Lab 1: Kinematic Transformations

Written By Tyler Smithline and Leo Jiang

ECE 489: Robot Dynamics and Control

Lab Section AB1

Introduction

The purpose of lab 1 is to learn how to control the CRS robot arm through Code Composer Studio IDE and to derive and apply the forward kinematic and inverse kinematic equations to the robot arm.

Calibrate

The first part of lab was to calibrate the robot arm to make sure the encoder motor angles matched the actual position of the robot. Since the robot needs a consistent starting position to zero the encoders, we start the robot with the joints pressed against the hard stops. Since this is not a convenient zero position, we use offsets to define a zero-position relative to the start position. These offsets need to be calibrated accurately for the robot to move to precise locations. We accomplished this calibration by setting the offsets to zero and manually positioning the arm in its zero position, starting the program to zero the encoders, and then manually moving the arm to the start position. The encoder values printed to Tera Term are the offsets used.

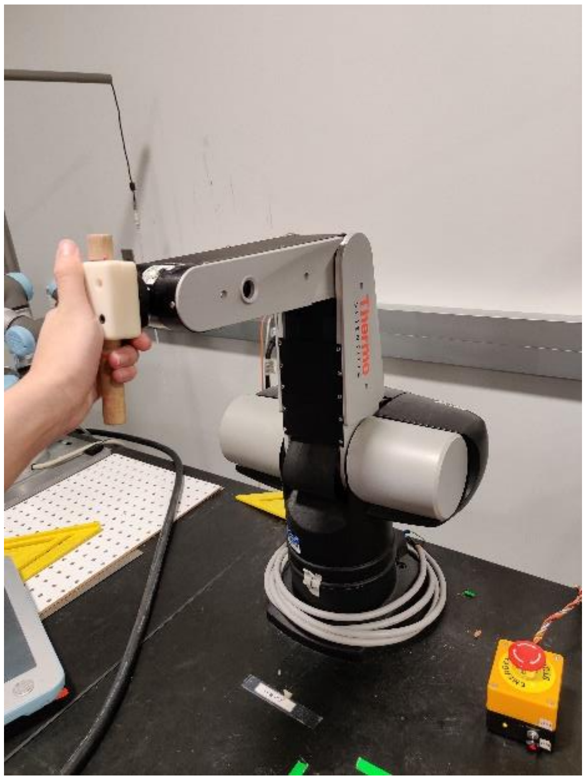


Figure 1. Zero Position of the CRS Robot Arm

Forward Kinematics

The second part of lab 1 is to calculate the end-effector position using DH angles. The first step in this calculation is determining the DH Angles according to the convention in the textbook. This convention is described below.

DH convention

a_i = distance along x_i from the intersection of the x_i and z_{i-1} axes to o_i

d_i = distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. If joint i is prismatic, d_i is variable

α_i = the angle from z_{i-1} to z_i measured about x_i

θ_i = the angle from x_{i-1} to x_i measured about z_{i-1}. If joint i is revolute, θ_i is variable.

