UNIVERSIDADE FEDERAL DE SANTA CATARINA - UFSC CAMPUS BLUMENAU

Departamento de Engenharia

Professor: Leonardo Mejia Rincon

Trabalho 2 de Introdução à Robótica Industrial

Carlos Eduardo dos Santos Junior
Marco Antônio Priotto Tonon

Blumenau, Novembro de 2019

Sumário

1. Apresentação do Tema

Neste presente trabalho, iremos realizar cálculos, simular e criar modelos para o robô Scara demonstrado na Figura 1 demonstrada abaixo, em robótica utilizamos o conceito de Jacobiana como ferramenta para calcular modelos de nosso manipulador, tanto o modelo estático que nos trará relação entre torques e forças como o modelo variacional que nos trará uma relação entre velocidades instantâneas das garras em relação às velocidades angulares das juntas, para isso utilizaremos a ferramenta da Jacobiana e conhecimentos previamentes estudados como os parâmetros de Denavit Hartenberg.

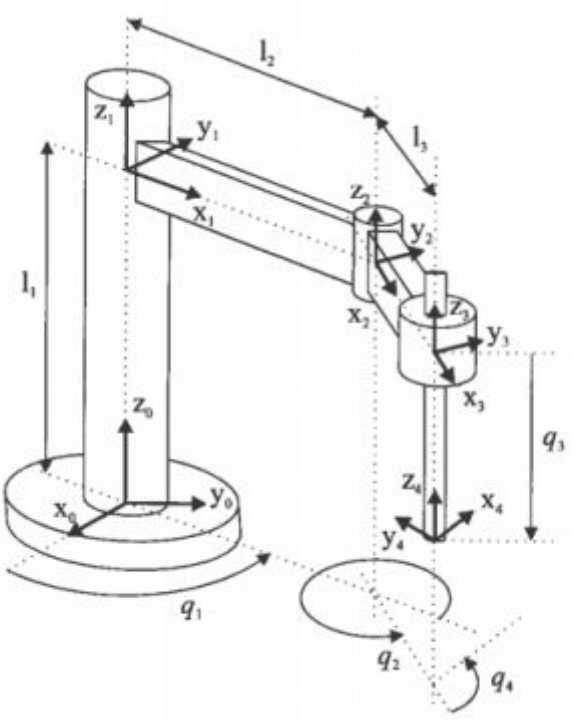

Figura 1: Robô SCARA.

Após a primeira etapa, iremos fazer com que nosso manipulador escreva a palavra “UFSC” e assim simular em MatLAB seu funcionamento com vídeo, vale ressaltar que toda a documentação e arquivos deste trabalho se encontrará no repositório do github <https://github.com/juninhocb/>. Após esta etapa iremos utilizar os conceitos de programação em robôs industriais e o Software Kuka Sim.Pro para simular o braço empilhando vários blocos em cima de um pallet.

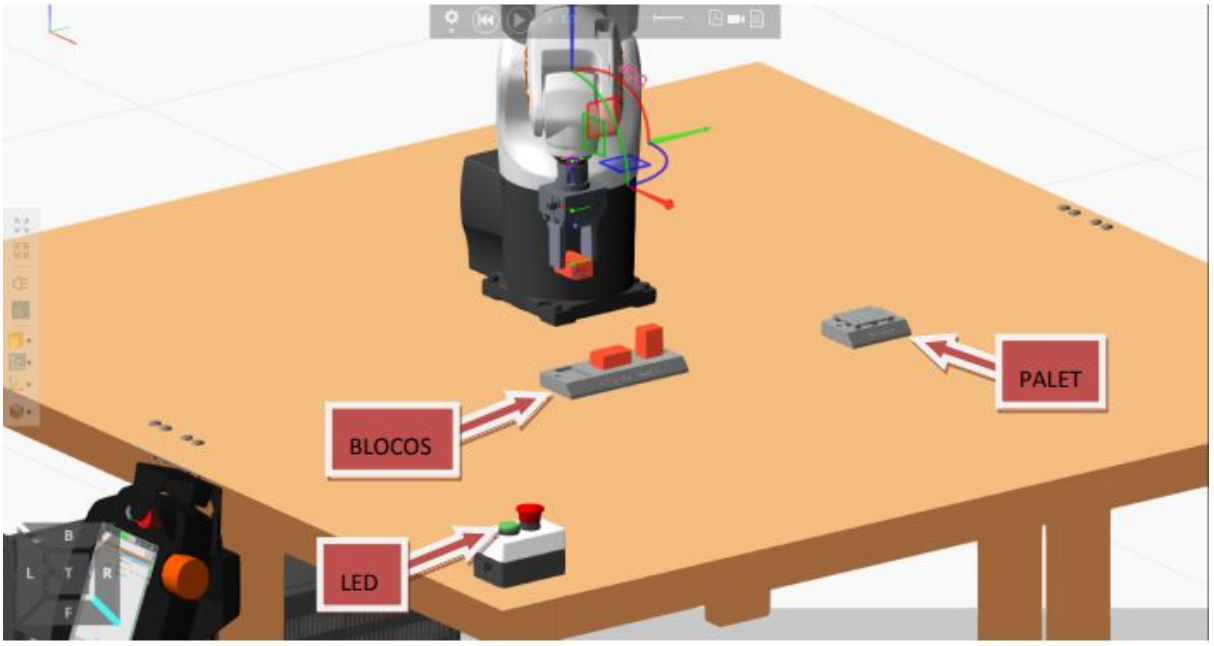

Figura 2: Área de Trabalho Kuka Sim Pró.

