

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 №7
«Прямая и обратная задача кинематики.
ДН-параметры»
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

Фотографии собранного робота

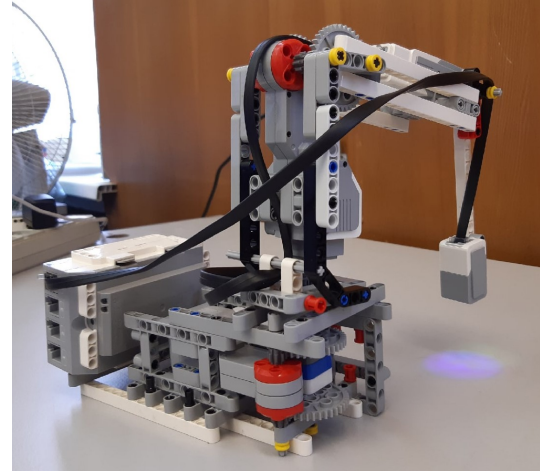
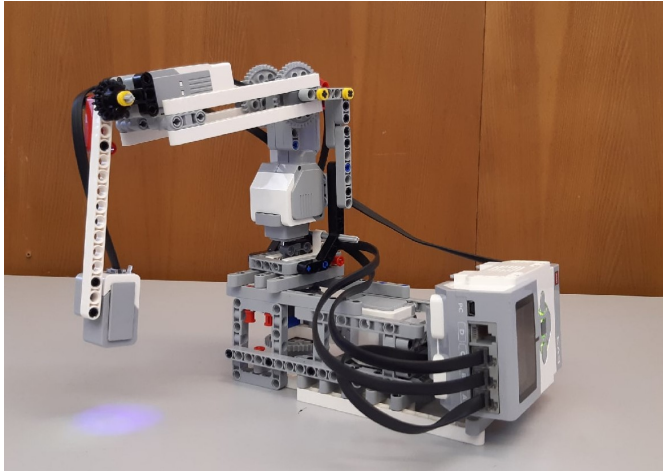


Рис. 1: Фотографии робота

У первого и второго мотора передаточное отношение равно 5, у третьего - $\frac{5}{3}$ (при нумерации моторов снизу вверх).

Параметры Денавита-Харденберга

Для упрощения расчетов и использования четырех величин вместо 6 можно использовать параметры Денавита-Харденберга. Это один из наиболее общих способов определить координаты хвата манипулятора по обобщенным координатам манипулятора (т. е. по углам, на которые поворачиваются звенья). Безусловно, координаты можно рассчитать и геометрически, как сделано далее, но в случае, когда звеньев много, этот способ оказывается неудобным.

Звено i	a_i	α_i	d_i	θ_i
1	0,7 см	$\frac{\pi}{2}$	18 см	θ_1
2	12,8 см	0	0	$\theta_2 + \frac{\pi}{2}$
3	12,8 см	0	0	θ_3

3 Коды для перехода из одной системы координат в другую

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

Вывод формул

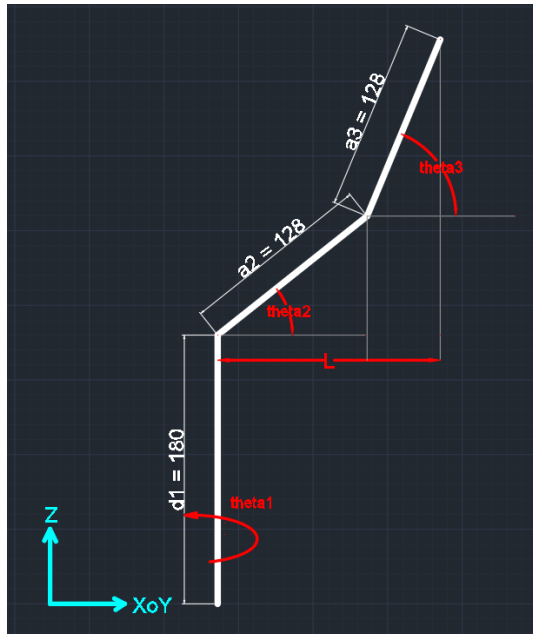


Рис. 2: Чертеж для вывода указанных формул

Из чертежа:

$$\begin{cases} z = d_1 + a_2 \sin \theta_2 + a_3 \sin \theta_3 \\ L = a_2 \cos \theta_2 + a_3 \cos \theta_3 \end{cases}$$

