Accessible Integration of Physiological Adaptation in Human-Robot Interaction

by

Austin Kothig

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2021

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Austin Kothig was the sole author for Chapters 1, 2, and 5 which were written under the supervision of Prof. Kerstin Dautenhahn. This thesis includes first-authored peer-reviewed material that has appeared in conference proceedings, published by Springer International Publishing and the Institute of Electrical and Electronics Engineers (IEEE).

Research presented in Chapters 3 and 4:

In Chapter 3, Austin Kothig, John Muñoz, Hamza Mahdi, and Alexander M. Aroyo collaboratively designed the initial framework described in [51] and later refined in [50] by Austin Kothig, John Muñoz, Alexander M. Aroyo. Each component of the framework is expanded on for this thesis by Austin Kothig. The Implementations of software and their descriptions was solely written by Austin Kothig.

In Chapter 4, the exercise routine in [50] was co-designed and run with John Muñoz and Austin Kothig. Sami Alperen Akgun participated in coding of some ROS specific implementations for the exercise scenario. The section describing other potential use cases was initially drafted by Austin Kothig and Hamza Mahdi in [51], and extensively expanded on for this thesis by Austin Kothig.

All authors reviewed and edited manuscripts.

> *Citations:*
>
> Austin Kothig, John Muñoz, Hamza Mahdi, Alexander M. Aroyo, and Kerstin Dautenhahn. 2020. HRI Physio Lib: A Software Framework to Support the Integration of Physiological Adaptation in HRI. In *International Conference on Social Robotics (ICSR '20)*. Springer International Publishing, 36-47.
>
> DOI: https://doi.org/10.1007/978-3-030-62056-1_4 [51]
>
> Austin Kothig, John Muñoz, Sami Alperen Akgun, Alexander M. Aroyo, and Kerstin Dautenhahn. Connecting Humans and Robots Using Physiological Signals – Closing-the-Loop in HRI. In *2021 30th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE.
>
> DOI: http://dx.doi.org/10.1109/RO-MAN50785.2021.9515383 [50]

Abstract

Technological advancements in creating and commercializing novel unobtrusive wearable physiological sensors have generated new opportunities to develop adaptive human-robot interaction (HRI). Detecting complex human states such as engagement and stress when interacting with social agents could bring numerous advantages to creating meaningful interactive experiences. Bodily signals have classically been used for post-interaction analysis in HRI. Despite this, real-time measurements of autonomic responses have been used in other research domains to develop physiologically adaptive systems with great success; increasing user-experience, task performance, and reducing cognitive workload.

This thesis presents the `HRI Physio Lib`, a conceptual framework, and open-source software library to facilitate the development of physiologically adaptive HRI scenarios. Both the framework and architecture of the library are described in-depth, along with descriptions of additional software tools that were developed to make the inclusion of physiological signals easier for robotics frameworks. The framework is structured around four main components for designing physiologically adaptive experimental scenarios: signal acquisition, processing and analysis; social robot and communication; and scenario and adaptation. Open-source software tools have been developed to assist in the individual creation of each described component.

To showcase our framework and test the software library, we developed, as a proof-of-concept, a simple scenario revolving around a physiologically aware exercise coach, that modulates the speed and intensity of the activity to promote an effective cardiorespiratory exercise. We employed the socially assistive QT robot for our exercise scenario, as it provides a comprehensive ROS interface, making prototyping of behavioral responses fast and simple. Our exercise routine was designed following guidelines by the American College of Sports Medicine. We describe our physiologically adaptive algorithm and propose an alternative second one with stochastic elements.

Finally, a discussion about other HRI domains where the addition of a physiologically adaptive mechanism could result in novel advances in interaction quality is provided as future extensions for this work. From the literature, we identified improving engagement, providing deeper social connections, health care scenarios, and also applications for self-driving vehicles as promising avenues for future research where a physiologically adaptive social robot could improve user experience.

Acknowledgements

I would like to first thank professor Kerstin Dautenhahn for her tremendous support and mentorship throughout this challenging and difficult journey.

I'd also like to thank my thesis readers, professor Ning Jiang and professor Fakhri Karray, for reading my thesis and providing thoughtful questions and feedback.

Thank you to Elaheh Sanoubari, your wise words, and mentorship have been pivotal in developing my academic maturity over the last two years.

Thank you to Hamza Mahdi, Sami Alperen Akgun, John Muñoz, and Alex Aroyo for your valued support and friendship on this project. Thank you to Jessy Song and Iris Fang who not only provided support on this project, but also gave me my first opportunity to be a mentor. A big thank you to the members of the Social and Intelligent Robotics Research Laboratory, for the consistent and high quality feedback.

A very special thank you to Juliette Teodoro for being the best support I could have asked for. I couldn't have reached this monumental milestone without you.

Finally, an enormous thank you to Marko Ilievski, Nicole Dillen, Mike Shackerman, and Eric Sosa for your continuous friendship and valuable feedback.

This research was undertaken, in part, thanks to funding from the Canada 150 Research Chairs Program. We acknowledge support from the Microsoft AI for Social Good Initiative.

Dedication

This is dedicated to my friends, loved ones, and family, whom I thank deeply for your unbridled support.

Table of Contents

List of Figures ix

List of Tables xiv

1 Introduction 1

1.1 Motivation . . . 1

1.2 Research Objectives and Challenges . . . 3

1.3 Contributions and Thesis Outline . . . 3

2 Related Work 5

2.1 Human-Robot Interaction . . . 5

2.2 Affective Computing . . . 8

2.2.1 State Representation . . . 9

2.3 Physiologically Adaptive Systems . . . 10

2.3.1 Physiological Signals . . . 11

2.3.2 Use in Research . . . 12

2.4 Existing Tools . . . 14

3 HRI Physio Lib 16

3.1 Framework . . . 17

3.1.1 Signal Acquisition . . . 17

3.1.2 Processing and Analysis . . . 19

3.1.3 Social Robot and Communication . . . 20
3.1.4 Scenario and Adaptation . . . 23
3.2 Architectural Structure . . . 23
3.2.1 High-Level Architecture . . . 24
3.2.2 Low-Level Architecture . . . 27
3.3 Implemented Applications And Tools . . . 36
3.3.1 Physio Receiver . . . 36
3.3.2 Polar Streamer . . . 38
3.3.3 Physio Streamer . . . 40
3.3.4 NeuroKit Processor . . . 40
3.3.5 NeuroKit Simulator . . . 42
3.4 Discussion . . . 42

4 QT Cardio-Aware Exercise Coach 44
4.1 Social Robots as Exercise Coaches . . . 45
4.2 Implementation of the Experimental Scenario . . . 46
4.2.1 The Socially Assistive QT Robot . . . 46
4.2.2 The Exercise Routine and Experimental Design . . . 47
4.2.3 Implementation Details of Experimental Scenario . . . 55
4.3 Results of Scenario and Future Extensions . . . 60
4.3.1 Limitations in Implementation . . . 65
4.4 Other Potential Use Cases . . . 66

5 Conclusion 69
5.1 Summary of Contributions . . . 69
5.2 Lessons Learned . . . 71
5.3 Limitations . . . 72
5.4 Future Directions . . . 72

References 74

List of Figures

2.1 Experimental setup: (Top) robot investigator (RI) debriefing a witness (W); (Middle) human investigator (HI) debriefing a witness (W); (Bottom) interrogation room display (Informed consent of participants has been obtained for the use of their photo). Obtained with permission [36]. 7

2.2 The Affect Grid. The x-axes relates to valence, or the perceived pleasure of the state; while the y-axes relates to arousal or the perceived intensity of the emotion. Design inspired by Russell *et al.* [84]. 10

2.3 A simulated electrocardiogram (ECG) to show the structural features of the signal. 12

3.1 Polar heart rate monitors. (left) Polar H10, a chest strap device capable of transmitting ECG, HR, RRI, and accelerometer data over Bluetooth. (right) Polar OH1, an armband optical device capable of transmitting PPG, HR, PPI, and accelerometer data also over Bluetooth. 18

3.2 A few examples of the many different ways facial expressions can be represented in social robots. (a) Multi-color LEDs under the face; (b) Screen-based face display; (c) Backlit projection onto the faceplate. . 21

3.3 Overview of the `HRI Physio Lib`. Color code explanation of the categories per each component as follows: *purple*–body process, *yellow*–hardware elements, *grey/white*–software elements with their individual components, and *blue*–data or information flow. 25

3.4 The Core namespace. Shown are the `RingBuffer` and `Graph` objects which are generic data structures that are part of the software library. Future generic implementations of useful data structures will be found here. 28

3.5 (left) The conceptual idea of a ring buffer storage. (right) The actual implementation of ring buffer storage. A pointer to the head and tail float along a flat array to indicate where the start and end indices are. The last index of the flat array precedes the first index to form a seamless ring shape. 28

3.6 The Factory namespace. Shown is the `StreamerFactory`, which implements the factory software design pattern to abstract the process of instantiating objects derived from the `StreamerInterface` found in the stream namespace. Future implementations of factory-like objects will be found here. 29

3.7 The Manager namespace. Shown are the `C++` manager classes which have been designed to *handle* and manage objects and functions which they encapsulate. In the case of the `ThreadManager`, it receives functions to spawn threads for, while managing their lifetime. The `PhysioManager` and `RobotManager` objects have the purpose to encapsulate polymorphic objects which are passed to them and manage their individual behaviors. White arrow heads represent inheritance. 30

3.8 The Processing namespace. Shown are the `C++` processing algorithms that have been developed so far. Future implementations of processing methods will be organized into this namespace. Black arrowheads represent composition relationships, while white arrowheads represent inheritance. 31

3.9 Equations for calculating the coefficients for various Butterworth filters. f_{cut} is the cutoff frequency for the high pass and low pass filters. f_{cent} is the center frequency for the band pass and band reject filters. f_{rate} is the sampling frequency of the expected signal which the bilinear transformation will be applied to. 32

3.10 The Social namespace. Shown is the abstract class `RobotInterface`. This `C++` interface acts as a starting point for developing robot controllers to inherit from for designing complex behaviors. 34

3.11 The Stream namespace. Shown are the `C++` derived classes from the `StreamerInterface`. The four concrete streamers implement the pure virtual functions from the parent class and encapsulate the necessary components to provide data communication using their namesake. White arrowheads represent inheritance. 35

3.12 The architecture of the `Physio Receiver`. The individual `C++` classes are shown, and how they relate to each other in a pseudo-UML format. Black arrowheads represent composition relationships, while white arrowheads represent inheritance. 37

3.13 The `Polar Streamer` an Android application designed to stream physiological data over Lab Streaming Layer. The application supports the Polar H10 chest strap heart rate monitor and the Polar OH1 armband optical heart rate monitor. 39

3.14 The `Physio Streamer` is an in-development desktop application for data acquisition from physiological devices. The user interface contains four "blocks". (top-left) Scan begins searching for Bluetooth devices which are broadcasting themselves that are near. Found devices are populated into the combo box which by default reads "No devices found". Clicking connect will establish a connection to said device, and add it to the list of connected devices (bottom-left). Connected devices can toggle streaming and recording based on the selected settings. (top-right) Settings for the selected device. Buttons for minimizing the current device back into the tray, or disconnecting the device are present. Update and clear change or reset the prefix of the LSL stream to the string currently in the text box. Checkboxes enable or disable acquisition of the relevant metric from the connected device. (bottom-right) If checked in the graph settings, relevant plots will occur in the display. 41

4.1 The splash screen reads QT Physio Coach and is shown to the participant during the exercise scenario. The splash screen contains an animated green ECG signal which loops across the screen. 48

4.2 The timeline of the exercise routine. Beginning with a 5-minute warm-up phase, followed by 10-minutes of conditioning exercise, and ending with a 5-minute cool-down phase. Activities that occurred in each phase and their duration are shown from top to bottom. 49

4.3 Basic marching; a complete description of movements for the activity in table 4.1. 50

4.4 Step-up reach and pull (a1); a complete description of movements for the activity in table 4.1. 50

4.5 Lateral knees (a2); a complete description of movements for the activity in table 4.1. 51

4.6 Both arms forward (a3); a complete description of movements for the activity in table 4.1. 51

4.7 Calibration phase of the exercise scenario. The QT robot plays relaxing music and videos while collecting the participants' heart rate for three minutes to compute the resting heart rate. 54

4.8 Conditioning phase of the exercise scenario. The QT robot leads the participant through a variety of cardio activities with an aerobic stepping platform. 55

4.9 The architecture of the `QT Physio Coach`. The individual `C++` classes are shown, and how they relate to each other in a pseudo-UML format. Black arrowheads represent composition relationships, while white arrowheads represent inheritance. 57

4.10 The architecture of the `QT Controller`. The individual `C++` classes are shown, and how they relate to each other in a pseudo-UML format. Black arrowheads represent composition relationships, while white arrowheads represent inheritance. 58

4.11 Overview of how the various modules used in the `QT Cardio-Aware Coach` scenario communicate. All devices (with the exception of the *Polar Device*) were connected over a local WiFi hotspot created by the QT robot. From left to right, the *Polar Device* connects to the *Polar Streamer* (running on the Android tablet) over Bluetooth Low Energy (BLE). Data received is streamed out onto Lab Streaming Layer (LSL) which is then caught by the *Physio Receiver*. The data is then transmitted over a Robot Operating System (ROS) publisher to the *QT Physio Coach* (the primary decision loop of the scenario). Selected behaviors are sent to the *QT Controller* over ROS which then translates the commands to correctly broadcasts actions to the QT robot's interfaces. 60

4.12 Change in heart rate (beats per minute – BPM) over time (minutes) of exercising with the `QT Cardio-Aware Coach`. The green shaded region between 125.2 (HRR_{40}) and 155.1 (HRR_{70}) BPM shows the target heart rate zone of the participant. The HR_{max} for our participant was 184.9 BPM and the HR_{rest} was 85.4 BPM. Top shows the four phases of the exercise experiment. With the exception of the calibration phase, times that do not have an orange box, the participant was instructed to perform a basic marching exercise. The activities used were: (a1) step-up reach and pull; (a2) lateral knees; (a3) both arms forward. 61

4.13 Probability distribution of a rule update occurring with the proposed fuzzy rule (algorithm 2). Using the measured HRR_{40} (125.2 BPM) and HRR_{70} (155.1 BPM) from the `QT Cardio-Aware Coach` scenario we find the HR_{target} to be 140.2 BPM. The closer the $HR_{exercise}$ is to the HR_{target}, the less likely it is for a rule update to occur. If the $HR_{exercise}$ is bellow HRR_{40} it is guaranteed to trigger a rule update ("speed up"), while if the $HR_{exercise}$ is above HRR_{70} a rule update ("slow down") will occur. 63

4.14 Probability distribution of rule update occurring with the proposed fuzzy rule (algorithm 2), over top the data from the `QT Cardio-Aware Coach` scenario. The closer the $HR_{exercise}$ is to HRR_{40} or HRR_{70}, the more likely a rule update is to occur. Red is used to show the likelihood of a "speed up" update occurring, while blue is used to show the likelihood of a "slow down" update occurring. A $HR_{exercise}$ bellow HRR_{40} or above HRR_{70} are guaranteed. See Fig. 4.13 for further detail. 64

List of Tables

4.1 Description of the different exercise activities used in the QT Cardio-Aware Exercise Coach use case. Note: Assumes a right-sided dominance; all activities are repeated and alternate with the opposite foot beginning the next cycle. 53

Chapter 1

Introduction

Wearable technology in recent years has grown in a significant way, becoming a common tool in people's lives to monitor their daily steps, caloric burn, exercise levels, heart rate, sleep patterns, and more [55]. The maturity of this technology in the domain of research and commercialization is still quite new. With that said, public adoption of this technology has boomed in recent years, so much so, that a 2019 study by the Pew Research Center [101] reported that over 1 in 5 U.S. adults reported that they regularly wore a smart-watch or fitness-tracker. The costs associated with equipping sensors such as electrocardiogram (ECG) and photoplethysmogram (PPG) onto small devices have fallen dramatically in recent years, resulting in widespread adoption. With the technology already in the hands of so many consumers, we have an opportunity to develop software capable of widespread adoption.

The development of physiologically adaptive systems is still relatively novel, with studies using the technology becoming somewhat more frequent in the domains of human-robot interaction (HRI) and human-computer interaction (HCI). The purpose of this project is to provide a framework for developing physiologically aware scenarios, as well as software and tools to make the integration of physiological measurements into experimental scenarios more accessible.

1.1 Motivation

A critical aspect of amiable HRI is the ability to reason about the subjective emotional states of humans. Evidence suggests that the mechanisms of empathy and perceiving the emotional states of others is phylogenetically ancient, going back to

common ancestors of mammals and birds [18]. Paiva *et al.* [71] describes how empathy in virtual agents could be a way of improving affinity, and overall perception of the interaction quality.

The visual medium has classically been used in emotional recognition, utilizing cues from facial and body language [24, 37, 91]. Typically these methods work well in clean and predictable scenarios, such as when a person's face is clearly visible in a picture or video. However, this technology can struggle in real-world applications, having challenges related to poor or uneven lighting, low resolution, motion blur, as well as incomplete and obstructed facial features (*e.g.,* people moving around freely, facing away from the camera). State-of-the-art methods trained on real-world datasets containing the aforementioned imaging conditions perform at near 50% accuracies in classifying common prototypical expressions [28, 67].

Another challenge in the utilization of the visual medium deals with end-users in the real-world. Caine *et al.* [11] surveyed a population of 18 older adults and found they had privacy concerns they felt when being monitored by either cameras or social robots (which also usually have cameras embedded). A. Sharkey and N. Sharkey [93] had similarly included loss of privacy as one of their major ethical issues in their guidelines for introducing social robots to eldercare scenarios. Wearable sensor and their widespread adoption may provide a more dignified non-invasive solution to well-being monitoring in older populations. Motti and Caine [62] found that while privacy concerns were present with wearable sensors, participants thought it was less of a concern compared to camera-based monitoring. With the increasing popularity of wearables and physiological sensing technologies, this has the potential to help close-the-loop in HRI scenarios.

Psychophysiology is the field that studies the relationship between human psychological states and physiological signals. Psychophysiological states such as stress, workload, and engagement have been studied previously, demonstrating that human physiological responses can be attributed to specific human states [10]. People are typically able to deduce this through a repertoire of visual and auditory cues including body language, facial expressions, gaze orientation, and tone of voice [74]. While some algorithms have been developed to discern some of these cues [22, 63], inferring people's affective states is by no means a categorically solved problem. Physiological data may provide a window into a person's affective state, allowing to capture unconscious responses associated with both the central and peripheral nervous systems. Their inclusion may be the modality necessary to provide exceptional quality interaction with social robots.

1.2 Research Objectives and Challenges

This thesis aims at understanding the state-of-the-art of physiologically adaptive systems, how they're designed, what features are important in adaptations, how real-time adaptations are perceived by users, and what adaptations social robots are capable of as a part of being an adaptive system.

This information is used to formulate a framework to act as a structural base from which future adaptive systems can be derived. Software tools and applications are also developed to assist and provide a foundation for developing physiologically adaptive scenarios. The research questions this project aims to answer are:

RQ1: How can we design a robotics software framework for the rapid development of physiologically aware robots?

RQ2: How can we develop a robotics software framework capable of integrating various communication libraries?

RQ3: Is the proposed software framework and HRI Physio Lib able to be used for the rapid development of physiologically aware robots capable of adapting their interactions?

RQ1 and RQ2 are addressed in Chapter 3, while RQ3 is addressed in Chapter 4.

1.3 Contributions and Thesis Outline

The contributions of this thesis can be summarized as:

1. Development and discussion of a general framework for developing physiologically adaptive HRI scenarios.
2. An open-source software library called `HRI Physio Lib`[1], released under BSD 3-Clause License.
3. Development of a practical use case scenario involving a physiologically aware exercise coach using our software library.

[1] https://github.com/kothiga/hri-physio

4. Supplemental discussion regarding additional potential use case scenarios where physiological adaptations could yield meaningfully positive user experiences.

The organization of this thesis has Chapter 2 providing the necessary background information on human-robot interaction, affective computing, physiologically adaptive systems, and what tools currently exist for facilitating physiological computing and complex robot interactions.

Next, Chapter 3 takes the information we've gathered to develop a structural framework for developing physiologically adaptive HRI scenarios. This chapter also includes the architectural structure of the software library that has been developed to assist in the development of these scenarios. The implementation of applications included in the software repository is detailed.

Chapter 4 uses the framework and software library to develop a physiologically adaptive scenario using the QT robot as an exercise coach. The chapter begins with a lite review of social robots and virtual agents in exercise-related studies, as motivation for the experimental scenario. The implementation of the physiologically adaptive scenario is detailed, including information about the socially assistive QT robot, our exercise routine, experimental design, and details of the software are provided. The results of the scenario and future extensions are discussed, providing improvement recommendations for the system, alternative rule updates, and limitations of the implementation. A supplemental discussion regarding additional use cases for physiologically adaptive systems is also provided as extensions for this work.

Lastly, Chapter 5 concludes with remarks about the research project. A summary of the contributions is discussed, along with lessons learned, limitations, and future directions for this work.

Chapter 2

Related Work

This chapter provides an overview of relevant research in order to provide the context and motivation for developing a framework and software library to facilitate the creation of physiologically adaptive HRI scenarios. In addition, we provide discussion about existing tools, and how they've influenced our design.

2.1 Human-Robot Interaction

Human-robot interaction (HRI) provides a unique challenge of combining computation interfaces and social interaction. Where domains such as human-computer interaction (HCI), rely on graphical interfaces for interaction, HRI simply has the robot's affordances (*e.g.,* body language, gaze, speech, expressions) as the interface for interaction. It is for this reason that the ability for HRI to possess some ability to dynamically adapt their behavior based on inferences about the human using (interacting with) said system is such a widely desired quality.

Previous studies have shown that a system capable of inferring and approximating the affective state of the user can lead to an increase in user-experience, task performance, and reduced cognitive workload [42, 81, 106]. Paiva *et al.* [71], describes that to truly provide a more natural and personalized HRI experience, robots too should possess some level of awareness into the subjective emotional (affective) states.

Oertel *et al.* [69] describes engagement as the utmost important concept in human-machine interaction. The authors state:

> "These agents [robots] all have a common goal, namely to have users continue interacting with them and thus manage users' engagement in the interaction. Thus, for human-agent interaction engagement, both perception and generation are important issues. Perceiving how engaged users are can be beneficial information for adapting agent behavior."

Detecting complex human states such as engagement and stress when interacting with social robots could bring numerous advantages to create meaningful interactive experiences. Robots of a great opportunity for sensor fusion, to combine multiple dimensions perceptions (*e.g.,* vision, auditory, touch) to understand contextual information. This is, of course, easier said than done. In many studies, this problem is partially side steps, relying on the addition of peripheral devices, so that a focus on the interaction can be developed.

