



Final Year Report

Human-Robot Interaction for
Older Adults

Randell Quizon



Table of Contents

Introductions	2
Background	3
Approach	6
Design a Human-Robot Interaction for Older Adults.....	6
Initial Plan	6
Work Plan	7
Functional and Non-functional Requirements	7
Functional Requirements	7
Non-functional Requirements	8
Reason Why I Used Functional and Non-functional Requirements	8
NaoQi Pepper	9
Implementation.....	9
Choregraphe	9
Python 2.7 SDK	10
QiChat	10
Chat Dialogue Box	12
Joke Dialogue Box	13
Games Dialogue Box	13
Results and Evaluation	14
Results.....	14
Heuristic Evaluation	15
Evaluation	16
Future Work.....	17
Conclusions	18
Reflection on Learning	19
References	21

Introductions

The number of older adults has continuously increased over the years because of the increasing rate of world population ageing. It is said that the current percentage of older adults will almost double in the next few decades (WHO: News Room: Fact Sheets: Detail: Ageing and Health, 2021). In our modern society, there are a few challenges that older adults face. For example, along with the rapidly flowing stream of time is new trends and knowledge, with it the feeling of being left behind by the generation is what many older adults feel, which can make them feel lonely and anxious. Along with ageing comes the problem of having difficulty in moving for some older adults as well, which leads to them usually wanting a nice little chat to with other people to entertain themselves as well as relieve any loneliness they feel. Some of us must have experienced our grandparents always wanting to have a conversation with us and they could keep on talking forever. However, most of the young children or adults are still active, and while we may want to catch up with the older adults, we are not always going to be available to sit down with them. Our research and project are aiming to solve the problems that these older adults are facing and decided to achieve it with human-robot interaction. I am pretty sure that older adults understand this as well because they were young once as well. The reason for this why we chose robot to solve the problems is because robots will always be available whenever older adults has the need and desire to use them. There are a lot of questions pertaining to robots though. How can human-robot interaction help solve these challenges or at least support older adults through these challenges? What are the advantages and disadvantages of human-robot interaction compared to similar technologies like a chatbot for example? What are the impacts of robots on society and are there any potential consequences of adapting robots to society? Are there any rules that robots must follow to remain ethical in societal standards? What kind of situations are robots useful to older adults? In this project, we will look at the problem of how we can use human-robot interactions to support humans, specifically focused on older adults.

Some robots, like NaoQi Pepper for example, has the ability to have a proper conversation with a human so we decided on designing a human-robot interaction by designing and implementing a robot that can communicate via interactive speech with users, which are older adults. We are aiming for the designed human-robot interaction to solve the problem of loneliness and anxiousness that most older adults feel in the modern days. The robot that we are going to be using is called 'Pepper' from SoftBank Robotics Developer Center. Pepper is a humanoid robot that can sense a human's actions once they are in its range via seeing or hearing. Pepper can respond back via speech or gesture animation, for example, if a user says 'hello', Pepper can respond with a 'hi' while waving at the same time. We will be mainly focusing on developing the speech animation area so that Pepper can manage a conversation with older adults. This means that we will be developing Pepper's skill in communication. A communication skill and other features is prioritised to suit older adults, however at the same time, these skills and features will be flexible outside the range of older adults. The development environment that we decided to use is mainly Choregraphe 2.5 and Python 2.7 with a Python SDK. We will be using NaoQi APIs that are built-in in Choregraphe but can also be installed if we plan to use Python outside of Choregraphe. QiChat is a part of a NaoQi Interaction Engine for dialogues, which will probably be the most used API since we will be creating dialogues between human and robots.

Now, how are we planning to solve these questions by designing a human-robot interaction using NaoQi? After realising and identifying some challenges those older adults are experiencing in society, we listed down three categories that Pepper will be doing for its interaction with older adults. These are chatting, joking, and playing games with the users. We believe these three categories can and will support older adults through these challenges, such as alleviating their loneliness and anxiousness with the robot entertaining them, as well as being able to sit down and have a nice little chat with the robot which as mentioned, one that they usually want to do since most of them have difficulty in moving around. Now, speaking of human-robot interaction, even

with all these features that were mentioned, it is hard to imagine robots to serve as a substitute for humans in supporting older adults. The main reason why it is very hard to imagine is because humans' judgement mostly stems from emotions instead of logic, which is the complete opposite of robots who solely relies on logic for their judgement. Pepper, however, has functions and features that allow them to exhibit affective empathy, which is a type of empathy that means the understanding of another person's emotions and responding back appropriately. These features and functions that exhibits affective empathy is demonstrated by imitating how humans respond when they are conversating with someone and understanding that person's emotions.

Using Choregraphe, especially QiChat, we will be designing and implementing the mentioned functions and features to NaoQi Pepper. Chatting with older adults will mainly consist of small talks or dialogues between the users and the robot. Pepper will respond to the users' questions, idle chat, and requests with a speech, gesture, or both. The conversation between the user and Pepper will have different subjects or topics that they can talk about, with each topic being connected to allow transition between them to create a conversation flow. Pepper will also be implemented with the ability to stick and continue the current conversation by asking appropriate questions that depends on the current state and flow of the conversation topic. Telling jokes function is a lot simpler because there is really no need to transition between topics. We will implement a trigger for Pepper for them to tell a joke. Users can trigger the robot to tell a joke by saying a phrase or request asking Pepper for a joke and it will say a random punchline among the list of punchlines that were implemented to Pepper. Playing games is the most complicated out of all of this in our opinion because of the need to think of what games can be played between older adults and Pepper. We decided on doing a quiz and a rock, paper, scissors game. Another reason why we found this to be complicated to implement is because of how we have to consider different outcomes and timings in the game.

Background

NaoQi Pepper, the robot that we are going to be using to solve the problem we are looking at, is a humanoid robot that as the ability to perceive human emotions by detecting and analysing human facial expressions. NaoQi Pepper is owned by SoftBank which is founded by Masayoshi Son and was introduced in Tokyo back on 5th of June 2014. NaoQi Pepper has been used and proven useful in a variety of ways, for example, NaoQi Pepper was installed and used as a receptionist in London, UK at Brainlabs. In Japan, NaoQi Pepper is used at banks and medical facilities. In sports events, a team of Pepper was said to perform as cheerleading squad at baseball game on 9th of July 2020. NaoQi Pepper is also used a lot for research and academic purposes, usually for students who studies computing or programming or any related courses to these. NaoQi Pepper is usually the robot used when doing research about human-robot interactions, which is a similar thing that we are doing right now since we are trying to solve the problem of how human-robot interactions can support older adults using NaoQi Pepper as the robot. I have seen a similar research to what I am conducting by an international team back in 2017 to develop Pepper to take care of older adults. It is said that the main purpose why NaoQi Pepper was developed is to make people enjoy life by supporting them to improve their lives, smoothening relationships, and forming connections with other people.

(Wikipedia: Wiki, n.d.)

There are researches out there that has been conducted already which really helped us with our research and project. One example of those research is the research titled Pepper Integration with Diagflow which is written by Richard on 11th of February 2019. He discussed about how NaoQi Pepper's face detection and recognition to be flawed, including the speech detection and recognition of Pepper. In one of his examples of NaoQi Pepper's flaw in this area is that it cannot differentiate between the word 'Bob' and 'Paul', as he tried to make NaoQi Pepper say the former

but instead resulting in saying the latter. Another one of the flaws that he mentioned about Pepper is Pepper's ability to learn by itself without the need for the developer to note down the problem and program the improvement to Pepper. Richard's hypothesis was that integrating Diagflow to Pepper will solve the mentioned problems, like improved learning ability, improved word recognition, and improved sensitivity to noise. He developed a code using Python to that integrates Diagflow. The conclusion of his hypothesis seems satisfying because he mentioned several advantages that resulted from integrating Diagflow, which were better understanding of human speech and improved Pepper as a technology. We took inspiration from his work by learning what he mentioned about Pepper's potential flaws. However, unlike his work, we did not integrate Diagflow to Pepper. Instead, I stuck with using the QiChat API of Choregraphe to code Pepper. We also have a Python SDK installed for Choregraphe just in case we need it but instead of integrating Diagflow to solve the potential flaws of Pepper, we worked around QiChat instead, like editing the command in order as well as adjusting tones and volumes for Pepper. We also added an error-catching function for Pepper so that any mishear or misunderstanding will be caught by Pepper.

