Curso 2022-2023

Robótica

Introduction to component-oriented programming and robot control

Estudiantes:
Álvaro Álvarez Zazo
Fco de Borja López Dols

Grupo 9

1º Semestre del curso 2022/23

Índice

1. Introducción a robots autónomos

1.1. ¿Qué son?

El termino robot autónomo sirve para definir a todo sistema, el cuál tiene la capacidad de realizar una determinada tarea sin la intervención del ser humano, ya sea tanto de manera directa, como indirecta.

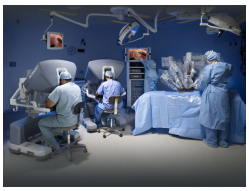
1.2. ¿Como funcionan?

Para llegar a tomar las decisiones de manera autónoma debe contar con acceso a la información acerca del entorno en el que opera.
La información a recabar le debe llegar a raíz de uno o más sensores (dependiendo del grado de autonomía del robot) tales como cámaras, sensores de proximidad, láser,... De tal manera que pueda llegar a comprender el entorno.
Una vez recabada la información, el robot actuará acorde a la situación en la que se encuentre respecto al entorno, moviéndose para ello entre los distintos protocolos de actuación previamente programados que le permitirá llevar a cabo el siguiente paso que le acerque más a la finalización de la tarea para la que fue programado.

1.3. Aplicaciones

La diversidad de aplicaciones de los robots autónomos a día de hoy son ilimitadas, pero las principales en las que se han llevado a cabo mayores progreso son las siguientes:

- **Militar** Para el uso de armamento autónomo, acciones de rastreo, seguimiento y protección.
- **Salud**En cirugías que requieran el uso de técnicas poco invasivas.
- **Atención al cliente**En la interacción con el cliente, ayudándole de una forma mas eficaz.
- **Exploración** Principalmente oceánica y espacial, con el fin de recopilar datos.
- **Industrial** Llevando a cabo tareas repetitivas y/o peligrosas, ayudando a reducir el número de accidentes, costes y errores.



(a) Robots en Salud

(b) Robots en Exploración

(c) Robots en el sector Servicios

(d) Robots en la Industria

Figura 1: Ejemplos de tipos robots

1.4. Entorno de trabajo

- **Coppelia Sim**: Simulador utilizado como interfaz basado en una arquitectura de control distribuido, donde a través de un script integrado se establece el escenario sobre el cual se ejecutará la tarea.
- **Forcefield:** Proyecto software en el cual se establecen la diferentes clases y funciones que definen el uso y comportamiento de los componentes que conforma el robot.
- **Joystick:** Periférico de entrada que consiste en una palanca o mando que gira sobre una base e informa sobre el ángulo y dirección al dispositivo que esta controlando.
- **Yolo:** YOLO (You Only Look Once), es un algoritmo que, mediante el uso de redes neuronales convolucionales, proporciona la detección de objetos en tiempo real. Este algoritmo se popularizo debido a su velocidad y precisión.

  Entre las varias aplicaciones que se le han dado, están la de detección de señales de tráfico, personas, zonas de aparcamiento y animales, dado que su uso mayoritario se ha centrado en el campo de la conducción.
- **Laser:** Dispositivo óptico que permite al robot calcular las coordenadas de los objetos, límites, etc que encuentre en su rango de propagación.
- **Camara:** Dispositivo para capturar imágenes del entorno del robot.

2. Tarea 2

2.1. Descripcion de la tarea a realizar

La tarea a realizar consiste en la programación del comportamiento del componente de un robot que lo dota de movimiento y analiza el entorno que lo rodea.
Los objetivos del comportamiento del componente es hacer que se recorra el máximo terreno adoptando 4 comportamientos distintos, estos serán activados en función de la situación en la que se encuentre el robot. También ha de tenerse en cuenta que este no estará solo sino que ha de moverse por su entorno en compañía de más obstáculos (otros

robots, objetos, muros, etc).

Los comportamientos a implementar son los siguientes:

- **Forward:** Movimiento recto y hacia delante.
- **Turn:** Parada y corrección de trayectoria.
- **Follow Wall:** Seguimiento del borde de una pared.
- **Spiral:** Movimiento en espiral.