Тогда итоговый результат равен

$$\begin{cases} x = \cos \theta_1 (a_2 \cos \theta_2 + a_3 \cos \theta_3) \\ y = \sin \theta_1 (a_2 \cos \theta_2 + a_3 \cos \theta_3) \\ z = d_1 + a_2 \sin \theta_2 + a_3 \sin \theta_3 \end{cases}$$

Код

```
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

Вывод формул

Матрица перехода из обобщенной СК в Декартову является произведением матриц перехода от $i - 1$ -го звена к i -ому: $T_n^0 = T_1^0 \cdot T_2^1 \cdot T_3^2$.

Итоговая матрица - результат следующего выражения:

$$T_3^0 = \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 \cos \alpha_1 & \sin \theta_1 \sin \alpha_1 & a_1 \cos \theta_1 \\ \sin \theta_1 & \cos \theta_1 \cos \alpha_1 & -\cos \theta_1 \sin \alpha_1 & a_1 \sin \theta_1 \\ 0 & \sin \alpha_1 & \cos \alpha_1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos \theta_2 & -\sin \theta_2 \cos \alpha_2 & \sin \theta_2 \sin \alpha_2 & a_2 \cos \theta_2 \\ \sin \theta_2 & \cos \theta_2 \cos \alpha_2 & -\cos \theta_2 \sin \alpha_2 & a_2 \sin \theta_2 \\ 0 & \sin \alpha_2 & \cos \alpha_2 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos \theta_3 & -\sin \theta_3 \cos \alpha_3 & \sin \theta_3 \sin \alpha_3 & a_3 \cos \theta_3 \\ \sin \theta_3 & \cos \theta_3 \cos \alpha_3 & -\cos \theta_3 \sin \alpha_3 & a_3 \sin \theta_3 \\ 0 & \sin \alpha_3 & \cos \alpha_3 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Код

```
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    d = [0, 0.18, 0, 0] # массив значений d_i
12
13    T_0_1 = np.array([[cos(theta1), -sin(theta1) * cos(alpha[1]), sin(theta1) *
14    ↪ sin(alpha[1]), a[1] * cos(theta1)],
15                      [sin(theta1), cos(theta1) * cos(alpha[1]), -cos(theta1) *
16    ↪ sin(alpha[1]), a[1] * sin(theta1)],
17                      [0, sin(alpha[1]), cos(alpha[1]), d[1]],
18                      [0, 0, 0, 1]])
19
20    T_1_2 = np.array([[cos(theta2), -sin(theta2) * cos(alpha[2]), sin(theta2) *
21    ↪ sin(alpha[2]), a[2] * cos(theta2)],
22                      [sin(theta2), cos(theta2) * cos(alpha[2]), -cos(theta2) *
23    ↪ sin(alpha[2]), a[2] * sin(theta2)],
24                      [0, sin(alpha[2]), cos(alpha[2]), d[2]],
25                      [0, 0, 0, 1]])
26
27    T_2_3 = np.array([[cos(theta3), -sin(theta3) * cos(alpha[3]), sin(theta3) *
28    ↪ sin(alpha[3]), a[3] * cos(theta3)],
29                      [sin(theta3), cos(theta3) * cos(alpha[3]), -cos(theta3) *
30    ↪ sin(alpha[3]), a[3] * sin(theta3)],
31                      [0, sin(alpha[3]), cos(alpha[3]), d[3]],
32                      [0, 0, 0, 1]])
33
34    T = T_0_1.dot(T_1_2).dot(T_2_3)
35    XYZ = T.dot(Q)
36    print("x = ", XYZ[0][0])
37    print("y = ", XYZ[1][0])
38    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)
```

