**אוניברסיטת תל אביב**

הפקולטה להנדסת חשמל



**מטע"ד אלחוטי לרחפן לדגימת מצלמה וחיישן אינרציאלי בזמן אמת**

**Real Time Wireless Drone’s Payload for**

**Sampling Images and IMU data**

פרויקט גמר לתואר שני בהנדסת חשמל

ע"י

**דן בלנארו**

מנחה הפרויקט: מר יהונתן מנדל

המעבדה לרחפנים אוטונומיים באוניברסיטת תל-אביב

דצמבר 2020

1. תוכן עניינים

[1 תוכן עניינים 2](#_Toc59627668)

[1 מבוא 3](#_Toc59627669)

[2 מטרות הפרויקט 4](#_Toc59627670)

[3 איפיון דרישות המערכת 4](#_Toc59627671)

[3.1 בחירת הרכיבים 4](#_Toc59627672)

[3.2 סקר שוק 5](#_Toc59627673)

[3.3 מצלמה 6](#_Toc59627674)

[3.4 חיישן IMU 6](#_Toc59627675)

[3.5 המודול האלחוטי 6](#_Toc59627676)

[3.6 הבקר 7](#_Toc59627677)

[3.7 חיבורים ואספקות מתחים 7](#_Toc59627678)

[3.8 סקירת רכיבי הפרויקט 8](#_Toc59627679)

[4 תכן מערכת 9](#_Toc59627680)

[4.1 תרשים בלוקים: 9](#_Toc59627681)

[4.2 הבקר - STmicro STM32H743 10](#_Toc59627682)

[4.2.1 BSP והקצאת GPIOs – Board Support Package 11](#_Toc59627683)

[4.2.2 מפת חלוקת שעון מרכזי 13](#_Toc59627684)

[4.2.3 פריפריות בשימוש 14](#_Toc59627685)

[4.3 מצלמה – ON Semiconductor MT9V034 19](#_Toc59627686)

[4.4 מודול הWIFI - Microchip ATSINC1500 20](#_Toc59627687)

[4.5 החיישן האינרציאלי – IMU – Invensense MPU6050 21](#_Toc59627688)

[5 ארכיטקטורת תוכנה 22](#_Toc59627689)

[5.1 ארכיטקטורה 22](#_Toc59627690)

[5.1.1 מבנה שכבות הקוד 22](#_Toc59627691)

[5.2 תכנת תחנה קרקעית 25](#_Toc59627692)

[5.3 ממשקי DEBUG, צריבה ושימוש במערכת 26](#_Toc59627693)

[6 תוצאות 27](#_Toc59627694)

[6.1 תצורת המדידה 27](#_Toc59627695)

[6.2 תוצאות 27](#_Toc59627696)

[7 סיכום 28](#_Toc59627697)

[7.1 סיכום 28](#_Toc59627698)

[7.2 עבודות המשך 28](#_Toc59627699)

[8 נספחים 29](#_Toc59627700)

[8.1 תמונת נוספות של המערכת 29](#_Toc59627701)

[8.2 ביבליוגרפיה 30](#_Toc59627702)

# מבוא

רחפנים כיום מנווטים באמצעות עיבוד החיישנים שמותקנים עליהם. אם עבור רחפנים בסביבות חיצוניות (outdoor) משתמשים בחיישנים כגון GPS, הרי שלרחפנים בסביבות סגורות (Indoor) לא ניתן לשתמש בחיישן כזה.

הפרויקט עוסק במימוש מטע"ד (payload) אלחוטי וקומפקטי לרחפן, למטרות חישה ודגימה של רכיבים לטיוב הניווט באמצעות אלגוריתם SLAM.

לטובת הנ"ל, ניתן להשתמש בחיישני עזר נוספים, כגון מצלמה - להתמצאות במרחב,  
ובחיישנים אינרציאליים (IMU)- שיכול להשלים מידע לאלגוריתמי עיבוד תמונה בנוגע למהירות התנועה וקצב התנועה של הרחפן. באמצעות היתוך מידע בין 2 החיישנים, ניתן להשיג מידע חזותי-אינרציאלי ולבצע מיפוי והתמצאות של המרחב (SLAM) [1].

רחפנים שמסוגלים לנווט בסביבה סגורה יכולים להביא תועלת רבה במקום שבה רחפנים לסביבות חיצוניות מביאים – לדוגמא ליישומי בנייה, חקלאות, רפואי, מסחרי, צבאי, קולנוע וגם לפנאי.

העבודה כאן מציגה שיתוף פעולה במעבדה לרחפנים אוטונומיים באוניברסיטת תל אביב TAU-ADL, על מנת לממש את מטרות הפרויקט

* סקר ספרות
* הבנת דרישות המערכת
* תכן על
* סקר שוק ובחירת רכיבים
* מימוש החומרה והתכנה לפרויקט
* חקר ביצועים
* סיכום

# מטרות הפרויקט

הפרויקט יממש מערכת לחילוץ מידע חזותי-אינרציאלי (visual inertial) מהחיישנים במטע"ד קטן, תחת הדרישות הבאו:

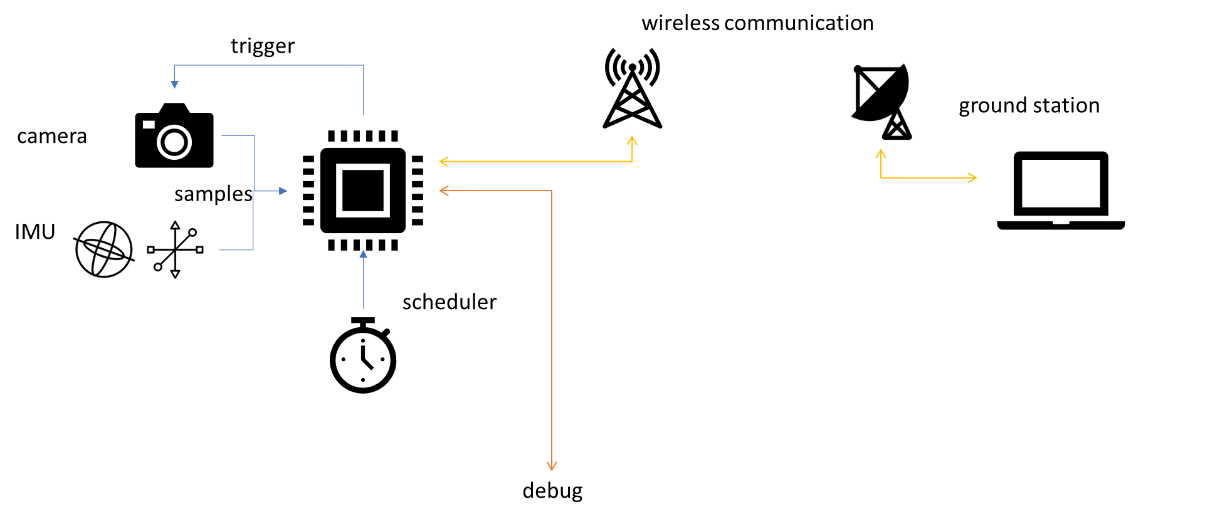
1. גודל קטן
2. צריכת זרם מינימלית
3. תקשורת באמצעים סטנדרטיים לתחנת קרקע – Wifi, BT, 900MHz dongles etc.
4. שליחת המידע עם חותמות זמן ברזולוציה של כ 1msec
   1. מדיד אינרצילי IMU ב6 צירים - תאוצה קווית בצירים x, y, z, תאוצה זוויתית בצירים x, y, z
   2. תמונה ממצלמה (Grayscale / RGB) – בקצב 20FPS
   3. המידע יישלח בשיהוי של כ 150msec
5. אפשרות להריץ את תחנת הקרקע ע"ג PC עם מערכת הפעלה Ubuntu 18.04
6. שמירת המידע בתיקיה ייעודית על מנת לחבר אח"כ את תוצרי המערכת לסביבת ROS

# איפיון דרישות המערכת

## בחירת הרכיבים

עלינו לבחור אריכטקטורה שתאפשר מימוש גמיש ויעיל מבחינת משאבים של דרישות הפרויקט.  
המערכת תורכב מתחנת קרקע, וממטע"ד שיותקן על הרחפן.  
בצד הבית לרשותנו מחשב PC, עם תקשורות אלחוטיות סטנדרטיות (Bluetooth, Wifi). אילוץ התקשורת הנחה בבחירת הרכיבים למטע"ד.

למטע"ד נבחר יחידת עיבוד מרכזית (CPU) שתנהל את לוגיקת דגימת החיישנים – מצלמה, מדיד אינרציאלי, תתזמן את דגימתם ותנהל את לוגיקת התקשורת האלחוטית.  
הדרישה ליחידות המטע"ד הינה שימוש ברכיבים סטנדרטיים מבחינת Time To Market, היתכנות מענה לאילוצי המערכת, יכולות debug גבוהות ותמיכה בקהילות פיתוח עם עדיפות לתשתיות קוד-פתוח.



## סקר שוק

טרם תהליך הפיתוח ערכתי סקר שוק על תשתיות ומוצרים שיתנו מענה לדרישות הפרויקט, אסכם כאן את החומרות שמצאתי ואת יתרונן וחסרונן למימוש המשימה.

לסיכום, נבחרה חומרת OpenMV H7, שאפרט את תהליך התאמת החומרה למשימה בהמשך.

|  |  |  |  |
| --- | --- | --- | --- |
| model | features | Pros | Cons |
| Bitcraze Crazyflie 2.1 | Small scale drone hardware with wireless communication and programming capabilities. Contains IMU on board.  STM32F405 MCU (Cortex-M4, 168MHz, 192kb SRAM) | Small scale factor, open-source community | Not exporting I/F for external camera.  The Bluetooth comm might be a little bit slow for our application. |
| 3DR Pixhawk | 3 IMU on board, wireless communication is based on ESP32 module (Wifi)  STM32F427 MCU (Cortex-M4, 180MHz, 256kb SRAM) | The hardware is versatile, open-source repo is well maintained | No JTAG interface for real time debug  No dedicated digital camera I/F.  Not compact. |
| Beagle Bone Blue | Board conaining an cortex a8, comm interfaces alongside Wifi & BT  OSD3358 (TI Cortex-A8 512MB DDR).  No IMU on board. | Versatile, open-source, heavy lifter.  No dedicated I/F for camera – a USB camera can be used with proper drivers | The use of an embedded Linux OS might be a little |
| ESP32 Camera Module Development Board (OV2640) | An ESP8266 based board (160MHz, 160KB RAM)  Has Wifi integrated!  But the debug interfaces might be difficult! | Open-source, the wifi is built into the MCU. | No dedicated camera port, utilizing an SPI might be slow for this application.  The use of SPI camera might be a limitation for future design.  Debug interface is not convenient for real-time debug |
| OpenMV H7 | Open-source small scale factor hardware and firmware. A product dedicated for image processing applications. User can utilize the hardware as the product is built as shileds (IMU, Wireless communication, Camera) | Contains versatile hardware and cameras.  Well maintained user forums and repo | Not all MCU’s GPIOs are available to the user via standard headers. |

## מצלמה

בחרנו מצלמת CMOS עם חשיפה של Global Shutter, על מנת להבטיח חשיפה אחידה של כל שורות הפיקסלים. השיקול נבע מהתאמת המצלמה לרחפן - פלטפורמה שזזה במהירות, והימנעות ממריחות בתמונה בזמן החשיפה, כמו הדוגמא בתמונה הבאה:

Rolling Shutter vs. Global Shutter
https://andor.oxinst.com/learning/view/article/rolling-and-global-shutter

Figure 1: rolling vs. global shutter

למצלמה יש חיישן בגודל 1/3inch עם רזולוציה מקסימלית של עד 752x480 פיקסלים, יכולת להפיק 60fps ברזלוציה מקסימלית, עם רזולציה של 10bit לפיקסל בגווני אפור. כמו כן, המצלמה מנגישה רגל strobe על מנת לאפשר לבקר לתזמן את רגע לכידת התמונה ובכך להבטיח את עדכניות התמונה.

