**Automatic Animal Feeder Using NAO Robot**

**Matthew Reilly, Julia Roscher**

**Draft Number #4**

**11/26/16**

**CNT 4104 Software Project in Computer Networks**

**Instructor: Dr. Janusz Zalewski**

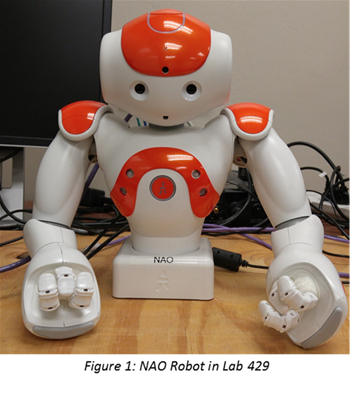
**Department of Software Engineering**

**Florida Gulf Coast University**

**Ft. Myers, FL 33965**

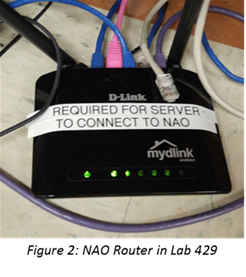
Copyright © 2016 by Florida Gulf Coast University

**1. Introduction**

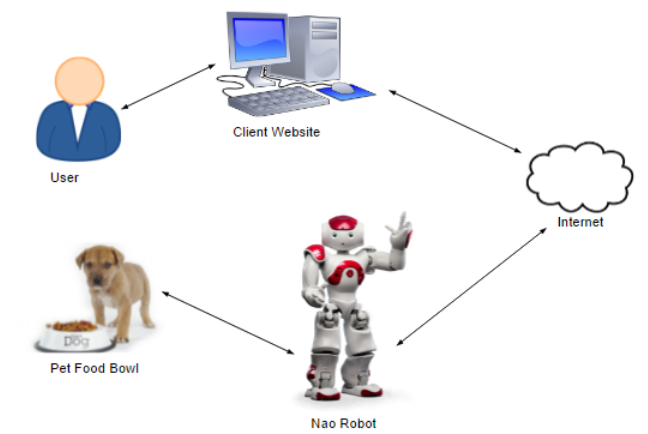
The NAO Robot from Aldebaran Robotics (Figure 1) [1] is the technology that can be used in automation. It is a humanoid robot capable of learning and performing tasks. The Robot can be programed directly in C++ and Python. However, other languages can be used to program the NAO from a remote Windows based machine [5]. In addition, the NAO Robot has the capabilities of moving its upper appendages in order to perform bodily tasks and vocalize in a way that is similar to that of an actual human. The Robot comes equipped with the ability to connect to the Internet via an Ethernet port or through the Wi-Fi. 

The objective of this project is to manipulate NAO Robot’s upper appendages to grasp and release targeted items. These targeted items will be common household utensils used for pet feeding, such as a scoop, food bowl, and dry pet food. The task of the NAO Robot is to automatically scoop out dry pet food from a set container and place the food into the pet’s food bowl. This is to be done with limited human interaction. In order for the NAO Robot to disperse food, the user will need to communicate with the robot via a web-application dedicated to that specific NAO Robot. The web-application will enable the user to tell the NAO Robot to dispense food in real time. In order to assist with the food dispersal, the NAO Robot will be equipped with algorithms that can calculate the location of the food scooper and dry pet food, to accurately place the food into the pet’s bowl. In order to accomplish this, it is best to create a permanent setup for all the utensils being used by the NAO Robot so that coordinates of each item are pre-known.

The NAO Robot animal feeder is of importance to on-the-go pet owners. They will no longer have to be afraid that their beloved pet is not being properly taken care of when they are away. With the help of the web-application the pet owner will be able to feed their pet remotely and not have to worry about rushing home to feed them. However, this form of feeding may not be optimal for more than a day, since it does not monitor the water intake of your pet.

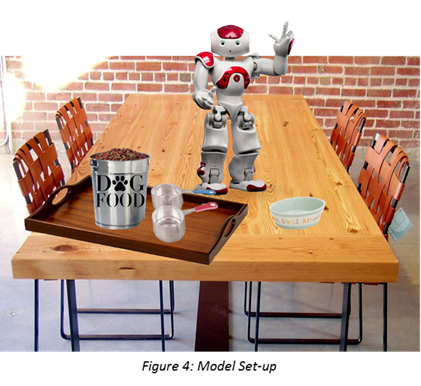
With the help of previously completed projects, the NAO Robot has the capability of being accessed outside of the local school network by using a specifically set-up router in Lab 429 [2]. This wireless router is shown in Figure 2. It will be used to connect the robot to the Internet, which will receive commands from a web-application. The web-application will be used by the user to control predefined aspects of the NAO Robot. 

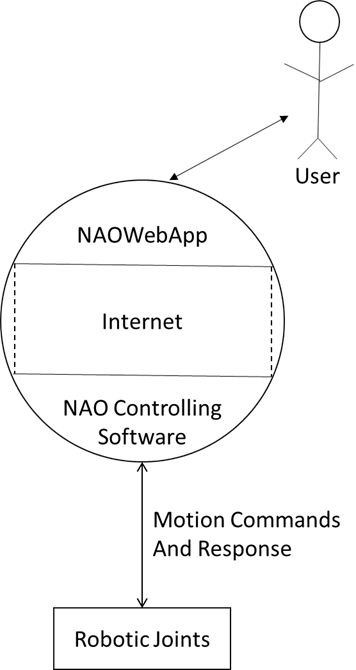
**2. Software Requirements Specification**

**2.1 Introduction to Requirements**

**Figure 3: Physical Diagram**

The objective of the project is to code the NAO Robot in a way that will allow it to pick up pet’s food out of a container, and drop the food into some type of food bowl. Figure 3 shows the

physical diagram. In this diagram, the User is using a client website to control the robot. The website will push commands onto a web-server which will then relay those commands to the NAO robot. The NAO will then send messages back through the chain of communication alerting the User if the task was completed or not. 

The Model Set-up (Figure 4) gives a visual representation of how the NAO Robot will interact with the physical world around it. The diagram details how the Robot will collect pet food, with the help of a scooper, and then deposit it into the pet’s food bowl. The real set-up will be set up slightly differently, since the NAO Robot being work with does not have any lower appendages. 

**Figure 5: Context Diagram**

Figure 5 is the context diagram of the project. This diagram goes over how each stage of the final product will work together. The User communicates with the website via the Internet and sends commands to the NAOWebApp controlling software. The software will then control the robot based upon the commands given to the software from the User. The software is the one directly communicating with the robot to move the robot's joints and displaying NAO feedback.

