Sapienza University of Rome

Human **R**obot **I**nteraction

Coach Pepper - Socially Assistive Robots for Physiotherapy

Hossam Arafat - 1803850
Karim Ghonim - 1774086

Supervisors: Prof. Luca Iocchi
Dr. Ron Petrick

July 2019

Contents

1 Introduction

Increases in life expectancy coupled with a record low birth rate mean that for the first time ever, the median age of Italy's residents is over 46 years old. Moreover, more Italians continue to look for work in the healthcare industry abroad than doctors and nurses move to Italy, resulting in a decrease in the workforce. Finally, according to the Istituto Nazionale Di Statistica (ISTAT) [8], As of January 1st 2019, Italians over the age of 65 continued to increase in relative and absolute terms, constituting 13.8 million (22.8 %) of the population.
A shrinking population, stagnant economy, and higher rate of older people mean that more people will require healthcare services. Key to maintaining and improving the overall health of elderly individuals is regular physical exercise, according to [2] and [5]. Furthermore, high perceived social interactions and social support have been shown to reduce the likelihood of depression [4], [7] and have a positive impact on the overall mental and physical well-being of older adults [6].
Socially assistive robotics (SAR) and methodologies based on sound human-robot interaction (HRI) principles can offer socially intelligent robots that provide physical exercise therapy, social interaction, and companionship for the elderly population.
To this end, this paper presents the theoretical background, design principles, and implementation details of a socially assistive robot system that aims to engage elderly users in simple physical exercise. To demonstrate the utility of robots as physiotherapy coaches, SoftBank Robotics' "Pepper" was chosen as it is an expressive, anthropomorphic robot that's safe for physical human-robot-interaction (pHRI) tasks.

2 Design Principles

After reviewing the literature regarding the use of socially assistive robotics in physiotherapy, we chose to adopt the design principles introduced in [3] and detailed below.

2.1 Fluid and Interactive

Psychology research in human motivation -especially Csikzentmihalyi's work into the theory of "flow"- assert that, to achieve a state of "flow"; i.e. maximal enjoyment of a specific task, the task must achieve all requirements of being an optimal challenge, as outlined in [1]. As such, we focused on presenting the human user with a variety of challenging exercises to prevent user boredom (Two were selected here just for demonstration purposes). Another requirement for a task to achieve a state of flow is that it establishes a clear goal with immediate and appropriate feedback [1]. This is why we clearly present each exercise using Pepper's on-board tablet, as outlined in section 4.3. Timely feedback is repeatedly offered to the user using Pepper's speech engine, described in section 4.1.

2.2 Personable and Intelligent

Throughout the exercise, Pepper greets, motivates, and praises the user by name. It also showcases empathy by frequently inquiring about the user's health and state. This promotes a positive, trustworthy user-robot relationship. Special care was taken in using a wide variety of phrases uttered by the robot -motivating or otherwise- to project intelligence and minimize any perception of repetitiveness by the user. This is detailed in sections 4.1, 4.2, and 4.5.

2.3 Task-driven

The interaction with the user should not only focus on conveying trust and motivating them, it must ensure they consistently perform the specified physical exercise and benefit from it. This is achieved by utilizing the different sensors on Pepper to monitor the human user's execution of the exercise, as outlined in section 4.5.

3 Interaction Scenario

A simple interaction was designed where "Pepper" would coach the user through the exercises prescribed to them by their doctor and stored on-board Pepper under their name. For demonstration purposes, this paper will choose a workout session comprising two exercises. The interaction starts by Pepper introducing itself, asking the user for their name and whether or not they are ready to start the entire exercise session. If the user responds negatively, Pepper would ask the user to schedule another session and come back when they're feeling better. If, however, they respond in the affirmative, the robot would then start the sequence for the first exercise which proceeds as follows:

1. An instructional video of the (first) exercise is displayed on Pepper's on-board tablet; a pre-recorded video of the instructor partnering with Pepper to perform the exercise

2. While the video is playing, Pepper would simultaneously provide the user with verbal instructions on how to perform the exercise, namely, how to maintain proper form throughout and touch either of its hands to allow it to count the user's repetitions

3. Pepper would then inquire if the user understood the instructions and is in position to start the first exercise

4. After receiving an affirmative response from the user, Pepper would then move its joints in a pre-programmed way that allows the user to begin executing the exercise with it