(Richard, 2019)

Another one of the researches or conducted studies by other people that really inspired us with our project is the study about 'Mobile, Socially Assistive Robots Incorporating Approach Behaviour: Requirements for Successful Dialogue with Dementia Patients in a Nursing Home' by Mio Nakamura, Kohei Ikada, Kazuki Kawamura, and Misato Nihei. The purpose of their study was to find out how effective is using a mobile socially assistive robot that integrates approach behaviour for older adults. The socially assistive robot that they used in conducting this study is the same robot that I am using for my project, which is SoftBank Robotics' NaoQi Pepper. Two of NaoQi Pepper's functions that were mentioned in their method were teleoperated mobility functions and teleoperated conversation functions. They designed an approach behaviour, or APB for short, using mobile socially assistive robots in a nursing home. The method of experimentation that they used to design the approach behaviour is by gathering a certain number of participants then verifying the effect of mobile socially assistive robot and finally extracting the requirements that lead to dialogue with seniors. The result of these experiments was demonstrated into three parts which were the comparison of dialogue frequency between stationary and mobile socially assistive robots, relationship between frequency of dialogue and MMSE scores, and analysis of participants' behaviour during approach behaviour when the dialogue was successful. In summary, the study that they conducted to find out how effective using a mobile socially assistive robot is, made of use of experiments and comparisons. The only part that was similar to our project was that they used NaoQi Pepper as the robot to be used in the project, as well as older adults being the primary target. One of the differences between their work and our work is that we were looking more on older adults in general, like in a normal household, etc., while they were looking more on older adults in nursing home. Another one of the differences between us is our experimentation methods. They invited participants to their experiments and had an actual experiment while we, on the other hand, had not used an actual experiment. The main reason for this is time because we did not have enough time to have an ethical report approved in time for the experiment. Instead, we used a heuristic evaluation to produces results and evaluate them properly. It may not be as accurate as an actual experiment but we trust that the results of using a heuristic evaluation would be satisfying and precise enough to be reliable. Another difference between our project would be us not incorporating approach behaviour in our project. We merely decided on the type of activities that older adults would like to take part in during conversations, with the thought that most of the times, older adults will be the one initiating the conversation with NaoQi Pepper.

(Nakamura, Ikeda, Kawamura, & Nihei, 2021)

There is also another work that gave us more useful knowledge when doing my project. This work is a book titled *Social Robotics* by Shuzhi Sam Ge, John-john Cabibihan, Miguel A. Salichs, Elizabeth Broadbent, Hongsheng He, Alan R. Wagner, and Alvaro Castro-Gonzalez (Eds.). The book consists of different articles about social robotics. The specific article in this book that I focused on is the article titled 'Dialogue Models for Socially Intelligent Robots' by Kristiina Jokinen. In this specific article that was mentioned, they conducted a study about dialogue modelling to allow interactions using natural language between users and robots. They said that 'dialogue interactions with social robots are "situated"', which means that interactions between users and social robots are not and cannot be specified in advance as the interaction happens depending in a real-world situation. They also mentioned that robots, unlike other technologies, are considered as intelligent agents instead of just tools to manipulate to reach an endgoal. In order to create their dialogue model, they identified a social robot's characteristic features as well as macro and micro environments to naturally interact with users. Along with this, they also included the different capabilities of humans and robots and put them side-by-side together using a table. With the previous features identified, they were ready to start creating they started by designing different models and architectures first. The first one being a dialogue management model which purpose is to empower the robot's communicative behaviour by considering the mentioned requirement about the different capabilities of humans and robots. Next, they designed the system architecture by designing a cognitive dialogue model, which includes communication enablements. Then next, they moved to designing the model for dialogue coherence and topic, which purpose is to allow fluent and smooth transition from one topic to another during dialogues. Next, they considered the multimodal interactions, which basically means non-verbal pauses during conversations with other people. Their next step was interaction and knowledge learning which is about the deep learning of the robot and how it can manage the knowledge coming from the users. Lastly, they designed an application and interaction model, which refers to care-giving knowledge. For more detailed information for each models, read the book which is citated at the end of this paragraph. It is referenced to in the reference section. As a summary of what they have done, they basically focused on identifying the basic requirements for spoken dialogue models to allow natural interactions between humans and social robots. Now, we have finished mentioning what they have done for their study, we are now going to discuss what we have done differently from them and what we took as an inspiration from them. One of the inspirations we took from their study was their discussion about human-robot dialogue interaction. The reason this part inspired us was because the main feature of our proposed solution to the problem is NaoQi Pepper having conversations with older adults. This shows mostly in the form of the term 'small talks', with us implementing several topics. Speaking of topics, another one of our inspirations from this project was their work on dialogue coherence and topic modelling because we had to think hard how the topics can be changed during conversations because a conversation does not usually stick in one topic. We took inspiration of how each topics can relate to each other in order to smoothly transition between each of them. Another one of the inspirations we took were their discussion about the use of multimodal signal, which basically means non-verbal interaction with the users. For our own project, we had to think about the natural pauses during conversations, including the length of time that it takes to consider a pause in conversation to be awkward. The biggest difference of their project from our project is their creation of dialogue models. Instead of designing dialogue models, we tried to understand instead how conversations in chatbot works. We chose different topics that are viable to talk with older adults, while preparing questions for the users by the robots, while predicting possible answers from the users.

(Ge, et al., 2018) (Jokinen, 2018)

Approach

Design a Human-Robot Interaction for Older Adults

The approach we took for the project of solving the problems about older adults that we mentioned in the introduction, like loneliness, anxiousness, feeling of getting left behind, etc. is to design a human-robot interaction for older adults. We decided on designing and implementing a robot that can assist older adults socially. This means any form of socialising from chatting, telling jokes, and playing games. The socially assistive robot that we decided to use is SoftBank Robotics' NaoQi Pepper. One of the reasons of why we decided on designing and implementing a human-robot interaction for older adults as the solution to solve some of the emotional and mental problems of older adults, like loneliness, anxiousness, etc. is because socially assistive robots like NaoQi Pepper can help humans experience a more interesting and enthusiastic lives. We could ask why not design a technology similar to 'Alexa', 'Safari', etc. instead of a robot? The reason for this is because unlike the mentioned technologies, an advantage of using a robot is that it is visually appealing to the human users because it can take any form or shapes. For example, it can be in a shape of a dog, it can be in shape of a bin, it can be in a shape of just hands, it can be in a humanoid shape like NaoQi Pepper, etc. Together with its visually appealing features to the human users, another advantage of designing a human-robot interaction is that it is more engaging compared to other artificial intelligence technologies like 'Alexa', 'Safari', etc. The reason that we believe that robots are more engaging to the human users is because they have the ability to replicate human actions during conversations, which we can call non-verbal communication or interaction, like nodding, pauses, waving, and other body languages that can significantly improve one's social skills even for humans. These features make it more immersive to the human users when they are having a conversation with the robots.

Initial Plan

The first approach I have put into practice when starting our final year project is to create an initial plan. The initial plan includes deciding and writing of our project description, project aims and objectives, and a work plan. The project description is almost similar to the introduction of this final year project report. We basically talked about the problems that we mentioned in the introductions, however, we discussed them in the form of rhetorical questions instead of explaining them on details. For example, questions like, 'how can robots support older adults?'. We also discussed there are potential solutions or features of the solutions to the problems of older adults that we mentioned. For example, reducing loneliness by the robot playing games or telling jokes with the older adults. For the project aims and objectives, we basically just divided the solution, which is 'designing a human-robot interaction for older adults' into individual step-by-step tasks. For example, before we could actually design a human-robot interaction for older adults, we need to understand more about socially assistive robots and older adults themselves, hence the first task was to understand how robots can support older adults' everyday tasks. For the work plan, in summary of what we have written in the work plan, it will be the tasks that we need to complete in the range of a week or more. We made a smart guess of how much time should be put in each task and assigned a range of weeks to finish them. For example, from week 5 to week 10, we should have already submitted an initial draft, start implementation progress of the project, meet with supervisors, and cross out every requirement that are completed. An advantage of writing an initial plan is that we get to have a clear general idea of what we need to complete for the project. For example, we have a clear idea of the problem and the solution we want as well as each task and the order and time it needs to be completed. A small disadvantage of writing an initial plan is that we write it at the start of the project so we can only make smart guesses about the plans that we are

writing for the project. Unexpected problems can occur in the middle of designing and implementing a human-robot interaction, which could lead to some plans not being followed, being changed, and can even be just removed from the whole plan.

