# SOAL 1

a. ROS adalah sebuah meta-operating system yang bersifat open-source yang digunakan untuk mengembangkan robot itu sendiri. ROS digunakan untuk memudahkan kita untuk mengendalikan robot sehingga pengguna dapat menghubungkan, mengontrol, dan mengintegrasikan komponen robot secara efisien. ROS penting dalam pengemabangan robotika modern karena diperlukan integrasi berbagai komponen robot seperti sensor, aktuator, dan kamera yang lebih efisien. Kita dapat melihat bagaimana komponen-komponen robot itu membaca data dan tersinkonisasi secara real-time sehingga robot kita dapat bekerja secara harmonis dalam satu sistem dan lebih presisi.

b. Perbedaan utama dari ROS dan ROS2 adalah ROS masih menggunakan the ROS Master-Slave Architecture dan the XML-RPC middleware. Sedangkan ROS2 sudah menggunakan Data Distribution Service (DDS) yang dimana, dari segi performa, ROS2 memiliki tingkat efisiensi yang lebih tinggi dan latensi yang rendah. Selain itu, karena sudah menggunakan fitur DDS untuk komunikasinya antar nodes nya, ROS2 memiliki tingkat real-time processing yang lebih tinggi dibandingkan ROS. Real-time processing sangatlah penting dalam pembuatan robotika supaya kita dapat membuat robot dengan lebih presisi. Lalu, ROS tidak memungkinkan kita untuk membuat lebih dari 1 nodes dalam satu process. Sedangkan dengan ROS2, kita dapat membuat multiple nodes dalam satu process.

Dari segi keamanan, ROS2 memiliki tingkat keamanan yang lebih tinggi. Lagi-lagi dikarenakan fitur DDS-nya. Fitur DDS sudah include security system yang menawarkan encryption, authentication, and access control. Fitur ini sangat penting untuk melindungi data kita.

Untuk pemeliharaan jangka Panjang, tentu ROS2 lebih baik karena, seperti sebelumnya, ROS2 memiliki tingkat keamanan yang lebih tinggi dan juga ROS2 menyediakan platform yang stabil untuk mengembangkan dan menerapkan aplikasi robotika.

Mengapa pengembang cenderung memilih ROS2 untuk proyek baru? Jawabannya mengapa tidak? Fitur-fitur yang diberikan oleh ROS2 jauh lebih baik dibandingkan ROS. Tetapi itu terserah untuk penggunanya. Jika kita tidak siap untuk memulai menggunakan ROS2, sebaiknya kita harus memikirkan lagi bagaimana cara kita dapat bersaing dengan orang-orang yang menggunakan ROS2

c. Simulasi robot sangatlah penting dalam pengembangan robot karena dengan melakukan simulasi, kita dapat mengurangi resiko kesalahan dalam membuat robot fisik. Dikarenakan dapat mengurangi resiko kesalahan dalam membuat robot fisik, kita juga dapat menghemat biaya dalam pembuatan robot karena kita tidak melakukan kesalahan langsung dengan komponen yang kita gunakan sehingga dapat menghindari pengeluaran biaya yang tidak perlu. Lalu dengan simulasi, kita bisa mensimulasikan bagaimana kondisi yang kita inginkan dengan virtual tanpa harus mensimulasikannya secara langsung

Contoh kasus:

Misal kita ingin membuat robot yang bisa memadamkan api kebakaran secara autonomous.

Tentu untuk membuat kondisi kebakaran secara langsung sangat sulit untuk disimulasikan, robot kita bisa terbakar dan itu sangat membuang-buang uang kita. Terus kalo robot kita sudah terbakar, kita harus buat lagi dan itu sangat membuang-buang waktu kita. Jadi, kita dapat mensimulasikan robot kita dengan virtual supaya dapat menghemat waktu dan biaya kita 😊.

d. Gazebo adalah simulator 3D yang terintegrasi dengan ROS. Dengan Gazebo, pengguna dapat mensimulasikan keadaan fisik dunia asli dengan tingkat realistis yang sangat tinggi sehingga kita dapat menguji robot kita tanpa risiko kerusakan fisik. Gazebo dapat mensimulasikan bagaimana pencahayaan, fisika realistis, dan kolisi yang terdapat di fisik dunia asli.

