DESIGN AND EVALUATION OF A NONVERBAL COMMUNICATION PLATFORM BETWEEN ASSISTIVE ROBOTS AND THEIR USERS

Anthony Threatt
Clemson University, tony.threatt@gmail.com

DESIGN AND EVALUATION OF A NONVERBAL COMMUNICATION PLATFORM BETWEEN ASSISTIVE ROBOTS AND THEIR USERS

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Planning, Design, and the Built Environment

by
Anthony Lee Threatt
August 2013

ABSTRACT

Assistive robotics will become integral to the everyday lives of a human population that is increasingly mobile, older, urban-centric and networked. The overwhelming demands on healthcare delivery alone will compel the adoption of assistive robotics. *How will we communicate with such robots, and how will they communicate with us?* This research makes the case for a relatively "artificial" mode of nonverbal human-robot communication that is non-disruptive, non-competitive, and non-invasive human-robot communication that we envision will be willingly invited into our private and working lives over time. This research proposes a non-verbal communication (NVC) platform be conveyed by familiar lights and sounds, and elaborated here are experiments with our NVC platform in a rehabilitation hospital. This NVC is embedded into the Assistive Robotic Table (ART), developed within our lab, that supports the well-being of an expanding population of older adults and those with limited mobility. The broader aim of this research is to afford people robot-assistants that exist and interact with them in the recesses, rather than in the foreground, of their intimate and social lives.

With support from our larger research team, I designed and evaluated several alternative modes of nonverbal robot communication with the objective of establishing a nonverbal, human-robot communication loop that evolves with users and can be modified by users. The study was conducted with 10-13 clinicians -- doctors and occupational, physical, and speech therapists -- at a local rehabilitation hospital through three iterative design and evaluation phases and a final usability study session. For our test case at a rehabilitation hospital, medical staff iteratively refined our NVC platform, stated a willingness to use our platform, and declared NVC as a desirable research path. In addition, these clinicians provided the requirements for human-robot interaction (HRI) in clinical settings, suggesting great promise for our mode of human-robot communication for this and other applications and environments involving intimate HRI.

DEDICATION

This dissertation is dedicated to my Mom and Dad, who encouraged me to always do my best throughout 25 years of education and beyond; my sister, who is an inspiration as her wit is only matched by her compassion; Grandma and Papa, who have unconditionally provided me and my family with love, support, and shelter; my extended family, who have always been inquisitive about my research; Melissa, Lillian, and Remy - thank you for your love, dedication, support, and patience throughout my graduate studies. I am eternally grateful for the sacrifices that you have made so that I could accomplish my academic aspirations. I love you all.

ACKNOWLEDGMENTS

I would first like to thank Dr. Keith Green for his guidance and support throughout my graduate studies. His insights and willingness for exploratory research have provided me the opportunity to accomplish my greatest academic achievement. This work would not be possible without the efforts of my committee members. I thank Dr. Ian Walker for his insights and encouragement, Dr. Johnell Brooks for her willingness to teach, explore, participate, and laugh, and Dr. Michael Ellison for always pushing me to prove him wrong.

This work would not have been made possible without Jessica Merino. Jessica is more than a colleague and friend - she has become family. Thank you for your efforts, your insights, your guidance, and your willingness to always push me and this work to its greatest possible conclusion.

A big thank you to the Architectural Robotics lab including: Artemiy Zheltov, Apoorva Kapadia, Joe Manganelli, Paul Yanik, Bryan Williman, and Ninad Pradhan. I have appreciated learning from, working with, and commiserating with you all.

Finally, I would like to thank the members of the Roger C. Peace Rehabilitation Hospital of the Greenville Health System -- Dr. Stan Healy, Dr. Kevin Kopera, Jeremy McKee, Melissa Youseff, Kylie Sprogis, Dr. Matt Crisler, Jimmy Bacon, Brad Looper, Yubin Xi and the doctors and physical, occupational, and speech therapists -- who helped make this research possible. This research was supported by the U.S. National Science Foundation under award IISSHB-116075.

TABLE OF CONTENTS

Table of Contents (Continued)

Table of Contents (Continued)

Table of Contents (Continued)

Page

LIST OF TABLES

List of Tables (Continued)

Table Page

List of Tables (Continued)

Table Page

LIST OF FIGURES

Figure Page

List of Figures (Continued)

CHAPTER ONE

INTRODUCTION AND PROBLEM STATEMENT

1.1 The motivation for nonverbal communication in assistive robotics

Inevitably, assistive robotics will become ubiquitous in the everyday lives of a human population that is increasingly mobile, older, urban-centric, and networked. The overwhelming demands on healthcare delivery alone will compel the adoption of assistive robotics. However, for assistive robotics, there exists a vast gap between humanoid robots (anthropomorphic, increasingly multi-functional, and intelligent) and the Roomba®, the "killer app" of robotics (formally abstract, single-purpose, and exhibiting lower-intelligence). Filling the gap between humanoids and the Roomba®, is the emergence of a class of assistive robots, "Architectural Robotics," cultivated by our lab in recent years (see Figure 1.1) (Green, 2008; Green, Gugerty, Walker, & Witte., 2006; 2005a; 2005, 2005b; Green, Walker, Brooks, Threatt, & Merino, 2011; Gross & Green, 2012). "Architectural Robotics" is a novel classification of cyber-physical artifacts of the built environment, at the scale of furniture or larger, that adapts and reconfigures to support the co-habitation of their users (Gross & Green, 2012). While offering far greater utility than the function-driven Roomba, Architectural-Robotic artifacts do not strive for human appearance or behavior as do humanoid robots, nor are they overtly imitative of other living things.

Particularly for assistive applications, Architectural-Robotics (and similar robots that might also fill this gap), have these virtues: 1.) they don't suffer the likelihood of users perceiving them as human and expecting them to act human; 2.) they don't suffer the possibility of falling into the "uncanny valley" (Mori, 1970); 3.) they don't compete with humans for attention; 4.) they don't participate in a "master-slave relationship" of human and humanoid; 5.) they aren't perceived as invading human privacy; and, 6.) they are not likely, operating on a lower cognitive level, to disrupt human-human communication. It is the latter of these virtues that frames the key objective of this research to design and evaluate an appropriate and effective nonverbal communication (NVC) platform for robots to communicate with people. This mode of communication, like Architectural-Robotics, dignifies what it is to be human by not competing with us, nor imposing on our social-emotional-cognitive constitution.

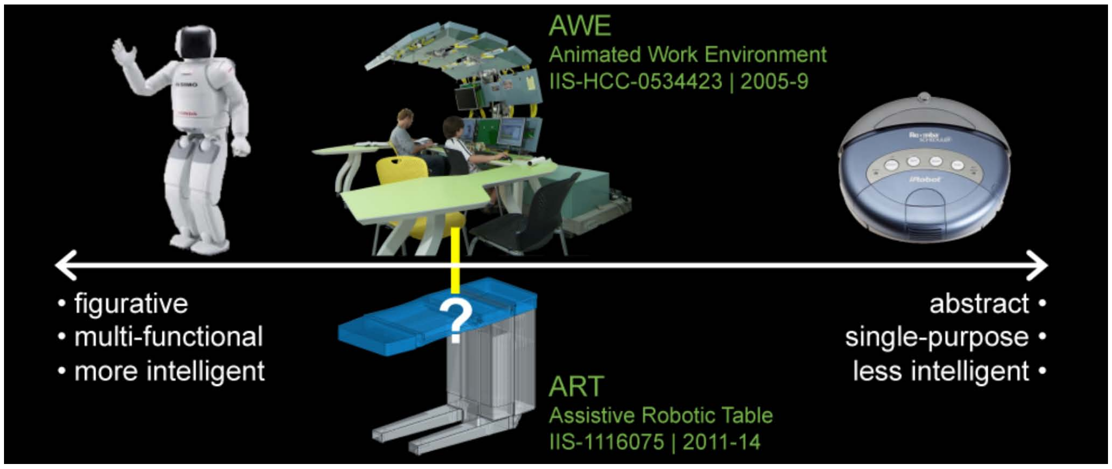


Figure 1.1: In the continuum of assistive robotics, there is a big gap between humanoid robots (as envisioned by many) and the likes of the Roomba. Our group is working in this vast in-between. How will we communicate with such robots, and how will they communicate with us?

It should be made clear that Architectural-Robotic artifacts (and other assistive robot platforms that may occupy the gap) do not preclude nor supplant their future co-existence with humanoid, pet, or single-function assistive robots. Tokens of all these robot classifications can comprise a future, expanded ecosystem of mutually-supportive, communicative, living, and artificial beings. Likewise, the NVC is envisioned as a promising candidate for employment in all classifications of robots, where it can overcome various technical challenges in robotics research and also be responsive to natural language and diverse human populations.

It is befitting that, for a computational artifact situated in the vast gap between Roomba® and humanoid, the mode of its human-robot communication should be more artificial than natural (i.e., human-like), to avoid unnecessary confusion and distraction in the lives of the computational artifact's users. The NVC elaborated herein is conveyed by the familiar means of audio-visual communication: low-cost lighting (colors and patterns) and sounds. This NVC is based on an understanding of cognitive, perceptual processes of non-verbal communication in humans and affords a communicative dialogue (i.e., acknowledging requests or providing requested feedback) that conveys the purpose of accomplishing tasks. Our employment of learning algorithms offers both user and robot the capacity to interrupt, query, and to correct the dialogue. This NVC conveys in the robot some semblance of emotional information (e.g., urgency, respect, or frustration) at a level that is not disconcerting and in a way the user might misconstrue as human (i.e. not as in Picard's sense of

"Affective Computing" [1995] that has inspired human-robot interaction (HRI) investigations employing humanoids that "match their mood" to human users, [Gonsior, Sosnowski, Buß, Wollherr & Kuhnlenz 2012]).

In a series of experiments, our larger research team designed and evaluated several alternative modes of nonverbal robot communication with the objective of establishing a nonverbal, human-robot communication loop that evolves with users and can be modified by users. This research is partly informed by American Sign Language (ASL) and particularly, its methods (Quenqua, 2012) for adding new vocabulary to what is the most-established form of NVC. We employed mixed methods that solicited user input and allowed the NVC to evolve (Quenqua, 2012). A significant part of the research involved integrating our NVC onto a specific Architectural-Robotic artifact, our own Assistive Robotic Table (ART) to evaluate human-robot communication up-close, "in-the-wild," at the Roger C. Peace Rehabilitation Hospital of the Greenville Health System (GHS) (described in Chapter 2). For our NVC platform, as for American Sign Language, it must be "tested and refined in everyday [exchanges to be] accepted quickly" (Quenqua, 2012, p. D1). ART supports post-stroke patient therapy and, more broadly, the well-being of an expanding population of older adults and those with limited mobility. The broader aim of this research is to afford people robot-assistants that exist and interact with them in the recesses, rather than in the foreground of their intimate and social lives. For many of us, this is the kind of non-disruptive, non-competitive, and non-invasive human-robot communication that we envision will be willingly invited into our private and working lives over time.

1.2 State-of-the-art assistive-robot communication and the need for NVC

This research is dedicated to forming a mode of nonverbal communication that an assistive robot "speaks" in its interaction with people. In the mid-20th century, Norbert Weiner reasoned that "it is quite possible for a person to talk to a machine, a machine to a person, and a machine to a machine" (Weiner, 1954, p. 76). What is the language of a robot, if not natural, spoken, human language? Briefly, what follows are the foundations for answering this question towards realizing the ambition of developing a nonverbal communication platform for assistive robotics.

1.2.1 Perceptual Primitives

For this research, the simple, nonverbal unit comprising the NVC is viewed as a "perceptual primitive." Perceptual primitives are "Gestalt-like primitive biases deriving from the architecture and functioning of the perceptual system" (Gervain & Mehler, 2007, p. 1). One way of understanding "perceptual primitive" is to distinguish it from the two, perhaps more familiar theories of language acquisition: the genetically-based, symbolic rule theory (identified with Chomsky); and the statistical learning theory, in which language is acquired piecemeal as evidence presents itself (identified with Elman and Tomasello) (Gervain & Mehler, 2007).

In the computing domain, the concept of "primitives" has been employed in vision research (Huang, Huang, Tan, & Tao, 2009) and in computer graphics (McNamara, 2009; Mezger, Winfried, & Giese, 2005). In these instances, primitives are used as a conceptual vehicle for capturing real world phenomena, as in evaluating the fidelity of simulated imagery relative to their real sources (McNamara, 2009), capturing the trajectories of more complex human body movements (e.g., "movement primitives" in karate) (Mezger et al., 2005), or within HRI research, facial recognition (Bartlett et al., 2005, 2006; Susskind, Hershey, & Movellan, 2004). Common to all these uses of perceptual primitives is a machine-centric perspective. That is, the concept of perceptual primitives is employed as a vehicle that captures the "outside" world of people, things and their physical surroundings. In this research we invert this relationship; the concept of perceptual primitives is employed as a vehicle for the robot to speak to us. The reasoning is if machines operate on the level of perceptual primitives, then it seems reasonable to employ the same kind of "robot-mindedness" to form the "language" a robot "speaks" to users. This kind of human-robot communication would establish two bandwidths of communication: 1.) a lower-bandwidth of sounds and lights for human-machine communication and 2.) a higher bandwidth of natural language for human-human communication. The lower bandwidth communication remains in the recesses of our lives, maintaining high-bandwidth human-human communication for human beings, conducting their private and social lives.

As noted here, perceptual primitives have received extensive attention in vision-related computer research; however, it is perceptual primitives as considered by neuro-cognitive research in its focus on language acquisition that is the scientific motivation for this investigation. In the first of a series of experiments conducted by Dehaene-Lambertz (Gervain & Mehler, 2007), newborn infants were presented with blocks representing two artificial grammars of three-syllable "sentences". The first of these artificial grammars had an ABB structure, where one syllable was immediately repeated (e.g. "mubaba", "tofefe", "pishosho", etc.); the second of these artificial grammars exhibited an ABC structure – without repetition (e.g., "mubafe", "tofesho", "pishoge", etc.). As measured by near infrared spectroscopy, the infants' brain activities in areas responsible for structural representation-building and integration show that the newborns immediately detected adjacent repetitions (i.e., ABB as distinct from ABC) as perceptual primitives, and that these perceptual primitives are later integrated by the infants into higher level, more general structural representations (Gervain & Mehler, 2007). As reported by Gervain and Mehler (2007), "these findings argue for the existence of auditory/linguistic perceptual primitives, such as representationally adjacent…repetitions. These primitive, Gestalt-like configurations are automatically detected by the auditory system; and in later processing, they get recruited as building blocks of higher-level representations" (p. 1). Forming a bridge between Weiner's ponderings in the middle of the last century and neuro-cognitive results like these, we recognize that human beings are genetically prepared to process nonverbal perceptual primitives and, indeed, begin doing so at birth. Curiously, it is in the movies that we discover the potential and delight of nonverbal communication by intelligent machines – that of R2D2, WALL-E, and the alien machine of Close Encounters of the Third Kind – that "speak" to us with combinations of lights and sounds. There is, however, a lack of understanding of how such nonverbal utterances could be employed in assistive robot communication (Read & Belpaeme, 2010). The intent of this research is to investigate how perceptual primitives generated by lights and sounds form a platform for NVC, particularly as it is embedded in our nonverbal, robot communication loop (Figure 1.2).

The iterative development of the NVC included multiple combinations and patterns of light-and-sound. The NVC is a human-robot communication loop situated in a real-world context--the Roger C. Peace Rehabilitation Hospital, inhabited by post-stroke patients, family members, and clinicians. The loop is comprised of: 1.) our developing NVC, 2.) our emergent (human) gesture recognition (as the NVC "spoken" by our human participants), and 3.) ART, an architectural robotic artifact of our design. In short in the real-world context of the Roger C. Peace Rehabilitation Hospital, post-stroke patients and their clinicians will communicate with the robot through gestures, which the robot learns (e.g., via the Growing Neural Gas (GNG) algorithm employed in our lab). In turn, the NVC of lights and sounds is "uttered" by the robot. If the human recipient should respond adversely to the qualities of the robot's utterance, the human recipient can gesture "in annoyance" to the robot, which will (through a learning algorithm) alter its "speech" to satisfy the human conversant, consequently rewarded by gesture or (haptic) "petting" of the robot – as do two people in an analogous human-human exchange. In this way, our human-robot communication constitutes a well-balanced, bi-directional mode of nonverbal communication. Our research team assumes that the user identified in our scenario was already conversant with our NVC prior to suffering a stroke; that the architectural robotic furniture (here "ART") was (as we envisioned) integral to the life of the healthy user at home. This implies that the user was literate in our NVC after suffering a stroke and is now making great use of the functionality of this technologically advanced piece of furniture that "grows" with its "co-inhabitant" – the user.

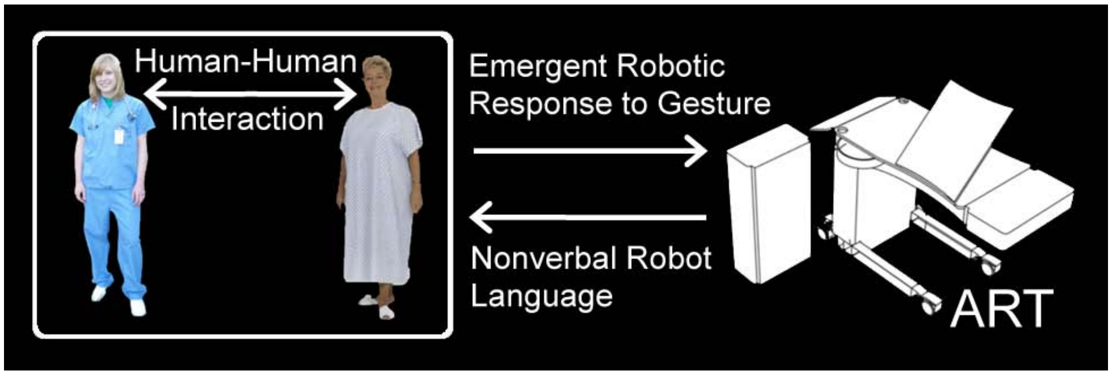


Figure 1.2: The proposed Human-Robot Communication Between ART, Post-Stroke Patients, and their Clinicians whereby human to human interaction is not interrupted by low-level communication between human and machine.

1.2.2 Nonverbal communication across bodies of knowledge

Nonverbal communication (NVC) is defined as communication outside formal language as used in the human acts of speaking and writing (Johansson, 1973). Ninety-three percent of human-human communication is said to be nonverbal communication (Johansson, 1973), not only comprised largely of body language and voice tonality but also, for instance, the visual characteristics of clothing worn by those engaged in social interaction. What is key, drawing from this body of literature, is the recognition that nonverbal communication is a rich, effective, and expedient form of communication; that more complex, rich "utterances" are comprised of simple units of communication (kinemes and morphemes being the minimum units); and that simple units of communication can be understood by themselves, without adding greater complexity, particularly if they are presented in a familiar social and/or environmental situation (i.e., context).

A foundational study of gestures and other forms of NVC for computational artifacts is found in the work of Justine Cassell (Cassell, 1998; Cassell et al., 1999) and contextualized by Rossini (2012) who defines nonverbal cues in robots as "Embodied Conversational Agents" (Cassell, 1998; Cassell et al., 1999, p. 520). Many investigations in HRI and many inspired by Cassell have explored the implementation of natural communication – the spoken word, often accompanied by gestures and other kinds of cues imitating human ones (Breazeal, et al., 2008; Cassell, 1998; Kirchner & Alempijevic, 2012; Lallée et al., 2010; Rossini, 2012). However, regarding human-machine communication, people tend to react adversely to robots issuing commands to them, or dictating the terms of their interaction with spoken language (Dautenhahn, 2007; Mutlu, Bartneck, Ham, Evers, & Kanda, 2011; Syrdal, Dautenhahn, Koay, & Walters, 2009). Instead, people are relatively more receptive to non-verbal communication emanating from robots (Dautenhahn, 2007; Komatsu, 2006; Syrdal et al., 2009).

Whatever side of the argument one takes, nonverbal communication has received much more attention from investigators working with humanoid or zoomorphic robots than with investigators employing robots that are not humanoid or zoomorphic. What is suggested by these wide-ranging investigations is that people can easily interpret the meaning of nonverbal utterances. (See Rossini (2012) and Read and Belpaeme (2010) for overviews of this literature.) People who are ill or in pain tend to reduce their level of verbal communication, making more use of nonverbal communication ("Pain in Non-verbal" 2011). The nonverbal communication of American Sign Language is reportedly more effective "than spoken English because of the linearity of spoken language" (Quenqua, 2012). These findings and observations further underscore the need for and desirability of a novel NVC-approach like the NVC platform presented in this study is to human-robot interaction.

Nonverbal communication involving non-humanoid, non-zoomorphic robots has been the focus of few investigations (Komatsu, 2006; Matsumoto, Fujii, Goan, & Okada, 2005; Okada, Sakamoto, & Suzuki, 2000; Yamada & Komatsu, 2007). The closest research ambition is that of Yamada and Komatsu (2007), who developed and evaluated a "minimalist" looking, non-anthropomorphic, LEGO MindStorms robot communicating by audio "beeps." Yamada and Komatsu also recognize the

challenges of the "uncanny valley" for anthropomorphic robots, as well as the rich and expedient characteristics of NVC. They cite Matsumoto's important concept of "minimal design" for robots (Matsumoto et al., 2005), which calls for agents having minimalist (non-anthropomorphized) appearance and communicating to users via minimal (low-information) means, as realized in their interactive robot,
"Muu" (Okada et al., 2000). Inspired by Matsumoto's concept, Yamada and Komatsu developed audio primitives by means of simple beeps of different durations and deflections. They found that sounds with decreasing intonation of shorter durations were perceived as expressing "agreement"; that sounds with increasing intonation regardless of intonation were perceived as communicating "disagreement"; and that flat sounds with longer durations were perceived as communicating "neutrality" or "hesitancy" (Komatsu, 2006). These beeps, mapped to the three robot "utterances," served as intuitive, effective perceptual primitives for the MindStorms robot in communication to users. Frequency (number of beeps), duration (in ms), and pitch (in Hz) over the duration were the measurements used for the NVC vocabulary. The NVC developed for the MindStorms robot was then compared to more natural communication, expressing the same three states, comprised of tail-wags, barks and blinking as presented by AIBO, the consumer-popular robotic dog. Eighteen students evaluated the robot communication by questionnaire (a six-point Likert scale; ANOVA) with respect to accuracy in mapping "utterance" to significance and with respect to the impressions of the robots on the participants. Example questions

included the following:

> Q1: Did you understand the robot's mind?
> Q2: Was the expressed information easily understandable?
> Q3: Did you enjoy the way that the robot expressed its information?
> Q4: Do you think that this robot can be part of our daily life?
> Q5: Do you think that this robot has emotions?
> Q6: Do you think that you can communicate effectively with this robot?

The experimental results suggested that communication by the MindStorms robot was more effective for and better received by participants than that of AIBO. In other words, the robot of "minimalist" appearance/communication outperformed the "familiar pet" robot (Komatsu, 2006). While Yamada and Komatsu's results are striking, quite obviously, they speak for a robot that 1.) exhibits "minimalist" appearance and communication (just three "utterances," and all of these limited to auditory primitives); and that 2.) is not intended to assist people. In Yamada and Komatsu's experiments, as well as in a similar one by Read and Belpaeme (2010), the human-robot communication is merely uni-directional: participants do not communicate with the robots but instead assume the role of the recipients of the robot's utterances. As Cassell points out, "there has been little research" situated "in the wild" and "focused on bi-directional human-robot communication employing models of nonverbal communications as both input and output" (Cassell et al., 1999, p. 521).

1.3 Key components of our human-robot communication loop

In this research, an NVC was developed and tested through an iterative, human-centered design and evaluation process. In addition, as reported in Klingspor, Demirs, and Kaiser (1997) we recognize that designing interfaces for Human-Robot Communication demands that 1.) the interface allow users to intuitively instruct the robot and 2.) feedback "must be provided to the user so that she can immediately understand what's happening on the robot's side" (p. 721). Towards realizing the ambition of an NVC integral to a human-robot communication loop, we employed ART, our well-suited Architectural Robotics hardware/software platform, as well as a novel means of communicating with the robot through a learning algorithm consonant with the approach and motivations presented here. Consequently, the missing piece in our system is the focus of this research: the development of an NVC "spoken" by the robot. To understand our human-robot communication system, we introduce in the following section one aspect that will support the development of the NVC platform.

1.3.1 "Emergent" gesture recognition via learning algorithm

Recognition of human gesture is integral to the development of intuitive human-robot interfaces, particularly those for assistive robotics facilitating rehabilitation and aging in place. With hand and arm gesticulation accounting for some 90% of gestured communication (Mitra & Acharva, 2007) research within the lab has been focused at this scale. The problem of gesture recognition typically involves a common set of issues to be addressed including sensing, data representation, pattern recognition, and machine learning (Yanik et al., 2012). A review of methods in each of these spaces has allowed us to select a set of approaches that overcomes various limitations and affords many desirable outcomes.

Typical sensing paradigms include wearable instruments (Jin et al., 2011; Lamentec & Bajcsy, 2004; Zhou et al., 2009), IR proximity sensors (Cheng, Chen, Razdan, & Buller, 2011; Ryu et al., 2010; Yanik et al., 2011), and, most often, cameras (Chen, Georganas, & Petriu, 2008; Kuno, Murashima, Shimada, & Shirai, 2000; Yun & Peng, 2009). These sensing strategies may suffer from practical difficulties, inadequately sparse data, or user discomfort due to compromised privacy (Beach et al., 2009; Demeris, Hensel, Skubic, & Rantz, 2008). To overcome these limitations, our work utilizes the depth-sensing capability of an RGB-D camera (Microsoft, 2012) that preserves user privacy while providing a sufficiently rich data set.

Data representations of gesture may be broadly categorized as parametric (dependent on the kinematics of the actor) versus holistic (based on sensor data statistics) (BenAbdelkader, Cutler, & Davis, 2004). Motion History Images (MHI) (Bobick, 1999; Karahoca & Nurullahoglu, 2008), histograms of gradients (HOGs) (Dalal, Triggs, Rhone-Alps, & Montbonnot, 2005), and self-similarity matrices (Cutler & Davis, 2002; Junejo, Dexter, Laptev, & Perez, 2008; Yanik et al., 2011) have been used to generate robust discriminants. It has also been shown that observers may recognize motion-by-motion trajectories (accelerations) at the actor's joints (Johansson, 1973; Rao, Yilmaz, & Shah, 2002). Use of the RGB-D camera's depth sensing capability allows us to represent data according to its 3D maxima of acceleration.

Numerous methods have been applied to the task of gesture classification. These commonly include Hidden Markov Models (Moni & Ali, 2009; Wilson & Bobick, 2000; Yamato, Ohya, & Ishii, 1992), neural network approaches (Kleinsmith, 2004; Touzet, 1997; Varkonyi-Koczy & Tusor, 2011), and clustering (Prasad & Nandi, 2009; Schlomer, Poppinga, Henze, & Boll, 2008). Our approach utilizes the Growing Neural Gas (GNG) algorithm (Angelopoulou, Psarrou, Garcia-Rodriguez, & Gupta, 2010; Fritzke, 1995; Stergiopoulou & Papamarkos, 2006) in order to effect clustering of gesture representations; to act as an associative memory for robotic response (Touzet, 1997; Yanik et al., 2012); and to track a moving distribution of input patterns (Holmstrom, 2002).

Machine learning allows the robotic agent to develop an associative mapping between sensory input and kinematic response. This often requires offline training with large data sets or extensive reinforcement learning trials. A primary goal of the research is to allow learning with a novice human "teacher". Thus, the endurance and patience of the user are of primary concern. We employ a form of reinforcement learning (Sutton & Barto, 1998; Watkins & Dayan, 1992) which combines a searchable feature map (Touzet, 1997) with a human-generated reward signal

(Blumberg et al., 2002; Kaplan, Oudeyer, Kubinyi, & Miklosi, 2002; Kuno et al., 2000; Thomaz & Breazeal, 2008) to reduce training time to levels tolerable by a human user.

Collectively, the methodology we employ provides an easily adoptable paradigm for inexperienced users, preserves privacy, and expedites learning in real time. Particularly novel is the robot's capacity to learn to recognize the gesture of a human partner; recognizing that gesture of human partners can vary due to perceived angle of reception, precise form of expression, and (most importantly) the varying capacity of individual users to perform the gesture given the current (and changing) status of their health. This combination of qualities is not only novel but also essential to an adaptive human-machine interface. In like ways, both directions of our proposed human-robot communication loop "take into account that gestures (as signs) are something 'alive', depending on cultural and human aspects, time and context" (Malizia & Bellucci, 2012, p. 37).

While this is not the focus of the research presented, the NVC platform requires this research to work effectively and efficiently. Within the larger lab, research is being conducted on the Growing Neural Gas algorithm and its relationship to ART and the NVC platform. Currently, the gesture interface is hardwired into the NVC platform as a proof of concept thus, allowing the user to interface with ART and the NVC platform more effectively.

1.4 Our team's early prototype of this kind of human-robot communication

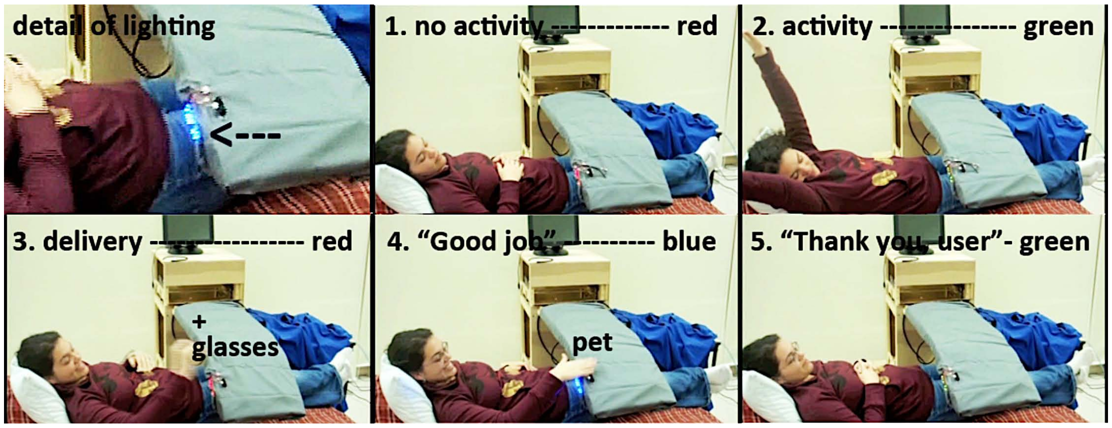

detail of lighting
<---
1. no activity ------------- red
2. activity --------------- green
3. delivery --------------- red
+
glasses
4. "Good job"---------- blue
pet
5. "Thank you, user"- green

Figure 1.3: Stills from our video, showing our early-pilot development of ART and our NVC loop. The assistive robotic table is conversing to Jessica with lights (numbers and durations of blinks; color), and Jessica is reinforcing the table's actions by "petting" it.

An early sketch (in hardware and software) of this human-robot communication system was generated in a graduate-level course. As can be seen in figure 4 of the video clip (http://youtu.be/Iui8cwEyApY) and as referenced in conference proceedings (Merino, Threatt, Walker, & Green, 2012; Threatt et al., 2012), the early working prototype of ART delivers eye glasses to the patient; the patient then retrieves the glasses; lighting conveys the systems status throughout the activity; and the patient reinforces the robot's "good" behavior with a haptic "pet"; which causes the robot to respond with "your welcome" in evocative lighting. While this is a rather crude prototype exhibiting simplified NVC and human gesture recognition (with learning), it is nevertheless a compelling proof-of-concept of the research vision.

1.5 Project Objectives

1. To iteratively design and evaluate a mode of nonverbal communication for assistive robotics.
2. To develop a mode of NVC that is understood by user(s) and robot.
3. To develop a mode of NVC that can be extended and customized over time as user needs change.
4. To develop a means to program the robot for new or modified NVC utterances.
5. To ponder the larger question of how much human intelligence is required of an assistive robot?

1.6 Hypothesis

H1: An assistive robot that conveys nonverbal communication can be readily understood by users, with the long term goal of those who are medically at-risk, such as post-stroke patients.

H2: Additionally, an assistive robot communicating in NVC does not need to be humanoid or otherwise explicitly life-like in appearance to be readily understood by users.

H3: Finally, our entirely nonverbal human-robot communication will be perceived as a desirable, low-invasive, and expedient communication mode for HRI, particularly in intimate human-robot interactions.

1.7 Research Questions

I. In an increasingly digital society, how might a non-verbal communication (NVC) platform be embedded in the Assistive Robotic Table to provide clinicians opportunities to best understand and interact with a patient’s recovery?

II. Will clinicians prefer an Assistive Robotic Table with an NVC platform over current stroke patient therapeutic practices?

III. What are the requirements for an NVC platform used in the rehabilitation of post-stroke patients?

IV. How might an NVC platform continue to support life-long rehabilitation?

CHAPTER TWO

CASE STUDY

2.1 Introduction

Medical facilities and healthcare personnel are overextended and costly. With the graying of the population, there is a smaller segment of the population to both care for, and pay for the well-being of, older and clinical populations (Houser, Fox-Grage, & Gibson, 2006). Data suggest that most people want service robots to assume an assistive role that compensates for their reduced capacities, more so than having robots physically resemble humans or “possess” virtues best ascribed to humans such as creative thinking, judgment, or friendship (Fong, Nourbakhsh, & Dautenhahn, 2003; Wada & Shibata, 2007). More broadly, previous robotics research for health and eldercare applications have tended to focus on specialized devices aimed at dedicated tasks such as rehabilitation robotics (Leifer, 1981; Tapus, Mataric, & Scassellati, 2007), robot-assisted surgery (Rosen & Hannaford, 2006; "Special Issue on Robots in Surgery," 1995), prosthetics (Dellon & Matsuoka, 2007; Leifer, 1981), or to replace humans engaged in healthcare-related activity (Dario, Guglielmelli, Allotta, & Carrozza, 1996; Forlizzi, 2005; Forlizzi, DiSalvo, & Gemperle, 2004; Pineau, Montemerlo, Pollack, Roy, & Thrun, 2002; Roy et al., 2000)

This chapter focuses on the development of the Assistive Robotic Table (ART), a new over-the-bed-table for aging in place. We envision ART as integral to the domestic life of its user(s), even as they transition from home to clinic and, hopefully, back home again. Users that have ART as part of their domestic landscape might rehabilitate faster while under the care of healthcare professionals in a rehabilitation hospital that also makes ART part of its clinical setting; ART is meant to be highly attractive and functional both at home and in clinic.

The design of ART was a challenge for our larger design team. To understand the people and systems with which ART interacts requires a “creative” approach to research, involving both creative design and engineering innovation concentrating “on functionality, aesthetics, and manufacturability simultaneously” ("Review of Hartfield, B. and Winograd, T. Bringing Design to Software," 1996, p. 165). Our larger lab has elaborated these design and engineering principles (Green, 2008) and, more popularly, these principles are employed by the designers of IDEO (2003).

Physically, ART is an over-the-bed table (OBT) universally found in hospital patient rooms with a novel, yet optional, plug-in upper extremity rehabilitation device used in therapy. ART comprises a smart OBT that makes the human-object interface, specifically when addressing positioning of the object around the patient, more usable and a novel therapy surface that helps patients perform upper extremity therapy exercises of the wrist and hand to aid in their ability to perform activities of daily living. The components of ART recognize and communicate with each other in interaction with human users. ART aims to augment the rehabilitation environment by improving patient well-being, rehabilitation, and staff productivity.

2.2 The Assistive Robotic Table (ART)

In the investigation described in the following chapters, we designed and evaluated several alternative modes of nonverbal, robot communication towards establishing an effective non-verbal communication (NVC) loop (as conceptualized in Figure 1.2*)* – one that is efficient, expedient, user-extendable, and user-customizable. For these experiments, we embedded our developing NVC platform in a real-world context, the Roger C. Peace Rehabilitation Hospital (RCP) of the Greenville Health System, where medical clinicians (and in the future their post-stroke patients) determined its effectiveness.

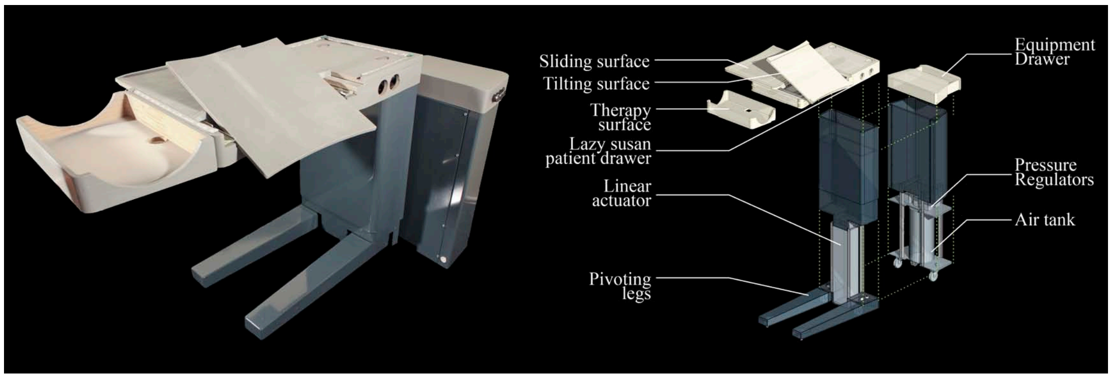


Figure 2.1: Presented are the features that comprise the Assistive Robotic Table (ART). The highlighted features are the result of a study defining ART.

As healthcare is becoming more digital, technological and increasingly costly, healthcare environments have yet to become embedded with digital technologies to support the most productive (physical) interaction across patients, clinicians, and the physical artifacts that surround and envelop them. This shortcoming motivated our larger team to envision home+, a suite of networked, distributed "robotic furnishings" integrated into existing domestic environments (for aging in place) and healthcare facilities (for clinical care). The over-the-bed table (OBT) that exists in most hospital rooms is cumbersome to control and maneuver, cheaply made, and often broken.

Our response to the over-the-bed table, the table we employ for this investigation is a novel one developed in our lab: the Assistive Robotic Table (ART) presented in Figure 2.1. The result of participatory design and evaluation with clinicians in a rehabilitation setting, and supported by a National Science Foundation Smart Health & Wellbeing award (IIS SHB IIS-1116075), ART is comprised of a cantilevered OBT (Manganelli, et al., 2013a; Manganelli, et al., 2013b). The robotic table has two degrees of freedom: it raises and lowers from its base and has a tilting work surface. At the extreme tip of the table surface is a continuum-robotic surface supporting post-stroke patient therapy, actuated by twelve pneumatic muscles (with, theoretically, infinite degrees of freedom) (Merino et al., 2012).

The surface was constructed of a spacer fabric made from polyester fibers on a triple bar warp knitting machine. The fibers of the material were woven to induce maximum curvature while providing stability for the arm and decreasing lateral compression. The material was also designed to be thick enough to allow muscles to be attached to both sides of the surface via elastic thread through the surface material and around the muscle down the entire length of the muscle and tied down at the two opposing end of each muscle. Figure 2.2 shows the final therapeutic surface and Figure 2.3 shows the intended interaction in the rehabilitation setting[1].

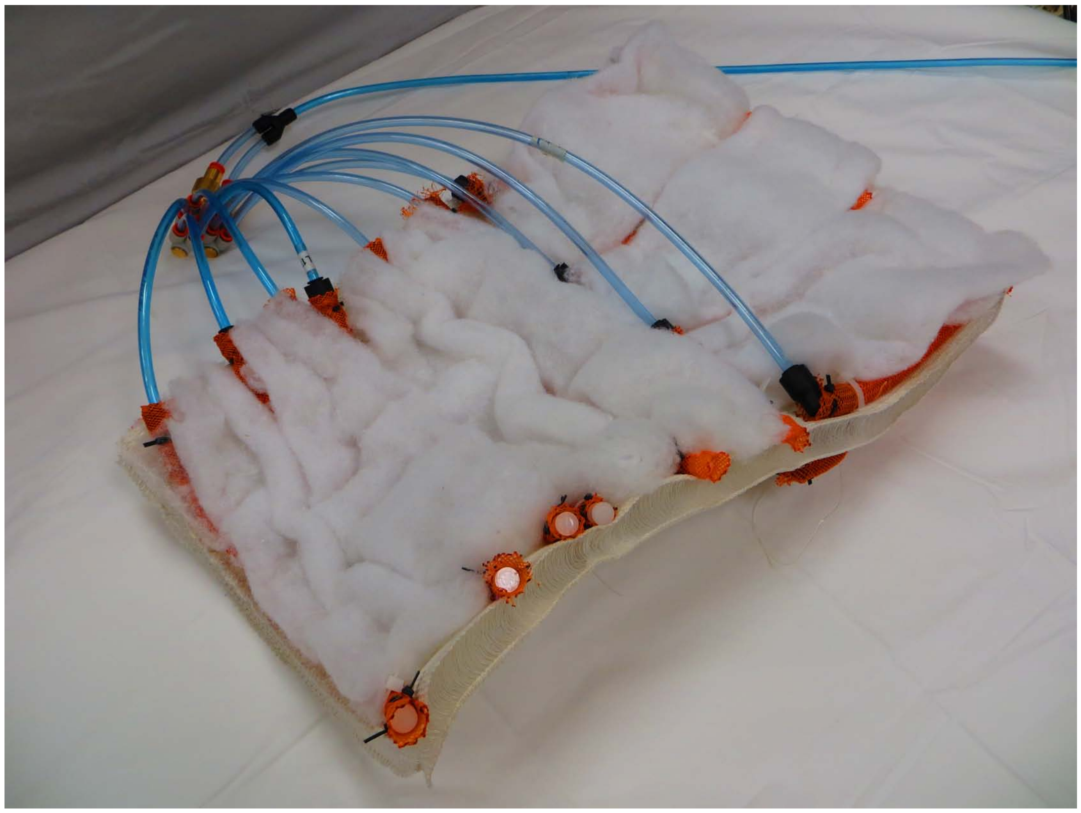


Figure 2.2: Final therapy surface prototype with 12 pneumatic muscles, spacer fabric, and cotton batting. The therapy surface was developed through a parallel iterative design and evaluation process along with ART.

The aim for this research was to develop an actively assistive table that when evaluated by clinicians and patients was an improvement (by subjective scales) over the current nightstands and OBTs used in the rehabilitation hospital setting. This current research follows a line of research conducted within our lab to understand the

[1] The following are videos from each phase of the iterative design process for the therapeutic surface (links are available as of 7/26/2013). Phase 1 http://youtu.be/BYsO6ZOOIoA, Phase 2 http://youtu.be/CXbs3f40y-k, Phase 4 http://youtu.be/-275vsGakI4, Phase 5 http://youtu.be/A2PldUhq2RE, Final Prototype http://youtu.be/ycvVfJapgJc

contents of a nightstand (Brooks et al., 2011a; Smolentzov, 2010), understand clinicians preferences for current OBTs (Manganelli, et al. 2012), design and evaluate a new nightstand (Brooks et al., 2012), understand how healthcare practitioners use OBTs (Manganelli, et al., 2013a), and to validate healthcare practitioner's requirements for an actively assistive robotic table (Manganelli, et al., 2013b). In order to accomplish this aim, the base, up and down mechanism, and operating controls were replaced. Features such as the tilting surface, lazy Susan patient drawer (later removed to implement the gestural interface), and clinician's work surface were added to the table to enhance the user's experience by (subjectively) increasing the user's effectiveness, efficiency, and satisfaction with ART over the current OBTs. Also, the continuum robotic surface (and later the mechanical column to control it) was designed to increase the effectiveness and efficiency of therapy sessions and was an integral piece of the engineering research aims. This research shows that ART is an improvement over the current OBT and, therefore, is an adequate environment to interact with the NVC platform. Furthermore, ART provides the NVC platform an entrance into and the first step for integration with our lab's vision for the larger patient hospital room ecosystem. The subsequent section describes one patient type -- post-stroke patients, who are prevalent in the southeastern United States and present both cognitive and physical deficits. The aim for ART and the NVC platform was to address both physical and cognitive patient deficits, as those living longer (in particular) may find themselves moving between home and clinic as they mature.

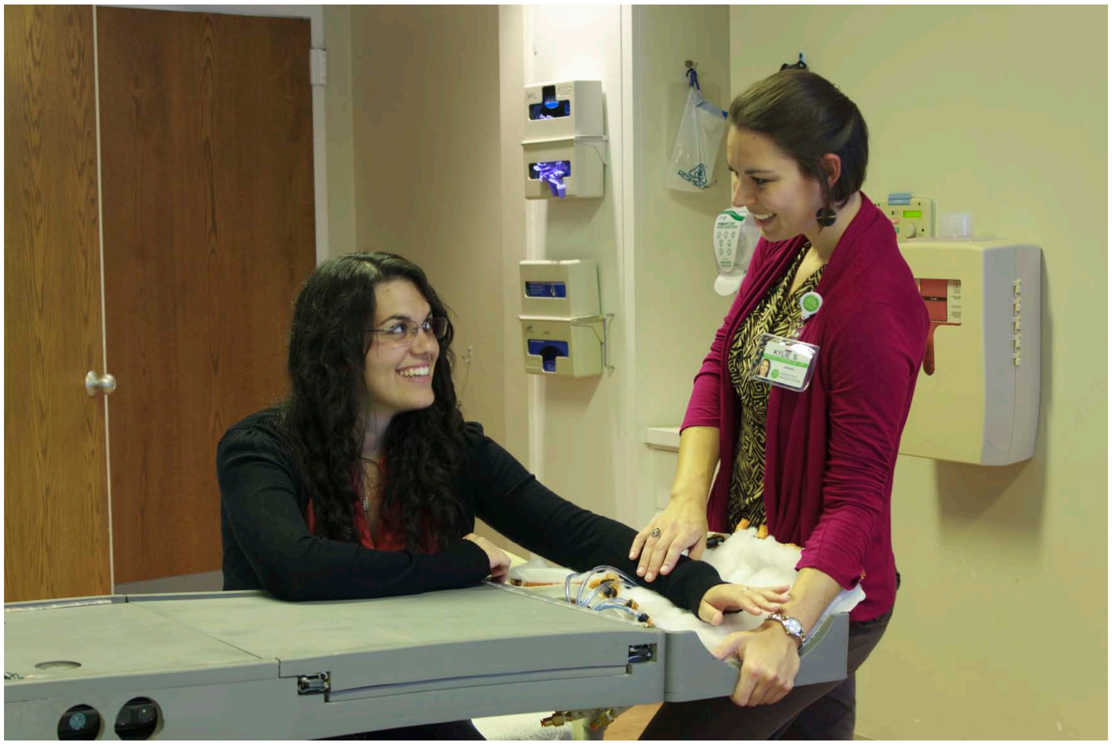


Figure 2.3: ART, the therapy surface, and rehabilitation: This figure shows the intended interaction with ART, the therapy surface, patient, and clinician in the rehabilitation setting.

2.3 Post-stroke patients

In the southeast United States, statistics show that the percentage of people affected by stroke is increasingly higher than in other United States regions, attributed to high blood pressure, smoking, and diet (National Stroke Association, 2012c). "In the United States, stroke is the fourth leading cause of death, killing over 133,000 people each year, and a leading cause of serious, long-term adult disability... [A]pproximately 795,000 strokes will occur this year" (National Stroke Association, 2012c). Rehabilitation within minutes, days, and weeks after a patient suffers a stroke is most important often requiring intense and direct interaction between the patient and therapist. Stroke survivors often suffer from cognitive deficits (short term memory loss, dementia, or aphasia) and/or physical deficits (visual field cuts or hemiparesis) often requiring lifelong rehabilitation; and due to the high cost of healthcare the patient performs the exercises at home alone (National Stroke Association, 2012a, 2012c).

Cognitively, aphasic patients may lose their "ability to communicate orally, through signs, or in writing" (Teasell, McClure, Katherine, Salter, & Murie-Fernandez, 2013, p. 21). Physically, hemiplegic patients may be inattentive to their upper extremities as they have "weakness or the inability to move one side of the body" (National Stroke Association 2012b). Unlike gait training that is a gross motor movement, the affected limb often requires the retraining of fine motor skills needed for activities of daily living. Additionally, a visual field cut may reduce a patient's peripheral vision. Rehabilitation methods include exterior sensory stimulation or other visual or auditory feedback strategies with increased intensity and difficulty in an attempt to increase the patient's awareness for the vision and limb neglect (National Institute of Health 2011).

We updated the OBT to provide a recovering stroke patient a companionable and therapeutic aid in those early rehabilitation hours. As effective post-stroke rehabilitation must occur in the minutes, hours, and days following the stroke and must continue during the weeks and years that follow, stroke patients with these deficits require technologies that are adaptable to the patient needs, increasing and decreasing assistance as required. As elaborated later in this thesis, the variety of personas and the subjective scales used to evaluate the features of ART and the addition of the therapy surface helped show ART's versatility with post-stroke patients with various abilities. The following section here is a discussion of the interactions of ART and the post-stroke patient facilitated by the NVC platform.

2.4 ART, post-stroke patients, and the non-verbal communication platform

> First, people would find working with [assistive robots] more enjoyable and would thus, feel more competent. Second, communicating with them would not require any additional training since humans are already experts in social interaction. Third, if the robot could engage in various forms of social learning (imitation, emulation, tutelage, etc.), it would be easier for the user to teach new tasks. Ideally, the user could teach the robot just as one would teach another person (Breazeal, 2004, p. 182).

