

WRITTEN BY  
PROF. KISUNG SEO  
EDITED BY  
ALDEBARAN ROBOTICS  
& NT RESEARCH, INC

# USING NAO: INTRODUCTION TO INTERACTIVE HUMANOID ROBOTS





# WORDS FROM THE AUTHOR

---

Robots are having a profound impact in various fields including manufacturing, extreme jobs, and service sectors. Robots will have a wider range of application in the near future. Humanoid robots are attracting the most attention compared to other robots because 1) they look similar to people so they seem friendlier and are recognized as being a better fit for helping (or replacing) humans for certain tasks, 2) much like humans, biped walking is possible and jobs can be performed using both hands, and 3) they mimic the most evolutionally outstanding human form and function.

Furthermore, humanoid robots are getting a lot of attention from educators and researchers because they are surrounded by challenging issues including difficulties in walking and general motion control, effectiveness issue with processing the recognition sensors, and implementation of intelligence. It has been very tough because there aren't a lot of commercialized robots we can use to develop new controls and intelligence algorithms and for fully utilizing the advanced features of these humanoid robots in actual educational and research sites. Fortunately, more humanoid robots are being released, and out of all of them, NAO from Aldebaran Robotics is the world's most widely known humanoid robot being used for education and research. In August of 2007, it was designated as the official platform for RoboCup (Robot Soccer World Cup) instead of Sony's Aibo (puppy robot), and has been adopted in Suzhou, China starting from the 2008 competition.

Humanoid NAO consists of 25 joints that make walking and general motion control possible. Diverse interactions are possible through wireless/cable network enabled communication, cameras, infrared sensors, microphone, speakers and LEDs. The software structure is based on open source embedded Linux and supports programming languages like C, C++, URBI, Python, and .Net Framework. It also provides a graphic-based programming called Choregraphe.

This book will try to focus on using Aldebaran's humanoid NAO robot to explain the environment and tools, programming techniques, and basic theory and applications for educational and research purposes of vocational high schools, universities, and the general public.

This book is largely divided into two parts: **Chapters 1-3** for beginners and **Chapters 4-6** for advanced users. **Chapters 1-3** introduce Choregraphe and Python necessary for basic NAO robot usage. **Chapters 4-6** handle information for professional use. I would like to advise anyone just learning about the NAO robot and people who are unfamiliar with C and Python to become familiar with the information in **Chapters 1-3**. **Chapters 2** and **4-6** are recommended for anyone with previous experience in robot programming or anyone who wants to perform specialized algorithms and control commands.

---

**Chapter 1** introduces the NAO robot and the Monitor program that can be used to verify NAO's internal memory and image processing. It will also explain how to do the initial setup for the system. Because this chapter discusses NAO's special features, it would be good for readers who are not quite familiar with NAO.

**Chapter 2** will teach you how to use Choregraphe, a graphic-based programming tool, to operate NAO. Choregraphe uses a program module called Diagram to explain how to program and how to set NAO's movements in Timeline. Additionally, it will provide a description of how to use box libraries and FTP in Choregraphe.

**Chapter 3** will have a short introduction to Choregraphe scripts and Python for NAOqi. There is a basic description of Python syntax and a discussion about creating and editing Choregraphe script boxes. This would be a good chapter if you are already familiar with Python.

**Chapter 4** explains the NAOqi framework which forms the foundation of the NAO robot and the DCM used for controlling all the devices. Special characteristics including the NAOqi framework structure, file structure, and Broker as well as the NAOqi framework are used to control NAO. It also explores how to load modules into NAO using Linux, C++, and cross-compiling as well as what to do when several commands are received in Time Command. There will also be an introduction to the structures of DCM controlled devices and how to synchronize using DCM's synchronization method.

Robot kinematics in **Chapter 5** explains NAO's joint structure and provides information for each joint. The Denavit-Hartenberg (DH) method is used to explain the calculation for forward kinematics. In addition, Python will be used to create an actual forward kinematics calculation program. This chapter will also describe inverse kinematics calculations and use Python to implement the inverse kinematics calculation program for NAO's right arm. You will need quite a bit of mathematical and robotics knowledge to understand the contents in **Chapter 5**.

Comprehensive Exercises in **Chapter 6** use the information thus far to look at different methods and examples for implementing NAO's applications. Advanced Choregraphe features and expansion methods will be used here and you will be able to practice using Timeline Editor. In addition, landmark recognition will be used to create a path finding program, and the multiplication example will help you learn some of the techniques for Python and NAOqi API. Last, but not least, image recognition will be used to classify objects and inverse kinematics and NAOqi usage will be explained.

It was considerably difficult to write this book because there was a disadvantage of dealing with such a specific model of humanoid robot.

