**פרויקט AutoRaceCar**



עדי מזוז: 054-2591059

דימה קולטונוב: 052-2987612

אסף אנטר: 0507116501

**תוכנן עניינים**

|  |  |
| --- | --- |
| [**מטרה**](#מטרה)**............................................................................................................................** | 4 |
| [**יעדים**](#יעדים)**.............................................................................................................................** | 4 |
| [**ניהול גרסאות**](#ניהול_גרסאות)**.................................................................................................................** | 5 |
| [**תיאור המערכת**](#תיאור_המערכת)**..............................................................................................................** | 6 |
| [סוללת מערכת](#סוללת_מערכת).............................................................................................................. | 8 |
| [מחשב מרכזי (xavier)](#מחשב_מרכזי)................................................................................................... | 8 |
| מצלמה....................................................................................................................... |  |
| מפצל USB(HUB)......................................................................................................... |  |
| Arduino uno............................................................................................................... | 10 |
| Arduino nano............................................................................................................. | 11 |
| סוללת מנועים.............................................................................................................. |  |
| חיישן תנועה [(optical flow).](#חיישן_תנועה)........................................................................................... |  |
| הספק ואנרגיה.............................................................................................................. |  |
| כבלים והטענה.............................................................................................................. |  |
| **ספריות עזר והתקנות......................................................................................................** |  |
| JetPack....................................................................................................................... |  |
| Cmake........................................................................................................................ |  |
| Qt.............................................................................................................................. |  |
| OpenGL...................................................................................................................... |  |
| Turbo-Jpeg................................................................................................................. |  |
| LibRealSense.............................................................................................................. |  |
| **ארדואינו – חיישן תנועה...................................................................................................** | 20 |
| **ארדואינו – מנועים – גלגלים............................................................................................** | 21 |
| **ממשקים........................................................................................................................** |  |
| ממשק מצלמה.............................................................................................................. |  |
| ממשק מנוע.................................................................................................................. |  |
| ממשק תקשורת סריאלית............................................................................................... |  |
| ממשק חיישן תנועה....................................................................................................... |  |
| ממשק TcpClient.......................................................................................................... |  |
| ממשק TcpServer......................................................................................................... |  |
| **מחלקות נוספות..............................................................................................................** |  |
| RemoteControl – gui................................................................................................... |  |
| JpegCompressor ....................................................................................................... |  |
| JepgDecompressor...................................................................................................... |  |
| Chaos......................................................................................................................... |  |
| **פערים והמלצות..............................................................................................................** |  |

**מטרה**

בניית פלטפורמת פיתוח לרכב אוטונומי הכוללת קבלת מידע מחיישנים שונים, יכולת הוספת חיישנים, שימוש בכלי פיענוח ושילוב אלגוריתמים.

**יעדים**

* תכנון ובניית הפלטפורמה – מכנית וחשמלית.
* סקר ספרות ובחירת כלים לכלל המערכת:
* חיישנים בסיסיים: מצלמה, חיישן תנועה, בקר מהירות, מיקרו קונטרולרים ומחשב מרכזי.
* סביבת עבודה ומערכת הפעלה.
* שפת פיתוח וספריות שונות.
* ממשק מצלמה:
* הוצאת תמונות: תמונת צבע, תמונת אינפרה ואדום ותמונת עומק.
* הוצאת נתוני gyro: מהירות זוויתית לשלוש דרגות חופש.
* הוצאת נתוני תאוצה: תאוצה קווית לשלוש דרגות חופש.
* נתוני intrinsics ו- extrinsics על הרכיבים השונים במצלמה.
* ממשק מנוע:
* כתיבת דרייבר הצרוב על מיקרו קונטרולר(ארדואינו ננו) המאפשר קבלת פקודות מגורם חיצוני ושליחתם למנועים.
* כתיבת ממשק חיצוני המאפשר תקשורת מול הארדואינו ושליחת פקודות למנועים.
* ממשק חיישן תנועה:
* כתיבת דרייבר הצרוב על מיקרו קונטרולר(ארדואינו אונו) המאפשר קריאת מידע והחיישן והעברתו לגורם חיצוני.
* כתיבת ממשק חיצוני המאפשר תקשורת מול הארדואינו וקבלת המידע.
* ממשקי תקשורת(tcp):
* ממשק client וממשק server המאפשרים העברת נתונים בין מחשבים.
* ממשק גרפי:
* ממשק המציג בזמן אמת את כל נתוני החיישנים: תמונה, מהירות זוויתית, תאוצה קווית, תנועה ומהירות קווית וכמו כן, בעל יכולת תקשורת מול מחשב מרוחק.

**ניהול גרסאות**

כל הפרויקט מנוהל ב github בהרשאת **private** כיוון שכל הפרויקט שייך לטכניון.

ניתן לעשות clone לפרויקט דרך הקישור:

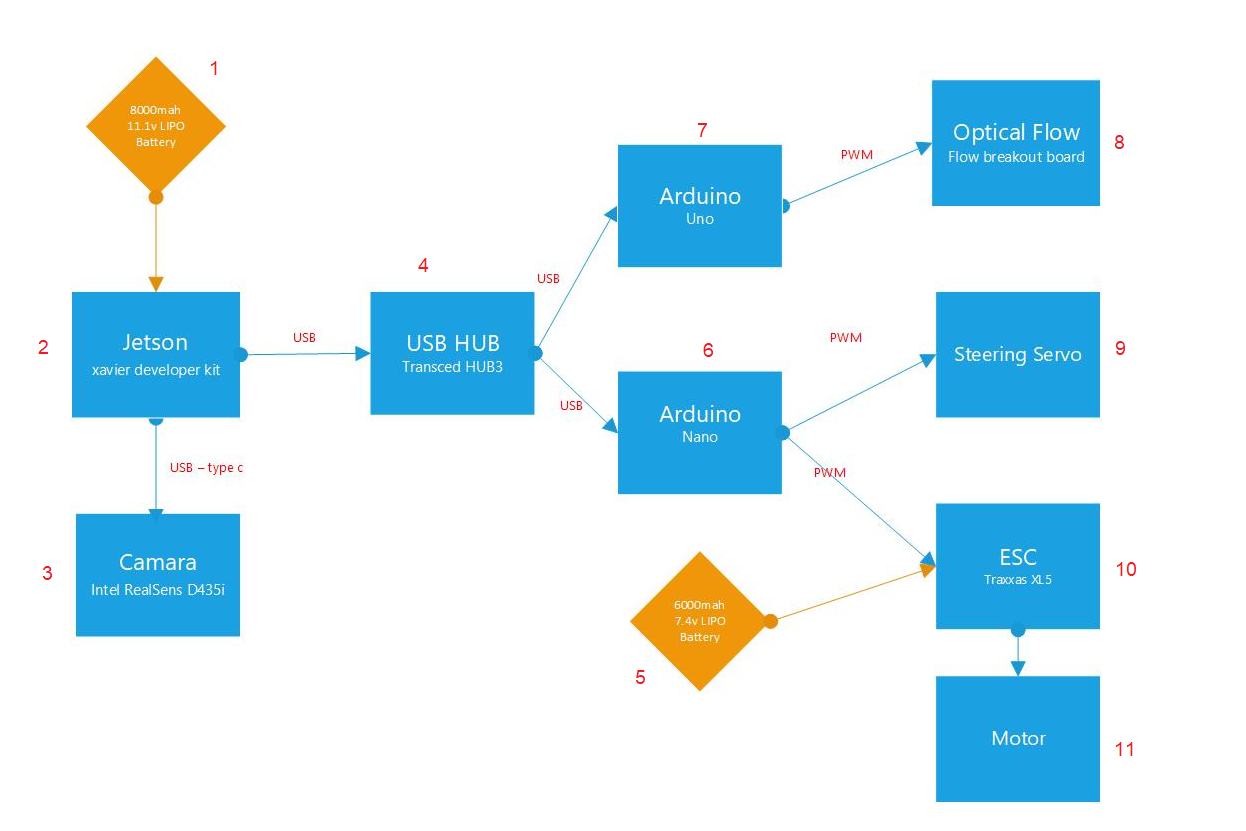
<https://github.com/asafanter/AutoRaceCar.git>

ממליצים להמשיך ולנהל את הגרסאות דרך הגיט גם לדורות הבאים כדי שיהיה מקום יחיד שנגיש לכל הפרויקט וגיבוי מלא.

