M. Huraira Anwer - 25100314 Sadaan Tahir - 25100224

EE 3002 L1 (Junior Design Studio - Robotics)

Spring 2025 - LAB 7

This week's lab explores **mapping**, **S**imultaneous **L**ocalization and **M**apping (**SLAM**), and **navigation** in ROS, with a focus on **Hector SLAM**, **GMapping**, and the **Navigation Stack**.

You will gain hands-on experience with fundamental principles and practical implementations of mapping and navigation techniques crucial for autonomous robotic systems. Through ROS tools and algorithms, you will explore environment mapping, SLAM, and path planning, gaining the expertise to configure and deploy autonomous navigation systems. This lab offers a valuable opportunity to enhance your robotics knowledge and develop practical skills with industry-standard tools for real-world applications.

Task 1: SLAM and Navigation using GMapping in Gazebo [20 MARKS]

1.1 Initial Setup [1 Mark]:

First, make a workspace called **lab7_ws**:

```
jdsrobo@ubuntu:~$ mkdir -p ~/lab7_ws/src
jdsrobo@ubuntu:~$ cd lab7_ws/
jdsrobo@ubuntu:~/lab7_ws$ catkin_make
```

Then, you need to setup the **TurtleBot3 package** (same as Lab 3). Follow these steps for a fresh install if you don't already have the package:

1. **Install the TurtleBot3 package** by typing this command in a new terminal: sudo apt-get install ros-noetic-turtlebot3.
2. Next, **clone the following packages** in the directory **jdsrobo@ubuntu:~/lab7/src$** using these terminal commands:
   - 2.1. **ROS msgs package for TurtleBot3**: git clone --branch noetic https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
   - 2.2. **TurtleBot3 package**: git clone --branch noetic https://github.com/ROBOTIS-GIT/turtlebot3.git

Note: If the github repos don't clone properly, download their zipped folders and unzip them under ~/lab7_ws/src/.

3. **Build** all the ROS packages within your workspace using catkin_make.
4. Next, to **optimize performance on your virtual machines**, add the following lines towards the very end of the **~/.bashrc file** (open it using gedit ~/.bashrc):
   - 4.1. export SVGA_VGPU10=0
   - 4.2. export TURTLEBOT3_MODEL=burger

4.3. source /opt/ros/noetic/setup.bash

5. **Save** the **~/.bashrc file** and apply the changes using this terminal command: source ~/.bashrc.
6. Next, clone the **simulation package** using these terminal commands:
   6.1. cd ~/lab7_ws/src
   6.2. git clone --branch noetic https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
7. **Build the workspace again** and make sure you source the setup.bash file by running source devel/setup.bash.

1.2 Creating an Environment Map using GMapping [10 Marks]:

In this task, you will generate a **2D map of a pre-made environment** using the **GMapping SLAM algorithm** while **manually navigating** the **TurtleBot3** in Gazebo. The generated map will be saved for later use in autonomous navigation.

Step 1. Now with the basic setup done, **install the GMapping package** using:

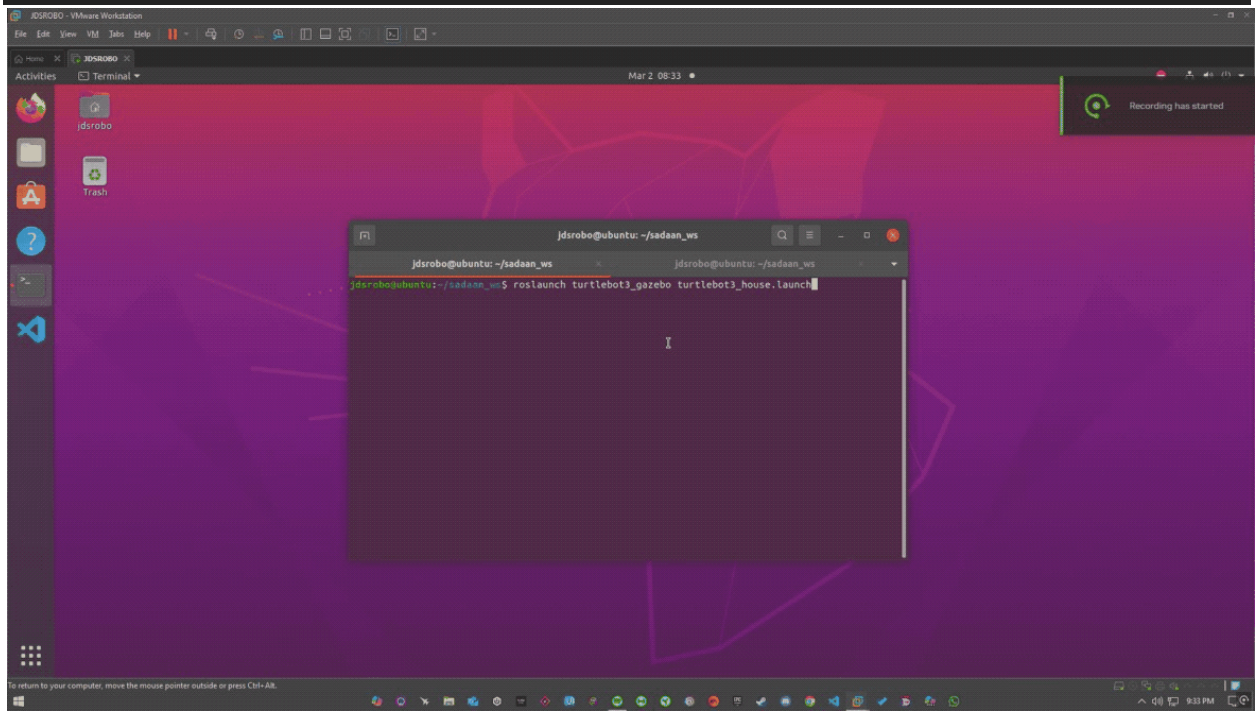
```
sudo apt-get install ros-noetic-gmapping
```

Step 2. **Source ROS** after the installation:

```
source /opt/ros/noetic/setup.bash
```

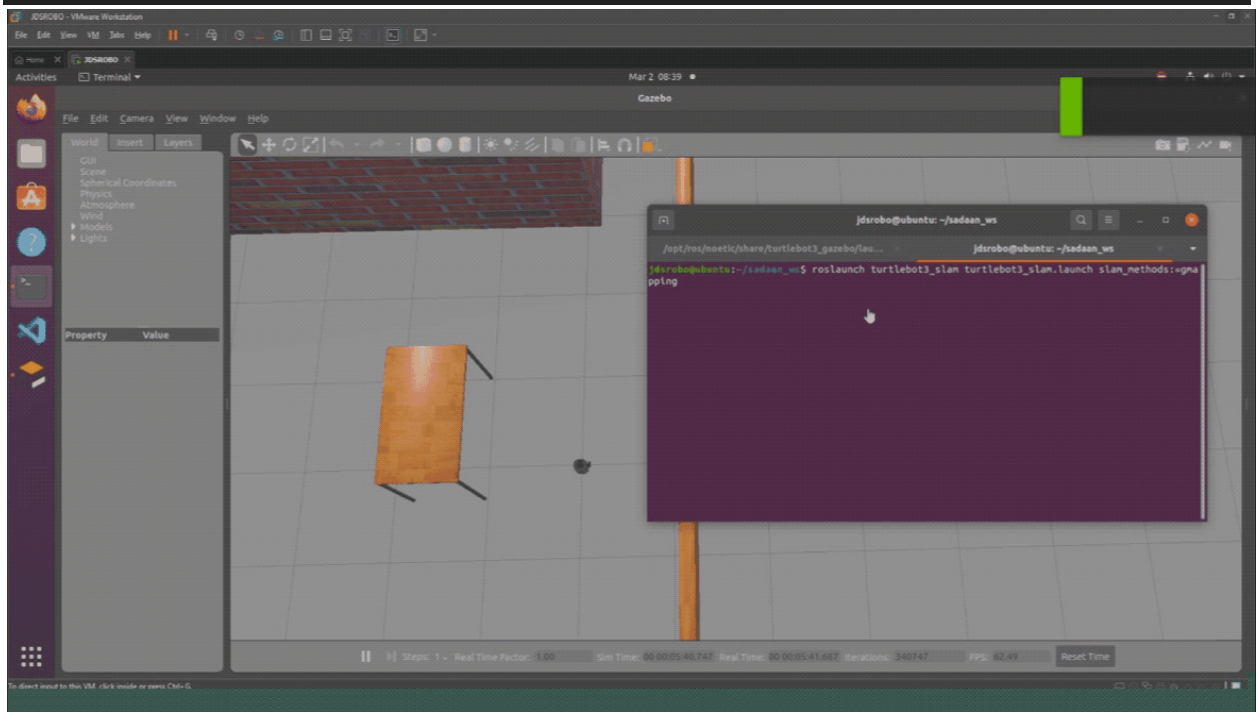
Step 3. **Launch the TurtleBot3** in a gazebo environment (takes awhile to load):

```
roslaunch turtlebot3_gazebo turtlebot3_house.launch
```



