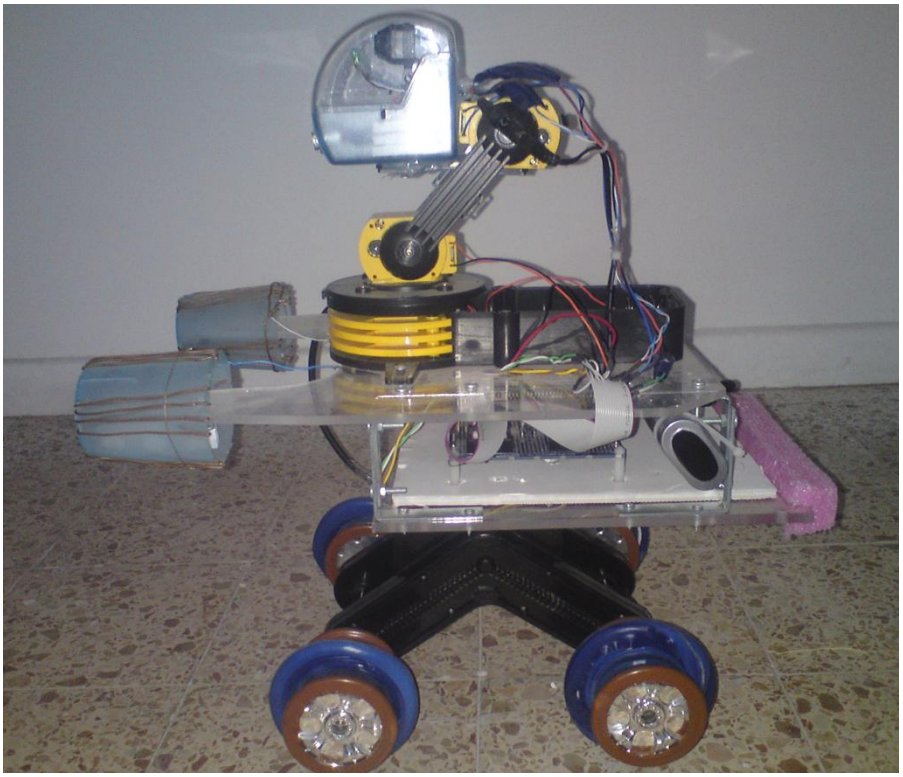
פרוייקט גמר - הנדסאים



יחידת חובט סיור אלחוטי

ספר זה מוגש כעבודת גמר לקבלת <u>תעודת הנדסאי</u> במגמת: <u>אלקטרוניקה</u>

מנחה: יוסי חזן

מגיש: אמנון אברמוב

חתימת ראש מגמה : מר שיפרנסקי מיכאל

חתימת מנחה פרויקט: מר יוסי חזן

תוכן העניינים :

מבוא :

"רובוט סיור אלחוטי" אבטיפוס, זהו רובוט סיור ביתי הנשלט דרך המחשב האישי באמצעות תוכנה יעודית.

הרובוט פועל על בסיס מעבד P89V51RD2 של פיליפס ממשפחת 8051 המעבד התעשייתי הנפוץ שמובנה עפ"י הארכיטקטורת המעבדים 8086 ו 8088 של אינטל.

בנוסף לבקר יש קיטים נוספים נילווים כמו מעגל גביש שאחראי על תזמון או "שעון" למעבד ומעגל הריסט האחראי לאיתחולו.

כמו כן הרכבנו לבקר תצוגת LCD שתי שורות על"מ שברצוננו שכל פקודה שהרובוט מקבל תוצג.

ברובוט מיושמים עקרונות מכנאיים שנלקחו בחשבון טרם בנייתו.

גוף הרובוט כלומר מבנהו הפיזי הוא לא קיט מוכן כמו שלרוב רואים אלא תוכנן באופן עצמאי מראשיתו, ברובוט מורכבת זרוע שבמפרקיה יש גיר ומנוע DC בקיבולת מתח של עד 5 וולט ו0.5 אמפר מעבר לכך יגרם נזק הן למפרק הזרוע והן למנוע, על הזרוע מולבשת מצלמת RF אלחוטית המשדרת ווידאו ללא קול למחשב באמצעות מקלט, המצלמה משדרת בתדר של 2.4 ghz .

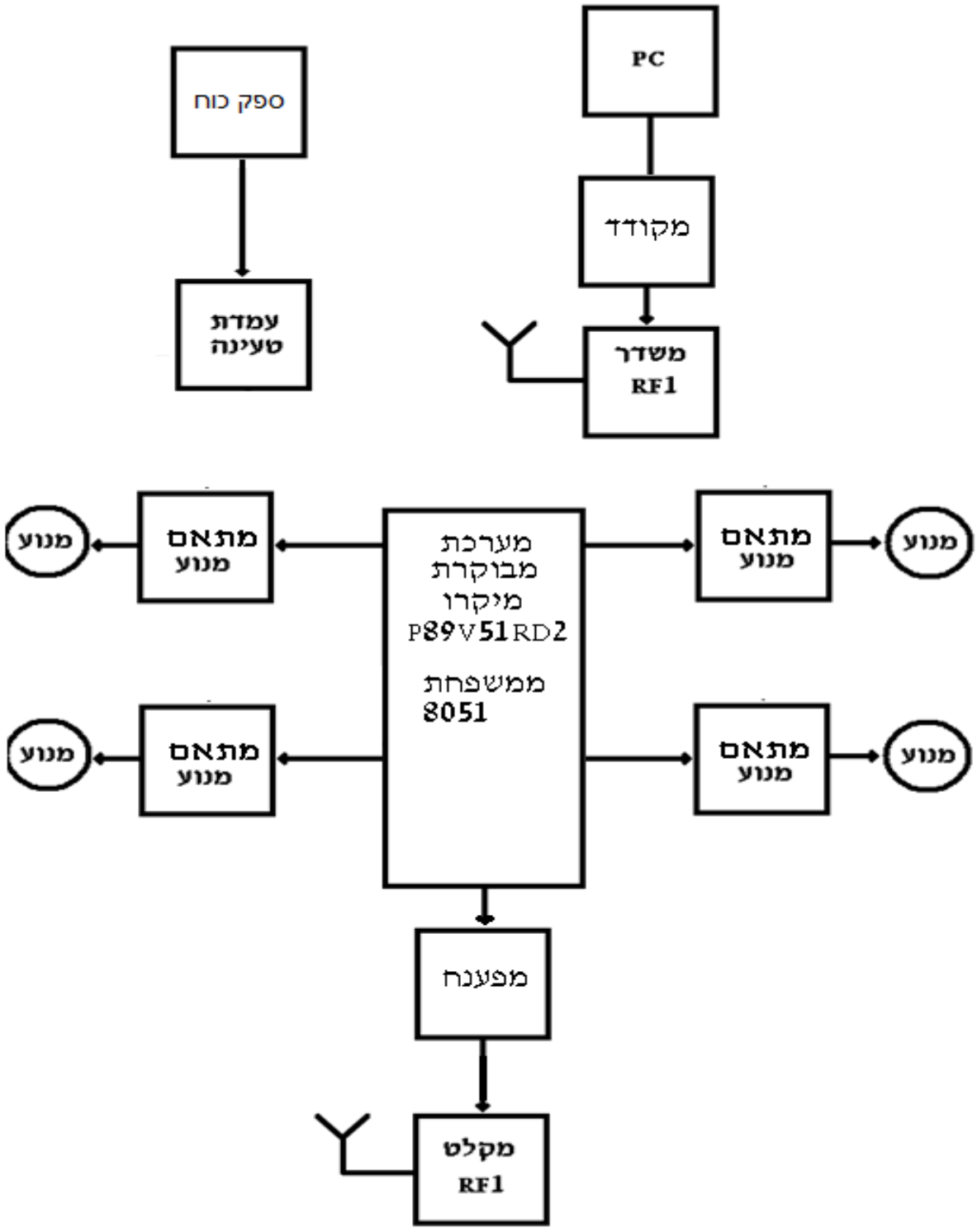
ניתן להקליט וידאו והמצלמה מותקנת על זרוע על"מ למנוע שטחים מחוץ לטווח המצלמה.

הרובוט מיועד לתפקוד בבית ויעודו ביטחוני בעיקרו, תפקיד הרובוט הוא שומר וגשש ביתי בזמן שהבעלים איננו נוכח.

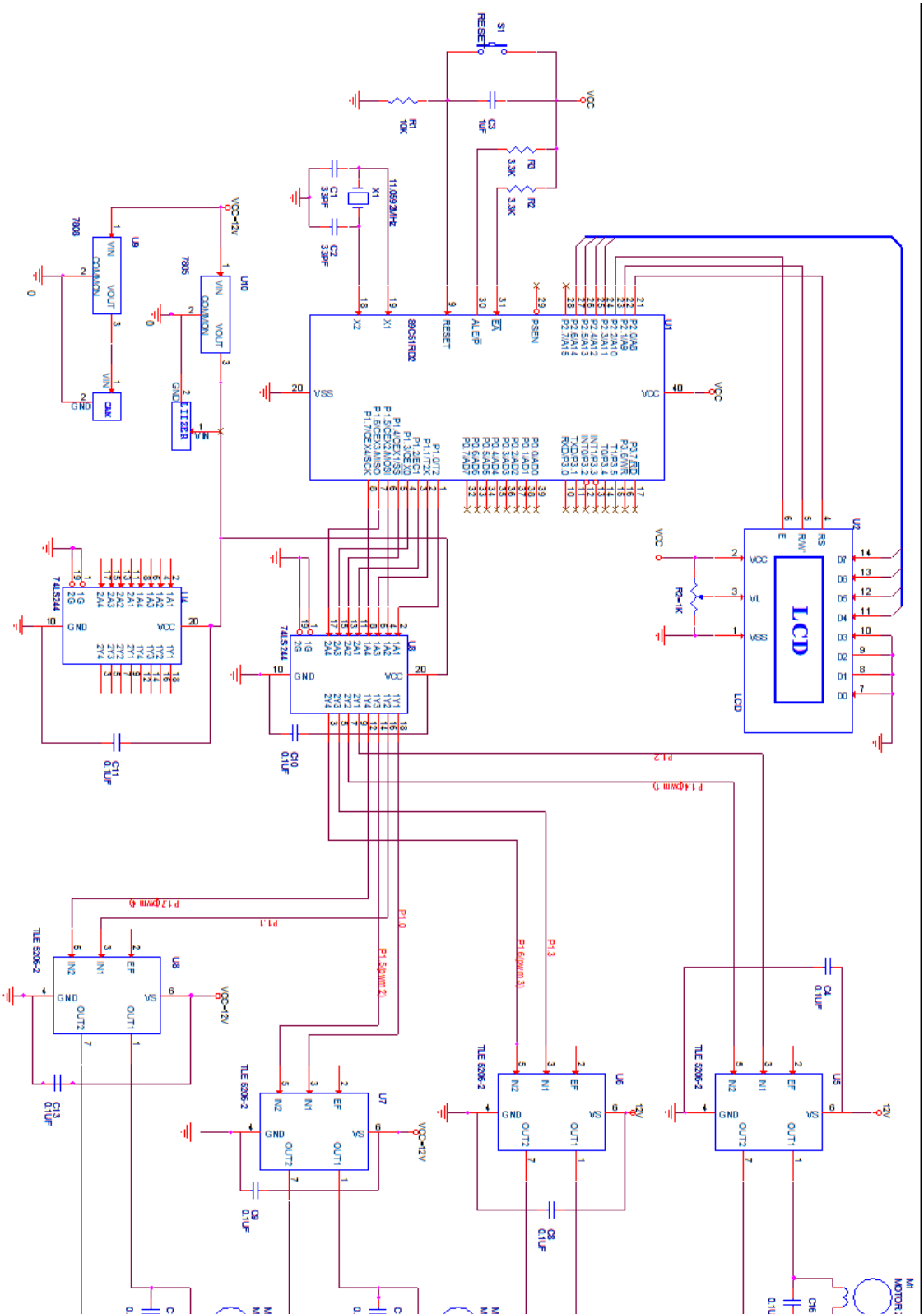
ניתן להשתלט על הרובוט דרך האינטרנט על ידי תוכנת השתלטות מרחוק על המחשב.

בסיס הרובוט כלומר ההרכב בצורת V הפוכה היכן שממוקמים מנועי ההנעה החזקים באופן יחסי של הרובוט, בכל צד המחובר מנוע לתמסורת של גלגלי שיניים המניעים שני גלגלים בכל צד בו זמנית בצורה זו כלומר כאשר זוג גלגלים נע בכיוון אחד וזוג גלגלים נע בכיוון ההפוך הרובוט מסתובב סביב צירו.

תרשים מלבנים :



שרטוט חשמלי מעגל הרובוט :



הסבר שרטוט חשמלי מעגל הרובוט :

- המיקרו בקר P89V51RD2 - זהו ה"מוח" שמחוברים אילו כל התקני הקלט/פלט, בנוסף התוכנית של הרובוט צרובה בו .
- מעגל ה- RESET -תפקידו הוא להחזיר את מצביע הכתובות לכתובת ההתחלתית של התוכנית בכדי שהתוכנית תעבוד מהתחלה בצורה תקינה .
- מעגל הגביש – תפקידו להפעיל את המיקרו בקר ובעקבותיו את המערכת כולה , הוא מורכב מגביש קוורץ שתדירותו 11.0592Mhz.
- מעגל המייצב מתח 7808 ו 7805 והספק כוח- תפקידו של הספק כוח הוא לספק למערכת כולה מתח וזרם , תפקידו של המייצב מתח הוא לייצב לנו את המתח שמתקבל מהספק כוח למתח של 5V .
- תצוגת LCD – תפקידה להציג על המסך את הנתונים שנקלטים מהתקני הקלט ולהציג לנו את הנתונים ,על ידי כך אנו יכולים לאתר שגיאות כמו כן אנו יכולים להציג כיוונים של תנועת הרובוט באמצעות התצוגה .
- מנועי DC – תפקידם הוא לאפשר לרובוט לנוע.
- דוחף זרם TLE 5206-2 –תפקידו להגדיל את הזרם למנועי DC על מנת שינועו .
- חוצץ 74LS244- תפקידו לבצע חציצה בין המיקור בקר לרכיבי הפריפריה על מנת להגן על המיקרו בקר .
- קבל לוחות (כחול) – הקבל הפשוט ביותר למימוש מורכב משני לוחות מוליכים מקבילים, שביניהם חומר מבודד אגירת המטען נעשית על ידי כך שעל שני לוחות הקבל נאספים מטענים - מטען חיובי q בלוח אחד ומטען שלילי q- בלוח שני. המטענים אינם יכולים לעבור מלוח אחד למשנהו משום שביניהם מפריד חומר מבודד. הפרש המטען הזה יוצר שדה חשמלי וכן מתח חשמלי V, הנמדד בין הדקי הקבל. קיבול של קבל לוחות הוא C=q/V.
- נגדים –הם רכיב חשמלי שתכונתו העיקרית היא התנגדות חשמלית כאשר עובר זרם חשמלי בנגד, נוצר על פניו מפל מתח . על פי חוק אוהם.התנגדות חשמלית נמדדת ביחידות אוהם (Ω), ונהוג לסמנה באות R. בנגד ישנה המרה של אנרגיה חשמלית לחום(אנרגיה תרמית).
- LED- דיודה פולטת אור (Light emitting diode - LED) הינה התקן מוליך למחצה אשר פולט אור.
- מקלט יחידה אחת,מתוך שתיים, שתפקידה לקלוט נתונים מה משדר.

המיקרו בקר 89C51RD2 :

הגדרה-

מיקרו בקר הינו סוג של מחשב.

הבקר עצמו כולל המון פונקציות מובנות שמקבילות לפונקציות שכבר קיימות במחשב PC.

מיקרו בקר בנוי מ- זיכרון, יחידת עיבוד (CPU), מונים, קוצבי זמן, התקני כניסות והתקני יציאות.

מיקרו בקרים נפוצים במערכות המבצעות מספר רב של חישובים, מערכות מורכבות.

פעולות אלו מתבצעות בעזרת התוכנה במקום ברכיבי חומרה שונים.

אם נסתכל היום כל התחום של מיקרו בקרים בתנופה, חברות מובילות מפתחות כל הזמן מיקרו

בקרים שיתנו מענה לדרישות המשתנות של הצרכנים השונים בשוק.

MC51– מיקרו בקרים, הקדמה כללית:

Micro controller- (מיקרו בקר) הינו בעצם מחשב הבנוי על פיסת סיליקון.

מיקרו בקר- 8051 פותח בשנות השמונים ע"י חברת אינטל. מיקרו בקר זה כולל רכיב זיכרון קריאה או כתיבה מסוג RAM בגודל 256 בתים ורכיב זיכרון לקריאה בלבד מסוג ROM שגודלו עד 64k בתים.

המיקרו בקר מכיל בנוסף עוד מספר דברים, מונים ברי תכונות, יחידת תקשורת טורית,פורטים וכו'.

מיקרו בקר 8051 הוא הראשון שהציג שיטת מיעון סיבית- גישה לסיביות.

שיטה זו מאפשרת גישה לכלל המידע מהסיבית ה32 בזיכרון החיצוני באופן פרטני של הרמת סיביות או הורדתן.

MC51 מביאה גישה שונה לפתרון בעיות בעזרת מחשב.

כשמתכננים מעגל שבנוי על מיקרון בקר זה, בעצם מכוונים למחשב בעל עוצמה גדולה שכולל פס נתונים רחב ופס כתובת גדול עם חיבור של זיכרון בעל קיבולת גדולה.

מיקרו בקר היא יחידת עיבוד מרכזית (יע"מ) הכוללת משאבים נוספים כגון זיכרון RAM,ROM, ממשקים לקלט ופלט, מונים קוצבי זמן, משדרים ומקלטים אסינכרוניים לתקשורת טורית.

בבקרים מתקדמים לעיתים יש גם- A/D ו-D/A וזה מאפשר לבנות מערכת מבוקרת מחשב עם מינימום רכיבים.

ההבדל בין הרכיבים השונים ששיכים למשפחת הבקר הנזכר הוא בעיקר בכמות הזיכרון ובכמות הטיימרים ברכיבים השונים. רוב הרכיבים בו עובדים עם גביש בתדר של MHZ 12.

<u>מיקרו בקר 89C51RD2 :</u>

מיקרו בקר זה הינו נגזרת מודרנית מה- 8051 עליו הרחבנו בנ"ל. לבקר זה יש זיכרון EPROM פנימי לא נדיף מסוג FLASH. המחיקה והצריבה נעשית כשהבקר נמצא על המעגל ע"י מילות בקרה בעזרת תוכנה ייעודית בתקשורת טורית. בנוסף לכך, גודל זיכרון ה- RAM בבקר זה גדול יותר מהבקר- 8051.

הזיכרון הפנימי של המיקרו בקר הנ"ל יכול להכיל את כל המשתנים והתוכנה של הפרויקט.

הפנייה למשתנים המוגדרים כבר בזיכרון הפנימי יעילה ונוחה יותר ואין צורך לחבר זיכרון חיצוני כדי לענות על הבקשות השונות.

במידה ובפרויקט מסוים גודל בקר זה לא יספיק להכיל את הנדרש בתוכנה ניתן להחליפו לבקר בעל זיכרון פנימי גדול יותר מבלי לשנות את התוכנה ומבלי להוסיף זיכרון חיצוני לבקר.

הגרסאות השונות של ה-8051 נמצאות בשימוש עד היום במכשירים רבים ולמעשה זהו המחשב הנפוץ בעולם- הוא נמצא בשימוש במקררים, מכונות כביסה, רמזורים, מיקרוגל וכמעט בכל מוצר אלקטרוניקה שנמכר היום בגלל פשטותו וגמישותו. עבור מכשירים שונים מיוצרים בדרך כלל רכיבים ייחודיים למוצר שהתוכנה הנדרשת מוטבעת בזיכרון הקבוע (ROM) שלהם וכך מוזילה את עלויות היצור.

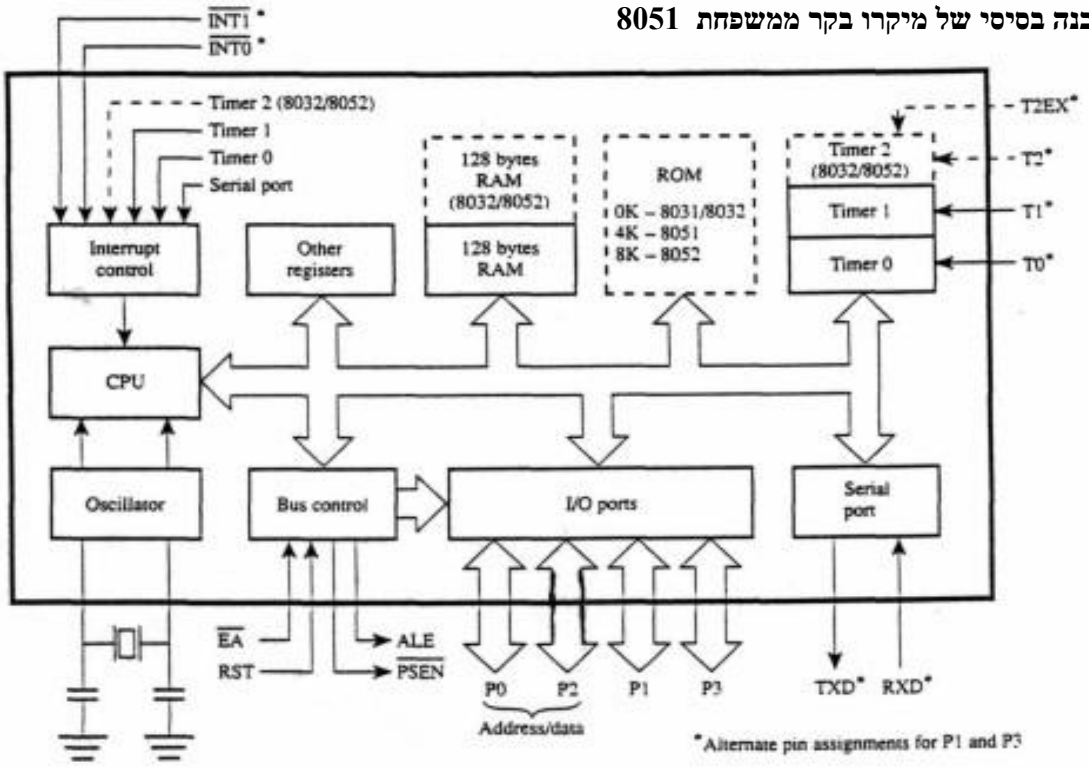
השיקולים בבחירת מעבד זה בפרויקט-

בפרויקט שלנו אנו השתמשנו בו מכוון שיש במיקרו בקר יחידות חדשות/משופרות שאנו נעזרים בהם יחידות אלו הם :

- זיכרון ה- ROM הוא על מנת לכתוב את התוכנית עליו ולאחר מכן רק לקרוא ממנו בבקר שהשתמשנו בפרויקט יש 64KB פלאש ממורי (המאפשר כתיבה מחיקה וכתיבה שוב לזיכרון ה– ROM מה שאין במיקרו בקרים ישנים יותר (8031) כמו כן התוכנית נשארת גם לאחר ניתוק מקור החשמל .
  זיכרון ה-ROM בנוי כך שכאשר נאתחל את המיקרו בקר התוכנית תתחיל לפעול מהכתובת התחלתית ובעצם תעבוד על פי התוכנית(ולא תתחיל מאיזה כתובת אקראית).
- יחידת PCA- בבקר זה בשונה מהבקרים הישנים יותר מאותה המשפחה ישנם חמש יציאות של PWM שכאשר אנו רוצים לעבוד עם PWM אנו יכולים כאשר אין זה תופס לנו את המעבד כלומר הבקר לא עסוק כל הזמן לייצר לנו גל ריבועי אין סופי (בכדי לייצר גל ריבועי אין סופי צריך לכתוב זאת בתוכנה ולהכניס ללולאה אין סופית).

- במהלך שנות לימודי ההנדסאים אנו למדנו על בקר זה על כן אנו יודעים ומכירים את הבקר בצורה טובה הן בכתיבת התוכנה לבקר והן בחומרה של הבקר .

מבנה בסיסי של מיקרו בקר ממשפחת 8051



ישנם שני שיטות לעבודה של המיקרו בקר עם התקני הקלט שיטת הסריקה-POLLING ושיטת הפסיקה-INTERRUPT או שניהם.

שיטת הסריקה-POLLING- בשיטה זו המיקרו בקר סורק כל הזמן את כל התקני הקלט כלומר הוא עובר התקן התקן ומחכה לשינוי

ושיטת הפסיקה-INTERRUPT – בשיטה זו המיקרו בקר נמצא כל הזמן או בביצוע פעולה או במנוחה כאשר הוא מקבל INTERRUPT הוא יכול להתייחס אילו ולקלוט את המידע (על פי הגדרות).

דוגמה לשני השיטות זה כמו מורה בכיתה כאשר המורה עובר כל תלמיד ובודק אותו (סריקה) או כאשר המורה מנהל שעור ואז ברגע שיש שאלה התלמיד מרים את היד ושואל (INTERRUPT).

בפרויקט אנו השתמשנו בשיטת הסריקה .

<u>מאפייני הבקר-</u>

- יחידת עיבוד מרכזית (CPU) של 80C51
- זיכרון FLASH של 64KB
- תדירות של עד 33MHz
- זיכרון RAM פנימי של 1024 בייט
- 6 מקורות לפסיקות
- 4 פורטים של קלט ופלט
- דרכים שונות לשליטה בעוצמה:
  - לעורר את הבקר מעוצמה נמוכה ע"י פסיקה חיצונית.
    (כלומר על ידי פסיקה להפעיל את הבקר)
  - הנמכת העוצמה המסופקת לבקר
  - מצב סרק (הפסקת מעבר אנרגיה לבקר)
  - ניתן לעצור את השעון או לחדש אותו
- פרוטוקול תקשורת אסינכרונית מסוג UART
  - גילוי כתובת אוטומטי
  - גילוי טעויות מובנה

המבנה של המעבדים ממשפחת 8051 מבוסס על ארכיטקטורת פון נוימן (פיתח פצצת אטום ופיתח מחשב אלקטרוני) וכולל את כל ארבעת המרכיבים שיש במחשב- זיכרון פנימי, פסים להעברת מידע, יחידת עיבוד מרכזית ומפתח קלט ופלט.

<u>מיקרו בקר הינו בעצם מעגל הכולל:</u>

- מעגל איפוס- reset
- זיכרונות RAM ו ROM פנימיים
- פסיקות- interrupt
- ALU- יחידה לוגית אוטומטית
- טיימרים ומונים
- פורטים של כניסה/יציאה מקבילים
- מעגל אקטיבי ליצירת אות שעון
- BUS- פסים
- CPU- יחידת עיבוד מרכזית

<u>הזיכרון במיקרו בקר- 89C51RD2:</u>

- זיכרון RAM – זהו זיכרון קטן בגודל 1024byte המשמש לשמירת נתונים שמשתנים תוך כדי הרצת התוכנית. כל הנתונים המוגדרים בתוכנית נשמרים בזיכרון RAM. הזיכרון עצמו נמחק לאחר ניתוק של מתח ההזנה.
- זיכרון FLASH- זהו זיכרון גדול ביותר (64KB) המשמש לשמירת תוכניות ונתונים. אפשר להעביר תוכנית לזיכרון זה ממחשב PC בתקשורת טורית. כדי להעביר תוכנית לזיכרון זה בבקר הנ"ל יש לחבר רגל SPEN של המיקרו ולאפס את המעגל (בעזרת הRESET) .פעולה זו מבצעת תוכנית שנמצאת בתוך הבקר. התוכנית עושה זאת בעזרת הפעלת תקשורת טורית, היא קובעת את קצב התקשורת ומעבירה נתונים ממחשב PC לזיכרון ה- FLASH כך שכתובת ההתחלה של התוכנית היא 0.

  פניה לזיכרון פנימי FLASH מתבצעת לאחר איפוס המערכת בגלל שרגל EA מחוברת ל- VCC.
- זיכרון EPROM- זיכרון גדול בעל גודל של 4KB. זיכרון זה משמש לשמירת תוכנית. הזיכרון ניתן לצריבה ע"י צורב EEPROM.

  בתהליך הצריבה הצורב בוחר כתובת פנימית בזיכרון על ידי מתן צירוף בינארי מתאים בקווי כתובת A0-A15 ואחר כך כותב סיבית של קוד התוכנית בכתובת הנבחרת בזרם גבוה.

  התוכן של EEPROM לא נמחק לאחר הפסקת מתח הזנה, ואי אפשר לכתוב לו תוך כדי התוכנית. ניתן לצרוב את EEPROM מחדש ע"י שימוש בצורב.

<u>מיפוי רגלי המיקרו בקר</u>

<u>VCC (פין 40)</u> - מתחבר להדק החיובי של ספק הכוח בן V5.

<u>VSS (פין 20)</u> - קו האדמה של הרכיב.

<u>XTAL1,XTAL2 (פינים 18 ו- 19)</u> - רגלי חיבור לגביש. לקווים אלו מחובר גביש הקובע את תדר הפעולה של המעבד, משני צידי הגביש מחוברים 2 קבלים לאדמה לקיזוז רעשים.

<u>RST (פין 9)</u> - אתחול המעבד (RESET). עם עלייתה ל-HIGH ה-CPU מפסיק את פעולתו. עם ירידתה ל-'0' לוגי, ה-CPU משנה את מצביע ההוראות לכתובת 0000, ובכך מתחיל את התכנית של המעבד מהתחלה.

<u>EA' - EXTERNAL ADDRESS (פין 31)</u>- כניסה זו מסמנת ל- CPU האם אזור התכנית נמצא בתחום הכתובות הנמוך (0000 - FFF0) שייך ל-ROM חיצוני או ל-ROM פנימי.

רגל זו נמצאת בשימוש רק במעבדי 8051 המכילים בתוכם ROM פנימי. במעבד מסוג 8031 יש לקצר קו זה ל-GND .

ALE (פין 30) - קו יציאה הנועל את הכתובת הנמוכה ב-ADDRESS (ENABLE LATCH LATCH). עולה ל-HIGH למשך STATE אחד (2 מחזורי שעון) בכל פעם שמתבצעת פנייה לזיכרון.

PSEN' - PROGRAM SET ENABLE - (פין 29) - קו יציאה, המציין לזיכרון ה-PROGRAM שה-CPU מבקש לקרוא נתון מאזור זה, כאשר קו זה יורד ל-LOW .

P1.0-P1.7 - (פינים 1-8) פורט זה משמש כפורט מבוא או פורט מוצא בלבד - ניתן לגשת לסיביות הפורט במיעון ישיר.

P0.0-P0.7 / AD0-AD7 (פינים 32-39) - קווי פורט 0, היכולים לשמש כפורט מבוא או כפורט מוצא. במקרה של חיבור רכיבי תמיכה חיצוניים לרכיב, מתפקדים קוים אלה כקווי AD0 - AD7 ,ומעבירים את החלק הנמוך של הכתובת בשלב המחזור הראשון של מחזור המכונה ואת הנתונים בשלב השני של מחזור המכונה.

P2.0-P2.7 / A8-A15 (פינים 21-28) - קווי פורט 2, היכולים לשמש כפורט מבוא או כפורט מוצא. במקרה של חיבור רכיבי תמיכה חיצוניים לרכיב, מתפקדים קוים אלה כקווי הכתובת הגבוהים A8-A15 .

P3.0-P3.7 (פינים 10-17) - קווי פורט 3, היכולים לשמש כפורט מבוא או כפורט מוצא. בנוסף לכך יכולים קוים אלה לתפקד כקווי בקרה שונים, המתוארים להלן :

P3.0 / RXD - קו כניסה של אות המגיע בתקשורת טורית אסינכרונית ל-UART.

P3.1 / TXD - יציאה של אות היוצא בתקשורת טורית אסינכרונית מה-UART.

P3.2 / INT0 - קו פסיקה חיצונית היכול לעבוד בדרבון רמה או בדרבון קצה.

