Algoritmo Autónomo para la Navegación de Turtlebots

Grado en Ingeniería Robótica

Proyecto Final Robots Móviles

Autores:
Thomas Cottone
Manuel Prieto Burgos
Martin Lucero Beiroa
Eladio Vicedo Ruiz

Enero 2023

Contenido

1. Introducción

Para este proyecto, se ha determinado la concepción de un algoritmo destinado a los *Turtlebots* del laboratorio, con el propósito de que sean capaces de moverse de manera autónoma sobre la planta baja de la Escuela Politécnica Superior III.

Para llevar a cabo esta tarea, se ha diseñado inicialmente el algoritmo, el cual se ha configurado para permitir que un robot simulado se desplace hacia *keypoints* previamente seleccionados, evitando obstáculos que puedan interponerse en su camino y trazando una ruta óptima. Posteriormente, se ha procedido a realizar el mapeo completo de la planta baja de la EPS III. Además, se ha intentado perfeccionar el algoritmo mediante la implementación de reconocimiento por voz, y se ha tratado de ponerlo a prueba con robots físicos.

2. Estado del arte

Son múltiples los trabajos y proyectos que se han realizado donde el movimiento autónomo de un robot es la parte fundamental y habilitadora del resto del proyecto, tal como es nuestro caso.

El trabajo de fin de grado “Localización y control de un robot autónomo” [1] de la estudiante de ingeniería robótica Carmen Ballester Bernabéu es un buen ejemplo de un trabajo donde se utiliza el movimiento autónomo de un robot basado en grafos y nodos, tal y como veremos más adelante que se va a realizar nuestro propio trabajo.

Para el reconocimiento de voz también se han realizado múltiples trabajos donde un robot recibe órdenes por voz para que luego realicen la tarea que se le ha ordenado completar. En el caso del trabajo de fin de grado “Reconocimiento de voz con el robot Félix” [2] realizado por Mario Parreño Lara, graduado de ingeniería informática se observa el uso del reconocimiento de voz en robots.

3. Mapeado

El mapeado de la planta baja de la EPS III, es decir, el entorno sobre el que operará el robot se ha realizado de manera similar a lo realizado en la práctica 2 de esta misma asignatura moviendo el robot por todas las zonas por las que nos fuese posible.

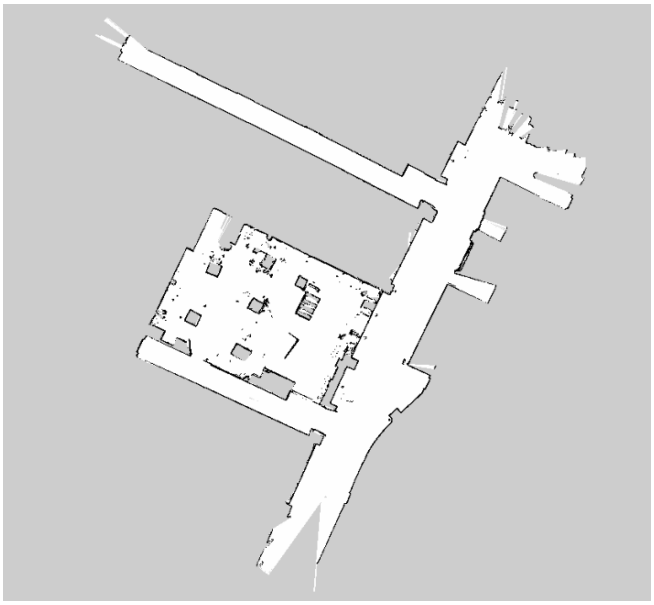
El primer paso fue realizar la conexión con el Turtlebot desde un ordenador del aula que fuese capaz de conectarse a la red wifi. Una vez habíamos establecido conexión lo único que teníamos que hacer era mover el robot por el entorno y mapear la mayor cantidad de terreno posible mientras se podía visualizar el resultado en RViz.

