ISTANBUL BILGI UNIVERSITY

INSTITUTE OF GRADUATE PROGRAMS

ELECTRICAL & ELECTRONIC ENGINEERING MASTER

DEGREE PROGRAM

DEVELOPMENT AND CHARACTERIZATION OF A WEARABLE RING PROVIDING HAPTIC FEEDBACK

FRANCK PAULIN LUDOVIG PEHN MAYO

119815016

Assist. Prof. Mehmet Ayyildiz

ISTANBUL

2022

DEVELOPMENT AND CHARACTERIZATION OF A WEARABLE RING PROVIDING HAPTIC FEEDBACK

HAPTİK GERİ BİLDİRİM SAĞLAYAN GİYİLEBİLİR BİR YÜZÜK GELİŞTİRİLMESİ VE KARAKTERİZASYONU

Franck Paulin Ludovig Pehn Mayo

119815016

Tez Danışmanı:	Prof. Dr. Mehmet Ayyildiz İstanbul Bilgi Üniversitesi	(İmza): ……
Jüri Üyesi:	Prof. Dr. Baykal Sarioglu İstanbul Bilgi Üniversitesi	(İmza): ……
	Prof. Dr Yusuf Aydin MEF Üniversitesi	(İmza): ……

Tezin Onaylandığı Tarih : 28.08.2022
Toplam Sayfa Sayısı : 82

Anahtar Kelimeler:	Keywords:
1) Dokunsal geribildirim	1) Haptic feedback
2) Titreşim	2) Vibration
3) Giyilebilir cihaz	3) Wearable device
4) Kalınlık	4) Thickness

ACKNOWLEDGEMENTS

I want to give my appreciation to my great supervisor Prof. Dr. Mehmet Ayyildiz who helped me through all the stages of this Thesis with his continued engagement, support, guidance, and tolerance during the process.

I would like to thank Kansu Oguz Canbek for contributing immensely by helping me in the coding aspect.

To all my fellow students and staff of the Department of Electrical and Electronics Engineering and the faculty as a whole, I would like to thank you all for being in my life during my educational journey and helping me grow.

Most importantly, to my dearest parents who were there for me from the beginning to the end of my studies.

ABSTRACT

DEVELOPMENT AND CHARACTERIZATION OF A WEARABLE RING PROVIDING HAPTIC FEEDBACK

Touchscreens can be found on a variety of devices, including ATMs, phones, laptops, kiosks, automobiles, and many others. In those applications, the user navigates to the interface with his finger to interact with the tactile information using instincts. Roughly, every haptic device uses vibration to interact with the user. The user's haptic experience is delicate due to the complexity of the tactile information. Associating kinesthetic experience is a key subject in enhancing the human haptic experience. In this investigation, we developed a novel characterized haptic ring equipped with up to four LRA motors in order to improve the human haptic perception. The proposed device uses subcutaneous clues to replicate the forces and moments applied to the proximal interphalangeal joint of the index finger. Thresholding experiments are conducted using a total of nine subjects to evaluate the effects of the vibration amplitude, line thickness, and orientation on human perception. The results showed that as the line's thickness and amplitude increased, so did the participants' ability to identify the shapes. The human participants can detect a minimum thickness of 2 mm at the lowest possible vibration level with a single motor. Additionally, the second vibration amplitude and vertical orientation result in the best recognition rates and times besides the control.

Keywords: haptic feedback, vibration, wearable device, thickness, amplitude, time, kinesthetic, orientation.

ÖZET

HAPTİK GERİ BİLDİRİM SAĞLAYAN GİYİLEBİLİR BİR YÜZÜK GELİŞTİRİLMESİ VE KARAKTERİZASYONU

Dokunmatik ekranlar, ATM'ler, telefonlar, dizüstü bilgisayarlar, kiosklar, otomobiller ve diğerleri dahil olmak üzere çeşitli cihazlarda bulunurlar. Bu uygulamalarda kullanıcı dokunsal bilgileri ve parmağını arayüzle etkileşim kurmak için kullanır. Kabaca, her dokunsal cihaz, kullanıcı ile etkileşim kurmak için titreşim kullanır. Kullanıcının dokunsal deneyimi, dokunsal bilginin karmaşıklığı nedeniyle hassastır. Kinestetik deneyimi ilişkilendirmek, insan haptik deneyimini geliştirmede kilit bir konudur. Bu araştırmada, insan haptik algısını iyileştirmek için dört adet LRA motoruyla donatılmış yeni bir haptik yüzük geliştirdik. Titreşim genliği, çizgi kalınlığı ve yönelimin insan algısı üzerindeki etkilerini değerlendirmek için toplam dokuz denek kullanılarak eşik deneyleri ölçüldü. Sonuçlar, çizginin kalınlığı ve genliği arttıkça, katılımcıların şekilleri tanımlama yeteneklerinin de arttığını gösterdi. İnsan katılımcılar, tek bir motorla mümkün olan en düşük titreşim seviyesinde minimum 2 mm'lik bir kalınlığı tespit etti. Ek olarak, ikinci titreşim genliği ve dikey yönlendirme, en iyi tanıma oranları ve süreleri ile sonuçlandı.

Anahtar Kelimeler: dokunsal geribildirim, titreşim, giyilebilir cihaz, kalınlık, genlik, zaman, kinestetik, yönelim.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1D:	One Dimension
LRA:	Linear Resonant Actuator
DC:	Direct Current
V:	Volt
A:	Ampere
UART:	Universal Asynchronous Receiver-Transmitter
CAN:	Controller Area Network
SPI:	Serial Peripheral Interface
I2C:	Inter-Integrated Circuit
GUI:	Graphical User Interface
GUIDE:	Graphical User Interface Development Environment
IDII:	Interaction Design Institute Ivrea
IoT:	Internet of a Thing
PWM:	Pulse Width Modulation
LoRa:	Long Range Radio
EEPROM:	Electrically Erasable Programmable Read-Only memory
IMU:	Inertia Measurement Unit
EMMC:	Embedded Multi-Media Card
DDR4 SDRAM:	Double Data Rate Fourth Generation Synchronous Dynamic
RAM:	Random-access Memory
SPI:	Serial Peripheral Interface
ERM:	Eccentric Rotational Mass
TPU:	Tensor processing

1. INTRODUCTION

This chapter emphasizes the problem statement and the purpose of the study. Also, it underlines its importance, limitations, and overall overview.

1.1 Background

Touching is a feeling that everyone is being familiar with. Experiencing touch daily regardless of one's age shows how it's an essential part of all departments of our existence such as mental actions, emotion, and doings [1]. A single touch may undoubtedly have a tremendous impact in numerous ways. Consequently, engineers and scientists are conducting extensive research to enhance the human touch–technology interaction via an interface called haptic feedback. A haptic device's major objective is to simulate the main aspects of touch and holding of an object such as forms, textures, weight, and others that are available within a simulated environment from the experience of interaction with a user [2] [3]. There are two specific modes of haptic feedback. Whenever an object is in contact with the user, the tactile reaction refers to the static status influenced by the skin's nerve terminal information whereas the kinesthetic reaction relates to the motion that occurred during that contact [4]. The kinesthetic, from a physiological standpoint, has to do with being aware of the position and motion inside that space. It is the portion of the somatosensory system that is aware of the body's interpretation and is carried throughout the body [5]. Texture, temperature, pressure, and vibration are all factors that are strongly attributable to cutaneous or tactile mechanical receptors in the skin [6].Furthermore, the glabrous skin (smooth, hairless) is by far the most sensitive to touch [7]. The kind of tactile or kinesthetic actuation applied to a body, the interaction between the user and the technology, and the virtual object algorithm program used all play a significant role in

replicating the stimulation responsible for the immersive and realistic perception of the user [8]. It is then essential to have a deep understanding in the kinesthetic and tactile effect in order to improve any user experience.

1.2 Problem statement

The growth of devices that uses haptic are becoming very consequent. Almost every household has at least a touchscreen device. The devices help the user to deeper interact with the haptic interface and give a more immersive experience. Mostly all our daily basis devices just communicate through tactile vibration. The tactile vibration outputs are sometimes complicated to understand. Furthermore, the kinesthetic aspect is often forgotten when designing the haptic devices.

1.3 The aim of the study

This study aims to develop a characterized flexible haptic ring will help us understand its impact when 1D geometric shapes are displayed on a touchscreen using a GUI. The main aim will be to find out the minimum thickness and amplitude threshold needed for the user to feel the vibration when the finger is inside the desired shapes. Concerning the shapes, there are two different orientations: vertical and horizontal. The prototype can be used by anyone since the ring is flexible. The Arduino Uno costs 10 USD, four LRA motors cost 3 USD, the driving motors module (DRV2605LEV-MD) costs 238 USD, a pack of jumping wires costs 1 USD, a DC power supply adapter (5V -10A) Costs 5 USD.

1.4 The significance of the study

Touchscreens are part of the numerous haptic gadgets that are used on a daily basis. In order to enhance the user experience, determining the minimum cross area or thickness the finger can detect when coupled with 1D orientations (horizontal and vertical) and haptic vibrations will give a clue on how to improve the user experience.

1.6 Overview of the study

The Thesis is composed of seven chapters which are introduction, related work, theoretical framework, system development process, and the implementation of developed system, conclusion and the recommendation.

Chapter 1 presents the global introduction of haptic feedback and the problem, the aims of the study, the significance of the study, the limitation of the study and the overview of the study.

Chapter 2 gives the background of the related research connected to the study

Chapter 3 describes the methods and materials used for the study

Chapter 4 presents the Hardware used for the experiment.

Chapter 5 presents the experiments that have been done

Chapter 6 presents the results of the experiment

Chapter 7 gives the conclusion

2. BACKGROUND

This chapter presents a set of connected works that will wrap up several parallel subjects on the project research. The following part will provide a broad overview of haptic feedback based on vibration and kinesthetic to support the ongoing thesis topic.

2.1 Related research on haptic feedback based on vibration

It's bizarre that our predominant, if not only, but connection to reality is also the tactile experience, which is infinitely less valuable to men than sight [9]. This assertion of Vladimir reminds us how important the sense of touch is significant on how we perceive and experience the world. In other words, haptic feedback, which is, in its most basic definition, everything related to the sense of touch, is an important feature that must be investigated in order to better our daily interactions with our surroundings. Haptic applications include gaming, robotics, medicine, and electronics, among others.

Haptics emerged immediately following World War II. It was established to address the issue of an abundance of radioactive elements. Being in direct contact with radioactive materials was not an option due to the hazard that represented radiation. Thanks to the kinetic linkage, a mechanical apparatus constructed at that time with the same delicateness and dexterity of human movement, addressed the issue. Soon after, pressure and friction feedback emerged to improve the human experience [10].

A user can get haptic feedback in two ways. The first way is through kinesthetic feedback, which is concerned with the determination of positioning, motion, and forces [11], and tactile feedback, which is linked with the feeling of vibration, pressure, and shear force through the skin [12].The kinesthetic and tactile aspect will be examined by analyzing the finger features and functions on one hand and the actuator utilized to carry out the experiment on the other hand.

The anatomy of the finger is unique. It is made up of mechanoreceptors, which are responsible for tactile feedback. Meissner's corpuscles, Pacinian corpuscles, Merkel's

disks, and Ruffini's corpuscles are the four basic types of mechanoreceptors that convey information regarding pressure, touch, vibration, and cutaneous tension.

The hairless skin is where we found mainly the Meissner’s corpuscles. These corpuscles are especially effective in transmitting signals about the relatively low-frequency vibrations produced when textured items are dragged across the skin [13].

The Pacinian’s corpuscles have an onion-like capsule. The capsule functions as a filtration system, permitting only brief irregularities at high frequencies to trigger nerve terminals in this scenario. Pacinian corpuscles adjust quicker and have a lower response threshold than Meissner's corpuscles. These properties suggest that Pacinian corpuscles perceive small surface textures or other mobile cues that induce high-frequency skin vibrations. They contribute to ten to fifteen percent of the cutaneous sensors in the hand [13]

Merkel's disk receptors account for twenty-five of hand receptors and are mostly found in the fingertip. The nerve terminal is assumed to be regulated since Merkel's disk is associated with slowly adapting nerves. As a result of these criteria, Merkel's disk is responsible for the static distinction of forms, edges, and rough textures. The cutaneous stretching caused by finger or limb motions is particularly sensitive to Ruffini's corpuscles. They make up roughly twenty percent of the sensors in the human hand and do not produce any tactile sensation when electrically triggered [13].They are not well understood yet.

Below is the table that summarizes the main mechanoreceptors.

Table 1: Mechanoreceptors features [13]

Category of receptors	**Location**	**Role**	**Adaptation rate**	**Activation threshold**
Meissner's corpuscles	Glabrous skin is the most common type.	Dynamic pressure Touch	Fast	Low
Pacinian corpuscles	Viscera, interosseous membranes, and subcutaneous tissue	Dynamic vibration High pressure	Fast	Low
Merkel's disks	Found in the entire skin and hair follicles	Static pressure Touch	Slow	Low
Ruffini's corpuscles	Found in the entire skin	Skin stretching	Slow	Low

When investigating an object, both kinesthetic and tactile information are needed. To conceptually offer significant force and movement to an area of the body, a haptic device providing kinesthetic response requires a large-scale mechanical architecture on the one hand, whereas due to the great sensibility of sensory receptors in human tissue, haptic devices for tactile information often have a smaller form aspect [12].

A large proportion of contemporary haptic devices that provide kinesthetic feedback is based on reality [14]. They have various advantages, including as high forces and torques, multiple degrees of freedom, and large frequency response. When compared to tactile haptic devices that just stimulate the skin, these capabilities enable such devices to produce more realistic haptic renderings [15] . Sajid Nisar et al. investigated on how kinesthetic feedback apply to a user's fingertip could impact a user's haptic perception. They created a wearable haptic device that could provide haptic inputs along the finger axis line as well as the finger's flexion-extension motion orientation. The methodologies used were psychophysical experiments for each of the two feedback directions to measure the users' haptic perception for each grounding area (behind the hand, proximal phalanx, and middle phalanx of the index finger). According to their findings, placing the wearable device at the back of the hand had a lower impact on experiencing kinesthetic feedback in both directions. The proximal and middle phalanx, on the other hand, had the highest accuracy in perceiving kinesthetic feedback in the flexion-extension direction. Therefore, the mounting's area has a huge impact on the user's haptic experience. The drawbacks that were emphasized is that while the user perception accuracy increases, the comfort level decreases as the mounting are going closer towards the fingertip from the back.

There are several types of actuators that are responsible for tactile feedback. Among them, we can enumerate the linear resonant actuator and the pneumatic actuation.

Pneumatic actuation employs regulated air pressure to move, expand, and inflate or deflate the mechanical parts in the end-effector that generate haptic signals [12]. The authors attempted to give many types of haptic feedback such as static pressure, high-frequency vibration, and impact through a single end-effector (ring actuator) in enhancing the

AR/VR feedback environment. They successfully demonstrated that pneumatic actuation could produce a variety of different feedback with outstanding results.

Federica Barontini et al. [16] aimed to send information about a prosthetic hand's grasp force via soft pneumatic force feedback. The device was made of two silicone chambers that were utilized to apply pressure to certain stump locations. During the gripping phase, they were able to adjust, amplify, and send a pressure stimulation to the robotic hand. The limitations of their project were that they did not conduct their experiment on genuine prosthesis users and had to further examine the effect of muscle changes in dimension with contraction.

Adding numerous chambers operated by numerous air pumps could be one approach. Furthermore, coupling air pressure with particle jamming [17] to represent output feedback in diverse orientations could also be an option to address it.

LRA is one of the most actuators used today. Due to its affordability and ubiquity, it is one of the most preferred actuators.

Stephens-Fripp et al. [18] developed an upper limb prosthetic sensory device that was aiming to enhance user feedback. They used nine LRA motors. They based their studies on J.H Kirman et al. [19], which emphasized the fact that the higher the actuators, the higher the accuracy of feedback. Furthermore, Benjamin Stephens-Fripp et al. [18] found that the number of actuators reduces considerably by what they called ISOI (the amount of time it takes for overlapping tactile stimulation to produce perceptual motion). During their experiment, an average recognition rate of 72 percent was achieved when six distinct grips and movement orientations were combined.

Hongbin Liu et al. [20] suggested an enhanced fingertip inherent contact detecting model based on 6-axis force/torque readings by incorporating the mechanical concept of a flexible skin. The contact-sensing method was found to achieve high speed and accuracy throughout the entire surface that was subjected to the magnitude and direction of friction forces, normal forces, and local torques. There were several drawbacks associated with the system. The light weight object must be moved in order to be explored. This was due

to the high value of the frictional force. Furthermore, the finger was trapped locally when following the contour control searches for a local minimum.

The surface following contour approach should be enriched in order to explore an unidentified object using hand manipulation in a stable and efficient manner.

The perception of haptic feedback is greatly influenced by vibrations. Understanding its vibration patterns and identifying the nature of the objects can help improve user experiences. Some researchers observed that the mechanoreceptors on the finger play a significant part in perceiving vibration frequencies, edge differentiation, texture, and pressure when in touch with an object or substance. However, Ruffini's corpuscles, which detect low-frequency vibration and pressure, are poorly understood. It is critical to investigate this since some devices, such as the one in our study, require low vibration and/or pressure for specific functions.

Furthermore, the mounting area is said to be critical to the user's haptic experience. It was observed that proximal and middle phalanx have the highest precision in perceiving kinesthetic feedback. This assertion is very close to the truth because it was observed during our experiments that even though one actuator was used; the vibration plays a huge role in detecting the shapes since the ring was mounted on the interphalangeal joint (between the proximal and middle phalanx). It also was comfortable for forty to one hour long. After that, the fingers of the participants were numb. In addition, concerning the actuators, it was discovered that pneumatic actuation can give different types of haptic feedback. The complexity of this method relies on the fact that each actuator needs two valves, the system size and weight is relatively cumbersome, and the air tank needs to be filled up every twenty minutes continuously. Furthermore, it was just possible of producing one form of feedback at once. Enhancing the rendering algorithm of the pneumatic system designed could be a way to simultaneously generating numerous sorts of haptic feedback Thanks to the LRA and the DRV260LEVM- MD driver that were used during our trials. The LRA is just powered up by voltage and can last as long as it is wished. As far as the driver is concerned, it was able to produce different tactile sensations. The use of multiple LRA increased the haptic accuracy feedback and reduce the time to

identify perceptual movement. The authors were using nine LRA motors. The latter will produce lot of noises and reduce the comfortability.

