

---

**UNIVERSIDADE FEDERAL DE ALAGOAS**  
**INSTITUTO DE COMPUTAÇÃO**  
**CURSO DE ENGENHARIA DE COMPUTAÇÃO**

**Robótica  
PROJETO SCARA**

Lilian Giselly Pereira Santos : **19111115**

Maria Fernanda Herculano M. da Silva: **19111118**

Pedro Henrique de Brito Nascimento : **19111287**

Maceió - AL

2022

Lilian Giselly Pereira Santos : *19111115*

Maria Fernanda Herculano M. da Silva: *19111118*

Pedro Henrique de Brito Nascimento : *19111287*

## **PROJETO SCARA**

*Atividade avaliativa apresentada à disciplina de Robótica, correspondente à segunda avaliação do semestre 2022.2 do curso de Engenharia de Computação da Universidade Federal de Alagoas, sob orientação do Prof. Ícaro Bezerra Queiroz de Araújo.*

Alagoas - AL

2022

---

# 1 Descrição do Projeto

A cinemática inversa do robô SCARA é utilizada para calcular os ângulos das juntas necessários para posicionar o efetuador final na posição desejada. Nessa atividade, iremos implementar a cinemática inversa de um robô SCARA no simulador Webots utilizando sua API para python.

## 1.1 Requisitos

- Simulador Webots (<https://cyberbotics.com/download>)
- Python instalado com numpy (<https://numpy.org/install/>)
- Pasta zipada do projeto scara.zip

## 1.2 Ambiente de simulação

Abra o arquivo scara/worlds/world.wbt no simulador. Selecione Tools > Preferences... > General e certifique-se que o campo Python command está setado para python (ou o caminho do python alternativo se você estiver utilizando Anaconda ou virtualenv). Em seguida abra o arquivo *scara/controllers/scara\_controller/scara\_controller.py* no editor de texto integrado do webots (selecione Edit controller, na árvore de objetos) ou abra num editor de sua preferência.

## 1.3 Objetivos

Seu objetivo principal é simular uma atividade do tipo pick and place utilizando a cena disponibilizada. O robô scara deve pegar o objeto (pato de borracha) e colocá-lo em cima da caixa de papelão (conforme a figura 1). Para isso você deverá utilizar a API do simulador em python para guardar os valores de posição necessários e implementar a função de cinemática inversa do manipulador. Ao iniciar a simulação o controlador irá abortar a execução levantando um erro, uma vez que o código está incompleto. Seu objetivo é completar as seções de código marcadas com o comentário TODO para que seja possível executar a tarefa.

## 1.4 Entendendo o robô SCARA

Antes de começar a implementar a cinemática inversa, é importante entender estrutura cinemática do robô. Um SCARA (do inglês Selective Compliant Articulated Robot Arm) é um robô de quadro eixos consistindo de 3 juntas rotacionais e 1 prismática. Sabendo disso, construa a tabela de Denavit-Hartenberg para essa cadeia cinemática e implemente a função fkine. Você pode utilizar a figura 2 como base para atribuição dos frames de referência e a 3 como referência das medidas em metros.

## 1.5 Derivando as expressões de cinemática inversa

Tendo em posse a cinemática direta, implemente a cinemática inversa. Para isso você vai assumir que receberá a pose do pato  $p = [x, y, z, \phi]$  em relação ao frame F0 de referência global (coincide com a base do robô). Sua função deve retornar os valores das juntas  $q = [q_1, q_2, q_3, q_4]$  que levam o efetuador final à pose p desejada.

## 1.6 Programando o controlador

No loop principal do programa você deve utilizar as funções implementadas para realizar a tarefa de pick and place. Dica: utilize a função scara.delay para aguardar um pouco entre chamadas da função scara.setPositions, uma vez que o sistema simulado possui dinâmica.

---

# 2 Cinemática Direta

O objetivo de calcular a cinemática direta é encontrar a pose do robô dado um conjunto posições das juntas. De inicio, foi construída a tabela de Denavit-Hartenberg de acordo com a tabela abaixo:

| $\theta$   | d     | a   | $\alpha$ | offset |
|------------|-------|-----|----------|--------|
| $\theta_1$ | 0.65  | 0.5 | 0        | 0      |
| $\theta_2$ | 0.1   | 0.5 | 0        | 0      |
| $\theta_3$ | 0     | 0   | $\pi$    | 0      |
| 0          | $d_4$ | 0   | 0        | 0.225  |

A visualização do SCARA foi possível dada a Robotics toolbox (Peter Corke), de acordo com as figuras abaixo.

