Designing the Haptic Interface for Morse Code

by

Michael Walker

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Mechanical Engineering
Department of Mechanical Engineering
College of Engineering
University of South Florida

Major Professor: Kyle Reed, Ph.D.
Stephanie Carey, Ph.D.
Don Dekker, Ph.D.

Date of Approval:
October 24, 2016

Keywords: Bimanual, Rehabilitation, Pattern, Recognition, Perception

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Dr. Kyle Reed, for providing guidance for my first steps in research work and exercising patience and understanding through my difficulties through the process of creating this thesis. I am thankful for my colleagues in REED lab, particularly Benjamin Rigsby and Tyagi Ramakrishnan, for providing helpful insight in the design of my experimental setup.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Two siblings have a muscular degenerative condition that has rendered them mostly blind, deaf and paraplegic. Currently, the siblings receive communication by close range sign language several feet in front of their vision. Due to the degenerative nature of their condition, it is believed that the siblings will eventually become completely blind and unable to communicate in this fashion. There are no augmented communication devices on the market that allow communication reception for individuals who cannot see, hear or possess hand dexterity (such as braille reading). To help the siblings communicate, the proposed communication device will transmit Morse code information tactically with vibration motors to either the forearm or bicep in the form of an armband wearable. However, no research has been done to determine the best haptic interface for displaying Morse code in a tactile modality. This research investigates multiple haptic interfaces that aim to alleviate common mistakes made in Morse code reception. The results show that a bimanual setup, discriminating dots/dashes by left/right location, yields 56.6% the amount of Morse code errors made under a unimanual setup of Morse code that uses temporal discrimination to distinguish dots and dashes. The bimanual condition resulted in less judgment interference that is either due to the brain having an easier time processing two separate tasks when judgments are shared between the hemispheres or a judgment buffer effect being present for temporal discrimination.

CHAPTER 1: INTRODUCTION

1.1 Motivation

Two siblings, a brother and sister, were born with a neuromuscular degenerative disease that has severely impaired their hearing, vision and ability to move. Currently, the siblings are completely deaf and nearly fully blind. The siblings have lost the neuromuscular capacity to breathe on their own. A machine pumps air through a tracheostomy tube to allow breathing, but cuts off use of their voice boxes. Currently, the siblings lip speaks words to a translator. The translator talks back with sign language several feet in front of the siblings' vision. It is expected that their degenerative condition will worsen the sibling's eyesight to the point where sign language will be an unfeasible method of receiving communication. Persons whom are blind and deaf typically use braille reading as a method of receiving communication. However, the siblings do not possess the physical dexterity to move their fingers in such a way to make use of braille. Currently, there is no augmented communication method or device that is suitable for the siblings. This research explores the possibility of using Morse code for communication through the tactile channel.

Morse code has historically been expressed in the audio and visual modalities, with dots and dashes being distinguishable by stimulus duration (a dash has stimulus duration three times longer than a dot). It is this author's theory that location of stimulus will allow a stronger perceptual disparity for the Morse code elements in a tactile modality. It is proposed that a

device with a haptic interface could transmit Morse code tactilely in the form of an armband worn on the left and right forearms.

1.2 Literature Overview

1.2.1 Errors in Morse Code

Richard Highland categorized frequent errors made in Morse code copying among 807 Air Force radio operator trainees who have passed a 7 wpm code check that were tasked to learn to copy Morse code received by an audio signal at 9 wpm though several code checks in a given day. Operators whom did not achieve 80% accuracy or did not perform a code check at least 1.5 times per day were discarded from the study, leaving 299 subjects. Errors in copying code were classified into four categories, what Highland refers to as the four "Factors". These categories were present in 85.8% of all errors made (Richard W. Highland, 1958). The description of these categories, as described by Highland, as well as their contribution of all errors made, is as follows:

1. *Dash Estimation (8.4%):* "This factor is confined to those signals which contain a number of dashes. In all of these signals, the dashes occur in a series either at the beginning or end of the signal or comprise the entire signal. In no case is there a dot interspersed within the series of dashes, either in the way the signal is sent or perceived. The error is always in estimating the correct number of dashes in the signal. This error may be one of omission or addition; that is, the *S* [. . .] may perceive one too many or one too few dashes, but he never "shortens" a dash to a dot or "lengthens" a dot to a dash. The number of dots in these signals is always perceived correctly."
2. *Dot Estimation (36.2%):* "This factor involves signals consisting mainly of dots. The dots always come in a row either at the beginning or end of the signal or else the signal

consists only of dots. No dash ever separates the dot sequences either in the signal actually sent or in the signal as perceived. This appears to be the dot counterpart of Factor I. In this factor the error is always in estimating the correct number of dots in a series. No errors are made in estimating the number of dashes. The factor includes both overestimates and underestimates of the number of dots in a series; however, it appears most strongly in errors of underestimation with signals containing a long series of dots."

3. *End-element Substitution (30.1%):* "The stimulus characters loaded on this factor are of varied types (i.e., predominantly dots, predominantly dashes and mixed elements), but the type of error made is completely consistent from character to character. In each case, an error of substitution is made and this error always occurs on the last element of the character sent. The substitution may be either a dot for a dash or a dash for a dot. The trainee, in these instances, always perceived the correct number of elements per character."
4. *Internal Error (11.1%):* "This factor extends to fewer variables than was the case with the previous factors, and interpretation, therefore, is not as secure. All the characters sent consist of both dots and dashes. These occur as two series, dots followed by dashes or dashes followed by dots. The characters do not involve changes from dots to dashes and back to dots again or changes from dashes to dots and back to dashes again. The distinguishing features of three out of four of these variables is that an internal substitution error is made. These three variables involve the sending of five-element characters; the substitution error occurs precisely in the middle element. Further, the error occurs at the end of the initial dot or dash series within the signal."

1.2.2 Stimulus Discrimination between Hemispheres

Representing Morse code elements with tactile stimulus brings the extra dimension of tactile space to be explored as a factor that could potentially improve tactile temporal numerosity judgments (TTNJ). Bradshaw concluded that an intermanual condition lead to faster completion of judgment tasks than an intramanual condition. The results imply that the processing load being shared between two hemispheres is more efficient than information processed in just one hemisphere (John L. Bradshaw, 1998). Craig came to the same conclusion in his research, where subjects were presented tactile patterns in rapid succession either inter or intramanually, noting that the intermanual advantage disappeared after a 400 ms delay between stimuli (Craig, 1985).

The hemispheric models of Friedman and Polsen state that the left and right hemispheres have a finite amount of processing resource pools. Interference occurs when multiple tasks are using the same processing resource pool of a hemisphere (Friedman & Polson, 1981). This model is further supported by Kinsbourne and Cook's experiment, where subjects were tasked to balance a wooden dowel on their left and right index fingers while speaking. An illustration of this experiment is shown in Figure 1.1. Results showed a decreased performance of dowel balancing in the right index finger when subjects were asked to repeat a verbalized sentence. Kinsbourne and Cooks postulated that this decline of performance when a verbal task was introduced was due to interference occurring in the left hemisphere. Right sided motor control and verbalization tasks have cortical centers in the left hemisphere. Kinsbourne and Cooks suggested that interference is a function of the distance between cortical spaces, with more interference occurring when this distance is shorter as a result of resource sharing (Marcel Kinsbourne, 1971) .

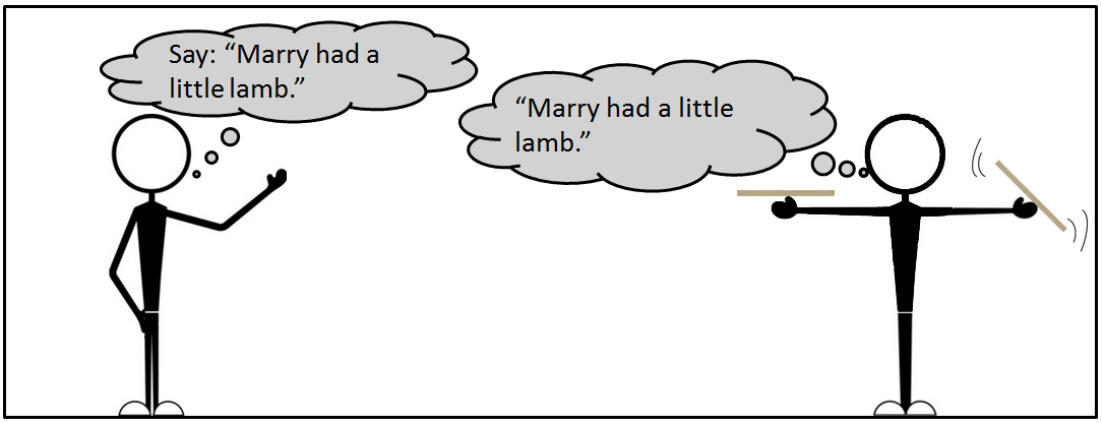

Figure 1.1: Visual representation of Kinsbourne and Cook's experiment. Subjects had a harder time balancing dowels with their right finger when performing a verbal task. This image was adapted.

Charron et al concluded that there exists stimulus degradation when performing an interhemispheric passage of information. The experimental setup (shown in Figure 1.2) measured accuracy of determining the span between two-points of tactile stimulus was same or different than another two-point span, either inter or interamanually. Smaller span differences close to JND showed a more significant advantage towards an interamanual condition 4.74% (Jean-Francois Charron, 1996). Charron shows that there is an interhemispheric disadvantage when performing a judgment task between hemispheres as stimulus information must pass from one hemisphere to the other for comparison and this information degrades over this time lapse.

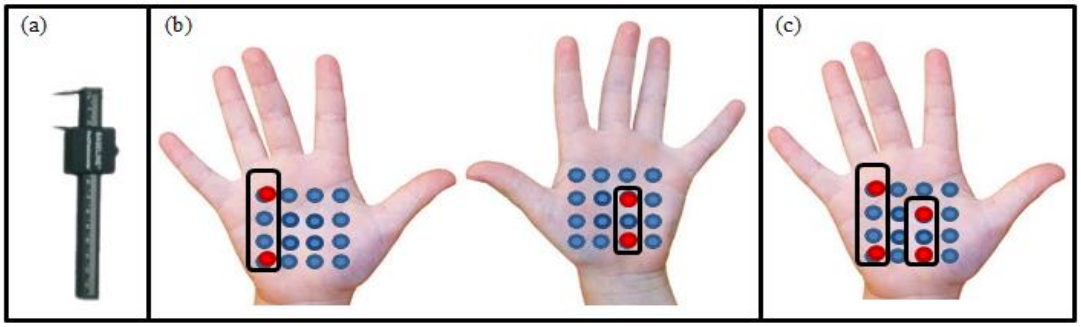

Figure 1.2: Charrn's experimental setup. (a) two-point aesthesiometer used to provide two point tactile stimulus. (b) intermanual condition where subjects were asked to confirm if two tactile patterns between stimulus pints were same or different. (c) Intramanual condition where the two tactile patterns are presented on the same hand. This image was adapted.

Naoki Iida Conducted several experiments to test the effects of unimanual and bimanual presentations of rapidly sequenced vibrations on tactile temporal numerosity judgments (TTNJs). For the unimanual condition, subjects received vibrations on two fingers on the same hand (the index and middle finger). In the bimanual condition, vibrations were sent to the left and right index fingers. Subjects were asked to count how many vibrations occurred for each stimulus location for both the unimanual and bimanual condition. Results showed a significantly higher success rate for this TTNJ task occurred when subjects received stimulus in the bimanual condition rather than the unimanual condition. It was also noted that when task performance went down, the numbers of vibrations were underestimated. This result suggests that stimulus labeling is an easier task to perform when stimulus is received separately between the hemispheres. Results from Naoki Iida's experiments also demonstrated that TTNJ task difficulty was largely a function of stimulus onset asynchrony (SOA) than numerocity of chained stimulus (Naoki Iida, 2016).

1.2.3 The Enumeration Process: Subtilizing vs. Counting

Verlaers concluded from his experiments that haptic subtilizing haptic geometrical patterns can take place. Subtilizing is most accurate for few items (>3) and fast enumeration (<100 items/sec) where counting is better suited for tasks of many items for slower enumeration (>200 items/sec). Subjects used their index finger to scan tactile bumps on a flat surface, similar to braille, in geometric patterns. to test if subjects were capable of performing the enumeration process of counting the dots faster when dots were organized in configured patterns (triangle, squares) versus being presented in a straight line. It was found that configured patterns lead to faster enumeration, which suggests subtilizing took place (K. Verlaers, 2015).

1.2.4 The Neurological Processing of Morse Code

Lara Schlaffke found that Morse code is a two-task process. The first task is a perception process of identifying stimulus length (deciding whether a stimulus is labeled as a dot or a dash). Once this stimulus has been successfully identified, a lexico-semantic analysis is performed to identify words from non-word elements (Lara Schlaffke, 2015).

1.2.5 Morse Code Timing

Relative timing is how Morse code elements are discerned from one another. Figure 1.3 shows the amount of time units to represent all the Morse code elements. Morse code speed is quantified in terms of words per minute (wpm). The word PARIS is considered as the standard word for calculating the value of a time unit from a known wpm speed.

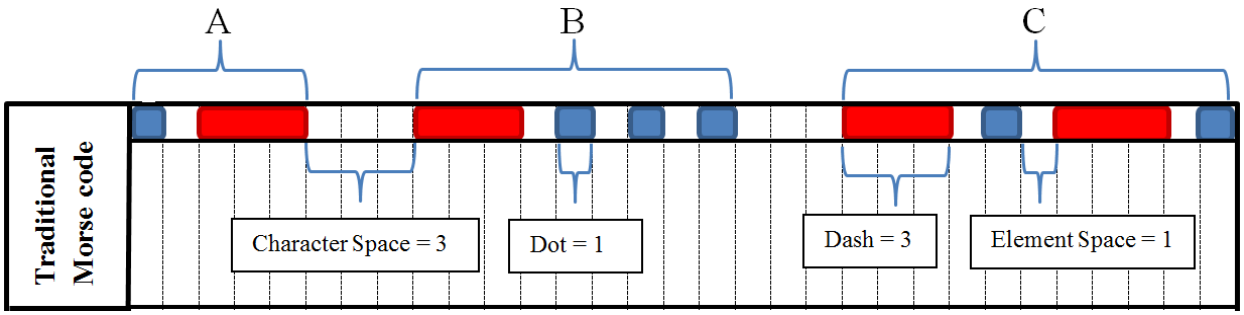

Figure 1.3: Morse code timing scheme. Morse code elements are distinguished in terms of the amount of time units a stimulus is active or inactive. Dots and dashes are represented with active stimulus contributing 1 and 3 time units respectively. Element gaps, letter gaps and word gaps are represented with inactive stimulus at 1, 3 and 7 time units respectively.

