f חוות דעת מנח ה

שם הפרויק ט – רחפן חצי אוטונומ י

בפרויקט זה פותח רחפן להפעלה בתוך מנהרות. הרחפן מתבסס על שני מצבי טיסה, אחד ידני לשליטה מרחוק כאשר טווח התקשורת מאפשר קשר עם המפעיל ושני אוטונומי על בסיס חיישנים. גם בשלב הידני, בגלל סביבת ההטסה האתגרית, יש צורך באמצעי עזר למטיס. הרחפן שתוכנן ונבנה במסגרת הפרויקט מתבסס על מודולים קיימים )כגון מסגרת קרבון, בק ר-טיסה מנועים ועוד( .התכנון דרש ידע רחב בתחום הרחפנים וניסיון שנרכש בהפעלת הציוד הרלוונטי.

במהלך השנה עמי ת בנה את הרחפן וביצע ניסויי טיסה רבים כדי לכיי ל את המערכות של ו בהתאם למטרות הפרויקט .הפיתוח כלל ג ם חיבור למחשב טיסה בו מומשו אלגוריתמים הניווט, יחד עם אמ צעי עזר למטיס )כלומר, אמצעים המסייעים להטסה ידנית בתווך אתגרי להטסה(. הפיתוח גם דרש תכנון והדפסה של חלקים ששולבו ברחפן. בנוסף בוצעו ניסויי טיסה רבים שאפשרות תובנות חשובות על הציוד שנבחר והאופן בו יש להמשיך את הפיתוח אחרי סיום הפרויקט הנוכחי.

מנחה: שי ארוגט י

תאריך: 24/09/04

חתימה: 

**§אוניברסיטת בן-גוריון בנג ב**

**הפקולטה למדעי ההנדס ה**

**המחלקה להנדסת מכונו ת**

**דו"ח פרויק ט**

# רחפן חצי אוטונומי

**24-102**

**שם הסטודנט: עמית גדג '207653965 תאריך הגשה: 24/80/27**

**שם המנחה: שי ארוגט י**

## אב תשפ"ד אוגוסט 2024

 **Ben Gurion University**

**Faculty of Science Engineering**

**Mechanical Engineering Department**

## Final Project Report

**Half Autonomous Drone**

**24-102**

**Author: Amit Gedj 207653965**

**Date: 27/08/2024**

**Supervisor: Shai Arogeti**

**September 2024 Av 5784**

## תקציר

במסגרת מלחמת חרבות ברזל עלה הצורך באמצעי המאפשר סריקה של מנהרות ללא סיכון של חיי אדם. כיום, ישנם מספר רחפנים בשוק הנותנים מענה לצורך ז ה ופרויקט זה מתחרה ברחפנים אלו ואף מציע פונקציות נוספות. פרויקט זה מתמקד בבניית רחפן המסוגל לטוס על ידי מפעיל ובנוסף לטוס בצורה אוטונומית. נקודת המעבר בין שתי אפשרויות ההטסה הינה נקודת איבוד התקשורת של המפעיל עם הרחפן.

בנקודת איבוד התקשורת הרחפן ימשיך טיסה וסריקה במנהרות עד הנקודה בה הסוללה לא תאפשר לו תעופה מלאה חזרה אל נקודת התקשורת האחרונה עם המפעיל. יכולת הטיסה האוטונומית מאפשרת חקירה מעמיקה יותר של המנהרות, דבר אשר כיום מוגב ל על ידי טווח הקליטה של המפעיל עם הרחפן.

הפיתוח בפרויקט זה עסק בבניית פלטפורמה ניסויית המאפשרת הטסה נוחה ובעלת חיישנים המאפשרים פיתוח של מערכות עזר המאפשרות שמירה על מיקום הרחפן במידה ולא מתקבלות פקודות מהמפעיל. עבו ר פלטפורמה זו ייכתבו אלגוריתמים רלוונטיים לפרויק ט ובדיקת היתכנות של הרעיון. הפלטפורמה מורכבת ממספר רכיבים בסיסיים אשר חיבורם ביחד מאפשר את הטיסה הבסיסית של הרחפן- שלדה, בקר טיסה, חיישן 𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤, מנועים ופרופלורים .מערכת ההטסה של הרחפן מבוססת על תוכנה קיימת בשם 𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡, תוכנה זו מבצעת את כלל חוגי הבקרה והשיערוך הנדרשים לטובת הטסת הרחפ ן ושליטה בחיישנים נוספים.

על מנת לאפשר את ההטסה האוטונומית, על הפלטפורמה מורכבים רכיבים נוספים- חיישני 𝐿𝑖𝐷𝐴𝑅 ומחשב זעיר. שימוש בתוכנה זו מאפשר התמקדות בחלק החדשני בפרויקט זה. המחשב מקבל את הנתונים מחיישני ה-𝐿𝑖𝐷𝐴𝑅, מבצע עיבוד נתונים ושולח לבקר טיסה פקודות הטסה. כלל הרכיבים המוזכרים מעלה חוברו למערכת אחת המאפשרת את בחינת הרעיון המוצג בפרויקט .

לאחר הרכבת הפלטפורמה שתכלול את כלל הרכיבים הנדרשים לרחפן, כולל קביעת הגברי בקרה המספקים יכולות טיסה יציבות של הרחפן עם אוסילציות מינימליות ותגובות מהירות לפקודות המפעיל, יחד עם חיבור של החיישנים הנוספים- 𝐿𝑖𝐷𝐴𝑅, 𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤. חיישנים המספקים את המדידות הנדרשות על מנת להטמיע את היכולות האוטונומיות הנדרשות. נכתבו מספר אלגוריתמים אשר מהווים שלב ראשון בפיתוח רחפן אוטונומי. האלגוריתמים שנכתבו בפרויקט זה נועדו להוכי ח היתכנות של רחפן מסוג זה , אשר יכול להגדיל את טווח הטיסה, ללא סיכון כוח אדם, ואינם אופטימליים. על מנת להגיע למוצר סופי נדרש לבצע אופטימיזציה של חלק מהאלגוריתמים .

## Abstract

As part of the Iron Sword War, the need arose for a means of scanning tunnels without endangering human life. Today, there are several drones on the market that meet this need, and this project competes with these drones and even offers additional functions. This project focuses on building a drone capable of flying by an operator and flying autonomously. The crossing point between the two flight options is the operator's point of loss of communication with the drone. At the point of loss of communication, the drone will continue flying and scanning the tunnels to the point where the battery will not allow it to fully fly back to the last point of communication with the operator. The autonomous flight capability enables deeper exploration of the tunnels, which is currently limited by the operator's range with the drone.

The development in this project dealt with the construction of an experimental platform that enables comfortable flight and has sensors that enable the development of auxiliary systems that enable maintaining the drone's position if no commands are received from the operator. For this platform, relevant algorithms for the project and a feasibility study of the idea will be written. The platform consists of a few basic components whose connection together enables the drone's basic flight – chassis, flight controller, optical flow sensor, engines and propellers. The drone's flight system is based on existing software called ArduPilot, which performs all the control and evaluation circles required to fly the drone and control additional sensors.

In order to enable autonomous flight, additional components are assembled on the platform – LiDAR sensors and a miniature computer. Using this software allows focusing on the innovative part of this project. The computer receives the data from the LiDAR sensors, performs data processing, and sends flight commands to the flight controller. All the components mentioned above were connected to a single system that enables the examination of the idea presented in the project.

After assembling the platform that will include all the components required for the drone, including setting control amplifiers that provide stable flight capabilities of the drone with minimal oscillations and quick responses to the operator's commands, along with the connection of the additional sensors - LiDAR, Optical Flow. Sensors that provide the measurements required to implement the required autonomous capabilities. Several algorithms have been written, which constitute the first stage in the development of an autonomous drone. The algorithms written in this project are designed to prove the feasibility of this type of drone, which can increase flight range, without personnel risk, and are not optimal. To reach a final product, it is necessary to optimize some of the algorithms.

## תוכן עניינים

1. מבו א ..................................................................................................................................... 1
   1. תיאור הפרויקט .............................................................................................................. 1
2. רקע ....................................................................................................................................... 2
   1. כלי טיס מסוג רב-לה ב ..................................................................................................... 2
   2. מודל קינמט י .................................................................................................................. 2
   3. המודל הדינמ י ................................................................................................................ 4
   4. חיישני ם......................................................................................................................... 5
3. ................................................................................................................ IMU .2.4.1
4. ............................................................................................................ LiDAR .2.4.2
5. ................................................................................................... Optical Flow .2.4.3
   1. מנועים ללא מברשו ת....................................................................................................... 6
   2. חיבורי תקשור ת ............................................................................................................. 7
6. ............................................................................................................. UART .2.6.1
7. .................................................................................................................. I2C .2.6.2
8. ................................................................................... 𝐿𝑖 − 𝑖𝑜𝑛 וסוללת 𝐿𝑖 − 𝑃𝑜 סוללת .2.7

2.8. תוכנות שליטה ברב-להב .................................................................................................. 8

1. .......................................................................................................... DroneKit ספריית .2.9

2.10. ניוו ט ............................................................................................................................. 9

1. .............................. :𝑆𝐿𝐴𝑀 − 𝑆𝑖𝑚𝑢𝑙𝑡𝑎𝑛𝑒𝑜𝑢𝑠 𝐿𝑜𝑐𝑎𝑙𝑖𝑧𝑎𝑡𝑖𝑜𝑛 𝑎𝑛𝑑 𝑀𝑎𝑝𝑝𝑖𝑛𝑔 .2.10.1

9 ............................................................... :𝑉𝐹𝐻 − 𝑉𝑒𝑐𝑡𝑜𝑟 𝐹𝑖𝑒𝑙𝑑 𝐻𝑖𝑠𝑡𝑜𝑔𝑟𝑎𝑚 .2.10.2

1. דרישות ומטרו ת ...................................................................................................................... 9
   1. מטרות הפרויק ט............................................................................................................. 9
   2. דרישות הפרויקט ............................................................................................................ 9
2. שלבי הפיתו ח ........................................................................................................................ 11
   1. הרכבת הרחפ ן .............................................................................................................. 11
      1. מגן סולל ה ....................................................................................................... 14
      2. רגלי הגבה ה ..................................................................................................... 14
   2. סכמה אלקטרוני ת ........................................................................................................ 16
   3. פרמטרים חשובים בתוכנת ArduPilot ............................................................................ 16
      1. פרמטרי משערך קלמ ן ....................................................................................... 16
3. ............................................................................................... 𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤 .4.3.2

4.3.3. פקודות למצב טיסה 𝐺𝑈𝐼𝐷𝐸𝐷.......................................................................... 17

4.4. פקודות הניתנות לשליחה למערכ ת .................................................................................. 18

1. ............................................................................................................. 𝐿𝑖𝐷𝐴𝑅 חיישני .4.5
   1. מערכת עזר- שמירת מרחק מקיר .................................................................................... 19
   2. מערכת הניוו ט .............................................................................................................. 19
   3. אלגוריתמי סיוע וניוו ט .................................................................................................. 20
      1. שמירת מרחק מינימל י ...................................................................................... 20
      2. קביעת עוגנים במהלך הניוו ט ............................................................................. 21
      3. ניווט ............................................................................................................... 23
2. סיכו ם .................................................................................................................................. 24
3. המלצות להמש ך .................................................................................................................... 26

6.1. הוספת חיישני 𝐿𝑖𝐷𝐴𝑅 .................................................................................................. 26

26 ...................................................................................... 𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤 הוספת חיישני .6.2

6.3. שיפור אלגוריתם הניוו ט ................................................................................................ 26

6.4. התערבות אקטיבית בפקודות המפעיל ............................................................................. 27

1. ביבליוגרפיה .......................................................................................................................... 27
2. נספחי ם ................................................................................................................................ 28
   1. נספח א'- קטעי קוד ....................................................................................................... 28
      1. קטע קוד- איגו ד פקודות לרחפן .......................................................................... 28
      2. קטע קוד- הפעלת חיישני 𝐿𝑖𝐷𝐴𝑅 ...................................................................... 38

## רשימת איורים

איור 1 - קונפיגורציות גוף שונות וכיוון סיבוב המנועים...............................................................2

איור 2 - מערכות הצירים המוגדרות עבור מודל זה......................................................................3

איור 3 - תיאור של החיישנים והמדידות השונות מחיישן IMU.....................................................5

איור 4 - אופן הפעולה של חיישן 𝐿𝑖𝐷𝐴𝑅...................................................................................6 איור 5 - חיישן 𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤 בסיסי בעל חיבור 𝑈𝐴𝑅𝑇............................................................6 איור 6 - סכמה של מנוע ללא מברשות........................................................................................7

איור 7 - איור סכמטי של חיבור רכיבים על ידי חיבור מסוג 𝑈𝐴𝑅𝑇...............................................7

איור 8 - איור סכמטי של חיבור מספר רכיבים לבקר אחד............................................................7 איור 9 - תוצאה רצויה מהשוואת 𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤 ל- 𝐺𝑦𝑟𝑜.......................................................11

12...........................................................𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤 𝑀𝑖𝑐𝑜𝑎𝑖𝑟 𝑀𝑇𝐹 − 01 איור 10 - חיישן

12...........................................𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤 לערכי החיישן 𝐺𝑦𝑟𝑜-איור 11 - השוואה בין ערכי ה

איור 12 - השוואה בין ערכים מכיולים......................................................................................12

איור 13 - מגן סוללה..............................................................................................................14

איור 14 - הרכבת מגן סוללה................. ...................................................................................14

איור 15 - רגלי הרחפן..................................................... .........................................................15

איור 16 – רגלי הרחפן מורכבות................................................................................................15

איור 17 - סכמה המציגה את החיבורים החשמליים במערכת.......................................................16

איור 18 – מבנה פקודה למהירות קווית.....................................................................................18

## רשימת טבלאות

**טבלה 1 – רשימת דרישות הפרויקט**..................................................................................10 **טבלה 2 – רשימת רכיבים שנבחרו לאחר ניסויים**................................................................13

## רשימת סימנים

תיאור מילול י

ן

סימ

ה

מיד

יחידות

𝜓

𝑟𝑎𝑑 זווית סבסוב- עול ם

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| זווית עלרוד- עולם | 𝑟𝑎𝑑 | | 𝜃 | |
| זווית גלגול- עול ם | 𝑟𝑎𝑑 | | 𝜙 | |
| מהירות סיבוב גלגול- גו ף | 𝑟𝑎𝑑 | | 𝑟 | |
| מהירות סיבוב עלרוד- גו ף | 𝑟𝑎𝑑 | | 𝑝 | |
| מהירות סיבוב סבסוב- גו ף | 𝑟𝑎𝑑 | | 𝑞 | |
| מהירות סיבוב מנו ע | 𝑟𝑎𝑑    𝑠𝑒𝑐 | | 𝜔 | |
| כוח דח ף | 𝑁 | | 𝐹 | |
| מומנט  מרחק המנוע ממרכז הכוב ד מקדם המנוע למומנ ט מקדם המנוע לכו ח | | | 𝑁 ⋅ 𝑚  𝑚  𝑘𝑔 ⋅ 𝑚2  𝑘𝑔 ⋅ 𝑚2 | | 𝜏  𝑙  𝑘    𝑏 | |
| מתח הסולל ה | | | 𝑉 | | 𝑉 | |

## 1. מבוא

### 1.1. תיאור הפרויק ט

כלי טיס בלתי מאוישים, או בשפה הצבאית המקובלת- כלי טייס מאוישים מרחוק )כטמ"מ(, הינם רובוטים הלוקחים חלק משמעותי בחיי היום- יום של מסגרות רבות בעולם. גופים פרטיים משתמשים בכטמ"מים לטובת אבטחת משרדים, תחרויות הטסות, צילום וכו'. גופים ביטחוניים משתמשים בכטמ"מים לטוב ת אבטח ת כוחות רגליים בשטח, סריקת מבנים ושטחים מוסתרים לעין כוחותינו. כאשר טווח השימושים כה רחב, קיימים נותני שירות רבים אשר מספקי ם מענה לתחומים נקודתיים ועל כן, לא לכל צורך קיים מענה.