5. Pepper would simultaneously say "Starting now", display a count-down for the duration the doctor specified for this exercise on the tablet, and start tracking the user's repetitions by saying "x Right" or "x Left", whenever the user touches Pepper's right or left hand respectively. Where x is the number of times the user has touched the right or left hand so far

6. Once the count-down for the first exercise is done, Pepper would commend the user on a job well done and inquire if they are healthy and in a good state to start the next exercise

7. If the user affirms with "Yes", Pepper would display another countdown video and verbally tell the user to rest between exercises for the duration specified by the doctor

8. Once the rest period is done, Pepper would then start the instructional video and verbal instructions for the second exercise in a manner identical to that of the first

9. Pepper would again ask the user if they are in a good state to start the second exercise

10. If so, Pepper would again move its joints in a way that allows for exercise execution and track the user's progress as it did during the first exercise

11. Once the countdown for the second (and final) exercise concludes, Pepper would commend the user on the good work they did and inform them that it is looking forward to seeing them next session

12. The entire interaction would then terminate

13. If the user responded with "No" to any of the previous "Are you okay?" inquiries posed by Pepper, it would ask them if they are in pain and need it to call their doctor

14. If so, they can simply say "Yes" for Pepper to get their doctor, or, if they are unable to verbally communicate due to their pain or a medical emergency, they can touch either of Pepper hands. Otherwise, Pepper would give the user a longer time to rest, display the countdown on the tablet, terminate the current session, and ask the user to schedule another session when they feel 100% ready, in order to prioritize their safety and well-being

4 Implementation

Softbank Robotics released "NAOqi Framework" which comes with a list of core modules that run on the Pepper robot and control it. These modules come with default functions and methods.

4.1 Speech Synthesis

To enable the robot to speak in a human-like manner, the NAOqi Framework provides a module responsible for audio input and output. It includes an "Animated Speech" module which allows the robot to incorporate gestures in its speech and talk in a more expressive, lively manner. This is achieved in two ways, firstly, Pepper comes with a set of predefined gestures, called "Animations", that can be invoked while saying a specific string by annotating that given string in a certain manner as follows:

- **^run(*a given animation*)** - Pepper stops talking, runs the given animation and then resumes talking
- **^start(*a given animation*)** - Pepper starts the given animation while talking
- **^stop(*a given animation*)** - Pepper stops the given animation
- **^wait(*a given animation*)** - Pepper suspends talking until the given animation is finished, then it resumes speech
- **^mode(*setting*)** - Change the mode of the autonomous "Speaking Movement" module, which we will introduce now

Secondly, Pepper comes with an autonomous module which is enabled by default: "Speaking Movement". This module has three modes:

- **random** - The robot autonomously selects a short neutral animation at random and executes it
- **contextual** - The robot selects a relevant animation from the set of predefined ones each time a hard-coded keyword like 'I' or 'you' is detected in an appropriate grammatical context (for example, it will point at itself when saying "I")
- **disabled** - The robot does **not** autonomously execute any animations; it only executes the ones invoked by the designer through their annotations

The "Animated Speech" module's function *say*() allows the robot to voice any string you pass to it and in our implementation, this function is called on its own dedicated Python thread to not interrupt the execution of other tasks the robot is performing. In order for the robot to execute the gestures and say the words in a properly synchronized manner, the module actually pre-processes the text inside the string before proceeding with any other step. Specifically, it first splits the text based on white spaces, parse it, and if the "Speaking Movement" module is set to **contextual**, a relevant animation will be added to any un-annotated word that the module recognizes such as "Hello", "I", "My".
It will be instructive to offer an example to clearly illustrate how one can tag a string in order to invoke the predefined animations:

Example 1: Annotated string with the "Speaking Movement" mode set to *disabled*

```
"Hello!  ^start(animations/Stand/Gestures/Hey_1)  Nice to meet you!"
```

In this case, the robot will do the following:

- The robot says "Hello!" (the "Speaking Movement" module does **not** autonomously assign any animation based on context here)
- The module reads the annotation **^start()** which tells it to start the animation enclosed in the parentheses, this will cause the robot to simultaneously:

  - execute the *animations/Stand/Gestures/Hey_1* animation
  - say “Nice to meet you!”

