The ROS Cookbook:Building Intelligent Robot Arms from Scratch

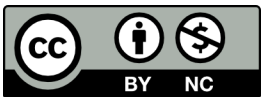
Peitao Shi and Helen Shi

April 25, 2026

The ROS cookbook: Building Intelligent Robot Arms from Scratch

by Peitao Shi and Helen Shi

Practical recipes to help you leverage building your robot arms from the scrach



April 2026: First Edition (for ROS 1)

Editors: Luis Garcia and James Martin

ISBN: 9798257200434

LCCN: 2026905921

Table of Contents

Acknowledgements

---

*I dedicate this book to my wife : **Li Li** whose unwavering support has been the foundation of my life and dreams and bring this project to completion; to my brilliant daughter **Helen**, for valuable contribution on transforming the rough draft of this book to the final manuscript.*

About the Authors

Peitao Shi is a dedicated Robotics hobbyist with over six years of academic experience in Robotics. He received his MS Robotics from Beijing University of Aeronautics and Astronautics(BUAA) and Ph.D Robotic Control Theory& Application from the Institute of Automation, Chinese Academy of Sciences(Beijing, China). Other than these, he also had solid mechanical engineering (B.S. from Shandong University, Jinan, China) and Electrical Engineering backgrounds (2nd MS from George Washington University).

Over the past 20+ years, He has built extensive C/C++ software programming experience working as software engineer across a range of companies and industries. currently, he is working as an embedded software/DSP for Thales Group.

Back to eight years ago. Peitao, worked as a contract robotic engineer for US Army (Army research Lab), where he gained his first hand-on experience with the Robot Operating system (ROS). Inspired by ROS open-architecture and concepts, which allowed him to explore new frontiers in robotics design and control, he started developing his own robot arm platform.

The entire work presented in this book has been accomplished in his spare time since 2018: He initially started from restoring a Puma260 industry robot-arm, and attempted to upgraded its controller with Altera FPGA and DRV11-J controller board. In recent years, he devoted himself into building his robot arms using ROS. In his free time, he also enjoys restoring and repairing vintage electrical devices, much like he does in building robots, a pursuit that blends creativity, engineering and passion.

Helen Shi contributed to the first chapter, Introduction, and did a lot of editing work for the rest of the chapters, from fixing grammar, spelling errors, and design diagrams to formatting the document with Markdown scripts.

Influenced by her father, she also has a strong interest in robots and AI (artificial intelligence) and assisted a lot of technical work in her father's robot projects.

In her spare time, Helen likes playing the violin and volleyball.

This page intentionally left blank.

Chapter 1

Introduction

As someone who loves robots, I bet I am not alone. Many of you might easily fall into the same trap as I found myself in: fascinated by shiny new robotic gadgets - no matter if it is an autonomous vehicle, a drone, or a humanoid robot– we buy them without a second thought, only to regret it later when finding out their functions are limited or impossible to extend.

Have you ever dreamed of owning an open robotic platform that truly belongs to you. I have owned several robots (all are robot arms), from the vintage Microbot to 1st generation industry robot PUMA260. I could program/control them for simple tasks. I also tried to replace the robot controller with FPGA and added-on hardware to do more advanced tasks. I felt them not so helpful except that I learned some skills and lessons. Not only could I not find matched Pum260 peripheral devices such as DVR-11 parallel board of the vintage PDP-11 computer, it was also hard to get a reliable software development environment (like the gcc compiler less than 2.0 running Redhat 5.0) that must be run in a virtual machine for Redhat 5.0 (1998), which is not even supported by Docker.

This issue can be completely overcome in the presence of ROS (Robot Operating System), a framework for developing robotics software. ROS is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. The software is structured as a large number of small programs that rapidly pass messages to one another. This paradigm was chosen to encourage the reuse of robotics software outside the particular robot and environment.

With the in-depth study of the ROS framework, I was able to transit my previous robot projects into the ROS environment without any significant difficulties. Especially, I can build my robot arm framework by reusing a lot of mature robot modules/nodes, and utility tools e.g., RViz, Gazebo, MoveIt!.

1.1 ROS on Raspberry PI

I started learning ROS in 2017 and first purchased two mini advanced 7-DOF desktop robot arms (inspired by 7Bot) for the purpose of learning ROS and integrated almost all my previous work into ROS Kinetic on Ubuntu16.04 . Due to the expiration of Ubuntu 16.04 EOL (end of life: April 2021), a lot of ROS packages, especially ROS robot controller spawners, new python scrips, are not easily upgraded. Therefore, I upgraded my ROS workspace to ROS (Ubuntu 18.04) and can rebuild it flawlessly without any compiling errors. Most ROS nodes were developed by `C++` (up to `C++11`) , and there are no more advanced C++ features except for the boost library.

