Instituto Tecnológico y de Estudios Superiores de Monterrey

TE3002B.502

Implementación de robótica Inteligente (Gpo 502)

Semestre: febrero - junio 2023

Actividad 6: (SLAM de LiDAR)

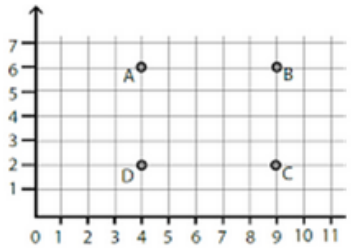
Alumno:

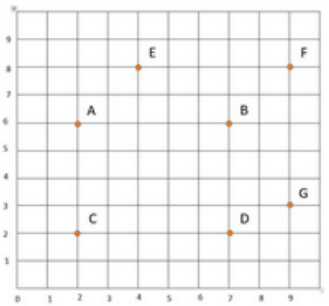
Fredy Yahir Canseco Santos A01735589

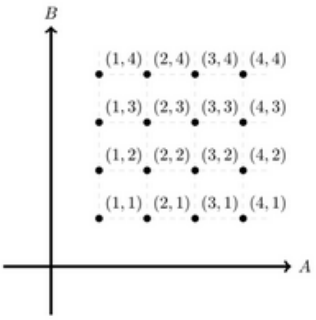
Profesor: Dr. Alfredo García Suárez

Fecha de entrega: 06 de Mayo del 2023

1. Implementar el código requerido para generar el seguimiento de los siguientes waypoints de forma aleatoria, ajustando los parámetros: sampleTime, tVec, initPose, lidar.scanAngles, lidar.maxRange, waypoints, controller.LookaheadDistance, controller.DesiredLinearVelocity y controller.MaxAngularVelocity. Evadiendo los obstáculos del mapa de navegación “exampleMap”







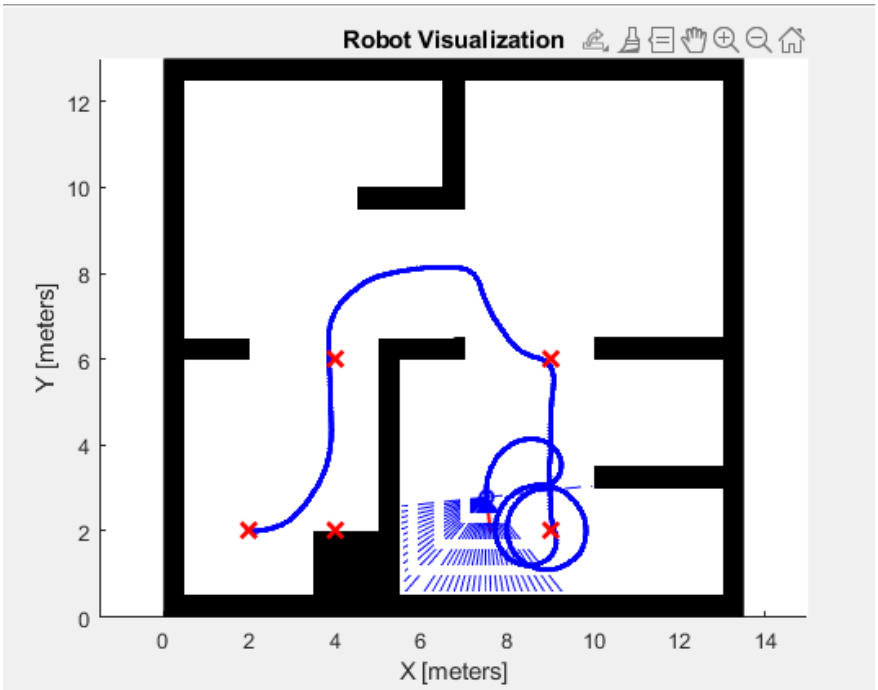
Trayectoria 1:

```
initPose = [2;2;0];            % Initial pose (x y theta)

% Create lidar sensor
lidar.scanAngles = linspace(-pi/2,pi/2,51);
lidar.maxRange = 3;

% Create waypoints
waypoints = [initPose(1:2)';4 2; 4 6; 9 6; 9 2];

% Pure Pursuit Controller
controller.LookaheadDistance = 0.5;
controller.DesiredLinearVelocity = 0.75;
controller.MaxAngularVelocity = 1.5;
```