**2.2 Requirements Specification**

2.2.1 NAO Controlling Software Requirements

1. The NAO Controlling Software shall be able to handle commands from users received via the NAOWebApp (Roscher)
2. The NAO Controlling Software shall send commands to the NAO Robot Motion API to move upper appendages (Reilly)
3. The NAO Controlling Software shall be able to complete a sequence of commands to NAO API once the first command is initiated (Roscher)
4. The NAO Controlling Software shall send an error message back to the NAOWebApp if the command was interrupted or not completed. (Reilly)
5. The NAO Controlling Software shall be able to send a message to the NAOWebApp once the requested action has been completed (Reilly)

2.2.2 NAOWebApp Requirements

1. The NAOWebApp shall allow access only to authorized users (Roscher)
2. The NAOWebApp shall give User ability to depict when food is dispersed to animal (Reilly)
3. The NAOWebApp shall be able to receive and display an error messages from the NAO Controlling Software to User in case of failed run (Roscher)
4. The NAOWebApp shall send the command to the NAO Controlling Software to activate the motions to disperse the requested food (Reilly)
5. The NAOWebApp shall be able to access the router that the NAO Robot is on (Roscher)

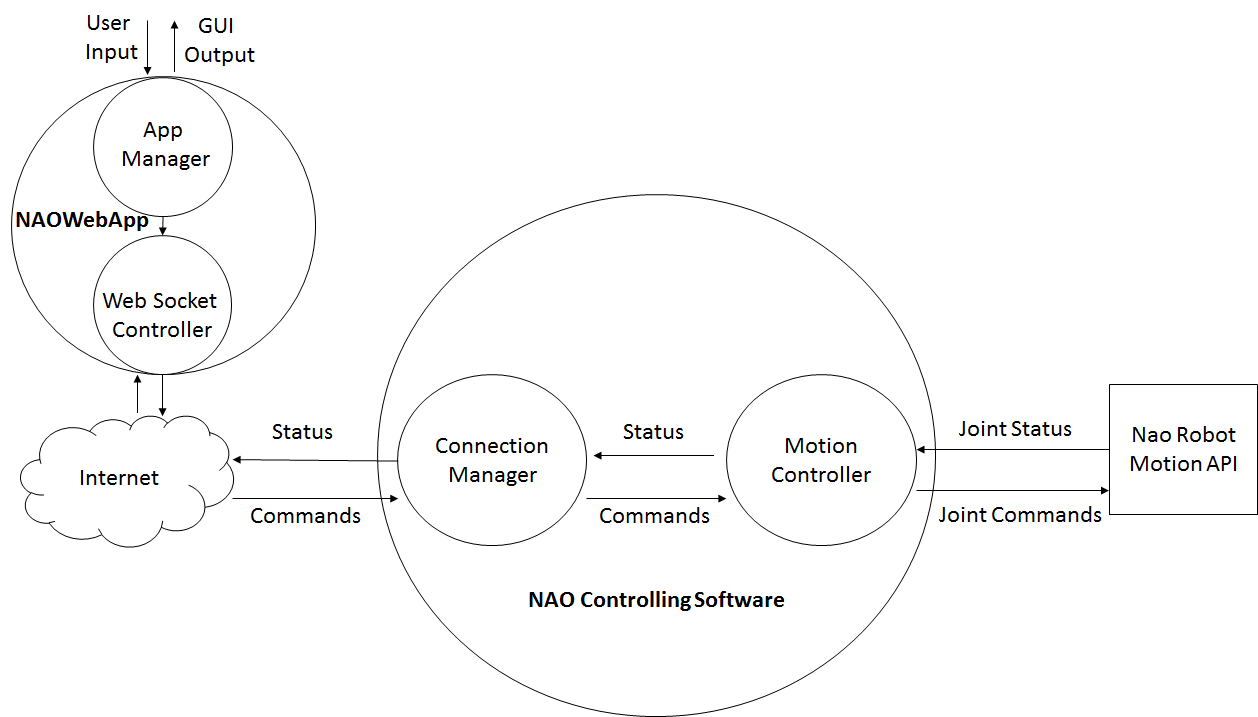
2.2.3 Design Constraints

1. The NAO Robot shall have all of its upper appendages present (Roscher)
2. The NAO Robot shall have complete dexterity of all its upper appendages (Reilly)
3. The NAO Robot shall successfully connect to the Internet via an Ethernet Cable (Roscher)
4. The NAO Robot shall have enough power to pick up the items it is expected to lift (Reilly)
5. The NAO Robot must have power in its batteries at all times.

**3. Design Description**

The design of this software is based around the NAO Controlling Software, as identified in Figure 5. The NAO Controlling Software runs on the NAO Robot’s on board computer. This Controlling Software will interact with the NAOWebAPP software via an Internet connection. The Controlling Software will run a script for the job that is given to it. This script will control the joints of the NAO Robot. The design description also covers how the NAOWebAPP interacts with the NAO Controlling Software which controls the NAO Robot’s movements.

**3.1 Software Architecture**



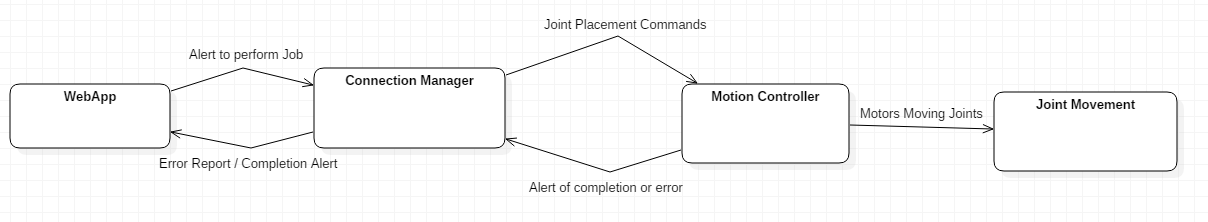
**Figure 6: Software Architecture Diagram**

The NAO Controlling Software Architecture Diagram (Figure 6) shows an overview of the relationship of the software components. The main components of the NAO Controlling Software include a Connection Manager and Motion Controller, while the main components of the NAOWebApp include the App Manager and the Web Socket Controller.

Users will connect to the NAO Controlling Software through the NAOWebApp. The NAOWebApp provides the GUI for the user to request for an action to be completed by the NAO Robot. The status of the request will also be present of this display. The App Manager takes in the User Input from the GUI and sends it to the Web Socket Controller. The Web Socket Controller sends the commands to the NAO Robot through the Internet to the NAO Controlling Software.

