UDACITY

NANODEGREE PROGRAM SYLLABUS

# Robotics Software Engineer

# Overview

This program will teach you:

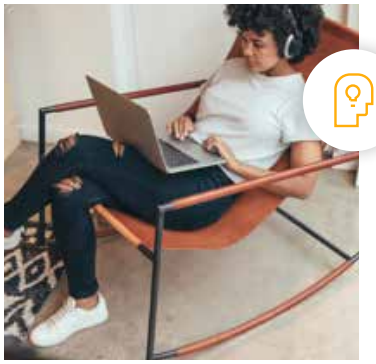
- The software fundamentals to work on robotics using C++, ROS, and Gazebo
- How to build autonomous robotics projects in a Gazebo simulation environment
- Probabilistic robotics, including Localization, Mapping, SLAM, Navigation, and Path Planning.

This program is comprised of 6 courses and 5 projects. Each project you build will be an opportunity to demonstrate what you've learned in the lessons. Your completed projects will become part of a career portfolio that will demonstrate to potential employers that you have skills in C++, ROS, Gazebo, Localization, Mapping, SLAM, Navigation, and Path Planning.

Depending on how quickly you work through the material, the amount of time required is variable. We have included an hourly estimate for each section of the program. If you spend about 10 hours per week working through the program, you should finish in 14 weeks (approximately 4 months).

**Estimated Time**: 4 Months at 10-15hrs/week

**Prerequisites**: Object-oriented programming



**Flexible Learning**: Self-paced, so you can learn on the schedule that works best for you

**Need Help?**
udacity.com/advisor
Discuss this program with an enrollment advisor.

# Course 1: Gazebo World

Learn how to simulate your first robotic environment with Gazebo, the most common simulation engine used by Roboticists around the world.

**Course Project**
Build My World

Use the tools that you've learned in Gazebo to build your first environment.

Key Skills Demonstrated:
- Launching a Gazebo Environment
- Designing in Gazebo

| | LEARNING OUTCOMES | |
|---|---|---|
| **LESSON ONE** | **Introduction to Gazebo** | • Work with the Gazebo simulator to build new environments, and deploy assets. |

# Course 2: ROS Essentials

Discover how ROS provides a flexible and unified software environment for developing robots in a modular and reusable manner. Learn how to manage existing ROS packages within a project, and how to write ROS Nodes of your own in C++.

**Course Project**
Go Chase It!

Demonstrate your proficiency with ROS, C++, and Gazebo by building a ball-chasing robot. You will first design a robot inside Gazebo, house it in the world you have built in the Build My World project, and code a C++ node in ROS to chase yellow balls.
Key Skills Demonstrated:

- Building Catkin Workspaces
- ROS node creation
- ROS node communication
- Using additional ROS packages
- Gazebo world integration
- Additional C++ practice
- RViz Integration

| | LEARNING OUTCOMES | |
|---|---|---|
| **LESSON ONE** | **Introduction to ROS** | • Obtain an architectural overview of the Robot Operating System Framework. |
| **LESSON TWO** | **Packages & Catkin Workspaces** | • Learn the ROS workspace structure, essential command line utilities, and how to manage software packages within a project. |
| **LESSON THREE** | **Write ROS Nodes** | • Write ROS nodes in C++. |

# Course 3: Localization

Learn how Gaussian filters can be used to estimate noisy sensor readings, and how to estimate a robot's position relative to a known map of the environment with Monte Carlo Localization (MCL).

**Course Project**
Where Am I?

You will interface your own mobile robot with the Adaptive Monte Carlo Localization algorithm in ROS to estimate your robot's position as it travels through a predefined set of waypoints. You'll also tune different parameters to increase the localization efficiency of the robot.

Key Skills Demonstrated:

- Implementation of Adaptive Monte Carlo Localization in ROS
- Understanding of tuning parameters required

| | LEARNING OUTCOMES | |
|---|---|---|
| **LESSON ONE** | **Introduction to Localization** | • Learn what it means to localize and the challenges behind it. |
| **LESSON TWO** | **Kalman Filters** | • Learn the Kalman Filter and its importance in estimating noisy data. |
| **LESSON THREE** | **Lab: Kalman Filters** | • Implement an Extended Kalman Filter package with ROS to estimate the position of a robot. |
| **LESSON FOUR** | **Monte Carlo Localization** | • Learn the MCL (Monte Carlo Localization) algorithm to localize robots. |
| **LESSON FIVE** | **Build MCL in C++** | • Code the MCL algorithm in C++ |

# Course 4: Mapping and SLAM

Learn how to create a Simultaneous Localization and Mapping (SLAM) implementation with ROS packages and C++. You'll achieve this by combining mapping algorithms with what you learned in the localization lessons.

