



The Iby and Aladar Fleischman  
Faculty of Engineering  
Tel Aviv University

הפקולטה להנדסה  
ע"ש איבי ואלדר פליישמן  
אוניברסיטת תל אביב



---

שם הפרוייקט:

רחפן זעיר ומבוקר בעל יכולת טיסה יציבה בתוך חדר  
בצורה אוטונומית מלאה.

---

פרויקט גמר מס' 1816-1-2-18

# דו"ח סיכום

#####

דן סיבוני

#####

מוחמד ג'מאל

מנחה: יונתן מנדל, אוניברסיטת תל אביב

מקום ביצוע הפרוייקט:

אוניברסיטת תל אביב, מעבדת לרחפנים אוטונומיים

## תוכן עניינים

|         |                             |
|---------|-----------------------------|
| 3.....  | תקציר                       |
| 5.....  | הקדמה                       |
| 7 ..... | מרכיבי המערכת               |
| 10..... | תכנון ומימוש                |
| 11..... | בחירת החומרה והרכבתה        |
| 13..... | שלבי הרצת התוכנה            |
| 20..... | אינטגרציה עם בקרת הרחפן     |
| 25..... | תוצרי הפרויקט               |
| 26..... | סרטוני הדגמה                |
| 28..... | סיכום, מסקנות והמלצות להמשך |
| 31..... | רשימת מקורות                |

## **תקציר**

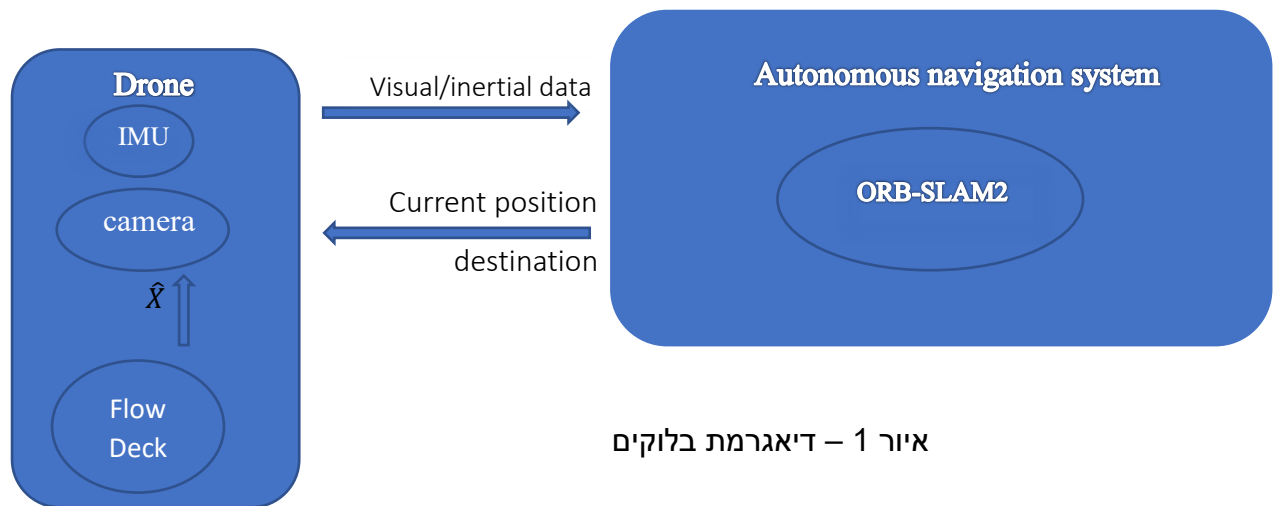
הפרויקט עוסק ברחפן זעיר ששוקל כמעט 30 גרם ואינו גדול מ 10 ס"מ מרובע שמקבל יכולת אוטונומית שמאפשרת לו לטוס בכלל חלל סגור ללא מערכת שליטה ישירה וללא מפעיל חיצוני שמלווה אותו בטיסה.

המרכיבים המרכזיים בפרויקט הם הננו רחפן עצמו שהוא בעצם Open Source Board שנועד לפרויקטים מסוג הזה, רכיב Flow Deck הנותן לרחפן בקרה על ההמראה בלבד ומצלמה אנאלוגית פשוטה היושבת על הרחפן ונותנת את הקלט לאלגוריתם שלנו: מערכת המאפשרת ליצור בקרת תנועה וניווט אוטונומיים בחלל סגור.

העבודה היא בשני התחומים, חומרה ותוכנה, כך שבשלבים הראשונים של הפרויקט יהי החלק של החומרה, בו נחבר את הפרופלורים, הסוללה והמצלמה שנחבר לגוף הלוח/הרחפן, לממש את האינטגרציה בין כל החלקים ולבדוק את תקינות הבקרה שתתקיים בינם מבחינת חומרה ותקשורת, החלק השני זה התוכנה של הפרויקט שהיא הנשמה שתניע המערכת הפיזית שתוארה קודם, בשלב זה יהי יותר מטלות, מהם עיבוד תמונה/סרט שמצלמת המצלמה בזמן אמת, וכתיבת אלגוריתמים לקביעת את מיקום המערכת הטסה ביחס לחדר הסגור. עם הנתונים שאוספו מהסרט ועם התוכנה שנממש נלמד את הננו רחפן איך לטוס לכיוונים ולמרחקים ספציפיים בצורה מסודרת ונכונה.

### **המערכת מורכבת מכמה מרכיבים :**

1. ננו-רחפן מסוג crazyflie - בעל בקר ניווט פנימי, יחידת IMU מובנית.
2. רכיב Flow Deck - המכיל חיישן גובה VL53L0x וחיישן אופטי מסוג PMW3901 למדידת התנועה ביחס לקרקע.
3. מצלמה זעירה - עם משדר רדיו מסוג crazypony יחד עם מקלט רדיו (5.8G) בחיבור USB.
4. מערכת שליטה – רצה בסביבת ROS ומכילה את האלגוריתם ORB-SLAM2 (SLAM).



תרשים הזרימה המתואר באיור 1, מתאר את התהליך הכללי הבא :

1. הרחפן מקבל וקטור מיקום מה Flow Deck (עבור תהליך ההמראה בלבד)
2. המצלמה משדרת את הווידאו בלייב למחשב עליו רצה מערכת השליטה.
3. באלגוריתם ה ORB-SLAM2 מחושב וקטור המיקום ב-3 ממדים  $(x,y,z)$ .
4. המידע נאסף, מעובד עם האלגוריתם, ונשלח בחזרה לרחפן כמיפוי ומיקומו בתוך המפה.
5. במקביל, נשלח לרחפן מיקום היעד הבא שלו.

## הקדמה

### מטרות הפרויקט:

- פיתוח מערכת טיסה אוטונומית של רחפן שתאפשר לרחפן לטוס בצורה אוטונומית בחדר סגור, ללא GPS, בזמן אמת.
- מערכת שתעבוד באזורים בהם אין כיסוי GPS או תקשורת כלשהי.
- לאפשר גם לרחפן-ננו לטוס בצורה מבוקרת – דבר שלרוב בא לידי ביטוי בעיקר ברחפנים גדולים ויקרים.
- מוצר בטוח לשימוש בסביבת אנשים/בע"ח ובסביבות פנים.

### המוטיבציה:

- תחום פיתוח הרחפנים בכלל, הוא מוקד עניין משמעותי בשנים האחרונות הן במחקר והן בתעשייה לצרכים שונים.
- בעוד קיימים ונפוצים פיתוחים רבים של הטסה ידנית של רחפן ע"י שלט. קיימים הרבה פחות פתרונות ופיתוחים בתחום טיסה אוטונומית לרחפנים, עוד פחות מזה של טיסה אוטונומית במרחב סגור ללא GPS.
- עצם המימוש על רחפן זעיר בעל יכולת לטוס כמעט בכל מקום תפוז תקרת זכוכית עבור פתרונות רבים לצרכים מעשיים ומחקריים בסביבות פנים וסביבות חוץ.

### דוגמאות למימושים אפשריים:

1. איתור ניצולים בבניין שהתמוטט אחרי רעידת אדמה, דבר, אסון טבע או ניווט לצוותי חירום.
2. מיפוי של מערות המסוכנות לכניסת לבני אדם.

### הגישה לפתרון הבעיה :

- השתמשנו ברחפן מזערי השוקל כ-30 גרם עם יחידת IMU (flight controller) מובנית בתוכו, מצלמה קטנה אשר מספקת לאלגוריתם שלנו את קלט הווידאו בזמן אמת מהרחפן ומהווה למעשה את כל המידע הנדרש להבנת המפה.
- ללוח של הרחפן חיברנו את רכיב ה-Flow-Deck על מנת לקבל את הקלטים החשובים לתהליך ההמראה והנחיתה.
- אלגוריתם ORB-SLAM2 (Simultaneous localization and mapping) אלגוריתם קוד פתוח עליו התבססנו ליצירת "GPS" בחדר סגור, כלומר, אנו מקבלים את וקטור המיקום המוצא של ה-SLAM.
- התקשורת בין הרחפן/מצלמה למערכת שעל המחשב מתבססת על משדרי/מקלטי רדיו ולכן היא בלתי תלויה ברשת תקשורת אחרת/GPS.
- תקשורת בו-זמנית בין הרחפן למערכת ולהיפך מאפשרת ניווט בזמן אמת.

