

Module Microcontrôleurs Option Systèmes Avancés

ERIC ABOUAF

11 octobre 2005

Réalisation d'un bras 3 axes

Année Scolaire 2005-2006

Résumé

Ce document est le rapport d'un mini-projet réalisé pour le module *Microcontrôleurs*. L'objectif de ce module était de découvrir l'architecture d'un microcontrôleur et de se familiariser avec les outils de développement et la programmation du C8051F310 de Silicon Laboratories.

Le projet consiste en un bras 3 axes réalisé à l'aide de servos-moteurs capable d'écrire sur une feuille de papier.



Table des matières

1	Introduction	3
1.1	Objectif	3
1.2	Cahier des charges	3
2	Réalisation du robot	3
2.1	Architecture du système	3
2.2	Construction mécanique	4
2.3	Electronique	4
2.3.1	Le servo-moteur	4
2.3.2	Le circuit	5
3	Programme	6
3.1	Timers	6
3.2	PWM	6
3.3	Port série	7
4	Conclusion	8
A	Code source complet du programme	9
B	Modélisation du robot sous Matlab	14
B.1	Introduction	14
B.2	Calcul numérique	14
B.2.1	Modélisation sous Matlab	14
B.3	Résultats	16
B.4	Traitement	16
B.5	Conclusion sur la méthode	18

1 Introduction

1.1 Objectif

L'objectif consistait à construire un robot à l'aide de 3 servos-moteurs capable d'écrire sur une feuille de papier. Une vidéo de démonstration du résultat obtenu est disponible à l'adresse suivante :

<http://www.neyric.com/comp/MIC/bras3axes.avi>

1.2 Cahier des charges

Le cahier des charges fixé est très succinct :

- Utilisation de 3 servos-moteurs comme actionneurs
- Utilisation du microcontrôleur C8051F310 de Silicon Laboratories
- Envoi des commandes au robot par port Série
- L'alimentation des servos-moteurs sera effectuée par une alimentation de laboratoire
- Aucune obligation n'est donnée sur la manière d'écrire (on ira au plus simple)

2 Réalisation du robot

2.1 Architecture du système

L'architecture choisie consiste à utiliser le microcontrôleur pour générer les signaux de commande des trois servo-moteurs (PWM). L'ordinateur dialoguera directement avec le robot par le port série (RS-232) du microcontrôleur et enfin les moteurs seront alimentés par une alimentation de laboratoire.

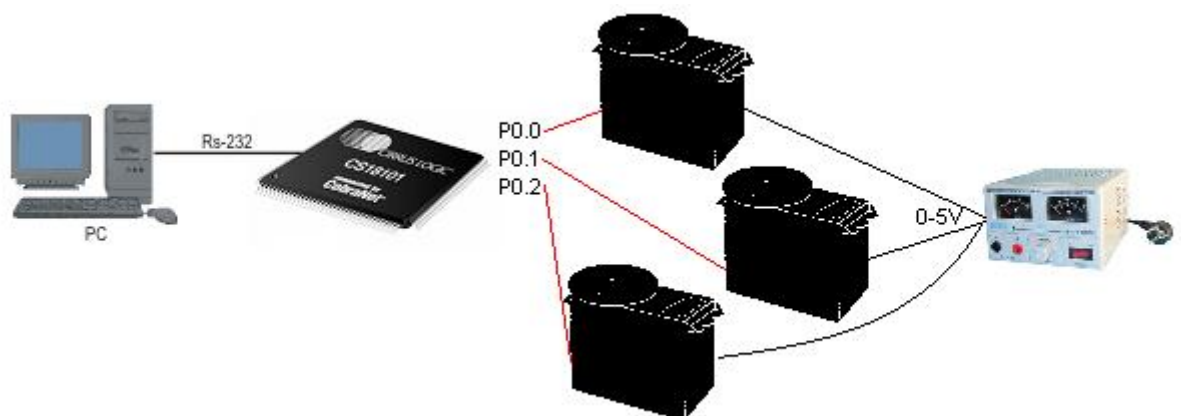


Schéma de l'architecture

2.2 Construction mécanique

La base de ce robot existait déjà. Il a fallu rajouter le troisième servo-moteurs ainsi que le bras le reliant au second puis fixer. Ce bras fut réalisé à l'aide d'accessoires pour servos-moteurs et de quelques petites pièces Mecano :