אחד מהתחומים אשר המענה בהם דל הינו רחפן אוטונומי במרחבים סגורים.

רוב הרחפנים אשר נמצאים כיום בשוק מיועדים לשימוש חיצוני, בשטחים פתוחים. היצרניות הגדולות של כלי הטיס, כגון 𝑫𝑱𝑰, משתמשות בחיישנים מתקדמים על מנת לאפשר יציבות של כלי הטיס. החיסרון העיקרי של חיישנים אלו הינו ההתבססות על תמונות וזיהוי שינוי חלקיקים במרחב. עבור תווך סגור אשר רווי באבק עקרון פעולה זה הינו בעייתי ולא נותן פתרון אופטימלי לייצוב הרחפן. כיום, מנהרות הן כלי בו נעשה שימוש בתחומים רבים ובין היתר בתחום הצבאי.

כתוצאה מכך שהמנהרות נמצאות מתחת לפני השטח, חקירת תוואי זה על ידי גורם אנושי מהווה סכנה ושהיה בתנאים מאתגרים בהם, בין היתר, תקשורת מוגבלת עם העולם החיצוני. עולם המנהור הינו עולם רווי מכשולים ואתגרים מכיוון שאין מבנה מוגדר של מנהרה. מנהרות שונות יכולות להיו ת בגדלים שונים, עם חסימות שונות ועם תשתיות שונות העוברות בהן. כמו כן, שהיה במנהרות עלולה לכלול תנאי סביבה קיצוניים כגון, מחסור בחמצן או טמפרטורות גבוהות. כתוצאה מהקשיים אשר הוצגו קודם לכן, ישנן סכנות רבות בתחום המהוות סכנת חיים ועקב כך שהתקשורת עם העולם החיצון הינה מוגבלת, אפשרויות החילוץ גם הן מוגבלות.

על מנת להתמודד עם בעיה זו ועם בעיות נוספות הקיימות בשדה הקרב, נבחנת האופציה של פיתוח רחפן המבצע חקירה וסקירה מוקדמת של המנהרות. רחפן זה מיועד לטוס בהפעלה ידנית על ידי מפעיל עד לנקודת איבוד התקשורת בין השניים ולאחר מכן הרחפן ימשיך לנוע בצורה אוטונומית, עד הנקודה הרחוקה ביותר שאליה הסוללה מאפשרת להגיע ולאפשר תנועה חזרה לנקודת התקשורת. תנועת הרחפן תתבסס על אמצעי

. 𝑶𝒑𝒕𝒊𝒄𝒂𝒍 𝑭𝒍𝒐𝒘 -ו 𝑳𝒊𝑫𝑨𝑹 חישה שוני ם כגון- חיישני

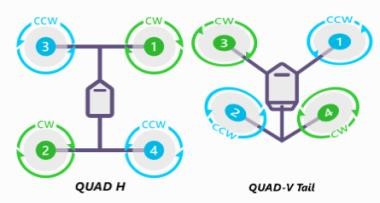
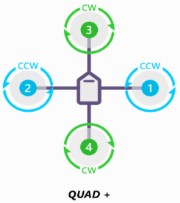
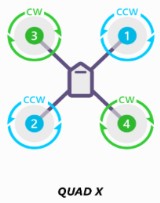
דו"ח זה יציג את הרקע התאורטי הנדרש לפרויקט ויכלול מושגים בסיסיים הנדרשים לפרויקט- רחפנים והדינמיקה שלהם, חיישנים, סוגי תקשורת ועוד. בנוסף, יוצגו תהליכי הפיתוח, האתגרים, דרישות וניהול סיכונים של פיתוח המערכת.

## 2. רקע

פרק זה יעסוק ברקע התאורטי ובסקר ספרות המקנה את הידע הבסיסי הנדרש בפרויקט זה.

### 2.1. כלי טיס מסוג רב-להב

כלי טיס מסוג רב-להב הינו כלי טיס המייצר כוח עילוי על ידי סיבוב מדחף בעל להבים. כלי טיס מסוג זה הנפוץ ביותר הינו בעל ארבעה מדחפים הממוקמים בכל אחד מקצוות מסגרת כלי הטיס. מסגרת כלי הטיס יכולה להיות בנויה במספר תצורות שונות כפי שניתן לראות באיור 1. כמו כן, כל זוג מנועים נגדיים מסתובב בכיוון הפוך, זאת על מנת לבטל את המומנטים הפועלים על הגוף.



**איור 1: קונפיגורציות גוף שונות וכיוון סיבוב המנועים**

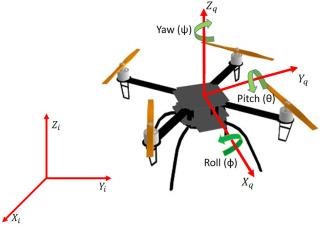
כאש ר כלל המנועים מסתובבים באותה מהירות, כל מנוע מייצר את אותו כוח דחף . כאשר כוח הדחף הכולל של המנועים גדול מכוח הכבידה הפועל על כלי הטיס, תתקבל הרמה של הכלי. על מנת ליצור תנועה כיוונית של הכלי נדרש ליצור כוחות לא שווים בין המנועים באמצעות מהירויות סיבוב שונות. לדוגמא, עבור תצורת גוף של 𝑄𝑢𝑎𝑑 𝐻, כאשר מנועים 1 ו-4 נעי ם מהר יותר ממנועים 2 ו-3, מתקבלת זווית גלגול וכתוצאה מכך כוח הדחף אשר מפעילים המנועים פועל בשני צירים שונים ומתקבלת תנועה צידית. כתוצאה מפירוק כוח הדחף למספר צירים, הכוח שפועל בכל צי ר הינו קטן יותר מהכוח השקול ולכן יש צורך במהירות סיבוב גבוהה יותר של המנועים על מנת לייצור כוח דחף גדול מספיק המאפשר שהיה בגובה קבוע. היתרונות הבולטים של הרב-להב הינ ם יכולת לנשיאת משקל רב, יכולת טיסה במבנים סגורים או באזורים צפופים, זאת עקב האפשרות לרחפנים בגדלים קטנים מאוד, ובעל פשטות מכנית. כלי טיס זה משמ ש במספר רב של תחומים- ביטחוני, מרוצים, צילום, שליחויות ועוד. מערכת דינמית זו הינה מערכת תת ממונעת, כלומר- מערכת בעלת מספר אמצעי הנעה נמוך יותר ממספר דרגות החופש של המערכ ת ולכן הינה מערכת מורכבת להפעלה. הפרק הבא יציג את אחת מהאפשרויות הפעלה ובקרה עבור מערכת זו.

### 2.2. מודל קינמט י

המודל הקינמטי מתאר את ההמרה של מהירויות כלי הטיס ממערכת הגוף למערכת העולם, כפי שהוצג על ידי אלון דוידי, 2011. הפיתוח מתבסס על מספר הנחו ת :

* מרכז המסה של הכלי הינו במרכז השטח.
* הכלי הינו גוף קשיח.
* המנועים הינם קשיחים ומהווים חלק מהכלי.

המודל מתבסס על שתי מערכות צירים עיקריות. האחת הינה מערכת צמודת גוף והשניה מערכת העולם, כפי שמוצ ג באיור 2 .



**איור**

**2**

**:**

**הצירים**

**המוגדרות**

**מערכות**

**עבור**

**מודל**

**זה**

**.**

**עם**

**הגוף**

**מערכת**

**אינדקס**

𝒒

**עם**

**אינדקס**

**ומערכת**

**העולם**

𝒊

לכלי ישנם שש דרגות חופש אשר כוללות את מצבו הזוויתי של הכלי במרחב ואת מיקום הכלי במרחב. על מנת להמיר את המהירות הזוויתית של הכלי ממערכת צמודת גוף, למערכת העולם נדרש להשתמש במטריצת סיבוב המביעה את סיבוב הכלי סביב כל ציריו.

c(𝜓) c(𝜃) c(𝜓)s(𝜃) s(𝜙) − s(𝜓) c(𝜙) c(𝜓)s(𝜃) c(𝜙) + s(𝜓)s(𝜙)

𝑅 = [s(𝜓)c(𝜃) s(𝜓)s(𝜃) s(𝜙) + s(𝜓) c(𝜙) s(𝜓) s(𝜃) c(𝜙) − c(𝜓)s(𝜙)] )2.1(

−𝑠(𝜃) 𝑠(𝜙)𝑐(𝜃) 𝑐(𝜙)𝑐(𝜃)

cos ≜𝑐 ; 𝑠𝑖𝑛 ≜ 𝑠 כאשר

הזוויות מתארות סיבוב באופן הבא: זווית סבסוב 𝜓 (yaw), זווית גלגול 𝜙 (roll) וזווית עלרוד 𝜃 (pitch)

את המהירות הקווית של הגוף במערכת האינרציאלית ניתן לקבל על ידי הכפלת וקטור המהירויות הקוויו ת לפי מערכת הגוף במטריצת הסיבו ב :

𝑋̇𝑖 c(𝜓)c(𝜃) c(𝜓)s(𝜃) s(𝜙) − s(𝜓)c(𝜙) c(𝜓) s(𝜃) c(𝜙) + s(𝜓)s(𝜙) 𝑋̇𝑞

[𝑌𝑖̇ ] = [s(𝜓)c(𝜃) s(𝜓) s(𝜃) s(𝜙) + s(𝜓)c(𝜙) s(𝜓)s(𝜃) c(𝜙) − c(𝜓)s(𝜙)] [𝑌𝑞̇ ] )2.3(

𝑍̇𝑖 −𝑠(𝜃) 𝑠(𝜙)𝑐(𝜃) 𝑐(𝜙)𝑐(𝜃) 𝑍̇𝑞

את המצב הזוויתי של הכלי במערכת אינרציאלית נציג על ידי זוויות אוילר כאשר במערכת הגוף המהירות הזוויתית של הכלי מובעת על ידי (𝑝, 𝑞, 𝑟). את הקשר בין המהירות הזוויתית של הכלי לבין ווקטור הנגזרות של זוויות אוילר ניתן להציג באופן הבא:

|  |  |  |  |
| --- | --- | --- | --- |
| 𝑟 1  [𝑝] = [0  𝑞 0 | 0  c(𝜙)  −𝑠(𝜙) | −s(𝜙) 𝜙̇  c(𝜃) s(𝜙)] [𝜃̇ ]  𝑐(𝜙)𝑐(𝜃) 𝜓̇ | )2.4( |

### 2.3. המודל הדינמ י

המודל הדינמי מבוסס על הצגה בה כוח הדח ף והמומנ ט של כל מנוע הינ ם פרופורציונל יים למהירות הסיבוב של המנוע .

|  |  |
| --- | --- |
| 𝐹 = 𝑏𝜔̅2 | )2.5( |
| 𝜏 = 𝑘𝜔̅2 | )2.6( |

כאשר 𝑏 ו-𝑘 הינם מקדמים המשתנים כתלות במספר גורמים, אך בעבודה זו ההנחה היא כי הינם קבועים וזהים בין המנועים. המשתנה ̅𝜔 מתאר את המהירות הזוויתית של המנוע. כעת, ניתן להגיד כי כוח הדחף השקול מכלל המנועים הינו :

𝐹𝑡𝑜𝑡 = 𝐹1 + 𝐹2 + 𝐹3 + 𝐹4 ).27(

את המומנטים הנוצרים סביב כל ציר כתוצאה מהמנועים ניתן לכתוב בצורה הבאה:

𝜏𝑥 = 𝑙(𝐹2 − 𝐹4)

𝜏𝑦 = 𝑙(𝐹3 − 𝐹1) )2.8(

𝜏𝑧 = 𝜏1 − 𝜏2 + 𝜏3 − 𝜏4

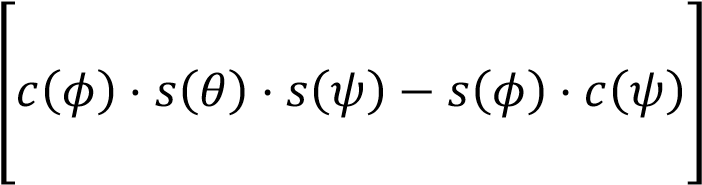
כאשר 𝑙 הינו המרחק של המנועים ממרכז הכותב של הכלי. את הכוחות והמומנטים ניתן להציג במטריצה בצורה הבאה:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 𝐹𝑡𝑜𝑡 𝑏  𝜏  [ 𝜏𝜙𝜃 ] = [−0𝑙𝑏  𝜏𝜓 𝑘 | 𝑏  𝑙𝑏  0  −𝑘 | 𝑏  0  𝑙𝑏  𝑘 | 𝑏 𝜔122  −𝑙𝑏] 𝜔𝜔232  0    −𝑘 [𝜔42] | ).29( |

את הקש ר בין הכוחות למשתנים המתארים את תנוע ת הרחפ ן ניתן לפתח בשיטת ניוטון-אוילר ובשיטת אוילר-לגראנז', כאן יוצגו המשוואות המתקבלות על ידי שימוש בשיטת ניוטון-אוילר .

המשוואות המתארות את התנועה הקווית של הרחפן במערכת אינרציאלית הינם :

𝑋̈ 0 1 𝑐

[𝑌̈ ] = [0] − 𝐹𝑡𝑜𝑡  ).210(

𝑍̈ 𝑔 𝑚 𝑐

משוואות התנועה הזוויתית המתקבלות תחת הנחת זוויות קטנות הן:

𝜙̈ 𝐼𝑥−1[(𝐼𝑦 − 𝐼𝑧)𝜃̇𝜓̇ + 𝜏𝜙]

[𝜃̈ ] = [𝐼𝑦−1[(𝐼𝑧 − 𝐼𝑥)𝜙̇𝜓̇ + 𝜏𝜃]] ).211(

𝜓̈ 𝐼𝑧−1[(𝐼𝑥 − 𝐼𝑦)𝜃̇𝜙̇ + 𝜏𝜓]

ניתן לשים לב כי מודל זה אינו לינארי ולכן תכנון מערכות בקרה הינו מורכב .

### 2.4. חיישני ם

על מנת לאפשר שליטה מדויקת בכלי הטיס והוספת מערכות עזר למטיס, נדרשים מספר חיישנים המאפשרים מדידות של פרמטרים המייצגים את כלי הטיס ומאפשרים לקבל מידע על מיקומו במרחב ביחס לאובייקטים אחרים. יש לציין כי כלל החיישנים אשר בשימוש בפרויקט זה הינם חיישנים אשר מבצעים מדידה בתדירות מסוימת ואינם מודדים באופן רציף ולכן אין מידע רציף מכלל החיישנים ונדרש להמתין לזמן הדגימה הבא על מנת לקבל מידע עדכני.