A continuación vamos a familiarizar al lector con el robot antes de entrar en más detalle del funcionamiento de cada comportamiento.

2.2. Descripción del robot

El robot es una plataforma dotada de ruedas que realizan el movimiento de este, se encargan del avance, retroceso y giro de este.
El método que controla el cambio de velocidad y giro es *"setSpeedBase(1,adv,rot)"*
Para el análisis de su entorno dispone de un láser que abarca 2500mm de frente y un radio de 180º.

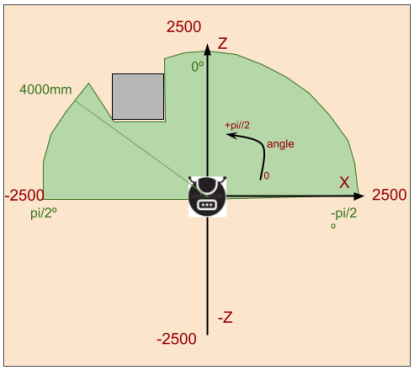


Figura 2: Esquema de amplitud del láser en el eje de coordenadas

Estos dos periféricos son los únicos que vamos a utilizar para controlar su movimiento.

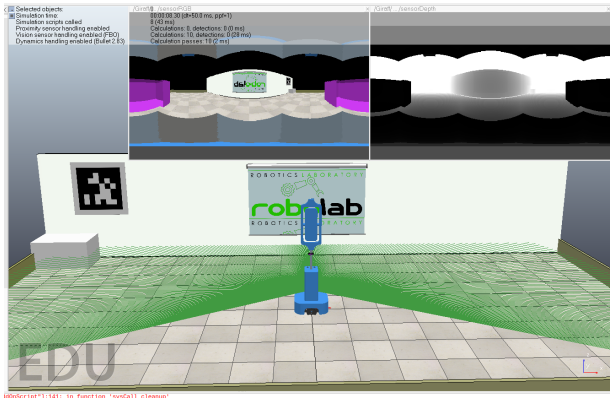


Figura 3: Vista del robot y amplitud del láser en el simulador Coppelia

2.3. Comportamiento del robot

Para desarrollar el comportamiento del robot, hemos pensado que este debe de estar en un continuo bucle de servicio en el cual se tiene que pasar por 3 etapas:

1. La lectura de datos por parte del láser del robot.
2. Interpretación de los datos a través de una máquina de estado. (4 comportamientos)
3. Guía de ordenes de control. (Asignación de valores)

2.3.1. Comportamiento del robot y máquina de estados

El robot al haber recibido nuevos datos, debe de elegir su comportamiento de acuerdo a estos.

Tras elegir cual va a ser su comportamiento devolverá una dato en forma de tupla para que sea asignado a los valores reales del robot es decir alterar los parámetros del método *"setSpeedBase(1,adv,rot)"*

A continuación se describen los estados y el comportamiento que puede tomar el robot:

Nota: Para los estados Turn, Follow wall y Spiral se divide el resultado de la captación del láser en tres secciones, copy(Frente), copyRigth(Derecha) y copyLeft(Izquierda).

- **Forward:** El cambio a este estado esta provocado por la caída por defecto de las demás condiciones de cambio, es decir siempre y cuando no se cumpla alguna de las condiciones para cambiar a otro estado el robot se mantendrá o cambiara al estado FORWARD que cambia la velocidad del robot a la máxima y cambia la rotación a 0 es decir sigue una trayectoria recta.

- **Turn:** Si la distancia a cualquier captación del láser es inferior a 500 mmm procede a ejecutar el método *TURNPROCmethod(float laserLeft, float laserRight)* cambia los valores de velocidad a 0 y rotación a una rotacion positiva o negativa en función de la captación de las secciones laterales del láser, si la captación del láser izquierdo es menor a la captacion del derecho la rotación ha de ser a la derecha para esquivar el obstáculo.

