Bringing Up a Differential-Driven Mobile Robot TurtleBot3 Burger

Moldir Zabirova, Valeriya Kostyukova, Daryn Kenzhebek

I. Robot Assembly

TURTLEBOT3 is a differential drive type ROS based mobile robot. It is intended to be used for education, research, hobby and product prototyping. For this laboratory, we used a BURGER model which consists of three main platforms on top of each other.

The main parts of the robot include 2 DYNAMIXEL's in the wheel joints, OpenCR controller, 360-degree LIDAR sensor, Raspberry Pi 3 as SBC, LiPo battery, and other mechanical structures. Because the robot was partially assembled, we gathered together a complete version according to the manual. The assembled robot is depicted on Fig. 1.

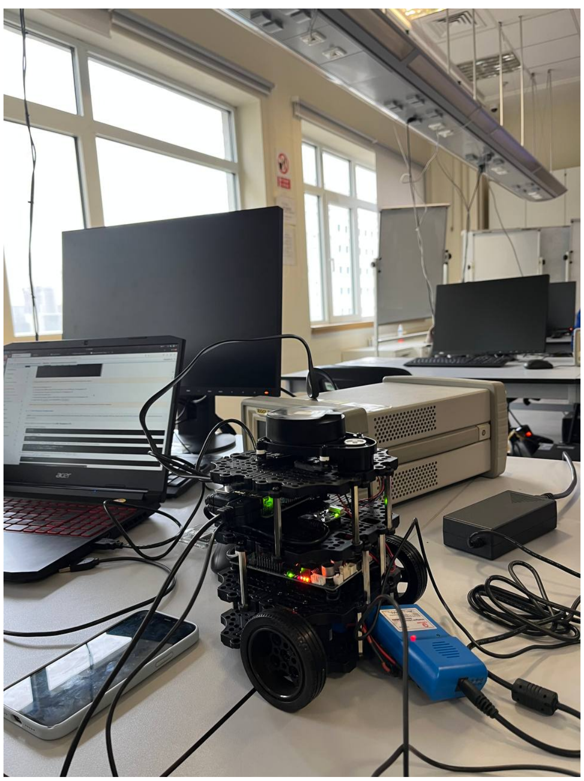

Fig. 1. Assembled TurtleBot3 Burger

Before connecting and using the robot, we familiarized with the main characteristic of the robot which are represented in Table 1 and on Fig. 2 [1].

II. PC Setup

Firstly, we installed the script for ROS Noetic which is fetched from the official ROBOTIS-GIT repository. Then all the necessary ROS packages were installed such as joystick control dependencies to mapping and navigation tools.

Items	Burger
Max translational velocity	0.22 m/s
Max rotational velocity	2.84 rad/s
Size	138 x 178 x 192 mm^3
Weight	1kg
IMU	Gyro 3 Axis Accel 3 Axis
Power connectors	5V / 4A
Charging time	2h 30m
Operating time	2h 30m