El mapeo del laboratorio de robótica fue satisfactorio y no surgió ningún problema, movimos el robot por el pasillo fuera de la clase y lo llevamos hasta los límites que la conexión WiFi del laboratorio nos permitiese.

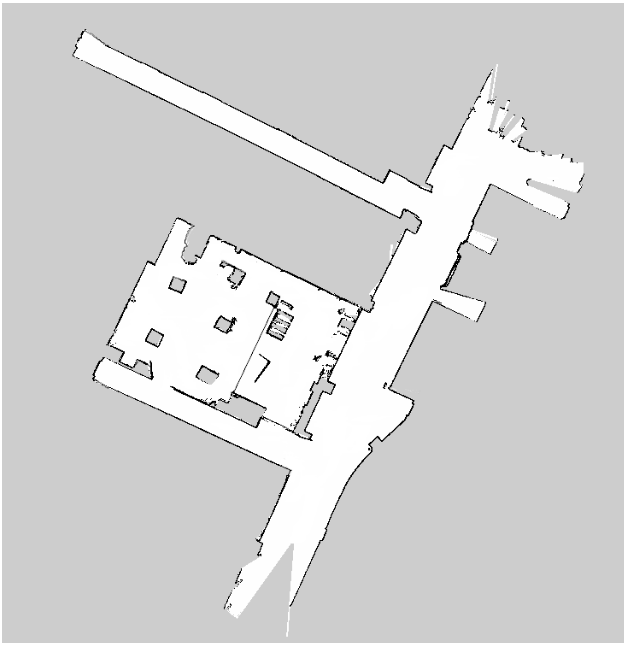
En esta segunda parte nos surgió un problema ya que cerca de una de las entradas al edificio el Turtlebot perdió la conexión en el punto mostrado en la siguiente imagen:

Imagen 1, punto de pérdida de conexión con el Turtlebot

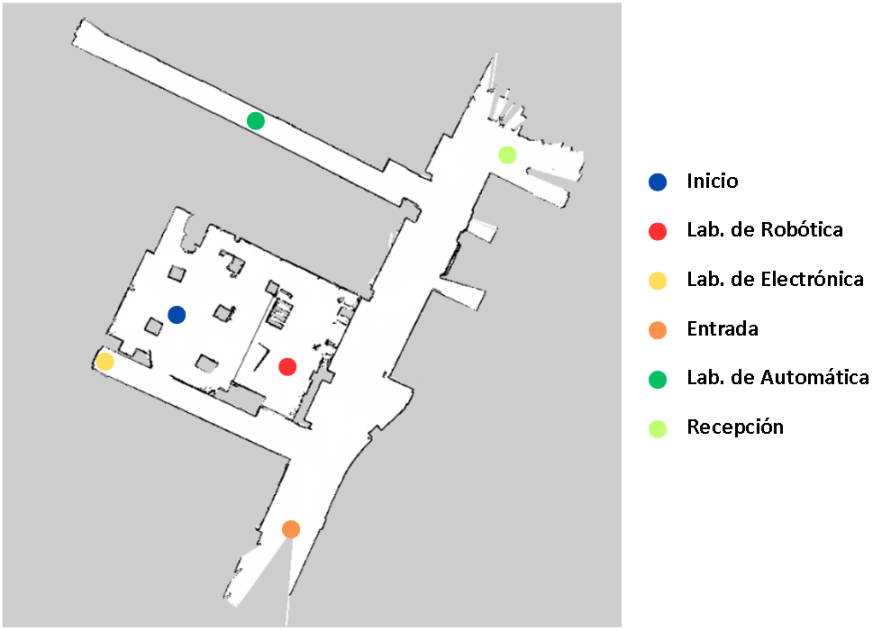
Esto nos llevó a tener que volver a realizar el mapeo, siendo este segundo y último intento el que nos dejaría con el siguiente mapa:



Nos dimos cuenta de que la limpieza y claridad del mapa se podía mejorar ya que, debido a posibles imperfecciones del sensor del robot, o a obstáculos como por ejemplo los compañeros que se encontraban en el laboratorio en ese momento había ruido en el mapa, el cual eliminamos dando como resultado el siguiente mapa:



Con el mapa terminado el siguiente paso era decidir los keypoints a los que el robot podría moverse. No pudimos acceder a ningún aula dentro del alcance del wifi con el robot, así que decimos establecer los siguientes keypoints en el mapa:



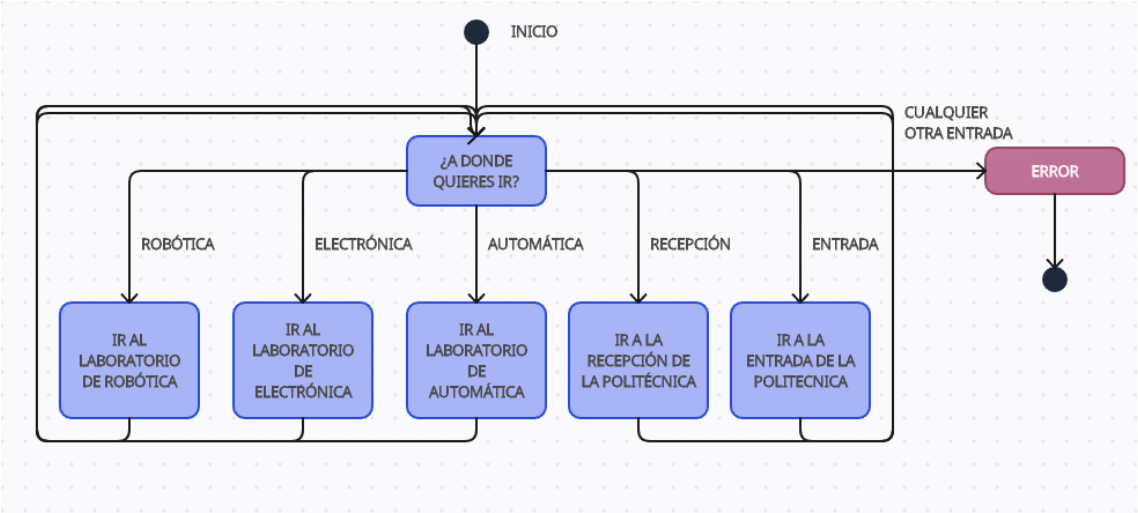
Esto nos proporcionará el resultado final del mapa sobre el que vamos a trabajar.

4. Desarrollo del Algoritmo

El algoritmo que se ha utilizado en las simulaciones se ha creado a partir del código creado para la realización de la práctica 3.

Este código constaba de una máquina de estados que movía, aleatoriamente, al Turtlebot por el mapa hasta que visualizará un objeto rojo. Al verlo, el robot volvía al punto marcado como BASE. Esta última función (volver a BASE) fue de interés a la hora de plantear el algoritmo. Se ha modificado esta función cambiando los puntos a los que se moverá el robot por los que se han presentado en la imagen 4.

Este es diagrama de estados del algoritmo final:



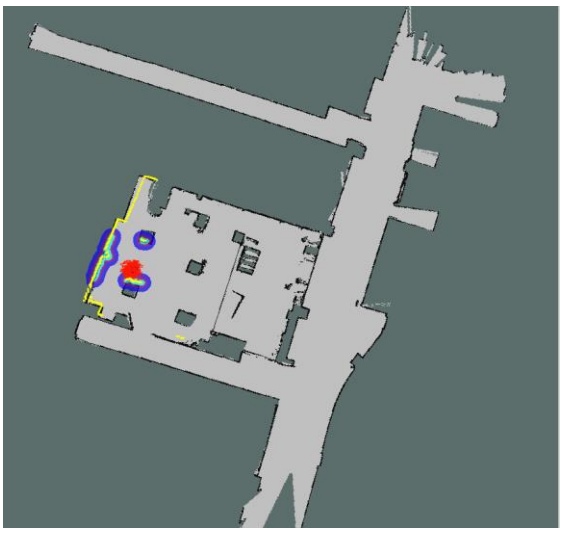
Una vez comenzada la ejecución, se pregunta al usuario por pantalla a donde quiere ir, las opciones son: ROBOTICA, ELECTRONICA, AUTOMATICA, RECEPCION y ENTRADA (sin acentos). Una vez que el usuario responda por teclado (en principio debería ser por voz, pero no se ha conseguido que funcione), manda al robot al *keypoint* correspondiente. Una vez el robot haya llegado, se volverá a preguntar a donde se quiere ir. Para terminar el programa y salir del bucle, el usuario deberá escribir cualquier palabra que no este dentro de las indicadas.

