Introduction to Robotics Winter 2023		Computer Engineering Dept. Prof. Ahad Harati

Assignment #6

(مطالب زیر در آزمایشگاه ۹ قبلا ذکر شده‌اند)

به استفاده از داده های حسگرها برای تخمین تغییر در موقعیت، جهت گیری و سرعت ربات در طول زمان نسبت به یک نقطه odometry می‌گویند. (به عنوان مثال فریم‌ای که این نقطه مرکز آن است می‌تواند *odom* باشد و مرکز آن در$z=0$ $y=0$ ،$x=0$ باشد). اطلاعات odometry معمولاً از حسگرهایی مانند wheel encoder و IMU بدست می‌آیند. نهایتا تبدیل بین فریم ربات و فریم odom باید در tf ارسال شود تا بشود مکان ربات را در دنیا تقریب زد.

در ROS، فریم مختصاتی که برای odometry استفاده می شود، به عنوان فریم odom شناخته می شود. فریم odom نقطه ای در جهان است که ربات برای اولین بار شروع به حرکت می کند. موقعیت و جهت گیری یک ربات در فریم odom با گذشت زمان و مسافت کم دقت تر می شود زیرا حسگرهایی مانند IMU (که شتاب را اندازه می گیرند) و wheel encoders (که تعداد دفعات چرخش هر چرخ را اندازه می گیرند) بدون خطا نیستند. به عنوان مثال، تصور کنید ربات شما به دیوار برخورد کند. چرخ های آن ممکن است به طور مکرر بچرخند بدون اینکه ربات به جایی حرکت کند (ما به این لغزش چرخ می گوییم). در این حالت odometry چرخ فکر می‌کند که ربات مسافتی را طی کرده است، در حالی که درجا زده است؛ ما باید این نادرستی ها را مدیریت کنیم.

بنابراین نحوه ارتباط فریم‌ها به این صورت می‌شود:

1. `map => odom`
2. `odom => base_link (or base_footprint)`
3. `base_link => base_laser (sensor base frames)`

تبدیل اول معمولا توسط سیستم localization and mapping ارسال می‌شود (برای آزمایش می‌شود تبدیل دستی و استاتیک نیز داد). نبدیل دوم توسط سیستم odometry مثل IMU ارسال می‌شود. تبدیل سوم توسط robot_state_publisher و فایل URDF ربات ارسال می‌شود.

برای تمام تمرین‌های آینده و پروژه **فقط از یک workspace** استفاده کنید. در صورت استفاده از workspace های متفاوت به ازای تمرین‌های مختلف، نمره منفی داده می‌شود.

فعالیت ۱

جدید ترین تغییرات مخزن arashsm79/eddiebot-ros را در ws خود pull کنید. در این تغییرات، فایل create_base.urdf.xacro تمیز تر شده است و eddie با استفاده از DiffDrive کنترل می‌شود.
در مستندات gazebo صفحه مربوط به پلاگین VelocityControl که در تمرین قبل استفاده شد تا eddie را حرکت دهیم و پلاگین DiffDrive را پیدا کنید. (detail level را روی ۵ بذارید تا بتوانید راحت‌تر جست‌وجو کنید)

در یک تا دو پاراگراف فرق این دو پلاگین و نحوه استفاده از آن‌ها را بنویسید.

فعالیت ۲

سنسور LIDAR از اشعه نور برای پیدا کردن فاصله تا اجسام استفاده می‌کند و برای مثال برای درست کردن یک نقشه از محیط می‌تواند استفاده شود. برای آشنایی کلی با نحوه کارکرد این سنسور، این صفحه را مطالعه کنید (نیازی به انجام آن نیست). قیمت این سنسورها در بازار به نسبت سنسورهایی مثل کینکت بیشتر است. از این رو در بعضی از کاربردها می‌توان از دوربین عمقی کینکت به جای سنسور گران قیمت LIDAR استفاده کرد.

