

VERSION 1.0

3.8.2019



Arigato

NAO Documentation

Human-Robot Interaction

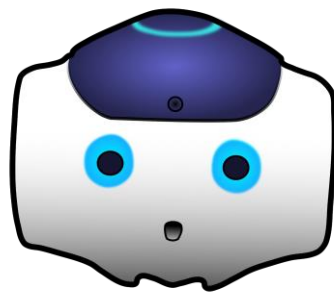
CREATED BY CASEY CHEN, HAILEY DHANENS, MATTHEW HARKER,
ANGIE QUACH, & DEREK VAUGHAN

ARIGATO, CENTRAL WASHINGTON UNIVERSITY
ELLENSBURG, WA

“Our mission is to create and showcase meaningful and exciting human-to-robot interaction using the Aldebaran NAO robots recently required by CWU.” – AriGato Robotics

CONTENTS

Part 1: Software Requirements Specification.....	X
Part 2: Testing Scenarios.....	X
Part 3: Bug Report / Problems Encountered.....	X
Part 4: User Manual.....	X
Part 5: Final Report.....	X
Part 6: References.....	X



AriGato

NAO Documentation Part 1:
Software Requirements Specification

CONTENTS

Section 1: Introduction.....	4
1.1 Problem.....	4
1.2 Scope.....	4
1.3 Definitions, Acronyms, and Abbreviations.....	4
1.4 Overview	6
Section 2: Overall Description.....	6
2.1 Project Perspective & Functionality.....	6
2.2 Suggested Functionalities	6
2.3 User Characteristics.....	7
2.4 Constraints and Limitations.....	7
2.5 Assumptions & Dependencies	8
2.6 High Level Design	9
Section 3: Requirements Overview.....	9
3.1: Requirements Checklist.....	9
3.2: Functional Requirements	9
3.3: Non-Functional Requirements	10
3.4: Requirements Q & A	10
Section 4: Requirements Analysis	11
4.1 Validity Check	11
4.2 Feasibility Check.....	11
4.3 Consistency Check.....	12
4.4 Project Timeline	12
Section 5: Prioritization of Requirements	13

SECTION 1: INTRODUCTION

1.1 PROBLEM

The AriGato team has been tasked with showcasing the capabilities of the Aldebaran NAO robots that were recently acquired by Central Washington University, by means of creating various forms of human-to-robot interaction using NAO's numerous technological features. The client has specifically asked for the robot's ability to recognize and follow verbal commands, display both basic and advanced movement options, and finally use of its cameras to some extent to all be incorporated into this software project.

1.2 SCOPE

NAO comes pre-packaged with a vast library of Application Programming Interfaces (API's) consisting of libraries of functions that will help AriGato achieve their tasked goals mentioned in Section 1.1. The team will create many modules for the NAO robot over the course of the three-month project, using the NAOqi framework (NAO's operating system and proprietary library of functions), as well as the Python programming language; these various modules will be created to accomplish all of the tasks desired by the client. These tasks will operate by interacting with the user via verbal communication and physical responses. The goal of these responses is to showcase the potential capabilities of the NAO robot, so it is important to make sure NAO has a wide variety of responses and functions of different types.

Many of NAO's features make use of its built-in high definition cameras and sound processors, which allow it to take in input through a variety of methods and produce output. The NAO does not require a constant connection to the internet, however some features will be limited when "off-the-grid"; such as when the NAO needs to fetch data or perform certain calculations. AriGato has the ability to upload programs and features to the NAO's memory unit – meaning NAO will not need to be connected to a computer or the internet at all times to be interacted with.

1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

ALDEBARAN

A French robotics company, acquired by SoftBank Robotics in 2015. Developer of NAO, NAOqi, and Choregraphe.

API

Application Program Interface.

AUTONOMOUS LIFE

The application housed on NAO that "gives it life". With Autonomous Life activated, NAO becomes visually alive – by moving, "breathing", and being aware of its surroundings.

CHOREGRAPHE

A multi-platform desktop application that allows users to create animations and behaviors for the NAO and test them in both simulated and real environments. It also allows users to monitor the NAO's visual and audio sensors.

CLIENT

Dr. Szilard Vajda, professor at Central Washington University's Computer Science department, the requester of this project

ETHERNET

A common form of network cable. It allows a connected device to join a local area network (LAN) in order to connect to and browse the internet.

LIBRARY

A collection of well-defined resources and implementations of behavior, written for/in a particular programming language for use by other developers to simplify and speed up development for a system.

MODULE

A program that is developed for use with/on the NAO robot. A module is classified as a feature for NAO that has a "trigger phrase", and then a timeline of events using Python code to have the NAO move around or speak certain phrases in responds to the corresponding trigger phrase.

NAO

An autonomous, fully programmable, humanoid robot designed by Aldebaran Robotics.

NAOQI

A Linux-based operating system stored in the robot's memory at all times; used for running and controlling features and programs.

SEE

The NAO robots cannot "see" in a physical sense but has cameras that it can use to record images to identify its surroundings.

SENSOR

Measures the robot's configurations, conditions, and its environment and sends such information to the robot for processing.

SOFTWARE

A set of computer instructions used to obtain input, and then manipulate that input in order to generate relevant output in terms of function and performance as specified by the user.

USER

A person who will interact with and make use of the NAO's various capabilities.

WI-FI

Short for "Wireless Fidelity". A means of allowing computers, smartphones, and other internet-enabled devices to communicate with one another wirelessly.

1.4 OVERVIEW

The purpose of this document is to describe the requirements of AriGato's Aldebaran NAO project. Section 2 provides an overall description of the project and its requirements, including the hardware, users and individual functionalities. This section is followed by a requirements overview (section 3) that describes more in-depth the required functionalities of the NAO robot, as well as a question and answer section. Section 4 then analyses the validity, feasibility, and consistency of these requirements. Finally, section 5 prioritizes these requirements.

SECTION 2: OVERALL DESCRIPTION

Note: This project will be independent and self-contained. It will not be a part of a larger system.

2.1 PROJECT PERSPECTIVE & FUNCTIONALITY

2.1.1 HARDWARE

This project will require the use of an Aldebaran NAO robot. AriGato will have access to a NAO robot provided by the CWU Robotics Lab, as well as an office space in the newly constructed Samuelson Building, complete with both Windows and Mac computers that can run the Choregraphe software. Choregraphe is fully functional on Windows and Mac, and available on Linux with limited support. Finally, a stable internet connection is required (and provided) for development of this project in order to access documentation pages, as well as make use of API and HTTP requests for various NAO modules.

2.1.2 USER INTERFACES (CONTROL METHODS)

The software has two possible modes of operation: The first by using a computer as a control terminal. In this mode, the NAO will be connected via Ethernet cable to a computer running Choregraphe. The Choregraphe software allows users to send premade or custom-made sets of instructions to the NAO, which will then be acted upon. This mode is limited however, as the robot can only go as far as the attached Ethernet will allow (basically, the robot must be in very close proximity to the control terminal computer). This mode is mainly for testing purposes.

The second mode, and main focus of this software project, is directly uploading the created modules to the NAO robot's memory unit located in its head. This will allow the NAO to operate with full functionality while completely disconnected from a computer, i.e., an Ethernet cord will not need to be attached to the robot for it to receive and act upon instructions. The NAO is capable of this thanks to its microphones and cameras. Users can interact with the robot using touch, motion, and spoken commands, using the multitude of sensors located around NAO's body.

2.2 SUGGESTED FUNCTIONALITIES

- Facial detection & recognition.
- Demanding physical tasks (e.g., dancing)

- Make the robot solve written equations.
- Give the robot ability to read and point out simple objects.
- Give the robot an understanding of American Sign Language (ASL) or simple hand gestures.
- Human/Robot game.
 - Like “Bop It”
 - Human is given instructions to perform within a certain amount of time
- Make robot act more human-like.
 - Casual standing position.
 - Reacts to stimulus with body language.