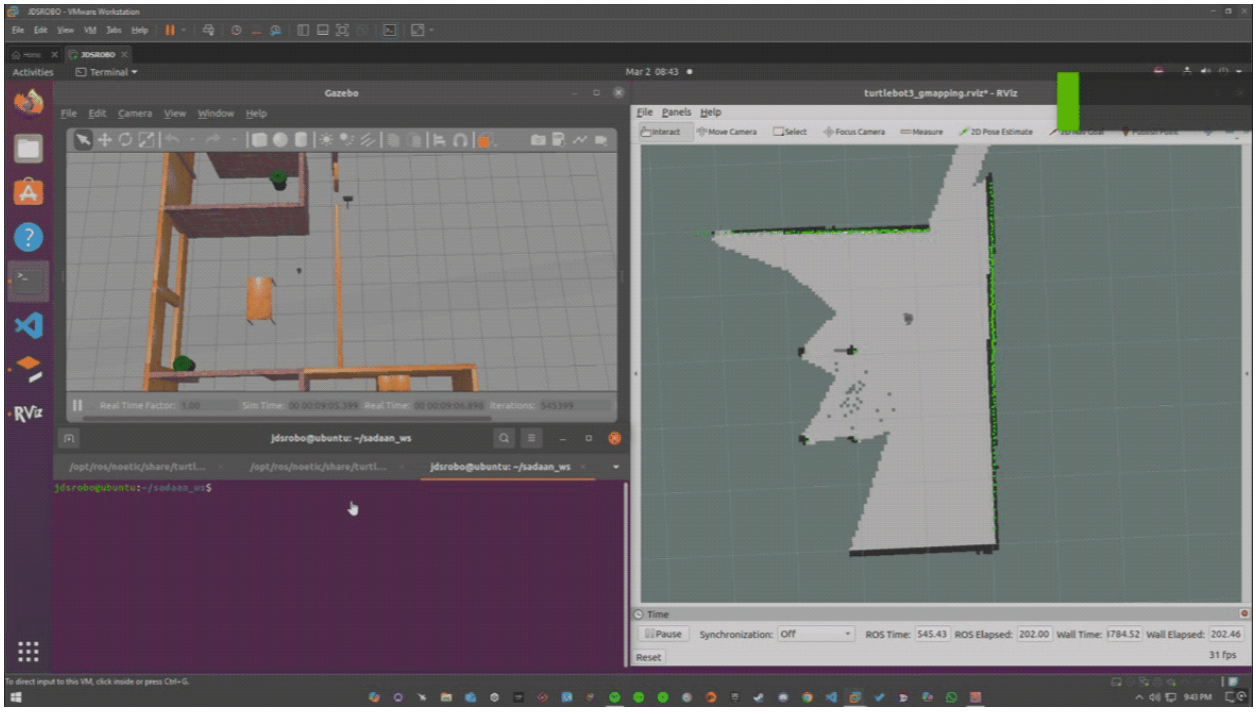
Step 4. Launch the **GMapping SLAM node**:

```
roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
```



Step 5. **Move the robot** around in the environment using the keyboard teleoperation node:

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```



Note: Ensure that you explore all areas so that the SLAM algorithm can generate a complete map. You will be needing this map in the next task, Task 1.3.

