UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

Firmware para robot autónomo utilizando FreeRTOS

Autor:
Alexis Martin Pojomovsky

Director:
Dr. Pablo De Cristóforis (FCEyN-UBA)

Jurados:
Esp. Ing. Diego Fernández (FI-UBA)
Esp. Ing. Edgardo Comas (CITEDEF/UTN-FRBA)
Esp. Ing. Gerardo Puga (UNLP)

Este trabajo fue realizado en las Ciudad Autónoma de Buenos Aires, entre enero de 2018 y diciembre de 2019.

Resumen

En el presente trabajo se describe el desarrollo de un robot móvil destinado a utilizarse por estudiantes universitarios en el aprendizaje de tópicos de robótica, tales como odometría, localización y navegación.

Esta propuesta se ajusta a dos importantes restricciones: disponibilidad de componentes en el mercado local y precio competitivo respecto a productos comerciales de similares características.

Se incluye además, un repositorio público en Github desde el que se puede acceder al software de ejemplo y documentación necesaria para comenzar a utilizar la plataforma.

Agradecimientos

Índice general

Índice de figuras

Índice de Tablas

Dedicado a... [OPCIONAL]

Capítulo 1

Introducción general

En este capítulo se introduce el campo de estudio de la robótica, contraste entre robótica de manipuladores y móvil, así como la importancia de una planta de pruebas física como motivación para la realización de este trabajo. Asimismo, se presentan los objetivos y el alcance del presente proyecto.

1.1. Robótica móvil

La robótica de manipuladores, también llamados brazos robóticos, se han ganado su puesto como ciudadanos de primera clase en la industria de la manufactura. Difícilmente podríamos al día de hoy, imaginarnos una planta de fabricación en serie que no disponga de estos dispositivos para la realización de tareas repetitivas y de alta precisión.

En la industria electrónica, por citar un ejemplo, los manipuladores son capaces de colocar componentes de montaje superfical con una precisión y velocidad por lejos sobre-humana, haciendo posible la elaboración de teléfonos celulares, computadoras portátiles, etc. Sin embargo, y a pesar de su innegable éxito, estos robots sufren de una desventaja particular: la falta de movilidad. En contraste con un manipulador fijo posee un rango de movimiento limitado que depende del sitio en que el mismo se encuentre instalado, un robot móvil es capaz de moverse a través de la planta, permitiendo el aprovechamiento de sus facultades donde sea que estas sean precisadas. Ante estas limitaciones, en los años noventa surgen los robots móviles.

Una definición correcta de robot móvil plantea la capacidad de movimiento sobre entornos no estructurados, de los que se posee un conocimiento incierto, mediante la interpretación de la información suministrada a través de sus sensores y del estado actual del vehículo.

El principal problema a resolver en un robot móvil es el de generar trayectorias y guiar su movimiento según éstas, en base a la información proveniente de los sensores externos, permitiendo al vehículo desplazarse entre dos puntos cualesquiera del ambiente de trabajo de manera segura, sin colisiones. Esto exige diseñar sistemas de control de trayectorias (posición, dirección, velocidad) en diversos niveles jerárquicos, de manera que el procesamiento de la información proveniente de los sensores externos asegure la mayor autonomía posible.

Los robots móviles operando en grandes ambientes no estructurados deben enfrentarse a significativas incertidumbres en la posición e identificación de objetos. En efecto, la incertidumbre es tal que, trasladarse desde un punto A hasta un punto B es una actividad arriesgada para un robot móvil, una actividad relativamente trivial para un manipulador industrial. En compensación por tener que enfrentarse con más incertidumbres del entorno, no se espera que un robot móvil siga trayectorias o alcance su destino final con el mismo nivel de precisión que se espera de un manipulador industrial (en el orden de las centésimas de milímetro).

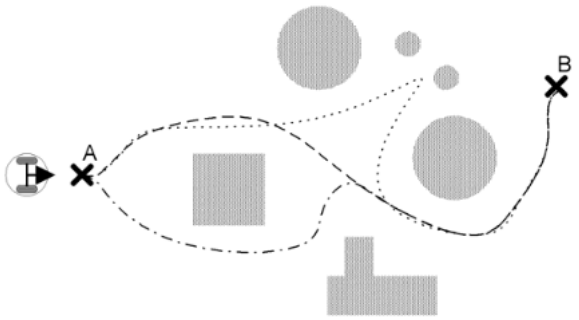

FIGURA 1.1: Algunas de las posibles trayectorias que podría adoptar el robot móvil.

