UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Integración de una computadora central en el Rover UVG para la ejecución de ROS

Protocolo de trabajo de graduación presentado por Diego Gerardo Mencos Caal, estudiante de Ingeniería Mecatrónica

Guatemala,

2022

Resumen

En la actualidad *Robot Operating System* (ROS) es una de las principales herramientas que se utiliza para el desarrollo de robots en toda clase de aplicaciones. Se caracteriza por ser versátil y de código abierto, por lo que se ha considerado a ROS una herramienta sumamente útil para resolver problemas de fabricación, optimización, investigación, recorrido, entre muchos más. Cuenta con una gran cantidad de librerías y recursos que facilitan al desarrollador crear programas para tareas sencillas hasta tareas mucho más complejas.

En este protocolo se presenta un propuesta de integración de ROS al Rover UVG, de manera que se pueda dotar de autonomía y sea capaz de cumplir con múltiples tareas. ROS provee herramientas muy útiles de simulación, como lo son Gazebo y Rviz, de manera que se obtenga un modelo realista y se puedan realizar todas las pruebas de funcionamiento correctamente. De igual forma se evaluará cual es el modelo óptimo de Raspberry Pi para ser utilizado como computadora central del Rover UVG y poder comunicarse con los módulos externos.

Antecedentes

La integración de *Robot Operating System* (ROS) en toda clase de robots cada vez es más común, ya sea manipuladores seriales, robots móviles con ruedas, drones, animatrónicos, entre muchos más. En donde se utiliza dicho sistema operativo para controlar robots con aplicaciones sencillas hasta mucho más complejas y dinámicas. A continuación se presentan distintos proyectos que integran ROS, así como el proyecto del Rover UVG con el cual se estará trabajando.

Ryerson University Rover

En [1] Daniel Snider de la universidad de Ryerson en Toronto Canadá describé la implementación de ROS en un Rover para la University Rover Challenge (URC) del año 2017. Se trabajó en una arquitectura para ROS de manera que existiesen cinco sistemas principales dentro del proyecto, los cuales abarcan el sistema autónomo del robot, el sistema de manejo, sistema de visualización, sistema de odometría y por último el sistema de posición. Cada uno de estos sitemas fue implementado como un paquete independiente en la arquitectura de ROS de forma que se pudiera organizar y estructura de mejor manera cada una de las partes del robot. En [1] se describe con detalle lo que se integra en cada paquete, la forma de comunicación entre los módulos, el lenguaje de programación utilizado, así como todos los sensores y actuadores empleados en cada sistema. Se utilizó una Raspberry Pi 3 para cargar el sistema operativo la cual se comunica a su vez con otros microcontroladores (como Arduino) que se encargan de poder leer los sensores, controlar los motores, entre otras diversas tareas.

Robot móvil autónomo basado en ROS

En [2] Jun Takamatsu expone la integración de ROS sobre un robot móvil autónomo en donde utilizan un LIDAR 2D y camaras RGB-D. Para dicha implementación utilizaron gran parte de los paquetes que vienen integrados dentro de ROS, en los cuales solo realizaron cambios mínimos de los parámetros que vienen por defecto para poder integrarlo a su robot. Su principal contribución fue obtener dos cofiguraciones del sistema de ROS para la navegación y autonomia del robot en trayectorias específicas. Para la primera configuración hicieron uso del LIDAR 2D junto con una Raspberry Pi 3, y para la segunda configuracion hicieron uso del LIDAR como la camara RGB-D para poder trabajar en conjunto con Intel NUC. Para ambas configuraciones lograron que el robot pudiera evadir obstáculos en su camino, así como parar en obstáculos inevitables.

Robot Explorador Modular

Hasta el año 2021 Hector Sagastume y Javier Archila realizaron las últimas modificaciones para el robot explorador modular. En [3] Hector Sagastume logró realizar una estructura capaz de adaptarse al sistema ya existente brindando protección a los componentes, reducción de tamaño, reducción de peso y optimización de espacio. De igual manera implementó motores con controlador que brindan mayores revoluciones por minuto sin comprometer la tracción del robot explorador, manteniendo una capacidad todo terreno. Sagastume también implementó cinco sensores ultrasónicos dentro de la estrucutra de forma que fuesen capaces de informar al usuario y al sistema de los cambios de proximidad y ambiente en su entorno. En [4] Javier Archila implementó el autopiloto PixHawk en conjunto con una Raspberry Pi 3 para manejo de telemetría a través de Internet y manejo remoto. Si bien en [3] y [4] fueron capaces de reestructurar dicho robot explorador y dotarlo de capacidades mecánicas y eléctricas para su funcionamiento, ambos concuerdan que se debe seguir trabajando y desarrollando el robot de manera que se puedan implementar nuevos módulos logrando incrementar las aplicaciones del robot.

