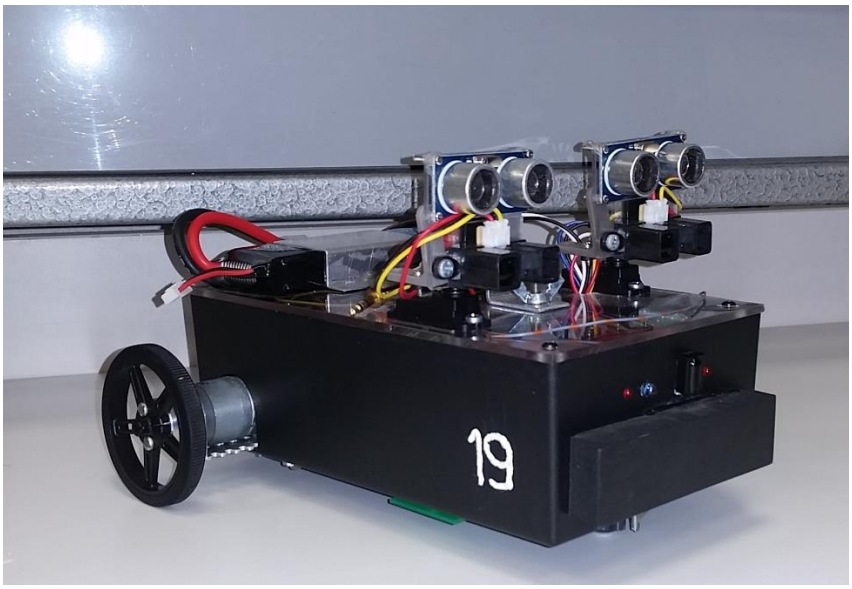
פרוייקט גמר
DCS

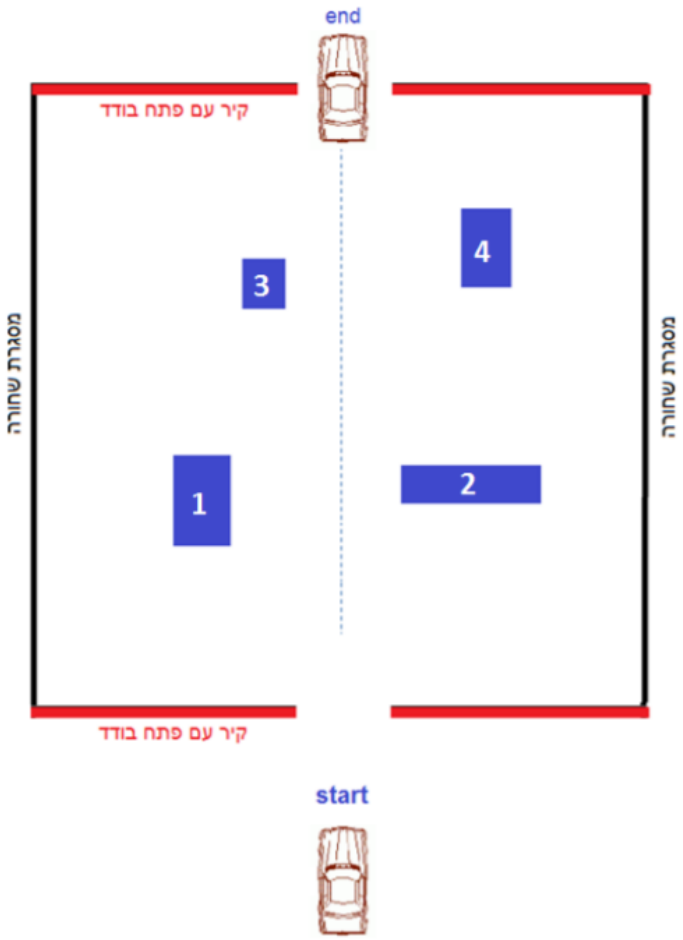


תיאור הפרוייקט

המשימה שהוגדרה היא לעבור מסלול מכשולים ובו 4 מכשולים. על הרכב להיכנס לזירה, לזהות את כל 4 המכשולים, לגעת בהם ולצאת מהזירה בזמן מינימלי ובאופן אוטונומי ללא התערבות חיצונית של המפעיל.

השלבים במשימה:

1. נקודת ההתחלה נמצאת בקו אווירי אל מול הכניסה לזירה, שם ימוקם הרכב תחילה.
2. בשלב הראשון נדרשנו לעבור דרך הפתח אשר רוחבו כפול מרוחב הרכב – הכניסה לזירה.
3. בשלב השני היינו צריכים לסוע כמטר וחצי ולהגיע לאזור המטרות.
4. באזור המטרות נדרשנו לזהות אותן ע"י סריקה בעזרת חיישני המרחק השונים, להסתובב אליהן ולגעת בהן, תוך ווידוא שכל מטרה זוהתה רק פעם.
5. לאחר אזור המכשולים עלינו לצאת מזירה מאחת משתי היציאות הממוקמות (בשונה ממה שמסומן באיור) לא בקו אווירי אל מול דלת הכניסה ומסלול הרכב אלא אחת מימין לקו זה ואחת משמאלו.



הקדמה

המשימה חולקה למספר שלבים, בכל מצב הוגדר מצב נסיעה מסוים לרכב ונכתב עבור מצב זה אלגוריתם ופוקציות ייעודיות מתאימות. בכל שלב, הרכב יודע לפי מצב הנסיעה שלו היכן הוא ממוקם ומה הפעולה הבאה.

השלבים הם:

שלב א' – על הרכב לעבור דרך הכניסה - פתח בקיר אשר רוחבו כפול מרוחב הרכב.

שלב ב' – נסיעה בקו ישר לאורך הזירה עד הגעה לאזור המכשולים.

שלב ג' – סריקת אזור המכשולים תוך זיהוי המטרות.

שלב ד' – לאחר זיהוי מטרה, על המכונית לסוע לכיוונה ולגעת בה.

שלב ה' – לאחר נגיעה במטרה חזרה לקו הישר האווירי לכניסה לזירה.

שלב ו' – חזרה על שלבים ד' וה' עבור כל אחת מארבעת המטרות.

