Lab 1: Kinematic Transformations

Caleb Albers & Karun Koppula

May 11, 2018

1 Introduction

This lab serves as an introduction to working with the Thermo CRS Catalyst-5 robotic manipulator. The different sections familiarized us with the initialization of the robot, reading data and sending commands to the robot, and working with forward and inverse kinematic calculations for the manipulator. This provides us with a solid foundation for working with more complex functionality of the robot. Precise control builds on the concepts derived in this lab and will prove instrumental as we move forwards through the course material.

2 Calculating Home Position

The procedure to calculate the "Home Position" of the robot was as follows,

1. Align the arm given the physical markings on the body to the best of our capabilities, i.e. the zero position.
2. Move the arm to the desired home position.
3. Use the printed output from the relative encoders to determine the angular offset using Tera Term.

3 Denavit-Hartenberg Parameters

3.1 Coordinate Frame Assignment

We computed Denavit-Hartenberg (D-H) parameters for three of the five joints on the Thermo CRS Catalyst-5 (leaving the two end-effector joints static). All three joints are revolute, starting with Joint 1, as described in Figure 1. We assigned the World Coordinate Frame such that Joint 1 revolves around the World z-axis.

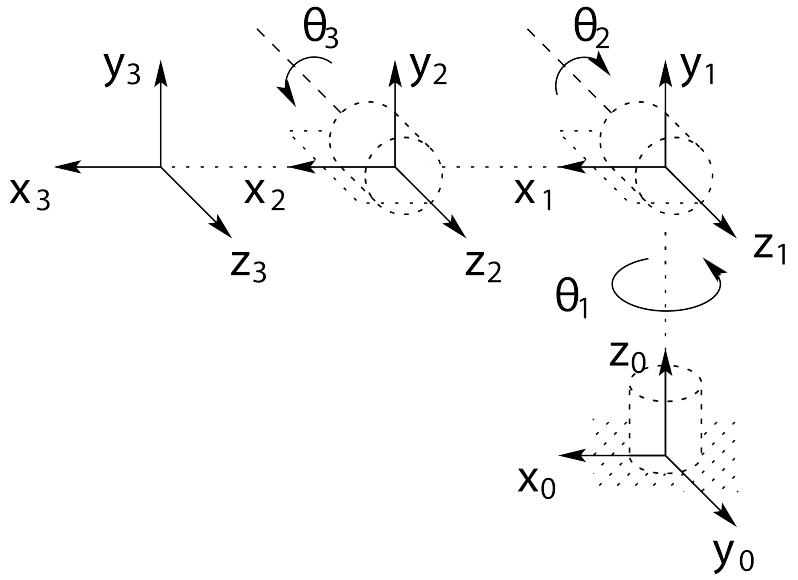

Figure 1: Denavit-Hartenberg (D-H) configuration of the Catalyst robot
(Figure adapted from Fig 3.1 in Ref. [1])

3.2 Parameter Measurements

Measurements of the D-H parameters are straightforward with respect to the robot dimensions given in Figure 2 (this Figure and others modified from it are courtesy of Ref. [2]). Since all joints are revolute, θ_i is variable in all cases. The length of the common normal for Joint 1, represented by a_1 in D-H notation, can be thought of as the radius of offset for Joint 1 with respect to the z-axis in the base frame. In this case, there is no radial offset, so $a_1 = 0$. The offset along the prior z-axis to Frame 1 is given by d_1, which for Joint 1 is equal to 10 inches. Additionally, the z-axis of Frame 0 is rotated by $\alpha_1 = \frac{-\pi}{2}$ with respect to Frame 1's z-axis.
No α rotations or d offsets exist for Joints 2 and 3, so the only parameters left to be measured are a_2 and a_3. Both of these are given by the length of their respective common normals. An easy visualization of these is given by the drawing in the bottom-right of Figure 2. The combined Denavit-Hartenberg parameters for all three joints can be seen below in Table 1.

