Unitree Go2 Quick Setup

Mauricio Matamoros

July 9, 2025

1 Requirements

To interface the Go2 you need a linux computer with `ssh` installed and a network cable (hereinafter addressed as the *link computer*). However this will only allow you to work *in situ*, which is not convenient.

To code nodes for the Go2 you'll also need to have ROS2 installed along with all the required compilers and tools (refer to the ROS2 tutorial for your distro for this purpose). In theory any ROS2 distro from Foxy on is capable of interfacing with the Go2, but the developed packages might not run at all into the Go2 should you use any distro other than Foxy. Thus, having Ubuntu 20.04 LTS with ROS2 Foxy Fitzroy as corss-development environment is **highly** recommended. Likewise, it is recommended to use a clean Ubuntu installation to prevent `colcon`[1] from mixing `python3` versions and rendering you able to build but unable to run nodes.

2 First steps in 5 minutes

To connect with the Go1 do follow these simple steps:

i) Plug an ethernet cable to the Go1 and your link computer.
Note: The Go1 does ****NOT**** have integrated wireless.

ii) Setup your wired interface with the fixed IP address `192.168.123.1`, for example with:
`ip addr add 192.168.123.1/24 dev eth0`
but replacing `eth0` with the name of the interface you'll be using.

iii) Connect to the Go1 via `ssh`
`ssh unitree@192.168.123.18`
providing the `123` password.

iv) Upon connecting, the Go2 queries whether to source Ros Foxy (1) or Ros Noetic (2). Choose option one.

v) Souce ROS:
`source /opt/ros/foxy/setup.bash`

vi) Execute the following command: `ros2 topic pub /sportmodecmd <msg_type> '<args>'`

3 Environment setup

The steps defined in Section 2 will create a temporary wired connection that will no longer exist on reboot. To avoid this you should create a permanent profile to quickly connect with the Go1 at convenience.

Another important step consists on sharing your internet connection with the Go1 to a) `apt update` the system, b) `apt install` new packages, c) `git pull` code updates from your repositories, and d) update the system's time. Since the Go1 does not feature an internal CR2032 battery to keep the system's clock running the system's time is always set to January 1, 1970 on every power-on. Most compilers complain about building files in the future, keeping the time in sync shortens the amount of messages dumped to `stderr`.

[1] `colcon` is the building tool used by ROS2 to make nodes.

First, Section 3.1 details how to set up a permanent network connection between the Go2 and the link computer. Then, Section 3.3 provides a snippet to share the link computer's wireless internet connection (or form any other adapter) with the Go2. Finally, Section 3.4 presents a convenience script to automate connections with the Go2. For the ROS Foxy Fitzroy installation please refer to the official installation guide.

3.1 Permanent local-area network connection

The first step consists on retrieving the name of your wired adapter, which can be easily done with `ifconfig` or `ip link show`. For example:

```
$ ifconfig -s
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
enp8s0            1500        0      0      0 0             0      0      0      0 BMU
lo               65536   112156      0      0 0        112156      0      0      0 LRU
wlp9s0            1500 14209550      0     12 0      12130714      0      0      0 BMRU
```

yields that our adapter is called `enp8s0`.

The next step consists in finding out whether your distro comes with Network Manager or can be set up the old way. To find out execute this command:

```
$ which nmtui-edit
/usr/bin/nmtui-edit
```

If the former command yields no output whatsoever, skip to Section 3.2.

Now we proceed to execute `nmtui-edit`. The next steps detailed below are also illustrated in Figure 1.

i) Using the navigation keys select `Add` to add a new connection.
ii) Select `Ethernet` as the connection type.
iii) Give a name to te connection such as `Unitree-Go2`.
iv) Set IPv6 to `<Disabled>`
v) Set IPv4 to `<Manual>` with the following settings:
   - Addresses → Add... → `192.168.123.1`
   - Gateway → `192.168.123.1`
   - DNS Servers → Add... → `132.248.10.2`
   - DNS Servers → Add... → `8.8.8.8`
   - Check `Never use this network for default route`

vi) Confirm changes by selecting `Ok`.
vii) Close the application with `Quit`.

After that the network interface should be set up when the link becomes available (unplug and reconnect the network cable).