The use of eye gaze has been ubiquitous in HRI studies. Gonzalez *et al.* [36] developed a system that makes robots spot liars. This system used pupil dilations recorded using a pair of Tobii Eyetracking glasses. Pupil dilations have been shown to increase when a person is under stressful situations that cause a higher cognitive load, such as not being prepared to lie. The scenario was set up with an investigator (human or robot), interviewing a witness (human participant), about a video that was shown to them of a person shoplifting. The goal for the participant was to either truthfully or deceptively (depending on assigned role) describe the scene they witnessed. The experimental setup for this study can be seen in Fig. 2.1. The authors found that the robot investigator's presence did not cause significant differences to the pupillometry data compared against a human investigator; as well, the machine learning model used in this scenario was found to accurately detect when a lie had occurred.

Besides pupillometry, there are other wearable sensors capable of recording autonomic response and physiological data, which can be used to develop systems that adapt to the users' affective states. Social robots have rarely been used in such physiologically adaptive systems due to multiple complexities from both the human-physiology and human-robot perspectives [54]. In the following sections, we'll take a small step back to discuss affective computing, state representation, physiologically adaptive systems, and some tools which we currently have for developing these systems.

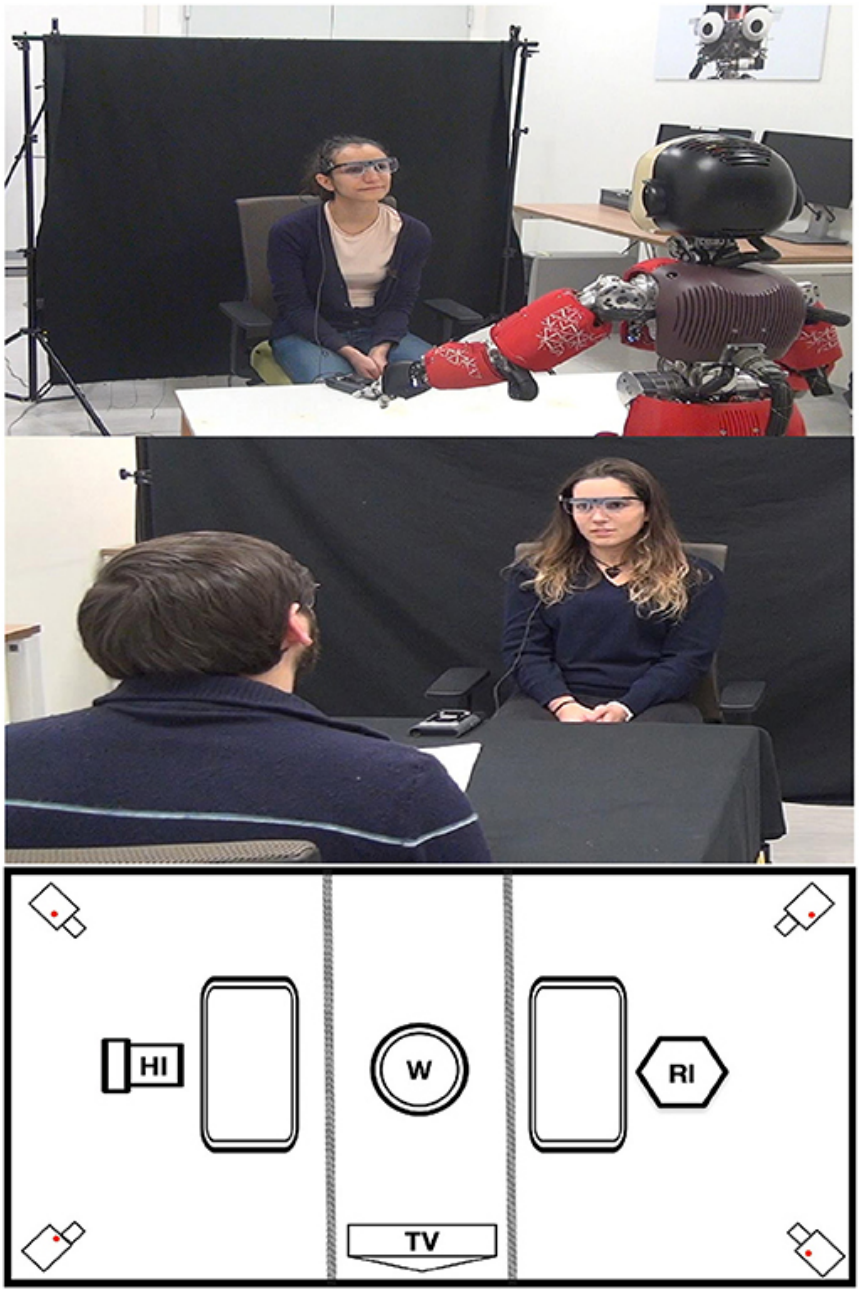


Figure 2.1: Experimental setup: (Top) robot investigator (RI) debriefing a witness (W); (Middle) human investigator (HI) debriefing a witness (W); (Bottom) interrogation room display (Informed consent of participants has been obtained for the use of their photo). Obtained with permission [36].

2.2 Affective Computing

Affective computing is defined by Picard [73] as "computing that relates to, arises from, or influences emotions". Previously described, the ability to sense and recognize the emotions of a user is a powerful asset, that can be used to empower individuals. One domain with a clear perspective of how this technology can be used for good is education. For a student, receiving tutoring can be a stressful and cognitively demanding endeavor. A system that is capable of identifying when frustrations of a user are rising could do well by deescalating the situation when the material isn't quite being retained. As well, detection of boredom and fatigue are useful indicators that the current approach is not working. Woolf *et al.* [103] developed an agent-based system to maximize the time of maintaining student interests in a tutoring scenario. Their system utilized high and low-level observational behaviors, along with skin conductance and facial electromyography (EMG), to approximate the students' affective state in terms of arousal and valence. They describe the purpose of their student is not only to improve the quality of teaching, but also "... to support a student's meta-affective state, or reflection about their emotion".

Robots are not anything new in affective computing. One such robot, PARO[1] is designed as a therapeutic robot. PARO has commonly been used in studies involving older populations. One such study by Šabanović *et al.* [86] used PARO with 10 participants who were residents of a nursing home with varying levels of dementia. Over the seven weekly therapy sessions, PARO was shown to provide improved cognitive abilities of the participants and attentiveness towards the robot and their environment.

KASPAR (Kinesics And Synchronization in Personal Assistant Robotics) [17] is another socially assistive robotics platform design around robot-assisted play and therapy for children with autism. Wainer *et al.* [102] used KASPER in a long-term study, across ten weeks, to understand the effects the robot had on the social behaviors of children with autism in a collaborative play scenario. The authors found that the children engaged with the robot and each other during play. As well, the children showed improvements in social behaviors with each other after collaboratively playing with the robot.

We've shown some of the benefits affective computing can provide for users, whether they're interacting with a computer-based interface or a socially interactive robot. One of the great challenges in affective computing is modeling, or representing

[1] http://www.parorobots.com/

a users' internal state. In the next section, we will describe a range of approaches used in the literature for representing the subjective experiences of participants.

2.2.1 State Representation

Affect can be measured through a variety of methods, with most naive approaches being rooted in scales and questionnaires for post-experiment analysis, as the most direct approach is to simply ask the participant what their experiences in a scenario were. Likert [53], and the Intrinsic Motivation Inventory (IMI) [85] are some of the most common of these scales. Likert is often depicted as a 5 or 7-point scale with responses ranging from "strongly disagree" to "strongly agree". IMI is similarly a 7-point scaling system, however, what sets it apart is that it has 45 items (questions) which are broken into different subscales to allow for multidimensional measures.

Other questionnaires and scales for system usability, enjoyment, positive affect, which appear frequently in the literature include the NASA Task Load Index (TLX) [40], Inclusion of Other in the Self (IOS) questionnaire [2], System Usability Scale (SUS) [9], the Godspeed Questionnaire Series (GQS) [5], and the Robotic Social Attributes Scale (RoSAS) [12], to name a few. These scales are valuable for post-processing analysis (after the experiment has concluded) to understand if the differences between two independent variables hold significance.

The first step in making intelligent decisions based on measured affect is to model it. Extensive work has been done to reduce complex subjective emotions into a simple continuous scale. One of these early models comes from Russel *et al.* [84], which imagined affect as a grid with two dimensions, valence, and arousal (see Fig. 2.2). *Valence* is associated with the pleasantry of an emotional state; *i.e.,* positive valence would be described as happy or enjoyable emotions, while negative valence would be sad or unpleasant emotions. *Arousal* is associated with the intensity of emotion; *i.e.,* high arousal would be described as being surprised or very focused, while low arousal would be feelings of boredom or sleepiness. Engagement—a topic discussed at length already, might be thought of as a combination of positive valance and high arousal. There are a number of datasets of physiological data which use variations of Russell's Affect Grid for self-reported scores of emotions from participants exposed to different stimuli. Some of these include ASCERTAIN [96], AMIGOS [61], DECAF [1], DEAP [49], and MAHNOB [95].

Another prominent model is designed by Mehrabian [60]. The Pleasure-Arousal-Dominance (PAD) model builds off of Russell's Affect Grid to add a third dimension, pertaining to the dominance the user feels. This is meant to help differentiate

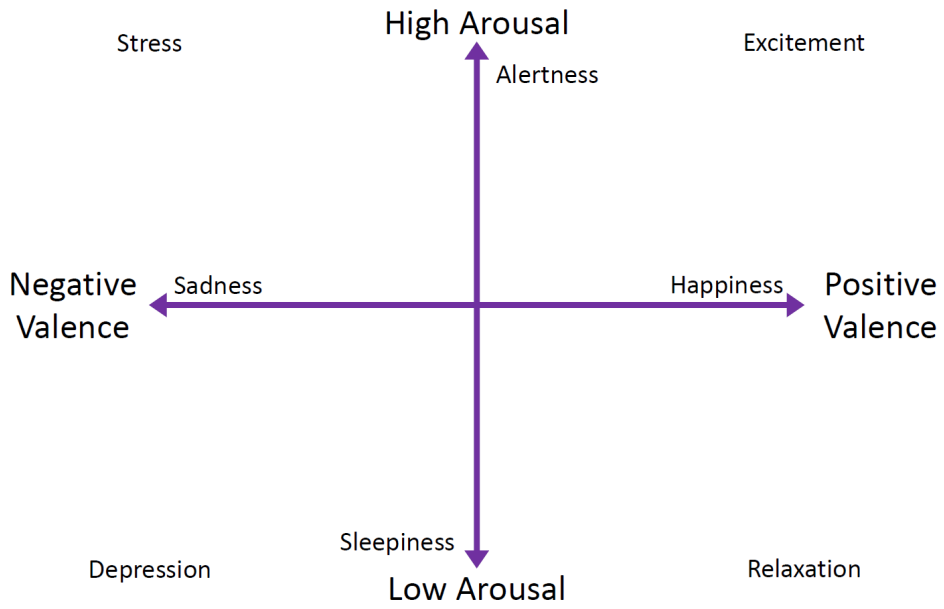


Figure 2.2: The Affect Grid. The x-axes relates to valence, or the perceived pleasure of the state; while the y-axes relates to arousal or the perceived intensity of the emotion. Design inspired by Russell *et al.* [84].

between very similar emotions such as excitement, which according to PAD is characterized as high pleasure, high arousal, and low dominance; from engagement which is characterized with the same pleasure and arousal, but with high dominance.

2.3 Physiologically Adaptive Systems

Psychophysiology is the field that studies the relationship between subjective mental states and objective physiological processes [10]. Psychophysiological states (also described as affective states previously) include experiences such as engagement, frustration, anxiety, and cognitive workload. Many metrics have been effectively used for affective state prediction, such as electroencephalography (EEG), electrocardiography (ECG), electrodermal activity (EDA), photoplethysmography (PPG), and pupillometry. The use of these physiological signals allows for data perception in (ranging levels of) non-invasive measurements through wearable devices. These measures ultimately could provide robots the ability to infer users' affective states through quantitative approximations of subjective feelings. This additional dimension of perception may enable a more natural and personalized HRI experience [71].

Physiologically adaptive response technology, according to Loewe and Nadj [54], can be broken up into three common categories: state display, assistance offering, and challenge adaptation. Their review of 44 articles provides an excellent summary of the state-of-the-art, and seven suggestions for future directions researchers might pursue for novel research. Their review provides analysis on the different contexts, signal types, algorithms, targeted states, and system adaptation types. As well, the benefits of incorporating physiological computing, specifically for HRI research has been discussed in a review by Roy *et al.* [83]. The authors discuss the need for online, real-time classification of operators' mental states to provide both operational safety and improved performance. This review has less to do with social robotics, however, it provides a welcome discussion into the need for inferences of robotic systems into human collaborative environments. The benefits of a closed-loop system where a robot is informed about the specific psychophysiological states of users with whom it is interacting can provide real-time adaptations in behavior to the benefit of the user.

2.3.1 Physiological Signals

We've described quite a few different types of physiological signals already. Not to overwhelm the scope of this thesis, we will focus primarily on ECG, as it has an extremely low degree of invasiveness and provides a data type from which a significant amount of knowledge can be extracted [39]. As seen in Fig. 2.3, an ECG has four points of interest for our discussion, namely the P-wave, R-peak, T-wave, and QRS complex. Heart rate variability (HRV) encompasses an important set of metrics that aim to explore the temporal dynamics of the heartbeats. HRV most importantly can be used to describe many psychophysiological states; including stress, cognitive workload, and engagement [48].

HRV specifically refers to the changes in the time interval between consecutive heartbeats. HRV can be computed through the time variability of consecutive R-peaks (also called the R-R interval—RRI). Shaffer and Ginsberg [90] provide an extensive overview of HRV metrics, however a few important time-domain features that we'd like to highlight include the following: SDNN, the standard deviation of the interval between successive heartbeats; SDRR, the standard deviation of the RRI; RMSSD, root-mean-square of successive RRI differences; pNN50, percentage of successive RRI that differ by more than 50 milliseconds. The frequency-domain of an ECG also provides useful features for describing affective states [99]. Some of the more useful frequency-domain features include the analysis of the absolute power of

high-frequency (HF) bands (0.15 to 0.4 Hz), low-frequency (LF) bands (0.04 to 0.15 Hz), and very-low-frequency (VLF) bands (0.0033 to 0.04 Hz). HRV features have been shown to be helpful for inferring complex emotions [75] and affective states [23]. For instance, stress levels of human participants can be defined in terms of variations of the ratio of low-frequency and high-frequency power (LF/HF) [99].

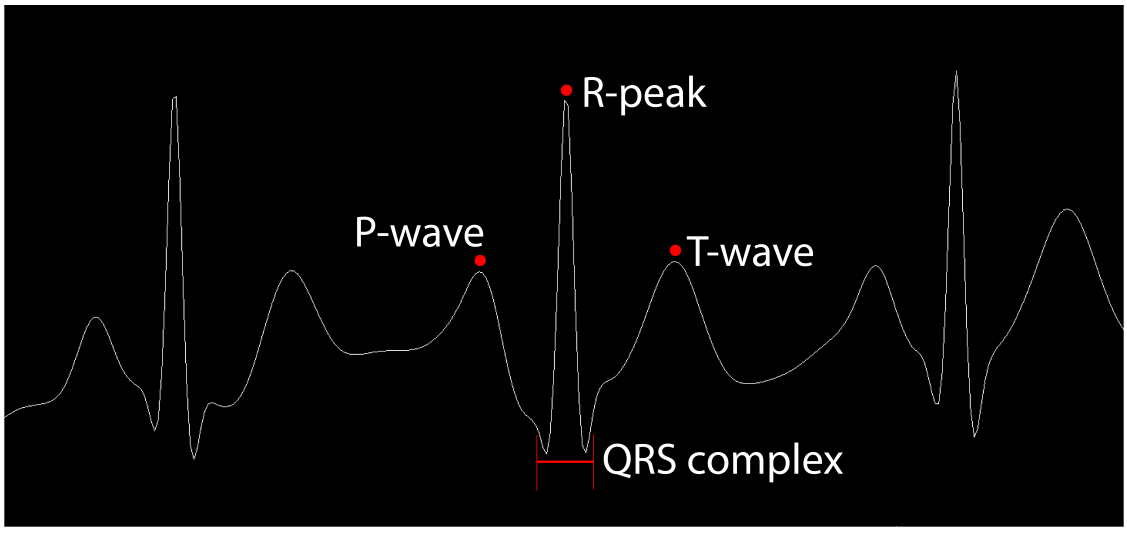


Figure 2.3: A simulated electrocardiogram (ECG) to show the structural features of the signal.

2.3.2 Use in Research

Unsurprisingly, the use of physiological data can be quite messy, as recorded signals can be filled with artifacts from movement, and electrical noise generated from surrounding electrical infrastructure. Filtering of the time-series data can be applied, but this doesn't immediately tell you about the contributor's affective state. Correlations can be made through classical machine learning techniques. Rani *et al.* [76] empirically studied the application of four different classical machine learning techniques for affect recognition in two cognitively demanding tasks, solving anagrams and playing Pong. The authors analyzed the success of K-Nearest Neighbor classifiers, Regression Trees, Bayesian Networks, and Support Vector Machines on recorded physiological measurements from 15 participants. The authors provide discussion as to how these closed-loop scenarios can be extended towards HRI scenarios.

While classical approaches perform well, the advent of deep learning challenges these methods. Faust *et al.* [30] reviewed 53 articles relating to the application of deep learning with physiological signals in health care scenarios. The authors explored the recent influx of articles published regarding deep learning and physiological data for classification tasks. In many recent publications, deep learning methods appear to be outperforming classical approaches when provided with large and diverse datasets. The review describes how 31 (of the 53) articles used data from one or more freely available datasets. The ability to generalize well to never before seen participants, especially in the case of using this data in an online-adaptive system, is an extremely valuable asset.

In HRI research, there have been a number of physiologically adaptive robotic scenarios that were developed. To complete this section, we highlight a few scenarios. Itoh *et al.* [43] developed an adaptive stress reduction system by using HR and HRV in a scenario involving a humanoid robot. In their experiment, the robot offered a hand-shake to participants who were stressed. They found that participants' stress levels, measured using HRV metrics, were reduced when the robot offered a hand-shake with an adaptive policy. Corrigan *et al.* [15] developed an adaptive system to detect the engagement of participants while interacting with the NAO robot. They highlighted that using physiological signals (EDA in this case) with post-experiment surveys provides a better internal-state model.

Shao *et al.* [92] designed non-verbal affective behaviors for the Pepper robot to evoke particular emotions in the participants. Users' affective state was approximated using surveys and EEG. This study showed a consistent relationship between the collected EEG data and self-reported valance and arousal levels. Furthermore, this study used the collected EEG data to train both a support vector matching and neural network, which found promising accuracies in classifying users' intended affective states. Similarly, Guo *et al.* [38] designed five emotional behaviors (joy, fear, neutral, sadness, and anger) for a small form humanoid robot called Alpha 2, using combinations of head, arm, and leg actions. The authors then examined how participants reacted to these emotional expressions through user-reported valence and arousal levels. EEG and pupillometry were recorded and analyzed to verify the success and intended responses of their designed robotic affective expressions.

2.4 Existing Tools

In this final section, we provide a brief overview of some of the many tools which exist to facilitate the development of physiological computing. This is not a comprehensive list of all existing tools, but some notable ones which have inspired us in our design of the framework and software library described in the following chapters.

In order to make use of psychophysiological measurements in HRI, a way to first acquire the data in real-time to interface with the robot is necessary to producing closed-loop interaction between robots and humans. EventIDE[2] is a commercial software solution for real-time capture and processing of physiological data with support for a wide range of physiological sensors. Another similar commercial software is iMotions[3] which is tailored extensively towards research in user-experience, emotion recognition, and mental workload.

BITalino [16] is a multi-parameter "do it yourself" (DIY) kit. This Arduino-based hardware toolkit allows for easy acquisition of ECG, EDA, EMG, and PPG data while having a relatively affordable cost. Benchmarking studies [6] have been performed to compare performance with research-grade biomedical devices from BioPac[4]. Another data acquisition solution is the FlyLoop [72], a lite-weight framework to assist users to easily fuse physiological signals coming from different devices such as the Muse headband, Tobii Eye Trackers, and Apple Watch.

Game development is a field where understanding how engaged a person is with your product can make or break your project. Thus, various software tools have been developed to provide ease of use for affective state prediction from physiological data, developed specifically for this domain. OpenViBE [80] is an open-source software platform that allows real-time processing of EEG. The platform offers various tools to integrate these signals into VR applications and games. PhysSigTK [78] is a toolkit for the Unity3D game engine designed around providing easily accessible real-time signal data for engagement experiments. The Biocybernetic Loop Engine (BL Engine) [65] provides an interface for designing physiologically adaptive scenarios in the Unity3D game engine.

There has been a great deal of recent work to streamline the development of complex HRI scenarios, and to fill in gaps between physical and social HRI. iCub-HRI [31] is a software framework that integrates various components of the iCub ecosystem.

[2] http://www.okazolab.com/
[3] https://imotions.com/
[4] https://www.biopac.com/

The framework also provides a 3D context-aware visual perception system, visual-tactile reactive controller, and the ability to store symbolic knowledge [68]. Additionally, Robotic Coach Architecture for Elder Care (ROCARE) [27], is another software architecture designed for adaptive, multi-user measurement of engagement in assistive living scenarios. The architecture features an easy to use GUI and utilizes audio, visual, EEG, and EDA measurements.

Chapter 3

HRI Physio Lib

This chapter describes the development of an open-source, integrative and modular software library created to facilitate the design of physiologically adaptive HRI scenarios. The `HRI Physio Lib` helps to streamline the acquisition, analysis, and translation of human body signals to be used as additional dimensions of perception in HRI applications using social robots. Through exploring the designs of previously developed systems, we can better provide a general structure and multi-purpose tools to help facilitate the integration of physiological signals into HRI applications.

The software framework has four main components: signal acquisition; processing and analysis; social robot and communication; and scenario and adaptation. Information gathered from the sensors is synchronized and processed to allow designers to create adaptive systems that can respond to detected human states.

Work in this chapter was motivated by two research questions which asked:

RQ1: How can we design a robotics software framework for the rapid development of physiologically aware robots?

RQ2: How can we develop a robotics software framework capable of integrating various communication libraries?

The goal of this software framework was to provide a comprehensive, integrative, and openly accessible set of tools aimed at facilitating the integration of physiological signals into HRI scenarios. The source code itself is hosted on GitHub[1].

[1] https://github.com/kothiga/hri-physio

3.1 Framework

Three key facets were carefully considered during the design of the framework. The first was related to the design of a system that is modular in nature to allow for easy integration and the addition of new sensors. The second being to develop a system that is flexible as to support multiple robot systems and middle-wares (*e.g.,* ROS, YARP). The last was related to the system being open-source with nonrestrictive licensing.

The software framework can be deconstructed into four main components: signal acquisition, processing and analysis, social robot and communication, and scenario and adaptation. We'll go through each of these building-block components describing their purpose in the larger framework, as well as provide an introduction to related tools, of which have a complete description later in section 3.3.

3.1.1 Signal Acquisition

Wearable technologies equipped with the ability to capture and analyze physiological data are becoming ubiquitous in daily life for many people [55]. The maturity of this technology in the domain of research and commercialization is still young; however, interest has exploded in recent years, so much so that in a 2019 study by the Pew Research Center [101] reported that over 1 in 5 U.S. adults said they regularly wore a smart-watch or fitness-tracker.

