ME 446
Lab 1 Report

Diyu Yang (dyang37)
Dongbo Wang (dwang49)
Instructor: Dan Block
Section: Wed. 2-5pm

1. DH Frames of CRS Robot Arm

Before anything could be done, we need to assign the DH Frames of the CRS Robot Arm. All three joints are revolute joints, so the z-axis must be aligned with the revolution actuation direction. We only need to determine the positive direction of the z-axis, to coincide with the motor angle positive direction. To do this, we manipulate the CRS Robot Arm and look at the motor angle values displayed on the Tera Term console. We then choose the revolution direction that changes the motor angles positively as our z-axis positive direction, using the right-hand rule. The DH frames assignment of the CRS robot arm are shown in **Figure 1**. The good thing about is assignment is that the torque direction of the joint motors coincides with the positive direction of the joint z-axis given by the right-hand rule. The positive direction of the joint motor torque is in the same direction that changes the motor angles positively, as shown in **Figure 1.**

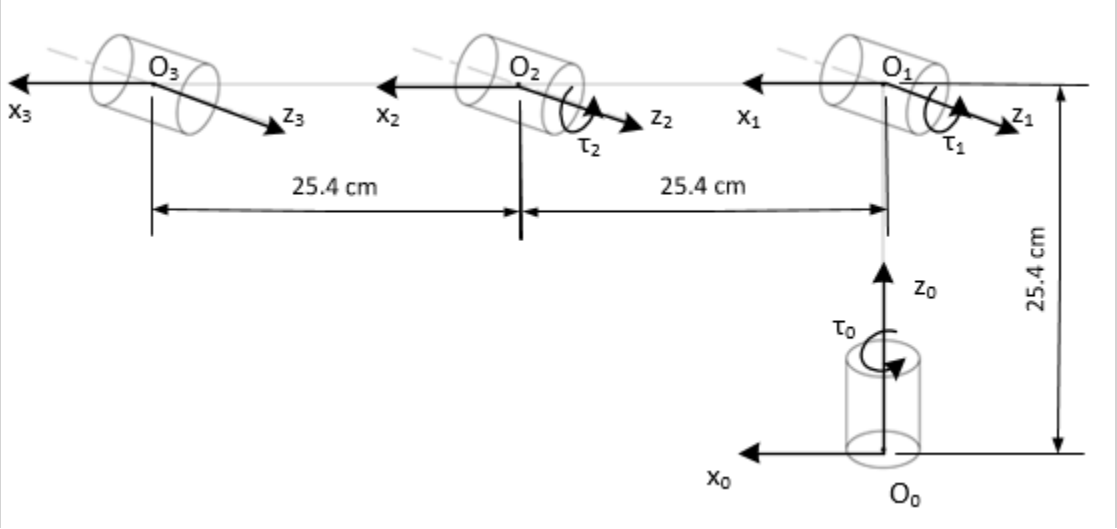

Figure 1. DH Frames Assignment

We have totally 3 joints, namely joint 1 to joint 3 and all of them are revolute joint, therefore the only joint variables are θ_1 to θ_3, one for each joint. With the DH Frame assignment, the given measurements and the parameters needed for each frame, we can construct the full DH Frame table as shown in **Table 1**. To find these variables, we followed the definition given by the DH Frame, where '$\mathbf{a_i}$' is the distance from $\mathbf{z_{i-1}}$ to $\mathbf{z_i}$ along $\mathbf{x_i}$; '$\boldsymbol{\alpha_i}$' is the angle from $\mathbf{z_{i-1}}$ to $\mathbf{z_i}$ with respect to $\mathbf{x_i}$; '$\mathbf{d_i}$' is the distance from $\mathbf{o_{i-1}}$ to $\mathbf{o_i}$ along $\mathbf{z_{i-1}}$; '$\boldsymbol{\theta_i}$' is the angle from $\mathbf{x_{i-1}}$ to $\mathbf{x_i}$ with respect to $\mathbf{z_{i-1}}$. The angle information could be obtained directly from the DH Frame assignment the length information is obtained from the specification sheet of the robot

arm. As mentioned before, θ_1 to θ_3 are the only joint variables, other values are constant measured in either centimeter (cm) for length or radiance (rad) for angle.

