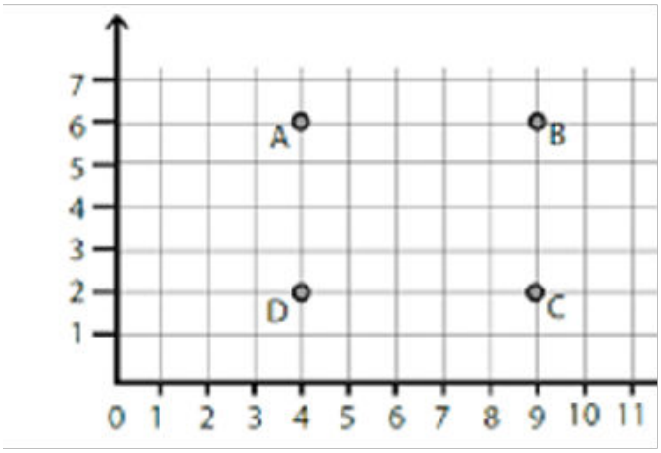
Reporte de mejora de desempeño

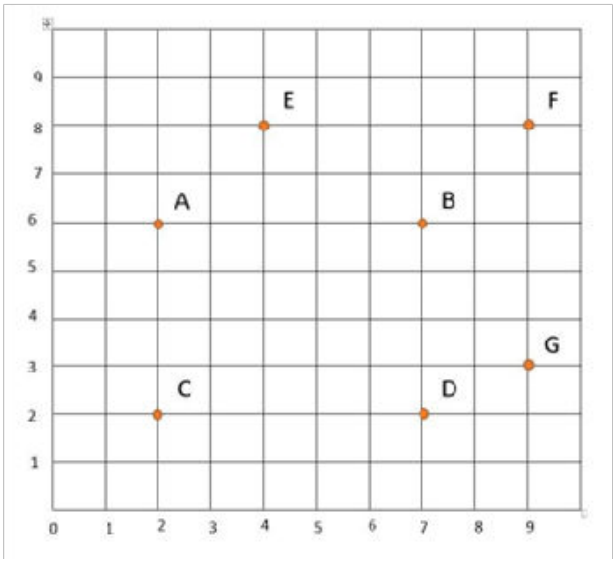
Autor: Jhonatan Yael Martínez Vargas

Instrucciones:

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”.

Ademas implementa 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 previos.





Tomando como referencia la imagen de arriba, se puede apreciar que existen 3 escenarios distintos para realizar las pruebas, sin embargo la metodologia a implementar para dar solucion a cada uno de las diversas rutas es la siguiente:

Establecer las propiedades fisicas del robot

```
% Define Vehicle
R = 0.05;                       % Wheel radius [m]
L = 0.18;                       % Wheelbase [m]
dd = DifferentialDrive(R,L);
```

Definir parametros de simulación dependiendo del mapa.

- **Caso 01**

```
% Sample time and time array
sampleTime = 0.1;              % Sample time [s]
tVec = 0:sampleTime:33;        % Time array

% Initial conditions
initPose = [4; 2; pi/2];           % Initial pose (x y theta)
pose = zeros(3,numel(tVec));   % Pose matrix
pose(:,1) = initPose;
```

- **Caso 02**

```
% Sample time and time array
sampleTime = 0.1;              % Sample time [s]
tVec = 0:sampleTime:47;        % Time array

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

- **Caso 03**

```
% Sample time and time array
sampleTime = 0.1;              % Sample time [s]
tVec = 0:sampleTime:41;        % Time array

% Initial conditions
initPose = [4; 2; 0];            % Initial pose (x y theta)
pose = zeros(3,numel(tVec));   % Pose matrix
pose(:,1) = initPose;
```

- **Caso 04**

```
% Sample time and time array
sampleTime = 0.1;              % Sample time [s]
tVec = 0:sampleTime:96;        % Time array

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

```
pose = zeros(3,numel(tVec));   % Pose matrix
pose(:,1) = initPose;
```

Con esto lo que se puede observar es que lo que cambia para cada uno de los mapas son las posiciones inicales y el tiempo de simulación ***"initPose"*** y ***"sampletTime"***

