רכב רובוט אוטונומי

קורס : מבנה מחשבים ספרתיים

מחלקה: הנדסת מחשבים שנה ב

מגישים: אסלן אסלן 302962493

עידן קלבו 201632163

סרטון 1

https://youtu.be/3SZ34QYP1EM

סרטון 2

https://youtu.be/TJbM67-VzEY

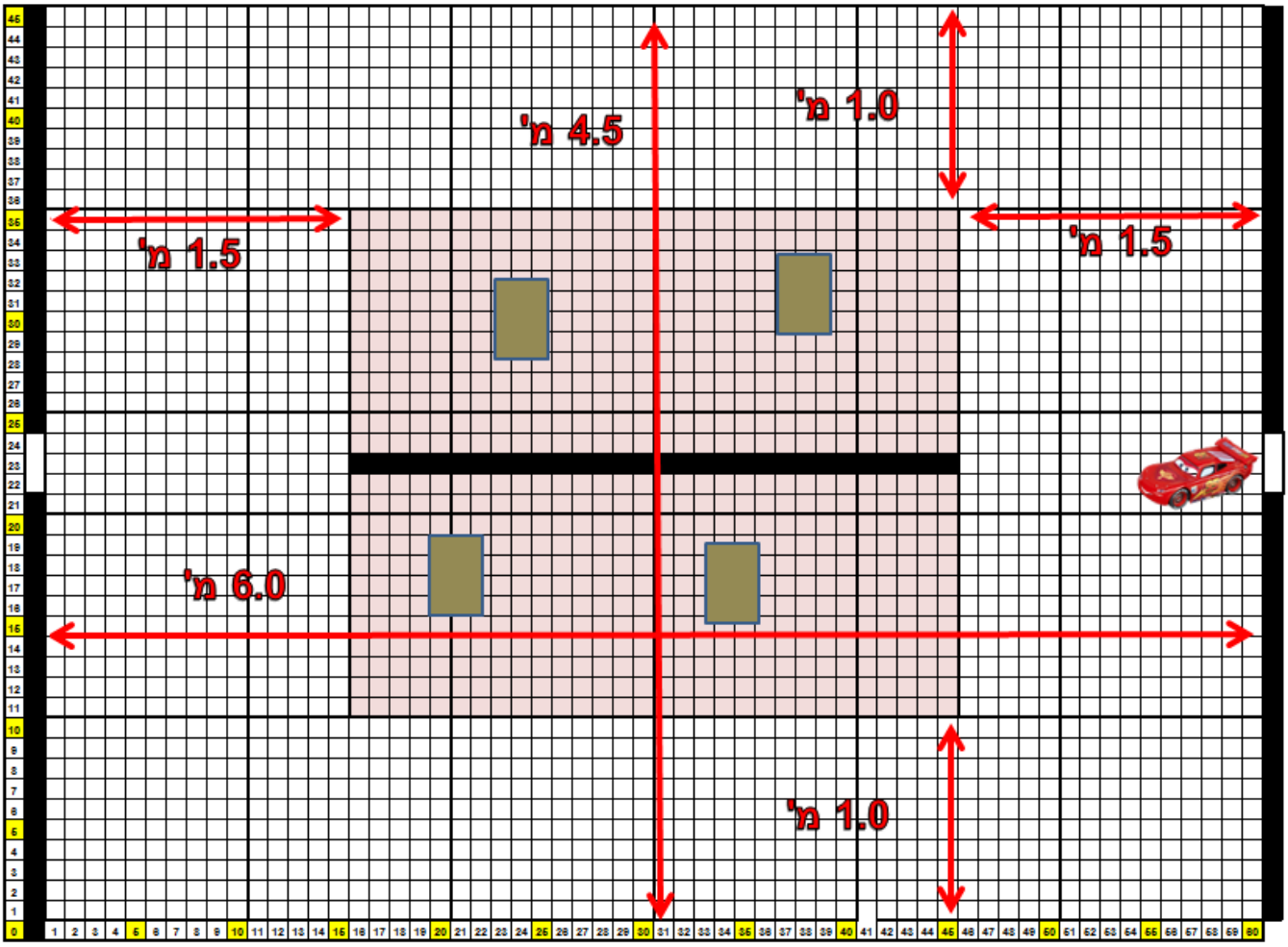
הקדמה:

במסגרת פרויקט הגמר בקורס מבנה מחשבים ספרתיים, נדרשנו לתכנן רכב אוטונומי הפועל ללא התערבות חיצונית אשר מקבל נתונים מסביבתו, מעבד את הנתונים ומסיק מסקנות להמשך פעולותיו.

מתווה התרגיל:

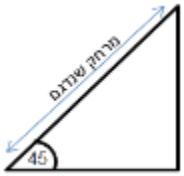
כניסה לזירה בגודל 4.5X6 מ' מציאה ופגיעה ב 4 מטרות אשר פזורות במרכז הזירה ויציאה מפתח היציאה בצדה השיני של הזירה (ראה תמונה 1)

תמונה 1



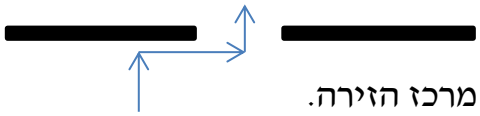
ראיון כללי של היישום:

שלב א: "סריקת מניפה" לחזית הרכב תוך חיפוש מרחק העולה על 1.5 מ' (מרחק מקסימלי לסריקה), זיהוי האם הפתח הינו מלפנים, מצד שמאל או מצד ימין. (סימון אדום בתמנה 2).



שלב ב: אם הכניסה מלפנים על הרכב לכוון חיישן מרחק בזווית 45 מעלות ולמדוד מרחק בעזרת חישוב [X sin(45) מרחק] .

