

המכללה הטכנולוגית של חיל האוויר שלוחת באר-שבע

# פרויקט גמר

## הנדסאי אלקטרוניקה

### בהתמחות מערכות אלקטרוניות

הנושא: רובוט אוסף קוביות.

מגישים: ירין אביסידריס ועידן לישע.

מנחה: לודמילה קוזינץ.

# הבעת תודה

ברצוננו להודות למספר אנשים שעזרו, תמכו ותרמו מזמנם לעשייה וגימור הפרויקט.

בראש ובראשונה תודה למנחת הפרויקט - לודמילה קוזניץ, אשר תמיד עזרה והקשיבה לאורך כול תהליך בניית הפרויקט וכמובן הדריך והראה את הדרך הנכונה והשלבים לעשייה בצורה הטובה ביותר.

תודה למורים – ישראל זוניס ויעקב מרש, שליוו אותי במשך כול תקופת הלימודים, ונתנו לי את הידע והבסיס לעשיית הפרויקט ותמיד נענו בחיוב לעזרה בשעת הצורך.

## **תוכן עניינים:**

### **פרק 1 – מבוא**

5	1.1 מבוא הפרויקט
6	1.2 הוראות הפעלת המערכת
7	1.3 תרשים מלבנים
8-9	1.4 הסבר תרשים מלבנים
8	1.4.1 מיקרו-בקר P89V51RD2
8	1.4.2 דרייבר למנועי DC-L293D
8	1.4.3 מנועי DC
8	1.4.4 מערך מקלטי/משדרי IR
9	1.4.5 חוצץ HCT24474
9	1.4.6 מנועי סרבו
9	1.4.7 תצוגת LCD

### **פרק 2-מעגל חשמלי**

10	2.1 סכימה חשמלי
10-34	2.2 הסבר סכימה חשמלית
11-20	2.2.1 מיקרו בקר P89V51RD2
20	2.2.2 תזמון המיקרו-בקר בתדר קבוע
20	2.2.3 מעגל ריסט ( Reset Circuit )
21-24	2.2.4 מעגל בקרת מנועים
26-28	2.2.5 חיישן מרחק מבוסס קול אולטרסאונד (HC-SR04)
29	2.2.6 ממיר מסוג STEPDOWN
29	2.2.7 PL-IRM0101
30-34	2.2.8 מנועי סרבו ( Servo Motor )

### **פרק 3 –פרוטוקולים**

35-38	3.1 פרוטוקול RECS (IR)
-------	------------------------

### **פרק 4-תוכנית**

39	4.1 תוכנית בשפת C
40	4.1.1 תרשים זרימה
41-51	4.1.2 קבצי h
52-59	4.1.3 תכנית ראשית
60-62	4.1.4 הסבר פונקציות

## **פרק 5-סימולציות**

63-67.....Keil ב- PWM סימולציית 5.1

## **פרק 6-סיכום ומסקנות**

69.....סיכום ומסקנות 6.1

69.....תקלות במהלך הפרויקט 6.2

## **פרק 7-זיווד 71-73**

## **פרק 8-ביביליוגרפיה**

75.....אתרים 8.1

## **פרק 9-נספחים**

76-82.....P89V51RD2 מיקרו-בקר 9.1

83-84.....L293D למנוע זרם דוחף 9.2

85-86.....(אפנון רוחב הפולס) PWM אפנון 9.3

87-88.....(SR04) מד מרחק 9.4

89.....DC Motor 9.5

90.....(SERVO) מנועי סרבו 9.6

91-93.....(Step Down) BUCK מסוג מייצב ממותג 9.7

## פרק 1:

### מבוא

בפרויקט זה פותח ומומש רובוט בצורת טנק שעליו מורכבת זרוע מכנית אשר נועדה להרים חפצים מהרצפה ולהחזירם אל ארגז אשר מונח בפינת החדר.

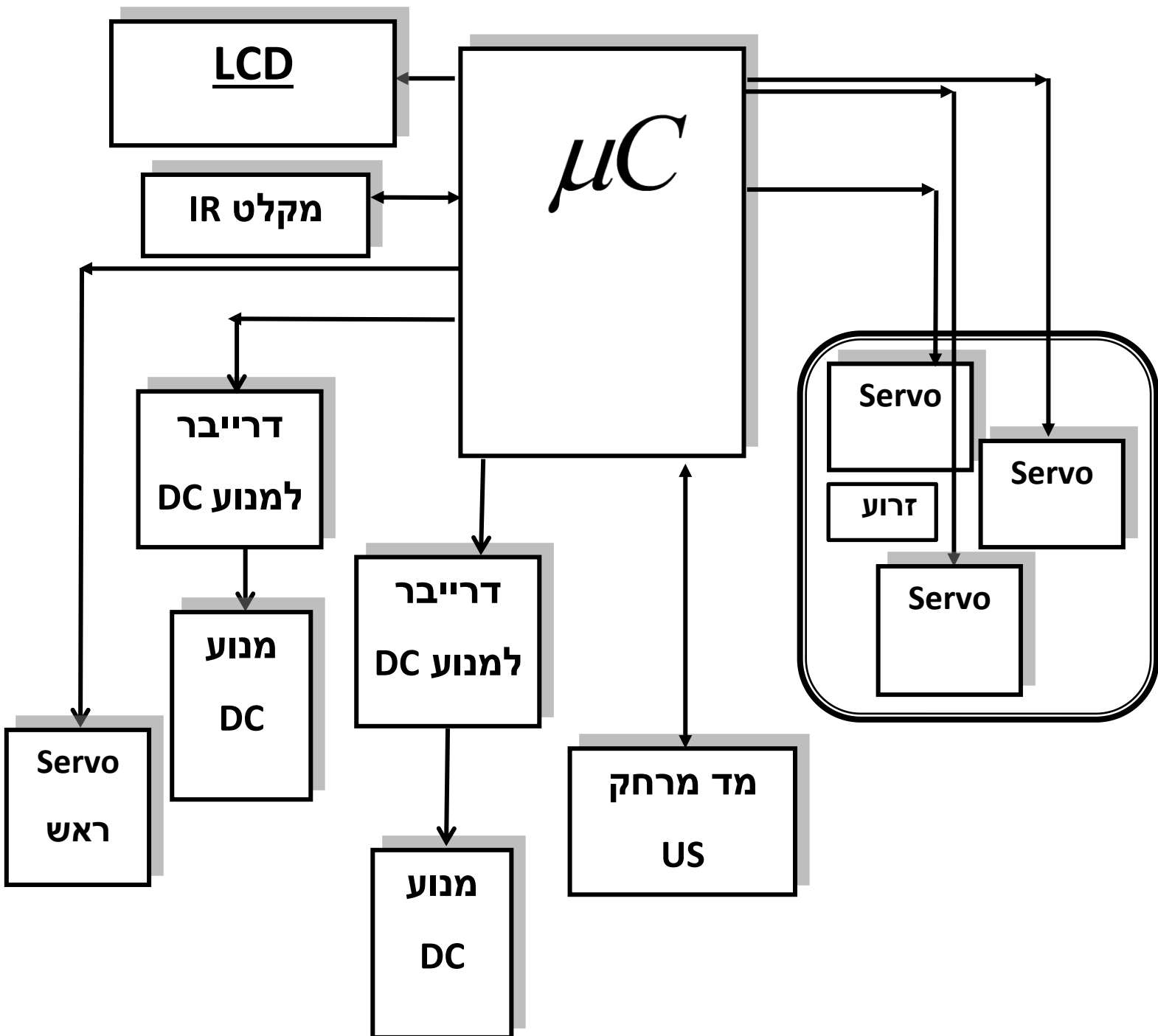
הפרויקט מורכב מהרובוט עצמו ושלט לתפעולו של הרובוט. הרובוט מורכב מבסיס טנק בעל שני מנועי DC, גלגלים המשולבים עם רצועות גומי, מיקרו אשר משמש כליבה של המערכת ודרכו מתנהלות כול הפקודות והנתונים לשאר הרכיבים, מתח הספקה על מנת לתפעל את כול רכיבים, פרוטוקול IR שתפקידו לקלוט מהשלט מידע, מייצב מתח של 5v לרכיבים הלוגים הנמצאים במעגל, חיישן מרחק אשר יהווה עבור הרובוט כבקרה להפסקת תנועתו בהבחנת מכשול – כלומר החיישן יחובר למנוע סרבו על מנת שינוע בתנועת סריקה ימינה ושמאלה, מצב זה פועל רק במצב אוטומטי בלבד. השלט מורכב מפרוטוקול IR שתפקידו זהה ל- IR המצוי על הרובוט ובנוסף לשמש לתפעול שני מצבים של הרובוט, סוללת הספקה לשם תפעול השלט, לחצני חצים ולוח מקשים ממוספר למימוש תנועתו של הרובוט והזרוע.

הרובוט יבצע את הפעולות אשר המשתמש יגדיר לו כגון: נסיעה קדימה, רגרס, פניות ימינה ושמאלה תוך מתן התחשבות לסביבתו, במצב אוטומטי כאשר הרובוט נמצא בחדר מרובע במידה וישנם מכשולים אשר מפריעים לו בדרך הוא ימנע מהם ע"י שינוי כיוון תנועתו לשם השלמת הנסיעה בצורה מושלמת ולאחר מכאן ישוב לביצוע פעולות המשתמש.

## **1.2 הוראות הפעלת המערכת**

1. מעבירים את הרובוט למצב כוח חיצוני (מתח סוללה אינו מקבל מתח מכניסת ה-usb).
2. מחברים פיזית את הכבלים של הסוללה לרובוט.
3. משתמשים במקור מתח נפרד של 5v שיכול לספק עד 3A לפי הדרישות להפעלת שלושת מנועי הסרבו של הזרוע במקביל.
4. כרגע הרובוט מוכן לקבל פקודות מהשלט כלומר הרובוט מוכן לשימוש (נסיעה + זרוע).

1.3 תרשים מלבנים



#### 1.4 הסבר תרשים מלבנים

##### 1.4.1: מיקרו בקר P89V51RD2

המיקרו בקר הוא היחידה המרכזית במעגל החשמלי.  
תפקידו הוא לבקר על פעילות כל היחידות המחוברות אליו.  
תפקידיו העיקריים של המיקרו בקר :  
הפעלת מנועי ה-DC ע"י שליחת נתונים לדרייבר L293D.  
עיבוד הנתונים המתקבלים מחיישני ה-IR ומחיישן ה-Ultrasonic.  
יצירת פולסים מתאימים למנועי הסרבו בשביל תפעול הזרוע והזזת הראש עליו יושב החיישן מרחק. (PWM).

##### 1.4.2 דרייבר למנועי L293D-DC :

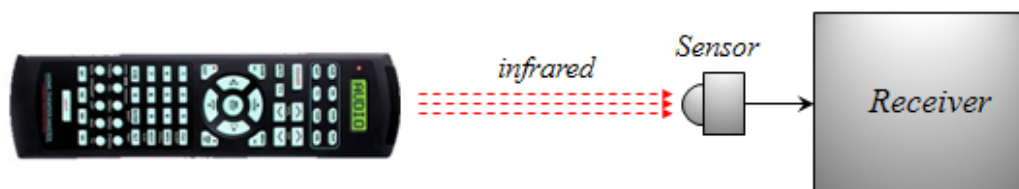
רכיב דוחף זרם אל המנועים. מחובר בין המיקרו בקר והמנועים.  
תפקידו הוא לספק את הזרם הנדרש למנועים להפעילם בהתאם למידע שנשלח אליו מהמיקרו בקר.

##### 1.4.3 מנועי DC :

רכיב אלקטרו מכאני, המבצע המרה של אנרגיה חשמלית לאנרגיה סיבובית מכאנית.  
תפקיד המנועים הוא לסובב את הגלגלים כך שהרובוט יוכל לנסוע.

##### 1.4.4 מערך מקלטי/משדרי IR :

השלט IR שולח אור infra-red אשר נקלט בעינית המקלט, המידע משודר באמצעות האור, גל הנושא שלנו מאופנן בתדר  $38\text{kHz}$ , הסיביות והזמנים שלהם מוגדר לפי פרוטוקול NEC \80 – RECS.



##### 1.4.6 חוצץ 74HCT244 :

תפקידו להעביר את אות המבוא למוצא מבלי להעמיס על המיקרו-בקר, הגברת זרם האות שיוצא מהבקר.  
תפקידו העיקרי הוא להגן על הבקר.



#### **1.4.7 מנועי סרבו:**

אנו משתמשים ב 3 מנועי סרבו עבור הזרוע כאשר 2 מתוכם לצירים של הזרוע ימינה\שמאלה, למעלה\למטה ומנוע אחרון לסיגרה ופתיחה של צבת הזרוע.

המנוע הרביעי משמש כ"ראש" להזזה של החיישן מרחק שמאלה וחזרה קדימה כדי לקבל משוב על המרחק של הרובוט מאובייקטים קדימה\שמאלה ועל ידי כך לצאת מהחדר שבנינו.

מנוע הסרבו שונה ממנוע ה DC בכך שהוא מקבל פולסים ולפי כך הוא מבין לאיזה כיוון הוא צריך להזיז את הגיר שלו בכדי להגיע לזווית הרצויה.

#### **1.4.8 תצוגת LCD:**

רכיב LCD הוא קיצור של Liquid Crystal Display ובעברית תצוגת גביש נוזלי. הינו התקן אופטואלקטרוני אשר מסוגל להציג אותיות, מספרים ותווים אחרים.

הוא משמש כיחידת פלט במערכת.

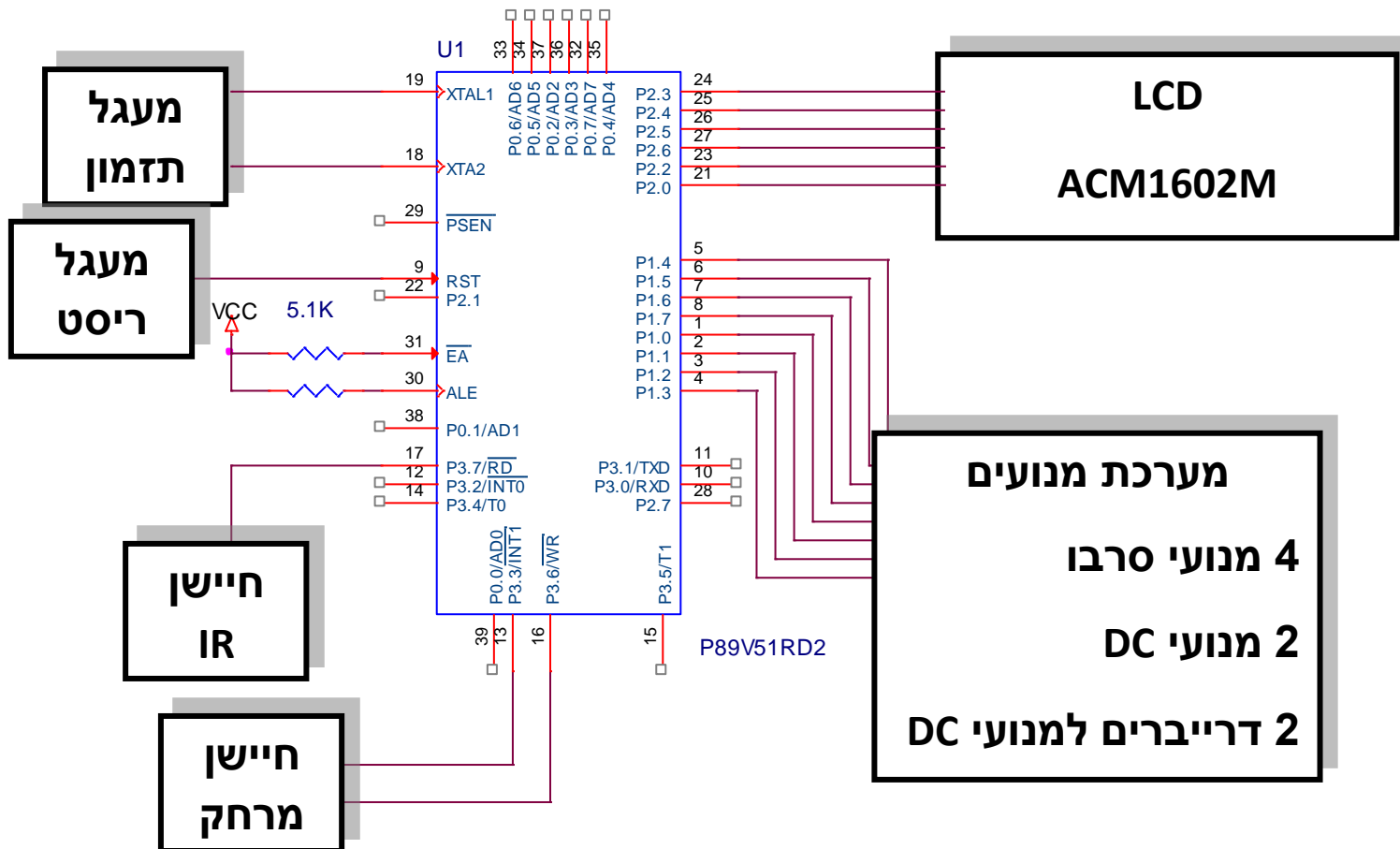
בעזרת ה lcd קיבלנו משוב על מדידת המרחק, השתמשנו בתצוגה כדי לראות שהרובוט נכנס למצבים שאנו רצינו להכניס אותו ובכך יכולנו לאתר שגיאות ובעיות שהיו בדרך.





## 2.2 הסבר סכימה חשמלית

### 2.2.1 מיקרו בקר P89V51RD2:





## **מאפיינים:**

מאפיינים:

P89V51RD2 של חברת NXP הוא מיקרו בקר מסדרת 80C51 עם זיכרון תוכנה 64KbytesFlash ו-1024BytesRAM.

CP של מיקרו בקר 80C51.

זיכרון תוכנה פנימי 64Kbytes Flash.

UART משופר.

PCA – Programmable Counter Array – מערך מונים הניתנים לתכנות הכוללים פונקציית PWM.

4 פורטים של קלט/פלט מקביליים בגודל 8bit כל אחד.

3 טיימרים/מונים בגודל 16bit.

Watchdog טיימר הניתן לתכנות.

8 מקורות פסיקה עם 4 רמות של עדיפויות.

## **תיאור חיבורים במיקרו בקר:**

- הדקים X1 ו-X2 הם קווי חיבור של גביש לתזמון הרכיב כאשר תדר העבודה הוא תדר הגביש.
- PORT1 – אליו מתחברים מנועי ה-DC, הדרייברים שלהם ומנועי הסרבו של הזרוע, דרך רכיב הגנה.  
למנוע אחד מתחברים ההדקים P1.0 ו-P1.1 ולמנוע השני מתחברים ההדקים P1.2 ו-P1.3.  
על מנת להפעיל את המנוע צריך להיות הפרש מתח בין הפורטים המחוברים אליו.  
הדקים P1.4 עד P1.7 מחוברים למנועי הסרבו אשר מפעילים את הזרוע.
- P2.0 ו-P2.2 עד P2.6 אליהם מחובר ה-LCD המשמש לנו לבדיקת נתוני הרובוט.
- P3.7 מחובר לחיישן IR אשר נועד לקבל פקודות מהשלט IR לגבי תנועות הרובוט והזרוע על ידי פרוטוקול IR.
- P3.3 ו-P3.6 אליהם מחובר חיישן האולטרסאונד, P3.3 הדק זה הוא פסיקת INT1 המחובר להדק ה-ECHO של חיישן האולטרסאונד. הדק P3.6 משמש להדק ה-TRIG.

## תכונות מיוחדות של המיקרו בקר:

### פסיקות:

פסיקה זה גורם המפסיק את הפעילות השגרתית של המעבד.

כאשר מתקבלת פסיקה המעבד יפסיק את הרצת התכנית הראשית והוא יבצע את תכנית הפסיקה. בסיום תכנית הפסיקה המעבד יחזור לפעילות השגרתית עד שתרחש פסיקה נוספת.

אוגרי המעבד :

Interrupt enable register 0 – IEN0 – אוגר אפשר פסיקות.

Bit	7	6	5	4	3	2	1	0
Symbol	EA	EC	ET2	ES	ET1	EX1	ET0	EX0

EX0 – אפשר פסיקה חיצונית 0.

ET0 – אפשר פסיקת טיימר 0.

EX1 – אפשר פסיקה חיצונית 1.

ET1 – אפשר פסיקת טיימר 1.

ES – אפשר פסיקת פורט טורי.

ET2 – אפשר פסיקת טיימר 2.

EA – על מנת להשתמש בפסיקה כלשהי יש לעלות את EA ל-1'. על מנת לחסום את כל הפסיקות יש להוריד את EA ל-0'.

שימושים של הפסיקות בפרויקט:

פסיקת INT1 פסיקה חיצונית 1 פסיקה זו קשורה למדידת המרחק בעזרת החיישן ultrasonic .

Interrupt priority – IP0 – אוגר לקביעת סדר עדיפויות של הפסיקות.

Bit	7	6	5	4	3	2	1	0
Symbol	-	PPC	PT2	PS	PT1	PX1	PT0	PX0

על מנת לתת עדיפות גבוהה יותר לאחת הפסיקות יש לעלות את הסיבית המתאימה ל-1'.

כאשר כמה פסיקות בעלות אותה רמת עדיפות, העדיפות תהיה בהתאם לסדר העדיפויות כך ש-PX0 היא בעלת העדיפות הגבוהה ביותר.