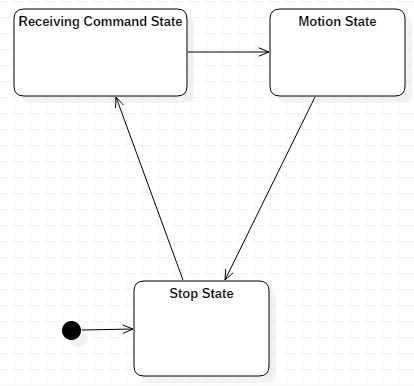
Clients connect with the Connection Manager via the Internet. Commands received by the Connection Manager from the NAOWebApp will be sent to the Motion Controller. The Motion Controller will then send the execution of the commands to the NAO Robot Motion API. The NAO Robot Motion API, is responsible for performing the joint movements necessary to scoop pet food into the pet’s bowl. The status of the commands will be sent back to Connection Manager and then to the client. Status messages will inform the User if their commands were successfully executed or not.

**3.2 Detailed Design**



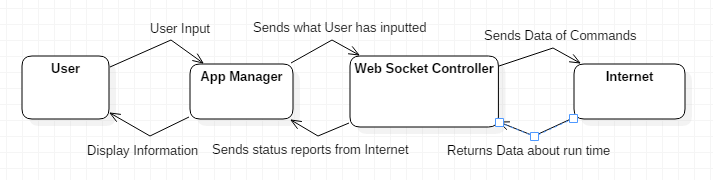
**Figure 7: Dynamic State Diagram for NAO Controlling Software**

In Figure 7, the Dynamic State Diagram goes over how each component of the software behaves with other components and modules. In this diagram, the NAOWebApp is the one who makes initial contact with Connection Manager which gathers the information from commands sent over the Internet. The Connection Manager sends the joint motion commands stored on the NAO Robot’s Computer to the Motion Controller which then processes those commands to directly send commands to the moving joints of the robot. The motion controller sends back an alert if any errors occur in the motions of the robot to the connection manager, which sends a completion report/error report to the NAOWebApp.



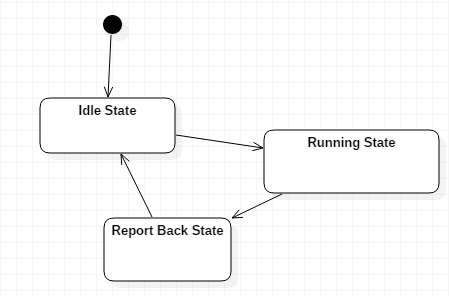
**Figure 8: State Chart for NAO Controlling Software**

As per the design constraints, the NAO Controlling Software will have three states. The first state will be the Stop State. The Stop State is where the software is at “sleep” until communication with the software has been reached. This Stop State is when the NAO has also finished its task. The Receiving Command State is when the NAO Controlling Software is actively receiving a command and preparing to send the appropriate motion commands to the NAO Robot’s joints. The Motion State is when the Robot is actively moving; the software is listening for any errors and will receive an exit code after the motions have been completed.



**Figure 9: Dynamic State Diagram for NAOWebApp**

The Dynamic State Diagram for the NAOWebAPP (Figure 9) shows the connections between the components of the NAOWebApp and how they interact with each other. The User initially will input a command which is put right to the App Manager. The App Manager translates what the User inputs into the NAOWebApp. The App Manager sends data from the User to the WebSocket Controller which controls how the data is being sent to the NAO Controlling Software. The commands and reports transfer over the Internet via the Web Socket Controller.



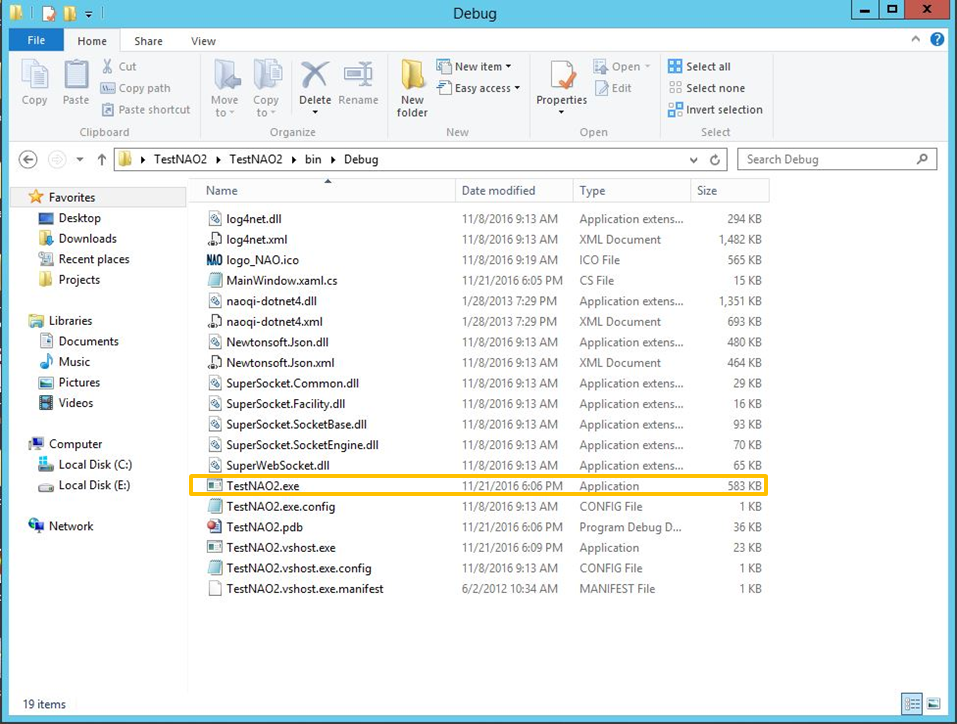
**Figure 10: State Diagram for the NAOWebApp**

The State Diagram for the NAOWebApp (Figure 10) goes over each state that the NAOWebApp goes through while operating. The WebApp initially starts at the Idle state, where the NAOWebApp is waiting for the User to interface with the machine. The Running State is when the App has been triggered by the User to send commands via the Internet to the NAO Controlling Software. The NAOWebApp then goes into the Report Back State where the App will listen for the NAO Controlling Software to report back on whether or not the run was successful, or if errors had come back. After the NAO has finished its run, and reported back, the NAOWebApp will go back into the Idle State, waiting for the user's next input.