To run ROS on an embedded system, I managed to get Ubuntu 18.04 running on the Raspberry Pi 4B mini-computer, then installed ROS Melodic on it and eventually migrated all ROS applications to Raspberry Pi platform successfully.

1.2 Robot Arm with ROS

Although there are numerous online resources and books on programming robots with ROS, I could not find a comprehensive resource to embrace a robotic project entirely with ROS. Most of them just illustrated ROS concepts, framework, and popular development tools in its ecosystem, like Moveit!, Gazebo. Although these tools can simplify your design process by creating robot configuration files that you can start and modify, there are no clear technical paths for beginners to build and design their robot arms with ROS concepts. Almost none of them systematically introduces the entire process of the robotic design process, i.e., starting from robot modeling, deriving your robot kinematics, offline simulation, and then finally real-time control.

On the other hand, ROS is an open software framework and has a large community in different groups. Every once in a while, when you are trying to get some sample projects or sample code to save your work, you will find many projects on GitHub repositories running on different ROS version platforms using various libraries or tools. It is actually not an easy job to reuse their code without fully understanding the code and resolving library dependence issues. Even if you incorporate them successfully in the beginning, later you will probably find that your system architecture becomes more intricate with many interconnected components and harder to maintain. Some projects use the latest `Eigen` mathematical library for matrix transformation, while some other projects were developed with the latest software library, like Boost `C++14` , and built with the newer version of CMake. All these constantly force you to upgrade your system to the latest OS and ROS platform. To allow you to not be distracted by all these dependencies, the rule of thumb of the author, albeit debatable, is to develop your code or your lightweight libraries if you only need a very small portion of library functions. It not only reduces your code scale and software dependency but also makes your software robust, concise, and controllable, thus easier to extend, migrate, and maintain.

1.3 Philosophy

In this book, my goal is to provide a completely open robot platform: you can change any robot arm submodules from the mechanic body to the servo motors, even the embedded platform. As long as you have your self-contained library and and modularized software stack, you can build them on different ROS platforms, you can also quickly switch hardware simply by changing hardware configuration files and/or developing new task launch files to have a new robot arm system.

ROS has evolved from ROS (1) to ROS2 fast within the last 10 years, and so has its host operation system, from Ubuntu 14.04 to the current 22.04. The fundamental ROS concepts have not been changed. So far, the author has not experienced any hurdles that cannot be resolved in ROS1, nor any urgent situation or motivation to mandatorily update my development environment to the latest version. Even for the latest ROS2 feature, like DDS (Data Distribution Service), the author has a similar lightweight solution (pluggable hardware communication mechanism) to inter-operate with different hardware. All in all, as long as the reader can master ROS essential concepts such as sharing or passing messages mechanisms. It will not be an issue to upgrade to ROS2.

To give readers a good starting point, I am trying to condense my learning process and elaborate on all sub-components' designs in this book, while hiding technical details (detours, mistakes) that I made before. Hope readers can grasp the essence of ROS programming from this book.

Installation

In this book, I am trying to hide the ROS installation process by pre-installing ROS and all robot arm applications on Ubuntu 18.04 and releasing it as a SD image for Raspberry Pi 4B. Python 2.7, 3.5 and 3.7 and `GCC-arm` cross compiler. The two best-supported languages: Python and `C++` are natively supported without extra cross-compiler installation.

This book also skips details on OS installation, building `uboot` , Ubuntu 18.04 kernel images, etc. If the reader wants to run ROS on different embedded platforms, he or she refers to ROS, Raspberry Pi and Yocto Linux on how to build customized images.

1.4 ROS Applications and Nodes

The application was first developed under Ubuntu 16.04 and Kinetic ROS. When migrating the platform to Raspberry Pi, the development OS was migrated to Ubuntu 18.04 Melodic ROS too to support 64-bit arm. Fortunately and unexpectedly, the updating from 16.06 and 18.04 was so smooth without any compiling errors.

All sample code in this book can be built and run on ROS Melodic or Kinetic. With ROS you will learn how to program and control your robot the easy way. Almost all samples in the book are developed in `C++` code except for some nodes like webserver and data learning with Tensorflow developed by Python. Readers feel free to develop their applications/nodes with Python or other programming languages like Perl, Rust.

I also extended the ROS message mechanism with UNIX Message IPC (Inter-Process Communication IPC)/TCP/UDP, which conceptually is similar to ROS2 DDS. It is up to the reader to determine which one is better when updating to ROS2.