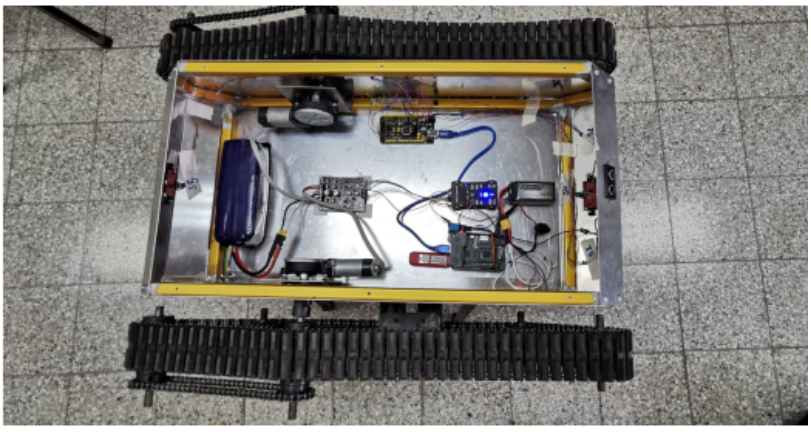

Figura 1: Arquitectura actual del Rover [3]

Robot Operating System

En [5] Marcos Izeppi trabajó en la implementación de un algoritmo capaz de navegar a través de terrenos desconocidos y siguiendo un comportamiento brindado por el algoritmo de optimización de enjambre causando que el robot converja hacia el mínimo local de la función de costo. Determinó la viabilidad de la utilización de ROS para simulaciones de algoritmos de búsqueda de trayectorias como lo es el algoritmo de Dijkstra y D*. Para ello utilizó Gazebo, entorno de simulación de robots que viene en conjunto con ROS. De igual forma Izeppi expone comandos básicos y componentes principales de ROS para poder comprender el funcionamiento y partes esenciales que conforman cualquier proyecto en dicho sistema operativo. Pese a que en [5] se describe información útil y necesaria para el conocimiento y utilización de ROS, el objetivo principal fue únicamete utilizar ROS como herramienta de simulación para comprobar la funcionalidad de dichos algoritmos, con lo cual no se aprovecha al máximo todos los recursos que el sistema operativo nos puede ofrecer así como un gran número de paquetes que pueden mejorar la eficiencia del proyecto y aumentar su alcance.

Justificación

Hasta el momento, solamente se ha logrado controlar el Rover UVG mediante control remoto lo cual límita de gran manera sus aplicaciones. Por ello, mediante la implementación de un sistema operativo capaz de controlar y manejar al robot de una manera adecuada, se puede ampliar el alcance del Rover para que sea apto para cumplir con distintas tareas más complejas por sí solo.

Por ello se considera a ROS como la herramienta óptima para su implementación dentro del Rover UVG. ROS se caracteriza por ser una plataforma de código abierto y versátil. Este sistema operativo cuenta con un gran número de paquetes adecuados para sensores/actuadores e incluso para la programación de tareas simples del robot. De igual manera, permite la creación de paquetes propios para la realización de tareas especificas así como la comunicación con sistemas o equipos externos.

Los robots exploradores modulares que se fabrican en distintas partes del mundo son utilizados para aplicaciones que abarcan desde el reconomiento, rescate y exploración hasta aplicaciones militares y espaciales. La integración de ROS al Rover UVG permitiría poder cumplir con alguna o múltiples de estas aplicaciones de manera que se tenga un prototipo bastante completo como base para futuras investigaciones o adaptaciones.

Objetivos

Objetivo General

Integrar una computadora central al Rover UVG que sea capaz de ejecutar ROS, con lo cual se pueda dotar de autonomía al robot y permita ampliar sus aplicaciones.

Objetivos Específicos

- Seleccionar una computadora central adecuada para ejecutar ROS y que sea integrable al Rover.
- Crear una imagen estandarizada del sistema operativo.
- Establecer los requerimientos de comunicación para los módulos de percepción y actuación del Rover e integrarlos para su funcionamiento en conjunto.
- Realizar simulaciones realistas del Rover para validar el funcionamiento de los módulos externos y su correcta integración con ROS.