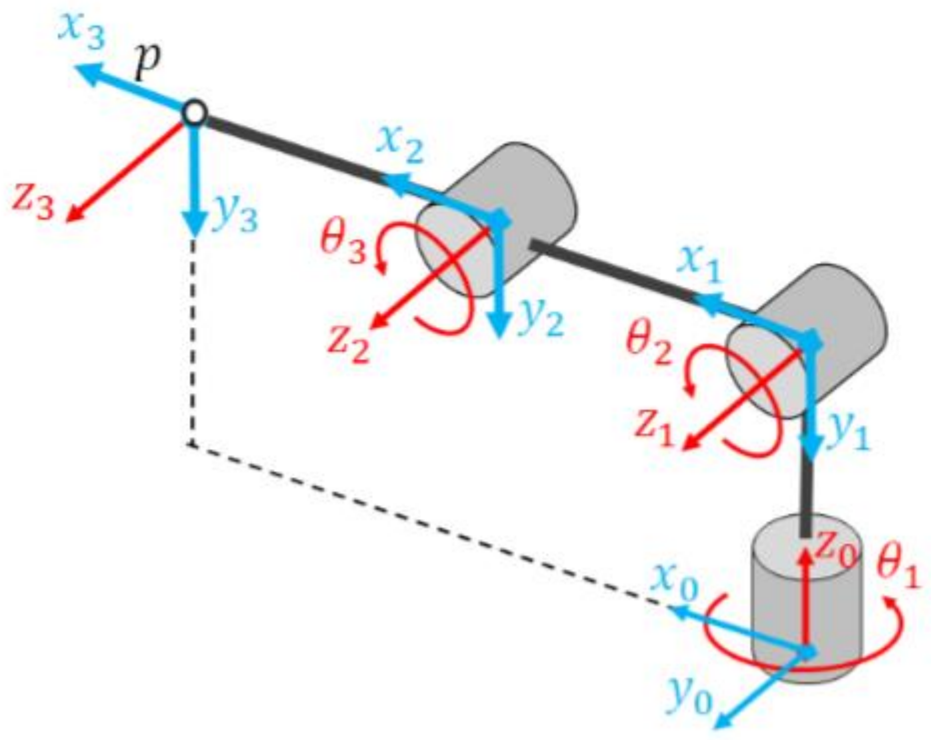


Figure 2: The robot arm orientation used to determine DH parameters

Using the orientation from the image above, we can determine DH parameters for the robot arm.

Joint	**a_i (meters)**	**α_i (degrees)**	**d_i (meters)**	**θ_i (degrees)**
1	0	-90	0.254	θ_1
2	0.254	0	0	θ_2
3	0.254	0	0	θ_3

Table 1: DH parameters for our robot arm

The DH values in the table above were calculated according to the DH convention above. For the **a_i** values, **a_1** is the location of joint 1 along **x_0**, and therefore is 0 because joint 1 is along joint 0's axis of rotation. **a_2** and **a_3** are both 0.254 meters (10 inches) because those joints are offset 10 inches from the previous joint along the x axis. Next, the **α_1** value is -90 degrees because it is measured 90 degrees from **z_0** to **z_1** measured about **x_1**. **α_2** and **α_3** are both 0 degrees since the x-axes for joints 1, 2, and 3 were all chosen to be in alignment with each other, and there is no rotation about the x axes between these joints. The **d_0** value is 10 inches because joint 1 is 10 inches above joint 0 and is therefore 10 inches along the **z_0** axis from joint 0. Since joints 1, 2, and 3 all lie on the same axes in the orientation we defined above, there is no offset along the z axes for joints 2 and 3 and therefore **d_2** and **d_3** are both 0. Finally, **θ_1, θ_2,** and **θ_3** are all equal to the DH motor angles that we will define below, because these values start as 0 per our initial orientation but will change as the robot moves. Using the DH parameters at each joint, we can express each joint's homogenous transformation matrix (HTM) relative to the previous joint using the following calculation:

$$H_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The HTM that represents the world frame to the end effector frame can be expressed as:

$$H_3^0 = H_1^0 * H_2^1 * H_3^2$$

The mechanical design of the arm is such that each motor doesn't directly actuate each joint, so we also need to calculate the DH theta in terms of motor theta. To accomplish this, we set up the robot in three different positions to formulate a three-variable system of equations.