שלב ז' – לאחר זיהוי כל המטרות נסיעה אל הקיר הנמצא בסוף הזירה.

שלב ח' – יציאה מהזירה דרך אחת משתי הדלתות הקיימות.

שלב א'

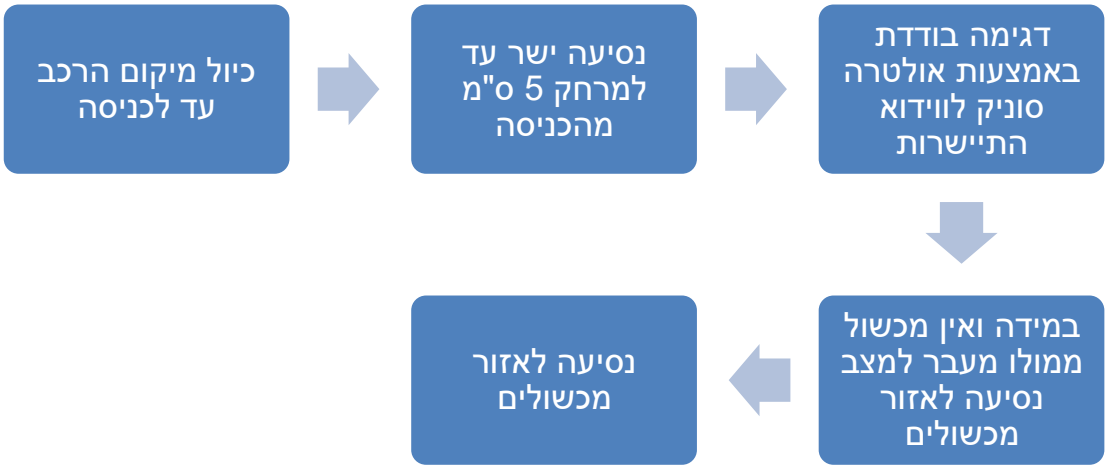
הרכב מונח תחילה במרחק X מפתח הכניסה (המרחק X הוגדר לנו לפני המשימה).

נכייל את מיקום הרכב - המרחק X לפי מה שיוגדר לנו. לאחר כן אוטונומית הרכב יסע לפי מרחק זה עד הגעתו למרחק 5 ס"מ מפתח הכניסה.

בהגעתו של הרכב אל פתח הכניסה ילקחו מספר דגימות ע"י חיישני האולטרה סוניק בזוויות שונות על מנת לוודא את התיישרותו אל מול פתח הכניסה.

במידה וזיהה כי לא הגיע ישר אל מול פתח הכניסה יבצע תיקונים על מנת להתיישר ולהיכנס בצורה מתאימה לזירה.

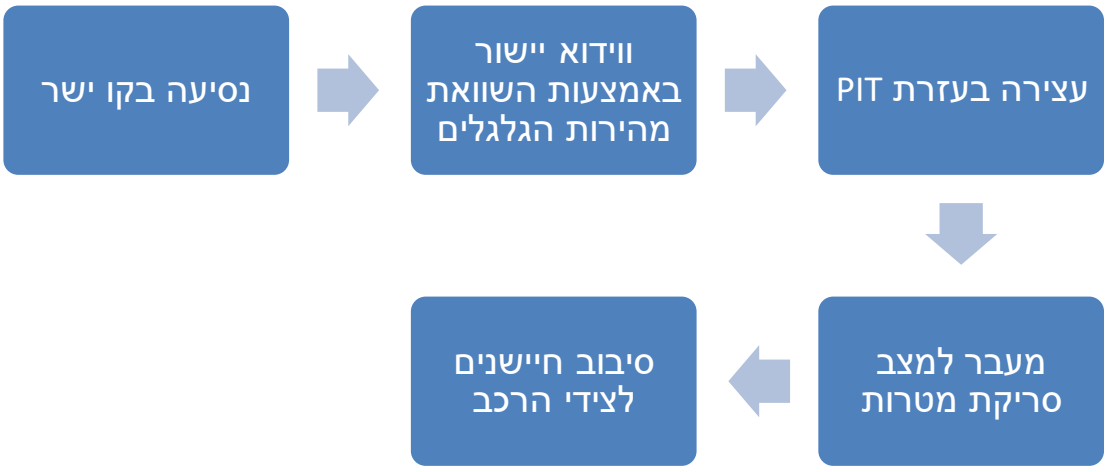
לאחר מכן ימשיך וייסע הרכב ישר ויעבור למצב נסיעה ב' – נסיעה 150 סנטימטרים אל אזור המטרות.



שלב ב'

בשלב זה מוגדר לרכב לסוע מרחק של 150 סנטימטרים עד הגיעו לאזור המכשולים.
המרחק (150 סנטימטרים) הוגדר לנו כחלק מהמשימה.
את הנסיעה ישר נעשה ע"י השוואת מהירות הגלגלים ותיקונים מתאימים בהתאם.
לאורך כל השלב הנוכחי יעשה חישוב ע"י זמן מהPIT והמהירות הנדגמת מהגלגלים כדי לחשב את המרחק והמיקום של הרכב בכל זמן נתון עד הגעו לסוף 150 הסנטימטרים.

לאחר הגעתו של הרכב וסיום מעבר 150 סנטימטרים יעבור הרכב למצב ג', אך לפני המעבר למצב הבא יסובבאת החיישנים לצידו ל הרכב(90 מעלות) ע"י מנועי הסרבו לטובת סריקת המטרות בשלב הבא.



שלבים ג'-ו'

באזור המכשולים הוגדר לנו שנמצאות 4 מטרות, לכן על הרכב לעבור את אזור המטרות תוך כדי סריקה ע"י חיישני המרחק הנמצאים בקידמתו ומכוונים לשני צידיו ב-90 מעלות.

הרכב יסע ישר ע"י אותו אלגוריתם בו השתמשנו בנסיעה ישר בשלב ב', ע"י השוואת מהירות הגלגלים ותיקונים מתאימים בהתאם. בנוסף לכך, באזור המכשולים נמצא קו שחור לאורך מרכזו הרכב נוסע בעת סריקת וזיהוי המטרות ולכן ניתן להתיישר גם לפי קו זה. בקוד שלנו נמצא אלגוריתם המזהה לפי הסנסורים לזיהוי הקו השחור הנמצאים בתחתית הרכב האם סטינו מנסיעה בקו ישר, במידה וכן יעשו תיקונים בהתאם.