- **Follow Wall:** Este estado se activa con una variable que condiciona el comportamiento del robot entre TURN y FOLLOW WALL (es decir tiene el mismo requisito a TURN), esta variable es cambiada cada 5000ms tiempo permitiendo así al robot además de esquivar tener la posibilidad de seguir una pared cuando esta franja de tiempo está activa.

$$\left.\begin{matrix} A<\alpha<B \\ C<\alpha<D \end{matrix}\right\} = s$$

Fórmula del movimiento 'Follow wall'

El comportamiento se basa en ajustar la velocidad del robot a la máxima y corregir la trayectoria de este en función de las secciones laterales del láser, cambiando la rotación unos grados en función de la distancia mas corta entre el robot y la pared.

- **Spiral:** Si la captación de la sección frontal del láser es superior a 1500mm y las secciones laterales superan los 1000mm entonces el robot interpreta que se encuentra en un área lo suficientemente amplia para hacer un barrido del terreno en espiral. El comportamiento es el siguiente tanto la rotación del robot como la velocidad es aumentada en cada iteración del bucle de servicio principal si este no ha cambiado del estado Spiral a otro distinto (en caso de ser cambiado se restablece tanto la rotación como la velocidad a unos valores prefijados), por consiguiente incrementa poco a poco su velocidad y rotación haciendo una espiral.

$$\mathrm{s} = \mathrm{A} + \mathrm{B} * \theta$$

2.4. Pruebas-Resultados del Robot

Para probar el desempeño del robot primero hemos creado una pequeña prueba con la cual observar como se comporta el robot en solitario y como se mueve en su entorno con la definición de comportamiento dada.

2.4.1. Preparación de prueba

La interfaz de prueba es la suministrada por el profesor en la carpeta "giraffmulti", esta interfaz permite ver el comportamiento general del robot.

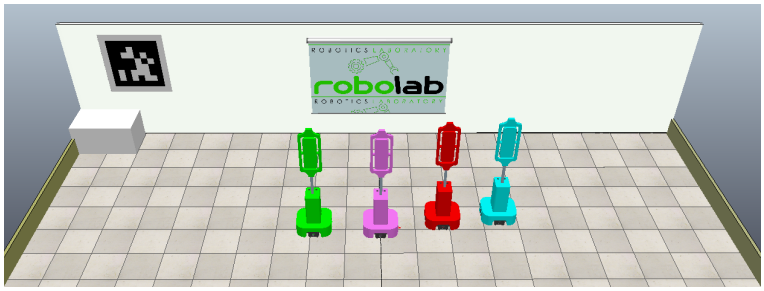


Figura 4: https://drive.google.com/file/d/1pPlOsI0BeBuLzs47H4DKClJV2T1MrSPM/view?usp=share$_{l}ink$)

Como se muestra en el anterior vídeo, el robot cambia de comportamiento dependiendo de su entorno pero es muy tosco para cubrir terreno, y esto debe de ser mejorado.

2.4.2. Ajuste de comportamiento

Los cambios que se han realizado a raíz de las pruebas son:

- **Hacer que el movimiento sea más fluido:** para ello se ha llevado a cabo un ajuste de los cambios de rotación y la velocidad en los estados (aumentar el valor de velocidad de 500m/s a 700mm/s y disminuir la rotación a 1 radian (57,2958º) ya que la rotación anterior era muy grande y tardaba mucho tiempo el robot en girar), esto llevó a que el robot se moviera más rápido pero chocara cuando alcanzaba cierta velocidad, por ello se cambio el método turn para ajustar la velocidad de frenado y la captación de la distancia de frenado en función de la velocidad actual del robot, del tal forma que podemos anticipar cuando debe el robot empezar a frenar y con que fuerza según lo rápido que este vaya.
- **Ajustar el comportamiento Spiral:** Se realizó un ajuste en el incremento de la velocidad (se aumento la velocidad en 20mm/s por cada iteración que pasaba en el estado Spiral)y el cambio de la rotación (incrementando 0,5 radianes por iteración en el estado Spiral) para que el inicio del método espiral fuera más rápido y progresará antes ya que perdía mucho tiempo haciendo una espiral bien hecha.
- **Modificación Turn:** Se cambió los grados de rotación en el método turn de forma que cubriera más terreno ya que se demoraba mucho en girar el robot y no se precisaba de una rotación tan grande para evitar el obstáculo.

