

Robotics Career Guide

A practical playbook for getting your first robotics role, or your next one: which jobs exist, what each one actually needs, how to build a portfolio that gets interviews, and how to prepare for the technical rounds.

Who this is for: students, software engineers moving into robotics, mechanical and electrical engineers who want to work closer to autonomy, and working roboticists planning a move.

How to use it: read section 1 to pick a target role, then use sections 3 to 6 as a checklist. You do not need everything here; you need the right subset for one role.

Contents

1. The robotics job map
2. Core skills, by role
3. The tools you will be asked about
4. A portfolio that gets interviews
5. Your resume and application
6. Technical interviews: what to expect
7. Switching into robotics from another field
8. Industries and where the work is
9. A 90-day plan
10. Checklists

1. The robotics job map

"Robotics engineer" is not one job. Most postings fall into one of the roles below, and the skills that win one can be almost irrelevant to another. Pick one or two target roles before you study or apply; it makes every later decision easier.

Role	What you do day to day	What hiring managers look for
Robotics software engineer	Builds the software that ties a robot together: drivers, middleware, state machines, task logic, tooling, deployment.	Strong C++ and/or Python, ROS 2, Linux, debugging on real hardware, clean code that others can maintain.
Perception engineer	Turns camera, lidar, radar and depth data into a model of the world: detection, segmentation, tracking, mapping.	Computer vision, 3D geometry, point clouds, deep learning, sensor calibration, data pipelines.
Localization and mapping (SLAM) engineer	Answers "where am I?" with state estimation, sensor fusion and map building.	Probability, Kalman filters and factor graphs, optimization, Lie groups for rotations, IMU and odometry models.
Motion planning engineer	Decides how the robot moves: paths, trajectories, collision avoidance, manipulation plans.	Search and sampling-based planners, optimization, kinematics, collision checking, real-time constraints.
Controls engineer	Makes the robot follow the plan accurately and stably: joint, force and whole-body control.	Control theory (PID, state space, LQR, MPC), dynamics, system identification, tuning on hardware.
Robot learning / ML engineer	Trains policies and models: imitation learning, reinforcement learning, foundation models for robots.	PyTorch, training at scale, simulation, data collection, evaluation rigor, reading and reproducing papers.
Mechanical / mechatronics engineer	Designs the physical robot: mechanisms, actuators, structures, end effectors, enclosures.	CAD, tolerance analysis, materials, design for manufacture, prototyping, test fixtures.
Electrical / embedded engineer	Designs boards, motor drives, power systems and the firmware that runs on them.	Schematic and PCB design, embedded C, RTOS, communication buses (CAN, SPI, I2C, EtherCAT), power electronics.
Simulation engineer	Builds simulated worlds and robots for testing and training.	Physics engines, scene and asset pipelines, sensor simulation, performance, sim-to-real validation.
Field / test / integration engineer	Gets robots working at customer sites and in test: bring-up, triage, data capture, reliability tracking.	Systematic debugging, Linux and networking, scripting, clear incident reports, calm under pressure.
Automation / industrial robotics engineer	Programs and integrates industrial arms and cells in factories and warehouses.	Vendor robot languages, PLCs, safety standards, fixtures and end-of-arm tooling, cycle-time optimization.

Tip. Read ten real postings for your target role and tally the requirements. The three or four that appear in most of them are your study plan. The long tail of "nice to haves" usually is not.

Seniority, roughly

- **Entry level:** can implement a well-scoped component with guidance and debug it on hardware.
- **Mid level:** owns a subsystem end to end, makes sound trade-offs, and unblocks themselves.
- **Senior:** designs across subsystems, anticipates failure modes in the field, and raises the level of the team.
- **Staff and principal:** sets technical direction across teams and is trusted with the hardest, most ambiguous problems.

2. Core skills, by role

Every robotics role shares a foundation. Above it, each specialty adds its own depth. Use this section to find gaps.