- Once the entire string is said, the robot abruptly stops the execution of the animation

Example 2: Annotated string with the ”Speaking Movement” mode set to *contextual*

” ^**start**(*animations/Stand/Gestures/Hey_1*) Hello! ^**wait**(*animations/Stand/Gestures/Hey_1*)
Nice to meet you!”

In this case, the robot will do the following:

- The module reads the annotation **^start()** which tells it to start the animation enclosed in the parentheses, this will cause the robot to simultaneously:
  - execute the (*animations/Stand/Gestures/Hey_1*) animation
  - say “Hello”
  - wait until the gestures / movements associated with the (*animations/Stand/Gestures/Hey_1*) animation are done before proceeding

- Since the mode is set to **contextual**, during the pre-processing step, the ”Speaking Movement” module recognized the ***you*** in “Nice to meet ***you***!”. This resulted in it doing the following:
  - select one animation associated with the hard-coded keyword ***you*** such as the (*animations/Stand/Gestures/You_1*) animation
  - make the robot say “Nice to meet you” and simultaneously
  - execute the (*animations/Stand/Gestures/You_1*) animation

- Once the entire string is said, the robot will wait until the animation itself is done as well

Example 3: Annotated string with the ”Speaking Movement” mode set to *disabled*

” ^**mode**(*disabled*) Hello! ^**mode**(*disabled*) Nice to meet you!”

In this case, the robot will do the following:

- The module reads the annotation **^mode(disabled)** which tells it to deactivate the autonomous ”Speaking Movement” module and not assign animations to any keywords it recognizes

- The robot says “Hello!”

- The robot says “Nice to meet you!” (again, the ”Speaking Movement” module does **not** autonomously assign an animation for the word ***you*** here)

4.2 Speech Recognition

Pepper’s built-in ”Automatic Speech Recognition (ASR)” module was used to detect predefined key phrases uttered by the user. As asking the ASR to detect a large, diverse vocabulary of words causes the system to detect a lot of false positives as well as lower the system’s single-word detection confidence, a limited vocabulary was chosen with only the words (”Yes”, ”No”, *”Luca”*). *”Luca”* - chosen here as it is one of the most common names in Italy, acts as the name of a hypothetical user.
Shrinking the size and diversity of the vocabulary detected by the system is a mixed blessing as it lowers detection errors but makes the system appear less versatile, interactive and intelligent overall. To rectify this issue and make the robot appear more capable and intelligent, it is made to:

- Frequently ask the user many ”Yes/No” questions to inquire about the user’s state, well-being, and need to contact the user’s doctor / care-provider

- Instruct the user on how to perform the next exercise
- Frequently inquire if the user is ready for the next exercise
- Frequently integrate the user's name in its dialogue

After supplying the ASR module with the three word vocabulary discussed above, the robot subscribes to the ASR service and starts listening for audio. When sound from a speaker is heard, a boolean is changed to "True" in a memory location that the "Memory" module calls "SpeechDetected" and gives the designer access to (The "Memory" module is one that is available in the NAOqi Framework and can be used to store and retrieve named values). Once the speaker utters a word that is in the predefined vocabulary -such as "Yes" in our example- an event called "Word Recognized" is raised. This event contains a Python list that contains the following information: [phrase i, confidence i] where *phrase* is one of the i phrases specified in the predefined vocabulary and *confidence* is the probability with which the ASR thinks that this phrase really is what was uttered by the human speaker.
The "Memory" module allows us to subscribe to the "Word Recognized" event using its "*subscriber*()" method. This method returns a dynamic object called "*signal*". This dynamic object has a "*connect*()" method that takes the name of a callback function written by the designer as its argument. "*connect*()" guarantees that the callback function will be called once the "Word Recognized" event is raised.
We can now clarify how we used a limited vocabulary of three words but were still able to make the robot execute a variety of speech-based interactions. Three callback functions were designed and connected to the "Word Recognized" event: one that specifies the response to hearing the human speaker utter his name (*Luca* is the example adopted here), another that specifies what to do when the human speaker says "Yes/No" to indicate their will to start / stop the exercises, and a final one that dictates what happens when the human speaker says "Yes/No" to indicate whether or not they'd like Pepper to call for their doctor. It is important to note that whenever the human speaker says *any* of the predefined keywords in the vocabulary, ***all*** callback functions are executed.
This presented a design challenge when it came to choosing when exactly which callback function should be executed and which to be ignored depending on the certain stage of the interaction the human speaker and Pepper are in. This challenge was overcome by using global variables -flags- that are shared by all threads and functions. Through the proper setting of these flags and if-conditions inside the functions, the correct pieces of code are executed at the right point of the interaction.
The first function, called "*onNameRecongized*", is triggered when the user says his name and is the simplest one, as Pepper begins the interaction by prompting the user for his name, and on speaking it, this function simply makes Pepper say "Hello *Luca*" and ask the human speaker if they are ready to start the session.
When the user responds "Yes/No" to indicate whether or not they are ready to start the session, this prompts the execution of the second function, "*onWordRecognized*". This function only runs if Pepper did indeed greet the user and it handles when Pepper should do the first, second, or no exercises at all. "*onWordRecognized*" is also responsible for giving the user rest as well as asking if they need their doctor between and after the exercises respectively.
Once Pepper asks the user if they need their doctor, it informs them that they can either touch any of its hands to ask her to call the doctor or answer verbally using "Yes/No". The "Yes/No" answer given by the user prompts the execution of the third and final function "*onPainRecognized*". This function does the following:

1. If the user says "Yes":
    - Pepper comforts the user and tells them that it is calling the doctor now while displaying a green "dialing" icon to showcase the call being placed
    - Pepper confirms when the call has been placed and comforts the user that their doctor will be here shortly

2. If the user says "No":
   - Pepper puts the health and safety of the human user first and gives them 30 seconds of rest before asking them to schedule another session when they're back to full fitness
3. If the user elects to touch either of Pepper's hands to confirm that they need their doctor, this case is handled by a different function, "*getDoctor*", which will be discussed in detail in section 4.5

All the different scenarios that might ensue as a result of the human user's "Yes/No" answers are outlined in section 3 and depicted in the flowchart in Appendix A.

4.3 Displaying Relevant Videos

A video is displayed using the "Tablet Service" module. The module is responsible for interfacing with the tablet and the web application running on the tablet's browser. The "*playVideo*" method is called which retrieves the desired video's URL from the web (video must be hosted on the web in MP4 format) and shows it on the screen.

4.4 Joint motion

The "Motion" module is responsible for controlling the movements of the robot's various joints, wheels as well as obstacle avoidance capabilities. To avoid singularities arising from the use of inverse kinematics, the joints are controlled by specifying commands in the joint space, ***not*** in the Cartesian space. Furthermore, all desired motions executed throughout the interaction scenario are done well far from the joint limits.

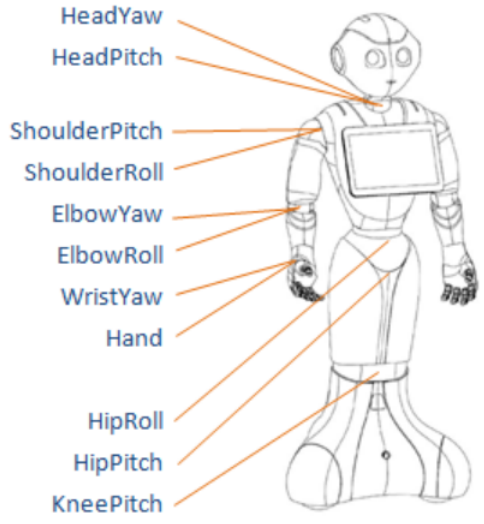
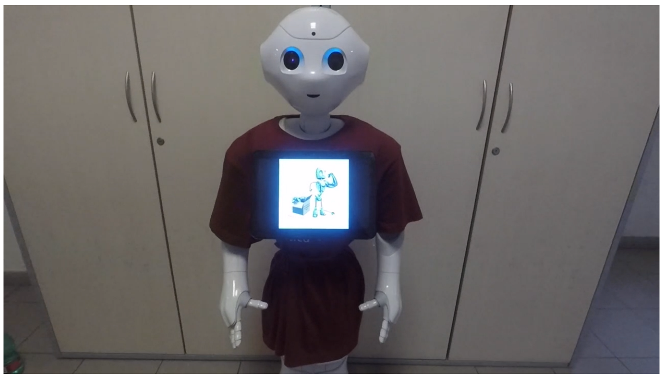
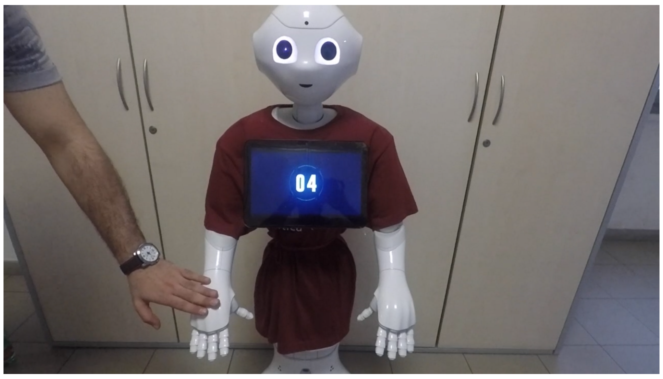