Work Plan

Creating a work plan for our project was one of our approaches that we did to start our final year project about designing a human-robot interaction for older adults. As mentioned in the previous section about the initial plan, we basically wrote a milestones plan for our final year project. In order to follow the work plan, we needed to complete certain tasks for each week or range of week. These tasks came from our main objective being divided into smaller tasks. Completing all of these small tasks will eventually allow us to complete the final year project about designing a human-robot interaction. One of the advantages of having a work plan is that it allows us to be conscious of the time for each week, which results in our time management being better. The reason for this is because having a deadline for each week for certain tasks, prevents us from cramming or condensing all these tasks in the last minute or procrastination in other words. Another one of the advantages of creating and using a work plan is that it can be very flexible. If something unexpected happens, we can, most of the time, easily adjust the tasks to other weeks. A perfect example would be around week 3 or 4, I tested positive for CoVid and made me feel really unwell. This caused me unable to complete my tasks during those weeks, so I adjusted the work plan so that I could finish them in the following week, while at the same time, completing majority of my supposed task for that week, if not completing them all, and complete the remaining bits the week after that along with the tasks for that week. However, you could also consider this advantage as a disadvantage because it means that the work plan cannot be 100% accurate all the time. We wrote the work plan at the start of the final year project, which means that most of what we have written are smart guesses and estimation without actual certainties for it so it is not 100% reliable that the work plan can be followed perfectly.

Functional and Non-functional Requirements

Functional Requirements

- Chatting
 - User greets Pepper with 'hi', 'hello', 'hey', etc.
 - Pepper asks 'How are you?'
 - If user says 'good', Pepper responds 'Nice I am doing well too'. > Move to next topic
 - If user says 'bad', Pepper responds 'Sorry to hear that. I can help cheer you up if you want. We could talk about the food, books, or pets.'
 - If user says books > move to books topic
 - If user says food > move to food topic
 - If user says pets > move to pets topic
 - Pepper asks 'Do we have a nice weather today?'
 - If user says 'yes', then Pepper responds 'Oh, it might be a great time to go for a walk.'
 - If user says 'no, then Pepper responds 'Oh that's too bad. Let's change the subject.'
 - Pepper asks 'What is your favourite food?'
 - User says their favourite food and Pepper responds 'That sounds delicious.'
 - Pepper asks 'Do you like to read books?'

- If user says 'yes', then Pepper responds 'That's great. What genres of books do you read?'
 - User says book genre, Pepper responds 'Interesting'
 - If user says 'no', then Pepper responds 'Oh okay. Let's change the topic then.'
 - Pepper asks 'Do you have a pet?'
 - If user says 'yes', then Pepper responds 'Wonderful. What pet do you have?'
 - User says pets, Pepper responds 'Oh very nice.'
 - If user says 'no', then Pepper responds 'Oh okay. If you would have one, what would it be?'
 - User says pets, Pepper responds 'Interesting choice.'
 - User asks, 'What is your name?', Pepper responds 'My name is Pepper. How are you?'
 - User asks, 'What is your favourite food?', Pepper responds 'I like rice and beef.'
 - User asks, 'Do you like to read books?', Pepper responds 'I can not read yet.'
 - User asks, 'What pet would you like?', Pepper responds 'It would be a dog for me.'
 - User says goodbye, Pepper responds with goodbye and shuts off.
 - If Pepper can not understand dialog, then Pepper says 'Sorry I do not understand that word.' or 'Sorry, please answer yes or no.'
 - If user does not respond, then Pepper says 'Hello?'
- Playing with them
 - Robot plays a quiz game by asking questions to the user
 - Robot plays rock, paper, scissors with the user
- Telling Jokes
 - User asks Pepper to tell a joke, then Pepper responds with a random joke.
 - Robot says 'Why aren't koalas actual bears? They don't meet the koalifications.'
 - Robot says 'What do Alexander the Great and Winnie the Pooh have in common? They have the same middle name.'
 - Robot says 'What do you call bears with no ears? B-'

Non-functional Requirements

- Appropriate gestures/movements when talking
- Clear words and volume
- Ensure no words can be taken as offensive
- Response time by robot (not sure if I can affect this or will this depend on the robot machine itself)
- Flexible speech function (speech functions can be used for many scenarios)

Reason Why I Used Functional and Non-functional Requirements

Functional requirements are requirements or specifications that you intend to implement into your system that will be the main purpose of the system's functions or features. It is the main purpose or reason for why users use the system. For example, a user turns on Pepper for the reasons of chatting with them, playing with them, or wanting to have a laugh. Non-functional requirements on the other hand are functions of the system, which does not necessarily mean unnecessary, but it is not the main purpose of the system's functions or features. It is also not the main purpose or reason for why users use the system. Turning on Pepper is a function of Pepper, but user does not start Pepper for the purpose of starting Pepper, nor do they start Pepper for the sake of turning Pepper off. One of

the reasons why we identified and wrote down our functional and non-functional requirements before implementation is because one of the advantages of writing them is that we are able to agree on the requirements and think carefully about the purpose of the system. It gives us a clear thought of the requirements that are needed for the system's purpose. We also get a clear view of requirements that are not necessarily the purpose of the system but are important for the system to have. Another one of the advantages of using a functional and non-functional requirement is that it adds limitations to the system. Without any limitations, the system could end up having and performing unnecessary processes, features, and functions, which could lead to users confusion, especially for this project, where older adults are the primary target and might have very little knowledge about technology.

NaoQi Pepper

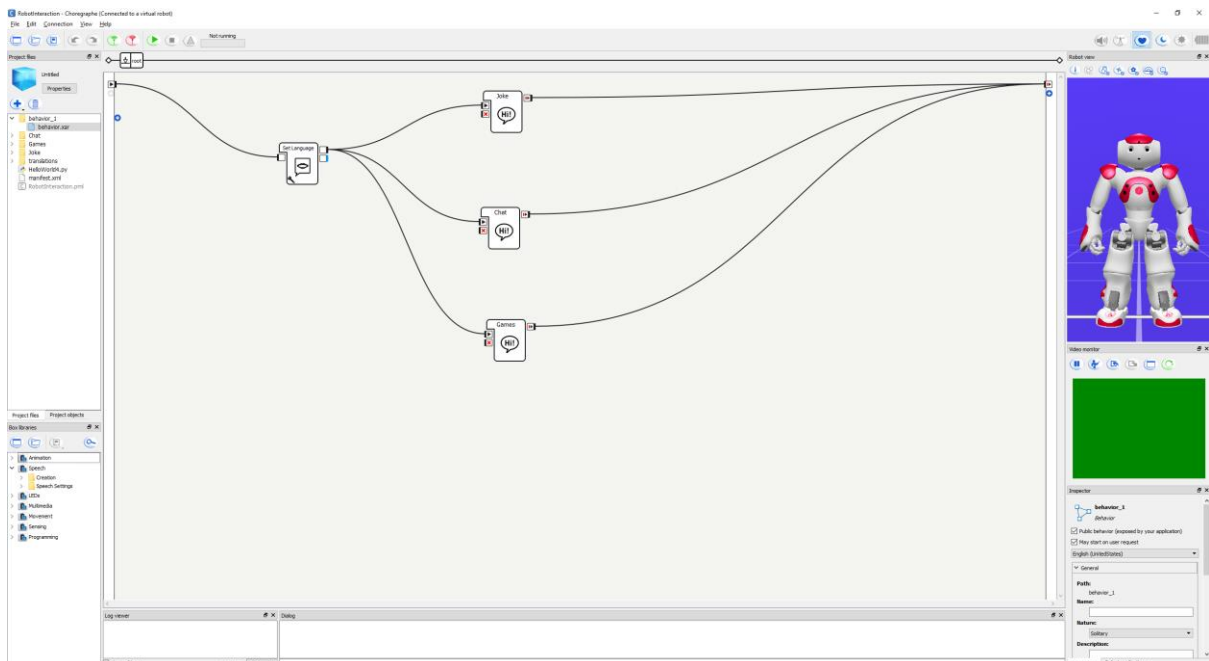
We decided on using NaoQi Pepper as the socially assistive robot that we are going to be using to approach the designing and implementing of human-robot interaction for older adults. NaoQi Pepper is a socially assistive robot that was developed by Softbank Robotics in Tokyo, Japan on 2014. A humanoid robot is the type of robot that NaoQi Pepper falls under. The version of NaoQi Pepper that we are using right now is version 2.5. NaoQi Pepper's dimensions while arms down in the current version is 1210 mm in height, 480 mm in width, and 425 mm in depth. When NaoQi Pepper's arms are spread out, the dimensions become 1355 mm in height, 1196.9 mm in width, and 648.2 mm in depth. NaoQi Pepper's motherboard specification in the current version has an ATOM E3845 for a processor, a quad core CPU, a 1.91 GHz clock speed, and a 4 GB DDR3 RAM. The NaoQi Pepper that I am going to be using is not owned by me but owned by Cardiff University. Most of the time during the design and implementation of the human-robot interaction for older adults, we have to use a virtual robot to test it. One of the main reasons why we decided to use NaoQi Pepper is that one of NaoQi Pepper's advantages is that it can perform expressive movements while talking to the human users. For example, NaoQi Pepper can wave while saying hi or even bow when apologising. Another one of the advantages of using NaoQi Pepper is that it can recognise speech quite well and the languages can be adjusted so it can recognise speeches from 15 languages at most. NaoQi Pepper is also accessible and programmable for a student like us, which is one of the advantages and reasons of why we chose it to be the robot that we are going to use for our final year project.