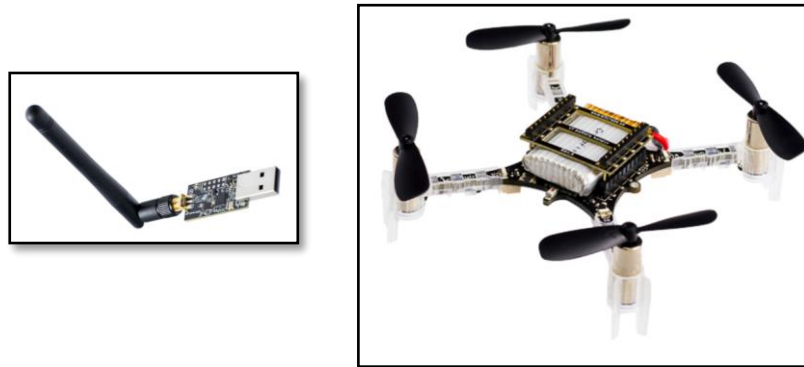
### השוואה עם עבודות אחרות :

- מערכות הניווט של רחפנים הקיימות היום בדרך כלל מסתמכות על שירותי GPS או על תקשורת WIFI, מה שהופך את המערכת שלנו לעצמאית הרבה יותר.
- בחרנו ברחפן קטן וזריז יותר יחסית לרחפנים אוטונומיים הקיימים בשוק, מה שמאפשר לו להיכנס למרחבים יחסית קטנים בצורה הרבה יותר בטוחה, בנוסף, הוא פחות ניתן לגילוי בהשוואה לרחפנים גדולים.

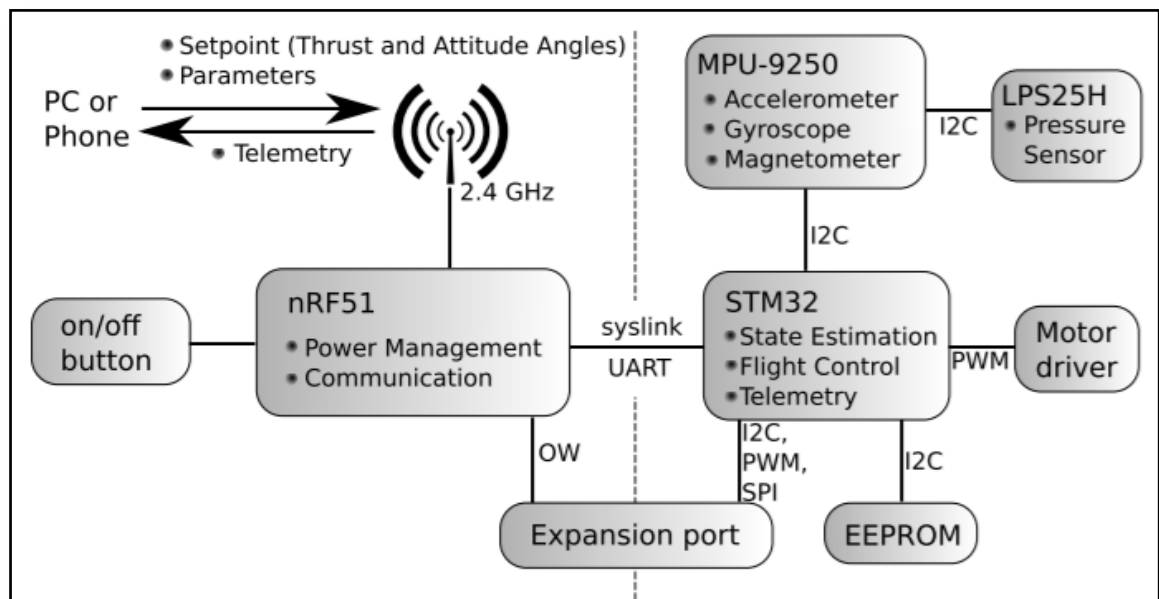
## מרכיבי המערכת

### 1. רחפן BitCraze Crazyflie 2.0:

- רחפן Open-Source, Open-Hardware שנבנה למטרות מחקר, השכלה ופיתוח.
- רחפן זעיר, 92 מ"מ מרחק אלכסוני מנוע-מנוע.
- קל משקל, 29 ג'.
- נשלט מ-PC באמצעות "CrazyRadio PA USB dongle".



איור 2 – רחפן Crazyflie 2.0 (ימין) ו-CrazyRadio PA USB dongle (שמאל)

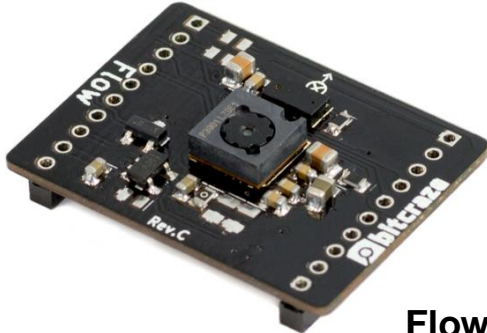


איור 3 – ארכיטקטורת חומרה – רחפן Crazyflie

- ארכיטקטורת החומרה של הרחפן (המוצגת באיור 4), מורכבת מ-2 מיקרו בקרים. מיקרו בקר ראשון STM32 (Cortex M4) הוא הבקר הראשי שאחראי על בקרת הטיסה ומיקרו בקר שני nRF51 (Cortex M0) אחראי על התקשורת האלחוטית, המתבצעת בתדרי 2.4GHz.

## 2. רכיב ה: Flow-Deck v2

המכיל חיישן גובה VL53L0X וחיישן אופטי מסוג PMW3901 למדידת התנועה ביחס לקרקע.



איור 4 – Flow Deck

## 3. המצלמה : CrazyPony FPV Nano-Camera with transmitter

תכונות:

- מצלמת AIO FPV אנלוגית מזערית עם אנטנת שידור רדיו.
- מכילה מודל פליטת תמונות מתוכנן במיוחד לעבוד עם רחפנים קלילים ובמקומות סגורים.
- משדרת עם הספק שידור נמוך
- ביצועים גבוהים וחסיונות לרעש
- משקל המצלמה הכולל פחות מ 5-6 גרם.
- זווית הקליטה/צילום לפחות 100-100 דרגות.
- מתח הכניסה 2-5V, זה הוא המתח שהסוללה המתאימה יכולה לספק.
- איכות טובה, 480X640 ויותר.
- נפח בין mm10X10X10 עד ל mm22X22X22



איור 5 - המצלמה



#### 4. מקלט Eachine:

##### תכונות:

- מקלט עם כפתורי חיפוש תדר ואופציה למצוא תדר השידור הכי טוב , שבמקרה שלנו זה התדר מהמצלמה (תדר 5.8G )
- קטן וקל לשימוש .
- בעל רגישות יציבה יותר ממקלטים אחרים.
- פיגור קטן של 100msec
- אנטנות המקלט: triple feed patch, Eachine



איור 6 – המקלט והאנטנה:

## **תכנון ומימוש**

העבודה על הפרויקט התחלקה למספר שלבים מרכזיים, בנינו טופולוגית עבודה למהלך הפרויקט חלקה תיאורטי וחלקה חומרתי או תוכנתי, חלקה נבנה לפי הוראות מתזות קודמות שקשורות לנושא שלנו, למשל בחירת הציוד, וחלק אחר נבנה לפי הוראות המנחה ומחשבות משותפת, עמדנו בהרבה מהתכנון עד שפגשנו מכשולים תוך כדי הפרויקט ובגללם שנינו חלק מהתכנון.

### **קידמנו את הרעיון חמישה שלבים:**

1. קניית הציוד הפיזי המתאים לדרישות הפרויקט והרכבתו בצורה תקינה (הרחפן, Flow deck, המצלמה, המשדר והמקלט).
2. עבדנו על השגת תקשורת תקינה בין הרחפן, המצלמה והמחשב שמריץ את התוכנה ברקע.
3. שלב התוכנה של הפרויקט: בשלב הזה סנכרנו את המידע שמקבלים אותו מהחיישן הצמוד לגוף הרחפן (IMU) והתמונות שמקבלים אותם מהמצלמה שהצמדנו אותה על הלוח (גוף הרחפן).  
המידע הזה מגיע למחשב דרך המקלט של המצלמה והמקלט של הרחפן, ובעזרת תוכנת ROS מכניסים אותו להיות הקלט של אלגוריתם ה-ORB SLAM2 שמעבד את הנתונים ועושה את המיפוי לאזור סביב הרחפן ובונה מפה למיקום ובעזרתו גם קובעים את הקואורדינטות של המיקום הנוכחי של המערכת בתוך המפה, הפלט של אלגוריתם זה הוא הווקטור  $(x,y,z)$  שמתאר את מיקום הרחפן בכל רגע בזמן אמת.  
בצעד האחרון משתמשים בפלט האלגוריתם כדי לשלוח אותו חזרה לרחפן, ובעזרת אינטגרציה של בקרת הרחפן נוכל לשלוח פקודת טיסה למקום מיסויים.
4. כתיבת ה Switch בין ה Flow Deck ל Slam: את שלב זה הוספנו לאחר שראינו שהרחפן מתקשה בהמראה בעזרת ה SLAM2 בלבד, לנתונים שלו לוקח זמן להתאפס ולכן הבנו שרכיב ה Flow Deck הוא זה שצריך לנהל את שלב ההמראה.
5. נאלצנו להוסיף שינויי קוד באלגוריתם שיאפשרו לנו לעבוד בשתי הדרכים, בשלב ההמראה הרחפן יקבל את וקטור המיקום מה Flow Deck ובלאחר מכן הרחפן יעבור לקבל נתונים אלו המחשב עליו רץ אלגוריתם ה SLAM2.

## **בחירת החומרה והרכבתה**

בחירת החלקים ומרכבים החומרתיים, זה הוא השלב הראשון ליצירת ננו רחפן אוטונומי, אלו הם המרכבים שהשתמשנו בהן לבניית המערכת הפיזית

### **1) הרחפן Flow deck v2+**

לוח הפיתוח הטס CrazyFlie 2.0 ממשפחת Bitcraze הוא המרכים המרכזי למערכת, open source board קל לשימוש ולפיתוח מערכות תוכנה עליו, קל במשקל (~20 גרם) בעל תקשורת טובה ונוחה עם מחשב (מערכת הפעלה LINUX).