Langkah-langkah dasar mengintegrasikan ROS dengan Gazebo:

1. Instalasi
    - Instalasi Operating System
      Instalasi sistem operasi Ubuntu atau varian lainnya.
    - Instalasi ROS dan Moveit
2. Membuat ROS Workspace
3. Mengontrol Aktuator Robot
   Untuk mengontrol joint aktuator pada robot, Anda dapat menggunakan beberapa pilihan berikut ini:
    - Mempublish ROS Topic melalui terminal
    - Mengontrol menggunakan (rqt_joint_trajectory_controller)
    - Mengontrol menggunakan program Python atau CPP
4. Mengontrol Aktuator Robot Menggunakan ROS Action
   Alternatif lain untuk mengendalikan joint aktuator pada robot yaitu menggunakan ROS Action. Kelebihan penggunaan ROS Action, terdapat feedback yang dapat digunakan untuk mengetahui kondisi terkini dari action yang sedang dijalankan.
5. Visualisasi Robot State Menggunakan RVIZ
   Sekarang gerakkan salah satu joint pada robot, maka secara otomatis visualisasi robot pada RVIZ akan menyesuaikan dengan kondisi robot pada Gazebo.

e.

Navigasi robot di dunia simulasi merupakan salah satu aspek penting dari robotika yang memungkinkan robot bergerak dari satu lokasi ke lokasi lain secara otonom. Sistem navigasi pada robot bekerja dengan cara mencari rute yang paling optimal dan dapat menghindari obstacle. Apa hal yang dibutuhkan untuk mengimplementasi navigasi pada robot

1. Map
   Hal yang paling utama dalam navigasi dari robot, tentu saja adalah sebuah map. Map ini bisa diberikan ke dalam robot atau bisa juga robot itu yang membuat map itu sendiri. SLAM (Simultaneous Localization And Mapping) dibuat agar robot dapat membuat map dengan bantuan manusia atau dengan sendirinya.
2. Pose of Robot
   Robot harus mengetahui di posisi apa dia berada dan ke arah mana dia menghadap. Sebuah robot harus dapat menghitung atau mengestimasi pose (position + orientation). Artinya sebuah robot harus ditentukan dahulu dimana posisi awal dia (x,y,z) dan ke arah mana dia menghadap (x,y,z,w). Untuk sekarang, cara untuk menghitung pose estimation adalah dengan dead reckoning. Dead reckoning adalah teknik navigasi yang menggunakan posisi awal, kecepatan, dan arah untuk menentukan posisi baru. Seberapa jauh robot bergerak diukur dengan seberapa banyak putaran rodanya.
3. Sensing
   Sensing sangat penting agar robot dapat menghindari sebuah obstacle seperti dinding atau objektif lainnya ayng terbaca oleh sensor. Sensor yang digunakan adalah sensor jarak dan sensor kamera. Sensor jarak bisa menggunakan ultrasonic sensor atau infrared