**Course Project**
Map My World

Students will interface their robot with an RTAB Map ROS package to localize it and build 2D and 3D maps of their environment. Students must put all the pieces together properly to launch the robot and then teleop it to map its environment.
Key Skills Demonstrated:
- SLAM implementation with ROS/Gazebo
- ROS debugging tools: rqt, roswtf

| | LEARNING OUTCOMES | |
|---|---|---|
| **LESSON ONE** | **Introduction to Mapping and SLAM** | • Learn the Mapping and SLAM concepts, as well as the algorithms. |
| **LESSON TWO** | **Occupancy Grid Mapping** | • Map an environment by coding the Occupancy Grid Mapping algorithm with C++. |
| **LESSON THREE** | **Grid-based FastSLAM** | • Simultaneously map an environment and localize a robot relative to the map with the Grid-based FastSLAM algorithm.<br>• Interface a turtlebot with a Grid-based FastSLAM package with ROS to map an environment. |
| **LESSON FOUR** | **GraphSLAM** | • Simultaneously map an environment and localize a robot relative to the map with the GraphSLAM algorithm. |

# Course 5: Path Planning and Navigation

Learn different Path Planning and Navigation algorithms. Then, combine SLAM and Navigation into a home service robot that can autonomously transport objects in your home!

**Course Project**
Home Service Robot

In this capstone project, you will use a SLAM package to autonomously map an environment. Then, you will interface your robot with a path planning and navigation ROS package to move objects within an environment.
Key Skills Demonstrated:

- Advanced ROS and Gazebo integration
- ROS Navigation stack 7
- Path planning

| | LEARNING OUTCOMES | |
|---|---|---|
| **LESSON ONE** | **Intro to Path Planning and Navigation** | • Learn what the lessons in Path Planning and Navigation will cover. |
| **LESSON TWO** | **Classic Path Planning** | • Learn a number of classic path planning approaches that can be applied to low-dimensional robotic systems. |
| **LESSON THREE** | **Lab: Path Planning** | • Code the BFS and A* algorithms in C++. |
| **LESSON FOUR** | **Sample-Based and Probabilistic Path Planning** | • Learn about sample-based and probabilistic path planning, and how they can improve on the classic approach |

# Course 6: Optional KUKA Path Planning

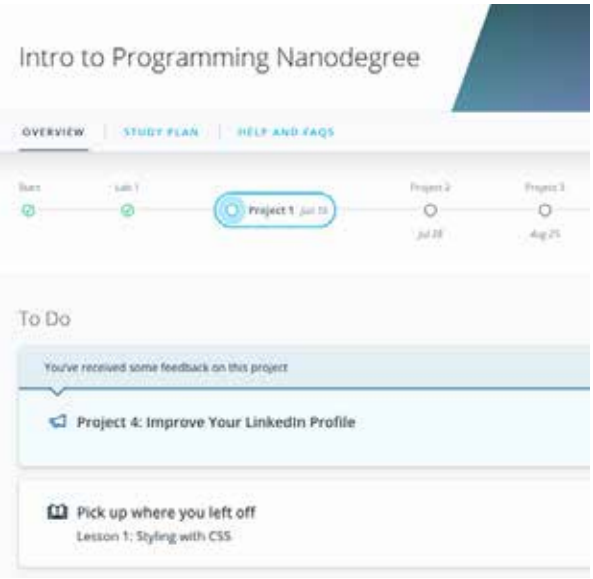
**Optional Course Project**
KUKA Path Planning

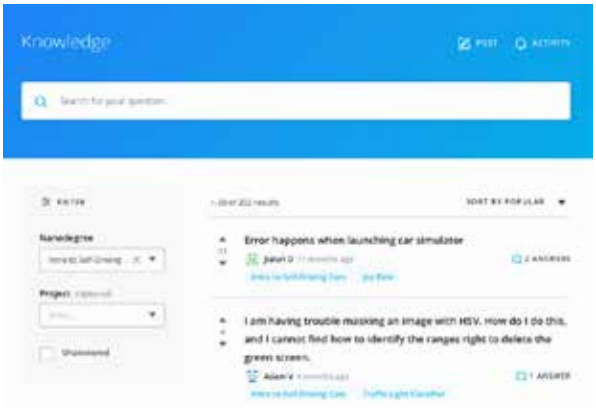
Students will apply what they have learned about ROS and path planning to search for a path and navigate a KUKA robot through a 2D maze.