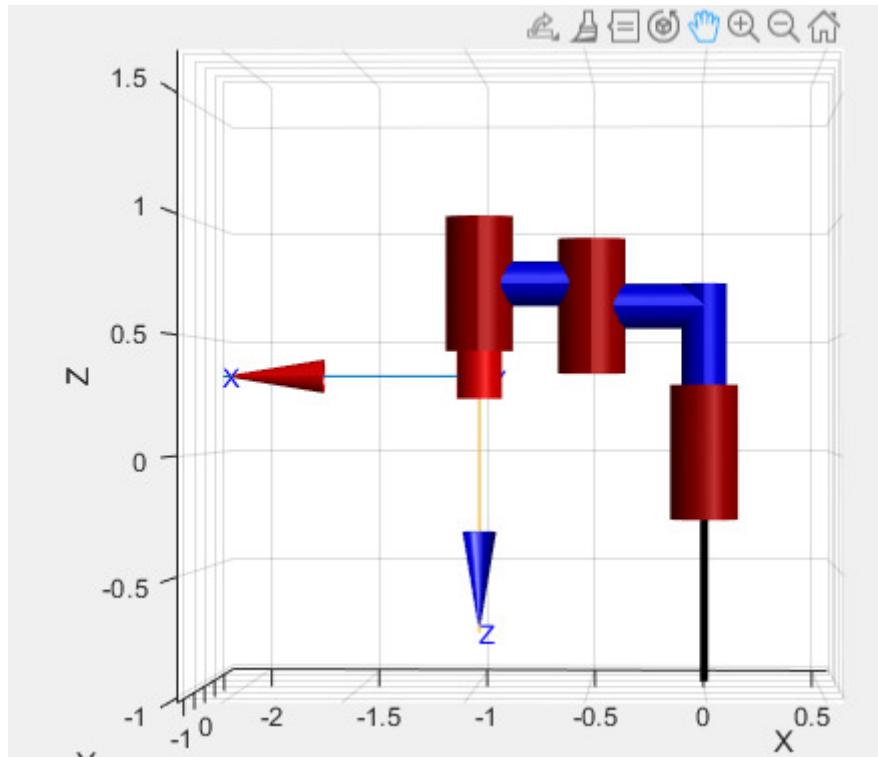


Figura 1: Robotics Toolbox - SCARA

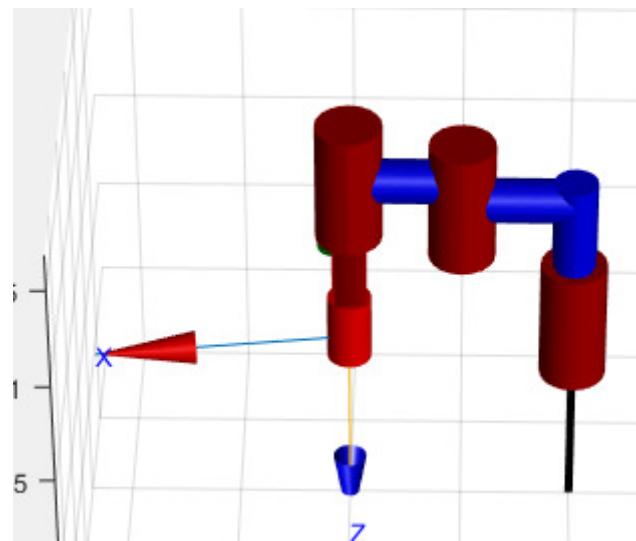


Figura 2: Robotics Toolbox - SCARA (Perspectiva)

