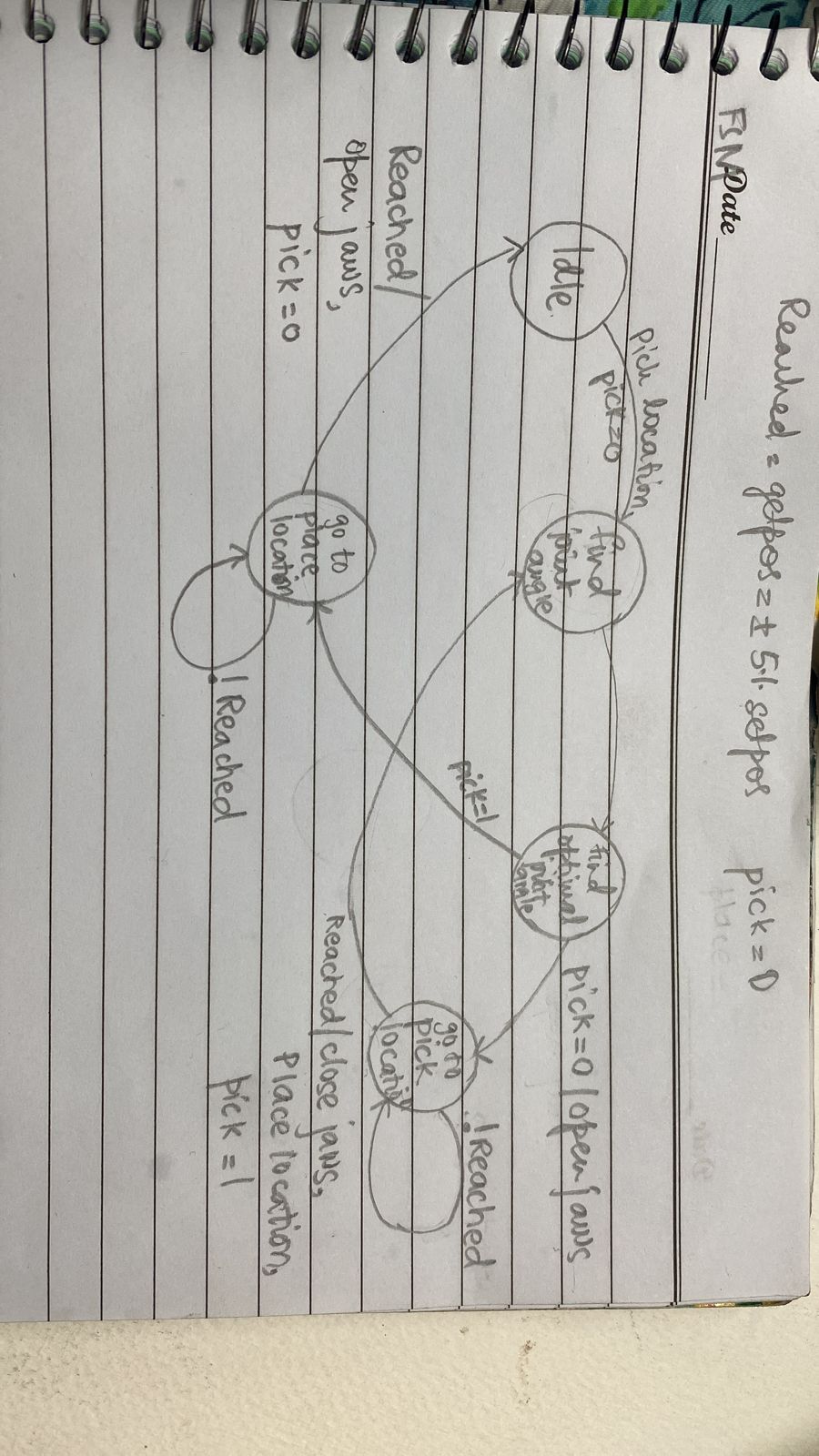
**Report Introduction to Robotics LAB6**

