UNIVERSITY OF CALIFORNIA,
IRVINE

The Resonating Arm Exerciser: An Ultra Low Cost, Non-Robotic Rehabilitation Device with Patient-Active Assistance

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Mechanical and Aerospace Engineering

by

Daniel Kevin Zondervan

Thesis Committee:
Professor David Reinkensmeyer, Chair
Professor James Bobrow
Professor Steven Cramer

2011

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Page

ACKNOWLEDGEMENTS

I would like to thank Dr. David Reinkensmeyer for his support and guidance throughout this study and the construction of this thesis. Funding was provided by the UC Global Health Travel Fellowship. Special thanks to Dr. Jorge Hernandez and Lorena Palofax at the Instituto Nacional de Neurología y Neurocirugía in Mexico City, MEX for their hospitality and assistance. The wheelchair used in the study was donated by Free Wheelchair Mission. Thank you to the UC Irvine Senior Design team that made an initial prototype of RAE.

ABSTRACT OF THE THESIS

The Resonating Arm Exerciser:
An Ultra Low Cost, Non-Robotic Rehabilitation
Device with Patient-Active Assistance

By

Daniel Kevin Zondervan

Master of Science in Mechanical and Aerospace Engineering

University of California, Irvine, 2011

Professor David J. Reinkensmeyer, Chair

Background: Robotic arm therapy devices that incorporate assistance can enhance arm recovery, motivate patients to practice, and allow therapists to deliver semi-autonomous training. However, such devices are expensive and complex, making them impractical for widespread use, particularly in settings with limited financial or technical resources. *Objective:* We sought to develop a device that could provide robot-like mechanical assistance for arm training at much lower cost. *Methods:* The Resonating Arm Exerciser (RAE) consists of a lever that attaches to the push rim of a wheelchair, a forearm support, and an elastic band that provides resistance. Patients use the lever to rock the wheelchair at its resonant frequency, which provides mechanical assistance while exercising their upper extremity. We performed a pilot study to test whether exercising with the device led to improvements in arm mobility. Eight participants with chronic stroke (35 ± 24 months since injury) and substantial arm impairment (initial upper extremity Fugl-Meyer score = [17 ± 8] / 66) exercised with RAE for eight 45 minute sessions over three weeks. Primary outcome measures were FM score and average active range of motion of RAE during a one-minute test. *Results:* The average increase in FM score was 8.5 ± 4.1 SD pts, and the average improvement in active range of motion was 66% ± 20% SD. Results were sustained after three months. *Conclusions:* Exercising with a low-cost mechanical device that snaps onto a manual wheelchair and uses resonance to assist arm movement can improve arm function in chronic hemiparetic stroke.

1. INTRODUCTION

The introduction of robotic therapy has allowed for sophisticated implementations of assistance in rehabilitation therapy. These implementations have been documented to help people with stroke recover arm movement through training (Kwakkel, et al. 2007). Unfortunately, the use of robotic therapy is limited due to high costs, lengthy production times, reliance on accurate sensors and powerful computing systems, and the need for technical training in order to maintain a robotic system. Innovative new methods for rehabilitation must be developed in order increase the availability of low-cost assisted therapy, and this challenge was at the heart of the development of the Resonating Arm Exerciser (RAE).

This thesis presents the detailed design of RAE along with the results from a small clinical case study of the device. The study tests the hypothesis that (1) it is feasible to use RAE for arm rehabilitation and (2) exercise with RAE leads to improvements in arm mobility for people with severe hemiparesis following chronic stroke.

2. LITERATURE REVIEW

The motivation for this study and the foundation of the hypothesis were both drawn from extensive previous work in the field. Many studies have shown that the human motor system retains substantial capacity for plasticity following neurological injuries such as stroke and spinal cord injury, and thus intensive rehabilitation exercise reduces long term motor impairment of both the upper and lower extremities (van der Lee et al. 2001; Sawaki 2005; Ada et al. 2006; Kloosterman et al. 2008). However, rehabilitation exercise delivered one-on-one with a therapist is expensive. There has thus been a rapid surge in the development of robotic and computer-based devices for partially automating intensive rehabilitation exercise (Mehrholz et al. 2008). While practice with such devices reduces arm impairment, the devices are relatively expensive and complex, making them impractical for use by many of the people in the world who could benefit from them, particularly people living in resource-poor conditions.

Developers of rehabilitation technology have previously noted the worldwide need for very low-cost rehabilitation devices. For example, several organizations have pursued the development of low-cost prosthetics and wheelchairs (Pearlman et al. 2008). Provision of one these chairs has been documented to improve the lives of people who receive them by increasing independent function and reducing pressure sore frequency (Shore 2008). However, there are relatively few very low-cost technologies to help people with weakened arms exercise their arms. This is an important gap to fill because arm exercise following neurologic injury can improve arm function and help prevent secondary complications such as contractures (van der Lee et al. 2001; Sawaki 2005; Ada et al. 2006; Kloosterman et al. 2008). If a person regains enough arm movement, then he or she may use the limb more frequently in daily life, further training the limb in a positive cycle, whereas if arm function stays below a threshold, a person may not use the limb, and function may decline (Schweighofer et al. 2009). From a pragmatic viewpoint, regaining enough arm strength to push a wheelchair can dramatically improve independence.

People with arm weakness can exercise their arms without technology, but if their arms are severely impaired, such exercise is difficult and compliance with autonomous exercise programs is low. Robotic therapy devices have been designed to provide "assistance-as-needed" to arm movement, mimicking the clinical technique of active assisted exercise (Marchal-Crespo and Reinkensmeyer 2008). Active assistance requires that the patient actively contributes to the movement, a feature of training thought to be important for motor learning and plasticity (Hu et al. 2009). Active assistance also allows patients with a high level of impairment to participate meaningfully in therapy by limiting frustration, increasing motivation, and promoting self-efficacy. Active assistance may also enhance sensory input that drives motor plasticity (Takahashi et al. 2008), and it can demonstrate correct movement patterns that enable better learning (Marchal-Crespo et al. 2009). Robots allow sophisticated forms of assistance to be provided for arm training; however, robotic therapy devices are typically expensive and complex, limiting their widespread use.

Researchers at UC Irvine have previously developed an arm therapy device, T-WREX (Sanchez et al. 2006), now sold as ARMEO, which made use of a spring-balanced arm support rather than robotics to assist arm movement. However, while effective in initial studies with people with stroke (Housman et al. 2009) and multiple sclerosis (Gijbels et al. 2011), ARMEO is still expensive because of the elaborate counterbalancing and link adjusting system, and because of the use of sensors and a computer for feedback. Thus, there is still room for improvement in the development of low cost methods of providing assisted therapy.

In addition to providing active assistance during training, several studies have incorporated the use of virtual environments, such as computer games, during therapy with positive results (Reinkensmeyer et al. 2002). There is speculation that the success of these virtual environments is due to the fact that patients tend to improve more when they are motivated and engaged in their rehabilitation (Sivak et al. 2009). Virtual environments also allow therapy to be largely automated, which reduces the amount of oversight required by a therapist, lowering the cost of the therapy. Furthermore, research has shown that patients tend to improve more when they are given meaningful feedback about their therapy (Dobkin et al. 2010). This type of feedback is inherent to computer games, but lower cost systems using very basic audio and visual feedback schemes have been shown to enhance the effects of therapy as well (Secoli et al. 2011).

3. DESIGN

The primary goal of the design process was to develop a device that could provide active assistance at a low cost. Secondary design goals were the development of computer interfacing, audio feedback, and visual feedback capabilities. The decision to develop a computer interface served the dual purpose of allowing us to record real time data from the device, as well as laying the groundwork for the implementation of virtual environments that patients could interact with. It was important to ensure that the primary goal could be accomplished independent of the success or failure of the secondary goals. Therefore, the design was split into two parallel processes: the development of the core RAE device, and the development of a control box peripheral device as a proof of concept for the future capabilities we selected.

3.1 Resonating Arm Exerciser

RAE is based on two key concepts. The first concept is to use resonance to assist movement. This concept was inspired in part by a previous study that found improved, long-term recovery of arm movement ability when stroke patients rocked themselves in a rocking chair with their impaired arm, which was placed in an air splint, during subacute rehabilitation (Feys et al. 1998; Feys et al. 2004). Sophisticated algorithms have been developed for robotic devices to allow them to learn rhythmic movements such as this and provide assistance in real time (Ronsse et al. 2011). However, a simple resonant system can accomplish this as well; it oscillates with a larger amplitude when it is pushed at its resonant frequency because it stores and releases energy in a manner synergistic to the ongoing movement. A passive resonant system will not move unless pushed, fulfilling the requirement that the exercise be “patient active”. Thus, resonance provides a way for weakened patients to amplify their movements, while still maintaining a causal relationship between amount of effort and size of the resulting movement. Resonance has been used previously with a gait robot to hide the inertia of the robot (Vallery et al. 2010).

The second concept was to integrate the resonant system with an existing low-cost piece of equipment: a manual wheelchair. Many people with arm impairment after stroke or spinal cord injury use wheelchairs, and it is common for people with a neurological injury to spend substantial time in a manual wheelchair during rehabilitation. In addition, as mentioned above, several low-cost wheelchairs have already been developed for use in resource-poor conditions. Our strategy was to reversibly convert a low-cost wheelchair into a therapeutic technology for the severely weak arm, essentially dual-purposing the chair into an exercise device as well as a mobility aid, with the advantages of lower net cost, convenience, accessibility, portability, and reduced need to transfer the patient to another device. The integration of these two concepts allowed us to successfully accomplish our primary design goal.

3.1.1 Detailed Design Description

We created a resonant system by attaching a lever to the wheel of a manual wheelchair and stretching an elastic band from the lever to opposite ends of the wheelchair frame. The resulting arm exercise device, RAE, is shown in Figure 1. When a user pushes on the lever the chair rolls back and forth, storing and releasing energy in the elastic band. If the user pumps the lever at the resonant frequency of the system then his or her arm's active range of motion will increase relative to that possible with a single push. The design was accomplished using only parts readily available from local distributors. The cost of materials for this design was under 40 USD

(Table 1). The wheelchair used in the study was provided by Free Wheelchair Mission; the chair costs 60 USD.

Figure 1: The Resonating Arm Exerciser (RAE) attached to a wheelchair in a right-handed configuration. A patient uses RAE by pushing rhythmically on the lever, rocking the chair at its resonant frequency.

Table 1: Raw materials cost breakdown for final RAE design.