2.4.3. Resultado final

Para analizar nuestro trabajo se ha hecho una prueba de recorrido del robot en un entorno suministrado de nombre aspirador, formado por una sala cortado a la mitad con un muro, con una abertura en medio que conectaba ambas mitades de la sala en el cual se analiza el terreno recorrido por este en un tiempo de 5 minutos y como resultado se ha obtenido lo siguiente:

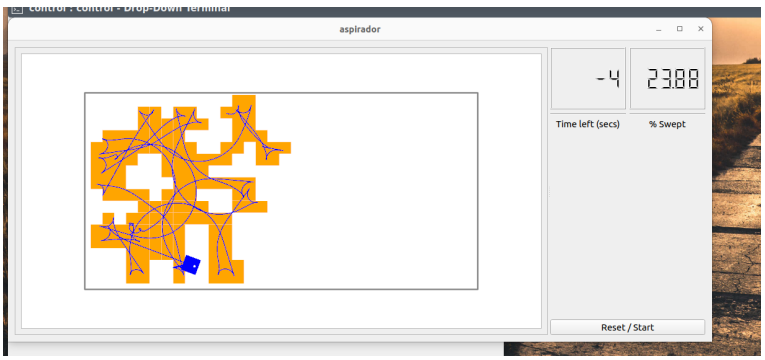


Figura 5: Prueba inicial previa a cambios

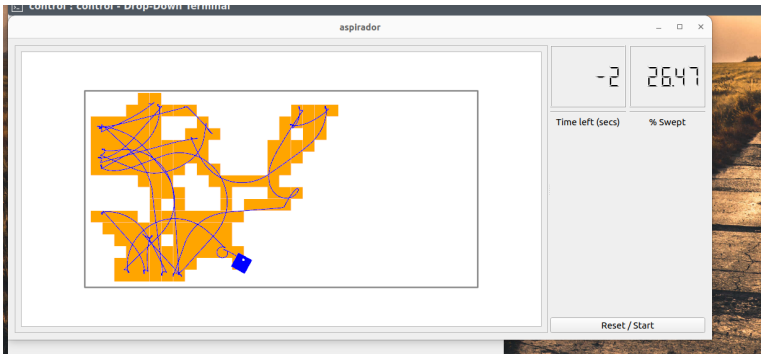


Figura 6: Prueba final tras haber realizado las pruebas

2.5. Conclusiones y futuras mejoras

Como conclusión final es que a pesar de no haber implementado el método "Follow wall.[el] robot no tiene mal desempeño, sabe reaccionar muy bien ante los obstáculos y no se mueve mal con los demás comportamientos que si están implementados, no obstante necesita implementar un método de seguir pared, ya que cuando hay habitaciones muy estrechas o puertas, este no sale de ese área y eso es un problema.

Como futuras mejoras aparte de la implementación del método "Follow wall", veríamos apropiado

- Hacer ajustes al cambio de estado "Forward", ya que creemos que recorrería más espacio dando más protagonismo a este estado, de esa forma, se debería ajustar el método espiral subiendo los parámetros de cambio a este estado, es decir que entre en modo espiral cuando detecte un espacio mas amplio.
- Trataría de ajustar mejor la adaptación de tiempo de reacción y frenado del robot para poder llevar el robot a velocidades más altas,
- Habría que hacer las pruebas pertinentes para ver si estas modificaciones mejoran el desempeño del robot.

3. Tarea 3

3.1. Descripción de la tarea a realizar

La tarea a realizar consiste en el desarrollo de un software que dote al componente anterior de la capacidad de moverse de acuerdo a los objetos que haya en su entorno, además de la capacidad de reconocer salas con el fin de trazar un grafo que describa el entorno.

Es decir se trata de hacer un robot explorador que nos informe de las salas y el contenido de estas.

Al ser una tarea compleja hemos decidido dividir el trabajo en 2 Fases:

- **Fase 1: Reconocimiento de objetos y movimiento** En esta fase, se tratará de dotar al robot de la capacidad de reconocer objetos y moverse hacia ellos.

