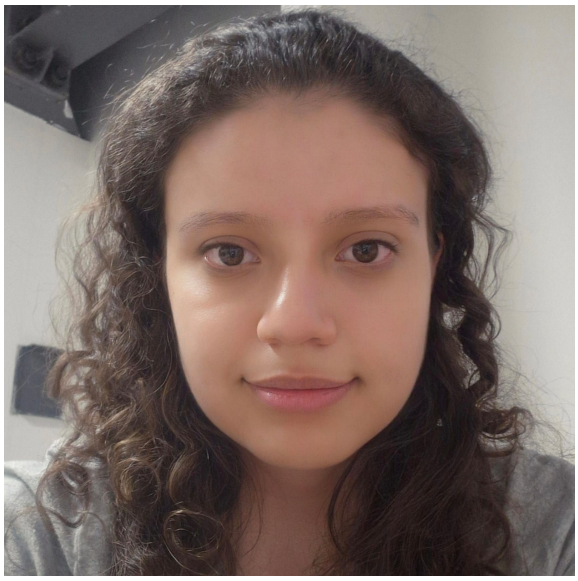
Evaluación 9: Planeación de Trayectorias

Paola Rojas Domínguez A01737136

Vide de evidencia: https://youtu.be/TMHGQ3cO0UU

Este script simula 15 robots diferenciales, cada uno siguiendo su propia trayectoria mediante controladores `controllerPurePursuit`. Al finalizar la simulación, se grafican las trayectorias ejecutadas por cada robot para recrear un rostro.



Inicialización

Se inicia limpiando la consola y la memoria para evitar conflictos de datos anteriores.

```
clear all
close all
clc
```

Definir vehículo

```
R = 0.1; % radio de rueda [m]
L = 0.5; % distancia entre ruedas [m]
dd = DifferentialDrive(R, L);
```

Parámetros de simulación

Se establecen los parámetros temporales para la simulación: tiempo de muestreo (`sampleTime`) y vector de tiempo (`tVec`).

```
sampleTime = 0.1;
r = rateControl(1/sampleTime);
t = 0;
```

Se definen variables booleanas para cada robot, las cuales servirán para determinar si un robot ha llegado a su destino o ha completado su trayectoria.

```
finished1 = false;
finished2 = false;
finished3 = false;
finished4 = false;
finished5 = false;
finished6 = false;
finished7 = false;
finished8 = false;
finished9 = false;
finished10 = false;
finished11 = false;
finished12 = false;
finished13 = false;
finished14 = false;
finished15 = false;
```

La variable `goalTolerance` se establece en 0.1. Esta tolerancia define cuán cerca debe estar un robot de su objetivo para considerar que ha llegado.

```
goalTolerance = 0.1;
```