Código Desarrollado

```
%% Simulation setup
% Define Vehicle
R = 0.1;                          % Wheel radius [m]
L = 0.5;                          % Wheelbase [m]
dd = DifferentialDrive(R,L);

% Sample time and time array
sampleTime = 0.1;              % Sample time [s]
tVec = 0:sampleTime:45;        % Time array

% Initial conditions
initPose = [2;2;0];            % Initial pose (x y theta)
pose = zeros(3,numel(tVec));   % Pose matrix
pose(:,1) = initPose;

% Load map
close all
load exampleMap

% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(-pi/2,pi/2,51);
lidar.maxRange = 3;

% Create visualizer
viz = Visualizer2D;
viz.hasWaypoints = true;
viz.mapName = 'map';
attachLidarSensor(viz,lidar);
```

```
%% Path planning and following

% Create waypoints
waypoints = [initPose(1:2)';
             4 2;
             4 6;
             9 6;
             9 2];

% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.5;
controller.DesiredLinearVelocity = 0.75;
controller.MaxAngularVelocity = 1.5;
```

```
% Vector Field Histogram (VFH) for obstacle avoidance
vfh = controllerVFH;
vfh.DistanceLimits = [0.05 3];
vfh.NumAngularSectors = 36;
vfh.HistogramThresholds = [5 10];
vfh.RobotRadius = L;
vfh.SafetyDistance = L;
vfh.MinTurningRadius = 0.25;
```

```
%% Simulation loop
r = rateControl(1/sampleTime);
for idx = 2:numel(tVec)

    % Get the sensor readings
    curPose = pose(:,idx-1);
    ranges = lidar(curPose);
    % Run the path following and obstacle avoidance algorithms
    [vRef,wRef,lookAheadPt] = controller(curPose);
    targetDir = atan2(lookAheadPt(2)-curPose(2),lookAheadPt(1)-curPose(1)) - curPose(3);
    steerDir = vfh(ranges,lidar.scanAngles,targetDir);
    if ~isnan(steerDir) && abs(steerDir-targetDir) > 0.1
        wRef = 0.5*steerDir;
    end

    % Control the robot
    velB = [vRef;0;wRef];                    % Body velocities [vx;vy;w]
    vel = bodyToWorld(velB,curPose);  % Convert from body to world

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

    % Update visualization
    viz(pose(:,idx),waypoints,ranges)
    waitfor(r);
end
```