2.3 Electronique

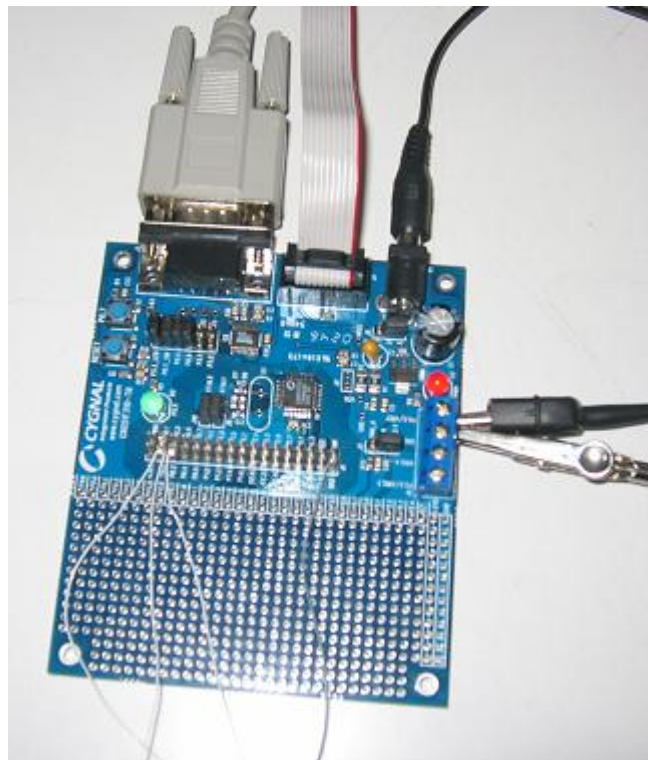
2.3.1 Le servo-moteur

Un servomoteur est constitué d'une partie mécanique (un moteur et un réducteur) et d'une partie commande (un potentiomètre interne variant avec l'angle de l'axe et un asservissement de position) intégrées dans un même boîtier. En fait, le servomoteur est commandé par des impulsions de largeur comprise entre 1ms et 2ms à une fréquence d'environ 50 Hz. La position angulaire du pignon est proportionnelle à la largeur de l'impulsion. Le servomoteur restera dans cette position tant qu'il recevra la même impulsion de commande. Et lorsque le servomoteur ne reçoit pas d'impulsions de commande, le pignon reste en position même contre un couple résistant mais qui doit rester modéré. Si on exerce une force importante, le pignon tourne jusqu'à une butée qui existe dans les 2 sens. Le faible courant sur la commande (maximum 50uA) autorise une connexion directe d'un port de sortie du microcontrôleur au servomoteur.



2.3.2 Le circuit

Le circuit électronique étant très simple, le projet a été wrappé directement sur la carte de développement. Les trois fils de commande des servos sont directement connectés aux ports P0.0 P0.1 et P0.2 du microcontrôleur. Un raccord entre les masses de la platine de développement et l'alimentation des servos doit être présente pour le bon fonctionnement du circuit.



3 Programme

Le programme de commande effectue les opérations suivantes :

- Initialisation du Crossbar
- Initialisation des Timers
- Initialisation du PCA
- Initialisation des 3 PWM

```
- Boucle Inifinie :  
SI( lettre sur port série )  
Trace lettre  
FIN SI
```

Pour initialiser le crossbar, nous avons besoin d'attribuer les ports suivants :
Port série, P0.0 = PWM0, P0.1 = PWM1 , P0.2 = PWM2 :

```
// Crossbar  
XBR0      = 0x01; // Enable crossbar  
XBR1      = 0x43; // UART+CEX0+CEX1+CEX2
```

