Tancrède ESNOUF
Sacha MUGNIER
Mathieu MOURGUES
Maratt SAUR

Livrable final :

Rapport technique

Tablette Traçante

Gr.1

Tancrède ESNOUF
Sacha MUGNIER
Mathieu MOURGUES
Maratt SAUR

I. Comprendre le sujet

1.1 Introduction:

_____Notre projet s'intéresse au fonctionnement d'une tablette traçante XY-plotter (voir figure 4 annexe pour avoir une vue d'ensemble). Celle-ci est composée de trois moteurs cependant nous n' utilisons que deux qui sont les moteurs de déplacement vertical et horizontal.

L'objectif initial de ce projet était de pouvoir tracer grâce à la tablette traçante, un contour d'une image préalablement choisie par un utilisateur. Au fur et à mesure du projet, en vue des difficultés à vectoriser une image et la tracer grâce à la table, nous avons dû nous contraindre à étudier le tracé d'une image de forme simple.

L'objectif retenu est de mettre en place une bibliothèque de formes géométriques que l'utilisateur pourrait choisir de tracer. Ce dernier demande quelle forme géométrique il souhaite tracer et la table exécute sa demande. Pour cela, nous avions mis en place une interface Homme-Machine -ou liaison série- avec le logiciel TeraTerm.

1.2 Différents rôles

Au commencement du projet, le travail a été séparé en différents groupes, une partie axée sur l'utilisation d'une image vectorielle -comment la tracer, comment la représenter en matrice pour que la carte puisse la traiter..., et une autre chargée de comprendre les branchements à réaliser, les caractéristiques des moteurs (comment modifier la période et le nombre de tour, ainsi que la vitesse de chaque moteur).

Mathieu et Sacha (les programmeurs du groupe) :

Notre travail a globalement tourné autour des différents codes en C pour faire fonctionner la table. De nombreux codes et tests ont été effectués tout au long des séances : essayer de vectoriser une image, comprendre le fonctionnement des périodes des moteurs X et Y pour pouvoir tracer différents angles...

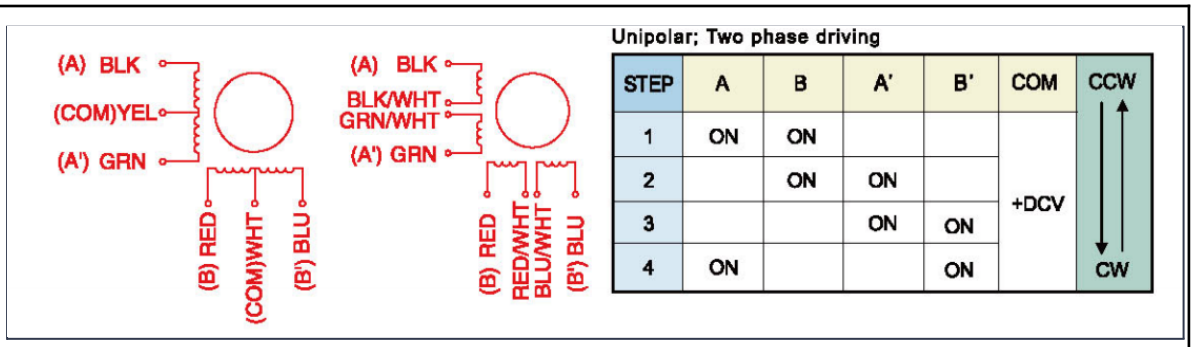
Tancrède et Maratt (les techniciens du groupe)

Nous nous sommes chargés des branchements entre les différents modules, -pont en H, platines L297 et L298N mais aussi de la mise en place de la liaison série grâce au logiciel TeraTerm. Mais également nous avons étudié leur différentes fonctions, comment nous devons les brancher, quel type d'information ils transmettent...

Globalement, au fur à mesure des séances, nous avons travaillé davantage en groupe pour comprendre, analyser et corriger chaque détails de nos codes et pour se débloquer des erreurs de compilations.