2.3 USER CHARACTERISTICS

There will be two classes of users that will interact with the NAO unit: regular users, and administrators (who also act as demonstrators).

The NAO robot will be interacting with a wide range of people that classify as “regular users,” such as computer science students, department faculty and staff, the general university population, university visitors, and more. The Users will have differing levels of experience. AriGato envisions the NAO to be used as a demonstration tool for any person that might be interested in the realm of computer science or robotics, such as primary school students or prospective college students. Regular users will interact with the NAO by giving it verbal commands and specific gestures or movements to invoke a corresponding response.

Administrators are people who will manage the NAO unit after the AriGato software team has delivered the final product. These users will manage the storage and maintenance of the NAO, as well as plugging the robot into control terminals, and charging it. They will also be responsible for keeping the NAO updated to any new firmware changes from Aldebaran, and possibly adding new functionality to AriGato’s software in the future, if desired. Finally, administrators will typically be the ones to demonstrate the NAO’s full range of capabilities to anyone interested in the robot.

2.4 CONSTRAINTS AND LIMITATIONS

The NAO model in use contains 8GB of onboard memory, meaning all code and additional data, such as stored images, will have to be able to be stored within this limited memory. While it is unlikely that this constraint will become a problem, it is something that should be kept in mind.

Another unfortunate constraint of the NAO robot is the overall quality of its various sensors around its body; including, but not limited to: its microphones, cameras, and gyroscopes. Although the NAO comes packed in with HD cameras in its eyes, it occasionally has difficulties locating a target (e.g., a user, or an item in front of it). Additionally, the sensors are not always accurate because a robot’s environments are always changing, so it will be taking in a lot of information at once, potentially causing slowdown or accuracy difficulties.

Furthermore, through AriGato’s independent study of the NAO’s capabilities, they have noticed that the unit’s microphones require commands to be given loudly, relatively slowly, and with extremely clear pronunciation in order for the NAO to correctly parse the commands and give

appropriate responses. It is also difficult for the microphones to pick up specific commands if it's in a crowded room with many different voices talking or faces in NAO's field-of-view. Despite these limitations, AriGato will work diligently to make their NAO software perform quickly and accurately.

Additionally, NAO should always be kept connected to the internet to make use of all the offered features. Though the NAO is functional without an internet connection, some commands will potentially require the NAO to make use of an internet connection to provide a response. This should not be an issue, as the NAO is primarily kept within university premises, which has a persistent Wi-Fi connection around the entire campus.

Given the steep cost of the robot, AriGato will also have to be mindful when programming physical activities, such as balancing or walking. If something were to go wrong, the robot may end up walking off a table, or falling onto a sharp object. The robot's arms have many mechanical joints which move along three axes. When programming an arm joint to specifically move, it could damage the hardware if the arm tries to move in a way it physically cannot. The same goes for NAO's leg and finger joints.

A small note should be made that the NAO robot has about a 90 minute active-use battery life, meaning that future users will only be able to use the NAO unplugged for around an hour and a half before it must be charged again. This constraint should not affect our development, as we will always be near charging ports and control terminal computers.

2.5 ASSUMPTIONS & DEPENDENCIES

The first assumption we've made about the NAO is that it will typically, if not always, be kept on university premises, where its status will be resting when not in use, and will always be in range of an internet connection. In the case that a future administrator of the NAO would like to take the unit out of the university network, it will need to be configured to the new location's internet settings.

The NAO robot can be programmed to do tasks such as using web APIs, but those functions would be a small portion of what it will be able to do. The assumption being made is that it will have access to the internet while being demonstrated. Not having any internet access would prevent it from being able to perform tasks using web API's.

Another assumption is that users will expect only the listed set of commands to be the set of functionalities the robot has (i.e., there is a finite list of functionality). Of course, there are a few small built-in commands that each NAO unit has (such as "What is your IP address?"), aside from these small built-in functions, the only things the NAO will be capable of doing is what we program it to do.

Finally, we are assuming that the hardware of the NAO unit will work properly at all times. However, the video and audio processing capabilities of the NAO do occasionally limit the robot from understanding every command correctly. We will be working around these potential technical limitations, and assume that the sensors will all work properly, 100% of the time. To clarify, no work will be done to improve the sensors of the NAO, as that is out of the scope of our software-focused project.

2.6 HIGH LEVEL DESIGN

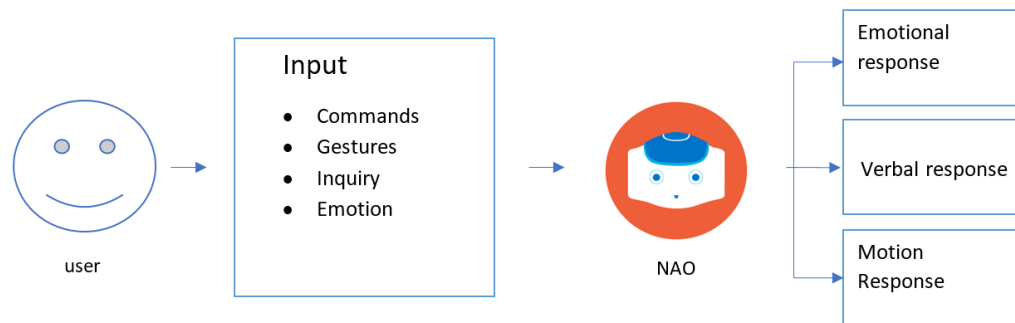


FIGURE 1 - USER-ROBOT INTERACTION

The user will input to the robot using verbal commands from a finite list. NAO will process and respond to these inputs with emotional responses, verbal responses and movement, depending on the input.

SECTION 3: REQUIREMENTS OVERVIEW

3.1: REQUIREMENTS CHECKLIST

NAO should be able to:

- Quickly recognize and act upon basic commands from a specific finite list.
- Quickly answer all questions and inquiries from a specific finite list.
- Demonstrate the movement of each appendage (arms, fingers, legs, etc.) when reacting to certain commands or gestures.
- Display advanced, possibly strenuous movement options (e.g., dancing).
- Recognize when a user has said “foul” language, and react by telling the user to watch their language, or some similar phrase.
- Access the internet for certain commands by means of API/HTTP requests.
- Use its cameras for facial detection & recognition features.
- Come provided with extensive documentation on the how’s, what’s, and why’s of the software system that has been programmed for it.

3.2: FUNCTIONAL REQUIREMENTS

- Be able to quickly recognize and act upon basic commands and gestures from a specific finite list. **[Requirement ID: FR-1]**
- Be able to quickly answer all questions and inquiries from a specific finite list. **[Requirement ID: FR-2]**
- Be able to demonstrate the movement of each appendage (arms, fingers, legs, etc.) when reacting to certain commands or gestures. **[Requirement ID: FR-3]**

3.3: NON-FUNCTIONAL REQUIREMENTS

- Be able to display advanced, possibly strenuous movement options (e.g., dancing). **[Requirement ID: NFR-1]**
- Be able to Access the internet for certain commands by means of API/HTTP requests. **[Requirement ID: NFR-2]**
- Be able to use its cameras for facial detection & recognition features. **[Requirement ID: NFR-3]**
- Be able to recognize when a user has said “foul” language, and react by telling the user to watch their language, or some similar phrase. **[Requirement ID: NFR-4]**
- Come provided with extensive documentation on the how’s, what’s, and why’s of the software system that has been programmed for it. **[Requirement ID: NFR-5]**

3.4: REQUIREMENTS Q & A

What will the system do?

- The goal of this project is to give the robot multiple functionalities of differing types to be showcased.

Should there be different modes of operation?

- Different modes of accessing NAO’s functionalities should be available.

What format should be considered for the input/output?

- Input for the robot will be verbal commands.
- Output will be verbal responses and physical movements, possibly using the internet.

Should we code responses to verbal commands?

- The robot should be able to respond to verbal commands as required by the operation it is performing.

Should the robot be able to recognize specific people?