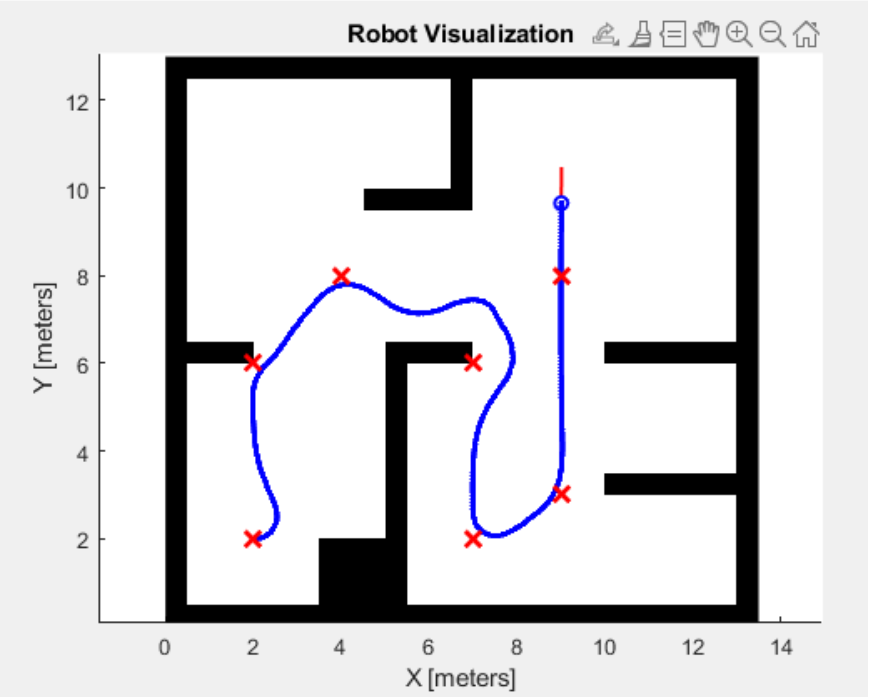
Trayectoria 2:

```
% Initial conditions
initPose = [2;2;0];          % Initial pose (x y theta)

lidar.scanAngles = linspace(-pi/2,pi/2,51);
lidar.maxRange = 1;

% Create waypoints
waypoints = [initPose(1:2)'; 2 2; 2 6; 4 8; 7 6; 7 2; 9 3; 9 8];

% Pure Pursuit Controller
controller.LookaheadDistance = 0.9;
controller.DesiredLinearVelocity = 0.75;
controller.MaxAngularVelocity = 1.5;
```



Código Desarrollado

```
%% Simulation setup
% Define Vehicle
R = 0.1;                        % Wheel radius [m]
L = 0.5;                        % Wheelbase [m]
dd = DifferentialDrive(R,L);

% Sample time and time array
sampleTime = 0.1;              % Sample time [s]
tVec = 0:sampleTime:45;        % Time array

% Initial conditions
initPose = [2;2;0];            % Initial pose (x y theta)
pose = zeros(3,numel(tVec));   % Pose matrix
pose(:,1) = initPose;

% Load map
close all
load exampleMap
```

```
% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(-pi/2,pi/2,51);
lidar.maxRange = 1;

% Create visualizer
viz = Visualizer2D;
viz.hasWaypoints = true;
viz.mapName = 'map';
attachLidarSensor(viz,lidar);
```

```
% Create waypoints
waypoints = [initPose(1:2)';
             2 2;
             2 6;
             4 8;
             7 6;
             7 2;
             9 3;
             9 8];

% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.9;
controller.DesiredLinearVelocity = 0.75;
controller.MaxAngularVelocity = 1.5;


% Vector Field Histogram (VFH) for obstacle avoidance
vfh = controllerVFH;
vfh.DistanceLimits = [0.05 3];
vfh.NumAngularSectors = 36;
vfh.HistogramThresholds = [5 10];
vfh.RobotRadius = L;
vfh.SafetyDistance = L;
vfh.MinTurningRadius = 0.25;
```

```
%% Simulation loop
r = rateControl(1/sampleTime);
for idx = 2:numel(tVec)

    % Get the sensor readings
    curPose = pose(:,idx-1);
    ranges = lidar(curPose);
    % Run the path following and obstacle avoidance algorithms
    [vRef,wRef,lookAheadPt] = controller(curPose);
    targetDir = atan2(lookAheadPt(2)-curPose(2),lookAheadPt(1)-curPose(1)) - curPose(3);
    steerDir = vfh(ranges,lidar.scanAngles,targetDir);
    if ~isnan(steerDir) && abs(steerDir-targetDir) > 0.1
        wRef = 0.5*steerDir;
    end

    % Control the robot
    velB = [vRef;0;wRef];                    % Body velocities [vx;vy;w]
    vel = bodyToWorld(velB,curPose);  % Convert from body to world

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

    % Update visualization
    viz(pose(:,idx),waypoints,ranges)
    waitfor(r);
end
```