P3.3 / INT1 - קו פסיקה חיצונית היכול לעבוד בדרבון רמה או בדרבון קצה.

P3.4 / T0 - כניסת אותות ל- TIMER0 במקרה שהוא מתפקד כמונה.

P3.5 / T1 - כניסת אותות ל- TIMER1 במקרה שהוא מתפקד כמונה.

P3.6 / WR - קו יציאה המציין ליחידות החיצוניות, שה-CPU מבקש לכתוב נתון.

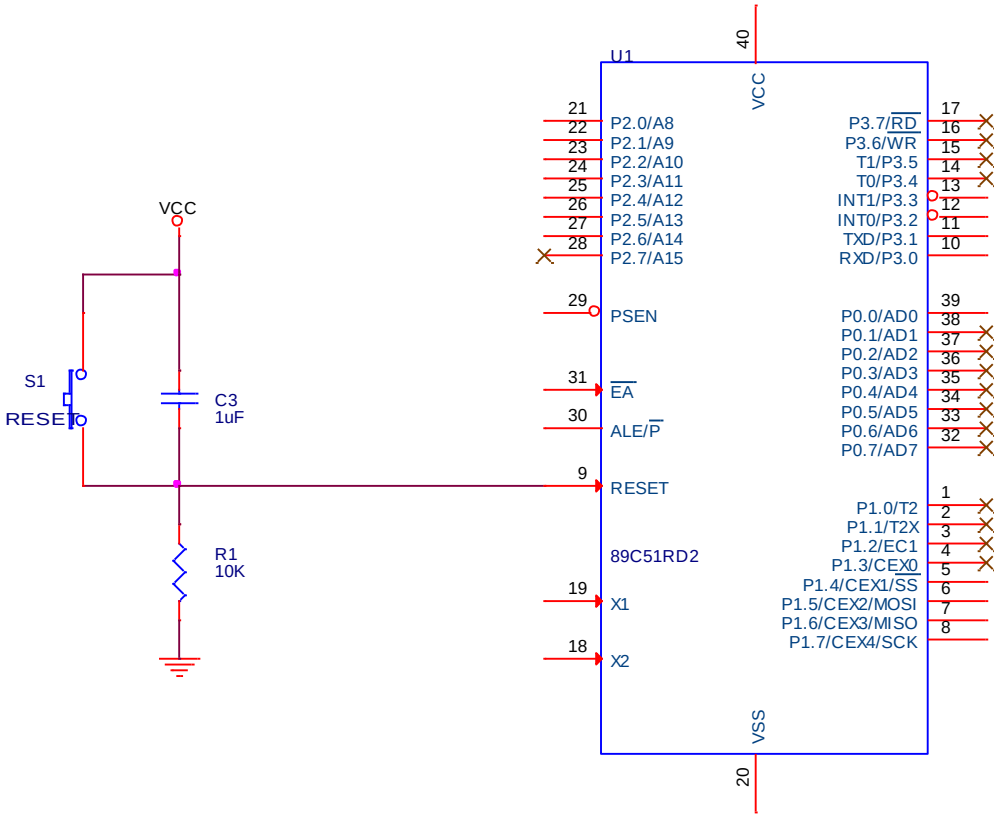
P3.7 / RD - קו יציאה המציין ליחידות החיצוניות, שה-CPU מבקש לקרוא נתון.

מעגל ה - RESET

מטרת הניסוי:

הבנת תפקיד ה-RESET ואופן פעולתו .

<u>רקע תיאורטי :</u>



<u>פעולת ה-RESET :</u>

הפעולה של ה- RESET הכרחית בעת חיבור הבקר למקור מתח על מנת להתחיל את התוכנית שצרובה בבקר מהתחלה ,כלומר התוכנית ההתחלתית קבועה והיא נמצאת בכתובת קבועה כאשר אנו נלחץ על הלחצן או נחבר מקור מתח המעבד ישר יקפוץ לכתובת של התוכנית הראשית .

בעת אתחול הבקר הוא מבצע מספר פעולת למרות שעדיין התוכנית לא רצה, בטבלה שנציג בהמשך נראה את הפעולות שמתבצעות .

<u>הרכיבים המשתתפים :</u>

לחצן Switch , קבל 1µ, נגד 10K , מתח VCC , אדמה .

<u>אופן הפעולה :</u>

תפקיד הקבל הוא גם לבצע את פעולת ה-RESET בעת הדלקה של המערכת או הפסקת מתח פתאומית וגם למנוע ריטוט מגעים בזמן הלחיצה על המתג.

במצב הראשוני לפני החיבור למתח VCC הקבל פרוק, כלומר אין עליו מתח כלל , ברגע חיבור למקור VCC עדיין אין על הקבל מתח (מתפקד כ-קצר) ואילו על הנגד נופל כל המתח VCC ולכן מגיע להדק ה-RESET של הבקר "1" לוגי (בבקר שלנו כך מתבצעת פעולת האתחול על ידי "1" לוגי בכניסת ההדק) על ידי כך מתבצעת פעולת האתחול של המערכת .

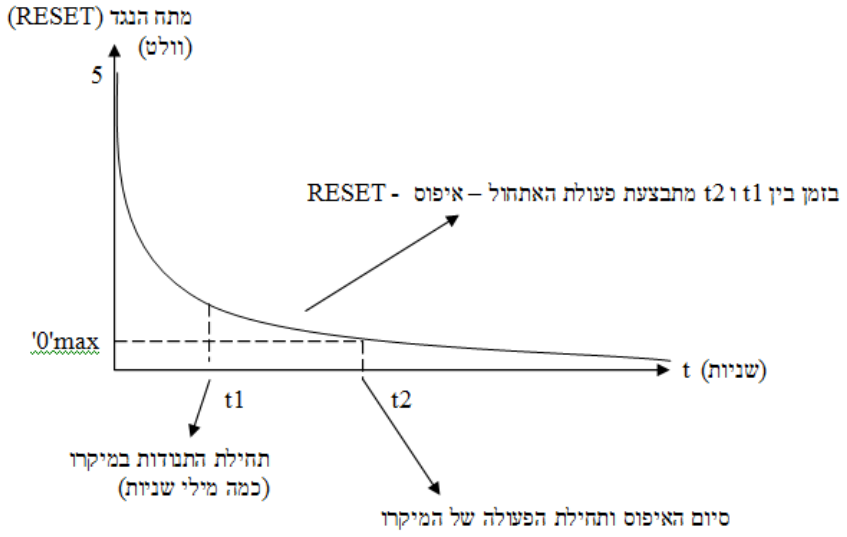
לאחר זמן מסוים הקבל נטען ואז הוא מתפקד כ-נתק , על הנגד נשאר מתח השווה ל- "0" לוגי ולכן בהדק ה-RESET של הבקר יש "0" לוגי .

פעולה זאת היא אתחול אוטומטי מכוון שהיא מתבצעת עם הדלקת המערכת (כאשר מזינים למעגל מתח) .

ברגע שנלחץ על ה-Switch אנו נקצר את הקבל כלומר הקבל יפרוק את המתח שיש עליו ופעולת האתחול תתבצע שוב על פי אותו העיקרון , פעולה זאת נקראת אתחול ידני .

חשוב לציין ש- "0" לוגי זהו לא 0V מוחלט לכן נהוג לכנותו '0' max ואילו "1" לוגי גם כן מתח המתקרב לVCC (לאחר 5 טאו) .

צורת הגל של פעולת ה RESET נראית כך :



על פי הגרף אנו רואים שכאשר המתח על הקבל מגיע ל- t1 אז מתחילות התנודות במיקרו בקר שלוקחות כמה מילי שניות על פי דפיי הנתונים של הבקר אנו רואים שהיצרן אומר שכדי שפעולת ה-RESET תפעל בצורה תקינה פעולת ה-RESET צריכה לעבור לפחות 2 זמני מכונה .

<u>זמן מכונה :</u> עבור גביש שמייצר תנודות של 12Mhz אז 12 מחזורי שעון הם 1Mhz על פי הנוסחה הבאה - $\frac{1}{12Mhz} * 12 = 1\mu s$ אפשר לראות שזמן מכונה הוא 1μs על כן אנו צריכים 2 זמני מכונה כלומר 2μs כדי לקבל פעולה תקינה(בין t1 ל t2 צריך לעבור 2μs לפעולה תקינה) .

על מנת שבפרויקט שלנו מעגל ה-RESET יעבוד בצורה תקינה אנו צריכים לקבוע את ה-טאו שלנו על ידי הקבל נגד בכדיי לעבור את ה-2μs אנו לקחנו נגד של 10KΩ וקבל של 1μF ולכן קבוע הזמן שלנו הוא : $C*R \rightarrow 10K\Omega*1\mu F = 10ms$

זמן זה הוא גדול מאוד ביחס ל -2μs ולכן פעולת ה-RESET תעבוד בצורה תקינה .

כאשר אנו נבצע פעולת RESET הרגיסטרים יקבלו את הערכים הבאים על פי הטבלה הבאה :

SFR Name	Reset Value
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	FFH
IP	XXX00000B
IE	0XX00000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SCON	00H
SBUF	Indeterminate
PCON	0XXX0000B

במידה והאתחול נעצר באמצע כלומר לא עברו 2 זמני מכונה לא כל הרגיסטרים יקבלו את הערכים הרצויים (על פי הטבלה), וכאשר התוכנית תרוץ היא תעשה דברים באקראי.

מהלך הניסוי :

כאמור לעיל ניסוי זה אנו נבצע בתוכניות הבאות המופיעות בפרויקט (לדוגמא: lcd_pwm). לאחר שצרבנו את התוכנית וחיברנו מקור מתח חיצוני, לחצנו על ה-Switch של ה-RESET וראינו שהתוכנית רצה מההתחלה וביצעה את הפקודות על פי התוכנית , התוכנית עמדה בדרישות על כן אנו יודעים שמעגל ה-RESET עובד ומתוכנן כמו שצריך .

לסיכום :

בניסוי זה למדנו את תפקידו של מעגל ה-RESET את אופי החיבור לבקר ,

את החישוב שיש לעשות על מנת שיעברו 2 זמני מכונה כמו כן למדנו את אופן פעולתו של מעגל ה-RESET

בנוסף למדנו אלו פעולות קורות לאחר ה-RESET כלומר איזה ערכים יקבלו הרגיסטרים ,

כמו כן הבנו מדוע צריכים לעבור 2 זמני מכונה לביצוע פעולה תקינה

מעגל הגביש :

מטרת הניסוי:

- הכרת אופן פעולת הגביש .
- בדיקת תדר התנודות של הגביש .

רקע תיאורטי:

התדר של הגביש (כמו שעון) הוא שמפעיל את הבקר וכתוצאה מכך את המערכת כולה , המיקרו בקר מבצע את הפעולות שלו בשלבים לכן חשוב שיהיה תזמון מדויק אות התדר מגיע ממקור חיצוני הנכנס למיקרו בקר כלומר גביש (כמו כן המעגל בנוי מגביש ושני קבלים) ,

הגביש לרוב עשוי מלוחית קוורץ דקה כאשר מבנהו צורת עיבודו ובעיקר עוביו קובעים את התכונות החשמלית שלו .

את הגביש אפשר או לחתוך מגביש גדול יותר אשר מצוי בטבע או על ידי "גידול " במעבדה .

תכונות הגביש :

- אם נחבר מתח חשמלי בין קצות הגביש הגביש יתכופף סביב המרכז שלו זאת אומרת שאם נחבר אל הגביש מתח חילופין אז הגביש ישנה כל פעם את כיוון הכיפוף שלו וזה מה שיוצר לנו את ההתנדנדות של הגביש פיסית .
- אם מפעילים לחץ פיסי על מרכז הגביש אנו נגרום לתנועה של אלקטרוני בתוך הגביש ולהצטברות מטען בקצוות הגביש .

אם נפעיל לחץ בכיוון הפוך זה יגרום לתנועת אלקטרונים הפוכה ולמטען הפוך בקוטביותו.
אם נפעיל לחץ בכיוונים מתחלפים אנו נגרום להופעת מתח מתחלף בקצות הגביש .

אם נאחד את שתי התכונות הללו , ניקח גביש מתנדנד ונפעיל עליו לחץ ואת המתחים הנוצרים בקצוות נחבר למגבר ומהמגבר נחזיר את המתח אל לוחות הגביש בקוטביות כזו שתחזק את התנודות הנוצרות אנו נקבל מתנד זאת אומרת מעגל שיוצר תנודות בלתי פוסקות בתדר מסוים כמובן שהתדר תלוי במבנה המכאני של אותו גביש וביכולתו להתנדנד .

הגביש מתנהג כמו מעגל בתדר תהודה (סליל ,קבל), היתרון הוא שהתדר קבוע והוא תלוי מעט מאוד בגורמים סביבתיים כמו טמפרטורה וכו'

על כל גביש כתוב תדר התנודות שלו למשל אצלנו בפרויקט תדר הגביש הוא: 11.0592MHz .

הגביש משפיע לנו על כל הרכיבים והוא למעשה מתזמן לנו את הסדר על כן אם אנו נשנה את תדר הגביש אז המעגל החשמלי לא יעבוד בצורה טובה , אפשר לדמות את הגביש לשעון שמחליף לי רמזור אדום כלומר כאשר יש לי כמה רכיבים ואחד מהם מהיר יותר אז הוא צריך לחכות לרכיב השני שהוא איטי יותר שגם הוא יסיים את הפעולה שלו (במידה והם תלויים אחד בשני) זאת אומרת שרק כאשר הרכיב האיטי יסיים את פעולתו אז יתחלף האור האדום של הרמזור ויאפשר את המשך הפעילות של המערכת ,

מכאן יוצא שאם אנו נחבר גביש מהיר יותר אז הרכיב האיטי לא יספיק לסיים את פעולתו וכבר יתחלף האור האדום כלומר הרכיב המהיר המשיך לפעולה הבאה כאשר הוא לא חיכה לרכיב האיטי או אפילו כאשר הוא עדיין לא קיבל נתון מהרכיב האיטי כלומר המערכת לא תעבוד בצורה הרצויה .

אם אנו נחבר גביש איטי יותר כל פעילות המעבד תהיה יותר איטית והמטרה שהמערכת תעבוד בצורה היעילה ביותר כלומר מינימום זמן וביצוע המשימה .

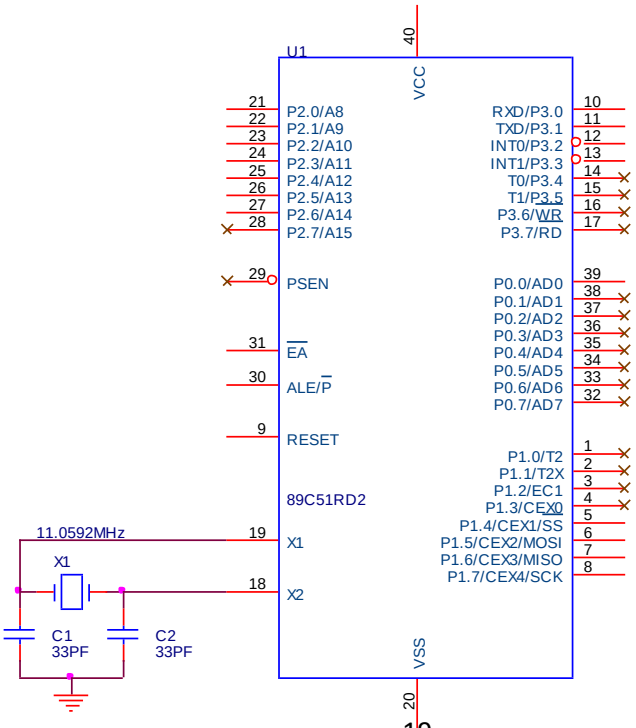
<u>קוצבי זמן במעבד :</u>

במעבד שלנו ישנם שני קוצבי זמן /מונים (TIMER 0, TIMER 1) , אורכו של כל אחד הוא 16 סיביות .

באמצעות קוצבי הזמן / מונים אפשר לממש פעולות הדורשות מדידות מדויקות של זמן .

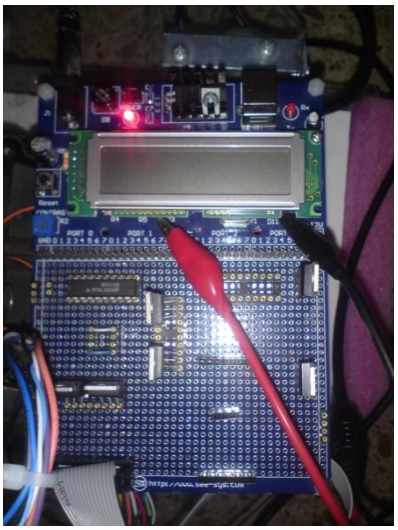
כל קוצב זמן יוצר סדרת דפקים בקצב של מחזור העבודה של המעבד , כלומר 1/12 ממחזור הדפקים של הגביש , באמצעות קוצבי הזמן אפשר למשל למדוד במדויק זמן בין אירועים , לספור אירועים , וכן ליצור סידרה של דפקים בתדרים שונים בהתאם לתוכנית .

כיצד הגביש מחובר לבקר :



<u>.מהלך הניסוי :</u>

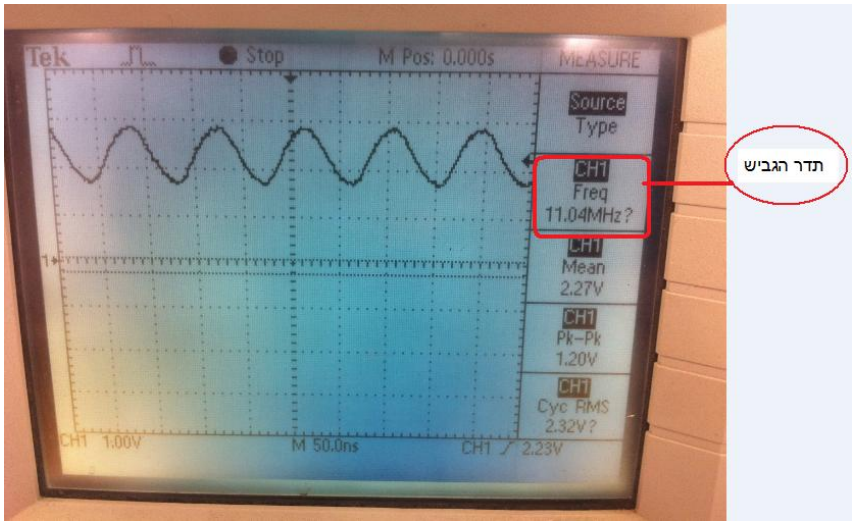
במהלך ניסוי זה חיברנו את הגביש לסקופ , את אחת מהדקי הגביש חיברנו אל היציאה האדומה של הפרוב ואת היציאה השחורה של הפרוב חיברנו אל הארקה בצורה הבאה (יש לשים לב לחוט הלבן שיוצא מהגביש שמחובר לתנין האדום של הפרוב והתנין השני של הפרוב מחובר לאדמה):



<u>תוצאות הניסוי :</u>

כמצופה אנו יכולים להבחין שהסקופ הראה לנו את תדר הגביש של המיקרו בקר שהוא : 11.04Mhz

כלומר התדר שמוטבע על הגביש (על פי היצרן) מאוד קרוב לתדר שנימדד .



הגביש	נתוני יצרן	תוצאות מעבדה
שמחובר למיקרו בקר	11.0592Mhz	11.04Mhz

מייצב 78XX:

מטרת הניסוי:

- הכרת מייצב מתח .
- מטרת מייצב המתח .
- אופן הפעולה עם המייצב מתח .
- שילוב מייצב המתח עם מעגל הספק כוח .

רקע תיאורטי :

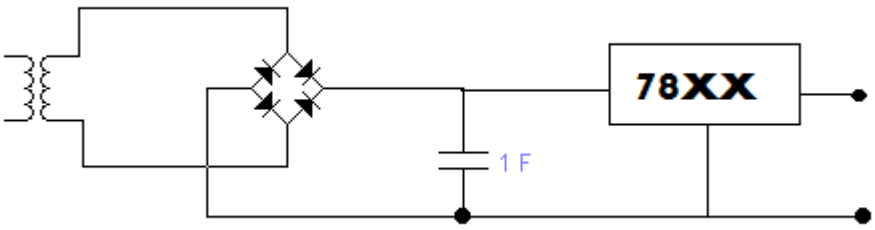
ישנם מספר מייצבי מתח מייצבים כגון משפחת ה – 78XX

78XX. זו משפחת רכיבים שנועדה לייצוב מתח קבוע. הקידומת 78 באה לציין ייצוב של מתח חיובי, הקידומת XX מציינת את מתח המוצא המיוצב. משפחה זו מיועדת לייצוב מתח הנע בין 5v עד 24v.

לכל מייצב מתח ישנו מספר קטלוגי על פי המספר הקטלוגי אנו יכולים לדעת איזה מייצב יש לנו .

לדוגמא : 78הוא מספר קבוע. 03הוא המתח שהמייצב מייצב במוצא . =7803

7803=3v	7812=12V
7805=5V	7815=15V
7806=6V	7824=24V
7808=8V	7830=30V



על פי ציור זה אנו רואים שאחרי הגשר דיודות המתח חילופין הופך להיות גל מיושר (חיובי) ולאחר הקבל (מזמן הפריקה שלו עד לשיא השני של המתח של הגל המיושר) אנו רואים מתח כמעט ישר (אבל לא מיוצב) . לאחר המייצב מתח, אנו נקבל מתח יציב .

כך אפשר להפוך מתח חילופין למתח ישר (חיובי) .

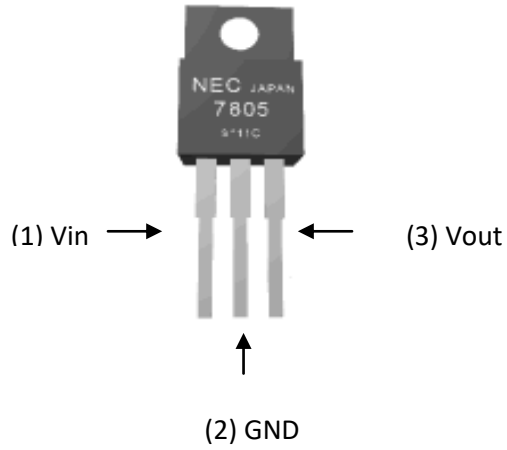
בכדיי שהמייצב מתח ייצב לנו מתח כמו שצריך ה-Δ בין הכניסה ליציאה צריך להיות לפחות 2V כלומר אצלנו בפרויקט אנו משתמשים במייצב 7805 כלומר המתח כניסה אילו חייב להיות מעל 7V

<u>רגלי הרכיב:</u>

1. Vin – מתח כניסה לא מיוצב.
2. Gnd – אדמה.
3. Vout – מתח מוצא מיוצב.

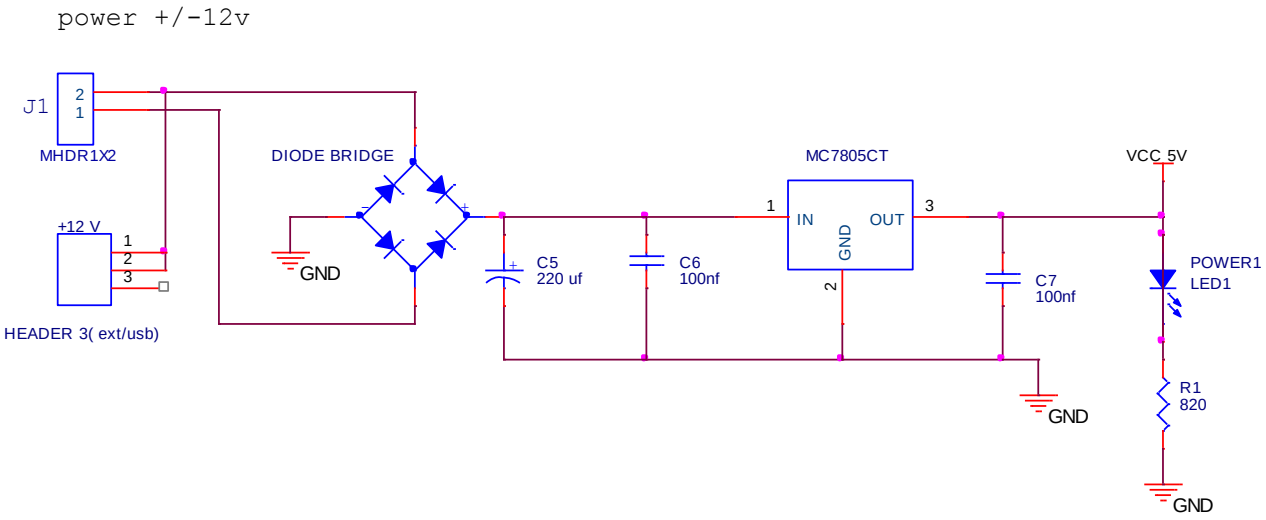
מכיוון שמתח המייצב מתקבל עם רעשים ישנם במעגל קבל במקביל למוצא המייצב (בין רגל Vout לאדמה).

<u>אופן חיבור הרכיב:</u>



בפרויקט השתמשנו במייצבי 7805 .

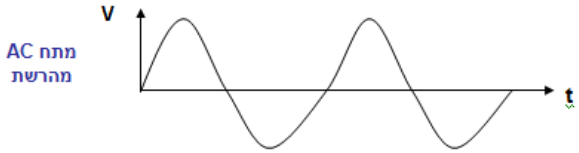
המייצב 7805 משמש ללוח המיקרו בקר



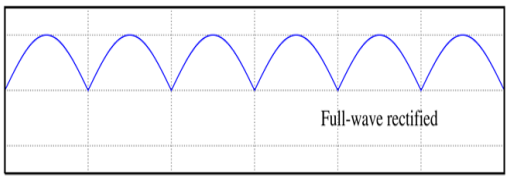
<u>הסבר המעגל :</u>

ה-HEADER 3 זהו ג'אמפר שמאפשר כניסה של מתח או מהחיבור USB דרך חיבור J5 או ממקור מתח דרך חיבור J1 (רגל 3 מחובר למתח ה- USB) .

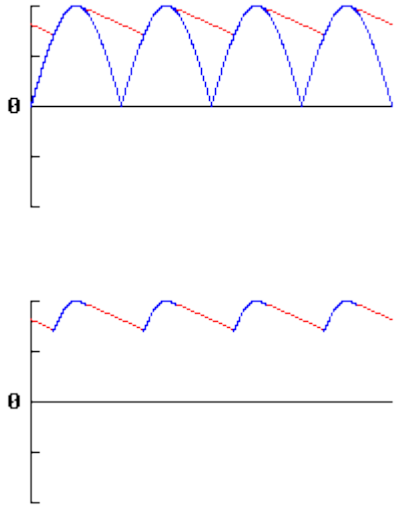
J1- מחבר של מקור מתח 12V (ספק כח או סוללות) AC .



DIODE BRIDGE- מיישר מתח חילופין (דו דרכי)



קבל C5- כדיי לישר את המתח כמו ע"פ האיור הבא (צריך להיות גדול):



קבל C6- בכדי לסנן רעשים של מתח (זאת אומרת מיישר אותו עוד יותר כלומר מקטין את האדוות).

MC7805CT- מייצב מתח על מנת שיצא לנו מתח ישר של 5V ללא אדוות .

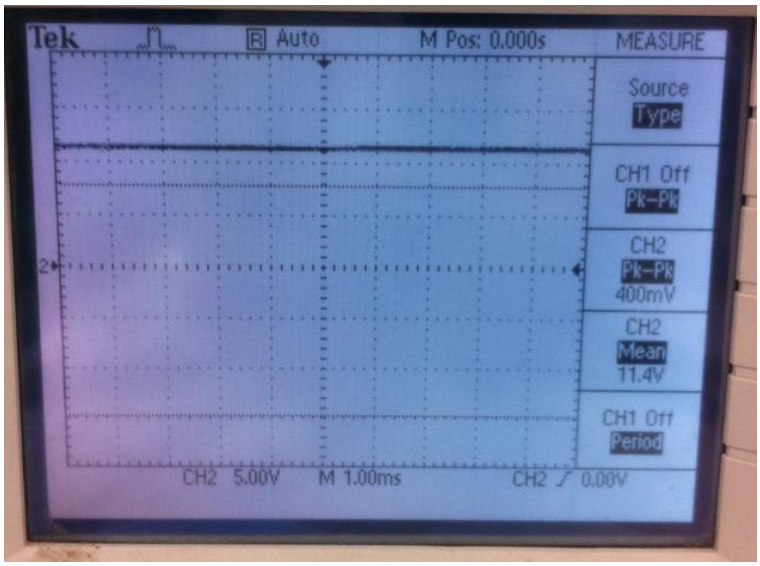
קבל C7- קבל זה לא הכרחי לייצוב אולם הוא יכול לשפר את הייצוב של המתח אך אם הקבל קטן מ- 0.1µF הוא יכול להפריע לייצוב (קבל סינון) , במעבר בין "0" לוגי ל- "1" לוגי ישנם רעשים (כמו מעבר בין קצר לנתק/מפסק) .

LED & RESISTOR- על מנת לומר לנו שיש מתח במעגל כלומר ישנו מקור מתח מחובר .

<u>מהלך הניסוי :</u>

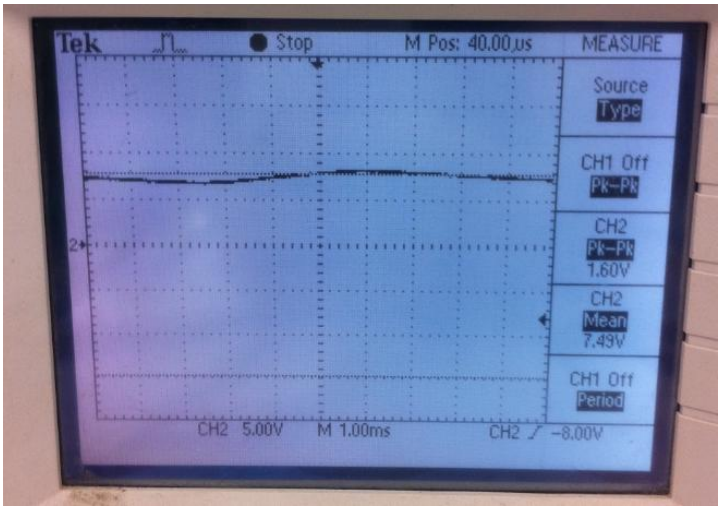
חיברנו את החיבור J1 אל ספק מתח חיצוני . לאחר מכן בדקנו את המתח על ידי הסקופ .

ליפני הגשר דיודות להלן התוצאות :



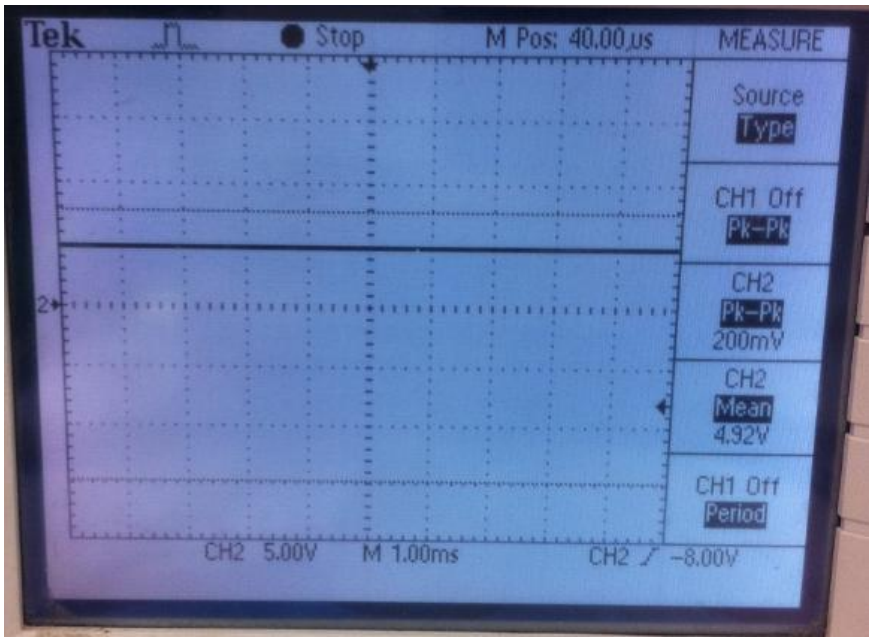
יש לשים לב למתח ה-MEAN שערכו הוא 11.4V ולערך ה- PK-PK שערכו 400mV

לפני המייצב מתח , להלן התוצאות :



יש לשים לב למתח ה-MEAN שערכו הוא 7.49V ולערך ה- PK-PK שערכו 1.60 V

לאחר המייצב מתח (על הלד והנגד) , להלן התוצאות :



יש לשים לב למתח ה-MEAN שערכו הוא 4.92V ולערך ה- PK-PK שערכו 200mV

<u>לסיכום :</u>

בניסוי זה הבנו את תפקידם של הגשר דיודות המייצב הקבלים (מעגל הספק כח) ,

אנו רואים שעל מנת שנוכל לייצר מתח מיוצב בצורה תקינה אנו צריכים את שלושת הרכיבים.