- Yes, facial recognition is required for this project.

How should the NAO react? Verbally? With body language? Use of the NAO eyes? A mixture of these?

- NAO should be able to react verbally and with body language/movements.

Who might be changing our code in the future?

- Administrators, future CS481 students, as well as robotics students may be viewing and editing this project’s code in the future.

How easy should we make it to add our functionality to a different kind of robot? Will that even be possible with the NAO API?

- Moving NAO functionality to other Aldebaran robots, such as Pepper, could be a challenge as other robot models from this company have different functionalities than NAO, such as not having legs or having a screen on its chest.

How many NAO units will we have access to?

- The team will have access to a single NAO robot, however there are 6 NAO’s in total on the CWU premises.

When will AriGato have access to NAO?

- The team will be able to access their provided office during all of the Samuelson building's open hours (Mon-Fri, 8am-8pm). Due to the team having exclusive access to this office-space, there should be no scheduling conflicts with other teams/individuals.

What documentation is required?

- Create a fully-fledged documentation featuring software requirements specification, user manual, testing scenarios/data, bug/problem report, final progress report, and references.

In what environments will the NAO be used?

- NAO will be used exclusively within the Samuelson building during the project's three-month period, as well as possibly being showcased at CWU's SOURCE presentations in mid 2019.

What programming languages will be used?

- Python 2.7.

Where can information on the NAO API be found?

- Information on the NAO, NAOqi, and Choregraphe can easily be found by looking up "Aldebaran NAO" online and finding the corresponding documentation. For easy navigation to these pages, visit the references section at the end of this document.

Can NAO identify the face, and does NAO have any facial mapping algorithms already equipped?

- NAO has an API that locates the user's face and maps out its important features. There is no need to "reinvent the wheel" for this portion of the project.

SECTION 4: REQUIREMENTS ANALYSIS

4.1 VALIDITY CHECK

The purpose of all the suggested requirements is to showcase the abilities of the robot, as specified by the client. Following basic commands showcases the robot's ability to understand language as well as its ability to follow through commands. Using NAO's joints and "muscles" to perform various physical movements will showcase the ability NAO has to be humanoid and autonomous. Identifying the programmers and learning names will show off NAO's facial mapping and facial recognition abilities.

This project's requirements are **valid**.

4.2 FEASIBILITY CHECK

AriGato's team of software engineers have extensive experience in all tools and technologies mentioned in this document. Each of them have already developed and ran small

test programs on the NAO unit, and performed extensive research on the different libraries and capabilities of the NAO. AriGato can confirm that there will be no conflicts in the different tools that they will make use of (Python, Choregraphe, etc.) and put together. Aldebaran's provided API's will allow a very timely development period, as functions such as "move right hand upwards" for example are already written and tested. It will be up to our team to make use of and combine these provided functionalities to make an overall quality software package.

Some of the suggested functionalities are not feasible in the amount of time that we have. Learning American Sign Language (ASL) was a suggestion, however, NAO does not natively recognize finger positions, and writing a function to do this would take an incredible amount of time and study, and therefore is out of the scope based upon the team's current skills and allotted time.

It is currently unclear how the NAO API's facial recognition function works; however it does have facial mapping. It may be difficult, but facial recognition is a distinct possibility

This project's requirements are **feasible**.

4.3 CONSISTENCY CHECK

Thorough analysis of the requirements checklist has not shown any conflicts of consistency. The Choregraphe software is capable of taking in as input all of the different languages and libraries discussed in this document, and allows the AriGato team to easily test new code in a safe environment. The NAO robot is capable of taking on different "states", such as a "waiting state", where it will sit idly and wait for some command to be given, or an "active state" where the NAO is currently performing an action. This will allow all of our functions and methods to concurrently exist in the robot's memory and be acted upon when necessary (i.e., when a command is given). There will be no consistency conflicts in this software project.

This project's requirements are **consistent**.

4.4 PROJECT TIMELINE

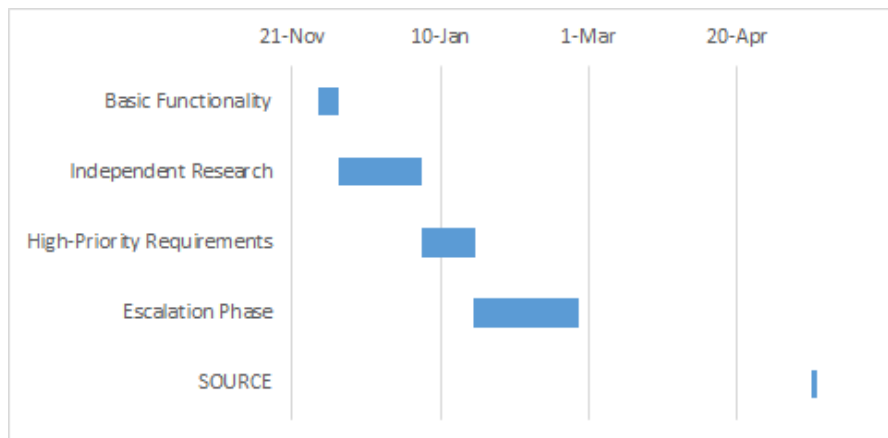


FIGURE 2 - PROJECT TIMETABLE

The team will begin this project by learning how to program basic functionality for NAO, until the winter break, in which they will begin independent research. January 3rd, when the break ends, the team will begin working on the high priority requirements. The team will divide into smaller groups of one to two, each working on a set of modules, accompanied by meeting weekly to test and discuss current module projects. When all the high priority modules have been completed, the escalation phase will begin.

During the escalation phase, team members will be in larger groups of 2 to 3, as the lower priority modules are more complex, and will require more work. Again, the teams will meet at least once a week to test modules and discuss if the groups should take on more modules. At the end of February, the final testing will begin, and the project will be presented to the client for final approval, followed by the final presentation in early March.

Finally, May 15th to May 16th, the team will present the NAO robot at CWU's Symposium Of University Research and Creative Expression (SOURCE).

SECTION 5: PRIORITIZATION OF REQUIREMENTS

Based on AriGato's conversations with their client, the main requirement they should focus on is getting the robot to be controlled independent of the Choregraphe software, that is, all features should be **autonomous** and not require connection to the control-software. Specifically, the priority is to get the NAO robot to be able to understand human language and interpret it by reacting with a corresponding action.

Goals that will absolutely be met:

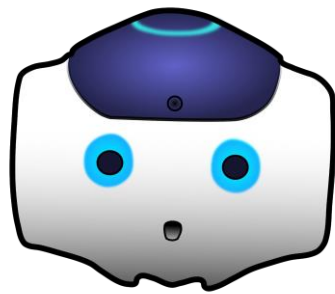
- Be able to quickly recognize and act upon basic commands and gestures from a specific finite list.
- Be able to quickly answer all questions and inquiries from a specific finite list.
- Be able to demonstrate the movement of each appendage (arms, fingers, legs, etc.) when reacting to certain commands or gestures.
- Be able to use NAO's cameras for facial detection & recognition features.
- Provide extensive documentation on the how's, what's, and why's of the software system that has been programmed for it.

Goals that we will strive to meet:

- Dance to music.
- Be able to display advanced, possibly strenuous movement options (e.g., dancing to music).
- Be able to Access the internet for certain commands by means of API/HTTP requests.

Goals that are possible, or "Stretch-Goals":

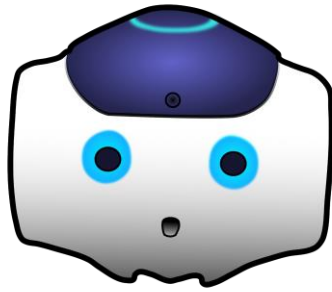
- Be able to recognize when a user has said "foul" language and react.
- Extra features, i.e., "Bop It" game.
- Read/solve written equations.



