



The Iby and Aladar Fleischman  
Faculty of Engineering  
Tel Aviv University

הפקולטה להנדסה  
ע"ש איבי ואלדר פליישרמן  
אוניברסיטת תל אביב



# ניווט אוטונומי של ננו-רחפן

פרויקט מס' 19-1-1-167

דו"ח סיכום

מבצעים:

302376736

עומר גביש

321928764

נתניאל פסחוב

מנחים:

אוניברסיטת ת"א

מר יונתן מנדל

מקום ביצוע הפרויקט:

המעבדה לרחפנים, אוניברסיטת ת"א

## תוכן עניינים

4.....	תקציר	
5.....	1 הקדמה	
6.....	2 רקע תיאורטי	
7.....	3 הכנות מקדימות	
8.....	3.1 תנאים מוקדמים	
9.....	3.2 קליברציה	
12.....	4 מימוש	
12.....	4.1 תיאור חמרה	
13.....	4.2 תיאור תוכנה	
14.....	4.3 הרצה	
16.....	5 תוצאות ביניים והמשך פיתוח	
17.....	6 סיכום, מסקנות והצעות להמשך	
18.....	מקורות	

## רשימת איורים

- 4..... איור 1 –דיאגרמת בלוקים
- 5..... איור 2-רכיבי המערכת
- 7..... איור 3 – רכיבי intel RealSense-d435i ו Bosch bno055
- 8..... איור 4-קליברציה עם april grid
- 9..... איור 5- קליברציה לIMU
- 10..... איור 6- קליברציה למצלמה
- 11..... איור 7 - הרצה עם intel RealSense-d435i ו Bosch bno055
- 12..... איור 8- סכמת חיבור המצלמה והרחפן (מתוך פרויקט משנים קודמות)
- 14..... איור 9 - קליברציה עם checkersboard
- 15..... איור 10- דיאגרמת בלוקים מפורטת
- 15..... איור 11- הרצה עם הרחפן
- 16..... איור 12 -דיאגרמת בלוקים חדשה
- 17..... איור 13-מערך חיישנים חדש

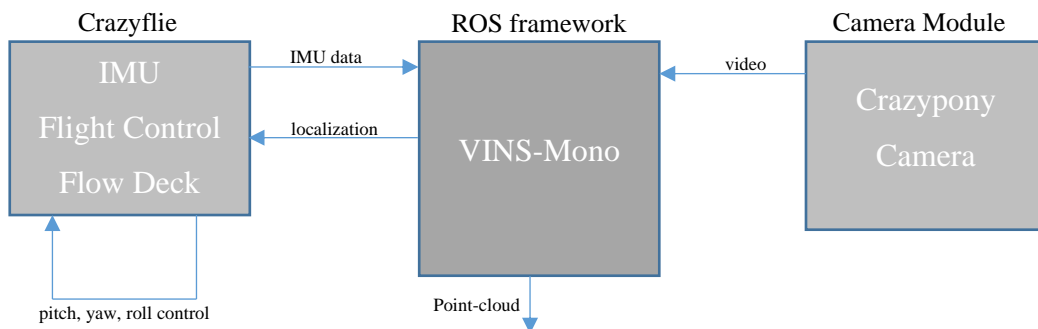
## תקציר

מהות הפרויקט – VISLAM על פלטפורמה קטנה ומוגבלת – ננו רחפן מסוג crazyflie עם optical-flow deck, ומצלמה אנאלוגית נפרדת מעוגנת לגוף הרחפן, בכדי להביאו למצב של ניווט אוטונומי במבנים ומיפויים.

מרכיבי המערכת :

1. נאנו-רחפן מסוג crazyflie עם בקר טיסה, משדר ו-IMU מובנה + flow deck.
2. מצלמת ננו FPV אנאלוגית עם משדרה 5.8GHz מובנה.
3. סביבת linux לביצוע ה VISLAM, עם מקלטים 5.8GHz – crazyradio.

האתגר בפרויקט הוא קליברציה ותיאום של החיישנים השונים לרמת הדיוק הגבוהה הנדרשת מהאלגוריתמים הקיימים ל tightly-coupled VI slam – האלגוריתם שנבחר בסופו של דבר היה VINS-Mono שמתירני יותר מבחינת תיאום חיישנים ואף יכול לשפר תיאום ב real-time, לעומת Rovio שדורש דיוק מוחלט. המימוש ל-VINS-Mono, ולתקשורת בין הרחפן למחשב ובין המצלמה למחשב נעשה בסביבת ROS.



איור 1 –דיאגרמת בלוקים

פירוט עיקרון הפעולה שבאיור 1 :

1. מדידות מה IMU והמצלמה משודרות ללפטופ, אליו מחוברים מקלט ה crazyradio ו eachine 5.8GHz, בהתאמה.
2. הנתונים מומרים ל ROS messages.
3. ה ROS messages הן הקלט למימוש VINS-Mono, הפלט הוא שערך מיקום, ו sparse pointcloud של נקודות העניין שנמצאו.
4. שערך המיקום מומר מ message – ROS ל type מוכר של ה firmware של ה crazyflie, ומשודר לבקר הטיסה שעל הרחפן כ reference לחוג הבקרה.
5. באופן נפרד, ה flow-deck שומר על הרחפן מאוזן – נדרשת שליטת מנועים בתדר גבוה בהרבה משניתן לקבל מ VINS-Mono.

## 1 הקדמה

מטרת הפרויקט היא בניית תשתית בסיסית לניווט אוטונומי של ננו-רחפן ("toothpick") בסביבה בה אין גישה ל GPS, שעדיין מהווה את הכלי הנפוץ ביותר לניווט רחפנים, ואין גישה למפה ידועה (Prior) של המבנה. הרחפנים הקיימים כיום בשוק אינם עונים על המפרט –

- ננו-רחפנים בד"כ מוגבלים מאוד מבחינה חומרית – ומנוטים בעזרת IMU בלבד, או עזר של מצלמה באיכות נמוכה המצלמת את הקרקע לביצוע OpticalFlow באיכות ירודה. המצלמה בקדמת הרחפן שמשרתת וידאו למרכז השליטה, אם קיימת, לא לוקחת חלק פעיל בניווט אוטונומי.
- רחפנים בהם כן ממומשים אלגוריתמים לניווט אוטונומי ללא GPS – גדולים מאוד ולא מתאימים לשימוש יום-יומי במעבדת רחפנים.

הבעיה של ניווט באזורים הנ"ל נפתרת בעזרת אלגוריתמי SLAM – Simultaneous Localization And Mapping. הבעיה הזו מהווה בעיית "ביצה ותרגולת" – בשביל מיקום מדויק צריך מפה, ובשביל לשערך מפה צריך מיקום מדויק. ישנם אינספור מאמרים המנסים לפתור את הבעיה הזו – אך ניווט אוטונומי של רחפנים בתעשייה מתרכז נכון להיום בשיטה הנעזרת במצלמה בלבד, ב IMU בלבד – או בשילוב נאיבי של שניהם בעזרת EKF, Information Filter ו Particle Filter.

