Proyecto Final de Robots Móviles

Planeación de Acciones Usando una Representación Espacio-Estado

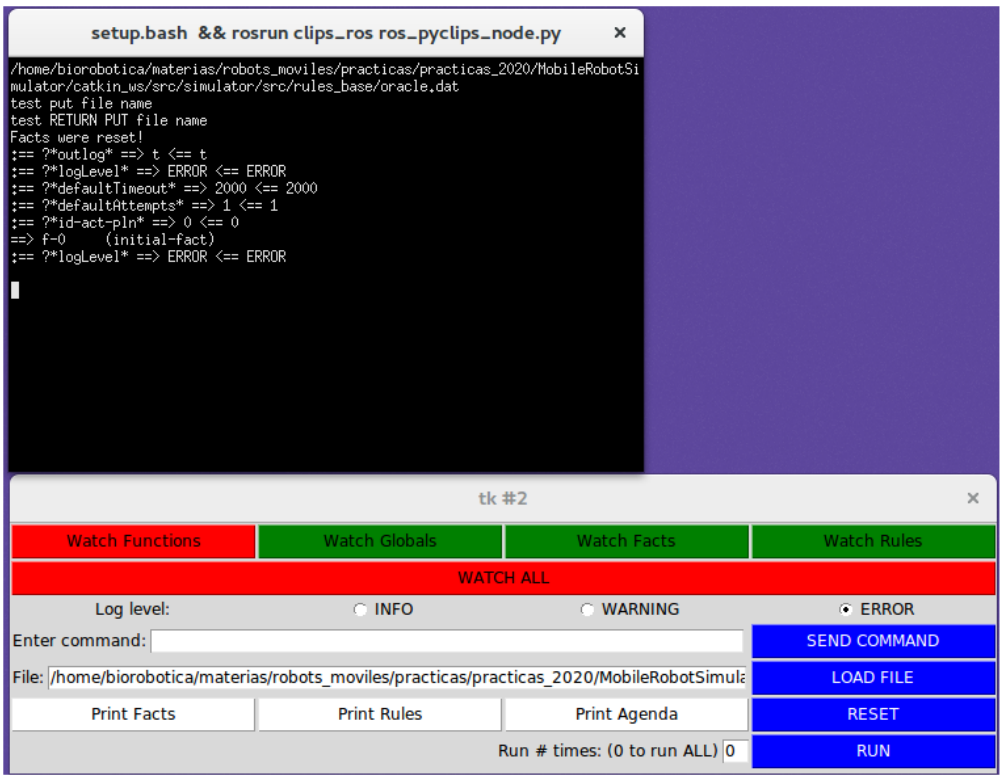
Objetivo: Familiarizar al alumno con la planeación de acciones usando sistemas basados con una representación espacio-estado.

Desarrollo: Para cada uno de los siguientes apartados, realizar los programas que se piden.

Fecha de entrega: Día del examen final.

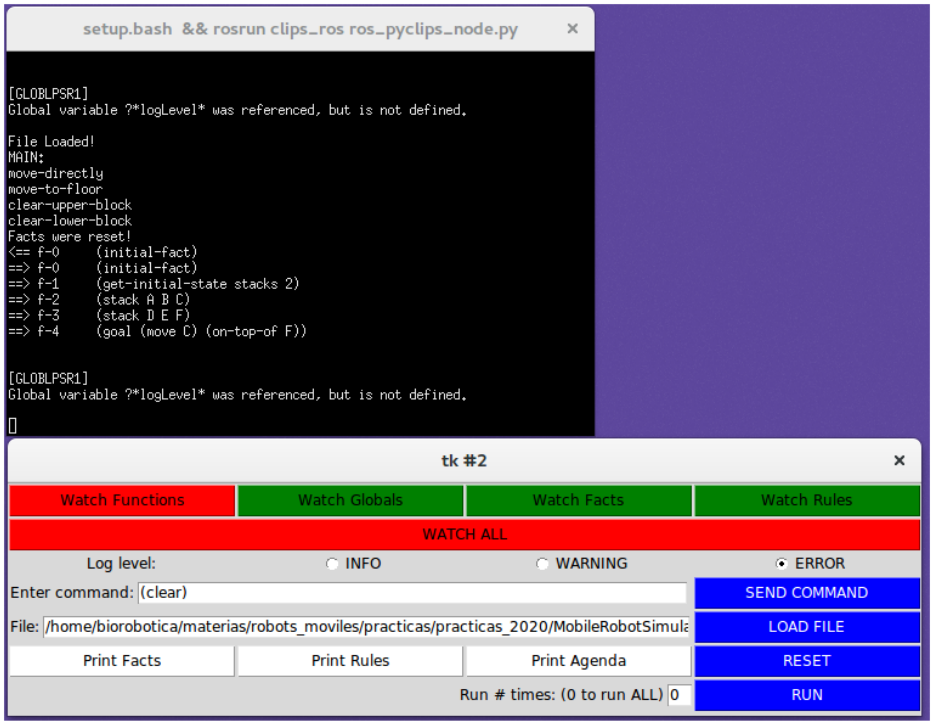
1.- El archivo:
"catkin_ws/src/simulator/src/action_planner/ViRBot_Cubes_ROS/cubes_stacks_original.clp" contiene el código de un sistema que controla la manipulación de cubos en diferentes configuraciones, usando el lenguaje basado en reglas CLIPS.