סרטון הדרכה נחמד למתחילים:

<https://www.youtube.com/watch?v=SWYqp7iY_Tc>

**תיאור המערכת**



המערכת מורכבת ממחשב מרכזי, xavier(2) אשר מקבל מתח מספק(1).

אל ה- jetson מחוברים מאופן ישיר מצלמה(3) ומפצל usb(4).

המפצל usb מכיל 4 יציאות ואליו מחוברים שני מיקרו קונטרולרים, Arduino uno(7)

ו – Arduino nano(6), בנוסף עוד שתי יציאות אלינן ניתן לחבר מקלדת ועכבר.

אל ה - Arduino uno מחובר החיישן תנועה optical flow(8).

אל – Arduino nano מחוברים servo(9) אשר שולט על הגלגלים ובקר מהירות(10) אשר שולט על המנועים(11) המקבלים מתח מסוללה(5) כאשר רכיבים 9,10,11,5 הם רכיבים אשר גלולים בחבילת הרכב.

עדי.

תיאור הפלטה.

הוספת שרטוט.

הוספת איש קשר ומספר טלפון סמיון.

**סוללת מערכת**

תמונה של הסוללה.

דגם.

כמות: 1.

מתח: 11.1[V].

אנרגיה: 8000[mAh].

**מחשב מרכזי (xavier)**

תמונה של ה xavier.

הסבר על מצבי עבודה וההספקים השונים.

ספק 19V וזרם 1.5A.

הסבר על הכבל שהכנו שמשתמש רק ב 12V כדי שלא יהיו הפרשי מתחים.

הסבר על הסקריפטים שמריצים בעליית מערכת ומה המשמעות שלהם.

מילה על זה שהוספנו רכיב wi-fi כי אין מובנה.

**מצלמה**

תמונה של המצלמה.

הספק וזרם, אם אני זוכר זרם 0.7A ומתח הפעלה 5V.

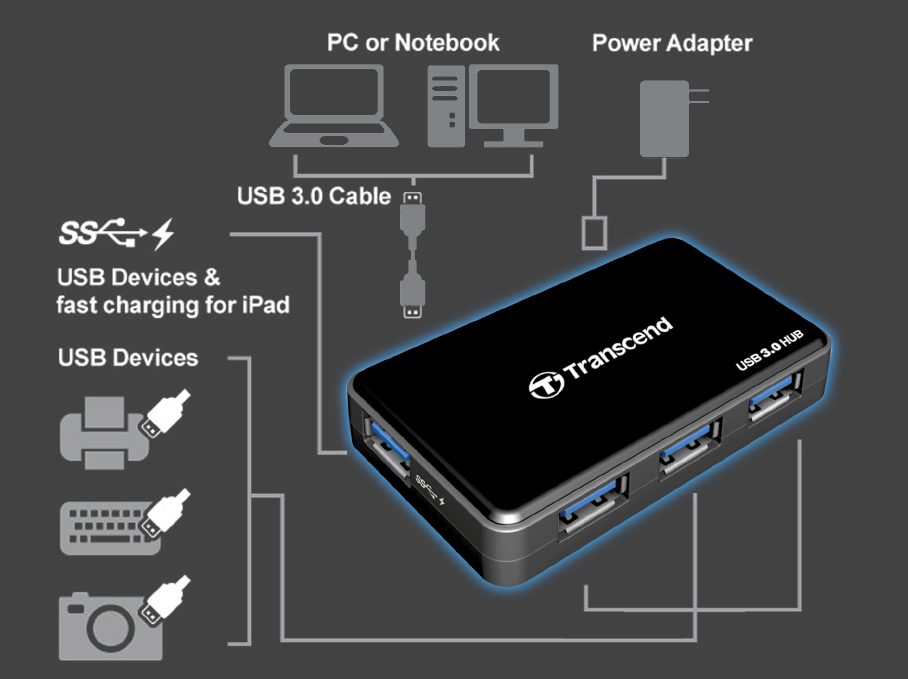
סוגי המידע שאפשר להוציא: תמונת צבע, אינפרה אדום, מפת עומק, מהירות זוויתית ונתוני תאוצה, קבועי מצלמה פנימיים (intrinsics) ומיקומי רכיבים (extrinsics).

תמונה נוספת של המצלמה עם המערכת צירים שלה.

עדכון עמוד בתוכן עניינים.

אולי לפרט על חוסר התאימות של המצלמה ולשקול החלפה??

**מפצל USB-HUB**



**תיאור:** המפצל מכיל כניסה אחת ו4 יציאות של usb 2.0/3.0.

**הספק:** המפצל יכול להיות מוזן ממתח של מחשב המחובר אליו או מספק כוח.

אם מוזן ממחשב אז צורך 5[V] ו- 0.9[A] (המצב הנוכחי).

ניתן גם לחברו ישירות לספק מתח ואז צורך 12[V] ו- 1.5[A].

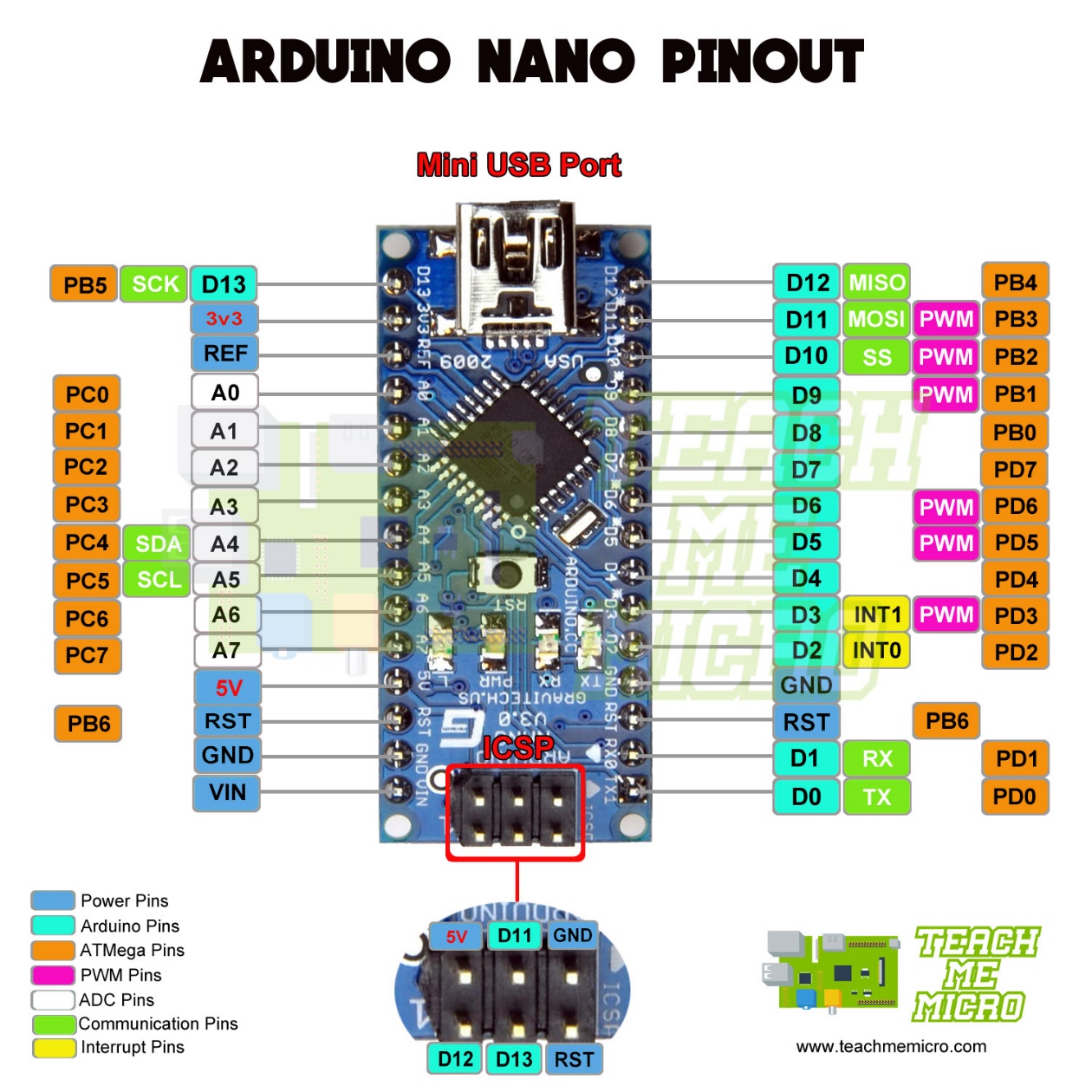
**Arduino uno**



צריכת מתח וזרם(5V ו- 0.05A).