על הרחפן חיברנו את רכיב ה-Flow-Deck, רכיב שמכיל בתוכו חיישן גובה מסוג VL53L0x שאחראי על מדידת הגובה מהקרקע (בשעת ההמראה והנחיתה בלבד), וחיישן אופטי מסוג PMW3901 שמודד תנועה יחסית לרצפה (שינוי בדפוסים על הקרקע)

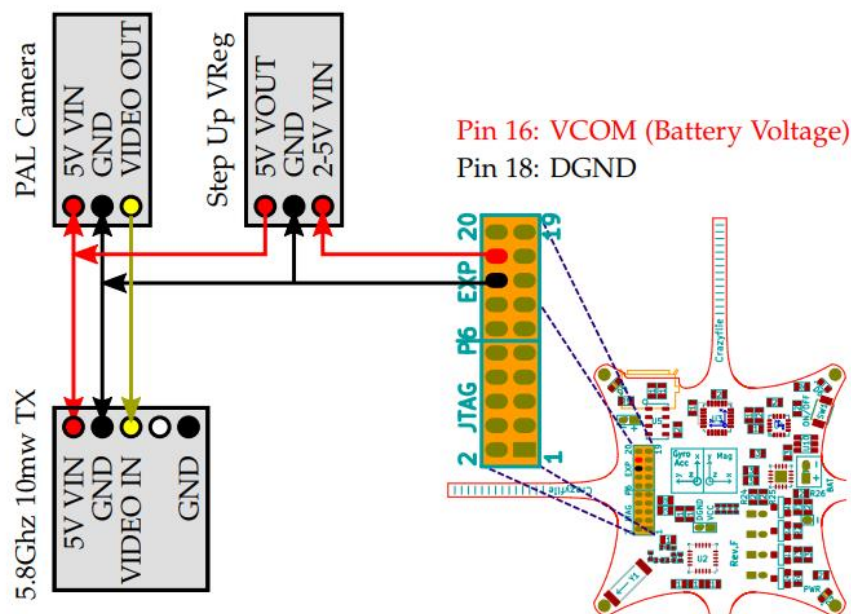
### **2) המצלמה**

בחיפוש הראשוני מלאנו את הרשימה שלנו במספר מצלמות שכולן עומדות ברוב הדרישות עם חקירת חוות הדעת של משתמשים וחוקרים שבצעו פרויקטים דומים וכן הוסיפו מצלמה לננו רחפנים, החלטנו להמשיך את יצירת המערכת שלנו עם CrazyPony FPV Nano-Camera שהזכרנו לפני, כך ששימוש במצלמה זו חוסך את החיבור של המשדר החיצוני ובכלל התאמת משדר לסוג המלצה המיסויים הזה, היא עובדת בתדרי 5.8GHz.

### **3) המקלט**

הרכיב האחרון הוא המקלט, בחרנו במקלט שעובד באותו תדר של המשדר המצורף למצלמה, וגם בחרנו בו להיות מתאים למערכת ההפעלה שנעבוד בה (LINUX), הוא בעל חיבור USB המתאים למחשבים, במשדר זה השתמשו גם בפרויקטים אחרים עם אותה המצלמה ומערכת ההפעלה שנתקין במחשבים שלנו. המקלט עובד בתדר 5.8GHz והבחירה בו נעשתה לאחר מספר ניסיונות של מקלטים אחרים, בחרנו בו מכיוון שהוא בעל הרזולוציה הכי טובה ביותר מבין המקלטים שבדקנו.

**חיבור המצלמה לרחפן נעשה בצורה הבאה:** לאחר קבלת כל הציוד, חיברנו את המצלמה לגוף הרחפן בצורה הבאה, כך שספקנו לה מתח של 5 וולט מהסוללה של הרחפן עצמו והדבקנו אותה



איור 7 – חיבור לגוף הרחפן



איור 8 – הרחפן עם המצלמה

## שלבי הרצת התוכנה:

### כמה הגדרות ומשפטים קשורות להמשך העבודה לפני שלבי הרצה:

- המידע על המיקום של הרחפן בערוץ מידע (המכונה topic בסביבת ROS).
  - עבור כל רכיב טס נפתח תהליך בשם Publisher שתפקידו הוא להאזין לחבילות מיקום של הרחפן שלו, להתאים את פורמט ההודעות שהתקבלו לפורמט הודעות המתאים לשרת הרחפנים (עניין טכני שאין טעם להרחיב עליו), ולבצע איפוס פרמטרים ראשוני של המשערך (Kalman) שרץ על גבי הרחפן.
  - בנוסף לתהליכים שתוארו, תהליך נוסף שנקרא Crazyfly\_server הוא שפותח עבור הרחפן ערוץ מידע לנתוני מיקום ולשירותים שונים (המראה, נחיתה וכו'). תהליך זה אחראי לבסוף להפצת החבילות לרחפן באמצעות מכשיר הרדיו.
  - קובץ Launch – קובץ קונפיגורציה ראשי של כלל הפרמטרים והתהליכים שייווצרו עם הארגומנטים המתאימים.
- (1) ROS:

**ROS**::: היא אוסף של מסגרות עבודה לפיתוח תוכנה המיועדת לפיתוח פרויקטים Open-Source בתחום הרחפנים.

ROS מייצרת אבסטרקציה בתהליך הפיתוח, ויכולת נוחה ויעילה לאינטגרציה בין מודולים שונים, שכל אחד נתמך ע"י חבילת ROS מתאימה. השתמשנו במיוחד בחבילה המתאימה לרחפן שלנו, crazyfly\_ros – חבילת ROS שנכתבה ב-UCLA, נוצרה לראשונה ב-2014.

תכננו לעבוד עם SLAM שנקרא VINS-MONO, אלגוריתם לעיבוד תמונה שבעזרת התמונות שקולטים מהמצלמה ביחד עם הנתונים שמחלצים מחיישן IMU מגוף הרחפן הוא בונה מפה לאזור שמצלמים וקובע את המיקום הנוכחי של המערכת, אבל באלגוריתם הזה נתקלנו בבעיה: הפלט שלו היה בתדירות קטנה 10Hz - מה שלא מתאים לקוד הבקרה שצורב על החומרה בלוח (הרחפן), לכן בחרנו להשתמש באלגוריתם אחר שנקרא ORB-SLAM2 שעושה אותה עבודה ללא שימוש בIMU (עבודה עם הIMU מבטיחה מיפוי מדויק יותר).

לשני האלגוריתמים השתמשנו באותו קלט של המצלמה (הlaunch שהוגדר לפני), ולשניהם גם היה אותו פלט, וקטור מיקום נוכחי.

בעזרת אלגוריתם הORB-SLAM2 השתמשנו בעוד חבילה לקליטה המידע מהIMU של הרחפן, באמצעותה ובעזרת קובץ launch (בתמונה הבאה) התאפשרה הקריאה של הפלט מהSLAM2.



## 2. הרצת ORB SLAM2

אלגוריתם SLAM אחר שמבצע אותה פעולה ללא שימוש בחיישן הIMU, בחרנו להשתמש בו אחרי שנתקלנו עם בעיית שידור של מיקום מהVINSMONO (האלגוריתם המקורי בו בחרנו לעבוד) כך שהוא משדר הנתונים בתדירות של 10hz מה שגורם לתנועה לא יציבה בטיסה של הרחפן, לעומת ה ORB-SLAM2 שמשדר בתדר של 30hz, יש להתקין את הgithub של הפרויקט ולהריץ את הlaunch של המצלמה ושל הרחפן ואז להריץ את הפקודה של הORB

```
$ roslaunch orb_slam2_wrapper orb_slam2_CF.launch
```

```
Open ▾ 1 Save
<!-- ARGS -->
<arg name="camera_frame" default="camera_frame"/>
<arg name="camera_optical_frame" default="camera_optical"/>
<arg name="pose_topic" default="orb_slam2_pose"/>
<arg name="camera_topic" default="/camera/image_raw"/>
<arg name="path_vocabulary" default="/lab/3rdparty/ORB_SLAM2/Vocabulary/ORBvoc.txt"/>
<arg name="path_settings" default="$(find orb_slam2_wrapper)/config/CF3.yaml"/>
<arg name="use_viewer" default="true"/>

<arg name="orb_var_x" default="0.001"/>
<arg name="orb_var_y" default="0.001"/>
<arg name="orb_var_z" default="0.001"/>
<arg name="orb_var_roll" default="0.001"/>
<arg name="orb_var_pitch" default="0.001"/>
<arg name="orb_var_yaw" default="0.03"/>

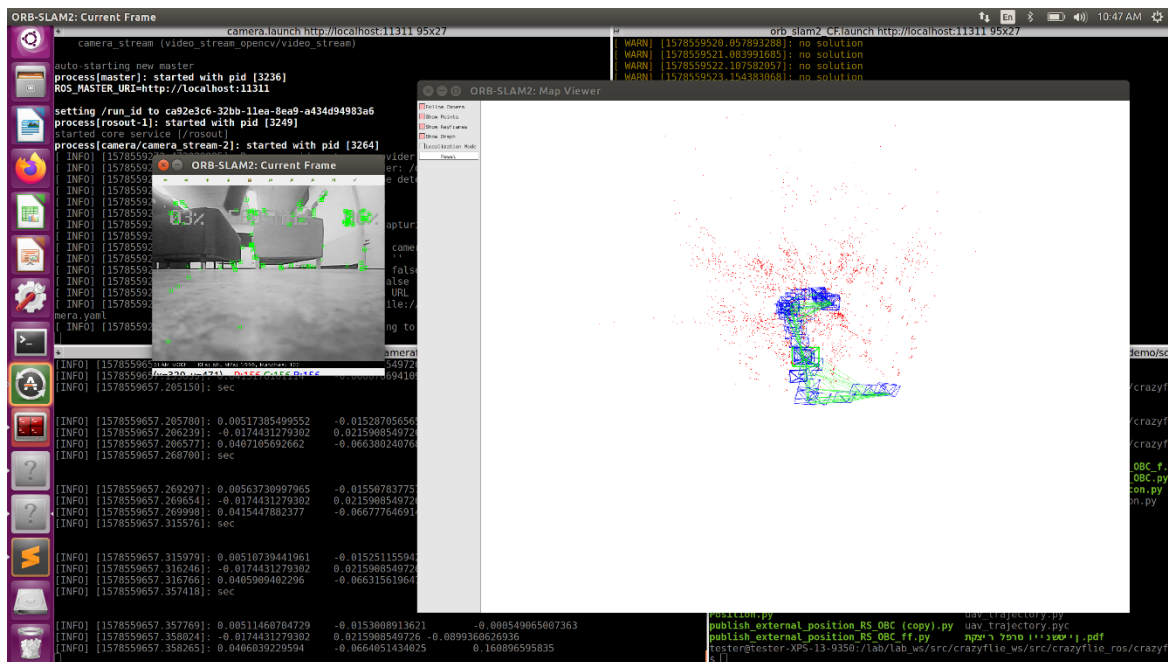
<!-- ORB SLAM 2 -->
<param name="camera_frame" value="$(arg camera_frame)"/>
<param name="camera_optical_frame" value="$(arg camera_optical_frame)"/>
<param name="pose_topic" value="$(arg pose_topic)"/>
<param name="camera_topic" value="$(arg camera_topic)"/>
<param name="path_vocabulary" value="$(arg path_vocabulary)"/>
<param name="path_settings" value="$(arg path_settings)"/>
<param name="use_viewer" value="$(arg use_viewer)"/>

<param name="orb_var_x" value="$(arg orb_var_x)"/>
<param name="orb_var_y" value="$(arg orb_var_y)"/>
<param name="orb_var_z" value="$(arg orb_var_z)"/>
<param name="orb_var_roll" value="$(arg orb_var_roll)"/>
<param name="orb_var_pitch" value="$(arg orb_var_pitch)"/>
<param name="orb_var_yaw" value="$(arg orb_var_yaw)"/>

<node pkg="orb_slam2_wrapper" type="orb_slam2_mono_node" name="orb_slam2_mono_node" output="screen"/>
</launch>
```

