תוכן עניינים

תקציר

כיום ישנה מגמה להעדיף כלי טיס בלתי מאויישים (מזל״ט) על פני מטוסים מאויישים.

למזל״ט יתרונות רבים:

א. בטיחות - אין סיכון לחיי אדם.

ב. עלות - עלות כלי הטיס הבלתי מאוייש נמוכה בהשוואה למטוס רגיל.

ג. ביצועים - מימדיו הקטנים של המזל״ט מאפשרים לו לבצע משימות מגוונות ואיכותיות יותר מכלי טיס רגיל.

אולם המזל״ט מופעל ע״י אדם. ולכן, למרות היתרונות שנמנו לעיל, קיימים בשיטת המפעיל האנושי כמה חסרונות:

א. תקציבים גבוהים - מעבר להשקעה במזל״ט עצמו, נדרשת השקעה של תקציבים גבוהים כדי להקים סביבה תומכת, כגון: סימולטורים, צוות טכני רחב מאוד

ב. הכשרה - יש צורך להכשיר צוות שיטיס את המזל״ט וישמור על כשירות.

ג. טעויות - מזל״ט המופעל ע״י אדם עלול לטעות בעת ביצוע המשימה בהתאם לשיקול דעת מוטעה של מפעילו. לעומת זאת מזל״ט המופעל על פי אלגוריתם מדויק יותר, והסיכוי לטעות נמוך יותר.

מטרת עבודה זו, הנווט האוטונומי, היא לשדרג את מערכת כלי הטיס הבלתי מאוייש הקיימת כיום.

היעדים הם:

א. המזל״ט יפעל באופן אוטונומי ללא התערבות אנושית.

ב. הנווט האוטונומי יסייע לכלי הטיס לבצע את משימתו תוך הימנעות מהיתקלות במכשולי דרך שונים.

ג. המערכת המתוכננת בפרויקט זה מותאמת למזל״ט הפשוט ביותר, שהוא קל וקטן בהרבה ממזל״ט רגיל (ראה בתרשים המצורף בעמוד הבא). לכן המזל״ט יהיה מסוגל לבצע באופן מדויק ומושלם ״משימות איכות״, שאינן ניתנות לביצוע ע״י כלי גדול יותר.

על פי סקר שוק שערכנו - רוב הפרויקטים בתחום עסקו בעיקר בניתוח תמונה ואנו עוסקים בזיהוי מכשולים בעזרת חיישני מרחק ושיערוך GPS.

The UAV in action

תיאור הפרויקט

המערכת המתוכננת כוללת ארבעה מרכיבים עיקריים:

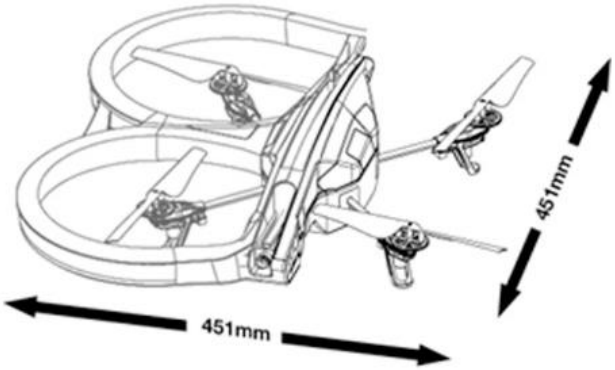
א. AR.Drone 2 - מזל"ט

נתונים כלליים:

1. מהירות מרבית: 11.11 m/s
2. גובה מקסימאלי: 100m

(לאור מגבלות בטיחות של רת"א)

3. מידות: 451x451x62mm
4. משקל: 380g



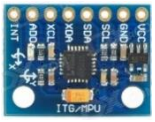
ב. **חיישני מרחק**:

1. מיועד לזיהוי עצמים המהווים סכנה מיידית.
נשתמש בחיישן IR מסוג GP2Y0A02YK של SHARP.
החיישן מזהה עצמים ממרחק 20 ס"מ עד למרחק של 150 ס"מ.
לינק לגיליון נתונים - www.erasme.org/IMG/gp2y0a02_e.pdf

2. מיועד לזיהוי עצמים המהווים מכשול שאינו סכנה מיידית.
נשתמש בחיישן Ultrasonic סוג LV-MaxSonar-EZ0 של MaxBotix.
החיישן מזהה עצמים ממרחק של 15 ס"מ עד למרחק של 500 ס"מ.
לינק לגיליון נתונים - www.maxbotix.com/documents/MB1000_Datasheet.pdf

ג. **רכיב GPS** - מיועד ליידע את המזל"ט על מיקומו הנוכחי.
נשתמש ברכיב Ublox NEO 6M.
לינק לגיליון נתונים -
www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/GY-NEO6MV2-GPS-Module-Datasheet.pdf

ד. **רכיב gyroscope** - מיועד ליידע את המזל"ט באוריינטציה בה הוא נמצא.
נשתמש ברכיב פשוט, זול וקומפקטי: HMC5883L.
לינק לגיליון נתונים -
ftp://imall.iteadstudio.com/Modules/IM130918001/DS_IM130918001.pdf



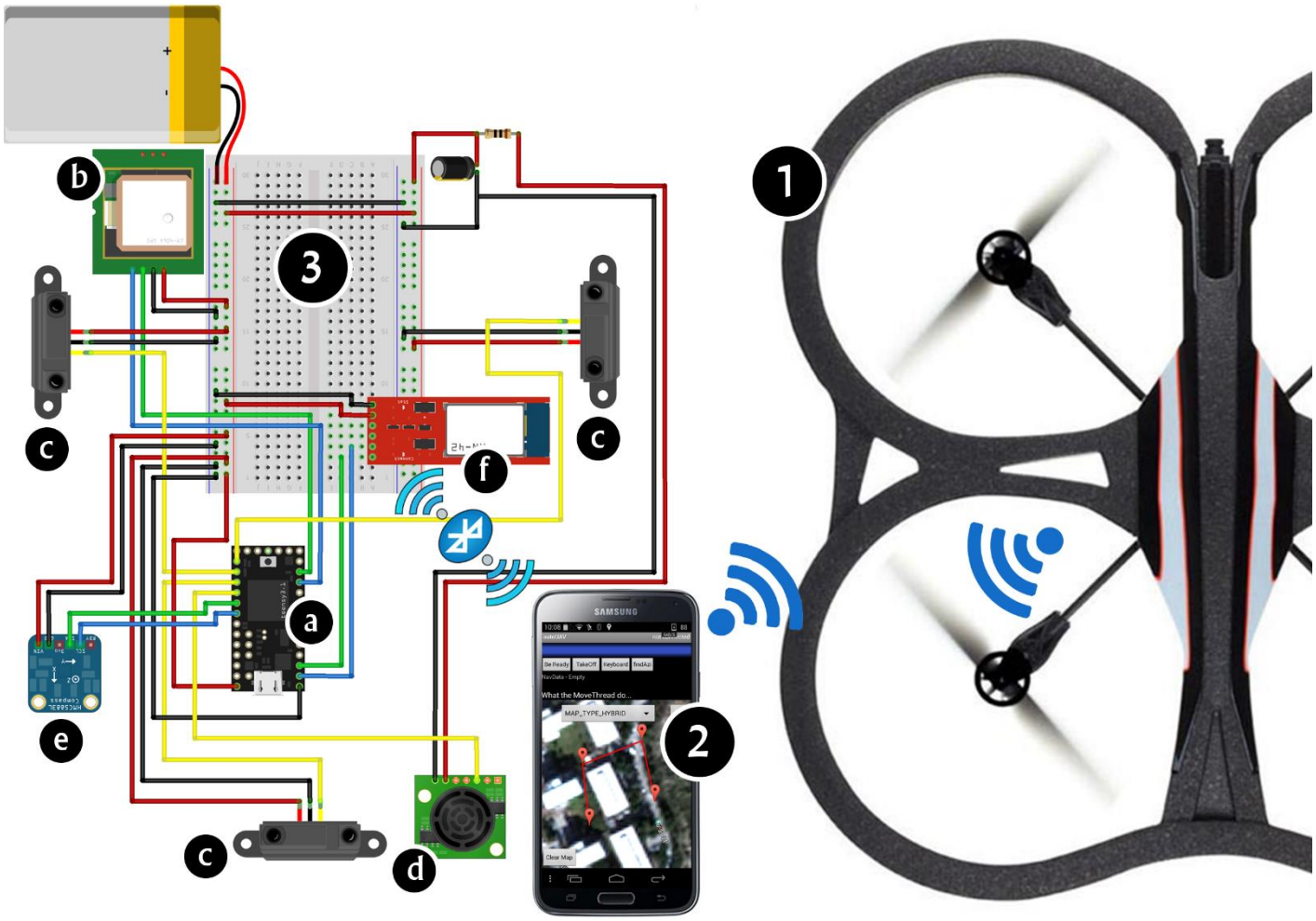
ה. **בקר טיסה** - מקבל את הנתונים מהחיישנים, מה-GPS ומה-Gyro ושולח את הנתונים לטלפון החכם.
נשתמש בבקר מסוג Teensy 3.1, שממדיו קטנים וכוח עיבוד הנתונים שלו חזק יחסית.
לינק לגיליון נתונים - https://www.pjrc.com/teensy/teensy31.html

