תוכן עניינים

1. תקציר 3
2. תיאור הפרויקט 4
3. אפיון המערכת
   1. סכמת המערכת 5
   2. use cases 6
   3. UML class diagram 7
   4. מחלקות עיקריות 8
4. פלטפורמת הפיתוח - האתגר הטכנולוגי ובחירת החומרה 0
5. אלגוריתמי המערכת
   1. אלגוריתם "מצבים" - modes 00
   2. אלגוריתם כולל 00
6. מימוש התוכנה מאב טיפוס ועד היום 00

תקציר

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

למזל"ט יתרונות רבים:

1. בטיחות - אין סיכון לחיי אדם.
2. עלות - עלות כלי הטיס הבלתי מאוייש נמוכה בהשוואה למטוס רגיל.
3. 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:

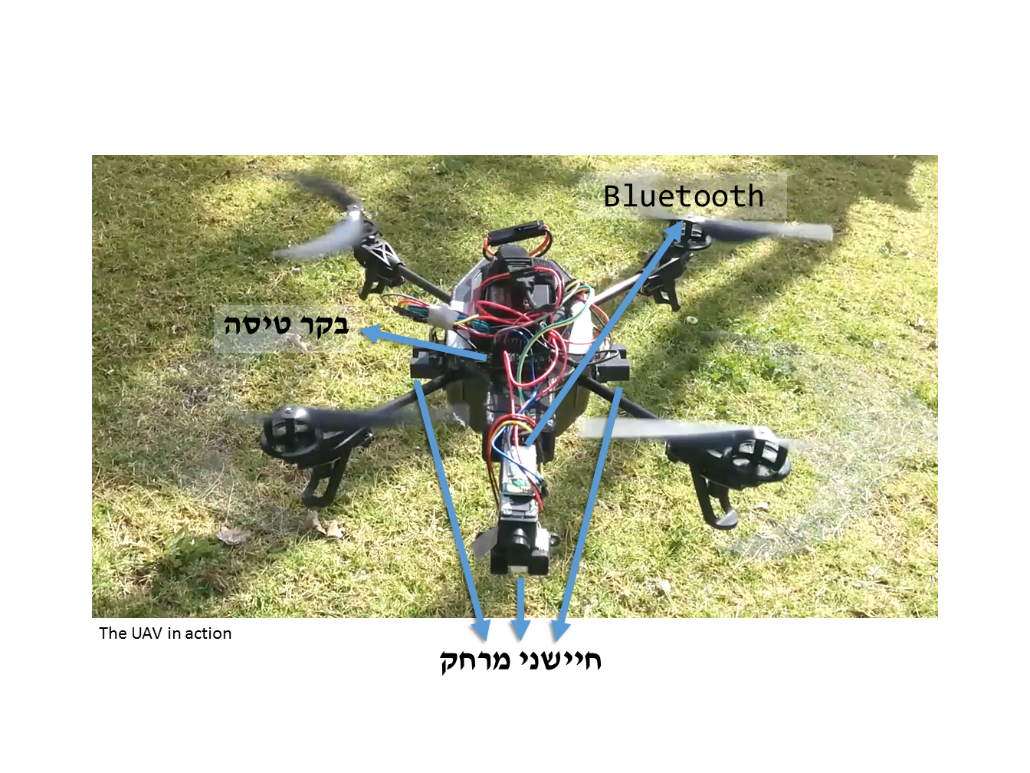
1. תקציבים גבוהים - מעבר להשקעה במזל"ט עצמו, נדרשת השקעה של תקציבים גבוהים כדי להקים סביבה תומכת, כגון: סימולטורים, צוות טכני רחב מאוד
2. הכשרה - יש צורך להכשיר צוות שיטיס את המזל"ט וישמור על כשירות.
3. טעויות - מזל"ט המופעל ע"י אדם עלול לטעות בעת ביצוע המשימה בהתאם לשיקול דעת מוטעה של מפעילו. לעומת 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.

