Dasar Teori

1. Webots

Webots adalah alat simulasi yang sangat berguna dalam merancang dan menguji perilaku robot di lingkungan maya. Dengan menggunakan Webots, saya dapat dengan mudah memodelkan berbagai jenis robot, mulai dari yang menggunakan roda, berjalan, hingga robot pengendali dan jenis lainnya. Saya memiliki fleksibilitas untuk merancang struktur fisik robot sesuai kebutuhan, mengatur parameter kinematika dan dinamika, serta mengimplementasikan algoritma kontrol yang dibutuhkan. Selain itu, Webots juga menyediakan berbagai sensor dan aktuator virtual yang dapat saya integrasikan dalam simulasi robotika. Dengan menggunakan Webots, saya dapat menguji dan mengoptimalkan performa robot sebelum melakukan implementasi fisik, yang akan membantu meningkatkan efisiensi dan keandalan sistem robotic yang saya kerjakan.

2. Fungsi Webots

Webots menawarkan berbagai fitur yang sangat berharga dalam pengembangan dan pengujian robot:

Simulasi Robotika: Dengan menggunakan Webots, saya dapat membuat model virtual dari robot dan lingkungan sekitarnya, serta melakukan pengujian terhadap perilaku dan performa robot dalam simulasi. Ini memungkinkan saya untuk melakukan eksperimen, mengoptimalkan algoritma kontrol, dan memvalidasi desain robot sebelum melakukan implementasi di dunia nyata. Webots juga menyediakan beragam sensor dan aktuator virtual yang dapat diintegrasikan dengan model robot, sehingga simulasi yang dihasilkan menjadi lebih realistis dan mendekati kondisi dunia nyata. Pengembangan dan Pemrograman Robot: Webots memberikan fasilitas untuk mengembangkan kode perangkat lunak robot dan menguji fungsionalitasnya dalam lingkungan simulasi. Dalam Webots, saya dapat merancang struktur fisik robot, mengatur parameter kinematik dan dinamik, serta mengimplementasikan algoritma kontrol yang diperlukan. Hal ini memungkinkan saya untuk memahami dan memvalidasi perilaku robot sebelum melakukan implementasi di dunia nyata. Dengan Webots, saya dapat dengan lebih efisien mengembangkan kode robot yang dapat bekerja dengan baik saat dihadapkan pada situasi nyata.

3. ROS – Robotics Operation System

ROS merupakan sebuah kerangka kerja perangkat lunak open-source yang secara luas dimanfaatkan dalam pengembangan robotika. Dengan menyediakan lingkungan terstruktur dan modular, ROS memfasilitasi integrasi komponen perangkat keras dan perangkat lunak dalam sistem robotika. Sebagai seorang Robotics Engineer dengan tingkat pendidikan PhD, saya menghargai fitur-fitur unggulan yang dimiliki ROS. Salah satunya adalah kemampuannya dalam mengelola komunikasi antara node-node yang beroperasi secara mandiri. Melalui penggunaan model publikasi-langganan (publish-subscribe), ROS memungkinkan node-node tersebut untuk berkomunikasi melalui penggunaan topik-topik, layanan-layanan, dan aksi-aksi yang ditentukan. Dengan adanya ROS, saya dapat dengan lebih efektif dan terstruktur

mengintegrasikan berbagai komponen dalam sistem robotika, serta memfasilitasi komunikasi dan kolaborasi antar-node dengan efisiensi yang tinggi.

4. Fungsi ROS

Dalam konteks pengembangan sebagai seorang Robotics Engineer, ROS memberikan manfaat utama dalam dua aspek penting:

Dalam komunikasi dan integrasi, ROS menyediakan infrastruktur yang kuat untuk menghubungkan berbagai komponen perangkat keras dan perangkat lunak dalam sistem robotika. Melalui model publikasi-langganan (publish-subscribe), terjadi pertukaran data dan perintah antara sensor, aktuator, dan algoritma kontrol dalam sistem robotika yang kompleks.