---

There wasn't much material about it, and the ones that were available were quite disorganized. I was also conflicted about how to handle the variety of readership because of the content and general difficulty of the subject matter. I am sincerely hoping that this book will serve as a good introduction to humanoid robots.

I would like to express my sincere gratitude to the people at NT Research Inc. who gave me both material and emotional support.

I would especially like to thank Jae-young Jang, Byung-yong Hyun, Su-hwan Hyun, Oh-sung Kwon, Jae-min Lee, and Young-kyun Kim in Intelligent Systems Laboratory for conducting the series of experiments with the NAO robot to help me verify the information in this book. Although every effort has been put into gathering information for this book, I am sure that there is still room for improvement, and I acknowledge that this is wholly due to the fact that I still have a lot to learn about this vast and amazing field.

**Ki-sung Suh**

# HOW TO USE THIS CURRICULUM

---

YOU ARE ALLOWED TO REPRODUCE  
THE CONTENT OF THIS BOOK AND TO SHARE  
IT WITH YOUR CLASSROOM ONLY.

Aldebaran Robotics does not warrant the accuracy of the provided content which shall be used at your own risk and under your control. Aldebaran Robotics disclaims all liability related to the use as well as the content. All rights not specifically granted herein are reserved to Aldebaran Robotics. Aldebaran Robotics and/or its licensor shall retain all rights, title and interest and ownership in and to the book and its content.

This curriculum has been done with the 1.8.16 version of Choregraphe, our programming software. However, most of the features are compatible with newer versions. The screenshot of the software included in this curriculum may be different depending of the version of Choregraphe you have.



# TABLE OF CONTENTS

---

> **WORDS FROM THE AUTHOR**

> **HOW TO USE THIS CURRICULUM**

> 1 - Introduction	8	> 5 - NAO Kinematics	182
NAO is...	10	Overview	184
Preparation	17	Transformation Matrix	185
Connecting NAO	18	NAO Structure	186
Monitor (Former name : Telepathe)	22	Kinematics	193
		Inverse Kinematics	205
> 2 - Choregraphe	26	> 6 - Comprehensive examples	216
Introduction and Interface	28	Choregraphe Application	218
Choregraphe-NAO connection	33	Motion Control – Timeline Editor	235
Box	35	Getting Directions Using	246
Event and time Centered Programming	39	Landmarks – Using Choregraphe	
Box Library	46	Memorizing the Multiplication Table	251
		– Python and NAOqi Application	
> 3 - Python	92	Combining Recognition and Movement	261
Before Getting Started	94	– Using Images for Object recognition	
Overview	95	and Grabbing Motion	
Data Types and Operators	98		
Control statements	105		
Functions	109		
Class	112		
Module	120		
Comprehensive Practice Through	124		
Choregraphe Script Modification			
References	133		
> 4 - NAOqi &DCM	134		
NAOqi Overview	136		
Structural Overview	138		
Using NAOqi	146		
Cross Compiling fo Loading	153		
Modules (Using C++, Linux)			
DCM Introduction	163		
Upper Level Architecture	164		
Low Level Architecture	168		
Preferences Files	172		
and Sub Preference Files			
DCM Bounds Methods	173		
DCM Synchronization Methods	181		

# 1 INTRODUCTION

---

## LEARNING

**Chapter 1** introduces the NAO robot and the Monitor program that can be used to verify NAO's internal memory. It will also explain how to do the initial setup for the system.

Because this chapter discusses NAO's special features, **it would be good for readers who are not yet familiar with NAO.**



---

## CONTENT

<b>1.1</b>	NAO is...	10
1.1.1	Common features	11
1.1.2	Configuration	12
1.1.3	Joint Configuration	14
1.1.4	Vision System	15
1.1.5	Audio	16
1.1.6	Software	16
<b>1.2</b>	Preparation	17
1.2.1	Package Configuration	17
1.2.2	Requirements	17
1.2.3	Software installation	17
<b>1.3</b>	Connecting NAO	18
1.3.1	Wired connection using Ethernet	18
1.3.2	Wireless connection using Wi-Fi	18
1.3.3	Using web service for default settings	20
1.3.4	File Transfer using FTP	21
<b>1.4</b>	Monitor (Former name : Telepathe)	22

# 2 CHOREGRAPHE

---

## LEARNING

**Chapter 2** will teach you how to use Choregraphe, a graphic-based programming tool, to operate NAO. Choregraphe uses a program module called Diagram to explain how to program and how to set NAO's movements in Timeline.

Additionally, it will provide a description of how to use box libraries and FTP in Choregraphe.