אם הכניסה לא מלפנים הרכב מתקדם לקיר שמולו, וכ - 10 ס"מ מהקיר הוא מסתובב 90 מעלות לעבר הפתח, מסובב את חיישן המרחק הקרוב לקיר למציאת הפתח ומתקדם עד לזיהוי ושוב מסתובב לכיוון הפתח.



שלב ג: החזרת חיישני המרחק לחזית והתקדמות של מטר לכיוון מרכז הזירה.

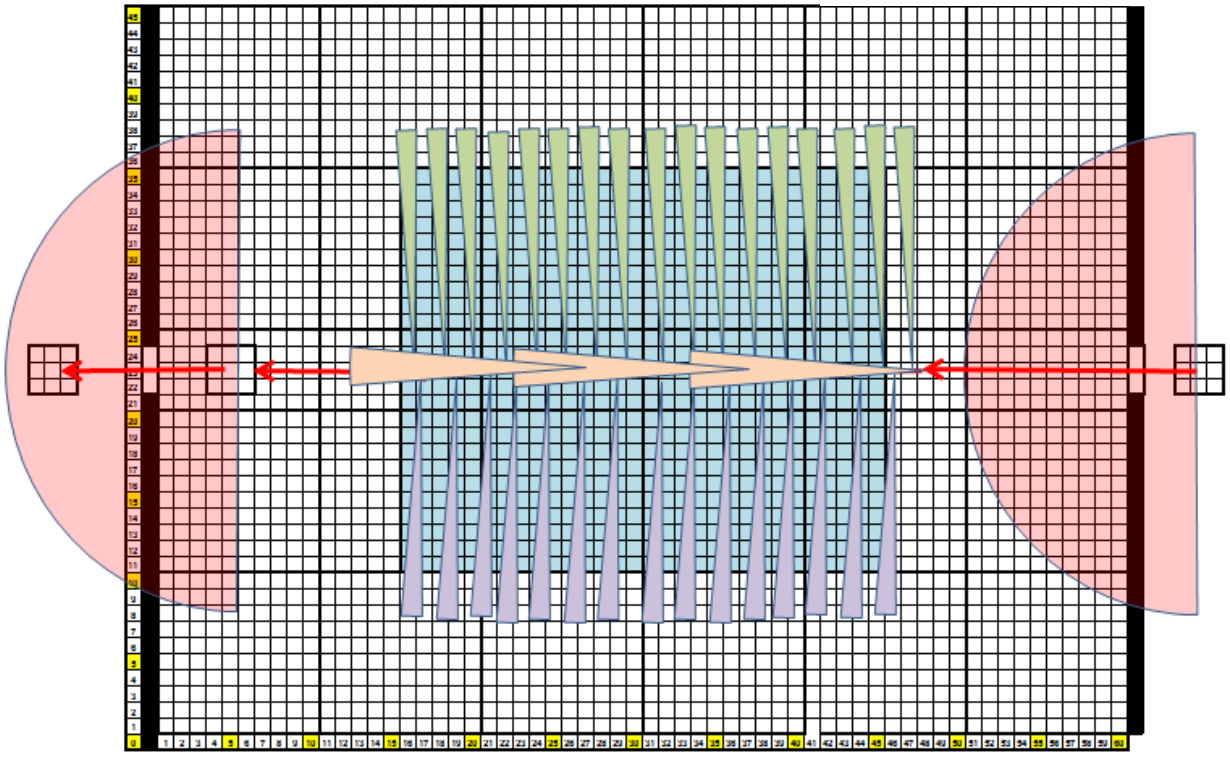
שלב ד: סיבוב חיישני המרחק ל90 מעלות, סריקה רוחבית של מרכז הזירה (3-4 מ') (סימון ירוק וסגול בתמנה 2).

שלב ביניים (מציאת מטרה) : בעט גילוי מטרה, הרכב מסתובב 90 מעלות לכיוונו ומסובב את החיישנים לחזית, מודד את המרחק המדויק שעליו לעבור על מנת לפגוע בו, נוסע הלוך פוגע במטרה וחוזר ברוורס למרכז הזירה.

שלב ה: החזרת חיישני המרחק לחזית והתקדמות של חצי מטר לכיוון היציאה מהזירה.

שלב ו: החזרת חיישני המרחק לחזית וחזרה על שלבי א',ב'.

תמונה 2



אופן שימוש ברכיבי חומרה :

מנועי DC:

על מנת לשלוט על מנועי ה DC יצרנו מספר פונקציות אשר מאפשרות לנו שליטה מלאה על אופן פעולתם ובעזרת "שתילת" פונקציות אלו במקומות השונים בקוד יצרנו את תנועת הרכב הרצויה :

```
void MotorsSpeed (int power, int side);
```

פונקציה אשר קובעת את ערכי ה Duty Cycle -של אות ה PWM - ומאפשרת שליטה על מהירותם המנועה. הפונקציה מקבלת ערך power לקביעת עוצמה ביו0 ל100.
בנוסף הפונקציה מאפשרת לשלוט על כל מנוע בנפרד או על שניהם ביחד על ידי בחירה side (0- שניהם, 1- מנוע1, 2- מנוע 2)

```
void Motorsdir (int dir);
```

פונקציה אשר קובעת את כיוון התנוע של הגלגלים , המשתנה dir מקבל את הערכים 12,3,6,9 (כיוונים לפי שעון מחוגים) וקובע את הגלגלים לנוע קדימה או אחורה בהתאמה למצב.

```
void MotorsBalance ();
```

פונקציה אשר עוזרת לנו לייצב את הרכב במהלך נסיע, בכל תנוע של הרכב ישנם קריאות חוזרות לפונקציה זו על מנת לשמור על תנועה שווה בין שני המנועים.
הפונקציה בודקת בעזרת שני משתני עזר כמה מסתובב כל מנוע ומגבירה או מנמיכה את המהירות של מנוע 1 בהתאם למהירות של מנוע 2.

```
void Turnright();
```

