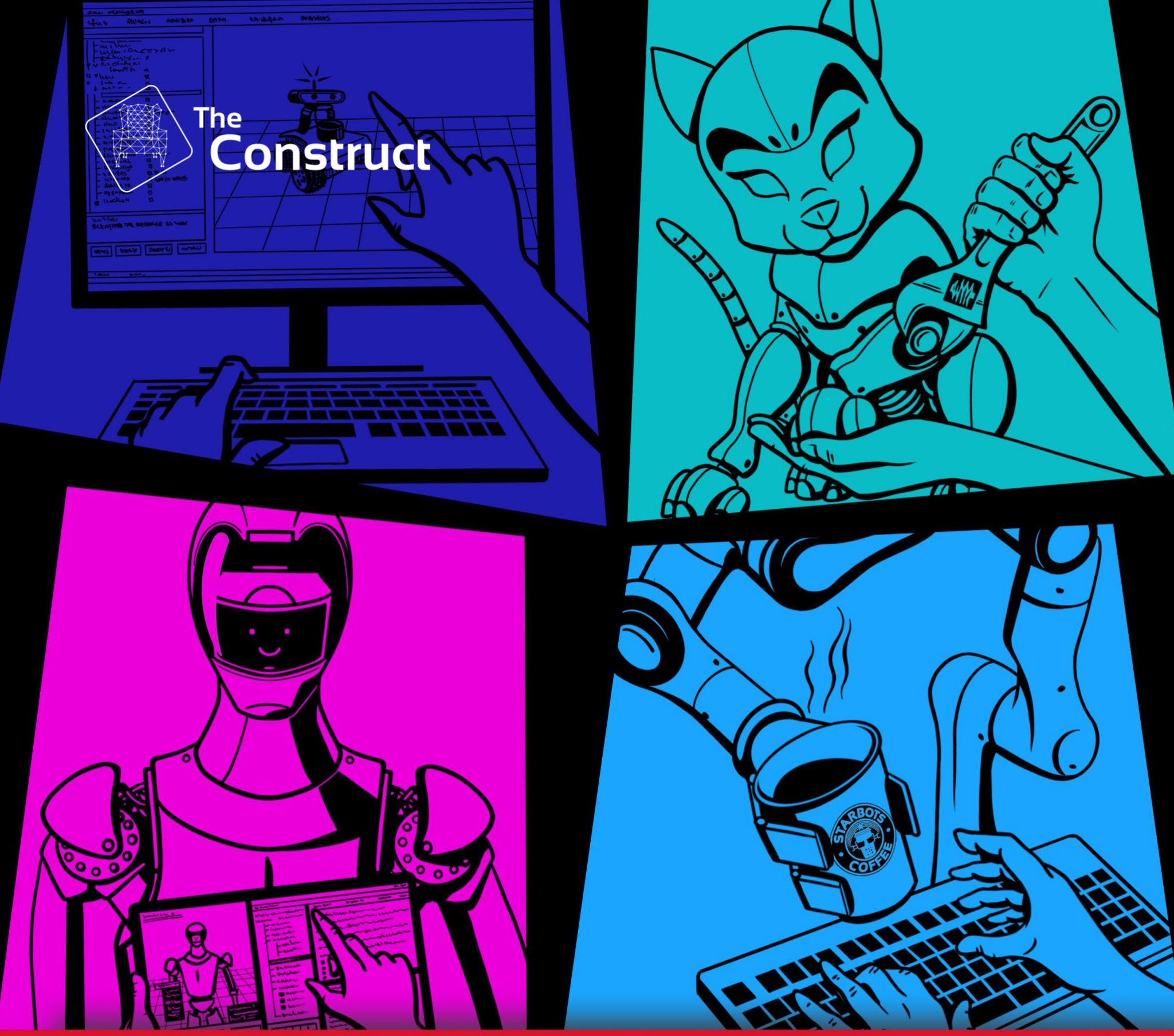


ROBOTICS DEVELOPERS GUIDE

A COMPLETE PRACTICAL GUIDE FOR ROBOTICS DEVELOPERS

Written by Ricardo Tellez
Cover design by Lorena Guevara
Infographics by Yuhong Lin

Copyright of The Construct Sim Ltd. 2024

CHAPTER 1:
WHAT IS A ROBOTICS DEVELOPER

If you are reading this, it is because you are either a robotics developer or you want to become one. A robotics developer is builds programs for robots.

Robotics developers do not need to develop new path-planning algorithms. They don't need to develop a new control paradigm. They don't need to invent a new object recognition system. That is the job of the roboticists and AI at the labs. Robotics developers need to know which algorithms exist, how to use them, when, and how to integrate them into the complete robotics application. Hence, the global robot always does what it is expected to do.

However, programming robots differ from programming an app or a web page. The program's hardware changes continuously (because the robot moves in the world) and often experiences unexpected problems, making the programs more complex than apps.

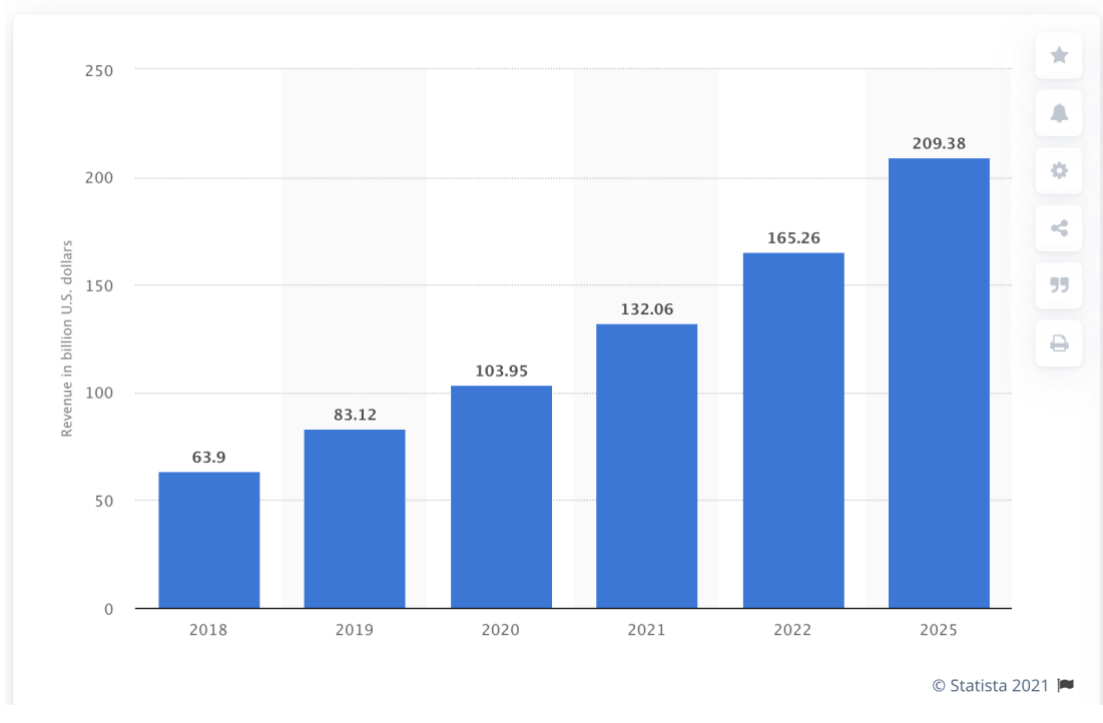
Robotic programs constantly interact with hardware to take action and perceive the result of that action. That goes beyond mere user interaction because now, the program considers the user interaction plus the robot's interaction with the world all that in a continuously changing environment.

WHY BECOME A ROBOTICS DEVELOPER?

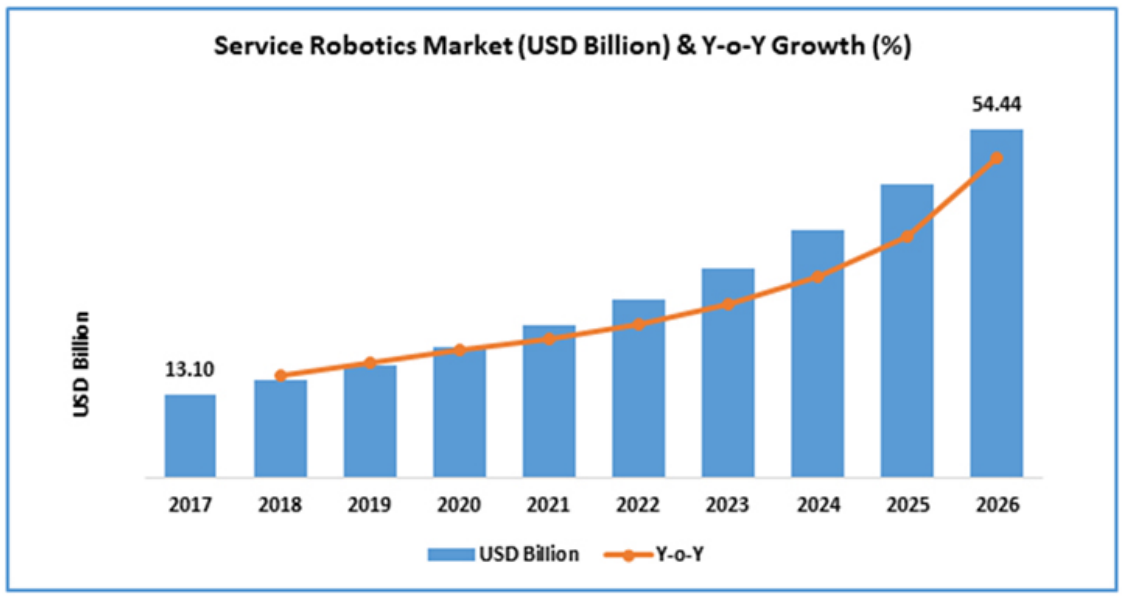
The first question you may have is why you should consider becoming a robotics developer. I have the following answers:

- **Robot programming has a big future**. The number of robots used in the world is increasing exponentially.

(in billion U.S. dollars)



I’m talking about **industrial robots**, and robots targeting commercial and consumer spaces. I’m talking about robots that interact with humans and help them with tasks. I’m talking about **service robots**.



- **More big companies are entering the robotics game**. Companies like Dyson and Tesla are creating robots. Additionally, Google's spin-off Intrinsic bought the Robot Operating System (ROS) team of developers to release Flowstate, their own software product for programming robots. That is a trend.

- **There are not enough engineers…yet**. Due to the increasing demand for robots, many companies now have a robotics division. I know that because every week, more companies join our online academy. They want to train their engineers to program robots. These companies do not have robotics divisions but are considering creating one and cannot find properly trained people that know how to program robots.

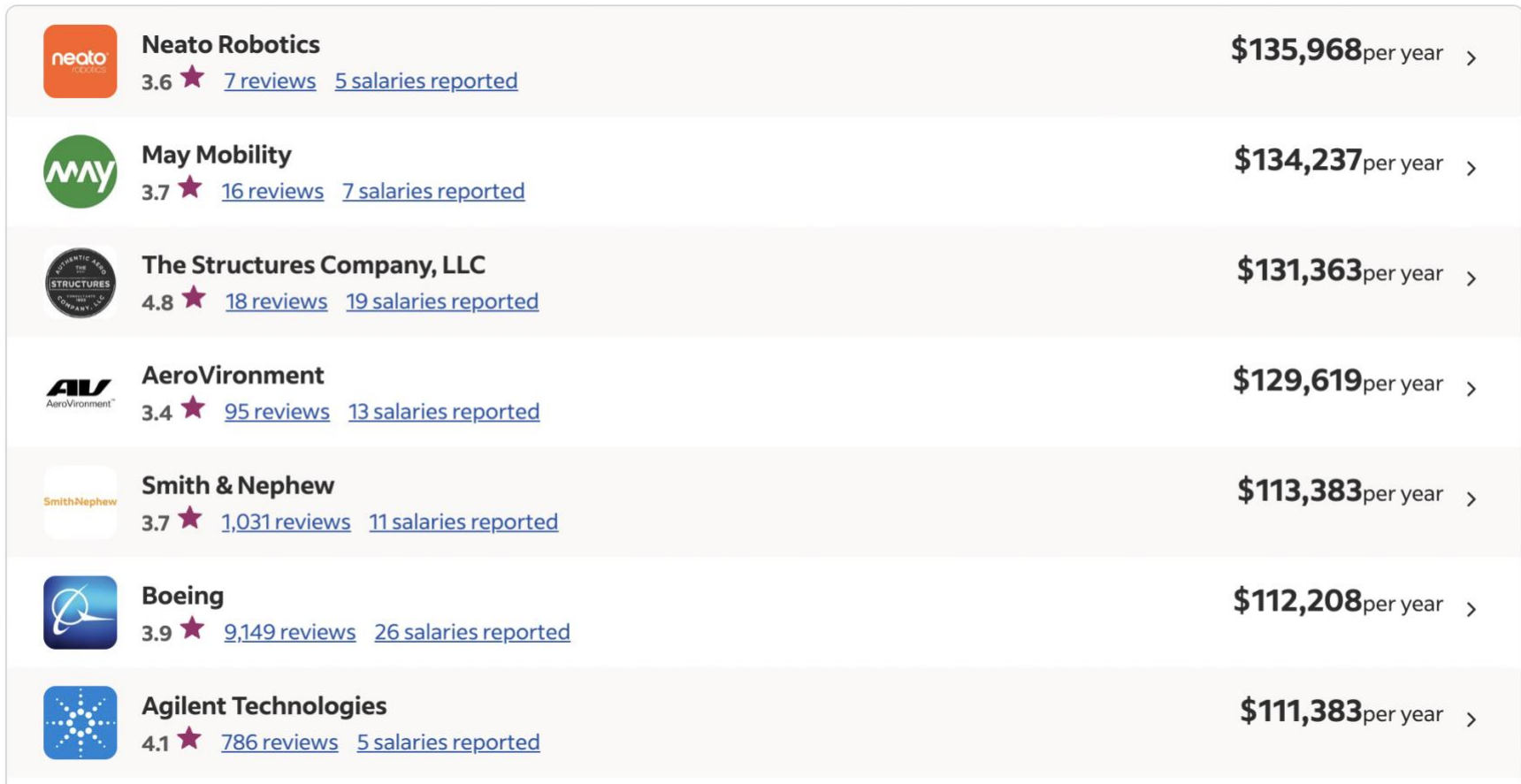
What is starting to happen in robotics already happened with data engineers and deep learning just a few years ago. Today, every company wants a machine learning engineer, hence, you can find machine learning engineers everywhere. The machine learning space is now too crowded.