כאשר אחד מהסנסורים, אולטרה סוניק הפועלים בזמן הסריקה, יזהה מטרה אחד מצידיו של הרכב נעבור למצב נסיעה ד'.

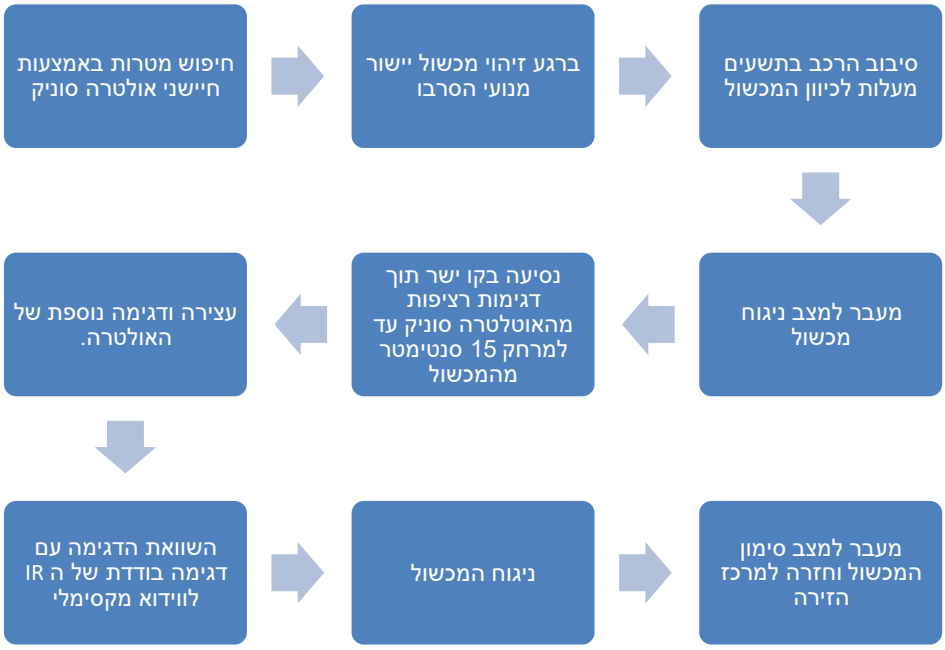
מצב נסיעה ד' כולל סיבוב ב-90 ע"י סיבוב מנוע גלגל אחד לכיוון המטרה שזוהתה וסיבוב מנועי הסרבו כך שהחיישנים יכוונו אל קדמת הרכב ויסתכלו ישר לזיהוי הגעה למטרה. לאחר מכן נעבור לנסיעה ישר אל כיוון המטרה עד למרחק 15 סנטימטרים מהמטרה שיחושב ע"י חיישני האולטרה סוניק ובמרחק זה נבצע דגימה אחת לווידוא המרחק ע"י סנסור ה-IR. לאחר הווידוא שאכן הגענו אל המטרה יתבצע ניגוח במטרה ומעבר למצב ה'.

במצב ה' ניסע ברוורס תוך דגימה ע"י הסנסורים הנמצאים בתחתית הרכב לזיהוי חזרה לקו השחור. כאשר יזהה הרכב את הקו השחור יסתובב חזרה 90 מעלות כך שיוכל להמשיך ליסוע ישר על הקו השחור ולהמשיך בחיפוש המטרות ע"י סיבוב מנועי הסרבו חזרה ל90 מעלות.

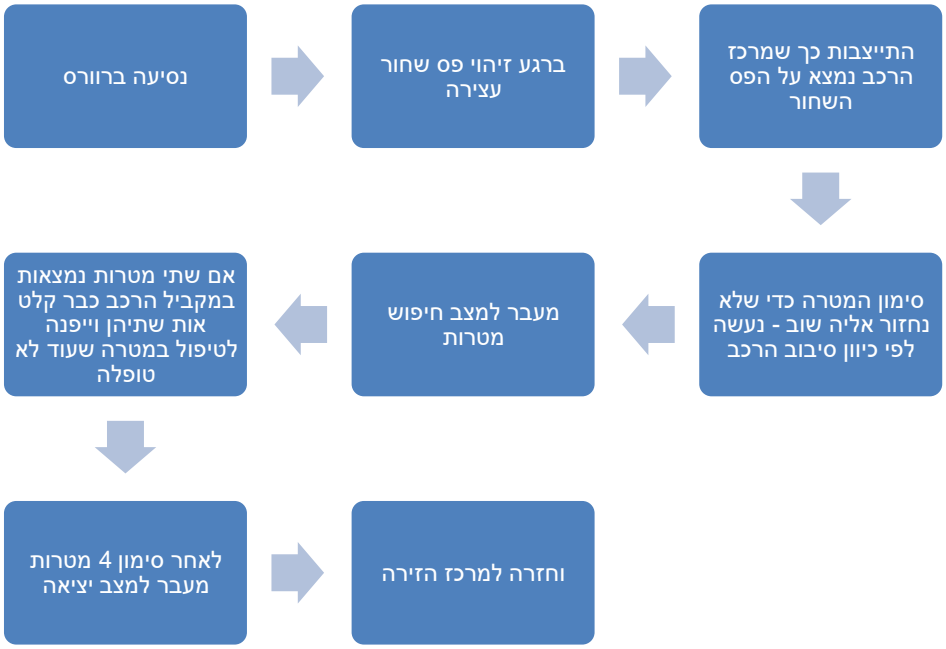
השלבים ד' וה' יעשו 4 פעמים לזיהוי ונגיעה ב-4 מטרות. ישנו אלגוריתם המטפל בכל המקרים ומקרי הקצה של זיהוי המטרות – זיהוי 2 מטרות במקביל משני צידי הרכב טיפול בכל אחת מהן, זיהוי כל מטרה פעם אחת בלי לחזור אליה שוב וכו'. בנוסף לכך, ישנו מונה שאכן סימנו והגענו לזיהוי 4 מטרות.