TABLE I
HARDWARE SPECIFICATIONS

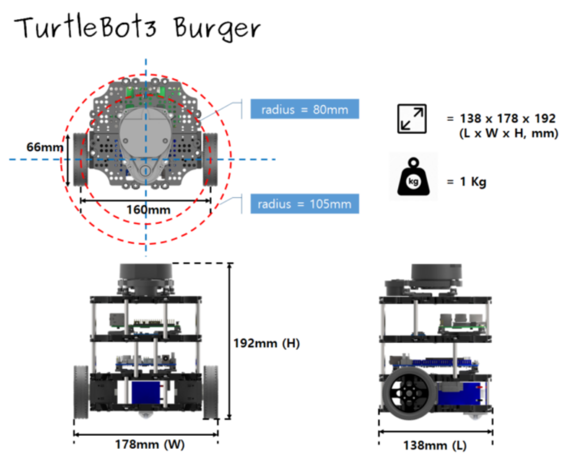

Fig. 2. Visual Repesentation, Dimensions, and Weight

```
sudo apt-get install ros-noetic-joy
ros-noetic-teleop-twist-joy \
  ros-noetic-teleop-twist-keyboard
  ros-noetic-laser-proc \
  ros-noetic-rgbd-launch
  ros-noetic-rosserial-arduino \
  ros-noetic-rosserial-python
  ros-noetic-rosserial-client \
  ros-noetic-rosserial-msgs
  ros-noetic-amcl ros-noetic-map-server \
  ros-noetic-move-base
  ros-noetic-urdf ros-noetic-xacro \
  ros-noetic-compressed-image-transport
  ros-noetic-rqt* ros-noetic-rviz \
  ros-noetic-gmapping ros-noetic-navigation
  ros-noetic-interactive-markers
```

The next step involved the installation of TurtleBot3 packages. Once the software setup is accomplished, the PC's network configuration is fine-tuned. The PC is connected to a WiFi device, and the assigned IP address is identified using the 'ifconfig' command. The ROS IP settings are then updated in the '.bashrc' file to record the acquired IP address as the ROS_MASTER_URI and ROS_HOSTNAME. This ensures proper communication between the PC and TurtleBot3 Burger. Finally, the '.bashrc' file is sourced to apply the changes and initiate a connection between the PC and the robot.

III. SBC AND OPENCRSETUP

The next step involded the setup of SBC which was Raspberry Pi 3. The microSD card was prepared by downloading the appropriate TurtleBot3 SBC image, which was consequently burned onto the microSD card using Raspberry Pi Imager Fig. 3.

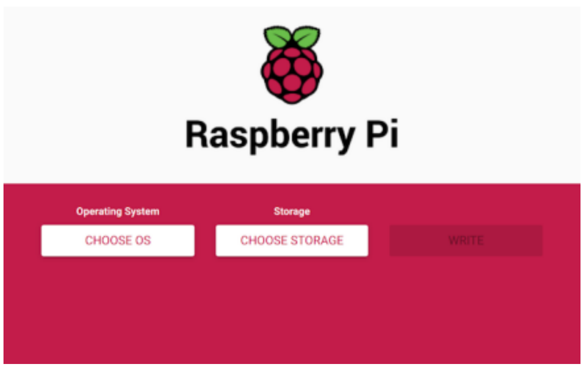

Fig. 3. Raspberry Pi Imager

The WiFi network settings were configured by editing the netplan file with the SSID and password information. Upon booting up the Raspberry Pi, we connected the HDMI cable, keyboard, and power source. The login credentials were the following:

```
login: ubuntu
password: turtlebot
```

ROS network configuration on the Raspberry Pi involves confirming the WiFi IP address, editing the .bashrc file to specify the ROS_MASTER_URI (IP of the remote PC) and ROS_HOSTNAME (IP of RPi).

Finally, we installed all the necessary packages, drivers, and dependencies for LIDAR LDS-01 sensor.The LDS_MODEL is exported to the bashrc file. The setup is completed by sourcing the bashrc file to apply the changes.

Connecting the OpenCR to the Raspberry Pi involves using a micro USB cable. Following the physical connection, the Raspberry Pi needs to be prepared for uploading the OpenCR firmware. The necessary packages are installed with commands that ensure compatibility and architecture support.

To upload the firmware to the OpenCR, users navigate to the opencr_update directory and execute the update script, providing the OPENCR_PORT and OPENCR_MODEL as parameters. This process ensures that the Raspberry Pi communicates effectively with the OpenCR, updating its firmware to the specified model.

IV. BINDING PC AND RASPBERRY PI

To establish communication between PC and RPi, both devises needed to be connected to the same network. Linux provides with powerful tool called ssh, with which we can connect to the terminal of the device, from another device.

