

2.1.2019



MIDTERM PROGRESS REPORT

PRESENTED BY: CASEY CHEN, HAILEY DHANENS, MATTHEW HARKER, ANGIE QUACH, DEREK
VAUGHAN

ARIGATO
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

SCOPE AND PURPOSE

Beginning early in November of 2018, the AriGato Robotics group was assigned the task of implementing human-to-robot interactions onto a factory-new Aldebaran NAO v6 robot that was acquired by Central Washington University’s robotics lab. At the time, the NAO robot was merely capable of providing some basic information about its onboard systems (e.g. battery percentage, or the unit’s IP address), a few simple vocal responses to commands (e.g. responding “Hello!” to an user), as well as some small movement options (e.g. raising its arm when prompted). None of these initial capabilities were particularly exciting, so it was up to the AriGato team to build upon these actions by using Aldebaran’s proprietary “Choregraphe” software, and some pre-developed Application Programming Interfaces (API’s).

Essentially given a blank-slate for a project, in terms of what the group could, and should implement, the group’s members began brainstorming ideas for what could fit the bill of “meaningful and exciting” interactions. NAO robots have practically limitless possibilities, provided that the people that are programming them have the necessary skills to create these interactions. Using programming languages such as Python, AriGato’s developers are able to take full control of NAO’s various appendages and sensors, even down to the angle and speed of moving its finger joints, for example.

It is AriGato’s hope that once this project is completed, come March 2019, the NAO robot will be able to be presented as a tool for generating interest in Central Washington University’s Computer Science program, as well as motivating underclassmen Computer Science students to finish their degrees by showing them what types of projects are possible when using what they will learn throughout the Computer Science program. AriGato envisions NAO to be used by a wide group of people around the University, ranging from current and prospective students or faculty, to the general Ellensburg population that may stop into the Samuelson building to see what STEM is all about. See the Gantt chart for a visual representation of our progress and plan below:

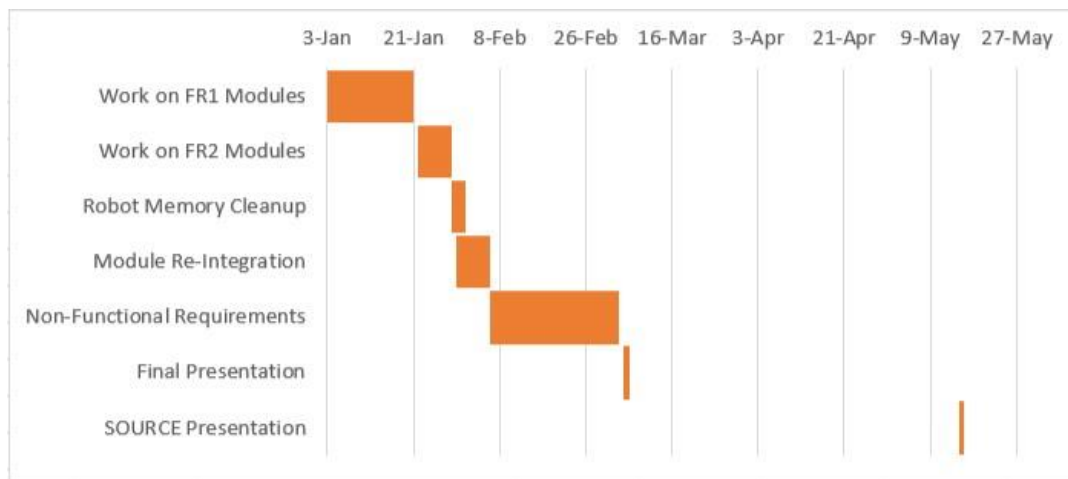
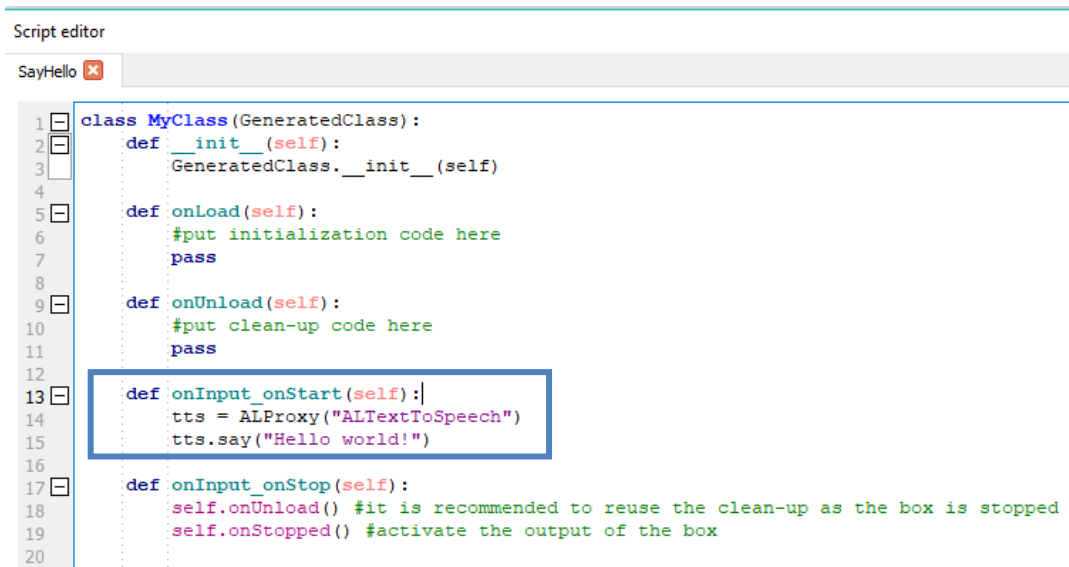