Part Description	**Quantity**	**Unit Cost**	**Total Cost**
Eyebolts	2 EA	$2.76	$5.52
Interlocking Hangers	2 EA	$0.72	$1.44
Rubber-Cushioned Loop Clamp	3 EA	$0.23	$0.69
Vinyl-Coated Steel Tool Holder	2 EA	$1.61	$3.22
Adjustable Bungee Cord	1 EA	$1.80	$1.80
PVC U-Channel	1 foot	$5.02	$5.02
Hinge	1 EA	$0.98	$0.98
2"x2"x3' Aluminum Square Tube	1 EA	$20.07	$20.07
Elastic Band	2 feet	$0.55	$1.10
Total Cost:			**$39.84**

RAE incorporated a three foot long, 2”x2” square aluminum tube with a notch on the bottom that allowed it to pivot on the push rim of a wheelchair and broom handle clamps screwed to the middle of the tube that snapped onto the wheelchair push rim to secure the tube in place (Figure 2). These clamps were placed on both sides of the device, allowing it to be secured to either wheel of the chair for right handed or left handed exercises. An elastic band was placed in tension along the outside of the chair, stretching between a point on the frame near the back and another by the footrest. When RAE was removed from the wheelchair, the elastic band could be tucked inside the arm rest so that it did not interfere with the normal operation of the chair. When RAE was attached, it could clip onto the band at any point along its length as shown in Figure 3. This allowed the neutral position of the device to be easily adjusted. The band then provided resistance when the patient moved RAE away from the neutral position and assistance as they moved towards it.

Figure 2: Left: The lower attachment point of RAE to the push rim of a wheelchair. RAE is able to pivot freely about this joint. Right: The upper attachment point of RAE to the push rim.

Figure 3: The attachment point between RAE and the elastic band. The tab on the elastic band fits securely into the slot on the side of RAE's main shaft and is held in place by the tension of the band. When RAE is not attached, the tab can easily slide to any point along the length of the elastic band where it remains during operation due to the friction of the rubber.

A padded plastic trough was hinged off of the main shaft to support a patient's forearm during therapy. Adjustable elastic bands connected between the main shaft and the forearm trough allowed for the level of weight support of the forearm trough to be adapted to each patient. Velcro straps were used to secure the user's arm in the trough and his or her hand to the main shaft during exercise. The user could use either a standard grip, in which he or she grips the shaft like a glass of water, or a "flat palm" grip as shown in Figure 4, below. Movement of RAE required shoulder flexion/extension, elbow flexion/extension, and wrist flexion/extension.

Figure 4: A participant's hand strapped to RAE in the flat palm grip.

3.1.2 Resonance Analysis

RAE assists a patient in obtaining a larger range of motion (moving further away from the neutral position) if he or she rocks back and forth at the resonant frequency of the system. To see this, approximate the system as a mass-spring-damper system, and assume a person can generate a maximum pushing force on the lever equal to F_{max}. Assume the stiffness of the elastic cords is K. Then the maximum distance the hand moves when the person pushes with maximum force is

$$X_{max} = \frac{F_{max}}{K}$$

Now, if the system is resonant (i.e. the damping ratio $\zeta < .707$), and the person pushes with a force $F = F_{max}\sin(\omega t)$, where ω is the resonant frequency of the system, then the distance the hand moves will be:

$$X_{max} = \frac{F_{max}}{K}\,A$$

where the “movement amplification gain” A is given by:

$$A = \frac{1}{2\zeta\sqrt{1-\zeta^2}}$$

This means that if the person still pushes with strength F_{max}, but at just the right time, periodically, then the amplitude of the hand movement will grow to be A times larger than is possible with just a single maximum push. Note that A depends on the damping ratio ζ which is given by the spring constant K (set by the elastic band), damping C (set by the friction in the system), and mass M (i.e. total inertia of the chair and person) of the system according to:

$$\zeta = \frac{C}{2\sqrt{KM}}$$

Since a small ζ is desirable for a high movement amplification gain, we reduced C by clamping the front wheels of the wheelchair so they always rolled forward/backward. By measuring the impulse response of RAE when attached to the wheelchair with a user seated in the chair, the damping ratio was determined to be about 0.35, yielding a movement amplification gain of 1.5. This means a user could increase his active range of motion or arm movement in the device by up to 50% if he or she rocked at the resonant frequency. Note that the average amplitude of rocking is proportional to the average force applied to the lever. If the user stops pushing, the device stops rocking. Also, it is important for the resonant frequency of the system to be within the physiologic range of human movement (~1 Hz) while still providing an appropriate range of motion for the arm. Resonant frequency is given by:

$$f = \frac{1}{2\pi}\sqrt{\frac{K}{M}}$$

The resonant frequency of RAE is in a physiologic range because the mass to be moved is large, as it includes the user's own weight as the chair rolls.

3.2 Control Box

The additional capabilities of a computer interface, audio feedback, and visual feedback were developed in a single Control Box (CB) shown below in Figure 5. The CB served as a prototype device demonstrating the feasibility of these peripherals, and should not be considered a final design. However, all three capabilities were implemented successfully, thus the development of the CB accomplished our secondary design goals.

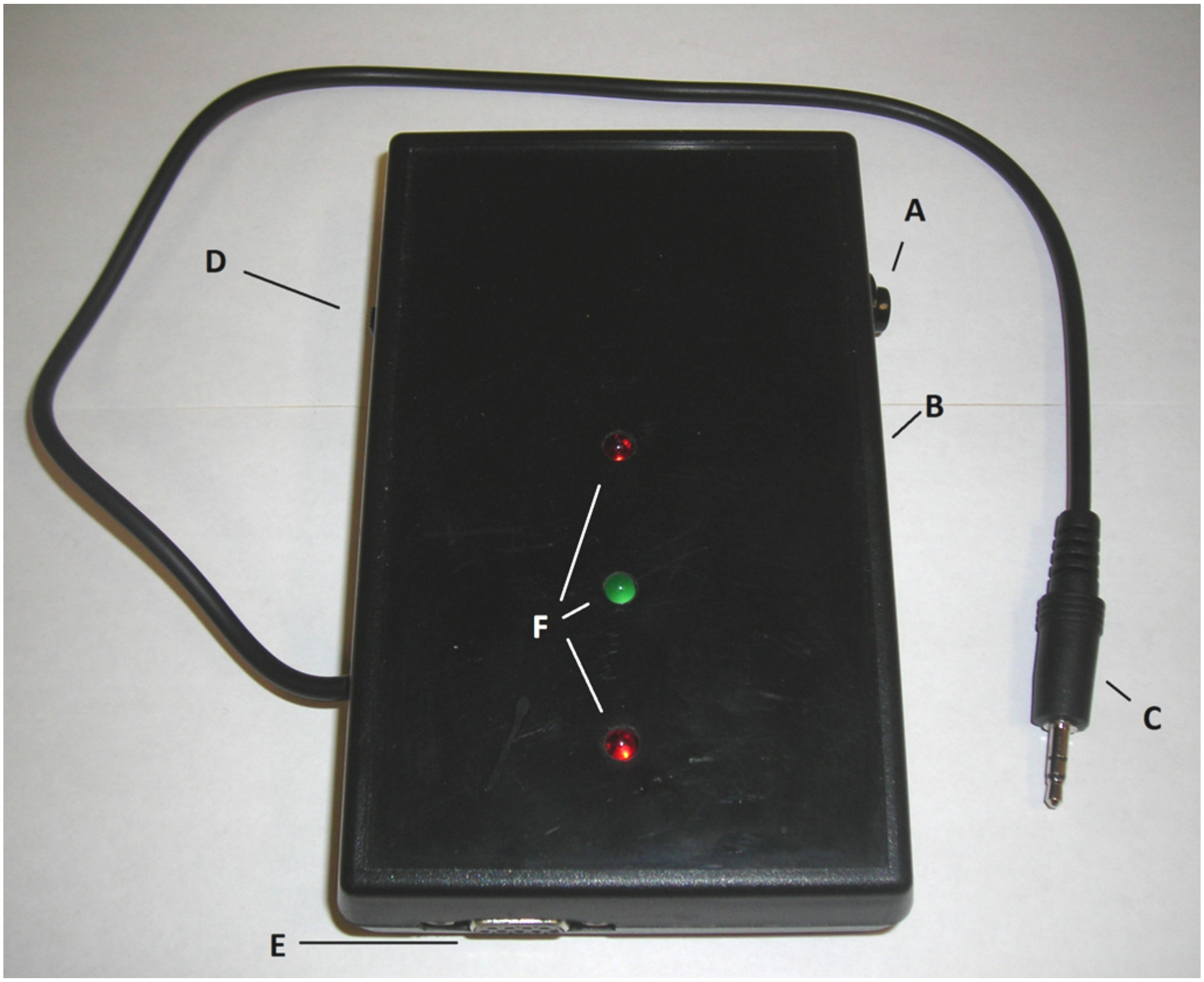

Figure 5: The assembled Control Box (CB). A: User input button; B: 1/8" headphone jack (not visible); C: 1/8" male audio plug; D: On/off switch; E: RS-232 port; F: LED array.

Each of the desired capabilities was developed independently, but they all built off of the same infrastructure that served as the core of the CB. This infrastructure included a PIC 16F876A microcontroller as the central processing unit, a 20 MHz crystal oscillator, a regulated 9 volt battery as the power supply, and a separate ADXL-213 low-g accelerometer unit as a high sensitivity tilt sensor (Figure 6). An on/off switch (Figure 5, D) was added to conserve power, as well as a single push button (Figure 5, A) as an input to the microcontroller, allowing a user to select between the various modes of the device. The CB source code is included in Appendix A.

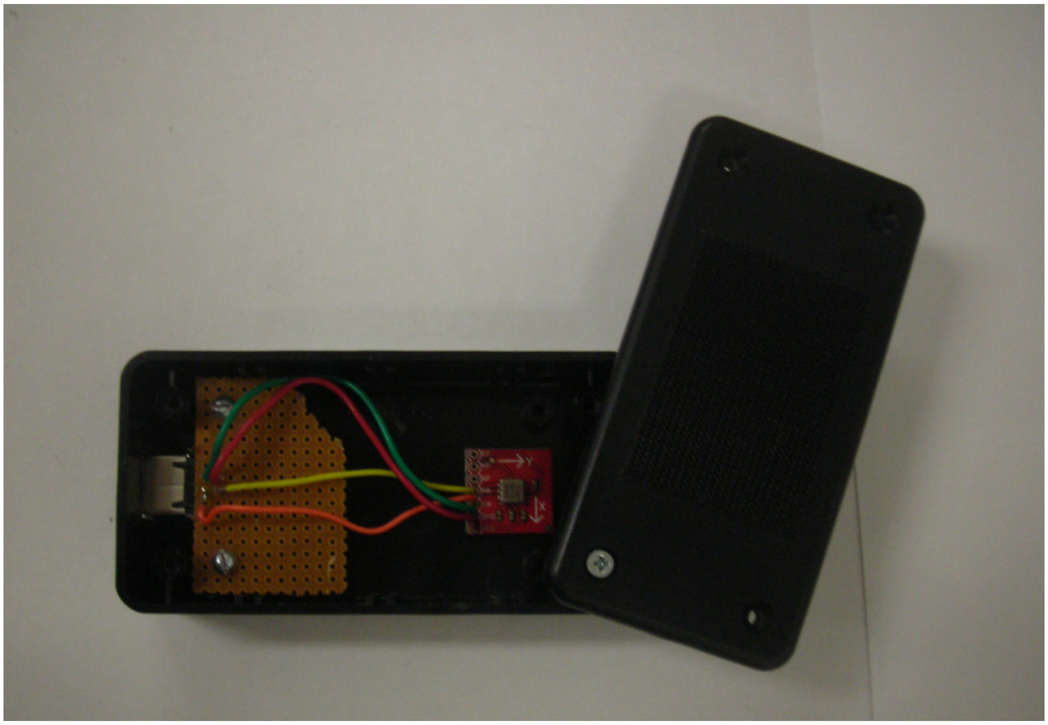

Figure 6: The tilt sensor unit developed to measure the angle of RAE relative to its neutral position during training. The tilt sensor interfaces with the CB via a standard USB cable.

