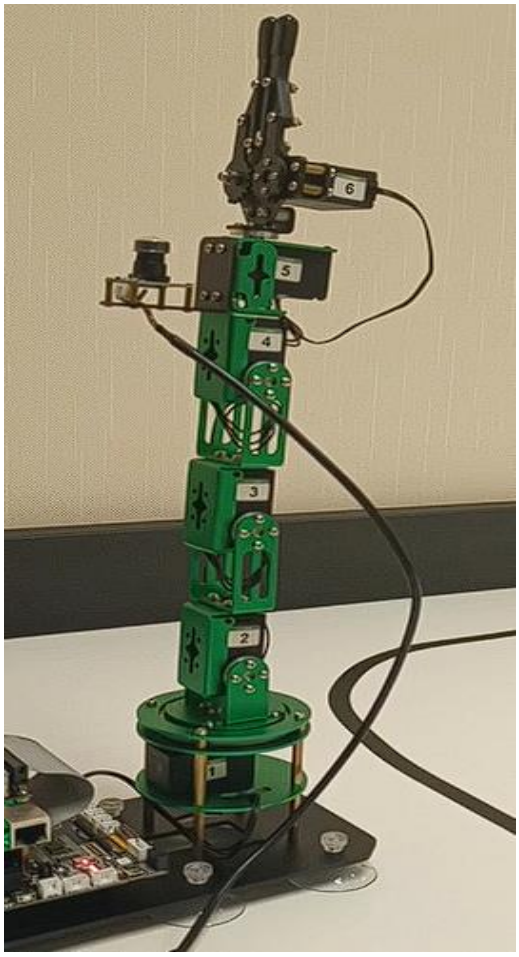
Прямая задача кинематики для манипулятора Dofbot

Dofbot – имеет 6 степеней свободы. Его рука может захватывать предметы от 1 до 6 см, массой 500 грамм. Первое звено вращательное, остальные поступательные. На роботе установлены 2 платы, Jetson Nano, отвечающая за управления роботом, и программируемая плата на Python Rasberi Pi 4B. Управление роботом происходит по wi-fi сети по shh протоколу. На роботе установлена камера.



Прямая задача кинематики заключается в нахождение координат кисти манипулятора по известным углам звеньев робота.

Функция для вычисления sin и cos

```
def calculate_sin_cos_degrees(angle_in_degrees):
    angle_in_degrees = transform_angle(angle_in_degrees)
    angle_in_radians = np.radians(angle_in_degrees)
    sin_value = np.sin(angle_in_radians)
    cos_value = np.cos(angle_in_radians)
    return sin_value, cos_value
```

Углы робота

```
angle1_q = 90#1
angle1_a = 90#2
angle2_a = -40#3
angle3_a = 90#4
#Всегда 0
angle2_q = 90 #3
angle3_q = 90
```

Длинны звеньев робота

```
l1 = 125
l2 = 85
l3 = 85
l4 = 200
```

Вычисления sin и cos углов

```
sq1, cq1 = calculate_sin_cos_degrees(angle1_q)
sa1, ca1 = calculate_sin_cos_degrees(angle1_a)

sq2, cq2 = calculate_sin_cos_degrees(angle2_q)
sa2, ca2 = calculate_sin_cos_degrees(angle2_a)

sq3, cq3 = calculate_sin_cos_degrees(angle3_q)
sa3, ca3 = calculate_sin_cos_degrees(angle3_a)
```

Матрица перехода к системе координат 2 звена

```
A = np.array([[cq1, -cq1*sq1, sa1*sq1, 0],
              [sq1, ca1*cq1, -sa1*cq1, 0],
              [0, sa1, ca1, l1],
              [0, 0, 0, 1]])
```

Матрица перехода к системе координат 3 звена

```
B = np.array([[cq2, -cq2*sq2, sa2*sq2, 0],
              [sq2, ca2*cq2, -sa2*cq2, 0],
              [0, sa2, ca2, l2],
              [0, 0, 0, 1]])
```

Матрица перехода к системе координат 4 звена

```
C = np.array([[cq3, -cq3*sq3, sa3*sq3, 0],
              [sq3, ca3*cq3, -sa3*cq3, 0],
              [0, sa3, ca3, l3],
              [0, 0, 0, 1]])
```

5 звено отвечает за сжатие кисти манипулятора.

Матрица перехода к системе координат к 6 звену

```
D = np.array([[1, 0, 0, 0],
              [0, 1, 0, 0],
              [0, 0, 1, l4],
              [0, 0, 0, 1]])
```

5 звено отвечает за сжатие кисти манипулятора.

Перемножив матрицы перехода, мы получим итоговую матрицу.

```
result = multiply_matrices(A, B)
result = multiply_matrices(result, C)
result = multiply_matrices(result, D)
```

Нужен последний столбец – это координаты манипулятора.

```
# Умножение на столбец
column_vector = np.array([0, 0, 0, 1])
final_result = np.dot(result, column_vector)
# Вывод координат
x, y, z = final_result[:3]
print(angle1_q, angle1_a, angle2_a, angle3_a)
print(f"x: {x:.0f}, y: {y:.0f}, z: {z:.0f}")
```