SOAL 1

a. ROS (Robot Operating System) adalah sebuat set library software dan tools-tools yang akan sangat membantu dalam pembuatan sebuah aplikasi robot. Mulai dari algoritma-algoritma canggih hingga alat pengembangan yang hebat, ROS akan memenuhi apa yang dibutuhkan dalam projek pembuatan robot. Adapun peran utama ROS dalam pengembangan robotic modern yaitu ROS memungkinkan pengembangan untuk berinteraksi dengan perangkat keras robot tanpa harus memahami detail spesifik perangkat, memungkinkan pengembang untuk menggabungkan berbagai komponen dan membuatnya lebih mudah untuk mengembangkan dan mengintegrasikan fungsionalitas baru, memfasilitasi komunikasi yang cepat dan dinamis antara berbagai komponen robot, seperti sensor dan actuator, dan dapat dengan mudah menambahkan dan mengkonfigurasi perangkat keras yang berbeda ke dalam sistem, sehingga meningkatkan fleksibilitas desain. ROS sangatlah penting untuk integrasi komponen robot karena ROS akan memainkan peran penting dalam standarisasi, kemandirian, fleksibilitas, dan reusabilitas.

b. Terdapat beberapa perbedaan dari ROS dan ROS2 yaitu mengenai platform, ROS 1 hanya mendukung Linux (Ubuntu) dan macOS, sedangkan ROS 2 juga mendukung Windows 10, memperluas fleksibilitas pengembangan aplikasi robotik. Mengenai bahasa pemograman, ROS 1 mendukung C++ 03 dan Python 2, sedangkan ROS 2 menggunakan C++ 11 dan mendukung Python 3.5 ke atas. Mengenai transport protocol, ROS 1 menggunakan protokol komunikasi TCPROS dan UDPROS, sementara ROS 2 menggunakan DDS (Data Distribution Service). Embeddednesia akan menjelaskan lebih lanjut tentang DDS dan perbedaannya dengan roscore pada ROS 1 di tulisan terpisah. Lalu mengenai build systemnya, ROS 1 menggunakan Catkin sebagai sistem build, sedangkan ROS 2 menggunakan Colcon sebagai alat build universal untuk mengatasi perbedaan dalam penggunaan alat build pada pengembangan komponen ROS. Keunggulan ROS2 dibandingkan ROS terletak pada performa, keamanan, dan pemeliharaan. ROS2 menggunakan DDS untuk komunikasi yang lebih cepat dan pengaturan Quality of Service (QoS) yang lebih baik. Selain itu, ROS2 menawarkan fitur keamanan built-in seperti otentikasi dan enkripsi, yang tidak tersedia di ROS. Dengan dukungan pengembangan yang lebih aktif dan kompatibilitas multiplatform (Linux, Windows, dan macOS), ROS2 memudahkan pemeliharaan jangka panjang. Oleh karena itu, ROS2 adalah pilihan yang lebih baik untuk aplikasi robotik modern.

c. Simulasi robotik memainkan peran krusial dalam pengembangan robot karena memberikan kesempatan bagi pengembang untuk menguji desain, algoritma, dan interaksi robot dalam lingkungan virtual sebelum membangunnya secara nyata. Keuntungannya mencakup penghematan biaya, karena kesalahan dapat ditemukan dan diperbaiki tanpa harus mengeluarkan biaya tinggi untuk prototipe fisik, serta efisiensi waktu, karena pengembang dapat dengan cepat mencoba berbagai skenario tanpa perlu pengaturan fisik yang kompleks.

Sebagai contoh, dalam pengembangan robot otonom untuk pengiriman barang, simulasi dapat digunakan untuk menguji rute dan strategi navigasi di lingkungan perkotaan, yang membantu mengurangi risiko kerusakan pada robot fisik dan mempercepat keseluruhan proses pengembangan.

d. Gazebo merupakan simulator robot 3D yang terhubung dengan ROS, memungkinkan pengguna untuk menciptakan lingkungan simulasi yang realistis dan memvisualisasikan perilaku robot dengan tingkat akurasi yang tinggi. Untuk menghubungkan ROS dengan Gazebo, langkah pertama adalah menginstal kedua platform tersebut di sistem. Setelah itu, buat workspace ROS dan siapkan model robot menggunakan file URDF atau SDF. Kemudian, buat paket ROS yang menyimpan file-file terkait simulasi dan gunakan perintah `roslaunch` untuk menjalankan Gazebo dengan model robot tersebut. Selanjutnya, integrasikan node ROS untuk mengendalikan robot dan menguji fungsionalitasnya dalam simulasi. Dengan pendekatan ini, pengembang dapat melakukan pengujian dan pengembangan robot dengan lebih efisien sebelum menerapkannya di dunia nyata.

e. Dalam navigasi robot di lingkungan simulasi, beberapa konsep penting perlu dipahami, termasuk mapping dan lokalisasi. Mapping merupakan proses menciptakan peta yang mencakup informasi tentang dinding, objek, dan rintangan di sekitar robot. Dalam simulasi, robot memanfaatkan sensor virtual seperti LiDAR atau kamera untuk mengumpulkan data lingkungan, yang selanjutnya digunakan untuk membuat peta dengan algoritma SLAM (Simultaneous Localization and Mapping). Algoritma ini memungkinkan robot untuk memetakan area sekaligus melacak posisinya. Sementara itu, lokalisasi adalah proses untuk menentukan posisi robot dalam peta yang telah dibuat. Robot membandingkan data sensor dengan peta untuk menemukan posisinya secara akurat. Metode yang umum digunakan untuk lokalisasi termasuk odometry, yang memperkirakan posisi berdasarkan pergerakan robot, serta filter seperti Kalman filter dan Particle filter yang menggabungkan data dari berbagai sumber. Untuk menerapkan fitur mapping dan lokalisasi pada robot penjaga, langkah pertama adalah menyiapkan sensor seperti kamera dan LiDAR untuk mendeteksi dan memetakan lingkungan. Kemudian, gunakan algoritma SLAM seperti GMapping untuk membangun peta dan melacak posisi robot. Selanjutnya, buat node ROS yang mengelola data sensor dan memungkinkan pembaruan peta secara real-time. Anda juga bisa menambahkan fitur deteksi gerakan atau pengenalan wajah untuk meningkatkan kemampuan robot. Setelah semua komponen terintegrasi, jalankan simulasi di Gazebo untuk menguji efektivitas robot dalam menjelajahi dan mengawasi area yang telah ditentukan.

f. TF (Transform) dalam ROS merupakan sebuah paket yang berfungsi untuk mengelola dan menggambarkan transformasi antar berbagai frame referensi dalam ruang tiga dimensi. Dengan TF, robot dapat memahami posisi dan orientasinya karena informasi mengenai hubungan spasial antara sensor, aktuator, dan komponen lain tersedia. Sebagai contoh, ketika

sebuah robot memanfaatkan LiDAR untuk memetakan lingkungan, TF dapat mengubah data dari frame sensor ke frame robot, sehingga robot dapat bergerak dengan tepat dan benar dalam simulasi. Dengan adanya TF, robot dapat secara real-time memperbarui posisinya dan berinteraksi dengan lingkungan di sekitarnya secara efisien, meskipun terjadi perubahan orientasi atau posisi.