1.1.- Para probar este código y su funcionamiento, habiendo comenzado el simulador de robots con start.sh, localice la terminal que muestra los mensajes de Clips y el GUI que maneja los comandos a CLIPS, como se muestran en la siguiente figura:

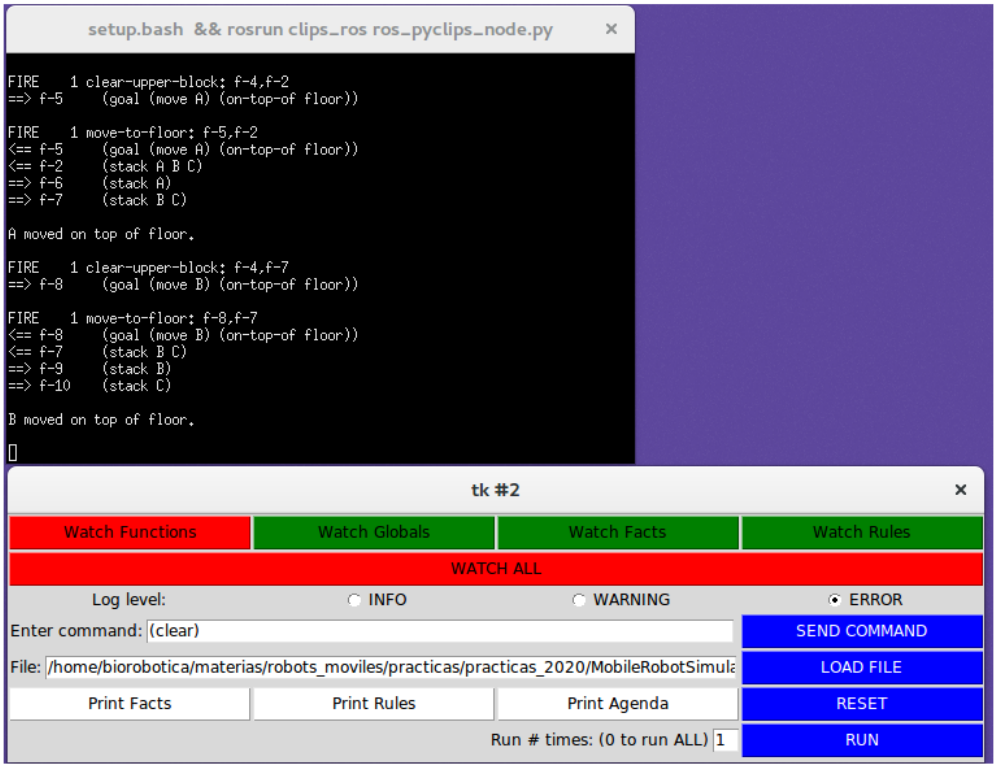


1.2.- En el campo de File seleccione el archivo antes mencionado y carguelo al sistema de Clips oprimiendo el botón de “LOAD FILE”. Si hubiera errores de compilación aparecen en la terminal que maneja los mensajes de Clips.