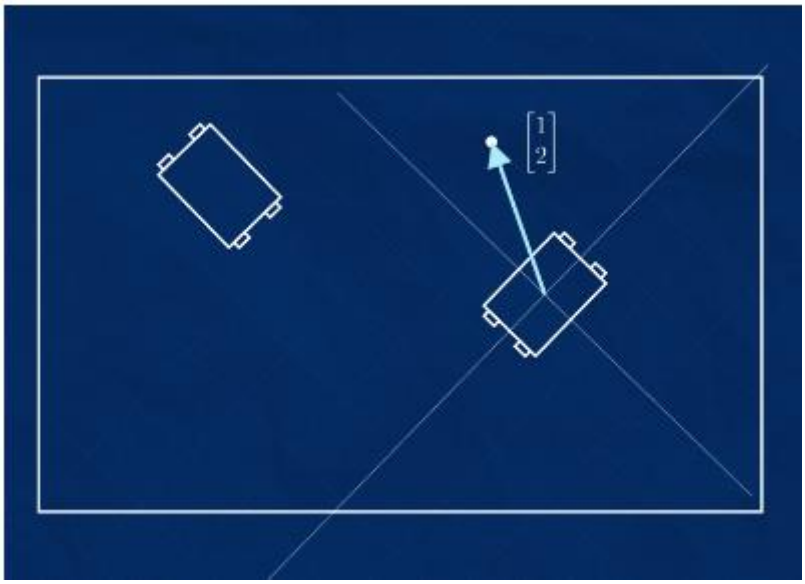
sensor. Sensor kamera bisa menggunakan Kinect, stereo cameras, mono cameras, dan lain lain

4. Path calculation and Driving
   Hal yang paling penting dalam sistem navigasi adalah menghitung jarak yang paling optimal untuk ke tujuan. Ini disebut path search and planning, banyak algoritma untuk menghitung ini ada A* Algorithm, potential field, particle filter, dan RRT

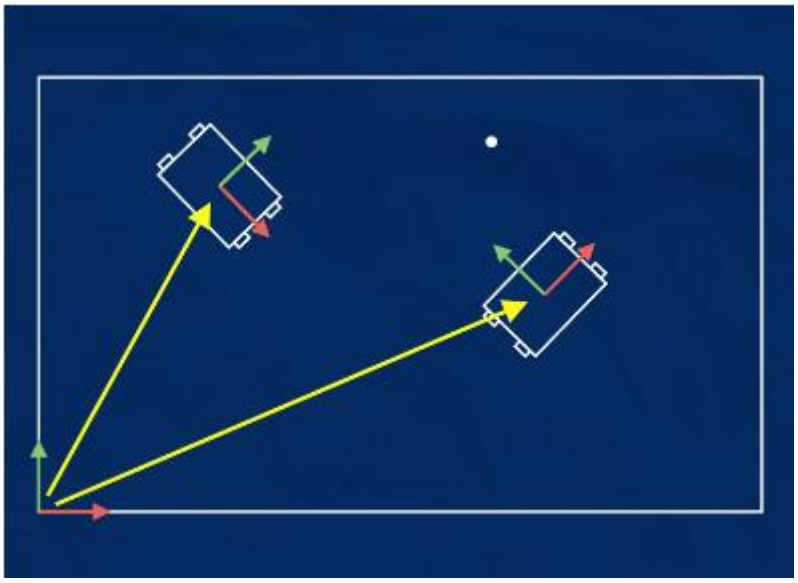
f.

TF merupakan ilmu matematika robot yang penting karena tool ini digunakan untuk menghitung pergerakan robot yang direpresentasikan dari satu sudut pandang, dan merepresentasikannya dalam sudut pandang berbeda.

Mengapa TF sangat berguna? Kita asumsikan dalam gambar




Disini terdapat dua robot dan salah satu robotnya ingin menuju ke suatu titik. Bagaimana robot tahu cara agar bisa kesana?



Untuk menyelesaikan masalah ini, kita harus tahu koordinat atau frames dari masing masing robot. Frames adalah sistem koordinat dalam robot dalam ruang 3D. Selanjutnya kita akan menentukan transform diantara frames. Transform memberi tahu kita translasi dan rotasi yang diperlukan supaya dapat ke frame yang berbeda.

TF di ROS memungkinkan robot untuk memahami dan menjaga konsistensi posisi dan orientasi antar berbagai bagian tubuh dan sensor dalam ruang tiga dimensi. Dengan TF, ROS dapat menangani semua kerangka koordinat secara otomatis, yang sangat penting untuk robotika simulasi dan dunia nyata.