**IMU .2.4.1**

IMU הינו ראשי תיבות של Inertial measuring unit, אשר מוצג סכמתי ת באיור 3, חיישן זה הינ ו נפוץ בעולם הרובוטיקה , מכיוון שבעזרתו ניתן לקבל מדידות של המצב הזוויתי של הגוף. החייש ן מורכב ממספר חיישנים- 3 חיישני ג'יירו אשר מודדים את המהירות הזוויתית של הרחפן, ו-3 מדי תאוצה אשר מודדים את הכוחות הפועלים על הרחפן. בעזרת חיבור הקריאות משני סוגי חיישנים אלו ניתן לקבל מדידה יחסית מדויקת של המצב הזוויתי של הרחפן באוויר. קיימים ג ם חיישני IMU אשר בנוסף לג'יירו ולמדי תאוצה, הם מורכבים גם ממגנטומט ר אשר מודד את עוצמת השדה המגנטי מסביב לכלי הטיס ,מידע המאפשר דיוק נוסף למדידות של המצב הזוויתי של כלי הטיס במקרים מסוימים. )Vectornav(



**איור**

**3**

**:**

**מחיישן**

**השונות**

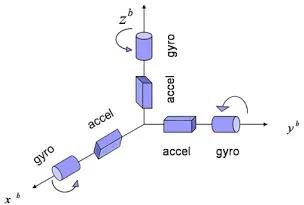
**והמדידות**

**החיישנים**

**של**

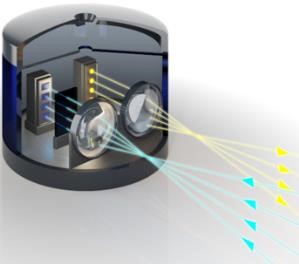
**תיאור**

**IMU**



#### LiDAR .2.4.2

Light Detection and Ranging( LiDAR( הינו חיישן הפולט קרני לייז ר כיווניים . לאחר פליטתם, מודד החיישן את הזמן הלוקח לקרני הלייזר לחזור וכך הוא מודד מרחק מאובייקטים. המרחק אותו יכול החיישן לאמוד תלוי בעוצמת קרן הלייזר אותה הוא פולט ובזמן ההמתנה של החיישן לקרן לחזור. יש מספר תצורות לחיישן מסוג זה, כאשר הנפוצה ביותר הינה חיישן הפולט ק רן לייזר בכיוון אחד ומאפשרת חישה של המרחב בכיוון רצוי. תצורה נוספת הינה חיישן מסתובב אשר מאפשרת לקבל מדידות °360 מהגוף. )MATLAB(



**איור**

**4**

**:**

**חיישן**

**של**

**פעולה**

**אופן**

**LiDAR**

**בצהוב**

**.**

**-**

**בכחול**

**,**

**גלים**

**פליטת**

**-**

**לחיישן**

**חזרתם**

**2.4.3. Optical Flow** חייש ן זה מאפשר זיהוי של תזוזת הרחפן בעזרת חיישן אופטי המזהה את שינוי המיקום של חלקיקים במשטח מעליו טס הרחפן, ביחס למיקומו. בעזרת מדידה זו, ומכיוון שהמדידה הינה יחסית לרחפן ,ניתן לחשב את מהירות הרחפן בשני צירים- 𝑥, 𝑦. חיישן כזה, בתוספת חיישן LiDAR, הפונה כלפי מטה, מאפשר קבלת מידע מלא על גובה הרחפן ועל מהירות התזוזה הקווית שלו, ובכך ניתן לאפשר שהייה אוטונומית של הרחפן במיקום קבוע .



**איור**

**5**

**:**

**חיישן**

**Optical Flow**

**חיבור**

**בעל**

**בסיסי**

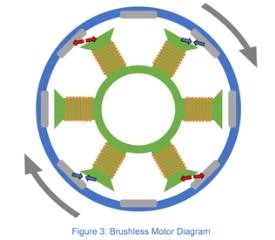
**UART**

בפרויקט זה, חיישן מסוג זה נדרש על מנת לאפשר מדידה של התנועה הקווית של הרחפן. מכיוון שהמדידה הינה של מהירות, כך גם הפקודות שינתנו לרחפן בהמשך הן פקודות של מהירות. חיישן זה מהווה אמצעי מרכזי בפיתוח הפרויקט.

### 2.5. מנועים ללא מברשו ת

ישנם שני סוגים עיקריים של מנועים חשמליים- מנוע מברשות ומנוע ללא מברשות. בכלי טיס מסוג רב-להב משתמשים במנועי ם ללא מברשות מכיוון שהם מאפשרים ביצועים טובים, בעלי פינוי חום יעיל יות ר ומאפשרים תחזוקה קלה יותר כתוצאה מכמות רכיבים נמוכה יותר. במנועים אלה קיימים מגנטים וסלילים. המגנטים בעלי קוטביות קבועה והם נמצאים בחלקו המסתובב של המנוע, הסלילים נמצאים בחלקו הנייח של המנוע ומחולקים לשלוש קבוצו ת בפאזה של °120 אחד מהשני. הפרש פאזה זה מאפשר רציפות בתנועת המנוע. במנועים אלה ייתכנו יותר משלושה צמדים של סלילים. כמו כן, מספר המגנטים ברוטור אינו זהה למספר הסלילים הנמצאים בסטטור. מכך שקיימות קבוצות הסלילים, נדרש בקר שליטה נוסף אשר יודע לתזמן את העברת הזרם בסלילים גם מבחינת תזמון מעבר הזרם גם מבחינת גודלו. בקר זה נקרא −𝐸𝑆𝐶 𝐸𝑙𝑒𝑐𝑡𝑟𝑜𝑛𝑖𝑐 𝑠𝑝𝑒𝑒𝑑 𝑐𝑜𝑛𝑡𝑟𝑜𝑙𝑙𝑒𝑟, והו א מקבל פקודות מהירות וקובע את קצב העברת הזרם בין הסלילים השונים במנוע. מהירות המנוע ברחפנים הינה פרופורציונלית למתח אשר נופל על המנוע. פרמטר זה נקרא

𝑘𝑉, ועבור מתח מסוים, מנוע בעל 𝑘𝑉 גבוה יותר יסתובב מהר יותר .



**איור**

**6**

**:**

**באדום**

**.**

**מברשות**

**ללא**

**מנוע**

**של**

**סכימה**

**-**

**פאזה**

**1**

**בכחול**

**,**

**-**

**פאזה**

**2**

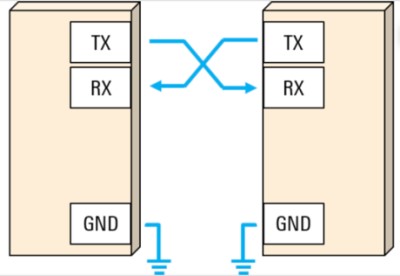
### 2.6. חיבורי תקשור ת

ישנם מספר פרוטוקולי תקשורת המאפשרים חיבור של מספר רכיבים על מנת שיוכלו להעביר מידע אחד עם השני. פרק זה יסקור את שני הפרוטוקולים הרלוונטיים לפרויקט זה .

**UART .2.6.1**

universal asynchronous receiver / transmitter או בקיצור UART, הינו פרוטוקול תקשורת המאפשר העברת מידע דו כיווני בין שני רכיבים. פרוטוקול תקשורת זה מורכב משני חיבורים Tx- Transmitter, Rx- Receiver. החיבור הראשון מיועד להעברת מידע החוצה מהרכיב והחיבור השני מיועד לקבלת מידע לרכיב. התקשורת יכולה להתבצע בשלוש קונפיגורציות שונות- מידע נשלח בכיוון אחד בלבד, מידע מועבר בשני הכיוונים אך בכל זמן נתון רק רכיב אחד מעביר מידע, מידע מועבר בשני הכיוונים ובאותו הזמן.

) Rohde & Schwarz USA, Inc.(



**איור**

**7**

**:**

**מסוג**

**חיבור**

**ידי**

**על**

**רכיבים**

**חיבור**

**של**

**סכמטי**

**איור**

**UART**

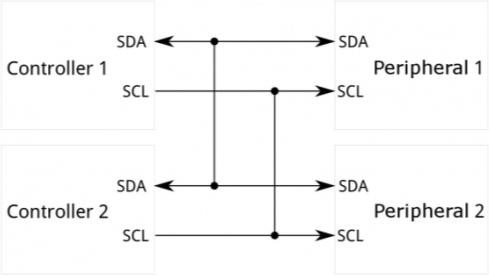
יש לשים לב כי בחיבור זה נדרש לחבר 𝑅𝑥 ↔ 𝑇𝑥 בין שני הרכיבים. את חיבור הפלט של רכיב אחד יש לחבר לחיבור הקלט של רכיב אחר .

**I2C .2.6.2**

SCL- Serial -ו SDA- Serial Data הינו פרוטוקול המבוסס על שני חיבורים Inter-Integrated Circuit

Clock. חיבור זה מאפשר תקשורת בין מספר רכיבים שונים עם בקר אחד. הרכיבים מחוברים לאותם חיבורי

SDA ו- SCL בבקר. כל רכיב הינו בעל כתובת, ובעזר תה הבקר יודע לפנות אל החיישן הנדרש. )Sparkfun (



**איור**

**8**

**:**

**מס**

**חיבור**

**של**

**סכמטי**

**איור**

**אחד**

**לבקר**

**רכיבים**

**פר**

במידה ומבוצע חיבור של מספר חיישנים זהים כלל החיישנים יהיו בעלי כתובת זהה, מצב זה אינו רצוי ולכןנדרש לשנות כתובת לחיישנים. שינוי הכתובת מתאפשר על ידי שימוש בחיבור חשמלי נוסף של חיבור בשם 𝑋𝑆𝐻𝑈𝑇, חיבור זה מאפשר כיבוי והדלקה של חיישן ובכך ניתן לבודד את החיישנים המחוברים ולשנות להם את הכתובת.

### 𝑳𝒊 − 𝒊𝒐𝒏 וסוללת 𝑳𝒊 − 𝑷𝒐 סוללת .2.7

ישנם שני סוגים עיקריים של סוללות בהם נעשה שימוש עבור רחפנים .הסוג הראשון הינו סוללת 𝐿𝑖 − 𝑃𝑜, והי א בעלת יכולת פריקה (𝐶 𝑟𝑎𝑡𝑒), גבוה. מה שמאפשר תגובה מהירה יות ר לדרישות המפעיל. סוללות מסוג זה משמשות בעיקר בתחום המרוצים ובהטסה מסוג 𝐹𝑟𝑒𝑒𝑠𝑡𝑦𝑙𝑒. הסוג השני הינו 𝐿𝑖 − 𝑖𝑜𝑛, סוללות מסוג זה בעלות קצב פריקה נמוך יותר משמעותית בהשוואה לסוג הראשון . כתוצאה מכך הרחפן יגיב לאט יותר לדרישות המפעיל. בנוסף, כתוצאה מקצב הפריקה הנמוך, סוללות אלה מאפשרות זמן תעופה גבוה יותר, זהו היתרון של סוללות אלה בהשוואה לסוללות 𝐿𝑖 − 𝑃𝑜. כמו כן, ניתן לפרוק סוללת 𝐿𝑖 − 𝑖𝑜𝑛 עד למתח נמוך יותר, ללא סכנה לפגיעה בסוללה, מאשר סוללת 𝑉 -𝐿𝑖 − 𝑃𝑜8.2 לעומת Robocraze( .3.5𝑉(

### 2.8. תוכנות שליטה ברב-להב

כיום קיימות מספר רב של תוכנות אותן ניתן לצרוב על בקר הטיסה (𝐹𝐶) על מנת לאפשר שליטה ברחפן.

התוכנות הנפוצות ביותר הן- 𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡, 𝑃𝑋4, 𝐵𝑒𝑡𝑎𝐹𝑙𝑖𝑔ℎ𝑡, 𝑖𝑁𝑎𝑣, כאשר כל תוכנה מתמקדת באופן טיסה אחר ובעלת חוזקות שונות .תוכנות אלה מכילות את הפילטרים, אמצעי השיערוך ,קבלת מידע מחיישנים ומהשלט ומתן פקודות למנועים . התוכנות 𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡 ו- 4𝑃𝑋 תומכות בחיבור של מחשב נוסף )𝐶𝑜𝑚𝑝𝑎𝑛𝑖𝑜𝑛 𝑃𝐶( ל-𝐹𝐶 ובעזרתו ניתן לתכנת באופן עצמאי רובד שליטה נוסף ברחפן בהתאם לדרישות המתכנן. ישנם מספר בקרים בהם ניתן להשתמש כ- 𝐶𝑜𝑚𝑝𝑎𝑛𝑖𝑜𝑛 𝑃𝐶 , כאש ר הנפוץ ביותר הינו 𝑅𝑎𝑠𝑝𝑏𝑒𝑟𝑦𝑃𝑖. על מיקרו בקר זה ניתן לתכנת אלגוריתמים בשפת תכנות 𝑃𝑦𝑡ℎ𝑜𝑛 המאפשרים שליחת פקודות ל-𝐹𝐶. קיימות מספר ספריות המקלות על תהליך שליחת הפקודות, בניהן ,𝐷𝑟𝑜𝑛𝑒𝐾𝑖𝑡, 𝑀𝐴𝑉𝑆𝐷𝐾

.𝑃𝑦𝑚𝑎𝑣𝑙𝑖𝑛𝑘, 𝑚𝑎𝑣𝑅𝑂𝑆

### DroneKit ספריית .2.9

כלל תוכנות השליטה ברחפנים מתבססות על פרוטוקול תקשורת בשם 𝑀𝐴𝑉𝑙𝑖𝑛𝑘. פרוטוקול זה מעביר את הנתונים הנדרשים בין המפעיל לבין הרחפן. בין הנתונים קיים- זמן תעופה, מצב סוללה, מצב זוויתי ועוד. הספרייה 𝐷𝑟𝑜𝑛𝑒𝐾𝑖𝑡 הינה ספרייה היודעת לקרוא את אופן הדיווחים, לנתחם ולהמיר לנתונים פשוטים הניתנים לשימוש. בנוסף, ספריה זו מאפשרת מתן פקודות לרחפן ובכך לאפשר שליטה בעזרת מחשב נוסף השולח פקודות. על מנת לאפשר את מתן הפקודות נדרש שהרחפן יהיה במצב טיסה "𝐺𝑈𝐼𝐷𝐸𝐷" מצב זה שומר על מיקום וגובה קבועים, ובמצב זה פקודות הטיסה מוגדרות להתקבל ממחשב אחר אשר מתכנן את מסלול הטיסה. ספריה זו אינה מקבלת עדכונים שוטפים יותר ולכן ייתכן כי פונקציות מסוימות המובנות בספריה לא יעבדו כנדרש. בעתיד יש לבחון מעבר לספריה 𝑃𝑦𝑀𝐴𝑉𝐿𝑖𝑛𝑘, ספריה זו בעלת עדכונים שוטפים יותר ומאפשרת רובד שליטה נמוך יותר מאשר 𝐷𝑟𝑜𝑛𝑒𝐾𝑖𝑡 ולכן יכולה לאפשר שליטה ביותר פרמטריםובצורה מהירה יותר. החיסרון משימוש בספריה זו הינו, מכיוון שהספרי יה נחשבת ברובד נמוך יותר, נדרש לשים לב לדקויות קטנות יותר .

### 2.10. ניוו ט

התחום של ניווט רובוטים הינו תחום רחב, כיום ישנם מספר אלגוריתמים מובילים אשר מיועדים לניווט ולוקליזציה של רובוטים, זאת על מנת לאפשר את התמצאותם בשטחים שאינם מוכרים. מרבית האלגוריתמים דורשים מספר רב של דגימות, ודגימות מכלל צידי הרובוט.

𝑺𝑳𝑨𝑴 − 𝑺𝒊𝒎𝒖𝒍𝒕𝒂𝒏𝒆𝒐𝒖𝒔 𝑳𝒐𝒄𝒂𝒍𝒊𝒛𝒂𝒕𝒊𝒐𝒏 𝒂𝒏𝒅 𝑴𝒂𝒑𝒑𝒊𝒏𝒈 **.2.10.1**

אלגוריתם זה נפוץ מכיוון שהוא מאפשר מיפוי של המרחב ולוקליזציה של הרובוט בחלל בו הוא נמצא. מנגד, אלגוריתם זה הינו בעל דרישות חישוביות גבוהות ודורש אופטימיזציה על מנת להריץ על מיקרו מחשב קטן כמו 𝑹𝑷𝑰 − 𝒁𝒆𝒓𝒐.

𝑽𝑭𝑯 − 𝑽𝒆𝒄𝒕𝒐𝒓 𝑭𝒊𝒆𝒍𝒅 𝑯𝒊𝒔𝒕𝒐𝒈𝒓𝒂𝒎 **.2.10.2**

אלגוריתם זה משתמש במדידות של חיישני המרחק על מנת לחשב היסטוגרמה פולרית המייצגת את מרחק המכשולים מהרובוט. גם כאן, בדומה ל- 𝑆𝐿𝐴𝑀, נדרשות מדידות מרחק רבות אשר יאפשרו לייצר היסטוגרמה בעלת מידע רב מספיק לטובת שליחת פקודה אופטימלית לתנועה הרובוט.