Definir la configuracion del lidar

- **Caso 01**

```
% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(-pi/2, pi/2, 360);
lidar.maxRange = 2.5;
```

- **Caso 02**

```
% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(-pi/2, pi/2, 360);
lidar.maxRange = 1.5;
```

- **Caso 03**

```
% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(-pi/2, pi/2, 360);
lidar.maxRange = 0.75;
```

- **Caso 04**

```
% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
```

```
lidar.scanAngles = linspace(-pi/2, pi/2, 360);
lidar.maxRange = 1.5;
```

En estos segmentos de codigo lo que se debe de entender es que las variables ***"lidar.scanAngles"*** tiene como parametros el rango minimo y maximo de aplitud que se va detectar el lidar, ademas de la cantidad de muestras por segundo que va a proveer este sensor, posteriormente la variable ***"lidar.maxRange"*** establece la "potencia" maxima que va a tener el sensor para poder detectar los objetos. Cabe mencionar que los valores no son muy altos pues el tenerlos de esta manera puede llegar a generar confusiones en el robot al momento de tomar ciertas decisiones.

Definir la trayectoria y la configuracion con la cual se va recorrer

- **Caso 01**

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

% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.125;
controller.DesiredLinearVelocity = 0.45;
controller.MaxAngularVelocity = 5.0;

% Vector Field Histogram (VFH) for obstacle avoidance
vfh = controllerVFH;
vfh.DistanceLimits = [0.25 60.0];
```

- **Caso 02**

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

```
% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.125;
controller.DesiredLinearVelocity = 0.45;
controller.MaxAngularVelocity = 10.0;

% Vector Field Histogram (VFH) for obstacle avoidance
vfh = controllerVFH;
vfh.DistanceLimits = [0.25 60.0];
```

- **Caso 03**

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

% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.125;
controller.DesiredLinearVelocity = 0.35;
controller.MaxAngularVelocity = 30.0;

% Vector Field Histogram (VFH) for obstacle avoidance
vfh = controllerVFH;
vfh.DistanceLimits = [0.25 60.0];
```

- **Caso 04**

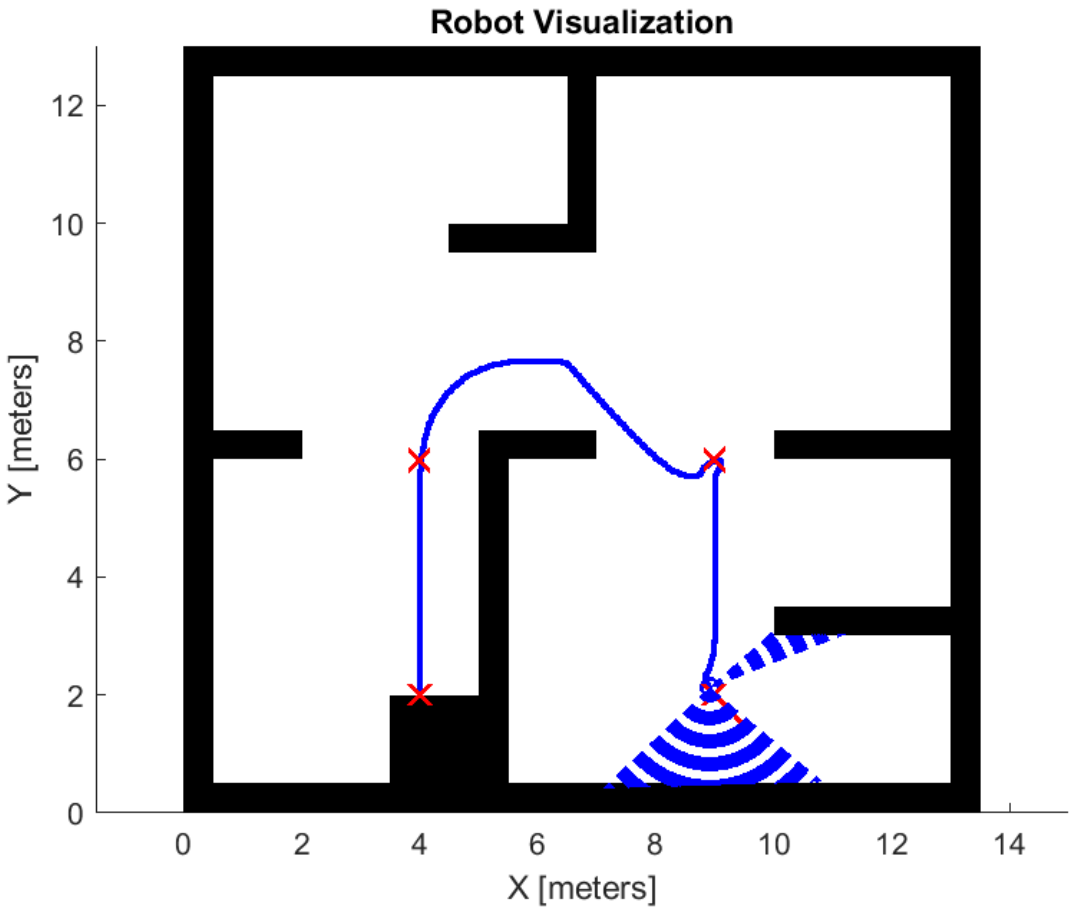
```
% Create waypoints
```

```
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 = 0.125;
controller.DesiredLinearVelocity = 0.45;
controller.MaxAngularVelocity = 20.0;

% Vector Field Histogram (VFH) for obstacle avoidance
vfh = controllerVFH;
vfh.DistanceLimits = [0.25 60.0];
```