Plain Text ▾ Tab Width: 8 ▾ Ln 8, Col 76 ▾ INS

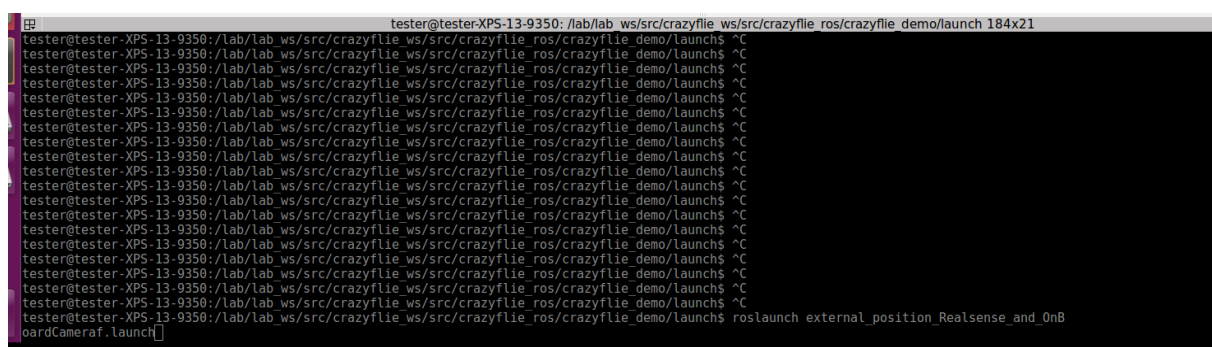
בתמונה הבאה ניתן לראות את האלגוריתם רץ בצד ימין עם הנקודות שרואה הרחפן בזמן אמת ואת מיקומו הנוכחי, בצד שמאל רואים את התמונה ממצלמת הרחפן ואת הפריים שנכנס כקלט לאלגוריתם ה-ORB-SLAM2.



### 3. הרצת ORB SLAM2 Launch:

ע"י פקודת ה-`roslaunch` עם שם הקובץ `external_position_Realsense_and_OnBoardCamera.launch` מריצים

זזה על מנת לקבל את המוצא של SLAM2 שהוא הקורדינטות X Y Z הנוכחיים של המיקום.



הפקודה להרצת הקובץ:

`---$ roslaunch external_position_Realsense_and_OnBoardCamera.launch`



## NOTE:

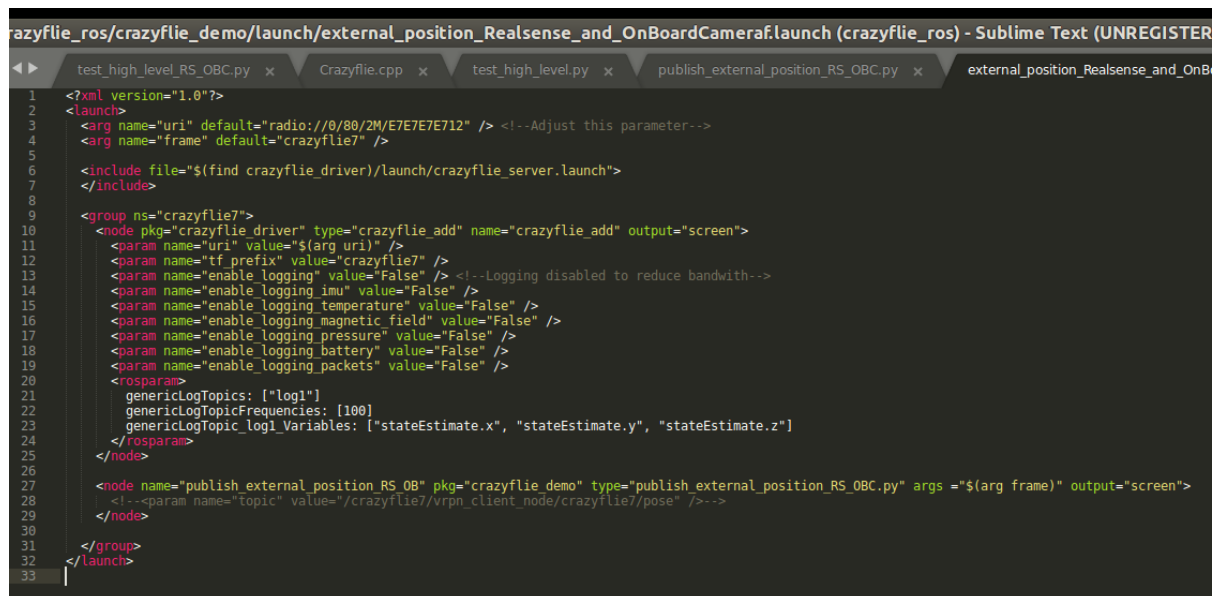
You must do calibration to your slam after every execution, this means you must check if the output of this file "external\_position\_Realsense\_and\_onBoardCamera" is the same as real life coordination, you need to check that manually like you lift your drone by your hand to 0.2m in z direction(height) and see if the output of this file gives you 0.2m in z direction .

To handle this calibration and to change your file output to be as the real life's coordination, you must go to :

- 1) Publish\_external\_position\_RS\_OBC.py file (see the second attached photo down)
- 2) Then go inside this file to where your coordination delta found, which for us it's in this format:

```
msg.point.x=(x-x0)*(caliberation's parameter)
msg.point.y=(y-y0)*(caliberation's parameter)
msg.point.z=(z-z0)*(caliberation's parameter)
```

So we change the parameter (caliberation's parameter) accordingly to different percent between coordination in real life and coordination from slam, so after adapting this parameter, we run again the file external\_position\_Realsense\_and\_onBoardCamera.launch and we **must see** that the output of this file (the coordination) is like real life coordination.



```
<?xml version="1.0"?>
<launch>
  <arg name="uri" default="radio://0/80/2M/E7E7E7E712" /> <!--Adjust this parameter-->
  <arg name="frame" default="crazyflie/" />

  <include file="$(find crazyflie_driver)/launch/crazyflie_server.launch">
  </include>

  <group ns="crazyflie7">
    <node pkg="crazyflie_driver" type="crazyflie_add" name="crazyflie_add" output="screen">
      <param name="uri" value="$(arg uri)" />
      <param name="tf_prefix" value="crazyflie7" />
      <param name="enable_logging" value="False" /> <!--Logging disabled to reduce bandwidth-->
      <param name="enable_logging_imu" value="False" />
      <param name="enable_logging_temperature" value="False" />
      <param name="enable_logging_magnetic_field" value="False" />
      <param name="enable_logging_pressure" value="False" />
      <param name="enable_logging_battery" value="False" />
      <param name="enable_logging_packets" value="False" />
      <rosparam>
        genericLogTopics: ["log1"]
        genericLogTopicFrequencies: [100]
        genericLogTopicLog1Variables: ["stateEstimate.x", "stateEstimate.y", "stateEstimate.z"]
      </rosparam>
    </node>

    <node name="publish_external_position_RS_OBC" pkg="crazyflie_demo" type="publish_external_position_RS_OBC.py" args="$(arg frame)" output="screen">
      <!--<param name="Topic" value="/crazyflie7/vrpn_client_node/crazyflie7/pose" />-->
    </node>
  </group>
</launch>
```