המחקר באוניברסיטאות, ובפרויקט, מתרכז דווקא בגישה שונה – שילוב המצלמה וה IMU למשוואת מחיר חכמה. האלגוריתמים הללו נקראים VISLAM – Visual Inertial SLAM.

VISLAM מדויק בהרבה מ SLAM מכיוון שהמצלמה וה IMU, כנעשה בהם שימוש נכון, מהווים חיישנים שמפצים על החסרונות אחד של השני –

- SLAM עם מצלמה בלבד עובד מעולה כשהתנועות איטיות (או אפסיות), כשהתאורה טובה ושהסביבה עשירה בנקודות עניין טובות. אם התנאים הללו אינם מתקיימים – נעבד עקיבה מהר מאוד.
- IMU אינו תלוי בנראות הסביבה, ויודע למצע תנועות יחסית חדות בצורה טובה – אך בתנועה איטית מידי ה SNR הגבוה של החיישנים הנפוצים יגרום לעיבוד עקיבה (Drift).

אנו, כאמור, נתרכז באלגוריתמים אלו – וספציפית ב VinsMono, בשל סלחנותו, היחסית, לתיאום לא אופטימלי בין החיישנים. גישה זו טומנת אתגרים חדשים שלא נתקלים בהם ב SLAM סטנדרטי, במיוחד בסביבת עבודה מוגבלת של ננו-רחפן (כ-30 גר' Maximum Payload).

לצורך הרצת ה VinsMono השתמשנו ב IMU המובנה שברחפן ומצלמה אנלוגית (מתחום מרוצי הרחפנים) שהותקנה על גביו. נתוני ה IMU שודרו בעזרת המשדר\מקלט שברחפן, והוידאו בעזרת משדר נפרד שעובד בתחום התדרים סביב 5.8GHz, בעל רוחב פס צר. המצלמה נבחרה בשל משקלה הקטן והמשדר המותאם.



איור 2-רכיבי המערכת

הפרויקט אמנם משלב חומרה ותוכנה – אך עיקרו תוכנה. החלק החומרתי נעשה ברובו בשנים קודמות, ומעבר לחיבור ע"פ סכמה נתונה ווידוי תקינות מתחים עם probe לא התחדש, ועל כן לא נרחיב מעבר להסבר קצר בפרק 4.1-חומרה.

### VISLAM:

כפי שהוזכר – אנו התרכזנו ב VISLAM מכיוון שהמצלמה וה IMU מהווים משלימים אחד לשני – ונתקלנו באתגרים של שילוב חיישנים (Data Fusion). נציג אותם כעת:

### מידול IMU:

$$\tilde{\omega}(t) = \omega(t) + b^g(t) + n^g(t)$$

$$\tilde{a}(t) = R(t)(a(t) - g) + b^a(t) + n^a(t)$$

כאשר  $\tilde{\omega}(t), \tilde{a}(t)$  הם התאוצה הלינארית והמהירות הסיבובית ביחס למערכת ה IMU,  $\omega(t), a(t)$  הם התאוצה האמיתית במערכת העולם והמהירות הסיבובית האמיתית במערכת ה IMU (מה שאנו מנסים לשערך),  $g$  תאוצת הכבידה במערכת העולם, ו-  $b^x, n^x$  רעש גאוסיאני אדיטיבי ו bias (או random walk) כאשר ה bias הוא יחסי לרעש לבן.

נאיבית, ניתן למצוא את המיקום הנוכחי ע"י אינטגרציה פשוטה של המודל הנ"ל:

$$p_{n+1} = p_n + \Delta t \cdot v_n + \iint (R(t)(\tilde{a}(t) - b^a(t)) + g) dt^2$$

ניתן לראות שהשגיאה בשיטה זו יחסית לזמן בריבוע – נקבל drift מהר מאוד.

### מידול המצלמה:

ע"מ להשתמש במצלמה – יש לבצע קליברציה מדויקת לפרמטרים האינטרינזים והאקטרינזים שלה (ביחס ל IMU). הפרמטרים האינטרינזים – מיקום הפיקסל האמצעי בתמונה ואורך המוקד לכל ציר – ע"מ למצוא הומוגרפיה בין נקודה 3D לנקודה 2D על המצלמה.

הפרמטרים האקטרינזים – רוטציה וטרנזלציה בין המצלמה וה IMU – ע"מ למצוא טרנפורמציה אפינית מבסיס המצלמה לבסיס ה IMU.

ההנחה היא שהפרמטרים נשארים קבועים לאורך הריצה.

תהליך עבודת המצלמה בזמן הרצת האלגוריתם:

- Feature Extraction
- Feature Matching
- Estimate new camera pose
- Minimize reprojection error, iteratively

ל Data Fusion משני החיישנים יש כמה סוגים.

- פתרון סגור של מערכת משוואות לינארית בעזרת Pseudo Inverse – נעשה בלתי אפשרי מבחינת עלות חישובית לאחר כמה שניות \ כמה עשרות landmarks. בפועל, נעשה בו שימוש בעיקר לאתחול פרמטרים.
- Filtering: עדכון איטרטיבי של ה state הכי עדכני של המערכת בלבד בעזרת פילטר באייסיני. בד"כ פחות מדויק מהשיטות האחרות – נקודת הלינארציה ל EKF תלויה במצב הקודם, שעלול להיות שגוי, ובמקרה

זה נאבד את העקיבה. בנוסף, הסיבוכיות של ה EKF עולה ריבועית עם מספר ה Landmarks, כך שגם הייתרון של מהירות שיטה זו נהיה זניח די מהר.

- Sliding Window: עבודה על חלון מצבים מתעדכן וצמצום שגיאה בעזרת פתרון non-linear IS. שיטה זו קצת יותר איטית – גם מעצם העבודה על מספר מצבים, וגם מכיוון שצריך לעדכן את החלון במהלך הריצה.
- Full Smoothing: עבודה על כלל המצבים מתחילת הריצה. שיטה זו הכי מדויקת, אך גם הכי איטית. בד"כ צריך לעשות sparsefication ל landmarks כדי שניתן יהיה ליישם.

את כלל השיטות הנ"ל ניתן לממש באלגוריתם Tightly Coupled או Loosely Coupled:

- Tightly Coupled: משערכים את ה pose בהיתן מידע מהמצלמה וה IMU – קרי, פונקציית המחיר כוללת את שניהם. באלגוריתמים מסוג זה נדרש תיאום זמנים חומרתי בין החיישנים.
- Loosely Coupled: שיערוך המיקום בנפרד בעזרת המצלמה, ובנפרד בעזרת ה IMU – ולאחר מכן שילובם בעזרת EKF או דומיו.