The shared foundation

- **Programming:** Python for tooling, data and prototyping; C++ for anything that runs on the robot under time pressure. Modern C++ (smart pointers, move semantics, RAII, templates in moderation) matters more than clever tricks.
- **Linear algebra and geometry:** vectors, matrices, coordinate frames, homogeneous transforms, and at least one rotation representation you are fluent in (rotation matrices and quaternions).
- **Probability:** Gaussians, Bayes' rule, and why sensor noise means every estimate has uncertainty.
- **Linux and the command line:** processes, networking basics, logs, permissions, shell scripting, and reading other people's build systems.
- **Version control and testing:** Git workflows, code review, unit tests for the logic, and integration tests in simulation.
- **Debugging real systems:** forming a hypothesis, isolating variables, and using logs, plots and recordings rather than guessing.

Depth by specialty

Perception

- Camera models, intrinsics and extrinsics, calibration
- Epipolar geometry, stereo, structure from motion
- Point cloud processing: filtering, registration (ICP), segmentation
- Detection and segmentation networks, dataset curation, evaluation metrics

Localization and SLAM

- Kalman, extended and unscented filters; particle filters
- Factor graphs and nonlinear least squares
- Visual and lidar odometry, loop closure
- IMU preintegration and time synchronization

Planning

- Graph search (A^* , D^* Lite), sampling (RRT, PRM) and their variants
- Trajectory optimization, smoothing, time parameterization
- Forward and inverse kinematics, Jacobians, singularities
- Behavior trees and finite state machines for task logic

Controls

- PID done properly: anti-windup, filtering, feedforward
- State space, controllability, LQR, model predictive control
- Rigid-body dynamics, impedance and force control
- Discretization, latency and sample-rate effects

3. The tools you will be asked about

Tools change, but a few appear in a large share of postings. Being comfortable with them signals that you can be productive in week one.

Tool or area	Why it matters	How to show it
ROS 2	The most common robotics middleware. Nodes, topics, services, actions, parameters, launch files, TF2 for coordinate frames, QoS settings.	A project with several nodes, a launch file, and a TF tree you can explain.
Simulation (Gazebo, Isaac Sim, MuJoCo, Webots)	Most testing and a lot of learning happens in sim before hardware.	A simulated robot doing a task end to end, with a short write-up of what broke on the way.
Movel2	Standard toolkit for manipulator motion planning in ROS 2.	A pick-and-place demo with a planning scene and collision objects.
Nav2	Standard navigation stack for mobile robots in ROS 2.	A mobile robot navigating a mapped environment with tuned costmaps.
OpenCV and PCL / Open3D	Core image and point cloud processing libraries.	A calibration or registration pipeline with measured accuracy.
PyTorch	Dominant framework for perception and robot learning work.	A trained model with an honest evaluation, not only a demo video.
Docker and CI	Reproducible builds and deployments are expected on most teams.	A repository that builds and runs its tests from one command.
Data and visualization (RViz, Foxglove, rosbag, plotting)	Debugging robots means recording and replaying data.	Plots in your write-ups that explain a failure and its fix.
CAD and embedded (for hardware roles)	SolidWorks, Fusion or Onshape; KiCad or Altium; STM32 / ESP32 class microcontrollers.	Photos, drawings and a bill of materials for something you built.

Do not tool-collect. One project that uses ROS 2, a simulator and one domain library deeply is worth more than a list of fifteen tools you touched once.

4. A portfolio that gets interviews

In robotics, a portfolio often matters more than grades or years of experience. Reviewers want evidence that you can make something work on a real or simulated robot, and that you understand why it works.

What a strong project looks like

1. **A clear problem.** "A mobile robot that docks to its charger within 2 cm" beats "an autonomous robot".
2. **Your own contribution, made obvious.** Say exactly which parts you wrote, and which came from libraries or teammates.
3. **Measurement.** Success rate over N trials, error in centimetres or degrees, latency, cycle time. Numbers turn a demo into engineering.
4. **Failure analysis.** What went wrong, how you found out, what you changed. This is the part interviewers ask about most.
5. **Reproducibility.** A README with setup steps, a short video or GIF, and code that builds.

Project ideas by target role

Target role	Project idea
Robotics software	A small ROS 2 robot (real or simulated) with a behavior tree that runs a multi-step task, plus recorded tests that replay bag files in CI.
Perception	Camera–lidar extrinsic calibration with a reprojection-error report, then object detection fused with depth to output 3D positions.
SLAM	Implement an EKF that fuses wheel odometry and IMU, compare it against ground truth in simulation, then add a landmark update.
Planning	Implement A* and RRT* yourself on a grid and in configuration space, benchmark them, then compare with an off-the-shelf planner.
Controls	Balance a cart-pole or an inverted pendulum with LQR, then MPC, on a simulator and if possible on cheap hardware. Show step responses.
Robot learning	Train an imitation-learning policy from demonstrations for a simple manipulation task in simulation, with success rates across seeds.
Hardware	Design and build a small gripper or actuator module, with CAD, a bill of materials, a test rig and measured performance.