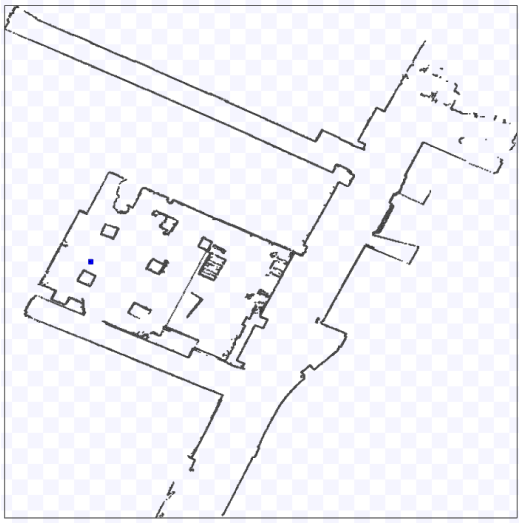
```
[INFO] [1705149359.719280, 79.500000]: State machine starting in initial state '
Preguntar' with userdata:
        []
Adonde quiere ir? ROBOTICA
[INFO] [1705149364.952394, 84.700000]: State machine transitioning 'Preguntar':'
ROB'-->'GoToROB'
[INFO] [1705149407.051885, 114.200000]: State machine transitioning 'GoToROB':'t
ROB'-->'Preguntar'
Adonde quiere ir? FIN
[INFO] [1705149413.853237, 119.500000]: State machine terminating 'Preguntar':'E
RROR':'stop'
```

5. Simulación

Para hacer las pruebas en simulación se ha utilizado el mapa creado en el apartado 3. Mapeado, tanto en *RViz* como en *Stage*. Se han modificado los archivos *.launch, .world* y *.yaml*. Estos dos últimos se encuentran en las carpetas *maps* y *worlds*, dentro de *stage_config.* Estos archivos se han modificado ya que había que redimensionar el tamaño del mapa para que concordase con el real. Se han cambiado los valores de tamaño de [10.0, 10.0, 0.0] a [33.0, 33.0, 2.0]. Además, se ha tenido que modificar la posición inicial manualmente ya que en un principio el robot estaba bien posicionado en *Stage,* pero fuera del mapa en *Rviz*.

Ha habido un problema a la hora de poner las posiciones, tanto inicial como *keypoints*. Este problema consiste en que los mapas de *RViz* y *Stage* tienen ubicado el [0,0] del mapa de manera diferente. *RViz* lo tiene ubicado en la esquina inferior izquierda mientras que *Stage* en el centro del mapa. Para solventar este problema se ha hecho uso de la herramienta "*2D Pose Estimate*" de RViz, que permite posicionar directamente al robot en la posición deseada. Una vez posicionado, se mira la posición actual utilizando el comando "*rostopic echo amcl_pose*".





Una vez este todo configurado, solo queda ejecutarlo, para ello hay que abrir dos terminales, en la primera se ejecutará el *.launch*, mediante el comando "*roslaunch navigation_stage mi_navigation.launch*". En la segunda terminal se entrará en la carpeta *src*, dentro de *navigation_stage*, y se ejecutará el código.

6. Funcionamiento del Robot Real

Una vez comprobado el correcto funcionamiento del código en simulación, se procede a probar el algoritmo en un robot real de laboratorio, más exactamente en un turtlebot.