3.2.1 Computer Interface

The computer interface was accomplished using RS-232 serial communication. This was straightforward to implement on the CB side using a MAX232 interface circuit. On the computer side, I selected Microsoft Visual Basic (VB) as my development environment both from previous experience with RS-232 protocol in VB and the ease of creating graphical user interfaces. I then linked the two systems with a USB-to-RS232 converter, and was able to successful transfer data between the CB and a computer.

To further demonstrate the type of virtual environment that could be developed using this interface, I created a basic program patients could interact with during exercise. When the program is started, the window shown in Figure 7 appears on the computer's monitor. The patients then click on the "Start!" button to begin. The red dot, seen on the right in the figure, moves upward along the blue line when the patient pushes RAE forward and downward along the line when the patient moves RAE backward. The "Count" number is incremented by one each time the patient moves through a full rocking motion. While very basic, this program shows how training with RAE can be translated into a virtual environment displayed on a computer screen. The source code is included in Appendix B.

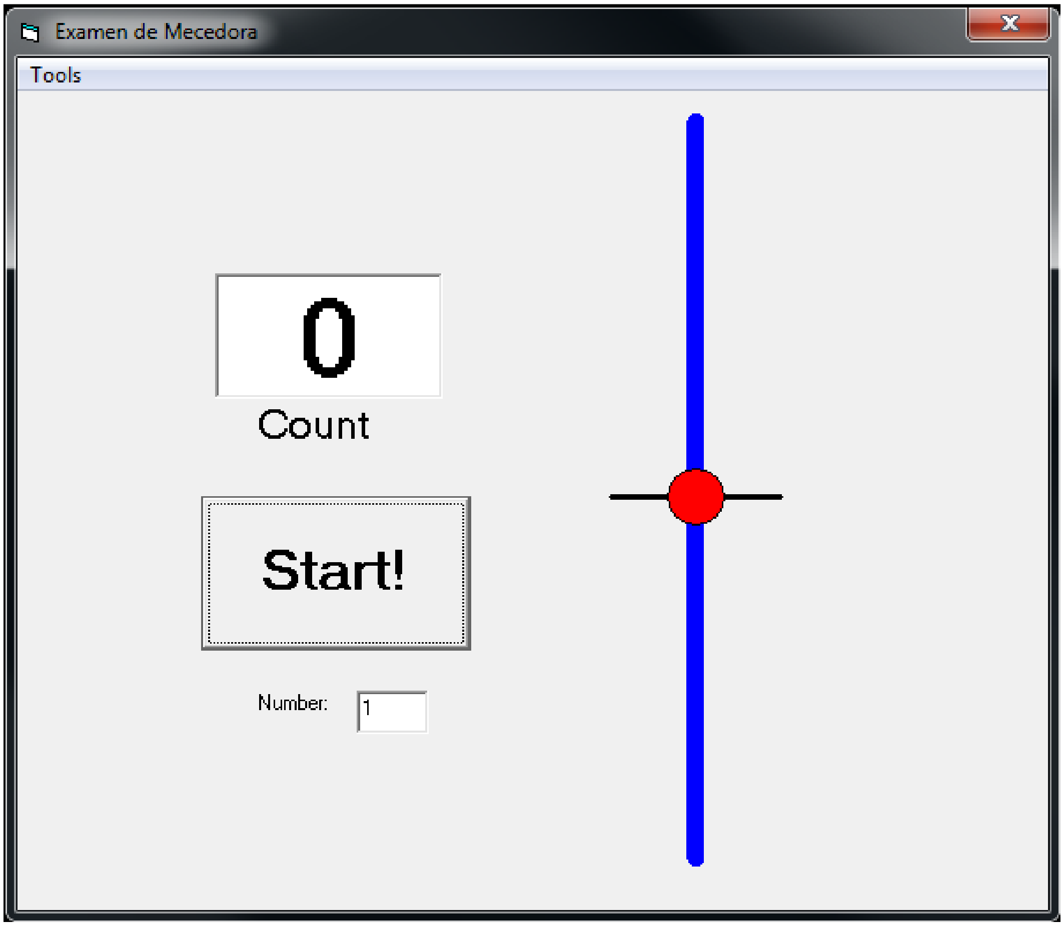

Figure 7: The graphical user interface of the program developed to demonstrate how users of RAE could interact with virtual environments.

3.2.2 Audio Feedback

The main principle behind the audio feedback scheme was to reward the patient sonically for performing the required task. In the case of RAE, that task was to rock consistently for a long period of time. An obvious choice as a reward for performing this task was to play music for the patient. This reward would be improved if the patient could select the music that they wanted to hear. The desired function of the feedback scheme was then to play music for a patient that he or she had selected only while they were rocking in RAE, and to stop the music if they stopped rocking.

The CB could already sense when RAE was being rocked and when it was stopped by reading the value of the tilt sensor, but I needed to design the hardware and software required to start and stop an audio signal at the appropriate times. I accomplished the hardware design by using a 50 kOhm digital potentiometer controlled by the microcontroller as a shunt to ground on the audio line. The output resistance of the circuit was << 50 kOhm, so the signal was not attenuated when the potentiometer was set to its maximum resistance. When the potentiometer was set to zero resistance, it effectively muted the audio signal by shorting it to ground.

For the software, I programmed a state machine in the microcontroller with 4 discrete states: neutral and moving forward, forward, neutral and moving backward, and backward. The microcontroller cycled through the states sequentially, transitioning first from *neutral and moving forward* to *forward* when the voltage of the tilt sensor went above the forward threshold voltage. Then it transitioned into *neutral and moving backward* when the tilt sensor voltage passed back below the forward threshold voltage. Finally it transitioned into the *backward* state and then back to *neutral and moving forward* when the tilt sensor voltage went below, then back above the backward threshold voltage. The microcontroller increased the resistance of the digital potentiometer if it progressed through each state within a predetermined time span. If it remained in any one state for longer than this time span, the resistance of the digital potentiometer was decreased, reducing the volume of the music until it was muted. This algorithm allowed the difficulty of maintaining an audio output to be changed by adjusting either the forward and backward threshold voltages, or the time the microcontroller was allowed to spend in each state before muting the audio signal.

In order to allow a patient to select the music that they heard while rocking, I wired a 1/8" male audio plug to the input line that could be plugged into any standard music playing device (Figure 5, C). I used 1/8" female audio plug on the output line to allow a patient to listen to the music with any standard pair of personal headphones (Figure 5, B). The nominal volume of the audio signal was controlled at the source device of the original sound.

Since I now had direct access to the audio signal being passed to the patient, I developed a method of inserting my own audio tracks into the signal whose volume was independent of the volume of the music signal. This would give therapists the ability to record instructions for use that could be played while the device was stationary. It could also be used to add audio cuing to help patients rock at the resonant frequency, or to play a notification sound when a patient reached a desired ROM.

To accomplish this, I used a SOMO-14D Audio-Sound Module that could be controlled by the microcontroller using simple I2C protocol. Audio tracks simply needed to be loaded onto a micro-SD card and they could be selected in any order for immediate playback. The playback volume could be set to one of 7 discrete values. This provided all of the functionality needed to play custom audio tracks at any time during a therapy session.

The final step was to combine the audio signal from the SOMO-14D module with the audio signal from the patient's music player into the single audio output channel. I implemented this using an op-amp summer circuit, shown in Figure 8. The input resistors ($R_1 - R_n$ in the figure) acted as voltage dividers when coupled with the digital potentiometers, but by selecting the resistance values to match the maximum resistance of the digital potentiometers and introducing a gain of 2 into the circuit I was able to eliminate attenuation of the signal. I also took advantage of the op-amp summer to combine the left and right channels of the input audio signal into a single mono channel.

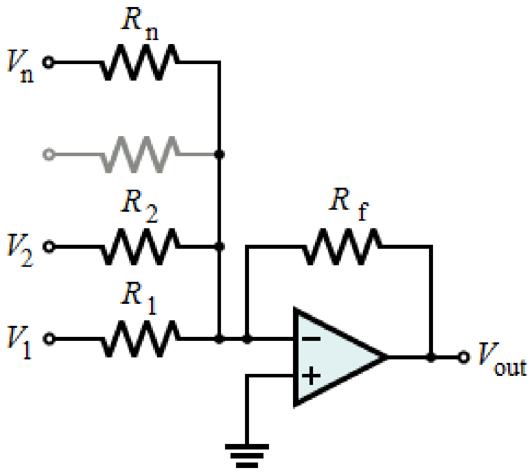

Figure 8: A basic op-amp summer circuit. The input voltages $V_1 - V_n$ are combined linearly into the output voltage, V_{out}.

The limiting factor in this design was the functionality of the SOMO-14D Audio Module. When an audio track was selected by the microcontroller, there was an audible “squeal” in the audio output before the track began. The effect this had was negligible for long audio tracks, but it caused problems in the development of an audio cuing scheme that could play a short “beep” track at the resonant frequency of the device, since the squeal caused by the hardware muted the beep. This limitation could be overcome by various methods, such as playing a long audio track of multiple audio cuing beeps instead of multiple tracks of single beeps.

3.2.3 Visual Feedback

The desired visual feedback scheme was to provide the patient with a visual representation of their range of motion in the device during exercise. This capability was self-contained on the CB by using only the three colored LEDs shown in Figure 5, F as the visual display. The software design piggybacked on the state machine I developed for audio feedback control. I programmed the state machine to turn on the green LED when the microcontroller was in either one of the two neutral states, and one of the two red LEDs when it was in either the forward or backward state. Again, the location of these states in physical space could be controlled by adjusting the forward and backward threshold voltages.

4. CASE STUDY

After the design was completed, I conducted a case study at the Instituto Nacional de Neurología y Neurocirugía (INNN) in Mexico City, Mexico to test the original hypothesis that RAE is feasible as an exercise device and will lead to improvements in arm function. The methodology and results from that case study are presented below.

4.1 Experimental Protocol

We recruited eight volunteers with a stroke from the outpatient population of INNN and they all provided informed consent. Inclusion criteria were > 6 months post injury, moderate to severe arm movement impairment defined as an upper extremity Fugl-Meyer (FM) score < 35 out of 66 (Fugl-Meyer et al. 1975), and willingness to refrain from additional rehabilitation for the upper extremities during the 6 week duration of the study. Demographic information regarding the participants is shown in Table 2.