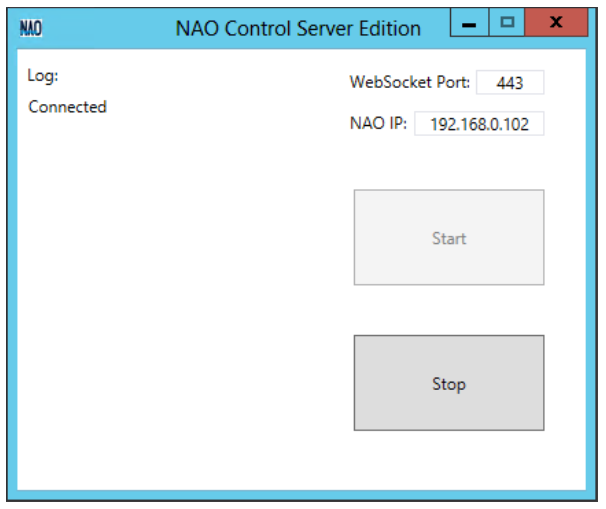
**4. Implementation and Testing**

**4.1 Assembly and Coding**

The The NAO Robot from Aldebaran Robotics has been already assembled at the beginning of this project. The NAO Robot, connects to the Internet through the DLINK DIR-605L router which is connected to the ROCK server [3] within the lab. In order to connect to the NAO it is important to learn what the NAO’s IP address is. By briefly pressing the power button, after the NAO is turned out, the NAO will say its IP address of 192.168.0.102.

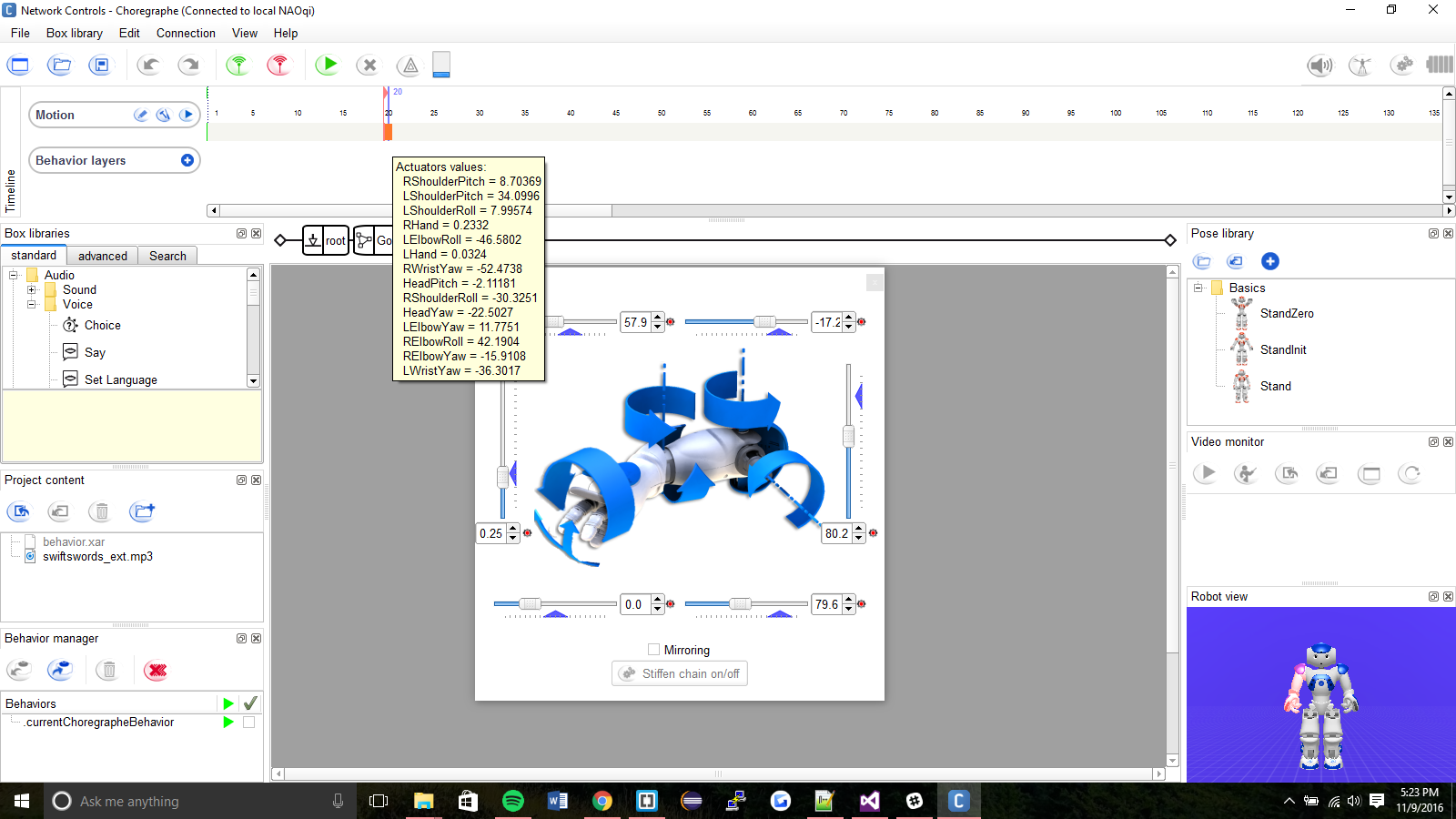


**Figure 11: File Location of Connection Manager**

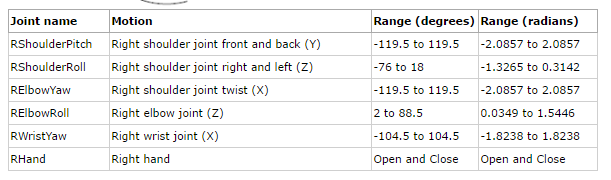
In order to connect a website remotely to the NAO using websockets, an application has to be created to translate the commands from the website to the NAO. This application has already been created in a previous project [4]. The application is titled “TestNAO2.exe” and is located in the project folder titled “TestNAO2” (Figure 11). Once the application is launched, it is called “NAO COntrol Server Edition” (Figure 12). This application is the Connection Manager module component. This program must be running and connected to the NAO robot in order for instructions to be successfully passed from the website to the NAO. The purpose of the application is to translate commands from the website to Rock server, where the application is located, to the NAO. When using the NAO Controlling Software, the WebSocket Port has to be set to 433. This port value can be changed if need be, but for the purpose of this project port 443 is used. Likewise, the NAO IP is 192.168.0.102. 