השימוש העיקרי בבקר זה הוא עבור שליטה בoptical flow, על הארדואינו צרוב הקוד BitcrazeArd , ויש ממשק חיצוני שיודע לתקשר עם הקוד הצרוב על הארדויאנו – לבקש ולקבל מידע מה- optical flow

**Arduino nano**



מתח וזרם(5V ו- 0.04A).

השימוש העיקרי בבקר זה הוא עבור שליטה במנוע ובהיגוי, על הארדואינו צרוב הקוד RaceCarArd , ויש ממשק חיצוני שיודע לתקשר עם הקוד הצרוב על הארדויאנו , הקוד הצרוב מחכה לקבל פקודות נסיעה מהממשק החיצוני

**סוללת מנועים**

תמונה של הסוללה.

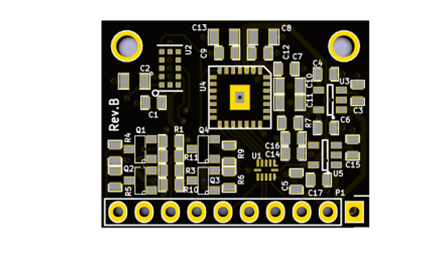
דגם.

כמות: 1.

מתח:

אנרגיה:

**חיישן תנועה**



**תיאור:** החיישן הוא חיישן תנועה המכיל בתוכו שני רכיבים: רכיב המודד

מרחק (VL5350x) למשטח תחתיו ורכיב המזהה

שינוי בתנועה ( optical pwm3901mb).

**דגם:** flow breakout bitcraze

**הספק(משולב):** 3.6[V] \* (0.09 + 0.019)[A] = 0.1[W]

**Fov:** 42[deg].

**Pixels\_x:** 30[px]

**Pixels\_y:** 30[px]

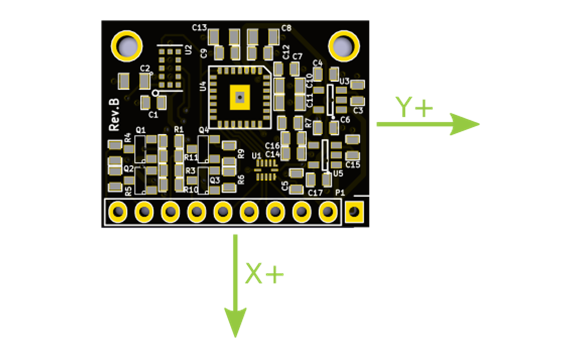
**פלט:** החיישן פולט שלושה סוגי מידע:

Dx – השינוי בפיקסלים בכיוון x. [px]

Dy – השינוי בפיקסלים בכיוון y. [px]

Range – המרחק מהרכיב למשטח תחתיו כאשר המינימום הוא 8[mm].

כאשר הצירים מוגדרים כך:



**חישוב מהירות:** ניתן לחשב מהירות מהפלט של החיישן על ידי הנוסחה הבאה:

**הספק ואנרגיה**

להלן טבלה המסכמת את ההפסקים של הרכיבים במערכת:

|  |  |  |  |
| --- | --- | --- | --- |
|  | מתח [V] | זרם [A] | הספק [W] |
| Jetson | 20 | 1.5 | 30 |
| מצלמה | 5 | 0.7 | 3.5 |
| מפצל USB | 5 | 0.9 | 4.5 |
| Arduino uno | 5 | 0.05 | 0.25 |
| Arduino nano | 5 | 0.04 | 0.2 |
| חיישן תנועה | 3.6 | 0.028 | 0.1 |

סך האנרגיה שהמערכת צורכת היא: 38.55[W].

הסוללה מספקת 11.1[V] וזרם של 8000[mAh] ולכן סך האנרגיה: 88.8[Wh].

כלומר הסוללה מספיקה לזמן עבודה של  .

**כבלים והטענה**

תמונות של הכבלים הרלוונטיים שהכנו.

אני לא זוכר את כולם אבל נראה לי:

* מהמטען לסוללת מערכת
* מהמטען לסוללת המנועים
* הכבל המיוחד שמאפשר הפעלה של סוללה וחשמל במקביל.

להסביר על כל אחד מהם בקצרה שפשוט ידעו להשתמש.

יהיה ממש מקצועי להסביר את המעגל של הכבל המיוחד, אם צריך עזרה אני יכול לעזור.

לעדכן עמוד בתוכן עניינים.

**ספריות עזר והתקנות**

על מנת שכל התוכן יעבוד כראוי יש להתקין את כל ספריות העזר והתוכנות הנדרשות, נחלק לשתי סביבות עיקריות: סביבת העבודה של ה – xavier המכילה את chaos, וסביבה המכילה את ה- RemoteControl.

ההתקנות עליהן נפרט הן:

**JetPack:** חבילה של nvidia המותאמת למוצרי ה- jetson השונים, בפרויקט הזה – xavier. מכילה מספר ספריות, בין היתר openCv, Cuda ועוד.

**Cmake:** ספרייה המאפשרת להגדיר קונפיגורציות כאשר מקמפלים קוד c/c++ בתצורה של cross-platform.

**Qt:** IDE לכתיבת c/c++ המאפשר פיתוח gui. מכיל המון class-ים וכמובן cross-platform.

**OpenGl:** ספרייה לתצוגה גרפית,ברוב המערכות הספרייה כבר מותקנת אבל ליתר ביטחון נרחיב גם על ספרייה זו.

**Turbo-Jpeg:** ספרייה המשמשת לדחיסת תמונות לפי האלגוריתם של jpeg, משתמש לדחיסת התמונות מהמצלמה טרם שליחתם ברשת.

**LibRealSense:** ספריית הממשק למצלמת ה – realsense.

**JetPack**

כאשר מתקינים את ה – xavier מההתחלה יש להתקין בנוסף את חבילה זו.

הסבר מפורט ו - וידאו המסביר את אופן ההתקנה אפשר למצוא בלינק:

<https://www.google.com/search?q=how+to+install+jetpack+xavier&oq=how+to+install+jetpack+xavier&aqs=chrome..69i57j0.10401j0j4&sourceid=chrome&ie=UTF-8#kpvalbx=_xudzXZSkL8GXkwXC3LqgCg18>

דגשים:

* את קובץ ההתקנה מורידים למחשב חיצוני אחר(host) **לא על ה – xavier.**

ההתקנה מתבצעת על מחשב ה – host עם כבל מחובר ל – xavier.

* יש לשים לב שבזמן ההתקנה עוברים למצב setup mode על ידי לחיצת שני כפתורים בו זמנית(מוסבר בווידאו).
* בשל חוסר זמן, לא חקרנו מספיק את יכולות החבילה, אנו ממליצים לשים דגש על לימוד החבילה לדורות הבאים שכן היא מותאמת לחומרה של ה – xavier.

**Cmake**

Cmake מאפשר לקמפל קוד c/c++ תחת הגדרות שונות (למשל הכרת נתיבי includes, קישור לקבצי dll/so/a/lib ועוד).

בכל פרויקט open source (ניתן למצוא בשפע ב github) יש קובץ הנקרא CmakeList.txt שמריצים אותו כאשר לאחר ההרצה נוצר קובץ makeFile ואז מקמפלים.

ניתן להתקין את Cmake גם בווינדוס וגם בלינוקס.

בלינוקס זה ייעשה על ידי השורות הבאות:

>> sudo apt-get install cmake

פרויקט יקומפל בעזרת השורות הבאות:

>> cmake .

>> make

הפקודה הראשונה(Cmake) מקבל כארגומנט path לתיקייה המכילה את הקובץ CmakeList.txt ומחפשת אותו לבד. במקום לשים path מלא ניתן לרשום נקודה, הנקודה מציינת את התיקייה הנוכחית, כלומר ניתן לרשום את הפקודה עם כל ה path או לפתוח את הטרמינל בתיקייה הרצויה ולהשתמש בנקודה.