Table 1: D-H Parameters

Joint	a_i	α_i	d_i	θ_i
1	0	$-\frac{\pi}{2}$	10	θ_1^*
2	10	0	0	θ_2^*
3	10	0	0	θ_3^*

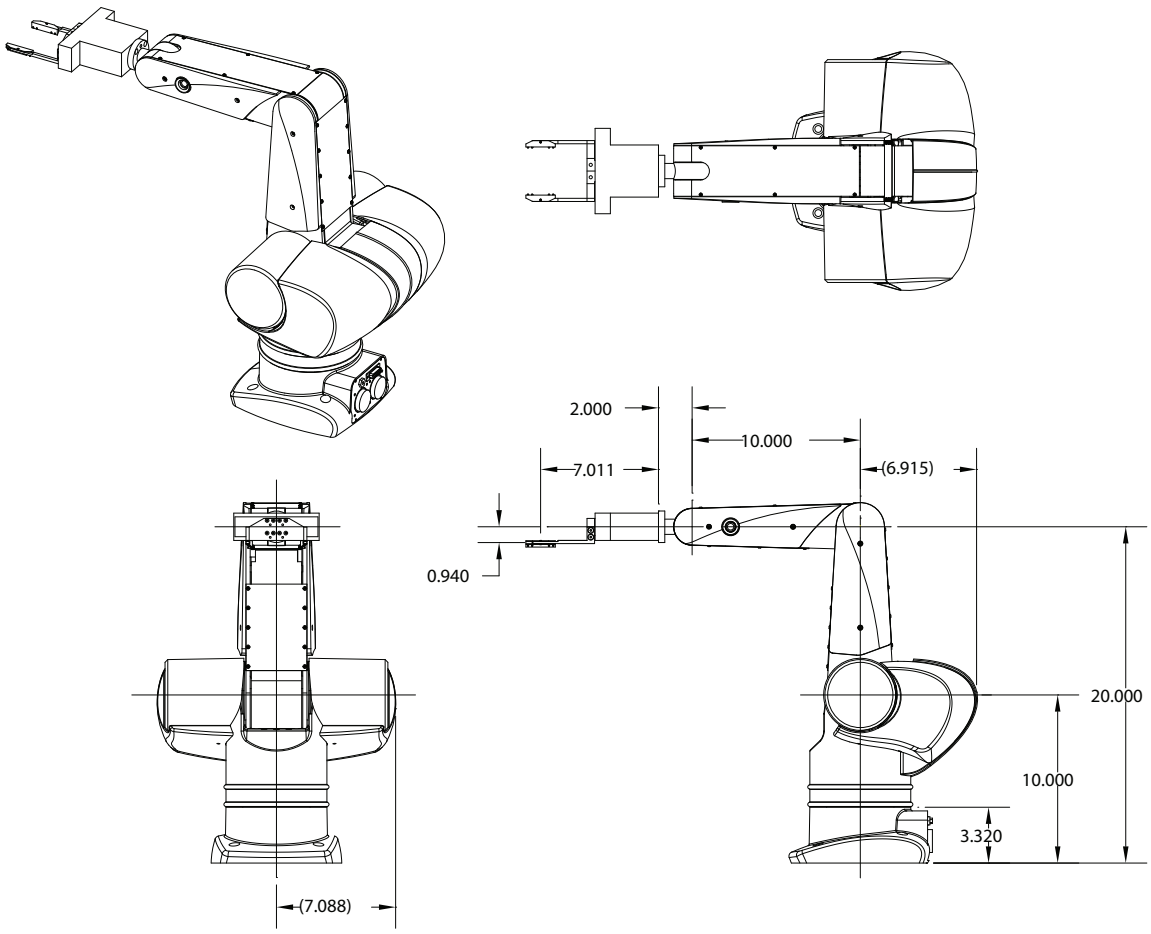

Figure 2: Robot Dimensions (inches)
(Figure courtesy of Ref. [2])

4 Forward Kinematics

4.1 Robotica

Inputting the Denavit-Hartenberg parameters into the Robotica software package allows for simple generation of the homogeneous transformation matrix T_3^0 (Eq. 1), the velocity Jacobian J_v (Eq. 2), and the angular Jacobian J_ω (Eq. 3).