Following up a claim by Reeves and Nass (1996) that humans are experts in social interaction, Breazeal (2004) proposed the paradigm of robot-as-social partner. Breazeal envisioned a world where sociable robotics would be, similar to Weiser's (1991) thoughts about computation, ubiquitous in society. Such robots would need to be designed so that anyone could easily interact with them. Therefore, for the designed platform to work effectively the user (clinicians and/or patients), ART, and the NVC must form a symbiotic relationship - each affecting the other.

ART provides the user with an easy-to-use and more enjoyable actively assistive table to support activities such as reading, eating, storing items, and performing therapy. ART provides an environment to develop the NVC platform. As a direct result, the NVC platform provides ART with 1.) a bi-directional communication interface, 2.) a way to position ART, and 3.) a way to accomplish tasks. However, without a user enforcing the interaction, the NVC and ART have no meaning; therefore, the NVC provides the user: 1.) a bi-directional communication interface to ensure the patient understands how the system can and is being used; 2.) a way to position ART; 3.) an ambient alert notification system; 4.) a possible therapy tool for post-stroke patients; and 5.) in the future a notification system to ensure the patient is performing therapy exercises adequately. As the relationship between ART, the NVC platform, and the user continues to grow, it is hoped that the communication platform will change and adapt to user needs.

Similar to previous research in our larger lab (see www.CU-iMSE.org), this research is a collaboration between the fields of architecture, engineering, human factors psychology, and rehabilitation therapy. The NVC platform is envisioned to further the field of architecture by providing a guide for creating conversant, iterative architecture; to advance engineering by extending the works of Mataric for sociable assistive robotics (Feil-Safer & Mataric, 2005; Tapus et al., 2007; Wade, Parnandi, & Mataric, 2011); to expand human factors psychology by developing appropriate communication models for robots, stroke patients, and clinicians and by designing an intuitive interface; and to show health care provider preferences for implementation of the NVC platform. The following section is a proposed scenario for interacting with the NVC platform.

2.5 Scenario

The following scenario envisions how the NVC for assistive-robotics is expected to operate over a short period at a rehabilitation hospital with a patient, her sister, and a clinician. This scenario suggests how the NVC promises to advance the wellbeing of a critical target audience.

Joanne was living a healthy life in her own home, which contained an assistive architectural robotic table amongst its furnishings. Joanne had quickly welcomed ART's functionality and aesthetics. It fit nicely in her home, and it gave her confidence that ART provided more supportive functions than a conventional table, whenever she needed them. Unexpectedly one day, Joanne suffered a stroke that primarily affected her left side. Presently, she is in her private room, reclining in bed, listening to a podcast on her iPod. Next to Joanne is ART. Joanne's sister, Amy, has arrived to visit Joanne and accompany her for the afternoon therapy session. As Amy enters Joanne's room, she waves her hands to signal her entry. Joanne interrupts her podcast and welcomes her sister with a few kinds words and by waving her right hand. Joanne's left arm lays limp in the bed, and she tells Amy that she is having issues with her left leg. Joanne reports her frustration with her physical condition and her short-term memory lapses. Amy asks about Joanne's barely eaten food, and Joanne tells her she is still having trouble swallowing, not to mention her difficulty seeing the left side of the tray without twisting herself about.

Joanne asks Amy to borrow her tablet computer to check her email. To accommodate the computer, Joanne would like ART to rise and position its flip tray for her. Joanne still feels a little unsteady holding things; fortunately, ART can provide the needed support for this activity. While Joanne and Amy continue their exchange, the following sub-linguistic dialogue occurs between Joanne and ART:

Joanne: [Gestures ART to kindly rise and tilt, as if to say, “ART, please rise and tilt for me.”]

ART: [Displays **two quick light flashes** and a **beep beep** sound , as if to say, “Yes, I am pleased to do that for you.”]

All the while, Joanne and Amy are chatting, catching up on recent news in their lives and the world. A few moments later:

Joanne: [Gestures for ART to rise, but ART does not comprehend at first].

ART: [Displays **blinking lights** and a sound that, if written, might be **ant ant**, as if to say, “Hmmm, I don’t know what you are asking of me. I am puzzled.”]

Joanne: [Makes the gesture once more in a way that ART comprehends, learns from, and responds with the correct behavior.]

To reinforce ART’s actions,

Joanne: [Runs her hand along ART’s sensors at the perimeter of the table, in what appears to be a “pet" to convey, “Thank you.”]

ART: [Displays **gradient on/off light pattern** and a **purrr** sound, as if to say, “I understand that I performed the task correctly!”]

In this scenario, ART communicates with Joanne at a nonverbal level so that ART neither competes for Joanne’s attention nor distracts she and her sister from the intimate human conversation and course of therapy. The NVC is a human-robot platform that renders the robot useful but neither intrusive nor invasive to human-human relations and the routines of individuals. Should Joanne find an utterance irritating, she gestures so, and ART presents her alternatives until she selects, by gesture or pet, an acceptable one. Should Joanne or Amy required the addition of a new utterance to the NVC platform, this can be added by way of a user-friendly tablet interface.

In the qualitative design study reported, we designed and evaluated a three phase pilot study for one non-verbal communication (NVC) for our Assistive Robotic Table (ART), an architectural-robotic artifact supporting post-stroke patients. Participants in our iterative design and evaluation process are medical staff (i.e., physicians, nurses, occupational, physical and speech therapists) at the Roger C. Peace Rehabilitation Hospital of the Greenville Health System (RCP). Because the design and evaluation of language like our NVC is complex, medical staff regularly treating post-stroke patients, rather than the patients themselves, are more apt participants in the early phases of developing that language. That is medical staff are experts in the condition of post-stroke patients, and patients may be overwhelmed with this novel human-robotic interface. In the next chapter, the iterative design and evaluation cycles of our investigation involving such medical staff are elaborated.

CHAPTER THREE

METHOD AND PROCEDURES

3.1 Introduction

This study sought to understand clinician preferences for a non-verbal communication (NVC) platform composed of lights and sounds for a robot, envisioned for intimate human-robot interaction. We developed an exploratory participatory design and evaluation process of an NVC in a regional rehabilitation hospital with a participant pool including doctors, occupational, physical, and speech therapists, as well as environmental service technicians. Each phase of this study sought to understand a variety of questions for the NVC platform, as well as to use a convergent design method. The following is the study design including the procedure for the three phase study.

3.2 Participants

Volunteers for this study consisted of research team members and a convenience sample of clinicians at Roger C. Peace Rehabilitation Hospital. Eight members of the research team participated in the pre-study activities. Thirteen health care subject-matter experts (stroke patient rehabilitation and general rehabilitation experts), including doctors, occupational, physical, speech therapists, and environmental service technicians – participated in phase I of the research study. Twelve subject-matter experts – all clinicians, including doctors, occupational, physical, and speech therapists – participated in phase II of the research study. Ten subject-matter experts – all clinicians, including doctors and occupational and physical therapists – participated in phase III of the research study. Two participants dropped out of this phase of the study for personal reasons; thus, 10 participants completed phase III. In the interest of protecting the privacy of this small, exploratory sample population and based upon the conditions of the approval for this study design by the hospital's institutional review board, demographic data are not presented here.

The 12 clinicians who participated during phase II also participated in phase I. The 10 clinicians who participated during phase III participated in all three phases. This is a result of three factors. The first is that the small convenience sample population used in this study did not allow for multiple rounds of participant recruitment from within the hospital. The second factor was the use of convergent methods to design the NVC platform - requiring participant to be familiar with the platform throughout the process. Finally, due to the novelty of the NVC platform there was desire to have a repeated measure - technology acceptance for the NVC platform. The limitation of the convenience sample provided an opportunity to develop the subsequent methods to define the trajectory of the study.

3.3 Study Design

3.3.1 Independent Variables

Independent variables for phase I of the study included two sounds for each of the 20 NVC actions for sound and two lighting prototype designs (e.g., individual LEDs or an LED screen). See Table 3.1 for a complete breakdown of the independent variables for phase I.

Phase I			
Sound			Light
Bend in	Drag	Reprimand	Individual LEDs
Bend out	Emergency	Stop	LED Screen
Can't do	Go	Swipe	
Come	I'm thinking	Tilt Back	
Confirm Request	Pet	Tilt forward	
Do not understand request	Select	Up	
Down	Something in the way		

Table 3.1: Presented are the independent variables tested during Phase I. Each sound had two sounds associated with it, were tested audibly, and the clinicians selected their preference between the two. The clinicians chose between the lighting options and used the chosen option to place the options on ART.

Independent variables for phase II of the study included two sounds for each of the 15 NVC actions for sound and two lighting sequences for each of the 17 NVC actions for lighting sequences. Additionally, there were seven buttons and eight gestures that initiated the sound communication, as well as eight buttons and nine gestures that initiated the light communication. The *Swipe, Drag*, and *Select* actions were not tested because a tablet interface was not a part of the ART prototype. See Table 3.2 for a complete breakdown of the independent variables for phase II.

Phase II			
Sound		Light	
Button	Gesture	Button	Gesture
Up	Come	Up	Come
Down	Go	Down	Go
Tilt forward	Stop	Tilt forward	Stop
Tilt Back	Confirm Request	Tilt Back	Confirm Request
Emergency	Do not understand request	Emergency	Do not understand request
Pet	Can't do	Pet	Can't do
Reprimand	I'm thinking	Reprimand	I'm thinking
	Something in the way	Bend Out	Something in the way
		Bend In	

Table 3.2: Presented are the independent variables tested during Phase II. Each sound and lighting sequence tested had two options, were either button or gesture actuated, and the clinicians chose which sound or lighting sequence best represented each NVC action.

Independent variables for phase III included five NVC actions that were assigned a button command and eight NVC actions assigned a gesture command. Three NVC actions were assigned a sound, five NVC actions were assigned a lighting sequence, and five NVC actions were assigned a sound, lighting sequence, or a sound and lighting sequence combination. See Table 3.3 for a complete breakdown of the independent variables for phase III as well as this video http://youtu.be/kR2JMkhJBJw.

Phase III		
Sound	**Light**	**Sound And Light**
Button	**Button**	**Button**
Down	Reprimand	Tilt Forward
	Bend In	Tilt Back
Gesture	**Gesture**	**Gesture**
Stop	Emergency	Go
Can't Do	I'm thinking	Do not understand request
	Come	Something in the way

Table 3.3: Presented are the independent variables tested during Phase III. Without prior training, the clinicians listened to or viewed the NVC action, chose what action they thought it was, and, if necessary, stated whether the sound, lighting sequence, or both together best demonstrated the presented action.

3.3.2 Dependent Variables

For phase I and II, dependent variables included the selection of the ART communication action choice, either *A, B, or Neither,* and comments given by the clinicians for open-ended response questions. Dependent variables for phase III included the ART communication action, the sound, lighting sequence or sound and light sequence combination that best described the action, the System Usability Scale score, and comments given by the clinicians for open-ended response questions about the ART communication action.

3.3.3 Setting

The study occurred in the home+ lab in the basement of Roger C. Peace Rehabilitation Hospital (RCP). The home lab is an extension of the research and therapy gyms at RCP and was designed as a studio apartment. The home lab includes a dining table with 6 chairs, a nightstand, a dresser, a pantry, counter space, and a bed. See figure 3.1.

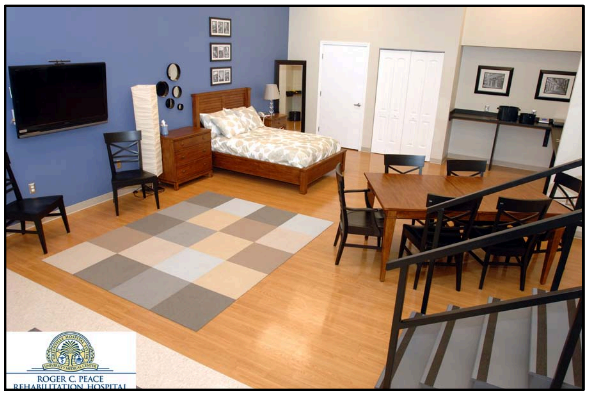


Figure 3.1: Home+ lab at Roger C. Peace Rehabilitation Hospital of the Greenville Health System.

3.4 Procedures

The methods of this study are presented in three sections: ***Phase I, Phase II, Phase III***. Phase I was conducted in November 2012, Phase II was conducted in March 2013, and Phase III was conducted in April 2013. The research timeline was affected by two factors. The first factor was a limitation set by our partnership with RCP that the research team could only interact with an individual clinician one time per month for up to one hour. The time constraint placed on the research team provided the second factor, a repeated measure study design, to be implemented (Cohen, Cohen, West, & Aiken, 2003). In this study design, the repeated measure was technology acceptance by the clinicians for the designed and evaluated NVC platform. This allowed the research team to see how clinician feelings changed throughout the research process. Described below are the procedures for each phase of the research study.

The procedure prior to the start of an experimental session was the same during each phase: the research moderator welcomed the clinicians, handed out the research study information sheet describing the purpose of the study, asked the participants to review it, and answered any questions the participants had about the study. Data for this descriptive study were collected through structured interviews and recorded on a personal computer. Each phase of the study was conducted as described in the following sections. At the end of each session, the moderator answered any questions the participants had about the session, reminded the participants about future sessions, asked the participants not to speak about the session with their colleagues who might participate in future sessions, and thanked the participants for their participation. Approval from the appropriate institutional review boards was obtained prior to data collection.

3.4.1 Phase I

To develop the NVC, eight members of the lab research team - two human factors specialists, four research coordinators and two PhD students, who were not subject-matter experts, participated in pre-study, brainstorming activities to provide a list of 20 actions by which the NVC could be matched to ART (e.g., *up, down, forward, back, correct action, something is in the way, I don't understand, etc.*). The lab members then met in small groups to generate potential sound and lighting sequences to describe the actions. Regarding which sounds and lights were best matched to a given action, there was consistency, both within the groups (e.g., by a group discussion) and between the groups (e.g., after all the focus groups were conducted, each team member completed a survey about his or her sound and lighting contribution). This information served as the beginning for the research study sessions. Each focus group session was conducted in less than 60 minutes. Participants completed the survey on their own time.

Focus groups were conducted over three days at RCP. The number of clinicians for each focus group was four, five, and four. The number of participants was determined by each clinician's schedule and provided the research team with an understanding of clinician requirements and to achieve a consensus for the developing NVC platform. (Rogers, Sharp, & Preece, 2011) The clinicians were told the purpose of this phase of the study was to evaluate lights and sounds, two features to be added to ART, as each related to patient-clinician communication with an assistive robot. Each session had one research moderator and a note-taker for every two clinicians. The clinicians sat across from the research moderator at a long table with the note-taker beside the clinician. Audio speakers to play the sounds were placed on the table in front of the clinicians equidistant from each other and the clinicians.

Two feedback methodologies were used: open-ended response and a forced-choice methodology. Open-ended questions were used to determine clinician preferences for NVC in healthcare environments. The forced choice methodology required the clinicians to verbally select their preference from a choice of *A, B, or Neither* after each sound played (two sounds for each of the 20 ART actions). Similarly, the clinicians chose between two lighting prototype designs (individual LEDs or LED screen) that were presented. The clinicians were told that the lighting would display information regarding the 20 ART actions. Finally, on a sheet of paper showing three architectural-drawn views of ART, each clinician marked where he or she thought the light communication displays should be located and verbally answered how he or she would customize the display. Each focus group took less than 60 minutes.

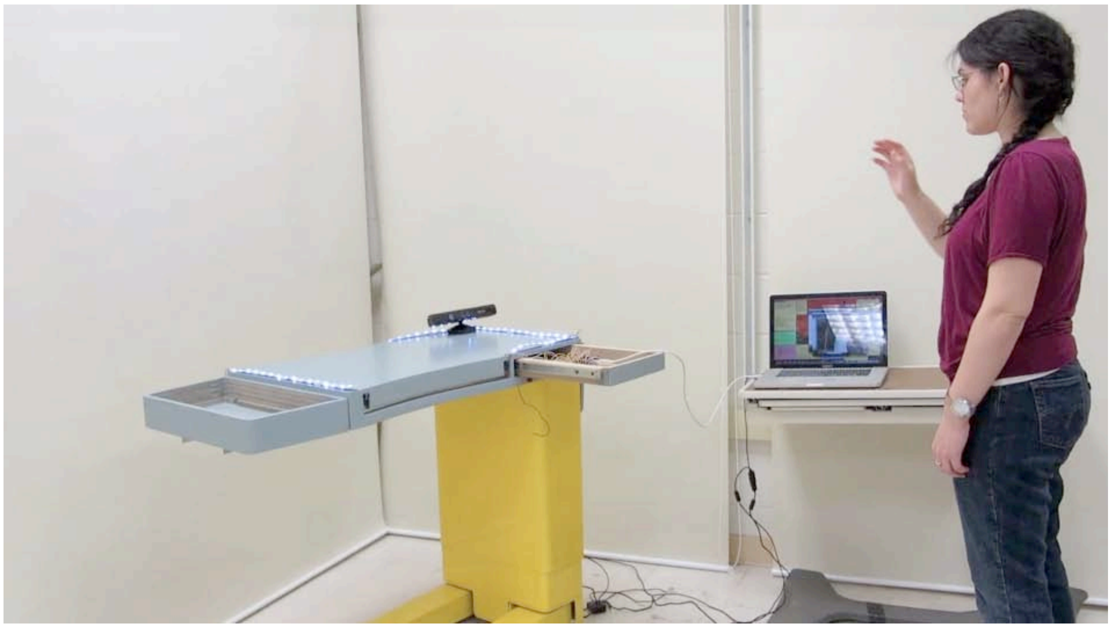


Figure 3.2: Presented is the NVC platform evaluation set-up for Phase II and III. Push buttons and a Microsoft Kinect® were used to actuate each NVC action and sounds were played through an adjacent computer.

3.4.2 Phase II

Clinicians at RCP were interviewed in focus groups of one to two clinicians over six days. The clinicians were told the purpose of the study was to evaluate lights and sounds, two features to be added to ART, as each related to patient-clinician communication with an assistive robot. Each session had two research moderators and a note-taker. The clinicians sat across from the research moderator and ART. LED strips and a Microsoft Kinect® (depth camera) were placed on top of ART and a computer, placed next to ART, was used to play the sounds.

Two feedback methodologies were used: a forced-choice methodology and open-ended response. The forced choice methodology required the clinicians to verbally select their preference from a choice of *A*, *B*, or *Neither* after each sound played (two sounds for each of the 15 ART actions). The *Bend Out* and *Bend In* actions were not tested because during phase I the clinicians stated that those actions did not need a sound. Similarly, the clinicians verbally selected their preference from a choice of *A*, *B*, or *Neither* between two lighting sequences for each of the 17 ART actions. Open-ended questions were used to determine clinician preferences for and possible improvements to the proposed NVC in healthcare environments. Each focus group took less than 60 minutes. Due to time constraints, four participants were unable to answer all questions after evaluating the lighting sequences.

3.4.3 Phase III

Clinicians at RCP were interviewed in focus groups of one to two clinicians over seven days. The clinicians were told the purpose of the study was to evaluate lights and sounds, two features to be added to ART, as each related to patient-clinician communication with an assistive robot. Each session had one research moderator and one note-taker. The clinicians sat across from the research moderator and ART. LED strips and a Kinect were placed on top of ART and a computer, placed next to ART, was used to play the sounds.

Two feedback methodologies were used: a forced-choice methodology and open-ended response. The clinicians were given a printed sheet of possible ART communication actions. Clinicians watched a demonstration of an ART communication action, and either a sound, lighting sequence, or a sound and lighting sequence combination was heard or viewed by the clinician. After watching the demonstration, the clinicians stated what communication action was demonstrated and, if appropriate, which sound, lighting sequence, or sound and lighting sequence best demonstrated the communication action. This procedure was repeated for the 15 ART communication actions. All communication methods were tested without the associated movement because some actions (e.g., *Go*, *Come*, *Stop*) did not have an associated movement developed in this prototype and there was a need to avoid biased results for the actions that worked in the prototype. Four ART communication actions (e.g., *Up, Pet, Confirm Request, Bend Out)* were not tested because during phase II the clinicians did not declare a majority preference for either a sound or lighting sequence. After completing the confirmation of the ART communication actions, clinicians answered a System Usability Scale survey. Finally, open-ended questions were used to determine clinician reasoning behind the answers given during Phase I. Two communication actions, *Do not understand request* and *I'm thinking*, only had nine responses because the research moderator inadvertently gave the answer to the communication action to the participants. Each focus group took less than 60 minutes.

3.5 Statistical Analysis

A frequency analysis was used to determine the sound or lighting sequence selection for each of the ART communication actions. Selections were chosen if they had a greater than 66 % preferred majority because of the small sample size, to ensure a "strong opinion" from the clinicians was achieved, and this selection criteria, developed by previous research in our lab (Brooks et al., 2012), showed the strong opinion criteria was appropriate. An exception was included for five lighting actions that had 58 % preferred majority because there was a 33 % selection of one of the other choices, and the research team wanted to evaluate as many complete communication actions as possible. The System Usability Scale survey was scored by the method provided by Brooke (1996), and a trimmed mean and standard deviation was calculated. A content analysis, developed by frequency analysis, was used on the open-ended response questions. It is the research team's experience that these comments provide insight into the ratings given, and using these statistical analyses help to understand the results presented in the subsequent chapter.

CHAPTER FOUR

RESULTS

4.1 Results

4.1.1 Phase I Results

Given the small sample size, descriptive statistics were assessed (i.e., no statistical validity could be determined). Scassellati, Admoni, and Mataric (2012) states that this is standard in robotics research. "User testing in robotics often focuses on careful qualitative and quantitative observations of small numbers of study participants; it is not uncommon to perform a second-by-second analysis of behavior for only three or four participants in a given study" (Scassellati et al., 2012, p. 277).

Table 4.1 shows the percentage preferences of the clinicians for the 20 sounds tested. More than two-thirds (66%) of the clinicians had to select the sound for it to be evaluated as a preferred sound. Interestingly, a specific *Can't Do* sound was chosen by all clinicians in the study. The clinicians maintained a majority preference for the *Reprimand* (92%), *Something in the way* (84%), and *Confirm request* (76%) sounds.

Of the seven sounds that had a preferred choice, three sounds (*Go, Bend Out, Bend In*) had a preference for *Neithe*r sound. The *Bend Out* and *Bend In* sounds relate to an added therapy surface feature (see Figure 2.2) that will be used by clinicians to help expedite stroke patient recovery. Overall, the clinicians did not feel that a sound was required for these movements because clinicians would be interacting with the patient during the therapy sessions. Because the *Go* sound is an important feature designed for the mobility of ART, it will be retested and evaluated in phase II with an interactive prototype to ensure that the sound is not required. The remaining sounds that did not have majority preference will also be retested in phase II with an interactive prototype.

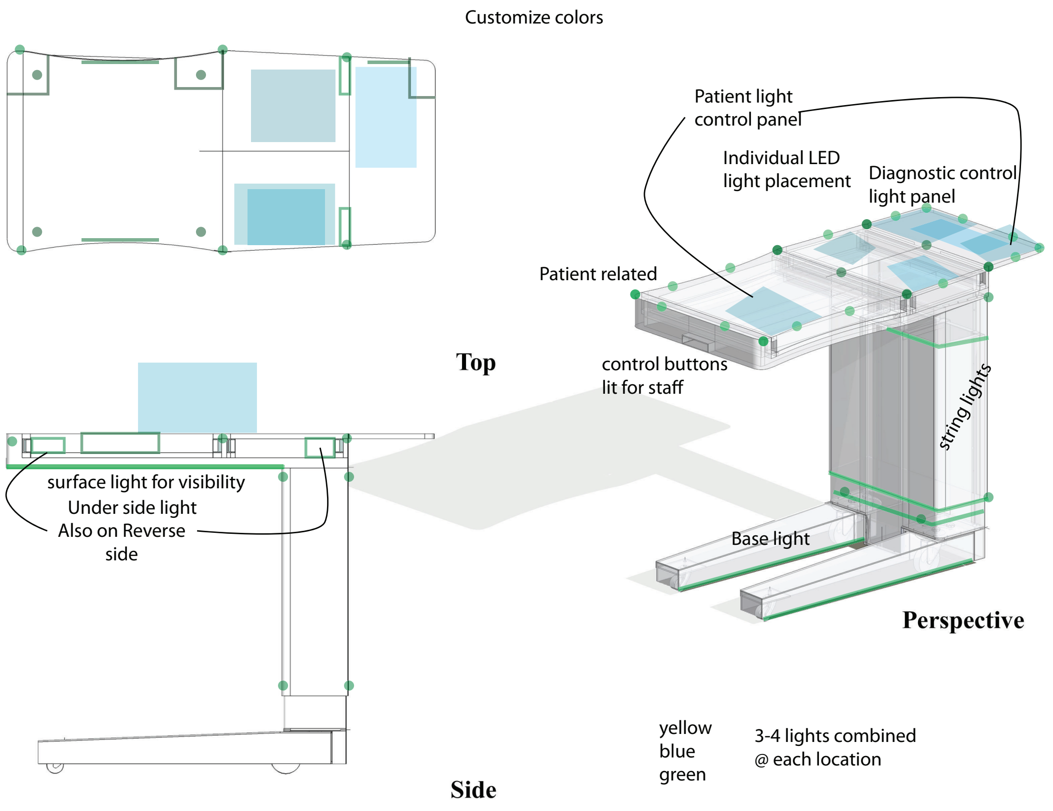
> 66% agreement	< 66% agreement
Can't Do (A)	Down
Reprimand (A)	Stop
Something in the way (B)	Pet
Confirm Request (B)	*Swipe*
Go (N)	Come
Bend out (N)	Emergency
Bend in (N)	I'm thinking
	Select
	Tilt Forward
	Tilt Back
	Do not understand request
	Up
	Drag

Table 4.1: The sound preferences of clinicians during phase I of the study. All sounds were retested during phase II except *Bend out* and *Bend in* because clinicians felt that these two actions would distract from therapy sessions.

Figure 4.1 shows the clinicians' location preferences for selected lighting display prototypes as a "heat map". Four participants chose an LED screen; six participants chose the individual LEDs; three participants chose both displays. The green color (lines and dots) represents the individual LEDs, and the blue color (larger rectangles) represents an LED screen. The color shade variations describe the number of participants who placed the lighting type in the same location. From this data, one can see a trend for the lighting displays. The individual LEDs were drawn on the edges of ART, while the LED screens were drawn primarily on ART's tabletop surface. Clinicians' preferences for customization of the lights included the brightness of the LEDs and the colors displayed - primarily red, green, and yellow. However, one participant noted that red and green should not be used due to patients who are colorblind.

At the beginning of each study session, clinicians were asked, "If ART had the ability to communicate, would the clinicians communicate with ART?" The clinicians unanimously agreed that they would like to communicate with ART. Clinicians then answered 12 open-ended questions regarding what types of information are appropriate in the healthcare setting, how the information should be communicated, and the interaction with stroke patients. Finally, clinicians were asked again if they would communicate with ART. Again, 100 % of the clinicians said that they would like to communicate with this assistive robot.

Interestingly, the clinicians proposed a nonverbal communication focus different from the focus of the research team; the clinicians proposed patient care terminology instead of "the state of ART" terminology (e.g., up, down, emergency, etc.) proposed by the research team. A content analysis, developed by frequency analysis, showed that 10 clinicians preferred that ART communicate orientation information (e.g., day, date, time, schedule, nurse's name) to the patients. Eleven clinicians stated that they would program ART to give the patients clinical reminders (e.g., bed safety, fall safety, therapy assistance) to assist in patient care. Despite no overwhelming majority, clinicians also stated that they would like ART to increase their ability to care for the patient by ART communicating to the clinician the patient's vital signs, if the patient attempted to get out of bed, and if the patient attempted to perform their therapy homework.



Customize colors
Patient light control panel
Individual LED light placement
Diagnostic control light panel
Patient related
Top
control buttons lit for staff
string lights
surface light for visibility
Under side light
Also on Reverse side
Base light
Perspective
yellow
blue
green
3-4 lights combined @ each location
Side

Figure 4.1: Presented in a heat map are the clinician results of lighting device selection and arrangement during phase I. The larger squares are LED screens and the dots and lines are individual LEDs.

After the first focus group, the research team determined that clinicians were proposing a different communication focus than the research team (patient care versus the state of ART). Two additional questions were subsequently added to our interview: "If ART had the ability to 'communicate' the way the research team proposed, would you use our NVC platform?" and "Do you think stroke patients would use the platform the research team proposed?" All clinicians who responded to these questions (N=9) said that they and their patients would communicate with ART, given the researchers' proposed platform. Additionally, two participants stated that their decision to use our proposed platform would also depend on the patient's condition. This line of questioning was designed to capture whether the clinicians had a change of mind concerning the NVC embedded in ART. In phase I, evidence suggested that despite the variation in proposed NVC platforms, the clinicians remained willing to communicate with an embedded NVC platform in ART.

4.1.2 Phase II Results

Table 4.2 shows the percentage of clinician preferences for the 15 sounds tested, 7 button actuated sounds and 8 gesture actuated sounds. More than two-thirds (66%) of the clinicians had to select the sound for it to be evaluated as a preferred sound. The clinicians maintained a majority preference for the Tilt Forward (75%), Tilt Back (67%), and Down (67%) button actuated sounds and the Stop (100%), Go (83%), Can't Do (75%), Do Not Understand Request (67%), and Something in the Way (67%) gesture actuated sounds.

The *Down*, *Go*, *Stop*, *Can't do*, *Something in the Way* sounds had a majority for the same sound in both phase I and phase II. Similar to the results during phase I, the *Up*, *Emergency, Come*, or *I'm Thinking* sound did not have a majority result. Clinicians commented that the two *Up* sounds were too similar and that the

Emergency sound needed to be more distinguishable to grab the attention of the staff. Even though the *Pet* and *Reprimand* sounds had a majority preference during phase I, there was no majority during phase II. Three clinicians questioned the reason for the *Pet* and *Reprimand* function of the table. During phase I the *Confirm Request* sound had a 66 % majority preference, but it did not during phase II. After seeing the intention of the action, two clinicians said that this was an added, unnecessary step; and one participant stated that this was more vocabulary for the patient to learn when interacting with ART.

When asked, "If ART could 'communicate' the way the research team proposed, would you use the sounds?"-- the clinicians had a mixed response with no majority preference between *yes, no*, and *maybe*. However, when asked, "Are the proposed sounds appropriate for the hospital setting?"-- nine clinicians stated "yes" and three stated " no". Extending these results, when asked, "Are the proposed sounds communicating what you would expect them to communicate?"-- eight clinicians stated "yes", two stated " no", and two stated " maybe". These results imply that while the sounds met expectations and were viewed as appropriate for the hospital, the participants were reticent in their acceptance of the proposed communication methods.

When asked, "What would you want to customize about the sounds?"-- despite no overwhelming majority, clinicians mentioned volume, a dependency on the patient's characteristics, tone, and making ART speak instead of using the sounds.

When asked, “When would the sounds be used?”--clinicians mentioned sounds would be used when ART moved and in a state of emergency. Finally, when asked, “Could you please describe the characteristics of a patient who might use the sounds?”-- the clinicians mentioned a patient with cognitive issues, visual issues, and mobility issues. This line of questions was meant to narrow the focus of a communication system utilizing sounds similar to the one designed. While no majority preference was determined, clinicians maintained trends seen throughout the study, such as the dependency on patient characteristics.

Table 4.2 shows the percentage preferences of the clinicians for the 17 lighting sequences tested, 9 button actuated sounds and 8 gesture based actuated sounds. More than two-thirds (66%) of the clinicians had to select the sound for it to be evaluated as a preferred sound. The clinicians stated a majority preference for the *Emergency* (83%) and *Reprimand* (75%) button actuated lighting sequences and the *Something in the Way* (100%), *Do Not Understand* (83%), and *I’m thinking* (75%) gesture actuated lighting sequences.

When asked, “If ART could ‘communicate’ the way the research team proposed, would you use the lights?” -- (N=10) six clinicians stated that it would depend on patient characteristics and four stated “yes”. Continuing with these results, when asked “Are the proposed lights appropriate for the hospital setting?” -- six clinicians stated “yes”, and four stated that it would depend on patient characteristics. Then when asked, “Are the proposed lights communicating what you would expect them to communicate?”-- six clinicians stated “yes”, one stated “ no”, and three stated that it would depend on the patient characteristics. These results show that unlike the sounds, which may be more relatable to current technology (e.g., phones and computers), the success of the lighting sequences will depend on patient characteristics.

Button			**Gesture**		
	Sound	**Lighting Sequence**		**Sound**	**Lighting Sequence**
Bend In	■	^A	Can’t Do	B	■
Bend Out	■	■	Come	■	^B
Down	A	■	Confirm Request	■	■
Emergency	■	A	Do not understand request	B	B
Pet	■	■	Go	A	^B
Reprimand	■	B	I’m Thinking	■	A
Tilt Back	A	^B	Something in the Way	B	A
Tilt Forward	A	^B	Stop	A	■
Up	■	■			

Table 4.2: Presented are the clinicians’ sound and light sequence preferences during phase II and tested during phase III[2]. The carrots show specific actions that had a 58% majority and 33% or less than second preference. The intention of the researchers was to test a complete platform.

Despite no majority preference, when asked, “When would the lights be used?”-- (N=8) clinicians stated emergency situations, at night, either day or night, and it would be used with patients with visual issues. Then when asked, “What

[2]A full description of the phase II sounds and light sequence results (tested during phase III) can be viewed here (link is available as of 7/26/2013).
http://www.youtube.com/watch?v=kR2JMkhJBJw&feature=share&list=UUwzcR9h7R6m5B_zQ5E2AHog

would you want to customize about the lights?”-- (N=8) clinicians mentioned the capability to turn the lights on or off , adjust the brightness, limit with what commands the lighting would be used, and change the colors. Finally, when asked, “Could you please describe the characteristics of a patient who might use the lights?”-- (N=10) clinicians mentioned the best patient types would be those who are: cognitively intact, non-verbal, not blind, not suffering from severe traumatic brain injury, not susceptive to seizure, not able to hear well, in need of visual feedback, or in need extra lighting to deal with depression. While there was no majority preference for these questions, clinicians offered a breadth of patient characteristics and customization options, as well as when the lights would be used.

A system like the one developed relies on ambient monitoring to interface with clinicians or patients; therefore, we asked questions to determine the best location for ambient monitoring in the room and on ART. Clinicians were asked, “If there were ambient monitoring in the room, where would it be placed?” (N=10) clinicians mentioned “ under the television” six times, “ in the ceiling” three times, and “at the end of the bed” two times. Finally, when asked, “If there were ambient monitoring on the table, where would it be placed to best serve you as a clinician?”-- 11 clinicians stated that ambient monitoring should be located in the area where the drawer is currently located. However, one clinician stated, “Not on the table but on the wall in the room. Somewhere centrally located to register the person and the table." Based on the results for phase II, the NVC platform was revised and evaluated during phase III.

4.1.3 Phase III Results

For each of the 13 ART communication actions tested during phase III, Table 4.3 shows the following: the number of clinicians who correctly selected the communication action without training; whether the clinician stated that they could use that action with training; if appropriate, which mode best described the action (sound, light sequence, or sound and light sequence combination); and the choice with the highest frequency response. To further clarify these results, the number of ART communication actions selected correct (trimmed M=3.38, SD=1.41) was calculated.

Clinicians correctly selected three of 13 communications with greater than or equal to 50% selection rate. Ten of 13 communication actions had a less than or equal to 30% selection rate, and two communication actions had a 0% selection rate (*Do not understand request* and *Bend in*). Clinicians stated that they often confused the *Can't do*, *Stop*, *Do not understand request*, and *Something in the way* actions. One clinician questioned when and under what conditions the *Reprimand* action would be used. For two ART communication actions (*Reprimand* or *Bend in*), clinicians did not have a 66% majority preference about whether training would improve the action.

ART Communication	No. Correct	With training	Mode that describes the action	Other suggestions
Emergency	7/10	3/3		Can't do (2)
I'm thinking	6/9	3/3		Come (2)
Something in the way	5/10	5/5	Both (2), Sound (8), Light (0)	Stop (2)
Can't Do	3/10	7/7		Something in the way (3), Do not understand (3)
Down	3/10	7/7		Emergency (2), Tilt Back (2)
Tilt Back	3/10	7/7	Both (3), Sound (7), Light (0)	Down (2), Tilt Forward (2)
Go	2/10	Y: 7/8 N: 1/8	Both (2), Sound (3), Light (5)	Bend In (5)
Stop	2/10	Y: 7/8 N: 1/8		Do not understand request (4), Down (2)
Come	1/10	9/9		Thinking (2), Reprimand (2)
Reprimand	1/10	Y: 5/9 N: 4/9		Go (3), Come (2), Stop (2), Thinking (2)
Tilt Forward	1/10	Y: 8/9 N: 1/9	Both (5), Sound (5), Light (0)	Something in the way (3), Come (2), Tilt Back (2)
Do not understand request	0/9	9/9	Both (5), Sound (3), Light (1)	Something in the way (4), Can't do (2), Reprimand (2)
Bend In	0/10	Y: 6/10 N: 4/10		I'm thinking (3), Go (2), Tilt Forward (2)

Table 4.3: Clinician responses during phase III for each of the NVC actions. Without training, clinicians chose the action presented, described what mode best described the action, and after an explanation of the action was given were asked if given training if they could understand the action.

The clinicians rated the ART communication action platform using the System Usability Scale (SUS) (trimmed M=50.31, SD=16.93). Based on Bangor, Kortum, and Miller (2009), when comparing the SUS score to an acceptability range developed by the authors, the SUS score is “marginal low to not acceptable”. Also using Bangor et al. (2009), when comparing the SUS score of the ART communication action platform to a grading scale, the SUS score receives an “F”.

Times mentioned	Explanation of choice	Times mentioned	Participants thoughts after given the correct action
23	Sounded like ...	15	Understand afterwards
14	Light Pattern	11	Sounded like ...
13	The lights looked like ...	8	The lights looked like ...
11	Directional movement of the lights	6	Direction of light movement
10	Only choice left		
8	Pace of lights		

Table 4.4: This table shows an overall categorical frequency analysis of the NVC actions -- explanation of clinician choice and thoughts after given the correct action.

Table 4.4 is an overall categorical frequency analysis of how clinicians made decisions about what ART communication was appropriate and their thoughts after they were given the correct answer. This table shows that clinicians used the look and sound of each action to make decisions about what the platform was trying to communicate. While clinicians mentioned 15 times that they understood the action after given the correct answer, the table shows that in their explanation the action continued to look or sound like another action. For the full table, see Appendix S.

Table 4.5 is a categorical frequency analysis of how clinicians made decisions about what ART communication was appropriate and their thoughts after being given the correct answer for each of the ART communication actions. A closer analysis for those actions answered less than or equal to 20%, reveals that the lights and sounds selected for those particular actions did not meet clinicians' expectations. When initially encountering *Go*, clinicians stated *Go* sounded like it was moving, and clinicians tried to make their selection of the communication action using clues from the *Go* light sequence. Yet, when given the correct answer, clinicians stated that the sound and light sequence did not make sense for *Go* because the sound the light sequence moved toward the clinicians instead of away from them. In the instance of *Stop*, clinicians commented that this action needed a stronger sound to distinguish it from other actions. Clinicians also stated that the random light sequence used for *Come* did not naturally map to the action and that a directional light sequence was needed. The lights for the *Reprimand* action appeared to be similar to other actions (e.g., *Go*, *Come*, *Stop*, *I'm thinking*), and clinicians perceived the lights as movement with a positive connotation. One clinician stated, "I would still say that light pattern doesn't seem like a negative feedback." Clinicians perceived a movement action for *Tilt forward*, but the sound and lights reminded clinicians of other ART communication actions. Furthermore, the *Do not understand request* action sounded or looked like other actions, and clinicians interpreted the negative connotation as a stronger action (e.g., *Something in the way*, *Can't do*, or *Reprimand*). Finally, when the clinicians viewed *Bend in,* they used the direction and the pace of the lights to make their selection. However, the direction and pace of lights did not match clinician expectations. To review the full table, see Appendix R. A discussion of the results is presented in the next chapter.

ART Communication	No. Correct	Explanation of choice	Participants' thoughts after given the correct action
Emergency	7/10	Fast light flash (3) Needs a sound (2) An alarm (2)	Needs a sound (2) Fast light flash (2)
I'm thinking	6/9	Directional movement of the lights (5) Pace of lights (4)	The lights looked like ... (2)
Something in the way	5/10	Sounded like car horn (3) Sounded like stop (2) The sound is cultural intuitive (2)	Understand afterwards (2)
Can't do	3/10	Sounded like a mistake (3) The sound (2) Something in the way (2)	Something in the way (3) Negative indication (2)
Down	3/10	Inflection of sound (4) Similar to expected tilt back sound (2) Only choice left (2) It's a negative sound (2)	Out of choices (3) Understand afterwards (3)
Tilt back	3/10	Expectations of movement (3) Sounded like it was going down (3)	Understand afterwards (4)
Go	2/10	Light pattern (5) Sounded like moving (3)	Light pattern doesn't make sense (5) The sound doesn't make sense (3) Sounded like it's moving (2) Understand afterwards (2)
Stop	2/10	Sounded like ... (3) Deeper sound (2)	Sound needs to be stronger (5) Understand afterward (2)

ART Communication	No. Correct	Explanation of choice	Participants' thoughts after given the correct action
Come	1/10	Only choice left (4) Light pattern (3) Looks like processing (2)	Random light pattern was not correct (5) Light pattern needs direction (2) Looks like ART will move (2)
Reprimand	1/10	The lights looked like ... (7) Light pattern (3) Pace of lights (2)	Perceived lights as a movement (3) Lights were positive, not negative (3) The lights looked like ... (3) I don't understand "Reprimand" (2)
Tilt forward	1/10	There will be movement (4) Alarming (2) The lights looked like ... (2)	A different tone needed (3) The lights looked like ... (3) Sounded like ... (3)
Do not understand request	0/9	Sounded like ... (5) The lights looked like ... (4) It's not an emergency (3) Negative connotation (2)	Interpreted a stronger command (2) Negative connotation (2)
Bend in	0/10	Directional movement of the lights (6) Pace of lights (2)	Direction of light movement (6) Pace of lights (3)

Table 4.5: This table shows the categorical frequency analysis for each of the NVC actions -- clinician choice explanation and clinician thoughts after given the correct action.

CHAPTER FIVE

DISCUSSION

5.1 Summary and discussion of findings

This study sought to understand clinician preferences for an NVC platform composed of lights and sounds for a robot, envisioned for intimate human-robot interaction. The exploratory, participatory design and evaluation three phase process sought to understand a variety of questions for the NVC platform, as well as to use a convergent design method. The subsequent sections discuss the findings for this study.

5.2 General implications of findings

Phase I provided 1.) insights on methodologies to iteratively design and evaluate NVC platforms; 2.) a sense of how clinicians view an NVC platform (patient care versus the state of ART); 3.) the preferences of users (clinicians) of an NVC platform for two features (lights and sounds); and 4.) a sense of whether clinicians and post-stroke patients might use an NVC that was integrated into an assistive robot intended for their use.

Phase II provided 1.) preferences of clinicians of a revised NVC platform (button actuated, gesture actuated, and an LED strip) installed into ART; 2.) a sense of whether clinicians would use the platform, the NVC's appropriateness in the hospital setting, and whether the platform was communicating as expected; 3.) a sense of when the platform would be used, the participants desired customization features, and a refined list of appropriate patient characteristics who might use the platform; and 4.) a sense of where clinicians felt ambient monitoring should be place in the room and on ART.

Phase III provided 1.) initial clinician choice for each of the NVC actions; 2.) insights on how clinicians made their selections for the NVC actions and how to improve the platform; 3.) if given training, whether the clinicians felt that the designed NVC action was appropriate; and 4.) a System Usability Scale score intended to measure the general usability of the NVC platform.

5.3 General limitations of study

The generalizability of these findings is limited by the following factors. First, this study included no patient input about the requirements and design of the NVC platform. Guided by the human factors faculty team member, the team concurred that the complex nature of this novel, cyber-physical artifact would be evaluated initially and iteratively by medical staff before inviting patients to participate in the iterative development of ART. In an attempt to overcome the patient-participant limitation, the clinicians assumed a medium-functioning stroke patient persona in order to determine whether a patient would be able to communicate with ART and how the patient might communicate with ART. Previous studies by our research team have shown that the clinicians are able to estimate a majority of patient preferences, but not the entire scope (Brooks et al., 2011a; Brooks et al., 2012;

Manganelli et al., 2012). Incorporating patient feedback will allow the NVC platform to be fully realized as a communication platform to aid in patient rehabilitation.

While similar research projects have a smaller sample size (Scassellati, Admoni, & Mataric, 2012), evaluation was conducted with a small sample (10-13) of clinicians from Roger C. Peace Rehabilitation Hospital in Greenville, South Carolina. As no demographic data were collected, these results cannot be generalized to other medical disciplines either at Greenville Health System or in the Southeast (i.e., cancer center, children's hospital, heart and vascular care, general medicine, primary care, surgery, women's hospital).

Finally, a limitation of the study was the time constraint given to the research team. This study was conducted in a one hour time slot during the clinician's lunch hour. Due to work- related responsibilities, some clinicians arrived after their scheduled time and ate lunch during the session. Often, a 60-minute session would be reduced to a 45-minute session or less. Given these time constraints, direct interaction was necessary to ensure that the session would only last during the time given.

5.4 Study discussion organized by pre-study hypotheses

An assistive robot that conveys nonverbal communication can be readily understood by users, even those who are medically at-risk, such as post-stroke patients. An examination of the pre-study hypotheses show mixed results for this study. Shown in phase III and discussed further in the following sections, using the evaluated NVC platform requires training. Thus, future research is required to design a more intuitive platform and to understand the amount of training required for an NVC platform in a rehabilitation setting. On the other hand, the NVC platform presented here might arguably work very well outside the clinical setting for non-clinical users interacting with and supported by assistive robots in everyday life rituals occurring at home, work, and schools. *Additionally, an assistive robot communicating in NVC does not need to be humanoid or otherwise explicitly life-like in appearance to be readily understood by users*. Throughout this research, and despite the low usability ratings of the NVC platform, clinicians easily assumed communication with ART via the NVC platform presented them. As an example, in phase I, clinicians expressed a need for a patient care terminology in addition to expressing a preference, in phase II, for the NVC platform in the hospital. *Finally, our entirely nonverbal human-robot communication will be perceived as a desirable, low-invasive, and expedient communication mode for HRI, particularly in intimate human-robot interactions*. The results of this study suggest that this hypothesis requires further research. In phase I, clinicians stated unanimously that they would use the platform; yet, in phase II, clinicians did not have a majority preference. Furthermore, clinicians stated that while the sounds could be used in the hospital, patient characteristics would determine the usability of the lighting sequences. Additionally, the clinicians' low correct selection rate of the NVC actions in phase III suggest that the platform would not be a low-invasive or an expedient communication mode. Broadly, this exploratory research provided requirements for an NVC platform in the rehabilitation setting, despite the poor results for this particular NVC platform. (Refer to Figure 5.1 for an explanation of these findings.) A discussion for each phase of this study is presented in the following sections.

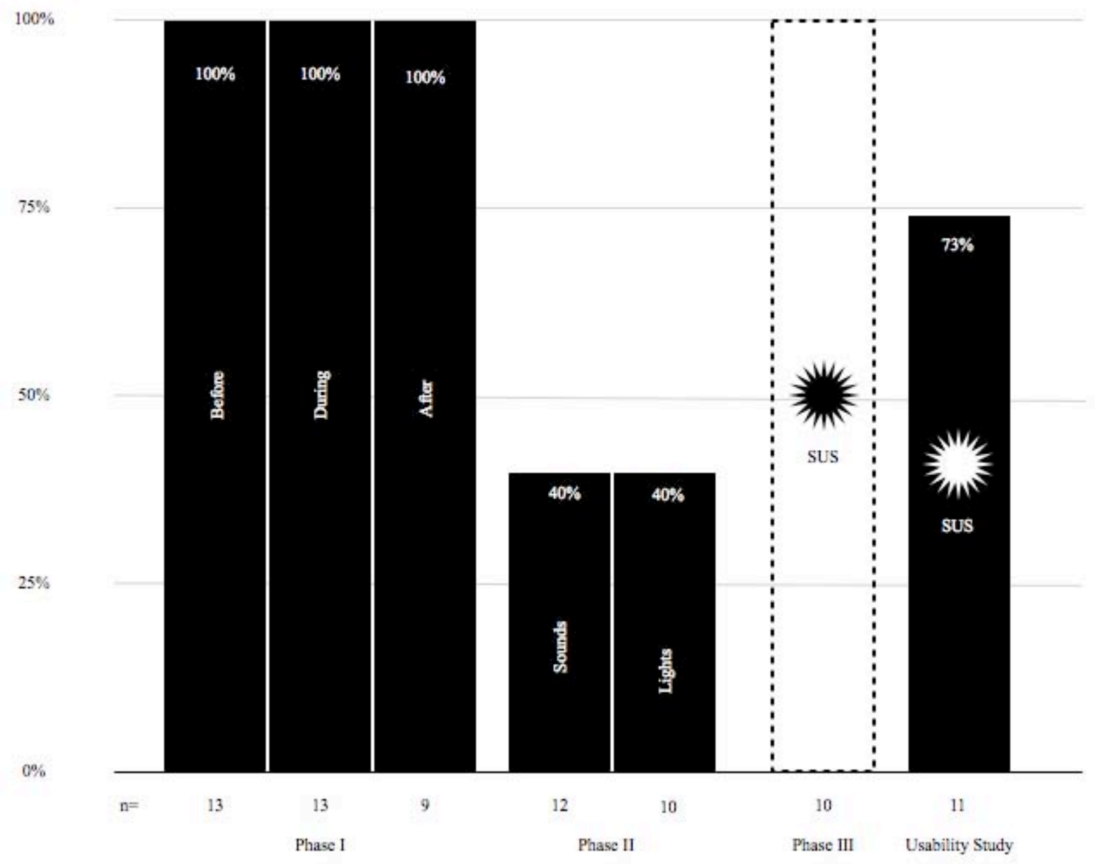


Figure 5.1: This figure shows the variability of clinician acceptance for the NVC platform throughout the research. One can see that acceptance declined during phase II but that over time (the final usability study -- Chapter 6) acceptance increased. The System Usability Study scores show that the researched NVC requires further testing to be used in practice.