יש להריץ את שתי הפקודות של המצלמה ושל ה-IMU (של הרחפן)-כפי שהוסבר בשלבים הראשונים- לאחר מכן להריץ את ה launch לעיל, מתוך ההרצה הוא קורא לקובץ **publish\_external\_position\_RS\_OBC.py**

```

azyruie_ros/crazyflie_demo/scripts/publish_external_position_RS_OBC.py (crazyflie_ros) - Sublime Text (UNREGISTERED)
test_high_level_RS_OBC.py x Crazyflie.cpp x test_high_level.py x publish_external_position_RS_OBC.py x external_position_RealSense_and_OnBoardCamera.launch x Hover.py x Crazyradio.cpp x x

1 #!/usr/bin/env python
2 import rospy
3 import tf
4 from nav_msgs.msg import Odometry
5 from geometry_msgs.msg import PointStamped, TransformStamped, PoseStamped, PoseWithCovarianceStamped # PoseStamped added to support vrpn_client
6 from crazyflie_driver.srv import UpdateParams
7 import time
8 from std_msgs.msg import String
9
10 X0 = 0
11 Y0 = 0
12 Z0 = 0
13
14 out = False
15
16
17 firstTransform = True
18
19
20 def onNewTransform(pose):
21     global msg
22     global pub
23     global firstTransform
24
25     global X0
26     global Y0
27     global Z0
28
29     rospy.loginfo("sec\n\n")
30     if firstTransform:
31         rospy.set_param("kalman/initialX", 0) #pose.pose.position.x # Change pose according to your topic
32         rospy.set_param("kalman/initialY", 0) #pose.pose.position.y
33         rospy.set_param("kalman/initialZ", 0) #pose.pose.position.z
34         update_params(["kalman/initialX", "kalman/initialY", "kalman/initialZ"])
35         rospy.set_param("kalman/resetEstimation", 1)
36         update_params(["kalman/resetEstimation"])
37         X0 = pose.pose.pose.position.x;
38         Y0 = pose.pose.pose.position.y;
39         Z0 = pose.pose.pose.position.z;
40         firstTransform = False
41     else:
42         msg.header.frame_id = pose.header.frame_id # Change pose according to your topic
43         msg.header.stamp = pose.header.stamp
44         msg.header.seq += 1
45
46
47         X2 = pose.pose.pose.position.x;
48         Y2 = pose.pose.pose.position.y;
49         Z2 = pose.pose.pose.position.z;
50
51         msg.point.x = (X2 - X0)*1.8
52         msg.point.y = (Y2 - Y0)*1.8
53         msg.point.z = (Z2 - Z0)*1.8
54
55         #rospy.loginfo("x=" + str(pose.pose.pose.position.x)[:6] + "... y=" + str(pose.pose.pose.position.y)[:6] + "... z=" + str(pose.pose.pose.position.z)[:6])
56         #rospy.loginfo("x=" + str(X0)[:6] + "... y=" + str(Y0)[:6] + "... z=" + str(Z0)[:6])
57         #rospy.loginfo("x sent=" + str(msg.point.x)[:6] + "... y sent=" + str(msg.point.y)[:6] + "... z sent=" + str(msg.point.z)[:6])
58         rospy.loginfo(str(pose.pose.pose.position.x) + " " + str(pose.pose.pose.position.y) + " " + str(pose.pose.pose.position.z))
59         rospy.loginfo(str(X0) + " " + str(Y0) + " " + str(Z0))
60         rospy.loginfo(str(msg.point.x) + " " + str(msg.point.y) + " " + str(msg.point.z))
61         pub.publish(msg)
62
63
64
65
66 def callback(data):
67     rospy.loginfo("enter ..HON..")
68     global out
69     if data.data == "true":
70         out = True
71         return
72
73
74 if __name__ == '__main__':
75     rospy.init_node('publish_external_position_RS_OBC', anonymous=True)
76     #topic = rospy.get_param("~topic", "/crazyflie7/vrpn_client_node/crazyflie7/pose")
77
78     rospy.wait_for_service('update_params')
79     rospy.loginfo("found update_params service")
80     update_params = rospy.ServiceProxy('update_params', UpdateParams)
81
82     msg = PointStamped()
83     msg.header.seq = 0
84     msg.header.stamp = rospy.Time.now()
85
86     pub = rospy.Publisher("external_position", PointStamped, queue_size=1)
87     '''rospy.loginfo("enter .....")
88     rospy.Subscriber("/chatter", String, callback)
89     while (out == False):
90         pass'''
91     rospy.Subscriber("/orb_slam2_pose", PoseWithCovarianceStamped, onNewTransform) # Change this according to your topic
92     rospy.spin()
93

```

```

test_high_level_RS_OBC.py x Crazyflie.cpp x test_high_level.py x publish_external_position_RS_OBC.py x ex
64
65
66 def callback(data):
67     rospy.loginfo("enter ..HON..")
68     global out
69     if data.data == "true":
70         out = True
71         return
72
73
74 if __name__ == '__main__':
75     rospy.init_node('publish_external_position_RS_OBC', anonymous=True)
76     #topic = rospy.get_param("~topic", "/crazyflie7/vrpn_client_node/crazyflie7/pose")
77
78     rospy.wait_for_service('update_params')
79     rospy.loginfo("found update_params service")
80     update_params = rospy.ServiceProxy('update_params', UpdateParams)
81
82     msg = PointStamped()
83     msg.header.seq = 0
84     msg.header.stamp = rospy.Time.now()
85
86     pub = rospy.Publisher("external_position", PointStamped, queue_size=1)
87     '''rospy.loginfo("enter .....")
88     rospy.Subscriber("/chatter", String, callback)
89     while (out == False):
90         pass'''
91     rospy.Subscriber("/orb_slam2_pose", PoseWithCovarianceStamped, onNewTransform) # Change this according to your topic
92     rospy.spin()
93

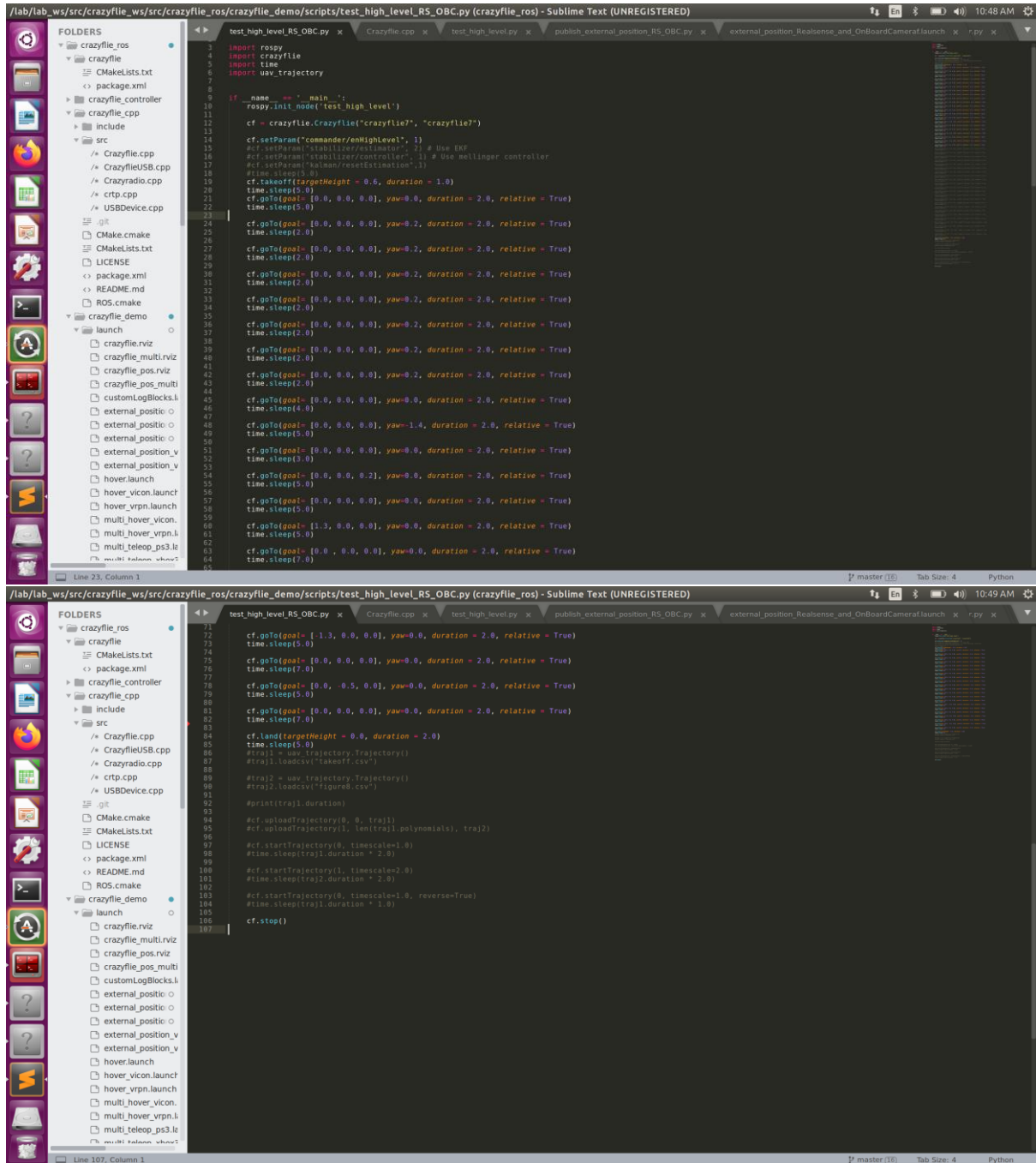
```

ה ORB- DATA מה- SLAM2 SUBSCRIBER כפי שנראה בתמונה (למטה) מקבל את ה DATA מה- ORB- SLAM2 וזו TOPIC אחר לוקח את DATA זו ומשדר אותה לרחפן, SERVER (זה בעצם תפקיד ה publisher)

ואז על הCHIP של הרחפן מתקבל ה OUTPUT של ה SLAM2 (XYZ) ונכנס לKALMAN על מנת לשערך את המידע ולעבירו אל הרחפן.