With so many different sensing technologies available; each running its own unique communication protocols it can be a daunting challenge to try and incorporate them into existing robotic frameworks. In an ideal scenario (security and privacy aside), it would be advantageous to subscribe to the raw data stream transmitted from the device (typically using Bluetooth Low Energy) to use the real-time readings of cardiovascular or accelerometer data within an intelligent system. To try and bridge the communication gap, we've chosen Lab Streaming Layer[2] (LSL) as our primary mode of signal acquisition.

LSL is a general-purpose, cross-platform communication library designed around providing an easily accessible stream of time-domain data (signal data from lab instruments, cardiovascular data, audio, *etc.*) onto the local area network (LAN). This decision was made as LSL has become a widely popular software library, with an open-source community and vendors providing projects developed for a large

[2] https://labstreaminglayer.readthedocs.io/

Figure 3.1: Polar heart rate monitors. (left) Polar H10, a chest strap device capable of transmitting ECG, HR, RRI, and accelerometer data over Bluetooth. (right) Polar OH1, an armband optical device capable of transmitting PPG, HR, PPI, and accelerometer data also over Bluetooth.

range of supported devices[3]. A majority of what has been developed relates to electroencephalogram (EEG) systems, however, other exciting projects have brought support for streaming data from the Tobii Eye trackers, as well as the Microsoft Kinect, and even human interface hardware like Nintendo Wiimotes controllers.

The signal acquisition component primarily relates to tools for collecting physiological signals and bringing them into the folds of robot systems. We developed a general-purpose bridge tool called the `Physio Receiver` capable of receiving streamed data from LSL, logging, buffering, and translating out onto ROS, YARP, or even LSL again. This tool is later expanded on in section 3.3.1. The tool is data agnostic (supporting different `C++` standard data types) and can receive streams from not only LSL, but also ROS and YARP as a result of its polymorphic design.

As well, we've developed two tools for streaming data from the Polar fitness trackers, specifically the chest strap heart rate monitor *Polar H10* and the armband optical heart rate monitor *Polar OH1*. These devices can be seen in Fig. 3.1. The Polar sensors were chosen as our first supported devices for their accuracy to sense cardiovascular activity [35], low-cost, commercial availability, general comfort of wearing [39], and most importantly ability to interface data from. The first of these

[3] https://labstreaminglayer.readthedocs.io/info/supported_devices.html

tools was a desktop application called `Physio Streamer` which is described later in section 3.3.3, and the second was an Android application called `Polar Streamer` which is also described later in section 3.3.2.

3.1.2 Processing and Analysis

The processing and analysis component primarily deals with methods that allow transforming the raw data from both humans and robots into interpretable features. In the `HRI Physio Lib`, we have included multiple existing and scientifically validated processing algorithms to process the body signals and extract the most used features.

For analyzing the cardiovascular data, we have included scripts for signal processing, outlier detection, and HRV analysis using NeuroKit2, an open-source `Python` toolbox [57]. This toolbox allows for the transformation of an ECG signal into RRIs which subsequently can be used to find various HRV features. In the time-domain this can include the SDNN, RMSSD, and pNN50. The frequency-domain features that can be acquired include LF, HF, and LF/HF ratio.

The module which uses the NeuroKit2 toolbox is called the `NeuroKit Processor` and is described later in section 3.3.4. In short, this module is a command-line interface that receives a specified LSL stream containing time series ECG data to be processed before streaming it back out onto LSL. The processed cardiovascular data, namely HRV features are what is most important for inferring complex human states such as emotions [75].

It is important to maintain common timestamps of data to better infer causality of physiological responses, *i.e.,* what specific feedback of the robot may have lead to the desired responses. Synchronization of recorded data is key in order to make these inferences possible. It is for this reason that we maintain the LSL timestamps throughout all transmissions.

We've also developed software to aid in the development of user-defined processing pipelines. A common requirement for this type of time-domain data is a way to clean signal artifacts produced by power line interference (50/60 Hz), electrodes location (*e.g.,* cross-talking), general movements, and other noise; digital filters should be applied to the raw data [46]. We've included in the `HRI Physio Lib` a suite of recursive Butterworth filters (*e.g.,* low-pass, high-pass, band-pass, band-reject) based on an outline from Dodge and Jerse [20] to assist in the data cleaning. As well, we've implemented methods for producing the Hilbert transform and spectrogram of

a signal using the popular lightweight Fast Fourier Transform library PocketFFT[4]. Other general mathematical methods can also be found in the software library.

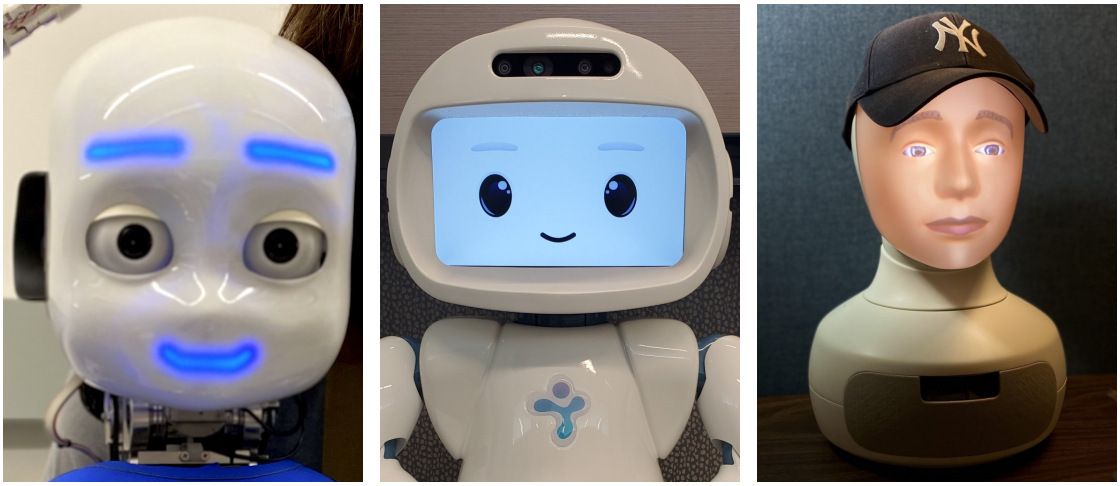
3.1.3 Social Robot and Communication

The social robot and communication component primarily has to do with communication with the socially interactive robots. To create physiologically adaptive HRI experiences, a set of robot states and capabilities should be previously defined. We've outlined six general feedback modalities which we believe to be common across most social robotic platforms. They are: screen-based, speech-sounds, vibrations, lights, facial expressions, body language & movement. Feedback modalities such as facial expression are discrete in that you have a limited number of appearances (*i.e.,* happy, calm, sad), while other modalities like lights can be thought of as continuous (range of values). This is of course not categorical, and completely dependent on the robot being developed around; *e.g.,* it may be the case that a particular robot only has *on* and *off* for some light feedback.

A way to describe screen-based feedback might be the use of playing pre-rendered video, images, or even something complex such as a game using some form of a flat-panel screen integrated with the robot; take for example the touch screen display on the chest of the social robotic platform Pepper from SoftBank Robotics[5], or the LCD display which is the face of the QT robot from LuxAI[6]. Other ways to use screen-based feedback might be through a peripheral device separate from the robot such as a computer monitor, television, or even a commercial tablet.

Speech-sounds might be thought of as any principal method of verbal communication which a robot is capable of. This could include anything from minimal "beeps" and "boops" to complex speech synthesis with emotional embeddings capable of tone and frequency modulation. There are a number of avenues of producing amiable speech with today's advancements; from premium services such as Acapela[7] and Amazon Polly[8] (which also offers a free tier) to open-source solutions like gnuspeech[9].

[4] https://gitlab.mpcdf.mpg.de/mtr/pocketfft/-/tree/cpp
[5] https://www.softbankrobotics.com/emea/en/pepper
[6] https://luxai.com/
[7] https://www.acapela-group.com/
[8] https://aws.amazon.com/polly/
[9] https://www.gnu.org/software/gnuspeech/



(a) iCub humanoid robot (b) QT robot (c) Furhat robot

Figure 3.2: A few examples of the many different ways facial expressions can be represented in social robots. (a) Multi-color LEDs under the face; (b) Screen-based face display; (c) Backlit projection onto the faceplate.

Vibration-based feedback is likely to be one of the least common modalities found on social robots, however, if present is to be expected in the form of one of two basic types of vibration motor, being either an eccentric rotating mass type or a linear resonant actuator type. In some cases, the usage of this feedback may be a discrete *on* and *off* or it may possess some continuous range of values that represent intensity.

Lights have classically been used as a medium for feedback in HRI. LEDs have previously been used to give robots an ambient-like display of low cognitive load information communication, allowing for glances and unintrusive contextual observations to inform the human of the robot's internal state (*e.g.,* low battery, task in progress, available for use) [79]. Light-based feedback has been observed by Kayukawa *et al.* [47] to have a significant effect on the decision-making of participants and effect their impressions of the robot, in an experimental prisoner's dilemma-like game. In this study different colored lights were used in the eyes of the NAO robot from SoftBank Robotics[10] to convey the *emotional* state of the robot (yellow: joy, red: anger, blue: sadness, pink: shame).

Facial expressions are likely to be specific and highly different between robot

[10] https://www.softbankrobotics.com/emea/en/nao

platforms. As an example, the iCub humanoid robot[11] has a set of LEDs under its faceplate which gives the perception that it has both a mouth and a set of eyebrows which can be set to one of 16 predefined colors 3.2a. As well, there are a number of redundant sets of LEDs that can give the imitation that the iCub is smiling, frowning, excited, confused, *etc.*. Another robot platform like the QT robot previously described uses an LCD display and plays pre-rendered `avi` video files to give the perception of smiling, laughing, talking, and more 3.2b. The Furhat robot has a revolutionary design for its facial expressions, allowing for extremely complex and detailed designs 3.2c. This robot uses a wide-angle light projection inside the head to display pre-rendered faces onto the back of the faceplate which shines through to give the perception of a realistic face. Regardless of how the robot is designed, facial expressions are an important aspect to consider in the experimental design.

The robot's body language & movements are the last feedback modality which we believe is the most important feedback for social robots. The levels of embodiment vary widely among robotics platforms, however, the use of the robot's body in communication with others is paramount in producing positive affinity when interacting with people [59]. It has also been shown in a study by Li and Chignell [52] that prerecorded gestures on a plush bear robot can be used to express *emotional* states through simple head and arm movements. Any sort of actuation of the robot can elicit affective responses from a person in an interaction scenario [92], so it is important to consider this greatly in the design. The way to interface with the robot's motor systems as well varies widely. For robots such as the QT robot which contains a total of 8 degrees of freedom (2 in the head, and 3 in each of the two arms) there exist two ROS interfaces, one for controlling specific joints by providing the exact degree of the joint, and a second which allows for recording and playback of motors; in the recording state, the motors unparked allowing for complex movements of the robot. Other robots such as the iCub offer similar motor interfaces to adjust its 53 degrees of freedom to specified degree positions, as well as provides an interface for controlling the arms and legs in Cartesian (operational) space.

As discussed later in section 3.2.2, the `HRI Physio Lib` contains an abstract class called `robotInterface` which can be used as the base to develop a concrete controller for adaptive scenarios. This abstract class contains numerous virtual functions that can be used as a starting point to build from or allows for polymorphic designs such as reusing a scenario with a different robot.

[11] https://www.iit.it/web/icub/

3.1.4 Scenario and Adaptation

The scenario and adaptation component primarily is responsible for merging the data to create individual scenarios for HRI experiments. As described by Fairclough and Gilleade [25], the creation of physiologically adaptive systems require a careful design process to ensure the adaption itself is "perceived as accurate, timely, intuitive and does not have any unintended consequences". The incoming physiological data must be interpreted and merged with the robot's feedback to produce responsive applications.

In order to make compelling adaptive scenarios, experimenters first need to define two important elements to the context of their closed-loop system. The first is what are the targeted human states described in terms of the physiological descriptors (*e.g.,* stress defined in terms of HRV metrics). The second being what behaviors are possible in terms of the agent's capabilities (*e.g.,* emotional expression through lights, voice, movements).

Designers of physiologically adaptive systems need to be aware of the specific body metrics and ranges used to describe the desired human states and use them accordingly for each scenario. As well, routines for the robot's behavior must also be specifically tailored to elicit desired responses from a human participant. For instance, a scenario centered around participant relaxation using a social robot can play defined routines such as pleasant background music, makes slow stable gestures, and speech to encourage breathing patterns that are beneficial to induce a relaxed state. Combining these two aspects with some form of a decision loop allows for real-time adaptations of the robot's behaviors.

To summarize, the framework provides a general structure and multi-purpose tools to help facilitate the integration of physiological signals into HRI applications. Later in Chapter 4 we go on to describe the implementation of a use case using our framework related to an exercise scenario mediated by a social robot.

3.2 Architectural Structure

If the framework is the tools we use to build constructs, the architecture is the blueprints that better define the software. IEEE 1471 [56] defines *architecture* as:

> "The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution."

In this next section, we will begin to descend into the details of what has been implemented, and how extensions of the framework can be used to develop further custom physiologically adaptive scenarios.

3.2.1 High-Level Architecture

The high-level architecture can be seen in Fig. 3.3. It is divided into the four primary components of the framework discussed in the previous section. The loop detailed in the figure begins first with the person wearing some sort of sensor to acquire physiological data. In our implementation for the *Signal Acquisition*, we first chose to focus primarily on an ECG chest strap (namely the Polar H10). For the later described `Polar Streamer`(in section 3.13), the *Data Receiver* was a software development kit (SDK) offered by the manufacturer for developing applications on mobile device platforms (Android and iOS). This SDK provided a good interface for connecting to the Polar device wirelessly to acquire a variety of measurements include the raw ECG stream, accelerometer data, heart rate data (measured in beats per minute), as well as computed RR intervals. Received data is prepared to be appropriate for sending out onto Lab Streaming Layer (LSL).

For future implemented devices the mode for the *Data Receiver* may be something completely different. For example, the Shimmer suite of devices can be interfaced by connecting directly over Bluetooth. They can also be queried through the local database (`db` file) created by Shimmer's proprietary device management application Consensys when requesting to log streamed Bluetooth data. As well, LSL is not the only possible end output for the *Signal Acquisition*, other middleware such as ROS and YARP could be used instead as the *Data Stream Outlet*. LSL was chosen as the primary mode of sending and receiving sensor data, as a number of open-source applications have already been developed to stream physiological data over LSL (previously discussed in section 3.1.1). The ease of using ROS and YARP has been made possible through an abstract design called the `StreamerInterface` and concrete implementations for the respective middlewares. This is further described in section 3.2.2.

Raw physiological signals are too noisy to describe psychophysiological states reliably for use in HRI experiments. A well-structured and careful process of signal filtering and feature extraction must be conducted to successfully transform physiological signals into usable descriptors and markers capable of representing the psychophysiological signatures of human responses [26]. To help facilitate the creation

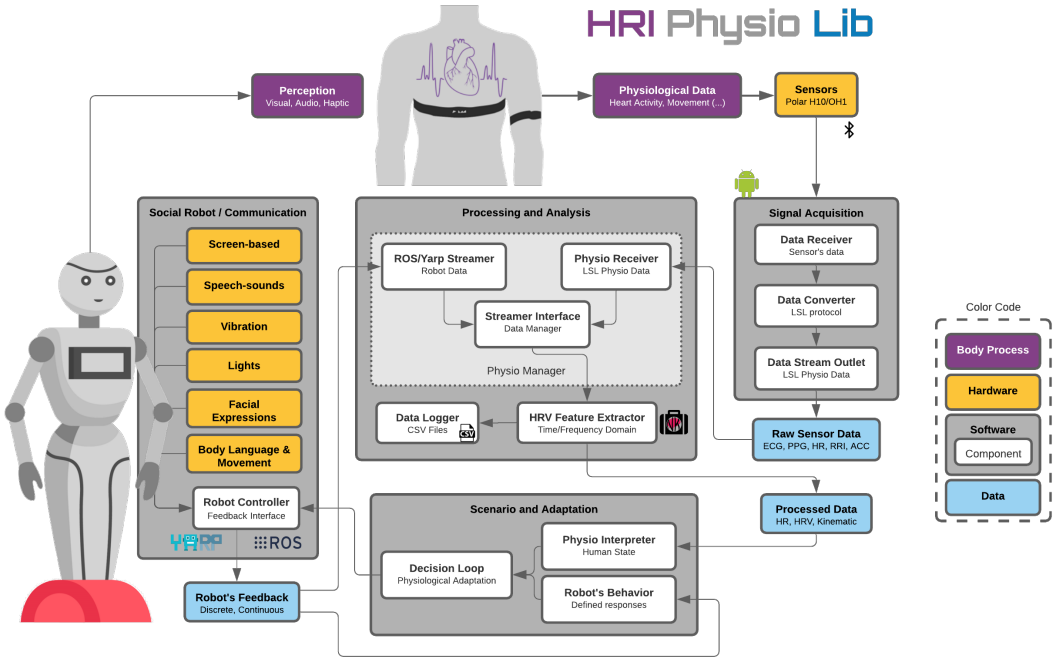


Figure 3.3: Overview of the `HRI Physio Lib`. Color code explanation of the categories per each component as follows: *purple*–body process, *yellow*–hardware elements, *grey/white*–software elements with their individual components, and *blue*–data or information flow.

of these pipelines we've provided implementations of common signal processing algorithms. Many of these signal processing algorithms require a fairly large window of data to properly function. To assist in this, a module has been created called the `Physio Receiver` (described in depth later in section 3.3.1) which acts as both a bridge between middlewares (*i.e.*, can receive data from LSL and transmit it back out onto ROS), as well as provides a simple way to efficiently log, buffer, and window received data. When a variable amount of data has been buffered, the `Physio Receiver` will then output a frame of data onto the specified network while keeping a variable number of samples from the end of the frame to be used as the beginning of the next. The primary reason to decouple data acquisition and processing is to provide a simple asynchronous design that ensures that data doesn't become lost should processing take longer than expected while new data samples arrive. The `Physio Receiver` is data agnostic and provides a needed level of separation of responsibilities in a modular system.

Continuing with the initial focus on ECG data, we've developed a processing module called the `NeuroKit Processor`, which leverages the previously discussed `Python` toolbox NeuroKit2. This module receives ECG data from a specified LSL stream, buffers the data, windows and processed it with a total of nine outputs LSL streams. These nine outputs include a cleaned ECG stream (of the same length as the specified frame size), a binary array (of the same length as the frame size) containing a 1 in the indices if an R-peak is present, and a single value representation for the mean heart rate of the frame. The remaining outputs are also single-valued samples that represent the HRV feature of the processed frame. They are the: SDNN, RMSSD, pNN50, LF band, HF band, and LF/HF ratio. This module is meant to be a simple feature extraction tool, and while it doesn't rely on the `Physio Receiver` for buffering its data, it isn't doing anything too computationally intensive as to result in lost data.

Processed data can now be passed to the *Scenario and Adaptation* component. The primary piece of this is the *Decision Loop* which acts as the primary driver for a scenario and feeds commands to the (yet to be described) *Robot Controller*. We describe later in Chapter 4 an exercise-based scenario to clearly convey how these components for a physiologically adaptive scenario might be implemented. The *Physio Interpreter* and *Robot's Behavior* components will be fairly contextual to the scenario which is being designed. For the exercise scenario we later describe these two components were quite minor, while for more complex scenarios aimed at approximating difficult human states and how the person in-the-loop responds to feedback from the robot, they will play a more significant role.

The final component deals with controlling the robot, namely the *Robot Controller*. Similar to the abstract design of the `StreamerInterface`, we developed an abstract `RobotInterface` with the purpose of being a template to which concrete implementations will follow. As described in section 3.1.3, we've outlined six feedback modalities that we believe to be common across most social robotic platforms. The abstract controller was used in the exercise scenario to develop the `QT Controller` (described in section 4.2.3). The benefit of pursuing an abstract design is related to polymorphism and the reusability of software. The software design pattern on which these two interfaces are based off is the *bridge* pattern, which aims to allow implementations to vary independently from the abstraction. Gamma *et al.* [33] explains the bridge pattern as being useful when:

> "... you want to avoid a permanent binding between an abstraction and its implementation. This might be the case, for example, when the implementation must be selected or switched at run-time."

This can be particularly useful if at a later point it is decided a different robotics platform should be used, or to reuse the same *Robot Controller* for a different future scenario. In practice though, implementations are likely to vary widely and be explicitly tailored towards the absolute needs of the robot with the context of the scenario.

3.2.2 Low-Level Architecture

In this section, we will go one layer deeper and detail the low-level architecture of the `HRI Physio Lib`. The software library itself is written in `C++` and is broken up into six *namespaces*, which act as identifiable organization groupings.

Core

The first namespace, *Core*, contains a set of generic implementations of data structures that are used elsewhere in the library. These data structures can also be used to help designers in building their own implementations. Currently, there are two classes in this category. The first is a simple `Graph` class, which could potentially be used for keeping track of state transitions in an experimental scenario.

The second is a template class called `RingBuffer`, which is used frequently throughout the library as a means of efficiently buffering information to later be purposed. The advantage of using this container over something like a `std::vector` or `std::list`, is really rooted during the enqueue and dequeue operations. Since we're expecting a lot of adding of data to the buffer, followed by large chunk removals, it is important for us to ensure a good level of performance and cache locality as to not cause unnecessary inefficiencies. For `std::vector`, it can be costly to *remove* items from the front, as all remaining elements are copied sequentially to the front. For `std::list` it is easy to remove elements from anywhere in the data structure, as the implementation is built using pointers, however, we are likely to lose performance due to poor stride locality of the stored data.

A ring buffer is conceptually a circular array; its implementation however is a flat array with a pointer to the head and tail (seen in Fig. 3.5) to indicate where the data begins and ends. The only drawback is the size of the buffer (allocated space) is not dynamic, meaning any requests to resize or expand the buffer requires copying all elements to new internal storage. This however can be averted by simply constructing with an appropriate amount of storage to the context. This implementation was

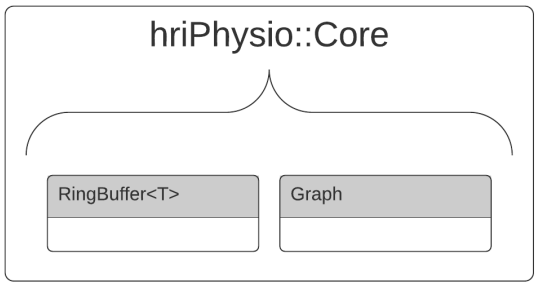


Figure 3.4: The Core namespace. Shown are the `RingBuffer` and `Graph` objects which are generic data structures that are part of the software library. Future generic implementations of useful data structures will be found here.

designed with the expectation that reading and writing to the data would need to be done asynchronously. We've ensured these methods are atomic in nature through the use of a `std::mutex` lock. Another advantage of our particular implementation is when needing to dequeuing; we've added a feature to allow for overlapping chunks of data through a *sliding window*, *i.e.,* when removing data from the buffer, there is an optional argument to keep a varied number of samples at the end.