5.4.1 Phase I Discussion

In an increasingly digital society, how might a non-verbal communication (NVC) platform be embedded in the Assistive Robotic Table to provide clinicians *opportunities to best understand and interact with a patient's recovery?* The research team developed an NVC consisting of lighting sequences and sounds and a gesture command interface, embedded into ART, to communicate with clinicians and their stroke patients. In phase I, clinicians provided their preferences for what sounds should be used as well as where and what type of lights should be used in the NVC platform. Additionally, clinicians provided insights to the question: *What are the requirements for an NVC platform used in the rehabilitation of post-stroke patients?*

Clinicians desired a platform that augmented their ability to care for the patient. Thus, the clinicians provided requirements for a patient care terminology platform. This terminology was seen as an expedient to help healthcare providers and patients mutually interact in patient recovery. However, the research team developed an NVC platform intended to allow the clinicians and patients a more efficient and effective way to operate ART. As shown, the clinicians stated that they would still use the developed NVC platform and speculated that their patients would use the platform as well.

Throughout phase I clinicians were asked, "If ART had the ability to communicate, would the clinicians communicate with ART?" The clinicians responded three different times (before, during, and after the study session) that they would communicate with ART. Before and during the session, participants unanimously agreed that they would communicate with ART. However, when asked after the session, two participants said communicating with ART would depend on a patient’s characteristics. While clinicians had knowledge of ART and had been participants in previous research studies, the sounds were played in the abstract, without a physical ART present and without incorporating the sounds into ART. As demonstrated in a later phase, the unanimous sentiment of the clinicians to communicate with ART changed and focused on the variability of the patient characteristics.

As stated, the clinicians proposed a different type of communication platform than the research team’s communication platform. The clinicians focused their terminology on patient care, as opposed to the state of ART (e.g., up, down, tilt forward, tilt back). While the clinicians did not reject the research team’s platform, future studies should address the discrepancy between the two systems of terminology. Future studies might begin with patient care terminology and explore possible input and output dialogs to best address efficiency and patient care in the rehabilitation context.

Interestingly, clinicians declared areas that they did not want the NVC platform to interfere with their interaction with the patients. The *Bend in* and *Bend out* sounds (75%), dedicated to a novel therapy surface component, were rejected. Clinicians thought that these sounds would distract from the patient’s therapy because it would be perceived as extra noise when clinicians needed patients to focus on the rehabilitation tasks. Future studies should investigate what scenarios are best served for an NVC platform to meet clinician and patient needs, possibly using other technologies (e.g., personal handheld devices) as case studies to determine where one might want communication and in what scenarios they would like to turn it off or have it only present one mode (i.e., cell phone - non-verbal cues versus sound). The needs of clinicians and patients are different; therefore, a system that adapts to the user is required.

5.4.2 Phase II Discussion

Interestingly, in phase II participants had mixed reactions and did not have a majority preference to the question, "If ART had the ability to communicate, would the clinicians communicate with ART?" However, the clinicians stated a high preference for both the appropriateness of the communication platform in the hospital and if the communication platform was communicating as expected. These results appear to illustrate the clinicians' low acceptance of the technology for their individual use, despite a majority positive preference for the technology. As an example one clinician stated, "I probably wouldn't use the sounds or I wouldn't use them all . . ." But when asked about use of the NVC platform in the hospital the clinician stated, "yes, I don't think the sounds would frighten or disturb." A future study should explore the reason behind this discrepancy.

The clinicians desired specific, customized features for the sound and light elements. The clinicians wanted to be able to adjust the volume of the sounds, as well as, turn on and off the lights. While these features are similar to those associated with other personal handheld devices, there is a concern that once this platform has these notifications removed, they will be removed indefinitely. The ART communication actions should be designed in a manner that the sounds and lights enhance the clinician or patient experience with the assistive robot. Future studies using the iterative design process should explore clinician preferences for the best interaction with the sounds and lights, as the results of the one proposed (the state of ART) do not show a high acceptance of the platform.

Finally, the clinicians provided an expanded list of patient characteristics for an NVC platform. Thus, the clinicians provided further clarity to the question: *What are the requirementsfor an NVC platform used in the rehabilitation of post-stroke patients?* For the sounds, these characteristics included a patient who is cognitively intact but has visual and/or mobility issues. For the lighting sequences, these characteristics included a patient who is 1.) cognitively intact with 2.) visual or 3.) mobility issues; a patient who is 4.) not verbal, 5.) not blind, 6.) not suffering from severe traumatic brain injury, 7.) not susceptive to seizure; as well as a patient who might be 8.) hard of hearing, 9.) in need of visual feedback or 10.) in need of extra lighting to deal with depression. Future research should explore these patient characteristics and seek to design a communication platform to address the needs of a broader, non-stroke related patient population with similar characteristics.

5.4.3 Phase III Discussion

Will clinicians prefer an Assistive Robotic Table with an NVC platform over current stroke patient therapeutic practices? In phase III, the research team measured the effectiveness of the NVC platform by having the clinicians respond to the platform without training by selecting NVC actions, complete an SUS survey, and explain their NVC actions selections. As was shown -- the low correct selection rate, a marginally acceptable SUS score, and clinician comments equating the lighting sequences and sounds to other NVC actions -- these results suggest that the NVC platform embedded into ART is not preferred over current stroke therapeutic practices. Furthermore, a goal of the research team was to design an intuitive platform. However, these results suggest that a complex platform, like the one designed, requires adequate training before use. An in-depth discussion of these results is presented in the following section.

Phase III sought to quantify the usability of the designed ART communication platform. Initially, clinicians were asked to match each ART communication action with a presented light sequence, sound, or a combination of light and sound. Despite clinicians being a part of the iterative design process, without training before the testing session, clinicians were unable to correctly select the actions when presented to them. As stated previously, 10 of 13 actions had a less than or equal to 30% correct selection rate (trimmed M=3.38, SD=1.41). These results present two issues with this research method. The first issue was the research team's ability to correctly and naturally map the lights and sounds to each action without clinicians having prior knowledge of the platform presented to them. One method that might be beneficial to future research would require clinicians to develop the sounds and lights themselves.

Another method might include multiple iterations of the phase I sound and light creation sessions by the research team. The second issue is the training required to use the designed platform. This training might be similar to the training required to use any technological device (e.g., phone, television, computer, etc.). Future research might explore how much and when training is required.

Important results of phase III included how clinicians made their choices and how to improve the platform. Primarily, clinicians used the sounds or lights presented to them to decide on the action. One participant stated that he or she used the research assistant's demonstration of a gesture-based action (*Something in the way*) to make a decision. However, clinicians stated, "It sounds like . . ." or "It looks like . . ." as they attempted to interpret each action.

The clinicians stated that some actions required different sounds or lights to make the action intelligible. For example, clinicians stated that *Emergency* needed a sound, *Stop* needed a stronger sound, and *Go* and *Come* needed a different light sequence. However, this is in contradiction with a large majority of clinicians who stated that these actions could be used with training. In previous phases, the *Emergency* sound did not have a majority preference, and the sounds did not satisfy clinician preferences for *Emergency*. However, when asked to develop new sounds, clinicians were reticent. This illustrates the point that more iterations by the research team are needed.

As an extension of these results, clinicians stated that they often confused the *Can't do*, *Stop*, *Do not understand request*, and *Something in the way* actions. All four of these actions were gesture based commands and communicated with at least a sound. While *Do not understand request* and *Something in the way* also communicated with a lighting sequence. Without training, *Something in the way* had the highest correct selection rate at 50% and *Do not understand request* had the lowest correct selection rate at 0%. In the current intended ART interaction, the gesture command will not change. Therefore, clinicians should clarify how they view each of these NVC actions and their relation to ART. In future iterations of this study, the research team should provide multiple iterations of the NVC sounds and lighting sequences at multiple times throughout the study (there was only one pre-study creation phase for the NVC actions). Finally, each of these NVC actions was evaluated with the clinician seated in front of ART. In the future, clinicians should use ART as it is intended, by positioning it around a hospital room, and encounter the NVC platform in an everyday environment. This may change the clinician's perspective on the NVC actions and the way each action is presented via the lighting sequences and sounds.

Finally, based on the SUS score, the NVC platform was not ready to be implemented in current rehabilitation practice. As stated previously, when comparing the NVC platform SUS score (trimmed M=50.31, SD=16.93) to an acceptability range (marginal low to not acceptable) and to a grading scale (F), the platform was poorly scored. Comparing these results to the results of previous phases, the number of clinicians who stated that they would communicate with the designed platform decreased from 100% during phase I to no majority preference (sounds) and dependency on patient characteristics (lights) during phase II. Also during phase II, clinicians stated that the designed platform was appropriate for the hospital and was communicating as expected. Interestingly, in phase III when asked, "*If given a period of training, do you think that this communication method is appropriate for this action?"* -- for all but two actions (*Reprimand* and *Bend in*) clinicians had a 66% majority preference that they could learn the action. Yet, when the clinicians completed the SUS to evaluate the system, based on the proposed interaction with the platform and not the number of actions they may have answered correctly, the ART communication platform scored low. Again, this may illustrate that the current NVC platform was not ready to be used in a rehabilitation setting and would require further research.

These results show that as the platform developed, clinician preferences for the platform declined, despite the high rating by the clinicians for the appropriateness of the platform in the hospital and its ability to communicate as expected. Comparing these earlier results with the results from phase III, there is a contradiction, as the clinicians believed highly in their ability to learn the platform but scored the overall usability low. These results parallel clinician comments in Manganelli, et al. (2013) about the current over-the-bed table used at the rehabilitation hospital. Clinicians expressed frustration about the usability of the device but were confident in their ability to use it to accomplish patient care objectives. Anecdotally, it appeared clinicians preferred “analog” therapy devices manipulated by the therapist and patient versus “digital” devices that may not work correctly, may break down, and *required* therapist work arounds to make the device work properly. While not investigated as a part of this research, future research should investigate the contradiction between clinician feelings to overcome technologies meant to simplify and aid in patient care and the low usability ratings.

In these three phases clinicians were not asked to estimate how long it would take them to learn the platform. However, comments by the clinicians about simplifying the platform to a limited number of gestures, light sequences, and gestures would speed up the process. Given the current number of possible combinations clinicians may take less than 30 minutes to learn the platform basics but, dependent upon how often clinicians interact with ART, it may take a couple of weeks for clinicians to learn the whole platform. Thus, clinicians may need to carry around a “cheat sheet” to completely understand the platform - an unfavorable solution. Future research should investigate possible solutions to simplify the platform to be more usable.

5.5 Future Directions

A goal that was not met during this study was the development of an intuitive platform that requires no training to be used in a rehabilitation hospital. Future studies should seek to expand the iterative design process throughout the duration of the study. Future studies should also develop a method that includes multiple rounds of divergent and convergent iterative design and evaluation techniques to ensure the ultimate success of the NVC platform. A refined methodology would also include patient feedback as well as an expanded participant pool and increased number of participants to apply the lessons learned in this study to multiple disciplines. Our current study was limited to the time constraints and ensuring the study was compliant with the current constraints of working within a real clinical setting.

An unexpected finding of this study was clinician preferences toward new technology in healthcare. Despite a decrease in clinician acceptance of the platform and a low usability score, clinicians believed that they could be trained to overcome the deficiencies of the NVC platform. Future research should attempt to find what other healthcare technologies clinicians feel they have had to overcome, as well as why and how clinicians feel that they have to overcome these technologies. These areas may open the possibility for research and may be appropriate for an architectural robotic solution.

In NVC research, researchers should consider user population input, ambient monitoring, the ability of the NVC platform to “understand” (i.e., learn of) its users, and the ability of an assistive robot like ART to convey information. A specific topic of future research, derived from this work with clinicians at a rehabilitation hospital, is an NVC platform that focuses on patient care terminology instead of the state of the assistive technology. NVC platforms must be integral with the robot, developed to accept multiple sources of input, act on the data given, and present data back to the user. More broadly, a dynamic NVC like ours may improve job performance of caregivers and increase patient satisfaction.

CHAPTER SIX

FINAL USABILITY STUDY

6.1 Introduction

"Usability testing, in general, involves representative users attempting representative tasks in representative environments" (Lazar, Feng, & Hochheiser, 2010, p. 252). This methodology provides researchers the opportunity to find flaws and potential user problems with a designed interface or product (Molich & Nielsen, 1990; Nielsen, 1992, 1993). Subsequently, usability testing requires a small number of participants, a similar constraint to the elaborated study (Lazar et al., 2010). This usability study is the conclusion of the exploratory, three phase, participatory design and evaluation of a non-verbal communication (NVC) platform.

This chapter focuses on a final usability study conducted to find potential problems and to understand clinician's thoughts about components of the NVC platform in a real world environment, a rehabilitation hospital patient room. Although the the ambient monitoring NVC action (*Emergency*) and the NVC notification action (*Something in the Way*) were hardwired into the system, clinicians understood and interacted with the platform. Also, clinicians observed a gestural interface, a proposed way to interact with the NVC platform, as it interacted with the Assistive Robotic Table (ART) in real-time. The results of this study are presented in the following sections.

6.2 Participants

Volunteers for this study consisted of research team members and a convenience sample of clinicians at Roger C. Peace Rehabilitation Hospital (RCP). Eleven subject-matter experts – all clinicians, including doctors and occupational and physical therapists – participated in the final usability study of the research. In the interest of protecting the privacy of this small, exploratory sample population and based upon the conditions of the approval for this study design by the hospital's institutional review board, demographic data are not presented here.

6.3 Study Design

6.3.1 Independent Variables

Independent variables for the study included two NVC actions, *Emergency* and *Something in the Way* and three NVC gesture commands *Rest, Therapy*, and *Up/Down*.

6.3.2 Dependent Variables

Dependent variables included open-response and forced choice questions about the *Emergency* and *Something in the Way* actions, open-response questions about the *Rest, Therapy*, and *Up/Down* gestures, LIKE, NEED, and EASY TO USE subjective preference ratings for each command as well as for the overall NVC platform, and a System Usability Scale for the overall NVC platform.

6.4 Procedure

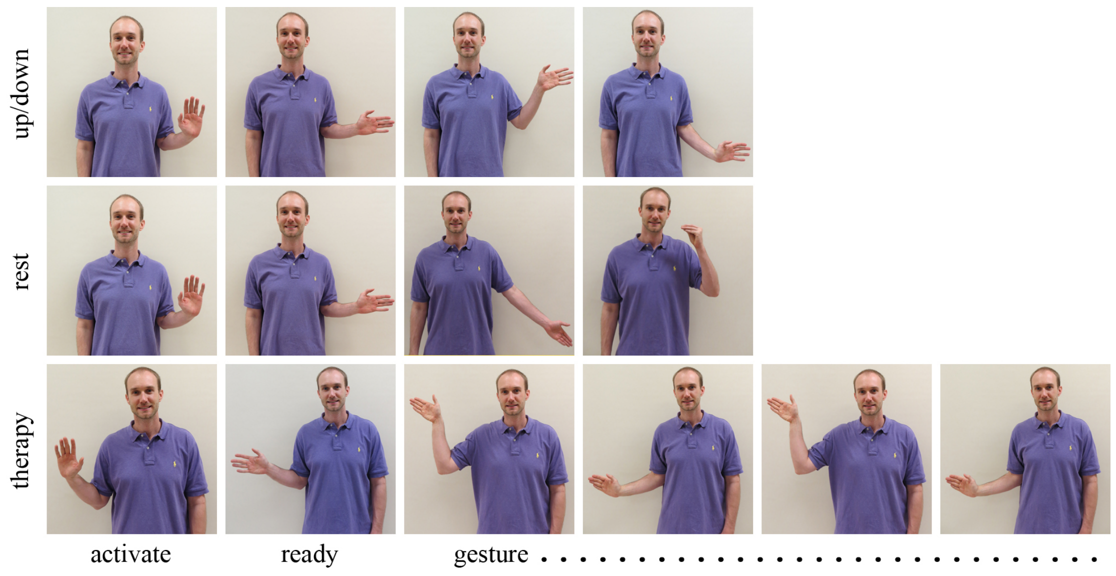


Figure 6.1: Presented is the proposed NVC gesture command interface. Each gesture must be activated and begun in a ready position before beginning the command.

Clinicians at RCP completed the usability study in an individual session. Eleven individual sessions were conducted over two days in a hospital patient room. This session had multiple aims, as it was the final usability session for ART and the NVC platform. Thus, the clinician was told the purpose of the study was to evaluate the usability of the Assistive Robotic Table: 1.) around the bed, 2.) around a chair for therapy, and 3.) a gesture command interface for the NVC platform. As a part of the study session, the clinician positioned around the room a fully functioning final ART prototype, integrated with the NVC platform consisting of lights and sounds. Each session had one research moderator and one note-taker. Each person sat in the room but out of the way of the participant. Two feedback methodologies were used: a forced-choice methodology and open-ended response.

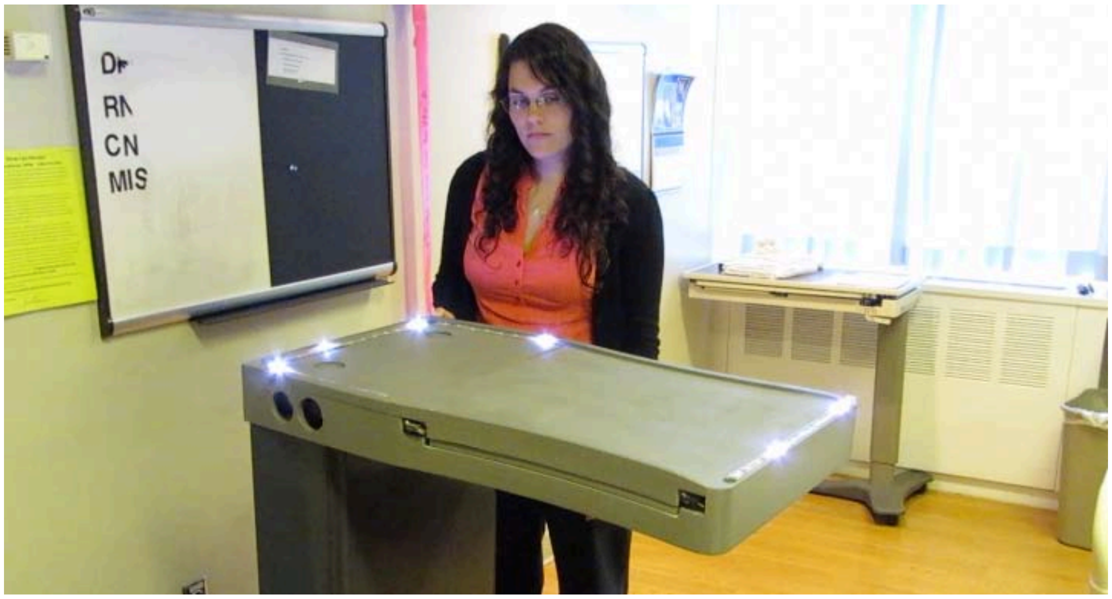
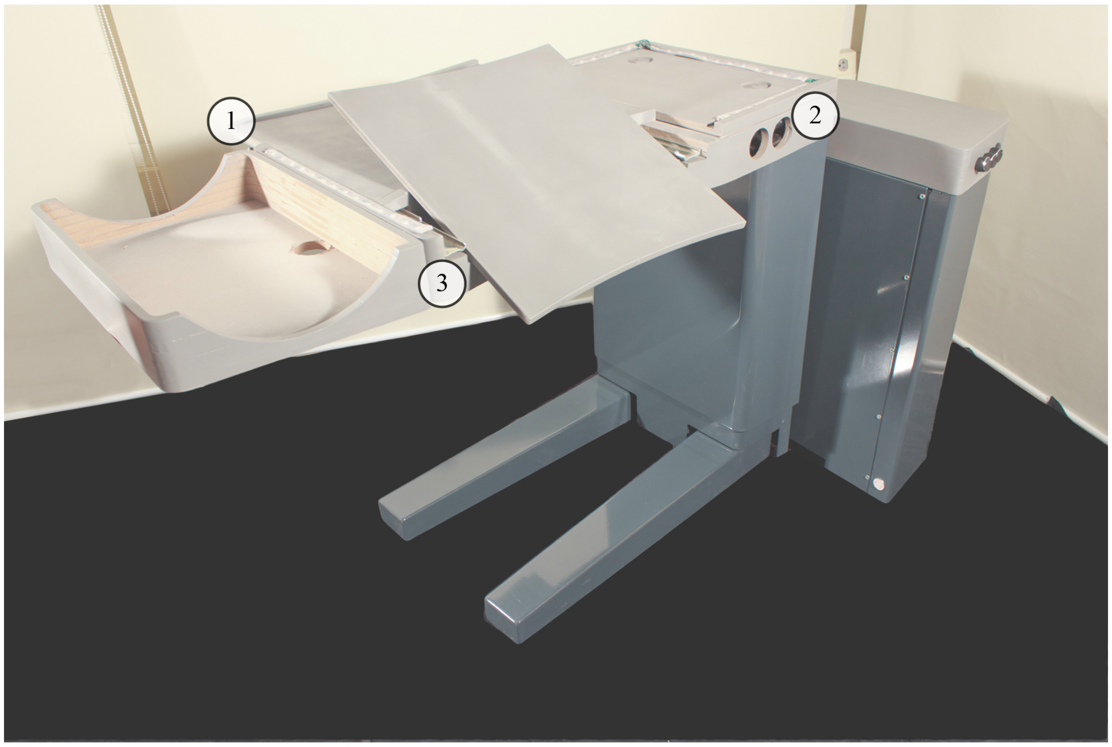


Figure 6.2: Presented is the NVC platform evaluation set-up for the Usability Test. When the *Down* button was pressed the first time, the *Something in the way* action activated.



1. LED light strip 2. RGB-D sensor 3. Speakers (not pictured)

Figure 6.3: Presented here are the NVC platform components. The LEDs make up the lighting sequences and the Microsoft Kinect® activates the gesture command interface. (Not pictured are the speakers.)

To begin the study, ART was in its lowest possible position and located against the wall perpendicular to the hospital bed. The clinician was instructed to move ART over to the bed and position ART over the bed, as if a patient were using ART to eat and then to read by extending the flip-up surface. When the clinician pressed the "up" button for the first time, the *Emergency* NVC action would activate. Then the participant answered two open-ended questions as well as a forced-choice question about interacting with the *Emergency* communication action. Each clinician was given a printed sheet of possible NVC communication actions to answer the forced-choice question. When the participant pressed the "down" button for the first time, the *Something in the Way* NVC action would activate. The clinician would evaluate *Something in the Way* similar to the *Emergency* action. The *Something in the Way* action did not operate for one participant. After completion of the ART portion of the usability study, the clinician evaluated three NVC gesture commands -- *Up/Down*, *Therapy,* and *Rest* -- that might be used to interact with ART. (See Appendix V for a complete list of questions.)

To evaluate the gesture commands, the research moderator demonstrated how to activate the command and how to begin the command. In response, ART would move in a preprogrammed way. *Up/Down* was programmed to move in real-time up or down, dependent upon the movement of the moderator; *Therapy* was preprogrammed to move up for a designed interval; and *Rest* was preprogrammed to move down for a designed interval. To evaluate the *Therapy* gesture*,* the clinician was told to imagine that he or she performed therapy with the same patient and at the same height (33 inches from the floor) each day and would use this gesture to position the table. The therapist would then use the *Therapy* command to position ART to a preprogrammed height to begin therapy. When evaluating the *Rest* command, therapists were told to imagine that after completing a meal a patient would push ART away from the bed. The patient would then use the *Rest* command to position ART to a preprogrammed, desired height next to the bed.

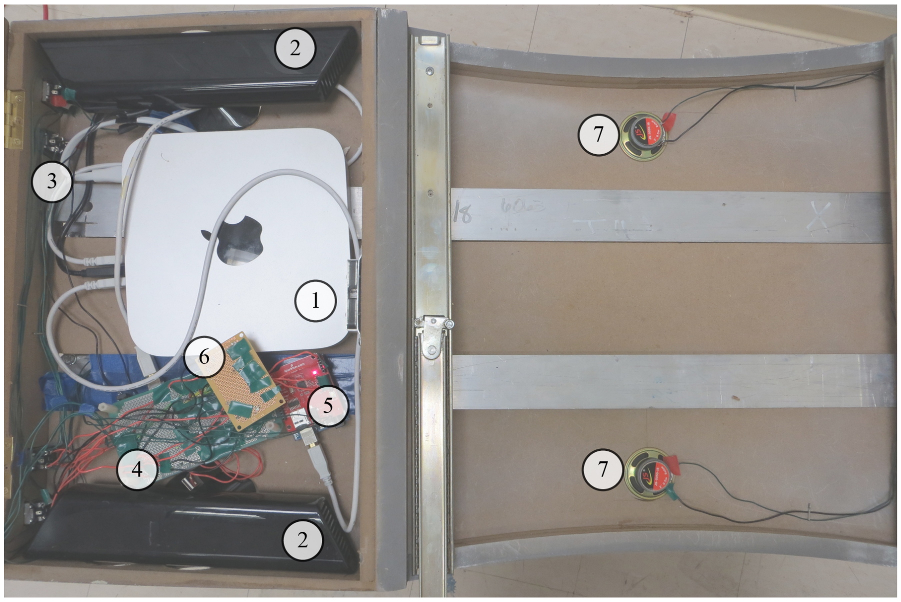

2
3
7
1
6
5
4
7
2
1. Mac Mini
2.Microsoft Kinect
3. Rocker switcht
4. (See Appendix V for electrical diagram)
5. MP3 player shield
6. (See Appendix V for electrical diagram)
7. 0.5W - 80 ohm Speakers

Figure 6.4: Presented are the components that comprise the NVC platform.

After the completion of each gesture, the clinician answered open-ended questions about their preferences for the gesture and ART's response, and completed subjective preference ratings (LIKE, NEED, and EASY TO USE)-- three scales developed within our lab to triangulate preferences (Smolentzov, 2010). Refer to Table 6.1 for the subjective preference rating scales. After answering questions related to the final gesture, the clinician answered open-ended questions about using ART with the intended NVC platform gestures.

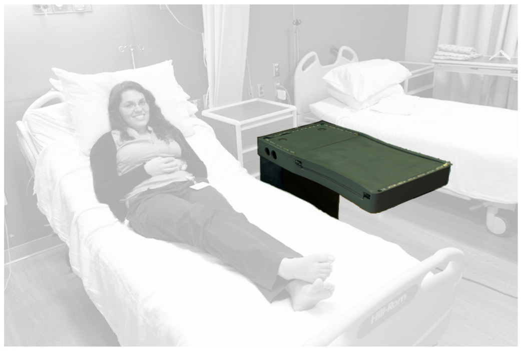
	1	2	3	4	5
Like	Strong Dislike	Dislike	Neutral	Like	Strongly Like
Need	Never Need	Rarely Need	Sometimes Need	Often Need	Always Need
Easy to Use	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

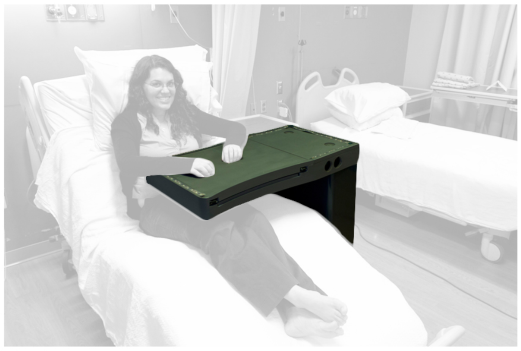
Table 6.1: An example of the subjective preference rating scales used to evaluate the NVC platform.

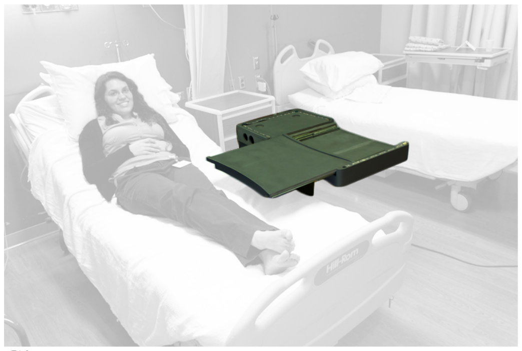
Finally, the clinician completed the evaluation by answering questions about his or her acceptance of the NVC platform, an estimation of how long it would take the clinician to learn the platform, if the NVC platform was a productive line of research, and whether the clinician would rather speak to ART and have ART speak back to him or her. The clinician also completed the subjective preference ratings and a SUS survey to evaluate the entire NVC platform, including the lighting sequences, sounds, and gesture interface. After the completion of the session, the clinician was thanked for his or her time and excused. The session took less than 60 minutes.

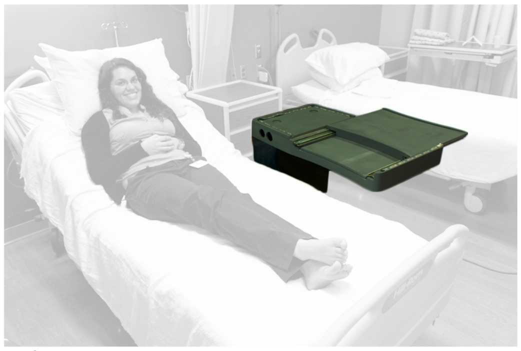
6.5 Results

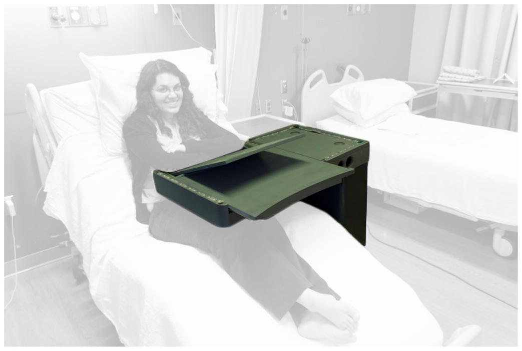
The results focus on three main areas. The first area is whether the clinicians understood the NVC actions within the context of the usability testing within a hospital patient room. The second area is the clinician's acceptance of the proposed gesture interface as well as the clinician's acceptance of the overall NVC platform. The results are discussed in the subsequent section.


Rest


Eating


Give


Take


Read

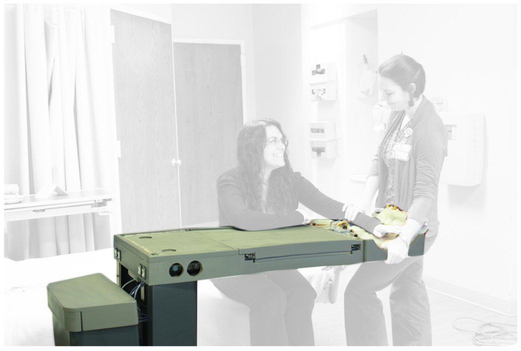

Therapy

Figure 6.5: Presented are the six proposed ART positions. It is intended that each of these positions would address a particular ART required function.

When clinicians were presented with the *Emergency* NVC lighting sequence action, fast paced flashing lights, clinicians chose other NVC actions including: *Can't do* (5), *I'm thinking* (2), *Something in the way* (2), *Stop* (1), and *Do not understand request* (1). One clinician stated, "I thought that flashing light was emergency, but because of what I did, I would say it's emergency because there's nothing … I'll pick ‘can't do.’” In order to make their decisions (N=10), the clinicians primarily used the lighting sequence presented (6). In most cases the clinicians knew that the NVC action was trying to alert them about something (5) but they were not sure what the NVC alert meant (4).

Did you like ART’s response to the gestured command?			
	Yes	Maybe	No
Up/Down	10	1	0
Therapy	7	2	2
Rest	10	0	1

Table 6.2: Presented are the clinician preferences for each of the gesture commands. Compared with the other two gesture commands, the clinicians were reticent about accepting the *Therapy* command. They cited the desire to use conventional rehabilitation practices.

When clinicians were presented with the *Something in the Way* NVC lighting sequence and sound action, the lights would individually illuminate, a sound played sounding like “urrp urrp”, and then the lights individually turned off, clinicians (N=10) chose *Can't Do* (4), *Something in the Way* (2), *I’m Thinking* (1), *Stop* (1), *Do not Understand Request* (1), and “Maybe I’ve put it down” (1). When making their decisions, clinicians used both ART not doing the action requested (down) and the lighting sequence (4), the lack of movement (3), or the lighting sequence (2). Clinicians had mixed thoughts about what had occurred with no major preference--stating “it couldn’t do it” (3) or “it’s too close to the bed/patient” (2). One clinician stated, “Well, it's not happy about something; I would think I did something wrong.” These results show that the clinicians understood that the NVC was trying to alert them to something, but the alert was not intuitive.

Overall, when the clinicians evaluated the NVC gesture commands, the clinicians had a strong majority like preference for ART's response to the command. Refer to table 6.2 for the clinicians' preference for each of the gesture commands. When asked what they liked about ART's response to the *Up/Down* gesture, clinicians stated, "It's intuitive" (4). When asked what they liked about ART's response to the *Therapy* gesture, clinicians stated, "It responded" (4). Even though the participants were told that the *Therapy* gesture was based on the American Sign Language gesture for *Therapy,* one clinician stated, "Because the gesture went down and the table went up - it was a confusing gesture - the direction of the table being inconsistent." When asked what they liked about ART's response to the *Rest* gesture, clinicians stated, "it responded" (4) and "preprogrammed to certain height with a specific gesture" (3). Primarily, clinicians liked that ART responded to the command given. These results combined with the ratings (see Table 6.3) show the clinicians' preferences for the NVC platform.

When using the three rating scales (LIKE, NEED, and EASY TO USE) for each of the NVC gesture commands, the clinicians "Liked" each command. However, the clinicians thought that the *Therapy* gesture would only be "Sometimes Needed". Clinicians were between "Neutral" and "Agree" for EASY TO USE concerning patients using the *Rest* command, especially patients in the bed. Overall, clinicians rated the NVC platform as "Like" and "Often Needed", but were between

“Neutral” and “Agree” about its ease of use. In order to fully understand these ratings, clinicians were asked a series of open-response questions.

	Up/Down	Therapy	Rest	Overall
Like	4.18	3.82	3.82	3.82
Need	3.73	3.18	3.73	3.82
Easy to Use	4.00	3.64	3.45	3.45

Table 6.3: Presented are the subjective ratings for the overall NVC platform and each of the gesture commands. Similar to the results of Table 6.2, the *Therapy* command scored the lowest for the Need rating.

When asked, “If ART had the ability to ‘communicate’ the way the research team proposed, would you use our platform?”-- eight clinicians stated *yes*, two stated *no*, and one stated *maybe*. Of the participants who stated they would use the NVC platform, the clinicians stated “ease of use” (4), “more functionality” (3), and “patient autonomy” (1) were the reasons they would use the NVC platform. However, not all of the clinicians preferred to use the platform. One of the clinicians who would not use the NVC platform stated, “Well I just find it ridiculous to be honest it's completely non-functional it's way over the top. It's going from the first wheel to the spaceship - it's too big of a jump. Just in this setting more complicated is not better in a hospital setting.”

Continuing this line of questioning, when asked, “Do you think this is a productive line of research?”-- seven clinicians stated *yes*, three stated *no*, and one stated *maybe*. Of the participants who stated the NVC platform is a productive line of research, the clinicians stated the reasons as, “there is a general need” (5), “the patients have a need" (1), and "the technology is less expensive" (1). The participant who was not sure stated the following:

> "I think there are a lot of good features that are a part of the table that would be beneficial for patients and care providers. There are some things that feel more frustrating or not necessary. If people have the money to purchase and use it at home [but] we have difficulty to get [the] patient to use a shower chair at home. I don't think they would be useful without money."

One participant who stated they did not think it was a productive line of research stated the following:

> "I just I think it's over the top for this particular setting and clientele I don't really understand if it's be helpful for patient and helpful for clinicians. I think it would be 10 times worse and it's very bulky. For clinicians our time is valuable and limited and it would cause me more stress."

These statements show that the NVC platform has barriers it will need to overcome in order to be accepted by a wider audience within the rehabilitation setting.

Finally, when asked, "Would you have preferred to simply speak to ART and ART speak to you?"-- six clinicians stated *yes*, four stated *no*, and one stated *maybe*. Of the participants who stated that they would prefer to speak to ART, the clinicians stated "speaking is easier than hand gestures" (4) and "the technology is appropriate" (2) as reasons. Of the participants who stated that they would not prefer to speak to ART, the clinicians stated "hand signals are easier in a room filled with other noise" (2), "voice recognition is difficult" (1), and "I don't like Siri" (1) as reasons. One participant stated, "To me hand signals would be easier. I speak to the patient all time and how would the table differentiate from me talking to the patient [versus the] table?"

In your estimation, how long would it take you to learn how to use the communication platform you used today?	
30 Minutes	3
1 Hour	2
1-2 hours	1
A couple of days	1
A few days - consistently	1
1 week - consistently	1
2 weeks	2

Table 6.4: Presented are the clinicians' estimation of time to learn the presented NVC platform. It is disconcerting that the current platform may require up to two weeks to learn. Given the time sensitive nature of healthcare it would be necessary to learn the platform in under one hour.

Despite the clinicians stating a preference for this NVC platform, when the clinicians rated the ART communication action platform using the System Usability Scale (SUS) (trimmed M=40, SD=14.49), it was rated poorly. Based on Bangor, Kortum, and Miller (2009), when comparing the SUS score to an acceptability range developed by the authors, the SUS score is "not acceptable". Also using Bangor et al. (2009), when comparing the SUS score of the ART communication action platform to a grading scale, the SUS score receives an "F". The result of SUS contradict the clinician's statements about the NVC platform and will be discussed further in the subsequent section.

Finally, the clinicians were asked to estimate the time required to train and learn the NVC platform. When asked, “How many teaching intervals (response and feedback) would you be willing to go through before you would expect ART to learn a given gesture command?”-- clinicians stated about three times (trimmed M=2.94, SD=1.18). When asked how long would it take for the clinicians to learn the platform, six of 11 of the clinicians stated under two hours. Refer to Table 6.4 for a complete breakdown of the clinician’s estimation. A discussion of the results is presented in the subsequent section.

6.6 Discussion

The usability study sought to understand clinician preferences for an NVC platform comprised of lights, sounds, and gestures for a robot, envisioned for intimate human-robot interaction. The study sought to understand how the NVC platform would perform when clinicians used the Assistive Robotic Table, clinician preferences for a gesture interface, and questions about clinician acceptance and learnability with the NVC platform. In the subsequent sections is a discussion of the findings for this study.

6.6.1 General implications of findings

The usability study provided 1.) the ability of clinicians, without training, to understand two NVC actions; 2.) clinician preferences for the gesture interface; 3.)

clinicians' estimation of the time required to teach and learn the NVC platform; 4.) clinicians' preferences for a verbal interface versus the designed NVC platform; 5.) a System Usability Scale score intended to measure the general usability of the overall NVC platform; and 6.) whether the clinicians would use the final designed NVC platform.

6.6.2 General limitations of study

The generalizability of these findings is limited by the following factor. Similar research projects, like the one conducted, have a small sample size (Scassellati, Admoni, & Mataric, 2012); evaluation during this study was conducted with a small sample (11) of clinicians from Roger C. Peace Rehabilitation Hospital in Greenville, South Carolina. As no demographic data were collected, these results cannot be generalized to other medical disciplines either at Greenville Health System or in the Southeast (i.e., cancer center, children's hospital, heart and vascular care, general medicine, primary care, surgery, women's hospital).

6.6.3 Usability Study

At the beginning of the study, clinicians were presented with two NVC actions. While the clinicians knew ART was trying to alert them to something, overall they were unsure of what the alert meant. Despite the clinicians participating in the design of the NVC platform and using ART in a more realistic scenario, these results support the results of phase III that users will require training before using the platform. These results suggest that the platform does not meet a goal of being intuitive without training. Future research should seek to design a system that is intuitive that requires minimal lighting sequences, sounds, and gestures commands. Future research should also design and evaluate an appropriate training protocol to ensure users adopt the NVC platform quickly.

Overall, the clinicians preferred the presented gestures. The preferences are shown by clinician responses to the open-response questions and Table 6.3. However, when asked after each gesture, “What would you like to change about this response?”-- clinicians stated “nothing” 16 times. While there was no majority preference for what to change for each gesture, discussing the results for each gesture provides direction for future research.

When addressing the *Up/Down* gesture, clinician statements included: going up and down a distance from the clinician would not be helpful; a desire to move ART without a delay (currently there is a recognition and then a ready phase); and there were concerns about about provider and patient safety, provider acceptance of the gesture technology, and patient requirements to raise ART while lying or sitting in bed. When addressing the *Therapy* gesture, clinician statements included: a preference for therapist only controls by keypad not by gesture, a preference to use the gesture to conduct therapy at multiple heights, a desire to change the gesture, and questions what other gestures would be required to make the table work at different heights -- such as a nurse setting up the table for the patient to eat. When addressing the *Rest* gesture, clinician statements included: a desire for a more consistent response to the gesture, a desire that the response would be similar to the gesture, and a desire to change how the table responded to the gesture. Additionally, there was a concern about the patient being able to learn the *Rest* gesture and a desire for voice control. Each statement for each of the gestures provides opportunities for future research, as they were not addressed within this study.

When asked about the time required by ART to learn a gesture command (trimmed M=2.94, SD=1.18), clinicians stated less than 3 intervals. This will require a learning algorithm that is able to quickly adapt to user input, an area of future research. As well, six of 11 clinicians stated a desire to be able to learn the platform in under two hours. However, five clinicians stated that what they saw would require anywhere from a couple of days up to two weeks. This is disconcerting for a platform desired to be intuitive and in a time sensitive industry.

Clinicians did not have a preference about speaking to ART instead of using the designed NVC platform. Some participants stated that speaking is easier than using hand gestures and citing that the technology is available; however, two participants mentioned that the other sounds in the hospital room could interfere with a verbal interface. This contradiction may provide an opportunity for future research: Where are there spaces in society with multiple stimuli, always or sometimes present, that would affect the success of a verbal interface? How would an NVC platform best serve those areas?

However, there are limitations of the current NVC platform. As mentioned, one limitation is the patient's cognitive and physical characteristics. Another limitation is the available technology. Currently, the Microsoft Kinect sensor used to control the NVC platform requires the user to be a set distance away from ART. As suggested in Phase III, the NVC lighting sequences and sound feedback should be limited to a small number as to not overburden the clinician or patient with too much to learn. This limitation also supports the amount of time the clinicians estimated for the NVC to learn a given gesture and for the clinician to learn the platform. As these results suggest, a majority of the clinicians stated that non-verbal communication is a productive line of research, even stating a need for this type of technology, but it requires more research to determine where in the healthcare setting it would be most effective.

In this study, clinicians stated a strong preference to use the NVC platform (8 of 11). These results support the clinicians' unanimous agreement to use the platform during Phase I, despite not having a preference during Phase II. These results may be the result of a highly developed platform and using the platform in a hospital room. Interestingly, these results are contrary to the System Usability Scale score.

A surprising finding was the clinician's System Usability Scale score (trimmed M=40, SD=14.49). The score rated the platform as "not acceptable" and as an "F"; however, this is counter to what the clinicians stated about the platform. In support of the platform, clinicians stated not only that they preferred the platform and that it was a productive line of research but also they did not prefer a verbal only platform. One possibility for this contradiction is that the number of proposed gestures (7 - *Give, Take, Eat, Rest, Read, Therapy,* and *Up/Down*) and the number of individual NVC actions (13, from phase III) overwhelmed the clinicians. However, in this study clinicians were only presented with two NVC actions and three gestures. The EASY TO USE rating (3.45) for the overall NVC platform may provide a clue about the low SUS score but in this study clinicians were not asked to explain the given ratings. Future studies should seek to answer why clinicians had these preferences.

6.7 Future Directions

How might an NVC platform continue to support life-long rehabilitation? While this question was not answered directly, it appears that an NVC platform must be a part of the patient's life before an illness, such as a stroke, occurs. Aging with the NVC platform (integrated into ART) already in place allows high-functioning users to communicate with assistive robots in his or her everyday life, without the stress of learning the platform and coping with the onset of the illness. Thus, the platform should be intuitive. An examples can be seen in the results of phase III and the usability study whereby the clinicians were unable to intuitively use the platform without training. Future studies should seek to expand the iterative design process throughout the duration of the study and should attempt to reduce the number of NVC actions and gestures required to adequately address the platform.

Additionally, the results of phase I provided information about where the NVC platform would be best utilized in the rehabilitation setting. Phase II extended these results by providing information about the required patient characteristics to use the NVC platform. Furthermore, the results of the usability study provided future research questions about the gesture command interface. These questions also provide the opportunity for a cross-disciplinary research effort. The research elaborated here benefits from a cross-disciplinary team of architects, engineers, and human factors researchers by which a complex system like the NVC platform has been designed. In the future, researchers should synthesize theses requirements to determine how to better implement an NVC platform in the rehabilitation setting.

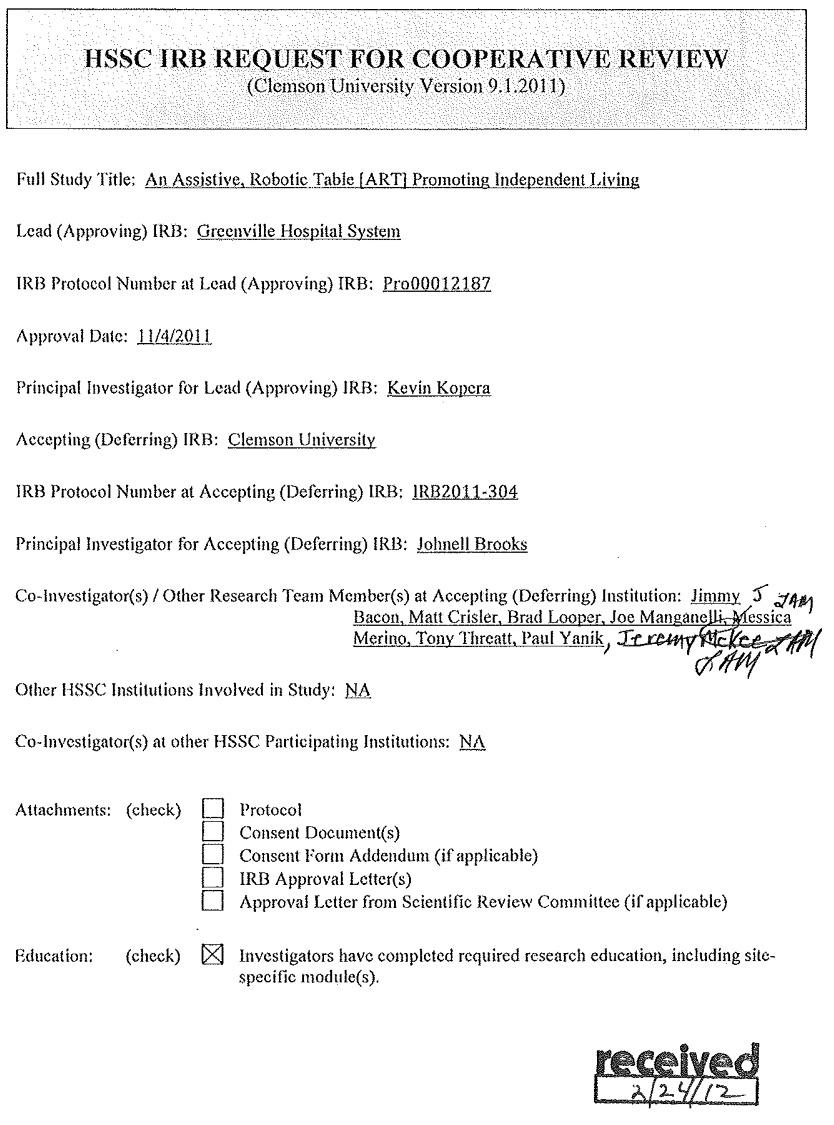
Finally, from the results in phase I and in the rehabilitation hospital setting, the NVC platform should focused on the clinicians and their use of patient care terminology. This may require the NVC to adapt to its surroundings, applying certain features at certain times. As human to human conversation is subjective and/or contextually based so should an NVC platform conform to its surroundings. Future research should investigate how an NVC might change from the home to the rehabilitation setting and back to the home to adequately facilitate life-long rehabilitation.