Marco teórico

Robotic Operating System (ROS)

Robot Operating System (ROS) es una colección de marcos flexibles para la escritura de software de robots. Se caracteriza por ser de código abierto y proveer conexiones entre procesos de múltiples dispositivos. Cuenta con un conjunto amplio de librerías y convenciones que facilitan al desarrollador la tarea de crear un comportamiento complejo y robusto en una amplia variedad de plataformas robóticas. Las principales bibliotecas de ROS han sido desarrolladas en su totalidad para ser utilizadas con el sistema operativo Ubuntu, con base en Linux y de forma experimental (con fallas y detalles por perfeccionar) funciona con IOS y Windows [6].

El ecosistema de ROS se compone de decenas de miles de usuarios en todo el mundo, que trabajan en pequeños proyectos de aficionados hasta sistemas de automatización industrial de gran escala. ROS fue diseñado para ser sistema operativo modular, de forma que los usuarios puedan seleccionar y elegir que partes son útiles y qué partes se prefiere implementar. La comunidad de usuarios de ROS definen una infraestructura común para proporcionar un punto de integración que ofrece el acceso a los controladores de hardware, las capacidades de robots genéricos, herramientas de desarrollo, librerías externas útiles, y más [7].

Los paquetes en ROS son el elemento principal de su arquitectura. Estos pueden contenedor los procesos de ROS en tiempo de ejecución (nodos), conjunto de datos, archivos de configuración o cualquier elemento que sea necesario para su funcionamiento. Cada paquete puede ofrecer una determinada función de manera que el software se pueda construir y reutilizar fácilmente. De igual forma, se pueden crear paquetes propios para el desarrollo de tareas específicas dentro de cualquier robot o proyecto donde se trabaje. Un nodo es un proceso que realiza cálculos, y se puede comunicar con otros nodos mediante la transmisión de *topics*, *RPC services* y el *Parameter Server*. Generalmente, cada sistema de control que conforma un robot se describe por medio de paquetes, y estos a su vez por medio de nodos con tareas específicas los cuales se comunican entre sí para lograr los principales objetivos del robot en desarrollo [8].

Gazebo

Gazebo es un simulador 3D multi-robot con propiedades dinámicas y cinemáticas completas. Ofrece la posibilidad de simular con precisión y eficiencia para diversidad de robots, objetos y sensores en ambientes complejos interiores y exteriores. Gazebo genera la realimentación realista de sensores, como las interacciones entre los objetos físicamente plausibles, incluida una simulación precisa de la física de cuerpo rígido. Es una herramienta gratuita de simulación, realmente útil para prueba y simulación de algoritmos elaborados, de manera que brinda la posibilidad de visualizar aciertos y fallas del robot en desarrollo [9].

La arquitectura de Gazebo está basada en el motor de simulación ODE (Open Dynamics Engine), creado por Russell Smith, aunque su desarrollo final se atribuye a Andrew Howard y Nate Koenig, por sus investigaciones en la Universidad del Sur de California [10]. Gazebo permite de forma directa, crear mundos y objetos idóneos para insertar el modelo robótico que se esté desarrollando, con el fin de visibilizar la manera en la que el mismo interactúa en un ambiente determinado, de manera que se pueda verificar el funcionamiento de sensores, aptitudes de movilidad e incluso de superación de obstáculos. La integración entre ROS y Gazebo es proporcionada por un conjunto de *plugins* de Gazebo que apoyan muchos robots y sensores existentes. Debido a que los *plugins* presentan la misma interfaz de mensaje que el resto del ecosistema ROS, se pueden escribir nodos ROS que sean compatibles con la simulación, los datos registrados y el hardware de los robots [7].

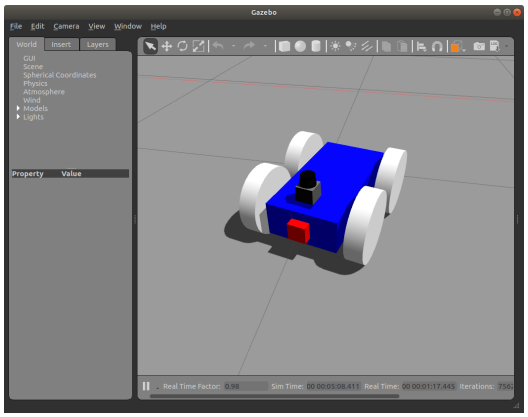

Figura 2: Entorno gráfico de Gazebo [7]

.