פונקציה אשר מסובבת את הרכב 90 מעלות ימינה. (משתמשת במשתנה עזר על מנת למדוד מרחק של תזוזה בכל גלגל)

```
void Turnleft();
```

פונקציה אשר מסובבת את הרכב 90 מעלות שמאלה. (משתמשת במשתנה עזר על מנת למדוד מרחק של תזוזה בכל גלגל)

```
void distance2Driving (int distance, int speed);
```

פונקציה אשר מאפשרת לנו לבחור מהירות מסוימת (speed) ומרחק מסוים (distance) לתזוזת הרכב, הפונקציה משתמשת במשתנה עזר למדידת מרחק וגם בפונקציות שהוסברו לעיל.

```
void ChangeSpeed(int Speed,int dir);
```

לאחר מספר ניסיונות כושלים לייצוב הרכב הבנו ששינוי מהירות הרכב מצריך פעולות סדורות עם שליטה שינוי וכיבוי אינטרפטים/שעונים , ולכן יצרנו פונקציה ייעודית לשינוי זה.
Speed – המהירות הנדרשת , dir – כיוון הנסיעה (ספרות בשעון מחוגים)

```
void RoundDahawin();
```

פונקציה אשר מסובבת את הרכב סיבוב 360 במקום .

```
void distance2REVERS (int distance, int speed);
```

פונקציה אשר משלימה לפונקציה distance2Driving אך בפונקציה זו הרכב ינוע רוורס.

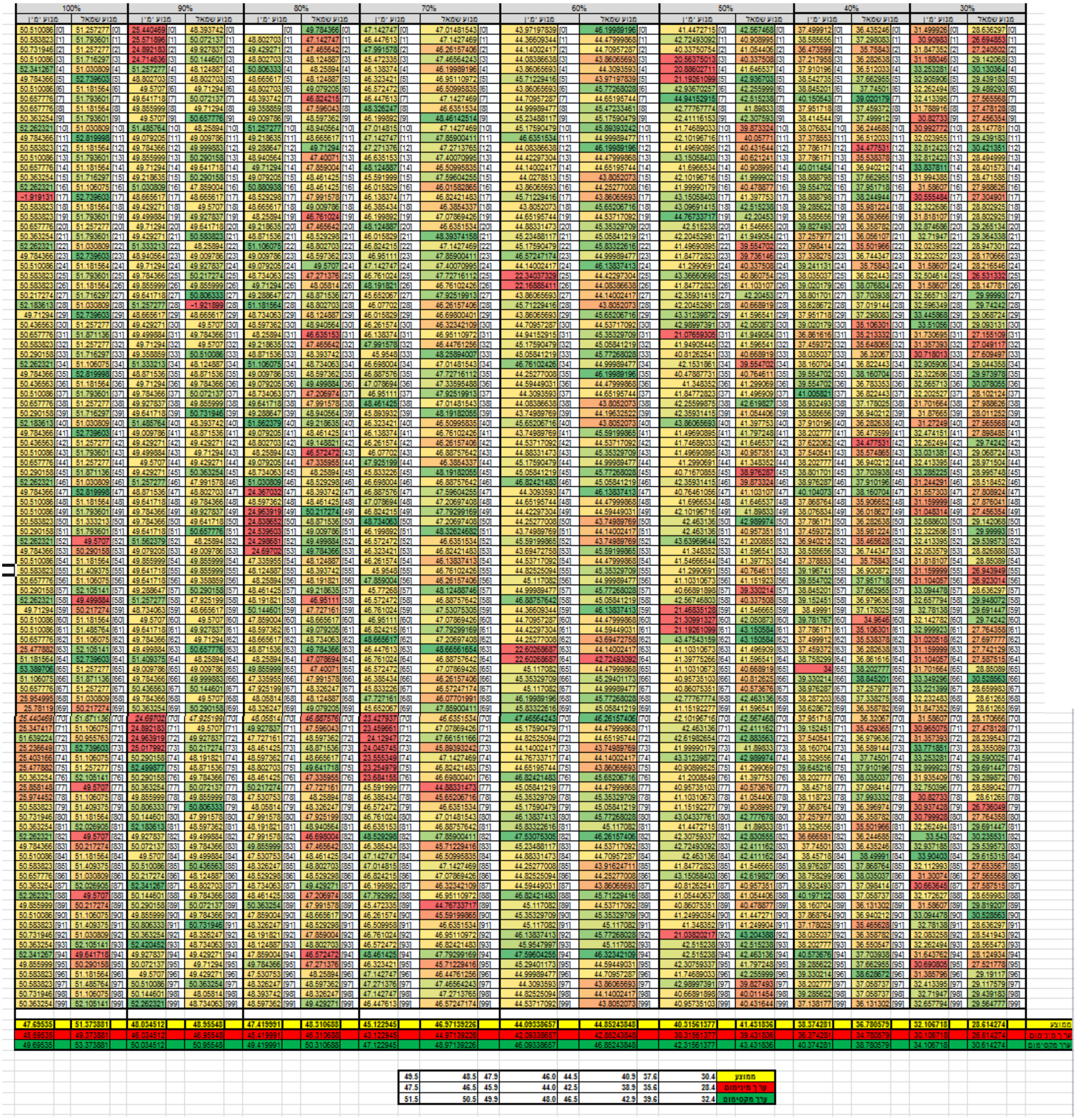
<u>ENCODER ומשוב לייצוב הרכב:</u>

הקדמה: את העבודה עםENCODER והמשוב לייצוב הרכב התחלנו עוד בשלב המקדים בו כתבנו תוכנת בדיקה. בשלב זה גילינו מספר בלתי מבוטל של שגיאות בשיטת ה Input Capture מהשעון TPM, את הבדיקה לשיטה זו עשינו בעזרת הכנסת ערכי המהירות של כל גלגל למערך (במהירויות שונות) ומצאנו שבמקרים רבים השיטה אינה מדויקת. **<u>בתמונה 3</u>** הוספנו את הדגימות והממוצעים שהתקבלו ב100 דגימות לכל גלגל (את הדגימות התחלנו לאחר מספר שניות בהם המנוע התייצב על העוצמה המבוקשת).

