מתקן פדלים חכם

ספר פרויקט - פרויקט גמר תשפ"ד

נעשה ביוזמה ותמיכה של בית החולים שיקומי "בית לוינשטיין"

מגישים: דור אגבבה ודוד אילוז

**חתימת המנחה**

**.**

**ניהול גרסאות:**

|  |  |  |  |
| --- | --- | --- | --- |
| מס' גרסה | תאריך | עודכן על ידי | הערות |
| 1.0 | 14 ספטמבר 2024 |  | גרסה ראשונה |

תוכן עניינים

[1. תקציר................................................................................................................................3](#_Toc157643129)

* 1. [תקציר בעברית.. 3](#_Toc157643130)
  2. [תקציר באנגלית 3](#_Toc157643131)

1. [טבלאות ומונחים. 4](#_Toc157643133)
   1. [טבלת איורים.. 4](#_Toc157643134)
   2. [טבלת טבלאות.. 4](#_Toc157643135)
   3. [מילון מושגים.. 4](#_Toc157643136)
2. [מבוא: 5](#_Toc157643137)
   1. [הגדרת הבעיה.. 5](#_Toc157643138)
   2. [האתגר הטכנולוגי.. 6](#_Toc157643139)
3. [דרכי פתרון.... 6](#_Toc157643140)
   1. [הפתרון המוצע: 6](#_Toc157643141)
4. [תוצר מצופה מהפרויקט. 7](#_Toc157643142)
5. [רעיונות דומים המהווים השראה 7](#_Toc157643143)
6. [סיכונים, אי וודאות ואילוצי הפרויקט. 7](#_Toc157643144)
7. [רקע וסקירת ספרות. 9](#_Toc157643145)
8. [מפרט דרישות פונקציונאליות.. 9](#_Toc157643146)
   1. [דרישות מהפרויקט.. 9](#_Toc157643147)
   2. [מפרט דרישות חומרה. 9](#_Toc157643148)
      1. [יחידת מחשוב: Raspberry pi 4 9](#_Toc157643149)
      2. [Raspberry Pi Camera 9](#_Toc157643150)
      3. [HC­SR04 Ultrasonic Sensor 9](#_Toc157643151)
9. [מה בוצע בפרויקט 10](#_Toc157643152)
   1. [RaspberryPi 11](#_Toc157643153)

[10.2. חיישני סונאר.. 11](#_Toc157643154)

[10.3. יומן הפרויקט.. 14](#_Toc157643155)

1. [הפרויקט כמקור לימוד ויישום הידע האקדמי......................................................................20](#_Toc157643156)
2. [תכנון מול ביצוע ...............................................................................................................20](#_Toc157643157)

[12.1. טבלת אבני דרך. 20](#_Toc157643158)

[12.2. טבלת משימות.. 21](#_Toc157643159)

[13. מבט לעתיד – הצעות לשיפור הפרויקט................................................................................23](#_Toc157643160)

[14. נספחים............................................................................................................................24](#_Toc157643161)

1. **תקציר**
   1. **תקציר בעברית**

הפרויקט הינו מתקן לשיפור החוויה ועידוד השימוש במתקן דיווש פשוט המשמש מטופלים בשיקום פיזי, כזה הנהוג בבתי חולים שיקומיים כמו בית לוינשטיין ברעננה. המתקן יאפשר מדידת פרמטרים כגון מהירות הדיווש, מרחק ועוד וישתמש בתוצאות המדידה כדי לספק ולהפסיק תגמול למטופל, כגון השמעת קריאות עידוד וחשיפת תמונות, וכדי לספק למטופל ולמטפל מידע על מהלך הטיפול.

* 1. **תקציר באנגלית**

The project is a device aimed at enhancing the experience and encouraging the use of a simple pedaling device used by patients in physical rehabilitation, such as those commonly found in rehabilitation hospitals like Beit Loewenstein in Ra'anana. The device will enable the measurement of parameters such as pedaling speed, distance, and more, and will use the measurement results to provide and withdraw rewards for the patient, such as playing encouraging sounds and revealing images. Additionally, it will provide both the patient and the therapist with information regarding the progress of the treatment.

1. **טב****לאות ומונחים**
   1. **טבלת איורים**

איור 1: גרף היצע עבודות לפי שנים בכוח עזר.....................................................................7

איור 2: כיסא גלגלים אוטנומי שנמצא בשדה התעופה ביפן..................................................9

איור 3: תוצאות ניסוי סונאר (אדם)..................................................................................13

איור 4: תוצאות ניסוי סונאר (מתכת)................................................................................13

איור 5: תוצאות ניסוי סונאר (קרטון)................................................................................13

איור 6: תוצאות ניסוי סונאר (בד).....................................................................................13

איור 7: מיקומי החיישנים על הכיסא................................................................................13

איור 8: מיקומי החיישנים ברסברי פיי..............................................................................14

איור 9: "עדשת עין דג"....................................................................................................14

איור 10: לוח שחמט לצורך קליברציה...............................................................................15

איור 11: תוצאות ניסוי מצלמה זווית 0..............................................................................16

איור 12: תוצאות ניסוי מצלמה זווית 30............................................................................16

איור 13: תוצאות ניסוי מצלמה זווית 60............................................................................16

איור 14: תוצאות ניסוי מצלמה זווית 90............................................................................16

איור 15: תוצאות ניסוי מצלמה.........................................................................................17

איור 16: סכמת בלוקים...................................................................................................23

* 1. **טבלת טבלאות**

טבלה 1: טבלת אבני דרך עמוד.................................................................................................18

טבלה 2: טבלת משימות..........................................................................................................19

טבלה 3: טבלת תוצאות ניסוי חיישנים......................................................................................33

טבלה 4: טבלת תוצאות ניסוי מצלמה.......................................................................................33

* 1. **מילון מושגים**

*CPU* (Central Processing Unit): המרכיב העיקרי של מחשב המבצע הוראות של תוכנית מחשב.

Raspberry Pi 5: מחשב קטן עם לוח יחיד שפותח על ידי קרן Raspberry Pi, פופולרי בשל הרבגוניות שלו ובמחיר סביר.

*Linux*: משפחה של מערכות הפעלה דמויות Unix בקוד פתוח המבוססות על ליבת לינוקס, בשימוש נרחב בשרתים, מחשבים ומערכות משובצות.

*חיישן*: מכשיר שמזהה ומגיב לכניסות מהסביבה הפיזית, וממיר אותן לאותות.

*Open Source*: תוכנה או חומרה עם רישיון המאפשר לכל אחד להשתמש, לשנות ולהפיץ אותם באופן חופשי.

1. **מבוא**

בית לוינשטיין הינו בית חולים המתמחה בשיקום פיזי, נפשי, ורפואי של חולים עם מגבלות פיזיות. לבית לוינשטיין כחלק מהשיקום הפיזי מספר מתקני כושר ואחד מהם הוא מתקן פדלים שמשמש עבור מטופלים לשיקום הגפיים ,ואף לחלק מהמטופלים פשוט פעילות גופנית כדי למנוע התנוונות .

* 1. **הגדרת הבעיה**

בעולם השיקום, מתקני פדלים הופכים לכלי חיוני המקל על מטופלים במהלך שיקומם. למטופלים אין רצון ומוטיבציה לדווש, וכדי להשגיח עליהם כל הזמן דרוש כוח אדם שאינו זמין תמיד. מתקני הפדלים שלנו מספקים להם את התמיכה החשובה והמתמשכת שזקוקים כדי לקדם את ההתקדמות ולחזק את הגפיים בדרך יעילה ונעימה.

1. **סקירת ספרות** **– פתרונות קיימים**

כיום יש פתרונות עבור אופניים רגילות בעיקר לדוגמא [TACK FLOW](https://www.garmin.co.il/product/1731?gad_source=1&gclid=CjwKCAjw5v2wBhBrEiwAXDDoJdPowxFtA4DrcQmFLdaGQKLhWMqVMIejizAj-S8TFuBcYwcyZ-09BBoC4ekQAvD_BwE) . חלק מהפתרונות אכן מתבססים על אופניים רגילות שמתקינים עליהן התקן שלמעשה הופך אותן לקבועות ,וקצב הדיווש לנתונים דיגיטליים. כמו כן יש תלות במכשיר שמריץ את האפליקציה מה שיתכן שאין לכל מטופל. כמו ואז גם אפשר להוסיף מסך וחוויות המשתמש הופכת למשהו הרבה יותר כיף . לדוגמא המוצר : [Zwift](http://www.zwift.com)  , שעשו את הפתרון על אופניים רגילות שהפכו להיות מקובעות עם התקן מסוים והוסיפו גם מסך עם אפליקציה.   
ישנם פתרונות שלקחו את החלק רחוק במציאת מסך להצגה והשתמשו במשקפי vr לדוגמא המוצר: [Blync](https://www.blync.bike/) .

1. **הפתרון המוצע-תיאור כללי:**

הפרויקט יהיה מערכת עם חיישן הול, בקר ומסך שתותקן על כל סוגי האופניים ותהפוך את חוויית הרכיבה לחוויה אינטראקטיבית. כחלק מהמערכת נציג על גבי מסך את מהירות הדיווש, חיווי בקול ע"י רמקול בתחילת הדיווש, במהלך הדיווש ובסוף הדיווש. בזמן הדיווש תיחשף תמונה לאט לאט וברגע שיפסיק הדיווש תפסק החשיפה. דו"ח סיכום הפעילות יופק בסוף הפעילות, ובכך המתאמן יוכל לעקוב אחר פעילותו.  
הפרויקט יחולק לשלושה חלקים כדי ליישם את כל הנאמר: תכן מכאני, תכן חשמלי, תכן תוכנה.

1. **מטרת הפרויקט**

הפרויקט נחוץ עבור מטופלים שמתקשים להניע את רגליהם או ידיהם ,ויש להחדיר בהם מוטיבציה , הפרויקט יעזור גם למטפלים ולמטופלים לעקוב אחר התקדמותם ולקבוע להם יעדים חדשים ובכך לקדם את תהליך השיקום.

1. **מפרט התוצר המצופה מהפרויקט**

התוצר הסופי של הפרויקט שלנו הוא טכנולוגיה הניתנת ליישום בפדלים מכאניות או כל סוג אופניים אשר יביאו את המטופלים לאימון ארוך ומהנה ואת המטפלים למעקב קל ומרובה בו זמנית על מספר מטופלים. הפרויקט ידע לדווח בזמן אמת על פעילות שמתרחשת ולהתחיל ולסיים באופן עצמאי אימון, כחלק מניתור הפעילות יוצגו למסך פרטי הפעילות כגון: מהירות, מרחק, קצב ממוצע, משך הפעילות, מספר הסיבובים ועוד.

הפרויקט גם יכלול מערכת שתעביר פידבק קולי בזמן אמת, כאשר המתעמל מתחיל את האימון, מגיע לנקודות ציון חשובות, או כאשר ישנה ירידה משמעותית בקצב.