Non-verbal communication requires further research to determine its place in the healthcare industry and outside it, in our everyday lives. This research has explored NVC as a part of the Assistive Robotic Table in the area of stroke rehabilitation. Arguably, the NVC platform presented might work very well outside the clinical setting for non-clinical users interacting with and supported by assistive robots in everyday life. As we come to interact with a broad-range of Assistive Robotics in our everyday lives -- whether at work, school, home, or the hospital -- NVC platforms must be intuitive, integral with the robot, and adaptable as they promise to be an important mode of human-machine communication.

APPENDICES

Appendix A

Institutional review board approval letter



HSSC IRB REQUEST FOR COOPERATIVE REVIEW
(Clemson University Version 9.1.2011)

Full Study Title: An Assistive, Robotic Table [ART] Promoting Independent Living

Lead (Approving) IRB: Greenville Hospital System

IRB Protocol Number at Lead (Approving) IRB: Pro00012187

Approval Date: 11/4/2011

Principal Investigator for Lead (Approving) IRB: Kevin Kopera

Accepting (Deferring) IRB: Clemson University

IRB Protocol Number at Accepting (Deferring) IRB: IRB2011-304

Principal Investigator for Accepting (Deferring) IRB: Johnell Brooks

Co-Investigator(s) / Other Research Team Member(s) at Accepting (Deferring) Institution: Jimmy J SAM Bacon, Matt Crisler, Brad Looper, Joe Manganelli, Jessica Merino, Tony Threatt, Paul Yanik, ~~Jeremy McKee~~ SAM SAM

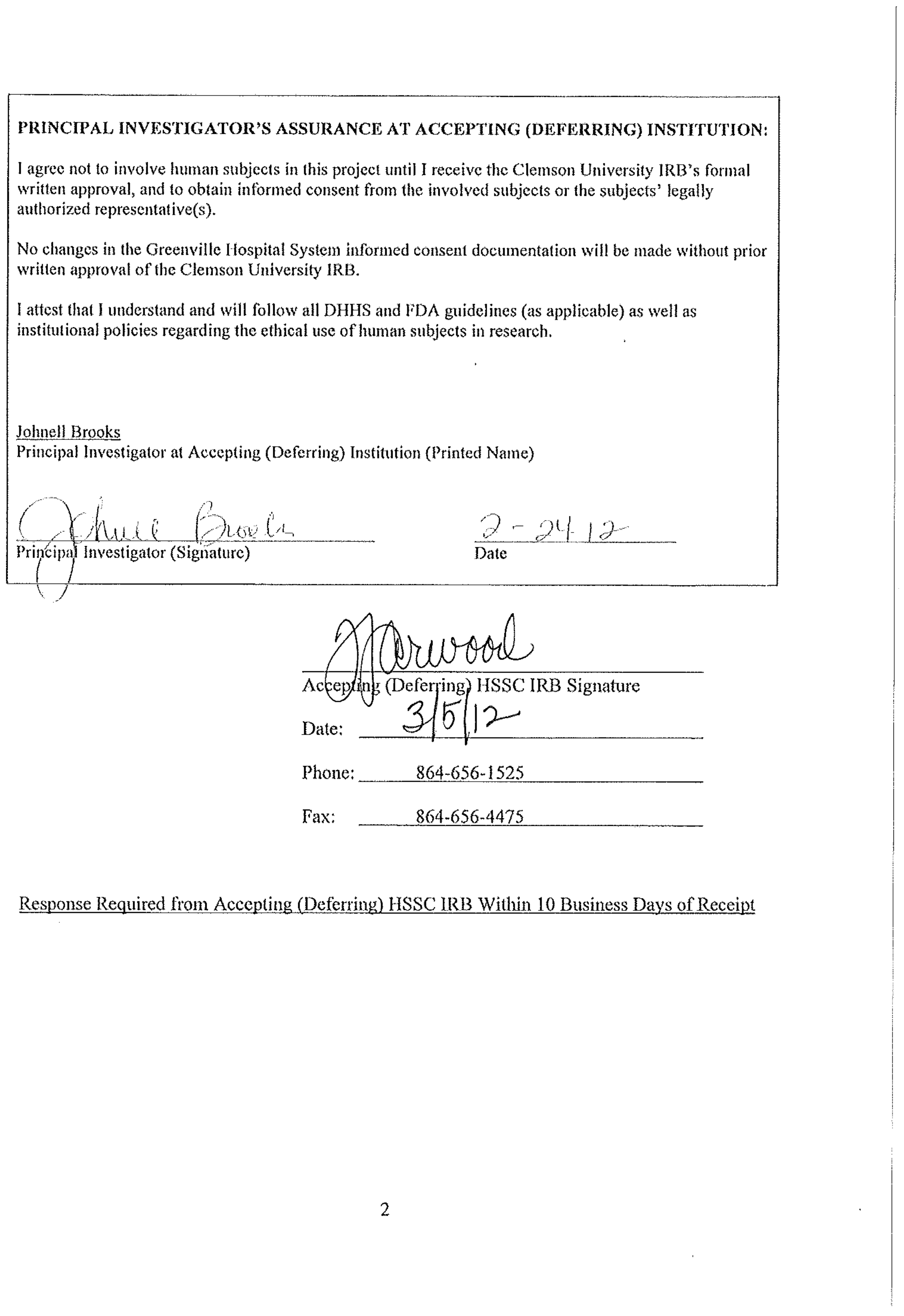
Other HSSC Institutions Involved in Study: NA

Co-Investigator(s) at other HSSC Participating Institutions: NA

Attachments:	(check)	☐	Protocol
		☐	Consent Document(s)
		☐	Consent Form Addendum (if applicable)
		☐	IRB Approval Letter(s)
		☐	Approval Letter from Scientific Review Committee (if applicable)
Education:	(check)	☒	Investigators have completed required research education, including site-specific module(s).

received 2/24/12

1



PRINCIPAL INVESTIGATOR'S ASSURANCE AT ACCEPTING (DEFERRING) INSTITUTION:

I agree not to involve human subjects in this project until I receive the Clemson University IRB's formal written approval, and to obtain informed consent from the involved subjects or the subjects' legally authorized representative(s).

No changes in the Greenville Hospital System informed consent documentation will be made without prior written approval of the Clemson University IRB.

I attest that I understand and will follow all DHHS and FDA guidelines (as applicable) as well as institutional policies regarding the ethical use of human subjects in research.

Johnell Brooks
Principal Investigator at Accepting (Deferring) Institution (Printed Name)

[illegible] 2-24-12
Principal Investigator (Signature) Date

[illegible]
Accepting (Deferring) HSSC IRB Signature

Date: 3/5/12

Phone: 864-656-1525

Fax: 864-656-4475

Response Required from Accepting (Deferring) HSSC IRB Within 10 Business Days of Receipt

2

Appendix B

Informational letter

Nightstand Study Information Sheet– October 28, 2011 IRB File #00012187
Page 1 of 1

RESEARCH STUDY INFORMATION SHEET

An Assistive, Robotic Table [ART] Promoting Independent Living

Study to be Conducted at:	*Roger C. Peace Rehabilitation Hospital* *701 Grove Road* *Greenville, SC 29605*
Sponsor Name:	*National Science Foundation*
Principal Investigator:	*Kevin Kopera, MD 864-455-6262*

INTRODUCTION
You are being asked to participate in a research study. The Institutional Review Board of the Greenville Hospital System has reviewed this study for the protection of the rights of human participants in research studies, in accordance with federal and state regulations. However, before you choose to be a research participant, it is important that you read the following information and ask as many questions as necessary to be sure that you understand what your participation will involve.

PURPOSE AND PROCEDURES
The purpose of this research is to design and build better furniture for hospitals. You are being asked to participate in a study approximately once a month for an hour per session. Your participation will involve:

- Being interviewed in a mock hospital room where you will answer questions regarding how you interact with patients and the room's equipment and furnishings
- Complete a card-sort where you will organize features and ideas written onto individual cards into groups
- Provide your feedback on new furniture designs.

POSSIBLE RISKS AND BENEFITS
There are no known risks associated with this research. No personally identifiable information will be recorded or used as part of this research study. There are no known benefits to you that would result from your participation in the study. This research may help design better furniture.

VOLUNTARY PARTICIPATION
Participation in this study is completely voluntary (your choice). You may refuse to participate or withdraw from the study at any time. If you refuse to participate or withdraw from the study, you will not be penalized or lose any benefits. Your decision will not affect your relationship with the Greenville Hospital System.

CONTACT FOR QUESTIONS
For more information concerning this study and research-related risks or injuries, or to give comments or express concerns or complaints, you may contact the principal investigator, Kevin Kopera, 864-455-6262. You may also contact a representative of the Institutional Review Board of the Greenville Hospital System for information regarding your rights as a participant involved in a research study or to give comments or express concerns, complaints or offer input. You may obtain the name and number of this person by calling (864) 455-8997.

A survey about your experience with this informed consent process is located at the following website:
www.ghs.org/research
Participation in the survey is completely anonymous and voluntary and will not affect your relationship with the Greenville Hospital System. If you would like to have a paper copy of this survey, please tell the principal investigator.

Greenville Hospital System
IRB Number: Pro00012187
Approved: 10/19/2012
Expiration: 10/18/2013

Appendix C

Pre-study sound and light sequences with lab members

	Group	Sound	Light
Up	1	wooop	▨
	2	brrrruuupp - (going up - voice command) increase in pitch - low to high	series of lights (display) / lights around the table top (green)
	3	woooooop - wuuuuu	two or three for lights - light it up as the table goes up - so that the patient knows where the table is on the track
	4	oooooooooo up in pitch - vrrrrrrrr softer but in pitch - woooop wooop wooop wooop	green lights should cycle up in 1/10 of a sec like an animation
Down	1	wuuuuu	▨
	2	bird - burrrrrrr - high to low	(green positive - flashing yellow in motion - red flash when first stopped and then green) Square display with leds - leds draw the picture of what is happening - color could all be the same color ** this goes for all motions unless otherwise noted
	3	wuuuuu - decreasing tone	same as up - with an indicator - like a thermometer
	4	woooooooong decreasing - quuuiiiisssh (hydraulics) - sounds that the chairs make when you let go - in GHS	red lights same as previous
Come	1	breathing out sound/sucking air in - woooosh	▨
	2	wooooiiish - friction sound	
	3	grrrruuuup - like a fart	Natural mapping way ... pattern of a cross up/down - left/right
	4	mmmmmmm - humming yoga sound	yellow flashing - bedazzled the front of this on the edge
Go	1	**opposite of come	▨
	2	spooosh - friction sound	
	3	ding once it's fully retracted - metal cling	
	4	swoooosh	red (undoing what you just did) flashing

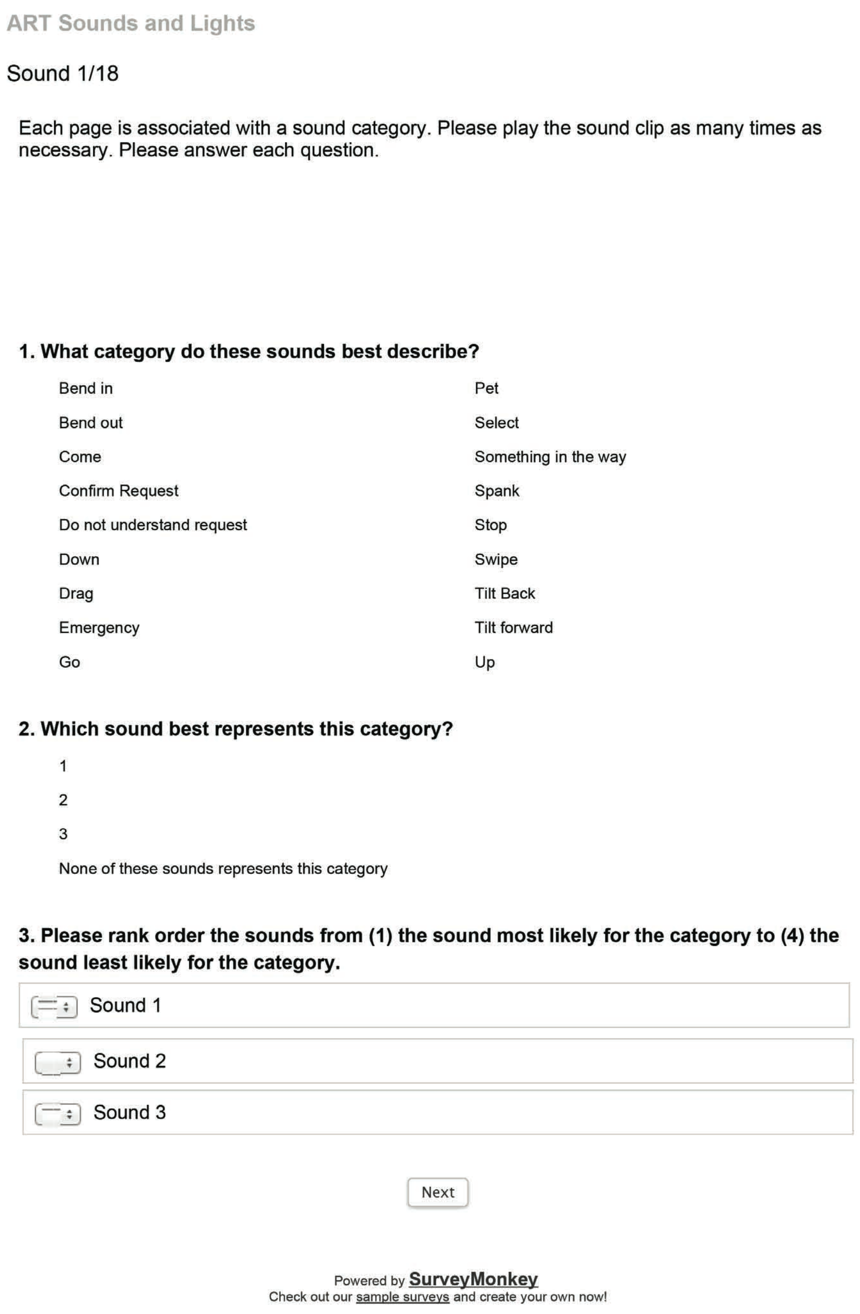
	Group	Sound	Light
Stop	1	click - terminal sound, for everything that exists / limit (of action)	▨
	2	bonk - negative sound	
	3	car breaking - frrrrrr might be too harsh (mild sound for stopping - friction - rub something and come to a stop)	turn all the lights off - or dim the lights low light intensity
	4	beep - one single beep	very pale white lights on each corner / flash red lights around the table top - square light flash red
Tilt forward	1	mmmmmm - higher pitch	▨
	2	burrrrzzzzppp - actuator sound	
	3	sounds should be distinct or like table raise - bzzzzz	pattern of a forward thing (animation - needs to be intuitive)
	4	womp womp womp - in increasing pitch (Mario one up) / ratching of a roller coaster - chick chick chick chick	yellow solid color when tilting / rim of a white light around the surface - soft white light
Tilt back	1	mmmmm - lower pitch	▨
	2	same as forward	
	3	bzzzzz - tone high to low	Pattern of lights for tilting back (animation - needs to be intuitive)
	4	womp womp womp womp - down in pitch / pssssss - bedeet bedeeet (right before closing - closing) psssss	red solid when tilting / white light would cut off
Bend out	1	combination of tilting sound w/ breathing out	▨
	2	wooo wooo wooo - wooooish - low/high pitch	
	3	(continuous deeeeee) tone up	add a degree for where the patient is at
	4	ocean makes waves hitting - woooo wuuush / vibration vvvvvvv like a blood pressure cuff	lt blue just cycles in the arm - can it glow lt blue
Bend in	1	combination of tilting sound w/ breathing in) -mec	▨
	2	(shooooo) high/low pitch	
	3	(same as out) tone down	
	4	wuuuuuuu / air going out sound - feel and hear	goes to drk blue

	Group	Sound	Light
Emergency	1	get out of the way (backing up or interrupted urp urp urp) spills versus physical emergencies, major/minor or levels (issues is time) urgent need has multiples - get out way is just get out of way Clinicians do they agree/ disagree - what else would they add?	
	2	(beep beep beep - fire alarm bell)	(flashing red lights)
	3	high pitched beep - dozzzz dozzz dozzz - like the fire alarm) (if something is spilled is different - different pitch or volume - dooo dooo lower dB)	lots of flashing lights in big case (frequency changes) (red is only for emergency case - moving is green - red is a strong emotion)
	4	alarm clock - mrrrp mrrrp mrrrp hey come clean me - HEY COME HERE NOW same sound but louder	yellow / red / white flashing together anywhere there is lights - same lights - the sound will change per emergency
Confirm request	1	tink - mmmhmm	
	2	(beep beep - positive)	(green) (2 quick flashing lights
	3	(like a chime - ding - dung dung dung - chord) - just do a ding	make a smile
	4	ba da dun - an ok / down tube for mario da da da - complete mario sound track - one up	green - green and blue lights / quick cycle
Do not understand request	1	woooomp	
	2	boooop	Yellow - do not understand Red - flash can't do Flash boop boop (color doesn't matter)
	3	(dun dun - quick) two separate things - make it sound like a question - question mark - dun dun (second have a higher pitch than the first)	a sad face - or a question mark
	4	urrrrrr - like a dog / aant aant	red and yellow / orange with a quick cycle to go with the noise
	1	cat purr	
Pet	2	purrrrr	What's a happy color? (affirmative color - green) Would need to use color in some way?

	Group	Sound	Light
Pet	3	(joyful sound - doo oooop - sounds like a smile - clapping sound)	a smile
	4	purrr / there is no sound for that - pop	fade white - to green - to white
Repriman d	1	womp wooooomp	
	2	burp burp - sad/low tone	Red Light in sequence with the beeps
	3	(oooooooo - bomp bomp - (decreasing tone) i know i'm not doing good)	a sad face if you've done something wrong
	4	facebook pop / aaant	build to red from white
Somethin g in the way	1	emergency	
	2	same noise as emergency - variable noise beeep beeep - slower beep	Yellow - warning Flashing yellow - indicating a hazard
	3	boo boo - very short	two short flash - rest for a while - two short flash (yellow)
	4	beep beep / aaaaaaaant i can't go there is something in the way	cycle the rainbow / flash red around
Swipe	1	wooosh	
	2	swooosh	
	3	shooooo	
	4	swipe from an iphone - swoooosh / zip up sound zzzzziiiip	
Drag	1	** none provided	
	2	feel the click - rougher - buzz or screech	
	3	woooo	
	4	dadadada / i want a buzz but nothing during - want a sound when you touch and untouch it - soft pop / same sound for contact uncontact	
Select	1	ding	
	2	telephone dial - or a beep	
	3	ca tah - like the mouse	
	4	button click - click - like the game of trouble middle button / check	
Can't do			
I'm thinking			

Appendix D

Pre-study sample survey page with lab members



ART Sounds and Lights

Sound 1/18

Each page is associated with a sound category. Please play the sound clip as many times as necessary. Please answer each question.

1. What category do these sounds best describe?

Bend in	Pet
Bend out	Select
Come	Something in the way
Confirm Request	Spank
Do not understand request	Stop
Down	Swipe
Drag	Tilt Back
Emergency	Tilt forward
Go	Up

2. Which sound best represents this category?

1

2

3

None of these sounds represents this category

3. Please rank order the sounds from (1) the sound most likely for the category to (4) the sound least likely for the category.

Sound 1

Sound 2

Sound 3

Next

Powered by SurveyMonkey
Check out our sample surveys and create your own now!

Appendix E

Independent Variables

Phase I			
Sound			Light
Bend in	Drag	Reprimand	Individual LEDs
Bend out	Emergency	Stop	LED Screen
Can't do	Go	Swipe	
Come	I'm thinking	Tilt Back	
Confirm Request	Pet	Tilt forward	
Do not understand request	Select	Up	
Down	Something in the way		

Table 3.1: Presented are the independent variables tested during Phase I. Each sound had two sounds associated with it, were tested audibly, and the clinicians selected their preference between the two. The clinicians chose between the lighting options and used the chosen option to place the options on ART.

Phase II			
Sound		**Light**	
Button	**Gesture**	**Button**	**Gesture**
Up	Come	Up	Come
Down	Go	Down	Go
Tilt forward	Stop	Tilt forward	Stop
Tilt Back	Confirm Request	Tilt Back	Confirm Request
Emergency	Do not understand request	Emergency	Do not understand request
Pet	Can't do	Pet	Can't do
Reprimand	I'm thinking	Reprimand	I'm thinking
	Something in the way	Bend Out	Something in the way
		Bend In	

Table 3.2: Presented are the independent variables tested during Phase II. Each sound and lighting sequence tested had two options, were either button or gesture actuated, and the clinicians chose which sound or lighting sequence best represented each NVC action.

Phase III		
Sound	**Light**	**Sound And Light**
Button	**Button**	**Button**
Down	Reprimand	Tilt Forward
	Bend In	Tilt Back
Gesture	**Gesture**	**Gesture**
Stop	Emergency	Go
Can't Do	I'm thinking	Do not understand request
	Come	Something in the way

Table 3.3: Presented are the independent variables tested during Phase III. Without prior training, the clinicians listened to or viewed the NVC action, chose what action they thought it was, and, if necessary, stated whether the sound, lighting sequence, or both together best demonstrated the presented action.

Appendix F

Phase I: Script

Introduction:
Thank you all for coming. Please read the research study information sheet and let me know if you have any questions.

Today we are going to look at two features – lights and sounds - that we'd like to add to the Assistive Robotic Table. This line of research comes from social robotics and linguistics. It is exploratory and will help us as we continue to develop the communication aspect in future research projects. We'd like your answers today to focus on a specific population - stroke patients. The middle 50%. Are there any questions?

Before we begin I'll ask you some questions concerning communication and stroke patients as it relates to ART.
Tony will ask the questions – Kylie, Joe, Jeremy will record answers per participant.

If ART had the ability to "communicate," would you communicate with it?

What would you want ART to communicate?
What information should ART communicate to patients?
What information should ART communicate to clinicians?
What information would you communicate to ART?
What type of information would be communicated by lights?
What type of information would be communicated by sounds?
What type of information would be communicated by lights and sounds?

How would you want ART to communicate with you?
How would you communicate with ART?

In your experience with stroke patients, what would be the difficulties of this type of communication system and stroke patients?

If ART had the ability to "communicate," would you communicate with it?

Now we'll look at some prototype sound configurations for the different movements ART may utilize. But, before we begin…

Johnell had difficulties distinguishing between the sounds created due to hearing loss – so because of this - Do you know if you have any hearing loss?

I'll demonstrate each action first and then I will go through each action and its corresponding sound individually.

Forced choice A/B sounds. Tony will play each sound group – the participants will respond their answers to recorders.

Finally we'd like to look at a prototype lighting configuration. The lighting in this scenario is used to communicate different movements ART may take. This is not reading or task lighting that might be applied to ART.

Forced choice A/B lights. Tony will play a lighting animation – the participants will respond their answers to recorders.

On the sheet provided, please draw where you would locate the lights.

What would you customize about the lighting configuration you chose?

I have two final questions for you.

If ART had the ability to "communicate" the way we proposed, would you use our system?
Do you think stroke patients would use the system we proposed?
**What would you say the difference is between the communication system you proposed and the one we presented?

I thank you for participating today. Between now and May we will be conducting regular interactions with you. Our research team will be in touch with you soon to schedule your next interaction. I thank you again for participating. Have a great day.

Appendix G

Phase I: Data Template

Participant X		[redacted]	Participant X
Open ended questions			
If ART had the ability to "communicate," would you communicate with it?		[redacted]	
		[redacted]	
What would you want ART to communicate?		[redacted]	
What information should ART communicate to patients?		[redacted]	
What information should ART communicate to clinicians?		[redacted]	
What information would you communicate to ART?		[redacted]	
What type of information would be communicated by lights?		[redacted]	
What type of information would be communicated by sounds?		[redacted]	
What type of information would be communicated by lights and sounds?		[redacted]	
		[redacted]	
How would you want ART to communicate with you?		[redacted]	
How would you communicate with ART?		[redacted]	
		[redacted]	
In your experience with stroke patients, what would be the difficulties of this type of communication system and stroke patients?		[redacted]	
		[redacted]	
If ART had the ability to "communicate," would you communicate with it?		[redacted]	

Forced Choice : Sounds

Up		[redacted]	
Down		[redacted]	
Come		[redacted]	
Go		[redacted]	
Stop		[redacted]	
Tilt forward		[redacted]	
Tilt back		[redacted]	
Bend out		[redacted]	
Bend in		[redacted]	
Emergency		[redacted]	
Confirm request		[redacted]	
Do not understand request		[redacted]	
Pet		[redacted]	
Reprimand		[redacted]	
Can't do		[redacted]	
I'm thinking		[redacted]	
Something in the way		[redacted]	
Swipe		[redacted]	
Drag		[redacted]	
Select		[redacted]	

Forced Choice : Lights

Lighting display		[redacted]	
What would you customize about the lighting configuration you chose?			

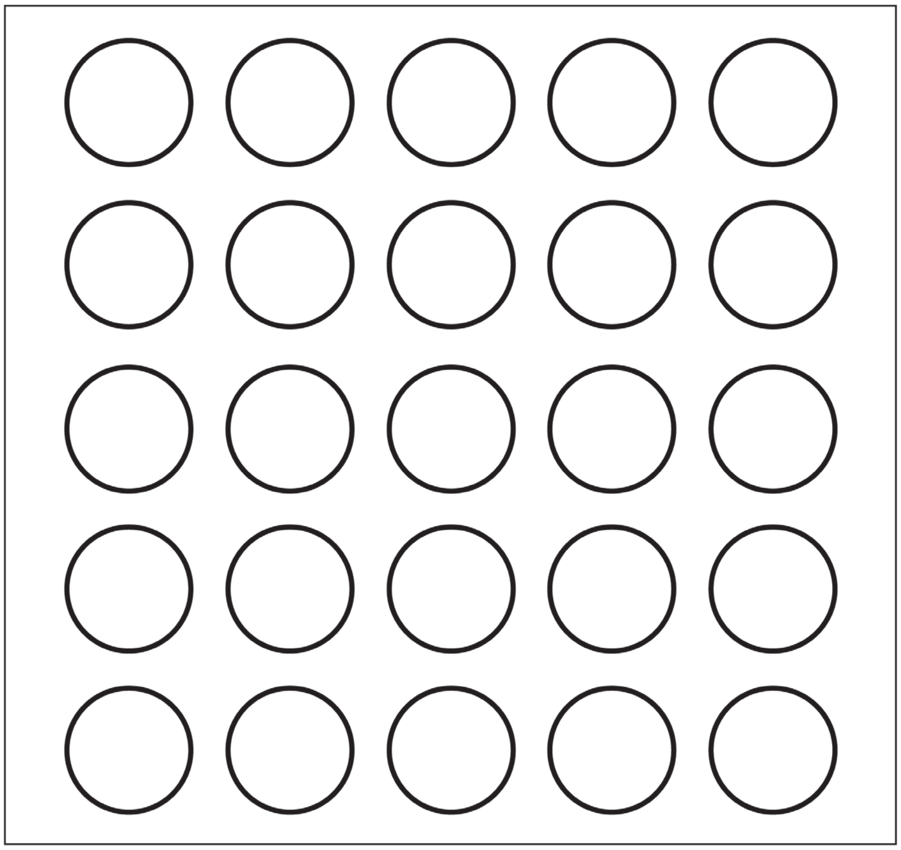
Open ended questions

If ART had the ability to "communicate" the way we proposed, would you use our system?		[redacted]	
Do you think stroke patients would use the system we proposed?		[redacted]	

**What would you say the difference is between the communication system you proposed and the one we presented?		[redacted]	

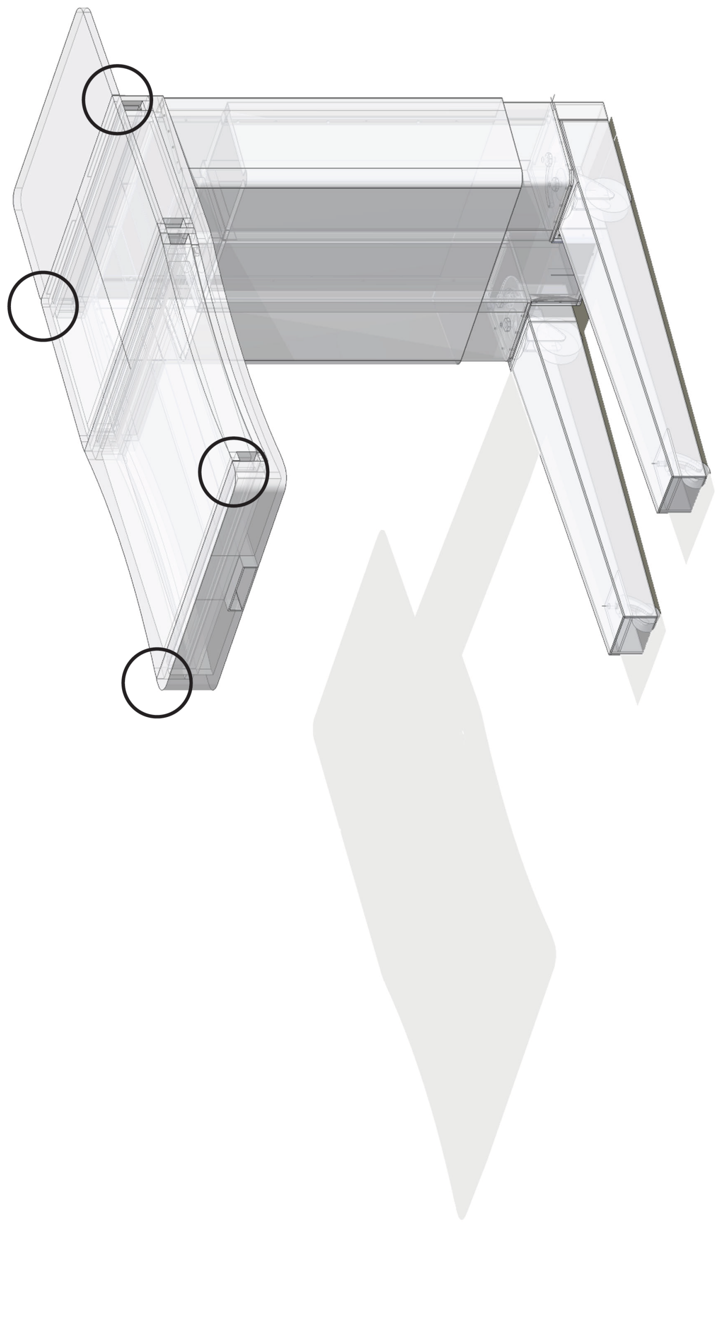
Appendix H

Phase I: Light Panel



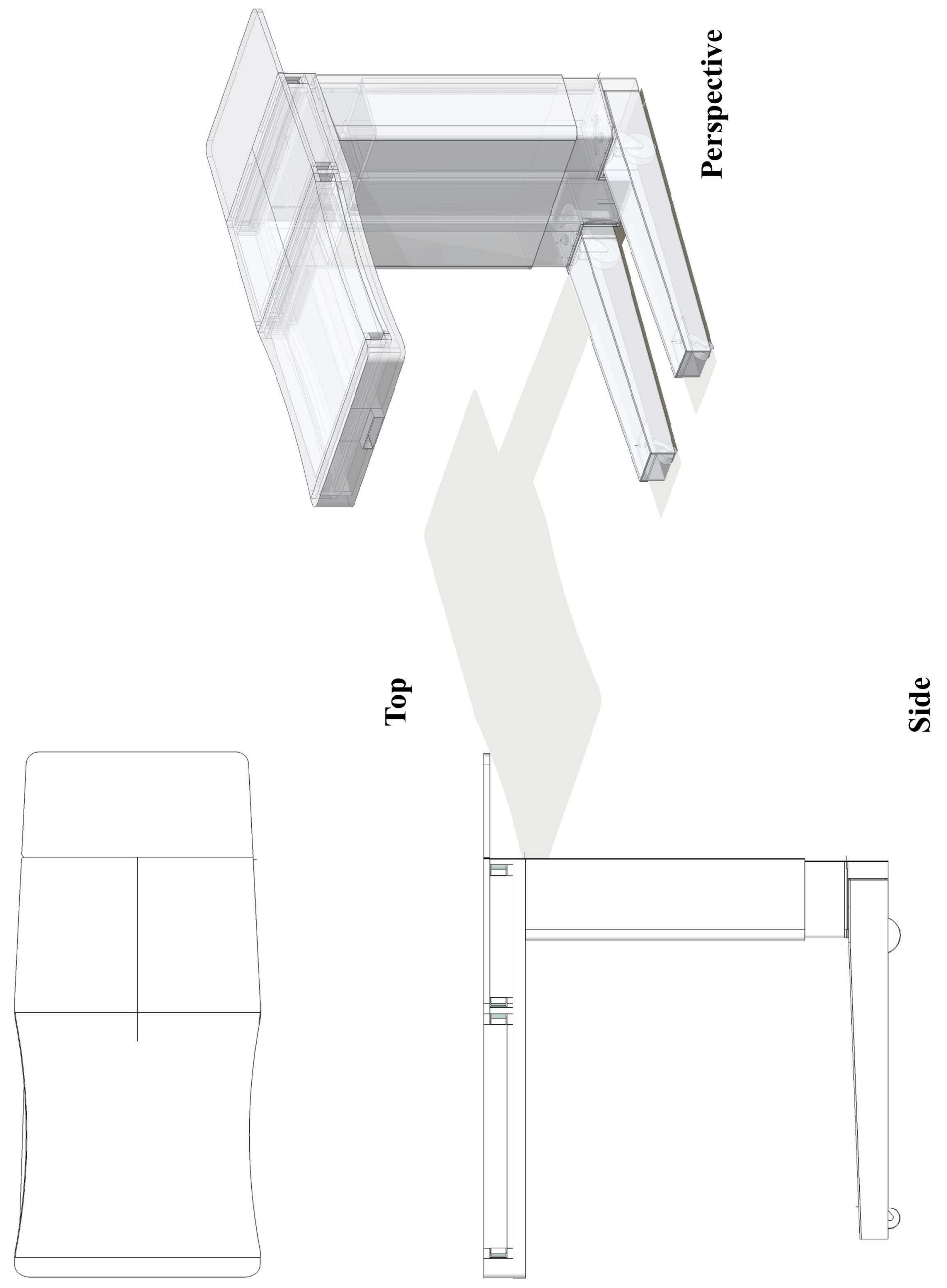
Appendix I

Phase I: Distributed Lights



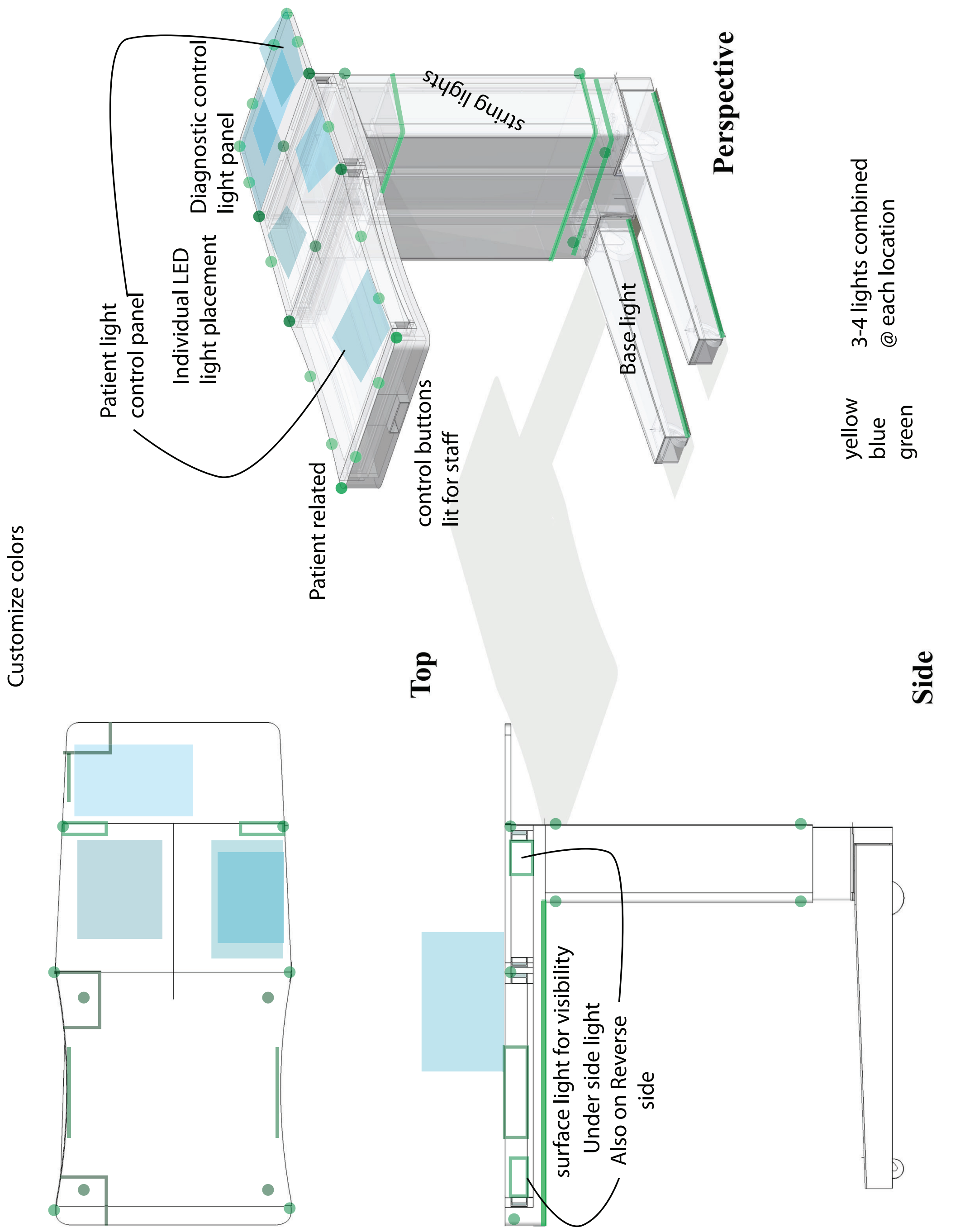
Appendix J

Phase I: Lighting Data Collection Sheet



Appendix K

Phase I: Light Heatmap



Perspective
Diagnostic control light panel
Individual LED light placement
Patient light control panel
string lights
Base light
Patient related
control buttons lit for staff
3-4 lights combined @ each location
yellow
blue
green
Customize colors
Top
Side
surface light for visibility
Under side light
Also on Reverse side

Appendix L

Phase II: Script

Thank you for participating in our next phase of the Assistive Robotic Table study. In front of you will see the research study information sheet please review it and let me know when you are finished. Do you have any questions?

Answer any questions

Today you will be evaluating the nonverbal communication feature of the Assistive Robotic Table. You will evaluate the sounds and the lights that make up the communication system.

Part I

In the first part you will evaluate the sounds. I want you to evaluate the sounds based on our medium persona, Ginny.

Hand over the persona.

For each sound played, please tell us either A, B or Neither. If you choose Neither - please tell us why and what the sound should be. Do you have any questions?

There are two types of communication with the sounds - either gesture or button based. The gesture that I make is arbitrary but it shows you the possibility of the system. We intend in the future to have a vocabulary of movements. Due to the nature of our system, I want you to be aware that there will be some down time in between sounds as Jessica loads each program.

The first type is __________. The first light is ______. Are you ready to begin.

Part II

In the second part you will evaluate the lights. I want you to evaluate the sounds based on our medium persona, Ginny.

For each sound played, please tell us either A, B or Neither. If you choose Neither - please tell us why and what the sound should be. Do you have any questions?

There are two types of communication with the sounds - either gesture or button based.

The first type is __________. The first light is ______. Are you ready to begin.

Appendix M

Phase II: Sample Data Collection Template

Participant X	
Forced Choice : Sounds	
Button	
Up	
Down	
Tilt forward	
Tilt back	
Emergency	
Pet	
Reprimand	
Gesture	
Come	
Go	
Stop	
Confirm request	
Do not understand request	
Can't do	
I'm thinking	
Something in the way	
Open Ended : Sounds	
If ART could "communicate" the way we proposed, would you use the sounds?	
Are the proposed sounds appropriate for the hospital setting? Why?	
Are the proposed sounds communicating what you would expect them to communicate? Why?	

Are the sounds proposed easy to use?	
Are the sounds proposed easy to understand?	
When would the sounds be used?	
What would you want to customize about the sounds?	
Do you think stroke patients would use the system we proposed?	
Could you please describe the characteristics of a patient who might use the sounds?	

Forced Choice : Lights

Gestures

Come	
Go	
Stop	
Confirm request	
Do not understand request	
Can't do	
I'm thinking	
Something in the way	

Button

Up	
Down	
Tilt forward	
Tilt back	
Emergency	
Pet	

Reprimand	
Bend out	
Bend in	

Open Ended : Lights

If ART could "communicate" the way we proposed, would you use the lights?	
Are the proposed lights appropriate for the hospital setting? Why?	
Are the proposed lights communicating what you would expect them to communicate? Why?	
Are the lights proposed easy to use?	
Are the lights proposed easy to understand?	
When would the lights be used?	
What would you want to customize about the lights?	
Do you think stroke patients would use the system we proposed?	
Could you please describe the characteristics of a patient who might use the lights?	
If there were ambient monitoring in the room, where would it be placed?	
If there were ambient monitoring on the table, where would it be placed to best serve you as a clinician?	

Appendix N

Phase II: Final sound and light sequence pairings

Button			Gesture		
	Sound	**Lighting Sequence**		**Sound**	**Lighting Sequence**
Bend In	■	^A	Can't Do	B	■
Bend Out	■	■	Come	■	^B
Down	A	■	Confirm Request	■	■
Emergency	■	A	Do not understand request	B	B
Pet	■	■	Go	A	^B
Reprimand	■	B	I'm Thinking	■	A
Tilt Back	A	^B	Something in the Way	B	A
Tilt Forward	A	^B	Stop	A	■
Up	■	■			

Table 4.2: Presented are the clinicians' sound and light sequence preferences[3] during phase II and tested during phase III. The carrots show specific actions that had a 58% majority and 33% or less than second preference. The intention of the researchers was to test a complete platform.

[3]A full description of the phase II sounds and light sequence results can be viewed here (link is available as of 7/26/2013).
http://www.youtube.com/watch?v=kR2JMkhJBJw&feature=share&list=UUwzcR9h7R6m5B_zQ5E2AHog

Appendix O

Phase III: Script

Thank you for participating in our next phase of the Assistive Robotic Table study. In front of you will see the research study information sheet please review it and let me know when you are finished. Do you have any questions?

Answer any questions

Today you will be evaluating the final phase of the nonverbal communication feature of the Assistive Robotic Table. There will be three parts to today's session. During the first part you will confirm the sounds and the lights assigned to each ART communication action. During the second part you will complete a System Usability Scale Survey. Finally, we will go over your answers and discuss them more in-depth. Do you have any questions?

Part I

In the first part you will confirm the sounds, lights or sound and lights assigned to each of the 13 ART communication actions - either a button or a gesture may actuate each action. You may use this sheet that shows you all of the possible actions. You may write on the sheet if that will help you in your decision process. I will tell you what the actuation method is and if the action has a sound, light, or a sound and light. For instance, if this were action one I may say a button actuates it and it only has a sound. After you tell me what the action is, I will ask you a few follow up questions. Reprimand is a negative reprimand. Do you have any questions about part I?

Begin part I.

Part II

Now that we have completed part I. Please fill out the System Usability Scale Survey. Since you did not interact with the system, I want you to answer the survey based on my interaction with the system and your impressions of the system demonstrated to you.

Hand out SUS.
Collect SUS.

Part III

Thank you for completing the SUS. Now we will look at your answers from Part I and discuss them.

Tell them – how many they answered incorrectly, what actions they missed, demonstrate the action, and ask why?

If given a period of training, do you think that this communication method is appropriate for this action?

Closing

Thank you for participating in today's session. We will be contacting you soon about next month's session. We ask that you not speak with your colleagues about today's session as it may invalidate our results. Do you have any questions?

Answer any questions

Thanks again for participating. Have a great day.

Appendix P

Phase III: Sample Data Collection Template

Participant X		
Forced Choice		Explanation of Answers
GL		Come
What is ART communicating?		
Why?		
BA		Tilt Back
What is ART communicating?		
Which mode best describes what ART is communicating?		
Why?		
GA		Something in the way
What is ART communicating?		
Which mode best describes what ART is communicating?		
Why?		
GL		Emergency
What is ART communicating?		
Why?		
BL		Reprimand
What is ART communicating?		
Why?		

GL	I'm thinking	
What is ART communicating?		
Why?		
GS	Can't Do	
What is ART communicating?		
Why?		
BS	Down	
What is ART communicating?		
Why?		
BL	Bend In	
What is ART communicating?		
Why?		
GA	Go	
What is ART communicating?		
Which mode best describes what ART is communicating?		
Why?		
BA	Tilt Forward	
What is ART communicating?		
Which mode best describes what ART is communicating?		
Why?		
GA	Do not understand request	
What is ART communicating?		

Which mode best describes what ART is communicating?		
Why?		
GS		Stop
What is ART communicating?		
Why?		

Appendix Q

NVC communication actions

ART Communication	No. Correct	With training	Mode that describes the action	Other suggestions
Emergency	7/10	3/3	▨	Can't do (2)
I'm thinking	6/9	3/3	▨	Come (2)
Something in the way	5/10	5/5	Both (2), Sound (8), Light (0)	Stop (2)
Can't Do	3/10	7/7	▨	Something in the way (3), Do not understand (3)
Down	3/10	7/7	▨	Emergency (2), Tilt Back (2)
Tilt Back	3/10	7/7	Both (3), Sound (7), Light (0)	Down (2), Tilt Forward (2)
Go	2/10	Y: 7/8 N: 1/8	Both (2), Sound (3), Light (5)	Bend In (5)
Stop	2/10	Y: 7/8 N: 1/8	▨	Do not understand request (4), Down (2)
Come	1/10	9/9	▨	Thinking (2), Reprimand (2)
Reprimand	1/10	Y: 5/9 N: 4/9	▨	Go (3), Come (2), Stop (2), Thinking (2)
Tilt Forward	1/10	Y: 8/9 N: 1/9	Both (5), Sound (5), Light (0)	Something in the way (3), Come (2), Tilt Back (2)
Do not understand request	0/9	9/9	Both (5), Sound (3), Light (1)	Something in the way (4), Can't do (2), Reprimand (2)
Bend In	0/10	Y: 6/10 N: 4/10	▨	I'm thinking (3), Go (2), Tilt Forward (2)

Table 4.3: Clinician responses during phase III for each of the NVC actions. Without training, clinicians chose the action presented, described what mode best described the action, and after an explanation of the action was given were asked if given training if they could understand the action.

