Miniproject-2

Hardware Implementation of Different Control Schemes on a 2R Manipulator

Submitted by

Shashank Ghosh, 21110196
Harikrishnan R, 21110071
Kanishka Varshini, 21110094
Parth Deshpande, 21110151

Department Of Mechanical Engineering
IIT Gandhinagar

Department of Mechanical Engineering
IIT Gandhinagar

Abstract

A 2R manipulator is a type of robotic arm or manipulator that consists of two revolute (rotational) joints connected by two links. Each "R" in the term "2R" represents a revolute joint, which allows for rotational motion.

The term "2R" is commonly used in robotics to describe the kinematic structure of a manipulator. In this case, it specifies that the manipulator has two rotational joints, typically referred to as "R1" and "R2," and two links connecting these joints. This configuration allows the manipulator to move in a two-dimensional plane and is often used in simple robotic arms or manipulators for tasks such as pick-and-place operations, painting, or assembly tasks.

Force control refers to the ability of the robot to control and adjust the force it applies to objects or surfaces during its operation. It is a crucial aspect of robotics that allows robots to interact with their environment in a more delicate and adaptive manner.

Contents

Chapter 1

Objective

In Miniproject 2 we are implementing Different Control Schemes like Position Control, Force Control etc. on Hardware (OS-AKE). We are using ESP 32 micro controller for controlling the motors. FSR 402 has been used for sensing the force for force control.

Chapter 2

Task 0

2.1 DH Parameters

2.1.1 Relevant Parameters of OSAKE-1

The Links Attached to the Motors have the following Parameters (Fig. 2.2,2.3) The DH parameters of the 2R Manipulator is attached in Table 2.1

Here,
$qrel_{dot}$=$q1_{dot}$-$q2_{dot}$

2.2 Jacobian

The python code for calculating the Jacobian is attached.

```
import numpy as np

#2R configuration
#All rotations are about the current z axes.
#D-H parameters d,theta,a,alpha

l=11,11.5 # link lengths

# rotation matrix about z axis
```

```
def Rzq(q_):
    R=[[np.cos(q_), -np.sin(q_), 0],
       [np.sin(q_), np.cos(q_), 0],
       [0, 0, 1]]
    return R

# rotation matrix about y axis
def Ryq(q):
    R=[[np.cos(q), 0, np.sin(q)],
       [0, 1, 0],
       [-np.sin(q), 0, np.cos(q)]]
    return R

# rotation matrix about x axis
def Rxq(q_):

    R=[[1, 0, 0],
       [0, np.cos(q_), -np.sin(q_)],
       [0, np.sin(q_), np.cos(q_)]]
    return R

# transformation matrix
def H(R=np.identity(3), d=[0,0,0]): #default no
   rotation, no translation
    H1=[[R[0][0],R[0][1], R[0][2],d[0]],
       [R[1][0],R[1][1], R[1][2],d[1]],
       [R[2][0],R[2][1], R[2][2],d[2]],
       [0,0, 0,1]]
    #print(H1
    return H1

def A(d,theta,a,al):
    d1=[0,0,d]
    a1=[a,0,0]
    print(theta)
    H1=H(np.identity(3),d1)
    H2= H(Rzq(theta))
    H3=H(np.identity(3),a1)
    H4=H(Rxq(al),[0,0,0])
    Hab=(np.linalg.multi_dot([
                H(np.identity(3),d1),
                H(Rzq(theta)),
```

```
                 H(np.identity(3),a1),
                 H(Rxq(al),[0,0,0])
                 ]))
    # Ai=[
    #      [np.cos(theta),-np.sin(theta)*np.cos(al)
    ,np.sin(theta)*np.sin(al)
    ,a*np.cos(theta)],
    #      [np.sin(theta),
    np.cos(theta)*np.cos(al),-np.cos(theta)*np.sin(al)
    ,a*np.sin(theta)],
    #      [0,np.sin(al),np.cos(al),d],
    #      [0,0,0,1]
    # ]
    return Hab

# inverse kinematics
def inv(x,y):
    theta=np.arccos((x**2+y**2-l[0]**2-l[1]**2)
    /(2*l[0]*l[1])) # in radians
    q1= np.arctan2(y, x) - np.arctan2(l[1] *
       np.sin(theta), l[0] + l[1]* np.cos(theta))
       #in radians
    q2= q1+theta
    return [q1,q2]

def endeffector(q,l):   # forward kinematics
    # transformation matrices
    q1,q2_=q[0],q[1]-q[0]   #q2_ is the relative
       angle
    # print("rel angles",q1,q2_)
    H01=A(0,q1,l[0],0)
    H12=A(0,q2_,l[1],0)

    H=[H01,H12]
    #position of end effector in 0 frame
    P=np.linalg.multi_dot([H01,H12,[0,0,0,1]])

    print("final␣position␣of␣the␣end␣effector␣with␣
       respect␣to␣the␣base␣frame␣:␣", P[0],"i␣
       +",P[1],"j␣+", P[2],"k")
    #return P, H
```