ו. **מעביר נתונים** - נשתמש ברכיב Bluetooth פשוט וקומפקטי:
JY-MCU, להעברת הנתונים מבקר הטיסה לטלפון החכם.
לינק לגיליון נתונים -

core-electronics.com.au/attachments/guides/Product-User-Guide-JY-MCU-Bluetooth-UART-R1-0.pdf

אפיון המערכת

סכמת המערכת:



מקרא:

1. מזל"ט.
2. טלפון חכם.
3. מערכת ניווט:
   a. בקר טיסה
   b. רכיב GPS
   c. חיישן מרחק (IR)
   d. חיישן מרחק (Sonar)
   e. רכיב אוריינטציה
   f. רכיב Bluetooth

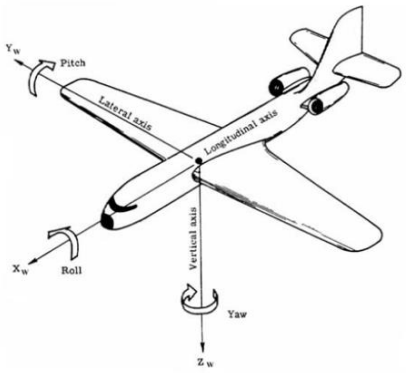
המערכת תפעל באופן הבא:

א. הכרטיס החכם (a) יאסוף את הנתונים הבאים:
   a. מדי מרחק - הסנסורים (c,d) יספקו מידע, אם קיימים עצמים במרחב של המזל"ט.
   b. מיקום - ה:GPS (b) יספק את קו האורך וקו הגובה.
   c. אוריינטציה - רכיב ה:Gyro (e) יספק את האוריינטציה של המזל"ט.

ב. הנתונים מסעיף א' יועברו ב-stream בעזרת רכיב ה-Bluetooth (f) אל רכיב ה:Bluetooth המובנה בטלפון החכם (2).

ג. נתוני NavData ישלחו מהמזל״ט (1) אל הטלפון החכם (2), הנתונים יכללו:

a. גובה מעל פני הקרקע
b. אוריינטציה: pitch ,roll
c. מהירות טיסה
d. מצב סוללה

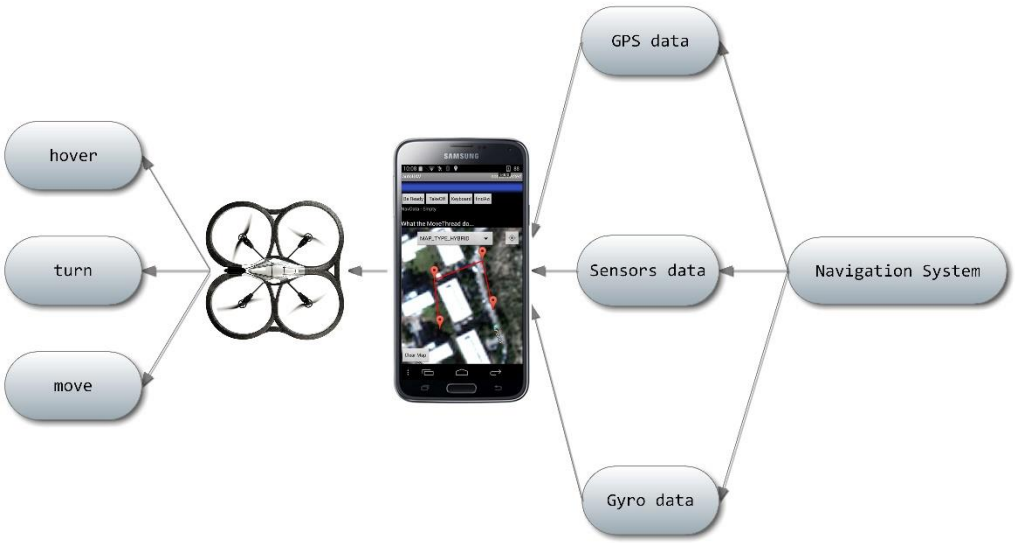


ד. הנתונים מהסעיפים הנ״ל (ג׳-ד׳) יעברו עיבוד בטלפון החכם.

ה. על סמך עיבוד הנתונים, הטלפון החכם ישלח פקודות אל המזל״ט (1) בעזרת רכיב ה:WiFi שמובנה בטלפון החכם.

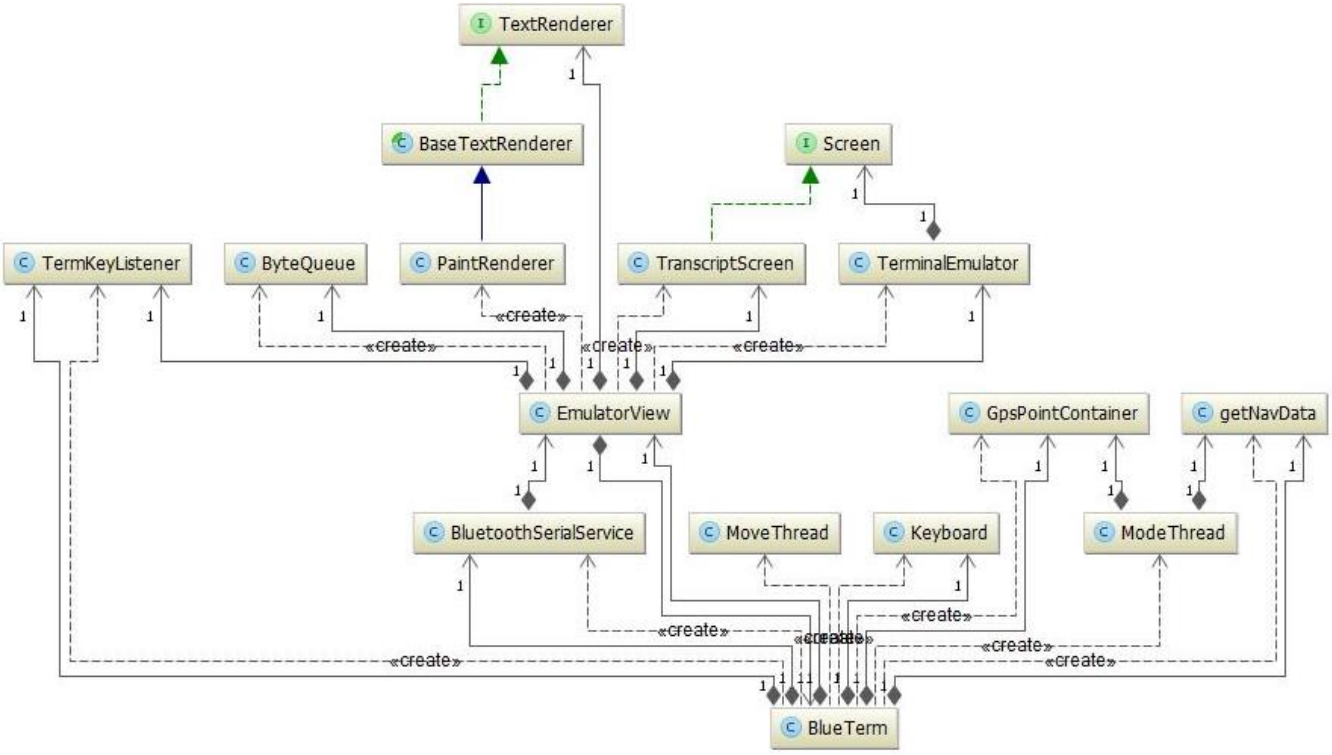
ו. פקודות חירום יעשו בעזרת שימוש בטכנולוגית ה:3G המובנית בטלפון החכם.

use cases:



:UML - Class diagram

דיאגרמה זו אינה כוללת את כל ה:open source השייך ל:AR-Drone, אלא רק את המחלקות אותן אנחנו פיתחנו:



מחלקות נוספות שאינן כלולות בדיאגרמה הן Cords האחראית על חישובי מרחק וזוויות בין 2 נקודות GPS. מחלקה נוספת הינה Function הכוללת פונקציות סטטיות המופעלות במחלקות השונות. (בתת הפרק הבא נרחיב בנושא המחלקות).

מחלקות עיקריות

מחלקת BlueTerm - המחלקה העיקרית של התוכנה. מחלקה זו יוצרת אובייקטים רבים, בין היתר: אובייקט מסוג ARDrone המייצג את המזל"ט עצמו, אובייקט לקבלת המידע מה:Bluetooth שנשלח מהבקר ועוד.

