Introduction to Robotics	Ferdowsi University of Mashhad
Instructor: Arash Sal Moslehian	Computer Engineering Dept.

Exercise 0: Getting to know Eddie

Please watch the introductory videos to Eddie that are available on Moodle.

- 01-Eddie-hardware-and-assembly
- 02-Eddie-command-set
- 03-Eddie-ROS2-Packages
- 04-Eddie-code-review
- 05-Eddie-hazards

Please keep in mind that you can make changes to the code in `eddiebot-ros` packages as you need.

Exercise 1: Setting up the Kinect

Clone the kinect_ros2 in your repository. Follow the instructions to build the package and its dependencies. You need to compile and install libfreenect from its GitHub repository before building kinect_ros2. Read the instruction carefully. You need to install the required dependencies (both for the main libraries and examples) through `apt`. `freeglut3-dev` for example is a required library for examples. Make sure you set the following `cmake` flags along with the other flags that you may need. (you need to consult the wiki or do a Google search if you are facing errors when building)

```
$ cmake   -DBUILD_OPENNI2_DRIVER=ON \
          -DBUILD_EXAMPLES=ON\
          -DBUILD_PYTHON3=OFF\
          ..
$ make
$ sudo make install
```

Set up the udev rules as described in the README. Run a `freenect-glview` and make sure both depth and RGB sensor are working. You can now proceed with the build of kinect_ros2. Make sure you source your local workspace and then try to run the node:

```
ros2 run kinect_ros2 kinect_ros2_node
```

Open rviz and add displays for both the depth image and RGB image topics and make sure they are working.

Exercise 2: Add timestamp to RGB images

While `kinect_ros2_node` is running, open `rqt` and go to the topic monitor. Check the timestamps for `image_raw` and `depth/image_raw` topics. Do both of them have valid timestamps? Browse through the code of `kinect_ros2` and find the place where it assigns a timestamp to depth images. Make sure that RGB images also get a timestamp. Open `rqt` and check to see if your changes have worked!

Exercise 3: bringup the Robot

Pull in the latest changes from the eddiebot-ros repository. Make sure all the dependencies are installed using `rosdep` and then build all the packages.

Connect the USB port from Eddie to the laptop that is going to stay on the robot. If you're using `distrobox` run the following command from outside the box: (change `ttyUSB0` accordingly)

```
sudo chmod a+rw /dev/ttyUSB0
```

run the following launch file to bring up the robot:

```
ros2 launch eddiebot_bringup eddie.launch.yaml
```

Check out the output and make sure you don't see any errors.

On a separate terminal run:

```
ros2 launch eddiebot_nav eddiebot.launch.py
```

This should start the nodes for Kinect, odometry, vel_controller, some static transforms, and fake laser scan.

Spin up another terminal and run:

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args -r /cmd_vel:=/eddie/cmd_vel
```

You should see the instructions for how to use teleop. Reduce both linear and angular velocities to somewhere near 0.2. Now start running the robot using the keys. Try all the different ways you can control it using teleop.

Launch the `view_model` file from `eddiebot` rviz; make sure to also launch the robot description with it (by setting the proper arguments).

- Start proper static transforms until all the errors are gone.
- Add the `PointCloud` display in rviz and choose the proper topic. (you may need to change the QoS setting to `BestEffort` for example. Keep this in mind for the rest of the exercises). Try to move around infront of the kinect and experiment with the cloud.
- Turn off the display for point cloud. Include a display for LaserScan and visualize the scan line. Experiment with the scan line, its FOV, and range.

Make sure you read all the launch files carefully. Questions will be asked regarding them during the interview.

Exercise 4: Networking

One laptop is going to stay on Eddie and the other laptop is going to do teleoperation. For this, both laptops need to be on the same network. You can use your phone's hotspot or use a modem/access point/wireless router and create your own network that way. The nodes that are used for teleoperation are run on the static laptop, and the nodes interfacing with Eddie are going to be run on the laptop that is mounted on the robot. Watch the video for ROS2 Networking in Moodle and breeze through the following links:

- The Robotics Back-End: ROS2 Multiple Machines Tutorial
- Discovery in ROS2
- ROS Domain ID
- The Construct: Separating ROS2 environments – ROS_DOMAIN_ID

After setting up the proper networking, try teleoperating Eddie using the two laptops. Bring up rviz on the remote laptop, try displaying the RGB image of the kinect.

Exercise 5: 2D SLAM

You are now ready for running your first SLAM. Please review phase 0 if you don't remember what SLAM is and where it is used for. The role of Simultaneous Localization and Mapping is to use different types of sensors, including laser scanners, cameras, encoders, GPS and IMUs, to accurately map environments with dynamic obstacles and changing surroundings. Laser scanners are commonly used for localization and mapping in industrial environments and are considered the most robust sensor. However, not many open-source laser scanner SLAM algorithms can build accurate maps of big spaces and do so in real-time using mobile processors. SLAM Toolbox, a fully open-source ROS2 package, was proposed to solve this problem by providing accurate mapping algorithms, improved graph optimization, and reduced compute time. It also offers kinematic map merging, multi-session mapping, and prototype lifelong and distributed mapping applications. SLAM Toolbox was integrated into ROS 2 Navigation2 project and was selected as the new default SLAM vendor, replacing GMapping, to provide real-time positioning in dynamic environments for autonomous navigation.

In this section, you will perform SLAM and navigation using slam_toolbox and nav2 the same way we did in simulation in phase 0. nav2 works with other SLAM methods too, not just slam_toolbox. In the next section, we will try to use nav2 with rtabmap.

You can read up on the following documents to have a more in depth understanding of nav2:

- (SLAM) Navigating While Mapping
- First-Time Robot Setup Guide
- Nav2 Paper

The most important task in this section is tuning the parameters for slam_toolbox using its configuration and for nav2 using its configuration. Their configuration parameters are available in a yaml file in the config directory of eddiebot_nav package. You need to do an in depth research on the internet and GitHub for fine-tuning the parameters for fake laserscans.

- The choice of where to conduct your SLAM is yours. You need to do a SLAM, save the map, and do a navigation just as in phase 0.
- Your location must contain at least two turns.
- You can perform it in a room or in the corridors.
- Make sure you record the parts of the session using a camera (your phone, for example). Both of the group members must be in the video.

Exercise 6: Analyze Kinematics

In your report, write an in depth analysis of the `eddiebot_odom` packages. Carefully describe the kinematic model of Eddie and explain how `eddiebot_odom` computes odometry from encoder data. (you need to explain all the math involved). Please also write another explanation for how wheel velocities are computed using the given velocity command for Eddie.

You can consult the following two papers in the docs directory (or other papers):

- An implementation of ROS Autonomous Navigation on Parallax Eddie platform
- SLAM-BOT-California State University, Sacramento

Make sure the code for odometry in `eddiebot_odom` is correct.

Try to improve the odometry calculation from wheel encoders, current heading, and instantaneous velocities through the services that the `eddie` node provides. You can also change the code for calculating the speed of wheels in the controller node of `eddiebot_bringup` package for more accurate movement (especially during pure rotation). Explain how your changes have improved the overall performance.

Exercise 7: Visual SLAM

There are a bunch of VSLAM libraries for mobile robots. But only a handful of them are being actively developed and are compatible with ROS2. You are free to use any one of them:

- stella_vslam (previously known as OpenVSLAM)
- ORB_SLAM3 with this ROS2 wrapper
- isaac_ros_visual_slam

rtabmap is the recommended library for this section. As such, we will provide some instructions and pieces of code for this library. You are to build up on those and perform a successful SLAM along with navigation. RTABMap is a mixed-modality SLAM approach that was released in 2013 and is still being actively maintained and supported. It is a flexible SLAM approach that covers a wide variety of input sensors such as stereo cameras, RGB-D cameras, fisheye cameras, odometry and 2D/3D lidar data. Unlike other methods studied, instead of creating feature maps, RTABMap creates dense 3D and 2D representation of the environment which allows for it to be a drop-in visual SLAM replacement to existing methods without further post-processing. It is already being used in hobby, research, and small-scale service robot applications as a 2D SLAM "alternative" and contains all the basic features for utilization in mobile robot applications. You can take a look at this paper to know more about rtabmap.

The `rtabmap` launch file in the `eddiebot_nav` package is an example of how you could launch rtabmap nodes. You need to modify this file so that you could perform SLAM and navigation using rtabmap and nav2 packages.

Consult the following links (as stated in the rtabmap's repository, most of the documentations for ROS1 version, such as the parameters for the ros wrapper and rtabmap itself is valid for ROS2):

- rtabmap wiki.ros
- rtabmap_slam wiki.ros
- rtabmap_ros humble branch turtlebot3 example
- rtabmap_ros humble branch demo
- InterbotiX X-Series LoCoBot ROS Packages
- rtabmap setup tutorialsThis is old and is for ROS1 and should be used only as an example of the overall process.
- Mapping and Navigation with Turtlebot This is old and is for ROS1 and should be used only as an example of the overall process.

Do the same tasks as in exercise 4. Perform SLAM and do navigation this time using rtabmap. (the maps don't need to be the same place)

Extra Credits

1) Use rtabmap's visual odometry in combination with wheel odometry through the use of extended kalman filter (skeleton codes are available in the eddiebot-ros repo)
2) Expose your phone's IMU to ROS2 constructs and use them to improve the odometry through an EKF filter.
3) Pull requests to the `eddiebot-ros` repository that fix a bug or improve a feature.
4) Improving the balance of the wheels and making the robot more stable while moving (less jerk)
5) Converting the distances from IR and UltraSonic sensors to laserscan messages (with correct frame and header)
6) Improving the odometry in Exercise 6
7) **Phase 3**:
    1) Create an in depth diagram of the exact steps and tools you need to do in order to proceed with your proposal in Phase 1.
    2) Complete the first few steps of your plan.

Submission

- Create an in depth report in English of all the exercises, launch files, codes, … that were used in this phase. Make sure to include screenshots for each exercise.
- Upload the PDF and the source text for your report.

- Take a recording of your group members and Eddie, while performing the SLAM and upload it with all the code you wrote in a ZIP file.

نکات:

- در هنگام تحویل از هر فرد به صورت شخصی مصاحبه گرفته می‌شود؛ نمره این فاز و فازهای دیگر برای تمام اعضای گروه یکسان و برابر با میانگین و یا کمینه نمره‌های اعضای گروه در مصاحبه است. بنابراین **از مشارکت همگروهی خود در پروژه اطمینان حاصل کنید.**
- در صورت عدم همکاری مناسب همگروهی خود، حداقل یک هفته قبل از اتمام مهلت فعالیت، با دستیاران در ارتباط باشید.