Throughout this book, you will learn the following:

- Chapter 2: Get started: Let get your first robot arm set up and run with sample launch files.
- Chapter 3: Set up your workspace: Get familiar with the robot programming environment and the ROS framework for the robot arm. Meanwhile, set up your robot software development environment.
- Chapter 4: Review robotics concepts and robot models in the ROS environment. Some essential concepts, such as robot frames, DH parameters to characterize robot arm kinematics, and `URDF` (Unified Robot Description Format), the most popular languages to characterize a robot in ROS environment are introduced.
- Chapter 5: Robot Kinematics. Introduction of 6-DOF(Degree of freedom) robot arm forward kinematics and inverse kinematics. Two popular inverse kinematic for our robot arms are derived, and software models are designed and incorporated into our robot libraries. This fundamental ROS library is essential and almost involved in our task-level projects.
- Chapter 6: Software Architecture: Take the software architecture developed on `Hex7Bot` as an example, and illustrate how software modules are organized and how to design an open architecture software framework under the ROS environment.
- Chapter 7: Teleop: Learn how to control the robot arm over `HMI` (Human-machine Interface) by exploring some technical approaches/hardware to manipulate the robot arm, such as jogging control robot arm in joint or `Cartesian` space with joysticks, keyboard, as well as remote controlling robot arms by cost-effective portable devices over a Web App.
- Chapter 8: Robot Controller: This chapter dissects `ros_control` architecture and explores how it works and is designed. Within the `ros_control` framework, it then illustrates how to develop own robot controllers with examples for Hex7bot arms.

- Chapter 9: Hardware Communication: This chapter is out of scope of ROS programming. It mainly focuses on how to modularize internal robot hardware components and standardize HMI interface by IPC - the ROS messages extension, which translates ROS messages to robot low-level hardware messages, and supports dynamic hardware connections: (simplex, half-duplex, duplex) between different hardware components (unicast or multicast).
- Chapter 10: Visualization and Simulation. Learn how to visualize, simulate, debug your robot arm within Rviz and Gazebo. In addition, the powerful tool MoveIt(setup_assitant) is also explored with an example of task-level moving planning, and obstacle avoidance predication.
- Chapter 11: Build Dual Arms: With one robot arm system (hardware and software framework) complete, we continue exploring building dual arms for dual-arm coordination tasks. This chapter only focuses on how to build dual-arm hardware and software platform, especially discussing how to extend ROS environment/workspace for multi-robotic system.
- Chapter 12: Dual-arm perception with the depth camera. This chapter discusses the integration of a 3D depth camera, i.e., the Realsense D435 camera into a dual-arm system. Meanwhile, explore how to calibrate the camera and its 3D (point cloud) data processing for complex tasks, such as object detection, tracking, and even grasping with both end-effectors.
- Chapter 13: Robot tracking under vision guidance: Develop an application to control the left/right arm to grasp an object based on 3D camera localization, and illustrate how to integrate camera data, calculate trajectories in dynamic environments.
- Chapter 14: Deep Learning: This chapter tries to explore and track the latest `AI` technologies and seek some solutions with popular deep learning models. Particularly, we practiced deep learning's `FNN` (Feedforward Neural Network) model on our dual-arm platform with TensorFlow.
- Chapter 15: Tips and Tricks: The last chapter shares some tips on debugging and troubleshooting, common pitfalls, as well as useful utility tools and methods during robot system & software development.

1.5 Is This Book Right For You?

This book was written for robotics developers, from amateurs to professionals. It covers all the aspects involved in a whole robotic system and shows how to build your own robot arm(s) from scratch in the ROS environment. The author tried to explain all concepts intuitively with diagrams, and minimum mathematical equations, as well as a large amount of code snippets (mainly as C++ code) and UML diagrams. To understand a large amount of self-explanatory code in the book, it requires readers to have an interdisciplinary background (e.g., mechanical and electrical engineering), especially C++/Python software programming experience. Although these sample code have been developed and verified on Hex7bot robot arms, they can be applied to the same type robots without many modifications. It can also serve as auxiliary material or a programming reference book for learning robotics.

For other aspects that the book covered, such as software development environment in a Linux environment and embedded software/firmware development, readers who lack these experiences can skip those sections/chapters by getting a pre-deployed system to build their robot arm.

1.6 Code Samples

Most code samples in this book are extracted from the author's Hex7Bot project directly. You can modify or extend these code snippets for your robot development. The author is not responsible for any damages or losses from applying the code to any production.