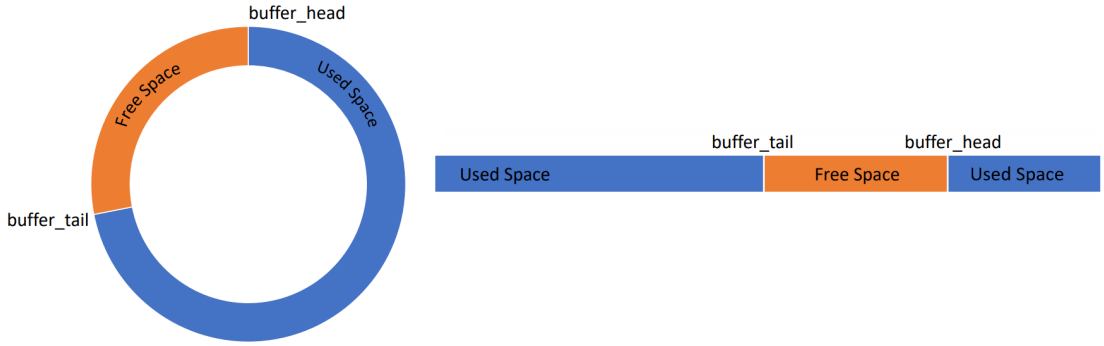


Figure 3.5: (left) The conceptual idea of a ring buffer storage. (right) The actual implementation of ring buffer storage. A pointer to the head and tail float along a flat array to indicate where the start and end indices are. The last index of the flat array precedes the first index to form a seamless ring shape.

Factory

The *Factory* namespace is a location for implementations based on the factory abstract design [33]. The idea of the factory pattern is to simply consolidate the creation of polymorphic objects without explicitly stating their concrete type. This abstraction is exposed to the user through enumeration or simply a string-based request. The benefit of utilizing a factory pattern is to bring both scalability and readability to the interface.

Currently, our factory namespace has only one class, called the `StreamerFactory`, which is used to produce the concrete implementations of the `StreamerInterface`. The benefit for us is that if we choose to support an additional mode of streaming data (*e.g.,* UDP socket, Bluetooth socket, MQTT[12]) all we'd need to do is add the new concrete implementation into the possible return types of the factory, and automatically any module using the `StreamerFactory` to produce derived classes of `StreamerInterface` will be able to utilize said new concrete streamer.

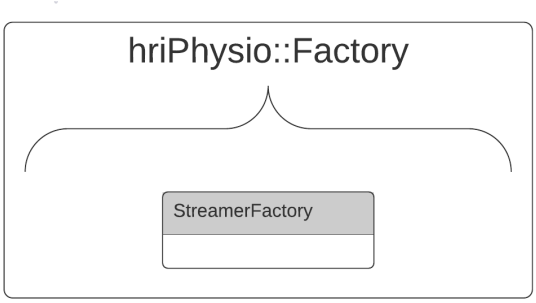


Figure 3.6: The Factory namespace. Shown is the `StreamerFactory`, which implements the factory software design pattern to abstract the process of instantiating objects derived from the `StreamerInterface` found in the stream namespace. Future implementations of factory-like objects will be found here.

Manager

The next namespace is called *Manager*, of which we have three classes. To begin we'll describe the `ThreadManager` class, a relatively simple design to abstract away the complexities of producing asynchronous behaviors, and create a reusable interface

[12] https://mqtt.org/

to which future driver like applications can inherit from. The `ThreadManager` has a method for spawning a new thread for a given function. This manager keeps a reference to the spawned thread so that when it's time to end the program it can clean everything up nicely. This manager also has a function which derived classes can utilize to check each loop for if the manager is still running, or should it stop itself.

The `PhysioManager` is the second class within the namespace. This class is derived from the `ThreadManager`, and is inspired by the producer-consumer pattern [44] since it is encapsulating two concrete implementations derived from the `StreamerInterfaces`. One of these streamers was used for receiving data (producer), while the other was used to send data (consumer). This manager also encapsulates the previously described `RingBuffer` object, which ensures atomic reads and writes of the buffer. Both the input and output streams ran on their own threads which the manager spawns. When the input stream received a new chunk of data, it first writes a copy of the data and its timestamps to a `csv` file prior to writing the data to the buffer. The output stream will periodically check to see if the buffer has accumulated a set minimum number of samples. When this has occurred it requests to dequeue the data.

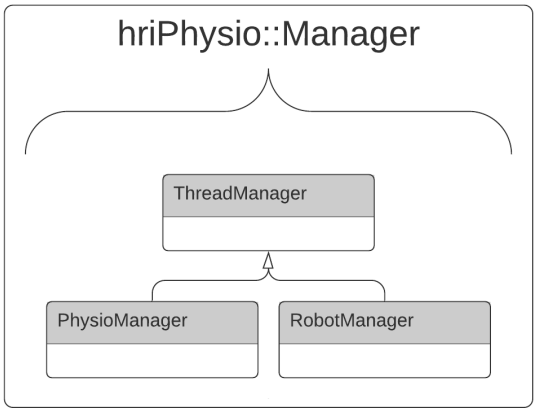


Figure 3.7: The Manager namespace. Shown are the `C++` manager classes which have been designed to *handle* and manage objects and functions which they encapsulate. In the case of the `ThreadManager`, it receives functions to spawn threads for, while managing their lifetime. The `PhysioManager` and `RobotManager` objects have the purpose to encapsulate polymorphic objects which are passed to them and manage their individual behaviors. White arrow heads represent inheritance.

The last class within this namespace is the `RobotManager`, whose purpose is to manage concrete implementations of the `RobotInterface` class. This manager is also derived from the `ThreadManager`, and uses the inherited methods to spawn threads for the `RobotInterface` that need to be run asynchronously (such as the robot's command receiving methods and of overridden an internal loop for the robot). This manager simply provides a convenient reusable interface to wrap a concrete robot controller in.

Processing

The namespace *Processing* contains a number of digital signal filtering methods. In the future, we'd like to expand the suite of implemented algorithms, but for now, we currently have five main methods implemented, which can be utilized for data cleaning and extraction.

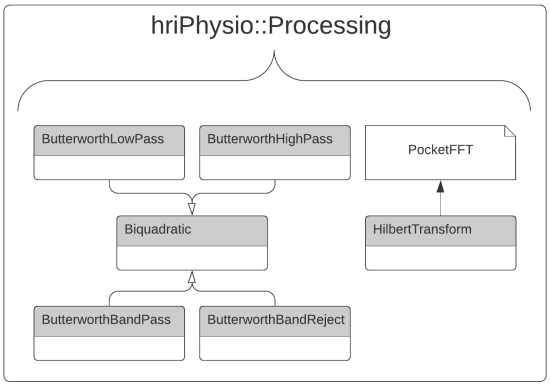


Figure 3.8: The Processing namespace. Shown are the `C++` processing algorithms that have been developed so far. Future implementations of processing methods will be organized into this namespace. Black arrowheads represent composition relationships, while white arrowheads represent inheritance.

The first class in the list is an abstract class used by four of the implemented methods. This abstract class is called `Biquadratic` and contains the necessary methods for implementing second-order infinite impulse response (IIR) filters. The only method required to be overridden by derived classes is the `updateCoefficients` method. The `Biquadratic` abstract class performs a bilinear transformation on a chunk of received data with the specified coefficients. Its design is based on an outline from Dodge and Jerse [20]. Derived from this abstract class is a suite of recur-

(a) Butterworth low-pass filter

$$
\begin{aligned}
C &= \frac{1}{\tan\left(\pi \, \frac{f_{cut}}{f_{rate}}\right)} \\
a_0 &= \frac{1}{1 + \sqrt{2}\, C + C^2} \\
a_1 &= 2\, a_0 \\
a_2 &= a_0 \\
b_1 &= 2\, a_0 \left(1 - C^2\right) \\
b_2 &= a_0 \left(1 - \sqrt{2}\, C + C^2\right)
\end{aligned}
$$

(b) Butterworth high-pass filter

$$
\begin{aligned}
C &= \tan\left(\pi \, \frac{f_{cut}}{f_{rate}}\right) \\
a_0 &= \frac{1}{1 + \sqrt{2}\, C + C^2} \\
a_1 &= -2\, a_0 \\
a_2 &= a_0 \\
b_1 &= 2\, a_0 \left(C^2 - 1\right) \\
b_2 &= a_0 \left(1 - \sqrt{2}\, C + C^2\right)
\end{aligned}
$$

(c) Butterworth band-pass filter

$$
\begin{aligned}
C &= \frac{1}{\tan\left(\pi \, \frac{BW}{f_{rate}}\right)} \\
D &= 2 \, \cos\left(2\, \pi \, \frac{f_{cent}}{f_{rate}}\right) \\
a_0 &= \frac{1}{1 + C} \\
a_1 &= 0 \\
a_2 &= -a_0 \\
b_1 &= -a_0 \, C \, D \\
b_2 &= a_0 \left(C - 1\right)
\end{aligned}
$$

(d) Butterworth band-reject

$$
\begin{aligned}
C &= \tan\left(\pi \, \frac{BW}{f_{rate}}\right) \\
D &= 2 \, \cos\left(2\, \pi \, \frac{f_{cent}}{f_{rate}}\right) \\
a_0 &= \frac{1}{1 + C} \\
a_1 &= -a_0 \, D \\
a_2 &= a_0 \\
b_1 &= -a_0 \, D \\
b_2 &= a_0 \left(1 - C\right)
\end{aligned}
$$

Figure 3.9: Equations for calculating the coefficients for various Butterworth filters. f_{cut} is the cutoff frequency for the high pass and low pass filters. f_{cent} is the center frequency for the band pass and band reject filters. f_{rate} is the sampling frequency of the expected signal which the bilinear transformation will be applied to.

sive Butterworth filters for low-pass, high-pass, band-pass, and band-reject filtering. These four filters simply implement a constructor and method for calculating the coefficients. The coefficients are also come from Dodge and Jerse, and are shown in Fig. 3.9. The `ButterworthLowPass`, `ButterworthHighPass`, `ButterworthBandPass`, and `ButterworthBandReject` are the respective derived classes that use these coefficients.

The other implementation that we have included is a simple implementation of the Hilbert transformation, which can be used to produce the analytical signal (also described as the envelope of the signal), which can be useful in designing efficient feature detection for ECG [64] and measuring the beat-to-beat time intervals from seismocardiogram (SCG) data [98]. This implementation can be found as `HilbertTransform`, and utilizes the `C++` compliant, header-only library, PocketFFT[13].

Social

The *Social* namespace contains the `RobotInterface`, an abstract class that can be used as a starting point for designing robot controllers. The interface only has a pure virtual function that must be overridden by the derived class, being the `configure` method. The interface contains a total of thirteen virtual functions which can optionally be overridden; if a method is not overridden the default implementation will be called stating that "Function *function-name* has not been implemented!!". Nine of the overridable methods are setter-like, providing support for setting peripheral (robot body part) state and velocity, emotion, gesture, speech, speech configuration, volume, audio file, and video file. The remaining four methods are getter-like, providing support for getting peripheral state and velocity, emotion, and commands for the robot (this could include something like an open ROS subscriber to receive commands from the outside the program). Derived classes can be passed to the `RobotManager` class (previously described) which will handle input requests, and calling for the appropriate overridden functions.

Stream

The final namespace of the `HRI Physio Lib`is called *Stream*, which contains generic implementations for receiving and sending streams of data. To start, we've created an abstract class called `StreamerInterface` which is ultimately a part of the previously described bridge pattern. The purpose of this interface is to allow the derived

[13] https://gitlab.mpcdf.mpg.de/mtr/pocketfft/-/tree/cpp

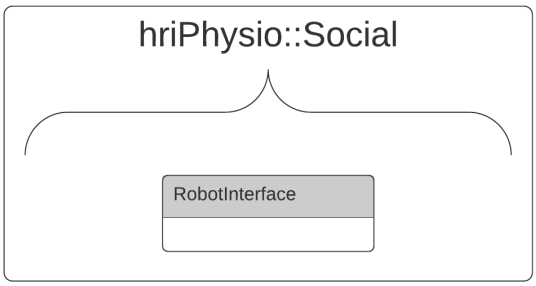


Figure 3.10: The Social namespace. Shown is the abstract class `RobotInterface`. This `C++` interface acts as a starting point for developing robot controllers to inherit from for designing complex behaviors.

concrete implementations a chance to vary widely, ultimately providing support for receiving and sending data with vastly different sources and protocols (*e.g.,* LSL, ROS, YARP). The `StreamerInterface` can also be taken advantage of to support arguably the most important facet of the bridge pattern, namely the ability to select or switch which derived class is used at run-time (thanks in part also to the `StreamerFactory`). The combination of all these is later shown in the description of the `Physio Receiver` in section 3.3.1. The `StreamerInterface` has a total of six pure virtual functions which a derived class must overload. The first two methods are for configuring the encapsulated streamer objects, namely `openInputStream` and `openOutputStream`.

The next two methods are for receiving data; the function names are both `receive`, however, the signatures (variables you pass in) are slightly different. The first function signature takes a reference to a `std::vector` with a custom data type called `hriPhysio::varType` (this data type is described in the next section, *Helpers*), which is used as the buffer where incoming data will be stored. The alternative function signature takes for its first variable a reference to a `std::string`. The second variable for both signatures is a pointer to a `std::vector` of type `double` which is used for storing the received timestamps. Not all streams will support sending or receiving timestamps, so this variable is by default pointing to the `nullptr` (*i.e.,* timestamps are optional). These two separate functions exist to allow the derived implementations to vary and specialize how they transmit signal data versus single character-based strings.

The last two methods which must be overridden are for sending data; the function names are both `publish`, and likewise to `receive`, follow the same two signatures

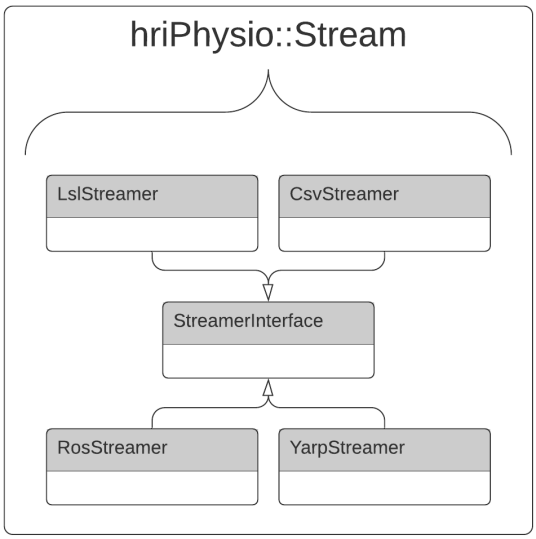


Figure 3.11: The Stream namespace. Shown are the `C++` derived classes from the `StreamerInterface`. The four concrete streamers implement the pure virtual functions from the parent class and encapsulate the necessary components to provide data communication using their namesake. White arrowheads represent inheritance.

(one for a buffer and another for optional timestamps). The only difference is since variables are being passed by reference (to prevent having to unnecessarily copy the data) both the data and timestamps are passed as constants.

We have four concrete implementations of the `StreamerInterface`. So far, we have created the `LslStreamer` which allows for communication with Lab Streaming Layer, `RosStreamer` for connecting to publishers and subscribers on the ROS network, `YarpStreamer` for connecting to nodes on the YARP network, and lastly `CsvStreamer` for writing data out to or reading from a `csv` file. Their implementations are all very similar implementing the six pure virtual functions to specifically tailor to their individual needs.

Helpers

While not a namespace, the `helpers.h` file contains a number of additional common generic functions and minor objects with reusability in mind. The previously mentioned `hriPhysio::varType` is a specific definition of the `C++` class `std::variant`, a

type-safe implementation of `union` introduced in `C++17`. Our definition provides support for the following standard `C++` data types: `char`, `int16_t`, `int32_t`, `int64_t`, `float`, and `double`. Utilizing a definition of `std::variant` effectively allows the data type of receiving and sending streams to be specified at run-time at the cost of the storage required by the largest data type (64-bits).

Complementary to the data type is the `hriPhysio::varTag`, a simple enumeration type for keeping track of what the actual data type was set at run-time. The only other object of note within the helpers file is a custom-made parser for command-line arguments, the `hriPhysio::ArgParser`.

3.3 Implemented Applications And Tools

Using our software library, we have developed a few multipurpose applications and tools with reusability in mind. A common issue in designing experimental scenarios is the great time investment in developing the technologies to make an experimental trial run like clockwork. We've attempted to alleviate this by carefully designing the following software to be used in any sort of relevant situation.

3.3.1 Physio Receiver

The `Physio Receiver` is the culmination of much of the previously described software library. The `Physio Receiver` was designed with the idea of bridging communication libraries. We entered the design of this tool with the knowledge that a great deal of open-sourced data acquisition tools have been developed using LSL. Should we want to take advantage of these already developed tools in HRI scenarios, we would need to ensure that using them is as *convenient* as possible should we want to truly make this technology accessible. For some, the thought of adding additional dependencies to an already otherwise complex software system (think a chaotic ROS dependency graph) can be nauseating, while others might be completely fine adding LSL into their system. The point is that we wanted to include a simple translation interface to bring physiological data into the systems roboticists are working in (namely ROS and YARP) without asking for too much additional development investment on their part.

Aside from bridging technologies, the `Physio Receiver` is also capable of accumulating the potentially shorter windows of data that a device may be transmitting,

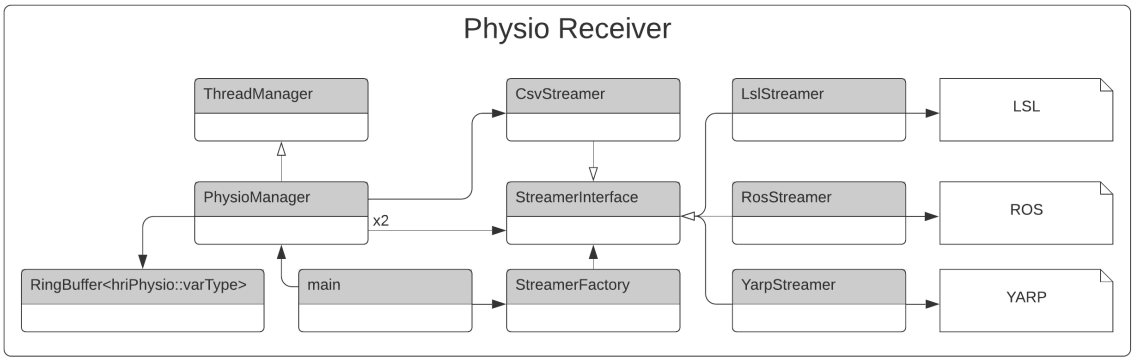


Figure 3.12: The architecture of the `Physio Receiver`. The individual `C++` classes are shown, and how they relate to each other in a pseudo-UML format. Black arrowheads represent composition relationships, while white arrowheads represent inheritance.

and buffering them until some expected length is reached. The process of separating the data accumulation from the processing provides the advantage of ensuring smaller chunks of data aren't missed while processing is in progress. The `Physio Receiver` also has an argument for specifying how many samples should overlap between the current and next window of data.

As seen in Fig. 3.12, the `Physio Receiver` primarily encapsulates an instance of the `StreamerFactory` and `PhysioManager`. Four command-line arguments are available, namely the `--input` and `--output` flags as well as `--conf` for providing the path to a `yaml` configuration file. An additional flag `--interactive` is also available, and allows the `PhysioManager` to run in a mode which allows for text-based inputs (primarily used for debugging). The input and output flags are used as inputs for the `StreamerFactory`, which will return a pointer to a derived `StreamerInterface` object. The `PhysioManager` has no knowledge about which derived class was created, but because of the polymorphic design of the `StreamerInterface`, all necessary methods to the `PhysioManager` have been required to be implemented.

The final instructions the `Physio Receiver` does before starting the manager is passing the specified configuration file. This file should contain a number of variables for specifying what the name of the input and output streams are, as well as the data type, sampling rate of the data, expected input frame length, requested output frame length, the number of channels being received, sample overlap between frames, the number of samples in the ring buffer which should be allocated, and whether or not to log data and where it should be stored. After configuring the manager, the `Physio Receiver` starts the manager; if specified the receiver then calls the

manager's interactive mode method. If this is not called, the receiver will call the manager's wait method which will patiently wait until an event causes all threads to close.

3.3.2 Polar Streamer

The `Polar Streamer` is an Android application (written in `Java`) design and implemented by two undergraduate coop students we recruited in the fall 2020 semester. We chose to pursue implementing this application for two reasons, the first relating to the accessibility to this sensor, and the second being the level of support the manufacture provides for interfacing with the sensor. To expand on the first point, the Polar H10 and OH1 sensors are commercial products that can be purchased for a little over $100 Canadian dollars. These sensors are also designed with exercise in mind, meaning they've been developed to be robust to various levels of noise caused by movement. As well, Gilgen *et al.* [35] have compared the Polar H10 sensors to an expensive research-grade sensor and showed that the signal quality of the H10 was extremely comparable.

The second point about developer support from the manufacturer comes in the form of an SDK[14], a compiled Android library (`aar`). This SDK provides a simple interface for connecting and interacting with their devices over Bluetooth Low Energy (BLE). The SDK is also actively supported by the manufacturer's development team, providing frequent updates and patches.

The `Polar Streamer` uses the SDK to connect to the Polar H10 and OH1 devices, to stream acquired data out onto LSL. The application opens an LSL stream for the metric when a user taps the *stream* button associated with the said metric. The name of the stream follows a standard format of `/Polar{H10/OH1}/{Device-ID}/{metric}` to easily communicate to the user what data is being sent on a particular stream name. When connected with a Polar H10 device the `Polar Streamer` can open an ECG stream with a sampling rate of approximately 130 Hz and an accelerometer stream which has the options of 25, 50, 100, and 200 Hz sampling rate. The SDK also allows the acquisition of heart rate and RR interval data which have already been processed through Polar's own proprietary algorithms. These two metrics have a sampling rate of approximately 1 Hz. Similarly, when a Polar OH1 device is connected, the PPG and accelerometer stream is opened in a similar manner, with access to the heart rate and PP interval data. The user-interface for the `Polar Streamer` can be seen in Fig. 3.13.