Table 2: Demographics of the participants in the study.

Patient Demographics			
Total Number of Patients	**8**		
Average Age	**52**	±	**15**
Average # of Months Post Injury	**35**	±	**24**
Gender	*Male*		*Female*
	4		**4**
Type of Stroke	*Ischemic*		*Hemmorrage*
	7		**1**
Side Affected	*Right*		*Left*
	3		***5***

We assigned participants to two groups based on their availability. Participants in the exercise-rest group (n = 3) exercised with the device for a total of 3 consecutive weeks, and then rested for 3 consecutive weeks; participants in the rest-exercise group (n = 5) reversed the order of exercise and rest. During the exercise period, the participants rocked with RAE for a total of six hours in eight forty-five minute sessions. We increased the stiffness of the elastic band after 4 sessions for every participant by stretching the band.

Primary outcome measures were the FM score and an automated measure of active range of motion (ROM) of the arm obtained using RAE. The same non-blinded therapist evaluated FM score at the start and end of both the 3 week rest period and the exercise period, and at a 3 month follow-up evaluation. We quantified active ROM of the arm at the beginning of each training session using the tilt sensor attached to RAE. The calibration relating the tilt sensor voltage to the angle of RAE is included in Appendix C. We asked the participants to rock in the chair fifty times and recorded the angle of RAE relative to the initial position at 50 Hz using the computer interfacing capabilities of the CB. We defined the range of motion as the average amplitude of the angle change during rocking. The participants repeated this test three times per session to establish an average for that day. We obtained two baseline measurements of ROM on two separate days for each subject before they began therapy. Each patient also indicated their arm pain level before and after each session on a visual analog pain scale from 0 to 10, with 0 being no pain, and 10 being the greatest pain possible.

4.2 Results and Analysis

The mean initial FM score for the eight participants was 17 ± 8 out of 66 points; i.e. the participants had substantial arm impairment. The FM score of the Rest-Exercise group did not increase during the rest period (Figure 9), indicating a stable baseline. The mean change in FM score after three weeks of exercise with RAE, averaged across all participants, was 8.5 ± 4.1 points, a significant change (t-test, $p < 0.001$). The FM score of the Exercise-Rest group continued to increase during the rest period, but this increase was not significant (t-test, $p = 0.20$). Figure 10 shows improvements in FM score were sustained at the 3 month follow-up for 6 subjects (t-test, $p = 0.49$). The other two subjects dropped out of the study due to loss of contact.

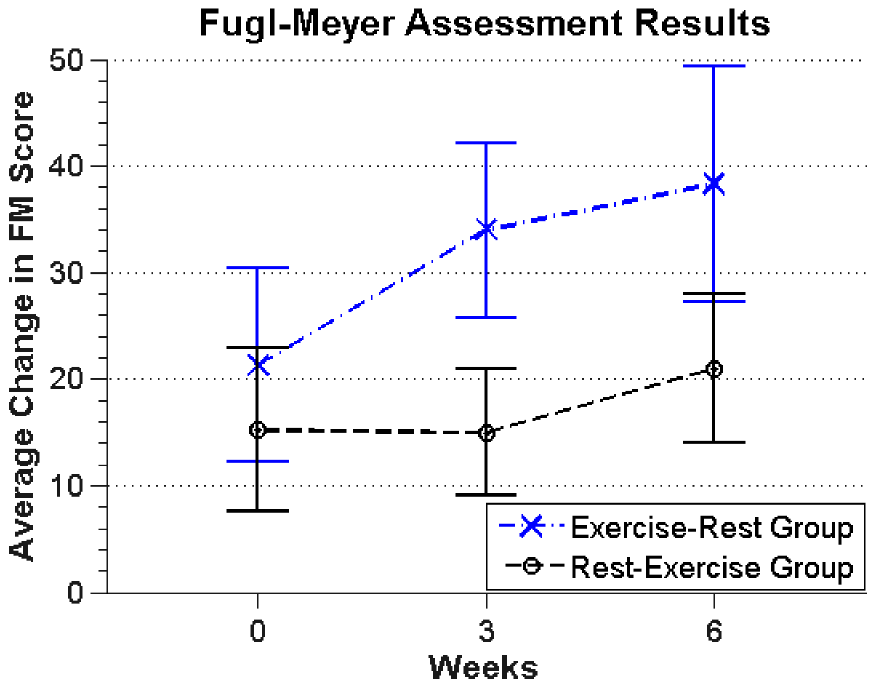

Figure 9: The mean FM scores for the Exercise-Rest (n = 3) and Rest-Exercise (n = 5) groups. Error bars show +/- 1 SD

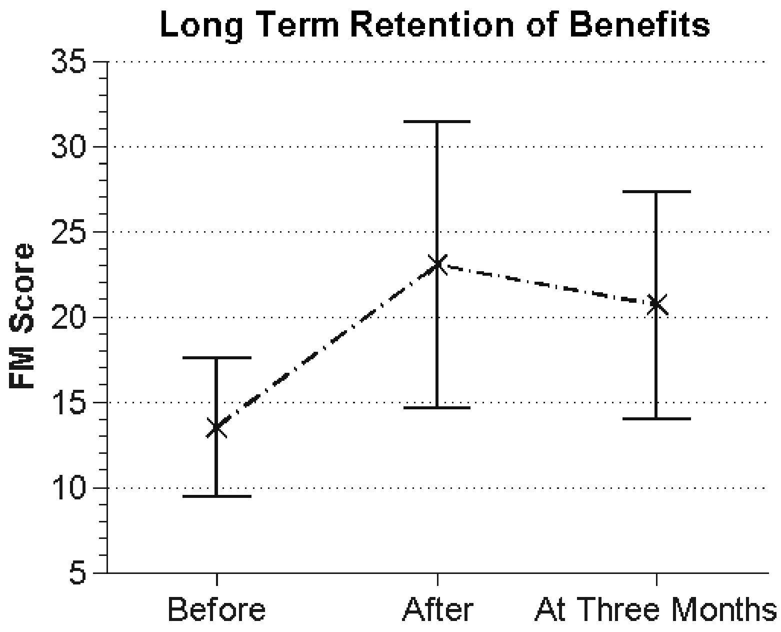

Figure 10: The mean FM scores (n = 6) from before therapy, immediately after therapy, and at a three month follow up assessment. A significant change in FM score was detected before and after therapy (p = 0.006), but no significant change was detected three months later (p = 0.49), although there does appear to be a slight downward trend. Error bars show +/-1 SD.

Average active ROM of the arm measured with RAE improved steadily across the three weeks of exercise (Figure 11), and the average data was well fit by a line with a slope of 1.98 degrees per session (R^2 = 0.80, p = .003). We excluded two subjects from this analysis who had full active ROM along RAE at study start. The overall average increase in active ROM was 14 ± 9.8 degrees, or 66% ± 20%, after three weeks of RAE exercise. Patient rating of arm pain increased slightly by a non-significant amount (p = 0.11) during a single exercise session, but returned to approximately its starting value by the next session (Figure 12).

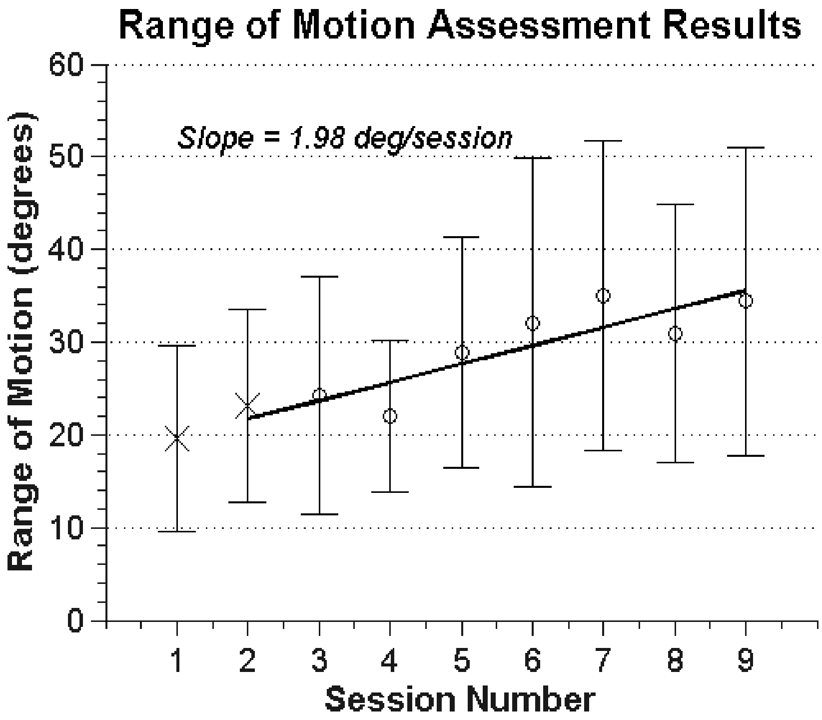

Figure 11: A plot of the mean active ROM for 6 participants (the remaining two participants had full range of motion along the device at study start). Error bars show +/- 1 SD. Each subject had 2 baseline measurements (X) and 7 measurements during training (o). The solid line is the linear regression showing a positive slope of 2.0 degrees per session (R^2 = 0.80, p = .003).

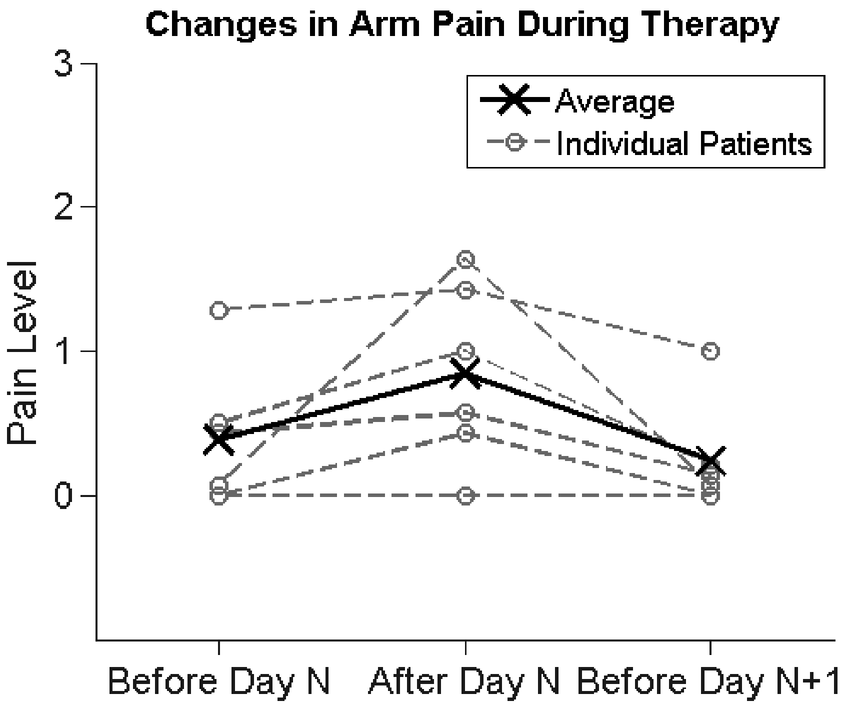

Figure 12: The results of the pain measurements, showing the average perceived levels of pain before a session, after that session, and before the following session. The dashed lines represent each individual patient and the solid line represents the mean values for all 8 subjects.