3.1 Timers

Le programme utilise 2 timers sur 8 bits en fonctionnement "autoreload". Le timer 0 servira au PCA pour la génération des commandes PWM. Il est calculé pour déborder toutes les 20ms (50 Hz). Le timer 1 basé sur SYSCLK/4 servira au port série. Voici le code pour initialiser ces deux timers :

```
// Timers :  
TCON      = 0x50; // Enable Timer0 et Timer1  
TMOD      = 0x22; // T0 et T1 en 8bits/autoreload  
CKCON     = 0x05; // T0 en sysclk  
           // et T1 en sysclk/4  
  
// Timer 0 (PCA)  
TH0       = 0xFF; // 20 ms
```

3.2 PWM

Pour les signaux de commande PWM, il faut d'abord configurer les ports en Pull-up sinon la puissance débité par le microcontrôleur n'est pas suffisante. Les

3 compteurs du PWM sont remis à 0 sur un débordement du Timer 0 (timer0 overflow). Les valeurs initiales de ces compteurs sont déterminées de manière empirique pour obtenir une position initiale centrée.

```
// Ports  
POMDOUT   = 0x07; // P0.0, P0.1, P0.2 => Pull-up  
  
// PCA
```

```
PCA0CN    = 0x40; // pca enabled

PCA0CPM0  = 0xC2; //
PCA0CPM1  = 0xC2; // PWM et Comparator function for module=0,1,2
PCA0CPM2  = 0xC2; //

PCA0MD    and= ~0x40; // {
PCA0MD    = 0x04; // PCA counter = Timer 0 overflow
PCA0CPL4  = 0x00; //
PCA0MD    |= 0x40; //  }

// Valeurs initiale des compteurs
PCA0CPL0  = 0x00;
PCA0CPH0  = 0xEF;

PCA0CPL1  = 0x00;
PCA0CPH1  = 0xEF;

PCA0CPL2  = 0x00;
PCA0CPH2  = 0xEF;
```

3.3 Port série

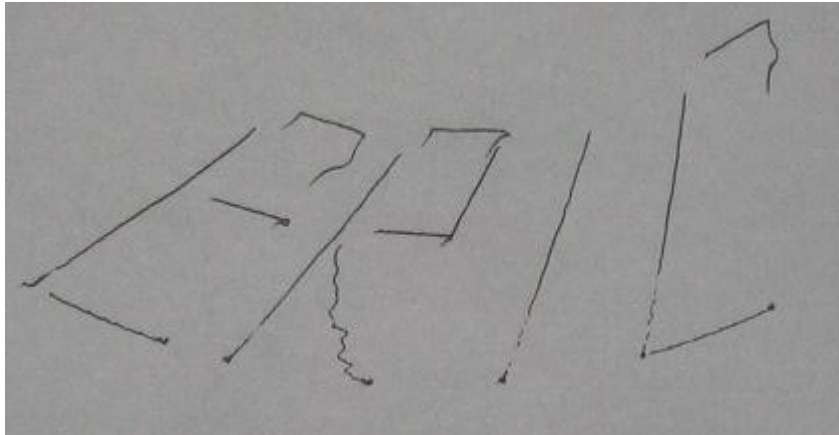
Le port série est géré par le périphérique "UART". Il se synchronise sur un overflow du timer1. Ce timer est calculé pour faire fonctionner le port RS-232 à 4900 bauds.

```
// UART
//SCON0    = 0x0C; // On active l'UART
SCON0     = 0x10;

// Timer 1 (UART)
TH1       = 0xB1;
```

4 Conclusion

Finalement, les positions ont été rentrées numériquement dans le programme et plusieurs lettres ont été préprogrammées à l'aide de ces positions. Le robot était finalement capable d'écrire des lettres telles que A,C,E,G,I,L,N et R. Voici un exemple de ce qui a été tracé :



Une vidéo de cette démonstration peut être trouvée à l'adresse suivante :
<http://www.neyric.com/comp/MIC/bras3axes.avi>
Bien évidemment, un bras comme celui-ci n'est pas très adapté au tracage, un système en coordonnées cartésiennes aurait certainement été plus simple.