Selain itu, ROS menyediakan kerangka kerja perangkat lunak yang terstruktur untuk mengorganisir dan mengelola kode robotika. Dengan ROS, tercipta arsitektur perangkat lunak yang modular dan berorientasi layanan. Hal ini memungkinkan reusabilitas kode yang tinggi dan mempermudah pengembangan kolaboratif dalam tim. Selain itu, tersedia perpustakaan perangkat lunak yang kaya dan beragam di dalam ROS, seperti algoritma navigasi, pemrosesan citra, manipulasi objek, dan lainnya, yang dapat digunakan untuk memperluas fungsionalitas robot.

Analisis & Hasil Code

Berdasarkan semua codingan yang ada, robot E-Puck dapat melakukan hal berikut:
1. Inisialisasi Perangkat:

Webots API digunakan untuk menginisialisasi perangkat yang digunakan oleh robot, seperti sensor jarak, sensor tanah, LED, dan motor.

Perangkat seperti sensor jarak dan sensor tanah diaktifkan menggunakan fungsi wb_distance_sensor_enable().

Motor kiri dan motor kanan diatur menggunakan fungsi wb_motor_set_position() dan wb_motor_set_velocity().

2 Sensor dan Aktuator:

Nilai-nilai sensor jarak dan sensor tanah diperoleh melalui fungsi wb_distance_sensor_get_value() dan disimpan dalam array distance_sensors_values dan ground_sensors_values.

Fungsi cliff_detected() digunakan untuk memeriksa deteksi rintangan oleh sensor tanah.

Nilai-nilai LED diatur menggunakan fungsi wb_led_set().

Kecepatan motor kiri dan motor kanan diatur melalui array speeds, yang dihitung berdasarkan nilai sensor jarak dengan memperhitungkan bobot dan offset tertentu. Nilai kecepatan tersebut kemudian disesuaikan dengan batasan maksimum dan minimum menggunakan kondisi if-else.

3. Loop Utama:

Loop utama berjalan terus-menerus menggunakan while (true) untuk menjalankan kontrol robot secara terus-menerus.

Setiap iterasi loop, nilai sensor dan aktuator diperbarui menggunakan fungsi-fungsi seperti reset_actuator_values(), get_sensor_input(), blink_leds(), run_braitenberg(), dan set_actuators().

Fungsi step() digunakan untuk menggerakkan simulasi robot melalui pemanggilan wb_robot_step().

4. Tindakan Robot:

Jika ada rintangan yang terdeteksi oleh sensor tanah (cliff_detected()), robot akan bergerak mundur selama 0.2 detik dan kemudian berbelok ke kiri selama 0.2 detik.

Jika tidak ada rintangan, robot akan mengikuti metode Braitenberg yang telah diimplementasikan dalam fungsi run_braitenberg().

Kesimpulan

Dalam laporan teknis ini, kami telah melakukan analisis dan implementasi kontrol gerakan pada robot E-Puck menggunakan platform Webots dan mengintegrasikannya dengan ROS. Melalui kode yang disediakan, kami berhasil mengontrol kecepatan roda kiri dan kanan sehingga robot E-Puck dapat melakukan gerakan maju, mundur, berbelok, dan berputar di tempat. Kami juga mengintegrasikan robot E-Puck dengan ROS menggunakan node "e_puck_manager", yang memungkinkan pengendalian robot melalui perintah kecepatan yang diterima melalui topik "/cmd_vel". Selain itu, kami memberikan kontrol waktu gerakan dengan mengubah arah gerakan roda kanan setelah periode waktu tertentu. Dengan demikian, melalui implementasi ini, kami berhasil menunjukkan kemampuan robot E-Puck untuk melakukan gerakan terkontrol dan berinteraksi dengan lingkungan melalui integrasi dengan ROS.

Daftar Pustaka

[1]. Joseph, L., & Cacace, J. (2021). Mastering ROS for Robotic Programming (3rd ed.). Packt Publishing