```

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' +
58         ↪ str(motorC.position) + '\t' + str(
59         ↪ saturate(U, -100, 100)) + '\t' + str(k_p[i] * e) + '\t' + str(k_d[i] * (e
60         ↪ - last_e) / dt) + '\t' +
61         ↪ str(k_i[i] * sum * dt) + '\n')
62     sum += e
63     last_e = e
64     last_t = time.time()
65     sum = 0
66     last_e = 0
67     last_t = time.time()
68     motors_set[i].run_direct(duty_cycle_sp=0)
69
70 file.close()

```

Код, реализующий движение по нескольким заданным точкам, аналогичен вышеуказанному, с единственной разницей в том, что есть массив массивов с точками, которые необходимо обойти, и внешний цикл `for j in range(n)`.

4.2 Одновременный поворот звеньев манипулятора на заданные углы

Единственное отличие от кода, представленного выше, заключается в следующем участке, где определяется управляющее воздействие.

```

1 while abs(q[0] - motors_set[0].position) > inaccuracy or abs(q[1] -
  ↪ motors_set[1].position) > inaccuracy or abs(q[2] - motors_set[2].position) >
  ↪ inaccuracy:
2     for i in range(3):
3         e[i] = q[i] - motors_set[i].position
4         dt = time.time() - last_t
5         U[i] = k_p[i] * e[i] + k_d[i] * (e[i] - last_e[i]) / dt + k_i[i] * sum[i] * dt
6         U[i] = U[i]/U_max*50
7         sum[i] += e[i]
8         last_e[i] = e[i]
9         last_t = time.time()
10        if abs(q[i] - motors_set[i].position) > inaccuracy:
11            ↪ motors_set[i].run_direct(duty_cycle_sp=saturate(U[i], -100, 100))
12    file.write(str(motorA.position) + '\t' + str(motorB.position) + '\t' +
13        ↪ str(motorC.position) + '\n')

```

5 Построенные в Matlab траектории движения робота

5.1 Скрипт на Matlab

```

1 results=dlmread('C:\Users\Anastasia\PycharmProjects\Robotics_Lab7\OUT
  ↪ 25.05\square2\square2_4___-90_90_0.txt'); %считываем файл
2 theta1=results(:,1);
3 theta2=results(:,2);
4 theta3=results(:,3);
5
6
7 theta1 = theta1*pi/180/5;
8 theta2 = theta2*pi/180/(-5);

```