Annexes

A Code source complet du programme

```

/*
    Contrôle de 3 servo-moteurs par PWM
*/

//-----
// Includes
//-----

#include <c8051f310.h>           // SFR declarations
#include <intrins.h>
#include <stdio.h>

void init(void);
void SendChar(char c);
int SendString(char *buffer);
char getch(void);

int code tableau[50][2] = {
{-3862, -5187 },{-3872, -5157 },{-3882, -5147 },{-3892, -5107 },
{-3912, -5077 },{-3922, -5047 },{-3942, -5007 },{-3962, -4977 },
{-3972, -4947 },{-3982, -4927 },{-3992, -4907 },{-4002, -4887 },
{-4012, -4857 },{-4032, -4817 },{-4032, -4817 },{-4052, -4757 },
{-4062, -4737 },{-4082, -4677 },{-4092, -4637 },{-4112, -4597 },
{-4122, -4597 },{-4132, -4547 },{-4142, -4497 },{-4152, -4457 },
{-4162, -4407 },{-4162, -4407 },{-4172, -4367 },{-4182, -4327 },
{-4182, -4267 },{-4182, -4237 },{-4192, -4207 },{-4192, -4177 },
{-4202, -4167 },{-4212, -4147 },{-4222, -4077 },{-4222, -4047 },
{-4222, -4027 },{-4222, -3997 },{-4222, -3957 },{-4222, -3937 },
{-4222, -3917 },{-4222, -3897 },{-4222, -3877 },{-4222, -3857 },
{-4222, -3847 },{-4222, -3827 },{-4222, -3807 },{-4202, -3787 },
{-4202, -3767 },{-4202, -3747 } };

//-----
// Init
//-----

void init(void)
{
    // Crossbar
    XBR0 = 0x01; // Enable crossbar
    XBR1 = 0x43; // UART+CEX0+CEX1+CEX2

    // Ports
    P0MDOUT = 0x07; // P0.0, P0.1, P0.2 => Pull-up

    // PCA
    PCA0CN = 0x40; // pca enabled

    PCA0CPM0 = 0xC2; //
    PCA0CPM1 = 0xC2; // PWM & Comparator function for module=0,1,2
    PCA0CPM2 = 0xC2; //

    PCA0MD &= ~0x40; // {
    PCA0MD = 0x04; // PCA counter = Timer 0 overflow
    PCA0CPL4 = 0x00; //
    PCA0MD |= 0x40; // }

    // Valeurs des compteurs
    PCA0CPL0 = 0x00;
    PCA0CPH0 = 0xEF;

    PCA0CPL1 = 0x00;
    PCA0CPH1 = 0xEF;

    PCA0CPL2 = 0x00;
    PCA0CPH2 = 0xEF;

    // On désactive le watchdog
    PCA0MD &= ~0x40;

    // Timers :
    TCON = 0x50; // Enable Timer0 & Timer1
    TMOD = 0x22; // T0 & T1 en 8 bits/autoreload
    CKCON = 0x05; // T0 en sysclk // et T1 en sysclk/4

    // Timer 0 (PCA)
    TH0 = 0xFF; // 20 ms

    // UART
    //SCON0 = 0x0C; // On active l'UART
    SCON0 = 0x10;

    // Timer 1 (UART)
    TH1 = 0xB1;
}

