UGV Rover manual

Arnoud Visser

July 2025

1 Introduction

In the course Vision for Autonomous Robots we are planning to use 10 UGV Rover robots for 40 students.

2 UGV Rover robot

The idea is behind the UGV Rover robot is an open-source mobile robot based based on powerful Jetson Orin ROS2 Kit[1]. Nvidia claims that the Jetson Orin Nano is the most affordable generative AI super computer. It delivers up to 67 TOPS of AI performance, which makes it possible to run Large Language models such as Liama 3.1 and Vision Tranformers such as DINOv2 on-board. The Jetson Orin basis is Nvidia JetPack, which is Ubuntu Linux kernel, including dedicated Nvidia drivers and the full CUDA toolbox.

On this basis Waveshare has build a UGV Rover robot, a robot with 6 wheels. On top of that, the robot has a pan-tilt camera with 160 deg field-of-view, a depth-camera and a Lidar. This sensor-suite makes it very suitable for a master course like 'Vision for Autonomous Robots'.

Figure 1: The UGV Rover Jetson Orin ROS2 from Waveshare

[1] `https://www.waveshare.com/wiki/UGV_Rover_Jetson_Orin_ROS2`

It has a 5M-pixel camera at the top, stereo cameras at the front and back and 4 wheels to drive around. Actually, the stereo cameras are developed by Luxonis for OpenCV, just as the RAE robot used in 2024 course, so most developed software should still work.

The UGV Rover robot runs on Ubuntu 22.04.05, with Nvidia Jetpack 6.0 installed on top. The UGV Rover originally came with docker image running the code dedicated to the UGV Rover. Docker is a system that separates your code from the operating system, because in a docker image a prepared Linux installation with a variety of installed packages can be loaded. In principal a good idea, but because the manufacturer of the robot didn't prepared, maintained and documented the docker image after the launch, we will use the UGV Rover drivers directly from the Nvidia Jetpack environment.

The UGV Rover drivers are implemented as ROS nodes. ROS stands for Robot Operating System, which is actually not an operating system, but middleware running above the OS. ROS has become the actual standard in robotic research & development; nearly every modern robot supports ROS. The idea of ROS is that a number of nodes are launched, each publishing one or more topics. For the UGV Rover such topics can be the pan-tilt camera images, or the current voltage level. Other nodes are not publishing but listening, which means that you can give them input. For the UGV Rover robot this is for instance the cmd_vel and led_ctrl (see Table 1).

/camera_info	/parameter_events
/cmd_vel	/processed_image
/image_raw	/rosout
/image_raw/compressed	/ugv/joint_states
/image_raw/compressedDepth	/ugv/led_ctrl
/image_raw/theora	/voltage
/joy	

Table 1: An overview of ROS topics published by the UGV rover

In the next section we will go step-by-step through the commands to start the robot and ROS-nodes. After that we will give a troubleshoots with tricks to use if one of those steps fails. This instructions has the precondition that you installed ROS humble at your laptop, as already described in Sec. 5

3 Start the UGV rover

Note that you build up a connection between your laptop and the UGV rover. Commands from your laptop are given in green, commands given from the UGV rover are given in blue. First thing is to connect to the robot.

You could directly login to the robot with the following command:

```
$ ssh jetson@ROVER_IP
```

At the UGV rover, you are at the Ubuntu 22.04.5 environment. You can check the ethernet connections with the command

You can get ID-information specific to your RAE robot with the following command:

```
$ ifconfig
```

At the moment we recommend to use the wifi_vislab as accesspoint (both for the UGV rover and your laptop).

When both the UGV rover and laptop are connected you can check the wireless connection with the ping command

```
$ ping ROVER_IP
```

And equivalent from the robot-side:

```
$  ping LAPTOP_IP
```

At the UGV Rover, it is time to start some of the ROS-nodes. Before you do that, make sure that the ROS_DOMAIN_ID to be equal to your group-number GROUP_ID (equal to the white sticker on your UGV rover). After that, launch the ROS-nodes from the UGV rover

The most fundamental node is ugv_bringup, which can be started with the command:

```
\ export ROS_DOMAIN_ID=GROUP_ID
\ source ~/ugv_ws/install/setup.bash
\ ros2 run ugv_bringup ugv_bringup &
```

This nodes publishes some internal readings, for instance the voltage, which should become lower than 9V. It can be checked with:

```
\ ros2 topic echo /voltage --once
```

Now we want also to be able to control the actuators of the robot. That is done with another node, which can be started with:

```
\ ros2 run ugv_bringup ugv_driver &
```

Now you should be able to turn on the leds of the UGV rover by sending the following message:

```
ros2 topic pub /ugv/led_ctrl std_msgs/msg/Float32MultiArray "{
    data: [9,0]}" -1
```

When you have ROS also installed at your laptop, and you are in the same domain, you should be able to turn the LEDs off remotely.

```
\ source /opt/ros/humble/setup.bash
\ export ROS_DOMAIN_ID=GROUP_ID
\ ros2 topic pub /ugv/led_ctrl std_msgs/msg/Float32MultiArray "{
    data: [0,0]}" -1
```

The LED-headlights should light up white, and turn off with the last message.