- **Fase 2: Interpretación del entorno** En esta fase se dotara al robot de la capacidad de moverse entre salas e identificar a estas con el fin de trazar un mapa de su entorno.

4. Fase 1: Reconocimiento de objetos y movimiento

4.1. Descripcion de la tarea a realizar

La tarea a realizar consiste en la modificación del comportamiento del componente anterior, dotándolo ahora de la capacidad de reconocer objetos.Para el reconocimiento de los objetos se hace uso del algoritmo YOLO.

Los objetivos del comportamiento del componente se basan en la utilización el hardware del robot (laser y cámara superior) para realizar una función de barrido que permita reconocer los objetos y la aproximación al más cercano. Para llevar a cabo los objetivos planteados, dotaremos al robot de 4 comportamientos entre los que iterará en función del estado en el que se encuentre.

Estos 4 comportamientos serán los siguientes:

- **Idle:** Estado inicial del robot.
- **Wait:** Espera debido a la perdida del objetivo.
- **Aproach:** Aproximación hacia un objetivo siguiendo una trayectoria lineal..
- **Search**: Búsqueda de un objeto en su entorno.

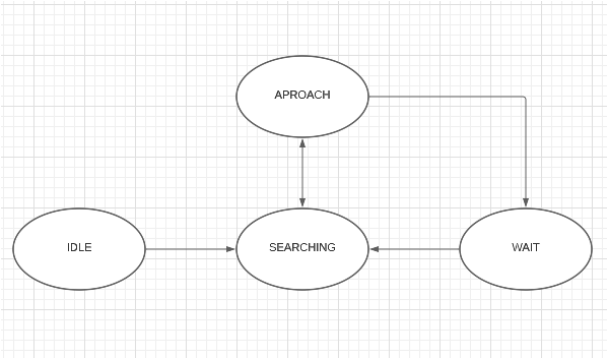


Figura 7: Máquina de estados

4.2. Descripcion del robot

Para esta entrega, el robot sigue siendo el mismo que el usado para realizar la Tarea 2. La diferencia está en que dado que se deben de reconocer objetos del entorno, será necesario el uso de la librería YOLO, la cual nos permitirá llevar a cabo las funciones requeridas para esta tarea. Por otro lado, en esta tarea se hará uso de la cámara frontal superior, la cual suministrará al componente imágenes RGB, que servirán para que el algoritmo YOLO realice las tareas de análisis y verificación de los objetos existentes en el entorno.

4.3. Comportamiento del robot

Para desarrollar el comportamiento del robot, éste debe de estar en un continuo bucle de servicio en el cual se tiene que pasar por 3 etapas:

1. La lectura de datos por parte del láser y la cámara frontal superior del robot.
2. Interpretación de los datos a través de una máquina de estado. (3 comportamientos)
3. Guía de ordenes de control.

4.3.1. Comportamiento y maquina de estados

Los comportamientos a implementar son los siguientes:

- **Idle:** Estado ocioso (no tiene ninguna funcionalidad) este estado se establece en para la inicialización del robot.
- **Wait:** Este estado es activado cuando el robot pierde su objetivo, se establece un intervalo de tiempo en el cual el robot mantiene la misma trayectoria que cuando tenía el objetivo y en el caso de que termine ese intervalo sin recuperar el objetivo, el robot cambiará su estado a SEARCHING(iniciará la búsqueda de un nuevo objetivo). En caso de haber recuperado el objetivo dentro del intervalo volverá al estado de aproximación APROACHING (Aproximación a un objetivo).
- **Aproach:** Estado de aproximación hacia un objetivo detectado siguiendo un trayectoria rectilínea. En caso de encontrarse a una distancia inferior a 900mm se entenderá que el robot a llegado al objetivo y pasará al estado SEARCHING(en búsqueda de un nuevo objetivo). A lo largo de la aproximación hacia el objetivo se irá comprobando que no haya otro objeto anterior al objetivo actual, en cuanto a la distancia con respecto al robot, ya que en caso de detección de otro objeto, este pasará a ser el nuevo objetivo. En caso de no detección del objetivo, incluido el predeterminado, se entenderá que ha perdido el objetivo destino, para lo cual pasará al estado WAITING(estado de espera).
- **Search**: Estado de búsqueda de un objeto, recibe por parámetro un conjunto de objetos que han sido reconocidos por el algoritmo de YOLO, recorre dicho conjunto en búsqueda del primer objeto que no sea del tipo robot (mediante uso de la función *find_if_not()*y lo establece como objetivo actual. *robot.set_current_target(objetivo);*. Una vez se haya establecido el objetivo se cambia el estado a APROACHING (aproximación al objetivo).