המערכת תעצור את חשיפת התמונה כאשר הקצב יורד, ותעודד את המתעמל להמשיך בפעילות. בנוסף, בסיום כל אימון, יופק דו"ח ביצועים המפרט את משך הפעילות, הקצב הממוצע, הקלוריות שנשרפו, והמרחק שעבר.

* 1. **מפרט דרישות מהפרויקט**

**דרישות פונקציונאליות:**

1. על המערכת לזהות תחילת תנועה ולהפעיל את המסך ואת שאר המערכת, ולהתחיל להקליט את נתוני האימון.
2. על המערכת לשלוח הודעות קוליות המשבחות את המתעמל בתחילת האימון, על מעברו בנקודות ציון . כאשר המערכת מזהה ירידה משמעותית בקצב , יופסקו התגמולים שניתנו לו (חשיפת תמונה) , והמערכת תנסה לעודד אותו להמשיך בפעילות.
3. על המערכת להפיק דו"ח ביצועים בסופו של כל אימון. הדו"ח יכלול: משך הפעילות, קצב ממוצע וגרף הקצב לאורך הפעילות.

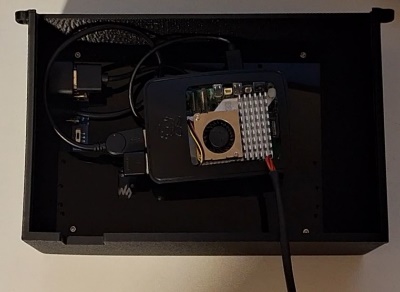
**דרישות נוספות (פיטצ'רים)**:

1. על המערכת לחשב ולשמור את הקלוריות שהמתאמן שרף בזמן האימון כמו כן לחשב את המרחק שעבר, ולהכלילם בדו"ח הביצועים.
2. על המערכת להיות מתאימה לסוגי אופניים שונים .

1. **תיאור המוצר**
   1. **רשימת רכיבים כוללת**
2. 32 GB SDHC card
3. ) Raspberry pi 5 with 5A power adapter ראה נספח )
4. Raspberry pi 5 fan coolant
5. Raspberry pi 5 case
6. 7 inch HDMI LCD (H) IPS Capacitive Touch Screen 600X1024
7. 7 inch lcd case
8. KY-003 Hall Magnetic Sensor Module (ראה נספח )
9. USB speaker
10. HDMI to micro HDMI cable 30 cm
11. Usb to micro usb 180 degree
12. HDMI male to Female 180 degree adapter
13. 3 pcs 1 meter cable female to female pin to pin
14. 4 pcs M2 25 mm screws with nuts
15. 2 pcs M3 15mm with nuts
16. Tablet flexible 130 cm arm
17. Custom 3d print case (ראה נספח )
18. Scotch strips paste female male
    1. **תיאור כללי**

המוצר מורכב משלושה חלקים , חלק מכאני , חלק תוכנתי וחלק חשמלי :

1. חלק מכאני – מודול המסך יכוסה בכיסוי מותאם אליו, לחלק האחורי שלו יודבק הבקר הרסברי פי 5 בתוך כיסוי פלסטיק שלו. כל זה יכנס לתוך מעטפת פלסטיק גדולה שהודפסה במיוחד במדפסת תלת מימד ויתפוס את המסך עם ברגים. המעטפת עצמה ניתנת לפתיחה בגב שלה (מכסה נשלף כלפי מעלה) ומאובטח באמצעות ברגים (ראה נספח ).   
   תופסן גמיש באורך כמטר ושלושים שיתחבר לפדלים שלנו , ויאחוז בצדו השני את מעטפת פלסטיק הגדולה שלנו.



איור 1 - חיבור מכאני



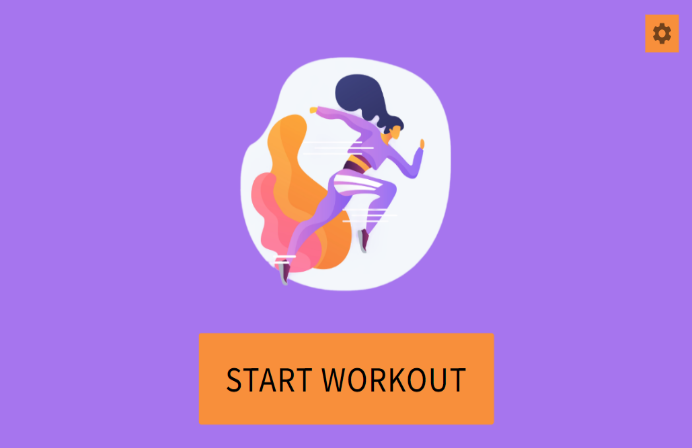
1. חלק תוכנתי – החלק התוכנתי מורכב משני חלקים : חלק Back-End ו Front –End .  
   Back-End – קוד בפייתון שירוץ באופן תמידי ויהיה אחראי על קבלת interrupts מהחיישן המגנטי וקידום המונים הנדרשים. כמו כן , יהיה אחראי על איפוס מונים כל כמות זמן מסוימת, הדרישה כרגע היא אחת לשניה . יתר על כן, יחושב המרחק שבוצע – מספר הסיבובים כפול היקף סיבוב (2\*pi\*disc radius ) ,קצב ממוצע של סיבובים לדקה ( מספר הסיבובים הכולל חלקי מספר הדק) , קלוריות יחושבו גם כן על פי חשבון משוערך שיכלול את זמן העבודה והמרחק . בנוסף יהיה עליו ליצור תקשורת באמצעות socket עם ה Front-End ולהעביר לו נתונים . (תרשים זרימה של אינטגרציית מערכת חומרה/תוכנה)

איור 2 - תרשים זרימה של אינטגרציית מערכת חומרה/תוכנה

Front-End – האפליקציה בנויה על בסיס FRAMEWORK בשם TAURI והקוד בו אנו כתבנו הוא REACT JS. הקוד אחראי על ה User Interface של המערכת כולה. בנוסף הקוד אחרי לתקשר בעזרת socket לפעולות שנקלטות בקוד הפייתון ומעדכנים את הFRONT-END ובמידת הצורך מעדכנים את ה BACK-END על עצירה/התחלה/איפוס מונה בהתאם לבקשת הלקוח לעצור/להתחיל/לסיים תוכנית. האפליקציה מורכבת משלושה מסכים, מסך פתיחה שבו יש אפשרות לבחור באייקון ההגדרות את קוטר הגלגל , ויש אפשרות להתחיל ידנית את הפעילות, אם במסך זה תבצעו 3 סיבובים במרווח זמן של 5 שניות התוכנית תתחיל בעצמה ותועברו למסך הפעילות. במסך הפעילות, יוצגו נתוני הפעילות שיתעדכנו כל שנייה: קלוריות שנשרפו, זמן עבודה, מספר הסיבובים שנעשו, מרחק במטרים, מהירות עכשווית(בשנייה האחרונה) שנמדדת בKm/H, מהירות ממוצעת שנמדדת על פני כל הפעילות שנמדדת גם היא ב Km/H, ומרחק עד ליעד הבא – בפעילות הוגדרו מראש יעדים להשגה (הראשוני הוא 500 מטר) ובכל פעם שהמשתמש יגיע ליעד תצטרף מדליה והוא יקבל הודעת שיבוח קולית על כך. כמו כן באמצע המסך נוכל למצוא ריבוע כתום שמאחוריו מסתתרת תמונה המחולקת ל256 חלקים. כל שנייה שבה המתאמן יבצע פעילות ייחשף חלק קטן מהתמונה. סט התמונות הינו תמונות של מפורסמים וידועים בישראל. בכך אנחנו מעודדים ומסקרנים את המשתמש להמשיך להתאמן. למתאמן היכולת להשהות את הפעילות כשירצה וגם לסיים את התוכנית כשירצה. כמו כן המשתמש יכול להגביר ולהנמיך את עוצמת השמע. במידה והמתאמן לא יתאמן כלל כדקה וחצי ברצף התוכנית תסיים בעצמה ותעבור למסך הסיכום. במסך הסיכום יופיעו כל הפרמטרים שנמדדו במהלך הפעילות ובנוסף יהיה גרף מהירות ופעילות. הגרף מציג בצורה מדגמית, כל נקודה מייצגת 10 שניות והערך עבורם הוא המהירות המקסימלית ב10 שניות האלה (גרף וורוד). כמו כן ניתן לראות גרף בינארי שמסמן האם המתאמן היה במנוחה או לא. הגרף הינו אינטראקטיבי וניתן לגעת בכל נקודה ולקבל מידע.  
כלל הקוד שביצענו מופיע כאן : [smartBicycleRSP](https://github.com/DorAgababa/smartBicycleRSP/tree/main) .

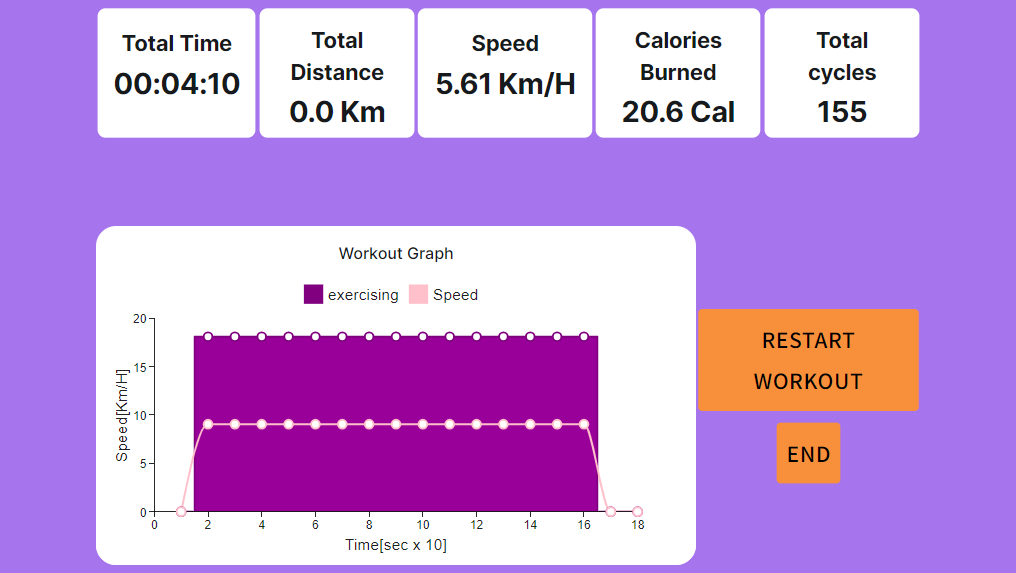
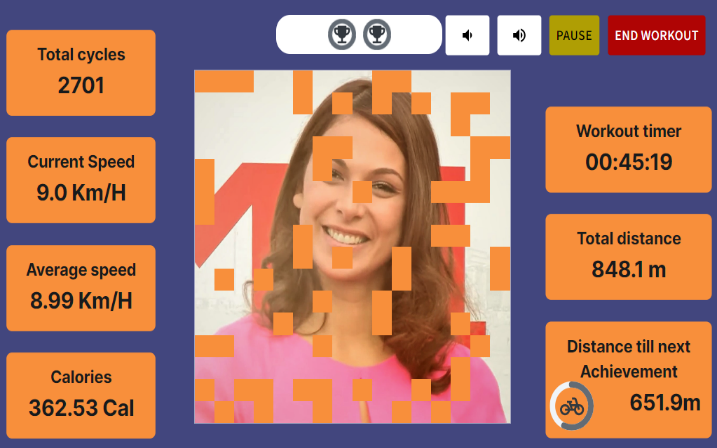