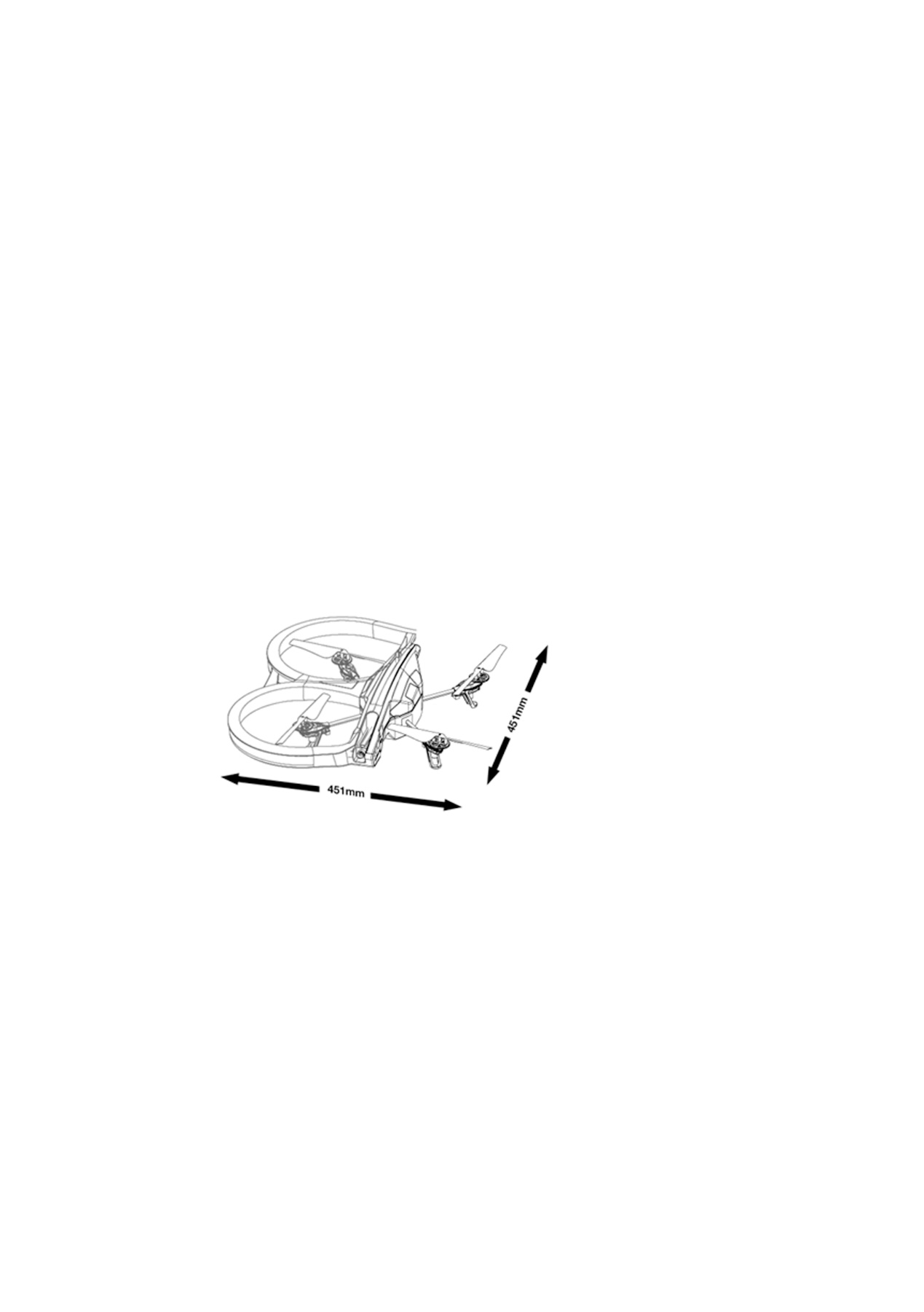
היעדים הם:

1. המזל"ט יפעל באופן אוטונומי ללא התערבות אנושית.
2. הנווט האוטונומי יסייע לכלי הטיס לבצע את משימתו תוך הימנעות מהיתקלות במכשולי דרך שונים.
3. המערכת המתוכננת בפרויקט זה מותאמת למזל"ט הפשוט ביותר, שהוא קל וקטן בהרבה ממזל"ט רגיל (ראה בתרשים המצורף). 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 בעיקר בזיהוי מכשולים בעזרת חיישני מרחק ושיערוך GPS.

****

תיאור הפרויקט

המערכת המתוכננת כוללת ארבעה מרכיבים עיקריים:

1. **AR.Drone 2 - מזל"ט**

נתונים כלליים:

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

(לאור מגבלות בטיחות של רת"א)

* 1. מידות: 451x451x62mm
  2. משקל: 380g

1. **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

* 1. 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

1. **רכיב GPS** - מיועד ליידע את הרחפן 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

****

1. **רכיב gyroscope** - מיועד ליידע את הרחפן באוריינטציה בה הוא נמצא.

נשתמש ברכיב פשוט, זול וקומפקטי: HMC5883L.

****לינק לגיליון נתונים - ftp://imall.iteadstudio.com/Modules/IM130918001/DS\_IM130918001.pdf

1. **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

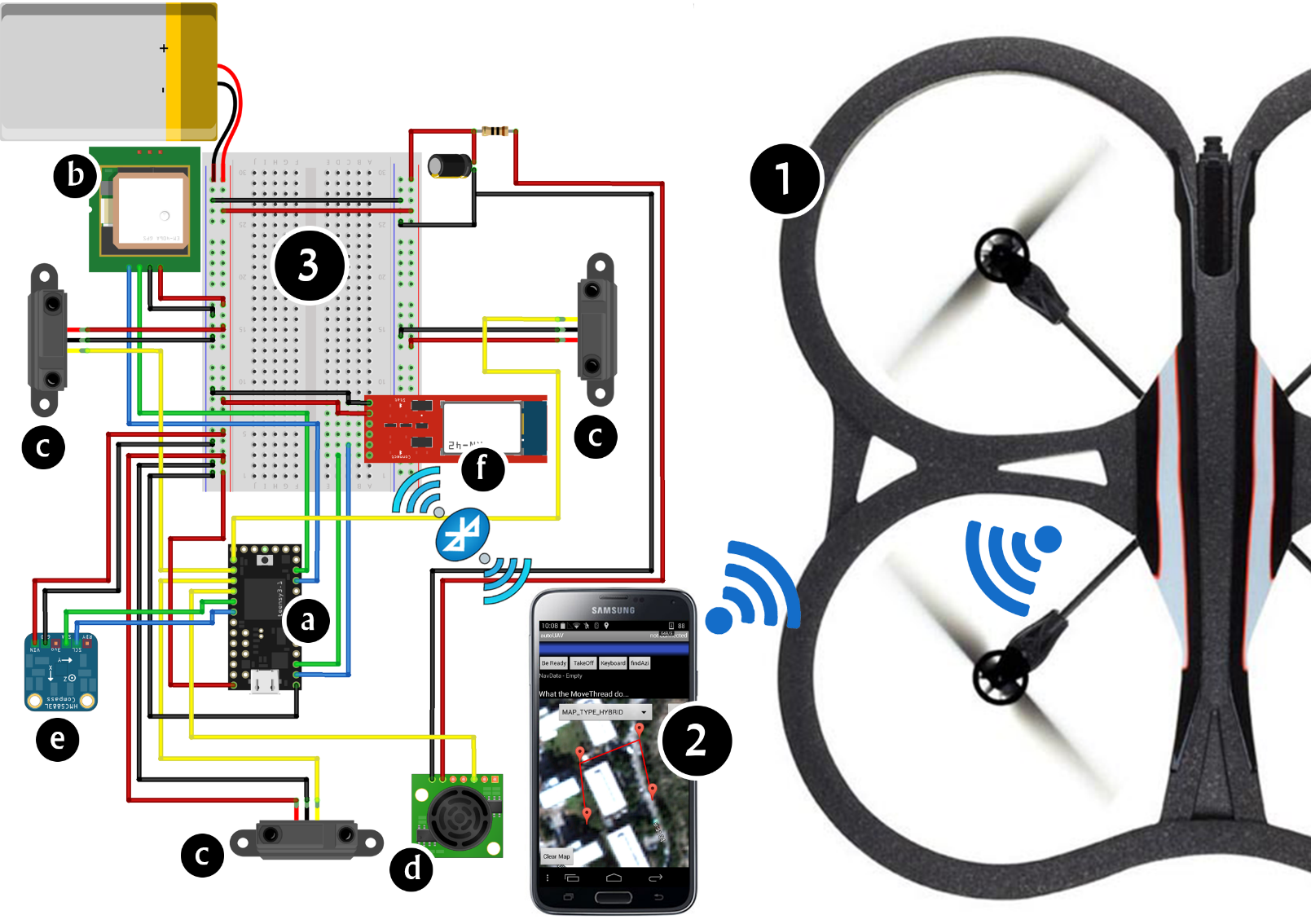
1. **מעביר נתונים** - נשתמש ברכיב Bluetooth פשוט וקומפקטי:

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

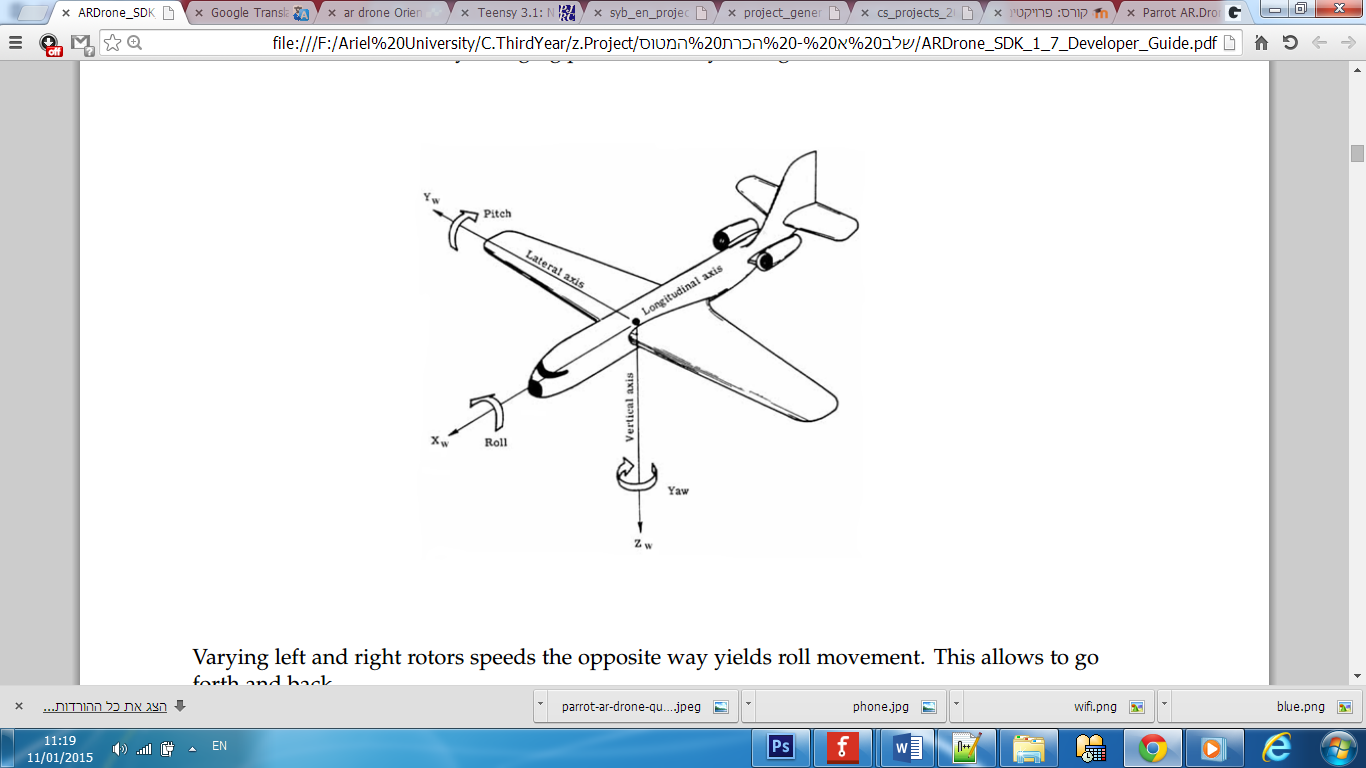
אפיון המערכת

סכמת המערכת:

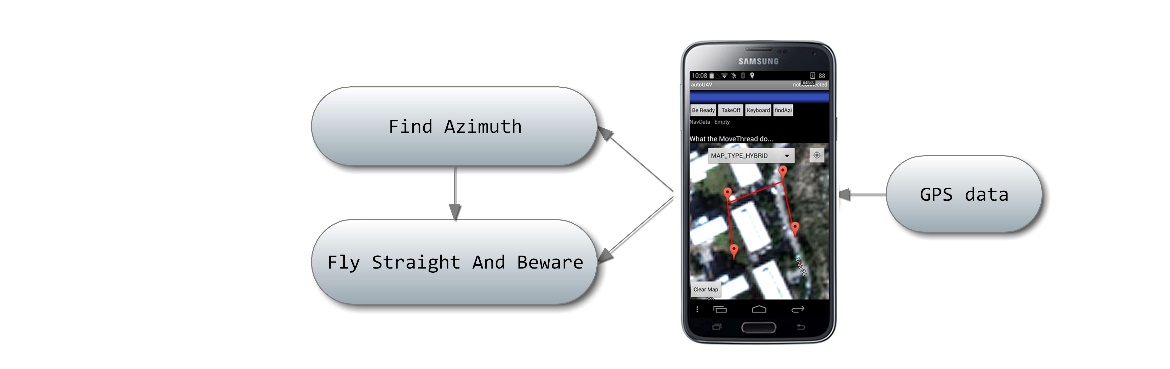
מקרא:

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

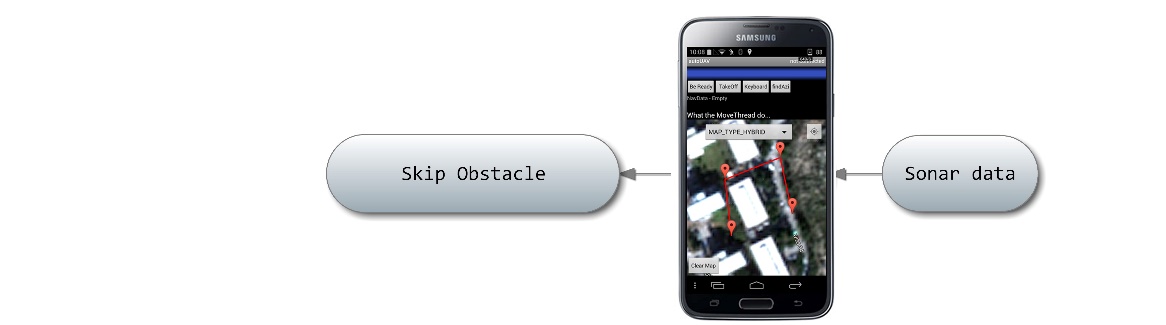
המערכת תפעל באופן הבא:

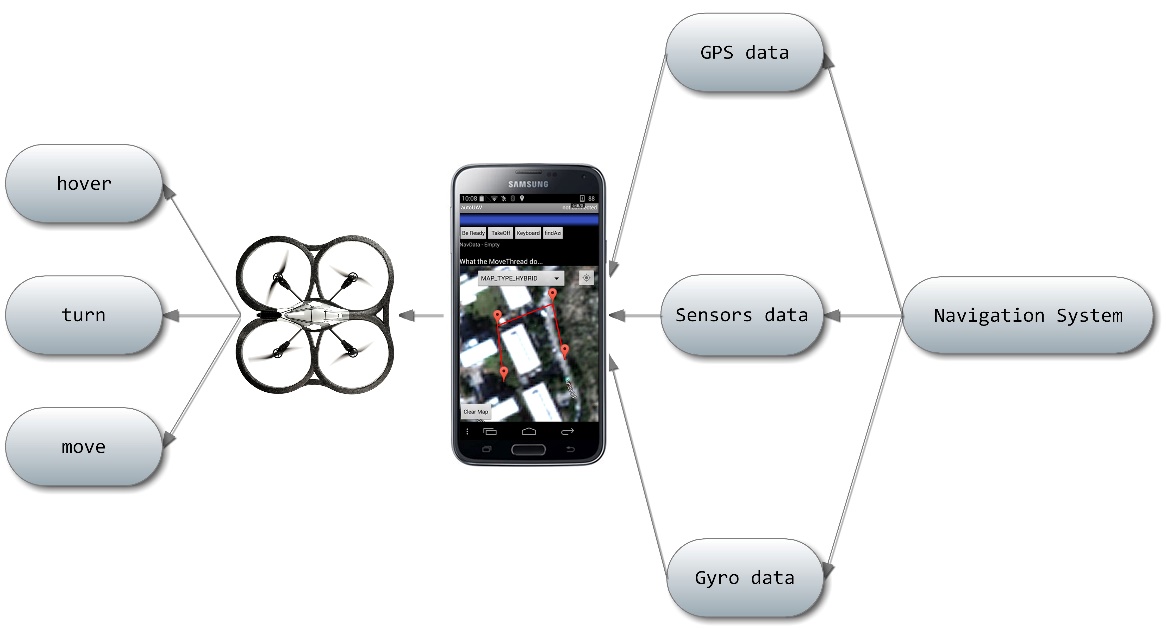
1. הכרטיס החכם (a) יאסוף את הנתונים הבאים:
   1. מדי מרחק - הסנסורים (c,d) יספקו מידע, אם קיימים עצמים במרחב של הרחפן.
   2. מיקום - ה:GPS (b) יספק את קו האורך וקו הגובה.
   3. אוריינטציה - רכיב ה:Gyro (e) יספק את האוריינטציה של הרחפן.
2. הנתונים מסעיף א' יועברו ב-stream בעזרת רכיב ה-Bluetooth (f) אל רכיב ה:Bluetooth המובנה בטלפון החכם (2).
3. נתוני NavData ישלחו מהרחפן (1) אל הטלפון החכם (2), הנתונים יכללו:
   1. גובה מעל פני הקרקע
   2. אוריינטציה: roll, pitch
   3. מהירות טיסה
   4. מצב סוללה
4. הנתונים מהסעיפים הנ"ל (ג'-ד') יעברו עיבוד בטלפון החכם.
5. על סמך עיבוד הנתונים, הטלפון החכם ישלח פקודות אל הרחפן (1) בעזרת רכיב ה:WiFi שמובנה בטלפון החכם.
6. פקודות חירום יעשו בעזרת שימוש בטכנולוגית ה:3G המובנית בטלפון החכם.

use cases:

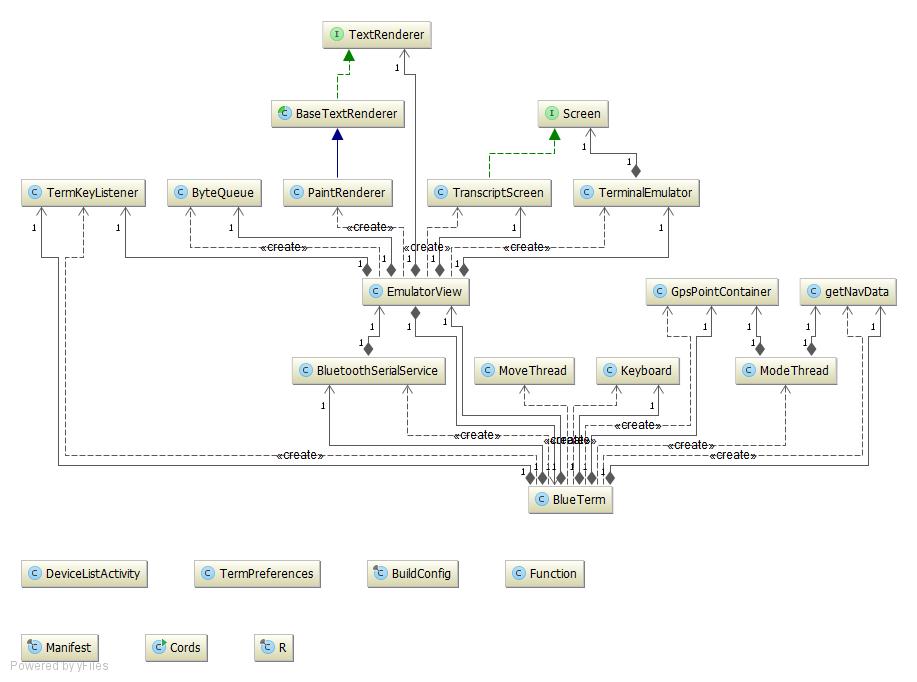








UML - Class diagram:

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, אובייקטים לקבלת המידע ב:Bluetooth שנשלח מהבקר ועוד.

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

תנועות המטוס תלויות ב-4 משתנים (שמאלה - ימינה, קדימה אחורה, למטה למעלה, סיבוב שמאלה סיבוב ימינה) הנשלחים לפונקציה move הקיימת באובייקט המזל"ט ARDrone. המחלקה מקבלת מערך בגודל 4 איברים המייצג את המשתנים הנ"ל. אם כל הערכים שווים לאפס - המזל"ט מקבל פקודה לרחף, 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, וברגע שהמזל"ט ממריא נוצר תהליך (אחד ויחיד) האחראי על "מצב" המטוס, וכפי שיוסבר לקמן בפרק "אלגוריתמי המערכת".

מחלקת Function - 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 הקטנים (IR), 3 - מדד של הסנסור הגדול, 4,5 - נקודת GPS (אורך רוחב), 6 - מצפן.

void appendLog(String text, String fileName)

פונקציה זו יוצרת קובץ Log המאגד בתוכו את כל נתוני הטיסה ונשמר ישירות לזיכרון של הטלפון החכם.

מחלקת Cords - 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;  
}

*/\*\*  
 \* 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;  
}

**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 כל החישובים 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.

|  |
| --- |
| https://www.pjrc.com/teensy/teensy31_front_small_green.jpg |
| 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.

|  |  |  |
| --- | --- | --- |
| C:\Users\Naor\Desktop\irVSultrasonic.jpg | | |
| Ultrasonic | vs | IR |

1. **Ultrasonic Sensor** - החיישן עובד על עיקרון דומה לרדאר או לסונאר, המעריכים את המרחק מהאובייקט על ידי פירוש ההדים מהרדיו או גלי הקול בהתאמה.

עבור זיהוי של עצמים המהווים סכנה מיידית - בחרנו את חיישני ה-**IR** מכיוון שמצאנו דגם ספציפי מאוד 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:

1. פיתוח הקוד ל:Teensy 3.1.

הפיתוח נעשה בסביבת עבודה של Arduino עם תוסף שיש להתקין בשביל לעבוד עם הכרטיס החכם.

שפת תכנות: C.

1. פיתוח הקוד לטלפון החכם מבוסס Android.

הפיתוח נעשה ב:IntelliJ IDEA 14 .

שפת תכנות: Java.

1. בשביל שנוכל לעבוד על הפרויקט כצוות ולתעד כל שלב בכתיבת הקוד - נעזרנו ב:GIT. קישור לכתובת ה: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.

1. מצב "מציאת אזימוט" Find\_Azimuth

במצב זה המזל"ט יסתובב במקום (yaw) אל עבר נקודת ה:GPS הראשונה ב:List.

1. מצב "טיסה אל היעד" Fly\_Straight\_And\_Beware

במצב זה המזל"ט יטוס בקו יישר אל היעד אא"כ יתקבל זיהוי של מכשול.

1. 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.