על פי הניסוי שעשינו אנו יכולים להבחין שליפני הגשר דיודות ישנו מתח ישר של 11.4V זאת מכוון שחיברנו את המעגל לספק כח חיצוני שגם הוא עם המערכת הנ"ל כלומר הכנסנו מתח ישר מיוצב ולכן התפקיד של הגשר דיודות במקרה זה הוא נמוך , על פי הסקופ בתמונה השנייה אנו רואים שהמתח הוא לא מיוצב כלומר ישנם תנודות במתח (אנו יכולים לראות שהמתח PK-PK הוא 1.60V כלומר ישנו איזשהו שינוי מתח (הן בגלל הקבלים הן בגלל הגשר דיודות) לאחר מכן בדקנו את המתח היוצא מהמייצב מתח 7805 ואנו יכולים לראות שהמתח שקיבלנו הוא 4.92V והוא מתח ישר ומיוצב כמו כן אפשר לשים לב ל- PK-PK שהוא מזערי ביותר וערכו הוא 200mV .

בניסוי זה למדנו כיצד לבנות מעגל שיאפשר את ייצוב המתח על פי בחירתנו

טיימרים:

הצורך במעגלים קוצבי זמן או טיימרים קשור לסביבת עבודתו של הבקר בתחומי התעשייה ואף בתחומים אחרים.

במערכות שונות יש צורך להמתין לתהליכים שונים.

באחרים יש צורך לשלוט על זמן פעולתם.

ניתן לבצע השהיית זמן בחומרה, אך פעולה זו מנטרלת את המעבד לפרק הזמן שהוא עוסק בהשהיה ובכך חלק גדול מהזמן אין המעבד מטפל בכל ההתקנים החיצוניים.

בבקר כדי למנוע עיסוק יתר של המעבד בתוכניות השהייה יצרו רכיבי השהייה בחומרה כך שאת תפקיד הטיימרים משחררים מהמעבד ותהליך זה מדויק יותר מהשהיית תוכנה.

המונים/קוצבי הזמן נקראים TIMER0 ו- TIMER1. קוצבי הזמן מורכבים מארבעה אוגרים בני 8 סיביות במערכת האוגרים ונקראים TH1,TH0 , TL1, TH1.

לקוצבי זמן אלה אפשריות עבודה שונות ומגוונות הנקבעות על ידי המשתמש בעזרת שני אוגרי בקרה TCON ו- TMOD.

האוגר TMOD-

אוגר זה מחולק לשני חלקים- חלק אחד מטפל ב TIMER0 והחלק השני מטפל ב TIMER1.

הסיבית C/T' קובעת אם הטיימר יתפקד כמונה או קוצב זמן אם סיבית זו ב- "0", מופנים ל המונים TL ו TH אותות המגיעים משעון ה C.P.U ומוחלקים ל12. זהו שימוש כקוצב זמן. אם C/T' נמצאת ב- "1", מופנה כניסת המונה לקו T0 או T1 שהם קווים חיצוניים לC.P.U .

הסיביות M0 ו M1 קובעות את אופן העבודה של קוצב הזמן.

הסיבית GATE זוהי סיבית האפשור של קוצב הזמן והיא צריכה להיות ב- "1".

<u>האוגר TCON-</u>

ארבעת הסיביות הגבוהות של אוגר ה- TCON מיועד לבקרה על אופן פעולת קוצב הזמן. ארבעת סיביותיו הנמוכות משמשות לקביעת אופן הפעולה של בקשות הפסיקה החיצוניות - האם תהיינה מסוג של דרבון קצה (Edge Triggering) או מסוג של דרבון רמה (Level Triggering).הסיביות IT0 ו - IT1 נקבעות בתוכנה ע"י המשתמש, בהתאם לאופן העבודה הרצוי. "1" בסיבית גורם שאותה פסיקה תהיה מסוג דרבון קצה ו- "0" גורם שאותה פסיקה תהיה מסוג של דרבון רמה.

הסיביות IE0 ו- IE1 הן למעשה קובעות אם נעבוד בדרבון קצה או בדרבון רמה. הן עולות ל- "1" כאשר מתקבלת בקשת פסיקה בדרבון קצה, ומתאפסות כאשר ה- C.P.U מבצע את בקשת הפסיקה. ניתן לפנות לאוגר TCON בתוכנה ולבדוק מצבן של סיביות אלה, להעלותן ל- "1" או לאפסן.

החלק הימני של TCON משמש לבקרת הפסיקות החיצוניות. הסיביות TR0 ו- TR1 משמשים להפעלה בתוכנה של קוצבי הזמן. אם סיבית TR ב- "1" מופעל קוצב הזמן בתנאי שסיבית GATE המתאימה נמצאת ב- "0".

סיבית GATE היא סיבית ב- TMOD. כאשר היא נמצאת ב- "1" ו- TR נמצאת ב- "1", משמשת כניסת הפסיקה החיצונית לבקרה על הקוצב. כאשר הכניסה ב- "0", מופעל קוצב הזמן, וכאשר היא ב- "1", מופסקת פעולתו.

כל דופק המגיע למונים TL ו- TH מקדם אותם ב- 1. כאשר הם מתמלאים ב- 1 ועוברים ל- 0 נוצרת סיבית הצפה (OVERFLOW) וזו מפעילה את בקשת פסיקה בהתאם (פסיקת TIMER0 או TIMER). בד"כ, רוצים שקוצב הזמן יספור מספר מסוים של דפקים ואז יצור בקשת פסיקה.

למונים של קצבי הזמן יש ארבעה אופני עבודה שונים הנקראים MODE0 - MODE3. אופנים אלה נקבעים לכל קוצב באמצעות הסיביות M0 ו- M1 שבאוגר TMOD.

אופני העבודה של ה-TIMERS 0,1 :

M1 = 0, M0 = 0 - MODE0

באופן עבודה זה, קוצב הזמן מתפקד כמתואר בציור המתאר את המונים/קוצבי הזמן, העובדה שהמונה TL מתפקד כמונה בן 5 סיביות בלבד ו TH מתפקד כמונה של 8 סיביות גורמת לכך שאנו נוכל למנות מ 0 עד 8191 פולסי שעות .

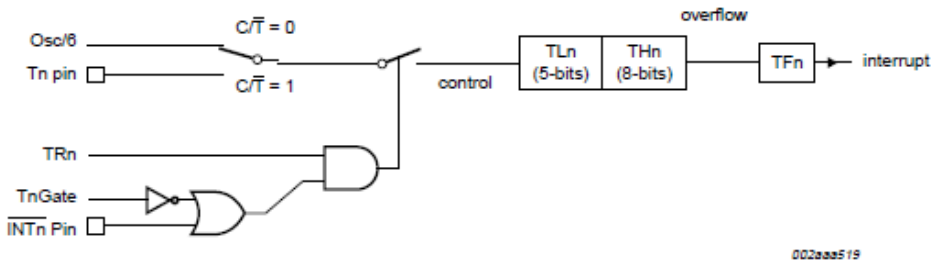


Fig 7. Timer/Counter 0 or 1 in Mode 0 (13-bit counter).

M1 = 0, M1 = 1 - MODE1

קוצב הזמן מתפקד בדיוק כמתואר בציור המתאר את המונים/קוצבי הזמן, כאשר המונים מתפקדים כמונה בן 16 סיביות.

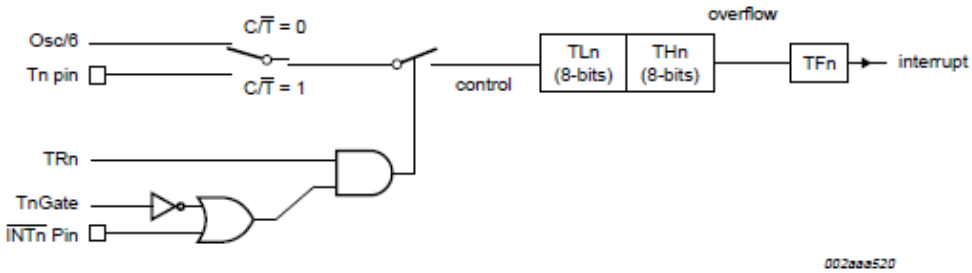


Fig 8. Timer/Counter 0 or 1 in Mode 1 (16-bit counter).

M1= 1, M0 = 0 - MODE2

באופן עבודה 2, משמש TL כמונה בן 8 סיביות ו-TH כ-LATCH. את TH טוענים בערך לטעינה חוזרת (RELOAD). בכל פעם שמסתיימת הספירה של המונה TL, הוא נטען אוטומטית מחדש בערך RELOAD, הנמצא ב-TH, ומתחיל שוב בריצה מאותו ערך.

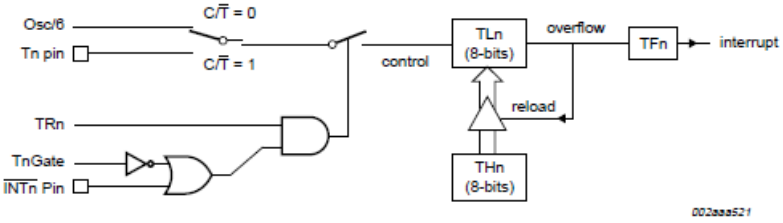


Fig 9. Timer/Counter 0 or 1 in Mode 2 (8-bit auto-reload).

M1 = 1, M0 =1 - MODE3

באופן עבודה זה מתפקד מונה 0 כשני מונים נפרדים בגודל 8 ביט הפועלים באופן עצמאי מונה זה יכול למנות מ-0 עד 255 פולסי שעון (ב mode זה טיימר 1 אינו פעיל)

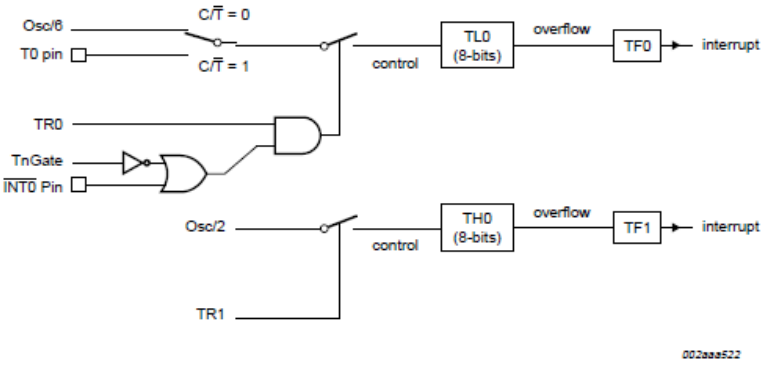


Fig 10. Timer/Counter 0 Mode 3 (two 8-bit counters).

כדי לחשב את המספר הדרוש לאחסון באוגרי הקוצב ע"מ לקבל השהייה באורך מסוים, יש לדעת את תדר שעון המערכת.

<u>חישוב זמנים ליצירת השהייה :</u>

Fosc- תדר השעון בבקר שלנו התדר הוא 11.0592Mhz.

f(timer)- תדר הטיימר .

T(timer) - זמן מחזור של הטיימר .

t(time)- הזמן הרצוי (אני מציב את הערך) .

P(pulses) - מספר פולסי שעון שהטיימר צריך לספור על מנת לקבל את הזמן הרצוי .

$$\frac{Fosc}{12} = \text{f(timer)} \rightarrow \frac{1}{\text{f(timer)}} = \text{T(timer)} \rightarrow \frac{\text{t(time)}}{\text{T(timer)}} = \text{p}(pulses)$$

ב- 8051 הטיימרים סופרים כלפי מעלה זאת אומרת שכדי שהטיימרים יספרו לנו את המספר הרצוי אנו צריכים לקחת את המספר המקסימאלי ולהפחית את מספר פולסי השעון

כלומר במקרה שאנו עובדים עם MODE 2 אז אנו עושים :

מספר שצריך להציב =TH=256- P(pulses)

בפרויקט השתמשנו ב MODE 1 אז אנו עובדים עם 16 ביט ולכן אנו עושים:

מספר שצריך להציב =TH+TL=FFFF- P(pulses)(ראה תוכנית עם פקודות הזזה)

<u>שימושים נוספים של הטיימרים בפרויקט:</u>

בפרויקט השתמשנו בטיימרים בליצור השהיות במספר דברים, בניהם-

בין שינוי פעולות של המנועים(הזזתם או שינוי כיוון)ייצרנו השהייה על מנת שלא יהיו קפיצות מתח בכדי שלא יינזק המנוע .

<u>תכנות ה- טיימרים :</u>

```
#include "wait_ms.h"


void T0_Wait_ms (unsigned int ms)    //This function to send a num to dyleay
{
	TMOD &= ~0x0f;           //save msb TMOD . 16-bit free run mode
	TMOD |=  0x01;          // 16-bit free run mode 1


	while (ms)                                //loop
	{
		TR0 = 0;                           // Stop Timer0
		TL0=(-(F_OSC/12000))&0X00FF;
		TH0=((-(F_OSC/12000))&0XFF00)>>8;  // Overflow in 1ms

		TF0 = 0;                           // Clear overflow indicator
		TR0 = 1;                           // Start Timer0
		while (!TF0);                      // Wait for overflow
		ms--;                              // Update ms counter
	}

	TR0 = 0;                              // Stop Timer0
```

```
}
```

על מנת לקבוע השהייה אנו צריכים לשלוח לפונקציה ערך לדוגמה :

```
T0_Wait_ms (1500) // dilay 1.5sec
```

מעגל ה – LCD:

<u>בפרוייקט זה אנו נשתמש בתצוגת הLCD כדי להציג את כיווני הנסיעה של הרובוט(קדימה,אחורה ולצדדים)</u>

<u>ובמידה והוא צריך הטענה הוא יודיע דרך תצוגת ה LCD</u>

<u>מטרת הניסוי:</u>

1. קוד בשפת C תצוגת הLCD תציג הודעות.

השימוש בתצוגה הוא כדי לתת אינפורמציה וויזואלית לגבי הנעשה במערכת בזמן אמת. לדוג' במידה והרובוט בטעינה אזי על הצג יופיע

NEED CHARG. צורה זו מסייעת גם באיתור תקלות מכיוון שכל פקודה שהבקר מקבל ישנה הודעה על התצוגה.

<u>רקע תיאורטי:</u>

תצוגת LCD, Liquid Crystal Display , היא התצוגה הנוחה והפופולארית ביותר להצגת הודעות. ישנם שני סוגים של תצוגות :האחד אלפא נומרי והשני גרפי. תצוגה גרפית שימושית

כאשר רוצים להציג ציורים/אייקונים שאינם חלק מהתווים המוכרים בקוד ASCII . כאשר מחברים תצוגה גרפית יש להדליק/לכבות כל נקודה ונקודה (פיקסלים) על גבי התצוגה על מנת ליצור את הצורה הנדרשת. לעומת זאת, בתצוגה אלפאנומרית ישנם אייקונים מוכנים של תווים (בדרך כלל מותאמים לקוד ASCII) וכדי להציג תו יש לשלוח את הקוד המתאים .

תפקידה

יחידות LCD מיוצרות על ידי יצרנים שונים למשל : Hitachi, Samsung, Lumex, Optrex , Seiko ועוד.

גודל התצוגה תלוי במספר השורות ומספר התווים בשורה. הגדלים הנפוצים :

מספר השורות : 1 , 2 , ו - 4 .

מספר תווים בשורה : 8 , 16 , 20 , 24 , 32 ו 40 .

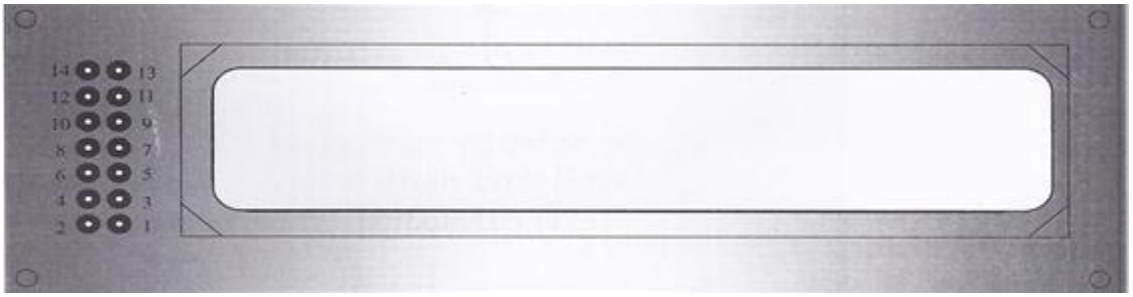
כל התצוגות מוארות באמצעות תאורת הסביבה ולחלקן יש אפשרות להארה אחורית (- Lightin Back).התצוגות מורכבות בדרך כלל על מעגל מודפס קטן ויש להן ממשק סטנדרטי ש 14 הדקים. החיווט להדקים נעשה באמצעות חיבור צמה עם מחברי IDC לפינים שמולחמים למעגל

המודפס או באמצעות הלחמה ישירה של חוטים לחורי הלחמה במעגל המודפס.

הדקי החיבור מסודרים בדרך כלל בשתי שורות באחד מהחלקים הצרים של התצוגה, או

בשורה אחת באחד מהחלקים הרחבים של התצוגה.

להלן מבנה של תצוגות LCD שלנו:



.

מספרי ההדקים מופיעים בדרך כלל על המעגל המודפס . גם אם אין סימון של מספרי ההדקים על המעגל המודפס קל לגלות מהם מספרו ההדקים, מכוון שהדק 1 הוא תמיד Ground והוא מחובר בדרך כלל למשטח הארקה גדול יותר על המעגל.

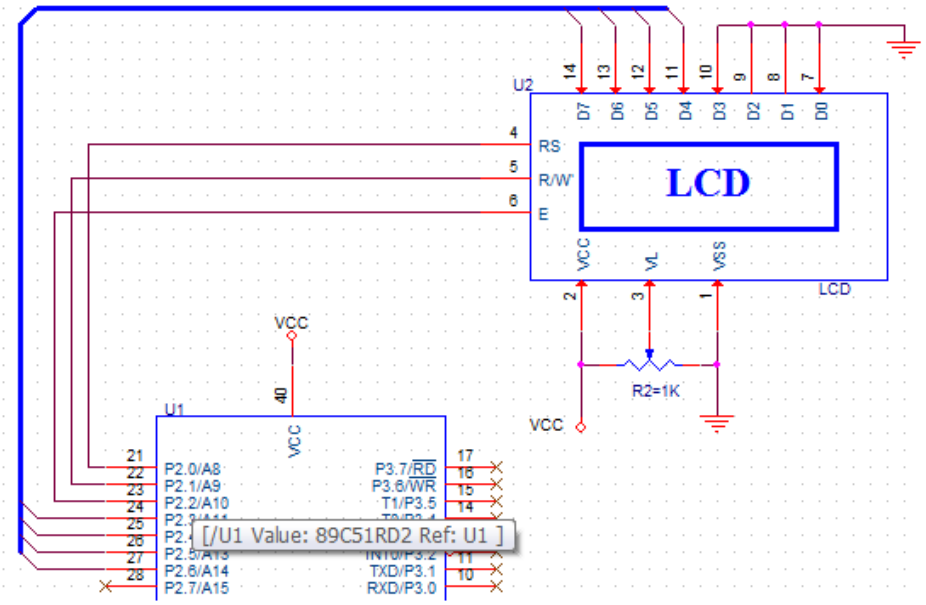
<u>התפקיד המקובל של ההדקים השונים מתואר בטבלה הבאה :</u>

פונקציה	שם PIN(הדק)	מספר PIN(הדק)

1	GND	אדמה
2	VCC	מתח כניסה (הפעלת הרכיב)
3	VEE	לשלוט על בהירות התצוגה
4	RS	קביעת מילת בקרה או נתון
5	R/W	שליטה על פעולה קריאה/כתיבה
6	E	כניסת בקרה ששולטת על ביצוע העברת המידע בין העולם החיצוני והתצוגה
7	D0	Data bit 0
8	D1	Data bit 1
9	D2	Data bit 2
10	D3	Data bit 3
11	D4	Data bit 4
12	D5	Data bit 5
13	D6	Data bit 6
14	D7	Data bit 7

האיור הבא מתאר את ההדקים הנ"ל כהדקי מתח וככניסות והדקי 1/0 של מערכת התצוגה:

כפי שרואים בשרטוט אנו משתמשים בשיטת העברת נתונים של 4 bit ולא 8 bit. הסבר לכך יינתן בהמשך הדוח.



מתח האספקה של התצוגות הוא בדרך כלל 5 Volt , אך התצוגות פועלות באופן

משביע רצון גם כאשר מתח הספק נע בין 4.4 Volt עד 6 Volt , ולעתים אף במתח

נמוך יותר של 3 Volt .

<u>הדק 3 (VEE)</u>

התפקיד של הדק 3 (VEE) הוא לשלוט על בהירות התצוגה (Contrast). השליטה בקוטביות נעשית באמצעות חיבור הדק זה למתח משתנה (למשל באמצעות פוטנציומטר). הפתרון הפשוט ביותר לחיבור של הדק זה הוא לחבר אותו ל - GND . בתצוגות מסוימות החיבור ל - GND נעשה כבר במעגל המודפס של התצוגה עצמה.

לתצוגה קיימים שלושה הדקים שהם כניסות בקרה : RS, RJW ו E .

<u>הדק 4 (RS)</u>

כניסת בקרה ששולטת על בחירת הרגיסטר (Register Select) שאליו ניגשים בתצוגה. כאשר כניסה זו נמצאת ב - Low , המשמעות של העברת byte לתצוגה היא כתיבת פקודה (command display), והמשמעות של קבלת byte מהתצוגה היא קריאת מצב (status information). כאשר כניסה זו (RS) נמצאת ב - High אפשר לכתוב מידע (character data) לתצוגה או לקרוא מידע מהתצוגה.

היות וכניסה זו בוחרת מידע כאשר היא נמצאת ב - High , ניתן היה לקרוא לה גם בשם data_select .

<u>הדק 5 (R/W)</u>

כניסת בקרה ששולטת על פעולת הקריאה (כאשר הכניסה נמצאת ב - High) או כתיבה (כאשר הכניסה נמצאת ב - Low).

אפשר היה לקרוא לכניסה זו גם בשמות הבאים : read / $\overline{\text{write}}$ read .

<u>ניתן לסכם את הפעולה של שתי כניסות הבקרה הנ"ל באופן הבא :</u>

פעולה	R/W	RS
כתיבת מילת בקרה	0	0
קריאה- LCD עסוק	1	0
כותב ל data register	0	1
קורא מהdata register	1	1

<u>הדק 6 (E)</u>

כניסת בקרה ששולטת על ביצוע העברת המידע בין העולם החיצוני והתצוגה.

כדי לאפשר את הפעילות, יש להעלות את E למצב High . כאשר E נמצאת ב - Low , התצוגה אינה מבצעת כל פעולה מכיוון שהיא מנותקת באופן חשמלי מהעולם החיצוני .

אפשר לקרוא לכניסה E גם בשם Enable .

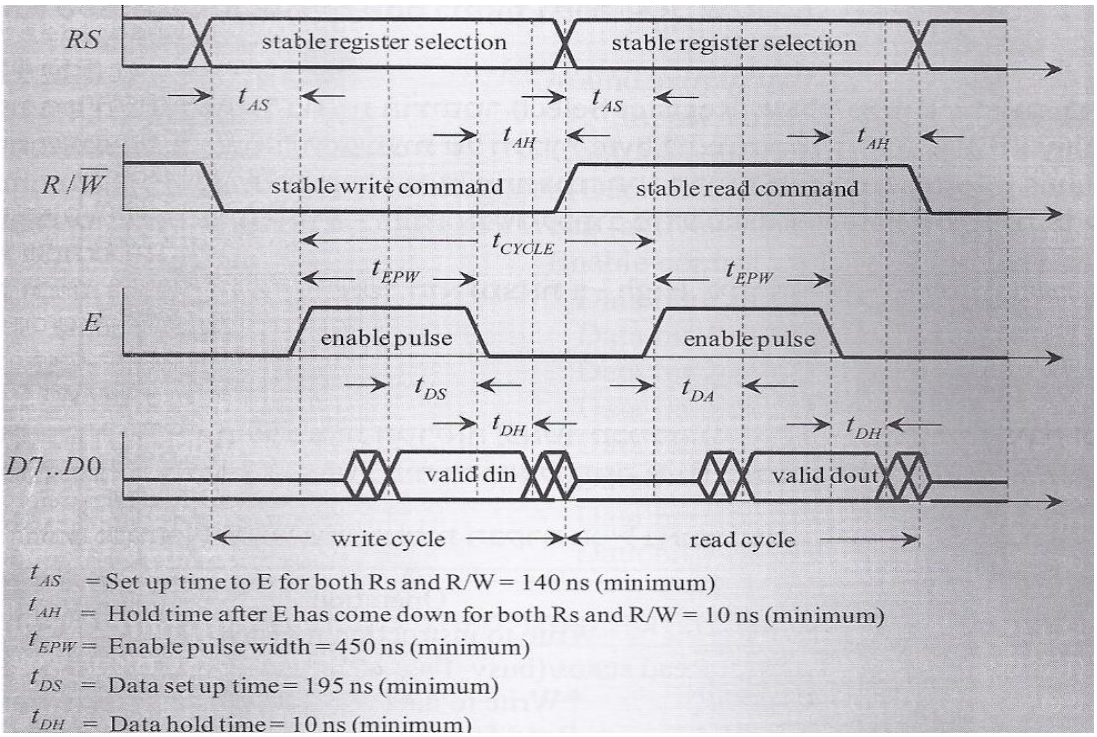
<u>ניתן לסכם את הפעולה של שלוש כניסות הבקרה הנ"ל באופן הבא :</u>

Data bus	פעולה	R/W	RS	E
--	לא מגיב	ø	ø	0
IN	כתיבת מילת בקרה	0	0	1
OUT	קריאה-LCD עסוק	1	0	1
IN	כתיבה לאוגר ה LCD(כתיבה לתצוגה)	0	1	1
OUT	קריאה מאוגר ה LCD	1	1	1

<u>הדקים 7 עד 14 (DO עד D7)</u>

שמונה הדקי BUS המאפשרים העברה של Bytes או Nibb1es (4 סיביות) בין העולם החיצוני והתצוגה.

דיאגרמת הזמנים הבאה מתארת את פעולת הכתיבה ופעולת הקריאה :



לכל התצוגות יש את אותו מיקרו בקר שמפעיל אותן, כלומר המיקרו בקר יודע להפעיל את.

התצוגה הגדולה ביותר (2×64), אך בתצוגות קטנות יותר המקומות הנותרים לא יוצגו .

בתצוגות של 4 שורות יש שני רכיבי מיקרו בקר, כשלכל אחד יש כניסת Enable נפרדת ושי האותות מאוחדים, כלומר שני המיקרו בקרים מקבלים את אותו מידע.

במיקרו-בקר של התצוגה ישנם שני זיכרונות, האחד מסוג ROM אשר נקרא CGRAM (Character generator RAM), והוא מכיל את הפיקסלים אשר יוצרים את כל התווים שתצוגה מכירה, והשני מסוג RAM אשר נקרא DDRAM , והוא מכיל את התווים המוצגים על מסך ה- LCD .

<u>להלן חלק מתכולת ה- CGRAM (Display data Ram) של תצוגת LCD :</u>

MSB / LSB	0000	0001	0010	0011	0100	0101	0110	0111
0000				0	@	P	`	p
0001			!	1	A	Q	a	q
0010			"	2	B	R	b	r
0011			#	3	C	S	c	s
0100			$	4	D	T	d	t
0101			%	5	E	U	e	u
0110			&	6	F	V	f	v
0111			'	7	G	W	g	w
0000			(	8	H	X	h	x
0001			)	9	I	Y	i	y
0010			*	:	J	Z	j	z
0011			+	;	K	[	k	{
0100			,	<	L	¥	l	|
0101			-	=	M	]	m	}
0110			.	>	N	^	n	→
0111			/	?	O	_	o	←

• החלק השני של הזיכרון CGRAM אינו אחיד והוא בדרך כלל מכיל אותיות וסימנים של שפות שונות לפי ארץ הייצור, כמו עברית יפנית וכו' .

• 32 המקומות הראשונים בזיכרון הם מקומות ריקים השמורים למשתמש, כלומר ניתן להוסיף 32 תווים/סימנים נוספים לפי הצורך .

<u>להלן מבנה הזיכרון DDRAM :</u>

Row 1	0	1	2	3	.....	D	E	0f
Row 2	40	41	42	43	.....	4d	4e	4f

בתצוגה בגודל 2x16 ישנן שתי שורות, כאשר השורה הראשונה מתחילה ב- 00H ומסתיימת ב- OfH והשורה השנייה מתחילה ב- H40 ומסתיימת ב- fH4 . כל הכתובות (בין OfH ל- H40) קיימות, אך מכיוון שתוכנן לא יוצג על גבי המסך, יש לדלג עליהן בעת הכתיבה.

<u>פקודות בקרה של התצוגה</u>

בתהליך הדלקת המתח לתצוגה, יכול להיות שתוצג קבוצה של ריבועים שחורים בשורה העליונה של התצוגה. אלו הם בעצם תווים של תצוגה כבויה. בשלב זה היה אולי צורך לווסת את הניגודיות של התצוגה עד שתווים אלו כמעט ייעלמו .בפרק זה לא נבצע פעולה זו .למרות מה שנאמר כרגע, התצוגה עלולה להתעורר לאחר הדלקת המתח גם במצב התחלתי אחר!

בדרך כלל בהדלקת המתח התצוגה עוברת תהליך אתחול עצמי אוטומטי .מומלץ לא לסמוך על פעולת האתחול העצמי הזו ,מכיוון שההתרחשות של פעולה זו עלולה להיות מותנית במהירות של עליית מתח הספק, והתהליך עלול לא להתבצע או להיות שונה במידה מסוימת בין תצוגות שמיוצרות על ידי יצרנים שונים.

גם אם פעולת האתחול אכן התבצעה, המצב ההתחלתי שבו נמצאת התצוגה הוא מצב כבוי. במצב זה לא יראו על התצוגה תווים כלשהם גם אם שלחנו לתצוגה תווים שונים. אנו נרצה כמובן לשנות מצב התחלתי זה למצב מופעל. לשם כך יהיה עלינו לבצע כמה פקודות בקרה על התצוגה.