## 3. דרישות ומטרות

### 3.1. מטרות הפרויק ט

פרויקט זה הינו מרכיב מרכזי כחלק מתוכנית פיתוח של רחפן אוטונומי אשר יתחרה ברחפנים הקיימים כיום בשוק, כגון 𝐷𝐽𝐼 𝐴𝑣𝑎𝑡𝑎. פרויקט מסוג זה מכיל מספר תחומי פיתוח שונים, כגון- פיתוח מערכת עזר אשר תאפשר הימנעות אקטיבית מהתקרבות יתרה לדפנות מסלול הטיסה ופיתוח מערכת ניווט אוטונומית. פרויקט זה יתמקד בפיתוח מערכת הניווט האוטונומית לשלבי ההטסה כאשר נותק הקשר עם מפעיל הרחפן תוך כדי שמירת מסלול התנועה וחזרה אחורה לנקודת התקשורת האחרונה. על ידי עמידה במספר מטרות:

1. תכנון, בניה והרכבה של רחפן מבוסס ArduPilot.
2. תכנון והרכבה של רכיבים תומכים לטובת אינטגרציה בין הרכיבים הנוספים הנדרשים לרחפן כגון

.𝑅𝑃𝐼 − 𝑍𝑒𝑟𝑜 -ומחשב נוס ף 𝐿𝑖𝐷𝐴𝑅 חיישני

1. כתיבה והטמעת אלגוריתמים המאפשרים יכולות אוטונומיות של הרחפן בסביבות סגורות.

### 3.2. דרישות הפרויק ט

הרחפן בפרויקט זה נדרש לעמוד במספר דרישות מרכזיות .פרק זה יציג את מפרט הדרישות ההנדסיות של המערכת.

**טבלה 1: רשימת דרישות הפרויקט**

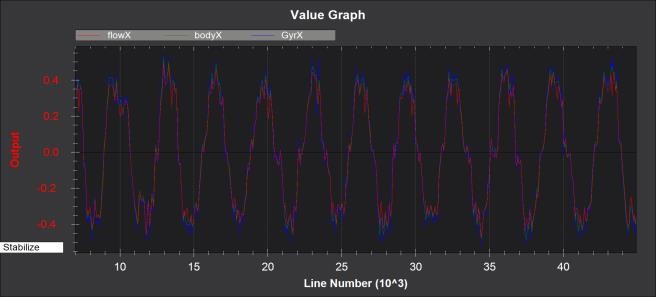
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| דרכי בדיקה  והוכחות | נימוק דרישה | רצוי/  הכרחי | ערך | דרישה מילולית | # |
| שימוש בשלדה ובפרופלורים  מתאימים של 5.3 אינץ' | ממדי המנהרה והשטחים הסגורים בהן יידרש לעבור הרחפן | הכרחי | קוטר פרופלור של 5.3 אינץ' | ממדי הרחפן | 1 |
| מדידת זמן טיסה  מטעינה מלאה של הסוללה ועד מתח מינימלי אפשרי | זמן תעופה מינימלי על מנת לבצע משימות  פשוטות הינו 5 דק', זמן  תעופה גבוה יותר יאפשר ביצוע משימות מורכבות יותר | רצוי  הכרחי | מקסימלי- 25 דק'  מינימלי- 5 דק | זמן תעופה | 2 |
| המספר המינימלי הנדרש הינו 4  חיישנים- צדדים, קדימה ולמטה | אמצעים אשר יאפשרו לרחפן לזהות את  הסביבה בכלל הכיוונים הנדרשים לצרכי הניווט | הכרחי הכרחי הכרחי רצוי רצוי | צדדי ם קדימה למטה  למעלה אחורה | אמצעי חישה | 3 |
| הטסת הרחפן צמוד לקיר וזיהוי בעין  האם הרחפן מבצע תיקונים באופן עצמאי על מנת להתרחק מהקירו ת | פיתוח מערכת אשר תעקוף את פקודות  המטיס במידה והפקודה  תוביל לכך שהרחפן יטוס במרחק הקטן מ -15 ס"מ מדפנות שטח הטיס ה | רצוי | שמירת מרחק  מינימלי של 15 ס"מ | מערכת עזר להימנעות מקירו ת | 4 |
| ביצוע ניסויי  המראה וכיול  חיישן ה 𝑂𝐹 על מנת לקבל זיהוי  מדויק ככל הניתן של התנועה הקווית של הרחפן | הרחפן נדרש בהינתן  פקודה מהשלט, להמריא ולשמור על גובה מסוים.  בנוסף, במהלך טיסה, במידה ולא התקבלה פקודה, הרחפן נדרש לשמור על מיקום קבוע | הכרחי | המראה באופן  עצמאי, ושמירה על מיקום קבו ע | יכולות אוטונומיות בסיסיות | 5 |
| בניית מסדרון עץ המדמה מנהרה והטסת הרחפן  במסדרון זה, זיהוי האם הרחפן מבצע את הבחירות בצורה עצמאי ת | הרחפן נדרש, באמצעות חיישני LiDAR, לבצע בחירה של נתיב טיסה רצוי על מנת להתקדם בשטח הסגור בצורה עצמאי ת | הכרחי | יכולת בחירת נתיב טיסה בצורה עצמאי ת | ניווט | 6 |
| לבצע טיסת ניסיון במסדרון ובדיקה האם הרחפן חוזר | נקודת החזור של הרחפן לנקודת התקשורת האחרונה הינה בהתאם | הכרחי | אינדיקציה  תמידית של מתח הסוללה | זיהוי מתח סוללה | 7 |
| במתח סוללה מספיק | למתח הסוללה ועל כן נדרש לקבל אינדיקציות על מתח הסוללה ולוודא האם המתח הקיים הינו מספיק לטובת טיסה חזרה לנקודת התקשור ת |  |  |  |  |

## 4. שלבי הפיתוח

הפיתוח מחולק לשני חלקים עיקריים- האחד, הרכבת הרחפן. חלק זה כולל הרכבה של החלקים הבסיסיים של הרחפן, צריבת התוכנה וחיבור החיישנים הנדרשים. השני, תכנון וביצוע של מערכת העזר למטיס ומערכת ניווט בסיסית. חלק זה יכלול תכנון מערכת בקרה לשמירה על מסלול רצוי ואלגוריתם המאפשר טיסה אוטונומית של הרחפן הכולל חזרה לנקודת תחילת הטיסה .בשני חלקי הפיתוח בוצעו ניסויים לבדיקת תקינות המערכות לפני התקדמות לשלב הבא.

### 4.1. הרכבת הרחפן

עם תחילת הפרויקט נבנה רחפן מבוסס 𝐹𝐶 מסוג 𝑆𝑝𝑒𝑒𝑑𝑦𝐵𝑒𝑒 𝐹405 𝑚𝑖𝑛𝑖 אך לאחר מספר הטסות וזיהוי כי התקבלו מדידות לא אמינות מהרכיבים על הבקר, כמו גובה לא נכון או שגיאות במדידות הג'ייר ו שהובילו לקביעת מצב זוויתי לא נכ ון, הוחלט לבנות רחפן נוסף המבוסס על בקר טיסה 4𝑆𝑝𝑒𝑒𝑑𝑦𝐵𝑒𝑒 𝐹405 𝑉, עבור בקר טיסה זה התקבלו מדידות אמינות יותר .מכיוון שתוכנת ההטסה היחידה, המאפשרת תכנות לאוטונומיה ותומכת בבקר טיסה זה, הינה 𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡 פרויקט זה יתבסס על תוכנה זו. כמו כן, חובר חיישן 𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤 מסוג 𝑀𝑎𝑡𝑒𝑘𝑠𝑦𝑠 3901 − 𝐿0𝑋 אשר כולל גם חיישן 𝐿𝑖𝐷𝐴𝑅 עד טווח של 2 מטר ובוצעו בדיקות לתקינות הטיסה. התקבלו תוצאות לא טובות אשר הובילו לרחפן לא יציב. על פי התיעוד של התוכנה 𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡, לאחר חיבור החיישן נדרש להשוות את ערכי המדידה של החיישן לערכי המדידה של הג'יירו ושיתקבלו קריאות דומות.



𝑮𝒚𝒓𝒐 **-ל** 𝑶𝒑𝒕𝒊𝒄𝒂𝒍 𝑭𝒍𝒐𝒘 **איור 9 : תוצאה רצויה מהשוואת**

בגרף המוצג מעלה ניתן לראות את התוצאות הרצויות מחיישן ה𝑂𝐹 על מנת שתתאפשר שהייה במקום בצורה מדויקת. הגרף מציג השוואה בין הערכים אשר נמדדו מהג'יירו לבין ערכי המהירות הזווית אשר נמדדו על ידי ה- 𝑂𝐹. בעזרת חיישן זה לא התקבלה התוצאה הרצויה של המדידות ולאחר מספר ניסיונו ת כיול התקבלה החלטה להחליף לחיישן 01 −𝑀𝑖𝑐𝑜𝑎𝑖𝑟 𝑀𝑇𝐹 , חיישן זה בעל חיישן זהה של

.עם טווח מקסימלי של 8 מטר 𝐿𝑖𝐷𝐴𝑅 אך בעל חיישן 𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤



**איור**

**10**

**:**

**חיישן**

𝑶𝒑𝒕𝒊𝒄𝒂𝒍

𝑭𝒍𝒐𝒘

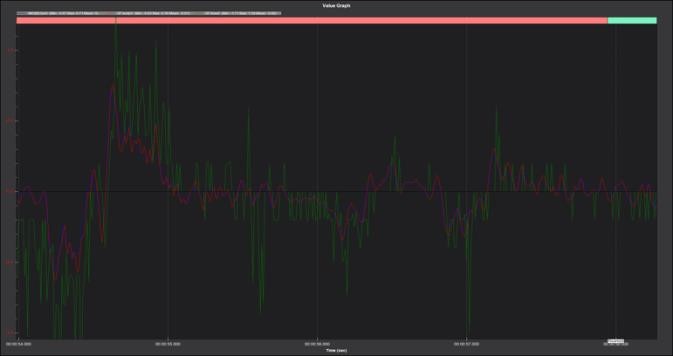
𝑴𝒊𝒄𝒐𝒂𝒊𝒓

𝑴𝑻𝑭

−

𝟎𝟏

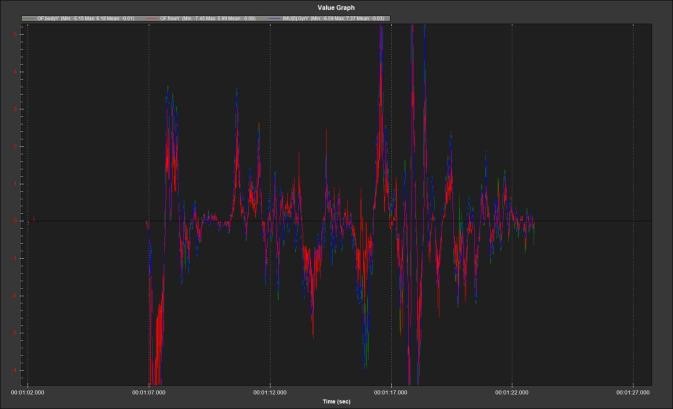
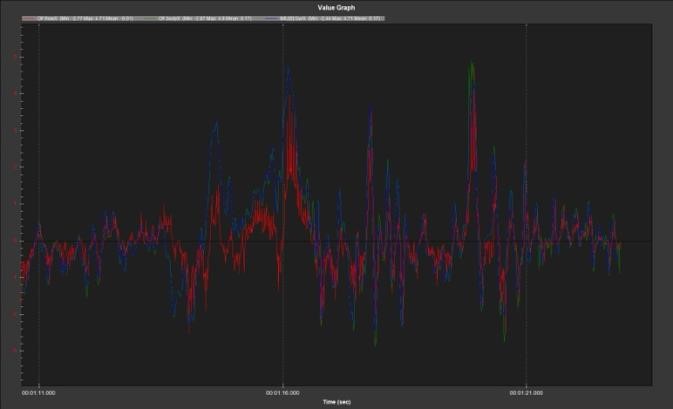
לאחר החלפת החיישן וביצוע ניסויים נוספים התקבלו הגרפים הבאים :



**איור 11 : השוואה בין ערכי ה-**𝑮𝒚𝒓𝒐 **לערכי החיישן** 𝑶𝒑𝒕𝒊𝒄𝒂𝒍 𝑭𝒍𝒐𝒘**. מימין- ציר X, משמאל - ציר Y**

מהגרף ניתן לראות כי ערכי 𝐹𝑙𝑜𝑤 של החיישן )באדום( גדולים יות ר בהשוואה לערכי הג'ייר ו ולערכי ה 𝐵𝑜𝑑𝑦 אותו מודד החייש ן )כחול וירוק בהתאמה(. כמו כן, ניתן לראות כי ערכי הג'יירו וערכי ה-𝐵𝑜𝑑𝑦 מאוד דומים אחד לשני. לאחר זיהוי הבדל זה , הקיים בשני הצירים ,התבצע שינוי לפרמט רים

𝐹𝐿𝑂𝑊𝐹𝑋𝑆𝐶𝐴𝐿𝐸𝑅,𝐹𝐿𝑂𝑊𝐹𝑌𝑆𝐶𝐴𝐿𝐸𝑅 המאפש רים לכייל את היחס בין 𝑂𝐹𝑓𝑙𝑜𝑤 ו- 𝑂𝐹𝑏𝑜𝑑𝑦. לאחר כיול פרמטרים אלה וקביעת ערכם 450 =𝐹𝐿𝑂𝑊𝐹𝑋𝑆𝐶𝐴𝐿𝐸𝑅  ו- 400 =𝐹𝐿𝑂𝑊𝐹𝑌𝑆𝐶𝐴𝐿𝐸𝑅  התקבלה ההתנהגות הרצויה של הרחפן והיכולת לשמור על מיקו ם אך קיימות אוסילציו ת של הרחפן בציר 𝑌, דבר המשפיע על יציבות הרחפן. לאחר כיולים של הגברי הבקרה בציר זה על מנת לאפשר התנהגות יציבה יותר של הרחפן התקבלו התוצאות הבאות .



**איור 12 : השוואה בין הערכים לאחר כיול. מימין- ציר X, משמאל- ציר Y**

עם סיום כיול חיישן זה בוצעו מספר ניסויי הטסה אשר בדקו את יכולות הרחפן לבדיקת יציבות ושמירה על מיקום כפי שנדרש. לאחר אחד מהניסויים חיישן זה נפגע ולא היה שמיש. המענה היה החלפה לחיישן נוסף של אותה חברה. החיישן החדש בו יש שימוש הינו 02 −𝑀𝑖𝑐𝑜𝑎𝑖𝑟 𝑀𝑇𝐹 , חיישן זה בעל רכיב 𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤 זהה לחיישן הקודם אך ה- 𝐿𝑖𝐷𝐴𝑅 הינו בעל טווח קצר יותר של 2.5 מטר. לאחר מספר בדיקות נמצא כי חיישן מודד ערכים לא מדויקים ובעלי רעש רב, הוחלט להקטין את תחום ההסתמכות על החיישן, מדידת חיישן זה תילקח כמדידה עיקרית לגובה כל עוד גובה הרחפן הוא נמוך מ-2 מטר. מכיוון שהרחפן נדרש לטוס במקומות סגורים, גובה מקסימלי זה אינו מגביל את יכול הרחפן לבצע את משימתו ולעמוד בדרישות הפרויקט. כמו כן, לאחר ביצוע מספר טיסות ניסוי בקונפיגור ציה הסופית של הרחפן בוצעו כיולים נוספים למערכת הבקרה על מנת לקבל תגובות מהירות יותר לפקודות השלט, הקטנת רעידות הרחפן, מניעת התחממות יתר של המנועים )כתוצאה מהגבר נגזרת גבוה.( בנוסף לתהליך חיבור החיישנים לבקר טיסה וכיולם ,חובר מחשב שליטה- 𝑅𝑎𝑠𝑝𝑏𝑒𝑟𝑟𝑦𝑃𝑖 𝑍𝑒𝑟𝑜 2𝑊 המאפשר שליחת פקודות על בסיס קוד 𝑃𝑦𝑡ℎ𝑜𝑛. למחשב זה חוברו שלושה חיישני 𝐿𝑖𝐷𝐴𝑅 מסוג 𝑉𝐿53𝐿1𝑋. חיישנים אלה מאפשרים מדידת מרחק של עד כ-4 מטר, מרחק המאפשר עמידה בדרישות הרחפן לתכנן מסלול טיסה באופן עצמאי.