Implementation

Choregraphe

Choregraphe is the software that we decided in using to develop and implement code to NaoQi Pepper. Choregraphe is a graphical tool that is developed by SoftBank Robotics, which allows us to implement and develop the humanoid socially assistive robot called NaoQi Pepper. Choregraphe allows us to develop NaoQi Pepper's software in a graphical way using boxes, like dialogue boxes for creating dialogues, python boxes for writing python scripts, diagram boxes, and timeline boxes. Aside from the mentioned boxes, it also has a lot more boxes to offer, like animation boxes, speech boxes, movement boxes, etc. These boxes allow us designers and developers to graphically implement animations, behaviours, and dialogues to NaoQi Pepper. The boxes that we have mostly used during the implementation was the 'Set Language' box from 'Speech' boxes category and dialogue boxes. We have the 'Set Language' box to set the language of the robot to English. We used three dialogue boxes with different topic names which are 'Chat', 'Joke', and 'Games'. Each of the dialogue box we implemented the dialogues that the human users and NaoQi Pepper will say during conversations, where the dialogues are related to the topic of each box. Choregraphe also has a virtual robot for simulation testing. Since we do not have our own NaoQi Pepper and the only one we can use is the Cardiff University's NaoQi Pepper, we have been using the virtual robot feature

inside Choregraphe to do a simulation test of our implemented code for NaoQi Pepper. While we have not used it much during implementation, Choregraphe does accept Python scripts to further implement a more advanced behaviour for NaoQi Pepper, which can be complicated with QiChat alone.



Python 2.7 SDK

We have not really found the need to write a script outside of Choregraphe for implementation of speech and behaviour of NaoQi Pepper. All of our codes are all written in Choregraphe itself using QiChat. We did install the Python 2.7 SDK because we thought it will be useful for the project or future improvements with the implementation. Boxes like 'Python' box or 'Set Language' box all looks like they contain Python scripts within them, so even if we are not really writing Python scripts outside of Choregraphe, we feel like installing the Python 2.7 SDK is very important and needed because of the reason of some of the boxes being constructed with Python scripts. We did have to be very careful during the installation of the Python 2.7 SDK because our personal computer already has Python 3 installed and in PATH as well, so we have to ensure that we can add Python 2.7 without replacing Python 3. One of the advantages of using a Python 2.7 SDK is that it is easier if you compare it Java, even if it is slightly more complicated than QiChat. Another one of the advantages of using a Python 2.7 SDK is that it allows us developers to implement Python modules for the robot and run it outside of Choregraphe. When writing a Python script using the Python 2.7 SDK, importing ALProxy from NaoQi is the first thing that needs to be done. The developer also needs to identify the IP address of the robot because the ALProxy will need it for its parameters.

QiChat

QiChat is the script language that we have been using to write Dialogue Topic boxes. All of our coding is done in Choregraphe itself because we have only been using Dialogue Topic boxes to construct and implement the dialogue between the human users and NaoQi Pepper. For what we have done for our implementation of the speech and behaviour of the robot, we have been using dialogue boxes for entirety of the implementation. This is also how we start a QiChat script, by adding a dialogue box and opening its file with the 'enu.top' format, which opens the QiChat script window. Every time we create or add a new dialogue box and open it, there is always a default script that appears which includes the two starting lines of a QiChat script, which shows the topic name of

the dialogue box and the language, which is set to 'enu' by default. Writing a QiChat script is different from writing a Python script. A QiChat script feels simpler and easier to write in our opinion. When we are writing the QiChat script for our final year project, we write it by adding rules using QiChat Syntaxes. A rule in a QiChat script usually needs an input from a human user and an output or response from the NaoQi Pepper. Since choregraphe is used for visual programming, the dialogue boxes need to be constructed where it will be loaded when running the code so that the dialogue topic box will be activated. We have different QiChat Syntaxes that can be used to add rules to the QiChat script. Examples of QiChat Syntaxes that we have used throughout the implementation of your final year project were concept, proposal, user rule, etc. Concept is usually used by us to pick a word that will represent different words or phrases that can be said in the same context with having similar meanings. This makes the dialogue between the human users and NaoQi Pepper to be more flexible. For example, we used the word greeting to represent words like hi, hello, hey, good morning, etc. There are functions that can be used like ^rand and ^first for 'concept' and we have used ^rand quite a bit for it. It basically chooses a random word from the list of words that are being represented. This is mostly used for preventing NaoQi Pepper to say the same word every time. Proposal have been used by us to create different conversational topics inside the dialogue topic boxes. When writing a proposal, it is followed by a sentence which is outputted by NaoQi Pepper. We also write a '%' followed by a word, which serves as the name of the proposal. This allows us to identify the proposal if we want the conversation to jump to a certain conversation topic. We have used functions like, ^nextProposal to move to the next proposal, ^goto to move to a certain proposal, ^gotoRandom to move randomly between proposals with the same name, gotoReactivate to move and reactivate a proposal after it has been deactivated, etc. A user rule is a QiChat Syntax rule that is used to create dialogues that human users will input, followed by a dialogue that a robot outputs. In a user rule, there are also user subrules that we used in the script for our project, which purpose is to create a conversation flow between human users and NaoQi Pepper. We have also added a gesture animation for one of the dialogues of NaoQi Pepper. We added them by writing ^run(animations/Stand/Gestures/Hey_1) which makes NaoQi Pepper wave while greeting the human users. Gesture animation is also part of our non-functional requirements and adding it makes it more interesting and engaging for the human users. We have also added two error catching functions in the QiChat script. These two are written as 'e:Dialog/NotUnderstood' and 'e:Dialog/NotSpeaking5'. We used 'e:Dialog/NotUnderstood' so that when human users say or input something that Pepper does not know or understand yet, then they will reply with a proper phrase like 'Sorry I do not understand' instead of just keeping quiet. The 'e:Dialog/NotSpeaking5' on the other hand gets activated when users does not say anything for 5 seconds. The number of seconds can be adjusted up to 20. When a user does not say anything or respond, then we can set a phrase for Pepper to call human users attention again, like 'Hello?'. The reason why we used 5 seconds is because of it takes around 4 seconds of silence for a conversation to turn awkward. We have also used voice shaping when we were writing QiChat scripts. There are many types of voice shaping like volume, speed, type, etc. The two that we used were speed and type voice shaping. Speed voice shaping affects the speed of NaoQi Pepper's dialogues. We set it by typing '\rspd=85\' before the word, phrase or sentence, where the number is adjustable depending if we want to make the speed slower or faster. The type voice shaping refers to the type of voice that Pepper is using. For example, we added '\style=joyful\' before the 'I win' phrase in 'Games' dialogue box. The reason why we added voice shaping is to make NaoQi Pepper more expressive.

