Rocky Report

Author:

Tomás Garrido (103134) tomas.garrido@tecnico.ulisboa.pt

2025 October

1 Introduction

The objective of this work was to evaluate practical methods for developing applications using the Unitree Go2 robot (hereafter referred to as Rocky). The main goals included establishing stable communication between the robot and a development workstation, accessing onboard sensor data such as camera streams, LiDAR measurements, and IMU readings, using perception algorithms—particularly YOLO-based object detection—and performing mapping and autonomous navigation experiments.

Two development approaches were analyzed:

- Unofficial ROS2 SDK (go2_ros2_sdk): https://github.com/abizovnuralem/go2_ros2_sdk
- Official Unitree SDK and developer documentation: https://support.unitree.com/home/en/developer

This report summarizes the setup process, practical workflow, encountered issues, and recommendations for future development.

2 Communication Modes

The Unitree Go2 supports two primary communication modes.

2.1 CycloneDDS (Wired Ethernet)

When connected directly to the educational board via Ethernet, communication is performed using CycloneDDS. This configuration provides a stable and low-latency connection, making it particularly suitable for motor-level testing and real-time control experiments. Since it does not depend on wireless network quality, it is the recommended option for latency-sensitive tasks, control algorithm validation, and debugging communication-related issues.

2.2 WebRTC (Wireless)

Wireless communication is achieved through WebRTC over Wi-Fi. This method enables real-time video streaming and simplifies the physical setup, as no Ethernet cable is required.

To establish a WebRTC connection:

1. Connect to the robot using the official Unitree mobile application.
2. Open Settings.
3. Access network information.
4. Retrieve the robot IP address.

When using this sdk set the environment variables:

```
export ROBOT_IP="192.168.X.X"
export CONN_TYPE="webrtc"
```

The main development was performed using the repository: https://github.com/abizovnuralem/go2_ros2_sdk

2.3 Environment Setup

Recommended environment:

- Ubuntu 22.04
- ROS2 Humble
- Python 3.10 (avoid newer versions if dependency issues appear)

Installation:

```
mkdir -p ros2_ws/src
cd ros2_ws/src
git clone --recurse-submodules https://github.com/abizovnuralem/go2_ros2_sdk.git
cd ..
colcon build --symlink-install
source install/setup.bash
```

2.4 Launching the System

```
export ROBOT_IP="192.168.X.X"
export CONN_TYPE="webrtc"
ros2 launch go2_robot_sdk robot.launch.py
```

Once the environment variables defining the robot IP address and connection type are set, the system can be launched using the provided ROS2 launch file. This process initializes the robot communication node, camera publishers, LiDAR publishers, the joint state broadcaster, and an RViz2 visualization instance.

2.5 YOLO Object Detection

Real-time object detection and tracking are also available in the *go2_ros2_sdk* repository. This implementation, originally developed by J. Francis, integrates a COCO-pretrained object detection model through PyTorch and TorchVision.

After launching the robot interface, the RGB image stream becomes available on the topic:

```
/go2_camera/color/image
```

The detection node can then be executed from a separate terminal:

```
source install/setup.bash
ros2 run coco_detector coco_detector_node
```

The detector subscribes to the image topic and publishes detection results to:

`/detected_objects`

Messages contain, for each detected object, the class identifier (`class_id`), a confidence score between 0 and 1, and bounding box information including the centroid coordinates in pixels (`bbox.center.x` and `bbox.center.y`). Detection messages can be inspected with:

```
ros2 topic echo /detected_objects
```

An annotated image stream with bounding boxes and labels can also be visualized:

```
ros2 run image_tools showimage --ros-args -r /image:=/annotated_image
```

The node allows runtime configuration parameters, such as enabling GPU acceleration and adjusting the detection threshold:

```
ros2 run coco_detector coco_detector_node --ros-args \
  -p publish_annotated_image:=False \
  -p device:=cuda \
  -p detection_threshold:=0.7
```

Detection results were successfully streamed in real time.

2.6 Mapping and Navigation

The SDK integrates with:

- slam_toolbox
- Nav2

Procedure:

1. Launch SLAM.
2. Move the robot manually.
3. Save generated map.
4. Load map for autonomous navigation tests.

Indoor mapping tests were performed. Although it is possible to save a map and use it to navigate the robot, this feature is not fully developed in this SDk. The robot struggled to follow a trajectory defined in the map and to avoid obstacles. To perform this task, it is recommended to use the official app, which, when tested, allowed for the planning of a path and enabled the robot to follow it.

2.7 How to use WebRTC to move the robot

To validate low-level motion control over the WebRTC interface, a custom Python script was developed using the https://github.com/abizovnuralem/go2_ros2_sdk. The script initializes a `Go2DriverNode`, establishes a WebRTC connection with the robot, and sequentially issues high-level sport-mode commands. The execution sequence consists of: (1) stand-up, (2) transition to balance stand mode (required for velocity control), (3) execution of a predefined gesture ("Hello"), and (4) a timed rotational motion command.

The rotation behavior is implemented by continuously publishing velocity commands generated through `gen_mov_command()`, specifying zero linear velocity and a non-zero angular velocity. Commands are sent at approximately 10 Hz for a duration of five seconds, after which a zero-velocity command is issued to stop the robot safely.

This script demonstrates direct command transmission through the WebRTC adapter and validates the correct integration between asynchronous Python execution, ROS2 initialization, and Unitree's sport-mode command interface. It also serves as a template for implementing more advanced closed-loop control behaviors.

This test script is available in the Rocky repository: https://github.com/Garrido75/Rocky_report.

3 Official Mobile Application

The official Unitree application allows:

- Manual teleoperation.
- Live video monitoring.
- Mapping and path following.
- Network configuration.
- Firmware updates.

Some remote connection features may require an account balance when the robot is not on the same local network.

What's free and how to charge your account

Some features of the official Unitree mobile application require payment. In particular, remote connections established when the robot and the controlling device are not on the same local area network (LAN) may incur usage charges. Similarly, if the robot relies on mobile data (e.g., 4G connectivity) due to the absence of a local internet connection, additional fees may apply.

It is important to note that development performed locally using either the unofficial ROS2 SDK or the official SDK does not generate any charges.

Recharging the robot's account is not fully automated and requires contacting Unitree customer service through the official shop website (https://shop.unitree.com) or by email. After the request is submitted, customer support provides a dedicated purchase link to add

credit to the robot's account. In some cases, manual clarification regarding billing or shipping address information may be required, even though the transaction concerns a digital service. Once payment is completed, the balance is updated by Unitree. Although the process may take some time, customer support has been responsive and helpful throughout the procedure.

4 Recommended Development Strategy

For continued development, it is recommended to use WebRTC for routine experimentation due to its flexibility and simplified setup, while reserving wired Ethernet communication for latency-critical control validation and debugging. In the future, adding a dedicated wireless interface to the educational board, which may further enhance flexibility when using the official SDK.

This video may help to perform this change: https://www.youtube.com/watch?v=X_LPNRlHOnk&list=PLxQx8K02En6vuXuK7Z--5DnuzBf2i_xma&index=1

References

repository for Unofficial SDK: https://github.com/abizovnuralem/go2_ros2_sdk
manual: https://support.unitree.com/home/en/developer
git hub Rocky: https://github.com/Garrido75/Rocky_report