Reporte de Proyecto:
Grocery Shopping con Manipulador Movil

Kevin Muñoz (15358)

***Resumen*—El proyecto consiste en realizar una simulacion con ayuda de la herramienta V-REP conectada con MatLab para poder realizar el recorrido de un modelo simplificado de una casa con el robot mobil Pioneer y utilizando un brazo robotico UR3 para colocar objetos en diferentes locaciones dentro de la casa.**

I. Descripción

El proyecto consiste en realizar el recorrido de un modelo de nuestra casa realizado en V-REP, y un manipulador UR3 permite que se muevan objetos desde la entrada de la casa a diferentes lugares dentro de la casa. Se realizo el recorrido de la casa por medio del algoritmo D estrella, desde una posicion inicial hasta una posicion final, y siguiendo el recorrido que se creo. Por medio de cinematica inversa y directa se utilizo el manipulador robotico para sujetar los objetos y luego dejarlos sobre una superficie.

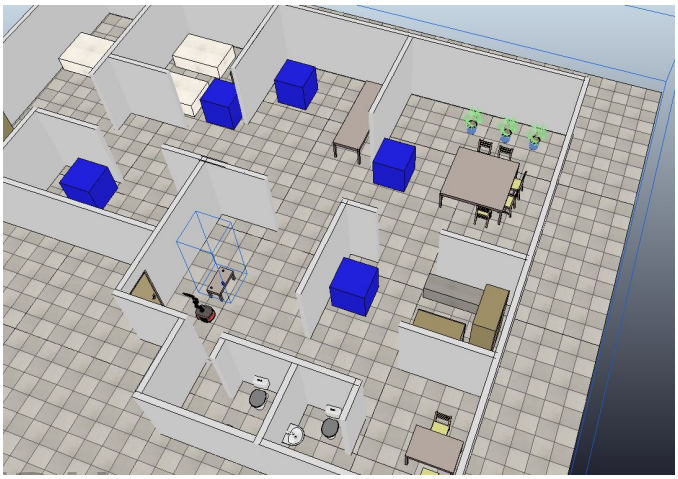

Fig. 1.Simulacion de VREP.

Se realizo la conexion de Matlab con VREP para poder realizar los calculos necesarios en matlab a partir de diferentes funciones, y lograr obtener informacion de la simulacion en VREP; y realizar las acciones a partid de la informacion procesada para poder mover el brazo UR3 y el robot Pioneer en VREP.

II. Diseño

A. Procesamiento de Imagenes

Se utilizo el Vision Sensor de vrep para poder realizar capturas dentro de la simulacion. Estas capturas fueron utilizadas con diferentes fines. Un sensor perimite tomar una fotografia a color para poder aplicar a la misma el thresholding que permite obtener solamente el color de las mesas (azul) sobre las cuales se dejaran los objetos.

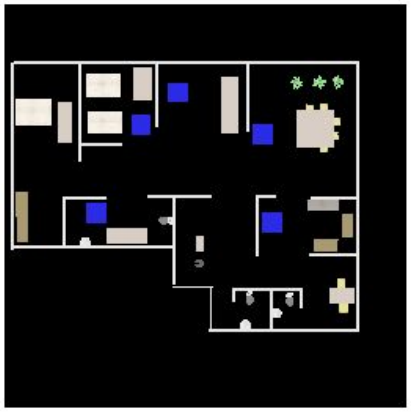

Fig. 2. Imagen a color obtenida con Vision Sensor.

El threshholding se realizo por medio de la aplicacion de Matlab Color Threshold, que permite realizar ajustes a una imagen restringiendo los colores visibles, y esta luego se exporto como una funcion para obtener solamente las figuras de las mesas. Luego se realizo un Control Point para realizar una correspondencia entre una imagen de checkerboard con cada cuadro de una longitud de 5 pixeles y una imagen de la simulacion de VREP tomada fuera de la aplicacion, ya que se requeria una mejor resolucion para esta correspondencia de puntos de control.