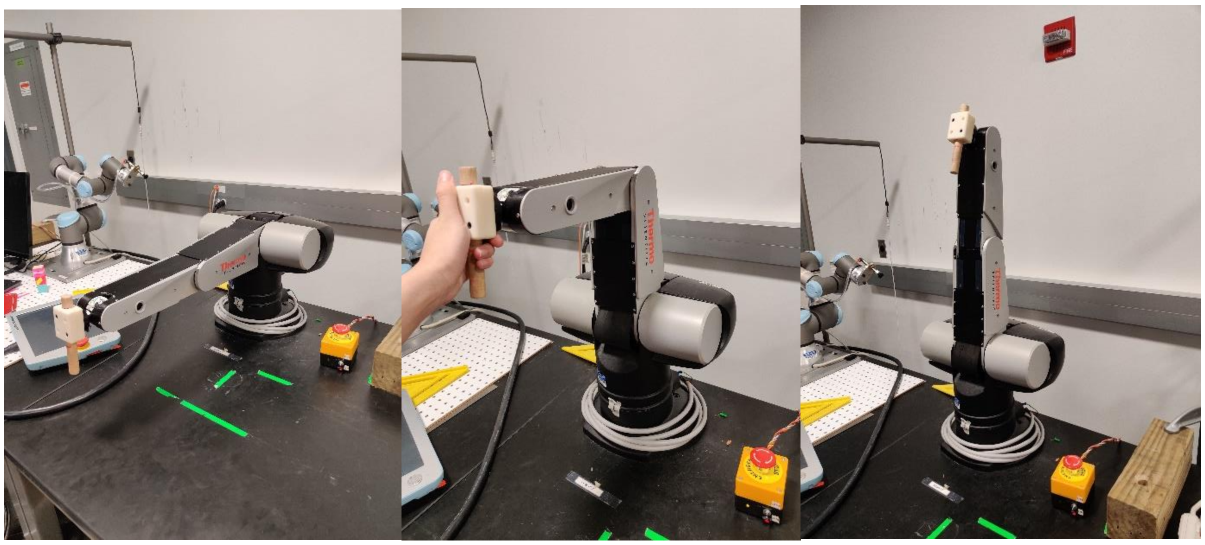


Figure 3. Three positions of the robot arm used for mapping DH angles to motor angles

From Figure 1, we can see the DH thetas for positions one, two, and three are [0,0,0], [0, -90, 90], and [0, -90, 0] respectively. The motor thetas can be found using the printf function in the C code. The motor thetas for each position are [0, 90, 0], [0, 0, 0], and [0, 0, -90]. Since the first motor is at the same position as the DH angle, motor theta 1 is the same as DH theta1. DH theta 2 has an offset to motor theta 2, and DH theta 3 is a combination of motor theta 2 and motor theta 3, so we can formulate the systems of equations as follows:

$$\theta_{DH1} = \theta_{motor1}$$

$$\theta_{DH2} = \theta_{motor2} + c4$$

$$\theta_{DH2} = c1 * \theta_{motor2} + c2 * \theta_{motor3} + c3$$

Plugging in the values we found earlier, we can find the values of the constants:

$$c1 = -1, c2 = 1, c3 = \frac{\pi}{2}, c4 = -\frac{\pi}{2}$$

Finally, we can express the H03 transformation matrix in term of motor thetas. This will give us the position and orientation of joint 3 relative to the robot's base. The Matlab code that creates H03 from the DH parameters is shown in Appendix A. H03 is shown below:

$$\begin{matrix} \cos(\theta_{m1})*\cos(\theta_{m3}) & -\cos(\theta_{m1})*\sin(\theta_{m3}) & \sin(\theta_{m1}) & 127*\cos(\theta_{m1})*\left(\cos(\theta_{m3})+\sin(\theta_{m2})\right)/500 \\ \cos(\theta_{m3})*\sin(\theta_{m1}) & -\sin(\theta_{m1})*\sin(\theta_{m3}) & \cos(\theta_{m1}) & 127*\sin(\theta_{m1})*\left(\cos(\theta_{m3})+\sin(\theta_{m2})\right)/500 \\ -\sin(\theta_{m3}) & -\cos(\theta_{m3}) & 0 & \dfrac{127*\cos(\theta_{m2})}{500}-\dfrac{127*\sin(\theta_{m3})}{500}+\dfrac{127}{500} \\ 0 & 0 & 0 & 1 \end{matrix}$$

Since a homogenous matrix is composed of both a 3x3 rotation matrix and 3x1 translation matrix, the forward kinematics result is simply equal to the 3x1 translation matrix for a given set of input motor theta angles. Therefore, we print take the last column of the H03 in code composer using printf to print out the calculated forward kinematic position (x, y, and z coordinates) in different positions. We validated our code by manually positioning the CRS robot arm and confirming that the measured position was equal to our calculated forward kinematic positions.

