

NAMA: Rega Candra Kirana

NIM: 1103228243

TKX 46-01

UAS

CHAPTER 1

Dalam bab ini, kita mempelajari pengenalan ROS, termasuk alasan memilih ROS sebagai sistem operasi utama robot, level file sistem ROS, tingkat grafis komputasi, dan level komunitas ROS. Alasan utama menggunakan ROS adalah karena ROS adalah yang terbaru dan paling potensial dalam jenisnya. Selain itu, terdapat banyak alat dan fungsi yang mempermudah visualisasi, coding, dan simulasi. ROS juga mendukung sensor kelas atas. ROS memiliki level file sistem yang berbeda, yaitu metapackage, package, dan komponen-komponen seperti package manifest, messages, services, codes, dan misc. Package adalah elemen sentral dalam program ROS yang digunakan untuk membuat paket ROS. Package Manifest mencakup segala hal tentang package, termasuk penulis, lisensi, dependensi, flag kompilasi, dan lainnya. Metapackage biasanya berisi paket virtual tanpa kode atau file seperti biasanya.

Ada beberapa komponen komputasi grafis seperti Nodes, master, parameter server, messages, topics, services, dan bags. Nodes adalah proses yang menjalankan komputasi. Master menyediakan registrasi nama dan proses pencarian untuk nodes lainnya. Parameter server digunakan untuk menyimpan data di lokasi sentral. Topics adalah setiap pesan di ROS yang dikirimkan melalui bus.

CHAPTER 2

Fase pertama dalam pembuatan robot melibatkan desain dan pemodelan. Kita dapat merancang dan memodelkan robot menggunakan alat CAD seperti Autodesk Fusion 360, SolidWorks, Blender, dan lainnya. Salah satu tujuan utama pemodelan robot adalah simulasi. Alat simulasi robotik dapat memeriksa cacat kritis dalam desain robot dan memastikan bahwa robot akan berfungsi sebelum masuk ke fase produksi. Dalam bab ini, kita akan membahas proses desain dua robot: satu manipulator dengan tujuh derajat kebebasan (DOF) dan satu robot dengan penggerak diferensial. Pada bab-bab selanjutnya, kita akan mencoba simulasi, cara membangun perangkat keras nyata, dan interfacing dengan ROS. Jika kita berencana membuat model 3D robot dan mensimulasikannya menggunakan ROS, kita perlu mempelajari beberapa paket ROS yang dapat membantu dalam desain robot.

Membuat model robot di ROS penting untuk berbagai alasan, seperti mensimulasikan dan mengontrol robot, memvisualisasikannya, atau menggunakan alat ROS untuk mendapatkan informasi mengenai struktur dan kinematika robot. ROS menyediakan beberapa paket untuk mendesain dan membuat model robot, seperti urdf, kdl_parser, robot_state_publisher, dan collada_urdf. Paket-paket ini membantu kita membuat deskripsi model 3D robot dengan karakteristik yang sesuai dengan perangkat keras sebenarnya.