4.4. Pruebas-Resultados del robot

4.4.1. Preparación de la prueba

Para probar el comportamiento del robot, se ha usado un escenario proporcionado por el profesor, el cuál consta de un único objeto de tipo “persona”, que deberá detectar en primer lugar, aproximarse hasta una cierta distancia declara por nuestra parte, y una vez llegada, volver a realizar la búsqueda de otro objeto distinto.

4.4.2. Resultado final de la prueba

Como se puede apreciar en el vídeo adjuntado en la figura 7, los resultados obtenidos son bastantes positivos, debido a que en el escenario de ejemplo, la sala consta de 1 objeto de tipo persona y otro objeto de tipo silla, y a lo largo de la prueba, efectivamente lleva a cabo el reconocimiento y aproximación por partida doble al objeto persona y una al objeto silla, en la medida en la que estos son reconocibles a la cámara del robot.

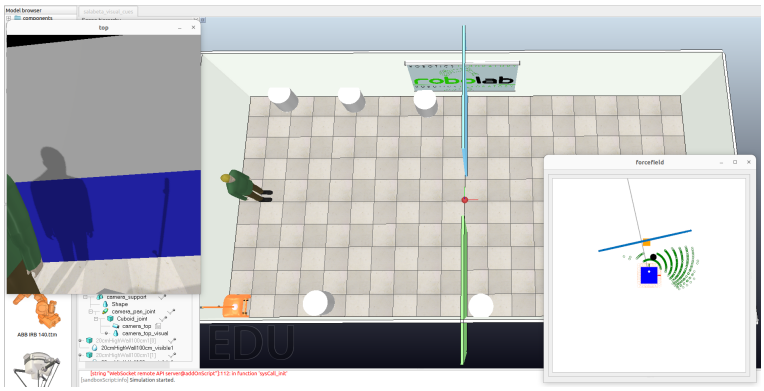


Figura 8: https://drive.google.com/file/d/10W7Iod1JECLNSQjql0bUb4BkK9pb5mV0/view?usp=share$_l ink$

4.5. Conclusiones y futuras mejoras

Los resultados obtenidos son muy buenos no obstante se debe llevar a cabo mejoras respecto a la recuperación de objetivos en caso de perdida y reconocimiento de la habitación (detectar puertas). Estas mejoras se harán en la futuras versiones del robot.

Por otro lado siempre es necesario probar más y distintos escenarios para ver su desempeño en orden de hacer al robot lo más preciso posible.

5. Fase 2: Interpretación del entorno

5.1. Descripción de la tarea a realizar

El objetivo de esta tarea es dar sentido al trabajo realizado a la fase anterior, utilizando la función de reconocer y moverse hacia objetos. Para ello se dotará al robot de la capacidad de interpretar el entorno. A continuación se expondrán cada uno de los problemas y soluciones para la realización de la tarea.

5.2. Reconocimiento de salas

El reconocimiento de salas es una funcionalidad difícil de implementar, ya que una sala es un concepto algo abstracto, según nuestra interpretación, una sala es **un espacio separado por una o más puertas que puede o no contener objetos**, entonces para que el robot sea capaz de reconocer de manera eficiente cuando esta en una sala y en que sala esta, debe de saber reconocer puertas y guardar los objetos que forman esa sala.