לאחר סיום שלב זה נעבור מנסיעה ישר וסריקת המטרות באזור זה למצב ז' לצורך זיהוי ההגעה אל הקיר של היציאה. לכן נסובב את הסרבו חזרה אל קידמת הרכב לפני מעבר המצב.

שלב ג+ד:


סיבוב הרכב בתשעים מעלות לכיוון המכשול
ברגע זיהוי מכשול יישור מנועי הסרבו
חיפוש מטרות באמצעות חיישני אולטרה סוניק
מעבר למצב ניגוח מכשול
נסיעה בקו ישר תוך דגימות רציפות מהאוטלטרה סוניק עד למרחק 15 סנטימטר מהמכשול
עצירה ודגימה נוספת של האולטרה.
מעבר למצב סימון המכשול וחזרה למרכז הזירה
ניגוח המכשול
השוואת הדגימה עם דגימה בודדת של ה IR לווידוא מקסימלי

שלב ה:


התייצבות כך שמרכז הרכב נמצא על הפס השחור
ברגע זיהוי פס שחור עצירה
נסיעה ברוורס
סימון המטרה כדי שלא נחזור אליה שוב - נעשה לפי כיוון סיבוב הרכב
מעבר למצב חיפוש מטרות
אם שתי מטרות נמצאות במקביל הרכב כבר קלט את שתיהן וייפנה לטיפול במטרה שעוד לא טופלה
וחזרה למרכז הזירה
לאחר סימון 4 מטרות מעבר למצב יציאה

שלבים ז'-ח'

שלב ז' – לאחר זיהוי כל המטרות נסיעה אל הקיר הנמצא בסוף הזירה.

שלב ח' – יציאה מהזירה דרך אחת משתי הדלתות הקיימות.

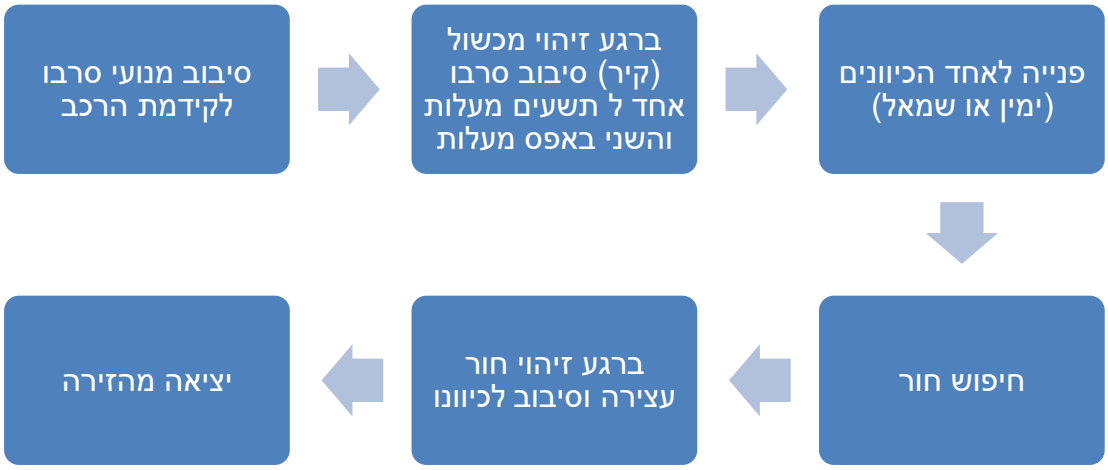
לאחר שסובבנו את הסנסורים לקדמת הרכב ע"י מנועי הסרבו נוכל להמשיך לסוע תוך כדי סריקה ע"י הסנסורים על מנת למצוא ולזהות את הקיר של היציאה.

הרכב יסע ישר ע"י אותו אלגוריתם בו השתמשנו בנסיעה ישר בשלב ב' ושלב ג', ע"י השוואת מהירות הגלגלים ותיקונים מתאימים בהתאם.

כאשר נזהה ע"י הסנסורים שהגענו למרחק 30 סנטימטרים מהקיר נעבור למצב ח' ונסתובב ב-90 מעלות ע"י סיבוב גלגל אחד בלבד לכיוון אחת היציאות.

נסובב את אחד הסרבו ל-90 מעלות – הסתכלות על הקיר לחיפוש היציאה(חור) והשני ישר מכוון לקדמת הרכב.

הסנסור המכוון הצידה וסורק את הקיר יקפיץ דגל ברגע שיזהה חור במספר דגימות עוקבות, לאחר הקפצת הדגל נסתובב אל היציאה בעזרת המנועים וניסע ישר ליציאה מהזירה.



פונקציות מרכזיות

1. **פונקציית תיקון מהירות ונסיעה ישר (ללא פס שחור):**

כדי לוודא נסיעה בקו ישר היה עלינו לוודא כי הגלגלים נוסעים באותו קצב.

על מנת לבצע זאת השתמשנו במשתנה גנרי בצורת define אשר אליו נזין את רמת החיכוך עם הריצפה ובהתאם אליו יתבצע סינכרון בין הגלגלים.

```
Inpute_Capture_ENC1();
Inpute_Capture_ENC2();

if (currentVelocity1 < currentVelocity2){//currentVelocity2 = TPM2_C1V, currentVelocity1 =
TPM1_C0V
TPM2_C1V -= (currentVelocity2 - currentVelocity1) * fractionRight;
TPM1_C0V += (currentVelocity2 - currentVelocity1) * fractionRight; }

else if (currentVelocity1 > currentVelocity2){//currentVelocity2 = TPM2_C1V, currentVelocity1
=  TPM1_C0V
TPM2_C1V += (currentVelocity1 - currentVelocity2) * fractionLeft;
TPM1_C0V -= (currentVelocity1 - currentVelocity2) * fractionLeft; }
```

ניתן לראות כי המשתנה הכללי שלנו הוא fractionLeft , fractionRight מכיוון שהמתח הנופל על כל מנוע DC אינו זהה נאלצנו להשתמש בשני משתנים.

אופן פעולת הפונקציה הוא קריאה לפונקציה אשר מבצעת inputCapture ומחשבת את מהירות כל גלגל ולאחר מכן חישוב הפרש המהירויות והכפלתו בחיכוך.

