FRONTIER · TAUGHT WITH ROS

Robotics

ROS 2, control loops, mapping and navigation. You drive a simulated robot, make it find its own way across a mapped space, and learn why the physical one behaves differently.

Foundational | 5 phases | 25 sessions | 38 contact hours

Syllabus

5 phases, 25 sessions. Each phase ends in something you have built.

1 Before Anything Moves

Get a Linux machine, a shell, and enough Python to write a loop that decides something.

4 sessions
$58 alone

1.1 What a Robot Actually Is

Sense, decide, act, and why every robot you build is that loop running fast.

FREE PREVIEW

1.2 Ubuntu the Way ROS Expects It

Install Ubuntu 24.04 in a VM or on metal, and get around it from the terminal alone.

1.3 Python That Does Not Stop

Variables, functions, loops, and a program that keeps running until you kill it.

1.4 A Controller Before ROS

Error, gain, and a proportional controller you write with no framework underneath it.

By the end: **An Ubuntu 24.04 machine you set up yourself, running a proportional controller you wrote in plain Python.**

2 The Graph From the Command Line

Inspect and drive a running robot without writing a line of code.

5 sessions
$58 alone

2.1 Sourcing, and Why Nothing Works Yet
Source the setup file, set ROS_DOMAIN_ID, and fix the two failures that stop most beginners.

2.2 Nodes and Turtlesim
Start nodes with ros2 run, list what is running, and move a robot with ros2 topic pub.

2.3 Topics, Services and Actions
The three ways nodes talk, told apart by watching each one live from the CLI.

2.4 Parameters and rqt
Retune a running node without restarting it, and read its logs in rqt_console.

2.5 Launch Files and Bags
Start a whole system with one command, record everything it said, and play it back.

By the end: **A recorded bag of a turtlesim run, replayed, with a written inventory of every node, topic and service in it.**

3 Code You Wrote

Build your own packages and close a control loop inside a ROS 2 node.

5 sessions
$58 alone

3.1 Workspaces, colcon and Overlays
Build a workspace, source the overlay in a fresh terminal, and see what breaks when you do not.

3.2 Your First Package
ros2 pkg create, package.xml, rosdep, and an entry point that actually launches.

3.3 Publisher and Subscriber in rclpy
A node, a timer, a callback, and two programs you wrote exchanging messages.

3.4 Your Own Interfaces and Parameters
Custom msg and srv files, declared parameters, and a node you can retune from the CLI.

3.5 Closing the Loop in a Node
Move your Python controller into a node, launch it with parameters, and reach the goal.

By the end: **A colcon package whose rclpy node reads a goal from a parameter and drives the robot to it.**

4 A Body and a World

Describe a robot precisely enough that the simulator, RViz and a planner all agree where it is.

5 sessions
$58 alone

4.1	**Frames and tf2**	base_link, odom and map, and reading the transform tree with tf2_echo when it breaks.
4.2	**URDF and Xacro**	Links, joints, inertia, and a robot model RViz renders without a single warning.
4.3	**Gazebo Harmonic**	SDF worlds, spawning your robot, and why Gazebo Classic stopped being an option in 2025.
4.4	**The ros_gz Bridge**	Get simulated lidar, IMU and odometry out of Gazebo and onto real ROS topics.
4.5	**Odometry and Its Drift**	Wheel odometry, an EKF from robot_localization, and measuring exactly how wrong you are.

By the end: **Your own URDF robot driving in Gazebo Harmonic with a clean TF tree and live lidar in RViz 2.**

5 Autonomy, Then Reality

Give the robot a goal instead of a velocity, then price what a physical one costs.

6 sessions
$58 alone

5.1	**Mapping with slam_toolbox**	Build a map from lidar and odometry while driving, then save it to disk.
5.2	**Localizing in a Known Map**	AMCL, setting an initial pose, and telling a lost robot apart from a drifting one.
5.3	**Navigating with Nav2**	Costmaps, planner, controller, behavior tree, and a goal sent from RViz.
5.4	**When Nav2 Gets Stuck**	Recovery behaviors, costmap tuning, and diagnosing a failed run from its recorded bag.
5.5	**Off the Simulator**	micro-ROS on a microcontroller base, ros2_control, calibration, and an E-stop within reach.
5.6	**What Learning Adds, and What It Costs**	Imitation learning, RL and sim-to-real named honestly, with the GPU bill each one brings.

By the end: **A robot that maps an unknown world and navigates it unattended, plus one recorded failure with a written cause.**

Tools you will use

ROS 2 Jazzy | colcon | rclpy | Gazebo Harmonic | RViz 2 | slam_toolbox | Nav2 | micro-ROS

What you will build

1. **The Robot You Described**
   Your URDF in Gazebo Harmonic, lidar live in RViz 2
2. **Map, Then Go**
   slam_toolbox into Nav2, hands off the keyboard
3. **One Real Robot**
   A micro-ROS base taking cmd_vel and publishing real odometry

Registration

Phase 1: Before Anything Moves	$58
Phase 2: The Graph From the Command Line	$58
Phase 3: Code You Wrote	$58
Phase 4: A Body and a World	$58
Phase 5: Autonomy, Then Reality	$58
Whole track, all 5 phases	**$240**

Buying every phase separately costs $290, so the whole track saves $50.

No payment is taken online. Registering creates an order and payment instructions are sent with it. Access opens once payment is confirmed.