

**TC**

**SELCUK ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ**

**LİSANS PROGRAMI**

**3 EKSENLİ ROBOTİK KOL SİMÜLASYONU**

**Mustafa ÇETİN**

**141225002**

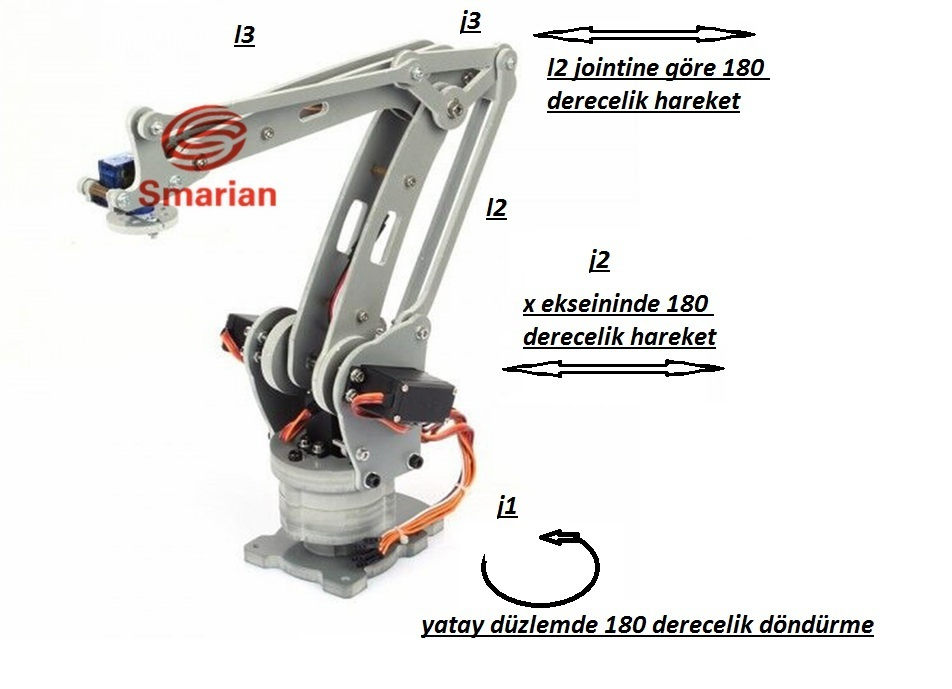
**DANIŞMAN**

**Dr.öğr AKİF DURDU**

**KONYA/2018**

3 EKSENLİ ROBOTİK KOL SİMÜLASYONU

Üç eksende hareket edebilen, üç eklemli bir robotik kol simülasyonudur.



şekilde görüldüğü gibi yatay eksende 180 derecelik bir acı ile sağa ve sola hareket edebilen ana bağlantı (j1) bu bağlantıya bağlı bir eklem (l1)’ yi ve tüm gövdeyi hareket ettirmektedir. 2. Bağlantı j2 ise x eksenine acı yapacak şekilde 180 derecelik bir alanda hareket ederek l2 kolu ile l3 kolunu hareket ettiriyor. 3. Bağlantı j3 ise sadece l3 kolunu hareket ettiriyor ve bu hareketi sadece l2 kolu ile yaptığı acı ile yapmaktadır. Ana bağlantıdan itibaren motorlara gerekli acı değerleri verildiğinde robot kolumuz yatay düzlemde 180 derecelik sağa sola dönüşle beraber l2 ve l3 kollarının uzunluğu kadar bir alana ulaşabilecek şekilde hareket etmektedir.

Bu bağlantılar aşağıda kodlarda nasıl olacağı açıklanmıştır.

GUI\_VSMP2O kodu;

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @GUI\_VSMP2O\_OpeningFcn, ...

'gui\_OutputFcn', @GUI\_VSMP2O\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function GUI\_VSMP2O\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

jinit=[90;60;-120];

assignin('base','jinit',jinit);

axes(handles.axes1);

FKdraw(jinit(1,1),jinit(2,1),jinit(3,1));

ax\_properties = get(gca);

assignin('base','pov',ax\_properties.View);

function varargout = GUI\_VSMP2O\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function pushbutton1\_Callback(hObject, eventdata, handles)

axes(handles.axes1);

jinit=evalin('base','jinit');

FKdraw(jinit(1,1),jinit(2,1),jinit(3,1));

set( handles.text1,'String', num2str(90) );

set( handles.text2,'String', num2str(60) );

set( handles.text3,'String', num2str(-120) );

function pushbutton3\_Callback(hObject, eventdata, handles)

val1=str2double(get(handles.edit1,'String'));

val2=str2double(get(handles.edit2,'String'));

val3 = str2double(get(handles.edit3,'String'));

val=[val1;val2;val3];

assignin('base','val',val);

set( handles.text3,'String', num2str(val3,3) );

set( handles.text2,'String', num2str(val2,3) );

set( handles.text1,'String', num2str(val1,3) );

axes(handles.axes1);

FKdraw(val1,val2,val3);

view(evalin('base','pov'));

function edit1\_Callback(hObject, eventdata, handles)

function edit1\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit2\_Callback(hObject, eventdata, handles)

function edit2\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit3\_Callback(hObject, eventdata, handles)