Primero que todo, se debe realizar una conexión vía wifi con el robot. Para ello, se utiliza unos de los ordenadores del laboratorio. Para conectarse al robot por terminal, se utiliza el comando *ssh turtlebot@ip_del_robot*

Una vez conectado con el robot, se debe arrancar la base y el láser del mismo. Para esto se necesitan dos terminales del robot. En cada una de ellas se debe lanzar una orden distinta, y no cerrarlas, o se perderá la conexión. En una terminal se debe lanzar *roslaunch turtlebot_bringup minimal.launch* y en otra *roslaunch turtlebot_bringup hokuyo_ust10lx.launch*

Para trabajar en el ordenador se deben utilizar otros terminales. Para comprobar si la conexión se ha realizado correctamente se puede escribir *rostopic list* en otra terminal y deberán aparecer los topics del ROS del robot.

Una vez hecho esto, se debe copiar el mapa que se utiliza en la simulación en el robot. Para ello se utiliza el comando *cp*. El mapa se guarda en la raíz del robot.

El último paso antes de poder lanzar el algoritmo en el robot sería abrir una aplicación de visualización para poder controlar y analizar el comportamiento del robot durante la ejecución. El programa que se utiliza es RViz, y se lanza en un terminal con la orden *rosrun rviz rviz*. Para cargar el mapa en RViz se deben escribir en una terminal del robot las siguientes órdenes:

export TURTLEBOT_3D_SENSOR=astra

roslaunch turtlebot_navigation amcl_demo.launch map_file:=/home/turtlebot/nombre_del_mapa.yaml

Con esto ya aparecerá el mapa en el visualizador. Antes de lanzar el algoritmo se debe localizar el robot en el mapa. Para ello se utiliza la herramienta *2D pose estimate*, que permite marcar la posición y orientación del robot en el mapa. Al realizar esta acción se presentó un problema. Este problema impedía indicar la posición del robot utilizando la herramienta. Para solucionarlo se cambió el *fixed frame* de RViz de *map* a *odom*, y de nuevo a *map*.

Con toda esta configuración, ya estaría todo listo para lanzar el algoritmo en el robot. Al lanzarlo se obtiene un error sobre una dependencia que se debe instalar en el ordenador de

laboratorio. Puesto que el usuario de los alumnos no tiene derechos para instalar nada en estos ordenadores, no se pudo realizar la prueba de esta manera.

Debido a este problema, se optó por intentar la conexión al robot con uno de los ordenadores personales de los alumnos. Se siguieron todos los pasos de la misma manera, pero no se consiguió la conexión aun así, puesto que el ordenador utilizado no tiene Linux nativo, sino en máquina virtual.

Por estos problemas no se pudo llegar a probar el algoritmo en un robot real y se muestra su funcionamiento en simulación solamente.

7. GitHub

Se ha creado un repositorio público en *GitHub* en el que se han incluido todos los ficheros necesarios para la ejecución del proyecto. Además, se ha creado un *README.md* en el que se han explicado muy detalladamente los pasos (alguien con poca experiencia en programación podría seguir los pasos) para que el usuario cree un directorio y pueda utilizarlo desde su ordenador. Enlace al repositorio en cuestión: https://github.com/tc50ua/navegacionautonoma

8. Bibliografía

[1] https://rua.ua.es/dspace/bitstream/10045/115947/1/Localizacion_y_Control_de_robots_autonomos_Ballester_Bernabeu_Carmen.pdf

[2] https://riunet.upv.es/bitstream/handle/10251/87015/PARRE%C3%91O%20-%20Reconocimiento%20de%20voz%20con%20el%20robot%20F%C3%A9lix.pdf?sequence=1

Repositorio en GitHub: https://github.com/tc50ua/navegacionautonoma

DEMO del proyecto: https://drive.google.com/file/d/1eLg4ecjoHnvMDJPgnpjWYGwZV419vazV/view?usp=sharing