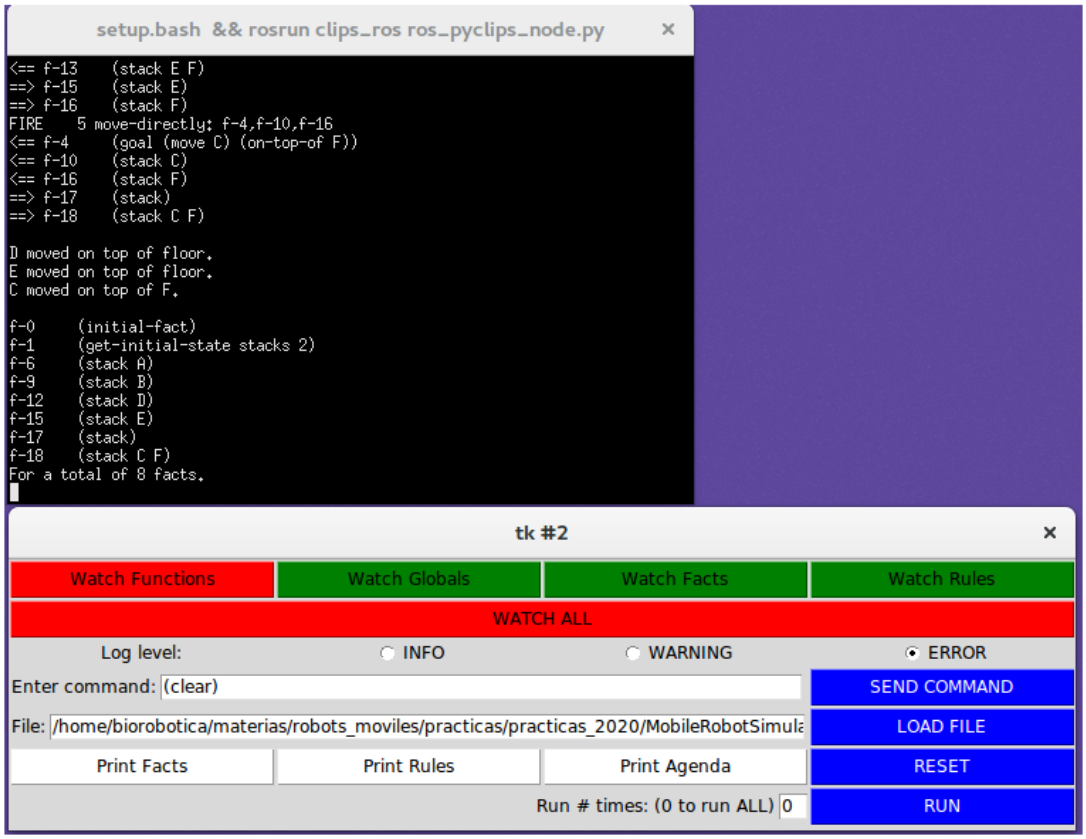
1.3.- Oprima el botón de “RESET” y observe la configuración inicial de los cubos en la terminal.



1.4.- En el campo “Run # times” coloque 1 para ejecutar paso a paso la ejecución de las reglas en Clips, disparándose una regla a la vez. Si se desea ejecutar más reglas cambiar este número; con 0 se ejecutan las reglas continuamente y se detiene cuando ya no haya más reglas que se disparen. Seleccione el botón “RUN” y vea la regla que se ejecutó, así como los hechos nuevos y los hechos borrados.



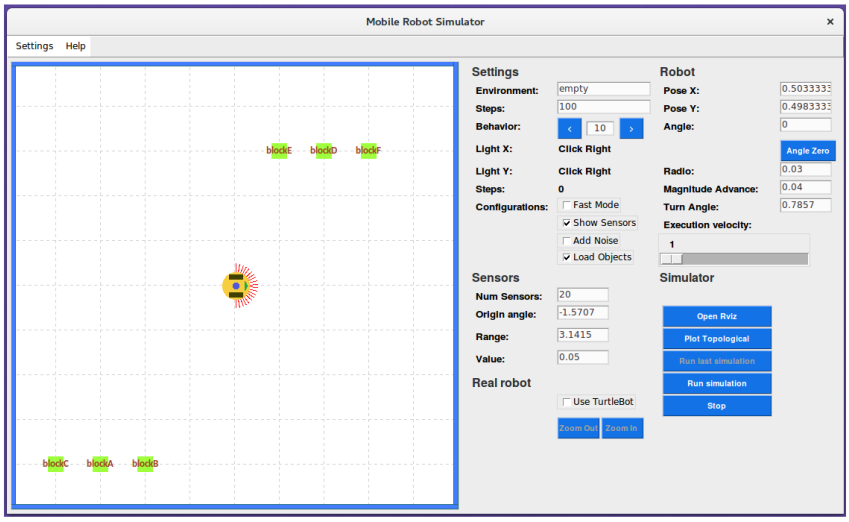
1.5.- Durante la ejecución del código de Clips se pueden ver los hechos que están en el sistema oprimiendo el botón “PRINT FACTS”.