**TMOD – Timer/Counter mode control register** – אוגר לבקרת אופן עבודה של טיימר/מונה.

Bit	7	6	5	4	3	2	1	0
Symbol	T1GATE	T1C/ $\bar{T}$	T1M1	T1M0	T0GATE	T0C/ $\bar{T}$	T0M1	T0M0

**CT – Counter/Timer** – סיבית זאת קובעת פעולת מונה אירועים או טיימר.

'0' בסיבית זו קובע אופן פעולה של טיימר. (מהשעון הפנימי).

'1' בסיבית זו קובע אופן פעולה של מונה. (כניסה מהדק Tx של המיקרו בקר).

**GATE** – כאשר סיבית זו ב-'1' המונה/טיימר יופעל רק כאשר בהדק INTx יש '1' ובהדק TRx יש '1'.

כאשר סיבית זו ב-'0' הטיימר יופעל כאשר בהדק TRx יש '1'.

**M1, M0** – סיביות לקביעת אופן הפעולה של הטיימר/מונה.

הטבלה הבאה מציגה את אופני הפעולה הנפוצים יותר:

סיבית M1		סיבית M0	תאור מצב העבודה של המונה	
0	1	Mode 1	שני האוגרים TH ו TL משמשים יחד כמונה אחד בן 16 סיביות. כאשר המונה מגיע לערך 0xFFFF, בדופק השעון הבא המונה יעבור ל 0x0000. דגל הפסיקה יעלה לאחד לוגי והמעבד יעבור לבצע תוכנית פסיקה במידה ואושרה.	
1	0	Mode 2	האוגר TL משמש כמונה 8 סיביות. האוגר TH משמש כנועל ערך התחלתי עבור האוגר TL. בכל פעם שאוגר TL מסיים ספירה כלומר מגיע ל- 0xFF, דגל הפסיקה עולה לאחד לוגי והערך שנימצא באוגר TH נטען שוב לאוגר TL וחוזר חלילה.	

שימושים של הטיימרים בפרויקט:

- במצב הבקרה הידנית כאשר הרכב נתון לשליטה על ידי השלט, timer0 נמצא ב Mode2 8bit טעינה אוטומטית כדי ליצור פולס מתאים לשליטה על מנועי הסרבו כמו שידוע מנועי הסרבו דורשים PWM כדי לסובב את הגיר שלהם לזווית הרצויה. Timer1 נמצא ב Mode1 16bit כדי לקלוט נתונים מהחיישן IR.
- במצב הבקרה אוטומטית Timer0 נשאר באותו מצב כדי לשלוט על מנועי הסרבו, אך Timer1 כרגע עם פסיקות כדי למדוד את המרחק מחיישן ה ultrasonic השינוי זה בטבלה הבאה:

Bit	7	6	5	4	3	2	1	0
Symbol	T1GATE	T1C/T	T1M1	T1M0	T0GATE	T0C/T	T0M1	T0M0

GATE של timer1 "1" בהגדרה זו רק כאשר הדק ה INT1 יעלה ל 1 הטיימר יחל לעבוד וכאשר ייפול ל '0' לוגי יתקבל פסיקה INT1 שהיא מטפלת במדידת המרחק.

Timer/Counter control register – TCON – אוגר לבקרת מונה/טיימר.

Bit	7	6	5	4	3	2	1	0
Symbol	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IT0,1 – סיביות אלו קובעות מתי תתרחש פסיקה.

'0' בסיבית זו – פסיקה תתרחש כאשר INT0,1 ברמה נמוכה.

'1' בסיבית זו – פסיקה תתרחש כאשר יש עלייה מ-'0' ל-'1' ב- INT0,1.

IE0,1 – סיבית זו עולה ל-'1' כאשר התרחשה פסיקה.

סיבית זו מתאפסת כאשר מתבצעת תכנית הפסיקה.

TR0,1 – סיביות להפעלה/כיבוי של הטיימר/מונה ('1' – הפעלה, '0' – כיבוי).

TF0,1 – סיביות אלו עולות ל-'1' כאשר הטיימר/מונה סיים את הספירה.

סיביות אלו מתאפסות כאשר המעבד מבצע את תכנית הפסיקה הרלוונטית.

שימושים של אוגר זה בפרויקט:

- במצב בקרה אוטומטית של הרובוט אנו משתמשים בחיישן ה ultrasonic, ובכדי להשתמש בו אנו צריכים להגדיר את הפסיקה int1 וקבענו IT1=1 זאת אומרת תתקבל פסיקה כאשר הדק ה ECHO של החיישן ירד ל '0' לוגי. להלן קטע קוד:

```
1. EX1=1; //enabling interrupt 1
2. TR1=1; // running timer 1, timer runs when gate is 1.
3. IT1=1; // falling edge interrupt 1
4. EA=1; // enabling all interrupts
```



## – Programmable Counter Array – PCA

מערך המונים PCA מורכב מחמשה מודולים הניתנים לתכנות באופני עבודה שוני וממונה משותף בן 16 סיביות המורכב משני אוגרים של 8 סיביות (CH ו CL).

אחד מאופני העבודה של ה- PCA הוא PWM שבו אנו משתמשים לבקרה על זווית גיר מנועי הסרבו.

אוגרים מיוחדים שבעזרתם מאתחלים את ה- PCA :

### - PCA counter control register – CCON

Bit	7	6	5	4	3	2	1	0
Symbol	CF	CR	-	CCF4	CCF3	CCF2	CCF1	CCF0

CCFX – סיביות פסיקה המשמשות עבור 5 המודולים. סיביות אלו עולות ל-'1' כאשר מתרחשת פסיקה מתאימה (אנחנו לא משתמשים בסיביות אלו בפרויקט).

CF – PCA Counter Overflow Flag – סיבית זו עולה ל-'1' כאשר המונה של ה- PCA מסיים את הספירה. ניתן להשתמש בדגל זה בשביל פסיקה של המונה.

CR – PCA Counter Run Control Bit – באמצעות סיבית זו מפעילים את המונה.

ההוראה שאנו משתמשים על מנת להפעיל את המונה :

1. CCON=0x40;

### PCA counter mode register – CMOD

Bit	Symbol	Description
7	CIDL	Counter Idle Control: CIDL = 0 programs the PCA Counter to continue functioning during Idle Mode. CIDL = 1 programs it to be gated off during idle.
6	WDTE	Watchdog Timer Enable: WDTE = 0 disables Watchdog timer function on module 4. WDTE = 1 enables it.
5 to 3	-	Reserved for future use. Should be set to '0' by user programs.
2 to 1	CPS1, CPS0	PCA Count Pulse Select (see Table 37 below).
0	ECF	PCA Enable Counter Overflow Interrupt: ECF = 1 enables CF bit in CCON to generate an interrupt. ECF = 0 disables that function.

CIDL-בקרת מנייה במצב IDLE כאשר '0' המונה ימשיך לפעול במצב IDLE, כאשר הוא יהיה '1' ה- PCA לא יפעל במצב IDLE.

WDTE- אפשר הפעלת טיימר של WATCHDOG ("כלב שמירה") תפקיד ה- WATCHDOG בבקר היא לבדוק שהבקר עובד כשורה ועושה את כל הפעולות כמתוכנן וגם בודק שהבקר מפיק תדר שעון שזה הרי גורם חשוב בפרוייקט.

כאשר WDTE=0 טיימר של ה- WATCHDOG לא יפעל וכאשר '1' הטיימר יפעל.

CPS1, CPS0-סיביות אלו משמשים ב בחירת פולס מתאים לביצוע המנייה של ה PCA מתוך 4 אופציות שהיצרן מציע.

ECF- אפשר פסיקת המנייה של PCA כאשר מתקבלת גלישה, כאשר סיבית זו '1' לוגי היא מאפשרת לסיבית CF באוגר CCON ליצור פסיקה, כאשר סיבית זו '0' היא מחסלת אפשרות זו.

#### Counter pulse select options:

CPS1	CPS0	Select PCA input
0	0	0 Internal clock, $f_{osc} / 6$
0	1	1 Internal clock, $f_{osc} / 6$
1	0	2 Timer 0 overflow
1	1	3 External clock at ECI/P1.2 pin (max rate = $f_{osc} / 4$ )

באמצעות הפקודה הבאה אנו בוחרים פולס ספירה מתאים בין 4 האפשרויות שהיצרן מציע, אנו בחרנו לעבוד עם גלישת timer0 כדי לבצע את הספירה.

1. CMOD=4;

#### CCAPMn PCA Modules Compare/Capture register

Bit	7	6	5	4	3	2	1	0
Symbol	-	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn

• חשוב לציין שאוגר זה רלוונטי עבור הפורטים P1.3-P1.7 שרק עבורם אפשר להוציא אות PWM עבור מנועי הסרבו, היצרן מציע 5 הדקים שעבורם אפשר להוציא PWM ברקע לתוכנית הראשית.

ECOMn-מאפשר את פעולת המשווה עבור הדק ה n, '1' לוגי יאפשר את הפעולה.

CAPPn- אופצית ה CAPTURE חיובית עבור הדק ה n, '1' אומר שהוא עובד עבור עלייה חיובית.

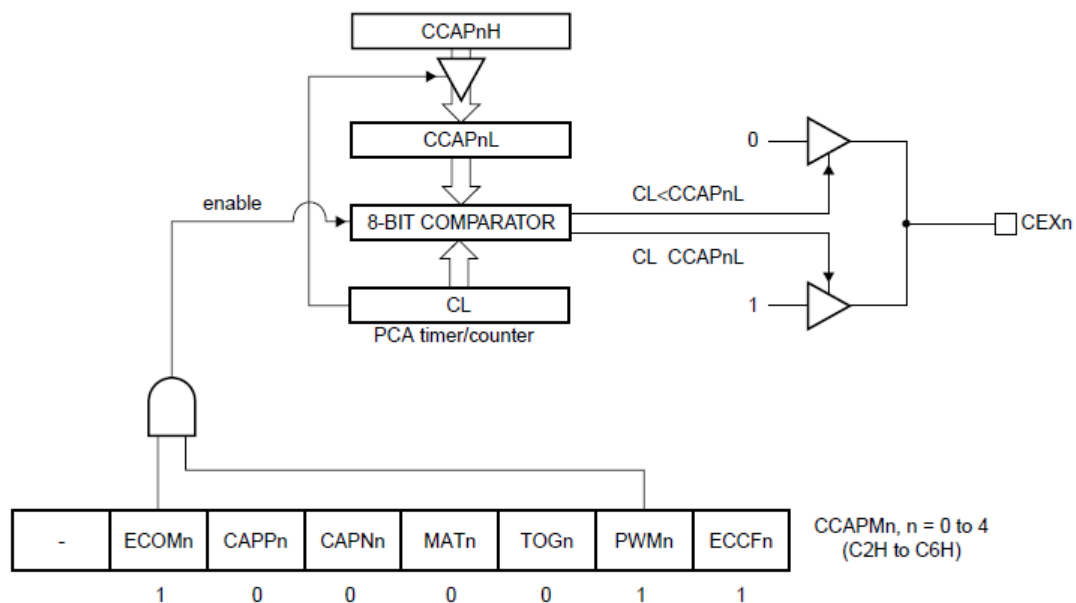
CAPNn- אופצית ה CAPTURE שלילית עבור הדק ה n, '1' אומר שהוא עובד עבור ירידה בהדק ה n.

TOGn- כאשר סיבית זו '1' וכשיש שיוון בין המנייה של ה PCA לבין המנייה של אוגר זה הדק ה n שנבחר מבצע מיתוג בין המצב הנוכחי שלו למצב ההופך (TOGGLE)

PWMn- סיבית זו מאפשר את ההדק n להיות מוצא PWM

להלן איור שמתאר את ההדק כשהוא מתפקד עם PWM ביציאה שלו:

באמצעות אוגרים CCAPnH ו-CCAPnL קובעים את ה-Duty Cycle של האות.



כשמונה מתחיל לספור, כל עוד התוכן שלו קטן מהערך באוגר CCAPnL במוצא יהיה '0'.

כאשר הערך שבאוגר CCAPnL גדול מהערך של המונה במוצא יהיה '1'.

כשמסתיימת הספירה האוגר CCAPnL נטען בערך של אוגר CCAPnH והמונה סופר מחדש, כך שמתקבל אות מחזורי במוצא עם Duty Cycle שניתן לבחירה.

בפרויקט אנו מייצרים אות PWM בהדקים P1.4-P1.7 עבור שליטה על מנועי הסרבו שאיתם אנו מזיזים את החיישן ultrasonic ושאר המנועים אשר מהווים צירים לזרוע שנמצאת על הרכב, שה"כ 3 מנועי סרבו לזרוע שלנו כאשר מנוע 1 מהווה ציר אופקי המנוע השני לציר אנכי והמנוע השלישי לסגירה/פתיחה של הזרוע בכדי להרים חפצים.

### 2.2.2 תזמון המיקרו-בקר בתדר קבוע:

מעגל תזמון של המיקרו-בקר מורכב מגביש של 11.0592 Mhz ושני קבלים שערכם 33pF. מעגל זה משמש ליצירת תדר השעון הנחוץ לעבודה תקינה של מיקרו בקר (קריאה, כתיבה, הפעלת רכיבים פנימיים, קריאה לפריפריה).

כדי לקבל תדר מדויק מחברים לגביש שני קבלים בעלי ערך של  $33\mu F$  במקביל.

ערך הקבלים נבחר לפי יצרן של המיקרו בקר. מעגל (מתנד בעל תדר 11.0592MHz ושני קבלים אשר מכניסים אותו לתדר תהודה) מהווה רשת משוב B של מתנד קולפיץ. המגבר של מתנד זה ממומש ע"י שער NOT פנימי שבמיקרו בקר.

השתמשנו בגביש ולא בסליל מפני שבמעגל דיגיטלי לא מומלץ להשתמש ברכיבים בעלי השראות וגם כן הגביש מדויק יותר.

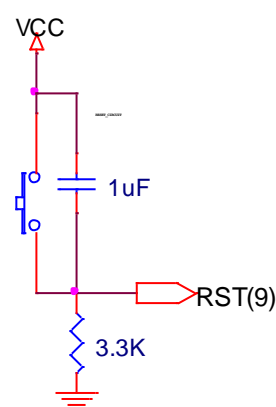
מעגל תזמון משמש ליצירת תדר שעון לשם הבטחת סנכרון (כלומר לשם שמירת תדירות קבועה ויציבה של עבודת הבקר) של כל פעולות המבוצעות ע"י בקר.

הדקים X1 ו- X2 הם קווי חיבור של גביש לתזמון הרכיב כאשר תדר העבודה הוא תדר הגביש.

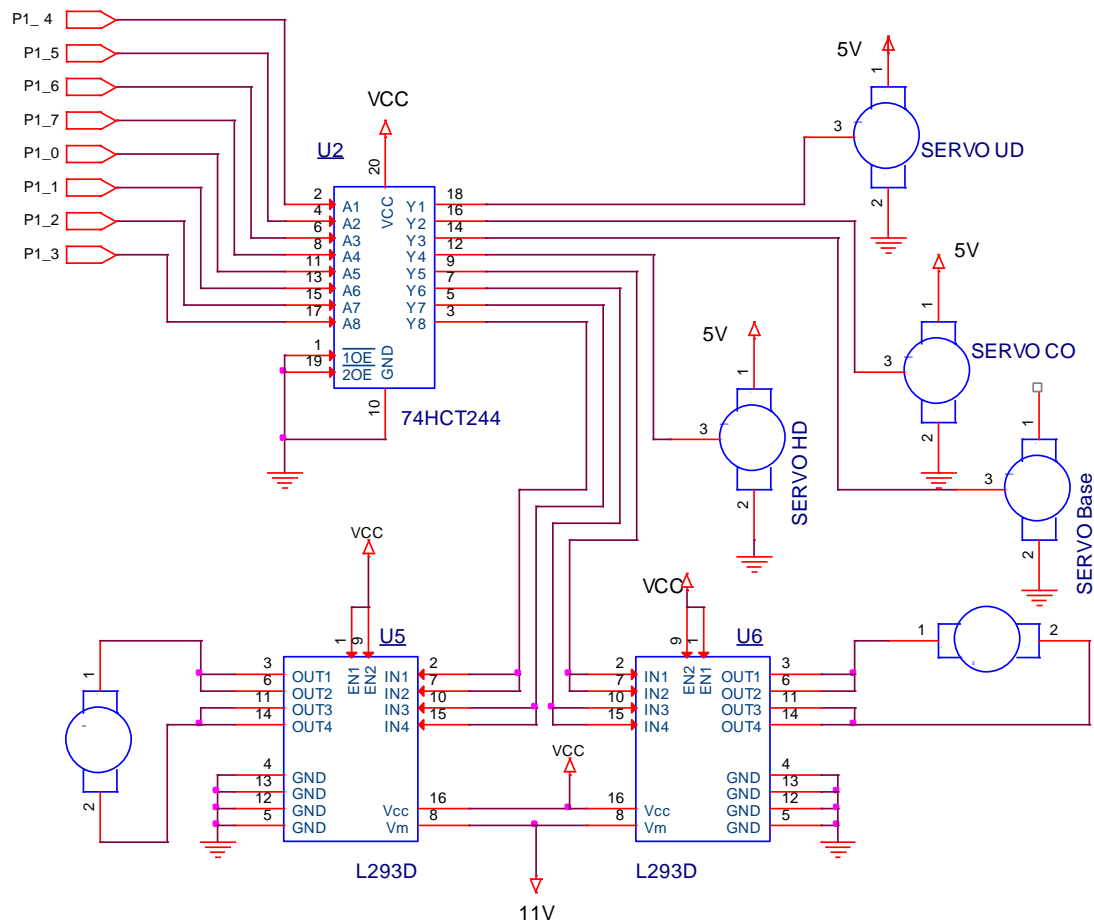
### 2.2.3 מעגל ריסט ( Reset Circuit ):

מעגל הריסט שלנו ממומש באמצעות לחצן N.O קבל 1uF ונגד 3.3k חיבור הלחצן הוא בתצורת pull-down, כאשר אנו נלחץ על הלחצן זרם יזורם דרך מסלול  $V_{CC}$  < קבל < נגד ויצור מפל מתח על הנגד שמפורש כ-HIGH בהדק ה RESET של הבקר, לפי דרישות היצרן מתח HIGH בהדק ה RESET למשך זמן של 2 זמני מכונה במקרה שלנו בערך 2us יגרמו לבקר לעשות RESET.

תפקיד הקבל הוא למנוע את בעית הריטוטים שנוצרת עקב הלחיצה המכאנית של האדם.



## 2.2.4 מעגל בקרת מנועים:



לרובוט מחוברים שני מנועים שמטרתם היא להזיז את הרובוט קדימה, ימינה שמאלה ואחורה.

בנוסף מחוברים 4 מנועי סרבו כאשר שלוש מתוכם לשם שליטה על הזרוע של הרכב והאחרון לשם הזזה של חישן המרחק.

### מנוע DC:

רכיב אלקטרו-מכאני המבצע המרה של אנרגיה חשמלית לאנרגיה מכאנית.

מנוע DC מבוסס על עקרון האלקטרומגנטיות, המאפשר יצירת שדה מגנטי על ידי העברת זרם חשמלי דרך סליל.



### אספקת מתח למנועים:

• לקחנו סוללת ליתיום 11.1V בעלת הספק של 1000mAh חיברנו אותה למטריצה והיא מוציאה לנו 10V מהנקודה הזו חיברנו מתח למנועי ה DC

• חיברנו בנוסף ממיר STEPDOWN שיורד את המתח למתח נמוך יותר שמתאים למנועי ה SERVO שהוא מתאים לפי דרישות היצרן ל (5v-7.4v) בחרנו לעבוד עם 6V ובכך הצלחנו להפעיל 2 מנועי DC ו 4 מנועי סרבו בעזרת סוללה אחת ברור שההספק של המנועי סרבו משתנה ותלוי האם הזרוע מרימה חפץ או האם אנו משתמשים בזרוע תוך כדי נסיעה.

### דרייבר דוחף זרם L293D:

תפקיד הדרייבר הוא לספק זרם למנועים, מכיוון שהזרם שהמיקרו בקר יכול לספק מהפורטים שלו לא מספיק גדול בשביל הפעלת המנועים.

כל ערוץ (מוצא של הדרייבר) יכול לספק עד 0.5A. אנו מחברים שני ערוצים לכל מנוע על מנת לאפשר זרם של עד 1A.