מחלקת MoveTherad - המחלקה יורשת ממחלקת Thread, אובייקט מסוג זה נוצר במחלקה BlueTerm. ברגע שהמזל"ט ממריא נוצר תהליך (אחד ויחיד) האחראי על תנועות המטוס.

תנועת המטוס תלויה ב-4 משתנים:

א. שמאלה - ימינה,
ב. קדימה אחורה,
ג. למטה למעלה,
ד. סיבוב שמאלה סיבוב ימינה.

ארבעת משתנים אלו נשלחים לפונקציה move הקיימת באובייקט המזל"ט ARDrone. אם כל הערכים שווים לאפס - המזל"ט מקבל פקודה לרחף במקום, אחרת - המזל"ט טס על פי הערכים הקיימים במערך:

```
public void run() {
    while (true) {
        try {
            if (Function.isAllZero(move, 4)) drone.hover();
            else
              drone.move(move[0], move[1], move[2], move[3]);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

מחלקת ModeThread - המחלקה יורשת ממחלקת Thread, אובייקט מסוג זה נוצר במחלקה BlueTerm. ברגע שהמזל"ט ממריא נוצר תהליך (אחד ויחיד) האחראי על "מצב" המטוס, וכפי שיוסבר לקמן בפרק "אלגוריתמי המערכת".

<u>מחלקת Function</u> - מחלקה ובה פונקציות סטטיות המופעלות במחלקות השונות, למשל:

```
float[] CutBlueString(String Bluetooth)
```

פונקציה זו מקבלת את ה:stream מה:Bluetooth כמחרוזת ומחזירה מערך של נתונים לפי הסדר הבא:

0,1,2 - מדדים של שלושת הסנסורים (IR),

3 - מדד של הסנסור הקידמי (Max Sonar),

4,5 - קווי אורך ורוחב ב:GPS

6 - מצפן (ערך מספרי בין 0 ל:360, כאשר 0=360=North.

```
void appendLog(String text, String fileName)
```

פונקציה זו יוצרת קובץ Log המאגד בתוכו את כל נתוני הטיסה ונשמר ישירות לזיכרון של הטלפון החכם. על מנת שלא לדרוס את קבצי ה:Log הקיימים - בכל הטסה נוצר קובץ חדש עם שם ייחודי (תאריך ושעה של ההטסה).

<u>מחלקת Cords</u> - את הבסיס למחלקה זו קיבלנו מד"ר בועז בן משה. מחלקה זו כוללת פונקציות סטטיות לחישובים שונים בתחום ה:GPS.

פונקציה לחישוב מרחק ואזימוט (זווית ביחס לצפון) בין 2 נקודות:

```
/**
 * this function computes the azimuth and distance in degrees
 * and meters between two lat/lon points.
 * @return [azm,dist,dz];
 */
public static double[] azmDist(LatLng ll1, LatLng ll2){
   double[] ans = new double[3];
   double[] vec = flatWorldDist(ll1,ll2);
   // 2D for now
   double dist = Math.sqrt(vec[0]*vec[0]+vec[1]*vec[1]);
   double ang = angXY(vec[0], vec[1]);
   ans[0] = ang;
   ans[1] = dist;
   ans[2] = vec[2];
   return ans;
}
```

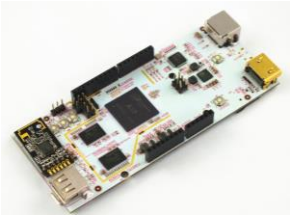
```java
/**
 * this method computes the flat world distance vector between
 * two global points (lat-north, lon-east, alt-above-sea)
 * assuming the two points are relatively close.
*/
public static double[] flatWorldDist(LatLng ll1, LatLng ll2) {
   double[] ans = new double[3];
   double dx = ll2.longitude-ll1.longitude; // delta lon east
   double dy = ll2.latitude-ll1.latitude; // delta lat north
   double dz = 0; // this Data is not used in our program
   if(Math.abs(dx)>0.3 | Math.abs(dy)>0.3) {return null;}
   double x = EARTH_RADIUS * Math.toRadians(dx) *
                     Math.cos(Math.toRadians(ll1.longitude));
   double y = EARTH_RADIUS * Math.toRadians(dy);
   ans[0] = x; ans[1]=y; ans[2] = dz;
   return ans;
}
```

