a.

Robot Operating System (ROS) adalah kerangkat perangkat lunak yang digunakan untuk pengembangan aplikasi robotika. Salah satu kemampuan dari ROS untuk mengembangkan dan menguji modul-modul secara terpisah sebelum mengintegrasikannya, hal ini memudahkan untuk melakukan uji coba masing-masing komponen sebelum mengintegrasikannya.

Setiap modul atau komponen dalam ROS disebut sebagai "node", yang dapat berkomunikasi satu sama lain melalui "topics", "services", atau "actions". ROS juga menyediakan berbagai tools seperti RViz untuk visualisasi, Gazebo untuk simulasi, dan berbagai package yang mendukung berbagai jenis sensor dan aktuator. Masing-masing komponen kemudian dapat digabungkan dengan sebuah nodes yang dihubungkan ke sebuah central hub. Hal ini membuat ROS sangat sentral dalam menghubungkan, mengontrol, dan mengintegrasikan komponen robot secara efisien.

Selain itu, ROS juga berkembang dengan pesat hingga pada tahun 2014 NASA mengumumkan robot yang digunakan di International Space Station, Robotnaut 2, menggunakan sistem ROS. Tidak hanya itu, Microsoft juga melakukan porting sistem ROS yang kemudian diikuti oleh Amazon dengan mengeluarkan Robomaker pada November 2018.

Sumber :
https://www.ros.org/
https://www.nasa.gov/directorates/heo/scan/engineering/technology/ros.html
https://aws.amazon.com/robomaker/

b.

Kelebihan terbesar ROS 2 dibandingkan ROS adalah penggunaan banyak nodes yang dapat dilakukan secara bersamaan dan juga penggunaan Data Distributed Services (DDS) membuat ROS 2 menjadi lebih superior dalam real-time processing dibandingkan ROS. ROS 2 juga mengalami improvement dibandingkan ROS, yaitu seperti autentikasi, enkripsi, dan control melalui DDS membuat ROS 2 jauh lebih aman dibandingkan ROS. ROS 2 juga lebih dapat diandalkan dibandingkan ROS 2, dengan menerapkan sistem yang terdistribusi ROS 2 meningkatkan keandalan perangkat. Selain itu, DDS juga meningkatkan performa dari ROS 2 sehingga lebih efisien dalam menjalankan aplikasi dengan banyak node.

Sumber :
https://docs.ros.org/en/foxy/index.html
https://www.model-prime.com/blog/ros-1-vs-ros-2-what-are-the-biggest-differences#:~:text=ROS%201%20uses%20the%20ROS,of%20service%20(QoS)%20parameters.

c.

Simulasi diperlukan untuk mengetahui secara aman, cepat, biaya murah, dan lebih mendalam tentang bagaimana robot akan di design dan dikontrol untuk penggunaan aman dan performa yang baik. Dalam simulasi, dapat juga dengan mudah menciptakan berbagai kondisi dan skenario yang mungkin sulit atau tidak mungkin direplikasi di dunia nyata, seperti kondisi cuaca ekstrem, rintangan yang kompleks, atau situasi darurat.

Salah satu contoh penerapan simulasi adalah dalam lini produksi di pabrik. Simulasi robot di lini produksi memungkinkan perancangan dan pengujian tata letak robot, algoritma pengendalian, dan interaksi antar robot sebelum diimplementasikan secara fisik di pabrik, memastikan operasi yang lancar dan efisien.

Sumber :
https://www.sciencedirect.com/science/article/pii/S0921889019301433
https://blogs.nvidia.com/blog/what-is-robotics-simulation/

d.

Gazebo adalah 3D dynamic simulator yang memiliki kemampuan untuk secara efisien melakukan simulasi kompleks dengan simulasi fisik dengan ketelitian yang lebih tinggi, sensor yang lebih banyak, dan tampilan untuk pengguna dan program. Gazebo mendukung simulasi yang kompleks dengan kemampuan fisik yang realistis, berbagai sensor seperti kamera, laser, dan lidar, serta menyediakan antarmuka pengguna yang interaktif untuk memvisualisasikan simulasi.

Cara Integrasi ROS dengan Gazebo :

1. Membuat Proyek Baru di ROS Development Studio dengan membuat workspace ROS baru dan menambahkan package yang diperlukan untuk simulasi Gazebo.
2. Menjalankan Gazebo melalui ROS dengan menggunakan perintah ROS seperti roslaunch, Gazebo dapat diluncurkan bersama dengan node ROS yang diperlukan.
3. Menghubungkan Node ROS dengan Gazebo**.** Node ROS dapat berkomunikasi dengan simulasi Gazebo melalui *topics*, *services*, atau *actions*, memungkinkan pengendalian robot dan pengumpulan data sensor secara real-time selama simulasi.
4. Menggunakan Plugins yang disediakan oleh Gazebo yang memungkinkan integrasi lebih dalam dengan ROS, seperti plugin sensor, plugin kontrol, dan plugin visualisasi.

Sumber :
https://docs.ros.org/en/foxy/index.html
https://classic.gazebosim.org/tutorials?tut=ros_overview

E.

Navigasi robot adalah bagaimana robot dapat berpindah dari satu titik ke titik yang lain tanpa menabrak apapun selama perjalanan. Proses ini melibatkan beberapa langkah kunci, termasuk pemetaan (*mapping*) dan lokalisasi (*localization*).

Mapping adalah proses pembuatan representasi lingkungan sekitar robot. Ini biasanya dilakukan dengan memanfaatkan data dari sensor seperti laser scanner atau lidar untuk mendeteksi hambatan dan fitur lingkungan. Hasil dari pemetaan adalah peta yang memvisualisasikan lokasi hambatan, dinding, dan objek lainnya di lingkungan.

Localization (Lokalisasi) dilakukan setelah peta dibuat, dimana robot perlu mengetahui posisinya dalam peta tersebut. Lokalisasi melibatkan menentukan posisi dan orientasi robot secara akurat dalam peta yang telah dibuat. Algoritma seperti AMCL (Adaptive Monte Carlo Localization) sering digunakan untuk tujuan ini, memanfaatkan data sensor untuk menyesuaikan posisi robot dalam peta.

Sumber :
https://docs.ros.org/en/foxy/index.html

f.

Transform adalah sistem yang melakukan pengecekan terhadap seluruh frame pada robot dengan melakukan mengekspresikan posisi frame satu relatif terhadap frame lainnya. Transform dalam ROS merujuk pada sistem koordinat yang digunakan untuk merepresentasikan posisi dan orientasi objek dalam ruang 3D yang memungkinkan pengembang untuk melacak dan mengubah posisi berbagai frame koordinat secara dinamis selama runtime.

Pada rover yang digunakan di Mars, transform digunakan untuk melacak posisi rover relatif terhadap posisi awalnya (map frame) serta posisi sensor dan aktuator lainnya (base_link, camera, dll.). Dengan demikian, rover dapat menentukan arah gerakannya, menghindari rintangan, dan melakukan tugas-tugas spesifik dengan akurasi tinggi.

Sumber :
https://docs.ros.org/en/foxy/index.html
https://foxglove.dev/blog/understanding-ros-transforms