```

```

//-----
// UART
//-----

// Envoie un char sur le port série et attend...
void SendChar(char c)
{
    SBUF0 = c;

    while( TI0 != 1 ) {} ;
    TI0 = 0;
}

// Envoie une chaine de char sur le port série
int SendString(char *buffer)
{
    int i = 0;

    while(buffer[i] != 0)
    {
        SendChar(buffer[i]);
        i++;
    }

    return i;
}

// Attends l'appui sur une touche :
char getch(void)
{
    char c;

    while( RI0 != 1 );

    // On lit la lettre
    c = SBUF0;
    RI0 = 0;

    return c;
}

//-----
// Commande du bras
//-----

// Valeurs des compteurs PWM
int cp[3];
int cp_min[3];
int cp_max[3];

void update_position()
{
    PCA0CPL0 = cp[0]&0x00FF;
    PCA0CPH0 = (cp[0]&0xFF00)/16/16;

    PCA0CPL1 = cp[1]&0x00FF;
    PCA0CPH1 = (cp[1]&0xFF00)/16/16;

    PCA0CPL2 = cp[2]&0x00FF;
    PCA0CPH2 = (cp[2]&0xFF00)/16/16;
}

// Tourne à l'angle donné en paramètre
void goto_angle( unsigned char angle )
{
    long temp = 58644+31*angle;
    cp[0] = (int) temp;
}

// Tourne à l'angle donné en paramètre
void set_theta1( int angle )
{
    long temp = 61064+29*angle;
    cp[1] = (int) temp;
}

// Tourne à l'angle donné en paramètre
void set_theta2( int angle )
{
    long temp = 59794-23*angle;
    cp[2] = (int) temp;
}

// Initialise les positions initiales
void init_position()
{
    goto_angle(90);
    set_theta1(0);
    set_theta2(-45);

    cp_max[0] = 0xFF00;
    cp_max[1] = 0xFF00;
    cp_max[2] = 0xFF00;

    cp_min[0] = 0xE000;
    cp_min[1] = 0xE000;
    cp_min[2] = 0xE000;
}

```

```

void wait()
{
    int i = 0;
    while( i != 0x1000 )
        i++;
}

void trace(int r1, int angle1, int r2, int angle2 )
{
    int i;
    int r_step;
    int angle_step;

    // Position de départ
    init_position();
    goto_angle(angle1);
    update_position();
    wait();
    cp[2] = tableau[r1][1] ;
    update_position();
    wait();
    cp[1] = tableau[r1][0];
    update_position();
    wait();

    if( r1 < r2 )
        r_step = 1;
    else if( r1 > r2 )
        r_step = -1;
    else
        r_step = 0;

    if( angle1 < angle2 )
        angle_step = 1;
    else if( angle1 > angle2 )
        angle_step = -1;
    else
        angle_step = 0;

    // Boucle 50 pas
    for( i = 0 ; i < 50 ; i++ )
    {
        if( r_step == 1 )
        {
            cp[1] = tableau[r1+ (r2-r1)*i/50 ][0];
            cp[2] = tableau[r1+ (r2-r1)*i/50 ][1];
        }
        else if( r_step == -1 )
        {
            cp[1] = tableau[r1+ (r1-r2)*i/50 ][0];
            cp[2] = tableau[r1+ (r1-r2)*i/50 ][1];
        }

        if( angle_step == 1 )
            goto_angle( angle1+ (angle2-angle1)*i/50 );
        else if( angle_step == -1 )
            goto_angle( angle1+ (angle1-angle2)*i/50 );

        update_position();
        wait();
    }

    // On revient en position initiale
    init_position();
    update_position();
    wait();
    wait();
    wait();
    wait();
}

// Toutes les lettres sont entre angle et angle-10
// et rayon 10 à 40
void print(char lettre, int angle)
{
    switch(lettre)
    {
        case 'A':    trace( 10, angle, 40, angle);
                    trace( 40, angle-10, 40, angle);
                    trace( 10, angle-10, 40, angle-10);
                    trace( 30, angle-10, 30, angle);
                    break;

        case 'N':    trace( 10, angle, 40, angle);
                    trace( 10, angle-10, 40, angle);
                    trace( 10, angle-10, 40, angle-10);
                    break;

        case 'E':    trace( 10, angle, 40, angle);
                    trace( 40, angle-10, 40, angle);
                    trace( 30, angle-10, 30, angle);
                    trace( 10, angle-10, 10, angle);
                    break;
    }
}