Configuración de los robots

Se crean múltiples robots, cada uno con su posición inicial (`initPose`) y su trayectoria (`waypoints`). El control de cada robot se realiza mediante un controlador `PurePursuit`, que es un algoritmo de control de trayectoria basado en la búsqueda de un punto de referencia a una distancia específica en la dirección de movimiento. Se configuran los parámetros de este controlador, como la distancia de anticipación (`LookaheadDistance`), la velocidad lineal deseada (`DesiredLinearVelocity`), y la velocidad angular máxima (`MaxAngularVelocity`).

```
% Robot 1 - Sonrisa
initPose1 = [20.86; 17.34; -pi/3];
pose1 = initPose1;
traj1 = pose1;
waypoints1 = [20.86, 17.34; 21.77,16.78; 23.7,16.46; 25.7,16.37; 27.22,16.4;
29.92,16.75; 30.8,17.36];
controller1 = controllerPurePursuit;
controller1.Waypoints = waypoints1;
controller1.LookaheadDistance = 0.5;
controller1.DesiredLinearVelocity = 1.0;
controller1.MaxAngularVelocity = 2.0;

% Robot 2 - Labios
initPose2 = [21.77; 16.78; 0];
pose2 = initPose2;
traj2 = pose2;
```

```
waypoints2 = [21.77,16.78; 23.12,17.25; 24.71,17.58; 25.75,17.16; 26.39,17.52;
27.98,17.22; 29.92,16.75; ...
              29.38,15.66; 27.63,14.77; 25.7,14.65; 23.66,14.94; 22,16;
21.77,16.78];
controller2 = controllerPurePursuit;
controller2.Waypoints = waypoints2;
controller2.LookaheadDistance = 0.5;
controller2.DesiredLinearVelocity = 0.8;
controller2.MaxAngularVelocity = 10.0;

% Robot 3 - Cara
initPose3 = [31.11; 11.13; pi/3];
pose3 = initPose3;
traj3 = pose3;
waypoints3 = [31.11, 11.13; 33.78, 13.41; 35.31, 15.22; 36.84, 18.95; 37.29, 21.51;
37.38, 25.05; 37.16, 29.06; 37.12, 32.82; 37.12, 35.67; 35.3, 37.35; 32.89, 39.94;
30, 42; 27.96, 40.81; 25.67, 40.59; 23.21, 42.06; 20.19, 41.02; 16.87, 39.21;
14.97, 37.09; 13.97, 34.33; 13.94, 32.67; 14.27, 30.04; 14.16, 26.82; 14.54, 23.21;
15.31, 19.32; 16.4, 15.99; 17.71, 13.74; 19.3, 12.48; 21, 11.17; 22.75, 9.91;
24.94, 9.37; 28.49, 9.53; 31.11, 11.13];
controller3 = controllerPurePursuit;
controller3.Waypoints = waypoints3;
controller3.LookaheadDistance = 0.5;
controller3.DesiredLinearVelocity = 1.0;
controller3.MaxAngularVelocity = 2.0;

% Robot 4
initPose4 = [29.06; 28.73; pi/3];
pose4 = initPose4;
traj4 = pose4;
waypoints4 = [29.06, 28.73; 29.8, 29.6; 30.91, 30.09; 32.1, 30.29; 33.15, 30.23;
34, 30];
controller4 = controllerPurePursuit;
controller4.Waypoints = waypoints4;
controller4.LookaheadDistance = 0.5;
controller4.DesiredLinearVelocity = 1.0;
controller4.MaxAngularVelocity = 2.0;

% Robot 5
initPose5 = [22.57; 28.16; 2*pi/3];
pose5 = initPose5;
traj5 = pose5;
waypoints5 = [22.57, 28.16; 22, 29; 20.92, 29.66; 19.85, 29.79; 18.99, 29.73; 18,
29.5; 17.24, 28.87];
controller5 = controllerPurePursuit;
controller5.Waypoints = waypoints5;
controller5.LookaheadDistance = 0.5;
controller5.DesiredLinearVelocity = 1.0;
controller5.MaxAngularVelocity = 2.0;
```

```
% Robot 6 - Iris izq
initPose6 = [30.57; 28.97; -pi/2];
pose6 = initPose6;
traj6 = pose6;
waypoints6 = [30.57, 28.97; 30.58, 28.74; 30.56, 28.48; 30.61, 28.19; 30.8, 27.92;
31, 27.75; 31.37, 27.56; 31.83, 27.53; 32.24, 27.68; 32.52, 27.92; 32.65, 28.19;
32.72, 28.52; 32.66, 28.94; 32.35, 29.24; 32.64, 29.1];
controller6 = controllerPurePursuit;
controller6.Waypoints = waypoints6;
controller6.LookaheadDistance = 0.5;
controller6.DesiredLinearVelocity = 1.0;
controller6.MaxAngularVelocity = 2.0;

% Robot 7 - Iris der
initPose7 = [20.77; 28.75; -pi/3];
pose7 = initPose7;
traj7 = pose7;
waypoints7 = [20.77, 28.75; 20.91, 28.46; 20.93, 28.09; 20.85, 27.71; 20.54, 27.47;
20.33, 27.32; 19.8, 27.26; 19.4, 27.4; 19.13, 27.6; 18.96, 27.86; 18.83, 28.3;
18.94, 28.6; 19.16, 28.78; 19.38, 28.93; 19.71, 28.96];
controller7 = controllerPurePursuit;
controller7.Waypoints = waypoints7;
controller7.LookaheadDistance = 0.5;
controller7.DesiredLinearVelocity = 1.0;
controller7.MaxAngularVelocity = 2.0;

% Robot 8 - Nariz
initPose8 = [24.47; 27.43; -pi/2];
pose8 = initPose8;
traj8 = pose8;
waypoints8 = [24.47, 27.43; 24.42, 26.23; 24.22, 25.27; 23.98, 24.44; 23.27, 23.73;
22.8, 23.07; 22.24, 22.31; 22.09, 21.53; 22.21, 20.84; 22.56, 20.52; 23.09, 20.38;
23.72, 20.29; 23.14, 20.69; 23.4, 20.8; 23.75, 20.79; 24.04, 20.58; 24.34, 20.52;
24.77, 20.31; 25.27, 20.24; 25.06, 20.28; 26.11, 20.41; 26.55, 20.59; 26.92, 20.78;
27.39, 20.84; 27.77, 20.77; 27.52, 20.28; 28.09, 20.2; 28.67, 20.37; 29.02, 20.76;
29.22, 21.31; 29.09, 21.84; 28.94, 22.35; 28.64, 22.94; 28.13, 23.64; 27.49, 24.47;
27.2, 25.45; 26.95, 26.3; 26.91, 27.5];
controller8 = controllerPurePursuit;
controller8.Waypoints = waypoints8;
controller8.LookaheadDistance = 0.5;
controller8.DesiredLinearVelocity = 0.1;
controller8.MaxAngularVelocity = 5.0;

% Robot 9 - Ceja izq
initPose9 = [36.29; 30.51; pi/2];
pose9 = initPose9;
traj9 = pose9;
waypoints9 = [36.29, 30.51; 36.22, 31.06; 35.96, 31.55; 35.6, 31.93; 35.07, 32.29;
34.43, 32.55; 33.62, 32.65; 32.99, 32.64; 32.26, 32.61; 31.52, 32.61; 30.71, 32.55;
29.92, 32.45; 29.34, 32.33; 29, 32; 28.66, 31.69; 28.51, 31.17; 28.51, 30.9; 28.97,
```

```
31.32; 29.61, 31.63; 30.29, 31.77; 31.07, 31.86; 31.69, 31.86; 32.42, 31.82; 33.07,
31.72; 33.65, 31.54; 33.67, 31.63; 34.17, 31.49; 34.78, 31.27; 35.19, 31.12; 35.72,
30.83; 36.29, 30.51];
controller9 = controllerPurePursuit;
controller9.Waypoints = waypoints9;
controller9.LookaheadDistance = 0.5;
controller9.DesiredLinearVelocity = 0.1;
controller9.MaxAngularVelocity = 5.0;

% Robot 10 - Pupila izq
initPose10 = [31.63; 28.93; pi];
pose10 = initPose10;
traj10 = pose10;
waypoints10 = [31.63, 28.93; 31.33, 28.76; 31.31, 28.45; 31.63, 28.27; 32, 28.5;
31.91, 28.83; 31.63, 28.93];
controller10 = controllerPurePursuit;
controller10.Waypoints = waypoints10;
controller10.LookaheadDistance = 0.5;
controller10.DesiredLinearVelocity = 0.7;
controller10.MaxAngularVelocity = 5.0;

% Robot 11
initPose11 = [19.91; 28.55; pi];
pose11 = initPose11;
traj11 = pose11;
waypoints11 = [19.91, 28.55; 19.56, 28.38; 19.57, 28.06; 19.86, 27.94; 20.13,
28.07; 20.18, 28.43; 19.91, 28.55];
controller11 = controllerPurePursuit;
controller11.Waypoints = waypoints11;
controller11.LookaheadDistance = 0.5;
controller11.DesiredLinearVelocity = 0.7;
controller11.MaxAngularVelocity = 5.0;

% Robot 12 - Ceja der
initPose12 = [15.01; 30.16; pi/3];
pose12 = initPose12;
traj12 = pose12;