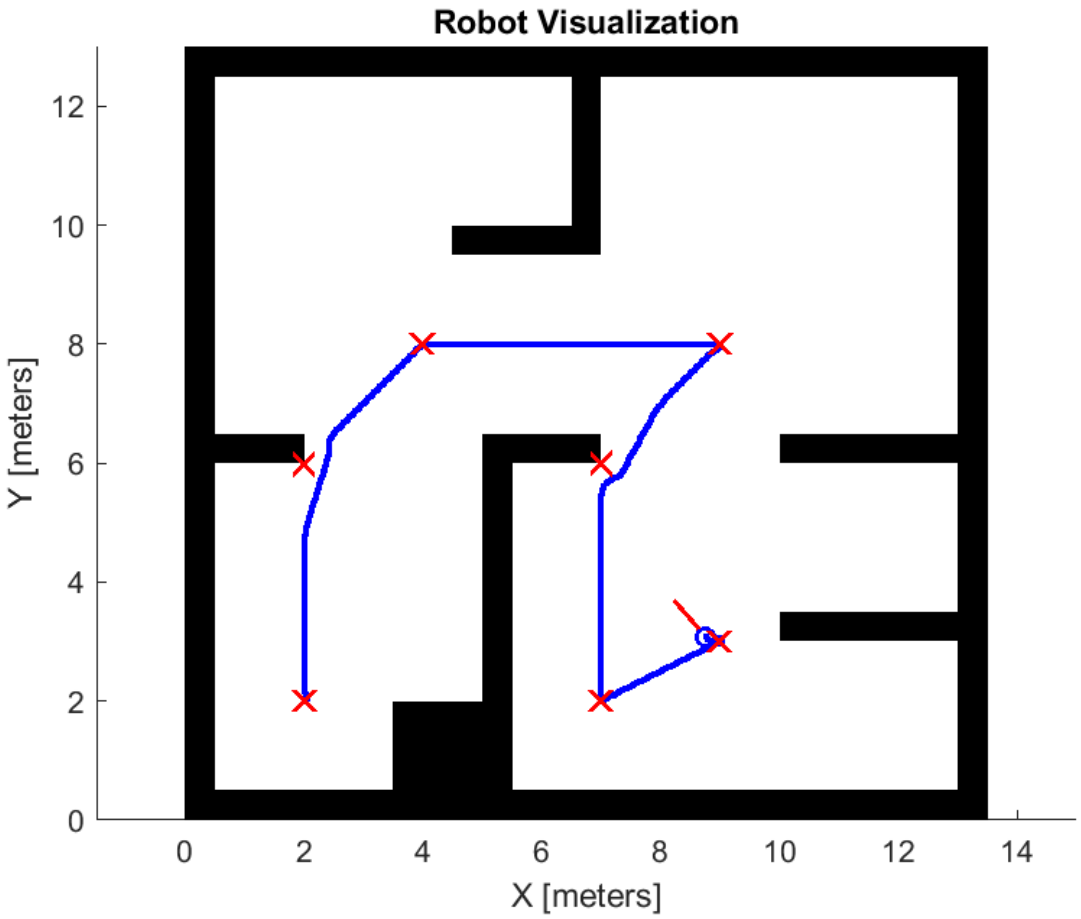
Como se puede observar la variable ***"waypoints"*** *va a ser la encargada de almacenar cada uno de los puntos que conforman la ruta, cabe mencionar que la ruta establecida por el robot no sigue el orden de los puntos como se muestra en la imagen , sin embargo la ruta que el robot sigue si los incluye todos, posteriormente la variable* **"controller.LookaheadDistance"** es la encargada de indicarle al robo que tan lejos puede ver, para las pruebas se manejo un valor muy bajo pues de esta manera se pudo aprecior una mejor precision en el seguimieto de la trayectoria, posteriormente las variables ***"controller.DesiredLinearVelocity"*** y ***"controller.MaxAngularVelocity"*** son las que establen los limites de velocidades lineal y angular que puede alcanzar el robot durante la simulación, finalemente dentro de estos bloques de codigo se debe mencionar que la variable ***"vfh.DistanceLimits"*** estable los rangos de deteción que va soportar el lidar, estos valores fueron establecidos a prueba error.