Script editor

Chat/Chat_enu.top

```

1 |topic: ~Chat()
2 |language: enu
3
4 |# Defining extra concepts out of words or group of words
5 |#concept:(hello) [hello hi hey "good morning" greetings]
6
7 |# Catching inputs and triggering outputs
8 |#u:(e:onStart) SonStopped=1
9
10 |# Replying to speech
11 |#u:(~hello) ~hello
12
13 |concept:(greetings) ^rand[hi hello hey "good morning" "good afternoon" "good evening"] {there} {"to you"}
14 |concept:(good) [fine "I am fine" "I'm fine" "I'm good" "I am good" good] {"thank you"}
15 |concept:(bad) [bad "not good" "not well"] {too} {so} {very} {feeling}
16 |concept:(food) [pasta beef pork rice pizza chicken fish seafood vegetables chinese italian mexican] {and}
17 |concept:(books_genres) [action classics "comic books" mystery fantasy history horror fiction romance "science fiction" "short stories" thrillers]
18 |concept:(pets) [dog cat bird fish hamsters ants "guinea pig" mice rabbit turtle] {a}
19 |concept:(goodbye) ^rand["good night" "good bye" bye]
20
21 |u:(~greetings) ^run(animations/Stand/Gestures/Hey_1) \rspd=85~greetings ^goto(ask)
22
23 |proposal: %ask \rspd=85How are you?
24 |   u1:(~good) \style=joyful\ \rspd=85Nice. I'm doing well too. ^nextProposal
25 |   u1:(~bad) \rspd=85Sorry to hear that. I can help cheer you up if you want. We could talk about the food, books, or pets.
26 |     u2:(food): ^goto(food)
27 |     u2:(books): ^goto(books)
28 |     u2:(pets): ^goto(pets)
29 |   u1:(e:Dialog/NotUnderstood) \rspd=85Sorry I do not understand that word. ^nextProposal
30
31 |proposal: %chat %weather \rspd=85Do we have a nice weather today?
32 |   u1:(yes) \style=joyful\ \rspd=85Oh, it might be a great time to go for a walk. ^gotoRandom(chat)
33 |   u1:(no) \rspd=85Oh that's too bad. Let's change the subject. ^nextProposal
34 |   u1:(e:Dialog/NotUnderstood) \rspd=85Sorry, please answer yes or no. ^stayInScope
35
36 |proposal: %chat %food What is your favourite food?
37 |   u1:(~food) \rspd=85That sound delicious. ^gotoRandom(chat)
38 |   u1:(e:Dialog/NotUnderstood) \rspd=85Sorry I do not understand that word. ^nextProposal
39
40 |proposal: %chat %books \rspd=85Do you like to read books?
41 |   u1:(yes) \style=joyful\ \rspd=85That's great. What genres of books do you read?
42 |     u2:(~books_genres) \rspd=85Interesting. ^nextProposal
43 |   u1:(no) \rspd=85Oh okay. Let's change the topic then. ^nextProposal
44 |   u1:(e:Dialog/NotUnderstood) \rspd=85Sorry, please answer yes or no. ^stayInScope
45
46 |proposal: %chat %pets \rspd=85Do you have a pet?
47 |   u1:(yes) \style=joyful\ \rspd=85Wonderful. What pet do you have?
48 |     u2:(~pets) \rspd=85Oh very nice. ^nextProposal
49 |   u1:(no) \rspd=85Oh okay. If you would have one, what would it be?
50 |     u2:(~pets) \rspd=85Interesting choice. ^nextProposal
51 |   u1:(e:Dialog/NotUnderstood) \rspd=85Sorry, please answer yes or no. ^stayInScope
52
53 |u:(What is your name?) \style=joyful\ \rspd=85My name is Nao Pepper. ^gotoReactivate(ask)
54 |u:(What is your favourite food?) \style=joyful\ \rspd=85I like rice and beef. ^gotoReactivate(food)
55 |u:(Do you like to read books?) \style=joyful\ \rspd=85I can't read yet. ^gotoReactivate(books)

```

Ln 1

Find

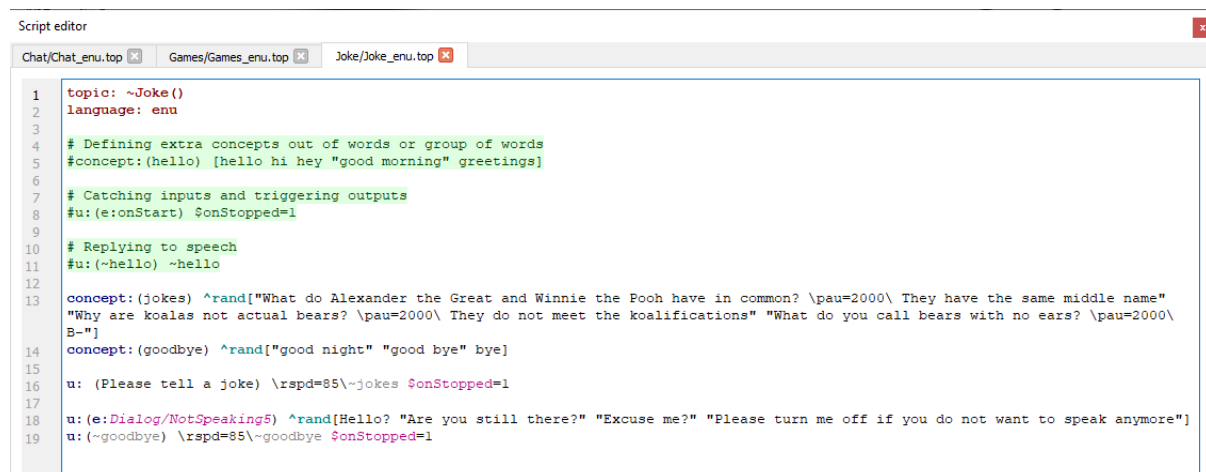
Chat Dialogue Box

The chat dialogue box has been added by us in Choregraphe because that is the dialogue topic box where we write the chatting functional requirements for the human-robot interaction. The main purpose of the chat dialogue box is for us to write QiChat scripts about dialogues between human users and NaoQi Pepper that will fall under the chatting category. This means that they will have conversational topics and conversational flows during the dialogues. The QiChat syntax rules that we used for the chat dialogue box are concept, proposal, user rules, and user subrules. We used concept mostly to have a single word represent different words that can mean the same if said in the same context. The main reason why we added this is because we cannot really accurately predict what the human users will say, so instead we listed down possible words that can be said for every context. For example, when human users say that they are good, they might say 'fine', 'I am fine', or 'I am good', and we use the word 'good' to represent these words. We have also added the ^rand function for one of the concepts because we used that concept for Pepper, and we want NaoQi Pepper to say random greeting words every time and not just the same one. We used proposal in the chat dialogue box for different conversational topics. For example, we gave these proposals names like, 'ask', 'chat', 'weather', 'food', 'books', and 'pets' because these are the conversational topics that we decided on adding. We also used user rules and user subrules for basically human user inputs. Especially for user subrules, which we used a lot because we needed it to create a conversational flow within the proposals. For example, when a human user answers that they like

reading books, then Pepper asks them what book-genres, which leads to another user subrule to continue the conversation.

