Nama: Valrama Wardhana Hariwidjaja
NIM: 1103204152
UAS ROBOTIKA

Berikut adalah sample code:

```
import rospy
from std_msgs.msg import String
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan
import tf
from visualization_msgs.msg import Marker

def publisher_node():
    rospy.init_node('minimal_publisher', anonymous=True)
    pub = rospy.Publisher('codeuas', String, queue_size=10)
    rate = rospy.Rate(10)

    while not rospy.is_shutdown():
        message = "Valrama Wardhana Hariwidjaja was here"
        rospy.loginfo(message)
        pub.publish(message)
        rate.sleep()

def callback(data):
    rospy.loginfo("Received message: %s", data.data)

def subscriber_node():
    rospy.init_node('minimal_subscriber', anonymous=True)
    rospy.Subscriber('codeuas', String, callback)

def robot_simulator():
    rospy.init_node('robot_simulator', anonymous=True)
    pub = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
    rate = rospy.Rate(10)

    while not rospy.is_shutdown():
        # Generate control commands for the simulated robot
        cmd_vel = Twist()
        cmd_vel.linear.x = 0.2
        cmd_vel.angular.z = 0.1
        pub.publish(cmd_vel)
        rate.sleep()

def transform_listener():
    rospy.init_node('transform_listener')
    listener = tf.TransformListener()

    rate = rospy.Rate(10)

    while not rospy.is_shutdown():
        try:
            listener.waitForTransform('/frame1', '/frame2', rospy.Time(0), rospy.Duration(1.0))
            (trans, rot) = listener.lookupTransform('/frame1', '/frame2', rospy.Time(0))
            rospy.loginfo("Translation: x=%.2f, y=%.2f, z=%.2f", trans[0], trans[1], trans[2])
```

```
            rospy.loginfo("Rotation: x=%.2f, y=%.2f, z=%.2f, w=%.2f", rot[0], rot[1], rot[2], rot[3])

        except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
            pass

        rate.sleep()
def marker_publisher():
    rospy.init_node('marker_publisher', anonymous=True)
    pub = rospy.Publisher('codeuas_marker', Marker, queue_size=10)
    rate = rospy.Rate(1)  # 1 Hz

    while not rospy.is_shutdown():
        marker = Marker()
        # Set marker properties
        marker.header.frame_id = "map"  # Frame ID tempat marker ditampilkan
        marker.type = Marker.SPHERE  # Jenis marker (dalam contoh ini: bola)
        marker.action = Marker.ADD  # Aksi yang diambil (dalam contoh ini: menambahkan marker)
        marker.pose.position.x = 1.0  # Posisi X marker dalam koordinat dunia
        marker.pose.position.y = 2.0  # Posisi Y marker dalam koordinat dunia
        marker.pose.position.z = 0.5  # Posisi Z marker dalam koordinat dunia
        marker.pose.orientation.x = 0.0  # Orientasi X marker
        marker.pose.orientation.y = 0.0  # Orientasi Y marker
        marker.pose.orientation.z = 0.0  # Orientasi Z marker
        marker.pose.orientation.w = 1.0  # Orientasi W marker
        marker.scale.x = 0.2  # Skala X marker
        marker.scale.y = 0.2  # Skala Y marker
        marker.scale.z = 0.2  # Skala Z marker
        marker.color.a = 1.0  # Alpha (transparansi) marker (0.0 - 1.0)
        marker.color.r = 1.0  # Komponen merah warna marker (0.0 - 1.0)
        marker.color.g = 0.0  # Komponen hijau warna marker (0.0 - 1.0)
        marker.color.b = 0.0  # Komponen biru warna marker (0.0 - 1.0)
        pub.publish(marker)
        rate.sleep()

def process_lidar_data():
    rospy.init_node('lidar_processor')
    rospy.Subscriber('/scan', LaserScan, lidar_callback)

def lidar_callback(data):
    ranges = data.ranges  # Mendapatkan data jarak dari sensor LIDAR
    min_range = min(ranges)  # Mencari jarak terdekat
    rospy.loginfo("Minimum range: %.2f meters", min_range)

if __name__ == '__main__':
    try:
        publisher_node()
        subscriber_node()
        robot_simulator()
        transform_listener()
        marker_publisher()
        process_lidar_data()
        rospy.spin()

    except rospy.ROSInterruptException:
        pass
```