הפקודה השנייה(make) חייבת להיות מופעלת בתוך תיקייה שבה נמצא קובץ בשם makeFile או רק make והיא מריצה אותו לבד, זה למעשה הקימפול.

**Qt**

Qt(שנהגה כמו cute) הוא IDE cross-platform(כלומר את אותו פרויקט ניתן לקמפל בכל מערכות ההפעלה) ובנוסף ניתן לפתח איתו gui באמצעות c++.

ניתן לפתח בעזרתו תוכנות console רגילות והוא מכיל מאגר אדיר של class-ים מתקדמים(מערכות קבצים, sockets, serialization ועוד).

ניתן להוריד את התוכנה החינמית opensource מהאתר הרשמי של qt לווינדוס או ללינוקס אך ב – xavier צריך להוריד אותו מהטרמינל ולא מהאתר שכן באתר אין הורדה לארכיטקטורת arm.

כדי להוריד יש להריץ:

>> sudo apt-get install qt5-default qtcreator

**OpenGl**

OpenGl היא ספרייה גרפית opensource אשר יודעת להשתמש במנועים הגרפיים באמצעות ה – gpu.

נדרש להתקין אותה גם כי הספרייה לדחיסת התמונות דורשת אותה וכמו כן, הדוגמאות לשימושי המצלמה ב – github של realsense משתמש בה.

כדי להתקין את הספרייה יש להריץ:

>> sudo apt-get install libglu1-mesa-dev freeglut3-dev mesa-common-dev

**Turbo-Jpeg**

יש להתקין את הספרייה זו אם נעשה שימוש בדחיסת תמונות באלגוריתם Jpeg.

לצורך פרויקט זה נכתבו שתי מחלקות בשימוש Turbo-Jpeg אשר יפורטו בהמשך.

אחת משמשת לדחיסת התמונה שמתקבלת מהמצלמה ושלחיתה ב – socket והשנייה משמשת לפריסת התמונה במחשב המקבל.

על מנת להתקין את הספרייה בלינוקס יש לבצע את ההוראות הבאות:

1. להוריד את תוכנת האסמבלי מלינק הבא:

<http://www.nasm.us/pub/nasm/releasebuilds/2.14.02/nasm-2.14.02.tar.xz>

1. לחלץ את הקבצים לתיקייה הרצויה.
2. לפתוח את הטרמינל בתיקייה הרצויה.
3. להריץ:

>> ./configure --prefix=/usr && make

1. להריץ:

>> sudo make install

1. להוריד את הספרייה של jpeg מהלינק הבא(יש להמתין 5 שניות לפני הורדה):

<https://downloads.sourceforge.net/libjpeg-turbo/libjpeg-turbo-2.0.2.tar.gz>

1. לחלץ את הקבצים לתיקייה הרצויה.
2. להריץ:

>> mkdir build &&

cd build &&

cmake -DCMAKE\_INSTALL\_PREFIX=/usr \

-DCMAKE\_BUILD\_TYPE=RELEASE \

-DENABLE\_STATIC=FALSE \

-DCMAKE\_INSTALL\_DOCDIR=/usr/share/doc/libjpeg-turbo-2.0.2 \

-DCMAKE\_INSTALL\_DEFAULT\_LIBDIR=lib \

.. && make

1. להריץ:

>> sudo make install

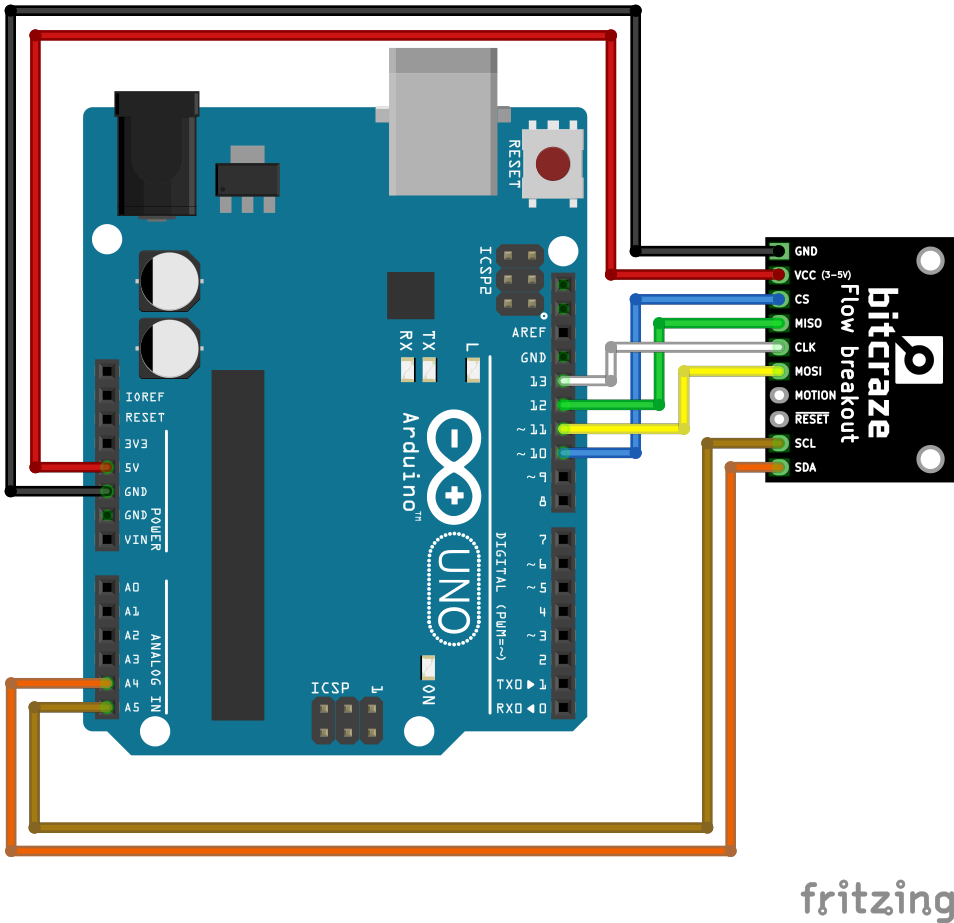
**LibRealSense**

יש דוגמאות מעל איך הסברתי על הספריות.

להסביר על אופן ההתקנה מפורט ככל שניתן.

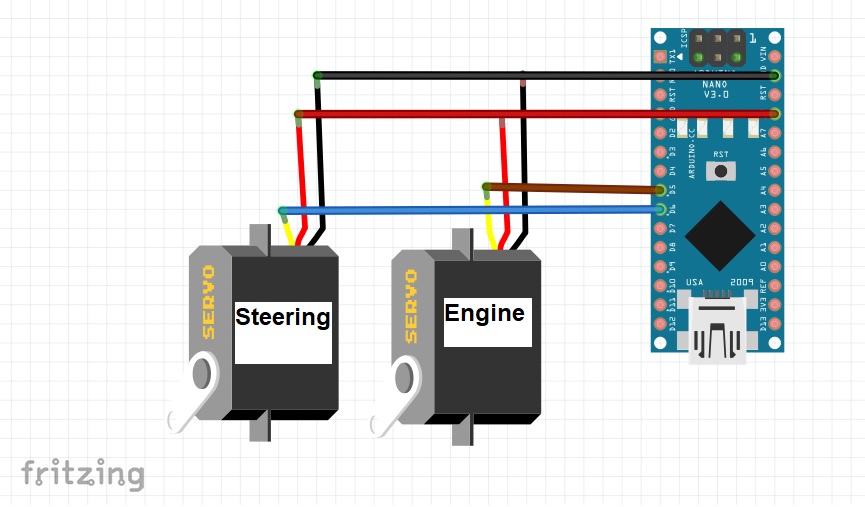
לצרף לינקים אם אפשר(לינק ל – github של הדוגמאות יהי מצויין).

**ארדואינו – חיישן תנועה**



החיבור בין הארדויאנו לבין הoptic flow תואם לציור זה (גם בצבעים), אל הbitcraze הולחמו פינים ( הרכיב מגיע ללא חיבור GPIO נוח) , החוטים הולחמו ליחדות שלמות עבור חיבור וניתוק נוחים של הרכיב.

**ארדואינו – מנועים – גלגלים**



החיבור בין הארדויאנו לבין הoptic flow תואם לציור זה (גם בצבעים).