Please be aware that the source code for the Hex7Bot project is proprietary and not available under the GPL

or as freeware. All copyrights are retained by the author, and anyone interested in obtaining the software license can contact the author directly.

1.7 Conventions

You will find several typographical conventions in this book:

- code block Code syntax highlighting for easy reading.
- Command line Monospace font to show command lines
- `Fixed width` variable,function name, namespace, data types, ros topics, etc.
- **Bold** Emphasize some terms and names.
- *Italic* Indicates URLs, email address, links, directory and pathnames, filenames, etc.

1.8 Summary

This chapter provides background for the robot arm project with ROS. Unlike regular textbooks, this book emphasizes robot system development with a real robot platform (hex7bot) closely, and the most examples or code are directly from that project. The chapters in this book are also arranged as a staged development of the project: from general system-level design or modeling to detailed software/hardware model designs, from single arm to dual-arm, from classic control to advanced topics (i.e., deep learning).

Driven by an open architecture philosophy, it is a completely open robot platform: even you don't have a Hex7Bot, but with a similar 6-DOF or 7-DOF robot arm body and Raspberry Pi, this book is also suitable for you. Let's start it.

Chapter 2

Get Started

The goal of my `Hex7bot` robot arm project was someday I can sell this platform as education or earning purpose, including robot arm mechanic body and/or ROS software solution. This product consists of a robot arm body and raspberrPI4B computer. Prior to shipping to customers, all software (Operating system, ROS, and software framework and default workspace) should be installed. The regular user should manipulate this robot arm by following user manual, powering it on, trying some basic examples, etc. In this chapter, we just presume that the user purchased this product, assembled the robot arm from its shipping box, and illustrate how to run it the first time. To some extent, you can think of this section a quick tutorial about the Hex7bot robot arm product.

Before starting the installation, we will make sure all robot arm components are in place. If you have a Hex7bot arm robot, the shipping box should contain the following items:

Parts name	**Qt**	**Description**
Robot body	1	Hex7bot robot arm assembly.
Raspberry Pi 4B	1	RaspberryPI 4B mini-computer.
microSD	1	RaspberrryPI OS image :Ubuntu 18.04/ROS Melodic.
AC power adapter	1	robot arm power adapter:$6Vx10A$.
USB-C	1	RaspberryPI USB-C Power cable: $15W@3A$.
Micro USB	1	Micro USB to USB cable: data cable.
mini-HDMI	1	Mini HDMI cable for Raspberry PI monitor.
HDMI monitor	1	not included.
USB keyboard	1	not included.
USB mouse	1	not included.

2.1 Set Up Robot Arm

1. Take out the robot body and put it on the flat surface as shown in the following figure 2.1. It is better to consolidate the robot base on the surface with screws or clamps. In addition, make sure there are no any objects within the robot's working space (avoid any collision).