CHAPTER 2: EXPERIMENTAL SETUP

2.1 Setup Design that Influences Perception

2.1.1 Bimanual versus Unimanual Haptic Presentation

Results from Kinsbourne and Cook's experiment suggest that hemispheric interference occurs when both a manual task and a verbal task occur simultaneously. Lara Schlaffke identifies Morse code being a two-task process of stimulus identification and lexico-semantic analysis. Sensory feedback from limbs is stored contralateral (stimulus identification) and lexico-semantic analysis occurs in the left hemisphere along with verbal tasks. It is also important to note that Charron found that stimulus degradation occurs between interhemispheric communications. In Kinsbourne and Cook's experiment, the bimanual task does not require interhemispheric communication. Information about how well one might balance a dowel on the right hand is unimportant to the task of balance on the left. In Charron's experiment, stimulus on one side of the body was compared to the other, but this task is not very challenging and does not include an additional stimulus that might interfere with this decision. Figure 2.1 represents where tasks are taking place for a haptic bimanual interface for Morse code. It can be seen in Figure 2.1 that both lexico-semantic analysis and dash stimulus identification occur on the same hemisphere. It is expected that this breakup of tasks would cause significantly less Morse code error than if dot and dash identification occurred on the same hemisphere, as these two tasks are far more similar to one another.

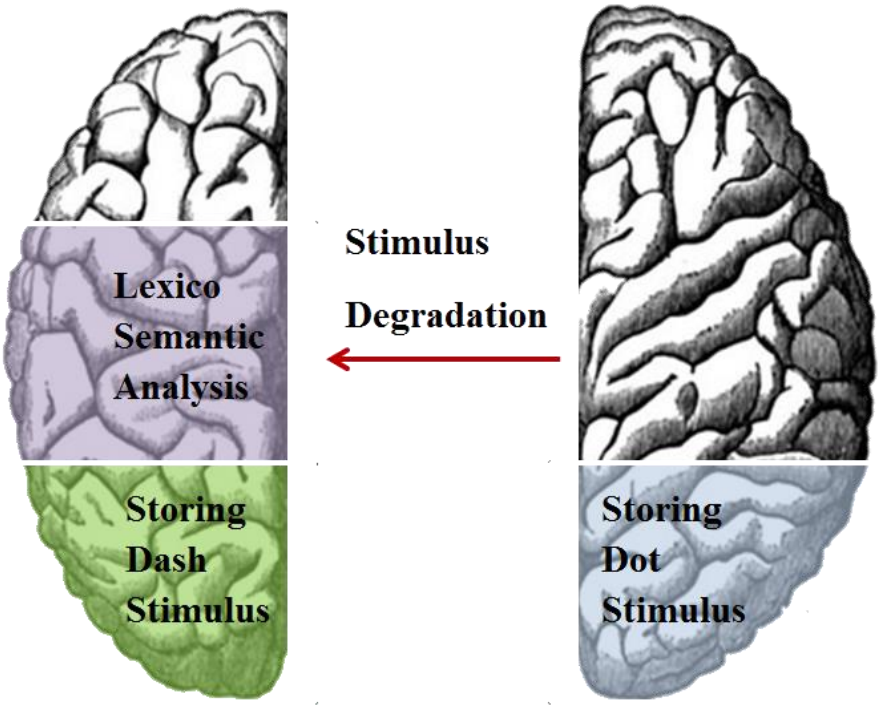

Figure 2.1: Representation of processing tasks in the left and right hemispheres of the brain when using a bimanual setup. This image was adapted.

Stimulus degradation occurs when dot stimulus must have its order of reception compared to dash stimulus. Interference occurs whenever a subject is in the process of figuring out what the perceived stimulus order translates to in terms of Morse code (lexico-semantic analysis) and additional stimulus begins to be received once the second or third character within a character string. With a unimanual interface, both dash and dot stimuli are being constantly compared to determine what stimuli is considered "long" or "short". This creates hemispheric interference whenever a subject is in the processes of making this comparison and an additional Morse code element is presented. Figure 2.2 provides an example of how interference might occur more often for a unimanual setup over a bimanual one.

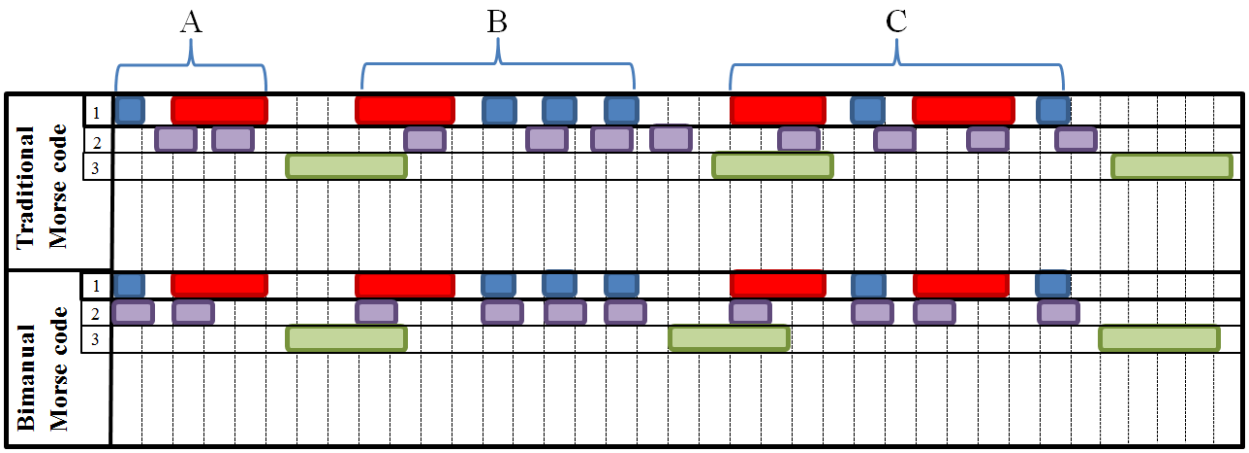

Figure 2.2: Time lapse of interference for unimanual and bimanual setups. The bar labeled "1" shows stimulus of Morse code. The bar labeled "2" shows time dedicated to a judgment task of determining if a stimulus is a dot or dash. In this example, the judgment time is slightly larger than one time unit and judgment time occurs for the unimanual condition after the length of a dot plus half of a time unit to confirm that stimulus has either ended (confirming stimulus is a dot) or continuing (confirmed a dash). In the bimanual case, this judgment task begins immediately, as all information necessary to determine what is a dot is or dash is instantaneous with stimulus presentation. Bar "3" represents the amount of judgment time it might take to classify the character being represented in Morse code. It can be seen that less overlap of stimulus and judgment (interference) will occur in a bimanual setup relative to a unimanual setup.

2.1.2 Farnsworth Spacing

Farnsworth spacing is a method of teaching Morse code where character elements and inter-element have a high wpm, but characters and word spaces are longer than usual. This method of Morse code learning encourages characters to be learned as patterns instead of analytically identifying a character by counting the amount of dots and dashes that make it up. In a study about how novices learn Morse code, Allan showed that novices whom learned Morse with a pattern recognition approach achieved higher wpm in copying Morse than those whom underwent an analytical approach. The pattern recognition group was taught the alphabet at the speed of 20 wpm to encourage memorization of letters as sound patterns rather than an analytical approach of memorizing how many dots/dashes were represented. The pattern recognition group had a significantly higher knowledge of the Morse code alphabet and reached higher wpm speeds much faster than the analytical group (Allan, 1958).

Farnsworth spacing serves two purposes in the experimental design. The first is to increase the perception challenge of stimulus counting and stimulus labeling by having characters represented at a fast pace. The second is to control difficulty by shrinking character spacing, reducing the judgment window for lexico-semantic analysis of a character and allowing more interference to occur.

2.2 Morse Code Characters used for Experiment

Morse code characters were chosen in such a way to provide fairly equal representations of Highland's four categories of most common Morse code errors. Figure 2.3 shows the twelve characters selected to be used in the experiment. The following is the list of the categorized error pairs present in the study:

1. Dash Estimation: (J-W), (W-J), (1-J), (J-1), (1-W), (W-1)
2. Dot Estimation: (H-S), (S-H), (5-H), (H-5), (5-S), (S-5)
3. Internal Error: (V-U), (V-U), (8-7), (7-8)
4. End Element Error: (V-H), (H-V), (5-4), (4-5)

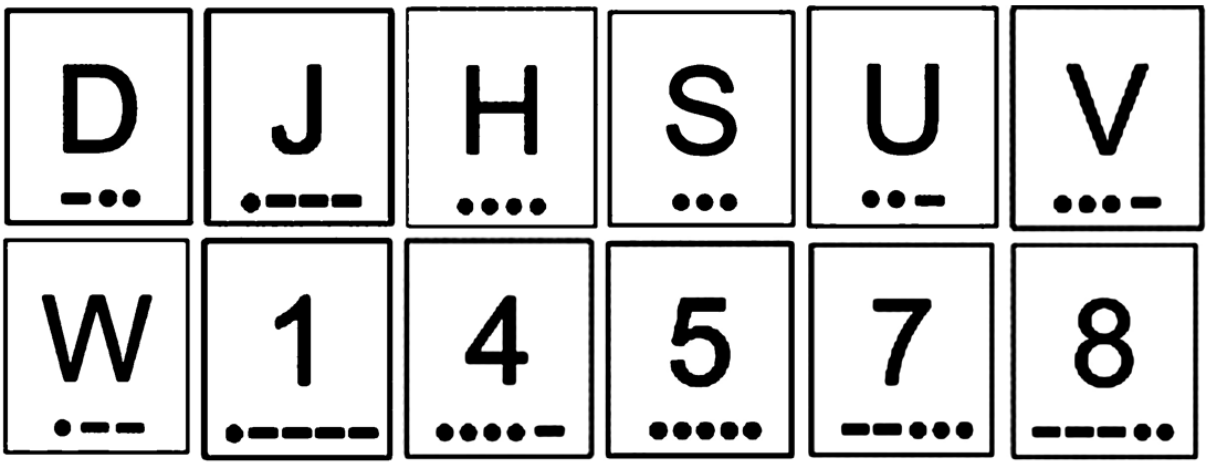

Figure 2.3: The twelve selected Morse code characters.

2.3 Experimental Design

The experiment was comprised of 8 subjects, 6 males and 2 females. All subjects were between the ages of 20-30. 6 subjects reported they were right handed, 1 subject was left handed and 1 subject was ambidextrous. None of the subjects had any prior experience with Morse code. All subjects were healthy with no conditions that hindered their ability to sense stimulus in their forearms. Each participant read and signed a consent form before the experiment that followed a protocol approved by the University of South Florida's Institutional Review Board. Subjects were given a copy of a sheet with the 12 Morse code characters to study before the experiment. Subjects had access to three documents throughout the entirety of the experiment:

1. An image of the four haptic setups and a brief description of how they worked.
2. A sheet with the 12 characters without their respective Morse code under them. This is to let the subject remember all possible character entries in the experiment.
3. A gridded sheet of paper where a subject could write down his answers after receiving Morse code. The MATLAB script is unable to have answers entered into it while sending out instructions to the vibration motors, making this sheet necessary to have so subjects don't forget their answers.

The order in which the haptic setups were tested was randomized for each subject. All haptic setups were balanced so that they all appeared in the first and second order twice. This was done so that if a learning curve did exist within the experiment, where subjects were becoming better at Morse code judgments over time that no setup would have a biased advantage or disadvantage if it occurred earlier in the experiment than another setup. Subjects were acoustically shielded with headphones playing rain drops to ensure that acoustic identification of stimulus played no role in the experiment.

Subjects took a competency test before the initiation of data collection in the experiment to ensure they knew all 12 of the Morse characters with 80% accuracy. Subjects then proceeded to a practice test with a Farnsworth spacing of 3 seconds between characters, where the subject can become accustomed to interpreting three Morse code characters in a row. Subjects wrote their answers on a gridded sheet of paper to copy the Morse code characters as they perceived them. Subjects were not allowed to write down the elements that represented Morse code (ex: _ _ _ - -) because this would allow them to focus on stimulus reception and identification, then have indefinite time to do Morse code translation to English text (lexico-semantic analysis).

2.4 Description of MATLAB Testing Program

Users are prompted to enter several pieces of information in the beginning of the script: their subject number so that data can be written into a unique excel file, haptic setup in which they are using, enter “y” for skipping to testing if they have already completed at least one haptic setup and enter “y” if they wish to take a verbal competency test to skip that section of the program code. After user input, the MATLAB program then creates a random permutation of the twelve characters twice and stores them into an array. As the program loops, an individual character is retrieved from the array and has its corresponding Morse code pulled up, stored as a five item array composed of the three strings “dot”, “dash” or “null”. This character array is sent to a function that powers selected pins in an Arduino Uno that actuate vibration motors with a pattern that reflects the given haptic setup. The program uses a variable called “Test” to progress through several sections called “phases” that have specific functions. For a more detailed description of the MATLAB program, refer to the flowchart, seen in Figure 2.4.

2.4.1 Test Phases and their Role

1. Teaches Morse code individual characters to subjects. Participants receive a character and its Morse code identification visually on the computer monitor and simultaneously receive the Morse code through the haptic interface.
2. Competency test to make sure subjects successfully learned Morse code characters. If subjects do not successfully identify individual Morse code characters 80% of the time, they must retake the test. If it is apparent that a subject knows all the Morse code characters, but is unable to successfully receive a score of 80% or higher on the competency test, a verbal test can be taken in lieu. The verbal test makes sure that perception mistakes don't gate a subject from proceeding with the experiment. Subjects were given prep talks after each unsuccessful attempt at the competency test and were given helpful mnemonics to get them through this portion of the experiment faster, as many subjects found this portion of the experiment aggravating.
3. Practice of three character strings. This is to familiarize the participants with character spacing. The practice portion of the experiment was only done for the first tested setup to due to time considerations concerning the duration of the experiment.
4. Three character string tests were results are recorded. This phase is repeated three times, with character spacing decreasing by one second after phase completion. The character spacing starts at three seconds, then shrinks to two seconds, then one second. If a subject mistypes an answer into the computer, they can circle their answer on the gridded paper and the answer was changed after the experiment to the intended one.