waypoints12 = [15.01, 30.16; 15.35, 30.74; 15.84, 31.17; 16.65, 31.5; 17.56, 31.82;
18.46, 31.99; 19.13, 31.93; 19.97, 31.84; 20.83, 31.86; 21.6, 31.99; 22.25, 31.75;
22.89, 31.49; 23.5, 31.12; 23.44, 30.57; 22.58, 30.6; 21.78, 30.62; 21.27, 30.75;
20.61, 30.96; 19.81, 31.02; 19.13, 31.08; 18.34, 31.07; 17.73, 30.92; 17.22, 30.65;
16.67, 30.37; 16.12, 30.22; 15.62, 30.19; 15.01, 30.16];
controller12 = controllerPurePursuit;
controller12.Waypoints = waypoints12;
controller12.LookaheadDistance = 0.5;
controller12.DesiredLinearVelocity = 0.1;
controller12.MaxAngularVelocity = 5.0;

% Robot 13
initPose13 = [30.57; 28.97; -2*pi/3];
```

```
pose13 = initPose13;
traj13 = pose13;
waypoints13 = [30.57, 28.97; 29.86, 28.69; 29.05, 27.93; 28.99, 27.65; 29.62,
27.66; 29.96, 27.6; 30.39, 27.58; 30.89, 27.42; 31.56, 27.33; 32.51, 27.4; 33,
27.5; 33.41, 27.88; 33.82, 28.25; 33.84, 28.55; 33.38, 28.84; 32.64, 29.1; 31.84,
29.15; 31.08, 29.08; 30.57, 28.97; 31.43, 29.39; 32.02, 29.44; 32.4, 29.39; 32.81,
29.38; 33.2, 29.33; 33.65, 29.19; 33.92, 29.14; 34, 29; 34.5, 29; 34.02, 28.77;
34.22, 28.68; 34.5, 28.5; 34.26, 28.44; 34, 28.5; 33.84, 28.55];
controller13 = controllerPurePursuit;
controller13.Waypoints = waypoints13;
controller13.LookaheadDistance = 0.5;
controller13.DesiredLinearVelocity = 0.1;
controller13.MaxAngularVelocity = 5.0;

% Robot 14 - Cabello
initPose14 = [21; 11.17; -pi/2];
pose14 = initPose14;
traj14 = pose14;
waypoints14 = [21, 11.17;
  21.652463620229046, 9.268068767775317;
  22.158895155118337, 7.368950511940476;
  22.602022748146464, 5.406528314244473;
  22.34880698070182, 3.697321883993116;
  21.90567938767369, 2.4312430467698882;
  20.460768122425048, 1.5926603654951412;
  20.740096162021235, 0.6687291576000544;
  21.0, 0.0;
  8.943193048936578, 0.04470894488368937;
  9.723187479509859, 1.1755484765280466;
  9.477359826839416, 2.229095559401367;
  9.16129570197742, 3.0368149896042462;
  9.126177465881643, 4.195716780764899;
  9.86366042389297, 5.354618571925551;
  9.502804963070625, 5.9480904087393105;
  8.771573944867418, 6.679321426942516;
  8.11346602848453, 7.3374293433254;
  6.651003992078113, 7.0449369360441185;
  5.48103436295298, 7.739606403337164;
  4.74980334474977, 8.727903737991763;
  4.6190351679617745, 10.229774425927532;
  5.362574365765088, 11.469006422266382;
  6.530993105170293, 12.60201853320476;
  5.185541223430965, 13.735030644143137;
  5.023355425473904, 14.962203803878273;
  4.304943214092533, 15.43193486516609;
  3.709002409656988, 16.660400443404928;
  4.192629663174138, 18.05915053504867;
  5.60850851746416, 19.61069716404159;
  5.487263446753063, 20.580657729730355;
  4.55771790463466, 21.30812815399693;
```

```
3.991907574649545, 23.126804214663366;
4.0, 24.0;
5.042698187479044, 24.743405157477977;
5.835190434482336, 25.16878849390789;
5.3589267091320245, 25.88318408193335;
4.684219764885751, 26.6769569575172;
4.525465189768981, 27.49057415499064;
4.565153833548173, 28.403412961912064;
4.886028326652011, 29.657388678181015;
5.476722228795189, 31.19832929246756;
6.0, 32.0;
6.8122040945102, 32.097211317468044;
7.414388888146557, 32.73503815428702;
8.2871798124927, 33.505147793415965;
8.343753523356167, 34.523474588958365;
8.145745535334031, 35.202359119319965;
8.372040378787899, 35.96610421597677;
8.722881483180181, 36.95413333861964;
9.077219038382148, 38.26245661936538;
9.567840268661804, 40.0068876603597;
10.440055789158972, 41.64229176129188;
11.448554984733821, 42.73256116191333;
12.94713734170242, 44.316710040192596;
14.785037962517537, 45.53334847932372;
16.312307492490664, 47.034732085060014;
18.590268825331936, 48.380800145375304;
20.764686461225878, 49.07972009976978;
22.266070066962172, 49.05383417553295;
23.482708506093307, 48.820860857401456;
24.388715854382447, 49.31269341790127;
25.553582445039915, 49.46800896332227;
27.106737899249815, 49.70098228145375;
28.426920035328376, 49.44212303908542;
31.067284307485203, 47.94073943334914;
33.034614549484594, 46.62055729727068;
34.0, 46.0;
34.76897147335238, 45.8439795701657;
36.115039533667684, 44.62734113103457;
38.154283210050934, 42.68091339056983;
40.34322609920141, 40.20952625765801;
41.578919665657324, 37.91466677709704;
42.285030275060706, 35.302057522304544;
42.0, 34.0;
41.4731282783127, 31.91249175085243;
40.843781181782305, 31.038398561226874;
41.22838218521755, 29.32517590956079;
40.62696754753443, 27.967675764824712;
40.142461699754385, 26.7079605605966;
41.06302281053647, 25.69049828025851;
42.12893567565256, 24.430783076030394;
```

```
42.2602815702696, 23.553558145807813;
42.589916338953714, 22.79733720588543;
42.2602815702696, 21.866603741365576;
42.33660322330601, 20.995215915859987;
42.82008870741974, 19.625340377537725;
43.061831449476614, 18.497207581272335;
43.142412363495566, 16.5229751878079;
43.10212190648609, 15.83803741864677;
42.364713269213844, 15.470239067054255;
42.0, 15.0;
43.426809478056875, 14.568307294477329;
44.588495495039794, 14.369161120137399;
44.422540349756524, 13.273857161267784;
43.69233771051012, 12.64322760919134;
42.76298889692378, 12.51046349296472;
41.80044905428078, 12.77599172541796;
42.265123461073955, 12.211744231454826;
43.061708158433674, 12.04578908617155;
44.0, 12.0;
44.787641669379724, 12.311317318624791;
44.84726911602918, 11.838113622789907;
44.63460870380947, 10.934306870856126;
44.06751427122357, 10.597594551508248;
43.12556995055035, 10.47188989000507;
41.95464544000794, 10.548254531996966;
42.36192353063139, 9.657333708758177;
42.46374305328725, 8.91914216950318;
42.66738209859897, 8.053676226928356;
42.38737841129535, 7.493668852321116;
41.90373567868001, 6.806387074394051;
42.15828448531966, 6.322744341778708;
42.75841871430853, 5.893708935452847;
43.412467478028965, 5.566684553592633;
43.86212600308676, 4.912635789872204;
44.27090648041202, 3.9315626442915614;
44.14827233721444, 2.909611450978392;
43.657735764424125, 2.092050496327856;
42.840174809773586, 1.6423919712700614;
42.06435360875939, 1.273614788866229;
41.689349791160495, 1.1202041362121373;
41.936511398214314, 1.0179303677760763;
42.72061028955745, 0.7537231326495849;
43.40243541246453, 0.4895158975230935;
43.794484858136094, 0.012238311488141397;
26.774464110097195, 0.0409033830000853;
27.0, 1.0;
27.919579794700663, 1.1922321775866445;
28.589961395346172, 1.5342636064874138;
29.0, 2.0;
29.356111796083898, 2.3414577786932287;
```