Executing roscore initializes the ROS master, laying the groundwork for TurtleBot3 operation. On the PC, a new terminal is opened, and an SSH connection to the Raspberry Pi is established using its IP address. The next command sets the TurtleBot3 model by exporting the TURTLEBOT3_MODEL variable as "burger". Following this, the roslaunch command is used to initiate the TurtleBot3 basic packages through the turtlebot3_robot.launch file.

```
ssh ubuntu@10.1.71.53
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_bringup
    turtlebot3_robot.launch
```

The printed summary provides crucial information about the configuration. The launched nodes include turtlebot3_core for serial communication, turtlebot3_diagnostics for diagnostics, and turtlebot3_lds for the LIDAR driver.

The ROS master URI is specified as in the bashrc file, indicating successful communication between the PC and the TurtleBot3 on the Raspberry Pi. The processes for core, LIDAR, and diagnostics are initiated, and the ROS Serial Python Node establishes a connection to the OpenCR board (Fig. 4).

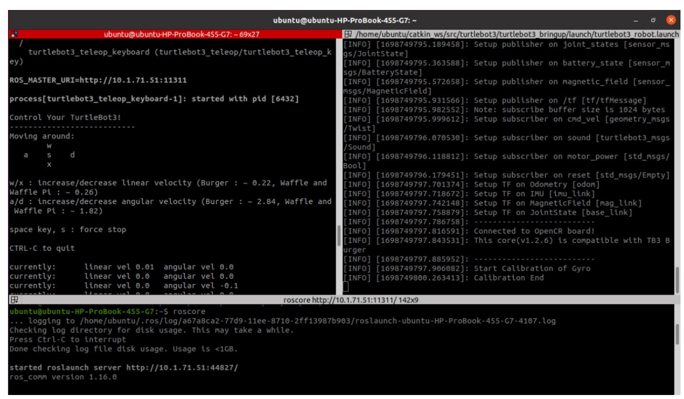

Fig. 4. Launching TurtleBot3

V. SIMPLE TELEOPERATION WITH KEYBOARD

To teleoperate the TurtleBot3 using a keyboard, we used the turtlebot3_teleop_key node.

```
roslaunch turtlebot3_teleop
    turtlebot3_teleop_key.launch
```

When the node is successfully launched, instructions for controlling the TurtleBot3 using the keyboard were displayed in the terminal (Fig. 4).

By pressing the w key, the linear velocity increases, moving the robot forward. Conversely, the x key is employed to decrease the linear velocity, decelerating or a complete stop. To manipulate the TurtleBot3's orientation, the a key augments the angular velocity, inducing a counterclockwise rotation. The d key, on the other hand, diminishes the angular velocity, stopping the rotation. In instances where an abrupt stop is imperative, the Space key or s key can be pressed (Fig. 5).

Fig. 5. Control of TurtleBot3's velocities on the floor

VI. Conclusion

In conclusion, the process of bringing up the TurtleBot3 Burger involved a series of steps, including robot assembly, PC setup, Single Board Computer (SBC) and OpenCR configuration, and the binding of the PC and Raspberry Pi. Through hardware assembly, software installations, and network configurations, we successfully created a functional and interconnected robotic system. The detailed specifications of the TurtleBot3 Burger were analyzed, providing information into its dimensions, weight, and key hardware components. The software setup on the PC and Raspberry Pi, including the installation of ROS packages and configuration of network settings, laid the foundation for communication between the devices. The integration of the OpenCR controller, Raspberry Pi, and LIDAR sensor demonstrated the collaborative functionality of these components. The teleoperation of the TurtleBot3 Burger using a keyboard showed the robot's responsiveness to dynamic velocity commands. Overall, this laboratory project has provided a hands-on experience in setting up and controlling a differential-driven mobile robot, offering valuable insights into robotics, ROS, and the TurtleBot3 platform. The further steps will include navigation and path planning in the labyrinth environment.

References

[1] Y. Name, "ROBOTIS e-Manual," ROBOTIS e-Manual. https://emanual.robotis.com/docs/en/platform/turtlebot3/features/specifications