Rviz

Rviz es la herramienta de visualización 3D con la que dispone ROS. Esta herramienta proporciona la representación de robots, datos de sensores como escaneos láser, nubes de puntos tridimensionales e imágenes de cámaras dentro del robot, entre otras. Rviz tiene funciones estrictas de visualización por lo que a diferencia de Gazebo, no provee esquemas de simulación de fricción, gravedad y otros, ni la construcción de escenarios de interacción. La forma en que funciona Rviz es simplemente leyendo e interpretando los diferentes datos contenidos en mensajes de ROS. Por tanto, es necesario tener un generador externo de estos mensajes, como puede ser un robot físico o simulado [11].

Lenguaje de programación XML

La siglas XML son la abreviación de "Extensible Markup Language", lo cual se trata de un lenguaje utilizado para estructurar la información en cualquier documento que contenga texto como lo son archivos de configuración de una aplicación específica, base de datos, hojas de cálculo, entre otras. La razón de su popularidad se debe al hecho de ser un estándar abierto y además libre, creado por W3C, el consorcio *World Wide Web*, en colaboración con un equipo de trabajo que incluye representantes de las compañías de software más importantes. El lenguaje XML fue creado en 1996 y desde ese momento su utilización tuvo un crecimiento sostenido [7].

La tecnología XML mantiene la información estructurada de la forma más abstracta y reutilizable posible. Por ello, un documento XML está formado las siguientes partes:

- **Prólogo:** Los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas.
- **Cuerpo:** El cuerpo tiene que contener sólo un elemento raíz, característica obligatoria para que el documento esté bien formado. Sin embargo es necesaria la adquisición de datos para su buen funcionamiento.
- **Elementos:** Los elementos XML pueden tener contenido (más elementos, caracteres o ambos) o bien ser elementos vacíos.
- **Atributos:** Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento. Deben ir entre comillas.
- **Entidades predefinidas:** Entidades para representar caracteres especiales para que no sean interpretados como marcado en el procesador XML.
- **Comentarios:** Son aclaraciones que pueden ser escritas para informar al lector, y son ignorados por el procesado.

Formato unificado de descripción para robots (URDF)

United Robotics Description Format (URDF) es un formato de lenguaje utilizado para la descripción de robots en el marco de gramática XML. Es una herramienta útil para el modelado de un robot con lo cual se pueden realizar pruebas de simulación y análisis. Para la creación del modelo no solamente se definen enlaces y uniones del robot, también es necesario conocer cierta información física de los componentes básicos, como los atributos de masa, inercia, colores y tipos de articulaciones, ya sea una articulación giratoria o una articulación traslacional [7].

Metodología

Como primer paso se deberá instalar una máquina virtual capaz de correr el sistema operativo Ubuntu Linux, para poder utilizar ROS. De igual forma se deberá instalar ROS

dentro de la máquina virtual y se deberán realizar pruebas para verificar que todo este funcionando correctamente. Se procederá a conocer comandos en la terminal para crear proyectos de ROS, paquetes, nodos y demás de manera que se este familiarizado con el entorno de trabajo de ROS. De igual forma se utilizará TurtleSim, el cual es un proyecto que ayuda a conocer el funcionamiento de ROS y ROS-PKG y la forma de simulación por medio de Gazebo. De forma paralela se tendrá que investigar el modelo más adecuado de Raspberry Pi para ser utilizado como computadora central, en donde se estará corriendo ROS en conjunto con todos los módulos externos.

Se procederá a crear el proyecto a implementar, por lo que se evaluará la mejor manera de crear los paquetes. Ya sea teniendo un paquete para cada sistema del Rover, o bien distribuyendo esto de una manera más eficaz. Se dividirán cada una de las tareas en nodos en donde estarán integrados todos los módulos externos (sensores/actuadores) y se establecerá la forma de comunicación entre nodos para crear una red de comunicación dentro y fuera de cada paquete. Teniendo todo esto, se creará el modelo URDF del Rover UVG para poder realizar las simulación por medio de Gazebo y utilizando la herramiento Rviz. Se comproborá el funcionamiento del robot utilizando cada módulo externo y colocando al robot dentro de distintos entornos.

Obteniendo resultados exitosos por medio de las simulaciones, se procederá a utilizar el modelo seleccionado de Raspberry Pi en donde se estará ejucutando el proyecto de ROS previamente realizado. Se conectarán los módulos externos (ya sea directamente o por medio de otros microcontroladores como Arduino) y se realizaran las pruebas físicas del robot. Se colocará al robot en distintos entornos para observar su comporamiento por medio de sus actuadores y los datos obtenidos de los respectivos sensores.