המצלמה מאפשר אפשרות לחשיפה אוטומטית, וזה המצב שבו עבדנו, עם קצב שעון מקסימלי של 27MHz.

נפחי פלט התמונה ישמשו אותנו בבחירת תצורת המערכת, מבחינת רזולוציה וfps מובאים להלן:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Resolution | width | height | fps | Mbit/sec |
| Full resolution | 752 | 480 | 20 | 57.8 |
| QVGA | 320 | 240 | 20 | 12.3 |

## חיישן IMU

נבחר רכיב לדגימת תאוצות ב6 צירים (3 רכיבי תאוצה בצירים ליניאריים ו3 רכיבי תאוצה בכיוון סיבובי). רכיבי התאוצה יאפשרו לאלגוריתם העיבוד על תחנת הקרקע, להבין תנועות יחסיות של הרחפן, ולטייב את עקיבת התמונה.

רזולוציות דגימה אופייניות של הרחפן תהיה כ250dps לתאוצה סיבובות, וכ+/-2g לתאוצה קווית עבור כל ציר.

## המודול האלחוטי

על המודול האלחוטי להיות בעל פרוטוקול תקשורת סטנדרטי לPC (Bluetooh /Wifi), עם ממשק תקשורת סריאלי לבקר המנהל.

נבחר לממש את התקשורת ע"ב Wifi בגלל יתרונות רוחב הסרט, פשטות החיבור אליו והיכולת העתידית לאגור מידע ממספר מטע"דים בו-זמנית ע"י התחברות של מספר מטע"דים לאותו Access Point.

המודול האלחוטי צריך להתחבר לתחנה קרקעית ע"ג PC, לקבל ולשלוח את המידע מצד המטע"ד.

## הבקר

לבקר יש מספר משימות באפליקציה שלנו:

* ניהול החיישנים והגדרותיהם
* סנכרון זמן דגימת החיישנים
* ניהול פרוטוקול תקשורת שמוצמדת אליו חותמת זמן
* ניהול תקשורת אלחוטית עם תחנת הקרקע
* ריכוז המידע מהחיישנים, ודחיסתו
* קיום ממשקי תקשורת ס ממשקי תקשורת SPI/ I2C לחיישנים, UART לצורך הודעות debug למשתמש
* ממשק JTAG/SWD לצריבה ודיבוג של התכנה בזמן אמת
* יש יתרון לבקר עם ממשק תקשורת סטנדרטי למצלמה, דוגמת LVDS או ממשק מקבילי

לבקר צריכים להיות ממשקי תקשורת שמתאימים לרכיבים שנבחרו, ויכולת לתת חותמת זמן לכל מידע שמגיע מהחיישנים, לטובת עיבוד המשך בתחנת הקרקע.

מכיוון שמערכת הפעלה מוסיפה עוד תקורה בנפח הקוד, בריצה ובזמן ניהול האירועים שלה, ועל מנת לאפשר פיתוח מהיר ואמין, עם מעטפת ביצועים real time, נבחר לממש את האפליקציה ללא מערכת הפעלה, אלא בכתיבה של התכנה bare-metal לבקר.

## חיבורים ואספקות מתחים

למערכת צריכה להיות אפשרות להיות מוזנת ממתחים סטנדרטיים, לדוגמת סוללת Lipo 3.7V, או מחיבור USB של 5V.

## סקירת רכיבי הפרויקט

מודול הOpenMV H7 נותן לנו מענה חומרתי לרוב דרישות הפרויקט, ובפרק הבא נסקור את השינויים שבוצעו במודולים הקיימים על מנת לתת מענה מיטבי לפרויקט.

כל רכיבי המערכת עמידים לעבודה בטמפרטורת חדר, ( -20 to +70 °C). ניתן לראות גם לפי התמונות שהם בנויים בצורת shields שיאפשרו ממשק נוח לחיבור בין החלקים השונים.

לנ"ל שידכנו צורב מדגם stlink v3 של חברת STmicro – שנתן לנו ממשק JTAG וחיבור UART למטרות דיבאג במערכת.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Weight | Size | Description | Model | Part |
| 19gr | 45x36x30mm  Incl. camera | Implements micropython, includes DCMI I/F, RGB LED, up to 480MHz  Idd = 170mA @ 3.3V  Regulator – up to 5.5v, 1A | STmicro STM32H743 | MCU  OpenMV Cam H7 |
| 10gr | 35x25x28mm | HFOV = 70.8°  VFOV = 55.6°  Idd = 200mA max | ON MT9V034 | Camera  Global Shutter Camera Module |
| ? | 20x25x7mm | Idd <4mA | Invensense MPU6050 | IMU |
| 7g | 36x27x20mm | Up to 12.5Mbps  Idd =100 mA @ 3.3V | Microchip ATwinc1500 | Wireless Communication  WiFi Shield Angle |

# תכן מערכת

נציג את בחירת הפרמטרים ונקודות העבודה לכל מודול כמו גם הממשקים בין הרכיבים השונים שמביאים נקודת עבודה אופטימלית לבעית הפרויקט.

## תרשים בלוקים:

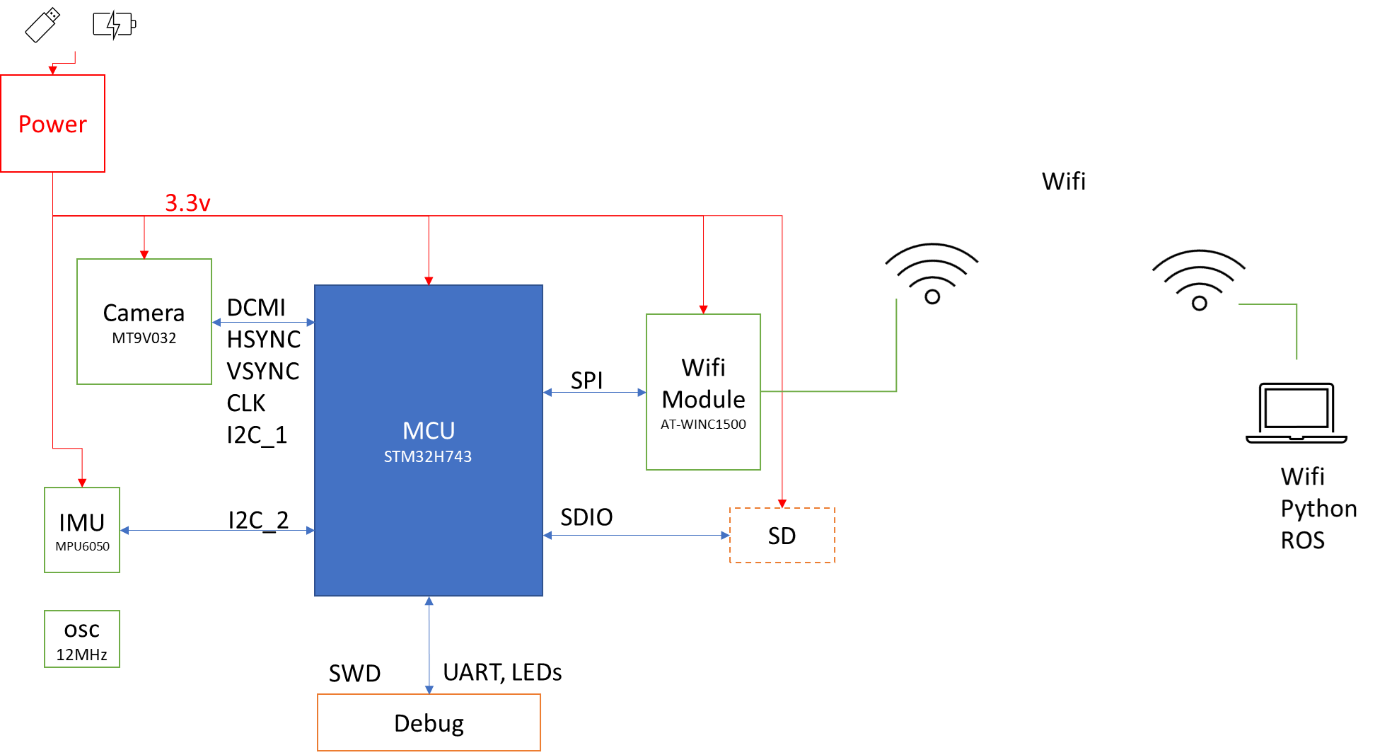


Figure 2:block diagram

יחידת אספקת המתחים מייצבת מתח 3.3V לצרכנים במעגל. ניתן להזין את המעגל באמצעות USB או סוללה חיצונית.

הבקר מקבל שעון ממתנד חיצוני של 12MHz לצורך הריצה שלו. כמו כן, הבקר מתחבר דרך ממשקי תקשורת שונים לצרכנים במעגל-

למודול הWifi ממומש חיבור תקשורת SPI תקני (MISO, MOSI, CLK, CS).

למודול הIMU מתחברים באמצעות I2C/

הממשק למצלמה כולל bus - ממשק מקבילי ברוחב 8 ביט, שנקרא DCMI, כמו גם קווי סינכרון לתחילת קליטה של שורה/עמודה חדשה, אות שעון למצלמה, וממשק I2C לקינפוג ושינוי פרמטרים במצלמה.

## הבקר - STmicro STM32H743

הבקר משמש אותנו להגדרת הפריפריות במעגל לנק' העבודה הרלוונטית, התקשורת ביניהן, תזמון האירועים, ניהול הפרוטוקול וניהול של המידע הנאגר.

לבקר יש אילוצי זכרון RAM, כאשר בהתמודדות עם תפיסה של מטריצות פיקסלים גדולות, הופכים להיות מגבלת מערכת שיש להתמודד איתה בהיבט האפליקציה. כמו כן מניתוח של בעית זמן השליחה, רוחב הסרט הנתון והקצבים שהתבקשנו לעמוד בהם, הבנו שעלינו לממש דחיסה לתמונה, לכן מימשנו כאן דוחס jpeg כפי שיפורט בפרקי המימוש בהמשך.