2. Modelo Estático Robô Scara

Primeiramente, iremos calcular os parâmetros de Denavit Hartenberg para o nosso manipulador demonstrado na Figura 1 e seus eixos de coordenadas, lembrando que fizemos algumas convenções que são, q3(da Figura 1) = L3 e q4(da Figura 1) = q3, Segue-se que:

	θ	d	a	α
OT1	q1 + 90°	L1	0	0
1T2	0	0	L2	0
2T3	q2	0	L3	0
3T4	q3	-L4	0	180°

Figura 3: Parâmetros de D-H.

Antes de prosseguirmos com os cálculos, utilizamos a biblioteca do Peter Corke para plotar nosso manipulador com os parâmetros encontrados.

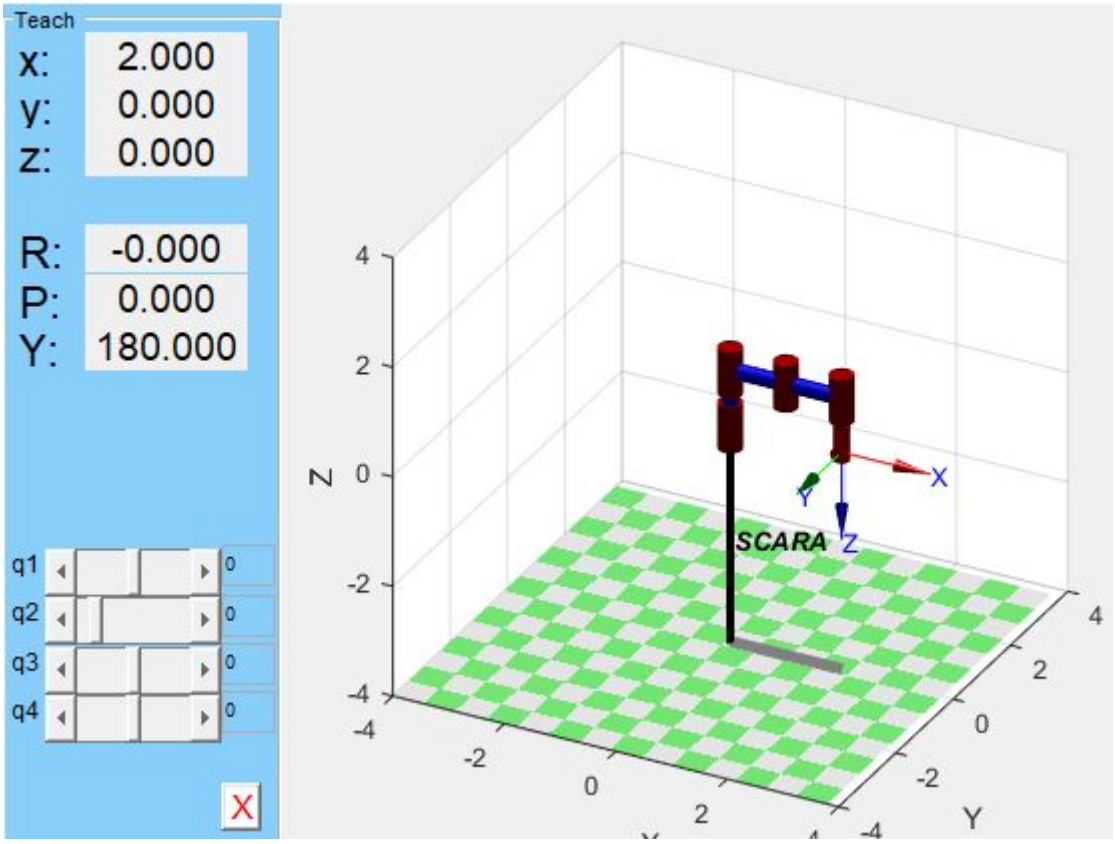

Figura 4: Representação Manipulador SCARA em MatLAB.

Note a semelhança com nosso robô representado na Figura 1.

```
SCARA:: 4 axis, RRRR, stdDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|         10|          0|          0|          0|
|  2|         q2|          0|          1|          0|          0|
|  3|         q3|          0|         10|          0|          0|
|  4|         q4|        -15|          0|    3.14159|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

Figura 5: Parâmetros em MatLAB.

O MatLAB por default instância os valores de theta que são os ‘q1’, ‘q2’ e etc, na figura ele coloca esse ‘ q1’ na junta 2, mas na prática, quando plotamos a Figura 4, e abrimos o mecanismo de rotação, vemos que q2 não se move, ou seja é sempre 0, como previmos anteriormente, isso nos demonstra que nossos parâmetros obedece a representação do SCARA no MatLAB, nota-se também que o X do efetuador(mostrado na figura) está de acordo com o X anterior de nossa representação de eixos, e Z do efetuador (mostrado na figura) está 180° de diferença do nosso eixo de Z de coordenadas, nota-se

também que quando rotacionamos o ângulo q4 ele apenas muda a orientação do efetuador mas não influencia na posição do sistema.

Utilizando-se da convenção de DH descrita abaixo:

```
DH =

[ cos(thetai), -cos(alfai)*sin(thetai),  sin(alfai)*sin(thetai), ai*cos(thetai)]
[ sin(thetai),  cos(alfai)*cos(thetai), -sin(alfai)*cos(thetai), ai*sin(thetai)]
[           0,              sin(alfai),              cos(alfai),             di]
[           0,                       0,                       0,              1]
```

Figura 6: Convenção de Denavit Hartenberg.

A partir daqui, iremos considerar nossos valores de Q1,Q2 e Q4 como Thetas(1,2,e 3) assim como Q3 que se tornou L4 e iremos calcular todas as matrizes de nosso sistema, dada a fórmula:

$$^{0}H_{4} = {}^{1}H_{4} \; x \, {}^{2}H_{4} \; x \, {}^{3}H_{4}$$

Temos então:

```
H01 =

[ cos(theta1 + 90), -sin(theta1 + 90), 0,  0]
[ sin(theta1 + 90),  cos(theta1 + 90), 0,  0]
[                0,                 0, 1, L1]
[                0,                 0, 0,  1]
```

Figura 7: H01.

```
H12 =

[ 1, 0, 0, L2]
[ 0, 1, 0,  0]
[ 0, 0, 1,  0]
[ 0, 0, 0,  1]
```

Figura 8: H12.

```
H23 =

[ cos(theta2), -sin(theta2), 0, L3*cos(theta2)]
[ sin(theta2),  cos(theta2), 0, L3*sin(theta2)]
[           0,            0, 1,             0]
[           0,            0, 0,             1]
```

Figura 9: H23.

```
H34 =

[ cos(theta4),  sin(theta4),  0,       0]
[ sin(theta4), -cos(theta4),  0,       0]
[           0,            0, -1, -theta3]
[           0,            0,  0,       1]
```

Figura 10: H34.

Fazendo as devidas multiplicações iremos ficar com as seguintes matrizes:

```
H01 =

[ cos(theta1 + 90), -sin(theta1 + 90), 0,  0]
[ sin(theta1 + 90),  cos(theta1 + 90), 0,  0]
[                0,                 0, 1, L1]
[                0,                 0, 0,  1]
```

Figura 11: H01(2).

```
H02 =

[ cos(theta1 + 90), -sin(theta1 + 90), 0, L2*cos(theta1 + 90)]
[ sin(theta1 + 90),  cos(theta1 + 90), 0, L2*sin(theta1 + 90)]
[                0,                 0, 1,                 L1]
[                0,                 0, 0,                  1]
```

Figura 12: H02.

```
H03 =

[ cos(theta1 + 90)*cos(theta2) - sin(theta1 + 90)*sin(theta2), - cos(theta1 + 90)*sin(theta2) - sin(theta1 + 90)*cos(theta2),
[ cos(theta1 + 90)*sin(theta2) + sin(theta1 + 90)*cos(theta2),   cos(theta1 + 90)*cos(theta2) - sin(theta1 + 90)*sin(theta2),
[                                                           0,                                                           0,
[                                                           0,                                                           0,
```

Figura 13: H03.

```
0, L2*cos(theta1 + 90) + L3*cos(theta1 + 90)*cos(theta2) - L3*sin(theta1 + 90)*sin(theta2)]
0, L2*sin(theta1 + 90) + L3*cos(theta1 + 90)*sin(theta2) + L3*sin(theta1 + 90)*cos(theta2)]
1,                                                                                      L1]
0,                                                                                       1]
```

Figura 14: H03 - Colunas 3 e 4

Como apenas iremos precisar das 2 últimas colunas da matriz 0H4:

H04 =

```
  0, L2*cos(theta1 + 90) + L3*cos(theta1 + 90)*cos(theta2) - L3*sin(theta1 + 90)*sin(theta2)]
  0, L2*sin(theta1 + 90) + L3*cos(theta1 + 90)*sin(theta2) + L3*sin(theta1 + 90)*cos(theta2)]
 -1,                                                                          L1 - theta3]
  0,                                                                                    1]
```

Figura 15: H04 - Colunas 3 e 4

Agora que já temos as matrizes de transformações, podemos calcular nossa Jacobiana.

2.1 - Jacobiana

De posse das matrizes demonstradas em 2, iremos calcular o Jacobiano, o Jacobiano consiste em uma matriz de dimensões 6 x N, onde N são os números de juntas que iremos considerar.

Para obter as 3 primeiras linhas do Jacobiano, precisaremos dos pontos Px,Py e Pz, onde os mesmos são obtidos da Matriz $^{0}H_{4}$, sendo:

Px = Linha 1 Coluna 4

Py = Linha 2 Coluna 4

Pz = Linha 3 Coluna 4

De posse dos pontos, iremos calcular a derivada em relação a cada theta(para revolução) e deslocamento(para prismática), como calcularemos para qualquer Jacobiana padrão.

$$\mathbf{J_G(a)} = \begin{bmatrix} \frac{\partial G_1}{\partial x_1}(\mathbf{a}) & \cdots & \frac{\partial G_1}{\partial x_n}(\mathbf{a}) \\ \vdots & \ddots & \vdots \\ \frac{\partial G_n}{\partial x_1}(\mathbf{a}) & \cdots & \frac{\partial G_n}{\partial x_n}(\mathbf{a}) \end{bmatrix}$$

Figura 16: Matriz Jacobiana.

Agora, precisaremos dos pontos que irão contemplar as 3 linhas subsequentes de nossa matriz Jacobiana, iremos denominar para eles Rx,Ry, Rw e Rz:

Onde:

Rx = Coluna 3 , exceto o ponto de Linha 4 e Coluna 3 da Matriz $^{0}H_{1}$

Ry = Será um vetor de zeros contendo 3 linhas dado por: [0 0 0], isso ocorre, pois trata-se de uma junta prismática.

Rw = Será a Coluna 3, exceto o ponto de Linha 4 e Coluna 3 da Matriz $^{0}H_{3}$

Rz = Será a Coluna 3, exceto o ponto de Linha 4 e Coluna 3 da Matriz $^{0}H_{4}$

Podemos comparar com o resultado do MatLAB e ver que os resultados bateram, apenas tivemos que separar em duas partes para que caiba no relatório, pois a matriz era demasiadamente extensa. Dessa forma temos abaixo a Coluna 1 da Matriz Jacobiana e logo após as Colunas 2 e 3 restantes:

```
J =

[ - L2*sin(theta1 + 90) - L3*cos(theta1 + 90)*sin(theta2) - L3*sin(theta1 + 90)*cos(theta2),
[   L2*cos(theta1 + 90) + L3*cos(theta1 + 90)*cos(theta2) - L3*sin(theta1 + 90)*sin(theta2),
[                                                                                         0,
[                                                                                         0,
[                                                                                         0,
[                                                                                         1,
```

Figura 17: Matriz Jacobiana.

```
- L3*cos(theta1 + 90)*sin(theta2) - L3*sin(theta1 + 90)*cos(theta2),  0]
  L3*cos(theta1 + 90)*cos(theta2) - L3*sin(theta1 + 90)*sin(theta2),  0]
                                                                   0, -1]
                                                                   0,  0]
                                                                   0,  0]
                                                                   1,  1]
```

Figura 18: Matriz Jacobiana - Colunas 2 e 3.

Como a última junta apenas se move na orientação do manipulador, podemos simplificar a última coluna da Matriz Jacobiana.

2.2 - Cinemática estática

De posse da Jacobiana, podemos calcular a cinemática estática de nosso robô SCARA, consideramos para tal a equação que descreve uma relação entre torques e forças utilizando-se do Jacobiano transposto, essa equação é dada por:

$$\tau = J^t \, x \, F$$

Esse modelo nos revela os torques das juntas em relação a forças e momentos da carga considerada. O próximo passo agora, é obter nossa Jacobiana transposta, que será mostrada na Figura abaixo, da mesma forma feita anteriormente, separamos a Matriz Jacobiana Transposta em 3, uma parte para cada coluna, como se trata de uma matriz 3x6, obtemos o seguinte:

```
x =

[ - sin(conj(theta1) + 90)*conj(L2) - cos(conj(theta2))*cos(conj(theta1) + 90)*conj(L3) - cos(conj(theta2))*sin(conj(theta1) + 90)*conj(L3)
[                                     sin(conj(theta2))*sin(conj(theta1) + 90)*conj(L3) - sin(conj(theta2))*cos(conj(theta1) + 90)*conj(L3)
[                                                                                                                                        0
```

Figura 19: Matriz Jacobiana Transposta - Coluna 1.

```
cos(conj(theta1) + 90)*conj(L2) + cos(conj(theta2))*cos(conj(theta1) + 90)*conj(L3) - cos(conj(theta2))*sin(conj(theta1) + 90)*conj(L3),
                                - sin(conj(theta2))*sin(conj(theta1) + 90)*conj(L3) - sin(conj(theta2))*cos(conj(theta1) + 90)*conj(L3),
                                                                                                                                     0,
```

Figura 20: Matriz Jacobiana Transposta - Coluna 2.

```
.  0, 0, 0, 1]
.  0, 0, 0, 1]
. -1, 0, 0, 1]
```

Figura 21: Jacobiana Transposta - Coluna 3.

Temos também a matriz a qual denominamos de “Y”, qual se trata da matriz que representa os torques a serem aplicados: T1, T2 e T3. Como podemos ver a seguir:

```
y =

 T1
 T2
 T3
```

Figura 22: Matriz Y - Representando os torques.

E por último, temos a matriz “Z”, a qual representa as Forças e Momentos que serão multiplicadas pelo Jacobiano Transposto, como segue abaixo:

```
z =

 F1
 F2
 F3
 M1
 M2
 M3
```

Figura 23: Matriz Z - Representando as Forças e Momentos.

Como Y = X * Z, obtemos a cinemática estática de nosso manipulador.

2.3 - Pontos de singularidade do manipulador

Para calcular os pontos de singularidade do manipulador, procuramos igualar as três primeiras linhas da Matriz Jacobiana à zero, dessa forma temos para a primeira linha:

$$
\begin{aligned}
J_{1x1} &= -L_2 sen(\theta_1 + 90) - L_3 sen(\theta_1 + 90) cos\theta_2 - L_3 cos(\theta_1 + 90) sen\theta_2 = 0 \\
J_{1x2} &= -L_3 cos(\theta_1 + 90) sen\theta_2 - L_3 sen(\theta_1 + 90) cos\theta_2 = 0 \\
J_{1x3} &= 0 = 0
\end{aligned}
$$

Para a segunda linha ficamos com o seguinte:

$$
\begin{aligned}
J_{2x1} &= L_2 cos(\theta_1 + 90) - L_3 sen(\theta_1 + 90) sen\theta_2 + L_3 cos(\theta_1 + 90) cos\theta_2 = 0 \\
J_{2x2} &= L_3 cos(\theta_1 + 90) cos\theta_2 - L_3 sen(\theta_1 + 90) sen\theta_2 = 0 \\
J_{2x3} &= 0 = 0
\end{aligned}
$$

E finalmente para a terceira linha, temos:

$$
\begin{aligned}
J_{3x1} &= 0 \\
J_{3x2} &= 0 \\
J_{3x3} &= -1
\end{aligned}
$$

Desse modo, fazendo os cálculos à mão, vemos claramente que utilizando $\theta_1 = 0$ e $\theta_2 = 90$, conseguimos “zerar” os elementos desejados (desconsideramos os demais ângulos “thetas” por não aparecem implicitamente nas equações acima).

Assim, observamos claramente que os valores de ângulos citados acima são uma singularidade do manipulador.

3. Elaboração da trajetória

Para este trabalho, utilizamos aprendemos vários conceitos em sala de aula que relacionam as velocidades e as trajetórias do nosso manipulador, sendo dois tipos de equações mais comuns a linear que é dada por y = ax + t, onde a é o nosso coeficiente, y é o nosso ângulo em função do tempo, a partir de velocidades conhecidas, encontramos valores para nossa equação que descreve o movimento, o segundo de tipo de equação é a equação cúbica dada por y = a0 + a1t + a2t^2 + a3t^3, onde usando a mesma ideia, iremos obter os valores para os coeficientes a0, a1, a2 e a3.

Neste trabalho utilizamos a biblioteca do Peter Corke para resolver estes problemas de simulação, porém tivemos que estudar uma biblioteca que é muito útil, quando se trata de robótica.