Inverse Kinematics

We can calculate an inverse kinematic solution of the robot using geometry. To calculate our geometric solution, we defined important values as shown below:

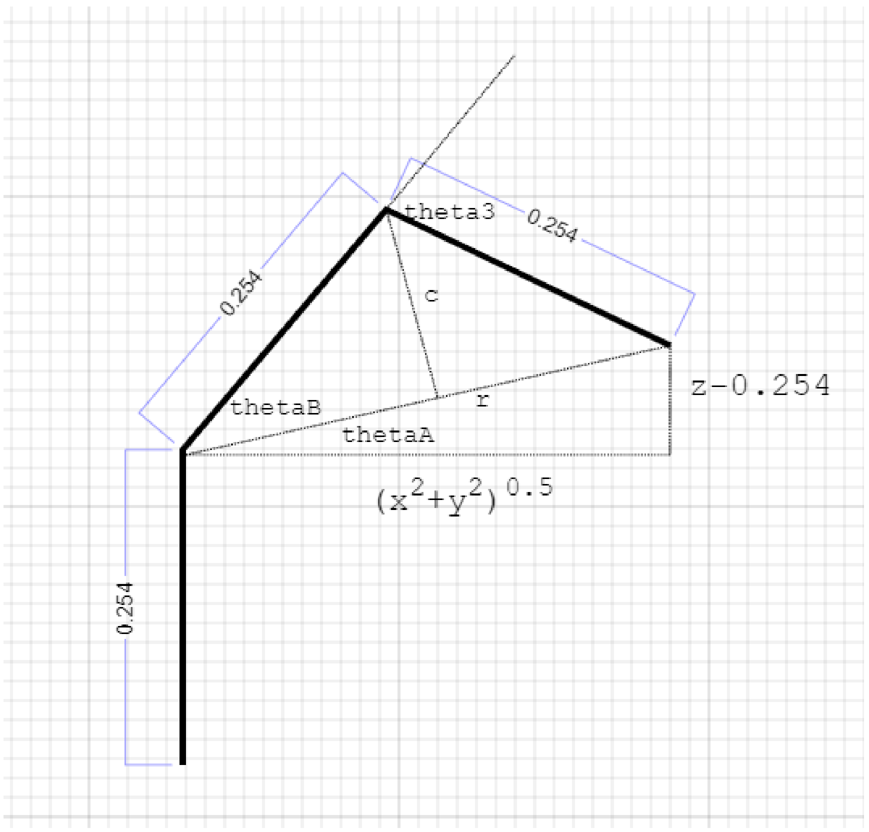


Figure 4. Inverse Kinematic Geometry

In the image above, x, y, and z are the coordinates of joint 3 (the end effector for our purpose) in the world frame. Since DH θ1 is the rotation of the arm, it relies only on x and y and is not impacted by z.

$$DH\ \theta_1 = \arctan\left(\frac{y}{x}\right)$$

To find DH θ2 and DH θ3, we can break down the upper two links into two triangles. As shown in figure 4, θA can be expressed as:

$$\theta_A = \arctan\left(\frac{z - 0.254}{\sqrt{x^2 + y^2}}\right)$$

The value for r can be found using Pythagorean theorem.

$$r = \sqrt{x^2 + y^2 + (z - 0.254)^2}$$

Because the two links forms an Isosceles triangle, we can find the length of c using Pythagorean theorem

$$c = \sqrt{0.254^2 - \left(\frac{r}{2}\right)^2} = \sqrt{0.254^2 - \frac{x^2 + y^2 + (z - 0.254)^2}{4}}$$

Thus, θB is:

$$\theta_b = \arctan\left(\frac{c}{0.5r}\right) = \arctan\left(\frac{2c}{r}\right)$$

Using θA and θB, we can find DH θ2 and DH θ3:

$$\theta_2 = -(\theta_a + \theta_b)$$

$$\theta_3 = 2 * \theta_b$$

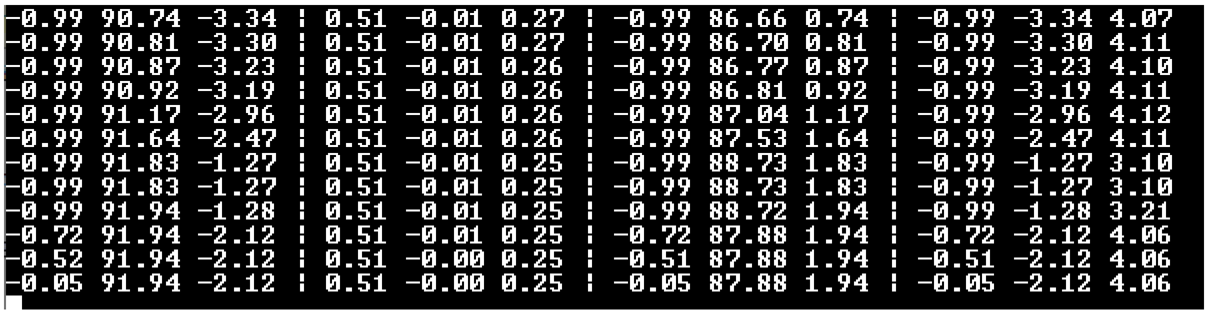
We then converted the DH thetas into motor thetas; This is the opposite of the conversion from motor thetas to DH thetas used in forward kinematics. The MATLAB code that calculates thetas is shown in Appendix B. We verified our results by comparing the joint angles provided by C code and the joint angles calculated from inverse kinematics.