FIGURE 1: ARIGATO ROBOTICS TEAM’S OVERALL PROJECT SCHEDULE

PROGRESS

The AriGato Robotics group has made strides towards their goal of implementing interactive commands and actions onto the NAO robot. The group began by researching how to create very small text-to-speech programs using Choregraphe, and then soon moved onto making NAO perform physical movements, such as taking a step forward or backwards. All of these are possible by using the “NAOqi” framework provided by Aldebaran, which gives direct access to NAO’s various capabilities. For example, when creating a text-to-speech program in Choregraphe (written in Python), the programmer simply must create a “Proxy” to a NAOqi module known as “ALTextToSpeech”, and then simply use the “.say()” command to have NAO say whatever the programmer would like (this is even possible in a few different languages!). A small example of this text-to-speech program can be seen in the screenshot below:



```
Script editor
SayHello x

1 class MyClass(GeneratedClass):
2     def __init__(self):
3         GeneratedClass.__init__(self)
4
5     def onLoad(self):
6         #put initialization code here
7         pass
8
9     def onUnload(self):
10        #put clean-up code here
11        pass
12
13    def onInput_onStart(self):|
14        tts = ALProxy("ALTextToSpeech")
15        tts.say("Hello world!")
16
17    def onInput_onStop(self):
18        self.onUnload() #it is recommended to reuse the clean-up as the box is stopped
19        self.onStopped() #activate the output of the box
20
```

FIGURE 2: TEXT-TO-SPEECH PROGRAM

This same method of creating proxies to various NAOqi modules is the basis to all development work on the NAO unit. There is a huge library of “AL” functions at the group’s disposal, a few examples are “ALMovement”, “ALFaceDetection”, and “ALBehaviorManager”. Once creating a proxy to any of these libraries, the programmer is given access to all of the various pre-designed commands, and is able to build upon them to make more exciting behaviors.

The first small batch of interactions (Referred to as Functional Requirements 1, or FR-1) have all been completed, and successfully integrated onto the NAO unit by AriGato. Below is a complete list of the integrated FR-1 commands:

- | | |
|----------------------------|---------------------------------------|
| -“How are you?” | -“Move your hands” |
| -“How old are you?” | -“Jazz hands” |
| -“What’s the temperature?” | -“Raise right/left foot” |
| -“Nod your head” | -“Raise right/left arm” |
| -“Do you know me?” | -“Walk left/right/forwards/backwards” |

-“Sing the anthem”

-“Who designed you?”