הדקים EN1 ו- EN2 מאפשרים את הערוצים OUT1,2 ו- OUT3,4 בהתאמה, כמוראה בטבלה:

Input	EN	Output
H	H	H
L	H	L
X	L	Z

H – רמה גבוהה

L – רמה נמוכה

X – לא רלוונטי

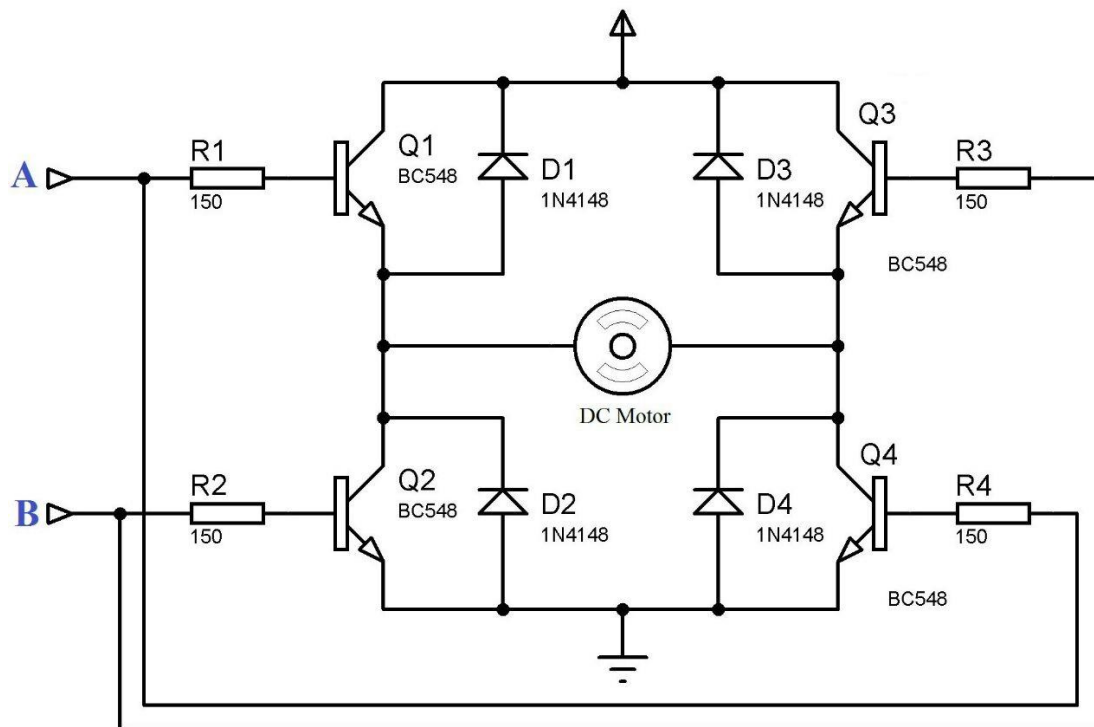
Z – עכבה גבוהה

הידק Vcc – מתח של 5V מסופק לכניסות הלוגיות כדי לצמצם בזבוז הספק.

הידק Vm – מתח המוצא של הרכיב - 10V לכל מנוע.

תפקיד נוסף של הדרייבר הוא להפעיל את המנוע בשני כיוונים ע"י ספק יחיד באמצעות מעגל H-Bridge:

ניתן לראות שעל ידי שינוי המתח בכניסות של הדרייבר אפשר למתג את המנוע בשני כיוונים שונים ומתח האספקה הוא עדיין  $V_m$ .



### דיודת הגנה

המתגים שבמעגל H-BRIDGE הם טרנזיסטורים ולכן יש צורך להוסיף דיודת הגנה שתהיה מחוברת במקביל למנוע (עומס השראי).

ב-L293D הדיודה בתוך הדרייבר ולכן הוסיפו את האות D לשם הדרייבר.

במצב מתמיד הזרם זורם דרך הסליל אשר מתנהג כקצר דרך הטרנזיסטור שנמצא ברוייה.

הבעיה היא כשמנתקים את V1 הטרנזיסטור נמצא בקיטעון ובגלל שאין קפיצות זרם בסליל הוא לא מספיק להתפרק.

במצב זה ללא דיודת הגנה הטרנזיסטור היה נשרף אך אם נחבר את דיודת ההגנה ניתן לראות שכשהטרנזיסטור נכנס לקיטעון מתח הסליל משנה את קוטביותו והוא מתפרק דרך הדיודה וכך הדיודה מגנה על הטרנזיסטור.

$$V_L = L \frac{di}{dt}$$

בזמן הטעינה של הסליל שינוי הזרם הוא חיובי אך בזמן הפריקה שינוי הזרם הוא שלילי ולכן המתח משנה קוטביות.

### בקרה על סיבוב המנועים :

על מנת לגרום למנועים להזיז את הרובוט לכיוון מסוים יש לשלוח נתונים לפורטים להם מחוברים הדרייברים של כל מנוע. הנתונים הדרושים לפעולה זו מתוארים בטבלה הבאה :

מנוע R		מנוע L		
P1.2	P1.3	P1.0	P1.1	כיוון
1	0	0	1	קדימה
0	1	1	0	אחורה
1	0	1	0	שמאלה
0	1	0	1	ימינה
0	0	0	0	עצירה

### רכיב חוצץ – 74HCT244 :

תפקידו הוא להפריד בין המיקרו בקר לבין מעגל בקרת המנועים.

74HCT244 הוא חוצץ ל- 8bit עם 3 מצבי מוצא, לרכיב יש שני מוצאים של 4bit הנשלטים ע"י הדקי OE.

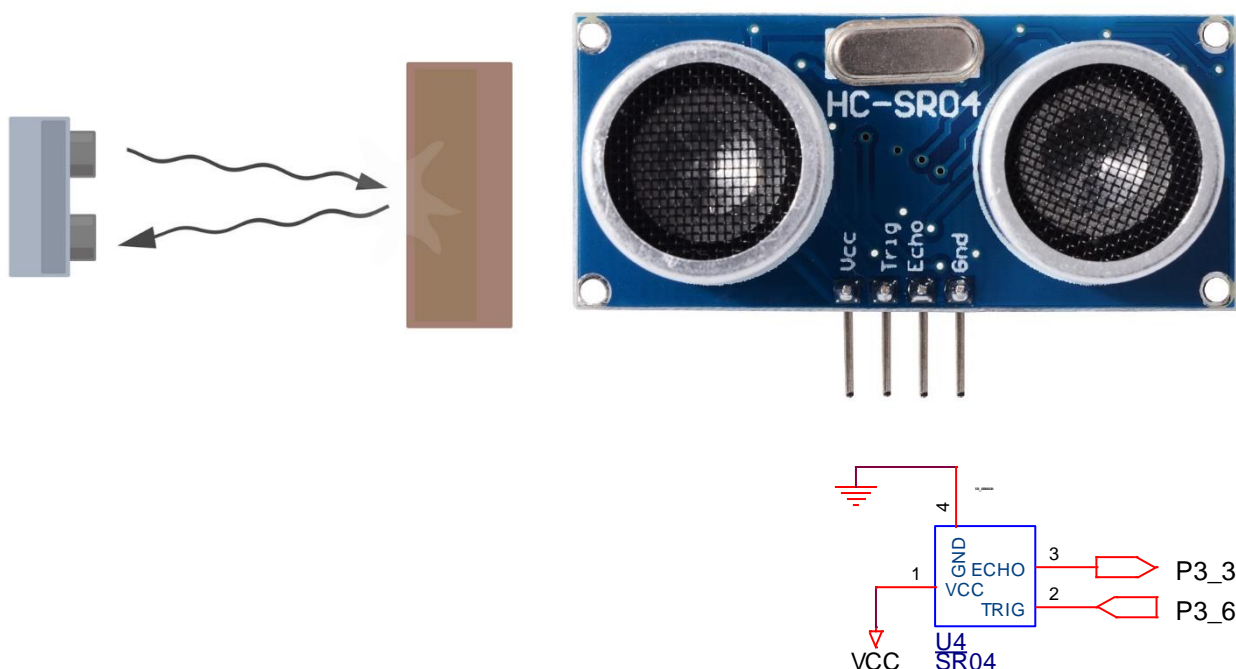
פעולת הרכיב מתוארת בטבלה הבאה :

Input		Output
<u>nOE'</u>	<u>nAn</u>	<u>nYn</u>
L	L	L
L	H	H
H	X	Z

אנו מחברים את הדקים OE' 1 ו- 2 לאדמה על מנת לאפשר את שני המוצאים, כדי שכל נתון שנשלח לחוצץ יעבור לדרייברים של המנועים.

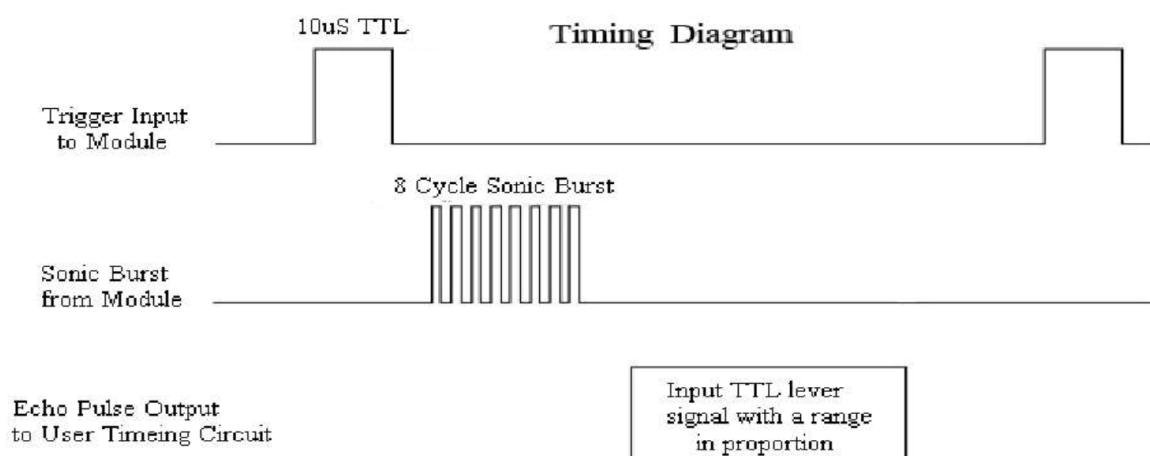


#### 2.2.4 חיישן מרחק מבוסס קול אולטראסוניד (HC-SR04):



חיישן זה כולל משדר ומקלט, החיישן משדר קול אולטראסוניד כאשר הוא מרגיש מתח HIGH ברגל ה- TRIGGER למשך זמן של לפחות 10 $\mu$ s, לאחר סיום השידור של הקול האולטראסוניד רגל ה ECHO עולה ל מתח HIGH עד שהמקלט מקבל את הקול חזרה ( זאת אומרת שהקול הספיק לחזור חזרה אל החיישן ) לפי הנתונים הללו ניתן להבין שהקול ביצע דרך כפולה, זאת אומרת הוא הספיק לפגוע באובייקט שנמצא מולו ולחזור חזרה אל המקלט ולכן את המרחק אשר נמודד בעזרת הTIMER בתוכנה נחלק ב2.

בעזרת הTIMER נמדוד את הזמן שלקח לקול האולטראסוניד לחזור אל החיישן ובכך יהיה ניתן



לחשב את המרחק מהאובייקט שממולו.

$$S = \frac{V * t[\mu s]}{2} = \frac{t[\mu s]}{2/V} = \frac{t[\mu s]}{K}$$

נשתמש בנוסחא ובטבלת יחידת המרת K הבאה :

ערך המרה K	V (מהירות)	טמפרטורה (C °)
60	331.5	0
59	337.5	10
58	343.5	20
57	349.5	30

בחרנו לעבוד עם ערך המרה K=58 כאשר טמפרטורת העבודה שלנו היא 20 מעלות.

בתוכנה הזמן שנמדד יחולק ב 58 ובכך יתקבל המרחק המדויק עבור טמפרטורת עבודה של 20 מעלות.

נשאלת השאלה כיצד אפשר לבצע מדידה מדויקת של מרחק בעזרת חיישן זה ?

על ידי הוספת חיישן טמפרטורה שבעזרתו נדע את טמפרטורת העבודה שלנו ובכך בעזרת התוכנה נוכל לבחור ערך המרה K מתאים ובכך נגיע למדידה מדויקת של המרחק.

```
1. void us_trig() { // trigger function for ultrasonic sensor
2.   trig=1;
3.   nop();nop();nop(); // delay of 14us
4.   trig=0;
5.   msec(50);
6. }
```

פונקציה למימוש ה TRIGGER שהיצרן דורש להוציא בהדק ה TRIGGER של החיישן יש לציין שהדרישה שלו היא פולס של 10us לפחות כאשר אנחנו נתנו לו 14us, בנוסף לכך יש לציין שלפי דרישות היצרן חייב להיות רווח של 50ms לפחות בין כל פולס ברגל ה TRIGGER.

```
1. void distance() interrupt 2 { // interrupt function (echo falls into int1 interrupt)
2.   time = TH1;
3.   time = time << 8; // shift left 8
4.   time = time | TL1; //
5.   dist = time / 58;
6.   TH1 = TL1 = 0;
7.   flag = 1;
8. }
```

פונקציה זו היא פונקצית-פסיקה והתוכנית תכנס אליה רק אחרי שיצרנו פולס לחיישן בעזרת הפונקציה us\_trig(), כאשר הדק 3.3 של הבקר ירד ל-"0" לוגי תתקבל פסיקת INT1 מסוג falling-edge, רגל 3.3 של הבקר מחוברת לרגל ה ECHO של החיישן, ECHO עולה ל-"1" לוגי לאחר מתן TRIGGER וסיום שידור הקול ויפול ל-"0" לוגי כשהקול יחזור אל החיישן. בזמן ש ECHO נמצא ב-"1" לוגי TIMER1 רץ ומודד את הזמן שהוא נמצא ב-"1" לוגי ומפסיק למדוד כאשר הוא נופל ל-"0" לוגי, מיד מתקבלת פסיקה ותוכנית ישר קופצת לפונקציה זו כדי לטפל במדידת הזמן. למשתנה dist נכנס המרחק שנמדד ביחידות CM.

חשוב לציין שאנו משתמשים בחיישן זה אך ורק כאשר אנו מפעילים את המצב האוטומטי כאשר הוא מתוכנת לצאת מחדר מרובע, ובכך הרובוט מחליט את כיוון הנסיעה לפי תוכנית שכתבנו ומשתמש בחיישן המרחק ככלי מפתח.

הטיימר שלנו עובד לפי זמן המכונה של הבקר שהינו בערך  $1\mu s$ , זאת אומרת הטיימר שלנו מונה זמן ב  $\mu s$ , מהירות הקול לפי הטבלה עבור טמפרטורת עבודה של 20 מעלות הינה  $343m/s$

אנו מעוניינים לקבל את המרחק מהחיישן ביחידות CM ולכן יש להמיר את המהירות ל  $CM/US$ , להלן הדרך:

$$v = 340m / s$$

$$v = 340 * 100cm = 34000cm / s$$

$$1\mu s = \frac{1sec}{1000000}$$

$$v(cm / \mu s) = \frac{34000}{1000000} = 0.034cm / \mu s$$

$$k = \frac{2}{v(cm / \mu s)} = \frac{2}{0.034} = 58.823$$



כמו שאפשר לראות בתמונות למעלה חיישן המרחק שלנו יושב על מנוע הסרבו, כאשר החיישן פונה קדימה (כיוון הרכב) הוא נמצא בזווית 0 של גיר המנוע, לפי נתוני היצרן ניתן לסובב את הגיר המנוע 120 מעלות, כאשר אנו צריכים במקרה שלנו רק 90 מעלות כדי לגרום לחיישן לדגום את המרחק משמאל הרכב, ובכך נדע איפה יש יציאה מהחדר המרובע שבנינו לו, כך הוא מקבל פיידבאק מהעולם החיצון ובעזרת הנתונים הללו הוא מקבל החלטות האם להמשיך ליסוע קדימה או לפנות שמאלה או האם הוא הגיע לפינה ועליו לבצע פניה ימינה ולהמשיך ליסוע לאורך הקיר.

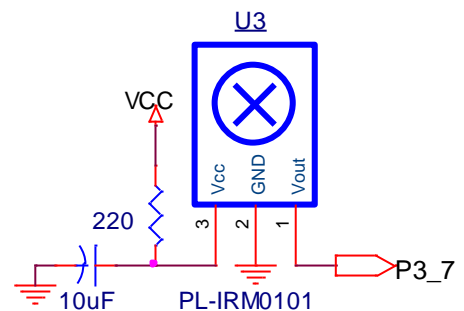
### 2.2.5 ממיר מסוג STEPDOWN:

אנו מחברים את הכניסה של ממיר זה לנקודה שבה יוצא מתח גם למנועי ה DC ומורידים את המתח לאיזור ה 6V בכדי לעבוד עם מנועי ה SERVO. היציאות של ממיר זה מתחברות להדקי ה VCC של ארבעת מנועי ה SERVO בפרויקט.



### 2.2.6 PL-IRM0101:

המעגל המתואר משמש לקליטת מידע מהמשדר (שלט). מעגל זה מכיל רכיב PL-IRM0101-3 הממיר מידע מ IR - לאות דיגיטלי בתחום תדרים. 36 - 46 KHz. בפרויקט זה אנו משדרים למקלט אות בתדר 38KHz. בנוסף המעגל מכיל קבל ונגד לפי דרישות היצרן.



את היציאה מהמשדר יש לחבר להדק במיקרו-בקר. בפרויקט זה הדק המוצא Vout של הרכיב מחובר להדק PORT 3.7 של המיקרו-בקר, על מנת לגלות את האות המשודר מהשלט-משדר ה IR ולשלוט במכשירים הביתיים.

המידע שמתקבל במודול IR שודר לפי פרוטוקל IR (80 – RECS – מוסבר בהמשך בפרק פרוטוקלים).

## 2.2.8 מנוע SERVO:

מנוע ה SERVO כולל בתוכו מנוע DC אך עם מערכת בקרה על זווית סיבוב הגיר של המנוע כאשר הוא מוגבל לזווית של 120 מעלות מקסימום.

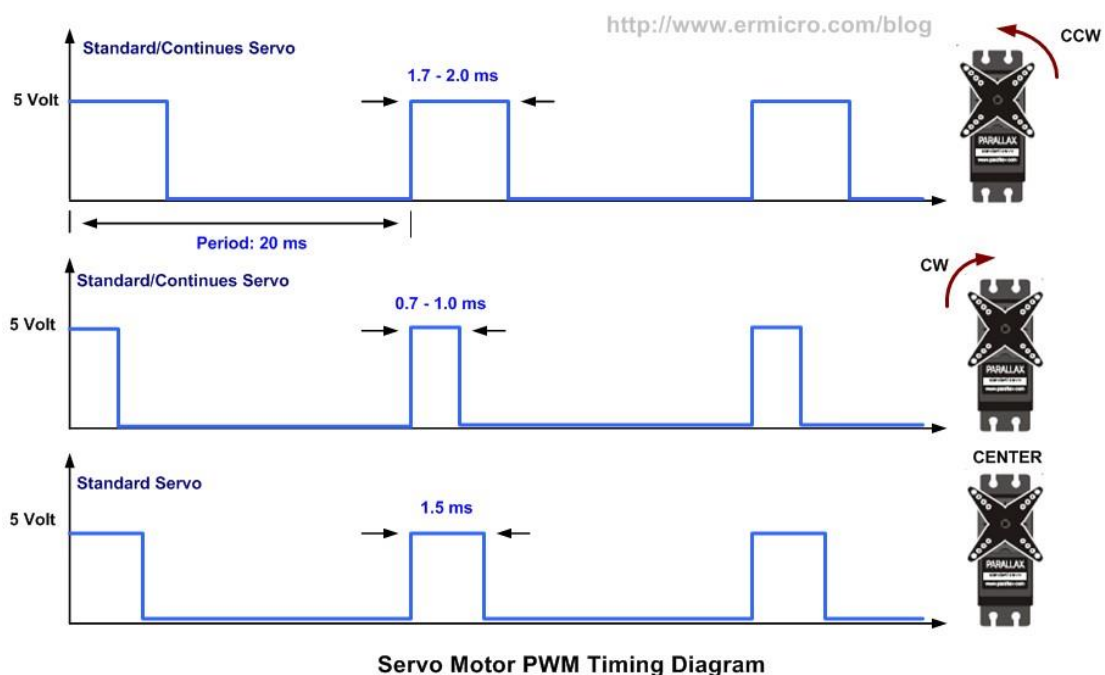
מנוע זה כולל 3 הדקים בשונה ממנוע ה DC הרגיל להלן ההדקים והסבר עליהם:

• GND – הדק האדמה.

• VCC – הדק המתח VCC עבור המנוע.

• SIGNAL – להדק זה אנו שולחים את האותות PWM על מנת לשלוט על הזווית שלו.

להלן דוגמא לאות ה PWM שהיצרן דורש בכדי לשלוט על זווית הגיר של מנוע ה SERVO :



כפי שאפשר לראות אנו שולטים על זווית גיר המנוע על ידי שינוי של זמן ה D.C של האות כאשר הוא נע בין 1ms-2ms, בתחום הזה אפשר לשלוט על זווית גיר המנוע בין 0-120 מעלות במנועים בפרויקט שלנו, יש מנועי סרבו שמציעים טווח רחב יותר של זוויות או טווחים קטנים יותר, D.C של 2ms יזיז את גיר המנוע לזווית המקסימלית 120 מעלות.

הבקר שלנו P89V51RD2 מסוגל להוציא PWM מ 5 הדקים P1.3-P1.7 כאשר אנו השתמשנו רק ב 4, P1.4-P1.7, על מנת ליצור אות זה אנו צריכים להשתמש במונה ה PCA של הבקר Programmable Counter Array ויש להגדיר את האוגרים שלו ואת תצורת המנייה שלו, להלן הסבר שכולל חלקים מהתוכנית הראשית כיצד יצרנו את האות PWM המתאים עבור המנועים.

אוגר בקרת ה- PCA :

**Table 36: CMOD - PCA counter mode register (address D9H) bit description**

Bit	Symbol	Description
7	CIDL	Counter Idle Control: CIDL = 0 programs the PCA Counter to continue functioning during Idle Mode. CIDL = 1 programs it to be gated off during idle.
6	WDTE	Watchdog Timer Enable: WDTE = 0 disables Watchdog timer function on module 4. WDTE = 1 enables it.
5 to 3	-	Reserved for future use. Should be set to '0' by user programs.
2 to 1	CPS1, CPS0	PCA Count Pulse Select (see Table 37 below).
0	ECF	PCA Enable Counter Overflow Interrupt: ECF = 1 enables CF bit in CCON to generate an interrupt. ECF = 0 disables that function.