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

$$\theta_{motor2} = \theta_{DH2} + \pi/2$$

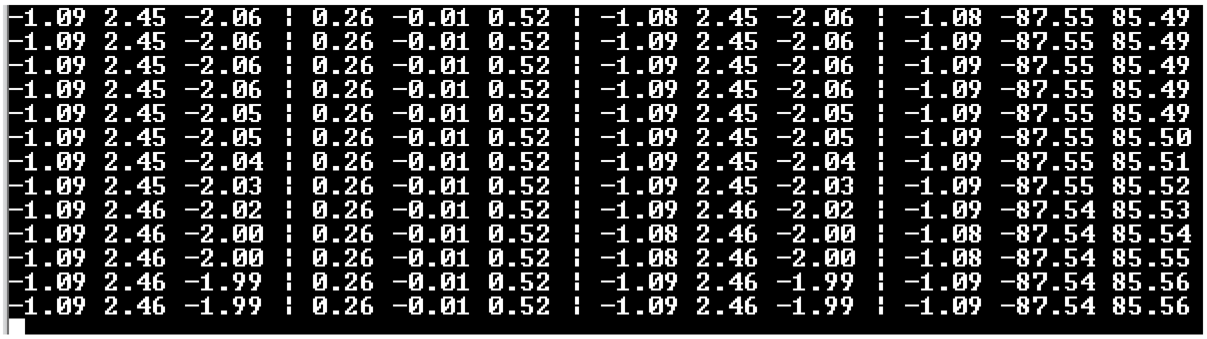
$$\theta_{motor3} = \theta_{motor2} + \theta_{3DH} - \pi/2$$

Below is the output for our inverse kinematic solution. Given the x, y, and z coordinates from our forward kinematic solution, the inverse kinematic code determines a set of motor thetas that will move the arm to the correct position and orientation. The tera term code prints four values, separated by a vertical line. These sets of values are motor thetas from encoders, cartesian coordinates from forward kinematics, motor thetas from inverse kinematics, and DH thetas from inverse kinematics. These values demonstrate that the calculated motor thetas are very close to the ones being read by the motor encoders.



