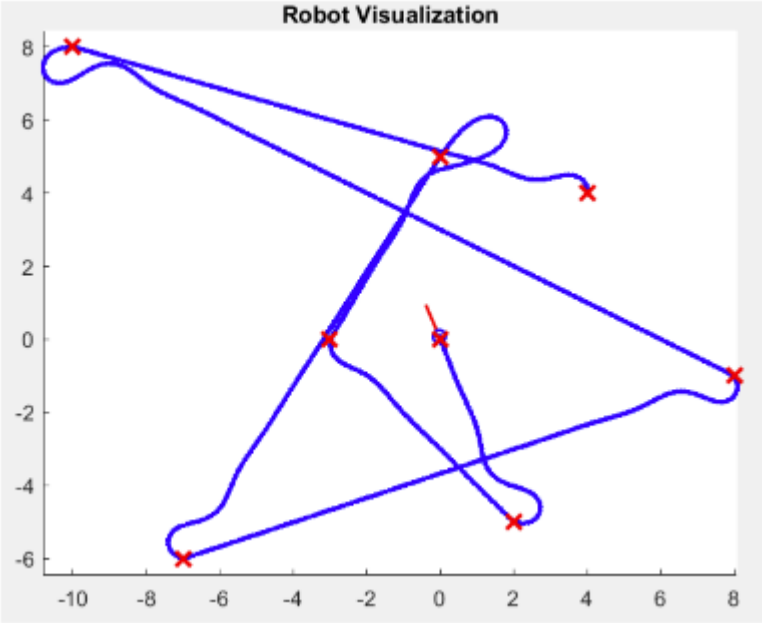
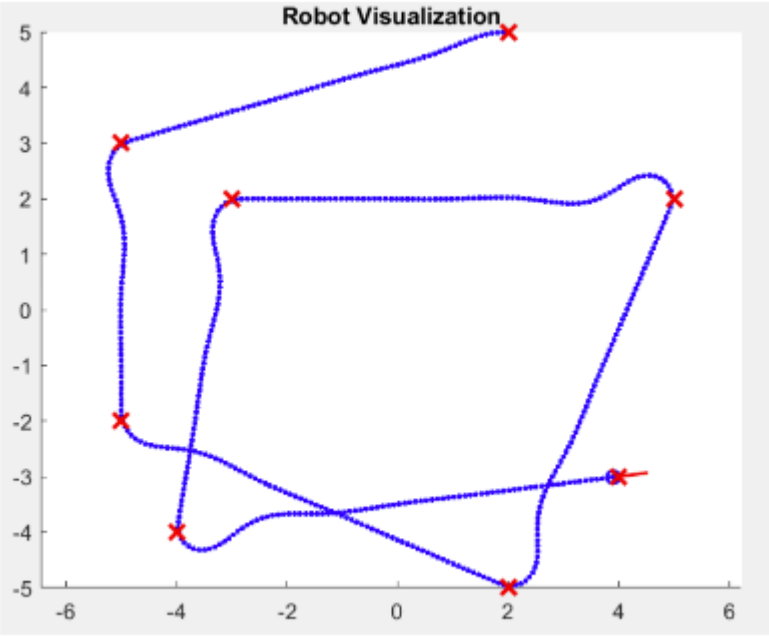
```
% Mariam Landa
%A01736672


R = 0.1;
L = 0.5;
dd = DifferentialDrive(R,L);

%% Simulation parameters
sampleTime = 0.05;               % Sample time [s]
tVec = 0:sampleTime:45.7;         % Time array

%CAMBIAR CONFORME A LAS ESPECIFICACIONES DE CADA TRAYECTORIA O DIBUJO
initPose = [4;4;pi/2];             % Initial pose (x y theta)
pose = zeros(3,numel(tVec));    % Pose matrix
pose(:,1) = initPose;

% Documente las trayectorias, pero cuando las quiero probar lo que hago es
% quitarle el comentario a la trayectoria, para que solo checarlas que se
% implementen correctamente.

%--------- Primera trayectoria ----------------------------------------
%se declaran waypoints par seguir la trayectoria, se cambia la posición
%inicial conforme al angulo
%initPose = [4;4;pi/2];
%sample time = 0.05
%tvec = 45.7
%lookaheadDistance = 0.4
%linear velocity = 2
%angular velocity = 10
%waypoints = [4,4; -10,8; 8,-1; -7,-6; 0,5; -3,0; 2,-5; 0,0];
```