We analyzed whether the observed changes in FM score correlated with the objective changes in active ROM measured with RAE. This analysis was done for the same six patients included in the ROM analysis above. One of the data sets did not show significant change in ROM, but it was still included for completeness (Figure 13). The changes in FM score due to exercise correlated strongly with the slope of the active ROM curves (CC = 0.89, Figure 14).

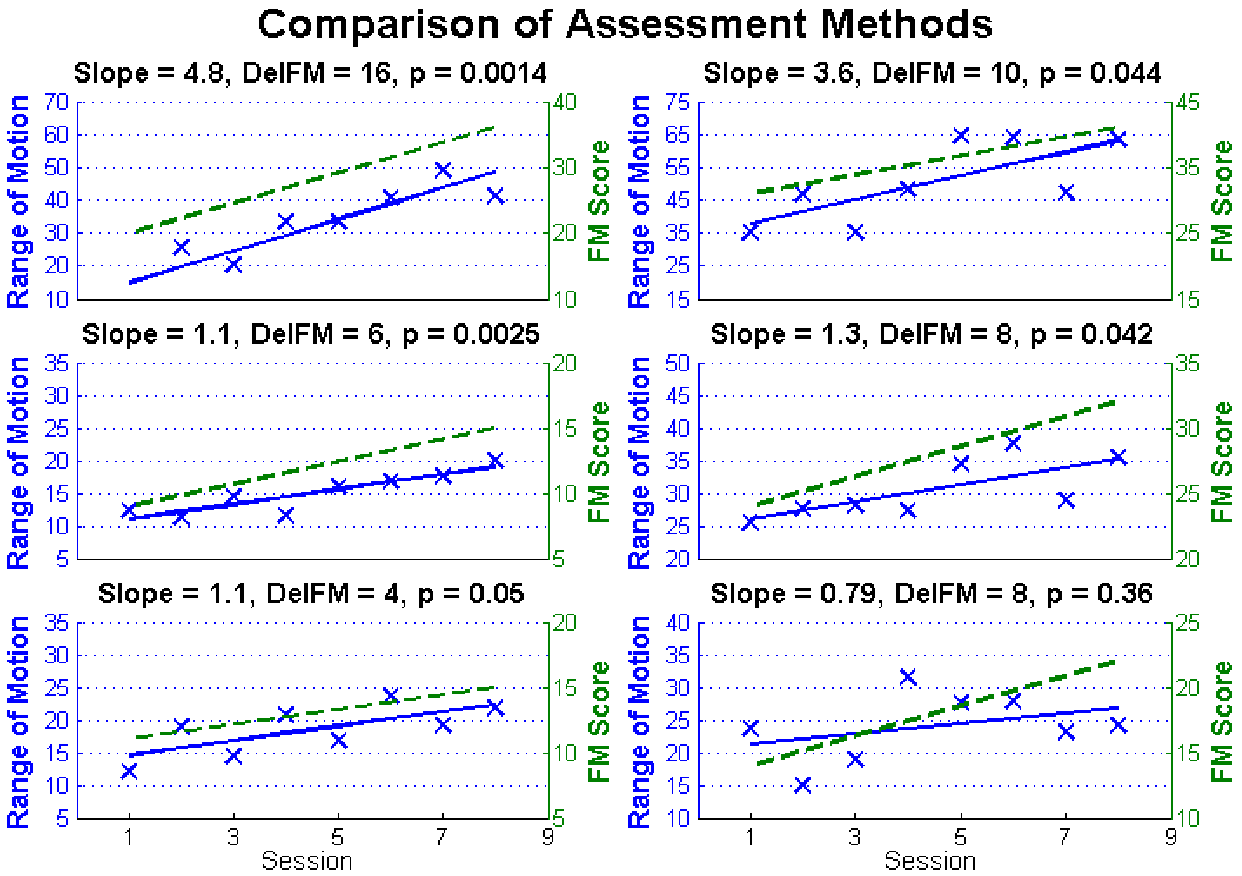

Figure 13: Comparison of the FM and active ROM assessments for 6 participants (the remaining two participants had full range of motion along the device at study start). The solid line represents the regression line for the active ROM data with slope and p-value shown, while the dashed line shows the change in FM score before and after training.

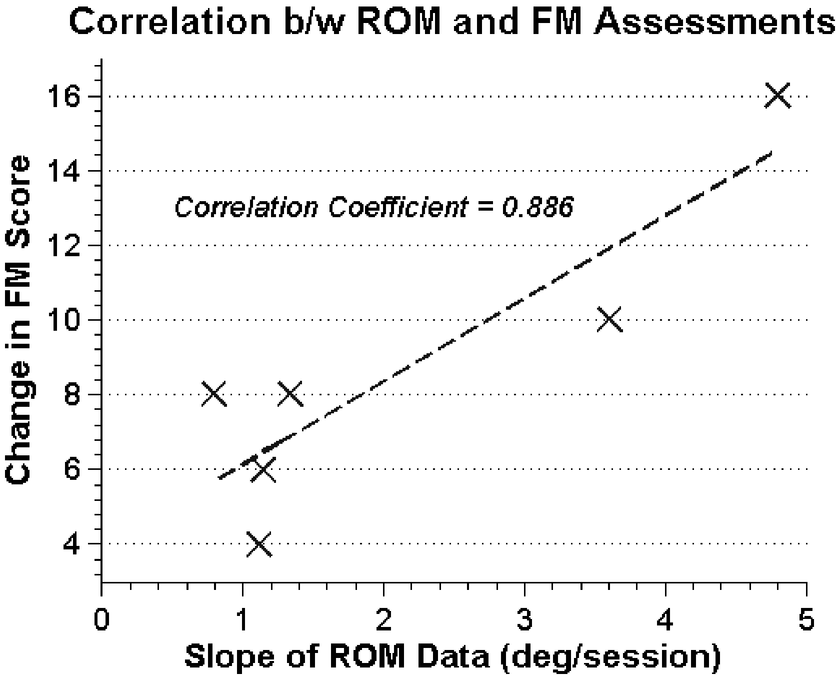

Figure 14: Comparison of the slope of the active ROM data vs. the change in FM score for 6 participants (the remaining two participants had full range of motion along RAE at study start). The dashed line is an estimate of a linear relationship between the two measurements ($R^2 = 0.79$, $p = 0.018$).

5. DISCUSSION AND CONCLUSIONS

We developed RAE, a very low cost device (< 40 USD), that allows people with substantial arm weakness to practice arm movement while receiving mechanical assistance for that movement. The device snaps onto a manual wheelchair push rim, and uses the principle of resonance to amplify movement. We found that people with a chronic stroke who trained with RAE for three weeks significantly improved their arm movement ability, as measured by both the Fugl-Meyer score and an objective assessment of active range of motion of the arm. We also found that this improvement was sustained at a three month follow-up assessment. The limitations of this study include the small population size and the use of a non-blinded therapist performing the Fugl-Meyer assessments. The latter of these two limitations is somewhat offset by the stable baseline observed in the FM scores of the Rest-Exercise group, as well as the strong correlation between the FM scores and the quantitative ROM data.

The use of mechanical resonance to assist movement of the weakened arm is a promising approach. As explained above, resonance requires that the patient be active to keep the system moving, and requires “goal directed” movement, as it penalizes movements that are not precisely timed by resisting them. Resonance also provides assistance much like a robotic therapy device: the amplitude of movement is proportional to the force applied, while the proportionality constant is tunable by altering system parameters (stiffness and damping). If one hypothesized that performing the most number of active repetitions possible in a given time is best for promoting recovery, then working with a resonant system is a good way to achieve many repetitions. Using the mean frequency from the ROM data of 0.87 Hz and an exercise period of 40 minutes, we estimated that the participants performed about 4000 movements per session

(2000 flexions and 2000 extensions), for a total number of at least 32,000 practice movements over the eight exercise sessions. These movements were arguably "non-functional", but the sheer number of movements practiced may have contributed to the observed recovery patients had with RAE.

RAE shares some similarities with other low-cost devices for arm therapy that have also been shown to be effective in reducing arm impairment after stroke. As mentioned earlier, one previous study used a rocking chair as an arm therapy tool (Feys et al. 1998; Feys et al. 2004). The patients rocked themselves with their impaired arm, which was braced in an air splint. RAE is a resonant system like the rocking chair but has several differences. RAE uses a lever to maximize the active range of motion of shoulder flexion/extension, elbow flexion/extension and wrist flexion/extension used for training, while rocking a rocking chair in an air splint requires smaller joint movements. In addition, RAE is easily adjustable; i.e. the resonant frequency and amount of amplification of arm movement of RAE can be adjusted by changing the elastic cord stiffness and damping, which is not possible with a rocking chair. Another difference is that RAE can be attached to a chair the patient is already using, rather than necessarily requiring a transfer.

Other low-cost devices have been developed that require rhythmic motion of the hand along a ramp or slide, in a motion somewhat similar to that used with RAE. For example, the BATRAC provides auditory cueing of rhythmic arm movements on a track using a metronome (Whitall et al. 2000; Luft et al. 2004; Richards et al. 2008). RAE is different in that it provides mechanical assistance using resonance, but similar in that both devices require that the users to try to time their movements. BATRAC does this explicitly by providing an auditory cue. RAE does this implicitly because pumping RAE at a frequency other than the resonant frequency requires greater effort. Auditory cuing could also be added to RAE, as demonstrated by the audio feedback capability of the CB.

The attractive features of RAE are that it costs less than 40 USD, requires only parts available at local hardware stores, and can be assembled with basic tools. RAE requires a manual wheelchair as well; the chair used in this study costs 60 USD. These qualities make it a viable candidate for worldwide use. The design of RAE could also be extended with minimal cost to include visual or audio feedback as a motivational tool for patients and a real-time assessment tool for therapists. Computer interfacing capabilities are also possible, and could be used to create virtual environments for patients to interact with during therapy. Other possible improvements include extensions for leg or hand exercise, a customizable attachment method for any size wheelchair, and a pedometer-like device that could count the repetitions a patient performed during therapy. RAE is a very promising device for the large population of stroke survivors that could benefit from assisted therapy but cannot afford robotic therapy. This is an important need to be met to ensure quality medical care for all people.

REFERENCES

Ada L, Dorsch S, Canning CG (2006) Strengthening interventions increase strength and improve activity after stroke: a systematic review. The Australian journal of physiotherapy 52: 241-248.

Dobkin BH, Plummer-D'Amato P, Elashoff R, Lee J (2010) International randomized clinical trial, stroke inpatient rehabilitation with reinforcement of walking speed (SIRROWS), improves outcomes. Neurorehabil Neural Repair 24:235-242.