Now you should also be able to control the pan-tilt of the camera, with the following message:

```
\ ros2 topic pub /ugv/joint_states sensor_msgs/msg/JointState "{
    name: ['pt_base_link_to_pt_link1',pt_link1_to_pt_link2],
    position: [0.52,0.26]}" -1
```

This will drive the pan-tilt camera to the position with the pan at 30° and the tilt at 15° (the message defines angles in radians). Last, but not least, is to control the wheels. It is wise to lift the robot up from the table before giving this command. The robot can drive off the table before you have typed the stop command.

```
\ ros2 topic pub --once /cmd_vel geometry_msgs/msg/Twist "{linear
    : {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z:
    1.8}}"
```

To ugv_driver activated the actuators of the UGV rover, for the sensors you have to start another node:

```
\ ros2 launch ugv_vision camera.launch.py
```

This node will published both the raw camera feed, as the rectified version where the distortion found at calibration is already corrected. You can visualize both streams with the following commands:

```
\ ros2 run image_view image_view --ros-args --remap image:=/
    image_raw &
\ ros2 run image_view image_view --ros-args --remap image:=/
    image_rect &
```

4 Troubleshooting

4.1 Connection to the RAE robot

When you cannot connect to the robot:

```
$ ssh jetson@192.168.0.185
```

Make sure that you and the robot are on the same wifi-access point. You can check a connection with:

```
$ ping 192.168.55.1
```

If that doesn't work, connect a USB-C cable directly to your laptop and login via the USB:

```
$ ssh root@192.168.55.1
```

At the UGV Rover, you are at the bare Ubuntu 22.04. Check the wifi-connection with:

```
$ ifconfig
```

4.2 The ROS2 nodes

The launch file of ROS is quite extensive, with several nodes starting up, including several info-message that look like error-messages.

Yet, it is not uncommon that the following command fails (for instance because the battery level is too low):

```
\ export ROS_DOMAIN_ID=42
\ ros2 launch rae_bringup robot.launch.py
```

This can be checked at both your robot and at your laptop side:

```
$ export ROS_DOMAIN_ID=42
$ ros2 topic list
```

When there are more than two topics, including `\cmd_vel`, you could try at the robot:

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist "{linear: {x:
    1.0}}" -1
```

The wheels of the robot should start turning. You can stop them again with:

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist "{linear: {x:
    0.0}}" -1
```

If this works, you could try the same commands at the laptop side.

When the response of `ros2 topic list` is only the two default ROS topics, the bringup has failed:

```
/parameter_events
/rosout
```

Often, your best option is to shutdown your robot, give it some rest while you recharge its battery

```
$ shutdown now
```

5 Installation

5.1 Ubuntu 22.04

Every ROS version is tightly coupled with a specific Ubuntu version. There are some initiatives to make ROS more platform independent, but most initiatives are experimental at best. When you have another OS than Ubuntu 22.04, we can recommend the RoboStack approach (see Sec. 5.3)

For ROS humble on Ubuntu 22.04, we follow the steps of Official setup.

5.1.1 Set locale

Make sure you have a locale which supports UTF-8.

```
locale # check for UTF-8

sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8

locale # verify settings
```

5.1.2 Add the ROS 2 apt repository

You will need to add the ROS 2 apt repository to your system.

```
sudo apt install software-properties-common
sudo add-apt-repository universe
```

```
sudo apt update && sudo apt install curl -y
export ROS_APT_SOURCE_VERSION=$(curl -s https://api.github.com/
    repos/ros-infrastructure/ros-apt-source/releases/latest |
    grep -F "tag_name" | awk -F\" '{print $4}')
curl -L -o /tmp/ros2-apt-source.deb "https://github.com/ros-
    infrastructure/ros-apt-source/releases/download/${
    ROS_APT_SOURCE_VERSION}/ros2-apt-source_${
    ROS_APT_SOURCE_VERSION}.$(. /etc/os-release && echo
    $VERSION_CODENAME)_all.deb" # If using Ubuntu derivates use
    $UBUNTU_CODENAME
sudo dpkg -i /tmp/ros2-apt-source.deb
```

