

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего
образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ

Факультет систем управления и робототехники

Отчет по лабораторной работе №6
«Разработка системы управления для
неполноприводного робота»
по дисциплине «Введение в профессиональную
деятельность»

Выполнили студенты гр. **R3135**

Трубицына А.М.

Вьюгина А.П.

Репин А.В.

Макаренко К.С.

Преподаватель: Перегудин А.А.,
ассистент фак. СУиР

1 Цель работы

Ознакомиться со способом нахождения параметров манипулятора и научиться переходить из декартовых координат в обобщенные и обратно.

2 Описание собранного робота

3 Коды для перехода из одной системы координат в другую

3.1 Функция для перехода из обобщенной СК в Декартову с помощью геометрических расчетов

```
1 def straight_transfer_formulas(theta1, theta2, theta3):
2     theta3 = theta2 - theta3
3     theta1 = np.deg2rad(theta1)
4     theta2 = np.deg2rad(theta2)
5     theta3 = np.deg2rad(theta3)
6     a = [0, 0.007, 0.128, 0.128] # массив значений a_i
7     alpha = [0, pi / 2, 0, 0]
8     d = [0, 0.18, 0, 0] # массив значений d_i
9     x = (a[2] * cos(theta2) + a[3] * cos(theta3)) * cos(theta1)
10    y = (a[2] * cos(theta2) + a[3] * cos(theta3)) * sin(theta1)
11    z = d[1] + a[2] * sin(theta2) + a[3] * sin(theta3)
12    print("x = ", x)
13    print("y = ", y)
14    print("z = ", z)
```

3.2 Функция для перехода из обобщенной СК в Декартову с помощью матричных расчетов

```
1 def straight_transfer(theta1, theta2, theta3):
2     theta3 = -theta3
3     # перевод из градусов в радианы
4     theta1 = np.deg2rad(theta1)
5     theta2 = np.deg2rad(theta2)
6     theta3 = np.deg2rad(theta3)
7     Q = np.array([[0], [0], [0], [1]])
8
9     a = [0, 0, 0.128, 0.128] # массив значений a_i
10    alpha = [0, pi / 2, 0, 0]
11    # alpha = [0, pi / 2, 0, pi / 2]
12    d = [0, 0.18, 0, 0] # массив значений d_i
13
14    T_0_1 = np.array([[cos(theta1), -sin(theta1) * cos(alpha[1]), sin(theta1) *
15    ↪ sin(alpha[1]), a[1] * cos(theta1)],
16    [sin(theta1), cos(theta1) * cos(alpha[1]), -cos(theta1) *
17    ↪ sin(alpha[1]), a[1] * sin(theta1)],
18    [0, sin(alpha[1]), cos(alpha[1]), d[1]],
19    [0, 0, 0, 1]])
20
21    T_1_2 = np.array([[cos(theta2), -sin(theta2) * cos(alpha[2]), sin(theta2) *
22    ↪ sin(alpha[2]), a[2] * cos(theta2)],
23    [sin(theta2), cos(theta2) * cos(alpha[2]), -cos(theta2) *
24    ↪ sin(alpha[2]), a[2] * sin(theta2)],
25    [0, sin(alpha[2]), cos(alpha[2]), d[2]],
26    [0, 0, 0, 1]])
27
28    T_2_3 = np.array([[cos(theta3), -sin(theta3) * cos(alpha[3]), sin(theta3) *
29    ↪ sin(alpha[3]), a[3] * cos(theta3)],
```

```

25         [sin(theta3), cos(theta3) * cos(alpha[3]), -cos(theta3) *
26           ↪ sin(alpha[3]), a[3] * sin(theta3)],
27         [0, sin(alpha[3]), cos(alpha[3]), d[3]],
28         [0, 0, 0, 1]])
29 T = T_0_1.dot(T_1_2).dot(T_2_3)
30 XYZ = T.dot(Q)
31 print("x = ", XYZ[0][0])
32 print("y = ", XYZ[1][0])
33 print("z = ", XYZ[2][0])