AriGato

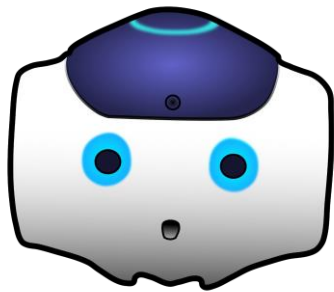
NAO Documentation Part 2:

Testing



Arigato

NAO Documentation Part 3:
Bug Report / Problems Encountered



AriGato

NAO Documentation Part 4:

User Manual

CONTENTS

Section 1: Introduction to the NAO Robot

1.1 What is NAO?.....	X
1.2 Who made NAO?.....	X

Section 2: NAO's Capabilities

2.1 Initial (Pre-Built) Capabilities.....	X
2.2 Verbal Responses (Q&A).....	X
2.3 Internet-Based Requests & Responses.....	X
2.4 Facial Detection & Recognition Responses.....	X
2.5 Basic Movement Options.....	X
2.6 Advanced Movement Options.....	X
2.7 Miscellaneous Modules.....	X

Section 3: Creating Your Own NAO Modules

3.1 Introduction to Choregraphe.....	X
3.2 The NAOqi Framework.....	X
3.3 Creating Your First Module.....	X
3.4 Integrating Your Module onto NAO.....	X

Section 4: Help

4.1 General FAQs.....	X
4.2 Troubleshooting.....	X

SECTION 1: INTRODUCTION TO THE NAO ROBOT

1.1 WHAT IS NAO?

NAO is an autonomous, humanoid, fully programmable robot. NAO robots are capable of 25 degrees of freedom, and thanks to their humanoid nature and design, are able to walk around, adapt, and interact with their surrounding environment. Furthermore, NAO has 4 directional microphones, loudspeakers, and 2 cameras capable of filming and analyzing the robot's environment, and human faces, for example. NAO is additionally capable of connecting to the internet by means of ethernet or Wi-Fi – this enables features such as http requests or big data analytics using the cloud.

All of these features add up to NAO's capabilities essentially being limitless, it is truly up to the developer's imagination to decide what the robot will eventually be capable of. Development for NAO can primarily be conducted in either Python or C++, though some other programming languages have small amounts of support as well (e.g. Java, MatLab).

Additional Specifications:

- Dimensions: 22.6 x 10.8 x 12.2 inches (574 x 311 x 275 mm)
- Weight: 12.08 pounds (5.48 kg)
- Autonomous Battery Life: 60 minutes active use, 90 minutes stationary use
- Operating System: Linux-Based NAOqi 2.8 (Linux Distro: Gentoo)
- Processor: Intel Atom E3845 @ 1.91 GHz

1.2 WHO MADE NAO?

The initial development of the NAO robot began as early as 2004. NAO was created by a French company known as Aldebaran, who was later acquired by SoftBank Robotics, a company based out of Japan, in 2015. The first public version of the NAO robot was released in 2008, however the version this project will be focusing on (NAO v6, or NAO Next Gen) was released to the public in 2014. Aldebaran also created the "Choregraphe" software that a bulk of the development of custom modules for NAO are made in.

SECTION 2: NAO'S CAPABILITIES

2.1 INITIAL (PRE-BUILT) CAPABILITIES

The original capabilities of the NAO are put on the robot by subscribing to the Aldebaran "Basic Channel." Official documentation on this channel lists and describes its capabilities.¹ Below is a small table listing some of the pre-built phrases that NAO can respond to, most of which are available in English, French, and Japanese Language settings.

¹ http://doc.aldebaran.com/2-1/nao/basic_channel_conversation.html

"How are you?"	"Can you say goodbye?"	"What can you do?"
"Tell me all you can do."	"How do I install an application?"	"How do I start an application?"
"What did I say?"	"Can you repeat please?"	"What is your IP address?"
"Are you connected to Internet?"	"What languages so you speak?"	"Speak French."
"Can you speak French?"	"Speak Japanese."	"Speak Chinese"
"Can you speak Chinese"	"Speak softer"	"Speak louder"
"Can you stand up?"	"Can you sit down?"	"Crouch."
"Lay down"	"Lift your right arm"	"Lay down on your back"
"Lay down on your belly."	"Stop Looking at me"	"What is your name?"
"Introduce yourself"	"How much do you weigh?"	"How tall are you?"

2.2 VERBAL RESPONSES (Q&A)

CAN YOU DO MY HOMEWORK?

Verbal queues:

- "Can you do my homework?"
- "Can you do my work?"
- "Can you do my math homework?"
- "Can you do my Computer Science homework?"
- Can you do my CS homework?"
- "Can you code for me?"
- "Can you do my lab for me?"

Description: NAO will respond by telling the user that they should do their own work!

DEVELOPERS

Verbal queues:

- "Who is developing your programs?"
- "Who is working on you?"
- "Who is in the capstone project?"

Description: NAO will list the team members of the AriGato capstone project.

FAVORITE CLASS

Verbal queues:

- “What is your favorite class?”
- “What is your favorite C.S. class?”
- “What is your favorite class at C.W.U.?”

Description: NAO will respond with his favorite computer science class.

FAVORITE COLOR

Verbal queues:

- “Do you like any colors?”
- “Do you have a favorite color?”
- “What is your favorite color?”

Description: NAO will respond with his favorite color.

FAVORITE PROFESSOR

Verbal queues:

- “Who is your favorite professor?”
- “Do you know any professors?”

Description: NAO will respond stating that his favorite professor is the AriGato group’s amazing supervisor, Dr. Davendra!!

FAVORITE SONG

Verbal queues:

- “Do you have a song you like?”
- “Do you like any songs?”
- “Do you like music?”
- “Do you have a favorite song?”
- “What is your favorite song?”

Description: NAO will respond with his favorite song, “Single Ladies”!

HOW OLD ARE YOU

Verbal queues:

- “How old are you?”
- “What is your age?”

Description: NAO will respond with a randomly selected humorous verbal response.

JOKES

Verbal queues:

- “Do you know any jokes?”
- “Tell me a joke.”
- “Can you tell me any jokes?”

Description: NAO will respond with a randomly selected interactive joke.

SEE YOU LATER, ALLIGATOR

Verbal queues:

- “See you later Alligator”

Description: NAO will respond with “in a while, crocodile”.

SING THE ANTHEM

Verbal queues:

- “Sing the [national] anthem.”
- “Sing [national] anthem.”
- “Can you sing the [national] anthem?”
- “Do you know the [national] anthem?”
- “Sing the Star Spangled Banner.”

Description: NAO will begin to “sing” an auto-tuned version of the American National Anthem (an .mp3 file is played over its loudspeakers) and patriotically place its hand over its heart.

Note: “[national]” in the Verbal Queues section means the word is optional within the command.

WHAT TIME IS IT?

Verbal queues:

- “What time is it?”
- “Do you know the time?”
- “Can you tell me the time?”
- “Tell me the time”

Description: NAO will respond with the current Time according to what time zone is given by IP. If Nao is connected to a VPN the time may not be local time.

2.3 INTERNET-BASED REQUESTS & RESPONSES

ROBOT MOVIE INFORMATION

Verbal queues:

- “Tell me about a movie.”
- “Do you know any movies?”
- “Do you like movies?”

Description: NAO uses an HTTP request to get information about various robot movies (from a finite list seen below) from the RottenTomatoes.com API, and then repeats the information gathered from the website to the user.

Complete Movie List:

Astro Boy	Big Hero 6	Blade Runner
Chappie	I Robot	The Iron Giant
Making Mr. Right	The Matrix	Pacific Rim
Real Steel	Robocop	Robot & Frank
Robot Overlords	Robots	Saturn 3
Spare Parts	Surrogates	The Terminator
Transformers	Wall-E	

TEMPERATURE

Verbal queues:

- “What is the current temperature of Ellensburg?”
- “What’s the current temperature?”
- “What is the temperature?”
- “How hot is it outside?”
- “How cold is it outside?”
- “How hot is it?”
- “How cold is it?”
- “What about the weather?”