2. Connect the back end with a Micro-USB cable and power adapter(6V/10A) as shown in Figure 2.2:

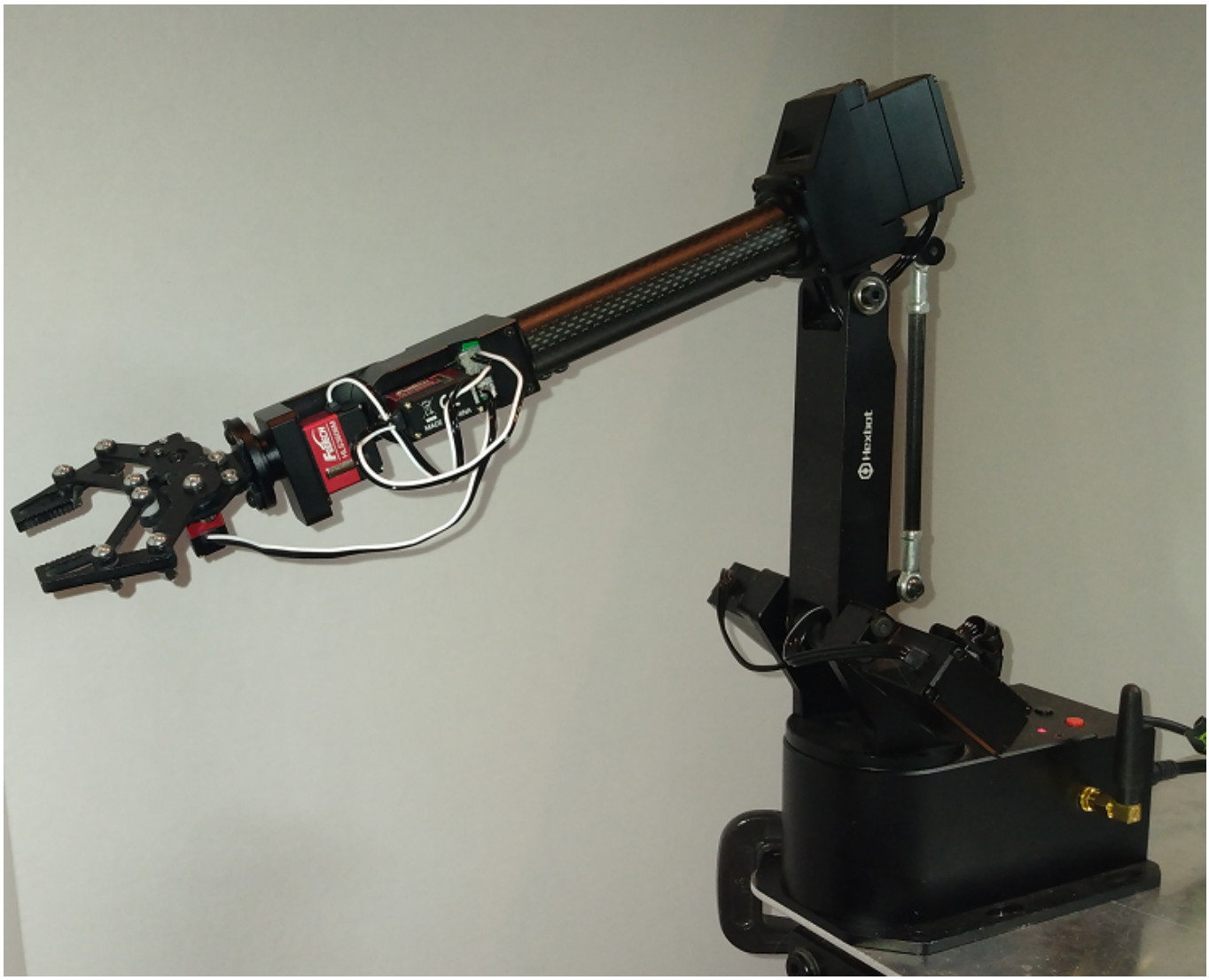


Figure 2.1: Hex7Bot Arm

Figure 2.2: Back end

2.2 Connect Raspberry PI 4B

The supported Raspberry PI model for Hex7bot Robot Arm is Raspberry Pi 4 Model B (Figure 2.3):

Figure 2.3: Raspberry Pi 4 Model B

SoC	**Memory**	**GPIO**	**Connectivity**
BCM2711	2GB 4GB 8GB	40-pin GPIO header	- 2 x micro HDMI - 2 x USB 2.0 - 2 x USB 3.0 - 3.5mm AV jack - Gigabit (1Gb/s) Ethernet RJ45 with PoE+ support - 2.4/5GHz dual-band 802.11ac `Wi-Fi` (120Mb/s) - `Bluetooth` 5, Bluetooth Low Energy (BLE) - `microSD` card slot - `USB-C` power (5V/3A (15W))

Note: two HDMI interfaces on Raspberry Pi 4B are Micro HDMI, not regular HDMI nor mini-HDMI used by mobile monitor or tablet (Figure 2.5).

Insert a 64G microSD card (Ubuntu 18.0 image) into Raspberry Pi 4B. If you have an OS image file only, you can use flash the image onto a 64G micro SD card. For instance, if you get the image named as *ubuntu-1804_raspberrypi_64G.img.xz* (this is a compressed 64G image). Use the bzip2 command to decompress it, or a tool like `7zip` on the Linux PC and burn the image to a 64G micro SD card with a USB-microSD adapter.

```
bzip2 -d -f /path/to/your/image/ubuntu-1804_raspberrypi_64G.img.xz
```

The uncompressed image size is 64GB. If there are no enough space on your host machine, run following command:

```
xzcat /path/to/your/image/ubuntu-1804_raspberrypi_64G.img.xz |
         sudo dd bs=4M of=/dev/sdX conv=fsync
sync
```

Where */dev/sdX* (e.g., */dev/sdb*) is the SD card device on Linux. Please note that it is the device name, not a partition */dev/sdb1*. This piped command chain allows for the decompression and writing process to happen simultaneously, without needing to create a separate, large uncompressed image file. After the dd command is complete, run sync to ensure all cached data is written to the SD card.