רשימת רכיבים בהם יש שימוש בפרויק ט לאחר ניסוים ובדיקת רכיבים שונים :

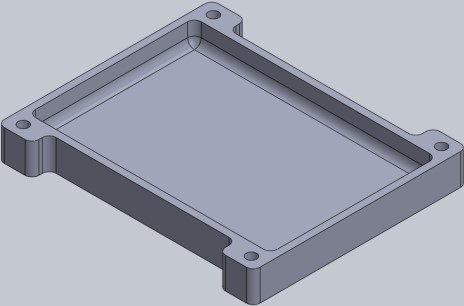
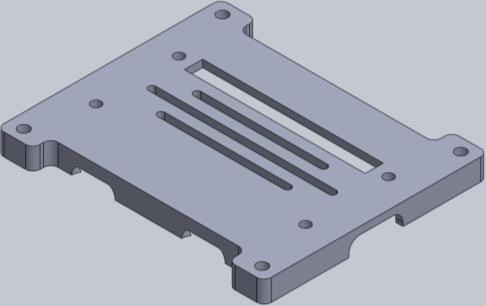
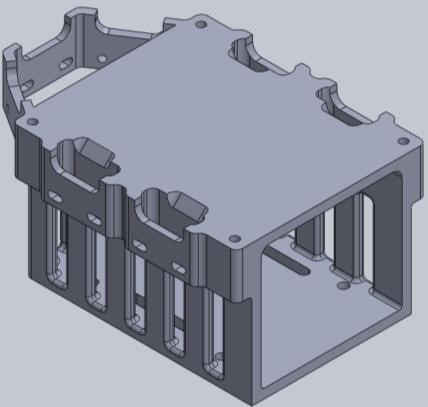
**טבלה 2: רשימת רכיבים שנבחרו לאחר ניסויים**

|  |  |  |  |
| --- | --- | --- | --- |
| חיבור חשמל י | נתונים | שם הרכי ב | סוג רכי ב |
| - | שלדת רחפן המתאימה  לפרופלורים של 3.5 אינץ' | 𝑆𝑝𝑒𝑒𝑑𝑦𝐵𝑒𝑒 𝐵𝑒𝑒35 | 𝐷𝑟𝑜𝑛𝑒 𝐹𝑟𝑎𝑚𝑒 |
| בעל 6 חיבורי 𝑈𝐴𝑅𝑇 | בקר זה מכיל בתוכו  𝐼𝑀𝑈, 𝐵𝑎𝑟𝑜𝑚𝑒𝑡𝑒𝑟 | 𝑆𝑝𝑒𝑒𝑑𝑦𝐵𝑒𝑒 𝐹405 𝑉4  𝑆𝑡𝑎𝑐𝑘 | 𝐹𝑙𝑖𝑔ℎ𝑡 𝐶𝑜𝑛𝑡𝑟𝑜𝑙𝑙𝑒𝑟  +𝐸𝑆𝐶 |
| מחובר ל- 𝐸𝑆𝐶, 3 חיבורים עבור שלושת הפאזות במנו ע | 𝐷 = 20 [𝑚𝑚] מנוע ים בעליℎ = 6 [𝑚𝑚]  𝐾𝑉 = 1950 | 𝑆𝑝𝑒𝑒𝑑𝑦𝐵𝑒𝑒 2006  −1950𝐾𝑉 | 𝑀𝑜𝑡𝑜𝑟𝑠 |
| מחוברים בעזרת 𝐼2𝐶  𝑅𝑃𝐼 𝑍𝑒𝑟𝑜 -ל | טווח מקסימלי של 8 מט ר | 𝑃𝑜𝑙𝑜𝑙𝑢 𝑉𝐿53𝐿1𝑋 | 𝐿𝑖𝐷𝐴𝑅 |
| -ב 𝑈𝐴𝑅𝑇3 -מחובר ל    𝐹𝐶 | הכולל 𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤 חיישן  גם 𝐿𝑖𝐷𝐴𝑅 בעל טווח מקסימלי  של 2.5 מטר | 𝑀𝑖𝑐𝑜𝑎𝑖𝑟 𝑀𝑇𝐹 − 02 | 𝑂𝑝𝑡𝑖𝑐𝑎𝑙 𝐹𝑙𝑜𝑤 |
| 𝑈𝐴𝑅𝑇2 מחובר ל | מחשב קל וקטן בעל יכולות  עיבוד מספיקות לדרישות המערכ ת | 𝑅𝑎𝑠𝑝𝑏𝑒𝑟𝑟𝑦𝑃𝑖 𝑍𝑒𝑟𝑜 2𝑊 | 𝐶𝑜𝑚𝑝𝑎𝑛𝑖𝑜𝑛 𝑃𝐶 |

על בסיס פלטפורמה זו נכתב קוד בסיסי המתבסס בספריה 𝐷𝑟𝑜𝑛𝑒𝐾𝑖𝑡 המאפשר תקשורת בין בקר הטיסה לבין ה-𝐶𝑜𝑚𝑝𝑎𝑛𝑖𝑜𝑛 𝑃𝐶. קוד זה נועד לנהל את בקר הטיסה. הקוד מורכב ממספר פונקציות הכוללות בין היתר פונקציות הנועדו ליצירת תקשורת בין המחשבים, קבלת המצב הזוויתי של הרחפן, שליחת פקודות תנועה ועוד. בנוסף, התבצעו בדיקות של חיבור מספר חיישני 𝐿𝑖𝐷𝐴𝑅 למחשב השליטה, קבלת מידע ומימוש פילטר אשר מסתמך על קריאות קודמות על מנת להעלות את רמת האמינות של המדידות.

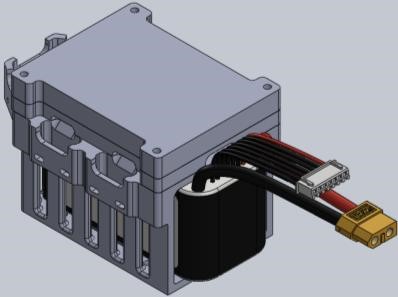
**4.1.1. מגן סוללה**

מכיוון שישנם רכיבים רבים שיש להוסיף לרחפ ן הוחלט לתכנן מארז אשר נועד להגן על הסוללה ,𝑅𝑃𝐼, לאפשר חיבור של חיישני ה-𝐿𝑖𝐷𝐴𝑅 תוך שמירה על פונקציונליות, הימנעות מחוטי חשמל חשופים ותוספת מינימלית למשקל הרחפן . המגן מורכב משלושה חלקים- גוף, המאפשר את אחסון הסוללה וחיבור של חיישני ה-𝐿𝑖𝐷𝐴𝑅, מכסה ביניים, המאפשר חיבור של ה-𝑅𝑃𝐼 וקיבוע חוטי חשמל כך שלא יהיו חשופים, ומכסה אשר סוגר את המגן באופן סופי.



**איור 13 : מגן סוללה. מימין לשמאל- גוף, מכסה ביניים ומכסה .**

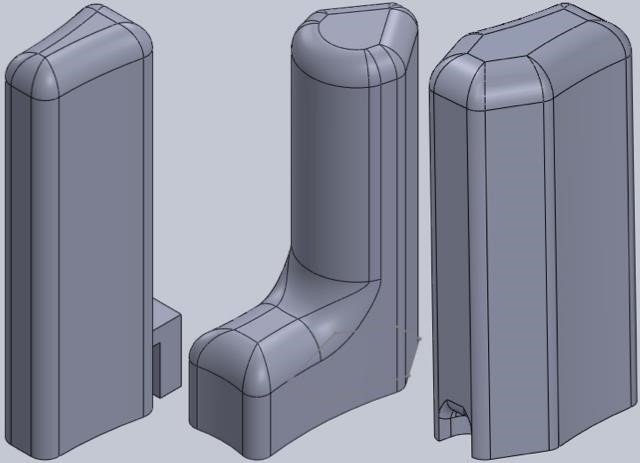
ניתן לראות בגוף המגן כי ישנם חיבורים לשבעה חיישני 𝐿𝑖𝐷𝐴𝑅. כעת מחוברים לרחפן שלושה חיישנים, זוהי הכמות המינימלית הנדרשת לטובת הפעלת הרחפן. הוספת חיישנים נוספים תאפשר חישובים של זווית הסבסוב של הרחפן והוספת פילטרים אשר ידייקו את המדידות. כמו כן, ניתן לראות כי במכסה הביניים יש פתחים המאפשרי העברת של חוטי החשמל בין השכבות לטובת גישה נוחה. בצורה מורכבת מערך זה נראה כך:



**איור 14 : הרכבת מגן הסוללה**

**4.1.2. רגלי הגבה ה**

השלדה עליה מבוסס הרחפן הינה מיועדת לרחפני מרוץ, ולכן ללא הגבהה משמעותית מהקרקע. הובחן כי בקונפיגורציה הבסיסית של השלדה חיישן ה-𝑂𝐹 היה קרוב מדי לרצפה ולכן לא התאפש ר לספק מדידות אמינות לטובת שיערוך מיקום הרחפן. הרגלים שתוכננו עבור הרחפן נועדו לספק הגבהה מספיקה על מנת שיתקבלו מדידות מחיישן ה-𝑂𝐹 כך שגם בשהייה על הרצפה תהיה מדידה אמינה של מיקום הרחפן.



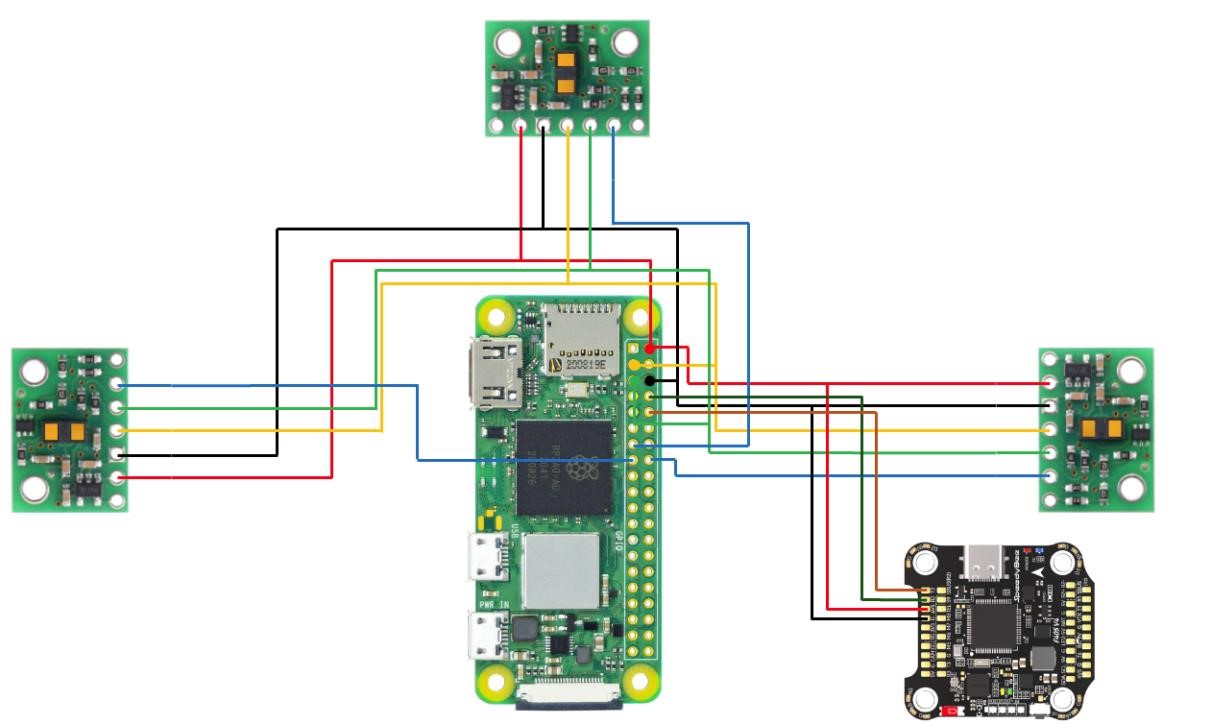
**איור 15 : רגלי הרחפן .מימין לשמאל- רגל אחורית, רגל קדמית ורגל צדדית**



**איור 16 : רגלי הרחפן מורכבות**

### 4.2. סכמה אלקטרוני ת

אופן החיבור החשמלי בין כלל המחשבים והחיישנים השונים מוצג באיור מטה.



**איור 17 : סכמה המציגה את החיבורים החשמליים במערכת**

בסכמה זו ניתן לראות כי כל חיבורי המתח במערכת משותפים, כאשר ההזנה של המתח ל-𝑅𝑃𝐼 ולחיישני 𝐿𝑖𝐷𝐴𝑅 הינה על ידי מתח המועבר באמצעות הסוללה, דרך בקר הטיסה. כמו כן, ניתן לראות את החיבור הטורי של כלל חיישני ה-𝐿𝑖𝐷𝐴𝑅 ואת החיבור של כלל חיבורי ה- 𝑋𝑆𝐻𝑈𝑇 של החיישנים )כחול כהה( לחיבורים שונים במחשב 𝑅𝑃𝐼.

### 4.3. פרמטרים חשובים בתוכנת ArduPilot

תוכנת ה-𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡 מתמקדת בהטסה מבוססת 𝐺𝑃𝑆. מכיוון שהרחפן בפרויקט זה מתבסס על חיישן 𝑂𝐹, נדרש לבצע מספר התאמות לפרמטרים של התכונה על מנת לאפשר הטסה מבוססת חיישן זה. בנוסף למוצג כאן, ישנם פרמטרים נוספים אשר משפיעים רבות על אופן טיסת הרחפן, כגון הגברי הבקרה, אך אינם מוצגים כאן מכיוון שהגדרתם יותר ברורה ודורשת ערכים אינדיבידואלים בהתאם לרחפן.

**4.3.1. פרמטרי משערך קלמ ן**

ישנם מספר משערכי קלמן שניתנים לשימוש בתוכנה, המומלץ, למועד כתיבת דוח זה, הינו קלמן פילטר 3.

על מנת להגדיר שימוש בחיישן ה-𝑂𝐹 ולא ב-𝐺𝑃𝑆 נדרש לשנות את הפרמטרים הבאים:

𝐸𝐾3\_𝑆𝑅𝐶1\_𝑃𝑂𝑆𝑋𝑌 = 0

𝐸𝐾3\_𝑆𝑅𝐶1\_𝑉𝐸𝐿𝑋𝑌 = 5

𝐸𝐾3\_𝑆𝑅𝐶1\_𝑃𝑂𝑆𝑍 = 1

1. 𝐸𝐾3\_𝑆𝑅𝐶1\_𝑉𝐸𝐿𝑍 =
2. 𝐸𝐾3\_𝑆𝑅𝐶1\_𝑌𝐴𝑊 =
3. 𝐸𝐾3\_𝑆𝑅𝐶\_𝑂𝑃𝑇𝐼𝑂𝑁𝑆 =

פרמטרים אלו מגדירים כי חיישן המדידה עבור מהירויות הרחפן הינו חיישן ה𝑂𝐹, כמו כן- חיישן המדידה של סבסוב הרחפן הינו מצפן.