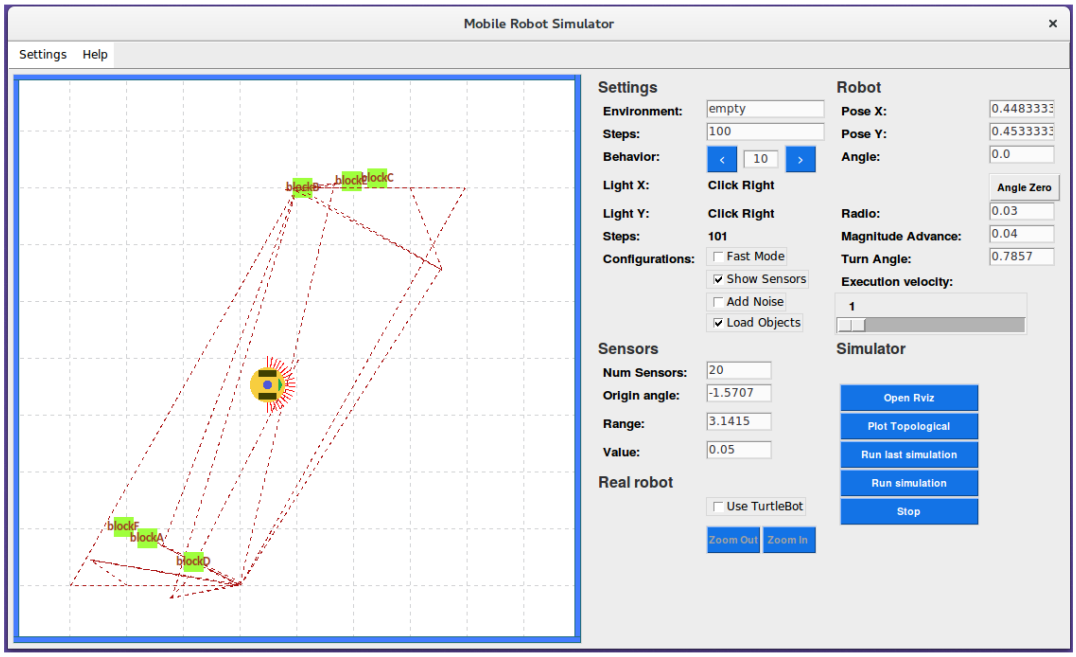
2.- La siguiente figura muestra una configuración de 6 bloques en el medio ambiente “empty”. Los bloques son cargados seleccionando la pestaña “Load Objects”. La posición y nombre de éstos se encuentran en el archivo:

catkin_ws/src/simulator/src/data/objects/objects.txt

blockA .2 .1
blockB .3 .1
blockC .1 .1
blockD .7 .8
blockE .6 .8
blockF .8 .8



Utilizando un versión modificada del programa basado en reglas mostrado en el punto 1 que ordena cubos, los bloques de la figura anterior son ordenados en otra configuración utilizando planeación de acciones. Para ver esta simulación seleccione el medio ambiente “empty”, el comportamiento 10, coloque los objetos con “Load Objects”, ponga el robot en cualquier posición y oprima el botón “Run simulation”. La posición final de la simulación se observa en la siguiente figura:



La selección del comportamiento 10 manda llamar la función action_planner() la cual se encuentra en “catkin_ws/src/simulator/src/action_planner/action_planner.h”.
En esta función se carga, en el nodo del sistema experto Clips, el archivo batch “catkin_ws/src/simulator/src/action_planner/ViRBot_Cubes_ROS/ROS_cubes.dat” el cual contiene varios archivos tipo Clips:

2.1.- objects_deftemplates.clp
Este archivo contiene las planillas utilizadas para representar los objetos, los cuartos, las configuraciones de los objetos y los objetivos.

2.2.- virbot_initial_state.clp
Este archivo contiene el estado inicial de los objetos, en donde están colocados y orden de sus posiciones, así mismo tiene la configuración final.