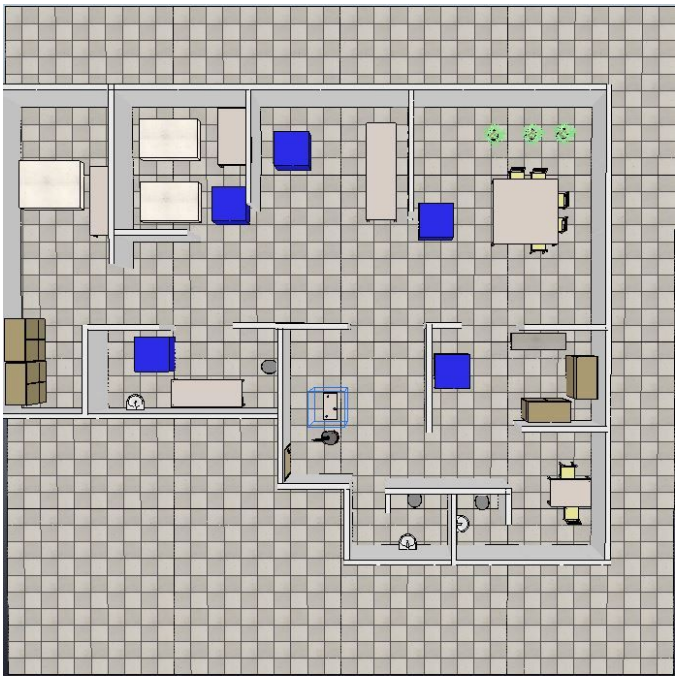

Fig. 3. Imagen para Control Point.

Luego se encontraron los centroides con ayuda de la funcion regionprops, que se encontraban mapeados en la simulacion de vrep, con coordenadas diferentes dado que en vrep la coordenada 0,0 no es una esquina.

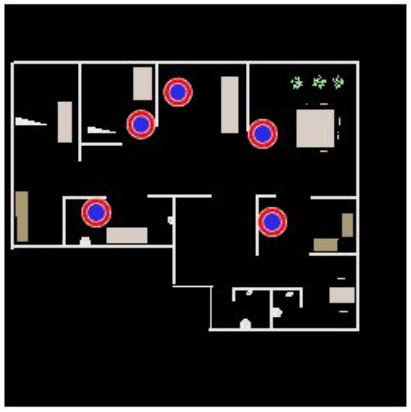

Fig. 4. Metas encontradas (circulos de centroide) luego de aplicar region props.

Para crear el mapa a recorrer se utilizo un vision sensor que tomara una fotografia en blanco y negro, a la cual se le aplico un threshold para obtener un occupancy grid que contuviera los obstaculos.

De igual manera se realizo el mismo procedimiento para obtener las posiciones de los objetos a recoger por el brazo.

B. Trayectorias

Las trayectorias se crearon por medio del algoritmo de navegacion D* (D estrella), y se utilizo ya que esta provee caminos de distancias cortas y facilita la replanificacion incremental. para este algoritmo se utilizo la imagen del mapa creada y un inflate de 8, el cual permite que los obstaculos detectados tengan bordes más grandes, lo que permite que en al realizar el recorrido no se tope con los obstaculos.

Se realizaron 5 metas, con 10 recorridos; un recorrido para llegar a la meta y un recorrido para regresar a la posicion en la que se encuentran los objetos.

A la funcion de d estrella se ingresa el mapa del que se desea realizar el recorrido, la posicion inicial y la posicion a la que se desea llegar. Estas posiciones se obtuvieron por medio de encontrar la pendiente dado que se tiene un mapa de 20x20 (de -10 a 10) y la resolucion que se tiene es de 256. Se obtiene la ecuación 12.75x + 128.5, con la cual se transforman las posiciones en vrep a posiciones en el mapa que se va a crear ya que es necesario realizar la conversion de distancia en el intervalo de la resolucion (1 a 256).

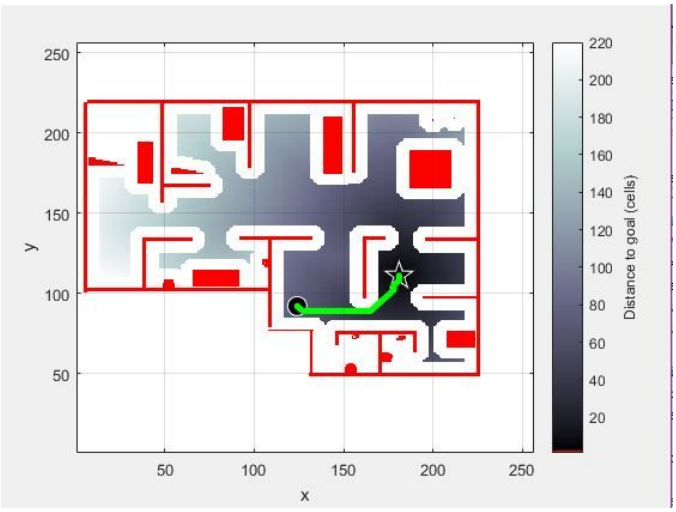

Fig.5 . Trayectoria generada #1.