עריכת קוטר גלגל



**מסך פתיחה**

**מסך פעילות**

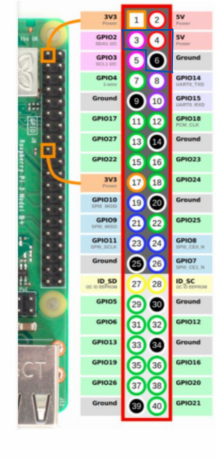
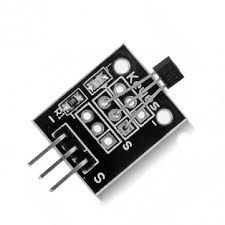


**מסך סיכום**

איור 3 - ממשק משתמש

1. חלק חשמלי – לבקר Raspberry pi 5 (ראה נספח) מחובר שנאי חשמלי תואם לחשמל בית או

מקור מתח נייד . לבקר יחובר חיישן הול מגנטי עם 3 פינים (ground,v5 ,data ) ראה איור מס. 4 .



5V

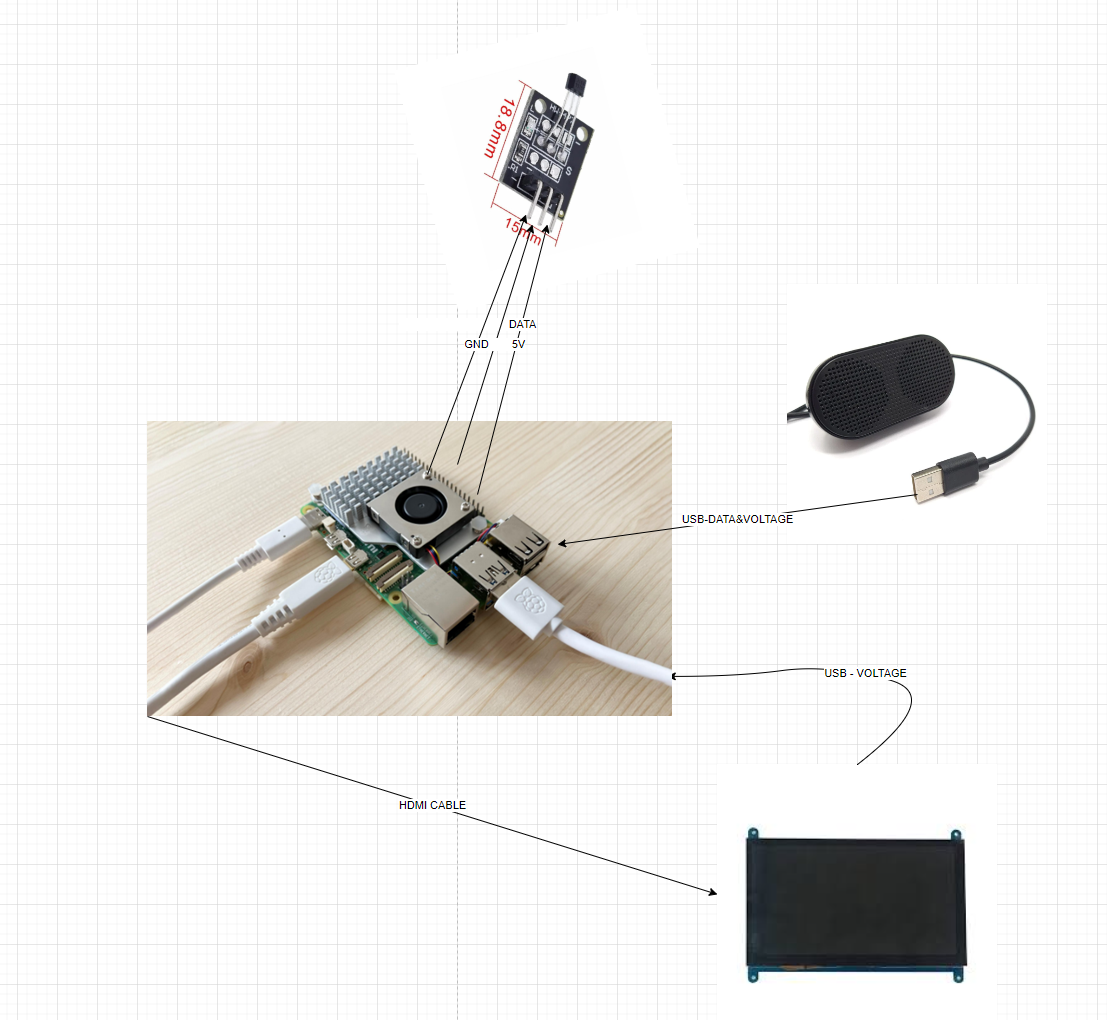
DATA

GND

איור 4 - חיבורי פינים ברסברי פיי

בנוסף יחובר מסך 7 אינץ עם חיבור hdmi בשימוש מתאם hdmi Micro to hdmi וכבל עם

usb micro להזנת חשמל מהבקר. ובנוסף לכל זה יחובר רמקול בחיבור USB.לצורך קירור הבקר נחבר מאוורר משולב צלעות קירור באמצעות 4 פינים קפיציים ומתאם ייחודי בעל 3 פינים. ראה תרשים מס. 5



איור 5 - תיאור מערכת חשמלית

* 1. **סביבת עבודה**

כרטיס RaspberryPi5 הוא Microprocessor אשר מסוגל לבצע פעולות רבות כמו מחשב רגיל. מערכת ההפעלה היא לינוקס, חינמית (Open Source) והיא עובדת ביותר מ90% מהמחשבים בעולם. התוכנה כתובה בשפת Python.

לפני התקנת מערכת ההפעלה, יש להוציא את כרטיס הזיכרון ולפרמט אותו. ניתן להיעזר בסרטון הבא: [How to Format and Reuse a Raspberry Pi SD Card (youtube.com)](https://www.youtube.com/watch?v=otztORT2zxk)

לאחר הפירמוט יש להכניס שוב את כרטיס הזיכרון ולצרוב עליו את מערכת ההפעלה שתצורף על ידינו. הערה חשובה: כאשר מתקינים את מערכת ההפעלה, יגיע השלב שבו בוחרים את מערכת ההפעלה, יש לבחור את האפשרות use custom ולסמן את הקובץ img שניתן על ידינו בלבד .

ניתן להיעזר במדריך שנכתב ע"י Raspberry Pi : [Set up your SD card | Setting up your Raspberry Pi | Coding projects for kids and teens](https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2)

1. **שלבים ואבולוציה – יומן הפרוייקט (יומן הפרויקט**

בשלב הראשון , התחלנו לנתח את הדרישות שהגיעו עם הפרויקט והתחלנו לחקור איזה רכיבים נצטרך. למעשה ההחלטה הראשונה העיקרית לעשות היא בחירת הבקר. כשהבנו שהצורך מהבקר יהיה גם להתממשק עם חיישנים (**GPIO'S** ) וגם להיות מסוגל להציג **GUI אינטראקטיבי** וכבד ו**גם להתחבר למסך ולקבל ממנו פקודות (לחיצות במסך מגע).**

לשם כך, מנחה הפרויקט ביחד עם אחראי המעבדות בקמפוס עזרו לנו לגבש החלטה על בקר ועל שאר הרכיבים עבור הפרויקט. לאחר בדיקת מאפייני עבודה ופונקציונליות של כל הרכיבים החלטנו על הרכיבים שאיתם נעבוד בפרויקט.

|  |  |  |  |
| --- | --- | --- | --- |
| שם הבקר | מכיל GPIO | מסוגל להציג GUI | מסוגל להתחבר למסך |
| Arduino | כן | מסוגל,בסיסי בלבד ופיתוח מורכב מאוד. | עם המסך הנוכחי לא. דרוש מסך מיוחד עם חיבורי pins. |
| ESP32 | כן | לא עומד בדרישות בסיסיות לשם הצגת GUI מבחינת כוח מחשוב. | עם המסך הנוכחי לא. דרוש מסך מיוחד עם חיבורי pins. |
| RaspberryPi | כן | כן | כן |

* 1. **RaspberryPi**

בתחילת העבודה על פרויקט הגמר, קיבלנו מהמעבדהOrangePi5 (בקר מקביל ל(Rasoberry pi. ניסינו להתקין עליו מערכת הפעלה בסיסית, השתמשנו בהמון מדריכים שונים אך לאחר שהושקע בכך זמן רב, וראינו שיש המון משתמשים ברחבי העולם שמדווחים כי הבקר חווה תקיעות רבות, ממשק המשתמש שלו לא משתווה לממשק של הרסברי פי, הוחלט להזמין ולעבוד עם RaspberryPi5.

ההבדלים בין גרסה 3 לגרסה 4 הם גדולים מאוד:

Raspberry Pi 5:

Processor: Broadcom BCM2712 64-bit quad-core ARM Cortex-A76.

RAM: Options for 4GB, or 8GB LPDDR4X SDRAM.

USB Ports: 2 USB 2.0 ports and 2 USB 3.0 ports.

Ethernet: Gigabit Ethernet , with support for 2.5 Gbps.

Video Output: Dual micro HDMI ports supporting up to 4K resolution at 60fps.

GPIO Pins: 40 GPIO pins.

Wireless: Dual-band Wi-Fi 6 (802.11ac) and Bluetooth 5.0.

Power Supply: USB Type-C.

MicroSD Card Slot: Yes.

לסיכום, בגרסה ה-4 יש לנו יכולות עיבוד מהירות משמעותית מהגרסה ה-3. בנוסף, ניתן להרחיב את כמות זיכרון הRAM, יציאות USB 3.0 להעברת נתונים מהירה יותר. Gigabit Ethernet לקישוריות רשת קווית מהירה יותר. יציאות מיקרו HDMI כפולות לתמיכה בתצוגה כפולה ורזולוציית 4K. ספק כוח USB Type-C לאספקת חשמל טובה יותר. יכולות אלחוטיות משופרות עם 802.11ac Wi-Fi ו-Bluetooth 5.0.

* 1. **חיישני סונאר**

HC­SR04 Ultrasonic Sensor הם חיישנים פופולאריים מאוד למדידת מרחק. משתמשים בגלים קוליים כדי לקבוע את המרחק בין החיישן לאובייקט.