<u>הטבלה הבאה מתארת את אוסף פקודות הבקרה הנפוצות בתצוגות מסוג זה:</u>

Instruction	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Description	Clock-Cycles
NOP	0	0	0	0	0	0	0	0	0	0	No Operation	0
Clear Display	0	0	0	0	0	0	0	0	0	1	Clear display & set address counter to zero	165
Cursor Home	0	0	0	0	0	0	0	0	1	x	Set adress counter to zero, return shifted display to original position. DD RAM contents remains unchanged.	3
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Set cursor move direction (I/D) and specify automatic display shift (S).	3
Display Control	0	0	0	0	0	0	1	D	C	B	Turn display (D), cursor on/off (C), and cursor blinking (B).	3
Cursor / Display shift	0	0	0	0	0	1	S/C	R/L	x	x	Shift display or move cursor (S/C) and specify direction (R/L).	3
Function Set	0	0	0	0	1	DL	N	F	x	x	Set interface data width (DL), number of display lines (N) and character font (F).	3
Set CGRAM Address	0	0	0	1	CGRAM Address						Set CGRAM address. CGRAM data is sent afterwards.	3
Set DDRAM Address	0	0	1	DDRAM Address							Set DDRAM address. DDRAM data is sent afterwards.	3
Busy Flag & Address	0	1	BF	Address Counter							Read busy flag (BF) and address counter	0
Write Data	1	0	Data								Write data into DDRAM or CGRAM	3
Read Data	1	1	Data								Read data from DDRAM or CGRAM	3

x : Don't care						
	I/D	1 0	Increment Decrement	R/L	1 0	Shift to the right Shift to the left
	S	1 0	Automatic display shift	DL	1 0	8 bit interface 4 bit interface
	D	1 0	Display ON Display OFF	N	1 0	2 lines 1 line
	C	1 0	Cursor ON Cursor OFF	F	1 0	5x10 dots 5x7 dots
	B	1 0	Cursor blinking	DDRAM : Display Data RAM CGRAM : Character Generator RAM		
	S/C	1 0	Display shift Cursor move			

<u>הסבר על אוסף הפקודות הבקרה שבטבלה:</u>

שורה מילת בקרה פעולה

2	01h	ניקוי המסך –תצוגה
4	06h	סמן בצד ימין
	08h	אור תצוגה כבוי , בלי סמן(בתחתית השורה) ,לא מהבהב
	09h	אור תצוגה כבוי , בלי סמן(בתחתית השורה) , מהבהב
	0Ah	אור תצוגה כבוי , עם סמן(בתחתית השורה) ,לא מהבהב
	0Bh	אור תצוגה כבוי , עם סמן(בתחתית השורה) , מהבהב
5	0Ch	אור תצוגה דלוק , בלי סמן(בתחתית השורה) ,לא מהבהב
	0Dh	אור תצוגה דלוק , בלי סמן(בתחתית השורה) , מהבהב
	0Eh	אור תצוגה דלוק , עם סמן(בתחתית השורה) ,לא מהבהב
	0Fh	אור תצוגה דלוק , עם סמן(בתחתית השורה) ,מהבהב
	38h	עבודה במצב של 8 ביט , 2 שורות ,5*7 פיקסל
	3Ch	עבודה במצב של 8 ביט , 2 שורות ,5*10 פיקסל
	2Ch	עבודה במצב של 4 ביט , 2 שורות ,5*10 פיקסל
7	28h	עבודה במצב של 4 ביט , 2 שורות ,5*7 פיקסל
	30h	עבודה במצב של 4 ביט , 2 שורות ,5*7 פיקסל
	34h	עבודה במצב של 8 ביט , 1 שורות ,5*10 פיקסל
	24h	עבודה במצב של 4 ביט , 1 שורות ,5*10 פיקסל
	20h	עבודה במצב של 4 ביט , 1 שורות ,5*7 פיקסל

התצוגה אצלנו בפרוייקט היא של 4 ביט ולא 8 ביט לכן מילת הבקרה ששתיהיה לנו

היא : 2Ch

יתרונות : כאשר החיבור של התצוגה הוא 4 ביט (ולא 8 ביט) אז אני מפנה 4 פורטים ל input ול output שאני יכול להשתמש בהם , נניח שאני משתמש בתצוגה של 8 ביט אז פורט

אחד תפוס .

בנוסף אני צריך לחבר את שלושת ההדקים של הבקרה (RS,R/W,E) זאת אומרת ש 11

פורטים נתפסו לי ונניח שיש רכיב שצריך פורט שלם אז התבזבזו לנו 5 פורטים (שלא

תמיד אפשר להשתמש בהם) .

הסרונות: 2 דברים צריך לשנות בתוכנה של ה- LCD

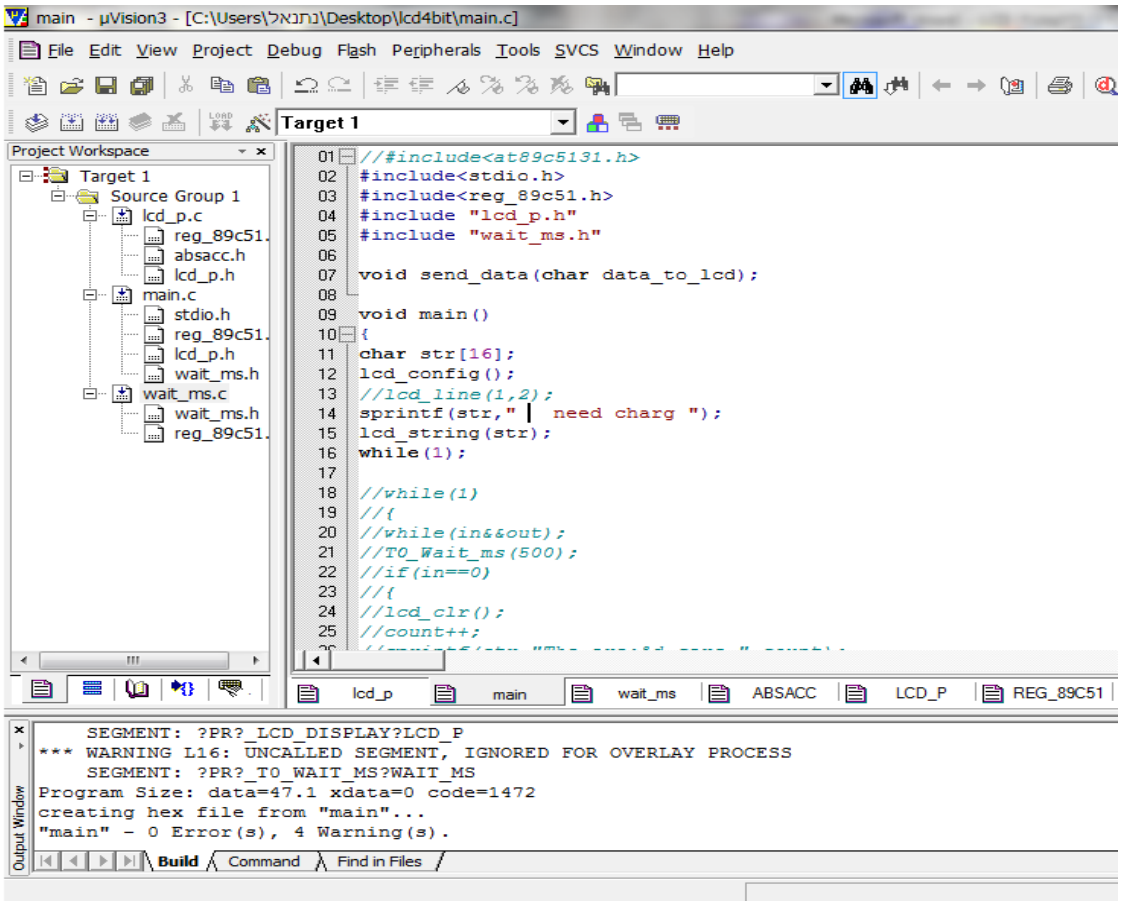
- מילת בקרה צריך לתכנת ל 4 ביט
- צריך לשלוח ערך של 8 ביט לתצוגה ומכוון שאי אפשר לשלוח אלה רק 4 ביט אנו צריכים לשלוח פעמיים את הערך .

הבקר בו אנחנו משתמשים הוא: 89V51RD2.

<u>מהלך הניסוי:</u>

<u>כתבנו את הקוד בתוכנת ה KELL UVISIONS:</u>

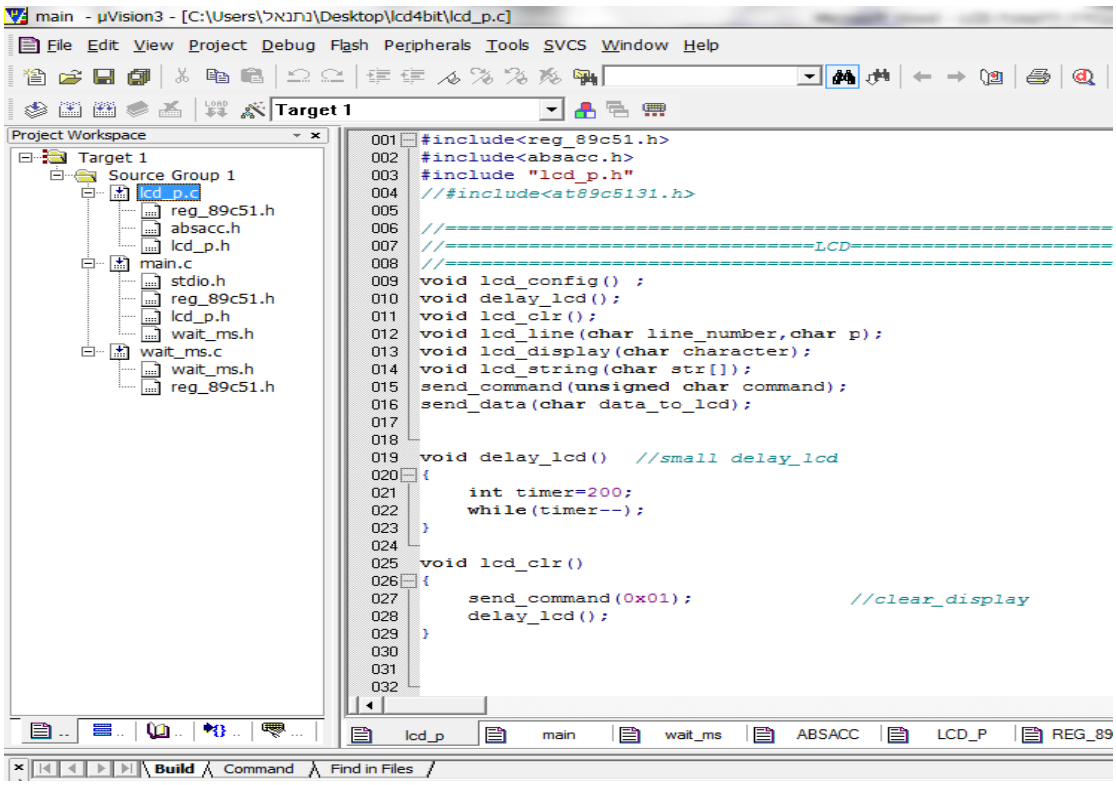
קוד הLCD (main) תלוי גם בתת ספריות עזר, כפי שמוצג בתצלום.



הסבר על ספריית main:

```
#include<stdio.h> // sprint() פרודות על ואחראית פלט/קלט פקודות שכוללת ספריה

#include<reg_89c51.h> //בקר למיקרו ספריה פונקצית צירוץ

#include "lcd_p.h" //LCD  לתצוגת פקודות פקודות קובץ

#include "wait_ms.h" // השהייה פקודות קובץ


void main()

{

char str[16];// תווים 16 של מערך הגדרת

lcd_config(); //לעבוד לתצוגה שמאפשרת פונקציה

sprintf(str,"   need charg   ")// התווים למערך המילה שליחת

lcd_string(str); //התווים ממערך התצוגה על המילה הצגת

while(1); //סופית אין לולאה
```

קוד הLCD (lcd_p) תלוי גם בתת ספריות עזר, כפי שמוצג בתצלום.



הסבר על ספריית lcd_p:

```
#include<reg_89c51.h>// בקר למיקרו ספריה פונקצית צירוץ

#include<absacc.h>

#include "lcd_p.h"// LCD  לתצוגת פקודות פקודות קובץ


//================================LCD=========================
```

```
void lcd_config() ;// תצוגה את שמאתחלת פונקציה

void delay_lcd();// השהייה פונקציית

void lcd_clr();// התצוגה את שמנקה פונקציה

void lcd_line(char line_number,char p);

void lcd_display(char character);

void lcd_string(char str[]);// התצוגה על אותו ומציגה התווים מערך את שלוקחת פונקציה

send_command(unsigned char command);// בקרה מילת לשליחת פונקציייה

send_data(char data_to_lcd);// לתצוגה תו לשליחת פונקציייה


void delay_lcd()  //small delay_lcd// קטנה השהייה פונקציית

{

        int timer=200;

        while(timer--);

}


void lcd_clr()// התצוגה את שמנקה פונקציה

{

        send_command(0x01);            //clear_display

        delay_lcd();

}


//=======================CONFIG THE LCD=====================


void lcd_config() //Initialization of The LCD, התצוגה של אתחול פונקציית

{

    RW=0;

    delay_lcd();

    send_command(0x2c);   //4bit,2 line,5*7 dot  4 bitה פ רואים שהגדרנו את התצוגה ל

    delay_lcd();

    send_command(0x0e);    // display on ,cursor on,cursor blink

    delay_lcd();

    send_command(0x01);     //clear_display

    delay_lcd();

    send_command(0x06);     // increment cursor,no display shift
```

```
}

//=======================LCD LINE NUMBER====================]

void lcd_line(char line_number,char p)// הרצוי במקום המילה את להציג שאפשרת פונקציה
(עמודה,שורה)

{

        switch(line_number)

        {

        case 1:

                delay_lcd();

                send_command(0x80+p);

                break;


        case 2:

                delay_lcd();

                send_command(0xc0+p);

                break;


        case 3:

                delay_lcd();

                send_command(0x94+p);

                break;


        case 4:

                delay_lcd();

                send_command(0xd4+p);

                break;

        }


}


//===============DISPLAY THE CHARACTER ON THE LCD===========

void lcd_display(char character)

{

        delay_lcd();

           send_data(character);     //Send The character to the LCD
```

```
}



void lcd_string(char str[])//  LCD  המערך את שלוקחת פונקציה ncher על אותו ומציגה
מטיפוס

{

        int i=0;

        while(str[i])

           {

                delay_lcd();


                if (str[i]>=0xa0)                                //if hebrew


                        send_data(str[i++]-0x40);

         else


                        send_data(str[i++]);

        }

}



send_command(unsigned char command)

{

    RW=0;

    RS=0;

    PORT_LCD &=0x87;

    PORT_LCD |=((command &0xf0 )>>1);

    E=0;

    delay_lcd();

    E=1;

    delay_lcd();

    E=0;


    PORT_LCD &=0x87;

    PORT_LCD |=((command &0x0f )<<3) ;
```

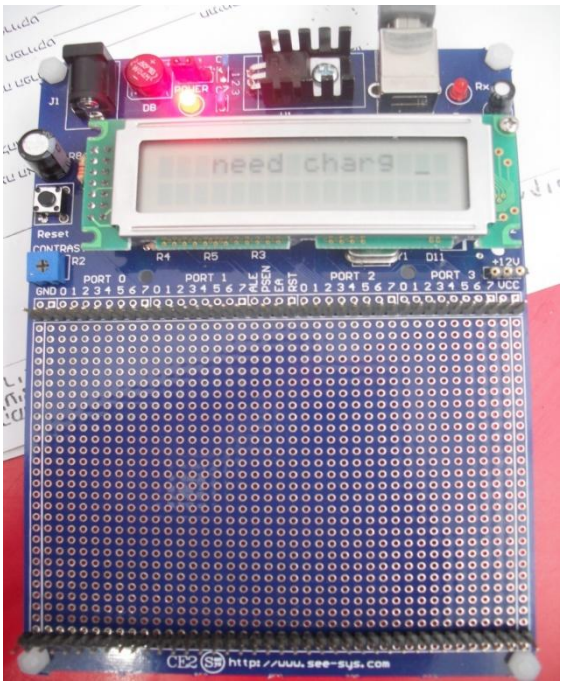
```
    RS=0;
    E=0;
    delay_lcd();
    E=1;
    delay_lcd();
    E=0;

}

send_data(char data_to_lcd)
{
    RW=0;
    RS=1;
    PORT_LCD &=0x87;
    PORT_LCD |= ((data_to_lcd &0xf0 )>>1);
    E=0;
    delay_lcd();
    E=1;
    delay_lcd();
    E=0;

    RS=1;
    PORT_LCD &=0x87;
    PORT_LCD |= ((data_to_lcd &0x0f )<<3) ;
    E=0;
    delay_lcd();
    E=1;
     delay_lcd();
     E=0;
}
```

הסבר על תת ספריה reg_89c51.h נימצא בנספחים בע"מ

הסבר על תת ספריה LCD_P.H:

```
#ifndef _LCD_H

#define _LCD_H

#include<reg_89c51.h>//for mc51 family

/********************************* LCD ******************************/

void lcd_config() ;// תצוגה את שמאתחלת פונקציה

void lcd_clr();// התצוגה את שמנקה פונקציה

void lcd_line(char line_number,char p);// הרצוי במקום המילה את להציג שאפשרת פונקציי
(עמודה,שורה

void lcd_display(char character);// השהייה פונקציית

void lcd_string(char str[]);//התצוגה על אותו ומציגה התווים מערך את שלוקחת פונקציה

/*********************************************************************/

/***************** LCD connecting to the port ***********************/

#define PORT_LCD P2

sbit E =P2^2;//  לפורט 2.2 Eתשם קביע

sbit RW=P2^1;//  לפורט 2.1 RW תשם קביע

sbit RS=P2^0;//  לפורט 2.0  RS תשם קביע

/*********************************************************************/

#endif /* _LCD_H */
```

לאחר שקימפלנו וצרבנו את הקוד לבקר שלנו זה התוצאה:



לבסוף לאחר שצרבנו לבקר ולחצנו על כפתור ה reset רואים את מה שהגדרנו שיהיה כתוב על LCD.

לסיכום:

בדוח למדנו על תצוגת הLCD ועל השימושים בפרוייקט שלנו, והסברנו את הקוד שצרבנו לבקר בשביל שיציג את ההודעה.

חוצץ 74F244 :

מטרת הניסוי :

- אופן חיבורו של הרכיב.
- תפקידו של הרכיב.
- אופן פעולתו .

רקע תיאורטי :

תפקידו של רכיב זה כשמו לחצוץ בין המיקרו בקר לכול עומס בכדי להגן על המיקרו בקר .

אנו רואים על פי הטבלה מטה שהזרם המרבי שהמיקרו בקר יכול לספק הוא :

ברמה נמוכה כאשר מתח המוצא הוא 1V , 3.5 mA.

ברמה הגבוה כאשר מתח המוצא הוא VDD-1.5V , 60µA.

10. Static characteristics

Table 67: DC electrical characteristics

T_{amb} = 0 °C to +70 °C or −40 °C to +85 °C; V_{DD} = 4.5 V to 5.5 V; V_{SS} = 0 V

Symbol	Parameter	Conditions	Min	Max	Unit
V_{IL}	LOW-level input voltage	4.5 V < V_{DD} < 5.5 V	−0.5	$0.2V_{DD}$ − 0.1	V
V_{IH}	HIGH-level input voltage	4.5 V < V_{DD} < 5.5 V	$0.2V_{DD}$ + 0.9	V_{DD} + 0.5	V
V_{IH1}	HIGH-level input voltage (XTAL1, RST)	4.5 V < V_{DD} < 5.5 V	$0.7V_{DD}$	V_{DD} + 0.5	V
V_{OL}	LOW-level output voltage (ports 1.5, 1.6, 1.7)	V_{DD} = 4.5 V; I_{OL} = 16 mA	-	1.0	V
V_{OL}	LOW-level output voltage (ports 1, 2, 3)[1]	V_{DD} = 4.5 V			
		I_{OL} = 100 µA	-	0.3	V
		I_{OL} = 1.6 mA	-	0.45	V
		I_{OL} = 3.5 mA	-	1.0	V
V_{OL1}	LOW-level output voltage (Port 0, ALE, $\overline{PSEN}$)[1][3]	V_{DD} = 4.5 V			
		I_{OL} = 200 µA	-	0.3	V
		I_{OL} = 3.2 mA	-	0.45	V
V_{OH}	HIGH-level output voltage (ports 1, 2, 3, ALE, $\overline{PSEN}$)[4]	V_{DD} = 4.5 V			
		I_{OH} = -10 µA	V_{DD} − 0.3	-	V
		I_{OH} = -30 µA	V_{DD} − 0.7	-	V
		I_{OH} = -60 µA	V_{DD} − 1.5	-	V
V_{OH1}	HIGH-level output voltage (Port 0 in External Bus Mode)[4]	V_{DD} = 4.5 V			
		I_{OH} = -200 µA	V_{DD} − 0.3	-	V

זאת אומרת שהזרם המקסימאלי שהמיקרו בקר יכול לספק הוא 3.5 mA.

המיקרו בקר לא יכול לספק זרם שהעומס צורך לכן ישנו צורך בחוצץ על מנת שיגדיל את הזרם דבר נוסף הוא על מנת שלא תקרה תקלה ואחד מהרכיבים הפריפריאליים יחזירו זרם או כתוצאה מחיבור לא נכון של הרכיבים, המיקרו בקר יכול להישרף .

כאמור לעיל במידה ואחד מהרכיבים הפריפריאליים צריך לצרוך זרם גדול יותר אנו צריכים לחבר לו את החוצץ על פי הטבלה למטה אנו יכולים לראות שהחוצץ יכול להוציא 64 mA ועם זרם כזה ישנם יותר אופציות לשימוש , החוצץ עובד בצורה כזאת שמכוון שיש לו התנגדות מבוא גבוה יחסית הוא צורך זרם של כמה µA בלבד ואינו מעמיס על המיקרו בקר וכך הוא מגן עליו .

DC Electrical Characteristics

Symbol	Parameter		Min	Typ	Max	Units	V_{CC}	Conditions
V_{IH}	Input HIGH Voltage		2.0			V		Recognized as a HIGH Signal
V_{IL}	Input LOW Voltage				0.8	V		Recognized as a LOW Signal
V_{CD}	Input Clamp Diode Voltage				−1.2	V	Min	I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	10% V_{CC}	2.4			V	Min	I_{OH} = −3 mA
		10% V_{CC}	2.0					I_{OH} = −15 mA
		5% V_{CC}	2.7					I_{OH} = −3 mA
V_{OL}	Output LOW Voltage	10% V_{CC}			0.55	V	Min	I_{OL} = 64 mA
I_{IH}	Input HIGH Current				5.0	µA	Max	V_{IN} = 2.7V
I_{BVI}	Input HIGH Current Breakdown Test				7.0	µA	Max	V_{IN} = 7.0V
I_{CEX}	Output HIGH Leakage Current				50	µA	Max	V_{OUT} = V_{CC}
V_{ID}	Input Leakage Test		4.75			V	0.0	I_{ID} = 1.9 µA All Other Pins Grounded
I_{OD}	Output Leakage				3.75	µA	0.0	V_{IOD} = 150 mV

ישנם 2 סיבות שאנו נשתמש בפרויקט שלנו בחוצץ למרות שישנם דוחפים למנוע שמגדילים גם הם את הזרם .

- אם במידה והדוחף למנוע נישרף כתוצאה מחיבור שגוי, והזרם /מתח הגבוה יגיעו להדקי המיקרו בקר הוא עלול להרוס את הבקר ולכן החוצץ מגן לנו על הבקר .
- הדוחפים למנוע בדרך כלל דורשים זרם בכניסה שלהם , הדוחף למנוע בנוי בכניסה כמו טרנז' שדורש זרם בבסיס כלומר דרוש זרם מינימום בכניסה לדוחף, כאמור לעיל אנו רואים שהמעבד יכול להוציא 3.5 mA מקסימום ולא בטוח שכול הדוחפים יסתפקו בזרם זה ולכן אנו מחברים את החוצץ שלא יהיו לנו בעיות עם הדוחפים .

החוצץ בנוי הרכיב 74LS244 בנוי מ- 8 מגברים 'חוצצים' כלומר 8 מגברי יחידה.

החוצץ מורכב משתי יחידות, שבכל אחת מהן מותקנים ארבעה מבואות וארבעה מוצאים. לכל אחת מן היחידות מבוא אפשור (G) נפרד.

כאשר אנו מאפשרים את היחידה, כלומר כאשר במבואות G1 או G2 יהיה את הערך '0', אז המבואות של אותה יחידה יועברו למוצאים. להלן טבלת האמת של הרכיב:

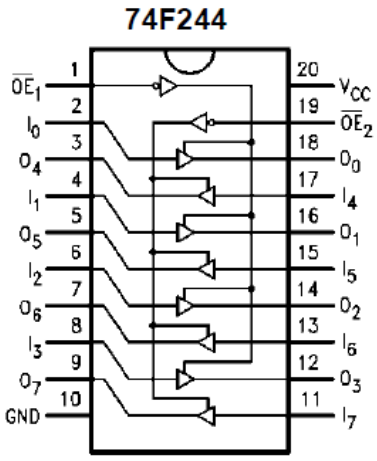
Inputs		Output
$1\overline{G}$,$2\overline{G}$	A	Y
H	X	Z
L	H	H
L	L	L

אנו יכולים לראות שלחוצץ ישנם 3 מצבים ביציאה :

- H- רמה לוגית גבוה.
- L- רמה לוגית נמוכה .
- Z- המצב השלישי , עכבה גבוה .

להלן המבנה המתאר את שתי היחידות הפנימיות ואת המבנה הפנימי של כל יחידה .

לקבלת מידע נוסף יש לעיין בדפי הנתונים.



<u>חיבור הרכיב בפרויקט שלנו :</u>

רגל 20 כניסה 5v.

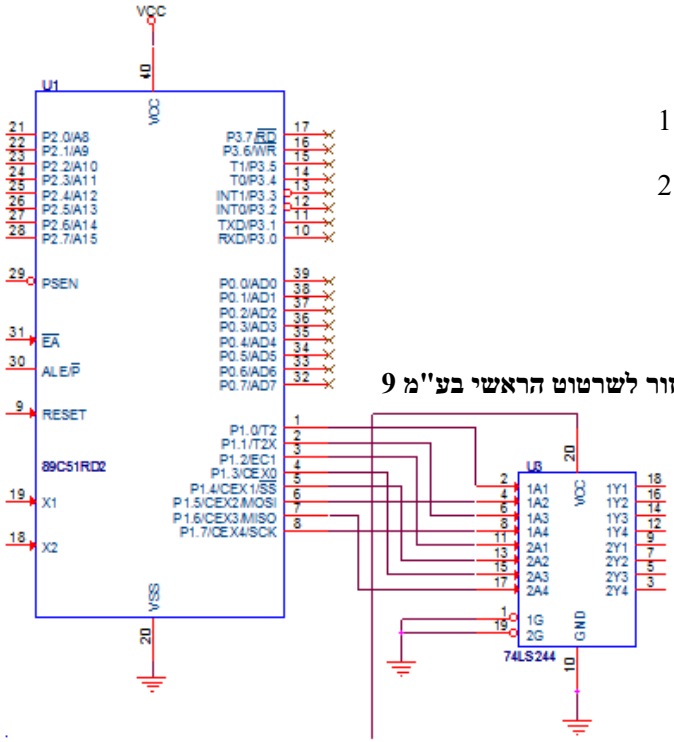
רגליים 15,14 מחוברים ל דוחף 1

רגליים 16,12 מחוברים ל דוחף 2

רגליים 9,7 מחוברים לדוחף 3

רגליים 5,3 מחוברים לדוחף 4

לצפייה בשרטוט הכללי יש לחזור לשרטוט הראשי בע"מ 9



אנו יכולים לראות שאת הדקים 1 ו 19 חיברנו לאדמה מכוון שהרכיב מאופשר ברמה לוגית נמוכה , את היציאה של החוצץ חיברנו לדרייבר של המנועים TLE-5206 2 שעליו נרחיב בהמשך .

כמו כן את החיבור בין החוצץ למיקרו בקר עשינו לפורטים של ה- PWM על מנת שנוכל לשלוט במהירות של הרובוט .

<u>סיכום :</u>

בניסוי זה למדנו מהו תפקידו של הרכיב כיצד הוא מגן על המיקרו בקר ואופן פעולתו

כמו כן למדנו כיצד צריך לחברו למיקרו בקר על מנת שיתפקד בצורה הנכונה , דבר נוסף שהזכרנו שהחוצץ עובד ברמה לוגית נמוכה כלומר החיבור במקרה שלנו הוא לאדמה .

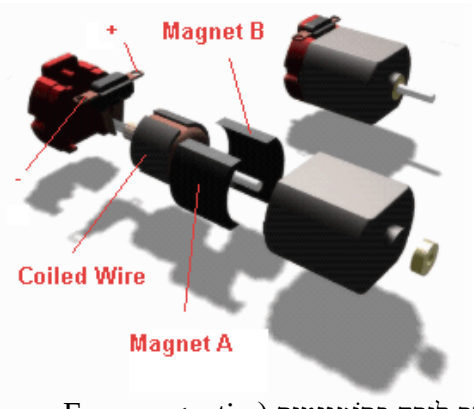
מנועי ה DC ומערכת ההנעה :

מטרת הניסוי:

א. הכרת מנועי הרובוט, מקומם ושימושם.
ב. מדידת טווח הזרמים והמתחים של המנועים.
ג. מדידת נקודות העבודה, עומס של המנועים.
ד. עקרונות המכנים של הרובוט.

רקע תיאורטי:

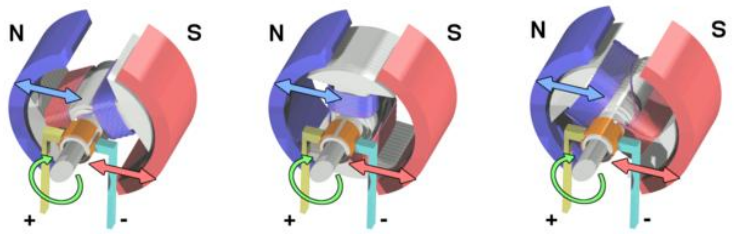
מנוע חשמלי הוא מכונה, הממירה אנרגיה חשמלית לאנרגיה מכנית. מנוע חשמלי מבוסס על עיקרון האלקטרומגנטיות, המאפשר יצירת שדה מגנטי על ידי העברת זרם חשמלי דרך סליל.

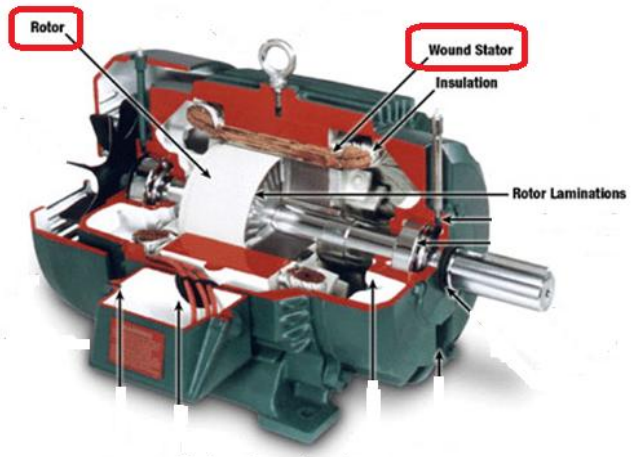


המנוע החשמלי בנוי על פי רוב משני חלקים עיקריים

1. **סטטור (Stator)**: מערכת סלילים המלופפים סביב ליבה פֶרוֹמגנטית (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 זורם 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.





Motor Construction

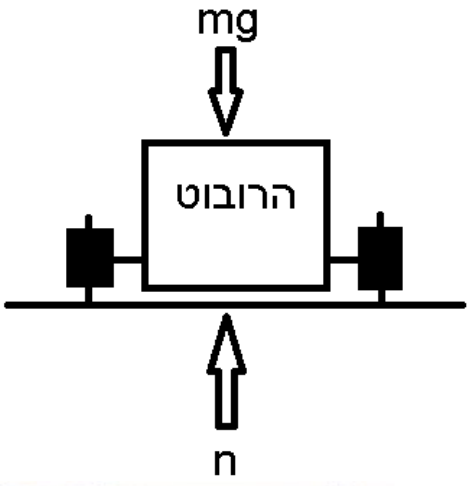
- ברובוט הסיור ישנם סה"כ 4 מנועים, 2 מנועים חזקים בעלי הספק גבוה שמשמשים להנעת הרובוט. ו2 מנועים יחסית קטנים השייכים למנוף המצלמה. כולם DC.
- כל המנועים מחוברים ומניעים מערכת ממסר וגיר המניעים את הרובוט.
- ברובוט נלקחו בחשבון גם עקרונות מכנים על"מ ליעל את המערכת גם מבחינה מכאנית.