1.3 Fonctionnalités

Afin de tracer des formes géométriques simples nous devons d'abord coder des déplacements verticaux et horizontaux. Puis nous aurions besoin de coder des déplacements selon un certain angle. Les moteurs pas à pas ont comme caractéristiques leur sens de rotation, le nombre de pas et la vitesse de rotation. Théoriquement, pour faire tourner pendant un tour un moteur de N pas à bobines il faut alimenter simultanément chaque bobine N fois (voir figure 1). Cependant, cela serait trop fastidieux et impliquerait de trop long codes, donc on utilise deux platines L297 et L298N qui permettent grâce à leur différentes entrées de contrôler ces paramètres (voir figure 2).



Unipolar; Two phase driving

STEP	A	B	A'	B'	COM	CCW
1	ON	ON			+DCV	↓
2		ON	ON			
3			ON	ON		
4	ON			ON		↑ CW

Figure 1: Schéma explicatif d'un moteur pas à pas. illustration : 0900766b813e843f.pdf (rs-online.com)

1.3.1. Théorie

La vitesse de rotation des moteurs est contrôlée par la sortie Pwm clock. En changeant, la valeur de sa période on peut fixer la vitesse de rotation donc la vitesse de translation des moteurs.

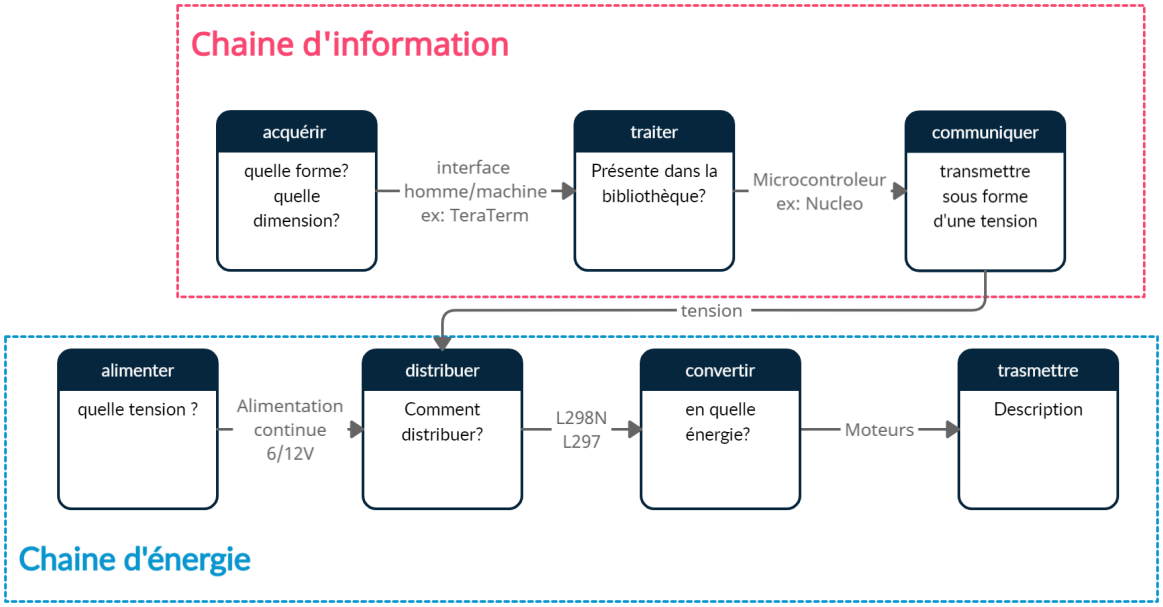
Pour faire varier le sens de rotation, on utilise la commande *cw* qui peut prendre comme valeur 0 -sens antihoraire- ou 1 -sens horaire. Et la commande *half* permet d'améliorer la précision des moteurs c'est-à-dire, tourner avec des demi-pas au lieu de pas entier.
En fixant, le temps de rotation d'un moteur, on peut le faire fonctionner sur une certaine distance à une vitesse constante. De même, en réglant le rapport des

périodes des signaux créneaux des deux moteurs, on peut faire avancer la pointe du stylo selon un angle $\theta = periode_Y/periode_X$.

II. Réaliser le prototype

2.1 Schémas électriques

Voici le schéma bloc de notre système. Afin d'obtenir les données de l'utilisateur, une interface homme-machine est mise en place grâce au logiciel TeraTerm. La partie programmation sera assurée par le microcontrôleur qui communique avec les modules L298N et L297.