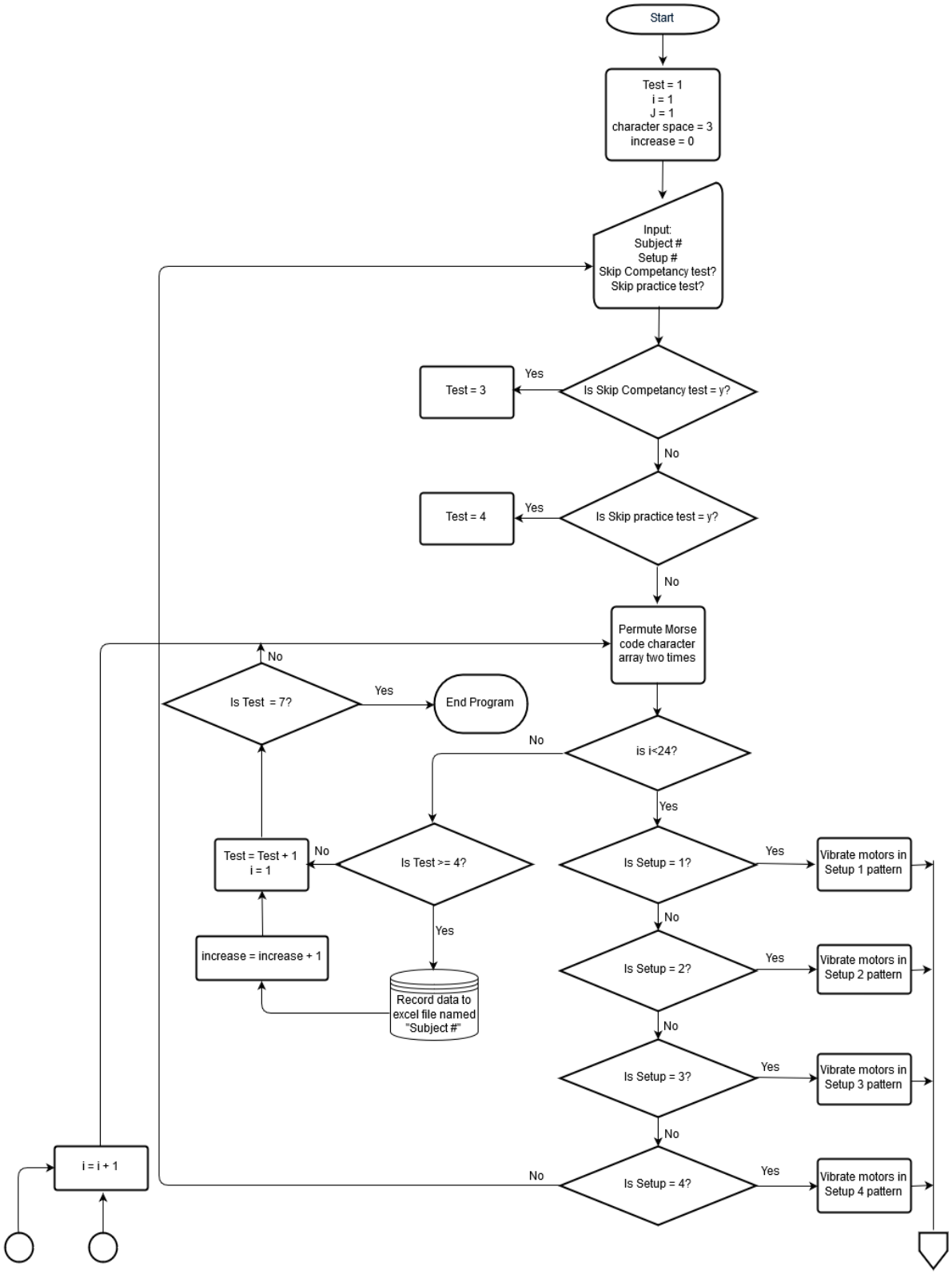

Figure 2.4: MATLAB Flowchart

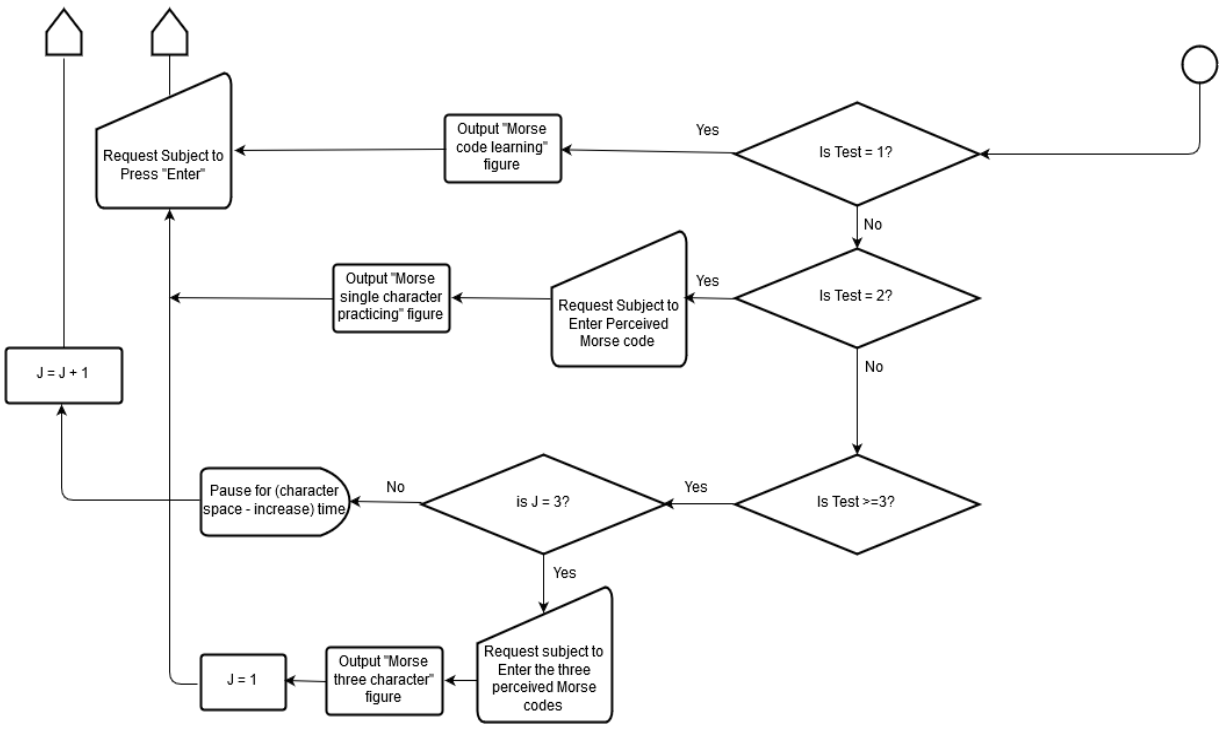

Figure 2.4: Continued.

2.5 Description of Haptic Setups

The four haptic setups can be reviewed in Figure 2.5. A more detailed description of the haptic setups and their intended comparisons are discussed below:

1. *Traditional unimanual:* Dots and dashes are represented on the left forearm. Dashes have duration three times longer than dots.
2. *Bimanual:* Dots are represented on the left forearm and dashes are represented on the right forearm. Dashes have duration three times longer than dots. For comparison, this is the same as Traditional unimanual, but the dots and dashes are displayed on different arms.
3. *Short dashes Bimanual:* Dots are represented on the left forearm and dashes are represented on the right forearm. Dashes have equivalent duration time as a dot (one time element). For comparison, this is the same as Bimanual (i.e., applied on different arms),

but the dots and dashes have the same length of time; only difference between them is the placement.

4. *Bimanual with motor intensity:* Dots are represented on the left forearm and dashes are represented on the right forearm. Dashes have duration three times longer than dots. Successively similar elements within a character increases the amount of motors triggered (up to a maximum of three motors). Motors were presented in an "L" shape. The motors were arranged in an "L" shape, shown in Figure 2.6. This shape was chosen after brief testing with other possible motor arrangements. It was believed that this particular shape allowed for a more discernable change in motor intensity when the motor shifts occurred entirely vertical or horizontal. This shape also takes advantage of the design space of wearing an armband. For comparison, this is the same as Bimanual, but repeated dots/dashes are presented in a different location.

Figure 2.5: Motor arrangement for haptic setup 4. Motors are actuated based on numeric labeling smallest to largest.

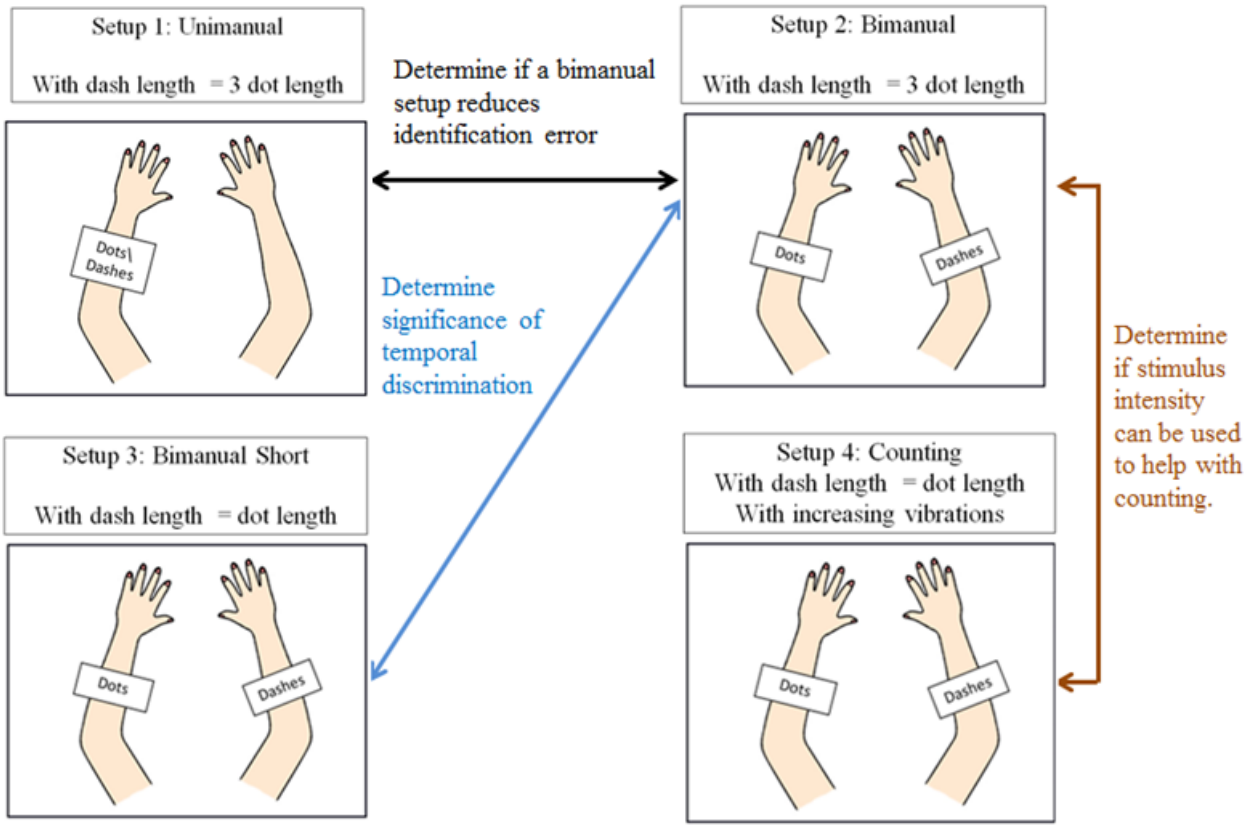

Figure 2.6: The four haptic setups. Arrows indicate descriptions of comparisons to be made in this study. This image was adapted.

2.5.1 Haptic Interfaces for Assisting in Enumeration Tasks

Two haptic setups have been designed to reduce error in the four categories of most common errors in Morse code described by Highland. Highland's error categories of dot and dash estimation error can be described as an error in enumeration. Three vibration motors formed into an L shaped pattern were designed to actuate sequentially when concurrently similar Morse elements are represented. The illusion of motor intensity serves as an additional means of assisting enumeration. Subtilizing for enumeration was experimented with, but did not make it into the experimental design as distinguishing between simultaneous shifts in motor intensities was too challenging to interpret.

2.5.2 Increasing Communication Speed with a Bimanual Setup

It is theorized that a bimanual setup will reduce the amount of stimulus labeling mistakes by using stimulus location as the primary mean of discrimination rather than stimulus duration. Using location as the stimulus identifier makes dash length redundant. If dashes can be the same length as dots (one time unit), Morse code communication can be received at a faster rate. PARIS represents the average English word and is composed of 50 time units. When dots are a single time unit, PARIS is made up of 42 time units, making the average English word be received 16% faster. Figure 2.7 further illustrates how a bimanual setup increases Morse code communication expedience.

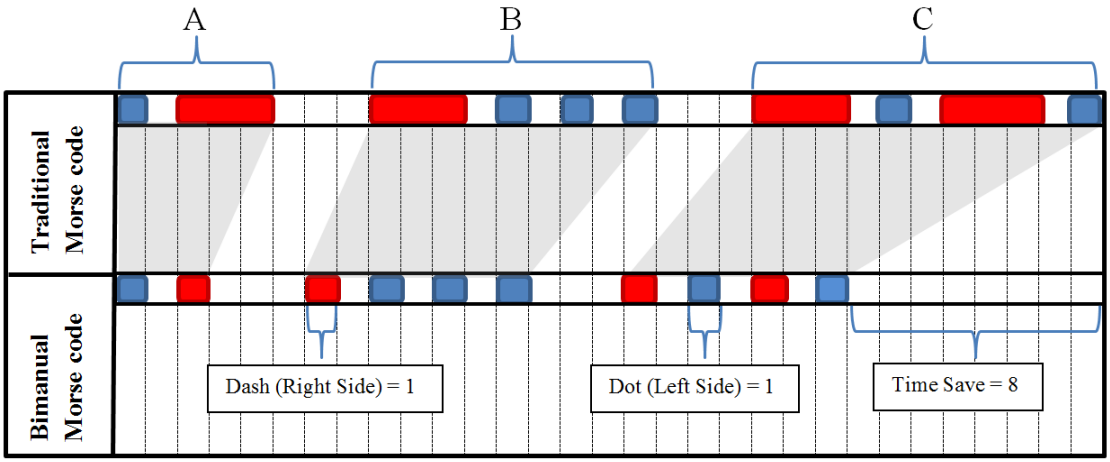

Figure 2.7: How a bimanual setup can reduce communication time. In this example, the string "ABC" is expressed 26% faster.

CHAPTER 3: RESULTS

The results will indicate if any haptic setups are statistically significant from one another by conducting a repeated measure ANOVA and post hoc test. It can also be determined if any of the haptic setups have any significantly different change in errors associated with the Test variable, that alters Farnsworth spacing, making Morse code judgment more challenging. It can also be shown if any statistically significant difference between one the haptic setups for each of the four categorized error types exist and if categorized errors appeared in patterns similar to Highland's study. The experimental design can also be review in its effectiveness of minimizing the learning curve by conducting an ANOVA and post hoc test based on the order in which the haptic setups were tested.

3.1 Setup Performance Overview

A two way ANOVA repeated measures analysis shows there is a statistically significant difference between the Setups ($F(3,21) = 5.062$, $p<0.05$) and the Tests ($F(2,14) = 40.650$, $p<0.001$) for mean error for all 8 subjects. A post hoc test was conducted among the setups, shown in Figure 3.1, reveals a significant difference between Unimanual Setup 1 and Bimanual Setup 2, where Bimanual Setup 2 showed 56.6% the number of errors that Unimanual Setup 1 had. Bimanual Short Setup 3 and Counting Setup 4 had no statistically significant difference between any of the setups. It can be seen in Figure 3.2 that Test 1 has 49% the number of errors than Test 3.