Link	d_i (in cm)	θ_i	a_i (in cm)	α
1	6.5	$q1_{dot}$	11	0
2	0	$qrel_{dot}$	11.5	0

Table 2.1: DH Parameters

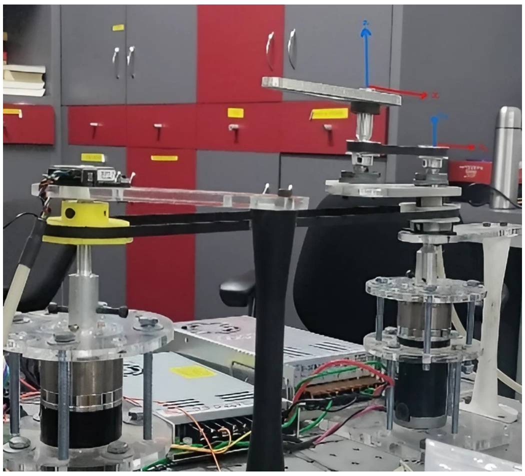


Figure 2.1: 2R Manipulator

```
# input angles to get the desired end effector
   position

for _ in range(2):
    x,y=eval(input("enter x,y coordinates: "))
    q=inv(x,y)
    endeffector(q,l)
```

Parameter	Value
Area	$16401.972\, mm^2$
Density	$0.003\, g/mm^3$
Mass	$114.18\, g$
Length	$110\, mm$
Volume	$42288.919\, mm^3$
Material	Aluminum

Figure 2.2: Parameters for Link1

Parameter	Value
Area	$10854.973\, mm^2$
Density	$0.003\, g/mm^3$
Mass	$69.65\, g$
Length	$115\, mm$
Volume	$25796.415\, mm^3$
Material	Aluminum

Figure 2.3: Parameters for Link2

Chapter 3

Task 1

Trajectory following In Task 1, We have have to trace an arbitrary trajectory. We have provided the input as [x_0,y_0]=[**0,15**] and [x_0,y_0]=[**10,10**]

The Controller follows an arbitrary trajectory to x_0,y_0.

3.1 Insights

Due to the slippage between the belt and drive pulley of Link 1, there were inconsistencies in achieving the Exact End effector position due to which there were unwanted errors.
There was also time lag between desired trajectory and actual Trajectory, tuning of the gains of the controller was a key problem as little overshoot from the value would produce undesirable results. Setting the appropriate value of PWM was

Chapter 4

Task 2

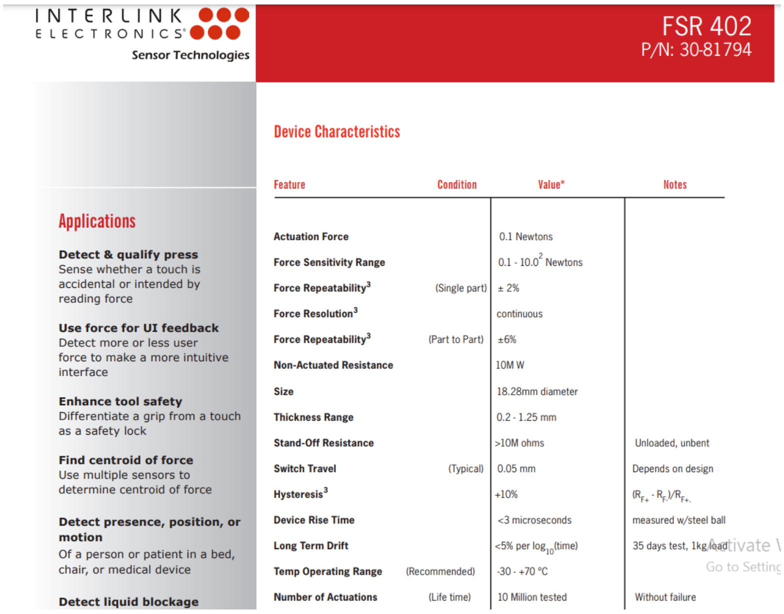
4.1 Force Control

In the context of force control, when applying an 8 Newton force in the 0i + 1j direction at the position (x = 10, y = 12), it's crucial to maintain precise control over this force. This control ensures that the robot exerts the specified force accurately, allowing for delicate or targeted interactions with objects in the vertical (y) direction at this specific location.