Pour contrôler les moteurs pas à pas et pour leur fournir une alimentation suffisante, nous avons réalisé un montage dit pont en H grâce aux modules L298N et L297, relié au microcontrôleur qu'est la carte Nucleo. La platine L298N est alimentée sous 6-12V par une alimentation en continue, elle connecte la platine d'entrée L297 avec les bobines du moteur pas à pas (voir figure 2). La platine L297 est composée de connecteurs d'entrée qui reçoivent les commandes du microcontrôleur et qui nécessitent d'être sous tension. Le détails des branchements est présenté juste au dessous (voir figure 3).

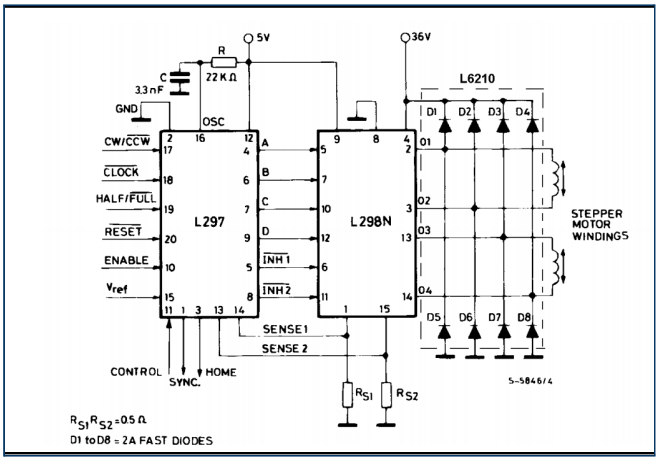


Figure 2: Schéma des différents branchements entre les platines L298N et L297. Source:

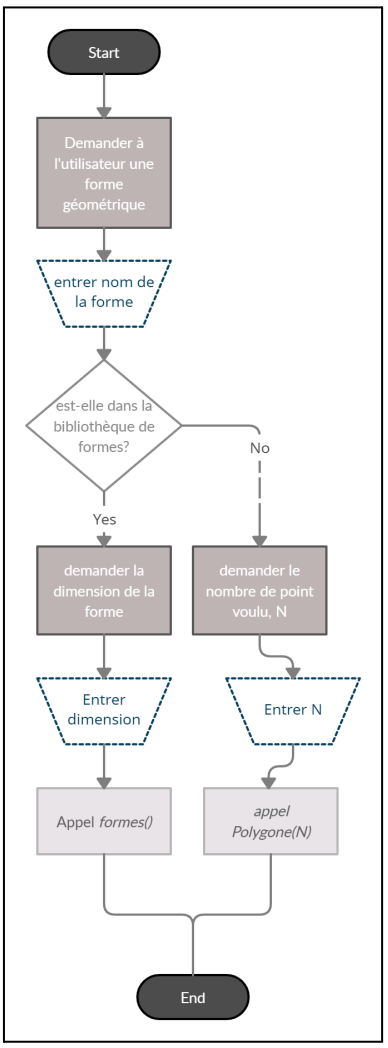
La carte L297 est reliée aux sorties arduino de la carte Nucleo, d'après le tableau suivant:

Entrées	cw	clock	half	reset	enable	Vref	control
Type/ alimentation	Digital	Pwm	Digital	5V	5V	5V	5V

Figure 3: Détails des branchements entre la carte Nucleo et le module L297.

2.2 Algorithmes

Le projet suit l'algorithme suivant. L'utilisateur entre un nom de forme géométrique, si celui-ci est dans la bibliothèque de fonctions présentes alors il doit entrer la dimension voulue. Sinon, il entre la valeur de N, le nombre de points du polygone souhaité. Dans ce dernier cas malheureusement il ne pourra pas changer la dimension des côtés.



III. Caractériser/ Valider le produit final

3.1) Tests de bases :

Au début du projet, nous avons fait des tests pour pouvoir faire marcher le moteur X et le moteur Y séparément, que le nombre de tours du moteur souhaité faisait bien la distance voulue.
Ensuite, nous avons fait marcher les moteurs X et Y simultanément avec la même période pour obtenir un angle de 45°. Nous avons eu quelques problèmes avec le moteur X qui se lançait une fois sur deux, ce qui nous a ralenti.

