

ROS code

Function - 1

```
1 import rclpy
2 from rclpy.node import Node
3 import math
4 from std_msgs.msg import Float32MultiArray
5 from geometry_msgs.msg import Pose
6
7 ##global variables; link lengths of a given robotic manipulator
8 l1=1
9 l2=1
10 l3=1
11
12 def angle_to_config(q_angles):
13
14     ## initialize angle variables with conversion of q angles to radians from degree
15     q1=math.radians(q_angles[0])
16     q2=math.radians(q_angles[1])
17     q3=math.radians(q_angles[2])
18
19
20
21     ## calculate robot's position vector using forward kinematics
22     x = (l3 * math.cos(q1)* math.cos(q2) * math.cos(q3)) - (l3 * math.cos(q1) * math.sin(q2) * math.sin(q3)) + (l2 * math.cos(q2)*math.cos(q1))
23     y = (l3 * math.sin(q1)* math.cos(q2)*math.cos(q3)) - (l3* math.sin(q1)* math.sin(q2) * math.sin(q3)) + (l2 * math.cos(q2)*math.sin(q1))
24     z = -(l3 * math.sin(q2)* math.cos(q3)) - (l3* math.cos(q2) * math.sin(q3)) - (l2* math.sin(q2)) + l1
25
26     ##x = (l3 * math.cos(q1)* math.cos(q2) * math.cos(q3)) - (l3 * math.cos(q1) * math.sin(q2) * math.sin(q3))
27     ##y = (l3 * math.sin(q1)* math.cos(q2)* math.cos(q3)) - (l3* math.sin(q1)* math.sin(q2) * math.sin(q3))
28     ##z = l1 -(l2* math.sin(q2)) - (l3 * math.sin(q2)* math.cos(q3)) - (l3* math.cos(q2) * math.sin(q3))
29
30     ## calculate rotation matrix components using forward kinematics
31     r11 = (math.cos(q1)* math.cos(q2) * math.cos(q3)) - (math.cos(q1)* math.sin(q2) * math.sin(q3))
32     r12 = -(math.cos(q1)* math.cos(q2) * math.sin(q3)) - (math.cos(q1)* math.sin(q2) * math.cos(q3))
33     r13 = -math.sin(q1)
34
35     r21 = (math.cos(q3) * math.sin(q1)) - (math.sin(q1) * math.sin(q2) * math.sin(q3))
36     r22 = -(math.sin(q1) * math.sin(q3)) - (math.sin(q1) * math.sin(q2) * math.cos(q3))
37     r23 = math.cos(q1)
38
39     r31 = -(math.sin(q2) * math.cos(q3)) - (math.cos(q2) * math.sin(q3))
40     r32 = (math.sin(q2) * math.sin(q3)) - (math.cos(q2) * math.cos(q3))
41     r33 = 0
42
43
44     return [r11,r12,r13, r21,r22,r23, r31,r32,r33, x, y,z]
45
46
```