**Figure 12: NAO Controlling Software**

Since the purpose of this project is to be able to feed an animal remotely, predefined movements had to be organized. These movements will be carried out when the User requests for food to be dispensed. In order to create these movements, the program Choregraphe was utilized. Choregraphe can be downloaded from the Aldebaran website or obtained through direct email contact with Aldebaran Support. This approach to the program was used because the Choregraphe downloaded is protected behind a login page, where only participants with a registered NAO can access. Choregraphe is a desktop application which helpls create animations and behaviors to control the NAO directly or view on a simulated robot [5]. In order to properly use Choregraphe to control the NAO, a NAOqi has to be implemented. The NAOqi is used to connect through Choregraphe to the NAO. For this project, the NAOqi goes through port 9559.

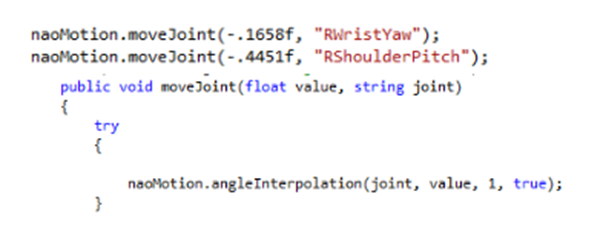


**Figure 13: Choregraphe Screenshot**

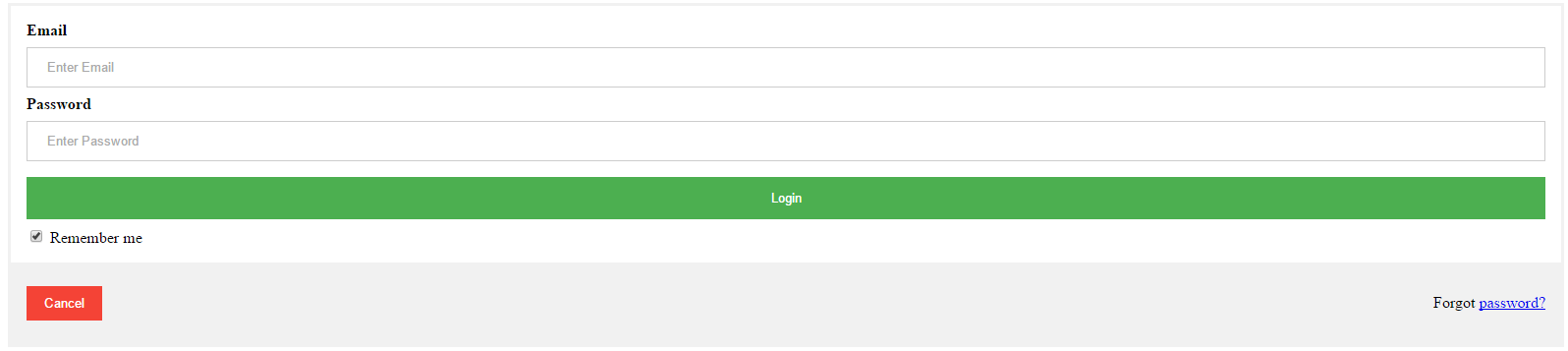
Figure 13 shows how Choregraphe can be used to program arm motion on the NAO. On the NAO the left and right arms can be moved and programed independently. Each upper appendage has 6 joint points that can be set in radians or degrees to any value in a predefined acceptable range [5]. The range of motion for each joint and the names of each joint on the right arm are shown in Figure 14. For the purpose of this project, only the right arm is used. The conglomeration of motion obtained by using Choregraphe constitutes the Motion Controller module. These movements are carried out by the NAO once it is instructed to dispense food. 

**Figure 14: Range of Motion for Right Arm of NAO**

When dealing with the actual code for the Motion Control module, the ALMotion Class [6] is imported to use methods to send data to the NAO. When datum are sent to a joint, the angle of the joint has to be set as a float in radians and the joint has to be named based on what joint it is on the robot. This is shown in Figure 15, where the method *angleInterpolation()* is being passed the specific joint and degree in radian to move the joint. Also, when dealing with joints the stiffness level has to be monitored. Depending on what is desired from the NAO the joint stiffnesses has to be lessened or enhanced. This can be done through the method *stiffnessInterpolation()* where a specific joint section can be set to zero to reduce stiffness and 0.6 to increase joint stiffness.