כאשר אותנו מעניין רק תצורת המנייה של ה-PCA נשנה את ה סיביות CPS1-CPS0 בכדי לבחור פולס מנייה מתאים מתוך ארבעת האופציות שהיצרן מציעה להלן האפשרויות והפקודה בכדי לבחור את המצב:

CMOD=4;

**Table 37: CMOD - PCA counter mode register (address D9H) count pulse select**

CPS1	CPS0	Select PCA input
0	0	0 Internal clock, $f_{osc} / 6$
0	1	1 Internal clock, $f_{osc} / 6$
1	0	2 Timer 0 overflow
1	1	3 External clock at ECI/P1.2 pin (max rate = $f_{osc} / 4$ )

בחרנו להשתמש באופציה ה שלישית שהיא גלישת טיימר 0, בחרנו להשתמש באופציה זו בגלל שאפשר לקבוע את טיימר אפס בתצורת מנייה של 8bit טעינה אוטומטית, הטיימר מונה מ ערך מסויים שבהמשך נחשב וכאשר הוא מגיע לערך המקסימלי עבור 8bit שזה FF, אוטומטית ברקע התוכנית הראשית אוגר ה TL נטען עם הערך של אוגר ה TH של מונה זה, זאת אומרת שאפשר להתחיל את המנייה כל הזמן מחדש ברקע לתוכנית הראשית ולכן בחרנו באופציה זו.

בכדי להגדיר את הטיימר בתצורה זו כתבנו את הפקודה הבאה: TMOD=0x12 פקודה זו תגדיר את טיימר 0 בתצורת 8bit טעינה אוטומטית ובנוסף תגדיר את טיימר 1 בתצורת 16 ביט ללא טעינה אוטומטית, השתמשנו בטיימר 1 עבור קליטת נתונים ממקלט ה IR.

#### כיצד ה PCA עובד?

PCA יכול להיות מונה/ טיימר, ואנו בחרנו לעבוד איתו לשם יצירת PWM כאשר אנו עובדים איתו לשם PWM הוא הופך ל 8bit ואת זה אנו מבצעים על ידי אפשר ה PWM עבור הדקי הבקר שרשומים מעלה, להלן אוגר הבקרה אשר אחרי לאפשר ה PWM עבור הדקים אלו (P1.3-P1.7) כאשר יש אוגר בקרה כזה לכל הדק בנפרד.

להלן האוגר ואיזה ערך נתנו לו בכדי לאפשר לו להוציא אות PWM, בנוסף לכך כחלק מתהליך יצירת האות אפשרנו את פעולת המשווה בין אוגר ה PCA לבין אוגר ה CCAPL שהוא אוגר נפרד לכל פורט.

**Table 40: CCAPMn - PCA modules compare/capture register (address CCAPM0 0DAH, CCAPM1 0DBH, CCAPM2 0DCH, CCAPM3 0DDH, CCAPM4 0DEH) bit alloc.**

Not bit addressable; Reset value: 00H

Bit	7	6	5	4	3	2	1	0
Symbol	-	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn

1. CCAPM1=0x42;
2. CCAPM2=0x42;
3. CCAPM3=0x42;
4. CCAPM4=0x42;

לאחר רצף פקודות אלו אפשרנו את פעולת ה PWM עבור ההדקים P1.4-P1.7 ואת פעולת המשווה בין אוגר ה CCAPL שהוא נפרד לכל הדק לבין אוגר ה CL שהוא אוגר ה PCA שהוא משותף לכל ההדקים.

לאחר בחירה של פולס מנייה מתאים בעזרת CMOD, שבמקרה שלנו כל גלישה של טיימר 0 מונה ה PCA יתקדם ב1 כאשר הערך המקסימלי שאליו הוא יגיעה בסוף המנייה הוא FF.

נשאלת השאלה כיצד אנו ניצור את PWM בתדר 50hz לפי דרישות היצרן עבור מנועי הסרבו ולהלן החישובים שעשינו בכדי לתת ערכים מתאימים לאוגרי הטיימר 0 ואוגרי ה CCAPH של כל הדק (CCAPH נטען ל CCAPL בכל גלישת אוגר ה CL, זה קורה בכדי שלא יהיו הפרעות, בעצם המתכנת מעדכן את CCAPH וCCAPL נשאר קבוע ומתעדכן אך ורק בכל גלישה של אוגר ה CL.)

תחילה נחשב את זמן המחזור  $T$ :  $T = \frac{1}{50\text{hz}} = 20\text{ms}$  לאחר מכן אנו מסתכלים על טיימר 0 ומבינים

שהגדרנו אותו בתצורת 8bit, זאת אומרת שהערך המקסימלי שהוא יוכל למנות הוא 255 ולכן נבצע

את הפעולה הבאה:  $78\text{us} = \frac{20\text{ms}}{\text{TIMER0}(TL0MAX)}$  כאשר  $TL0MAX = 255$ , זאת אומרת שכל

78us צריכה להיות גלישת טיימר 0 הטיימר שלנו עובד בקצב הזמן מכונה של הבקר שהינו 1.085us,

זאת אומרת שהטיימר שלנו ימנה 255 פולסי זמן מכונה של הבקר זאת אומרת  $255 * 1.085\text{us}$  שיוצא

276us כאשר אנו צריכים לקבל גלישה רק אחרי 78us ולכן אנו נצטרך להתחיל את אוגרי הטיימר

בערך שונה מ 0 בכדי לקבל את הזמן הזה ולכן:  $71(dec) = 49(hex)$  כאשר  $\frac{78\text{us}}{1\text{tmachine}}$

$1.085\text{us}$ ,  $t_{machine} = 1.085\text{us}$ , אז נבצע פעולת חיסור בין הערך המקסימלי של האוגר לבין הערך 49(HEX)

ולכן  $FF(hex) - 49(hex) = B8$ , זאת אומרת את הערך B8 נצטרך לתת לאוגרי ה TH0 ו TH1

בתחילת התוכנית כדי לקבל גלישת טיימר כל 78us.

אם כל 78us מתבצעת קידום של אוגר ה CL ב 1 אפשר לומר:

$78\text{us} * CL(MAX) = 19.89\text{ms} \rightarrow 20\text{ms}$  כאשר  $CL(MAX) = 255$  ואנו רואים שבאמת נוצר

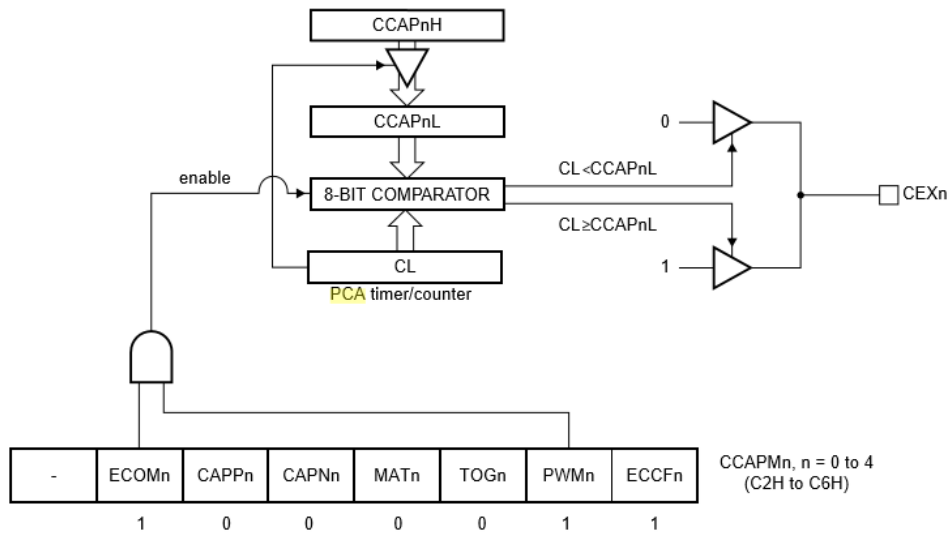
אות בזמן מחזור של 20ms שהינו תדר של 50hz.

להלן קוד בתחילת התוכנית כדי לתת לאוגרי ה טיימר ערכים מתאימים כפי שחישבנו :

1. TH0=0xB8;
2. TL0=0xB8;

3. TR0=1;

TR0=1 בכדי להפעיל את פעולת הטיימר של טיימר 0, להלן איור שמתאר את שינוי הלוגי של כל ההדקים שאפשרו להם פעולת PWM כתלות באוגר ה PCA - CL.



002aaa541

Fig 25. PCA PWM mode.

\*חשוב לציין שאיור זה מתאים עבור כל הדק שאליו אנו עובדים עם PWM.

אוגר ה CL מתחיל את המנייה שלו מערך 0 עד ערך מקסימלי של FF ובכדי לשנות את ה D.C של אותו הדק נצטרך לשנות את ערך אוגר ה CCAPL כאשר לכל הדק יש אוגר כזה נפרד, את השינוי אנו לא עושים ישירות, זאת אומרת שאנו משנים את CCAPH ו CCAPL נטענן לערך CCAPH אך ורק אחרי גלישת אוגר ה CL (כאשר CL הגיעה ל FF ואז התבצעה עוד מנייה זאת אומרת הוא יתאפס ל 0)

כעת נסביר איך מתבצעת שינוי ב D.C של האות ה PWM בעזרת שינוי של אוגר ה CCAPL.

כפי שניתן לראות באיור ההדק של הפורט שאיתו אנו עובדים ב PWM יהיה ב רמה לוגיית נמוכה – '0' כאשר אוגר ה CL קטן בערכו מאוגר ה CCAPL, ו יהיה ברמה לוגית גבוהה כאשר CL יהיה גדול בערכו מאוגר ה CCAPL. כעת נחשב איזה ערך צריך לתת לאוגר ה CCAPL בכדי לקבל D.C של 1ms עבור האות שהוא כולו 20ms – 50hz כפי שהיצרן דורש.

אנו יודעים שרק אחרי גלישת טיימר 0 שזה כל 78us אוגר ה CL יתקדם ב 1. ולכן בכדי לקבל רמה לוגית גבוהה – '1' לוגי במוצא ההדק למשך זמן של 1ms עבור האות שכולו 20ms לפי נתוני היצרן אות כזה ברגל ה SIGNAL של המנוע תזיז את גיר המנוע לזווית ה 0 כפי שניתן לראות בגרף שצורף בתחילת ההסבר על מנועי הסרבו, להלן החישוב :



$$(CL(max) - X) * 78us = 1ms$$

$$CL(max) = 255$$

$$20m - 78uX = 1ms$$

$$78uX = 19m$$

$$X = 243.58 \rightarrow 244$$

לכן נתנו לאוגר CCAPH את הערך 244 בכדי לקבל D.C של 1ms בתדר של 50hz בתחילת התוכנית כמובן שניתן לשנות את אוגר ה CCAPH בכדי לשלוח על הזווית כעט נחשב את הערך שצריך לתת לאוגר זה בכדי להגיע לזווית המקסימלית של המנוע שהינה 120 מעלות, ה D.C הדרוש עבור זווית זו הוא 2ms להלן החישוב :

$$(CL(max) - X) * 78us = 2ms$$

$$CL(max) = 255$$

$$20m - 78uX = 2ms$$

$$78uX = 18ms$$

$$x = \frac{18m}{78u} = 230$$

לאחר החישובים הללו אנו מבינים שערך של אוגרי ה CCAPL עבור כל הדק שהגדרנו לשימוש מנועי סרבו ינוע בין 230-244 כאשר 244 היא הזווית ה 0 ו 230 היא הזווית המקסימלית 120 מעלות.

לאחר כל ההגדרות והחישובים שביצענו בכדי ליצור את האות לפי דרישות היצרן נשאר להפעיל את פעולת המונה של ה PCA להלן האוגר CCON שהינו אוגר הבקרה של מונה זה

**Table 38: CCON - PCA counter control register (address 0D8H) bit allocation**

Bit addressable; Reset value: 00H

Bit	7	6	5	4	3	2	1	0
Symbol	CF	CR	-	CCF4	CCF3	CCF2	CCF1	CCF0

נתנו לאוגר זה את הערך CCON=0x40 בכדי להפעיל את פעולת המנייה על ידי שינוי סיבית ה CR כאשר סיבית זו '1' לוגי אז מונה ה PCA יתחיל לעבוד.

**פרק 3:**  
**פרוטוקולים**

## פרוטוקול IR-RECS-80:

- אור אינפרא אדום (תת אדום) הוא קרינה אלקטרומגנטית באורך גל הגדול מזה הנראה בעין.
- אורכי הגל נעים בטווח של 750 ננומטר עד 30,000 ננומטר.
- שימושים בקרינה זו:
  - מצלמות הרואות ביום ובלילה.
  - תקשורת בין מכשירים.
  - ועד לחימום.
- בפרויקט שלנו אנו משתמשים בפרוטוקול זה על מנת ליצור קשר בין השלט לבין הרכב בכדי שיוכל לבצע את תנועותיו.
- לבקר מחובר חיישן IR אשר קולט את הנתונים מהשלט וכך יודע לבצע את פעולות הרכב.

### להלן התשדורת פרוטוקול RECS – 80 :



#### מבנה השידור

מבנה השידור של המידע יתבצע על ידי שליחה של 32 סיביות אשר מציינות כתובת או מידע. התשדורת מתחלקת ל-4 בתים (8bit), כאשר:

- בית ראשון מכיל את כתובת השלט.
- בית שני מכיל את כתובת השלט לאחר פעולת NOT.
- בית שלישי מכיל את המידע (לחצן שנלחץ).
- בית רביעי מכיל את המידע לאחר פעולת NOT.

#### תשדורת המידע נשלחת מסיבית ה-LSB אל סיבית ה-MSB.

#### זמני שידור סיביות '0' לוגי ו-'1' לוגי.

סיבית מידע '0' לוגי או '1' לוגי מתקבלת על ידי משך זמן המחזור של השידור כאשר:

- סיבית המידע נמצאת למשך 1T במצב '0' לוגי ואחר כך במצב '1' לוגי למשך 1T מוגדרת כסיבית המייצגת '0' לוגי.
- סיבית המידע נמצאת למשך 1T במצב '0' לוגי ואחר כך במצב '1' לוגי למשך 3T מוגדרת כסיבית המייצגת '1' לוגי.

נשאלת השאלה מדוע השלט משדר את המידע וכתובת השלט פעמיים, פעם אחת רגיל ופעם אחת אחרי פעולת NOT? דבר זה קורה בכדי לבדוק שהתקשורת אכן אמינה ולא לקלוט שגיאות ובכך לקלוט נתונים שלעולם לא נשלחו באמת, באמצעות הבקר אנחנו עושים פעולת

NOT למידע שנשלח כ NOT ואז אנחנו מקבלים את המידע ומשווים את המידע המקורי למידע שהיתקבל לאחר פעולת NOT על המידע שהיה NOT המידע.

זהו תקין של הזמנים מתקבל לפי הטבלה הזו אפשר לראות התאמה בין הבדיקה עצמה בשורות הקוד שהוספתי למטה לבין הנתונים בטבלה.

להלן טבלה מרכזת את תזמוני התשדורת

The chosen time units (with 4T) for RECS80-32 (in micro seconds):

Time Unit	Min	Typ	Max
1T	250	565	1130
3T	1130	1695	1978
4T	1978	2260	3390
8T	3390	4520	6780
16T	6780	9040	12000

Worst Case Tolerance 12%

להלן קטע קוד מתוך הפונקציה bit remote() אשר מזהה את 16T שהוא חלק מ- START BIT:

```

1. TL1=TH1=0;
2. TR1=1;
3. while(!bit_IR); //bit_IR up
4.   TR1=0; //stop timer
5. time_units =TH1;
6. time_units =time_units <<8; //shift left 8
7.   time_units =time_units |TL1; //
8.   if(time_units <11060 && time_units >6248) // check 16T
9. {

```

לאחר זיהוי תקין של פרק זמן של 16T ברמה לוגית נמוכה התוכנית ממשיכה לבדיקת זמן של 8T ברמה לוגית גבוהה כאשר היא מהווה את החלק השני של ה START BIT.

```

1. TL1=TH1=0;
2. TR1=1;
3. while(bit_IR); //bit_IR down
4.   TR1=0; //stop timer
5. time_units =TH1;
6. time_units =time_units <<8;
7.   time_units =time_units |TL1;
8.   if(time_units <6248 && time_units >3124)
9. {

```

לאחר זיהוי תקין של פרק זמן של 8T ברמה לוגית גבוהה התוכנית סיימה לפענח את ה-START BIT וממשיכה לפענוח ה-ADDRESS של השלט כאשר הסיבית '0' ו'1' מוגדרות כך:

#### זמני שידור סיביות '0' לוגי ו-'1' לוגי.

- סיבית מידע '0' לוגי או '1' לוגי מתקבלת על ידי משך זמן המחזור של השידור כאשר:
  - סיבית המידע נמצאת למשך 1T במצב '0' לוגי ואחר כך במצב '1' לוגי למשך 1T מוגדרת כסיבית המייצגת '0' לוגי.
  - סיבית המידע נמצאת למשך 1T במצב '0' לוגי ואחר כך במצב '1' לוגי למשך 3T מוגדרת כסיבית המייצגת '1' לוגי.

```
1. for(len_arr=0;len_arr<4;len_arr++)
2. {
3.     num=0;
4.     for(num_bit=0;num_bit<8;num_bit++)
5.         {//==== 1T =====//
6.             while(!bit_IR); //bit_IR up
7.
8.             //==== 3T =====//
9.             TL1=TH1=0;
10.            TR1=1;
11.            while(bit_IR); //bit_IR down
12.            TR1=0; //stop timer
13.            time_units =TH1;
14.            time_units =time_units <<8;
15.            time_units =time_units |TL1;
16.            if(time_units <1823 && time_units >1041)//3t
17.                num|=1<<num_bit;
18.        } //for8
19.        arr_remote[len_arr]=num;
20.    }
```

לולאה זו רצה 4 פעמים כאשר הפעם הריאשונה היא מפענחת את ADDRESS לאחר מכן היא מפענחת את NOT ADDRESS ולאחר מכן DATA ואחרי זה את NOT DATA. לאחר שהיא מסיימת לקבל את ארבעת הבתים הללו היא בודקת האם התקשורת אמינה על ידי פעולת NOT על כתובת שנשלחה כ NOT ועל המידע שנשלח כ NOT בכדי לבצע השוואה בין המידע המקורי ולבין המידע שנשלח כ NOT וכך רואים אם התקשורת אכן תקינה, להלן הקטע קוד שאחרי לך:

```
1. if(arr_remote[0]==~arr_remote[1]) //address && !address
2.     if(arr_remote[2]==~arr_remote[3])
3.         return 1;
4. return 0;
```

אם ADDRESS שווה ל NOT של NOT ADDRESS זאת אומרת אם ADDRESS שווה ל ADDRESS אז אכן יש תקשורת אמינה לאחר מכן הוא מבצע את אותה פעולה על DATA ו

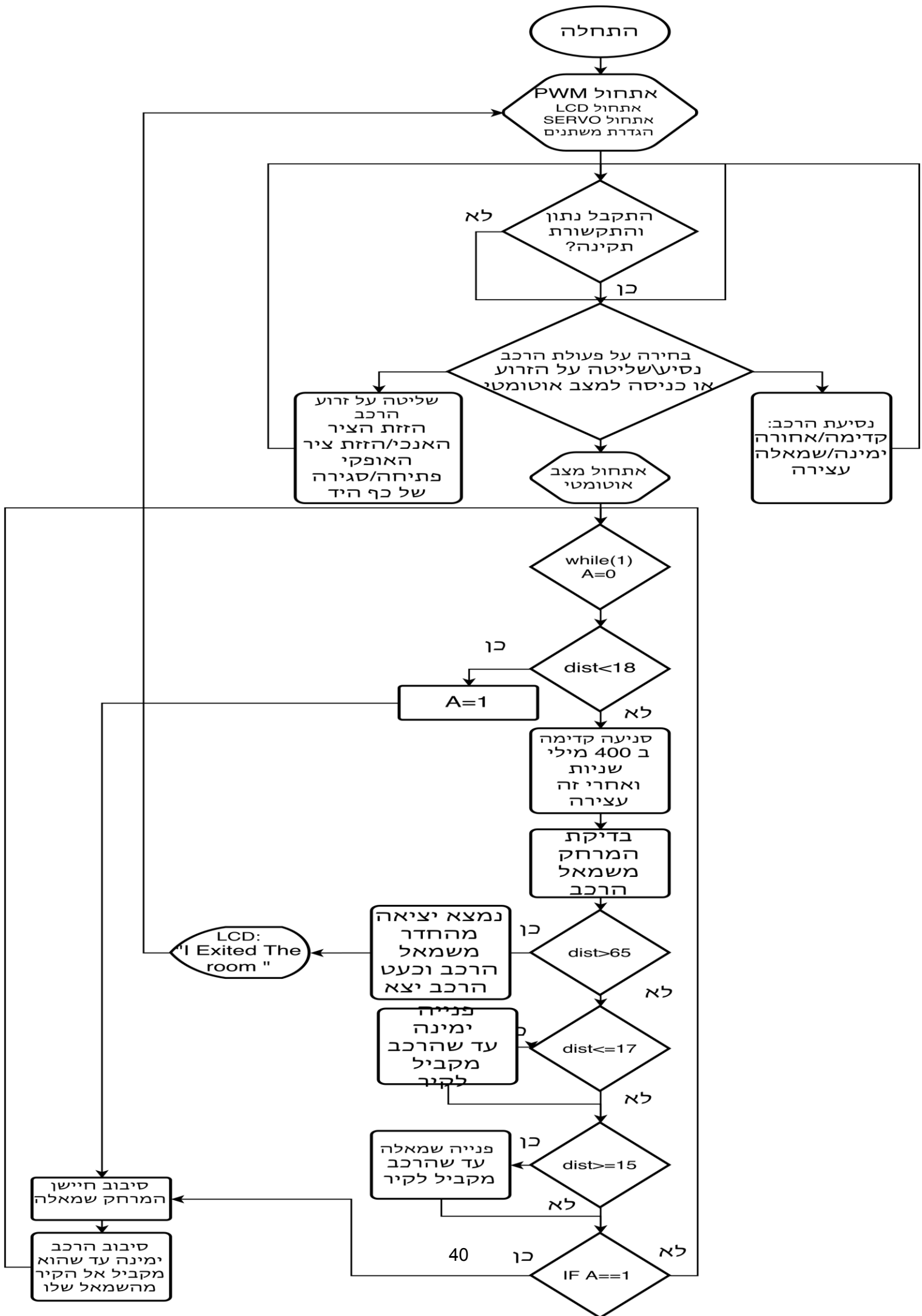
NOT DATA בכדי לבדוק שוב כדי לוודאות שהתקשורת אכן אמינה. אם הבדיקות היו תקינות הפונקציה תחזיר 1 בנוסף לכך בתוך המערך arr\_remote[] נמצאים הנתונים של ADDRESS NOT DATA DATA NOT ADDRESS בסדר עולה מ 0 עד 3 זאת אומרת שבמקום ה 2 של המערך arr\_remote נמצא ה DATA שלנו.