**Nimra Sohail ns06867**

**Areeb Adnan Khan ak06865**

Task 6.1



Task 6.2

%%code for the pick and place

%set the z-axis to be minimum 60 or 50

pick\_coordinates = [150,180,60,0];

place\_coordinates = [-150,-180,50,0];

current\_state = 0

while (1)

if current\_state == 0 %%This means that the robot is in the idle condition ready to move

pick = 0;

setAngles(arb,[0 0 0 0 0 0]); %%initial state details of this function will be explained later on.

location = [-pi/2 pi/2 0 0 0 0]

location1 = location - 0.05 %this is done because arb.getpos doesnt actually matches with the calculated angles it has some error

location2 = location + 0.05 %upper limit of the error

if (arb.getpos>location1 & arb.getpos<location2)

current\_state = current\_state + 1

else

current\_state = 0;

end

elseif current\_state == 1

if pick == 0

solutions = findJointAngles\_O(pick\_coordinates(1),pick\_coordinates(2),pick\_coordinates(3),pick\_coordinates(4));

elseif pick == 1

solutions = findJointAngles\_O(place\_coordinates(1),place\_coordinates(2),place\_coordinates(3),place\_coordinates(4));

end

current\_state = current\_state + 1;

elseif current\_state == 2

final\_sol = optimial\_org(solutions);

%after finding the optimal solution we pass this to the setAngles

%fuction

if (pick == 0)

setAngles(arb,final\_sol);

location = final\_sol;

location1 = location - 0.05 %this is done because arb.getpos doesnt actually matches with the calculated angles it has some error

location2 = location + 0.05 %upper limit of the error

if (arb.getpos>location1 & arb.getpos<location2)

setAngles(arb,[final\_sol(1),final\_sol(2),final\_sol(3),final\_sol(4),-1.5]); % this is done so that the jaws are closed

pick = 1;

current\_state = 1;

end

elseif pick == 1

setAngles(arb,final\_sol);

location = final\_sol;

location1 = location - 0.05 %this is done because arb.getpos doesnt actually matches with the calculated angles it has some error

location2 = location + 0.05 %upper limit of the error

if (arb.getpos>location1 & arb.getpos<location2)

setAngles(arb,[final\_sol(1),final\_sol(2),final\_sol(3),final\_sol(4),0]); % this is done so that the jaws are open

pick = 0;

current\_state = 0;

end

end

end

end

additional functions:

function setAngles(arb,angles)

theta1 = error1(angles(1)+pi/2);

theta2 = error1(angles(2)-pi/2);

theta3 = angles(3);

theta4 = angles(4);

theta5 = angles(5);

arb.setpos([theta1,theta2,theta3,theta4,theta5],[50 50 50 50 50]);

end

function solution = optimial\_org(allsolutions)

%before this we have already add the mod with all the 4 solutions and if

%any of the 4 soltuions is not within the range of -150 to 150 it will be

%dropped

%position = arb.getpos();

position = [pi/2 -pi/2 pi/4 0];

position(1) = position(1) - pi/2;

position(2) = position(2) + pi/2;

error = 0;

a = [];

for i=1:4

error = abs(allsolutions(i,1)-position(1))+ abs(allsolutions(i,2)-position(2)) + abs(allsolutions(i,3)-position(3)) + abs(allsolutions(i,4)-position(4));

a(i)=error;

end

perfect = find(min(a));

solution = allsolutions(perfect,:);

end

function [corr\_theta] = error1(theta)

corr\_theta = mod(theta+pi,2\*pi) - pi;

if ((corr\_theta > 150\*(pi/180) && corr\_theta < pi) | (corr\_theta < -150\*(pi/180) && corr\_theta > -pi))

disp("Angles limits are outside 150 degrees or -150 degrees")

disp([theta])

end

end

Video Link: [Robot Placement.mp4](https://habibuniversity-my.sharepoint.com/:v:/g/personal/ak06865_st_habib_edu_pk/ESKbvgbwphxGutZhTQ5jHg8BhlZi9ikK1vhhlMHE53CaMw?e=o6KeVZ)

Strategy:

The strategy was to execute the fsm in the most simple way possible which is why the 5 states were reduced to just 3 states. First of all, we check the state of the fsm if it is in the 0 state then it means it is in idle state, after this, the state will move to 1 checking that whether the pick is 0 or 1. If the pick is 0 then this means the robotic arm needs to go to the place location where it will grip the cube. If the pick is 1 then this means that the robotic arm will go to place the cube at the place coordinates . It will make use of the findjointAngles and optimal solution functions developed before. The function SetAngles add the offset and then applies the mod through the use of the error1 function so that our DH frames and servo motors (physical) configuration are aligned.