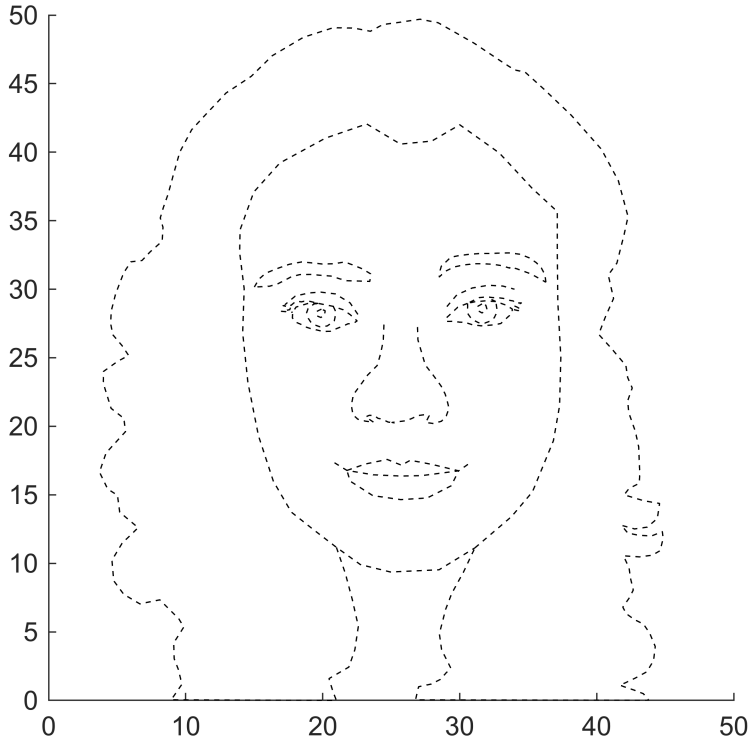
```
  28.6779750130575, 3.6966226253906047;
  28.516195509841648, 4.86503014861622;
  28.965583018774577, 6.6266291836333;
  29.421060185358495, 7.811001146032092;
  30.282719506168917, 9.377654456596497;
  31.11, 11.13;];
controller14 = controllerPurePursuit;
controller14.Waypoints = waypoints14;
controller14.LookaheadDistance = 0.5;
controller14.DesiredLinearVelocity = 1.0;
controller14.MaxAngularVelocity = 5.0;

% Robot 15 - Ojo der
initPose15 = [19.71; 28.96; 0];
pose15 = initPose15;
traj15 = pose15;
waypoints15 = [19.71, 28.96; 20.77, 28.75; 21.5, 28.5; 22, 28.15; 22.51, 27.69;
22.22, 27.49; 21.85, 27.48; 21.5, 27.32; 21.02, 27.13; 20.55, 26.93; 20.01, 26.92;
19.6, 27.02; 19.12, 27.13; 18.68, 27.27; 18.43, 27.43; 18.1, 27.57; 17.79, 27.9;
17.81, 28.32; 18.06, 28.7; 18.55, 28.85; 19.08, 29.01; 19.71, 28.96; 18.84, 29.17;
18.46, 29.08; 18.1, 29.12; 17.88, 28.99; 17.7, 28.83; 17.11, 28.8; 17.66, 28.69;
17.6, 28.51; 16.99, 28.39; 17.81, 28.32];
controller15 = controllerPurePursuit;
controller15.Waypoints = waypoints15;
controller15.LookaheadDistance = 0.5;
controller15.DesiredLinearVelocity = 0.1;
controller15.MaxAngularVelocity = 10.0;
```

