

Lesson 4 The Movement of Robotic Arm on XYZ Axes

1.Working Principle

Based on the previous basic lessons of forward and inverse kinematics, this lesson will explain how to apply them.

Firstly, return to the initial position and ensure the coordinate system of the starting position of robotic arm. Then use reverse kinematics to control the robotic arm. Calculate the solution of the pitch angle of robotic arm by using the new coordinate system of robotic arm and use the solution of the pitch angle to set the servo angle.

The source code of program is located in:/home/ubuntu/armpi_pro/src/armpi_pro_demo/kinematics_demo/kinematics_demo.py

```



40 if __name__ == '__main__':
41     # Initialization node
42     rospy.init_node('kinematics_demo', log_level=rospy.DEBUG)
43     rospy.on_shutdown(stop)
44     # Servo publish
45     joints_pub = rospy.Publisher('/servo_controllers/port_id_1/multi_id_pos_dur', MultiRawIdPosDur, queue_size=1)
46     rospy.sleep(0.2) # Delay takes effect
47
48     # Set the initial position
49     target = ik.setPitchRanges((0.0, 0.12, 0.15), -90, -180, 0) # Get the solution with kinematics
50     if target: # Determine whether there is a solution
51         servo_data = target[1]
52         # Drive the robotic arm to move
53         bus_servo_control.set_servos(joints_pub, 1500, ((1, 200), (2, 500), (3, servo_data[servo3]),
54             (4, servo_data[servo4]), (5, servo_data[servo5]), (6, servo_data[servo6])))
55     time.sleep(1.5)
56     # move to 0.15m along x-axis
57     target = ik.setPitchRanges((0.15, 0.12, 0.15), -90, -180, 0)
58     if target:
59         servo_data = target[1]
60         bus_servo_control.set_servos(joints_pub, 1500, ((1, 200), (2, 500), (3, servo_data[servo3]),
61             (4, servo_data[servo4]), (5, servo_data[servo5]), (6, servo_data[servo6])))
62     time.sleep(1.5)
63     # Return to the initial position.
64     target = ik.setPitchRanges((0.0, 0.12, 0.15), -90, -180, 0)
65     if target:
66         servo_data = target[1]
67         bus_servo_control.set_servos(joints_pub, 1500, ((1, 200), (2, 500), (3, servo_data[servo3]),
68             (4, servo_data[servo4]), (5, servo_data[servo5]), (6, servo_data[servo6])))
69     time.sleep(2)
70     # Move to 0.2m along y-axis
71     target = ik.setPitchRanges((0.0, 0.20, 0.15), -90, -180, 0)
72     if target:
73         servo_data = target[1]
74         bus_servo_control.set_servos(joints_pub, 1500, ((1, 200), (2, 500), (3, servo_data[servo3]),
75             (4, servo_data[servo4]), (5, servo_data[servo5]), (6, servo_data[servo6])))
76     time.sleep(1.5)

```

2. Operation Steps

i It should be case sensitive when entering command and the “Tab” key can be used to complete the keywords.

1) Turn on ArmPi Pro and connect to the system desktop via No Machine.

2) Click  Applications and select  Terminal Emulator in pop-up interface to open the terminal.

3) Firstly, stop the program that starts automatically after booting up.

Otherwise, the following programs may fail to execute. Enter command “`sudo systemctl stop start_node.service`” and press “Enter”. After executing, the terminal will not report an error.

```
ubuntu@ubuntu:~$ sudo systemctl stop start_node.service
```

4) Enter command “`roscore`” to start ros node and press “Enter”. After entering, the terminal will print prompt.

```
ubuntu@ubuntu:~$ roscore
```

```
SUMMARY
=====

PARAMETERS
* /rostdistro: melodic
* /rosversion: 1.14.13

NODES

auto-starting new master
process[master]: started with pid [20549]
ROS_MASTER_URI=http://ubuntu:11311/

setting /run_id to 14fea7c6-17b4-11ed-804e-2766002af086
process[rosout-1]: started with pid [20561]
started core service [/rosout]
```

5) Open a new terminal. Enter command “`roslaunch hiwonder_servo_controllers start.launch`” and press “Enter” to start the servo node. After executing, the

terminal will print a prompt, as the figure shown below:

```
ubuntu@ubuntu:~$ cd armpi_pro/src/arpmp_demo/kinematics_demo/

* /hiwonder_servo_manager/serial_ports: [{'fake_read': Tr...
* /rostdistro: melodic
* /rosversion: 1.14.13

NODES
/
  hiwonder_joint_states_publisher (hiwonder_servo_controllers/joint_state_publisher.py)
  hiwonder_servo_manager (hiwonder_servo_controllers/controller_manager.py)

ROS_MASTER_URI=http://ubuntu:11311

process[hiwonder_servo_manager-1]: started with pid [20744]
process[hiwonder_joint_states_publisher-2]: started with pid [20745]
[INFO] [1660030026.212543]: 1: Pinging motor IDs 1 through 6...
[INFO] [1660030026.217976]: port_id 1: Found 6 motors , initialization complete.
[INFO] [1660030026.334971]: Starting Joint State Publisher at 20Hz
```

6) Open a new terminal, enter command “cd armpi_pro/src/arpmp_demo/kinematics_demo/” and press “Enter” to enter the directory of game programmings.

```
ubuntu@ubuntu:~$ cd armpi_pro/src/arpmp_demo/kinematics_demo/
```

7) Enter command “python3 kinematics_demo.py” and press “Enter” to start the game. After entering, the terminal will print a prompt.

```
ubuntu@ubuntu:~/arpmp_demo/kinematics_demo$ python3 kinematics_demo.py
```

3. Project Outcome

After starting the game, robotic arm will return to the initial position. Next, Robotic arm will move to 0.15m along the x-axis first and then return to the initial position. Then move to 0.2m along the y-axis first and then return to the initial position. Finally, move to 0.24m along z-axis first and then return to the initial position.

4. Kinematics Analysis

4.1 Import function library

Before controlling the robotic arm, import Python function library for the movement of robotic arm.

```
4 import sys
5 import time
6 import rospy
7 from kinematics import ik_transform
8 from armpi_pro import bus_servo_control
9 from hiwonder_servo_msgs.msg import MultiRawIdPosDur
```

The “kinematics” library is mainly used here.

The packaged inverse kinematics program is called here.

```
28 ik = ik_transform.ArmlK()
```

4.2 Calculate Servo angle

Calculate the solution α of pitch angle based on the given coordinates, pitch angle and the range of pitch angle.

```
49 target = ik.setPitchRanges((0.0, 0.12, 0.15), -90, -180, 0)
# Get the solution with kinematics
```

The first parameter “(0.0, 0.12, 0.15)” is the coordinates of x, y and z axes.

The second parameter “-90” is the pitch angle.

The third and fourth parameter “-180” and “0” are the range of pitch angle.

Determine if there is a solution for calculating the pitch angle. If there is a solution, the servo can be controlled. The pulse width obtained by calculating the pitch angle is assigned to “servo_data” and the servo angle of 1, 2, 3, 4, 5, and 6 are set to control servo.

```
51 servo_data = target[1]
52 # Drive the robotic arm to move
53 bus_servo_control.set_servos(joints_pub, 1500, ((1, 200), (2, 500), (3, servo_data['servo3']),
54 (4, servo_data['servo4']), (5, servo_data['servo5']), (6, servo_data['servo6'])))
```