(elem)+' '
        fichier.write(info)

    fichier.write('\n')

    for elem in liste_y:
        info = str(elem)+' '
        fichier.write(info)

    fichier.write('\n')
```

```
def stockage_fichier(liste_traces :list, nom_fichier="data.txt") -> None:
    """
    Stockage des traces dans un fichier susnommé (defaut : data.txt)
            1er ligne : nombre de trajectoire
            ecriture des traces avec la fonction "ecriture_trace"


        liste_traces : listes contenant pour chaque trace la liste des mouvements des deux
moteurs
    """
    #ouverture fichier
    fichier = open(nom_fichier,"a")

    #ecriture du nombre de trace

    fichier.write('\nNouvelle trace\n')

    #ecriture des traces
    for elem in liste_traces:
        liste_x, liste_y = elem
        ecriture_trace(liste_x, liste_y, fichier)

    #fermeture fichier
    fichier.close()


if  __name__=='__main__':
    print('done fonctions_fichier')


#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
"""


#---------------------------           ---------------------------------------


def conversion_deplacement_moteur(liste:list, point_precedent :(int,int))-> list:
    """
    convertie la liste d'abscisse en liste de deplacement moteur
        point inital à partir de l'origine
        deplacement relatif par rapport au point precedent
    """
    l = [liste[0] - point_precedent] #point initial par rapport au point de la trace precedente

    for k in range(len(liste)-1):
        l.append(liste[k+1]-liste[k]) # ecart au point précédent

    return l
```

```
if  __name__=='__main__':
    print('done fonctions_generales')


#!/usr/bin/env python3
# -*- coding: utf-8 -*-



from fonctions_fichier import stockage_fichier

from fonctions_generales import conversion_deplacement_moteur
from fonctions_VectorDrawable import listes_pointsSVG




def programme()-> None :

info = str(input("\n rentrer les données SVG : ->"))

liste_x, liste_y = listes_pointsSVG(info)

#test pour inkscape
#u = ['M']
#X,Y = liste_x, liste_y
#for k in range(len(Y)):
      #     u.append(X[k])
      #     u.append(Y[k])
#print(u)

liste_traces                    =                   [(conversion_deplacement_moteur(liste_x,0),
conversion_deplacement_moteur(liste_y,0))]
stockage_fichier(liste_traces)

if  __name__ == '__main__':

    programme()
    print('\n programme ok ')
```