3. METHODS AND MATERIALS

This chapter presents all the elements that have been used to make the experiment. Thus, the hardware and software that have been utilized as the main components to make the experiment will be described and we also will emphasize its advantages as well as its disadvantages.

3.1 Mechanical Design

3.1.1 Arduino

The expansion of technology has hugely contributed to the development of new smart devices designed to make the human experience easier in a multitude of ways. One way could be the use of a device that aims to control, sends signals to other devices, apparatus, and even humans. An Arduino plays that role perfectly. Arduino is an open-source electronics system that provides simple hardware and software to make it simple to use [21] .

The Arduino's creation has sparked a great deal of debate. Hernando Barragán who began the project by designing wiring as part of his master's thesis at IDII in Italy in 2003, claims that he does not receive enough credit and that his wiring source code was duplicated by two students, Massimo Banzi and Mellis, in 2005; both students at the same institute after he left [22]. Furthermore, after experimenting with a variety of microcontrollers to connect to his Wiring, he claimed that he settled on the Atmel Atmega128 and purchased an Atmel STK500 evaluation board with a particular port for the ATmega128. He then went to BDMICRO and purchased a MAVRIC board with the ATmega128 soldered on it. His project's language was defined as followed

- pinMode ()
- pinRead () which is now digitalRead ()
- pinWrite () which is now digitalWrite ()

- analogRead ()
- analogWrite ()
- delay ()

And many more.

Twenty-five Wiring printed circuit boards were purchased by IDII in March 2004 in order to see the final prototype.

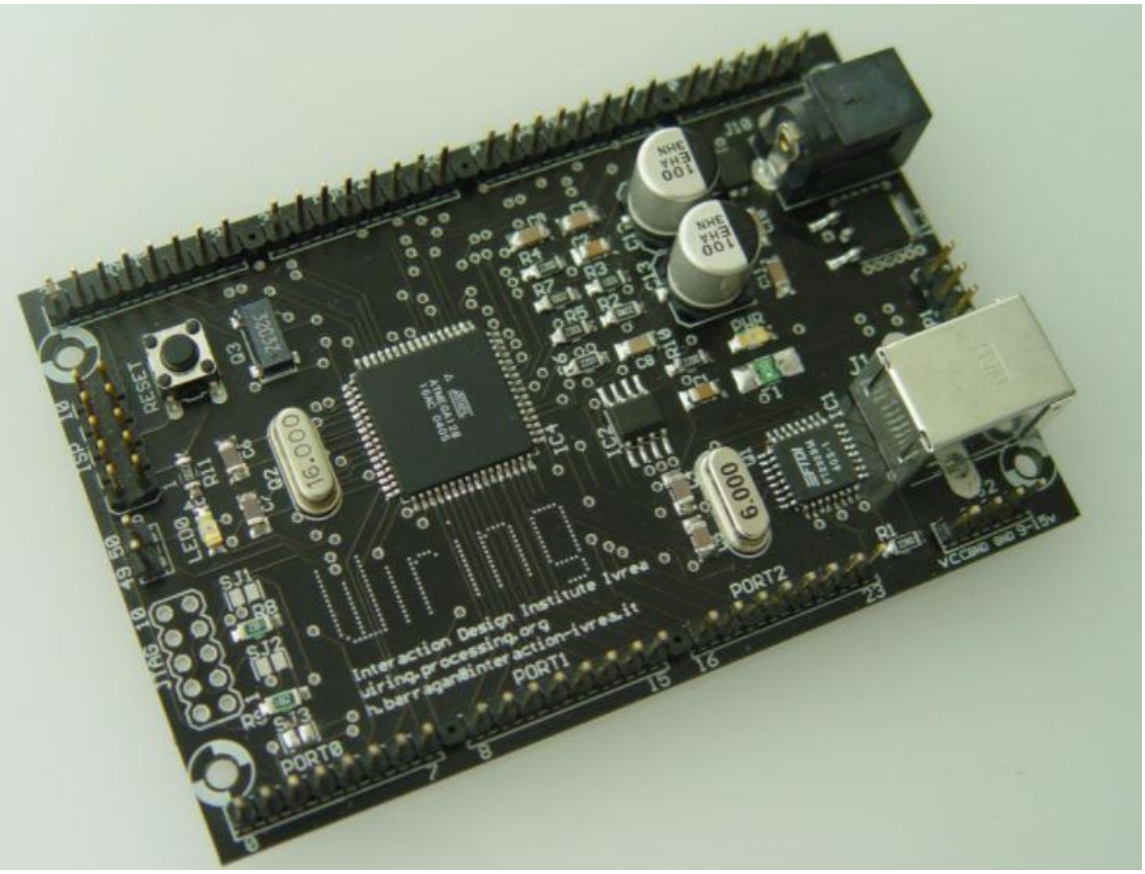

Figure1: First Wiring Hardware [22].

According to Hernando Barragán, after cloning his Wiring source code, Massimo Banzi, David Mellis, and David Cuartielle just added support to the Atmega128 microcontroller to the Wiring hardware and then called it Arduino as a new project.

Table 2: Classic Arduino Board [21]

	Arduino Due	**Arduino Leonardo**	**Arduino Mega 2560**	**Arduino Uno R3**	**Arduino Micro**
USB connector + Connectivity	Micro USB	Micro USB- B	USB-B	USB-B	Micro USB
Memory	96KB SRAM/ 512KB flash	2.5kb SRAM/ 32KB Flash/ 1KB EEPROM	8KB SRAM/ 256KB Flash/ 4KB EEPROM	2kb SRAM/ 32KB Flash/ 1KB EEPROM	2.5KB SRAM/ 32KB Flash/ 1KB EEPROM
Language of programming	Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE
Microcontroller	AT91SAM3X8E	ATmega32u4	ATmega2560	ATmega328P	ATmega32u4
Voltage and Dc	Voltage supplied at 7-12V; Operating at 3.3V 9mA per I/O “group 1” Pin 3mA per I/O “group 2” Pin	Voltage supplied at 7-12V; Operating at 5V 10 mA per I/O pin	Voltage supplied at 7-12V; Operating at 5V 20 mA per I/O pin	Voltage supplied at 7-12V; Operating at 5V 20 mA per I/O pin	Voltage supplied at 7-12V; Operating at 5V 10 mA per I/O pin
Vehicle of Communication	CAN / I2C / UART / SPI	UART / I2C /SPI	UART / I2C /SPI	UART / I2C /SPI	UART / I2C /SPI
Size (mm)	101.5 x 53.3	68.6 X 53.3	101.5 x 53.3	68.6 X 53.4	48 X 18

3.1.3 Why choosing Arduino UNO?

The Arduino that has been chosen is the Arduino Uno. I chose the Arduino Uno because it is simple to program, inexpensive, lot of documentation and tutorials about it. Furthermore, the Arduino Uno has an I2C communication that will help communicate with the motor drivers and has a decent microcontroller.

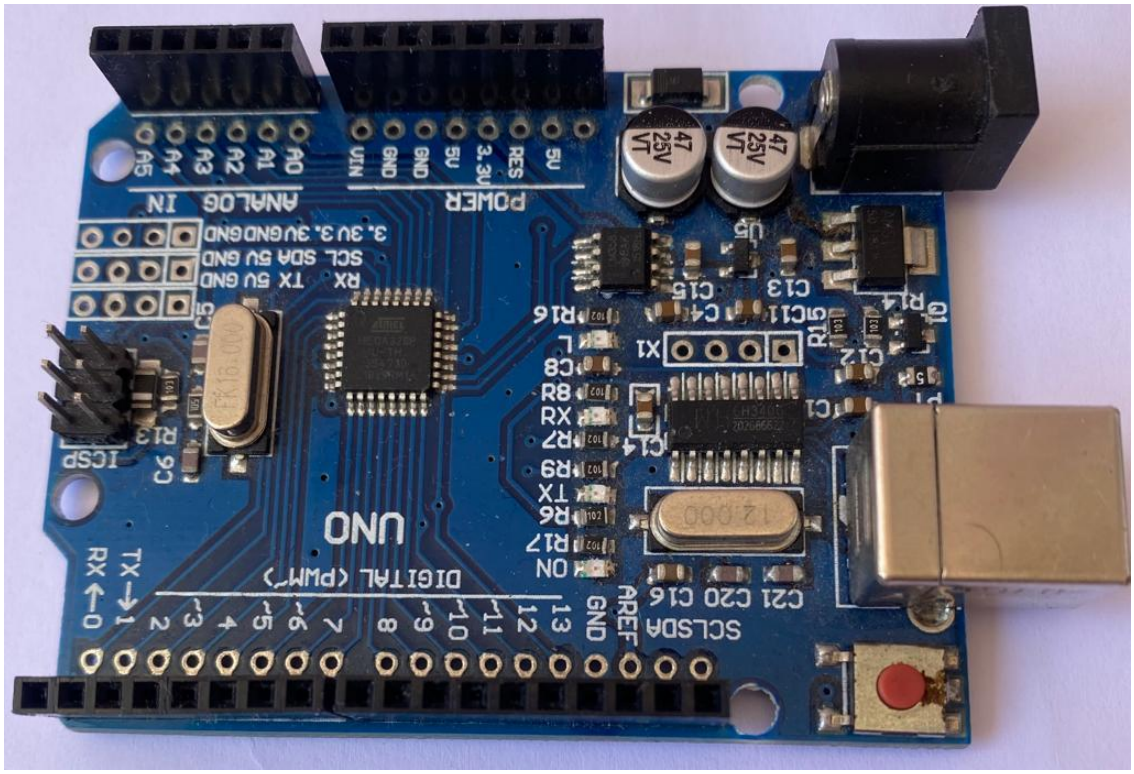

Figure 2: An Arduino Uno module

3.2 Haptic Motor drivers

The two most popular haptic motor drivers are drv2605 and drv2605L. They are quite similar (see Table 3.6) but the Drv2605L has an edge over the drv2605. If an invalid back-EMF is detected, the Drv2605 L can instantly go to open loop mode (Drv2605L documentation, 2018).

Figure 3: Drv2605L haptic motor driver

The main choice of this type of haptic drivers were due to its I2C communication.

3.2.1 Why I2C is a perfect fit?

In order to answer this question, let's compare and analyze the other available options.

3.2.1.1 Serial UART ports

Clock data is not transmitted between devices through serial connections. As a result, they should match on a data rate before proceeding. The two communicating devices should have clocks that are almost at the same pace, or else the data will be distorted. Furthermore, every cycle of information has at least one beginning and stop bit, which implies that for each 8 bits of information conveyed, 10 pieces of information transmission are needed which reduces the transmission rate [23].

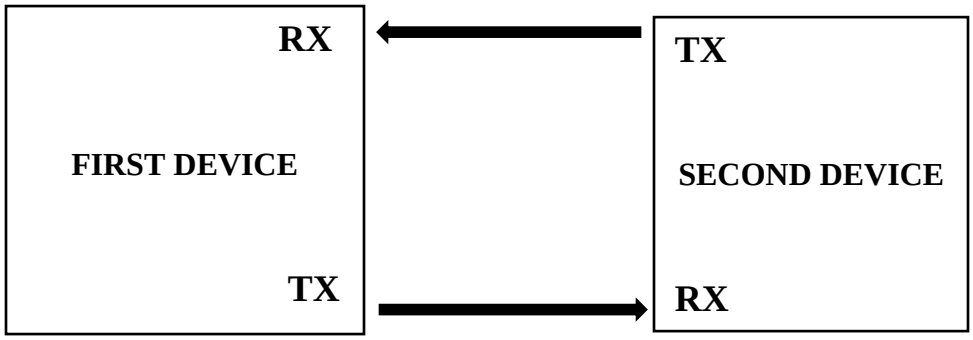

Figure 4: UART Communication diagram *[23]*

3.2.1.2 Serial Peripheral Interface

The number of pins required for SPI is the most evident disadvantage. Four lines are required to connect a single controller to a single peripheral via an SPI bus. Every extra peripheral device necessitates one more chip select I/O pin on the controller. It can easily be concluded that this will lead to a rapid increase of pins connections that could lead to confusion.

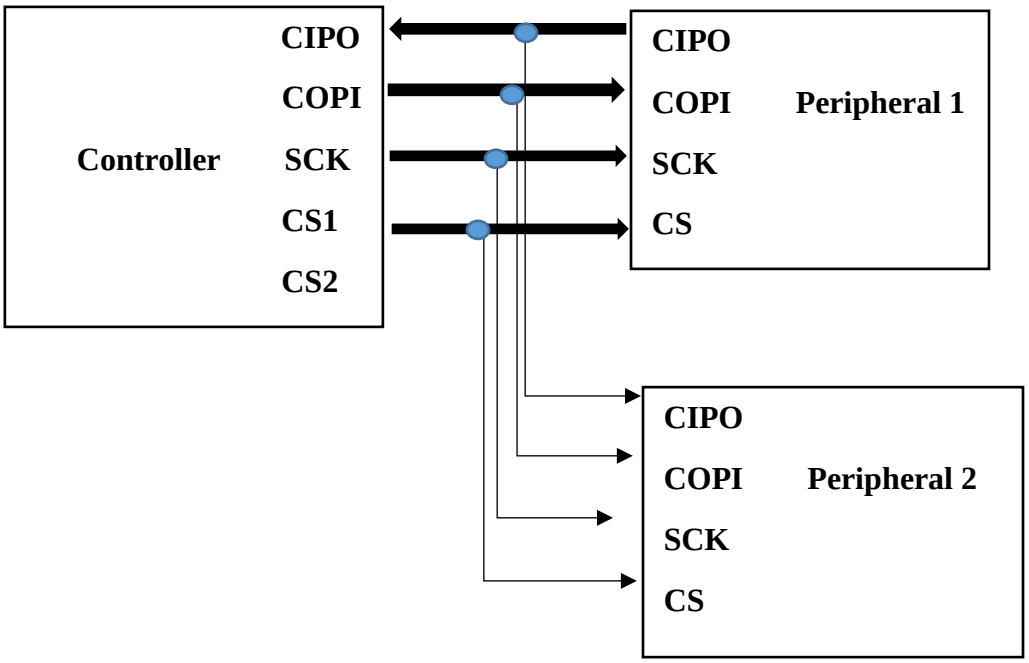

Figure 5: SPI communication diagram [23]

3.2.1.3 Inter-Integrated Circuit (I2C)

Similarly, to UART port, I2C only requires two main wires. However, those two wires can accommodate just over 1000 peripheral devices. In addition, every I2C bus provides 2 signals: SDA and SCL.

SDA is the wire that is responsible to send and receive data from the master and slave while SCL is the wire that is responsible for the clock signal.

As opposed to UART or SPI associations, I2C transport drivers are "open channel," and that implies they can pull the related signal line low yet not drive it high. Accordingly, there can be no transport conflict when one device attempts to drive the line high while one more attempt to pull it low, eliminating the chance of driver harm or excessive power dissemination in the framework. Each sign line is furnished with a pull-up resistor to reestablish the sign to high when no device is stating it low [23] .

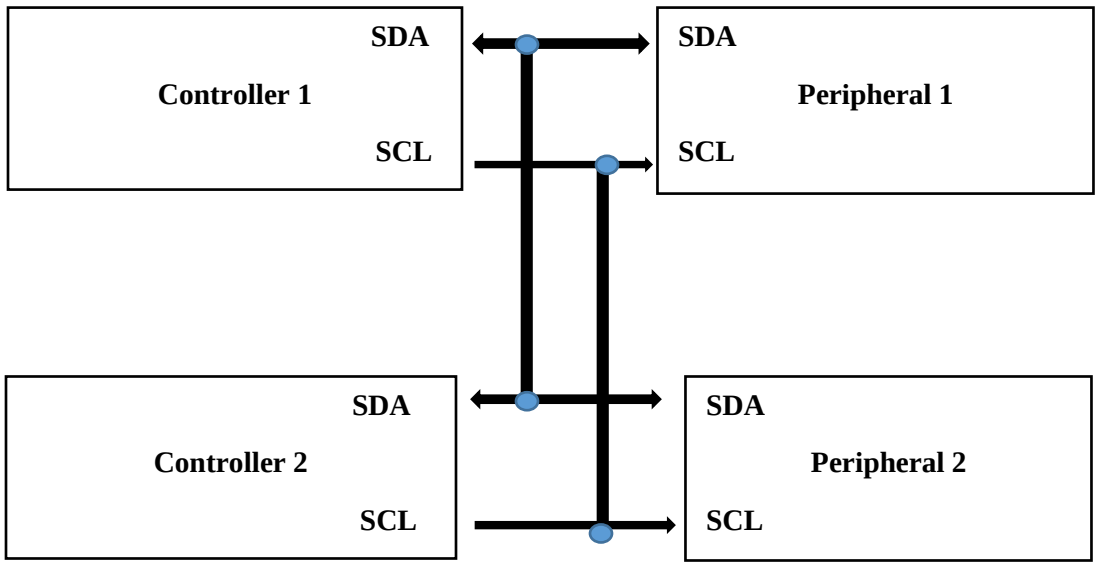

Figure 6: I2C Communication diagram [23]

Table 3: DRV2605 VS DRV2605L [24]

	Drv2605	**Drv2605L**
Flexible Haptic/ Vibra driver	LRA / ERM	LRA/ERM
I2C Controlled Digital Playback Engine	Real-Time Playback (RTP).	Waveform Sequencer and Trigger. Real-Time Playback (RTP). I2C Dual-Mode Drive (Open and closed-loop).
Smart Loop Architecture	Automatic Overdrive/ Braking (LRA/ERM). Automatic Resonance Tracking (LRA). Automatic Actuator Diagnostic (LRA/ERM). Automatic Level Calibration (LRA/ERM).	Automatic Overdrive/ Braking (LRA/ERM). Automatic Resonance Tracking (LRA). Automatic Actuator Diagnostic (LRA/ERM). Automatic Level Calibration (LRA/ERM). Wide Support for Actuator Models.
Licence Immersion Touchsense 2200 features	Integrated Immersion Effect Library. Audio-to-Vibe	Integrated Immersion Effect Library. Audio-to-Vibe
Applications	Mobiles phones and tablets. Watches and wearables devices. Touch- enables devices. Human-machines interfaces	Mobiles phones and tablets. Watches and wearables devices. Touch- enables devices. Human-machines interfaces. Electronic Point of sales.
Hardware input	N/A	Hardware Trigger Input

3.2.1.4 DRV2605LEVM-MD Evaluation Module

The DRV2605LEVM-MD evaluation module is a platform for the DRV2605LDGS to be evaluated. The setup comes with eight DRV2605L devices, an MSP430F5510 microcontroller (MCU), and terminal output allows for up to eight LRAs or ERMs, DRV2605L-integrated waveforms, and capacitive buttons to illustrate the DRV2605L's capabilities. In addition, The TCA9548A I2C switch is integrated into the system to regulate which I2C connections of the available eight DRV2605L drivers are joined to the master input I2C bus. The switch enables users to connect any arrangement of networks to the master input I2C bus [25].