## פרק 4:

## תוכנית

4.1 תכנית בשפת C

4.1.1 תרשים זרימה:





```
1. // for P89V51RD2
2. #ifndef __general_h__
3. #define __general_h__
4.
5. typedef unsigned char byte;
6. typedef unsigned int uint;
7.
8. #define ACK 0
9. #define NACK 1
10.     #define TRUE 1
11.     #define FALSE 0
12.     //===== IR CODES DEFINED =====
13.     #define _0 22
14.     #define _1 12
15.     #define _2 24
16.     #define _3 94
17.     #define _4 8
18.     #define _5 28
19.     #define _6 90
20.     #define _7 66
21.     #define _8 82
22.     #define _9 74
23.     #define _plus 64
24.     #define _minus 25
25.     #define _play 21
26.     #define _right 9
27.     #define _left 7
28.     #define _test 68
29.     #define _spin 67
30.     #define _menu 71
31.     #define _shutdown 69
32.     //===== end ir codes defin
33.     e=====
34.
35.     void msec(uint ms);
36.     void nop(void);
37.
38.
39.     void nop(void){ } // delay 5usec
40.
41.
42.     void msec(uint ms)
43.     {
```

```

44.      uint n;
45.      uint i;
46.      //ms=ms*4.11;
47.
48.      for (n=0; n<ms; n++)
49.      {
50.          for (i=0; i<112; i++);
51.      }
52.  }
53.
54.  void usec(uint us)
55.  {
56.      while(us--) nop();
57.  }
58.  #endif

```

Lcdp2.h

```

1.  //=====
    =====
2.  //                                "lcdp2.h"
3.  //=====
    =====
4.  #ifndef __lcd_4_h__
5.  #define __lcd_4_h__
6.
7.  #define RS_lcd      P2_0
8.  #define RW_lcd      P2_1
9.  #define E_lcd       P2_2
10.     #define DATA_lcd  P2
11.
12.     //=====
        =====
13.     void set_lcd1(bit char_command,byte lcd_d)
14.     {
15.         RW_lcd = 0 ;
16.         E_lcd = 0 ;
17.         RS_lcd = char_command ;
18.         DATA_lcd &= 0x87;
19.         DATA_lcd |= ((lcd_d &0xf0 )>>1);
20.         E_lcd = 1 ;
21.         nop();
22.         E_lcd = 0 ;
23.         msec(3);
24.     }
25.
26.     void set_lcd(bit char_command,byte lcd_d)
27.     {
28.         RW_lcd = 0 ;

```

```

29.      E_lcd = 1 ;
30.      RS_lcd =char_command ;
31.      DATA_lcd &= 0x87;
32.      DATA_lcd |= ((lcd_d &0xf0 )>>1);//write 4bit High

33.      E_lcd = 1 ;
34.      nop();
35.      E_lcd = 0 ;
36.
37.
38.      RW_lcd = 0 ;
39.      E_lcd = 1 ;
40.      RS_lcd = char_command ;
41.      DATA_lcd &= 0x87;
42.      DATA_lcd |= ((lcd_d &0x0f )<<3);//write 4bit Low

43.      E_lcd = 1 ;
44.      nop();
45.      E_lcd = 0 ;
46.      msec(3);
47.
48.
49.  }
50.
51.
52.  //=====
  =====
53.  /*
54.    I/D -
      when high, cursor moves right DDRAM addr. inc. by 1
55.    -
      when low,  cursor moves left  DDRAM addr. dec. by 1
56.    SH  - Shift of entire display
57.    D   -
      Display on/off (high = on; low = off; Data stored)
58.    C   -
      Cursor  on/off (high = on; low = off; ID  stored)
59.    B   -
      Cursor blink on/off (high = blink on; low = blink off)
60.    S/C - 1 - display; 0 - cursor;
61.    R/L - 1 - right;   0 - left;
62.    DL  - Interface data length ( 1 - 8 bit; 0 -
      4 bit;)
63.    N   - Display line number (1 - two lines; 0 -
      one line;)
64.    F   - Display font type (1 - 5x11; 0 - 5x8;)
65.    */
66.  //=====
  =====

```

```

67.     void init_lcd()
68.     {
69.
70.         msec(50); // Recovery time after power on
71.
72.         set_lcd1(0,0x30);//  0 0 1 DL N F 0 0    -
        FUNCTION SET - 8bit interface
73.         set_lcd1(0,0x30);//  0 0 1 DL N F 0 0    -
        FUNCTION SET - 8bit interface
74.         set_lcd1(0,0x30);//  0 0 1 DL N F 0 0    -
        FUNCTION SET - 8bit interface
75.         set_lcd1(0,0x2C);//  0 0 1 DL N F 0 0    -
        FUNCTION SET - 4bit interface
76.
77.         set_lcd(0,0x2C);//  0 0 1 DL N F 0 0    -
        FUNCTION SET - 4bit interface
78.
79.         set_lcd(0,0x0c);//  0 0 0 0 1 D C B      -
        DISPLAY ON/OFF CONTROL
80.
81.         set_lcd(0,0x06);//  0 0 0 0 0 1 I/D SH   -
        ENTRY MODE SET
82.
83.         set_lcd(0,0x14);//  0 0 0 1 S/C R/L 0 0   -
        CURSOR OR DISPLAY SHIFT
84.
85.         set_lcd(0,0x01);//  0 0 0 0 0 0 0 1     -
        DISPLAY CLEAR
86.
87.     }//init_lcd()
88.
89.     void lcd_puts(byte location,const byte *str)
90.     {
91.         set_lcd(0,location);
92.         while(*str) set_lcd(1,*str++);
93.     } //lcd_put_str(byte location,const byte *str)
94.     void lcd_clr() {
95.         set_lcd(0,0x01);
96.     }
97.     #endif
98.
99.
100.    //=====
    =====
101.    //                               End Of File "lcd_p2.h"
102.    //=====
    =====

```

```

1. //=====
   =====
2. //                                     "REG89C51.h"
3. //=====
   =====
4.
5. #ifndef __REG_89C51_H__
6. #define __REG_89C51_H__
7.
8. /* interrupts vectors */
9. #define EXT0_INTERRUPT    0
10.     #define TIMER0_INTERRUPT  1
11.     #define EXT1_INTERRUPT    2
12.     #define TIMER1_INTERRUPT  3
13.     #define SERIAL0_INTERRUPT 4
14.     #define TIMER2_INTERRUPT  5
15.     #define PCA_INTERRUPT     6
16.     #define SERIAL1_INTERRUPT 7
17.     #define EXT2_INTERRUPT     8
18.     #define EXT3_INTERRUPT     9
19.     #define EXT4_INTERRUPT    10
20.     #define EXT5_INTERRUPT    11
21.     #define WDT_INTERRUPT     12
22.
23.     /* BYTE Register */
24.     sfr CKCON    = 0x8F;
25.     sfr B        = 0xF0;
26.     sfr SP       = 0x81;
27.     sfr DPL      = 0x82;
28.     sfr DPH      = 0x83;
29.     sfr TMOD     = 0x89;
30.     sfr TL0      = 0x8A;
31.     sfr TL1      = 0x8B;
32.     sfr TH0      = 0x8C;
33.     sfr TH1      = 0x8D;
34.     sfr IP       = 0xB8;
35.     sfr ICON     = 0xFF;
36.     sfr CKSEL    = 0x85;
37.     sfr OSCCON   = 0x86;
38.     sfr FCON     = 0xD1;
39.     sfr EECON    = 0xD2;
40.     sfr EETIM    = 0xD3;
41.

```

```

42.
43.     /* PORT 0 */
44.     sfr  P0    = 0x80;
45.     sbit P0_7  = P0^7;
46.     sbit P0_6  = P0^6;
47.     sbit P0_5  = P0^5;
48.     sbit P0_4  = P0^4;
49.     sbit P0_3  = P0^3;
50.     sbit P0_2  = P0^2;
51.     sbit P0_1  = P0^1;
52.     sbit P0_0  = P0^0;
53.
54.     /* PORT 1 */
55.     sfr  P1    = 0x90;
56.     sbit P1_7  = P1^7;
57.     sbit P1_6  = P1^6;
58.     sbit P1_5  = P1^5;
59.     sbit P1_4  = P1^4;
60.     sbit P1_3  = P1^3;
61.     sbit P1_2  = P1^2;
62.     sbit P1_1  = P1^1;
63.     sbit P1_0  = P1^0;
64.
65.     sbit CEX4  = P1^7;
66.     sbit CEX3  = P1^6;
67.     sbit CEX2  = P1^5;
68.     sbit CEX1  = P1^4;
69.     sbit CEX0  = P1^3;
70.     sbit ECI   = P1^2;
71.     sbit T2EX  = P1^1;
72.     sbit T2    = P1^0;
73.
74.     /* PORT 2 */
75.     sfr  P2    = 0xA0;
76.     sbit P2_7  = P2^7;
77.     sbit P2_6  = P2^6;
78.     sbit P2_5  = P2^5;
79.     sbit P2_4  = P2^4;
80.     sbit P2_3  = P2^3;
81.     sbit P2_2  = P2^2;
82.     sbit P2_1  = P2^1;
83.     sbit P2_0  = P2^0;
84.
85.     /* PORT 3 */
86.     sfr  P3    = 0xB0;
87.     sbit P3_7  = P3^7;
88.     sbit P3_6  = P3^6;
89.     sbit P3_5  = P3^5;
90.     sbit P3_4  = P3^4;

```

```

91.      sbit P3_3 = P3^3;
92.      sbit P3_2 = P3^2;
93.      sbit P3_1 = P3^1;
94.      sbit P3_0 = P3^0;
95.
96.      sbit RD    = P3^7;
97.      sbit WR    = P3^6;
98.      sbit T1    = P3^5;
99.      sbit T0    = P3^4;
100.     sbit INT1   = P3^3;
101.     sbit INT0   = P3^2;
102.     sbit TXD    = P3^1;
103.     sbit RXD    = P3^0;
104.
105.     /* PORT 4 */
106.     sfr  P4     = 0xC0;
107.     sbit P4_7   = P4^7;
108.     sbit P4_6   = P4^6;
109.     sbit P4_5   = P4^5;
110.     sbit P4_4   = P4^4;
111.     sbit P4_3   = P4^3;
112.     sbit P4_2   = P4^2;
113.     sbit P4_1   = P4^1;
114.     sbit P4_0   = P4^0;
115.
116.     /* PORT 5 */
117.     sfr  P5     = 0xE8;
118.     sbit P5_7   = P5^7;
119.     sbit P5_6   = P5^6;
120.     sbit P5_5   = P5^5;
121.     sbit P5_4   = P5^4;
122.     sbit P5_3   = P5^3;
123.     sbit P5_2   = P5^2;
124.     sbit P5_1   = P5^1;
125.     sbit P5_0   = P5^0;
126.
127.
128.     /* ACC */
129.     sfr  ACC     = 0xE0;
130.     sbit ACC_7   = ACC^7;
131.     sbit ACC_6   = ACC^6;
132.     sbit ACC_5   = ACC^5;
133.     sbit ACC_4   = ACC^4;
134.     sbit ACC_3   = ACC^3;
135.     sbit ACC_2   = ACC^2;
136.     sbit ACC_1   = ACC^1;
137.     sbit ACC_0   = ACC^0;
138.
139.     /* PCON */

```

```

140.    sfr  PCON  = 0x87;
141.    sbit SMOD = 0x8E;
142.    sbit SMOD1 = 0x8E;
143.    sbit SMOD0 = 0x8D;
144.    sbit POF   = 0x8B;
145.    sbit GF1   = 0x8A;
146.    sbit GF0   = 0x89;
147.    sbit PD    = 0x88;
148.    sbit IDL   = 0x87;
149.
150.
151.    /* TCON */
152.    sfr  TCON  = 0x88;
153.    sbit TF1   = TCON^7;
154.    sbit TR1   = TCON^6;
155.    sbit TF0   = TCON^5;
156.    sbit TR0   = TCON^4;
157.    sbit IE1_  = TCON^3;
158.    sbit IT1   = TCON^2;
159.    sbit IE0_  = TCON^1;
160.    sbit IT0   = TCON^0;
161.
162.
163.    /* SCON */
164.    sfr  SCON_1 = 0xC0;
165.    sfr  SCON   = 0x98;
166.    sbit SM0   = SCON^7;
167.    sbit FE    = SCON^7;
168.    sbit SM1   = SCON^6;
169.    sbit SM2   = SCON^5;
170.    sbit REN   = SCON^4;
171.    sbit TB8   = SCON^3;
172.    sbit RB8   = SCON^2;
173.    sbit TI    = SCON^1;
174.    sbit RI    = SCON^0;
175.
176.    sfr  SBUF_1 = 0xC1;
177.    sfr  SBUF   = 0x99;
178.
179.    sfr  BRL    = 0x9A;
180.
181.
182.    sfr  WDTRST = 0xA6;
183.    sfr  WDTPRG = 0xA7;
184.
185.    sfr  AUXR1  = 0xA2;
186.
187.    /* IE */
188.    sfr  IE     = 0xA8;

```



```

189.     sfr    IE0      =    0xA8;
190.     sbit   EA       =    IE^7;
191.     sbit   EC       =    IE^6;
192.     sbit   ET2      =    IE^5;
193.     sbit   ES       =    IE^4;
194.     sbit   ET1      =    IE^3;
195.     sbit   EX1      =    IE^2;
196.     sbit   ET0      =    IE^1;
197.     sbit   EX0      =    IE^0;
198.
199.     sfr    SADDR     =    0xA9;
200.
201.
202.     sfr    SADDR_1   =    0xAA;
203.
204.
205.
206.     sfr    AUXR      =    0x8E;
207.
208.     sfr    SADEN      =    0xB9;
209.
210.     sfr    SADEN_1    =    0xBA;
211.
212.     sfr    IPH        =    0xB7;
213.     sfr    IPH0       =    0xB7;
214.
215.     sfr    IPL        =    0xB8;
216.     sfr    IPL0       =    0xB8;
217.
218.     sbit   PPC       =    IPL^6;
219.     sbit   PT2       =    IPL^5;
220.     sbit   PS        =    IPL^4;
221.     sbit   PT1       =    IPL^3;
222.     sbit   PX1       =    IPL^2;
223.     sbit   PT0       =    IPL^1;
224.     sbit   PX0       =    IPL^0;
225.
226.
227.     /*    T2CON    */
228.     sfr    T2CON     =    0xC8;
229.     sbit   TF2       =    T2CON^7;
230.     sbit   EXF2      =    T2CON^6;
231.     sbit   RCLK      =    T2CON^5;
232.     sbit   TCLK      =    T2CON^4;
233.     sbit   EXEN2     =    T2CON^3;
234.     sbit   TR2       =    T2CON^2;
235.     sbit   CT2       =    T2CON^1;
236.     sbit   CPRL2     =    T2CON^0;
237.

```

```

238.
239.      /* T2MOD */
240.      sfr  T2MOD  =  0xC9;
241.      sfr  RCAP2L  =  0xCA;
242.      sfr  RCAP2H  =  0xCB;
243.      sfr  TL2     =  0xCC;
244.      sfr  TH2     =  0xCD;
245.
246.      sfr  BDRCON  =  0x9B;
247.
248.      sfr  BDRCON_1 =  0x9C;
249.
250.
251.      /* PSW */
252.      sfr  PSW     =  0xD0;
253.      sbit  CY      =  PSW^7;
254.      sbit  AC      =  PSW^6;
255.      sbit  F0      =  PSW^5;
256.      sbit  RS1     =  PSW^4;
257.      sbit  RS0     =  PSW^3;
258.      sbit  OV      =  PSW^2;
259.      sbit  UD      =  PSW^1;
260.      sbit  P       =  PSW^0;
261.
262.
263.      /* CCON */
264.      sfr  CCON     =  0xD8;
265.      sbit  CF      =  CCON^7;
266.      sbit  CR      =  CCON^6;
267.      sbit  CCF4    =  CCON^4;
268.      sbit  CCF3    =  CCON^3;
269.      sbit  CCF2    =  CCON^2;
270.      sbit  CCF1    =  CCON^1;
271.      sbit  CCF0    =  CCON^0;
272.
273.
274.      // CMOD //
275.      sfr  CMOD     =  0xD9;
276.      sfr  CCAPM0   =  0xDA;
277.      sfr  CCAPM1   =  0xDB;
278.      sfr  CCAPM2   =  0xDC;
279.      sfr  CCAPM3   =  0xDD;
280.      sfr  CCAPM4   =  0xDE;
281.
282.      sfr  CL       =  0xE9;
283.      sfr  CCAP0L   =  0xEA;
284.      sfr  CCAP1L   =  0xEB;
285.      sfr  CCAP2L   =  0xEC;
286.      sfr  CCAP3L   =  0xED;

```

```

287.     sfr  CCAP4L  =  0xEE;
288.
289.     sfr  CH      =  0xF9;
290.     sfr  CCAP0H  =  0xFA;
291.     sfr  CCAP1H  =  0xFB;
292.     sfr  CCAP2H  =  0xFC;
293.     sfr  CCAP3H  =  0xFD;
294.     sfr  CCAP4H  =  0xFE;
295.
296.
297.     #endif
298.     //=====
=====
299.     //                               End Of File "REG89C51.h"

300.     //=====
=====

```

```

1. // wirtten by Yarin Avisidris & Idan Lisha and guided by
   Ludmila koz
2. #include "REG_89C51.h"
3. #include "general.h"
4. #include "lcdp2.h"
5. #define bit_IR P3_7
6. bit flag=0;
7. sbit trig=P3^6;
8. sbit echo=P3^3; // echo=gate=p3.3
9. byte data arr_remote[4];
10.    byte distances[2];
11.    uint time,dist;
12.    //===== ULTRASONIC SENSOR INTERRUPTS A
   ND FUNTIONS=====
13.    void us_trig() { // trigger function for ultrasonic s
   ensor
14.        trig=1;
15.        nop();nop();nop();// delay of 14us requirement
   s are atleast 10us for sr_04 Trigger
16.        trig=0;
17.        msec(50);
18.    } // end trigger
19.    void distance() interrupt 2 { // interrupt function
   (echo falls int1 interrupt)
20.        time =TH1;
21.        time =time<<8;//shift left 8
22.        time =time|TL1; //
23.        dist=time/58;
24.        TH1=TL1=0;
25.        flag=1;
26.    }
27.    //=====END OF ULTRASONIC SENSOR INT
   ERUPTS AND FUNCTIONS=====
28.    //=====REMOTE IR FUNC
   TION=====
29.
30.    bit remote()
31.    {
32.        byte data num,num_bit,len_arr;
33.        uint data time_units;
34.        //===== 16T =====//
35.
36.        TL1=TH1=0;
37.        TR1=1;
38.        while(!bit_IR); //bit_IR up
39.        TR1=0; //stop timer
40.        time_units =TH1;

```

```

41.         time_units =time_units <<8;//shift left 8
42.         time_units =time_units |TL1; //
43.         if(time_units <11060 && time_units >6248)// chec
k 16T
44.         {
45.             //===== 8T =====//
46.             TL1=TH1=0;
47.             TR1=1;
48.             while(bit_IR);//bit_IR down
49.             TR1=0; //stop timer
50.             time_units =TH1;
51.             time_units =time_units <<8;
52.             time_units =time_units |TL1;
53.             if(time_units <6248 && time_units >3124)
54.             {
55.                 //===== start of transmitting(DATA) ===
=====//
56.                 for(len_arr=0;len_arr<4;len_arr++)
57.                 {
58.                     num=0;
59.                     for(num_bit=0;num_bit<8;num_bit++)
60.                     {//===== 1T =====//
61.                         while(!bit_IR);//bit_IR up

62.                         //===== 3T =====//
63.                         TL1=TH1=0;
64.                         TR1=1;
65.                         while(bit_IR); //bit_IR down

66.                         TR1=0; //stop timer
67.                         time_units =TH1;
68.                         time_units =time_units <<8;
69.                         time_units =time_units |TL1;
70.                         if(time_units <1823 && time_u
nits >1041)//3t
71.                             num|=1<<num_bit;
72.                     }//for8
73.                     arr_remote[len_arr]=num;
74.                 }//for4
75.                 if(arr_remote[0]==~arr_remote[1]) //address
&& !address
76.                     if(arr_remote[2]==~arr_remote[3]
)
77.                         return 1;
78.                 return 0;
79.             } // 8t
80.             return 0 ;
81.         }//16t check
82.         return 0;