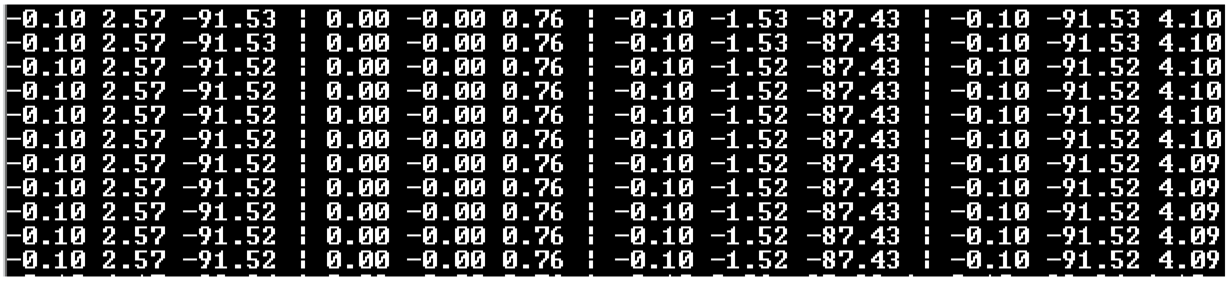
```
-0.99 90.74 -3.34 ¦ 0.51 -0.01 0.27 ¦ -0.99 86.66 0.74 ¦ -0.99 -3.34 4.07
-0.99 90.81 -3.30 ¦ 0.51 -0.01 0.27 ¦ -0.99 86.70 0.81 ¦ -0.99 -3.30 4.11
-0.99 90.87 -3.23 ¦ 0.51 -0.01 0.26 ¦ -0.99 86.77 0.87 ¦ -0.99 -3.23 4.10
-0.99 90.92 -3.19 ¦ 0.51 -0.01 0.26 ¦ -0.99 86.81 0.92 ¦ -0.99 -3.19 4.11
-0.99 91.17 -2.96 ¦ 0.51 -0.01 0.26 ¦ -0.99 87.04 1.17 ¦ -0.99 -2.96 4.12
-0.99 91.64 -2.47 ¦ 0.51 -0.01 0.26 ¦ -0.99 87.53 1.64 ¦ -0.99 -2.47 4.11
-0.99 91.83 -1.27 ¦ 0.51 -0.01 0.25 ¦ -0.99 88.73 1.83 ¦ -0.99 -1.27 3.10
-0.99 91.83 -1.27 ¦ 0.51 -0.01 0.25 ¦ -0.99 88.73 1.83 ¦ -0.99 -1.27 3.10
-0.99 91.94 -1.28 ¦ 0.51 -0.01 0.25 ¦ -0.99 88.72 1.94 ¦ -0.99 -1.28 3.21
-0.72 91.94 -2.12 ¦ 0.51 -0.01 0.25 ¦ -0.72 87.88 1.94 ¦ -0.72 -2.12 4.06
-0.52 91.94 -2.12 ¦ 0.51 -0.00 0.25 ¦ -0.51 87.88 1.94 ¦ -0.51 -2.12 4.06
-0.05 91.94 -2.12 ¦ 0.51 -0.00 0.25 ¦ -0.05 87.88 1.94 ¦ -0.05 -2.12 4.06
```

Figure 5. DH Zero Position



```
-1.09 2.45 -2.06 ¦ 0.26 -0.01 0.52 ¦ -1.08 2.45 -2.06 ¦ -1.08 -87.55 85.49
-1.09 2.45 -2.06 ¦ 0.26 -0.01 0.52 ¦ -1.09 2.45 -2.06 ¦ -1.09 -87.55 85.49
-1.09 2.45 -2.06 ¦ 0.26 -0.01 0.52 ¦ -1.09 2.45 -2.06 ¦ -1.09 -87.55 85.49
-1.09 2.45 -2.06 ¦ 0.26 -0.01 0.52 ¦ -1.09 2.45 -2.06 ¦ -1.09 -87.55 85.49
-1.09 2.45 -2.05 ¦ 0.26 -0.01 0.52 ¦ -1.09 2.45 -2.05 ¦ -1.09 -87.55 85.49
-1.09 2.45 -2.05 ¦ 0.26 -0.01 0.52 ¦ -1.09 2.45 -2.05 ¦ -1.09 -87.55 85.50
-1.09 2.45 -2.04 ¦ 0.26 -0.01 0.52 ¦ -1.09 2.45 -2.04 ¦ -1.09 -87.55 85.51
-1.09 2.45 -2.03 ¦ 0.26 -0.01 0.52 ¦ -1.09 2.45 -2.03 ¦ -1.09 -87.55 85.52
-1.09 2.46 -2.02 ¦ 0.26 -0.01 0.52 ¦ -1.09 2.46 -2.02 ¦ -1.09 -87.54 85.53
-1.09 2.46 -2.00 ¦ 0.26 -0.01 0.52 ¦ -1.08 2.46 -2.00 ¦ -1.08 -87.54 85.54
-1.09 2.46 -2.00 ¦ 0.26 -0.01 0.52 ¦ -1.08 2.46 -2.00 ¦ -1.08 -87.54 85.55
-1.09 2.46 -1.99 ¦ 0.26 -0.01 0.52 ¦ -1.09 2.46 -1.99 ¦ -1.09 -87.54 85.56
-1.09 2.46 -1.99 ¦ 0.26 -0.01 0.52 ¦ -1.09 2.46 -1.99 ¦ -1.09 -87.54 85.56
```