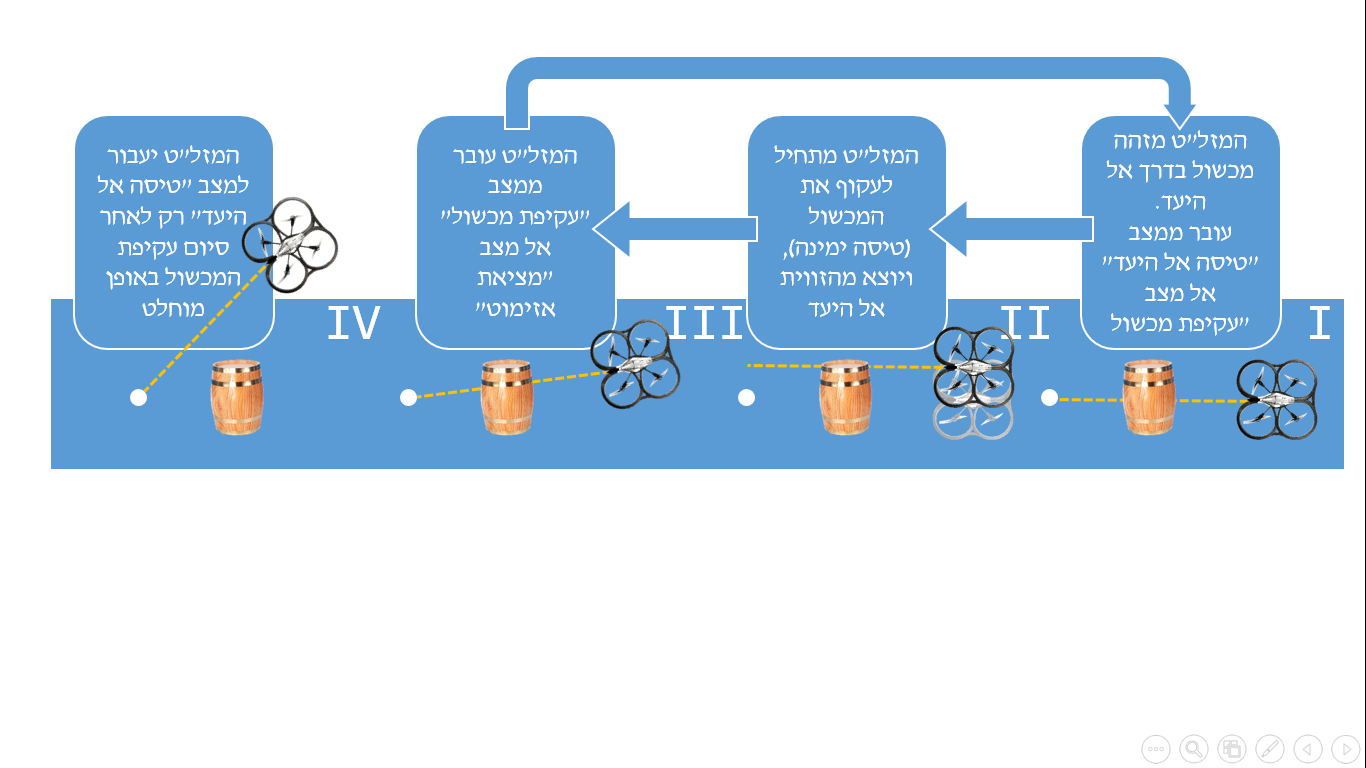
1. מצב "טיסה ידנית" Manual\_Flight

במצב זה המטוס יתעלם מהנתונים הנשלחים מבקר הטיסה והשליטה עוברת באופן מלא אל המשתמש.

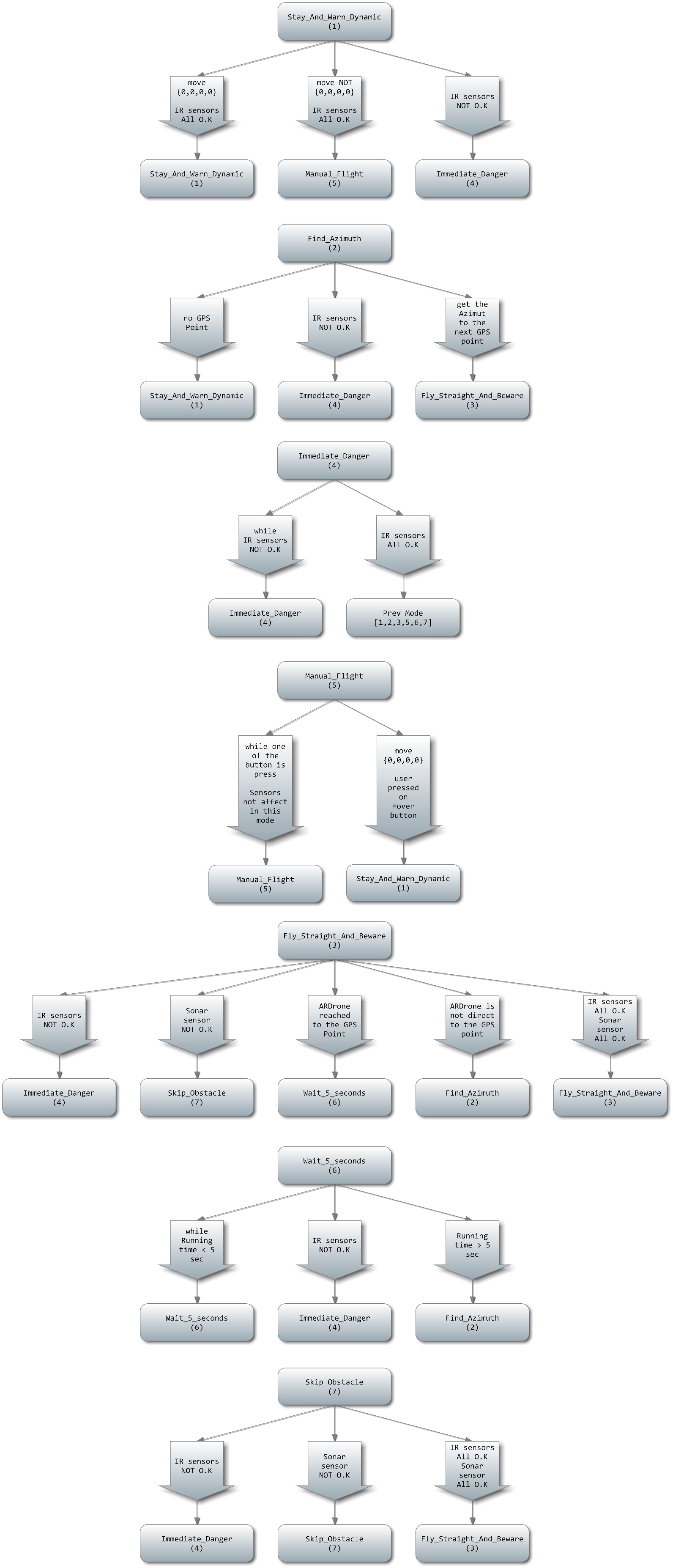
1. מצב "המתנה 5 שניות במקום" Wait\_5\_seconds