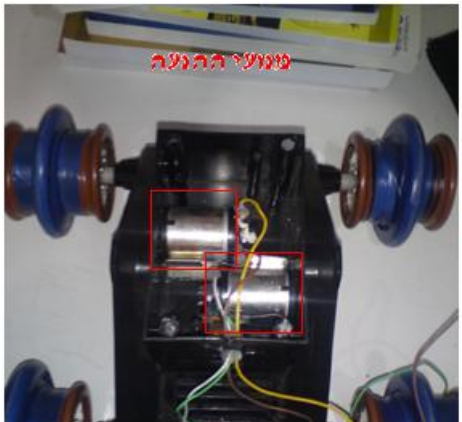
f=mg

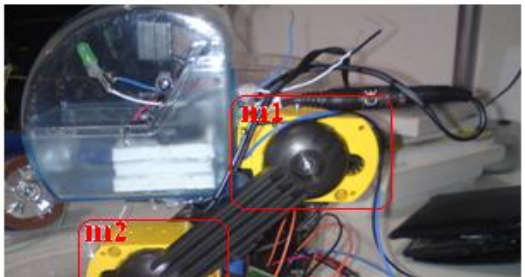
n = הכוח שמפעיל משקל הרובוט על הרצפה

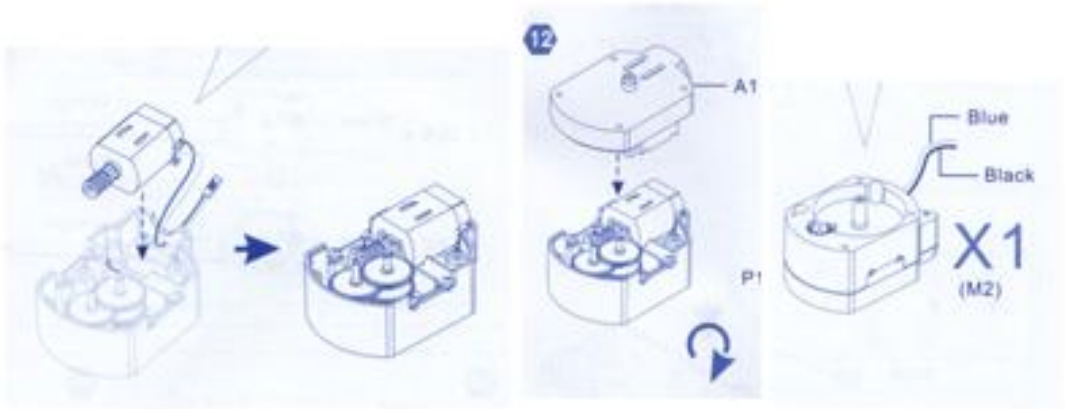
ככל ש mg גדל ז"א משקל הרובוט גדל כך המנועים יותר מתאמצים ומתבזבזת יותא אנרגיה והסוללה נגמרת מהר יותר ונוצר בלאי מוגבר במנועים

לכן לאוך כל זמן יצירת הרובוט נלקחו בחשבון גם עקרונות מכניים אלו על"מ לקבל יעול מירבי



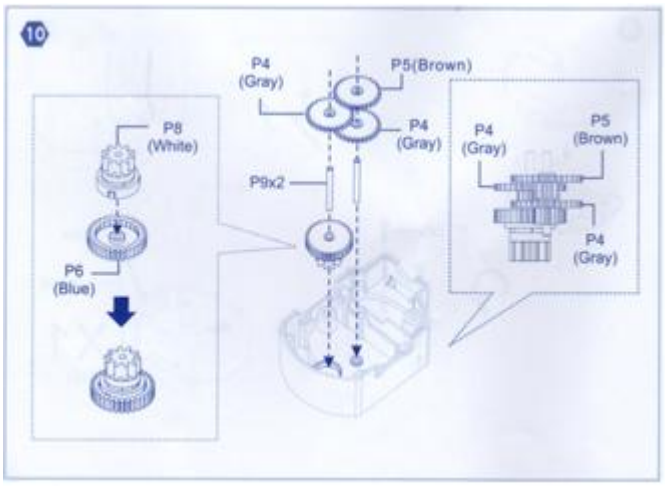


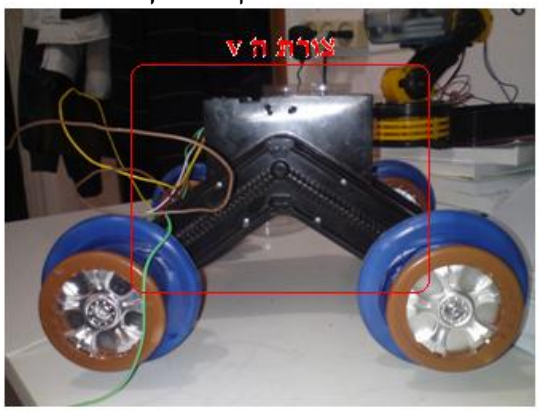




מפרט לגבי מיקום המנועים

מפרט לגבי מבנה מפרקי המנוף





- צורת ה V של בסיס הרובוט מאפשרת לרובוט לשאת משקלים כבדים יותר מכיוון שלא נופל לחץ על המנועים אלא על צירי המרכב וגלגלי שניים המחוברים לשני המנועים בכל צד.
- המנוף מורכב משלושה מנועים (שניים בשימוש) הממוקמים במפרקים ובתוך המפרק ישנה מערכת גלגלי שניים המאפשרים למנוף להישאר במצב האחרון אשר היה. במבנה זה יש יתרון אפילו על מנועי הצעד!.
- בגלגלי הרובוט בוצע שינוי לפי כמה עקרונות מכאניים של חוקי ניוטון.
- בזמן עומס משקל על הרובוט אנו רוצים יהיה כמה שפחות עיוות בצורת הצמיג על"מ לחסוך באנרגיה ז"א שאם הצמיג שטוח המנועים ישקיעו יותר אנרגיה לכן הקטנו את שטח חתך הגלגלים של הרובוט ויצרנו אותם קשיחים בכדי שלא יוצרו שקיעות כמו של צמיג אויר לדוג. ניתן לראות זאת באיורים בעמוד הקודם.

<u>חיבור קבל למנוע הDC:</u>

מדוע חיברנו קבל במקביל למנוע הDC?

חברנו קבל למנוע ה dc מכיוון שהקבל סופג קפיצות מתח בכך שהוא נטען ודרכו מסופק למנוע זרם יותר רציף.

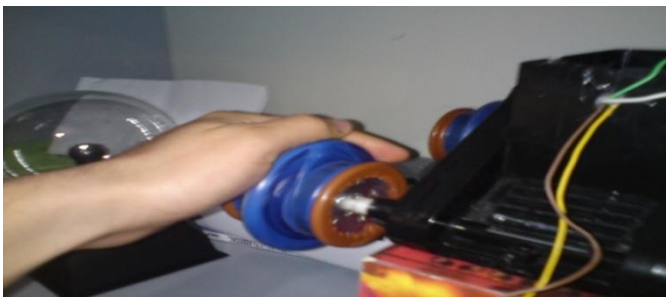
<u>מהלך הניסוי:</u> (מנועי ההנעה) בדיקה כל מנוע בנפרד:

א. מדידת טווח המתח והזרם המינימאליים במנועי ההנעה ללא התנגדות משקל וחיכוך.

(זרם המינימאלי = 0.45A מתח מינימאלי= 0.8V) להנעה מינימאלית

(זרם המינימאלי = 0.75A מתח מינימאלי =10V) להנעה אידיאלית

ב. מדידת טווח המתח והזרם המינימאליים במנועי ההנעה <u>עם</u> התנגדות משקל וחיכוך.



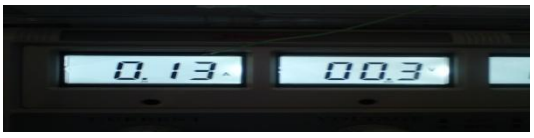
(הזרם המינימאלי = 1.2A מתח מינימאלי = 0.8V) **להנעה מינימאלית**

(הזרם המינימאלי = 1.93A מתח מינימאלי =9.6V) **להנעה אידיאלי**

מסקנת ניסוי: כמה שיש יותר עומס כך צריכת הזרם במנועים גדלה וכך הספק העומס גדל וכתוצאה מכך מתבזבזת יותר אנרגיה.

<u>מהלך הניסוי</u>: (מנועי המנוף)

- מדידת טווח המתח והזרם.



(זרם המינימאלי = 0.13A מתח מינימאלי = 0.3V) **להנעה מינימאלית**

(זרם המינימאלי = 0.18A מתח מינימאלי = 3.2V) **להנעה אידיאלית.**

לסיכום:

בדקנו את יכולת המנועים במצב ללא עומס ולאחר מכל במצב עם עומס בכדיי שנוכל להשוות ולראות מה הזרם והמתח הנדרשים לרובוט בכדי שיוכל נוע בצורה תקינה.

ראינו בדוח שזרם העומס של מנועי ההנעה הוא כמעט שתי אמפר ועלול גם לעלות אם המשקל או העומס על הרובוט יהיה יותר גדול לכן כפי שמצוין בדוח הדוחף (דרייבר) ובדוח הסוללות .. הבנו שאנו צריכים סוללות עם זרם יציאה גדול, מעל 2 אמפר ודוחפים חזקים.. לכן בחרנו בדוחף עם קיבולת זרם עד 6 אמפר.

הפעלת המנוע במהירויות שונות בעזרת PWM:

מטרת הניסוי:

- עבודה עם PWM
- שליטה על מהירות הסיבוב מנוע לזרם ישר(מנוע DC) כולל תצוגה על מסך הLCD.

רקע טאורטי:

אפנון רוחב הפס-בקרת הספק- Modulation) PWM

הקדמה

קיימות מס' שיטות לשלוט על מהירות המנוע.

דרך א'– הינה לספק מתח שונה למנוע ולהיפכך נקבל מהירות שונה, החיסרון הינו שבנגד הטורי המחור למנוי נבזבז הספק והפרוייקט אנו עובדים על סוללות.

דרך ב'-הינה לספק את כל מתח הסוללה כמעט באופן ישיר אך בתדירות קבועה, ובאמצעות אפנון רוחב הדופק תקבע מהירות המנוע.

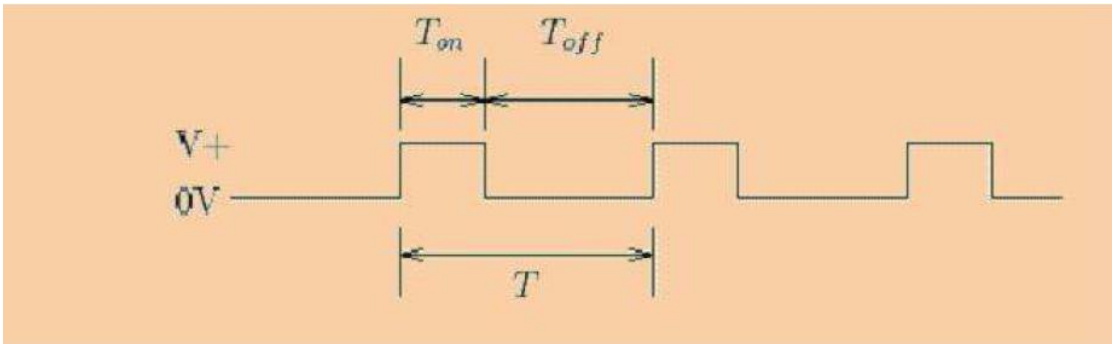
בקרת הספק באמצעות אִפנוּן רוחב פולס-PWM (Pulse Width Modulation) היא שיטה חסכונית ושכיחה לצורך שליטה על הספק חשמלי הנמסר לצרכן בהפעלת מערכות חשמליות מהסוגי מהבאים: עוצמת החימום של גוף חימום, עוצמת הארה של נורת ליבון, **שליטה על מהירות הסיבוב שלמנוע לזרם ישר** ועוד.

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.

בקרת PWM

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(ראו תרשים) :בחלק אחד, המתמשך T_{off} שניות-המתח המסופק הוא0(אפס),וביתרת המחזור,במשך $T_{\text{on}} = \text{T} - \text{T}_{\text{off}}$ -המתח הוא +V.

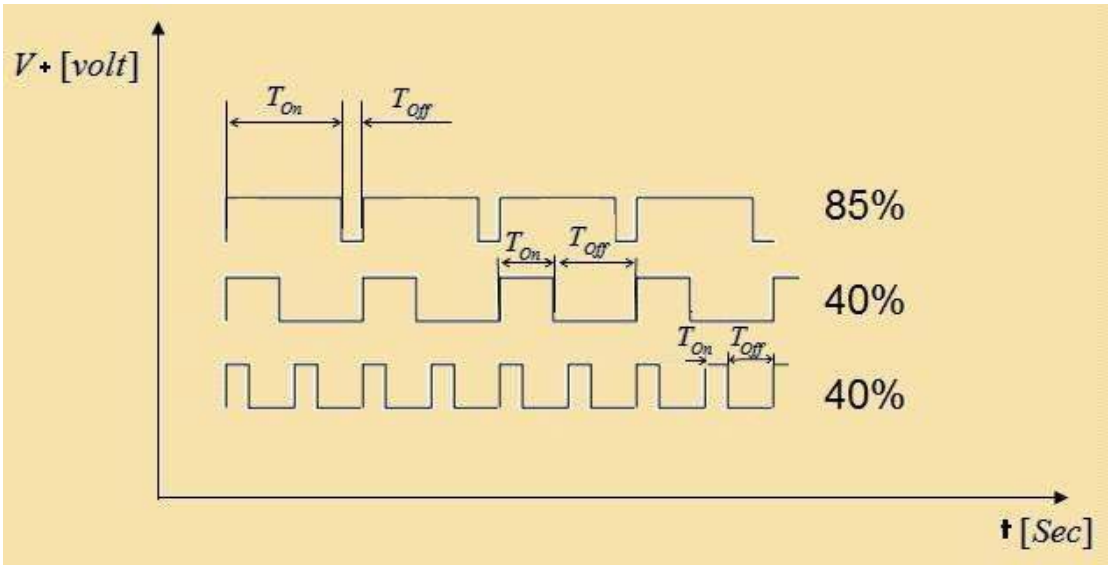
זמן המחזור של כל גל מסומן ב-T,משך הזמן בו נמצא הגל במצב מתח On מסומן ב-T_{on}, ומשך הזמן בו



נמצא הגל במצב מתח Off מסומן ב-T_{off}.

בתרשים הבא מתוארים שלושה גלים שונים, המתארים כל אחד,בהתאמה פונקציית PWM בעלת מאפיינים שונים:

1.הפונקציה העליונה מתארת תהליך, שבו 85% מזמן המחזור T_1 שורר על הצרכן מתח +V, ניתן גם לרשום $T_{\text{on}} = 0.85 \cdot \text{T}_1$.
2. הפונקציה השנייה מתארת תהליך, שבו זמן המחזור T_2 שווה לזמן המחזור של T_1 שבפונקצייה הקודמת: אולם, משך הזמן בו על הצרכן שורר מתח +V, מהווה רק 40% מזמן המחזור $\text{T}_2 = \text{T}_1$. ניתן גם לרשום $T_{\text{on}} = 0.4 \cdot \text{T}_2$.
3. הפונקציה השלישית מתארת תהליך שבו זמן המחזור T_3 שווה למחצית זמן המחזור של כל אחת משתי הפונקציות הקודמות $\text{T}_2 = \text{T}_1 = 2 \cdot \text{T}_3$, משך הזמן שבמהלכו על הצרכן מושרה מתח +V מהווה 40% מזמן המחזור T_3. ניתן גם לרשום $T_{\text{on}} = 0.4 \cdot \text{T}_3$.



<u>הספק בבקרת PWM</u>

ככלל הגודל הפיסקלי הקרוי הספק הוא גודל המתאר קצב (קצב עבודה או קצב של מעבר אנרגיה).
לפיכך זהו גודל רגעי, תלוי בזמן. במערכת חשמלית ניתן לחשב את ההספק הרגעי P(t) בזמן t ע"י מכפה המתח V(t) על הצרכן בזמן t בזרם I(t) העובר דרך הצרכן באותו זמן (t) .

בצורה מתמטית נרשום: $P(t) = v(t) \cdot i(t)$

הספק במעגל חשמלי ניתן גם להגדיר כקצב שינוי האנרגיה במעגל. בשפה המדעית ההספק הוא נגזרת כתלות בזמן של העבודה המתבצעת ע"י הצרכן, או נגזרת כתלות בזמן של פונקציית תהליך מעבר אנרגיה לצרכן.

הגודל הפיסיקלי הקרא "הספק ממוצע" הוא תוצר של תהליך הנמשך פרק זמן נתון. "הספק ממוצע" מתאר את היחס בין כמות (סיכום או אינטגרל) העבודה או כמות האנרגיה, שנמסרה לצרכן או בוצעה ע"י הצרכן, לבין פרק הזמן בו התהליך נמשך $P = \frac{\Delta W}{\Delta t}$.

ע"י שינוי יחס הזמנים בין זמן ההפעלה T_{on} לזמן המחזור T_1 על המנוע מושרה מתח +V או מתח 0 (אפס) בהתאמה. זכרו, ההספק הרגעי Pשל צרכן במעגל חשמלי, שווה למכפלת המתח השורר על הצרכן בזרם I העובר דרך הצרכן $P(t) = v(t) \cdot i(t)$

נניח כאשר על הצרכן שורר מתח +V הזרם I העובר דרך צרכן קבוע. יוצא אפוא, כי ההספק הממוצע של הצרכן נמצא ביחס ישר למשך הזמן בו מושרה מתח +V על הצרכן ובמקרה שלנו ליחס שבין T_{on} ל T.
ככל שהיחס $\frac{T_{on}}{T_{off}}$ גדול יותר, ההספק הנמסר לצרכן גדול יותר.

כלומר, אם נתאר באמצעות גרף את תלות הזרם במתח בכל פרק זמן, ההספק הממוצע הנמסר לצרכן מיוצג באמצעות מכפלת השטח הכלוא בין גרף פונקציית תהליך PWM לבין ציר הזמן בגורם הקבוע $\frac{1}{T}$ בגבולות פרק זמן זה. מכיוון ש $\frac{1}{T}$ הוא גודל קבוע, גם השטח מתחת לגרף המתח כתלות בזמן פרופורציוני להספק.

PWM בבקר שלנו

מבנה מעגל ה PWM

PCA הינו חלק מהאופציות הקיימות ברכיב במסגרת המונה המתוכנת -PWM מעגלה (Programmable Counter Array).בבקר שלנו יש חמישה מחוללים לגל ריבועי שרוחב הדופק שלהם ניתן לשינוי, כלומר חמישה ערוצי PWM מובנים שיכולים לשמש כפורטים רגילים.

<u>ההדקים המשתתפים במערך המונה המתוכנת הם:</u>

P1.2 -ניתן להשתמש בו ככניסה לאות שעון למונה .

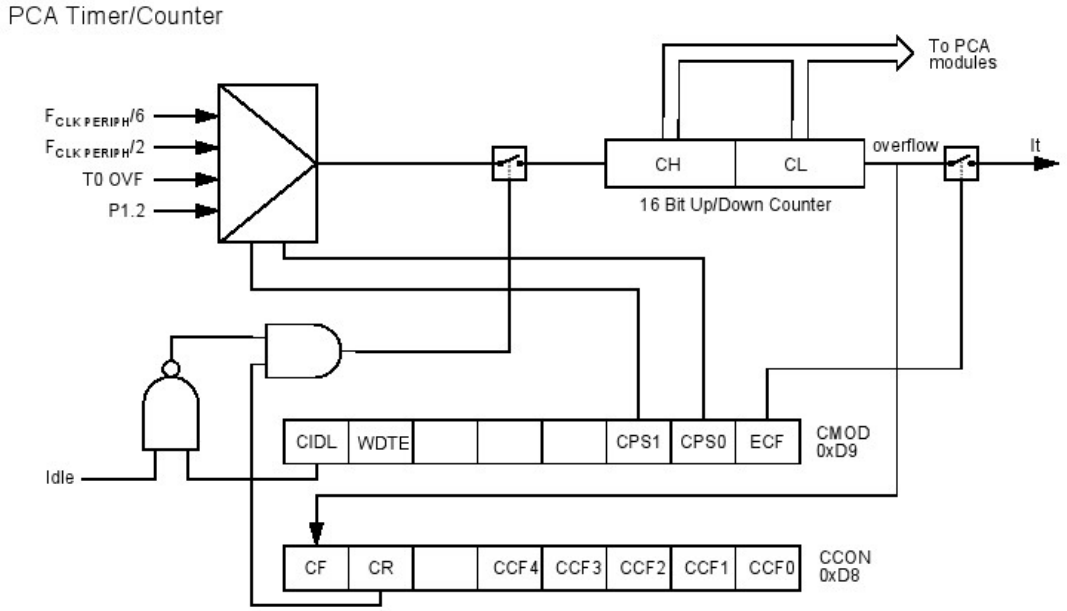
P1.3,P1.4,P1.5,P1.6,P1.7 -5 מוצאים שיכולים בין השאר לשמש ליציאות PWM.

חמשת המחוללים היוצרים את האות הריבועי פועלים במקביל לתבנית הראשית .האותות מתקבלים בחמישה הדקים של Port 1.

ניתן לראות זאת בטבלה הבאה:

ההדק	מרכיב במערך המונים
P1.3/cex0 - לא בשימוש בפרוייקט	16 – bit Module 0
P1.4/cex1	16 – bit Module 1
P1.5/cex2	16 – bit Module 2
P1.6/cex3	16 – bit Module 3
P1.7/cex4	16 – bit Module 4

חמשת ערוצי ה -PWM הם חלק ממערך המונים המותקנים במיקרו בקר .מערך המונים PCA (Programmable Counter Array),כולל חמש יחידות (Modules) שאפשר לתכנת לאופני עבודה שונים (כאשר אחד מהם הוא ייצור של אות PWM).



מערך המונים מורכב ממונה משותף ברוחב של 16 סיביות .המונה מורכב משני אוגרים המסומנים ב-CH ו-CL .

רוחבו של כל אוגר הוא שמונה סיביות .מתכנתים את האוגרים באמצעות שני האוגרים המיוחדים CMOD ו-CCON .

באמצעות האוגר CMOD בעדכון ערכי הביטים CPS0 ו -CPS1- אנו נבחר את קצב העבודה של השעון האחראי על אפנון רוחב הדופק. ערך תדר ה-PWM תלוי בשעון של PCA.לכל חמשת ערוצי ה-PWM קיים אותו תדר, מכיוון שלכולם יש במשותף את אותו שעון ה-PCA. בחרנו בפרוייקט ,קצב חלוקה של 6, ז"א ערכי הביטים ב-CPS0 ו-CPS1 הינם "0".

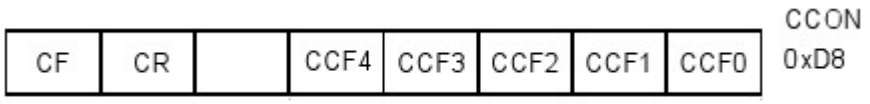
CMOD Register

CMOD - PCA Counter Mode Register (D9h)

7	6	5	4	3	2	1	0
CIDL	WDTE	-	-	-	CPS1	CPS0	ECF

Bit Number	Bit Mnemonic	Description
2	CPS1	**PCA Count Pulse Select** CPS1 CPS0 Selected PCA input 0 0 Internal clock $f_{CLK\ PERIPH}/6$ 0 1 Internal clock $f_{CLK\ PERIPH}/2$ 1 0 Timer 0 Overflow 1 1 External clock at ECI/P1.2 pin (max rate = $f_{CLK\ PERIPH}/4$)
1	CPS0	

באוגר המיוחד CCON מותקנים דגלי הפסיקה (סיביות CCFn) של חמשת היחידות של מערך ה-PCA.
הדגלים האלה אינם פעילים במצב העבודהPWM. הסיבית השביעית באוגר CCON היא סיבית הבקר CR.
באמצעות אוגר זה אנו נאפשר את ריצת השעון כאשר נעלה את הערך של ה CR ל- "1".

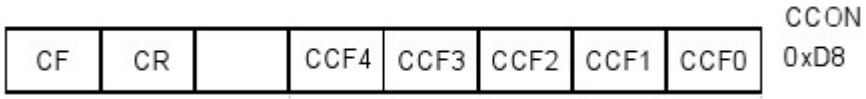


שמונה הסיביות של האוגר המיוחד CCON

ההוראה המפעילה את המונה היא:

אפשר למונה לרוץ// ; CCON=0x40

0	1	0	0	0	0	0	0



חובה להציב ב-CR ע"י התוכנה ערך של ערך "1 " לוגי כדי שה -PCA ירוץ, עצירת ה-PCA נעשית ע"י איפוס סיבית זאת

.

נוסף על המונה המשותף לכל היחידות, לכל יחידה (Module) יש שלושה אוגרים משלה:

1)האוגר CCAPnH.

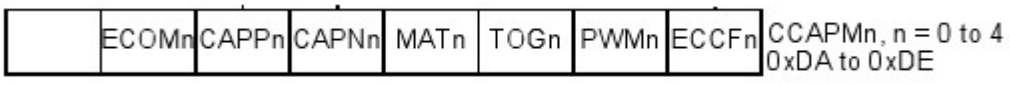
2)האוגר CCAPnL.

3)האוגר CCAPMn.

האות nמייצג את מספר היחידות (n= אחת מן הספרות 0,1,2,3,4) ובהתאמה את הדקי פורט 1.

תפקידם של שני האוגרים CCAPnL ו- CCAPnH הוא לאחסן את הנתונים של היחידה, ברוחב של 16 סיביות. תפקידו של אוגר הCCAPM הוא לקבוע את אופן העבודה של היחידה

תאור של שמונה הסיביות של האוגר CCAPMn, שבאמצעותם קובעים את אופן העבודה של היחידה.



כדי שהיחדה תפעל באופן העבודה PWM(אפשור), יש להציב בסיבית PWMn של האוגר (הסיבית השניה) ובסיבית ECOMn של האוגר (הסיבית השביעית) של הערך "1" לוגי, ובשאר הסיביות להציב "0" לדוגמא:כדי הגדיר את מודול 1 (הדק P1.4), כערוץ PWM יש לכתוב את ההוראה הבאה:

מודול 1 מוגדר באופן עבודה של // :CCAPM1=0x42

0	1	0	0	0	0	1	0

	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	CCAPMn, n = 0 to 4 0xDA to 0xDE

על פי אותו עקרון, ההוראה : CCAPM2=0x42 מגדירה את יחידה 2 כערוץ PWM, באותו אופן מגדירים גם את היחידות האחרות (3 ו-4).

אופן השליטה על ה*PWM*

בתרשים הבא מוצג מבנה עקרוני של ערוץ PWM, התרשים מתאר את אופן הפעולה של 5 המודולים, המבוסס על שימוש ב-3 אוגרים:

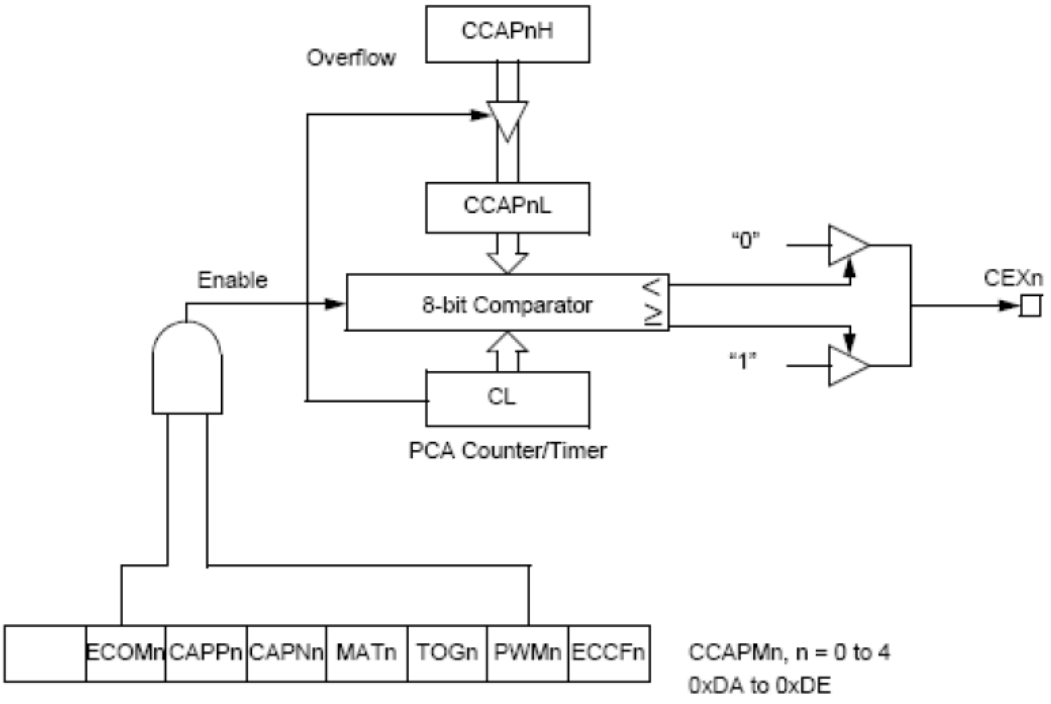
1) האוגר CCAPnH.

2) האוגר CCAPnL.

3) האוגר CCAPMn.

כל אחד מחמשת ערוצי ה-PWM מורכבים באופן הבא, תרשים עעקרוני של אופן הפעולה.

Figure 30. PCA PWM Mode



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

Bit Number	Bit Mnemonic	Description
6	ECOMn	**Enable Comparator** Cleared to disable the comparator function. Set to enable the comparator function.
1	PWMn	**Pulse Width Modulation Mode** Cleared to disable the CEXn pin to be used as a pulse width modulated output. Set to enable the CEXn pin to be used as a pulse width modulated output.

PCA Module Modes (CCAPMn Registers)

ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMm	ECCFn	Module Function
1	0	0	0	0	1	0	8-bit PWM

הסבר:

- קביעה באוגר CCAPMn "1" לוגי בסיביות -ECOMn ו-PWMm לקבלת עבודה של PWM.
- קביעת קצב העבודה של השעון ע"י ערכי הביטים CPS0 ו- CPS1 באוגר CMOD.
- תוכן האוגר CCAPnH נטען אל האוגר CCAPnL

❖ המונה CL שרוחבו 8 סיביות, גדל ב-1.