Joke Dialogue Box

The joke dialogue box has also been added by us in Choregraphe in order to meet the ‘tell a joke’ functional requirement. The QiChat script that we wrote for the joke dialogue box is for the main purpose of creating a dialogue between human users and NaoQi Pepper about Pepper telling jokes. This dialogue topic box is a lot shorter than the other two boxes because there is not that much to write aside from punchlines. We only wrote two concepts for this dialogue topic box, one for jokes with a random function for the reason that Pepper will always randomly pick a punchline from the list to reduce predictability. For example, when human users ask Pepper to tell a joke, Pepper can say, "What do Alexander the Great and Winnie the Pooh have in common? They have the middle name". The other concept is basically just for goodbye when human users want to end conversations with NaoQi Pepper. The only other QiChat Syntax rule that we used is the user rules. We used user rules just for user to input and ask Pepper to tell a joke, as well as error catching in case human users does not reply. The last user rule is just for the user to say goodbye to Pepper. We also added a little bit of voice shaping, where we used \pause=2000\. The reason is so that Pepper waits a bit before saying the punchline.



```
Script editor
Chat/Chat_enu.top Games/Games_enu.top Joke/Joke_enu.top
1 topic: ~Joke()
2 language: enu
3
4 # Defining extra concepts out of words or group of words
5 #concept:(hello) [hello hi hey "good morning" greetings]
6
7 # Catching inputs and triggering outputs
8 #u:(e:onStart) $onStopped=1
9
10 # Replying to speech
11 #u:(~hello) ~hello
12
13 concept:(jokes) ^rand["What do Alexander the Great and Winnie the Pooh have in common? \pau=2000\ They have the same middle name"
14 "Why are koalas not actual bears? \pau=2000\ They do not meet the koalifications" "What do you call bears with no ears? \pau=2000\
15 B-"]
16 concept:(goodbye) ^rand["good night" "good bye" bye]
17
18 u: (Please tell a joke) \rspd=85\~jokes $onStopped=1
19
20 u:(e:Dialog/NotSpeaking5) ^rand[Hello? "Are you still there?" "Excuse me?" "Please turn me off if you do not want to speak anymore"]
21 u:(~goodbye) \rspd=85\~goodbye $onStopped=1
```

Games Dialogue Box

The games dialogue box was also made by us in Choregraphe in order to meet the ‘playing games’ functional requirement for the final year project. The main purpose of the written QiChat script here is for us to create the dialogue between human users and NaoQi Pepper when playing a game between them. The games dialogue topic box is long like the chat dialogue topic box. The QiChat syntax rules that we have written here were concept, proposal, user rule, and user subrules. The concept in this dialogue topic box is used for different variations of word for the word ‘play’ that the human users might say. For example, for the word ‘play’, the human users could say ‘play games’, ‘play game’, ‘let us play a game’, etc. The proposal is used for the different parts for each game. For example, in quiz game, we have three proposals where each proposal has a quiz question, and in rock, paper, scissor game, each proposal is for the possible outcomes of the game. We have used the user rules and user sub rules as the continuation of the conversation flow for each part of the game. For example, if NaoQi Pepper asks the human user if they want to play another go, then it will go to the subrule where user can say yes or no.

```
Script editor
Chat/Chat_enu.top Games/Games_enu.top Joke/Joke_enu.top
1 |topic: ~Games()
2 |language: enu
3
4 # Defining extra concepts out of words or group of words
5 #concept:(hello) [hello hi hey "good morning" greetings]
6
7 # Catching inputs and triggering outputs
8 #u:(e:onStart) SonStopped=1
9
10 # Replying to speech
11 #u:(~hello) ~hello
12
13 concept:(play) [play "play games" "play game"] {"Let us"} {"I would like to"} {a}
14 concept:(goodbye) ^rand["good night" "good bye" bye]
15
16 u:(play) \rspd=85\Okay. Would you like to play some quiz game or rock paper scissor?
17
18 u:(quiz) ^gotoRandom(quiz)
19 u:(rock paper scissor) ^goto(rps)
20
21 [proposal: %quiz \rspd=85\What does the WWW stand for?
22   u1:(World Wide Web) \style=joyful\ \rspd=85\Correct. Would you like another question?
23   u2:(yes) ^gotoRandom(quiz)
24   u2:(no) \rspd=85\Okay.
25   u1:(e:Dialog/NotUnderstood) \rspd=85\Sorry, wrong answer! The correct answer is World Wide Web. Would you like another
26   question?
27   u2:(yes) ^gotoRandom(quiz)
28   u2:(no) \rspd=85\Okay.
29 ]
30 [proposal: %quiz \rspd=85\In what year were the first Air Jordan shoes released?
31   u1:(1984) \style=joyful\ \rspd=85\Correct. Would you like another question?
32   u2:(yes) ^gotoRandom(quiz)
33   u2:(no) \rspd=85\Okay.
34   u1:(e:Dialog/NotUnderstood) \rspd=85\Sorry, wrong answer! The correct answer is World Wide Web. Would you like another
35   question?
36   u2:(yes) ^gotoRandom(quiz)
37   u2:(no) \rspd=85\Okay.
38 ]
39 [proposal: %quiz \rspd=85\According to Greek mythology, who was the first woman on earth?
40   u1:(Pandora) \style=joyful\ \rspd=85\Correct. Would you like another question?
41   u2:(yes) ^gotoRandom(quiz)
42   u2:(no) \rspd=85\Okay.
43   u1:(e:Dialog/NotUnderstood) \rspd=85\Sorry, wrong answer! The correct answer is World Wide Web. Would you like another
44   question?
45   u2:(yes) ^gotoRandom(quiz)
46   u2:(no) \rspd=85\Okay.
47 ]
48 [proposal: %rps \rspd=85\Rock, paper, scissor, Rock!
49   u1:(rock) \rspd=85\Draw! Would you like another go?
50   u2:(yes) ^gotoRandom(rps)
51   u2:(no) \rspd=85\Okay.
52   u1:(paper) \rspd=85\You win!
53   u2:(yes) ^gotoRandom(rps)
54   u2:(no) \rspd=85\Okay.
55   u1:(scissor) ^run(animations/Stand/Emotions/Positive/Happy_4) \style=joyful\ \rspd=85\I win!
56   u2:(yes) ^gotoRandom(rps)
57 ]
```

Results and Evaluation

Results

We have used heuristic evaluation to test and evaluate our implemented functions for NaoQi Pepper. The main reason why we used heuristic evaluation instead of doing an actual experiment with other people is the lack of time to get ethical report to be approved. One of the advantages of using heuristic evaluation for testing is that it helps detect any problems with the system's usability, in this case, any problems with how each dialogue topic box work, whether they are working as expected or not. Another one of the advantages of using a heuristic evaluation is that it is a quicker method which is perfect for situations where we get restricted by time. However, with the advantages comes the disadvantages of using a heuristic evaluation. When it comes to accuracy of results, doing an actual experiment with participant will always have a more accurate result than a heuristic evaluation result. One of the reasons for this is that bias can easily get mixed during the while evaluation process. Also, even without bias, it is very difficult to cover everything with a heuristic evaluation as it takes a lot more time getting more information to increase the accuracy of a heuristic evaluation. The only reason why we used a heuristic evaluation is that it was perfect for our situation, which is being restricted by time.

Note: We have a video of NaoQi Pepper's functions working, which we will be uploading with the document and the code.

Heuristic Evaluation

We used the table template of a heuristic evaluation that we have done before.

Chat

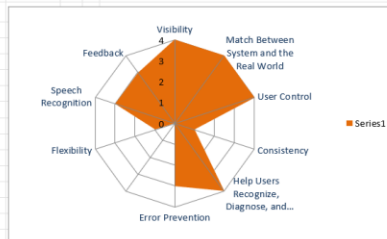
Functional Requirement 1: "Chatting"			
Checkpoint	Severity	Comments	
Visibility	1	4	+ Once running, Pepper can be clearly seen and heard when talking.
Match Between System and the Real World	1	4	+ Using our code, we are able to interact with Pepper in real life.
User Control	1	3	+ Pepper can take be controlled through speech, sound, and device.
Consistency	0	3	+ I can have a continous conversation with Pepper - Sometimes Pepper' speech recognise does not recognise the exact word and may respond completely different from expected.
Help Users Recognize, Diagnose, and Recover From Errors	1	1	+ Pepper responds I do not understand if user says a word that Pepper cannot understand
Error Prevention	-1	3	- No error preventions at all
Flexibility and efficiency of use	1	1	+ Some part of the conversations can be activated using different words, for example, greetings can be said with hi, hello, hey, good morning, etc.
Speech Recognition	0	4	+ Pepper's speech recognition works well enough to finish a whole chatting conversation. - Pepper's speech recognition is not perfect and sometimes can misunderstand us.
Feedback	1	1	+ Pepper lights up when starting, talking, or moving.