3.2 Permanent local-area network connection, the old way

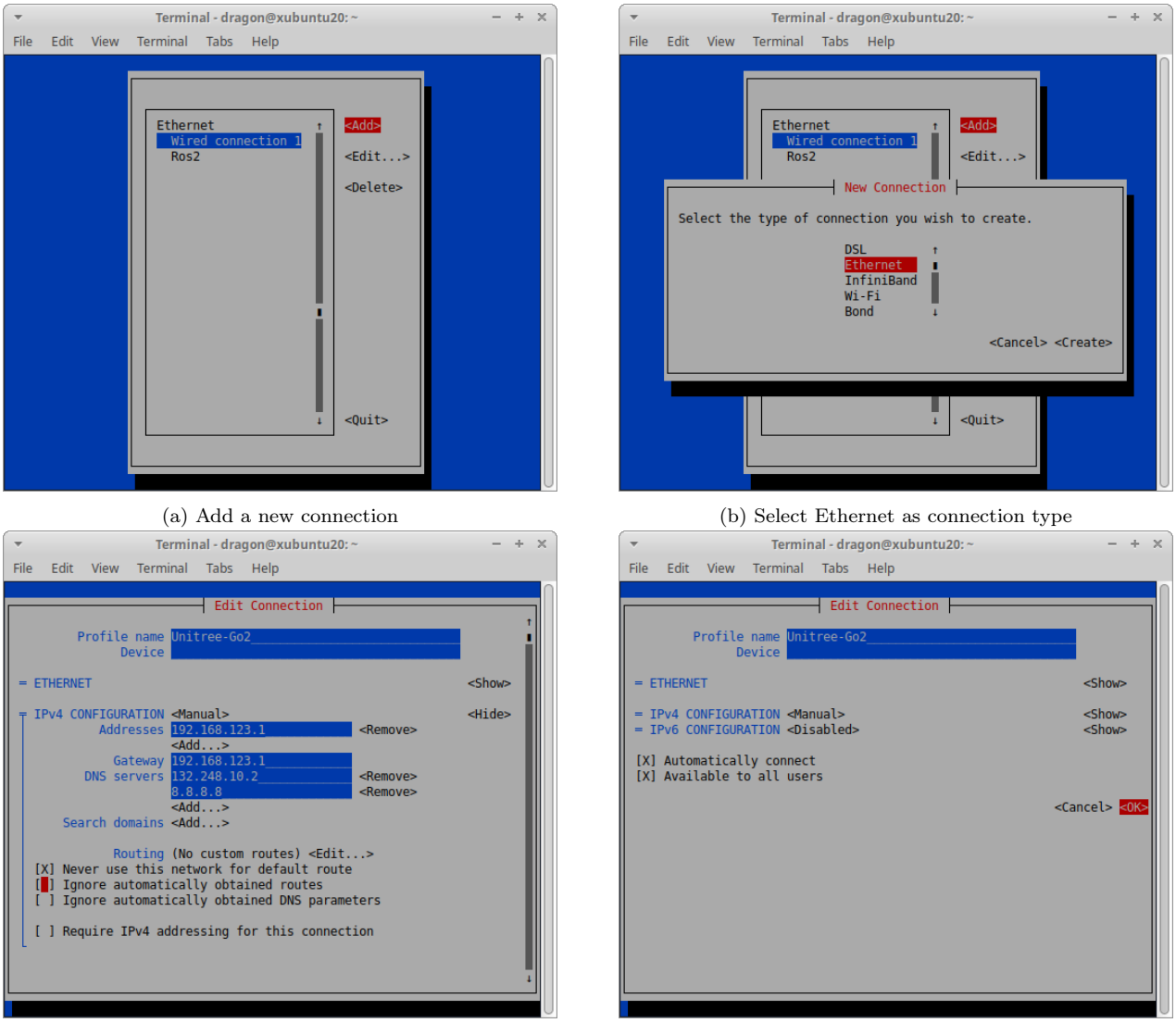
Setting up a network connection in pre-Network Manager times is as easy as writing a configuration file.

Type the following command in the console, making sure to replace `eth0` with the name of your network adapter:

```
$ sudo cat <<EOF > /etc/network/interfaces
auto eth0
iface eth0 inet static
address 192.168.123.1
gateway 192.168.123.1
netmask 255.255.255.0
network 192.168.123.0
broadcast 192.168.123.255
EOF
```

To apply the new settings just restart the network daemon:

```
$ sudo /etc/init.d/networking restart
```



(a) Add a new connection

(b) Select Ethernet as connection type

(c) Fill in all IPv4 settings

(d) Ok to save

Figure 1: Setting up a wired connection with `nmtui-edit`

Note: In case of mistakes, you can always edit the file `/etc/network/interfaces` with `vim` or `nano`. Should there be a `/etc/network/interfaces.d` directory, it is always possible to create a file with the name of the network interface inside said directory to achieve the same effect.

3.3 Sharing the internet connection

Unitree recommends upgrading the system and the `unitree_sdk2` to the latest version to avoid possible problems. As explained in Section 3, the easiest way to get an internet in the Go2 is by sharing the internet access of the link computer. For this purpose, the following script is provided. Do replace the name of the interfaces at your convenience.

route-internet-go2.sh

```
#!/usr/bin/env bash

# Interface with internet connection
WLAN0=wlan0
# Interface connected to the Go1
ETH0=eth0

sudo bash -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
iptables -t nat -A POSTROUTING -o ${WLAN0} -j MASQUERADE
iptables -A FORWARD -i ${ETH0} -o ${WLAN0} -j ACCEPT
iptables -A FORWARD -i ${WLAN0} -o ${ETH0} -m state --state RELATED,ESTABLISHED
```

For convenience, and to prevent interfering with other network connections, internet sharing changes are not made permanent. Thus, the script must be executed before connecting to the Go2 after every reboot.

To disable internet sharing afterwards, suffices with executing the line:

```
$ sudo bash -c 'echo 0 > /proc/sys/net/ipv4/ip_forward'
```

3.4 Automating ssh connections

The script presented below will wait for the Go2 to be online before connecting to it via ssh.

connect-go2.sh

```
#!/usr/bin/env bash

# Go2 IP Address
GO2_IP=192.168.123.18

echo -n "Waiting for Unitree robot"
while ! ping -c 1 -n -w 1 ${GO2_IP} &> /dev/null
do
  echo -n "."
done
echo -e "\nConnected!"
ssh unitree@${GO2_IP}
```

4 Moving the Go2 with ROS' `joy_node`

Although documentation claims it is possible to move the Go2 by means of the `/sportsmode` topic, all provided examples make use of a custom API. This guide builds on top of said API to expose the high-level presets and functions that operate the Unitree Go2. It is worth notice that we will be using a wired USB controller.
The steps to operate the Go2 with ROS Foxy Fitzroy are the following:

1) Connect to the Go2 robot and choose 1 (Ros Foxy) as default environment.

2) Install the joystick tools and the `joy` ROS package.

```
$ sudo apt update && sudo apt install joystick ros-foxy-joy
```

3) Create a workspace directory and navigate into it, for example:

```
$ mkdir -p test_ws/src
$ cd test_ws/src
```

4) Inside `src`, clone the required repositories

```
$ git clone -b ft/go2-base --depth 1 --single-branch \
      https://github.com/kyordhel/unitree-go2-demos.git go2_demo
$ git clone --depth 1 https://github.com/kyordhel/go2_joystick_teleop.git joystick_teleop
```

5) Navigate inside `go2_demo` and retrieve the vendor's ROS packages

```
$ cd go2_demo
$ ./fetch-unitree-pkgs.sh
```

6) Move back into the workspace's root directory, source ROS, and build the example

```
$ cd ../..
$ source /opt/ros/foxy/setup.bash
$ colcon build -symlink-install
```

7) Source the workspace once the packages finish building

```
$ source install/setup.bash
```

8) Before proceding any further, we must test that the control node executes correctly. Run it with

```
$ ros2 run go2_demo dogbase
```

   The Go2 should move slightly. Kill the node with `CTRL+C` and the Go2 should lay down.

If the `dogbse` node executes successfully and the Go2 moves (and there is no reason for it not to) you are ready to proceed, with the next steps to operate the robot with a gamepad via ROS. It's worth noticing that the following steps can be executed either in the Go1 itself or in the link computer. Steps 9) to 13) follow an *in situ* approach.

9) Plug the USB gamepad/joystick into the robot and re-launch the control node:

```
$ ros2 run go2_demo dogbase
```

   The Go2 should stand up again.