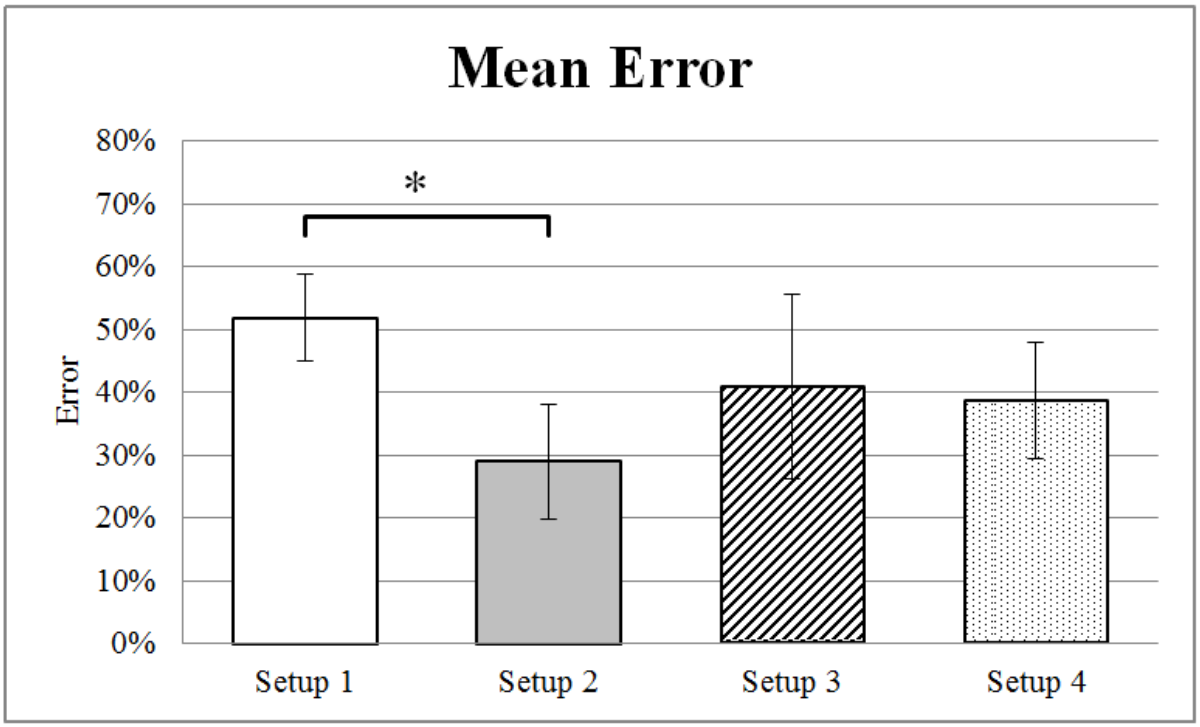

Figure 3.1: Post hoc test between all setups for mean error. Setups 1 and 2 are significantly different from each other. Error bars represent the 95% confidence intervals.

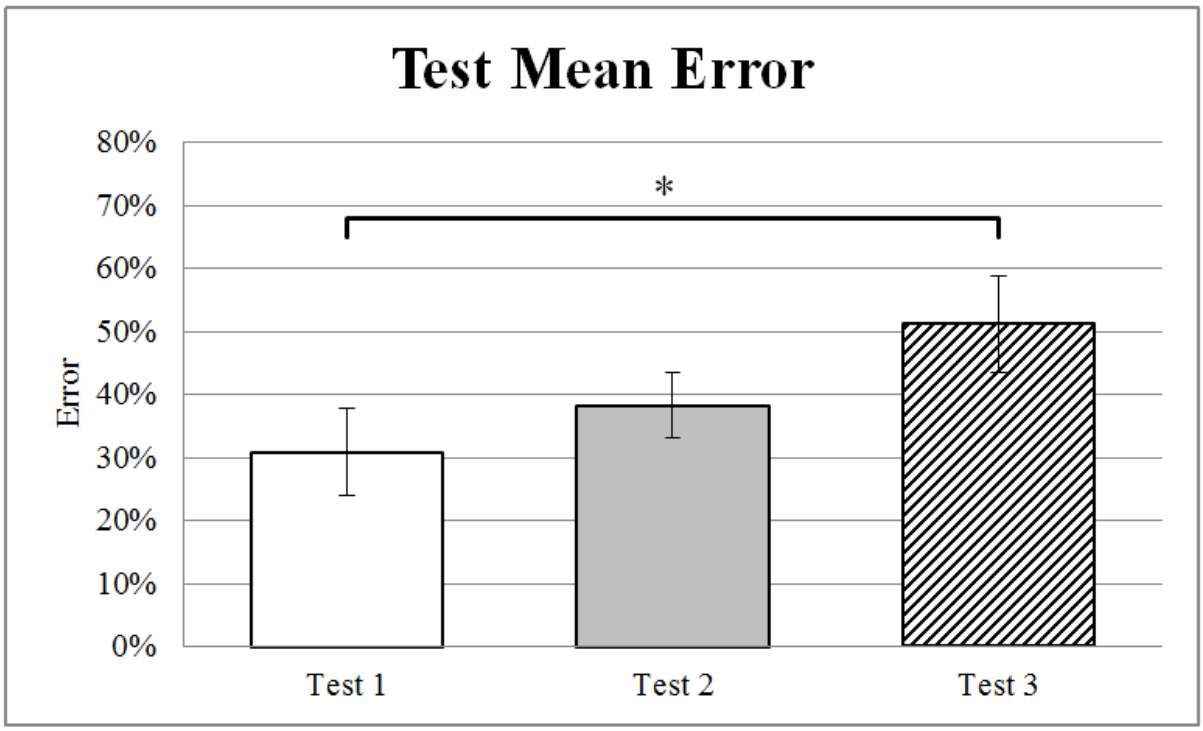

Figure 3.2: Post hoc test between all tests for mean error. Error bars represent the 95% confidence intervals.

3.2 Significance of Order

There is no statistically significant difference between for the order in which the setups were taken. Figure 3.3 shows the post hoc test for all mean error based on order. Although there is no statistical significance, there does appear to be a decrease between the first setup tested and the remaining setups. However, because of the experimental design, the four setups were balanced between which ones were tested first. Thus, order is not considered to be a major effect in these experiments.

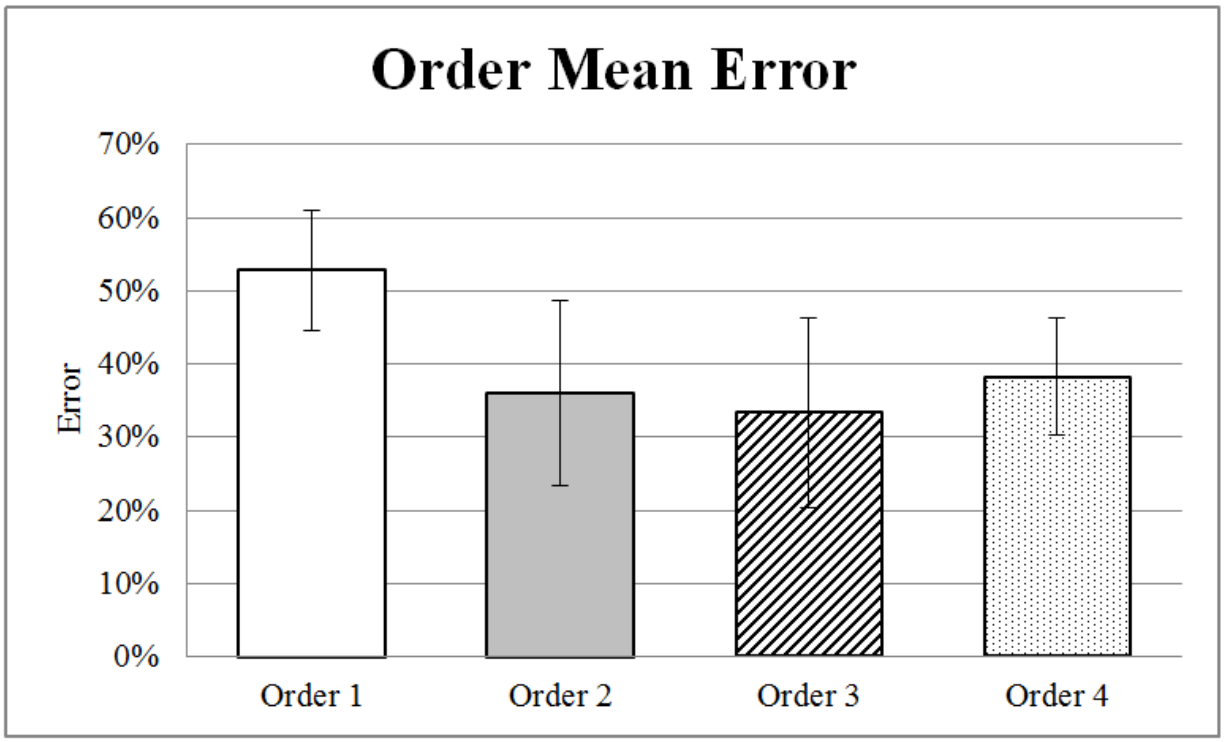

Figure 3.3: Post hoc tests of all mean error based on order. Error bars represent the 95% confidence intervals.

3.3 Categorical Errors

There were no statistically significant findings between any of the haptic setups for Highland's categorized Morse code errors. Tables 1 and 2 can be referred to better understand the relation of categorized error pairs with Highland's study. Table 3.1 lists error occurrences for each categorical error among the four setups along with Highland's results. Table 3.2 lists the

amount of categorical error pairs among all possible error pairs for the current study and Highland's study.

Table 3.1: Percent contribution towards errors made of the four categorized errors within each setup. Highland's errors in his study are also shown.

Error Category	**Setup 1**	**Setup 2**	**Setup 3**	**Setup 4**	**Highland's**
Categorized Error	30.7%	42%	37.7%	36%	85.8%
Dot Estimation	11.4%	19.2%	13.3%	12.9%	36.2%
Dash Estimation	9.8%	12.0%	16.0%	12.9%	8.4%
Internal Error	7.2%	10.2%	8.0%	8.9%	30.1%
End Element Error	0.60%	0.40%	1.30%	11.10%	11.1%
Other Error	69.3%	58%	62.3%	64%	14.2%

Table 3.2: Frequency of error pairs to appear within all possible error pair combinations.

Error Category	**Current Study**	**Highland's Study**
Dot Estimation	6 (4.5%)	11 (0.9%)
Dash Estimation	6 (4.5%)	18 (1.4%)
Internal Error	4 (3.0%)	26 (2.1%)
End-Element Error	4 (3.0%)	14 (1.11%)
Total Possible Error Pairs	132 (100%)	1,260 (100%)

Categorized character error pairs represented 15.15% (20 error pairs out of 132 total error pairs available) and made up an average of 39.13% of all errors between the 4 haptic setups in the experiment. In Highlands study, categorized errors represented 5.47% (69 error pairs out of 1260 total error pairs available) and made up 85.8% of all errors made. In this study, categorized error pairs were 2.58 times (39.13%/15.15%) more likely to result in an error than non-categorized pairs while in Highlands study, a categorized pair was 15.69 times (85.8%/5.47%) more likely to result in an error relative to non-categorized pairs. Subjects in Highland's study were 6.07 times (15.69/2.58) more likely to make a categorical error than the subjects in this study instead of a non-categorized error.

CHAPTER 4: DISCUSSION

Using stimulus location to discern dots and dashes on the left and right forearms for Bimanual Setup 2 resulted in a statistically significant difference to haptic Unimanual Setup 1 that used stimulus duration as a stimulus identifier instead. Setup 2 showed 56.6% the number of errors that Unimanual Setup 1 had. In other words, this research successfully proved that representing Morse code with a bimanual setup reduces errors made by over half compared to the traditional method of representing Morse code. For the clients who will be receiving Morse code as an alternative communication method, a bimanual haptic setup will allow fewer errors in receiving communication. This finding is also significant for complicated haptic interfaces with a multiple, quickly successive feeds of tactile stimulus. Such interfaces that meet this criterion are vibrotactile body interfaces for alerting blind individuals of obstacles. The need to design complex full body haptic interfaces might arise as an extra element of immersion in conjunction with a VR platform, or as a means of avoiding obstacles for persons who are visually impaired.

Tests 3 had a statistically significant higher amount of errors than Test 1. By reducing judgment time by 2 seconds between Morse code characters, Test 1 had half (49%) the amount of errors of Test 3. This finding is consistent with Naoki Iida's finding that for quick successive tactile stimulus judgments; the temporal distance between stimuli serves as a prominent factor in judgment difficulty. A smaller temporal distance between Morse code characters (Farnsworth spacing) would allow for faster communication reception by the clients. The transition from English to Morse code can be viewed as a transition between using a 26 letter alphabet to a two letter alphabet. It is likely that the transition to Morse code communication will be aggravating to

the clients. The clients will likely want to challenge themselves and reduce the Farnsworth spacing, creating higher error in perception. This makes a bimanual haptic interface more appealing to the clients, as it reduces error and allows the clients to reduce Farnsworth spacing more so than a unimanual setup would.

The significance between Unimanual Setup 1 and Bimanual Setup 2 was found to not have a statistically significant relation to Highland's categorized errors. It is this authors interpretation that the statistically significance between haptic setups 1 and 2 is due to a judgment buffer effect that is present when stimulus duration is the identifier for dots and dashes, as described in Figure 2.2. The judgment buffer exists since it is impossible to identify a dot or dash until the full duration of a dot has been represented. If the stimulus ends, then it can be concluded the stimulus was a dot. If the duration of stimulus continues, it can be classified as a dash. This delay in stimulus identification could lead to more frequent overlapping of three tasks: stimulus reception attention, stimulus identification and Morse code translation to English characters (lexico-semantic analysis). The existence of a judgment buffer clouds any conclusion of confirming that the statistically significant difference between Bimanual Setup 2 and Unimanual Setup 1 validates Kinsbourne and Cook's hemispherical interference theory. This theory might be in play in this experiment, but the judgment buffer effect makes interference scenarios occur in higher quantities for setup 1. To confirm that the significance between setup 1 and 2 could be due to hemispherical interference, a follow up experiment must be performed when the judgment buffer does not exist. Such a study is described in the future works section of this thesis.

4.1 Significance of Results

Subjects in Highland's study were approximately 6 times more likely to make a categorical error than a non-categorical error than the subjects in this study. Highland had to discard a considerable amount of subject's data from his study, as he required discarded code checks that fell below 80% accuracy, since he deemed these code checks to be impairment of a subject getting lost in the code, making random errors. The subjects in this study perceived Morse code characters 60% of the time among all haptic setups. Subjects had a 71% accuracy using Bimanual Setup 2, the best performing interface. If Highland believes a below 80% accuracy is determinate that a subject has gotten lost in the code and is making random guesses, then by that benchmark, the average subject of this study was making a majority of random errors due to not making judgments about Morse code pattern fast enough. This distinction is important to note, as it means the root cause of the statistically significant difference between Bimanual Setup 2 and Unimanual Setup 1 is due to a higher probability of judgment overlap between the three processed tasks of stimulus reception, stimulus identification and lexico-semantic analysis. In other words, interference of tasks seems to occur more often in setup 1 than setup 2.