Description: NAO retrieves weather information from OpenWeatherMap.com and replies with the current temperature* in Ellensburg Washington. Due to this module being intended for use in Ellensburg exclusively, if one wants to change the city, they will have to go into the module’s code and change it manually.

*This module works only in Fahrenheit, not Celsius.

WEATHER-BASED CLOTHING RECOMMENDER

Verbal queues:

- “What should I wear outside today?”
- “What should I wear outside?”
- “Should I wear a jacket today?”

- “Should I wear a jacket?”
- “What should I wear today?”
- “What should I wear?”
- “Do I need a jacket today?”
- “Do I need a jacket?”
- “What clothing should I wear today?”
- “What clothing should I wear?”
- “What outfit should I wear today?”
- “What outfit should I wear?”

Description: NAO retrieves weather information from OpenWeatherMap.com and replies with a recommendation for types of clothing to wear outdoors based upon the current weather conditions (e.g., windy weather would cause NAO to recommend a jacket). Due to this module being intended for use in Ellensburg exclusively, if one wants to change the city, they will have to go into the module’s code and change it manually.

2.4 FACIAL DETECTION & RECOGNITION RESPONSES

AGE GUESSER

Verbal queues:

- “Can you guess how old I am?”
- “Guess my age.”
- “How old do you think I am?”
- “What is my age?”
- “How old am I?”

Description: NAO will use its facial detection and mapping abilities to attempt a guess at the user’s age. The guesser is not the most accurate, however the module is enjoyable and humorous.

FACIAL RECOGNITION

Verbal queues:

- “Do you know me?”
- “Do you remember me?”
- “Do you know my name?”
- “Do you know who I am?”

Description: Provided that a human face is within view - if NAO can recognize the face within 6 seconds, NAO will greet the person, if not, the module will time out, and NAO will say “Sorry, I do not recognize you”.

Note: Learning new faces is currently only possible when the NAO unit is connected to a computer and the “Learn Face” box within Choregraphe is ran (i.e., there is no type of “Learn my face” command to have NAO learn a face on the fly).

MOOD GUESSER

Verbal queues:

- “Can you guess how I’m feeling?”
- “Guess my mood.”
- “How do you think I’m feeling?”
- “What is my mood?”

Description: NAO will use its facial detection and mapping abilities to attempt a guess at the user’s current mood based upon their facial expression. Similar to the “Age Guesser” module, the guesser is not the most accurate, however the module is enjoyable and humorous.

The moods that NAO recognizes are:

- **Happy** (NAO recognizes a wide smile).
- **Surprised** (NAO Recognizes a shocked looking face, categorized by a wide, open mouth).
- **Sad** (NAO recognizes a frown)
- **Angry** (NAO recognizes a scowl, categorized by a pursed frown, middle-lowered and furrowed eyebrows, flared nostrils).
- **Neutral** (NAO recognizes a straight, bored-looking face with no visible emotion being shown).

2.5 BASIC MOVEMENT OPTIONS

JAZZ HANDS

Verbal queues:

- “Jazz hands.”
- “Do jazz hands.”
- “Be jazzy.”
- “Can you do jazz hands?”

Description: NAO performs “jazz hands” with its hands, essentially a small wrist motion.

MOVE FINGERS

Verbal queues:

- “NAO, wiggle your fingers.”
- “Move fingers.”
- “Can you wiggle your fingers.”
- “Do your fingers move.”

- “Move your fingers.”

Description: NAO will open and close its hands to demonstrate the mobility of its hands and fingers.

NOD YES

Verbal queues:

- “Nod yes.”
- “Can you nod for me?”
- “Nod your head.”
- “Nod your head for me.”

Description: NAO will move its head up and down to demonstrate the mobility of its head/neck.

RAISE LEFT/RIGHT FOOT

Verbal queues:

- “Raise your left/right foot.”
- “Raise your left/right leg.”
- “Move your left/right foot.”
- “Move your left/right leg.”
- “Balance on your left/right foot.”

Description: NAO will lean to the side and begin to balance on one leg while lifting the corresponding foot in order to demonstrate the mobility of its legs and potential balancing capabilities.

TURN AROUND

Verbal queues:

- “Turn around”
- “Spin around”
- “Can you turn around?”

Description: Nao will turn 180 degrees

TURN HEAD LEFT/RIGHT

Verbal queues:

- “Turn your head left/right.”
- “Turn head left/right.”
- “Head left/right.”

- “Move your head to the left/right.”
- “Look left/right.”

Description: NAO will turn its head to the corresponding direction in order to further demonstrate the mobility of its head/neck.

TURN LEFT/RIGHT

Verbal queues:

- “Turn left/right”
- “Rotate left/right”
- “Can you turn left/right?”
- “Can you rotate left/right?”

Description: Nao will ask the desired degrees to turn, and then turn in the direction and distance accordingly.

WALK FORWARD/BACKWARD/LEFT/RIGHT

Verbal queues:

- “Walk <direction>.”
- “Move <direction>.”
- “Step <direction>.”
- “Take a step <direction>.”
- “Move your legs for me”
- “Move your legs”

Description: NAO will move 0.2 meters in the desired direction. The last two queues will make Nao walk forward.

Note: Directional options include forward, backward, left, & right.

2.6 ADVANCED MOVEMENT OPTIONS

DANCE MOVES

Verbal queues:

- “Can you dance?”
- “Can you dance like Beyoncé?”
- “Can you dance to Single Ladies?”
- “Do the Beyoncé.”

Description: NAO will do a short dance routine, coordinated to Beyoncé’s “Single Ladies”, which will play over its loudspeakers during the dance.

PUSHUPS

Verbal queues:

- “Pushups.”
- “Do pushups.”
- “Do some pushups.”
- “Can you do pushups?”
- “Do you know how to do pushups?”
- “Can you do press-ups?”

Description: NAO will ask how many pushups you would like to be performed (he can do between 1 and 10 inclusive), and will proceed to do that many pushups.

WARNING

NAO **must** have plenty of clear and empty space around, in front, and behind it to do pushups, and must be kept away from any ledges - otherwise damage to the NAO unit is highly possible. Furthermore, making NAO do excessive amounts of pushups without a break can lead to overheating motors and will cause NAO to shut down.

2.7 MISCELLANEOUS MODULES

NO SWEARING

Description: NAO listens for swear words or other foul language, and responds to the user telling them that they should not use such language.

Note: This is a persistent module, meaning NAO is running it at all times. There is no specific command to trigger this module, instead NAO listens for words from a list of “bad words” to trigger it.

STOP MODULE

Verbal queues:

- “Stop.”
- “Exit.”

Description: NAO immediately stops the module it is currently running.

Note: This feature is available on modules that might take a while to complete (e.g., “Sing the Anthem”), or involve multiple responses. Therefore, it is not available on every module.

SECTION 3: CREATING YOUR OWN NAO MODULES

3.1 INTRODUCTION TO CHOREGRAPHE

Choregraphe is multi-platform desktop application that allows users to visually program modules for the NAO robot (as well as other robot models created by SoftBank Robotics, such as Pepper). Choregraphe links up with the NAOqi framework (discussed in the following subsection), enabling users to easily create animations and behaviors for NAO, as well as test their creations using a simulated virtual NAO robot, or a physical robot connected via ethernet.

The Choregraphe software comes packaged with a ton of pre-built functions for NAO (known as “boxes” within Choregraphe) such as a “Say Box” which easily allows for text-to-speech programs, or a “Movement Box” which users can take advantage of to have their NAO robot move a specific distance in a specific direction. The biggest draw of Choregraphe however is the ability it gives users to create their own custom boxes, which can be combined to create an entirely custom sequence of events (known as a module) for NAO. These custom boxes can be written in a multitude of programming languages, but this documentation will be focusing on the Python language, as that is what AriGato has solely used in their project.