Presenting it

- Put the 30-second version first: one image or GIF, one sentence on the problem, one result number.
- Keep a one-page write-up per project: problem, approach, results, what you would do next.
- Pin your two best repositories. Delete or archive half-finished ones that dilute the signal.
- Competitions and teams count: university robotics teams, autonomous racing, underwater and aerial challenges. Say what you owned.

5. Your resume and application

Resume rules that matter in robotics

- **One page** until you have significant experience; two at most.
- **Lead bullets with outcomes:** "Cut localization drift from 1.2 m to 0.3 m over a 500 m route by adding loop closure" beats "Worked on SLAM".
- **Name the stack** in context: "C++17, ROS 2, Nav2, Gazebo" next to the project that used them, not only in a skills block.
- **Show hardware contact.** Many teams screen for people who have debugged real robots. Say so when you have.
- **Link the portfolio** near the top, and make sure the links work on a phone.

Tailor every application

1. Copy the posting's top four requirements.
2. Make sure each one has a matching bullet or project on your resume, using the posting's words where honest.
3. Write three or four sentences for the cover note: why this company's robot, which of your projects is closest to their problem, and what you would want to work on.

Where applications actually come from

- Specialist robotics job boards and company career pages, checked weekly.
- Referrals: robotics is a small field; a short, specific message to an engineer about their work often gets a reply.
- Open-source contributions to tools the company uses (ROS 2 packages, simulators, drivers). Maintainers notice.
- Conferences, meetups and competitions, where teams recruit directly.

6. Technical interviews: what to expect

Robotics interview loops vary by company, but most combine the stages below. Ask the recruiter which ones you will face, and prepare for those.

Stage	What it tests	How to prepare
Coding	Data structures, algorithms, clean code. Often in C++ for on-robot roles.	Practise in the language you will use. Know vectors, maps, priority queues, graph search, and memory ownership in C++.
Robotics fundamentals	Transforms, rotations, kinematics, probability, filtering.	Be able to derive and explain, not only recite: compose two transforms on a whiteboard, explain a Kalman update.
Domain deep dive	Your specialty at depth: perception, SLAM, planning, controls or learning.	Review the canonical algorithms for your area and their failure modes.
System design	How you would architect a robot or a subsystem: data flow, latency, failure handling, testing.	Practise designing a delivery robot, a pick-and-place cell, or a drone inspection system out loud.
Project deep dive	Whether you truly built what your resume says.	Know every decision, number and bug in your top two projects.
Behavioral	Collaboration, ownership, how you handle field failures and disagreement.	Prepare five short stories with situation, action and measurable result.

Questions that come up often

- How do you convert between a rotation matrix and a quaternion, and why might you prefer one?
- Given a robot's pose in the world and a point seen by its camera, how do you get the point in world coordinates?
- Walk through the predict and update steps of a Kalman filter. What happens if the noise estimates are wrong?
- When would you choose a sampling-based planner over a search-based one?
- Your robot works in simulation but not on hardware. How do you find out why?
- How would you test a perception system before it ships?
- Design the software for a warehouse robot that picks items from shelves. What fails first?

The sim-to-real question is a favourite because it shows how you debug. A strong answer is systematic: check timing and latency, sensor noise and calibration, actuator limits and delays, unmodelled friction or contact, and compare recorded data from both worlds.

7. Switching into robotics from another field

From software engineering

- Your advantage: code quality, testing, infrastructure and scale. Robotics teams often need these badly.
- Your gap: math (transforms, probability) and real-hardware debugging.
- Bridge: learn ROS 2, build one simulated robot project end to end, then buy or borrow a small robot kit to experience real sensor noise. Target robotics software, tooling, simulation or infrastructure roles first.