𝑶𝒑𝒕𝒊𝒄𝒂𝒍 𝑭𝒍𝒐𝒘 **.4.3.2**

לצורך הגדרת חיישן ה-𝑂𝐹 נדרש להגדיר את הפרמטרים הבאים:

𝑆𝐸𝑅𝐼𝐴𝐿𝑥\_𝐵𝐴𝑈𝐷 = 115

1. 𝑆𝐸𝑅𝐼𝐴𝐿𝑥\_𝑃𝑅𝑂𝑇𝑂𝐶𝑂𝐿 =

𝐹𝐿𝑂𝑊\_𝑇𝑌𝑃𝐸 = 5

𝑅𝑁𝐺𝐹𝑁𝐷1\_𝑇𝑌𝑃𝐸 = 10

𝑅𝑁𝐺𝐹𝑁𝐷1\_𝑀𝐴𝑋\_𝐶𝑀 = 250

𝑅𝑁𝐺𝐹𝑁𝐷1\_𝑀𝐼𝑁\_𝐶𝑀 = 0

𝑅𝑁𝐺𝐹𝑁𝐷1\_𝑂𝑅𝐼𝐸𝑁𝑇 = 25 (𝐷𝑜𝑤𝑛𝑤𝑎𝑟𝑑)

פרמטרים אלו מגדירים באיזה חיבור 𝑈𝐴𝑅𝑇 התבצע החיבור של חיישן ה-𝑂𝐹, את פרוטוקול התקשורת בו הוא עובד, פרמטרים הקובעים את תחום העבודה של חיישן ה-𝐿𝑖𝐷𝐴𝑅 ואת הכיוון בו מודד החיישן. בפרויקט זה, החיישן חובר לחיבור 3𝑈𝐴𝑅𝑇, כפי שמוצג בטבלה מס '1, ולכן הגדרות אלה בוצעו עבור 3𝑆𝐸𝑅𝐼𝐴𝐿.

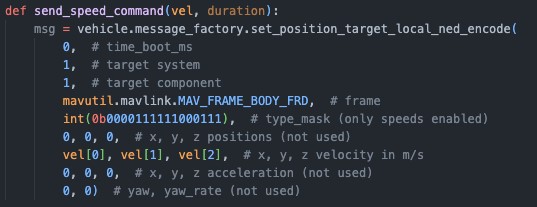
**4.3.3. פקודות למצב טיסה** 𝑮𝑼𝑰𝑫𝑬𝑫

מכיוון שמצב הטיסה בו נדרש לעבוד הינו 𝐺𝑈𝐼𝐷𝐸𝐷, יש להגדיר כי במצב זה בקר הטיסה נדרש לקבל פקודות מהירות ופקודות דחף. הפרמטר אשר מגדיר זאת הינו:

𝐺𝑈𝐼𝐷\_𝑂𝑃𝑇𝐼𝑂𝑁𝑆 = 9

### 4.4. פקודות הניתנות לשליחה למערכת

התוכנה 𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡 מאפשרת מספר אופציות שונות של פקודות אשר ניתן לשלוח לבקר טיסה, זאת בהתאם למצב הטיסה בו נמצאים ולחיישנים המחוברים לרחפן. במידה ונמצאים במצב טיסה 𝐺𝑈𝐼𝐷𝐸𝐷, ומחובר לרחפן חיישן 𝐺𝑃𝑆, ניתן לשלוח פקודות של תנועה לנ.צ מסוים במהירות ותאוצה מסוימים. מכיוון שהמערכת המפותחת בפרויקט זה מבוססת בעיקרה על חיישן ה- 𝑂𝐹, וחיישן זה מודד את מהירות טיסת הרחפן בצירים 𝑋, 𝑌. הפקודות הניתנות לשליחה לרחפן הן פקודות מהירות קוויות, תחת מצב טיסה 𝐺𝑈𝐼𝐷𝐸𝐷. ניתן גם לשלוח פקודות בסיסיות יותר של מהירות זוויתית תחת מצב טיסה 𝐺𝑈𝐼𝐷𝐸𝐷\_𝑁𝑂𝐺𝑃𝑆 אך פקודות אלה יסבכו את האלגוריתם וידרשו לבנות מערכת בקרה נוספת אשר תאפשר תגובה דינמית רצויה על המצב הזוויתי של הרחפן. מכיוון שמערכת בקרה זו כבר קיימת כחלק מהתוכנה על בקר הטיסה, הפקודות שישלחו בפרויקט זה יהיו פקודות מהירות קווית, תחת מצב טיסה 𝐺𝑈𝐼𝐷𝐸𝐷. מכיוון שהמערכת 𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡 מבוסס על הפרוטוקול 𝑀𝐴𝑉𝐿𝐼𝑁𝐾, הפקודה שתישלח לבקר טיסה צריכה להיות בפרוטוקול זה גם כן. לאחר מספר ניסויים נמצא כי הפקודה אשר עובדת ומאפשרת שליחת פקודות מהירות לרחפן הינה:



**איור 18 : מבנה פקודה למהירות קווית**

ניתן לראות כי המערכת צירים בה ניתנת הפקודה הינה מערכת צירים צמודת גוף אשר הצירים שלה מסודרים קדימה, ימינה, מטה) 𝐹𝑅𝐷(. כמו כן, יש משמעות גדולה לביט הנשלח למערכת. ביט זה מגדיר מה סוג הפקודה שתשלח ואם יש להתחשב גם בתאוצות או בשינוי זווית הסבסוב של הרחפן. מכיוון שקוד זו נועד לשלוח רק פקודה למהירות קווית, הביט שנשלח לבקר טיסה מגדיר התייחסות אך ורק לפקודת מהירות. ערך זה ניתן לשליחה גם בבינארי וגם בדצימלי, כאן הערך נשלח בדצימלי-

.𝑖𝑛𝑡(0𝑏0000111111000111)

### 𝑳𝒊𝑫𝑨𝑹 חיישני .4.5

חיישני ה-𝐿𝑖𝐷𝐴𝑅 ידגמו בתדירות של 𝐻𝑧 1.0, עם דגימה של מדידה חדשה יתבצע פילטור של הערכים על מנת לדייק את הערך המדוד. הפילטור יתבצע על ידי משקול של המדידה הנוכחית יחד עם שתי המדידות הקודמות, זאת על מנת למנוע את האפשרות בה כתוצאה מרעשי מדידה, המדידה הנוכחית תהיה רחוקה משמעותית מהערך האמיתי. כמו כן, ההשמה של המדידות בקוד תהיה לתוך משתנה מסוג מילון בפייתון.

משתנה זה מאפשר לשייך שם וערך מספרי לכל משתנה. משתנה זה מוגדר באופן הבא:

𝑟𝑎𝑛𝑔𝑒 = {"Front" : value, "Left" : value, "Right" : value, }

השמת הערכים למשתנה מסוג זה הינה נוחה מכיוון שהיא מאפשרת פניה לערך המדידה לפי פניה לכיוון המדידה. הפילטר אשר ממומש על המדידה החדשה מוגדר:

𝑟𝑎𝑛𝑔𝑒 = 𝐾𝑓 ⋅ 𝑟𝑎𝑛𝑔𝑒𝑘 + 𝐾𝑓−1 ⋅ 𝑟𝑎𝑛𝑔𝑒𝑘−1 + 𝐾𝑓−2 ⋅ 𝑟𝑎𝑛𝑔𝑒𝑘−2

האינדקס 𝑘 מסמל את צעד הזמן של המדידות. הקבועים 2−𝐾𝑓, 𝐾𝑓−1,𝐾𝑓 הינם המשקלים של כל מדידה לפני עדכון הערך המדוד הנוכחי. ערכי קבועים אלה:

𝐾𝑓−1 = 0.3 ;𝐾𝑓−1 = 0.2 ;𝐾𝑓 = 1 − 𝐾𝑓−1 − 𝐾𝑓−2

### 4.6. מערכת עזר- שמירת מרחק מקיר

מערכת העזר נועדה לשלוח פקודות לרחפן על מנת שישמור על מרחק של לפחות 15 ס"מ. על ידי קבלת מדידות של מרחק על ידי חיישני ה- 𝐿𝑖𝐷𝐴𝑅, המערכת תזהה במידה ובאחד מהכיוונים בהם קיים חיישן האם הרחפן קרוב יתר על המידה לקיר, מצב אשר מוביל לתופעות זרימ ה שכתוצאה מהן האוויר מקרב את הרחפן לקיר. במידה והרחפן קרוב מדי, תינתן פקודה ליצירת כוח דחף אשר יגרום לתנועה של הרחפן הרחק מהקיר. כפי שהוצג מעלה, הפקודות הניתנות לרחפן הינן פקודות של מהירות קווית ולכן הגדרת הכוח נובעת מהמהירות הקווית הרצויה .המהירות מחושבת על ידיד אינטרפולציה לינארית באופן הבא:

𝑣𝑚𝑎𝑥 − 𝑣𝑚𝑖𝑛

𝑣 = 𝑣𝑚𝑖𝑛 + (𝐿 − 𝐿𝑚𝑎𝑥) ⋅ )4.1(

𝐿𝑚𝑖𝑛 − 𝐿𝑚𝑎𝑥

כאשר הפרמטרים 𝑣𝑚𝑎𝑥, 𝑣𝑚𝑖𝑛,𝐿𝑚𝑎𝑥,𝐿𝑚𝑖𝑛 הינם פרמטרים אשר נקבעו באופן שרירותי לפי הגבלות המערכת. כמו כן, מכיוון שעם הקרבה לקיר כך גם גדלות תופעות הזרימה, נרצה שעם התקרבות לקיר מהירות התנועה של הרחפן תהיה גבוהה יותר . ניתן לראות ממשוואה 41. כי הנקודות על בסיסן מתבצעת האינטרפולציה הן [𝐿𝑚𝑎𝑥,𝑣𝑚𝑖𝑛],[𝐿𝑚𝑖𝑛,𝑣𝑚𝑎𝑥]. הערכים אשר נקבעו עבור נקודות אלה הם:

𝐿𝑚𝑖𝑛 = 0 𝑐𝑚

𝐿𝑚𝑎𝑥 = 15 𝑐𝑚

𝑣𝑚𝑖𝑛 = 0.2 𝑚 )4.2(

𝑠

𝑚

𝑣𝑚𝑎𝑥 = 0.5

𝑠

### 4.7. מערכת הניוו ט

מערכת הניווט בפרויקט נדרשת לנתח את ערכי המרחקים בכלל צידי הרחפן, להבחין מה הכיוון הפנוי, לשלוח פקודת סבסוב לרחפן על מנת שיפנה לכיוון זה ושליחת פקודה להמשך תנועה קדימה. המערכת מתעדפת המשך תנועה קדימה, אך ורק במידה והתנועה קדימה אינה מתאפשרת הרחפן יפנה לכיוון אחר הפנוי ביותר. בכל נקודת זמן בה נדרש מהמערכת לקבל החלטה, ערך זווית הסבסוב ישמר במערך ייעודי על מנת שיהיה תיעוד של מסלול הרחפן לצורך תנועה בכיוון ההפוך לטובת חזרה לנקודת ההתחלה. כמו כן, מערכת זו נדרשת לבצע הבחנה האם הרחפן נמצא בסביבה סגורה כגון מסדרון צר, או בסביבה פתוחה, כגון חדר או מצב של זיהוי קיר מצד אחד בלבד. עבור המצב הראשון, של מסדרון צר, מערכת הניווט נדרשת לשמור על מרכז הנתיב, בכך ישמר מרחק אידיאלי מדפנות הנתיב. במצב של מסדרון רחב, המערכת תזהה את הקיר הקרוב ביותר ותשתמש בו כעוגן לניווט. כלומר, שמירה על מרחק קבוע מקיר אחד. כלל פקודות הניווט יינת נ ו במהירות קבועה, זאת מכיוון שאין צורך בביצוע תמרונים מהירים. מהירות זו תיקרא מהירות שיוט וגודלה

𝑚

𝑣𝑐𝑟𝑢𝑖𝑠𝑒 = 0.7 )4.3(

𝑠

התנאי לתחילת טיסה חזרה מבוסס על מתח הסוללה הנשאר. עם תחילת הטיסה האוטונומית, מתח הסוללה ישמר במשתנה, 𝑉𝑎𝑢𝑡𝑜, אשר יהווה ערך להשוות אליו את המתח הנוכחי 𝑉𝑐𝑢𝑟𝑟. תחת ההנחה כי טיסת הרחפן החלה עם סוללה הטעונה באופן מלא [𝑉𝑠𝑡𝑎𝑟𝑡 = 4.2 [𝑉, החישוב לחזרת הרחפן יבוצע באופן הבא:

𝑖𝑓 (𝑉𝑠𝑡𝑎𝑟𝑡 − 𝑉𝑐𝑢𝑟𝑟) == (𝑉𝑐𝑢𝑟𝑟 − 𝑉𝑒𝑛𝑑): )4.4(

𝑆𝑡𝑎𝑟𝑡 𝑔𝑜 𝑏𝑎𝑐𝑘

כאשר [𝑉𝑒𝑛𝑑 = 3.5 [𝑉. חישוב זה מניח כי זמן המתח הנצרך הטיסה חזור זהה למתח בטיסה הלוך. כמו כן, הוא מתחשב בכך שזמן הטיסה כולל את זמן ההטסה שבוצע על ידי המפעיל ולא רק את זמן הטיסה האוטונומי.

### 4.8. אלגוריתמי סיוע וניוו ט

פרק זה יציג את האלגוריתמים הקיימים, יפרט את אופן עבודתם וידון על ביצועי האלגוריתמים בניסויים.

**4.8.1. שמירת מרחק מינימל י**

אלגוריתם זה נועד לשמירה על מרחק מינימלי של לפחות 15 ס"מ ממכשולים בצדדים, זאת על מנת להימנע מתופעות הזרימה אשר יגרמו להתנגשות הרחפן בקיר.

|  |
| --- |
| def safe\_dist(LiDAR\_data, LiDAR\_keys):  """  Control system to maintain safe distance from walls.  The function assumes that the drone is in a corridor with width of 30cm or larger.  The minimum distance from the wall is 15 cm. the amount of thrust needed is linear to the distance from the wall.  """  wall\_dir = None  # Check if distance is bellow minimum value: yaw\_curr = drone.attitude["yaw"] for key in LiDAR\_keys: if LiDAR\_data[key] < MIN\_WALL\_DISTANCE: wall\_dir = key wall\_distance = LiDAR\_data[key] |
| # Calculate the speed using Linear interpolation speed = np.interp(wall\_distance, [0, MIN\_WALL\_DISTANCE], [MAX\_SPEED,MIN\_SPEED]) # Send control command: if wall\_dir == "Front":  # Need to stop moving forward velocity = [0, 0, 0] elif wall\_dir == "Left1": # Need to move right velocity = [0, speed, 0] elif wall\_dir == "Right1": # Need to move left velocity = [0, -speed, 0]    # Send movement commands:  drone.linear\_speed(velocity, duration=0.1) drone.condition\_yaw(yaw\_curr, direction=1, relative=False) |

תחילה, הפונקציה עוברת על כלל המדידות מחיישני ה-𝐿𝑖𝐷𝐴𝑅 ובודקת אם המרחק הנקרא הינו קטן יותר מ-15 ס"מ. במידה וכן, מתבצע חישוב למהירות התגובה הנדרשת על ידי ביצוע אינטרפולציה לינארית כאשר הקבועים מוצגים במשוואה )4.2( ולבסוף נשלחות פקודות מהירות וזווית סבסוב לרחפן.