Appendix R

ART communication action themes frequency content analysis

ART Communicati on	No. Corre ct	Explanation of choice	Participants thoughts after given the correct action
Emergency	7/10	**Needs a sound (2), Fast light flash (3), An alarm (2)**, It is signaling something (1), Light pattern (1), Indicates a problem (1)	**Needs a sound (2), Fast light flash (2)**, Implies a problem (1)
I'm thinking	6/9	**Directional Movement of the lights (5), Pace of lights (4)**, Something will happen over a period of time (1), Light pattern like an hour glass (1), It's like "Dun Dun" (1), I don't know (1)	**The lights looked like ... (2)**, Pace of lights (1), Understand afterwards (1)
Something in the way	5/10	**Sounded like car horn (3), Sounded like stop (2), The sound is cultural intuitive (2)**, It was gesture command (1), Seemed like answer would fit (1), The noise (1)	**Understand afterwards (2)**, I don't know (1), Seems to be more permanent (1), Sounded like a car horn (1), Sounded like "uh huh" (1), The lights didn't help (1),
Can't do	3/10	**Sounded like a mistake (3)**, **The sound (2)**, **Something in the way (2)**, "I don't know what you want" (1), Redo the command (1), There's a problem (1)	**Something in the way (3)**, **Negative indication (2)**, Sounded like a question (2), There's a problem (1)
Down	3/10	**Inflection of sound (4), Similar to expected tilt back sound (2), Only choice left (2), It's a negative sound (2)**	**Out of choices (3), Understand afterwards (3)**, Inflection goes down (1), Reprimand needs a sound (1)
Tilt back	3/10	**Expectations of movement (3), Sounded like it was going down (3)**, Best Choice (1), Light and sound together (1), Light brings attention to movement (1), Sounded like it was coming closer (1)	**Understand afterwards (4)**, Not associated the sound with movement (1), People would associate with actual action (1), Sounded like coming towards you (1), Sound needs to be little higher or lower to distinguish (1)
Go	2/10	**Light Pattern (5), Sounded like moving (3)**, Doesn't fit any other options (1), Sounded like ... (1),Only choice left (1), Not because of sound (1)	**Light pattern doesn't make sense (5), The sound doesn't make sense (3), Sounded like its moving (2), Understand afterwards (2)**, The lights don't make sense (1)

ART Communication	No. Correct	Explanation of choice	Participants thoughts after given the correct action
Stop	2/10	**Sounded like ... (3), Deeper sound (2)**, It was a brief sound (1), Short and halting sound (1), Sounded like backward (1), Only choice left (1)	**Sound needs to be stronger (5), Understand afterward (2)**, Already used stop (1), Could be do not understand(1), Sounded like ... (1), Need for lights (1), Sounded like a movement (1)
Come	1/10	**Only choice left (4)**, **Light pattern (3)**, **Looks like processing (2)**, Wasn't sure of choices (1), Use of a hand gesture (1)	**Random light pattern was not correct (5)**, **Light pattern needs direction (2)**, **Looks like ART will move (2)**, Looks like request (1), Could be "I'm thinking" (1), Faster light pace (1), Participant was confused (1)
Reprimand	1/10	**The lights look like ... (7), Light Pattern (3), Pace of lights (2)**, Submissive light (1), Light pattern is simple (1), Doesn't need an auditory signal (1)	**Perceived lights as a movement (3), Lights were positive, not negative (3), The lights looked like ... (3), I don't understand "Reprimand" (2)**
Tilt Forward	1/10	**There will be movement (4), Alarming (2), The lights look like ... (2)**, It's intuitive (1), Seemed negative (1)	**A different tone needed (3), The lights looked like ... (3), Sounded like ... (3)**, Understand afterwards (1), I don't know (1)
Do not understand request	0/9	**Sounded like ... (5)**, **The lights look like ... (4)**, **It's not an emergency (3)**, **Negative connotation (2)**, Only choice left (1), Something is wrong (1), It hit something (1), It sounded like "I can't" (1)	**Interpreted a stronger command (2)**, **Negative connotation (2)**, Similar to another option (1), Participant had "no idea" (1), Implied something is wrong (1), Interpreted the pace of the sound (1)
Bend In	0/10	**Directional movement of the lights (6)**, **Pace of lights (2)**, Constant movement of lights (1), No sound (1), Only choice left (1)	**Direction of light movement (6)**, **Pace of lights (3)**, Need for different colors (1), Why this pattern of lights? (1), Didn't know bend in (1)

Appendix S

ART communication action themes frequency content analysis

Times mentioned	Explanation of choice	Times mentioned	Participants thoughts after given the correct action
23	Sounded like ...	**15**	Understand afterwards
14	Light pattern	**11**	Sounded like ...
13	The lights look like ...	**8**	The lights looked like ...
11	Directional Movement of the lig	**6**	Direction of light movement
10	Only choice left	**5**	Light pattern doesn’t make sense
8	Pace of lights	**5**	Random light pattern was not correct
4	Inflection of sound	**5**	Sound needs to be stronger
4	There will be movement	**4**	Pace of lights
3	Expectations of movement	**3**	A different tone needed
3	Fast light flash	**3**	Lights were positive, not negative
3	It’s not an emergency	**3**	Out of choices
3	Sounded like a mistake	**3**	Perceived lights as a movement
3	Sounded like car horn	**3**	Something in the way
3	Sounded like it was going down	**3**	The sound doesn’t make sense
3	Sounded like moving	**2**	Fast light flash
2	Alarming	**2**	I don’t know
2	An alarm	**2**	I don’t understand “Reprimand”
2	Deeper sound	**2**	Interpreted a stronger command
2	I don’t know	**2**	Light pattern needs direction
2	It’s a negative sound	**2**	Looks like ART will move
2	Looks like Processing	**2**	Needs a sound
2	Needs a sound	**2**	Negative connotation
2	Negative connotation	**2**	Negative indication

Times mentioned	Explanation of choice	Times mentioned	Participants thoughts after given the correct action
2	Similar to expected tilt back sou	**2**	Sounded like a question
2	Something in the way	**2**	Sounded like its moving
2	Sounded like stop	**1**	Already used stop
2	The sound	**1**	Could be "I'm thinking"
2	The sound is cultural intuitive	**1**	Could be do not understand
1	Best choice	**1**	Didn't know bend in
1	Constant movement of lights	**1**	Faster light pace
1	Doesn't fit any other options	**1**	Implied something is wrong
1	Doesn't need an auditory signal	**1**	Implies a problem
1	Indicates a problem	**1**	Inflection goes down
1	It hit something	**1**	Interpreted the pace of the sound
1	It is signaling something	**1**	Looks like request
1	It was a brief sound	**1**	Need for different colors
1	It was gesture command	**1**	Need for lights
1	It's intuitive	**1**	Not associated the sound with movement
1	It's like "Dun Dun"	**1**	Participant had "no idea"
1	Light and sound together	**1**	People would associate with actual actio
1	Light brings attention to movem	**1**	Seems to be more permanent
1	Light pattern is simple	**1**	Similar to another option
1	Light pattern like an hour glass	**1**	Sound needs to be little higher or lower to distinguish
1	No sound	**1**	Sounded like "uh huh"
1	Not because of sound	**1**	Sounded like a car horn
1	Redo the command	**1**	Sounded like coming towards you
1	Seemed like answer would fit	**1**	Reprimand needs a sound
1	Seemed negative	**1**	The lights didn't help

Times mentioned	Explanation of choice	Times mentioned	Participants thoughts after given the correct action
1	Short and halting sound	**1**	The lights don't make sense
1	Something is wrong	**1**	There's a problem
1	Something will happen over a period of time	**1**	Why this pattern of lights?
1	Sounded like backward		
1	Sounded like "I can't"		
1	Sounded like it was coming closer		
1	Submissive light		
1	The noise		
1	There's a problem		
1	Use of a hand gesture		
1	Wasn't sure of choices		

Appendix T

Patient personas

Low Functioning Patient

Ted is a 71 year old male with hypertension, admitted 1 week ago after suffering a severe ischemic stroke. Ted has no movement in his left arm and has “tunnel vision”.

Medium Functioning Patient

Ginny is a 64 year old female with diabetes, admitted 2 weeks ago after suffering an ischemic stroke. She has no fine motor control in right arm and forgets recent events.

High Functioning Patient

Bob is a 52 year old male with a family history of hypertension, admitted 1 week ago after suffering a mild ischemic stroke. He lacks full fine motor control.

Appendix U

Final usability testing: Script

Thank you for participating in our last phase of the Assistive Robotic Table study. In front of you will see the research study information sheet please review it and let me know when you are finished. Do you have any questions?

Answer any questions

Today you will be evaluating the usability of the Assistive Robotic Table. There will be three parts to today's session. During the first part you will use ART around the bed, during the second part you will use ART around a chair, and finally you use a gesture control system. During each part you will answer questions related to that part. Do you have any questions?

Part I

With ART against the wall please move ART over the bed.

Participant should wheel ART next to bed; press the up button [Emergency lights will go off].

What do you think has occurred?
How do you know?
If you had to pick from this list, what do you think has occurred?

Participant should press the up button again and then wheel ART over the bed. Participant might press the down button [Something in the Way lights].

What do you think has occurred?
How do you know?
If you had to pick from this list, what do you think has occurred?

Please move the flip-up surface in position so the patient can read. The flip-up up surface does not flip-up so tell me when you can go no further.

Participant should push back the top surface and tell you they can go no further.

What are your thoughts about your experience positioning ART over the bed?

Now that we have completed part I, please fill out the System Usability Scale Survey.

Hand out SUS.
Collect SUS.

Part II

Now please move ART in front of the chair to conduct therapy on the patient's left upper extremity. Please also move the therapy surface and mechanical column in place to conduct therapy.

What are your thoughts about your experience positioning ART for therapy?

Now that we have completed part II, please fill out the System Usability Scale Survey.

Hand out SUS.
Collect SUS.

Part II : B
Please move ART and the mechanical column to their original position. You're going to evaluate ART using two scales and while you are doing that Jessica is going to set up for Part III.

Please look at this list and select 5 words to describe ART or your experience with ART.

Now you'll evaluate each feature of ART using the LIKE, NEED and EASY TO USE scales.

Part III

Our final activity is to use and evaluate the gesture interface. The first gesture is up and down. I'll demonstrate it for you, have you practice the gesture and then you'll interact with ART. Are you ready?

Demonstrate the up/down gestures. Have participant practice the gesture and then have them complete the gesture. Have patient sweep hands being their back when they are complete.

1. Did you like ART's response to the gestured command?
2. What did you like about it?
3. What would you like to change about this response?

Please rate the communication system you used today using the LIKE, NEED, and EASY TO USE scale.

Demonstrate the Therapy gesture. Have participant practice the gesture and then have them complete the gesture. Participants were told to imagine that they conducted therapy with the same patient each day and at the same height and would use the gesture to control ART.

1. Did you like ART's response to the gestured command?
2. What did you like about it?
3. What would you like to change about this response?

Please rate the communication system you used today using the LIKE, NEED, and EASY TO USE scale.

5.*Demonstrate the Rest gesture. Have participant practice the gesture and then have them complete the gesture. Participants were told to imagine the patient had finished eating and pushed ART away from the bed and wanted ART at a specific height next to the bed.*

1. Did you like ART's response to the gestured command?
2. What did you like about it?
3. What would you like to change about this response?

Please rate the communication system you used today using the LIKE, NEED, and EASY TO USE scale.

4. If ART did not respond in the manner you would prefer, how might you convey to ART that the response was not satisfactory?
5. If you wanted ART to do more or less of a given response, how might you indicate that desire to ART?
6. Would you prefer that ART be preprogrammed to understand a few common gesture commands, or would you prefer to define your own commands?
7. If you were teaching ART a set of gesture commands, how would you want to convey your satisfaction or dissatisfaction with ART's responses as it learned?

8. How many teaching intervals (response and feedback) would you be willing to go through before you would expect ART to learn a given gesture command?

Now that we have completed part III, please fill out the System Usability Scale Survey.

Hand out SUS.
Collect SUS.

Thank you. We have a few final questions.

If ART had the ability to "communicate" the way we proposed, would you use our system? Why?

In your estimation, how long would it take you to learn how to use the communication platform you used today?

Do you think this is a productive line of research? Why?

Would you have preferred to simply speak to ART and ART speak to you? Why?

Please rate the communication system you used today using the LIKE, NEED, and EASY TO USE scale.

Closing

Thank you for participating in today's session. We ask that you not speak with your colleagues about today's session as it may invalidate our results. Do you have any questions?

Answer any questions

Thanks again for participating this past year. Have a great day.

Appendix V

Final usability testing: Data Collection Template

Items:							
Adjustable Legs		Shape of table top surface		Flip-up table top surface		10. Table Controls	
Like		Like		Like		Like	
Need		Need		Need		Need	
Easy to Use		Easy to Use		Easy to Use		Easy to Use	
2. Up/down mechanism		Lip on table top surface				11. Therapy Surface	
Like		Like				Like	
Need		Need				Need	
Easy to Use		Easy to Use				Easy to Use	
Level table top				9. Cup holders		Mechanical Column for Therapy Surface	
Like				Like		Like	
Need				Need		Need	
Easy to Use				Easy to Use		Easy to Use	

Part I	
What do you think has occurred?	
How do you know?	
If you had to pick from this list, what do you think has occurred?	

What do you think has occurred?	
How do you know?	
If you had to pick from this list, what do you think has occurred?	
What are your thoughts about your experience positioning ART over the bed?	
SUS	
Part II	
What are your thoughts about your experience positioning ART for therapy?	
SUS	
Words	
LIKE / NEED / EASY TO USE	
Up/Down	
Did you like ART’s response to the gestured command?	
What did you like about it?	
What would you like to change about this response?	
Please rate the communication system you used today using the LIKE, NEED, and EASY TO USE scale.	
LIKE	
NEED	
EASY TO USE	
Read	
Did you like ART’s response to the gestured command?	
What did you like about it?	
What would you like to change about this response?	

Please rate the communication system you used today using the LIKE, NEED, and EASY TO USE scale.	
LIKE	
NEED	
EASY TO USE	
Sleep	
Did you like ART's response to the gestured command?	
What did you like about it?	
What would you like to change about this response?	
Please rate the communication system you used today using the LIKE, NEED, and EASY TO USE scale.	
LIKE	
NEED	
EASY TO USE	
If ART did not respond in the manner you would prefer, how might you convey to ART that the response was not satisfactory?	
If you wanted ART to do more or less of a given response, how might you indicate that desire to ART?	
Would you prefer that ART be preprogrammed to understand a few common gesture commands, or would you prefer to define your own commands?	
If you were teaching ART a set of gesture commands, how would you want to convey your satisfaction or dissatisfaction with ART's responses as it learned?	
How many teaching intervals (response and feedback) would you be willing to go through before you would expect ART to learn a given gesture command?	
If ART had the ability to "communicate" the way we proposed, would you use our system? Why?	
Please rate the communication system you used today using the LIKE, NEED, and EASY TO USE scale.	

LIKE	
NEED	
EASY TO USE	

Appendix W

Clinician preferences for gesture command interface

Did you like ART's response to the gestured command?			
	Yes	Maybe	No
Up/Down	10	1	0
Therapy	7	2	2
Rest	10	0	1

Table 6.2: Presented are the clinician preferences for each of the gesture commands. Compared with the other two gesture commands, the clinicians were reticent about accepting the *Therapy* command. They cited the desire to use conventional rehabilitation practices.

Appendix X

Clinician subjective scale ratings for the overall NVC platform and each of the gesture commands

	Up/Down	Therapy	Rest	Overall
Like	4.18	3.82	3.82	3.82
Need	3.73	3.18	3.73	3.82
Easy to Use	4.00	3.64	3.45	3.45

Table 6.3: Presented are the subjective ratings for the overall NVC platform and each of the gesture commands. Similar to the results of Table 6.2, the *Therapy* command scored the lowest for the Need rating.

Appendix Y

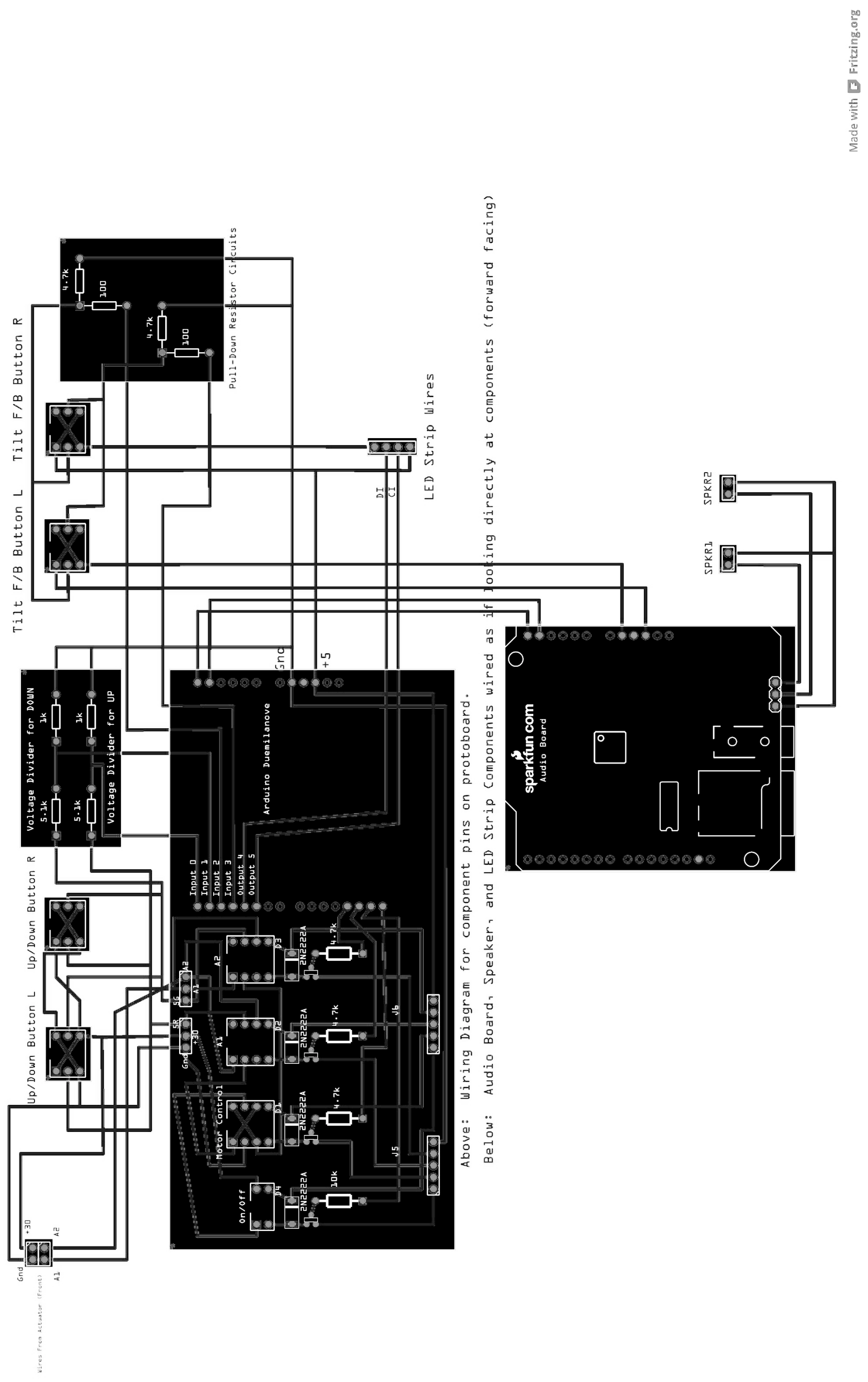
Clinician estimation of time to learn the NVC platform

In your estimation, how long would it take you to learn how to use the communication platform you used today?	
30 Minutes	3
1 Hour	2
1-2 hours	1
A couple of days	1
A few days - consistently	1
1 week - consistently	1
2 weeks	2

Table 6.4: Presented are the clinicians' estimation of time to learn the presented NVC platform. It is disconcerting that the current platform may require up to two weeks to learn. Given the time sensitive nature of healthcare it would be necessary to learn the platform in under one hour.

Appendix Z

NVC electrical diagram



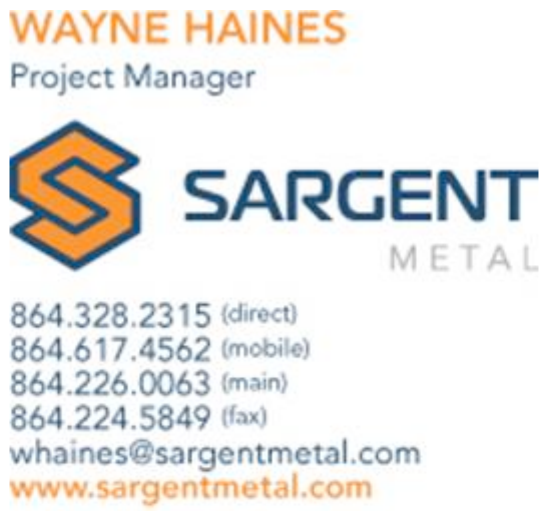
Appendix AA

Steel manufacturer contact information

Sargent Metal

Steps for fabricating with them:

1. If at all possible, provide them shop drawings with an .stp file (Solidworks, Alibre, or Geomagic). This will eliminate the time and cost of having them redraw our work.
2. The first point of contact is Bobby Weir, in Sales (bweir@sargentmetal.com).
3. Once we have an understanding with Bobby, we should email our request to quotes@sargentmetal.com. Our email is then sent to multiple heads at SM.
4. Push for a Point of Order - or "PO" (i.e. a quote) ***ASAP***. Without the PO, our order it is ***NOT*** in the system.
5. Our contact in the fabrication process is Wayne Haynes, an engineer at SM.


WAYNE HAINES
Project Manager

SARGENT
METAL

864.328.2315 (direct)
864.617.4562 (mobile)
864.226.0063 (main)
864.224.5849 (fax)
whaines@sargentmetal.com
www.sargentmetal.com

Appendix BB

Phase II Code: Arduino - Sound

Button: Up & Down

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int output1=6;
int input1=7;
int valDown=0;
int valUp=0;

static int press=0;
int waiting=0;

int dataPin = 2;
int clockPin = 3;

static int countDown=0;
static int countUp=0;

LPD8806 strip = LPD8806(52, dataPin, clockPin);


void setup() {

  pinMode(output1, OUTPUT);
  pinMode(input1,INPUT);
  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();

  Serial.begin(9600);
}

void loop(){

  digitalWrite(output1, HIGH);
  valDown=analogRead(A0);
  valUp=analogRead(A5);

  float voltageDown=valDown*(0.5/1024.0);
  float voltageUp=valUp*(0.5/1024.0);

  //Serial.println(voltageDown);
  //Serial.println(voltageUp);

  while(voltageDown>0.4){
    press++;
    if(press==1){
      if(countDown==0){
        Serial.write(1);
        countDown++;
      }
      else{
        Serial.write(2);
        countDown=0;
      }
    }
    valDown=analogRead(A0);
    voltageDown=valDown*(0.5/1024.0);
    //Serial.println(press);
    //Serial.println(voltageDown);
    //Serial.println(voltageUp);

  }

   //while(voltageDown<0.4 && voltageUp<0.4){
    //waiting++;
    //valDown=analogRead(A0);
    //voltageDown=valDown*(0.5/1024.0);
  //}
  //if(waiting>200){
    //press=0;
    //waiting=0;
  //}

  //Serial.println(press);

    //press=0;

  while(voltageUp>0.4){
    press++;
    if(press==1){
      if(countUp==0){
        Serial.write(3);
        countUp++;
      }
      else{
        Serial.write(4);
        countUp=0;
      }
    }
    valUp=analogRead(A5);
    voltageUp=valUp*(0.5/1024.0);
    //Serial.println(voltageDown);
    //Serial.println(voltageUp);
  }
   while(voltageDown<0.4 && voltageUp<0.4){
    waiting++;
    valUp=analogRead(A5);
    voltageUp=valUp*(0.5/1024.0);
    valDown=analogRead(A0);
    voltageDown=valDown*(0.5/1024.0);
    //Serial.println(voltageDown);
    //Serial.println(voltageUp);
  }
  if(waiting>200){
    press=0;
    waiting=0;
  }
    //press=0;

  }
```

Button: Tilt Forward

```
int valPress = 7;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status

static int count = 0;
static int press=0;
int waiting=0;
static int countPress=0;

void setup() {

  pinMode(valPress, INPUT);

   Serial.begin(9600);
}

void loop(){


  valPress=digitalRead(7);


  while(valPress == HIGH){
    press++;
    if(press==1){
      if(countPress==0){
        Serial.write(1);
        countPress++;
      }
      else{
        Serial.write(2);
        countPress=0;
      }
    }
    valPress=digitalRead(7);

  }

   while(valPress == LOW){
    waiting++;
    valPress=digitalRead(7);
  }
  if(waiting>200){
    press=0;
    waiting=0;
  }
}
```

Button: Tilt Back

```
int valPress = 7;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status

static int count = 0;
static int press=0;
int waiting=0;
static int countPress=0;

void setup() {

  pinMode(valPress, INPUT);

   Serial.begin(9600);
}

void loop(){


  valPress=digitalRead(7);


  while(valPress == HIGH){
    press++;
    if(press==1){
      if(countPress==0){
        Serial.write(1);
        countPress++;
      }
      else{
        Serial.write(2);
        countPress=0;
      }
    }
    valPress=digitalRead(7);

  }

   while(valPress == LOW){
    waiting++;
    valPress=digitalRead(7);
  }
  if(waiting>200){
    press=0;
    waiting=0;
  }
}
```

Button: Emergency

```
int valPress = 7;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status

static int count = 0;
static int press=0;
int waiting=0;
static int countPress=0;

void setup() {
  pinMode(valPress, INPUT);

   Serial.begin(9600);
}

void loop(){


  valPress=digitalRead(7);


  while(valPress == HIGH){
    press++;
    if(press==1){
      if(countPress==0){
        Serial.write(1);
        countPress++;
      }
      else{
        Serial.write(2);
        countPress=0;
      }
    }
    valPress=digitalRead(7);

  }

   while(valPress == LOW){
    waiting++;
    valPress=digitalRead(7);
  }
  if(waiting>200){
    press=0;
    waiting=0;
  }
}
```

Button: Pet

```
int valSoft = A3;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status

static int count = 0;
static int press=0;
int waiting=0;
static int countSoft=0;

void setup() {

  pinMode(valSoft, INPUT);

   Serial.begin(9600);
}

void loop(){


  valSoft=analogRead(A3);


  while(valSoft >= 900){
    press++;
    if(press==1){
      if(countSoft==0){
        Serial.write(1);
        countSoft++;
      }
      else{
        Serial.write(2);
        countSoft=0;
      }
    }
    valSoft=analogRead(A3);

  }

   while(valSoft < 900){
    waiting++;
    valSoft=analogRead(A3);
  }
  if(waiting>200){
    press=0;
    waiting=0;
  }
}
```

Button: Reprimand

```
int valPress = 7;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status

static int count = 0;
static int press=0;
int waiting=0;
static int countPress=0;


void setup() {

   pinMode(valPress, INPUT);

   Serial.begin(9600);
}

void loop(){

  valPress=digitalRead(7);


  while(valPress == HIGH){
    press++;
    if(press==1){
      if(countPress==0){
        Serial.write(1);
        countPress++;
      }
      else{
        Serial.write(2);
        countPress=0;
      }
    }
    valPress=digitalRead(7);
  }

   while(valPress == LOW){
    waiting++;
    valPress=digitalRead(7);
  }
  if(waiting>200){
    press=0;
    waiting=0;
  }
}
```

Appendix CC

Phase II Code: Processing - Sound

Button: Up & Down

```
import SimpleOpenNI.*;
//import java.util.Iterator;
//import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Serial port; // Create object from Serial class
int val=0; // Data received from the serial port
int test=214;

Minim minim;
AudioPlayer downsound1;
AudioPlayer down2;
AudioPlayer up1;
AudioPlayer up2;

void setup()
{

  println(Serial.list()); //This shows the various serial port options
  String portName = Serial.list()[0]; //The serial port should match the one the Arduino is hooked to
  port = new Serial(this, portName, 9600); //Establish the connection rate


  minim = new Minim(this);
  downsound1=minim.loadFile("Down1.wav");
  down2=minim.loadFile("Down2.wav");
  up1=minim.loadFile("Up1.wav");
  up2=minim.loadFile("Up2.wav");


}

void draw()
{

  if (port.available() > 0) {
    val = port.read();
    println(val);
  }

  if(val==1){
    //down2.play();
    //down2.rewind();
    downsound1.play();
    downsound1.rewind();
    println(214);

  }
  else if(val==2){
    down2.play();
    down2.rewind();
  }
  else if(val==3){
    up1.play();
    up1.rewind();
  }
  else if(val==4){
    up2.play();
    up2.rewind();
  }
  val=0;

}
```

Button: Tilt Forward

```
//import SimpleOpenNI.*;
//import java.util.Iterator;
//import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Serial port; // Create object from Serial class
int val=0; // Data received from the serial port
int test=214;

Minim minim;
AudioPlayer tiltforward1;
AudioPlayer tiltforward2;

void setup()
{

  println(Serial.list()); //This shows the various serial port options
  String portName = Serial.list()[0]; //The serial port should match the one the Arduino is hooked to
  port = new Serial(this, portName, 9600); //Establish the connection rate


  minim = new Minim(this);
  tiltforward1=minim.loadFile("TiltForward1.wav");
  tiltforward2=minim.loadFile("TiltForward2.wav");


}

void draw()
{

  if (port.available() > 0) {
    val = port.read();
    println(val);
  }

  if(val==1){
    tiltforward1.play();
    tiltforward1.rewind();

  }
  else if(val==2){
    tiltforward2.play();
    tiltforward2.rewind();
  }

  val=0;

}
```

<u>Button: Tilt Back</u>

```
//import SimpleOpenNI.*;
//import java.util.Iterator;
//import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Serial port; // Create object from Serial class
int val=0; // Data received from the serial port
int test=214;

Minim minim;
AudioPlayer tiltback1;
AudioPlayer tiltback2;

void setup()
{

  println(Serial.list()); //This shows the various serial port options
  String portName = Serial.list()[0]; //The serial port should match the one the Arduino is hooked to
  port = new Serial(this, portName, 9600); //Establish the connection rate


  minim = new Minim(this);
  tiltback1=minim.loadFile("TiltBack1.wav");
  tiltback2=minim.loadFile("TiltBack2.wav");


}

void draw()
{

  if (port.available() > 0) {
    val = port.read();
    println(val);
  }

  if(val==1){
    tiltback1.play();
    tiltback1.rewind();

  }
  else if(val==2){
    tiltback2.play();
    tiltback2.rewind();
  }

  val=0;

}
```

Button: Emergency

```
//import SimpleOpenNI.*;
//import java.util.Iterator;
//import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Serial port; // Create object from Serial class
int val=0; // Data received from the serial port
int test=214;

Minim minim;
AudioPlayer emergency1;
AudioPlayer emergency2;

void setup()
{

  println(Serial.list()); //This shows the various serial port options
  String portName = Serial.list()[0]; //The serial port should match the one the Arduino is hooked to
  port = new Serial(this, portName, 9600); //Establish the connection rate


  minim = new Minim(this);
  emergency1=minim.loadFile("Emergency1.wav");
  emergency2=minim.loadFile("Emergency2.wav");


}

void draw()
{

  if (port.available() > 0) {
    val = port.read();
    println(val);
  }

  if(val==1){
    emergency1.play();
    emergency1.rewind();

  }
  else if(val==2){
    emergency2.play();
    emergency2.rewind();
  }

  val=0;
}
```

Button: Pet

```
//import SimpleOpenNI.*;
//import java.util.Iterator;
//import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Serial port; // Create object from Serial class
int val=0; // Data received from the serial port
int test=214;

Minim minim;
AudioPlayer pet1;
AudioPlayer pet2;

void setup()
{

  println(Serial.list()); //This shows the various serial port options
  String portName = Serial.list()[0]; //The serial port should match the one the Arduino is hooked to
  port = new Serial(this, portName, 9600); //Establish the connection rate


  minim = new Minim(this);
  pet1=minim.loadFile("Pet1.wav");
  pet2=minim.loadFile("Pet2.wav");


}

void draw()
{

  if (port.available() > 0) {
    val = port.read();
    println(val);
  }

  if(val==1){
    pet1.play();
    pet1.rewind();

  }
  else if(val==2){
    pet2.play();
    pet2.rewind();
  }

  val=0;

}
```

Button; Reprimand

```
//import SimpleOpenNI.*;
//import java.util.Iterator;
//import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Serial port; // Create object from Serial class
int val=0; // Data received from the serial port
int test=214;

Minim minim;
AudioPlayer Reprimand1;
AudioPlayer Reprimand2;

void setup()
{
  println(Serial.list()); //This shows the various serial port options
  String portName = Serial.list()[0]; //The serial port should match the one the Arduino is hooked to
  port = new Serial(this, portName, 9600); //Establish the connection rate


  minim = new Minim(this);
  Reprimand1=minim.loadFile("Reprimand1.wav");
  Reprimand2=minim.loadFile("Reprimand2.wav");


}

void draw()
{

  if (port.available() > 0) {
    val = port.read();
    println(val);
  }

  if(val==1){
    Reprimand1.play();
    Reprimand1.rewind();

  }
  else if(val==2){
    Reprimand2.play();
    Reprimand2.rewind();
  }

  val=0;

}
```

Gesture: Come

```
import SimpleOpenNI.*;
import java.util.Iterator;
import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Minim minim;
AudioPlayer sound1;
AudioPlayer sound2;


static int time=0;
int check=0;
float xcoord0, xcoord99, diff;

SimpleOpenNI      context;

// NITE
XnVSessionManager sessionManager;
XnVFlowRouter     flowRouter;

PointDrawer       pointDrawer;

PrintWriter output;
```

```
void setup()
{


  minim = new Minim(this);
  sound1=minim.loadFile("Come1.wav");
  sound2=minim.loadFile("Come2.wav");

  output = createWriter("out.txt");
  context = new SimpleOpenNI(this);

  // mirror is by default enabled
  context.setMirror(true);

  // enable depthMap generation
  if(context.enableDepth() == false)
  {
     println("Can't open the depthMap, maybe the camera is not connected!");
     exit();
     return;
  }

  // enable the hands + gesture
  context.enableGesture();
  context.enableHands();

  // setup NITE
  sessionManager = context.createSessionManager("Click,Wave", "RaiseHand");

  pointDrawer = new PointDrawer();
  flowRouter = new XnVFlowRouter();
  flowRouter.SetActive(pointDrawer);

  sessionManager.AddListener(flowRouter);

  size(context.depthWidth(), context.depthHeight());
  smooth();
}

void draw()
{
  background(200,0,0);
  // update the cam
  context.update();

  // update nite
  context.update(sessionManager);

  // draw depthImageMap
  image(context.depthImage(),0,0);

  // draw the list
  pointDrawer.draw();
}

void keyPressed()
{
  switch(key)
  {
  case 'e':
    // end sessions
    sessionManager.EndSession();
    println("end session");
    break;
  }
}

/////////////////////////////////////////////////////////////////////////////////////////////////
// session callbacks

void onStartSession(PVector pos)
{
  println("onStartSession: " + pos);
}

void onEndSession()
{
  println("onEndSession: ");
}

void onFocusSession(String strFocus,PVector pos,float progress)
{
  println("onFocusSession: focus=" + strFocus + ",pos=" + pos + ",progress=" + progress);
}


/////////////////////////////////////////////////////////////////////////////////////////////////
// PointDrawer keeps track of the handpoints

class PointDrawer extends XnVPointControl
{
  HashMap    _pointLists;
  int        _maxPoints;
  color[]    _colorList = { color(255,0,0),color(0,255,0),color(0,0,255),color(255,255,0)};

  public PointDrawer()
  {
    _maxPoints = 30;
    _pointLists = new HashMap();
  }

  public void OnPointCreate(XnVHandPointContext cxt)
  {
    // create a new list
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));

    println("OnPointCreate, handId: " + cxt.getNID());
  }

  public void OnPointUpdate(XnVHandPointContext cxt)
  {
    //println("OnPointUpdate " + cxt.getPtPosition());
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));
  }

  public void OnPointDestroy(long nID)
  {
    println("OnPointDestroy, handId: " + nID);
```

```
 // remove list
 if(_pointLists.containsKey(nID))
    _pointLists.remove(nID);
}

public ArrayList getPointList(long handId)
{
 ArrayList curList;
 if(_pointLists.containsKey(handId))
   curList = (ArrayList)_pointLists.get(handId);
 else
 {
   curList = new ArrayList(_maxPoints);
   _pointLists.put(handId,curList);
 }
 return curList;
}

public void addPoint(long handId,PVector handPoint)
{
 ArrayList curList = getPointList(handId);

 curList.add(0,handPoint);
 if(curList.size() > _maxPoints)
   curList.remove(curList.size() - 1);


   output.println(handPoint);
   if(handPoint.x>150){
     xcoord0=handPoint.x;
     //output.println("START");
     check=0;
   }
   else if(handPoint.x<-200){

       if(check==0){
       //output.println("TRIGGER");

       if(time==0){
       sound1.play();
       sound1.rewind();
       time=1;
       }
       else{
       sound2.play();
       sound2.rewind();
       time=0;
       }


     }
     check=1;

     }


   //count++;
   //output.println(handPoint);
```

```
  }

  public void draw()
  {
    if(_pointLists.size() <= 0)
      return;

    pushStyle();
      noFill();

      PVector vec;
      PVector firstVec;
      PVector screenPos = new PVector();
      int colorIndex=0;

      // draw the hand lists
      Iterator<Map.Entry> itrList = _pointLists.entrySet().iterator();
      while(itrList.hasNext())
      {
        strokeWeight(2);
        stroke(_colorList[colorIndex % (_colorList.length - 1)]);

        ArrayList curList = (ArrayList)itrList.next().getValue();

        // draw line
        firstVec = null;
        Iterator<PVector> itr = curList.iterator();
        beginShape();
          while (itr.hasNext())
          {
            vec = itr.next();
            if(firstVec == null)
              firstVec = vec;
            // calc the screen pos
            context.convertRealWorldToProjective(vec,screenPos);
            vertex(screenPos.x,screenPos.y);
          }
        endShape();

        // draw current pos of the hand
        if(firstVec != null)
        {
          strokeWeight(8);
          context.convertRealWorldToProjective(firstVec,screenPos);
          point(screenPos.x,screenPos.y);
        }
        colorIndex++;
      }

    popStyle();
  }

}

void stop(){
  minim.stop();
  super.stop();
  output.flush();
  output.close();
}
```

Gesture: Go

```
import SimpleOpenNI.*;
import java.util.Iterator;
import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Minim minim;
AudioPlayer sound1;
AudioPlayer sound2;


static int time=0;
int check=0;
float xcoord0, xcoord99, diff;

SimpleOpenNI      context;

// NITE
XnVSessionManager sessionManager;
XnVFlowRouter     flowRouter;

PointDrawer       pointDrawer;

PrintWriter output;

void setup()
{


  minim = new Minim(this);
  sound1=minim.loadFile("Go1.wav");
  sound2=minim.loadFile("Go2.wav");

  output = createWriter("out.txt");
  context = new SimpleOpenNI(this);

  // mirror is by default enabled
  context.setMirror(true);

  // enable depthMap generation
  if(context.enableDepth() == false)
  {
     println("Can't open the depthMap, maybe the camera is not connected!");
     exit();
     return;
  }

  // enable the hands + gesture
  context.enableGesture();
  context.enableHands();

  // setup NITE
  sessionManager = context.createSessionManager("Click,Wave", "RaiseHand");

  pointDrawer = new PointDrawer();
  flowRouter = new XnVFlowRouter();
  flowRouter.SetActive(pointDrawer);

  sessionManager.AddListener(flowRouter);
  size(context.depthWidth(), context.depthHeight());
  smooth();
}

void draw()
{
  background(200,0,0);
  // update the cam
  context.update();

  // update nite
  context.update(sessionManager);

  // draw depthImageMap
  image(context.depthImage(),0,0);

  // draw the list
  pointDrawer.draw();
}

void keyPressed()
{
  switch(key)
  {
  case 'e':
    // end sessions
    sessionManager.EndSession();
    println("end session");
    break;
  }
}

/////////////////////////////////////////////////////////////////////////////////////////////////
// session callbacks

void onStartSession(PVector pos)
{
  println("onStartSession: " + pos);
}

void onEndSession()
{
  println("onEndSession: ");
}

void onFocusSession(String strFocus,PVector pos,float progress)
{
  println("onFocusSession: focus=" + strFocus + ",pos=" + pos + ",progress=" + progress);
}


/////////////////////////////////////////////////////////////////////////////////////////////////
// PointDrawer keeps track of the handpoints

class PointDrawer extends XnVPointControl
{
  HashMap    _pointLists;
  int        _maxPoints;
  color[]    _colorList = { color(255,0,0),color(0,255,0),color(0,0,255),color(255,255,0)};
  public PointDrawer()
  {
    _maxPoints = 30;
    _pointLists = new HashMap();
  }

  public void OnPointCreate(XnVHandPointContext cxt)
  {
    // create a new list
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));

    println("OnPointCreate, handId: " + cxt.getNID());
  }

  public void OnPointUpdate(XnVHandPointContext cxt)
  {
    //println("OnPointUpdate " + cxt.getPtPosition());
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));
  }

  public void OnPointDestroy(long nID)
  {
    println("OnPointDestroy, handId: " + nID);

    // remove list
    if(_pointLists.containsKey(nID))
       _pointLists.remove(nID);
  }

  public ArrayList getPointList(long handId)
  {
    ArrayList curList;
    if(_pointLists.containsKey(handId))
      curList = (ArrayList)_pointLists.get(handId);
    else
    {
      curList = new ArrayList(_maxPoints);
      _pointLists.put(handId,curList);
    }
    return curList;
  }

  public void addPoint(long handId,PVector handPoint)
  {
    ArrayList curList = getPointList(handId);

    curList.add(0,handPoint);
    if(curList.size() > _maxPoints)
      curList.remove(curList.size() - 1);


      output.println(handPoint);
      if(handPoint.x>150){
        xcoord0=handPoint.x;
        //output.println("START");
        check=0;
      }
      else if(handPoint.x<-200){
        if(check==0){
        //output.println("TRIGGER");

        if(time==0){
        sound1.play();
        sound1.rewind();
        time=1;
        }
        else{
        sound2.play();
        sound2.rewind();
        time=0;
        }


      }
      check=1;

      }


    //count++;
    //output.println(handPoint);



}

public void draw()
{
  if(_pointLists.size() <= 0)
    return;

  pushStyle();
    noFill();

    PVector vec;
    PVector firstVec;
    PVector screenPos = new PVector();
    int colorIndex=0;

    // draw the hand lists
    Iterator<Map.Entry> itrList = _pointLists.entrySet().iterator();
    while(itrList.hasNext())
    {
      strokeWeight(2);
      stroke(_colorList[colorIndex % (_colorList.length - 1)]);

      ArrayList curList = (ArrayList)itrList.next().getValue();

      // draw line
      firstVec = null;
      Iterator<PVector> itr = curList.iterator();
      beginShape();
        while (itr.hasNext())
        {
          vec = itr.next();
          if(firstVec == null)
            firstVec = vec;
          // calc the screen pos
          context.convertRealWorldToProjective(vec,screenPos);
```

```
            vertex(screenPos.x,screenPos.y);
          }
        endShape();

        // draw current pos of the hand
        if(firstVec != null)
        {
          strokeWeight(8);
          context.convertRealWorldToProjective(firstVec,screenPos);
          point(screenPos.x,screenPos.y);
        }
        colorIndex++;
      }

    popStyle();
  }

}

void stop(){
  minim.stop();
  super.stop();
  output.flush();
  output.close();
}
```

Gesture: Stop

```
import SimpleOpenNI.*;
import java.util.Iterator;
import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Minim minim;
AudioPlayer sound1;
AudioPlayer sound2;


static int time=0;
int check=0;
float xcoord0, xcoord99, diff;

SimpleOpenNI      context;

// NITE
XnVSessionManager sessionManager;
XnVFlowRouter     flowRouter;

PointDrawer       pointDrawer;

PrintWriter output;

void setup()
{


  minim = new Minim(this);
  sound1=minim.loadFile("Stop1.wav");
  sound2=minim.loadFile("Stop2.wav");
```

```
 output = createWriter("out.txt");
 context = new SimpleOpenNI(this);

 // mirror is by default enabled
 context.setMirror(true);

 // enable depthMap generation
 if(context.enableDepth() == false)
 {
   println("Can't open the depthMap, maybe the camera is not connected!");
   exit();
   return;
 }

 // enable the hands + gesture
 context.enableGesture();
 context.enableHands();

 // setup NITE
 sessionManager = context.createSessionManager("Click,Wave", "RaiseHand");

 pointDrawer = new PointDrawer();
 flowRouter = new XnVFlowRouter();
 flowRouter.SetActive(pointDrawer);

 sessionManager.AddListener(flowRouter);

 size(context.depthWidth(), context.depthHeight());
 smooth();
}

void draw()
{
 background(200,0,0);
 // update the cam
 context.update();

 // update nite
 context.update(sessionManager);

 // draw depthImageMap
 image(context.depthImage(),0,0);

 // draw the list
 pointDrawer.draw();
}

void keyPressed()
{
 switch(key)
 {
 case 'e':
   // end sessions
   sessionManager.EndSession();
   println("end session");
   break;
 }
}

/////////////////////////////////////////////////////////////////////////////////////////////////
// session callbacks
```

```
void onStartSession(PVector pos)
{
  println("onStartSession: " + pos);
}

void onEndSession()
{
  println("onEndSession: ");
}

void onFocusSession(String strFocus,PVector pos,float progress)
{
  println("onFocusSession: focus=" + strFocus + ",pos=" + pos + ",progress=" + progress);
}


/////////////////////////////////////////////////////////////////////////////////////////////////
// PointDrawer keeps track of the handpoints

class PointDrawer extends XnVPointControl
{
  HashMap    _pointLists;
  int        _maxPoints;
  color[]    _colorList = { color(255,0,0),color(0,255,0),color(0,0,255),color(255,255,0)};

  public PointDrawer()
  {
    _maxPoints = 30;
    _pointLists = new HashMap();
  }

  public void OnPointCreate(XnVHandPointContext cxt)
  {
    // create a new list
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));

    println("OnPointCreate, handId: " + cxt.getNID());
  }

  public void OnPointUpdate(XnVHandPointContext cxt)
  {
    //println("OnPointUpdate " + cxt.getPtPosition());
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));
  }

  public void OnPointDestroy(long nID)
  {
    println("OnPointDestroy, handId: " + nID);

    // remove list
    if(_pointLists.containsKey(nID))
       _pointLists.remove(nID);
  }

  public ArrayList getPointList(long handId)
  {
    ArrayList curList;
    if(_pointLists.containsKey(handId))
      curList = (ArrayList)_pointLists.get(handId);
    else
    {
      curList = new ArrayList(_maxPoints);
      _pointLists.put(handId,curList);
    }
    return curList;
  }

  public void addPoint(long handId,PVector handPoint)
  {
    ArrayList curList = getPointList(handId);

    curList.add(0,handPoint);
    if(curList.size() > _maxPoints)
      curList.remove(curList.size() - 1);


      output.println(handPoint);
      if(handPoint.x>150){
        xcoord0=handPoint.x;
        //output.println("START");
        check=0;
      }
      else if(handPoint.x<-200){

          if(check==0){
          //output.println("TRIGGER");

          if(time==0){
          sound1.play();
          sound1.rewind();
          time=1;
          }
          else{
          sound2.play();
          sound2.rewind();
          time=0;
          }


        }
        check=1;

        }


      //count++;
      //output.println(handPoint);



  }

  public void draw()
  {
    if(_pointLists.size() <= 0)
      return;

    pushStyle();
      noFill();
```

```
      PVector vec;
      PVector firstVec;
      PVector screenPos = new PVector();
      int colorIndex=0;

      // draw the hand lists
      Iterator<Map.Entry> itrList = _pointLists.entrySet().iterator();
      while(itrList.hasNext())
      {
        strokeWeight(2);
        stroke(_colorList[colorIndex % (_colorList.length - 1)]);

        ArrayList curList = (ArrayList)itrList.next().getValue();

        // draw line
        firstVec = null;
        Iterator<PVector> itr = curList.iterator();
        beginShape();
          while (itr.hasNext())
          {
            vec = itr.next();
            if(firstVec == null)
              firstVec = vec;
            // calc the screen pos
            context.convertRealWorldToProjective(vec,screenPos);
            vertex(screenPos.x,screenPos.y);
          }
        endShape();

        // draw current pos of the hand
        if(firstVec != null)
        {
          strokeWeight(8);
          context.convertRealWorldToProjective(firstVec,screenPos);
          point(screenPos.x,screenPos.y);
        }
        colorIndex++;
      }

    popStyle();
  }

}

void stop(){
  minim.stop();
  super.stop();
  output.flush();
  output.close();
}
```

Gesture: Confirm Request

```
import SimpleOpenNI.*;
import java.util.Iterator;
import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Minim minim;
```

```
AudioPlayer sound1;
AudioPlayer sound2;


static int time=0;
int check=0;
float xcoord0, xcoord99, diff;

SimpleOpenNI      context;

// NITE
XnVSessionManager sessionManager;
XnVFlowRouter     flowRouter;

PointDrawer       pointDrawer;

PrintWriter output;

void setup()
{


  minim = new Minim(this);
  sound1=minim.loadFile("ConfirmRequest1.wav");
  sound2=minim.loadFile("ConfirmRequest2.wav");

  output = createWriter("out.txt");
  context = new SimpleOpenNI(this);

  // mirror is by default enabled
  context.setMirror(true);

  // enable depthMap generation
  if(context.enableDepth() == false)
  {
     println("Can't open the depthMap, maybe the camera is not connected!");
     exit();
     return;
  }

  // enable the hands + gesture
  context.enableGesture();
  context.enableHands();

  // setup NITE
  sessionManager = context.createSessionManager("Click,Wave", "RaiseHand");

  pointDrawer = new PointDrawer();
  flowRouter = new XnVFlowRouter();
  flowRouter.SetActive(pointDrawer);

  sessionManager.AddListener(flowRouter);

  size(context.depthWidth(), context.depthHeight());
  smooth();
}

void draw()
{
  background(200,0,0);
  // update the cam
```

```
 context.update();

 // update nite
 context.update(sessionManager);

 // draw depthImageMap
 image(context.depthImage(),0,0);

 // draw the list
 pointDrawer.draw();
}

void keyPressed()
{
 switch(key)
 {
 case 'e':
   // end sessions
   sessionManager.EndSession();
   println("end session");
   break;
 }
}

/////////////////////////////////////////////////////////////////////////////////////////////////
// session callbacks

void onStartSession(PVector pos)
{
 println("onStartSession: " + pos);
}

void onEndSession()
{
 println("onEndSession: ");
}

void onFocusSession(String strFocus,PVector pos,float progress)
{
 println("onFocusSession: focus=" + strFocus + ",pos=" + pos + ",progress=" + progress);
}


/////////////////////////////////////////////////////////////////////////////////////////////////
// PointDrawer keeps track of the handpoints

class PointDrawer extends XnVPointControl
{
 HashMap    _pointLists;
 int        _maxPoints;
 color[]    _colorList = { color(255,0,0),color(0,255,0),color(0,0,255),color(255,255,0)};

 public PointDrawer()
 {
   _maxPoints = 30;
   _pointLists = new HashMap();
 }

 public void OnPointCreate(XnVHandPointContext cxt)
 {
   // create a new list
```

```
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));

    println("OnPointCreate, handId: " + cxt.getNID());
  }

  public void OnPointUpdate(XnVHandPointContext cxt)
  {
    //println("OnPointUpdate " + cxt.getPtPosition());
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));
  }

  public void OnPointDestroy(long nID)
  {
    println("OnPointDestroy, handId: " + nID);

