

COURSE BRIEF / ROS 2 JAZZY / 21 OCTOBER 2026

Nodes to Maps

A practical introduction to ROS 2, from the first message to a phone-sensor mapping project.

This course is both a five-lecture crash course and a reusable self-learning guide. Students learn how small programs exchange typed data, how to build and inspect those programs, and how to assemble a sensor pipeline they can explain and debug.

Platform	Teaching format	Capstone
ROS 2 Jazzy Ubuntu 24.04 LTS	Five lectures 9 guided hours, plus setup	Android phone + laptop Optional monocular SLAM

What learners will be able to do

- Inspect nodes, topics, message types and connections with the ROS command line.
- Build Python and C++ publishers/subscribers and use generated custom interfaces.
- Choose between topics, services and actions; configure nodes with parameters and launch files.
- Explain QoS compatibility, coordinate frames, transforms and timestamp-dependent data.
- Receive phone camera and motion data in ROS, inspect it, and record/replay a rosbag.
- Describe calibration and demonstrate sparse monocular mapping after the backend and phone have been qualified.

Audience and preparation

No robotics background is assumed. Learners need basic Python functions and classes; C++ scaffolds support the third lecture. Setup takes about 1-2 hours, with a separate instructor rehearsal for the phone lab. A laptop with 16 GB RAM and 30 GB free disk is comfortable. Use an Android 8.0+ phone. The same native app connects by wired USB or wireless over the phone hotspot.

Timing that matches the current course

Lectures 1-4 take 90 minutes each. The full fifth lecture takes 180 minutes, for nine contact hours. For five 90-minute meetings, teach phone transport and a prepared mapping demonstration in the final meeting, then assign the full mapping run as a follow-up lab. Setup is additional.

This brief summarizes the supplied course edition. Simulated teaching screens explain expected behavior; real-device and live mapping qualification remains a separate step.

LECTURE PLANS / 01-02

Understand the graph. Then build it.

Each session combines an explanation, a working example, a deliberate fault and evidence that the learner can interpret.

LECTURE 01 / 90 MIN

Think in nodes and messages

Outcome: Inspect typed streams and diagnose a broken connection.

Session sequence	Minutes
Mental model: nodes, topics, publishers and subscribers	15
Turtlesim and command-line inspection	25
Names, remapping and debugging	20
Hands-on graph lab	20
Checkpoint and reasoning	10

Practical work: Drive turtlesim, inspect its graph and read a message interface. Remap one command endpoint, observe the broken connection and repair it using graph evidence.

Checkpoint: Identify which component publishes or subscribes, explain the type contract, and show how names affect connectivity.

LECTURE 02 / 90 MIN

Write your first Python nodes

Outcome: Build a publisher, subscriber and generated interface.

Session sequence	Minutes
Workspace and package structure	10
Complete Python publisher	20
Packaging and executable entry points	15
Subscriber and custom interfaces	20
Hands-on implementation lab	25

Practical work: Create rclpy nodes with timers and callbacks, register an installed executable, build/source the workspace, and exchange a meaningful custom message.

Checkpoint: Run the installed nodes and explain the difference between source code, a Node object, a package and an executable. Explain when callbacks run and how interfaces are generated.

LECTURE PLANS / 03-04

Make the system communicate reliably.

The same ROS graph supports multiple languages and interaction styles. Learners connect those ideas to timing and geometry.

LECTURE 03 / 90 MIN

The same graph, in C++

Outcome: Connect C++ to Python and reason about QoS.

Session sequence	Minutes
Translate familiar node concepts into C++	15
C++ publisher/subscriber and CMake build	25
Quality of Service compatibility	20
Break delivery, inspect and repair	20
Checkpoint	10

Practical work: Build the supplied rclcpp examples, connect them to Python peers and compare reliability settings. Inspect offered/requested QoS when a typed topic is visible but messages fail to arrive.

Checkpoint: Demonstrate cross-language delivery and justify a compatible QoS choice using endpoint evidence. The session provides a C++ scaffold rather than a full language course.

LECTURE 04 / 90 MIN

Connect a complete system

Outcome: Choose interactions and transform a measurement between frames.

Session sequence	Minutes
Service request/response	15
Action goals, feedback, results and cancellation	15
Parameters and launch files	20
TF frames, point transforms and time	25
Checkpoint	15

Practical work: Call an area service, run and cancel a counting action, configure and launch nodes, and work through a rotated/translated point. Relate map, odom, base and optical frames to lookup direction and time.

Checkpoint: Choose a topic, service or action for a task; explain a source-to-target transform and why a phone camera-to-IMU transform requires measured geometry.

LECTURE PLAN / 05

A phone that helps build a map

A 180-minute guided capstone brings sensing, networking, ROS inspection, geometry and recording into one reproducible experiment.



The phone app sends camera frames and independent accelerometer/gyroscope samples to the laptop. The ROS bridge validates and timestamps them, then publishes images, camera information, motion and diagnostics. A separate ORB-SLAM3 adapter consumes calibrated images.