Step 6. **Save the generated map**. The command below will generate **map.pgm** and **map.yaml** in your home directory:

```
rosrun map_server map_saver -f ~/map
```

1.3 Navigation using the Saved Map [9 Marks]:

Using the previously created map, in this task, you will **autonomously navigate** the TurtleBot3 from one point to another using the **ROS Navigation Stack**, with path planning and obstacle avoidance handled by **move_base** in **RViz**.

Step 1. Relaunch the environment:

```
roslaunch turtlebot3_gazebo turtlebot3_house.launch
```

Step 2. Ensure that the required navigation packages are installed:

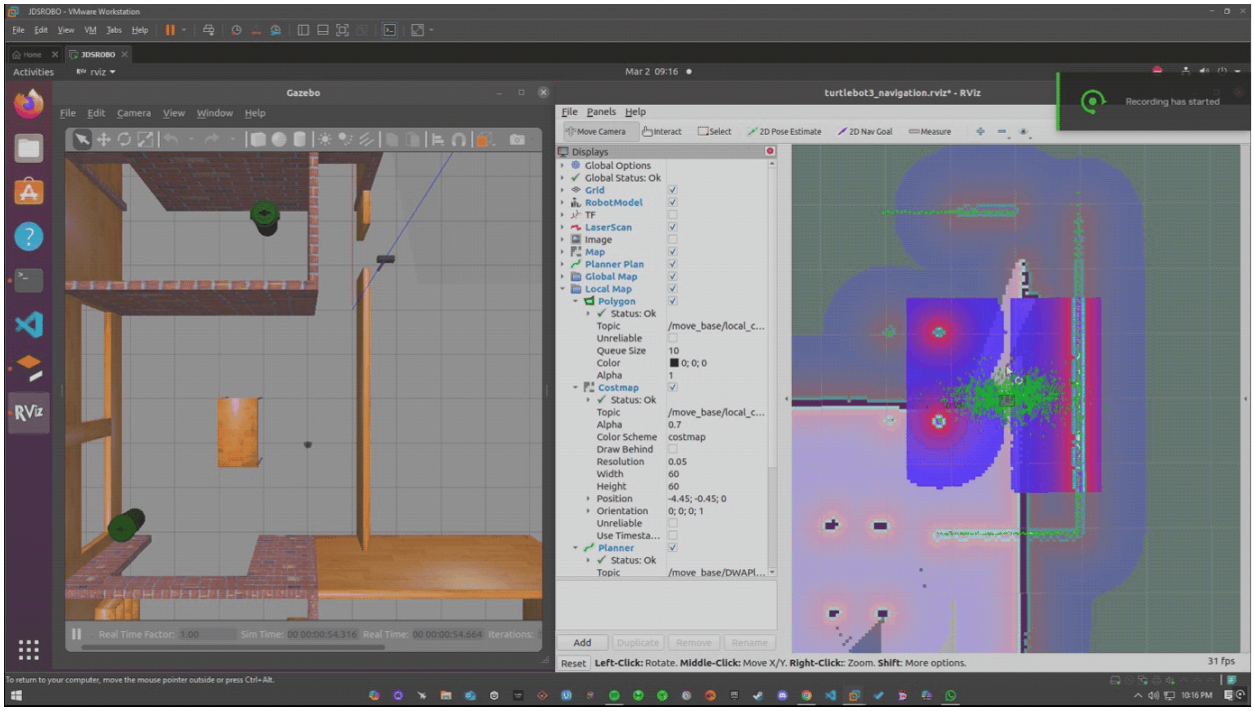
```
sudo apt-get install ros-noetic-dwa-local-planner ros-noetic-navigation

source /opt/ros/noetic/setup.bash
```

Optional Step: If you installed TurtleBot3 from source, then rebuild your workspace before the next step.