    // remove list
    if(_pointLists.containsKey(nID))
       _pointLists.remove(nID);
  }

  public ArrayList getPointList(long handId)
  {
    ArrayList curList;
    if(_pointLists.containsKey(handId))
      curList = (ArrayList)_pointLists.get(handId);
    else
    {
      curList = new ArrayList(_maxPoints);
      _pointLists.put(handId,curList);
    }
    return curList;
  }

  public void addPoint(long handId,PVector handPoint)
  {
    ArrayList curList = getPointList(handId);

    curList.add(0,handPoint);
    if(curList.size() > _maxPoints)
      curList.remove(curList.size() - 1);


      output.println(handPoint);
      if(handPoint.x>150){
        xcoord0=handPoint.x;
        //output.println("START");
        check=0;
      }
      else if(handPoint.x<-200){

          if(check==0){
          //output.println("TRIGGER");

          if(time==0){
          sound1.play();
          sound1.rewind();
          time=1;
          }
          else{
```

```
        sound2.play();
        sound2.rewind();
        time=0;
        }


      }
      check=1;

      }


    //count++;
    //output.println(handPoint);



}

public void draw()
{
  if(_pointLists.size() <= 0)
    return;

  pushStyle();
    noFill();

    PVector vec;
    PVector firstVec;
    PVector screenPos = new PVector();
    int colorIndex=0;

    // draw the hand lists
    Iterator<Map.Entry> itrList = _pointLists.entrySet().iterator();
    while(itrList.hasNext())
    {
      strokeWeight(2);
      stroke(_colorList[colorIndex % (_colorList.length - 1)]);

      ArrayList curList = (ArrayList)itrList.next().getValue();

      // draw line
      firstVec = null;
      Iterator<PVector> itr = curList.iterator();
      beginShape();
        while (itr.hasNext())
        {
          vec = itr.next();
          if(firstVec == null)
            firstVec = vec;
          // calc the screen pos
          context.convertRealWorldToProjective(vec,screenPos);
          vertex(screenPos.x,screenPos.y);
        }
      endShape();

      // draw current pos of the hand
      if(firstVec != null)
      {
        strokeWeight(8);
        context.convertRealWorldToProjective(firstVec,screenPos);
        point(screenPos.x,screenPos.y);
      }
      colorIndex++;
    }

   popStyle();
  }

}

void stop(){
  minim.stop();
  super.stop();
  output.flush();
  output.close();
}
```

Gesture: Do Not Understand Request

```
import SimpleOpenNI.*;
import java.util.Iterator;
import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Minim minim;
AudioPlayer sound1;
AudioPlayer sound2;


static int time=0;
int check=0;
float xcoord0, xcoord99, diff;

SimpleOpenNI      context;

// NITE
XnVSessionManager sessionManager;
XnVFlowRouter     flowRouter;

PointDrawer       pointDrawer;

PrintWriter output;

void setup()
{


  minim = new Minim(this);
  sound1=minim.loadFile("DoNotUnderstand1.wav");
  sound2=minim.loadFile("DoNotUnderstand2.wav");

  output = createWriter("out.txt");
  context = new SimpleOpenNI(this);

  // mirror is by default enabled
  context.setMirror(true);

  // enable depthMap generation
  if(context.enableDepth() == false)
  {
    println("Can't open the depthMap, maybe the camera is not connected!");
    exit();
    return;
  }

  // enable the hands + gesture
  context.enableGesture();
  context.enableHands();

  // setup NITE
  sessionManager = context.createSessionManager("Click,Wave", "RaiseHand");

  pointDrawer = new PointDrawer();
  flowRouter = new XnVFlowRouter();
  flowRouter.SetActive(pointDrawer);

  sessionManager.AddListener(flowRouter);

  size(context.depthWidth(), context.depthHeight());
  smooth();
}

void draw()
{
  background(200,0,0);
  // update the cam
  context.update();

  // update nite
  context.update(sessionManager);

  // draw depthImageMap
  image(context.depthImage(),0,0);

  // draw the list
  pointDrawer.draw();
}

void keyPressed()
{
  switch(key)
  {
  case 'e':
    // end sessions
    sessionManager.EndSession();
    println("end session");
    break;
  }
}

/////////////////////////////////////////////////////////////////////////////////////////////////
// session callbacks

void onStartSession(PVector pos)
{
  println("onStartSession: " + pos);
}

void onEndSession()
{
  println("onEndSession: ");
```

```
}

void onFocusSession(String strFocus,PVector pos,float progress)
{
  println("onFocusSession: focus=" + strFocus + ",pos=" + pos + ",progress=" + progress);
}


/////////////////////////////////////////////////////////////////////////////////////////////////////
// PointDrawer keeps track of the handpoints

class PointDrawer extends XnVPointControl
{
  HashMap    _pointLists;
  int        _maxPoints;
  color[]    _colorList = { color(255,0,0),color(0,255,0),color(0,0,255),color(255,255,0)};

  public PointDrawer()
  {
    _maxPoints = 30;
    _pointLists = new HashMap();
  }

  public void OnPointCreate(XnVHandPointContext cxt)
  {
    // create a new list
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));

    println("OnPointCreate, handId: " + cxt.getNID());
  }

  public void OnPointUpdate(XnVHandPointContext cxt)
  {
    //println("OnPointUpdate " + cxt.getPtPosition());
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));
  }

  public void OnPointDestroy(long nID)
  {
    println("OnPointDestroy, handId: " + nID);

    // remove list
    if(_pointLists.containsKey(nID))
       _pointLists.remove(nID);
  }

  public ArrayList getPointList(long handId)
  {
    ArrayList curList;
    if(_pointLists.containsKey(handId))
      curList = (ArrayList)_pointLists.get(handId);
    else
    {
      curList = new ArrayList(_maxPoints);
      _pointLists.put(handId,curList);
    }
    return curList;
  }
```

```
public void addPoint(long handId,PVector handPoint)
{
 ArrayList curList = getPointList(handId);

 curList.add(0,handPoint);
 if(curList.size() > _maxPoints)
   curList.remove(curList.size() - 1);


   output.println(handPoint);
   if(handPoint.x>150){
     xcoord0=handPoint.x;
     //output.println("START");
     check=0;
   }
   else if(handPoint.x<-200){

       if(check==0){
       //output.println("TRIGGER");

       if(time==0){
       sound1.play();
       sound1.rewind();
       time=1;
       }
       else{
       sound2.play();
       sound2.rewind();
       time=0;
       }


     }
     check=1;

     }


   //count++;
   //output.println(handPoint);



}

public void draw()
{
 if(_pointLists.size() <= 0)
   return;

 pushStyle();
   noFill();

   PVector vec;
   PVector firstVec;
   PVector screenPos = new PVector();
   int colorIndex=0;

   // draw the hand lists
   Iterator<Map.Entry> itrList = _pointLists.entrySet().iterator();
   while(itrList.hasNext())
```

```
    {
     strokeWeight(2);
     stroke(_colorList[colorIndex % (_colorList.length - 1)]);

     ArrayList curList = (ArrayList)itrList.next().getValue();

     // draw line
     firstVec = null;
     Iterator<PVector> itr = curList.iterator();
     beginShape();
       while (itr.hasNext())
       {
         vec = itr.next();
         if(firstVec == null)
           firstVec = vec;
         // calc the screen pos
         context.convertRealWorldToProjective(vec,screenPos);
         vertex(screenPos.x,screenPos.y);
       }
     endShape();

     // draw current pos of the hand
     if(firstVec != null)
     {
       strokeWeight(8);
       context.convertRealWorldToProjective(firstVec,screenPos);
       point(screenPos.x,screenPos.y);
     }
     colorIndex++;
    }

   popStyle();
  }

}

void stop(){
  minim.stop();
  super.stop();
  output.flush();
  output.close();
}
```

<u>Gesture: Can't Do</u>

```
import SimpleOpenNI.*;
import java.util.Iterator;
import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Minim minim;
AudioPlayer sound1;
AudioPlayer sound2;


static int time=0;
int check=0;
float xcoord0, xcoord99, diff;
```

```
SimpleOpenNI      context;

// NITE
XnVSessionManager sessionManager;
XnVFlowRouter     flowRouter;

PointDrawer       pointDrawer;

PrintWriter output;

void setup()
{


  minim = new Minim(this);
  sound1=minim.loadFile("CantDo1.wav");
  sound2=minim.loadFile("CantDo2.wav");

  output = createWriter("out.txt");
  context = new SimpleOpenNI(this);

  // mirror is by default enabled
  context.setMirror(true);

  // enable depthMap generation
  if(context.enableDepth() == false)
  {
     println("Can't open the depthMap, maybe the camera is not connected!");
     exit();
     return;
  }

  // enable the hands + gesture
  context.enableGesture();
  context.enableHands();

  // setup NITE
  sessionManager = context.createSessionManager("Click,Wave", "RaiseHand");

  pointDrawer = new PointDrawer();
  flowRouter = new XnVFlowRouter();
  flowRouter.SetActive(pointDrawer);

  sessionManager.AddListener(flowRouter);

  size(context.depthWidth(), context.depthHeight());
  smooth();
}

void draw()
{
  background(200,0,0);
  // update the cam
  context.update();

  // update nite
  context.update(sessionManager);

  // draw depthImageMap
  image(context.depthImage(),0,0);
  // draw the list
  pointDrawer.draw();
}

void keyPressed()
{
  switch(key)
  {
  case 'e':
    // end sessions
    sessionManager.EndSession();
    println("end session");
    break;
  }
}

/////////////////////////////////////////////////////////////////////////////////////////////////
// session callbacks

void onStartSession(PVector pos)
{
  println("onStartSession: " + pos);
}

void onEndSession()
{
  println("onEndSession: ");
}

void onFocusSession(String strFocus,PVector pos,float progress)
{
  println("onFocusSession: focus=" + strFocus + ",pos=" + pos + ",progress=" + progress);
}


/////////////////////////////////////////////////////////////////////////////////////////////////
// PointDrawer keeps track of the handpoints

class PointDrawer extends XnVPointControl
{
  HashMap    _pointLists;
  int        _maxPoints;
  color[]    _colorList = { color(255,0,0),color(0,255,0),color(0,0,255),color(255,255,0)};

  public PointDrawer()
  {
    _maxPoints = 30;
    _pointLists = new HashMap();
  }

  public void OnPointCreate(XnVHandPointContext cxt)
  {
    // create a new list
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));

    println("OnPointCreate, handId: " + cxt.getNID());
  }

  public void OnPointUpdate(XnVHandPointContext cxt)
  {
    //println("OnPointUpdate " + cxt.getPtPosition());
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));
  }

  public void OnPointDestroy(long nID)
  {
    println("OnPointDestroy, handId: " + nID);

    // remove list
    if(_pointLists.containsKey(nID))
       _pointLists.remove(nID);
  }

  public ArrayList getPointList(long handId)
  {
    ArrayList curList;
    if(_pointLists.containsKey(handId))
      curList = (ArrayList)_pointLists.get(handId);
    else
    {
      curList = new ArrayList(_maxPoints);
      _pointLists.put(handId,curList);
    }
    return curList;
  }

  public void addPoint(long handId,PVector handPoint)
  {
    ArrayList curList = getPointList(handId);

    curList.add(0,handPoint);
    if(curList.size() > _maxPoints)
      curList.remove(curList.size() - 1);


      output.println(handPoint);
      if(handPoint.x>150){
        xcoord0=handPoint.x;
        //output.println("START");
        check=0;
      }
      else if(handPoint.x<-200){

          if(check==0){
          //output.println("TRIGGER");

          if(time==0){
          sound1.play();
          sound1.rewind();
          time=1;
          }
          else{
          sound2.play();
          sound2.rewind();
          time=0;
          }


        }
        check=1;
        }


      //count++;
      //output.println(handPoint);



  }

  public void draw()
  {
    if(_pointLists.size() <= 0)
      return;

    pushStyle();
      noFill();

      PVector vec;
      PVector firstVec;
      PVector screenPos = new PVector();
      int colorIndex=0;

      // draw the hand lists
      Iterator<Map.Entry> itrList = _pointLists.entrySet().iterator();
      while(itrList.hasNext())
      {
        strokeWeight(2);
        stroke(_colorList[colorIndex % (_colorList.length - 1)]);

        ArrayList curList = (ArrayList)itrList.next().getValue();

        // draw line
        firstVec = null;
        Iterator<PVector> itr = curList.iterator();
        beginShape();
          while (itr.hasNext())
          {
            vec = itr.next();
            if(firstVec == null)
              firstVec = vec;
            // calc the screen pos
            context.convertRealWorldToProjective(vec,screenPos);
            vertex(screenPos.x,screenPos.y);
          }
        endShape();

        // draw current pos of the hand
        if(firstVec != null)
        {
          strokeWeight(8);
          context.convertRealWorldToProjective(firstVec,screenPos);
          point(screenPos.x,screenPos.y);
        }
        colorIndex++;
      }

    popStyle();
  }
```

```
}

void stop(){
  minim.stop();
  super.stop();
  output.flush();
  output.close();
}
```

<u>Gesture: I'm thinking</u>

```
import SimpleOpenNI.*;
import java.util.Iterator;
import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Minim minim;
AudioPlayer sound1;
AudioPlayer sound2;


static int time=0;
int check=0;
float xcoord0, xcoord99, diff;

SimpleOpenNI      context;

// NITE
XnVSessionManager sessionManager;
XnVFlowRouter     flowRouter;

PointDrawer       pointDrawer;

PrintWriter output;

void setup()
{


  minim = new Minim(this);
  sound1=minim.loadFile("Thinking1.wav");
  sound2=minim.loadFile("Thinking2.wav");

  output = createWriter("out.txt");
  context = new SimpleOpenNI(this);

  // mirror is by default enabled
  context.setMirror(true);

  // enable depthMap generation
  if(context.enableDepth() == false)
  {
     println("Can't open the depthMap, maybe the camera is not connected!");
     exit();
     return;
  }

  // enable the hands + gesture
  context.enableGesture();
  context.enableHands();
  // setup NITE
  sessionManager = context.createSessionManager("Click,Wave", "RaiseHand");

  pointDrawer = new PointDrawer();
  flowRouter = new XnVFlowRouter();
  flowRouter.SetActive(pointDrawer);

  sessionManager.AddListener(flowRouter);

  size(context.depthWidth(), context.depthHeight());
  smooth();
}

void draw()
{
  background(200,0,0);
  // update the cam
  context.update();

  // update nite
  context.update(sessionManager);

  // draw depthImageMap
  image(context.depthImage(),0,0);

  // draw the list
  pointDrawer.draw();
}

void keyPressed()
{
  switch(key)
  {
  case 'e':
    // end sessions
    sessionManager.EndSession();
    println("end session");
    break;
  }
}

/////////////////////////////////////////////////////////////////////////////////////////////////////
// session callbacks

void onStartSession(PVector pos)
{
  println("onStartSession: " + pos);
}

void onEndSession()
{
  println("onEndSession: ");
}

void onFocusSession(String strFocus,PVector pos,float progress)
{
  println("onFocusSession: focus=" + strFocus + ",pos=" + pos + ",progress=" + progress);
}
```

```
/////////////////////////////////////////////////////////////////////////////////////////////////
// PointDrawer keeps track of the handpoints

class PointDrawer extends XnVPointControl
{
  HashMap    _pointLists;
  int        _maxPoints;
  color[]    _colorList = { color(255,0,0),color(0,255,0),color(0,0,255),color(255,255,0)};

  public PointDrawer()
  {
    _maxPoints = 30;
    _pointLists = new HashMap();
  }

  public void OnPointCreate(XnVHandPointContext cxt)
  {
    // create a new list
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));

    println("OnPointCreate, handId: " + cxt.getNID());
  }

  public void OnPointUpdate(XnVHandPointContext cxt)
  {
    //println("OnPointUpdate " + cxt.getPtPosition());
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));
  }

  public void OnPointDestroy(long nID)
  {
    println("OnPointDestroy, handId: " + nID);

    // remove list
    if(_pointLists.containsKey(nID))
       _pointLists.remove(nID);
  }

  public ArrayList getPointList(long handId)
  {
    ArrayList curList;
    if(_pointLists.containsKey(handId))
      curList = (ArrayList)_pointLists.get(handId);
    else
    {
      curList = new ArrayList(_maxPoints);
      _pointLists.put(handId,curList);
    }
    return curList;
  }

  public void addPoint(long handId,PVector handPoint)
  {
    ArrayList curList = getPointList(handId);

    curList.add(0,handPoint);
    if(curList.size() > _maxPoints)
      curList.remove(curList.size() - 1);
    output.println(handPoint);
    if(handPoint.x>150){
      xcoord0=handPoint.x;
      //output.println("START");
      check=0;
    }
    else if(handPoint.x<-200){

        if(check==0){
        //output.println("TRIGGER");

        if(time==0){
        sound1.play();
        sound1.rewind();
        time=1;
        }
        else{
        sound2.play();
        sound2.rewind();
        time=0;
        }


      }
      check=1;

      }


    //count++;
    //output.println(handPoint);



}

public void draw()
{
  if(_pointLists.size() <= 0)
    return;

  pushStyle();
    noFill();

    PVector vec;
    PVector firstVec;
    PVector screenPos = new PVector();
    int colorIndex=0;

    // draw the hand lists
    Iterator<Map.Entry> itrList = _pointLists.entrySet().iterator();
    while(itrList.hasNext())
    {
      strokeWeight(2);
      stroke(_colorList[colorIndex % (_colorList.length - 1)]);

      ArrayList curList = (ArrayList)itrList.next().getValue();

      // draw line
      firstVec = null;
```

```
        Iterator<PVector> itr = curList.iterator();
        beginShape();
          while (itr.hasNext())
          {
            vec = itr.next();
            if(firstVec == null)
              firstVec = vec;
            // calc the screen pos
            context.convertRealWorldToProjective(vec,screenPos);
            vertex(screenPos.x,screenPos.y);
          }
        endShape();

        // draw current pos of the hand
        if(firstVec != null)
        {
          strokeWeight(8);
          context.convertRealWorldToProjective(firstVec,screenPos);
          point(screenPos.x,screenPos.y);
        }
        colorIndex++;
      }

    popStyle();
  }

}

void stop(){
  minim.stop();
  super.stop();
  output.flush();
  output.close();
}
```

<u>Gesture: Something in the Way</u>

```
import SimpleOpenNI.*;
import java.util.Iterator;
import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Minim minim;
AudioPlayer sound1;
AudioPlayer sound2;


static int time=0;
int check=0;
float xcoord0, xcoord99, diff;

SimpleOpenNI      context;

// NITE
XnVSessionManager sessionManager;
XnVFlowRouter     flowRouter;

PointDrawer       pointDrawer;
```

```
PrintWriter output;

void setup()
{


  minim = new Minim(this);
  sound1=minim.loadFile("SomethingInTheWay1.wav");
  sound2=minim.loadFile("SomethingInTheWay2.wav");

  output = createWriter("out.txt");
  context = new SimpleOpenNI(this);

  // mirror is by default enabled
  context.setMirror(true);

  // enable depthMap generation
  if(context.enableDepth() == false)
  {
     println("Can't open the depthMap, maybe the camera is not connected!");
     exit();
     return;
  }

  // enable the hands + gesture
  context.enableGesture();
  context.enableHands();

  // setup NITE
  sessionManager = context.createSessionManager("Click,Wave", "RaiseHand");

  pointDrawer = new PointDrawer();
  flowRouter = new XnVFlowRouter();
  flowRouter.SetActive(pointDrawer);

  sessionManager.AddListener(flowRouter);

  size(context.depthWidth(), context.depthHeight());
  smooth();
}

void draw()
{
  background(200,0,0);
  // update the cam
  context.update();

  // update nite
  context.update(sessionManager);

  // draw depthImageMap
  image(context.depthImage(),0,0);

  // draw the list
  pointDrawer.draw();
}

void keyPressed()
{
  switch(key)
  {
```

```
  case 'e':
    // end sessions
    sessionManager.EndSession();
    println("end session");
    break;
  }
}

/////////////////////////////////////////////////////////////////////////////////////////////////
// session callbacks

void onStartSession(PVector pos)
{
  println("onStartSession: " + pos);
}

void onEndSession()
{
  println("onEndSession: ");
}

void onFocusSession(String strFocus,PVector pos,float progress)
{
  println("onFocusSession: focus=" + strFocus + ",pos=" + pos + ",progress=" + progress);
}


/////////////////////////////////////////////////////////////////////////////////////////////////
// PointDrawer keeps track of the handpoints

class PointDrawer extends XnVPointControl
{
  HashMap    _pointLists;
  int        _maxPoints;
  color[]    _colorList = { color(255,0,0),color(0,255,0),color(0,0,255),color(255,255,0)};

  public PointDrawer()
  {
    _maxPoints = 30;
    _pointLists = new HashMap();
  }

  public void OnPointCreate(XnVHandPointContext cxt)
  {
    // create a new list
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));

    println("OnPointCreate, handId: " + cxt.getNID());
  }

  public void OnPointUpdate(XnVHandPointContext cxt)
  {
    //println("OnPointUpdate " + cxt.getPtPosition());
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));
  }

  public void OnPointDestroy(long nID)
  {
    println("OnPointDestroy, handId: " + nID);
```

```
 // remove list
 if(_pointLists.containsKey(nID))
    _pointLists.remove(nID);
}

public ArrayList getPointList(long handId)
{
 ArrayList curList;
 if(_pointLists.containsKey(handId))
   curList = (ArrayList)_pointLists.get(handId);
 else
 {
   curList = new ArrayList(_maxPoints);
   _pointLists.put(handId,curList);
 }
 return curList;
}

public void addPoint(long handId,PVector handPoint)
{
 ArrayList curList = getPointList(handId);

 curList.add(0,handPoint);
 if(curList.size() > _maxPoints)
   curList.remove(curList.size() - 1);


   output.println(handPoint);
   if(handPoint.x>150){
     xcoord0=handPoint.x;
     //output.println("START");
     check=0;
   }
   else if(handPoint.x<-200){

       if(check==0){
       //output.println("TRIGGER");

       if(time==0){
       sound1.play();
       sound1.rewind();
       time=1;
       }
       else{
       sound2.play();
       sound2.rewind();
       time=0;
       }


     }
     check=1;

     }


   //count++;
   //output.println(handPoint);
```

```
  }

  public void draw()
  {
    if(_pointLists.size() <= 0)
      return;

    pushStyle();
      noFill();

      PVector vec;
      PVector firstVec;
      PVector screenPos = new PVector();
      int colorIndex=0;

      // draw the hand lists
      Iterator<Map.Entry> itrList = _pointLists.entrySet().iterator();
      while(itrList.hasNext())
      {
        strokeWeight(2);
        stroke(_colorList[colorIndex % (_colorList.length - 1)]);

        ArrayList curList = (ArrayList)itrList.next().getValue();

        // draw line
        firstVec = null;
        Iterator<PVector> itr = curList.iterator();
        beginShape();
          while (itr.hasNext())
          {
            vec = itr.next();
            if(firstVec == null)
              firstVec = vec;
            // calc the screen pos
            context.convertRealWorldToProjective(vec,screenPos);
            vertex(screenPos.x,screenPos.y);
          }
        endShape();

        // draw current pos of the hand
        if(firstVec != null)
        {
          strokeWeight(8);
          context.convertRealWorldToProjective(firstVec,screenPos);
          point(screenPos.x,screenPos.y);
        }
        colorIndex++;
      }

    popStyle();
  }

}

void stop(){
  minim.stop();
  super.stop();
  output.flush();
  output.close();
}
```

Appendix DD

Phase II Code: Arduino - Light

Button: Up/Down

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int output1=6;
int input1=7;
int valDown=0;
int valUp=0;

int dataPin = 2;
int clockPin = 3;

static int countDown=0;
static int countUp=0;

LPD8806 strip = LPD8806(52, dataPin, clockPin);


void setup() {

  pinMode(output1, OUTPUT);
  pinMode(input1,INPUT);
  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();

  Serial.begin(9600);
}

void loop(){

  digitalWrite(output1, HIGH);
  valDown=analogRead(A0);
  valUp=analogRead(A5);

  float voltageDown=valDown*(0.5/1024.0);
  float voltageUp=valUp*(0.5/1024.0);

  Serial.println(voltageDown);
  Serial.println(voltageUp);

  if(voltageDown>0.4){

    if(countDown==0){
      colorChasedown(strip.Color(127,127,127), 20);
      Clear(1000);
      countDown++;
    }
    else{
      GradientDown(80);
      Clear(1000);
      countDown=0;
    }
  }

  if(voltageUp>0.4){
    if(countUp==0){
      colorChase(strip.Color(127,127,127), 20);
      Clear(1000);
      countUp++;
    }
    else{
      GradientUp(80);
      Clear(1000);
      countUp=0;
    }
  }

  else{
    for(int i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }
  }

}

void colorChasedown(uint32_t c, uint8_t wait) {
  int i;
  int j;

for (j=0; j < 4; j++){

  for (i=strip.numPixels()-1; i >= 0; i--) {
    strip.setPixelColor(i, 0);  // turn all pixels off
  }

  for (i=strip.numPixels()-1; i >=0; i--) {
      strip.setPixelColor(i, c); // set one pixel
      strip.show();              // refresh strip display
      delay(wait);               // hold image for a moment
      strip.setPixelColor(i, 0); // erase pixel (but don't refresh yet)
  }
  strip.show(); // for last erased pixel
}
}

#define PI 3.14159265
void GradientDown(int8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
      int m=floor(127/nLeEDs);
      if(x==0){
        r=x*m;
        g=x*m;
        b=x*m;}

      else if(x%4==2){
        r=x*m;
        g=x*m;
        b=x*m;
      }

      strip.setPixelColor(i, -r, -g, -b);
    }
    strip.show();
    if(x==0){
      delay(500);
    }
      else{
    delay(wait);
      }
  }
}




void colorChase(uint32_t c, uint8_t wait) {
  int i;
  int j;

for (j=0; j < 4; j++){

  for (i=0; i < strip.numPixels(); i++) {
    strip.setPixelColor(i, 0);  // turn all pixels off
  }

  for (i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, c); // set one pixel
      strip.show();              // refresh strip display
      delay(wait);               // hold image for a moment
      strip.setPixelColor(i, 0); // erase pixel (but don't refresh yet)
  }
  strip.show(); // for last erased pixel
}
}

#define PI 3.14159265
void GradientUp(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
      int m=floor(127/nLeEDs);
      if(x==0){
        r=x*m;
        g=x*m;
        b=x*m;}

      else if(x%4==2){
          r=x*m;
          g=x*m;
          b=x*m;
      }
      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    if(x==0){
      delay(500);
    }
      else{
    delay(wait);
      }
  }
}



void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}
```

Button: Tilt Forward

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

int inPin = 7;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status

static int count = 0;

LPD8806 strip = LPD8806(52, dataPin, clockPin);


void setup() {
  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();

  pinMode(inPin, INPUT);
}

void loop(){

val = digitalRead(inPin);

  if (val == HIGH) {
    if (count == 0) {

    On(80);

    Clear(80);

    count += 1;

  }

  else if (count == 1){

    Half(80);

    Clear(80);

    count = 0;
  }

}
}


void On(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x>=0){
        r=127;
        g=127;
        b=127;}

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }


void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}


void Half(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x<16){
         if (i>16){
        r=127;
        g=127;
        b=127;
         }

         else{
          r=0;
          g=0;
          b=0;
         }
       }

       else {
         if (i<=16){
        r=127;
        g=127;
        b=127;
         }

         else{
          r=0;
          g=0;
          b=0;
         }
       }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }
```

<u>Button: Tilt Back</u>

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

int inPin = 7;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status

static int count = 0;

LPD8806 strip = LPD8806(52, dataPin, clockPin);


void setup() {
  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();

  pinMode(inPin, INPUT);
}

void loop(){

 val = digitalRead(inPin);

  if (val == HIGH) {

    if (count == 0) {

     On(80);

     Clear(80);

    count += 1;

  }

  else if (count == 1){

    Half(80);

    Clear(80);

    count = 0;
  }

}
}


void On(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x>=0){
        r=127;
        g=127;
        b=127;}

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }


void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}
```

```
void Half(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x<16){
         if (i<16){
        r=127;
        g=127;
        b=127;
         }

         else{
          r=0;
          g=0;
          b=0;
         }
       }

       else {
         if (i>=16){
        r=127;
        g=127;
        b=127;
         }

         else{
          r=0;
          g=0;
          b=0;
         }
       }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }
```

Button: Emergency

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

int inPin = 7;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status

static int count = 0;

LPD8806 strip = LPD8806(52, dataPin, clockPin);
```

```
void setup() {
  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();

  pinMode(inPin, INPUT);
}

void loop(){

  val = digitalRead(inPin);

  if (val == HIGH) {

    if (count == 0) {

    OnOff(80);

    Clear(80);

    count += 1;

  }

  else if (count == 1){

    wave(2, 40);

    Clear(80);

    count = 0;
  }

}
}

void OnOff(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<strip.numPixels(); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {

      if(x%2==0){
        r=32;
        g=32;
        b=32;}

      else if(x%2==1){
          r=0;
          g=0;
          b=0;
      }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
  }
```

```
 }

void Clear(uint8_t wait) {
 int i;

 for(i=0; i<strip.numPixels(); i++) {
   strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

 }

 strip.show(); // Refresh to turn off last pixel
 delay(wait);
}

 #define PI 3.14159265
void wave(int cycles, uint8_t wait) {
 float y;
 byte  r, g, b, r2, g2, b2;


 // Need to decompose color into its r, g, b elements
 g = 32;
 r = 32;
 b = 32;

 for(int x=0; x<(strip.numPixels()*5); x++)
 {
   for(int i=0; i<strip.numPixels(); i++) {
     y = sin(PI * (float)cycles * (float)(x + i) / (float)strip.numPixels());
     if(y >= 0.0) {
       // Peaks of sine wave are white
       y  = 1.0 - y; // Translate Y to 0.0 (top) to 1.0 (center)
       r2 = 127 - (byte)((float)(127 - r) * y);
       g2 = 127 - (byte)((float)(127 - g) * y);
       b2 = 127 - (byte)((float)(127 - b) * y);
     } else {
       // Troughs of sine wave are black
       y += 1.0; // Translate Y to 0.0 (bottom) to 1.0 (center)
       r2 = (byte)((float)r * y);
       g2 = (byte)((float)g * y);
       b2 = (byte)((float)b * y);
     }
     strip.setPixelColor(i, r2, g2, b2);
   }
   strip.show();
   delay(wait);
 }
}
```

Button: Reprimand

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

int inPin = 7;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status
```

```
static int count = 0;

LPD8806 strip = LPD8806(52, dataPin, clockPin);


void setup() {
  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();

   pinMode(inPin, INPUT);
}

void loop(){

  val = digitalRead(inPin);

  if (val == HIGH) {

    if (count == 0) {

    GradientUp(80);

    GradientDown(80);

    count += 1;

  }

  else if (count == 1){

    ReprimandWompWomp(80);

    count = 0;
  }

}
}



#define PI 3.14159265
void GradientUp(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
      int m=floor(127/nLeEDs);
      if(x==0){
        r=x*m;
        g=x*m;
        b=x*m;}

      else if(x%4==2){
          r=x*m;
          g=x*m;
          b=x*m;
      }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    if(x==0){
      delay(500);
    }
      else{
    delay(wait);
      }
  }
}

#define PI 3.14159265
void GradientDown(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
      int m=floor(127/nLeEDs);
      if(x==strip.numPixels()-1){
        r=0;
        g=0;
        b=0;}

      else if(x%4==2){
          r=x*m;
          g=x*m;
          b=x*m;
      }

      strip.setPixelColor(i, -r, -g, -b);
    }
    strip.show();
    if(x==strip.numPixels()-1){
      delay(500);
    }
      else{
    delay(wait);
      }
  }
}

void ReprimandWompWomp(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<68; x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {

      if(x<15){
        r=127;
        g=127;
        b=127;}

      else if(x>15 && x<23){
          r=0;
          g=0;
          b=0;
      }

      else if(x>23 && x<61){
         r=127;
         g=127;
         b=127;}

      else if(x>61){
          r=0;
          g=0;
          b=0;
      }



      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
  }
  }
```

Button: Bend Out

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

int inPin = 7;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status

static int count = 0;

LPD8806 strip = LPD8806(52, dataPin, clockPin);


void setup() {
  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();

  pinMode(inPin, INPUT);
}

void loop(){
 val = digitalRead(inPin);

  if (val == HIGH) {

    if (count == 0) {

    BendOutScan(80);

    BendOutScan(80);
    Clear(80);

    count += 1;

  }

  else if (count == 1){

    On(80);

    Clear(80);

    count = 0;
  }

}
}

// "Larson scanner" = Cylon/KITT bouncing light effect
void BendOutScan(uint8_t wait) {
  int i, j, pos, dir;
  byte  r, g, b;
  r=127;
  g=127;
  b=127;

  pos = 0;
  dir = 1;

  for(i=0; i<((strip.numPixels()-1)); i++) {
    // Draw 5 pixels centered on pos.  setPixelColor() will clip
    // any pixels off the ends of the strip, no worries there.
    // we'll make the colors dimmer at the edges for a nice pulse
    // look
    strip.setPixelColor(pos - 2, strip.Color(r/8, g/8, b/8));
    strip.setPixelColor(pos - 1, strip.Color(r/4, g/4, b/4));
    strip.setPixelColor(pos, strip.Color(r, g, b));
    strip.setPixelColor(pos + 1, strip.Color(r/4, g/4, b/4));
    strip.setPixelColor(pos + 2, strip.Color(r/8, g/8, b/8));

    strip.show();
    delay(wait);
    // If we wanted to be sneaky we could erase just the tail end
    // pixel, but it's much easier just to erase the whole thing
    // and draw a new one next time.
    for(j=-2; j<= 2; j++)
        strip.setPixelColor(pos+j, strip.Color(0,0,0));

    pos += dir;
    if(pos < 0) {
      pos = 1;
      dir = -dir;
    }
  }
}

void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}

void On(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x>=0){
        r=127;
        g=127;
        b=127;}

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
       }
  }
```

Button: Bend In

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

int inPin = 7;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status

static int count = 0;

LPD8806 strip = LPD8806(52, dataPin, clockPin);


void setup() {
  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();

  pinMode(inPin, INPUT);
}

void loop(){

 val = digitalRead(inPin);

  if (val == HIGH) {

    if (count == 0) {
```

```
    BendIn(80);        // red, slow

    BendIn(80);

    Clear(80);

    count += 1;

  }

  else if (count == 1){

    On(80);

    Clear(80);

    count = 0;
  }

}
}

void BendIn(uint8_t wait) {
  int i, j, pos, dir;
  byte r, g, b;
  r=127;
  g=127;
  b=127;

  pos = (strip.numPixels()-1);
  dir = 1;

  for(i=(strip.numPixels()-1); i<((strip.numPixels()-1)*2); i++) {
    // Draw 5 pixels centered on pos.  setPixelColor() will clip
    // any pixels off the ends of the strip, no worries there.
    // we'll make the colors dimmer at the edges for a nice pulse
    // look
    strip.setPixelColor(pos - 2, strip.Color(r/4, g/4, b/4));
    strip.setPixelColor(pos - 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos, strip.Color(r, g, b));
    strip.setPixelColor(pos + 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos + 2, strip.Color(r/4, g/4, b/4));

    strip.show();
    delay(wait);
    // If we wanted to be sneaky we could erase just the tail end
    // pixel, but it's much easier just to erase the whole thing
    // and draw a new one next time.
    for(j=-2; j<= 2; j++)
        strip.setPixelColor(pos+j, strip.Color(0,0,0));
    // Bounce off ends of strip
    pos += dir;
    if(pos < 0) {
      pos = 1;
      dir = -dir;
    } else if(pos >= strip.numPixels()) {
      pos = strip.numPixels() - 2;
      dir = -dir;
    }
  }
```

```
}


void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}

void On(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x>=0){
        r=127;
        g=127;
        b=127;}

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }
```

Gesture: Come

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

char val; //Data received from serial port

LPD8806 strip = LPD8806(nLeEDs, dataPin, clockPin);


void setup() {

  Serial.begin(9600);

  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();
}

void loop(){
  if(Serial.available()){
    val=Serial.read();
  }

  if(val=='O'){
    Half(80);
    Clear(500);
  }

  else if(val=='T'){
    dither(80);
    Clear(500);
  }

  else{
    for (int i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, 0, 0, 0);
    }
      strip.show();
  }

}

void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}


void Half(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x<16){
         if (i<16){
        r=127;
        g=127;
        b=127;
         }

         else{
          r=0;
          g=0;
          b=0;
         }
       }

       else {
         if (i>=16){
        r=127;
        g=127;
        b=127;
        }

        else{
         r=0;
         g=0;
         b=0;
        }
       }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }

  void dither(uint8_t wait) {

  // Determine highest bit needed to represent pixel index
  int hiBit = 0;
  int n = strip.numPixels() - 1;
  byte r, g, b;
  r=127;
  g=127;
  b=127;

  for(int bit=1; bit < 0x8000; bit <<= 1) {
    if(n & bit) hiBit = bit;
  }

  int bit, reverse;
  for(int i=0; i<(hiBit << 1); i++) {
    // Reverse the bits in i to create ordered dither:
    reverse = 0;
    for(bit=1; bit <= hiBit; bit <<= 1) {
      reverse <<= 1;
      if(i & bit) reverse |= 1;
    }
    strip.setPixelColor(reverse, r, g, b);
    strip.show();
    delay(wait);
  }
  delay(250); // Hold image for 1/4 sec
}
```

Gesture: Go

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

char val; //Data received from serial port

LPD8806 strip = LPD8806(nLeEDs, dataPin, clockPin);
```

```
void setup() {

  Serial.begin(9600);

  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();
}

void loop(){

  if(Serial.available()){
    val=Serial.read();
  }

  if(val=='O'){
    Half(80);
    Clear(500);
  }

  else if(val=='T'){
     On (5);
     Go(80);
    Clear(500);
  }

  else{
    for (int i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, 0, 0, 0);
    }
      strip.show();
  }

}

void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}


void Half(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x<16){
         if (i>16){
        r=127;
        g=127;
        b=127;
         }

         else{
          r=0;
          g=0;
          b=0;
         }
       }

       else {
         if (i<=16){
        r=127;
        g=127;
        b=127;
         }

         else{
          r=0;
          g=0;
          b=0;
         }
       }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }

  void Go(uint8_t wait) {

  // Determine highest bit needed to represent pixel index
  int hiBit = 0;
  int n = strip.numPixels() - 1;
  byte r, g, b;
  r=0;
  g=0;
  b=0;

  for(int bit=1; bit < 0x8000; bit <<= 1) {
    if(n & bit) hiBit = bit;
  }

  int bit, reverse;
  for(int i=0; i<(hiBit << 1); i++) {
    // Reverse the bits in i to create ordered dither:
    reverse = 0;
    for(bit=1; bit <= hiBit; bit <<= 1) {
      reverse <<= 1;
      if(i & bit) reverse |= 1;
    }
    strip.setPixelColor(reverse, r, g, b);
    strip.show();
    delay(wait);
  }
  delay(250); // Hold image for 1/4 sec
}

void On(uint8_t wait) {
  byte  r, g, b;
  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x>=0){
        r=127;
        g=127;
        b=127;}

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }
```

Gesture: Stop

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

char val; //Data received from serial port

LPD8806 strip = LPD8806(nLeEDs, dataPin, clockPin);


void setup() {

  Serial.begin(9600);

  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();
}

void loop(){

  if(Serial.available()){
    val=Serial.read();
  }

  if(val=='O'){
    ReprimandWompWomp(80);
    Clear(1000);
  }

  else if(val=='T'){
    GradientDown(80);
    Clear(1000);
  }

  else{
    for (int i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, 0, 0, 0);
    }
      strip.show();
  }

}

void ReprimandWompWomp(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<80; x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {

      if(x<10){
        r=127;
        g=127;
        b=127;}

      else if(x>10 && x<20){
          r=0;
          g=0;
          b=0;
      }

      else if(x>20 && x<30){
         r=127;
         g=127;
         b=127;}

      else if(x>30 && x<40){
          r=0;
          g=0;
          b=0;
      }

      else if(x>40 && x<50){
         r=127;
         g=127;
         b=127;}

      else if(x>50 && x<60){
          r=0;
          g=0;
          b=0;
      }

      else if(x>60 && x<70){
         r=127;
         g=127;
         b=127;}

       else if(x>70){
         r=0;
         g=0;
         b=0;}

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
  }
```

```
 }

 void Clear(uint8_t wait) {
 int i;

 for(i=0; i<strip.numPixels(); i++) {
   strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

 }

 strip.show(); // Refresh to turn off last pixel
 delay(wait);
}


 #define PI 3.14159265
void GradientDown(uint8_t wait) {
 byte  r, g, b;

 for(int x=0; x<(strip.numPixels()); x++)
 {
   for(int i=0; i<strip.numPixels(); i++) {
     int m=floor(127/nLeEDs);
     if(x==strip.numPixels()-1){
       r=0;
       g=0;
       b=0;}

     else if(x%4==2){
         r=x*m;
         g=x*m;
         b=x*m;
     }

     strip.setPixelColor(i, -r, -g, -b);
   }
   strip.show();
   if(x==strip.numPixels()-1){
     delay(500);
   }
     else{
   delay(wait);
     }
 }
}
```

Gesture: Confirm Request

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

char val; //Data received from serial port

LPD8806 strip = LPD8806(nLeEDs, dataPin, clockPin);
```

```
void setup() {

  Serial.begin(9600);

  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();
}

void loop(){

  if(Serial.available()){
    val=Serial.read();
  }

  if(val=='O'){
    Confirm(80);
    Clear(1000);
  }

  else if(val=='T'){
    wave(2, 40);
    Clear(1000);
  }

  else{
    for (int i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, 0, 0, 0);
    }
      strip.show();
  }

}

void Confirm(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<18; x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {

      if(x<3){
        r=127;
        g=127;
        b=127;}

      else if(x>3 && x<6){
          r=0;
          g=0;
          b=0;
      }

      else if(x>6 && x<9){
         r=127;
         g=127;
         b=127;}

      else if(x>9 && x<12){
        r=0;
        g=0;
        b=0;
      }

      else if(x>12 && x<15){
        r=127;
        g=127;
        b=127;}

      else if(x>15){
        r=0;
        g=0;
        b=0;
      }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
  }
  }

  void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}

#define PI 3.14159265
void wave(int cycles, uint8_t wait) {
  float y;
  byte  r, g, b, r2, g2, b2;


  // Need to decompose color into its r, g, b elements
  g = 0;
  r = 0;
  b = 0;

  for(int x=0; x<(strip.numPixels()*5); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
      y = sin(PI * (float)cycles * (float)(x + i) / (float)strip.numPixels());
      if(y >= 0.0) {
        // Peaks of sine wave are white
        y  = 1.0 - y; // Translate Y to 0.0 (top) to 1.0 (center)
        r2 = 127 - (byte)((float)(127 - r) * y);
        g2 = 127 - (byte)((float)(127 - g) * y);
        b2 = 127 - (byte)((float)(127 - b) * y);
      } else {
        // Troughs of sine wave are black
        y += 1.0; // Translate Y to 0.0 (bottom) to 1.0 (center)
        r2 = (byte)((float)r * y);
        g2 = (byte)((float)g * y);
        b2 = (byte)((float)b * y);
      }
      strip.setPixelColor(i, r2, g2, b2);
    }
    strip.show();
    delay(wait);
  }
}
```

Gesture: Do not Understand Request

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

char val; //Data received from serial port

LPD8806 strip = LPD8806(nLeEDs, dataPin, clockPin);


void setup() {

  Serial.begin(9600);

  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();
}

void loop(){

  if(Serial.available()){
    val=Serial.read();
  }

  if(val=='O'){
    colorWipe(60);
    Clear(1000);
  }

  else if(val=='T'){
    OnOff(175);
    Clear(1000);
  }

  else{
    for (int i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, 0, 0, 0);
    }
      strip.show();
  }

}

void colorWipe(uint8_t wait) {
  int i, x;
  byte r, g, b;

  for (x=0; x<(strip.numPixels()); x++)
  {
  for (i=0; i < strip.numPixels(); i++) {

    if (x%2==0){
      r=32;
      g=32;
      b=32;
    }

    else if (x%2==1){
      r=0;
      g=0;
      b=0;
    }

      strip.setPixelColor(i, r, g, b);
      strip.show();
  }
      delay(wait);
}
}

void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}

void OnOff(uint8_t wait){
  int i, x;
  byte r, g, b;

  for (x=0; x<strip.numPixels(); x++)
  {
    for (i=0; i<strip.numPixels(); i++){

     if(x%2==0){

      if(i%2==0){
        r=127;
        g=127;
        b=127;
      }

      else if(i%2==1){
        r=0;
        g=0;
        b=0;
      }
      }
     else if(x%2==1){
      if(i%2==0){
        r=0;
        g=0;
        b=0;
      }

      else if(i%2==1){
        r=127;
        g=127;
        b=127;
      }
      }

       strip.setPixelColor(i, r, g, b);
    }

      strip.show();
      delay(wait);
  }


}
```

Gesture: Can't Do

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

char val; //Data received from serial port

LPD8806 strip = LPD8806(nLeEDs, dataPin, clockPin);


void setup() {

  Serial.begin(9600);

  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();
}

void loop(){

  if(Serial.available()){
    val=Serial.read();
  }

  if(val=='O'){
    CantDo(80);
    Clear(1000);
  }
  else if(val=='T'){
    Thinking(10);
    Clear(1000);
  }

  else{
    for (int i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, 0, 0, 0);
    }
      strip.show();
  }

}

void CantDo(uint8_t wait) {
  int i, x, pos, dir, pos2, dir2;
  byte r, g, b;
  r=32;
  g=32;
  b=32;

  pos = ((strip.numPixels()/2)-1);
  pos2 = ((strip.numPixels()/2)-1);
  dir = 1;
  dir2 = -1;

  for(i=0; i<strip.numPixels(); i++) {
    // Draw 5 pixels centered on pos.  setPixelColor() will clip
    // any pixels off the ends of the strip, no worries there.
    // we'll make the colors dimmer at the edges for a nice pulse
    // look

    strip.setPixelColor(pos - 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos, strip.Color(r, g, b));
    strip.setPixelColor(pos + 1, strip.Color(r/2, g/2, b/2));

    strip.setPixelColor(pos2 - 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos2, strip.Color(r, g, b));
    strip.setPixelColor(pos2 + 1, strip.Color(r/2, g/2, b/2));


    strip.show();
    delay(wait);
    // If we wanted to be sneaky we could erase just the tail end
    // pixel, but it's much easier just to erase the whole thing
    // and draw a new one next time.
   int j=-2;
   for(x=-2; x<= 2; x++){
       strip.setPixelColor(pos+x, strip.Color(0,0,0));
       strip.setPixelColor(pos2-x, strip.Color(0,0,0));
   }

       j+=-1;
    // Bounce off ends of strip
    pos += dir;
    pos2 += dir2;
    if(pos < 0) {
      pos = 1;
      dir = -dir;
    }
    else if (pos2 < 0){
      pos2 = 1;
      dir2 = -dir2;
    }

   if(pos >= strip.numPixels()) {
      pos = strip.numPixels() - 2;
      dir = -dir;
    }

    else if(pos2 >= strip.numPixels()) {
      pos2 = strip.numPixels() - 2;
      dir2 = -dir2;
    }
  }
}

void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}

void Thinking(uint8_t wait) {
  int i, j, pos, dir;
  byte r, g, b;
  r=127;
  g=127;
  b=127;

  pos = 0;
  dir = 1;

  for(i=0; i<((strip.numPixels()-1) * 8); i++) {
    // Draw 5 pixels centered on pos.  setPixelColor() will clip
    // any pixels off the ends of the strip, no worries there.
    // we'll make the colors dimmer at the edges for a nice pulse
    // look
    strip.setPixelColor(pos - 2, strip.Color(r/4, g/4, b/4));
    strip.setPixelColor(pos - 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos, strip.Color(r, g, b));
    strip.setPixelColor(pos + 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos + 2, strip.Color(r/4, g/4, b/4));

    strip.show();
    delay(wait);
    // If we wanted to be sneaky we could erase just the tail end
    // pixel, but it's much easier just to erase the whole thing
    // and draw a new one next time.
    for(j=-2; j<= 2; j++)
        strip.setPixelColor(pos+j, strip.Color(0,0,0));
    // Bounce off ends of strip
    pos += dir;
    if(pos < 0) {
      pos = 1;
```

```
        dir = -dir;
      } else if(pos >= strip.numPixels()) {
        pos = strip.numPixels() - 2;
        dir = -dir;
      }
    }
}
```

<u>Gesture: I’m thinking</u>

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

char val; //Data received from serial port

LPD8806 strip = LPD8806(nLeEDs, dataPin, clockPin);


void setup() {

  Serial.begin(9600);

  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();
}

void loop(){

  if(Serial.available()){
    val=Serial.read();
  }

  if(val=='O'){
    Thinking(80);
    Clear(1000);
  }

  else if(val=='T'){
    GradientUp(80);
    GradientDown(80);
    Clear(1000);
  }

  else{
    for (int i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, 0, 0, 0);
    }
      strip.show();
  }