However, that is not the case for robotics programming. There are very few robotics developers in the world. Now is the best time to jump onto the robotics wagon and be one of the first in this field.

- **The demand for robotics engineers is very high**. You only have to look at the robotics-worldwide mailing list to see how many job offers relating to robotics programmers are posted daily. The offers include various jobs: you can work for companies, startups, and research institutes, or even do an internship, PhD, or Post-Doc.

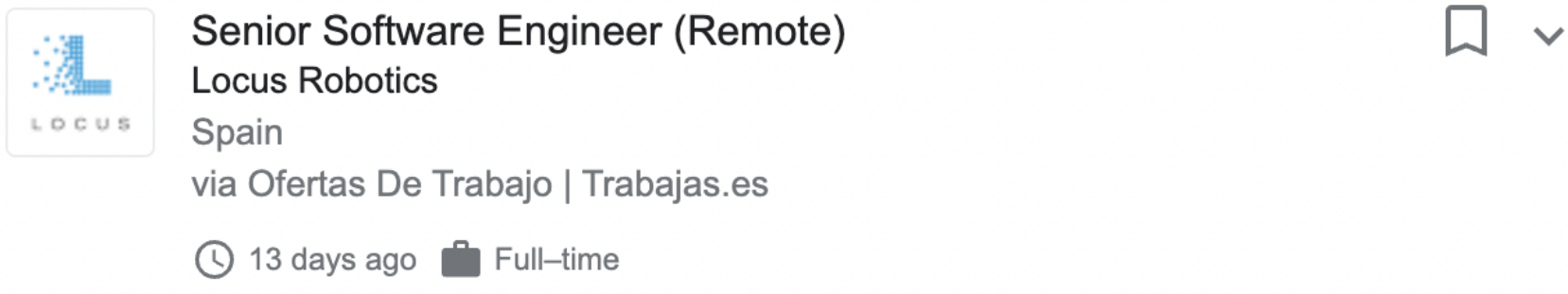
Check the weekly **list of robotics jobs** on The Construct's jobs page divided by continents.

- **Salaries**. Salaries for robot programming are quite good. You can check the average salaries for different countries on this website. See below an example for the United States:



Salaries for robotics software developer jobs (as of July 2023)

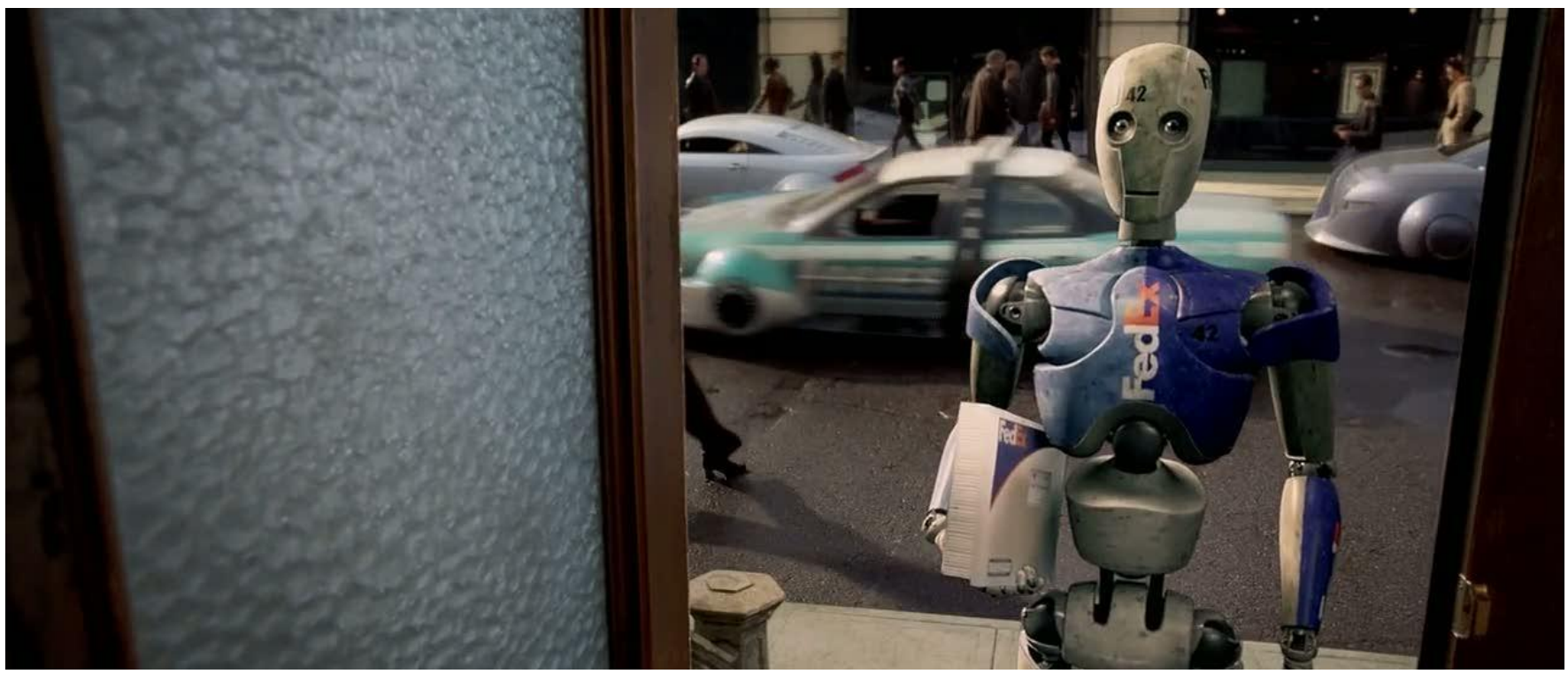
- **You may be able to work remotely.** If you work in software for robots, chances are that you can program the robots using simulations and DevOps tools without having to be on site. See more information below.



- **It is super cool!** Let's face it. A software developer for robots is a lot cooler than a software developer for accounting or for pizza apps that deliver to you at the beach. Society considers robots to be one of the coolest things in technology.

- **You can make a difference in robotics**. What is especially interesting about service robots is that the key to having a useful robot is not the hardware, but the software. The hardware that allows a robot to clean your house already exists. The most significant limiting factor for service robots is software. So, your job can contribute to finally bringing useful robots to real life.

- **You can make a difference in society**. All Hollywood movies depict robots as bad things that will hate and enslave us. If you are a robotics developer, you can change this narrative by creating robots that are useful, good, and willing to help us. **The robots of the future depend on you**.



Screenshot from *I Robot* movie, showing a case of how robots can be useful

The point is that robots will take over many jobs currently performed by humans. It is inevitable. And those robots must be programmed by someone. Would you like it to be you?

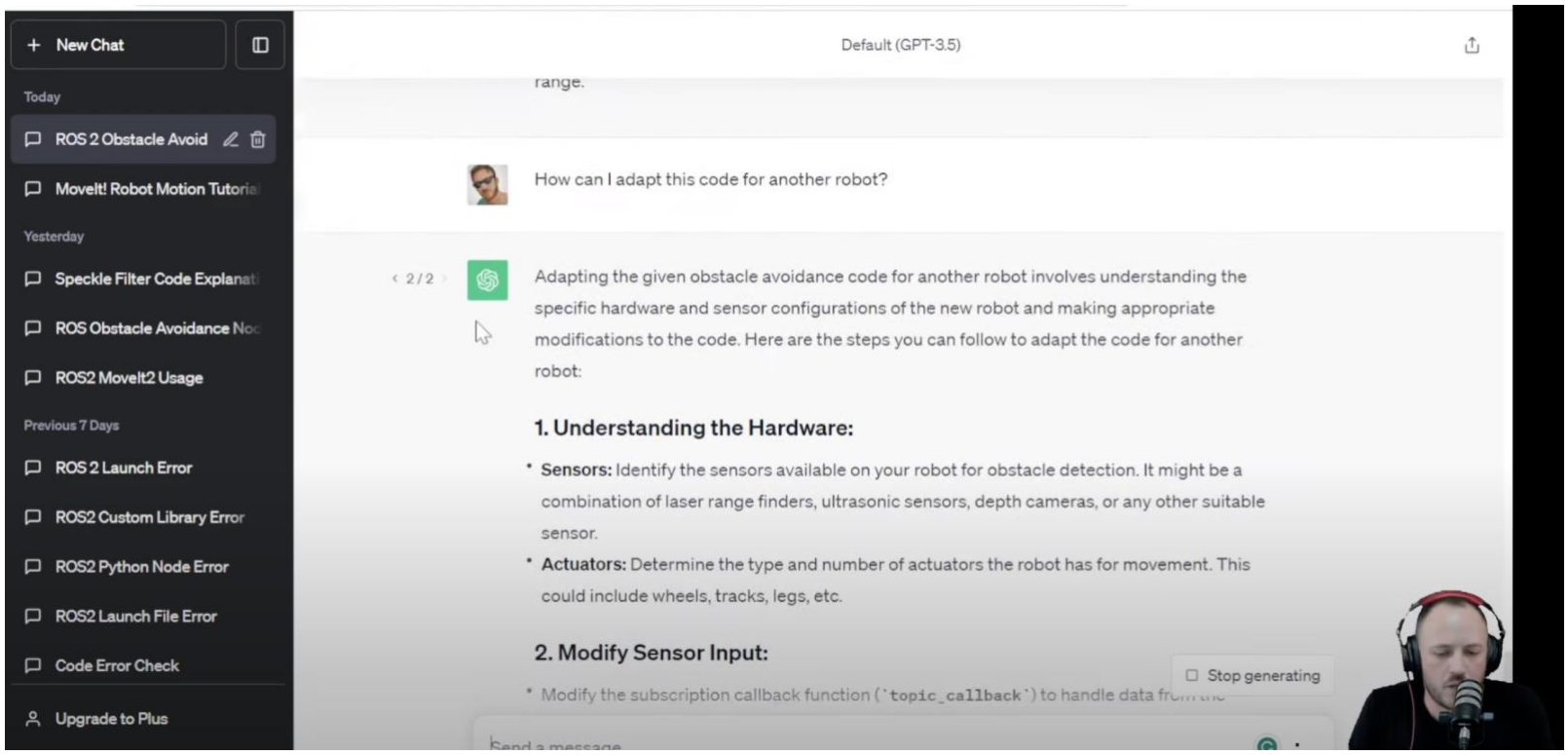
WHAT ABOUT CHATGPT TAKING THOSE JOBS?

There is a lot of hype about ChatGPT and the like taking programmer's jobs. Let me tell you clearly: this is not going to happen soon. Maybe 20 years from now, AIs will be able to program anything, but for now, they are far from it.

AIs can improve developers' throughput, make developers more efficient, and help a single developer create more than before.

However, Ais cannot create the programs required for robots to work and deliver tasks. That subject is highly complex, and no AI can do it (only a few humans in the world can -that is us, the robotics developers!). Not now, or in the next 20 years.

Yes, you will need to incorporate AI to do better work. Yes, you will have to continuously adapt over the next 20 years and increase your productivity. That is true. But your job will not be in jeopardy for the next couple of decades.



So now is the time to jump into software development for robotics.

WHY ARE THERE NOT ENOUGH DEVELOPERS FOR ROBOTICS?

Reason 1: lack of a learning path

One of the main reasons is **the lack of a comprehensive curriculum that provides the necessary skills for robot programming.**

Until very recently, **there was no need for robotics developers**. Robots could not do anything useful, so **robots were relegated to the labs**. However, as robots become more skilled, **they are leaving the lab** and going to homes, facilities, etc.