3.1 - Escrita “UFSC”

Primeiramente, utilizamos as funções fornecidas pelo Professor do laboratório 8, para conseguir realizar os cálculos dos ângulos que iremos utilizar neste projeto da escrita UFSC, em uma folha, desenhamos UFSC e escolhemos pontos para tal desenho, como demonstra a imagem abaixo.

Utilizamos o código general para executar, e o código chama a função objetivo, na função objetivo colocamos os pontos que serão mostrados abaixo, cada ponto, nos entregavam ângulos em graus e observamos também que alguns pontos não estavam em alcance de nosso manipulador, notamos isto também quando rodamos o código general a variável bestval, ao qual esperávamos um valor muito próximo a zero.

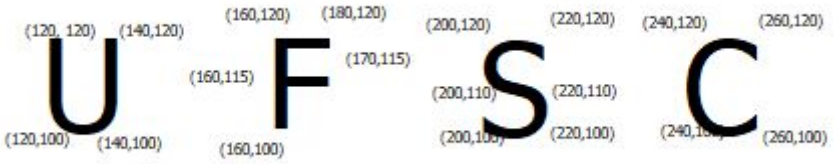

Figura 24: Convenção de pontos para escrita UFSC.

Logo após isto, plotamos todos estes pontos descritos na imagem acima para confirmar que este pontos iria fazer tal desenho, assim como obter os ângulos que iremos utilizar na biblioteca para o SCARA fazer a escrita, obtivemos então o seguinte resultado para os pontos escolhidos da escrita.

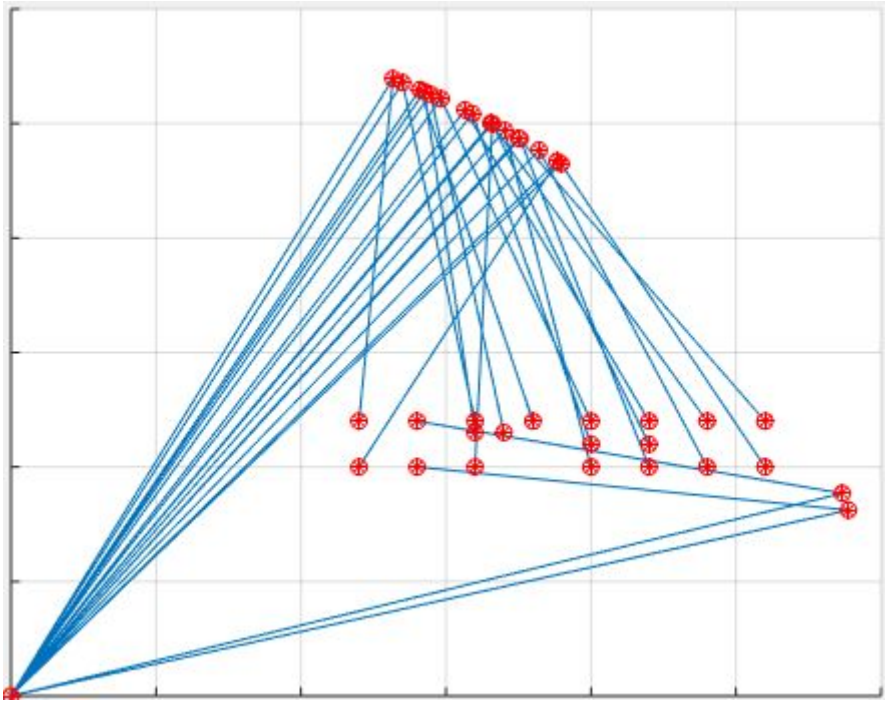

Figura 25: Pontos plotados no MatLAB para utilizar os melhores ângulos.

Ligando as juntas em linha reta do nosso efetuador, iremos perceber a escrita UFSC, do tipo quadrada e após isso, fizemos um novo plot de nosso SCARA para ter uma dimensão do quão grande seria o braço do manipulador em relação aos pontos que achamos, obtendo a imagem mostra abaixo.

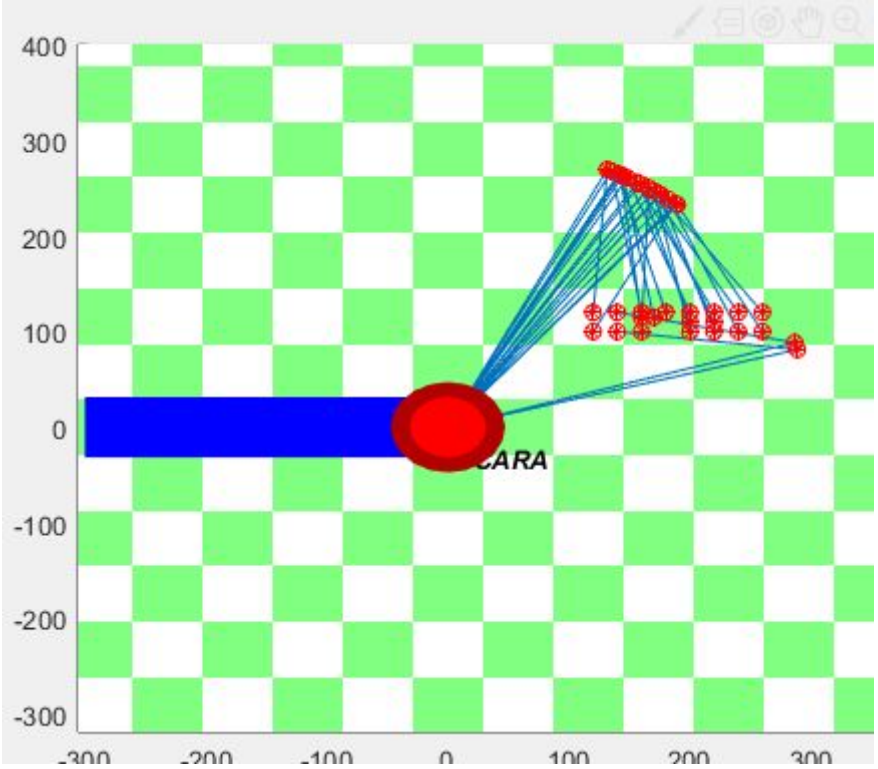

Figura 26: Tamanho dos pontos em relação ao SCARA.

Pela imagem, percebe-se que o tamanho do SCARA é bastante grande em relação aos pontos desenhados, sendo um pouco imperceptível de sua escrita, porém fizemos vários teste para verificar as letras e ele realmente executa conforme o planejado, sendo assim, tentamos arrumar essa diferença de tamanho, porém sem sucesso, talvez apenas trocando os pontos utilizados para este desenho.

A idéia era que a cada iteração (exceto onde o robô passa de letra para letra) o robô mantivesse o plot, tentamos isso e lemos várias coisas sobre, mas sem nenhuma eficácia, portanto, infelizmente não conseguimos manter o que estava sendo escrito para visualizar os resultados depois.