Trayectorias de los robots

Cada robot tiene una serie de puntos de referencia (waypoints) que definen su trayectoria. Estos waypoints son puntos en el espacio que el robot debe seguir para completar su movimiento. Cada robots tiene una treayectoria, como la "Sonrisa" o la "Nariz", que están diseñadas para dibujar una figura específica cuando se combinan las trayectorias de todos los robots. Podemos ver la figura que se espera que los robots trazen.

```
figure;
axis equal;
xlim([0 50]); ylim([0 50]);
hold on;
plot(waypoints1(:,1), waypoints1(:,2), 'k--');
plot(waypoints2(:,1), waypoints2(:,2), 'k--');
plot(waypoints3(:,1), waypoints3(:,2), 'k--');
plot(waypoints4(:,1), waypoints4(:,2), 'k--');
plot(waypoints5(:,1), waypoints5(:,2), 'k--');
plot(waypoints6(:,1), waypoints6(:,2), 'k--');
plot(waypoints7(:,1), waypoints7(:,2), 'k--');
plot(waypoints8(:,1), waypoints8(:,2), 'k--');
plot(waypoints9(:,1), waypoints9(:,2), 'k--');
plot(waypoints10(:,1), waypoints10(:,2), 'k--');
plot(waypoints11(:,1), waypoints11(:,2), 'k--');
```

```
plot(waypoints12(:,1), waypoints12(:,2), 'k--');
plot(waypoints13(:,1), waypoints13(:,2), 'k--');
plot(waypoints14(:,1), waypoints14(:,2), 'k--');
plot(waypoints15(:,1), waypoints15(:,2), 'k--');
```



Para los robots que terminan su trayectoria en la misma coordenada que su posición inicialn se definen contadores que se incrementan cada vez que el robot entra dentro del área de tolerancia alrededor de la meta.Definimos la última posición de su trayectoria (waypointsX) como su meta (goalX).

```
goal2 = waypoints2(end, :)';
goal3 = waypoints3(end, :)';
goal9 = waypoints9(end, :)';
goal10 = waypoints10(end, :)';
goal11 = waypoints11(end, :)';
goal12 = waypoints12(end, :)';
goal13 = waypoints13(end, :)';
goal15 = waypoints15(end, :)';
```

Inicializamos el contador goalXCount en 0 para registrar cuántas veces entra al área de la meta

```
goal2Count = 0;
goal3Count = 0;
goal9Count = 0;
goal10Count = 0;
goal11Count = 0;
goal12Count = 0;
goal13Count = 0;
```

```
goal15Count = 0;
```

Inicializamos la bandera inGoalX en false para detectar transiciones de entrada a la meta

```
inGoal2 = false;
inGoal3 = false;
inGoal9 = false;
inGoal10 = false;
inGoal11 = false;
inGoal12 = false;
inGoal13 = false;
inGoal15 = false;
```

Simulación

La simulación es útil para asegurarse de que los robots sigan sus trayectorias correctamente y se ubiquen en el espacio de manera coherente con la figura que deben formar. Este bloque implementa la simulación en tiempo discreto de 15 robots móviles diferenciales. Cada robot tiene un conjunto de puntos de referencia (`waypoints`) a seguir, un controlador propio, y su propia lógica de finalización. La simulación se ejecuta hasta que todos los robots hayan alcanzado sus respectivas metas.

```
while ~finished1 || ~finished2 || ~finished3 || ~finished4 || ~finished5 ||
~finished6 || ~finished7 || ~finished8 || ~finished9 || ~finished10 || ~finished11
|| ~finished12 || ~finished13 || ~finished14 || ~finished15
    t = t + sampleTime;

    % === Robot 1 ===
    if ~finished1
        goal1 = waypoints1(end, :)';
        [vRef1, wRef1] = controller1(pose1);
        [wL1, wR1] = inverseKinematics(dd, vRef1, wRef1);
        [v1, w1] = forwardKinematics(dd, wL1, wR1);
        velB1 = [v1; 0; w1];
        vel1 = bodyToWorld(velB1, pose1);
        pose1 = pose1 + vel1 * sampleTime;
        traj1(:, end+1) = pose1;

        % En cada robot, reemplaza esta parte:
        if norm(pose1(1:2) - goal1) < goalTolerance
            finished1 = true;
        end
    end

    % === Robot 2 ===
    if ~finished2
        [vRef2, wRef2] = controller2(pose2);
        [wL2, wR2] = inverseKinematics(dd, vRef2, wRef2);
        [v2, w2] = forwardKinematics(dd, wL2, wR2);
        velB2 = [v2; 0; w2];
        vel2 = bodyToWorld(velB2, pose2);
```

```
        pose2 = pose2 + vel2 * sampleTime;
        traj2(:, end+1) = pose2;

        dist2 = norm(pose2(1:2) - goal2);
        if dist2 < goalTolerance
            if ~inGoal2
                goal2Count = goal2Count + 1;
                inGoal2 = true;
            end
        else
            inGoal2 = false;
        end
        if goal2Count >= 2
            finished2 = true;
        end
    end

    % === Robot 3 ===
    if ~finished3
        [vRef3, wRef3] = controller3(pose3);
        [wL3, wR3] = inverseKinematics(dd, vRef3, wRef3);
        [v3, w3] = forwardKinematics(dd, wL3, wR3);
        velB3 = [v3; 0; w3];
        vel3 = bodyToWorld(velB3, pose3);
        pose3 = pose3 + vel3 * sampleTime;
        traj3(:, end+1) = pose3;

        dist3 = norm(pose3(1:2) - goal3);
        if dist3 < goalTolerance
            if ~inGoal3
                goal3Count = goal3Count + 1;
                inGoal3 = true;
            end
        else
            inGoal3 = false;
        end
        if goal3Count >= 2
            finished3 = true;
        end
    end

    % === Robot 4 ===
    if ~finished4
        goal4 = waypoints4(end, :)';
        [vRef4, wRef4] = controller4(pose4);
        [wL4, wR4] = inverseKinematics(dd, vRef4, wRef4);
        [v4, w4] = forwardKinematics(dd, wL4, wR4);
        velB4 = [v4; 0; w4];
        vel4 = bodyToWorld(velB4, pose4);
        pose4 = pose4 + vel4 * sampleTime;
```

```
        traj4(:, end+1) = pose4;

        % En cada robot, reemplaza esta parte:
        if norm(pose4(1:2) - goal4) < goalTolerance
            finished4 = true;
        end
    end

    % === Robot 5 ===
    if ~finished5
        goal5 = waypoints5(end, :)';
        [vRef5, wRef5] = controller5(pose5);
        [wL5, wR5] = inverseKinematics(dd, vRef5, wRef5);
        [v5, w5] = forwardKinematics(dd, wL5, wR5);
        velB5 = [v5; 0; w5];
        vel5 = bodyToWorld(velB5, pose5);
        pose5 = pose5 + vel5 * sampleTime;
        traj5(:, end+1) = pose5;

        % En cada robot, reemplaza esta parte:
        if norm(pose5(1:2) - goal5) < goalTolerance
            finished5 = true;
        end
    end

    % === Robot 6 ===
    if ~finished6
        goal6 = waypoints6(end, :)';
        [vRef6, wRef6] = controller6(pose6);
        [wL6, wR6] = inverseKinematics(dd, vRef6, wRef6);
        [v6, w6] = forwardKinematics(dd, wL6, wR6);
        velB6 = [v6; 0; w6];
        vel6 = bodyToWorld(velB6, pose6);
        pose6 = pose6 + vel6 * sampleTime;
        traj6(:, end+1) = pose6;

        % En cada robot, reemplaza esta parte:
        if norm(pose6(1:2) - goal6) < goalTolerance
            finished6 = true;
        end
    end

    % === Robot 7 ===
    if ~finished7
        goal7 = waypoints7(end, :)';
        [vRef7, wRef7] = controller7(pose7);
        [wL7, wR7] = inverseKinematics(dd, vRef7, wRef7);
        [v7, w7] = forwardKinematics(dd, wL7, wR7);
        velB7 = [v7; 0; w7];
        vel7 = bodyToWorld(velB7, pose7);
```