---

Levando em consideração os frames inicial ao final, os parâmetros DH resultaram na seguinte matriz de transformação homogênea:

$$T_0^4 = \begin{bmatrix} \cos(\theta_1 + \theta_2 + \theta_3) & \sin(\theta_1 + \theta_2 + \theta_3) & 0 & 0.5 \cos(\theta_1) + 0.5 \cos(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2 + \theta_3) & -\cos(\theta_1 + \theta_2 + \theta_3) & 0 & 0.5 \sin(\theta_1) + 0.5 \sin(\theta_1 + \theta_2) \\ 0 & 0 & -1 & 0.525 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 3: Matriz de Transformação Homogênea - (Forward Kinematics)

O código em Python da implementação da função da cinemática direta está a seguir. Recebe como parâmetro uma lista de 4 floats correspondente aos três ângulos das juntas rotacionais e um deslocamento da junta prismática, e retorna a pose da garra através de uma matriz de transformação 4x4.

```

1 def fkine(q):
2     mat = [
3         [c(q[0] + q[1] + q[2]), s(q[0] + q[1] + q[2]), 0, 0.5*c(q[0]) + 0.5*c(q
4             [0] + q[1])],
5         [s(q[0] + q[1] + q[2]), -c(q[0] + q[1] + q[2]), 0, 0.5*s(q[0]) + 0.5*s(q
6             [0] + q[1])],
7         [0, 0, -1, 0.525 - q[3]],
8         [0, 0, 0, 1]]
9
10    return mat

```

### 3 Cinemática Inversa

O objetivo de calcular a cinemática inversa é saber quais os posicionamentos das juntas dado um pose específico do efetuador ( $x$ ,  $y$ ,  $z$  e  $\phi$ ). A solução para encontrar os ângulos desejados é igualar a matriz de transformação calculada na cinemática direta com uma matriz de transformação do frame da base do robô diretamente para o efetuador, no qual existe uma translação em  $x$ ,  $y$  e  $z$ , além de uma rotação em  $X$  de  $\pi$  seguida de uma rotação em  $Z$  de  $\phi$ .

Para esta solução, será necessário inicialmente calcular a matriz de transformação da base para o efetuado, dado por:

$$T_B^E(x, y, z, \phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & x \\ \sin(\phi) & \cos(\phi) & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\pi) & -\sin(\pi) & 0 \\ 0 & \sin(\pi) & \cos(\pi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 & x \\ \sin(\phi) & -\cos(\phi) & 0 & y \\ 0 & 0 & -1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Igualando esta matriz a calculada na cinemática direta, é possível obter uma solução algébrica para encontrar os ângulos das juntas desejados, como pode ser observado a seguir:

$$T_B^E(x, y, z, \phi) = T_B^E(\theta_1, \theta_2, \theta_3, d_4)$$

$$\begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 & x \\ \sin(\phi) & -\cos(\phi) & 0 & y \\ 0 & 0 & -1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1 + \theta_2 + \theta_3) & \sin(\theta_1 + \theta_2 + \theta_3) & 0 & 0.5\cos(\theta_1) + 0.5\cos(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2 + \theta_3) & -\cos(\theta_1 + \theta_2 + \theta_3) & 0 & 0.5\sin(\theta_1) + 0.5\sin(\theta_1 + \theta_2) \\ 0 & 0 & -1 & 0.525 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Igualando alguns termos representantes da rotação, temos que:

$$\begin{cases} \cos(\phi) = \cos(\theta_1 + \theta_2 + \theta_3) \\ \sin(\phi) = \sin(\theta_1 + \theta_2 + \theta_3) \end{cases}$$

$$\frac{\cos(\phi)}{\sin(\phi)} = \frac{\cos(\theta_1 + \theta_2 + \theta_3)}{\sin(\theta_1 + \theta_2 + \theta_3)}$$

$$\begin{aligned} \operatorname{tg}(\phi) &= \operatorname{tg}(\theta_1 + \theta_2 + \theta_3) \\ \phi &= \theta_1 + \theta_2 + \theta_3 \end{aligned}$$