Feys H, De Weerdt W, Verbeke G, Steck GC, Capiau C, Kiekens C, Dejaeger E, Van Hoydonck G, Vermeersch G, Cras P (2004) Early and repetitive stimulation of the arm can substantially improve the long-term outcome after stroke: a 5-year follow-up study of a randomized trial. Stroke; a journal of cerebral circulation 35: 924-929.

Feys HM, De Weerdt WJ, Selz BE, Cox Steck GA, Spichiger R, Vereeck LE, Putman KD, Van Hoydonck GA (1998) Effect of a therapeutic intervention for the hemiplegic upper limb in the acute phase after stroke: a single-blind, randomized, controlled multicenter trial. Stroke 29: 785-792.

Fugl-Meyer AR, Jaasko L, Leyman I, Olsson S, Steglind S (1975) The post-stroke hemiplegic patient. 1. a method for evaluation of physical performance. Scand J Rehabil Med 7: 13-31.

Gijbels D, Lamers I, Kerkhofs L, Alders G, Knippenberg E, Feys P (2011) The Armeo Spring as training tool to improve upper limb functionality in multiple sclerosis: a pilot study. Journal of NeuroEngineering and Rehabilitation 8: 5.

Housman SJ, Scott KM, Reinkensmeyer DJ (2009) A Randomized Controlled Trial of Gravity-Supported, Computer-Enhanced Arm Exercise for Individuals With Severe Hemiparesis. Neurorehabilitation and Neural Repair 23: 505-514.

Hu XL, Tong KY, Song R, Zheng XJ, Leung WW (2009) A comparison between electromyography-driven robot and passive motion device on wrist rehabilitation for chronic stroke. Neurorehabil Neural Repair 23: 837-846.

Kloosterman MG, Snoek GJ, Jannink MJ (2008) Systematic review of the effects of exercise therapy on the upper extremity of patients with spinal-cord injury. Spinal cord: the official journal of the International Medical Society of Paraplegia.

Kwakkel G, Kollen BJ, Krebs HI (2007) Effects of Robot-Assisted Therapy on Upper Limb Recovery After Stroke: A Systematic Review. Neurorehabilitation and Neural Repair 22: 111-21.

Luft AR, McCombe-Waller S, Whitall J, Forrester LW, Macko R, Sorkin JD, Schulz JB, Goldberg AP, Hanley DF (2004) Repetitive bilateral arm training and motor cortex activation in chronic stroke: a randomized controlled trial. JAMA : the journal of the American Medical Association 292: 1853-1861.

Marchal-Crespo L, McHughen S, Cramer SC, Reinkensmeyer DJ (2009) The effect of haptic guidance, aging, and initial skill level on motor learning of a steering task. Experimental Brain Research 201: 209-220.

Marchal-Crespo L, Reinkensmeyer DJ (2008) Review of control strategies for robotic movement training after neurologic injury. Journal of Neural Engineering and Rehabilitation 6: 20.

Mehrholz J, Platz T, Kugler J, Pohl M (2008) Electromechanical and robot-assisted arm training for improving arm function and activities of daily living after stroke. Cochrane database of systematic reviews (Online) (4): CD006876.

Pearlman J, Cooper RA, Krizack M, Lindsley A, Wu Y, Reisinger KD, et al. (2008) Lower-limb prostheses and wheelchairs in low-income countries. IEEE Eng Med Biol Mag 27(2): 12–22.

Reinkensmeyer, D.J., Pang, C.T., Nessler, J.A., et al. (2002) Web-based telerehabilitation for the upper extremity after stroke. IEEE Transactions on Neural Systems and Rehabilitation Engineering 10: 102–108.

Richards LG, Senesac CR, Davis SB, Woodbury ML, Nadeau SE (2008) Bilateral arm training with rhythmic auditory cueing in chronic stroke: not always efficacious. Neurorehabilitation and Neural Repair 22: 180-184.

Ronsse R, Vitiello N, Lenzi T, van den Kieboom J, Carrozza M. C., and Ijspeert A. J. (2011) Human-robot synchrony: flexible assistance using adaptive oscillators. IEEE Trans Biomed Eng 58(4): 1001-1012.

Sanchez RJ, Liu J, Rao S, Shah P, Smith R, Cramer SC, Bobrow JE, Reinkensmeyer DJ (2006) Automating arm movement training following severe stroke: functional exercises with quantitative feedback in a gravity-reduced environment. IEEE Transactions on Neural and Rehabilitation Engineering 14: 378-389.

Sawaki L (2005) Use-dependent plasticity of the human motor cortex in health and disease. IEEE Eng Med Biol Mag 24: 36-39.

Schweighofer N, Han CE, Wolf SL, Arbib MA, Winstein CJ (2009) A functional threshold for long-term use of hand and arm function can be determined: predictions from a computational model and supporting data from the Extremity Constraint-Induced Therapy Evaluation (EXCITE) Trial. Phys Ther 89: 1327-1336.

Secoli R, Milot MH, Rosati G, Reinkensmeyer D (2011) Effect of visual distraction and auditory feedback on patient effort during robot-assisted movement training after stroke, Journal of NeuroEngineering and Rehabilitation 8: 21.

Shore SL (2008) Use of an economical wheelchair in India and Peru: Impact on health and function. Med Sci Monit 14: 71-79.

Sivak M, Mavroidis C, Holden MK (2009) Design of a low cost multiple user virtual environment for rehabilitation (MUVER) of patients with stroke. Stud Health Technol Inform 142: 319–324.

Takahashi CD, Der-Yeghiaian L, Le V, Motiwala RR, Cramer SC (2008) Robot-based hand motor therapy after stroke. Brain: a journal of neurology 131: 425-437.

Vallery H, Duschau-Wicke A, Riener R (2010) Hiding robot inertia using resonance. Conf Proc IEEE Eng Med Biol Soc 1: 1271-1274.

van der Lee J, Snels I, Beckerman H, Lankhorst G, Wagenaar R, Bouter L (2001) Exercise therapy for arm function in stroke patients: a systematic review of randomized controlled trial. Clinical Rehabilitation 15: 20-31.

Whitall J, McCombe Waller S, Silver KH, Macko RF (2000) Repetitive bilateral arm training with rhythmic auditory cueing improves motor function in chronic hemiparetic stroke. Stroke; a journal of cerebral circulation 31: 2390-2395.

