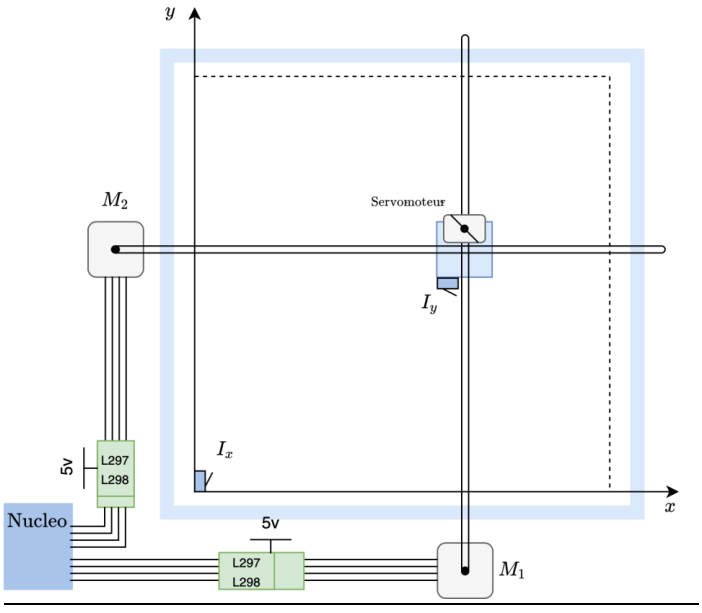
Rapport technique
Table traçante



Corentin Nannini, Selena Rippe, Coline Auffret et Florian Motyl

Introduction :

Notre but est de pouvoir tracer des figures complexes (dessins…) à l'aide de la table traçante. Nous avons décidé de nous intéresser dans un premier temps à la mise en marche de nos moteurs pas à pas permettant de réaliser un tracé quelconque.

Une fois cette étape réalisée, notre objectif a été de câbler un joystick afin d'être en mesure de diriger notre stylo où on voulait, et donc de pouvoir dessiner.

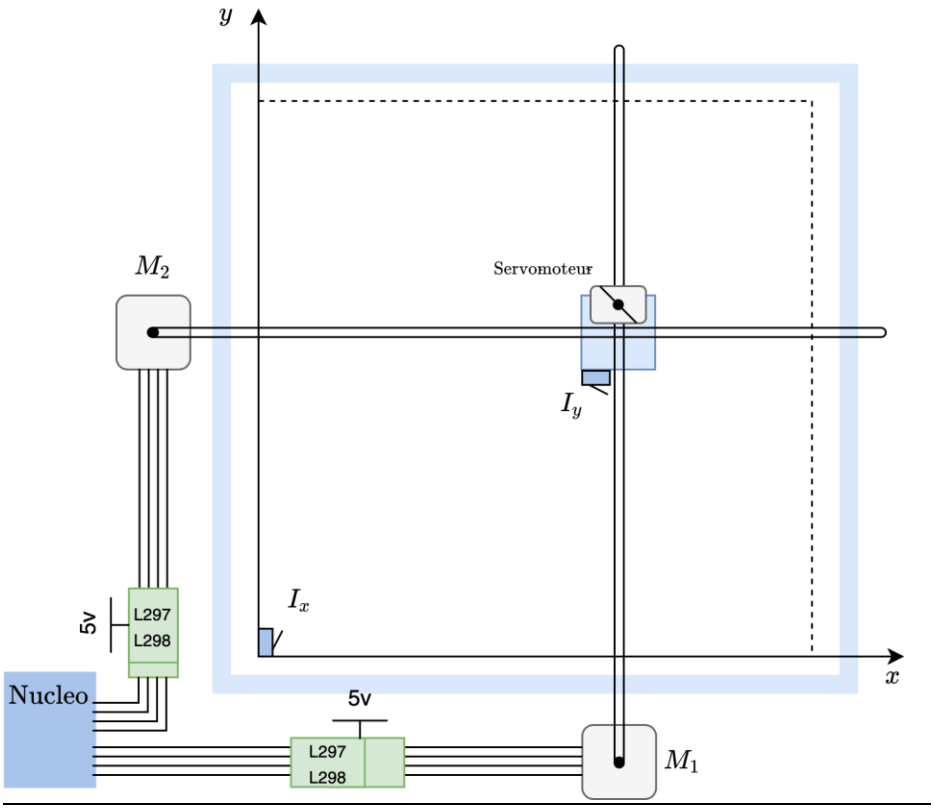
Enfin, nous avons voulu coder un jeu de morpion pour rendre plus intéressante l'utilisation de la table traçante et du joystick.