בעת הדלקה ראשונית של הרכב, חייב ללחוץ לחיצה ארוכה על הכפתור שנמצא על הspeed controller עד שיהבהב בירוק.

על מנת שהמנוע יקבל פקודות נסיעה צריך לשלוח אל הבקר מהירות פקודת ARM, (נעשה על ידי הקוד הרשום בארדואינו)

**ממשקים**

יעד עיקרי של הפרויקט הוא כתיבת הממשקים מול הרכיבים והחיישנים השונים.

רוב הממשקים נכתבו בתצורה של pure virtual class כדי לבודד את הקבצים כמה שניתן וליצור הפרדה בין ממשק למימוש.

בתצורה זו, מאוד נח ליצור קבצי lib/dll/so ועשות include רק לקובץ הממשק.

הסבר נוסף לכתיבת ממשקים ב- c++ אפשר למצוא בלינק הבא:

<https://en.wikibooks.org/wiki/More_C%2B%2B_Idioms/Interface_Class>

**ממשק מצלמה:**

להכניס ממשק פה.

לעדכן עמוד בתוכן עניינים.

**ממשק מנוע:**

**Arduino**

**תיאור המחלקה:**

Arduino היא מחלקה היורשת מהמחלקה האבסטרקטית MotorAPI – מחלקה המגדירה את פונקציות הממשק הדרושות (למקרה של שינוי עתידי מארדואינו)

המחלקה מכילה בתוכה פונקציות ממשק עבור כל הרכיבים המחוברים לארדואינו , המנוע וההיגוי

פרמטרים עבור המחלקה:

speed : מהירות נינתנת במספר שלם חסר יחידות הנע בין [-30,30]

מספר זה מייצג את הערך הנשלח אל הPWM של הארדואינו , כאשר pwm=90 הוא האפס מבחינת מהירות (ומשמש גם להדלקת המנוע) והמספר הממשי שנשלח הוא 90+speed

Angle : הזווית נשלחת במעלות , הערכים האפשריים [30,150], זווית של 90 מייצגת גלגלים ישרים

**פונקציות הממשק:**

bool Arduino::connect()

**תיאור הפונקציה:**

הפונקציה מתחברת לתקשורת סיראלית אל מול הארדואינו לממשק.

**פרמטרים:**

אין.

**ערך החזרה:**

מצביע לממשק.

int Arduino::getAngle()

**תיאור הפונקציה:**

מחזירה את הזווית הנוכחית ( שמורה במשתנה פנימי במחלקה).

**פרמטרים:**

אין.

**ערך החזרה:**

הזווית הנוכחית.

int Arduino::getSpeed()

**תיאור הפונקציה:**

מחזירה את המהירות הנוכחית ביחידות PWM ( שמורה במשתנה פנימי במחלקה).

**פרמטרים:**

אין.

**ערך החזרה:**

המהירות הנוכחית הנוכחית.

Arduino &Arduino::drive(const int &wanted\_speed, const int &wanted\_angle)

**תיאור הפונקציה:**

הפונקציה שולחת פקודת נסיעה למנוע והגלגלים וגורמת לו לנסוע במהירות והזווית הנתונות עד לפקודה חדשה

**פרמטרים:**

wanted\_speed – המהירות הרצויה

wanted\_angle – הזווית הרצויה

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

Arduino &Arduino::changeSpeed(const int &wanted\_speed)

**תיאור הפונקציה:**

הפונקציה שולחת פקודת נסיעה למנוע וגורמת לו לנסוע במהירות הנתונה עד לפקודה חדשה

**פרמטרים:**

wanted\_speed – המהירות הרצויה

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

Arduino& Arduino::changeSpeedBy(const int &delta)

**תיאור הפונקציה:**

הפונקציה שולחת פקודה למנוע לשנות את המהירות במספר הנתנות

**פרמטרים:**

delta – בכמה יחידות לשנות את המנוע מהמצב הנוכחי

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

Arduino &Arduino::changeAngle(const int &wanted\_angle)

**תיאור הפונקציה:**

הפונקציה שולחת פקודה להיגוי וגורמת לו לשנות את הגלגלים בזווית הנתונה עד לפקודה חדשה

**פרמטרים:**

wanted\_angle – הזווית הרצויה

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

Arduino& Arduino::changeAngleBy(const int &delta)

**תיאור הפונקציה:**

הפונקציה שולחת פקודה להיגוי וגורמת לו להוסיף או להחסיר מעלות מזווית הגלגלים

**פרמטרים:**

delta – בכמה מעלות לשנות את הזווית מהמצב הנתון

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

Arduino &Arduino::stop()

**תיאור הפונקציה:**

הפונקציה שולחת פקודת עצירה, עוצרת את המנוע ומיישרת את הגלגלים

**פרמטרים:**

אין

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

Arduino &Arduino::driveCurrentState(){

**תיאור הפונקציה:**

הפונקציה שולחת פקודה להמשיך לנסוע במצב הנוכחי

**פרמטרים:**

אין

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

**ממשק תקשורת סריאלית**

**תיאור:**

ISerial הוא ממשק הנועד לממש תקשורת סריאלית בין מכשירים.

השימוש העיקרי בפרויקט שלנו הוא ליצור תקשורת בין המחשב (במקרה הנ"ל xavier) ל Arduino nano ולהעביר פקודות לבקר מנוע דרך תקשורת זו.

המחלקה נתמכת על ידי לינוקס בלבד וחלק מהשדות שלה הוא נתיב לקובץ סריאלי (ברגע שמחברים arduino למחשב נפתח קובץ file descriptor או בקיצור fd).

דרך קובץ זה ניתן לבצע פקודות קריאה וכתיבה כפי שיוסבר בהמשך.

בנוסף קיימת מחלקה שנקראת Serial אשר יורשת ממחלקה זו שבה נמצא כל המימוש, הפונקציות המפורטות בהמשך מוגדרות בממשק ISerial וממומשות במחלקה Serial.

פונקציות הממשק:

static std::shared\_ptr<ISerial> create()

**תיאור הפונקציה:**

הפונקציה יוצרת מופע של הממשק ומחזירה מצביע לממשק.

**פרמטרים:**

אין.

**ערך החזרה:**

מצביע לממשק.

virtual ISerial &connect(const string &path = "/dev/ttyACM0")

**תיאור הפונקציה:**

הפונקציה פותחת ערוץ תקשורת סריאלי לקובץ שמועבר כפרמטר.

**פרמטרים:**

path – הקובץ שנפתח עבורו ערוץ סריאלי, הנתיב הדיפולטיבי הוא נתיב אוטומטי שנפתח כאשר מחברים את ה- arduino.

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

virtual void flush()

**תיאור הפונקציה:**

הפונקציה מרוקנת את הקובץ הסריאלי.

**פרמטרים:**

אין

**ערך החזרה:**

אין

virtual ISerial &write(const string &msg)

**תיאור הפונקציה:**

הפונקציה כותבת לערוץ התקשורת מחרוזת מטיפוס std::string.

**פרמטרים:**

msg – המחרוזת שנשלחת לערוץ התקשורת.

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

virtual ISerial &write(const char &msg)

**תיאור הפונקציה:**

הפונקציה כותבת לערוץ התקשורת תו יחיד.

**פרמטרים:**

msg – התו שנשלח לערוץ התקשורת.

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

virtual string read(const uint &len)

**תיאור הפונקציה:**

הפונקציה קוראת מחרוזת מטיפוס std::string מערוץ התקשורת.

פונקציה חוסמת.

**פרמטרים:**

len – אורך המחרוזת שיש לקרוא.

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

virtual void read(char \*dst, const uint &len)

**תיאור הפונקציה:**

הפונקציה קוראת מידע מערוץ התקשורת.

פונקציה חוסמת.

**פרמטרים:**

dst – היעד שאליו יש לקרוא את המידע.

len – אורך המחרוזת שיש לקרוא.

**ערך החזרה:**

מחזיר את הממשק חזרה לצורך שרשור פונקציות.

virtual bool isConnected() const

**תיאור הפונקציה:**

הפונקציה בודקת האם יש חיבור סריאלי או לא.

**פרמטרים:**

אין

**ערך החזרה:**

מחזירה true אם יש חיבור, אחרת מחזירה false.

virtual ~ISerial()

**תיאור הפונקציה:**

הפונקציה משחררת את כל המשאבים של התקשורת הסריאלית.