Link	a (cm)	α (rad)	d (cm)	θ (rad)
#1	0	-π/2	25.4	θ_1
#2	25.4	0	0	θ_2
#3	25.4	0	0	θ_3

Table 1. DH Frames Parameters

With is DH Frame Parameters Table, we are ready to compute the forward kinematics equations in Mathematica.

2. Forward Kinematics with Mathematica

In this part, we will be using the Robotica library in Mathematica to generate the forward kinematics equations. The procedures are as follows:

a) Invoke Robotica

First, we need to invoke the Mathematica build-in library: Robotica.

```
<< robotica.m
Robotica version 3.60.
Copyright 1993 Board of Trustees, University of Illinois
All rights reserved.
Email questions, comments, or concerns to m-spong@uiuc.edu.
SetDelayed::write: Tag TrigFactor in TrigFactor[e_] is Protected. >>
```

Figure 2. Invoke Robotica

b) Load DH Parameters Data File

The Robotica requires the input data in a '.txt' file using the following format. And the 'DataFile[]' command will load the data into Mathematica.

```
DOF=3
(Robotica requires a line between DOF and joint1)
joint1 = revolute
a1 = 0
alpha1 = Pi/2
d1 = 25.4
theta1 = q1
joint2 = revolute
a2 = 25.4
alpha2 = 0
d2 = 0
theta2 = q2
joint3 = revolute
a3 = 25.4
alpha3 = 0
d3 = 0
theta3 = q3
```

Figure 3. dhframe.txt

```
DataFile["C:\\dyang37dwang49\\dhframe.txt"]

No dynamics data found.


Kinematics Input Data

---------------------


 Joint   Type        a       alpha     d       theta

 1       revolute    0       -Pi/2.    25.4    q1
 2       revolute    25.4    0         0       q2
 3       revolute    25.4    0         0       q3
```

Figure 4. Load Input Files

c) Generate Transformation Matrices

The function used in Robotica to generate the Forward Kinematics Equations is 'FKin[]'.

```
FKin[]

Jacobian  J(6x3)

Jacobian Formed :


T[2,3]
T[1,3]
T[1,2]
T[0,3]
T[0,2]
T[0,1]
T[0,0]
```

Figure 5. Generate Transformation Matrices

d) Get the Simplified Equations

```
In[10]:= MatrixForm[Simplify[T[0, 3]]]

Out[10]//MatrixForm=
(Cos[q1] Cos[q2 + q3]  -Cos[q1] Sin[q2 + q3]  Sin[q1]  Cos[q1] (Cos[q2] (25.4 + 25.4 Cos[q3]) - 25.4 Sin[q2] Sin[q3]))
(Cos[q2 + q3] Sin[q1]  -Sin[q1] Sin[q2 + q3]  -Cos[q1]  Sin[q1] (Cos[q2] (25.4 + 25.4 Cos[q3]) - 25.4 Sin[q2] Sin[q3]))
(    Sin[q2 + q3]           Cos[q2 + q3]          0.                25.4 + 25.4 Sin[q2] + 25.4 Sin[q2 + q3]          )
(         0                      0                0.                                1.                               )
```

Figure 6. Get Simplified Equations

The final forward Kinematics Equations are:

$$\begin{cases} x = 25.4 * \cos(\theta_1) * [\cos(\theta_2) + cos(\theta_2 + \theta_3)] \\ y = 25.4 * \sin(\theta_1) * [\cos(\theta_2) + cos(\theta_2 + \theta_3)] \\ z = 25.4 * [1 + \sin(\theta_2) + sin(\theta_2 + \theta_3)] \end{cases}$$

The implementation that we do in the C file follows this formula strictly.

3. Finding Relationships between motor angles and DH angles

We first find the relationship between θ_1 and motor angles by noticing that θ_1 is always equal to θ_{1motor}. We then find out the relationship between θ_2 and motor angles by noticing a constant $\frac{\pi}{2}$ offset between θ_2 and θ_{2motor}. For the relationship between θ_3 and motor angles, we noticed that the value of θ_3 is independent with θ_{1motor}, since changing only θ_{1motor} with other two motor angles fixed will have no effect on the value of θ_3. So we set up the equation $\theta_3 = c_2\theta_{2motor} + c_3\theta_{3motor} + \alpha$, and solved the equation by taking three sets of data:

θ_3(degree)	θ_{2motor}(degree)	θ_{3motor}(degree)
128.92	-24.62	14.29
114.36	13.78	38.14
92.14	32.18	34.32

Table 2. Motor Angle Data

Therefore, by the three equations above we solved:

$$c_2 = 1$$

$$c_3 = -1$$

$$\alpha = \frac{\pi}{2}$$

Answer:

$$\theta_1 = \theta_{1motor}$$

$$\theta_2 = \theta_{2motor} - \frac{\pi}{2}$$

$$\theta_3 = \theta_{3motor} - \theta_{2motor} + \frac{\pi}{2}$$

4. Inverse Kinematics

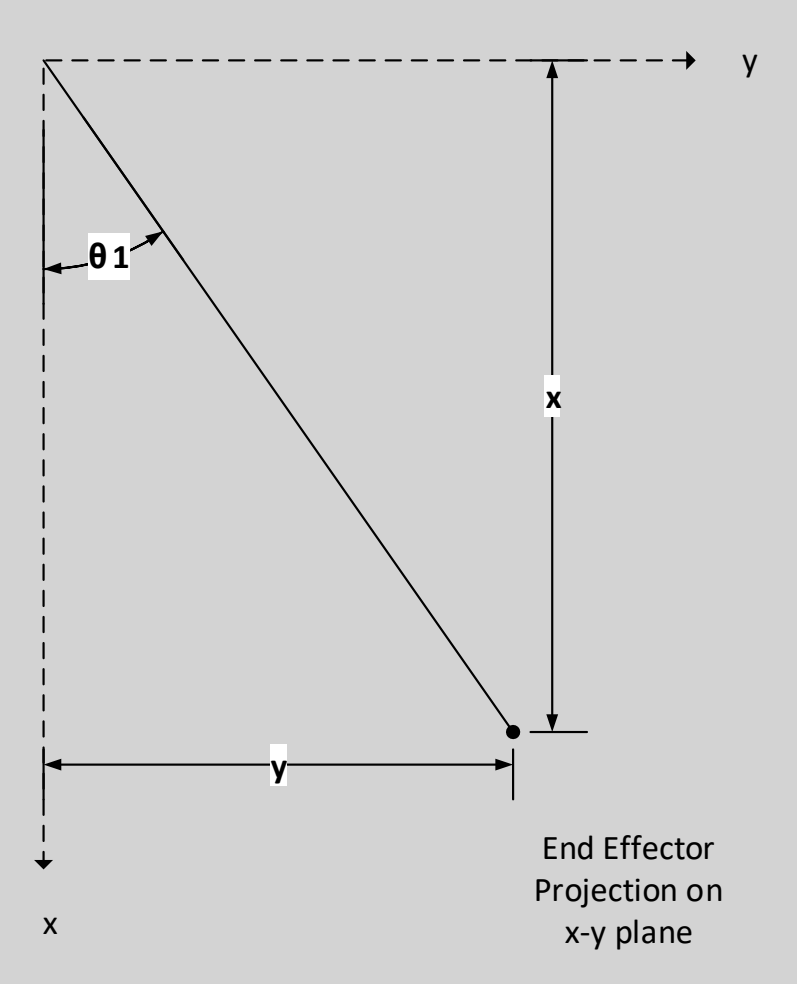

Figure 7. Invers Kinematics of θ_1

From **Figure 7,** we can see that θ_1 can be easily calculated using the triangle property of the following:

$$\tan\theta_1 = \frac{y}{x}$$

Therefore θ_1 can be expressed using the following equation:

$$\theta_1 = atan2(y, x)$$

Where x, y, z are the position of end effector in world coordinate frame.