Matériels utilisés :

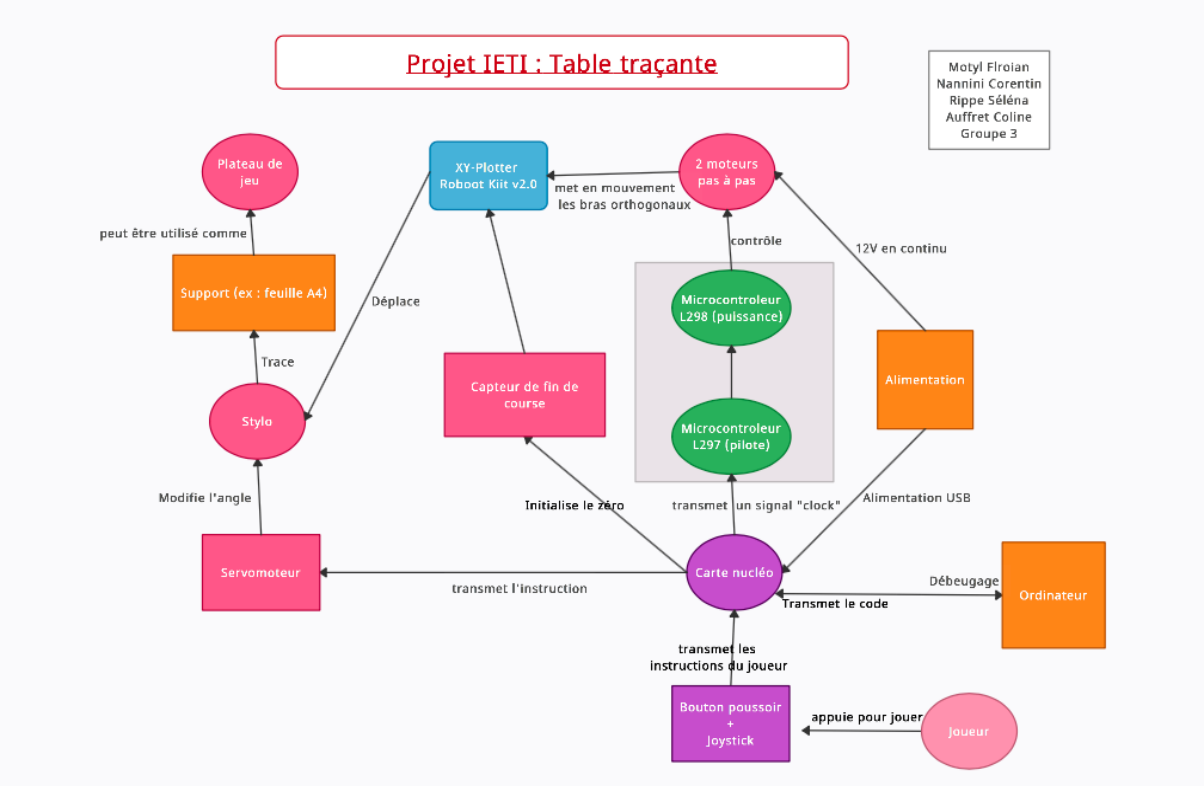
- **2 moteurs pas à pas sur leurs bras perpendiculaires**
- **1 servomoteur**
- **1 système permettant de tenir le stylo à l'intérieur de la structure de bras**
- **2 cartes L298**
- **2 cartes L297**
- **1 carte Nucléo**
- **1 Joystick Parallax 27800**
- **Des fils**
- **Une "bread board"**
- **Un GBF**
- **Une alimentation courant continu (deux voies)**
- **Un compilateur C vers STM32 : Mbed**
- **Stylo pour tracer**
- **Feuille de papier pour tracer dessus**

Dans ce rapport technique, nous allons vous détailler les étapes de réalisation de nos différents objectifs afin que vous puissiez les réaliser vous même.

Schéma du système :



Découpage fonctionnel :



Descriptif des fonctionnalités :

À chaque démarrage du programme, le système effectue une mise à zéro. Pour cela on utilise deux boutons de fin de course. Cela servait originellement à pouvoir définir l'origine pour un repère de coordonnées.