10) Open another two terminal windows; then SSH-login into the Go1, enter the workspace directory, and source on each terminal:

```
$ ssh unitree@192.168.123.18
$ cd test_ws && source /opt/ros/foxy/setup.bash && source install/setup.bash
```

11) Identify the device index of your gamepad/joystick[2] with `joy_enumerate_devices` on terminal 2

```
$ ros2 run joy joy_enumerate_devices
Joystick Device ID : Joystick Device Name
-----------------------------------------
                 0 : Generic USB Joystick
```

12) Launch `joy_node` on terminal 2 specifying the device index as parameter

```
$ ros2 run joy joy_node 0
```

13) Finally, on terminal 1, run `joystick_teleop/` node

```
$ ros2 run joystick_teleop joystick_teleop
```

You should now be able to move the Go2 with the main analogue stick of your game control.

4.1 Customizing Gamepad's mapping

To customize your joystick/gamepad button and axis bindings to high-level presents you need to know how linux understands your device through the `jstest` utility. But first, it is necessary to get the device name of the input device. We achieve this by listing all devices in `/dev/input`:

```
$ ls /dev/input
by-id  by-path  event0  event1  event2  event3  event4
event5  event6  event7  js0  mice  mouse0  mouse1
```

In the example above, the gamepad is listed as `js0`. Thus we can proceed to execute `jstest` with the path of the device as parameter:

```
$ jstest /dev/input/js0
Driver version is 2.1.0.
Joystick (Generic   USB  Joystick  ) has 7 axes (X, Y, Z, Rx, Rz, Hat0X, Hat0Y)
and 12 buttons (Trigger, ThumbBtn, ThumbBtn2, TopBtn, TopBtn2, PinkieBtn, BaseBtn, BaseBtn2,
BaseBtn3, BaseBtn4, BaseBtn5, BaseBtn6).
Testing ... (interrupt to exit)
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0
Buttons:  0:off  1:off  2:off  3:off  4:off  5:off  6:off  7:off  8:off  9:off 10:off 11:off
```

A new line will be printed with each change in the device, so output can be a bit verbose. It is highly recommended to take note on the mappings for the axes and buttons of the pysical and the logical device.

Now open the `test_ws/src/joystick_teleop/joystick_teleop/joystick_teleop.py` file with your preferred plain text editor such as vim or nano, and locate the `joyCallback` function shown in Listing 1:

Listing 1: Function `joyCallback` in `joystick_teleop.py`

```
1  def joyCallback(self, msgJoy):
2      handled = self.handleButtons(msgJoy)
3      if handled or self.__busy:
4          return
5
6      leftStickX  = msgJoy.axes[0]
7      leftStickY  = msgJoy.axes[1]
8      rightStickX = msgJoy.axes[2]
9      rightStickY = msgJoy.axes[3]
10
11     lx, az = self.axis2lxaz(leftStickX, leftStickY)
```

The important function here is `axis2lxaz()` that takes two paramenters: the values of the X and Y axes to use as values in the range $[-1, 1]$; and returns a tuple with the X-speed of the robot (forward and backward) as well as the rotation over the Z axis. It is inside this function where you can adjust the displacement and rotation speed of the Go1 (see Listing 2).

[2] If your joystick/gamepad is not listed, it might be due to lack of permissions or because the device is not supported by `joy`. To know whether it is lack of permissions, execute the `groups` command and look for the group `input` in the list. Should this group not be listed, run `sudo usermod -a -G input $USER` and reboot the Go1. However, if `input` is listed, then your gamepad is not compatible with `joy`.

Listing 2: Function `axis2lxaz` in `joystick_teleop.py`

```
def axis2lxaz(self, x, y):
    magnitude = math.sqrt(x**2 + y**2)
    if magnitude < 0.1:
        return 0.0, 0.0
    lx = (1.6 if self.__turbo else 0.8) * y
    az = (1.2 if self.__turbo else 0.9) * x
    return lx, az
#end def
```

Likewise, the preset-button mapping can be customized in the `handleButtons` function of the same file. In these function, button-bindings that should supress publishing a speed command shall return `True`. Presets for tricks are published as a string to the `go2_trick` topic that is received and further interpreted by the `dogbase` node.