مخزن depthimage_to_laserscan را در ws خود کلون کنید:

```
git clone https://github.com/ros-perception/depthimage_to_laserscan.git -b ros2
```

سپس با rosdep وابستگی‌ها را نصب کنید. این نود داده depth_image را می‌گیرد، یک سطر از آن را انتخاب می‌کند و آن را به یک LaserScan تبدیل می‌کند. یک لانچ فایل بنویسید که پس از راه اندازی دنیا maze همراه با eddie در آن، تمام کارهای زیر را انجام می‌دهد (یادآوری: به نحوه اضافه کردن remapping در لانچ فایل مراجعه کنید):

1. برای تاپیک‌های زیر یک پل درست می‌کند:
    1. /model/eddiebot/tf و سپس آن را به tf بازنگاشت می‌کند. (دقت کنید که tf است نه /tf)
    2. /kinect_rgbd_camera/camera_info
    3. /kinect_rgbd_camera/depth_image
2. نود تبدیل کننده عکس عمقی به اسکن لیسر را راه می‌اندازد. باید تاپیک‌های depth_camera_info و depth را به درستی بازنگاشت کنید. تاپیک جدید مربوط به scan را echo کنید و محتویات هدر آن را ببینید. نام فریم‌ای که در هدر مشخص شده است چیست؟ این فریم بیانگر چیست و چه رابطه‌ای با نقطه‌هایی دارد که در پیام قرار دارند؟
3. Rviz2 را اجرا کنید و قسمت LaserScan را به آن اضافه کنید و تاپیک مربوطه را مشخص کنید. مقدار size را در LaserScan برروی 0.1 قرار دهید.
4. با استفاده از دستور

```
ros2 run tf2_ros static_transform_publisher "0" "0" "0" "0" "0" "0" "foo" "bar"
```

برای مثال می‌توانید یک تبدیل ایستا صفر از فریم پدر foo به فریم فرزند bar در tf ارسال کنید. برای تمام فریم‌های زیر تبدیل مناسب را انجام دهید تا زمانی که خطاهای آن‌ها در rviz از بین بروند:

1. "eddiebot/base_footprint" → "base_footprint"
2. "world" → "eddiebot/odom"
3. "camera_link" → "camera_depth_frame"

حال ماژول teleop را در gazebo اضافه کنید وeddie را تکان دهید. به نزدیکی دیوارهای مختلف رفته و از نمایش داده شدن آن‌ها در rviz توسط LaserScan اطمینان حاصل کنید و اسکرین شات بگیرید.

فعالیت ۳

با استفاده از

ros2 run tf2_tools view_frames

درخت tf را نمایش دهید و در حد یک پاراگراف در رابطه با درخت تولیدی توضیح دهید. درخت تولید شده را در فایل ZIP تمرین خود قرار دهید.

نکات:

- لطفا در لانچ فایل تا جای امکان از ExecuteProcess و توابع نظیر آن استفاده نکنید و شرح قسمت‌های مختلف مانند اجرای Nodeها را به درستی بنویسید.
- اگر مایل بودید حرکت چرخ‌ها را نیز در rviz مشاهده کنید، پارامتر use_sim_time را برای اجرای گزیبو و تمام پل‌ها و تبدیل‌های ایستایی که می‌سازید در حالت درست قرار دهید.

```
$ ros2 launch eddiebot_gazebo eddiebot_gz_sim.launch.py world:='maze'
use_sim_time:='true'

$ ros2 run ...  --ros-args -p use_sim_time:='True'
```

- در مخزن depthimage_to_laserscan حتما شاخه را روی ros2 قرار دهید و بعد README آن را بخوانید.
- بعد از تنظیم rviz برای بار اول، از داخل منو های آن می‌توانید کانفیگ خود را در یک فایل ذخیره کنید و در لانچ فایل برای اجرای rviz در دفعات بعد، کانفیگ ذخیره شده خود را به آن بدهید. برای مثال به پکیج eddiebot_rviz می‌توانید مراجعه کنید.
- دقت کنید مدل ربات حتما باید در rviz نیز وجود داشته باشد. ماژول‌های زیر باید در rviz وجود داشته باشند:
  - RobotModel
  - TF

تحویل

از نتیجه تمرین‌ها اسکرین‌شات گرفته و در یک فایل PDF با نام و نام خانوادگی و شماره دانشجویی قرار دهید و همراه با تمام کدها در یک فایل ZIP در سامانه آموزش مجازی دانشگاه ارسال کنید.