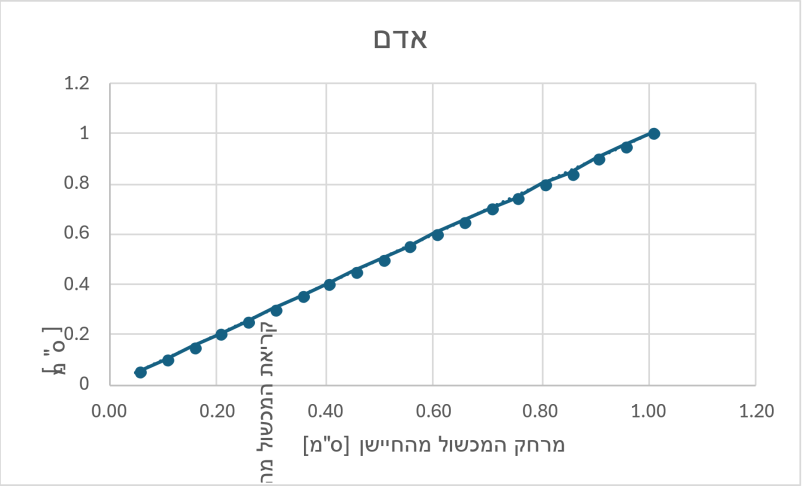
עיקרון העבודה:

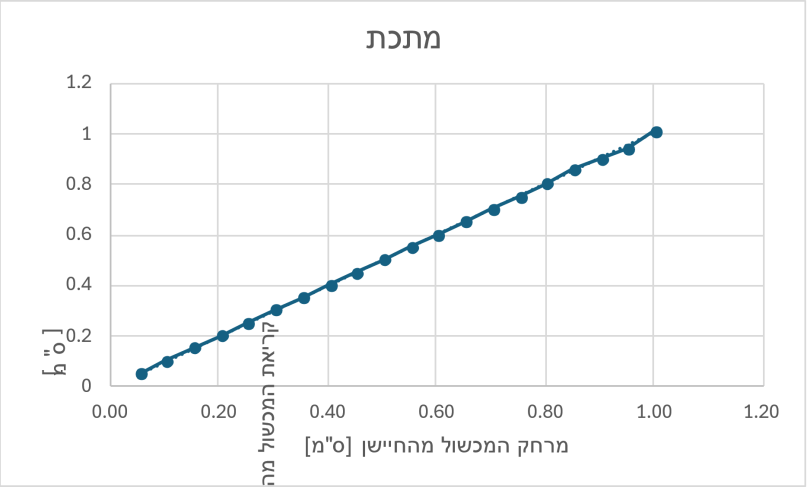
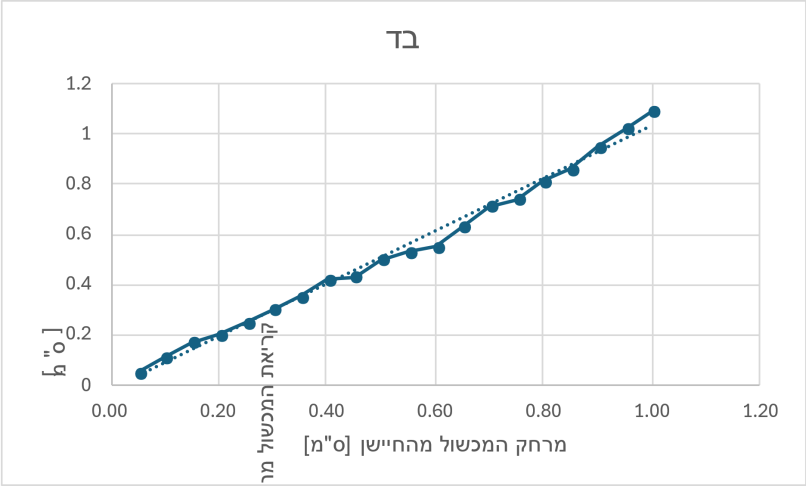
החיישן שולח גלי קול ומודד את הזמן שלקח לגל לחזור לאחר שפגע באובייקט.

נוסחה:

הפונקציה לחישוב המרחק כבר בנויה וניתן לראות זאת בנספחים.

הניסויים שביצענו היו: לקחת כמה סוגי מכשולים כגון: בד, מתכת, קרטון ואדם.

להלן התוצאות:

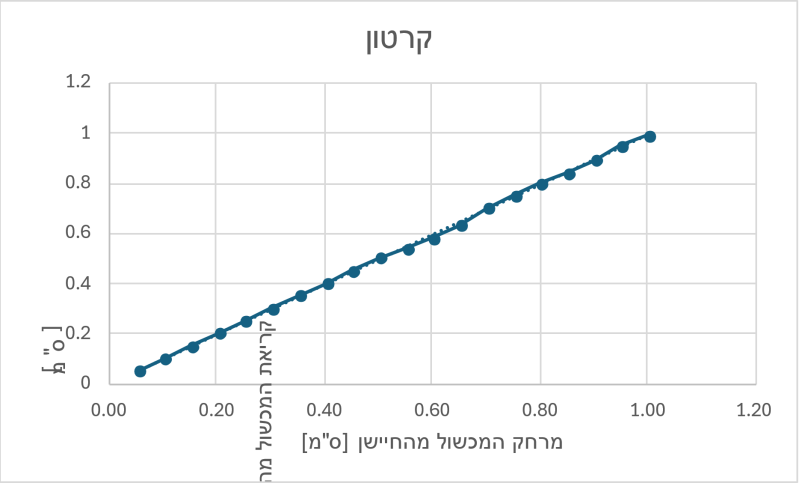
**

איור 6: תוצאות ניסוי בד

איור 5: תוצאות ניסוי קרטון

איור 2: תוצאות ניסוי מתכת

איור 3: תוצאות ניסוי אדם

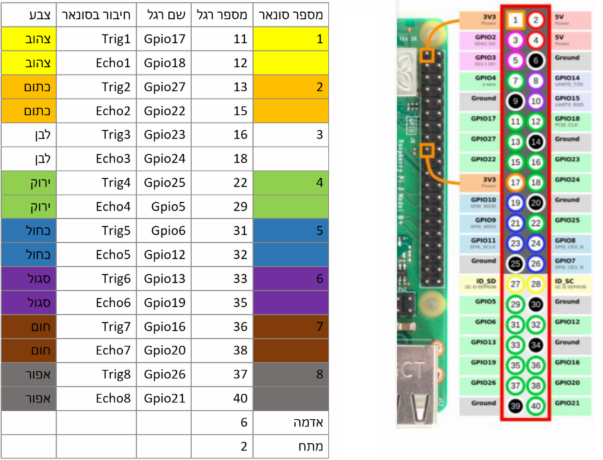


ניתן לראות שהחיישנים יכולים לזהות את כל סוגי החומרים שהזכרנו ובדיוק רב. קבענו לכל חיישן מקסימום זיהוי של 4 מטר בשביל הרלוונטיות של המכשול ובשביל לא להיכנס לזמנים ארוכים מדי עד לקבלת קריאה מהחיישן.

ς S VEI 
S't3 
.ς-τςι. הצבנו 8 חיישנים מסביב לכיסא בצורה הבאה:

איור 7: מיקומי החיישנים על הכיסא

כל מספר חיישן מייצג את הזווית שאליה הוא מכוון. כאשר הכיסא נע ישר, הזווית היא אפס.

לא היה ניתן להוסיף עוד חיישנים מאחר ואין מספיק חיבורי GPIO, ניתן לראות את החיבורים של החוטים לפינים לפי צבעי החוטים:

איור 8: מיקומי החיישנים ברסברי פיי

* 1. **יומן הפרויקט**

לאחר שהתגברנו על מכשול הגרסה של RaspberryPi , התקנו את הספרייה העדכנית של OpenCV.

הדפסנו ברקודים על גבי דפי A4 בגדלים של 8.5X8.5 ס"מ, הפעלנו את המצלמה כדי לראות שאכן אנחנו רואים אותם בבירור וכיוונו את הפוקוס במצלמה (באופן ידני) כך שנראה בבירור את הברקוד.

איור 9: "עדשת עין דג"

קליברציה (כיול) של המצלמה

אחת הבעיות הראשונות שאיתן נתקלנו היה שהמצלמה מצלמת בסגנון Fisheye lense "עדשת עין דג" – כינוי לעדשות צילום בעלות אורך מוקד קצר במיוחד, כלומר עדשות "רחבות" מאוד. וזה יוצר עיוות משמעותי של התמונה ואף תמונה "כדורית". עיוות זה יוצר תחושה שהתמונה צולמה מבעד לעיניו של דג מתחת למים.

בשביל לתקן את הבעיה הזאת היינו צריכים לעשות קליברציה (כיול) למצלמה.

בעת כיול מצלמה, קובעים את הפרמטרים הפנימיים של המצלמה, הכוללים מאפיינים כמו אורך מוקד, מרכז אופטי ומקדמי עיוות עדשה.

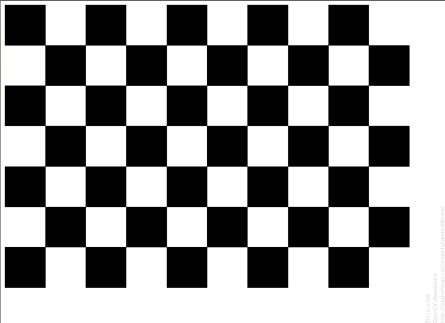
Camera Matrix:

זוהי מטריצה ​​3x3 המייצגת את הפרמטרים הפנימיים של המצלמה. הוא כולל את אורך המוקד (הן בכיווני x והן בכיווני y) ואת המרכז האופטי (הנקודה העיקרית) של התמונה. זה חיוני להפיכת נקודות תלת-ממד לקואורדינטות תמונה דו-ממדיות. עם cameraMatrix, אתה יכול לתקן תמונות, לבצע טרנספורמציות של פרספקטיבה ולתקן עיוות עדשה.

distCoeffs:

מקדמים אלו מייצגים את העיוות שמציגה עדשת המצלמה. עיוות העדשה יכול להיות משני סוגים: עיוות רדיאלי (עיוות חבית או כרית סיכה) ועיוות משיק. מקדמי העיוות עוזרים לתקן את העיוותים הללו, ומאפשרים מדידות מדויקות יותר ואיכות תמונה טובה יותר.

למעשה, cameraMatrix ו- distCoeffs הם מרכיבים חיוניים בכיול המצלמה שכן הם מאפשרים מיפוי מדויק בין עולם התלת-ממד למישור התמונה הדו-ממדית על ידי התחשבות במאפייני המצלמה הפנימיים ועיוותי העדשה.