1.2. Motivación

El estudio de la robótica en las universidades de la región se encuentra mayormente avocada a la robótica de manipuladores. Esto tiene sentido desde un punto de vista de oferta y demanda en la industria local, sin embargo, el mercado internacional esta viviendo una fuerte demanda de profesionales capaces de entender y aplicar técnicas de robótica móvil.

Teniendo en cuenta que hasta hace solo un puñado de años atrás, los dispositivos requeridos para llevar a cabo estas tareas hacían que el estudio, desarrollo y aplicación de tareas de robótica móvil sean extremadamente costosos, aún a pequeña escala, la realidad actual es diferente. Hoy, y gracias a la masificación de ciertas tecnologías, es posible armarse de toda una gama de sensores, actuadores y sistemas capaces de procesar el flujo de datos requerido para estas aplicaciones con muy poco dinero, en muchos casos reutilizando componentes originalmente destinados a otros propósitos.

Aún con la importante reducción de precios vivida en los últimos años, las plantas de prueba ofrecidas por empresas internacionales para fines didácticos continúan siendo prohibitivas para muchas instituciones educativas, así como también para la mayoría de los estudiantes. Es por esto que el presente trabajo se desarrolla en torno a la propuesta de planta de pruebas para el estudio y aplicación de técnicas de robótica móvil. Esta propuesta se centra en dos restricciones importantes, bajo costo y disponibilidad en el mercado local.

1.3. Descripción de tecnologías

Blabla.

1.4. Objetivos y alcance

Blabla.

Capítulo 2

Ensayos y Resultados

En este capítulo se detallan las pruebas realizadas tanto de manera individual, para comprobar que tanto los componentes de *hardware* como de *software* funcionen como se espera, así como las pruebas de integración realizadas entre los componentes individuales y el *framework* ROS.

2.1. Pruebas funcionales del hardware

Aquí se describen las pruebas que se realizaron individualmente a cada uno de los componentes del robot.

2.1.1. Sensor Microsoft Kinect 360

El Kinect es un dispositivo complejo que posee varios componentes internos, todos con diferente identificador USB, que se conectan mediante un cable único. Este cable posee un conector propietario especial que implementa, además de los hilos del estándar USB 2.0, uno de alimentación a 12 VCC y su respectiva tierra. Por cuestiones de disponibilidad y precio, se procedió a remover manualmente este conector y añadir manualmente un conector estándar USB macho y un plug de alimentación a 12 VCC.

Para testear adecuadamente el funcionamiento del sensor con la adaptación casera, se procedió a conectar el cable USB a un PC con Linux y la alimentación a una fuente externa de 12 VCC. Aquí, se utilizó el comando lsusb, disponible como parte del paquete usb-utils en Ubuntu, y se procedió a listar los dispositivos USB advertidos por el sistema operativo:

Una vez corroborado esto, se procedió a verificar que la nube de puntos se esté generando correctamente. Para esto se utilizaron las siguientes herramientas y paquetes de software:

- libfreenect: librería de código abierto para el sensor Kinect.
- rqt_image_viewer: herramienta integrada de ROS para visualizar imágenes RGB y nube de puntos.

2.1.2. Unidad de medición inercial InvenSense MPU6050

Para corroborar el correcto funcionamiento de la IMU se procedió a conectarlo a un Arduino MEGA2560 y se utilizó la librería Adafruit_MPU6050, la cual se encuentra activamente mantenida por la compañía Adafruit.

Se procedió así, al envío de información RAW via puerto serie y se procedió a graficar, individualmente, cada uno de los componentes de los vectores de aceleración y giro proveídos por el sensor.

2.2. Pruebas funcionales del software

Aquí se describen las pruebas que se realizaron individualmente a cada uno de los módulos de software que componen el sistema.

2.2.1. Envío y recepción de datos entre el robot y microcontrolador

Al tratarse esta de una base móvil comercial, y gracias a la existencia de un documento oficial publicado por el fabricante con información detallada sobre el protocolo de comunicación del robot, se dispone de la gama completa de comandos para manipular los actuadores, así como para leer los sensores disponibles.