```

9   theta3 = theta3*pi/180/(-5/3);
10  theta3 = theta2-theta3;
11
12  a = [ 0.007, 0.128, 0.128];
13  d = [ 0.18, 0, 0];
14  for i = 1:size(results)
15      x(i)=(a(2)*cos(theta2(i)) + a(3)*cos(theta3(i)))*(cos(theta1(i)));
16      y(i) = (a(2) * cos(theta2(i)) + a(3) * cos(theta3(i)))*(sin(theta1(i)));
17      z(i) = d(1) + a(2) * sin(theta2(i)) + a(3) * sin(theta3(i));
18  end
19
20  hold on;
21  line = plot3(x,y,z, 'magenta');
22  line.LineWidth = 2.5;
23  grid on;

```

5.2 Построенные графики

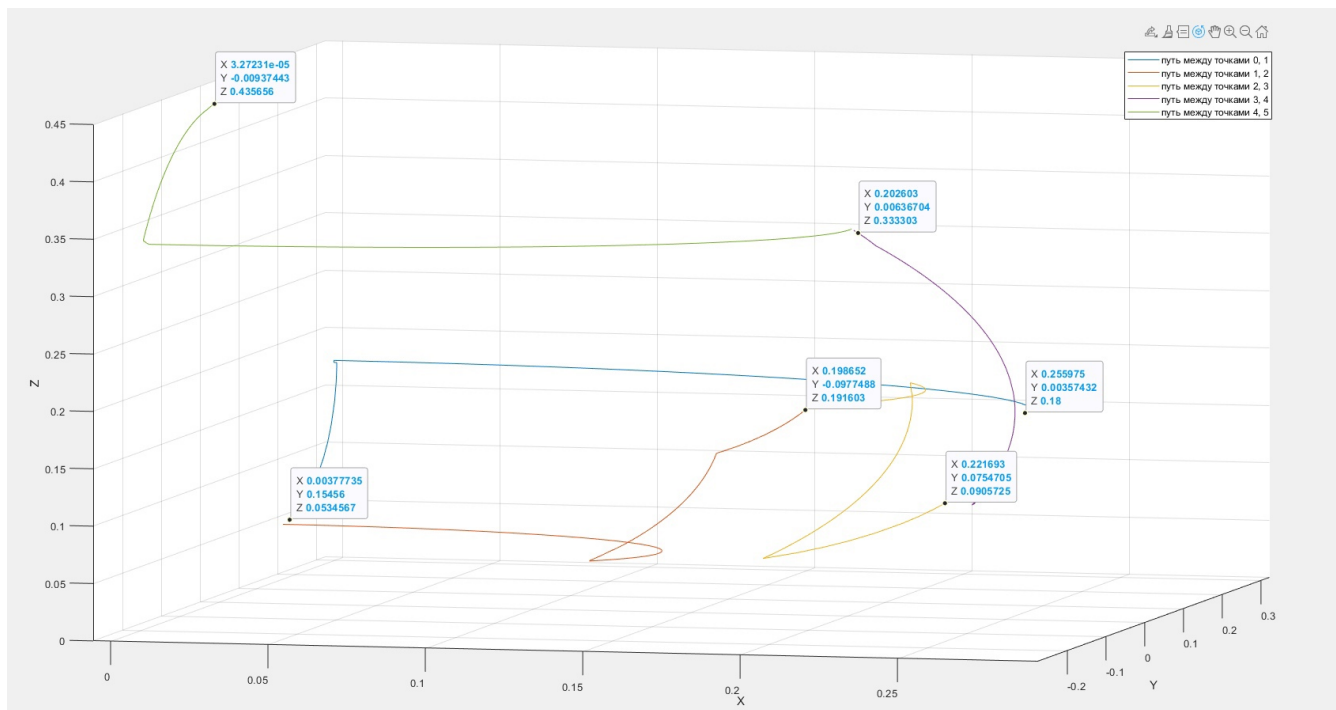


Рис. 3: Обход манипулятором 5 точек с поочередным вращением моторов

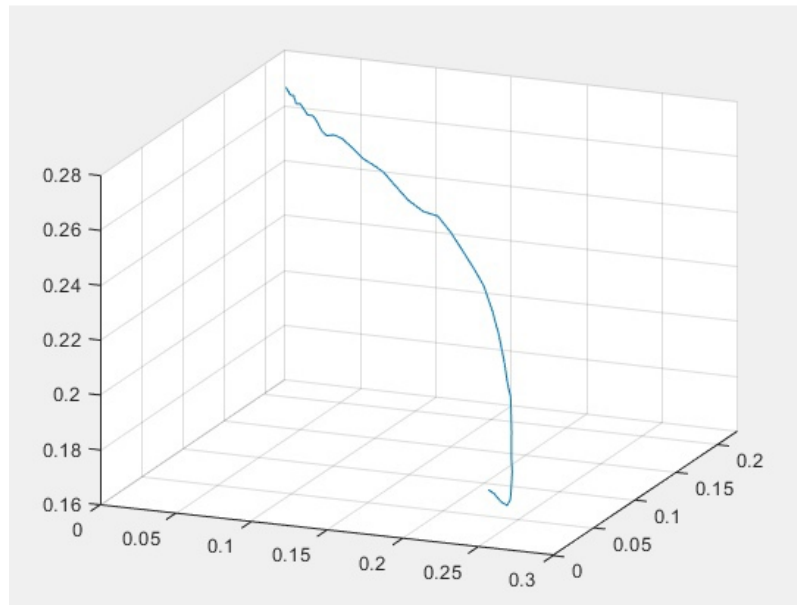


Рис. 4: Перемещение в точку с одновременным вращением моторов

6 Использование датчика цвета ColorSensor

В процессе работы появилась идея с помощью манипулятора реализовать механизм, который будет определять цвет предмета и записывать его координату.



Так как датчик определяет цвета достаточно неточно, то мы придумали некоторый алгоритм калибровки для попадания в классический 8-битный RGB интервал. Для этого мы удерживали датчик, смотрящий на белую бумагу, в одном положении, искали максимальное значение каждой компоненты света. В файл записывалось текущее значение компоненты, деленной на соответствующую максимальную и умноженное на 255. Также возникла идея построить график каждой из компонент и выделить пиковые значения.

6.1 Скрипты для реализации этой задачи

Код для калибровки датчика