APPENDIX A: CONTROL BOX SOURCE CODE

```
//PROGRAM for control board that reads tilt sensor, plays music
//
// PROCESSOR : PIC16F876A
// CLOCK      : 20MHz, EXTERNAL
// SPEED     : 9600 bps(1bit=104us)

#include     <pic.h>
#include     <htc.h>
#include     "m_i2c_1.h"


#define XTAL_FREQ 20MHZ                     //-- Define the crystal frequency
#define BITNUM(adr, bit)       ((unsigned)(&adr)*8+(bit))
#define I2CLOW  0                           //-- Puts pin into output/low mode
#define I2CHIGH 1                           //-- Puts pin into Input/high mode

#ifndef MHZ
 #define     MHZ    *1000                   /* number of kHz in a MHz */
#endif
#ifndef KHZ
 #define     KHZ    *1                      /* number of kHz in a kHz */
#endif

//unsigned char delayus_variable;
#define PIC_CLK 20000000
#include     "delay.h"

//********* I2C Bus Timing - uS ************
#define I2CSTARTDELAY 50
#define I2CSTOPDELAY  50
#define I2CDATASETTLE 20
#define I2CCLOCKHIGH  100
#define I2CHALFCLOCK  50
#define I2CCLOCKLOW   100
#define I2CACKWAITMIN 50



__CONFIG(WDTDIS & LVPDIS & BORDIS & HS & PWRTEN) ;

unsigned char ch;
static bit SCL      @ BITNUM(PORTC,3);      //-- The SCL output pin
static bit SCL_TRIS @ BITNUM(TRISC,3);      //-- The SCL Direction Register Bit
static bit SDA      @ BITNUM(PORTC,4);      //-- The SDA output pin
static bit SDA_TRIS @ BITNUM(TRISC,4);      //-- The SDA Direction Register Bit
static bit SOMO_CLK @ BITNUM(PORTB,2);
static bit SOMO_DAT @ BITNUM(PORTB,4);
static bit LED_F   @ BITNUM(PORTA,2);       //-- The Forward LED
static bit LED_M   @ BITNUM(PORTC,0);       //-- The Middle LED
static bit LED_B   @ BITNUM(PORTC,1);       //-- The Back LED
static bit RheoDC  @ BITNUM(PORTB,0);       //-- DC for Rheostat
static bit RheoUC  @ BITNUM(PORTC,5);       //-- UC for Rheostat

//Just simple delay
void Delay(unsigned long cntr) {
      while (--cntr != 0);
}
```

```
//*****************************************************RS232 CODE
void InitUsart(void) {

        // TX Pin - output
        TRISC6 = 0;

        // RX Pin - input
        TRISC7 = 1;

        // RX Setting, 8bit, enable receive,
        RCSTA = 0x90;

        // TX Setting, 8bit, Asynchronous mode, High speed
        TXSTA = 0x24;

        // Set Baudrade - 9600 (from datasheet baudrade table)
        SPBRG = 129;

}

void  WriteByte(unsigned char byte) {

        // wait until register is empty
        while(!TXIF);

        // transmite byte
        TXREG = byte;
}

unsigned char ReadByte(void) {

        // wait to receive character
        while(!RCIF);

        // return received character
        return RCREG;
}


//******************************************SOMO-14D CODE**************
void WriteSomo(unsigned short data) {
        int i;
        TRISB2 = 0;                         //Set CLK as output
        TRISB4 = 0;                         //Set DATA as output

        //send start bit
        SOMO_CLK = 0;
        DelayMs(2);                         //wait 2 ms

        //clock in the first 15 bits
        for (i=0;i<15;i++) {
                if ( (data & 0x80)== 0) {
                        SOMO_DAT = 0;
                } else {
                        SOMO_DAT = 1;
                }
                SOMO_CLK = 1;               //pulse the clock for 200 us
                DelayUs(200);
                SOMO_CLK = 0;
```

```
            data = data<<1;                 //shift the data word over
      }

      //clock in the 16th bit
      if ( (data & 0x80)== 0) {
                   SOMO_DAT = 0;
      } else {
                   SOMO_DAT = 1;
      }
      SOMO_CLK = 1;                         //pulse the clock for 200 us
}

/*---------------------------------------------------------------
      Reads the ADC level input on a specified ADC channel.
      Takes in an 10-bit ADC channel number.
      Returns an 10 bit number that signifies this level.
      Approximate sampling time = 76.8us
      ---------------------------------------------------------------*/
unsigned int ReadADC(unsigned char ADC_Channel){

      volatile unsigned int ADC_VALUE;

      /* Selecting ADC channel */
  ADCON0 = (ADC_Channel << 3) + 1;          /* Enable ADC, Fosc/32 */

      ADIE     =    0;                      /* Masking the interrupt */
      ADIF     = 0;                         /* Resetting the ADC interupt bit */

      ADRESL=      0;                 /* Resetting the ADRES value register */
      ADRESH=      0;

  DelayUs(150);

  ADGO = 1;
      /* Staring the ADC process */
  while(!((ADCON0&0x04)==0)) continue;      /* Wait for conversion complete */


      ADC_VALUE    =       ADRESL;          /* Getting HSB of CCP1 */
      ADC_VALUE    += 256*(ADRESH&0x03);    /* Getting LSB of CCP1 */

  return (ADC_VALUE);     /* Return the value of the ADC process */
}


/////////////////////////////////////////////////////////////
// Reads one byte from the EEprom at the specified address //
// and returns it                                          //
/////////////////////////////////////////////////////////////
unsigned char ReadByteFromEE(const unsigned char address)
{
      unsigned char byte;             // Variable hold the data that is read

      EEADR = address;                // Read from this address

      EEPGD = 0;                      // Point to EE memory
      RD     = 1;                     // Initiate a read cycle

      byte = EEDATA;                          // Fetch byte from dataregister
      return byte;                            // Return the read byte
```

```
}

////////////////////////////////////////////////////////////
// Writes one byte to the EEprom at the specified address //
////////////////////////////////////////////////////////////
void WriteByteToEE(unsigned char data, const unsigned char address)
{
      EEADR  = address;                        // Address to write to
      EEDATA = data;                           // Data to write

      WREN   = 1;                              // Enable writes to the EEProm
      GIE    = 0;                              // Disable interrupts during write

      EECON2 = 0x55;                           // Write "password" to EECON2
      EECON2 = 0xAA;
      WR     = 1;                              // Initiate a write cycle

      while(!EEIF);                            // Wait for write to complete
      WREN   = 0;                              // Disable writes to EEProm
      EEIF   = 0;                              // Clear "write complete" flag
}


// main function
void main( void ) {

      //variable declarations:
      int Select;
      double Difficulty=1;

      int ControlState=1;
            //State 1 = EasyMode, 2 = HardMode, 3 = ExpertMode, 4 = UARTMode

      unsigned char Session_Num;
      unsigned char Session_Count;

      int count=0;
      char count_hi;
      char count_lo;

      int i;

      int    timeout=0;
      int Arm_State=1;          //State 1 = Rest, 2 = Forward reach, 3 = Passing
middle on return, 4 = Backward reach

      unsigned int Yvoltage;
      unsigned int Yref;
      unsigned int Yref_check;

      //double Xvoltage;
      //double Xref;
      //double Xref_check;

//Startup Routine
begin:
      INTCON       = 0x0;               // Disable inerupt
      CMCON  = 0x07;                    // Comparators off
      ADCON1 = 0b11000100;      // use this for ADC (Right Justified, 32 Tosc)
      CVRCON  = 0;                      // Not sure what this does...
```

```
TRISA2       = 0;               // Forward LED pin as output
TRISC0       = 0;               // Middle LED pin as output
TRISC1       = 0;               // Back LED pin as output

SCL_TRIS     = 0;               // EEPROM CLK as output

TRISA5       = 1;               // Button as input

TRISA1   = 1;       //Set ADC channels as inputs (only using Y right now)
TRISA3   = 1;

TRISB0       = 0;               // UC as output
TRISC5       = 0;               // DC as output

TRISB1       = 0;               // SOMO_Reset as output

EEPGD        = 0;               // Point to EE memory

//turn off Sound
RheoDC = 0;                     //DC low
RheoUC = 1;                     //UC high

//Blink LEDs
LED_M = 0;
LED_B = 1;
LED_F = 0;
DelayMs(250);
LED_B = 0;
LED_M = 1;
DelayMs(250);
LED_M = 0;
LED_F = 1;
DelayMs(250);
LED_F = 0;

//reset SOMO
PORTB &= ~0x02;
DelayMs(10);
PORTB |= 0x02;

//Grab the current Session #
Session_Num = ReadByteFromEE(0x00);   //read the value stored at 0x0000

//grab the reference voltage for the rest position, do some averaging to
      make sure its correct (blink LED during process)
Yref = ReadADC(1);
      while(1) {
             LED_M = 1;
             DelayMs(1);
             LED_M = 0;
             Yref_check = ReadADC(1);
             if ((Yref_check > Yref-1)&(Yref_check < Yref+1)) {
                   Yref = (Yref + Yref_check)/2;
                   break;
             } else {
                   Yref = Yref_check;
             }
      }
```

```
//Turn on the bottom light, to indicate Easy difficutly selected
LED_B = 1;


// wait for button press or arm movement
while(1) {
    Yvoltage = ReadADC(1);
    Select = 0;
    if ((PORTA&0x20)==0) {
        Select = 1;
    } else if  (Yvoltage < Yref-5) {
        break;
    }

    // if the button is pushed, go to next state
    if (Select==1){
        ControlState++;
        if (ControlState == 2) {
            Difficulty = 1.5;
            LED_B = 0;
            LED_M = 1;
            LED_F = 0;
            DelayMs(200);
        } else if (ControlState == 3) {
            Difficulty = 2;
            LED_B = 0;
            LED_M = 0;
            LED_F = 1;
            DelayMs(200);
        } else if (ControlState == 4) {
            int command=0;

            //Turn on all LEDs
            LED_B = 1;
            LED_M = 1;
            LED_F = 1;

            // Init Interface
            InitUsart();

            // Send a byte to user
            // WriteByte('?');

            while(1) {
                ch = ReadByte();
                if (ch=='d') {              //download command
                    command = 1;
                } else if (ch=='u') {       //erase command
                    command = 2;
                } else if (ch=='q') {       //quit UART mode command
                    command = 3;
                } else if (ch==1) {         //run test command
                    command = 4;
                }

                if (command==1) {
                    char info;
                    Session_Count = 1;
                    WriteByte('S');
                    Session_Num = ReadByteFromEE(0x00);
```

```
                        WriteByte(Session_Num);
                        WriteByte('#');
                        for (i=0;i<Session_Num;i++) {
                            info = ReadByteFromEE(Session_Count);
        //read the value stored at 0x0000 + SC*2 (Count_Hi)
                            WriteByte(info);
                            info = ReadByteFromEE(Session_Count +
1);     //read the value stored at 0x0001 + SC*2 (Count_Lo)
                            WriteByte(info);
                            Session_Count = Session_Count + 2;
                            WriteByte(',');
                        }
                        command = 0;
                    } else if (command==2) {
                        char info;
                        Session_Count = 0;
                        WriteByte('Y');
                        WriteByte('?');
                        ch = ReadByte();
                        Session_Num = 1;
                        if (ch == 'y') {
                            //clear the session number
                            WriteByteToEE(0x01,0x00);
                            //clear counts for previous sessions
                            WriteByteToEE(0x00,0x01);
                            WriteByteToEE(0x00,0x02);

                        }
                        command = 0;
                    } else if (command==3) {
                        Difficulty = 1;
                        LED_B = 1;
                        LED_M = 0;
                        LED_F = 0;
                        ControlState = 1;
                        break;
                    } else if (command==4) {
                        int test_count=0;
                        char out_voltage;
                        int WaitForHalf = 1;
                        int WaitForFull = 0;
                        while(1) {
                            Yvoltage = ReadADC(1);
                            out_voltage = (char)(Yvoltage - Yref
+ 127);

                            WriteByte(out_voltage);
                            if (WaitForHalf == 1) {
                                if (out_voltage > 127 + 4) {
                                    WriteByte(1);
                                    WaitForHalf = 0;
                                    WaitForFull = 1;
                                }
                            } else if (WaitForFull ==1) {
                                if (out_voltage < 127 - 3) {
                                    WriteByte(0);
                                    WaitForFull = 0;
                                    WaitForHalf = 1;
                                    test_count++;
                                }
                            }
```

```
                                      if (test_count > 49) {
                                             WriteByte(255);
                                             break;
                                      }
                                      DelayMs(20);
                                }
                          }
                   }
             }
      }
}


//WriteSomo(0x0000);                  //play instructional audio file???
//DelayMs(4);
//while(1){
//     if ((PORTB&0x20)==0) {
//            break;
//     }
//}

while (1) {

      Yvoltage = ReadADC(1);

      if (Yvoltage < Yref - 13*Difficulty) {       //here the arm is forward
             LED_F = 1;
             LED_M = 0;
             LED_B = 0;
             DelayMs(100);            //this prevents fluttering
             if (Arm_State == 1) {
                   Arm_State = 2;
                   timeout=0;
                   RheoUC = 0;
                   RheoDC = 1;
             }
      } else if (Yvoltage > Yref + 6*Difficulty)   { //reached back limit
             LED_F = 0;
             LED_M = 0;
             LED_B = 1;
             DelayMs(100);            //prevents fluttering
             if (Arm_State == 3) {
                   Arm_State = 4;
                   timeout=0;
                   RheoUC = 0;
                   RheoDC = 1;
             }
      } else if ((Yvoltage > Yref - 1)&&(Yvoltage < Yref + 1)) {
             //here in middle section
             LED_F = 0;
             LED_M = 1;
             LED_B = 0;
             DelayMs(100);                   //prevents fluttering
             if (Arm_State == 2) {
                   Arm_State = 3;
                   timeout=0;
                   RheoUC = 0;
                   RheoDC = 1;
             } else if (Arm_State == 4) {
                   Arm_State = 1;
```

```
                timeout=0;
                RheoUC = 0;
                RheoDC = 1;
                count++;
                //flash LEDs to indicate a cycle
                LED_M = 0;
                DelayMs(100);
                LED_M = 1;
                DelayMs(100);
                LED_M = 0;
                DelayMs(100);
                LED_M = 1;
            }
        }
       if (timeout>40) {
            //here patient stopped exercising for ~3 sec, turn off sound
            RheoUC = 1;
            RheoDC = 0;
        }
        if ((PORTA&0x20)==0) {
            //here they pushed button, stop the training to store the count
            break;
        }

        timeout++;                  //increment the timeout counter
    }


    //turn off the iPod sound
    RheoUC = 1;
    RheoDC = 0;

    //turn off LEDs
    LED_B = 0;
    LED_M = 0;
    LED_F = 0;

    //now store the count for this session
    Session_Count = (Session_Num*2)-1;
    count_hi = count/256;
    count_lo = count%256;
    //write the hi-byte of the count (b/c count could be >255, needs 2 bytes)
    WriteByteToEE(count_hi,Session_Count);
    //write the lo-byte of the count here
    WriteByteToEE(count_lo,Session_Count+1);
    Session_Num=Session_Num+1;
    WriteByteToEE(Session_Num,0x00);

    //WriteSomo(0x0003)         //maybe play outro sound file???
    //DelayMs(4);
    //while(1){
    //    if ((PORTB&0x20)==0) {
    //        break;
    //    }
    //}

    goto begin;

}
```