תהליך הקליברציה מתבצע ע"י הדפסת לוח שחמט על דף A4, והדבקות על משטח ישר וקשיח (אנחנו השתמשנו בגב של קלסר). לתוך הקוד שמבצע את הקליברציה מעבירים את גודל לוח השחמט במ"מ. הקוד יודע לזהות את לוח השחמט, שומר בערך 27 תמונות (זה מספר התמונות המינימלי שצריך) כדי לבצע את הקליברציה באופן מושלם. ואותו קוד יודע לקחת את התמונות הללו, וע"י פונקציה מובנית בOpenCV , הוא יודע לזהות את הפינות של לוח השחמט ולהחזיר את המטריצה של המצלמה והמקדמים.

איור 10: לוח שחמט לצורך קליברציה

לאחר ביצוע חד פעמי של הפעולה הזאת, יש לנו את המקדמים והמטריצה ובכל פעם שנרצה לצלם תמונה כלשהי, נעביר את המטריצה והמקדמים לפונקציה שתפקידה ליישר את התמונה והיא מחזירה לנו תמונה ישרה.

לאחר מכן רצינו לראות איך המצלמה יכולה להראות לנו בזמן אמת את התמונה לאחר היישור. מאחר ואנחנו משתמשים בRaspberryPi4 עם יכולת עיבוד גדולה במיוחד, אנחנו מצליחים להפעיל את המצלמה ולראות בזמן אמת תמונה לאחר יישור (הוא מצליח לקחת את המקדמים ולחשב תוך כדי צילום ולגרום לתמונה להתיישר).

קריאת המידע מן הברקוד

השלב הבא היה להתחיל לפענח את מה שאנחנו רואים. אז תחילה כל מה שרצינו היה לראות איזה מידע הוא קורא מהברקוד, התחלנו לצלם ולהדפיס תוך כדי צילום את המידע שהוא קורא, השתמשנו בברקוד שאת תוכנו אנחנו הפקנו, ואכן הוא הצליח לקרוא ולפענח זאת.

מידע נוסף מן התמונה

התוכנה מוצאת את פינות הברקוד, ומדווחת אותן בצורה של קואורדינאטות הפיקסלים המתאימים.

חישוב מיקום וזווית הפנייה של המצלמה

לאחר מכן השתמשנו בפונקציה של OpenCV שנקראת solvePnP() : (Perspective-n-Point)

על ידי שימוש במיקום פינות הברקוד, הן במציאות והן בתמונה, גיאומטריית הטלה (פרויקציה) ונתוני המצלמה, הפונקציה `()solvePnP`, יודעת לחשב מיקום הברקוד המצולם ביחס למצלמה. על ידי טרנספורמציות נוספות הופכים את החישוב לקבלת המידע הנדרש, שהוא מיקום המצלמה ביחס לברקוד

הפונקציה `solvePnP:

קלט:

* נקודות תלת ממדיות המתאימות לפינות הברקוד, במערכת קואורדינטות שראשיתה במרכז הברקוד (נקודות אובייקט).
* נקודות הדו-ממד המתאימות של הטלת הברקוד על חיישן המצלמה, בתוך התמונה. אלו ניתנות בפיקסלים מפינת התמונה
* מטריצת המצלמה (`cameraMatrix`).
* מקדמי עיוות (`distCoeffs`) לתיקון עיוות העדשה.

פלט:

Rotation Vector & Translation Vector המייצגים את תנוחת האובייקט המצולם (אצלנו הברקוד) ביחס למערכת הקואורדינטות של המצלמה.

* Rotation Vector: וקטור הסיבוב מייצג את סיבוב הברקוד ביחס למערכת הקואורדינאטות של המצלמה. הוא מיוצג בדרך כלל כווקטור תלת מימדי המגדיר ציר סיבוב ואת זווית הסיבוב סביב הציר הזה. וקטור זה יכול לשמש ישירות כדי להפוך נקודות ממערכת הקואורדינטות של המצלמה למערכת הקואורדינטות של הברקוד..
* Translation Vector: מייצג את מיקום האובייקט (אצלנו הברקוד) ביחס למצלמה. הוא מיוצג בדרך כלל כווקטור תלת מימדי המציין את התזוזה של האובייקט לאורך צירי x, y ו-z.. יחד עם וקטור הסיבוב, הוא מתאר במלואו את התנוחה (מיקום וכיוון) של האובייקט במערכת הקואורדינאטות של המצלמה.

הערכים שקיבלנו היו הזזה וסיבוב של הברקוד במערכת הקואורדינאטות של המצלמה. אך אנחנו היינו צריכים את ההזזה והסיבוב של המצלמה, במערכת הקואורדינטות של הברקוד. כדי לחשב זאת ביצענו את התהליך הבא:

לאחר שקיבלנו את ווקטור הרוטציה, הפכנו אותו למטריצת הרוטציה ע"י פונקציה מובנית של OpenCV בשם rodrigues.

חישבנו את המטריצה ההופכית של מטריצת הרוטציה והכפלנו אותה בווקטור ההזזה. וזה נתן לנו את הקואורדינטות של המצלמה במערכת הצירים של הברקוד. ע"י שימוש במטריצת הסיבוב חישבנו את הזוויות yaw, pitch, roll.

השתמשנו בפונקציה מובנית בפייתון שלוקחת את מטריצת הסיבוב, ומשתמשת בטריגונומטריות כדי לחשב את זוויות שלוש זויות הסיבוב המתארות את הכיוון של מערכת הקואורדינטות המסובבת ביחס לזו המקורית.

מייצג את הסיבוב סביב ציר z, Pitch מייצג את הסיבוב סביב ציר הy , מייצג את הסיבוב סביב ציר הx.

לאחר שבנינו את הפונקציות לחישובי המרחקים והזוויות, התחלנו בניסויים. ניסוי הקרוב למציאות היה להתקין את הברקוד על התיקרה ולהסיע את המצלמה במעגל סביב היטל מרכזו על הריצפה. בגלל חוסר הניידות של המצלמה בשלב זה, לא יכולנו לעשות זאת, ובמקום זה ביצענו פעולה אקוויוולנטית, של התקנת הברקוד על הקיר, הצבת המצלמה בצורה נייחת במקום וזוית המתאימים, ואז סיבוב הברקוד סביב מרכזו. לצורך זה

תקענו מסמר באמצע הברקוד על הקיר והנחנו את המצלמה בזווית של 45 מעלות ביחס לברקוד.

סובבנו את התמונה בכל פעם בעוד 30 מעלות ושמרנו אותה. עד שהגענו לאותה זווית שבה התחלנו. לאחר מכן פענחנו את התמונה ע"י הפונקציה שבנינו ורצינו לבדוק אם אכן המיקומים שקיבלנו תואמים את המציאות.

תמונה שמכילה קיר, צילום מסך, אדם, בתוך מבנה

התיאור נוצר באופן אוטומטיתמונה שמכילה אדם, לבוש, קיר, בתוך מבנה

התיאור נוצר באופן אוטומטיאאאא

איור 12: ניסוי מצלמה זווית 30

איור 11: ניסוי מצלמה זווית 0

תמונה שמכילה אדם, לבוש, קיר, בתוך מבנה

התיאור נוצר באופן אוטומטי

איור 14: ניסוי מצלמה זווית 90

איור 13: ניסוי מצלמה זווית 60

להלן התוצאות:

איור 15: תוצאות ניסוי מצלמה

לאחר שקיבלנו את הערכים שרצינו.

המרת מיקום ביחס לברקוד למיקום ביחס לנקודת ייחוס כללית

ניתן להתקדם בפרויקט ולהציב ברקודים לפי נקודות הציון במרחב התנועה. הדפסנו ברקודים שכל המידע שיש בתוכם זה מספר המציין ID של הברקוד. מספר זה משמש כאינדקס לטבלה, שמכילה מידע על הברקוד, כולל מיקום הראשית שלו (OFFSET) במערכת קואורדינאטות כללית ומשותפת לכל הברקודים... כאשר המצלמה קוראת את הברקוד ומפענחת את המספר, המערכת מוסיפה את ה Offset המתאים לקואורדיטות המצלמה שקיבלנו מתוך החישובים, ביחס לברקוד המתאים, וכך מקבלים את מיקום המצלמה ביחס למערכת הקואורדינאטות המשותפת.

הכנסת הנתונים למפת איזור התנועה (בית החולים)

הבדיקה האולטימטיבית של התוצאות היא חיבור שני המודולים של הפרוייקט, העברת התוצאות למודול חישוב הנתיב והימנעות ממכשולים, וצפיה בתנועת הכסא. מכיוון שהחיבור טרם בוצע, היה צורך להציג את התוצאות ולהמחישן בצורה אחרת לצורך זה כתבנו תוכנה אשר מציגה מפה של איזור התנועה (כמו בית החולים), כאשר על המפה מופיע סימן למיקום המצלמה (הכסא), סימן המציין את כיוון הפנייה שלו, וכן מופיעים סימנים למיקום הברקודים השונים, ולמיקום המיכשולים שהתגלו. את הנתונים שקיבלנו אנחנו מכניסים לתוך פונקציה שמעדכנת את המפה. אותה פונקציה היא כללית וניתנת לשינוי ע"י המשתמש. ניתן לשרטט קירות חדרים וגודל קומה. כאשר הכיסא נע מברקוד לברקוד, המפה מתעדכנת בזמן אמת ומראה את מיקומו של הכיסא, את הכיוון שאליו הכיסא פונה ואת המכשולים שעומדים בפניו.

1. **הפרויקט כמקור לימוד ויישום הידע האקדמי**

למדנו דברים חדשים במהלך העבודה שלנו בפרויקט:

* Linux - כדי לעבוד עם כל הספריות איתן עבדנו, שבעזרתן ניתן לצלם ולפענח ברקודים, היינו צריכים להתקין אותם דרך הסביבה של Linux.
* RaspberryPi – למדנו איך להריץ קוד על מחשב זה, שנותן ביצועים גבוהים מאוד ובעל יכולות עיבוד גבוהות.
* גיבוי – למדנו שצריך לגבות כל דבר קטן במהלך הפרויקט. למדנו זאת בדרך הקשה - בזמן העבודה על הפרויקט, הפעלנו פונקציה שמחקה לנו את כל מה שעשינו. לא גיבינו ולכן היינו צריכים להתקין ולכתוב את הכל מההתחלה.
* מצלמה – למדנו מושגים חדשים מעולם הצילום ועיבוד תמונה ממוחשב (Fisheye , מטריצת המצלמה, מקדמי עיוות וכו').

קורסים מהתואר ששימשו אותנו:

* אלגברה ליניארית לצורך הכפלת מטריצות ווקטורים.
* מבוא לבינה מלאכותית ששם למדנו Python.