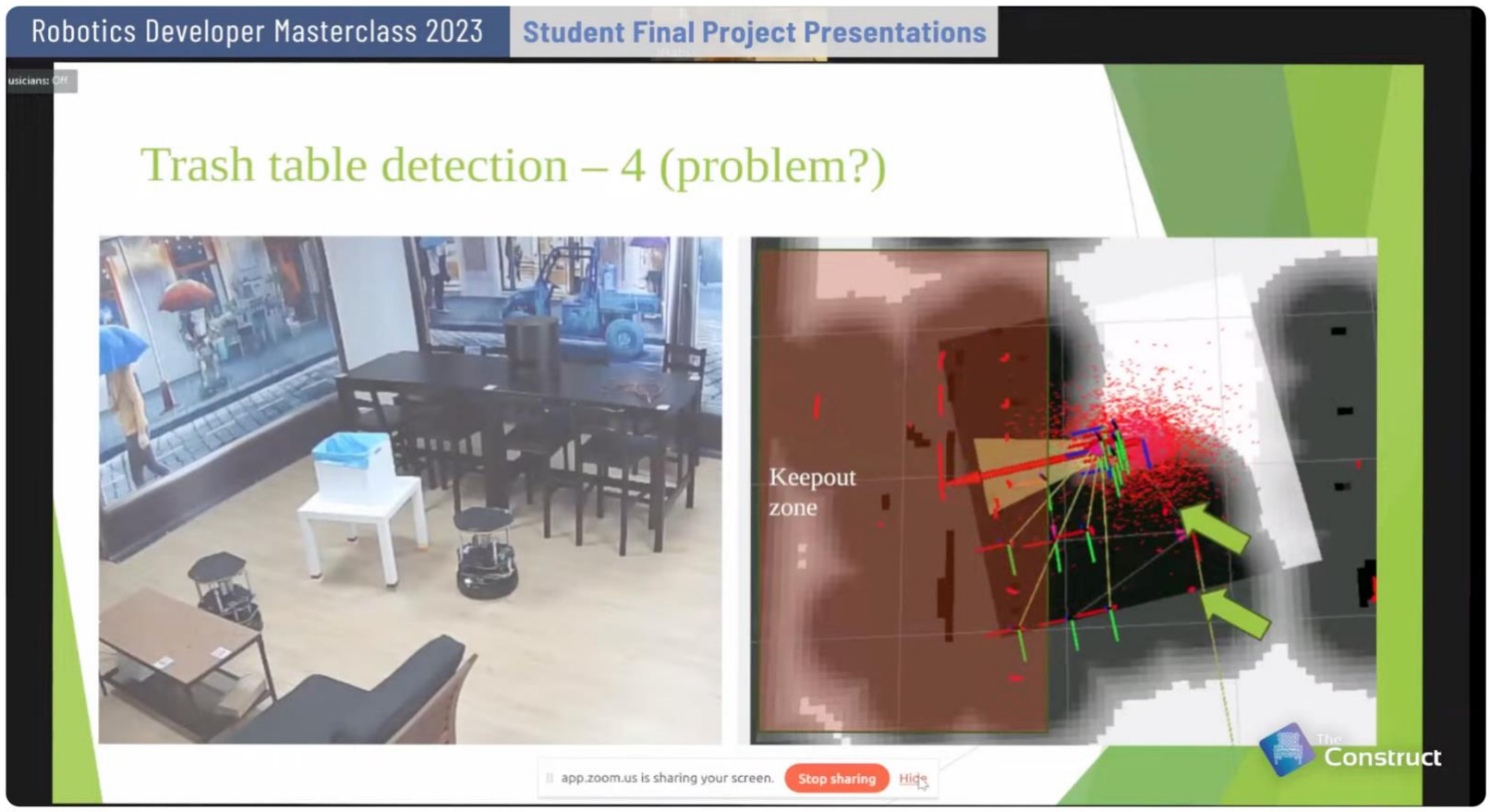
It is there that robotics developers become necessary, to build robust robotics products.

The problem is that **there is no clear path to becoming a robotics developer**, that is, a person who uses software skills to program robots.

Usually, in the past, robot programmers were roboticists who knew a lot about mechanics and electronics and just enough about computer programming. However, *just enough* is not valid anymore. If we want to build robotics products, we need pro programmers.
However, **software programmers cannot go directly to work as robotics developers**.

A robotics developer must be good at programming, know robotics, and know the basics to understand how robots work, the main structure of their components, how to get data from the sensors, how to send commands to the actuators, and a little more. They must also be able to

trust that the hardware will work as expected (in the same way that the computer works as expected). To master that, robotics developers may also need to study robotics theory.



Let me say it again:

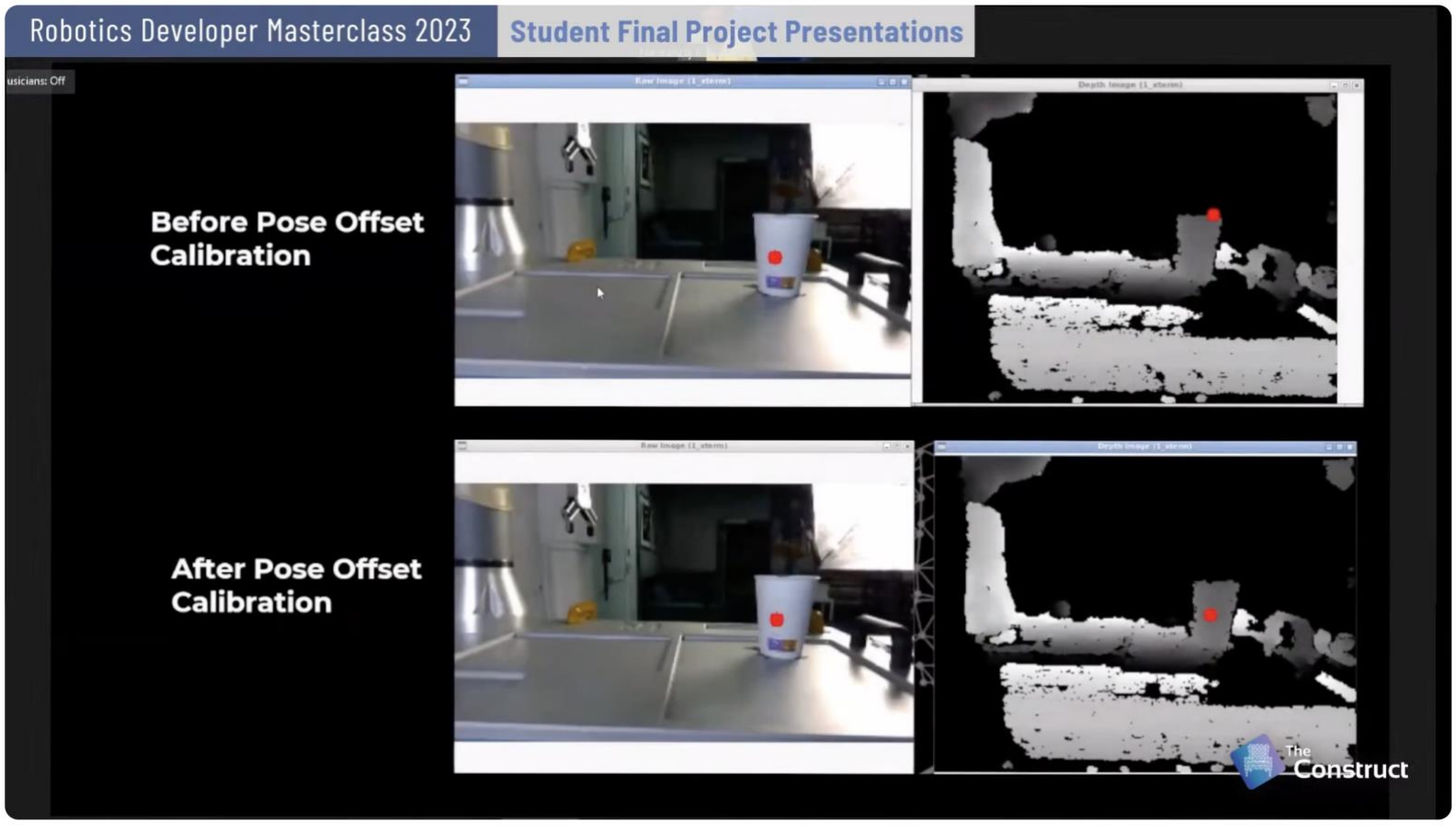
Robotics developers do not need to develop a new path-planning algorithm. They don't need to develop a new control paradigm. They don't need to invent a new object recognition system. That is the job of the roboticists and AI at the labs. Robotics developers need to know which algorithms exist, how to use them, when to use them, and how to integrate them into the complete robotics application. Hence, the global robot always does what it is expected to do.

Until now, there has been no clear path to becoming a robotics developer. However, I describe a full learning path in the next chapter of this guide.

Reason 2: software developers hate hardware

The second main reason is that, in general, **software developers do not like to deal with hardware**. You are likely a developer and have never thought about entering into the robotics realm. You probably think that by programming for robots, you need to know about **electronics** and maybe even **mechanics**. You probably think that hardware and software are so coupled in robots that you cannot touch one without touching the other. That interaction with the hardware is something that many software developers don't like. After all, they decided to become developers of software, not hardware!!

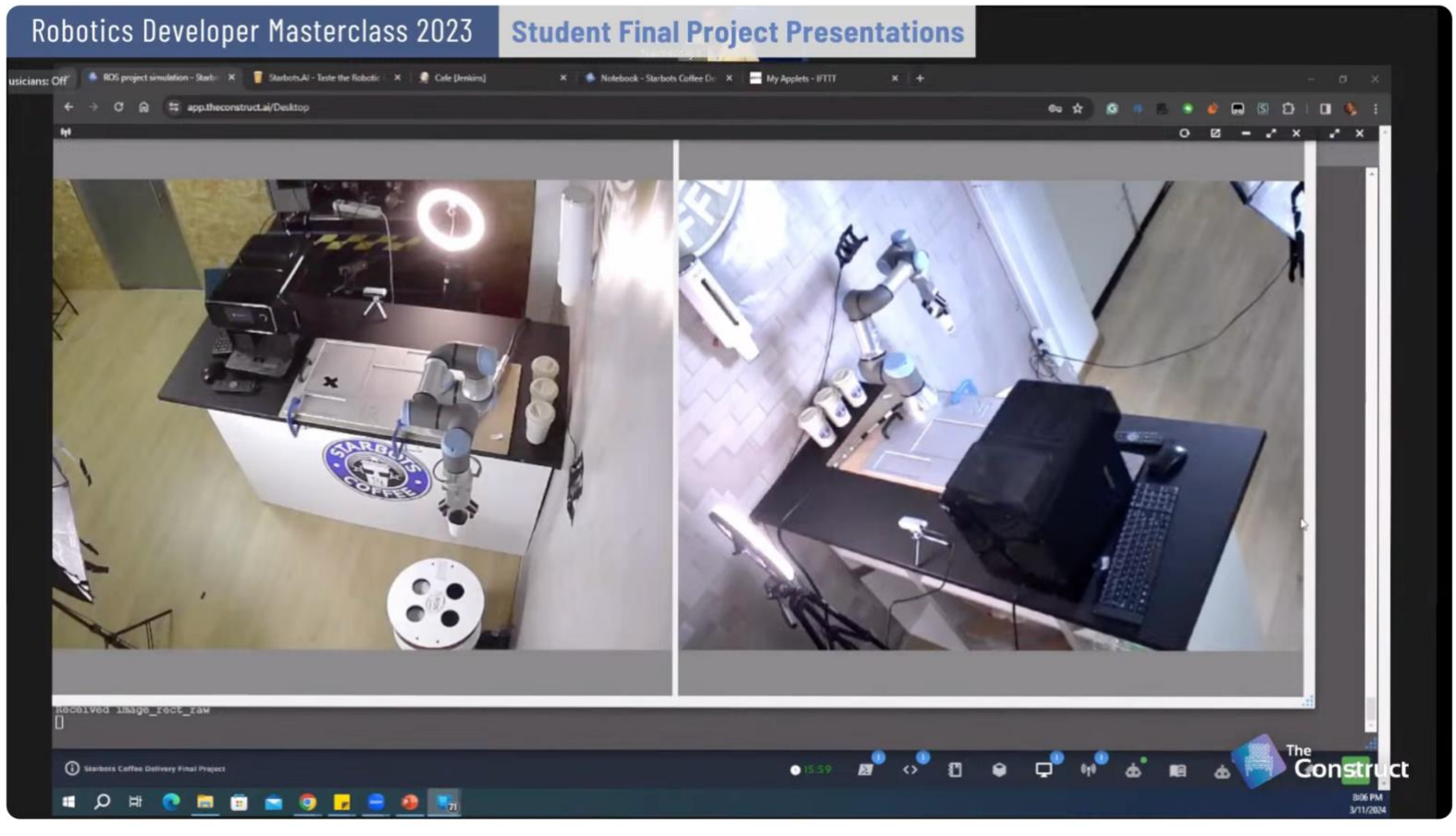
Fortunately, that hardware interaction can be removed at present for robotics developers.