5.1.3 Install ROS 2 packages

ROS 2 packages are built on frequently updated Ubuntu systems. It is always recommended that you ensure your system is up to date before installing new packages.

```
sudo apt update
sudo apt upgrade
```

```
sudo apt install ros-humble-desktop
sudo apt install ros-dev-tools
```

Don't forget to add the setup script to your environment

```
# Replace ".bash" with your shell if you're not using bash
# Possible values are: setup.bash, setup.sh, setup.zsh
source /opt/ros/humble/setup.bash
```

5.2 Installing UGV-drivers

The development of ROS is always in a workspace, in this case the `ugv_ws`, which typically is installed in your home-directory:

```
cd
git clone -b ros2-humble-develop https://github.com/waveshareteam
    /ugv_ws.git
```

Before you can build this workspace, you first have to install some dependencies

```
sudo apt-get install ros-humble-nav2-msgs ros-humble-map-msgs
sudo apt-get install ros-humble-nav2-costmap-2d
sudo apt-get install ros-humble-rosbridge-suite
sudo apt-get install ros-humble-nav2-bringup
sudo apt-get install ros-humble-usb-cam ros-humble-depthai-*
sudo apt-get install ros-humble-joint-state-publisher-*
sudo apt-get install ros-humble-robot-localization
sudo apt-get install ros-humble-imu-tools
sudo apt-get install ros-humble-cartographer-ros
sudo apt-get install ros-humble-apriltag ros-humble-apriltag-msgs
     ros-humble-apriltag-ros
sudo apt-get install ros-humble-ros-gz
```

In addition, add `export UGV_MODEL=ugv_rover` and `export LDLIDAR_MODEL =ld19` to your `~/.bashrc` file.

```
source ~/.bashrc
```

Now you can build the code in `ugv_ws`:

```
cd ~/ugv_ws
colcon build --cmake-args -Wno-dev --packages-select cartographer
     costmap_converter_msgs explore_lite
```

```
colcon build --cmake-args -Wno-dev --packages-select
    openslam_gmapping slam_gmapping --executor sequential
colcon build --cmake-args -Wno-dev --packages-select
    rf2o_laser_odometry robot_pose_publisher teb_msgs vizanti
    vizanti_cpp vizanti_demos vizanti_msgs vizanti_server
colcon build --cmake-args -Wno-dev --packages-select
    ugv_base_node ugv_interface ugv_bringup ugv_chat_ai
    ugv_description ugv_gazebo ugv_nav ugv_slam ugv_tools
    ugv_vision ugv_web_app --symlink-install
source install/setup.bash
```

To be checked: are there no `ros-humble-*` for none of those packages? Should they all be installed from source? Moved the package `ugv_base_node` `ugv_interface` to be in the last part (with the ugv-drivers), which works.

5.3 RoboStack

If you have an Ubuntu 24.04, Ubuntu 20.04, Windows or Mac laptop, we recommend to use RoboStack.

For ROS humble in a RoboStack environment, we follow the steps of Official setup.

For Unix-like platforms like Mac & Linux

```
curl -L -O "https://github.com/conda-forge/miniforge/releases/
    latest/download/Miniforge3-$(uname)-$(uname -m).sh"
bash Miniforge3-$(uname)-$(uname -m).sh
```

Choose all default options. Activate conda with:

```
eval "$~/miniforge3/bin/conda shell.bash hook)"
```

Next step is to install Mamba

```
conda install mamba -c conda-forge
```

Next is to create an environment to install the ROS packages in:

```
mamba create -n ros_humble_env python=3.11
mamba init
source ~/.bashrc
mamba activate ros_humble_env

# this adds the conda-forge channel to the new created
    environment configuration
conda config --env --add channels conda-forge
# and the robostack channel
conda config --env --add channels robostack-staging
# remove the defaults channel just in case, this might return an
    error if it is not in the list which is ok
conda config --env --remove channels defaults

mamba deactivate
```

Next, install ROS humble

```
mamba activate ros_humble_env
mamba install ros-humble-desktop
```

Now, you have the same functionality as ROS humble running on Ubuntu 22.04, as long as you don't forget to use the environment created:

```
mamba activate ros_humble_env
```

In this environment you should install any other `ros-humble-*` package with `mamba` instead of `apt-get`

```
mamba install ros-humble-*
```

For instance, the dependencies of the ugv_ws:

```
mamba install ros-humble-nav2-msgs ros-humble-map-msgs ros-humble
    -nav2-costmap-2d
mamba install ros-humble-image-geometry
mamba install ros-humble-rosbridge-suite
mamba install ros-humble-nav2-bringup
mamba install ros-humble-joint-state-publisher-*
mamba install ros-humble-robot-localization
```

You can always come back to your usual environment with:

```
mamba deactivate
```