Figure 1: List of Pepper's joints and their names

More specifically, This is done using the "*angleInterpolation*()" function which takes the names of the arm joints (shown in Figure 1) to be moved, the values to move them to, and the time in which to do so. The function then moves the joints in a smooth path from the default configuration to the desired one; both configurations are shown in Figure 2.


(a) Pepper's default shoulder and arm position

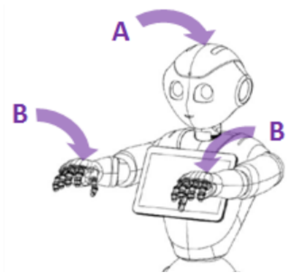


(b) Arm position Pepper takes during the exercise, allowing the user to touch the tactile sensor on its hands

Figure 2: Pepper's different joint configurations

4.5 Sensing & Execution Monitoring

Pepper is equipped with a suite of sensors to facilitate human-robot interaction, including the three tactile sensors shown in Figure (3a) on the head and both arms respectively. To track the user's progress during the exercises and be able to motivate them, Pepper keeps track of how many times the user touched either hand.
This is done using the "Touch" and "Memory" modules where the former detects whenever one of the robot's tactile sensors is touched and then writes the boolean value of "True" in the named memory location allocated for each tactile sensor. The latter module can then be used to read that boolean value from memory.
For example, whenever the robot's left hand is touched, the "Touch" module would return a list as such: [“LArm”, True] and the "Memory" module can be used to access the boolean value of "True" by using the "$getData()$" function to access the memory location associated with the location "Device/-SubDeviceList/RHand/Touch/Back/Sensor/Value".



(a) Tactile sensors on Pepper's Head and Hands

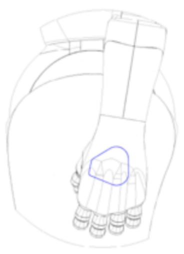

(b) Exact tactile area on Pepper's hand

Figure 3: Pepper's tactile sensors

More specifically, whenever an exercise starts, a dedicated Python thread is launched for the entire duration of the exercise (10 seconds in this implementation) to move the arm to the desired position shown in Figure (2b) and monitor the status of the touch sensors. Moreover, two variables are created to store how many times the hands have been touched so far. Whenever the boolean value of the right or left hand sensors changes to "True", the robot would say that the specific hand was touched x times and increment the variable associated with that hand.
In order to implement the design principles set out for the interaction when it comes to motivating the user, whenever 3 or 6 seconds of the exercise pass, Pepper would say "Good job *Luca*" and "Keep it up *Luca*, just 4 seconds left". And once the 10 seconds allocated for the exercise are done, Pepper informs the user "Time is up. Well done *Luca*". The dedicated Python thread responsible for monitoring the touch sensors would then be shut down and the interaction would continue as outlined in section 3.

Finally, whenever Pepper asks the user if they would like it to call their doctor / caregiver, another dedicated Python thread for the "*getDoctor*" function -that monitors the touch sensors in the exact same fashion described above- is launched. When either hand is touched by the user, Pepper assures the user by saying "Calling the doctor now", displaying a picture on its tablet indicating that the call is being placed, and then saying "Done, successfully called the Doctor, Doctor should be here soon". The dedicated Python thread would then be shut down, Pepper would switch to Idle mode, simply waiting for the user's doctor to arrive.