ROBOTICISTS PROGRAMMING ROBOTS

Due to that lack of software developers, robot programming is done by roboticists, the people who build the robots. Perhaps some programmers are not directly involved in creating the robot, but they have no problem getting into the hardware and trying to fix it when something goes wrong. But let's face it. Most developers are better programmers than roboticists. That is why robotics could benefit from having many expert programmers enter the field.

The good news is that attracting developers to the field is easier than ever. **Thanks to the Robot Operating System (ROS), you can completely abstract the hardware from the software**, so you can program a robot by knowing **the robot's ROS API and testing it on a simulation**. Using the ROS API, you can forget about the hardware and concentrate on the software that makes the robot do what you want.



WHAT IS THE ROBOT ROS API?

The ROS API is the list of ROS topics, services, action servers and messages that a given robot is providing to give access to its hardware, that is, sensors and actuators. If you are not familiar with ROS, you may not understand what those terms mean. Simply put, in the developer's language, topics/services/messages are like the software functions you can call to get data from the sensors or make the robot take action. It also includes the parameters you can pass to those functions.

Most modern robot builders are providing off-the-shelf ROS APIs, like the ROS-Components shop (http://www.roscomponents.com), which provides all its robot hardware running with an ROS API.

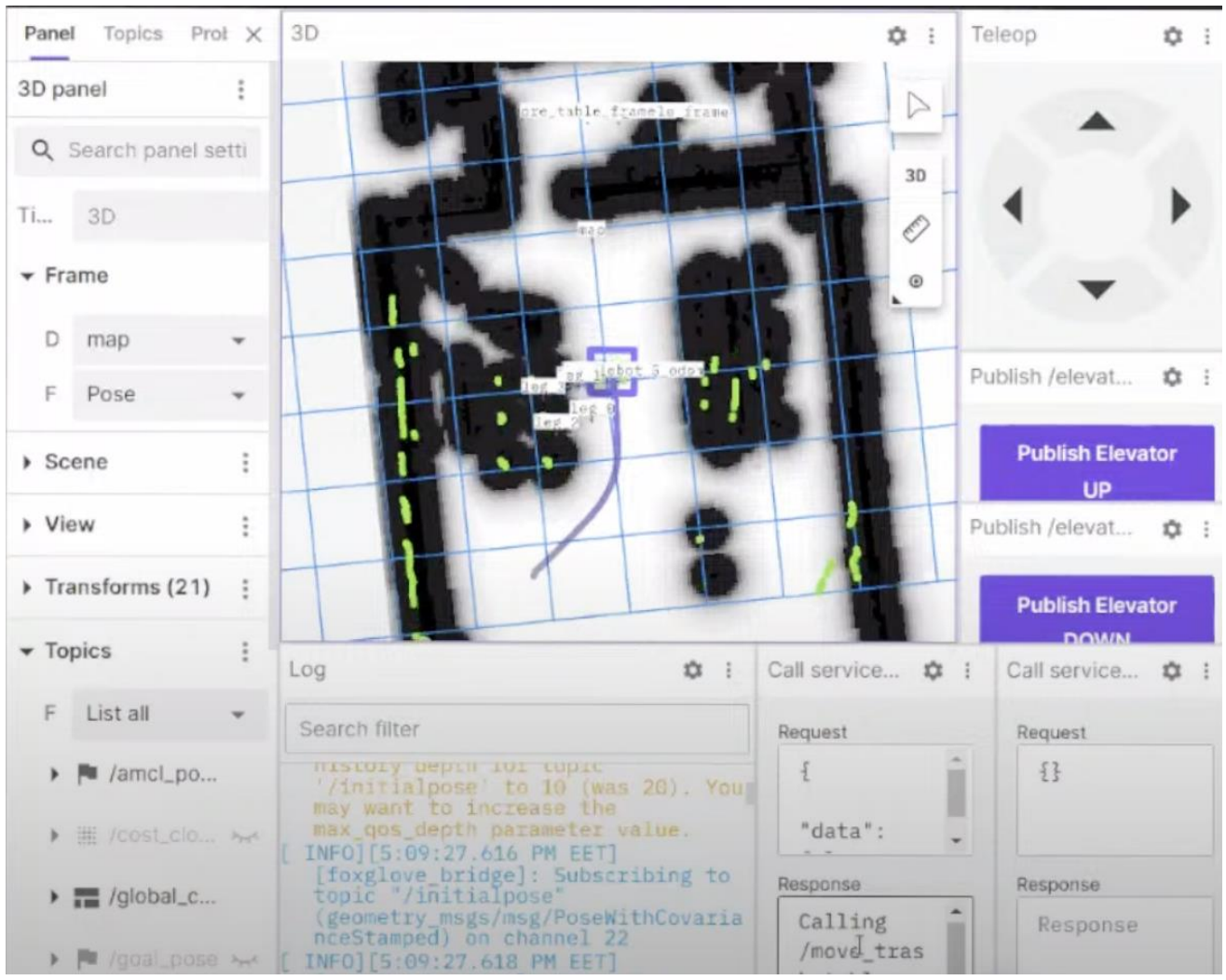
If the robot you want to work with does not run ROS you can make it work by ROSifying it. To ROSify means to adapt your robot to work with ROS. To ROSify a robot usually requires knowledge to access the hardware. You need to learn how to communicate with the electronics that provide the sensor data or access the motors of the robot. In this book, we are not dealing with that subject because it gets out of scope for developers.

So, for the rest of the book, we will assume that you have access (or are willing to have) to a robot that is already ROSified. That is actually the trend in robotics.

WHAT IS ROS ANYWAY?

ROS stands for Robot Operating System (http://www.ros.org). Even if it says so, ROS is not a real operating system since it goes on top of Linux Ubuntu (also on top of Mac and recently, on top of Windows). ROS is a framework on top of the OS that allows the abstraction of the hardware from the software. This means, you can think in terms of software for all the robot hardware. And that is good news for you because this implies that you can actually create programs for robots without having to deal with the hardware. Yeah!

ROS works on Linux Ubuntu or Linux Debian. Experimental support already exists for OSX and Gentoo, and a version for Windows in under way, but we don't recommend using them yet unless you are and expert. Check this page (https://tinyurl.com/y6z36nru) for more information about how to use ROS on those systems.



CHAPTER 2:

A LEARNING PATH TO BECOME ROBOTICS DEVELOPER

Maybe **you have finished an engineering degree but feel it is not applicable to real jobs.** So, you decide to **go online to find the practical content** required to become a Robotics Developer. Once you have the content, it is a matter of following it and acquiring the knowledge, right?

Well, yes and no. The problem with that approach is that **there is too much content online** about robotics and software development for robotics.

The amount of content material (free and paid) is so massive that unless somebody provides you with a proper roadmap, you will lose a lot of time in the learning process.

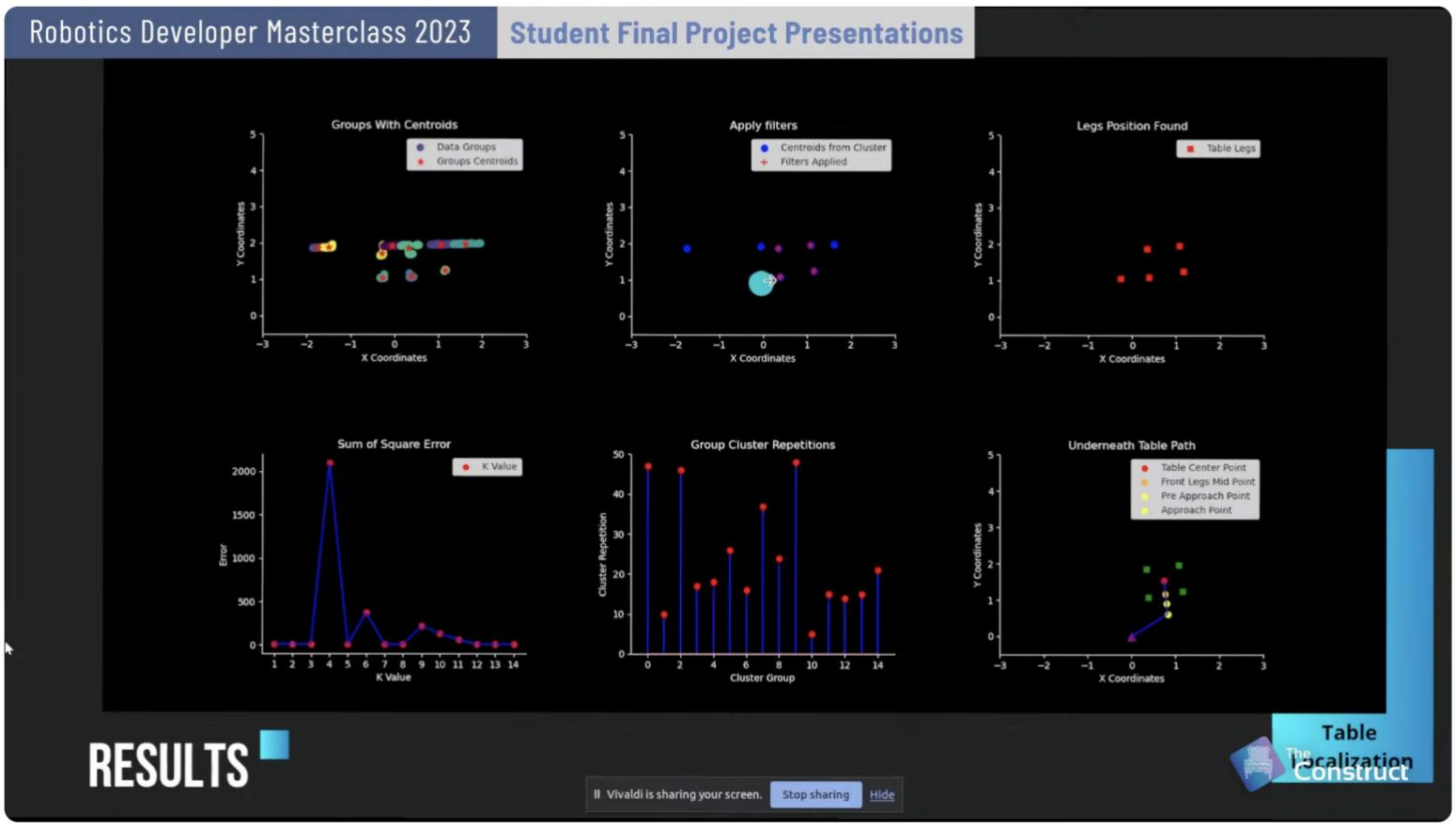
So, to effectively learn by yourself, you need to figure out the following:

- **Which content should you select?**
- **And how should you prioritize the learning content?**

In this chapter, I'll answer those questions. I'll show you what subjects you must master and in which order **to become a Robotics Developer**. I have assumed that you are starting with zero knowledge (i.e., basic high school math skills only).

This post describes the same structure that we follow in our Robotics Developer Masterclass. Our **6-month** learning plan guides you **from zero to Robotics Developer** and helps you land a job by means of our **included internships at a robotics company**. So, I can assure you that the learning plan works!

0. PRE-REQUISITES



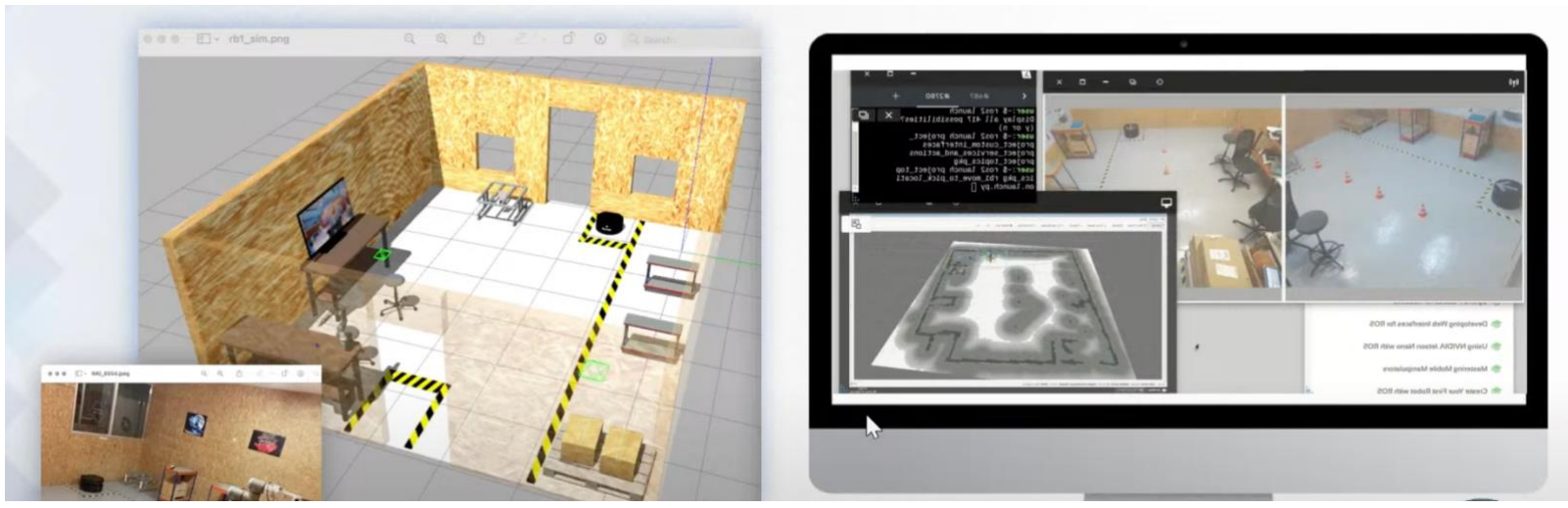
First, you require a foundation for the systems used to learn more complicated subjects. The minimum basic knowledge required is the following:

- **Linux**. Linux is the base of most professional robotics systems. For that reason, Linux knowledge is a must. Take this free course to learn Linux essentials for robotics.
- **Python 3**. Python allows you to create quick prototypes and test your ideas before going into a production solution. Take this free course to learn Python essentials for robotics.
- **Basic math**. A minimum high school level math equivalent is a requirement to understand basic robotics concepts. Take this free course to learn essential math concepts for robotics.