```

```

83.     }
84.     //===== END OF REMO
TE FUNCTION=====
85.     //===== Start of DC motors
functions=====
86.     void forward()
87.     {
88.         P1_0=0; P1_2=1;
89.         P1_1=1; P1_3=0;
90.
91.     }
92.     void backward() {
93.         P1_0=1; P1_2=0;
94.         P1_1=0; P1_3=1;
95.
96.     }
97.     void left() {
98.         P1_0=1; P1_2=1;
99.         P1_1=0; P1_3=0;
100.
101.    }
102.    void right() {
103.        P1_0=0; P1_2=0;
104.        P1_1=1; P1_3=1;
105.
106.    }
107.    }
108.    void motorstop() {
109.        P1_0=0; P1_2=0;
110.        P1_1=0; P1_3=0;
111.
112.    }
113.    void servo_x(bit control) {
114.        if(control==1 & CCAP2H>230 ) CCAP2H--
; // 1 LOGIC AT CONTROL IS ADDIN ANGLE
115.        if(control==0 & CCAP2H<244 ) CCAP2H++; // 0 LOGIC
AT CONTROL IS SUB ANGLE
116.    }
117.    void servo_y(bit control) {
118.        if(control==1 & CCAP3H>230 ) CCAP3H--
; // 1 LOGIC AT CONTROL IS ADDIN ANGLE
119.        if(control==0 & CCAP3H<244 ) CCAP3H++; // 0 LOGIC
AT CONTROL IS SUB ANGLE
120.    }
121.    void servo_init() {
122.        CCAP1H=244;
123.        CCAP2H=238;
124.        CCAP3H=244;
125.        CCAP4H=244;

```

```

126.     }
127.     //===== end of DC Motors fun
    ctions=====
128.     void main()
129.     {
130.         bit A=0;
131.         byte f;
132.         byte prev=10000;
133.         init_lcd();
134.         servo_init();
135.         start:
136.         TMOD=0x12; //timer 0 8 bit auto reload pwm for serv
            o motor, timer1 mode1(ir config)
137.         TH0=0xB8;
138.         TL0=0xB8;
139.         TR0=1;
140.         CCAPM1=0x42;
141.         CCAPM2=0x42;
142.         CCAPM3=0x42;
143.         CCAPM4=0x42;
144.         CMOD=4;
145.         CCON=0x40;
146.         while(bit_IR) lcd_puts(0x84,"IR wait");//waiting
            for steady mode to fall (start of comm)
147.         lcd_clr();
148.         while(1)
149.         {
150.             if(remote()){
151.                 switch(arr_remote[2]) {
152.                     case _plus: lcd_clr();forward();
                        break;
153.                     case _minus: lcd_clr(); backward
                        ();break;
154.                     case _play: lcd_clr(); motorstop
                        ();break;
155.                     case _left: lcd_clr(); left();br
                        eak;
156.                     case _right: lcd_clr(); right();
                        break;
157.                     case _6: lcd_clr(); servo_x(1)
                        ; break;
158.                     case _4: lcd_clr(); servo_x(0);
                        break;
159.                     case _5: lcd_clr();
160.                         if(CCAP4H==244) CCAP4H
                            =230; else CCAP4H=244; break; // open/close
161.                     case _2: lcd_clr(); servo_y(0);
                        break;

```

```

162.                case _8: lcd_clr(); servo_y(1);    break;
163.                case _spin: if(CCAP1H==244) CCAP1H=234; el
se CCAP1H=244; break;
164.                case _test: lcd_clr();    TMOD=0x
92;//timer 1 int1 interrupt for ultrasonic timer0 8bit au
toreload for pwm
165.                    EX1=1; //enabling interrupt
1        1
166.                    TR1=1; // running timer 1, tim
er runs when gate is 1.
167.                    IT1=1; // falling edge inter
rupt 1
168.                    EA=1; // enabling all interr
upts
169.                    TL1=TH1=0;
170.                    CCAP2H=238; // arm in middle
171.                //        forward();
172.                //        control();
173.                //        while(1) {
174.                //
175.                //                                us_t
rig();
176.                //        while(!flag);
177.                //        flag=0;
178.                //    set_lcd(0,0x85);
179.                //    set_lcd(1,dist/100+0x30);
180.                //    dist=dist%100;
181.                //    set_lcd(1,dist/10+0x30);
182.                //    set_lcd(1,dist%10+0x30);
183.                //        }
184.
185.                while(1) {
186.
187.                    A=0;
188.                    while(A==0) {
189.                        if(dist<=18) {
190.                            A=1;
191.                            motorstop();
192.                        }
193.                        if(A==0) {
194.                            forward();
195.                            msec(400);
196.                            motorstop();
197.                        }
198.                        us_trig();
199.                        while(!flag);
200.                        flag=0;

```



```

200.                                     if(dist<=18 & A==0) {
201.                                         lcd_puts(0x84,"B");
202.                                     A=1;
203.                                     }
204.
205.
206.                                     if(A==0) {
207.                                         CCAP1H=234;
208.                                     msec(400);
209.                                     us_trig();
210.                                     while(!flag);
211.                                     flag=0;
212.                                     if(dist>65) {
213.                                         //FINDING EXIT AT LEFT
214.                                         //AND EXITING THE ROOM
215.                                         //FROM THE LEFT
216.                                         forward();
217.                                         msec(1000);
218.                                         left();
219.                                         msec(800);
220.                                         motorstop();
221.                                         forward();
222.                                         msec(3000);
223.                                         motorstop();
224.                                     lcd_puts(0x84,"I exited the room !");
225.                                     goto start;
226.                                     }
227.                                     ///////
228.                                     /////// MOVING STR
229.                                     ///////
230.                                     ///////
231.                                     if(dist<=17) {
232.                                         right();
233.                                         while(dist<=prev) {
234.                                             us_trig();
235.                                         while(!flag);
236.                                         flag=0;
237.                                         prev=dist;
238.                                         us_trig();
239.                                         while(!flag);
240.                                         flag=0;
241.                                         }
242.
243.                                     }
244.                                     if(dist>=15) {
245.                                         left();

```

```

246.             while(dist>=prev) {
247.                 us_trig();
248.             while(!flag);
249.             flag=0;
250.             prev=dist;
251.             us_trig();
252.             while(!flag);
253.             flag=0;
254.             }
255.
256.         }
257.     }
258.     //msec(500);
259.     CCAP1H=244;
260.     msec(100);
261.
262. }
263.
264.
265.
266.         ///// this is for EDGE solvi
ng when the car needs to rotate right with controll
267.         /////
268.
269.         if(A==1) {
270.             CCAP1H=234;
271.             msec(250);
272.         }
273.         while(A==1 ) {
274.             f=0;
275.             right();
276.             lcd_puts(0x84,"C");
277.
278.             while(f<2) {
279.                 us_trig();
280.                 while(!flag);
281.                 flag=0;
282.                 prev=dist;
283.
284.                 us_trig();
285.                 while(!flag);
286.                 flag=0;
287.                 if(dist>prev) f++;
288.             }
289.             CCAP1H=244;
290.             A=0;
291.             motorstop();
292.         }
293.     } // end while(1) automatic

```

```
294.                                     }
295.
296.                                     }
297.                                     }
298.                                     }
```

#### **4.1.4 הסבר פונקציות**

##### **חיישן מרחק SR04:**

פונקציה ה `us_trig` מייצרת TRIGGER מתאים לרגל ה TRIGGER של החיישן לפי הדרישות יצרן נדרש פולס חיובי ברוחב 10us לפחות כאשר יש רווח של 50ms בין כל פולס, בפרוייקט שלנו נתנו לו 14us.

אנו יוצרים את הפולס על ידי פעולה פשוטה של הרמת הדק ה TRIGGER למשך הזמן הנדרש בעזרת `Delay`.

לחישוב המרחק אנו משתמשים בפונקציה `distance() interrupt 2` שהיא פונקציה פסיקה חיצונית INT1 שמתקבלת כאשר הדק ה ECHO של החיישן נופל מ '1' לוגי ל '0' לוגי והתוכנית הראשית מיד קופצת לפונקציה זו כדי לטפל בחישוב המרחק, חשוב לציין שהדק זה לעולם לא יפול מ '1' ל '0' ללא מתן TRIGGER מתאים בעזרת הפונקציה הריאשונה ולכן אנו יודעים שפסיקה עלולה להופיע לאחר הפעלת הפונקציה `us_trig`

כשרגל ה ECHO של החיישן נמצאת במצב '1' טיימר 1 רץ ומונה את הזמן וכאשר ההדק נופל ל '0' התוכנית הראשית קופצת לפונקציה זו בכדי לחשב את המרחק, הזמן נמצא בתוך האוגרים של הטיימר לאחר חיבור של שתי אוגרי ה TH1 ו TL1 בעמצאות פעולת אור לוגי אנו מקבלים את הזמן וניתן לחלק אותו ב 58 בכדי להגיע אל המרחק ב יחידות CM.

##### **חיישן מקלט IR:**

לשם קליטת נתונים מהחיישן IR אנו משתמשים בפונקציה `bit remote()` פונקציה זו היא פונקציה שמחזירה ערך מסוג BIT זאת אומרת 0 או 1 כאשר 1 מסמל שהתקשורת תקינה והנתונים נשמרו במערך `arr_remote` ו 0 שהתקשורת לא תקינה.

לפי פרוטוקול RECS-80 יש זמנים קבועים עבור הסיביות '1' ו '0' לוגי שאפשר לראות בחלק של פרוטוקולים בספר, בפונקציה אנו מודדים את הזמן שבו המקלט משדר רמה לוגית נמוכה ורמה לוגית גבוהה ובכך מבינים איזה סיבית שודרה '0' או '1' ובכך מפענחים את המידע, לפי פרוטוקול זה נשלחים 4 בתים אל החיישן כאשר הם בסדר הזה ADDRESS, NOT, ADDRESS, DATA, NOT. זאת אומרת נשלחים סה"כ 32 סיביות לא כולל START BIT, הפונקציה מבצעת פעולת NOT על המידע שנשלח כ NOT ומשווה אותו עם המידע שנשלח בצורה מקורית ובכך מגיע למסכנה שאכן התקשורת תקינה, ובנוסף הפונקציה ממפה את הבתים הללו במערך `arr_remote` כאשר הוא מערך של 4 ובתא ה 2 נמצא המידע שלנו שהוא חשוב לנו לצורך שימוש בנתונים שנשלחים.

##### **הפעלת מנועי DC:**

לשם הפעלת מנועי ה DC שיש לנו 2 בפרוייקט שעובדים בדומה לתנועה של טנק יש לנו 5 פונקציות, `forward()`, `backward()`, `left()`, `right()`, `motorstop()` כאשר כל פונקציה דואגת

להעלות את ההדקים שמחוברים לדרייבר מנועי ה DC בהתאמה כזו שתצור תנועה מתאימה לדוגמא forward() תצור פעולה של נסיעה קדימה כאשר היא משנה את הדקי הבקר בצורה הבאה :

1. P1\_0=0; P1\_2=1;
2. P1\_1=1; P1\_3=0;

צירוף זה של ההדקים יגרום לנסיעה קדימה, צירוף הפוך של ההדקים יגרום לנסיעה אחורה

1. P1\_0=1; P1\_2=0;
2. P1\_1=0; P1\_3=1;

בדומה ל פונקציות left(),right() הם דואגות לפניה שמאלה וימינה של הרכב ועובדות בצירופים שונים בדומה לפונקציות קדימה ואחורה, הפונקציה האחרונה שהיא דואגת לעצירה מלאה של הרכב נקראת motorstop() וכוללת את הרצף הבא:

1. P1\_0=0; P1\_2=0;
2. P1\_1=0; P1\_3=0;

### הפעלת מנועי הסרבו:

מנועי הסרבו הם בעצם מנועי DC עם מערכת בקרה אשר מקבלת 2 נתונים, היא מקבלת את מיקום הגיר הנוכחי שנמצא המנוע כרגע ומקבלת PWM אשר מציין לאיזה זווית היא שואפת להגיע, בעזרת המערכת בקרה שנמצאת במנוע הוא יודע שיש שגיאה או סטייה בין המיקום הנוכחי של גיר המנוע ולבין המיקום הסופי שאליו הוא צריך להגיע בכדי שהגיר תהיה בזווית הרצויה לפי ה PWM שניתן לו בהדק ה SIGNAL שלו.

בכדי ליצור PWM בבקר זה שהוא כולל PCA לשם יצירת PWM הפעלנו את ה PCA בעזרת הפקודה הבאה: CCON=0x40;

והגדרנו שההדקים P1.4-P1.7 יהיו הדקים שמוצאים PWM וה PWM יוצא ברקע לתוכנית הראשית, זאת אומרת שהתוכנית הראשית לא מטפלת ביצירת ה PWM עבור המנועים, אותות אלו נוצרים ברקע לתוכנית.

לפי דרישות היצרן של מנוע הסרבו בכדי לשלוט עליו נדרש פולס בתדר 50hz כאשר Ton נע בין 1ms-2ms כאשר 1ms מציין זווית 0 ו 2ms מציין זווית מקסימלית של 120 מעלות את Ton אנו משנים בעזרת שינוי של ערך ה CCAP של ההדק המתאים כאשר CCAP2 הוא בשביל הציר האופקי CCAP3 בשביל הציר האנכי ו CCAP4 בשביל סגירה או פתיחה של כף היד של הזרוע.

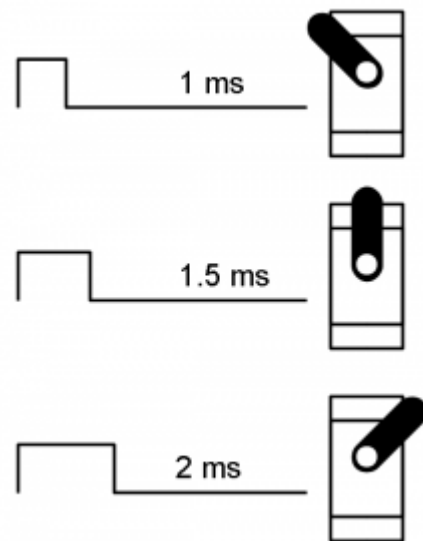
להלן דוגמא של איתחול מנועי ה-servo : servo\_init()

```

1. void servo_init() {
2.     CCAP1H=244;
3.     CCAP2H=238;
4.     CCAP3H=244;
5.     CCAP4H=244;
6. }

```

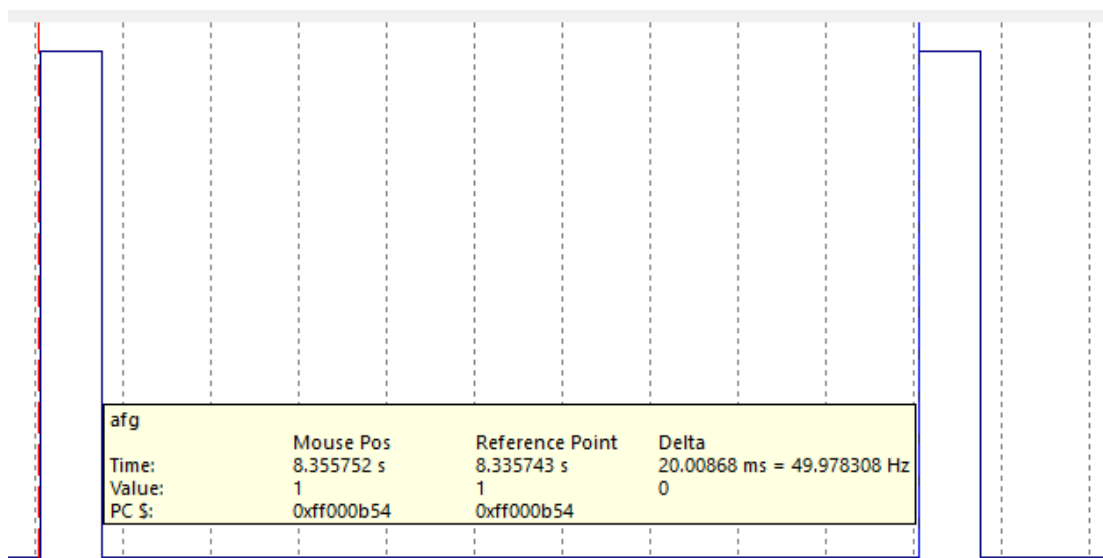
CCAP1 שייך להזזת החיישן מרחק, כפי שניתן לראות לכל ה CCAPH של ההדקים נתנו את הערך 244 שאומר זווית 0, חוץ מ CCAP2H שניתן לו הערך 238 שאומר להזיז את הציר האנכי של הזרוע למרכז הרכב, זווית 0 למנוע האנכי של הרכב תזיז את הזרוע אנכית בערך 60 מעלות ימינה.



**פרק 5:**  
**סימולציות**

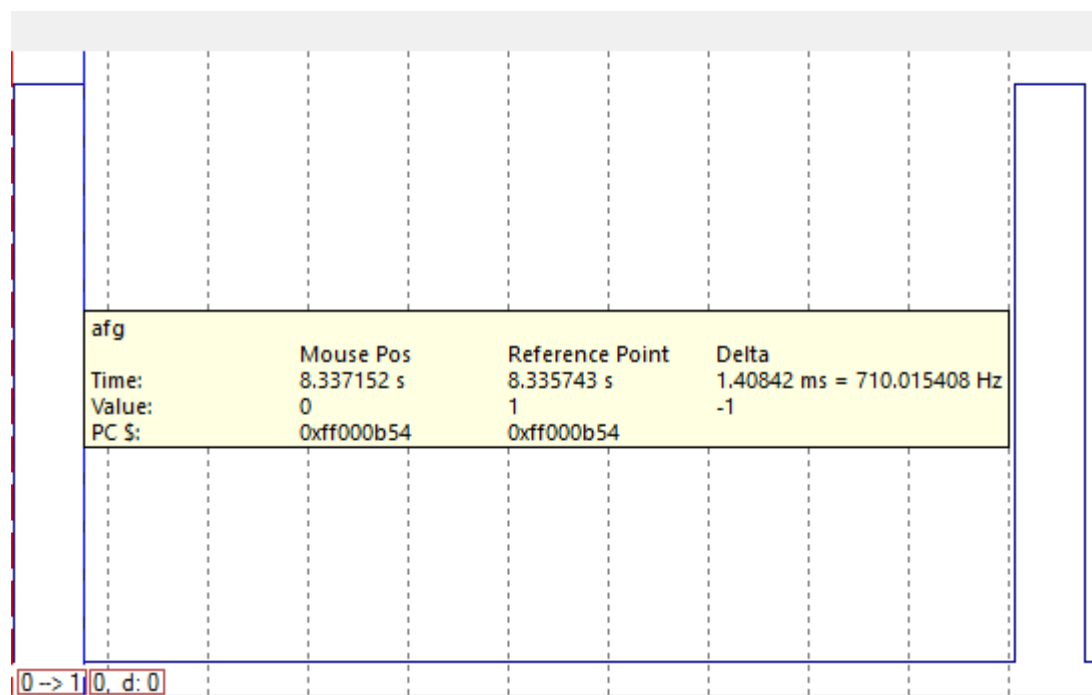
### סימולציית PWM ב- Keil :

הסימולציה הבאה מראה את PWM להדק P1.5 שהוא המנוע האנכי של הזרוע



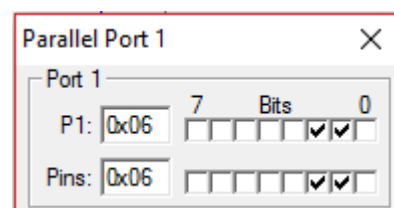


אפשר לראות שהתדר הוא 50hz וגם ניתן לראות שרוחב הפולס הוא 1.4ms שזה נמצא בין 1ms ל 2ms בפונקציה Servo\_init() נתנו פקודה לציר זה לזוז למרכז הרכב אפשר לראות שבאמת 1.4ms נמצא מרכז בין 1ms ל 2ms להלן הגרף :

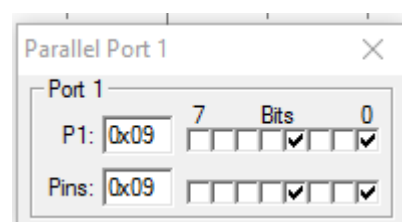


### סימלוציית צירוף לפורט 1 בשביל מנועי ה DC ב- Keil:

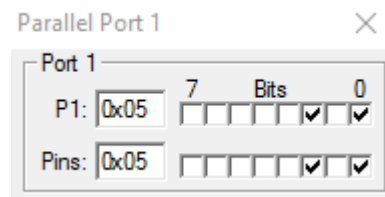
עבור הפונקציה forward() שדואגת לנסיעה קדימה