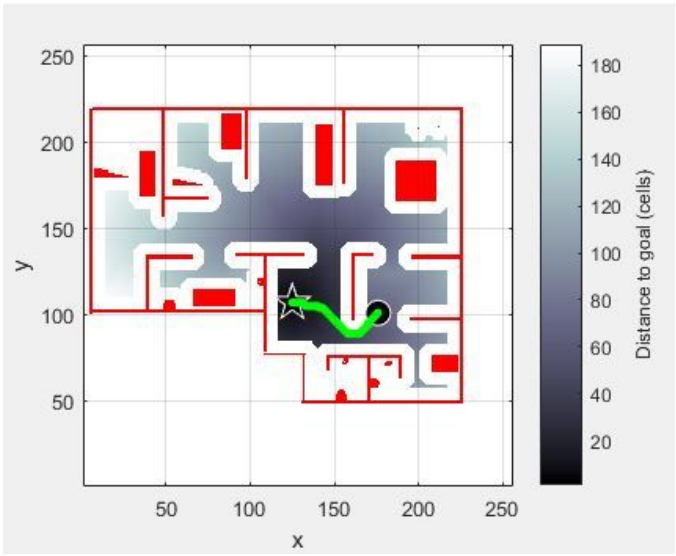

Fig.6 . Trayectoria generada #2.

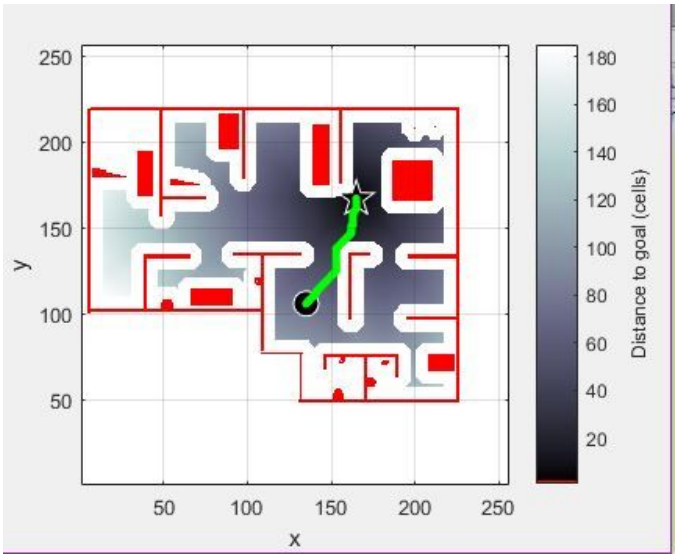

Fig.7 . Trayectoria generada #3.

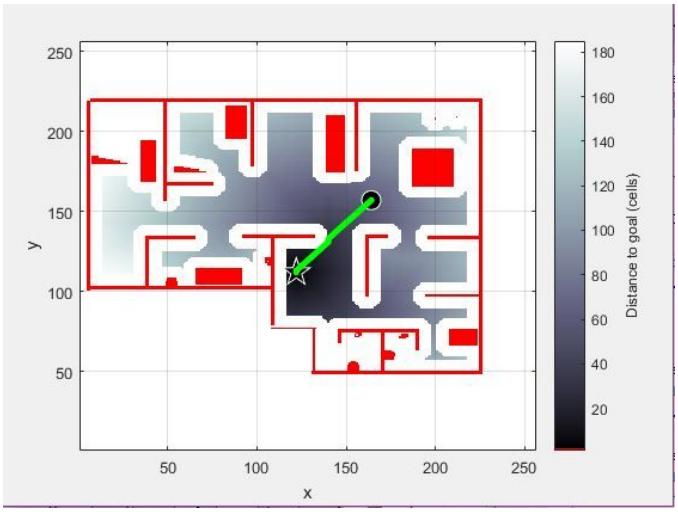

Fig.8 . Trayectoria generada #4.