Step 3. Launch the navigation stack with the saved map:

```
roslaunch turtlebot3_navigation turtlebot3_navigation.launch
map_file:=$HOME/map.yaml ## This is a single terminal command ##
```



Task 2: Mapping this Lab using Hector SLAM [20 MARKS]

About Hector SLAM: https://wiki.ros.org/hector_slam.

2.1 Visualizing RP Lidar Data on Jetson Nano [10 Marks]:

In this task, you will **connect** the **RP Lidar sensor** to the **Jetson Nano**, **set up** the necessary **ROS package**, **enable the USB connection**, and **publish the LaserScan data**. You will then visualize the laser scan data in **RViz** by defining a **static transform** from the **robot's base** to the **Lidar frame**.

Step 1. Connect the **RP Lidar A3M1** to the **Jetson Nano**. Follow the steps given in the previous Lab's Manual. These links might also help:

Link 1:
https://www.youtube.com/watch?v=usfoVjUZxY4&t=418s&ab_channel=SoroushMortezapour

Link 2:
https://www.generationrobots.com/media/LD310_SLAMTEC_rplidar_datasheet_A3M1_v1.0_en.pdf?srsltid=AfmBOopcA6fVyH-oS9Za7yQYvC17Mk46JskRpCFFuc644x3liIFlH4rr

Step 2. **Create** a new ROS workspace called **lab7_ws** and clone the following RPLidar ROS package into the **src** folder:

https://github.com/Slamtec/rplidar_ros.git

Step 3. **Build** the workspace using **catkin_make** and **source** its **setup.bash** file.

Step 4. **Check** the connected USB devices on your Jetson Nano (**make sure the RPLidar is connected**) using this terminal command:

```
ls /dev/ttyUSB*
```

Give read/write permissions to the USB port where the Lidar is connected:

```
sudo chmod 666 /dev/ttyUSB0 ## Assuming the Lidar is connected to USB0 ##
```

Step 5. **Launch the Lidar node** to **publish** its data

```
roslaunch rplidar_ros rplidar_a2m12.launch
```

Verify if the data is actually being published:

```
rostopic list

rostopic echo /scan
```

Step 6. **Get** the **Frame ID** of the LaserScan topic. Note this frame_id, as you will be needing it afterwards.

```
rostopic echo /scan | grep frame_id
```

Step 7 - (Crucial). In order to **visualize** the LaserScan data in RViz, we need to **determine an inertial robot frame** (i.e base_link), at which our Lidar is mounted on the robot. Hence, we will be using the **tf2 library** to **publish a static transform from our robot's center**.

First, it is pivotal that you understand the intuition behind these transforms. Use these links to get a better overview of what we are dealing with here:

https://foxglove.dev/blog/understanding-ros-transforms
https://automaticaddison.com/coordinate-frames-and-transforms-for-ros-based-mobile-robots/

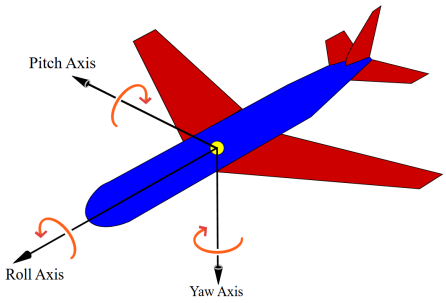
Now, with the theory out of way, **install the tf2 package**:

```
sudo apt update

sudo apt-get install ros-noetic-tf2-ros ros-noetic-tf2-tools ros-noetic-tf2-msgs
ros-noetic-tf2-py

source /opt/ros/noetic/setup.bash
```

Measure the x,y and z distance **(in meters**) and rotation (**in radians**) from the robot's center:
x_val, y_val, z_val, yaw_val, pitch_val and **roll_val**.