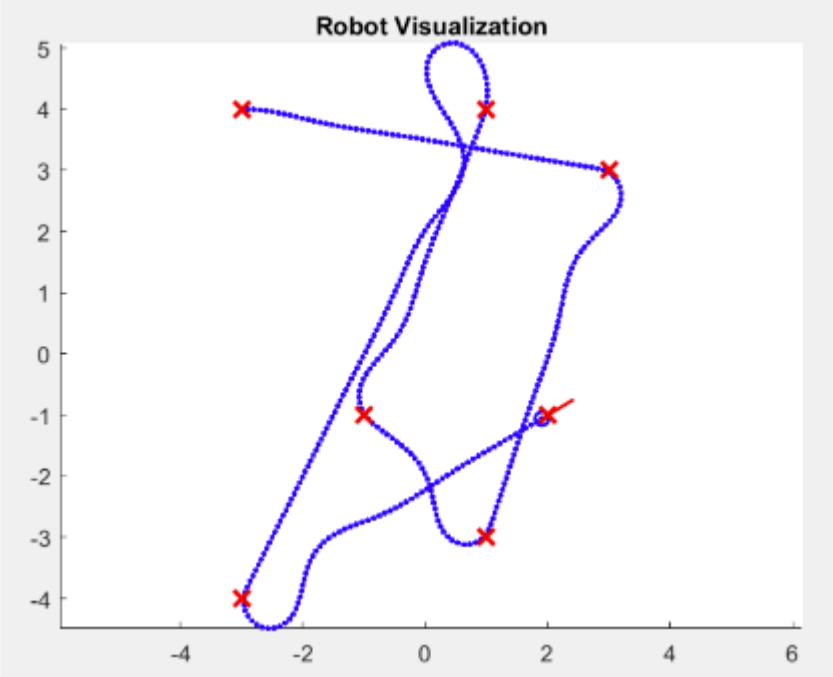



```
%--------- Segunda trayectoria ----------------------------------------
%initPose = [2;5;2*pi/2];
%sample time = 0.05
%tvec = 25.4
%lookaheadDistance = 0.4
%linear velocity = 2
%angular velocity = 10
%waypoints = [2,5; -5,3; -5,-2; 2,-5; 5,2; -3,2; -4,-4; 4,-3];
```



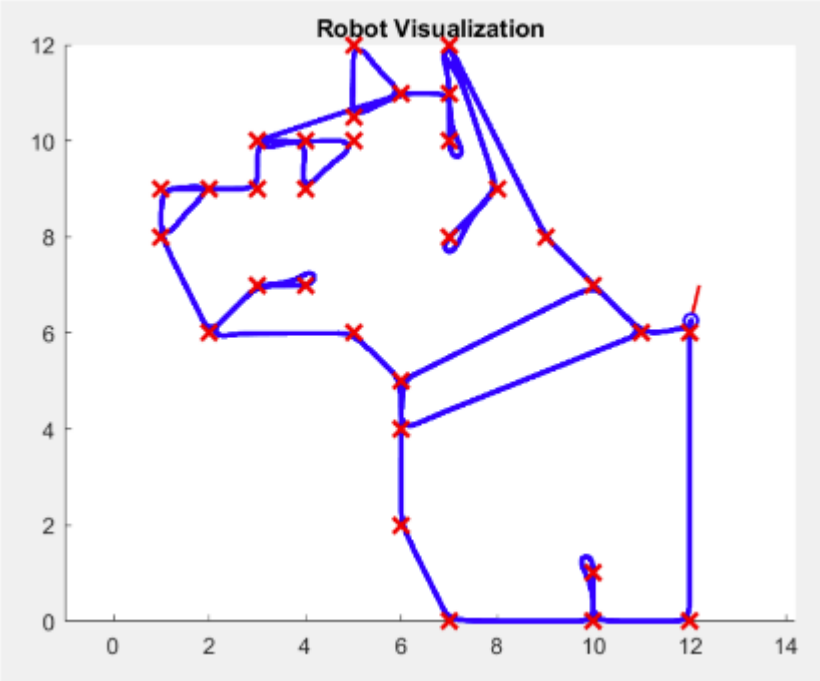

```
%---------- Tercera trayectoria ----------------------------------------
%initPose = [-3;4;0];
```

```
%sample time = 0.05
%tvec = 19.8
%lookaheadDistance = 0.4
%linear velocity = 2
%angular velocity = 10
%waypoints = [-3,4; 3,3; 1,-3; -1,-1; 1,4; -3,-4; 2,-1];
```