את הערך המתקבל מוסיפים לערך ספירה של הטיימר של הגלגל האיטי ומחסרים ערך זה מהערך ספירת הטיימר של הגלגל המהיר.

באופן זה אנו גם שומרים על סינכרון בין הגלגלים וגם מוודאים שהגלגלים לא יחרגו מתחום מסויים כי אנו מבצעים איזון בהחסרה והוספה על **שני** הגלגלים ולא רק על גלגל אחד.

נשים לב כי באופן זה אנו גם מוודאים נסיעה ישר כי אם היחס בין האנקורים הוא 1:1 אזי הגלגלים מסתובבים באותו קצב ולכן האוטו נוסע ישר.

2. **פונקציית נסיעה ישר בפס שחור:**

בפונקציה זו המזהה את הקו השחור הנמצא על הרצפה ע"י הסנסורים הנמצאים בתחתית המכונית נשתמש עבור שני מקרים:

א. עבור מצב נסיעה של סריקת המטרות בוא אנו נמצאים במרכז השירה וצריכים לסוע בקו ישר על הקו השחור ולכן נדאג שהקו השחור יימצא תחת הסנסור המרכזי המזהה אותו.

ב. עבור מצב נסיעה של חזרה למצב סריקת המטרות לאחר התנגשות במטרה, כלומר נסיעה ברוורס לאחר התנגשות וחיפוש הקו השחור ע"י הסנסורים. נחכה עד שהקו השחור יזוהה בכל הסנסורים הנמצאים בתחתית הרכב וכך נהיה בטוחים שחזרנו למרכז השירה וניתן להסתובב ולחזור לנסיעה בקו ישר על הקו השחור לצורך סריקת מטרות.

פונקציית black:

נקרא לפונקציה זו במצב נסיעה של סריקת מטרות ובמצב נסיעה רוורס – חזרה לסריקה לאחר התנגשות בלבד. הקריאה תעשה מקובץ ה- inputCapture לאחר דגימת האולטרה סוניק.

כאשר נגיע לפונקציה זו נזהה איזו בדיקה לזיהוי הקו השחור יש לבצע לפי מצב הנסיעה. עבור מצב סריקה נקרא לפונקציה מתאימה המוודא הימצאות הקו השחור במרכז המכונית ונסיעה בקו ישר – straight. עבור מצב חזר לסריקה נקרא לפונקציה מתאימה למציאת הקו השחור וחזרה למרכז – searchBlack.

```
void black()
{
	switch (driveMode){
	case 0://scanning
	straight();
	break;
	case 1://bump - no black strip
	break;
	case 2://door- no black strip
	break;
	case 3://pass - no black strip
	break;
	case 4://return
	searchBlack();
			break;


		}

}
```

פונקציית straight:

בודקת האם אנו במצב תקין – כלומר הקו השחור נמצא תחת הסנסור המרכזי מתוך חמשת הסנסורים הנמצאים מתחת למכונית לזיהוי הקו השחור. במידה ולא נחפש היכן כן זוהה הקו השחור וכך נדע מהי הסטייה מהמרכז ונסיעה בקו ישר ולאיזה כיוון ובהתאם נעשה תיקונים גסים או עדינים יותר.

```
void straight()
{
	if(RotationDirection=0)
	{
		if(!(GPIOE_PDIR & PORT_LOC(4)))//middle
		{

			if(GPIOE_PDIR & PORT_LOC(2))//left edge
			{
				TPM2_C1V-=4;
				TPM1_C0V+=4;
			}
			else if(GPIOE_PDIR & PORT_LOC(3))//left
			{
				TPM2_C1V-=2;
				TPM1_C0V+=2;
			}


			if(GPIOA_PDIR & PORT_LOC(16))//right edge
			{
				TPM2_C1V+=4;
				TPM1_C0V-=4;
```

```
                    }
                    else if(GPIOE_PDIR & PORT_LOC(5))//right
                    {
                            TPM2_C1V+=2;
                            TPM1_C0V-=2;
                    }
            }
    }
}
```

פונקציית sreachBlack:

בודקת האם זוהה הקו השחור בכל הסנסורים, כלומר המכונית נמצאת במצב מאונך לקו השחור אחרי שנסעה ברוורס והגיעה חזרה למרכז הזירה. במצב זה עלינו להסתובב חזרה לכיוון התנועה בקו ישר על הקו השחור.

ישנו דגל בו אנו משתמשים SawBlack שבהתאם אליו נדע האם עלינו לחזור למצב סריקה וזיהינו את הקו השחור או שעלינו להמשיך לסע ברוורס ולהמשיך לחפש.

הדגל יעלה במידה וזוהה עלייה בערך הלוגי בכל הפורטים המייצגים את חמשת הסנסורים שקלטו את הקו השחור.

```
void searchBlack()
{
        if((GPIOE_PDIR & 0x0000003C) && (GPIOA_PDIR & PORT_LOC(16)))
        {
                SawBlack= 1;
        }
        else
        {
                SawBlack= 0;
        }
```

3. פונקציות לוגיקת נסיעה

לוגיקת הנסיעה מתבסת על הימצאות הרכב במצב נסיעה מסוים לפיו יידע מה השלב הבא, היכן הוא נמצא בזירה ומה עליו לעשות בכל רגע נתון.

מצב הנסיעה מיוצג ע"י המשתנה driveMode. לפי משתנה זה במפר מקומות בקוד בכל פעם שהרכב צריך לפעול הפעולה תיעשה לפי המשתנה. עיקר אופן הפעולה ייבחר לאחר כל דגימה של האולטרה סוניק לפי ה cases הבאים המוצגים.

```
switch (driveMode) {
    case 0: //scanning
    black();
    maxSpeed();
    scanObstacle();
    break;
    case 1://bump
    bumpObstacle();
    break;
    case 4://return black line
    reverse();
    returnScanning();
    break;
    case 6:// search wall
    maxSpeed();
    searchWall();
    break;
    case 7:
    maxSpeed();// search exit
    searchExit();
    break;
    case 8://done
    maxSpeed();
    break;
    default :
    maxSpeed();
}
```

פונקציית scanObstacle:

בעת מצב סריקה נכנס לפונקצייה זו ונבדוק האם זיהינו מכשול בכל אחד מהצדדים. נפרט את כל המצבים עבור אחד הצדדים של האולטרה:

א. עבור case 0 – לא זוהה מכשול נוסף(בצד השני) ולכן ניתן להמשיך לעבור למצב נסיעה של התנגשות ללא כל טיפול נוסף ולשנות את מצב המכשולים (משתנה twoObstacles) שמסמן זיהוי וטיפול במכשול מצד ימין.