Hay que hacer notar que la posición de los objetos en este archivo debe coincidir con la que se encuentra en el archivo objects.txt antes mencionado.
Por ejemplo, el bloque blockA en virbot_initial_state.clp esta definido como:
(item (type Objects) (name blockA)(room bedroom)(zone deposit)(image blockA)(attributes pick)(pose 0.2 0.1 0.0))

En objects.txt esta definido como: blockA .2 .1

Estas posiciones deben coincidir, ya que en el simulador, cuando el robot va a recoger un objeto y llega a la posición conocida de éste, si el objeto no se encuentra ahí el robot no lo va poder recoger.

2.3.- ros_clips.clp
Este archivo contiene las reglas que se utilizan para recibir y mandar comandos a otros nodos de ROS.

2.4.- planning_cubes.clp
Este archivo contiene las reglas para hacer una planeación Espacio-Estado, para que dado un estado inicial de los bloques y un estado final planear las acciones para lograr ese objetivo.

2.5.- exe_plan.clp
Dado el plan encontrado, este archivo contiene las reglas para ejecutarlo usando OPERADORES que se encuentran en otros nodos de ROS.

Por ejemplo para llevar el bloque C que se encuentra en el cuarto “bedroom” en la zona “deposit” y se quiere llevar a un lugar libre del cuarto “livingroom” en la zona “storage”, se necesitan hacer las siguientes acciones:

a) Ir primero al cuarto en donde se va a tomar el objeto:
goto bedroom deposit
El nodo de ROS que hace esta operación es “motion_planner_node”

b) Encontrar el objeto: find-object blockC
El nodo de ROS que hace esta operación es “find_obj_node", debido a que por el momento no se puede usar un robot real, solamente se simula la detección del objeto.

c) Acercarse al objeto: mv blockC
El nodo de ROS que hace esta operación es “motion_planner_node”

d) Tomar el objeto: grab blockC
El nodo de ROS que hace esta operación es “manipulator_node", debido a que por el momento no se puede usar un robot real, solamente se simula esta manipulación.

e) Ir al cuarto en donde se se va a dejar al objeto:
goto livingroom storage

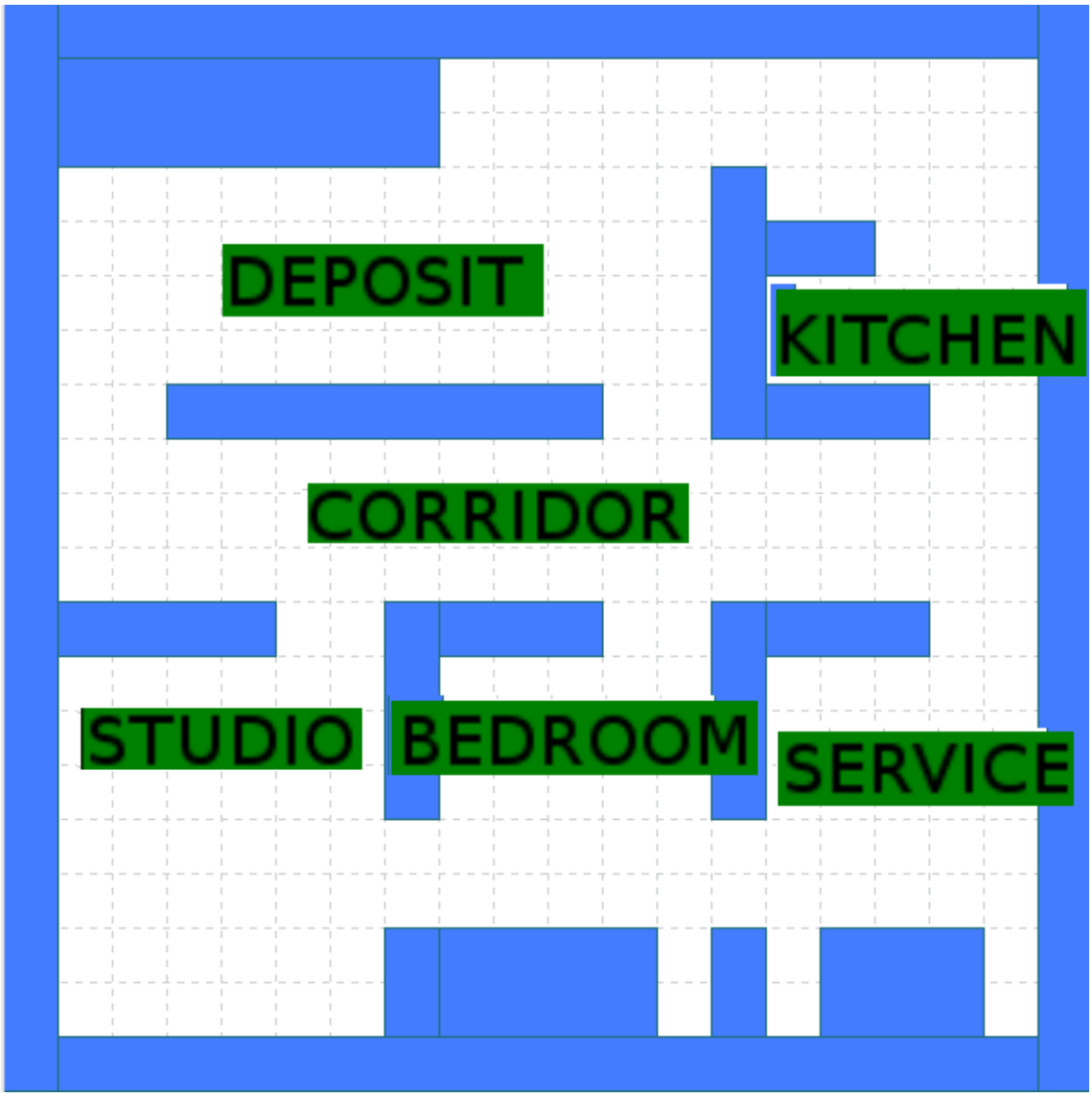
f) Encontrar un espacio libre para dejar el objeto: find-object freespace