VinsMono הוא אלגוריתם Tightly Coupled Pseudo-Sliding-Window, כך שלא כל State חדש נכנס – צריך לעבור סף איכות לפני כן, אחרת נזרק. אתחול המערכת נעשה דווקא בגישת Filtering עם EKF, ולא בפתרון מערכת סגורה.

אלגוריתם נוסף שנוסה הוא Tightly Coupled Filtering – Rovio. בסופו של דבר VinsMono נבחר מכיוון שמסוגל להתגבר על חוסר תיאום זמנים בין המצלמה וה IMU ועל עבודה עם מצלמות rolling shutter.

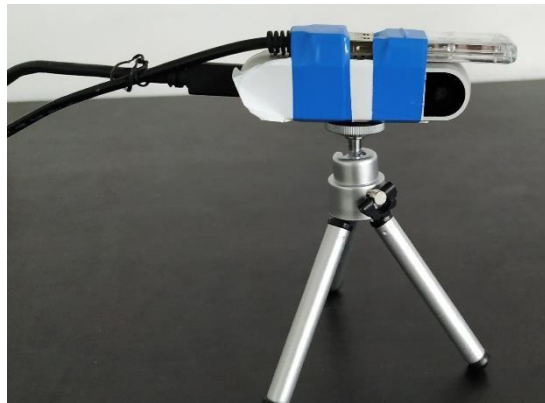
#### Loop Closure

למרות ש VSLAM משמעותית יותר מדויק מ SLAM סטנדרטי, עדיין נקבל לרוב drift קטן לאחר ריצה ממושכת. לכן, מתבצע Online Loop Closure. ביצוע Loop Closure, בפשטות, מקטין את אי הוודאות האדיטיבית של ה states האחרונים במקרה שנמצא שהם מתאחדים עם state בעל שגיאה אפשרית קטנה יותר.

### 3 הכנות מקדימות

עקב קשיים בפרויקטים משנים קודמות התחלנו בביצוע הפרויקט עבור IMU ומצלמה רובסטים יותר – Bosch bno055 ו intel RealSense-d435i. ה RealSense הוא חיישן ייעודי לביצוע SLAM, וכולל:

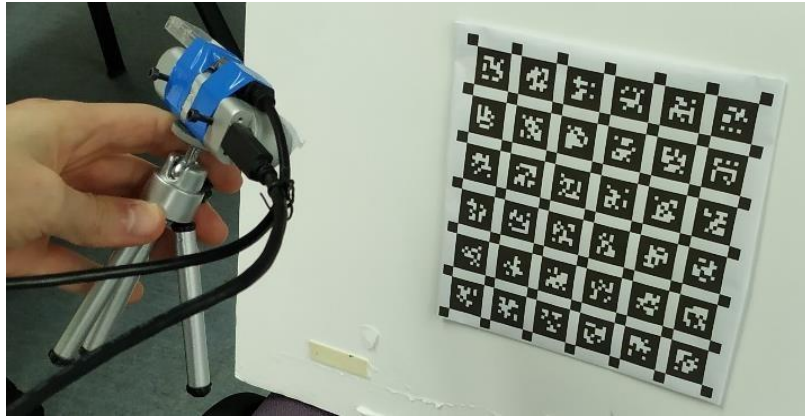
- High Frequency Accelerometer, Gyro.
- Two global-shutter monochrome cameras (i.e. depth-sensor)
- A (usually unused, for visualization purposes only) rolling-shutter RGB camera.



איור 3 – רכיבי intel RealSense-d435i ו Bosch bno055

השתמשנו רק במצלמת RGB, כך שמערכת החיישנים תהיה הכי קרובה למה שבסופו של דבר יהיה ברחפן.

השלב הראשון בעבודה עם האלגוריתם VINS-mono, הוא בדיקה האם הוא יכול לפעול בצורה חלקה עם מצלמה IMU באיכות גבוהה – במקרה שלנו המצלמה היא RGB global shutter - realsense d435i של חברת אינטל. כדי להריץ VINS-mono, היינו צריכים לכייל את המצלמה ואת ה-IMU, תחילה בנפרד, באמצעות imu\_utils ו-Kalibr, ואז לסנכרן ביניהם, גם באמצעות Kalibr. עם תוצאות הכיול הפעלנו VINS-mono ברמת דיוק גבוהה יחסית.



איור 4-קליברציה עם april grid

### 3.1 תנאים מוקדמים

סביבת עבודה של LINUX שיכולה לעבוד עם kernel בגרסה גבוהה מ-4.16 אבל נמוכה מ-4.8 (לדוגמה: Ubuntu 16).

ספריות רבות בפרויקט משתמשות ב-ceres-solver לפתירת משוואות אי ליניאריות. [[1]]

התקנת ros-kinetic: נדרש לעקוב אחר מדריך ההתקנה עד החלק של יצירת catkin\_workspace. [[2]]

ספריית ros של bno055\_usb\_stick [[3]]

ספריית realsense-SDK, כמו כן ספריית ros, realsense-ros [[4]] [[5]]

עבור הקליברציה, ספריות ros: Kalibr ros, imu\_utils. [[6]]

ספריית ros של VINS-mono. [[7]]

קבצי ה-launch והקונפיגורציה ניתן להוריד מעמוד github שלנו. [[8]]