Nos elementos das translações temos portanto as seguintes igualdades:

$$\begin{cases} x = 0.5\cos(\theta_1) + 0.5\cos(\theta_1 + \theta_2) \\ y = 0.5\sin(\theta_1) + 0.5\sin(\theta_1 + \theta_2) \\ z = 0.525 - d_4 \end{cases}$$

$$\begin{cases} x^2 = 0.25\cos^2(\theta_1) + 0.25\cos^2(\theta_1 + \theta_2) + 0.5\cos(\theta_1) \cdot \cos(\theta_1 + \theta_2) \\ y^2 = 0.25\sin^2(\theta_1) + 0.25\sin^2(\theta_1 + \theta_2) + 0.5\sin(\theta_1) \cdot \sin(\theta_1 + \theta_2) \\ z = 0.525 - d_4 \end{cases}$$

$$x^2 + y^2 = 0.25 \cdot [\cos^2(\theta_1) + \sin^2(\theta_1)] + 0.25 \cdot [\cos^2(\theta_1 + \theta_2) + \sin^2(\theta_1 + \theta_2)] + 0.5 \cdot [\cos(\theta_1) \cdot \cos(\theta_1 + \theta_2) + \sin(\theta_1) \cdot \sin(\theta_1 + \theta_2)]$$

$$x^2 + y^2 = 0.25 + 0.25 + 0.5 \cdot \{\cos(\theta_1) \cdot [\cos(\theta_1)\cos(\theta_2) - \sin(\theta_1)\sin(\theta_2)] + \sin(\theta_1) \cdot [\sin(\theta_1)\cos(\theta_2) + \sin(\theta_2)\cos(\theta_1)]\}$$

$$x^2 + y^2 = 0.5 + 0.5 \cdot \{\cos^2(\theta_1)\cos(\theta_2) - \cos(\theta_1)\sin(\theta_1)\sin(\theta_2) + \sin^2(\theta_1)\cos(\theta_2) + \sin(\theta_1)\sin(\theta_2)\cos(\theta_1)\}$$

$$x^2 + y^2 = 0.5 + 0.5\cos(\theta_2)$$

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - 0.5}{0.5}\right)$$

Voltando para as igualdades dos termos de translação considerando  $\theta_2$  como uma variável conhecida, podemos calcular as demais variáveis desejadas, como é demonstrado a seguir:

$$x = 0.5\cos(\theta_1)\cos(\theta_2) - 0.5\sin(\theta_1)\sin(\theta_2) + 0.5\cos(\theta_1)$$

$x = k_1\cos(\theta_1) - k_2\sin(\theta_1)$ , onde  $k_1 = 0.5\cos(\theta_2)$  e  $k_2 = 0.5\sin(\theta_2)$

$$y = 0.5\cos(\theta_1)\sin(\theta_2) + 0.5\sin(\theta_1)\cos(\theta_2) + 0.5\sin(\theta_1)$$

$$y = k_1\sin(\theta_1) + k_2\cos(\theta_1)$$

Considerando que  $\gamma = \text{Atan2}(k_2, k_1)$  e consequentemente  $r = \sqrt{k_1^2 + k_2^2}$  temos que:

$$\begin{cases} k_1 = r \cdot \cos(\gamma) \\ k_2 = r \cdot \sin(\gamma) \end{cases}$$

$$x = r \cdot \cos(\gamma) \cdot \cos(\theta_1) - r \cdot \sin(\gamma) \cdot \sin(\theta_1)$$

$$\frac{x}{r} = \cos(\gamma + \theta_1) \quad y = r \cdot \cos(\gamma) \cdot \sin(\theta_1) - r \cdot \sin(\gamma) \cdot \cos(\theta_1)$$

$$\frac{y}{r} = \sin(\gamma + \theta_1)$$

$$\gamma + \theta_1 = \text{Atan2}(y, x) \longrightarrow \theta_1 = \text{Atan2}(y, x) - \text{Atan2}(k_2, k_1)$$

Portanto, temos que:

$$\theta_3 = \phi - \theta_1 - \theta_2$$

$$d_4 = 0.525 - z$$

O código em Python da implementação da função da cinemática inversa está a seguir. Recebe como parâmetro a pose desejada da garra com os parâmetros previamente mencionados ( $x, y, z$  e  $\phi$ ) e retorna uma lista de 4 floats correspondente aos ângulos das juntas.

```

1 def invkine(x, y, z, phi, l1=0.5, l2=0.5, offset=0.525):
2     c2 = (x ** 2 + y ** 2 - l1 ** 2 - l2 ** 2) / (2 * l1 * l2)
3     if np.abs(c2) > 1:
4         print("Position out of reach!")
5         return False
6
7     s2 = np.sqrt(1 - c2 ** 2)
8     q2 = np.arctan2(s2, c2)
9
10    k1 = l1 + l2 * c2
11    k2 = l2 * s2
12
13    q1 = np.arctan2(y, x) - np.arctan2(k2, k1)
14    q3 = offset - z
15    q4 = phi - q1 - q2
16
17    return [q1, q2, q3, q4]