ניתן לראות כי ערכים רבים אינם עומדים בסטיית תקן סבירה ולכן עברנו לעבודה בשיטה של ספירת "קליקים" לפי החזרים מסיבוב הגלגל וחישוב מרחק מול מהירות.

<u>בתמונה 4</u> ניתן לראות את הקוד שכתבנו בתחילה בשיטת Input Capture.

תמונה 3



	30%		40%		50%		60%		70%		80%		90%		100%	
	מנוע שמאל	מנוע ימין	מנוע שמאל	מנוע ימין	מנוע שמאל	מנוע ימין	מנוע שמאל	מנוע ימין	מנוע שמאל	מנוע ימין	מנוע שמאל	מנוע ימין	מנוע שמאל	מנוע ימין	מנוע שמאל	מנוע ימין
ממוצע	28.614274	32.106718	36.780579	38.374281	41.431836	40.31561377	44.85243848	44.09338657	46.97139226	45.122945	48.310688	47.419991	48.95548	48.034512	51.373881	47.69535
ערך מינימום	26.614274	30.106718	34.780579	36.374281	39.431836	38.31561377	42.85243848	42.09338657	44.97139226	43.122945	46.310688	45.419991	46.95548	46.034512	49.373881	45.69535
ערך מקסימום	30.614274	34.106718	38.780579	40.374281	43.431836	42.31561377	46.85243848	46.09338657	48.97139226	47.122945	50.310688	49.419991	50.95548	50.034512	53.373881	49.69535

ממוצע	30.4	37.6	40.9	44.5	46.0	47.9	48.5	49.5
ערך מינימום	28.4	35.6	38.9	42.5	44.0	45.9	46.5	47.5
ערך מקסימום	32.4	39.6	42.9	46.5	48.0	49.9	50.5	51.5

תמונה 4

```
void FTM0_IRQHandler(){
    if(TPM0_STATUS & TPM_STATUS_TOF_MASK ){                                  //TPM0 Overflow
        TPM0_CountOverflow++;
    }

    if(TPM0_STATUS & TPM_STATUS_CH2F_MASK && (TPM0_C2SC & TPM_CnSC_CHIE_MASK)){                    //Clock Rise Right Engine
        StRightEng[IndexEnc1] =  TPM0_C2V + (TPM0_CountOverflow * MUDULO_REGISTER);
        IndexEnc1++;
        if (IndexEnc1==2){
            StRightEng [2] = velocity_CPS (StRightEng [0],StRightEng [1]);
            IR_Dis_TPM0_2;          //Interrupt disable TPM0_CH3

        }
    }

    if(TPM0_STATUS & TPM_STATUS_CH3F_MASK && (TPM0_C3SC & TPM_CnSC_CHIE_MASK)){                    //Clock Rise Left Engine
        StLeftEng [IndexEnc2] = TPM0_C3V + (TPM0_CountOverflow * MUDULO_REGISTER);
        IndexEnc2++;
        if (IndexEnc2==2){
            StLeftEng [2] = velocity_CPS (StLeftEng [0],StLeftEng [1]);
            IR_Dis_TPM0_3;  //Interrupt disable TPM0_CH2
        }
    }

}
```

```
    // calculat velocity (cm per second)
double velocity_CPS (double x1 ,double x2){
    double time = ((x2 - x1)/(40*18750));
    double distance = (6*3.14159265358979)/408;
    return  distance/time;
}


    void MotorsBalance (){
    if (averageRightEng > averageLeftEng){
        MotorsSpeed (EnginePower[1]+1,2);
    }
    else{
        if (averageLeftEng < averageRightEng){
            MotorsSpeed (EnginePower[1]-1,2);
        }
    }
}
```

ENCODER ומשוב לייצוב הרכב (המשך):

העבודה בשיטה של ספירת "קליקים" מצריכה קריאה חוזרת בפרקי זמן קבועים של משתני עזר השוואה ביניהם ושינוי המהירות המנועים בהתאם
(כל זאת עושה הפונקציה **MotorsBalance** שהוסברה בחלק הקודם).
המשתנים Enc1 ו-Enc2 סופרים את מספר "קליקים" בכל מנוע בנפרד ואילו המשתנה cuntENC עוזר לנו למדוד מרחק כללי שעבר הרכב. (בתמונה 5 ניתן לראות את החישוב של המרחק אותו עובר הרכב)

```
void FTM0_IRQHandler(){
        //Clock Rise Right Engine
        if(TPM0_STATUS & TPM_STATUS_CH2F_MASK &&(TPM0_C2SC & TPM_CnSC_CHIE_MASK)){
                Enc1++;
         }
        //Clock Rise Left Engine
        if(TPM0_STATUS & TPM_STATUS_CH3F_MASK &&(TPM0_C3SC & TPM_CnSC_CHIE_MASK)){
                Enc2++;
         }
         cuntENC++;
         Reset_IR_flags_TPM0;
}
```

תמונה 5

		מרחק במילימטר	כמות "קליקים"
		0.46199892	
		97.01977313	210
		97.48177205	211
		97.94377096	212
60	קוטר גלגל	98.40576988	213
		98.8677688	214
3.141593	"פאי"	99.32976772	215
		99.79176664	216
188.4956	מרחק שעובר הרכב כל 408 "קליקים"	100.2537656	217
		100.7157645	218
		101.1777634	219
		101.6397623	220
		102.1017612	221
		102.5637602	222
		103.0257591	223
		103.487758	224
		103.9497569	225
		104.4117558	226
		104.8737548	227
		105.3357537	228
		105.7977526	229
		106.2597515	230
		106.7217504	231
		107.1837494	232
		107.6457483	233
		108.1077472	234
		108.5697461	235
		109.031745	236
		109.493744	237
		109.9557429	238
		110.4177418	239