Finally, use the following command to publish the static transform:

```
rosrun tf static_transform_publisher x_val y_val z_val yaw_val pitch_val roll_val
base_link <lidar_frame> 100
```

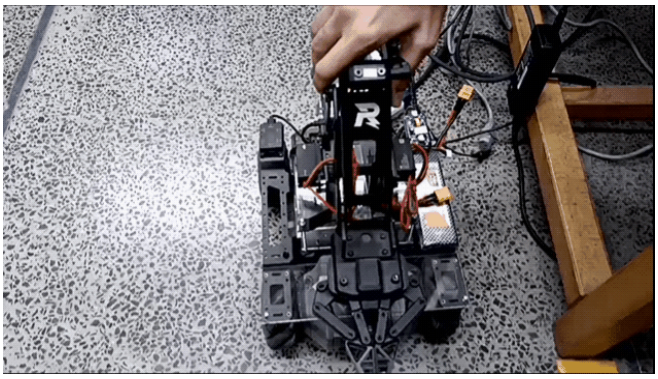
*Note: Replace **x_val y_val z_val yaw_val pitch_val** and **roll_val** with your measured values and add the actual **frame_id** found in **Step 6** in place of **<lidar_frame>***

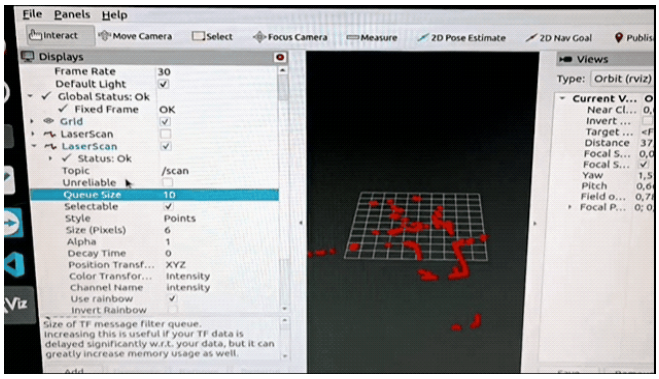
Step 8. **Visualize** the **LaserScan data in RViz** using this terminal command:

```
rosrun rviz rviz
```

With RViz launched, in "**Global Options**", set **Fixed Frame** to **base_link** and click "**Add**" (bottom left), select **LaserScan**, and choose **/scan**.

Change the "**Style**" to **Points** and "**Size**" to **6 Pixels** for better visibility of the **LaserScan** data, as shown in the GIFs below on the next page.





2.2 Mapping using Hector SLAM:

This task involves setting up **Hector SLAM** and integrating it with the RP Lidar A3M1. **Hector SLAM will be used for generating a map of the robot's present environment**.

Step 1. Before proceeding to the Hector SLAM setup, install some dependencies like Qt5 tools needed for visualisation in Hector SLAM:

```
sudo apt-get update

sudo apt-get install qt5-qmake qtbase5-dev
```

Step 2. Clone the Hector SLAM repository (given below) under **~/lab7_ws/src/** and build it:

```
git clone -b noetic-devel https://github.com/tu-darmstadt-ros-pkg/hector_slam.git
```

Note: After the build completes, make sure to source your workspace.

Step 3. Now, **modify the launch file** for the **hector_mapping package** to configure the robot's **base frame** and **odometry frame**, **laser scan topic** and to publish a static **transform** between the Lidar and the robot's base frame to link the Lidar's position relative to the robot.

Open the launch file in a text editor:

```
gedit ~/lab7_ws/src/hector_slam/hector_mapping/launch/mapping_default.launch
```

Modify the following lines to set the **base frame** and **odom frame** to **base_link**:

```
<arg name="base_frame" default="base_link"/>
<arg name="odom_frame" default="base_link"/>
<arg name="scan_topic" default="laser"/>
```

Add a **static transform publisher node** (as a new line) to broadcast the transform from **base_link** to the **Lidar's frame**:

```
<node pkg="tf" type="static_transform_publisher" name="base_to_laser_broadcaster"
args="x_val y_val z_val yaw_val pitch_val roll_val base_link lidar_frame 100"/>
```

*Note: Replace **x_val y_val z_val yaw_val pitch_val** and **roll_val** with your measured values and add the actual **frame_id** of the Lidar in place of **<lidar_frame>**.*