1. **תכנון מול ביצוע**
   1. **טבלת אבני דרך**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **מס' אבן הדרך** | **תיאור אבן הדרך** | **תאריך סיום** | **סה"כ שעות אדם** | **תוצר מדיד** |
| **1** | דוח מכין | 14/11/2022 | 90 | דוח מכין |
| **2** | דוח התקדמות | 04/05/2023 | 50 | דוח התקדמות של 25 עמוד לפחות |
| **3** | חומרת מצלמה עובדת באופן מלא | 31/01/2024 | 250 | קריאת בקוד והוצאת מיקומים וזוויות רלוונטיות |
| **4** | חומרת חיישנים עובדת באופן מלא | 31/01/2024 | 250 | מדידת מיקומי מכשולים תואמת למציאות |
| **5** | מפה | 31/01/2024 | 400 | תצוגת המיקום ביחס לבית החולים והמכשולים במפה |
| **6** | יום פרויקטים + הגנה + הדגמה מעשית | 05/02/2024 | 170 | מצגת + ספר פרויקט + פרויקט עובד |

טבלה 1: אבני דרך

* 1. **טבלת משימות**

|  |  |
| --- | --- |
| **מס"ד** | **משימה** |
| 1 | פתיחת פרויקט |
| 1.1 | בחירת הפרויקט מבין כל הרשימות |
| 1.2 | פגישה ראשונית עם המנחה להכרות ויצירת אבני דרך |
| 1.3 | כתיבת דו"ח אפיון |
| 1.4 | מחקר וקריאה על הבעיה ופתרונות מוצעים |
| 1.5 | פגישה עם בית החולים וסיור שטח |
| 1.6 | בחירת חיישנים |
| 1.7 | שליחת טיוטה למנחה |
| 2 | עבודה עם החומרה |
| 2.1 | קבלת החומרה |
| 2.2 | הבנת הדרישות מכל רכיב |
| 2.3 | בדיקת דפי נתונים |
| 2.4 | בדיקת התאמה של החומרה לדרישות |
| 2.5 | בנייה ועיצוב תשתית לחומרה |
| 2.6 | קבלת אישור רכישת החומרה החסרה |
| 2.7 | קבלת החומרה החסרה |
| 3 | הפעלת החומרה |
| 3.1 | הפעלת raspberry pi 3 |
| 3.1.1 | אתחול הזיכרון |
| 3.1.2 | הורדת מערכת ההפעלה לזיכרון |
| 3.1.3 | העלאת מערכת ההפעלה ב- raspberry pi 3 |
| 3.2 | הפעלת raspberry pi 4 |
| 3.2.1 | אתחול הזיכרון |
| 3.2.2 | הורדת מערכת ההפעלה לזיכרון |
| 3.2.3 | העלאת מערכת ההפעלה ב- raspberry pi 4 |
| 3.3 | הפעלת חיישני סונאר |
| 3.3.1 | בניית קוד לחיישנים |
| 3.3.2 | בדיקות ושיפור הקוד |
| 3.3.3 | ניסוי חיישנים |
| 3.4 | הפעלת המצלמה |
| 3.4.1 | בניית קוד לזיהוי ברקוד |
| 3.4.2 | בדיקות ושיפור הקוד |
| 3.4.3 | ניסוי מצלמה |
| 4 | דוח התקדמות |
| 4.1 | מעקב אחר תקלות והתקדמות |
| 4.2 | מחקר וקריאת מאמרים נוספים |
| 4.3 | יצירת גאנט |
| 4.4 | כתיבת הדו"ח |
| 4.5 | שליחת טיוטה למנחה |
| 4.6 | הכנת מצגת |
| 5 | שיפורי קוד |
| 5.1 | שיפור התמצאות במסלול |
| 5.2 | זיהוי גודל מכשול |
| 5.3 | זיהוי מכשול נע ומהירותו |
| 6 | פרוטוקול תקשורת |
| 6.1 | בחירת פרוטוקול וקריאת חומרים |
| 6.2 | קבלת חומרים מאחראי המעבדות |
| 6.3 | כתיבת קוד |
| 6.4 | סנכרון בין החומרות |
| 7 | התאמת הפרויקט למיפוי בית החולים |
| 7.1 | שינויי קוד בהתאם לקורדינטות |
| 7.2 | שינויי רזולוציה בהתאם לגובהי התקרות |
| 7.3 | קבלת מיקומי מכשולים קבועים |
| 7.4 | הזנת המכשולים בקוד |
| 8 | התממשקות עם המודול המקביל |
| 8.1 | סנכרון בין הפרויקטים |
| 8.2 | בדיקת פרויקט מלא |
| 9 | התממשקות לתשתיות בית החולים |
| 9.1 | קביעת מיקומי הברקודים בבית החולים |
| 9.2 | יצירת תשתיות בבית החולים |
| 9.3 | ניסוי המודל במסלול הנסיעה בבית החולים |
| 10 | יום פרויקטים + הדגמה מעשית |
| 10.1 | הכנת פוסטר |
| 10.2 | הכנת מצגת |
| 10.3 | הגנות |
| 10.4 | הכנת ספר פרויקט |
| 10.5 | "חזרה גנרלית" |
| 10.6 | הגשת הפרויקט |

טבלה 2: טבלת משימות

1. **מבט לעתיד – הצעות לשיפור הפרויקט**

יכולות שניתן לשפר:

* מכשולים: ניתן לשפר את יכולת גילוי המכשולים ע"י קניית חיישן שסורק את המכשולים הניצבים בפני הכיסא 360 מעלות סביב הכיסא.
* מצלמה: ניתן לשפר את יכולת זיהוי הברקוד ע"י קניית מצלמה עם רזולוציה הרבה יותר גבוהה כדי לזהות ברקוד ממרחק גדול יותר, לדוגמה: תקרה גבוהה מ2 מטר.
* פתיחת דלתות: ניתן לשפר את תנועת הכיסא בתוך בית החולים שלא יהיה רק מחדר לחדר אלא גם כניסה לאותו חדר שבו אמור המטופל לקבל טיפול ע"י מערכת שתוכל לפתוח דלתות אוטומטית.
* מעבר בין דלתות הנפתחות אוטומטית/ כניסה ויציאה ממעלית.

הפרויקט שלנו עוסק בהפחתה בעומס מהעבודה של הסניטרים, כל רעיון יביא להפחתת העומס, וכך נוכל לשפר את מתן הטיפול במטופלים.

1. **נספחים**

נספח א: סכמת בלוקים

נספח ב: קוד קליברציה למצלמה

נספח ג: חישוב המרחק והזווית שבו הכיסא נמצא מהברקוד

נספח ד: פונקציה לחישוב נקודת ציון של הכסא

נספח ה: פונקציה ליצירת המפה והדפסתה בזמן אמת

נספח ו: פונקציה לחישוב מרחקי המכשולים מהחיישנים

נספח ז: הפונקציה הראשית שמפעילה את כל הפונקציות

נספח ח: טבלת מדידות ניסוי חיישנים

נספח ט: טבלת מדידות ניסוי מצלמה

נספח י: רשימת מקורות

1. **נספח א: סכמת בלוקים**

תמונה שמכילה תרשים, טקסט, קו, צילום מסך

התיאור נוצר באופן אוטומטי

איור 16: סכמת בלוקים

* 1. הסבר על סכמת הבלוקים:
     1. התחלה – הכיסא ניצב במרחב.
     2. הפעלה המצלמה – המצלמה מתחילה לפעול ולצלם כל מה שמולה.
     3. זיהוי הברקוד – התוכנה מזהה הברקוד.
     4. פיענוח נקודת הציון + זווית הכיסא ביחס לברקוד – התוכנה מקבלת כקלט את התמונה של הברקוד, ומפענחת אותו, היא מגלה את נקודת הציון של הכיסא ובאיזה זווית היא נמצאת ביחס לברקוד.
     5. הדפסת המידע על המפה – המפה מקבלת כקלט את נקודת הציון ואת הזווית yaw. ומדפיסה את המידע על גבי המפה.
     6. הפעלת החיישנים – החיישנים מתחילים לפעול ע"י שליחת גלי קול.
     7. פיענוח מרחק המכשול מהכיסא – התוכנה מחשבת מרחק של מכשולים מכל המידע שהתקבלו מהחיישנים.
     8. הדפסת המידע על המפה – המפה מקבלת כקלט את זווית המכשול מהכיסא והמרחק מהכיסא ומדפיסה אותו על המפה.

1. **נספח ב: קוד הקליברציה למצלמה**

import numpy as np

import cv2

import glob, os

def calibrate\_cam():

folder = '/home/dn/project/env/lib/python3.9/site-packages/temp/'

# FIND CHESSBOARD CORNERS - OBJECT POINTS AND IMAGE POINTS

chessboardSize = (9, 6)

frameSize = (640, 480)

# termination criteria

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001)

# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)

objp = np.zeros((chessboardSize[0] \* chessboardSize[1], 3), np.float32)

objp[:, :2] = np.mgrid[0:chessboardSize[0], 0:chessboardSize[1]].T.reshape(-1, 2)

size\_of\_chessboard\_squares\_mm = 27 # 3.7 cm

objp = objp \* size\_of\_chessboard\_squares\_mm

# Arrays to store object points and image points from all the images.

objpoints = [] # 3d point in real world space

imgpoints = [] # 2d points in image plane.

images = sorted(glob.glob(os.path.join(folder, '\*.png'))) # Update the file extension as needed

for img\_path in images:

img = cv2.imread(img\_path)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Find the chess board corners

ret, corners = cv2.findChessboardCorners(gray, chessboardSize, cv2.CALIB\_CB\_ADAPTIVE\_THRESH + cv2.CALIB\_CB\_NORMALIZE\_IMAGE)

# If found, add object points, image points (after refining them)

if ret:

objpoints.append(objp)

corners = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)

imgpoints.append(corners)

# Draw and display the corners

cv2.drawChessboardCorners(img, chessboardSize, corners, ret)

cv2.imshow('drawChessboardCorners', img)

cv2.waitKey(1000)

cv2.destroyAllWindows()

# CALIBRATION

ret, cameraMatrix, distCoeffs, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, frameSize, None, None)

print("The calibration was successfully performed :)")

return ret, cameraMatrix, distCoeffs, rvecs, tvecs

1. **נספח ג: חישוב המרחק והזווית שבו הכיסא נמצא מהברקוד (ביחס למערכת צירים של ביה"ח)**

import cv2

import numpy as np

from time import sleep

from location\_function import \*

qr\_code\_side\_length = 380 # in millimeters

cameraMatrix = np.array([[417.10181923, 0, 310.87906165], [0, 420.99094604, 237.63151562], [0, 0, 1]], dtype=np.float32)

distCoeffs = np.array([-0.42033994, 0.10038047, 0.00514208, -0.0069535, 0.05811156], dtype=np.float32)

def qr\_code(image, cameraMatrix, distCoeffs, qr\_code\_side\_length):

# Input image containing the QR code

undistorted\_image = cv2.undistort(image, cameraMatrix, distCoeffs)

# Define the 3D model points of the QR code's corners in real-world coordinates

# Assuming a square QR code with known side length (e.g., 100 mm)

model\_points = np.array([

(-(qr\_code\_side\_length)/2, -(qr\_code\_side\_length)/2, 0), # top-left corner

((qr\_code\_side\_length)/2, -(qr\_code\_side\_length)/2, 0), # top-right corner

((qr\_code\_side\_length)/2, (qr\_code\_side\_length)/2, 0), # bottom-right corner