The larger percentage of random error present in this study in comparison to Highlands is likely largely due to this study using novices with no prior Morse code experience. A novice is more likely to make a random error as they will find themselves getting lost in the code more often than their more experienced counterparts who instead make more nuanced mistakes described with Highland's categorized errors. Having a small pool of Morse code characters could have also aided in a decrease in characterized errors, as the categorized errors are

represented 9.68% more than in Highland's study, where categorized errors occur far less, allowing a subject to be caught off guard to a higher degree.

Since the significant reduction of errors of the bimanual setup relative to the unimanual setup does not seem to follow any particular pattern, the root cause of such a result might lie within the judgment buffer effect shown in Figure 2.2, where judgment of stimulus must be delayed after its presence when stimulus duration dictates what is considered a dot or dash. This judgment buffer would not occur for a bimanual setup as stimulus location discrimination instantaneously provides all necessary information for stimulus labeling. With stimulus detection and judgment tasks being performed simultaneously more often for the unimanual setup, the circumstance of interference is present more often.

4.2 Difficulty Scaling of Tests

The post hoc of all error data showed a statistically significant difference in error between test 1 and test 3. Reducing the pause between Morse code characters (Farnsworth spacing) by two seconds increased error occurrence by 20.3%. However, test 2 was statistically insignificant compared to setups 1 and 3. Therefore, the reduction interval to Farnsworth spacing (set at 1 second for this study) should be increased. To better understand the how much Farnsworth spacing can be present in the study without compromising perception difficulty, we can extrapolate errors made in the three test difficulties (assuming spacing and difficulty have a linear relationship) and determine when character spacing will result in only 20% error (the error percent that was allowable for the competency test, where subjects had indefinite time to judge a single Morse character). The linear slope between Tests was calculated in Equation 1 to be 10.15%. This means for every second the Farnsworth spacing increases, error decreases by approximately 10%. Since subjects struggled to achieve 80% success (20% error) in the

competency test and had indefinite time to make a judgment, it can be concluded that having a Farnsworth spacing that allows 20% error is equivalent to having indefinite time (character spacing becomes irrelevant in Morse code judgment). Therefore, since Test 1 shows to have 30% error with 3 seconds of Farnsworth spacing, the Farnsworth spacing should not exceed 4 seconds, as this is when error is extrapolated to be at 20%.

$$\text{EQ 1: } m = \frac{(Test\ 2 - Test\ 1) + (Test\ 3 - Test\ 2)}{2} = \frac{(38.3\% - 30.9\%) + (51.2\% - 38.3\%)}{2} = 10.15\%$$

4.3 Minimizing the Effect of the Learning Curve

Learning between the four experimental setups was minimized by initiating a competency before experimental data was collected, along with balancing the four setups so that each setup appeared in the first and second order twice though the experiment. The order in which the setups are tested has minimal effect on the results of the study. It was important to make sure the study measures perception between the haptic setups and not learning the 12 Morse characters as a subject progresses through the experiment. The competency test serves to ensure all subjects undergo the experiment with equal knowledge of the Morse code characters being tested so they have less reason to learn when data is collected. The post hoc test shows that there was no significant difference between the orders in which setups were taken. There is a fairly large, but not statistically significant, difference in the first haptic setup taken, suggesting that a subject is getting better at Morse code perception for the first tested haptic setup (presence of learning). The 80% passing rate of the competency test typically required a subject to undergo multiple attempts to pass, with two instances of subjects who were unable to ever pass the test and had to take a verbal test instead. The passing rate could be made slightly lower so a verbal test is not required, as many subjects seemed to have made frequent categorized errors instead of random errors, suggesting they knew the character meaning but struggled with perception.

4.4 Other Applications

4.4.1 Virtual Reality

Virtual reality displays offer the ability to train for a situation via simulation. This can serve as a risk free, convenient method to prepare for tasks like surgery. In such a simulation, haptic feedback is important to convey the amount of pressure and its direction from a tooltip being applied to a patient's body. It may be more intuitive to display both direction and force feedback onto a one handed haptic interface, but such an interface might lead to interference between discerning the two stimuli. In this case, a bimanual setup might yield more clarity to separate the stimulus of force feedback and direction.

4.4.2 Obstacle Avoidance for Persons who are Visually Impaired

Alerts for object collision are provided to a person who is visually impaired via haptic interface. Information of location and proximity of obstacles are important to keep track of. It might be advantageous to use a bimanual haptic interface to separate these two streams of stimuli. An example would be to have one interface that vibrates a cane handle in the direction of the obstacle and another interface on the other hand that intensifies vibration as proximity is reduced.

4.5 Future Work

4.5.1 Determine if Hemispheric Interference Occurs For a Morse Code Task

The presence of a judgment buffer for the unimanual setup makes it very difficult to validate Kinsbourne and Cook's theory of separate hemispheres having a finite resource pool with these experiments results, as task interference occurs in higher quantities for the unimanual setup in relation to the bimanual setup. A better measure of interference would be to compare a unimanual setup attached to the left arm with that of a unimanual setup on the right arm. The

right arm setup would have all information processed in the right hemisphere, while the left arm setup processes stimulus in the right hemisphere and performs lexico-semantic analysis in the left hemisphere.

4.5.2 Determine if a Judgment Buffer Occurs For a Morse Code Task

A follow up experiment is required confirm that the statistically significant difference between Bimanual Setup 1 and Unimanual Setup 2 is due to the judgment buffer effect and not hemispheric interference. The experimental design for such an experiment will compare two haptic setups: one setup being the same as setup 1 for this study, where stimulus is identified with stimulus duration. The other setup will also be unimanual, but use stimulus location to distinguish dots and dashes, where dots will be represented on the forearm and dashes will be represented on the bicep. If it is found that setup 1 is statistically significantly more likely to make an error, it can be concluded that a judgment buffer does indeed exist.

4.5.3 Design a More Intuitive Haptic Setup for Counting

It was surprising that using three motors to increase motor intensity had resulted in a higher mean error compared to setup two, its most similar counterpart. It was expected that configuring three vibration motors in a geometric shape would assist in enumeration tasks. This setup is similar to how subjects in Verlaers's study were able to improve in enumeration judgment tasks when performing braille finger scanning when the braille was organized in three element geometric shapes.

However, comments of subjects after testing noted that the haptic display of setup 4 was confusing. This could either be due to shifts in motor intensity being too much information to process, or it could mean that the "L" shaped motor arrangement was unintuitive in conveying the presence of successively similar Morse elements. It is proposed that a radial growth of

vibration would be more intuitive than a growth in an “L” shape. Figure 4.1 shows the proposed design of a radial haptic interface.

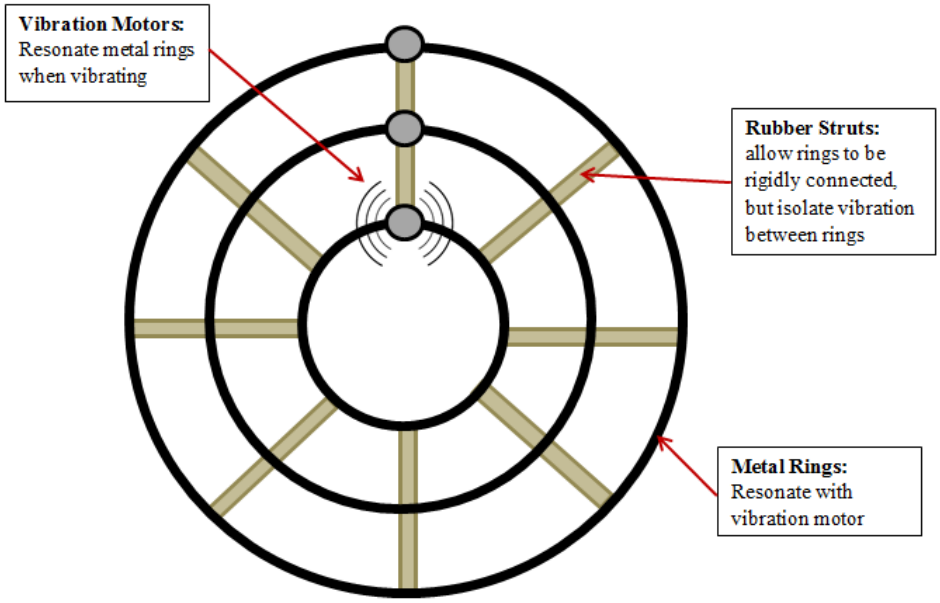

Figure 4.1: Proposed motor arrangement design to facilitate counting with motor intensity.

CHAPTER 5: CONCLUSION

This study showed that using stimulus that is identified with bimanually opposite locations results in statistically significantly lower errors in Morse code perception than using stimulus duration to identify stimuli in a unimanual condition. The error results from the subjects did not follow any of the common Morse code error categories from Highland, meaning that error mistakes followed no specific error pair patterns and are therefore random. Random mistakes are likely due to subjects getting lost in Morse code, which occurs when lexico-semantic analysis is not successfully judged fast enough and an overlapping of tasks takes place when new stimulus is introduced (interference).

It is suspected this interference can be either from hemispheric interference theory, where tasks are capable of being processed easier when overlapping due to information being synthesized in separate hemispheres, or due to a judgment buffer effect, where the inherent delay in stimulus identification when using stimulus duration as a stimuli identifier results in more interference. To be certain if the statistically significant difference between setups 1 and 2 is due to either hemispheric interference or a judgment buffer, their needs to be a follow up study that test haptic setups that separate these two phenomenon.

The conclusion of a bimanual setup resulting in statistically significantly fewer errors in Morse code directly benefits the individuals in which this research is dedicated towards. A bimanual haptic interface that results in 56.6% of the amount of errors of the traditional dot/dash temporal discrimination interface will allow for a far less frustrating transition into adapting to

Morse code as a new means of communication in the short term and in the long term, allow for faster communication without compromise of lower Morse code perception accuracy.

REFERENCES

Allan, M. D. (1958). A Pattern Recognition method of Learning Morse code. *British Journal of Psychology*, 59-64.

Craig, J. C. (1985). Attending to two fingers: Two hands are better than one. *Perception & Psychophysics*, 496-511.

Friedman, A., & Polson, M. C. (1981). Hemispheres as independent resource system: Limited-capacity processing and cerebral specialization. *Journal of Experimental Psychology*, 1031-1058.

Jean-Francois Charron, I. C. (1996). Intermanual Transfer of Somaesthetic Information: A Two-Point Discrimination Experiment. *Neuropsycholoyia*, 873-877.

John L. Bradshaw, M. E. (1998). An Intermanual Advantage for Tactual Matching. *Monash University*, 763-770.

K. Verlaers, J. W. (2015). The Effect of Perceptual Grouping On Haptic Numerosity Perception. *Atten Percept Psychophys*, 353–367.

Lara Schlaffke, N. N.-W. (2015). From Perceptual to Lexico-Semantic Analysis—Cortical Plasticity Enabling New Levels of Processing. *Human Brain Mapping*, 4512–4528.

Marcel Kinsbourne, J. C. (1971). Generalized and Lateralized Effects Of Concurrent Verbalization On A Unimanual Skill. *Quarterly Journal of Experimental Psychology*, 341-345.

Naoki Iida, S. K. (2016). Comparison of Tactile Temporal Numerosity Unimanual and Bimanual Presentations. *Perception*, 99–113.

Richard W. Highland, E. A. (1958). An Empirical Classification of Error Patterns in Receiving Morse Code. *Journal of Applied Psychology*, 112-119.

APPENDIX A: ERROR PAIR DATA

Table A.1: Error pair data per setup for all subjects.

Setup 1		Answer											
All Subjects		D	H	J	S	U	V	W	1	4	5	7	8
User	D	26	4	1	4	6	3	4	1	1	4	2	2
	H	1	24	3	2	0	1	1	1	4	14	0	1
	J	1	1	20	1	1	6	5	12	1	0	1	2
	S	4	9	3	31	7	2	2	0	0	3	0	1
	U	3	1	0	3	24	11	8	2	4	1	1	0
	V	2	0	4	1	2	15	2	3	14	0	1	2
	W	5	2	3	0	6	3	21	0	2	0	0	2
	1	3	3	6	0	0	2	3	21	2	1	2	5
	4	0	0	2	1	0	2	1	1	14	2	1	2
	5	1	2	0	5	0	0	0	1	3	23	1	1
	7	2	2	4	0	2	3	1	5	1	0	30	2
	8	0	0	2	0	0	0	0	1	2	0	9	28

Dot Estimation	35
Dash Estimation	29
Internal error	24
End Element error	6
Other error	205
Correct	277
incorrect	299

Setup 2		Answer											
All Subjects		D	H	J	S	U	V	W	1	4	5	7	8
User	D	37	1	1	0	3	2	4	1	2	1	2	1
	H	1	34	0	4	0	0	0	1	1	13	1	1
	J	2	0	34	0	2	3	5	8	0	0	0	1
	S	0	11	0	43	2	3	1	0	1	3	1	1
	U	3	0	2	0	35	7	3	0	1	0	0	1
	V	2	0	2	1	1	28	1	0	13	0	1	0
	W	2	0	3	0	4	1	33	0	3	0	0	0
	1	0	0	4	0	1	3	0	35	0	0	2	1
	4	0	0	1	0	0	1	0	0	27	1	0	0
	5	0	1	0	0	0	0	0	0	0	29	0	0
	7	0	0	1	0	0	0	1	3	0	0	34	2
	8	1	1	0	0	0	0	0	0	0	1	7	40

Dot Estimation	32
Dash Estimation	20
Internal error	17
End Element error	1
Other error	97
Correct	409
incorrect	167

Setup 3		Answer											
All Subjects		D	H	J	S	U	V	W	1	4	5	7	8
User	D	31	2	4	0	2	2	4	1	4	2	5	2
	H	1	37	0	6	0	0	0	0	0	10	1	1
	J	2	0	20	1	2	4	2	18	1	0	2	4
	S	0	4	0	37	0	2	2	0	0	5	0	1
	U	8	0	2	0	29	4	5	1	1	0	1	4
	V	2	0	3	1	4	26	4	0	16	1	0	1
	W	2	0	13	0	7	0	29	0	1	0	0	0
	1	1	1	4	0	1	2	0	19	0	0	2	1
	4	1	0	1	1	1	6	2	5	24	2	2	0
	5	0	4	1	2	2	1	0	1	0	28	1	0
	7	0	0	0	0	0	1	0	3	0	0	34	7
	8	0	0	0	0	0	0	0	0	1	0	0	27

Dot Estimation	31
Dash Estimation	37
Internal error	15
End Element error	2
Other error	150
Correct	341
incorrect	235

