הטכניון - מכון טכנולוגי לישראל
TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY
הפקולטה להנדסת חשמל
המעבדה לבקרה רובוטיקה ולמידה חישובית

Control Robotics and Machine Learning Laboratory

דוח פרויקט: סמסטריאלי – אביב תשפ"ג

נושא הפרויקט: הנעת יד רובוטית באמצעות חיישן גלי מוח

מגישים:
דוד מולין 213012073
זוהר מילמן 213784549

מנחה:
קובי כוחיי

תוכן עניינים

תקציר

קטיעת איברים בכלל וידיים בפרט עלולה להיגרם כתוצאה מתאונת עבודה, תאונת דרכים, פציעה בלחימה או באופן מולד. ישנם פתרונות שונים הנותנים מענה לקטועי הידיים על מנת לאפשר להם להתנהל כרגיל ככל הניתן. בשוק קיימים מוצרים שונים, ביניהם יקרים, כבדים ומסורבלים לשימוש יום-יומי. כמו כן, בין הפתרונות השונים מוצעים גם דרכי הנעת הזרוע באמצעות איברים שונים, ביניהם שרירי הזרוע הקטועה, אשר אינה מועדפת עבור נכים הסובלים מכאבי פנטום. הפרויקט שלנו מנסה לתת מענה לחסרונות אלו.

בעבר פותחה בטכניון על ידי סטודנטית לתואר שני - זרוע רובוטית קלת משקל ובעלת מחיר ייצור נמוך. הפיתוח בשיתוף עמותת "Haifa3D" המספקת מוצר זה ללא עלות לכל קטוע יד שפונה לעמותה בהתאמה אישית למידותיו ובהתאם לצרכיו האישיים.

הפרויקט המוצג נעשה בשיתוף פעולה עם העמותה ומטרתו היא הנעת הזרוע הרובוטית באמצעות אותות שונים המופקים בעזרת חיישן גלי מוח מסוג MUSE. לחיישן קיימים מספר סוגים שונים של אותות הניתנים להפקה ממנו, ומטרתנו בפרויקט זה היא מציאת האותות הנוחים ביותר לשימוש מתוכם, והרכבת טריגרים מהם, שיוכלו לשלוט בתנועת היד. הפרויקט כולל פיתוח מקיף של אפליקציה, המאפשרת חיבור אלחוטי הן ליד והן לMUSE, ומבצעת את הקישור בניהם ואת עיבוד האותות הדרוש.

מבוא

<u>תיאור כללי של הפרויקט:</u>

בפרויקט אנו משתמשים באפליקציה שפיתחנו, אשר מתחברת לחיישן הMUSE, אשר נענד על ראש המשתמש, וקולטת ממנו מידע על אירועי מצמוץ ואירועי נעילת לסת של המשתמש. המערכת מסננת את האותות הללו, ומפיקה מהם טריגרים, באמצעותם היא שולחת פיקודים בפרוטוקול הבלעדי של היד הרובוטית, אליה האפליקציה גם מחוברת. תנועות היד האפשריות הן סיבוב כף היד, ותנועה של כל אחת מבין 4 אצבעות (אין שליטה על האגודל מכיוון שהוא מכני). כל שיש על המשתמש לבצע מבחינת כיול ראשוני, הוא הורדת האפליקציה (המתאימה למכשירי אנדרואיד), הדלקת הMUSE ולבישתו כראוי, והתחברות עם האפליקציה אל הMUSE ואל היד.

<u>פרוטוקול Bluetooth Low Energy (BLE)</u>

BLE (Bluetooth Low Energy) הוא תקן תקשורת אלחוטי שפותח על ידי קבוצת Bluetooth Special-Interest Group (SIG) ונועד לתקשורת קצרת טווח בין מכשירים שונים. הוא מבוסס על טכנולוגיית Bluetooth, אך תוכנן במיוחד לשימוש במכשירים הדורשים צריכת אנרגיה נמוכה, כגון חיישנים, שעונים חכמים, ציוד רפואי, ואביזרי כושר. היתרון המרכזי של BLE הוא צריכת האנרגיה הנמוכה שלו, שמאפשרת למכשירים לעבוד על סוללות קטנות לאורך זמן ממושך, לעיתים חודשים ואף שנים, מבלי להחליף סוללה, והוא עדיין מספק טווח עבודה של עד כמה עשרות מטרים, ומאפשר חיבור בין מכשיר אחד למספר מכשירים בו זמנית.

התקשורת בפרוטוקול BLE מתבצעת במבנה של Client-Server כאשר מכשיר אחד משמש כלקוח (Client) והשני כשרת (Server). הנתונים מועברים בין המכשירים באמצעות מנגנון של קריאה (Read) כתיבה (Write) והתראות (Notifications).

המבנה הלוגי של הנתונים ב BLE מבוסס על מושגים של Service ו Characteristic:

- Service - קבוצת Characteristics שמאגדת בתוכה נתונים הקשורים לנושא או פונקציונליות מסוימת. ה Service יכול להיות סטנדרטי או מותאם אישית.
- Characteristic: יחידת המידע הבסיסית שניתן לקרוא, לכתוב או לקבל עליה התראה בפרוטוקול BLE. כל Characteristic מכיל נתון מסוים, כמו למשל כמות הסוללה הנשארת, והוא מזוהה על ידי מזהה ייחודי שנקרא UUID. מאפיינים יכולים להיות קריאים, ניתנים לכתיבה, או נתמכים בהתראות.