$$T_3^0 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & 20c_1cos(q_2+\frac{q_3}{2})cos(\frac{q_3}{2}) \\ s_1c_{23} & -s_1s_{23} & c_1 & 20s_1cos(q_2+\frac{q_3}{2})cos(\frac{q_3}{2}) \\ -s_{23} & -c_{23} & 0 & -10(-1+s_2+s_{23}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$J_v = \begin{bmatrix} -20s_1cos(q_2+\frac{q_3}{2})cos(\frac{q_3}{2}) & -10c_1(s_2+s_{23}) & -10c_1s_{23} \\ 20c_1cos(q_2+\frac{q_3}{2})cos(\frac{q_3}{2}) & -10s_1(s_2+s_{23}) & -10s_1s_{23} \\ 0 & -20cos(q_2+\frac{q_3}{2})cos(\frac{q_3}{2}) & -10c_{23} \end{bmatrix} \tag{2}$$

$$J_\omega = \begin{bmatrix} 0 & -s_1 & -s_1 \\ 0 & c_1 & c_1 \\ 1 & 0 & 0 \end{bmatrix} \tag{3}$$

4.2 θ_{Motor} Correction

To be articulated independently, some joints on the Catalyst-5 robot require two motors to be driven simultaneously. To discover the conversion between these D-H parameter thetas (the actual joints) and the motor thetas, we calculated the motor and D-H theta values for four configurations of the robot manipulator shown in Figure 3. The positions were chosen to simplify the calculations required, where the joints were positioned at angles at 0 or $\pm\frac{\pi}{2}$ such that the trigonometric functions in the rotation matrices reduced to ones and zeros. The solution to the linear equations gives this set of equations for the motor theta values.

$$\theta_{Motor,1} = \theta_{DH,1} \tag{4}$$

$$\theta_{Motor,2} = \theta_{DH,2} + \frac{\pi}{2} \tag{5}$$

$$\theta_{Motor,3} = \theta_{DH,3} + \theta_{DH,2} \tag{6}$$

Position	$\theta_{DH,2}$	$\theta_{DH,3}$	$\theta_{Motor,2}$	$\theta_{Motor,3}$
1	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	0	0
2	$-\frac{\pi}{2}$	0	0	$-\frac{\pi}{2}$
3	0	0	$\frac{\pi}{2}$	0
4	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$

Table 2: D-H and Motor Thetas

Figure 3: Robot Positions for Testing

Substituting these corrections back into the Robotica calculations yields simplified forms for the Homogeneous Transformation (Eq. 7) and the Jacobians (Eq. 8 & 9). It is clear that the angular velocity Jacobian remains the same in both versions.

$$T_3^0 = \begin{bmatrix} c_1c_3 & -c_1s_3 & -s_1 & 10c_1(c_3+s_2) \\ c_3s_1 & -s_1s_3 & c_1 & 10s_1(c_3+s_2) \\ -s_3 & -c_3 & 0 & 10(1+c_2-s_3) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$J_v = \begin{bmatrix} -10s_1(c_3+s_2) & 10c_1(c_2-c_3) & -10c_1s_3 \\ 10c_1(c_3+s_2) & 10s_1(c_2-s_3) & -10s_1s_3 \\ 0 & -10(c_3+s_2) & -10c_3 \end{bmatrix} \tag{8}$$

$$J_\omega = \begin{bmatrix} 0 & -s_1 & -s_1 \\ 0 & c_1 & c_1 \\ 1 & 0 & 0 \end{bmatrix} \tag{9}$$

4.3 Physical Verification

To verify the correctness of the solution, the arm was moved to a point in the workspace and the output coordinates from the forward kinematics function was compared to an approximate measurement of the physical coordinates using a ruler. The physical error and RMS value are shown in Table 3. The physical measurements were taken using two bendable metal rulers 18 inches long and estimates of the positions of the global axes. It was hard to determine exact normals to the position of the end joint, which likely contributed to error in the measurement.

Position	error_x	error_y	error_z	$\text{error}_{\text{RMS}}$
(11.96, -10.57, 9.03)	0.405	0.305	0.54	0.782
(2.82, 11.29, 11.36)	0.18	0.165	0.11	0.268
(13.12, 13.13, 4.99)	0.255	0.255	0.24	0.433