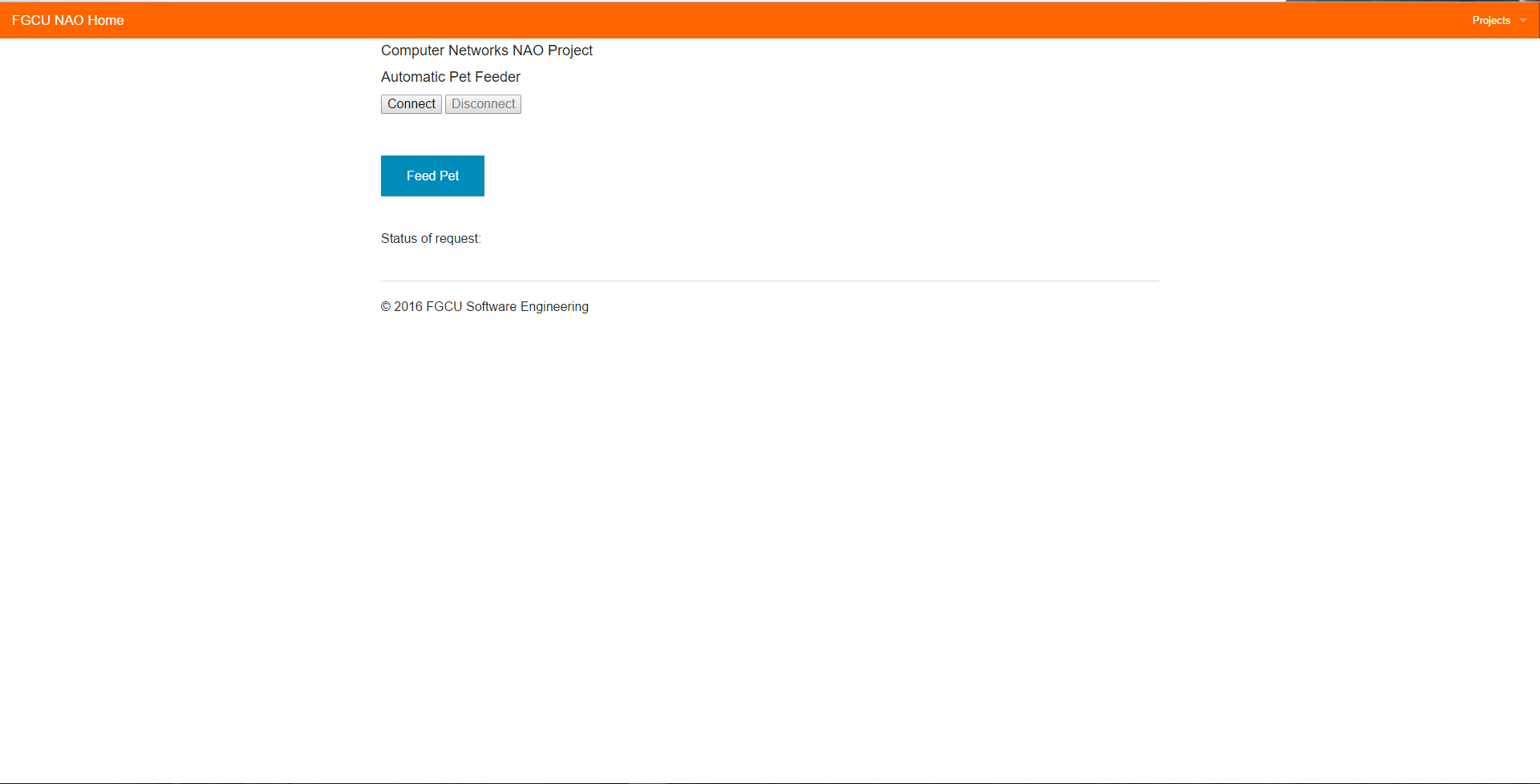
**Figure 15: Arm Motion Code Snippet**

The combination of the Motion Control module and the Connection Manager module constitute the creation of the NAO Controlling Software. It is built in C#. In order to make the NAO compatible with C# the .NET SDK has to be used. The .NET SDK comprises the NAO Robot MotionAPI module. The .NET SDK is no longer supported by Aldebaran so a third party source had to be used to acquire the software. The .NET SDK allows one to call any NAOqi method in C#, Visual Basic, or F# [5].

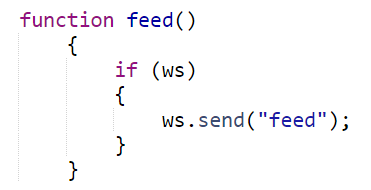


**Figure 16: Automatic Pet Feeder Webpage Login**

Shown in Figure 16 is the login page for the Automatic Pet Feeder Webpage. This sign-in option is to allow access to only authorized users. For the purpose of this project, a guest account was created to allow access. The username email is “Hello@gmail.com” and the password is “pass”. The username and password are case-sensitive. If these credentials are not properly entered, access will not be given to the NAO. Once the appropriate login information has been entered, the user will be redirected to the Automatic Pet Feeder’s main webpage. These pages are created through the conglomeration of HTML and a validate JavaScript file. .

**Figure 17: Automatic Pet Feeder Webpage**

Featured in Figure 17 is the Automatic Pet Feeder Webpage. This webpage is hosted on the ROCK Server at nao.fgcu.edu. Upon first loading the Webpage, the user will have to select “Connect”. This will send a request to the NAO Controlling Software, through the Web Socket Controller, asking if the webpage can have access to the NAO. Upon acceptance, the button becomes unselectable and the “Disconnect” button becomes an option. This webpage corresponds to the App Manager module. In this module, the User will be able to request the action “Feed Pet”. By selecting this option the request will be sent to the Web Socket Controller module, which will send the command to the NAO Controlling Software module. Upon completing the action, a message is sent back to the User on the web page indicating that it is done. This message is displayed under the “Status of request”.

**Figure 18: Send Command From Web Page Code Snippet**

Shown in Figure 18 is how the Web Socket Controller sends commands to the NAO Controlling Software. When the button is selected a string is sent to the NAO Controlling Software. The NAO Controlling Software looks out for when such a predefined string is sent. Since the command being sent is “Feed Pet” the string that is sent through the Web Socket Controller is “feed”. When “Disconnect” is selected, the connection between the NAO Controlling Software and the NAOWebApp is closed. The NAOWebApp module is the conglomeration of the App Manager module and the Web Socket Controller module.

**4.2 Testing**

The tests for the NAO Interaction between the NAO Controlling Software and the NAOWebApp are described in this section. The specific requirements tested are listed with the Test Case ID.

|  |
| --- |
| **Test Case ID:** 01, Referring to Requirement 1 of NAO Controlling Software Requirements  **Objective:** Test the functionality of the NAO Controlling Software’s ability to handle commands received from the Users of the NAOWebApp.  **Description:** Program the NAO Controlling Software to be able to handle incoming commands sent via the NAOWebApp.  **Test Conditions:** NAO Controlling Software shall be connected to the NAO and be listening for commands via a websocket.  **Expected Results:** The commands from the NAOWebApp shall be able to be received by the NAO Controlling Software.    **Result Test Case ID:** 01  **Who Ran Test:** Matthew Reilly  **Environment:** Windows Server 2012  **Pass/Fail:** Pass |

|  |
| --- |
| **Test Case ID:** 02, Referring to Requirement 2 of NAO Controlling Software Requirements  **Objective:** Test the functionality of sending commands to the NAO Robot Motion API to move upper appendages.  **Description:** Once a request has been made for the NAO to move its upper appendages, the request has to go through the NAO Robot Motion API to convert the request into something the NAO can understand.  **Test Conditions:** NAO Controlling Software shall be connected to the NAO and the NAO Robot Motion API shall be listening for commands to translate via a websocket  **Expected Results:** Upon receiving a command, the NAO will move its upper appendages.    **Result Test Case ID:** 02  **Who Ran Test:** Julia Roscher  **Environment:** Windows Server 2012  **Pass/Fail:** Pass |

|  |
| --- |
| **Test Case ID:** 03, Referring to Requirement 3 of NAO Controlling Software Requirements  **Objective:** Test the functionality of the NAO Controlling Software’s ability to send motion commands to the NAO Robot.  **Description:** Program the NAO Controlling Software to be able to send a string of motion commands to the NAO Robot in order to make the Robot move.  **Test Conditions:** NAO Controlling Software shall be connected to the NAO which will be turned on.  **Expected Results:** The commands from the NAO Controlling Software shall be able to be received and executed by the NAO Robot.    **Result Test Case ID:** 03  **Who Ran Test:** Matthew Reilly  **Environment:** Windows Server 2012  **Pass/Fail:** Pass |