Key Skills Demonstrated:
- Path planning
- Using C++ and Python with external ROS API

| | LEARNING OUTCOMES | |
|---|---|---|
| **LESSON ONE** | **Project Introduction** | • Learn the requirements of the project. |
| **LESSON TWO** | **Project Details** | • Learn the project specifications and how to get started. |

# Our Classroom Experience





## REAL-WORLD PROJECTS

Build your skills through industry-relevant projects. Get personalized feedback from our network of 900+ project reviewers. Our simple interface makes it easy to submit your projects as often as you need and receive unlimited feedback on your work.

## KNOWLEDGE

Find answers to your questions with Knowledge, our proprietary wiki. Search questions asked by other students, connect with technical mentors, and discover in real-time how to solve the challenges that you encounter.

## STUDENT HUB

Leverage the power of community through a simple, yet powerful chat interface built within the classroom. Use Student Hub to connect with fellow students in your program as you support and learn from each other.

## WORKSPACES

See your code in action. Check the output and quality of your code by running them on workspaces that are a part of our classroom.

## QUIZZES

Check your understanding of concepts learned in the program by answering simple and auto-graded quizzes. Easily go back to the lessons to brush up on concepts anytime you get an answer wrong.

## CUSTOM STUDY PLANS

Preschedule your study times and save them to your personal calendar to create a custom study plan. Program regular reminders to keep track of your progress toward your goals and completion of your program.

## PROGRESS TRACKER

Stay on track to complete your Nanodegree program with useful milestone reminders.

# Learn with the Best

## Sebastian Thrun

INSTRUCTOR

As the founder and president of Udacity, Sebastian's mission is to democratize education. He is also the founder of Google X, where he led projects including the Self-Driving Car, Google Glass, and more.

## David Silver

CURRICULUM LEAD

David Silver leads the School of Autonomous Systems at Udacity. Before Udacity, David was a research engineer on the autonomous vehicle team at Ford. He has an MBA from Stanford, and a BSE in computer science from Princeton.

## Karim Chamaa

COURSE DEVELOPER

Karim started his early career as a Mechanical Engineer. He earned his M.S. in Mechatronics and Robotics Engineering from NYU. His specialties include Kinematics, Control, and Electronics.

## Julia Chernushevich

COURSE DEVELOPER

Julia is an experienced educator and robotics specialist. Her previous work experiences include teaching Mechatronics Engineering at the University of Waterloo and designing electric vehicles for underground mines.

# All Our Nanodegree Programs Include:

### EXPERIENCED PROJECT REVIEWERS

**REVIEWER SERVICES**

- Personalized feedback & line by line code reviews
- 1600+ Reviewers with a 4.85/5 average rating
- 3 hour average project review turnaround time
- Unlimited submissions and feedback loops
- Practical tips and industry best practices
- Additional suggested resources to improve

### TECHNICAL MENTOR SUPPORT

**MENTORSHIP SERVICES**

- Questions answered quickly by our team of technical mentors
- 1000+ Mentors with a 4.7/5 average rating
- Support for all your technical questions

### PERSONAL CAREER SERVICES

**CAREER SUPPORT**

- Resume support
- Github portfolio review
- LinkedIn profile optimization

# Frequently Asked Questions

## PROGRAM OVERVIEW

### WHY SHOULD I ENROLL?

Demand for software engineers with advanced robotics skills far exceeds the current supply of qualified talent. This makes this an ideal time to pursue career advancement in this field, and this program represents a great opportunity to develop and practice core robotics skills such as C++, ROS, and probabilistic robotics algorithms such as Localization, Mapping, SLAM, Path Planning and Navigation.

You will graduate from this Nanodegree program having completed five hands-on robotics projects in the Gazebo simulator; these will serve as portfolio pieces demonstrating your acquired skills to hiring managers and recruiters. These skills will help you pursue and advance a career in the robotics field.

### WHAT JOBS WILL THIS PROGRAM PREPARE ME FOR?

As a Robotics Software Engineer, you'll be equipped to bring value to a wide array of industries and be eligible for many roles.

Your opportunities might include:

- developing pick and place robotics systems for advanced manufacturing.
- developing the next surgical robot for the healthcare industry.
- building the next form of package delivery either on the ground or in the air.
- designing exploratory robots that can be deployed to discover new planets.
- designing rescue robots to assist people in disasters.