g) Ir a ese lugar: go freespace
El nodo de ROS que hace esta operación es “motion_planner_node”

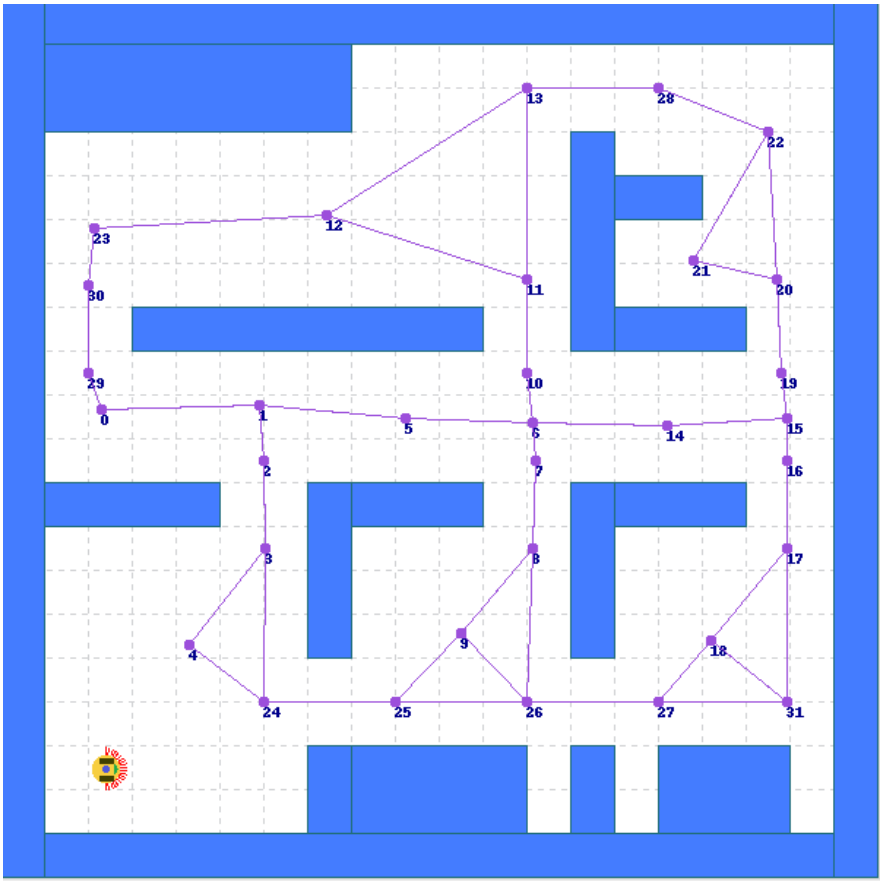
h) Dejar el objeto: drop blockC
El nodo de ROS que hace esta operación es “manipulator_node", debido a que por el momento no se puede usar un robot real, solamente se simula esta manipulación.