3.2 THE NAOQI FRAMEWORK

“NAOqi” is the NAO robot’s operating system. It is a programming framework that is built and runs off of the Gentoo Linux Distribution. NAOqi is the main software residing in NAO’s memory unit, and controls all of the robot’s motors, sensors, and functionalities. According to Aldebaran’s documentation, “it answers to common robotics needs including: parallelism, resources, synchronization, [and] events²”.

NAOqi offers a fully fleshed-out API for both C++ and Python, giving users access and manipulation of the full range of NAO’s capabilities. There is a vast library of classes and their respective functions that can be called and expanded upon to create your own modules. A few examples of these classes are “ALTextToSpeech”, “ALMotion”, and “ALLeds”, which each offer a variety of functions related to the class (e.g., ALMotion has a “moveTo” function that enables the NAO to walk). The “boxes” discussed in Section 3.1 rely on these API classes and functions to easily create modules for NAO.

² NAOqi Documentation Page: <http://doc.aldebaran.com/1-14/dev/naoqi/index.html>

3.3 CREATING YOUR FIRST MODULE

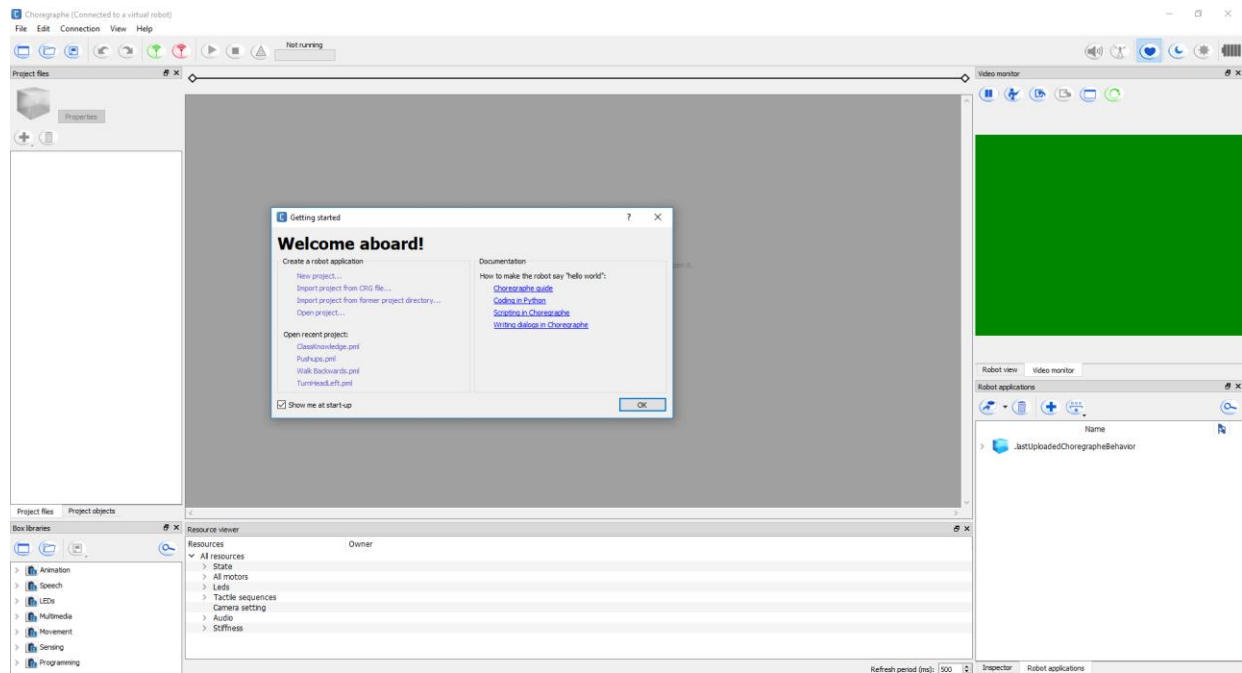


FIGURE 3 - FIRST OPENING CHOREGRAPHE

When first opening Choregraphe a window will pop up saying “Welcome aboard!” It will have a documentation section with a quick “Hello World” tutorial, and links to recent projects if there are any. To begin, either click “New project...” or exit out of the window, which will create a new project anyway. Start by navigating to the “File” menu in the top left corner of Choregraphe, and selecting “Project Properties”.

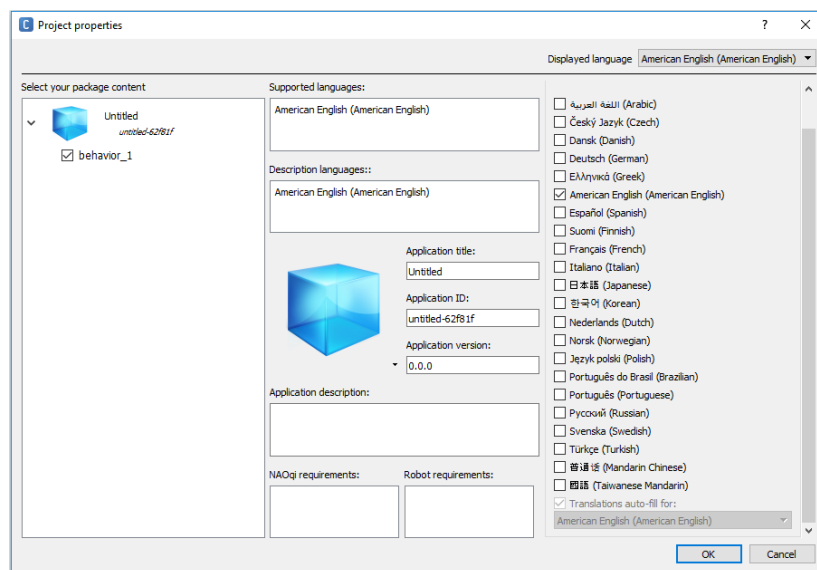


FIGURE 4 - PROJECT PROPERTIES

Fill in the “Application title” section with an accurate and descriptive title, so that when you eventually install your module onto the robot you will know exactly which module it is (it will otherwise be untitled which can become quickly confusing). Next click on the text “behavior_1” under the blue cube in the “Select your package content” section (upper left) of the window shown above in figure 2.

FIGURE 5 - BEHAVIOR PROPERTIES

Fill out the following sections shown in figure 3 after clicking “behavior_1”:

- **Name:** Make sure your module’s name is self-documenting (i.e., The name should be relevant to what the module does). A good example for a simple “Hello World” module would be “hello”, and a bad example would be “module 1”, or “myModule”.
- **Description:** This section will let future users know what your module is supposed to do, and possibly why you have created the module (to serve some certain purpose, for example). It is best to have your description be straight to the point, and not too long – but also informative with all the relevant information for the module included.
- **Nature:** Nature has three possible settings: “Interactive”, “Solitary”, and “No Nature”. **Make sure your module’s nature is set to Interactive.** This means that a user can trigger the module with verbal trigger sentences while NAO’s autonomous life feature is on. Solitary means that NAO will perform the module when he is not being interacted with, and can be interrupted at any time with Interactive modules. There is no available description for No Nature, so it is best not to make use of this mode.
- **Trigger sentences:** Trigger sentences are what NAO listens for to perform a corresponding module. Make sure each trigger sentence is unique, such that they don’t share a phrase with

another module that is already on the robot. It is also good to make a couple of similar trigger phrases for each module, so that the module can be more generally commanded.

(Example: For an arm raising module, instead of just saying “lift arm”, also put “lift YOUR arm”, “Can your lift your arm” and “lift your arm please.”)

- **Loading responses:** Loading responses are intended to be what the NAO unit will say after being told a trigger phrase, before the module is actually performed. However, in AriGato’s experience, these loading responses do not actually work, and the robot only ever uses the default phrases of “Okay” and “Let’s go!”.
- **Permissions:** Permissions gives NAO the ability to perform a module while in the process of sitting down, standing up, or sitting in the charging station. The charging station is for NAO models that are not currently in the possession of CWU, thus being not relevant to the AriGato project or this documentation.
- **Launch trigger condition:** Launch trigger conditions gives NAO the ability to perform the module autonomously (i.e., without an user prompting, according to the conditions described). **DO NOT** use trigger conditions on modules that you wish only to be triggered by a user interaction.