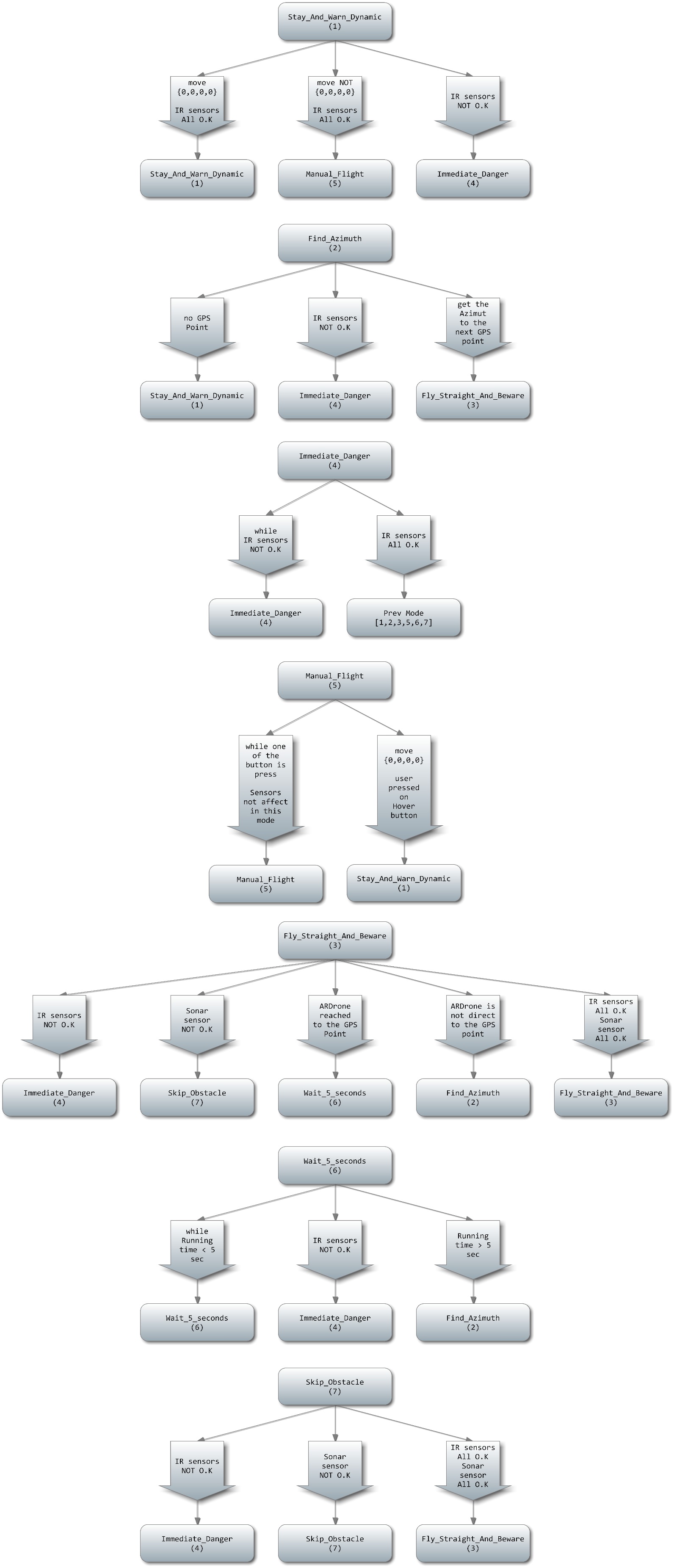
מצב זה נועד בשביל שהמזל"ט ימתין בנקודת ה:GPS אליה הוא הגיע ועד ליציאה אל הנקודה הבאה.

1. 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):