**4.8.2. קביעת עוגנים במהלך הניווט**

אזורי הניווט האוטונומיים הינם משתנים ויכולים להיות אזורים צרים ואזורים רחבים. לכן, נדרשת הבחנה באלגוריתם על מנת לקבוע עוגנים אשר בהם ישתמש הרחפן הזמן הניווט. במידה והאזור בו נמצא הרחפן הינו צר, יידר ש הרחפן לשמור על מרכז המסלול הקיים, כך ימנע מהתקרבות יתרה לקירות. במידה והאזור רחב, הרחפן יתקרב לקיר אח ד, זאת על מנת להשתמש בקיר זה כעוגן ולהימנע ממצב בו הרחפן נמצא במרכז חלל גדול וחיישני ה-𝐿𝑖𝐷𝐴𝑅 מודדים מרחקים גדולים ולא אמינים.

def corridor\_type(LiDAR\_data, yaw\_dec):

""" This Function is used to determine the corridor type based on the LiDAR data.

The function will return the corridor type: "Small" or "Large".

Small is when there are walls from two sides.

Large is when there is a wall from one side only and the other side is away.

"""

global CORRIDOR

|  |
| --- |
| # Check if the corridor is small: if LiDAR\_data["Left1"] < SMALL\_COR\_MAX and LiDAR\_data["Right1"] < SMALL\_COR\_MAX: CORRIDOR = "Small" main\_wall = min(LiDAR\_data, key=LiDAR\_data.get) # Return the key for the minimum distance wall.  save\_mid(drone,LiDAR\_data) else:  CORRIDOR = "Large" main\_wall = min(LiDAR\_data, key=LiDAR\_data.get) # Return the key for the minimum distance wall.  return main\_wall, CORRIDOR    def save\_mid(LiDAR\_data):  """  Function to navigate in small corridors. Save mid-lane yaw\_dec purpose if to save all the yaw decisions to be able to go back to the previous decision in case of a dead end or back to the start point.  """  diff = LiDAR\_data['Right1'] - LiDAR\_data['Left1'] if diff > 0: # Left wall is closer  speed = CRUISE\_SPEED # Move Right else: # Right wall is closer speed = -CRUISE\_SPEED # Move Left velocity = [0, speed, 0] yaw\_curr = drone.attitude["yaw"] # Send movement commands:  drone.linear\_speed(velocity, duration=0.05) drone.condition\_yaw(yaw\_curr, direction=1, relative=False) # so in case of same yaw, it will not change.    def save\_1wall(LiDAR\_data, yaw\_dec, close\_wall):  if LiDAR\_data[close\_wall] <= MIN\_WALL\_DISTANCE: # The distance from the wall is less than 15 cm if close\_wall == "Left1":  speed = CRUISE\_SPEED # Move right else:  speed = -CRUISE\_SPEED # Move left    if LiDAR\_data[close\_wall] > MIN\_WALL\_DISTANCE: # The distance from the wall is greater than 15 cm if close\_wall == "Left1": |
| speed = -CRUISE\_SPEED # Move left else:  speed = CRUISE\_SPEED # Move right    velocity = [0, speed, 0] yaw\_curr = drone.attitude["yaw"] # Send movement commands:  drone.linear\_speed(velocity, duration=0.1) drone.condition\_yaw(yaw\_curr, direction=1, relative=False) |

הפונקציה הראשונה נועדה לבצע את ההבחנה האם הרחפן נמצא באזור צר או רחב. זאת על ידי בדיקה האם מתקבלת מדידה מרחק משני צידי הרחפן או האם צד אחד מודד ערכים קטנים, והצד השני מודד ערכים מאוד גדולים הקרובים למרחק המקסימלי אותו החיישן יכול למדוד.

הפונקציה השניי ה נועדה לאפשר שמירה על מרכז המסלול. פונקציה זו משתמשת במדידות חיישני ה-𝐿𝑖𝐷𝐴𝑅 לזהות את ההפרש בין מדידות החיישנים ובכך קובעת לאיזה כיוון נדרש לטוס. אם 0 >𝑑𝑖𝑓𝑓 , הקיר השמאלי הינו קרוב יותר מהקיר הימני ולכן נדרש תנועה ימינה. מכיוון שמערכת הצירים בה ניתנות הפקודות היא מערכת 𝐹𝑅𝐷, פקודת תזוזה ימינה הינה דורשת מהירות חיובית. במידה ו- 0 <𝑑𝑖𝑓𝑓 , קיר ימין הינו קרוב יותר ונדרשת תנועה שמאלה.

הפונקציה השלישית נועדה למסלולים רחבים. פונקציה זו מקבלת את האבחון של 𝑐𝑜𝑟𝑟𝑖𝑑𝑜𝑟\_𝑡𝑦𝑝𝑒 איזה קיר הכי קרוב. ומבצעת התקרבות לקיר זה במידה והרחפן רחוק יותר או מתרחקת מהקיר במידה והרחפן קרוב מדי.

**4.8.3. ניווט**

פונקציה זו הינה אלגוריתם הניווט הבסיסי שפותח בפרויקט זה. תעדוף הכיוונים הינו בסיסי- במידה וניתן להמשיך לטוס קדימה, המשך טיסה קדימה. במידה ולא ניתן- לבחון איזה מהכיוונים הינו פנוי יותר, לבצע פניה כך שהרחפן יהיה עם הפנים לכיוון הפנוי ולאחר מכן המשך טיסה קדימה, בנתיב החדש.

|  |
| --- |
| def navigate(LiDAR\_data, yaw\_dec): # find the side with the most space yaw\_curr = drone.attitude["yaw"]    # if front is clear: if LiDAR\_data["Front"] > LiDAR\_data["Left1"] and LiDAR\_data["Front"] > LiDAR\_data["Right1"]:  wall\_clr = "Front"  # In case front is not clear: elif LiDAR\_data["Left1"] > LiDAR\_data["Right1"]:  wall\_clr = "Left" |
| else:  wall\_clr = "Right"    yaw\_curr = drone.attitude["yaw"]    # Send control command: if wall\_clr == "Left1": # Need to turn Left yaw\_new = 90 direction = -1 elif wall\_clr == "Right1": # Need to turn Right yaw\_new = 90 direction = 1 else: #Front is clear  # Need to stop moving forward yaw\_new = yaw\_curr    # Send movement commands: velocity = [CRUISE\_SPEED, 0, 0] drone.condition\_yaw(yaw\_new, direction, relative=True) time.sleep(0.05) #save the current yaw print("New Direction: ", wall\_clr) |

## 5. ניסויים

במהלך הפיתוח בוצעו מספר ניסויים על מנת לאמת את יכולות הרחפן. פרק זה יציג את הניסויים ואת תוצאותיהם.

### 5.1. ניסויי המרא ה

על מנת לאמת את יכולות הרחפן להמריא באופן אוטונומי לגובה מוגדר מראש בוצעו מספר ניסויים הבודקים את יכולת זו. בין הניסויים בוצעו שינויים בפילטורים המבוצעים על חיישני הגובה השונים על מנת לקבל את התגובה הרצויה. התוצר הסופי שהתקבל הינו-



**איור 19 : גובה הרחפן. באדום- ברומטר, בירוק חיישן** 𝑳𝒊𝑫𝑨𝑹

באיור 19 ניתן לראות שני תהליכי המראה ונחיתה. הראשון מתחיל ב 6:45 שניות והשנה מתחיל ב 7:30 שניות. באדום ניתן לראות את הערך המדוד על ידי הברומטר ובירוק את הערך המדוד על ידי חיישן ה-𝐿𝑖𝐷𝐴𝑅 אשר מחובר דרך חיישן ה-𝑂𝐹. ניתן לראות באדום כי המדידה מאוד רועשת ומאוד גבוהה בהשוואה למוצג בירוק. הפקודה שנשלחה לרחפן הינה המראה לגובה של מטר, שהיה ונחיתה. סט פקודות זה נשלח פעמיים לרחפן על מנת לבדוק את תגובת הרחפן. מגרף זה ניתן להסיק כי חיישן הברומטר רווי ברעשים והמדידה שלו אינה אמינה ועל כ ן הגובה בהינת ן פקודות ההמראה והנחיתה יקבע על ידי חיישן ה-𝐿𝑖𝐷𝐴𝑅. מהגרף הירוק של חיישן ה-𝐿𝑖𝐷𝐴𝑅 ניתן לזהות עליה הדרגתית בגובה הרחפן ,𝑂.𝑆 אשר הוביל לערך מקסימלי של 11. מ' ותנודתיות סביב הערך של 1 מ. ' כמו כן, ניתן לזהות ירידה הדרגתית בגובה והמראה נוספת. מניסוי זה ניתן לראות כי התוצאות שהתקבלו הינן טובות ומראות את יכולות הרחפו להמריא באופן עצמאי ושמירה על גובה ללא צורך בהתערבות המפעיל.

## 6. סיכום

בפרויקט זה נבנה רחפן אוטונומי אשר נועד לטוס באזורים צרים בהם נדרשת הימנעות ממכשולים, כאשר המכשול העיקרי הינו קיר. בוצע סקר ספרות וסקר שוק למערכות הנדרשות ונבחרו רכיבים המתאימים ביותר בהתאם לדרישות הפרויקט ולתקציב מינימלי. בוצעו מספר הרכבות במהלך הפרויקט לטובת תיקון ושיפור הרחפן עד אשר התקבלה פלטפורמה עמידה בעלת כלל הרכיבים הנדרשים.

עקב השימוש בתוכנה קיימת (𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡), הפרויקט עמד במספר מכשולים ובהגבלות הנובעות מאופי התוכנה. להמשך, לצורך הרחבת תחום העבודה של פרויקט זה, יתכן וידרש לפתח תוכנה ייעודית לבקרת רחפנים על מנת להתגבר על כך ש-𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡 מונע מספר אפשרויות טיסה כתוצאה מכך שאין שימוש ב-

.𝐺𝑃𝑆

כמו כן, בוצע פיתוח של מספר אלגוריתמי ם לטובת אוטומציית הרחפן. האלגוריתם העיקרי הינו אלגוריתם הניווט, בפרויקט זה פותח אלגוריתם בסיסי ולא אופטימלי אשר נועד להראות היתכנות של הרעיון. להמשך, נדרש לתכנן ולפתח אלגוריתם יעיל יותר בעל יכולות נוספות אשר יאפשרו לוקליזציה ומיפוי של השטח בו טס הרחפן. בנוסף לאלגוריתם הניווט, פותח אלגוריתם אשר נועד לשמור על מרחק מינימלי מקירות, להמשך, נדרש לבצע אינטגרציה של אלגוריתם זה כחלק משרשרת הבקרה של הרחפן בעת הטסה על ידי פקודות מפעיל ולא בצורה אוטונומית. זאת על מנת להקל על המפעיל בעת טיסה.

## 7. המלצות להמשך

עם סיום פרויקט זה , המבנה הבסיסי של רחפן הורחב וכעת כולל בנוסף גם שלושה חיישני 𝐿𝑖𝐷𝐴𝑅 וחיישן 𝑂𝐹 הפונה כלפי מטה. קונפיגורציה זו של החיישנים הינה המינימלית הנדרשת לפיתוח הרחפן. ישנן מספר המלצות להמשך אשר יכולות לשפר את ביצועי הרחפן וכעת נציג אותן.

### 7.1. הוספת חיישני 𝑳𝒊𝑫𝑨𝑹

הוספת חיישנים אל ו על מנת שקונפיגורצית החיישנים תהיה: 3 קדימה, 2 לכל צד, 1 אחורה, 1 למעלה, 1 למטה תאפשר לבצע פילטורים יעילים יותר על המדידות לטובת העלאת האמינות. כמו כן, תתאפשר מדידה של סבסוב הרחפן בתוך התווך הסגור. קונפיגורציה זו תאפשר להרחיב את יכולות ההימנעות ממכשולים גם במידה וישנם מכשולים הנמצאים בגובה, מעל הרחפן.

### 𝑶𝒑𝒕𝒊𝒄𝒂𝒍 𝑭𝒍𝒐𝒘 הוספת חיישני .7.2

החיישן העיקרי עליו מסתמך הרחפן הינו חיישן ה-𝑂𝐹. בפרויקט זה פותח רחפן המתבסס על חיישן יחיד מסוג זה. חוסר תפקוד בחיישן יוביל לקריסה של הרחפן ואי עמידה במטרה. על ידי הוספת 3 חיישני 𝑂𝐹 נוספים אשר יתווספו לחיישני 𝐿𝑖𝐷𝐴𝑅 הקיימים אשר פונים לצדדים וחיישן נוסף אשר יפנה כלפי מעלה תתאפשר מדידה נוספת של מהירויות הרחפן ובכך להקטין את הסיכוי לנפילת הרחפן. נדרש לקחת בחשבון כי מנהרות הינן אזור מאובק, דבר המשפיע על חיישן ה-𝑂𝐹. כתוצאה מדחיפת האוויר מהרחפן, זיהו י המשטח עליו נע הרחפן יהיה קשה וחסום על ידי חלקיקי אבק. הוספת חיישנים אלו יאפשרו לבצע מדידות נוספות של מהירויות הרחפן בצירים הרלוונטים.

### 7.3. שיפור אלגוריתם הניוו ט

אלגוריתם הניווט אשר פותח בפרויקט זה הינו אלגוריתם בסיסי אשר נועד לבחון את היתכנות הרעיון. על מנת למקסם את יכולות הניווט נדרש לפתח אלגוריתם אשר יאפשר, מעבר לניווט הרחפן, גם מיפוי של המנהרה כאשר בסוף שלב הניווט יספק הרחפן מפה של המנהרה בה טס.

### 7.4. התערבות אקטיבית בפקודות המפעי ל

עקב הגבלות של 𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡, אין מצב טיסה המאפשר שליטה של המפעיל על התערבות של מחשב נוסף כך שבקר הטיסה יקבל פקודות גם משלט המפעיל וגם ממחשב נוסף. דבר זה מונע את האפשרות של סיוע אקטיבי למפעיל שימנע ממכשולים. הפתרון האפשרי יחד עם תוכנת ה-𝐴𝑟𝑑𝑢𝑃𝑖𝑙𝑜𝑡 הינו מסורבל ודורש העברת פקודות השלט ל-𝑅𝑃𝐼, זיהוי האם הפקודה תוביל להתנגשות במכשול ולעקוף את פקודה זו. פתרון זה אינו אידיאלי עקב השלבים הרבים אשר נדרשים ועקב כך שאין יכולת למנוע את שליחת הפקודה לבקר טיסה עד לקבלת תגובה מהמערכת התומכת.