**פרמטרים:**

אין.

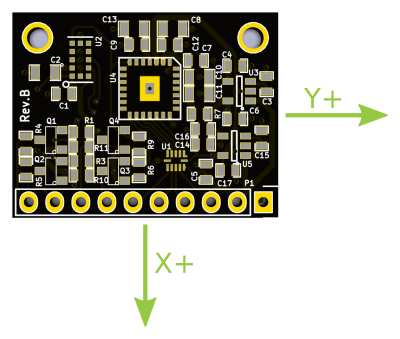
**ערך החזרה:**

אין.

**ממשק חיישן תנועה:**

**Bitcraze**

מערכת הצירים של הרכיב :

****

pixel resolution is 30x30

FOV is 42°

More data on : <https://wiki.bitcraze.io/breakout:flow>

אנו משתמשים ברכיב על מנת לקבל מבנה בשם Flow שמכיל 4 שדות

Dx - תזוזה ביחידות פיקסלים בציר x בין דגימות

Dy - תזוזה ביחידות פיקסלים בציר y בין דגימות

Dt - הזמן במילי שניות בין דגימות

Range – המרחק מהרצפה במילימטרים

עבור המרה בין יחידות פיקסלים למרחק נשתמש במשוואה הבאה:

**תיאור המחלקה:**

המחלקה עוטפת את התקשורת מול רכיב ה – optic sensor ומעבירה ממנו מידע אל המשתמש

**פונקציות הממשק:**

void bitcraze::setup()

**תיאור הפונקציה:**

הפונקציה מתחברת אל רכיב הoptic sensor (מצורף ציור של החיבור מול הארדואינו)

**פרמטרים:**

ללא

**ערך החזרה:**

ללא

bool isConnected()

**תיאור הפונקציה:**

הפונקציה בודקת אם אנו מחוברים אל הארדואינו

**פרמטרים:**

ללא

**ערך החזרה:**

True במידה ואנו מחוברים.

Bitcraze &requestFlowData();

**תיאור הפונקציה:**

הפונקציה שולחת הוראה לארדואינו להתחיל לשלוח מידע מהחיישן

**פרמטרים:**

ללא

**ערך החזרה:**

מצביע לממשק.

Bitcraze &stopStream();

**תיאור הפונקציה:**

הפונקציה שולחת הוראה לארדואינו להפסיק לשלוח מידע מהחיישן

**פרמטרים:**

ללא

**ערך החזרה:**

מצביע לממשק.

Flow getFlowOutput() ;

**תיאור הפונקציה:**

הפונקציה קוראת את המידע שהחיישן שולח דרך הארדויאנו

**פרמטרים:**

ללא

**ערך החזרה:**

Flow – מבנה המכיל את המידע מהחיישן

**ממשק TcpClient**

**תיאור המחלקה:**

ITcpClient הוא ממשק הנועד ליצור client והתחבר לשרת בתקשורת TCP.

בנוסף קיימת מחלקה שנקראת TcpClient אשר יורשת ממחלקה זו שבה נמצא כל המימוש, הפונקציות המפורטות בהמשך מוגדרות בממשק ITcpClient וממומשות במחלקה TcpClient.

פונקציות הממשק:

static std::shared\_ptr<ITcpClient> create()

**תיאור הפונקציה:**

הפונקציה יוצרת client חדש אשר יכול להתחבר לשרת ומחזירה מצביע לממשק.

**פרמטרים:**

אין.

**ערך החזרה:**

אין.

virtual void connect(const string& ip, const unsigned short& port) const

**תיאור הפונקציה:**

הפונקציה מתחברת לשרת קיים באמצעות ה – ip וה – port של השרת.

**פרמטרים:**

Ip – כתובת ה- ip של השרת.

Port – מספר הפורט של השרת.

**ערך החזרה:**

אין.

virtual bool isConnected() const

**תיאור הפונקציה:**

הפונקציה מחזירה אם קיימת תקשורת עם השרת או לא.

**פרמטרים:**

אין.

**ערך החזרה:**

מחזירה true אם יש תקשורת, אחרת מחזירה false.

virtual void disconnect()

**תיאור הפונקציה:**

הפונקציה מנתקת את התקשורת מהשרת.

**פרמטרים:**

אין.

**ערך החזרה:**

אין.

virtual void receive(char \*dst, const uint &len, const uint &timeout\_sec = 3)

**תיאור הפונקציה:**

הפונקציה קוראת מידע מהשרת.

**פרמטרים:**

Dst – היעד אליו יקרא המידע.

Len – אורך המידע בבתים.

Timeout\_sec – זמן ההמתנה המקסימלי עד שהתכנית תמשיך, אם הזמן שווה ל – 0 התכנית תמשיך מיד אם אין מידע לקרוא.

**ערך החזרה:**

וקטור של בתים המכיל את המידע שנקרא.

virtual void send(const std::vector<char>& data) const noexcept

**תיאור הפונקציה:**

הפונקציה שולחת מידע לשרת.

**פרמטרים:**

Data – המידע שיש לשלוח.

**ערך החזרה:**

אין.

virtual void send(const string& message) const noexcept

**תיאור הפונקציה:**

הפונקציה שולחת מחרוזת לשרת.

**פרמטרים:**

message – המחרוזת שיש לשלוח.

**ערך החזרה:**

אין.

virtual void send(const char\* data, const uint &len) const noexcept

**תיאור הפונקציה:**

הפונקציה שולחת מידע לשרת.

**פרמטרים:**

data – מצביע למידע שיש לשלוח.

Len – אורך המידע בבתים.

**ערך החזרה:**

אין.

virtual ~ITcpClient() noexcept

**תיאור הפונקציה:**

שחרור כל המשאבים.

**פרמטרים:**

אין.

**ערך החזרה:**

אין.

**ממשק TcpServer**

**תיאור המחלקה:**

ITcpServer הוא ממשק הנועד ליצור שרת ולפתוח תקשורת TCP בין מחשבים.

בנוסף קיימת מחלקה שנקראת TcpServer אשר יורשת ממחלקה זו שבה נמצא כל המימוש, הפונקציות המפורטות בהמשך מוגדרות בממשק ITcpServer וממומשות במחלקה TcpServer.

פונקציות הממשק:

static std::shared\_ptr<ITcpServer> create()

**תיאור הפונקציה:**

הפונקציה יוצרת Server חדש ומחזירה מצביע לממשק.

**פרמטרים:**

אין.

**ערך החזרה:**

מצביע לממשק.

virtual void bind(const string &ip, const unsigned short &port,

const int &max\_num\_of\_clients) noexcept

**תיאור הפונקציה:**

הפונקציה פותחת תקשורת TCP ומתחברת ל – Ip כשרת.

**פרמטרים:**

Ip – מחרוזת המכילה את כתובת ה – ip של מחשב ה – host שבו יפתח השרת.

Port – מספר פורט שרירותי שמוגדר בין המשתמשים שבניהם נפתחת התקשורת.

Max\_num\_of\_client – מספר המשתמשים המקסימלי שיכולים להתחבר לשרת.

**ערך החזרה:**

אין.

virtual bool isBind() const noexcept

**תיאור הפונקציה:**

הפונקציה בודקת אם קיימת תקשורת בין השרת למחשב ה – host.

**פרמטרים:**

אין.

**ערך החזרה:**

מחזירה true אם נפתחה תקשורת, אחרת מחזירה false.

virtual bool hasConnectionWithSocket(const Socket &socket)

**תיאור הפונקציה:**

הפונקציה בודקת אם קיימת תקשורת בין השרת ל – client מסוים.

**פרמטרים:**

Socket – המזהה של ה - client שאיתו יש לבדוק את התקשורת.

**ערך החזרה:**

מחזירה true אם קיימת תקשורת, אחרת מחזירה false.

virtual Socket waitForConnections(const uint &timeout\_sec)

**תיאור הפונקציה:**

הפונקציה מחכה להתחברות client חדש ומחזירה Socket(שם אחר ל int) אשר בעזרתו אפשר לגשת ל - client שהתחבר.

**פרמטרים:**

Timeout\_sec – הזמן המקסימלי שהפונקציה תמתין לבקשת התחברות חדשה.