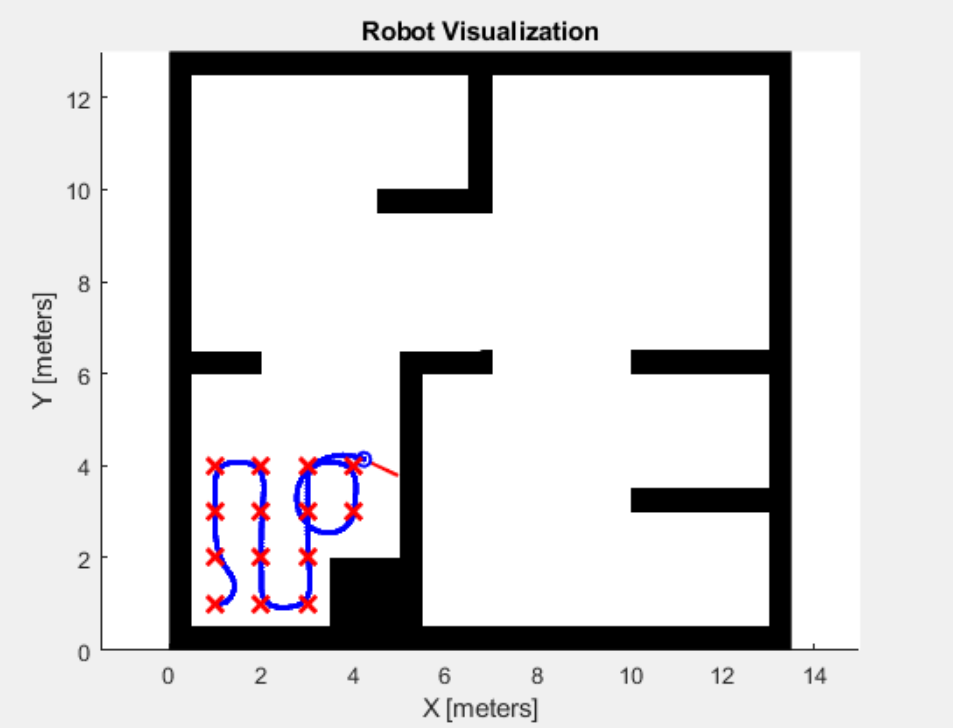
Trayectoria 3:

```
% Initial conditions
initPose = [1;1;0];            % Initial pose (x y theta)

lidar.scanAngles = linspace(-pi/2,pi/2,51);
lidar.maxRange = 0.3;

% Create waypoints
waypoints = [initPose(1:2)'; 1 1; 1 2; 1 3; 1 4; 2 4; 2 3; 2 2; 2 1; 3 1; 3 2; 3 3; 3 4; 4 4; 4 3];

% Pure Pursuit Controller
controller.LookaheadDistance = 0.5;
controller.DesiredLinearVelocity = 0.75;
controller.MaxAngularVelocity = 2;
```