ב. עבור case 3 – הדגל עלה בבדיקה עבור מכשול מצד שמאל ולכן נרים דגל כי זוהה גם מכשול בימין ולכן אנו במצב 2 של זיהוי שני מכשולים בו זמנית משני צידי הרכב. נחכה לסוף טיפול צד שמאל ורק אז נפנה לטפל במכשול הנמצא מימין.

ג. עבור case 4 – חזרנו לאחר טיפול במכשול מצד שמאל (במקרה של שני מכשולים בו זמנית) וכעת ניתן לטפל במשתנה מצד ימין ולהוריד את הדגל לאחר מכן.

ד. עבור case 7 – סיימנו לטפל בכל המכשולים שהיו (להתנגש ולחזור) אם היו שניים או אחד וכעת נרצה להמשיך וליסוע מעט ישר ללא חיפוש מכשולי על מנת שלא נטפל ונתנגש באותתו המכשול פעמיים. לכן נחכה 10 דגימות לאחר סיום טיפול במטרה עד לזיהוי והמשך טיפול במטרות הבאות.

```
void scanObstacle() {
    if (Sonic2_Distance < 100 && counterObstacles4<4) {//if right sens
    switch (twoObstacles){
        case 0://no obstacles
            if (moneObstacle2 == 2) {
                ServosConfig_Max();
                moneObstacle2 = 0;
                RotateLeft();
                dealyRotationLeft(0);
                driveMode = 1; //bump
                RotationDirection = 1; //right
                stopMoving();
                maxSpeed();
                twoObstacles=1;//right flag on
                TurningBackToEncoders();
            } else if (moneObstacle2 < 2) {
                moneObstacle2++;
            }
        break;
        case 3://left
            twoObstacles=2;
        break;
        case 4://go to right after handling left
            if (moneObstacle2 == 2) {
                ServosConfig_Max();
                moneObstacle2 = 0;
                RotateLeft();
                dealyRotationLeft(0);
                driveMode = 1; //bump
                RotationDirection = 1; //right
                stopMoving();
                maxSpeed();
                TurningBackToEncoders();
            } else if (moneObstacle2 < 2) {
                moneObstacle2++;
            }
            twoObstacles=1;//right
            break;
        case 7://done with obstacles continue drive for some samples
            if(moneAfterReturn==10)
            {
                moneAfterReturn=0;
                twoObstacles=0;
                if (moneObstacle2 == 2) {
                    ServosConfig_Max();
                    moneObstacle2 = 0;
                    RotateLeft();
                    dealyRotationLeft(0);
                    driveMode = 1; //bump
                    RotationDirection = 1; //right
                    stopMoving();
                    maxSpeed();
                    TurningBackToEncoders();
                } else if (moneObstacle2 < 2) {
                    moneObstacle2++;
                }
            }
            else if(moneAfterReturn<10)
            {
                moneAfterReturn++;
```

באופן דומה מאוד הלוגיקה נעשית עבור צד שמאל.

פונקציית bumpObstacle:

```
void bumpObstacle() {
    if (Sonic1_Distance < 15 && Sonic2_Distance < 15 ){//|| (IR1_Distance == 15 && IR2_Distance == 15)) {
        stopMoving();
        driveMode = 4;
        TurningBackToEncoders();
    } else {
        TurningBackToEncoders();
    }
}
```

לאחר זיהוי מכשול באחד הצדדים ולאחר שפנינו לטפל בו ניסע תוך כדי סריקה עד ל-15 סנטימטר למכשול נוודא שהגענו למכשול ע"י ה-IR ונתנגש בו.

פונקציית returnScanning:

לאחר התנגשות במכשול נרצה לחזור למרכז הזירה לעקוב אחרי הקו השחור ולסרוק אחר מכשולים נוספים. נעשה זאת ע"י נסיעה רוורס וחיפוש הקו השחור על מנת לחזור לעקוב אחריו. נקרא לפונקציית black שלפי זיהוי מצב הנסיעה תקרא לפונקציה searchBlack שתעלה את הדגל SawBlack ל-1 במידה וזוהה הקו השחור ע"י הסנסורים.

```
void returnScanning() //driveMode=4
{
    black();
    if (SawBlack == 1) {
        if (RotationDirection == 1) //right
        {
            ServosConfig_Min();
            RotateRight();
            dealyRotationRight(0);
            stopMoving();
            RotationDirection =0;
            if(twoObstacles==1)
            {
                twoObstacles = 7;//go to handle other side
            }
            else if (twoObstacles==2)
            {
                twoObstacles=5;//go to left
            }
            SawBlack=0;
        }
        if (RotationDirection == 2) //left
            {
            ServosConfig_Min();
            RotateLeft();
            dealyRotationLeft(0);
            stopMoving();
            RotationDirection =0;
            if(twoObstacles==3)
            {
                twoObstacles = 7;
            }
            else if (twoObstacles==2)
            {
                twoObstacles=4;//go to right
            }
            SawBlack=0;
        }
        driveMode = 0; //scanning
        counterObstacles4++;//done with another obstacle
    }
```

במידה ואכן זה המצב נסתובב חזרה לנסיעה ישרה על הקו השחור והמשך חיפוש שאר המטרות.
כמובן שיש טיפול עבור העלאת דגלים למקרה של שני מכשולי בו זמנית והצורך לחכות מספר דגימות לאחר סיום טיפול במכשולים הנ"ל.