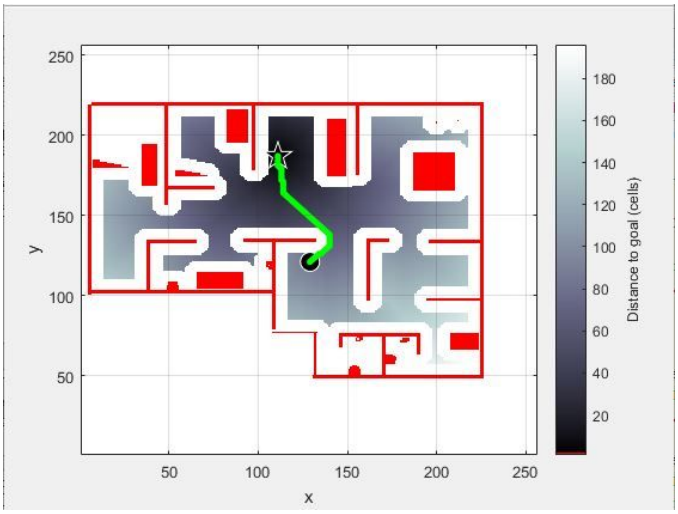

Fig.9 . Trayectoria generada #5.

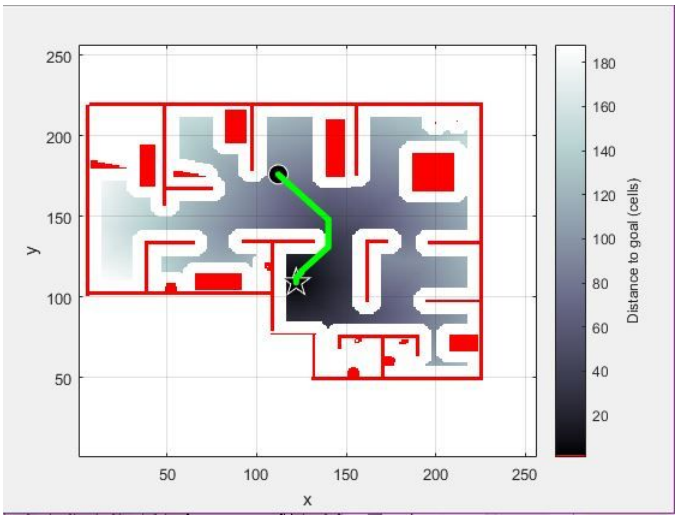

Fig.10 . Trayectoria generada #6.

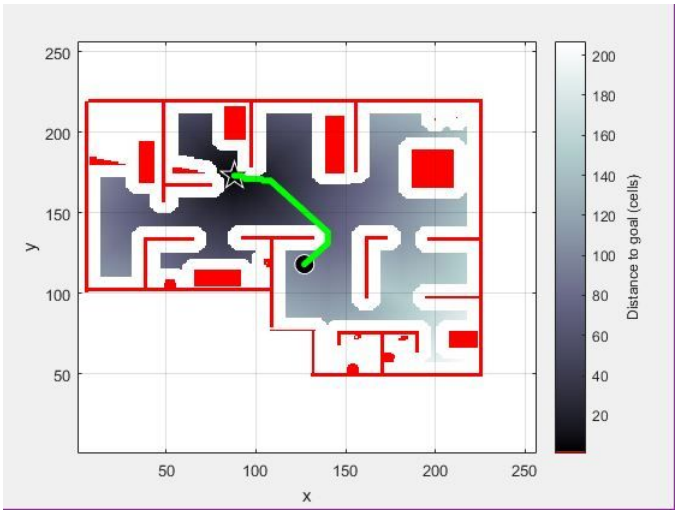

Fig.11 . Trayectoria generada #7.

C. *UR3*

Para mover el manipulador UR3 se definio el brazo robotico dentro de matlab, asi como sus juntas. Primero se definio una pose inicial para el robot, una pose intermedia y una pose final. Estas permiten obtener una trayectoria que seguira el manipulador por medio de la funcion mtraj, luedo de haber obtenido la cinematica inversa de las poses. la cinematica inversa se obtuvo con una funcion creada en matlab, utilizando el algoritmo de Levenberg-marquardt, en donde el Jacobiano es igual $J^\top \left(JJ^\top + \lambda^2 I\right)^{-1}$ .En este metodo numerico iterativo se tiene como argumento una pose deseada del efector final, una configuracion inicial y un parametro que determina la expresion a sustituir por la inversa del jacobiano, y regresa una configuracion deseada q; se utiliza tambien la cinematica directa dentro del mismo, asi como error de orientacion utilizando cuaterniones.