(-(qr\_code\_side\_length)/2, (qr\_code\_side\_length)/2, 0) # bottom-left corner

], dtype=np.float32)

# Detect and decode the QR code

detector = cv2.QRCodeDetector()

data, points, straight\_qrcode = detector.detectAndDecode(undistorted\_image)

if data:

# Assuming that the QR code has been detected and decoded successfully

# Convert the detected points to the correct format

image\_points = points.squeeze()

# Use solvePnP to estimate the pose of the QR code

\_, rotation\_vector, translation\_vector = cv2.solvePnP(model\_points, image\_points, cameraMatrix, distCoeffs)

# Convert rotation vector to rotation matrix

rotation\_matrix, \_ = cv2.Rodrigues(rotation\_vector)

# Calculate Euler angles (yaw, pitch, roll) in radians

yaw = np.arctan2(rotation\_matrix[1, 0], rotation\_matrix[0, 0])

pitch = np.arctan2(-rotation\_matrix[2, 0], np.sqrt(rotation\_matrix[2, 1]\*\*2 + rotation\_matrix[2, 2]\*\*2))

roll = np.arctan2(rotation\_matrix[2, 1], rotation\_matrix[2, 2])

yaw\_degrees = np.degrees(yaw)

pitch\_degrees = np.degrees(pitch)

roll\_degrees = np.degrees(roll)

# ----------- inverse ----------- #

inverse\_rotation\_matrix = np.linalg.inv(rotation\_matrix)

p\_inv = np.dot(inverse\_rotation\_matrix, translation\_vector)

x\_inv, y\_inv, z\_inv = p\_inv

x\_inv = np.round(x\_inv, 2)

y\_inv = np.round(y\_inv, 2)

z\_inv = np.round(z\_inv, 2)

z\_inv = z\_inv / 1000

[x\_inv, y\_inv] = calc\_location\_hospital(x\_inv, y\_inv, data)

else:

x\_inv = 555

y\_inv = 555

z\_inv = 555

yaw\_degrees = 555

pitch\_degrees = 555

roll\_degrees = 555

data = 555

return x\_inv,y\_inv,z\_inv,yaw\_degrees,pitch\_degrees,roll\_degrees,data

1. **נספח ד: פונקציה לחישוב נקודת ציון של הכסא**

def map\_dictionary(data):

my\_dict = {"1":[1.5,1.375],

"2":[3.0,1.375],

"3":[4.5,1.375],

"4":[6,1.375]}

return my\_dict[data]

def calc\_location\_hospital(x, y, data):

[OffSet\_x,OffSet\_y] = map\_dictionary(data)

map\_x = OffSet\_x + x / 1000

map\_y = OffSet\_y + y / 1000

return [map\_x, map\_y]

1. **נספח ה: פונקציה ליצירת המפה והדפסתה בזמן אמת**

import matplotlib.pyplot as plt

import numpy as np

from time import sleep

def init\_room\_map():

# Display the room map:

# Get the dimensions of the space

x\_dimension, y\_dimension = 10.0, 5.0

# Build the room map

room\_map = build\_room\_map(x\_dimension, y\_dimension)

print("\nRoom Map:")

for item, info in room\_map.items():

print(f"{item}: {info}")

fig, ax = plt.subplots(figsize=(10, 5))

ax.set\_xlim(0, x\_dimension)

ax.set\_ylim(0, y\_dimension)

return ax, room\_map, x\_dimension, y\_dimension

def build\_room\_map(x\_dimension, y\_dimension):

room\_map = {}

room\_map = {

'1': {'x': 1.5-0.19, 'y': 1.375-0.19, 'size': 0.38}, # qrcode 1

'2': {'x': 3.0-0.19, 'y': 1.375-0.19, 'size': 0.38}, # qrcode 2

'3': {'x': 4.5-0.19, 'y': 1.375-0.19, 'size': 0.38}, # qrcode 3

'4': {'x': 6-0.19, 'y': 1.375-0.19, 'size': 0.38}, # qrcode 4

'5': {'x': 7.5-0.19, 'y': 1.375-0.19, 'size': 0.38}, # qrcode 5

'6': {'x': 9-0.19, 'y': 1.375-0.19, 'size': 0.38}, # qrcode 6

'7': {'x': 10.5-0.19, 'y': 1.375-0.19, 'size': 0.38}, # qrcode 7

'8': {'x': 12-0.19, 'y': 1.375-0.19, 'size': 0.38}, # qrcode 8

'9': {'x': 13.5, 'y': 1.375-0.19, 'size': 0.38}, # qrcode 9

'10': {'x': 15-0.19, 'y': 1.375-0.19, 'size': 0.38}, # qrcode 10

'11': {'x': 3.2-0.19, 'y': 2.5-0.19, 'size': 0.38}, # qrcode 11

'12': {'x': 3.2-0.19, 'y': 4-0.19, 'size': 0.38}, # qrcode 12

'13': {'x': 3.2-0.19, 'y': 5.5-0.19, 'size': 0.38}, # qrcode 13

'14': {'x': 3.2-0.19, 'y': 7-0.19, 'size': 0.38}, # qrcode 14

'15': {'x': 15-0.19, 'y': 2.875-0.19, 'size': 0.38}, # qrcode 15

'16': {'x': 13.5-0.19, 'y': 2.875-0.19, 'size': 0.38}, # qrcode 16

'Wall1' : {'start\_x': 0.0, 'start\_y': 0.0, 'end\_x': 16.5, 'end\_y': 0.0},

'Wall2' : {'start\_x': 16.5, 'start\_y': 0.0, 'end\_x': 16.5, 'end\_y': 4.95},

'Wall3' : {'start\_x': 11.4, 'start\_y': 4.95, 'end\_x': 16.5, 'end\_y': 4.95},

'Wall4' : {'start\_x': 11.4, 'start\_y': 2.75, 'end\_x': 11.4, 'end\_y': 4.95},

'Wall5' : {'start\_x': 3.95, 'start\_y': 2.75, 'end\_x': 11.4, 'end\_y': 2.75},

'Wall6' : {'start\_x': 3.95, 'start\_y': 2.75, 'end\_x': 3.95, 'end\_y': 8.5},

'Wall7' : {'start\_x': 2.45, 'start\_y': 8.5, 'end\_x': 3.95, 'end\_y': 8.5},

'Wall8' : {'start\_x': 2.45, 'start\_y': 2.75, 'end\_x': 2.45, 'end\_y': 8.5},

'Wall9' : {'start\_x': 0.0, 'start\_y': 2.75, 'end\_x': 2.45, 'end\_y': 2.75},

'Wall10': {'start\_x': 0.0, 'start\_y': 0.0, 'end\_x': 0.0, 'end\_y': 2.75}

}

return room\_map

def visualize\_room\_map(room\_map, x\_dimension, y\_dimension, ax=None):

if ax is None:

# Create a figure and axis representing the space

fig, ax = plt.subplots(figsize=(x\_dimension, y\_dimension))

ax.set\_xlim(0, x\_dimension)

ax.set\_ylim(0, y\_dimension)

# Plotting the room positions, QR codes, and walls

for item, info in room\_map.items():

if item == 'SpaceSize':

# Skip plotting the space size

continue

if 'size' in info:

# Plot a QR code for each added QR code

rect = plt.Rectangle((info['x'], info['y']), info['size'], info['size'], edgecolor='black',

facecolor='darkorange', alpha=0.7)

ax.add\_patch(rect)

ax.text(info['x'] + info['size'] / 2, info['y'] + info['size'] / 2, item, ha='center', va='center',

color='black')

elif 'x\_size' in info:

# Plot a rectangle for each room

rect = plt.Rectangle((info['x'], info['y']), info['x\_size'], info['y\_size'], edgecolor='black',

facecolor='xkcd:sky blue', alpha=0.5)

ax.add\_patch(rect)

ax.text(info['x'] + info['x\_size'] / 2, info['y'] + info['y\_size'] / 2, item, ha='center', va='center')

elif 'start\_x' in info:

# Plot a line for each wall

ax.plot([info['start\_x'], info['end\_x']], [info['start\_y'], info['end\_y']], color='black', linewidth=2)

# Create a legend

handles = [

plt.Rectangle((0, 0), 1, 1, color='darkorange', alpha=0.7, ec='black'),

plt.Line2D([0, 0], [0, 1], color='black', linewidth=2)]

labels = ['QR Code', 'Wall']

ax.legend(handles, labels, loc='upper left')

plt.grid(True)

return ax

def visualize\_chair(ax, x, y, direction\_degrees, distance1, distance2, distance3, distance4, distance5, distance6,

distance7, distance8):

# Plot a blue dot for the chair

ax.plot(x, y, 'bo', markersize=13)

line\_length = 2

# Calculate and plot the position at 0 degrees

x\_distance0 = x + line\_length \* np.cos(np.radians(direction\_degrees + 90))

y\_distance0 = y + line\_length \* np.sin(np.radians(direction\_degrees + 90))

ax.plot([x, x\_distance0], [y, y\_distance0], marker='o', linewidth=2.5, color='blue')

# Calculate and plot the position of the obstacle at 157.5 degrees

obstacle\_x\_distance1 = x + distance1 \* np.cos(np.radians(direction\_degrees + 157.5))

obstacle\_y\_distance1 = y + distance1 \* np.sin(np.radians(direction\_degrees + 157.5))

ax.plot(obstacle\_x\_distance1, obstacle\_y\_distance1, 'ro', markersize=7)

# Calculate and plot the position of the obstacle at 127.5 degrees

obstacle\_x\_distance2 = x + distance2 \* np.cos(np.radians(direction\_degrees + 127.5))

obstacle\_y\_distance2 = y + distance2 \* np.sin(np.radians(direction\_degrees + 127.5))

ax.plot(obstacle\_x\_distance2, obstacle\_y\_distance2, 'ro', markersize=7)

# Calculate and plot the position of the obstacle at 97.5 degrees

obstacle\_x\_distance3 = x + distance3 \* np.cos(np.radians(direction\_degrees + 97.5))

obstacle\_y\_distance3 = y + distance3 \* np.sin(np.radians(direction\_degrees + 97.5))

ax.plot(obstacle\_x\_distance3, obstacle\_y\_distance3, 'ro', markersize=7)

# Calculate and plot the position of the obstacle at 82.5 degrees

obstacle\_x\_distance4 = x + distance4 \* np.cos(np.radians(direction\_degrees + 82.5))

obstacle\_y\_distance4 = y + distance4 \* np.sin(np.radians(direction\_degrees + 82.5))

ax.plot(obstacle\_x\_distance4, obstacle\_y\_distance4, 'ro', markersize=7)