תיאור מפורט

<u>תהליך העבודה:</u>

שלב א- בחירת בקר ולמידת סביבת העבודה:

בחרנו בפרויקט שלנו להשתמש באפליקציית אנדרואיד בתור הבקר שלנו, באמצעותו אנו מתחברים אל הMUSE ואל היד.
הסיבות העיקריות לבחירה זו היו:

- זמינות: לרוב המוחלט של המשתמשים ישנה גישה למכשיר טלפון בו הם משתמשים ביום יום
- קלות משקל: השימוש בטלפון אומר שאין צורך להוסיף שום משקל נוסף על היד עצמה (כמו במקרה של מיקרו בקר), דבר שעשוי להקל על המשתמשים ארגונומית.
- צריכת הספק נמוכה: החיבור הן ליד והן לMUSE מתבצע בפרוטוקול BLE (Bluetooth low energy), מה שממזער את צריכת ההספק של האפליקציה.
- תכנות קל מחדש: ניתן לגשת לקוד האפליקציה ולערוך אותו מכל מחשב עם תוכנת android studio, וניתן כך להתקין מחדש ולעדכן את האפליקציה בטלפון בקלות

שלב ב- קבלת קוד לגישה לMUSE:

בשלב הראשון בפרויקט היה עלינו להבין מהן היכולות של חיישן זה ואיזה סוג מידע נוכל לדלות מתוכו. גילינו כי השימוש הסטנדרטי במכשיר זה הוא בכלל לצורך ניטור עומק של מדיטציה או שינה, והוא לא תוכנן לתת מידע יותר מדי שימושי לצרכי Brain BCI (computer interface). בכל זאת ראינו כי היה קיים סט תוכנות המיועד למפתחים בעזרתן ניתן לקבל גישה לכל המידע שהחיישן אוסף, ובכל זאת להפיק ממנו מידע שימושי. בכל זאת נתקלנו בבעיה, שכן גרסת המכשיר שהייתה בידנו, MUSE 2, הייתה גרסה יחסית ישנה של המכשיר, ומסתבר שהתוכנות המתאימות לגרסה זו של המכשיר כבר נגרטו והתמיכה בהן הופסקה. נאלצנו נפנות באופן ישיר לתמיכה הטכנית של החברה שמייצרת את המכשיר הזה בתקווה שנוכל לקבל גישה לקבצי קוד רלוונטיים שיוכלו לעזור לנו כמפתחים להתעסק עם המכשיר. למזלנו, פניתינו נענתה, וקיבלנו לקוד דוגמא של אפליקציה שיכולה להתחבר למכשיר ולהוציא ממנו מידע. קוד זה היווה את הבסיס התכנתי של הפרויקט שלנו החל משלב זה.

שלב ג- דליית מידע מתוך הMUSE:

עכשיו כשהייתה לנו גישה לDATASTREAM מתוך המכשיר, יכולנו לראות שהמידע אותו אנו יכולים לקבל הוא:

- מידע ערוצי EEG טהור
- מידע על מהירויות זוויתיות בשלושה צירים (מתקבל דרך גירוסקופ)
- מידע על תאוצות קוויות בשלושה צירים (מתקבל דרך מד תאוצה)

- מידע האם התרחש מצמוץ
- מידע האם התרחשה נעילת לסת

כיוון המחשבה הראשוני שלנו היה האם נוכל להשתמש במידע הEEG כדי לקבל אינדיקציה על תנועות מוטוריות כלשהן. לצערנו גילינו שמכשיר הMUSE ממקם את האלקטרודות במקומות על הראש שאינן קרובים לאזורים המעידים על פעילות מוטורית, ולכן הבנו שלא נצליח להפיק מידע שימושי לנו מערוצי הEEG הטהורים.
כיוון המחשבה הבא שלנו היה ניצול הגירוסקופ ומד התאוצה לקבלת מידע על תנועות ראש. התחלנו לכתוב קוד שמאפשר לנו לקבל את הנתונים הללו ולקבל מתוכם טריגרים, למשל של הסטת הראש ימינה או שמאלה, אך לבסוף לא המשכנו בכיוון הזה. הסיבה לכך הייתה שבעקבות שיחה עם קטועי יד, הבנו שהם לא מעוניינים כרגע בשליטה ביד באמצעות תנועות ראש, ומבחינתנו אם המשתמשים של הפרויקט הזה לא מעוניינים בשליטה באופן הזה, אין לנו סיבה להמשיך. לבסוף, בחנו את האפשרות להשתמש במצמוצים ונעילות לסת בתור מקור המידע שלנו.
בסופו של דבר, מידע מהסוג הזה מכונה Artifacts, ולרוב הוא מייצג הפרעות לאותות הEEG הרגילים, אבל במקרה שלנו דווקא אותו ה"רעש" התגלה להיות המידע המעניין מבחינתנו. לאחר עבודת תכנות לא פשוטה הבנו כיצד ניתן לגרום לMUSE לשדר את המידע הזה (לא היה קל לגרום לו לשדר את המידע הזה, שכן הוא אינו חלק מהמידע הסטנדרטי המשודר, היום ומבחינת הMUSE מדובר ב"רעש"), וראינו שאכן נוכל לראות באופן עקבי שכאשר אנו ממצמצים או נועלים את הלסת האפליקציה מציגה את זה, אם כי ראינו גם שהיו False Positives, ולכן נדרשנו לבצע עיבוד על האותות הללו.

שלב ד- עיבוד אותות הMUSE:

על מנת להקטין את מספר הFalse Positives שלנו, היינו צריכים לבצע סוג מסוים של עיבוד על האות כדי שהוא יוכל לסנן מקרים בהם יש הופעה רגעית של אירוע מצמוץ או נעילת לסת. כמובן שאותו הסינון חייב להיות סיבתי, היות ואנו נדרשים לבצע אותו בזמן אמת. על מנת למזער את זמן החישוב, בחרנו במימוש יחסית פשוט של LPF המיוצג בעזרת משוואת ההפרשים הבאה:

$$y[n] = a * y[n-1] + (1-a) * x[n]$$

כאשר אנחנו ממירים את האות הבוליאני שלנו לאות בינארי $x[n]$, ואנו קובעים שמתקיים מצמוץ כאשר

$y[n] \geq b$, וa, b אלו פרמטרים אשר נקבעים בשלב זה ידנית כך שנקבל תוצאות יציבות כרצוננו. סוג סינון זה קרוי Exponential smoothing.[1] גרפים להמחשת הסינון מופיעים בנספחים.

שלב ה – למידת הזרוע ושליחת פקודות:

[1] Exponential Moving Average

בשלב זה למדנו את העבודה מול הזרוע הרובוטית הקיימת. שליחת הפקודות מתבצעת בתקשורת אלחוטית ובאמצעות פקודות שהוגדרו מראש בזרוע המבצעות פתיחה או סגירה של האצבעות ומבצעות תנועה סיבובית של כף היד. בשלב זה היינו צריכים גם לכתוב באפליקציה שלנו מודול אשר אחראי על התחברות ותקשורת עם היד.

שלב ו – קביעת טריגרים:

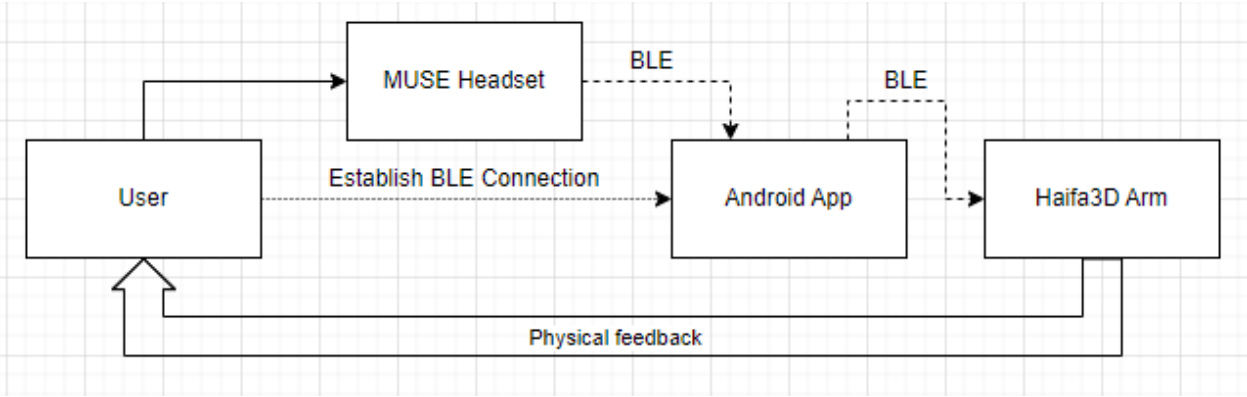
בסופו של דבר, היינו צריכים לקבוע כיצד נתרגם את האותות המסוננים מהחיישן לתנועות יד. אחרי התייעצות, הגענו למסקנה כי ישנן שלוש פעולות עיקריות שאנו צריכים להיות מסוגלים לעשות עם היד:

- נעילת היד במצבה הנוכחי\הסרת נעילה
- פתיחת\סגירת כל היד
- סיבוב כף היד

לצערנו כשבדקנו את הפעולות שיד יכולה לבצע, נראה שהייתה בעיה כלשהי בחומרה או בקוד הפנימי של היד, שמנעה מהסיבוב לעבוד כראוי, ולכן בעל כורחנו ויתרנו על פעולה זו. נותרנו עם שתי הפעולות הראשונות, כאשר עבור נעילת מצב היד, או הסרת הנעילה, בחרנו בטריגר של שלושה מצמוצים מהירים ברצף. עבור פתיחת וסגירת היד, בחרנו בטריגר של נעילת לסת פשוט, שכמובן פועל רק בזמן שהיד לא במצב נעול.

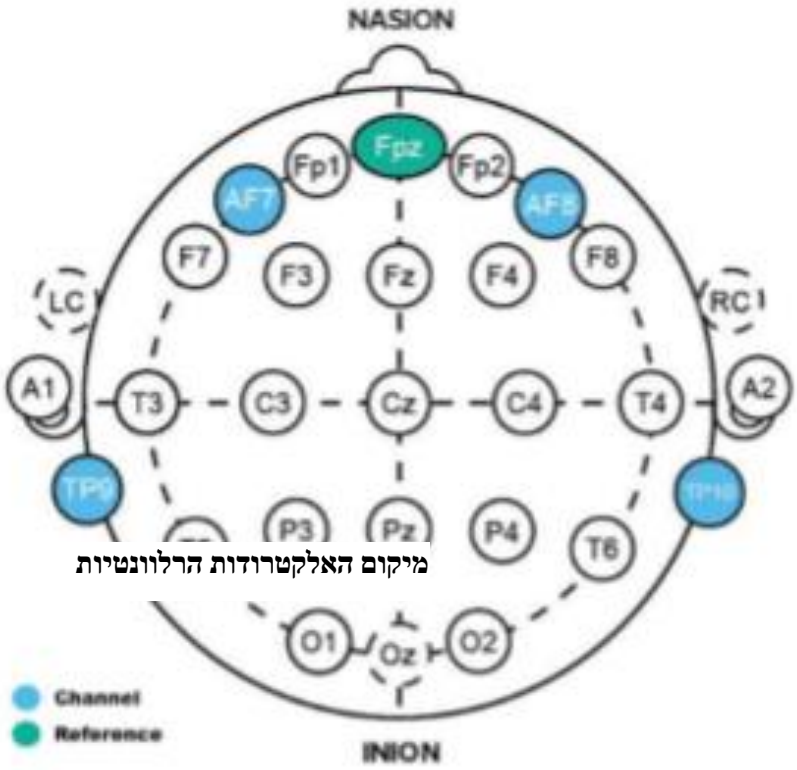
שלב ז – אינטגרציה ובניית אפליקציה סופית

<u>סכמת בלוקים כללית:</u>



תיאור הומרתי

חיישן ה - Muse מבית חברת Interaxon הינו התקן EEG המודד את פעילותו החשמלית של המוח. המכשיר במקור משמש לניטור אחר איכות המדיטציה הנעשית ע"י המשתמש שחובש את החיישן דרך אפליקציה ייעודית בטלפון הנייד.



מיקום האלקטרודות הרלוונטיות

המדידה נעשית באמצעות 7 חיישני אלקטרודות יבשות הפועלים בו זמנית ומראים ארבעה ערוצי מידע של פעילות מוחית:

$TP9$: ממוקם מעל אוזן שמאל.

$TP10$: ממוקם מעל אוזן ימין.

$AF7$: ממוקם בצד שמאל של המצח.

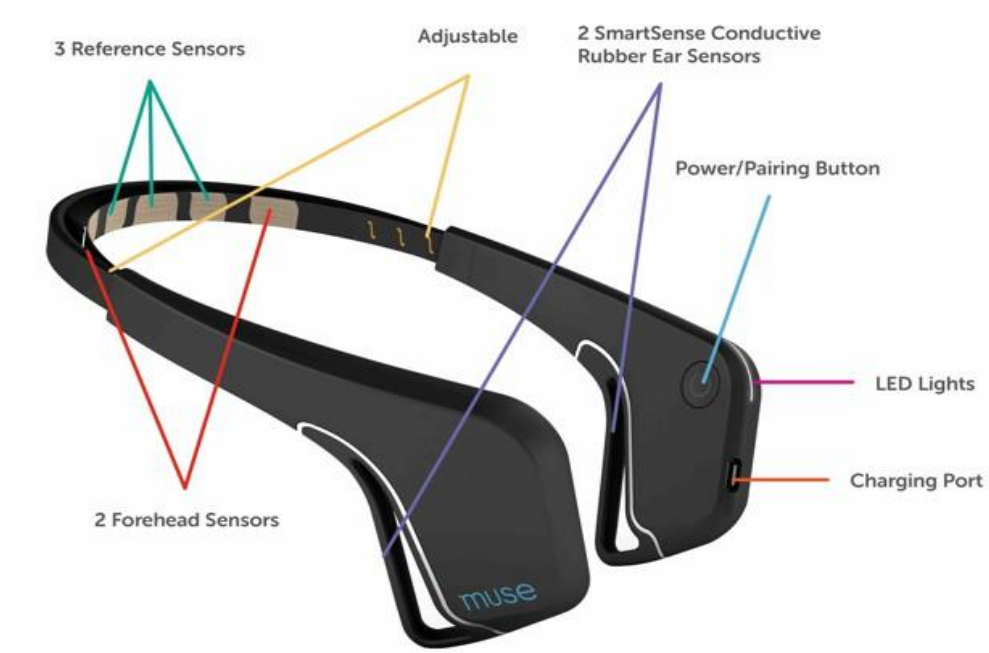
$AF8$: ממוקם בצד ימין של המצח.

REF: 3 אלקטרודות הממוקמות במרכז המצח ויוצרות מעגל DRL[2] שמטרתו היא ביטול הCommon Mode Interference.

בנוסף, קיימים ל muse חיישן gyro (חיישן אוריינטציה) וחיישן accelerometer (חיישן תאוצה), אך בהם לא נעשה לבסוף שימוש לטובת הפרויקט.

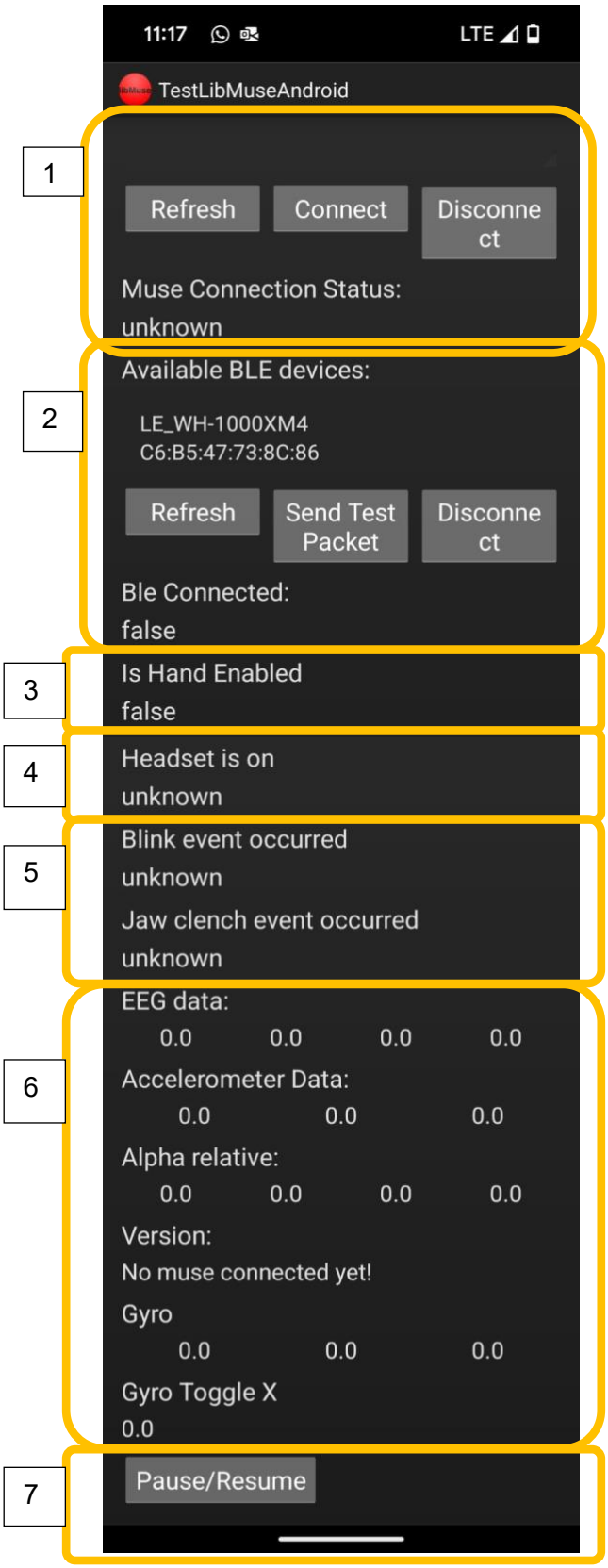
[2] "Improving Common-Mode Rejection Using the Right-Leg Drive Amplifier"

למכשיר קיימת סוללה המספיקה לעבודה של כ- 5 שעות רצופות והתקשורת עם הרצועה מתבצעת באמצעות חיבור BLE



מכשיר ה-MUSE

תיאור ממשק האפליקציה



להלן מצורף צילום מסך של האפליקציה (במצב בו היא אינה מחוברת לMUSE), כאשר חלקי האפליקציה השונים על המסך הם:

1. חלק זה אחראי על החיבור לMUSE, כאשר אם הMUSE דולק ונמצא על יד הטלפון, לחיצה על כפתור הrefresh תראה על המסך את שם המכשיר, וניתן יהיה להתחבר אליו ולהתנתק ממנו עם הכפתורים המתאימים.
2. חלק זה אחראי על החיבור אל היד, כאשר נראה בצילום המסך שאנו מחוברים להתקן BLE כלשהו (אילו זו הייתה היד היינו רואים כחלק מהשם את המילים Haifa3D). כאשר לא מחוברים להתקן, מוצגת רשימה של כל ההתקנים הזמינים, ובלחיצה על אחד מהם האפליקציה תתחבר אליו כמו שנראה בתמונה. ניתן לעשות Refresh לחיבור, להתנתק מהמכשיר, או לשלוח Test packet, מה שרלוונטי רק במקרה של היד, ומה שזה עושה זה פשוט לשלוח את כל הPresets של היד ברצף כדי לראות שהיא מגיבה כראוי.
3. חלק זה מציג משתנה בוליאני, שאומר האם היד כרגע נעולה או לא (ניתן לנעול או לבטל נעילה בעזרת שלושה מצמוצים).
4. חלק זה מציג משתנה בוליאני שאומר האם הMUSE נלבש כמו שצריך. אם הוא מראה FALSE זה אומר שצריך להדק את הMUSE ולוודא שכל האלקטרודות במגע עם הראש.
5. חלק זה מציג האם המשתמש מצמץ בו נעל את הלסת בעזרת משתנים בוליאניים מתאימים, כאשר מדובר כבר באות המסונן.
6. פה מוצג מידע נוסף שהMUSE קולט שאינו רלוונטי לפרויקט בתצורתי הנוכחית, והוא כולל מידע EEG, מספר גרסה של הMUSE, מידע מהאקסלרומטר והג'יירוסקופ, ומידע האם התבצעה תנועת ראש מהירה הצידה (Gyto Toggle X).

כל המידע הזה אינו בשימוש בפרויקט זה, אך הוא חלק מהאפליקציה והוא עשוי להיות שימושי לפרויקטים עתידיים.

7. זהו פשוט כפתור עצור\המשך לכל התהליכים של האפליקציה. הוספנו את הכפתור הזה כדי שיהיה לנו כפתור עצירה חיצוני מעבר לטריגרים ולכיבוי פיזי של הMUSE, שמאפשר עצירה פשוטה של השליטה ביד דרך הMUSE מבלי לנתק את המכשירים. כפתור זה יכול להיות שימושי בהקשר של שילוב השליטה ביד עם הMUSE עם שליטה ביד באמצעים אחרים (כמו כתף או כף רגל) ואפשר כיבוי\הדלקה נוחים של מצב שליטה זה.

תיאור תוכנתי

עיקר העבודה בפרויקט שלנו בסופו של דבר הייתה כתיבתה של האפליקציה הדרושה. לא היה לנו כלל ניסיון קודם בתכנות אפליקציות לאנדרואיד, ולכן חלק ניכר ממאמצינו כלל גם למידה של שפת JAVA וכיצד בנויות אפליקציות. האפליקציה הסופית שלנו גדלה בסופו של דבר לנפח גדול מאוד של קוד (מעל 1500 שורות), שכן חלק הארי בו הוא תשתיות קוד שאינן מכילות אלגוריתמיקה הקשורה בפרויקט.
בכל זאת, נוכל לומר כי ישנם שני חלקים עיקריים בקוד שלנו:

- קוד הדואג לחיבור לMUSE, דליית המידע ממנו באופן שוטף, ביצוע עיבוד אות ומעקב אחר הטריגרים ושליחתם ליד
- קוד הדואג לסריקת התקני BLE כלליים (לרבות היד), ומאפשר התחברות אליהם, וכן קובע את שליחת הפיקודים להיות בפרוטוקול התקשורת שהיד מצפה לקבל.

<u>מבנה הקוד הנוגע בMUSE:</u>

הקוד מחולק למספר חלקים עיקריים:

1: חיבור BLE לMUSE-

כאשר מתבצעת לחיצה על כפתור החיבור לMUSE, האפליקציה מחפשת את הMUSE (על ידי פונקציה פנימית של הSDK שלא נכתבה על ידינו), ומתחברת אליו. אחד מהפרמטרים החשובים שיש להגדיר בעת ההתחברות לMUSE הוא איזה סוג של מידע אנו מעוניינים לדלות מתוכו בתקשורת שוטפת, שמתנהלת אסינכרונית עם שאר התהליכים באפליקציה. במקרה שלנו, חשוב להגדיר שאנו מעוניינים לקבל את המידע שמוגדר בMUSE כARTIFACTS, שמכילות את המידע על מצמוצים ונעילות לסת.

2: קבלת המידע מהMUSE -

הגדרנו בקוד את הפונקציה recieveMuseArtifactPacket, אשר נקראת אוטומטית בכל פעם שהMUSE משדר מידע. בפונקציה זו אנו מקבלים את המידע על האם הMUSE נלבש כראוי, וכן האם התרחש מצמוץ והאם התרחשה נעילת לסת. אנו שומרים בקוד משתנים גלובליים המייצגים את הממוצעים האקספוננציאליים שלנו של ערכי המצמוץ ונעילת הלסת, ומיד אחרי קבלת הערכים החדשים אנו מבצעים את הסינוס המעריכי, ואם הוא עולה על הרף שקבענו, נאמר שבאמת התרחש מצמוץ או נעילת לסת בהתאמה. באותה הפונקציה נבדוק גם האם הטריגרים מתקיימים ונשלח פיקודים ליד בהתאמה.

3: בדיקת טריגרים –

בדיקת שני הטריגרים שלנו מתבצעת בבצורה הבאה:

- עבור פתיחה או סגירה של היד, נרצה שנעילת לסת תהיה הטריגר. בכל זאת, איננו יכולים סתם ככה לומר שברגע שנעילת לסת היא TRUE לבצע פתיחה או סגירה של היד, והסיבה לכך היא שהMUSE משדר מידע כל כמה עשרות מילי שניות, ועל כן יהיה לנו רצף של כמה פקטות שערכן יהיה TRUE. כדי להימנע מרצף של טריגרים, הגדרנו פשוט פרמטר של זמן חוסר תגובה T. כלומר, ברגע שהייתה נעילת לסת, מופעל הטריגר, ולא נגיב יותר לנעילת לסת במשך

T פקטות. כיוונו את הערך של הפרמטר ידנית כך שהרגשנו שההשהיה היא לא ארוכה מדי, אך עדיין יותר ארוכה ממשך של נעילת לסת קצרה.

- עבור נעילה או הסרת נעילה של היד, קבענו כטריגר שלושה מצמוצים מהירים. כדי לממש את זה, עשינו טיימר שמתחיל לספור ברגע שנקלט מצמוץ אחד, ואם במהלך אותו הטיימר מתבצעים שני מצמוצים נוספים, רק אז נחשיב את הטריגר. כדי לקלוט מצמוצים נוסיף משתנה שמודד שינויים (מעבר בין TRUE לFALSE או ההפך), ואם המתנה הזה מגיע ל4 תוך הזמן של הטיימר, הטריגר ייקלט. בתום בטיימר המונה מתאפס כמובן.

4: הפעלת הזרוע –

גם בתוך פונקציית קבלת החבילות, ברגע שמתרחש טריגר, נשלח פקודה מתאימה ליד. במקרה של פתיחה או סגירה, נעקוב אחרי המצב הנוכחי שלנו, ונשלח את הPRESET שמחליף אותו למצב ההפוך (בפרוטוקול של היד PRESET 0 הוא סגירה ו1 הוא פתיחה). במקרה של נעילת היד, פשוט נגדיר משתנה גלובלי שאומר האם היד נעולה או לא, וכל פעם שיש שלושה מצמוצים, נפעיל על המשתנה היפוך. כדי לבצע את הנעילה, כל מה שנדרש מאיתנו לעשות הוא להתנות את ההפעלה של הטריגר של פתיחה או סגירה, גם בכך שהיד אינה נעולה, בנוסף לשאר הלוגיקה של הטריגר.

<u>מבנה הקוד הנוגע בהתקני BLE כלליים:</u>

כל פונקציונליות השימוש בהתקני BLE כלליים ממומשת תחת המחלקות BleManager ו DeviceAdapter, כאשר BleManager מנהלת את החיבורים והסריקה ו DeviceAdapter מנהלת את עדכון הUI. מחלקה רלוונטית נוספת היא HandCommands, אשר משתמשת באובייקט BleManager שכבר מחובר למכשיר ושולחת פקטות בפרוטוקול של היד לService הרלוונטי של היד. אובייקט בודד מכל אחת מהמחלקות הנ"ל נוצר כאשר האפליקציה נפתחת, ומשומש לאורך כל הLifecycle של האפליקציה.

1: סריקת הסביבה להתקני BLE –

לשם סריקת הסביבה להתקני BLE השתמשנו בBluetoothLeScanner סטנדרטי, אשר מוצע ב android.bluetooth.le.BluetoothLeScanner. כאשר מייצרים אובייקט שכזה, יש לייצר גם אובייקט ScanCallBack, אשר מגדיר לסורק כיצד להתנהג בכל סריקה ואותו אנחנו מייצרים בתוך BleManager. על מנת להתאים את הסורק לצרכינו, ביצענו Override ל3 פונקציות של ScanCallBack, והן OnScanResult, OnBatchScanResult ו OnScanFailed. עבור שתי הפונקציות הראשונות, השינוי שביצענו הוא שליחת המכשירים שהסורק מוצא לאובייקט מסוג DeviceAdapter, על מנת שזה יעדכן את הUI, ועבור השלישית השינוי שביצענו הוא הדפסה של סיבת השגיאה במקרה שהיא מתרחשת.

האפליקציה מתחילה לסרוק מהרגע שהיא נפתחת, ועוצרת ומתחילה מחדש את הסריקה כל 10 שניות. מאחר ואנו מוקפים במכשירי BLE רבים מאוד, האפליקציה מראה את תוצאות הסריקה אשר מפרסמות את שם המכשיר בלבד. כל עוד אין מכשיר מחובר, האפליקציה ממשיכה לסרוק.

2: התחברות למכשיר BLE –

בכל רגע האפליקציה מציגה רשימה של מכשירי BLE אפשריים על המסך, המתודה onDeviceClick בMainActivity מנהלת את תגובת האפליקציה ללחיצה על מכשיר מתוך הרשימה. באופן כללי, התהליך שקורה כאשר לוחצים על מכשיר הוא שהאפליקציה מתנתקת מכל מכשיר שכרגע מחובר אליה, מחברת את אובייקט הBleManager למכשיר הנבחר, מפסיקה לסרוק ומעדכנת את DeviceAdapter כך שיציג את המכשיר המחובר בלבד.

פונקציית ההתחברות למכשיר ממומשת תחת BleManager, והיא משתמשת באובייקט BluetoothGatt אשר מוצע תחת android.bluetooth.BluetoothGatt. אובייקט זה מייצג את הדרך בה שני מכשירים מתקשרים בBLE, ומציע פונקציות שמאפשרות לכתוב ל Characteristics של Services. מתודת connectToDevice של BleManager בעצם מחברת את אובייקט ה BluetoothGatt השייך ל BleManager למכשיר שנבחר.

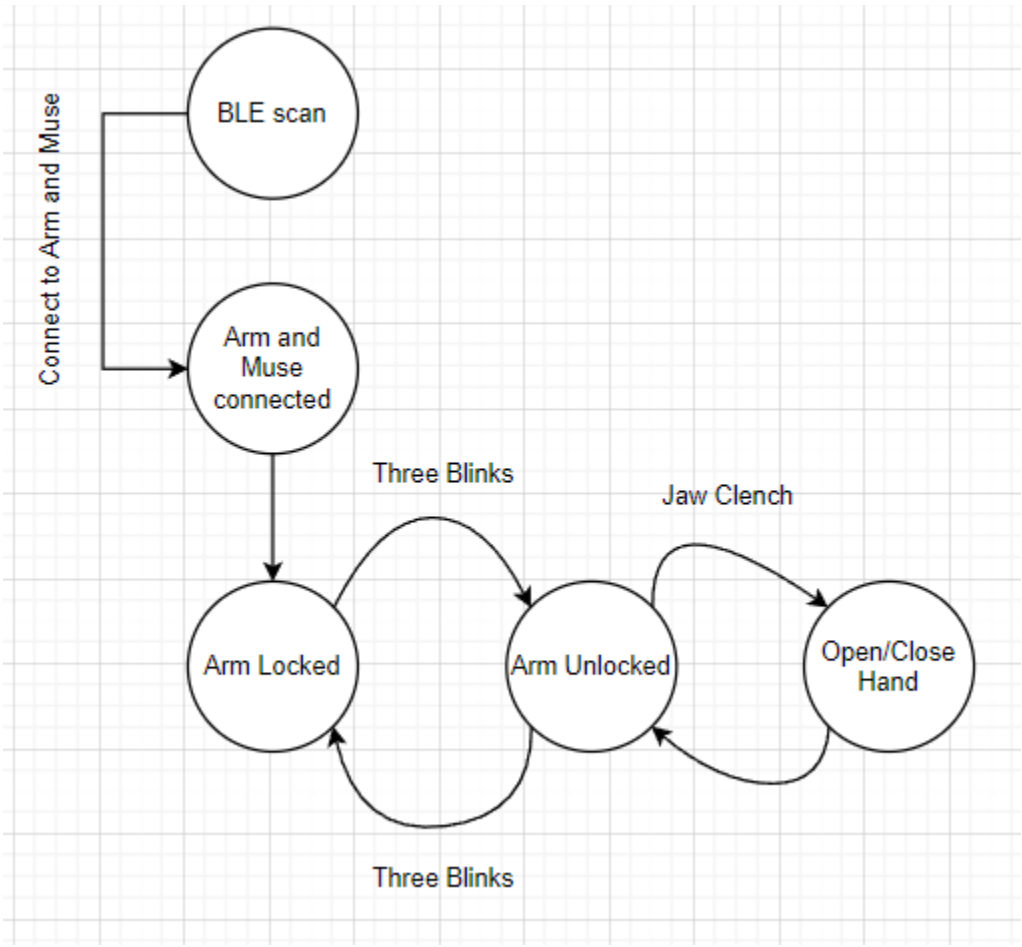
3: שליחת הודעות ליד –

כאשר אנחנו פותחים את האפליקציה, אנחנו מייצרים אובייקט HandCommands אשר מחובר ל BleManager של האפליקציה. מרגע שאנחנו מחברים מכשיר ל BleManager, ניתן לשלוח אליו הודעות דרך HandCommands. האובייקט משתמש בפונקציות של BluetoothGatt (אשר אותו הוא משיג מתוך BleManager) על מנת לכתוב ל Trigger Service של היד את הדרוש. הUUID של הTrigger Service וה Characteristic הוא משתנה קבוע בHandCommands, אשר מקבל את הערכים:

```
private static final UUID triggerServiceUUID =
UUID.fromString("e0198002-7544-42c1-0000-b24344b6aa70");
private static final UUID triggerCharacteristicUUID =
UUID.fromString("e0198002-7544-42c1-0001-b24344b6aa70");
```

אשר לקחנו מתוך הדוקומנטציה של היד.

מכונת המצבים במערכת:



כל המעברים במכונה שאין עליהם כיתוב הם מעברים אוטומטיים. האלגוריתמיקה באמצעותה אנו מזהים את הטריגרים פורטה בסעיף 3 במבנה הקוד הנוגע בMUSE, וחיבור האפליקציה ליד מוסבר לעיל בפירוט מבנה הקוד הנוגע בהתקני BLE כלליים.

תוצאות

להן מצורף סרטון המדגים את הפעלת היד:

WhatsApp Video 2024-08-29 at 7.03.30 PM.mp4

רעיונות לשיפור והמשך מחקר עתידי

חומרה:

- תיקון היד כך שיעבדו PRESETS נוספים מעבר לפתיחת או סגירת כל היד.
- שימוש בגרסה יותר מתקדמת של הMUSE (כגון הMUSE S) בעל אלקטרודות גמישות, כך שיהיה ניתן להרכיב את המכשיר בצורה יותר צמודה, וכן לא יהיה צורך בהידוקו, מה שיקל על חבישתו לקטועי יד.
- שימוש בקורא גלי מוח בעל אלקטרודות באזורים במוח האחראים על מוטוריקה, כך שיהיה אפשר לדלות מידע על תנועות שריר נוספות מגלי הEEG.

תוכנה:

- התוכנה הנוכחית שפיתחנו עובדת כראוי, אך היא כאמור רק אפליקציית אנדרואיד. בשאיפה שלנו היינו שמחים לאפשר גם למשתמשי IOS להנות מהפרויקט שלנו, אך מכיוון שמדובר במערכות הפעלה שונות לחלוטין, הדבר ידרוש שכתוב כמעט מאפשר של כל האפליקציה ולכן אינו נופל במסגרת הפרויקט שלנו.

מחקר:

- פיתוח אלגוריתם בינה מלאכותית לעיבוד אותות המצמוץ ונעילת הלסת לעיבוד הטריגרים, ככל הנראה על ידי איסוף דאטה נרחב של הטריגרים הדרושים ואימון רשת מתאימה (או לחילופין הפעלת שיטות של תכנון ולמידה מחיזוקים).

תודות

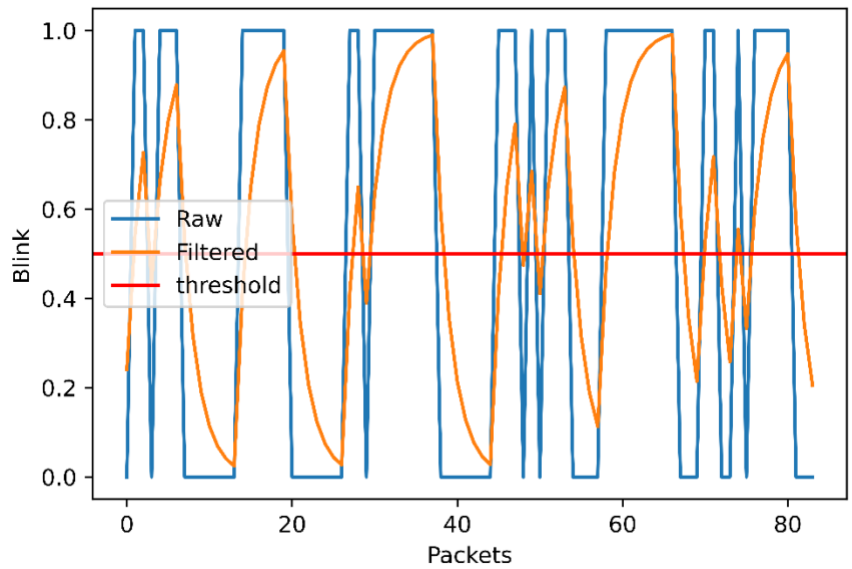
- למר קובי כוחיי - מנחה הפרויקט וראש מעבדת CMRL.
- למעבדת CMRL בפקולטה להנדסת חשמל ומחשבים.
- לעוזי שוורדון ויעקב מלבסקי מעמותת Haifa3D
- לרונית פיסו מהחממה החברתית בטכניון
- לשונית ולבוריס על תמיכה טכנית והחייאת היד כשהיא שבקה חיים
- תודה מיוחדת לסרן א' מחט"ל, שהוכיח לנו שפריצות הדרך הכי חשובות יכולות להגיע מהמקומות הכי לא צפויים.

נספחים

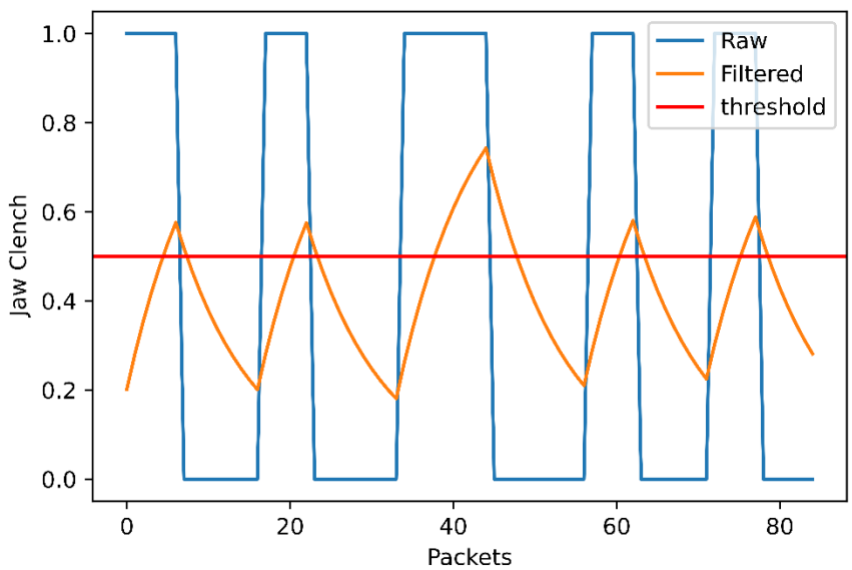
- קוד הפרויקט:

https://github.com/ZoharMilman/Muse-Project-

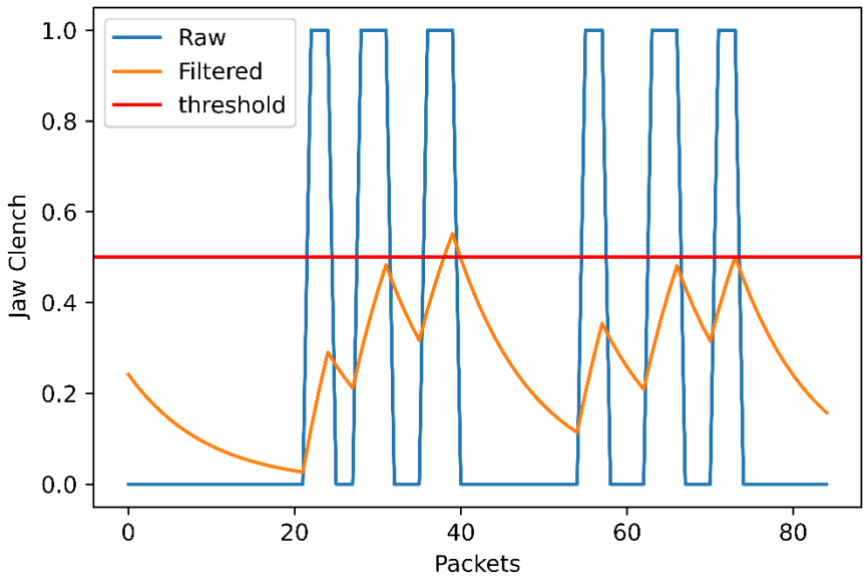
- גרפים להמחשת סינון האותות:



סינון אות המצמוץ עבור $a = 0.6, b = 0.5$



סינון אות נעילת הלסת עבור $a = 0.9, b = 0.5$



סינון אות נעילת הלסת עבור $a = 0.9, b = 0.5$ ניתן לראות כי הסינון פה יותר אגרסיבי מבאות המצמוץ, ולכן איפה שבמצמוץ האות המסונן עולה על הרף, זה לא המצב כאן (וזו הייתה כוונתינו, כי אנו רוצים לקלוט עבור נעילת הלסת פעולה אחת, בעוד שעבור המצמוץ נרצה לקלוט שלושה מצמוצים ברצף)