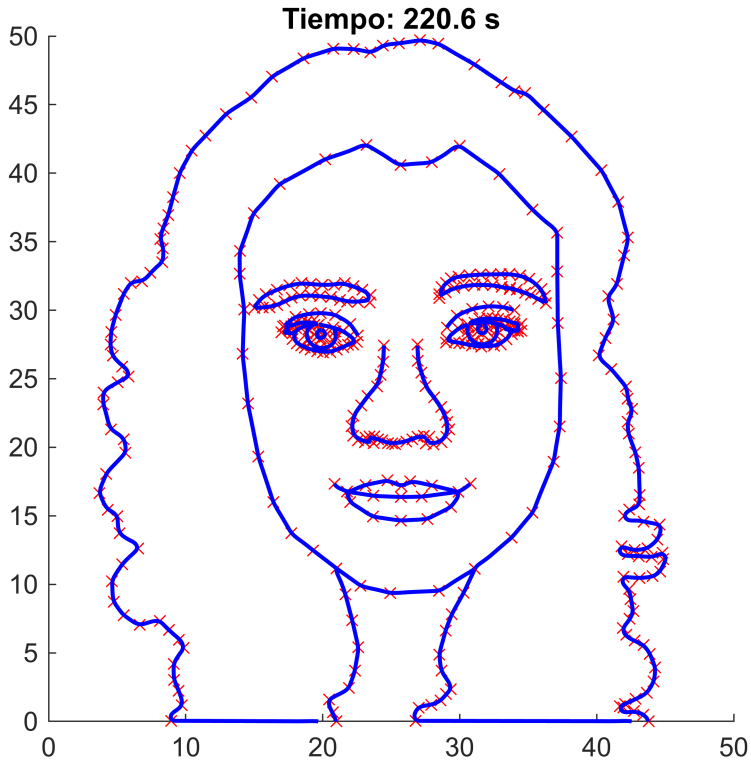
```
        pose7 = pose7 + vel7 * sampleTime;
        traj7(:, end+1) = pose7;

        % En cada robot, reemplaza esta parte:
        if norm(pose7(1:2) - goal7) < goalTolerance
            finished7 = true;
        end
    end

        % === Robot 8 ===
    if ~finished8
        goal8 = waypoints8(end, :)';
        [vRef8, wRef8] = controller8(pose8);
        [wL8, wR8] = inverseKinematics(dd, vRef8, wRef8);
        [v8, w8] = forwardKinematics(dd, wL8, wR8);
        velB8 = [v8; 0; w8];
        vel8 = bodyToWorld(velB8, pose8);
        pose8 = pose8 + vel8 * sampleTime;
        traj8(:, end+1) = pose8;

        % En cada robot, reemplaza esta parte:
        if norm(pose8(1:2) - goal8) < goalTolerance
            finished8 = true;
        end
    end

    % === Robot 9 ===
    if ~finished9
        [vRef9, wRef9] = controller9(pose9);
        [wL9, wR9] = inverseKinematics(dd, vRef9, wRef9);
        [v9, w9] = forwardKinematics(dd, wL9, wR9);
        velB9 = [v9; 0; w9];
        vel9 = bodyToWorld(velB9, pose9);
        pose9 = pose9 + vel9 * sampleTime;
        traj9(:, end+1) = pose9;

        dist9 = norm(pose9(1:2) - goal9);
        if dist9 < goalTolerance
            if ~inGoal9
                goal9Count = goal9Count + 1;
                inGoal9 = true;
            end
        else
            inGoal9 = false;
        end
        if goal9Count >= 2
            finished9 = true;
        end
    end
```

```
    % === Robot 10 ===
    if ~finished10
        [vRef10, wRef10] = controller10(pose10);
        [wL10, wR10] = inverseKinematics(dd, vRef10, wRef10);
        [v10, w10] = forwardKinematics(dd, wL10, wR10);
        velB10 = [v10; 0; w10];
        vel10 = bodyToWorld(velB10, pose10);
        pose10 = pose10 + vel10 * sampleTime;
        traj10(:, end+1) = pose10;

        dist10 = norm(pose10(1:2) - goal10);
        if dist10 < goalTolerance
            if ~inGoal10
                goal10Count = goal10Count + 1;
                inGoal10 = true;
            end
        else
            inGoal10 = false;
        end
        if goal10Count >= 2
            finished10 = true;
        end
    end

    % === Robot 11 ===
    if ~finished11
        [vRef11, wRef11] = controller11(pose11);
        [wL11, wR11] = inverseKinematics(dd, vRef11, wRef11);
        [v11, w11] = forwardKinematics(dd, wL11, wR11);
        velB11 = [v11; 0; w11];
        vel11 = bodyToWorld(velB11, pose11);
        pose11 = pose11 + vel11 * sampleTime;
        traj11(:, end+1) = pose11;

        dist11 = norm(pose11(1:2) - goal11);
        if dist11 < goalTolerance
            if ~inGoal11
                goal11Count = goal11Count + 1;
                inGoal11 = true;
            end
        else
            inGoal11 = false;
        end
        if goal11Count >= 2
            finished11 = true;
        end
    end

    % === Robot 12 ===
    if ~finished12
```

```
        [vRef12, wRef12] = controller12(pose12);
        [wL12, wR12] = inverseKinematics(dd, vRef12, wRef12);
        [v12, w12] = forwardKinematics(dd, wL12, wR12);
        velB12 = [v12; 0; w12];
        vel12 = bodyToWorld(velB12, pose12);
        pose12 = pose12 + vel12 * sampleTime;
        traj12(:, end+1) = pose12;

        dist12 = norm(pose12(1:2) - goal12);
        if dist12 < goalTolerance
            if ~inGoal12
                goal12Count = goal12Count + 1;
                inGoal12 = true;
            end
        else
            inGoal12 = false;
        end
        if goal12Count >= 2
            finished12 = true;
        end
    end

    % === Robot 13 ===
    if ~finished13
        [vRef13, wRef13] = controller13(pose13);
        [wL13, wR13] = inverseKinematics(dd, vRef13, wRef13);
        [v13, w13] = forwardKinematics(dd, wL13, wR13);
        velB13 = [v13; 0; w13];
        vel13 = bodyToWorld(velB13, pose13);
        pose13 = pose13 + vel13 * sampleTime;
        traj13(:, end+1) = pose13;

        dist13 = norm(pose13(1:2) - goal13);