Function2

```
45
46
47 def config_to_angle(config):
48     ##extract postion info from the geometry message recived
49     ##position: holds info of [x,y,z] position of an end effector
50
51     position = config
52
53
54     ##extract x,y,x values
55     x_value = position.x
56     y_value = position.y
57     z_value = position.z
58
59     ##Calculate joint angles theta1, theta2 and theta3 for the given robot manipulator
60     ##seeing from top view we can calculate theta_1
61     theta_1 = math.atan2(y_value,x_value)
62
63     ## calculate theta2 using law of cosine; r_value and s_value are temporary variables used for calculations
64     ## seeing from top view, we can calculate r_value
65     r_value = x_value / math.cos(theta_1)
66     s_value = z_value - l1
67     ##using law of cosine of traingle
68     cos_alpha = -((r_value ** 2 + s_value ** 2)- l2**2 - l3**2)/(2*l2*l3))
69     ##print (cos_alpha)
70     theta_3 = math.acos(cos_alpha)
71     theta_3 = math.pi - theta_3
72
73
74     ## calculate theta2, seeing from side view, using subtraction of angles
75     theta_2 = math.atan2(s_value, r_value) - math.atan2(l3 * math.sin(theta_3),( l2 + l3 * math.cos(theta_3)) )
76
77
78
79     ##now convert radians to degree
80     theta_1 = math.degrees(theta_1)
81     theta_2 = math.degrees(theta_2)
82     theta_3 = math.degrees(theta_3)
83
84
85     return [theta_1, theta_2, theta_3]
86
87
88
89
```

```

90
91
92
93 class my_node(Node):
94
95     def __init__(self):
96         super().__init__('my_node')
97         ##this subscriber recives joint angle values and calculates robot end effector configurations using forward kinematics
98         self.subscription_1 = self.create_subscription(
99             Float32MultiArray,
100             'angle_to_config',
101             self.listener_callback_1,
102             10)
103         ## creating a subscriber to recieve configuration values-pose of the end effector
104         ## it calculates joint angles from end effector configurations using Inverse Kinematics
105         self.subscription_2 = self.create_subscription(
106             Pose,
107             'config_to_angle',
108             self.listener_callback_2,
109             10)
110
111         self.subscription_1 # prevent unused variable warning
112         self.subscription_2
113
114     def listener_callback_1(self, msg):
115
116         configurations = angle_to_config(msg.data)
117
118         self.get_logger().info('Robot position vector is: "%s"' % configurations[9:12])
119         self.get_logger().info('Robot rotation matrix is: "%s"' % configurations[0:9])
120
121     def listener_callback_2(self, msg):
122         joint_angles = config_to_angle(msg.position)
123         self.get_logger().info('Robot joint angles are [theta_1, theta_2, theta_3]: "%s"' % joint_angles)
124
125
126
127
128
129
130
131 def main(args=None):
132
133     rclpy.init(args=args)
134     my_node_subscriber = my_node()
135     rclpy.spin(my_node_subscriber)
136
137     # Destroy the node explicitly
138     # (optional - otherwise it will be done automatically
139     # when the garbage collector destroys the node object)
140     my_node_subscriber.destroy_node()
141     rclpy.shutdown()
142
143
144 if __name__ == '__main__':
145     main()

```

Results

Case 1

Input to forward kinematics

```
swati@swati:~/ros2_humble$ ros2 topic pub /angle_to_config std_msgs/Float32MultiArray "{data:[90,60, 60]}"
publisher: beginning loop
publishing #1: std_msgs.msg.Float32MultiArray(layout=std_msgs.msg.MultiArrayLayout(dim=[], data_offset=0), data=[90.0, 60.0, 60.0])

publishing #2: std_msgs.msg.Float32MultiArray(layout=std_msgs.msg.MultiArrayLayout(dim=[], data_offset=0), data=[90.0, 60.0, 60.0])

^Cswati@swati:~/ros2_humble$
```

Output configs - input to inverse kinematics

```
swati@swati:~/ros2_humble$ ros2 run my_package1 my_node
[INFO] [1696612724.479931895] [my_node]: Robot position vector is: "[2.465190328815662e-32, 3.3306690738754696e-16, -0.7320508075688774]"
[INFO] [1696612724.480235100] [my_node]: Robot rotation matrix is: "[-3.061616997868381e-17, -5.302876193624535e-17, -1.0, -0.249999999999999978, -1.299038105676658, 6.123233995736766e-17, -0.8660254037844388, 0.49999999999999998, 0]"
[INFO] [1696612725.472164307] [my_node]: Robot position vector is: "[2.465190328815662e-32, 3.3306690738754696e-16, -0.7320508075688774]"
[INFO] [1696612725.473268502] [my_node]: Robot rotation matrix is: "[-3.061616997868381e-17, -5.302876193624535e-17, -1.0, -0.249999999999999978, -1.299038105676658, 6.123233995736766e-17, -0.8660254037844388, 0.49999999999999998, 0]"
```

```
swati@swati:~/ros2_humble$ ros2 topic pub /config_to_angle geometry_msgs/Pose "{position: {x: 2.465190328815662e-32, y: 3.3306690738754696e-16, z: -0.732050807568877}, orientation: {x: 0.0, y: 0.0, z: 0.0, w: 1.0}}"
publisher: beginning loop
publishing #1: geometry_msgs.msg.Pose(position=geometry_msgs.msg.Point(x=2.465190328815662e-32, y=3.3306690738754696e-16, z=-0.732050807568877), orientation=geometry_msgs.msg.Quaternion(x=0.0, y=0.0, z=0.0, w=1.0))

^Cswati@swati:~/ros2_humble$
```

```
swati@swati:~/ros2_humble$ ros2 run my_package1 my_node
[INFO] [1696612817.558257565] [my_node]: Robot joint angles are [theta_1, theta_2, theta_3]: "[90.0, -120.00000000000001, 60.00000000000036]"
```