APPENDIX B: VISUAL BASIC SOURCE CODE

```
Option Explicit

' Define the Sleep function from the Win32 API
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

'Public Variables
Dim MAINR$, Data$, ReceivedData$, TotalData$, TimeData$
Dim ReceivedDataLen, StartTime As Long
Dim FrontScore, BackScore, TrialNumber, TotalDataLen As Integer



Private Sub Calibration_Click(Index As Integer)
    Dim Angle$, InputValue$, dump$
    Dim InVal As Double
    Dim N, RetVal As Integer

    ' Reset the serial input buffer and send the command to trigger the
            device into outputting data
    MSComm1.InBufferCount = 0
    MSComm1.Output = Chr$(1)

    'Open up the file to write
    Open MAINR$ + Format(Date$, "d-mmm") + "_Calibration.txt" For Output As
            #1

    'Want to grab 10 angles
    For N = 1 To 6
        'Wait for user to be ready
        RetVal = MsgBox(Str$(N), 0, "Position Chair")

        'Grab the value at this angle
        dump$ = MSComm1.Input
        While MSComm1.InBufferCount < 0
        Wend
        InputValue$ = Right$(MSComm1.Input, 1)
        InVal = Val(Str$(Asc(InputValue$)))

        'Save this pair
        Print #1, Str$(N) + "    " + Str$(InVal) + " " + Str$(InVal - 127)
    Next N

    Close #1

End Sub

Private Sub Form_Load()
    Dim Temp1$, Temp2$
    Dim PortCount, Port, SelectedPort, RetVal As Integer

    'Find the save folder
    Temp1$ = App.Path
    If Right$(Temp1$, 1) <> "\" Then Temp1$ = Temp1$ + "\"
    MAINR$ = Temp1$ + "Output\"   ' Save folder

    'Open the COMM port
    Port = 3                    ' Default Com Port
```

```
    SelectedPort = Port
    If MSComm1.PortOpen = True Then
      MSComm1.PortOpen = False
    End If
    On Error GoTo porterror
    PortCount = 0
OPENPORT:
    MSComm1.CommPort = Port
    MSComm1.Settings = "9600,N,8,1"
    MSComm1.PortOpen = True
    If Port <> SelectedPort Then
      Temp1$ = "Serial Port Error"
      Temp2$ = "The selected Serial Port is unavailable.  However, Serial
                   Port " + Str$(Port)
      Temp2$ = Temp2$ + " has been located. Do you wish to use the port?  A
                   no answer terminates the application."
      RetVal = MsgBox(Temp2$, 4, Temp1$)
      If RetVal <> 6 Then End
    End If

    'Initialize Settings and Variables
    Form1.Top = 0

    Exit Sub

porterror:
    PortCount = PortCount + 1
    If PortCount > 8 Then
       Temp1$ = "Serial Port Error"
       Temp2$ = "The selected Serial Port is unavailable.  Please check your"
       Temp2$ = Temp2$ + " computer's serial port setup and try again later."
       RetVal = MsgBox(Temp2$, 0, Temp1$)
       Exit Sub
    End If
    If Port > 1 And PortCount = 1 Then Port = 1 Else Port = Port + 1
    If Port > 8 Then Port = 1
    Resume OPENPORT

End Sub


Private Sub Command1_Click()
    Dim Count, N As Integer
    Dim outfile$


    If Timer1.Enabled = False Then
        ' Reset the serial input buffer and send the command to begin
        MSComm1.InBufferCount = 0
        MSComm1.Output = Chr$(1)

        'Clear variables
        ReceivedData$ = ""
        Data$ = ""
        TotalData$ = ""
        TimeData$ = ""
        TotalDataLen = 1
        StartTime = 0
        ReceivedDataLen = 0
        FrontScore = 0
```

```
        BackScore = 0

        'Grab the trial number
        TrialNumber = Val(TrialNum.Text)

        'Begin
        StartTime = Timer
        Timer1.Enabled = True
        Command1.Caption = "Alto"
    ElseIf Timer1.Enabled = True Then
        'Stop
        Timer1.Enabled = False
        Command1.Caption = "Empieza"

        'Write normalized TotalData with Time info to file
        outfile$ = Format(Date$, "d-mmm")
        outfile$ = outfile$ + "_TimedData_" + Str$(TrialNumber) + ".txt"
        Open MAINR$ + outfile$ For Output As #1
        For N = 0 To Len(TotalData$) - 1
            Print #1, (Mid$(TimeData$, N * 7 + 1, 7)) + "  " +
Str$(Val(Str$(Asc(Mid$(TotalData$, N + 1, 1)))) - 127)
        Next N
        Close #1

        Count = Val(CountText.Text)

        'Write Scores to file
        outfile$ = Format(Date$, "d-mmm")
        outfile$ = outfile$ + "_Scores_" + Str$(TrialNumber) + ".txt"
        Open MAINR$ + outfile$ For Output As #1
        Print #1, "Average Max Bit Change Backward = "
        Print #1, Str$(BackScore / Count)
        Print #1, "Average Max Bit Change Forward = "
        Print #1, Str$((-1) * FrontScore / Count)
        Print #1, "Total Time in Seconds = "
        Print #1, Str$(Timer - StartTime)
        Close #1

        'Increment Trial Number
        TrialNumber = TrialNumber + 1
        TrialNum.Text = Str$(TrialNumber)
    End If

End Sub


Private Sub Timer1_Timer()
    Dim Temp1$, Temp2$, outfile$
    Dim Count, RetVal As Integer
    Dim M, N, CountFlagMark, HalfFlagMark As Long
    Dim Max, Min, Value, NewDataLen As Double
    Dim Score As Integer

    If MSComm1.InBufferCount > 0 Then
        ReceivedData$ = MSComm1.Input
        ReceivedDataLen = Len(ReceivedData$)

        'Look for the markers
        CountFlagMark = InStr(ReceivedData$, Chr$(0))
        HalfFlagMark = InStr(ReceivedData$, Chr$(1))
```

```
'Just finished a full cycle
If CountFlagMark <> 0 Then
    'Increment Count
    Count = Val(CountText.Text)
    Count = Count + 1
    CountText.Text = Str$(Count)

    'Grab the remaining data for this cycle and find the max
    Data$ = Data$ + Left$(ReceivedData$, CountFlagMark - 1)
    TotalData$ = TotalData$ + Left$(ReceivedData$, CountFlagMark - 1)
    Max = 0
    For N = 1 To Len(Data$)
        Value = Val(Str$(Asc(Mid$(Data$, N, 1))))
        If Value > Max Then Max = Value
    Next N
    BackScore = BackScore + Max - 127  'for now, just store raw bit
                                                 difference

    'Reset the Data string to look for the min
    Data$ = Right$(ReceivedData$, ReceivedDataLen - CountFlagMark)
    TotalData$ = TotalData$ + Right$(ReceivedData$, ReceivedDataLen -
           CountFlagMark)

'just finished a  half cycle
ElseIf HalfFlagMark <> 0 Then
    'Grab the remaining data for this cycle and find the min
    Data$ = Data$ + Left$(ReceivedData$, HalfFlagMark - 1)
    TotalData$ = TotalData$ + Left$(ReceivedData$, HalfFlagMark - 1)
    Min = 0
    For N = 1 To Len(Data$)
        Value = Val(Str$(Asc(Mid$(Data$, N, 1)))) - 127
        If Value < Min Then Min = Value
    Next N
    FrontScore = FrontScore - Min 'for now, just store raw bit
                                          difference
    Data$ = Right$(ReceivedData$, ReceivedDataLen - HalfFlagMark)
    TotalData$ = TotalData$ + Right$(ReceivedData$, ReceivedDataLen -
           HalfFlagMark)

'still in the middle of a cycle, just append the new data
Else
    Data$ = Data$ + ReceivedData$
    TotalData$ = TotalData$ + ReceivedData$
End If

'Save Time Data
NewDataLen = Len(TotalData$)
For M = TotalDataLen To NewDataLen
    Temp1$ = Trim$(Str$(Timer - StartTime))
    Temp2$ = ""
    For N = 1 To 7
        If N < Len(Temp1$) Then
            Temp2$ = Temp2$ + (Mid$(Temp1$, N, 1))
        Else
            Temp2$ = Temp2$ + "0"
        End If
    Next N
    TimeData$ = TimeData$ + Temp2$
Next M
```

```
        TotalDataLen = Len(TotalData$)


        'Got to 50 cycles
        If Right$(ReceivedData$, 1) = Chr$(255) Then
            Score = (FrontScore + BackScore) / 50
            Temp1$ = "Terminado!"
            Temp2$ = "Su puntuacion es " + Str$(Score)
            RetVal = MsgBox(Temp2$, 0, Temp1$)

            'Write TotalData to file
            'outfile$ = Format(Date$, "d-mmm")
            'outfile$ = outfile$ + "_AllData_" + Str$(TrialNumber) + ".txt"
            'Open MAINR$ + outfile$ For Binary As #3
            'Put #3, , TotalData$
            'Close #3

            'Write normalized TotalData with Time info to file
            outfile$ = Format(Date$, "d-mmm")
            outfile$ = outfile$ + "_TimedData_" + Str$(TrialNumber) + ".txt"
            Open MAINR$ + outfile$ For Output As #1
            For N = 0 To Len(TotalData$) - 2
                Print #1, (Mid$(TimeData$, N * 7 + 1, 7)) + "  " +
                   Str$(Val(Str$(Asc(Mid$(TotalData$, N + 1, 1)))) - 127)
            Next N
            Close #1

            'Write Scores to file
            outfile$ = Format(Date$, "d-mmm")
            outfile$ = outfile$ + "_Scores_" + Str$(TrialNumber) + ".txt"
            Open MAINR$ + outfile$ For Output As #1
            Print #1, "Average Max Bit Change Backward = "
            Print #1, Str$(BackScore / 50)
            Print #1, "Average Max Bit Change Forward = "
            Print #1, Str$((-1) * FrontScore / 50)
            Print #1, "Total Time in Seconds = "
            Print #1, Str$(Timer - StartTime)
            Close #1

            'Increment Trial Number
            TrialNumber = TrialNumber + 1
            TrialNum.Text = Str$(TrialNumber)

            'Stop
            Timer1.Enabled = False
            CountText.Text = "0"
            Command1.Caption = "Empieza"
            Exit Sub
        End If


        'Graphics
        For N = 1 To ReceivedDataLen
            Value = Val(Str$(Asc(Mid$(ReceivedData$, N, 1)))) - 127
            Shape1.Top = 3240 + (60 * Value)
            Text1.Text = Str$(Value)
        Next N

    End If
End Sub
```

APPENDIX C: TILT SENSOR ANGLE TO VOLTAGE CALIBRATION

The function relating the voltage of the tilt sensor to the angle of RAE was determined by measuring the voltage change at specific angles of RAE as measured by a protractor. This was done for a neutral position of RAE of both 45 degrees and 55 degrees. A quadratic fit was computed in Matlab due to the non-linearity of the tilt sensor. The data is plotted below with the quadratic line fit.