Figure 6: Motor Zero Position



```
-0.10 2.57 -91.53 ¦ 0.00 -0.00 0.76 ¦ -0.10 -1.53 -87.43 ¦ -0.10 -91.53 4.10
-0.10 2.57 -91.53 ¦ 0.00 -0.00 0.76 ¦ -0.10 -1.53 -87.43 ¦ -0.10 -91.53 4.10
-0.10 2.57 -91.52 ¦ 0.00 -0.00 0.76 ¦ -0.10 -1.52 -87.43 ¦ -0.10 -91.52 4.10
-0.10 2.57 -91.52 ¦ 0.00 -0.00 0.76 ¦ -0.10 -1.52 -87.43 ¦ -0.10 -91.52 4.10
-0.10 2.57 -91.52 ¦ 0.00 -0.00 0.76 ¦ -0.10 -1.52 -87.43 ¦ -0.10 -91.52 4.10
-0.10 2.57 -91.52 ¦ 0.00 -0.00 0.76 ¦ -0.10 -1.52 -87.43 ¦ -0.10 -91.52 4.10
-0.10 2.57 -91.52 ¦ 0.00 -0.00 0.76 ¦ -0.10 -1.52 -87.43 ¦ -0.10 -91.52 4.09
-0.10 2.57 -91.52 ¦ 0.00 -0.00 0.76 ¦ -0.10 -1.52 -87.43 ¦ -0.10 -91.52 4.09
-0.10 2.57 -91.52 ¦ 0.00 -0.00 0.76 ¦ -0.10 -1.52 -87.43 ¦ -0.10 -91.52 4.09
-0.10 2.57 -91.52 ¦ 0.00 -0.00 0.76 ¦ -0.10 -1.52 -87.43 ¦ -0.10 -91.52 4.09
-0.10 2.57 -91.52 ¦ 0.00 -0.00 0.76 ¦ -0.10 -1.52 -87.43 ¦ -0.10 -91.52 4.09
```

Figure 7. Straight Up

Conclusion

In this lab, we familiarized ourselves with the CRS robot arm, Code Composer Studio, and how we can use kinematic transformations to position the arm at a specific location. More specifically, we used DH parameter conventions and the known geometry of the robot arm to calculate forward and inverse kinematics for the first three joints of the arm. Additionally, we found the conversion from DH thetas to motor thetas. Using these calculations, we can now determine the location of the end effector from known motor thetas as well as the motor thetas from a desired end effector position. We will be able to use these tools in future labs to accomplish more complicated tasks.

Appendix A: Forward Kinematics Code to Calculate H03

```
al1 = -sym(pi)/2;
al2 = 0;
al3 = 0;
a1 = 0;
a2 = 0.254;
a3 = 0.254;
d1 = 0.254;
d2 = 0;
d3 = 0;

c1 = -1;
c2 = 1;
c3 = sym(pi)/2;
c4 = -sym(pi)/2;
syms mt1 mt2 mt3
t1 = mt1;
t2 = mt2 + c4;
t3 = c1*mt2 + c2*mt3 + c3;

% DH1 transformation
A1 = [[cos(t1) -sin(t1)*cos(al1) sin(t1)*sin(al1) a1*cos(t1)];
      [sin(t1) cos(t1)*cos(al1) -cos(t1)*sin(al1) a1*sin(t1)];
      [0 sin(al1) cos(al1) d1];
      [0 0 0 1]];

% DH2 transformation
A2 = [[cos(t2) -sin(t2)*cos(al2) sin(t2)*sin(al2) a2*cos(t2)];
      [sin(t2) cos(t2)*cos(al2) -cos(t2)*sin(al2) a2*sin(t2)];
      [0 sin(al2) cos(al2) d2];
      [0 0 0 1]];

% DH3 transformation
A3 = [[cos(t3) -sin(t3)*cos(al3) sin(t3)*sin(al3) a3*cos(t3)];
      [sin(t3) cos(t3)*cos(al3) -cos(t3)*sin(al3) a3*sin(t3)];
      [0 sin(al3) cos(al3) d3];
      [0 0 0 1]];

% End effector frame relative to base coordinate frame
H03 = simplify(A1*A2*A3)
```