Le joystick (avec le bouton poussoir) est dirigé par un joueur, dans n'importe quelle direction (360°). La direction choisie par le joueur fournit une information qui va être traitée par la carte Nucléo. La nucléo reçoit sur deux canaux des valeurs de tension (en `analog_inx.read_u16()`**) qui sont reliées à l'angle du joystick. Avec de la trigonométrie on peut aisément relier chaque couple de tension à une valeur d'angle. Cela donne la fonction** `read_joy(int *en)` **que l'on peut voir en annexe.**

Pour utiliser cette valeur d'angle pour orienter le tracé il faut pouvoir faire en sorte de tracer des traits obliques. Cela n'est pas aisé au premier abord car les deux moteurs ne peuvent pas facilement être contrôlés simultanément directement avec les signaux du microcontrôleur (Nucléo). Cela donne lieu à la fonction `void deplacement_angle(double dist, double angle).` **La manière dont nous avons procédé pour effectuer des tracés avec un angle sera expliquée dans une partie suivante, cependant il faut noter l'utilisation de deux cartes électroniques qui sont placées entre le microcontrôleur et les moteurs : les L298 et L297.**

Les cartes L297 ont pour rôle de recevoir des informations d'horloge pour piloter la vitesse de rotation du moteur (en PWM) et des informations de fonctionnement (HALF/FULL pour contrôler le pas complet ou demi-pas, ENABLE pour activer ou désactiver le moteur, CW/CCW le sens de rotation…) et de les transformer en signal que le moteur peut recevoir (des pulses sur quatres canaux avec une certaines synchronisation pour mettre en place un rotation).

La carte L298 quant à elle permet de gérer la puissance fournie au moteur. En effet, les moteurs demandent chacun environ 6W pour fonctionner correctement, la carte Nucléo ne peut fournir aisément une telle puissance. Ainsi, la L298 placée après la L297 est alimentée par une alimentation externe. Ainsi, les moteurs tirent alors leur puissance de l'alimentation de laboratoire.

Le moteur fonctionne donc et met en mouvement les bras orthogonaux de telle sorte à ce qu'il fasse bouger le stylo de la table traçante dans la direction que lui a imposé le joueur au départ.

Nous avons également la possibilité que l'information fournie aux moteurs proviennent non pas du joystick mais d'un code de l'ordinateur. Dans ces cas-là, le code est analysé par la carte nucléo qui va alors, tout comme avec le joystick, faire remonter l'information jusqu'à la table traçante.

Par ailleurs, une fonctionnalité essentielle de notre système est le servomoteur. Celui-ci nous sert à contrôler si le stylo touche la feuille ou s'il est levé pour ne pas qu'il écrive. Ce servomoteur est codé à l'aide de l'ordinateur.

Dans le cas de notre morpion, nous avons créé un code qui trace directement les cases nécessaires au jeu. On voit alors l'utilité du servomoteur lors de cette étape. Puis c'est ensuite au joueur, avec le joystick de choisir la case où il veut jouer. Le code du morpion transmet en conséquence, l'information à la carte nucléo nécessaire à la réalisation d'un rond ou d'une croix dans le cas choisie par le joueur.

Faire des tracés avec un angle

À l'aide des branchements PWM, et vu qu'on a affaire à des moteurs pas à pas, la carte Nucléo transforme l'information analogique de la direction en une information "clock", c'est-à-dire qui délivre un signal plus ou moins long à 1 ou 0, spécifique à la direction choisie par le joueur.

Pour déplacer un moteur d'un certain nombre de pas (le moteur a des pas discrets qui correspondent à l'angle entre deux positions successives du rotors), on le met en marche en activant un PWM avec une période T voulue, on met en pause le programme durant une durée nT puis on met fin au signal PWM. On effectue alors un pas sur ce moteur.

Pour effectuer des tracés obliques il faut pouvoir actionner les deux moteurs simultanément. C'est pour cela que l'on utilise des cartes L297, celles-ci reçoivent un signal en PWM avec une certaine fréquence et génèrent un signal adéquat pour actionner les moteurs. La fréquence des signaux PWM permet de contrôler la vitesse des moteurs. Ainsi, si pendant une durée τ le moteur 1 a une vitesse v_1 et le moteur 2 une vitesse v_2, alors la tête va effectuer un angle $\alpha = arctan\left(\frac{v_1}{v_2}\right)$ (en prenant garde aux sens de déplacement des moteurs). En jouant sur le sens de déplacement des moteurs (pin CW/CCW sur la L297, cf. fiche technique du composant), on peut effectuer tous les angles entre 0° et 360°.