From mechanical or electrical engineering

- Your advantage: physical intuition, design for manufacture, test discipline.
- Your gap: software practices, usually C++ and Linux.
- Bridge: write the software for something you designed; learn Git and testing; target mechatronics, controls, embedded or integration roles that value both sides.

From data science or machine learning

- Your advantage: modelling, training and evaluation rigor.
- Your gap: real-time constraints, geometry, and data that comes from sensors rather than tables.
- Bridge: a perception or robot-learning project in simulation with honest evaluation; learn coordinate frames thoroughly.

From the trades, technician and military roles

- Hands-on experience with machines, wiring and maintenance is valuable in field service, test, integration and industrial automation.
- Vendor training on industrial robots and PLCs, plus scripting skills, open a path towards automation engineering.

8. Industries and where the work is

Sector	Typical work	Skills that stand out
Warehouse and logistics	Mobile robots, picking arms, sortation, fleet management.	Navigation, manipulation, fleet software, reliability.
Manufacturing and industrial automation	Robot cells, inspection, welding, assembly.	Industrial arms, PLCs, safety standards, vision inspection.
Autonomous vehicles and trucking	Perception, prediction, planning, simulation at scale.	Deep learning, sensor fusion, large-scale data and evaluation.
Drones and aerial	Inspection, mapping, delivery, defence.	State estimation, flight control, embedded, regulation awareness.
Agriculture	Harvesting, weeding, monitoring in unstructured outdoor settings.	Robust perception outdoors, ruggedized hardware, field work.
Medical and surgical	Surgical systems, rehabilitation, lab automation.	Precision control, safety-critical software, regulated development.
Humanoids and general-purpose robots	Whole-body control, learning-based manipulation, large models for robots.	Robot learning, controls, simulation, large-scale data collection.
Space, defence and underwater	Rovers, manipulators, autonomous vehicles in extreme settings.	Reliability engineering, autonomy under limited communication.
Research labs and universities	Publishing, prototypes, teaching.	Papers, open-source code, depth in one area.

Pay varies a lot by country, city, sector and seniority. Compare offers using current, local salary data for the specific role rather than national averages.

9. A 90-day plan

Days 1–30: choose and close gaps

- Pick one or two target roles using section 1. Collect ten real postings.
- List the requirements that appear in most of them. Rate yourself honestly on each.
- Spend most of your time on the two biggest gaps. For many people that is ROS 2 plus one math area.
- Set up a clean development environment and a public repository you will be proud of.

Days 31–60: build one flagship project

- Choose a project from section 4 that matches your target role.
- Ship a first version early, then spend most of the time measuring, failing and improving.
- Write the one-page write-up and record a short video as you go, not at the end.

Days 61–90: apply and prepare

- Rewrite your resume around the flagship project and its numbers.
- Apply to a focused list each week, tailoring each application. Ask for referrals with specific messages.
- Practise coding in the language you will use, and run mock interviews on fundamentals and system design.
- After each interview, write down every question you were asked and close the gaps.

10. Checklists

Before you apply

- ☐ I can name my target role and why
- ☐ My top two projects have numbers and a failure story
- ☐ Every repository I link builds from its README
- ☐ My resume matches the posting's top four requirements
- ☐ My portfolio link works on a phone
- ☐ I have a three-sentence cover note for this company

Fundamentals I can explain

- ☐ Composing and inverting transforms
- ☐ Rotation matrices and quaternions
- ☐ Forward and inverse kinematics
- ☐ Kalman filter predict and update
- ☐ A* versus RRT trade-offs
- ☐ PID tuning and its limits

Before the interview

- ☐ I know which stages the loop includes
- ☐ I have practised coding in the interview language
- ☐ I can whiteboard my flagship project's architecture
- ☐ I have five behavioral stories with results
- ☐ I have read about the company's robot and its likely hard problems
- ☐ I have three thoughtful questions for them

Evaluating an offer

- ☐ What will I own in my first six months?
- ☐ How much time is on real hardware versus simulation?
- ☐ How are field failures handled, and who is on call?
- ☐ Who will mentor me, and how are engineers reviewed?
- ☐ How does compensation compare with current local data?
- ☐ Is the company funded for the time it needs to ship?

Find your next role. Browse current openings across software, perception, controls, hardware and field roles on Robotics Engineer Jobs, and set up alerts for the roles you are targeting.