Así pues debemos resolver las siguientes cuestiones:

- **¿Como reconocer puertas?**
- **¿Como identificar salas?**

5.2.1. Reconocimiento de puertas

El reconocimiento de puertas conlleva un gran problema, el software de YOLO no es capaz de reconocer puertas, ya que una puerta no es un objeto como tal, sino que más bien es una situación del entorno, es decir, una puerta es un hueco en una pared con unas medidas determinadas. Por tanto, lo primero que debemos hacer es crear un método del componente capaz de reconocer huecos en la pared y tras esto, buscar la forma de que la puertas sean interpretadas como objetos a los ojos de nuestro robot.

- **Método de reconocimiento de puertas detect:** El método de detect, encuentra todos los picos al alcance del láser del robot, y filtra los picos que no definen una puerta, es decir los picos espurios que no guarde la relación correcta entre sus distancias para ser una puerta.
- **Creación de la clase Generic Object:** Se necesita crear una nueva clase que englobe a las puertas halladas y los objetos de Yolo, ya que Yolo no tiene soporte para la detección de puertas. Esta nueva clase GenericObject tiene el cometido de hacer los objetos Yolo y las puertas comparables, dotando a las puertas de características que definen a los objetos Yolo, todo esto, con el fin de poder trabajar con los objetos en una misma estructura de datos, un vector de objetos genéricos.

5.2.2. Identificación de salas

La identificación de salas es imprescindible de cara al correcto trazado de la ruta, dado que el final de la ejecución llegará el momento en el que el robot vuelva a pasar por una sala por segunda vez.
Para identificar y así diferenciar una sala de otra no visitada, se usará el conjunto de objetos hallados (incluyendo la cantidad de cada uno, en caso de repetición), dado que tanto en el escenario propuesto, como en un caso real, no existen dos salas que contengan idénticamente el mismo conjunto de objetos.

5.3. Trazado de mapa

La base de esta tarea es la de realizar un trazado, por el cual sea capaz el robot de cruzar a través de todas las salas del escenario, y a su vez reconocer si ya se ha pasado por esa sala. La estructura sobre la cuál basarse a la hora de llevar a cabo la tarea, será la de un grafo, en el que los nodos serán cada una de las salas contenidas en el escenario, y los enlaces, las puertas. Para realizar el cometido, el programa se ha desarrollado siguiendo el siguiente orden de ejecución:

1. Dado que al comenzar la ejecución, el robot se encuentra en una sala en estado IDLE (ocioso), pasará al estado SEARCHING, para reconocer todos los objetos que se encuentren en la sala y hacer la comparación con el conjunto de salas visitadas hasta ahora.

   - **Ya está guardada:** Finalizará la ejecución.
   - **No esta guardada:** Se pasará al paso 2.

2. Se guardará en el objeto grafo la sala con su vector de objetos, como un nodo de dicho grafo.

3. De entre los objetos hallados, se comprobará si existe una puerta.

   - **En caso afirmativo:** Se pasará a un estado APPROACH (aproximación) hacia la puerta, y el guardado de la puerta como enlace entre el nodo recién guardado y la siguiente sala.
   - **En caso negativo:** Se finalizará la ejecución del programa ya que se interpretará que ya se ha recorrido el mapa.

4. Una vez avanzado hasta la puerta, se entrará en un nuevo estado, llamado CROSS, en el que se cruza la puerta y a su vez se actualiza el objeto grafo. Finalmente, se retornará al paso 1, ya que se encontrará en otra sala.

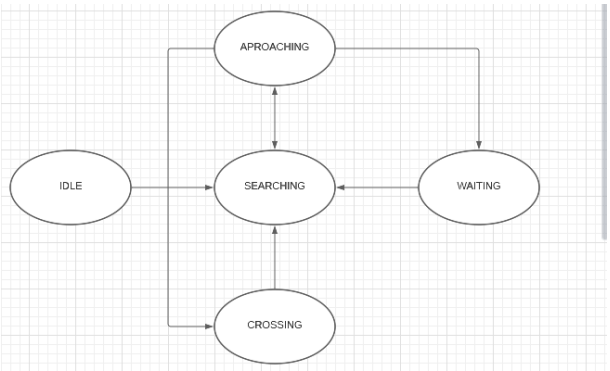


Figura 9: Máquina de estados

5.4. Pruebas-Resultados

5.4.1. Preparación de la prueba

Para comprobar el comportamiento el robot, se ha usado un escenario, el cuál consta de 6 salas, comunicadas entre si a través de huecos en las paredes, que trataremos como puertas, y distintos conjuntos de objetos en cada una de ellas, a fin de hacer distinguibles unas de otras. El robot, para este escenario, deberá de recorrer las seis salas, sin ningún impedimento.