Case 2:

Input to forward kinematics

```
swati@swati:~/ros2_humble$ ros2 topic pub /angle_to_config std_msgs/Float32MultiArray "{data:[0,60,60]}"
publisher: beginning loop
publishing #1: std_msgs.msg.Float32MultiArray(layout=std_msgs.msg.MultiArrayLayout(dim=[], data_offset=0), data=[0.0, 60.0, 60.0])

publishing #2: std_msgs.msg.Float32MultiArray(layout=std_msgs.msg.MultiArrayLayout(dim=[], data_offset=0), data=[0.0, 60.0, 60.0])

publishing #3: std_msgs.msg.Float32MultiArray(layout=std_msgs.msg.MultiArrayLayout(dim=[], data_offset=0), data=[0.0, 60.0, 60.0])

^Cswati@swati:~/ros2_humble$
```

Config output

```
swati@swati: ~/ros2_humble
swati@swati: ~/ros2_humble
swati@swati: ~/ros2_humble

swati@swati:~/ros2_humble$ ros2 run my_package1 my_node
[INFO] [1696612981.386015652] [my_node]: Robot position vector is: "[3.3306698738754696e-16, 0.0, -0.7320508075688774]"
[INFO] [1696612981.386398632] [my_node]: Robot rotation matrix is: "[-0.4999999999999998, -0.8660254037844388, -0.0, 0.0, -0.0, 1.0, -0.8660254037844388, 0.4999999999999998, 0]"
[INFO] [1696612982.374927380] [my_node]: Robot position vector is: "[3.3306698738754696e-16, 0.0, -0.7320508075688774]"
[INFO] [1696612982.376107365] [my_node]: Robot rotation matrix is: "[-0.4999999999999998, -0.8660254037844388, -0.0, 0.0, -0.0, 1.0, -0.8660254037844388, 0.4999999999999998, 0]"
[INFO] [1696612983.374198422] [my_node]: Robot position vector is: "[3.3306698738754696e-16, 0.0, -0.7320508075688774]"
[INFO] [1696612983.375302750] [my_node]: Robot rotation matrix is: "[-0.4999999999999998, -0.8660254037844388, -0.0, 0.0, -0.0, 1.0, -0.8660254037844388, 0.4999999999999998, 0]"
```

```
swati@swati: ~/ros2_humble
swati@swati: ~/ros2_humble
swati@swati: ~/ros2_humble

swati@swati:~/ros2_humble$ ros2 topic pub /config_to_angle geometry_msgs/Pose "[position: {x: 3.3306698738754696e-16, y: 0.0, z: -0.7320508075688774}, orientation: {x: 0.0, y: 0.0, z: 0.0, w: 1.0})"
publisher: beginning loop
publishing #1: geometry_msgs.msg.Pose(position=geometry_msgs.msg.Point(x=3.3306698738754696e-16, y=0.0, z=-0.7320508075688774), orientation=geometry_msgs.msg.Quaternion(x=0.0, y=0.0, z=0.0, w=1.0))
publishing #2: geometry_msgs.msg.Pose(position=geometry_msgs.msg.Point(x=3.3306698738754696e-16, y=0.0, z=-0.7320508075688774), orientation=geometry_msgs.msg.Quaternion(x=0.0, y=0.0, z=0.0, w=1.0))
publishing #3: geometry_msgs.msg.Pose(position=geometry_msgs.msg.Point(x=3.3306698738754696e-16, y=0.0, z=-0.7320508075688774), orientation=geometry_msgs.msg.Quaternion(x=0.0, y=0.0, z=0.0, w=1.0))
^Cswati@swati:~/ros2_humble$
```

Output of inverse kinematic

```
swati@swati: ~/ros2_humble
swati@swati: ~/ros2_humble
swati@swati: ~/ros2_humble

swati@swati:~/ros2_humble$ ros2 run my_package1 my_node
[INFO] [1696613054.996573266] [my_node]: Robot joint angles are [theta_1, theta_2, theta_3]: "[0.0, -119.99999999999997, 59.999999999999986]"
[INFO] [1696613055.988159475] [my_node]: Robot joint angles are [theta_1, theta_2, theta_3]: "[0.0, -119.99999999999997, 59.999999999999986]"
[INFO] [1696613056.986294846] [my_node]: Robot joint angles are [theta_1, theta_2, theta_3]: "[0.0, -119.99999999999997, 59.999999999999986]"
```