SOAL 1 BAGIAN A – F

Valentino Daniel Kusumo - 19624107

a) ROS adalah Meta-Operating System Robot yang berjalan di atas Sistem Operasi (berbeda dengan Sistem Operasi → perangkat lunak system). ROS merupakan *framework* untuk membuat perangkat lunak robot. Didalam *framework* tersebut terdapat perangkat lunak tools, library, dan package yang dapat digunakan untuk mengontrol perangkat keras robot. ROS ini bersifat *open source* (dikembangkan secara bebas & terbuka). ROS juga fleksibel karena dapat dikombinasikan dengan beberapa perangkat lunak & menggunakan beberapa *library* lainnya.

Dengan adanya ROS, pengguna tidak perlu lagi membuat perangkat lunak robot dari awal dan ROS ini perangkat lunak robot yang saling berbagi sehingga memudahkan dalam mengembangkan sebuah robot secara bersama – sama.

ROS penting untuk integrasi berbagai komponen robot karena ROS dapat menghubungkan berbagai elemen (elektronika, mekanik, dan program) sehingga pembuatan robot sangat dimudahkan ketika menggunakan ROS dan dapat dibuat menggunakan Arduino juga. ROS dapat menghubungkan berbagai komponen, seperti sensor, aktuator, dan kamera menggunakan *node publisher* (register ke parameter server), kemudian baru mengirim *data message* dengan *topic* sebagai namanya. Setelah itu, *node subscriber* dijalankan (register ke parameter server). Setelah *node subscriber* telah diregister, *node publisher* dan *node subscriber* dapat saling berkomunikasi

b) Pengembang cenderung memilih ROS2 untuk proyek baru mungkin karena ROS akan diberhentikan pada tahun 2025. Selain itu, tujuan ROS2 dibuat adalah untuk mengatasi beberapa keterbatasan ROS dan untuk memenuhi kebutuhan aplikasi robotik yang lebih kompleks dan modern. Berikut beberapa perbandingan ROS dan ROS2 dalam hal performa, keamanan, dan pemeliharaan jangka panjang.

Pembanding	ROS	ROS2
Performa	ROS dibangun dengan arsitektur yang lebih terpusat dan bergantung pada **Zero-Copy dan TCP/IP** untuk komunikasi antar proses (kurang efisien dalam komunikasi *real time*)	ROS 2 menggunakan **Data Distribution Service (DDS)** untuk komunikasi, yang memberikan lebih banyak fleksibilitas dan efisiensi dalam hal komunikasi waktu nyata
Keamanan	Keamanan tidak menjadi prioritas utama dalam desain ROS, dan tidak ada protokol keamanan yang terintegrasi (dibuat tahun 2007)	ROS 2 memiliki fitur keamanan bawaan, termasuk otentikasi, enkripsi, dan kontrol akses
Pemeliharaan jangka panjang	ROS memiliki siklus hidup yang lebih tua dan saat ini berada di fase pemeliharaan. Meskipun masih banyak digunakan, pengembang yang baru memulai mungkin menemukan keterbatasan dalam dukungan dan pembaruan	ROS 2 dirancang dengan mempertimbangkan pemeliharaan jangka panjang, dengan pembaruan reguler dan dukungan yang lebih baik

c) Simulasi robotik tentu sangat penting dalam pengembangan robot karena memungkinkan pengguna untuk menguji, memvalidasi, dan mengoptimalkan desain robotik dalam lingkungan *virtual* sebelum membangun robot fisik dengan memberikan banyak keuntungan, seperti **pengujian yang tidak akan berdampak (beresiko) pada komponen fisik** (Komponen fisik, seperti sensor, aktuator, dan kamera dapat diuji oleh pengguna secara *virtual*), **menghemat biaya anggaran** (Dengan melakukan simulasi robotic terlebih dahulu, pengguna pastinya dapat menghemat biaya karena tidak perlu mengeluarkan biaya lebih untuk pembelian komponen fisik sebagai uji coba karena pengguna dapat mencari masalah terlebih dahulu yang mungkin terjadi dengan melakukan simulasi ini), dan **menghemat waktu** (Simulasi robotik ini dapat membantu pengguna dalam menghemat waktu karena pengguna tidak perlu menghabiskan waktu banyak ketika ada kesalahan terjadi)

- Contoh kasus :

  Pengujian robot dengan fitur navigasi

  Pengguna dapat melakukan simulasi terlebih dahulu untuk melihat apakah algoritma navigasi pada robot sudah bekerja dengan baik ketika ada rintangan didepan atau ada sesuatu yang menghalangi jalan robot. Ketika simulasi sudah berhasil, pengguna tentu dapat menghemat biaya dan waktu dari melakukan simulasi ini.

d) Gazebo adalah simulator robot 3D yang memungkinkan simulasi fisik, visualisasi robot, dan lingkungan sekitarnya. Gazebo sering digunakan dalam pengembangan robot karena mendukung simulasi yang realistis, termasuk dinamika fisik, deteksi tabrakan, dan rendering sensor secara akurat. Pengguna menggunakan Gazebo karena meskipun hanya simulasi yang dilakukan secara *virtual*, tetapi kondisinya sudah sangat mendekati kondisi *real* seakan melakukan pengujian pada robot fisik.