Good books about those subjects:

- Python Basics: A Practical Introduction to Python 3 (English Edition)
- Linux for Beginners: An Introduction to the Linux Operating System and Command Line
- Basic Maths: High School Math Made Simple

1.FOUNDATIONS FOR ROBOT PROGRAMMING

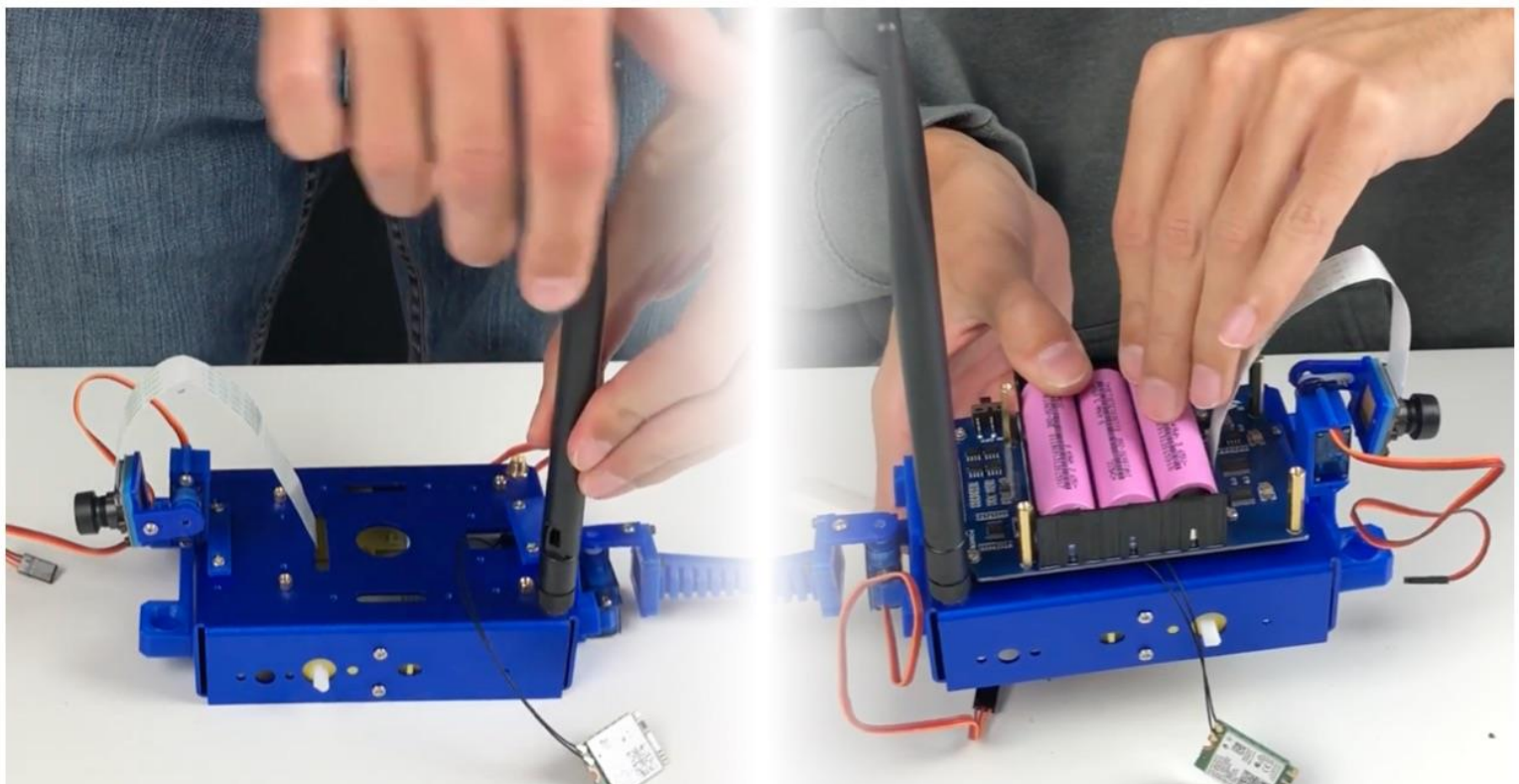


In this section, you will build your foundation skills for robot programming. This includes the following topics:

- **C++, in-depth**. Most robotics jobs require you to have an excellent knowledge of C++, so you need to start learning it as soon as possible and gain experience. The critical point for learning C++ is to dedicate time to practice on real projects. That is why the sooner you start, the better.
- **Git version control**. Git is a tool that allows developers to work collaboratively. Understand that when working in a company, there is no concept of “my code”. Your code will have to work with another engineer’s code. Git is the tool used in every robotics project to facilitate this collaboration. Master this tool and develop your robotics projects using Git, even if you study alone.
- **Basics of ROS2**. ROS is the standard in robotics. Learn and use this framework in the rest of your learning. ROS has the same problem as C++: you can learn the theory quickly, but real learning begins when you practice with different robots and situations. So, base your learning on this tool and practice as much as possible during your learning phase.
  - Learning the basics of ROS includes learning all the basic concepts of **packages**, **messages**, **topics**, **services**, and **actions**, as well as **robot modeling using URDF** and the concept of **transformations (TF)**. However, that is the bare minimum you need to know in ROS.
  - Which ROS version should you learn? Now, you need to know ROS2.
- **Robot simulations**. Simulations are an essential tool in robot development. After all, if you don’t want to deal with the hardware, you will need a place to test your code. That is in the simulation. So go and learn how to do simulations with the **Gazebo simulator.** That is the default simulator in the ROS world and integrates well.

2. BUILD YOUR OWN ROBOT

Even if you are looking for a developer's job (this is programming robots, not building them), you still require a basic knowledge of the parts and pieces of a robot and how they interact together to build the final robotic system.



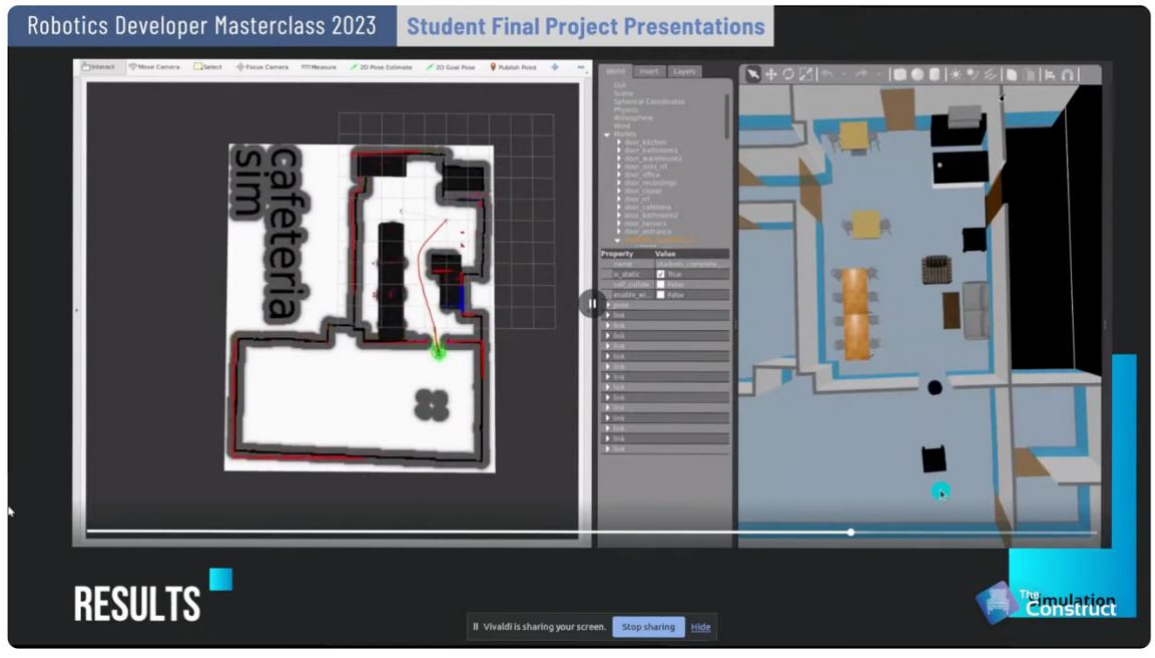
You should build your own robot, even if a basic one, to see how the various parts interact physically. This process will give you a grounded knowledge of physical elements that you can later apply to your programming, an understanding of why some software components are required, why they do not work, and isolate problems when something doesn't work.

Aim to build a **wheeled robot with at least two wheels, encoders, batteries, and one sensor (a camera is the cheapest version, but if you can afford it, include a LIDAR).**

Once the robot is built, you will need to add ROS controllers to it. But since your ROS knowledge at this time will be low, I recommend you start building a robot that already has the ROS controllers developed by somebody else so you can practice cloning the repo, compiling, and installing those controllers.
I do not recommend trying to create the controllers for that robot at this stage because it will take a considerable amount of time. Of course, you will learn a lot, but you can speed up the process by reading another person's and understanding it. After all, you will have to do that process (cloning repos, understanding other people's code, installing…) often once you land a job!

As for the robot, I recommend the Jetbot or the Nanosaur, both based on NVIDIA boards. I recommend those because they provide all the pieces you need, as well as the ROS controllers.

3. ADVANCED ROBOT CONTROL PROGRAMMING



Up to this point, you will have mastered the basic building blocks required for a robotics developer. Now, go to the next level, to **apply that knowledge to a robot, making it accomplish tasks**. You must master a robot's different skills and apply them to task resolution.

- **Robot navigation**, or how to make the robot autonomously move from one location to another
- **Robot perception**, or how to make the robot perceive its environment and detect the relevant parts for its task
- **Robot manipulation**, or how to make the robot act upon the objects of the environment by grasping, pushing, and releasing them, etc
- How to add **control** to robot motors

4. ROBOTICS THEORY BASICS

Now it is time to learn the theory. Even if you are going to work in robotics development and use many solutions already made for you, a minimum of robotics theory is necessary to understand what happens when something goes wrong. Also, knowing the theory will allow you to build better algorithms for robotics.

This section should include the following:

- **Mobile kinematics**
- **Arm kinematics**
- **Kalman filters**
- **Robot dynamics**
- **Robot control**
- **Path planner algorithms**

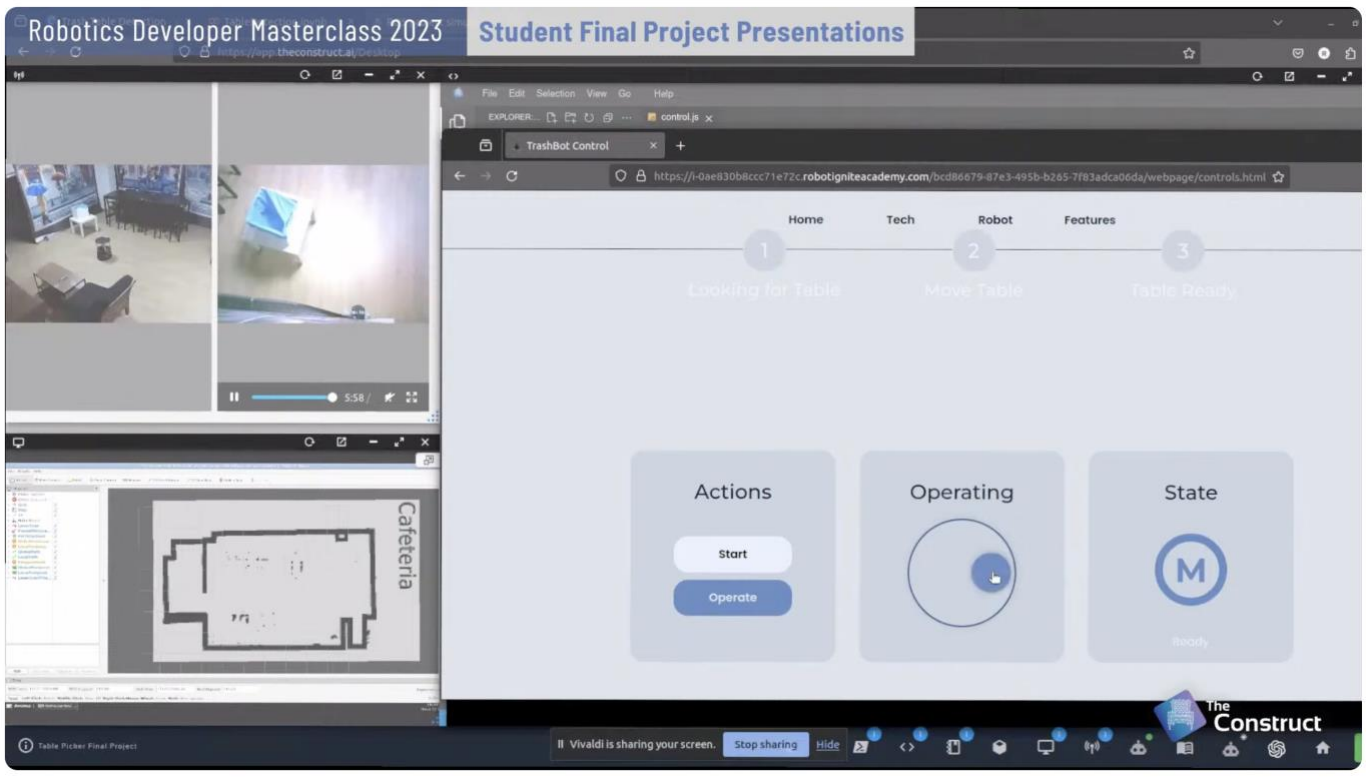
I recommend that you search for material that teaches you the algorithms while making you implement them in robots. I have found that this is the most efficient way to learn all that theory, make it stick in your head, and find the relation with what you use in your robot implementations.