5 Future Work

5.1 Advanced tracking of exercise execution

Utilizing the recent advances in Deep Learning and computer vision, the famous "OpenPose" Neural Network can be used to extract the "skeleton" of the human user from an input image, i.e. the position and orientation of each joint in the human body, as shown in Figure (4). Given a sequence of images from Pepper's cameras in real time, this data can be fed to the network in order to recognize the specific exercise that the human is performing and whether or not they are performing it using the proper form. This will allow us to introduce exercises that do not need the user to physically touch any sensors on Pepper, increasing the diversity of exercises even further and ensuring user enjoyment.

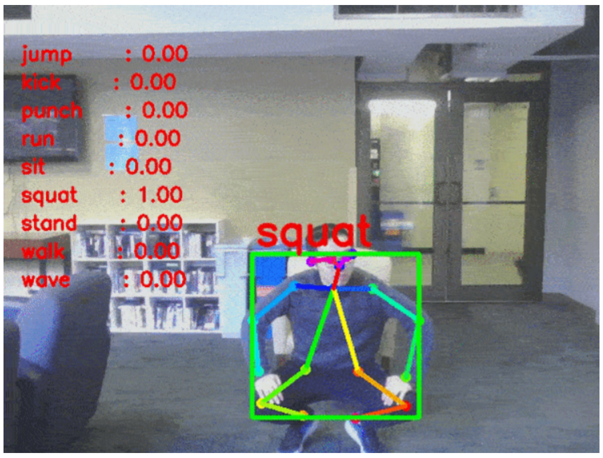


Figure 4: OpenPose is able to extract the "skeleton" of a human from an input image and classify the kind of activity the human is engaged in

6 Conclusion

As the percentage of older adults in the populations of developed countries continues to increase, more and more healthcare providers will start relying on robots in hospitals -or even dispatch them as caregivers at home- to provide the elderly with much needed physical exercise, in a socially positive and motivating manner.
In this work we demonstrated how socially assistive robotic systems (SAR) can be designed and implemented according to design principles backed by research in psychology and motivation. We chose Pepper robot for our implementation and demonstrated how it could guide a human user through therapeutic physical exercises in a safe and reliable manner that puts the health of the user first. Two exercises were shown here for demonstration purposes but the same approach can easily be scaled for any number of exercises.