[14] https://github.com/polarofficial/polar-ble-sdk

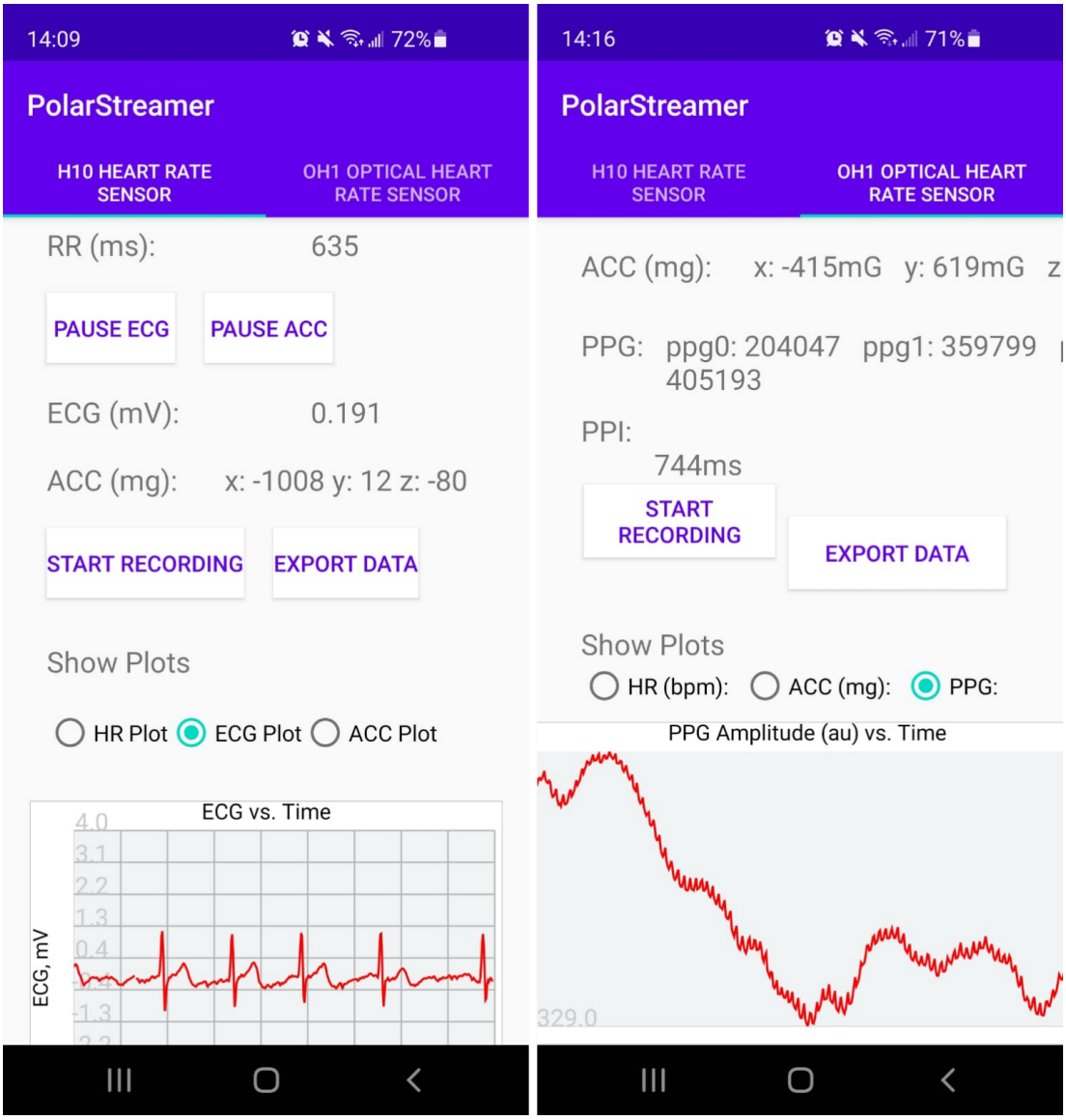


Figure 3.13: The `Polar Streamer` an Android application designed to stream physiological data over Lab Streaming Layer. The application supports the Polar H10 chest strap heart rate monitor and the Polar OH1 armband optical heart rate monitor.

3.3.3 Physio Streamer

The `Physio Streamer` is an *in-development* desktop data acquisition application. The application is written in `Python`, and uses `PyQt5`[15] (`Python` bindings for the cross-platform `C++` library Qt-v5[16]) for its graphical user-interface. The application also uses the cross-platform package Bleak[17] to allow the application to connect to BLE devices by acting as a GATT server. This application is being developed with cross-platform requirements in mind, thus all `Python` packages used to develop this are required to support both Windows and Linux at the very least.

The purpose of this application is to be another possible tool that can be utilized to easily provide a stream of physiological data onto the network. This application will utilize LSL for its initial output stream. The application is planned to initially support the Polar H10 and OH1 devices, with plans to later try expanding support to the Shimmer3[18] suite of sensors which can also be interfaced with over Bluetooth. This application is not yet complete, however, an early build of the application can be seen in Fig. 3.14.

3.3.4 NeuroKit Processor

The `NeuroKit Processor` is a relatively simple command-line interface tool for simple feature extractions of ECG data, using the `Python` toolbox NeurKit2. This module connects to a specified LSL stream which presumably contains ECG. The module buffers the received chunks of data up to a specified number of samples before processing the data. The `NeuroKit Processor` has a total of nine output streams, two of these produce a frame of the same length as the specified buffer size; the first contains cleaned ECG data, and the second contains a binary array where a 1 indicates the presence of an R-peak on that same ECG wave. The remaining seven output streams are all single value features of the frame; this includes the mean heart rate, and the following HRV features: SDNN, RMSSD, pNN50, LF band, HF band, and LF/HF ratio. As the module is not doing anything too computationally intensive, we've designed it to do its own buffering (and not rely on the `Physio Receiver`). This module is also capable of providing overlapping samples between frames of processed data. As well, the module maintains the original ECG signal's timestamps.

[15] https://pypi.org/project/PyQt5/
[16] https://www.qt.io/
[17] https://pypi.org/project/bleak/
[18] http://www.shimmersensing.com/

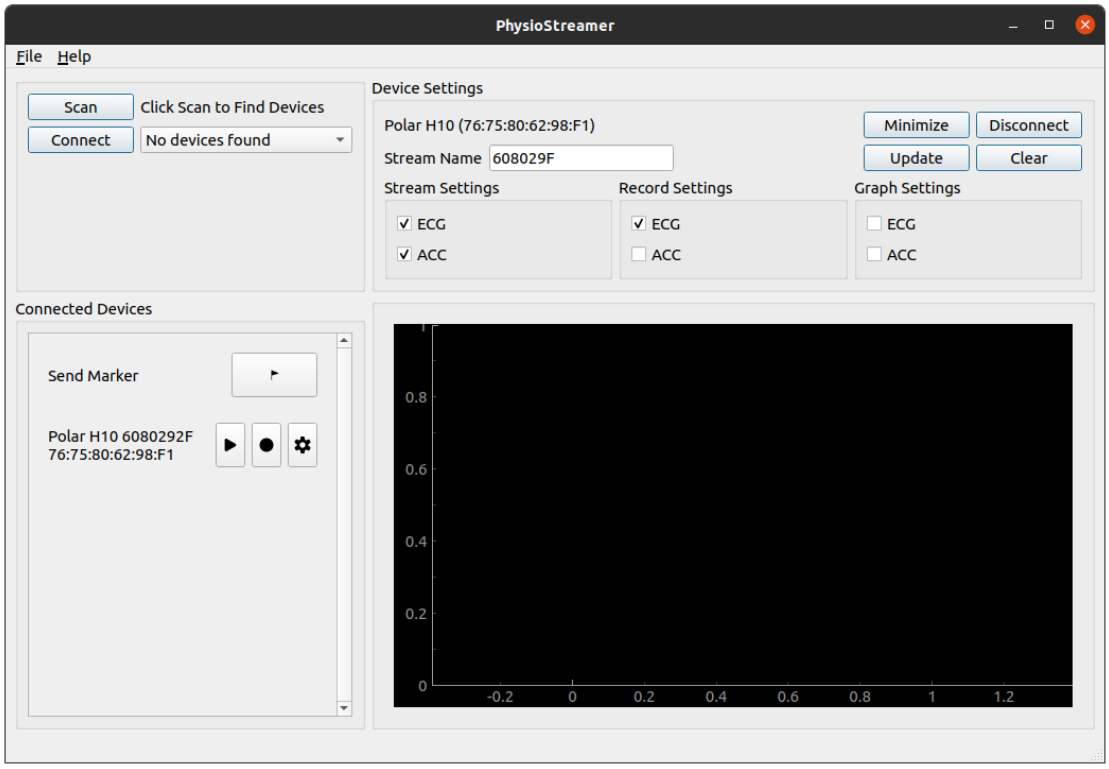


Figure 3.14: The `Physio Streamer` is an in-development desktop application for data acquisition from physiological devices. The user interface contains four "blocks". (top-left) Scan begins searching for Bluetooth devices which are broadcasting themselves that are near. Found devices are populated into the combo box which by default reads "No devices found". Clicking connect will establish a connection to said device, and add it to the list of connected devices (bottom-left). Connected devices can toggle streaming and recording based on the selected settings. (top-right) Settings for the selected device. Buttons for minimizing the current device back into the tray, or disconnecting the device are present. Update and clear change or reset the prefix of the LSL stream to the string currently in the text box. Checkboxes enable or disable acquisition of the relevant metric from the connected device. (bottom-right) If checked in the graph settings, relevant plots will occur in the display.

In the future, this module is planned to be expanded to support processing for PPG and EDA data. The NeuroKit2 has a wide collection of processing methods for various different physiological signals.

3.3.5 NeuroKit Simulator

The `NeuroKit Simulator` is a module that was initially developed to assist in the debugging phase of the `Physio Receiver` module. This module is also a relatively simple command-line interface tool for streaming out simulated physiological data out onto LSL. The `Python` toolbox NeuroKit2 contains methods for producing simulated ECG, PPG, EDA, EMG, and respiratory (RSP) data. We've included a sixth additional data type which produces a linear (incremental) line of data to assist in the trickiest of debugging cases.

Each of the simulated physiological signal types uses slightly different flags with the exception of the following three which were used in all the simulators. They were `--seconds` for the duration of the signal, `rate` for the sampling rate, and `--noise` for the noise to be introduced in the signal. The ECG and PPG both utilized the `--bpm` for the heart rate to simulate (measured in beats per minute). EDA used the `--scr` flag for the number of skin conductance responses to produce, and `--drift` for the amount of signal drift to add. EMG used the `--bnum` and `--bdur` flags for the number and duration of bursts to synthesize. Lastly, RSP used the `--resp` flag for the respiratory rate to simulate.

3.4 Discussion

We have detailed in this chapter a robotics framework to act as a general structure for developing physiologically adaptive HRI scenarios. In section 3.1, we began by describing the various components which should be considered when designing an experimental scenario and aimed in doing so to answer RQ1 which asked:

RQ1: How can we design a robotics software framework for the rapid development of physiologically aware robots?

We first sought to detail the current problems in acquiring real-time streams of data and provided recommendations for how to incorporate physiological data into a

system. Next, we looked through the literature to understand what physiological signals are most frequently used, and what types of processing algorithms and features would be useful for an adaptive system. We then went through and identified six of what we believed were the most common feedback modalities which could be used in an adaptive scenario by a social robot. Lastly, we discussed what factors should be taken into consideration when designing the scenario itself, and what might be useful to keep in mind when developing the adaptations for the robot.

With the general structure established, we moved onto designing software to address each part of the broader framework. In section 3.2 we aimed to describe at a high-level what the architectural *loop* might look like applying our framework. This section continued on to describe at a low-level the software which we've created, in direct response to RQ2 which asked:

RQ2: How can we develop a robotics software framework capable of integrating various communication libraries?

We've developed a software library called the `HRI Physio Lib` to address this. Our library utilizes a number of software design patterns to provide scalable interfaces capable of utilizing different popular communication libraries such as LSL, ROS, and YARP. The code and software which we've provided act as a starting point to provide more rapid development of physiologically aware systems.

As with many research applications, the software developed needs to be highly contextual to the experiment at hand to ensure a seamless and non-distracting experience for any human participant you're trying to collect data from. As a result, our software library may not be perfect for every researchers' experimental scenario. However, we've provided enough software and elements of consideration to ensure researchers can utilize at least a small amount of our work to more efficiently develop their own physiologically adaptive scenarios.

In the next chapter, we will go on to addressing RQ3 which asks if the proposed framework in practice can be used to rapidly develop scenarios.

Chapter 4

QT Cardio-Aware Exercise Coach

This chapter describes the motivation, experimental design, and implementation of a use case scenario framed around exercise mediated by a robot coach. The goal of this scenario was to have a robot within the loop of a cardio-based fitness routine, in which the intensity and speed are actively adapted in real-time using physiological measures from the participant. Furthermore, details in the implementation are related back to the `HRI Physio Lib` outlined in Chapter 3. Work in this chapter was motivated by RQ3 which asked:

RQ3: Is the proposed software framework and HRI Physio Lib able to be used for the rapid development of physiologically aware robots capable of adapting their interactions?

The results of the use case are explored along with a discussion of other potential scenarios in which a physiologically aware robot may provide an improved user experience. Due to the COVID-19 pandemic, in-person human studies had been prohibited; therefore, the use case scenario was conducted with one of the postdoctoral fellows of the Social and Intelligent Robotics Research Lab (SIRRL) as the participant. All in all, this use case aims to show and validate the functionality of the proposed framework (*i.e.,* the data flow, logging system, and communication with the robot) more than systematically validate the use of a robot as an exercise coach.

4.1 Social Robots as Exercise Coaches

Promotion of physical activity has never before been a more apparent necessity. Durations of sedentary behaviors such as sitting time in young adults have increased as a result of stay-at-home orders during the COVID-19 pandemic [105], with mixed observations of increased physical activity dependent on motivation [82]. Physical inactivity is a common contributor to poor health, leading to increased rates of morbidity and cardiovascular diseases [104]. Thus, efforts to increase motivation to engage people in daily physical activities and better promote healthy lifestyles may be a valuable endeavor to showcase the development of a physiologically adaptive system using the `HRI Physio Lib`.

The intersection of exercise and games poses an interesting use case for adaptive systems. Previous studies in exergaming have shown in general to yield positive user-experiences and overall engagement in the system [58]. Physiological measurements have not been uncommon in the exergaming space, however, it can potentially be difficult to determine if a change in physiology is the result of differences in affect or if they're in response to the high-intensity exercise. A study by Barathi *et al.* [4] aimed to discover which physiological measurements are best suited for affect recognition in high-intensity exergaming. Their study identified that skin conductivity, pupil dilations, and pupil fixations were positively correlated with affect, while blinking was negatively correlated.

Other exergaming studies have found success in using physiologically adaptive systems. Muñoz *et al.* [66] used real-time heart rate metrics acquired from a PPG monitor on a smartwatch to modulate the difficulty of a pong-inspired exergame. Their game not only increased the amount of time participants spent in recommended levels of cardio-respiratory exertion but also provided a positive user-experience in comparison to a non-adaptive version of the game.

Video game-based approaches provide levels of immersion that can attract and provide engaging experiences to persuade people to "go just a bit longer". It may be the case that robots could offer a convenient level of authority to engage people to–like games, "go just a bit longer". A study by Cormier *et al.* [14] found the authority which robots wield could be used to command participants to engage in a repetitively tedious task that involved renaming thousands of image files from `jpg` to `png`. Aroyo *et al.* [3] found that this robotic persuasion could be taken a step further to push people into doing things they understand as wrong. Additionally, Schneeberger *et al.* [87] found that virtual agents also held levels of authority to be just as persuasive as a human-instructor in a scenario that required participants to

do “stressful” and “shameful” tasks.

Robots are not necessarily a new mode for studies involving exercise. Sussenbach *et al.* [97] showed that a social robot could be used in a workout scenario to promote higher motivation and improved training effects when compared against a text-display system. A related study by Schneider and Kummert [88] compared social robots against virtual agents in an exercise experiment which found that social robots provided more motivation and engaged participants longer than virtual agents. In a follow-up study, Schnieder and Kummert developed an online preference learning system to adapt to a participant during an exercise scenario via verbal feedback [89]. They found that an adaptive social robot resulted in an overall more engaging experience for the participants.

4.2 Implementation of the Experimental Scenario

Using fitness as a focal point for a compelling and practical use case, we designed an experimental scenario revolving around a physiologically aware exercise coach. The scenario we developed involved a human participant being led through a cardio-based fitness routine, in which the intensity and speed were actively adapted in real-time using physiological measures of the participant. By adapting the speed of the activity, we aim to encourage an effective cardiorespiratory exercise for the participant by maintaining them in a healthy training zone (*i.e.,* not over or under exerting themselves over the duration of the activity).

Using the tools and software from the `HRI Physio Lib` described in Chapter 3, we designed a controller for the robot and peripheral systems (audio and visual) which integrate with the socially interactive robotics platform QT robot. Exercise intensities were adaptively changed by using heart rate (HR) readings from the participant. We set our goal for this use case to illustrate the functioning of the `HRI Physio Lib` and how it might be used to rapidly develop a scenario with the robot in-the-loop.

4.2.1 The Socially Assistive QT Robot

For this use case, we utilized the socially assistive QT robot (seen previously in Fig. 3.2b), a commercially available platform from LuxAI[1]. The QT robot provides an

[1] https://luxai.com/

accessible gesture, emotion (face-display), and text-to-speech interfaces which made quick prototyping of behavioral responses easy. The emotion and text-to-speech interfaces could be accessed through a ROS publisher, while the gesture interface could be accessed through a ROS service client. Custom-made audio and video interfaces were developed which allowed exercise music and reference videos to be dynamically swapped throughout the scenario (these are described more in-depth in section 4.2.3).

The QT robot has a few additional interfaces which we did not utilize in this scenario; one of which can be used for 3D body and facial tracking using the Intel RealSense D435 depth camera that is embedded in the robot's head. In future studies with real human participants, this interface could be utilized for scoring the participant throughout the exercise. The full documentation for the robot can be found on its ROS wiki page[2].

4.2.2 The Exercise Routine and Experimental Design

An exercise routine was created following the American College of Sports Medicine's (ACSM) guidelines for cardiorespiratory exercise prescription [34]. The routine was comprised of four phases: a three-minute calibration–to establish the participant's HR_{rest}, followed by a five-minute warm-up exercise, ten-minutes of cardiorespiratory exercise, and lastly a five-minute cool-down exercise. The phases can be seen in Fig. 4.2.

The Calibration Phase

During the calibration phase, the participant was seated on a chair in front of a table that held the QT robot, a computer monitor, computer speakers, and a laptop. The laptop acted as a command server which had the computer monitor and speakers connected to it. The complete setup can be seen in Fig. 4.7.

The participant wore the Polar H10 chest-strap sensor which was connected to the Android application `Polar Streamer` described in section 3.3. The `Polar Streamer` transmitted the data over Lab Streaming Layer (LSL) to the `Physio Receiver`, running on the laptop computer where the data was logged, buffered, and relayed out onto ROS. The main behavioral loop `QT Physio Coach` subscribed to this ROS

[2] http://wiki.ros.org/Robots/qtrobot

node, giving the robot knowledge of the participant's real-time HR. The data received by the behavioral loop was stored into HR_{buffer}.

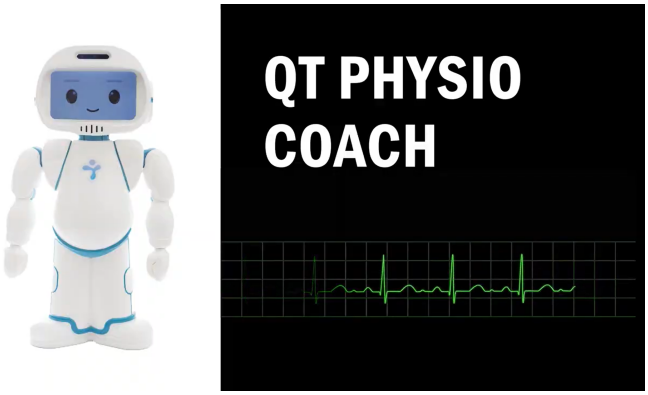


Figure 4.1: The splash screen reads QT Physio Coach and is shown to the participant during the exercise scenario. The splash screen contains an animated green ECG signal which loops across the screen.

For the duration of the calibration phase, the robot instructed the participant to relax and enjoy the music while it collected their HR. As feedback, the robot would periodically make encouraging faces with its face-display, speak to participants reminding them to breathe and relax, as well as informing the amount of time left in the calibration phase.

At the end of the calibration phase, the resting heart rate (HR_{rest}) was calculated by taking the mean of the collected HR_{buffer}. At this time, the maximal heart rate (HR_{max}) was also approximated by using the following regression equation from Tanaka *et al.* [100]

$$HR_{max} = 208 - 0.7 \cdot Participant_{age} \tag{4.1}$$

The maximal heart rate is then used to find the heart rate reserve (HRR) through the difference with the resting heart rate

$$HRR = HR_{max} - HR_{rest} \tag{4.2}$$

The ACSM provides recommendations for cardiorespiratory fitness training, saying that people should exert themselves into certain targeted HR zones defined in

terms of HRR percentages [34]. For our exercise scenario, we established a light-to-moderate target HR zone between 40% and 70% of the HRR (HRR_{40} and HRR_{70} respectively). This target HR zone can be seen as the green shaded region in Fig. 4.12.

At the end of the calibration phase, the QT robot thanks the participant for their patience, notifies them of what their calculated HR_{rest}, and tells them that it will try to keep their HR between HRR_{40} and HRR_{70}.

The Exercise Phases

At the beginning of the warm-up phase, the QT robot asked the participant to move the chair out of the way and replace it with an aerobic stepping platform. The stepping platform has an adjustable height, which we set at its lowest of 4 inches. This can be seen in Fig. 4.8.

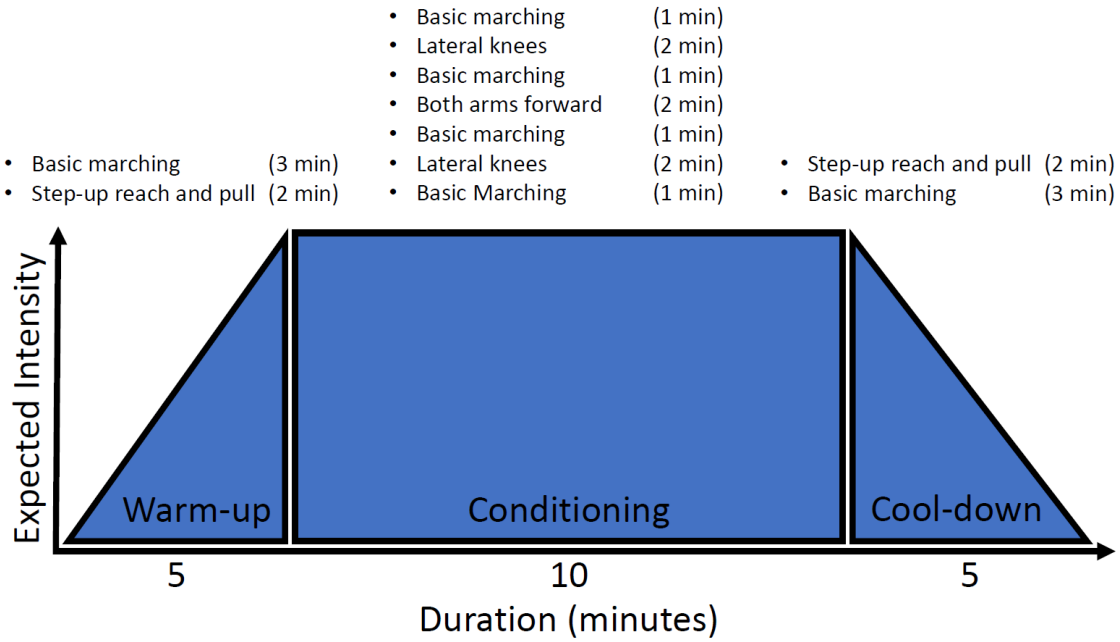


Figure 4.2: The timeline of the exercise routine. Beginning with a 5-minute warm-up phase, followed by 10-minutes of conditioning exercise, and ending with a 5-minute cool-down phase. Activities that occurred in each phase and their duration are shown from top to bottom.