3.2) Tests des capteurs

Nous avions, initialement branchés deux capteurs, un selon X et l'autre selon Y, en fin de course des moteurs. Nous les avions câblés dans l'optique de créer une

fonction qui placerait le crayon au centre de la table, pour un retour à l'origine. Après plusieurs tests de cette fonctionnalité, nous n'arrivions à rien de concluant. Les câbles avec lesquels nous avions soudé le capteur X devaient être défectueux car le capteur marchait très mal voir pas du tout.
Nous avons donc décidé de supprimer cette fonctionnalité, qui nous prenait du temps à mettre en place et qui ne fonctionnait pas, pour nous concentrer sur le cœur du projet.

3.3) Tests des différentes fonctionnalités

Nous avons testé séparément tous nos programmes de traçage de forme avant de les relier à une requête sur l'ordinateur avec TeraTerm.
Le majeur problème rencontré était avec la fonction cercle(), nous n'arrivions pas à tracer un cercle avec plus de 8 points, les calculs des distances entre les points buguaient. Nous nous sommes placés en demi-pas (ce qui a augmenté la précision) et nous avons pu tracer un cercle avec plus de 12 points.

Après avoir installé la liaison série avec le logiciel TeraTerm, nous avons vérifié qu'elle fonctionnait bien. Grâce aux commandes "pc.printf" le long des différents code, nous avons pu vérifier que la liaison était effective et en plus déboguer nos codes comme la fonction cercle().

3.4) Validation des fonctionnalités.

Enfin, nous avons fait fonctionner simultanément nos fonctions et validé une par une la correspondance entre la forme rentrée sur TeraTerm et la forme effectivement tracée par la table. C'est ainsi que nous avons pu valider le fonctionnement de notre projet.

IV. Comprendre les étapes de réalisations/ choix

4.1. Planning

1ère séance :

- ❏ Comprendre la programmation d'un moteur pas à pas.
- ❏ Faire tourner le moteur d'essai avec deux bobines via la carte nucleo
- ❏ Première approche du fonctionnement des moteurs mis à disposition de la tablette traçante.

2ème séance :

- ❏ Définir l'organisation/déroulement du projet : planning, schéma bloc, fonctionnement moteur de la table traçante

- ❏ Programmation des moteurs pas-à-pas.
- ❏ Familiariser avec les platines L298N et L297.
- ❏ Coder le fonctionnement d'un moteur.

3ème séance : 23/03

- ❏ Câbler le montage de la semaine dernière, faire un essai de fonctionnement
  GROUPE 1
- ❏ Se renseigner pour traiter une image vectorielle noire ou blanche 1D.
  GROUPE 2
- ❏ Étudier la correspondance entre le nombre de tours et le rapport cyclique des pulses envoyés (f(α))
- ❏ Étudier le lien entre le nombre de tour et le déplacement sur les axes X et Y (lois entrée-sortie)
  Marges prévues:
  - On n'attend pas en fin de cette séance que le code soit entièrement réalisé/ écrit. Cette problématique pourra être réétudiée pour les prochaines séances.
  - On attend que la relation entre la translation du chariot et la rotation du moteur s'est explicitement comprise, étudiée et exploitée.

4ème séance : 30/03

- ❏ Écrire le code pour traiter une image vectorielle noire ou blanche 1D.
- ❏ Tracer une figure simple en une seule dimension.
  - ❏ ligne droite de longueur préalablement définie et variante
  - ❏ pointillés
- ❏ Tracer **séparément** sur les deux axes X et Y des consignes simples.

5ème séance : 6/04

- ❏ Écrire le code pour traiter une image vectorielle noire ou blanche 2D.
  - ❏ Tracer **séparément** sur les deux axes X et Y une image.
- ❏ Mettre en place des capteurs permettant de fixer l'origine sur la tablette. (non abouti)
- ❏

6ème séance : 12/04

- ❏ Créer la bibliothèque de formes géométriques
  - ❏ Mettre en place une fonction permettant de tracer une forme géométrique de N points.
    GROUPE 1
  - ❏ Mettre en place une **communication RS232** entre le PC et la carte Nucleo
    GROUPE 2

- ❏ Écrire le code demandant à l'utilisateur d'entrer la forme choisie.

Marges prévues:

A l'issue de cette séance, la communication RS232 sera parfaitement comprise et mise en application. La fonction permettant de tracer une forme de N points pourra ne pas être achevée durant cette séance.

