Technical Report of Robot Autonomy : Simple Jetbot

Nama : Harits Maulana Muzakki
NIM : 1103204166

ROS 2

ROS 2 (Robot Operating System 2) adalah sebuah middleware robotika yang menyediakan seperangkat alat dan pustaka untuk mengembangkan, membangun, dan menjalankan aplikasi robotik. ROS 2 adalah penerus ROS 1 dan dirilis pada tahun 2015. ROS 2 bertujuan untuk mengatasi beberapa keterbatasan ROS 1, seperti kinerja real-time, dukungan untuk platform yang berbeda, dan skalabilitas untuk sistem skala besar.

Arsitektur ROS 2 didasarkan pada seperangkat node yang longgar-coupled yang berkomunikasi satu sama lain menggunakan model komunikasi publish-subscribe atau client-server. Komunikasi dilakukan melalui seperangkat topik, layanan, dan aksi. Berikut adalah gambaran singkat dari setiap konsep ini:

1. Node: Node adalah program eksekusi yang melakukan tugas tertentu. Mereka dapat berkomunikasi dengan node lain dengan mempublikasikan pesan ke topik atau memanggil layanan.
2. Topik: Topik adalah bus bernama di mana node bertukar pesan. Node dapat mempublikasikan pesan ke topik atau berlangganan ke topik untuk menerima pesan.
3. Layanan: Layanan adalah entitas bernama yang menyediakan pola komunikasi permintaan-respons antara node. Node dapat memanggil layanan untuk meminta beberapa fungsionalitas dan menunggu respons.
4. Aksi: Aksi serupa dengan layanan, tetapi mereka menyediakan pola komunikasi yang lebih fleksibel di mana node dapat mengirimkan permintaan untuk memulai

aksi, menerima umpan balik selama pelaksanaan aksi, dan mendapatkan hasil akhir ketika aksi selesai.

Arsitektur ROS 2 juga mencakup seperangkat komponen inti yang menyediakan infrastruktur untuk komunikasi, penemuan, dan manajemen sistem. Komponen-komponen ini adalah:

1. Middleware: Middleware menyediakan protokol komunikasi dan transportasi yang mendasar untuk ROS 2. Ini mengabstraksi detail komunikasi dari node dan memungkinkan mereka untuk berkomunikasi satu sama lain terlepas dari bahasa pemrograman atau sistem operasi mereka.
2. RCL: ROS 2 Client Libraries (RCL) menyediakan API untuk node berinteraksi dengan middleware dan sistem runtime ROS 2.
3. DDS: Data Distribution Service (DDS) adalah implementasi middleware yang digunakan oleh ROS 2. Ini menyediakan cara standar untuk bertukar data antara node dan memastikan komunikasi yang andal dan efisien.
4. Penemuan: Penemuan adalah proses untuk menemukan dan mengidentifikasi node dan topik, layanan, dan aksi yang tersedia di dalam sistem. ROS 2 menggunakan mekanisme penemuan terdesentralisasi berdasarkan DDS.
5. Manajemen sistem: Manajemen sistem menyediakan alat dan layanan untuk memantau dan mengelola status node dan komunikasi di dalam sistem. Ini termasuk alat untuk logging, debugging, dan profiling node dan middleware.

Di ROS 2, node adalah proses individu yang melakukan tugas-tugas tertentu dalam sistem robotik. Node dirancang agar longgar terkait, artinya mereka dapat berkomunikasi satu sama lain melalui pesan tanpa harus mengetahui apa pun tentang implementasi yang mendasari dari node lain.

Node berkomunikasi satu sama lain menggunakan pola pesan publish/subscribe atau pola klien/server, tergantung pada jenis komunikasi yang diperlukan. Mereka dapat mempublish pesan ke topik, yang merupakan bus bernama tempat pesan ditukar antara node, atau memanggil layanan, yang merupakan entitas bernama yang menyediakan komunikasi permintaan/respon antara node. Node juga dapat

menggunakan aksi, yang menyediakan pola komunikasi yang lebih fleksibel di mana node dapat mengirim permintaan untuk memulai aksi, menerima umpan balik selama pelaksanaan aksi, dan mendapatkan hasil akhir ketika aksi selesai.

Berikut adalah contoh program Python sederhana yang membuat node ROS 2 dan mempublish pesan ke topik:

```
import rclpy
from std_msgs.msg import String

def talker():
    # Inisialisasi node ROS 2
    rclpy.init()
    node = rclpy.create_node('talker')

    # Buat penerbit untuk topik "chatter"
    publisher = node.create_publisher(String, 'chatter', 10)

    # Publikasikan pesan setiap detik
    msg = String()
    msg.data = 'Hello, world!'
    while rclpy.ok():
        publisher.publish(msg)
        node.get_logger().info('Published: "%s"' % msg.data)
        rclpy.sleep(1)

    # Matikan node
    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    talker()
```

Dalam contoh ini, program membuat node ROS 2 bernama "talker" dan membuat penerbit untuk topik "chatter". Program kemudian mempublish pesan ke topik setiap detik, mencatat setiap pesan yang dipublish.

Robot yang disimulasikan kali ini adalah Jetbot sederhana yang dapat bergerak ke arah tertentu. Berikut kodingannya :

```
# Import the necessary modules
from controller import Robot, Keyboard

# Create a robot instance
robot = Robot()

# Get the motor devices
left_motor = robot.getDevice('left wheel motor')
right_motor = robot.getDevice('right wheel motor')

# Set the target position of the motors
left_motor.setPosition(float('inf'))
right_motor.setPosition(float('inf'))

# Set the velocity of the motors
left_motor.setVelocity(0.0)
right_motor.setVelocity(0.0)

# Create a keyboard instance
keyboard = Keyboard()

# Enable the keyboard
keyboard.enable(64)

# Set the robot's initial direction
```

```
direction = 0.0

# Run the simulation
while robot.step(64) != -1:
    # Get the pressed key
    key = keyboard.getKey()

    # Update the robot's direction based on the pressed key
    if key == Keyboard.UP:
        left_motor.setVelocity(3.0)
        right_motor.setVelocity(3.0)
    elif key == Keyboard.DOWN:
        left_motor.setVelocity(-3.0)
        right_motor.setVelocity(-3.0)
    elif key == Keyboard.LEFT:
        direction += 0.1
        left_motor.setVelocity(2.0)
        right_motor.setVelocity(-2.0)
    elif key == Keyboard.RIGHT:
        direction -= 0.1
        left_motor.setVelocity(-2.0)
        right_motor.setVelocity(2.0)
    else:
        left_motor.setVelocity(0.0)
        right_motor.setVelocity(0.0)

    # Set the robot's direction
    robot.setOrientation([0, 1, 0, direction])
```

Dengan kodingan tersebut, robot akan dikontrol oleh keyboard yang juga dikendalikan oleh kita sendiri.