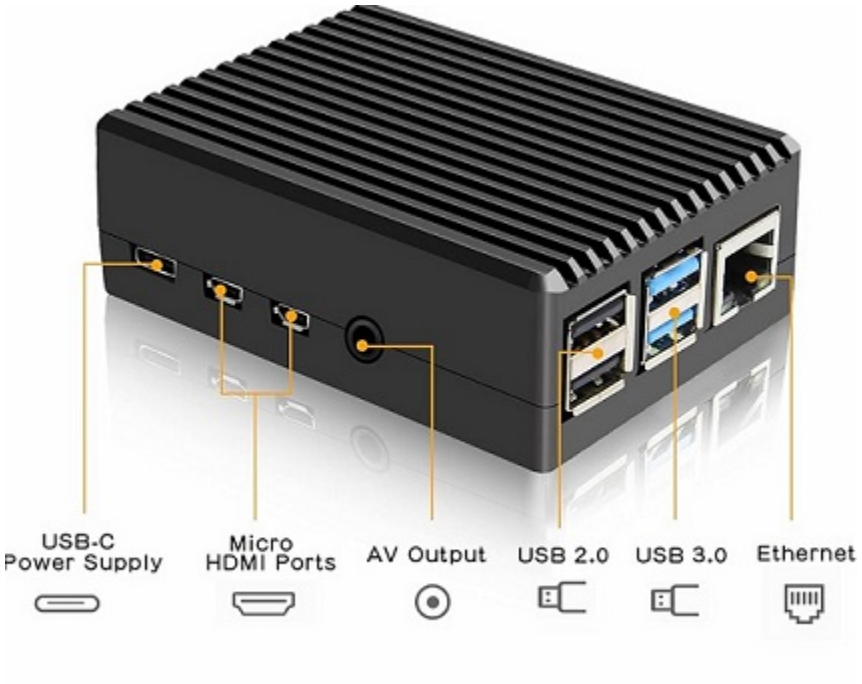


Figure 2.4: Raspberry Pi box case

Connect Raspberry Pi 4B and the robot arm as shown the following diagram (Figure 2.5):