Cronograma de actividades

					Julio				Agosto					Septiembre			
No.	Nombre de la tarea	Fecha de inicio	Fecha de finalización	Estado	04.07.2022	11.07.2022	18.07.2022	25.07.2022	01.08.2022	08.08.2022	15.08.2022	22.08.2022	29.08.2022	05.09.2022	12.09.2022	19.09.2022	26.09.2022
1	**Conocer entorno ROS**																
1.1	Instarlar máquina virtual	04.07.2022	11.07.2022	Abierto													
1.2	Instalar ROS	04.07.2022	18.07.2022	Abierto													
1.3	Conocer comandos útiles para ROS	11.07.2022	25.07.2022	Abierto													
1.4	Utilizar TurtleSim para interactuar con el entorno	18.07.2022	01.08.2022	Abierto													
1.5	Creación de paquetes y nodos de prueba	25.07.2022	01.08.2022	Abierto													
2	**Creación de proyecto**																
2.1	Organizar y plantear estructura de proyecto	25.07.2022	01.08.2022	Abierto													
2.2	Creación de paquetes según estructura	01.08.2022	15.08.2022	Abierto													
2.3	Creación de nodos para módulos externos	08.08.2022	15.08.2022	Abierto													
2.4	Hablar con demás compañeros con respecto a sus nodos para sensores/actuadores	15.08.2022	22.08.2022	Abierto													
2.5	Comunicación entre nodos	15.08.2022	22.08.2022	Abierto													
2.6	Pruebas de comunicación	15.08.2022	29.08.2022	Abierto													
3	**Simulación**																
3.1	Realizar URDF Rover UVG	22.08.2022	22.08.2022	Abierto													
3.2	Interactuar entorno Gazebo	22.08.2022	29.08.2022	Abierto													
3.3	Crear un mundo para el robot	29.08.2022	05.09.2022	Abierto													
3.4	Realizar pruebas dentro de distintos entornos	05.09.2022	19.09.2022	Abierto													
4	**Raspberry Pi**																
4.1	Investigar modelo adecuado de Raspberry Pi	04.07.2022	11.07.2022	Abierto													
4.2	Realizar pruebas de descarga de sistema operativo	12.09.2022	19.09.2022	Abierto													
4.3	Descargar proyecto	19.09.2022	19.09.2022	Abierto													
4.4	Realizar pruebas físicas en distin	19.09.2022	26.09.2022	Abierto													
4.5	Resultados finales	26.09.2022	26.09.2022	Abierto													
5	Redactar documento de tesis	30.05.2022	26.09.2022	Abierto													

Figura 3: Cronograma de actividades

Índice preliminar

Referencias

[1] D. Snider, M. Mirvish, M. Barcis y V. A. Tezer, "University Rover Challenge: Tutorials and Team Survey," en *Robot Operating System (ROS)*, Springer, 2019, págs. 315-370.

[2] S. Gatesichapakorn, J. Takamatsu y M. Ruchanurucks, "ROS based autonomous mobile robot navigation using 2D LiDAR and RGB-D camera," en *2019 First international symposium on instrumentation, control, artificial intelligence, and robotics (ICA-SYMP)*, IEEE, 2019, págs. 151-154.

[3] H. Sagastume, "Diseño Mecánico, Selección de Motores e Implementación de Sensores para un Robot Explorador Modular," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[4] J. Archila, "Diseño e implementación de capacidades automáticas de navegación para un Robot Explorador Modular," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[5] M. Izeppi, "Aplicación de Herramientas de Aprendizaje Reforzado y Aprendizaje Profundo en Simulaciones de Robótica de Enjambre con Restricciones Físicas," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[6] M. Quigley, K. Conley, B. Gerkey y col., "ROS: an open-source Robot Operating System," en *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, pág. 5.

[7] R. Merino López, "Creación de modelo URDF del robot manfred," Tesis de mtría., Universidad Carlos III de Madrid, 2014.

[8] Open Robotics, *Nodes*, `http://wiki.ros.org/es/Nodes`, 2021.

[9] Open Source, *Gazebo*, `https://gazebosim.org/home`, 2021.

[10] N. Koenig y A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," en *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, IEEE, vol. 3, 2004, págs. 2149-2154.

[11] Open Source, *Rviz*, `http://wiki.ros.org/rviz`, 2021.