3.2 - Cálculos de Torques

O nosso manipulador se movimenta apenas nos eixos X e Y, quando obtivemos os ângulos theta1 e theta2 para a trajetória do manipulador, estavamos sempre assumindo que o mesmo se movimenta apenas nestes eixos, então, o que acontece neste caso é de que o “chão” irá sempre responder de forma constante com em 1 newton a força do nosso manipulador, vale afirmar também que nossa escrita gera uma força apenas no eixo Z, sendo este, constante durante todo o movimento.

4. Programação da Kuka

Como nos foi pedido, elaboramos um programa no software KUKA.Sim PRO de forma que empilhava os três blocos postos em cima da mesa conforme simulação. Abaixo segue a imagem da simulação após elaborada a tarefa.

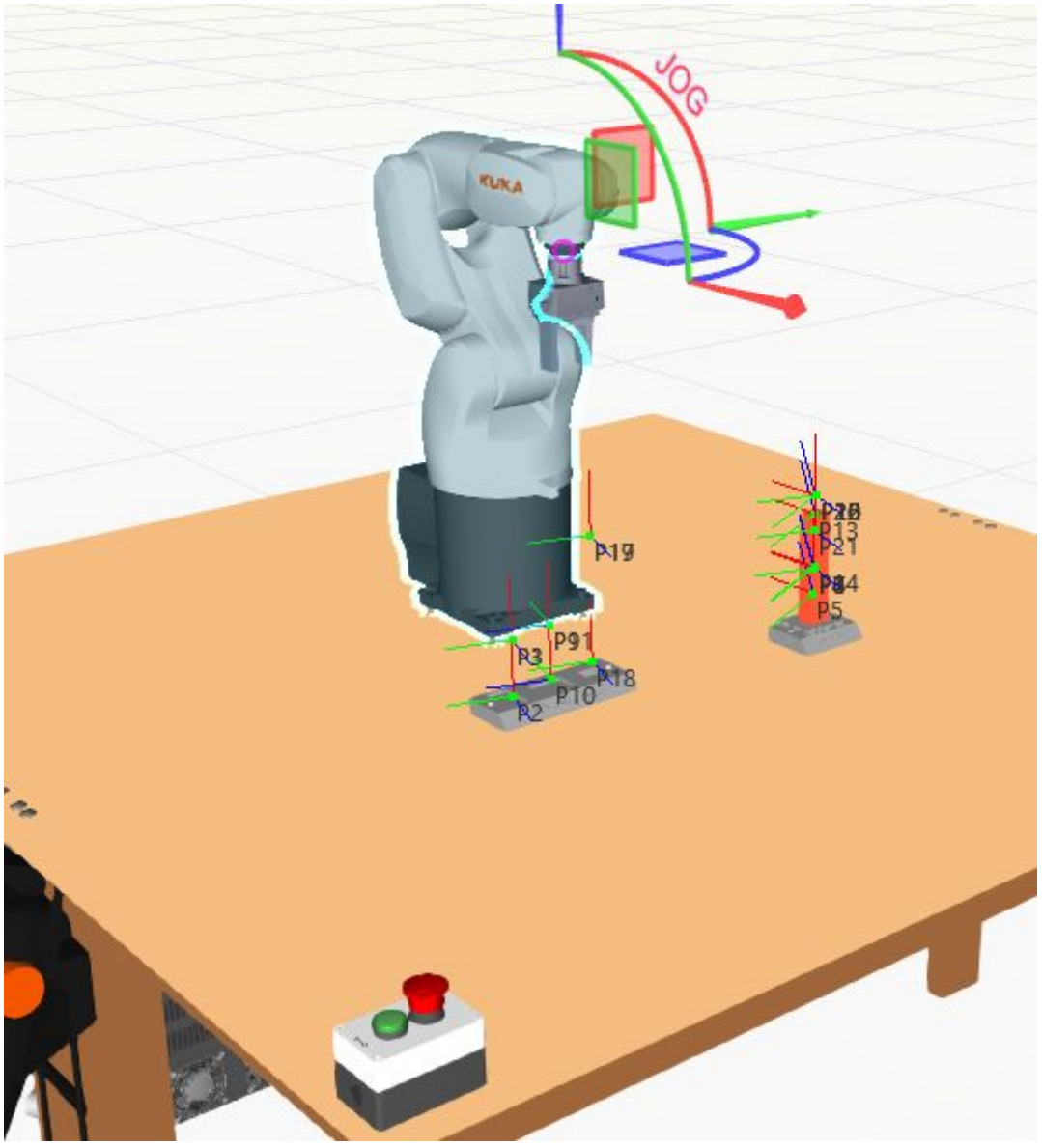

Figura 27: Simulador Kuka.

Conseguimos também fazer com que o LED verde (na base da mesa) piscasse por três vezes para sinalizar o começo da rotina e repetisse esse ato (piscar o LED) por mais três vezes ao final da rotina.

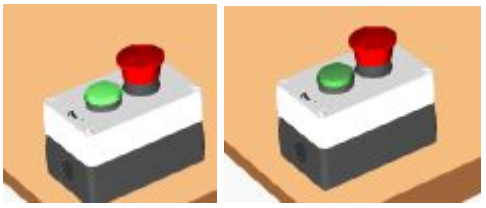

Figura 28: Representação do LED (Aceso/Apagado)

Basicamente os comandos utilizados nesta parte foram os mesmos aprendidos em laboratório referente à prática deste conteúdo:

LIN - Elabora uma trajetória linear.

OUT - Seguida por um número (referente a saída desejada), significa se queremos nível lógico TRUE ou FALSE para determinada saída do manipulador, por exemplo, “OUT 10 = TRUE” se refere ao LED, “OUT 8” se refere à Garra Fechada e “OUT 9” se refere à abertura.

PTP - Também referente à trajetória, elabora a movimentação do manipulador entre um ponto inicial e um ponto final.

WAIT - Como o próprio nome já diz, em tradução, significa uma espera em segundos dependendo do número desejado que acompanha o comando.

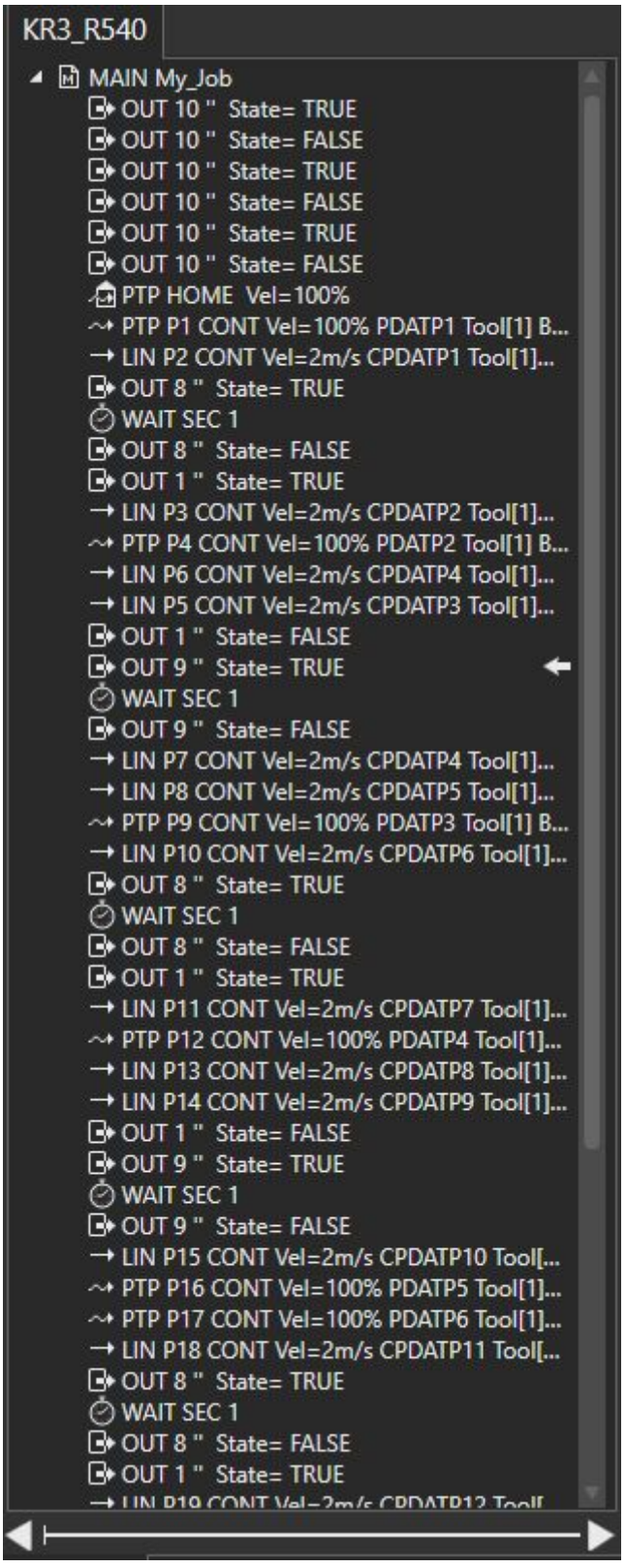

Figura 29: Código para elaboração da tarefa acima

Acima aqui, podemos ver em figura o código de programação feito pelo grupo e utilizado no programa, devido a captura de imagem, para caber no relatório, ficou faltando uma pequena parte após o comando “OUT 10”, porém consideramos que seria apenas uma