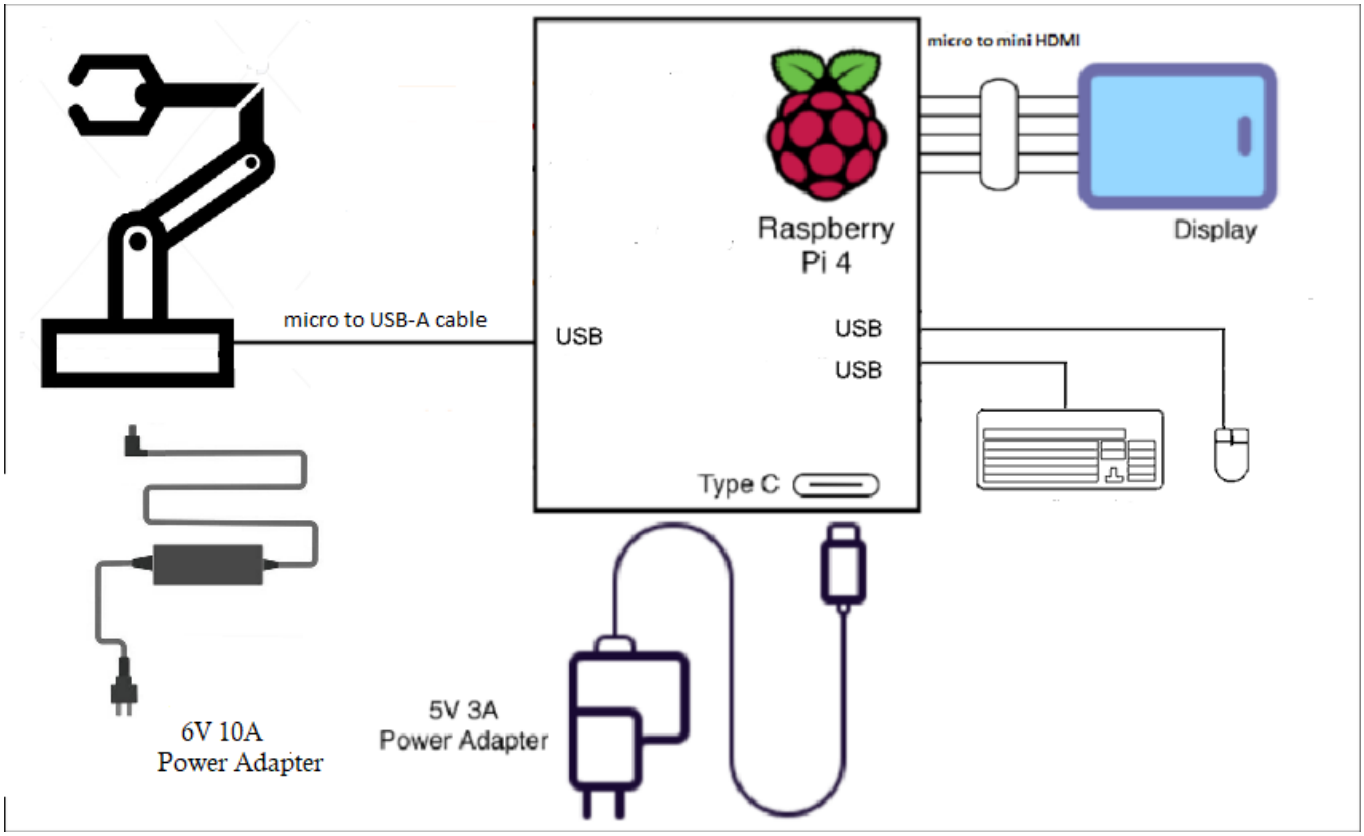


Figure 2.5: Connect cables

ℹ️ Make sure Raspberry Pi Power Adapter(not included in the package)is at least 15W(5V/3A).
ℹ️ MicroSD card (image) is inserted in Raspberry SD slot.
ℹ️ It is recommend to connect robot power adapter to the power inlet with switch button as the emergence stop function.

2.3 First Run the Robot

The ROS application in the SD image was not configured to auto-run. When Raspberry is powered on from Raspberry's bootloader to the Uboot and stopped at Ubuntu logon Window. Logging onto Ubuntu with the following default credentials:

- **Username:** ubuntu
- **Password:** password

Open one terminal and set up up the ROS working environment by running ROS setup script:

1. Setup ROS working environment by sourcing the ROS setup script.

```
source /opt/ros/melodic/setup.bash
```

2. Go to the ROS workspace and compiling all robot ARM applications. The later chapters will discuss these nodes, libraries in detail.

```
cd ~/projects
cd ros_ws
```

3. Source ROS development environment again so that the ROS can find all ROS launch files under this workspace.

```
source devel/setup.bash
```

4. Run the first ROS launch file. This launch file will start ROS core, load all included node and start accepting commands from the keyboard.

```
roslaunch rbt_controllers hex7bot_key_jog_control.launch
```

5. Type key `1` , `2` , `3` , `4` , `5` , `6` for individual joint movement. Press `+` , `-` to increase and decrease jogging steps, press "O" to open the gripper and press `C` to close the gripper.

If the robot arm could move upon the key inputs, congratulations, you have the first robot arm platform setup successfully.

2.4 Teleop Robot Arm

Alternatively, if you could make your Raspberry Pi PC connect to your home network, either over an Ethernet cable to your router or Wi-Fi connection.

1. Get your Raspberry Pi mini-PC IP address (DHCP address) from your router, say 192.168.1.164.
2. Run the same ROS launch file *kex7bot_key_jog_control.launch* as the last section.
3. Using your smartphone, tablet in your home network to access Raspberry Pi Ubuntu with url: `192.168.1.164:8080` , press the `connect` button to connect your robot arm. You can use any popular web browsers, or Apps that support WebSocket connection. If the connection is successful, a prompted dialog `Connection established` (Figure 2.6) pops up to remind the user that the robot is ready for web client control. Pressing the `OK` button, the status circular button will become green shown as figure 2.7