}

void Thinking(uint8_t wait) {
  int i, j, pos, dir;
  byte r, g, b;
  r=127;
  g=127;
  b=127;

  pos = 0;
  dir = 1;

  for(i=0; i<((strip.numPixels()-1) * 3); i++) {
    // Draw 5 pixels centered on pos.  setPixelColor() will clip
    // any pixels off the ends of the strip, no worries there.
    // we'll make the colors dimmer at the edges for a nice pulse
    // look
    strip.setPixelColor(pos - 2, strip.Color(r/4, g/4, b/4));
    strip.setPixelColor(pos - 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos, strip.Color(r, g, b));
    strip.setPixelColor(pos + 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos + 2, strip.Color(r/4, g/4, b/4));

    strip.show();
    delay(wait);
    // If we wanted to be sneaky we could erase just the tail end
    // pixel, but it's much easier just to erase the whole thing
    // and draw a new one next time.
    for(j=-2; j<= 2; j++)
        strip.setPixelColor(pos+j, strip.Color(0,0,0));
    // Bounce off ends of strip
    pos += dir;
    if(pos < 0) {
      pos = 1;
      dir = -dir;
    } else if(pos >= strip.numPixels()) {
      pos = strip.numPixels() - 2;
      dir = -dir;
    }
  }
}

void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}

#define PI 3.14159265
void GradientUp(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
      int m=floor(127/nLeEDs);
      if(x==0){
        r=x*m;
        g=x*m;
        b=x*m;}

      else if(x%4==2){
          r=x*m;
          g=x*m;
          b=x*m;
      }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    if(x==0){
      delay(500);
    }
      else{
    delay(wait);
      }
  }
}

#define PI 3.14159265
void GradientDown(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
      int m=floor(127/nLeEDs);
      if(x==strip.numPixels()-1){
        r=0;
        g=0;
        b=0;}

      else if(x%4==2){
          r=x*m;
          g=x*m;
          b=x*m;
      }

      strip.setPixelColor(i, -r, -g, -b);
    }
    strip.show();
    if(x==strip.numPixels()-1){
      delay(500);
    }
      else{
    delay(wait);
      }
  }
}
```

Gesture: Something in the Way

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;
```

```
char val; //Data received from serial port

LPD8806 strip = LPD8806(nLeEDs, dataPin, clockPin);


void setup() {

  Serial.begin(9600);

  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();
}

void loop(){

  if(Serial.available()){
    val=Serial.read();
  }

  if(val=='O'){
    OnOff(80);
    Clear(1000);
  }

  else if(val=='T'){
    dither(80);
    Go(80);
    Clear(1000);
  }

  else{
    for (int i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, 0, 0, 0);
    }
      strip.show();
  }

}

void OnOff(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<strip.numPixels()+1; x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {

      if(x%5==0){
        r=32;
        g=32;
        b=32;}

      else if(x%5==1){
          r=0;
          g=0;
          b=0;
      }
      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
  }
  }

void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}

  void dither(uint8_t wait) {

  // Determine highest bit needed to represent pixel index
  int hiBit = 0;
  int n = strip.numPixels() - 1;
  byte r, g, b;
  r=127;
  g=127;
  b=127;

  for(int bit=1; bit < 0x8000; bit <<= 1) {
    if(n & bit) hiBit = bit;
  }

  int bit, reverse;
  for(int i=0; i<(hiBit << 1); i++) {
    // Reverse the bits in i to create ordered dither:
    reverse = 0;
    for(bit=1; bit <= hiBit; bit <<= 1) {
      reverse <<= 1;
      if(i & bit) reverse |= 1;
    }
    strip.setPixelColor(reverse, r, g, b);
    strip.show();
    delay(wait);
  }
  delay(250); // Hold image for 1/4 sec
}

void Go(uint8_t wait) {

  // Determine highest bit needed to represent pixel index
  int hiBit = 0;
  int n = strip.numPixels() - 1;
  byte r, g, b;
  r=0;
  g=0;
  b=0;

  for(int bit=1; bit < 0x8000; bit <<= 1) {
    if(n & bit) hiBit = bit;
  }
  int bit, reverse;
  for(int i=0; i<(hiBit << 1); i++) {
    // Reverse the bits in i to create ordered dither:
    reverse = 0;
    for(bit=1; bit <= hiBit; bit <<= 1) {
      reverse <<= 1;
      if(i & bit) reverse |= 1;
    }
    strip.setPixelColor(reverse, r, g, b);
    strip.show();
    delay(wait);
  }
  delay(250); // Hold image for 1/4 sec
}

void On(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x>=0){
        r=127;
        g=127;
        b=127;}

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }
```

Appendix EE

Phase II Code: Processing - Light

```
/* --------------------------------------------------------------------------
 * SimpleOpenNI NITE Hands
 * --------------------------------------------------------------------------
 * Processing Wrapper for the OpenNI/Kinect library
 * http://code.google.com/p/simple-openni
 * --------------------------------------------------------------------------
 * prog:  Max Rheiner / Interaction Design / zhdk / http://iad.zhdk.ch/
 * date:  03/19/2011 (m/d/y)
 * ----------------------------------------------------------------------------
 * This example works with multiple hands, to enable mutliple hand change
 * the ini file in /usr/etc/primesense/XnVHandGenerator/Nite.ini:
 *  [HandTrackerManager]
 *  AllowMultipleHands=1
 *  TrackAdditionalHands=1
 * on Windows you can find the file at:
 *  C:\Program Files (x86)\Prime Sense\NITE\Hands\Data\Nite.ini
 * ----------------------------------------------------------------------------
 */

import SimpleOpenNI.*;
import java.util.Iterator;
import java.util.Map;
import processing.serial.*;

Serial port; // Create object from Serial class
int val; // Data received from the serial port

static int time=0;
int check=0;
float xcoord0, xcoord99, diff;

SimpleOpenNI      context;

// NITE
XnVSessionManager sessionManager;
XnVFlowRouter     flowRouter;

PointDrawer       pointDrawer;

PrintWriter output;

void setup()
{

  println(Serial.list()); //This shows the various serial port options
  String portName = Serial.list()[0]; //The serial port should match the one the Arduino is hooked to
  port = new Serial(this, portName, 9600); //Establish the connection rate

  //output = createWriter("out.txt");
  context = new SimpleOpenNI(this);

  // mirror is by default enabled
  context.setMirror(true);

  // enable depthMap generation
  if(context.enableDepth() == false)
  {
     println("Can't open the depthMap, maybe the camera is not connected!");
     exit();
     return;
  }

  // enable the hands + gesture
  context.enableGesture();
  context.enableHands();

  // setup NITE
  sessionManager = context.createSessionManager("Click,Wave", "RaiseHand");

  pointDrawer = new PointDrawer();
  flowRouter = new XnVFlowRouter();
  flowRouter.SetActive(pointDrawer);

  sessionManager.AddListener(flowRouter);

  size(context.depthWidth(), context.depthHeight());
  smooth();
}

void draw()
{
  background(200,0,0);
  // update the cam
  context.update();

  // update nite
  context.update(sessionManager);

  // draw depthImageMap
  image(context.depthImage(),0,0);

  // draw the list
  pointDrawer.draw();
}

void keyPressed()
{
  switch(key)
  {
  case 'e':
    // end sessions
    sessionManager.EndSession();
    println("end session");
    break;
  }
}

/////////////////////////////////////////////////////////////////////////////////////////////////
// session callbacks

void onStartSession(PVector pos)
{
  println("onStartSession: " + pos);
}

void onEndSession()
```

```
{
  println("onEndSession: ");
}

void onFocusSession(String strFocus,PVector pos,float progress)
{
  println("onFocusSession: focus=" + strFocus + ",pos=" + pos + ",progress=" + progress);
}


/////////////////////////////////////////////////////////////////////////////////////////////////
// PointDrawer keeps track of the handpoints

class PointDrawer extends XnVPointControl
{
  HashMap    _pointLists;
  int        _maxPoints;
  color[]    _colorList = { color(255,0,0),color(0,255,0),color(0,0,255),color(255,255,0)};

  public PointDrawer()
  {
    _maxPoints = 30;
    _pointLists = new HashMap();
  }

  public void OnPointCreate(XnVHandPointContext cxt)
  {
    // create a new list
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));

    println("OnPointCreate, handId: " + cxt.getNID());
  }

  public void OnPointUpdate(XnVHandPointContext cxt)
  {
    //println("OnPointUpdate " + cxt.getPtPosition());
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));
  }

  public void OnPointDestroy(long nID)
  {
    println("OnPointDestroy, handId: " + nID);

    // remove list
    if(_pointLists.containsKey(nID))
       _pointLists.remove(nID);
  }

  public ArrayList getPointList(long handId)
  {
    ArrayList curList;
    if(_pointLists.containsKey(handId))
      curList = (ArrayList)_pointLists.get(handId);
    else
    {
      curList = new ArrayList(_maxPoints);
      _pointLists.put(handId,curList);
    }
    return curList;
  }

  public void addPoint(long handId,PVector handPoint)
  {
    ArrayList curList = getPointList(handId);

    curList.add(0,handPoint);
    if(curList.size() > _maxPoints)
      curList.remove(curList.size() - 1);

//If hand gesture detected, activate lights from Arduino
      //output.println(handPoint);
      if(handPoint.x>150){
        xcoord0=handPoint.x;
        //output.println("START");
        port.write('N'); //send an N to deactivate light sequence
        check=0;
      }
      else if(handPoint.x<-200){
          if(check==0){
          //output.println("TRIGGER");

          if(time==0){
          port.write('O'); //send an O to activate the first light sequence
          time=1;
          }
          else{
          port.write('T'); //send an T to activate the second light sequence
          time=0;
          }
        }
        check=1;
        port.write('N');

        }

  }

  public void draw()
  {
    if(_pointLists.size() <= 0)
      return;

    pushStyle();
      noFill();

      PVector vec;
      PVector firstVec;
      PVector screenPos = new PVector();
      int colorIndex=0;

      // draw the hand lists
      Iterator<Map.Entry> itrList = _pointLists.entrySet().iterator();
      while(itrList.hasNext())
      {
        strokeWeight(2);
        stroke(_colorList[colorIndex % (_colorList.length - 1)]);

        ArrayList curList = (ArrayList)itrList.next().getValue();

        // draw line
```

```
        firstVec = null;
        Iterator<PVector> itr = curList.iterator();
        beginShape();
          while (itr.hasNext())
          {
            vec = itr.next();
            if(firstVec == null)
              firstVec = vec;
            // calc the screen pos
            context.convertRealWorldToProjective(vec,screenPos);
            vertex(screenPos.x,screenPos.y);
          }
        endShape();

        // draw current pos of the hand
        if(firstVec != null)
        {
          strokeWeight(8);
          context.convertRealWorldToProjective(firstVec,screenPos);
          point(screenPos.x,screenPos.y);
        }
        colorIndex++;
      }

    popStyle();
  }

}

void stop(){

  super.stop();
  output.flush();
  output.close();
}
```

Appendix FF

Phase III Code: Arduino

Day_1:

```
#include "LPD8806.h"
#include "SPI.h"

int nLeEDs = 52;

int dataPin = 2;
int clockPin = 3;

//variables for DOWN actuator button
static int press=0;
static int waiting=0;
int output1=6;
int valDown=0;

static int switch_mode_b=0;
static int switch_mode_f=0;

//variables for pushbuttons
int ReprimandPin = 7;   // choose the input pin (for a pushbutton)
int Reprimandval = 0;     // variable for reading the pin status

int TiltForwardPin = 8;   // choose the input pin (for a pushbutton)
int TiltForwardval = 0;     // variable for reading the pin status

int TiltBackPin = 9;   // choose the input pin (for a pushbutton)
int TiltBackval = 0;     // variable for reading the pin status

int BendInPin = 12;   // choose the input pin (for a pushbutton)
int BendInval = 0;     // variable for reading the pin status

int DownPin = 13;   // choose the input pin (for a pushbutton)
int Downval = 0;

char gesture_triggered; //Data received from serial port

LPD8806 strip = LPD8806(nLeEDs, dataPin, clockPin);


void setup() {

  pinMode(output1, OUTPUT);
  pinMode(ReprimandPin, INPUT);
  pinMode(TiltForwardPin, INPUT);
  pinMode(TiltBackPin, INPUT);
  pinMode(BendInPin, INPUT);
  pinMode(DownPin, INPUT);

  Serial.begin(9600);

  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();
}

void loop(){

  Reprimandval = digitalRead(ReprimandPin);
  TiltForwardval = digitalRead(TiltForwardPin);
  TiltBackval = digitalRead(TiltBackPin);
  BendInval = digitalRead(BendInPin);
  Downval = digitalRead(DownPin);

  if(Serial.available()){
    gesture_triggered=(char)Serial.read();
  }

  //If the Reparimand pushbutton is pressed, play the corresponding light sequence
  if(Reprimandval==LOW){
    //Serial.write('R');
    ReprimandWompWomp(80);
    Clear(80);
  }

//  if(BendInval==LOW){
//   BendIn(80);        // red, slow
//    BendIn(80);
//    Clear(80);
//  }

  //If the Tilt Forward or Tilt Back pushbuttons are pressed, play their corresponding light sequences
  //and send a signal to processing to play the corresponding sounds
  if(TiltForwardval==LOW){
    Serial.println(switch_mode_f);
    if(switch_mode_f==0){
      Serial.write('F');
      switch_mode_f++;
    }
    else if(switch_mode_f==1){
      Serial.println('flag');
    On(80);
    Clear(80);
    switch_mode_f++;
    }
    else{
      Serial.write('F');
      On(80);
      Clear(80);
      switch_mode_f=0;
    }
    delay(200);
  }


if(TiltBackval==LOW){
    if(switch_mode_b==0){
      Serial.write('B');
      switch_mode_b++;
    }
    else if(switch_mode_b==1){
    On(80);
    Clear(80);
    switch_mode_b++;
    }
    else{
      Serial.write('B');
      On(80);
      Clear(80);
      switch_mode_b=0;
    }
    delay(500);
  }

  if(BendInval==LOW){
    //Serial.write('R');
    BendIn(80);
    Clear(80);
  }

  if(Downval==LOW){
    Serial.write('D');
    delay(500);
  }


    //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  //Send signal to Processing to play the sound corresponding to the Down button
//  digitalWrite(output1, HIGH);
//  valDown=analogRead(A0);
//
//  float voltageDown=valDown*(0.5/1024.0);
//  //Serial.println(voltageDown);
//
//  if(voltageDown>0.4){
//    press++;
//    if(press==1){
//      Serial.write('D');
//      //Serial.println("Pressed!");
//    }
//    valDown=analogRead(A0);
//    voltageDown=valDown*(0.5/1024.0);
//    //Serial.println(press);
//    //Serial.println(voltageDown);
//
//  }
//
//   else if(voltageDown<0.4){
//    waiting++;
//    valDown=analogRead(A0);
//    voltageDown=valDown*(0.5/1024.0);
//    //Serial.println(voltageDown);
//    //Serial.println(voltageUp);
//  }
//  if(waiting>200){
//    press=0;
//    waiting=0;
//  }
//    //press=0;


  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if(gesture_triggered=='C'){
    dither(80);
    Clear(500);
  }

  else if(gesture_triggered=='E'){
    OnOff(80);
    Clear(80);
  }

  //I'm Thinking Lights
  else if(gesture_triggered=='T'){
    Thinking(80);
    Clear(1000);
  }

  else if(gesture_triggered=='G'){
    Half(80);
    Clear(500);
  }

  else if(gesture_triggered=='U'){
    OnOff(175);
    Clear(1000);
  }

  else if(gesture_triggered=='W'){
    dither(80);
    Go(80);
    Clear(1000);
  }




}
```

BendIn:

```
void BendIn(uint8_t wait) {
  int i, j, pos, dir;
  byte r, g, b;
  r=127;
  g=127;
  b=127;

  pos = (strip.numPixels()-1);
  dir = 1;

  for(i=(strip.numPixels()-1); i<((strip.numPixels()-1)*2); i++) {
    // Draw 5 pixels centered on pos.  setPixelColor() will clip
    // any pixels off the ends of the strip, no worries there.
    // we'll make the colors dimmer at the edges for a nice pulse
    // look
    strip.setPixelColor(pos - 2, strip.Color(r/4, g/4, b/4));
    strip.setPixelColor(pos - 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos, strip.Color(r, g, b));
    strip.setPixelColor(pos + 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos + 2, strip.Color(r/4, g/4, b/4));

    strip.show();
    delay(wait);
    // If we wanted to be sneaky we could erase just the tail end
    // pixel, but it's much easier just to erase the whole thing
    // and draw a new one next time.
    for(j=-2; j<= 2; j++)
        strip.setPixelColor(pos+j, strip.Color(0,0,0));
    // Bounce off ends of strip
    pos += dir;
    if(pos < 0) {
      pos = 1;
      dir = -dir;
    } else if(pos >= strip.numPixels()) {
      pos = strip.numPixels() - 2;
      dir = -dir;
    }
  }
}
```

Clear:

```
void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}
```

Dither:

```
 void dither(uint8_t wait) {

  // Determine highest bit needed to represent pixel index
  int hiBit = 0;
  int n = strip.numPixels() - 1;
  byte r, g, b;
  r=127;
  g=127;
  b=127;

  for(int bit=1; bit < 0x8000; bit <<= 1) {
    if(n & bit) hiBit = bit;
  }

  int bit, reverse;
  for(int i=0; i<(hiBit << 1); i++) {
    // Reverse the bits in i to create ordered dither:
    reverse = 0;
    for(bit=1; bit <= hiBit; bit <<= 1) {
      reverse <<= 1;
      if(i & bit) reverse |= 1;
    }
    strip.setPixelColor(reverse, r, g, b);
    strip.show();
    delay(wait);
  }
  delay(250); // Hold image for 1/4 sec
```

```
}
```

Go:

```
void Go(uint8_t wait) {

  // Determine highest bit needed to represent pixel index
  int hiBit = 0;
  int n = strip.numPixels() - 1;
  byte r, g, b;
  r=0;
  g=0;
  b=0;

  for(int bit=1; bit < 0x8000; bit <<= 1) {
    if(n & bit) hiBit = bit;
  }

  int bit, reverse;
  for(int i=0; i<(hiBit << 1); i++) {
    // Reverse the bits in i to create ordered dither:
    reverse = 0;
    for(bit=1; bit <= hiBit; bit <<= 1) {
      reverse <<= 1;
      if(i & bit) reverse |= 1;
    }
    strip.setPixelColor(reverse, r, g, b);
    strip.show();
    delay(wait);
  }
  delay(250); // Hold image for 1/4 sec
}
```

Half:

```
void Half(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x<16){
         if (i>16){
        r=127;
        g=127;
        b=127;
         }

         else{
          r=0;
          g=0;
          b=0;
         }
       }

       else {
         if (i<=16){
        r=127;
        g=127;
        b=127;
         }
         else{
          r=0;
          g=0;
          b=0;
         }
       }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }
```

On:

```
void On(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x>=0){
        r=127;
        g=127;
        b=127;}

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }
```

OnOff:

```
void OnOff(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<strip.numPixels(); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {

      if(x%2==0){
        r=32;
        g=32;
        b=32;}

      else if(x%2==1){
          r=0;
          g=0;
          b=0;
      }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
  }
}
```

ReprimandWompWomp:

```
void ReprimandWompWomp(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<68; x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {

      if(x<15){
        r=127;
        g=127;
        b=127;}

      else if(x>15 && x<23){
          r=0;
          g=0;
          b=0;
      }

      else if(x>23 && x<61){
         r=127;
         g=127;
         b=127;}

      else if(x>61){
          r=0;
          g=0;
          b=0;
      }



      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
  }
}
```

Thinking:

```
void Thinking(uint8_t wait) {
  int i, j, pos, dir;
  byte r, g, b;
  r=127;
  g=127;
  b=127;

  pos = 0;
  dir = 1;

  for(i=0; i<((strip.numPixels()-1) * 3); i++) {
    // Draw 5 pixels centered on pos.  setPixelColor() will clip
    // any pixels off the ends of the strip, no worries there.
    // we'll make the colors dimmer at the edges for a nice pulse
    // look
    strip.setPixelColor(pos - 2, strip.Color(r/4, g/4, b/4));
    strip.setPixelColor(pos - 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos, strip.Color(r, g, b));
    strip.setPixelColor(pos + 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos + 2, strip.Color(r/4, g/4, b/4));

    strip.show();
    delay(wait);
    // If we wanted to be sneaky we could erase just the tail end
    // pixel, but it's much easier just to erase the whole thing
    // and draw a new one next time.
    for(j=-2; j<= 2; j++)
        strip.setPixelColor(pos+j, strip.Color(0,0,0));
    // Bounce off ends of strip
    pos += dir;
    if(pos < 0) {
      pos = 1;
      dir = -dir;
    } else if(pos >= strip.numPixels()) {
      pos = strip.numPixels() - 2;
      dir = -dir;
    }
  }
}
```

TurnOff:

```
void TurnOff(uint8_t wait) {
  for (int i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, 0, 0, 0);
    }
      strip.show();
}
```

Appendix GG

Phase III Code: Processing

Day 1:

```
/* --------------------------------------------------------------------------
 * SimpleOpenNI NITE Hands
 * --------------------------------------------------------------------------
 * Processing Wrapper for the OpenNI/Kinect library
 * http://code.google.com/p/simple-openni
 * --------------------------------------------------------------------------
 * prog:  Max Rheiner / Interaction Design / zhdk / http://iad.zhdk.ch/
 * date:  03/19/2011 (m/d/y)
 * ----------------------------------------------------------------------------
 * This example works with multiple hands, to enable mutliple hand change
 * the ini file in /usr/etc/primesense/XnVHandGenerator/Nite.ini:
 *  [HandTrackerManager]
 *  AllowMultipleHands=1
 *  TrackAdditionalHands=1
 * on Windows you can find the file at:
 *  C:\Program Files (x86)\Prime Sense\NITE\Hands\Data\Nite.ini
 * ----------------------------------------------------------------------------
 */

import SimpleOpenNI.*;
import java.util.Iterator;
import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Serial port; // Create object from Serial class
int button_trigger; // Data received from the serial port



///Order of Operations
int stop = 2;
int cantdo=5;
int come=0;
int emergency=3;
int imthinking=6;
int go=1;
int understand=4;
int somethingway=7;
int numberofgestures=8;

static int switch_mode_g=0;
static int switch_mode_u=0;
static int switch_mode_w=0;

Minim minim1;
Minim minim2;
Minim minim3;
Minim minim4;
Minim minim5;
Minim minim6;
Minim minim7;
Minim minim8;
```

```
AudioPlayer downsound1;
AudioPlayer Stop1;
AudioPlayer CantDo2;
AudioPlayer TiltForward2;
AudioPlayer TiltBack2;
AudioPlayer Go2;
AudioPlayer DoNotUnderstand2;
AudioPlayer SomethingInTheWay1;

static int time=0;
int check=0;
int checky=0;
float xcoord0, xcoord99, diff;

SimpleOpenNI      context;

// NITE
XnVSessionManager sessionManager;
XnVFlowRouter     flowRouter;

PointDrawer       pointDrawer;

PrintWriter output;

void setup()
{

  println(Serial.list()); //This shows the various serial port options
  String portName = Serial.list()[0]; //The serial port should match the one the Arduino is hooked to
  port = new Serial(this, portName, 9600); //Establish the connection rate

  minim1 = new Minim(this);
  minim2 = new Minim(this);
  minim3 = new Minim(this);
  minim4 = new Minim(this);
  minim5 = new Minim(this);
  minim6 = new Minim(this);
  minim7 = new Minim(this);
  minim8 = new Minim(this);
  downsound1=minim1.loadFile("Down1.wav");
  Stop1=minim2.loadFile("Stop1.wav");
  CantDo2=minim3.loadFile("CantDo2.wav");
  TiltForward2=minim4.loadFile("TiltForward2.wav");
  TiltBack2=minim5.loadFile("TiltBack2.wav");
  Go2=minim6.loadFile("Go2.wav");
  DoNotUnderstand2=minim7.loadFile("DoNotUnderstand2.wav");
  SomethingInTheWay1=minim8.loadFile("SomethingInTheWay1.wav");

  //output = createWriter("out.txt");
  context = new SimpleOpenNI(this);

  // mirror is by default enabled
  context.setMirror(true);

  // enable depthMap generation
  if(context.enableDepth() == false)
  {
    println("Can't open the depthMap, maybe the camera is not connected!");
    exit();
    return;
  }
  // enable the hands + gesture
  context.enableGesture();
  context.enableHands();

  // setup NITE
  sessionManager = context.createSessionManager("Click,Wave", "RaiseHand");

  pointDrawer = new PointDrawer();
  flowRouter = new XnVFlowRouter();
  flowRouter.SetActive(pointDrawer);

  sessionManager.AddListener(flowRouter);

  size(context.depthWidth(), context.depthHeight());
  smooth();
}

void draw()
{

  if (port.available() > 0) {
    button_trigger = port.read();
    //println(val);


  if(button_trigger=='D'){
    //println("Play Sound!");
    downsound1.play();
    downsound1.rewind();
  }

  else if(button_trigger=='F'){
    //println("Play Forward!");
    TiltForward2.play();
    TiltForward2.rewind();
  }

  else if(button_trigger=='B'){
    TiltBack2.play();
    TiltBack2.rewind();
  }
  }


  background(200,0,0);
  // update the cam
  context.update();

  // update nite
  context.update(sessionManager);

  // draw depthImageMap
  image(context.depthImage(),0,0);

  // draw the list
  pointDrawer.draw();
}
```

Point Drawer:

```
// PointDrawer keeps track of the handpoints

class PointDrawer extends XnVPointControl
{
  HashMap    _pointLists;
  int        _maxPoints;
  color[]    _colorList = { color(255,0,0),color(0,255,0),color(0,0,255),color(255,255,0)};

  public PointDrawer()
  {
    _maxPoints = 30;
    _pointLists = new HashMap();
  }

  public void OnPointCreate(XnVHandPointContext cxt)
  {
    // create a new list
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));

    println("OnPointCreate, handId: " + cxt.getNID());
  }

  public void OnPointUpdate(XnVHandPointContext cxt)
  {
    //println("OnPointUpdate " + cxt.getPtPosition());
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));
  }

  public void OnPointDestroy(long nID)
  {
    //println("OnPointDestroy, handId: " + nID);

    // remove list
    if(_pointLists.containsKey(nID))
       _pointLists.remove(nID);
  }

  public ArrayList getPointList(long handId)
  {
    ArrayList curList;
    if(_pointLists.containsKey(handId))
      curList = (ArrayList)_pointLists.get(handId);
    else
    {
      curList = new ArrayList(_maxPoints);
      _pointLists.put(handId,curList);
    }
    return curList;
  }

  public void addPoint(long handId,PVector handPoint)
  {
    ArrayList curList = getPointList(handId);

    curList.add(0,handPoint);
    if(curList.size() > _maxPoints)
      curList.remove(curList.size() - 1);
```

```
//If hand gesture detected, activate lights from Arduino
      //output.println(handPoint);
      if(handPoint.x>150){
        xcoord0=handPoint.x;
        //output.println("START");
        port.write('N'); //send an N to deactivate light sequence
        check=0;

      }
      else if(handPoint.x<-200){
        checky=0;
          if(check==0){
          //output.println("TRIGGER");

          //port.write sends signal to Arduino to activate corresponding light sequence
          //.play plays the corresponding sound from Processing

          //Stop ****Sound Only****
          if(time==stop){
            Stop1.play();
            Stop1.rewind();
          }

          //Can't Do ****Sound Only****
          else if(time==cantdo){
            CantDo2.play();
            CantDo2.rewind();
          }

          //Come ****Lights Only****
          else if(time==come){
            port.write('C'); //send an T to activate the second light sequence
          }

          //Emergency ****Lights Only****
          else if(time==emergency){
            port.write('E'); //send an T to activate the second light sequence
          }

          //I'm Thinking ****Lights Only****
          else if(time==imthinking){
            port.write('T'); //send an T to activate the second light sequence
          }

          //Go ****Lights and Sound****
          else if(time==go){
            if(switch_mode_g==0){
            port.write('G'); //send an T to activate the second light sequence
            switch_mode_g++;
            }
            else if(switch_mode_g==1){
            Go2.play();
            Go2.rewind();
            switch_mode_g++;
            }
            else{
              port.write('G');
              Go2.play();
              Go2.rewind();
              switch_mode_g=0;
            }

          }

          //Do Not Understand ****Lights and Sound****
          else if(time==understand){
            if(switch_mode_u==0){
              port.write('U'); //send an T to activate the second light sequence
              switch_mode_u++;
            }
            else if(switch_mode_u==1){
              DoNotUnderstand2.play();
              DoNotUnderstand2.rewind();
              switch_mode_u++;
            }
            else{
              port.write('U');
              DoNotUnderstand2.play();
              DoNotUnderstand2.rewind();
              switch_mode_u=0;
            }
          }

          //Something in the Way ****Lights and Sound****
          else if(time==somethingway){
            if(switch_mode_w==0){
              port.write('W'); //send an T to activate the second light sequence
              switch_mode_w++;
            }
            else if(switch_mode_w==1){
              SomethingInTheWay1.play();
              SomethingInTheWay1.rewind();
              switch_mode_w++;
            }
            else{
              port.write('W');
              SomethingInTheWay1.play();
              SomethingInTheWay1.rewind();
              switch_mode_w=0;
            }
          }

        }
        check=1;
        port.write('N');

        }

       else if(handPoint.y>500){
         println("Reached");
         if(checky==0){
           time++;
           checky=1;
         }
          if(time==numberofgestures)  time=0;
       }
      println(time);

    }

    public void draw()
  {
    if(_pointLists.size() <= 0)
      return;

    pushStyle();
      noFill();

      PVector vec;
      PVector firstVec;
      PVector screenPos = new PVector();
      int colorIndex=0;

      // draw the hand lists
      Iterator<Map.Entry> itrList = _pointLists.entrySet().iterator();
      while(itrList.hasNext())
      {
        strokeWeight(2);
        stroke(_colorList[colorIndex % (_colorList.length - 1)]);

        ArrayList curList = (ArrayList)itrList.next().getValue();

        // draw line
        firstVec = null;
        Iterator<PVector> itr = curList.iterator();
        beginShape();
          while (itr.hasNext())
          {
            vec = itr.next();
            if(firstVec == null)
              firstVec = vec;
            // calc the screen pos
            context.convertRealWorldToProjective(vec,screenPos);
            vertex(screenPos.x,screenPos.y);
          }
        endShape();

        // draw current pos of the hand
        if(firstVec != null)
        {
          strokeWeight(8);
          context.convertRealWorldToProjective(firstVec,screenPos);
          point(screenPos.x,screenPos.y);
        }
        colorIndex++;
      }

    popStyle();
  }

}
```

keyPressed:

```
void keyPressed()
{
  switch(key)
  {
  case 'e':
    // end sessions
    sessionManager.EndSession();
    println("end session");
    break;
  }
}
```

onStartSession:

```
////////////////////////////////////////////////////////////////////////////////////////////////////
// session callbacks

void onStartSession(PVector pos)
{
  println("onStartSession: " + pos);
}

void onEndSession()
{
  println("onEndSession: ");
}

void onFocusSession(String strFocus,PVector pos,float progress)
{
  println("onFocusSession: focus=" + strFocus + ",pos=" + pos + ",progress=" + progress);
}

////////////////////////////////////////////////////////////////////////////////////////////////////
```

stop:

```
void stop(){

  super.stop();
  output.flush();
  output.close();
}
```

Appendix HH

Usability Study Code: Processing

PG_altered

```
/* --------------------------------------------------------------------------
 * SimpleOpenNI NITE Hands
 * --------------------------------------------------------------------------
 * Processing Wrapper for the OpenNI/Kinect library
 * http://code.google.com/p/simple-openni
 * --------------------------------------------------------------------------
 * prog:  Max Rheiner / Interaction Design / zhdk / http://iad.zhdk.ch/
 * date:  03/19/2011 (m/d/y)
 * ----------------------------------------------------------------------------
 * This example works with multiple hands, to enable mutliple hand change
 * the ini file in /usr/etc/primesense/XnVHandGenerator/Nite.ini:
 *  [HandTrackerManager]
 *  AllowMultipleHands=1
 *  TrackAdditionalHands=1
 * on Windows you can find the file at:
 *  C:\Program Files (x86)\Prime Sense\NITE\Hands\Data\Nite.ini
 * ----------------------------------------------------------------------------
 */

import SimpleOpenNI.*;
import java.util.Iterator;
import java.util.Map;
import ddf.minim.*;
import processing.serial.*;

Serial port; // Create object from Serial class
int button_trigger; // Data received from the serial port
int startsession=0;



///Order of Operations
int stop = 2;
int cantdo=5;
int come=0;
int emergency=3;
int imthinking=6;
int go=1;
int understand=4;
int somethingway=7;
int numberofgestures=8;

static int switch_mode_g=0;
static int switch_mode_u=0;
static int switch_mode_w=0;

int xboundary=100;
int yboundary=100;
int yboundaryupdown=0;
int yupdownmax=250;
int yboundaryrest=200;
int gesturexUB=150;
int gesturexLB=-200;
int starttimegesture, endtimegesture, starttimerest, endtimerest, starttimetherapy, endtimetherapy;
int timerequirement=4000;
int timediffgesture, timediffrest, timedifftherapy;
int br=30; //br=broundary range
int countchangegesture=0;

static int updown=0;
static int actuatoron=0;
static int first=1;
float ycoord0;
static int count=0;

Minim minim1;
Minim minim2;
Minim minim3;
Minim minim4;
Minim minim5;
Minim minim6;
Minim minim7;
Minim minim8;
Minim minim9;
AudioPlayer downsound1;
AudioPlayer Stop1;
AudioPlayer CantDo2;
AudioPlayer TiltForward2;
AudioPlayer TiltBack2;
AudioPlayer Go2;
AudioPlayer DoNotUnderstand2;
AudioPlayer SomethingInTheWay1;
AudioPlayer learn;

static int time=0;
int gesture=0;
int therapygesture=0;
int halfcomplete=0;
int restgesture=0;
static int restgestureacting=0;
int changegesture=0;
float xcoord0, xcoord99, diff;
static int gesturemode=0; //Allows to change between Do Not Understand and Up Movement for UP/DOWN
Gesture
int incident=0;

SimpleOpenNI      context;

// NITE
XnVSessionManager sessionManager;
XnVFlowRouter     flowRouter;

PointDrawer       pointDrawer;

PrintWriter output;

void setup()
{

  println(Serial.list()); //This shows the various serial port options
  String portName = Serial.list()[6]; //The serial port should match the one the Arduino is hooked to
  port = new Serial(this, portName, 9600); //Establish the connection rate

  minim1 = new Minim(this);
  minim2 = new Minim(this);
  minim3 = new Minim(this);
  minim4 = new Minim(this);
  minim5 = new Minim(this);
  minim6 = new Minim(this);
  minim7 = new Minim(this);
  minim8 = new Minim(this);
  minim9 = new Minim(this);
  downsound1=minim1.loadFile("Down1.wav");
  Stop1=minim2.loadFile("Stop1.wav");
  CantDo2=minim3.loadFile("CantDo2.wav");
  TiltForward2=minim4.loadFile("TiltForward2.wav");
  TiltBack2=minim5.loadFile("TiltBack2.wav");
  Go2=minim6.loadFile("Go2.wav");
  DoNotUnderstand2=minim7.loadFile("DoNotUnderstand2.wav");
  SomethingInTheWay1=minim8.loadFile("SomethingInTheWay1.wav");
  learn=minim9.loadFile("learn.wav");

  //output = createWriter("out.txt");
  context = new SimpleOpenNI(this);

  // mirror is by default enabled
  context.setMirror(true);

  // enable depthMap generation
  if(context.enableDepth() == false)
  {
    println("Can't open the depthMap, maybe the camera is not connected!");
    exit();
    return;
  }

  // enable the hands + gesture
  context.enableGesture();
  context.enableHands();

  // setup NITE
  sessionManager = context.createSessionManager("Click,Wave", "RaiseHand");

  pointDrawer = new PointDrawer();
  flowRouter = new XnVFlowRouter();
  flowRouter.SetActive(pointDrawer);

  sessionManager.AddListener(flowRouter);

  size(context.depthWidth(), context.depthHeight());
  smooth();
}

void draw()
{

  if (port.available() > 0) {
   button_trigger = port.read();
   //println(val);


  if(button_trigger=='D'){
   //println("Play Sound!");
   //downsound1.play();
   //downsound1.rewind();
  }

  else if(button_trigger=='F'){
    //println("Play Forward!");
    //TiltForward2.play();
    //TiltForward2.rewind();
  }

  else if(button_trigger=='B'){
    //TiltBack2.play();
    //TiltBack2.rewind();
  }
  }


  background(200,0,0);
  // update the cam
  context.update();

  // update nite
  context.update(sessionManager);

  // draw depthImageMap
  image(context.depthImage(),0,0);

  // draw the list
  pointDrawer.draw();
}
```

PointDrawer

```
// PointDrawer keeps track of the handpoints

class PointDrawer extends XnVPointControl
{
  HashMap    _pointLists;
  int        _maxPoints;
  color[]    _colorList = { color(255,0,0),color(0,255,0),color(0,0,255),color(255,255,0)};

  public PointDrawer()
  {
    _maxPoints = 30;
    _pointLists = new HashMap();
  }

  public void OnPointCreate(XnVHandPointContext cxt)
  {
    // create a new list
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));

    println("OnPointCreate, handId: " + cxt.getNID());
  }

  public void OnPointUpdate(XnVHandPointContext cxt)
  {
    //println("OnPointUpdate " + cxt.getPtPosition());
    addPoint(cxt.getNID(),new
PVector(cxt.getPtPosition().getX(),cxt.getPtPosition().getY(),cxt.getPtPosition().getZ()));
  }
```

```
public void OnPointDestroy(long nID)
{
 //println("OnPointDestroy, handId: " + nID);

 // remove list
 if(_pointLists.containsKey(nID))
    _pointLists.remove(nID);
}

public ArrayList getPointList(long handId)
{
 ArrayList curList;
 if(_pointLists.containsKey(handId))
   curList = (ArrayList)_pointLists.get(handId);
 else
 {
   curList = new ArrayList(_maxPoints);
   _pointLists.put(handId,curList);
 }
 return curList;
}

public void addPoint(long handId,PVector handPoint)
{
 ArrayList curList = getPointList(handId);

 curList.add(0,handPoint);
 if(curList.size() > _maxPoints)
   curList.remove(curList.size() - 1);

   //println(handPoint.y);
   //*if hand is in top right quadrant*\\
   if(handPoint.x>gesturexUB && handPoint.y>yboundary){

     //*Add in boundary to rest gesture if it's not completed*\\
     if(handPoint.x>(gesturexUB+br))  gesture=0;

     restgesture=0;
     restgestureacting=0;

     //*if the hand is moving to make the general gesture*\\
     if(gesture==0){
       if(handPoint.x<(gesturexUB+br)){
         //*record the time the gesture began and set the trigger for the general gesture*\\
         starttimegesture=millis();
         gesture=1;
         println("Start Gesture");
       }
     }
     //if gesture was triggered by accident, reset
     //else  gesture=0;

     ////////////////////////END BEGIN GESTURE//////////////////////



     ////////////////////////START THERAPY MOVEMENT/////////////////
     //*if the hand is moving ot make the therapy gesture*\\
     if(therapygesture==0 && halfcomplete==0){
       if(handPoint.y<(yboundary+br)){
         println("Start Therapy");
```

```
          for(int c=0; c<50; c++)  port.write('R');



          //*record the time the therapy gesture began and set the trigger for the therapy gesture*\\
          starttimetherapy=millis();
          therapygesture=1;
        }
        port.write('O');
        port.write('Q');
      }
      //if therapy gesture was triggered by accident, reset
      //else if(therapygesture!=2)  therapygesture=0;


      //*if the second half of the therapy movement is in progress*\\
      if(halfcomplete==1){
        if(handPoint.y<(yboundary+br)){
          //*record the time the second swipe began and set the trigger for the therapy gesture*\\
          starttimetherapy=millis();
          timedifftherapy=starttimetherapy-endtimetherapy;
        }
        //*if the gesture wasn't completed in time, don't do anything and reset the therapy gesture*\\
        else timedifftherapy=timerequirement+1;

        if(timedifftherapy<timerequirement){
          println("Therapy Two Begin");
          therapygesture=2;
          println(therapygesture);
          halfcomplete=0;
        }
        else if(timedifftherapy>timerequirement && handPoint.y>(yboundary+br)){
          println("Alert");
          therapygesture=0;
          halfcomplete=0;
        }
      }
      if(therapygesture==2){
        if(handPoint.y<(yboundary+br)){
          println("Start Therapy Two");
          //*record the time the second swipe began and set the trigger for the therapy gesture*\\
          starttimetherapy=millis();

        }
      }

    }

    //////////////////////END BEGIN THERAPY///////////////

    /////////////////////BEGIN END GESTURE////////////////

    //*if the hand has reached the top left quadrant*\\
    else if(handPoint.x<gesturexLB && handPoint.y>yboundary && restgestureacting==0){
      //*Add in boundary to rest gesture if it's not completed*\\
//        if(handPoint.x<(gesturexLB-br)){
//          gesture=0;
//        }


      therapygesture=0;
```

```
halfcomplete=0;

if(handPoint.x>(gesturexLB-br)){
  endtimegesture=millis();
  timediffgesture=endtimegesture-starttimegesture;
}
else timediffgesture=timerequirement+1;
if(timediffgesture<timerequirement){
  println("End Gesture");
  println(gesture);
  changegesture=0;
  if(gesture==1){

  //*port.write sends signal to Arduino to activate corresponding light sequence*\\
  //*.play plays the corresponding sound from Processing*\\

  //Stop ****Sound Only****
  if(time==stop){
    println("Stop Sending");
    port.write('S');
  }

  //Can't Do ****Sound Only****
  else if(time==cantdo){
    port.write('X');
  }

  //Come ****Lights Only****
  else if(time==come){
    port.write('C');
  }

  //Emergency ****Lights Only****
  else if(time==emergency){
    port.write('E');
  }

  //I'm Thinking ****Lights Only****
  else if(time==imthinking){
    port.write('T');
  }

  //Go ****Lights and Sound****
  else if(time==go){
    port.write('G');
  }

  //Do Not Understand ****Lights and Sound****
  else if(time==understand){
    port.write('N');
  }

  //Something in the Way ****Lights and Sound****
  else if(time==somethingway){
    port.write('W');
  }

}
gesture=0;
port.write('Q');
```

```
       }
       else  gesture=0;
      }

      //////////////////////END END GESTURE//////////////////

      //////////////////////BEGIN CHANGE GESTURE MODE///////////////

       else if(handPoint.y>500){

         if(countchangegesture>100){
           if(changegesture==0){
             time++;
             changegesture=1;
             println("Gesture Changed");
             println(time);
           }
           if(time==numberofgestures)  time=0;
         }
         countchangegesture++;
       }

//         println("Reached");
//         if(changegesture==0){
//           time++;
//           changegesture=1;
//           println(time);
//         }
//          if(time==numberofgestures)  time=0;
//       }

       //////////////////////END CHANGE GESTURE MODE/////////////////


      /////////////CODE FOR SECOND HALF OF THERAPY MOVEMENT////////////
      //*if hand reaches bottom right quadrant*\\
      else if(handPoint.y<-yboundary && handPoint.x>xboundary){

        //*record the end time of the downward swipe of the first therapy movement*\\
        //*or record the start time of the upward swipe to start the second therapy movement*\\
        if(handPoint.y>(-yboundary-br)){
            endtimetherapy=millis();
            timedifftherapy=endtimetherapy-starttimetherapy;
        }
        //*if the hand settles in the quadrant, don't do anything*\\
        else  timedifftherapy=timerequirement+1;

        //*if the therapy gesture has been triggered, flag that it is half completed*\\
        if(therapygesture==1 && halfcomplete==0){
          println("Half Complete");
          if(timedifftherapy<timerequirement){
            println("End First Therapy");
            halfcomplete=1;
            println(halfcomplete);
          }
          else {
            therapygesture=0;
            halfcomplete=0;
          }
        }
```

```
    //*if gesture is complete, engage social interaction*\\
    else if(therapygesture==2){
      if(timedifftherapy<timerequirement){
        println("End Therapy");
        for(int i=0; i<200; i++){
          port.write('D');
        }
        port.write('Q');
        //learn.play();
        //learn.rewind();
        halfcomplete=0;
        therapygesture=0;
      }
      else{
        therapygesture=0;
        halfcomplete=0;
      }
    }

    //*if the hand settles in the quadrant, don't do anything*\\
    else  timedifftherapy=timerequirement+1;
  }

  ////////////CODE FOR REST MOVEMENT///////////

  //*hand is in bottom left quadrant*\\
  else if(handPoint.y<-yboundaryrest && handPoint.x<-xboundary){
    therapygesture=0;
    halfcomplete=0;
    gesture=0;

    //*Add in boundaries to rest gesture if they're not completed*\\
    if(handPoint.y<(-yboundaryrest-br)){
      restgesture=0;
      restgestureacting=0;
    }

    //*hand has started making the gesture*\\
    if(restgesture==0){
      if(handPoint.y>(-yboundaryrest-br)){
        println("Start Rest");
        for(int c=0; c<50; c++)  port.write('R');


        //*set the flag that will keep the up/down movement from being triggered*\\
        restgestureacting=1;
        //*record the time the gesture started*\\
        starttimerest=millis();
        //*set the flag saying that the rest gesture has begun*\\
        restgesture=1;
      }
      port.write('O');
      port.write('Q');
    }
    //if the gesture was triggered by accident or the user didn't follow through, restart the gesture
//      else{
//        restgesture=0;
//        restgestureacting=0;
//      }
  }
```

```
//*if the hand has reached the top left quadrant*\\
else if(handPoint.y>yboundaryrest && handPoint.x<-xboundary){

  println("Above Rest boundary!");
  therapygesture=0;
  halfcomplete=0;

  //*if the rest gesture was in progress and it was completed in the required time*\\
  if(restgesture==1){
    println("Trying to Rest");
    if(handPoint.y<(yboundaryrest+br)){
      println("Almost End Rest");
      //*record the time the gesture started*\\
      endtimerest=millis();
      timediffrest=endtimerest-starttimerest;
    }
    else  timediffrest=timerequirement+1;

    if(timediffrest<timerequirement){
      println("End Rest");
      restgestureacting=0;
      //*complete the social interaction and reset the gesture*\\
      for(int i=0; i<200; i++){
        port.write('U');
      }
      port.write('Q');
      //learn.play();
      //learn.rewind();
      restgesture=0;
    }
    //*if the gesture wasn't complete in the time given or if the gesture hadn't been started, reset the gesture*\\
    else{
      restgesture=0;
      restgestureacting=0;
    }
  }
}


////////////////////CODE FOR UP/DOWN MOVEMENTS/////////////////////

//*If hand is between the start up/down boundaries and the rest gesture is not currently in action*\\
if(handPoint.y>yboundaryupdown && handPoint.y<yupdownmax && handPoint.x<-xboundary){
  therapygesture=0;
  halfcomplete=0;
  gesture=0;

  if(count>50){
    println("Triggering Up/Down");
    for(int c=0; c<50; c++)  port.write('R');



    restgesture=0;
    restgestureacting=0;

    if(first==1){
      ycoord0=handPoint.y;
      actuatoron=1;
    }

  first++;
  count=0;
  }
  port.write('O');
  port.write('Q');
count++;
}

//*If the hand is above the maxium boundary and the rest gesture is not in action*\\
else if(handPoint.y>yupdownmax+100 && handPoint.x<-xboundary && restgestureacting==0){
  therapygesture=0;
  halfcomplete=0;


  updown=1;
  actuatoron=0;
  first=1;

  port.write('Q');
  //println("SEND DON'T MOVE COMMAND");

}

//*If the hand is below the minimum boundary and the rest gesture is not in action*\\
else if(handPoint.y<yboundaryupdown-100 && handPoint.x<-xboundary && restgestureacting==0){
  therapygesture=0;
  halfcomplete=0;
  gesture=0;


  //println("Stop");
  updown=0;
  actuatoron=0;
  first=1;

  port.write('Q');
  //println("SEND DON'T MOVE COMMAND");

}

//*If the hand is moving upward and the rest gesture is not in action*\\
else if(handPoint.y-ycoord0 < 0 && handPoint.x<-xboundary && restgestureacting==0){
  therapygesture=0;
  halfcomplete=0;
  gesture=0;


  updown=0;
  if(actuatoron==1){


    port.write('U');
    println("SENDING DOWN COMMAND");
    }

}

//*If the hadn is moving downward and the rest gesture is not in action*\\
    else if(handPoint.y-ycoord0 > 0 && handPoint.y>yboundaryupdown && handPoint.x<-xboundary &&
restgestureacting==0){
      therapygesture=0;
      halfcomplete=0;
      gesture=0;

      updown=1;
      if(actuatoron==1){


        port.write('D'); //send an T to activate the second light sequence
        println("SENDING UP COMMAND");
        }

    }

 }

 public void draw()
 {
   if(_pointLists.size() <= 0)
      return;

   pushStyle();
      noFill();

      PVector vec;
      PVector firstVec;
      PVector screenPos = new PVector();
      int colorIndex=0;

      // draw the hand lists
      Iterator<Map.Entry> itrList = _pointLists.entrySet().iterator();
      while(itrList.hasNext())
      {
        strokeWeight(2);
        stroke(_colorList[colorIndex % (_colorList.length - 1)]);

        ArrayList curList = (ArrayList)itrList.next().getValue();

        // draw line
        firstVec = null;
        Iterator<PVector> itr = curList.iterator();
        beginShape();
          while (itr.hasNext())
          {
            vec = itr.next();
            if(firstVec == null)
              firstVec = vec;
            // calc the screen pos
            context.convertRealWorldToProjective(vec,screenPos);
            vertex(screenPos.x,screenPos.y);
          }
        endShape();

        // draw current pos of the hand
        if(firstVec != null)
        {
          strokeWeight(8);
          context.convertRealWorldToProjective(firstVec,screenPos);
          point(screenPos.x,screenPos.y);
        }
        colorIndex++;
      }

    popStyle();
  }

}
```

keyPressed

```
void keyPressed()
{
  switch(key)
  {
  case 'e':
    // end sessions
    sessionManager.EndSession();
    println("end session");
    break;
  }
}
```

onStartSession

```
/////////////////////////////////////////////////////////////////////////////////////////////////
// session callbacks

void onStartSession(PVector pos)
{
  println("onStartSession: " + pos);
  for(int c=0; c<50; c++)  port.write('L');


}

void onEndSession()
{
  println("onEndSession: ");

}

void onFocusSession(String strFocus,PVector pos,float progress)
{
  port.write('O');
  port.write('Q');
  println("onFocusSession: focus=" + strFocus + ",pos=" + pos + ",progress=" + progress);

}


/////////////////////////////////////////////////////////////////////////////////////////////////
```

stop

```
void stop(){

  super.stop();
  output.flush();
  output.close();
}
```