```

1 red_component = blue_component = green_component = 0
2 counter = 0
3 while time.time() - time_start < 0.5:
4     counter += 1
5     red_component += color_sensor.value(0)
6     green_component += color_sensor.value(1)
7     blue_component += color_sensor.value(2)
8 red_ideal = red_component/counter
9 green_ideal = green_component/counter
10 blue_ideal = blue_component/counter

```

Код для построения графиков

```

1 results=dlmread('C:\Users\Anastasia\PycharmProjects\Robotics_Lab7\OUT
  ↳ 8.06\code_color2.txt'); %считываем файл
2 time = results(:, 1);
3 red = results(:, 5);
4 green = results(:, 6);

```



```

5 blue = results(:, 7);
6 hold on;
7 plot(time, red, 'r');
8 plot(time, green, 'g');
9 plot(time, blue, 'b');

```

6.2 Полученный график интенсивностей RGB-компонент отраженного света

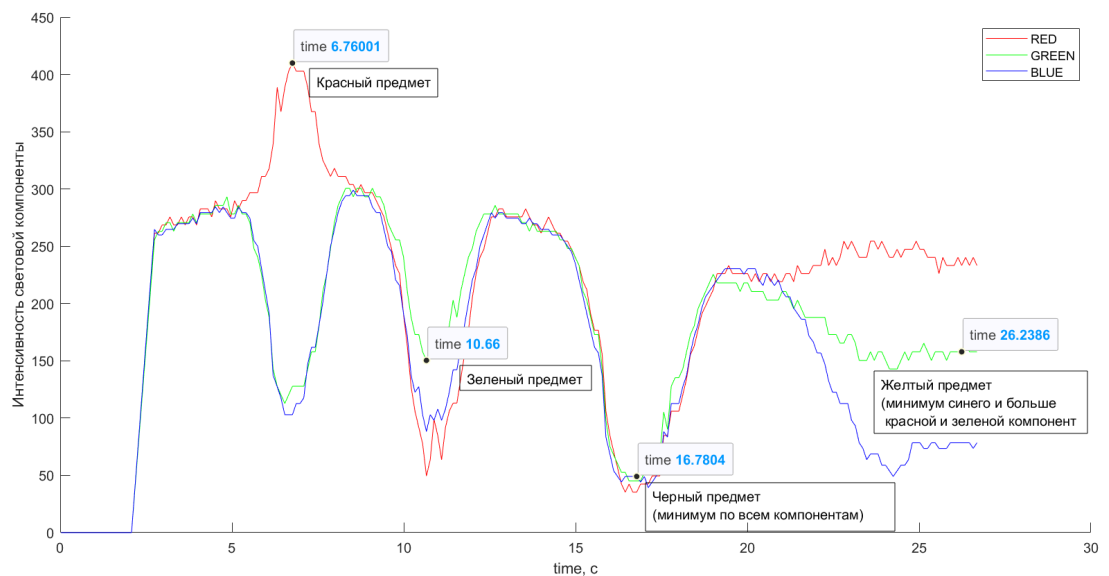


Рис. 5: График интенсивностей компонент

При анализе графика можно заметить 4 момента времени, в которые так или иначе устанавливается некоторое пиковое (т. е. максимальное или минимальное) значение по какой-то из компонент.

- Максимальное значение у красного, минимальные у синего и зеленого определяют красный цвет
- Значение зеленого больше, чем у синего и красного - значит, цвет - зеленый
- Минимум по всем компонентам определяет черный цвет