Now that you have set all of the relevant project properties, navigate back to the main screen (known as the “workbench”) of Choregraphe. **Note:** For this tutorial we will be showing you have to create custom boxes using Python code, but will also make short mention of how to find and use the pre-built functions that are packaged with Choregraphe.

Right click anywhere within the empty workbench, and hover your mouse over the “Create a new box” option, inside of the submenu that appears, select “Python...”. Your screen should open a new dialog box called “Edit box”, and look the same as below in Figure 4:

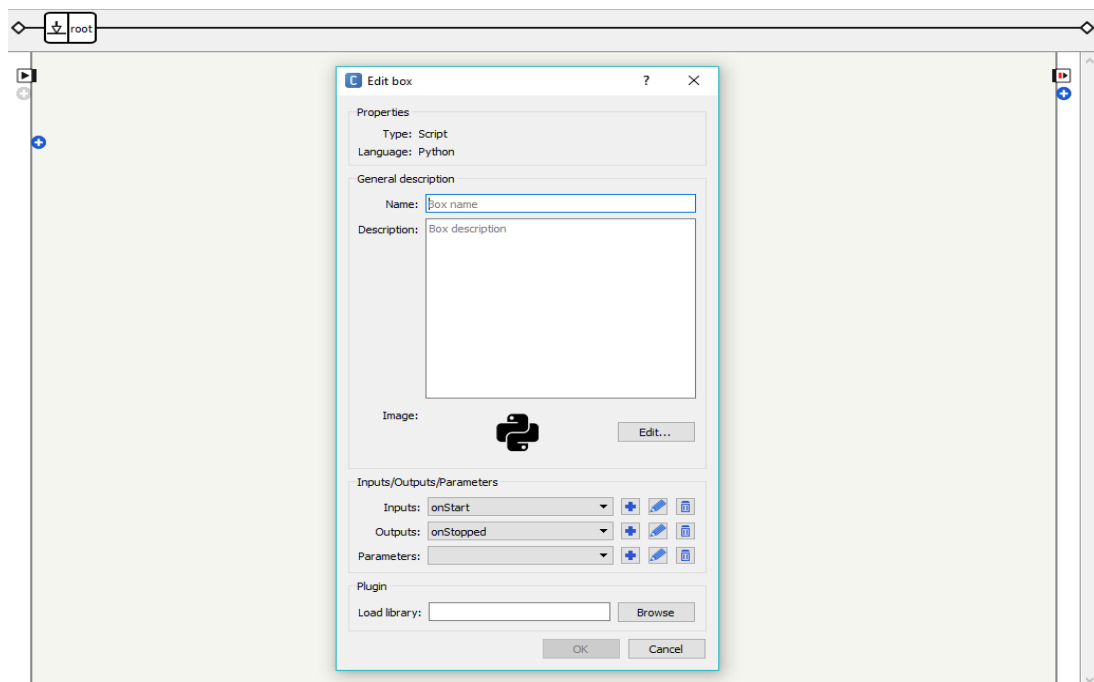


FIGURE 6 - EDIT BOX

Fill out the following sections shown in figure 4 after clicking “Python...”:

- **Name:** Your box’s name should follow suit with the module and application names you’ve previously set, in other words, it should be self-documenting and accurate to what the box will be doing. Think of this like a function name while programming. As previously stated, a good box name for a simple Hello World code would be something like “TextToSpeechBox”, or “SayHelloBox”.
- **Description:** Similar to the project description that you’ve already written, this section should be a relatively short, but informative definition of what it is that your newly created box will be doing.
- **Input:** Allows the user to add multiple start inputs.
- **Output:** Allows different outputs to be added to the module
- **Parameters:** Allows parameters to be added to the module that can be used within the module.
- **Plugin:** If your module is dependent on third-party libraries, you can add them in the load library section, under “Plugin”.

To add the script into the module, double click your python box, and a python script window will open, as shown in figure 4. Your code will be written in the `onInput_onStart` function. Delete the “pass” from this function, and uncomment the “`on_Stopped()`”

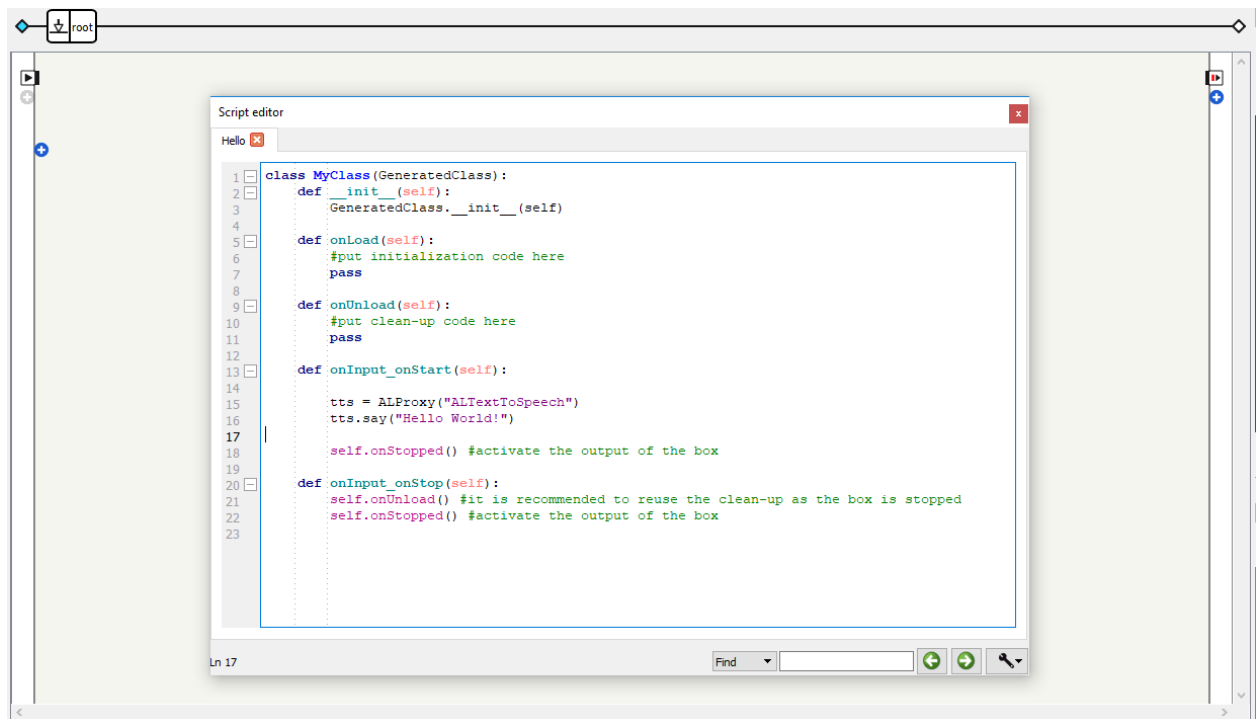


FIGURE 7 - PYTHON SCRIPT

The first line in our function in figure 5 (line 15), “`tts = ALProxy("ALTextToSpeech")`”, creates a proxy, or a link, using the “`ALProxy`” function provided in NAO’s API. This function takes a string as its parameter,

describing what the proxy is for. “ALTextToSpeech” for instance, converts text to speech, and “ALMotion” has functions that gives you the ability to control NAO’s motions.

The second line (line 16), “tts.say(“Hello World!”)”, uses the text to speech proxy that we just made, and calls its function “say()”, using the String parameter “Hello World!”. When this is called, it will make NAO say “Hello World!”.