Para la cinematica directa se genera una matriz de transformacion homogenea para los parametros Denavit-Hartenberg, la cinematica directa se calcula a partir de la misma. La cinematica directa del manipulador se retorna en la funcion que tiene como parametro de entrada un vector columna de configuracion q.

Resumiendo, con una pose deseada para el efector final, la cinematica directa del manipulador, se puede obtener de forma numerica la cinematica diferencial para ejecutar el algoritmo iterativo $\mathbf{q}_{k+1} = \mathbf{q}_k + J^{-1}(\mathbf{q}_k)\mathbf{e}_k,$ a partir de una configuracion inicial q0, se emplea la combinacion de error de posicion y orientacion, este ultimo correspondiendo a la parte vectorial del cuaternion unitario de error $\Delta\mathcal{E}(\mathbf{q}_k) = \mathcal{E}_d * \mathcal{E}^{-1}(\mathbf{q}_k) = \begin{bmatrix}\Delta\eta(\mathbf{q}_k)\\ \Delta\boldsymbol{\epsilon}(\mathbf{q}_k)\end{bmatrix}$ que tiende a cero conforme el cuaternion de orientacion actual del efector final se parece al cuaternion de orientacion deseada.

El jacobiano del manipulador se calcula para cierta configuracion q, en forma de vector columna.