For θ_2 and θ_3, we can solve the inverse kinematics using the following geometric approach:

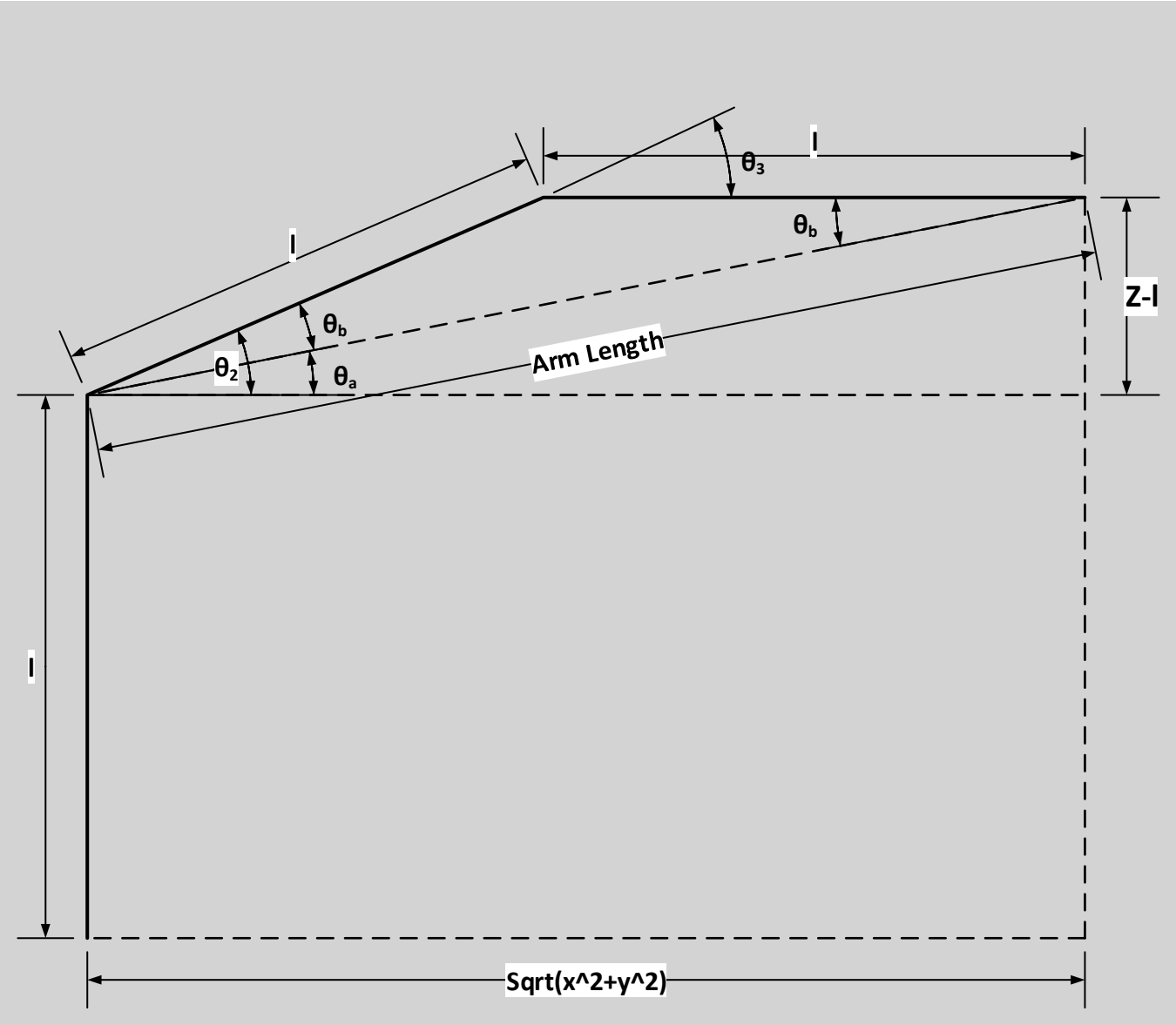

Figure 8. Inverse Kinematics for θ_2 and θ_3

l = 25.4cm, l_{arm}= Arm Length

As indicated by Figure above, we can express θ_2 as: $-\theta_2=\theta_a + \theta_b$, where θ_a can be solved as follows:

$$\tan\theta_a = \frac{z-l}{\sqrt{x^2+y^2}}$$

Therefore θ_a can be expressed as follows:

$$\theta_a = atan2(z - l, \sqrt{x^2 + y^2}) \quad (1)$$

As shown in **Figure 9**, l_{arm} can be expressed by the following equation:

$$l_{arm} = \sqrt{x^2 + y^2 + (z - l)^2}$$

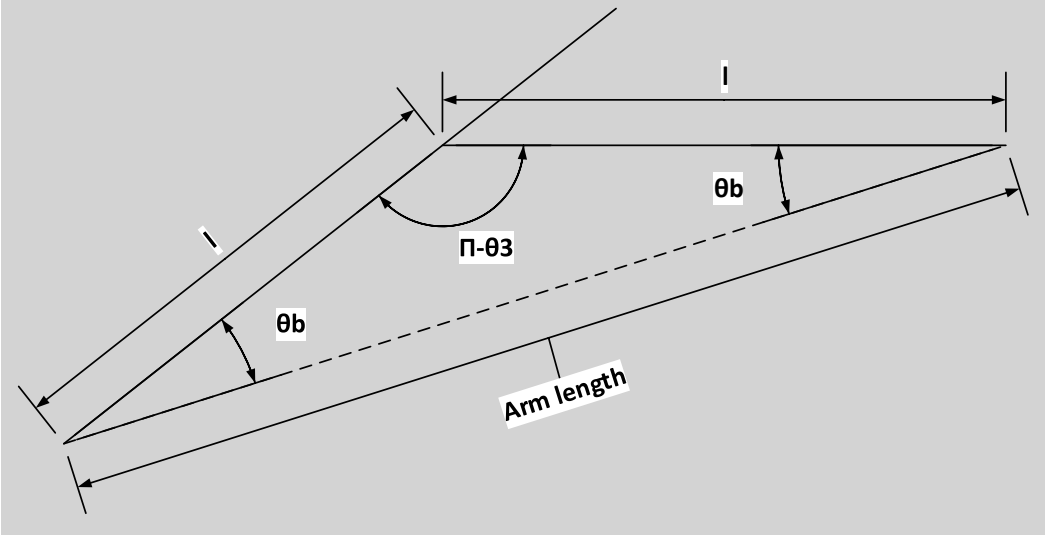

Figure 9. Inverse Kinematics for θ_3

Using Law of Cosines in the above triangle, l_{arm} can also be expressed as follows:

$$l_{arm}{}^2 = 2l^2 + 2l^2 \cos(\pi - \theta_3) \quad (2)$$

We also noticed from **Figure 9**, that $(\pi - \theta_3) + 2\theta_b = \pi$, therefore

$$2\theta_b = \theta_3 \quad (3)$$

Using equations (2) and (3), we express θ_b as follows:

$$\theta_b = \cos^{-1}(\frac{l_{arm}}{2l})$$

As a result, we express $\theta_2 = -\theta_b - \theta_a$ as follows:

$$\theta_2 = -atan2\left(z - l, \sqrt{x^2 + y^2}\right) - \cos^{-1}\left(\frac{l_{arm}}{2l}\right)$$

Where expression for θ_a is given by equation (1).

Using equation (3), we can finally express θ_3 as follows:

$$\theta_3 = 2\theta_b = 2\cos^{-1}\left(\frac{l_{arm}}{2l}\right)$$

Answer:

$$\tan\theta_1 = \frac{y}{x}$$

$$\theta_2 = -atan2\left(z - l, \sqrt{x^2 + y^2}\right) - \cos^{-1}\left(\frac{l_{arm}}{2l}\right)$$

$$\theta_3 = 2\theta_b = 2\cos^{-1}\left(\frac{l_{arm}}{2l}\right)$$

Where $l_{arm} = \sqrt{x^2 + y^2 + (z - l)^2}$ **,** $l = 25.4cm$

We successfully verified our inverse kinematics solution by printing our solution for DH angles onto Tera Term Terminal. The inverse kinematics solution agrees with the DH angle calculated using motor angles.

5. Conclusion

This is the first lab for ME 446. In this lab, we practiced D-H convention, derived solution for forward kinematics, and solved inverse kinematics using geometric approach.