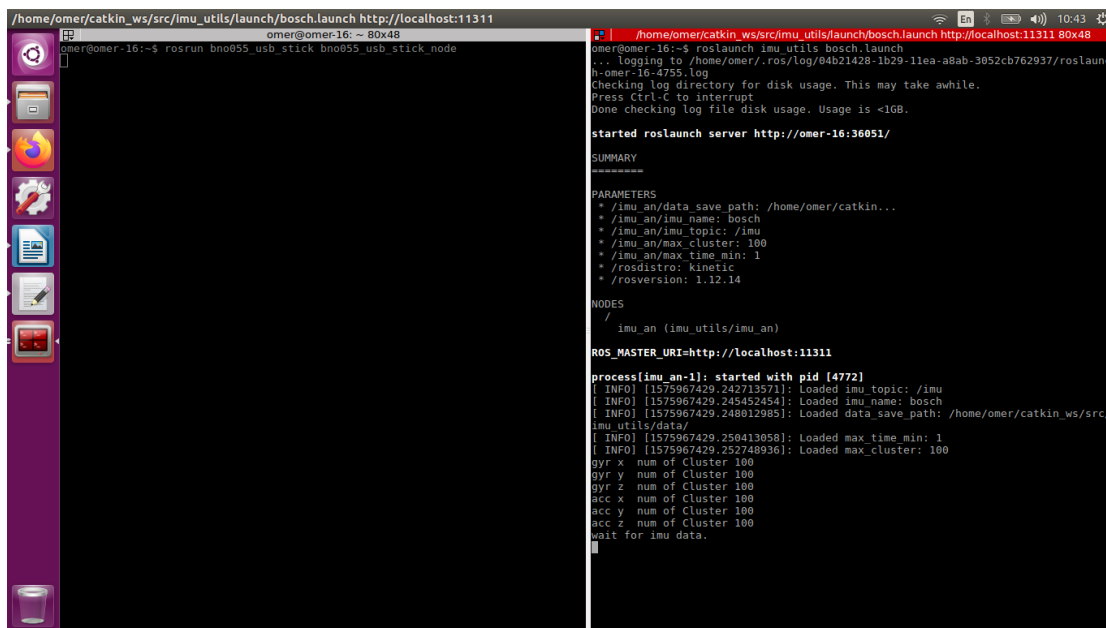


## 3.2 קליברציה

נדרש להפעיל את תוכנת ros ע"י הרצת פקודת 'roscore' בטרמינל. בתהליך הקליברציה נשים לב שבכל טרמינל שנפתח, נדרש להריץ source devel לסביבת העבודה.

### IMU

לפני הרצת הקליברציה, ניקח את קובץ `bosch.launch` מהגית'הוב שלנו לתיקיית `imu_utils/launch`. נחבר את הIMU למחשב, נוודא כי הוא יציב במהלך הריצה. נפתח שני טרמינלים. באחד נתחיל את הIMU node (הטרמינל השמאלי באיור מטה), בטרמינל השני נתחיל את הcalibration node (הטרמינל הימני באיור מטה). הריצה לוקחת כחצי שעה עד לקבלת התוצאות, אשר יהיו בתיקייה `imu_utils/data/bosch*`.



```
o@omer-16:~$ roslaunch imu_utils bosch.launch http://localhost:11311
o@omer-16:~$ roslaunch imu_utils bosch.launch
.. Logging to /home/o@omer-16/.ros/log/04b21428-1b29-11ea-a8ab-3052cb762937/roslaunch
h-omer-16-4755.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://omer-16:36051/

SUMMARY
=====
PARAMETERS
* /imu_an/data_save_path: /home/o@omer-16/catkin...
* /imu_an/imu_name: bosch
* /imu_an/imu_topic: /imu
* /imu_an/max_cluster: 100
* /imu_an/max_time_min: 1
* /roscistro: kinematic
* /rosversion: 1.12.14

NODES
/
  imu_an (imu_utils/imu_an)

ROS_MASTER_URI=http://localhost:11311

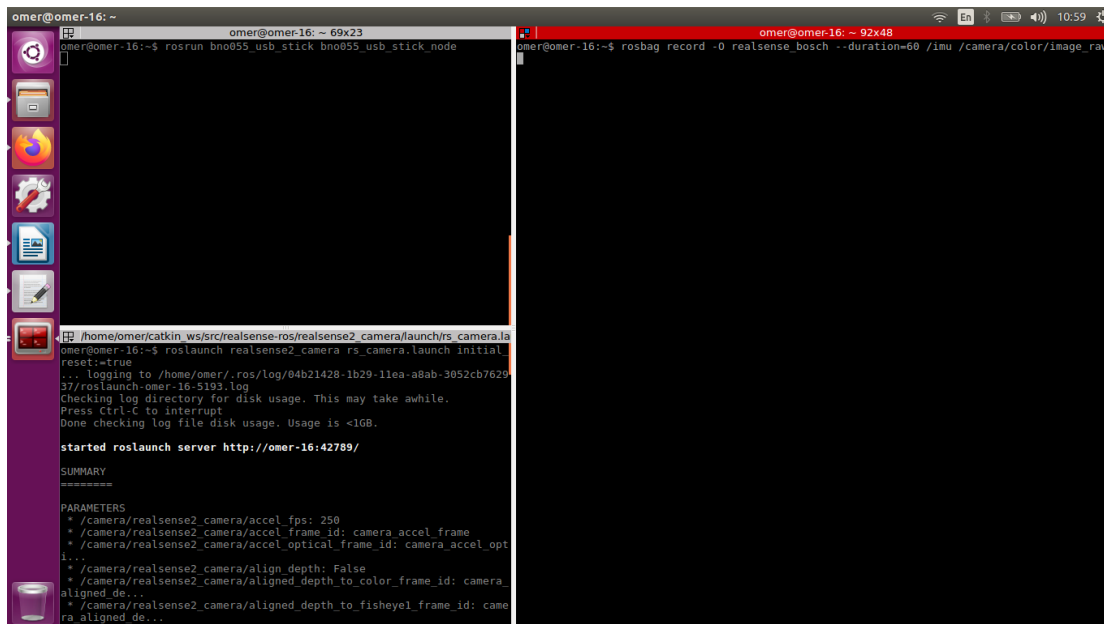
process[imu_an-1]: started with pid [4772]
[ INFO] [1575967429.242713571]: Loaded imu_topic: /imu
[ INFO] [1575967429.245452454]: Loaded imu_name: bosch
[ INFO] [1575967429.248012985]: Loaded data_save_path: /home/o@omer-16/catkin_ws/src/
imu_utils/data/
[ INFO] [1575967429.250413058]: Loaded max_time_min: 1
[ INFO] [1575967429.252748936]: Loaded max_cluster: 100
gyr x num of Cluster 100
gyr y num of Cluster 100
gyr z num of Cluster 100
acc x num of Cluster 100
acc y num of Cluster 100
acc z num of Cluster 100
wait for imu data.
```

איור 5- קליברציה לIMU

### מצלמה

נוודא כי הIMU עדיין מחובר, כעת נחבר לUSB3 את מצלמת הrealsense. נדפיס את הapril grid שלנו [8], זה מאוד מפשט את איסוף הנתונים בתהליך משום שניתן להשתמש בלוחות כיול גלויים באופן חלקי, לכן הלוח הנ"ל הופך לבחירה המומלצת. נבצע מדידה של רכיבי הלוח ונערוך את הקובץ `april_grid.yaml`.