מנועי servo:

מנועי הservo נשלטים בעזרת ערכי Duty Cycle - של אות ה- PWM.
גם בחלק זה כמו במנועי הDC יצרנו פונקציות אשר עזרו לנו לשלוט במנועים ואותם "שתלנו" במקומות הנחוצים בקוד. (את החישובים לזווית ניתן לראות בתמונה 6)

```
void Servo1SetPos (int Servo1Position);
void Servo2SetPos (int Servo2Position);
```

שתי פונקציות אלו מקבלות זוית רצויה ומזיזות את מנוע הסרבו לזווית זו (כל פונקציה שולטת במנוע שונה)

```
void ServoFront(int side);
```

פונקציה זו מביא את מנוע הסרבו לחזית הרכב, ניתן לשלוט על כל מנוע בנפרד או על שניהם ביחד על ידי בחירת side (0- שניהם, 1- מנוע1, 2- מנוע2)

```
void Servoscan(int side);
```

פונקציה זו מזיזה מנוע סרבו נבחר side (0- שניהם, 1- מנוע1, 2- מנוע2) מזיזה אותו 10 מעלות כלפי חוץ, עד הגעה ל100 מעלות וחוזר לחזית.

תמונה 6

מעלות	T-pwm	T-on	ערך רגיסטר	הקסדצימלי
0	0.568	0.02272	426	1A9
5	0.613333	0.024533	460	1CB
10	0.658667	0.026347	494	1ED
15	0.704	0.02816	528	210
20	0.749333	0.029973	562	232
25	0.794667	0.031787	596	254
30	0.84	0.0336	630	276
35	0.885333	0.035413	664	298
40	0.930667	0.037227	698	2BA
45	0.976	0.03904	732	2DC
50	1.021333	0.040853	766	2FD
55	1.066667	0.042667	800	31F
60	1.112	0.04448	834	341
65	1.157333	0.046293	868	363
70	1.202667	0.048107	902	385
75	1.248	0.04992	936	3A7
80	1.293333	0.051733	970	3C9
85	1.338667	0.053547	1004	3EB
90	1.384	0.05536	1038	40D
95	1.429333	0.057173	1072	42F
100	1.474667	0.058987	1106	451
105	1.52	0.0608	1140	473
110	1.565333	0.062613	1174	495
115	1.610667	0.064427	1208	4B7
120	1.656	0.06624	1242	4D9
125	1.701333	0.068053	1276	4FB
130	1.746667	0.069867	1310	51D
135	1.792	0.07168	1344	53F
140	1.837333	0.073493	1378	561
145	1.882667	0.075307	1412	583
150	1.928	0.07712	1446	5A5
155	1.973333	0.078933	1480	5C7
160	2.018667	0.080747	1514	5E9
165	2.064	0.08256	1548	60B
170	2.109333	0.084373	1582	62D
175	2.154667	0.086187	1616	64F
180	2.2	0.088	1650	671

0.568	MinServoAngle
2.2	MaxServoAngle
0.045333333	OneServoAngle
18750	

חיישני מרחק IR:

בעזרת חיישני מרחק IR ממדנו את המרחקים מהמטרות מהכניסה ומהיציאה.
שני חיישני המרחק מחוברים ל ADC0_B ולכן היה צורך בכל פעם לשנות את ערוץ הכניסה לADC.
את הערך הדגום היה עלינו להמיר למרחק על ידי יצירת מערך שישמש כ"כרטיס טווחים" ולכן היה צורך לכייל תחילה את החיישנים ולהכניס את הערכים ל"כרטיס טווחים".
לשם כך יצרנו פונקציה ייעודית. `void set_CardRanges();`.

את הערכים בדקנו הם מוצגים בטבלה (תמונה 7).

בנוסף היה עלינו ליצור מספר פונקציות עזר למדידת מרחק.

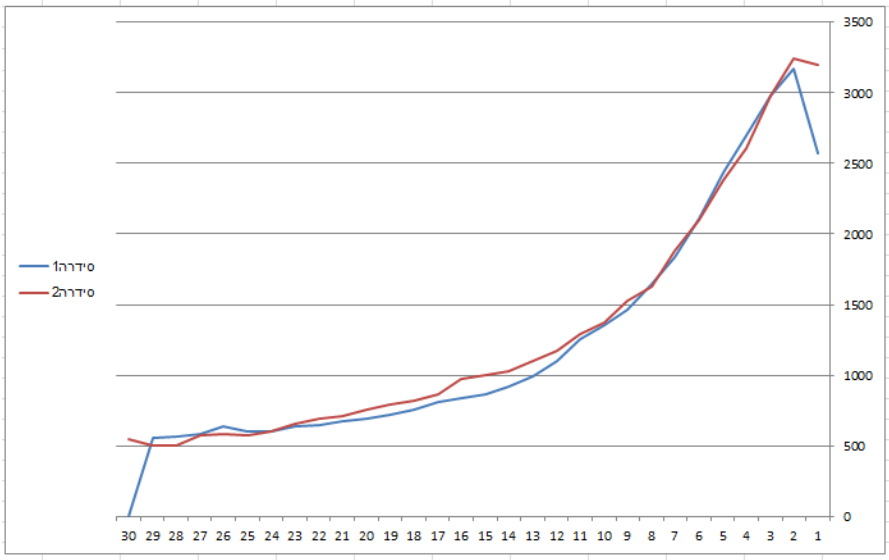
```
void distance0 (int dir);
```

פונקציה זו נעזרת באחת משתי הפונקציות הבאות ומחזירה מרחק מחיישן שצוין במשתנה dir

```
void distance1 ();
void distance2 ();
```

שתי פונקציות אלה מחפשות במערך את השני הערכים הקרובים ביותר לערך הדגום, ובעזרת ערכים אלו מחשבת את המרחק בס"מ.