5. DEVELOPMENT TOOLS FOR ROBOTICS

So far, you have learned a lot about programming robots and making them do things. However, when you apply that knowledge to a job in a robotics company, you also need to understand the tools required for software development in a professional environment. Again, you will not be alone, doing your own code. You will be part of a team that develops the full robot software together. Then, you need to learn to use the tools used for proper software development.

Among those tools include:

- **Docker**
- **Jenkins or a similar tool to automate processes**
- **Unit tests**
- **Functional tests**
- **Continuous integration**



The goal is to understand how all the development tools integrate to create a complete pipeline of deployment, testing, and error detection. This is **continuous integration**. So, you need to learn how to perform continuous integration, integrate Git pushes with Jenkins actions, deploy dockers with the code to test, launch the tests and associated simulations, and generate reports based on the success of the tests.

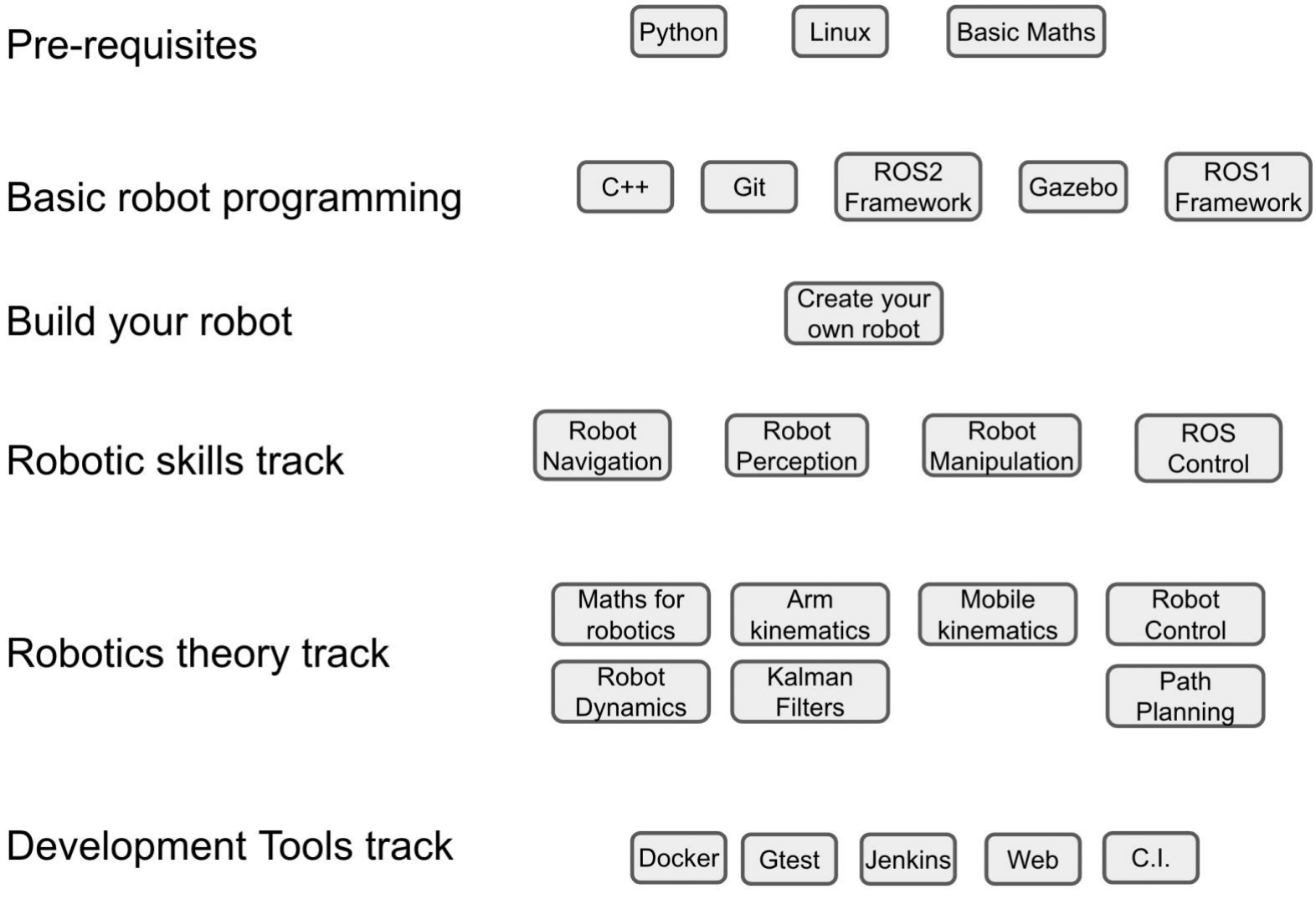
6. EXTRA SUBJECTS

As additional optional subjects, I recommend studying the following:

- **ROS1**. Many companies still work on ROS1, so you need to know how it works. Even if very similar to ROS2, you need to understand some differences in creating software for ROS1 robots.
- **Web development** to create interfaces for robots. The robots of the future will be from many different manufacturers, and the web interface is the only universal interface between the robots and human devices. So, robot interfaces will not be based on apps but on the web.

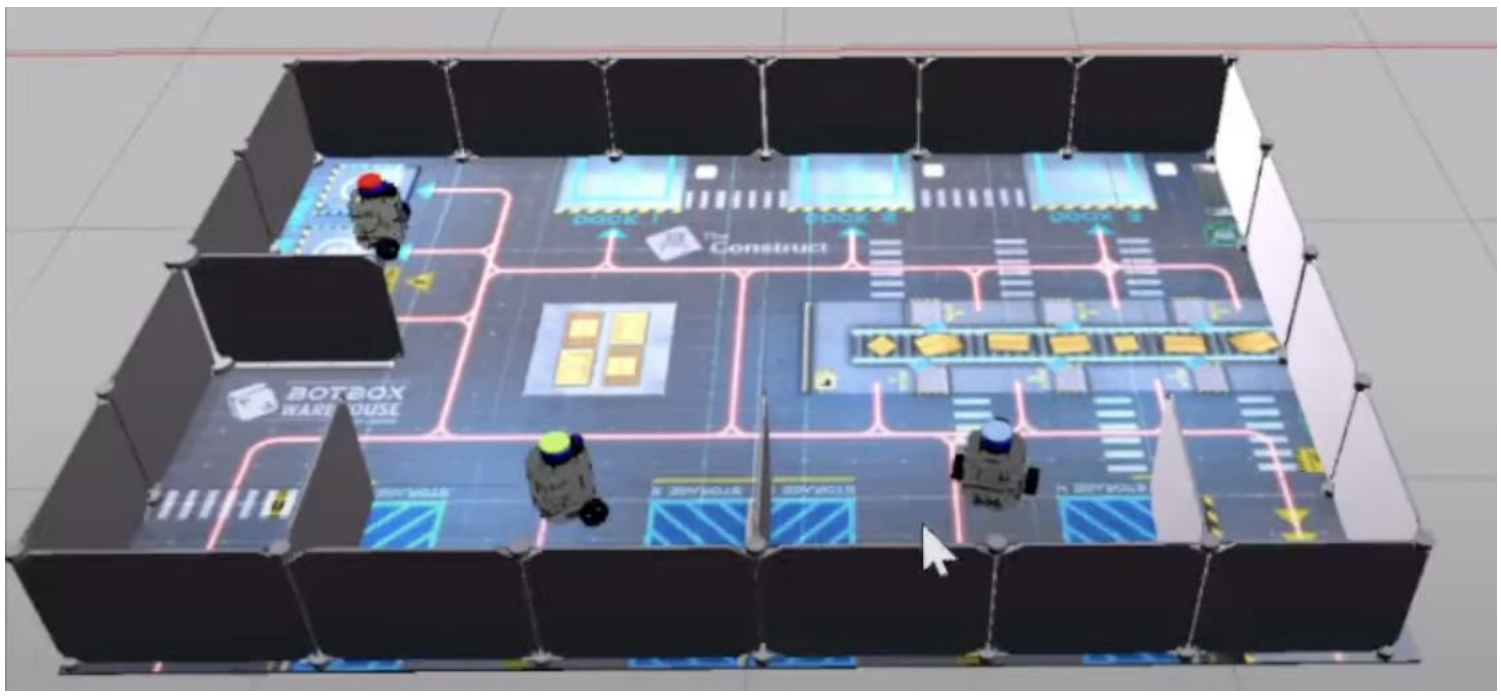
Learn how to create web interfaces that display the robot's status and allow you to control it remotely.

FULL SCHEMA OF SUBJECTS TO LEARN



TIP: PRACTICE WHILE STUDYING

My final recommendation is that you do not just watch videos or read books to learn all this material. If you want to gain knowledge, you need to practice the material along every step of your study. That is the only way you can master the subjects.



For that, I recommend you find sources that provide you with robot simulations or at least Git repos with code that you can download and to test your solutions. You can check The Construct's public repo of free robot simulations and download some simulations to practice with them. You can also check The Construct Open Classes, delivered online each Tuesday, which provide a lot of code and simulations already working on any computer without requiring any installation, and everything for free.

START NOW!

So now you can search the internet for the best resources to provide you with the materials above. **This should be your plan**:

1. Dedicate a few days to select the best materials by reviewing them.
2. Make a list of the materials you selected for each subject in a document with links to the videos/posts/courses. If you are relying on books, order them now.
3. Then, make a calendar detailing when you will finish each subject. This is mandatory, especially if you are following the program by yourself. The calendar will act as accountability, which is the most challenging part when studying alone.
4. Then, concentrate on those materials until you finish. Do not redo your list, or you will never end. The number of resources out there is too huge! You should concentrate on learning rather on compiling more material (that is a way of procrastination and signals that you lack willpower).