תכן הלוגיקה והעיבוד של הבקר, צריכים להתכנס מבחינת ציר הזמן לצילום, דגימה, עיבוד ושליחה לתחנה הקרקעית – על מנת לעמוד בדרישות הפרויקט של צילום ושליחה בreal time.  
לצד כל חלק נסקור את הזמן האופייני שלוקח לכל חלק ונחשב את תקציב הזמנים.  
 - משך הזמן לחשיפת חיישן המצלמה והעברת התמונה לבקר  
- משך הזמן לדחיסת התמונה  
- משך הזמן לקבלת מידע מהחיישן האינגרציאלי  
- משך הזמן לשליחת המידע

להלן דיאגרמת בלוקים של פריפריות הבקר, גישות הזכרון והממשקים האפשריים לBSP.

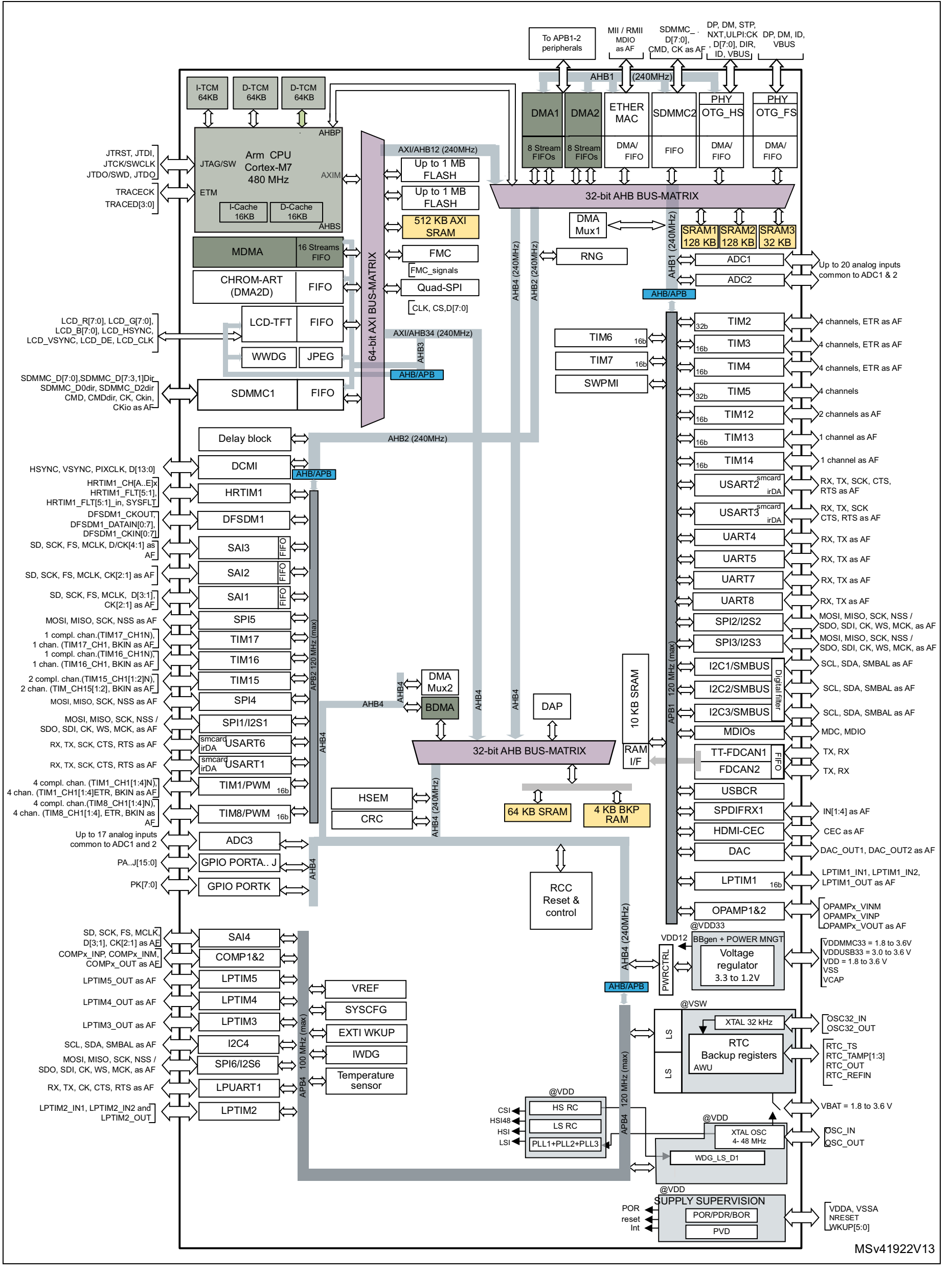


Figure 3: STM32H743 block diagram

### BSP והקצאת GPIOs – Board Support Package

להלן הpinout שבחרנו למימוש הבקר ולהקצאת ה GPIO – General Purpose Input/Output.

הממשקים החשמליים (אספקות ואדמות)- הם אילוץ במימוש, ושאר ה pinout אליו צריכים להתכנס אליו לפי שרטוטי הschematics של חומרת ה OpenMV, מפורט בתמונה הבאה.

כמו כן,התוספת של חיישן הIMU דרך חיבורי הheader, אילצה אותנו לפרוס קווים נוספים לטובת UART, באמצעות הלחמה של maget wire ע"ג רגלי הבקר, באופן שיאפשר שימוש במקביל גם בממשק הdebug של ה UART וגם על מנת לתקשר עם רכיב הIMU. הלחמת קווי התקשורת בוצעה על רגלי הבקר PA0 (22), PA1 (23, בנוסף לחיבור קו אדמה לטובת התקשורת.