תמונה 7



S2			S1		
דגימה	ס"מ	אינדקס	דגימה	ס"מ	אינדקס
3195	10	[0]	2571	10	[0]
3246	15	[1]	3170	15	[1]
2983	20	[2]	2977	20	[2]
2611	25	[3]	2700	25	[3]
2383	30	[4]	2432	30	[4]
2103	35	[5]	2105	35	[5]
1885	40	[6]	1838	40	[6]
1626	45	[7]	1645	45	[7]
1530	50	[8]	1461	50	[8]
1375	55	[9]	1351	55	[9]
1290	60	[10]	1251	60	[10]
1175	65	[11]	1105	65	[11]
1102	70	[12]	989	70	[12]
1028	75	[13]	923	75	[13]
997	80	[14]	868	80	[14]
972	85	[15]	837	85	[15]
864	90	[16]	811	90	[16]
822	95	[17]	753	95	[17]
789	100	[18]	724	100	[18]
758	105	[19]	689	105	[19]
709	110	[20]	676	110	[20]
688	115	[21]	649	115	[21]
659	120	[22]	637	120	[22]
604	125	[23]	601	125	[23]
577	130	[24]	602	130	[24]
588	135	[25]	634	135	[25]
570	140	[26]	582	140	[26]
503	145	[27]	569	145	[27]
501	150	[28]	552	150	[28]
549	155	[29]	0	155	[29]

מיפוי:

לצורך חישוב מרחק מהיציאה, הגעה למטרות, חזרה מהם, תיעוד מסלול הרכב, ומיקום המטרות היה צורך במיפוי הזירה. לשם כך הגדרנו מערך דו מימדי בגודל 60X45 כך שכל תא מייצג 10 סמ"ר.

- המפה מאותחלת ב - 0.
- מיקום הרכב מסומן ב -2
- מסלול הרכב מסומן ב - 1
- מטרות מסומנות ב- 3

בתמונה 2 ניתן ליראות סקיצה של המערך הדו מימדי.

בנוסף הגדרנו פונקציה ייעודית לעדכון המפה

```
void updateMap (int distance, int dir);
```

אשר מקבלת מרחק distance וכיוון dir ומסמנת על גבי המפה את ההתקדמות הרכב.

בנוסף על מנת לפסול אפשרות של זיהוי כפול סימנו מקומות בהם נעשה כבר זיהוי ולפני יציאה לפגיעה במטרה הרכב בודק האם הוא היה כבר במיקום זה.

פונקציות ומשתנים נוספים:

1. בזמן התכנון שמנו לב לכך שיש לנו צורך להשהיות מרובות בזמנים משתנים במהלך ריצת הקוד. לכן יצרנו את הפונקציה `void InitPIT1(int x)` אשר מאפשרת להכניס מספר ששוה ערך ל1/100 שניה ובעזרת המשתנה `flagpit1` קבענו האם עבר הזמן הנדרש או לא.

2. משתנים נוספים:

`int Mainflag;`	משתנה לטבלת מצבים
`int flag;`	משתנה לטבלת מצבים
`int EnginePower [2];`	שומר את עוצמת המנועים
`int ServoAngle [2];`	שומר את זווית המנועים
`int xLocation,yLocation;`	מיקום נוכחי של הרכב

<u>טבלת מצבים:</u> (מצורף תרשים זרימה)
את טבלת המצבים חילקנו לשתיים :

מצב ראשי - לפי מיקום הרכב

מצבים משניים – לכל אחד מהמצבים הראשיים קיים מספר מצבים משניים בו הוא יכול להימצא.

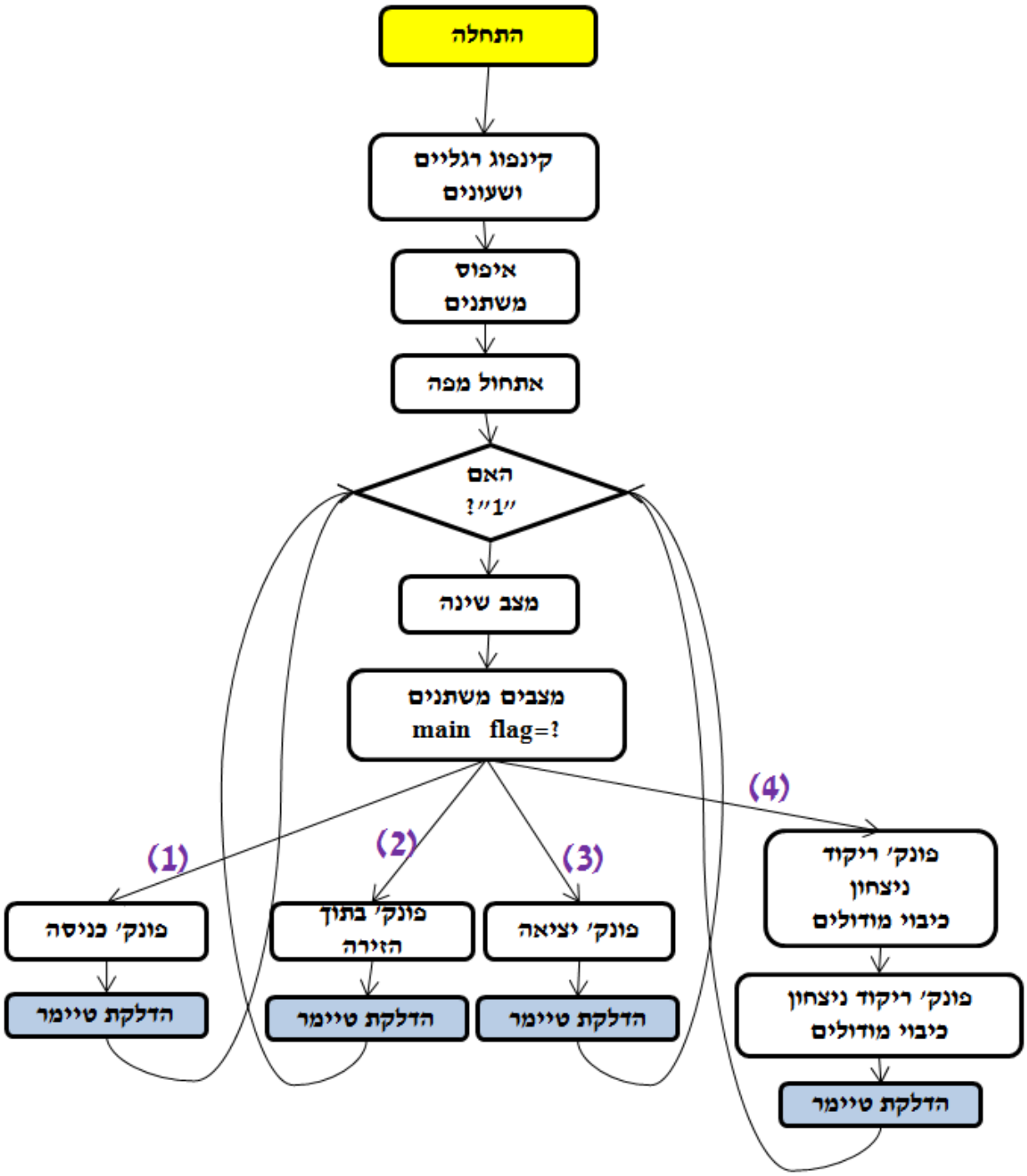
(לכל פעולה של הרכב התאמנו הדלקת נורות LED בשביל לעקוב אחרי אופן הפעולות ומצבו הנוכחי.)