---

## CONTENT

<b>2.1</b>	Introduction and Interface	28
2.1.1	Menu	29
2.1.2	Box Library	30
2.1.3	Diagramming Space	30
2.1.4	3D NAO	31
2.1.5	Predefined Position Library	32
2.1.6	Video Monitor	32
<b>2.2</b>	Choregraphe-NAO connection	33
2.2.1	Connection Settings	33
2.2.2	Enslaving	34
2.2.3	File Transfer	34
<b>2.3</b>	Box	35
2.3.1	Structure	35
2.3.2	Box generation	36
<b>2.4</b>	Event and time Centered Programming	39
2.4.1	Event-Based Programming	39
2.4.2	Time-based Programming	42
<b>2.5</b>	Box Library	46
2.5.1	LED Library	46
2.5.2	Sensors Library	49
2.5.3	Logic Library	53
2.5.4	Tool Library	60
2.5.5	Math Library	63
2.5.6	Motion Library	65
2.5.7	Walk Library	72
2.5.8	Audio Library	77
2.5.9	Video Library	83
2.5.10	Tracker Library	88
2.5.11	Communication Library	89

# 3 PYTHON

---

## LEARNING

**Chapter 3** will have a short introduction to Choregraphe scripts and Python for NAOqi. There is a basic description of Python syntax and a discussion about creating and editing Choregraphe script boxes.

**This would be a good chapter if you are already familiar with Python**

---

## CONTENT

<b>3.1</b>	Before Getting Started	94	<b>3.6</b>	Class	112
<b>3.2</b>	Overview	95	3.6.1	Declaration	112
3.2.1	Determining the Dynamic Data Type	95	3.6.2	Relationship between Class and Instance	113
3.2.2	Platform-Independent Language	96	3.6.3	Constructor and destructor	114
3.2.3	Simple and Easy Syntax	96	3.6.4	Static Method	115
3.2.4	Built-in Data Structures like Strings, Lists, Tuple, and Dictionary	96	3.6.5	Operator Overloading	115
3.2.5	Automatic Memory Management	97	3.6.6	Inheritance	117
3.2.6	Diverse Libraries	97	<b>3.7</b>	Module	120
3.2.7	Scalability	97	3.7.1	Module Usage	120
<b>3.3</b>	Data Types and Operators	98	3.7.2	Module Creation	121
3.3.1	Variable Name	98	<b>3.8</b>	Comprehensive Practice Through Choregraphe Script Modification	124
3.3.2	Operator, Numeric and String Representation	98	3.8.1	Random Eyes Box Script	124
3.3.3	Lists, Tuple, Dictionary	101	3.8.2	Using Python to Create New Choregraphe Boxes	126
<b>3.4</b>	Control statements	105	<b>3.9</b>	References	133
3.4.1	The “if” Statement	105			
3.4.2	The “while” Statement	106			
3.4.3	The “for” Statement	107			
3.4.4	Range Method	107			
<b>3.5</b>	Functions	109			
3.5.1	Definition	109			
3.5.2	Return Value	110			
3.5.3	Parameter	110			
3.5.4	Pass	111			

# 4 NAOqi & DCM

---

## LEARNING

**Chapter 4** explains the NAOqi framework which forms the foundation of the NAO robot and the DCM used for controlling all the devices. Special characteristics including the NAOqi framework structure, file structure, and Broker as well as the NAOqi framework are used to control NAO.

It also explores how to load modules into NAO using Linux, C++, and cross-compiling as well as what to do when several commands are received in Time Command. There will also be an introduction to the structures of DCM controlled devices and how to synchronize using DCM's synchronization method.

---

## CONTENT

<b>4.1</b>	NAOqi Overview	136	<b>4.7</b>	Low Level Architecture	168
4.1.1	About NAOqi	136	4.7.1	Device Overview	168
4.1.2	NAOqi Term Definitions	137	4.7.2	Devices and Definitions of the Auxiliary Devices	168
<b>4.2</b>	Structural Overview	138	4.7.3	List of Communication Bus	170
4.2.1	File Structure	140	4.7.4	Device Type and List	170
4.2.2	Broker	142	4.7.5	Auxiliary device Type and List	171
<b>4.3</b>	Using NAOqi	146	<b>4.8</b>	Preferences Files and Sub Preference Files	172
4.3.1	Setting the Environment or Using Python	146	4.8.1	Introduction	172
4.3.2	Project Setup for NAOqi C++ Programming	149	4.8.2	Structural Overview	172
4.3.3	Simple Example Using NAOqi	150	<b>4.9</b>	DCM Bounds Methods	173
4.3.4	NAOqi Option	151	4.9.1	getTime	173
4.3.5	Remote Option	152	4.9.2	Set	173
4.3.6	NAOqi in NAO	152	4.9.3	createAlias	177
<b>4.4</b>	Cross Compiling fo Loading Modules (Using C++, Linux)	153	4.9.4	setAlias	178
4.4.1	Preparing the Cross-compile Tools	153	4.9.5	Special	180
4.4.2	Cross Compiling Process Using CMake and Make	154	<b>4.10</b>	DCM Synchronization Methods	181
4.4.3	NAO Robot Setup	159			
4.4.4	Module Execution	162			
<b>4.5</b>	DCM Introduction	163			
<b>4.6</b>	Upper Level Architecture	164			
4.6.1	Structural Overview	164			
4.6.2	Time command	164			
4.6.3	Time command linear interpolation	165			