עבור הפונקציה backward() שדואגת לנסיעה אחורה



עבור הפונקציה left() שדואגת לפניה שמאלה



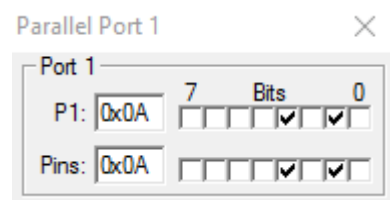
Parallel Port 1

Port 1

P1: 0x05

Pins: 0x05

עבור הפונקציה right() שדואגת לפניה ימינה



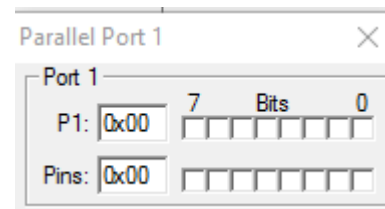
Parallel Port 1

Port 1

P1: 0x0A

Pins: 0x0A

עבור הפונקציה motorstop() שדואגת לעצירה מלאה של הרכב



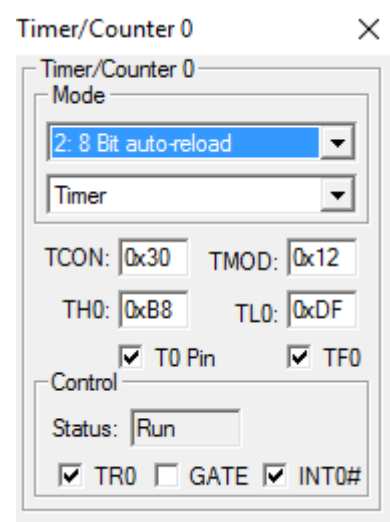
Parallel Port 1

Port 1

P1: 0x00

Pins: 0x00

הגדרת timer0 בשביל יצירת PWM בעזרת PCA



Timer/Counter 0

Mode

2: 8 Bit auto-reload

Timer

TCON: 0x30

TMOD: 0x12

TH0: 0xB8

TL0: 0xDF

☒ T0 Pin ☒ TF0

Control

Status: Run

☒ TR0 ☐ GATE ☒ INT0#

הגדרת timer1 בשביל קריאת נתונים מהחיישן מקלט IR :

Timer/Counter 1

Mode

1: 16 Bit Timer/Counter

Timer

TCON: 0x30 TMOD: 0x12

TH1: 0x00 TL1: 0x00

☒ T1 Pin ☐ TF1

Control

Status: Stop

☐ TR1 ☐ GATE ☒ INT1#

הגדרת ה PCA בשביל ה PWM :

PCA Timer

PCA Timer

Input Mode: Timer 0 overflow

Status: Run ☒ CR

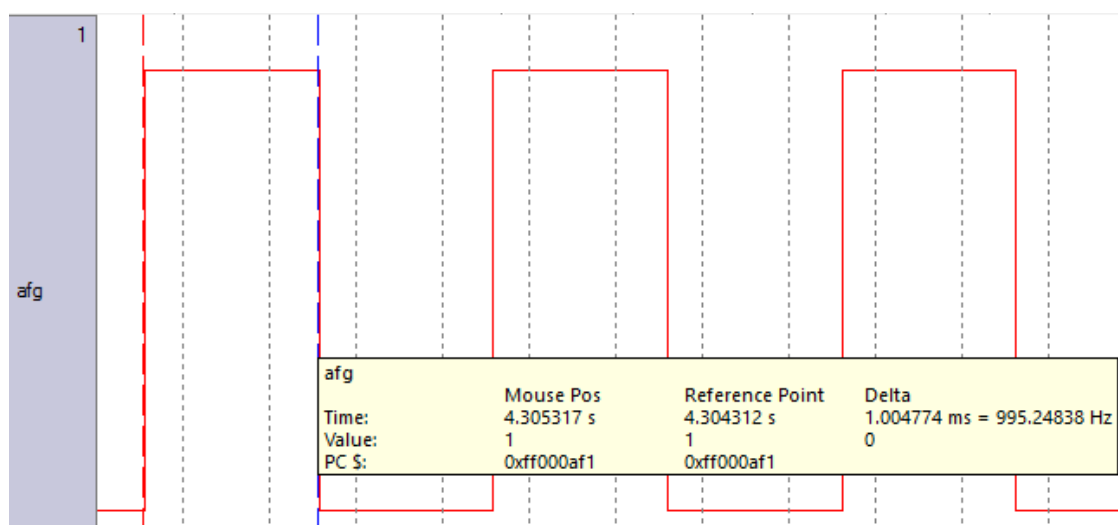
CH/CL: 0x22EF ☐ CF

CMOD: 0x04 ☒ ECI

CCON: 0x40 ☐ CIDL

☐ WDTE

סימולציה של ה פונקציה msec() שהיא נמצאת ב general.h אשר יוצרת דיילי באורך זמן של 1ms עבור כל 1 שהיא מקבלת זאת אומרת שהפונקציה תצור דיילי של 20ms אם נרשום msec(20); להלן הגרף:



## פרק 6:

### סיכום ומסקנות

-

### **6.1 סיכום ותקלות:**

בפרויקט זה בנינו רובוט מכאני עם זרוע אשר נשלט על ידי שלט IR , ויוצא מהחדר באופן אוטומטי.

הרובוט והזרוע מופעלים באופן ידי על ידי השלט מטרת הזרוע היא להרים חפצים מהרצפה , הרובוט מופעל ממנועי DC והזרוע מופעלת ממנועי SERVO .  
היציאה מהחדר מתבצעת באופן אוטומטי על ידי חיישן אולטרסוני אשר מודד את מרחק הרכב מקיר החדר וכאשר המרחק גדול מהנדרש האו יודע שהוא מצא יציאה.

בפרויקט זה למדנו איך לעבוד עם מנועי SERVO איך ניתן לתפעל אותם לפי זוויות, למדנו איך להשתמש בחיישן האולטרסוני , איך למדוד אתו מרחק ואיך הוא יודע את המרחק בעזרת רגל ה ECHO שיש לו.

שיטת העבודה שלנו בפרויקט הייתה קודם כל לדעת איך פועל כל רכיב או חיישן בנפרד ולאחר מכאן לחבר אותו לפרויקט ולהתחיל לעבוד אתו.

### **6.2 תקלות במהלך הפרויקט:**

1. כאשר חיברנו את ההדק Signal ממנוע הסרבו ישירות להדקי הבקר אז המנוע אינו פעל ולאחר מכן חיברנו דרך החוצץ וכך המנועים פעלו.

## פרק 7:

### זיווד









**פרק 8:**  
**ביבליוגרפיה**

## **אתרים:**

<http://www.keil.com/dd/docs/datashts/philips/p89v51rd2.pdf>

<http://wikipedia.org>

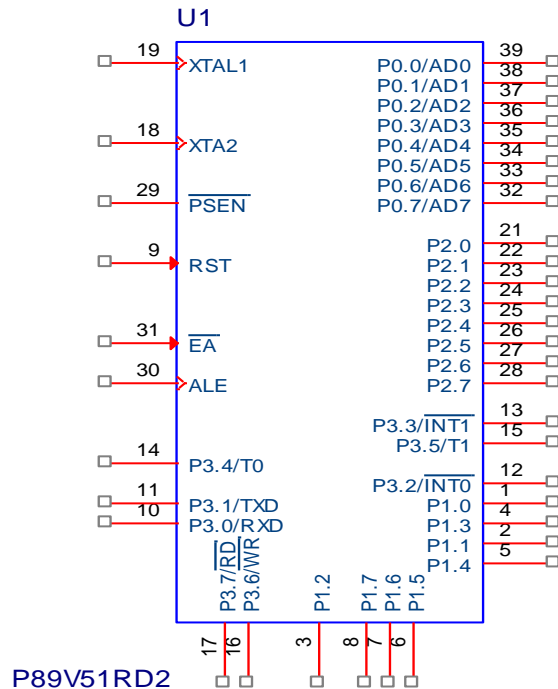
<http://www.micropik.com/PDF/HCSR04.pdf>

<http://shai.eguru-il.com>

## **פרק 9:**

## **נספחים**

## 9.1 מיקרו-בקר P89V51RD2



### תכונות:

- זיכרון פלאש 64KB.
- זיכרון נתונים 1KB RAM
- In system programming- ISP – מאפשר לצרוב תכנית למיקרו-בקר מבלי להוציא אותו מן המעגל.
- In application programmable- IAP - ניתן לשנות הגדרות של זיכרון התוכנה פלאש אונליין.
- תדרי עבודה עד 40MHz.
- מספק 12 דפקי שעות במחזור מכונה אחת (ברירת מחדל) או ניתן לבחור 6 דפקי שעות למחזור מכונה אחת.
- פרוטוקול תקשורת טורי סינכרוני SPI.
- UART- בקר תקשורת אסינכרונית משופר.

- PCA – (programmable counter array) - מערך מונים עם בקרת Duty cycle (מחזור פעולה) - PWM (pulse width modulation)
- ארבע שמונה ביט I/O פורטים מהם שלושה הדקים של פורט אחד מספקים זרם גבוה עד 16mA כל אחד.
- שלושה טיימרים בני 16 ביט.
- טיימר watchdog הניתן לתכנות.
- שמונה מקורות פסיקה עם 4 רמות עדיפות.

### תיאור הדקים:

**RESET** – קו זה משמש לאיפוס הרכיב. האיפוס נועד לכוון את המיקרו בקר לבצע את תכנית ההפעלה שלו החל מכתובת 0000H בזיכרון התוכנית. איפוס הרכיב מתבצע ע"י מתן רמה לוגית גבוהה "1".

**X1, X2** – קווי חיבור הגביש לתזמון הרכיב כאשר תדר העבודה הוא תדר הגביש.

\* נחיצות הגביש היא לשם שמירה על תדירות קבועה.

**PORT0** – קווי קלט/פלט היכולים להתנהג כמפתח דו כיווני ניתן לגשת בתוכנה לכל הדק בנפרד או לכל המילה בת ה-8 סיביות. מפתח זה הוא מפתח מרובב בזמן (7AD-0AD), כלומר על מפתח זה יופיעו גם נתונים וגם כתובות חלק נמוך.

**PORT1** - מפתח קלט/פלט המשמש לחיבור של אמצעי קלט/פלט שונים כמו לוח מקשים לדוגמא.

**PORT2** - מפתח זה מהווה את החלק הגבוה של הכתובות (15A-8A), שימושי בעיקר להרחבת הזיכרון.

**PORT3** - משמש כמפתח קלט/פלט או כמפתח הבקרה והפסיקות ומכיל את ההדקים הבאים:

**RXD** - הדק לקליטה טורית של נתונים.

**TXD** - הדק לשידור טורי של נתונים.

**INT1, INT0** - הדקי קבלת פסיקה חומרה פעילות או בירידה או ברמה "0".

**T1, T0** - הדק להפעלת טיימר, '0' בצורה חיצונית (OT) והדק להפעלת טיימר '1' בצורה חיצונית (1T)

**WR'-RD'** - הדקי בקרת כתיבה או קריאה לזיכרון נתונים חיצוני.

**ALE** - הדק המבצע את ההפרדה בפורט 0 המרובב בין פס הנתונים לבין פס הכתובות חלק נמוך. "0" - בהדק זה יעברו הכתובות חלק נמוך למוצא.

"1" - בהדק זה יעברו נתונים מהמיקרו או אל המיקרו.

**'PSEN** - הדק לציון קריאה מזיכרון תכנית חיצוני. ניתן להתייחס להדק זה כהדק RD/נוסף במיקרו המשמש לקריאה מזיכרון תכנית חיצוני.

**'EA** - הדק בקרה לגבי זיכרון תוכנה להדק זה 2 אופציות:  
 כאשר נחבר הדק זה לאדמה תתאפשר עבודה עם זיכרון תוכנה חיצוני.  
 כאשר נחבר הדק זה ל - VCC תתאפשר עבודה עם זיכרון תוכנה פנימי.

#### **אוגרים ב-8051:**

#### **האוגר: IE(Interrupt Enable)**

EA	-	-	ES	ET1	EX1	ET0	EX0
----	---	---	----	-----	-----	-----	-----

EA - ע"י מתן '0 בסיבית זו ממסכים את כל הפסיקות וע"י מתן '1 בסיבית זו יש לנו גישה לאפשר את הפסיקות השונות.

ES - אפשר /מיסוך פסיקת ה. UART-

ET1 - אפשר/מיסוך פסיקת טיימר 1.

EX1 - אפשר /מיסוך פסיקת INT1.

ET0 - אפשר / מיסוך פסיקת טיימר 0.

EX0 - אפשר/מיסוך פסיקת INT0.

#### **האוגר: IP(Interrupt Priority)**

-	-	-	PS	PT1	PX1	PT0	PX0
---	---	---	----	-----	-----	-----	-----

- PS- מתן עדיפות לפסיקת UART.
- PT1- מתן עדיפות לפסיקת TIMER1.
- PX1- מתן עדיפות לפסיקת INT1.
- PT0- מתן עדיפות לפסיקת TIMER0.
- PX0- מתן עדיפות לפסיקת INT0.

#### **TCON(Timer Control): האוגר**

TF1	TR1	TF0	TRO	IE1	IT1	IE0	IT0
-----	-----	-----	-----	-----	-----	-----	-----

**באוגר זה 8 סיביות בקרה אך רק 4 מאיתן שייכות לאפיוני הטיימר-**

IT0 / IT1 - '0' בסיבית זו תקבע כי תצורת הדרבון תהיה רמה שלילית בפסיקות INT0/INT1.

'1' בסיבית זו יקבע כי תצורת הדרבון תהיה דרבון קצה שלילי בפסיקות INT0 / INT1.

IE0/IE1 - היתרון הגדול של סיביות אלו הוא בשימוש בתשאול אשר לא מצריך את אפשר הפסיקה אלא דגימת סיביות אלו.

TR0 - כאשר נשים '1' בסיבית זו יתחיל טיימר 0 לרוץ.

TR1 - כאשר נשים '1' בסיבית זו יתחיל טיימר 1 לרוץ.

TF0 - דגל זה עולה ל-'1' כאשר טיימר 0 מגיע לגלישה.

TF1 - דגל זה עולה ל-'1' כאשר טיימר 1 מגיע לגלישה.

#### **TMOD(Timer MODE): האוגר**

GATE	C\T	M1	M0	GATE	C\T	M1	M0
------	-----	----	----	------	-----	----	----

אוגר זה מכיל 8 סיביות : 4 ה LSB- מתייחסות לטיימר 0 ו-4 ה MSB- לטיימר 1.

T/C- אם סיבית זו ב-"0" אז הטיימר משמש כקוצב זמן שסופר עפ"י תדר המתנד של המיקרו זמן מכונה

אחד השווה ל- 1usec. אם סיבית זו ב-"1" הטיימר משמש כמונה אירועים חיצוניים בסיביות T0/T1

והוא יספור עפ"י דרבוני קצה חיוביים.

M0 ו- M1- סיביות אלו יקבעו באיזה מצב פעולה יעבוד הטיימר-

תיאור המצבים מתואר בטבלה הבאה:



MM1	MM0	תיאור מצב
00	00	טיימר 13 סיביות
00	11	מונה/טיימר 16 סיביות
11	00	טעינה אוטומטית
11	11	מיועד לקביעת BAUD

**מונה 13 סיביות** - בצורת עבודה זו הטיימר מונה בגודל 13 סיביות כלומר כל TL וחמש סיביות מ TH - כאשר שלושת הסיביות אינן משפיעות.

אופן פעולה זה נוצר כדי ליצור התאמה בין רכיבי MCS51 לרכיבי משפחת MCS48 האיטיים פי 32 ממשפחת MCS51.

**מונה 16 סיביות** - בתצורת עבודה זו הטיימרים מתפקדים הן כטיימרים והן כמונים בעלי 16 סיביות כל אחד.

**מונה 8 סיביות טעינה אוטומטית** - משמש בעיקר ליצירת קצב שידור וקליטה ביחידת ה UART. במצב זה הטיימר מקדם את TL מהערך שנמצא ב-TH וכאשר מגיע למקסימום הערך שב-TH נכנס בצורה אוטומטית ל TL והספירה ממשיכה.

קביעת ה BAUD - מצב זה נועד לקביעת קצב השידור/קליטה של סיביות ביחידת ה- UART לצורך פעולתו, דורש ה UART גל ריבועי מחזורי אשר יקבע את קצב העברת הנתונים.

\*אנו נשתמש ב TIMER1 - שיציאתו מחוברת ל UART - כיציאת שעון.

את אופן פעולת ה- TIMER1 נקבע למצב של טעינה אוטומטית ונטען לתוכו ערך ראשוני אשר יקבע על פי הנוסחה הבאה:

$$Baud\ Rate = \frac{F_{OSC} \cdot 2^{SMOD}}{2 \cdot 12 \cdot 16 \cdot (256 - TH_1)}$$

## האוגר – SCON(Serial Control) אוגר בקרת התקשורת הטורית:

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
-----	-----	-----	-----	-----	-----	----	----

תפקיד אוגר זה הוא לקבוע את תצורת העבודה של יחידת התקשורת הטורית.

RI - סיבית המציינת כי אוגר-SBUF מלא וכי נקלטה מילה.

סיבית זו עולה ל-1' אוטומטית ומציינת פסיקת UART קליטה.

על המתכנת להוריד סיבית זו בתוכנה על מנת לאפשר קליטת בתים נוספים.

TI - מציינת כי אוגר-SBUF ריק ושודר BYTE

סיבית זו עולה ל-1' אוטומטית ומציינת פסיקת UART שידור.

על המתכנת להוריד סיבית זו בתוכנה על מנת לאפשר שידור בתים נוספים.

RB8 - סיבית תשיעית נקלטת, שימושית למטרת בקרת שגיאות.

TB8 - סיבית תשיעית משודרת (בדרך כלל שימושית לזוגיות) ערכה נקבע ע"י מערכת התוכנה,

שימושית לבקרת שגיאות.

במערכת רבת מעבדים מאפשרת למקלט לברור את תווי המידע הנקלטים ע"י סיבית זו שתשודר

בסיום כל תו.

SM2 - סיבית המיועדת לתקשורת רבת מעבדים.

מאפשרת למקלט לברור את תווי המידע הנקלטים ע"י הסיבית התשיעית של כל תו.

כאשר סיבית זו ב-1' המיקרו יתייחס לתו הנקלט בתנאי שהסיבית התשיעית 1'.

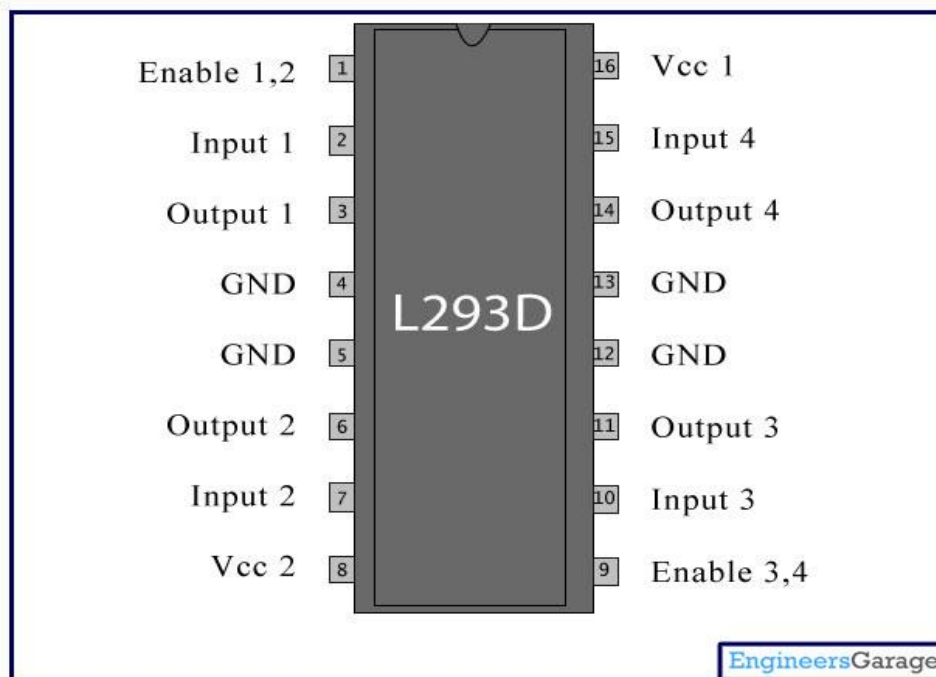
כאשר סיבית זו ב-0' המיקרו יתייחס לכל תו נקלט ללא התייחסות לסיבית ה-9.

SM0 , SM1-

SM0	SM1	MODE
0	0	Shift register (fosc/12)
0	1	8 bit uart קצב משתנה
1	0	9 bit uart (fosc/64)
1	1	9 bit uart קצב משתנה

FOS - תדר הגביש המחובר למיקרו.

## 9.2 דרייבר דוחף זרם למנוע L293D



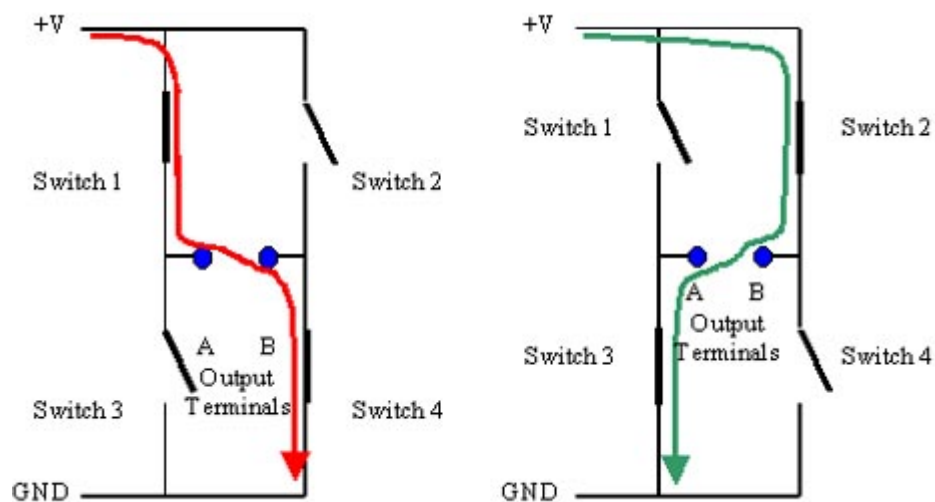
הדרייבר הוא רכיב דוחף זרם למנועים. דרייבר זה תואם למנועי AC ו DC. נשלט ע"י יחידת מיתוג במבואות שלו (כמו מיקרו-בקר). הדרייבר מהווה "H-BRIDGE", לכן מאפשר לשנות כיוון סיבובי מנועים ע"י מיתוג הכניסות שלו.