5.4.2. Resultado final de la prueba

Como no hemos podido lograr una versión funcional que implemente todas las funciones necesarias para trazar el mapa del entorno del robot, hemos decidido crear esta gráfica, que muestra un ciclo de la maquina de estado que condensa los pasos que daría el robot en su correcto funcionamiento.

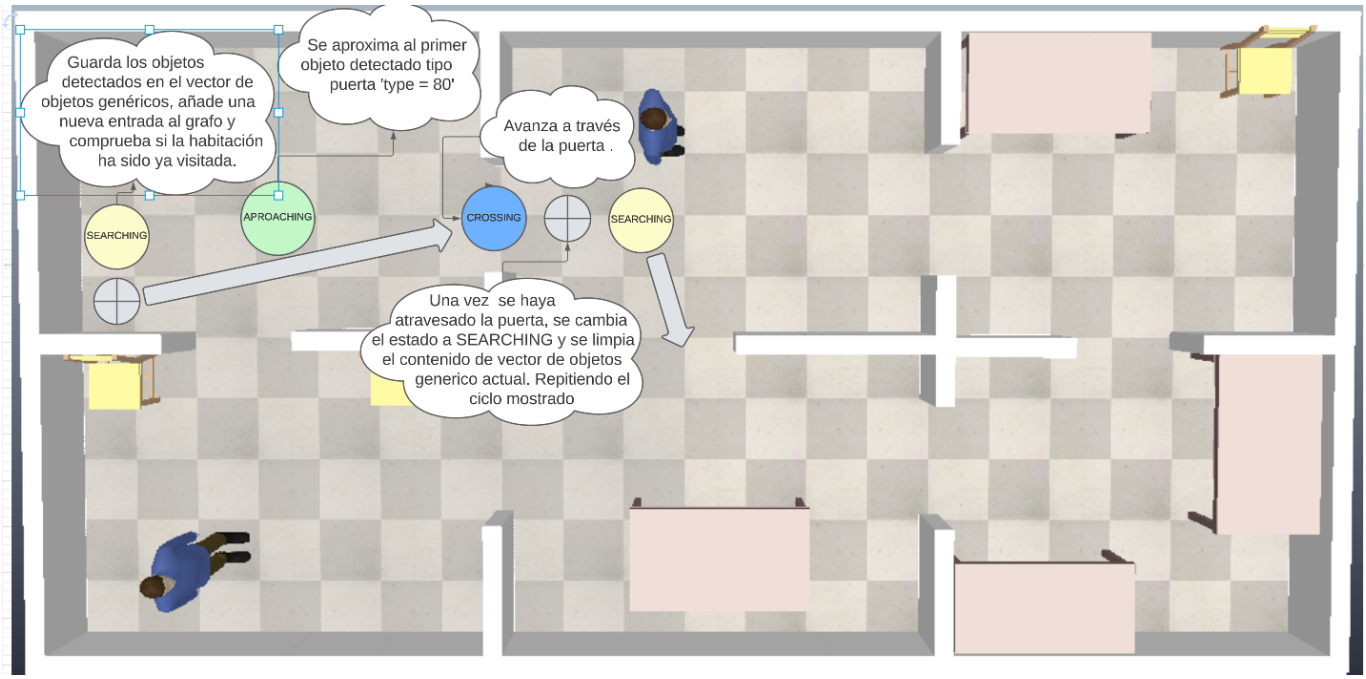


Figura 10: Modelo de comportamiento del robot

6. Bibliografía

- **Consultas sobre manejo de overleaf:** https://www.overleaf.com/learn/latex/Questions
- **Consultas sobre programación en C:** https://stackoverflow.com/
- **Introducción a los robots autónomos:** https://tiposderobot.com/