```

```

        case 'L':    trace( 10, angle, 40, angle);
                    trace( 10, angle-10, 10, angle);
                    break;

        case 'C':    trace( 10, angle, 40, angle);
                    trace( 40, angle-10, 40, angle);
                    trace( 10, angle-10, 10, angle);
                    break;

        case 'G':    trace( 10, angle, 40, angle);
                    trace( 40, angle-10, 40, angle);
                    trace( 10, angle-10, 10, angle);

                    trace( 10, angle-10, 30, angle-10);
                    trace( 30, angle-10, 30, angle);
                    break;

        case 'I':    trace( 10, angle-5, 40, angle-5);
                    break;

        case 'R':    trace( 10, angle, 40, angle);
                    trace( 40, angle-10, 40, angle);
                    trace( 30, angle-10, 40, angle-10);
                    trace( 30, angle-10, 30, angle);
                    trace( 10, angle-10, 30, angle);

                    break;

        default: break;
    }
}

//-----
// MAIN Routine
//-----

char code texte [] = "Bienvenue_!\r\n\r\nPour_commander_le_bras_\r\n_d:_droite_1\r\n_m:_gauche2\r\n";

void main (void)
{
    char buffer [30];
    int pos;
    int angle;

    // Variables temporaires
    char c;

    // Variables locales
    pos = 0;
    angle = 90;

    // Initialisation
    init ();

    // Initialisation des positions
    init_position ();
    update_position ();

    // Message d'accueil
    SendString(texte);

    // Boucle principale du programme
    while(1)
    {
        // On lit la lettre
        c = getch();

        switch(c)
        {
            case '_':

                sprintf(buffer, "{_%d,_%d_}\r\n", cp[1], cp[2]);
                SendString(buffer);

                /* tableau[pos][0] = theta1;
                tableau[pos][1] = theta2;
                pos++;*/

                break;

            case 'p':

                sprintf(buffer, "pos:_%d\r\n", pos);
                SendString(buffer);
                break;

            case 't':

                if( pos < 49)
                pos++;
                cp[1] = tableau[pos][0];
                cp[2] = tableau[pos][1] ;
                break;

            case 'g':

                if( pos > 0)
                pos--;
                cp[1] = tableau[pos][0];
                cp[2] = tableau[pos][1] ;

```

```
                                break;
    case 'f':
        angle++;
        goto_angle(angle);
        break;
    case 'h':
        angle--;
        goto_angle(angle);
        break;
    case 'm':
        print('E', 140);
        print('R', 125);
        print('I', 110);
        print('C', 95);
        break;
    case 'q':
        if( cp[1] > cp_min[1])
            cp[1] -= 10;
        break;
    case 's':
        if( cp[1] < cp_max[1])
            cp[1] += 10;
        break;
    case 'w':
        if( cp[2] > cp_min[2])
            cp[2] -= 10;
        break;
    case 'x':
        if( cp[2] < cp_max[2])
            cp[2] += 10;
        break;
    default: break;
    }
    update_position();
} // Fin du while(1);
```

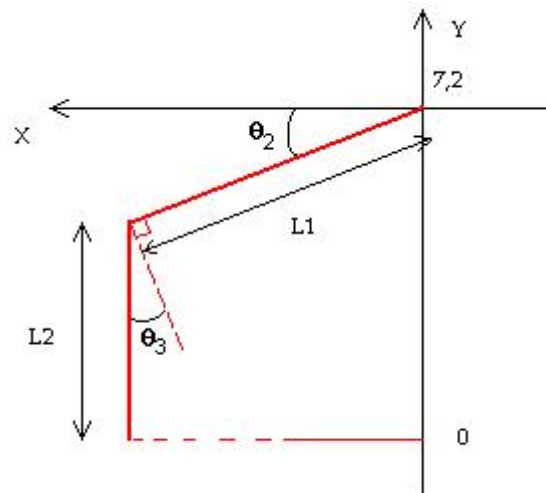
B Modélisation du robot sous Matlab

B.1 Introduction

Pour pouvoir écrire à l'aide de ce robot, il faut conserver la pointe du stylo sur la feuille de papier. La partie mécanique du robot incite à travailler en coordonnées polaires. On se retrouve alors avec une équation non linéaire à 2 inconnues (θ_2 et θ_3).