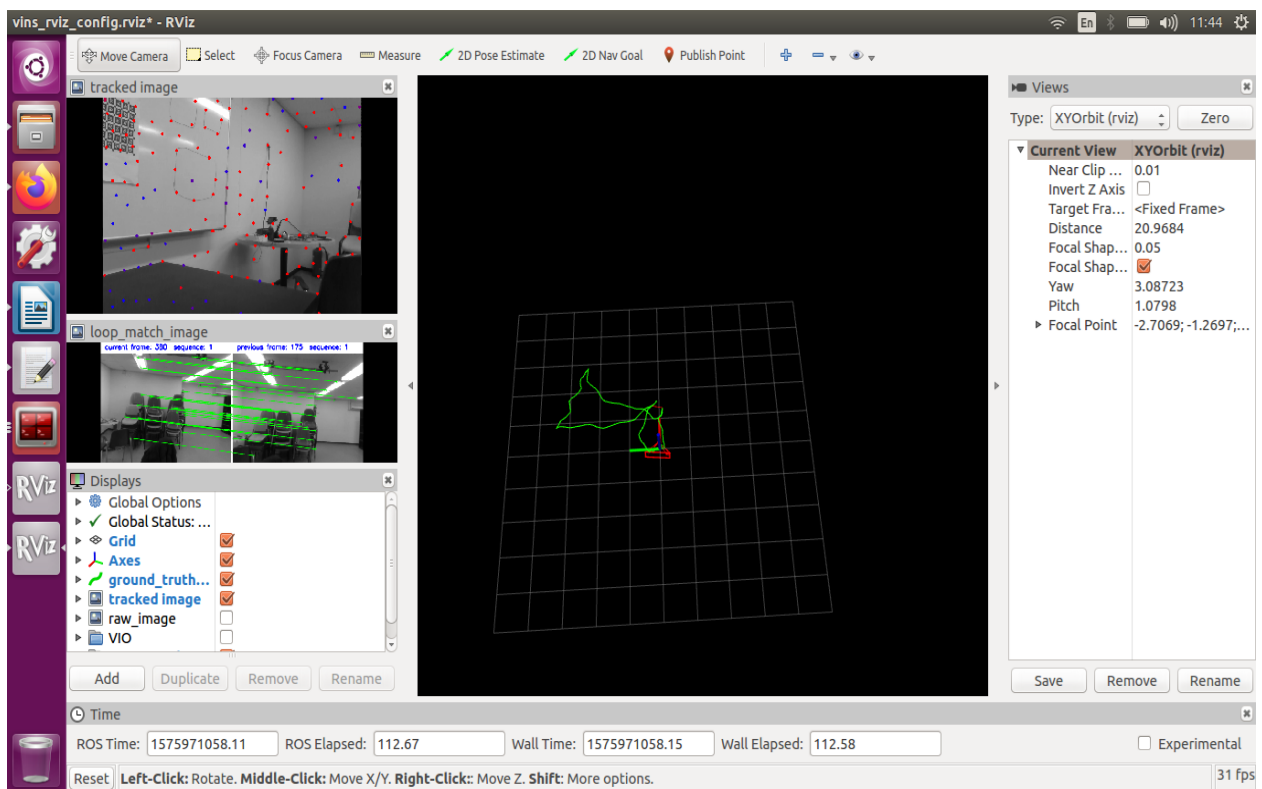
נפתח שלושה טרמינלים. באחד נריץ את הIMU node, בשני נריץ את הrealsense node. נרצה להקליט תנועות שונות של המצלמה והIMU מול הapril grid, ניעזר בגית'הוב של Kalibr. לכן בטרמינל השלישי נקליט קובץ `bag*` את topics של `/camera/color/image_raw` ושל `/imu`.



איור 6- קליברציה למצלמה

עבור הקליברציה, נריץ את הפקודה:  
`kalibr_calibrate_cameras --model pinhole-radtan --target april_6x6.yaml --bag [].bag --topics /camera/color/image_raw`  
 כאשר נחליף את [] במיקום של קובץ bag שהקלטנו, התוצאות יהיו בלcamchain.yaml].

סנכרון בין המצלמה לIMU  
 נבצע עריכה של תוצאות הקליברציה של IMU כדי שיהיו באותו פורמט של Kalibr, כפי שניתן לראות במדריך בgithub של Kalibr.  
 לאחר מכן נריץ את הפקודה:  
`kalibr_calibrate_imu_camera --cams [0].yaml --bag [1].bag --imu [2].yaml --target april_6x6.yaml`  
 כאשר [0] מציין את תוצאות הקליברציה של המצלמה, [1] מציין קובץ ההקלטה, [2] מציין תוצאות הקליברציה של IMU בפורט המתאים.



## איור 7 - הרצה עם Bosch bc055 ו intel RealSense-d435i

התוצאות של הרצה ידנית עם Bosch bc055 ו intel RealSense-d435i היו אחידות, ובדיוק גבוה.  
VinsMono הצליח לפצות על חוסר התיאום החומרתי שאילצנו online.

## 4.1 תיאור חמרה

**רחפן:** ננו-רחפן מסוג crazyflie2.1 של bitcraze. הייתרון המרכזי של הרחפן, מעבר למשקלו, הוא שהלוח הוא PCB open-source, כלומר ניתן לחבר רכיבים חיצוניים, ולשנות את ה firmware לפי דרישה. בנוסף, השתמשנו ב deck הרחבה לרחפן – flow deck – שמממש חוג בקרה מבוסס EKF ע"ס ה IMU המובנה וחיישן מרחק (מהקרע) הנמצא עליו. בפרויקט נעשה גם שינוי מינורי ב firmware, כך שלאחר מס' פקודות – חוג הבקרה יזן ע"י המיקום מ VinsMono, ולא מה flow-deck.

בנוסף, קיים על הרחפן משדר\מקלט מובנה לתקשורת בעזרת crazyradio שמגיע עם הרחפן.

### מצלמה:

מצלמת CrazyPony-nano עם משדר רדיו 5.8GHz מובנה. מתחום מרוצי הרחפנים – ולכן בעלת latency נמוך (<30ms) ומשקל כולל מזערי של כ 6gr.

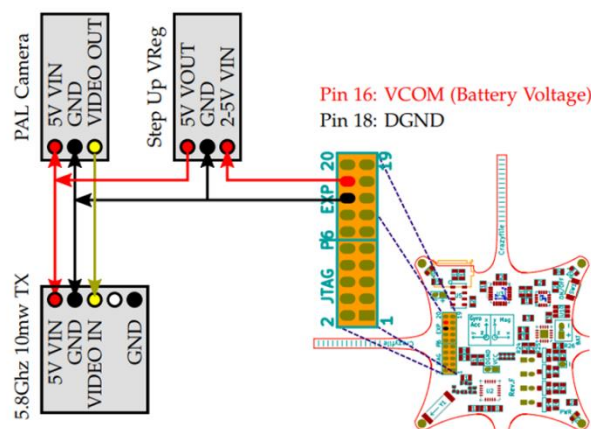
למצלמה גם חסרונות מהותיים:

- איכות ירודה.
- הפרעות שידור.
- אין אפשרות ל external triggering, ולכן גם אין אפשרות לתיאום חומרת. יתרה מכך – ה Timestamp נקבע רק בהגעת ה frame למחשב ה host, ולכן אין הכרח שה framerate אף יהיה קבוע, וככל הנראה ייפגע בזרימה של שידור.
- מצלמת FPV – התמונה המתקבלת מותאמת למשקפי VR, ועל כן צריך לבצע עיבוד תמונה שרק מגביר את הרעש.