Código Desarrollado

```
%% Simulation setup
% Define Vehicle
R = 0.1;                          % Wheel radius [m]
L = 0.5;                          % Wheelbase [m]
dd = DifferentialDrive(R,L);

% Sample time and time array
sampleTime = 0.1;              % Sample time [s]
tVec = 0:sampleTime:45;        % Time array

% Initial conditions
initPose = [1;1;0];            % Initial pose (x y theta)
pose = zeros(3,numel(tVec));   % Pose matrix
pose(:,1) = initPose;

% Load map
close all
load exampleMap
```

```
% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(-pi/2,pi/2,51);
lidar.maxRange = 0.3;

% Create visualizer
viz = Visualizer2D;
viz.hasWaypoints = true;
viz.mapName = 'map';
attachLidarSensor(viz,lidar);
```

```
%% Path planning and following


% Create waypoints
waypoints = [initPose(1:2)';
             1 1;
             1 2;
             1 3;
             1 4;
             2 4;
             2 3;
             2 2;
             2 1;
             3 1;
             3 2;
             3 3;
             3 4;
             4 4;
             4 3];


% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.5;
controller.DesiredLinearVelocity = 0.75;
controller.MaxAngularVelocity = 2;
```

```
% Vector Field Histogram (VFH) for obstacle avoidance
vfh = controllerVFH;
vfh.DistanceLimits = [0.05 3];
vfh.NumAngularSectors = 36;
vfh.HistogramThresholds = [5 10];
vfh.RobotRadius = L;
vfh.SafetyDistance = L;
vfh.MinTurningRadius = 0.25;
```

```
%% Simulation loop
r = rateControl(1/sampleTime);
for idx = 2:numel(tVec)

    % Get the sensor readings
    curPose = pose(:,idx-1);
    ranges = lidar(curPose);
    % Run the path following and obstacle avoidance algorithms
    [vRef,wRef,lookAheadPt] = controller(curPose);
    targetDir = atan2(lookAheadPt(2)-curPose(2),lookAheadPt(1)-curPose(1)) - curPose(3);
    steerDir = vfh(ranges,lidar.scanAngles,targetDir);
    if ~isnan(steerDir) && abs(steerDir-targetDir) > 0.1
        wRef = 0.5*steerDir;
    end

    % Control the robot
    velB = [vRef;0;wRef];                    % Body velocities [vx;vy;w]
    vel = bodyToWorld(velB,curPose);  % Convert from body to world

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

    % Update visualization
    viz(pose(:,idx),waypoints,ranges)
    waitfor(r);
end
```

2. Implementar el código requerido para generar el seguimiento de los siguientes waypoints de forma secuencial: (1, 2), (2, 10), (11, 8), (8, 2), (8, 8) y (1, 2) ajustando los parámetros: sampleTime, tVec, initPose, scanAngles, lidar.maxRange, waypoints,controller.LookaheadDistance, controller.DesiredLinearVelocity y controller.MaxAngularVelocity. Evadiendo los obstáculos del mapa de navegación “exampleMap”.

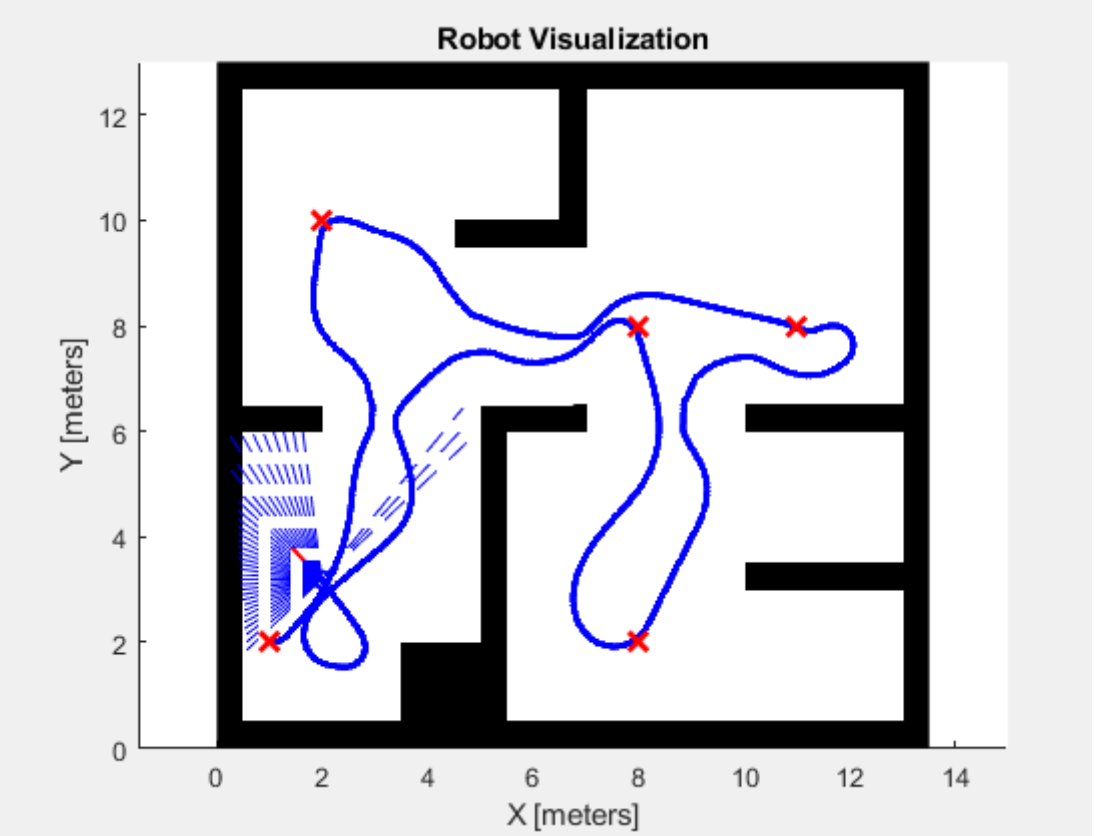
Trayectoria:

```
% Initial conditions
initPose = [1;2;0];          % Initial pose (x y theta)

lidar.scanAngles = linspace(-pi/2,pi/2,51);
lidar.maxRange = 5;

% Create waypoints
waypoints = [initPose(1:2)'; 1 2; 2 10; 11 8; 8 2; 8 8; 1 2];
```

```
% Pure Pursuit Controller
controller.LookaheadDistance = 0.5;
controller.DesiredLinearVelocity = 0.75;
controller.MaxAngularVelocity = 2;
```