The DRV2506LEVM-MD module was a perfect fit for my project because it allowed me to run multiple motors simultaneously and besides that each motor could be run independently. Furthermore, the fact that all the drivers sit in on one framework makes it less cumbersome.

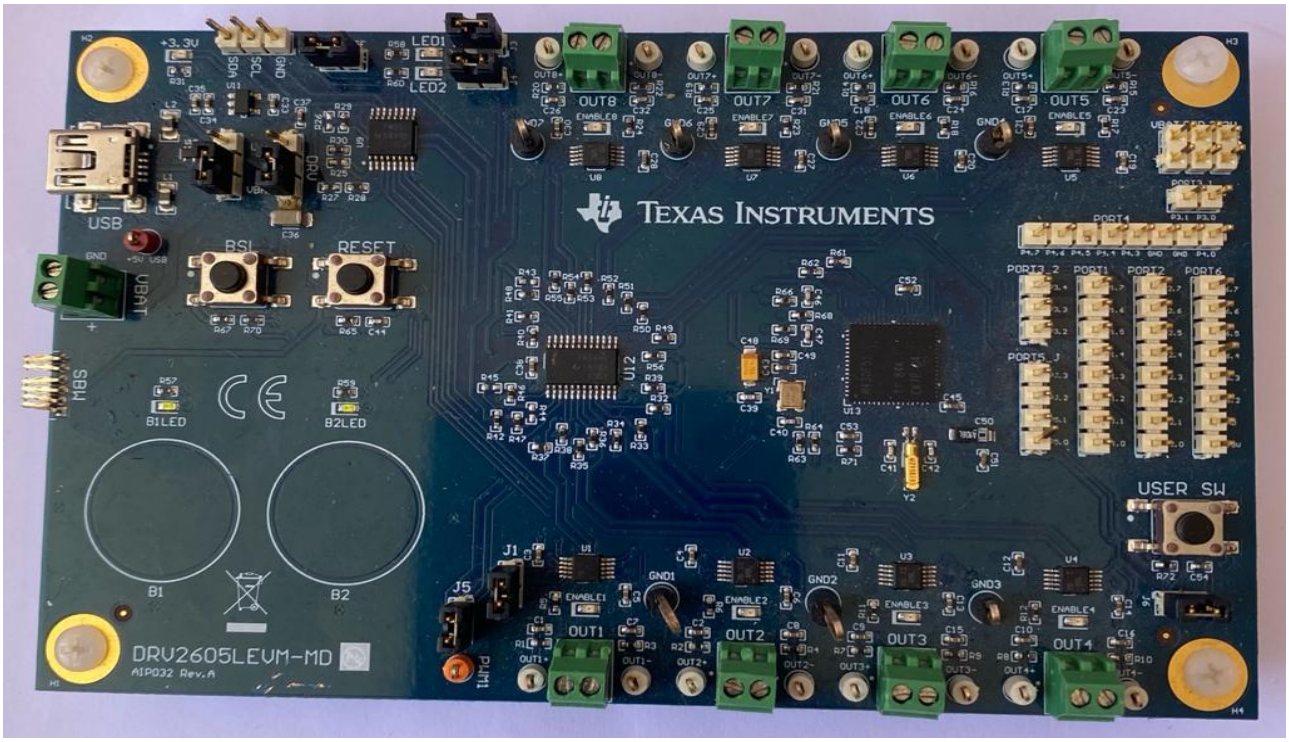

Figure 7: The DRV2506LEVM-MD Module Driver

In order to use the DRV2605LEV-Md module, some changes were made in order to connect it to the Arduino Uno. R61 and R62 resistors are 0 ohm resistors which connects the SDA and SCL of the MSP430F5510IRGC microcontroller to TCA9548APW and TCA9554PWR ICs. To connect the Arduino to the DRV2605LEVM-MD, I had to remove the R61 and R62 resistors to disconnect the MSP430F5510IRGC microcontroller and be able to control the DRV2605LDGS motor drivers using Arduino.

3.3 Haptic Motor (LRA)

There are several types of haptic motors. Among them, one of most popular is LRA was the ideal choice for the system due to its specificities, see table 9.

The LRA model that was used was an NFP-ELV0832B.

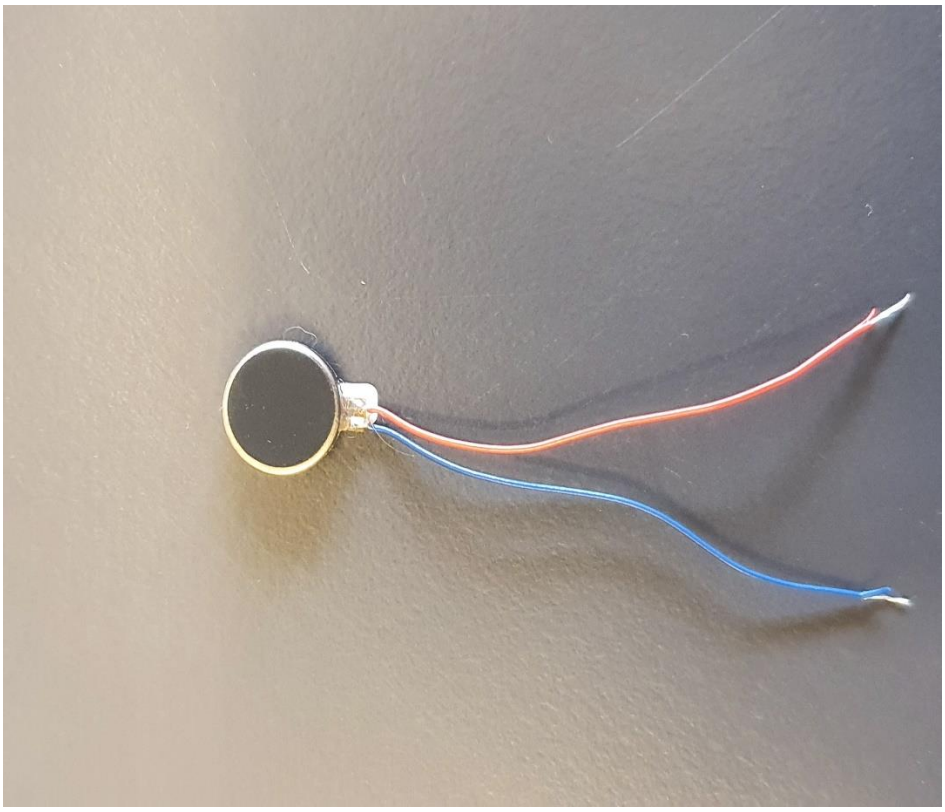

Figure 8: Linear Resonant Actuator

Table 4: LRA Specifications [26]

Item	Characteristics
Diameter	8 +/- 0.1 mm
Height	3.25 +/- 0.1mm
Vibration's acceleration	1.2 to 1.5 Grms
Frequency's operation	235 +/- 5 Hertz
Rising's time	50 milliseconds at peak
Falling's time	80 milliseconds at peak
Rated Voltage	1.8 Vrms
Noise due to the mechanics	50 dB(A) at peak
Weight	1 Gram
Current rating	90 mArms at peak

3.4 Arduino Uno shield

The shield is a board that is mounted over the Arduino board in order to increase the functionalities of the project.

The mean reason why I used the shield was because of the wiring between the Arduino Uno and the DRV2605LEVM-MD driver. I did not want to use a breadboard. In the lab, the Arduino Shield was conceived and built. The cabling was designed to communicate through I2C while simultaneously powering the DRV2605LEVM-MD via the Arduino.

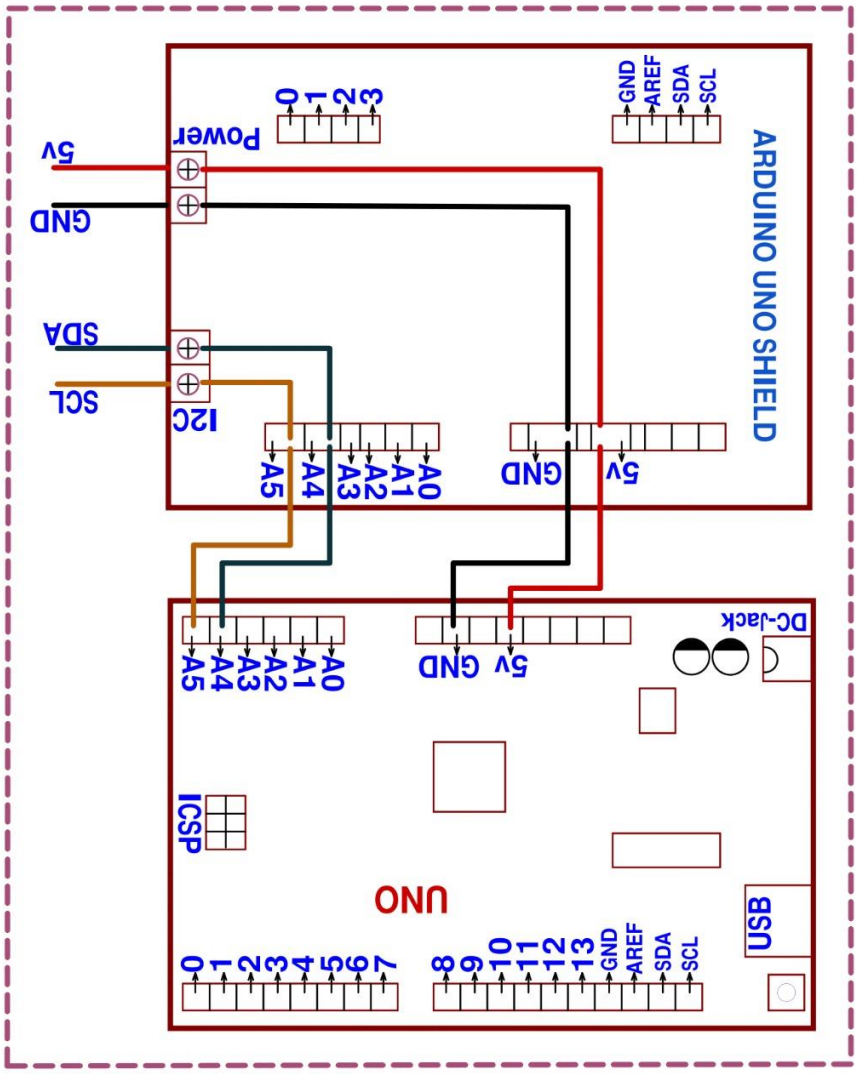

Figure 9: Arduino shield –Arduino Uno connection schematic diagram

3.5 Jump wires

A jump wires are electrical wires that do not require soldering to join two points, equipment, or circuits (Tatsuo Katayama, 2005).

Each end of the wire is either a female-female junction, male-male junction or a male-female junction. I used the four jump wires in my project in order to connect the Arduino Uno and the DRV2605LEV-MD module.

Initially, the connection was made to power the DRV2605LEVM-MD module. The first electrical wire (male-male wire) was used to connect the ground from the Arduino Uno to the ground of DRV2605LEVM-MD module. The second one (male-male wire) was used to connect the 5V from the Arduino Uno to the "+" of the driver. Similarly, the connection was made for the I2C communication between the Arduino Uno and the driver module. The pin A5 (SCL) was connected to the SCL output of the driver and finally the A4 (SDA) pin was connected to the SDA output of the driver (Both were male-female wires).

3.6 Ring

The ring was designed using SolidWorks 2017. I was aiming to design a flexible ring that will be able to fit almost everyone. The ring will have four holes with the LRA dimensions. Each hole will be separated at an angle of 90 degrees. Two prototypes have been made. The first was done with a relatively strong material that was less comfortable and painful in the long run. The final ring is 3D-printed using a custom-made 3D-Printer with flexible NinjaFlex TPU filament (85A) [27]

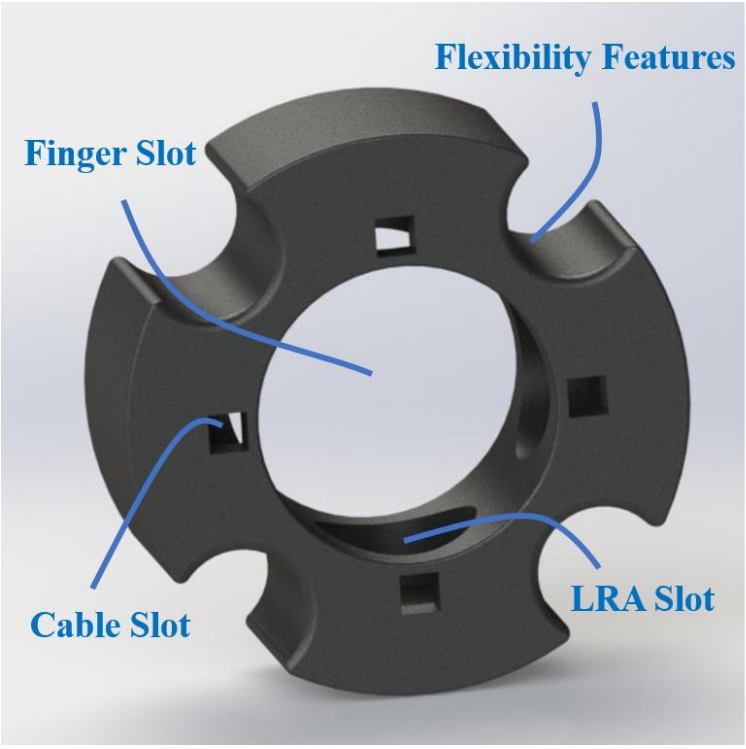

Figure 10: CAD ring design

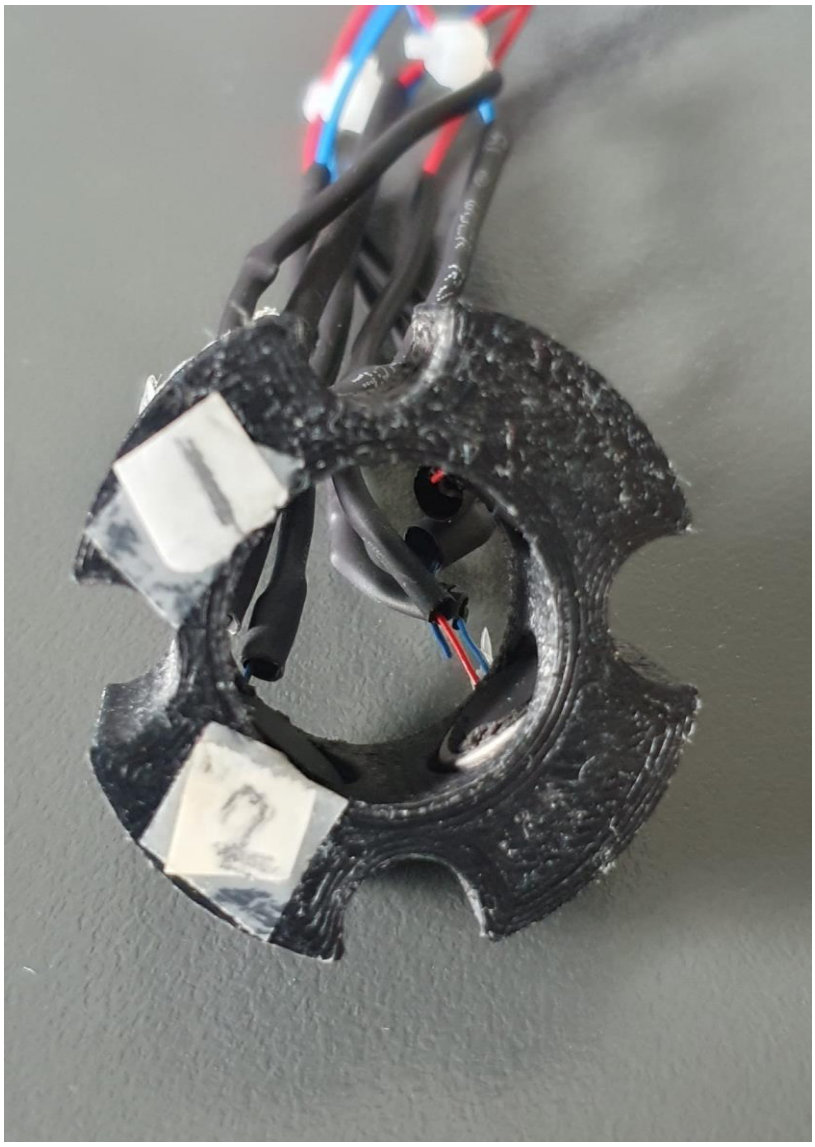

Figure 11: Final ring after 3D printing

4 SOFTWARE INTERFACE

4.2.1 MATLAB GUI

Graphical user interfaces (GUIs) are visual interactive interfaces between the user and the software that eliminates the need to learn another programming language or commands to run the apps [28].

In MATLAB, there are three approaches to developing a GUI.

- Create an easy application from a script
- Develop an interactive application
- Build a programmatic application

I chose to create an interactive app because it allows me to build the user interface using a drag-and-drop environment. This can be done through either App Designer or GUIDE.

App Designer features a more user-friendly interface than GUIDE and is more sophisticated (see Table 3.6). When attempting to construct a GUI, MATLAB recommends this software. GUIDE will disappear in the years ahead. I chose to utilize App Designer, and everything was going swimmingly until I discovered that App Designer does not provide smooth feedback when used with a touchscreen. To obtain any feedback, I had to tap on the screen. The touchscreen feedback is at the heart of my project. I am hopeful that MATLAB will be able to resolve this issue in future editions. As a result, I opted to use GUIDE.

Table 5: GUIDE vs App Designer [29]

	GUIDE	APP DESIGNER
Code debugger	YES	YES
Code folding	YES	YES
Global component rename	NO	YES
Utility Functions via Custom Application Methods	NO	YES
Custom Components	NO	YES
Printing	YES	YES
ActiveX Control	YES	NO
Share as MATLAB App Install File	YES	YES
Share as MATLAB Web Application	NO	YES
2D & 3D Plotting	YES	YES

Figure 11: Screenshot of the GUI (GUIDE) when the finger was on the band area

4.3 Arduino and MATLAB communication issues and solution

Before illustrating the issue, the flow chart below is giving an idea of what is expected in the experiment.

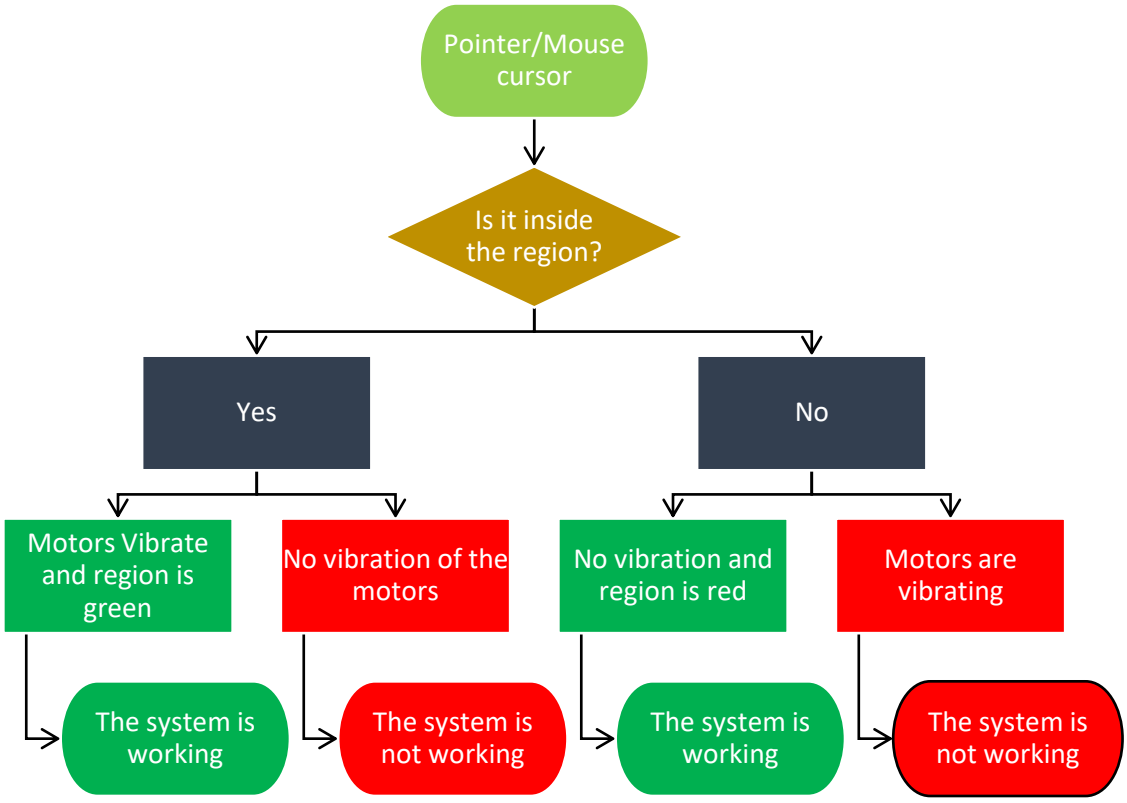

Figure 12: Diagram of pointer behavior output expectations

4.3.1 Arduino Code problem

A major issue arose during the communication between Arduino and MATLAB GUI. The problem was with the timing when the pointer was traveling in and out of the desired location at a high rate. Even though the motors were outside the required zone, the motors continued to vibrate.

Technically, the code was developed on the Arduino utilizing the Serial library to receive MATLAB data. “Serial. Available” method was polled in the program's main loop, and data was read using “Serial.readBytesUntil” (character, buffer, length). The Arduino Serial library was slowed due to its polling approach for getting data over serial and its small buffer capacity. There was a high possibility of missing the incoming data since the code may be preoccupied with other tasks such as turning on the motor or changing the speed of the motors. The incoming data could not be read at that time because the code was working on other tasks rather than reading the incoming data.

3.3.2 MATLAB Issue

The Arduino code was tested separately in the first phase using its Serial Terminal and afterward with Docklight (COM Port Software). On every Serial command sent, the Haptic Motors responded as predicted. I tested the Arduino code with high-speed data using the Docklight program.

The commands were no longer working as planned or tested after connecting the Arduino COM port to MATLAB. I came up with the idea of watching the MATLAB commands being sent using Virtual COM Port Software. Following testing, it was discovered that MATLAB sends an extra two characters with each instruction. The two characters were MATLAB Terminator Characters, namely CR and LF (Line Feed).

3.3.3 Solution

As a result, I used the Interrupt-based USART (Universal Synchronous and Asynchronous Receiver-Transmitter) library for Arduino to fix the problem. The interrupt handler ISR (USART RX_vect) can receive all character data supplied through the USART.

There is no risk of missing the data because the ISR can handle all incoming data on the spot. The code will then proceed to the ISR to handle the incoming data before returning to the normal code working flow. Following that observation, I modified the code on the Arduino side to handle these two Terminator Characters as well before completing the MATLAB received string. After the adjustments, the code works well and behaves as planned.

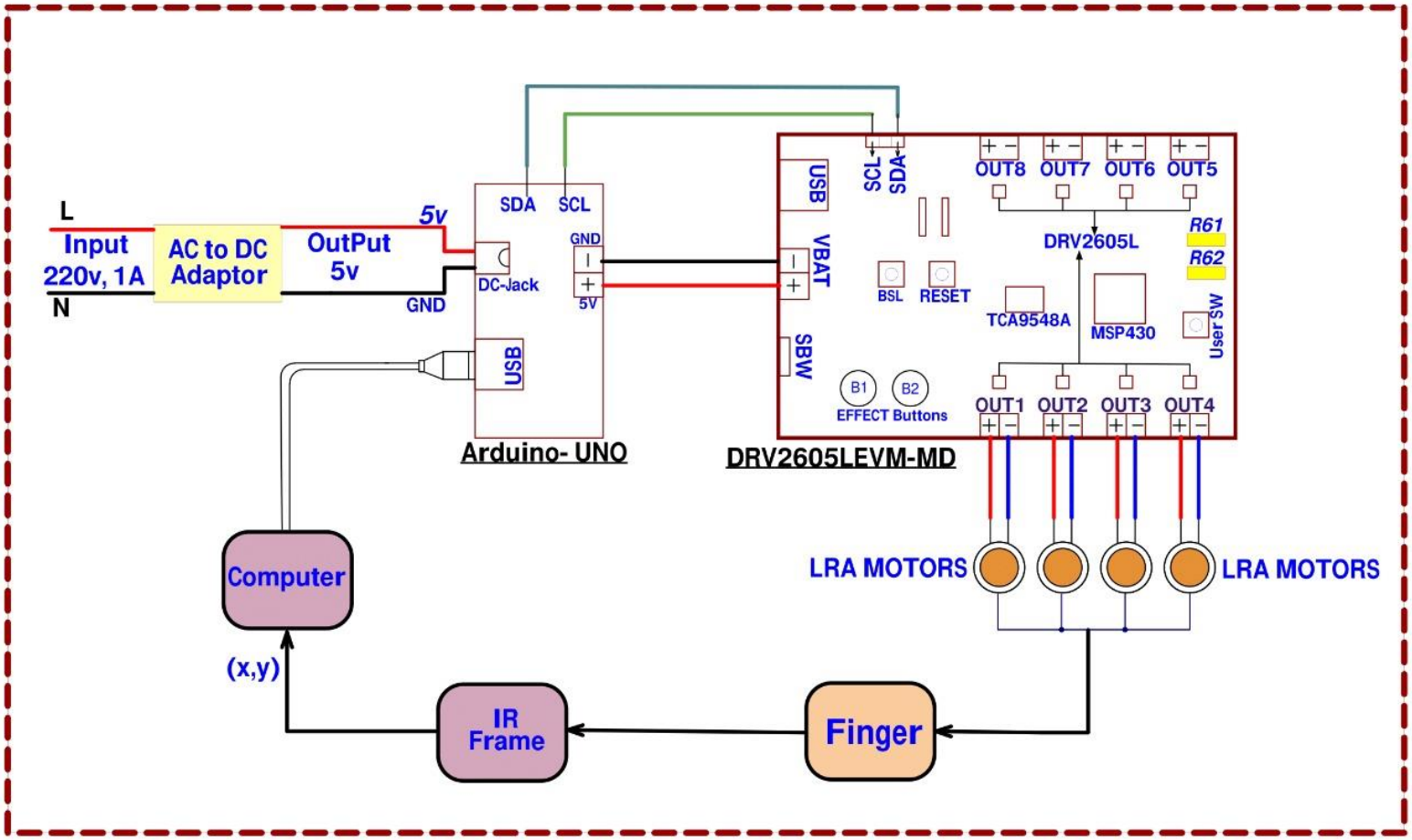

Figure 13: Haptic system Diagram, R61 and R62 are 0 ohm resistors and were removed

5. EXPERIMENTS

To interact with the participants, a 21-inch surface touchscreen (HP P223) was used. The participants were seated in front of the screen. The MATLAB software was used in order to produce the GUI (GUIDE) responsible to detect and record the user finger's position, time, and vibration experiences. Concerning the hardware, aside from the screen, a ring with four embedded haptic motors, a driver module (DRV26005LEVM-MD), an Arduino UNO, and an AC to DC convertor were also used. Figure 14 illustrates the setup that we used for our experiments.

We looked at the effect of vibrations with various amplitudes to determine the line thickness threshold required for participants to perceive and identify the intended lines of research. We also attempted to determine how much time was spent in order to find each line.

5.1 Participants

Nine people took part in the study (4 males and 5 females with an average age of 35, the youngest 21 and the oldest 42). They all rely on touchscreen devices daily. The participants were focused and serious during the investigation. Data from people who were not completely serious were not considered. Each participant was blindfolded, and a White noise background sound was played on the premises from YouTube so that the subject would not hear the vibrations of the motor. Furthermore, two elastic strings were used to attach the cablings on the forearm in order to avoid the cables to be all around the place as seen in figure 14.

5.2 Stimuli

The experimental stimuli were a straight line with two orientations, vertical or horizontal. The stimuli are positioned in the center of the screen. There were four different line thickness options (one of which was the blank case, which had a thickness of zero mm).

The line thickness is determined by trial and error based on preliminary experiments. The amplitude of the vibrations is controlled by 8-bit outputs.

Because of the driver, the minimum output value required to run an LRA motor is 154 corresponding to (154/255) x 1.8 Vrms = 1.09 Vrms and the maximum value is 255 corresponding to (255/255) x 1.8 Vrms = 1.8 Vrms where 1.8 represent the rated voltage of the LRA

We used 154 output values as Level 1 amplitude which is 1.09 Vrms and 179 as Level 2 amplitude which (179/255) x 1.8 Vrms = 1.26 Vrms in our experiments. Even though the haptic ring is made up of four LRA motors, only one of them is used to determine the threshold for human perception.

5.3 Procedure

A training session was held prior to the experiments to familiarize the participants with the haptic system. To avoid tiredness and fatigue, the experiments were divided into two sessions of the same length (10 minutes break). During the experiments, no questions were permitted, and no time constraints were imposed on the participants. Each experiment has 80 trials. Each trial event is a combination of three options: orientation (vertical, horizontal, Control - no stimulus), amplitude (Level 1 and Level 2), and thickness (2, 3, 4, and 0 mm - control). There could be a total of 16 trial events. Each trial event is repeated 5 times for a total of 80 trials for each subject. The subjects are instructed to use their index finger to explore the screen and report back to the experimenter whether they feel the stimuli. The experimenter used a mouse to record the subjects' responses to the experiment application via the GUI interface.

The sequence of events is chosen at random, and each participant receives the same random sequence of events.

After finishing the instructions, the participants immediately began the actual experiment. Their task was to explore the tactile lines on the screen with their finger on the touchscreen.

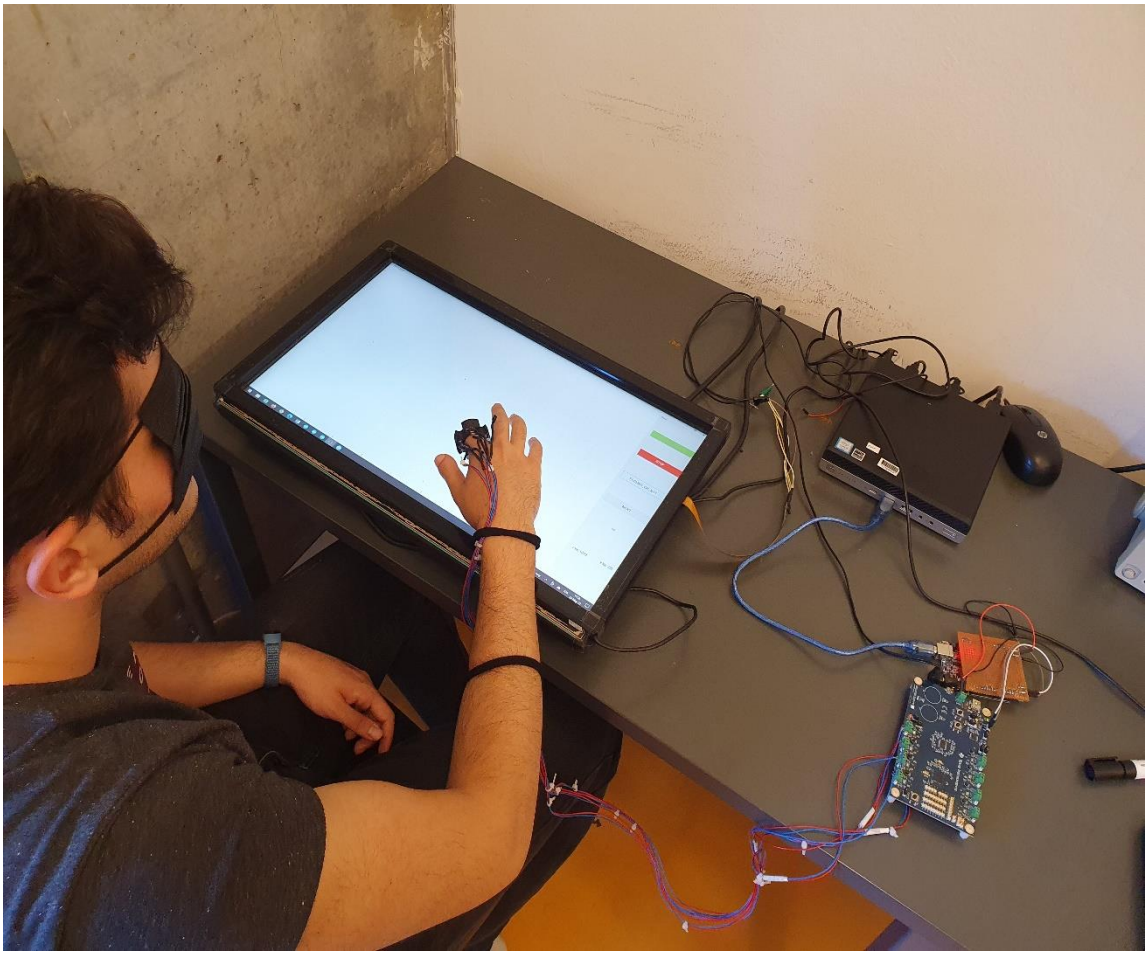

Figure 14: One subject participating in the experiment.

The independent variables on which the entire experiment was centered were thickness, amplitude, and orientation. Recognition time and recognition rate were used in order to evaluate the user performance.

In our case, the recognition rate is the proportion of accurately identified thickness coupled with vertical or horizontal orientation when experiencing diverse amplitudes, whereas the recognition time is the time it takes subjects to accurately identify the lines (varying thicknesses) when facing diverse amplitudes and orientations. They are respectively expressed in percentages and in seconds.

6. RESULTS AND DISCUSSION

In all the figures, the “blank” that represents zero thickness has the highest value. It was included for psychological reasons. The goal was to keep the participants from expecting vibrations all the time.

The case processing summary table translates the total number of cases N in the analysis. There are 720 cases in our experiment: 4 thicknesses x 2 orientations x 2 amplitudes x 5 repetitions x 9 participants = 720.

6.1 Recognition Rate and Time

To prevent controversy, a blank case (zero thickness, no vibration) during the experiments. The amplitude, orientation, and thickness were the independent variables while the time and the recognition rate were the dependent variables.

The figure 16 depicts the average recognition rate of the participants when subjected to different thicknesses (2 mm, 3 mm, 4mm, and blank). "TRUE" means that the participants got the answers right, while "FALSE" means the opposite. The blank thickness, which corresponds to no thickness, has the highest recognition rate, followed by 3mm, 4mm, and 2mm, in that order. In terms of recognition rate, the 3mm and 4mm were very close.

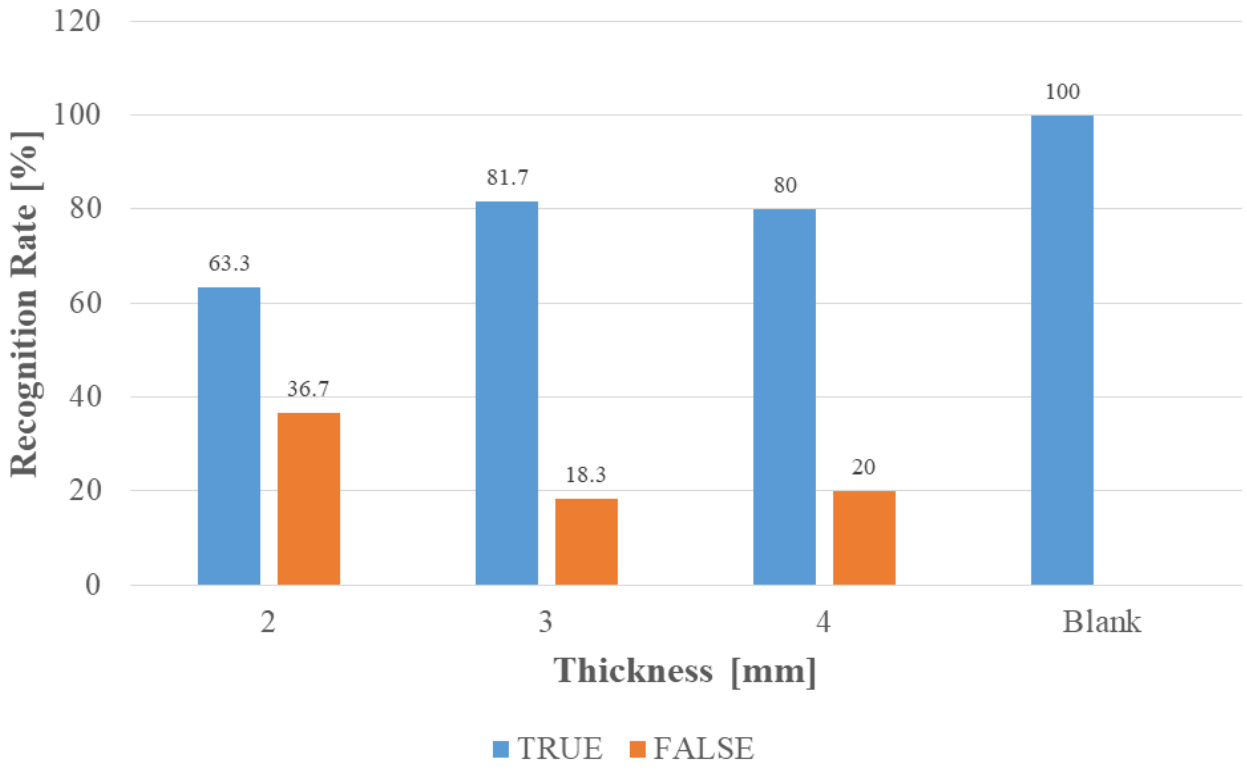

Figure 15: Average Recognition Rate of participants when subjected to different thicknesses; Blank = no thickness

Figure 17 shows the participants' average recognition rate when subjected to various orientations (Horizontal, Vertical, and blank). "TRUE" indicates that the participants correctly answered the questions, while "FALSE" indicates the opposite. Blank, vertical, and horizontal have the highest recognition rates.

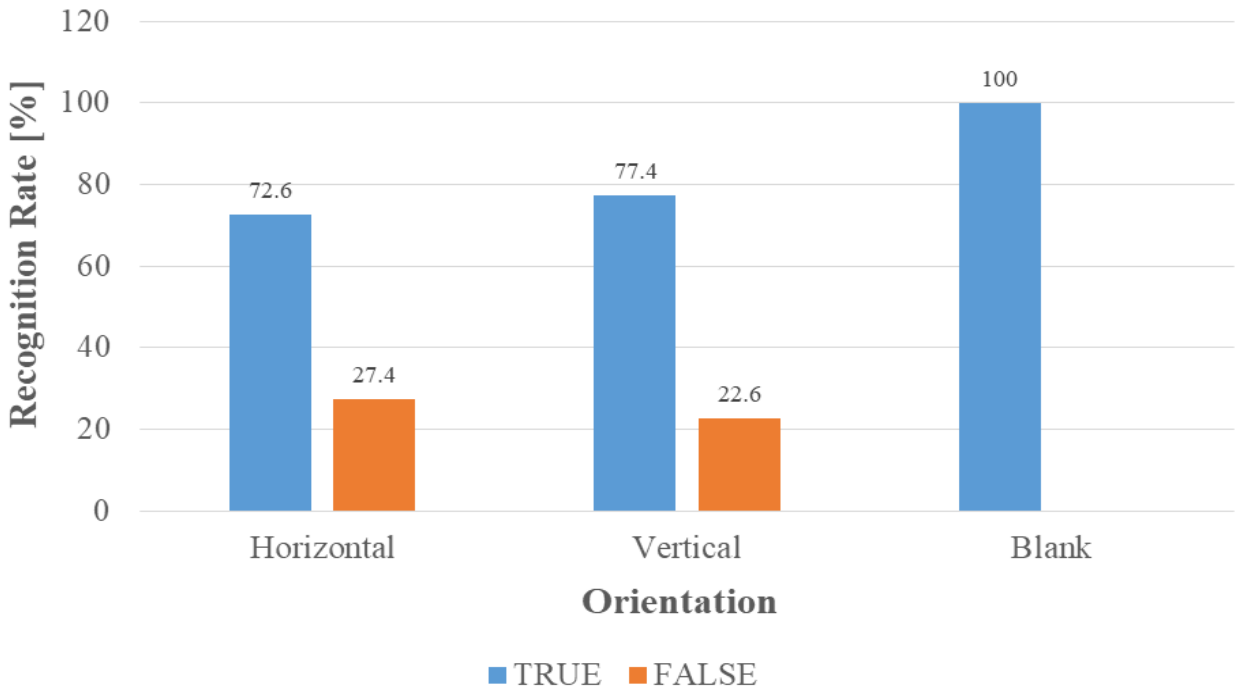

Figure 16: Average Recognition Rate of participants when subjected to different orientations; Blank = no thickness

Figure 18 depicts the participants' average recognition rate when subjected to various amplitudes (Level 1, Level 2, and blank). "TRUE" indicates that the participants correctly answered the questions, while "FALSE" indicates the opposite. Level 1 relates to the first amplitude, 155, and corresponds to x volts, while Level 2 equates to the second amplitude, 179, and corresponds to Y volts. Blank, Level 2, and Level 1 have the highest recognition rates.

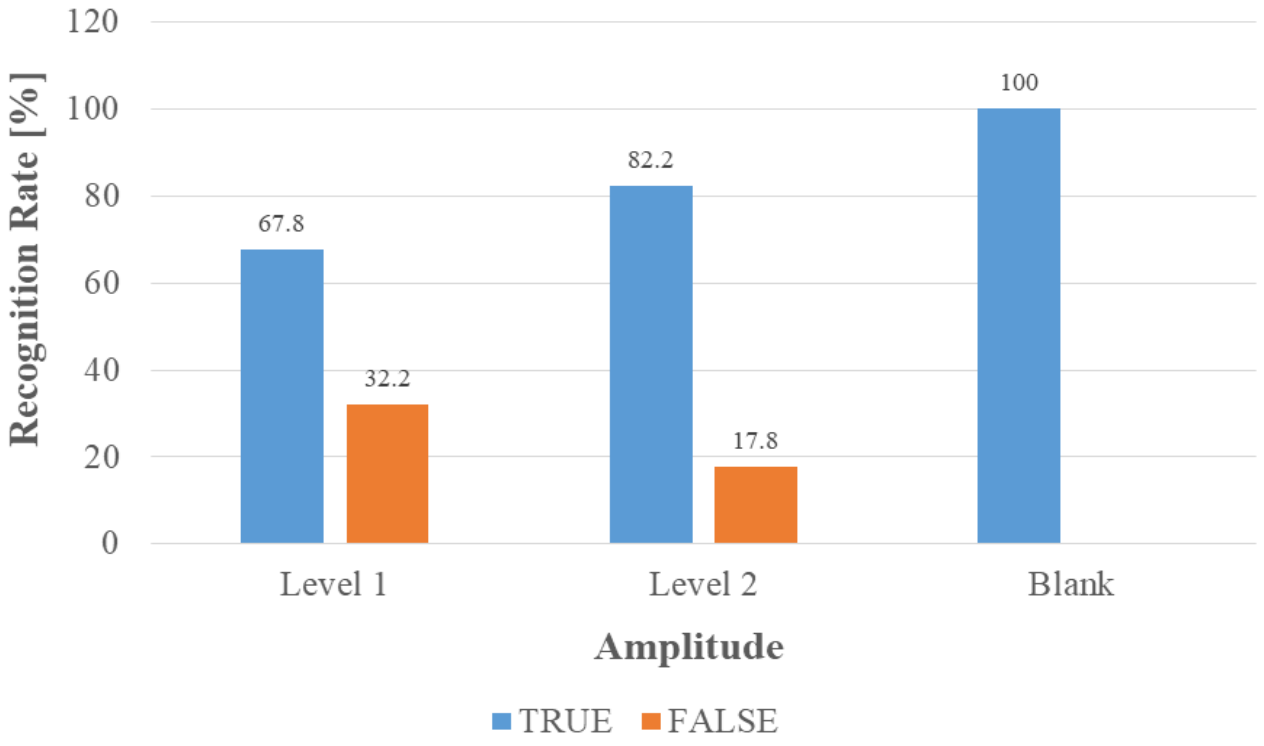

Figure 17: Average Recognition Rate of participants when subjected to different amplitudes; Blank = no thickness

Figure 19 depicts the average recognition rate of participants under two conditions: thickness and orientation. The orientations and thicknesses stay unchanged. Our findings indicate that when coupled with increasing thickness, vertical orientations have higher recognition rates than horizontal orientations.

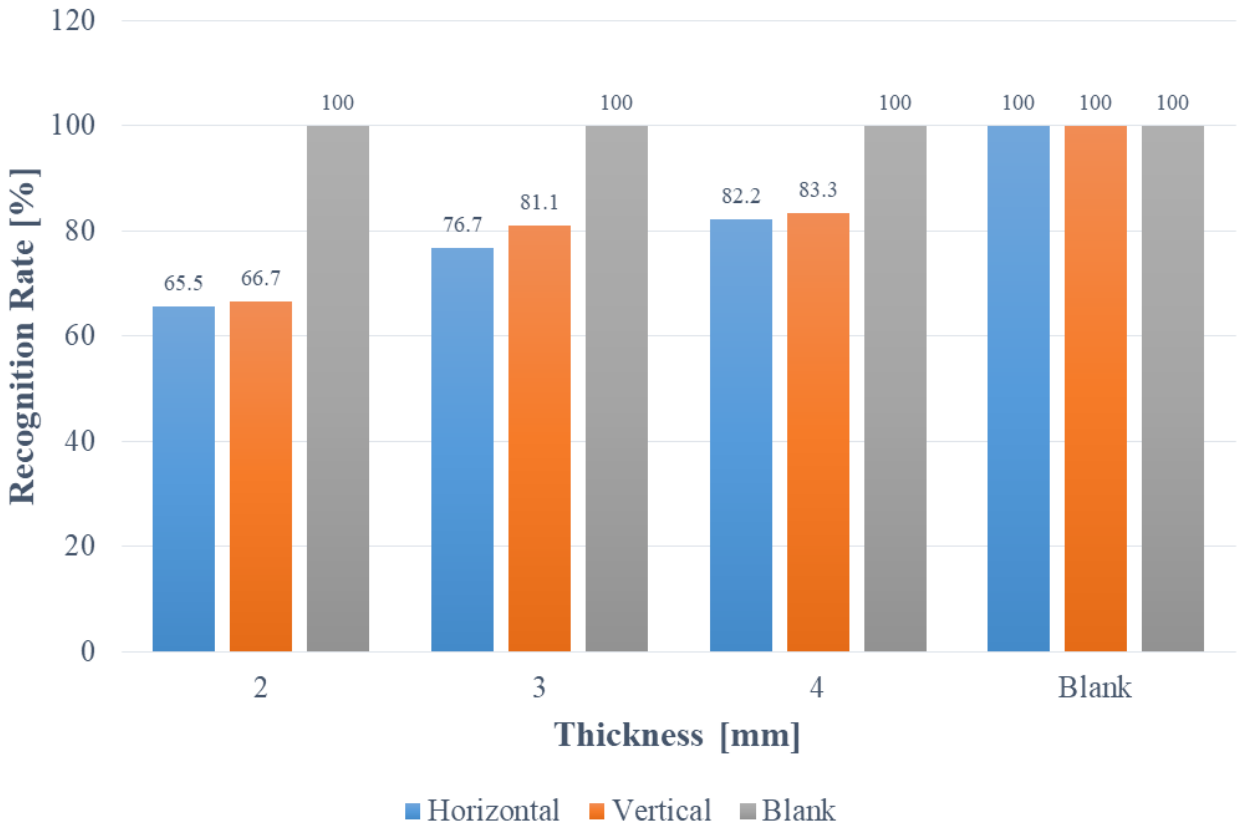

Figure 18: Average Recognition Rate of the participants when experiencing different thicknesses and orientations; Blank= No thickness

The average recognition time of participants when exposed to various thicknesses is visualized in Figure 20. Time was expressed in seconds. Participants spent more time recognizing the smaller thickness, according to our findings. The participants were able to detect the lines faster as the thickness increased. Blank has the highest recognition time.

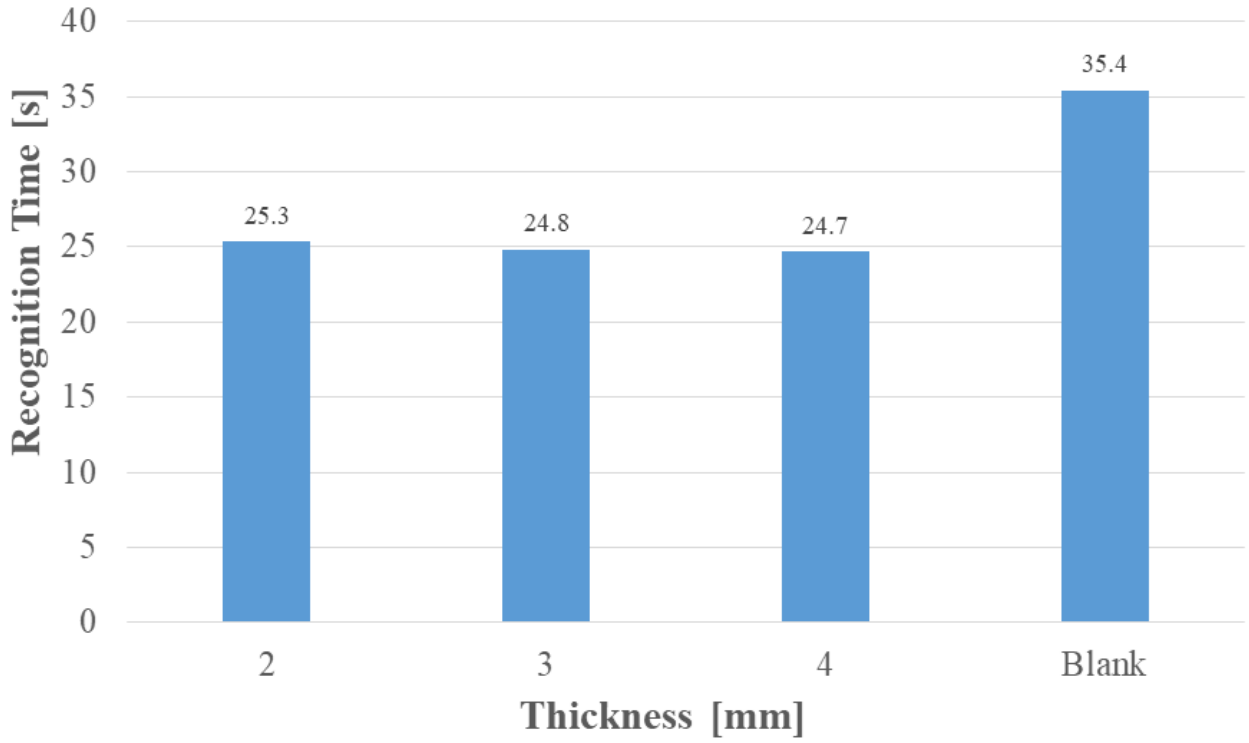

Figure 19: Average Recognition Time of the participants when subjected to different thicknesses; Blank = No thickness.

The investigation sought to ascertain the recognition rate when thickness and orientation were factors. As illustrated in figure 15, the blank had the highest recognition rate. Also, the subjects were able to detect horizontal lines better than vertical orientations as the amplitudes increased. Finally, the blank, Level 2, and Level 1 have respectively the highest recognition rates.

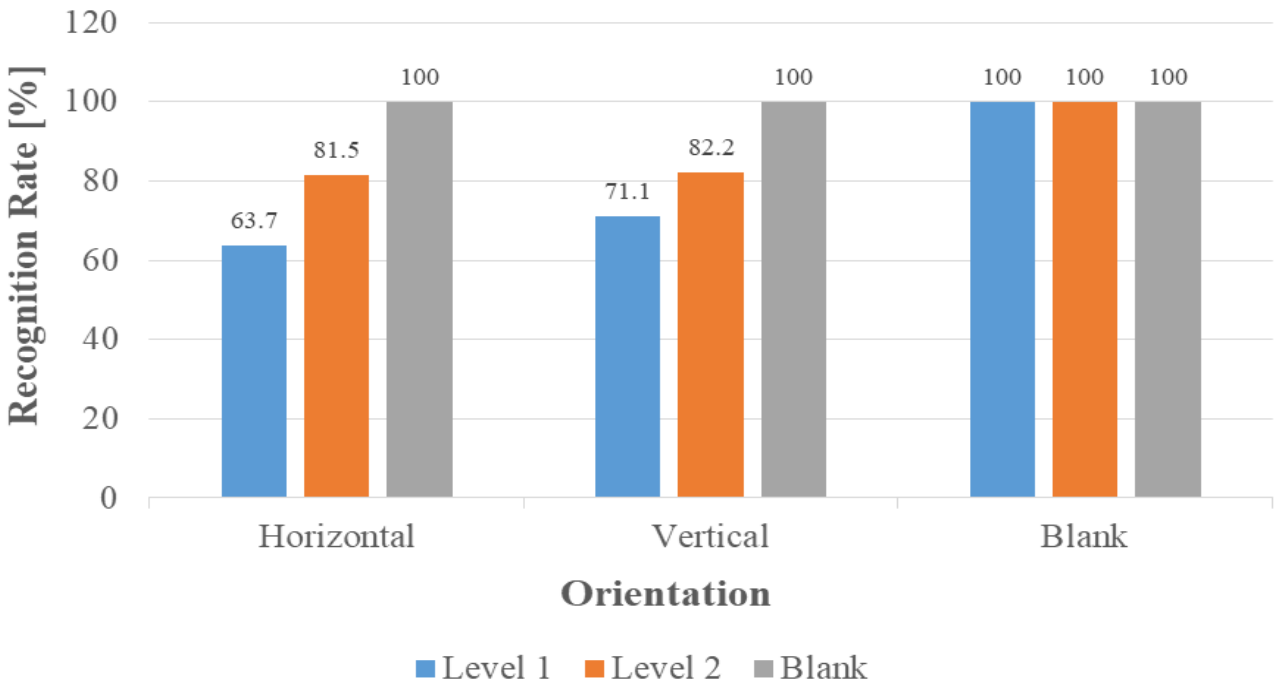

Figure 20: Average Recognition Rate of the participants when experiencing different amplitudes and orientations; Blank = no thickness, Level 1: First amplitude, Level 2: Second amplitude.

Figure 22 illustrates the average recognition rate of participants under two conditions: thickness and amplitude. The thicknesses are 2 mm, 3 mm, 4 mm, and "blank." The amplitudes Level 1 and Level 2 remained the same. Our results suggested that as the thicknesses and amplitudes of the lines increased, participants were more proficient at distinguishing them.

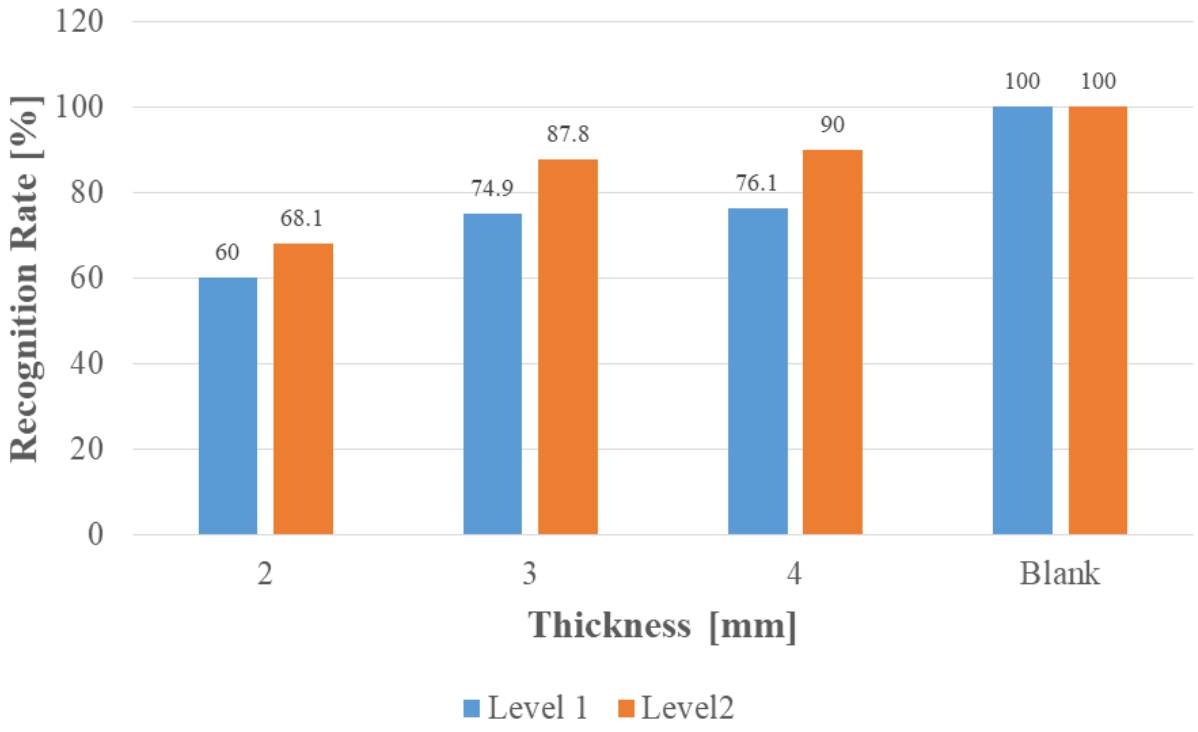

Figure 21: Average Recognition Rate of the participants when experiencing different thicknesses and amplitudes; Control = No thickness, V = Vertical, H= Horizontal, Level 1: First amplitude, Level 2: Second amplitude.

Figure 23 displays the participants' average recognition times under different amplitude Level 1 and Level 2 conditions. According to our research, the participants spent more time identifying the lines at Levels 1, 2, and "Blank" respectively.

.

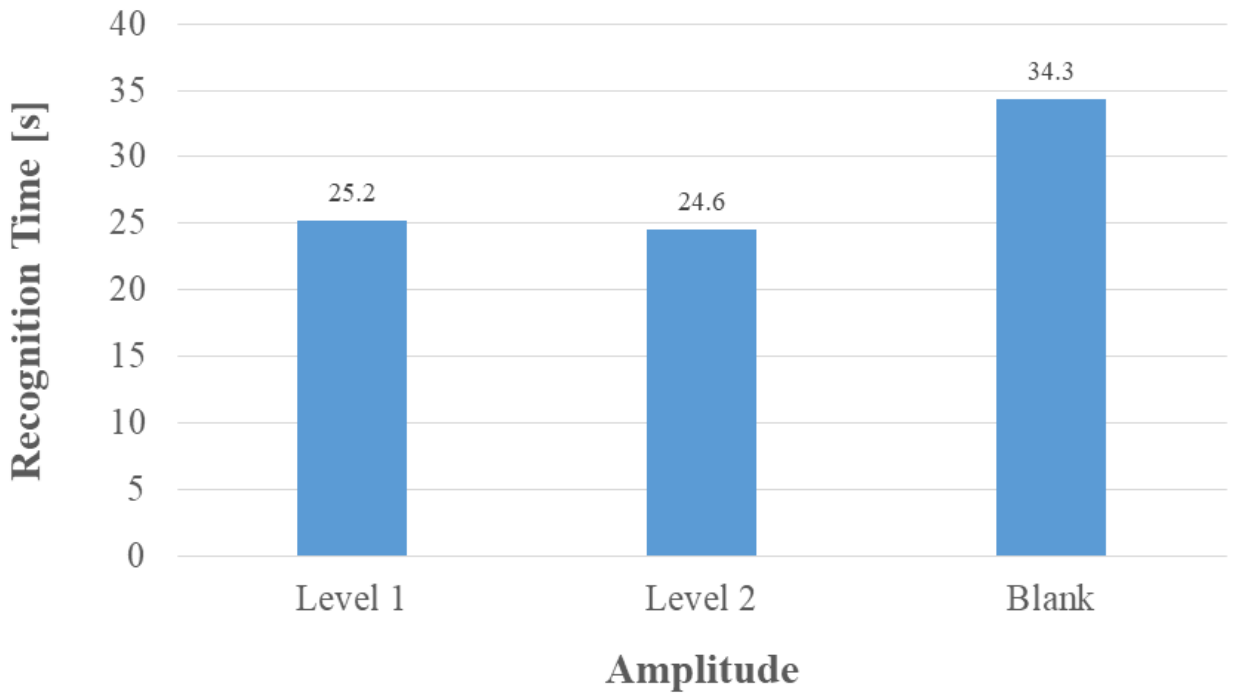

Figure 22: Average Time spent by the participants in recognizing the lines when subjected to the amplitudes. Level 1: First amplitude, Level 2: Second amplitude.

.

7. STATISTICAL ANALYSIS

7.1 Impact of Thickness, Amplitude, and Orientation on Recognition Rate

A binary logistic regression was run through Jamovi software to test the impact of Thickness, Amplitude, and Orientation on recognition rate. A significant model was produced ($p < .001$; $R^2_{McF} = .16$) showing that 16.2% of the total variation in recognition rate was jointly explained by all the independent variables. See Table 6.

Table 6: Overall Model Test of thickness, amplitude and orientation on recognition rate

Model Fit Measures

		Overall Model Test		
Model	**R^2_{McF}**	**χ^2**	**df**	**p**
1	0.162	113	3	< .001

Table 7 shows the coefficients of the model. There is a significant and a positive impact of thickness on recognition rate (*Estimate* = 0.55, t = 4.51, $p < 0.001$, Odds Ratio = 1.74). The increase in thickness by 1 unit (from 2 to 3 and 3 to 4) results in an increase in odds of being recognition rate (True) by 74%. Results also show that there is a positive and significant influence of amplitude on recognition rate (*Estimate* = 0.98, $t = 5.00$, $p < 0.001$, Odds Ratio = 2.65). The change in Amplitude by 1 unit (from Horizontal to Vertical and Vertical to Blank) results in an increase in odds of being recognition rate (True) by 165%. Table 11 shows that there is a positive and significant influence of orientation on recognition rate (*Estimate* = 0.42, $t = 2.16$, $p < 0.05$, Odds Ratio = 1.52). The change in orientation by 1 unit (from Level 1 to Level 2 and Level 2 to Blank) results in an increase in the odds of being recognition rate (True) by 52%.

Model Coefficients - Correct_not

Predictor	Estimate	SE	t	p	Odds ratio
Intercept	-1.961	0.417	-4.70	< .001	0.141
Thickness	0.554	0.123	4.51	< .001	1.739
Amplitude	0.975	0.195	5.00	< .001	2.651
Orientation	0.419	0.194	2.16	0.031	1.520

Note. Estimates represent the log odds of "Correct_not = true" vs. "Correct_not = false". "SE" is the standard error. "t" is the regression coefficient (Estimate) divided by the standard error (SE). "p" is the probability value. Odds ratio is the ratio of the likelihood that something will happen to the likelihood that it will not.

Table 7: Thickness, amplitude, and orientation statistic parameters; Correct_not represents the recognition rate

7.2 Impact of Thickness, Amplitude, and Orientation on Time

Multiple regression was used to investigate the Impact of Thickness, Amplitude, and Orientation on Time. A significant model was produced (F (3,716) = 8.34; $p < .001$; R^2 =.0.02) showing that 2.9% of the total variation in Time was jointly explained by all the independent variables. See Table 8.

Model Fit Measures

		Overall Model Test			
Model	Adjusted R^2	F	df1	df2	p
1	0.0297	8.34	3	716	< .001

Table 8: Overall Model Test of thickness, amplitude and orientation on recognition time

Results in Table 9 shows that there is insignificant impact of Thickness (*Estimate* = 0.53, t = 0.91, p = 0.36), Amplitude (*Estimate* = 1.67, t = 1.88, p = 0.06) and Orientation (*Estimate* = 0.92, t = 1.04, p = 0.29) on recognition time.

Model Coefficients - Time

Predictor	Estimate	SE	t	p
Intercept	18.378	1.328	13.839	< .001
Thickness	0.531	0.583	0.911	0.363
Amplitude	1.670	0.885	1.886	0.060
Orientation	0.922	0.885	1.041	0.298

Table 9: Thickness, amplitude, and orientation statistic parameters; Correct_not represents the recognition time

A significant model was produced (F (3,716) = 8.34; $p < .001$; R^2 =.0.109) showing that 10.9% of the total variation in recognition rate was jointly explained by all the independent variables. See Table 10.

Model Fit Measures

		Overall Model Test			
Model	Adjusted R^2	F	df1	df2	p
1	0.109	30.3	3	716	< .001

Table 10: Overall Model Test of Thickness vs Amplitude, Thickness vs orientation, and Amplitude vs orientation on the recognition rate.

Results in Table 11 show that there is a significant impact of Thickness vs Amplitude (*Estimate* = 0.039, $t = 3.344$, $p < .001$) on the recognition rate while thickness vs amplitude and Amplitude vs Orientation are insignificant.

Model Coefficients - Correct_not

Predictor	Estimate	SE	t	p
Intercept	0.64987	0.0225	28.856	< .001
Thickness ✻ Amplitude	0.03919	0.0117	3.344	< .001
Thickness ✻ Orientation	-0.00367	0.0117	-0.313	0.754
Amplitude ✻ Orientation	-0.00605	0.0145	-0.419	0.676

Table 11: Thickness vs Amplitude, Thickness vs orientation, and Amplitude vs orientation on the recognition rate

7. LIMITATIONS OF THE STUDY

Some challenges and restrictions came up throughout the study that made us adapt and change some plans.

- The coronavirus pandemic has tremendously impacted the project since I sometimes could not go to the university lab and work. Furthermore, the DRV2605LEVM-MD module was purchased abroad in Germany, it took two months and a half to come.
- MATLAB which is used as a user interface could not record the mouse cursor/finger duration on the displayed region when moving at a certain speed. This shows us that MATLAB is not efficiently responding to a real-time action when subjected at a high speed.
- The experiment was firstly done in MATLAB App Designer but at the end of the setup, I, unfortunately, found that MATLAB App Designer does not recognize the finger on the touchscreen. This leads to stepping back and using MATLAB GUIDE.
- MATLAB GUIDE is sometimes a headache when trying to debug errors. There are times that a code can overwrite the previous works you did before. If you don't have any previous backups, you will have to start all the work from scratch. In addition, MATLAB GUIDE is going to be removed in a future release as it is mentioned in their platform. As a consequence, it cannot be used as a solid vehicle in the long run.
- The LRA motors cables were too thin and weak. It was very challenging to manipulate them since they use to get easily cut when a little force was applied to them.

8. CONCLUSION

I suggested a new wearable ring that provides haptic feedback from the kinesthetic channel and can be used in conjunction with tactile interfaces that provide subcutaneous tactile feedback.

I performed human experiments to determine the bare minimum of parameters for presenting stimuli to users. Thanks to the statistical analysis that demonstrate that there is an overall impact of thickness, orientation, and amplitude on recognition rate. However, thickness, amplitude, and orientation have an insignificant impact on recognition time. Furthermore, thickness vs amplitude has a significant impact on recognition rate while thickness vs amplitude and orientation vs amplitude are insignificant. When only one motor is stimulated, the subjects can identify a minimum thickness of 2 mm at the lowest possible vibration level (Level 1). Also, the second vibration amplitude and vertical orientation produced the best human perception outcomes. We discovered that orientations play a significant role in identifying the shapes. Vertically oriented 1D lines can be detected much more easily. We believe this is due to the elliptical shape of the finger. When the finger travels horizontally rather than vertically, the cross-section area from the major axis of the fingertip contains more mechanoreceptors than the minor axis. We intend to conduct additional characterization experiments and create virtual reality applications in the future in order to design a multimodal haptic feedback system that includes the ring and electrovibration.

REFERENCES

[1] M. konnikova, “New yorker,” 2015. [Online]. Available: https://www.newyorker.com/science/maria-konnikova/power-touch. [Accessed 15 6 2022].

[2] R. Golledge, M. Rice and R. Jacobson, “Multimodal interfaces for representing and accessing geospatial information,” *Frontiers of Geographic Information Technology,* p. 38, 2006.

[3] Vera Zasúlich Pérez Ariza, Mauricio Santís-Chaves, “HAPTIC INTERFACES: KINESTHETIC VS. TACTILE SYSTEMS”, journal of Technical-scientifical biannual, vol. 13, pp. 13-29

[4] J. Carter and D. Fourney, “Research Based Tactile and Haptic Interaction Guidelines,” *Guidelines On Tactile and Haptic Interactions Conference (GOTHI-05),* pp. 84-92, 2005.

[5] M. Fogtmann, J. Fritsch and K. Kortbek, “Kinesthetic Interaction,” *Revealing the Bodily Potential in Interaction Design. Proceedings of the 20th Australasian Conference on Computer Human Interaction Designing for Habitus and Habitat,* pp. 89-96, 2008.

[6] L. Chen, M. Holbein and J. Zelek, “Intro to haptic communications for high school students,” *in Proceedings - IEEE International Conference on Robotics and Automation*, 2006.

[7] J. H. Kirman, “Tactile apparent movement: The effects of number of stimulators,” *Journal of Experimental Psychology*, vol. 103, no. 6, pp. 1175–1180, 1974, place: US Publisher: American Psychological Association.

[8] T. Lim, G. Germánico and M. Castillo, “ARE YOU HAPTIC A BAD DAY?,” in *Proceedings of the ASME 2014 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2014*, 2014.

[9] L. maddox, Nabokov's Novel in English, Athens: Bob Nance, 2009.

[10] Nathan Hurst, Can You Feel Me Now? The Sensational Rise of Haptic Interfaces

2013. [Online]. Available: www.wired.com/2013/02/haptics/

[11] H. Culbertson, S. B. Schorr, and A. M. Okamura, "Haptics: The Present and Future of Artificial Touch Sensation," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 385–409, 2018, eprint: https://doi.org/10.1146/annurev-control-060117-105043. [Online]. Available: https://doi.org/10.1146/annurev-control-060117-105043

[12] A. Talhan, H. Kim, and S. Jeon, "Tactile Ring: Multi-Mode Finger-Worn Soft Actuator for Rich Haptic Feedback," IEEE Access, vol. 8, pp. 957–966, 2020, conference Name: IEEE Access.

[13] Purves D, Augustine GJ, Fitzpatrick D, et al., editors. Neuroscience. 2nd edition. Sunderland (MA): Sinauer Associates; 2001. Mechanoreceptors Specialized to Receive Tactile Information. Available from: https://www.ncbi.nlm.nih.gov/books/NBK10895

[14] F. Chinello, M. Malvezzi, C. Pacchierotti, and D. Prattichizzo, "A three DoFs wearable tactile display for exploration and manipulation of virtual objects," *in 2012 IEEE Haptics Symposium (HAPTICS)*, Mar. 2012, pp. 71–76, iSSN: 2324-7355.

[15] S. Nisar, M. O. Martinez, T. Endo, F. Matsuno, and A. M. Okamura, "Effects of Different Hand-Grounding Locations on Haptic Performance With a Wearable Kinesthetic Haptic Device," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 351–358, Apr. 2019, conference Name: IEEE Robotics and Automation Letters.

[16] Federica Barontini , Manuel G. Catalano , Giorgio Grioli et al. "Wearable Integrated Soft Haptics in a Prosthetic Socket," *Soft Robotics for Human Cooperation and Rehabilitation,*2010

[17] A.Stanley & J. Gwilliams &. A.Okamura. (“Haptic jamming: A deformable geometry, variable stiffness tactile display using pneumatics and particle jamming.”) in *World Haptics Conference,* Washington, 2013.

[18] B. Stephens-Fripp, R. Mutlu, and G. Alici, “Using Vibration Motors to Create Tactile Apparent Movement for Transradial Prosthetic Sensory Feedback,” *in 2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob),* Aug. 2018, pp. 213–218, iSSN: 2155-1782.

[19] J. H. Kirman, “Tactile apparent movement: the effects of number of stimulators,” *7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob),* vol. 103, pp. 1175-80, 1974.

[20] Hongbin Liu, Kien Cuong Nguyen, Véronique Perdereau, et al. “Finger contact sensing and the application in dexterous hand manipulation”, *Autonomous Robots* vol. 39, pp. 25–41 (2015)

[21] A. Documentation, “Arduino cc,” [Online]. Available: https://docs.arduino.cc/. [Accessed 15 06 2022].

[22] H. Barragan, “The Untold History of Arduino,” [Online]. Available: https://arduinohistory.github.io/. [Accessed 15 6 2022].

[23] SPARKFUN DOCUMENTATION. [Online]. Available: https://learn.sparkfun.com/tutorials/i2c/introduction. [Accessed 15 6 2022]

[24] TEXAS INSTRUMENT DOCUMENTATION, “DRV2605L,” [Online]. Available: https://www.ti.com/product/DRV2605L. [Accessed 15 06 2022].

[25] TEXAS INSTRUMENT DOCUMENATION, “DRV2605LEVM*MD,” [Online]. Available: https://www.ti.com/tool/DRV2605LEVM-MD#tech-docs. [Accessed 15 06 2022].

[26] “microdcmotors,” [Online]. Available: https://microdcmotors.com/product/8mm-linear-resonant-actuator-3mm-type-mdm-elv0832b-205hz-or-235hz. [Accessed 16 06 2022].

[27] B. T¨uys¨uz, E. G¨uney, D. Serpiciler, and M. Ayyildiz, “A Low-Cost 3D-Printed Soft Pneumatic Actuator,” *in 2021 13th International Conference on Electrical and Electronics Engineering (ELECO)*, Nov. 2021, pp. 124–128.

[28] MATHWORKS, “MATLAB GUI,” [Online]. Available: https://in.mathworks.com/discovery/MATLAB-gui.html. [Accessed 15 6 2022].

[29] MATHWORKS, “MATLAB APPDESIGNER,” [Online]. Available: https://in.mathworks.com/products/MATLAB/app-designer/comparing-guide-and-app-designer.html. [Accessed 15 06 2022].

[30] Y. Shao, V. Hayward, and Y. Visell, “Spatial patterns of cutaneous vibration during whole-hand haptic interactions,” *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 4188–4193, Apr. 2016, publisher: Proceedings of the National Academy of Sciences. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1520866113

APPENDIX

APPENDIX A: INITIALISATION OF THE MOTORS IN ARDUINO UNO

```
#include <Sparkfun_DRV2605L.h> //SparkFun Haptic Motor Driver Library
#include <DRV2605LEVM_MD.h>
#include <Wire.h> //I2C library

SFE_HMD_DRV2605L HMD; //Create haptic motor driver object
DRV2605LEVM_MD multi_drv; //Create multi motor driver object

uint8_t status_reg_result;
uint8_t control3_reg_result;

uint8_t reg_data = 0xB5; // Hex code for choosing unsigned int for RTP
//uint8_t reg_data = 0xAB; // Hex code for choosing signed int for RTP

uint8_t motor_speed = 0x00;

// Motor selection typdefs
motor_sel motor_no = NO_MOTOR;

// Communication variables
int i = 0;
String input_cmd;
int incomingByte = 0; // for incoming serial data

int speed_cmd = 150;


void setup() {

  Serial.begin(38400);
  HMD.begin();

  // Autocalibrate the motors at the start
  Serial.println("Autocalibrating the motor drivers...");
  // Start autocalibration by setting the GO bit to 1
  HMD.Mode(0x07);

  // Initialize the motor drivers for RTP mode
```

```
  Serial.println("Intializing the motor drivers...");

 // HMD.Mode(0); // Internal trigger input mode -- Must use the GO() function to
trigger playback.
  HMD.Mode(0x05); // RTP Mode -- Must use the RTP function to trigger amplitude.
  HMD.MotorSelect(0xB6); // LRA motor, 4x Braking, Medium loop gain, 1.365x
back EMF gain
  HMD.Library(6); //1-5 & 7 for ERM motors, 6 for LRA motors

  // Set the RTP data format type to unsigned int
  control3_reg_result = HMD.readDRV2605L(CONTROL3_REG);
  Serial.print("Control3 Reg :");
  Serial.println(control3_reg_result, HEX);

  HMD.writeDRV2605L(CONTROL3_REG, reg_data);
  Serial.println("Setting the DATA_FORMAT_RTP to unsigned...");
  control3_reg_result = HMD.readDRV2605L(CONTROL3_REG);
  Serial.print("Control3 Reg :");
  Serial.println(control3_reg_result,HEX);

  //Serial.println("Setting the CONTROL2_REG to unidirectional...");
  //HMD.writeDRV2605L(CONTROL2_REG, 0x7F);

  HMD.go();
  status_reg_result = HMD.readDRV2605L(STATUS_REG);
  Serial.print("Autocalibration done! With the status code of ");
  Serial.println(status_reg_result, HEX);

  Serial.println("Listening the MATLAB commands...");

}

void loop() {

  if (Serial.available() > 0) {
    // read the incoming byte:
    input_cmd = Serial.readString();
    speed_cmd = input_cmd.toInt();
  }

  // CHECK FOR COMMUNICATION
  input_cmd = "MOTOR_1_2_3_4\r\n";

  // CHECK FOR MOTOR NO
  if (input_cmd == "NO_MOTOR\r\n") {
    motor_no = NO_MOTOR;
```

```
  motor_speed = 0;
  multi_drv.initialize_motors(motor_no);
  Serial.println("No motor is selected");
 }
 else if(input_cmd == "MOTOR_1\r\n") {
  motor_no = MOTOR_1;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 1 is selected");
 }
 else if(input_cmd == "MOTOR_2\r\n") {
  motor_no = MOTOR_2;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 2 is selected");
 }
 else if(input_cmd == "MOTOR_3\r\n") {
  motor_no = MOTOR_3;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 3 is selected");
 }
 else if(input_cmd == "MOTOR_4\r\n") {
  motor_no = MOTOR_4;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 4 is selected");
 }
 else if(input_cmd == "MOTOR_1_2\r\n") {
  motor_no = MOTOR_1_2;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  //Serial.println("Motor 1 and 2 are selected");
 }
 else if(input_cmd == "MOTOR_1_3\r\n") {
  motor_no = MOTOR_1_3;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 1 and 3 are selected");
 }
 else if(input_cmd == "MOTOR_1_4\r\n") {
  motor_no = MOTOR_1_4;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 1 and 4 are selected");
 }
```

```
  else if(input_cmd == "MOTOR_2_3\r\n") {
    motor_no = MOTOR_2_3;
    motor_speed = 128;
    multi_drv.initialize_motors(motor_no);
    Serial.println("Motor 2 and 3 are selected");
  }
  else if(input_cmd == "MOTOR_2_4\r\n") {
    motor_no = MOTOR_2_4;
    motor_speed = 128;
    multi_drv.initialize_motors(motor_no);
    Serial.println("Motor 2 and 4 are selected");
  }
  else if(input_cmd == "MOTOR_3_4\r\n") {
    motor_no = MOTOR_3_4;
    motor_speed = 128;
    multi_drv.initialize_motors(motor_no);
    Serial.println("Motor 3 and 4 are selected");
  }
  else if(input_cmd == "MOTOR_1_2_3_4\r\n") {
    motor_no = MOTOR_1_2_3_4;
    motor_speed = 128;
    multi_drv.initialize_motors(motor_no);
    //Serial.println("Motor 1 , 2 ,3 and 4 are selected");
  }


  HMD.RTP(speed_cmd);
  //delay(4);
  HMD.writeDRV2605L(LRARESPERIOD_REG, 42);
  uint8_t freq_reg_value = HMD.readDRV2605L(LRARESPERIOD_REG);
  //uint8_t freq_reg_value =
multi_drv.read_DRV2605LEVM_MD(DRV2605L_ADDR , 0x22);
  Serial.print("Frequncy Reg :");
  Serial.println(freq_reg_value, DEC);
//
 }
```

APPENDIX B: STOPPING THE MOTORS FROM RUNNING ARDUINO CODE

```
#include <Sparkfun_DRV2605L.h> //SparkFun Haptic Motor Driver Library
#include <DRV2605LEVM_MD.h>
#include <Wire.h> //I2C library

SFE_HMD_DRV2605L HMD; //Create haptic motor driver object
DRV2605LEVM_MD multi_drv; //Create multi motor driver object

uint8_t status_reg_result;
uint8_t control3_reg_result;

uint8_t reg_data = 0xB5; // Hex code for choosing unsigned int for RTP
//uint8_t reg_data = 0xAB; // Hex code for choosing signed int for RTP

uint8_t motor_speed = 0x00;

// Motor selection typdefs
motor_sel motor_no = NO_MOTOR;

// Communication variables
int i = 0;
String input_cmd;
int incomingByte = 0; // for incoming serial data

int speed_cmd = 150;


void setup() {

  Serial.begin(38400);
  HMD.begin();

  // Autocalibrate the motors at the start
  Serial.println("Autocalibrating the motor drivers...");
  // Start autocalibration by setting the GO bit to 1
  HMD.Mode(0x07);

  // Initialize the motor drivers for RTP mode
  Serial.println("Intializing the motor drivers...");

 // HMD.Mode(0); // Internal trigger input mode -- Must use the GO() function to
trigger playback.
  HMD.Mode(0x05); // RTP Mode -- Must use the RTP function to trigger amplitude.
```

```
  HMD.MotorSelect(0xB6); // LRA motor, 4x Braking, Medium loop gain, 1.365x
back EMF gain
  HMD.Library(6); //1-5 & 7 for ERM motors, 6 for LRA motors

  // Set the RTP data format type to unsigned int
  control3_reg_result = HMD.readDRV2605L(CONTROL3_REG);
  Serial.print("Control3 Reg :");
  Serial.println(control3_reg_result, HEX);

  HMD.writeDRV2605L(CONTROL3_REG, reg_data);
  Serial.println("Setting the DATA_FORMAT_RTP to unsigned...");
  control3_reg_result = HMD.readDRV2605L(CONTROL3_REG);
  Serial.print("Control3 Reg :");
  Serial.println(control3_reg_result,HEX);

  //Serial.println("Setting the CONTROL2_REG to unidirectional...");
  //HMD.writeDRV2605L(CONTROL2_REG, 0x7F);

  HMD.go();
  status_reg_result = HMD.readDRV2605L(STATUS_REG);
  Serial.print("Autocalibration done! With the status code of ");
  Serial.println(status_reg_result, HEX);

  Serial.println("Listening the MATLAB commands...");

}

void loop() {

  if (Serial.available() > 0) {
    // read the incoming byte:
    input_cmd = Serial.readString();
    speed_cmd = input_cmd.toInt();
  }

  // CHECK FOR COMMUNICATION
  input_cmd = "MOTOR_1_2_3_4\r\n";

  // CHECK FOR MOTOR NO
  if (input_cmd == "NO_MOTOR\r\n") {
    motor_no = NO_MOTOR;
    motor_speed = 0;
    multi_drv.initialize_motors(motor_no);
    Serial.println("No motor is selected");
  }
  else if(input_cmd == "MOTOR_1\r\n") {
```

```
  motor_no = MOTOR_1;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 1 is selected");
 }
 else if(input_cmd == "MOTOR_2\r\n") {
  motor_no = MOTOR_2;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 2 is selected");
 }
 else if(input_cmd == "MOTOR_3\r\n") {
  motor_no = MOTOR_3;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 3 is selected");
 }
 else if(input_cmd == "MOTOR_4\r\n") {
  motor_no = MOTOR_4;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 4 is selected");
 }
 else if(input_cmd == "MOTOR_1_2\r\n") {
  motor_no = MOTOR_1_2;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  //Serial.println("Motor 1 and 2 are selected");
 }
 else if(input_cmd == "MOTOR_1_3\r\n") {
  motor_no = MOTOR_1_3;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 1 and 3 are selected");
 }
 else if(input_cmd == "MOTOR_1_4\r\n") {
  motor_no = MOTOR_1_4;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 1 and 4 are selected");
 }
 else if(input_cmd == "MOTOR_2_3\r\n") {
  motor_no = MOTOR_2_3;
  motor_speed = 128;
  multi_drv.initialize_motors(motor_no);
  Serial.println("Motor 2 and 3 are selected");
```

```
  }
  else if(input_cmd == "MOTOR_2_4\r\n") {
    motor_no = MOTOR_2_4;
    motor_speed = 128;
    multi_drv.initialize_motors(motor_no);
    Serial.println("Motor 2 and 4 are selected");
  }
  else if(input_cmd == "MOTOR_3_4\r\n") {
    motor_no = MOTOR_3_4;
    motor_speed = 128;
    multi_drv.initialize_motors(motor_no);
    Serial.println("Motor 3 and 4 are selected");
  }
  else if(input_cmd == "MOTOR_1_2_3_4\r\n") {
    motor_no = MOTOR_1_2_3_4;
    motor_speed = 128;
    multi_drv.initialize_motors(motor_no);
    //Serial.println("Motor 1 , 2 ,3 and 4 are selected");
  }


  HMD.RTP(speed_cmd);
  //delay(4);
  HMD.writeDRV2605L(LRARESPERIOD_REG, 42);
  uint8_t freq_reg_value = HMD.readDRV2605L(LRARESPERIOD_REG);
  //uint8_t freq_reg_value =
multi_drv.read_DRV2605LEVM_MD(DRV2605L_ADDR , 0x22);
  Serial.print("Frequncy Reg :");
  Serial.println(freq_reg_value, DEC);
//
}
```