-“What time is it?”

PROBLEMS ENCOUNTERED

The first issue that AriGato ran into while working on this project dealt with the actual installation of the Choregraphe software, where the bulk of the development work is done. Multiple members of the team had issues getting Choregraphe to properly connect to the NAOqi framework, essentially rendering Choregraphe as useless. This is because not only are all of the modules that are pre-programmed for NAO by Aldebaran housed within NAOqi, but the Choregraphe software is unable to connect to a simulated or physical robot without a connection to the framework, meaning testing was also impossible. A few of the workarounds that AriGato came up with were installing Choregraphe/NAOqi onto backup computers owned by group members (these computers thankfully did not have installation issues), or running what is known as a “bin executable file” in the background of Choregraphe (Though the team is unsure why exactly this fixed the issues with the software).

Another problem that is still plaguing the project as of right now is the infinite looping of certain modules after being installed onto the NAO robot. AriGato is still investigating what the root cause of this issue is, however the best guess is that because each member of the team individually integrated their own modules onto NAO, perhaps some settings were changed around unknowingly during the installation process; meaning each member of the team might have installed their modules in a slightly different way (different installation settings selected, for example) – causing some issues with NAO’s internal systems when attempting to select a command to run. The remedy for this problem that AriGato is currently trying out is to factory-reset the NAO unit, and have a singular group member do all of the installation work to assure no issues of consistency.

MOVING FORWARD (TO-DO)

The next steps for AriGato are first and foremost to fix the infinite looping error that is affecting the NAO unit. As discussed in the previous section, it is the team’s hope that a simple factory-reset, and reinstallation of the modules will remedy the issue. If this problem persists, the source of the problems must be from the way the modules created by AriGato were created/written, and they will have to be remade from scratch.

After the infinite looping bug has been fixed, AriGato will swiftly finish up the last of the functional requirements as discussed in the project’s Software Requirements Specification document. The top priority of these remaining functional requirements is to enhance the facial recognition module that has already been completed (“Do you know me?”), the team’s current ideas are to have NAO store information about the different people it can recognize (e.g., NAO will remember a user’s favorite color if it is told). Another feature AriGato is hoping to complete soon is the ability to play small games with the NAO unit, for example, a small trivia game, or having NAO toss a small red ball at a target.

Once the functional requirements are completely wrapped up, extensive testing will begin on the more advanced ones (games, etc.). The planned testers are the AriGato team members, CWU staff/faculty, and fellow classmates/colleagues.

The final thing that AriGato hopes to finish is the non-functional requirements. The team has decided to hold off on starting these until the functional half of the requirements has been completed, as they have been deemed more important by the team, and their client. Non-functional requirements consist of having NAO detect and respond with emotion, detecting mean-spirited language (i.e. swear words) from users, and some small dance routines.

REMAINING QUESTIONS

AriGato still needs to investigate the following things:

- What is the root cause of the “infinite looping” bug?
- Is integration with Google or Amazon’s Alexa possible on NAO?
- Is integration with a music streaming service (e.g. Spotify) possible on NAO?
- Is it possible to implement a “Stop” command, to stop NAO in the middle of responding to an action?
- How far can NAO’s movements be pushed? For example, would a push-up be possible?
- Can a machine-learning based Python script be implemented onto NAO?
- Is it possible to mute certain words from NAO’s vocabulary?

EXPECTED RESULTS

Come March 2019, AriGato is planning to have all functional requirements successfully integrated and tested. Currently, the non-functional requirements are being considered as “stretch goals”, and after a sufficient amount of research has been completed on the logistics of implementing these various functionalities, the team will reevaluate the list of non-functional requirements, and make changes if need be. One possibility for the non-functionals is that they will have to be toned-down to a simpler version if they prove too difficult to implement in the limited remaining time of the project. AriGato will continue to have bi-weekly meetings with their client, Dr. Vajda, to assure that he is satisfied with the progress that the team has made, and the eventual goals they hope to reach before the final presentation date. The AriGato Robotics team is certain that the finished product will be a quality one.