Vuelva a ejecutar la simulación, pero ahora con el número de pasos igual a 1, para que vaya observando como se va ejecutando el plan generado por el planeador de espacio estado.

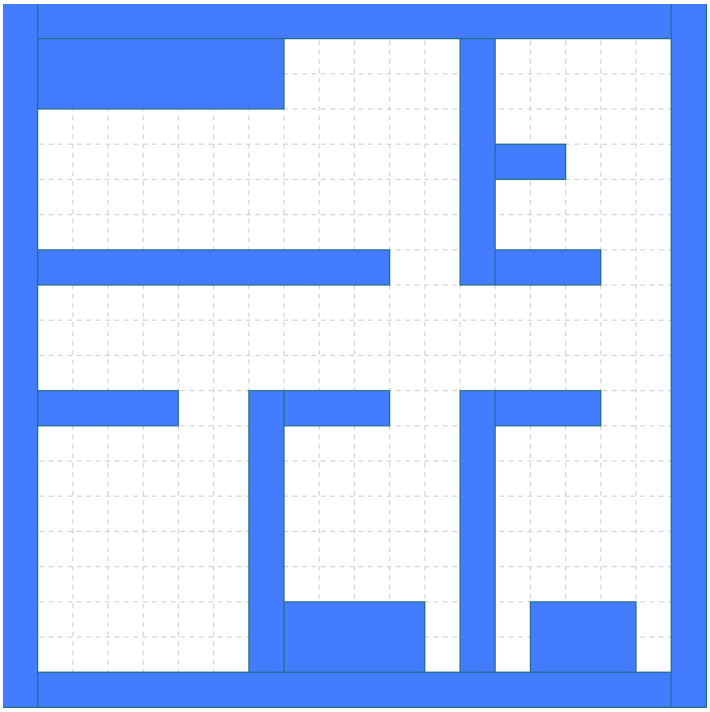
3.- En siguiente figura se muestra la representación simbólica de la configuración de una casa, la cual contiene 5 cuartos: “Kitchen, Bedroom, Corridor, Service, Studio y Deposit”.



Encuentre un mapa topológico en forma manual, como el que se muestra en la siguiente figura, con varios nodos en cada cuarto, que permita la navegación de un robot desde cualquier origen a un destino, utilizando campos potenciales o algoritmos de máquinas de estados y cualquiera de los algoritmos de búsqueda de caminos Dijkstra o A*



Considere que durante la navegación algunas puertas pueden estar cerradas, como se muestra en la siguiente figura, cuando el robot encuentre esta situación el robot debe encontrar rutas alternativas.



4.- En el cuarto “Corridor”, en el lado izquierdo de éste, el robot que entrega productos de Amazon, dejo en la puerta de entrada dentro de la casa, los siguientes productos: Apple, Sushi, Milk, Soap, Perfume y Shampoo.

Los siguientes productos tienen que ser llevados por la robot de la casa, al refrigerador de la cocina: Apple, Sushi, Milk.
La manzana esta encima de la caja del Sushi, ésta encima de la leche, la leche esta en el piso. Se necesita guardar en el refrigerador primero la leche, después la manzana y por último el Sushi.
Los demás objetos están encima de otros en este orden: Soap, Perfume, Shampoo; se deberá llevar primero el Shampoo, después el Perfume y al último el jabón, a la mesa del cuarto de servicio. El robot se encuentra originalmente en el cuarto STUDIO.

Modifique los archivos de Clips del planeador de acciones descrito en el punto 2 para lograr lo que se esta solicitando. Este sistema deberá ser flexible con respecto a solucionar cualquier configuración inicial y final, además de evadir obstáculos desconocidos. Muestre con el simulador la operación del robot llevando los objetos y colocandolos en el orden preestablecido.

5.- Repita algo parecido en el punto 3, pero ahora el robot debe llevar primero un libro a un niño, que se encuentra en el cuarto y después un martillo a la Mamá, que se encuentra en el estudio. El martillo se encuentran en un mueble del deposito; el libro esta en un librero del estudio.

6.- Como punto extra asuma que algunos de los objetos no se encuentran en la posición indicada y que el robot los necesita buscar en otros cuartos.