CHAPTER 3

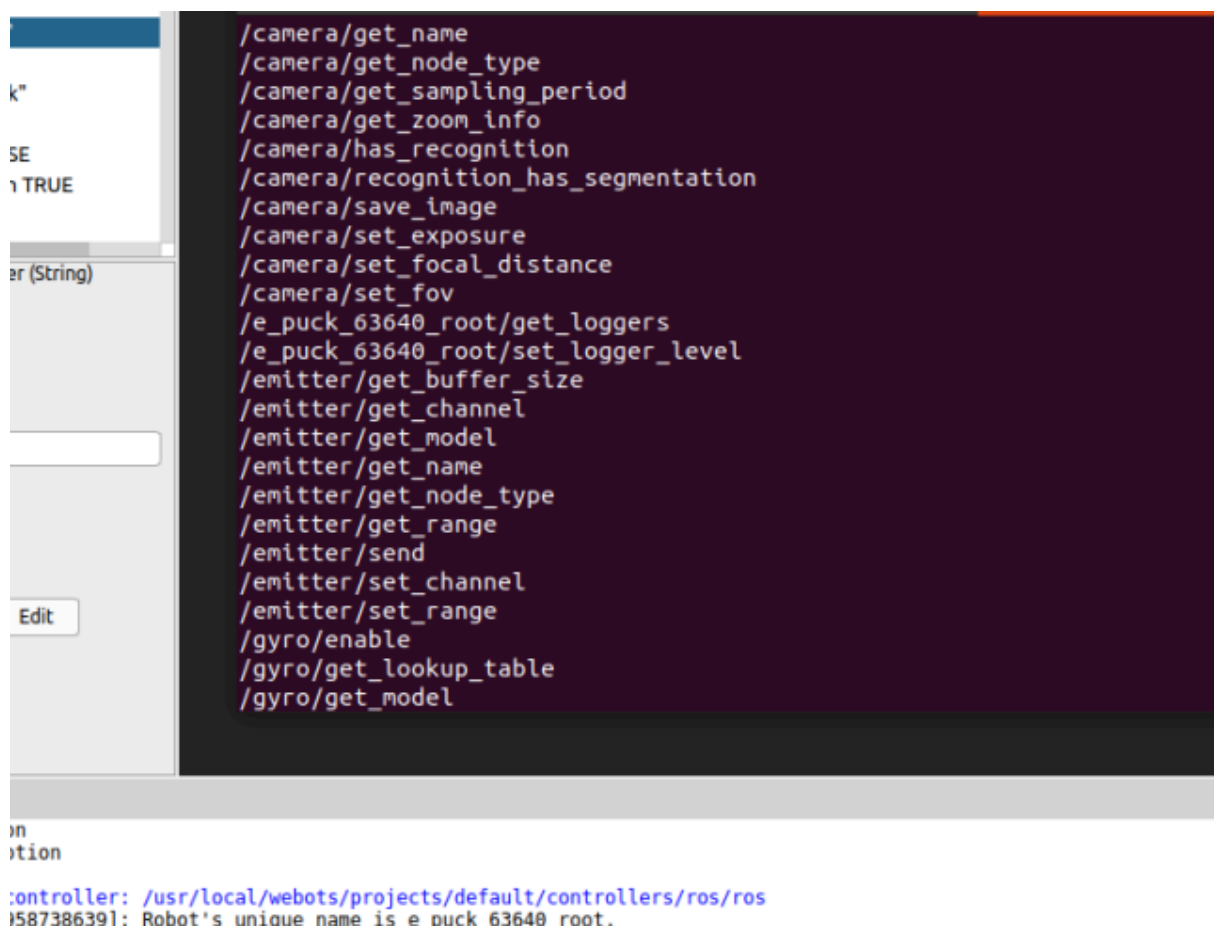
Pada bab ini, mempelajari cara menggunakan ROS untuk aplikasi pemodelan 3D. Mempelajari apa itu Unified Robot Description Format (URDF) sebagai file model robot yang sudah dibuat. Tetapi sayangnya tidak dapat melakukan percobaan tersebut karena GitHub tidak dapat diakses. Tapi kita masih dapat melihat di buku tersebut untuk cara membuat robot dengan kode yang sudah disediakan. URDF adalah file yang digunakan ketika kita membuat model robot di RVIZ. Setelah kita buat, kita dapat memvisualisasikannya di aplikasi ROS tersebut, dengan RVIZ. Format URDF bisa dipelajari dahulu

ketika membuat model robot. Tetapi ada beberapa kekurangannya diantaranya memiliki keterbatasan dalam kemudahan si pengguna, kemampuan untuk digunakan kembali, modularitas, dan kemudahan pemrograman. Oleh sebab itu ada format file lain untuk model robot, yaitu xacro. Xacro merupakan solusi dari beberapa kekurangan yang terdapat di URDF, untuk menyederhanakan URDF dan memudahkan dalam pembuatan pemrograman. Format xacro dapat diubah menjadi URDF jika diperlukan.

CHAPTER 4

Di bab ini, kita akan melihat beberapa konsep lanjutan dalam ROS, seperti ROS pluginlib, nodelets, dan plugin Gazebo. Kita akan membahas fungsi dan aplikasi masing-masing konsep dan melihat contoh untuk mendemonstrasikan cara kerjanya. Kita telah menggunakan plugin Gazebo di bab-bab sebelumnya untuk mendapatkan perilaku sensor dan robot di dalam simulator Gazebo. Di bab ini, kita akan melihat cara membuatnya. Kita juga akan membahas bentuk lain dari ROS node, yang disebut ROS nodelets. Fitur-fitur ini dalam ROS diimplementasikan menggunakan arsitektur plugin yang disebut pluginlib.