```java
public static double angXY(double dx,double dy){
   double a0 = Math.atan2(dy, dx);
   double ans = rad2Deg(a0);
   return ans;
}
```

האתגר הטכנולוגי ובחירת החומרה

בחירת מזל"ט. חיפשנו את המזל"ט המתאים ביותר לצרכים שלנו, שיהיה קל משקל וקטן. בחרנו ב-AR.Drone.2 מבית Parrot. מפעיל המזל"ט מתחבר אליו באמצעות WiFi. בנוסף, המזל"ט שולח נתונים בזמן אמת אודות הטיסה (גובה, אוריינטציה, מהירות טיסה וכו'). למזל"ט מחוברות 2 מצלמות, קדמית ותחתונה. בעזרת נתוני הטיסה והמצלמות אנו יכולים לנתח בזמן אמת את התנהגות המטוס. למזל"ט האפשרות "להינעל" (Optic Flow) על אובייקט מסוים בקרקע, כך הוא יכול להישאר יציב במהלך הריחוף מעל האובייקט. בנוסף, AR.Drone.2 ניתן לתכנות, וקיים לכך תיעוד מלא ומפורט. כמו כן יש הרבה ספריות לשליטה במזל"ט ברחבי הרשת.

בחירת בקר טיסה. בקר הטיסה הינו המוח המקבל את כל הנתונים מהמזל"ט, מהסנסורים ומה-GPS, ושולח אותם אל הטלפון החכם לביצוע חישובים ע"מ לתת למזל"ט פקודות בכל זמן נתון. יש הרבה בקרים בשוק, ואנו היינו צריכים לבחור בקר שיענה על כל הדרישות: קל לתפעול, בעל ממדים קטנים, בעל דיוק וכוח עיבוד נתונים חזק.


PCDuino 2

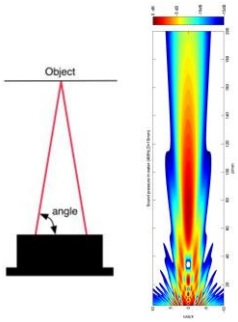
התחלנו להתנסות ב-**PCDuino.2**, כרטיס המכיל בתוכו miniPC וגם Arduino. הוא יכול להריץ מערכת Android ואת מערכת ההפעלה Ubuntu, אך התכונות החזקות שבו לא התאימו לפרויקט.


teensy 3.1v

המשכנו עם **Arduino uno**. בקר זה ענה לדרישותינו, אך רצינו להחליפו בבקר יותר קטן, שאינו נופל ממנו מבחינת כוח העיבוד. לכן בחרנו לעבוד עם **teensy 3.1v** - בקר קטן מאוד, שכוח העיבוד שלו עולה על כרטיסי ה-Arduino.

בחירת חיישני מרחק. יש סוגים רבים של חיישני מרחק. התנסינו בשני סוגים:

1. **IR Sensor** - החיישן שולח קרן אור אינפרא אדום לאובייקט, והקרן מוחזרת לחיישן. החיישן מודד את זווית ההחזרה של קרן האור ומחשב את המרחק מהאובייקט.
2. **Ultrasonic Sensor** - החיישן עובד על עיקרון דומה לרדאר או לסונאר, המעריכים את המרחק מהאובייקט על ידי פירוש ההדים מהרדיו או גלי הקול בהתאמה.



IR vs Ultrasonic

עבור זיהוי של עצמים המהווים סכנה מיידית - בחרנו את חיישני ה-**IR** מכיוון שמצאנו דגם קטן, מדויק יחסית, ומשקלו נמוך. בשביל לזהות מכשול שאינו מהווה סכנה מיידית העדפנו לבחור בחיישן Ultrasonic שמזהה עצמים לטווח גדול יותר.

בחירת רכיב GPS. רכיב ה-GPS מהווה חלק מרכזי במערכת הניווט האוטונומי. כיום יש הרבה רכיבי GPS קטנים ומדויקים. אנו בחרנו ברכיב GPS בשם: Ublox NEO 6M. הוא מאוד מדויק, וקל לתפעול.

שפת פיתוח וסביבת עבודה

נושא זה מתחלק ל-3 חלקים:

א. פיתוח הקוד ל:Teensy 3.1.
הפיתוח נעשה בסביבת עבודה של Arduino עם תוסף שיש להתקין בשביל לעבוד עם הכרטיס החכם.
שפת תכנות: C.

ב. פיתוח הקוד לטלפון החכם מבוסס Android.
הפיתוח נעשה ב: IntelliJ IDEA 14.
שפת תכנות: Java.

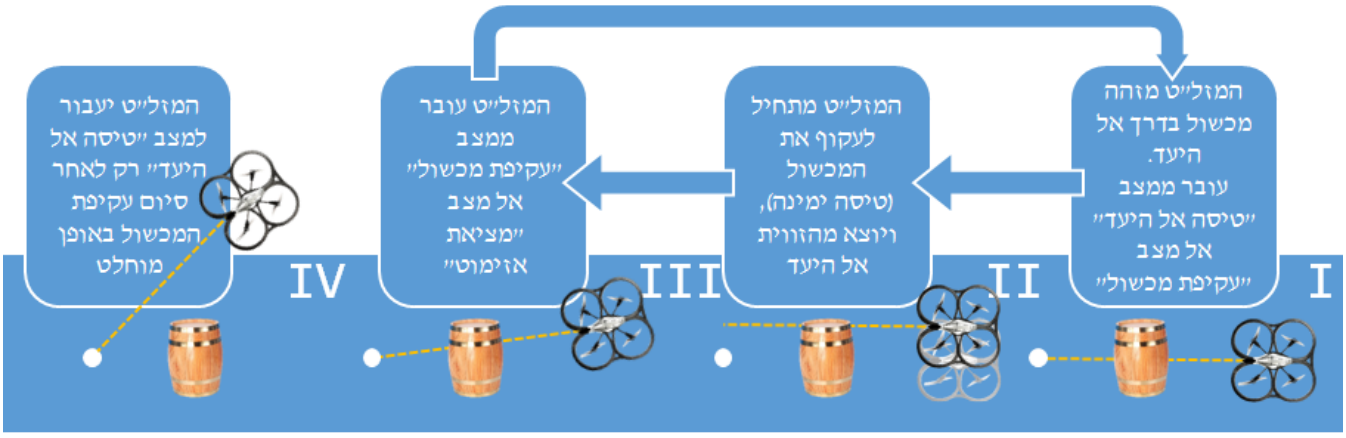
ג. בשביל שנוכל לעבוד על הפרויקט כצוות ולתעד כל שלב בכתיבת הקוד - נעזרנו ב:GIT. קישור לכתובת הפרויקט ב:GitHUB נמצא בעמוד השער.

אלגוריתמי המערכת

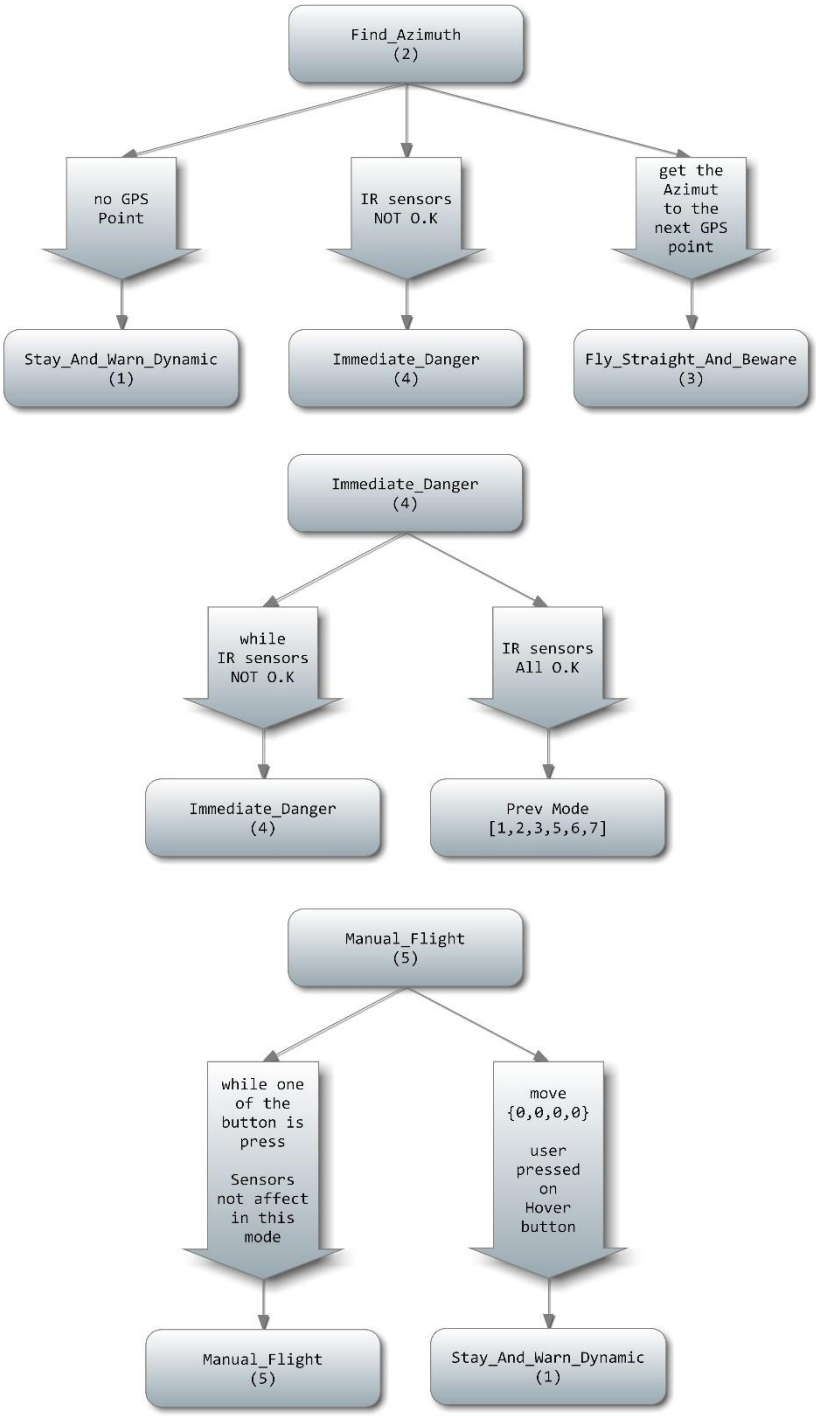
אלגוריתם "מצבים" - modes