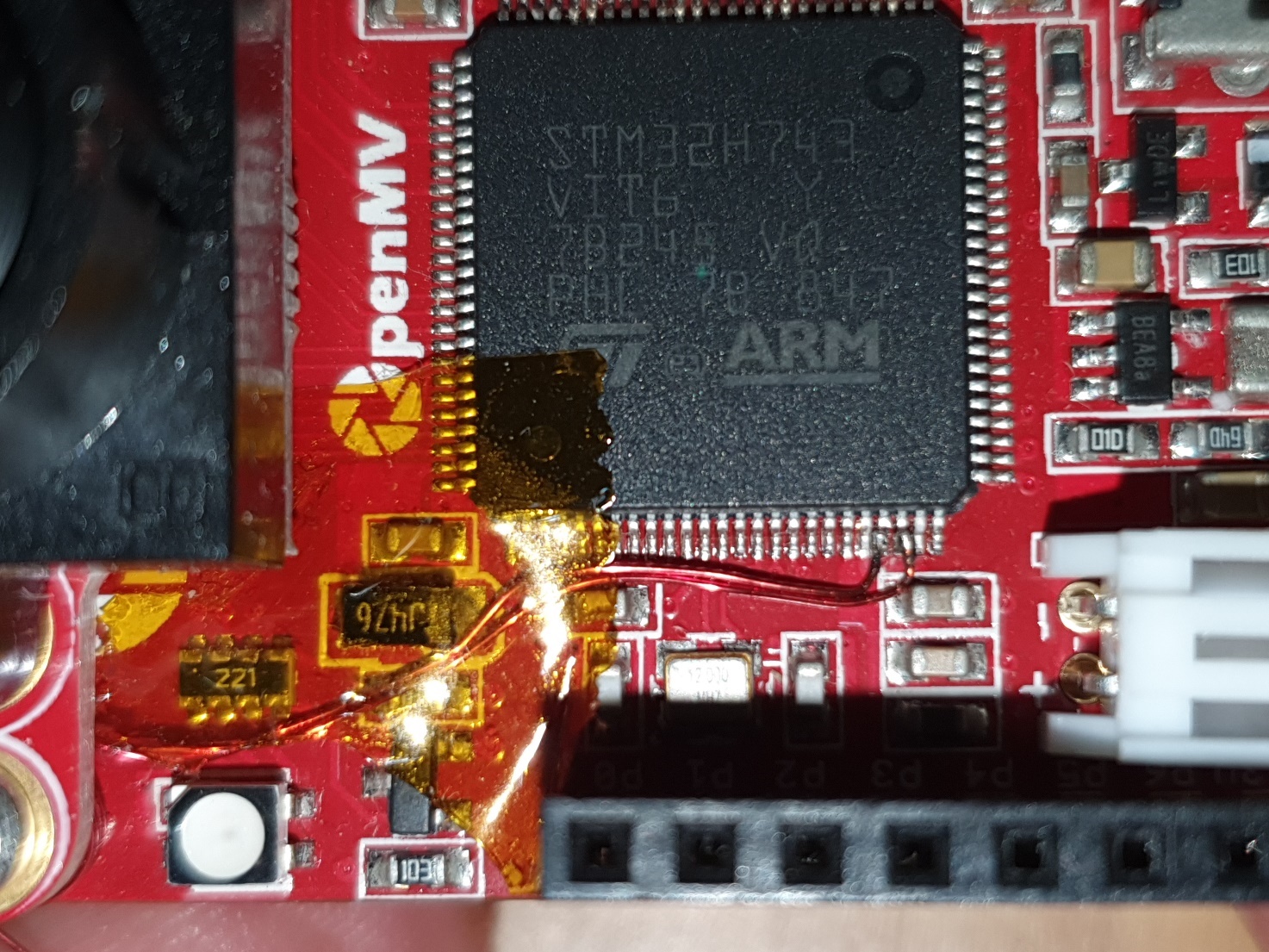


Figure 4: Debug UART wiring

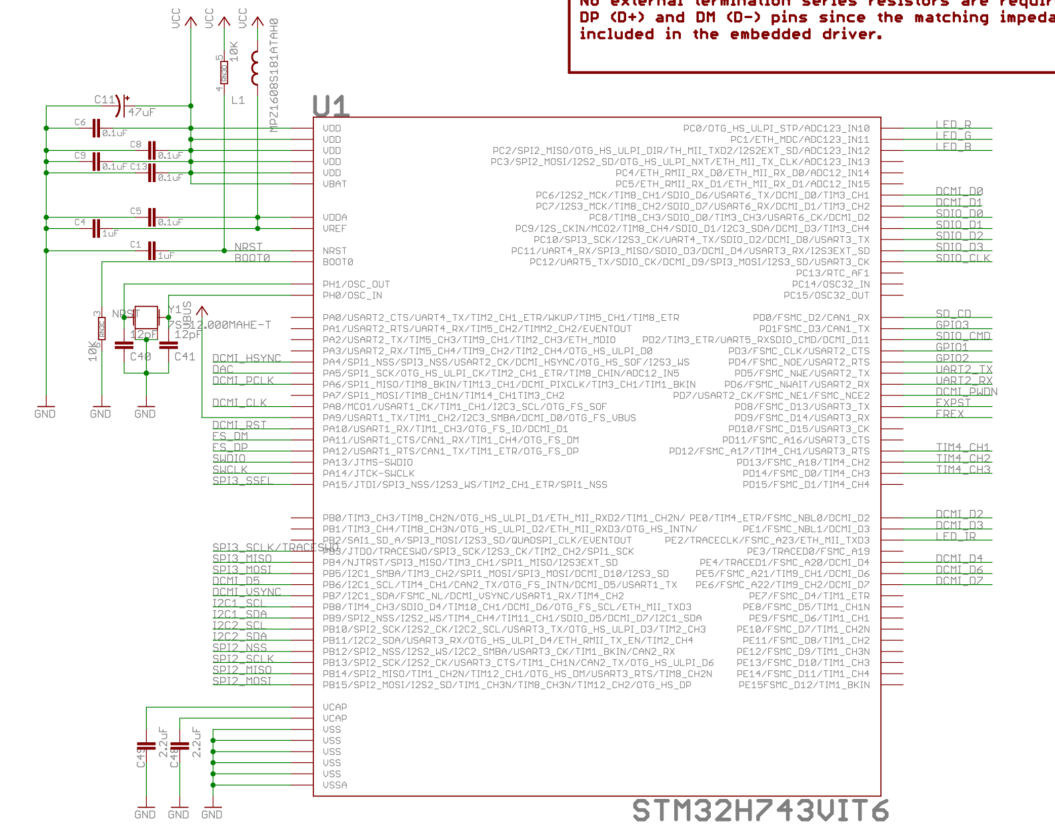


Figure 5: OpenMV H7 (c) schematics

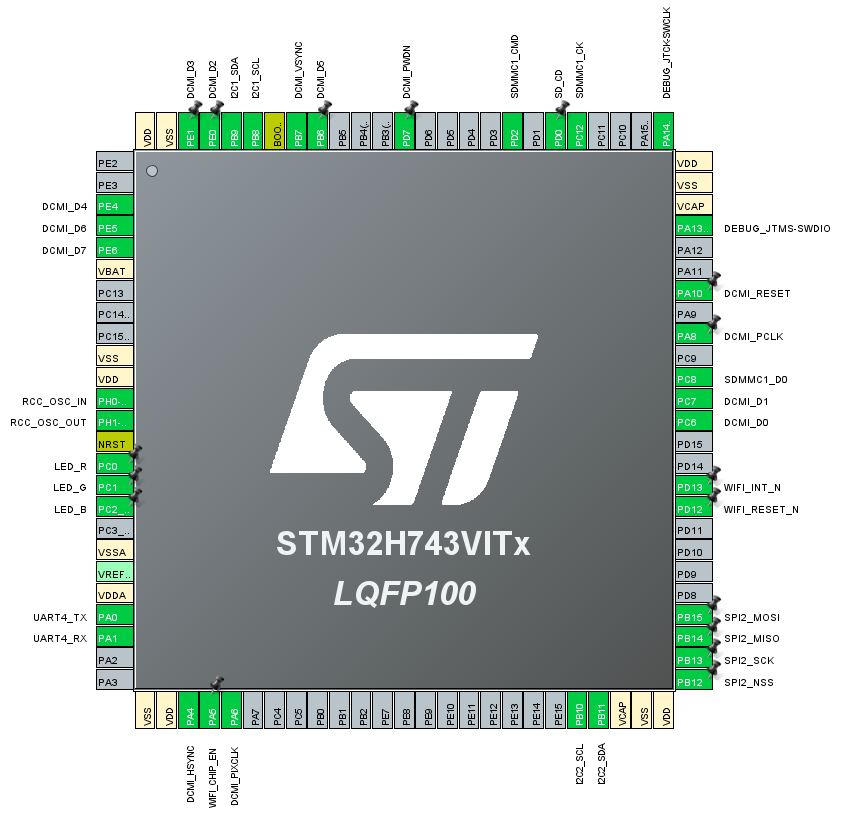


Figure 6: STM32H7 pinout

### מפת חלוקת שעון מרכזי

באמצעות סביבת הCubeMx הגדרנו את חלוקת השעונים לחלקי הבקר. בגלל סיבות שיפורטו בהמשך, השתמשנו במכפילי שעון ומחלקים כדי לעבוד בקצב העבודה המקסימלי שהבקר מאפשר – 480MHz.

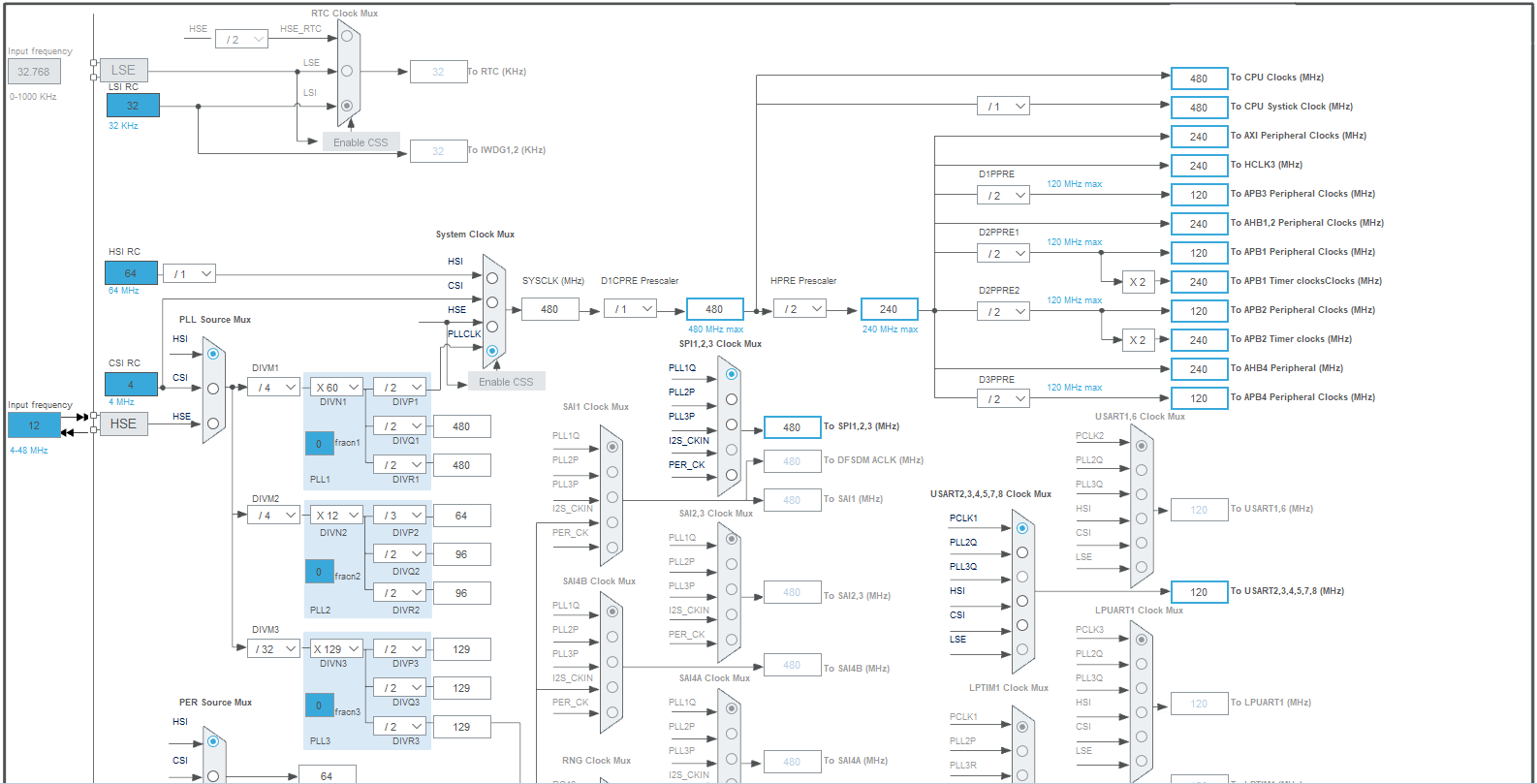


Figure 7: CPU clocks regime

### פריפריות בשימוש

להלן ניצול משאבי הבקר

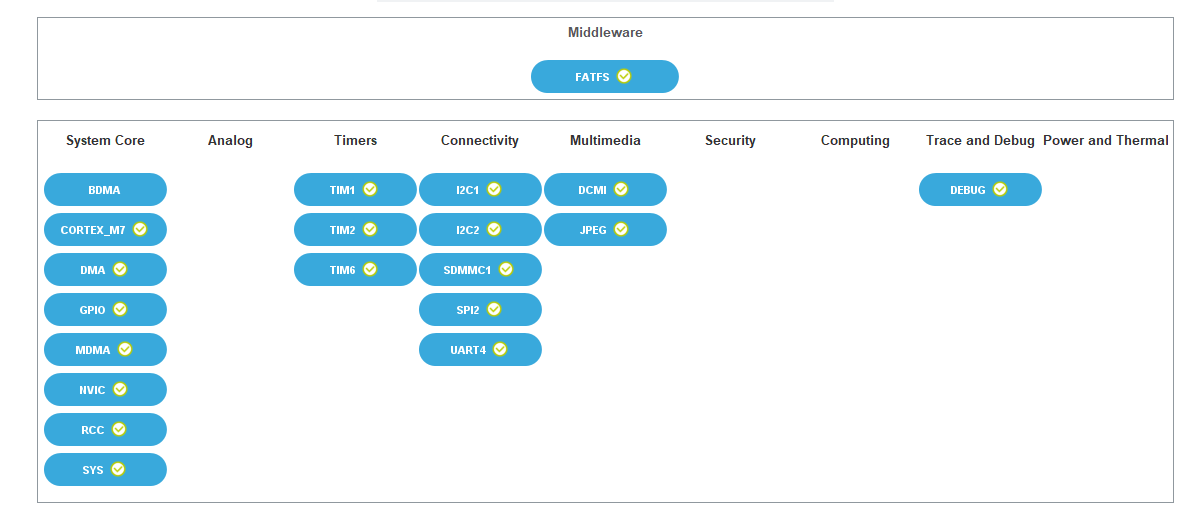


Figure 8 : CPU peripherals

#### DMA/ MDMA

זוהי פריפרייה שמאפשרת גישת זכרון ישירה בין אזורי זכרון שונים בבקר, עם העברת מידע בכלי חומרתי, שאינו מעכב את ריצת התכנה. באפליקציה שלנו - מוגדר כגישה ישירה בין זכרון פריפריית תפיסת המצלמה (DCMI) לזכרון הRAM, כמו כן בין הRAM לפריפריית דוחס התמונות.

היכולת הזו מאפשרת לבקר לתקשר במהירות וביעילות ולקבל מידע מהפרירפיות השונות שמפורטות כאן, מבלי לעכב באופן משמעותי את התכנה.  
לדוגמא – דחיסת התמונה לפורמט JPEG ממומשת אצלנו ע"י העברת 2 שורות בכל פעם לפריפריית הדחיסה חומרתית והמתנה לביצוע עיבוד הפריפרייה. כאשר הפריפרייה מסיימת את החישוב והדחיסה על 2 השורות, היא מעבירה באמצעות הDMA את המידע למרחב הזכרון בRAM (Random Access Memory) שהוגדר בבקר להמשך העבודה, הבקר מקבל פסיקה (interrupt) ובכך מקבל טריגר להמשיך להעביר את נתח התמונה הבא לעיבוד הפריפרייה החומרתית וכך חוזר חלילה, עד לסיום הדחיסה.