פונקציית searchWall:

```
void searchWall() {
    if (Sonic1_Distance < 30 && Sonic2_Distance < 30) {
        stopMoving();
        servoTurnRight(); //1 servo look at wall, 2 servo straight
        RotateRight();
        dealyRotationRight(0);
        stopMoving();
        driveMode = 7;
        RGB_LED_OFF;
    }
        else
     {
        TurningBackToEncoders();
     }
}

void searchExit() {
    if (Sonic1_Distance > 60) //noticed the exit-hole
            {
        if (moneExit == 4) {
            moneExit = 0;
            stopMoving();
            ServosConfig_Max(); //1 servo look at wall, 2 servo straight
            RotateLeft();
            dealyRotationLeft(0);
            stopMoving();
            TurningBackToEncoders();
            RGB_LED_ON;
            driveMode = 8;
        } else {
            moneExit++;
        }
    }
        else
     {
        TurningBackToEncoders();
     }
}
```

פונקציה זו מטפלת בחלק מהיציאה מהזירה. לאחר שזוהו 4 מטרות או שסיימנו לסוע מרחק מסוים והגענו לאזור סוף הזירה נתחיל בחיפוש הקיר וסריקה קדימה עם הסנסורים. ברגע שהסנסורים יזהו את הקיר במרחק 30 סנטימטר מהם יתבצע סיבוב לנסיעה לאורך הקיר עם סיבוב אחד הסנסורים על מנת שיפנה לכיוון הקיר כעת ויחפש את פתח היציאה.

פונקציית searchExit:

לאחר שהסתובבנו לנסיעה לאורך הקיר וסובבנו את הסרבו לחיפוש הפתח בקיר על מנת לצאת, נחכה עד לזיהוי מרחק הגדול מ-60 סנטימטר - כלומר זוהה חור ע"י האולטרה סוניק המחפש את היציאה. כאשר אכן תזוהה היציאה הרכב יסתובב לכיוונה וימשיך לסוע ישר על מנת לצאת מהזירה ולסיים את המסלול.

4. פונקציית מיקום על ציר y

כדי לדעת איפה אנחנו על ציר y לכל אורך הדרך השתמשנו בנתוני הזירה כדי לדעת בברור היכן אנחנו לכל אורך הדרך. הגדרנו משתנים גלובליים אשר סימנו לנו גם באיזה מקטע אנחנו (כניסה לדלת, חיפוש פס שחור, אזור מכשולים וחיפוש יציאה).

```
if(currentVelocity1 < 60 && currentVelocity2 < 60)
{
      if(currentVelocity1 > 0 && currentVelocity2 > 0)
          AvgVelocity = ((currentVelocity1+currentVelocity2)*0.004);
}

else if (currentVelocity1 < 60)
{
      if(currentVelocity1 > 0 )
            AvgVelocity = currentVelocity1*0.008;
}
else if (currentVelocity2 < 60)
{
      if(currentVelocity2 > 0)
            AvgVelocity = currentVelocity2*0.008;
}
else
      AvgVelocity = 0.336;
```

בעזרת המהירות **<u>הממוצעת</u>** שאנו מחשבים אנו מחסרים כל פעם מהמשתנה הגלובלי שלנו (אשר מקבל ערך בהתאם לחלק במשימה בו אנו נמצאים כרגע) את המהירות*זמן הכניסה לפונקציה (נמצאת ב PIT) וכך אנו יודעים כמה מרחק עברנו בכל רגע.
הערה: קיימת פונקציה נוספת בקוד אשר מבצעת את המיפוי של המכשולים הלכה למעשה, עליה נרחיב בהמשך.

5. פונקציות נסיעה וסיבוב

1. אנו נוסעים בקווים ישרים בלבד לכן נדרשות 4 פונקציות נסיעה – קדימה, אחורה, סיבוב ב 90 מעלות ימינה וסיבוב ב 90 מעלות שמאלה.
2. אופן המימוש זהה אנו שולחים לפונקציה מרכזית (motor dir and speed) את סוג הנסיעה בו אנו מעוניינים ובהתאם לנתון מקבלים את אופן התנועה הנדרש.

```
void Motor_Dir_and_Speed(int Direction, int Speed) {

      switch (Direction) {
      case 0:     //forward MOTOR1 & MOTOR2
            GPIOC_PDOR = 0;
            GPIOC_PDOR |= PORT_LOC(10) + PORT_LOC(6);       //0x0440
            break;
      case 1:     //backward MOTOR1 & MOTOR2
            GPIOC_PDOR = 0;
            GPIOC_PDOR |= PORT_LOC(7) + PORT_LOC(5);        //0xA0
            break;
      case 2: //turn right
            TPM2_C1V = MUDULO_REGISTER/2; //100% duty cycles /
            prescaler (= Speed)
            GPIOC_PDOR |= PORT_LOC(6);          //0x40
            break;
      case 3: //turn left
            TPM1_C0V = MUDULO_REGISTER/2;//100% duty cycles /
            prescaler (= Speed)
            GPIOC_PDOR |= PORT_LOC(10);         //0x400
            break;
      default:    //stops
            GPIOC_PDOR = 0;
            TPM2_C1V = TPM1_C0V = 0;
            break;
      }

      if (Direction < 2)
      {
            TPM2_C1V = MUDULO_REGISTER / Speed; //100% duty cycles /
            prescaler (= Speed)
            TPM1_C0V = MUDULO_REGISTER / Speed;//100% duty cycles /
            prescaler (= Speed)
      }
```

6. סיבוב מנועי סרבו

השליטה במנועי הסרבו נעשית בשני אופנים:

א. סיבוב באמצעות סריקה – שינויים של 10 מעלות כל פעם טווח תנועה מקסימאלי של 130 מעלות לכל כיוון.

ב. סיבוב חד על ידי הזנה ישירה של הזווית הרצויה.