❖ מתבצעת השוואה בין הערך של המונה CL ובין הערך של האוגר CCAPnL,כל עוד מתקיים CL≥התנאי כאשר מתקיים (CEXn). מתקבל "0 " במוצא הערוץ CCAPnL > CL ,

(CEXn).מתקבל "1 " במוצא הערוץCCAPnL ,

נטען שוב אל האוגר ,(-00 CCAPnH תוכן האוגר - 0 (FF גולש מ 255 -ל CL כאשר המונה♉ .CCAPnL

נ

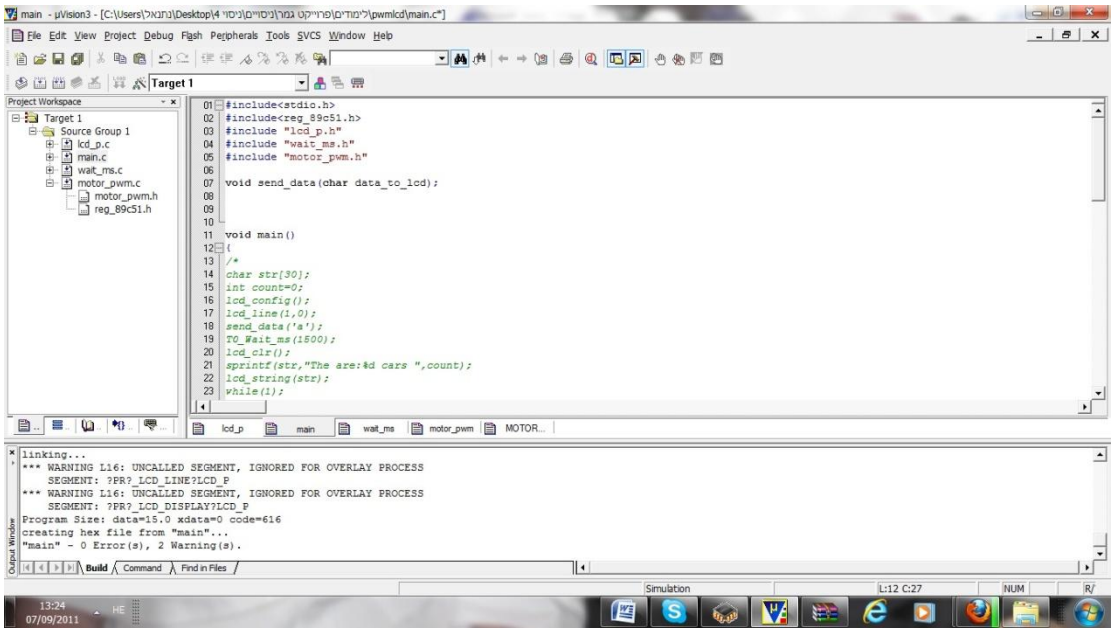
וצר גל ריבועי PWM .במוצאו של ערוץ♉

הוא שמונה סיביות ,ערך המנייה המירבי שלו הוא . 255 לדוגמא ,אם CL היות שרוחבו של המונה בערך של , 100 אזי במשך זמן של 100 דפקים יתקבל "0 " במוצא CCAPnH טוענים את המונה הערוץ ,ולאחר מכן במשך זמן המנייה של 156 דפקים נוספים יתקבל "1 " במוצא הערוץ .לפיכך, גורם המחזור הוא ,בקירוב ,שווה ל , 0.609 -כי במשך 156/256 מזמן המחזור מתקבל "1 " במוצא הערוץ.

PCA.במערך המונים -PWM על מנת לשלוט על 2 מנועים בו -זמנית ,נגדיר שני אופני עבודה של ה של כל -D.C של מנוע שמאל .ה PWM של מנוע ימין ומונה 2 עבור בקרת PWM מונה 1 עבור בקרת יחידה תלוי בשינוי השימוש ביחידות הלכידה באוגר.CCAPLn

<u>הניסוי:</u>

כתבנו את התוכנית הבאה בתוכנה ה KELL UVISIONS : (בשפת C)



התוכנית:

Main:

```
#include<stdio.h>// sprint() ספריה שכוללת פקודות קלט/פלט ואחראית על פרודות
#include<reg_89c51.h>//  צירוץ פונקצית ספריה למיקרו בקר
#include "lcd_p.h"// LCD קובץ פקודות פקודות לתצוגת
#include "wait_ms.h"//  קובץ פקודות השהייה
#include "motor_pwm.h"// PWM קובץ פקודות של המנוע


void main()

{
lcd_config();//פונקציה שמאפשרת לתצוגה לעבוד
lcd_clr();//פונקציה שמנקה את התצוגה
lcd_string("    go street   ");// go street מציג על התצוגה את המשפט
motors(90,90);//motor forward// T זמן של 255 מתוך 90 הם המנועים של PWM ה
T0_Wait_ms(3000);//השהייה של 3 שניות
motors(0,0);//stop motors
lcd_clr();//פונקציה שמנקה את התצוגה
lcd_string("    ZROA A   ");// ZROA A מציג על התצוגה את המשפט
motors_zroa(50,0);// PWM  255 מתוך PWM  40 זזה קדימה מהירות של zroa a
T0_Wait_ms(1500);//השהייה של דקה וחצי
motors_zroa(0,0);//  STOP MOTORS ZROA
motors_zroa(-50,0);//  PWM  -255 מתוך PWM  -40 זזה אחורה מהירות של zroa a
```

```
T0_Wait_ms(1500);// השהייה של דקה וחצי
motors_zroa(0,0);// STOP MOTORS ZROA
T0_Wait_ms(500);// השהייה של חצי דקה


lcd_clr();//פונקציה שמנקה את התצוגה
lcd_string("    go back     ");// go back מציג על התצוגה את המשפט
motors(-90,-90);//   T  של זמן -255 מתוך -90 הם המנועים של PWM
T0_Wait_ms(3000);//השהייה של 3 שניות
motors(0,0);//stop motors
lcd_clr();//פונקציה שמנקה את התצוגה
lcd_string("    zroa b     ");// go back מציג על התצוגה את המשפט
motors_zroa(0,50);// PWM  255 מתוך PWM  40 של zroa b זה קדימה מהירות
T0_Wait_ms(1500);//השהייה של דקה וחצי
motors_zroa(0,0);// STOP MOTORS ZROA
motors_zroa(0,-50);// PWM  -255 מתוך PWM  -40 של zroa b זה קדימה מהירות
T0_Wait_ms(1500);//השהייה של דקה וחצי
motors_zroa(0,0);// STOP MOTORS ZROA
T0_Wait_ms(500);//השהייה של חצי דקה

lcd_clr();//פונקציה שמנקה את התצוגה
lcd_string("    go left    ");// go left מציג על התצוגה את המשפט
motors(-190,190);   //LEFT

T0_Wait_ms(3000);//השהייה של 3 שניות
motors(0,0);//stop motors
T0_Wait_ms(500);//השהייה של חצי דקה

lcd_clr();//פונקציה שמנקה את התצוגה
lcd_string("    go right   ");// go right מציג על התצוגה את המשפט
motors(190,-190);   //RIGHT
T0_Wait_ms(3000);//השהייה של 3 שניות
motors(0,0);//stop motors
T0_Wait_ms(500);//השהייה של חצי דקה
while(1); //  לולאה

}//end main
```

<u>הסבר על קובץ הפקודות של המנוע PWM "motor_pwm.c":</u>

קובץ זה מסביר כיצד יצרנו את הפקודות הבאות:

`motors(90,90);` = שולט על מהירות וכיוון מנועי הגלגלים

`motors_zroa(0,50);`= ושולט על מהירות וכיוון מנועי הזרועה

שבעזרתם אנו מפעילים את המנועים וקובעים באיזה מהירות ובאיזה כיוון הם ינועו (בעזרת PWM)

```
#include "motor_pwm.h"


/*
P1.3  CCAP0
P1.4  CCAP1
P1.5  CCAP2
P1.6  CCAP3
P1.7  CCAP4
*/


motors(int left,int right)//  יצירת מבנה הפקודה
{
motor_left(left);//P1.0,P1.5
motor_right(right);//P1.1,P1.7
}

motor_left(int pwm) //P1.0,P1.5
{
      CCAPM2=0X42; //enable pwm 2
      CCON=0X40;  // PCA counter run
      if(pwm>=0)//  בודק אם אחורה או קדימה
      {
            PHASE_MOTOR_LEFT=1;//P1.0- קדימה
            CCAP2H=pwm;  //P1.5- PWM כ 1.5 מגדיר את פורט
      }
      else
      {
            PHASE_MOTOR_LEFT=0; //P1.0- אחורה
            CCAP2H=pwm; //P1.5- PWMכ 1.5 מגדיר את פורט
      }
}


motor_right(int pwm)
{
      CCAPM4=0X42; //enable pwm 4
      CCON=0X40;   // PCA counter run
      if(pwm>=0) // בודק אם אחורה או קדימה
      {
            PHASE_MOTOR_RIGHT=1; //P1.1-קדימה
            CCAP4H=pwm; //P1.7 - PWMכ 1.7 מגדיר את פורט
      }
      else
      {
            PHASE_MOTOR_RIGHT=0; //P1.1-אחורה
            CCAP4H=pwm; //P1.7 - PWMכ 1.7 מגדיר את פורט
      }
}



 /*---------------motors zroa------------------------------



motors_zroa(int zroa_a,int zroa_b)// יצירת מבנה הפקודה
{
motor_zroa_a(zroa_a);// p1.4,p1.2
```

```
motor_zroa_b(zroa_b);// p1.3,p1.6
}

motor_zroa_a(int pwm) // p1.4,p1.2
{
	CCAPM1=0X42; //enable pwm 1
	CCON=0X40;  // PCA counter run
	if(pwm>=0) // בודק אם אחורה או קדימה
	{
		ZROA_A=1;//P1.2 -קדימה
		CCAP1H=pwm;  //P1.4- PWMכ 1.4 מגדיר את פורט
	}
	else
	{
		ZROA_A=0; //P1.2 -אחורה
		CCAP1H=pwm; //P1.4- PWMכ 1.4 מגדיר את פורט
	}
}


motor_zroa_b(int pwm) // p1.3,p1.6
{
	CCAPM3=0X42; //enable pwm 3
	CCON=0X40;   // PCA counter run
	if(pwm>=0) // בודק אם אחורה או קדימה
	{
		ZROA_B=1; //P1.3 - קדימה
		CCAP3H=pwm; //P1.6
	}
	else
	{
		ZROA_B=0; //P1.3 - אחורה
		CCAP3H=pwm; //P1.6

	}
}
```

<u>הסבר על הקוד:</u>

יצרנו את המבנה של הפקודה שיכול להפעיל 2 מנועים אחר כך הכנסנו ל אוגר CCAPM3 את הערך 42H על מנת שיאופשר כ PWM אחר כך הכנסנו ל 40H CCON בשביל להפעיל את המונה

ואז עשינו השוואה אם המספר שאנו מכניסים יהיה גדול מ 0 אז פורט 1.3 יקבל "1" לוגי ופורט 1.6 יוגדר כ PWM (יסע קדימה, ובמיהרות שהכנסנו) , ואם המספר שהכנסנו יהיה קטן מ 0 אז פורט 1.3 יקבל "0" לוגי ופורט 1.6 יוגדר כ PWM (יסע אחורה, ובמיהרות שהכנסנו), כל 4 המנועים מוגדרים כך.

<u>הסבר על התוכנית:</u>

הרובוט יסע קדימה במהירות של 90(PWN) כ 3 שניות ויציג בתצוגת ה LCD את המשפט "go street" ויעצור

לאחר מכן ZROA A תתחיל לנוע קדימה למשך דקה וחצי ותנוע אחורה למשך דקה וחצי ותציג התצוגת ה LCD את המשפט "ZROA A" .

לאחר מכן הרובוט יסע אחורה במהירות של 90-(PWM) למשך 3 שניות ויציג בתצוגת הLCD את המשפט " go back"

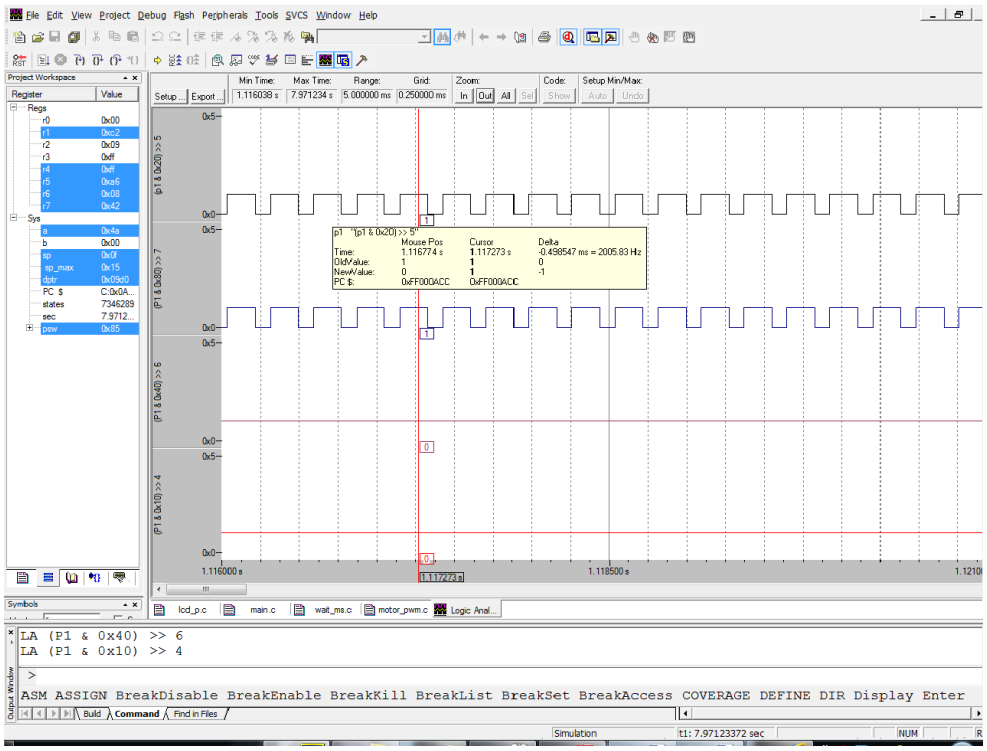
אחר כך ZROA B תתחיל לנוע קדימה למשך דקה וחצי ולאחר מכן תנוע אחורה גם למשך של דקה וחצי ותציג בתצוגת ה LCD את המשפט" zroa b" .

לאחר מכן הרובוט יסתובב שמאלה למשך 3 שניות ויציג בתצוגת הLCD את המשפט " go left" .

ואחר כן יסתובב ימינה למשך 3 שניות ויציג בתצוגת הLCD את המשפט " go right"

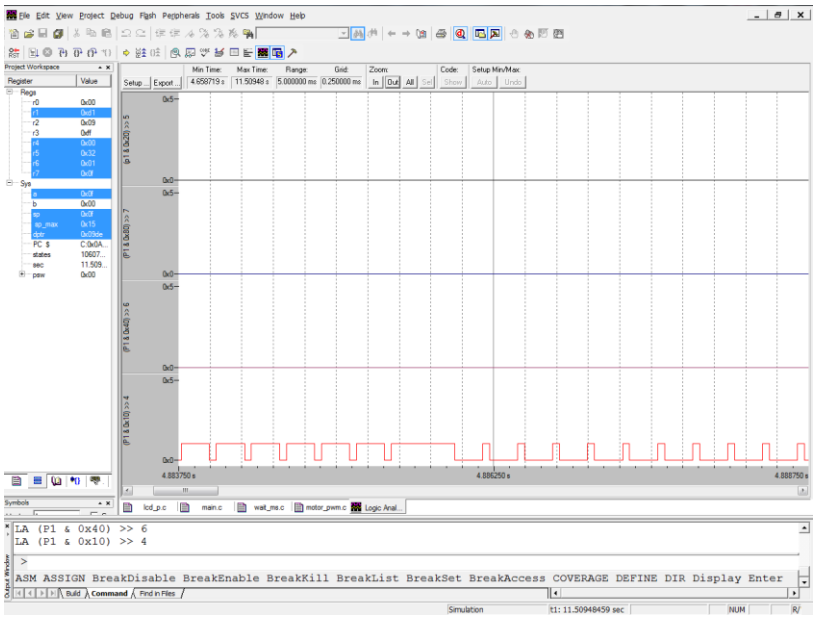
<u>לאחר שכתבנו את הקוד קימפלנו אותו ועשינו סימולציה ואלה התצאות:</u>

כאשר הרובטו נוסע קדימה:



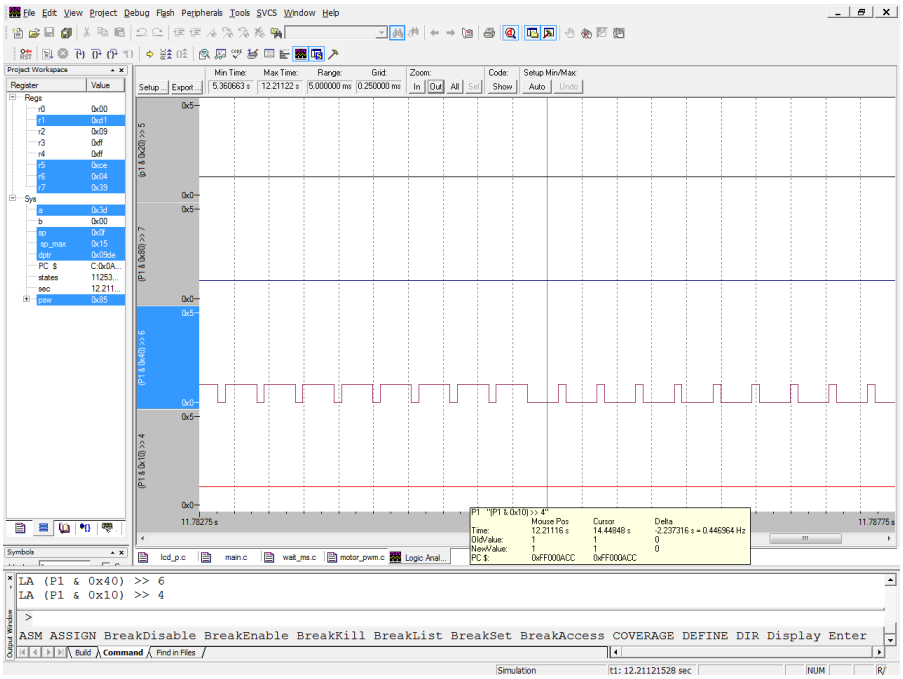
אנו יכולים לראות בסימולציה ש 2 פורטים 1.7 ו-1.5 מוגדרים כ PWM בשתיהם במשך 90 דפקים המצב הוא "1" ואילו כל השאר במצב "0" לוגי, 2 המנועים נוסעים קדימה במהירות PWM 90

כאשר זרוע הרובוט (motor zroa a) זזה קדימה ואחורה:



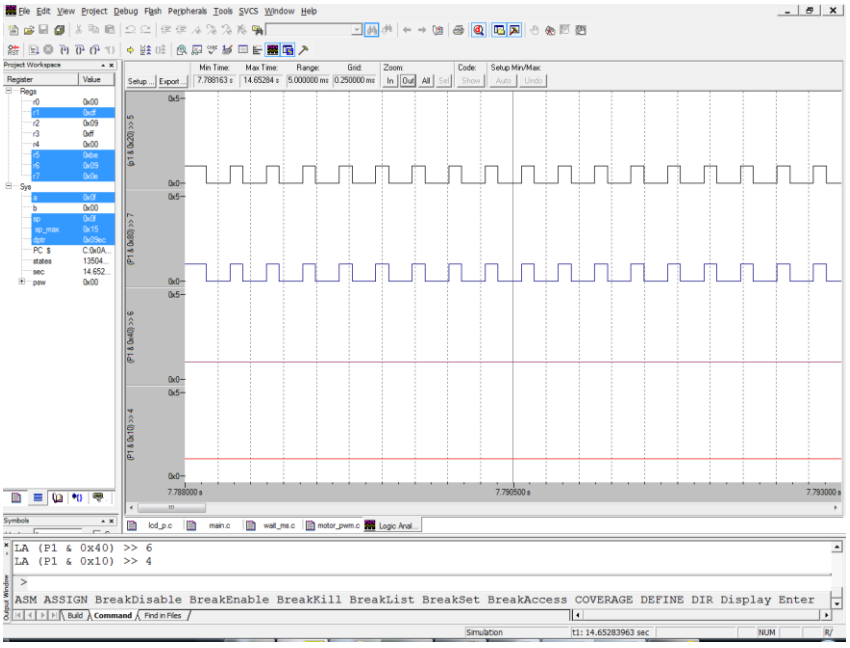
אנו יכולים לראות בסימולציה ש פורט 1.4 מוגדר כ PWM במשך 50 דפקים המצב הוא "1" ואילו כל השאר במצב "0" לוגי(זז קדימה) , ומיד אחר כך הוא מתהפך במשך 50 דפקים המצב הוא "0" ואילו כל השאר במצב "1" לוגי(זז אחורה)

כאשר זרוע הרובוט (motor zroa b) זזה קדימה ואחורה:



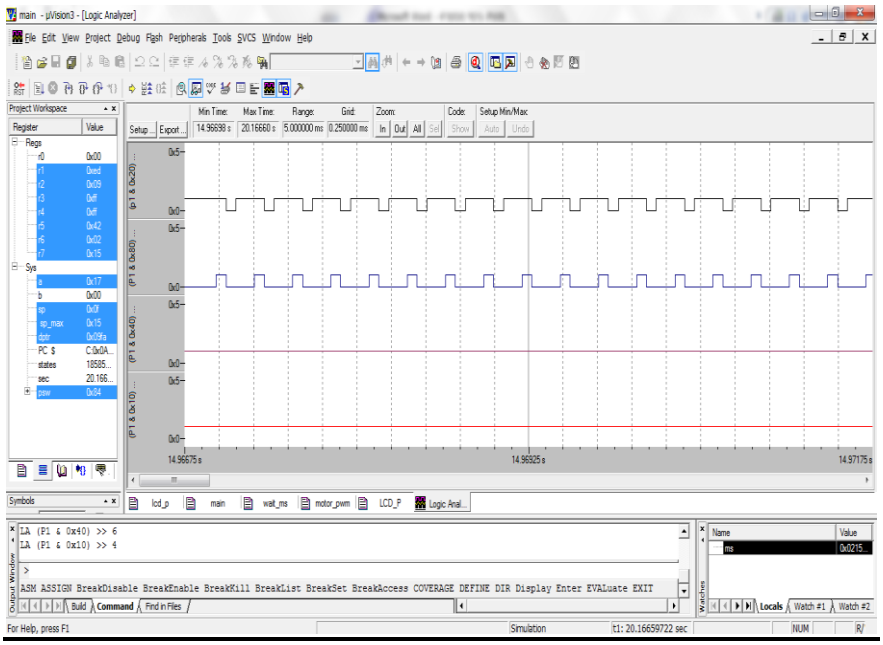
אנו יכולים לראות בסימולציה ש פורט 1.6 מוגדר כ PWM במשך 50 דפקים המצב הוא "1" ואילו כל השאר במצב "0" לוגי(זז קדימה) , ומיד אחר כך הוא מתהפך במשך 50 דפקים המצב הוא "0" ואילו כל השאר במצב "1" לוגי(זז אחורה).

כאשר הרובוט נוסע אחורה:



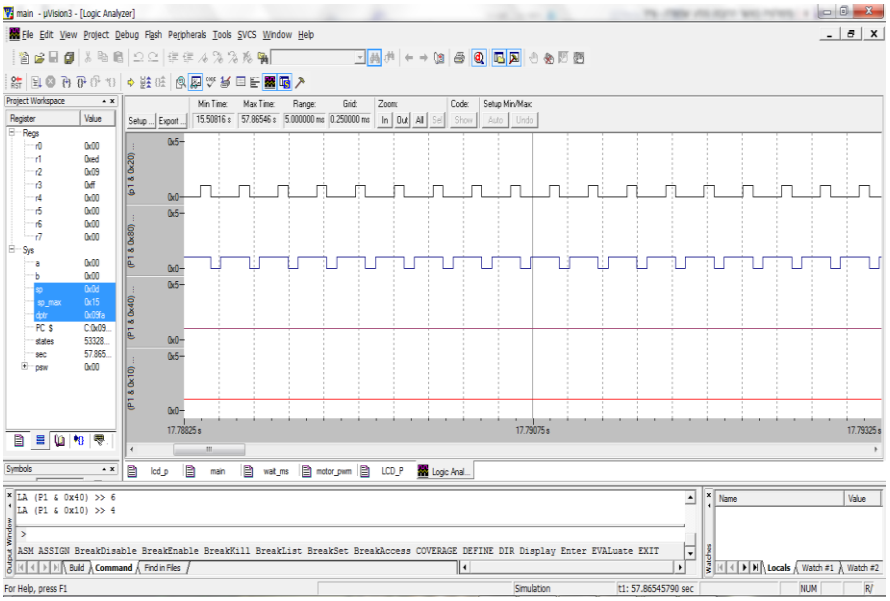
אנו יכולים לראות בסימולציה ש 2 פורטים 1.7 ו-1.5 מוגדרים כ PWM בשתיהם במשך 90-דפקים המצב הוא "0" ואילו כל השאר במצב "1" לוגי, 2 המנועים נוסעים אחורה במהירות 90-PWM

כאשר הרובוט נוסע שמאלה:



אנו יכולים לראות בסימולציה ש 2 הפורטים 1.7 ו-1.5 מוגדרים כ PWM וזזים בכיוונים שונים, פורט 1.7 במשך 190 דפקים המצב הוא "1" ואילו כל השאר במצב "0" לוגי (זז קדימה), ואילו פורט 1.5 במשך 190- דפקים המצב הוא "0" ואילו כל השאר במצב "1" לוגי(זז אחורה)

כאשר הרובוט נוסע ימינה:



אנו יכולים לראות בסימולציה ש 2 הפורטים 1.7 ו-1.5 מוגדרים כ PWM וזזים בכיוונים שונים, פורט 1.7 במשך 190- דפקים המצב הוא "0" ואילו כל השאר במצב "1" לוגי (זז אחורה), ואילו פורט 1.5 במשך 190 דפקים המצב הוא "1" ואילו כל השאר במצב "0" לוגי(זז קדימה)

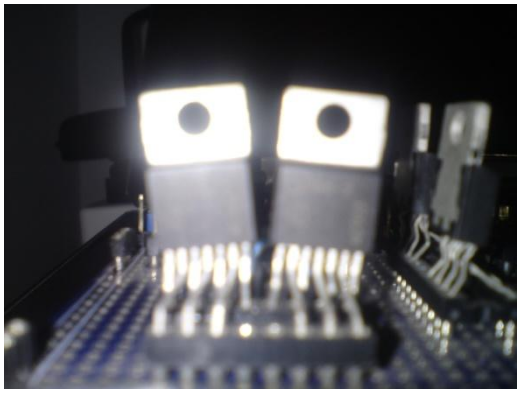
קימפלנו את התוכנית וצרבנו לבקר

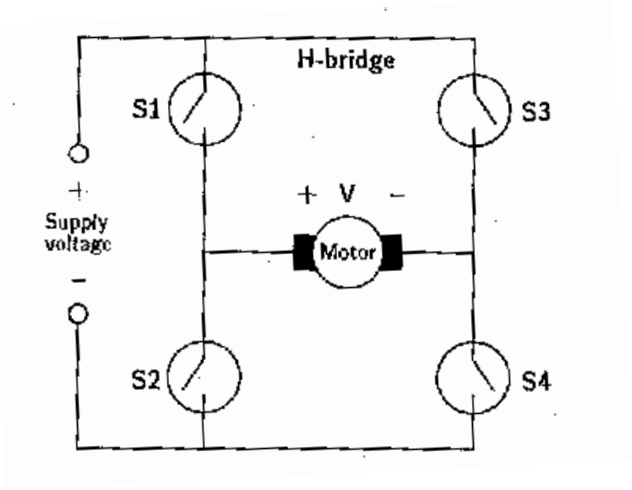
דוחף למנוע TLE 5206-2 :

המיקרו בקר אינו מסוגל להפעיל מנוע באופן ישיר מאחר והוא לא יכול לספק את הזרם החשמלי שדרוש למנוע. כתוצאה מכך, יש צורך במעגל חשמלי "מגשר", כך שהכוח הדרוש למנוע יגיע ממקור מתח אחר והמיקרו בקר רק ישלח אותות להפעלת המנוע.

מעגל חשמלי זה ניתן ליישום באמצעות H-Bridge .

H-Bridge כולל ארבעה מפסקים שמחוברים בצורת H כאשר המנוע נמצא במרכז ה-H.

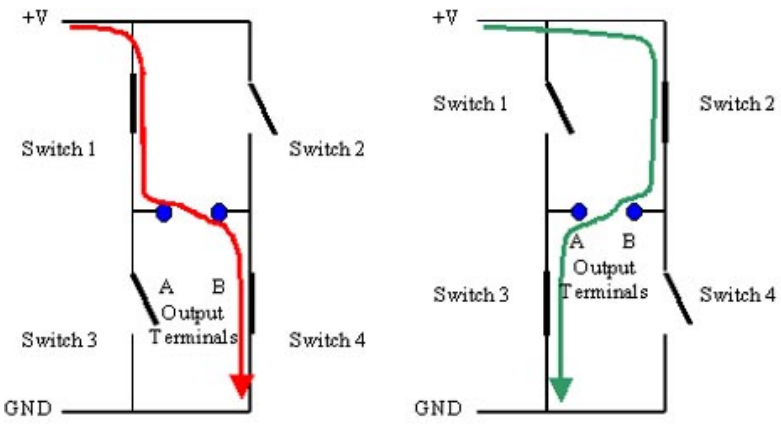




כדי להפוך את כיוון פעולתו של המנוע יש לשנות את כיוון הזרם במעגל.

לשם כך, יש לפתוח ולסגור מפסקים מסוימים.

בתרשים ניתן לראות את פעולת המפסקים:



כאשר המפסקים 1 ו-4 סגורים A מחובר להדק החיובי ו-B להדק השלילי.

כאשר המפסקים 2 ו-3 סגורים יש שינוי במצבם של A ו-B. A מחובר להדק השלילי ו-B להדק החיובי.

כתוצאה מכך, המנוע נע בכיוון ההפוך.

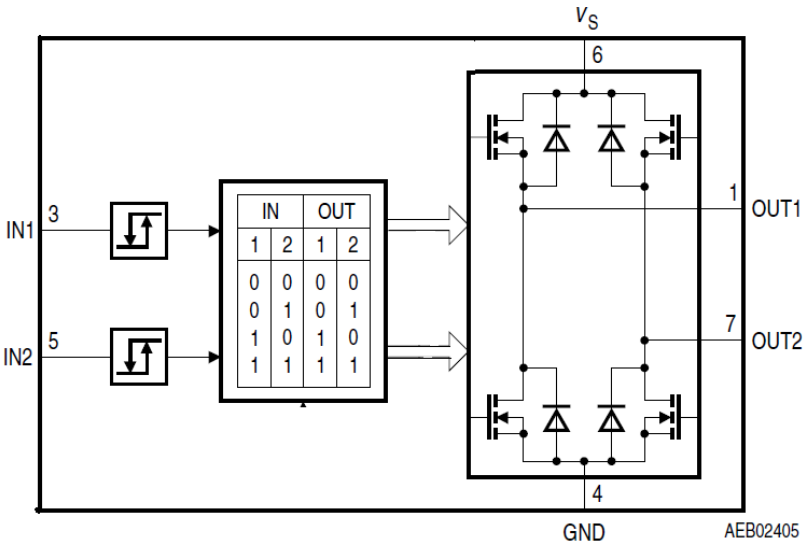
בנוסף, יש נם מעגלי H-Bridge שהם מוליכים למחצה.

במעגלים אלו במקום מפסקים ישנם טרנזיסטורים : שניים מסוג PNP ושניים מסוג NPN.

בדומה למפסקים יש להפעיל שני טרנזיסטורים שנמצאים בצדדים שונים של המעגל.

הפעלת שני טרנזיסטורים שנמצאים למעלה או למטה לא יניע את המנוע.

אנו בפרויקט השתמשנו ב- H-Bridge for DC-Motor Applications TLE 5206-2.



<u>הסבר על היציאות השונות-</u>

OUT1 - יציאת מתח (pwm/מתח ישר) מהדרייבר אל המנוע.

OUT2 - - יציאת מתח (pwm/מתח ישר) מהדרייבר אל המנוע.

IN1 - כניסת אות לוגית (pwm/מתח ישר) מהבקר אל הדרייבר

IN2 - כניסת אות לוגית (pwm/מתח ישר) מהבקר אל הדרייבר

GND - הארכה/אדמה

VD - מתח הזנה של הרכיב על פי מתח הזנה זה, ניתן לקבוע את המתח המקסימאלי ביציאות

השונות.

ברגל 3 ו5 אנו רואים שניתן להכניס אות pwm לדרייבר.