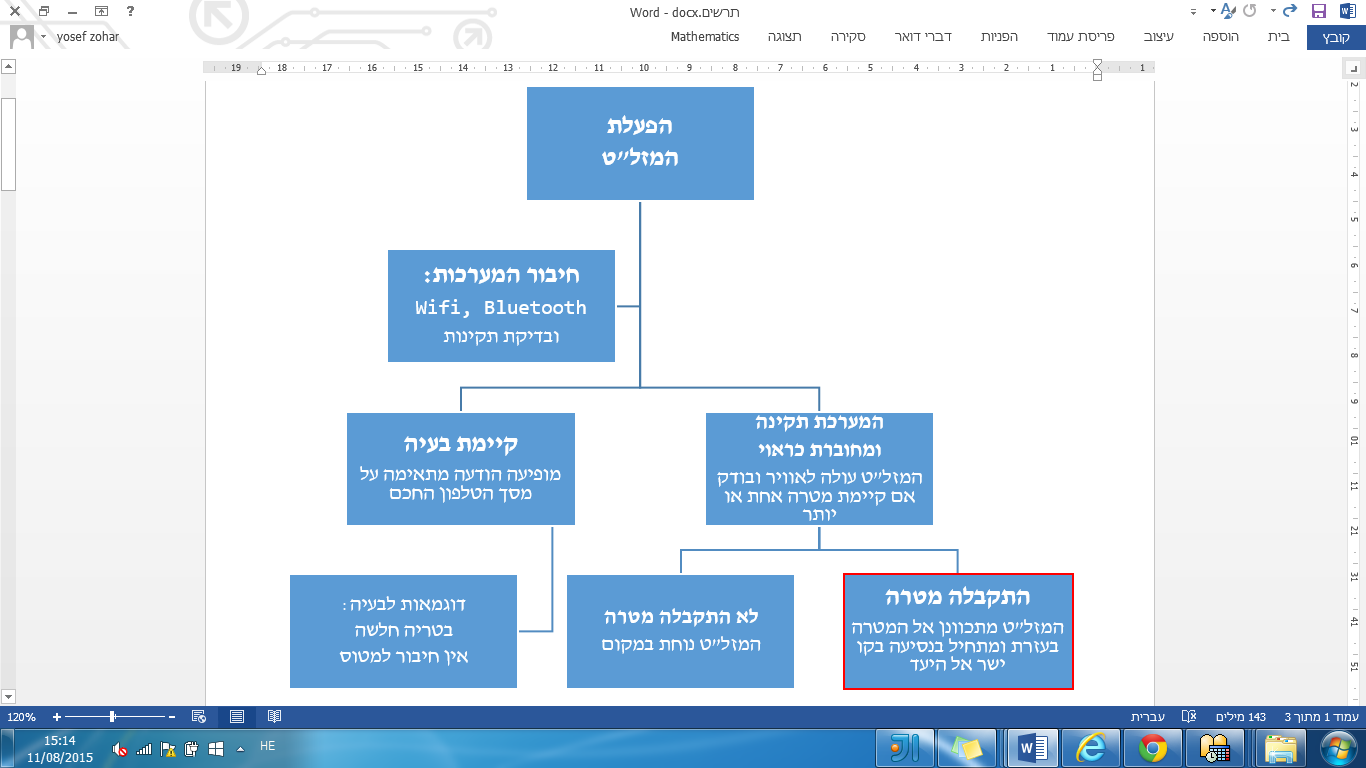


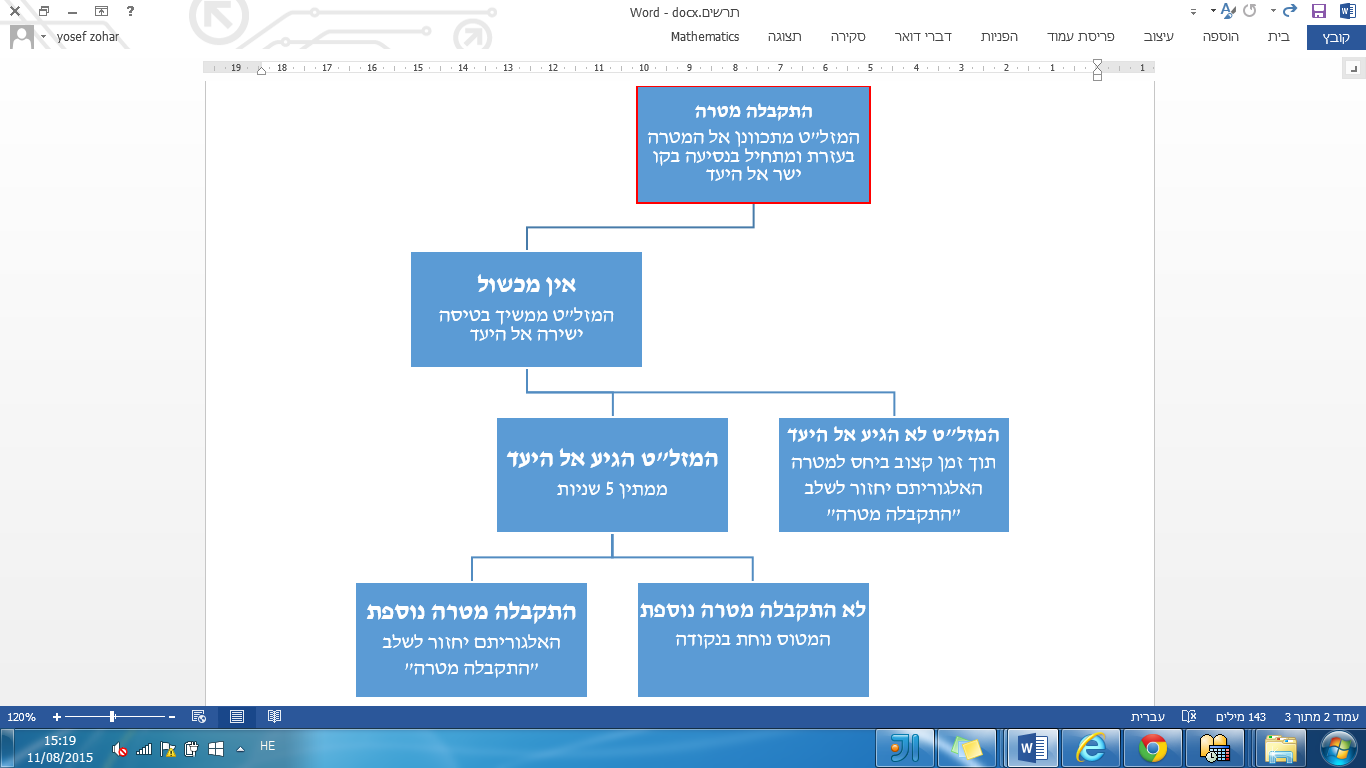


אלגוריתם כולל

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, ולכן נעבוד לפי תרשים זה. אך בהמשך התרשים ישנם מצבים המחזירים אותנו לשלב "התקבלה מטרה", ושלב זה מתפצל לשניים: ללא מכשולים (תרשים זה), ועם 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", ושלב זה מתפצל לשניים: כולל מכשולים (תרשים זה), וללא מכשולים (התרשים הקודם).