עוד שימוש לDMA אצלנו, הוא להעברת המידע מהמצלמה לזכרון הבקר.

#### Cache and MPU

לטובת תקשורת והעברת מידע בין אזורי זכרון שונים, הוגדרה גישה מותרת לאזורי זכרון בהתאם להגדרות הDCMI והJPEG לפי אילוצי הבקר.

#### JPEG Encoder

כפי שתואר בפרקים הקודמים, דרישות הפרויקט שמבקשות תמונה ברזולוציה טובה דיה (נניח 320x240), בקצב של 20fps, מאלצת רוחב סרט גדול מאוד של כ~12Mbps.  
מאוחר יותר, התברר שהרוחב סרט האפקטיבי שמצליחים להוציא מרכיב התקשורת האלחוטית הוא כ3Mbps, עובדה שהפכה את השימוש בדוחס להכרח.

הפרמטרים לדוחס התמונה מאפשרים שימוש במרחב צבעים שונה   
להלן דוגמא של תמונה שנדחסה באמצעות פריפריית הדוחס החומרתית שבבקר, תחת אילוצי השימוש שבה:

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| JPEG\_RGB\_FORMAT | JPEG\_RGB888 |
| JPEG\_CHROMA\_SAMPLING | JPEG\_420\_SUBSAMPLING |
| JPEG\_YCBCR\_COLORSPACE | JPEG\_COLOR\_SPACE |
| JPEG\_IMAGE\_QUALITY | variable |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| W | 400 | 400 | 400 |
| H | 240 | 240 | 240 |
| Size | 281KB | 10.8KB | 64.2KB |
| Q | - | 75 | 100 |
| Size factor | 1 | 0.0384 | 0.228 |

ניתן לראות שאפשר להקטין את התמונה, מבלי לאבד איכות באופן ניכר, בפקטור ~0.04 ביחס לנפח המקורי.  
הקטנת נפח התמונה, תאפשר להעביר תמונה בlatency נמוך ומבלי לפתוח הרבה נפח.

מבחינת לוח הזמנים לעיבוד, עבור גודל תמונה 640x480, לפי דפי הנתונים יש לקחת בתקציב הזמנים.

להלן תיאור תהליך דחיסת התמונה בפריפרייה החומרתית שבבקר:

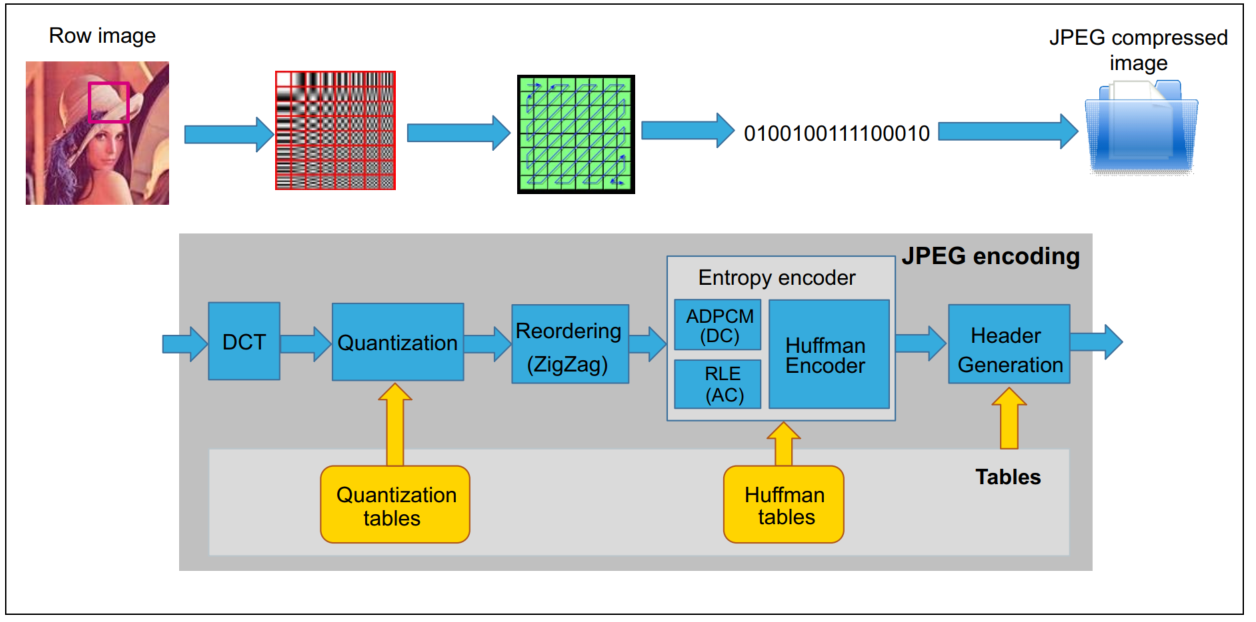


Figure 9: JPEG encoding flow

מהלך הדחיסה בנוי מקבלת מפת ביטים שלמה של התמונה (לדוג' 320\*240 פיקסלים), ולאחר מכן קריצה של חלקים מהתמונה והעברתם לדוחס – 16 שורות בכל פעם, המרתם למרחב צבעים YCbCr וליחידות המרה מינימליות (MCU) לטובת דחיסה (DCT, קוונטיזציה, קידוד אנטרופיה, קידוד Huffman), וייצוג קובץ הjpeg המתאים.

תהליך הקריצה, הגדרת הפרמטרים לדוחס, וטיפול הפלט הדוחס הם בצד התכנה. תהליך הדחיסה עצמו וההמרה מתבצע באופן חומרתי.

#### Timers

TIM1 – שעון המצלמה,במצב עבודה של OC (מייחצן אות שעון ע"ג רגל מוגדרת בבקר, כקלט למצלמה) לתדר 25MHz.

TIM2 – שעון זמן הדגימה לחיישן הIMU - מכוון זמן מחזור 1msec (1kHz).

TIM6 – שעון זמן ניהול השליחה למודול הWifi - מכוון זמן מחזור 0.25msec (4kHz).

#### UART

משמש אותנו למטרת תקשורת סריאלית בין המחשב ול debug של המערכת - ממשק הודעות טקסטואליות שמוצמת לחותמת זמן לטובת חקר ביצועים של המערכת. מוגדר להיות בקצב 115,200bps. ממשק הdebug יומחש בפרק ביצועי המערכת.

#### SPI

SPI2 – למודול הWifi, בקצב 60MHz[הערה: הקצב הזה חורג מהקצב המומלץ של היצרן, שהוא עד 48MHz ]

#### DCMI – Digital Camera Interface

הבקר מכיל ממשק למצלמה על מנת לתקשר עם מצלמות חיישני CMOS באמצעות bus ברוחב 8-14bit על מנת לקבל וידאו. הפריפריה מאחדת את הממשק והפרוטוקל לתזמון הפיסלים ביחס לאות השעון PIXCLK שיוצא מהחיישן, כמו גם סנכרון לאותות סימון השורות-עמודות של הframe, שקרויים VSYNC ו – HSYNC בהתאמה.

הממשק למצלמה באופן כללי, מורכב מ:

* קווי שליטה למצלמה (טריגרגים, reset)
* קווי מידע ותקשורת מחיישן המצלמה (המידע מחיישן הADC שדוגם את הפיקסלים). הסבר על הפרוטוקול מפורט בפרק המצלמה.
* קווי תקשורת להגדרת פרמטרים לעבודת המצלמה (זמן חשיפה, רזולוציה)

אצלנו הממשק מוגדר ברוחב 8bit, עם סנכרון מצד הבקר. עם מימוש העברת הframe באמצעות DMA2.

השימוש בDMA מאפשר לבקר להעביר את המידע מפריפריית הDCMI למרחב הזכרון שמבצע עיבוד על התמונה.  
פריפריית הDCMI נמצאת שייך לAHB2, יחד עם DMA1 + DMA2. על מנת להעביר את המידע לאזור הזכרון בו רצה האפליקציה ב – AXI-SRAM, עלינו להגדיר MPU (Memory Protection Unit) עם הרשאה לגישה בין אזורי הזכרון הנ"ל.  
זמן החשיפה האופייני של מצלמה (~16msec) וזמן השליפה של המידע מהמצלמה הוא זניח  
ומסתכם בכ-.

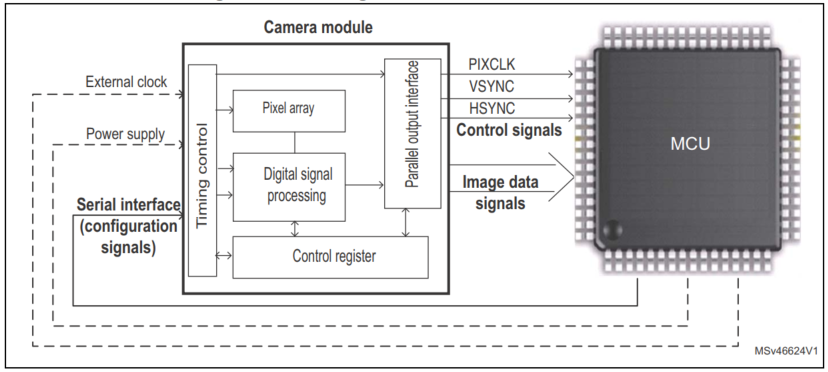


Figure 10: MCU-Camera DCMI I/F

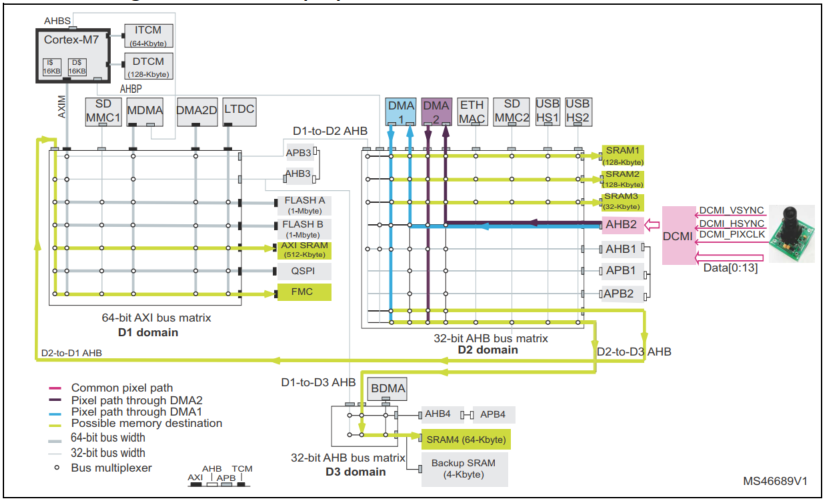


Figure 11: DCMI memory access

פלט התמונה שנשמר מהמצלמה, הינו מפת ביטים (bitmap), לפי גודל התמונה שהוגדר, כאשר כל פיקסל בגוון אפור בגודל 8bit לפיקסל. התמונה תישמר בבאפר במרחב הAXI\_SRAM שיוקצה מראש לגודל התמונה הרצוי.  
המצלמה אצלנו הוגדרה לרזולוציית תמונה של 320X240 פיקסלים.