Table 3: Forward Kinematics Calculation Error

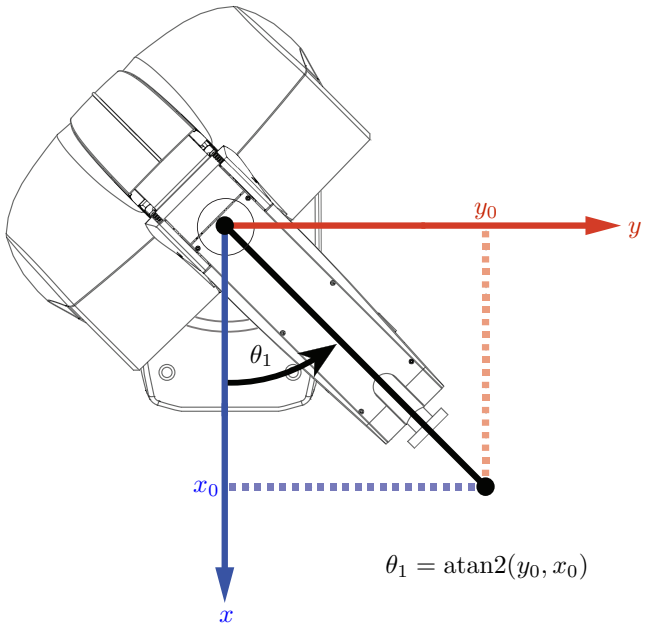

Figure 4: Inverse Kinematics Top View

5 Inverse Kinematics

5.1 Elbow-Up Geometric Approach

The sections of the robot that we modeled consist of a two-link planar arm mounted on top of a revolute joint aligned with the world x-axis. This configuration simplifies the inverse kinematics considerably, since the rotation about the world z-axis, the angle of rotation for the base motor, can be considered separately from the elbow and shoulder motors. With the given position of the end effector in the plane given by, (x_0, y_0) as shown in Figure 4, the base motor angle is solved for by,

$$\theta_1 = \text{atan2}(y_0, x_0) \tag{10}$$

The $\text{atan2}(y, x)$ function is used in this case to account for the possible positions of the robot in all four quadrants.
The elbow angle is solved for next, using the law of cosines given by Figure 5. We solve for the angle θ_C, given all three sides of the triangle in question. The upper two lengths are

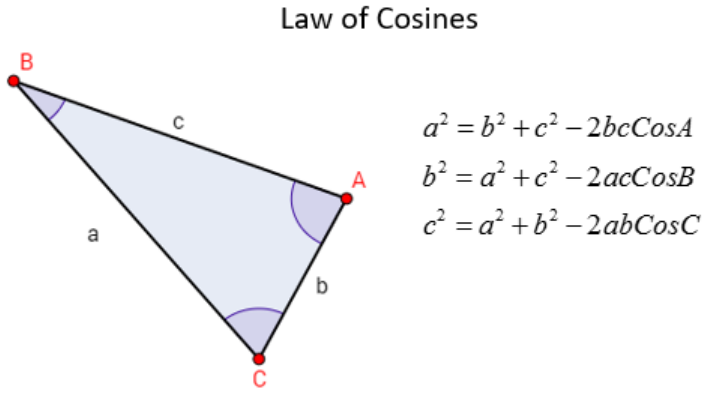

Figure 5: Law of Cosines
(Figure courtesy of Ref. [3])

given by a_2 and a_3, while the bottom length is given by Pythagoras' Theorem using the quantities derived from the end effector position. The quantities in question are shown in Figure 6.

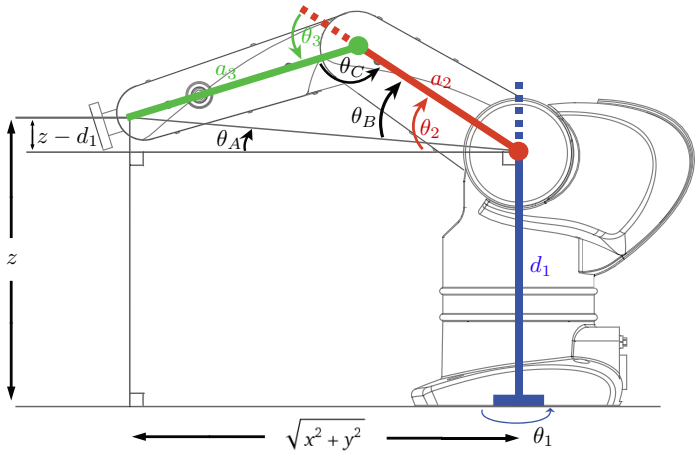

Figure 6: Inverse Kinematics Angles

Since θ_3 and θ_C are supplementary, the solution is given by,

$$\theta_3 = \pi - \arccos\left(\frac{-(x^2 + y^2 + (z-10)^2 - 200)}{200}\right) \tag{11}$$