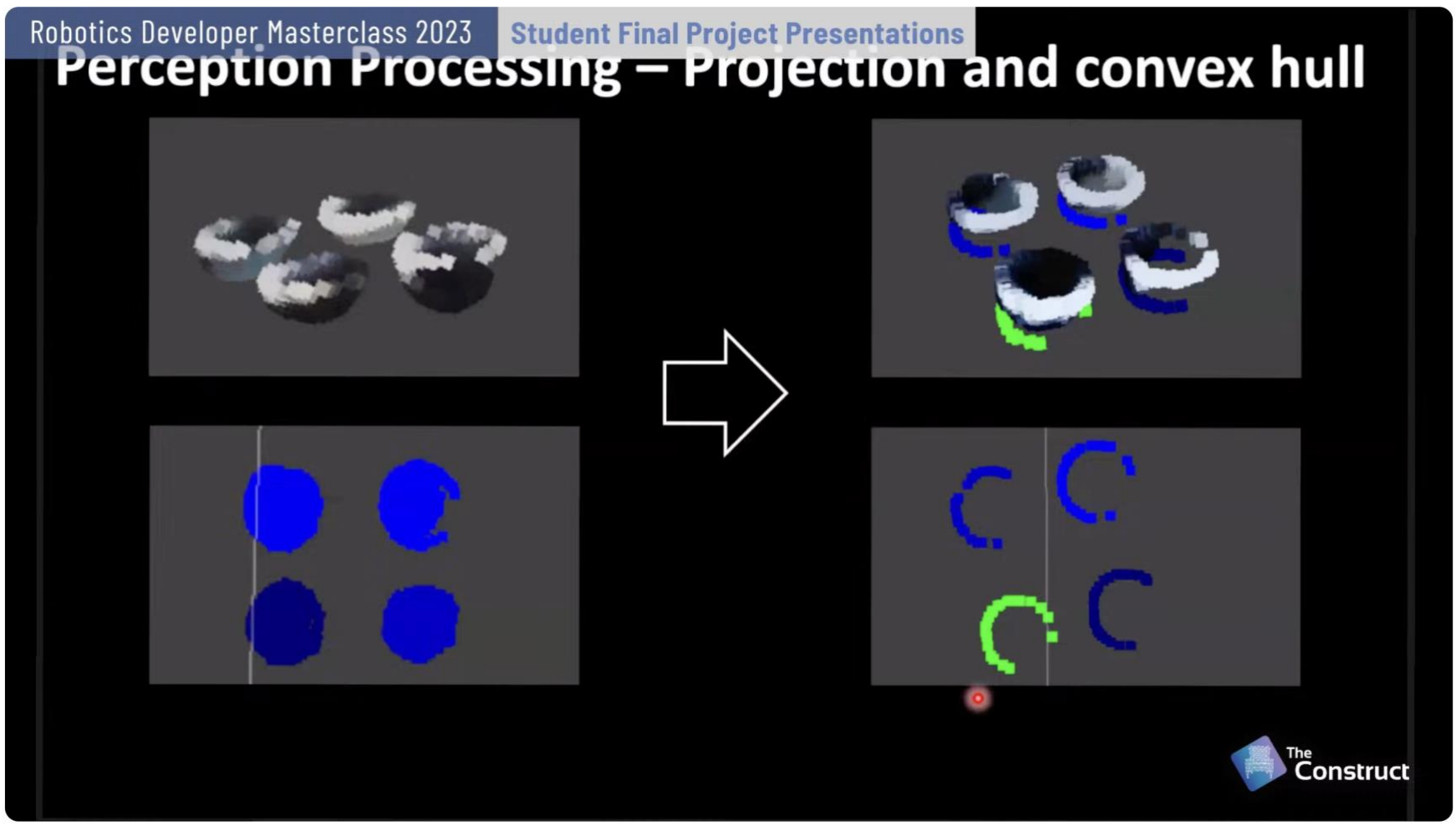
IF WILLPOWER IS YOUR WEAK POINT...

Consider that the weakest point in your path is your willpower. The main drawback of the self-learning approach is that it does not have the *optimal path* or find the *optimal materials*. It is having the will to follow your plan until the end. This is where most self-guided learning fails at a +80% rate.

Check out our Robotics Developer Masterclass if you want to avoid that pitfall and wish to have a team that supports you, motivates you, and pushes you forward. This program provides you with a path and all the **resources**, a **group of tutors supporting your learning**, **simulated and real robots**, **projects**, and a final **internship** once you graduate.

Our Robotics Developer Masterclass is **project-based learning with employability in mind**. Of course, we teach you robotics theory, too, but we emphasize the tools that you will use when working for a robotics company as a robotics developer.

And we will be encouraging you throughout the program! Check it out! **Batches start in March and September each year**. Check the Q&A session about the Robotics Developer Masterclass we deliver on Youtube



CHAPTER 3:
GET A JOB!

Once you have the skills you are ready to start looking for a job as a Robotics Developer.

TYPES OF JOBS YOU CAN HAVE AS A ROBOTICS DEVELOPER

As a robotics developer you can have five types of jobs:

1. Work as a hired robotics developer for a robotics company. That is by far the most liked type of job.
2. Create your own startup, producing your own robotics product.
3. Create a consultancy service where you help other companies develop their robotics products and charge them for your time and expertise.
4. Do research and teaching at the university about robotics.
5. Create educational material for people who want to become a roboticist.

STEPS TO GET A JOB IN ROBOTICS DEVELOPMENT

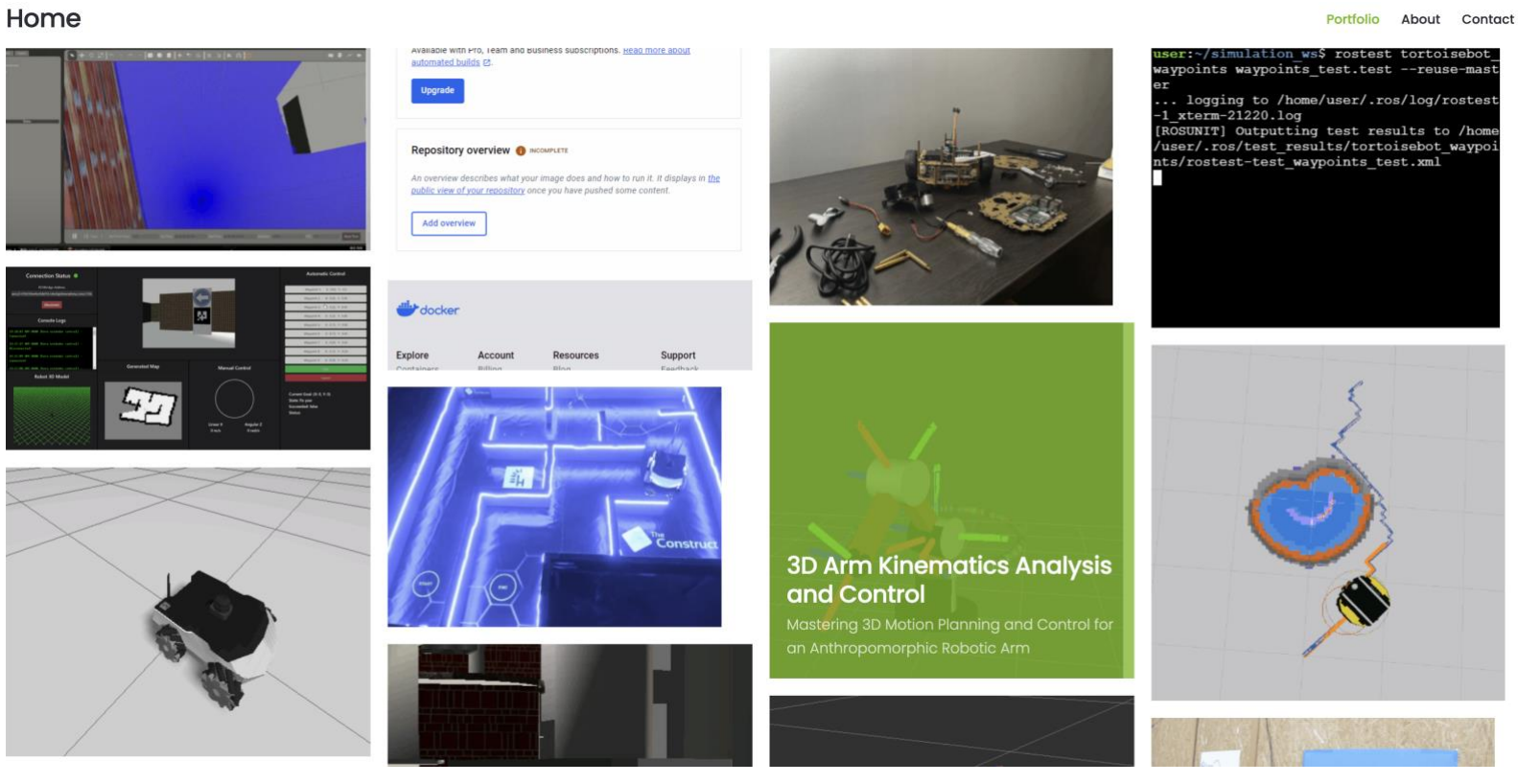
To get a job, we recommend the following steps:

1. Create a portfolio with your best projects
2. Apply for an internship at a robotics company
3. Apply for a robotics developer job

CREATE A PORTFOLIO

The portfolio is a web page that contains a summary of your best projects in robotics development, with details. The goal is to show it to your future employers during the selection process so they can evaluate you based on your work, not your degrees.

One example of a good portfolio is that of The Construct's student Miguel Solís.



This portfolio includes a visual grid of the most important projects that Miguel has done.

If you don't have a set of projects, that is where to start.

How to start projects

1. Do what we indicated above: study while applying it to projects. Then you have the set of projects already done!

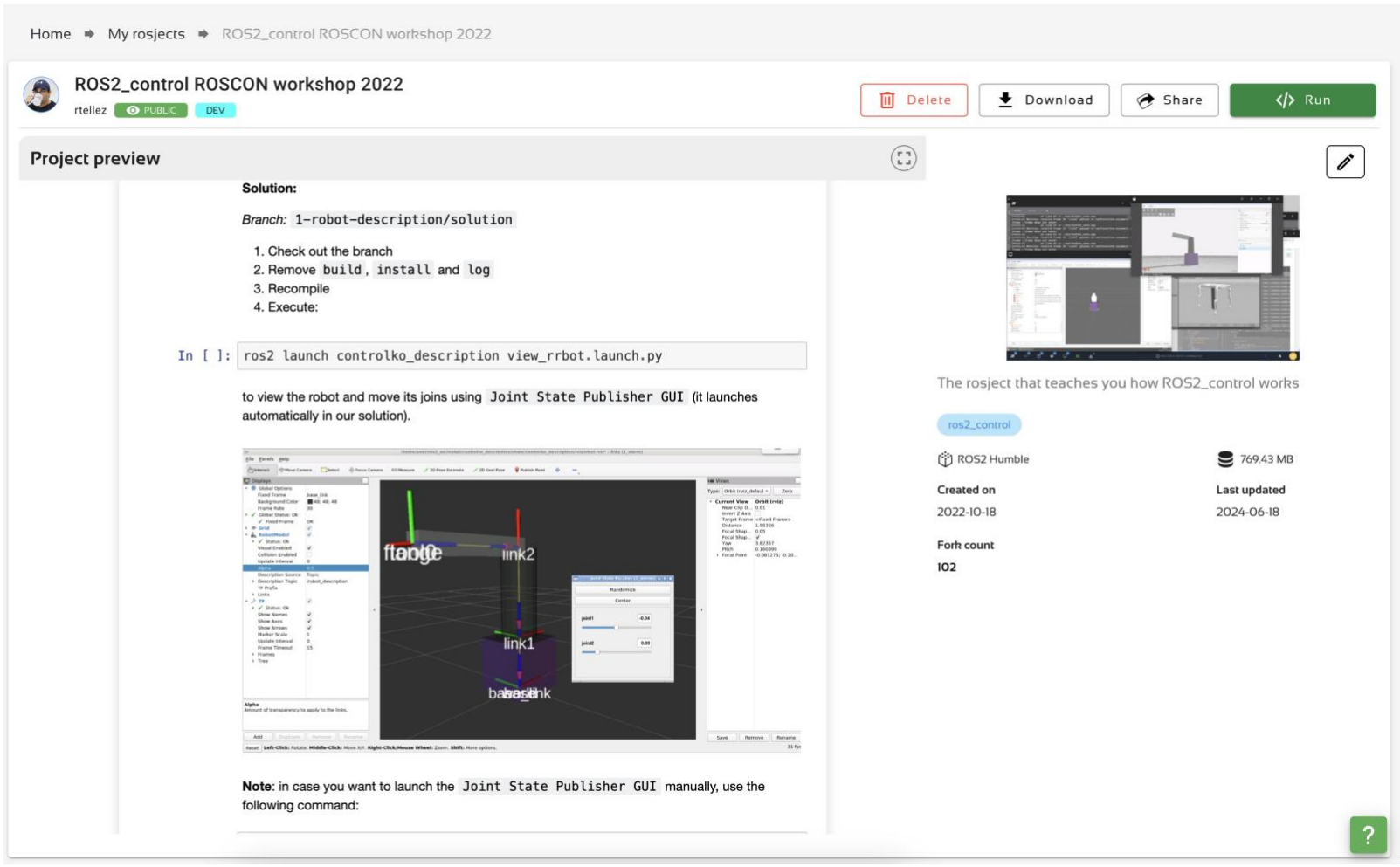
2. If you have zero projects created:

   - Pick a robotics problem that you would like to solve and find the solution. Repeat with a few more projects.

   - Search over the internet for an existing open source project that interests you and contribute by solving a bug, adding a feature, writing documentation, providing examples of usage, and especially, participating in the regular project meetings. Fight to push your proposal to the official project repo so you can include those commits in your portfolio. This is valuable as a project because you will show an employer that you understand the development process (on top of that, you will be helping open source to advance).

Remember to make your projects open source, that is, in a public repo, so your employer can review them easily during your application. You can put your code on a git (like Github, Bitbucket, Gitlab, …) or inside a rosject at The Construct.

I recommend the second option because [rosjects]() allow the employer to launch your project and see the results without installing anything on his computer. Also, the employer can launch your rosject from any computer, and it is 100% guaranteed that the project will work for your employer in the same way it works for you. That is a quick and safe way to show case your projects.