- Смесь красного и зеленого с минимальным количеством синего определяет желтый цвет. На этом моменте манипулятор замер, поэтому это значение является не пиковым, а установившимся. Однако, если бы робот продолжил движение, то значения бы также продолжили меняться.

Тогда, если проводить расчет для каждой точки, то получаются следующие результаты:

- Координата красной крышки: (0.07, -0.08, 0)
- Координата зеленого объекта: (0.1, -0.025, 0)
- Координата черной крышки: (0.09, 0.06, 0)
- Координата желтой детали: (0.02, 0.11, 0)

Координата z у всех трех объектов нулевая, потому что они лежат в плоскости XoY .

Действительно, аналитические выкладки верны, так как на самом деле ситуация выглядела следующим образом:

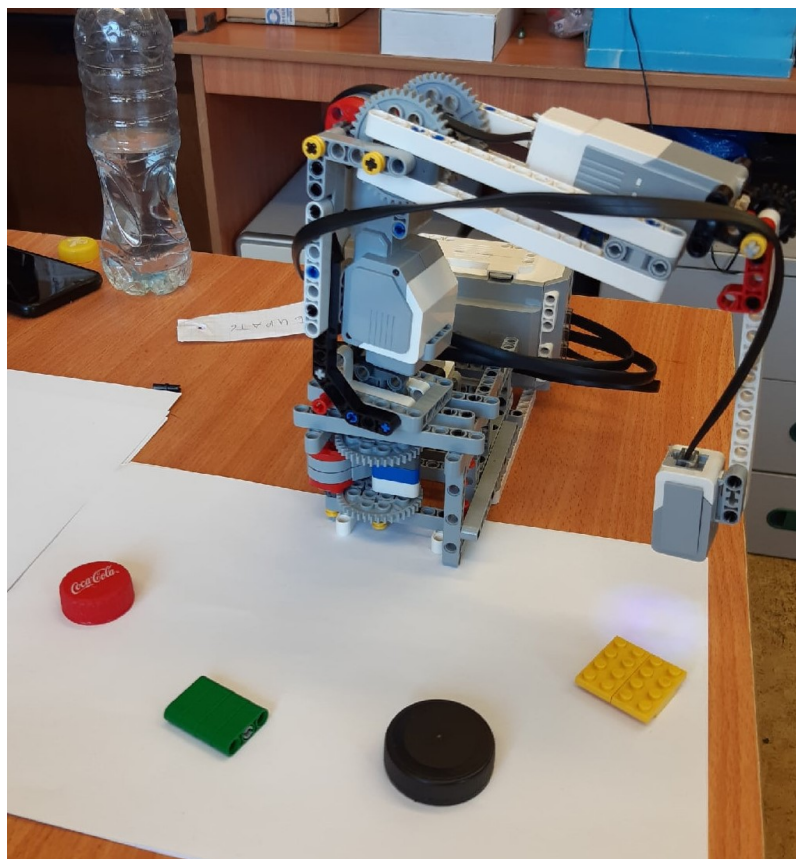


Рис. 6: Робот с крышками

7 Выводы

В процессе выполнения лабораторной работы ознакомились со способами нахождения параметров манипулятора и научились переходить из декартовых координат в обобщенные и обратно.

Более того, была поставлена задача - научить робота распознавать цвета и узнать координату предмета того или иного цвета. Задача выполняется в полуавтоматическом режиме - робот записывает данные в файл, затем происходит ручной анализ графика и выборка необходимых углов, по которым в дальнейшем происходит расчет координат.