```

3.3 Функция для перехода из Декартовой СК в обобщенную

```

1 def backwards_transfer(x, y, z):
2     a = [0, 0, 0.128, 0.128] # массив значений a_i
3     d = [0, 0.18, 0, 0] # массив значений d_i
4     r1 = sqrt(x ** 2 + y ** 2)
5     r2 = z - d[1]
6     r3 = sqrt(r1 ** 2 + r2 ** 2)
7
8     theta1 = atan(y / x)
9     psi1 = acos((a[2] ** 2 + r3 ** 2 - a[3] ** 2) / (2 * a[2] * r3))
10    psi2 = atan(r2 / r1)
11    theta2 = psi2 + psi1
12    psi3 = acos((a[2] ** 2 + a[3] ** 2 - r3 ** 2) / (2 * a[2] * a[3]))
13    theta3 = pi - psi3
14
15    theta1 = np.rad2deg(theta1)
16    theta2 = np.rad2deg(theta2)
17    theta3 = np.rad2deg(theta3)
18
19    print("theta1 = ", theta1)
20    print("theta2 = ", theta2)
21    print("theta3 = ", theta3)

```

4 Код для движения робота

4.1 Поочередный поворот звеньев манипулятора на заданные углы

```

1 from ev3dev.ev3 import *
2 import time
3
4
5 def saturate(x, left, right):
6     if x > right: x = right
7     if x < left: x = left
8     return x
9
10
11 sound = Sound()
12 sound.set_volume(100)
13 sound.beep()
14
15 # первая координата - motorA
16 # вторая координата - motorB
17 # третья координата - motorC
18 q0 = [90, 45, 45]
19
20 # калибровка координат

```

```

21 q0 = [saturate(q0[0], -180, 180), saturate(q0[1], -70, 40), saturate(q0[2], -120,
    ↪ 100)]
22 q = [5 * q0[0], -5 * q0[1], -5/3 * q0[2]]
23
24 # значение коэффициентов в градусных мерах
25 k_p = [0.3, 0.3, 0.1]
26 k_i = [0.25/60, 0.25/60, 0]
27 k_d = [1/60, 1/60, 0]
28
29
30 motorA = LargeMotor('outA')
31 motorB = LargeMotor('outB')
32 motorC = MediumMotor('outC')
33
34 motorA.position = 0
35 motorB.position = 0
36 motorC.position = 0
37
38 timeStart = time.time()
39 last_t = time.time()
40 sum = 0
41 last_e = 0
42 inaccuracy = 5 # погрешность в градусах
43 U_max = 6.97
44
45 name = str(q0[0]) + "_" + str(q0[1]) + "_" + str(q0[2]) + ".txt"
46 file = open(name, 'w')
47
48 motors_set = [motorA, motorB, motorC]
49
50 for i in range(3):
51     while abs(q[i] - motors_set[i].position) > inaccuracy:
52         e = q[i] - motors_set[i].position
53         dt = time.time() - last_t
54         U = k_p[i] * e + k_d[i] * (e - last_e) / dt + k_i[i] * sum * dt
55         U = U/U_max*100
56         motors_set[i].run_direct(duty_cycle_sp=saturate(U, -100, 100))
57         file.write(str(motorA.position) + '\t' + str(motorB.position) + '\t' +
    ↪ str(motorC.position) + '\t' + str(
58             saturate(U, -100, 100)) + '\t' + str(k_p[i] * e) + '\t' + str(k_d[i] * (e
    ↪ - last_e) / dt) + '\t' +
59             str(k_i[i] * sum * dt) + '\n')
60         sum += e
61         last_e = e
62         last_t = time.time()
63     sum = 0
64     last_e = 0
65     last_t = time.time()
66     motors_set[i].run_direct(duty_cycle_sp=0)
67
68 file.close()

```

5 Выводы