Il faut prendre garde à certains cas limites. Lors du calcule de la vitesse adéquate pour certains angles (typiquement 0°, 90°…), l'un des deux moteurs aura une valeur de période de PWM très grande. Cela peut entraîner une erreur dans l'exécution du programme du microcontrôleur. On a pu régler ce problème en désactivant le moteur (a l'aide d'un variable `enableX`, om `X` désigne le numéro du moteur) au-delà d'une certaine valeur seuil ($T > 1000\,ms$) de période de PWM.

Le servo moteur

Le servomoteur est placé en dessous du stylo, il permet de le lever ou descendre pour tracer ou non. Une variable globale de notre programme nous permet de connaître son état en permanence. On dispose alors de deux fonctions `void servo_up()` **et** `void servo_down()` **qui servent à piloter la position du servomoteur et de changer son état dans le programme.**

Analysons

```
void servo_up(){
    servo_mot.write(0.05);
    servo = 1;
}
```

Où `servo_mot` **est une sortie en PWM, on règle alors la position angulaire du servomoteur avec la rapport cyclique de** `servo_mot`**. Enfin on affecte a la variable** `servo` **la bonne valeur d'état (1 levé et 0 descendu).**

Repère de coordonnée

Dans notre programme nous avons des variables globales : `x`**,** `y`**,** `xstep` **et** `ystep` **qui stockent les coordonnées de la position de la tête à chaque instant.** `x`**,** `y` **en millimètre et** `( xstep, ystep)` **en nombre de pas moteur. Ces variables sont mises à jour à chaque appel à une fonction de déplacement.**

Les coordonnées n'ont pas pu être utilisées dans notre programme, en effet à chaque déplacement, il y un légère différence entre le nombre de pas réels qu'effectue les moteur et celui calculé dans le programme. On voit très vite des erreurs importantes entre la position réelle de la tête et celle dans le programme.

Ce système aurait pu nous permettre de tracer un trait entre deux points de coordonnées connues et donc potentiellement tracer des figures plus complexes. Le jeu de morpion aussi se servait de ces coordonnées.