The first exercise begins with the QT robot telling the participant to follow its lead with a "basic marching" activity (described in table 4.1). During this period,

Figure 4.3: Basic marching; a complete description of movements for the activity in table 4.1.

Figure 4.4: Step-up reach and pull (a1); a complete description of movements for the activity in table 4.1.

motivational music began playing at 115 beats per minute (BPM). This music would be changed with a ±5 BPM version of the same song during a rule update (described later in this section; see algorithm 1). The ceiling for the speed of the exercise music was 135 BPM, and the floor was 110 BPM.

The procedure of each exercise activity was similar, varying in length, intensity, and which activity is being performed. The overall schedule of the routine can be seen in Fig. 4.2. At the beginning of each exercise activity, a short reference video of a human fitness instructor performing the current exercise begins playing. During this, the QT robot asks the participant "Try to imitate the video, while keeping pace with the music". The reference video looped onto itself for 30 seconds, before the robot's decision loop turned the video to a splash screen reading `QT Physio Coach`, with a photo of the robot and a looping animation of an ECG waveform (see Fig. 4.1). The decision to cut away from the reference video of the fitness instructor to a splash

Figure 4.5: Lateral knees (a2); a complete description of movements for the activity in table 4.1.

Figure 4.6: Both arms forward (a3); a complete description of movements for the activity in table 4.1.

screen after 30 seconds was made in an attempt to guide the participant's attention towards the robot once they were aware of the current exercise.

When the reference video started playing, the QT robot also began performing a gesture unique to the current exercise activity. Along with the music, the speed of these gestures was also modified by ±5% during a rule update. The initial speed multiplier was set to 95% the original recorded gesture speed, maxing out at 115% and a minimum of 90%.

Throughout the current exercise activity, the QT robot would periodically do one of three behaviors at random. The first possible behavior was to deliver some motivational speech to the participant, which was chosen at random from a predefined list of positive phrases; the second behavior was to display a positive facial expression on the QT robot's face-display; the third behavior had the QT robot telling the participant what their current average HR reading was (using the last 30 seconds of

buffered data). This random behavior decision was on a 20 second timer and would have a lower priority of action if a rule update were to also happen.

Algorithm 1: Heart Rate Rule Update used in the QT Physio Coach

$HR_{exercise} = mean(HR_{buffer})$
if $HR_{exercise} < HRR_{40}$ **then**
  Increase the speed of music and gestures;
  Tell the user we're picking up the pace
else if $HRR_{70} < HR_{exercise}$ **then**
  Decrease the speed of music and gestures;
  Tell the user not to over exert themselves
else
  Provide encouragement to continue keeping up to the pace
end

During the whole exercise scenario, HR data is streamed to and buffered by the decision loop. A rule update occurred every 30 seconds and is detailed in algorithm 1. In summary, the mean of the HR data currently in the buffer (HR_{buffer}) is computed to find the average HR of the participant ($HR_{exercise}$) for some window of time. After this data has been used, the buffer is cleared, discarding all old data. This buffer is also cleared at the beginning of each exercise activity, to ensure each rule update contains approximately 30 seconds worth of HR data.

Now that the template for activities has been described, we can better describe the macro view of the routine. As well, the complete description for each activity can be seen in table 4.1.

As mentioned earlier, the warm-up phase began with three minutes of a "basic marching" activity. The "basic marching" activity is used throughout the entire routine as a means to give the participant a short break between periods of the high-intensity activities to recover, further keeping them between 40% and 70% of their *HRR* (target HR zone). After the initial "basic marching" activity, the first high-intensity activity began, two minutes of a "step-up reach and pull" activity. This activity concluded the warm-up phase.

The purpose of the conditioning phase is to keep the participant engaged and within the target HR zone for the approximately ten minute long phase duration. This phase begins with one minute of the "basic marching" activity before beginning another high-intensity activity; the first of which was an activity called "lateral knees" for two minutes. After the "lateral knees" concluded another minute of the

Table 4.1: Description of the different exercise activities used in the QT Cardio-Aware Exercise Coach use case. Note: Assumes a right-sided dominance; all activities are repeated and alternate with the opposite foot beginning the next cycle.

Exercise Activity	Description	Reference
Basic marching	• Right foot steps onto the platform • Left foot steps onto the platform • Right foot steps off the platform • Left foot steps off the platform	Fig. 4.3
Step-up reach and pull (a1)	• Right foot steps onto the platform • Both arms reach forward • Left foot bends backward at the knee • Both arms pull down • Left foot drops back to the floor • Right foot steps off the platform	Fig. 4.4
Lateral knees (a2)	• Right foot steps onto left side of the platform • Left knee raises to be parallel with the floor • Left foot bends backward at the knee to be near the back of the right knee • Arms pull forward or backward to provide balance • Left foot drops back to the floor • Right foot steps off the platform	Fig. 4.5
Both arms forward (a3)	• Right foot steps onto the platform • Both arms reach forward past the platform • While the left foot reaches backward to provide balance • Both arms lower and left foot returns to the floor • Right foot steps off the platform	Fig. 4.6

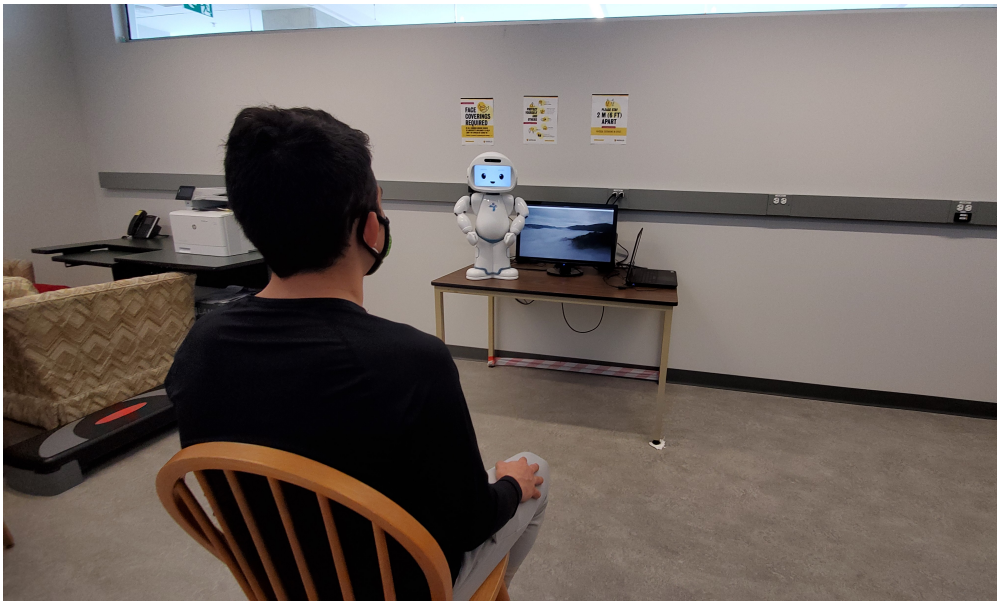


Figure 4.7: Calibration phase of the exercise scenario. The QT robot plays relaxing music and videos while collecting the participants' heart rate for three minutes to compute the resting heart rate.

"basic marching" began. This was followed by two minutes of the "both arms forward" activity which is easily the highest intensity activity of the four. When this finished, another minute of "basic marching" began. Finally, the conditioning phase concluded with two minutes of "lateral knees" and one additional minute of "basic marching".

The final phase was the cool-down phase, which has the purpose to gradually lower the heart rate and blood pressure of the participant back to a baseline level. Abrupt stops after conditioning exercises can lead to blood pooling in the extremities which can cause cardiovascular complications such as dizziness and even fainting [34]. The cool-down phase began with the same "step-up reach and pull" activity that was in the warm-up for two minutes, before finishing with three-minutes of the "basic marching" activity.

At the end of the cool-down phase, the QT robot congratulated and thanked the participant for completing the fitness routine. Both, the physiological and the robot's feedback were synchronously stored in `csv` files using the *Data Logger* component of the `Physio Receiver`.

This use case scenario was conducted with one of the postdoctoral fellows of the

Figure 4.8: Conditioning phase of the exercise scenario. The QT robot leads the participant through a variety of cardio activities with an aerobic stepping platform.

Social and Intelligent Robotics Research Lab (SIRRL) as the participant due to the COVID-19 pandemic making in-person studies nonideal. The data recorded was used to produce Fig. 4.12, which shows the cardiovascular behavior and how the robot was capable to push the participant to achieve and maintain the individual target HR zone. This is discussed more in-depth in section 4.3.

4.2.3 Implementation Details of Experimental Scenario

In the previous subsection, a variety of moving parts was described. In this section, we'll go in-depth to discuss the implementation details of each component used which allows the experimental scenario to run smoothly. Other components in the scenario included the `Polar Streamer` and `Physio Receiver` which were outlined in section 3.3.

QT Physio Coach

The first of these was the `QT Physio Coach`, which was the decision loop and was responsible for controlling the scenario. This module acts as the implementation of

the *Scenario and Adaptation* component shown in Fig. 3.3. Like in this figure, the *physio interpreter*, *robot's behavior*, and *decision loop* were used as inspiration for designing the facets of this software.

As seen in Fig. 4.9, the main creates an instance of the `QtPhysioCoach` class, which it then configures, and proceeds to start the scenario. The coach inherits from the `ThreadManager` class methods to provide a simple asynchronous threading interface for the scenario. The coach starts by opening a provided `yaml` configuration file and sets up the ROS interfaces for sending and receiving data. This file contains information regarding the participant such as their name and age; the names of input and output ports; paths to audio and videos used; as well as lists of speech prompts which the coach will pull randomly from. The coach has a ROS subscriber for receiving the heart rate data from the participant, and a ROS publisher for providing commands to the `QT Controller`, which directly interfaces with the robot. Next, the coach initializes a `RingBuffer` which is used to store the received heart rate data.

After finishing the configuration, the coach initializes two threads, one for the scenario, and another for the input loop (which receives the heart rate data from the ROS subscriber). If the `QT Physio Coach` was started in *interactive* mode, it will wait for the user to provide the command "start"; else the scenario will start right away. As previously described in section 4.2.2, the scenario involves a calibration phase to calculate the participant's resting heart rate, which is further used to calculate the HRR_{40} and HRR_{70} used in the rule update. The coach follows a linear process of executing the exercise for some specified duration with some motivational dialogue in between. We describe later in section 4.3.1 some of the limitations with this particular implementation.

QT Controller

The next component was the `QT Controller`, which was the aspect that received commands from the *Scenario and Adaptation* component (`QT Physio Coach`). The `QT Controller` contains a concrete implementation of the abstract `RobotInterface` class called `QtController`, and provides implementations for the feedback modalities to execute on the QT robot. The *Robot Controller* component of Fig. 3.3 can be thought of as a funnel to interface with the robot, simplifying and abstracting away the details needed to accomplish the feedback modalities used in the scenario.

As seen in Fig. 4.10, the `QT Controller` follows a similar architectural structure as the `Physio Receiver` (described in section 3.3.1). The controller begins by instantiating a `QtController` which is passed to the `RobotManager`. The manager

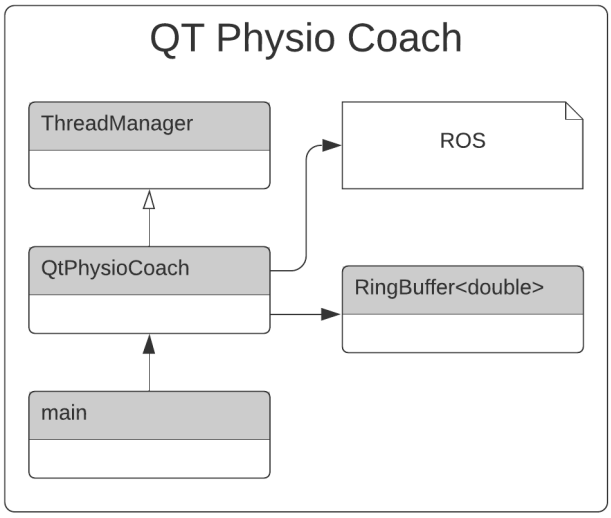


Figure 4.9: The architecture of the `QT Physio Coach`. The individual `C++` classes are shown, and how they relate to each other in a pseudo-UML format. Black arrowheads represent composition relationships, while white arrowheads represent inheritance.

is described in section 3.2.2, but in short, its purpose is to encapsulate an object derived from `RobotInterface` and provide streamlined functionality.

The `QtController` encapsulates seven ROS publishers and three ROS service clients for controlling both the QT robot as well as the peripheral audio and video interfaces (described next). Of the six feedback modalities outlined in the *Social Robot and Communication* component (described in section 3.1.3), we have implemented four which we felt were relevant and feasible for the QT robot in our experimental scenario; they are *screen-based*, *speech-sounds*, *facial expressions*, and *body language & movements*.

For *screen-based*, we used a peripheral computer monitor to display videos to the participant during the exercise scenario. This was done by sending messages to the video interface containing the path to the `mp4` file it should open and begin playing. The *speech-sounds* were implemented in two parts. The speech portion was handled through the QT robot's speech interface which allowed for real-time text to speech interface, while the sound portion (like the screen-based feedback) was handled through an external audio interface which handled playing music through the peripheral computer speakers. *Facial expressions* were handled through the QT robot's emotion interface and would change the robot's facial display to a pre-rendered expression. Lastly, the *body language & movement* feedback was implemented through the QT

robot's gesture playback interface. Prior to the scenario, four gestures were recorded using the robot's gesture interface; which effectively allows releasing the PWM motors and recording movements of the robot being puppeted by the developer, and allows for playback of those gestures at a variable speed. The `QtController` was also designed to allow individual controlling of specific joints on the QT robot, however, this is not used in the experimental scenario.

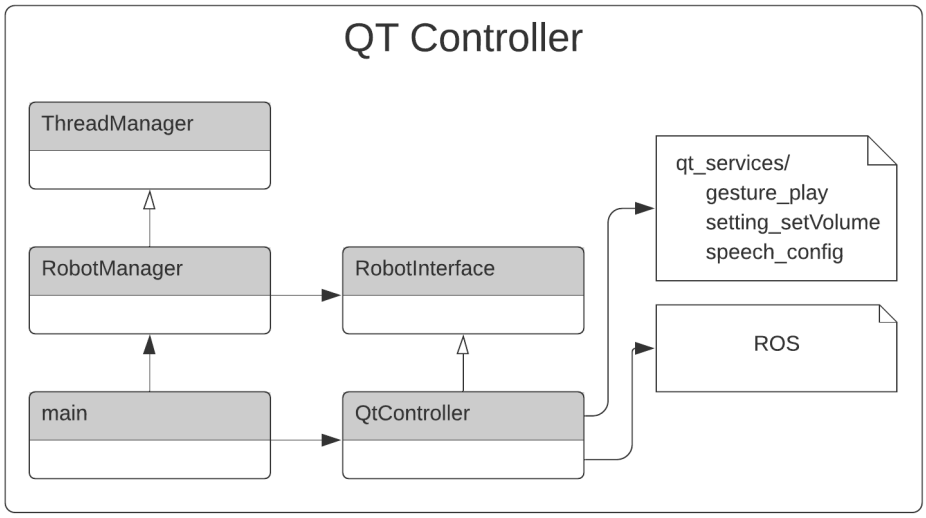


Figure 4.10: The architecture of the `QT Controller`. The individual `C++` classes are shown, and how they relate to each other in a pseudo-UML format. Black arrowheads represent composition relationships, while white arrowheads represent inheritance.

Audio and Video Interfaces

As described in the previous section, the exercise routine involved a lot of changing of audio and video elements throughout the course of the exercise routine. To provide smooth transitions and looping of assets, a custom-made audio and video interface was developed. The general structure for both interfaces was quite similar, in that both ROS nodes were written in `Python` and had a ROS subscriber exposed for receiving a `ros::std_msgs::String` which interrupts the interface and tells it which file to dynamically change to.

The `Audio Streamer`[3] used the `Python` packages `SoundFile`[4] for opening and decoding audio files and the `PortAudio`[5] binding `PyAudio`[6] to open an audio stream for playing the requested audio files. To prevent annoyances from the *clipping* of the audio when a new file was requested, a fade-in and fade-out behavior was implemented. The duration of this could be changed through a command-line flag `--fade`, which is set to a default value of one second. As well, a looping behavior was also included which would reset the playback window to the beginning when an audio track ended.

The `Video Streamer`[7] used the `Python OpenCV` binding[8] to open and decode the video file, as well as to open a window to visualize and playback the video. The reference videos described in section 4.2.2 have a duration anywhere between 14 and 30 seconds long; to keep the visuals of the experimental scenario smooth, when the current video ends it loops back to the beginning. Unlike with the `Audio Streamer`, when the `Video Streamer` receives a new file to begin playing, the video will immediately be swap to the new video without fading-out.

[3] https://github.com/kothiga/hri-physio/tree/main/modules/audioStreamer
[4] https://pysoundfile.readthedocs.io/en/latest/
[5] http://www.portaudio.com/
[6] https://people.csail.mit.edu/hubert/pyaudio/
[7] https://github.com/kothiga/hri-physio/tree/main/modules/videoStreamer
[8] https://github.com/opencv/opencv-python

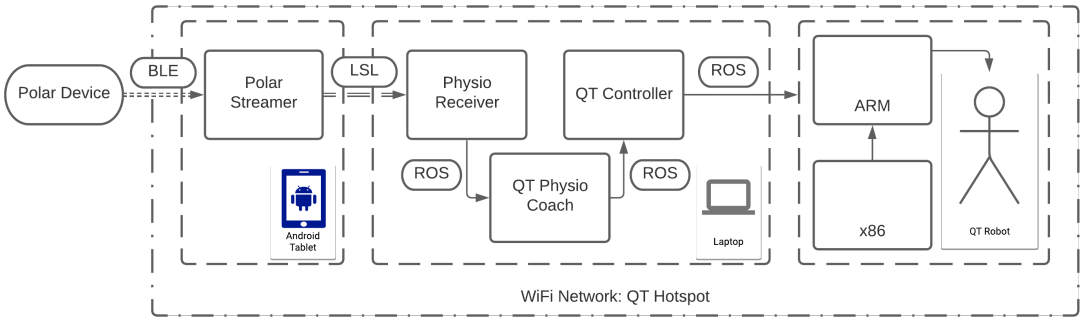


Figure 4.11: Overview of how the various modules used in the `QT Cardio-Aware Coach` scenario communicate. All devices (with the exception of the *Polar Device*) were connected over a local WiFi hotspot created by the QT robot. From left to right, the *Polar Device* connects to the *Polar Streamer* (running on the Android tablet) over Bluetooth Low Energy (BLE). Data received is streamed out onto Lab Streaming Layer (LSL) which is then caught by the *Physio Receiver*. The data is then transmitted over a Robot Operating System (ROS) publisher to the *QT Physio Coach* (the primary decision loop of the scenario). Selected behaviors are sent to the *QT Controller* over ROS which then translates the commands to correctly broadcasts actions to the QT robot's interfaces.

4.3 Results of Scenario and Future Extensions

The heart rate over the course of the experimental scenario can be seen in Fig. 4.12. Our first steps at developing an experimental scenario revolving around an exercise coach with physiologically aware behavior was successful at maintaining our single human participant within their calculated target HR zone. While this mock experiment was done with a postdoctoral fellow who was familiar with the project and therefore results cannot be empirically described to make any generalizable claims, it does manage to demonstrate the feasibility of using the `HRI Physio Lib` as an adaptive mechanism to empower social robots with physiological awareness. Our aim was to answer RQ3 which asked:

RQ3: Is the proposed software framework and HRI Physio Lib able to be used for the rapid development of physiologically aware robots capable of adapting their interactions?

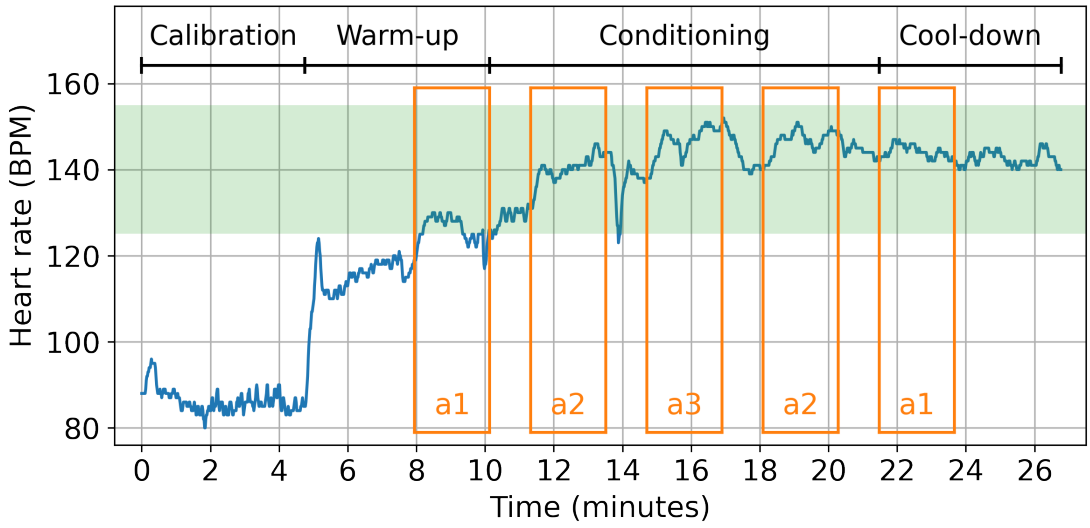


Figure 4.12: Change in heart rate (beats per minute – BPM) over time (minutes) of exercising with the `QT Cardio-Aware Coach`. The green shaded region between 125.2 (HRR_{40}) and 155.1 (HRR_{70}) BPM shows the target heart rate zone of the participant. The HR_{max} for our participant was 184.9 BPM and the HR_{rest} was 85.4 BPM. Top shows the four phases of the exercise experiment. With the exception of the calibration phase, times that do not have an orange box, the participant was instructed to perform a basic marching exercise. The activities used were: (a1) step-up reach and pull; (a2) lateral knees; (a3) both arms forward.

We've highlighted key components in developing a physiologically adaptive HRI scenario from signal acquisition to real-time adaptation through facilitation of the `HRI Physio Lib`. It is difficult to provide the qualitative research question with a definitive accept or deny, however, our use case (while simple) demonstrates how a physiologically aware scenario might be implemented.