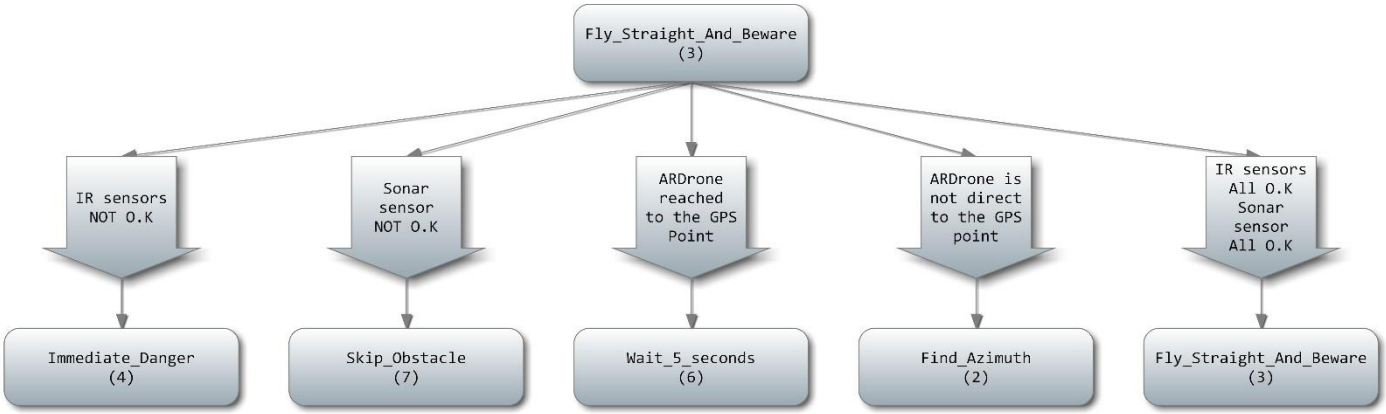
על מנת שהמזל"ט יוכל לנוע בצורה אוטונומית ולהגיב לסובב אותו בצורה נכונה - החלטנו להוסיף "תהליך" (Thread) שיהיה אחראי על "המצב" בו המזל"ט נמצא על פי הנתונים שמתקבלים מהחומרה ומהתוכנה. חילקנו את פעולות המזל"ט ל"מצבים" (modes) שונים:

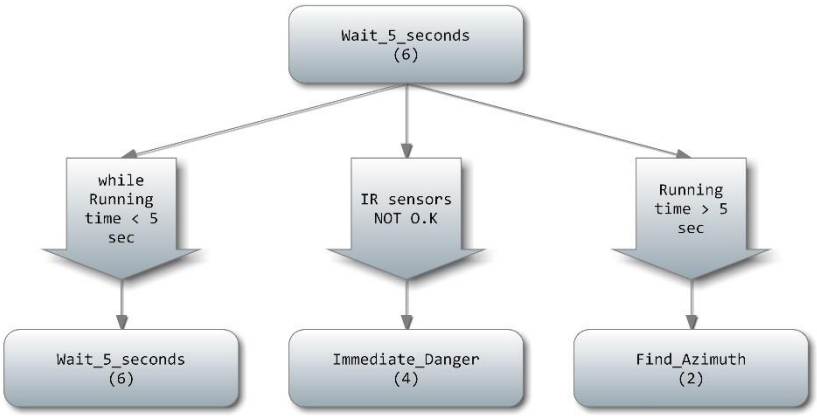
1. מצב "מרחף במקום" Stay_And_Warn_Dynamic
במצב זה המזל"ט יעמוד במקומו ויזהר ממכשולים דינאמיים.
2. מצב "מציאת אזימוט" Find_Azimuth
במצב זה המזל"ט יסתובב במקום (yaw) אל עבר נקודת ה:GPS הראשונה ב:List.
3. מצב "טיסה אל היעד" Fly_Straight_And_Beware
במצב זה המזל"ט יטוס בקו יישר אל היעד אא"כ יתקבל זיהוי של מכשול.
4. מצב "סכנה מיידית" Immediate_Danger
במצב זה המזל"ט יברח מהמכשול באופן מיידי ולאחר שהסכנה תחלוף - יחזור המזל"ט אל ה:mode הקודם בו הוא היה.
5. מצב "טיסה ידנית" Manual_Flight
במצב זה המטוס יתעלם מהנתונים הנשלחים מבקר הטיסה והשליטה עוברת באופן מלא אל המשתמש.
6. מצב "המתנה 5 שניות במקום" Wait_5_seconds
מצב זה נועד בשביל שהמזל"ט ימתין בנקודת ה:GPS אליה הוא הגיע ועד ליציאה אל הנקודה הבאה.
7. מצב "התגברות על מכשול" Skip_Obstacle
במצב זה המזל"ט יעקוף את המכשול העומד מולו ולאחר מכן יחזור למצב "טיסה אל היעד". אם עקיפת המכשול תגרום למזל"ט לצאת מהזווית אל היעד - המטוס יעבור למצב "מציאת אזימוט" וחוזר חלילה, נדגים זאת בעזרת תרשים קצר:

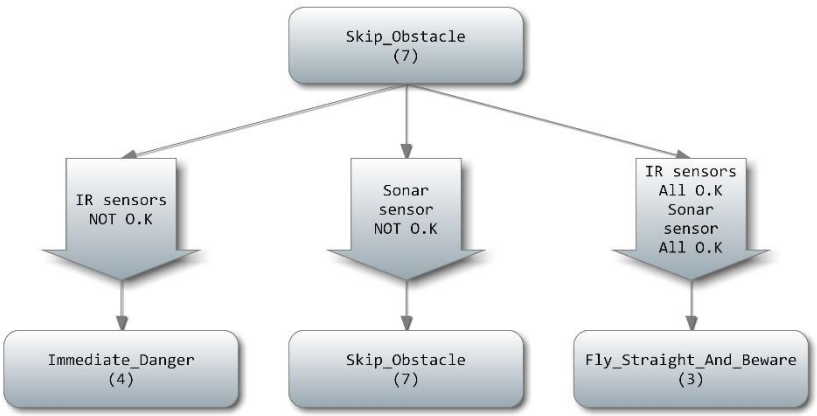


לפני שנעבור לאלגוריתם הכללי, נמפה את המצבים השונים - מאיזה מצב לאיזה מצב ניתן לעבור ומדוע (חץ מסמל את הסיבה למעבר) :




Fly_Straight_And_Beware
(3)
IR sensors
NOT O.K
Sonar
sensor
NOT O.K
ARDrone
reached
to the GPS
Point
ARDrone is
not direct
to the GPS
point
IR sensors
All O.K
Sonar
sensor
All O.K
Immediate_Danger
(4)
Skip_Obstacle
(7)
Wait_5_seconds
(6)
Find_Azimuth
(2)
Fly_Straight_And_Beware
(3)

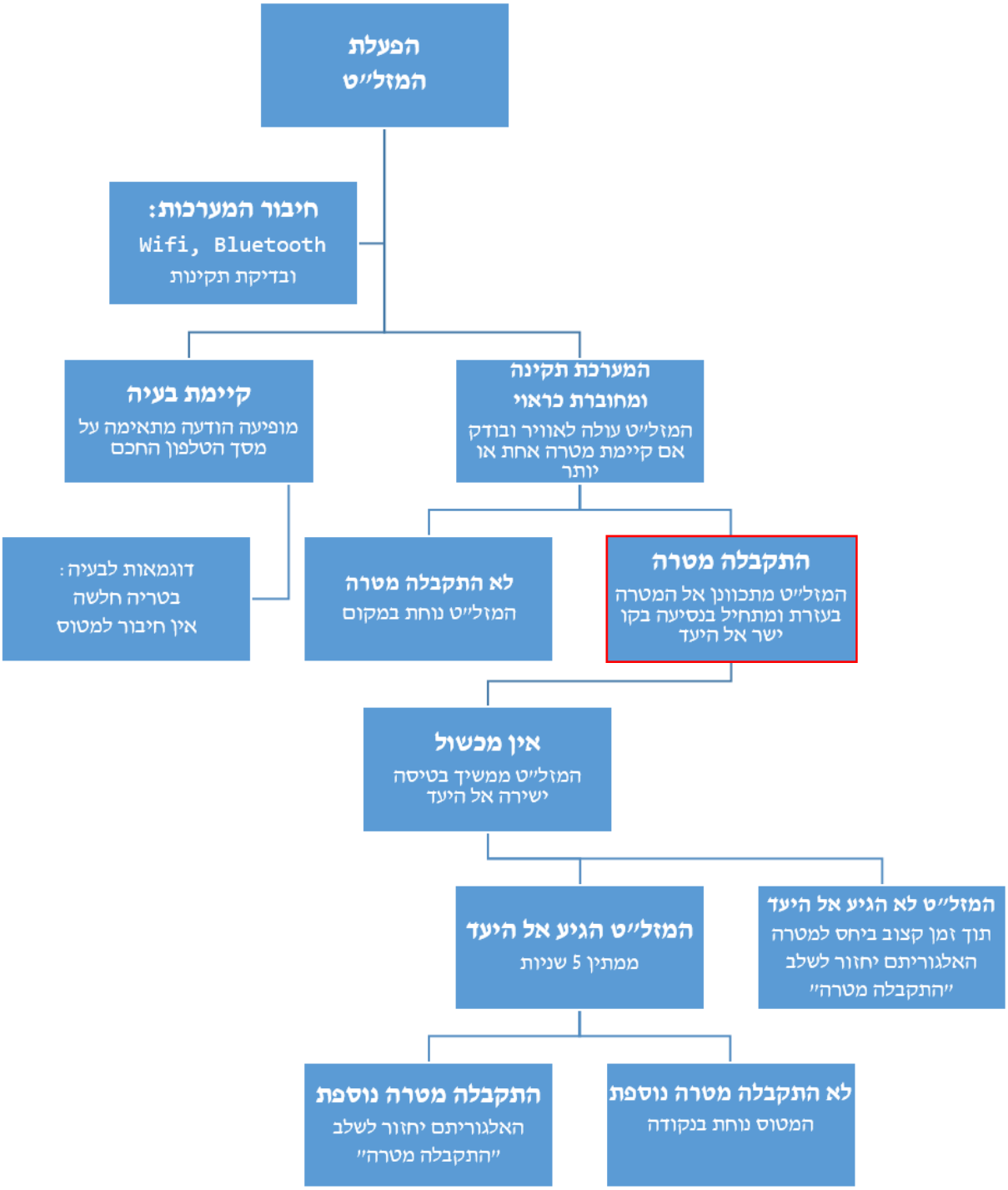

Wait_5_seconds
(6)
while
Running
time < 5
sec
IR sensors
NOT O.K
Running
time > 5
sec
Wait_5_seconds
(6)
Immediate_Danger
(4)
Find_Azimuth
(2)


Skip_Obstacle
(7)
IR sensors
NOT O.K
Sonar
sensor
NOT O.K
IR sensors
All O.K
Sonar
sensor
All O.K
Immediate_Danger
(4)
Skip_Obstacle
(7)
Fly_Straight_And_Beware
(3)

אלגוריתם כולל

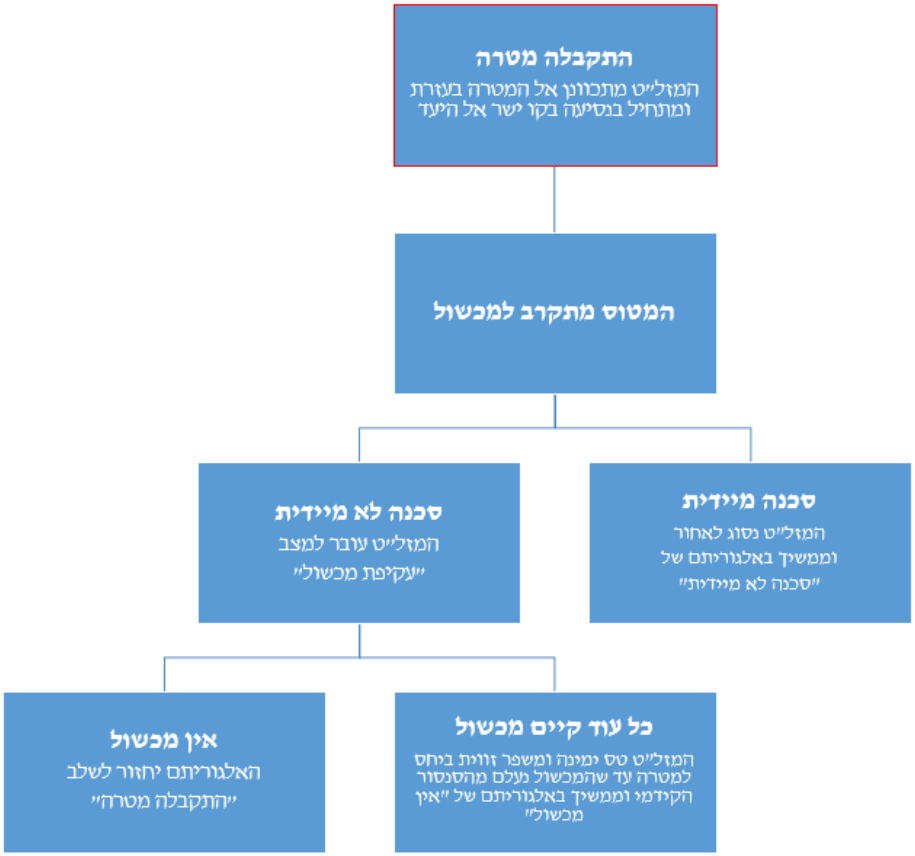
אלגוריתם זה כללי יותר - הוא מתחיל משלב הפעלת המזל"ט ועד לסיום המשימה (מכיל את אלגוריתם המצבים הנ"ל):

חשוב לשים לב: ייתכן שבשלב ראשון לא יהיו מכשולים בדרך אל היעד, ולכן נעבוד לפי תרשים זה. אך בהמשך התרשים ישנם מצבים המחזירים אותנו לשלב "התקבלה מטרה", ושלב זה מתפצל לשניים: ללא מכשולים (תרשים זה), או "עם מכשולים" (התרשים הבא).



תרשים זה הינו המשך לתרשים הקודם (החל מהריבוע האדום). התרשים מתאר את המצב "סכנה" (מיידית / לא מיידית), ואת המצב "התגברות על מכשול".

חשוב לשים לב: ייתכן שבשלב ראשון נגיע למכשול, ולכן נעבוד לפי תרשים זה. אך בהמשך התרשים ישנם מצבים המחזירים אותנו לשלב "התקבלה מטרה", שלב זה מתפצל לשניים: כולל מכשולים (תרשים זה), וללא מכשולים (התרשים הקודם).