Appendix II

Usability Study Code: Arduino

Arduino_1

```
#include "LPD8806.h"
#include "SPI.h"
#include <SdFat.h>
#include <SdFatUtil.h>
#include <SFEMP3Shield.h>
#include <Wire.h>

//Required for MP3 Player
SdFat sd;
SFEMP3Shield MP3player;
int trackdelay = 4000;
//int track;

void setup() {

  //Serial.begin(9600);

  //start the shield
  sd.begin(SD_SEL, SPI_HALF_SPEED);
  MP3player.begin();
  MP3player.setVolume(0, 0); // commit new volume

  //Read data sent from Arduino 2 telling which sound to play
  Wire.begin(4);                // join i2c bus with address #4
  Wire.onReceive(receiveEvent); // register event
  receiveEvent(2);

}

void loop(){
//  MP3player.playTrack(1);
//  delay(7000);
//    Serial.print("Got Signal");
  //receiveEvent(2);
  //MP3player.playTrack(x);

}
```

receiveEvent

```
// function that executes whenever data is received from master
// this function is registered as an event, see setup()
void receiveEvent(int howMany)
{
  while(1 < Wire.available()) // loop through all but the last
  {
    char c = Wire.read(); // receive byte as a character
    Serial.print(c);         // print the character
  }
  int x = Wire.read();    // receive byte as an integer
  Serial.println(x);         // print the integer
  MP3player.playTrack(x);
}
```

Arduino_2_Final

```
#include "LPD8806.h"
#include "SPI.h"
#include <SdFat.h>
#include <SdFatUtil.h>
#include <SFEMP3Shield.h>
#include <Wire.h>

//Required for Up/Down Actuator Control of Relays
int MotorRelay = 13;
int OnOffRelay = 12;
int UserMicroRelay1 = 11;
int UserMicroRelay2 = 10;
int val1 = 0;
int val2 = 0;
int val3 = 0;
int microtakeover = 0;
int MicroOnOff = 0;
int UpDown = 0;
char val;

int processing=0;

int forwardincident=0;
int backincident=0;

static int enteredDown=0;
static int enteredUp=0;

//Required for Light Strips
int nLeEDs = 52;
int dataPin = 4;
int clockPin = 5;
LPD8806 strip = LPD8806(52, dataPin, clockPin);

//variables for UP and DOWN actuator buttons
static int pressUp=0;
static int pressDown=0;
static int waiting=0;
int output1=6;
int input1=7;
int valDown=0;
int valUp=0;

static int countDown=0;
static int countUp=0;

static int switch_mode_b=0;
static int switch_mode_f=0;
static int switch_mode_w=0;
static int switch_mode_u=0;
static int switch_mode_g=0;


//variables for buttons
int TiltForwardPin = 2;   // choose the input pin (for a pushbutton)
int TiltForwardval = 0;     // variable for reading the pin status

int TiltBackPin = 3;   // choose the input pin (for a pushbutton)
int TiltBackval = 0;     // variable for reading the pin status

int UpPin = 0;   // choose the input pin (for a pushbutton)
int Upval = 0;     // variable for reading the pin status

int DownPin = 1;   // choose the input pin (for a pushbutton)
int Downval = 0;

char serial_signal; //Data received from serial port


void setup() {

  Wire.begin();

  Serial.begin(9600);

  //Initialize Pins for Up/Down Actuator Control
  pinMode(MotorRelay,OUTPUT);
  pinMode(OnOffRelay, OUTPUT);
  pinMode(UserMicroRelay1, OUTPUT);
  pinMode(UserMicroRelay2, OUTPUT);

 //Initialize pins for Buttons
  //pinMode(output1, OUTPUT);
  pinMode(TiltForwardPin, INPUT);
  pinMode(TiltBackPin, INPUT);
  pinMode(UpPin, INPUT);
  pinMode(DownPin, INPUT);

  //LED strip variables
  // Start up the LED strip
  strip.begin();
  // Update the strip, to start they are all 'off'
  strip.show();

  //Actuator Control Variables
  pinMode(output1, OUTPUT);
  pinMode(input1,INPUT);

}

void loop(){

//
  //delay(200);
  TiltForwardval = analogRead(TiltForwardPin);
  //Serial.print("Tilt forward is ");
  //Serial.print(TiltForwardval);
  //Serial.print('\n');
  TiltBackval = analogRead(TiltBackPin);
  //Serial.print("Tilt back is ");
  //Serial.print(TiltBackval);
  //Serial.print('\n');
  //delay(1000);


  //delay(1000);


   //Wire.beginTransmission(4); // transmit to device #4
```

```
      //Wire.write(12);              // sends one byte
      //Wire.endTransmission();    // stop transmitting
//
 //delay(5000);

//digitalWrite(UserMicroRelay1, LOW);
//delay(2000);
//digitalWrite(UserMicroRelay1, HIGH);
//delay(2000);
//digitalWrite(UserMicroRelay1, LOW);
//delay(2000);



  if(Serial.available()){
    //The Arduino can receive multiple types of signals from Processing
    //Gestures will send signals from Processing to activate lights and audioboard:
    //Come = C
    //Emergency = E
    //I'm Thinking = T
    //Go = G
    //Do Not Understand = N
    //Something in the Way = W
    //Up (Do Not Understand) = N
    //Up = U

    //Stop = S
    //Can't Do = X

    serial_signal=(char)Serial.read();
    Serial.println(serial_signal);

    //if(serial_signal=='U' || serial_signal=='D')  processing=1;
    //else  processing=0;
  }



  //If the Tilt Forward or Tilt Back pushbuttons are pressed, play their corresponding light sequences
  //and send a signal to processing to play the corresponding sounds
  while(TiltForwardval==1023){
    //forwardincident=1;
   Serial.println("Tilting Forward");
   //Serial.println();
   //Serial.print(TiltForwardval);
   //Serial.println();
    //if(switch_mode_f==0){
      //Use I2C to send track to play to Audioboard
      Wire.beginTransmission(4); // transmit to device #4
      Wire.write(12);              // sends one byte
      Wire.endTransmission();    // stop transmitting
      //delay(500);

      //This delay does not allow any other signals to trigger the system for 4 seconds
      //delay(trackdelay);
    //}
//    else if(switch_mode_f==1){
//      //Serial.println('flag');
//    On(80);
//    Clear(80);
//    //switch_mode_f++;
//    }
    //else if(switch_mode_f==2){
      //Use I2C to send track to play to Audioboard
      //Wire.beginTransmission(4); // transmit to device #4
      //Wire.write(12);              // sends one byte
      //Wire.endTransmission();    // stop transmitting
      //delay(500);

      //This delay does not allow any other signals to trigger the system for 4 seconds
      //delay(trackdelay);
      On(80);
      Clear(80);
      //switch_mode_f=0;
    //}
    //delay(200);
    TiltForwardval=analogRead(TiltForwardPin);

  }
//  if(forwardincident==1){
//    switch_mode_f++;
//    forwardincident=0;
//    if(switch_mode_f==3)  switch_mode_f=0;
//  }



while(TiltBackval==1023){
//  backincident=1;
    Serial.println("Tilting Back");
    //Serial.println();
    Serial.print(TiltBackval);
    //Serial.println();
//    if(switch_mode_b==0){
//      //Use I2C to send track to play to Audioboard
//      Wire.beginTransmission(4); // transmit to device #4
//      Wire.write(14);              // sends one byte
//      Wire.endTransmission();    // stop transmitting
//      //delay(500);
//      //This delay does not allow any other signals to trigger the system for 4 seconds
//      //delay(trackdelay);
//      //switch_mode_b++;
//    }
//    else if(switch_mode_b==1){
//    On(80);
//    Clear(80);
//    //switch_mode_b++;
//    }
    //else{
      //Use I2C to send track to play to Audioboard
      Wire.beginTransmission(4); // transmit to device #4
      Wire.write(14);              // sends one byte
      Wire.endTransmission();    // stop transmitting
      //delay(500);

      //This delay does not allow any other signals to trigger the system for 4 seconds
      //delay(trackdelay);
      On(80);
      Clear(80);
      //switch_mode_b=0;
    //}
    //delay(500);
    TiltBackval=analogRead(TiltBackPin);
  }
//    if(backincident==1){
//    switch_mode_b++;
//    backincident=0;
//    if(switch_mode_b==3)  switch_mode_b=0;
//  }



//  if(Downval==1023){
//      //Use I2C to send track to play to Audioboard
//      Wire.beginTransmission(4); // transmit to device #4
//      Wire.write(3);              // sends one byte
//      Wire.endTransmission();    // stop transmitting
//      //delay(500);

      //This delay does not allow any other signals to trigger the system for 4 seconds
      //delay(trackdelay);
    //delay(500);
  //}


//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////



////////////////////////////////////BEGIN GESTURE LIGHT AND SOUND CONTROL////////////////////////////////////

  //Come
  if(serial_signal=='C'){
    dither(80);
    Clear(500);
  }

  //Emergency
  else if(serial_signal=='E'){
    OnOff(80);
    Clear(80);
  }

  //I'm Thinking Lights
  else if(serial_signal=='T'){
    Thinking(80);
    Clear(1000);
  }

  //Go
  else if(serial_signal=='G'){
    if(switch_mode_g==0){
      Half(80);
      Clear(500);
      switch_mode_g++;
    }
    else if(switch_mode_g==1){
      //Use I2C to send track to play to Audioboard
      Wire.beginTransmission(4); // transmit to device #4
      Wire.write(8);              // sends one byte
      Wire.endTransmission();    // stop transmitting
      //delay(500);
      //This delay does not allow any other signals to trigger the system for 4 seconds
      //delay(trackdelay);
      switch_mode_g++;
    }
    else{
      //Use I2C to send track to play to Audioboard
      Wire.beginTransmission(4); // transmit to device #4
      Wire.write(8);             // sends one byte
      Wire.endTransmission();    // stop transmitting
      //delay(500);

      Half(80);
      Clear(500);
      switch_mode_g=0;
    }
  }

  //Do Not Understand
    else if(serial_signal=='N'){
    if(switch_mode_u==0){
      OnOff(175);
      Clear(1000);
      switch_mode_u++;
    }
    else if(switch_mode_u==1){
      //Use I2C to send track to play to Audioboard
      Wire.beginTransmission(4); // transmit to device #4
      Wire.write(24);            // sends one byte
      Wire.endTransmission();    // stop transmitting
      //delay(500);

      //This delay does not allow any other signals to trigger the system for 4 seconds
      //delay(trackdelay);
      switch_mode_u++;
    }
    else{
      //Use I2C to send track to play to Audioboard
      Wire.beginTransmission(4); // transmit to device #4
      Wire.write(24);            // sends one byte
      Wire.endTransmission();    // stop transmitting
      //delay(500);

      OnOff(175);
      Clear(1000);
      switch_mode_u=0;
    }
  }


  //Something in the Way
  else if(serial_signal=='W'){
    if(switch_mode_w==0){
      dither(80);
      Go(80);
      Clear(1000);
      switch_mode_w++;
    }
    else if(switch_mode_w==1){
      //Use I2C to send track to play to Audioboard
      Wire.beginTransmission(4); // transmit to device #4
      Wire.write(33);            // sends one byte
      Wire.endTransmission();    // stop transmitting
      //delay(500);

      //This delay does not allow any other signals to trigger the system for 4 seconds
      //delay(trackdelay);
      switch_mode_w++;
    }
    else{
      //Use I2C to send track to play to Audioboard
      Wire.beginTransmission(4); // transmit to device #4
      Wire.write(33);              // sends one byte
      Wire.endTransmission();    // stop transmitting
      //delay(500);

      dither(80);
      Go(80);
      Clear(1000);
      switch_mode_w=0;
    }
  }

  //Stop
  else if(serial_signal=='S'){
      //Use I2C to send track to play to Audioboard
      Wire.beginTransmission(4); // transmit to device #4
      Wire.write(9);              // sends one byte
      Wire.endTransmission();    // stop transmitting
      //delay(500);

      //This delay does not allow any other signals to trigger the system for 4 seconds
      //delay(trackdelay);
  }

  //Can't Do
  else if(serial_signal=='X'){
      //Use I2C to send track to play to Audioboard
      Wire.beginTransmission(4); // transmit to device #4
      Wire.write(29);              // sends one byte
      Wire.endTransmission();    // stop transmitting
      //delay(500);

      //This  does not allow any other signals to trigger the system for 4 seconds
      //delay(trackdelay);
  }

  else if(serial_signal=='L'){
    strip.setPixelColor(0, 255, 255, 255);
    strip.setPixelColor(15, 255, 255, 255);
    strip.setPixelColor(28, 255, 255, 255);
    strip.setPixelColor(41, 255, 255, 255);
    strip.show();
  }

  else if(serial_signal=='R'){
    strip.setPixelColor(20, 255, 255, 255);
    strip.setPixelColor(23, 255, 255, 255);
    strip.setPixelColor(33, 255, 255, 255);
    strip.setPixelColor(36, 255, 255, 255);
    strip.show();
  }
```

```
 else if(serial_signal=='O'){
   strip.setPixelColor(0, 0, 0, 0);
   strip.setPixelColor(15, 0, 0, 0);
   strip.setPixelColor(20, 0, 0, 0);
   strip.setPixelColor(23, 0, 0, 0);
   strip.setPixelColor(28, 0, 0, 0);
   strip.setPixelColor(33, 0, 0, 0);
   strip.setPixelColor(36, 0, 0, 0);
   strip.setPixelColor(41, 0, 0, 0);
   strip.show();
 }

/////////////////////////////////////END GESTURE LIGHT AND SOUND CONTROL//////////////////////////////////////


/////////////////////////////////////BEGIN ACTUATOR CONTROL//////////////////////////////////////

 //////////////////BUTTONS////////////////
 //digitalWrite(output1, HIGH); //Turn the relays on

 //valDown=analogRead(A0);
 //valUp=analogRead(A5);
 //Downval=analogRead(DownPin);
 //Upval=analogRead(UpPin);

 //float voltageDown=valDown*(0.5/1024.0);
 //float voltageUp=valUp*(0.5/1024.0);

 //Serial.println(voltageDown);
 //Serial.println(voltageUp);

 //while(voltageDown>0.4){


    Upval = analogRead(UpPin);
 //Serial.print("Up is ");
 //Serial.print(Upval);
 //Serial.print('\n');
 Downval = analogRead(DownPin);
 //Serial.print("Down is ");
 //Serial.print(Downval);
 //Serial.print('\n');


   //while(Downval>500){
     while(Downval>300 && processing==0){
     //Serial.print("Going Down");
   //Serial.println();
   //Serial.print(Downval);
   //Serial.println();
     //Serial.println();
     enteredDown=1;
   //pressDown++;
   if(pressDown==0){
     //if(countDown==0){

       //First time down button is pressed, say Something in the Way and deactivate actuator
       digitalWrite(OnOffRelay, LOW);
       digitalWrite(MotorRelay, LOW);
       digitalWrite(UserMicroRelay1,HIGH);
       digitalWrite(UserMicroRelay2,HIGH);
```

```
      //Serial.print("Shouldn't Move!");

      //Use I2C to send track to play to Audioboard
      Wire.beginTransmission(4); // transmit to device #4
      Wire.write(33);              // sends one byte
      Wire.endTransmission();    // stop transmitting
      //delay(500);

      dither(80);
      Go(80);
      Clear(1000);
      //countDown++;
    //}
  }
    else{
      //Serial.print("Should Move");
      //Serial.println();
      digitalWrite(MotorRelay, LOW);
      digitalWrite(UserMicroRelay1,LOW);
      digitalWrite(UserMicroRelay2,LOW);
      digitalWrite(OnOffRelay, LOW);
      //Serial.write(2);
      //countDown=0;

  }


  //valDown=analogRead(A0);
  Downval=analogRead(DownPin);
  //Serial.print(Downval);
  //Serial.println();
  //voltageDown=valDown*(0.5/1024.0);
  //Serial.println(press);
  //Serial.println(voltageDown);
  //Serial.println(voltageUp);

}


if(enteredDown==1)  pressDown=1;


//while(voltageUp>0.4){
  //pressUp++;
  //for power supply
 // while(Upval>500){
   while(Upval>300 && processing==0){
   //Serial.print("Going Up");
       //Serial.println();
 //Serial.print(Upval);
 //Serial.println();
    //Serial.println();
  //pressUp++;
  enteredUp=1;
  if(pressUp==0){
    //if(countUp==0){
      //First time Up buttom is pressed, say Emergency and deactivate actuator
      digitalWrite(OnOffRelay, LOW);
      digitalWrite(UserMicroRelay1,HIGH);
```

```
        digitalWrite(UserMicroRelay2,HIGH);
        OnOff(80);
        Clear(80);
        //countUp++;
      //}
    }
      else{
        digitalWrite(UserMicroRelay1,LOW);
        digitalWrite(UserMicroRelay2,LOW);
        digitalWrite(OnOffRelay, LOW);
        //Serial.write(4);
        //countUp=0;
      }

    //valUp=analogRead(A5);
    Upval=analogRead(UpPin);
    //voltageUp=valUp*(0.5/1024.0);
    //Serial.println(voltageDown);
    //Serial.println(voltageUp);
  }
  //countUp++;
  if(enteredUp==1)  pressUp=1;

   //while(voltageDown<0.4 && voltageUp<0.4){
    if(Downval<300 && Upval<300){
      //Serial.print("Not Moving");
      //digitalWrite(OnOffRelay, LOW);
    waiting++;
    //valUp=analogRead(A5);
    Upval=analogRead(UpPin);
    //voltageUp=valUp*(0.5/1024.0);
    //Downval=analogRead(0);
    Downval=analogRead(DownPin);
    //voltageDown=valDown*(0.5/1024.0);
    //Serial.println(voltageDown);
    //Serial.println(voltageUp);
  }



  if(waiting>200){
    //pressDown=0;
    //pressUp=0;
    waiting=0;
  }


////////////////GESTURES///////////////
if(serial_signal=='U'){
    microtakeover=1;
    MicroOnOff=1;
    UpDown=0;

    processing=1;
  }

  else if(serial_signal=='D'){
    microtakeover=1;
    MicroOnOff=1;
    UpDown=1;
    processing=1;
  }

  else{
    microtakeover=0;
    MicroOnOff=0;
    UpDown=0;
  }

  //microtakeover=1; //The programmer controls the output to the linear actuator by switching the
UserMicroRelays (R2 and R3).
  //OnOff=1; //The OnOff Relay (R4) either supplies or does not supply the opposite 30 V signal.
  //UpDown=0; //The Motor Relay (R1) causes the actuator to move up or down when the microtakeover is active.
  //0 goes up and 1 goes down

  if (microtakeover==1){
    digitalWrite(UserMicroRelay1,HIGH);
    digitalWrite(UserMicroRelay2,HIGH);
  }
  else{
    digitalWrite(UserMicroRelay1,LOW);
    digitalWrite(UserMicroRelay2,LOW);
  }
  if (MicroOnOff == 1){
    digitalWrite(OnOffRelay,HIGH);
  }
  else{
    digitalWrite(OnOffRelay,LOW);
  }
  if (UpDown == 1){
    digitalWrite(MotorRelay, HIGH);
  }
  else{
    digitalWrite(MotorRelay, LOW);
  }

//////////////////////////////////END ACTUATOR CONTROL/////////////////////////////

}
```

BendIn

```
void BendIn(uint8_t wait) {
  int i, j, pos, dir;
  byte r, g, b;
  r=127;
  g=127;
  b=127;

  pos = (strip.numPixels()-1);
  dir = 1;

  for(i=(strip.numPixels()-1); i<((strip.numPixels()-1)*2); i++) {
    // Draw 5 pixels centered on pos.  setPixelColor() will clip
    // any pixels off the ends of the strip, no worries there.
    // we'll make the colors dimmer at the edges for a nice pulse
    // look
    strip.setPixelColor(pos - 2, strip.Color(r/4, g/4, b/4));
    strip.setPixelColor(pos - 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos, strip.Color(r, g, b));
    strip.setPixelColor(pos + 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos + 2, strip.Color(r/4, g/4, b/4));

    strip.show();
    delay(wait);
    // If we wanted to be sneaky we could erase just the tail end
    // pixel, but it's much easier just to erase the whole thing
    // and draw a new one next time.
    for(j=-2; j<= 2; j++)
        strip.setPixelColor(pos+j, strip.Color(0,0,0));
    // Bounce off ends of strip
    pos += dir;
    if(pos < 0) {
      pos = 1;
      dir = -dir;
    } else if(pos >= strip.numPixels()) {
      pos = strip.numPixels() - 2;
      dir = -dir;
    }
  }
}
```

Clear

```
void Clear(uint8_t wait) {
  int i;

  for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, 0); // Erase pixel, but don't refresh!

  }

  strip.show(); // Refresh to turn off last pixel
  delay(wait);
}
```

Dither

```
 void dither(uint8_t wait) {

  // Determine highest bit needed to represent pixel index
  int hiBit = 0;
  int n = strip.numPixels() - 1;
  byte r, g, b;
  r=127;
  g=127;
  b=127;

  for(int bit=1; bit < 0x8000; bit <<= 1) {
    if(n & bit) hiBit = bit;
  }

  int bit, reverse;
  for(int i=0; i<(hiBit << 1); i++) {
    // Reverse the bits in i to create ordered dither:
    reverse = 0;
    for(bit=1; bit <= hiBit; bit <<= 1) {
      reverse <<= 1;
      if(i & bit) reverse |= 1;
    }
    strip.setPixelColor(reverse, r, g, b);
    strip.show();
    delay(wait);
  }
  delay(250); // Hold image for 1/4 sec
}
```

Go

```
void Go(uint8_t wait) {

  // Determine highest bit needed to represent pixel index
  int hiBit = 0;
  int n = strip.numPixels() - 1;
  byte r, g, b;
  r=0;
  g=0;
  b=0;

  for(int bit=1; bit < 0x8000; bit <<= 1) {
    if(n & bit) hiBit = bit;
  }

  int bit, reverse;
  for(int i=0; i<(hiBit << 1); i++) {
    // Reverse the bits in i to create ordered dither:
    reverse = 0;
    for(bit=1; bit <= hiBit; bit <<= 1) {
      reverse <<= 1;
      if(i & bit) reverse |= 1;
    }
    strip.setPixelColor(reverse, r, g, b);
    strip.show();
    delay(wait);
  }
  delay(250); // Hold image for 1/4 sec
}
```

Half

```
void Half(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x<16){
         if (i>16){
        r=127;
        g=127;
        b=127;
         }

         else{
          r=0;
          g=0;
          b=0;
         }
       }

       else {
         if (i<=16){
        r=127;
        g=127;
        b=127;
         }
        else{
         r=0;
         g=0;
         b=0;
        }
       }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }
```

On

```
void On(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<(strip.numPixels()); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {
       if(x>=0){
        r=127;
        g=127;
        b=127;}

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
      }
  }
```

OnOff

```
void OnOff(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<strip.numPixels(); x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {

      if(x%2==0){
        r=32;
        g=32;
        b=32;}

      else if(x%2==1){
          r=0;
          g=0;
          b=0;
      }

      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
  }
}
```

ReprimandWompWomp

```
void ReprimandWompWomp(uint8_t wait) {
  byte  r, g, b;

  for(int x=0; x<68; x++)
  {
    for(int i=0; i<strip.numPixels(); i++) {

      if(x<15){
        r=127;
        g=127;
        b=127;}

      else if(x>15 && x<23){
          r=0;
          g=0;
          b=0;
      }

      else if(x>23 && x<61){
         r=127;
         g=127;
         b=127;}

      else if(x>61){
          r=0;
          g=0;
          b=0;
      }



      strip.setPixelColor(i, r, g, b);
    }
    strip.show();
    delay(wait);
  }
}
```

Thinking

```
void Thinking(uint8_t wait) {
  int i, j, pos, dir;
  byte r, g, b;
  r=127;
  g=127;
  b=127;

  pos = 0;
  dir = 1;

  for(i=0; i<((strip.numPixels()-1) * 3); i++) {
    // Draw 5 pixels centered on pos.  setPixelColor() will clip
    // any pixels off the ends of the strip, no worries there.
    // we'll make the colors dimmer at the edges for a nice pulse
    // look
    strip.setPixelColor(pos - 2, strip.Color(r/4, g/4, b/4));
    strip.setPixelColor(pos - 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos, strip.Color(r, g, b));
    strip.setPixelColor(pos + 1, strip.Color(r/2, g/2, b/2));
    strip.setPixelColor(pos + 2, strip.Color(r/4, g/4, b/4));
    strip.show();
    delay(wait);
    // If we wanted to be sneaky we could erase just the tail end
    // pixel, but it's much easier just to erase the whole thing
    // and draw a new one next time.
    for(j=-2; j<= 2; j++)
        strip.setPixelColor(pos+j, strip.Color(0,0,0));
    // Bounce off ends of strip
    pos += dir;
    if(pos < 0) {
      pos = 1;
      dir = -dir;
    } else if(pos >= strip.numPixels()) {
      pos = strip.numPixels() - 2;
      dir = -dir;
    }
  }
}
```

TurnOff

```
void TurnOff(uint8_t wait) {
  for (int i=0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, 0, 0, 0);
    }
      strip.show();
}
```

REFERENCES

Angelopoulou, A., Psarrou, A., Garcia-Rodriguez, J., & Gupta, G. (2010). *Tracking gestures using a probabilistic self-organising network.* Paper presented at the International Joint Conference on Neural Networks.

Association, N. S. (2012a). National Stroke Association Retrieved July 21, 2012, from www.stroke.org

Association, N. S. (2012b). Paralysis - Hemiparesis - National Stroke Association Retrieved June 9, 2013, from http://www.stroke.org/site/PageServer?pagename=hemiparesis

Association, N. S. (2012c). Stroke 101 Fact Sheet Retrieved July 21, 2012, from http://www.stroke.org/site/DocServer/STROKE_101_Fact_Sheet.pdf?docID=4541

Bartlett, M., Littlewort, G., Frank, M. G., Lainscsek, C., Fasel, I., & Movellan, J. (2005). *Recognizing facial expression: Machine learning and application to spontaneous behavior.* Paper presented at the IEEE International Conference on Computer Vision and Pattern Recognition, Osaka, Japan.

Bartlett, M., Littlewort, G., Frank, M. G., Lainscsek, C., Fasel, I., & Movellan, J. (2006). *Fully automatic facial action recognition in spontaneous behavior.* Paper presented at the 7th International Conference on Automatic Face and Gesture Recognition.

Beach, S., Schulz, R., Downs, J., Matthews, J., Barron, B., & Seelman, K. (2009). Disability, age, and informational privacy attitudes in quality of life technology applications: Results from a national Web survey. *ACM Transactions on Accessible Computing, 2*(1), 1-21.

BenAbdelkader, C., Cutler, R. G., & Davis, L. S. (2004). Gait recognition using image self-similarity. *EURASIP Journal on Applied Signal Processing*, 572-585.

Blumberg, B., Downie, M., Ivanov, Y., Berlin, M. P., Johnson, P., & Tomlinson, B. (2002). Integrated learning for interactive synthetic characters. *ACM Transactions on Graphics, 21*, 417-426.

Bobick, A. F. (1999). Movement, Activity and Action: The Role of Knowledge in the Perception of Motion. *Philosophical Transactions of the Royal Society B: Biological Sciences, 352*(1358), 1257-1266.

Breazeal, C. (2004). Social Interations in HRI: The Robot View. *IEEE Transactions on Systems, Man, and Cybernetics, 34*(2).

Breazeal, C., Siegel, M., Berlin, M., Gray, J., Grupen, R., Deegan, P., . . . McBean, J. (2008). *Mobile, dexterous, social robots for mobile manipulation and human-robot interaction.* Paper presented at the SIGGRAPH '08, New York.

Brooks, J. O., Smolentzov, L., DeArment, A., Logan, W., Green, K. E., Walker, I., . . . Yanik, P. (2011a). Toward a "Smart" Nightstand Prototype: An Examination of Nightstand Table Contents and Preferences. *Health Environments Research and Design Journal, 4*(2), 91-108.

Brooks, J. O., Smolentzov, L., DeArment, A., Logan, W., Green, K. E., Walker, I., . . . Yanik, P. (2011b). Toward a "Smart" Nightstand Prototype: An Examination of Nightstand Table Contents and Preferences. *HERD, 4*(2), 91-108.

Brooks, J. O., Smolentzov, L., Mossey, M. E., Carroll, C., Kendrick, K., Sprogis, K., . . . Green, K. (2012). Group Differences in Preferences for a Novel Nightstand. *Health Environments Research and Design Journal, 5*(4), 86-95.

Cassell, J. (1998). *A Framework for Gesture Generation and Interpretation*. Cambridge and New York: Cambridge University Press.

Cassell, J., Bickmore, W. T., Billinghurst, M., Campbell, L., Chang, K., Vilhjalmsson, H. H., & Yan, H. (1999). *Embodiment in conversational interfaces: Rea*. Paper presented at the SIGCHI.

Chen, Q., Georganas, N. D., & Petriu, E. M. (2008). Hand gesture recognition using Haar-like features and a stochastic context-free grammar. *IEEE Transactions on Instrumentation and Measurement, 57*(8), 1562-1571.

Cheng, H. T., Chen, A. M., Razdan, A., & Buller, E. (2011). *Contactless gesture recognition system using proximity sensors.* Paper presented at the IEEE International Conference on Consumer Electronics (ICCE).

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied Multiple Regression/ Correlation Analysis for the Behavioral Sciences* (Third ed.). Mahwah, New Jersey: Lawrence Erlbaum Associates, Inc.

Cutler, R., & Davis, L. S. (2002). Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(8), 781-796.

Dalal, N., Triggs, B., Rhone-Alps, I., & Montbonnot, F. (2005). *Histograms of Oriented Gradients for Human Detection.* Paper presented at the IEEE Computer Society Conference on Computer Vision and Pattern Recognitio.

Dario, P., Guglielmelli, E., Allotta, B., & Carrozza, M. C. (1996). Robotics for Medical Applications. *IEEE Robotics and Automation Magazine, 3*(3), 44-56.

Dautenhahn, K. (2007). Socially intelligent robots: Dimensions of human-robot interaction. *Philosophical Transactions of the Royal Society of London - Series B: Biological Sciences, 362*(1480), 679-704.

Dellon, B., & Matsuoka, Y. (2007). Prosthetics, Exoskeletons, and Rehabilitation. *IEEE Robotics and Automation Magazine, 14*(1), 30-34.

Demeris, G., Hensel, B. K., Skubic, M., & Rantz, M. (2008). Senior residents' perceived need of and preferences for "smart home" sensor technologies. *International Journal of Technology Assessment in Health Care*, 120-124.

Feil-Safer, D., & Mataric, M. J. (2005). *Defining Socially Assistive Robotics*. Paper presented at the Internaitonal Conference on Rehabilitation Robots, Chicago, IL.

Fong, T., Nourbakhsh, I., & Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and Autonomous Systems, 42*, 143-166.

Forlizzi, J. (2005). Robotic products to assist the aging population. *Interactions, 12*(2), 16-18.

Forlizzi, J., DiSalvo, C., & Gemperle, F. (2004). Assistive robotics and an ecology of elders living independently in their homes. *Human-Computer Interaction, 19*, 25-59.

Fritzke, B. (1995). A Growing Neural Gas Network Learns Topologies. *Advances in Neural Information Processing Systems, 7*(7), 625-632.

Gervain, J., & Mehler, J. (2007). *Perceptual primitives in language acquisition: Near infrared spectroscopy studies with neonates.* Paper presented at the Conference on Biolinguistics: Language Evolution and Variation, Venice.

Gonsior, B., Sosnowski, S., Buß, M., Wollherr, D., & Kuhnlenz, K. (2012, October 7-12). *An emotional adaption approach to increased helpfulness towards a robot.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, Viamoura, Algarve, Portugal.

Green, K. E. (2008). *Back to the future: Three educational experiments in interactive architecture as anticipated in 1960s visionary architecture.* Paper presented at the Association of Collegiate Schools of Architecture (ACSA) 2008 National Conference, Houston.

Green, K. E., Gugerty, L., Walker, I. D., & Witte., J. (2006). *Three robot-rooms/the AWE project.* Paper presented at the CHI'06: Extended Abstracts on Human Factors in Computing Systems, Montreal.

Green, K. E., Gugerty, L. J., Walker, I. D., & Witte, J. C. (2005a). *Architecture plus: The collaborative Animated Work Environment design research project.* Paper presented at the Association of Collegiate Schools of Architecture National Conference, Salt Lake City, UT.

Green, K. E., Gugerty, L. J., Walker, I. D., & Witte, J. C. (2005, September). *AWE (Animated Work Environment): Ambient intelligence in working life.* Paper presented at the 2005 Conference on Intelligent Ambience and Well-Being (Ambience), Tampere, Finland.

Green, K. E., Gugerty, L. J., Walker, I. D., & Witte, J. C. (2005b). *CREATING AWE: Formulating a research process for an interdisciplinary effort to build an Animated Work Environment (AWE).* Paper presented at the International Symposium on Intelligent Environments, Microsoft Research and Cambridge University, Cambridge, England, UK.

Green, K. E., Walker, I. D., Brooks, J. O., Threatt, T., & Merino, J. (2011, September). *An Assistive Robotic Table (ART) Promoting Independent Living.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, workshop on New and Emerging Technologies in Assistive Robotics, San Francisco, CA.

Gross, M., & Green, K. E. (2012). Architectural Robotics, Inevitably. *Interactions, xix*(1), 28-33.

Health, N. I. o. (2011). Post-Stroke Rehabilitation Fact Sheet: National Institute of Neurological Disorders and Stroke (NINDS) Retrieved June 9, 2013, from http://www.ninds.nih.gov/disorders/stroke/poststrokerehab.htm#whatis

Holmstrom, J. (2002). *Growing Neural Gas: Experiments with GNG, GNG with Utility and Supervised GNG.* (masters thesis), Uppsala University.

Houser, A., Fox-Grage, W., & Gibson, M. J. (2006). Across the states: Profiles of long-term care and independent livingAARP Public Policy Institute (7th ed.).

Huang, Y., Huang, K., Tan, T., & Tao, D. (2009, September). *A novel visual organization based on topological perception.* Paper presented at the 9th Asian conference on Computer Vision.

IDEO. (2003). *IDEO Method Cards: 51 Ways to Inspire Design*. Palo Alto: IDEO and William Stout.

Jin, S., Li, Y., Lu, G., Luo, J., Chen, W., & Zheng, X. (2011, March). *SOM-based hand gesture recognition for virtual interactions.* Paper presented at the IEEE International Symposium on VR Innovation (ISVRI).

Johansson, G. (1973). Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics, 14*(2), 201-211.

Junejo, I., Dexter, E., Laptev, I., & Perez, P. (2008). *Cross-View Action Recognition from Temporal Self-Similarities.* Paper presented at the European Conference on Computer Vision.

Kaplan, F., Oudeyer, P. Y., Kubinyi, E., & Miklosi, A. (2002). Robotic clicker training. *Robotics and Autonomous Systems, 38*(3), 197–206.

Karahoca, A., & Nurullahoglu, M. (2008). *Human motion analysis and action recognition.* Paper presented at the 1st WSEAS International Conference on Multivariate Analysis and its Application in Science and Engineering.

Kirchner, N., & Alempijevic, A. (2012). A Robot Centric Perspective on the HRI. *Journal of Human-Robot Interaction, 1*(2), 135-157. doi: 10.5898/JHRI1.2Kirchner

Kleinsmith, A. (2004). *Exploring Nonverbal Communication in Human-Machine Interaction: A Categorical Approach to Affective Gesture Recognition.* (masters thesis), University of Aizu.

Klingspor, V., Demiris, J., & Kaiser, M. (1997). Human-robot-communication and machine learning. *Applied Artificial Intelligence Journal, 11*, 719-746.

Komatsu, T. (2006). *Audio subtle expressions affecting user's perceptions.* Paper presented at the International Conference on Intelligent User Interface, San Diego.

Kuno, Y., Murashima, T., Shimada, N., & Shirai, Y. (2000). *Interactive gesture interface for intelligent wheelchairs.* Paper presented at the IEEE International Conference on Multimedia and Expo.

Lallée, S., Lemaignan, S., Lenz, A., Melhuish, C., Natale, L., Skachek, S., . . . Dominey, P. (2010). *Towards a platform-independent cooperative human-robot interaction system: I. perception.* Paper presented at the IROS, Taipei.

Lamentec, J. C., & Bajcsy, P. (2004). *Recognition of arm gestures using multiple orientation sensors: gesture classification.* Paper presented at the 7th IEEE International Conference on Intelligent Transporation Systems.

Lazar, J., Feng, J. H., & Hochheiser, H. (2010). *Research Methods in Human-Computer Interaction*. West Sussex, UK: John Wiley and Sons, Ltd.

Leifer, L. (1981). Rehabilitative Robots. *Robotic Age*.

Malizia, A., & Bellucci, A. (2012). The Artificiality of natural user interfaces. *Communications of the ACM, 55*(3), 36-38.

Manganelli, J., Threatt, A., Brooks, J. O., Healy, S., Merino, J., Yanik, P., . . . Green, K. E. (2013). Examination of how and why over-the-bed tables are used: Use cases and needs from healthcare providers. *Health Environments Research and Design Journal*, Manuscript submitted for publication.

Manganelli, J., Threatt, A., Brooks, J. O., Healy, S., Merino, J., Yanik, P., . . . Green, K. E. (2013a). Examination of how and why over-the-bed tables are used: Use cases and needs from healthcare providers. *Health Environments Research and Design Journal*, Manuscript submitted for publication.

Manganelli, J., Threatt, A., Brooks, J. O., Merino, J., Yanik, P., Healy, S., . . . Green, K. (2013b). Validating over-the-bed tables use cases & need statements: Health care providers assessment. *Health Environments Research and Design Journal*, Manuscript submitted for publicaton.

Manganelli, J., Threatt, A., Brooks, J. O., Smolentzov, L., Mossey, M., Healy, S., . . . Green, K. (2012). Examination of overbed tables: Health care povider & user preferences. *Health Environments Research and Design Journal*, Manuscript submitted for publication.

Matsumoto, N., Fujii, H., Goan, M., & Okada, M. (2005, August). *Minimal design strategy for embodied communication agents.* Paper presented at the 14th IEEE International Workshop on Robot-Human Interaction, Nashville, TN.

McNamara, A. (2009, September). *Exploring perceptual equivalence between real and simulated imagery.* Paper presented at the 9th Asian conference on Computer Vision.

Merino, J., Threatt, A. L., Walker, I. D., & Green, K. E. (2012). *Forward Kinematic Model for Continuum Robotic Surfaces*. Paper presented at the International Conference on Intelligent Robotics Systems, Villamoura, Portugal.

Mezger, M., Winfried, I., & Giese, M. A. (2005). *Trajectory synthesis by hierarchical spatio-temporal correspondence: comparison of different methods.* Paper presented at the 2nd symposium on Applied perception in graphics and visualizatio.

Microsoft. (2012). Microsoft Xbox 360 + Kinect Website Retrieved December 17, 2012, from http://www.xbox.com/en-US/kinect

Mitra, S., & Acharva, T. (2007). Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 37*(3), 311-324.

Molich, R., & Nielsen, J. (1990). Improving a Human-Computer Dialogue. *Communications of the ACM, 33*(3), 338-348.

Moni, M. A., & Ali, A. B. M. S. (2009). *HMM based hand gesture recognition: A review on techniques and approaches.* Paper presented at the IEEE International Conference on Computer Science and Information Technology.

Mori, M. (1970). The Uncanny Valley. *Energy, 7*(4), 33-35.

Mutlu, B., Bartneck, C., Ham, J., Evers, V., & Kanda, T. (2011, November 24-25). *Social robotics.* Paper presented at the Third International Conference on Social Robotics, Amsterdam.

Nielsen, J. (1992). *Finding Usability Problems Through Heuristic Evaluation*. Paper presented at the CHI, Monterey, CA.

Nielsen, J. (1993). *Usability Engineering*. London: Academic Press.

Okada, M., Sakamoto, S., & Suzuki, N. (2000, July). *Muu: Artificial creatures as an embodied interface.* Paper presented at the 27th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2000), New Orleans, LA.

Pain in non-verbal elderly largely undetected by family caregivers. (2011). *The Journal of Pain of the American Pain Society*. Retrieved from http://www.ampainsoc.org/press/2011/pain_non-verbal.htm.

Picard, R. (1995). Affective Computing (Vol. 321): MIT Technical Report.

Pineau, J., Montemerlo, M., Pollack, M., Roy, N., & Thrun, S. (2002, October). *Probabilistic control of humanrobot interaction: Experiments with a robotic assistant for nursing homes.* Paper presented at the The second IARP/IEEE/RAS Joint Workshop on Technical Challenges for Robots in Human Environments (DRHE).

Prasad, J. S., & Nandi, G. C. (2009). *Clustering method evaluation for hidden markov model based real-time gesture recognition.* Paper presented at the International Conference on Advances in Recent Technologies in Communication and Computing.

Quenqua, D. (2012, December 4). Pushing Science's Limits In Sign Language Lexicon, *The New York Times*.

Rao, C., Yilmaz, A., & Shah, M. (2002). View-Invariant Representation and Recognition of Actions. *International Journal of Computer Vision, 50*(2), 203-226.

Read, R., & Belpaeme, T. (2010, October 29). *Interpreting non-linguistic utterances by robots: Studying the influence of physical appearance.* Paper presented at the 3rd international workshop on Affective interaction in natural environments, Firenze, Italy.

Reeves, B., & Nass, C. (1996). *The media equation: How people treat computers, television, and nwe media like real people and places.* New York, NY: Cambridge University Press.

Review of Hartfield, B. and Winograd, T. Bringing Design to Software. (1996). from http://hci.stanford.edu/bds/8p-ideo.html

Rogers, Y., Sharp, H., & Preece, J. (2011). *Interaction Design: Beyond Human-Computer Interaction* (3rd ed.). West Sussex, UK: John Wiley & Sons, Ltd.

Rosen, J., & Hannaford, B. (2006). Doc at a distance. *IEEE Spectrum, 43*(10), 34-39.

Rossini, N. (2012). *Reinterpreting Gesture as Language. Language "in Action."*. Amsterdam: IOS Press.

Roy, N., Baltus, G., Fox, D., Gemperle, F., Goetz, J., Hirsch, T., . . . Thrun, S. (2000). *Towards Personal Service Robots for the Elderly.* Paper presented at the Workshop on Interactive Robots and Entertainment (WIRE 2000).

Ryu, D., Um, D., Tanofsky, P., Koh, D. H., Ryu, Y. S., & Kang, S. (2010). *T-less: A novel touchless human-machine interface based on infrared proximity sensing.* Paper presented at the IEEE/RJS International Conference on Intelligent Robots and Systems.

Salter, K., Teasell, R., Bhogal, S., Zettler, L., & Foley, N. (2013). EBRSR - 14. Aphasia, from http://www.ebrsr.com/uploads/Aphasia-SREBR-SREBR-15_1.pdf

Scassellati, B., Admoni, H., & Mataric, M. (2012). Robots for Use in Autism Research. *The Annual Reviw of Biomedical Engineering, 14*, 275-294.

Schlomer, T., Poppinga, B., Henze, N., & Boll, S. (2008). *Gesture recognition with a wii controller.* Paper presented at the 2nd International Conference on Tangible and Embedded Interaction.

Smolentzov, L. (2010). *Desired Characteristics of 'Smart' Nightstands for Higher and Lower Functioning Older Adults.* (master's thesis), Clemson University, Retrieved form ProQuest LLC.

Special Issue on Robots in Surgery. (1995). *IEEE Engineering in Medicine and Biology Magazine, 14*(3).

Stergiopoulou, E., & Papamarkos, N. (2006, October). *A new technique for hand gesture recognition.* Paper presented at the International Conference on Image Processing.

Susskind, J., Hershey, J., & Movellan, J. (2004, October 20). *Exact inference in robots using topographical uncertaintymaps.* Paper presented at the 2nd International Conference on Development and Learning, The Salk Institute, San Diego.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction* (Vol. 1): Cambridge Univ Press.

Syrdal, D. S., Dautenhahn, K., Koay, K. L., & Walters, M. L. (2009). *The negative attitudes towards robots scale and reactions to robot behaviour in a live human-robot interaction study.* Paper presented at the New Frontiers in Human-Robot Interaction, AISB 2009, Edinburgh, UK.

Tapus, A., Mataric, M., & Scassellati, B. (2007). Socially assistive robotics. *IEEE Robotics and Automation Magazine, 14*(1), 35-42.

Teasell, R., McClure, A., Katherine, Salter, & Murie-Fernandez, M. (2013). Evidence-Based Review of Stroke Rehabilitation - D. Cognitive Recovery Post-Stroke Educational Supplement Retrieved August 1, 2012, from http://www.ebrsr.com/~ebrsr/uploads/D_Cognitive_Disorders_(PR).pdf

Thomaz, A. L., & Breazeal, C. (2008). Teachable Robots: Understanign human teaching behavior to build more effective robot learners. *Artificial Intelligence, 172*, 716-737.

Threatt, A. L., Merino, J., Green, K. E., Walker, I. D., Brooks, J. O., Ficht, S., . . . Yanik, P. (2012). *A Vision of the Patient Room as an Architectural-Robotic Ecosystem*. Paper presented at the IROS, Villamoura, Portugal.

Touzet, C. F. (1997). Neural reinforcement learning for behaviour synthesis. *Robotics and Autonomous Systems, 22*(3), 251-281.

Varkonyi-Koczy, A. R., & Tusor, B. (2011). Human–computer interaction for smart environment applications using fuzzy hand posture and gesture models. *IEEE Transactions on Instrumentation and Measurement, 60*(5), 1505-1514.

Wada, K., & Shibata, T. (2007). Living with seal robots – Its sociopsychological and physiological influences on the elderly at a care house. *IEEE Transactions on Robotics, 23*(5), 972-980.

Wade, E., Parnandi, A. R., & Mataric, M. J. (2011). *Using Socially Assistive Robotics to Augment Motor Task Performance in Individuals Post-Stroke*. Paper presented at the International Conference on Intelligent Robots and Systems, San Francisco, CA.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*(3), 279-292.

Weiner, N. D. C., Boston. 1954. (1954). *The human use of human beings: Cybernetics and society*. Boston: De Capo.

Weiser, M. (1991). The Computer for the 21st Century Retrieved December 5, 2010, from http://sandbox.xerox.com/want/papers/ubi-sciam-sep91.pdf

Wilson, A. D., & Bobick, A. F. (2000). *Realtime online adaptive gesture recognition.* Paper presented at the 15th International Conference on Pattern Recognition.

Yamada, S., & Komatsu, T. (2007). Designing Simple and Effective Expressions of Robot's Primitive Minds to a Human. In N. Sankar (Ed.), *Human-Robot Interaction*. Vienna: Itech.

Yamato, J., Ohya, J., & Ishii, K. (1992). *Recognizing human action in timesequential images using hidden markov model.* Paper presented at the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

Yanik, P. M., Manganelli, J., Merino, J., Threatt, A. L., Brooks, J. O., Green, K. E., & Walker, I. D. (2012). *Use of Kinect Depth Data and Growing Neural Gas for Gesture Based Robot Control.* Paper presented at the 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth2012), La Jolla, CA.

Yanik, P. M., Merino, J., Manganelli, J., Smolentzov, L., Walker, I. D., Brooks, J. O., & Green, K. E. (2011). Sensor placement for activity recognition: comparing video data with motion sensor data. *International Journal of Circuits, Systems and Signal Processing*(5), 279-286.

Yun, L., & Peng, Z. (2009). *An automatic hand gesture recognition system based on Viola-Jones method and SVMs.* Paper presented at the 2nd International Workshop on Computer Science and Engineering.

Zhou, S., Shan, Q., Fei, F., Li, W. J., Kwong, C. P., Wu, P. C. K., . . . Liou, J. Y. J. (2009). *Gesture recognition for interactive controllers using mems motion sensors.* Paper presented at the 4th IEEE International Conference on Nano/ Micro Engineered and Molecular Systems.