As future work, this experimental scenario could be extended to a larger human participant study to properly provide evidence data and rigorous analysis to support the efficacy of such a physiologically aware exercise coach. A between-participant experimental design would suit the scenario well, with the independent variables comparing a dynamic adaptive robot coach with a *static* non-adaptive robot coach. Dependent variables could include measurements from average heart rate over time; to other interesting metrics related to maintaining stepping frequency to the BPM of the music; as well as a number of additional physiological measures such as blink rate, pupil dilation, gaze fixations (collected from a commercial eye tracker such as

Tobii), or skin conductance (electrodermal activity collected from a wearable GSR sensor such as the Shimmer3 GSR+ Unit). Additional qualitative dependent variables such as engagement or positive affect might be collected through questionnaires such as Likert [53] or the Intrinsic Motivation Inventory (IMI) [85]. Modes of collecting this could be through either or both scheduled experience sampling during the exercise scenario (between exercise activities) and post-trial qualitative feedback of experience.

There are a number of observed shortcomings to the current design. For starters, in our one-shot run of the exercise scenario, the `QT Physio Coach` never had the opportunity to reduce the speed and intensity of the activity; the reason for this being the participant's heart rate never went above the calculated HRR_{70} (155.1 BPM). While it's nice to imagine that this speaks to the effectiveness of the robot coach to maintain the participant in their target HR zone, it's more likely that 70% of the HRR was a bit high. Further exploration should first be done to better evaluate the "best" regions for an upper bound. On closer examination of the data and events, there were a total of four instances where the `QT Physio Coach` increased the speed, all of which occurred prior to the beginning of the "step-up reach and pull" (a1) activity in the warm-up phase (between 5 and 8 minutes into the session). Further thought is needed in the design of this phase, with consideration of potentially limiting the number of speed-ups that can occur in the warm-up phase. Additionally, it may be worth considering a 60 second pause on rule updates after one occurs to prevent successive speed-ups.

Algorithm 2: Proposed Fuzzy Heart Rate Rule Update

$HR_{exercise} = mean(HR_{buffer})$
$HR_{target} = mean(HRR_{40},\ HRR_{70})$
$HR_{norm} = \dfrac{abs(HR_{exercise} - HR_{target})}{HR_{target} - HRR_{40}}$
$\varepsilon = rand(0,\ 1)$
if $HR_{exercise} < HR_{target}$ *and* $\varepsilon < HR_{norm}$ **then**
    Increase the speed of music and gestures;
    Tell the user we're picking up the pace
else if $HR_{target} < HR_{exercise}$ *and* $\varepsilon < HR_{norm}$ **then**
    Decrease the speed of music and gestures;
    Tell the user not to over exert themselves
else
    Provide encouragement to continue keeping up to the pace
end

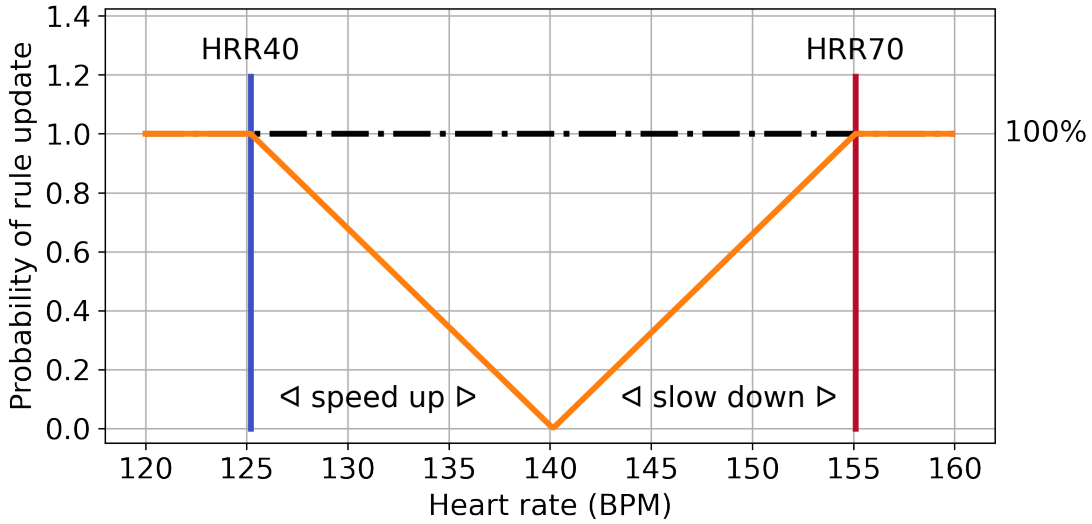


Figure 4.13: Probability distribution of a rule update occurring with the proposed fuzzy rule (algorithm 2). Using the measured HRR_{40} (125.2 BPM) and HRR_{70} (155.1 BPM) from the `QT Cardio-Aware Coach` scenario we find the HR_{target} to be 140.2 BPM. The closer the $HR_{exercise}$ is to the HR_{target}, the less likely it is for a rule update to occur. If the $HR_{exercise}$ is bellow HRR_{40} it is guaranteed to trigger a rule update ("speed up"), while if the $HR_{exercise}$ is above HRR_{70} a rule update ("slow down") will occur.

As well, it may be more desirable to use a "fuzzy logic" method over the current "binary" (is over or under) decision method described in algorithm 1 for rule updates. Such a fuzzy method could be based on the distance of the mid target HR zone to the upper and lower bounds to determine increasing or decreasing of the speed and intensity of the activity.

Other considerations should be taken regarding the warm-up and cool-down phases; specifically with the rule updates and when they occur. At present, the exercise scenario may have a rule update occur during the warm-up, conditioning, or cool-down phases. It may be worth considering restricting the decision of a rule update to only the conditioning phase and having scheduled decreases to the speed and intensity during the cool-down phase. The overall purpose of the cool-down phase is to bring the participant's heart rate to some baseline level to prevent dizziness or even potential fainting caused by abrupt stopping after high-intensity exercise [34]. Looking at Fig. 4.12, the participant is very much maintained in their target HR

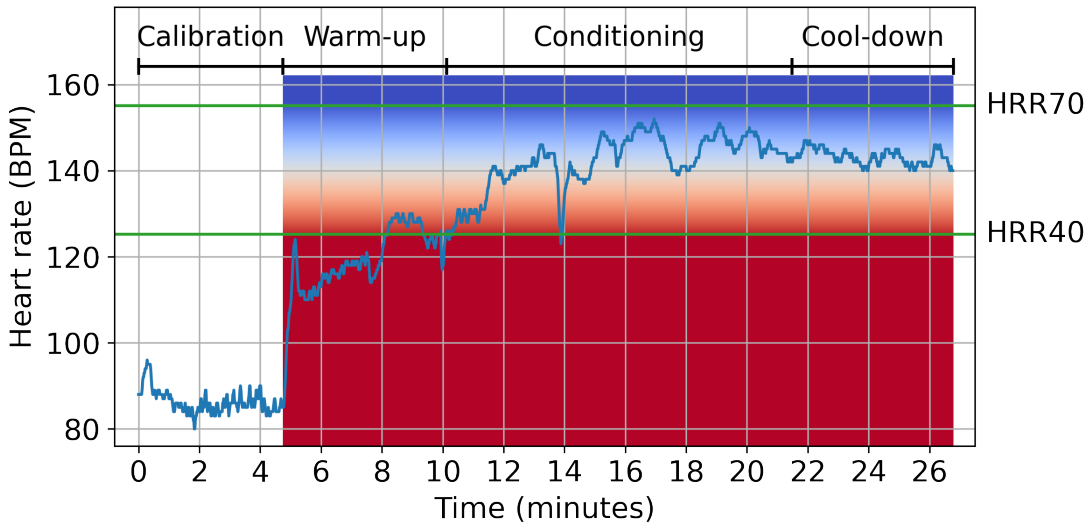


Figure 4.14: Probability distribution of rule update occurring with the proposed fuzzy rule (algorithm 2), over top the data from the `QT Cardio-Aware Coach` scenario. The closer the $HR_{exercise}$ is to HRR_{40} or HRR_{70}, the more likely a rule update is to occur. Red is used to show the likelihood of a "speed up" update occurring, while blue is used to show the likelihood of a "slow down" update occurring. A $HR_{exercise}$ bellow HRR_{40} or above HRR_{70} are guaranteed. See Fig. 4.13 for further detail.

zone during the cool-down phase, when their heart rate should instead be gradually descending below HRR_{40}.

Final considerations for a future human participant study might include providing the QT robot with more knowledge of the person it's interacting with. This might include interesting verbal dictation elements such as acknowledgment prior to the beginning of the warm-up phase where the participant would be required to tell the QT robot they're "ready" to begin, or providing the participant the ability to explicitly tell the QT robot to "slow down" and that they'd like to "speed up" the exercise scenario. Currently, the Intel RealSense D435 depth camera located in the QT robot's head is not being utilized in any capacity. Its incorporation could provide a number of additional elements to the experimental scenario including gamification through a scoring system-based repetitions of the skeletal-frame (pose) of the participant.

4.3.1 Limitations in Implementation

As with many designs, there are always improvements that can be made. A short-list of aspects that could have improved the speed of development of the use case and make the scenario overall more smooth would have been the inclusion of some sort of state machine library such as rFSM[9] or BehaviorTree.CPP[10]. The current implementation of the exercise scenario is rather *linear* in nature, therefore any attempt to introduce additional complexities would require quite a bit of additional development. This use case was simply naively taking the mean of the HR data stored in the buffer. The Polar SDK[11] that was used to develop the Android application described in section 3.3.2 provides a processed HR in the form of beats per minute. In a more complex scenario, it would be preferred to decouple the implementation of processing modules from the scenario and adaptation.

As described earlier, the current design does not consider any auditory or visual components, both of which are accessible through the QT robot. By adding some form of speech dictation to allow the participant to acknowledge the QT robot or manually decrease the speed could add a level of social interaction to make a potential dynamic adaptive robot coach a more enjoyable experience over a *static* non-adaptive robot coach. As well, introducing vision could add fun gamification elements such as a scoring system of the participant based on repetitions.

[9] https://orocos.org/stable/documentation/rFSM/index.html
[10] https://www.behaviortree.dev/
[11] https://github.com/polarofficial/polar-ble-sdk

4.4 Other Potential Use Cases

In this last section, we'd like to highlight a few additional use cases in which we envision the addition of physiologically adaptive components could provide exciting advances to the user-experience. This future extensions section is meant to act as a collection of ideas for directions that this technology could be taken and to encourage researchers who are looking for domains to apply physiological sensing to social robots.

Improving Engagement

Engagement is a paradigm in human-computer interaction (HCI) literature that is strongly agreed as a priority for designers' interfaces [21]. ECG, EEG, and heat fluctuations have been used previously by Belle *et al.* [7] to develop an engagement and attention detection system using physiological data that was captured from 8 participants watching a series of videos. Bian *et al.* [8] showed the potential of a physiologically-sensitive system for driving training in a scenario with participants who had autism spectrum disorder, using a virtual-reality driving simulator. Their developed system incorporated a fusion of EDA, PPG, and respiration to develop a multimodal model of engagement. Real-time modifications were mode to the difficulty of the driving task either based on an engagement-sensitive (ES) based rule using a combination of the participant's engagement level and their performance; or a strictly performance-sensitive (PS) based rule. The authors showed with a 5-point Likert scale that between the two groups, the participants in the ES group had a significantly higher engagement, compared to the purely PS group. As well, participants within the ES group had subjectively reported that they liked the difficulty adjustments more than the PS group did.

Having a window into the engagement of a person interacting with a robot could yield similar increases to the effectiveness of the interaction. Rani and Sarkar [77] have proposed a framework for implicit human-robot communication based on physiological measurements to maintain task engagement of human operators when working with robots. In their proof-of-concept teleoperation experiment, they showed the usefulness of an adaptive robot that reliably predicted the engagement of the user. In another study, Foster *et al.* [32] developed the socially aware bartending robot JAMES (Joint Action for Multimodal Embodied Social Systems) to gauge the engagement of patrons (participants) through facial tracking to improve the interaction quality. A review by Oertel *et al.* [69], found that only 2% of the studies in

their curated list of papers on engagement with human-agent interaction contained physiological measurements in their evaluation methodology.

The potential for physiological measurements to be used for real-time readings of user engagement to make intelligent adaptions is still quite novel and deserves future investigation as wearable technology becomes evermore present in real-world scenarios.

Deeper Social Connections

Closely related to engagement, providing deeper social connections with a social robot is another area where the applications of physiologically-sensitive systems might excel. Robots have been shown to evoke emotional responses in children when they're treated unfairly and forced to go "into a closet" [45]. Choi *et al.* [13] showed that emotional expressions of virtual agents can cause physiological responses in the person interacting with them. An extension of using physiological sensors could be to provide robots with additional context and perception as to how its interactions affect the human. Reactions to improve arousal and reduce stress with the robot would aim to improve social connections.

Health Care Scenarios

Health care applications are an obvious extension of wearable technology, with enormous social benefits associated. Wearable devices such as the *Apple Watch* and *Fitbit* already provide support for leading users through exercise and meditation scenarios. It would be interesting to use such sensors alongside an adaptive assistive robot to promote well-being in older populations. Fasola and Matarić [29] developed both a virtual and robotic exercise coach for use with older populations. The authors found that both mediums provided an engaging scenario to promote well-being. Described earlier in this chapter, Muñoz *et al.* [66] developed a physiologically-augmented exergame for older populations that provides real-time difficulty adaptations based on the heart rate measured from a wearable device.

A. Sharkey and N. Sharkey [93] outline areas where the introduction of robotic applications to assist older populations would be appropriate and provide six ethical concerns which should greatly be considered in any sort of development. It is with great importance that any care a social robot provides is not a direct replacement for care from a human, however, it can be used to augment the interaction. Depending on the comfort and privacy perception of physiological sensors by older populations,

it could be used as a non-invasive form of well-being monitoring, which also maintains the dignity of the wearer.

Difficulty adjustment in rehabilitation tasks is another area well suited for real-time physiological measures. Ozkul *et al.* [70] compared two different algorithms for difficulty adjustments in a rehabilitation scenario. Physiological measurements were recorded in this scenario to provide objective evaluation alongside subjective ratings to compare the two adaptation mechanisms. Future studies in a similar rehabilitation task could potentially use the physiological measurements being recorded as another input feature for real-time adaptations in rehabilitation.

Health care scenarios offer novel situations to truly make positive differences in the lives of users. Physiologically endowed social robots provide a unique application space for future studies in facets of health care.

Self-Driving Vehicles

While not specifically related to social robots, autonomous vehicles pose a particularly interesting problem space to incorporate physiological signals. In a study relating to passenger comfort by Dillen *et al.* [19], electrodermal activity (EDA), was found to be a significant predictor of comfort and anxiety in passengers in their self-driving vehicle experimental scenario. The authors envisioned in their future works section a system that modulates passenger-specific preferred driving profiles based on implicit feedback from wearables. With the proliferation of smart wearable technology, it doesn't seem too far-reaching that your future *Apple Watch* might have a feature that connects to your future vehicle to communicate that the current driving style is causing discomfort.

Chapter 5

Conclusion

This thesis presents a general framework and an open-source software library for developing physiologically adaptive HRI scenarios. The `HRI Physio Lib` provides functionality to assist in the development of the necessary components of adaptive HRI scenarios, including *signal acquisition*, *processing and analysis*, *social robot and communication*, and *scenario and adaptation*. To showcase our framework and test the software library, we developed a simple scenario revolving around a physiologically aware exercise coach. This is followed by a supplemental discussion about other HRI domains where the addition of physiologically adaptive mechanisms could result in interesting advances in user-experience.

5.1 Summary of Contributions

In this section, we provide the research questions for this project, and a brief answer for each based on what has been done.

RQ1: How can we design a robotics software framework for the rapid development of physiologically aware robots?

We chose to pursue a design that acted as a sort of roadmap or checklist for what components are necessary for developing physiologically adaptive HRI scenarios. We began by imagining a circuit starting with a person and ending up at a robot, which doesn't have an explicit connection to the person besides its affordances to provide

perceptual feedback. We broke this loop into four primary components to provide focus to the problems and challenges associated with each; they are *signal acquisition*, *processing and analysis*, *social robot and communication*, and *scenario and adaptation*. The first component primarily dealt with the problems in acquiring real-time streams of data and provide recommendations for how to incorporate physiological data into a system. Next, we looked through the literature to understand what physiological signals are most frequently used, and what types of features would be useful for an adaptive system. For the robot portion, we identified six of what we believe are the most common feedback modalities which could be used in an adaptive scenario by a social robot. Lastly, we discussed what factors should be taken into consideration when designing the scenario itself, and what might be useful to keep in mind when developing the adaptations for the robot.

After providing the framework, we wanted to develop software that might be useful for prototyping adaptive scenarios. For our research group, it was important to not only consider ROS-based robots but to look at providing support for other platforms such as the iCub humanoid robot. It is with this that we formulated our second research question:

RQ2: How can we develop a robotics software framework capable of integrating various communication libraries?

Our first step in pursuit of answering this question was to choose a programming language. `C++` was the obvious choice as it is the primary development language for LSL, ROS, and YARP, as well as my most proficient. `Python` was used to rapidly develop auxiliary tools and modules, and `Java` was used to develop an application for Android devices. To ensure simple integration between physiological devices (likely using LSL to initially stream data out), we carefully considered a range of design patterns to implement tools to act as translators between communication libraries. A philosophy for our design of accessibility was to ensure roboticists did not have to do too much additional integration and technological investment to begin using physiological sensors in their projects.

The `HRI Physio Lib` is a software library that contains sets of abstract classes that can be derived from, as well as developed tools and applications with the purpose of being used for prototyping and quickly creating adaptive scenarios. Implementing software is a feat, but we felt we needed to show how the software could be used in tandem with the framework to structure an adaptive scenario, which brought us to the third research question:

RQ3: Is the proposed software framework and HRI Physio Lib able to be used for the rapid development of physiologically aware robots capable of adapting their interactions?

We aimed to design a scenario with a clear and simple premise to act as a "base case" for demonstrating how an adaptive scenario could be implemented. Exercise lead by a social robot seemed like an exciting focus, as the abundance of literature on robot-mediated exercise provided a convincing foundation to build off of. Previously developed physiologically adaptive scenarios in HCI have utilized exercise as a focus with motivations rooted in health and engagement to improve user-experience.

We developed our prototype scenario over the course of a little over a month and tried a pilot run with one of the postdoctoral fellows involved in the project. Running the experiment revealed a number of areas for improvement, as well as extensions for a future study. Details for this were presented in section 4.3.

5.2 Lessons Learned

Working with physiological sensors can be arduous, requiring a great deal of technical investment. It could, however, be an investment that propels HRI research into exciting new territories, providing social robots with newfound abilities to adapt and infer complex passive emotional states of their human interaction partner, ultimately with the coal of improving the quality of the interaction.

Engineering a system from a perspective of generalizability is a difficult endeavor. The design of our framework was a task of providing a roadmap for future HRI research, as a lot of the exciting experimental scenarios which we wanted to pursue at the beginning of this project were simply not possible due to the COVID-19 pandemic.

Auxiliary to the research itself, the challenges of disjointly working as a team in a global pandemic proved difficult in the beginning. When it became apparent that in-person studies would not be feasible for the foreseeable future the project itself needed to be pivoted and a different direction chosen; thus, we set our attention on conceptual ideas and frameworks. The true lessons learned in this project are rooted in adapting and overcoming these challenges. The support of supervision, mentors, and friends alike has proven to be vital in the success of this project.

5.3 Limitations

This framework is obviously not a blanket solution for developing physiologically adaptive scenarios but instead is designed to be a foundation from which development can begin. This work has been developed with extension in mind. It is for that reason the *scenario and adaptation* component of Fig. 3.3 might feel lacking. This component is for all intents and purposes up to the designer, as HRI scenarios are ultimately too contextual to what knowledge is trying to be gained.

The framework we've developed is a high-level idea about what necessary components should be incorporated when designing a physiologically adaptive scenario. Software that we've developed might not be right for all scenarios, but it might be helpful for the initial prototyping of a scenario. Our design philosophy for the software library was based on modularity and reusability; so that designers can pick and choose which assets suit their needs and develop what was missing.

In section 4.3.1 we highlighted some of the limitations that we believed were present in the implementation of the QT Cardio-Aware Exercise Coach scenario and provided recommendations for how we'd like to improve on the design.

5.4 Future Directions

Future directions can be broken up into two groups: project level, and detail level. For the project level, section 4.4 provides a discussion for how existing HRI domains can be augmented by physiologically adaptive robots. We outlined engagement, social connections, health care, and self-driving vehicles as topics this work can be extended towards. As well, we provided several details in section 4.3 for future extensions of the QT Cardio-Aware Exercise Coach scenario, and what changes can be made for developing the prototype into a full human participant study.

For detail, there are several changes that can be made to improve the software structure. For starters, providing a Docker[1] container of the repository for *releases* would be an asset for both deployments, as well as providing a common development environment. Further development of data acquisition tools would not only help streamline the process of connecting physiological data into the decision loop of robots but can also benefit unrelated research areas in incorporating physiological

[1] https://www.docker.com/

data into their systems. Finally, further expanding the methods provided in the *processing* namespace (described in section 3.1.2) would be of great use.

Additional directions for our framework may be to include aspects of Edge Computing [94] and Cloud Computing [41]. In many home and health care scenarios, it is unlikely that adequate processing power will be readily available. For this reason, a great deal of development has been done to offload data processing to specialized external locations. Because we are working with physiological (health) data, it is also important to consider the privacy and sensitivity of any data that *leaves* the network. As processing becomes more sophisticated and demanding, simple mobile devices and local machines may be unable to support the required processing. As a result, it may be necessary to consider and draw a line for what data is acceptable to transmit for external processing, and what data can be locally computed.

References

[1] Mojtaba Khomami Abadi, Ramanathan Subramanian, Seyed Mostafa Kia, Paolo Avesani, Ioannis Patras, and Nicu Sebe. DECAF: Meg-based multimodal database for decoding affective physiological responses. *IEEE Transactions on Affective Computing*, 6(3):209–222, July 2015.

[2] Arthur Aron, Elaine N. Aron, and Danny Smollan. Inclusion of other in the self scale and the structure of interpersonal closeness. *Journal of Personality and Social Psychology*, 63(4):596–612, 1992.

[3] Alexander Mois Aroyo, T Kyohei, Tora Koyama, Hideyuki Takahashi, Francesco Rea, Alessandra Sciutti, Yuichiro Yoshikawa, Hiroshi Ishiguro, and Giulio Sandini. Will people morally crack under the authority of a famous wicked robot? In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 35–42. IEEE, August 2018.

[4] Soumya C. Barathi, Michael Proulx, Eamonn O'Neill, and Christof Lutteroth. Affect recognition using psychophysiological correlates in high intensity VR exergaming. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–15. ACM, April 2020.

[5] Christoph Bartneck, Dana Kulić, Elizabeth Croft, and Susana Zoghbi. Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *International Journal of Social Robotics*, 1(1):71–81, November 2008.

[6] Diana Batista, Hugo Plácido da Silva, Ana Fred, Carlos Moreira, Margarida Reis, and Hugo Alexandre Ferreira. Benchmarking of the BITalino biomedical toolkit against an established gold standard. *Healthcare Technology Letters*, 6(2):32–36, March 2019.

[7] Ashwin Belle, Rosalyn Hobson, and Kayvan Najarian. A physiological signal processing system for optimal engagement and attention detection. In *2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW)*, pages 555–561. IEEE, November 2011.