Task 6.3

clc;clear;

%first find A1 A2 A3 A4

syms theta\_1(t) theta\_2(t) theta\_3(t) theta\_4(t);

A1 = createA(theta\_1(t),142,0,pi/2)

A2 = createA(theta\_2(t),0,104,0)

A3 = createA(theta\_3(t),0,104,0)

A4 = createA(theta\_4(t),0,80,0)

T0\_4=simplify(A1\*A2\*A3\*A4)

%NOW WE WILL EXTRACT THE LAST COLUMN OF THIS HOMOGENOUS TRANFORMATION TO

%TAKE POSITION VECTOR OF END EFFECTOR

%syms Jv;syms Jw;

%J=[Jv;Jw];

position\_vector=T0\_4(1:3,4)

%NOW WE WILL POPULATE THE Jv mattrix by differentiating the positin vector

%WITH RESPECT TO EACH THETA

col\_1 =diff(position\_vector,theta\_1(t));

col\_2 =diff(position\_vector,theta\_2(t));

col\_3 =diff(position\_vector,theta\_3(t));

col\_4 =diff(position\_vector,theta\_4(t));

Jv = [col\_1 col\_2 col\_3 col\_4]

simplify(Jv)

%%NOW FINDING THE Jw 3 by 3 matrix

col1 = A1(:,3);

col2 = A2(:,3);

col3 = A3(:,3);

col4 = A4(:,3);

Jw=[col1 col2 col3 col4]

J=[Jv;Jw]%final jacobian

Output:

J =

[-8\*sin(theta\_1(t))\*(13\*cos(theta\_2(t)) + 10\*cos(theta\_2(t) + theta\_3(t) + theta\_4(t)) + 13\*cos(theta\_2(t) + theta\_3(t))), -8\*cos(theta\_1(t))\*(13\*sin(theta\_2(t)) + 10\*sin(theta\_2(t) + theta\_3(t) + theta\_4(t)) + 13\*sin(theta\_2(t) + theta\_3(t))), -8\*cos(theta\_1(t))\*(10\*sin(theta\_2(t) + theta\_3(t) + theta\_4(t)) + 13\*sin(theta\_2(t) + theta\_3(t))), -80\*cos(theta\_1(t))\*sin(theta\_2(t) + theta\_3(t) + theta\_4(t))]

[ 8\*cos(theta\_1(t))\*(13\*cos(theta\_2(t)) + 10\*cos(theta\_2(t) + theta\_3(t) + theta\_4(t)) + 13\*cos(theta\_2(t) + theta\_3(t))), -8\*sin(theta\_1(t))\*(13\*sin(theta\_2(t)) + 10\*sin(theta\_2(t) + theta\_3(t) + theta\_4(t)) + 13\*sin(theta\_2(t) + theta\_3(t))), -8\*sin(theta\_1(t))\*(10\*sin(theta\_2(t) + theta\_3(t) + theta\_4(t)) + 13\*sin(theta\_2(t) + theta\_3(t))), -80\*sin(theta\_1(t))\*sin(theta\_2(t) + theta\_3(t) + theta\_4(t))]

[ 0, 104\*cos(theta\_2(t)) + 80\*cos(theta\_2(t) + theta\_3(t) + theta\_4(t)) + 104\*cos(theta\_2(t) + theta\_3(t)), 80\*cos(theta\_2(t) + theta\_3(t) + theta\_4(t)) + 104\*cos(theta\_2(t) + theta\_3(t)), 80\*cos(theta\_2(t) + theta\_3(t) + theta\_4(t))]

[ sin(theta\_1(t)), 0, 0, 0]

[ -cos(theta\_1(t)), 0, 0, 0]

[ 0, 1, 1, 1]

[ 0, 0, 0, 0]