#### 4. הרצת קוד פיתון הסופי הוא :

ע"י פקודת `python test_high_level_RS_OBC.py` מריצים את הקובץ כך שיהיה יקבל את ה SETPOINTS מקובץ פייתון זה, וכך ידע לאן לטוס על פי ה DATA של SLAM2.



```
1 import rospy
2 import crazyflie
3 import time
4 import uav_trajectory
5
6 if __name__ == '__main__':
7     rospy.init_node('test_high_level')
8
9     cf = crazyflie.Crazyflie("crazyflie7", "crazyflie7")
10
11     cf.setParam("command/enhHighLevel", 1)
12     # cf.setParam("stabilizer/estimator", 2) # Use EKF
13     # cf.setParam("stabilizer/controller", 1) # Use mclinger controller
14     # cf.setParam("balancer/resetEstimation", 1)
15     #time.sleep(5.0)
16     cf.takeoff(targetHeight = 0.6, duration = 1.0)
17     time.sleep(5.0)
18     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.0, duration = 2.0, relative = True)
19     time.sleep(5.0)
20
21     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.2, duration = 2.0, relative = True)
22     time.sleep(2.0)
23
24     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.2, duration = 2.0, relative = True)
25     time.sleep(2.0)
26
27     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.2, duration = 2.0, relative = True)
28     time.sleep(2.0)
29
30     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.2, duration = 2.0, relative = True)
31     time.sleep(2.0)
32
33     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.2, duration = 2.0, relative = True)
34     time.sleep(2.0)
35
36     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.2, duration = 2.0, relative = True)
37     time.sleep(2.0)
38
39     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.2, duration = 2.0, relative = True)
40     time.sleep(2.0)
41
42     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.2, duration = 2.0, relative = True)
43     time.sleep(2.0)
44
45     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.0, duration = 2.0, relative = True)
46     time.sleep(4.0)
47
48     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=-1.4, duration = 2.0, relative = True)
49     time.sleep(3.0)
50
51     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.0, duration = 2.0, relative = True)
52     time.sleep(3.0)
53
54     cf.goTo(goal=[0.0, 0.0, 0.2], yaw=0.0, duration = 2.0, relative = True)
55     time.sleep(5.0)
56
57     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.0, duration = 2.0, relative = True)
58     time.sleep(5.0)
59
60     cf.goTo(goal=[1.3, 0.0, 0.0], yaw=0.0, duration = 2.0, relative = True)
61     time.sleep(5.0)
62
63     cf.goTo(goal=[0.0, 0.0, 0.0], yaw=0.0, duration = 2.0, relative = True)
64     time.sleep(7.0)
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

## אינטגרציה עם בקרת הרחפן

משתמשים ב ROSTOPIC שיוצא מהאלגוריתם (ORB-SLAM2) המכיל את וקטור המיקום, מחלצים ממנו את הנתונים בעזרת הקוד הבא:

```
publish_external_position_RS_OBC.py
```

לו קוראים תוך כדי הפעלת launch של הרחפן:

```
external_position_Realsense_and_OnBoardCamera.launch
```

לאחר מכן בעזרת קוד הבקרה שצורב ללוח (firmware) מצליח הרחפן לטוס מהמקום הנוכחי ליעד המבוקש ע"י קובץ הפייתון: `test_high_level_RS_OBC.py` השולח פקודות הנעה לרחפן עם המסלול הנדרש בעזרת הפלט של ה-SLAM.

## לסיכום:

1) הקוד הראשון מפעיל את המצלמה ונותן לנו ACCESS למצלמה ול FRAMES שהיא מצלמת.

2) הקוד השני מריץ את ה SLAM ומקבל כקלט את ה FRAMES מהמצלמה.

3) הקוד השלישי הוא בעצם מאפשר לקבלת המדע על ה מיקום מה SLAM2 (X Y Z) ושולח אותם לרחפן.

4) הקוד הרביעי שולח פקודות הנעה לרחפן עם המסלול הנדרש בעזרת הפלט של ה-SLAM2.

## קוד FIRMWARE ל CHIP הראשי של הרחפן עבור ה SWITCH:

הורדנו FIRMWARE שהוא OPENSOURCE מאתר של BITCRAZE שהוא:

<https://github.com/bitcraze/crazyflie-firmware>

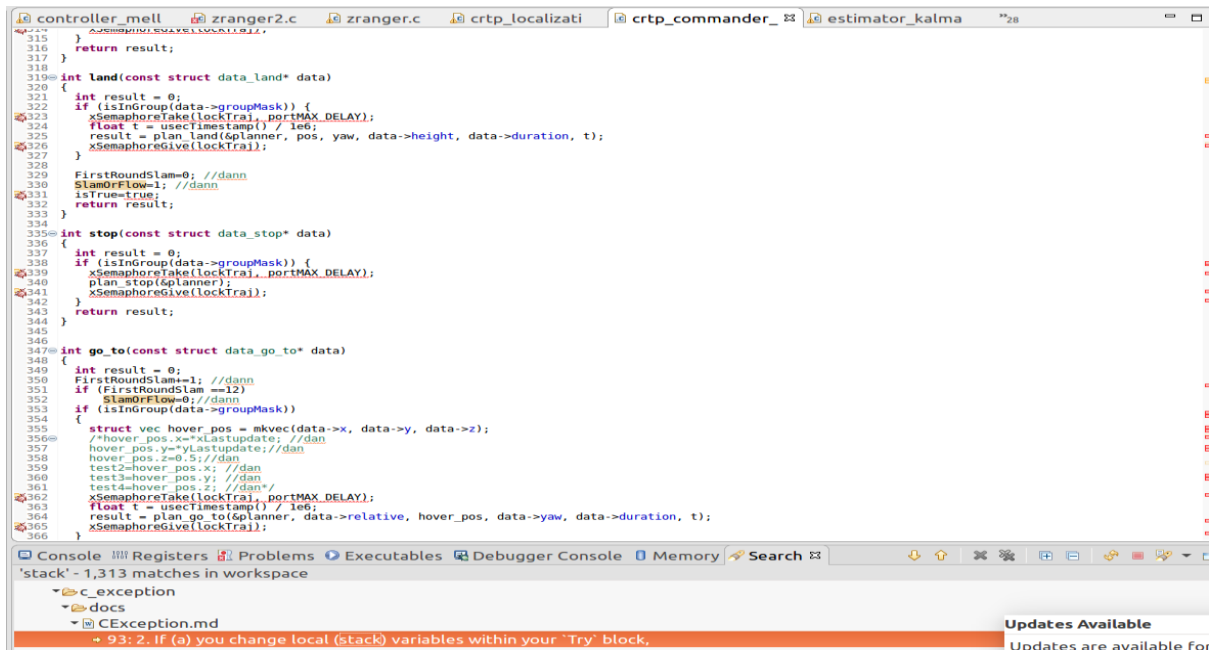
ע"י פקודת `git clone` ב LINUX הורדנו את "הפירמוויר" של הרחפן ואחר מכן עשינו שינויים בקוד על מנת לעשות את ה SWITCH בין data של ה-SLAM2 ל-FLOWDECK.

\*\*השינויים בקוד מסומנים ע"י המילה "dann" כהערה.

## הסבר כללי לגבי ה SWITCH שכתבנו:

כששולחים פקודת GOTO מ PYTHON script הפרמטרים של GOTO והקריאה לה נעשית בקוד ה FIRMWARE בתוך פונקצית `int go_to`. בתוכה ממשנים את ערכו של ה FLAG ל 1 שזה מפעיל את ה SLAM ומבטל את קבלת מדע מ FLOWDECK.

בנוסף, כששולחים פקודת land מ PYTHON script הפרמטרים של land והקריאה לה נעשית בקוד ה FIRMWARE בתוך פונקצית `int land` בה משנים בחזרה את ערך ה FLAG ל 0 לצורך הפעלת ה FLOWDECK בנחיתה, בנוסף, פעולה זו מפסיקה את קבלת המידע מה SLAM.



```
controller_mell  zranger2.c  zranger.c  crtp_localizati  crtp_commander_  estimator_kalma  "28
315 }
316 return result;
317 }
318
319 int land(const struct data_land* data)
320 {
321     int result = 0;
322     if (isInGroup(data->groupMask)) {
323         xSemaphoreTake(lockTraj, portMAX_DELAY);
324         float t = usecTimestamp() / 1e6;
325         result = plan_land(&planner, pos, yaw, data->height, data->duration, t);
326         xSemaphoreGive(lockTraj);
327     }
328     FirstRoundSlam=0; //dann
329     SlamOrFlow=1; //dann
330     isTrue=true;
331     return result;
332 }
333
334 int stop(const struct data_stop* data)
335 {
336     int result = 0;
337     if (isInGroup(data->groupMask)) {
338         xSemaphoreTake(lockTraj, portMAX_DELAY);
339         plan_stop(&planner);
340         xSemaphoreGive(lockTraj);
341     }
342     return result;
343 }
344
345
346 int go_to(const struct data_go_to* data)
347 {
348     int result = 0;
349     FirstRoundSlam=1; //dann
350     if (FirstRoundSlam == 12)
351         SlamOrFlow=0; //dann
352     if (isInGroup(data->groupMask))
353     {
354         struct vec hover_pos = mvec(data->x, data->y, data->z);
355         /*hover_pos.x=xLastupdate; //dan
356         hover_pos.y=yLastupdate; //dan
357         hover_pos.z=0.5; //dan
358         test2=hover_pos.x; //dan
359         test3=hover_pos.y; //dan
360         test4=hover_pos.z; //dan*/
361         xSemaphoreTake(lockTraj, portMAX_DELAY);
362         float t = usecTimestamp() / 1e6;
363         result = plan_go_to(&planner, data->relative, hover_pos, data->yaw, data->duration, t);
364         xSemaphoreGive(lockTraj);
365     }
366 }
```

Console 1,313 matches in workspace

- c\_exception

- docs

- CException.md

- 93:2. If (a) you change local (stack) variables within your 'Try' block,

Updates Available

Updates are available for

כשנכנסים לפונקציית GOTO מתחילים קבלת DATA מה SLAM2 ואזי בפירמוור  
מקבל את DATA מה SLAM2 באמצעות פונקציית extpositionHandler שהיא  
מטפלת ב DATA של SLAM2

אנחנו גם עשינו שינוי שם על מנת שבזמן שעושים את ה SWITCH חייב להיות רציף -  
רגע לפני ה SWITCH כלומר X,Y,Z אחרונים שהתקבלו מ FLOWDECK עם אחרי  
רגע אחרי ה SWITCH.

כלומר X Y Z הראשונים שהתקבלו מ SLAM2 . –עשינו תיקון מידע על הרגע שלפני  
ה-SWITCH עם רגע אחרי ה-SWITCH – זאת על מנת שהרחפן לא יאבד את  
המסלול שלו.