Annexe 1 : Code

```
/* mbed Microcontroller Library
 * Copyright (c) 2019 ARM Limited
 * SPDX-License-Identifier: Apache-2.0
 */

#include "mbed.h"
#include "platform/mbed_thread.h"
#include <math.h>
#define Tmin 8
#define NB_LIG 3
#define NB_COL 3

PwmOut mot1(D5); // sortie du moteur 1
PwmOut mot2(D6); // Sortie de moteur 2
DigitalOut sens1(D3); // sens de fonctionnement moteur 1 (1 ou 0)
DigitalOut sens2(D4); // sens de fonctionnement moteur 2 (1 ou 0)
DigitalIn fin1(A1); // variables stockant l'état des boutons poussoirs
DigitalIn fin2(A0);
DigitalIn bouton(USER_BUTTON);
AnalogIn analog_inx(A2); // variables pour l'usage du joystick
AnalogIn analog_iny(A3);
int enable_joy = 1; // variable pour empecher l'action du joystick si celui-ci n'est
pas actioner
PwmOut servo_mot(D10);
Serial pc(USBTX, USBRX);
int servo = 1; // stocke l'état du servo moteur 1-> monté; 0-> baissé
//variable stockant la position de la tête à chaque instant
double x = 0, y = 0;
int xstep = 0, ystep = 0 ;
//
//rapport cyclique envoyé au cartes controlant le moteur
double RC = 0.2;


typedef enum {FALSE, TRUE} Boolean;
int grille[NB_LIG][NB_COL]; /* grille du morpion valeurs possibles VIDE=0, ROND=1 ou
CROIX=2 */

/* indique quel sera le prochain joueur a mettre un pion dans la grille ie soit ROND
soit CROIX */
int prochainJoueur = 2;


void INIT_ORIGIN(){
    int init1, init2;
    int WAIT=2000;
    int t = 5;
    mot1.period_ms(t);
    mot2.period_ms(t);

    mot1.write(0);
```

```
    mot2.write(0);

    sens1 = 0;
    sens2 = 1;

    init1 = fin1.read();
    init2 = fin2.read();

    while(init1 == 1){
        mot1.write(RC);
        thread_sleep_for(t);
        mot1.write(0);
        init1 = fin1.read();
    }
    mot1.write(0);

    while(init2 == 1){
        mot2.write(RC);
        thread_sleep_for(t);
        mot2.write(0);
        init2 = fin2.read();
    }
    mot2.write(0);

    x = 0;
    y = 0;
    ystep = 0;
    xstep = 0;

}

void servo_up(){
    servo_mot.write(0.05);
    servo = 1;
}

void servo_down(){
    servo_mot.write(0.075);
    servo = 0;
}

int dist_to_NbStep(double dist){
    // Dist en mm
    int res;
    res = (dist * 1000)/184;
    return res;
}

// fonction qui deplace les moteurs individuellement
void deplacement_mot1(int NbS, int sens, int T){
    mot1.period_ms(T);
    sens1 = sens;
    mot1.write(RC);
    thread_sleep_for(NbS*T);
    mot1.write(0);
    x = x + (2*sens -1)*NbS *1.0* 184 / 1000;
    xstep = xstep + + (2*sens -1)*NbS;
}
```

```
void deplacement_mot2(int NbS, int sens, int T){
    mot2.period_ms(T);
    sens2 = sens;
    mot2.write(RC);
    thread_sleep_for(NbS*T);
    mot2.write(0);
    y = y + (-2*sens + 1)*NbS * 1.0 *184 / 1000;
    ystep = ystep + (-2*sens + 1)*NbS;
}

void deplacement_moteurs(int NbS1, int s1, int T1, int enable1, int NbS2, int s2,
int T2, int enable2){
    // fonciton qui deplace les moteurs simultanement
    // ils faut que NbS1*T1 soit a peu pres égale NbS2*T2

    int buffer;

    if(enable1 == 1) mot1.period_ms(T1);
    if(enable2 == 1) mot2.period_ms(T2);
    sens1 = s1;
    sens2 = s2;

    //on calcul le temps de parcours
    if (NbS1*T1 > NbS2*T2) buffer = NbS1*T1;
    else buffer = NbS2*T2;

    // L'un des deux moteurs peut etre desactivé si on ne veut tracer que selon un
axe
    if (enable1 == 1){
        mot1.write(RC);
    }
    if (enable2 == 1){
        mot2.write(RC);
    }

    if (enable1 + enable2 >= 1) thread_sleep_for(buffer);
    mot1.write(0);
    mot2.write(0);

    //On met a jour les coordonnées de la tete
    if (enable1 == 1){
        x = x + (2*s1 -1)*NbS1 *1.0* 184 / 1000;
        xstep = xstep + (2*s1 -1)*NbS1;
    }
    if (enable2 == 1){
        y = y + (-2*s2 + 1)*NbS2 *1.0* 184 / 1000;
        ystep = ystep + (-2*s2+1)*NbS2;
    }


}

void deplacement_angle(double dist, double angle){
    // en cm
    // les angle sont orientés selon le cercle trigo
    int s1, s2;
    int T1, T2;
    double X1, Y1;
    int en1, en2;