Appendix: Code (main.c)

```
#include <tistdtypes.h>
#include <coecsl.h>
#include "user_includes.h"
#include "math.h"

// These two offsets are only used in the main file user_CRSRobot.c  You just need to
create them here and find the correct offset and then these offset will adjust the
encoder readings
float offset_Enc2_rad = -0.4238;
float offset_Enc3_rad = 0.2571;


// Your global varialbes.

long mycount = 0;

#pragma DATA_SECTION(whattoprint, ".my_vars") ///visible by matlab
float whattoprint = 0.0;

#pragma DATA_SECTION(theta1array, ".my_arrs") //visible by matlab
float theta1array[100];

#pragma DATA_SECTION(theta2array, ".my_arrs") //visible by matlab
float theta2array[100];

long arrayindex = 0;

float printtheta1motor = 0;
float printtheta2motor = 0;
float printtheta3motor = 0;
float printtheta1dh = 0;
float printtheta2dh = 0;
float printtheta3dh = 0;
float printendx = 0;
float printendy = 0;
float printendz = 0;
float printtheta1inv = 0;
float printtheta2inv = 0;
float printtheta3inv = 0;


// Assign these float to the values you would like to plot in Simulink
float Simulink_PlotVar1 = 0;
float Simulink_PlotVar2 = 0;
float Simulink_PlotVar3 = 0;
float Simulink_PlotVar4 = 0;


// This function is called every 1 ms
void lab(float theta1motor,float theta2motor,float theta3motor,float *tau1,float
*tau2,float *tau3, int error) {
```

```
        *tau1 = 0;
        *tau2 = 0;
        *tau3 = 0;

        //Motor torque limitation(Max: 5 Min: -5)

        // DH Frame Angle
        float theta1dh = theta1motor;
        float theta2dh = theta2motor - PI/2;
        float theta3dh = theta3motor-theta2motor+PI/2;

        // Forward Kinematics
        float endx = 25.4*cos(theta1dh)*(cos(theta2dh) + cos(theta2dh+theta3dh));
        float endy = 25.4*sin(theta1dh)*(cos(theta2dh) + cos(theta2dh+theta3dh));
        float endz = 25.4*(1-sin(theta2dh)-sin(theta2dh+theta3dh));

        // Forward Kinematics
        float theta1inv = atan2(endy,endx);
        float armlength = sqrt(endx*endx+endy*endy+(endz-25.4)*(endz-25.4));
        float theta2inv = - atan2(endz-25.4,sqrt(endx*endx+endy*endy)) -
acos(armlength*armlength/2/25.4/armlength);
        float theta3inv = 2 * acos(armlength*armlength/2/25.4/armlength);

        // save past states
        if ((mycount%50)==0) {

                theta1array[arrayindex] = theta1motor;
                theta2array[arrayindex] = theta2motor;

                if (arrayindex >= 100) {
                        arrayindex = 0;
                } else {
                        arrayindex++;
                }

        }



        if ((mycount%500)==0) {
                if (error != 0){
                        serial_printf(&SerialA, "error position\n\r");
                }
                else{
                if (whattoprint > 0.5) {
                        serial_printf(&SerialA, "I love robotics\n\r");
                } else {
                        printtheta1motor = theta1motor;
                        printtheta2motor = theta2motor;
                        printtheta3motor = theta3motor;
                        printtheta1dh = theta1dh;
                        printtheta2dh = theta2dh;
                        printtheta3dh = theta3dh;
                        printendx = endx;
                        printendy = endy;
```

```
                    printendz = endz;
                    printtheta1inv = theta1inv;
                    printtheta2inv = theta2inv;
                    printtheta3inv = theta3inv;
                    SWI_post(&SWI_printf); //Using a SWI to fix SPI issue from
sending too many floats.
             }}
             GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; // Blink LED on Control Card
             GpioDataRegs.GPBTOGGLE.bit.GPIO60 = 1; // Blink LED on Emergency Stop
Box
      }


      Simulink_PlotVar1 = theta1motor;
      Simulink_PlotVar2 = theta2motor;
      Simulink_PlotVar3 = theta3motor;
      Simulink_PlotVar4 = 0;

      mycount++;
}

void printing(void){
      serial_printf(&SerialA, "Motor Angle: %.2f %.2f,%.2f
\n\r",printtheta1motor*180/PI,printtheta2motor*180/PI,printtheta3motor*180/PI);
      serial_printf(&SerialA, "Joint Angle: %.2f %.2f,%.2f
\n\r",printtheta1dh*180/PI,printtheta2dh*180/PI,printtheta3dh*180/PI);
      serial_printf(&SerialA, "End Position: %.2f %.2f,%.2f
\n\r",printendx,printendy,printendz);
      serial_printf(&SerialA, "Inverse Angle: %.2f %.2f,%.2f
\n\r",printtheta1inv*180/PI,printtheta2inv*180/PI,printtheta3inv*180/PI);
}
```