Se definio la matriz de Denavit Hartemberg con ayuda de matlab, con la funcion mdl_ur3.

```
UR3 [Universal Robotics]:: 6 axis, RRRRRR, stdDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|     0.1519|          0|     1.5708|          0|
|  2|         q2|          0|   -0.24365|          0|          0|
|  3|         q3|          0|   -0.21325|          0|          0|
|  4|         q4|    0.11235|          0|     1.5708|          0|
|  5|         q5|    0.08535|          0|    -1.5708|          0|
|  6|         q6|     0.0819|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

Fig. 12. Matriz DH del manipulador UR3.

D. *Pioneer P3-DX*

Para el robot Pioneer se implemento el algoritmo de control punto a punto para un robot diferencial.

$$\dot{\phi}_r = \frac{v + \omega\ell}{r}, \qquad \dot{\phi}_\ell = \frac{v - \omega\ell}{r},$$

en donde

$$v = K_p e_p, \qquad \omega = K_o e_o,$$

$$e_p = \left\| \begin{bmatrix} x_g \\ y_g \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \right\|, \qquad e_o = \arctan 2\left(\frac{\sin(\theta_g - \theta)}{\cos(\theta_g - \theta)}\right), \qquad \theta_g = \arctan 2\left(\frac{y_g - y}{x_g - x}\right).$$

Fig. 13. Algoritmo de Control Punto a Punto.

Se utilizaron los valores de 0.4 y 0.9 para las constantes Kp y Ko respectivamente. Se implemento una condicion de error con la cual se detiene el Pioneer cuando se encuentra cerca de la posicion a la que se desea llegar.

III. Discusión de Resultados

Al realizar la simulacion se obtuvieron diferentes errores. Uno de los más notables fue que en la simulacion no siempre se lograba agarrar el objeto, y no se cambiaba el offset propuesto. Esto represento una dificultd ya que muchas veces se podia agarrar el primer objeto pero el segundo no se podia, o viceversa; asi como tambien el objeto se quedaba atrapado en la garra en algunas iteracions, por lo que el objeto no llegaba a su posicion final deseada.

Tambien se observo que al realizar las grabaciones en VREP por medio de la herramienta propia del programa, resultaba bastante dificil observar el correcto funcionamiento de la simulacion, probablemente por el consumo de recursos extra que se tienen al grabar.

Otro de los errores que se obtuvieron fue que al realizar el Control Point con una imagen obtenida directamente del Vision Sensor se tenian resultados en los cuales no se podia observar totalmente el mapa (fig 14). Al realizar el mismo proceso por medio de una fotografia tomada desde fuera de VREP se obtuvieron distintos valores de centroides, como por ejemplo se sabe que el origen de vrep se encuentra a la mitad de la escena, y al obtener los centroides con la funcion regionprops estos se encuentran respecto a un origen que no esta en la misma posicion que la simulacion. para la mesa 2 se obtuvieron los resultado del centroide en 8,7, y sabiendo que dentro de vrep la mesa tenia la posiion de -3.2,4.02 se tenia que el origen se encontraba corrido 11.2 en x y 11.02 en y, y con la mesa 3 se obtenia que estaba corrido 10.37 en x y 10.63 en y, por lo que se encontraron diferentes valores para cada meta. Para minimizar este error se realizo el mismo proceso pero solo con 1 mesa a la vez (fig 15), y no con todas (fig 16), pero se obtuvieron resultados muy similares; y esto afecta el resultado de colocar los objeto en el lugar correcto.

Referencias

Corke, P. (2017). Robotics, Vision and Control Fundamental Algorithms in MATLAB. 2nd ed. Basel, Switzerland: Springer International Publishing.

Craig, J. (2005). Introduction to Robotics Mechanics and Control. 3rd ed. Upper Saddle River, NJ: Pearson Prentice Hall.

IV. Anexos

Fig. 14. Imagen tomada desde vrep.

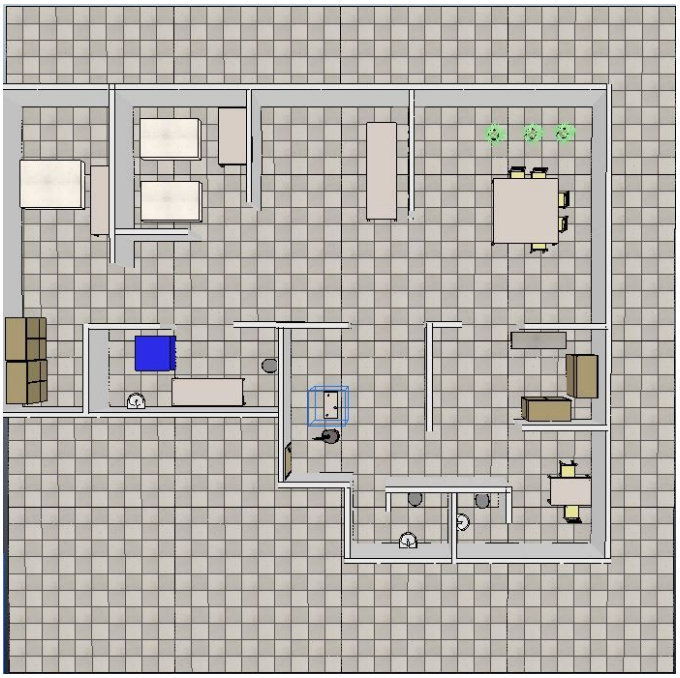

Fig. 15. Una meta.

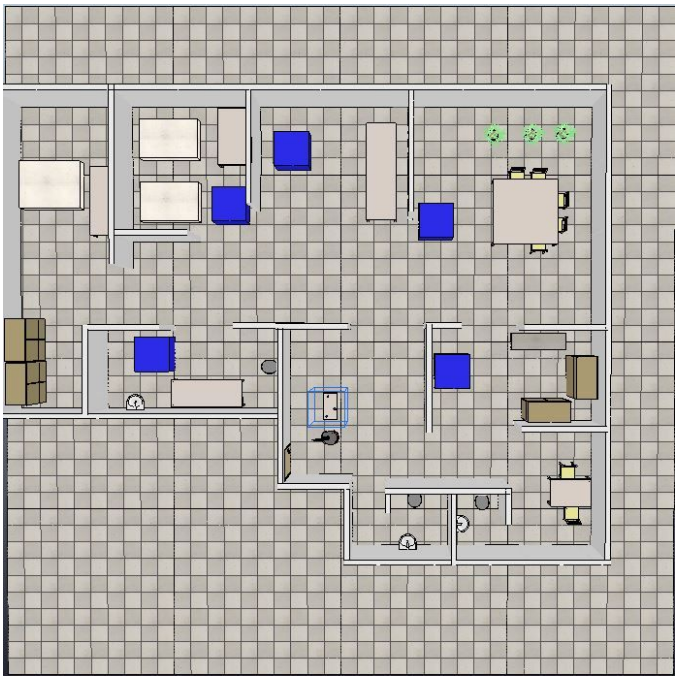

Fig. 16. Todas las metas.

Link Video:

Funcionamiento

https://youtu.be/Z7vRJfMRlWU

Trayectorias

https://youtu.be/Oq5NXMMnBdM