### HOW DO I KNOW IF THIS PROGRAM IS RIGHT FOR ME?

The Robotics Software Engineer Nanodegree program is designed for those looking to pursue or advance a career in the robotics field. In this program, you will learn and practice the core robotics skills that employers have told us serve as the foundation for the work robotics engineers do: C++, ROS, Gazebo, and robotics algorithms such as Localization, Mapping, SLAM, Path Planning, and Navigation amongst others.

If you want to work in a field where you get to see your solutions come to life, and solve some of the world's most difficult and exciting problems, the Robotics Software Engineer Nanodegree program is right for you.

### HOW IS THE ROBOTICS SOFTWARE ENGINEER NANODEGREE PROGRAM DIFFERENT FROM YOUR MACHINE LEARNING ENGINEER NANODEGREE PROGRAM OR YOUR SELF-DRIVING CAR ENGINEER NANODEGREE

# FAQs Continued

### PROGRAM?

The Robotics Software Engineer Nanodegree program focuses on teaching the core robotics skills needed for a successful robotics software engineering career. The program focuses on Localization, Mapping, SLAM, Path Planning, and Navigation. These are taught using C++ and the Robot Operating System (ROS) framework.

**The Self-Driving Car Engineer Nanodegree** program focuses entirely on a specialized application of robotics—it uses robotics concepts and applies them to a self-driving car. If your primary interest is in the application of robotics, machine learning, and artificial intelligence to autonomous vehicles, then this is the program for you. However, if you want to learn and practice core robotics skills in C++ and ROS, with an emphasis on robotics algorithms, then the Robotics Software Engineer Nanodegree program is your best option.

### I HAVE GRADUATED FROM THE ROBOTICS SOFTWARE ENGINEER NANODEGREE PROGRAM BUT I WANT TO KEEP LEARNING. WHERE SHOULD I GO FROM HERE?

Both our **Self-Driving Car Engineer** and **Flying Car and Autonomous Flight Engineer Nanodegree** programs address specific areas of robotics and autonomous systems. If you want to continue your education either on the ground or in the air, take one of these exciting Nanodegree programs next!

## ENROLLMENT AND ADMISSION

### DO I NEED TO APPLY? WHAT ARE THE ADMISSION CRITERIA?

There is no application. This Nanodegree program accepts everyone, regardless of experience and specific background.

### WHAT ARE THE PREREQUISITES FOR ENROLLMENT?

To succeed in this Nanodegree program, you should have experience with the following:

- Advanced knowledge in any object-oriented programming language,
- preferably C++
- Intermediate Probability
- Intermediate Calculus
- Intermediate Linear Algebra
- Basic Linux Command Lines

### IF I DO NOT MEET THE REQUIREMENTS TO ENROLL, WHAT SHOULD I DO?

We created the **Intro to Self-Driving Cars Nanodegree** program to help prepare prospective students for more advanced programs. That introductory

# FAQs Continued

program will teach you the skills you need to be successful in the Robotics Software Engineer Nanodegree program, including C++, linear algebra, calculus, and statistics.

## TUITION AND TERM OF PROGRAM

### HOW IS THIS NANODEGREE PROGRAM STRUCTURED?

The Robotics Software Engineer Nanodegree program is comprised of content and curriculum to support five (5) projects. We estimate that students can complete the program in four (4) months, working 10 hours per week.

Each project will be reviewed by the Udacity reviewer network. Feedback will be provided and if you do not pass the project, you will be asked to resubmit the project until it passes.

### HOW LONG IS THIS NANODEGREE PROGRAM?

Access to this Nanodegree program runs for the length of time specified in the payment card above. If you do not graduate within that time period, you will continue learning with month to month payments. See the **Terms of Use** and **FAQs** for other policies regarding the terms of access to our Nanodegree programs.

### WHAT SPECIAL HARDWARE WILL I NEED IN THIS PROGRAM?

The core of this Nanodegree program focuses on robotics applications in software. You will learn and practice robotics skills using the Gazebo simulator.

## SOFTWARE AND HARDWARE

### WHAT SOFTWARE AND VERSIONS WILL I NEED IN THIS PROGRAM?

For this Nanodegree program, you will use the Robot Operating System (ROS) and Gazebo. You will code primarily with C++. These platforms and languages are freely available.

We will provide you with a GPU-enabled Linux Workspace that runs in your browser, and an internet connection is required. Optionally, you can install the Linux image on a Virtual Machine.