**ערך החזרה:**

Socket ל - client שהתחבר.

virtual unsigned long getNumOfConnectedClients() const

**תיאור הפונקציה:**

מחזירה את מספר ה – clients המחוברים לשרת.

**פרמטרים:**

אין.

**ערך החזרה:**

מספר ה – clients המחוברים לשרת.

virtual void receive(const Socket &socket, char \*dst, const uint &len, const uint &timeout\_sec = 3)

**תיאור הפונקציה:**

הפונקציה קוראת מידע מ - client מסוים שמחובר לשרת.

**פרמטרים:**

Socket – המזהה של ה - client שממנו יש לקרוא את המידע.

Len – אורך המידע בבתים.

Timeout\_sec – זמן מקסימלי להמתנה למידע, אם הזמן שווה ל – 0, התכנית תמשיך מיד.

**ערך החזרה:**

אין.

virtual void send(const Socket& socket, const std::vector<char>& data) const noexcept

**תיאור הפונקציה:**

הפונקציה שולחת מידע ל - client מסוים שמחובר לשרת.

**פרמטרים:**

Socket – המזהה של ה - client שאליו יש לשלוח את המידע.

Data – המידע שיש לשלוח.

**ערך החזרה:**

אין.

virtual void send(const Socket& socket, const string& message) const noexcept

**תיאור הפונקציה:**

הפונקציה שולחת מחרוזת ל - client מסוים שמחובר לשרת.

**פרמטרים:**

Socket – המזהה של ה - client שאליו יש לשלוח את המידע.

message – המחרוזת שיש לשלוח.

**ערך החזרה:**

אין.

virtual void send(const Socket& socket, const char \*data, const uint &len) const noexcept

**תיאור הפונקציה:**

הפונקציה שולחת מידע ל - client מסוים שמחובר לשרת.

**פרמטרים:**

Socket – המזהה של ה - client שאליו יש לשלוח את המידע.

data – מצביע למידע שיש לשלוח.

Len – אורך המידע בבתים.

**ערך החזרה:**

אין.

virtual ~ITcpServer() noexcept

**תיאור הפונקציה:**

שחרור כל המשאבים.

**פרמטרים:**

אין.

**ערך החזרה:**

אין.

**מחלקות נוספות**

מלבד הממשקים מול החיישנים נכתבו מחלקות נוספות אשר משלבות בין הממשקים השונים:

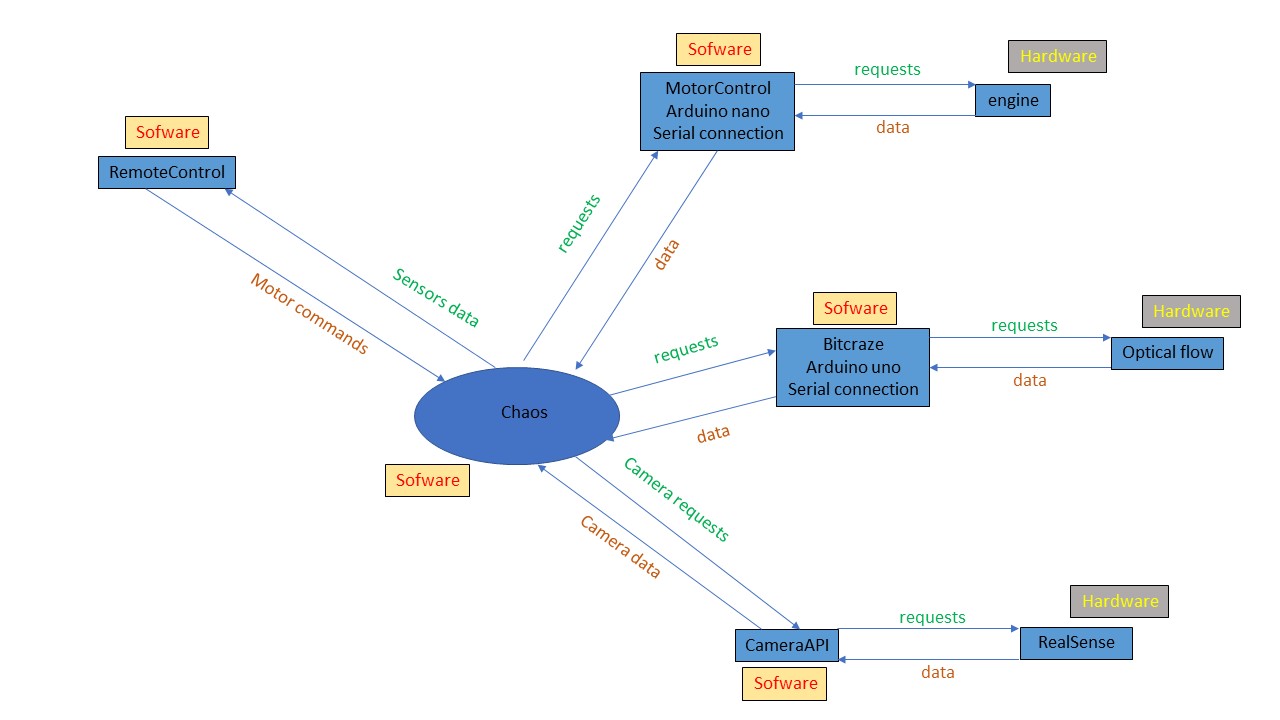
**RemoteControl:** gui אשר מופעל במחשב מרוחק ולו 2 מטרות:

* הצגת המידע מהחיישנים.
* שליטה ברכב באמצעות החיצים במקלדת.

**JpegCompressor:** מחלקה המשתמש לדחיסת תמונות.

**JpegDecompressor:** מחלקה המשמשת לפריסת תמונות דחוסות.

**Chaos:** הפרויקט העוטף.

תיאור כללי של מערכת התוכנה:

**RemoteControl – Gui**

**תיאור:**

ה – RemoteControl הוא כלי שפותח באמצעות Qt והוא ממשק גרפי.

הפרויקט משמש כ – TcpServer עבור ה – Chaos לצורך קבלת תמונות.

הפרויקט משתמש כ – TcpClient עבור ה – Chaos לצורך שליחת פקודות למנועים.

הפרויקט מציג נתונים מהמצלמה: תמונת צבע, אינפרה אדום או מפת עומק ויודע לבקש מה – Chaos את סוג המידע בזמן ריצה.

הפרויקט מציג נתוני תנועה בזמן אמת כגון: נתוני gyro, נתוני accelometer ונתוני optical flow.

\*כל המידע המוצג הוא מידע גולמי (raw data) ולא נעשה עליו שום עיבוד.

**אופן פעולה:**

הפרויקט מריץ מספר threads, אלו מנוהלים על ידי Qt לא באופן רגיל, לימוד על כך הוא נושא בפני עצמו ולכן נסביר את הרעיון הכללי.

כאשר הפרויקט מופעל נפתח ה – gui ורואים את כל התצוגה.

לאחר מכן יש להריץ את Chaos ולקבל חיווי ש – Chaos ו – RemoteControl מחוברים.(ה – RemoteControl מתייחס ל – Chaos כאל camera כי הוא ממתין ממנו לנתוני מצלמה).

כעת מוזרמים מה – Chaos ל – RemoteControl נתוני המצלמה וחיישן התנועה זמן אמת.

בנוסף בזמן ריצה ניתן להזין את ה – ip וה – port של המחשב שעליו מורץ ה – Chaos ולהתחבר אליו כ – client.

כאשר פעולה זו מצליחה מתקבל חיווי הנקרא Controller.

במצב זה ניתן להשתמש בחיצים כדי לשלוח פקודות ל – Chaos שישלח פקודות לארדואינו שישלח פקודות למנועים.

למעשה יש thread שמאזין ל – Chaos לקבל נתונים ו – thread נוסף שמאזין ללחיצת הכפתורים ושולח את הפקודות ל – Chaos.

כמו כן לפרויקט יש שדה מטיפוס JpegDecompressor הפורס את התמונה שהגיעה ומציג אותה.

**JpegCompressor**

**תיאור המחלקה:**

JpegCompressor היא מחלקה העוטפת את הספרייה של Turbo-jpeg לשימוש מצומצם של דחיסת תמונה על פי האלגוריתם של jpeg.

פונקציות הממשק:

JpegCompressor()

**תיאור הפונקציה:**

הפונקציה יוצרת מופע חדש של המחלקה.