```

---

## 4 Controlador Pick and Place

Para que a simulação ocorra de maneira adequada, é necessário que um controlador pick and place seja implementado. No código a seguir está o código main do projeto, que faz a chamada de funções auxiliares para seu funcionamento. Em suma, esse trecho de código pega a posição do pato e realiza a cinemática inversa para direcionar a garra para aquela posição para enfim pegá-lo. Em seguida, faz o mesmo com a posição da caixa. Por fim, ele redireciona o robô para a posição inicial do pato.

```
1 if __name__ == "__main__":
2     scara = Scara()
3
4     # Main loop:
5     # Perform simulation steps until Webots is stopping the controller
6
7     finished = False
8     while scara.step() != -1:
9         new_pos, new_yaw, colliding = scara.is_colliding()
10
11     if(colliding == False and finished == False):
12         x, y, z, phi = scara.getDuckPose()
13         q1, q2, q3, q4 = invkine(x, y, z, phi, 11=0.5, 12=0.5, offset=0.525)
14         scara.set_position([q1, q2, q3, q4])
15         scara.delay(3000)
16         scara.hold()
17     elif(colliding == True and finished == False):
18         x2, y2, z2, phi2 = 0.85, -0.3, 0.6, -np.pi/2
19         q1, q2, q3, q4 = invkine(x2, y2, z2, phi2, 11=0.5, 12=0.5, offset=0.525)
20         scara.set_position([q1, q2, q3, q4])
21         finished = True
22         scara.delay(3000)
23         scara.release()
24     else:
25         q1, q2, q3, q4 = invkine(x, y, z, phi, 11=0.5, 12=0.5, offset=0.525)
26         scara.set_position([q1, q2, q3, q4])
27         scara.delay(2000)
```

Classe Scara que define e implementa funções auxiliares utilizadas no trecho de código acima:

```
1 class Scara:
2     def __init__(self):
3         # create the Robot instance.
4         self.robot = Supervisor()
5
6         # get the time step of the current world.
7         self.timestep = int(self.robot.getBasicTimeStep())
8
9         # get joint motor devices
10        self.joints = [self.robot.getDevice("joint%d" % i) for i in range(1, 5)]
11
12        # get duck reference
13        self.duck = self.robot.getFromDef("DUCK")
14
15        # get gripper reference
16        self.gripper = self.robot.getFromDef("GRIPPER")
17
18        self.grasp = False
19        self.grasp_prev = False
20
21    def set_position(self, q):
22        """
23            Set the joint positions of the SCARA robot.
24            Input: q - joint angles (list of 4 floats)
25        """
26        for joint, value in zip(self.joints, q):
27            joint.setPosition(value)
28
29    def is_colliding(self, ds=0.15):
30        """
31            Check if the gripper is colliding with the duck.
32            Input: ds - safety distance (float)
33            Output: new_pos - new gripper position (list of 3 floats)
34                new_yaw - new gripper yaw (float)
35                colliding - True if colliding, False otherwise (bool)
36
```

```

36     """
37     dp = np.array(self.duck.getPose()).reshape(4, 4)[-1, -1]
38     gt = np.array(self.gripper.getPose()).reshape(4, 4)
39     gp = gt[:-1, -1]
40     gy = np.arctan2(gt[1, 0], gt[0, 0])
41     return (
42         (gp + np.array([0.0, 0.0, -0.5 * ds])).tolist(),
43         gy,
44         (np.linalg.norm(dp - gp) < ds),
45     )
46
47     def step(self):
48         """
49             Perform one simulation step.
50             Output: -1 if Webots is stopping the controller, 0 otherwise (int)
51         """
52         if self.grasp is True and self.grasp_prev is False:
53             print("GRASP STARTED")
54             self.grasp_prev = True
55         elif self.grasp is False and self.grasp_prev is True:
56             print("GRASP ENDED")
57             self.grasp_prev = False
58
59         if self.grasp:
60             new_pos, new_yaw, colliding = self.is_colliding()
61             self.duck.resetPhysics()
62             if colliding:
63                 self.duck.getField("translation").setSFVec3f(new_pos)
64                 self.duck.getField("rotation").setSFRotation([0.0, 0.0, 1.0, new_yaw])
65
66         return self.robot.step(self.timestep)
67
68     def delay(self, ms):
69         """
70             Delay the simulation for a given time.
71             Input: ms - delay time in milliseconds (int)
72         """
73         counter = ms / self.timestep
74         while (counter > 0) and (self.step() != -1):
75             counter -= 1
76
77     def hold(self):
78         """
79             Hold the duck.
80         """
81         self.grasp = True
82
83     def release(self):
84         """
85             Release the duck.
86         """
87         self.grasp = False
88
89     def getDuckPose(self):
90         """
91             Get the duck pose.
92             Output: position - duck position (list of 3 floats)
93                     yaw - duck yaw (float)
94         """
95         pose = np.array(self.duck.getPose()).reshape(4, 4)
96         position = pose[:-1, -1].tolist()
97         yaw = np.arctan2(pose[1, 0], pose[0, 0])
98         return position + [yaw]