|  |
| --- |
| **Test Case ID:** 04, Referring to Requirement 4 of NAO Controlling Software Requirements  **Objective:** Test the functionality of sending an error message back to the NAOWebApp if the command was interrupted or not completed.  **Description:** The NAO Controlling Software shall be able to send a message to the NAOWebApp if an error was encountered while trying to execute User request.  **Test Conditions:** The NAO shall be set-up and connected to the NAOWebApp through the NAO Controlling Software. A request shall be sent by User through the NAOWebApp but the request shall be interrupted by promptly turning off the NAO.  **Expected Results:** The NAO Controlling Software shall breakout of its process before the command can be completed and NAOWebApp shall display an error message for User to see that their request was not completed.  **Result Test Case ID:** 04  **Who Ran Test:** Julia Roscher  **Environment:** Windows Server 2012  **Pass/Fail:** Fail, communication with the NAO is one way. |

|  |
| --- |
| **Test Case ID:** 05, Referring to Requirement 5 of NAO Controlling Software Requirements  **Objective:** Test the functionality of the NAO Controlling Software’s ability to report back to the NAO Web App if food is dispersed correctly.  **Description:** Program the NAO Controlling Software to send message back to the NAOWebApp with a confirmation that the feeding process has completed.  **Test Conditions:** NAO Controlling Software shall be running, along with the NAOWebApp.  **Expected Results:** After the NAO Controlling Software has finished controlling the NAO Robot dispersing food, the Software will send a message to the NAOWebApp informing the App if the process of dispersing food has completed or not.    **Result Test Case ID:** 05  **Who Ran Test:** Matthew Reilly  **Environment:** Windows Server 2012  **Pass/Fail:** Pass |

|  |
| --- |
| **Test Case ID:** 06, Referring to Requirement 1 of NAOWebApp Requirements  **Objective:** Test the functionality of allowing access only to authorized users.  **Description:** The NAOWebApp shall be email and password protected so only authorized users can access the NAO Automatic Animal Feeder application.  **Test Conditions:** User shall navigate to the NAO Automatic Animal Feeder webpage on http://nao.fgcu.edu/. Upon trying to access the webpage the user will be prompted to enter in an email and password. The account credentials are: email “Hello@gmail.com” and password “pass”. These are case-sensitive.  **Expected Results:** Upon entering in the correct credentials the user will be welcomed to the NAO Automatic Animal Feeder main webpage. If incorrect credentials are entered, then the user will be prompted to retry.    **Result Test Case ID:** 06  **Who Ran Test:** Julia Roscher  **Environment:** Windows Server 2012  **Pass/Fail:** Pass |

|  |
| --- |
| **Test Case ID:** 07, Referring to Requirement 2 of NAOWebApp Requirements  **Objective:** Test the functionality of allowing User to decide when food is dispersed from the animal feeder.  **Description:** The NAO should only perform an action will that action is requested by the user.  **Test Conditions:** The NAO shall be turned on and connected via the NAO Controlling Software to the NAOWebApp. After the NAO has completed starting up, no command shall be sent to the NAO for 10 minutes. During this time the NAO should not move. After 10 minutes the NAO will be told a command and shall execute it.  **Expected Results:** The NAO shall sit still for the 10 minutes it is not sent a command and shall execute the command to “Feed Pet” upon command.    **Result Test Case ID:** 07  **Who Ran Test:** Matthew Reilly  **Environment:** Windows Server 2012  **Pass/Fail:** Pass. However, asking the NAO to sit still for 10 minutes was sometime proven difficult because the NAO would begin overheating and have to be turned off to cool down. |

|  |
| --- |
| **Test Case ID:** 08, Referring to Requirement 3 of NAOWebApp Requirements  **Objective:** Test the functionality of the webpage to receive and display an error message from the NAO Controlling Software if the requested action failed.  **Description:** The NAOWepApp shall be able to display an error message to the User if a requested action was not completed by the NAO.  **Test Conditions:** The User shall be connected to the NAO via the NAOWepApp and the NAO Controlling Software. Upon sending a request from the User via the NAOWepApp the NAO will be shut down, so that the command can not be completed.  **Expected Results:** It is expected that they NAOWepApp will display an error message that the requested command to “Feed Pet” was not completed.    **Result Test Case ID:** 08  **Who Ran Test:** Julia Roscher  **Environment:** Windows Server 2012  **Pass/Fail:** Fail. There is only one way communication setup with the NAO. |

|  |
| --- |
| **Test Case ID:** 09, Referring to Requirement 4 of NAOWebApp Requirements  **Objective:** Test the functionality of the NAOWebApp’s ability to activate the NAO Controlling Softwares motion commands to disperse food.  **Description:** Program the NAOWebApp to be able to communicate with the NAO Controlling Software in order to send an activation command in order to run motions to disperse food.  **Test Conditions:** NAOWebApp shall be running on the rock server and the NAO Controlling Software must be connected to the NAO Robot.  **Expected Results:** The commands from the NAOWebApp shall be able to be taken in by the NAO Controlling Software to run motion commands.    **Result Test Case ID:** 09  **Who Ran Test:** Julia Roscher  **Environment:** Windows Server 2012  **Pass/Fail:** Pass |

|  |
| --- |
| **Test Case ID:** 10, Referring to Requirement 5 of NAOWebApp Requirements  **Objective:** Test the functionality of the NAOWebApp’s ability connect to the same router that the NAO Robot is on.  **Description:** Program the NAOWebApp to be able to connect to the same socket and IP address that the NAO Robot is on.  **Test Conditions:** NAOWebApp shall be running on the rock server as a live page, and the NAO Robot shall be turned on.  **Expected Results:** When the User hits the connect button on the NAOWebApp the Web App will connect to the NAO Robot on its same IP and port.    **Result Test Case ID:** 10  **Who Ran Test:** Matthew Reilly  **Environment:** Windows Server 2012  **Pass/Fail:** Pass |

|  |
| --- |
| **Test Case ID:** 11, Referring to Requirement 1 of Design Constraints  **Objective:** Test the functionality of the NAO Robot's ability to move its upper appendages.  **Description:** The NAO Robot should have the ability to move its upper appendages without aid of the User.  **Test Conditions:** Using Choregraphe the NAO Robot will be told to move its upper arms by moving to “StandZero”. This motion is a basic motion predefined by Choregraphe.  **Expected Results:** The NAO Robot shall move its upper arms upon being told by Choregraphe.    **Result Test Case ID:** 11  **Who Ran Test:** Julia Roscher  **Environment:** Windows Server 2012 and Choregraphe  **Pass/Fail:** Pass |