#### I2C

I2C1 – עבור מודול המצלמה, בקצב 400KHz

I2C2 – לחיישן הIMU, בקצב 400KHz

#### SD + FATFS

למטרות debug וחקר ביצועים, הושמש ממשק לשמירת פלטי המצלמה והדחיסה ע"ג כרטיס SD במערכת קבצים FATFS. השימוש בפיצ'ר הנ"ל אפשרי באמצעות הדלקת דגל בקימפול הקוד – בקובץ camera\_mngr.h - אפשור הdefine של SAVE\_INPUT\_IMG\_ON\_SD ו - SAVE\_OUTPUT\_IMG\_ON\_SD יאפשרו את שמירת תמונה הקלט והפלט בהתאמה על כרטיס הSD. לשים לב שהפעולות הנ"ל יכולות להשפיע על קצב עיבוד ושליחת הנתונים, לכן יש לעדכנם בהתאם.  
כמו כן, יש לפתוח את פלט הbmp שנשמר באמצעות סקריפט קצר ב raw\_2plot.py .

## מצלמה – ON Semiconductor MT9V034

המצלמה שבשימוש אצלנו היא מצלמת Global Shutter עם חיישן CMOS בגודל 1/3inch. כמו כן, תת הדגם שאנחנו עובדים איתו מכיל רזולוציה של עד 752x 480 בגווני אפור עם רזולציה של 10bit דגימת ADC לפיקסל.

ממש המצלמה מובנה באמצעות פריסת המעגל של OpenMV וממומש לפי הדרישה בדפי הנתונים.

רגלי הSCLK/SDATA מחווטות לרגלי I2C1 של הבקר.  
רגלי הSYSCLK מחווטת לאות שעון שהבקר מייצר (בתדר 25MHz), כאשר המצלמה תומכת באות שעון עד 27MHz.

ממשק הExposure מאפשר שליטה על חשיפת החיישן של המצלמה, אולם לא מימשנו אותו כאן במערכת (ניתן לממש אותו ע"י הגדרת רגל PB4 בבקר וניהול החשיפה ביחס לתפיסת התמונה.

רזולוצית המצלמה נקבעה מטעמי זכרון וקצבים – להיות על 320x240 (QVGA).

הממשק המקבילי של פלט המצלמה – Dout (9:0) נכנס לרגלי הDCMI שהוגדרו בבקר כאשר מטעמי חומרה ואינטגרציה, ממומשים בפועל רק 8bit של התקשורת, קרי Dout(9:2). פרוטוקול העברת המידע בין החיישן למצלמה מפורט בתמונה בהמשך.

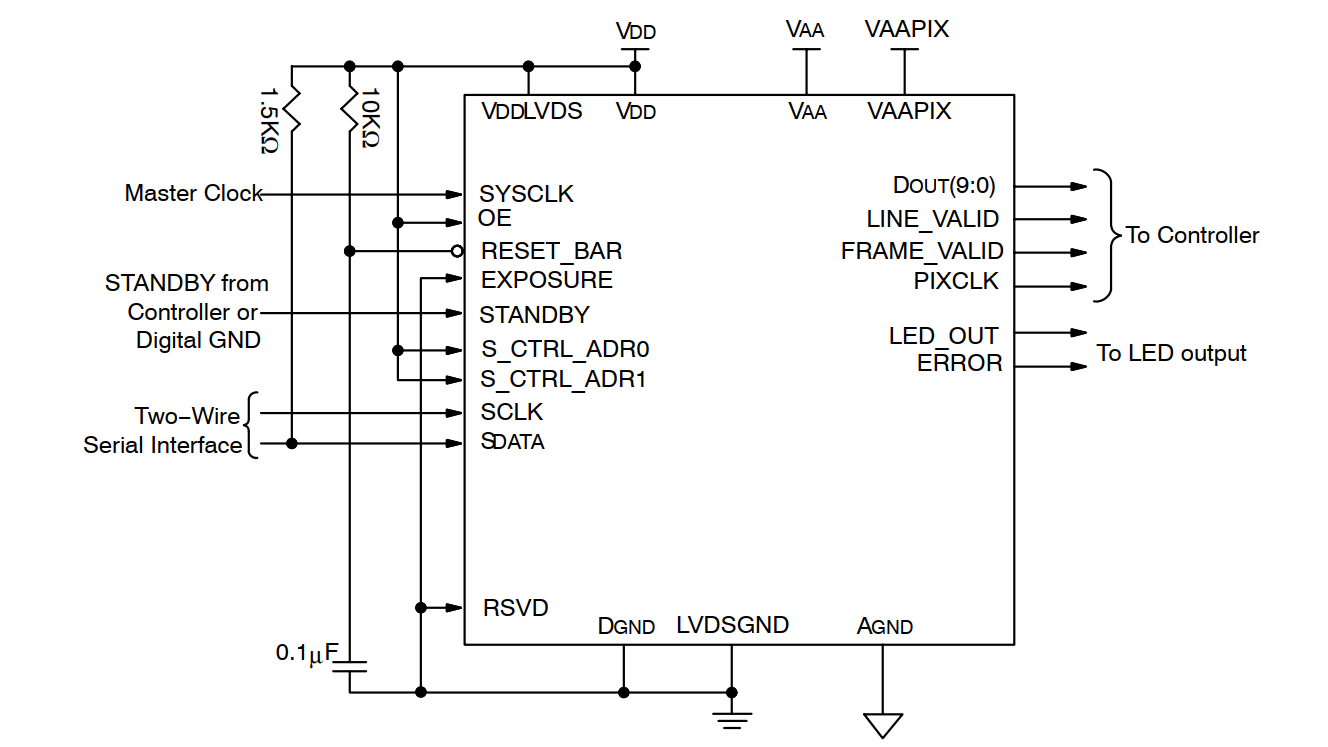


Figure 12: MT9034V pinout

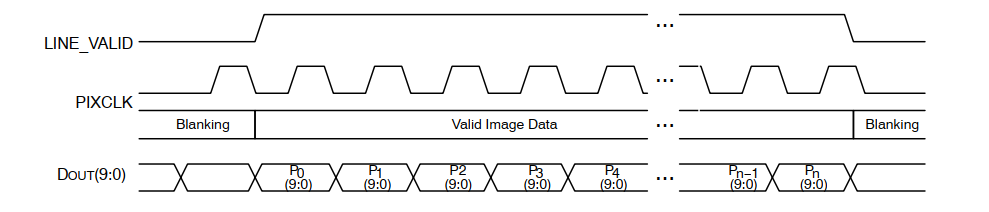


Figure 13 : Pixel data protocol

## מודול הWIFI - Microchip ATSINC1500

מודול הWifi מאפשר לנו לקיים תקשורת עם כל מודם Wifi סטנדרטי, במצבי עבודה של Client (STA) או Host (AP).



Figure 14: ATWinc1500 diagram

|  |  |
| --- | --- |
| Value | Feature |
| 802.11b/g/n | WLAN standard |
| SPI, 28MHz max | Host interface |
| 802.11b: 1, 2, 5.5, 11Mbps  802.11g: 6, 9, 12, 18, 24, 36, 48, 54Mbps | Data rate |
| -76dBm @ 802.11g 48Mbps OFDM | Sensitivity |
| 18.5dBm max @802.11b 11Mbps | Output power |
| Ivbatt = 268mA, Ivddio = 22mA max @802.11g | Power consumption |

#### ביצועי המודול

מהעבודה של השותף שלי לחלק השני של הפרויקט, נמדד כי הביצועים האפקטיביים של המודול הינם כ 3Mbps. עובדה שהפכה את עבודה הדחיסה להכרח, כמו כן, אילצה את גודל התמונה האפקטיבי שאנחנו יכולים לעבוד איתו, להספיק מבחינת תקציב הזמן לדחוס ולשלוח, כאמור 320X240 פיקסלים.

## החיישן האינרציאלי – IMU – Invensense MPU6050

החיישן נועד להקליט תאוצות לינאריות וסיבוביות של הרחפן. החיישן הזה מציג ביצועים טובים מבחינת טווח דגימה וצריכות זרמים. כמו כן, חיפשנו רכיב עם דרייברים ותמיכה ברשת, שאכן הרכיב הזה עומד פרמטר.

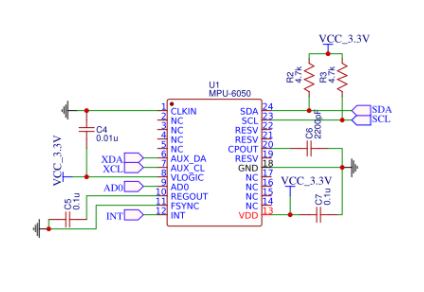


Figure 15: MPU 6050 breakout schematic

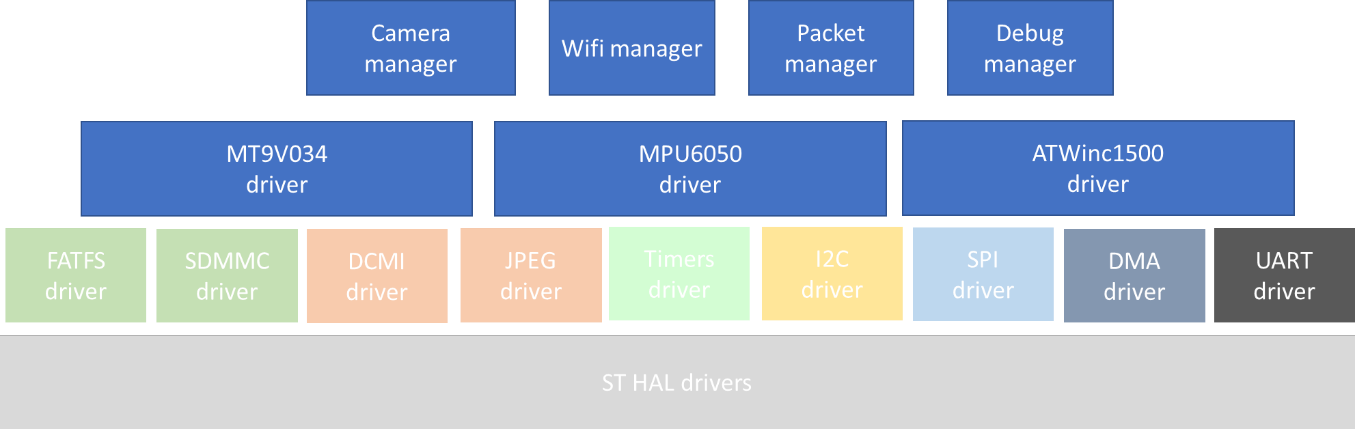
|  |  |
| --- | --- |
| Value | Feature |
| ±250 ±500 ±1000 ±2000 | Gyro fullscale [dps] |
| 131 65.5 32.8 16.4 | Gyro sensitivity [LSB/dps] |
| 8KHz max | Gyro output data rate |
| ±2 ±4 ±8 ±16 | Accelerometer fullscale [g] |
| 16,384 8192 4096 2048 | Accelerometer sensitivity [LSB/g] |
| 1KHz max | Accelerometer output data rate |
| I²C | Digital Output |
| 10,000g | Shock resistant |
| I = 3.8mA, V =2.375v-3.46v | Power consumption |

את החיישן יש להגדיר (config) לאחר עליית המתח לרכיב. יש להגדיר רזולוציה (full scale) עבור gyro ועבור הaccelerometer, כמו כן יש להגדיר את קצב רענון הנתונים – הoutput data rate.  
לאחר מכן, ניתן לקרוא באופן יזום בכל רגע נתון את דגימות החיישן, ולאחסן אותן בבקר לצורך שליחה.