```

Na figura abaixo é possível observar o passo a passo do processo de pick and place. Inicialmente, o SCARA se move em direção ao pato, o agarra e o transporta até a caixa, retornando em seguida à posição de repouso onde o pato estava anteriormente.

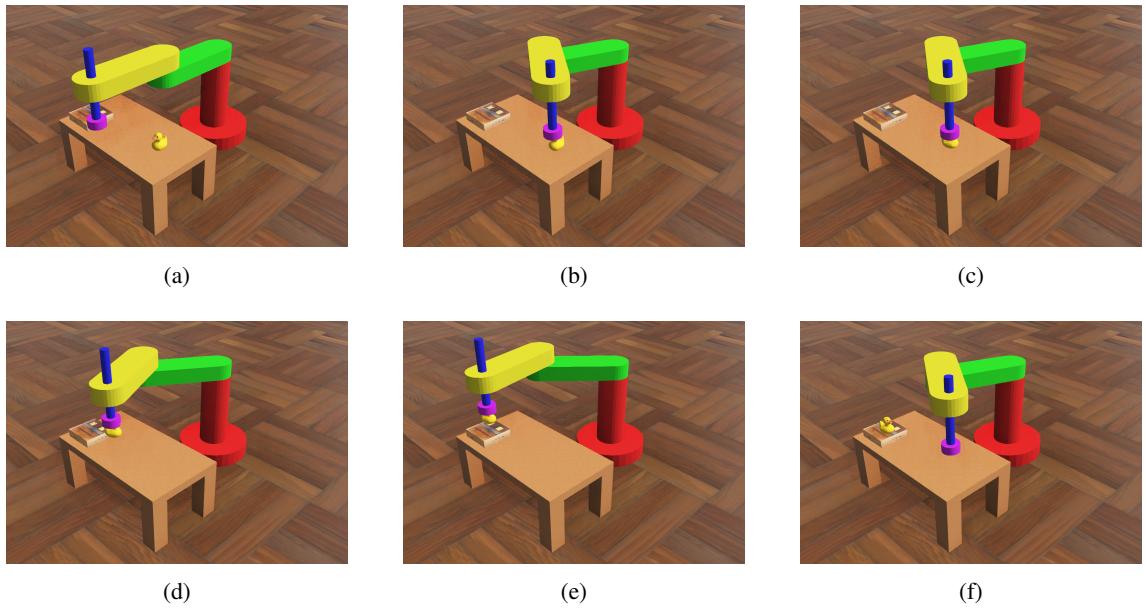


Figura 4: Passo a passo da simulação

## 5 Simulação

- O vídeo da simulação do Projeto - SCARA esta disponivel no [Google Drive](#).
- O repositório com os arquivos da simulação do Projeto - SCARA esta disponivel no [GitHub](#).