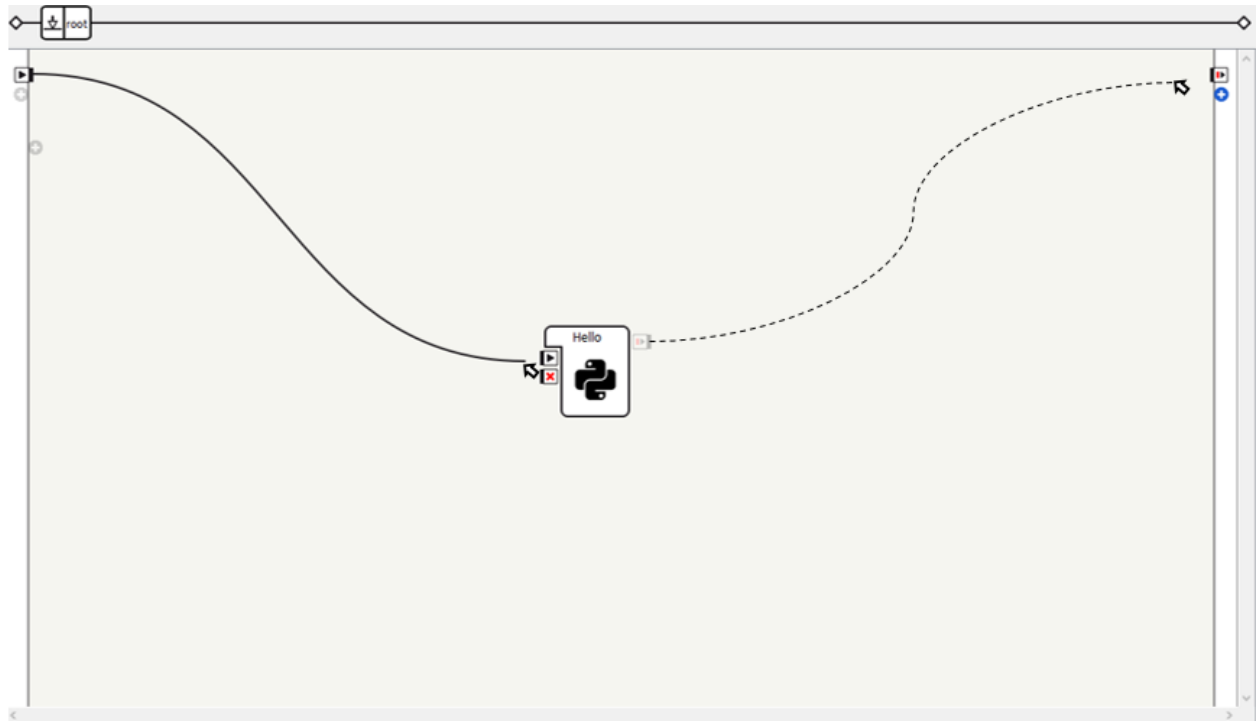


FIGURE 8 - DRAW THE LINES

To finish your project, create a connection from on start in the upper left corner to the on start on your python box as shown in figure 6. This starts your function when the behavior starts. If there is no connection from the on start in the upper left corner to any of your functions, your module will do nothing, and run infinitely. Drawing an arrow from the on stop on your function to the on stop in the upper right corner, as shown in figure 6, will exit the module after the Python script has finished.

It is important to remember to connect something to the on stop in the upper right corner, so that there is some exit condition, however if the on start signal hits a dead end in your code it will just exit, so it is not completely necessary.

3.4 INTEGRATING YOUR MODULE ONTO NAO

To integrate your module, first you must connect NAO to your computer via Ethernet cable. Then, hit the green connection button in the upper left corner of Choregraphe, which will bring up a list of connections. Click your NAO robot, and then click the select button. You may have to wait for a moment while NAO connects to your computer.

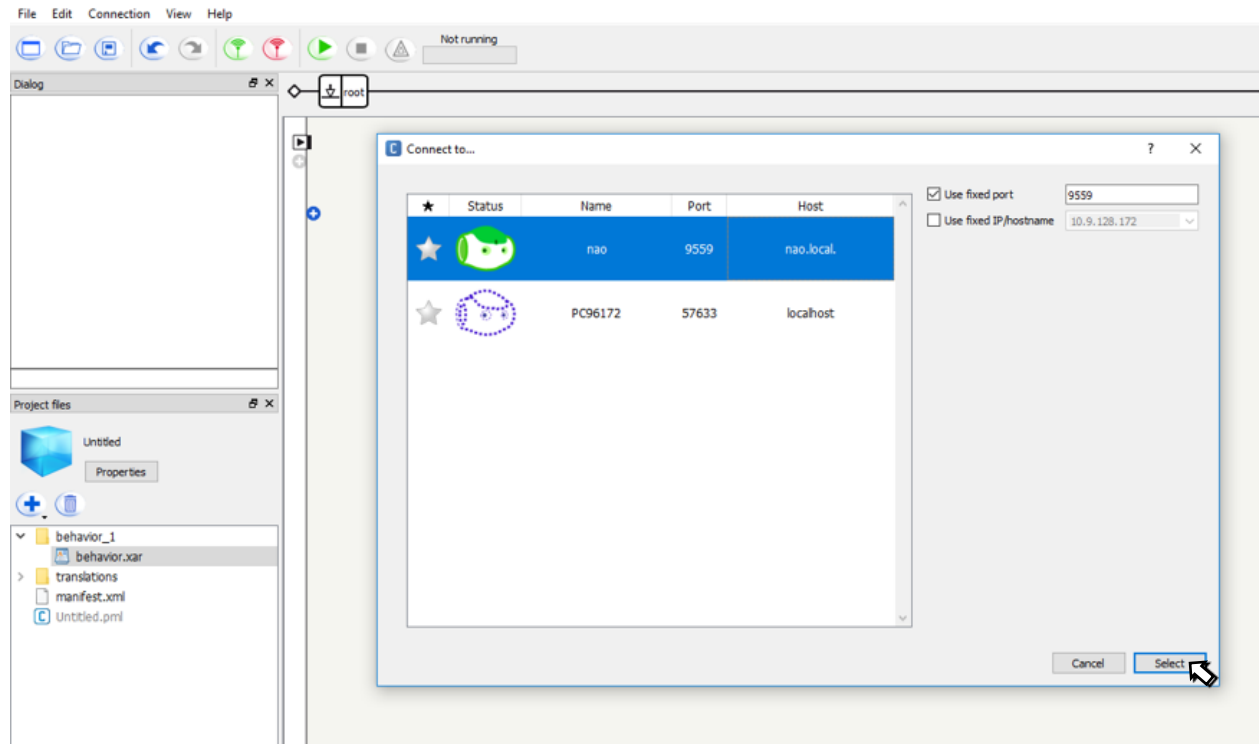


FIGURE 9 - CONNECTING YOUR ROBOT

Now adding your module should be simple. In the view tab, make sure “Robot Applications” is turned on. Then, click robot applications tab in the lower right corner, or wherever else it may appear, to open it. Now all you need to do is click the button with NAO’s head on it in the upper right corner of the robot application box shown in figure 8, and the module will be loaded and ready to go.

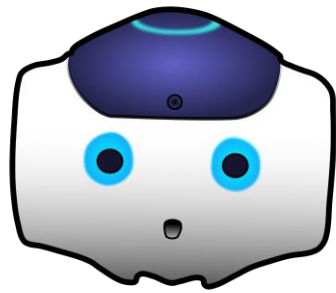


FIGURE 10 - ROBOT APPLICATIONS

SECTION 4: HELP

4.1 GENERAL FAQS

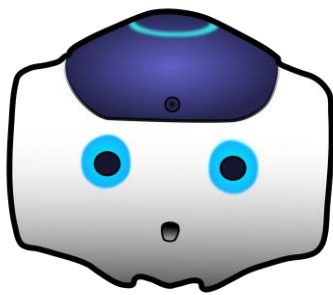
4.2 TROUBLESHOOTING



Arigato

NAO Documentation Part 5:

Final Report



AriGato

NAO Documentation Part 6:

References