# ארכיטקטורת תוכנה

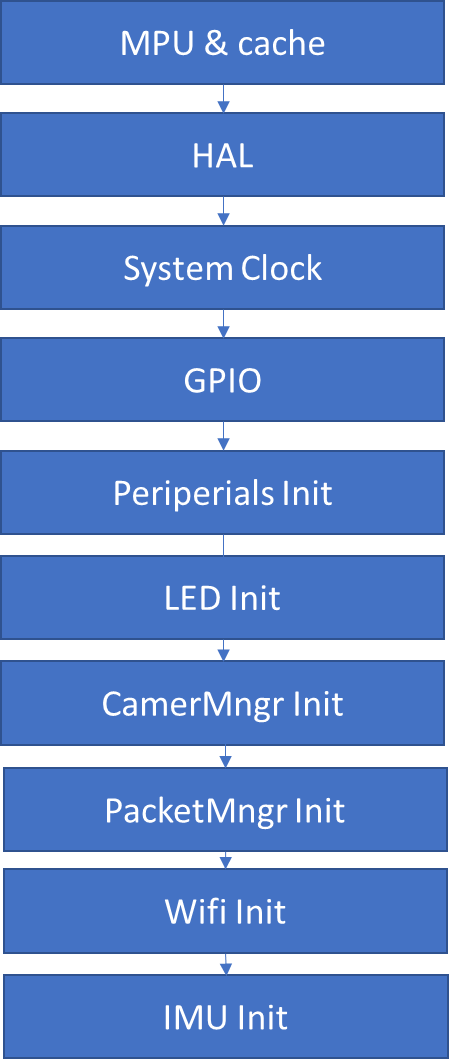
## ארכיטקטורה

### מבנה שכבות הקוד



#### שלב האתחול של הבקר

עליית הקוד צריכה להבטיח הדלקת הפריפריות, הקצאת הזכרונות, הגדרות השעונים ואתחול כל מודול חומרתי לפי הפרמטרים שפורטו בפרק הקודם.  
הקוד מתחיל מהגדרת מרחבי הזכרון בתחילת ריצת הקוד, לאחר מכן אתחול הHAL (Hardware Abstraction Layer - שכבת קוד שחברת STMicro מפרסמת כהפשטה של שכבת החומרה בבקר, לדרייברים תכנתיים פשוטים יותר), הגדרות שעונים, ולאחר מכן אתחול של כל שאר הפריפריות והמודולים שלנו לפי השעונים והזכרונות שהוקצו להם.



#### לוגיקת תפיסת התמונה ודחיסתה

להלן מצורף תרשים הזמן על תפיסת התמונה, דחיסתה והכנתה לשליחה, בכל פרק זמן של אירוע שליחה שמוגדר אחת ל60msec כפי ששותפי לעבודה על המוצר הגדיר בקובץ packet\_mng.h.

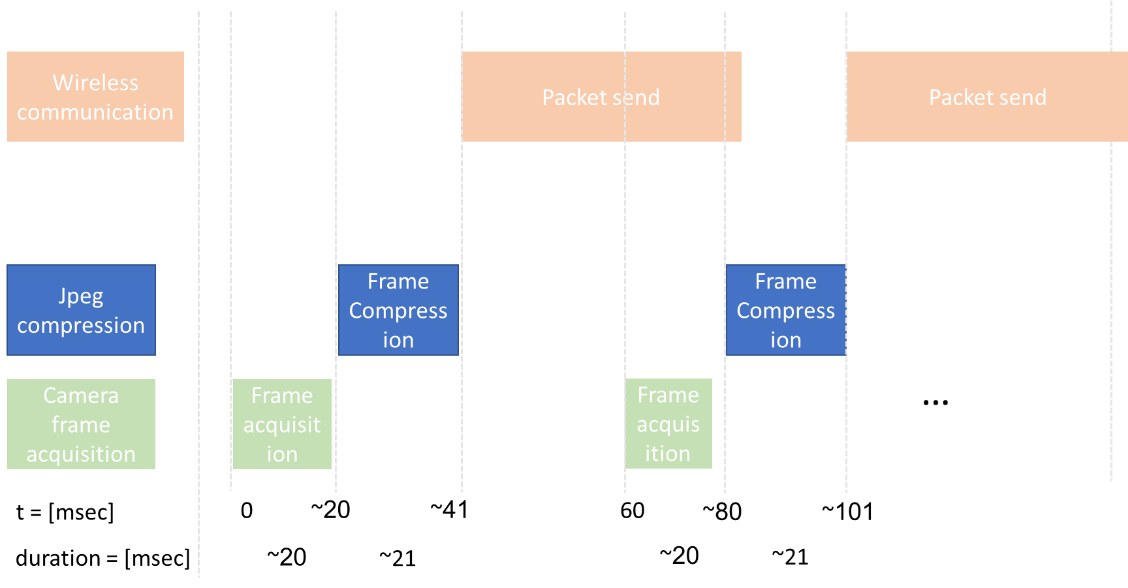


Figure16 : compression timeline

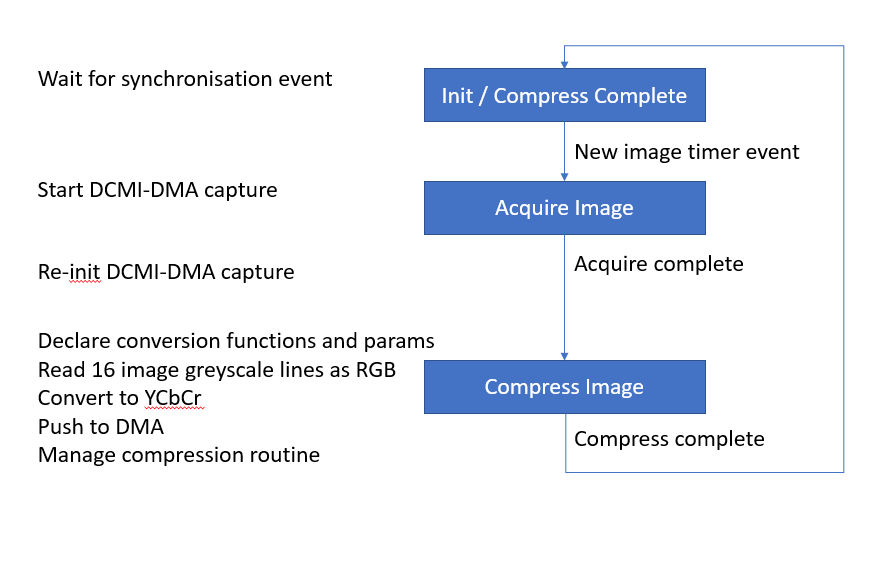


Figure 17: Image handling routine

מובא כאן פלט debug בזמן אמת של תקשורת מול תחנת קרקע. זהו צילום מסך של תכנת Docklight שמנגישה את תקשורת הUART של הבקר.  
ליד כל שורה יש חותמת זמן בmsec מרגע עליית הבקר,ובטקסט את האירועים שקורים.

ניתן לראות שהזמן האופייני לתפיסת התמונה ודחיסתה תלוי במאפייני התמונה וכמה היא עשירה בגרדיאנטים  
זמן אופייני לתפיסת תמונה – עד כ 20msec.  
זמן אופייני לדחיסת תמונה – כ 21msec.

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| SOF | A message with IMAGE data is being sent to ground station |
| acq s | Acquisition of new frame from camera started |
| imu | A message with IMU data is being sent to ground station |
| acq dur X[msec] | Acqusition of frame took X msec |
| comp s | JPEG compression started |
| jout size = X[b], jcmp Y[msec] | JPEG compression took Y msec with output of size X bytes |

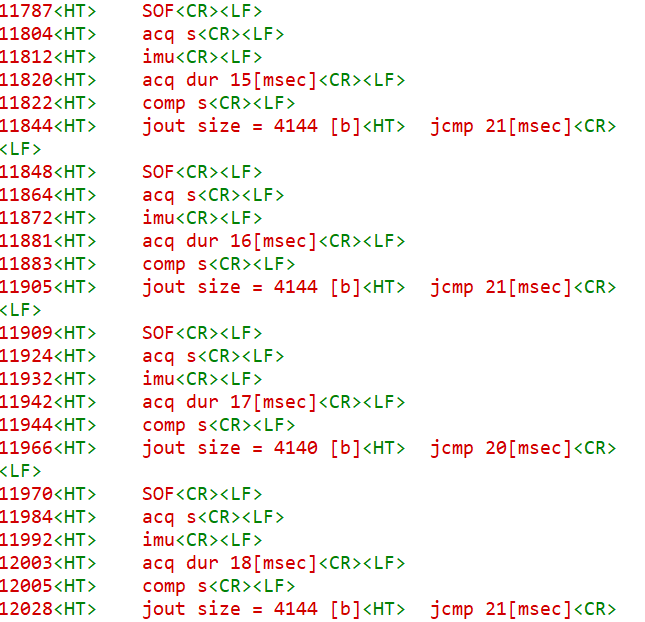


Figure 18: acquisition and compression debug output

## תכנת תחנה קרקעית

לטובת חקר ביצועים ונוחות הפיתוח, פותח כלי להצגת התמונות בזמן אמת כתוספת לתכנת התחנה הקרקאית שפיתח שותפי לפרויקט.

התצוגה מאפשרת לקבל את התמונה העדכנית ומציגה את ההשהיות בקבלת התמונת.

התכנה, בסביבת פייתון מהווה את הניהול והטריגר לתחילת שליחת המידע מהמטע"ד והעברת המידע למחשב.

לציין שבגלל סיבה שאינה ידועה, החלק העליון והתחתון של התמונה יוצא מעוות, לכן בתקנת הקרקע אנו חותכים חלק מהצד העליון והתחתון של התמונה על מנת להביא פריים תקין לתצוגה.