B.2 Calcul numérique

La solution exploitée consiste à faire varier les angles θ_2 et θ_3 de manière exhaustive (avec un pas assez fin) et de conserver les positions qui conduisent à $Y_{\text{stylo}}=0$.



On obtient au final un tableau contenant un nombre de position déterminé donnant θ_2 et θ_3 en fonction du rayon r .

B.2.1 Modélisation sous Matlab

% Fonction N

```
function point = N(theta1,L1)
```

```
point = [ L1*cos(theta1*2*pi/360) , L1*sin(theta1*2*pi/360) ];
```

```
return;
```

% Fonction M

```
function point = M(theta1,theta2,L2)
```

```
point = [ L2*(cos(theta2*2*pi/360)*cos(theta1*2*pi/360)-sin(theta2*2*pi/360)*sin(theta1*2*pi/360)
```

```
return;
```

```
%=====
```

```
% Calcul des positions Y = 0
```

```
%=====
```

```
clear;
```

```
clf;

% Dimensions du bras
L1 = 6.2;
L2 = 6.3;

% Point origine
org = [ 0 , 7.2 ];

% angle theta1_minimum ( environ -8° )
theta1_max = -asin( (7.2-L2)/L1 )*360/(2*pi);
theta1_min = -20;

n_steps = 30;

% initialisation du tableau ( r, t1, t2 )
tableau = zeros( n_steps*2, 3);

% calcul n_steps pas
for i = 1 : n_steps

    theta1 = theta1_min+ (theta1_max-theta1_min)/n_steps*i;

    % on fait parcourir theta2 de -90 à +90
    for theta2 = 0:-1:-180

        % Point N
        N1 = N(theta1, L1) + org;

        % Point M
        M1 = M(theta1, theta2, L2) + N1;

        % si l'ordonnée est à 0 (ou presque) on sauve le point
        if M1(2) > -0.05
            if M1(2) < 0.05

                if tableau(n_steps+i,1) == 0
                    tableau(n_steps+i,1) = M1(1);
                    tableau(n_steps+i,2) = theta1;
                    tableau(n_steps+i,3) = theta2;
                else
                    tableau(i,1) = M1(1);
                    tableau(i,2) = theta1;
                    tableau(i,3) = theta2;
                end
            end
        end
    end
end
end
end
```

```

%=====
% Tri du tableau :
%=====

for i = 1 : 2*n_steps

    % On compare l'élément i à tous les j suivants
    min = tableau(i,1);
    index = i;

    for j = 1 : 2*n_steps-i
        if tableau(i+j,1) < min
            index = i+j;
            min = tableau(i+j,1);
        end
    end

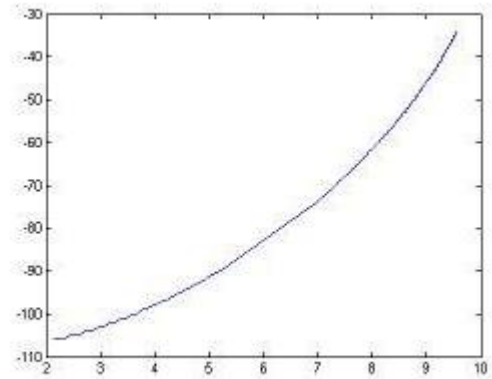
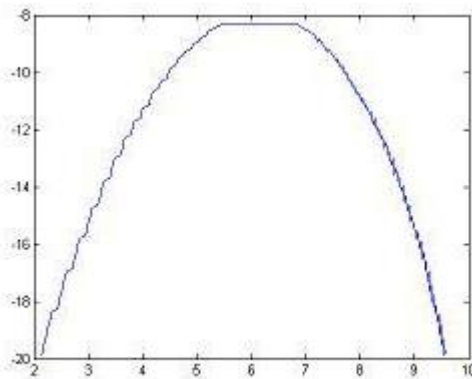
    if index ~= i
        temp = tableau(i,1:3);
        tableau(i,1:3) = tableau(index,1:3);
        tableau(index,1:3) = temp(1:3);
    end

end
end