<u>סיבוב חד</u>

```
//for bump
void  ServosConfig_Max(){//both motors are in the front of the car
looking strait

      TPM1_C1V = Zero_Degrees; // 0 degrees
      TPM0_C4V = Max_Degrees; // 180 degrees
      servo1State = servo2State = 0; //indicator for both motors are
strait
}

//for scanning
void ServosConfig_Min(){//both motors are at 90 degrees of the car
looking aside

      TPM1_C1V  = Min_Motor1; // 90 degrees
      TPM0_C4V  = Min_Motor0; // 90 degrees
      servo1State = servo2State = 1;//indicator for both motors are
looking aside
}

void servoTurnLeft(){
      TPM1_C1V = Zero_Degrees; // 0 degrees
      TPM0_C4V  = Min_Motor0; // 90 degrees
}
//for exit
void servoTurnRight(){
      TPM1_C1V  = Min_Motor1; // 90 degrees
      TPM0_C4V = Max_Degrees; // 180 degrees
                                                                        }
```

סיבוב באמצעות סריקה (נעשה בעזרת ה PIT)

```
if (Servo_Counter < 13) { //13) {
if (opMode == 2)//sides--->front
      TPM0_C4V += (Ten_degrees_Shift - 3); // -3 for calibration
purposes
      TPM1_C1V -= Ten_degrees_Shift;
      dealyUltra();
            }

else // front---->sides
{
      TPM0_C4V -= (Ten_degrees_Shift - 3); // -3 for calibration
purposes
      TPM1_C1V += Ten_degrees_Shift;
      dealyUltra();
}
```

7. פונקציית דגימה (UltraSonic Samples+Input Capture)

הפונקציות שלובות זו בזו וזה היה האתגר העקרי בפרוייקט מבחינתנו – כיצד לסנכרן בין אופני העבודה בצורה יעילה, שכן המודולים מתחברים לאותו הטיימר באותם הערוצים וכדי לעבור ממצב אחד לאחר היה עלינו לקנפג כל פעם מחדש את המודול.

אופן הפעולה – **להוסיף**

```
void FTM0_IRQHandler() {
      // getting the capture and counter values


if (TPM0_STATUS & 0x08) //CH3 Interrupt
                  {

TPM0_STATUS |= 0x08; //Resets the interrupt flag of the channel;

TPM0_C3SC |= TPM_CnSC_CHF_MASK; //resets interrupt flag at the
channel


if (TPM0_MOD == 0xC350 && degel1 == 0) //ultra sonic
                         {


if (RisingEdge1 == 1) //from low to high
                              {

RisingEdge1 = 0;

sonicTOF1 = 0;

Sonic1 = TPM0_C3V ; //the value of the timer when we sent it


}

else //from high to low

{

      RisingEdge1 = 1;


Sonic1_tag = TPM0_C3V;


if (sonicTOF1 == 1)

{

Sonic1_tag += 0xC350;

}

else if (sonicTOF1 == 2)

{

Sonic1_tag += 2*0xC350;
```

```
}

Sonic1 = (Sonic1_tag - Sonic1); //the delta counter after returning

Sonic1_Distance = Sonic1*0.0227;// - 8;// the distance measure by the
formula

degel1 = 1;

}

}

else if ( TPM0_MOD != 0xC350 ) //encoders

{

Motor1_Old = tmp1; //now the new is not updated

tmp1 = Motor1_New = TPM0_C3V;//sets the TMP current value

if (TOF1 == 1)

Motor1_New += MUDULO_REGISTER;

else if (TOF1 == 2)

Motor1_New += 2*MUDULO_REGISTER;

TOF1 = 0;//Resets the counter for ENC1

PIT_MCR &= ~PIT_MCR_MDIS_MASK;// enable the PIT module

}

}

else if (TPM0_STATUS & 0x04) //CH2 Interrupt - motor2 TPM2_C1V

{

TPM0_STATUS |= 0x04; //Resets the interrupt flag of the channel;

TPM0_C2SC |= TPM_CnSC_CHF_MASK;//resets interrupt flag at the channel

if (TPM0_MOD == 0xC350 && degel2 == 0)//ultra sonic

{

if (RisingEdge2 == 1) //from low to high
```

```
{
RisingEdge2 = 0;
sonicTOF2 = 0;
Sonic2 = TPM0_C2V; //the value of the timer when we sent it

}
else //from high to low
{
RisingEdge2 = 1;
Sonic2_tag = TPM0_C2V;
if (sonicTOF2 == 1)
{
      Sonic2_tag += 0xC350;
}
else if (sonicTOF2 == 2)
{
      Sonic2_tag += 2*0xC350;
}

Sonic2 = (Sonic2_tag - Sonic2); //the delta counter after returning
Sonic2_Distance = Sonic2*0.0227;// - 8;// the distance measure by the
formula

degel2 = 1;

}
}
else if (TPM0_MOD != 0xC350) //encoders
{
Motor2_Old = tmp2; //now the new is not updated
tmp2 = Motor2_New = TPM0_C2V;//sets the TMP current value

if (TOF2 == 1)
```

```
Motor2_New += MUDULO_REGISTER;

else if (TOF2 == 2)

Motor2_New += 2*MUDULO_REGISTER;

TOF2 = 0;//Resets the counter for ENC2

PIT_MCR &= ~PIT_MCR_MDIS_MASK;// enable the PIT module

}

}

else if (TPM0_STATUS & 0x0100 || TPM0_SC & 0x80) //TOF

{

TPM0_STATUS |= 0x0100; //resets TOF flag at STATUS

TPM0_SC |= 0x80;//resets TOF flag at SC

if(TPM0_MOD != 0xC350)

{

TOF1++;

TOF2++;

}

else if (TPM0_MOD == 0xC350)

{

sonicTOF2++;

sonicTOF1++;

}

}

if ((TPM0_MOD == 0xC350 && degel1 + degel2 == 2) || (TPM0_MOD ==
0xC350 && sonicTOF1 + sonicTOF2 > 2 ))

{

if (TPM0_STATUS & 0x08)

TPM0_STATUS |= 0x08; //Resets the interrupt flag of the channel;

if (TPM0_STATUS & 0x04)

TPM0_STATUS |= 0x04;//Resets the interrupt flag of the channel;
```

```
if(TPM0_STATUS & 0x0100 || TPM0_SC & 0x80)
{
TPM0_STATUS |= 0x0100; //resets TOF flag at STATUS
TPM0_SC |= 0x80;//resets TOF flag at SC
}
if(sonicTOF2 > 2 || Sonic2_Distance > 120)
Sonic2_Distance = 900;
if(sonicTOF1 > 2 || Sonic1_Distance > 120)
Sonic1_Distance = 900;
```