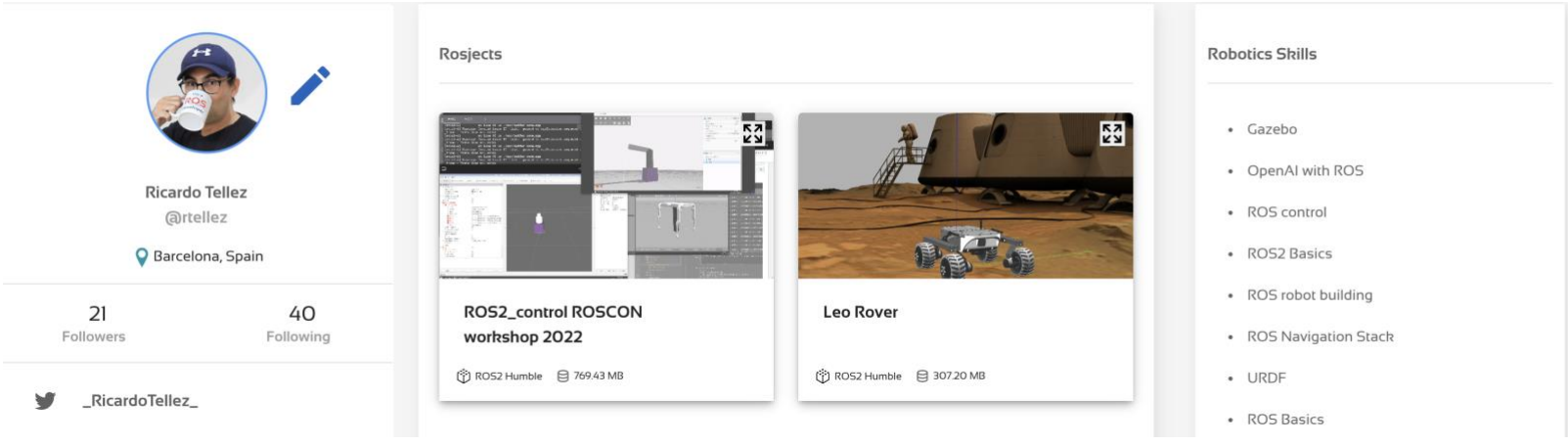
Create the portfolio

Once you have a set of projects (a minimum of 3), it is time to showcase them. The portfolio is how you list all your projects in a single place. To do this, you need to create a web page listing all the projects you have created. For each project, include the following:

1. A cool picture of GIF of what the project does
2. A cool title for the project
3. A description page where you include:
    - An explanation of what the project is about
    - How did you achieve the project
    - A link to the source code
    - (Optional) A video of the final result

Do not showcase all your projects, just the most important or impressive ones. The evaluator will not have time to see all your projects, so make it easy for him to understand your skills.

If you created your project as a rosject, then your portfolio page, description page, and picture are already ready inside The Construct. You only need to provide your employer with the link to your profile in The Construct, and the portfolio will appear.



APPLY FOR AN INTERNSHIP

The easiest way to get a robotics job is to apply for an internship at a robotics company. Usually, the barrier of entry for those positions is a lot lower. So, you have more chances to get into the company and then, once there, apply for a job.

Actually, the most likely thing is that the company offers you a job when your internship is about to end. The reason is that robotics companies are always short on robotics developers who know their craft. Also, one of the problems in companies is hiring somebody who looks very good during the selection process, but after a few months of work, you can see that it will not fit in the company. That is terrible for a robotics company where resources have to be contained.

If you get an internship and deliver an awesome job, you have a high chance that the company will want to hire you because they know how you work and deliver. Also, they know how difficult it is to get committed people for the team. So, if you show commitment to the internship, you will get a job offer very soon.

The key is that you deliver an awesome job during the internship. And what does that mean? It is not just that you know your craft, but also the attitude towards going the extra mile. You need to show that you are willing to go this extra mile and surpass what is expected. You need to show that you care about the result.

Okay, final remark about this: if you don't feel like delivering an awesome job once you have been hired, then don't do an awesome job during the internship either just to get hired. That is even worse because you are actually cheating (making the company believe that your level of commitment is this high when the one you will actually have after signing the contract is that low).

Final, final remark: if you showed commitment and in the end they did not offer you a job, do not worry about it. This could be due to many reasons that are not related to you. Maybe they have no open position right now. Maybe the person in charge is not a committed person who can see your value. Maybe they are worried about a project and did not pay much attention to your work…. Do not despair. In that case, once the internship finishes, thank the team, request a recommendation letter, and repeat the process.

APPLY FOR A JOB

Once you have finished the internship and you did not get a job proposal, you have two options: either you repeat with another internship or go straight to get a job. The criteria for deciding which one to do would be based on the feedback you got from the internship and your feelings about whether you think it is time to move to a real job. That is a personal decision.

Whenever you decide it is time, start applying for jobs. You can find a lot of job postings on Internet. I recommend you the three following resources:

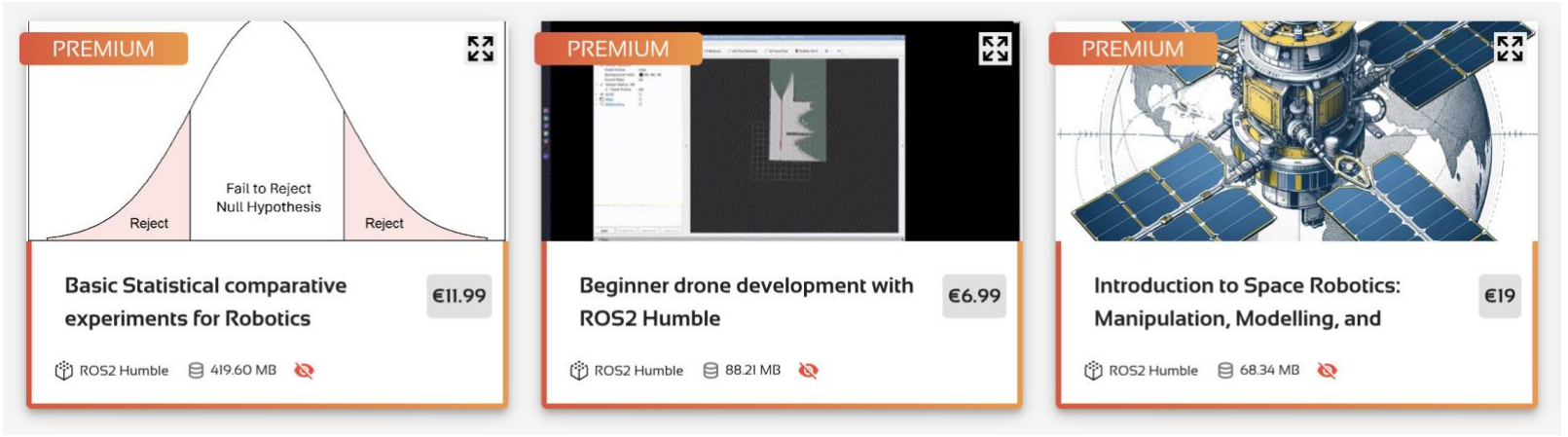
1. Search on LinkedIn for robotics jobs. You will get a lot of them. Filter at will.
2. Subscribe to the robotics worldwide mailing list and start receiving job positions daily.
3. Check The Construct weekly update list of robotics jobs, divided by countries.

In any case, the portfolio is still the most important thing when looking for a job. So, keep polishing it and creating more projects while searching for a job.

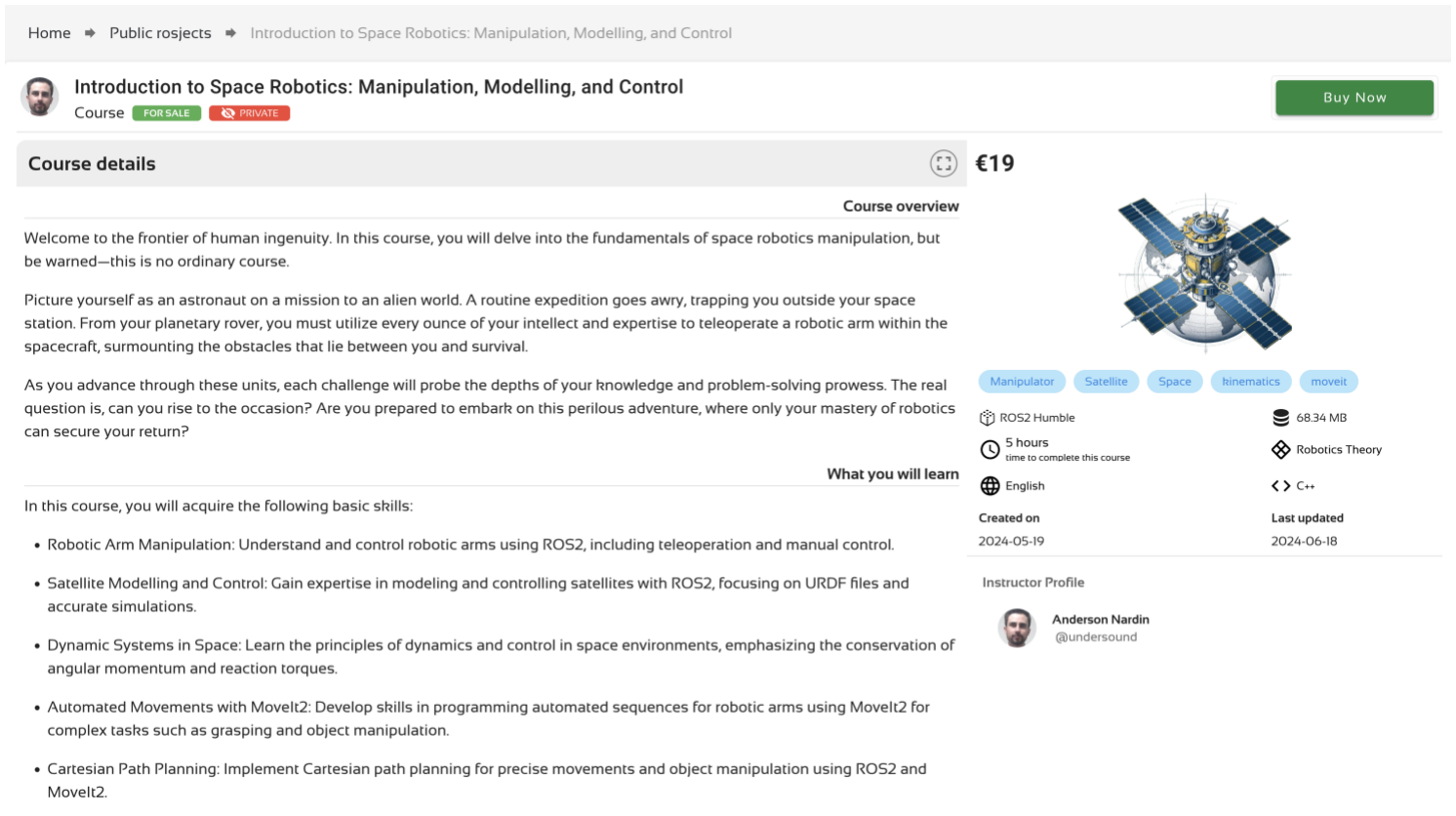
Unfortunately, if you apply to robotics developer positions, they will likely ask you for a test based on C++. The level required is extremely high. So, you will need to have a special training in C++.

I recommend you take the training for developers who want to work for Google.

IF YOU WANT TO CREATE EDUCATIONAL MATERIAL FOR ROBOTICS DEVELOPERS



There are many ways to create educational material to learn robotics. At The Construct, we provide a unique and professional way to create courses that are automatically distributed across our database of students. You can create courses based on simulations in an easy graphical way, apply for publishing, and distribute them quickly. Check our page of Public Rosjects for further instructions.



ABOUT OFFICIAL UNIVERSITY TITLES

Unfortunately, a university degree is still required for many robotics jobs. Not for all, but many still require you to have such a credential.

The reason is that having a title is a good filter for companies to ensure that a minimum knowledge is there.

Fortunately, this is becoming increasingly obsolete because having a title doesn't ensure that the future employee will have the practical skills necessary to deliver the job, not even the basic knowledge of the theory. So, more and more companies are not taking official University titles and are focusing more on demonstrating practical skills.

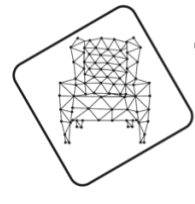
However, if you have a degree, show it on your portfolio page. That is a plus, of course.

If you don't have it, don't worry and look for showing a good portfolio, especially of collaboration with open-source projects. Having your name attached to important robotics open-source projects is the best way to make your employer interested in your proposal, even if you don't have a title.

APPENDIX

Here, you will find additional information about programming for ROS. These additional resources can help you learn and improve your ROS level as a developer.

- The ROSCON is the official annual ROS conference created by the creators of ROS, the Open Source Robotics Foundation.
- The Robotics Developers Day Conference is an annual online robotics conference created by The Construct for robotics developers. Attendees practice at the same time that the speaker explains. You can watch the presentations on The Construct's YouTube channel.
- The Robotics Developers Podcast. A weekly podcast with insights about ROS from real ROS experts worldwide.
- ROS Best practices. A series of online documents that will help you create better ROS code.
- ROS C++ Style guide.
- The Construct public repo of robot simulations (check the branch name for different ROS versions).
- ROS Components is a portal for the sale of robotics products with support for ROS.
- ROS.org official tutorials for learning ROS.
- How to build a ROS-based robot, series of YouTube videos.



The Construct