[8] Dayi Bian, Joshua Wade, Amy Swanson, Amy Weitlauf, Zachary Warren, and Nilanjan Sarkar. Design of a physiology-based adaptive virtual reality driving platform for individuals with ASD. *ACM Transactions on Accessible Computing (TACCESS)*, 12(1):1–24, February 2019.

[9] John Brooke. System usability scale (SUS): a quick-and-dirty method of system evaluation user information. *Reading, UK: Digital Equipment Co Ltd*, 43:1–7, 1986.

[10] John T Cacioppo, Louis G Tassinary, and Gary Berntson. *Handbook of psychophysiology*. Cambridge university press, 2 edition, 2007.

[11] Kelly Caine, Selma Šabanovic, and Mary Carter. The effect of monitoring by cameras and robots on the privacy enhancing behaviors of older adults. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction - HRI '12*, pages 343–350. ACM Press, 2012.

[12] Colleen M Carpinella, Alisa B Wyman, Michael A Perez, and Steven J Stroessner. The robotic social attributes scale (RoSAS). In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 254–262. ACM, March 2017.

[13] Ahyoung Choi, Celso De Melo, Woontack Woo, and Jonathan Gratch. Affective engagement to emotional facial expressions of embodied social agents in a decision-making game. *Computer Animation and Virtual Worlds*, 23(3-4):331–342, May 2012.

[14] Derek Cormier, Gem Newman, Masayuki Nakane, James E Young, and Stephane Durocher. Would you do as a robot commands? an obedience study for human-robot interaction. In *International Conference on Human-Agent Interaction*, 2013.

[15] Lee J. Corrigan, Christina Basedow, Dennis Küster, Arvid Kappas, Christopher Peters, and Ginevra Castellano. Mixing implicit and explicit probes: Finding a ground truth for engagement in social human-robot interactions. In

Proceedings of the 2014 9th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 140–141. ACM, March 2014.

[16] Hugo Plácido da Silva, Ana Fred, and Raúl Martins. Biosignals for everyone. *IEEE Pervasive Computing*, 13(4):64–71, October 2014.

[17] Kerstin Dautenhahn, Chrystopher L. Nehaniv, Michael L. Walters, Ben Robins, Hatice Kose-Bagci, N. Assif Mirza, and Mike Blow. KASPAR – a minimally expressive humanoid robot for human-robot interaction research. *Applied Bionics and Biomechanics*, 6(3-4):369–397, December 2009.

[18] Frans B.M. de Waal. Putting the altruism back into altruism: The evolution of empathy. *Annual Review of Psychology*, 59(1):279–300, January 2008.

[19] Nicole Dillen, Marko Ilievski, Edith Law, Lennart E. Nacke, Krzysztof Czarnecki, and Oliver Schneider. Keep calm and ride along: Passenger comfort and anxiety as physiological responses to autonomous driving styles. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13. ACM, April 2020.

[20] Charles Dodge and Thomas A Jerse. *Computer Music: Synthesis, Composition, and Performance*. Macmillan Library Reference, 1985.

[21] Kevin Doherty and Gavin Doherty. Engagement in HCI: Conception, theory and measurement. *ACM Computing Surveys (CSUR)*, 51(5):1–39, January 2019.

[22] Sidney D'Mello, Blair Lehman, Jeremiah Sullins, Rosaire Daigle, Rebekah Combs, Kimberly Vogt, Lydia Perkins, and Art Graesser. A time for emoting: When affect-sensitivity is and isn't effective at promoting deep learning. In *International Conference on Intelligent Tutoring Systems*, pages 245–254. Springer Berlin Heidelberg, 2010.

[23] Gernot Ernst. *Heart Rate Variability*. Springer, 2014.

[24] Irfan A. Essa and Alex Paul Pentland. Coding, analysis, interpretation, and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):757–763, July 1997.

[25] Stephen Fairclough and Kiel Gilleade. Construction of the biocybernetic loop: A case study. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction - ICMI '12*, pages 571–578. ACM Press, 2012.

[26] Stephen H. Fairclough. Fundamentals of physiological computing. *Interacting with Computers*, 21(1-2):133–145, January 2009.

[27] Jing Fan, Dayi Bian, Zhi Zheng, Linda Beuscher, Paul A. Newhouse, Lorraine C. Mion, and Nilanjan Sarkar. A robotic coach architecture for elder care (ROCARE) based on multi-user engagement models. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(8):1153–1163, Aug 2017.

[28] Yin Fan, Xiangju Lu, Dian Li, and Yuanliu Liu. Video-based emotion recognition using CNN-RNN and C3D hybrid networks. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*. ACM, October 2016.

[29] Juan Fasola and Maja J Matarić. A socially assistive robot exercise coach for the elderly. *Journal of Human-Robot Interaction*, 2(2):3–32, June 2013.

[30] Oliver Faust, Yuki Hagiwara, Tan Jen Hong, Oh Shu Lih, and U Rajendra Acharya. Deep learning for healthcare applications based on physiological signals: A review. *Computer Methods and Programs in Biomedicine*, 161:1–13, July 2018.

[31] Tobias Fischer, Jordi-Ysard Puigbò, Daniel Camilleri, Phuong D. H. Nguyen, Clément Moulin-Frier, Stéphane Lallée, Giorgio Metta, Tony J. Prescott, Yiannis Demiris, and Paul F. M. J. Verschure. iCub-HRI: A software framework for complex human–robot interaction scenarios on the iCub humanoid robot. *Frontiers in Robotics and AI*, 5:22, March 2018.

[32] Mary Ellen Foster, Andre Gaschler, and Manuel Giuliani. Automatically classifying user engagement for dynamic multi-party human–robot interaction. *International Journal of Social Robotics*, 9(5):659–674, July 2017.

[33] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, and Design Patterns. *Elements of Reusable Object-Oriented Software*, volume 99. Addison-Wesley Reading, Massachusetts, 1995.

[34] Ann L Gibson, Dale Wagner, and Vivian Heyward. *Advanced Fitness Assessment and Exercise Prescription, 8E*. Human kinetics, 2018.

[35] Rahel Gilgen-Ammann, Theresa Schweizer, and Thomas Wyss. RR interval signal quality of a heart rate monitor and an ECG holter at rest and during exercise. *European Journal of Applied Physiology*, 119(7):1525–1532, April 2019.

[36] Jonas Gonzalez-Billandon, Alexander M. Aroyo, Alessia Tonelli, Dario Pasquali, Alessandra Sciutti, Monica Gori, Giulio Sandini, and Francesco Rea. Can a robot catch you lying? a machine learning system to detect lies during interactions. *Frontiers in Robotics and AI*, 6, July 2019.

[37] Hatice Gunes and Massimo Piccardi. Bi-modal emotion recognition from expressive face and body gestures. *Journal of Network and Computer Applications*, 30(4):1334–1345, November 2007.

[38] Fu Guo, Mingming Li, Qingxing Qu, and Vincent G. Duffy. The effect of a humanoid robot's emotional behaviors on users' emotional responses: Evidence from pupillometry and electroencephalography measures. *International Journal of Human–Computer Interaction*, 35(20):1947–1959, March 2019.

[39] Katrin Hänsel, Romina Poguntke, Hamed Haddadi, Akram Alomainy, and Albrecht Schmidt. What to put on the user: Sensing technologies for studies and physiology aware systems. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14. ACM, April 2018.

[40] Sandra G. Hart and Lowell E. Staveland. Development of NASA-TLX (task load index): Results of empirical and theoretical research. In *Advances in Psychology*, pages 139–183. Elsevier, 1988.

[41] Brian Hayes. Cloud computing. *Communications of the ACM*, 51(7):9–11, July 2008.

[42] Lawrence J. Hettinger, Pedro Branco, L. Miguel Encarnacao, and Paolo Bonato. Neuroadaptive technologies: Applying neuroergonomics to the design of advanced interfaces. *Theoretical Issues in Ergonomics Science*, 4(1-2):220–237, January 2003.

[43] Kazuko Itoh, Hiroyasu Miwa, Yuko Nukariya, Massimiliano Zecca, Hideaki Takanobu, Stefano Roccella, Maria Carrozza, Paolo Dario, and Atsuo Takanishi. Development of a bioinstrumentation system in the interaction between a human and a robot. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2620–2625. IEEE, October 2006.

[44] Kevin Jeffay. The real-time producer/consumer paradigm: A paradigm for the construction of efficient, predictable real-time systems. In *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice - SAC '93*, pages 796–804. ACM Press, 1993.

[45] Peter H. Kahn, Takayuki Kanda, Hiroshi Ishiguro, Nathan G. Freier, Rachel L. Severson, Brian T. Gill, Jolina H. Ruckert, and Solace Shen. "robovie, you'll have to go into the closet now": Children's social and moral relationships with a humanoid robot. *Developmental Psychology*, 48(2):303–314, March 2012.

[46] Eugenijus Kaniusas. *Biomedical Signals and Sensors I: Linking Physiological Phenomena and Biosignals*. Springer Berlin Heidelberg, 2012.

[47] Yuki Kayukawa, Yasutake Takahashi, Takuya Tsujimoto, Kazunori Terada, and Hiroyuki Inoue. Influence of emotional expression of real humanoid robot to human decision-making. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE, IEEE, July 2017.

[48] Johannes Keller, Herbert Bless, Frederik Blomann, and Dieter Kleinböhl. Physiological aspects of flow experiences: Skills-demand-compatibility effects on heart rate variability and salivary cortisol. *Journal of Experimental Social Psychology*, 47(4):849–852, July 2011.

[49] Sander Koelstra, Christian Muhl, Mohammad Soleymani, Jong-Seok Lee, Ashkan Yazdani, Touradj Ebrahimi, Thierry Pun, Anton Nijholt, and Ioannis Patras. DEAP: A database for emotion analysis; using physiological signals. *IEEE Transactions on Affective Computing*, 3(1):18–31, January 2012.

[50] Austin Kothig, John Muñoz, Sami Alperen Akgun, Alexander M. Aroyo, and Kerstin Dautenhahn. Connecting humans and robots using physiological signals – closing-the-loop in HRI. In *2021 30th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, August 2021.

[51] Austin Kothig, John Muñoz, Hamza Mahdi, Alexander M. Aroyo, and Kerstin Dautenhahn. HRI Physio Lib: A software framework to support the integration of physiological adaptation in HRI. In *International Conference on Social Robotics*, pages 36–47. Springer International Publishing, November 2020.

[52] Jamy Li and Mark Chignell. Communication of emotion in social robots through simple head and arm movements. *International Journal of Social Robotics*, 3(2):125–142, September 2011.

[53] Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22:55, 1932.

[54] Nico Loewe and Mario Nadj. Physio-adaptive systems – a state-of-the-art review and future research directions. In *28th European Conference on Information Systems (ECIS)*, 2020.

[55] Tatjana Loncar-Turukalo, Eftim Zdravevski, José Machado da Silva, Ioanna Chouvarda, and Vladimir Trajkovik. Literature on wearable technology for connected health: Scoping review of research trends, advances, and barriers. *Journal of Medical Internet Research (JMIR)*, 21(9):e14017, September 2019.

[56] Mark W Maier, David Emery, and Rich Hilliard. Software architecture: Introducing IEEE standard 1471. *Computer*, 34(4):107–109, April 2001.

[57] Dominique Makowski, Tam Pham, Zen J Lau, Jan C. Brammer, François Lespinasse, Hung Pham, Christopher Schölzel, and S. H. Annabel Chen. NeuroKit2: A Python toolbox for neurophysiological signal processing. *Behavior Research Methods*, 53(4):1689–1696, February 2021.

[58] Amir Matallaoui, Jonna Koivisto, Juho Hamari, and Ruediger Zarnekow. How effective is "exergamification"? a systematic review on the effectiveness of gamification features in exergames. In *Proceedings of the 50th Hawaii International Conference on System Sciences (2017)*. Hawaii International Conference on System Sciences, 2017.

[59] Derek McColl and Goldie Nejat. Recognizing emotional body language displayed by a human-like social robot. *International Journal of Social Robotics*, 6(2):261–280, January 2014.

[60] Albert Mehrabian. Pleasure-Arousal-Dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, 14(4):261–292, December 1996.

[61] Juan Abdon Miranda-Correa, Mojtaba Khomami Abadi, Nicu Sebe, and Ioannis Patras. AMIGOS: A dataset for affect, personality and mood research on individuals and groups. *IEEE Transactions on Affective Computing*, 12(2):479–493, April 2018.

[62] Vivian Genaro Motti and Kelly Caine. Users' privacy concerns about wearables. In *International Conference on Financial Cryptography and Data Security*, pages 231–244. Springer Berlin Heidelberg, 2015.

[63] Sankha S. Mukherjee and Neil Martin Robertson. Deep Head Pose: Gaze-direction estimation in multimodal video. *IEEE Transactions on Multimedia*, 17(11):2094–2107, November 2015.

[64] S. K. Mukhopadhyay, M. Mitra, and S. Mitra. Time plane ECG feature extraction using Hilbert transform, variable threshold and slope reversal approach. In *2011 International Conference on Communication and Industrial Application*, pages 1–4. IEEE, December 2011.

[65] J. E. Muñoz, E. R. Gouveia, M. Cameirão, and S. Bermúdez I. Badia. The Biocybernetic Loop Engine: An integrated tool for creating physiologically adaptive videogames. In *Proceedings of the 4th International Conference on Physiological Computing Systems*, pages 45–54. SCITEPRESS - Science and Technology Publications, 2017.

[66] John E. Muñoz, M. Cameirão, S. Bermúdez i Badia, and E. Rubio Gouveia. Closing the loop in exergaming – health benefits of biocybernetic adaptation in senior adults. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, pages 329–339. ACM, October 2018.

[67] Hong-Wei Ng, Viet Dung Nguyen, Vassilios Vonikakis, and Stefan Winkler. Deep learning for emotion recognition on small datasets using transfer learning. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 443–449. ACM, November 2015.

[68] Phuong D.H. Nguyen, Fabrizio Bottarel, Ugo Pattacini, Matej Hoffmann, Lorenzo Natale, and Giorgio Metta. Merging physical and social interaction for effective human-robot collaboration. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, November 2018.

[69] Catharine Oertel, Ginevra Castellano, Mohamed Chetouani, Jauwairia Nasir, Mohammad Obaid, Catherine Pelachaud, and Christopher Peters. Engagement in human-agent interaction: An overview. *Frontiers in Robotics and AI*, 7:92, August 2020.

[70] Fatih Ozkul, Yunus Palaska, Engin Masazade, and Duygun Erol-Barkana. Exploring dynamic difficulty adjustment mechanism for rehabilitation tasks using physiological measures and subjective ratings. *Institution of Engineering and Technology (IET) Signal Processing*, 13(3):378–386, May 2019.

[71] Ana Paiva, Iolanda Leite, Hana Boukricha, and Ipke Wachsmuth. Empathy in virtual agents and robots. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 7(3):1–40, October 2017.

[72] Evan M. Peck, Eleanor Easse, Nick Marshall, William Stratton, and L. Felipe Perrone. FlyLoop: A micro framework for rapid development of physiological computing systems. In *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 152–157. ACM, June 2015.

[73] Rosalind W. Picard. *Affective Computing*. Cambridge, MA: MIT Press, 2000.

[74] J.A. Pineda and E. Hecht. Mirroring and mu rhythm involvement in social cognition: Are there dissociable subcomponents of theory of mind? *Biological Psychology*, 80(3):306–314, March 2009.

[75] Pierre Rainville, Antoine Bechara, Nasir Naqvi, and Antonio R. Damasio. Basic emotions are associated with distinct patterns of cardiorespiratory activity. *International Journal of Psychophysiology*, 61(1):5–18, July 2006.

[76] Pramila Rani, Changchun Liu, Nilanjan Sarkar, and Eric Vanman. An empirical study of machine learning techniques for affect recognition in human-robot interaction. *Pattern Analysis and Applications*, 9(1):58–69, April 2006.

[77] Pramila Rani and Nilanjan Sarkar. Operator engagement detection and robot behavior adaptation in human-robot interaction. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005.

[78] Stefan Rank and Cathy Lu. PhysSigTK: Enabling engagement experiments with physiological signals for game design. In *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 968–969. IEEE, September 2015.

[79] Daniel J. Rea, James E. Young, and Pourang Irani. The roomba mood ring: An ambient-display robot. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction - HRI '12*, pages 217–218. ACM Press, 2012.

[80] Yann Renard, Fabien Lotte, Guillaume Gibert, Marco Congedo, Emmanuel Maby, Vincent Delannoy, Olivier Bertrand, and Anatole Lécuyer. OpenViBE: An open-source software platform to design, test, and use brain–computer interfaces in real and virtual environments. *Presence: Teleoperators and Virtual Environments*, 19(1):35–53, February 2010.

[81] Raphael Rissler, Mario Nadj, Maximilian Xiling Li, Michael Thomas Knierim, and Alexander Maedche. Got Flow? using machine learning on physiological data to classify flow. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–6. ACM, April 2018.

[82] Cristina Romero-Blanco, Julián Rodríguez-Almagro, María Dolores Onieva-Zafra, María Laura Parra-Fernández, María Del Carmen Prado-Laguna, and Antonio Hernández-Martínez. Physical activity and sedentary lifestyle in university students: Changes during confinement due to the COVID-19 pandemic. *International Journal of Environmental Research and Public Health*, 17(18):6567, Sep 2020.

[83] Raphaëlle N Roy, Nicolas Drougard, Thibault Gateau, Frédéric Dehais, and Caroline P. C. Chanel. How can physiological computing benefit human-robot interaction? *Robotics*, 9(4):100, November 2020.

[84] James A. Russell, Anna Weiss, and Gerald A. Mendelsohn. Affect Grid: A single-item scale of pleasure and arousal. *Journal of Personality and Social Psychology*, 57(3):493–502, 1989.

[85] Richard M. Ryan. Control and information in the intrapersonal sphere: An extension of cognitive evaluation theory. *Journal of Personality and Social Psychology*, 43(3):450–461, 1982.

[86] Selma Šabanović, Casey C Bennett, Wan-Ling Chang, and Lesa Huber. PARO robot affects diverse interaction modalities in group sensory therapy for older adults with dementia. In *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*, pages 1–6. IEEE, June 2013.

[87] Tanja Schneeberger, Sofie Ehrhardt, Manuel S. Anglet, and Patrick Gebhard. Would you follow my instructions if i was not human? examining obedience towards virtual agents. In *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 1–7. IEEE, September 2019.

[88] Sebastian Schneider and Franz Kummert. Comparing the effects of social robots and virtual agents on exercising motivation. In *International Conference on Social Robotics (ICSR)*, pages 451–461. Springer, Springer International Publishing, 2018.

[89] Sebastian Schneider and Franz Kummert. Comparing robot and human guided personalization: Adaptive exercise robots are perceived as more competent and

trustworthy. *International Journal of Social Robotics (ICSR)*, 13(2):169–185, February 2020.

[90] Fred Shaffer and Jay P. Ginsberg. An overview of heart rate variability metrics and norms. *Frontiers in Public Health*, 5:258, September 2017.

[91] Caifeng Shan, Shaogang Gong, and Peter W. McOwan. Beyond facial expressions: Learning human emotion from body gestures. In *Procedings of the British Machine Vision Conference 2007*, pages 1–10. British Machine Vision Association, 2007.

[92] Mingyang Shao, Matt Snyder, Goldie Nejat, and Beno Benhabib. User affect elicitation with a socially emotional robot. *Robotics*, 9(2):44, June 2020.

[93] Amanda Sharkey and Noel Sharkey. Granny and the robots: Ethical issues in robot care for the elderly. *Ethics and Information Technology*, 14(1):27–40, July 2012.

[94] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, October 2016.

[95] Mohammad Soleymani, Jeroen Lichtenauer, Thierry Pun, and Maja Pantic. A multimodal database for affect recognition and implicit tagging. *IEEE Transactions on Affective Computing*, 3(1):42–55, January 2011.

[96] Ramanathan Subramanian, Julia Wache, Mojtaba Khomami Abadi, Radu L. Vieriu, Stefan Winkler, and Nicu Sebe. ASCERTAIN: Emotion and personality recognition using commercial sensors. *IEEE Transactions on Affective Computing*, 9(2):147–160, April 2018.

[97] Luise Süssenbach, Nina Riether, Sebastian Schneider, Ingmar Berger, Franz Kummert, Ingo Lütkebohle, and Karola Pitsch. A robot as fitness companion: Towards an interactive action-based motivation model. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 286–293. IEEE, August 2014.

[98] Mojtaba Jafari Tadi, Eero Lehtonen, Tero Hurnanen, Juho Koskinen, Jonas Eriksson, Mikko Pänkäälä, Mika Teräs, and Tero Koivisto. A real-time approach for heart rate monitoring using a hilbert transform in seismocardiograms. *Physiological Measurement*, 37(11):1885–1909, September 2016.

[99] Joachim Taelman, Steven Vandeput, Arthur Spaepen, and Sabine Van Huffel. Influence of mental stress on heart rate and heart rate variability. In *4th European Conference of the International Federation for Medical and Biological Engineering (IFMBE) Proceedings*, pages 1366–1369. Springer Berlin Heidelberg, 2009.

[100] Hirofumi Tanaka, Kevin D Monahan, and Douglas R Seals. Age-predicted maximal heart rate revisited. *Journal of the American College of Cardiology*, 37(1):153–156, January 2001.

[101] Emily A Vogels. About one-in-five americans use a smart watch or fitness tracker, January 2020. https://pewrsr.ch/37IaaN4. Accessed 2021-08-17.

[102] Joshua Wainer, Ben Robins, Farshid Amirabdollahian, and Kerstin Dautenhahn. Using the humanoid robot kaspar to autonomously play triadic games and facilitate collaborative play among children with autism. *IEEE Transactions on Autonomous Mental Development*, 6(3):183–199, September 2014.

[103] Beverly Woolf, Winslow Burleson, Ivon Arroyo, Toby Dragon, David Cooper, and Rosalind Picard. Affect-aware tutors: Recognising and responding to student affect. *International Journal of Learning Technology*, 4(3-4):129–164, 2009.

[104] Deborah Rohm Young, Marie-France Hivert, Sofiya Alhassan, Sarah M. Camhi, Jane F. Ferguson, Peter T. Katzmarzyk, Cora E. Lewis, Neville Owen, Cynthia K. Perry, Juned Siddique, and Celina M. Yong. Sedentary behavior and cardiovascular morbidity and mortality: A science advisory from the american heart association. *Circulation*, 134(13):e262–e279, September 2016.

[105] Chen Zheng, Wendy Yajun Huang, Sinead Sheridan, Cindy Hui-Ping Sit, Xiang-Ke Chen, and Stephen Heung-Sang Wong. COVID-19 pandemic brings a sedentary lifestyle in young adults: A cross-sectional and longitudinal study. *International Journal of Environmental Research and Public Health*, 17(17):6035, August 2020.

[106] Jianlong Zhou, Jinjun Sun, Fang Chen, Yang Wang, Ronnie Taib, Ahmad Khawaji, and Zhidong Li. Measurable decision making with GSR and pupillary analysis for intelligent user interface. *ACM Transactions on Computer-Human Interaction (ToCHI)*, 21(6):1–23, January 2015.