**פרמטרים:**

אין.

**ערך החזרה:**

אין.

JpegCompressor(const uint32 &width, const uint32 &height,

const JpegCompressor::Format &format,

const uint32 &quality\_percent)

**תיאור הפונקציה:**

הפונקציה יוצרת מופע חדש של המחלקה.

**פרמטרים:**

Width – רוחב התמונה בפיקסלים.

Height – גובה התמונה בפיקסלים.

Format – בחירה בין RGB ל – GREY\_SCALE

**ערך החזרה:**

אין.

void setParams(const uint32 &width, const uint32 &height,

const JpegCompressor::Format &format,

const uint32 &quality\_percent);

**תיאור הפונקציה:**

הפונקציה מעדכנת את הפרמטרים אם יש צורך בשינוי בזמן ריצה.

**פרמטרים:**

Width – רוחב התמונה בפיקסלים.

Height – גובה התמונה בפיקסלים.

Format – בחירה בין RGB ל – GREY\_SCALE

**ערך החזרה:**

אין.

void compress(const uint8 \*input)

**תיאור הפונקציה:**

הפונקציה דוחסת את התמונה ושומרת את הפלט.

**פרמטרים:**

Input – מצביע לתמונה, גודל תמונת ה – Input כבר נקבע ב – constructor או על ידי setParams, כמו כן הפלט נשמר בתוך המחלקה וניתן לקבלו על ידי getOutput.

**ערך החזרה:**

אין.

uint64 getCompressedSize()

**תיאור הפונקציה:**

הפונקציה מחזירה את גודל התמונה לאחר דחיסה.

**פרמטרים:**

אין.

**ערך החזרה:**

גודל התמונה בבתים של התמונה שנדחסה, אם לא נדחסה תמונה יוחזר 0.

uint8 \*getOutput()

**תיאור הפונקציה:**

הפונקציה מחזירה את התמונה שנדחסה.

**פרמטרים:**

אין.

**ערך החזרה:**

התמונה הדחוסה, אין צורך לדאוג לניהול זיכרון ואת הגודל ניתן לקבל באמצעות

getCompressedSize.

~JpegCompressor()

**תיאור הפונקציה:**

משחרר את כל המשאבים.

**פרמטרים:**

אין.

**ערך החזרה:**

אין.

**JpegDecompressor**

**תיאור המחלקה:**

JpegDecompressor היא מחלקה העוטפת את הספרייה של Turbo-jpeg לשימוש מצומצם של פריסת תמונה על פי האלגוריתם של jpeg.

פונקציות הממשק:

JpegDecompressor()

**תיאור הפונקציה:**

הפונקציה יוצרת מופע חדש של המחלקה.

**פרמטרים:**

אין.

**ערך החזרה:**

אין.

JpegDecompressor(const uint32 &width, const uint32 &height,

const JpegDecompressor::Format &format)

**תיאור הפונקציה:**

הפונקציה יוצרת מופע חדש של המחלקה.

**פרמטרים:**

Width – רוחב התמונה בפיקסלים.

Height – גובה התמונה בפיקסלים.

Format – בחירה בין RGB ל – GREY\_SCALE

**ערך החזרה:**

אין.

JpegDecompressor &setParams(const uint32 &width, const uint32 &height, const JpegDecompressor::Format &format);

**תיאור הפונקציה:**

הפונקציה מעדכנת את הפרמטרים אם יש צורך בשינוי בזמן ריצה.

**פרמטרים:**

Width – רוחב התמונה בפיקסלים.

Height – גובה התמונה בפיקסלים.

Format – בחירה בין RGB ל – GREY\_SCALE

**ערך החזרה:**

אין.

void decompress(uint8 \*input, const uint64 &compressed\_size)

**תיאור הפונקציה:**

הפונקציה פורסת את התמונה ושומרת את הפלט.

**פרמטרים:**

Input – מצביע לתמונה הדחוסה.

Compressed\_size – גודל התמונה הדחוסה בבתים, הפלט נשמר במחלקה וניתן לקבלו על ידי שימוש ב – getOutput.

**ערך החזרה:**

אין.

int32 getBytesPerPixel()

**תיאור הפונקציה:**

הפונקציה מחזירה כמה בתים מוקצה לכל פיקסל.

**פרמטרים:**

אין.

**ערך החזרה:**

מספר הבתים המוקצה לכל פיקסל( 3 בתים עבור RGB ו – 1 בתים עבור GREY\_SCALE).

uint8 \*getOutput()

**תיאור הפונקציה:**

הפונקציה מחזירה את התמונה שנפרסה.

**פרמטרים:**

אין.

**ערך החזרה:**

התמונה הפרוסה בגודל המקורי שלה, אין צורך לדאוג לניהול זיכרון.

~JpegDecompressor()

**תיאור הפונקציה:**

משחרר את כל המשאבים.

**פרמטרים:**

אין.

**ערך החזרה:**

אין.

**Chaos**

להסביר על המחלקה העיקרית Racecars(לדעתי צריך לשנות שם).

לא צריך לפרט על הפונקציות ומספיק להסביר באותו אופן שבו הסברתי על RemoteControl.

קצת על ה threads ושלמעשה הפרויקט עוטף ומשתמש בכל הממשקים שנכתבו.

לשים את הדגש שהוא משמש כ – server מול ה – RemoteControl לקבלת פקודות למנועים אך גם משמש כ – client אל מול ה – RemoteControl לשליחת מידע מהחיישנים.

כדי שלא יאכלו כאפה תסביר שב – main יש thread שתפקידו היחידי הוא לצאת מהתכנית בצורה נכונה על ידי לחיצה על q.

להוסיף שעסקנו בעיקר בפיתוח ולא בהסעת הרכב אז הפעלנו את התכנית ידנית, אם הם מעוניינים אז תמליץ שהתכנית תופעל אוטומטית בעליית מערכת.

**פערים והמלצות**

* יש לנהל את הפרויקט מהתחלת דרכו ועד סופו בגיט! גם לצרכי ניהול גרסאות, גם לצרכי תמיכה וגם לצרכי גיבוי. אין מה להתחיל לכתוב קוד אם לא יודעים להמהשתמש בפונקציונליות הבסיסית של הגיט.
* המצלמה וה – xavier מופעלות על ידי חברות מתחרות ולכן התאימות בניהן נמוכה. חלק מהתקלות הצלחנו לפתור אך אם אפשרי, אנו ממליצים להחליף את המצלמה.

אחת הבעיות היא שלפעמים היא נתקעת בשידור מפת העומק, דבר הקורה גם ב – gui הרשמי שלהם.

תקלה דומה קרתה גם בשידור תמונת הצבע אך זו נפתרה על ידי הגדלת

ה – buffer של ה – usb ובכלל, על מנת שהספרייה תתמוך נאלצנו

לעדכן kernel ב – xavier.

* ה – speed controller המובנה ברכב פחות נח, יש להדליק אותו בכל הפעלה מחדש והתצורה שבא מוצאים את ה – trigger שלו דרך הארדואינו בעייתית. ישנם בקרים אוטומטיים ודיגטליים שמציעים ממשקים נוחים יותר.
* כרגע יש סוללת מערכת אחת וסוללת מנועים אחת, בסביבת הפיתוח משתמשים לרוב בחשמל אבל כאשר תסיעו את הרכב הרבה מומלץ שיהיה עוד סוללה אחת מכל סוג – כאשר אחת בפעולה, השנייה בטעינה.
* אמנם הסענו את הרכב אך לא הרצנו אלגוריתמיקה ולא דחפנו את ה – xavier לקצה היכולת, במצב זה ההספק המקסימלי שלו הוא 30[W], אל אף שבחישוב ההספק התחשבנו בהספק המקסימלי מומלץ לעבור לסוללה של 20[V] על מנת לעלות את זמן הפעולה.
* המפצל USB-HUB מוזן כרגע על ידי ה – xavier. לאופי הנוכחי של בניית המערכת זה מספיק בהחלט. למצב שבו יתחברו חיישנים נוספים למפצל יש להזינו באמצעות ספק מתח.
* אם אכן יתחברו חיישנים נוספים למפצל, אז כדאי להגדיל את כמות היציאות באמצעות מפצל אחר כיוון שמומלץ להשאיר תמיד 2 יציאות פנויות מחיישנים (למקלדת ולעכבר).