Código Desarrollado

```
%% Simulation setup
% Define Vehicle
R = 0.1;                        % Wheel radius [m]
L = 0.5;                        % Wheelbase [m]
dd = DifferentialDrive(R,L);

% Sample time and time array
sampleTime = 0.1;              % Sample time [s]
tVec = 0:sampleTime:80;        % Time array

% Initial conditions
initPose = [1;2;0];            % Initial pose (x y theta)
pose = zeros(3,numel(tVec));   % Pose matrix
pose(:,1) = initPose;

% Load map
close all
load exampleMap
```

```
% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(-pi/2,pi/2,51);
lidar.maxRange = 5;

% Create visualizer
viz = Visualizer2D;
viz.hasWaypoints = true;
viz.mapName = 'map';
attachLidarSensor(viz,lidar);
```

```
%% Path planning and following

% Create waypoints
waypoints = [initPose(1:2)';
             1 2;
             2 10;
             11 8;
             8 2;
             8 8;
             1 2];

% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.5;
controller.DesiredLinearVelocity = 0.75;
controller.MaxAngularVelocity = 2;
```

```
% Vector Field Histogram (VFH) for obstacle avoidance
vfh = controllerVFH;
vfh.DistanceLimits = [0.05 3];
vfh.NumAngularSectors = 36;
vfh.HistogramThresholds = [5 10];
vfh.RobotRadius = L;
vfh.SafetyDistance = L;
vfh.MinTurningRadius = 0.25;
```

```
%% Simulation loop
r = rateControl(1/sampleTime);
for idx = 2:numel(tVec)

    % Get the sensor readings
    curPose = pose(:,idx-1);
    ranges = lidar(curPose);
```

```
    % Run the path following and obstacle avoidance algorithms
    [vRef,wRef,lookAheadPt] = controller(curPose);
    targetDir = atan2(lookAheadPt(2)-curPose(2),lookAheadPt(1)-curPose(1)) - curPose(3);
    steerDir = vfh(ranges,lidar.scanAngles,targetDir);
    if ~isnan(steerDir) && abs(steerDir-targetDir) > 0.1
        wRef = 0.5*steerDir;
    end

    % Control the robot
    velB = [vRef;0;wRef];                    % Body velocities [vx;vy;w]
    vel = bodyToWorld(velB,curPose);  % Convert from body to world

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

    % Update visualization
    viz(pose(:,idx),waypoints,ranges)
    waitfor(r);
end
```