Mediante las pruebas descriptas a continuación, se procedió a verificar el subconjunto de periféricos que fueron implementados en el microcontrolador que actúa como interfaz entre el robot y el sistema ROS, corroborando que los mismos funcionen tal y como se describen en el documento mencionado anteriormente.

Para el envío de órdenes desde el microcontrolador hacia el robot, se analizó la respuesta por parte del mismo ante una serie definida de comandos, ante los cuales se corroboró de manera manual que la respuesta sea la esperada.

2.2.2. Publicación y recepción de mensajes ROS en el microcontrolador

El proceso en el que se probó la correcta publicación de mensajes desde el microcontrolador hacia ROS, involucró la utilización de las siguientes herramientas integradas del mismo:

- rosserial_python serial_node
- rostopic list
- rostopic echo
- rostopic hz
- RViz

Para verificar la existencia de los tópicos anunciados por el microcontrolador fué necesario, primeramente, ejecutar el nodo serial_node, el cual se encarga de la serialización y des-serialización de los mensajes transmitidos mediante UART o Ethernet, actuando de puente entre el protocolo intermedio de rosserial y ROS.

TABLA 2.1: Sub-conjunto de comandos utilizados para testear el correcto funcionamiento del la base móvil

Comando	Descripción	Verificación
START	Se inicializa el protocolo OpenInterface	El robot deja de ignorar comandos
STOP	Se finaliza el protocolo OpenInterface	El robot empieza a ignorar comandos
SAFE	El robot pasa a modo seguro	El robot deja de moverse si es levantado del suelo
FULL	El robot pasa a modo full	El robot no deja de moverse a menos que reciba explícitamente un comando STOP
DRIVE DIRECT	El robot recibe comandos de velocidadpara cada rueda	El robot empieza a moverse inmediatamente después de recibir el comando
SENSORS	Se solicita la lectura del estado actualde un sensor	El robot responde con el mensaje correspondiente al sensor solicitado

Luego de haber ejecutado el nodo rosserial en la computadora host, fué posible verificar la existencia de los tópicos invocando el comando 'rostopic list', el cual despliega una lista de los tópicos que se encuentran anunciados.

Una vez conocida la lista de tópicos, es posible reproducirlos en la consola mediante el comando 'rostopic echo' así también, es posible verificar su frecuencia de publicación mediante el comando 'rostopic hz'.

Luego de verificar la existencia de los mensajes en cada tópico, es posible graficarlos utilizando la herramienta RViz. Esto resulta especialmente útil para mensajes compuestos por varios campos, como los del tipo IMU.

2.3. Pruebas de integración

Estas pruebas integran las tareas de alto nivel que se pretende puedan ser utilizadas por los usuarios que adopten la plataforma. Estas involucran la interacción entre todos los componentes del sistema en simultáneo, así como su interacción con el sistema ROS.

Las pruebas se encuentran ordenadas de manera tal que las primeras formen parte de los requisitos para las siguientes y así respectivamente, concluyendo con la prueba de integración

2.3.1. Estimador de pose a partir de IMU

Mediante el nodo imu_complementary_filter, integrado como parte del paquete imu_tools[1], se procede a procesar la información *raw* proveniente del la IMU. Este nodo provee una estimación de la orientación del robot a partir de lecturas de