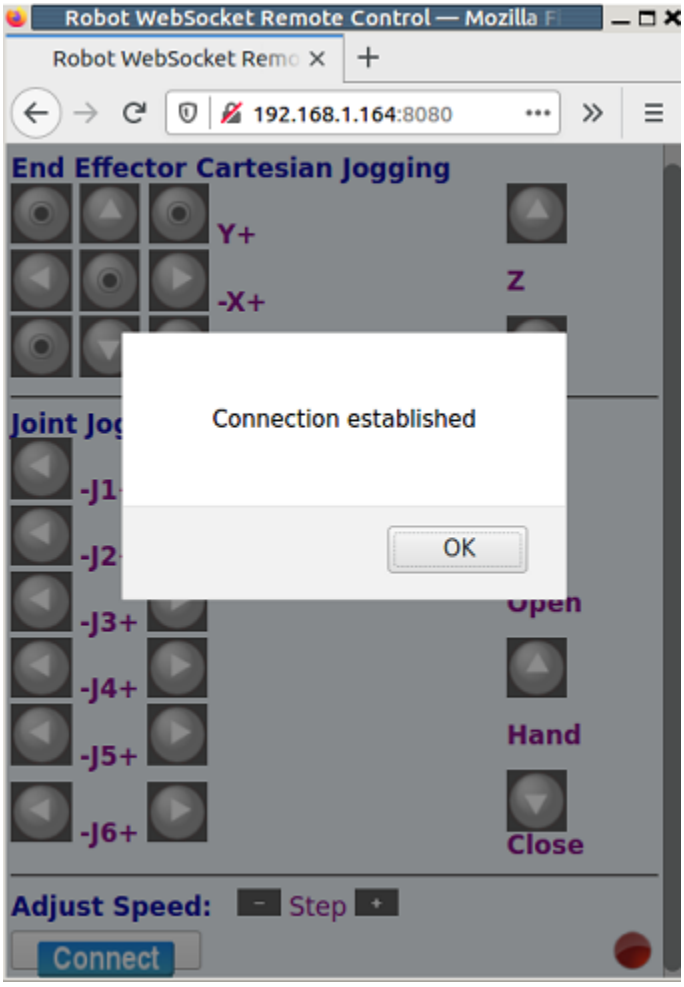


Figure 2.6: Remote connect robot arm

4. The user can control the robot in Cartesian and joint space, adjust jogging speed, and open/close gripper. Meanwhile, the robot updates(publishes) robot joints position(state) back to web client every 1 second (configurable parameter).

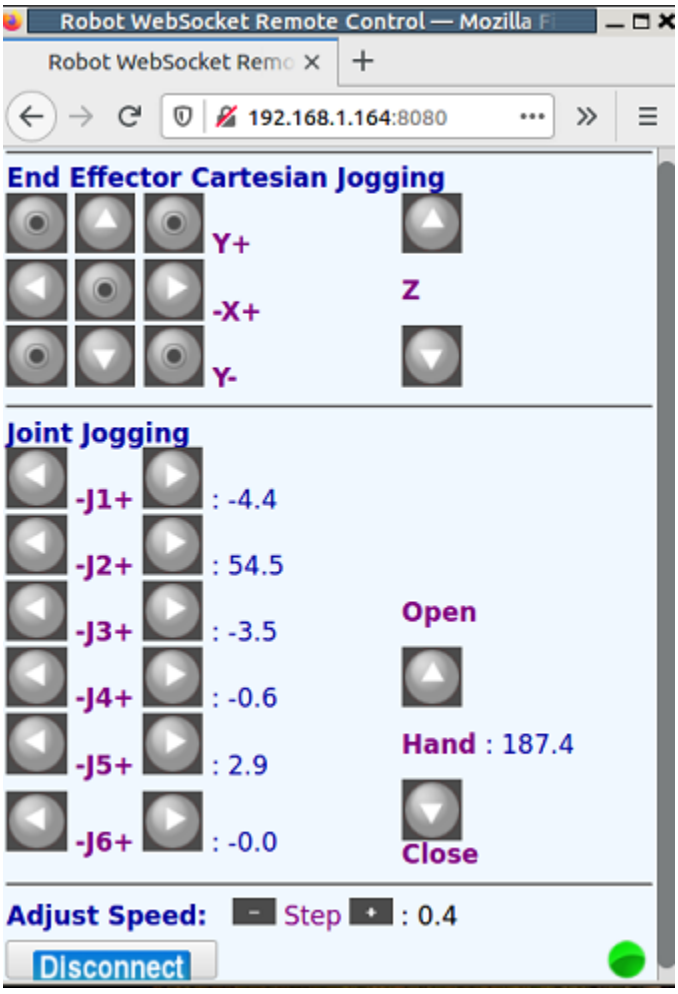


Figure 2.7: Teleop robot arm

You will learn how to develop the remote robot control app in Chapter 7.

2.5 Summary

This chapter demonstrates how to set up and run the Hex7bot arm without knowledge of ROS and software programming skills, more like a product user manual.

The author has pre-installed everything,including

- Control board: Arduino Atmega2560 firmware
- Raspberry Pi 4B: Ubuntu operating system, ROS development environment, and other dependencies, as well as demo scripts etc., required by software development).

The user can jump directly into the ROS software development environment without an installation and configuration process. The later chapters will elaborate in depth on these pre-installed samples from different design aspects. Almost all code snippets in this book otherwise mention will be found on this disk.

Prior to jumping into your ROS development journey, it is better to back up/clone your current image disk. In case some bad thing happens, you can recover your system to the original (factory) states.

The following is the command for cloning your SD image to a spare blank SD disk.

```
sudo dd bs=4M if=/dev/sdX status=progress | xz > my-raspberry-pi.img.xz
```

The next chapters will lead you on how to build and expand robot arm functions step by step.