Technical Report:

1. Pendahuluan

Tujuan dari laporan teknis ini adalah untuk mendokumentasikan implementasi beberapa node ROS untuk berbagai fungsionalitas, termasuk penerbitan, langganan, simulasi robot, mendengarkan transformasi, penerbitan marker, dan pemrosesan data lidar. Node-node ROS ditulis dalam bahasa Python dan menggunakan berbagai paket ROS seperti rospy, std_msgs, geometry_msgs, sensor_msgs, tf, dan visualization_msgs.

2. Deskripsi Node

a. Node minimal_publisher:

Menginisialisasi node ROS dengan nama 'minimal_publisher'.
Menerbitkan pesan string dengan topik 'codeuas' dengan laju 10 Hz.
Pesan yang diterbitkan adalah "Valrama Wardhana Hariwidjaja was here".

b. Node minimal_subscriber:

Menginisialisasi node ROS dengan nama 'minimal_subscriber'.
Melanggani topik 'codeuas' dan mendefinisikan fungsi panggilan callback untuk menangani pesan yang diterima.
Ketika sebuah pesan diterima, mencatat data pesan tersebut.

c. Node robot_simulator:

Menginisialisasi node ROS dengan nama 'robot_simulator'.
Menerbitkan pesan Twist dengan topik '/cmd_vel' dengan laju 10 Hz.
Menghasilkan perintah kontrol untuk simulasi robot dengan kecepatan linear 0,2 m/s dan kecepatan angular 0,1 rad/s.

d. Node transform_listener:

Menginisialisasi node ROS dengan nama 'transform_listener'.
Menyiapkan transform listener untuk mendengarkan pembaruan transformasi antara '/frame1' dan '/frame2'.
Mendapatkan informasi translasi dan rotasi antara dua frame tersebut dan mencatat nilainya dengan laju 10 Hz.

e. Node marker_publisher:

Menginisialisasi node ROS dengan nama 'marker_publisher'.
Menerbitkan pesan Marker dengan topik 'codeuas_marker' dengan laju 1 Hz.
Mengkonfigurasi sebuah marker dengan properti-posisi, orientasi, skala, dan warna.
Marker ini mewakili sebuah bola dan ditampilkan dalam frame 'map'.

f. Node lidar_processor:

Menginisialisasi node ROS dengan nama 'lidar_processor'.
Melanggani topik '/scan' untuk menerima pesan LaserScan.
Mengimplementasikan fungsi panggilan lidar_callback untuk memproses data lidar.
Mencari nilai jarak minimum dari data lidar yang diterima dan mencatatnya.

3. Eksekusi dan Ketergantungan

Kode ini membutuhkan ketergantungan berikut untuk diinstal:

rospy: Pustaka Python ROS untuk fungsionalitas ROS

std_msgs: Paket ROS untuk pesan standar

geometry_msgs: Paket ROS untuk pesan geometri

sensor_msgs: Paket ROS untuk pesan sensor

tf: Paket ROS untuk transformasi koordinat

visualization_msgs: Paket ROS untuk pesan visualisasi

Pastikan semua ketergantungan diinstal dengan benar sebelum menjalankan kode ini. Eksekusi kode dapat dilakukan dengan menjalankan file Python yang berisi kode ini, dan memonitor keluaran konsol untuk menangkap pesan log atau kesalahan yang mungkin terjadi.

4. Kesimpulan

Dalam laporan teknis ini, telah dijelaskan implementasi beberapa node ROS yang berbeda untuk berbagai fungsionalitas, seperti penerbitan, langganan, simulasi robot, mendengarkan transformasi, penerbitan marker, dan pemrosesan data lidar. Setiap node memiliki peran dan tugasnya sendiri dalam ekosistem ROS. Kode ini dapat diperluas atau dimodifikasi lebih lanjut sesuai dengan kebutuhan aplikasi yang spesifik. Penting untuk memastikan instalasi ketergantungan yang tepat dan memantau keluaran konsol untuk menemukan masalah atau kesalahan yang mungkin terjadi