# 5 NAO KINEMATICS

---

## LEARNING

Robot kinematics in Chapter 5 explains NAO's joint structure and provides information for each joint. The Denavit-Hartenberg (DH) method is used to explain the calculation for forward kinematics. In addition, Python will be used to create an actual forward kinematics calculation program.

This chapter will also describe inverse kinematics calculations and use Python to implement the inverse kinematics calculation program for NAO's right arm.

## PREREQUISITE

You will need quite a bit of mathematical and robotics knowledge to understand the contents in **Chapter 5**



---

## CONTENT

<b>5.1</b>	Overview	184
<b>5.2</b>	Transformation Matrix	185
<b>5.3</b>	NAO Structure	186
5.3.1	Link Information	186
5.3.2	Joint Information	188
5.3.3	Head Joint	190
5.3.4	Arm Joints	190
5.3.5	Pelvic Joints	187
5.3.6	Leg Joints	187
<b>5.4</b>	Kinematics	193
5.4.1	Overview	193
5.4.2	Calculating the Forward Kinematics of the Right Hand	196
5.4.3	Forward Kinematics Calculation Using Python and NAOqi	201
<b>5.5</b>	Inverse Kinematics	205
5.5.1	Overview	205
5.5.2	Using Python to calculate the forward kinematics of the right arm	209
5.5.3	Using Inverse Kinematics to Control Movements	210

# 6

# COMPREHENSIVE EXAMPLES

---

## LEARNING

Comprehensive Exercises in **Chapter 6** use the information thus far to look at different methods and examples for implementing NAO's applications. Advanced Choregraphe features and expansion methods will be used here and you will be able to practice using Timeline Editor.

In addition, landmark recognition will be used to create a path-finding program, and the multiplication example will help you learn some of the techniques for Python and NAOqi API. Last, but not least, image recognition will be used to classify objects and inverse kinematics and NAOqi usage will be explained.

---

## CONTENT

<b>6.1</b>	Choregraphe Application	218	<b>6.5</b>	Combining Recognition and Movement – Using Images for Object recognition and Grabbing Motion	261
6.1.1	Program Configuration	218	6.5.1	Object Recognition	262
6.1.2	NAOqi API	219	6.5.2	Grabbing the Object Using Inverse Kinematics Analysis	263
6.1.3	Keyframe	226	6.5.3	Grabbing the object using inverse kinematics analysis	264
6.1.4	Timeline Editor	231	6.5.4	Combining recognition and Grabbing Motion 5	267
<b>6.2</b>	Motion Control – Timeline Editor	235			
6.2.1	Saving NAO’s Actual Movements	235			
6.2.2	Adjusting NAO’s movements	238			
6.2.3	Controlling Joint Movements	242			
<b>6.3</b>	Getting Directions Using Landmarks – Using Choregraphe	246			
6.3.1	Studying and Recognizing Landmarks	247			
6.3.2	Programming	248			
<b>6.4</b>	Memorizing the Multiplication Table – Python and NAOqi Application	251			
6.4.1	Singular Number (Multiplicand) Calculator	252			
6.4.2	Adjusting the Multiply Box	254			
6.4.3	“Say” Box Expansion	255			
6.4.4	Box Placement and connection	258			

# ABOUT THE AUTHOR

---



## KI-SUNG SUH

- |              |  |
|--------------|--|
| 1986         | Yonsei University School of Electrical Engineering, BS.  |
| 1988         | Yonsei University School of Electrical Engineering, Master of Engineering.   |
| 1993         | Yonsei University School of Electrical Engineering, Ph.D.  |
| 1993-1998    | Seokeyeong University, Department of Industrial Engineering, Department of Electronic Engineering, Assistant Professor |
| 1999-2003    | Michigan State University, Genetic Algorithms Research and Applications Group, Research Associate.                     |
| 2002-2003    | Michigan State University, Electrical & Computer Engineering, Visiting Assistant Professor.                            |
| 2004-Present | Seokeyeong University, Department of Electronic Engineering, Associate Professor                                       |

Areas of interest include Intelligent Robot, Evolutionary Computation, Genetic Programming, Evolutionary Neural Networks, and Evolutionary Design.