```

```
    X1 = dist * cos((angle * 3.1415) / 180);
    Y1 = dist * sin((angle * 3.1415) / 180);

    if (X1 > 0){
    s1 = 1;}
    else{s1 = 0;
    X1 = - X1;
    }
    if (Y1> 0){s2 = 0;}
    else{s2 = 1;
    Y1 = - Y1;
    }

    int NbS1 = dist_to_NbStep(X1);
    int NbS2 = dist_to_NbStep(Y1);


    if(X1 > Y1){T1 = Tmin;
    T2 = (Tmin*X1)/Y1;
    }
    else{T2 = Tmin;
    T1 = (Tmin*Y1)/X1;
    }

    if(T1 > 1000) en1 = 0;
    else en1 = 1;

    if(T2 > 1000) en2 = 0;
    else en2 = 1;

    deplacement_moteurs(NbS1, s1, T1, en1, NbS2, s2, T2, en2);

}

void AtoB(double xA, double yA, double xB, double yB){

    // deplace la tete vers le point A puis trace une ligne de A vers B

    servo_up();
    double depX = x - xA;
    double depY = y - yA;

    int sensX, sensY;

    if(depX > 0) sensX = 0;
    if(depX < 0) sensX = 1;

    if(depY > 0) sensY = 1;
    if(depY < 0) sensY = 0;

    deplacement_mot1(abs(dist_to_NbStep(depX)), sensX, 8);

    deplacement_mot2(abs(dist_to_NbStep(depY)), sensY, 8);

    servo_down();

    double dist = sqrt((xA-xB)*(xA-xB) + (yA-yB)*(yA-yB));
    double angle = atan2(xA-xB,yA-yB);
```

```
    angle = 180 * angle / 3.1415 + 180;

    deplacement_angle(dist, angle);

    servo_up();
}

void deplace_stylo(double xA, double yA){
    // déplace la tête vers le point A, x et y en cm
    servo_up();
    double depX = x - xA;
    double depY = y - yA;

    int sensX, sensY;

    if(depX > 0) sensX = 0;
    if(depX < 0) sensX = 1;

    if(depY > 0) sensY = 1;
    if(depY < 0) sensY = 0;

    deplacement_mot1(abs(dist_to_NbStep(depX)), sensX, 8);
    deplacement_mot2(abs(dist_to_NbStep(depY)), sensY, 8);
}


double read_joy(int *en){
    int joyx;
    int joyy;
    double x;
    double y;
    double a;

        joyx = analog_inx.read_u16();
        joyy = analog_iny.read_u16();

        joyx=joyx-33000;
        joyy=joyy-32800;
        x=joyx*1.0/33000;
        y=(-1)*(joyy*1.0/32800);

        if (y>0){
         a=acos(x);
    }
    else{
        a=-acos(x);
    }
    a = 180 * a / 3.1415;

    if(abs(x) > 0.05 || abs(y) > 0.05){
        *en = 1;
    }

    return a;
}

void trace_plateau(){
    INIT_ORIGIN();
    servo_up();
```

```
    deplacement_angle(100, 0);
    servo_down();
    deplacement_angle(300, 90);
    servo_up();
    deplacement_angle(100, 0);
    servo_down();
    deplacement_angle(300, 270);
    servo_up();
    deplacement_angle(100, 0);
    deplacement_angle(100, 90);
    servo_down();
    deplacement_angle(300, 180);
    servo_up();
    deplacement_angle(100, 90);
    servo_down();
    deplacement_angle(300, 0);
    servo_up();
    INIT_ORIGIN();
}

void initialiseGrille() {
  int i, j;
  for (i=0; i < NB_LIG; i++) {
    for (j=0; j < NB_COL; j++) {
      grille[i][j] = 0;
    }
  }
}

void dessinepion(int col, int ligne){
    int i;
    if (prochainJoueur == 1){
     deplace_stylo(50+100*col,10+100*ligne);
     //dessine un cercle
     for(i=0;i<360;i=i+2){
        deplacement_angle( 1 , i);
         }

      }
      //dessine une croix
    else{
     AtoB(col*100, 100+ligne*100, 100+col*100, ligne*100);
     AtoB(col*100, ligne*100, 100+col*100, 100+100*ligne);

      }


}

void metUnPionSurLaGrille() {
  int ligne, col;
  int i;
  int j;
  x=(xstep*184)/1000;
  y=(ystep*184)/1000;
 if(x<100){col=0;}
 if((x>=100)&(x<200)){col=1;}
 if(x>=200){col=2;}
 if(y<100){ligne=0;}
```

```
 if((y>=100)&(y<200)){ligne=1;}
 if(y>=200){ligne=2;}

      if (grille[ligne][col] == 0)
        {
        grille[ligne][col] = prochainJoueur;
            if (prochainJoueur == 1){
                 dessinepion(col,ligne);
                 prochainJoueur = 2;
           }

          else{
              dessinepion(col,ligne);
             prochainJoueur = 1;
        }
    }
}


Boolean testeFinJeu() {
  int i,j;
  int joueurGagnant; /* pour connaitre quel est le gagnant ie soit CROIX soit ROND
*/
  Boolean estFini = FALSE;

  /* Teste s'il y a un gagnant */