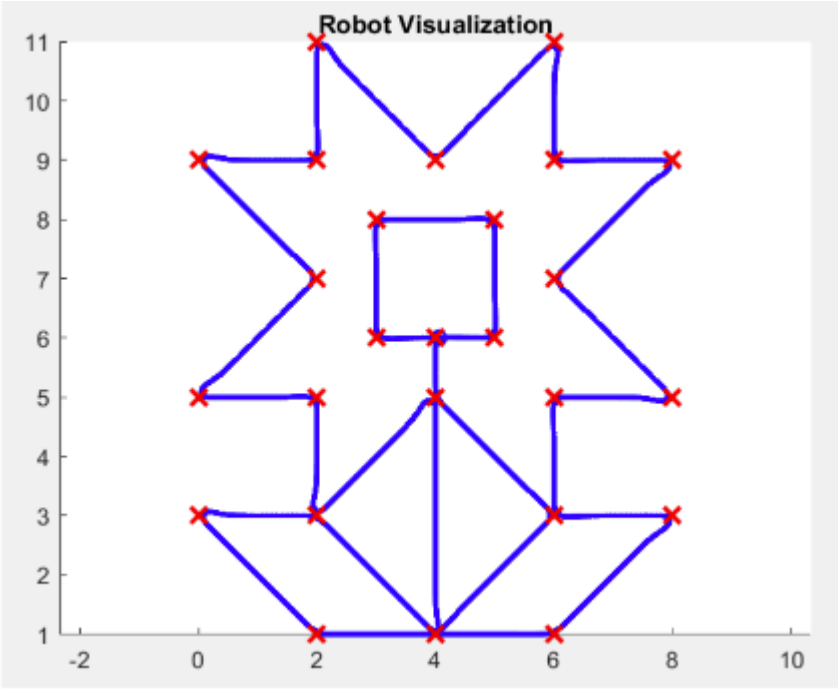


El perro se ajusto a modo que hiciera bien los giros, la velocidad inicial se bajo y la velocidad angular necesita un valor.

```
% ---------- Trayectoria del perro -------------------------------------------
%initPose = [12;6;pi/2];
%sample time = 0.05
%tvec = 185
%lookaheadDistance = 0.36
%linear velocity = 0.45
%angular velocity = 10
%waypoints = [12,6; 11,6; 10,7; 9,8; 7,12; 8,9; 7,8; 8,9; 7,12; 7,10; 7,11;
%    6,11; 5,10.5; 5,12; 6,11; 3,10; 4,10; 5,10; 4,9; 4,10; 3,10; 3,9; 2,9;
%    1,9; 1,8; 2,9; 1,9; 1,8; 2,6; 3,7; 4,7; 3,7; 2,6; 5,6; 6,5; 6,4; 11,6;
%    10,7; 6,5; 6,2; 7,0; 10,0; 10,1; 10,0; 12,0; 12,6]
```