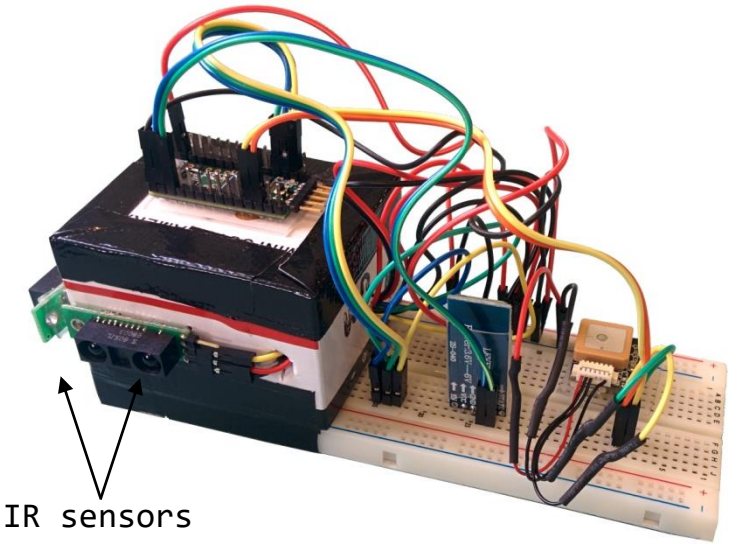
בדיקות וניסויים

ניסוי מס' 1

בניסוי זה בדקנו מעין תחנה סטטית להעברת נתונים מהבקר אל הטלפון החכם.

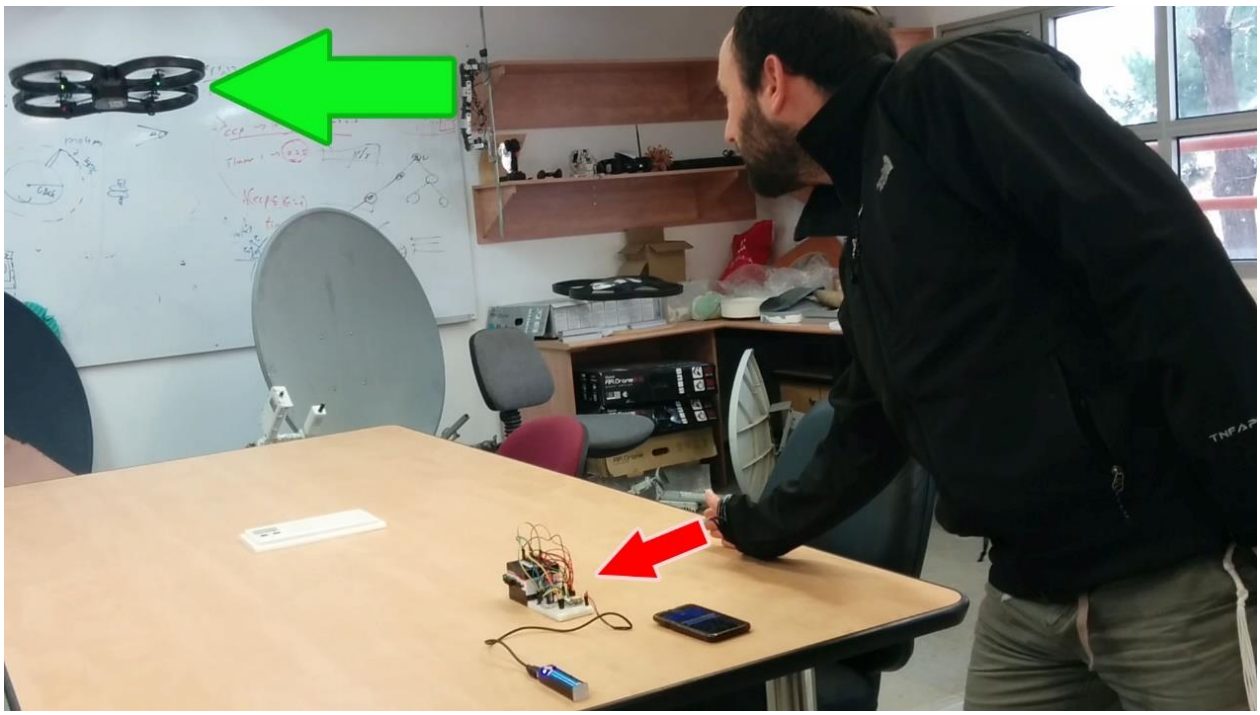
רצינו לבדוק כיצד המזל"ט יגיב לעצמים דינאמיים. הסנסורים (IR) חוברו לקובייה סטטית (ראה איורים).

בכל פעם קרבנו מכשולים אל הקוביה, ובהתאם לכך המזל"ט קיבל פקודות להתחמק מהסכנה "המתקרבת" אליו".



המערכת בפעולה:

חץ אדום - סכנה מתקרבת מימין. חץ ירוק - תגובת המזל"ט, "בריחה" שמאלה.



תוצאות הניסוי:

א. המזל"ט הגיב בצורה טובה לפקודות שנשלחו אליו מהאפליקציה.
ב. תגובת המזל"ט הייתה אגרסיבית מדי.
ג. לאחר שחזרנו על הניסוי מספר פעמים - שמנו לב לכך שהמזל"ט מגיב בצורה לא נכונה ביחס למיקום המכשול.

לקחים להמשך פיתוח:

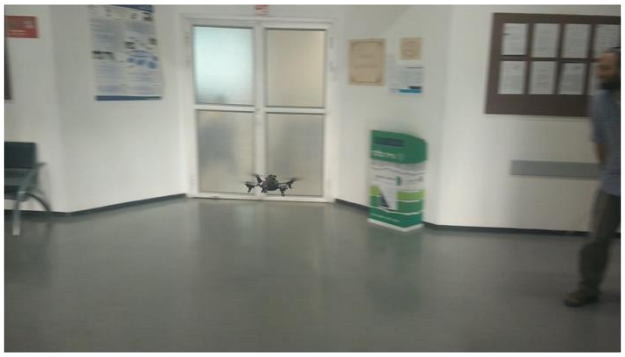
א. בשביל לגרום למזל"ט להגיב בצורה פחות אגרסיבית - יש לשנות את ערך הנטיה המקסימאלי של המזל"ט. ככל שהזוית תהיה קטנה יותר - התגובה תהיה פחות אגרסיבית, אך מצד שני יש להתחשב בעובדה שהמזל"ט צריך להגיב מהר במצבים מסויימים, ולכן נצטרך לבנות מערכת שמשנה את הערך המקסימאלי לפי המצבים השונים.
ב. לאחר חקירה ממושכת, ו:Debug של התוכנה, שמנו לב שה:stream שמגיע מרכיב ה:Bluetooth מגיע קטוע לעיתים רחוקות. בנינו אלגוריתם שמונע ממצב כזה להשפיע על הנתונים.

ניסוי מס׳ 2

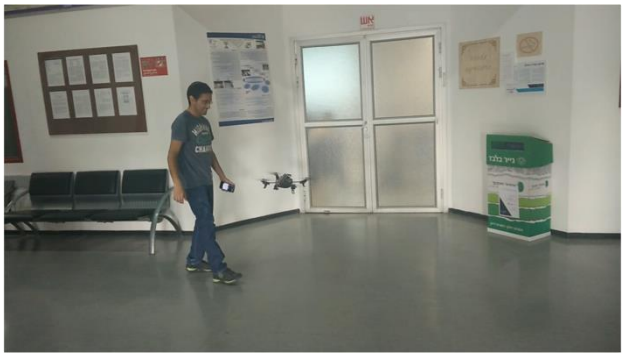
בניסוי זה הרכבנו את המערכת הסטטית מניסוי מס׳ 1, על גבי המזל״ט.

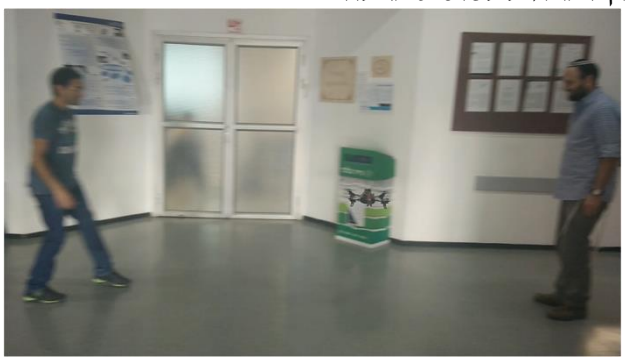
ביצענו בדיקות של עצמים דינאמיים (עצמים המתקרבים אל המזל״ט):

המזל״ט מזהה מכשול משמאל ומתחיל לטוס ימינה



המזל״ט מזהה מכשול מימין ומתחיל לטוס שמאלה





תוצאות הניסוי: המזל״ט הגיב בצורה טובה לפקודות שנשלחו אליו מהאפליקציה.

ניסוי מס' 3

בניסוי זה עשינו בדיקת אמת בתנאי שטח:

תוצאות הניסוי:

IR Sensors רגישים לאור השמש, מה שגורם להם להעביר נתונים לא נכונים ביחס למכשולים הקיימים סביבם.

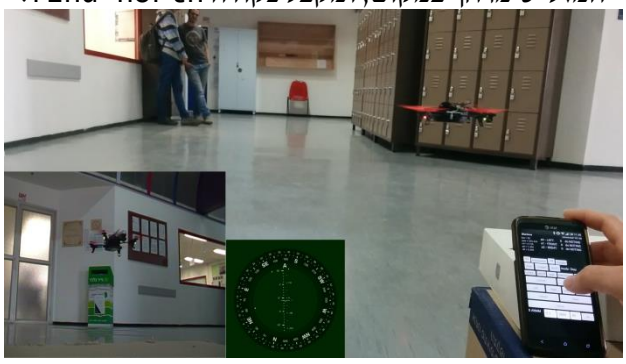
לקחים להמשך פיתוח:

יש לשקול את החלפת הסנסורים ל:Ultrasonic Sensors.

ניסוי מס' 4

בניסוי זה בדקנו האם האלגוריתם למציאת הצפון בדרך הקצרה ביותר אכן עובד. אם הטלפון החכם ידע היכן נמצא הצפון ביחס אליו - הוא יוכל לחשב ולדעת איפה נמצאת נקודת היעד. (ראה בהרחה - אפיון המערכת, מחלקות עיקריות).

המזל"ט מרחף במקום, ומקבל פקודה find-north.

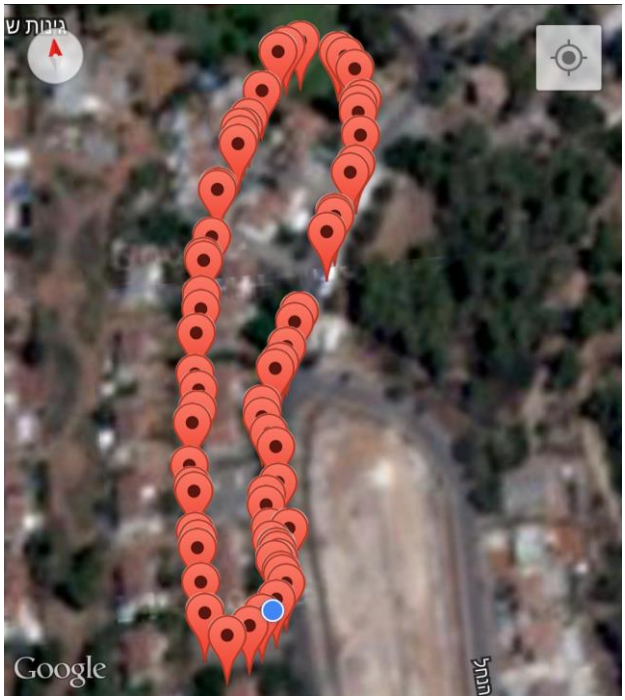


המזל"ט מסתובב (בדרך הקצרה) ומוצא את הצפון.

תוצאת הניסוי: המזל"ט הגיע לצפון בהצלחה!

ניסוי מס׳ 5

בניסוי זה בדקנו את מערכת הניווט:



בתחילה בדקנו שהמערכת משערכת את מיקום המזל״ט בצורה נכונה (ראה איור).

לאחר שהמערכת נמצאה תקינה, הוספנו אפשרות להגדיר נקודות ציון באפליקציה, כך שהמזל״ט יטוס אליהן לפי סדר יצירתן (ראה פירוט עמ׳ 24).

הניסוי בוצע ללא הטסת המזל״ט, אלא באופן ידני - האפליקציה הציגה את הפקודות שיש לבצע, והמפעיל ביצען.

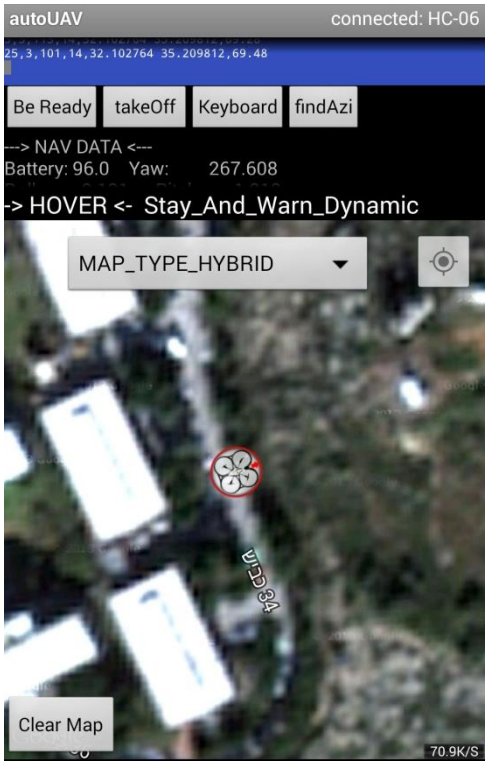
ניסוי זה - כלל בתוכו את ניסויים 1-4, הפקודות שהוצגו, כללו - מעבר מכשול, מציאת אזימוט והגעה אל היעד.

תוצאות הניסוי: האלגוריתם שיצרנו לטיסה בין 2 נקודות GPS עובד היטב.