הדרייבר מסוגל לספק עד 600mA ביציאות ומתח בתחומים: 4.5V to 36V. דרייבר זה עובד בצורה של דרייברים כפולים, כלומר 4 יציאות עבור 2 מנועים. הרכיב עובד בשיטת ייצוג מתחים TTL במבואו ומגביר זרם במוצאו ע"י חיבור טרנזיסטורים כמגברי זרם. במצב כזה – זרם מוצא של קולט (collector) של טרנזיסטור הקודם נכנס להדק בסיס (BASE) של טרנזיסטור הבא. וכך הלאה בחיבור קסקדה. נוצרת הגברת זרם לפי:  $IC = b \cdot ib$ .

## הדקים:

- הדקים IN1-4 – כניסות מיתוג לוגי.
- הדקים OUT 1-4 - יציאות רכיב.
- ENABLE1,2 – הדקים לאפשר המוצאים (OUT1,2).
- ENABLE 3,4 - הדקים לאפשר המוצאים (OUT3,4).
- VCC - מתח הספקה .
- -VCC2 מתח מוצא למנועים.
- -GND – אדמת רכיב.

H-BRIDGE כולל 4 מפסקים (טרנזיסטורים) שמחוברים בצורת H כאשר המנוע במרכז ה-H.



כשמפסקים 1-4 סגורים A מחובר להדק החיובי ו- B מחובר להדק השלילי

כשמפסקים 2-3 סגורים A מחובר להדק השלילי ו- B מחובר להדק החיובי. כתוצאה מכך המנוע נע בכיוון ההפוך.

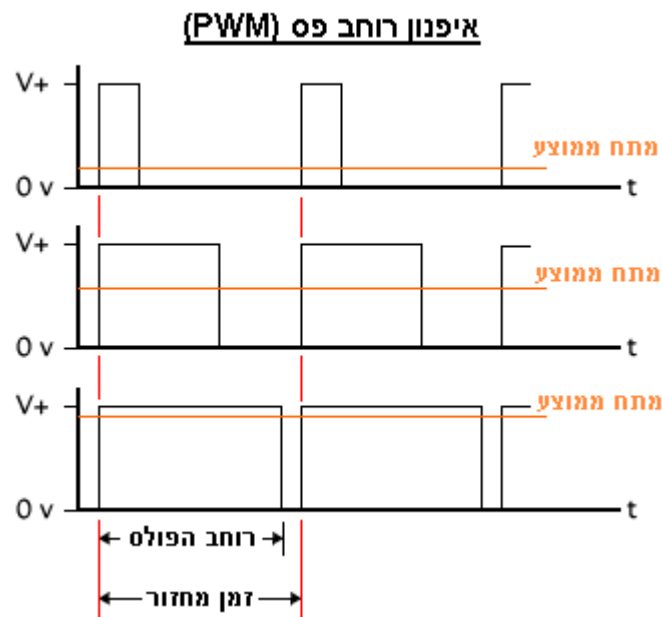
### 9.3 אפנון PWM (אפנון רוחב הפולס)

בבקרת PWM משתמשים בתהליך בו המתח יכול לקבל שני ערכים: 0 (אפס) או  $V+$ . תהליך מיתוג המתח הוא תהליך מחזורי, שבו מסומן זמן המחזור ב-T. כל מחזור מחולק לשניים בחלק אחד, המתמשך  $T_{off}$  שניות - המתח המסופק הוא 0 (אפס), וביתרת המחזור, במשך  $T_{on} = T - T_{off}$  - המתח הוא  $V+$ .

זמן המחזור של כל גל מסומן ב-T, משך הזמן בו נמצא הגל במצב מתח On מסומן ב-  $T_{on}$ , ומשך הזמן בו נמצא הגל במצב מתח Off מסומן ב-  $T_{off}$ .

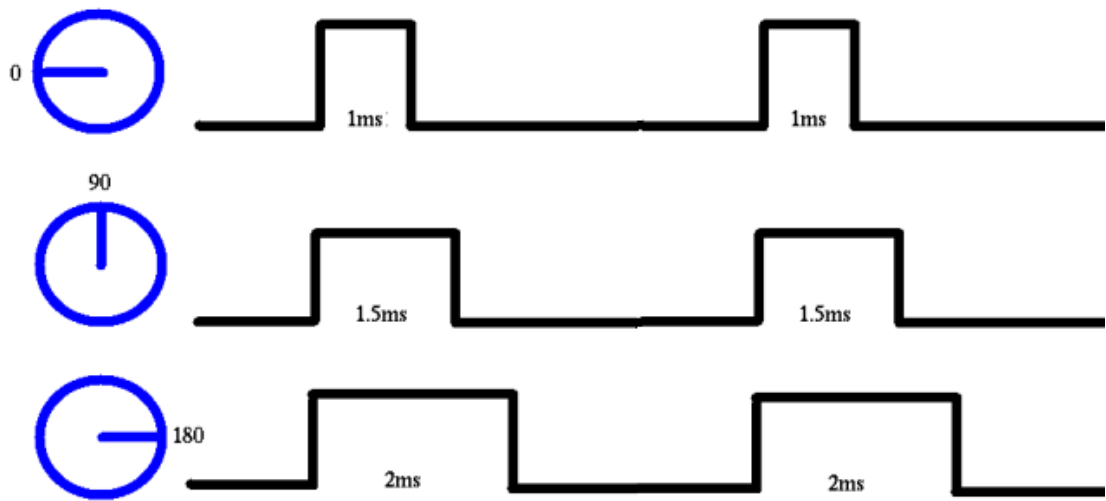
הבקרה שאיתה אנו משתמשים במקרה של הפרויקט שלנו היא בקרה על זווית מנועי הסרבו בתוך מנוע הסרבו ישנה בקרה אשר יודעת באיזה זווית נמצא עכשיו המנוע ועל ידי נתינת פולס של PWM המנוע יודע לאיזו זווית רצויה הוא צריך להגיע.

בשיטה זו מפעילים ומנתקים בתדירות גבוהה את מקור המתח המחובר לצרכן (נגד או מנוע); כך מתקיים תהליך שבו על הצרכן שורר מתח כתלות בזמן בעל אופי של גל ריבועי. מאפייני הפונקציה של מיתוג מקור המתח קובעים את ההספק הממוצע שמקבל הצרכן.



במקרה שלנו השימוש באיפנון PWM הוא שונה, בפרויקט שלנו איפנון PWM פועל באופן הבא: הבקרה שאיתה אנו משתמשים במקרה של הפרויקט שלנו היא בקרה על זווית מנועי הסרבו בתוך מנוע הסרבו ישנה בקרה אשר יודעת באיזה זווית נמצא עכשיו המנוע ועל ידי נתינת פולס של PWM המנוע יודע לאיזו זווית רצויה הוא צריך להגיע.

בנוסף הזווית המקסימלית שאליה מגיע מנוע הסרבו בזמן של 2ms היא 120 מעלות.



#### **9.4 מד מרחק (SR04)**

חיישן מרחק אולטרה סוני משדר וקולט גל בתחום אולטרה סוני ( מעבר לתדר השמיעה) , בתדירות KHZ 40. החיישן משדר פולס בפרק זמן של 8 מחזורים בתדר KHZ 40 וממתין לקבלת הד חוזר. תפקידו לגלות מרחק של גופים ממעגל המשדר- מקלט.

#### **מבנה פיזי**



#### **מאפיינים**

- מתח ספק - 5 וולט
- זרם - 15 mA אופייני.
- תדירות - 40KHz .
- טווח מקסימאלי - 4 מטר.
- טווח מינימאלי - 2 ס"מ .
- רגישות - גילוי בקוטר 3 ס"מ עד מרחק גדול מ 2 מטר.
- פולס התנעה - פולס של מינימום 10 מיקרו שניות ברמת מתח TTL .
- פולס הד - אות TTL חיובי ברוחב התלוי בטווח.
- מידות קטנות - 45mm\*20mm\*15mm

## עקרון המדידה

גל הקול מתפשט בחלל פוגע בעצם וחוזר למקלט, כלומר מבצע דרך השווה לפי 2 מהמרחק של העצם מהחיישן. מהירות התפשטות גל הקול שווה למהירות הקול לכן הזמן שלוקח לגל הקול מרגע השידור עד לחזרתו הוא יחסי ליניארי למרחק של העצם מהחיישן. בפרויקט אנו מודדים את הזמן ובאמצעותו מציגים את המרחק.

מהירות הקול תלויה בתווך בו עובר הקול ובלחץ. בגובה פני הים מהירות הקול היא 1200 ק"מ/שעה שהם 333.33 מטר לשנייה.

ערך המרה K	V (מהירות)	טמפרטורה C °t
60	331.5	0
59	337.5	10
58	343.5	20
57	349.5	30
56	355.5	40
55	361.5	50

בחרנו לעבוד עם ערך המרה K=58 כאשר טמפרטורת העבודה שלנו היא 20 מעלות.

אנו מעוניינים לקבל את המרחק מהחיישן ביחידות CM ולכן יש להמיר את המהירות ל CM/US, להלן הדרך:

$$v = 340m / s$$

$$v = 340 * 100cm = 34000cm / s$$

$$1us = \frac{1sec}{1000000}$$

$$v(cm / us) = \frac{34000}{1000000} = 0.034cm / us$$

$$k = \frac{2}{v(cm / us)} = \frac{2}{0.034} = 58.823$$

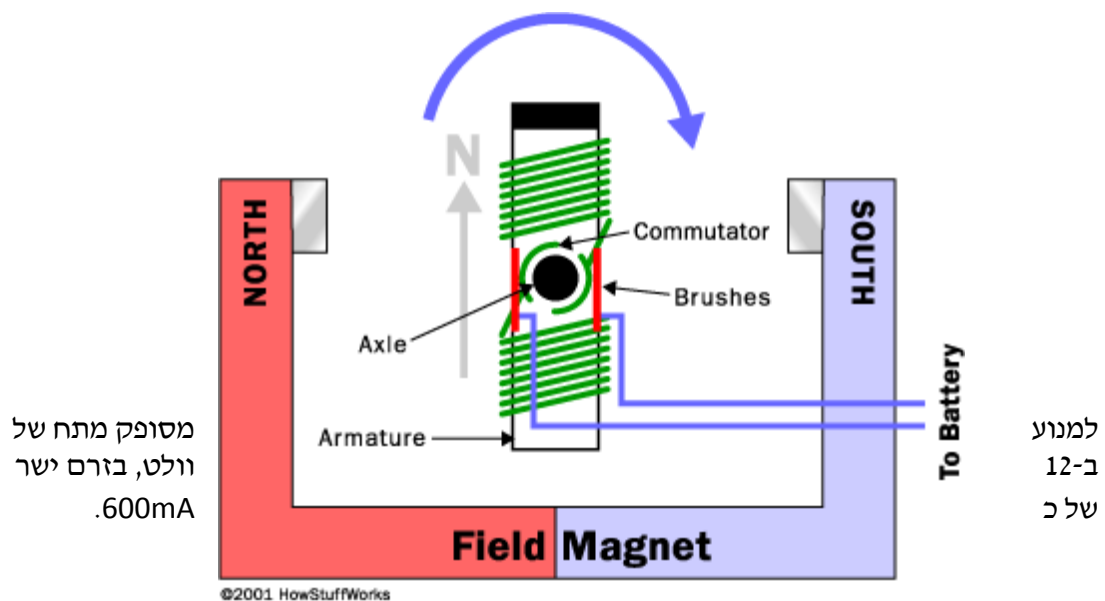


### :DC Motor 9.5

מנוע חשמלי הוא מכונה הממירה אנרגיה חשמלי למכנית. מנוע חשמלי מבוסס על עיקרון האלקטרומגנטיות. המאפשר יצירת שדה מגנטי על ידי העברת זרם חשמלי דרך סליל.

המנוע החשמלי בנוי על פי רוב משני חלקים עיקריים:

- (1) **סטטור (Stator)** מערכת סלילים המלופפים סביב ליבה פרוֹמגנטית (Ferromagnetic Core) המקובעת למקומה. הסטטור יכול להיות מורכב גם משני מגנטים רבי עוצמה. המגנטים מסודרים כך שקוטביהם (צפון ודרום) המופנים לכיוון הרוטור מנוגדים.
- (2) **רוטור (Rotor)** ציר העובר בתוך הסטטור ועליו מלופפים שלושה סלילים. ציר זה חופשי להסתובב. כאשר זרם חשמלי דרך הסלילים שברוטור, נוצר שדה מגנטי סביבם (דרך הליבה). שדה מגנטי זה מפעיל כוח על הציר העובר דרכו, וזה מסתובב עקב המומנט (כח סיבובי). העברת זרם חשמלי מקוטע, בצורה מבוקרת, מאפשרת צירוף תנועות זוויתיות קטנות לסיבובים שלמים.



### 9.6 מנועי סרבו (SERVO) :

מנוע סרבו הוא מנוע זרם ישר (DC Motor) בעל מערכת תמסורת פנימית של גלגלי שיניים ובקרה אלקטרונית על מיקום המנוע. מה שמיחד מנועי סרבו היא העובדה שהם אינם מסתובבים בצורה חופשית כמו מנועי DC, אלא נעים על פי זווית – לרוב בין 0 ל-180 מעלות.

מנועי סרבו פועלים בחוג סגור, כלומר הינם בעלי בקרה על מיקום המנוע, ובעלי יכולת תיקון פערים מהמיקום הרצוי.

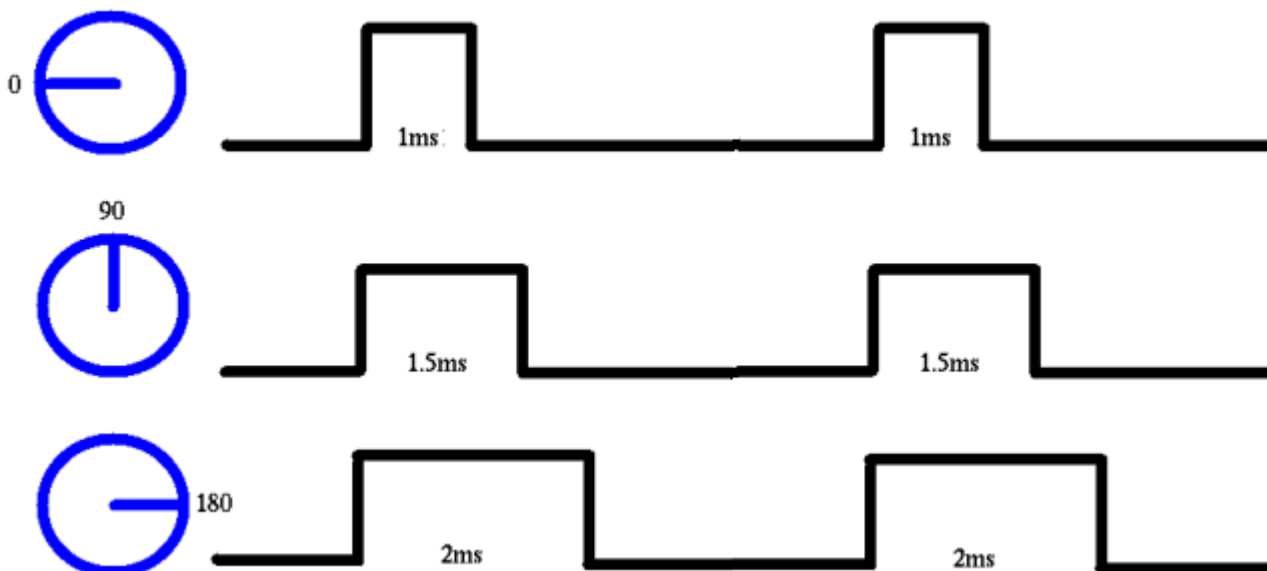
שליטה במנועי סרבו מבוצעת על ידי שליחת אות דיגיטאלי אל חוט הבקרה של המנוע. הרעיון הכללי הוא שליחת גל מרובע (Square Wave) אל המנוע, כאשר **אורך הגל** הוא זה שקובע את הזווית אליה ינוע המנוע.

לדוגמה, כאשר נספק למנוע גל בו רוחב הפולס הוא 1 מילי-שנייה, המנוע ינוע אל זוויתו המינימאלית – 0 מעלות.

כאשר נספק למנוע גל בו רוחב הפולס הוא 1.5 מילי-שנייה, המנוע ינוע אל זוויתו האמצעית – 90 מעלות.

כאשר נספק למנוע גל בו רוחב הפולס הוא 2 מילי-שנייה, המנוע ינוע אל זוויתו הגדולה ביותר – 180 מעלות.

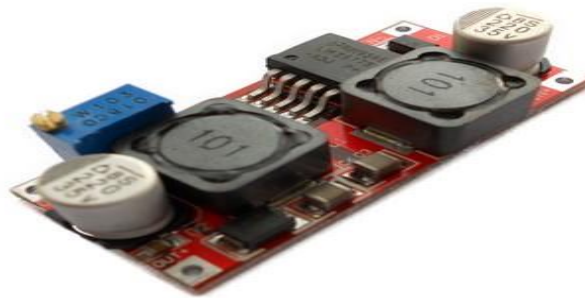
תרשים סכמטי :



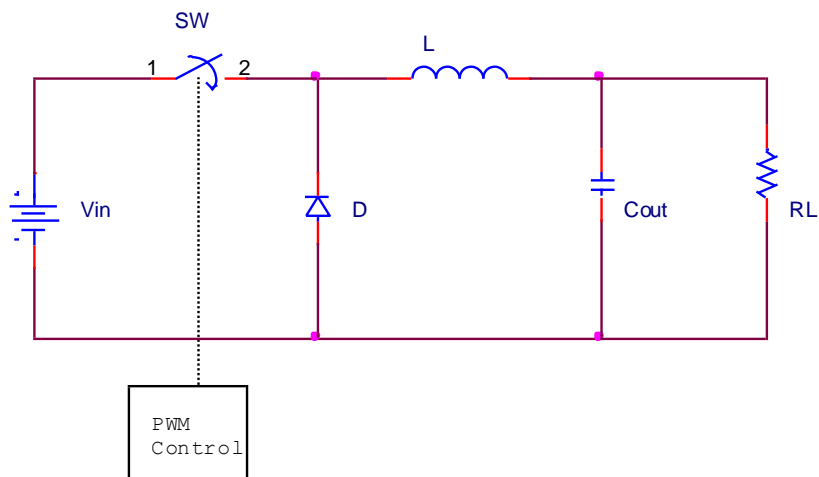
בכרטיס העבודה ישנם 2 מנועים כאלו שנותבי המיד שלהם מחוברים אל P0\_0,P0\_1 הם מרכיבים בעצם את הזרוע המכנית.

במקרה שלנו מנוע הסרבו בזמן 2ms הזווית היא 120 מעלות.

### 9.7 מייצב ממותג מסוג BUCK (Step Down)



ממיר BUCK הוא ממיר המיועד להוריד מתח. ממיר זה אינו מבודד, כלומר אין הפרדה חשמלית בין מתח הכניסה ומתח היציאה. לאחרונה פותחו לממיר זה מספר שיטות המאפשרות נצילות גבוהה מ-90%, על-ידי שימוש במתג סינכרוני במקום הדיודה להורדת הפסדי הולכה, ומיתוג בזרם אפס או מתח אפס כך שאין הפסדי מיתוג.

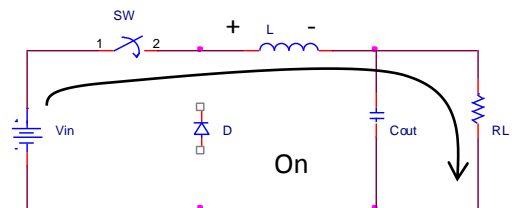


### למעגל שני מצבים:

1. כאשר המתג SW במצב ON, הסליל L מעביר מתח למוצא ונטען ממתח הכניסה.
2. כאשר המתג SW במצב OFF, הסליל L מעביר מתח למוצא ומתפרק דרך נגד העומס.

$$V_L = L \frac{dI_L}{dt}$$

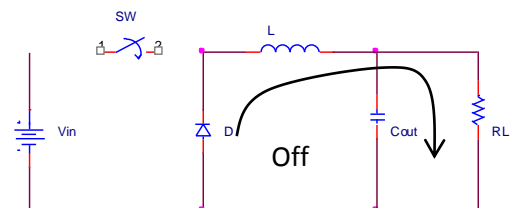
### המתג סגור (Ton):



שיפוע הזרם חיובי ולכן במצב זה המתח על פני הסליל:

$$V_L = V_{in} - V_{out}$$

### המתג פתוח (Toff):



שיפוע הזרם שלילי ולכן במצב זה המתח על פני הסליל:

$$V_L = V_{out}$$

כאשר הממיר במצב יציב, ממוצע המתח על פני הסליל חייב להיות אפס ולכן מתקיים הקשר:

$$(V_{in} - V_{out})t_{on} = -(0 - V_{out})t_{off}$$

## פונקצית התמסורת:

$$V_{out}/V_{in} = t_{on}/(t_{on}+t_{off})=D_{on}$$

צורות גלים:

