The Iby and Aladar Fleischman
Faculty of Engineering
Tel Aviv University

הפקולטה להנדסה
ע"ש איבי ואלדר פליישמן
אוניברסיטת תל אביב

מערכות חישה וטלמטריה

פרויקט מס' 22-1-1-2666

דו"ח סיכום

מבצעים:

יונתן אמיר 207021858

יורי לוקץ' 311809867

מנחה:

מר שמחה לייבוביץ אוניברסיטת ת"א

מקום ביצוע הפרויקט:

מעבדת בקרה

תוכן עניינים

רשימת איורים

תקציר

פרויקט גמר זה עוסק בבניית תשתית מבוססת על סביבת הפיתוח **Arduino** עבור פרויקטים עתידיים בהנדסת חשמל בדגש על פרויקטים בתחום הפיזיותרפיה.

פרויקטים רבים שנעשים בתחום ההנדסה מקדישים זמן רב בפיתוח תשתיות בסיסיות לפני שעבודה מתקדמת יותר יכולה להיעשות. דבר זה יוצר, אפוא, עיכוב בפרוייקטים ולכן – מכיוון שהזמן אליהם מוגבל – ירידה פוטנציאלית בתוצר. בפרויקט זה, אנו ניתן מענה לבעיה זו על ידי כך שנספק **תשתית אחידה ורחבה** ככל הניתן בעבור פרויקטים אחרים שיהיו זקוקים לה.

סביבת העבודה והתשתית תכיל את **כניסות הסנסורים השונים ויציאותיהם** ובנוסף לכך חיווים ויזואליים וצלילים. סביבה זו מכילה סנסורים שימושיים לדוגמה, וכמו כן תיעוד שמתאר איך ניתן להרחיב את החומרה והתוכנה לכלול יכולות נוספות.

הקדמה

בפרק זה יתוארו:

- מטרות הפרויקט
- המוטיבציה
- הגישה לפתרון הבעיה
- השוואה כנגד עבודות ואלגוריתמים/מימושים קיימים בנושא

מטרתנו העיקרית היא להקים תשתית מבוססת על סביבת הפיתוח Arduino שתשמש כבסיס למיזמים עתידיים בתחום הנדסת חשמל, עם דגש מיוחד על פרויקטים הקשורים לפיזיותרפיה.

המטרה היא לספק תשתית מוכנה, שחיישניה נבדקו על ידינו, והיא תהיה קלה לשימוש עבור סטודנטים ותשמש כתשתית אחידה לעבודה.

סביבת Arduino מספקת מענה לפרוייקטים הדורשים מיקרו-בקר בעל עצמת עיבוד מוגבלת, דוגמת מכשירי Edge. במקביל, פרוייקט גמר נוסף של צוות אחר עובד על תשתית מבוססת Raspberry Pi, המתאימה לפרוייקטים שדורשים יכולות עיבוד בסדר גודל של מחשב.

אחד העקרונות המנחים של הפרוייקט הוא פשטות ההרחבה שלו. לצורך זה, בחרנו להשתמש במערכת M5Stack. זוהי מערכת אינטגרטיבית המבוססת על ESP32S3, שכוללת חיישנים מסוגים שונים הניתנים לחיבור בצורה פשוטה, ומתאימה מאד לעבודת Prototyping. מעת עתה נתייחס לM5Stack כבקר לצורך פשטות הכתיבה, אך ניתן לזכור כי מדובר במערכת חיישנים, תקשורת, ופלט. ניתן לתכנת על מערכת זו בסביבת Arduino כפי שנכתוב בהמשך.

פיתחנו תשתית קושחה ועבודה מול הבקר M5Stack עבור התממשקות עם החיישנים, למצב עבודה עצמאי (STANDALONE) או מצב עבודה של חיבור MASTER-SLAVE כשהבקר שלנו היננו הslave. פיתוח תשתית Python שמתפקדת כmaster, שניתן לעבוד איתה ב-Windows באמצעות חיבור סריאלי או Wi-Fi ולקבל מחוון מחיישנים או להפעיל חיישני פלט (כמו רמקול). ניתן להתחבר ל-Raspberry Pi באמצעות Wi-Fi. ההסבר על עבודה עם התשתית נמצא בקבצי הפרויקט עצמו, וכיצד ניתן להרחיב את יכולותיה במידת הצורך. יש לציין כי הפרוייקט מכוון לפעולה בSlave Mode, ומצב הפעולה של Standalone הוא מצב דמו.

רקע תיאורטי

בעולם הפרוייקטים, שדורש פיתוח מהיר וחדשנות באופן קבוע, יש חשיבות גדולה לפיתוח תשתיות המאפשרות למהנדסים וחובבים כאחד לפתח את רעיונותיהם. פורצת דרך בנושא זה היתה חברת Arduino האיטלקית, שהוציאה כרטיס פיתוח מבוסס Atmel AVR תחת רשיונות קוד פתוח LGPL וCC. הארדואינו במהרה הביא לרניסנס בעולם באלקטרוניקה החובבנית, מכיוון שהוא אפשר לחובבים רבים גישה לחומרה – דבר שהיה בעבר יקר ומסובך. הארדואינו התבסס על מעבד 8 ביט פשוט וזול, עם יכולות לא מרשימות במיוחד, אבל הוכיח את עצמו כמצליח לאחר שאנשים

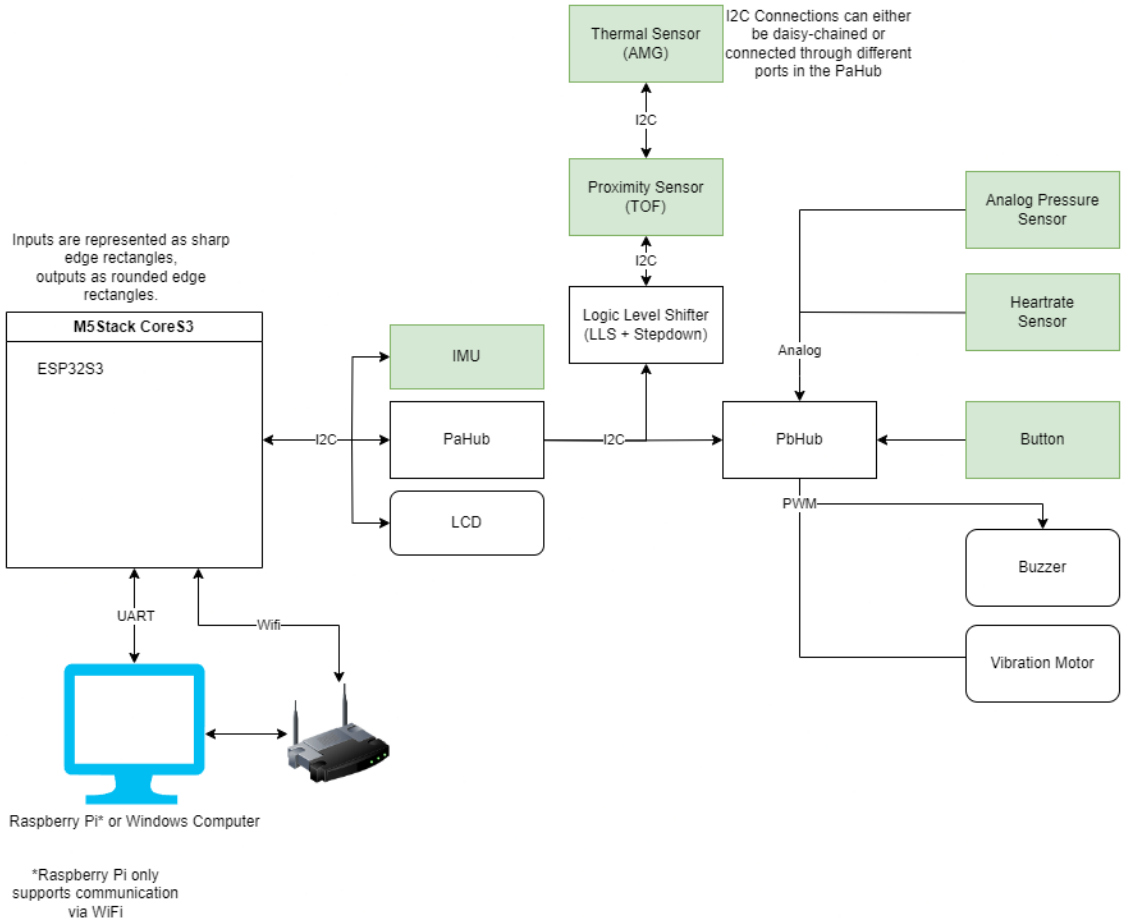
ברחבי העולם גילו שהפרוייקטים שלהם ניתנים למימוש בצורה מינימליסטית, ושקלות השימוש של המערכת פיצתה על היכולות הבסיסיות של המעבד.

מאז כניסתו של הארדואינו לשוק, חברות רבות השתמשו ברשיון הOpen Source של הארדואינו כדי לייצר העתקים זולים יותר, מבוססים על בקרים חזקים יותר, באותו Form Factor כמו הארדואינו (הצורה הבסיסית של הכרטיס). אחת ממשפחות הבקרים הנפוצות לשימוש בהעתקים אלו היא משפחת ESP של חברת Espressif הסינית. מתוך בקרים אלו, הבקר ESP32 הוא בקר מודרני של 32 ביט בעל יכולות מוגברות לאלו של הAVR הקשיש, בעל יכולות WiFi וBluetooth מובנות שאינן דורשות תוספים ושעון כפול של 80MHz ו160MHz. בקר זה אומץ בקהילת החובבים בגלל מחירו הנמוך והיכולות המרשימות שלו, למרות שבתחילת חייו התיעוד של יכולותיו היה פחות מרשים ממקבילותיו המערביות (דוגמת בקרים מתוצרת חברת ST).

הESP32 הוא הבקר בו אנחנו משתמשים, בתוך חבילת M5StackCoreS3. חבילה זו מבוסת על שרשור מרכיבי I2C והרחבתם בעזרת Hubs, והיא עוצבה לנוחות שימוש מירבית בשלב האבטיפוס.

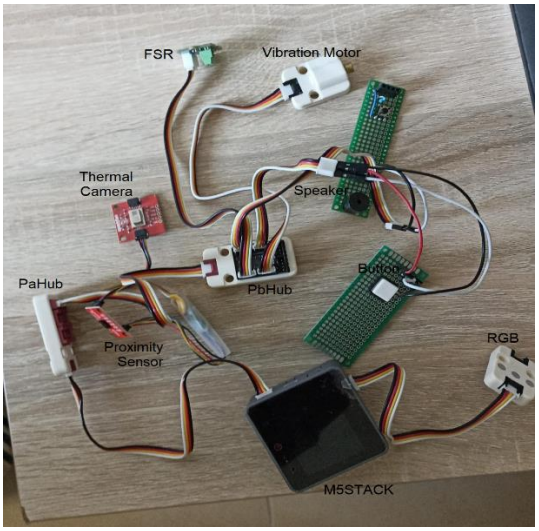
מימוש

מימוש המערכת מפורט בדיאגרמת הבלוקים הבאה.



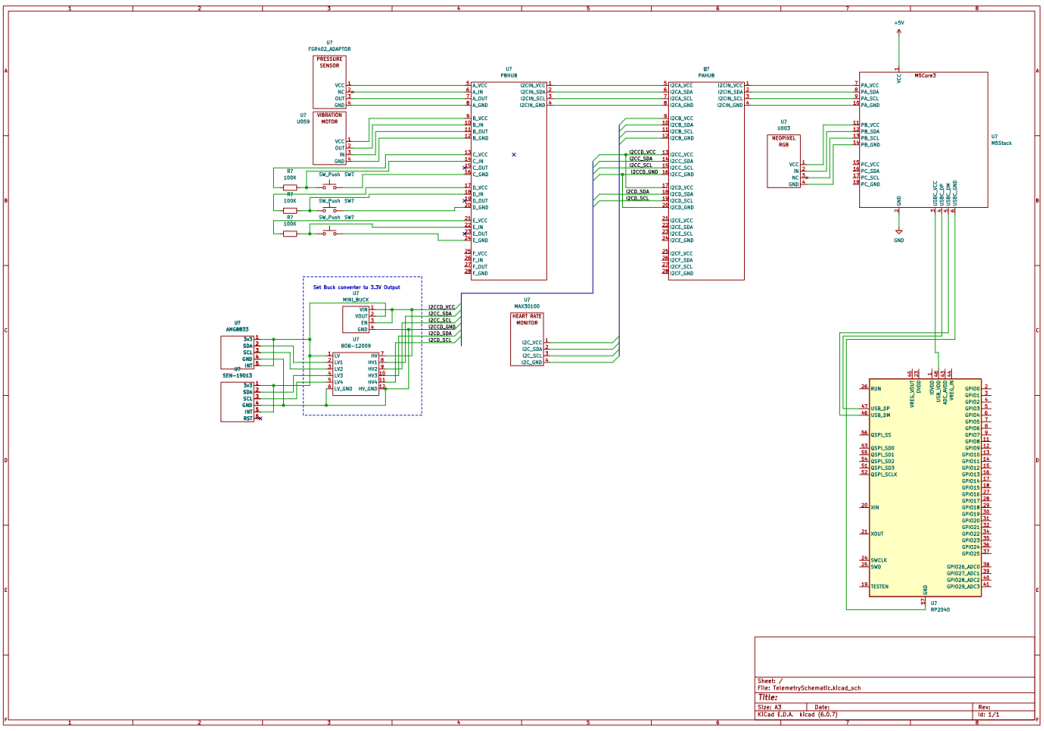
איור 1 –דיאגראמת בלוקים

באיור הבא ניתן לראות את חיבורי המערכת הפיזיים. ניתן לראות כי החיבורים הם מסוג Grove, המאפשרים פרוטוטייפינג פשוט במקרה של שימוש בשאסי אותו יתכנן בעל הפרוייקט. הבקר עצמו ניתן לכינון עם המסך בכיוון המשתמש, או שניתן יהיה להחביא אותו ולחשוף רק פורטים לתקשורת.



איור 2-מבנה המערכת

תיאור חומרה



איור 3- תיאור חומרה של חיבורים מול הבקר

M5STACK

בפרויקט זה, נשתמש ב M5Stack CORES3 ESP32S3-כבקר הראשי. ה M5Stack- ששייך למשפחת המיקרו-בקרים, ESP32 מציע ביצועים חזקים ותאימות רחבה עם ספריות תוכנה וחומרה רבות, ולכן בעל התאמה גבוהה לפרויקט שלנו. הוא כולל חלק מן חיישנים משולבים הדרושים לפרויקט, מה שמפשט את תהליך ההתקנה ומקטין את הסיכוי לטעויות בחיבורים. בנוסף, הM5Stack יעיל מבחינת צריכת החשמל וניתן לתכנות בקלות באמצעות Arduino IDE, ומציע גמישות בהתאמת מספר חיישנים, ומאפשר לסטודנטים לחקור ולחדש בתחום היישומים המבוססים חיישנים.

PAHUB

הצורך להרחיב את יציאת הI2C נפתר באמצעות האב שדרכו אנו יכולים להוסיף עוד חיישנים עם חיבור Grove .אנו משתמשים ב -(B040-U) AP9548PCA שדרכו אנו יכולים לחבר שישה חיישנים במקביל.

PBHUB

רכיב זה הוא האב עם יציאת I2C ודרכו ניתן לחבר עד שישה חיישנים שאינם I2C , כמו GPIO,PWM,ANALOG וכו'. מכיל פנימית בתוכו ממיר ADC ומכיל בקר פנימי STM32F030.

מד תאוצה וג'ירו

רכיב Bosch BMI 270 הינו IMU, המובנה בתוך M5Stack, אשר משלב בתוכו חיישן בעל 6 צירים מתוכם ג'ירוסקופ תלת צירי ברזולוציה 16 סיביות ומד תאוצה תלת צירי ברזולוציה 16 סיביות.

מד מרחק

מד מרחק SparkFun Qwiic ToF Imager VL53L5CX. היננו חיישן בעל 64 פיקסלים, טווח סריקה של עד כ4 מטרים ומפתח ראיה של עד כ-63 מעלות. סנסור זה משלב בתוכו מערך SPAD, מסנני אינפרא אדום פיזיים ואלמנטים אופטיים עקיפים בכדי לקבל ביצועים טובים ביותר עבור טווחי תאורה שונים וסביבות שונות.

מצלמה תרמית

מצלמה תרמית SparkFun Grid-EYE Amg8833 סנסור זה מכיל מערך תרמופילים בגודל 8 על 8 (64 פיקסלים) אשר מזהה באמצעות אינפרא אדום טמפרטורה. דיוק הרכיב היננו $\pm 2.5°C$, וטווח טמפרטורות $0°C - 80°C$. מצלמה תרמית זו יכולה לזהות חום גוף אדם עד כ7 מטרים וקצב הפריימים שלו ניתן לתכנות בין 1-10 פריימים לשנייה.

מד לחיצה

חיישן FSR402 הינו נגד חישת כוח, עם אזור יחיד אשר מותאם לשימוש בבקרת מגע אנושי של מכשירים אלקטרוניים. מדובר בנגד חישת כוח - מכשיר דו-חוטי בעל חיישן סרט עבה פולימרי אשר מפגין ירידה בהתנגדות עם הפעלת הכוח עליו. הסיגנל המוחזר הוא אנלוגי, ונדגם בעזרת מחלק נגדים.

LCD

LCD מובנה בm5stack הדגם היננו ILI9342C. זהו מסך המשמש לתצוגה בעל יכולת רזולוציה טובה וצג צבעוני.

VIBRATION MOTOR

מנוע רטט המכיל גלגל אקסצנטרי מתכתי. מהירות הסיבוב שלו היננו של 8800RPM.

RGB

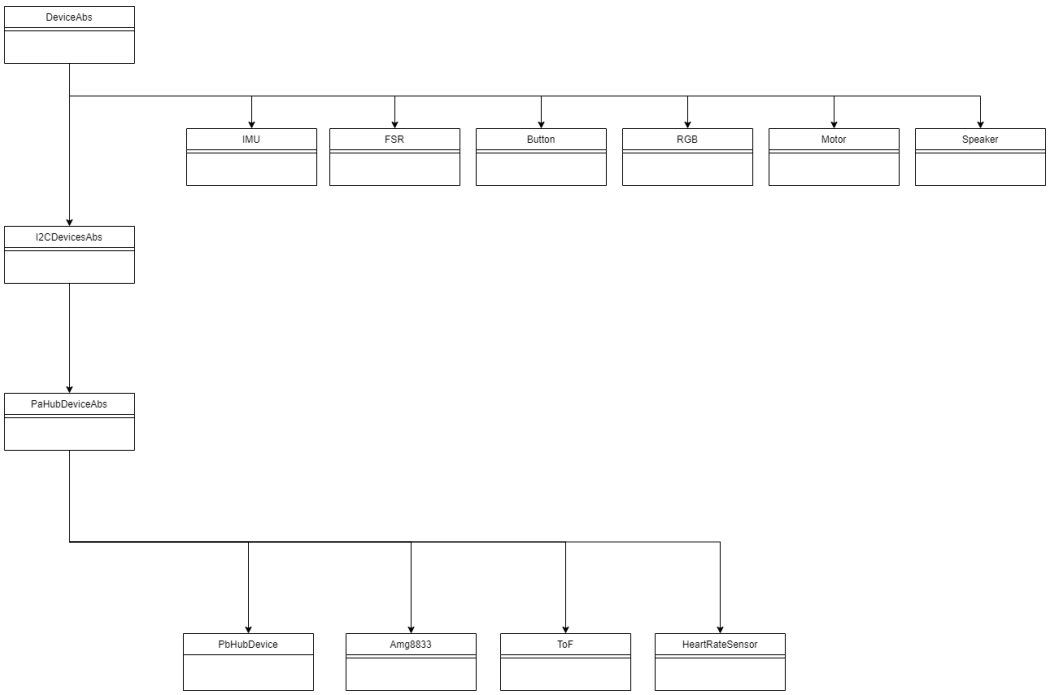
יחידת RGB עם 3 נוריות לד בודד.

תיאור תוכנה

התוכנה נתמכת ב windows, ובRaspberry Pi OS, מערכת ההפעלה של Raspberry PI. התוכנה מתחלקת ל2 חלקים – הראשון, הקושחה שתרוץ על הבקר והשני, הקוד שיפקד על הבקר בMaster Device.

DESIGN

הקושחה פותחה ב ++C באמצעות Arduino IDE. החלק המשמעותי בעבודה היה Design המחלקות של הפרויקט, שיהיה בצורה מודולרית ויאפשר בהמשך הוספה או הסרה של סנסורים או יכולות נוספות לסנסורים הקיימים. הסיבה לכך היא החשיבות בעינינו לשמור על הפרוייקט בר הרחבה ככל האפשר ובקלות. זוהי היררכיית המחלקות שרלוונטיות ל**סנסורים** בלבד:



איור 4-תיאור מחלקות התוכנה

המחלקה הבסיסית DeviceAbs, היא מחלקה אבסטרקטית שכולם יורשים ממנה שמכילה יכולות בסיסיות שכל סנסור תומך בהן, לכן הMethods שם וירטואליות.

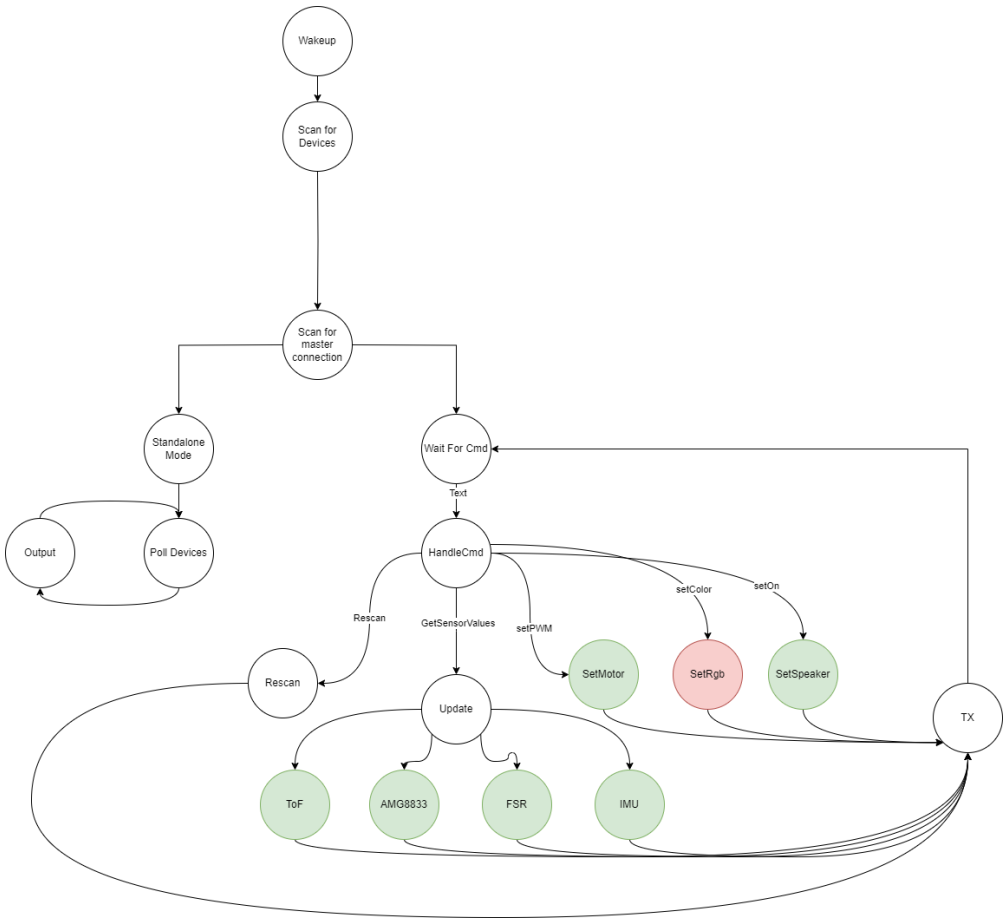
כל מה שאינו I2C (חוץ ממד תאוצה והג'ירו שמובנה כבר בm5 ולכן לא היה צורך לשרשר בהירכיה), יורש באופן ישיר מDeviceAbs (סנסורים אנאלוגים/PWM).

כל חיישן שמתחבר ל PaHub ,יורש מ PaHubDeviceAbs, שמכיל יכולות רלוונטיות לכל סנסורים שעובדים עם I2C, ואת אופן הפעולה עם ההאב.

המחלקה שבה משתמשים יעזרו לצורך שימוש בstandalone/slave, יהיה מחלקה שנקראת M5Telemetry. זו מחלקה שבה מתרחש הניהול של כל המערכת, הפקודות האפשריות שניתן לשלוח במצב SLAVE, ניהול מצב STANDALONE, סריקת מכשירים מחוברים וכדומה. במחלקה זו מתואר מעין API שניתן להשתמש בו כדי לפקד על המכשיר.

תיאור נוסף ומפורט של המחלקות ניתן למצוא בתיעוד של הפרוייקט באנגלית.

להלן מכונת המצבים:



איור 5-מכונת מצבים ארדואינו

נסביר כעת על כל חלק במכונת מצבים

סריקת מכשירים מחוברים

בעיה מהותית הקיימת, בהינתן ומחובר רכיב I2C, בPaHub, כיצד נוכל לוודא שהוא מחובר במוליפלקסר? ביצענו בקושחה פתרון עבור בעיה זו.

הפתרון הינו חיפוש בכל מרחב כתובת הi2c, וכאשר יש כתובת ידועה באמצעות החיפוש הצלחנו למצוא סנסורים מתוך רשימת סנסורים אפשריים – לכן את אותם סנסורים ניתן למצוא באמצעות המימוש שלנו באופן אוטומטי (לא דורש שינוי תכנוני), הדבר היחידי שנדרש רק להוסיף אותם למערך הסנסורים של הapi, בכדי שהתוכנה תכיר שקיים סנסור כזה.

בנוגע לשאר הפרוטוקולים (למעט I2C) – היות ואין אפשרות לוודא באופן חד משמעי אם אכן מחוברים או לא – אנו מאלצים את המשתמש להגדיר האם הם מחוברים או לא(כלומר **באחראיות המשתמש לוודא שמה שמחובר מתועד בקושחה**), ולהיכן מחובר (הסלוט בהאב שאליו הסנסור מחובר).

STANDALONE

במצב זה, הבקר רץ באופן עצמאי, עושה POLLING על המידע המגיע מהסנסורים המחוברים ומדפיס למסך. ישנם 2 אפשרויות:

- להגדיר זמן מסויים שתוצאות סנסורים יוצגו למסך (לדוגמה כל 10 שניות),
- הגדרת כפתור, ולתת אופציה למשתמש מעבר בין הדפסות של סנסורים באמצעות הכפתור.

הגדרת OUTPUT SENSORS- כלומר מנוע ויברציה, Speaker תחת תנאים מסוימים של מוצאים מסוימים דורש שינוי התנהגות בקוד FW, כלומר הוספה תמיכה שלהם במחלקות הרלוונטיות.

SLAVE MODE

כאמור המצב העיקרי והמטרה העיקרית של הפרויקט. בתחילת עולם באמצעות מחלקה שנקראת CommandHandler, מנסה להתחבר ל-MASTER(המחשב או הpi) באמצעות חיבור סיריאלי(UART), או WIFI. בכדי להשתמש בWiFi, **לפני** צריבת הגרסא יש לשנות את פרטי הWifi שרוצים להתחבר אליו.

עם ההתחברות CommandHandler, ממתין למידע מהMASTER, ובהתאם לבקשה שמקבל מטפל בפקודה ואם יש צורך בסיום הפקודה מחזיר מידע חזרה (נרחיב בהמשך על כיצד זה עובד).

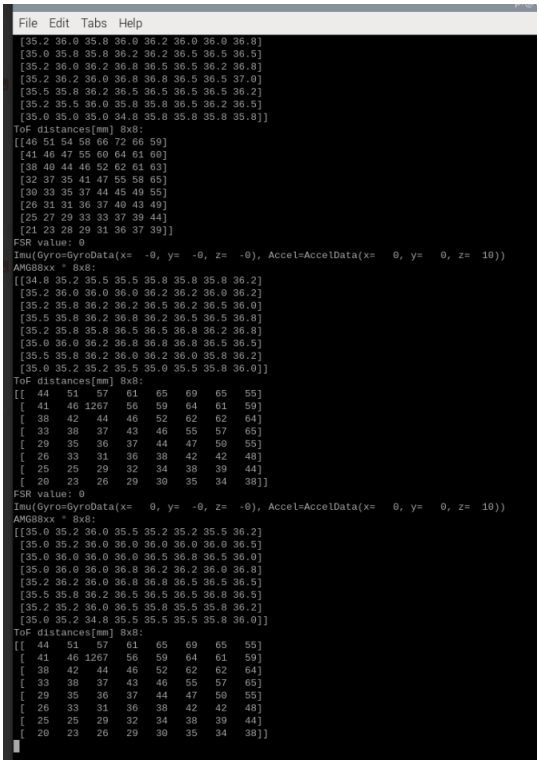
איור 6-פקודת סריקה שנשלח דרך ראספברי פאי

יצרנו Python API, שמתנהג כMaster, והוא זה שמחכה להתחברות של הבקר אליו. עם ההתחברות דרכו ניתן לשלוח פקודות כמו בקשה של ערך של סנסור מסויים ואותה בקשה נשלחת לבקר דרך המחלקה CommandHandler שצוינה קודם לכן.

הוספנו פקודות נוספות כמו להפעיל את המנוע ויברציה ולכבותו, או להפעיל את הבאזר דרך הפייתון.

המטרה של API שיצרנו היא להקל על המשתמש ויצירת כללי מערכת אמינים לשליחת וקבלת מידע מהבקר.

כאמור מצב זה נבדק מול התוכנה של RASPBERRY PI, שמתנהג כמאסטר. את התוצרים ניתן להמיר למטריצת NUMPY, לצורך אנליזה(נעזרנו בהם לניתוח תוצאות בדיקות החומרה) או הדפסה ידידותית למשתמש, למסך.

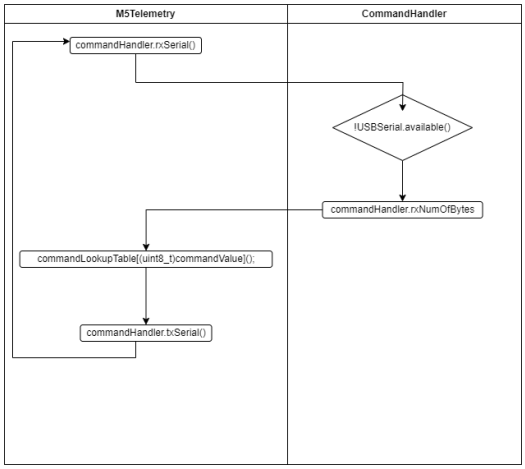


איור 7-ערכי הסנסורים המתקבלים מארדואינו מודפסים לראספברי פאי

כעת נרחיב על 2 סוגי החיבורים(מכונת מצבים שלהם).

SERIAL

בחיבור באמצעות USB, כמו שהוסבר קודם לכן, החיבור הוא UART שמשתמש בCOM Port בכדי לבצע אמולציה של תקשורת טורית. זוהי מכונת המצבים ההתנהגותית כאשר שולחים פקודה באמצעות הmaster:



איור 8-מכונת מצבים במצב של חיבור סריאלי

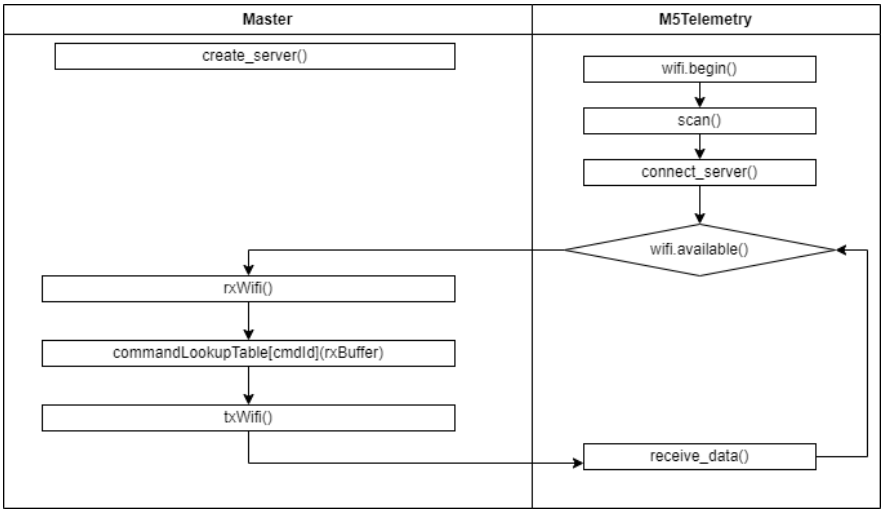
כפי שניתן לראות זוהי התנהגות מחזורית, בתחילת עולם בCommandHandler, אם מקבלים מידע, עוברים לשלב השני של המכונת מצבים שהוא קבלת המידע אל תוך BUFFER פנימי (תוכנה). עם קבלת המידע מהMASTER, קוראים בlut, שקיים ל API, בו כל פקודה יש מספר סידורי מוגדר מראש על ידי המשתמש. באותה פקודה, יודעים לשחזר את המידע

שהועבר מהMASTER, בתור ארגומנטים לאותה פקודה, מימוש הפקודה, ואם הסיום קוראים לפונקציה פנימית של CommandHandler, שיודעת לשדר בחזרה שסיים לעבוד, ובמידת הצורך להחזיר מידע חזרה לmaster.

חשוב להדגיש כפי שהוסבר כאן, שברמת המשתמש, במקרה של הוספת סנסור לבנק הסנסורים, יהיה צורך בהוספת פקודות חדשות או שינוי קיימות, לשנות ידנית בצד הmaster ובצד הslave את הארגומנטים של אותם פונקציות / או כיצד לפרסר את המידע שכל צד מקבל.

- **לשים לב, שהבקר תמיד ינסה להתחבר לחיבור סריאלי(זמן ריצה האופטימלי) לפני מצב WiFi.**
- **נתמך רק בwindows.**

WIFI



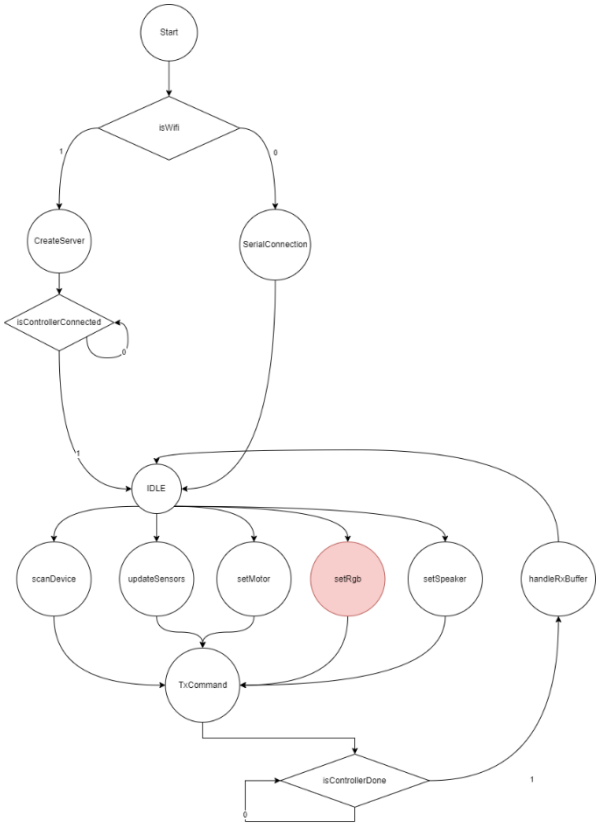
איור 9-מכונת מצבים במצב התחברות של ויפי

ניתן לראות שהפתרון עבור WIFI, מורכב יותר היות וזה דורש הקמת שרת בצד של הmaster, בתחילת עולם. על **המשתמש** בטעינת התוכנה אל הבקר(UPLOAD), לדאוג לרשום את פרטי הwifi הנכונים וכתובת הip שבו השרת נמצא(ע"י ipconfig בוינדוס, או ע"י ifconfig בפי).

במהלך הbootup, אם לא הצליח ה master להתחבר לslave באמצעות חיבור סריאלי, ינסה להתחבר דרך WIFI. לאחר שהצליח להתחבר לwifi, הבקר ינסה להתחבר לשרת, ועם הצלחה להתחברות בשרת, ההתנהגות התוכנתית זהה לפעילות הסריאלית. הבקר ימתין לפקודה מהמאסטר, ועם קבלת הפקודה יטפל בה ויחזיר בחזרה שסיים ובמידת הצורך יעביר מידע נוסף שהמאסטר ביקש מראש(תלוי פקודה).

מכונת המצבים של תוכנת המאסטר(פייתון)

להלן מכונת המצבים בצד של המאסטר



איור 10-מכונת מצבים של מסטר(פייתון)

ניתן לראות שבתחילת עולם בודקים, האם המשתמש במאסטר ביקש לייצר חיבור באמצעות Wifi או Serial. במידה ומדובר על חיבור בWifi, המאסטר מייצר שרת ומחכה לחיבור של הבקר. במידה ומדובר על חיבור סריאלי – הפייתון מייצר חיבור סריאלי מול הבקר.

לאחר ההתחברות, הפייתון במצב IDLE, ומחכה לפקודה שהמשתמש יכניס, את אותה פקודה הוא משדר לבקר, והבקר יטפל בה. עם סיום הפקודה הבקר יודיע למאסטר על האם יש מידע להעביר בחזרה, במידה וכן המאסטר יחכה לשידור כל המידע מהבקר. לאחר מכן במידה ויש מידע שהתקבל מהבקר הפייתון בהתאם לפקודה מטפל בבקשה, במידה ואין מידע הbuffer (שמכיל את כל המידע) יהיה ריק ויחזור למצב IDLE שוב.

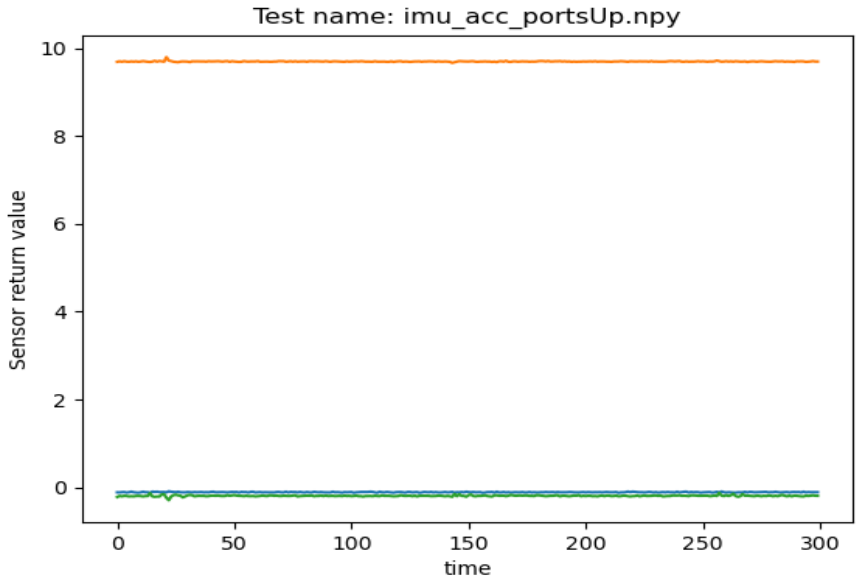
בדיקות ותוצאות בדיקה

לאחר הרכבת המערכת, ביצענו סט של בדיקות על רכיבי המערכת. הבדיקות ותוצאותיהן מתוארות בפירוט בדוקומנטציה של המערכת באנגלית. הבדיקות נעשו בחיבור למחשב Windows בחיבור סיריאלי או וויפי.

את התוצאות הפקנו באמצעות שימוש במצב Slave, ע"י לקיחת סט דגימות לבדיקה מסוימת. לחלק מהסנסורים נדרש מספר בדיקות בעוד שבחלקם נדרש רק בדיקה אחת(תלוי מורכבות סנסור).

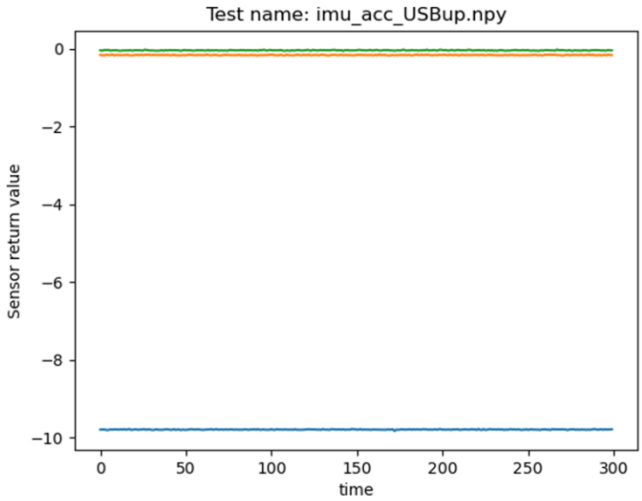
מד תאוצה וג'ירו

הבדיקות שנעשו על מד התאוצה והגירוסקופ קבעו את התנהגות החיישן במנוחה ובתנועה.



איור 11-תוצאות בדיקה תאוצה במצב סטטי

החזקנו את המד תאוצה באוויר(מצב סטטי). ניתן לראות שקיבלנו בציר Z בקירוב טוב את תאוצת כדור הארץ $9.81[\frac{m}{s^2}]$ ובשאר הצירים השינוי בציר X,Y יציב.



איור 12-תוצאות בדיקה סטטית של אוריינטציה

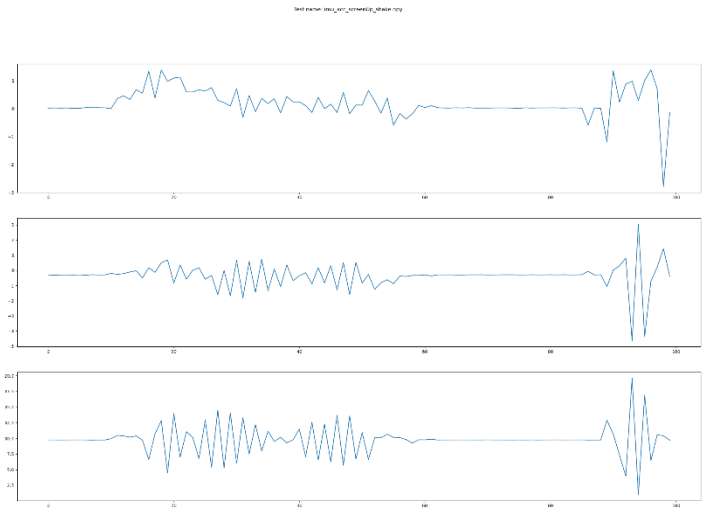
ביצענו בדיקה סטטית בכדי לוודא את האוריינטציה של המד תאוצה וג'ירו, נשים לב ש:

- מיקום המסך למעלה חיובי בציר Z
- מיקום ה USB-למטה חיובי בציר X
- מיקום יציאת IIC למעלה חיובי בציר Y

כלומר קיבלנו את התוצאות שציפינו שנקבל בבדיקה זו.

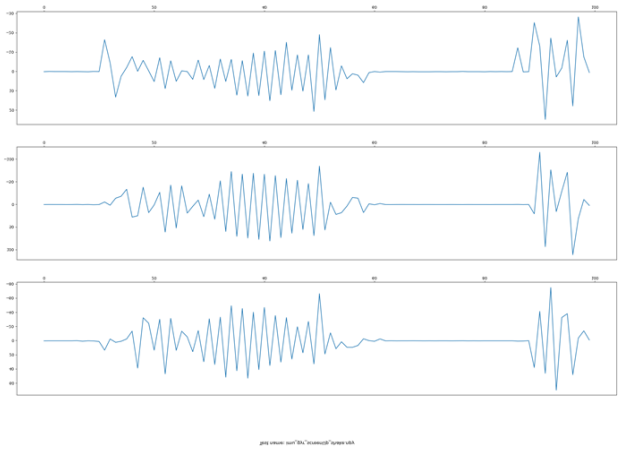
טסטים דינמיים של המד תאוצה וג'ירו

ביצענו ניעור של ההמד תאוצה וג'ירו , ולאחר מכאן הרמה שלו, בכדי להקל ביצענו הפרדה בין תוצאות הג'ירו, לתוצאות Accelerometer.



איור 13-תוצאות טלטול במד תאוצה

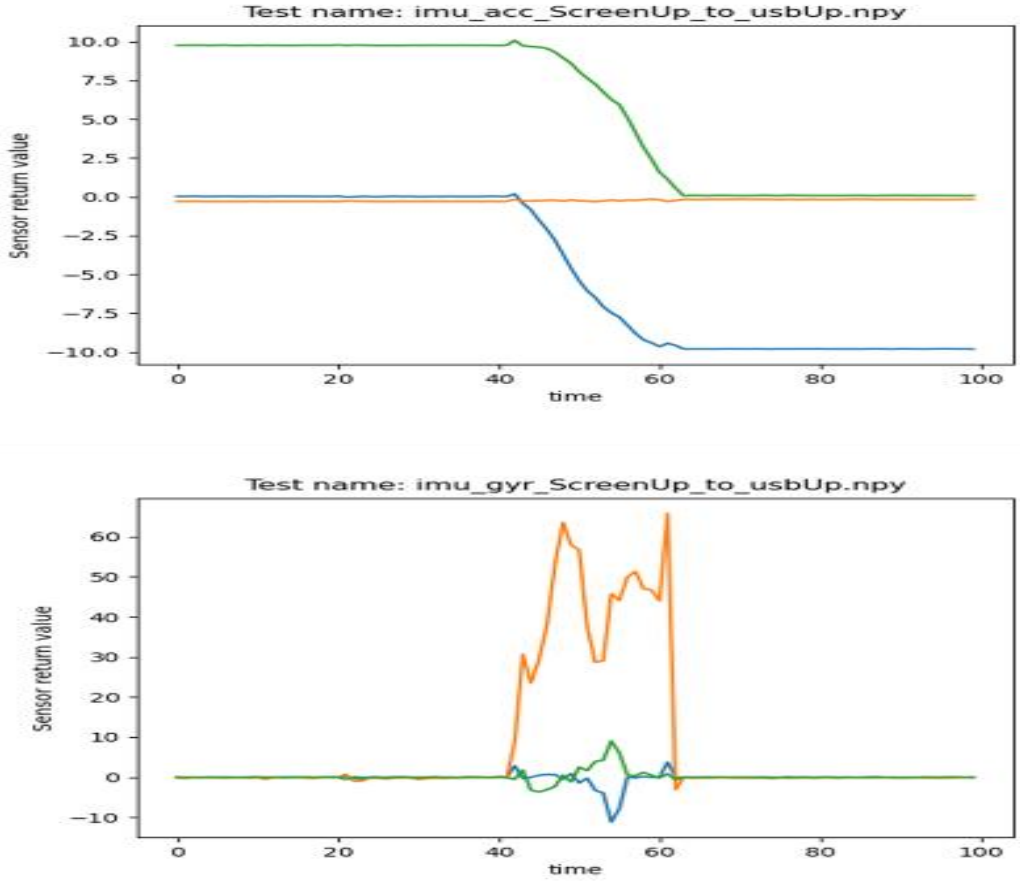
ניתן לראות על פי מד התאוצה שיש ניעור (אפשר לראות ש-3 הצירים מושפעים כפי שצופה כי עשינו תנועות מעגליות בניעור).



איור 14-תוצאות טלטול בג'ירו

ניתן לראות בג'ירו שאכן יש ויברציות המתארות ניעור (נראות בצורת רשרוש), לכן נשיק שהג'ירו והמד תאוצה במצב דינמי נותנות תוצאות טובות ומתארות בצורה ובה את הפעולה שביצענו.

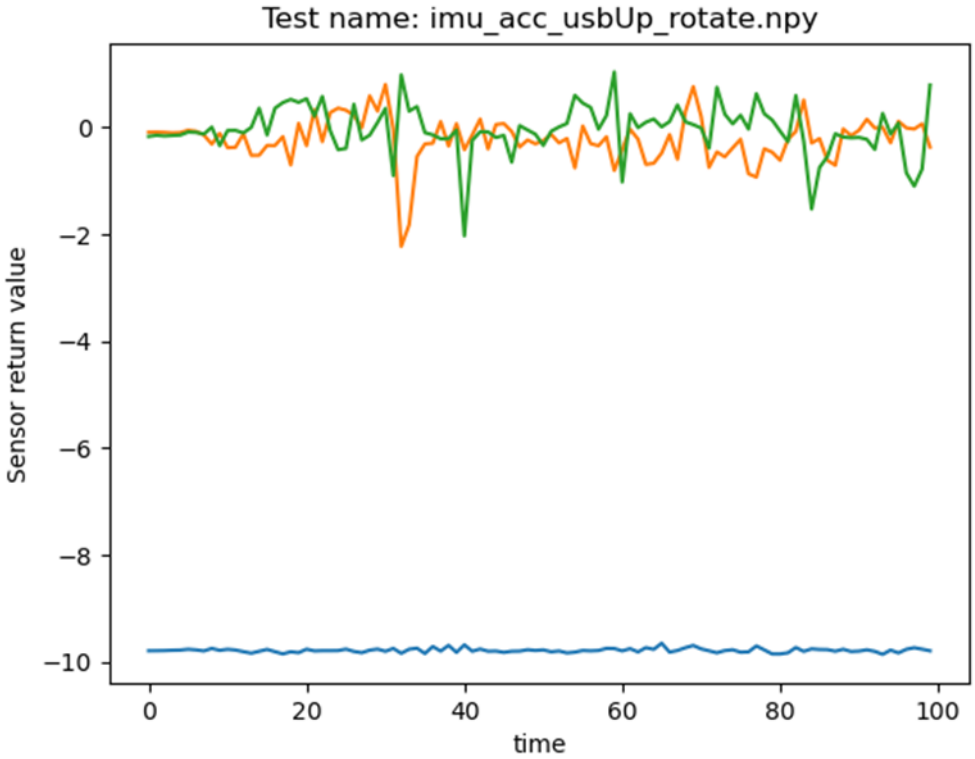
הבדיקה הבאה מתארת שינוי מנח של המערכת, ובה ניתן לראות תנועה חלקה של האקסלרומטר.



איור 15-תוצאות בדיקת ויברציה

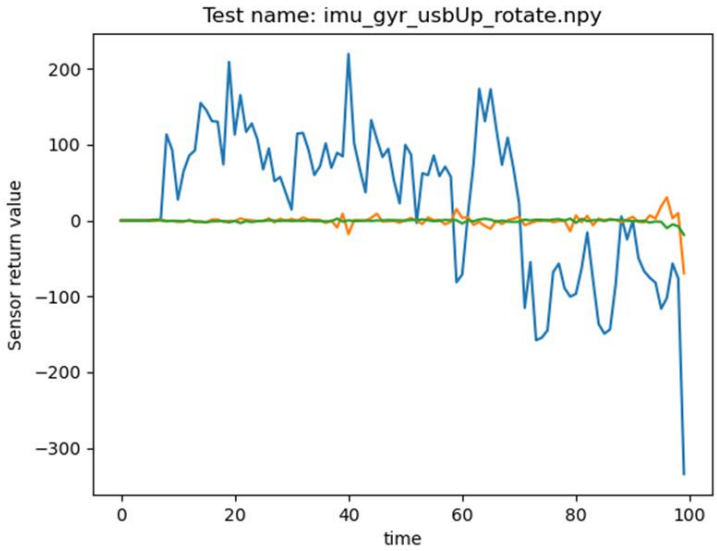
בבדיקה המכשיר הונח על השולחן עם המסך מעלה, ולאחר כמה שניות מנחו שונה בעדינות למצב בו פורט הUSB מצביע למעלה. ניתן לראות בתוצאות את התנועה ב2 צירים באקסלרומטר, ואילו בציר השלישי בגירוסקופ. זוהי תוצאה טובה, כי אם נחשוב על התנועה המבוצעת, אכן ישנו שינוי ב2 צירים באקסלרומטר, ואילו הציר השלישי חווה סבסוב.

הבדיקה האחרונה שביצענו זו בדיקת רוטציה של המד תאוצה וג'ירו, על ידי 'נעילת' שתי צירים אחרים וקיום של ציר אחד בלבד:



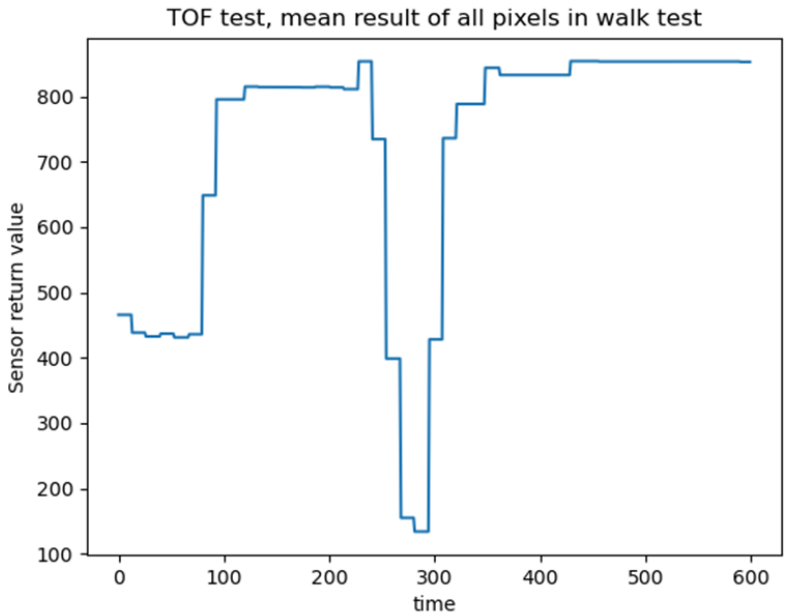
איור 16-תוצאות בדיקת רוטציה של מד התאוצה

על סמך גרף התאוצה ניתן לראות כי אכן 2 הצירים אחרים נעולים וציר הרוטציה אכן בעל תאוצה.



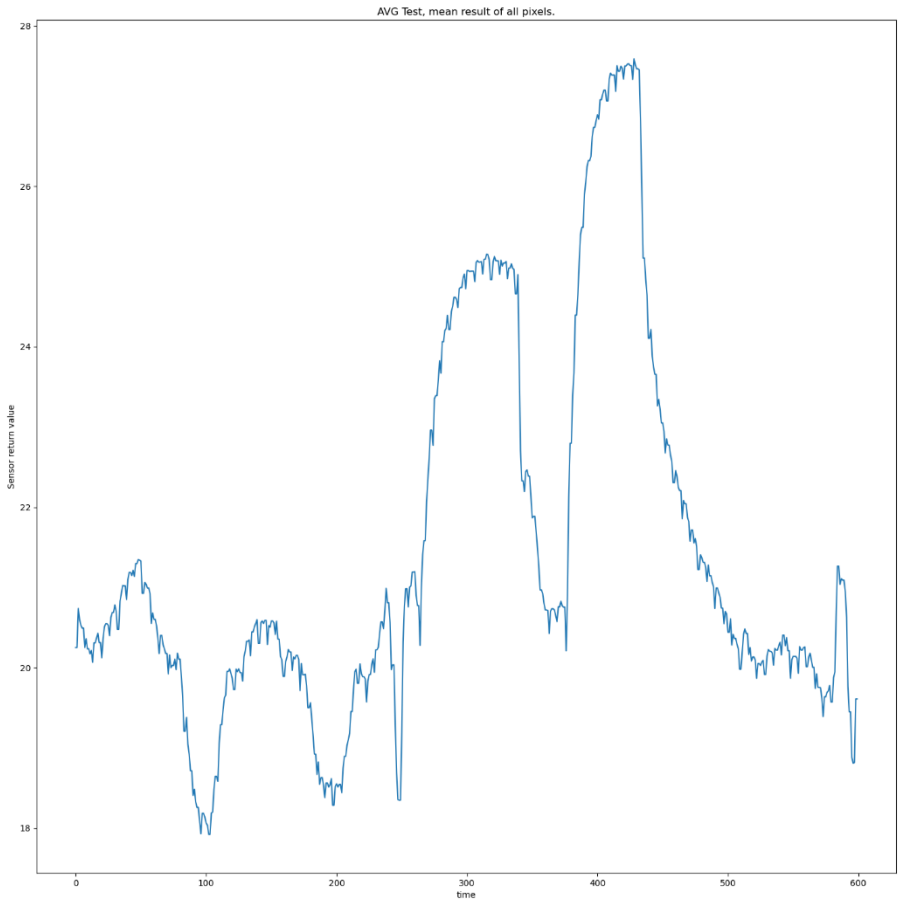
איור 17-תוצאות בדיקה רוטציה של הג'ירו

ציר הרוטציה באופן ברור מושפע יותר מאשר הערכים אחרים שבמהלך הבדיקה היו באזור ה-0(מה שאומר שהסנסור זיהה באופן טוב את ציר הרוטציה שהזזנו).



איור 18-תוצאות בדיקה ממוצע כל פיקסלים של מד המרחק

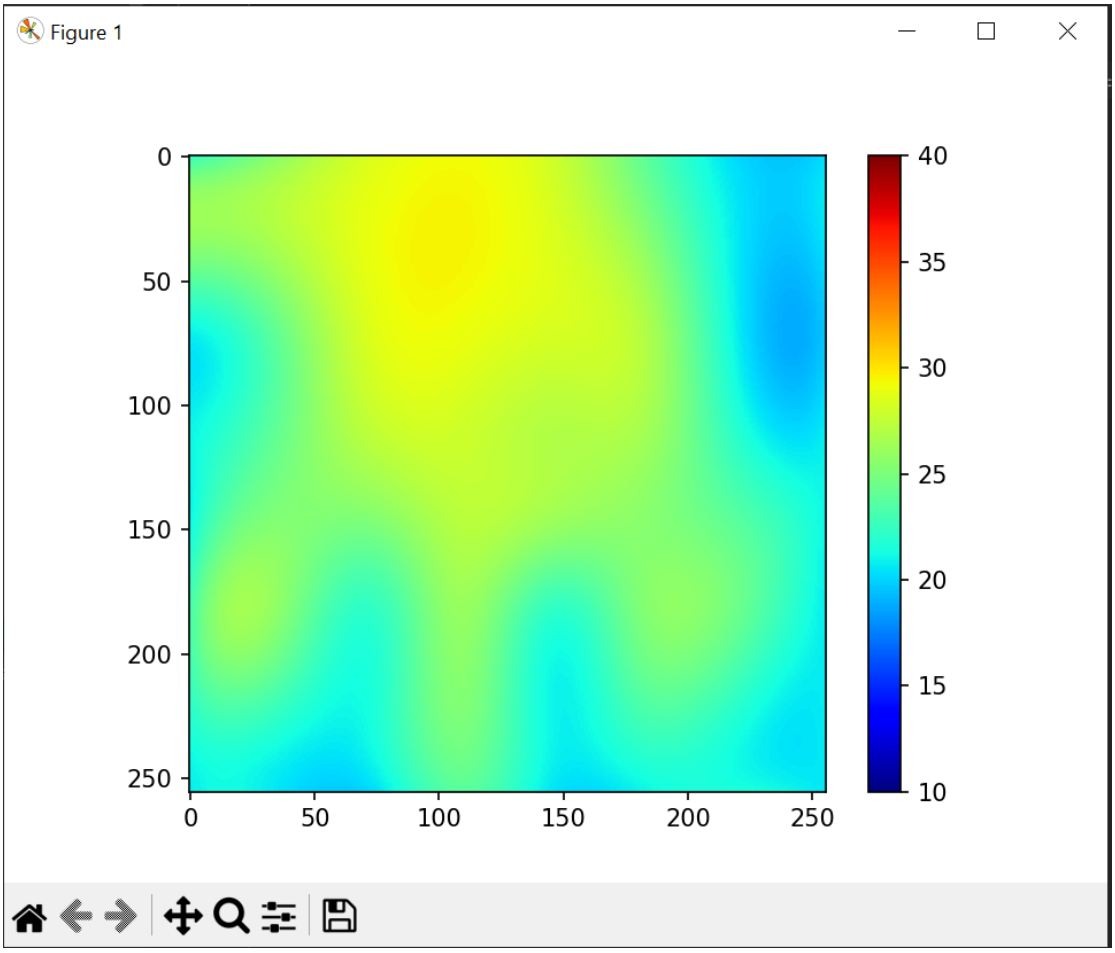
בבדיקה זו הנסיין התרחק והתקרב למד המרחק במעבדת בקרה. התוצאה המוצגת היא ממוצע כל הפיקסלים, אך תוצאה של כל פיקסל בנפרד מוצגת בתיעוד.



איור 19-תוצאות בדיקה ממוצע כל הפיקסלים של המצלמה תרמית

עמדנו מול המצלמה תרמית, ניתן לראות שהוא משקף בצורה טובה את טמפ' הגוף של אדם העומד מול המצלמה (ממוצע הדגימות של כל 64 הפיקסלים).

בדיקה נוספת זה נחת יד מול המצלמה התרמית ובאמצעות אינטרפולציה לוודא שאנחנו רואים את היד:

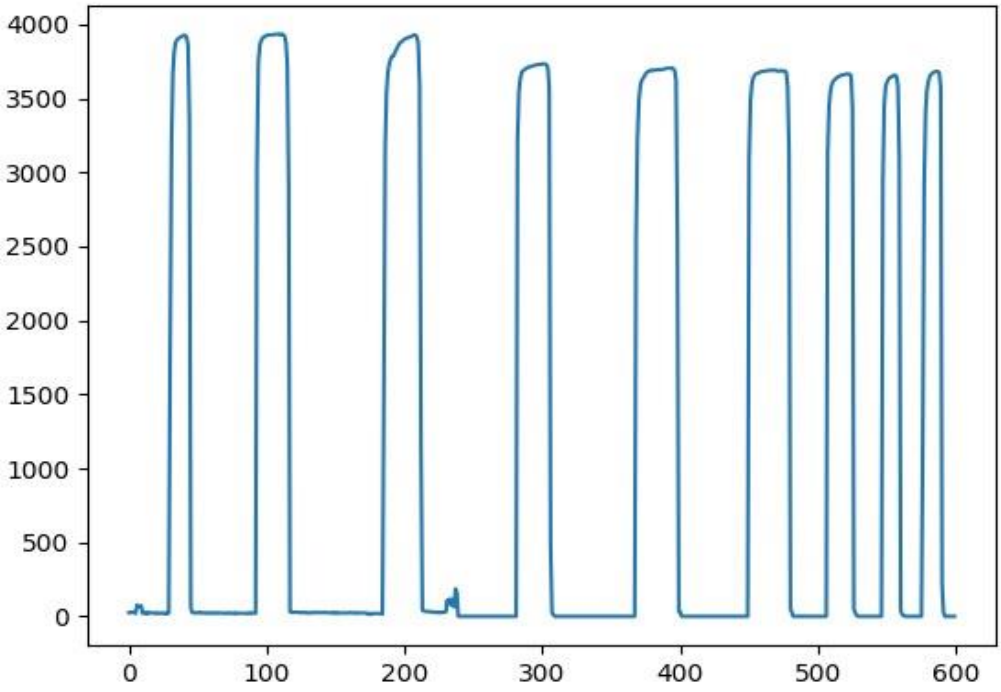


איור 20-בדיקת מצלמה תרמית ע"י הנחת יד מול המצלמה ואינטרפולציה של הפיקסלים

ניתן לראות שבערך קיבלנו את צורת היד, צריך לקחת בחשבון שמדובר רק על 64 פיקסלים לכן קשה לראות תמונה מדויקת, אך סה"כ ניתן לראות צורה של יד כפי שצופה.

מד לחץ

איור 21-בדיקת מד לחץ על ידי לחיצה ושחרור



ניתן לראות שמד הלחץ משקף בצורה טובה את הפעולה שביצענו שזה לחיצה ושחרור עם האצבע.

הערות לגבי כישלונות בבדיקה

- מלבד לחיבור USB שביצע התנהגות לא צפויה במערכת הפעלה לינוקס(ולכן נכון לרגע זה אינו אמין) וה-RGB, המערכת תפקדה מעבר למצופה מבחינת ביצועים - תדירות דגימה של יותר מ10 הרץ.

תיעוד הפרויקט

תיעוד הפרויקט הינו בעל ערך משמעותי, שמטרתו להקל על סטודנטים שישתמשו בעתיד בתשתית זו וגם להבטיח להם להיעזר בתשתית שנבדקה ואמינה. הפרויקט והקוד נמצא בgithub, יש תיעוד לקוד התוכנה.

הסיבה שבחרנו בgithub,היות וזו פלטפורמה נוחה לניהול פרויקטים, שבה מספר קבוצות יכולות להשתמש בתשתית ולהכניס יכולות נוספות ללא תלות בקבוצות אחרות ובפרט, יכולת ניהול פרוייקטים מודולרית, כמו בחירת גרסה מסוימת.

כאמור בחלק התוכנתי, בקוד עצמו יש תיעוד מפורט של כל מחלקה וכל פונקציה ומה מהות, בנוסף לכך בחלק המאסטר ישנו גם תיעוד של כל הפעולות שמתבצעות מאחורי הקלעים (אף על פי שסביר להניח שלא ישנו את היסודות של התשתית שמימשנו), ישנם שני חלקים לתוכנה, תוכנה הקושחה שנמצאת בתיקיית main, ותוכנת הMaster(הapi פייתון), שנמצא בתוך CLI. ישנו בתוך הגיטהאב, דמו עבור שני מצב הריצה מול המאסטר, למצב ריצה STANDALONE, יש

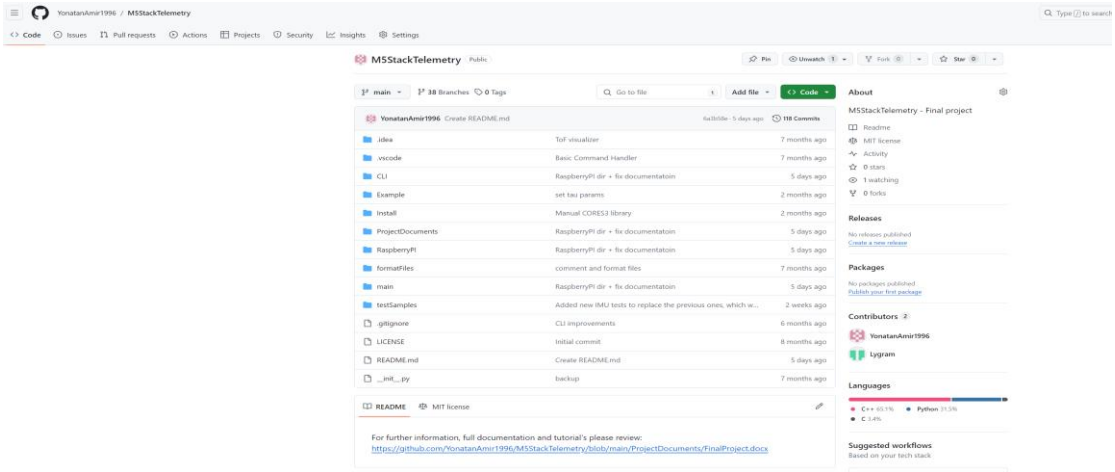
בתוכנה לדרוש לרוץ במצב זה או בהינתן ולא מצליחים להכנס למצב SLAVE, הבקר ירוץ במצב STANDALONE.
הוספנו כלים כמו ThermalVisualizer שמטרתם להמחיש כיצד ניתן להיעזר בתשתית שעשינו ליצירת כלים נוספים בצד המאסטר.

בנוסף לכך עשינו דמו המראה את כל היכולות באמצעות STANDALONE, וגם דמו נקודתי להוצאת מידע מסנסורים(שככל הנראה יהיה עיקר השימוש בתשתית). בנוסף לכך יש תיעוד שעשינו, אשר מסביר בפרוט-פרוט על הסיבה לפרוייקט, דרך הפיתרון - כלומר על החומרה של הפרויקט, כיצד התוכנה של הפרויקט תמומש(תכנון), כיצד איך היא מומשה בפועל ומכונות מצבים של חלקים עיקריים בתוכנה, ותכנון המחלקות הראשי של הפרוייקט.

בנוסף לכך יש מדריכים שצירפנו לתיעוד שנועדו להקל על המשתמש, כמו כיצד להתקין את התוכנות הנדרשות, כיצד לעבוד מול התשתית שביצענו, כיצד ניתן יהיה להרחיב את יכולותיה והתוצאות בדיקות שעשינו המוכיחות את יכולות ורמת הביצועים של הסנסורים שנבחרו לפרויקט זה. בנוסף לזה צירפנו קישורים למדריכים מומלצים לפייתון + ארדואינו(במידת הצורך), והסבר כללי של כיצד לעבוד עם Arduino, ואיך לעבוד עם הapi, שפיתחנו עבור תשתית המאסטר(של הפייתון).

הוספנו מדריך להוספת סנסור חדש וכיצד להוסיף פקודה חדשה שיהיה ניתן לשלוח לבקר, כיצד בקושחה לטפל בפקודה החדשה שנשלחה והסבר כיצד עובד שליחה וקבלה של מידע לבקר שמתפקד בתור הslave . המטרה של התיעוד להיות כמה שיותר שקוף ומודולרי למשתמש העתידי.

מצורף גם ה design החומרתי בתוך הגיט, במידה ויהיה צורך להרחבה או הסרת סנסורים נוספים – ובכך ימנע עיצוב ותכנון מחדש, של החלק החומרתי. בנוסף לכך הוספנו תייקיה לpi לגיט שבמידת הצורך אם ירצו, יוכלו להוסיף את כל המדריכים ההתקנות והדברים הרלוונטיים אליהם בתיקיה זו, ובכך לאגד תיעוד זה ואולי בהמשך תיעודיים עתידים על בקרים נוספים שיעזרו בהם.(כמובן שיהיה ניתן לבנות תיקיות חדשות).



איור 22-דף גיטהאב הראשי של הפרוייקט

להלן הקישור:

- תיעוד- https://github.com/YonatanAmir1996/M5StackTelemetry/blob/main/ProjectDocuments/FinalProject.docx
- הפרויקט - https://github.com/YonatanAmir1996/M5StackTelemetry

סיכום, מסקנות והצעות להמשך

דברים הטעונים לשיפור בפרויקט:

- במקור היה צורך שpi יתמוך בחיבור סריאלי, אך בעקבות זה שבחרנו בקר חדש מדי – לא היה תמיכה ולכן הpi עובד רק באמצעות WIFI.
- הrgb נכשל בבדיקות שביצענו מולו ואופרטיבית התגלה כלא אמין לשימוש.

מסקנות:

- עדיף לעבוד לפעמים עם גרסה לא הכי עדכנית של מערכת, אך אמינה יותר.
- קנייה של כמה סוגי חיישנים מאותו סוג ובדיקת אמינות של כל אחד.

ניתן יהיה להרחיב לתמיכת סנסורים נוספים(הפרויקט מודולרי מראש עבור אופציה זו). ניתן להכין פרויקטים מגוונים בguiב(יש דוגמאות בgit של הפרויקט עצמו). בנוסף יהיה אפשר לעדכן את קוד התוכנה של החומרה של m5.

[1] "M5Stack CoreS3 ESP32S3 IoT Development Kit Specification https://shop.m5stack.com/products/m5stack-cores3-esp32s3-lotdevelopment-kit

[2] Bosch IMU specification: https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi270/

[3] PaHub specification - https://shop.m5stack.com/products/i2c-hub-1-to-6-expansion-unit-pca9548apw

[4] PbHub specification - https://shop.m5stack.com/products/i-o-hub-1-to-6-expansion-unit-stm32f0

[5] SparkFun Qwiic ToF Imager - VL53L5CX https://www.sparkfun.com/products/18642

[6] SparkFun GRID-EYE Amg8833 - https://www.sparkfun.com/products/14607

[7] FSR402 Data sheet https://www.trossenrobotics.com/productdocs/2010-10-26-DataSheet-FSR402-Layout2.pdf

[8] Vibration motor unit - https://shop.m5stack.com/products/vibration-motor-unit