```
controller_mell  zranger2.c  zranger.c  crtp_localizati  crtp_commander_  estimator_kalma  28
130 case EXT_POSITION_PACKED:
131     extPositionPackedHandler(pk);
132     break;
133 default:
134     break;
135 }
136 }
137
138
139 static void extPositionHandler(CRTPPacket* pk)
140 {
141     if (!SlamOrFlow) //dann
142     {
143         static float x0,y0; //dann
144         const struct CrtpExtPosition* data = (const struct CrtpExtPosition*)pk->data;
145         if (FirstRoundSlam == 12 && !isTrue)
146         {
147             xOffset=(xLastupdate-data->x); //dann
148             x0=xOffset;
149             yOffset=(yLastupdate-data->y); //dann
150             y0=yOffset;
151             isTrue=false;
152         }
153     }
154     ext_pos.x = data->x+x0; //dann
155     ext_pos.y = data->y+y0; //dann
156     ext_pos.z = data->z;
157     ext_pos.stdev = extPosStdDev; //dann
158     estimatorEnqueuePosition(&ext_pos);
159     tickOfLastPacket = xTaskGetTickCount();
160 }
161
162
163 static void genericLocHandle(CRTPPacket* pk)
164 {
165     uint8_t type = pk->data[0];
166     if (pk->size < 1) return;
167
168     if (type == LPS_SHORT_LPP_PACKET && pk->size >= 2) {
169         bool success = lpsSendLppShort(pk->data[1], &pk->data[2], pk->size-2);
170
171         pk->port = CRTP_PORT_LOCALIZATION;
172         pk->channel = GENERIC_TYPE;
173         pk->size = 3;
174         pk->data[2] = success?1:0;
175         crtpSendPacket(pk);
176     } else if (type == EMERGENCY_STOP) {
177         stabilizerSetEmergencyStop();
178     } else if (type == EMERGENCY_STOP_WATCHDOG) {
179         stabilizerSetEmergencyStopTimeout(DEFAULT_EMERGENCY_STOP_TIMEOUT);
180     } else if (type == EXT_POSE) {
```

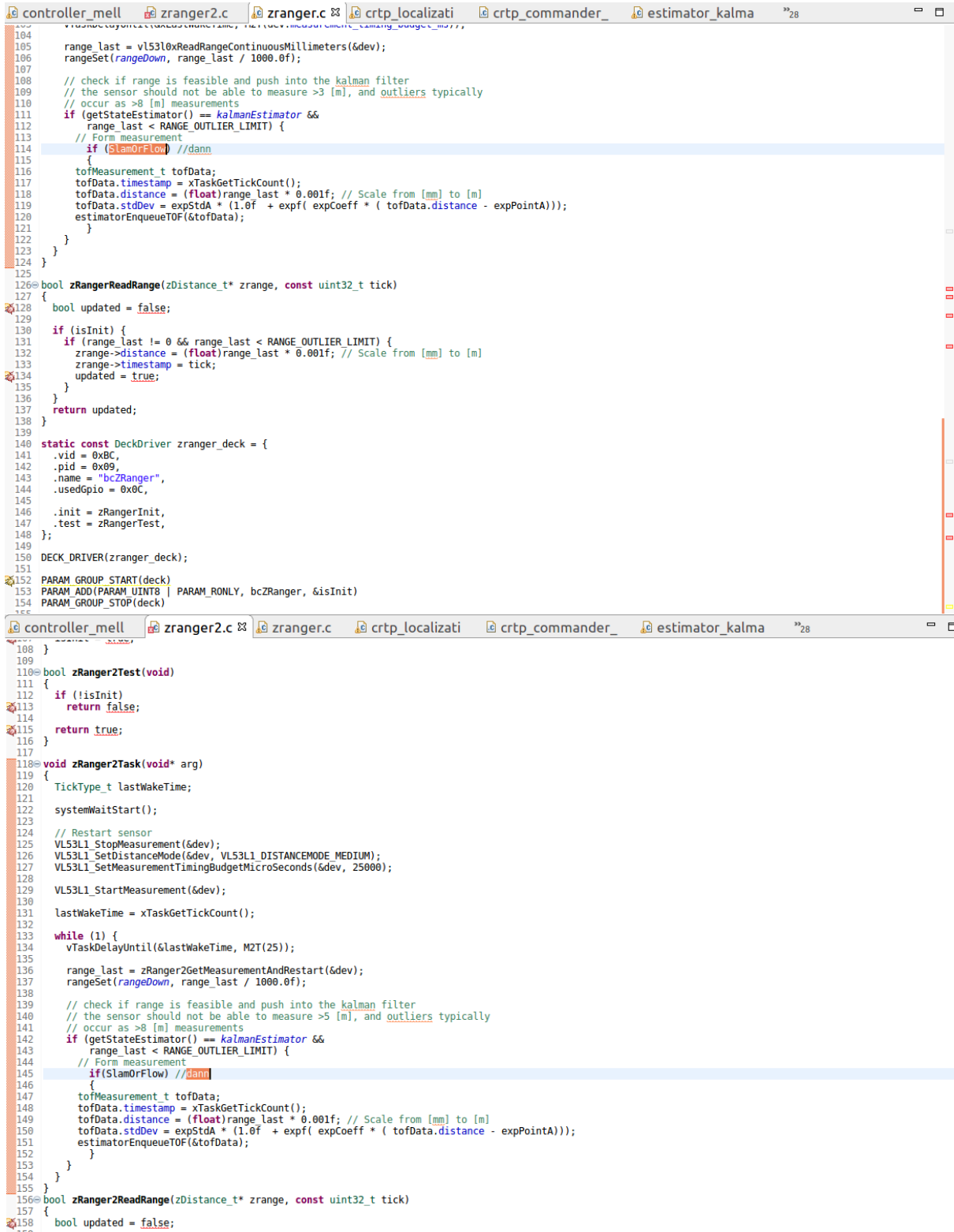
בתמונה מטה כתבנו תנאי נוסף בתוך ה WHILE על מנת לא לקבל מדע בזמן שמקבלים מידע מ SLAM2 כלומר  $SlamOrFlow = 0$ . תפקידו של תנאי זה הוא שלא תהיה התערבות בו זמנית בין מדע של FLOWDECK לבין מדע של SLAM.

```

controller_mell  zranger2.c  zranger.c  crtp_localizati  crtp_commander_  estimator_kalma  28
384     doneUpdate = true;
385 }
386
387 poseMeasurement t pose;
388 while (stateEstimatorHasPoseMeasurement(&pose))
389 {
390     kalmanCoreUpdateWithPose(&scoreData, &pose);
391     doneUpdate = true;
392 }
393
394 tdoaMeasurement t tdoa;
395 while (stateEstimatorHasTDOPacket(&tdoa))
396 {
397     kalmanCoreUpdateWithTDOPacket(&scoreData, &tdoa);
398     doneUpdate = true;
399 }
400
401 flowMeasurement t flow;
402 while (stateEstimatorHasFlowPacket(&flow) && SlamOrFlow) //dann
403 {
404     kalmanCoreUpdateWithFlow(&scoreData, &flow, sensors); // dann
405     xLastUpdate=state->position.x;//dann
406     yLastUpdate=state->position.y;//dann
407     doneUpdate = true;
408 }
409
410 /**
411  * If an update has been made, the state is finalized:
412  * - the attitude error is moved into the body attitude quaternion,
413  * - the body attitude is converted into a rotation matrix for the next prediction, and
414  * - correctness of the covariance matrix is ensured
415  */
416
417 if (doneUpdate)
418 {
419     kalmanCoreFinalize(&scoreData, sensors, osTick);
420     if (! kalmanSupervisorIsStateWithinBounds(&scoreData)) {
421         scoreData.resetEstimation = true;
422         DEBUG_PRINT("State out of bounds, resetting\n");
423     }
424 }
425
426 /**
427  * Finally, the internal state is externalized.
428  * This is done every round, since the external state includes some sensor data
429  */
430 kalmanCoreExternalizeState(&scoreData, state, sensors, osTick);
431 }
432
433
434 void estimatorKalmanInit(void) {
435     // ...

```

שתי פונקציות בתמונה מטה מוגדרות עבור ה FLOWDECK והן תמיד ירוצו כל עוד ה FLOWDECK מורכב חומרית לרחפן. הן מספקות מידע של הגובה (ציר Z) לקאלמן אז על מנת לבטל את זה בזמן שמקבלים מידע מ SLAM2 עשינו שינוי עבור תנאי IF כפי שנראה בשתי תמונות תמונה.



