TurtleBot3 Setup Guide

Contents

1. Custom TurtleBot3 Image Builder

1.1. Outline

This tool utilises Packer and Podman to automate the creation of Raspberry Pi 4 images for TurtleBot3 robots. The tool can be found [here](#). *Disclaimer:* this system was designed on Linux, YMMV on Windows or MacOS but most instructions are transferable.

A pre-made image for the TB3 Burger (LDS-03, PiCamera (`libcamera` stack)) can be found [here](#). Make sure to follow the Section 1.4 instructions once you have flashed the MicroSD card (see Section 1.3.3). For this pre-built image, the username is `tb` and the password is `password`; you should change the password to something else. The image is built with the University of Southampton's IoT network as a placeholder; after booting, edit the file `/etc/netplan/50-cloud-init.yaml` and input your network details.

1.2. Prerequisites
- **Python 3.11+**
- **Podman** (Containerised Packer execution)
- **Dependencies**: `pip install -r requirements.txt`

1.3. Usage Pipeline

1.3.1. Configuration
Customise the build via TOML files in `configs/`. Multiple WiFi networks can be defined for robust connectivity.

`toml`

```toml
[model]
type = "waffle" # or "burger"

[ros]
domain_id = 42

[[network]]
ssid = "Swarm_Lab_5G"
password = "secure_password"
```

See the `configs/example.toml` to view a complete example.

> **Changing the image size and ISO**
>
> It is not recommended to change the ISO that the config points to and the target image size (unless adding new packages/files).

1.3.2. Build Process

Run the build script to generate the .img (or compressed .img.xz) file.

python

```python
python build.py --config configs/my_robot.toml --verbose
```

The build may take over 1 hour to build and customise the image.

1.3.3. Deployment

Flash the image to a MicroSD card. Ensure you identify the correct device node using `fdisk -l`.

bash

```bash
xz -dc <IMAGE>.img.xz | sudo dd of=/dev/<DEVICE> status=progress
```

After flashing, expand the main partition to fill the MicroSD card. I'd recommend a tool like GParted to expand it to fill the entire MicroSD card.

1.4. Post-Installation

- Log in with the credentials used in your config file
- First Boot: The system automatically configures the hostname (based on MAC address), OpenCR firmware, and Pi Camera. A reboot is required after the first initialisation. **Do not reboot immediately**—wait a few minutes for each process to finish.
- Use the system as normal.

1.5. Key Automated Features

- ROS2 Humble Hawksbill & TurtleBot3 stack installation.
- OpenCR firmware configuration via boot-time service.
- MAC-based Hostname: Automatically sets turtlebot_XX_XX_XX.
- `libcamera` setup for Pi Camera stack
- Netplan configuration for multiple pre-defined SSIDs.

1.5.1. Automatic Bringup

This image has a service to automatically run `ros2 launch turtlebot3_bringup robot.launch.py`. To enable this, run the following.

bash

```bash
systemctl enable bringup.service # Run on boot
systemctl start bringup.service # Run now
systemctl status bringup.service # Check it launched with no errors
```

2. Setting up Zenoh with Tailscale for ROS 2 TurtleBot3

2.1. Prerequisite

You should have a working Turtlebot3 Waffle with the correct image. If you do not have this, read the previous section. You will also need a virtual machine to run the remote environment.

You will need a VM to run on the remote PC; you can download VMWare Workstation Pro for Windows [here](#).

2.2. Setup Tailscale on VM and TurtleBot3

1. Create a Tailscale account using your Gmail account or GitHub
2. Install Tailscale on the VM:

```bash
curl -fsSL https://tailscale.com/install.sh | sh
```

3. Start Tailscale to add the VM as a device:

```bash
sudo tailscale up
```

4. Install Tailscale on the TurtleBot3 and add it to the same Tailscale network.
5. Connect the TurtleBot3 to Tailscale and enable SSH:

```bash
sudo tailscale up --ssh
```

6. Ensure you connect using the correct username:

```bash
<username>@computername:~$
```

2.3. Setup Zenoh on VM and TurtleBot3

1. Install Zenoh (RMW implementation) on both devices:

```bash
sudo apt update && sudo apt install ros-humble-rmw-zenoh-cpp
```

2. Confirm the installation:

```bash
ros2 doctor --report | grep rmw
```

You should see rmw_zenoh_cpp listed.

3. Initialise the ROS 2 environment and set the Zenoh middleware:

```bash
source /opt/ros/humble/setup.sh
source ~/.bashrc
export RMW_IMPLEMENTATION=rmw_zenoh_cpp
```

4. Verify middleware configuration:

```bash
ros2 doctor --report | grep middleware
```

Expected output:

```bash
middleware name: rmw_zenoh_cpp
```

A warning may appear. This is normal.

2.4. Starting the Zenoh Middleware

One device must run as the router, the other as a client. We recommend running the router on the VM and having the robot as a client.