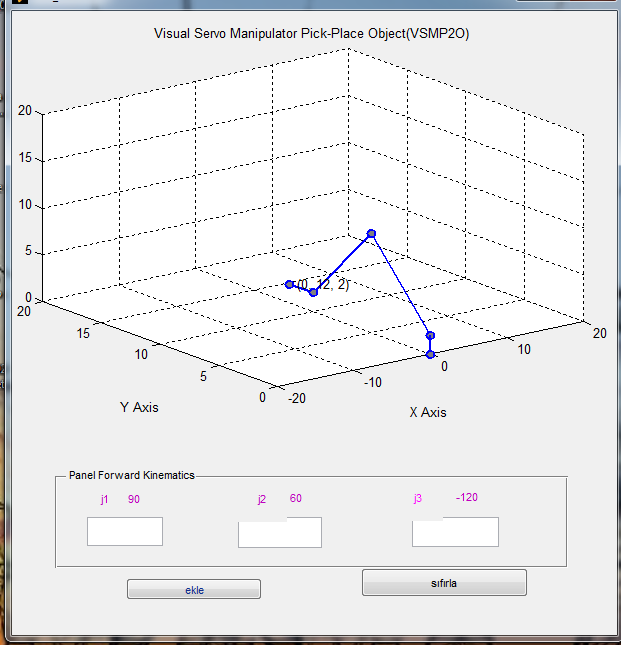
function edit3\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

Verilen bu kod çalıştırıldığında bizim yazmış olduğumuz ara yüz ekrana geliyor ve biz buradan j1,j2 ve j3 açılarını ayarlayabiliyoruz. Bu ara yüzü GUI\_VSMP2O.fig olarak kaydediyoruz.



Grafikte eklemler ilk ayarladığımız açı derecelerinde gözüküyor. Yeni değerler girip ekle dediğimizde yeni konumlarını XYZkine programı sayesinde göreceğiz.

function [Q] = XYZkine(FK)

%DRAWKINE Summary of this function goes here

% Detailed explanation goes here

%P=[FK(1:3,4) FK(1:3,8) FK(1:3,12) FK(1:3,16)];

Q1=[0 FK(1,4) FK(1,8) FK(1,12) FK(1,16)];

Q2=[0 FK(2,4) FK(2,8) FK(2,12) FK(2,16)];

Q3=[0 FK(3,4) FK(3,8) FK(3,12) FK(3,16)];

Q=[Q1;Q2;Q3];

End

**DHkine.m** fonksiyonu eklem açılarını, eklem kayması, bağlantı uzunlukları ve büküm açısını hesaplayan fonksiyondur.

function [ FK ] = DHkine(j)

%collum 1=joint angle, 2=joint offset, 3=link lenght, 4=twist angle

T01=DHmatrix(j(1,1),j(2,1),j(3,1),j(4,1));

T12=DHmatrix(j(1,2),j(2,2),j(3,2),j(4,2));

T23=DHmatrix(j(1,3),j(2,3),j(3,3),j(4,3));

T34=DHmatrix(j(1,4),j(2,4),j(3,4),j(4,4));

T02=T01\*T12;

T03=T02\*T23;

T04=T03\*T34;

FK=[T01 T02 T03 T04];

End

**DHmatri.m** bu kod DHkine.m fonksiyonun hesaplanmasında kullanılan matrisi içerir ve buradaki matrise göre DHkine.m fonksiyonu çalışır.

function [Mdh] = DHmatrix(theta,d,a,alpha)

%DHMATRIX Summary of this function goes here

% Detailed explanation goes here

% inputannya DH parameter yaa

Mdh=[cosd(theta) -sind(theta)\*cosd(alpha) sind(theta)\*sind(alpha) a\*cosd(theta);

sind(theta) cosd(theta)\*cosd(alpha) -cosd(theta)\*sind(alpha) a\*sind(theta);

0,sind(alpha),cosd(alpha),d;

0,0,0,1];

end

**fk draw.m** fonksiyonu alınan bilgilerin hepisini toplayıp i,şlem yaparak en son ekranda ki görseli bize veren fonksiyondur.

function [ FK ] = FKdraw( j1,j2,j3 )

%initial parameter

%[j0 j1 j2 j3;d0 d1 d2 d3;a0 a1 a2 a3;t0 t1 t2 t3]

%theta3=360-theta1-theta2

j4=360-(j2+j3);

j=[j1 j2 j3 j4;2 0 0 0;0 10 10 2;90 0 0 0];

FK=DHkine(j);

Q=XYZkine(FK);

%J=Jacobian(FK); %Not Yet, it use for IK

%dj=deltaJ(dxyz,J);

plot3(Q(1,:),Q(2,:),Q(3,:),'-o','LineWidth',2,'MarkerSize',6,'MarkerFaceColor',[0.5,0.5,0.5]);grid on;%axis([-31,31,-31,31,0,31]);

text(Q(1,5),Q(2,5),Q(3,5),[' (', num2str(Q(1,5),3), ', ', num2str(Q(2,5),3),', ', num2str(Q(3,5),3), ')']);

title('Visual Servo Manipulator Pick-Place Object(VSMP2O)')

xlabel('X Axis');

ylabel('Y Axis');

zlabel('Z Axis');

axis([-20 20 0 20 0 20]);

h = rotate3d;

h.Enable = 'on';

h.ActionPostCallback = @mypostcallback;

assignin('base','FK',FK);

end

function mypostcallback(obj,evd)

%disp('A rotation is about to occur.');

ax\_properties = get(gca);

assignin('base','pov',ax\_properties.View);

end

%use evalin('base',a) to get variable a from workspace

%assignin('base','a\_rms',a\_rms) to write variable a\_rms to workspace