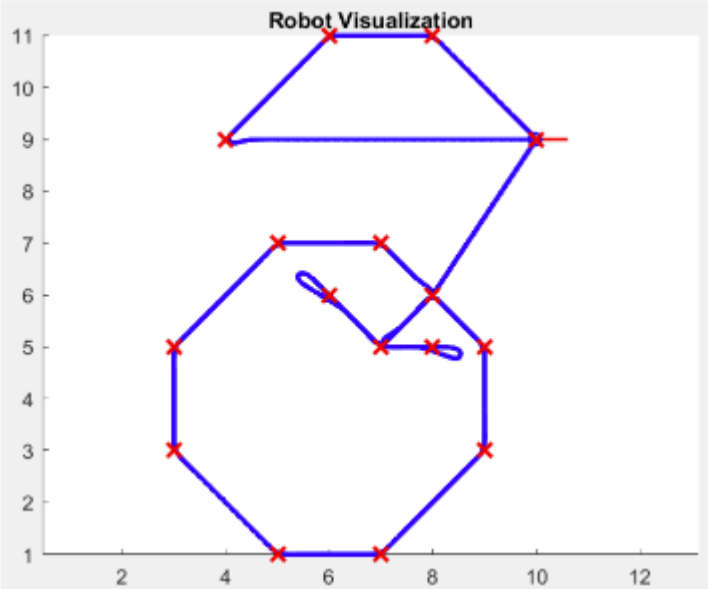

```
% ---------- Trayectoria de la flor
-------------------------------------------------
%initPose = [4;1;pi/4];
%sample time = 0.05
%tvec = 115
%lookaheadDistance = 0.25
%linear velocity = 0.6
%angular velocity = 10
%waypoints = [4,1; 4,5; 2,3; 2,5; 0,5; 2,7; 0,9; 2,9; 2,11; 4,9; 6,11; 6,9; 8,9;
%  6,7; 8,5; 6,5; 6,3; 8,3; 6,1; 2,1; 0,3; 2,3; 4,1; 6,3; 4,5; 4,6; 5,6; 5,8; 3,8;
3,6; 5,6]
```



Hay una velocidad lineal mas rapida, ya que no esta tan detallado, la orientación es pi/4 ya que fue la mejor que se ajusto.

```
% ---------- Trayectoria de la cereza
-----------------------------------------------
%initPose = [8;6;pi/2];
%sample time = 0.05
%tvec = 71.5
%lookaheadDistance = 0.22
%linear velocity = 0.65
%angular velocity = 27
%waypoints = [8,6; 7,7; 5,7; 3,5; 3,3; 5,1; 7,1; 9,3;
%    9,5; 8,6; 7,5; 6,6; 7,5; 8,5; 7,5; 8,6; 10,9; 8,11; 6,11; 4,9; 10,9]
```

Se declaro la orientación en pi/2, el valor de lookheadDistance más bajo, aumente la velocidad lineal y tambien la angular.




```
% Create visualizer
viz = Visualizer2D;
viz.hasWaypoints = true;

%% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
%muy largo no sabremos a que way point va a ir
controller.LookaheadDistance = 0.4; %muy corto no vera el waypoint
controller.DesiredLinearVelocity = 2;
controller.MaxAngularVelocity = 10;

%% Simulation loop
close all
r = rateControl(1/sampleTime);
for idx = 2:numel(tVec)
    % Run the Pure Pursuit controller and convert output to wheel speeds
    [vRef,wRef] = controller(pose(:,idx-1));
    [wL,wR] = inverseKinematics(dd,vRef,wRef)


    % Compute the velocities
    [v,w] = forwardKinematics(dd,wL,wR);
    velB = [v;0;w]; % Body velocities [vx;vy;w]
    vel = bodyToWorld(velB,pose(:,idx-1));  % Convert from body to world

    % Perform forward discrete integration step
    pose(:,idx) = pose(:,idx-1) + vel*sampleTime;

    % Update visualization
    viz(pose(:,idx),waypoints)
```

```
    waitfor(r);

end
```