```

B.3 Résultats

Après ce calcul, on récupère donc ces courbes qui représentent theta2 et theta3 en fonction du rayon :



B.4 Traitement

A partir de ces valeurs discrètes, on réalise une interpolation polynômiale de degré 3 :

```
% Interpolation de theta1

Mat_1 = zeros(4,4);
y = zeros(4,1);

Mat_1(1,1) = tableau(1,1)^3;
Mat_1(1,2) = tableau(1,1)^2;
Mat_1(1,3) = tableau(1,1);
Mat_1(1,4) = 1;
y(1,1) = tableau(1,2);

Mat_1(2,1) = tableau(n_steps,1)^3;
Mat_1(2,2) = tableau(n_steps,1)^2;
Mat_1(2,3) = tableau(n_steps,1);
Mat_1(2,4) = 1;
y(2,1) = tableau(n_steps,2);

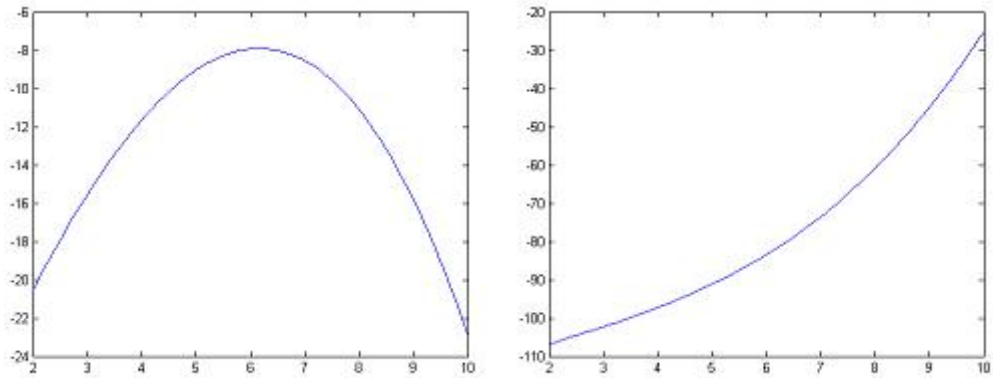
Mat_1(3,1) = tableau(n_steps+1,1)^3;
Mat_1(3,2) = tableau(n_steps+1,1)^2;
Mat_1(3,3) = tableau(n_steps+1,1);
Mat_1(3,4) = 1;
y(3,1) = tableau(n_steps+1,2);

Mat_1(4,1) = tableau(2*n_steps,1)^3;
Mat_1(4,2) = tableau(2*n_steps,1)^2;
Mat_1(4,3) = tableau(2*n_steps,1);
Mat_1(4,4) = 1;
y(4,1) = tableau(n_steps*2,2);

coeffs_1 = Mat_1\y;

r=2:0.1:10;
plot(r, coeffs_1(1,1)*r.^3+coeffs_1(2,1)*r.^2+coeffs_1(3,1)*r+coeffs_1(4,1) )
```

On obtient alors deux polynômes de degré 3 dont l'écart au courbes précédentes est très faible :



B.5 Conclusion sur la méthode

L'implémentation de ces courbes dans le programme a posé quelques problèmes au départ car il était impossible de faire du calcul en nombres flottants. En multipliant par 1000 on se ramène à l'utilisation de long avec une précision suffisante.

Malheureusement, la modélisation effectuée était trop éloignée du robot réel. De ce fait, les angles calculés ne correspondaient plus à $Y_{stylo} = 0$ sur le robot réel. Il aurait fallu choisir un modèle un peu plus complexe, mais pour raison de temps, cette technique a été abandonnée.