Joke

Functional Requirement 2: "Telling Jokes"			
Checkpoint	Severity	Comments	
Visibility	1	4	+ Pepper can clearly be seen or heard when telling a joke.
Match Between System and the Real World	1	4	+ Pepper is able to talk to me in the real world and they tell a joke when the human user asks for it.
User Control	1	4	+ Users can ask Pepper any time to make a joke by either typing or speaking
Consistency	0	1	+ Pepper will always say a joke when asked - Pepper will eventually repeat the joke after a while
Help Users Recognize, Diagnose, and Recover From Errors	0	1	+ If user has not say anything after a while it, Pepper tries to call the attention of the user. - If user says something Pepper does not understand, it just ignores it
Error Prevention	-1	3	- There is no error prevention.
Flexibility	-1	2	- Pepper can only tell a joke with the word 'joke'
Speech Recognition	1	4	+ Speech can be recognised well by Pepper - Pepper does sometimes misunderstand users.
Feedback	1	4	- Pepper responds back when talked to by moving, talking, or lighting up.

Games

Functional Requirement 3: "Playing with them"			
Checkpoint	Severity		Comments
Visibility	1	4	<ul style="list-style-type: none"> + Users can hear and see Pepper when doing a gesture or when speaking.
Match Between System and the Real World	0	4	<ul style="list-style-type: none"> + Pepper can be used in real life for quiz games - Rock, paper, and scissor has a problem with the timing because users might have different timing from Pepper, which is an unrealistic gameplay of this kind.
User Control	1	4	<ul style="list-style-type: none"> + Users can ask Pepper to play a game any time using speech or text.
Consistency	0	1	<ul style="list-style-type: none"> + Pepper can keep asking quiz questions and play rock paper scissors for some time. - Pepper will eventually reach an end in the game somewhere and users will have to ask to play
Help Users Recognize, Diagnose, and Recover From Errors	1	4	<ul style="list-style-type: none"> + When user says a wrong answer in the quiz or say something outside rock paper scissor then the /NotUnderstood/ error catcher will catch the mistake. - There is no function or feature to prevent error.
Error Prevention	-1	3	
Flexibility	1	1	<ul style="list-style-type: none"> + Users can ask Pepper to play by either just saying play, 'let's play', etc.
Speech Recognition	0	3	<ul style="list-style-type: none"> + Pepper can recognise answers pretty well - Pepper can sometimes not recognise speech and makes mistake.
Feedback	1	3	<ul style="list-style-type: none"> - Pepper lights up, talks, moves, and makes a sound when starting up or receiving input from user.



Evaluation

Based off the results from the heuristic evaluation and the actual testing with Pepper on video, we are going to be evaluating the whole design and implementation of the system. One of the positive things about the design and implementation of our human-robot interaction is how clear Pepper is speaking. By adjusting the speed of NaoQi Pepper's dialogue, we were able to slow down how fast it speaks, which makes it a lot easier to understand especially for older adults. Another one of the positive things of our design and implementation is that we were able to develop the human-robot interaction for older adults closer to a real-life use. For example, Pepper, while it might not be perfect, can make a whole chat conversation with the human users using the code we wrote. Another one of the positive things of our designed and implemented system is that NaoQi Pepper will always be consistently available when the older adults need them, except for maybe when the battery runs out, but what I mean is the script itself allows users to have a continuous conversation with Pepper whenever. Another one of the good things about our human-robot interaction for older adults system are the error catching functions. Users are always notified when they make a mistake or error with the input. For example, if a user does not respond to Pepper, after 5 seconds, Pepper will re-confirm the user's presence by asking 'Hello?', 'Excuse me?' etc. Another good thing that we have implemented in the system is NaoQi Pepper's flexibility. The concept and ^rand functions that we added to the QiChat script made Pepper better adapt to different possible words that could be said by users. Another one of the good things of Pepper is the feedback function of NaoQi Pepper. Some of them could be apart of my code, while some of them were built-in NaoQi Pepper already. For example, when Pepper is starting up or opening up, it plays a song to let the users know to wait as Pepper is turning on.

One of the negative things about the human-robot interaction with older adults that we designed is that there is no error prevention at all. There is an error catcher function for the system but that is for once the error or mistake has been done. Another one of the negative things of our human-robot interaction for older adults system is that in the joke dialogue topic box, the flexibility of the system none compared to the other two dialogue boxes. Users can only say the word 'joke' to make NaoQi Pepper to tell a joke. Another one of the negative things of our human-robot interaction system is in

the games dialogue topic box. The main reason why is because of the timing of rock, paper, scissor since NaoQi Pepper's delay in it is always set to two seconds, so human users have the possibility to cheat all they want. Another one of the negative things of our human-robot interaction for older adults system is the consistency. The reason for this is because sometimes NaoQi Pepper makes mistakes when recognising speech. For example, we tested saying 'play' once, but it recognises it as 'hey', so it greeted us back instead. Another one of the negative things is NaoQi Pepper's speech recognition itself. The reason is the same reason with the consistency, it does not always recognise the human user's speech properly.

One of the things that can be improved for our human-robot interaction system for older adults is adjusting the timing for the rock, paper, scissors game. It is true that it can work right now if the human users adjust their timing to NaoQi Pepper's timing, but we cannot rely on that as our human-robot interaction system's main purpose is to support older adults, therefore, we need to find a way to improve the timing where it will be flexible enough for a certain range of human users. Another one of the things that can be improved for our human-robot interaction system for older adults is finding a way to improve the consistency of its functions. If the problem is with NaoQi Pepper the robot itself, then maybe we can adjust the script so that it can improve its speech recognition, and also, we can improve the flow of the joke and games dialogue topic boxes for improvement in consistency too. Another one of the improvements that can be made is the error prevention of the system. We already managed to implement error catching, however, the difficult part is how to implement error prevention for NaoQi Pepper. We are not even sure yet if it is possible to implement in the current version of NaoQi Pepper, but more research relating to NaoQi Pepper can give us the information we need to improve this function.

Future Work

In future works, we are planning to improve the chat dialogue topic box of our human-robot interaction system for older adults. One of our future works will be to for this dialogue topic box is to add more proposal in order to increase the conversational topics that the older adults and NaoQi Pepper can chat about. For example, we could add proposal for conversational topics like movies, games, news, hobbies, etc. Another one of our future works that we are planning to for our current system is to improve the flow of dialogues between older adults and NaoQi Pepper even better. At the moment the flow of conversation in the chat dialogue topic box is alright, but it is very clear that a lot of improvements can still be made. For example, when the user asks Pepper's name, instead of going back to the greeting proposal, we can have a new proposal instead after that to make the flow of conversation flow more smoothly. We are also planning to adjust the QiChat scripts we wrote so that NaoQi Pepper's speech recognition could have the possibility to improve than what it currently is now. We are also planning on adding an error prevention function to the human-robot interaction system for older adults. This might need more research though as we have not seen anything to do with error prevention in QiChat. This will be an important feature for older adults, who mostly are unfamiliar with technologies.

We are also planning to improve the joke dialogue topic box for our human-robot interaction system for older adults for our future works. The first thing that we are going to work at in this dialogue topic box is the flexibility the words that the older adults can say when asking NaoQi Pepper to tell a joke. We could use the word 'joke' as the representative of the other word that can be said by the older adults. We will name a concept as 'joke', then add a list of words that an older adult will possibly say when asking NaoQi Pepper to tell a joke. One of our future works for here was also to add more jokes on the list because at the number of jokes in the list, it will just be a short amount of

time until all the jokes on the list have been heard of, which can lead to the older adults users to get tired of hearing the same punchlines. We are also planning to adjust some of the scripts in this dialogue topic box, like in the chat dialogue topic box, so that NaoQi Pepper's speech recognition can have a chance to improve.