### מקלט:

מקלט רדיו 5.8GHz, שעובד ישירות מול ספרייט v2in. המקלט בעל latency יחסית נמוך של 100ms – אך מדובר עדיין בהפרש של סדרי גודל מהרצוי בהרצת VISLAM שאינו מפצה על חוסר תיאום חומרת. מתוך התייעוד של Rovio, ההמלצה היא על תיאום של עד 3ms, כאשר שעם 5ms-10ms עוד ניתן להריץ בתנאי תנועה איטית, וכל דבר מעל 20ms – לא יעבוד.

ע"ס ה latency של המצלמה, של ה IMU, ושל המשדרים נוכל לתת ל VinsMono ניחוש התחלתי להפרש הזמנים עליו יבצע אופטימיזציה: **32ms**.



איור 8- סכמת חיבור המצלמה והרחפן (מתוך פרויקט משנים קודמות)

## 4.2 תיאור תוכנה

כאמור, כלל התוכנה מומשה בסביבת ROS. להלן כמה מושגים חשובים:

- Node** – תוכנית עצמאית שרצה ברקע. Node יכולה לקבל, לעבד ולפרסם מידע ל sockets.
- Topic** – גשר לפרסום וקבלת מידע – בעצם קישור שם ב LUT ב ROS למספר socket מסוים, סוג של Bind.
- Message** – Topics יכולים לפרסם Messages (Publish) ולקבל Messages (Subscribe) מוגדרות מראש בלבד.
- Package** – Node או אוסף Nodes בעלות מכנה שנבנות ביחד.

העובדה ש ROS עובד על sockets מאפשר לנו, פרט לתקשורת בין Nodes על אותו Host – לתקשר גם עם Nodes על Slaves אחרים, או עם Sockets נפרדים מ ROS. כך לדוגמא, אנו יכולים להריץ Node שמתקשר עם קבצים ב /dev/\* (IMU, Cam) ממיר את המידע להודעות ומפרסם אותם – לכל Node אחר שיידרש. דוגמא נוספת היא ביזור תכנית מורכבת – VinsMono, לדוגמא, מורכב משלוש Packages, כשלכל אחת מס' Nodes, ויכולה לעבוד באופן עצמאי מהאחרות.

הפרויקט עצמו מורכב משלושה חלקים עיקריים, כולם כתובים ב ++c, בתוספת scripts קטנים ב python, shell:

Video-stream-opencv:

בעצם ROS Wrapper לספריית native של linux – v4l, עם פונקציות opencv. ה Package מורכב מ Node יחיד – תפקידו לקבל מידע ממצלמה ב /dev/videoXX או קובץ וידאו, להמיר למידע שניתן לעבד ב opencv, לבצע עיבוד תמונה (אופציונלי), להמיר ל ROS Message, ולפרסם (Publish) ל Topic מסוים.

אנו עושים שימוש ב Package זה ע"מ להמיר מידע ממצלמת ה CrazyPony ל ROS Message. חיבור המצלמה למחשב פותח קובץ /dev/video01, ולאחר הרצת פקודת ה ROS:

```
roslaunch video_stream_opencv camera.launch video_stream_provider:=1
```

נקבל את ה ROS Message המבוקש שמפורסם לתוך topic בשם /camera/image\_raw. משמעות הפקודה הנ"ל – הרצת Node עם פרמטרים שמוגדרים בקובץ בשם camera.launch שנמצא ב package בשם video\_stream\_opencv, כאשר אנו עושים remapping למשתנה בשם video\_stream\_provider מהערך ה default שלו לערך 1.

### Crazyflie-ros

ROS Wrapper ל driver של bitcraze לשליטה ובקרה ברחפן crazyflie מסביבת linux. מכיל מספר packages, כאשר לכל package מספר Nodes. אנו עושים שימוש ב package בשם crazyflie\_driver, ובאופן מסוים גם ב crazyflie\_demo.

Crazyflie\_driver מכיל למעשה שני Nodes – crazyflie\_server, ו crazyflie\_add. Crazyflie\_server מאתחל קו תקשורת בין רחפני crazyflie למחשב. מרימים עם הפקודה:

```
roslaunch crazyflie_driver crazyflie_server
```

לאחר מכן ניתן להריץ את הפקודה:

```
roslaunch crazyflie_driver crazyflie_add uri:="/0/80/2M/XXXXXX"
```

שמרימה Node מסוג crazyflie\_add, ומאפשרת תקשורת בין ה crazyflie\_server לרחפן עם ה uri המצויין. כעת – אנו יכולים לקבל מידע מה IMU של הרחפן כ ROS Message, ובנוסף אנו יכולים לשלוח פקודות שליטה לרחפן. כתבנו גם launch file במאחד את שני התהליכים.

## VinsMono

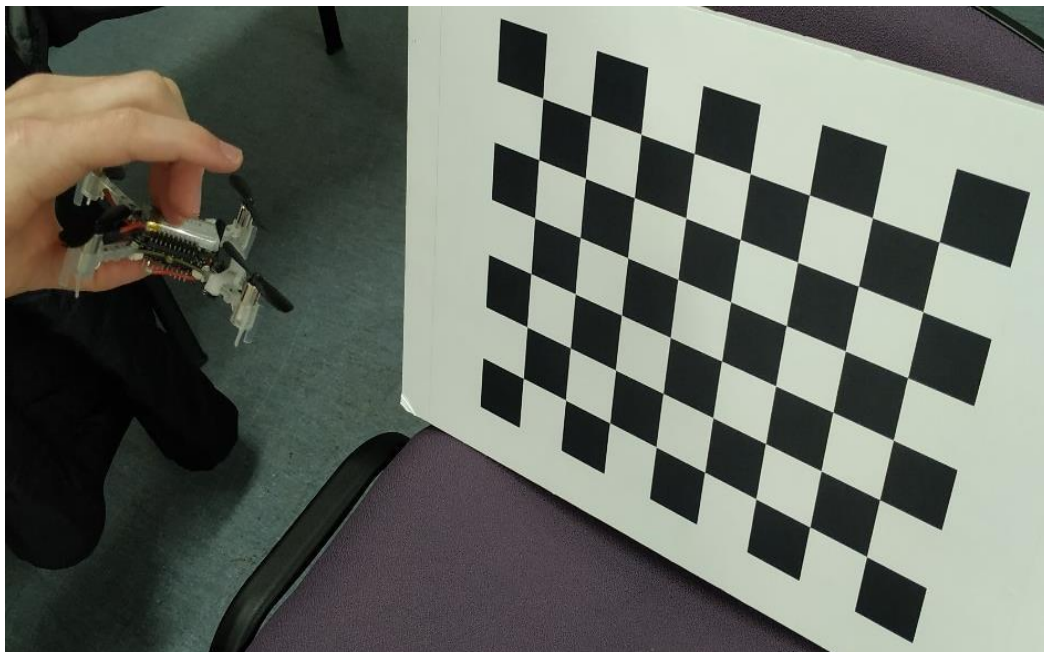
מכיל שלוש Packages בהן אנו עושים שימוש:

**Feature\_tracker** – מכיל Node יחיד. מקבל מידע מהמצלמה, ומפרסם point cloud של features טובים לעקיבה  
**Vins\_estimator** – מכיל Node יחיד. מקבל מידע מהtopics שמפרסמים מידע מה IMU וה feature\_tracker, ובעצם מבצע VISLAM. מפרסם, בין השאר, point cloud מעודכן, ו estimated odometry.  
**Pose\_graph** – בוחר, זורק ושומר keyframes, שומר היסטוריית estimated pose לצרכי ויזואליזציה, ובנוסף אחראי על loop closures.