מימוש התוכנה מאב טיפוס ועד היום

האפליקציה שפיתחנו בנויה ביסודה על אפליקציה בשם BlueTerm שזמינה לשימוש חופשי.
קישור לקוד מקור: https://github.com/johnhowe/BlueTerm.

אפליקציה זו מאפשרת לשלוח ולקבל תשדורות Bluetooth בצורה קלה ויעילה.
על בסיס אפליקציה זו הוספנו את המודולים של המזל"ט.



הנתונים שמגיעים מה: Bluetooth כ: stream

כפתורים לשליטה במזל"ט

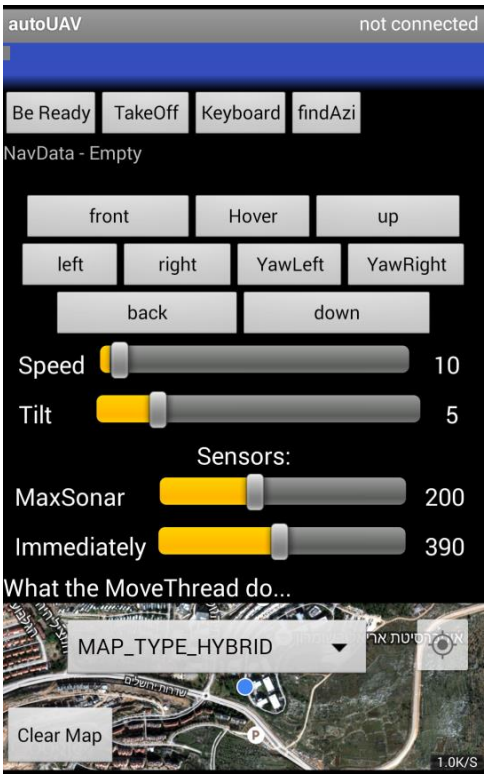
נתוני המזל"ט (ניתן לגלול ולקבל נתונים נוספים)

ה: mode הנוכחי בו נמצא המזל"ט

אפשרות לשינוי סוג המפה - Hybrid\Terrain

המפה מציינת את מיקומו של המזל"ט, כולל כיוון טיסה (מסומן חץ אדום).

כפתור המיועד למחיקת המסלול המתוכנן (יוסבר להלן)



בלחיצה על הכפתור Keyboard - מוצגות הגדרות נוספות:

א. הטסה ב: mode "שליטה ידנית".

ב. קביעת מהירות הטיסה

ג. גודל זווית מקסימאלי

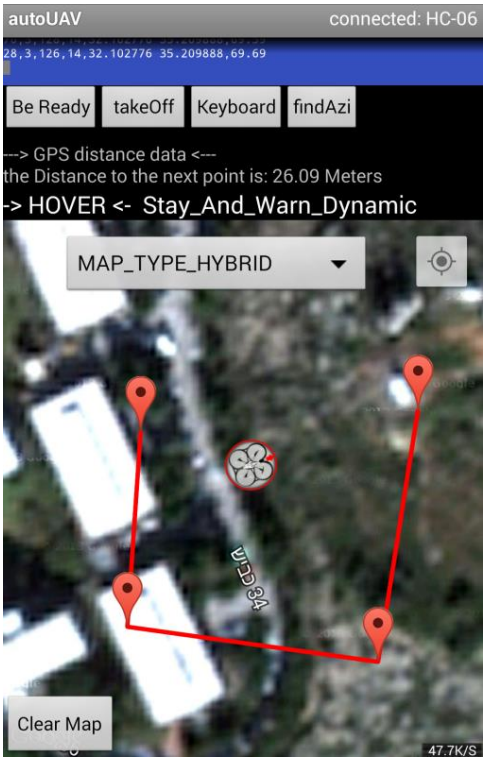
ד. טווח זיהוי של הסנסורים

מצב האפליקציה לאחר תכנון מסלול:

נתוני GPS ביחס לנקודה היעד הראשונה.

תכנון מסלול בעל 4 נקודות ציון. ניתן לשנות ולהסיר כל נקודה, והמסלול יתעדכן בהתאמה.

כפתור המאפשר למחוק את כל נקודות הציון בלחיצה אחת.



המשך המחקר והפיתוח

ישנן כמה נקודות הטעונות שיפור ושדרוג:

1. עיבוד תמונה: כאמור, הפרויקט הנ״ל לא משלב בתוכו עיבוד תמונה, על אף שלמזל״ט Ardrone 2 קיימות 2 מצלמות מובנות. עיבוד תמונה יכול לשפר מאוד את האלגוריתם למעבר מכשולים, ולבצע משימות נוספות:
   a. זיהוי מטרה במרחב על סמך צבע או צורה.
   b. קבלת פקודות שונות על ידי זיהוי ברקודים (כגון: זיהוי מיקום לנחיתה וכדומה).
   c. מעקב אחרי עצמים דינאמיים.
   d. שיערוך מרחקים.

2. כאמור, חיישני המרחק המותקנים כרגע על המזל״ט אינם מתפקדים בצורה מלאה. ישנן כמה וכמה אופציות חלופיות, למשל: LIDAR-Lite v2. חיישן רב עוצמה המזהה עצמים למרחק של כ:40 מטר, קל משקל ומדויק מאוד.

נקודה נוספת שיש לשים עליה את הדעת: המזל״ט Ardrone 2 מוגבל מאוד במשקל המקסימאלי אותו הוא יכול לשאת. על כן המערכת צריכה להיות קלה ומצומצמת ככל האפשר. המערכת כרגע נמצאת בסטטוס של "overflow", ויש צורך לצמצמה (כגון: טלפון חכם קל משקל, רכיבים קלים וקטנים יותר).

סקר ספרות

1. **Project Name**: Node Copter
   Control of the AR.Drone using node.js.
   Nodecopter team create client protocol with Node.js, platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications
   Link: http://www.nodecopter.com/

2. **Project Name**: Autonomous People Tracking using AR-Drone 2.0
   Using the front camera so that the plane will follow a certain character, People detection is done using Histogram of Gradient (HOG) feature and tracking with a particle filter.
   Link: https://www.youtube.com/watch?v=SJY1zxVtWsU

3. **Project Name**: ARDrone Target Tracking with OpenCV - OpticaFlow
   Using optic flow to follow a signal
   Link: https://www.youtube.com/watch?v=C95bngCOv9Q

4. **Project Name**: Helping Robots See
   Explore computer-vision-based algorithms and machine-learning methods to help an ARDrone quad-rotor helicopter autonomously navigate