Session sequence	Minutes
Pipeline and clocks	25
Pair the Android app; inspect ROS topics	30
Frames and calibration review	20
Break	10
Mapping and rosbag recording	50
Diagnose faults using evidence	25
Discussion, report and extensions	20

What happens in the lab

- Pair using a reachable secure WebSocket endpoint, session token and server certificate fingerprint. Choose Option 1: wired USB with ADB reverse, or Option 2: wireless over the phone hotspot. WSL forwarding is documented for both.
- Display the camera stream, inspect rates and native motion components, and identify the phone camera timestamp source.
- Load calibration measured for the transmitted camera mode. Initialize mapping with slow translation through a textured scene, then record and replay a short consented run.

What students submit

Phone/app identity, camera mode and clock source; calibration and backend versions; commands and rate evidence; a bag manifest/checksum; one diagnosed failure; and a real map screenshot/trajectory when mapping succeeds. Rubric: transport 25%, timing/frames/calibration 30%, reproducibility 25%, diagnosis 20%.

Scope and extension path

The baseline produces sparse monocular landmarks with arbitrary scale. IMU streams support inspection; the supplied mapper uses images. Camera-IMU fusion needs measured extrinsics, timing/noise qualification and an integration stage. Dense metric reconstruction is a further project. Prepare TLS, backend compilation, calibration and a reference recording before teaching this session.

THE WEBSITE / LEARNING EXPERIENCE

One guide, two ways to learn

The static website uses the same authoritative Markdown notes as the standalone reading bundle.

Self-study mode	Lecture mode
Full reading guide, exercises, quiz feedback and model explanations. Students can revisit sections and follow linked references.	Section-by-section presentation with navigation controls and a timer. Instructors use the same examples and checkpoints during class.

Tools supplied to every learner

- Search across course topics; navigate lessons, resource pages and section outlines.
- Use interactive concept labs for the ROS graph, communication behavior, transforms and phone bandwidth.
- Answer 32 scenario-based quiz questions with explanatory feedback, plus 12 written prompts with model explanations and evidence criteria.
- Save section progress, personal notes and written responses locally in the browser; export progress/notes from Resources.
- Copy code and commands, inspect 48 source files, download individual source files and use printable reading views.
- Compare expected behavior with 18 SVG diagrams/simulated screens and one conceptual equipment illustration. The illustrations are labelled; actual results depend on the learner's system.

Six development and extension annexes

Annex	What it covers
A / Android app	APK installation, Camera2, SensorManager, timestamps, secure transport and builds.
B / ROS receiver	Packet validation, image decoding, clock conversion and ROS publication.
C / Calibration	Checkerboard camera calibration, axes, camera-IMU geometry and inertial qualification.
D / ROS workflows	Services, actions, bags, TF, robot descriptions and further projects.
E / USB or wireless	Two routes for the same Android APK: USB with ADB reverse, or Wi-Fi over the phone hotspot.
F / WSL 2	Ubuntu on Windows, USB/ADB, mirrored Wi-Fi, scoped forwarding and graphics checks.

Supporting resources

The 15 resource pages include these six annexes plus a CLI field guide, frames/transforms reference, phone-lab preparation, client/receiver contracts, visual gallery, troubleshooting, official references/glossary, course scope and learning outcomes, and worked solutions/rubric.

LEARNING MATERIALS / GETTING STARTED

Your course materials

Use the website during lessons and the downloadable notes and code for independent practice.

Provided material	Purpose
Static website ZIP	Host the full guide with illustrations, learning modes and linked downloads.
Illustrated Markdown notes ZIP	Read lessons, annexes and assessments offline, with images and linked artifacts.
Android test APK + checksum	Development-signed classroom sensor client for Android 8.0+.
Android Studio source	Inspect or modify the native app and follow its build instructions.
ROS workspace ZIP	Four beginner packages: interfaces, Python, C++ and bringup.
Optional SLAM adapter ZIP	Monocular ROS adapter, calibration converter and preparation guides; external engine required.

Prepare your learning environment

Start with Ubuntu 24.04 and ROS 2 Jazzy. Follow the setup guide to build the supplied workspace and run your first nodes. Install the Android app before the phone-sensor project.

For the capstone, connect the phone by USB or wireless, inspect its camera and motion topics, and calibrate the exact camera mode. The optional mapper needs a separately prepared ORB-SLAM3 engine and vocabulary. Follow the phone lab guide for recording and replay.

Choose how to study

Open the course in self-study mode for complete explanations and exercises, or use lecture mode to follow a session. Download the illustrated Markdown notes for offline reading. Save your progress and notes in the browser, and export them from Resources.

Further reading

Continue learning with the official references linked throughout the course:

[ROS 2 Jazzy documentation](#)

[Android Camera2](#)

[WSL networking](#)

[ORB-SLAM3 project](#)

Course text and teaching illustrations: CC BY 4.0. Core/phone-client code: Apache-2.0. Optional SLAM adapter: GPL-3.0-or-later. Preserve the supplied attribution and licenses.