13 Avril première présentation du projet

7ème séance : 4/05

- ❏ Revue et correction de la présentation du 13 avril.
- ❏ Finir fonction permettant de tracer une forme géométrique de N points.
- ❏ Écrire le code demandant à l'utilisateur d'entrer la forme choisie.

7ème séance (bis): 6/05

- ❏ Finir le codage de la fonction permettant de tracer n'importe quelle forme géométrique de N points.

8ème séance : 11/05

- ❏ Présenter le travail.

4.2. Difficultés rencontrées

A posteriori, nous n'avons pas pu tenir correctement le planning due à l'accumulation de problèmes rencontrés, de différentes natures:
Tout d'abord, de nombreux problèmes de branchements nous ont freiné comme des câbles détériorés, des faux contacts, le moteur X qui se bloquait sans raison... En effet, le montage nécessitait de nombreux câbles afin de relier les différents modules et les faux contacts étaient fréquents.
De plus, nous avons rencontré un certain nombre d'erreurs lorsqu'on compilait les codes. Par exemple, il a fallu forcer le programme à "ne rien faire" afin qu'il ne déroule pas le code. En effet, pour la fonction *angle()* (cf annexe), le programme ne tenait pas compte de la condition de la boucle *while()* qui exigeait d'attendre que le compteur se décrémente jusqu'à la valeur 0, pour corriger cela nous avons dû rajouter la condition, *compteurX != o){__nop();}* .
De plus, certaines séances où nous aurions dû étudier plusieurs fonctionnalités différentes ont été gênées par du matériel défectueux comme par exemple, initialement nous avions prévu d'intégrer des capteurs de fin de courses pour repérer une origine des axes sur la tablette. Malheureusement , ceux-ci ont dû être retirés car ils ne fonctionnaient pas correctement -après avoir remis en question de nombreuses fois notre code associé.

Cependant, nous avons su mettre de côté les fonctionnalités et objectifs secondaires qui exigeaient plus de temps pour nous focaliser sur la réalisation d'autres objectifs. Nous avons su garder notre sang froid et continuer d'avancer malgré tout.

Le travail d'équipe nous a permis de nous soutenir face aux différents problèmes et de répartir la charge de travail. Comme chacun de nous avait des compétences particulières, nous avons pu répartir et profiter de chacun. Par exemple, face à un problème chacun de nous proposait une solution, ce qui nous a permis de le résoudre rapidement. De même, chacun de nous avait un point de vue différent sur la manière de coder les différentes fonctions, après délibération nous avons convergé vers une solution optimale. Cependant, la difficulté dans un travail en groupe reste avant tout la capacité propre à chacun de rester attentif, garder son sang froid et d'écouter les autres. En résumé, notre groupe n'a pas été freiné par des différences entre les membres, nous avons plutôt réussi à nous coordonner et à avancer selon les performances de chacun -prendre une pause quand le besoin s'en faisait sentir et partager les différentes tâches selon les avis en les aptitudes.

V. Annexe

Détails des codes.

Déclarations des sorties:

```
#include "mbed.h"
#include <math.h>
// Moteur X
DigitalOut halfX(D15);
PwmOut aclockX(D6);
DigitalOut cwX(D13);
// Moteur Y
DigitalOut halfY(D7);
PwmOut aclockY(D11);
DigitalOut cwY(D9);
// Déclaration liaison série.
Serial pc(USBTX, USBRX);
// Tickers
Ticker time1;
Ticker time2;
// initialisations compteurs
int compteurX;
int compteurY;
```

Codes de la fonction de décrémentation

```
void incrementX() {
  if(compteurX != 0){
  compteurX -= 1; }}

void incrementY(){
  if(compteurY != 0){
  compteurY -= 1; }}
```