```

104
105     range_last = vls3l0xReadRangeContinuousMillimeters(&dev);
106     rangeSet(rangeDown, range_last / 1000.0f);
107
108     // check if range is feasible and push into the kalman filter
109     // the sensor should not be able to measure >3 [m], and outliers typically
110     // occur as >8 [m] measurements
111     if (getStateEstimator() == kalmanEstimator &&
112         range_last < RANGE_OUTLIER_LIMIT) {
113         // Form measurement
114         if (SlamOrFlow) //dann
115         {
116             tofMeasurement_t tofData;
117             tofData.timestamp = xTaskGetTickCount();
118             tofData.distance = (float)range_last * 0.001f; // Scale from [mm] to [m]
119             tofData.stdDev = expStdA * (1.0f + expf( expCoeff * ( tofData.distance - expPointA)));
120             estimatorEnqueueTOF(&tofData);
121         }
122     }
123 }
124 }
125
126 bool zRangerReadRange(zDistance_t* zrange, const uint32_t tick)
127 {
128     bool updated = false;
129
130     if (isInit) {
131         if (range_last != 0 && range_last < RANGE_OUTLIER_LIMIT) {
132             zrange->distance = (float)range_last * 0.001f; // Scale from [mm] to [m]
133             zrange->timestamp = tick;
134             updated = true;
135         }
136     }
137     return updated;
138 }
139
140 static const DeckDriver zranger_deck = {
141     .vid = 0x8C,
142     .pid = 0x09,
143     .name = "bcZRanger",
144     .usedGpio = 0x0C,
145
146     .init = zRangerInit,
147     .test = zRangerTest,
148 };
149
150 DECK_DRIVER(zranger_deck);
151
152 PARAM_GROUP_START(deck)
153 PARAM_ADD(PARAM_UINT8 | PARAM_ROMLY, bcZRanger, &isInit)
154 PARAM_GROUP_STOP(deck)
155
156
157 }
158
159 bool zRanger2Test(void)
160 {
161     if (!isInit)
162         return false;
163     return true;
164 }
165
166 void zRanger2Task(void* arg)
167 {
168     TickType_t lastWakeTime;
169
170     systemWaitStart();
171
172     // Restart sensor
173     VLS3L1_StopMeasurement(&dev);
174     VLS3L1_SetDistanceMode(&dev, VLS3L1_DISTANCEMODE_MEDIUM);
175     VLS3L1_SetMeasurementTimingBudgetMicroSeconds(&dev, 25000);
176
177     VLS3L1_StartMeasurement(&dev);
178
179     lastWakeTime = xTaskGetTickCount();
180
181     while (1) {
182         vTaskDelayUntil(&lastWakeTime, M2T(25));
183
184         range_last = zRanger2GetMeasurementAndRestart(&dev);
185         rangeSet(rangeDown, range_last / 1000.0f);
186
187         // check if range is feasible and push into the kalman filter
188         // the sensor should not be able to measure >5 [m], and outliers typically
189         // occur as >8 [m] measurements
190         if (getStateEstimator() == kalmanEstimator &&
191             range_last < RANGE_OUTLIER_LIMIT) {
192             // Form measurement
193             if (SlamOrFlow) //dann
194             {
195                 tofMeasurement_t tofData;
196                 tofData.timestamp = xTaskGetTickCount();
197                 tofData.distance = (float)range_last * 0.001f; // Scale from [mm] to [m]
198                 tofData.stdDev = expStdA * (1.0f + expf( expCoeff * ( tofData.distance - expPointA)));
199                 estimatorEnqueueTOF(&tofData);
200             }
201         }
202     }
203 }
204
205 bool zRanger2ReadRange(zDistance_t* zrange, const uint32_t tick)
206 {
207     bool updated = false;
208
209     if (isInit) {
210         if (range_last != 0 && range_last < RANGE_OUTLIER_LIMIT) {
211             zrange->distance = (float)range_last * 0.001f; // Scale from [mm] to [m]
212             zrange->timestamp = tick;
213             updated = true;
214         }
215     }
216     return updated;
217 }
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```



## **תוצרי הפרויקט**

ביצועי המערכת מבחינת זמן אמת

- בזמן אמת קיבלנו שידור מהמצלמה והרחפן עם עיכוב קטן מאוד "delay" שאינו משפיע על ביצוע המערכת.
- המערכת תוכננה לעבד מידע בזמן אמת ולכן המעקב אחרי הרחפן היה עם דיוק רב .
- אחת הבעיות בזמן אמת הייתה שערור המרחקים / אורך המסלולים בזמן אמת כך שהיה צריך לעשות כיול ל SLAM כך שיהיה מותאם למרחקים כפי במציאות. הצלחנו להתגבר על בעיה זו על ידי זיהוי אותן תכונות ומיקום ובכך האלגוריתם יכול לתפור את המפה ולתקן אותה כשחוזרים לאותן הנקודות במפה.

## סרטוני הדגמה:



בוידאו זה ניתן לראות את הרחפן שלנו טס באמצעות רכיב ה Flow-Deck בלבד, מבצע תנועה מעגלית בגובה קבוע ונוחת. לרכיב זה ישנה סחיפה מסוימת ולכן הוא לא חוזר לאותה הנקודה בדיוק, לצורך כך, התחלנו לעבוד עם SLAM2 כפי שניתן לראות בוידאו הבא:



ניתן לראות שהרחפן חוזר בדיוק לאותה הנקודה (עם סטייה של סנטימטרים בודדים  
(כתוצאה מכך הליך ההמראה התבצע עם Deck Flown).



בסרטון זה ניתן לראות טיסה בחדר, מצד שמאל של המסך רואים את המיפוי של  
ה-SLAM על מסך המחשב, האלגוריתם מסמן נקודות קבועות בפריימים  
שמתקבלים בקלט הווידאו וכך קובע את מיקומו בחדר  $(x,y,z)$ .

## סיכום, מסקנות והמלצות להמשך

סיכום כללי מרגע קבלת המידע X,Y,Z מה PYTHON SCRIPT עד שהרחפן יפעל בהתאם:

This is an explanation for flow's diagram of receiving data X Y Z till arriving to power distribution control( where there the drone is going to work accordingly):

we've done explanation for Z coordination, but the same analogy is going with X Y MEASUREMENT.

1. The height measurement is done in the z-ranger deck driver and pushed into the estimator.
2. When using the flow or Slam(Any External Position), the kalman filter estimator is used. The height measurement is entered in the kalman filter by the function kalmanCoreUpdateWithTof.
3. Eventually the kalman filter generates an estimated pose including the current estimated height.

Then, the controller, outputs motor control(power distribution) in order to get the current estimated height closer to the desired height setpoint.

### בחינת תוצאות הפרויקט מול המטרות שהוגדרו מלכתחילה:

- בפרוייקט זה ביצענו שינויים רבים בדרך, התמודדנו עם בעיות רבות לאורך הפרוייקט שגרמו לנו לחקור את הנושא עוד יותר לעומק ולפתח מערכת עובדת וטסה בצורה טובה למדי ובדיוק רב הרבה יותר ממה שציפינו.
- ההישג המרכזי של הפרוייקט, היא בכך שכיום קיימת תשתית ופלטפורמה להטסה אוטונומית של רחפנים זעירים, יצרנו תשתית למערכות רבות ופתחנו את התחום לפרוייקטים נוספים באוניברסיטה.
- הפרוייקט מפשט את נושא הסטת הרחפנים הזהירים בחללים ללא GPS ופותח את הדלת ליצירת אלגוריתמי תנועה וניווט מתקדמים בתחומים שונים ומגוונים באמצעות API פשוט של מתן פקודות טיסה לנצ. מסויים(go to) או ע"י מסלול לרחפן.

### הצעות לשיפור ביצועי המערכת:

- המגבלה הגדולה ביותר של עולם הרחפנים כיום היא הסוללה, ברגע שמגבלה זו תיפתר זה יביא לשינוי אדיר בעולם כולו, ולא רק בשוק הרחפנים.
  - שיפור קצב העיבוד של האלגוריתם ע"י העברת העבודה ממחשב הלינוקס לג'טסון (Jetson TX2). פעולה זו תקטין את Latency - שנוצרת כתוצאה מתהליך העיבוד של המחשב.
  - מציאת מחשב בעל כוח עיבוד גבוה מספיק שיכול להיות מוטס על הרחפן ולהקטין את ה Latency כמעט לאפס (כמו למשל Nano Jetson).
-

### אפשרויות להמשך פעילות (פיתוח/מחקר) עתידית:

- אינטגרציה עם סנסורים להתחמקות ממכשולים (דוגמת Multi Ranger) על גבי הרחפן. בסרטון הבא מודגמת היכולת של מערכת מסוג זה אותה בדקנו במסגרת המחקר המקדים שלנו.



ניתן לראות שה Multi-Ranger מזהה את היד ושולח פקודה מידית לרחפן לטוס לצד השני (בסרטון הרחפן טס ללא SLAM)

- אינטגרציה עם מודולים נוספים קיימים בסביבת ROS .
- הפרוייקט שלנו יכול להוות בסיס לפרויקטים נוספים כמו:
  1. זיהוי ניצולים לאחר רעידת אדמה.
  2. חקירת מנהרות שלא נגישות לכניסת אדם
  3. זיהוי חומרים מסוכנים (עשן, גז, קרינה רדיואקטיבית וכדומה)
- כתיבת ובחינת אלגוריתמים מתקדמים לתנועה מורכבת במרחב ו/או אלגוריתמים לשיתוף פעולה בין קבוצה של רחפנים אוטונומיים , לסריקה וניווט במקביל.

## **רשימת מקורות:**

1. <https://www.bitcraze.io/start/>
2. [https://github.com/ros-drivers/video\\_stream\\_opencv](https://github.com/ros-drivers/video_stream_opencv)
3. [https://github.com/raulmur/ORB\\_SLAM2](https://github.com/raulmur/ORB_SLAM2)
4. [https://github.com/ECEBORG/Kinect-ORB\\_SLAM2](https://github.com/ECEBORG/Kinect-ORB_SLAM2)
5. [http://wiki.ros.org/orb\\_slam2\\_ros](http://wiki.ros.org/orb_slam2_ros)