etapa repetitiva, sendo o que foi mostrado já cabível para demonstração de todos os comandos utilizados para elaboração do empilhamento dos blocos.

4.2 - Resultados

Caso o leitor queira ver o resultado da simulação feita por nosso código, o mesmo deverá abrir o repositório do Github e ter instalado em seu computador o Kuka Sim Pró, após baixar o repositório de Robótica Industrial, basta apenas rodar o programa e apertar no botão play.

5. Conclusão

Este trabalho serviu de muito grande valia ao nosso grupo, pois com ele pudemos observar na prática toda a teoria vista em aulas, principalmente no que tange à cinemática de manipuladores, tivemos a oportunidade de realizar os cálculos de cinemática e utilizá-los para elaborar uma função com o manipulador, no caso a escrita “UFSC”.

Quanto ao processo de escrita citado acima, ressaltamos que ficou demasiadamente pequeno, porém observando a pontuação pode-se observar a movimentação pedida, que segundo nossos testes que fizemos com letra ele estará executando o caminho correto.

Foi também muito importante para o grupo aprender a resolver determinados imprevistos, dúvidas, tivemos muitas vezes que recorrer à recursos externos e buscar cada vez mais conhecimento para realizar todas as atividades propostas, fato este que evoluiu nosso aprendizado.

Outro ponto que merece destaque e foi muito bacana de se realizar, é a programação utilizando a biblioteca do peter corke que foi algo que sempre buscamos para facilitar nossa vida, além do documento do Matlab, vimos também vários vídeos no Youtube e códigos prontos utilizando a biblioteca, fato que também nos ajudou a melhorar nossa destreza com a lib que serviu de grande aprendizado e sensação de “dever cumprido” ao ver todos os códigos funcionando, no qual ainda sabemos que os mesmos podem ser melhorados e otimizados.

6. Referências

[1] ALGORITHMS. Devec3, Dísponível em: <http://www.icsi.berkeley.edu/~storn/code.html>. Acesso em: 25 de nov. de 2019.

[2] Corke, Peter. Robotics, Vision and Control: Fundamental Algorithms in MATLAB. Springer Verlag NY, 1, 2011. ISBN: 3642201431

[3] MATLAB DOCUMENTATION. MatLAB: help, c2019. Disponível em: <https://www.mathworks.com/help/matlab/>. Último Acesso em: 28 de nov. de 2019.