על פי הטבלת אמת המוצגת לעיל, אנו יכולים לקבוע את כיוון המתח הרצוי ועל ידי כך לשנות את כיוון המנוע.

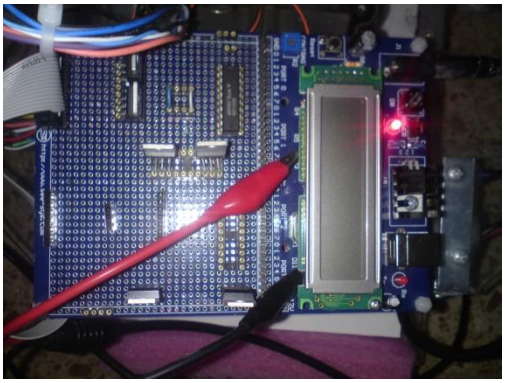
הדרייבר עובד ע"פ שיטת H-Bridge for המוסברת בתחילת הדף, ועל ידי כך נותן לנו את המתח/ הזרם הרצוי במנוע ומסייע לנו לקבוע את כיוון הרובוט.

המיתוג נעשה על ידי הטרנזיסטורים .

<u>מהלך הניסוי :</u>

בכדי להבין את המוסבר לעיל טוב יותר אנו נבצע ניסוי ואנו נבדוק מה למעשה הדרייבר מבצע .

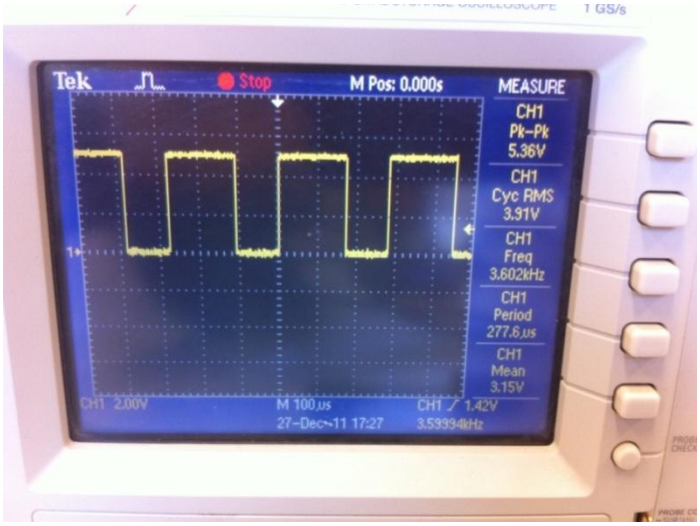
בהתחלה בדקנו את מתח הסוללה , חברנו רב מודד ל(+) של הסוללה ול (–) של הסוללה :



לאחר מכן אנו נחבר את הסקופ לבקר, את התנין השחור לGND (אדמה) ואת התנין האדום פעם אחת לפורט P1.5 ופעם לפורט P1.7 כמו כן נבדוק גם את P1.4 ואת P1.6 .

אנו נראה את הגלים ואת התדר שיש לנו בסקופ:

<u>פורט P1.5 :</u>

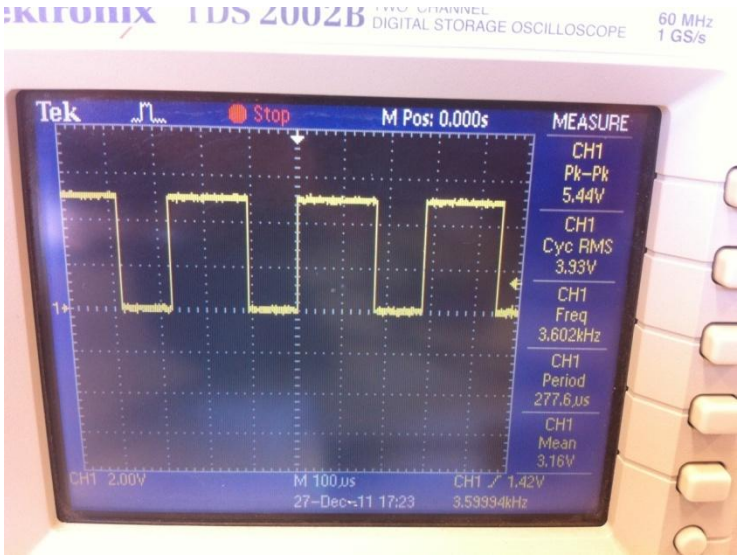


אנו רואים שהמתח הממוצע Vmean הוא: 3.15V

התדר FREQ הוא : 3.602Khz .

והמתח PK-PK הוא : 5.36V

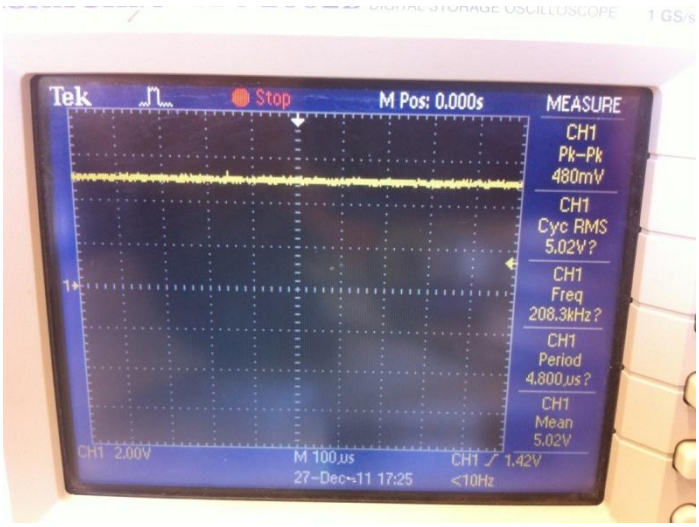
<u>פורט P1.7 :</u>



אנו רואים שהמתח הממוצע Vmean הוא: 3.16V

התדר FREQ הוא : 3.602Khz והמתח PK-PK הוא : 5.44V

<u>פורט P1.4 :</u>

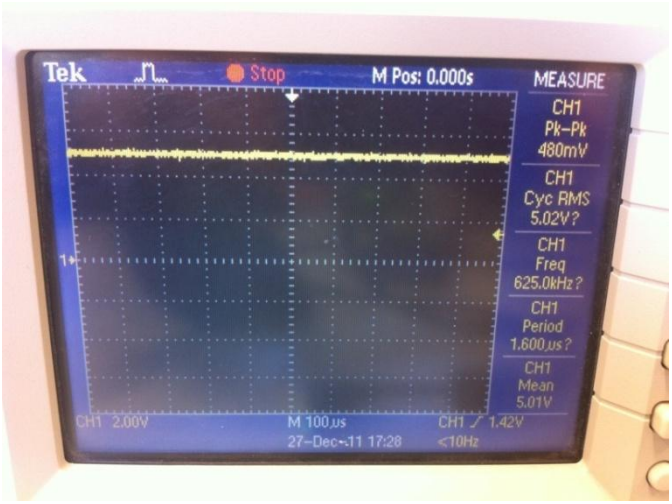


אנו רואים שהמתח הממוצע Vmean הוא: 5.02V("1" לוגי = VCC)

התדר FREQ הוא : 208.3Khz תדר גבוה מאוד (מתח ישר) .

והמתח PK-PK הוא : 480mV (אין אמפליטודה (מזערי)

<u>פורט P1.6 :</u>



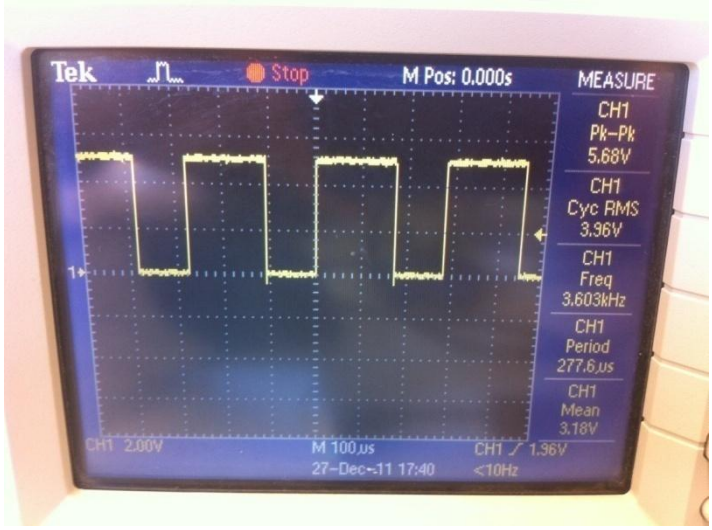
אנו רואים שהמתח הממוצע Vmean הוא: 5.01V("1" לוגי = VCC)

התדר FREQ הוא : 625Khz תדר גבוה מאוד (מתח ישר) .

והמתח PK-PK הוא : 480mV (אין אמפליטודה (מזערי))

לאחר מכן אנו חיברנו את הסקופ ביציאה של ה- 74LS244(חוצץ) לראות שהאות שהתקבל מהפורטים הועבר מהחוצץ ללא שינוי (אחרי החוצץ וליפני הדוחף למנוע) .

הדק 5 של TLE 5206-2 שמחובר דרך החוצץ לפורט P1.5 :

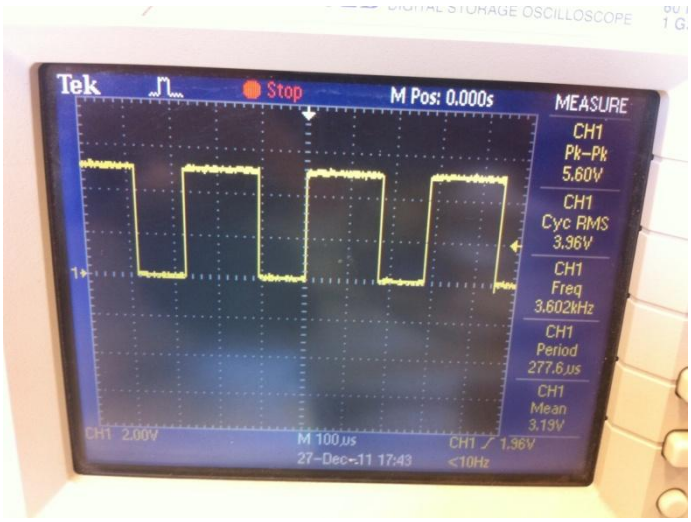


אנו רואים שהמתח הממוצע Vmean הוא: 3.18V

התדר FREQ הוא : 3.603Khz .

והמתח PK-PK הוא : V5.68

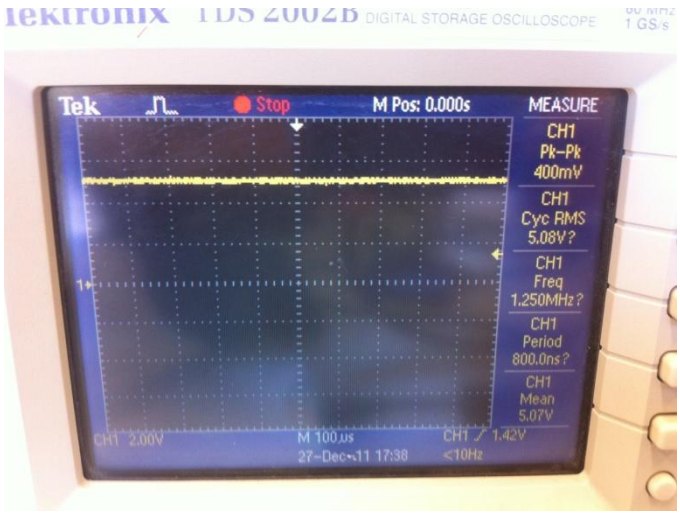
הדק 5 של TLE 5206-2 השני שמחובר דרך החוצץ לפורט P1.7 :



אנו רואים שהמתח הממוצע Vmean הוא: 3.19V

התדר FREQ הוא : 3.602Khz . והמתח PK-PK הוא : 5.60V

הדק 3 של TLE 5206-2 שמחובר דרך החוצץ לפורט P1.4 :

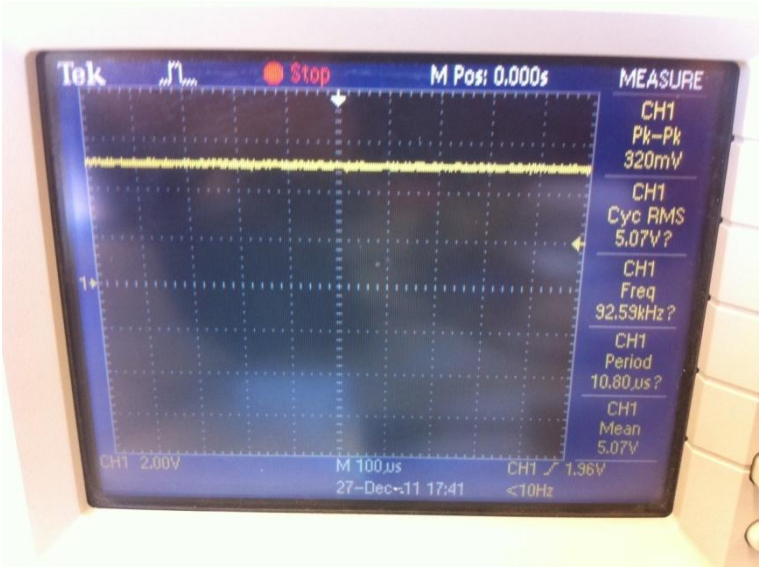


אנו רואים שהמתח הממוצע Vmean הוא: 5.07V("1" לוגי = VCC)

התדר FREQ הוא : 1.250Mhz תדר גבוה מאוד (מתח ישר) .

והמתח PK-PK הוא : 400mV (אין אמפליטודה (מזערי)) .

<u>הדק 3 של TLE 5206-2 השני שמחובר דרך החוצץ לפורט P1.6 :</u>



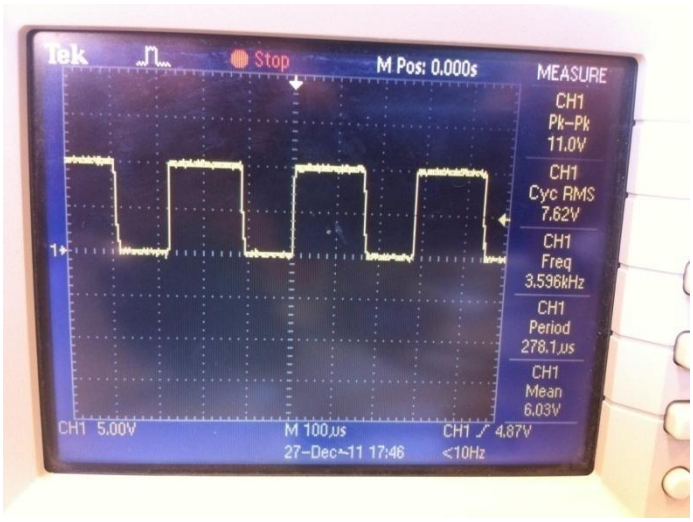
אנו רואים שהמתח הממוצע Vmean הוא: 5.07v("1" לוגי = VCC)

התדר FREQ הוא : 92.59Khzתדר גבוה מאוד (מתח ישר) .

והמתח PK-PK הוא : 320mV (אין אמפליטודה (מזערי)) .

לאחר מכן אנו חיברנו את הסקופ ביציאה של ה- TLE 5206-2 (דוחף למנועים) (אחרי הדוחף וליפני המנוע) לראות את השינוי שיש לאחר ה- TLE 5206-2 ולראות איזה גלים מגיע לנו למנועים (זאת אומרת מה המתח PK-PK לאחר הדוחף בהדקים Y2,Y4 ואת התדר מוצא) .

<u>הדק 7 של TLE 5206-2 שמחובר דרך החוצץ לפורט P1.5 :</u>

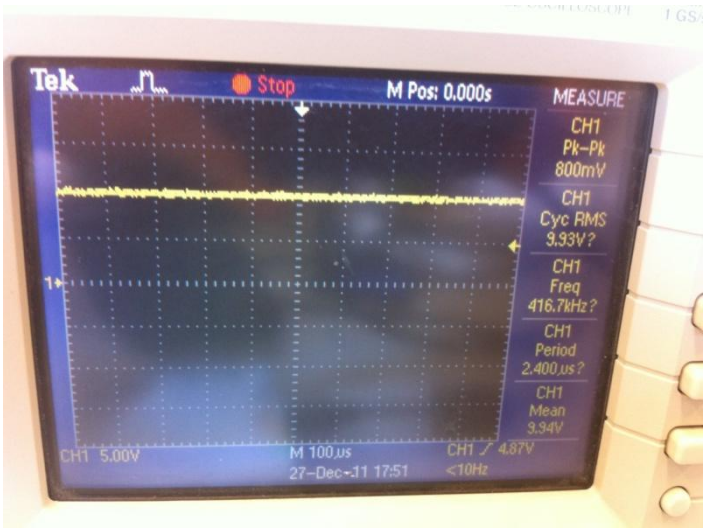


אנו רואים שהמתח הממוצע Vmean הוא: 6.03V

התדר FREQ הוא : 3.596Khz

והמתח PK-PK הוא : 11V

<u>הדק 7 של TLE 5206-2 שמחובר דרך החוצץ לפורט P1.7 :</u>



אנו רואים שהמתח הממוצע Vmean הוא: 9.93V

התדר FREQ הוא : 416.7Khz .והמתח PK-PK הוא : 800mV

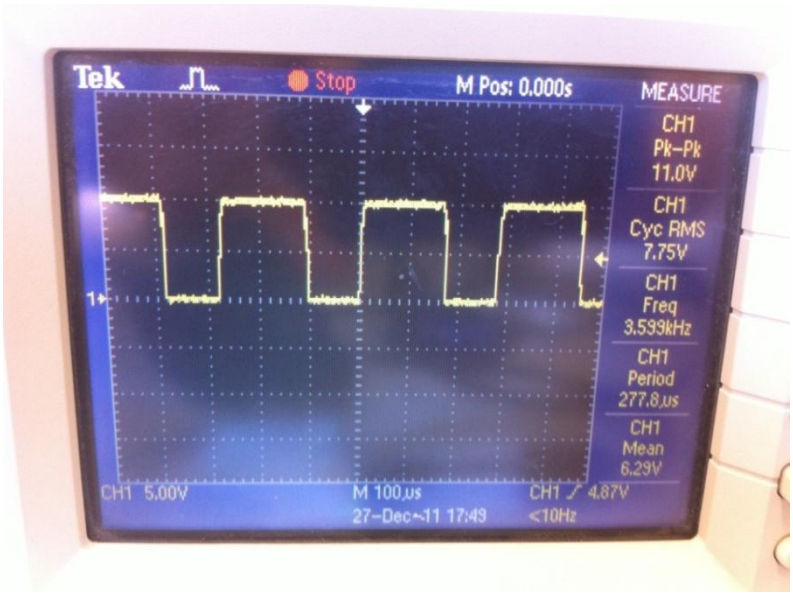
הדק 1 של TLE 5206-2 שמחובר דרך החוצץ לפורט P1.4 :

אנו רואים שהמתח הממוצע Vmean הוא: 9.82V("1" לוגי = VCC)

התדר FREQ הוא : 25Khz תדר גבוה מאוד (מתח ישר) .

והמתח PK-PK הוא : 200mV (אין אמפליטודה (מזערי)) .

הדק 1 של TLE 5206-2 שמחובר דרך החוצץ לפורט P1.6 :



אנו רואים שהמתח הממוצע Vmean הוא: 6.29V("1" לוגי = VCC)

התדר FREQ הוא : 3.599Khz תדר גבוה מאוד (מתח ישר) .

והמתח PK-PK הוא : 11V .

לאחר מכן אנו חיברנו את המנועים דרך החוצץ והדוחף למנוע לפורטים וראינו שהמנועים מסתובבים לכיוון הנכון (על פי התוכנית) .

לסיכום:

כמו כן ראינו את פורטים P1.4 ו P1.6 לפני הדוחף למנועים (שני מדידות ראשונות) המתח היה מתח ישר לכן PK-PK היה מזערי התדר היה גבוה מאוד וה-Vmean היה המתח הישר כלומר "1" לוגי.

כמובן שלאחר הדוחף למנועים המתח הישר עלה לכמעט 10V לעומת המתח הקודם שהיה

בין 3.5V-5V.

אנו יכולים לשים לב שהמתח ביציאה של הבקר לדוגמא בפורט 1.5 הוא 5.36 V ואילו המתח שאנו רואים ביציאות של הדרייבר הוא 11V

כמצופה הדרייברים הגבירו את המתח ואת הזרם המועבר למנוע (על פי ניסוי מנועי DC אפשר לראות שהמנוע צורך זרם יותר מאשר יכול לספק המיקרו בקר).

תאים חשמליים (סוללות) :

<u>רקע תיאורטי:</u>

בעולם חשמלי אנו חייבים זמינות לחשמל כל הזמן, הסוללה החשמלית הראשונה הייתה תא שבתוכו היו ערוכים לסירוגין, בזה אחר זה, לוחות עשויים אבץ ונחושת.

התא היה בנוי בצורה כזו שלאחר לוח מסוג אחד היה הלוח של הסוג השני.

לוחות אלה הם האלקטרודות (האנודה והקתודה) שהם מופרדות ע"י חומר שהוא רווי באלקטרוליט, שהוא כל חומר המכיל יונים חופשיים, שדרכם עוברים האלקטרונים מהאנודה לקתודה וכך נוצר חשמל. כל תא כזה ובו לוח אבץ, לוח נחושת וביניהם אלקטרוליט הוא תא חשמלי העומד בפני עצמו, וניתן לחברו לתא הבא אחריו בטור, ליצירת מתח חשמלי גבוה יותר.

ההמצאה הזו עזרה רבות לחקור את החשמל ותכונותיו, אך לאחר שימוש ממושך בסוללה הייתה מצטברת על האלקטרודות שכבה של גז המבודדת אותן מבחינה חשמלית.

לאחר כמה שנים, הומצאה הסוללה היבשה שבאה לפתור את הבעיה.

בסוללה זו היו שלושה מרכיבים עיקריים:

1. קתודה - (קוטב חיובי) שבה מתרחש תהליך חיזור, שבו נקלטים אלקטרונים מהמעגל החיצוני (צרכן החשמל).
2. אנודה - (קוטב שלילי) שבה מתרחש תהליך חמצון, שבו נמסרים אלקטרונים למעגל החיצוני.
3. אלקטרוליט - תווך המפריד בין האנודה לקתודה ומאפשר את יצירת הפרש הפוטנציאלים ביניהן. הוא מכיל בסיס או חומצה או סוג של מלח.

המגע בין האלקטרוליט לבין 2 הלוחות גורם לתגובה כימית המתבטאת בנדידת יונים דרך התמיסה, ואם מעבירים בין הלוחות חוט מוליך אז דרך אותו חוט יזרום זרם חשמלי.

בפרויקט שלנו השתמשנו בסוללת מסוג ניקל מטל נטענות נטענת.

סוללת ניקל מטל NIMH:

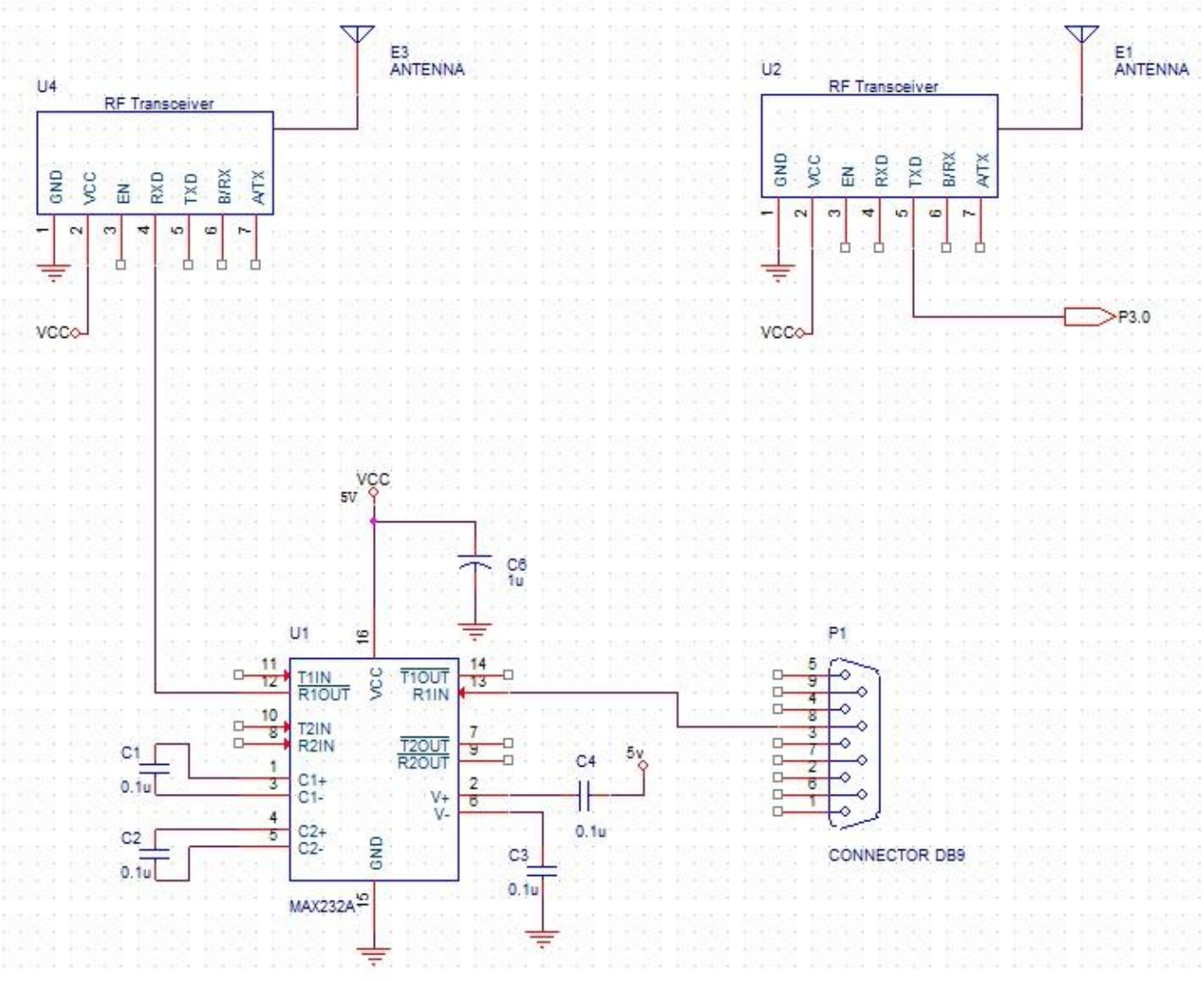
בטריות ניקל מטל הם מעין סוללות נטענות כי היא מאוד דומה ל-ניקל קדמיום (NiCd) הסוללה, אבל בניגוד סוללות NiCd, בסוללות NiMH יש מימן כולך סגסוגת כמו אנודת במקום באמצעות קדמיום . NiMHקיצור היא הרווחת עבור ניקל מתכת הידריד הסוללות. בתא זה, ניקל משחק את תפקיד של קתודית, דומים לאלו של תאים .NiCd ניקל מטאל הידריד (NiMH) סוללות יכול להחזיק שתיים או שלוש פעמים את היכולת מגודל מקבילה הסוללה NiCd ועל ההשפעה זיכרון אינו כה משמעותי. אבל, אם לעומת ליתיום, סוללת את צפיפות האנרגיה של volumetric ניקל מטאל הידריד (NiMH) סוללות נמוכה ואת יכולת פריקה עצמית גבוהה.

צפיפות אנרגיה ספציפית של חומר NiMH כבר כמעט 70 · ש W לק"ג / 250 kJ) ק"ג), יחד עם צפיפות אנרגיה volumetric של שעה על · 300 W / ל The penlight משותפת בגודל (א"א) תאים בעלי יכולות נומינלי C כי טווחי from1100 mA ש · כדי ש · mA ב 2700 ו 1.2 בדרך כלל הם דורג בשיעור 0.2 .C × למרות יכולת הפריקה נחשב תפקיד ההפוכה של קצב הפריקה, אך עד שער 1 C × כמעט אין הבדל משמעותי כזה.

לפי ההיסטוריה של ניקל מטאל הידריד (NiMH) סוללות הטכנולוגיה NiMH הומצא ע"י סטנפורד Ovshinskyבאמצע 1980 sציון הראשון הסוללות הצרכן NiMH החלו להופיע.

בשנת 1994 Ovshinsky, של חברה ECD Ovonics, נכנסה לשותפות עם קשרים כללי מוטורס כדי לפתח ביצועים גבוהים ניקל מטאל הידריד (NiMH) סוללות עבור המכונית החשמלית של GM Ev1. כיום, הטכנולוגיה NiMH כבר ברישיון - Cobasys מיזם משותף בין Ovonics ECD ו Chevron Corporation.הייישומים סוללות NiMH כולל רכב היברידיים כמו תובנה הונדה / טויוטה פריוס Civicאו, בתוספת ומוצרי צריכה אלקטרוניי

שרטוט חשמלי מעגל המשדר:



הסבר שרטוט חשמלי מעגל המשדר:

המחשב מוציא נתונים דרך הפרוטוקול rs232

קונקטור rs232 פרוטוקול תקשורת טורית שמאפשרת למיקרוקונטרולר להעביר מידע ולקבל מידע ממקורות חיצוניים ולמקורות חיצוניים, המידע עובר ומתקבל מהמחשב לרובוט.

לצורך שידור למחשב יש להוסיף רכיב נוסף, שיאפשר לבצע שידור בפרוטוקול RS232 ויאפשר תיאום רמות מתח בין USART לבין ה-RS232.
האותות ב-USART משתמשים ברמות לוגיות של 0 עד 5 וולט. דבר זה מתאים למצב שבו רוצים לבצע תקשורת בין רכיבים ברמות לוגיות כאלה, אך במקרה של RS232 אנו נדרשים לרמות מתח שונות: מתחים נמוכים מ--5V לצורך ייצוג רמה לוגית "1", ומתחים גבוהים מ-5V לצורך ייצוג רמה לוגית "0". לפיכך, כדי להשתמש בפרוטוקול זה, עלינו לבצע תיאום של רמות מתח. הדבר זה מתאפשר על ידי מתאם רמות כגון **MAX232**. MAX232 הִנו רכיב פשוט, הפועל מהזנה יחידה של 5V

משדר מקלט:

המשדר מקלט מורכב משתי יחידות זהות שיכול לשמש גם משדר וגם מקלט.

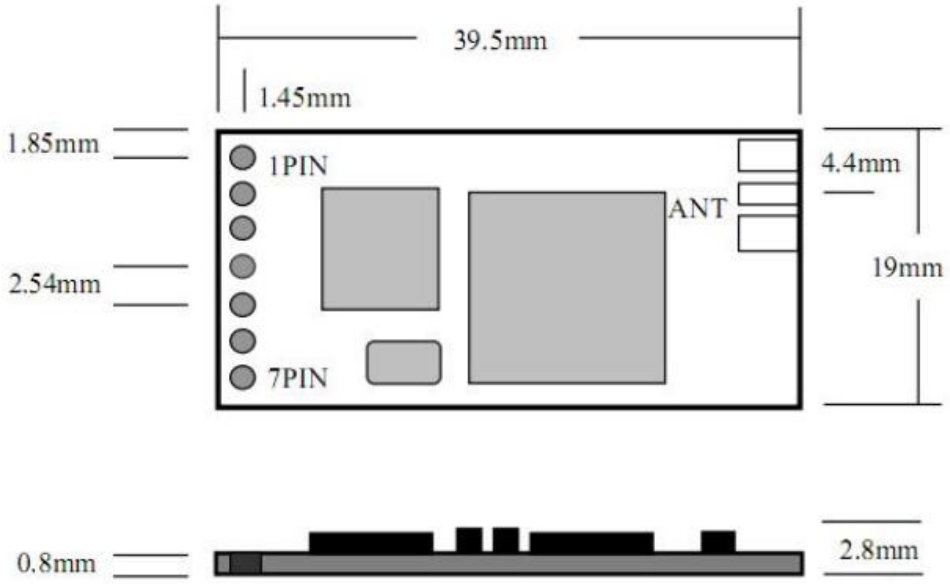
טווח התדר שידור נע בין 431 ל 478 Mhz .