א. כניסה
1. הכנות של הרכב מבחינת קינפוגים, ומיקום החיישנים.
2. סריקות של הסרבו למציאת פתח
3. התקדמות לקיר
4. פניה לכיוון הפתח וסיבוב חיישן מתאים לכיוון הפתח.
5. התקדמות עד הגעה לפתח.
6. סיבוב הרכב לכיוון הפתח
7. כניסה לפתח.

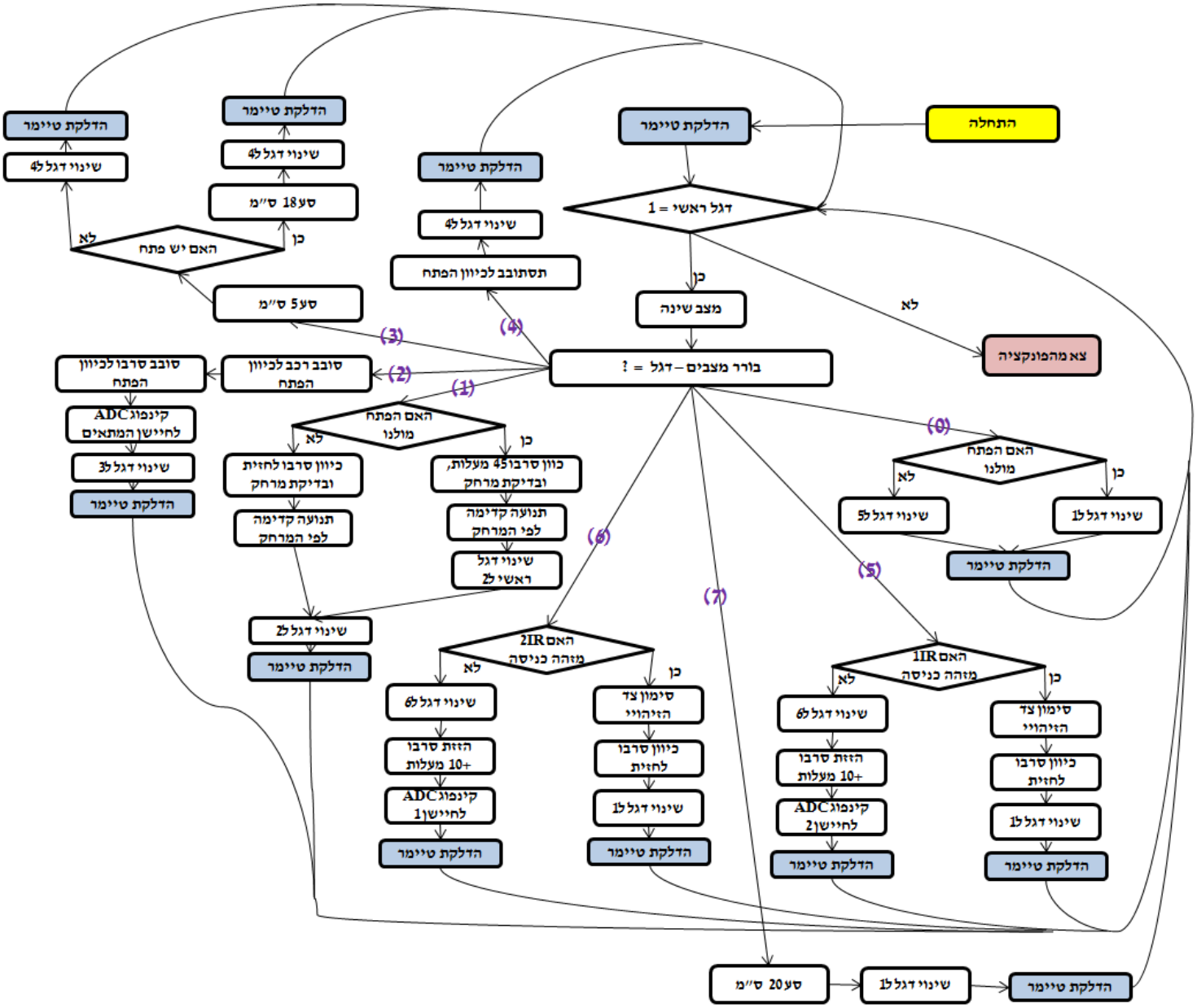
ב. בתוך הזירה
1. הכנות של הרכב מבחינת קינפוגים, ומיקום החיישנים.
2. תנועה לכיוון מרכז הזירה (+סימון במפה)
3. הזזרת החישנים לצידי הרכב ותחילת תנועה. (+סימון במפה)
4. חיפוש מטרות מימין ומשמאל לסירוגין (חזרה על פעולה זו עד למציאת מטרה או עד ליציאה מהאיזור המוגדר כמרכז הזירה. (אם נמצא מטרה סיבוב הרכב 90 מעלות לכיוונו)
5. **(במקרה של מציאת מטרה)** הזזת הסרבו לחזית
6. **(במקרה של מציאת מטרה)** בדיקת מרחק מהמטרה
7. **(במקרה של מציאת מטרה)** תנועה לכיוון המטרה(+סימון במפה)
8. **(במקרה של מציאת מטרה)** חזרה למרכז הזירה ברוורס (+סימון במפה)
9. סיבוב הרכב 90 מעלות לכיוון היציאה (+סימון במפה)