   Link: https://www.youtube.com/watch?v=o_O2o2ly-34&list=PL63FC304EF5263230&index=13

5. **Project Name**: AR Drone Face Tracking
   Quadcopter programmed to detect and follow a face. A Haar feature detector is used to identify a face and a camshift algorithm is used to track the face. A controller reorients the quadcopter to center the face within the video frame.
   Link: https://www.youtube.com/watch?v=VghVtljvWew

6. **Project Name**: quadcopter project
   Obstacle Avoidance System For Parrot AR.Drone Quadcopter, very good work with the IR sensors. Work with Node.js and JavaScript.
   Links: https://tsouthprojects.wordpress.com/category/2-quadcopter/,
   https://www.youtube.com/watch?v=WUs2IV1kdU

7. **Project Name**: Extended Wifi Signal Antennae kit serves to increase wifi range and reliability. The antenna is connected directly to the drone and increases the WiFi range. It seems in Ardrone2 no antenna connector as in Ardrone 1.
   Link: http://ardroneshow.com/ar-drone-2-0-modded-main-board-wifi-antennae-kit/

8. **Project Name**: Autonomous robotic plane flies indoors
   The MIT researchers have completed a series of flight tests in which an autonomous robotic plane running their state-estimation algorithm successfully threaded its way among pillars in the parking garage under MIT's State Center.
   Link: http://newsoffice.mit.edu/2012/autonomous-robotic-plane-flies-indoors-0810

Sensors

9. **Sharp infrared IR ranger comparison**
   Sharp infrared detectors and rangers boast a small package, very low power consumption and a variety of output options. In order to maximize each sensor's potential, it is important to understand how these types of IR sensors work, their effective ranges, and how to interface to them.
   Link: http://www.acroname.com/articles/sharp.html

10. **Ultrasonic Sensors**
   Datasheets, Application Notes, Pictures, & Other Documents
   This site has tutorials on how to operate the sensors. In addition, there are programs that show how to work with Arduino cards.
   Link: http://www.maxbotix.com/downloads.htm

GPS

11. **u-center GNSS evaluation software**
   The u-center GNSS evaluation software provides a powerful tool for evaluation, performance analysis and configuration of u-blox GNSS receivers. Its unique flexibility makes the u-center

GNSS evaluation software an invaluable tool for evaluation, analysis and configuration of u-blox GNSS receivers. u-blox GNSS receivers can be configured using the u-center evaluation software.
Link: http://www.u-blox.com/en/evaluation-tools-a-software/u-center/u-center.html

12. NMEA Parser - Java Language
Use for parse the GPS data.
Link:
https://github.com/HvB/UsbGps4Droid/blob/master/src/org/broeuschmeul/android/gps/nmea/util/NmeaParser.java