  /* L'algorithme utilise est le plus facile mais n'est pas le plus efficace
     car on n'utilise pas la position du dernier pion ajoute sur la grille. Cette
information
     permettrait de reduire le temps de la recherche.
     De plus, cet algo suppose que la taille de la matrice est de 3 par 3
  */
  /* si la case 1,1 est VIDE, cela signifie que les diagonales, la ligne 1 et la
colonne 1 ne sont
     pas gagnantes
  */
  if (grille[1][1] != 0) {
    if (/* colonne 1 */ ((grille[0][1] == grille[1][1]) && (grille[1][1] ==
grille[2][1])) ||
    /* ligne 1 */ ((grille[1][0] == grille[1][1]) && (grille[1][1] == grille[1][2]))
||
    /* diagonale */ ((grille[0][0] == grille[1][1]) && (grille[1][1] ==
grille[2][2])) ||
    /* autre diag */ ((grille[0][2] == grille[1][1]) && (grille[1][1] ==
grille[2][0]))) {
      joueurGagnant = grille[1][1]; /* ie ROND ou CROIX */
      estFini = TRUE;
    }
  }

  /* si la case 0,0 est vide, cela signifie que la ligne 0 et le colonne 0 ne sont
pas gagnantes */
  if ((!estFini) && (grille[0][0] != 0)) {
    if ( /* ligne 0 */ ((grille[0][0] == grille[0][1]) && (grille[0][1] ==
grille[0][2])) ||
     /* colonne 0*/ ((grille[0][0] == grille[1][0]) && (grille[1][0] ==
grille[2][0]))) {
      joueurGagnant = grille[0][0];
      estFini = TRUE;
```

```
    }
  }

  /* si la case 2,2 est vide, cela signifie que la ligne 2 et la colonne 2 ne sont
gagnantes */
  if ((!estFini) && (grille[2][2] != 0)) {
    if ( /* ligne 2 */ ((grille[2][0] == grille[2][1]) && (grille[2][1] ==
grille[2][2])) ||
     /* colonne 2 */ ((grille[0][2] == grille[1][2]) && (grille[1][2] ==
grille[2][2]))) {
      joueurGagnant = grille[2][2];
      estFini = TRUE;
    }
  }

  if (estFini) {
    pc.printf("Felicitations au joueur ayant les ");
    if (joueurGagnant == 1)
      pc.printf("ronds ");
    else
      pc.printf("croix ");
    pc.printf("qui a gagne.\n");
    return TRUE;
  }

  /* teste si la grille n'est pas pleine */
  for (i=0; i<NB_LIG; i++) {
    for (j=0; j<NB_COL; j++) {
      if (grille[i][j] == 0) /* Au moins une case est vide donc le jeu n'est pas fini
*/
    return FALSE;
    }
  }
  return TRUE;
}

void draw_square(double size, double orientation){

}

int main(){
      //demo
      servo_mot.period_ms(20);
      servo_up();
      INIT_ORIGIN();




      //on se rend au centre du plateau
     /*
      deplace_stylo(10,10);
      servo_down();

      //on dessine un cercle
      int i;
      for(i =0; i <360; i ++){
        deplacement_angle(1, 1.0*i);
      }
```

```
      //on dessine des lignes

      deplacement_angle(50, 180);
      deplacement_angle(20, 45);
      servo_up();
      deplacement_angle(15, 270);
      servo_down();
      deplacement_angle(20, 120);
          */
      deplace_stylo(100,100);
      servo_down();

      //On trace une étoile
      int star_lengh = 100;
    deplacement_angle(star_lengh, 0);
    deplacement_angle(star_lengh, 180 + 36);
    deplacement_angle(star_lengh, 36+36);
    deplacement_angle(star_lengh, 360-54);
    deplacement_angle(star_lengh, 90+54);

      //on utilise le joystick pour tracer
   double angle;
   int bouton_n = 1, bouton_n_1 = 1;

    while(1){
        angle = read_joy(&enable_joy);
        if(enable_joy == 1){
            deplacement_angle(1, angle);
            enable_joy = 0;
        }
        // on regarde le front montant
        bouton_n = bouton_n_1;
        bouton_n_1 = bouton.read();

        if(bouton_n_1 == 0 && bouton_n == 1){
            if(servo == 1) servo_down();
            else servo_up();
        }
        //pc.printf("x = %lf y = %lf ystep = %d ystep = %d", x, y, xstep, ystep);
    }
         /*

    trace_plateau();
   initialiseGrille();

    do {
        do{  angle = read_joy(&enable_joy);
            if(enable_joy == 1){
                 deplacement_angle(1, angle);
               enable_joy = 0;}
            }while(bouton==1);

        metUnPionSurLaGrille();
    }while(!testeFinJeu());

 */
}
```