|  |
| --- |
| **Test Case ID:** 12, Referring to Requirement 2 of Design Constraints  **Objective:** Test the dexterity of the NAO Robot’s upper appendages.  **Description:** The NAO Robot shall be able to move its upper appendages to their full range of motion. This is to be done by manually moving the NAO’s upper appendages.  **Test Conditions:** Manually take the NAO Robot’s upper appendages and move them in all directions. Ensuring that the upper appendages can move to desired position.  **Expected Results:** The upper appendages of the NAO Robot shall be able to be manually moved as desired.    **Result Test Case ID:** 12  **Who Ran Test:** Julia Roscher  **Environment:**  **Pass/Fail:** Pass |

|  |
| --- |
| **Test Case ID:** 13, Referring to Requirement 3 of Design Constraints  **Objective:** Test the ability to connect to the Internet via an Ethernet Cable.  **Description:** The robot will be connected up to the the Internet via an Ethernet Cable and if the robot can interact via the NAO Controlling Software, the robot is successfully connected.  **Test Conditions:** Using the NAO controlling software on the Rock Server, the robot will be connected to and told to do certain movements.  **Expected Results:** The NAO Robot will be able to perform movements sent to it from the NAO Controlling Software, and communicate with the NAO Controlling Software    **Result Test Case ID:** 13  **Who Ran Test:** Matthew Reilly  **Environment:** Windows Server 2012  **Pass/Fail:** Pass |

|  |
| --- |
| **Test Case ID:** 14, Referring to Requirement 4 of Design Constraints  **Objective:** Test the NAO Robot’s ability to lift objects of a certain weight.  **Description:** The NAO Robot shall be able to lift up certain objects in order to perform certain tasks, without the motor’s failing.  **Test Conditions:** The NAO Robot with be connected to Choregraphe and sent movements in order to lift objects.  **Expected Results:** The NAO Robot shall be able to lift objects that are desired in order to perform the task of feeding a pet.    **Result Test Case ID:** 14  **Who Ran Test:** Julia Roscher  **Environment:** Windows Server 2012 and Choregraphe  **Pass/Fail:** Pass |

|  |
| --- |
| **Test Case ID:** 15, Referring to Requirement 5 of Design Constraints  **Objective:** Test that the battery power level of the NAO is full.  **Description:** Test of the ability of the NAO to be able to be on at all times, in order to perform tasks at all times.  **Test Conditions:** The NAO shall be connected up via a power cord to the wall, and be able to be in an on position.  **Expected Results:** The NAO will always be charged as the charging cable will always be plugged in. This can be confirmed by tapping the power button of the NAO after it is turned in. By tapping the power button the NAO will give a verbal feedback about its battery status.    **Result Test Case ID:** 15  **Who Ran Test:** Matthew Reilly  **Environment:** Windows Server 2012  **Pass/Fail:** Pass |

**5. Conclusion**

**5.1 Summary**

The completed project accomplished the goal of using the NAO to remotely dispense food from a web application. Upon setting up the components of the NAO Automatic Animal Feeder, the user simply has to successfully login to the Automatic Animal Feeder webpage and select the “Feed Pet” button. By selecting the button prompts the NAO to perform the actions necessary to dispense the food.

**5.2 Difficulties**

During the course of the project difficulties and problems arose that had to be resolved. These problems ranged from basic functionality to esthetics.

One of the biggest problems encountered was the issue of connecting to the NAO over Florida Gulf Coast University’s network. This issue was resolved through analyzing of past projects documentation. Through this investigation, it was learned which port and IP address were appropriate to use when communicating with the NAO and that websockets were used to enable communication between the NAO and the Server. It was also learned that direct communication with the NAO through the .NET API could only be done when logged into the Administrator account of the Server computer located in Lab 429.

Another difficulty that surfaced, was the conversion of the arm motions created in Choregraphe to the Motion Control module application written in C#. Creating arm motion in Choregraphe was not the difficult part, that part came later when it came time to export those movements to the main application. When exporting motion through Choregraphe only C++ or Python are exportable formats. So the problem arose in trying to convert the Python code into C# code that could be understood by the .NET API of the NAO. A lot of digging had to be done on the Aldebaran website to locate the documentation needed.

Another problem, was setting up the login webpage so that only authorized users could use the NAO Automatic Feeder Webpage. At first the authorization was implemented via PHP but the method post() was not functioning as intended and information was not being transferred properly between the client side HTML and the server side PHP. This was fixed by implementing Javascript instead of PHP.

Lastly, an issue faced was that of having the NAO send an error message back to the user on the Automatic Animal Feeder webpage if an error occurred. This was difficult since the NAO would have to push a notification to the webpage and the webpage would then have to know how to display the message. It is still unknown how to best enable notification to be sent by the NAO to a webpage when using websockets.

**5.3 Further Developments**

Future developments could be completed to further enhance the Automatic Animal Feeder. These enhancements would increase the usability of the project. One such idea, would be to implement a database and timer system. That way the user could set the NAO to dispense food twice a day at a predetermined time. It would also be useful, to enable the video feature of the NAO. This could be used by the user to view their pet and confirm that the NAO completed the requested action. Another useful feature, would be to equip the NAO with various measuring capabilities. That way the user could request for the NAO to dispense ½ of food at one point and a cup at a later point. In order to enable this feature, it would be useful to implement NAO Learning so that they NAO can learn where the tools it require are instead of implemented predefined motions. Another feature, could be to be program the NAO to not only dispense food but also water. This way the basic care of the pet would be hands off for the User. Lastly, enabling the NAO to push notifications back to the webpage would be useful so that the user can know when a requested action was not completed by the NAO.

**6. References**

[1] SoftBank Robotics, Find out more about NAO, SoftBank Robotics, September 27, 2016, https://www.ald.softbankrobotics.com/en/cool-robots/nao/find-out-more-about-nao

[2] Gafford, S, Gibson, L, *Robotics for Online Learning.* Florida Gulf Coast University, April 30, 2016

[3] A. Hughes, *Instructions for use of ROCK Server*. Florida Gulf Coast University, Febuary 11, 2014

[4] A. Hughes, *Instructions of use of the NAO Robot Websocket Server*. Florida Gulf Coast University, March 9, 2014

[5] Aldebaran Robotics, NAO Software 1.14.5 Documentation. Aldebaran, November 9, 2016, http://doc.aldebaran.com/1-14/index.html

[6] Aldebaran Robotics, Class ALMotion, Aldebaran, November 9, 2016,

http://doc.aldebaran.com/java/2-1/com/aldebaran/qi/helper/proxies/ALMotion.html#angleInterpolation-java.lang.Object-java.lang.Object-java.lang.Object-java.lang.Boolean-