Setup 4		Answer											
All Subjects		D	H	J	S	U	V	W	1	4	5	7	8
User	D	29	0	0	0	8	4	1	1	1	0	4	3
	H	0	36	2	9	0	1	2	0	0	5	0	2
	J	3	0	28	0	0	1	2	15	1	0	2	2
	S	1	4	0	32	0	1	0	0	1	1	0	0
	U	6	0	2	1	30	5	11	2	2	0	2	1
	V	3	1	4	0	3	21	5	1	14	0	3	1
	W	4	0	6	0	5	1	26	0	1	0	2	1
	1	2	1	6	0	0	0	0	28	0	2	0	0
	4	0	1	0	1	0	8	0	1	27	1	2	0
	5	0	5	0	4	1	1	0	0	0	39	1	0
	7	0	0	0	1	1	2	1	0	1	0	31	12
	8	0	0	0	0	0	3	0	0	0	0	1	26

Dot Estimation	28
Dash Estimation	29
Internal error	21
End Element error	3
Other error	142
Correct	353
incorrect	223

APPENDIX B: DOCUMENTS PRESENT IN EXPERIMENT

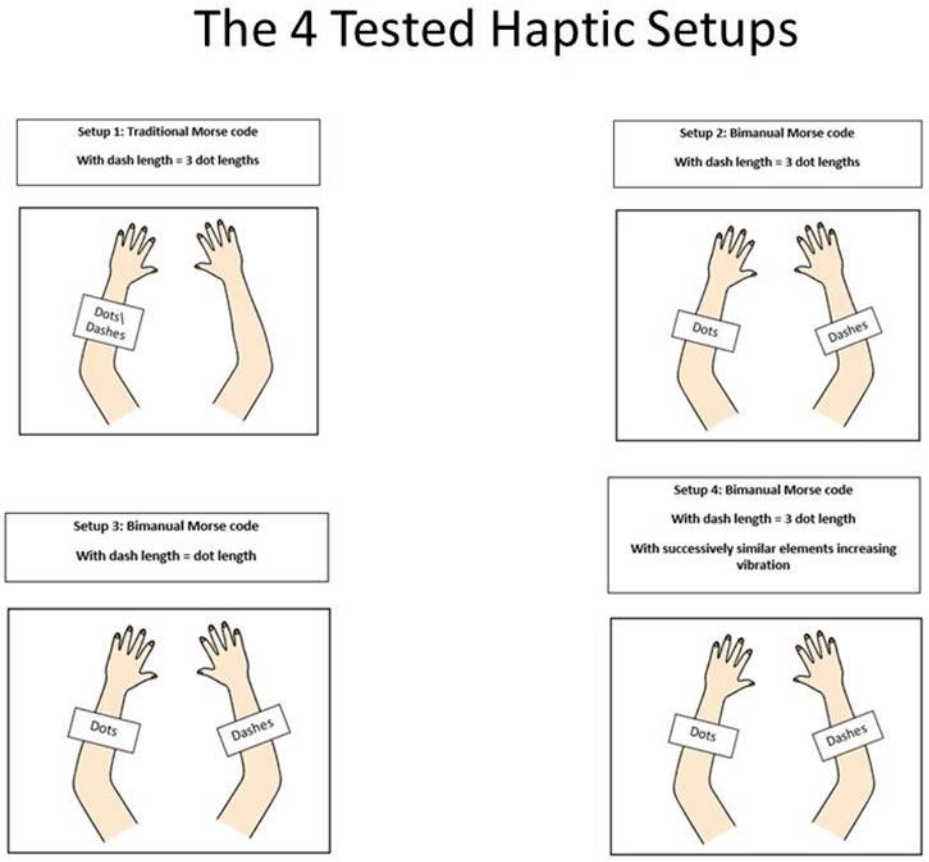

Figure B.1: Overview of the four haptic setups. This sheet of paper was available at all times throughout the experiment.

Subject #	
Vibration setup #	
Gender	M / F
Lef/right handed	Left / Right
Morse Code experience	Yes / No

Practice			
Problem	First	Second	Third
1			
2			
3			
4			
5			
6			
7			
8			

Test 1			
Problem	First	Second	Third
1			
2			
3			
4			
5			
6			
7			
8			

Test 2			
Problem	First	Second	Third
1			
2			
3			
4			
5			
6			
7			
8			

Test 3			
Problem	First	Second	Third
1			
2			
3			
4			
5			
6			
7			
8			

Figure B.2: Sheet of paper used to record answers for three character string tests. This sheet of paper was available at all times throughout the experiment.

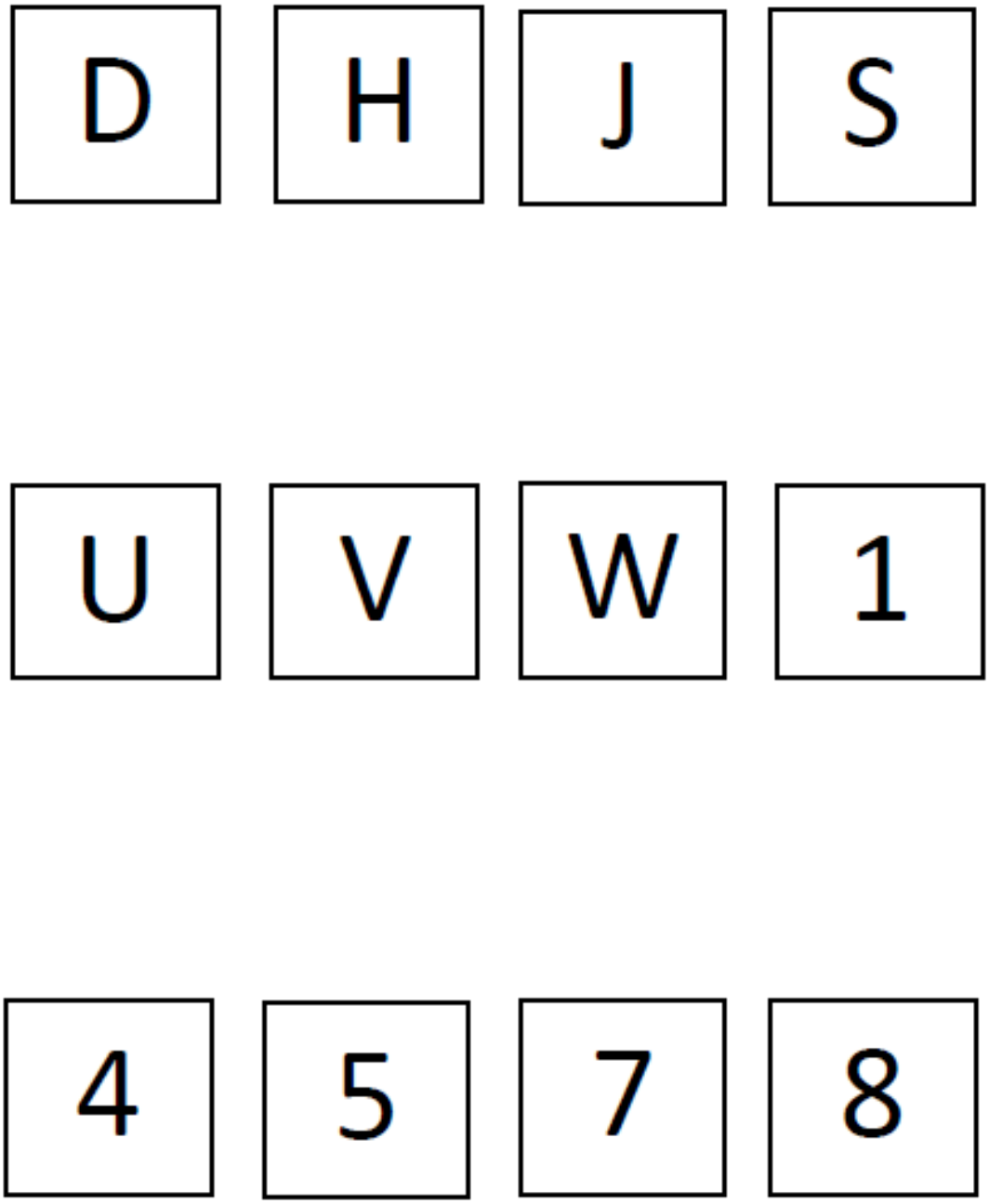

Figure B.3: Sheet of paper showing the 12 possible characters. This sheet of paper was available at all times throughout the experiment.

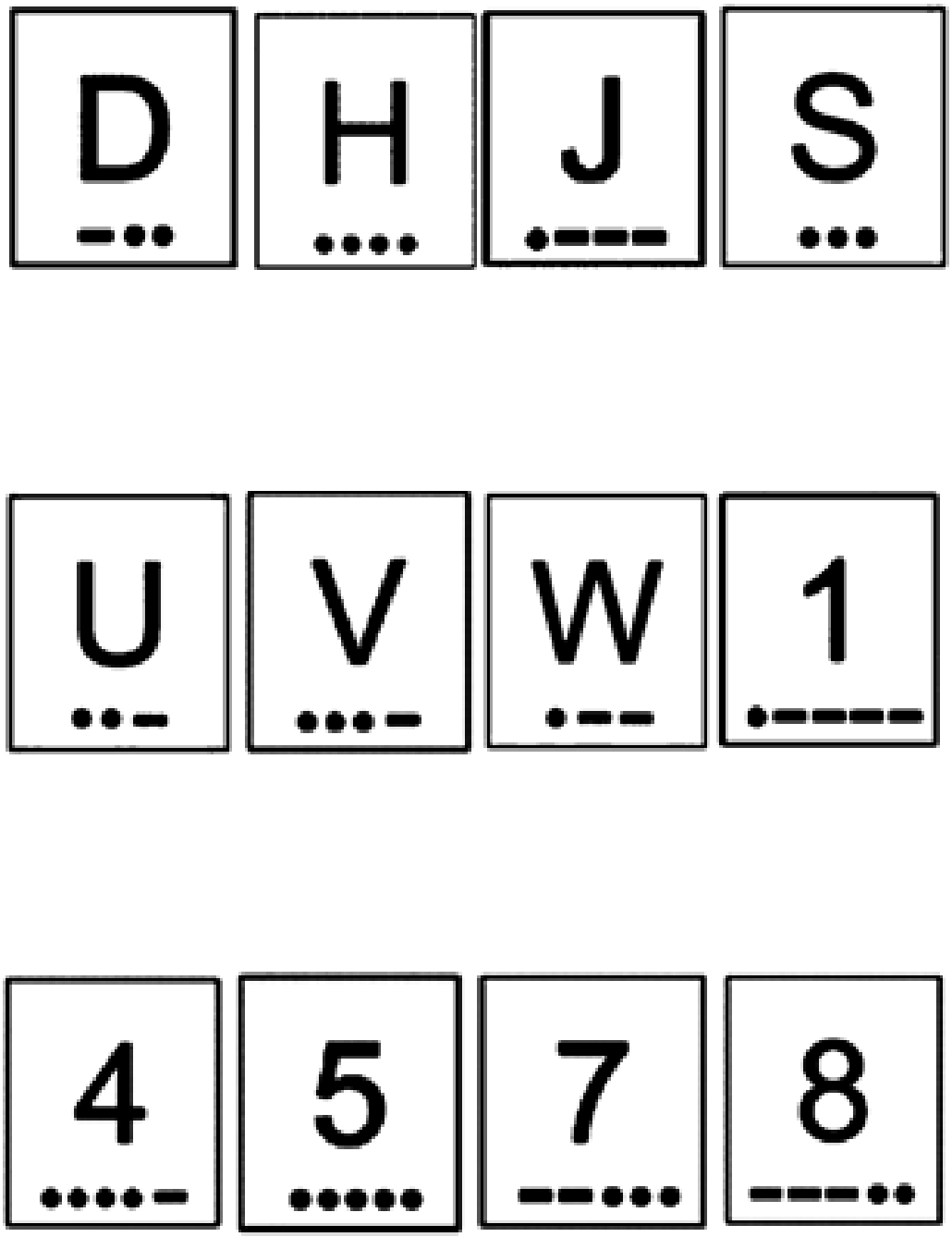

Figure B.4: Sheet of paper with the 12 Morse code characters for studying purposes. This sheet of paper was provided before the experiment. This sheet of paper was not allowed to be viewed once testing began.

APPENDIX C: MATLAB CODE

C.1 Main Script

C.1.1 Contents

- defining variables
- allocate sheet for data storage on excel
- workaround code so that MATLAB writes into excel much faster
- instructions for the several portion of the experiment
- Element time durations calculated for Setup 1 (Traditional)
- recalculate element and letter spacing when dash duration equals dot duration for setup 5 (OLD)
- recalculate element and letter spacing for bimanual presentation (setups 6 + 7) (OLD)
- Morse code identifiers
- Identifying letters from answer key permutation
- Identifying letters from user input
- Sending Morse code to user
- creating display interface for test 1
- creating display interface for test 2
- plotting Test 2
- creating display interface for test 3
- creating display interface for test 4,5,6,7
- plotting Test 4,5,6,7
- write data to excel sheet after test completion

```
clc
clear all
```

C.1.2 Defining Variables

```
n = 12;              % the amount of letters
Trail_amount = 7;    % how many trails are there
Cycle_amount = 2;    % how many cycles for each trail
increase = 0;        % variable that increases wpm based on what test level is active
problem =0;          % variable that counts what problem # your on
correct = 0;         % variable that counts correct letters identified per test
Total_letters = n*Cycle_amount;    % how many letters there are in a test
c1 = 0;              % variable that changes display for test 2 if correct
cell_number = 1;     % variable that increaeses cell position in excel
cell_id = 0;         % variable that turns cell_number into a string
A_Uno = arduino('com4','uno'); % identifies Arduino UNO microcontroller
Test = 1;            % initialize testing phase count
Cycle = 1;
skip = 0;
pass = 1;
```

C.1.3 Allocate Excel Sheet for Data Storage

```
flag = 'n';  % until information is confirmed correct, rerun prompt
while flag == 'n';
prompt = 'Enter your participant number. Ex: 1,2,3, exc.';
```

```
participant = input(prompt);

prompt = 'Enter your Experimental Type. Ex: 1,2,3, exc.';
vib_setup = input(prompt);

prompt = 'Confirm if this information is correct. Type "y" for yes or "n" for no';
flag = input(prompt,'s');
clc
end
```

C.1.4 Workaround Code so that MATLAB Writes Into Excel Much Faster

```
participant_id = num2str(participant);
vib_setup_id = num2str(vib_setup);
filename = strcat('Subject',participant_id);
File = strcat('C:\Users\mpw\Desktop\Morse_exp\',filename,'.csv');
```

C.1.5 Instructions for Segments of Experiment

```
while Test <= Trail_amount;
 if Test == 1
 prompt = 'Have you completed this experiment yet? Type "y" for yes or "n" for no';
 skip = input(prompt,'s');
 clc
 if strcmp(skip,'y') == 1
 Test = 5;
 end

 end
 % take verbal test in lieu of competancy test incase perception gates
 %

  if Test == 1
 prompt = 'Do you want to take the verbal test?  Type "y" for yes or "n" for no';
 skip = input(prompt,'s');
 clc
 if strcmp(skip,'y') == 1
 Test = 4;
 end

 end
    % skips old code that teaches 3 charters in a row.
    if Test == 3
       Test = 4;
    end

if Test == 1
 Part1 = imread();
 subplot(1,1,1), imshow(Part1);
 disp('Instructions: in Phase 1, you will learn the Morse code for 12')
 disp('letters/numbers (D, H, J, S, U, V, W, 1, 4, 5, 7, 8). You will be shown a character')
 disp('with its respective Morse code underneath it. The device attached to your')
 disp('arm(s) will then alert you via a vibration or a pressing sensation of')
 disp('whether a Morse code element is a dot or dash.')
 disp(' ')
 disp(' ')
 prompt = 'Press “Enter” to begin';
 str = input(prompt,'s');
 clc
end
```

```
if Test == 2
 Part1 = imread();
 subplot(1,1,1), imshow(Part1);
 disp('Instructions: in Phase 2, you will be practicing your knowledge')
 disp('on the 12 characters you just learned.  The device on your arm(s)')
 disp('will relay to you Morse code elements. You will then type the')
 disp('character you think was relayed to you. After you have done this,')
 disp('you can see if your answer was correct or not. On the top of the')
 disp('display, you can see how many letters you')
 disp('got correct and what problem you are on out of the total problems')
 disp('there are for the set. You need to recieve a score of 80% or higher to')
 disp('proceed to the next portion of the experiment')
 disp(' ')
 disp(' ')
 prompt = 'Press "Enter" to continue';
 str = input(prompt,'s');
 clc
end

if Test == 3
 Part1 = imread();
 subplot(1,1,1), imshow(Part1);
 disp('Instructions: In Phase 3, you will be learning how to understand')
 disp('Morse code when characters are sent in series. There will be a pause')
 disp('from the device you are wearing to indicate when a letter has ended')
 disp('and a new one begins.')
 disp(' ')
 disp(' ')
 prompt = 'Press "Enter" to continue';
 str = input(prompt,'s');
 clc
end

if Test == 4
 Part1 = imread();
 subplot(1,1,1), imshow(Part1);
 disp('Instructions: In Phase 4, you will be practicing your ability to')
 disp('identify 3 letters in series. After all 3 letters have been alerted')
 disp('to you, you can then enter your answers. For the remainder of the')
 disp('problem sets, you will be given these types of problems with 3')
 disp('characters in series. This is a practice set; your results will')
 disp('not be recorded.')
 disp(' ')
 disp(' ')
 prompt = 'Press "Enter" to continue';
 str = input(prompt,'s');
 clc
end

if Test == 5
 Part1 = imread();
 subplot(1,1,1), imshow(Part1);
 disp('In Phase 5, you will be tested on your ability to identify 3 characters')
 disp('in series. After all 3 letters have been alerted to you, you can')
 disp('then enter your answers. Answers are to be entered one at a time')
 disp('This is the first out of three test sets. Your results will be recorded.')
 disp(' ')
 disp(' ')
 prompt = 'Press "Enter" to continue';
```

```
 str = input(prompt,'s');
 clc
end

if Test == 6
 Part1 = imread();
 subplot(1,1,1), imshow(Part1);
 disp('Instructions: In Phase 6, you will be tested on your ability to')
 disp('identify 3 characters in series. After all 3 characters have been alerted')
 disp('to you, you can then enter your answers. Characters will now be alerted')
 disp('to you at a slightly faster rate.')
 disp(' ')
 disp(' ')
 prompt = 'Press "Enter" to continue';
 str = input(prompt,'s');
 clc
end

if Test == 7
 Part1 = imread();
 subplot(1,1,1), imshow(Part1);
 disp('Instructions: In Phase 7, you will be tested on your ability to')
 disp('identify 3 characters in series. After all 3 characters have been alerted')
 disp('to you, you can then enter your answers. The pace of the characters')
 disp('has been further increased. This is the final phase')
 disp('of the experiment.')
 disp(' ')
 disp(' ')
 prompt = 'Press "Enter" to continue';
 str = input(prompt,'s');
 clc
end
```

C.1.6 Element Time Durations Calculated for Setup 1 (Traditional)

```
wpm =15 ;
PARIS = (50/60);    %50 elements/sec for the word "paris"
element_time = 1/(wpm*(PARIS));      %seconds/element
dot = element_time;
dash = 3*element_time;
null = 0;  % null variable to allow formation of matrix
element_gap = element_time;
letter_gap = 3 - increase;
```

C.1.7 Recalculate Element and Letter Spacing for Setup 5 (OLD)

```
% Unused code
if vib_setup == 5
n_dots = 31;
n_dashes = 17;
n_spaces = 37;
n_letter_spaces = 8;
n_sub_spaces = 25;
total_space_time = (n_spaces*element_gap)+ n_letter_spaces*letter_gap;
time_val_trad = (n_dots*dot)+(n_dashes*dash)+ total_space_time;
total_space_time = n_sub_spaces*element_gap + n_letter_spaces*letter_gap;
total_ele_time = (n_dots*dot) + (n_dashes*dot);
time_val_sub = total_ele_time + total_space_time;
excess_time = time_val_trad - time_val_sub;
new_total_ele_time = total_ele_time + excess_time;
```

```
syms dot
eqn = (n_dots*dot) + (n_dashes*dot) == new_total_ele_time;
dot = double(solve(eqn,dot));
time_val_new = (n_dots*dot)+(n_dashes*dot)+ total_space_time;
end
```

C.1.8 Recalculate Element and Letter Spacing for Setups 6 and 7 (OLD)

```
if vib_setup == 7 || vib_setup == 6
n_dots = 31;
n_dashes = 17;
n_spaces = 37;
n_letter_spaces = 8;

% find how much time it takes to express the 12 letters in the letter set
% in a string. Then, set dash duration to dot duration and resize element
% and letter time gaps so that it takes the same time.

total_space_time = (n_spaces*element_gap)+n_letter_spaces*letter_gap;
time_val_trad = (n_dots*dot)+(n_dashes*dash)+ total_space_time;
time_val_bi = (n_dots*dot)+(n_dashes*dot)+ total_space_time;
excess_time = time_val_trad - time_val_bi;
new_total_space_time = total_space_time + excess_time;
syms new_ele_gap
eqn = n_spaces*element_gap + n_letter_spaces*9*new_ele_gap == new_total_space_time;
element_gapp = double(solve(eqn,new_ele_gap));
letter_gap = (9*element_gapp);
total_space_time = (n_spaces*element_gap)+n_letter_spaces*letter_gap;

% code that validates if new element gap timing results in identical test
% duration as traditional method.
time_val_new = (n_dots*dot)+(n_dashes*dot)+ total_space_time;
end
```

C.1.9 Morse Code Identifiers

```
% matlab likes all the strings to be the same character leangth, hence:
% "dott" is spelled like this and not like "dot".

MorseD = {'dash'; 'dott'; 'dott'; 'null'; 'null'};
MorseH = {'dott'; 'dott'; 'dott'; 'dott'; 'null'};
MorseJ = {'dott'; 'dash'; 'dash'; 'dash'; 'null'};
MorseS = {'dott'; 'dott'; 'dott'; 'null'; 'null'};
MorseU = {'dott'; 'dott'; 'dash'; 'null'; 'null'};
MorseV = {'dott'; 'dott'; 'dott'; 'dash'; 'null'};
MorseW = {'dott'; 'dash'; 'dash'; 'null'; 'null'};
Morse1 = {'dott'; 'dash'; 'dash'; 'dash'; 'dash'};
Morse4 = {'dott'; 'dott'; 'dott'; 'dott'; 'dash'};
Morse5 = {'dott'; 'dott'; 'dott'; 'dott'; 'dott'};
Morse7 = {'dash'; 'dash'; 'dott'; 'dott'; 'dott'};
Morse8 = {'dash'; 'dash'; 'dash'; 'dott'; 'dott'};
```

C.1.10 Identifying Letters from Answer Key Permutation

```
    for Cycle = 1:1:Cycle_amount;
p=randperm(n);  % permuntating 12 integers to represent answers per cycle

L_answer = cell(1,n);
%M_answer = cell(n,3);
```

```
for i = 1:1:n

if p(i) == 1
    L_answer(i) = {};
    M_answer{i} = MorseD;
    correct_letter{i} = 'd';
end
if p(i) == 2
    L_answer(i) = {};
    M_answer{i} = MorseH;
    correct_letter{i} = 'h';
end
if p(i) == 3
    L_answer(i) = {};
    M_answer{i} = MorseJ;
    correct_letter{i} = 'j';
end
if p(i) == 4
    L_answer(i) = {};
    M_answer{i} = MorseS;
    correct_letter{i} = 's';
end
if p(i) == 5
    L_answer(i) = {};
    M_answer{i} = MorseU;
    correct_letter{i} = 'u';
end
if p(i) == 6
    L_answer(i) = {};
    M_answer{i} = MorseV;
    correct_letter{i} = 'v';
end
if p(i) == 7
    L_answer(i) = {};
    M_answer{i} = MorseW;
    correct_letter{i} = 'w';
end
if p(i) == 8
    L_answer(i) = {};
    M_answer{i} = Morse1;
    correct_letter{i} = '1';
end
if p(i) == 9
    L_answer(i) = {};
    M_answer{i} = Morse4;
    correct_letter{i} = '4';
end
if p(i) == 10
    L_answer(i) = {};
    M_answer{i} = Morse5;
    correct_letter{i} = '5';
end
if p(i) == 11
    L_answer(i) = {};
    M_answer{i} = Morse7;
    correct_letter{i} = '7';
end
if p(i) == 12
    L_answer(i) = {};
    M_answer{i} = Morse8;
```

```
    correct_letter{i} = '8';
end
end
%end
```

C.1.11 Identifying User Input Letters

```
% preallocating array sizes
L_user = cell(1,n);
i = 0;

while i < n
val = 0; % variable that counts letters
flag = 0; % variable that identifies if an incorrect letter is pressed

% how many letters sequenced
if Test <=2
    Letter_amount = 1;
else
    Letter_amount = 3;
end
tic;
while val < Letter_amount
i = i + 1;
% dont ask for user input for test 1 and test 3
if Test == 1 || Test == 3
break
end
```

C.1.12 Sending Morse Code to User

```
if Test == 2
temp = M_answer{i};
if pass == 1;    % dont activate motors again if invalid letter choice was given.
    %pause(letter_gap)
Trad_morse(A_Uno, element_time, element_gap, temp, vib_setup);
end
else
if pass == 1;    % dont activate motors again if invalid letter choice was given.
    j = 0;
while j < 3
        pause(letter_gap)
temp = M_answer{i + j};
j = j + 1;
Trad_morse(A_Uno, element_time, element_gap, temp, vib_setup);
end
end
end

pass = 0;

    prompt = 'What letter did you feel?';
str = input(prompt,'s');        %storing input as string

if str == 'd'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = 'd';
    flag = 1;
end
```

```
if str == 'h'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = 'h';
    flag = 1;
end
if str == 'j'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = 'j';
    flag = 1;
end
if str == 's'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = 's';
    flag = 1;
end
if str == 'u'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = 'u';
    flag = 1;
end
if str == 'v'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = 'v';
    flag = 1;
end
if str == 'w'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = 'w';
    flag = 1;
end
if str == '1'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = '1';
    flag = 1;
end
if str == '4'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = '4';
    flag = 1;
end
if str == '5'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = '5';
    flag = 1;
end
if str == '7'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = '7';
    flag = 1;
end
```

```
if str == '8'
    L_user(i) = {};
    val = val + 1;
    user_letter{i} = '8';
    flag = 1;
end
response_time{i} = toc;
current_cycle{i} = Cycle;
if flag == 0
disp('the letter you entered is invalid');
disp('please enter a valid letter');
i = i - 1;
end
flag = 0;
end

problem = problem + 1;
pass = 1;
```

C.1.13 Creating Display Interface for Test 1

```
if Test == 1

str2 = sprintf('%g more sets to go',Total_letters - problem);

[answer] = imread(L_answer{i});
subplot(1,1,1), imshow(answer);
title(str2)

%pause(2)  %lets user read picture before stimulus is sent.

% sent morse code stimulus
temp = M_answer{i};
Trad_morse(A_Uno, element_time, element_gap, temp, vib_setup)

%pause(0.5)

prompt = 'Press "Enter" to continue';
str = input(prompt,'s');
clc
end
```

C.1.14 Creating Display Interface for Test 2

```
if Test == 2
% evaluating correctness of letters

if L_user{i} == L_answer{i}
[correctness] = imread();
correct = correct + 1;
c1 = c1 + 1;
c_data{i} = 1;
else
[correctness] = imread();
c_data{i} = 0;
end

[answer] = imread(L_answer{i});
[user_input] = imread(L_user{i});
```

C.1.15 Plotting Test 2

```
str1 = sprintf('%g Correct Letters                ',correct);
str2 = sprintf('%g more to go',Total_letters - problem);
%str3 = sprintf('Response time %g s',round(response_time,2,'significant'));
str4 = 'You Chose';
str5 = 'Correct Answer';
str6 = sprintf('%s              %s',str1,str2);
str7 = '_____________________________________________________________________________________________________'

if c1 == 0
subplot(1,3,1), imshow(user_input)
title (str4)
subplot(1,3,2), imshow(correctness)
%title (str2)
subplot(1,3,3), imshow(answer)
title (str5)
else
subplot(1,2,1), imshow(user_input)
title (str4)
subplot(1,2,2), imshow(correctness)
%title (str2)
end

subtitle(str6);
subtitle(str7);

c1 = 0;
warning off
 prompt = 'Press "Enter" to continue';
 str = input(prompt,'s');
warning on
 clc
 clf      % clear figure so subject is not distracted
 %pause(1) % allow subject to ready himself
end
```

C.1.16 Creating Display Interface for Test 3

```
if Test == 3
i = i + 2;

str2 = sprintf('%g more sets to go',(Total_letters - problem*3)/3);

[answer1] = imread(L_answer{i-2});
[answer2] = imread(L_answer{i-1});
[answer3] = imread(L_answer{i});

subplot(1,3,1), imshow(answer1)
subplot(1,3,2), imshow(answer2)
title(str2)
subplot(1,3,3), imshow(answer3)

%pause(2)  %lets user read picture before stimulus is sent.

j = 2;
while j >= 0
       pause(letter_gap)
temp = M_answer{i - j};
```

```
j = j - 1;
Trad_morse(A_Uno, element_time, element_gap, temp, vib_setup);
end

%pause(0.5)

prompt = 'Press "Enter" to continue';
str = input(prompt,'s');
clc

end
```