Appendix B: MATLAB Code Used to Calculate IK Solution in terms of Motor Thetas

```
syms x y z

% c offsets that convert motor thetas do dh thetas; not used here
c1 = -1;
c2 = 1;
c3 = sym(pi)/2;
c4 = -sym(pi)/2;

% Inverse kinematic calculations derived from arm geometry
thetaa = atan2((z-0.254),(x^2+y^2)^0.5);
r = (x^2+y^2+(z-0.254)^2)^0.5;
c = (0.254^2-(r/2)^2)^0.5;
thetab = atan2(2*c,r);

% Calculated DH thetas using geometric calculations from above
theta1 = atan2(y,x);
theta2 = -(thetaa+thetab);
theta3 = 2*thetab;

% Conversion from dh thetas to motor thetas
motortheta1 = theta1
motortheta2 = simplify(vpa(theta2 + pi/2))
motortheta3 = simplify(vpa(theta3 + motortheta2 - pi/2))
```

Appendix C: Lab 1 C Code (From Code Composer Studio)

```c
#include "math.h"
#include "F28335Serial.h"

#define PI          3.1415926535897932384626433832795
#define TWOPI       6.283185307179586476925286766559
#define HALFPI      1.5707963267948966192313216916398
#define GRAV        9.81

// 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 = -0.4454;
float offset_Enc3_rad = 0.2436;
//float offset_Enc2_rad = 0.0;
//float offset_Enc3_rad = 0.0;

// Your global variables.
long mycount = 0;

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

#pragma DATA_SECTION(theta1array, ".my_arrs")
float theta1array[100];
#pragma DATA_SECTION(theta2array, ".my_arrs")
float theta2array[100];

long arrayindex = 0;
int UARTprint = 0;

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

float motortheta1 = 0;
float motortheta2 = 0;
float motortheta3 = 0;

float theta1 = 0;
float theta2 = 0;
float theta3 = 0;

float x = 0;
float y = 0;
float z = 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 = 0.0;
    *tau2 = 0.0;
    *tau3 = 0.0;

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

    // save past states
    if ((mycount%50)==0) {
        theta1array[arrayindex] = theta1motor;
        theta2array[arrayindex] = theta2motor;

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

    if ((mycount%500)==0) {
        UARTprint = 1;
        GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; // Blink LED on Control Card
        GpioDataRegs.GPBTOGGLE.bit.GPIO60 = 1; // Blink LED on Emergency Stop Box
    }
    printtheta1motor = theta1motor;
    printtheta2motor = theta2motor;
    printtheta3motor = theta3motor;

// Forward kinematic calculations that gives calculated x,y,z values given motor thetas
    x = (127.0*cos(theta1motor)*(cos(theta3motor) + sin(theta2motor)))/500.0;
    y = (127.0*sin(theta1motor)*(cos(theta3motor) + sin(theta2motor)))/500.0;
    z = (127.0*cos(theta2motor))/500.0 - (127.0*sin(theta3motor))/500.0 + 127.0/500.0;

// Inverse kinematic calculations that produce motor thetas that achieve the desired end
effector position
    motortheta1 = atan2(y,x);
    motortheta2 = 1.570796 - 1.0*atan2(1.0*z - 0.254, sqrt(x*x + y*y)) -
1.0*atan2(2.0*sqrt(0.064516 - 0.25*x*x - 0.25*y*y - 0.25*(z - 0.254)*(z-0.254)), sqrt((z -
0.254)*(z - 0.254) + x*x + y*y));
    motortheta3 = 1.0*atan2(2.0*sqrt(0.064516 - 0.25*x*x - 0.25*y*y - 0.25*(z - 0.254)*(z-
0.254)), sqrt((z - 0.254)*(z-0.254) + x*x + y*y)) - 1.0*atan2(1.0*z - 0.254, sqrt(x*x + y*y));

// Conversion of motor thetas to DH angles
    theta1 = motortheta1;
    theta2 = motortheta2 - PI/2;
    theta3 = motortheta3 - motortheta2 + PI/2;

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

    mycount++;
}

// Print motor thetas from encoders, forward kinematic results, inverse kinematic results, and
dh inverse kinematic results
void printing(void){
    serial_printf(&SerialA, "%.2f %.2f %.2f | %.2f %.2f %.2f | %.2f %.2f %.2f | %.2f %.2f %.2f
\n\r",printtheta1motor*180/PI,printtheta2motor*180/PI,printtheta3motor*180/PI, x,y,z,
motortheta1*180/PI, motortheta2*180/PI, motortheta3*180/PI, theta1*180/PI, theta2*180/PI,
theta3*180/PI);
}
```