        if dist13 < goalTolerance
            if ~inGoal13
                goal13Count = goal13Count + 1;
                inGoal13 = true;
            end
        else
            inGoal13 = false;
        end
        if goal13Count >= 3
            finished13 = true;
        end
    end

    % === Robot 14 ===
    if ~finished14
        goal14 = waypoints14(end, :)';
        [vRef14, wRef14] = controller14(pose14);
```

```
        [wL14, wR14] = inverseKinematics(dd, vRef14, wRef14);
        [v14, w14] = forwardKinematics(dd, wL14, wR14);
        velB14 = [v14; 0; w14];
        vel14 = bodyToWorld(velB14, pose14);
        pose14 = pose14 + vel14 * sampleTime;
        traj14(:, end+1) = pose14;

        % En cada robot, reemplaza esta parte:
        if norm(pose14(1:2) - goal14) < goalTolerance
            finished14 = true;
        end
    end

    % === Robot 15 ===
    if ~finished15
        [vRef15, wRef15] = controller15(pose15);
        [wL15, wR15] = inverseKinematics(dd, vRef15, wRef15);
        [v15, w15] = forwardKinematics(dd, wL15, wR15);
        velB15 = [v15; 0; w15];
        vel15 = bodyToWorld(velB15, pose15);
        pose15 = pose15 + vel15 * sampleTime;
        traj15(:, end+1) = pose15;

        dist15 = norm(pose15(1:2) - goal15);
        if dist15 < goalTolerance
            if ~inGoal15
                goal15Count = goal15Count + 1;
                inGoal15 = true;
            end
        else
            inGoal15 = false;
        end
        if goal15Count >= 2
            finished15 = true;
        end
    end

    % Visualización
    cla;
    plot(waypoints1(:,1), waypoints1(:,2), 'rx');
    plot(waypoints2(:,1), waypoints2(:,2), 'rx');
    plot(waypoints3(:,1), waypoints3(:,2), 'rx');
    plot(waypoints4(:,1), waypoints4(:,2), 'rx');
    plot(waypoints5(:,1), waypoints5(:,2), 'rx');
    plot(waypoints6(:,1), waypoints6(:,2), 'rx');
    plot(waypoints7(:,1), waypoints7(:,2), 'rx');
    plot(waypoints8(:,1), waypoints8(:,2), 'rx');
    plot(waypoints9(:,1), waypoints9(:,2), 'rx');
    plot(waypoints10(:,1), waypoints10(:,2), 'rx');
    plot(waypoints11(:,1), waypoints11(:,2), 'rx');
```

```
    plot(waypoints12(:,1), waypoints12(:,2), 'rx');
    plot(waypoints13(:,1), waypoints13(:,2), 'rx');
    plot(waypoints14(:,1), waypoints14(:,2), 'rx');
    plot(waypoints15(:,1), waypoints15(:,2), 'rx');
    plot(traj1(1,:), traj1(2,:), 'b', 'LineWidth', 1.5);
    plot(traj2(1,:), traj2(2,:), 'b', 'LineWidth', 1.5);
    plot(traj3(1,:), traj3(2,:), 'b', 'LineWidth', 1.5);
    plot(traj4(1,:), traj4(2,:), 'b', 'LineWidth', 1.5);
    plot(traj5(1,:), traj5(2,:), 'b', 'LineWidth', 1.5);
    plot(traj6(1,:), traj6(2,:), 'b', 'LineWidth', 1.5);
    plot(traj7(1,:), traj7(2,:), 'b', 'LineWidth', 1.5);
    plot(traj8(1,:), traj8(2,:), 'b', 'LineWidth', 1.5);
    plot(traj9(1,:), traj9(2,:), 'b', 'LineWidth', 1.5);
    plot(traj10(1,:), traj10(2,:), 'b', 'LineWidth', 1.5);
    plot(traj11(1,:), traj11(2,:), 'b', 'LineWidth', 1.5);
    plot(traj12(1,:), traj12(2,:), 'b', 'LineWidth', 1.5);
    plot(traj13(1,:), traj13(2,:), 'b', 'LineWidth', 1.5);
    plot(traj14(1,:), traj14(2,:), 'b', 'LineWidth', 1.5);
    plot(traj15(1,:), traj15(2,:), 'b', 'LineWidth', 1.5);
    if ~finished1, drawRobot(pose1, 'b'); end
    if ~finished2, drawRobot(pose2, 'b'); end
    if ~finished3, drawRobot(pose3, 'b'); end
    if ~finished4, drawRobot(pose4, 'b'); end
    if ~finished5, drawRobot(pose5, 'b'); end
    if ~finished6, drawRobot(pose6, 'b'); end
    if ~finished7, drawRobot(pose7, 'b'); end
    if ~finished8, drawRobot(pose8, 'b'); end
    if ~finished9, drawRobot(pose9, 'b'); end
    if ~finished10, drawRobot(pose10, 'b'); end
    if ~finished11, drawRobot(pose11, 'b'); end
    if ~finished12, drawRobot(pose12, 'b'); end
    if ~finished13, drawRobot(pose13, 'b'); end
    if ~finished14, drawRobot(pose14, 'b'); end
    if ~finished15, drawRobot(pose15, 'b'); end
    title(sprintf('Tiempo: %.1f s', t));
    drawnow;
    waitfor(r);
end
```