# Calculate and plot the position of the obstacle at 54.5 degrees

obstacle\_x\_distance5 = x + distance5 \* np.cos(np.radians(direction\_degrees + 54.5))

obstacle\_y\_distance5 = y + distance5 \* np.sin(np.radians(direction\_degrees + 54.5))

ax.plot(obstacle\_x\_distance5, obstacle\_y\_distance5, 'ro', markersize=7)

# Calculate and plot the position of the obstacle at 22.5 degrees

obstacle\_x\_distance6 = x + distance6 \* np.cos(np.radians(direction\_degrees + 22.5))

obstacle\_y\_distance6 = y + distance6 \* np.sin(np.radians(direction\_degrees + 22.5))

ax.plot(obstacle\_x\_distance6, obstacle\_y\_distance6, 'ro', markersize=7)

# Calculate and plot the position of the obstacle at -82.5 degrees

obstacle\_x\_distance7 = x + distance7 \* np.cos(np.radians(direction\_degrees - 82.5))

obstacle\_y\_distance7 = y + distance7 \* np.sin(np.radians(direction\_degrees - 82.5))

ax.plot(obstacle\_x\_distance7, obstacle\_y\_distance7, 'ro', markersize=7)

# Calculate and plot the position of the obstacle at -97.5 degrees

obstacle\_x\_distance8 = x + distance8 \* np.cos(np.radians(direction\_degrees - 97.5))

obstacle\_y\_distance8 = y + distance8 \* np.sin(np.radians(direction\_degrees - 97.5))

ax.plot(obstacle\_x\_distance8, obstacle\_y\_distance8, 'ro', markersize=7)

plt.draw()

def clear\_and\_redraw(ax, room\_map, x\_dimension, y\_dimension, x\_coordinate, y\_coordinate, direction\_degrees, distance1, distance2, distance3, distance4, distance5, distance6, distance7, distance8):

plt.ion()

ax.clear()

ax = visualize\_room\_map(room\_map, x\_dimension, y\_dimension, ax) # Redraw the room map

visualize\_chair(ax, x\_coordinate, y\_coordinate, direction\_degrees, distance1, distance2, distance3, distance4, distance5, distance6, distance7, distance8) # Visualize the current chair point with direction arrow and obstacles

plt.pause(0.1) # Pause to allow the plot to update

plt.ioff() # Turn off interactive mode

1. **נספח ו: פונקציה לחישוב מרחקי המכשולים מהחיישנים**

def distance(MAX\_DISTANCE, sensor1, sensor2, sensor3, sensor4, sensor5, sensor6, sensor7, sensor8):

sensor\_array = [sensor1, sensor2, sensor3, sensor4, sensor5, sensor6, sensor7, sensor8]

distance = [0, 0, 0 ,0 ,0 ,0 ,0 ,0]

for i in range(8):

if (i != 2):

if sensor\_array[i].distance is not None:

distance[i] = sensor\_array[i].distance

if distance[i] >= MAX\_DISTANCE:

distance[i] = 555

else :

distance[i] = 0

print("Error with sensor #{}".format(i))

#print("distance{} = {}".format(i+1,distance[i]))

return distance

1. **נספח ז: הפונקציה הראשית שמפעילה את כל הפונקציות**

from Sonar import distance

from time import sleep

from decimal import Decimal, ROUND\_HALF\_UP

import sys

import numpy as np

import cv2

import glob, os

from camera\_main import \*

from map\_functions import \*

from location\_function import \*

import os

os.environ['QT\_QPA\_PLATFORM\_PLUGIN\_PATH'] = '/path/to/your/python/site-packages/Qt/plugins/platforms/'

MAX\_DISTANCE = 400

sonar1\_degree = 67.5

sonar2\_degree = 37.5

sonar3\_degree = 7.5

sonar4\_degree = 352.5

sonar5\_degree = 322.5

sonar6\_degree = 292.5

sonar7\_degree = 187.5

sonar8\_degree = 172.5

sensor1 = DistanceSensor(echo=18, trigger=17, max\_distance=MAX\_DISTANCE)

sensor2 = DistanceSensor(echo=22, trigger=27, max\_distance=MAX\_DISTANCE)

sensor3 = DistanceSensor(echo=24, trigger=23, max\_distance=MAX\_DISTANCE)

sensor4 = DistanceSensor(echo=5, trigger=25, max\_distance=MAX\_DISTANCE)

sensor5 = DistanceSensor(echo=12, trigger=6, max\_distance=MAX\_DISTANCE)

sensor6 = DistanceSensor(echo=19, trigger=13, max\_distance=MAX\_DISTANCE)

sensor7 = DistanceSensor(echo=20, trigger=16, max\_distance=MAX\_DISTANCE)

sensor8 = DistanceSensor(echo=21, trigger=26, max\_distance=MAX\_DISTANCE)

camera = cv2.VideoCapture(0)

qr\_code\_side\_length = 380 # in millimeters

cameraMatrix = np.array([[417.10181923, 0, 310.87906165], [0, 420.99094604, 237.63151562], [0, 0, 1]], dtype=np.float32)

distCoeffs = np.array([-0.42033994, 0.10038047, 0.00514208, -0.0069535, 0.05811156], dtype=np.float32)

[ax, room\_map, x\_dimension, y\_dimension] = init\_room\_map()

saved\_x = 0

saved\_y = 0

saved\_yaw = 0

try:

while True:

[distance1, distance2, distance3, distance4, distance5, distance6, distance7, distance8] = distance(MAX\_DISTANCE, sensor1, sensor2, sensor3, sensor4, sensor5, sensor6, sensor7, sensor8)

sleep(0.1)

ret, image = camera.read()

[x,y,z,yaw,pitch,roll,data] = qr\_code(image, cameraMatrix, distCoeffs, qr\_code\_side\_length)

if data != 555:

clear\_and\_redraw(ax, room\_map, x\_dimension, y\_dimension, x, y, float(yaw), distance1, distance2, distance3, distance4, distance5, distance6, distance7, distance8)

print(data, x, y, int(yaw), "{:.2f}".format(distance1) , "{:.2f}".format(distance2), "{:.2f}".format(distance3), "{:.2f}".format(distance4), "{:.2f}".format(distance5), "{:.2f}".format(distance6), "{:.2f}".format(distance7), "{:.2f}".format(distance8))

saved\_x = x

saved\_y = y

saved\_yaw = float(yaw)

else:

clear\_and\_redraw(ax, room\_map, x\_dimension, y\_dimension, saved\_x, saved\_y, saved\_yaw, distance1, distance2, distance3, distance4, distance5, distance6, distance7, distance8)

print('-', '-', '-', '-', "{:.2f}".format(distance1) , "{:.2f}".format(distance2), "{:.2f}".format(distance3), "{:.2f}".format(distance4), "{:.2f}".format(distance5), "{:.2f}".format(distance6), "{:.2f}".format(distance7), "{:.2f}".format(distance8))

except KeyboardInterrupt:

print('interrupted!')

1. **נספח ח: טבלת מדידות ניסוי חיישנים**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **סוג החומר** | **בד** | **אדם** | **מתכת** | **קרטון** |
| 0.05 | 0.05 | 0.05 | 0.05 | 0.05 |
| 0.10 | 0.11 | 0.1 | 0.1 | 0.1 |
| 0.15 | 0.17 | 0.15 | 0.15 | 0.15 |
| 0.20 | 0.2 | 0.2 | 0.2 | 0.2 |
| 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |
| 0.30 | 0.3 | 0.3 | 0.3 | 0.3 |
| 0.35 | 0.35 | 0.35 | 0.35 | 0.35 |
| 0.40 | 0.42 | 0.4 | 0.4 | 0.4 |
| 0.45 | 0.43 | 0.45 | 0.45 | 0.45 |
| 0.50 | 0.5 | 0.5 | 0.5 | 0.5 |
| 0.55 | 0.53 | 0.55 | 0.55 | 0.54 |
| 0.60 | 0.55 | 0.6 | 0.6 | 0.58 |
| 0.65 | 0.63 | 0.65 | 0.65 | 0.63 |
| 0.70 | 0.71 | 0.7 | 0.7 | 0.7 |
| 0.75 | 0.74 | 0.74 | 0.75 | 0.75 |
| 0.80 | 0.81 | 0.8 | 0.8 | 0.8 |
| 0.85 | 0.86 | 0.84 | 0.86 | 0.84 |
| 0.90 | 0.95 | 0.9 | 0.9 | 0.89 |
| 0.95 | 1.02 | 0.95 | 0.94 | 0.95 |
| 1.00 | 1.09 | 1 | 1.01 | 0.99 |

*טבלה 3: טבלת מדידות ניסוי חיישנים*

1. **נספח ט: טבלת מדידות ניסוי מצלמה**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x\_n | y\_n | z\_n | yaw | pitch | roll |
| 48.14 | -420.68 | -800.2 | -178.22 | 1.66 | 26.54 |
| -157.71 | -376.62 | -805.57 | -151.89 | -10.8 | 23.37 |
| -332.44 | -256.58 | -804.59 | -122.85 | -21.34 | 15.7 |
| -392.94 | -8.01 | -830.89 | -90.48 | -24.32 | -1.12 |
| -351.91 | 192.36 | -810.61 | -57.97 | -21.35 | -14.16 |
| -206.07 | 325.62 | -829.18 | -28.79 | -11.33 | -21.32 |
| -48.69 | 422.13 | -808.57 | -0.08 | -1.69 | -26.62 |
| 130.22 | 371.45 | -823.04 | 28.69 | 8.95 | -22.84 |
| 330.78 | 256.12 | -800.41 | 57.44 | 21.28 | -16.02 |
| 417.93 | 30.19 | -800.88 | 89.66 | 26.53 | -0.63 |
| 387.2 | -188.85 | -796.22 | 122.35 | 23.68 | 13.88 |
| 251.41 | -355.33 | -797.57 | 152.86 | 14.4 | 23.58 |
| 48.14 | -420.68 | -800.2 | 178.22 | 1.66 | 26.54 |

טבלה 4: טבלת מדידות ניסוי מצלמה

1. **רשימת מקורות**

|  |  |
| --- | --- |
| **[1]** | אתר דבר, כתבה משנת 2016 <https://www.davar1.co.il/22882/> |
| **[2]** | <https://www.alljobs.co.il/User/SalaryCompound/position/?JobCategoryID=1724> alljobs |
| **[3]** | רשת 13, כתבה משנת 2017 <https://13tv.co.il/item/online/gadgets/ntr-1265179/> |
| **[4]** | <https://www.aisrael.org/?CategoryID=1340&ArticleID=57714> |
| **[5]** | <https://www.ynet.co.il/articles/0,7340,L-4214495,00.html> |
| **[6]** | <https://www.bizportal.co.il/list/tags/18062> |
| **[7]** | <https://www.youtube.com/watch?v=otztORT2zxk> |
| **[8]** | [Projects | Computer coding for kids and teens | Raspberry Pi](https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2) |
| **[9]** | <https://www.waveshare.com/wiki/7inch_HDMI_LCD_(C)> |
| **[10]** | <https://www.realvnc.com/en/connect/download/viewer/> |