2.4.1. On the router machine

1. Stop any existing ROS services:

```bash
pkill -9 -f ros
ros2 daemon stop
```

2. Start the Zenoh router:

```bash
ros2 run rmw_zenoh_cpp rmw_zenohd
```

3. Open a new terminal and get the router's Tailscale IP:

```bash
tailscale ip
```

2.4.2. On the client machine

1. Set the Zenoh client configuration, replacing with the router's Tailscale IP:

```bash
export ZENOH_CONFIG_OVERRIDE='mode="client";connect/endpoints=["tcp/<router_tailscale_ip>:7447"]'
```

2. Confirm connectivity:

```bash
ros2 doctor --report | grep middleware
```

Output should show only:

```bash
middleware name: rmw_zenoh_cpp
```

2.5. Testing with Talker–Listener

1. Verify the ROS domain ID on both devices:

```bash
echo $ROS_DOMAIN_ID
```

2. If blank or different, set them to the same value:

```bash
export ROS_DOMAIN_ID=<domain_id_integer>
```

3. On the router machine:

```bash
source /opt/ros/humble/setup.sh
source ~/.bashrc
ros2 run demo_nodes_cpp talker
```

4. On the client machine:

```bash
ros2 run demo_nodes_cpp listener
```

Messages should be received.

2.6. Manual Camera Setup

> **Setting Up the Camera Again**
>
> If you previously tried to set up the camera, make sure to uninstall all dependencies and remove the `libcamera` repository beforehand.

The instructions to set up the camera manually differs slightly from the ROBOTIS instructions. First install the dependencies (make sure not to install Meson or Ninja from `apt`).

```bash
sudo apt update
```

```bash
sudo apt install -y python3-pip git python3-jinja2 libboost-dev libgnutls28-dev openssl libtiff-dev pybind11-dev qtbase5-dev libqt5core5a libqt5widgets5 cmake python3-yaml python3-ply libglib2.0-dev libgstreamer-plugin1.0-dev
sudo apt install ros-humble-camera-ros
pip install meson ninja
```

Then we can clone the repository and build the package.

```bash
mkdir -p ~/turtlebot3_ws/src && cd ~/turtlebot3_ws/src
git clone -b v0.5.2 https://github.com/raspberrypi/libcamera.git
cd libcamera
meson setup build --buildtype=release -Dpipelines=rpi/vc4,rpi/pisp -Dipas=rpi/vc4,rpi/pisp -Dv4l2=true -Dgstreamer=enabled -Dtest=false -Dlc-compliance=disabled -Dcam=disabled -Dqcam=disabled -Ddocumentation=disabled -Dpycamera=enabled
ninja -C build -j 2
ninja -C build install -j 2
```

Then add the following line to you `~/.bashrc` file.

```bash
export LD_LIBRARY_PATH=/usr/local/lib/aarch64-linux-gnu:$LD_LIBRARY_PATH
```

Restart the robot and test the camera.

2.7. Camera Test

On the Turtlebot:

```bash
export RMW_IMPLEMENTATION=rmw_zenoh_cpp
ros2 launch turtlebot3_bringup camera.launch.py format:=RGB888 width:=640 height:=480
```

Recommended formats:

```bash
BGR888 (64x64 - 3280x2464)
RGB888 (64x64 - 3280x2464)
```

On the Remote PC:

```bash
ros2 run rqt_image_view rqt_image_view
```

Select /camera/image_raw/compressed.

2.8. Image Transport Optimisation

Raw image topics are very high bandwidth (approximately 27 MB/s for 640×480 RGB8 at 30 FPS).

Install plugins:

```bash
sudo apt install ros-humble-image-transport-plugins
```

Python launch file node:

```python
from launch_ros.actions import Node

image_compressed_republisher = Node(
    package='image_transport',
    executable='republish',
    name='image_compressed_republisher',
    arguments=[
        'raw',
        'compressed',
        '--ros-args',
        '-r', 'in:=/camera/image_raw',
        '-r', 'out:=/camera/image_raw/compressed'
```

```
    ],
    output='screen'
)

ld.add_action(image_compressed_republisher)
```

Verify:

```bash
ros2 topic list | grep compressed
```

2.9. Useful ROS 2 Commands

```bash
ros2 topic hz <topic_name>
ros2 topic bw <topic_name>
ros2 topic echo <topic_name> | head
ros2 run rqt_image_view rqt_image_view
ros2 run image_tools showimage --ros-args -r image:=<camera_image_topic>
```

Always source after changes:

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

2.10. Troubleshooting Common Issues

1. Old ROS processes:

```bash
pkill -9 -f ros
ros2 daemon stop
```

2. Environment not sourced.
3. Zenoh environment variables not exported.
4. Incorrect Tailscale IP:

```bash
tailscale ip
```

5. Middleware errors:

```bash
ros2 doctor --report | grep middleware
```

6. Talker–Listener issues.
7. Error treating timestamp for received data.

First check clock status on both:

```bash
timedatectl status
```

Look for: System clock synchronised: yes NTP service: active

If not synchronised, enable it:

```bash
sudo timedatectl set-ntp true
```

Then verify again:

```bash
timedatectl status
```

3. Version History

Version	Date	Edited By
v2.2	2026-02-19	TG
v2.1	2026-02-19	TG
v2.0	2026-02-18	TG, AO
v1.1	2026-02-18	TG
v1.0	2026-02-17	TG, AO