Step 4. Now, open the launch file, **tutorial.launch**:

```
gedit ~/lab7_ws/src/hector_slam/hector_slam_launch/launch/tutorial.launch
```

Step 5. Change the **simulation time parameter** to **false**.

```
<param name="/use_sim_time" value="false"/>
```

Step 6. Now, before launching the mapping node, ensure your Lidar is correctly publishing the data. Launch the **Lidar node**:

```
roslaunch rplidar_ros rplidar_a2m12.launch
```

Verify that the Lidar data is being published:

```
rostopic echo /scan
```

Step 7. Launch the updated Hector SLAM mapping launch file:

```
roslaunch hector_slam_launch tutorial.launch
```

Step 8. **Visualize the mapping in RViz** by running this terminal command (in another terminal):

```
rviz
```

In **RViz**, set **Fixed Frame** to **map**, “**Add**” **LaserScan** and set the topic to **/scan** and “**Add**” **Map** to visualize the generated map.

2.3 Mapping the Lab:

This task involves **generating the complete map of the Feedback/JDS-Robo lab**. Initially a **ROSBag** will be recorded while moving the robot around. Later, we will replay this bag with **Hector SLAM** to build the final map.

Step 1. Insert the **Jetson Nano** into your **RoboMaster EP Core** and connect the **Lidar**. Use double-sided tape to place the Lidar towards the front of the EP Core. The **Buck Converter** and **Lipo Battery** also need to be fixed onto the EP Core via double-sided tape.

Note: Look at the images on the last page for reference.

Step 2. Access the **Jetson Nano** via the **Remote Desktop Connection** app on your **Windows machines**. Ensure the **XRDP-setup** is completed on the Jetson before this step.

Step 3. To ensure proper frame alignment between the map and the laser scanner, publish a **static transform** for the Lidar:

```
rosrun tf static_transform_publisher 0 0 0 0 0 0 1.0 map laser 10
```

This command essentially links **map** to the **laser** frame, ensuring correct coordinate transformations.

Step 4. Disable simulation time before recording the ROSBag:

```
rosparam set use_sim_time false
```

This will ensure that the real-time sensor data is recorded with the actual system clock.

Step 5. Launch the Lidar node and verify if Lidar data is being published:

```
roslaunch rplidar_ros rplidar_a2m12.launch
```

Step 6. Navigate to your home directory or preferably **~/Documents** and start recording the Lidar data and the TF transformations:

```
rosbag record -O my_bag.bag /scan /tf
```

Now with the above command entered in terminal, begin moving the robot (**slowly**) around the entire lab. You can stop the recording when you think an adequate portion of the lab has been mapped correctly by the Lidar. Use CTRL + C to stop the recording.

Step 7. Now disconnect the Lidar, restart Jetson Nano and connect it to a monitor in the lab.

Step 8. Run roscore in a new terminal:

```
roscore
```

Step 9. When using a recorded bag instead of real-time sensor data, **we need to enable simulation time in the Hector SLAM launch file**. Open the **tutorial.launch** file and change the **use_sim_time** parameter to **true**:

```
gedit ~/lab7_ws/src/hector_slam/hector_slam_launch/launch/tutorial.launch

<param name="/use_sim_time" value="true"/>
```

Or alternatively, you can type this in terminal:

```
rosparam set use_sim_time true
```

Step 10. Launch the Hector SLAM node:

```
roslaunch hector_slam_launch tutorial.launch
```

Then, navigate to the directory where you stored the bag file and play it:

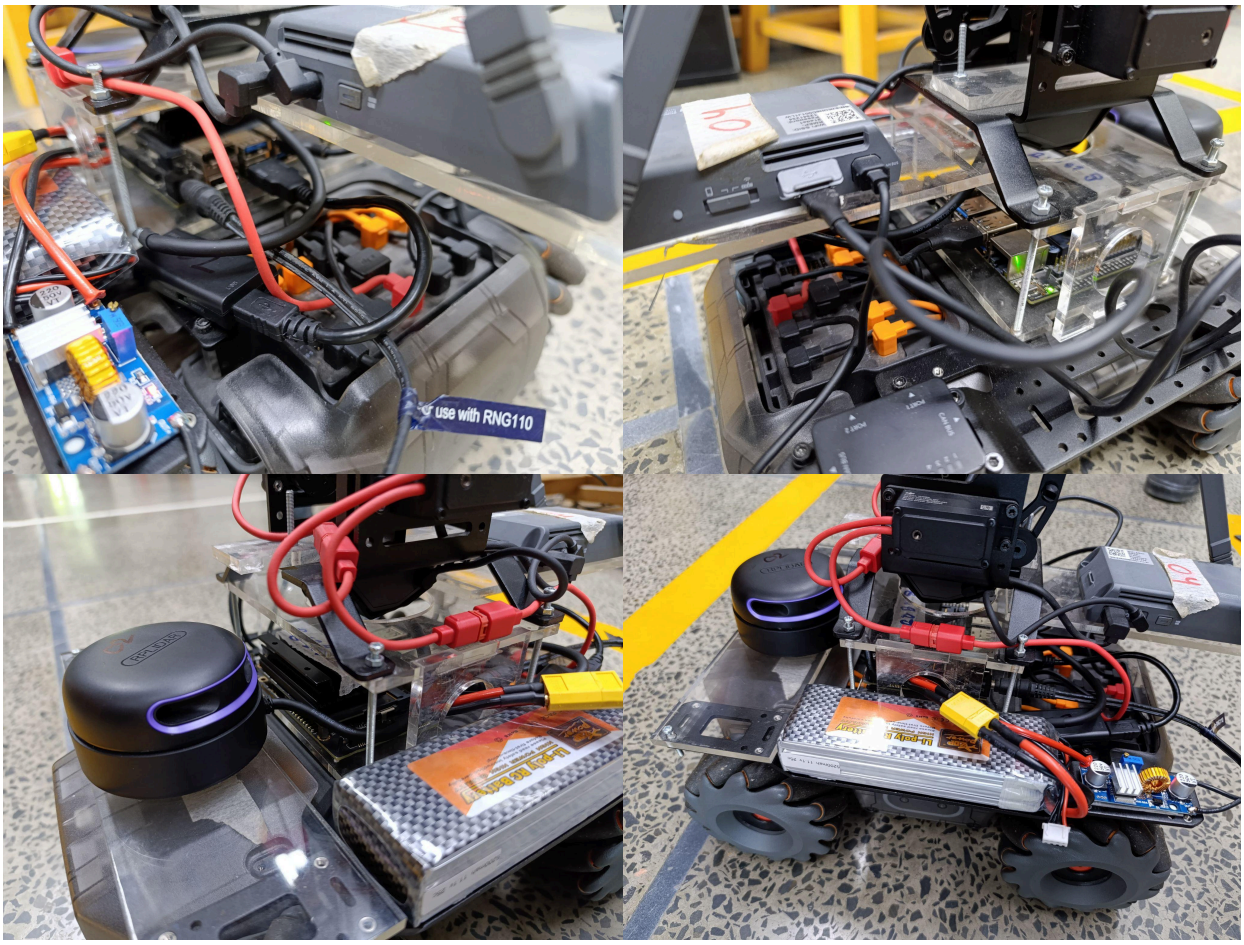
```
cd ~/Documents ## Preferred location of storing the bag file ##
```

```
rosbag play my_bag.bag --clock
```

Step 11. Once the mapping is complete, save the generated map using:

```
rosrun map_server map_saver -f ~/map_rcs_lab
```

This command saves the map as an image and a **.yaml** file in your home directory under ~/map_rcs_lab.



Reference images for setting up the RoboMaster EP Core (its chaotic I know)

Submission Requirements:

1. Include a **zip file** containing the following items:
   1.1. **All files for all Tasks inclusive of videos and screenshots wherever necessary**.
2. Submit a **LAB report** in PDF format (please do not submit word files). This report should provide explanations for all tasks performed along with screenshots, whenever suitable.