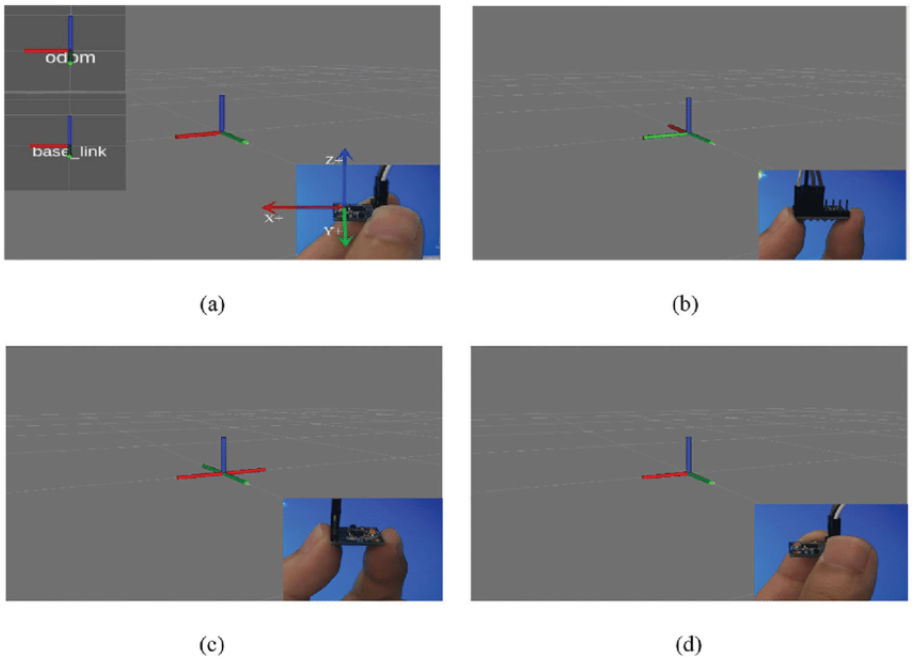

FIGURA 2.1: El ángulo *yaw* mostrado en RViz. (a) Posición inicial. (b) Posición rotada a 90 grados. (c) Posición rotada a 180 grados. (d) Posición rotada a 180 grados en sentido anti-horario.

velocidad angular y aceleraciones, sin la necesidad de un magnetómetro. A su salida, est nodo publica mensajes estándares del tipo sensor_msgs/Imu[2], requeridos por el nodo encargado de la localización del robot.

2.3.2. Tele-operación del robot mediante un nodo ROS

Esta prueba consistió en operar el robot mediante un nodo de ROS, el cual se encarga de publicar mensajes del tipo geometry_msgs/Twist[3]. Este mensaje es recibido por el robot y aplica los comandos de velocidad a cada una de las ruedas. Para esto se utilizó el nodo teleop_twist_keyboard:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

Este nodo permitió comandar el movimiento del robot de forma manual utilizando el teclado de la computadora mediante el uso de los siguientes comandos:

```
Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
```

[2]https://github.com/ccny-ros-pkg/imu_tools

[2]https://docs.ros.org/melodic/api/sensor_msgs/html/msg/Imu.html

```
e/c : increase/decrease only angular speed by 10 %
anything else : stop

CTRL-C to quit
```

Esta prueba se consideró satisfactoria el verificar que el robot era efectivamente controlado mediante este nodo. Las velocidades deseadas o *set-points* se verificaron recién durante las pruebas de odometría.

2.3.3. Odometría

Esta prueba esta basada en el mismo procedimiento utilizado para tele-operación del robot, con la diferencia que se procedió, además, a comprobar el cálculo de velocidad basado en la lectura arrojada por los encoders. Para esto se utilizó el nodo calculador de odometría elaborado por el autor en lenguaje C++, incluido en el repositorio del presente trabajo.

Para realizar la prueba, se procede de la misma manera que para la de tele-operación, con la diferencia de que se abre, además, el nodo de odometría antes de enviar los comandos de velocidad.

Para lanzar el nodo de odometría utilizamos el siguiente comando:

```
rosrun odom_publisher odom_publisher
```

Una vez listo, se hace un eco del tópico de odometría, el cual arrojará los valores publicados en la consola. La consigna es comparar los valores de velocidad publicados en el tópico de odometría con los comandos enviados desde el nodo `teleop_twist_keyboard`, debiendo ser éstos similares.

2.3.4. Generación de mapa

Para generar un mapa se utilizó la herramienta gmapping[4], ofrecida como un paquete de ROS. Esta herramienta es capaz de generar un *occupancy grid* en 2D, muy similar a un plano civil, a partir de las lecturas de *laserscan* generadas por el sensor Kinect, mas la información de odometría colectada por el robot.

2.3.5. Localización en mapa pre-existente

Esta prueba consistió en colocar el robot en un punto arbitrario del mapa sin proveerle información previa sobre su localización. Aquí el robot debe ser capaz de utilizar sus sensores para reconocer su entorno cercano y reconocer características conocidas entre este y el mapa, siendo capaz de estimar su ubicación. Para este procedimiento se utilizó el mapa generado en el apartado anterior en conjunto con la aplicación RViz para visualizar el estado del procedimiento.

[3] https://docs.ros.org/melodic/api/geometry_msgs/html/msg/Twist.html

[4] https://wiki.ros.org/gmapping

2.3.6. Navegación sobre mapa pre-existente

La navegación sobre un mapa pre-existente esta basada en todas las pruebas de integración anteriores y representa el punto culminante de la serie. Aquí se utilizó la aplicación RViz no solo para visualizar el mapa y la localización del robot, sino también para el envío de puntos de consigna dentro del mapa a los que el robot debe ser capaz de llegar utilizando toda su batería de sensores y estimadores a la vez que evita colisionar con las paredes y objetos presentes.