Resultados

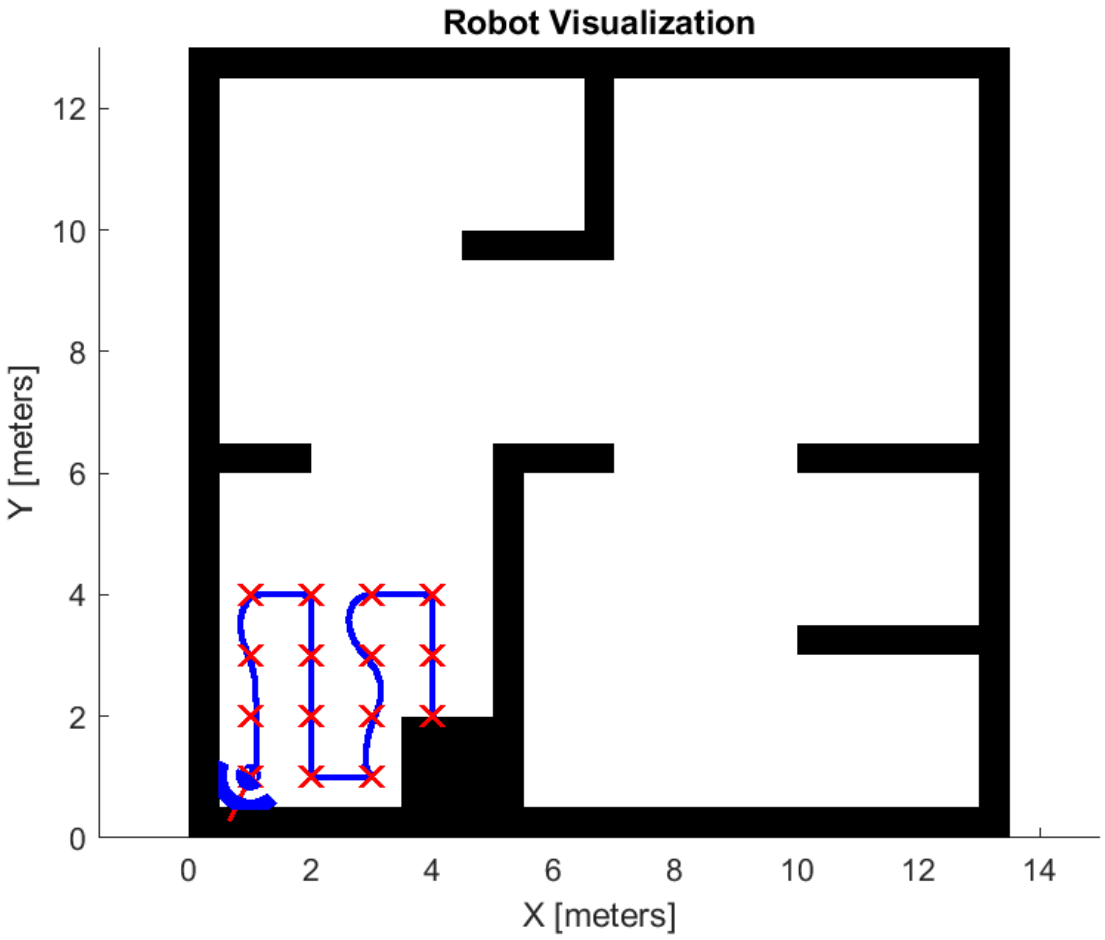
- **Caso 01**



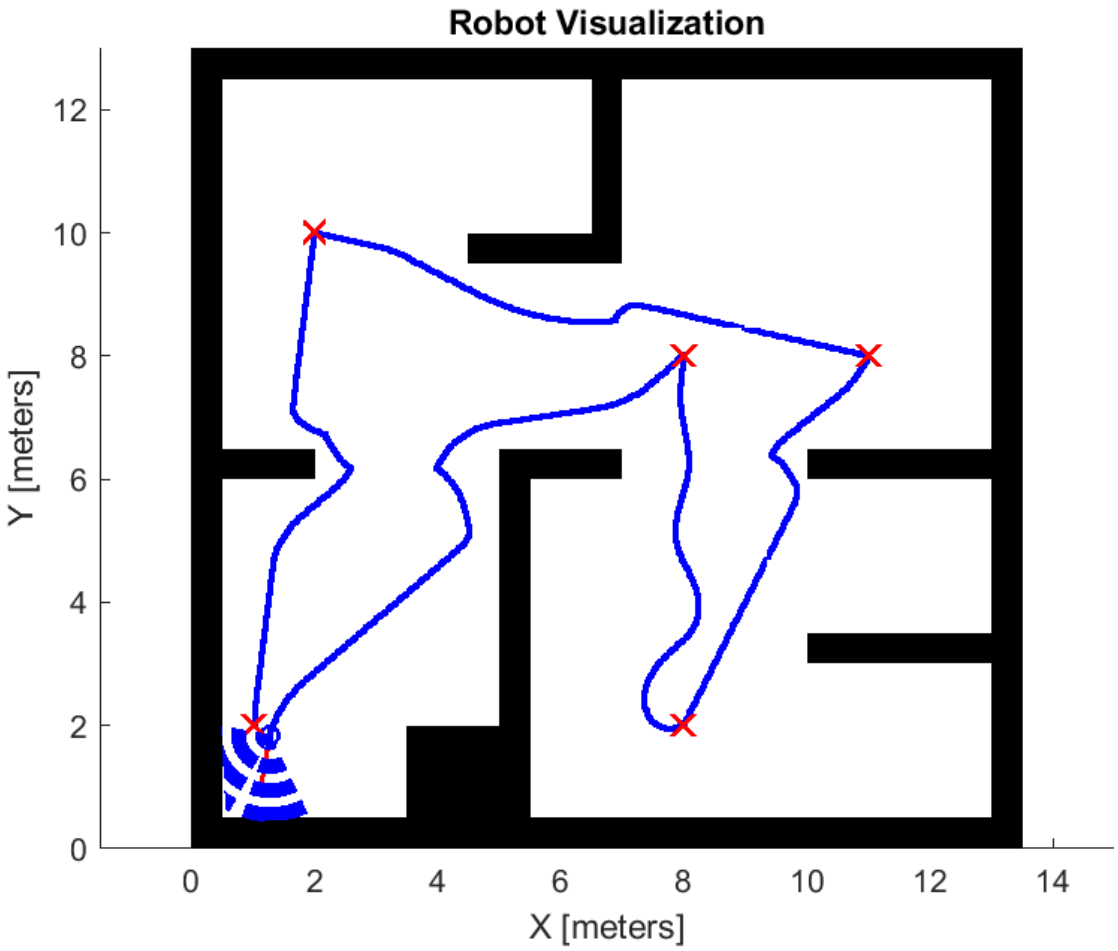
- **Caso 02**



- **Caso 03**



- **Caso 04**


Robot Visualization
Y [meters]
X [meters]
0
2
4
6
8
10
12
14