## 8. ביבליוגרפיה

Understanding UART, Rohde& Schwarz[, https://www.rohde-schwarz.com/us/products/testhttps://www.rohde-schwarz.com/us/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart\_254524.htmland-measurement/essentials-test-equipment/digital-oscilloscopes/understandinghttps://www.rohde-schwarz.com/us/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart\_254524.htmluart\_254524.html#:~:text=UART%20stands%20for%20universal%20asynchronous,and%20rec](https://www.rohde-schwarz.com/us/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart_254524.html#:~:text=UART%20stands%20for%20universal%20asynchronous,and%20receive%20in%20both%20directions)

[.eive%20in%20both%20directions](https://www.rohde-schwarz.com/us/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart_254524.html#:~:text=UART%20stands%20for%20universal%20asynchronous,and%20receive%20in%20both%20directions) I2C, Sparkfun, <https://learn.sparkfun.com/tutorials/i2c/all>Connect ESC & Motors, Ardupilot, [https://ardupilot.org/copter/docs/connect-escs-andhttps://ardupilot.org/copter/docs/connect-escs-and-motors.htmlmotors.html](https://ardupilot.org/copter/docs/connect-escs-and-motors.html)

A Review of quadrotor UAV: Control and SLAM methodologies ranging from conventional to innovative approaches, Guray Sonugur,

<https://www.sciencedirect.com/science/article/abs/pii/S0921889022002317>Inertial Measuring Unit, [https://i1.wp.com/bestperformancegroup.com/wphttps://i1.wp.com/bestperformancegroup.com/wp-content/uploads/2012/05/Fig\_5.jpgcontent/uploads/2012/05/Fig\_5.jpg](https://i1.wp.com/bestperformancegroup.com/wp-content/uploads/2012/05/Fig_5.jpg)

Hermetic Packages for LiDAR Sensors, Schott[https://www.schott.com/enhttps://www.schott.com/en-th/products/hermetic-packages-for-lidar-sensors-p1000281/applicationsth/products/hermetic-packages-for-lidar-sensors-p1000281/applications](https://www.schott.com/en-th/products/hermetic-packages-for-lidar-sensors-p1000281/applications)

Optical Flow 3901-L0X, Mateksys, <https://www.mateksys.com/?portfolio=3901-l0x>

How Brushless Motors work & How to test them, Emrik Joner, September 29, 2023, <https://www.tytorobotics.com/blogs/articles/how-brushless-motors-work>

Optical flow sensor testing and setup, Ardupilot, [https://ardupilot.org/copter/docs/commonhttps://ardupilot.org/copter/docs/common-optical-flow-sensor-setup.htmloptical-flow-sensor-setup.html](https://ardupilot.org/copter/docs/common-optical-flow-sensor-setup.html)

MTF-01 Optical flow, RoboFusion, [https://robofusion.net/products/mtf-01-integrated-laserhttps://robofusion.net/products/mtf-01-integrated-laser-rangefinder-optical-flow-module-for-unmanned-aerial-vehicles-uavsrangefinder-optical-flow-module-for-unmanned-aerial-vehicles-uavs](https://robofusion.net/products/mtf-01-integrated-laser-rangefinder-optical-flow-module-for-unmanned-aerial-vehicles-uavs)

Lithium-ion vs Lithium-polymer batter, Robocraze, [https://robocraze.com/blogs/post/lithiumhttps://robocraze.com/blogs/post/lithium-ion-vs-lithium-polymer-batteryion-vs-lithium-polymer-battery](https://robocraze.com/blogs/post/lithium-ion-vs-lithium-polymer-battery)

Itroduction to LiDAR, MathWorks, [https://www.mathworks.com/help/lidar/ug/lidarhttps://www.mathworks.com/help/lidar/ug/lidar-processing-overview.htmlprocessing-overview.html](https://www.mathworks.com/help/lidar/ug/lidar-processing-overview.html)

Vector Field Histogram, MathWorks, [https://www.mathworks.com/help/nav/ug/vector-fieldhttps://www.mathworks.com/help/nav/ug/vector-field-histograms.htmlhistograms.html](https://www.mathworks.com/help/nav/ug/vector-field-histograms.html)

What is inertial measuring unit, Vectornav, [https://www.vectornav.com/resources/inertialhttps://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu?gad\_source=1&gclid=Cj0KCQjwxeyxBhC7ARIsAC7dS3\_jltLGyyW0-08VLTvzWoPPf9VKRhuf3h71sUSJ\_fGNqVMHY2IEKX4aAhADEALw\_wcBnavigation-articles/what-is-an-inertial-measurement-unithttps://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu?gad\_source=1&gclid=Cj0KCQjwxeyxBhC7ARIsAC7dS3\_jltLGyyW0-08VLTvzWoPPf9VKRhuf3h71sUSJ\_fGNqVMHY2IEKX4aAhADEALw\_wcBimu?gad\_source=1&gclid=Cj0KCQjwxeyxBhC7ARIsAC7dS3\_jltLGyyW0-](https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu?gad_source=1&gclid=Cj0KCQjwxeyxBhC7ARIsAC7dS3_jltLGyyW0-08VLTvzWoPPf9VKRhuf3h71sUSJ_fGNqVMHY2IEKX4aAhADEALw_wcB)

[08VLTvzWoPPf9VKRhuf3h71sUSJ\_fGNqVMHY2IEKX4aAhADEALw\_wcB](https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu?gad_source=1&gclid=Cj0KCQjwxeyxBhC7ARIsAC7dS3_jltLGyyW0-08VLTvzWoPPf9VKRhuf3h71sUSJ_fGNqVMHY2IEKX4aAhADEALw_wcB)

## 9. נספחים

### 9.1. נספח א'- קטעי קו ד

**9.1.1. קטע קוד- איגוד פקודות לרחפ ן**

|  |
| --- |
| """  This module will contain all functions needed to fetch data from the flight controller using dronekit First all the getters are set and then the setters.  https://github.com/dronekit/dronekit-python/blob/master/examples/set\_attitude\_target/set\_attitude\_target.py |

https://github.com/dronekit/dronekit-python/pull/712/commits/61e4dbda8693128133d488048f52a5187129f208

|  |
| --- |
| https://mavlink.io/en/messages/common.html#SET\_ATTITUDE\_TARGET https://ardupilot.org/copter/docs/common-non-gps-navigation-landing-page.html#common-non-gps-navigationlanding-page https://github.com/sieuwe1/Autonomous-Ai-drone-scripts/blob/main/modules/control.py    """    from dronekit import \* import time from pymavlink import mavutil import numpy as np    vehicle = None    attitude = { 'pitch': 0,  'roll': 0,  'yaw': 0,  'thrust': 0  }  lidar\_rng = 0    home\_position = None    Ts =1/50    def connect\_vehicle(connection\_string, baud\_rate=921600, wait\_ready=True):  # Connect to the Vehicle global vehicle if vehicle == None:  vehicle = connect(connection\_string, baud=baud\_rate, wait\_ready=wait\_ready) print('Connected to vehicle')    def close\_connection():  global vehicle |

if vehicle != None:

vehicle.close() print('Vehicle connection closed') vehicle = None

def get\_mode(): global vehicle if vehicle != None:

|  |
| --- |
| return vehicle.mode.name else:  print('Vehicle not connected') return None    # On local frame because we dont have GPS and we need the distance from the home position which defined using the local frame def get\_location(): global vehicle if vehicle != None:  return vehicle.location.local\_frame else:  print('Vehicle not connected') return None    def get\_altitude(): global vehicle if vehicle != None:  return vehicle.location.global\_relative\_frame.alt else:  print('Vehicle not connected') return None    def get\_velocity(): global vehicle if vehicle != None:  return vehicle.velocity else:  print('Vehicle not connected') return None    def get\_attitude(): |

global vehicle if vehicle != None:

return vehicle.attitude # Pitch, Yaw, Roll return None

def get\_battery\_info():

|  |
| --- |
| global vehicle if vehicle != None:  return vehicle.battery else:  print('Vehicle not connected') return None    def get\_flightmode():  global vehicle if vehicle != None:  return vehicle.mode.name else:  print('Vehicle not connected') return None    def get\_home\_location():  global vehicle if vehicle != None:  return vehicle.location.local\_frame else:  print('Vehicle not connected') return None    def get\_heading(): global vehicle if vehicle != None:  return vehicle.heading else:  print('Vehicle not connected') return None    def get\_groundspeed(): |

global vehicle if vehicle != None: return vehicle.groundspeed

return None

# Channel can be a string and will import to channel num using the mapping. if it is a number it will be used as

|  |
| --- |
| is def read\_channel(channel): if type(channel) == str:  channel\_mapping = {"Throttle": 3, "Roll": 1, "Pitch": 2, "Yaw": 4, "AUX1": 5, "AUX2": 6, "AUX3": 7, "AUX4": 8} global vehicle if vehicle != None:  return vehicle.channels[channel\_mapping(channel)] else:  print('Vehicle not connected') return None    def set\_groundspeed(speed): global vehicle if vehicle != None:  vehicle.groundspeed = speed print('Vehicle groundspeed set to %s' % speed) else:  print('Vehicle not connected')    def set\_flightmode(mode):  global vehicle if vehicle != None:  vehicle.mode = VehicleMode(mode) while not vehicle.mode.name==mode: #Wait until mode has changed print(" Waiting for mode change ...") time.sleep(1) print('Vehicle mode set to %s' % mode) else:  print('Vehicle not connected')    def set\_velocity(velocity):  global vehicle |

if vehicle != None:

vehicle.velocity = velocity print('Vehicle velocity set to %s' % velocity)

def set\_channel(channel, value): channel\_mapping = {"Throttle": 3, "Roll": 1, "Pitch": 2, "Yaw": 4, "AUX1": 5, "AUX2": 6, "AUX3": 7, "AUX4": 8}

|  |
| --- |
| global vehicle if vehicle != None:  vehicle.channels.overrides[channel\_mapping(channel)] = value print('Channel %s set to %s' % (channel, value)) else:  print('Vehicle not connected')    def arm(mode= "GUIDED"):  global vehicle if vehicle != None:  #while not vehicle.is\_armable:  # print('Waiting for vehicle to become armable...')  # time.sleep(1) vehicle.mode = VehicleMode(mode) while not vehicle.mode.name==mode: print(" Waiting for mode change ...") time.sleep(1) vehicle.armed = True while vehicle.armed != True:  time.sleep(0.1) print('Vehicle armed') else:  print('Vehicle not connected')    def disarm():  global vehicle if vehicle != None:  vehicle.armed = False print('Vehicle disarmed') else:  print('Vehicle not connected') |

def takeoff(TargetAltitude):

global vehicle if vehicle != None: print("Taking off!")

|  |
| --- |
| vehicle.simple\_takeoff(TargetAltitude) # Take off to target altitude while True:  current\_altitude = vehicle.location.global\_relative\_frame.alt print("Altitude: ", current\_altitude) if current\_altitude >= TargetAltitude \* 0.95:  print("Reached target altitude") break time.sleep(1) else:  print('Vehicle not connected')    def land():  global vehicle if vehicle != None: print("Landing!") vehicle.mode = VehicleMode("LAND") while vehicle.location.global\_relative\_frame.alt >= 0.1:  print("Altitude: ", vehicle.location.global\_relative\_frame.alt) time.sleep(1) print("Landed!") else:  print('Vehicle not connected')    def send\_heading\_command(heading, relative = True):  global vehicle  if relative:  is\_relative = 1 if vehicle != None:  speed = 0 direction = 1 # 1 = CW and -1 = CCW    if heading < 0: heading = heading \* -1 direction = -1  msg = vehicle.message\_factory.command\_long\_encode(0, 0, mavutil.mavlink.MAV\_CMD\_CONDITION\_YAW, 0, heading, speed, direction, is\_relative, 0, 0, 0) vehicle.send\_mavlink(msg) else: |

|  |
| --- |
| print('Vehicle not connected')    def attitude\_listener(self, name, msg):  global attitude attitude['pitch'] = msg.pitch attitude['roll'] = msg.roll attitude['yaw'] = msg.yaw  #print(f"Updated Attitude Data: {attitude}")    def lidar\_listener(self, name, msg):  global lidar\_rng lidar\_rng = msg.distance  #print(f"Updated Lidar Data: {attitude}"    def add\_attributes():  global vehicle if vehicle != None:  vehicle.add\_attribute\_listener('attitude', attitude\_listener) vehicle.add\_attribute\_listener('rangefinder', lidar\_listener) else:  print('Vehicle not connected')    def arm\_and\_takeoff(aTargetAltitude):  global vehicle    #set default groundspeed low for safety print ("setting groundspeed to 3") vehicle.groundspeed = 2    print ("Basic pre-arm checks")  # Don't try to arm until autopilot is ready while not vehicle.is\_armable:  print (" Waiting for vehicle to initialise...") time.sleep(1)    print ("Arming motors")  # Copter should arm in GUIDED mode vehicle.mode = VehicleMode("GUIDED") vehicle.armed = True |

|  |
| --- |
| # Confirm vehicle armed before attempting to take off while not vehicle.armed:  print (" Waiting for arming...") time.sleep(1)    print ("Taking off!") vehicle.simple\_takeoff(aTargetAltitude) # Take off to target altitude    # Wait until the vehicle reaches a safe height before processing the goto (otherwise the command # after Vehicle.simple\_takeoff will execute immediately). while True:  # print (" Altitude: ", vehicle.location.global\_relative\_frame.alt) print (" Altitude: ", lidar\_rng)  #Break and return from function just below target altitude.  #if vehicle.location.global\_relative\_frame.alt>=aTargetAltitude\*0.95: if lidar\_rng>=aTargetAltitude\*0.95: print ("Reached target altitude") break time.sleep(1)    def condition\_yaw(heading, direction=1, relative=True):  """  Send MAV\_CMD\_CONDITION\_YAW message to point vehicle at a specified heading (in degrees).    This method sets an absolute heading by default, but you can set the `relative` parameter to `True` to set yaw relative to the current yaw heading.    By default the yaw of the vehicle will follow the direction of travel. After setting the yaw using this function there is no way to return to the default yaw "follow direction of travel" behaviour (https://github.com/diydrones/ardupilot/issues/2427)     * heading (int): The heading in degrees (0 = North, positive values clockwise) * direction (int): Direction (1 = clockwise, -1 = counter-clockwise)   """  # create the CONDITION\_YAW command using command\_long\_encode() msg = vehicle.message\_factory.command\_long\_encode(  0, 0, # target system, target component  mavutil.mavlink.MAV\_CMD\_CONDITION\_YAW, #command  0, #confirmation heading, # param 1, yaw in degrees |
| 0, # param 2, yaw speed deg/s direction, # param 3, direction -1 ccw, 1 cw relative, # param 4, relative offset 1, absolute angle 0  0, 0, 0) # param 5 ~ 7 not used  # send command to vehicle vehicle.send\_mavlink(msg)    def send\_speed\_command(vel, duration):  msg = vehicle.message\_factory.set\_position\_target\_local\_ned\_encode( 0, # time\_boot\_ms  1, # target system  1, # target component mavutil.mavlink.MAV\_FRAME\_BODY\_FRD, # frame int(0b0000111111000111), # type\_mask (only speeds enabled)  0, 0, 0, # x, y, z positions (not used) vel[0], vel[1], vel[2], # x, y, z velocity in m/s  0, 0, 0, # x, y, z acceleration (not used)  0, 0) # yaw, yaw\_rate (not used)    def linear\_speed(vel,duration):  start\_time = time.time() print("Sending Speed command") while time.time() - start\_time < duration:  send\_speed\_command(vel) time.sleep(Ts) print("Speed command finished")    if \_\_name\_\_ == "\_\_main\_\_":  connect\_vehicle('/dev/ttyS0') add\_attributes() print(get\_flightmode()) if get\_flightmode() == "GUIDED":  arm\_and\_takeoff(1) disarm() close\_connection()  exit() |

#### 9.1.2. קטע קוד- הפעלת חיישני 𝑳𝒊𝑫𝑨𝑹

|  |
| --- |
| import VL53L1X as VL import RPi.GPIO as GPIO import drone import numpy as np    range = [] range\_p1 = [] range\_p2 = []    Kf1 = 0.3  Kf2 = 0.2  Kf = 1 - Kf1 - Kf2    def setup(tof\_list, XSHUT\_list):    for i, XSHUT in enumerate(XSHUT\_list):  GPIO.setup(XSHUT\_list[i], GPIO.OUT) GPIO.output(XSHUT\_list[i], False) for i,tof in enumerate(tof\_list):  tof\_list[i] = VL.VL53L1X(i2c\_bus=1, i2c\_address=0x29) GPIO.output(XSHUT\_list[i], True) tof\_list[i].open()  tof\_list[i].change\_address(0x30 + 2\*i) GPIO.setup(XSHUT\_list[i], GPIO.IN) tof\_list[i].open() tof\_list[i].start\_ranging(2) tof\_list[i].set\_timing(20000, 25) # 20 micro s timing budget and 25 milli s inter-measurement delay this will give 50 Hz update rate  return tof\_list    def read\_dist(tof\_list,range): for i,tof in enumerate(tof\_list):  range[i] = tof\_list[i].get\_distance()/10 # Returns distance in cm return range    def filter\_dist(range, range\_p1, range\_p2,keys): for i in keys: |
| range[i] = Kf\*range[i] + Kf1\*range\_p1[i] + Kf2\*range\_p2[i] if range[i] < 0: range[i] = 0 if range[i] > 200: range[i] = 200 return range    def close\_connection(tof\_list): for i,tof in enumerate(tof\_list):  tof\_list[i].stop\_ranging() tof\_list[i].close()  GPIO.cleanup() |