Capítulo 3

Conclusiones

En este capítulo se describen los aportes generados con el trabajo realizado, detallando los logros obtenidos. Así también, se especifican las técnicas mediante las cuales se hizo posible la ejecución del proyecto. Por último, se deja constancia de las metas originales que no pudieron ser abordadas dentro del alcance final, identificando las propuestas de acción a futuro.

3.1. Conclusiones generales

En este trabajo se completó la primer iteración en el ciclo de diseño e implementación de una planta de pruebas destinada a utilizarse en el aprendizaje de técnicas de robótica móvil a nivel personal o universitario.

Se pudo desarrollar una planta de pruebas completa y funcional en base a componentes disponibles en el mercado local, integrándola con el framework de robótica ROS. Se pudo desarrollar un firmware capaz de funcionar como una interfaz entre una base móvil comercial y ROS, así como interfacear los sensores necesarios con dicho framework de modo a posibilitar su uso inmediato en aplicaciones de navegación autónoma. Este desarrollo sienta las bases y estructura para una segunda iteración, dejando abiertas una serie de propuestas con mejoras que pueden implementarse en base a las necesidades que se le vayan presentando al usuario.

A lo largo del desarrollo de este trabajo final, se aplicaron extensivamente los conocimientos adquiridos durante la carrera. Mas allá de que el conjunto de asignaturas posibilitaron la apreciación de un panorama muy completo en lo que refiere a diseño y ejecución de un proyecto de sistemas embebidos, se destacan a continuación aquellas materias cuyo impacto en el desarrollo de este trabajo:

- Protocolos de Comunicación: se aplicaron los conocimientos adquiridos en la utilización de protocolos UART e I2C. Así también, se implementó la librería rosserial, la cual permite ser utilizada de manera intercambiable tanto con comunicación UART como Ethernet, utilizadas durante la etapa inicial y final del proyecto, respectivamente.
- Sistemas Operativos de Tiempo Real (I y II): se utilizaron los conocimientos adquiridos aplicando FreeRTOS como sistema operativo para el sistema propuesto.

- Arquitectura de microprocesadores: resultó necesario tener conocimientos sobre la Arquitectura ARM Cortex M para la programación de la plataforma STM32 NUCLEO y el uso efectivo de sus periféricos.
- Programación de microprocesadores: se aplicaron las buenas prácticas de programación que fueron mostradas a lo largo de la materia. Se empleó un formato de código consistente a modo de obtener código más modular, legible y reutilizable.
- Gestión de Proyectos en Ingeniería: la elaboración de un Plan de Proyecto para organizar el Trabajo Final, el cual facilitó en gran manera la ejecución del mismo y evitó demoras innecesarias.

Por lo tanto, se concluye que los objetivos planteados al comienzo del trabajo han sido alcanzados satisfactoriamente, habiéndose cumplido con los criterios de aceptación del sistema final y obtenido conocimientos valiosos para la formación profesional del autor.

3.2. Trabajo a futuro

Se plantea a continuación una lista de mejoras que podrían implementarse a fin de aumentar las capacidades del robot, las cuales se espera puedan ser abordadas por los usuarios que hayan decidido adoptar la plataforma, a quienes se exhorta a hacer aportes públicos al repositorio de modo a que todos puedan verse beneficiados con los mismos:

- Agregar soporte para sensores y actuadores disponibles en el robot Roomba que aún no han sido implementados en el sistema.
- Agregar un magnetómetro a la IMU via SPI o bien, reemplazarla por una IMU que ofrezca 9 grados de libertad, como la MPU9250, de modo a que la estimación de la pose del robot sea mas precisa.
- Utilizar la salida de potencia del controlador del motor de la aspiradora para alimentar el sensor Kinect, eliminando la necesidad de una batería externa.
- Reemplazar el actual computador OrangePI PC con un Nvidia Jetson Nano o similar, de modo a poder procesar la información proveniente desde una mayor cantidad de sensores.
- Implementar un segundo sensor Kinect apuntando en dirección contraria al actual para duplicar el campo de visión.
- Agregar soporte para ROS 2.