ג. יציאה
1. הכנות של הרכב מבחינת קינפוגים, ומיקום החיישנים.
2. נסיעה ישר עד לזיהוי הקיר היציאה.
3. מעבר למצב "כניסה לזירה"
4. עצירה מוחלטת של כל השעונים.

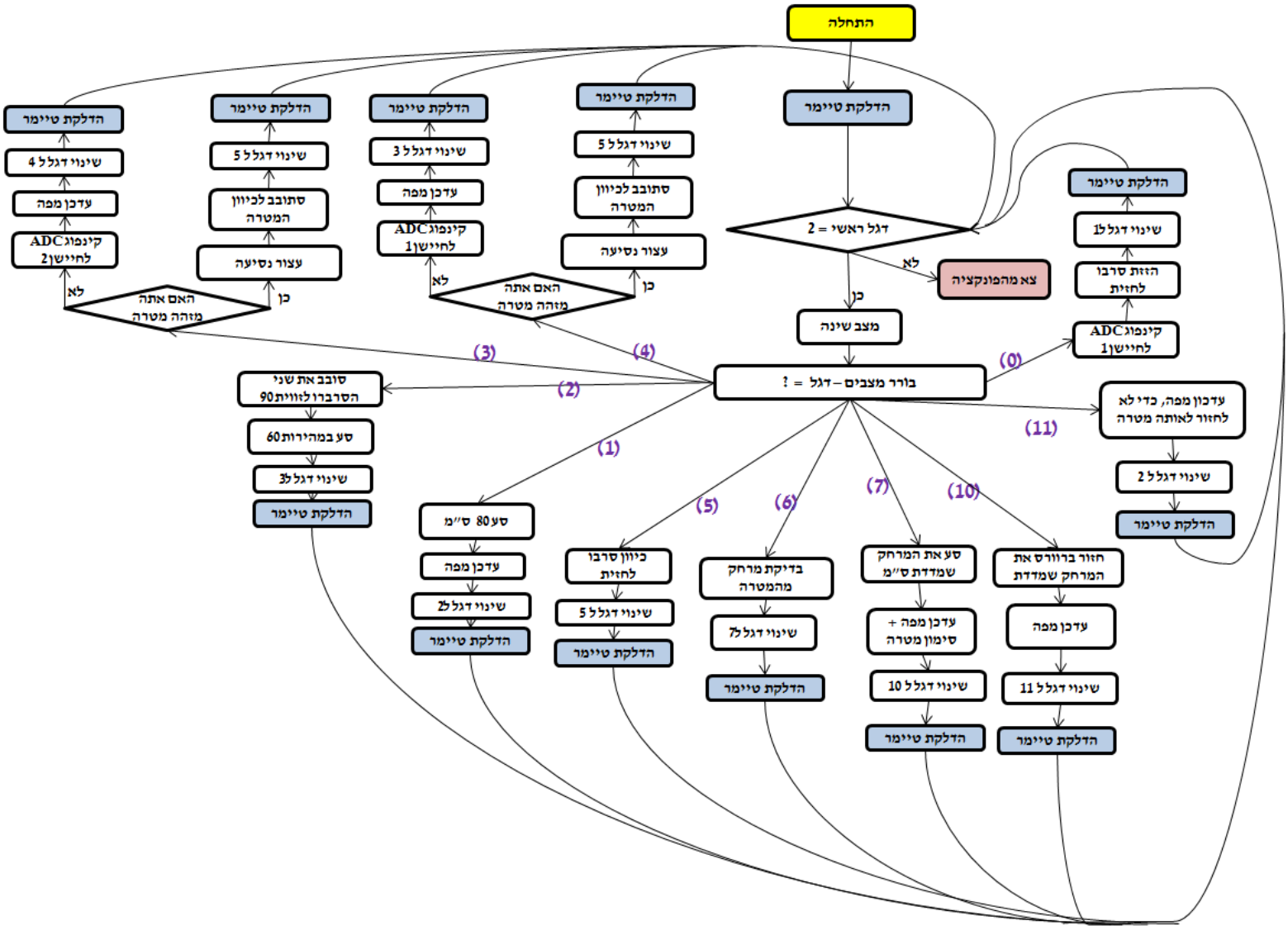
טבלת מצבים ראשית



פונקציית כניסה לזירה



פונקציית בתוך הזירה



פונקציית יציאה מהזירה