The shoulder angle is solved using a simplification that takes into account the fact that both links of the planar arm are of the same length. First the angle is broken down into two separate angles, the first from the horizontal plane to the line between the shoulder joint

and the end effector and the second between that line and the first link of the planar arm itself. The first angle in the decomposition, θ_A can be solved for using an arc-tangent. The "y" component is given by the vertical position of the end effector, adjusted by the height of the base link, $z - d_1$. The "x" position is given by the radial distance of the end effector from the world z-axis, $\sqrt{x^2 + y^2}$. The second angle, θ_B uses the geometric rule that the inscribed angle is equal to half of the central angle. The configuration where the arm is fully extended corresponds to the diameter of the circle. Since θ_3 has already been solved for, θ_B can be taken to be half of that value.

$$\theta_2 = -(\text{atan2}((z - 10), \sqrt{x^2 + y^2}) + \frac{\theta_3}{2}) \tag{12}$$

5.2 Verification

Verification for the inverse kinematics functionality was provided by moving the arm to specific locations and checking that the motor angle outputs of the inverse kinematics matched the measured motor angles read by the encoders. Common sense checks were also used to see whether the calculated motor angles matched with appropriate rotations of the manipulator joints. For given positions, the error between the actual and calculated motor positions and RMS error are shown in Table 4. The error was zero across the board for the significant digits given by the motor readings. This is in contrast to the error in the forward kinematic readings, which were used to derive the motor angles. We believe that the forward kinematics are much more accurate than portrayed, due to the gross lack of precision in measuring the physical position of the end reference frame.

Position	error_{θ_1}	error_{θ_2}	error_{θ_3}	$\text{error}_{\text{RMS}}$
(10, 0, 20)	0	0	0	0
(13.12, 13.12, 5)	0	0	0	0
(0, -20, 10)	0	0	0	0

Table 4: Inverse Kinematics Calculation Error

6 Conclusion

This lab was multifaceted; We started with an introduction to the Thermo CRS Catalyst-5 robot via printing angle measurements and controlling hardware in C. From there, we assigned reference frames, built a Denavit-Hartenberg representation, and moved on to describing the forward kinematics of the robot. Once complete, a geometric approach was taken in solving the inverse kinematics. The lab was concluded with combining the forward and inverse kinematics equations in such a way that we were able to verify the validity and accuracy of our work by physically measuring the absolute position of the robot in the world frame and comparing it with a theoretical solution, which proved correct.

References

[1] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2005. ISBN: 9780471649908. URL: https://books.google.com/books?id=wGapQAAACAAJ.

[2] Inc. American Robot Sales. *Engineering Information*. URL: http://www.robotsdotcom.com/si.htm#enginfo.

[3] C. Zwikker. *The advanced geometry of plane curves and their applications: (formerly titled: Advanced plane geometry)*. Dover Publications, 1963. URL: https://books.google.com/books?id=Uk06AAAAMAAJ.

A lab.c code

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


#define PI 3.14159

// 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 = -0.37;
//float offset_Enc3_rad = 0.27;

float offset_Enc2_rad = -0.427257;
float offset_Enc3_rad = 0.230558;


// Your global variables.

long mycount = 0;

#pragma DATA_SECTION(whattoprint, ".my_vars")
float whattoprint = 0.0;

#pragma DATA_SECTION(theta1array, ".my_arrs")
float theta1array[100];

#pragma DATA_SECTION(theta2array, ".my_arrs")
float theta2array[100];


long arrayindex = 0;

float printtheta1motor = 0;
float printtheta2motor = 0;
float printtheta3motor = 0;

float x_pos = 0;
float y_pos = 0;
float z_pos = 0;

// inverse kinematics
float theta_1 = 0;
float theta_2 = 0;
float theta_3 = 0;

float motor_theta_1 = 0;
float motor_theta_2 = 0;
float motor_theta_3 = 0;
```

```
49
50 // Assign these float to the values you would like to plot in Simulink
51 float Simulink_PlotVar1 = 0;
52 float Simulink_PlotVar2 = 0;
53 float Simulink_PlotVar3 = 0;
54 float Simulink_PlotVar4 = 0;
55
56
57 //This function calculates the motor positions given a desired x,y,and z
       position
58 void inverse_kinematics(float x, float y, float z){
59
60     //The DH angles are calculated using a geometric analysis of the possible
       configurations of the robot
61     theta_1 = atan2(y,x);
         //The base DH angle is the inverse tangent between the x and y
       coordinates
62     theta_3 = 3.14159 - acos(-(pow(x,2)+pow(y,2)+pow(z-10,2)-200)/200);
         //The elbow DH angle is calculated using the law of cosines
63     theta_2 = -(atan2(z-10,sqrt(pow(x,2)+pow(y,2))) + theta_3/2);
         //The shoulder DH angle is calculated using Pythagoras' Theorem and the
       half angle formula
64
65     //The DH angles must be converted to motor theta angles using appropriate
       transformation
66     motor_theta_1 = theta_1;
67     motor_theta_2 = theta_2 + PI/2;
68     motor_theta_3 = theta_3 + theta_2;
69 }
70
71 //This function calculates the forward kinematics of the manipulator given the
        motor positions
72
73 void forward_kinematics(float motor1, float motor2, float motor3){
74
75     //The forward kinematics function uses the translational vector from the
       full DH matrix calculated in Robotica
76     x_pos = 10*cos(motor1)*(cos(motor3)+sin(motor2));
77     y_pos = 10*sin(motor1)*(cos(motor3)+sin(motor2));
78     z_pos = 10*(1+cos(motor2)-sin(motor3));
79
80     }
81
82 // This function is called every 1 ms
83 void lab(float theta1motor,float theta2motor,float theta3motor,float *tau1,
       float *tau2,float *tau3, int error) {
84
85
86     *tau1 = 0;
87     *tau2 = 0;
88     *tau3 = 0;
89
```

```
90      //Motor torque limitation(Max: 5 Min: -5)
91
92      // save past states
93      if ((mycount%50)==0) {
94
95          theta1array[arrayindex] = theta1motor;
96          theta2array[arrayindex] = theta2motor;
97
98          if (arrayindex >= 100) {
99              arrayindex = 0;
100         } else {
101             arrayindex++;
102         }
103     }
104
105     /*
106      * Forward Kinematics
107      *
108      * based on motor angles
109      */
110
111     forward_kinematics(theta1motor, theta2motor, theta3motor);
112
113     /*
114      * Inverse Kinematics
115      *
116      * based on {x,y,z} pos calculated above
117      */
118
119     inverse_kinematics(x_pos, y_pos, z_pos);
120
121     if ((mycount%500)==0) {
122         if (whattoprint > 0.5) {
123             serial_printf(&SerialA, "I love robotics\n\r");
124         } else {
125             printtheta1motor = theta1motor;
126             printtheta2motor = theta2motor;
127             printtheta3motor = theta3motor;
128
129
130             SWI_post(&SWI_printf); //Using a SWI to fix SPI issue from sending
     too many floats.
131         }
132         GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; // Blink LED on Control Card
133         GpioDataRegs.GPBTOGGLE.bit.GPIO60 = 1; // Blink LED on Emergency Stop
    Box
134     }
135
136     Simulink_PlotVar1 = theta1motor;
137     Simulink_PlotVar2 = theta2motor;
138     Simulink_PlotVar3 = theta3motor;
139     Simulink_PlotVar4 = 0;
```

```
140
141     mycount++;
142 }
143
144 void printing(void){
145     serial_printf(&SerialA, "%.2f %.2f,%.2f    \n\r",printtheta1motor,
        printtheta2motor,printtheta3motor);
146     serial_printf(&SerialA, "x: %.2f,    y: %.2f,   z: %.2f    \n\r",x_pos,y_pos,
        z_pos);
147     //serial_printf(&SerialA, "Estimated IK solution: theta1: %.2f,    theta2:
        %.2f,   theta3: %.2f    \n\r",theta_1,theta_2,theta_3);
148     serial_printf(&SerialA, "Estimated IK solution: motor_theta1: %.2f,
        motor_theta2: %.2f,   motor_theta3: %.2f    \n\r",motor_theta_1,
        motor_theta_2,motor_theta_3);
149 }
```

Listing 1: lab.c