CHAPTER 5



```
/camera/get_name
/camera/get_node_type
/camera/get_sampling_period
/camera/get_zoom_info
/camera/has_recognition
/camera/recognition_has_segmentation
/camera/save_image
/camera/set_exposure
/camera/set_focal_distance
/camera/set_fov
/e_puck_63640_root/get_loggers
/e_puck_63640_root/set_logger_level
/emitter/get_buffer_size
/emitter/get_channel
/emitter/get_model
/emitter/get_name
/emitter/get_node_type
/emitter/get_range
/emitter/send
/emitter/set_channel
/emitter/set_range
/gyro/enable
/gyro/get_lookup_table
/gyro/get_model

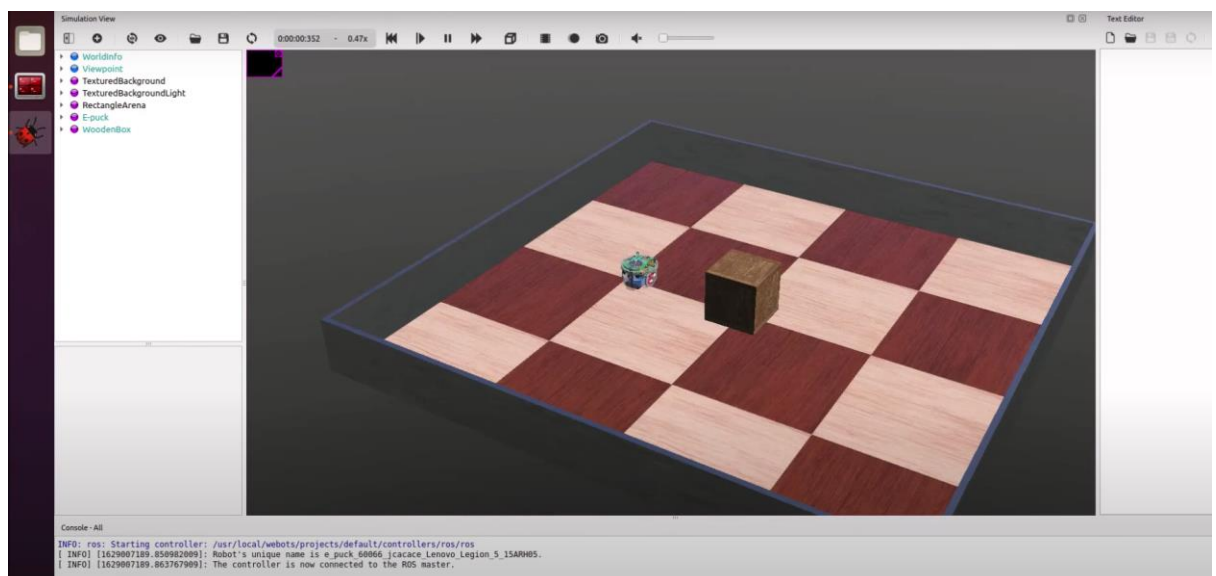
controller: /usr/local/webots/projects/default/controllers/ros/ros
1587386391: Robot's unique name is e_puck_63640 root.
```

```

1 #include <webots/Robot.hpp>
2 #include <webots/Motor.hpp>
3 #include <webots/DistanceSensor.hpp>
4
5
6 #define MAX_SPEED 6.28
7
8 //64 Milliseconds
9 #define TIME_STEP 64
10
11 using namespace webots;
12
13 int main(int argc, char **argv) {
14   Robot *robot = new Robot();
15
16   // get a handler to the motors and set target position
17   Motor *leftMotor = robot->getMotor("left wheel motor");
18   Motor *rightMotor = robot->getMotor("right wheel motor");
19   leftMotor->setPosition(INFINITY);
20   rightMotor->setPosition(INFINITY);
21   double t=0.0;
22
23   double l_direction=1.0;
24   double r_direction=1.0;
25
26   while(true) {
27
28     leftMotor->setVelocity( l_direction*MAX_SPEED*0.1);
29     rightMotor->setVelocity( r_direction*MAX_SPEED*0.1);
30     robot->step(TIME_STEP);
31     t+= TIME_STEP;
32
33     printf("t: %f\n", t);
34     if ( t > 2000 ) {
35       r_direction*=-1.0;
36     }
37     if( t > 4000) {
38       r_direction = 1.0;
39       t = 0.0;
40     }
41   }
42 }
43
44 delete robot;
45

```

Webots untuk menggunakan ROS. Pada buku tersebut dijelaskan bahwa cara untuk menginstall Webots pada ubuntu dan menggunakannya dengan ROS.



CHAPTER 6

Node `move_group` berfungsi sebagai integrator yang menghubungkan fungsionalitas melalui plugin untuk pemecah kinematika dan perencanaan gerak. Setelah perencanaan gerak selesai, lintasan yang dihasilkan dikirim ke pengontrol melalui antarmuka `FollowJointTrajectoryAction` untuk dieksekusi pada robot atau simulator nyata. MoveIt! dapat diintegrasikan dengan RViz GUI dan Gazebo untuk mengendalikan lengan robot. Perencanaan gerak adalah teknik untuk menemukan jalur optimal bagi robot agar dapat bergerak dari posisi awal ke posisi tujuan tanpa menabrak rintangan. Ini memerlukan deskripsi robot dan geometri lingkungan, yang dapat direpresentasikan menggunakan file URDF. Kendala dapat ditambahkan ke perencanaan gerak, seperti posisi, orientasi, visibilitas, batas sendi, atau kendala khusus yang ditentukan pengguna. Adaptor permintaan perencanaan gerak melakukan praproses dan pascaproses pada permintaan dan respons perencanaan gerak. MoveIt! menawarkan fleksibilitas untuk mengganti algoritma IK dengan menggunakan plugin dan sudah mengintegrasikan FK serta perhitungan Jacobian. Pemeriksaan tabrakan di MoveIt! dilakukan menggunakan objek `CollisionWorld` dengan dukungan untuk berbagai jenis objek. Matriks Tabrakan yang Diizinkan (ACM) digunakan untuk mengoptimalkan komputasi pemeriksaan tabrakan. Untuk menghubungkan lengan robot dengan MoveIt!, node `move_group` memerlukan URDF, SRDF, file konfigurasi, dan topik status sendi, serta informasi TF. Setup Assistant di MoveIt! dapat digunakan untuk menghasilkan semua elemen ini untuk paket konfigurasi MoveIt!.