ב package crazyflie\_demo כתבנו python script מבוסס על אחת הדוגמאות – אנו בעצם עושים subscribe ל estimated pose topic, ממירים את ה ROS Message להערכת מיקום שה crazyflie יכול לקרוא – ושולחים לחוג הבקרה.

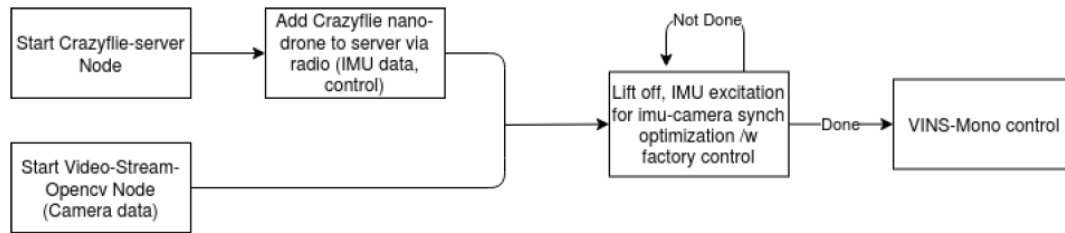
## **4.3 הרצה**

תחילה – ביצענו קליברציה, כפי שביצענו עבור realsense+bno055, עבור ה crazyflie. מחיין שאיכות התמונה ירודה – נאלצנו להשתמש ב checkersboard במקום ב April-grid. בנוסף – קליברציה של סנכרון המצלמה וה IMU לא הצליחה להתכנס.

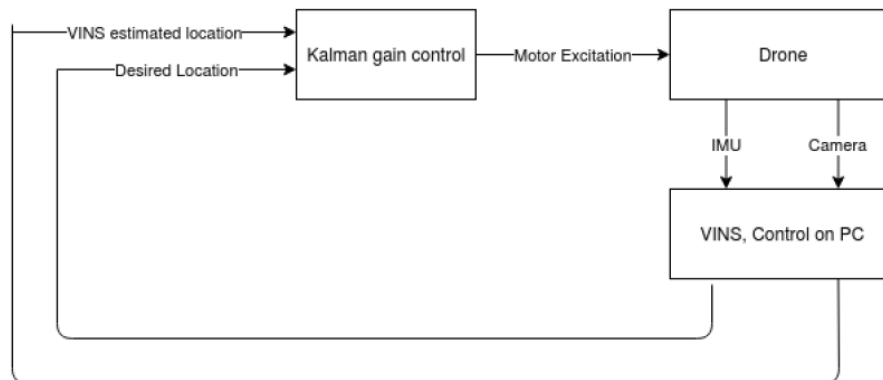


**איור 9 - קליברציה עם checkersboard**

הקליברציה כאמור, אינה מדויקת – אנו מסתמכים על אתחול online טוב, חישוב time offset אינקרמנטלי, ואיכות שידור טובה ורציפה לאורך הריצה. תנאים אלו בד"כ אינם מתקיימים.