Improving the games dialogue topic box is also part of our future works for our human-robot interaction system for older adults. Adding more quiz questions is first in the list because we only have 3 quiz trivia questions at the moment. After a while, the older adults will probably get bored of it if they keep hearing the same questions asked by NaoQi Pepper. This is the reason why we are planning to increase the questions for the quiz game. We are also planning to improve the timing of the rock, paper, scissor game so that the older adults users will not need to adjust the timing themselves. We are going to try improving the timing by making it flexible, at least to a certain range of people because making the timing completely flexible might prove to be difficult and time-consuming. Another one of the future works that we are planning for this dialogue topic box is to add more games that older adults can play with NaoQi Pepper. We are currently thinking of ideas about this because having only two games in NaoQi Pepper's arsenal can get old quickly. Human users always want something new in a system so we will be constantly adding new features for the system. We are also planning to improve the transition between each game because at the moment, users cannot switch games directly, but needs to stop or finish the current game first, then asking to play again to choose the game they want.

Conclusions

In conclusion of the final year project, we could say we have come far for quite a fair way after all we have done to complete the project. We have used many sources and research that helped in gaining the knowledge we need to start with our final year project. There were so many sources out there, where some are similar to what we are doing for our final year project, which is to design a human-robot interaction for older adults, while some are different but has elements in it that proved to be very helpful for starting my project. A lot of varying types of sources, from web sites to journal articles to YouTube videos to thick books have been all useful with the knowledge that they contained and shared.

We used a lot of different approaches to ready ourselves for the implementation of the project. The first one approach which has the need for us to understand the knowledge and decide on the solution we think is best for the project. We decided on designing a human-robot interaction for older adults as the solution to the problems we identified, which are mostly about older adults' growing rate and the problems that they are doing both emotionally and mentally. The next approach was creating an initial plan which included the work plan in it. The initial plan contained the project description, project aims and objectives, and work plan. This approach was very helpful because we get to have a clear goal because we have it written down, and even though plans changed midway, having the work plan idea for what needs to be done in each week has been helpful with productivity. We also wrote our functional and non-functional requirements as one of our approaches to the project. Functional requirement are requirements that are going to be the main purpose of the system's function or use, while non-functional requirements are requirements that are important to the system but not the main purpose of use of the system. Writing the functional and non-functional requirements made it easier for us to prioritise what to complete first. Finally, last but not the least, there is NaoQi Pepper, which is the socially assistive robot that we used to approach the project.

During the implementation, Choregraphe has been a really useful tool for me to visually construct and program NaoQi Pepper's speech and behaviour. We used Choregraphe because it was also owned by SoftBank Robotics which is the one who developed NaoQi Pepper. We have also discussed about Python 2.7 SDK for the implementation and why it was important to install it even though we did not really utilise when we were implementing the final year project. One of the reasons is because it will be very helpful when we do need it for a bit more complex coding, especially when we try to improve the system in the future. Of course, the language that we were primarily using for the entirety of the project, QiChat, has been one of the main players of us completing this project. QiChat has so many syntaxes and rules, which helped in constructing and creating speeches and behaviours for NaoQi Pepper. QiChat was a lot simpler and easier to use compared to coding with Python language outside of Choregraphe. Dialogue topic boxes are what we use to start writing a QiChat script. We added and created three dialogue topic boxes in Choregraphe for our final year project. These three are chat dialogue box, joke dialogue box, and games dialogue box. Each of the dialogue box was named after the three main functional requirements that we wrote. Inside each of these boxes are scripts that builds up the dialogues for each topic between the older adults and NaoQi Pepper.

Reflection on Learning

The biggest challenge for us during this entire final year project is that we do not know anything about programming with robots because we have no experience with it before. There was a huge learning curve when we started discussing about what needs to be done for the final year project. At the start of the project, we had to leave the first two to three weeks for pure research because we really did not have any idea of how NaoQi Pepper works. It truly does feel like jumping with a blindfold to another world. We have my supervisor, Carolina Fuentes Toro, who really helped a lot with our final year project. We remember that in the first week, we did a lot of research but failed to find anything useful, however, during our first meeting with Carolina, she sent us many helpful sources that was really helpful in gaining more knowledge about the project. We discussed on how often we need to meet to which we answered every week. People may think why we must meet every week, well in my mind, that was more of a pressure to ourselves so that every week we will be pressured to do some work to show for during the next meeting. If we were not meeting every week, we are pretty sure that we probably would have slacked off majority of the time. Since we were meeting every week, we were able to do some work progress constantly even if it was slow some of the times. We also found it difficult installing Choregraphe and Python 2.7 SDK. Installing Choregraphe was quite easy but Python 2.7 SDK became a bit complicated for us because we already have a Python 3 in our personal computer an PATHS. We remember taking a week before figuring out what the problem was with Python 2.7 because at first, it was saying that the modules cannot be found.

Aside from the new environment with working with Pepper, one of the challenges is not having the actual NaoQi Pepper robot. Before we started playing around with Choregraphe, we had no idea as to how we could test our code. At first, we thought that we need to come to campus every time we need to test, but Carolina did explain that there is a virtual robot within Choregraphe for simulation testing. We also had the challenge of learning what QiChat is and that it was an entirely different language. Though, after grasping the basics of QiChat, we were able to understand it a little bit and even realise that it was much easier to write scripts with QiChat for NaoQi Pepper than to write scripts with Python 2.7. It still took more time to get used to writing scripts with QiChat for NaoQi Pepper. We started learning about QiChat Syntax and rules, so we found out that we have a lot more options to use and add for our project. After learning QiChat Syntax, our project became steadier

during the implementation. We started writing QiChat scripts using the syntax and rules that we learned, as for testing we were using the virtual robot mostly. It left us in a bit of a disadvantage because we have no way of learning how it actually sounded because the virtual robot has no voice. Testing it for the first time in the Cardiff University laboratory, the difference of what we thought was working quite well to the problems that arise were quite a few. After testing for the first time with the actual robot, we learned about voice shaping which was really helpful in fixing the problems that we encountered in the lab. When working at home, we made sure to not forget about voice shaping when writing the scripts.

Overall, the new experience and environment was challenging and definitely slowed us down. Unforeseen circumstances also happened during the project itself, for example, like us getting CoVid-19 around week 3 or 4 we think. It definitely delayed our work progress on our final year project. However, even with all these challenges we enjoyed ourselves working on something new and challenging. We really learned a lot and gained knowledges that were not thought during Lectures. In Lectures, we are pretty sure that we did not learn nor use Choregraphe and QiChat. We feel really luck to have this chance to experience a new environment even though it was challenging because we learned and gained lots of new information and knowledge. So, overall, we really feel grateful for having this project as our final year project.

References

- Ge, S. S., Cabibihan, J.-j., Salichs, M. A., Broadbent, E., He, H., Wagner, A. R., & Castro-Gonzalez, A. (2018). *Social Robotics*. Qingdao, China: Springer.
- Jokinen, K. (2018). Dialogue Models for Socially Intelligent Robots. In S. S. Ge, J.-j. Cabibihan, M. A. Salichs, E. Broadbent, H. He, A. R. Wagner, & A. Castro-Gonzalez, *Social Robotics* (pp. 140-151). Qingdao, China: Springer.
- Nakamura, M., Ikeda, K., Kawamura, K., & Nihei, M. (2021). Mobile, Socially Assistive Robots Incorporating Approach Behaviour: Requirements for Successful Dialogue with Dementia Patients in a Nursing Home. *Journal of Intelligent & Robotic Systems* (2021), 1-11.
- Richard. (2019, February 11). *Medium: BlogEmTech: Pepper Intergration with Diagflow*. Retrieved from Medium: <https://blogemtech.medium.com/pepper-integration-with-dialogflow-1d7f1582da1a>
- WHO: News Room: Fact Sheets: Detail: Ageing and Health. (2021, October 4). Retrieved from World Health Organization: <https://www.who.int/news-room/fact-sheets/detail/ageing-and-health>
- Wikipedia: Wiki. (n.d.). Retrieved from Wikipedia: [https://en.wikipedia.org/wiki/Pepper_\(robot\)](https://en.wikipedia.org/wiki/Pepper_(robot))