4.2 FSR402

We have used FSR 402 sensor to Measure the force applied by the End effector.The sensor works on the principle of change in resistivity when applied an external pressure.The details of the sensor are also shown in Datasheet attached. The code used to convert The analog readings to Force(N) is attached as follows:-

```
int fsrPin = 33;      // the FSR and 10K pulldown are
   connected to a0
int fsrReading;      // the analog reading from the
   FSR resistor divider
int fsrVoltage;      // the analog reading converted
   to voltage
```



INTERLINK ELECTRONICS

Sensor Technologies

FSR 402

P/N: 30-81794

Applications

Detect & qualify press
Sense whether a touch is accidental or intended by reading force

Use force for UI feedback
Detect more or less user force to make a more intuitive interface

Enhance tool safety
Differentiate a grip from a touch as a safety lock

Find centroid of force
Use multiple sensors to determine centroid of force

Detect presence, position, or motion
Of a person or patient in a bed, chair, or medical device

Detect liquid blockage

Device Characteristics

Feature	Condition	Value*	Notes
Actuation Force		0.1 Newtons	
Force Sensitivity Range		0.1 - 10.0[2] Newtons	
Force Repeatability[3]	(Single part)	± 2%	
Force Resolution[3]		continuous	
Force Repeatability[3]	(Part to Part)	±6%	
Non-Actuated Resistance		10M W	
Size		18.28mm diameter	
Thickness Range		0.2 - 1.25 mm	
Stand-Off Resistance		>10M ohms	Unloaded, unbent
Switch Travel	(Typical)	0.05 mm	Depends on design
Hysteresis[3]		+10%	$(R_{F+} - R_{F-})/R_{F+}$.
Device Rise Time		<3 microseconds	measured w/steel ball
Long Term Drift		<5% per $\log_{10}$(time)	35 days test, 1kg load
Temp Operating Range	(Recommended)	-30 - +70 °C	
Number of Actuations	(Life time)	10 Million tested	Without failure

Figure 4.1: FSR Dataspecs

```
unsigned long fsrResistance;  // The voltage
    converted to resistance
unsigned long fsrConductance;
double fsrForce;       // Finally, the resistance
    converted to force

void setup(void) {
  Serial.begin(9600);   // We'll send debugging
      information via the Serial monitor
}

void loop(void) {
  fsrReading = analogRead(fsrPin);
  Serial.print("Analog reading = ");
  Serial.println(fsrReading);

  // analog voltage reading ranges from about 0 to
      1023 which maps to 0V to 5V (= 5000mV)
  fsrVoltage = map(fsrReading, 0, 4095, 0, 5000);
  Serial.print("Voltage reading in mV = ");
  Serial.println(fsrVoltage);
```

```
  if(fsrVoltage == 0) {
    Serial.println("No␣pressure");
  } else {
    // The voltage = Vcc * R / (R + FSR) where R =
       10K and Vcc = 5V
    // so FSR = ((Vcc - V) * R) / V          yay math!
    fsrResistance = 5000 - fsrVoltage;       //
       fsrVoltage is in millivolts so 5V = 5000mV
    fsrResistance *= 10000;                  // 10K
       resistor
    fsrResistance /= fsrVoltage;
    Serial.print("FSR␣resistance␣in␣ohms␣=␣");
    Serial.println(fsrResistance);

    fsrConductance = 1000000;             // we
       measure in micromhos so
    fsrConductance /= fsrResistance;
    Serial.print("Conductance␣in␣microMhos:␣");
    Serial.println(fsrConductance);

    // Use the two FSR guide graphs to approximate
       the force
    if (fsrConductance <= 1000) {
      fsrForce = fsrConductance / 80;
      Serial.print("Force␣in␣Newtons1:␣");
      Serial.println(fsrForce);
    } else {
      fsrForce = fsrConductance - 1000;
      fsrForce /= 30;
      Serial.print("Force␣in␣Newtons2:␣");
      Serial.println(fsrForce);
    }
  }
  Serial.println("--------------------");
  delay(10);
}
```

4.3 Setup and Insights

We used a force sensor and attached it to an MDF sheet to try and measure the force reaction, as seen in the image and the video. We used feedback control to try to analyze the behaviour of the manipulator after force application, but we ran into issues. The motor controlling link 1 stalled and refused to move even after inputting high rpms. This caused the link not to move and touch the sensor. The force sensor also gave very high values of force when not in use and interfered with the force control function. The entire phenomenon is documented in the video to show where the setup failed.

Chapter 5

Task 3

5.1 Spring Behaviour

To make the 2R bot behave like a Linear spring. We used Mean position [x_0,y_0] as [5,5].

5.2 Insights

The Friction was variable on both the pulleys. Also the Current Sensor wasn't working on one of the motors which caused issues in taking the reading values. Our Code theoretically should run properly but the manipulator wasnt responsive at low deviations which caused the task to fail.

Chapter 6

Project_Repository

Here is the Repository containing all the data.

1. Task Codes
2. TestVideos