Langkah – Langkah mengintegrasikan ROS dengan Gazebo :

1. Mengunduh ROS dan Gazebo

2. Membuat paket ROS (beri nama paket ROS (ex : my_simulations) → buat folder **peluncuran** dan **dunia** dalam paket my_simulations → buat file (my_world.launch) dalam folder **peluncuran** → masukkan konten pada file tersebut (klik alat -> IDE) → buat file (empty_world.world) dalam folder **dunia** → tambahkan konten pada file tersebut → paket ROS terbuat
3. Membuat dunia Gazebo menggunakan ROS (Klik **Simulasi** -> **Pilih File** , lalu pilih **my_world.launch.** Ini akan secara otomatis memuat versi web gazebo, yang disebut gzweb)

e) Sistem yang dapat memandu robot untuk melakukan perpindahan dari posisi awal hingga ke posisi akhir yang diinginkan. Selain itu, navigasi pada robot harus mampu beradaptasi dengan lingkungan yang berubah-ubah karena tentu saja robot yang diharapkan untuk bekerja tidak hanya menetap dalam lingkungan yang monoton/statik begitu saja tanpa ada perubahan, seperti sistem pada *autonomous driving car* yang harus selalu beradaptasi dengan jalan yang tidak selalu sama dan berubah – ubah. Untuk membantu navigasi pada robot tersebut diperlukan beberapa konsep, seperti **mapping** (proses di mana robot membangun peta lingkungan di sekitarnya) dan *localization* (lokalisasi)

Mapping menggunakan metode **SLAM** (*Simultaneous Localization and Mapping*) yang adalah metode yang digunakan untuk membangun peta sambil melakukan lokalisasi secara bersamaan dengan data dari sensor, seperti lidar (*light detection and ranging*) dan kamera

Localization menggunakan metode **AMCL** (*Adaptive Monte Carlo Localization*) yang adalah metode lokalisasi probabilistik yang sering digunakan dalam ROS. **AMCL** menggunakan filter partikel untuk memperkirakan posisi robot pada peta dengan membandingkan data sensor saat ini (misalnya data lidar) dengan peta yang sudah ada

Cara mengimplementasikan pada robot :

1. Menyiapkan simulasi robot, seperti TurtleBot di lingkungan simulasi dengan Gazebo (*roslaunch turtlebot3_gazebo turtlebot3_world.launch*)

2. Membangun Peta dengan SLAM
   (*roslaunch turtlebot3_slam turtlebot3_slam.launch*)
3. Melokalisasi Robot pada Peta
   (*roslaunch turtlebot3_navigation turtlebot3_navigation.launch*)

f) TF (*Transform*) adalah Paket Ros . Paket ini memungkinkan pengguna untuk terus melacak beberapa frame koordinat dari waktu ke waktu. Paket ini memungkinkan melakukan perhitungan dalam satu frame dan kemudian mengubahnya ke frame lain.

TF membantu dalam hal navigasi pada robot dengan mentransformasi frame global (peta) (bersifat global dan tetap (tidak berubah)) ke frame lokal (robot) (bersifat lokal dan terus berubah). **Map frame** adalah **sistem koordinat global** yang digunakan untuk mewakili peta lingkungan di mana robot bergerak. Peta ini bisa berupa hasil pemetaan menggunakan algoritma **SLAM** atau peta statis yang sudah ada sebelumnya. **Base_link frame** adalah sistem koordinat lokal yang terikat langsung pada robot. Frame ini bergerak bersama dengan robot, dan mewakili pusat geometris robot. Semua sensor dan aktuator robot biasanya dirujuk terhadap frame ini

Bagaimana TF membantu memastikan robot bergerak dengan benar dalam simulasi ?

- Ketika frame global (peta) sudah diberikan dan tugas robot adalah untuk pergi ke suatu lokasi yang ada dalam peta, tetapi frame lokal (robot) berbeda dengan frame global (peta) sehingga robot tidak mengetahui koordinatnya secara pasti. Dibutuhkanlah TF untuk mentransformasi dari satu frame koordinat ke frame lain, yaitu frame lokal yang dipahami oleh robot. Dengan begitu, robot dapat berjalan menuju lokasi tersebut.