Codes simples de déplacement

```
void diagonale(int sX, int sY){
  time1.attach(&incrementX, 1); // la fonction attach() appelle la fonction incrémentX toutes les secondes
  // rotation x
  aclockX.period_ms(10);
  aclockX.write(0.5);  // paramètre non essentiel, permet simplement d'avoir un déplacement fluide
  halfX=0;
  cw=sX;
```

```
  // rotation y
  aclockY.period_ms(10);
  aclockY.write(0.5);
  halfY=0;
  cwY=sY;
  while(compteurX != 0){___nop();}
  aclockX.write(0);
  aclockY.write(0);}

 void ligne_droite_x(int sX){
  time1.attach(&incrementX, 1);
  aclockX.period_ms(10);
  aclockX.write(0.5);
  halfX=0;
  cwX=sX;
  while(compteurX != 0){___nop();}
  aclockX.write(0);
  time1.detach();}

void ligne_droite_y(int sY){
  time2.attach(&incrementY, 1);
  aclockY.period_ms(10);
  aclockY.write(0.5);
  halfY=0;
  cwY=sY;
  while(compteurY != 0){___nop();}
  aclockY.write(0);
  time2.detach();}

void hexagone(int D){
// Ici l'entier D permet de fixer la valeur du compteur donc celle du trait. Sachant qu'un tour est de l'ordre de 3,5 cm, et qu'un tour est réalisé pour deux décrémentation du compteur donc la taille du trait est égale à D*3,5/2 cm.
   compteurY=D;
   compteurX=D;
   ligne_droite_y(1);
   compteurY=D;
   diagonale(1,1);
   compteurX=D;
   ligne_droite_x(1);
   compteurX=D;
```

```
    diagonale(1,0);
    ligne_droite_y(0);
    compteurY=D;
    compteurX=D;
    diagonale(0,0);
    compteurX=D;
    ligne_droite_x(0);
    compteurX=D;
    diagonale(0,1);}

void carre(int D){
   compteurX=D;
   ligne_droite_x(0);
   compteurY=D;
   ligne_droite_y(0);
   compteurX=D;
   ligne_droite_x(1);
   compteurY=D;
   ligne_droite_y(1);}

void losange(int D){
   compteurX=D;
   diagonale(0,0);
   compteurX=D;
   diagonale(0,1);
   compteurX=D;
   diagonale(1, 1);
   compteurX=D;
   diagonale(1,0); }
```

Codes tracé d'un polygone de N points:

```
/* Fonction permettant de tracer un cercle mais plus généralement une forme géométrique de N points.
Entrée: nombre de points pts, et la précision en demi pas ou pas  grâce à la commande pas.*/

int Polygome(int pts, int pas){
  //Traçage d'un polynôme de N points
 int i;
  int sX; // Défini le sens de rotation du moteur horizontal
  int sY; // Défini le sens de rotation du moteur vertical
  double X[pts]; // Nombre de points pour chaque matrice X et Y
  double Y[pts];
  double theta[pts]; // Liste de déplacements angulaires
  for(i=0;i<pts;i++){
    theta[i]=i*6.28/pts;}   // Remplissage des 3 listes
  for(i=0;i<pts;i++){
    X[i]=r*cos(theta[i]);
    Y[i]=r*sin(theta[i]);}
   for(i=0;i<pts-1;i++){
    //Calcul des distances selon x et y
    compteurX = 1;
    double x=X[i+1]-X[i]; // définitions du pas à effectuer entre chaque point
    double y=Y[i+1]-Y[i];
    if (x>0)
      {sX=0;} // si le pas est positif on tourne dans le sens anti-horaire
    else
      {sX=1;
      x = -x;}  // Important car dans la suite on ne pourra pas définir une sortie PWM de période négative.
    if (y>0)
        {sY=0;}
    else
        {sY=1; y = -y;}
    angle(sX,sY,y*10,x*10,pas);} // on appelle la fonction angle()
    compteurX=1; // On redéfinit un compteur pour le dernier point à tracer
    double x=X[0]-X[pts-1];
    double y=Y[0]-Y[pts-1];
    if (x>0)
      {sX=0;}
    else
        {sX=1;
        x = -x;}
    if (y>0)
```

```
        {sY=0;}
    else
        {sY=1; y = -y;}
    if ((y<0.0001)&(-0.0001<y)){ //Dans cette borne, le moteur s'emballe, on ne peut pas traiter des écarts si faibles.
      y=0;}
    if ((x<0.0001)&(-0.0001<x)){
      x=0;}
    angle(sX,sY,y*10,x*10,pas);
  while(1){}
}
```