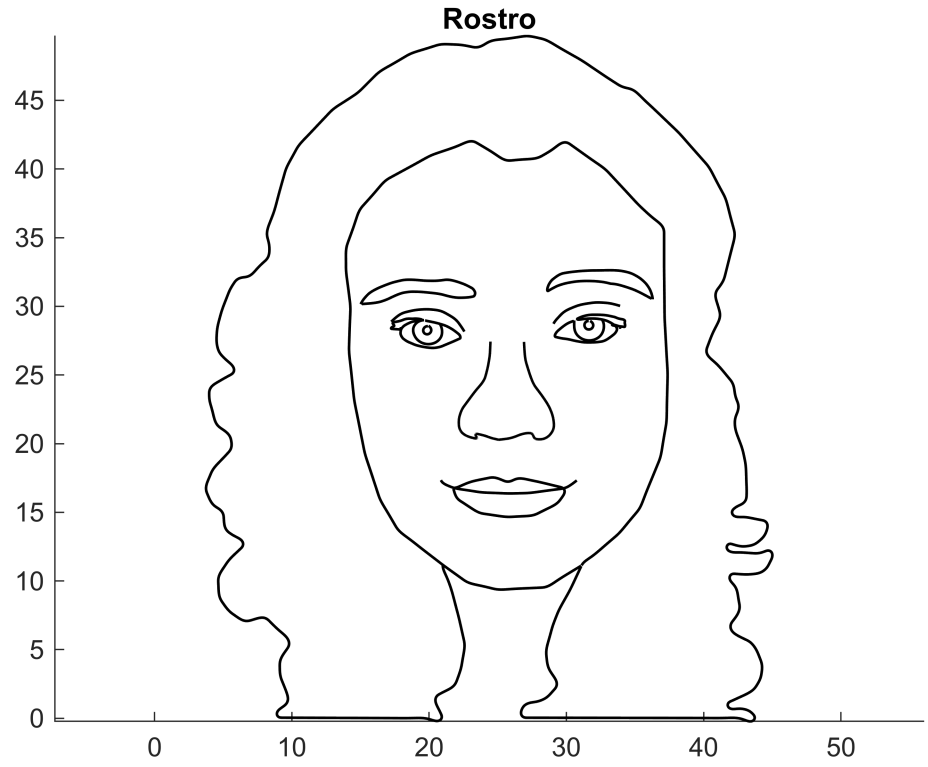


Visualización

Al final del código, se genera una figura que visualiza las trayectorias de los robots mostrando así la imagen final del rostro.

```
figure;
hold on;
axis equal;
title('Rostro');

% Dibujar las trayectorias
plot(traj1(1,:), traj1(2,:), 'k', 'LineWidth', 1);
plot(traj2(1,:), traj2(2,:), 'k', 'LineWidth', 1);
plot(traj3(1,:), traj3(2,:), 'k', 'LineWidth', 1);
plot(traj4(1,:), traj4(2,:), 'k', 'LineWidth', 1);
plot(traj5(1,:), traj5(2,:), 'k', 'LineWidth', 1);
plot(traj6(1,:), traj6(2,:), 'k', 'LineWidth', 1);
plot(traj7(1,:), traj7(2,:), 'k', 'LineWidth', 1);
plot(traj8(1,:), traj8(2,:), 'k', 'LineWidth', 1);
plot(traj9(1,:), traj9(2,:), 'k', 'LineWidth', 1);
plot(traj10(1,:), traj10(2,:), 'k', 'LineWidth', 1);
plot(traj11(1,:), traj11(2,:), 'k', 'LineWidth', 1);
plot(traj12(1,:), traj12(2,:), 'k', 'LineWidth', 1);
plot(traj13(1,:), traj13(2,:), 'k', 'LineWidth', 1);
plot(traj14(1,:), traj14(2,:), 'k', 'LineWidth', 1);
```

```
plot(traj15(1,:), traj15(2,:), 'k', 'LineWidth', 1);
```




```
function drawRobot(pose, color)
    x = pose(1); y = pose(2); theta = pose(3);
    radius = 0.15; % Radio del robot

    % Coordenadas del círculo
    theta_circ = linspace(0, 2*pi, 100);
    circleX = radius * cos(theta_circ) + x;
    circleY = radius * sin(theta_circ) + y;

    % Dibujar el círculo
    fill(circleX, circleY, color, 'FaceAlpha', 0, 'EdgeColor', 'b', 'LineWidth', 2);

    % Punto inicial de la flecha (en el borde del círculo, hacia adelante)
    x0 = x + radius * cos(theta);
    y0 = y + radius * sin(theta);

    % Dibujar la dirección del robot con una flecha
    quiver(x0, y0, cos(theta)*0.25, sin(theta)*0.5, 0, 'r', 'LineWidth', 2);
end
```