Appendices

A Interaction Flowchart

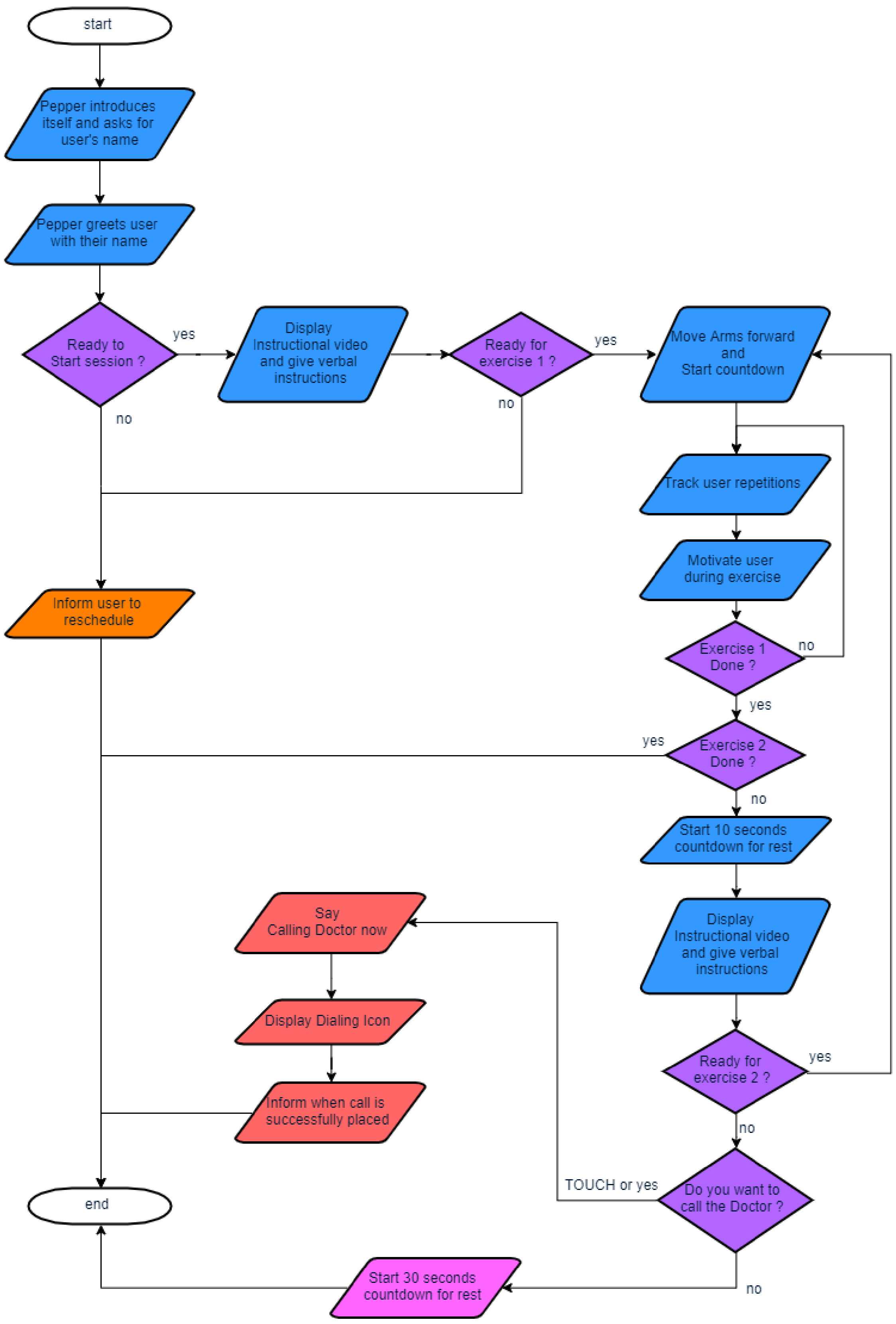

start
Pepper introduces itself and asks for user's name
Pepper greets user with their name
Ready to Start session ?
yes
no
Display Instructional video and give verbal instructions
Ready for exercise 1 ?
yes
no
Move Arms forward and Start countdown
Track user repetitions
Motivate user during exercise
Exercise 1 Done ?
no
yes
Exercise 2 Done ?
yes
no
Inform user to reschedule
Start 10 seconds countdown for rest
Display Instructional video and give verbal instructions
Ready for exercise 2 ?
yes
no
Say Calling Doctor now
Display Dialing Icon
Inform when call is successfully placed
TOUCH or yes
Do you want to call the Doctor ?
no
end
Start 30 seconds countdown for rest

Figure 5: Flowchart of all possible interaction scenarios. Decision nodes are shown in purple and the different scenarios are shown in different colors. In orange: User is not ready for the entire session / either exercise. In blue: User performs both exercises without any pain or needing to call the doctor. In pink: User performs the first exercise but is *not* ready for the second exercise **and** does *not* want their doctor. In red: User performs the first exercise but is *not* ready for the second exercise and wants Pepper to call their doctor.

References

[1] Mihaly Csikszentmihalyi. Beyond boredom and anxiety. san francisco. *CA, US: Jossey-Bass*, 1975.

[2] Doreen Dawe and Robin Moore-Orr. Low-intensity, range-of-motion exercise: invaluable nursing care for elderly patients. *Journal of Advanced Nursing*, 21(4):675–681, 1995.

[3] Juan Fasola and Maja J Matarić. Socially assistive robot exercise coach: motivating older adults to engage in physical exercise. In *Experimental Robotics*, pages 463–479. Springer, 2013.

[4] Linda K George, Dan G Blazer, Dana C Hughes, and Nancy Fowler. Social support and the outcome of major depression. *The British Journal of Psychiatry*, 154(4):478–485, 1989.

[5] MARION ET McMURDO and Lucy Rennie. A controlled trial of exercise by residents of old people's homes. *Age and ageing*, 22(1):11–15, 1993.

[6] ZB Moak and Arpana Agrawal. The association between perceived interpersonal social support and physical and mental health: results from the national epidemiological survey on alcohol and related conditions. *Journal of public health*, 32(2):191–201, 2009.

[7] Eugene S Paykel. Life events, social support and depression. *Acta Psychiatrica Scandinavica*, 89:50–58, 1994.

[8] Istituto Nazionale Di Statistica. Demographic indicators. *Journal of Advanced Nursing*, 2019.