C.1.17 Creating Display Interface for Test 4,5,6,7

```
if Test >= 4
% evaluating correctness of letters
if L_user{i-2} == L_answer{i-2}
[correctness1] = imread();
correct = correct + 1;
c1 = c1 + 1;
c_data{i-2} = 1;
else
[correctness1] = imread();
c_data{i-2} = 0;
end
% _____
if L_user{i-1} == L_answer{i-1}
[correctness2] = imread();
correct = correct + 1;
c1 = c1 + 1;
c_data{i-1} = 1;
else
[correctness2] = imread();
c_data{i-1} = 0;
end
%_____
if L_user{i} == L_answer{i}
[correctness3] = imread();
correct = correct + 1;
c1 = c1 + 1;
c_data{i} = 1;
else
[correctness3] = imread();
c_data{i} = 0;
end
[answer1] = imread(L_answer{i-2});
[answer2] = imread(L_answer{i-1});
[answer3] = imread(L_answer{i});
[user_input1] = imread(L_user{i-2});
[user_input2] = imread(L_user{i-1});
[user_input3] = imread(L_user{i});
```

C.1.18 Plotting Test 4,5,6,7

```
str1 = sprintf('%g Correct Letters',correct);
str2 = sprintf('%g more sets to go',(Total_letters - problem*3)/3);
%str3 = sprintf('Response time %g s',round(response_time,2,'significant'));
str4 = 'You Chose';
str5 = 'Correct Answer';
str6 = sprintf('%s            %s',str1,str2);
```

```
str7 =
'_____________________________________________________________________________________________'

if c1 < 3
subplot(3,3,1), imshow(user_input1)
subplot(3,3,2), imshow(user_input2)
subplot(3,3,3), imshow(user_input3)
subplot(3,3,4), imshow(correctness1)
subplot(3,3,5), imshow(correctness2)
subplot(3,3,6), imshow(correctness3)
subplot(3,3,7), imshow(answer1)
subplot(3,3,8), imshow(answer2)
subplot(3,3,9), imshow(answer3)

title(subplot(3,3,1),{str1},'FontSize', 14, 'FontWeight', 'bold');
title(subplot(3,3,2),{str7},'FontSize', 14, 'FontWeight', 'bold');
title(subplot(3,3,3),{str2},'FontSize', 14, 'FontWeight', 'bold');
title(subplot(3,3,5),{str4;str7})
title(subplot(3,3,8),{str7;str5})

else
subplot(2,3,1), imshow(user_input1)
subplot(2,3,2), imshow(user_input2)
subplot(2,3,3), imshow(user_input3)
subplot(2,3,4), imshow(correctness1)
subplot(2,3,5), imshow(correctness2)
subplot(2,3,6), imshow(correctness3)

title(subplot(2,3,1),{str1;' '})
title(subplot(2,3,2),{' ';str7;' '})
title(subplot(2,3,3),{str2;' '})
title(subplot(2,3,5),{str4;str7})

end
c1 = 0;
warning off
 prompt = 'Press "Enter" to continue';
 str = input(prompt,'s');
warning on
 clc
 clf      % clear figure so subject is not distracted
 pause(1) % allow subject to ready himself
end
end
% proceed to next test after amount of cycles have been satisfied

if Test >= 5;

    % opening up Excel
%Excel = actxserver('Excel.Application');
%if ~exist(File,'file')
%    ExcelWorkbook = Excel.workbooks.Add;
%    ExcelWorkbook.Sheets.Add;
%    ExcelWorkbook.SaveAs(File,1);
%    ExcelWorkbook.Close(false);
%end
%invoke(Excel.Workbooks,'Open',File);


for i = 1:1:n
```

```
user_value = {'User Letter';user_letter{i}};
answer_value = {'Answer Letter';correct_letter};
letters_correct = {'Letters Correct';c_data};
Reaction_time = {'Reaction Time';response_time};

cell_number = cell_number + 1;
cell_id = num2str(cell_number);

% create string to be used as cell location for each column
col1 = strcat('A',cell_id);
col2 = strcat('B',cell_id);
col3 = strcat('C',cell_id);
col4 = strcat('D',cell_id);
col5 = strcat('E',cell_id);
col6 = strcat('F',cell_id);

%Recording = imread('Letters/Recording_Data.jpg');
%imshow(Recording)
% write data to respected column
sheetname = strcat('Setup',vib_setup_id);
warning off

%xlswrite(File,user_letter{i},sheetname,col1);
%xlswrite(File,correct_letter{i},sheetname,col2);
%xlswrite(File,c_data{i},sheetname,col3);
%xlswrite(File,response_time{i},sheetname,col4);
%xlswrite(File,current_cycle{i},sheetname,col5);
%xlswrite(File,Test,sheetname,col6);
%response_time{i}
dlmwrite(File,[user_letter{i} correct_letter{i} num2str(c_data{i}) num2str(current_cycle{i}) num2str(Test) vib_setup_id],'-
append','delimiter',',')

warning on
n;
%sprintf('%g % of the data has been recorded',ceil(i/n*100))
i;
%if i == 12 % pause for excel to close
%pause(2)
%end
end
clc
%prompt = 'Press "Enter" to Continue.';
%participant = input(prompt);
%clc
%invoke(Excel.ActiveWorkbook,'Save');
%Excel.Quit
%Excel.delete
%clear Excel
end

flag1 = 0;
if Cycle == Cycle_amount
    if Test == 2
    score = correct/Total_letters;
    if score >= 0.8   % particpant must recieve a score of 80% or higher
    Test = Test + 1;
    else
    fprintf('You recieved a score of %f percent. You require a',double(score*100))
    disp('score of 80% or greater to proceed to the next phase.')
    disp('You must re-take the practice phase')
```

```
   prompt = 'Press "Enter" to continue';
   str = input(prompt,'s');
   clc
   end
   else
   Test = Test + 1;
   end
   Cycle = 0;
   problem = 0;
   correct = 0;

   if Test >= 5
      increase = increase + 1
   end
end
   Cycle = Cycle + 1;    % go to next cycle and permutate new letters
%end
```

C.1.19 Write Data to Excel Sheet after Test Completion

```
% skip  writing data if on learning tests 1 or 3.
   end
end
Done = imread();
imshow(Done)
```

C.2 Setup Function

C.2.1 Contents

- vibration motor setup
- Vibration setup 1: Traditional Morse code
- Vibration setup 2: left/right presentation with same dash duration
- Vibration setup 3: left/right presentation with dot = dash
- Vibration setup 4, counting with three motors
- Vibration setup 5: bilateral subitizing (removed from experiment)

```
function Trad_morse(A_Uno, element_time, element_gap, temp, vib_setup)
```

C.2.2 Vibration Setup 1: Traditional Morse Code

```
if vib_setup == 1;
for j = 1:1:5
duration = temp{j};
if strcmp(duration,'dott') == 1
   duration = element_time;
   writeDigitalPin(A_Uno, 11, 1);  % activate pin
   pause(duration)              % stay active for set duartion
   writeDigitalPin(A_Uno, 11, 0);  % deactivate pin
   pause(element_gap)
end
if strcmp(duration,'dash') == 1
   duration = 3*element_time;
   writeDigitalPin(A_Uno, 11, 1);  % activate pin
   pause(duration)              % stay active for set duartion
   writeDigitalPin(A_Uno, 11, 0);  % deactivate pin
   pause(element_gap)
end
```

```
if strcmp(duration,'null') == 1
    duration = 0;
end

end
end
```

C.2.3 Vibration Setup 2: Left/Right Presentation

```
if vib_setup == 2;
for j = 1:1:5
duration = temp{j};
if strcmp(duration,'dott') == 1
    duration = element_time;
    n = 12;
end
if strcmp(duration,'dash') == 1
    duration = 3*element_time;
    n = 9;
end
if strcmp(duration,'null') == 1
    duration = 0;
end
if duration ~= 0   % dont output null elements (causes a flicker of activation)
digitalout2(A_Uno, n, duration, element_gap);
end
end
end
```

C.2.4 Vibration Setup 3: Left/Right Presentation with Dot Equal Dash

```
if vib_setup == 3;
for j = 1:1:5
duration = temp{j};
if strcmp(duration,'dott') == 1
    duration = element_time;
    n = 12;
end
if strcmp(duration,'dash') == 1
    duration = element_time;
    n = 9;
end
if strcmp(duration,'null') == 1
    duration = 0;
end
if duration ~= 0   % dont output null elements (causes a flicker of activation)
digitalout2(A_Uno, n, duration, element_gap);
end
end
end
```

C.2.5 Vibration Setup 4: Counting with Three Motors

```
if vib_setup == 4 || vib_setup == 6;
    dott_counter = 0;   % counters used to track repeated dott or dashes.
    dash_counter = 0;
for j = 1:1:5

    if j == 1
        prev_element = 0;
```

```
    end

duration = temp{j};
if strcmp(duration,'dott') == 1
    duration = element_time;
    dash_counter = 0;
    % determine if previous element is identical to current one

    if duration == prev_element % after first element, count sequence.
    dott_counter = dott_counter + 1;
    end

    if dott_counter == 0;
    writeDigitalPin(A_Uno, 12, 1);  % activate pin
    pause(duration)                 % stay active for set duartion
    writeDigitalPin(A_Uno, 12, 0);  % deactivate pin
    pause(element_gap)
    end

    if dott_counter == 1;
    writeDigitalPin(A_Uno, 12, 1);  % activate pin
    writeDigitalPin(A_Uno, 11, 1);  % activate pin
    pause(duration)                 % stay active for set duartion
    writeDigitalPin(A_Uno, 12, 0);  % deactivate pin
    writeDigitalPin(A_Uno, 11, 0);  % deactivate pin
    pause(element_gap)
    end

    if dott_counter == 2;
    writeDigitalPin(A_Uno, 12, 1);  % activate pin
    writeDigitalPin(A_Uno, 11, 1);  % activate pin
    writeDigitalPin(A_Uno, 10, 1);  % activate pin
    pause(duration)                 % stay active for set duartion
    writeDigitalPin(A_Uno, 12, 0);  % deactivate pin
    writeDigitalPin(A_Uno, 11, 0);  % deactivate pin
    writeDigitalPin(A_Uno, 10, 0);  % activate pin
    pause(element_gap)
    dott_counter = 0;
    end


prev_element = duration;
end
if strcmp(duration,'dash') == 1
    if vib_setup == 6
    duration = 3*element_time;
    else
    duration = element_time;
    end
    dott_counter = 0;
    % determine if previous element is identical to current one

    if duration == prev_element   % after first element, count sequence.
    dash_counter = dash_counter + 1;
    end


    if dash_counter == 0;
    writeDigitalPin(A_Uno, 9, 1);  % activate pin
    pause(duration)                % stay active for set duartion
```

```
    writeDigitalPin(A_Uno, 9, 0);  % deactivate pin
    pause(element_gap)
    end

    if dash_counter == 1;
    writeDigitalPin(A_Uno, 9, 1);  % activate pin
    writeDigitalPin(A_Uno, 8, 1);  % activate pin
    pause(duration)               % stay active for set duartion
    writeDigitalPin(A_Uno, 9, 0);  % deactivate pin
    writeDigitalPin(A_Uno, 8, 0);  % deactivate pin
    pause(element_gap)
    end

    if dash_counter == 2;
    writeDigitalPin(A_Uno, 9, 1);  % activate pin
    writeDigitalPin(A_Uno, 8, 1);  % activate pin
    writeDigitalPin(A_Uno, 7, 1);  % activate pin
    pause(duration)               % stay active for set duartion
    writeDigitalPin(A_Uno, 9, 0);  % deactivate pin
    writeDigitalPin(A_Uno, 8, 0);  % deactivate pin
    writeDigitalPin(A_Uno, 7, 0);  % activate pin
    pause(element_gap)
    dash_counter = 0;
    end

prev_element = duration;
end
%if strcmp(duration,'null') == 1
%    duration = 0;
%end

%if duration ~= 0   % dont output null elements (causes a flicker of activation)
%digitalout2(A_Uno, n, duration, element_gap);
%end

end
end
```

C.2.6 Vibration Setup 5: Bilateral Subtilizing (OLD)

```
if vib_setup == 5
  for j = 1:1:5

      duration = temp{j};
      if strcmp(duration,'dott')  == 1
      vib_type(j) = 1;
      end

      if strcmp(duration,'dash')  == 1
      vib_type(j) = 2;
      end
  end
   s = SplitVec(vib_type);
   groups = length(s);
   for i = 1:1:groups
   cluster = s{i};
   cluster_size = length(cluster);
   if cluster(1) == 1        %dot
        n = 12;
        for m = 1:1:cluster_size
```

```
        if n < 10
            pause(element_time)
        writeDigitalPin(A_Uno, 12, 0);
        writeDigitalPin(A_Uno, 11, 0);
        writeDigitalPin(A_Uno, 10, 0);
            pause(element_gap)
        writeDigitalPin(A_Uno, 12, 1);
        writeDigitalPin(A_Uno, 11, 1);
        writeDigitalPin(A_Uno, 10, 1);
            n = n + 1;
        end
        writeDigitalPin(A_Uno, n, 1);  % activate pin
        n = n - 1;
        end
        pause(element_time)
        writeDigitalPin(A_Uno, 12, 0);
        writeDigitalPin(A_Uno, 11, 0);
        writeDigitalPin(A_Uno, 10, 0);
        pause(element_gap)
    end

    if cluster(1) == 2         %dash
        n = 9;
        for m = 1:1:cluster_size
            if n < 7
                pause(element_time)
        writeDigitalPin(A_Uno, 9, 0);
        writeDigitalPin(A_Uno, 8, 0);
        writeDigitalPin(A_Uno, 7, 0);
                pause(element_gap)
        writeDigitalPin(A_Uno, 9, 1);
        writeDigitalPin(A_Uno, 8, 1);
        writeDigitalPin(A_Uno, 7, 1);
                n = n + 1;
            end
        writeDigitalPin(A_Uno, n, 1);  % activate pin
        n = n - 1;
        end
        pause(element_time)
        writeDigitalPin(A_Uno, 9, 0);
        writeDigitalPin(A_Uno, 8, 0);
        writeDigitalPin(A_Uno, 7, 0);
        pause(element_gap)
    end
    end
end


% END OF FUNCTION
end
```

C.3 Motor Test

```
clc
clear all
A = arduino('com4','uno');
```

```
disp('LEFT MOTOR 1')
writeDigitalPin(A, 12, 1);
pause(3)
writeDigitalPin(A, 12, 0);
disp('done')
pause(3)

disp('LEFT MOTOR 2')
writeDigitalPin(A, 11, 1);
pause(3)
writeDigitalPin(A, 11, 0);
disp('done')
pause(3)

disp('LEFT MOTOR 3')
writeDigitalPin(A, 10, 1);
pause(3)
writeDigitalPin(A, 10, 0);
disp('done')
pause(3)

disp('RIGHT MOTOR 1')
writeDigitalPin(A, 9, 1);
pause(3)
writeDigitalPin(A, 9, 0);
disp('done')
pause(3)

disp('RIGHT MOTOR 2')
writeDigitalPin(A, 8, 1);
pause(3)
writeDigitalPin(A, 8, 0);
disp('done')
pause(3)

disp('RIGHT MOTOR 3')
writeDigitalPin(A, 7, 1);
pause(3)
writeDigitalPin(A, 7, 0);
disp('done')
pause(3)
```