APPENDIX C: MAIN CODE IN THE ARDUINO THAT COMMUNICATES WITH MATLAB

```
/**
Arduino ATmega328P Datasheet Link
Read the datasheet for any Register definition

https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

**/

// Header files for Haptic Motors and handling AVR Interrupts
#include <Sparkfun_DRV2605L.h>          //SparkFun Haptic Motor Driver Library
#include <DRV2605LEVM_MD.h>                 // For DRV2605L
#include <Wire.h>                           //I2C library
#include "avr/io.h"                         // For using Arduino Registers
#include <avr/wdt.h>                        // For using Watchdog Timer
#include "avr/interrupt.h"                  // For using Global & USART Interrupts


SFE_HMD_DRV2605L HMD;                       //Create haptic motor driver object
DRV2605LEVM_MD multi_drv;           //Create multi motor driver object

uint8_t status_reg_result = 0;
uint8_t control3_reg_result = 0;
uint8_t LRA_reg_result = 0;
uint8_t reg_data = 0xB9; // Hex code for choosing unsigned int for RTP

// Motor selection typdef
char speed_cmd;                                         // For storing Motor Speed
motor_sel motor_no = NO_MOTOR;                  // enum for Number of Motors
int     motor_speed = 128;                  // Variable for Motor Speed

// Communication variables
volatile byte str_index = 0;                // String Index
volatile bool stringComplete = false; // Boolean to indicate the Sent string is complete
volatile bool newWord = false;                  // Boolean to indicate the Data completion
volatile bool recvCR = false;               // Boolean to indicate the reception of Carriage Return
```

```
volatile bool recvLF = false;                   // Boolean to indicate the reception of
Line Feed
volatile char cmdArr[10];                       // Array for Storing the actual
command
volatile uint8_t ind_arr = 0;                   // Index of the extracted received
message

// Functions Declarations
void USART_Init(void);                          // USART Initialization
void USART_Disable(void);                       // USART Disable function
void USART_putc(unsigned char);                 // USART function for
transmitting Single Byte (One Character)
void USART_puts(char*);                         // USART function for
transmitting a string
void USART_newline(void);                       // USART function for
transmitting NewLine

// Serial ISR for Receiving the MATLAB Data
// Every time a Character is being read by Arduino Serial (USART), this function will
be called automatically.
// The function is implemented to receive the whole string from MATLAB character
by character
ISR(USART_RX_vect)
{
   char ReceivedByte;
   ReceivedByte = UDR0; // Fetch the received byte value into the variable "
ByteReceived "

   if(str_index == 0) // Receiving 1st Byte, speed of the motor
   {
      speed_cmd = ReceivedByte; // Copying Data from ReceivedByte into speed_cmd
                  str_index++; // Incrementing the string Index to not receive the speed
again
   }
   if(ReceivedByte == '&') // Checking for '&' separater
   {
      newWord = true; // Making a variable true, to be used for the next received
character
   }
   if(newWord == true && ReceivedByte != '&' && ReceivedByte != '!') // Receiving
Number of motor data, e.g "1234"
   {
      cmdArr[ind_arr++] = ReceivedByte;
   }
   if(str_index != 0 && ReceivedByte == '!') // Checking END of message by
comparing with '!' character
```

```
   {
                    newWord = false;
                    recvCR = true;
                    cmdArr[ind_arr] = '\0'; // Adding NULL terminator for making a valid
string
                }
                if(recvCR == true && ReceivedByte == 0x0D) // Receiving Carriage
Return, 0x0D
                {
                    recvCR = false;
                    recvLF = true;
                }
                if(recvLF == true && ReceivedByte == 0x0A) // Receiving Line Feed,
0x0A
                {
       stringComplete = true; // Making a variable true, to be used in main loop -->
Important

                    // Resetting all the variables, to be used for the next incoming message.
e.g, "4&1234!"
       str_index = 0;
       ind_arr = 0;
       newWord = false;
                    recvLF = false;
   }
}

void setup() {

   wdt_disable(); // Disable Watchdog Timer
   cli(); // Disable Global Interrupt Register
   USART_Init(); // Initialization USART with Interrupts
   sei(); // Enable the Global Interrupt Enable flag so that interrupts can be processed

   USART_puts("Testing MATLAB Motors");
   USART_newline();

   HMD.begin();

   USART_puts("Status Register 0x");
   USART_putc(HMD.readDRV2605L(STATUS_REG));
   USART_newline();
   // Autocalibrate the motors at the start
   USART_puts("Autocalibrating the motor drivers...");
   USART_newline();
   // Start autocalibration by setting the GO bit to 1
```

```
   HMD.Mode(0x07);

   // Initialize the motor drivers for RTP mode
   USART_puts("Intializing the motor drivers...");
   USART_newline();
   HMD.Mode(0x05); // RTP Mode -- Must use the RTP function to trigger
amplitude.
   HMD.MotorSelect(0xB6); // LRA motor, 4x Braking, Medium loop gain, 1.365x
back EMF gain
   HMD.Library(6); //1-5 & 7 for ERM motors, 6 for LRA motors

   // Get the LRA register value
   //LRA_reg_result = HMD.readDRV2605L(OLP_REG);
   //Serial.print("LRA freq Reg :");
   // Serial.println(LRA_reg_result, DEC);

   // Set the LRA register value
   HMD.writeDRV2605L(OLP_REG, 0x3A);
   LRA_reg_result = HMD.readDRV2605L(OLP_REG);
   USART_puts("New LRA freq Reg :");
   USART_putc(LRA_reg_result);
   USART_newline();

   // Set the RTP data format type to unsigned int
   control3_reg_result = HMD.readDRV2605L(CONTROL3_REG);
   USART_puts("Control3 Reg :");
   USART_putc(control3_reg_result);
   USART_newline();

   HMD.writeDRV2605L(CONTROL3_REG, reg_data);
   USART_puts("Setting the DATA_FORMAT_RTP to unsigned...");
   USART_newline();

   control3_reg_result = HMD.readDRV2605L(CONTROL3_REG);
   USART_puts("Control3 Reg :");
   USART_putc(control3_reg_result);
   USART_newline();

   HMD.go();
   status_reg_result = HMD.readDRV2605L(STATUS_REG);
   USART_puts("Autocalibration done! With the status code of ");
   USART_putc(status_reg_result);
   USART_newline();

   USART_puts("Listening the MATLAB commands...");
   USART_newline();
```

```
   multi_drv.initialize_motors(NO_MOTOR);
   HMD.RTP(0);
                // Enabling Watchdog Timer for 4sec, if wdt_reset() not called in 4sec,
Arduino will be Reset automatically.
   wdt_enable(WDTO_4S);
}

void loop() {
   wdt_reset(); // Known as "Watchdog Kick", if not called in less than 4sec, Arduino
will be Reset automatically

   if(stringComplete == true)// Checking if the whole message has been received. e.g.
"0&1234!"
   {
      stringComplete = false; // Reset the variable, only set by ISR(USART_RX_vect)
      /* USART_puts("Complete");
      USART_newline();
      USART_newline(); */
                  // Checking for valid message using strcmp(For comparing two strings)
      if(strcmp(cmdArr, "1234") == 0)                        // For Motors 1,2,3,4
      {
         // USART_puts("1234");
         motor_no = MOTOR_1_2_3_4;
         multi_drv.initialize_motors(motor_no);
      }
                  else if(strcmp(cmdArr, "1") == 0)      // For Motors 1
      {
         motor_no = MOTOR_1;
         multi_drv.initialize_motors(motor_no);
      }
                  else if(strcmp(cmdArr, "2") == 0)      // For Motors 2
      {
         motor_no = MOTOR_2;
         multi_drv.initialize_motors(motor_no);
      }
                  else if(strcmp(cmdArr, "3") == 0)      // For Motors 3
      {
         motor_no = MOTOR_3;
         multi_drv.initialize_motors(motor_no);
      }
                  else if(strcmp(cmdArr, "4") == 0)      // For Motors 4
      {
         motor_no = MOTOR_4;
         multi_drv.initialize_motors(motor_no);
      }
```

```
                else if(strcmp(cmdArr, "12") == 0)    // For Motors 1, 2
        {
          motor_no = MOTOR_1_2;
          multi_drv.initialize_motors(motor_no);
        }
                else if(strcmp(cmdArr, "13") == 0)    // For Motors 1, 3
        {
          motor_no = MOTOR_1_3;
          multi_drv.initialize_motors(motor_no);
        }
                else if(strcmp(cmdArr, "14") == 0)    // For Motors 1, 4
        {
          motor_no = MOTOR_1_4;
          multi_drv.initialize_motors(motor_no);
        }
                else if(strcmp(cmdArr, "23") == 0)    // For Motors 2, 3
        {
          motor_no = MOTOR_2_3;
          multi_drv.initialize_motors(motor_no);
        }
                else if(strcmp(cmdArr, "24") == 0)    // For Motors 2, 4
        {
          motor_no = MOTOR_2_4;
          multi_drv.initialize_motors(motor_no);
        }
                else if(strcmp(cmdArr, "34") == 0)    // For Motors 3, 4
        {
          motor_no = MOTOR_3_4;
          multi_drv.initialize_motors(motor_no);
        }
                else
                {
                        motor_no = NO_MOTOR;
          multi_drv.initialize_motors(motor_no);
                }

        /* USART_newline();

        USART_puts("Speed: "); */
                // Switch is equivalent to if else statement, except it directly jumps to
the valid statement
                // Checking which speed is received, and assigning the speed to motors
accordingly.
        switch(speed_cmd)
        {
          case '0':
```

```
            motor_speed = 128;
            // USART_putc('0');
            break;
         case '1':
            motor_speed = 154;
            break;
         case '2':
            motor_speed = 179;
            break;
         case '3':
            motor_speed = 205;
            break;
         case '4':
            motor_speed = 230;
                                 // USART_putc('4');
            break;
         case '5':
            motor_speed = 255;
            break;
         default:
            motor_speed = 128;
            break;
      } // END Switch

      // USART_newline();

   }// END "if" for stringComplete check

               // Use the speed command
               HMD.RTP(motor_speed);
}// END loop

/**
Below functions are for using USART0 of Arduino, ATmega328P
In the datasheet, please read Section 19 (USART0), Starting Page 143

**/

// Initialization USART of AVR
void USART_Init(void)
{
               // Baud Rate Selection Register, Datasheet Page 165
   //UBRR0 = 103; //9600
   //UBRR0 = 8; //115200
               UBRR0 = 25; //38400
               // USART Initialization, Datasheet Page 148
```

```
   UCSR0C = (1<<USBS0)|(3<<UCSZ00);
            // Registers for Enabling RX, TX of USART, Datasheet Page 160,
UCSRnB
            // Enable RX Complete Interrupt, Receiver Enable & Transmitter Enable
bits
   UCSR0B = ((1<<RXCIE0) | (1<<RXEN0) | (1<<TXEN0));
}

// Disabling USART of AVR
void USART_Disable(void)
{
            // Disable RX Complete Interrupt, Receiver Enable & Transmitter Enable
bits
   UCSR0B = ((0<<RXCIE0) | (0<<RXEN0) | (0<<TXEN0));
}

// Transmitting a character on USART of AVR
void USART_putc(unsigned char data)
{
   /* Wait for empty transmit buffer */
   while (!( UCSR0A & (1<<UDRE0)))
   ;
   /* Put data into buffer, sends the data */
   UDR0 = data;
}

// Transmitting a string on USART of AVR
void USART_puts(char* s)
{
   // transmit character until NULL is reached
   while(*s > 0)
   {
      USART_putc(*s++);
   }
}

// Transmitting newline on USART of AVR
void USART_newline(void)
{
   USART_putc(0x0D);        // Transmitting Carriage Return, '\n'
   USART_putc(0x0A);        // Transmitting Line Feed, '\r'
}
```

APPENDIX D: MAIN MATLAB CODE

```
function varargout = franck_guide(varargin)
% FRANCK_GUIDE MATLAB code for franck_guide.fig
%      FRANCK_GUIDE, by itself, creates a new FRANCK_GUIDE or raises the
existing
%      singleton*.
%
%      H = FRANCK_GUIDE returns the handle to a new FRANCK_GUIDE or the
handle to
%      the existing singleton*.
%
%      FRANCK_GUIDE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FRANCK_GUIDE.M with the given input
arguments.
%
%      FRANCK_GUIDE('Property','Value',...) creates a new FRANCK_GUIDE or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before franck_guide_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to franck_guide_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help franck_guide

% Last Modified by GUIDE v2.5 22-Feb-2022 13:54:14

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
   'gui_Singleton',  gui_Singleton, ...
   'gui_OpeningFcn', @franck_guide_OpeningFcn, ...
   'gui_OutputFcn',  @franck_guide_OutputFcn, ...
   'gui_LayoutFcn',  [] , ...
   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before franck_guide is made visible.
function franck_guide_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to franck_guide (see VARARGIN)

% Choose default command line output for franck_guide
handles.output = hObject;
% Plot patch on uiaxes
%hold on

% Read experiment data from a CSV file

[~,~,data] = xlsread('excel_data_final.xlsx');
data(1,:) = []; % remove the header line

% randomly permute the rows of data without repeating the value:
data = data(randperm(size(data,1)),:);

numeric_data = cell2mat(data(:,[1 2 3 4 6]));

handles.v_thickness_1 = numeric_data(:,1); % numeric
handles.v_thickness_2 = numeric_data(:,2);
handles.h_thickness_1 = numeric_data(:,3);
handles.h_thickness_2 = numeric_data(:,4);
handles.amplitude = data(:,5); % cell array of char vectors
handles.v_or_h_array = numeric_data(:,5);
handles.f_df = data(:,7);


handles.exp_counter = 1;
set(handles.text_exp_counter,'String',num2str(handles.exp_counter));

handles.region1 = [];
```

```
% Create the Arduino serial object
handles.arduinoObj = serialport('COM3', 38400);
configureTerminator(handles.arduinoObj,'CR/LF');
%
for i=1:8
   handles.message = readline(handles.arduinoObj);
   disp(handles.message)
end

create_patch(handles);



% Update handles structure
%guidata(hObject, handles);

% UIWAIT makes Hapticfinal wait for user response (see UIRESUME)
% uiwait(handles.finger);

function create_patch(handles)
if ishandle(handles.region1)
   delete(handles.region1);
end
v_or_h = handles.v_or_h_array(handles.exp_counter);
if v_or_h == 0 % Vertical line
   v_thick1 = handles.v_thickness_1(handles.exp_counter);
   v_thick2 = handles.v_thickness_2(handles.exp_counter);
   handles.region1 = patch( ...
      'Parent',handles.axes1, ...
      'XData',[v_thick1 v_thick2 v_thick2 v_thick1], ...
      'YData',[-10 -10 10 10], ...
      'FaceColor','red');
    set(handles.axes1,'XLim',[-5 0],'YLim',[-10 10]);
% set(handles.axes1,'XLim',[-10 10],'YLim',[-10 10]);
else % Horizontal line
   h_thick1 = handles.h_thickness_1(handles.exp_counter);
   h_thick2 = handles.h_thickness_2(handles.exp_counter);
   handles.region1 = patch( ...
      'Parent',handles.axes1, ...
      'XData',[-10 10 10 -10], ...
      'YData',[h_thick1 h_thick1 h_thick2 h_thick2], ...
      'FaceColor','red');
    set(handles.axes1,'YLim',[0 5],'XLim',[-10 10]);
% set(handles.axes1,'XLim',[-10 10],'YLim',[-10 10]);
end
set(handles.axes1,'XGrid','on','YGrid','on');
```

```
axis(handles.axes1,'equal');
% Update handles structure
guidata(handles.finger,handles);
% call the button motion fcn to update the new patch's color:
finger_WindowButtonMotionFcn(handles.finger);



% UIWAIT makes franck_guide wait for user response (see UIRESUME)
% uiwait(handles.finger);


% --- Outputs from this function are returned to the command line.
function varargout = franck_guide_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VAR ARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Start_button.
function Start_button_Callback(hObject, eventdata, handles)
% hObject    handle to Start_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.fileID = fopen('Start_Stop.txt','w');
handles.t = timer('ExecutionMode', 'fixedRate', ...
    'Period', 0.5, ...
    'TasksToExecute', Inf, ...
    'TimerFcn', {@timerCallback, handles.finger});
start(handles.t);
set(handles.Start_button,'Enable','off'); % -> Disable the button
guidata(hObject,handles);% -----> do this to save the updated handles object


function timerCallback(~,~,f)
handles = guidata(f);
 %fprintf(fileID,'(X, Y, time) = (%g, %g, %s)\n', get(0, 'PointerLocation'),
datetime('now'));
 fprintf(handles.fileID,'(X, Y, time) = (%g, %g, %s, %s, %s)\n', get(0,
'PointerLocation'), second(datetime('now')), ...
    handles.amplitude{handles.exp_counter},handles.f_df{handles.exp_counter});
%fprintf('calling timer callback\n');
```

```
% --- Executes on button press in Next_button.
function Next_button_Callback(hObject, eventdata, handles)
% hObject    handle to Next_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)uiconfirm(handles.UIFigure,'Are You sure?','Confirm Close',...
handles = guidata(hObject);

new_counter = handles.exp_counter + 1;

if new_counter > numel(handles.v_thickness_1)
  msgbox('Experiment is Done!','DONE');

   return
end
f = msgbox('Operation Completed','NEXT');
 set(handles.text_exp_counter,'String',num2str(new_counter));
% drawnow();
handles.exp_counter = new_counter;
% delete the old patch and create a new one:
create_patch(handles);

% --- Executes on button press in Yes_button.
function Yes_button_Callback(hObject, eventdata, handles)
% hObject    handle to Yes_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

answer = questdlg('Did You Feel it?', ...
   'Vibration Feeling', ...
   'YES','NO','CANCEL ','CANCEL ');
% Handle response
switch answer
   case 'YES'
      fileID= fopen('Vibr_ans.txt2','a');
      YES = "I FEEL IT";
      fprintf (fileID, "(%s,%s)\n",YES,handles.f_df{handles.exp_counter});
      fclose(fileID);
      %disp([answer ' yes i feel it.'])
      %Yes_answer = 1;
   case 'NO'
      fileID= fopen('Vibr_ans.txt2','a');
      NO = "I DON'T FEEL IT";
      fprintf (fileID, "(%s,%s)\n",NO,handles.f_df{handles.exp_counter});
```

```
        fclose(fileID);
        %disp([answer ' no i don"t feel it.'])
        %No_answer = 2;
    case 'CANCEL'
        disp('cancel')
        Cancel = 0;
end



% --- Executes on button press in Stop_button.
function Stop_button_Callback(hObject, eventdata, handles)
% hObject    handle to Stop_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles = guidata(hObject);
stop(handles.t) %whenever we want to stop.
fclose(handles.fileID);
set(handles.Start_button,'Enable','on'); % -> Enable the button
guidata(hObject,handles);

% --- Executes on mouse motion over figure - except title and menu.
function finger_WindowButtonMotionFcn(hObject, eventdata, handles)
% hObject    handle to finger (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles = guidata(hObject);
pos = get(hObject, 'currentpoint'); % get mouse location on figure
global x;
global y;
x = pos(1);
y = pos(2); % assign locations to x and y
set(handles.xloc, 'string', ['x loc:' num2str(x)]); % update text for x loc
set(handles.yloc, 'string', ['y loc:' num2str(y)]); % update text for y loc
% Determine if mouse is within the region

p_x = get(handles.region1,'XData');
p_x = p_x([1 2]);
p_y = get(handles.region1,'YData');
p_y = p_y([1 3]);

ax_xl = get(handles.axes1,'XLim');
ax_yl = get(handles.axes1,'YLim');

ax_units = get(handles.axes1,'Units');
if ~strcmp(ax_units,'pixels')
```

```
    set(handles.axes1,'Units','pixels')
end
ax_pos = get(handles.axes1,'Position'); % axes1 position in pixels
if ~strcmp(ax_units,'pixels')
    set(handles.axes1,'Units',ax_units);
end

% convert the patch XData and YData from axes coordinates to figure coordinates in
pixels
p_x = (p_x-ax_xl(1))/(ax_xl(2)-ax_xl(1))*ax_pos(3)+ax_pos(1);
p_y = (p_y-ax_yl(1))/(ax_yl(2)-ax_yl(1))*ax_pos(4)+ax_pos(2);

if x >= p_x(1) && x <= p_x(2) && y >= p_y(1) && y <= p_y(2)
    set(handles.region1,'FaceColor','g');

        writeline(handles.arduinoObj, handles.amplitude{handles.exp_counter})
else
    set(handles.region1,'FaceColor','r');
    writeline(handles.arduinoObj, '0&1!')
end
```

APPENDIX E: CODE USED TO DETERMINE THE TIME FROM EACH PARTCIPANT FROM EXCEL IN MATLAB

```
tT=readtable('EXPERiMENT_SU88.xlsx','Sheet',9,'Range','A5066:A5213','ReadVariableNames',0);
tT.Var1=strtrim(extractAfter(tT.Var1,'='));
tT.Var1=strtrim(extractBetween(tT.Var1,'(',')'));
T=split(tT.Var1,',');
T=str2double(T(:,3));
nMin=sum(diff(T)<0);
TElapsed=duration(0,0,60*nMin+T(end)-T(1))
```