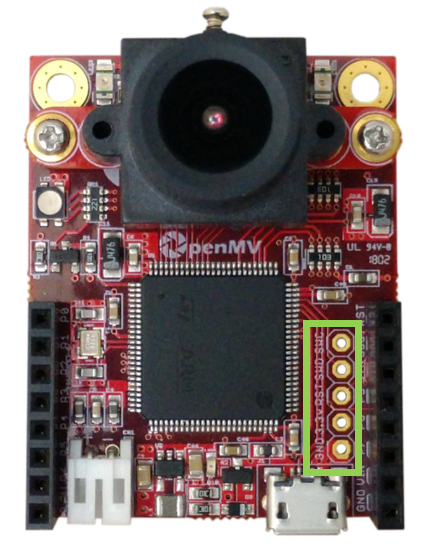


Figure 19: benchmark and payload's live view

## ממשקי DEBUG, צריבה ושימוש במערכת

לצורך פיתוח הקוד והאלגוריתם ולהשמשת המערכת השתמשנו בכלים הבאים:

* הזנת מתח, אפשרית באחת מ2 השיטות הבאות:
  + מתח USB באמצעות מחבר micro-USB על הכרטיס (השיטה המומלצת)
  + מתח 3.7v מסוללת LiPo- באמצעות מחבר ייעודי ע"ג הכרטיס.
* STM32Cube IDE - compiler ו- debugger מבוסס eclipse לבקרי STM32.
* STLINK V3 – צורב לבקר STM32, מאפשר צריבה דרךJTAG או SWD. בפרויקט הזה השתמשנו בצריבה באמצעות SWD, שאפשר צריבה ע"ג 2 רגליים (SWDIO, SWCLK), בheader ייעודי ע"ג הOpenMV. הממשק לצורב מצריך גם מתן קו לוגי גבוה 3.3V, קו אדמה משותף GND, ואצלנו מושמש גם קו RESET חיצוני.



* ממיר תקשורת USB to UART – לצורך הצגת הודעות מצב מהמערכת. החיבור מורכב מקו אדמה משותף GND, רגל Tx מהבקר, ורגל Rx מהבקר (לא מושמש תכנתית ממשק Rx).

|  |  |
| --- | --- |
| Figure : Debug Setup | Figure 21: UART setup wiring |

# תוצאות

## תצורת המדידה

בוצעו מדידות להבנת מעטפת הביצועים של המוצר. מערך המדידה כלל חיבור של מצלמת המט"ד לתחנה קרקעית, הקרנת שעון ע"ג התחנה הקרקעית, ושידורה במסך סמוך. הסטאפ הזה הוקלט.  
לאחר מכן יש תהליך של מיצוי המידע, שהוא מעבר frame-by-frame על הסרטון המוקלט באמצעות תכנה  
דוגמת VLC player, תיעוד השעון של התחנה הקרקעית (הground truth שלנו), ותיעוד תמונה שהמטע"ד שלח.  
באמצעות כך יש לנו וקטור של frames עוקבים של המטע"ד והשיהויים בין כל פריים.

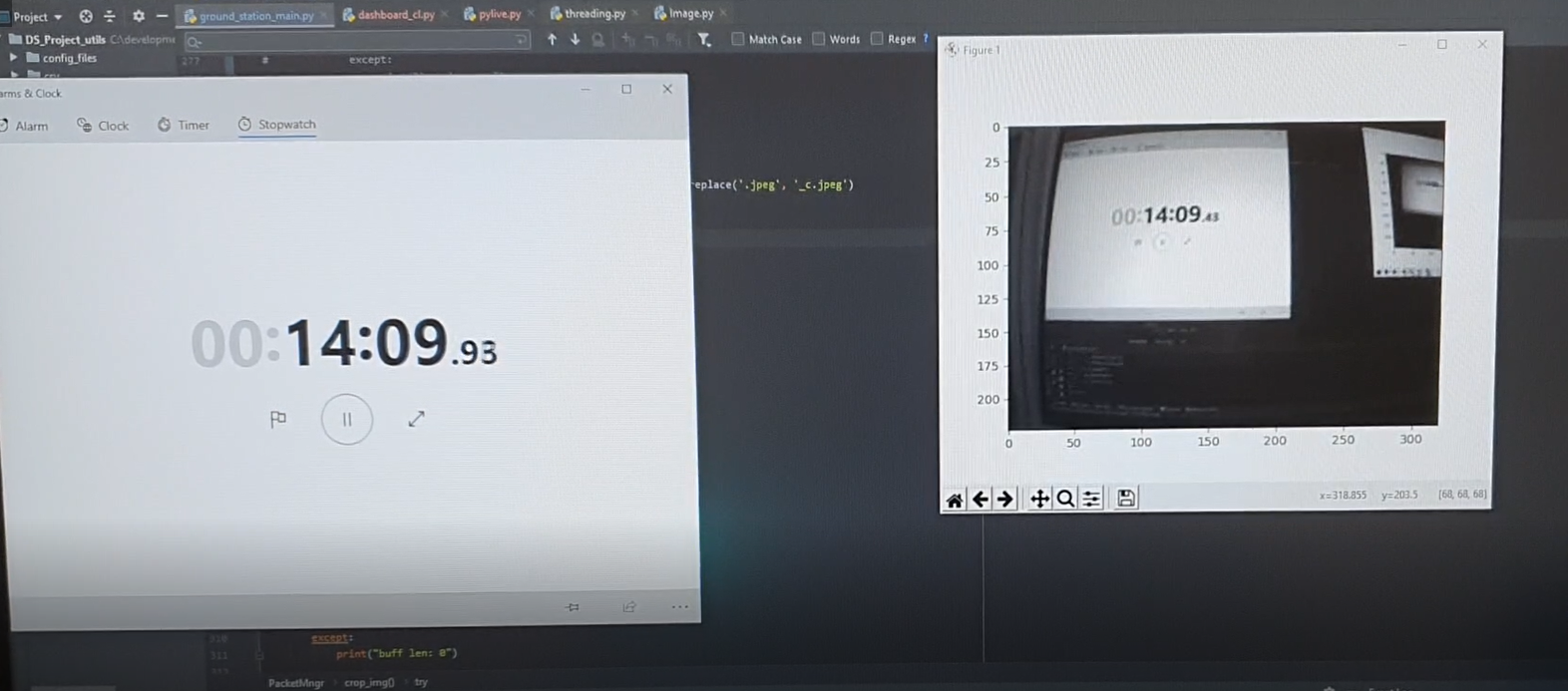


Figure 22: Delay inspection measuring setup

## תוצאות

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| delay |  |  |  |  |
| frames counted | mean delay [msec] | stdev [msec] | max [msec] | min [msec] |
| 16 | 237.50 | 27.73 | 270 | 200 |

ניתן לראות את ביצועי המערכת, עם שיהוי אופייני של כ 316msec בין כל פריים, עם קצב אופייני של 16fps

# סיכום

## סיכום

הפרויקט מציג מטע"ד קטן וחדשני, ככלי עזר למחקר ולעבוד המשך, שמביא נתונים ויזואליים-אינרציאליים לתחנת קרקע.

במעבדת הרחפנים האוטונומיים באוניברסיטת תל אביב, נעשה שימוש רב בניווט רחפנים באמצעות SLAM ע"י קבלת מידע חזותי של מיקום הרחפן. ההבנה שהיתוך מידע אינרציאלי יכול לשפר את ביצועי הSLAM, הביאו אותנו לחפש מוצרים שיכולים לתת מענה לבעיה הטכנולוגית.

הפרויקט יצא לדרך מתוך הבנה שלא קיימת מערכת מדף כזו מוכנה לצרכי המעבדה, ועלינו לתכנן ולבנות כזו בעצמנו.

המעבדה סיפקה לנו את הצרכים ההנדסיים שנזדקקנו להם, החל מחומרות, כלי עבודה, סביבת עבודה והנחיה טכנולוגית.

שלבי הפיתוח, מתוך עבודה בסביבת אילוצים נתונה, הביאו אותנו לבצע שיקולים הנדסיים כדוגמת שימוש נבון ברוחב הפס הנתון, עבודה בגודל תמונות וקצב נתונים מסוים.

הפרויקט המשותף של תחנת הקרקע והמטע"ד, שמורים בrepository של מעבדת הרחפנים:

<https://github.com/tau-adl/Drones-Navigation-Payload>

ברצוני להודות למר יוני מנדל - המנחה שלי בפרויקט הזה, ולשנהב עובדיה- השותף לבניית המערכת הזו.

## עבודות המשך

הפרויקט הוכיח היתכנות מבחינת חומרה ותכנה להשלמת מטרת הפרויקט. אולם יש עוד מקום לשיפור - על מנת לתת ביצועים מספקים ואמינים לטובת השמשת אלגוריתמי ניווט ועיבוד post processing ערכי על תחנה קרקעית

1. השמשת מצבי שינה בבקר, או מציאת תדר עבודה אופטימלי על מנת להשיג צריכות זרמים אופטימליים למטע"ד שמוזן ממתח סוללה
2. עריכת הרכיבים ע"ג מעגל ייעודי – דבר שיפתור בעיות אלמ"ג, רובוסטיות של המערכת, יקטין גודל ויפחית משקל
3. בחינת שימוש במצלמה FLIR על מנת לתמוך באפליקציות ניווט לא בחדר סגור
4. התאמת קצב דגימת המצלמה להתמודדות עם תופעות ריצוד תאורת פלורוסנט בתמונות
5. בחינת התועלת בתוספת תאורה חיצונית לתמונה – דוגמאת מבזק או תאורה להצפת הסצינה
6. עבודה עם מודם המטע"ד בתור Client ולא בתור AP, יאפשר עבודה עם תחנה קרקעית אחת אל מול יותר ממטע"ד יחיד

# נספחים

## תמונת נוספות של המערכת

|  |  |
| --- | --- |
|  |  |
|  |  |

## ביבליוגרפיה

1. VersaVIS: An Open Versatile Multi-Camera Visual-Inertial Sensor Suite, Florian Tschopp , Michael Riner , Marius Fehr , Lukas Bernreiter , Fadri Furrer , Tonci Novkovic , Andreas Pfrunder , Cesar Cadena , Roland Siegwart , and Juan Nieto, Sensors 2020
2. **OpenMv** - <https://openmv.io/>
3. **Global Shutter vs. Rolling Shutter image (Figure 1)** - <https://andor.oxinst.com/learning/view/article/rolling-and-global-shutter>
4. **Digital Camera Interface** – and STmicro document - dm00373474
5. **MPU6050 breakout schematic** - <https://easyeda.com/replicagames/mpu6050>
6. **MPU6050 driver -** <https://github.com/sinadarvi/SD_HAL_MPU6050>
7. **OpenMV repository** - <https://github.com/openmv/openmv>
8. **Digital Camera Interface (DCMI) for STM32 MCUs** – AN5020
9. **On Semiconductor MT9V034** - Datasheet
10. **STM32H743xI/G** – Datasheet
11. **Invensense MPU6050** - Datasheet
12. **Microchip ATWINC1500** - Datasheet