Code pour tracer un angle

```
/* Fonction permettant de tracer un angle préalablement choisi comme un rapport des deux périodes des moteurs, par rapport à l'horizontal.
Argument d'entrée:  le sens de rotation sX et sY en donnant la valeur 0 ou 1 pour orienté le tracé de l'angle
La période T_X et T_Y -des réels- permettant de tracer un angle de valeur T_Y/T_X.
Le pas qui peut prendre comme valeur 0 ou 1, selon la précision voulue.
Auteurs; Esnouf, Mourgues, Mugnier et Saur, 18/05/2021.*/

void angle(int sX, int sY, double T_X, double T_Y, int pas){
  time1.attach(&incrementX, 1); // appelle la fonction attach() toutes les secondes
  // rotation x
  if(T_X != 0){
    aclockX.period_ms(T_X);
    aclockX.write(0.5);}
  else{
    aclockX.write(0);}
  halfX=pas; // choisir si on est se place en pas entier ou demi pas
  cwX=sX;
  // rotation y
  if(T_Y != 0){
    aclockY.period_ms(T_Y);
    aclockY.write(0.5); }
  else{
    aclockY.write(0);}
  halfY=pas;
  cwY=sY;
  while(compteurX != 0){___nop();} // on force l'algorithme à ne rien faire
```

```
  aclockX.write(0);
  aclockY.write(0);} // on arrête la rotation des moteurs
```

Code Final

```
/* fonction permettant de demander à l'utilisateur d'entrer la forme géométrique souhaitée.*/
void Projet_final(){
  char forme[256];
  int N;
  int D;
  pc.printf("entrer la forme voulue \r\n");
  pc.scanf("%s",&forme);
  pc.printf("%s", forme);
  if (strcmp(forme,"losange") == 0){ // strcmp évalue la chaine de caractère rentrée avec celle en argument.
        pc.printf("entrer la dimension voulue \r\n");
        pc.scanf("%d",&D);
        losange(D);}
   else {
            if (strcmp(forme,"carre") == 0){
                  pc.printf("entrer la dimension voulue \r\n");
                   pc.scanf("%d",&D);
                  carre(D);}
            else {
                   if(strcmp(forme,"hexagone") == 0) {
                        pc.printf("entrer la dimension voulue \r\n");
                        pc.scanf("%d",&D);
                        hexagone(D);}
                  else {
                        pc.printf("entrer le nombre de point \r\n");
                        pc.scanf("%d",&N);
                        Polygone(N);}}}}
```

VI. Bibliographie

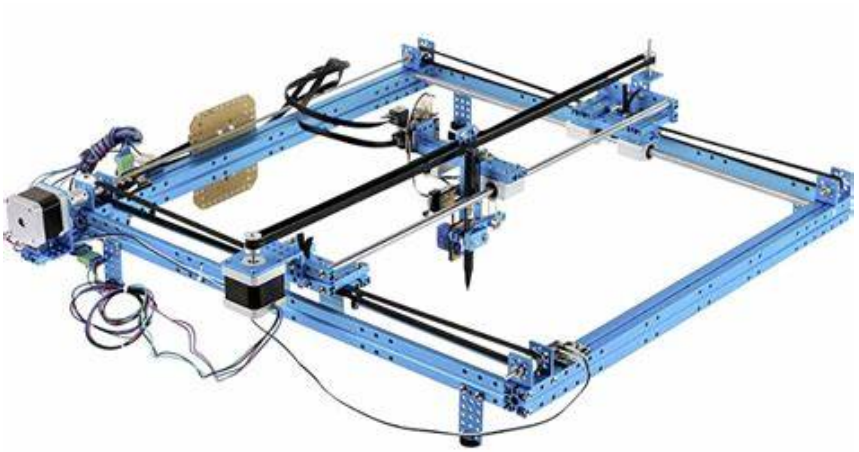


Figure 4: Vu d'ensemble de la tablette traçante. Image empruntée à Makeblock-Roboter-Bausatz-XY-Plotter-Robot-Kit-V2.0.jpg (1822×1025) (re-in.de)

Doc du moteur:
0900766b813e843f.pdf (rs-online.com) et figure 1:
Doc lense moteur:
Présentation PowerPoint (institutoptique.fr)
Fiche technique avec quelques caractéristiques:
http://www.orbit-dz.com/product/xy-plotter-robot-kit-v2-0-p-with-electronic/

Complément sur les modules L297N et L298 et figure 2:
Commande de moteur pas à pas avec circuit L297 (positron-libre.com)