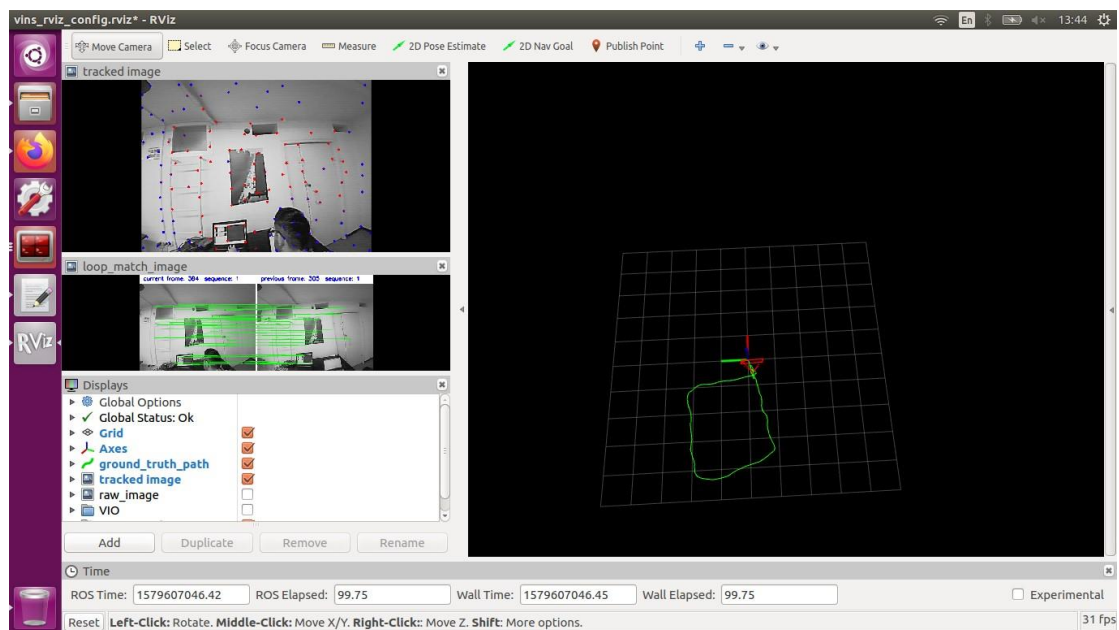


## Vins-Mono in depth



## איור 10- דיאגרמת בלוקים מפורטת

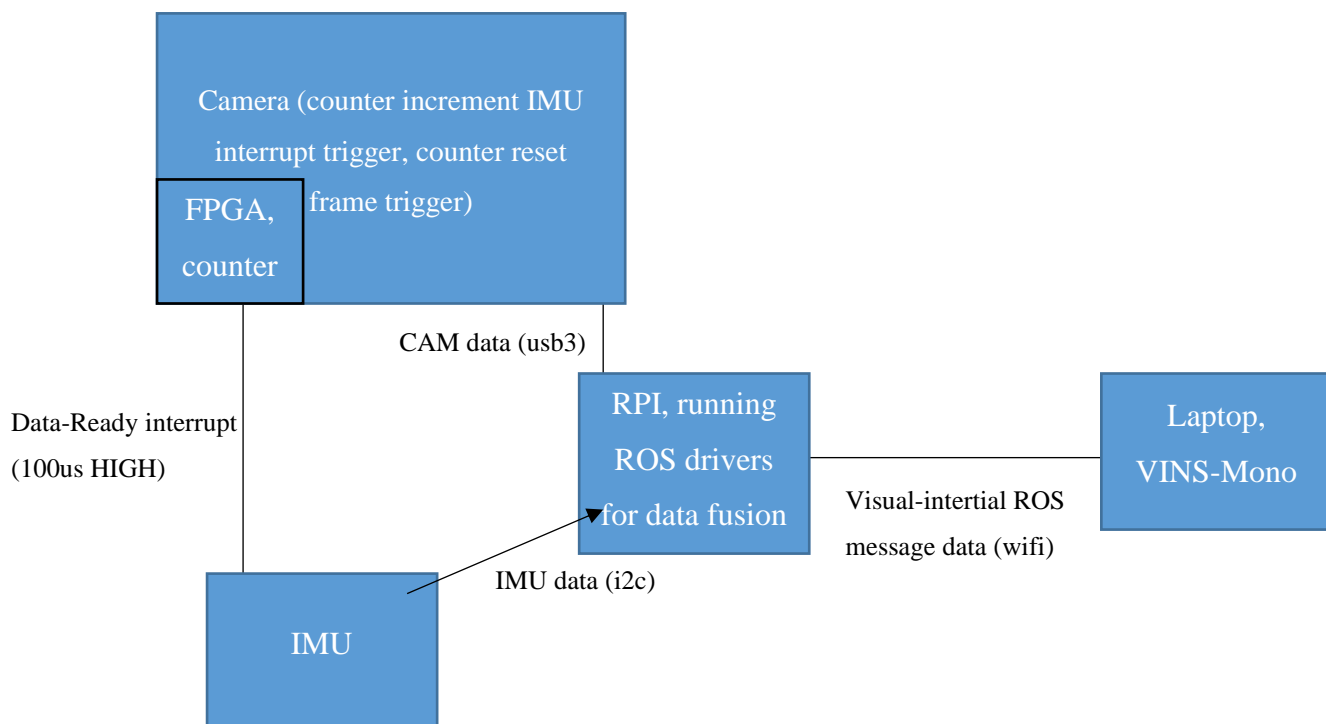
את ההרצה אנו מבצעים ע"י הרמת ה crazyflie server והוספת רחפן (launch file יחיד), ותחילת video stream, כפי שפורט למעלה. לאחר מכן נותנים ל VINSMono לבצע אתחול ומאמתים את יציבותו. בשלב הזה אנו מאחדים את ראשית הצירים של הבקרה המובנית של הרחפן עם ראשית הצירים של VINSMono עם טרנזלציה פשוטה – ושולחים פקודת המראה לרחפן. דוגמא להרצה טובה:



## איור 11- הרצה עם הרחפן

## 5 תוצאות ביניים והמשך פיתוח

- האלגוריתם לא רץ טוב עם crazyflie, בגלל השידור משני משדרים שונים של מידע אינרציאלי ווידאו וקביעת timestamp רק בקבלה. דוגמא להרצת crazyflie, לעומת realsense:
  - crazyflie - <https://www.youtube.com/watch?v=m14CSGI2Z4M>
  - realsense – <https://www.youtube.com/watch?v=TM-clpDDngE>
- עקב כך הוחלט לנסות שני מערכי חיישנים:
  - Realsensed435i – כותבי VinsMono שחררו לאחרונה מדריך לעבודה עם החיישן הזה עבור VinsFusion, אך ניתן לנסות להשתמש בו גם ל VinsMono. במקום לעבוד עם המצלמת color, rolling-shutter, ניתן לעבוד עם אחת ממצלמות העומק, שכן מסונכרנת עם ה IMU של החיישן, ע"י כיבוי ידני של ה IR Projector, ושימוש ב topic של תמונת החום כתמונה מונוכרומטית רגילה.
  - קישור למדריך בביבליוגרפיה [10]
  - פיתוח מערך חיישנים חדש, וכתובת drivers ל ROS, המבוסס על MPU9150 ומצלמת mvbluefox3-M5. הרעיון המרכזי במערך החיישנים הזה הוא שימוש ב IMU בתור micro-controller לשליחת hardware trigger מסונכרן למצלמה ללקיחת תמונה, קריאת המידע והמרתו לפורמט תואם ROS על raspberry-pi, ושדור לאחר, כבר בפורמט ROS, למחשב המריץ VINS-Mono.



איור 12 -דיאגרמת בלוקים חדשה

קישורים לדרייברים שפותחו:  
 עבור המצלמה: bluefox - [11]  
 עבור הIMU: mpu9150 - [12]





## מקורות

### קישורים למקורות באינטרנט:

- [1] הוראות התקנה לceres-solver:  
<http://ceres-solver.org/installation.html>
- [2] הוראות התקנה ROS-KINETIC:  
<http://wiki.ros.org/kinetic/Installation/Ubuntu>
- [3] ספריית rosl של bno055\_usb\_stick:  
[https://github.com/yoshito-n-students/bno055\\_usb\\_stick](https://github.com/yoshito-n-students/bno055_usb_stick)
- [4] ספריית realsense-SDK:  
<https://github.com/IntelRealSense/librealsense/blob/master/doc/installation.md#video4linux-backend-preparation>
- [5] ספריית realsense-ros:  
<https://github.com/IntelRealSense/realsense-ros>
- [6] ספריות rosl: imu\_utils, Kalibr  
[https://github.com/gaowenliang/imu\\_utils](https://github.com/gaowenliang/imu_utils)
- [7] ספריית rosl של VINS-mono:  
<https://github.com/HKUST-Aerial-Robotics/VINS-Mono>
- [8] עמוד הgithub שלנו:  
<https://github.com/tau-adl/Omer-Nataniel-Project>
- [9] mvImpact SDK:  
[https://www.matrix-vision.com/manuals/SDK\\_CPP/index.html](https://www.matrix-vision.com/manuals/SDK_CPP/index.html)
- [10] VinsFusion – Realsense435i:  
<https://github.com/HKUST-Aerial-Robotics/VINS-Fusion/issues/6>
- [11] דרייבר bluefox3:  
[https://github.com/tau-adl/bluefox3\\_ros.git](https://github.com/tau-adl/bluefox3_ros.git) (docs.odt for specifics)
- [12] דרייבר mpu9150:  
<https://github.com/tau-adl/mpu9150.git>