טווח קליטה הוא בין 800 ל 1000 מטר 1200bps.

מתח הזנה בין 3.5 ל 5.5 וולט.

בעל יכולת לסינון רעש מובנה.

להלן מדגם נתונים:



Parameter	Value
Power Supply	3.3 – 5.5 (± 50mV ripples)
Frequency Range	431MHz to 478MHz (1KHz frequency step)
Frequency Interleaving	200KHz
Output Power	20mW (10 levels adjustable)
Receiving Sensitivity	-117dBm@1200bps
airborne speed	1200 – 9600bps
Speed In Port	1200 – 9600bps
Parity Check	8E1/8N1/801
Operating Humidity	10% - 90%
Operating Temperature	-20? - 70?
Transmission Mode Current	?35mA@10mW
Receiving Mode Current	?28mA
Sleep current	?5uA
Transmission Distance	800-1000m (visible range in open area)

Pinout:

Pins	Definition	Description
1	GND	0V
2	VCC	3.3V-5.5V
3	EN	POWER ENABLE(?1.6V) or SUSPENDED ENABLE(?0.5V Hibernation)
4	RXD	UART input, TTL
5	TXD	UART output, TTL
6	B/RX	RS485- or RS232 RX

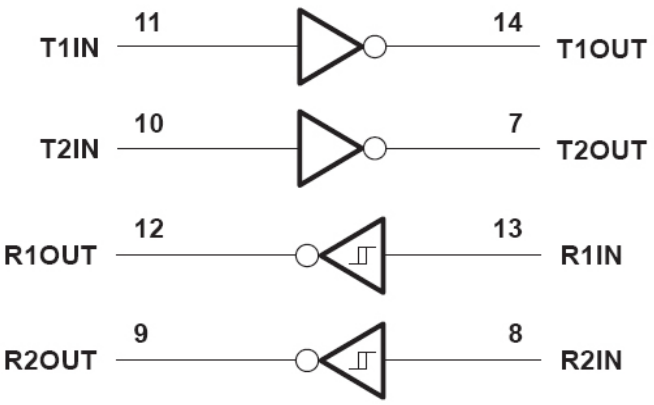
7	A/TX	RS485+ or RS232 TX

מתאם רמות- MAX232 Driver/ Receiver:

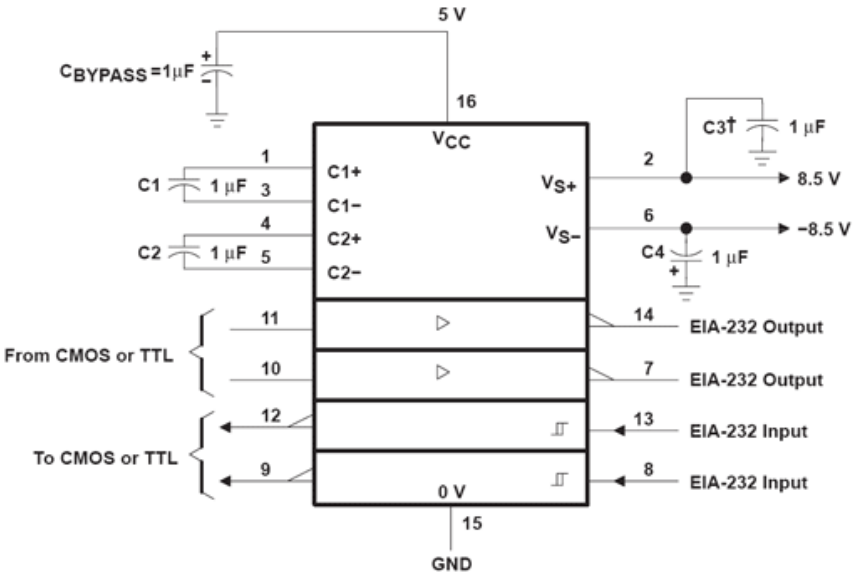
לצורך הוצאת מידע לשידור או לצורך קליטת מידע משתמשים ב-USART. רכיב זה מספיק בהחלט אם רוצים לשדר מ-PIC אחד ל-PIC אחר, אך הוא אינו מספיק לצורך שידור המידע מתוך ה-PIC אל המחשב. לפיכך, לצורך שידור למחשב יש להוסיף רכיב נוסף, שיאפשר לבצע שידור בפרוטוקול RS232 ויאפשר תיאום רמות מתח בין USART לבין ה-RS232.

האותות ב-USART משתמשים ברמות לוגיות של 0 עד 5 וולט. דבר זה מתאים למצב שבו רוצים לבצע תקשורת בין רכיבים ברמות לוגיות כאלה, אך במקרה של RS232 אנו נדרשים לרמות מתח שונות: מתחים נמוכים מ-5V- לצורך ייצוג רמה לוגית "1", ומתחים גבוהים מ-5V לצורך ייצוג רמה לוגית "0". לפיכך, כדי להשתמש בפרוטוקול זה, עלינו לבצע תיאום של רמות מתח. הדבר זה מתאפשר על ידי מתאם רמות כגון **MAX232**. MAX232 הִנו רכיב פשוט, הפועל מהזנה יחידה של 5V ובעל שני מתאמים באריזה אחת.

מבנה סכמתי:



צורת חיבור בסיסית :



הסבר:

- מוצא ה- USART (המידע המשודר אל המחשב) מתחבר לפין 10 או 11. רמות המידע מומרות לערכי מתחים חדשים, המותאמים ל-RS232 ומוצאות בפינים 7 או 14. מכאן המידע מתקדם לעבר המחשב.

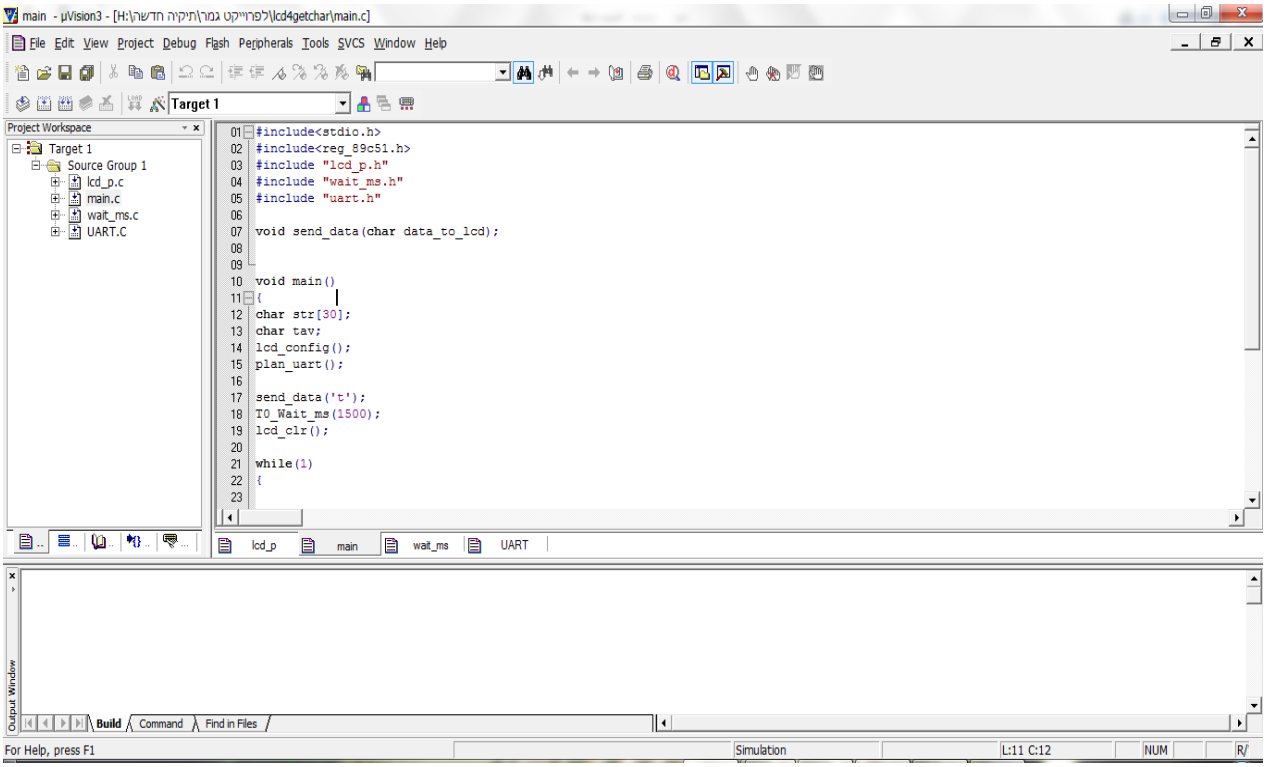
המידע המשודר מהמחשב מתחבר לפין 8 או 13 ברכיב. כאן מתקיים שוב תיאום רמות, אך זהו תיאום הפוך שיתאים ל-USART. האותות המתואמים מוצאים דרך פין 9 או 12 (כנזכר, יש שני מתאמים ברכיב כך שהפינים מופיעים בזוגות. מסיבה זו יש צורך לבחור באחד מהם ולהשתמש בפינים הרלוונטיים) ומתחברים אל ה-PIC . בפרויקט זה אנו לא משתמשים במידע חוזר מהמחשב.

מעגל ה - uart :

חלק זה למעשה החלק שקולט את האותות המקלט ומצע פעולות בהתאם, ז"א אם בקר הרובוט מקבל לדוג את התו A אזי הרובוט נע קדימה ולחליפין B אזי אחורה.

כתבנו את הקוד בתוכנת ה KELL UVISIONS:

קוד ה uart (main) תלוי גם בתת ספריות עזר, כפי שמוצג בתצלום.



MAIN:

```
#include<stdio.h>//sprint()  סיפריה שכוללת פקודות קלט/פלט ואחראית על
פקודות
#include<reg_89c51.h>
#include "lcd_p.h"//LCD  קובץ פקודות לתצוגת
#include "wait_ms.h"//  קובץ פקודות השהייה
#include "uart.h"

void send_data(char data_to_lcd);

void main()
{
char str[30]; // תווים 30 של מערך הגדרת
char tav;
lcd_config();//לעבוד לתצוגה שמאפשרת פונקציה
plan_uart();

send_data('t'); //לתצוגה  t האות שליחת
T0_Wait_ms(1500); // וחצי דקה של השהייה
lcd_clr();//  פונקציה שמנקה את התצוגה

while(1)
{

lcd_line(1,0);
tav=getchar();
sprintf(str,"%c ",tav);
lcd_string(str);//   לתצוגה לזרק str  ב שנימצא מה


}//end while(1)

}//end main
```

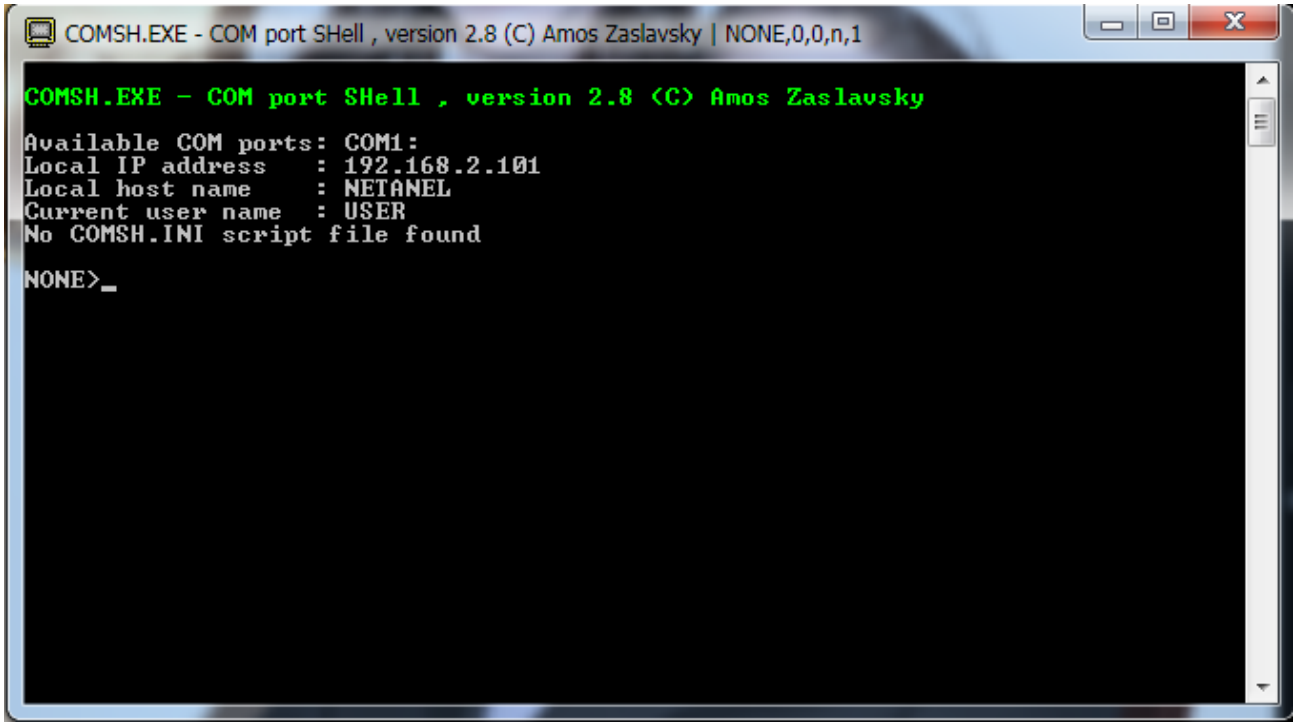
```
#include<reg_89c51.h>
//#include<at89c5131.h>
#include "uart.h"

 void plan_uart()       /* Function that plan the UART */
  {
  TR1=0;
  TH1=-(F_OSC/BAUD)/384;
  TMOD|=0x20;               /* planing the counter to autoreload   */
  SCON=0x52;                /* planing the UART to work with 8 bit */
  TR1=1;
  RI=0;
  }


 void plan_uart_timer2()       /* Function that plan the UART  for
timer2*/
  {
  TR2=0;
  RCAP2H=((-(F_OSC/BAUD)/32)&0XFF00)>>8;
  RCAP2L=(-(F_OSC/BAUD)/32)&0X00FF;
  T2CON=0X30;///////////////
  TR2=1;
  RI=0;
  }
```

תוכנת COMSH שליטה דרך PC:

יצירת תקשורת טורית לבדיקת המערכת באמצעות תוכנת COMSH

הפעלת COMSH אינה דורשת התקנה כל שהיא, והיא נעשית באמצעות ביצוע Double-Click על הקובץ COMSH.EXE שה - Icon שלו נראה כך :

הפעלת התכנית גורמת להופעת מסך טכסטואלי שד



ומה למסך שבאיור הבא :

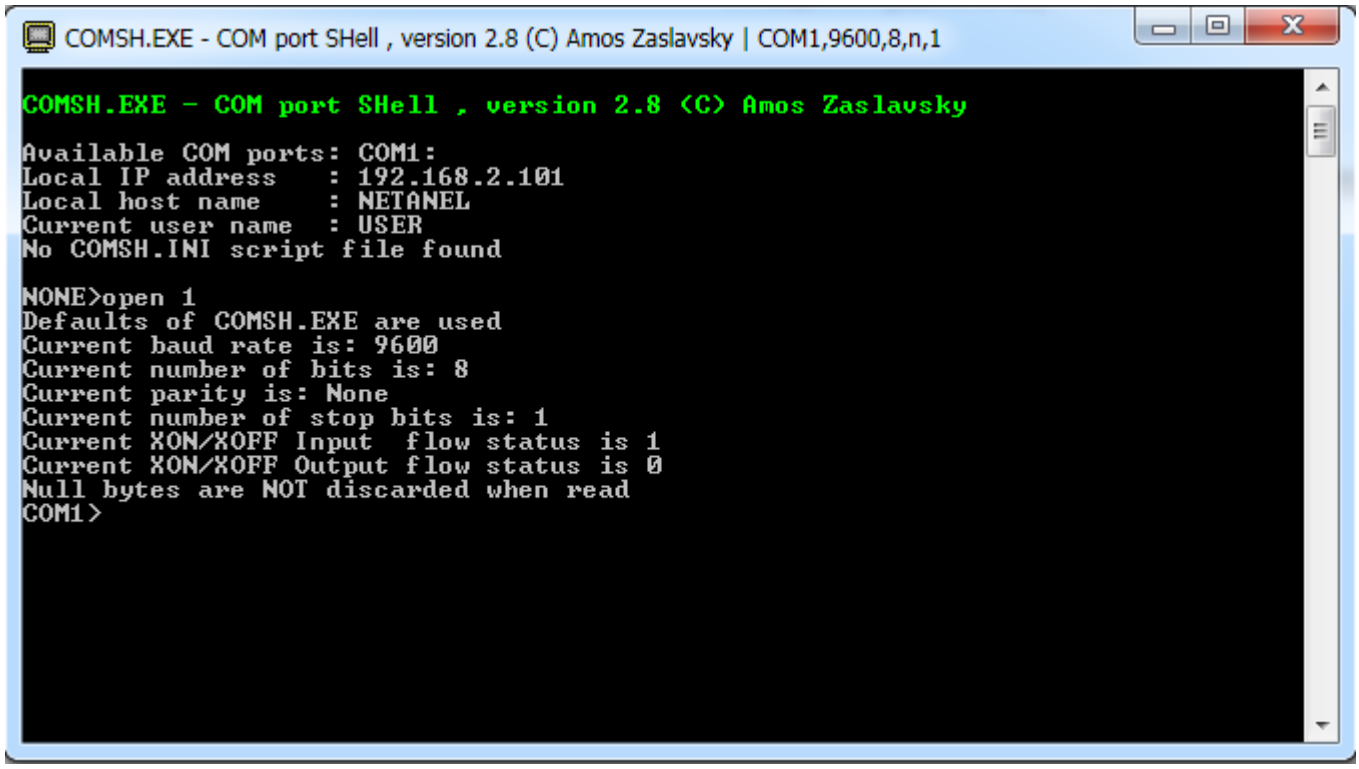
התכנית COMSH היא SHELL טכסטואלי נוח לשימוש עם התקני COM-ports (התקנים טוריים).

ה- Prornpt שנקרא NONE מראה שעדיין אין חיבור ל- port טורי כל שהוא.

כדי ליצור (לפתוח) חיבור ל- COM 1 הקש את הפקודה הבאה :

open1

שים לב שבעקבות פתיחת COM 1 לתקשורת, מתקבל גם דיווח על פרמטרי התקשורת למשל אצלינו:



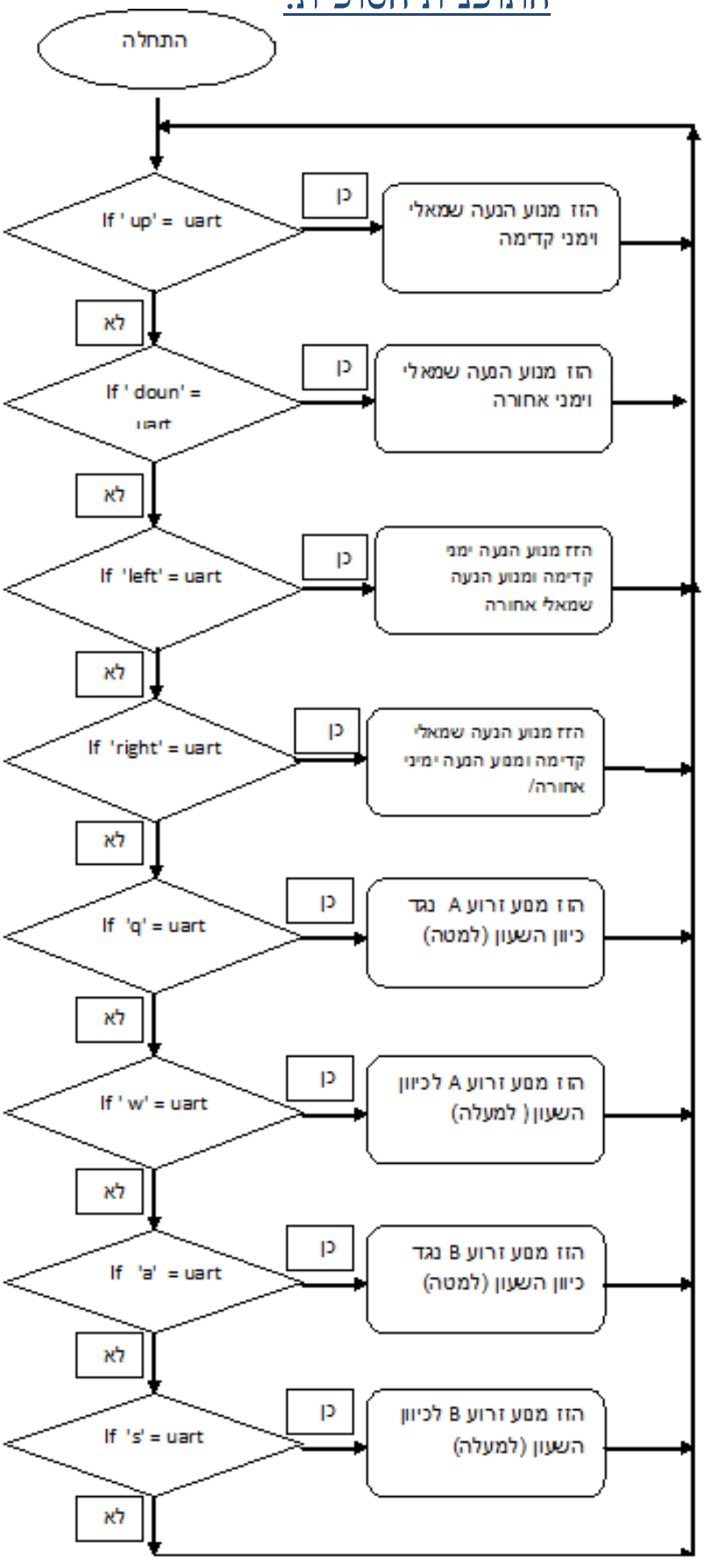
בשלב הבא נפעיל את המסוף (Terminal) הטכסטואלי באמצעות הפקודה: term

כפי שאנו רואים נקבל את ההודעה הבאה:

```
COM1>term
Terminal parameters set by TERMSET defaults
Local echo is turned OFF
Clearing the TX & RX Buffers

Dumb-Terminal is active
```

כל אות שנכתוב תופיע בתצוגת ה LCD

התוכנית הסופית:



```
#include<stdio.h>// sprint() ספריה שכוללת פקודות קלט/פלט ואחראית על פרודות
#include<reg_89c51.h>// צירוף פונקצית ספריה למיקרו בקר
#include "lcd_p.h"// LCD  קובץ פקודות לתצוגת
#include "wait_ms.h"// קובץ פקודות השהייה
#include "uart.h"// UART קובץ פקודות ל
#include "motor_pwm.h"// PWM קובץ פקודות של המנוע

void send_data(char data_to_lcd);


void main()
{
char str[30];
char tav;
lcd_config();
plan_uart();

send_data('t');
T0_Wait_ms(1500);
lcd_clr();

while(1)
{

lcd_line(1,0);
tav=getchar();
sprintf(str,"%c ",tav);
lcd_string(str);
}
 if (P3^0=='W')                    //בודק עם מקבל את האות W ב ASCII עם כן נוסע קדימה
   {
   lcd_config();//פונקציה שמאפשרת לתצוגה לעבוד
   lcd_clr();//פונקציה שמנקה את התצוגה
   lcd_string("   go street  ");// go street מציג על התצודה את המשפט
   motors(90,90);//motor forward//90 שני המנועים נוסעים קדימה במהירות
   T0_Wait_ms(1000);//השהייה של שניה
   motors(0,0);//stop motors
   }
               //END IF
if (P3^0=='S')             //בודק עם מקבל את האות S ב ASCII עם כן נוסע אחורה

    {
     lcd_config();//פונקציה שמאפשרת לתצוגה לעבוד
     lcd_clr();//פונקציה שמנקה את התצוגה
     lcd_string("   go back    ");// go back מציג על התצודה את המשפט
     motors(-90,-90);//-90 שני המנועים נוסעים אחורה במהירות
     T0_Wait_ms(1000);//השהייה של שניה
     motors(0,0);//stop motors
         }
              // END IF
if (P3^0=='A')              //בודק עם מקבל את האות A ב ASCII עם כן נוסע שמאלה
    {
     lcd_config();//פונקציה שמאפשרת לתצוגה לעבוד
     lcd_clr();//פונקציה שמנקה את התצוגה
     lcd_string("   go left   ");// go left מציג על התצודה את המשפט
     motors(-190,190);  //LEFT
     T0_Wait_ms(1000);// השהייה של שניה
     motors(0,0);//stop motors
         }
              // END IF

if (P3^0=='D')              //בודק עם מקבל את האות D ב ASCII עם כן נוסע ימינה
    {
     lcd_config();//פונקציה שמאפשרת לתצוגה לעבוד
     lcd_clr();//פונקציה שמנקה את התצוגה
```

```
     lcd_string("   go right  ");// go right  מציג על התצוגה את המשפט
     motors(190,-190);  //RIGHT
     T0_Wait_ms(1000);//השהייה של שניה
     motors(0,0);//STOP MORTORS
         }
              // END IF
if (P3^0=='I')              // בודק עם מקבל את האות I ב ASCII עם כן מזיז זרועה A קדימה
    {
     lcd_config();//פונקציה שמאפשרת לתצוגה לעבוד
     lcd_clr();//פונקציה שמנקה את התצוגה
     lcd_string(" GO ZROA A ");//GO ZROA A מציג על התצוגה את המשפט
     motors_zroa(50,0);//
     T0_Wait_ms(1000);//השהייה של שניה
     motors_zroa(0,0);// STOP MOTORS ZROA
         }
              // END IF
if (P3^0=='K')              // בודק עם מקבל את האות K ב ASCII עם כן מזיז זרועה A אחורה
    {
     lcd_config();//פונקציה שמאפשרת לתצוגה לעבוד
     lcd_clr();//פונקציה שמנקה את התצוגה
     lcd_string(" BACK ZROA A ");//BACK ZROA A מציג על התצוגה את המשפט
     motors_zroa(-50,0);//
     T0_Wait_ms(1000);//השהייה של שניה
     motors_zroa(0,0);// STOP MOTORS ZROA
         }
              // END IF
if (P3^0=='U')              // בודק עם מקבל את האות U ב ASCII עם כן מזיז זרועה B קדימה
    {
     lcd_config();//פונקציה שמאפשרת לתצוגה לעבוד
     lcd_clr();//פונקציה שמנקה את התצוגה
     lcd_string("  go zroa b  ");//GO ZROA B מציג על התצוגה את המשפט
     motors_zroa(0,50);//
     T0_Wait_ms(1000);// השהייה של שניה
     motors_zroa(0,0);// STOP MOTORS ZROA
         }
if (P3^0=='J')              // בודק עם מקבל את האות J ב ASCII עם כן מזיז זרועה B אחורה
    {
     lcd_config();//פונקציה שמאפשרת לתצוגה לעבוד
     lcd_clr();//פונקציה שמנקה את התצוגה
     lcd_string("  back zroa b  ");//BACK ZROA B מציג על התצוגה את המשפט
     motors_zroa(0,-50);//
     T0_Wait_ms(1000);//השהייה של שניה
     motors_zroa(0,0);// STOP MOTORS ZROA
         }
}//end main
```

קובץ מצורף מס 1- Lcd_p

```
#include<reg_89c51.h>//  צירוף פונקציית ספריה למיקרו בקר

#include<absacc.h>

#include "lcd_p.h"// lcd קובץ פקודות לתצוגת ה



//================================LCD=========================

void lcd_config() ;//  פונקציה שמהתחלת את התצוגה

void delay_lcd();//פונקציית השהייה

void lcd_clr();// פונקציה שמנקה את התצוגה
```

```
void lcd_line(char line_number,char p);
void lcd_display(char character);
void lcd_string(char str[]);// התצוגה על אותו ומציגה התווים מערך את שלוקחת פונקציה
send_command(unsigned char command);// פונקציה לשליחת מילת בקרה
send_data(char data_to_lcd);//  פונקציה לשליחת תו לתצוגה


void delay_lcd()  //small delay_lcd// פונקציית השהייה קטנה
{
        int timer=200;
        while(timer--);
}


void lcd_clr()// פונקציה שמנקה את התצוגה
{
        send_command(0x01);            //clear_display
        delay_lcd();
}


//=======================CONFIG THE LCD=====================


void lcd_config() //Initialization of The LCD, פונקציית אתחול של התצוגה
{
    RW=0;
    delay_lcd();
    send_command(0x2c);   //4bit,2 line,5*7 dot  4 bit רואים פה שהגדרנו את התצוגה ל
    delay_lcd();
    send_command(0x0e);    // display on ,cursor on,cursor blink
    delay_lcd();
    send_command(0x01);     //clear_display
    delay_lcd();
    send_command(0x06);     // increment cursor,no display shift
}
//=======================LCD LINE NUMBER====================]
void lcd_line(char line_number,char p)// פונקציה שמאפשרת להציג את המילה המקום
הרצוי(עמודה,שורה
```

```
{
        switch(line_number)
        {
        case 1:
                delay_lcd();
                send_command(0x80+p);
                break;

        case 2:
                delay_lcd();
                send_command(0xc0+p);
                break;

        case 3:
                delay_lcd();
                send_command(0x94+p);
                break;

        case 4:
                delay_lcd();
                send_command(0xd4+p);
                break;
        }

}

//===============DISPLAY THE CHARACTER ON THE LCD===========
void lcd_display(char character)
{
        delay_lcd();
           send_data(character);     //Send The character to the LCD
}
```

```
void lcd_string(char str[])//LCD ומציגה אותו על ה char  פונקציה שלוקחת מערך מטיפוס
{
        int i=0;
        while(str[i])
           {
                delay_lcd();

                if (str[i]>=0xa0)                              //if hebrew

                        send_data(str[i++]-0x40);
         else

                        send_data(str[i++]);
        }
}



send_command(unsigned char command)
{
    RW=0;
    RS=0;
    PORT_LCD &=0x87;
    PORT_LCD |=((command &0xf0 )>>1);
    E=0;
    delay_lcd();
    E=1;
    delay_lcd();
    E=0;

    PORT_LCD &=0x87;
    PORT_LCD |=((command &0x0f )<<3) ;
    RS=0;
    E=0;
    delay_lcd();
```

```
    E=1;
    delay_lcd();
    E=0;

}

send_data(char data_to_lcd)
{
    RW=0;
    RS=1;
    PORT_LCD &=0x87;
    PORT_LCD |= ((data_to_lcd &0xf0 )>>1);
    E=0;
    delay_lcd();
    E=1;
    delay_lcd();
    E=0;

    RS=1;
    PORT_LCD &=0x87;
    PORT_LCD |= ((data_to_lcd &0x0f )<<3) ;
    E=0;
    delay_lcd();
    E=1;
     delay_lcd();
     E=0;
}
```

קובץ מצורף מס 2- Wait_ms

```
#include "wait_ms.h"

void T0_Wait_ms (unsigned int ms)
{
      TMOD &= ~0x0f;                        // 16-bit free run mode
```

```
	TMOD |=  0x01;


	while (ms)
	{
		TR0 = 0;                              // Stop Timer0
		TL0=(-(F_OSC/12000))&0X00FF;
		TH0=((-(F_OSC/12000))&0XFF00)>>8;   // Overflow in 1ms

		TF0 = 0;                              // Clear overflow
indicator
		TR0 = 1;                              // Start Timer0
		while (!TF0);                         // Wait for overflow
		ms--;                                 // Update ms counter
	}

	TR0 = 0;                                // Stop Timer0
                                    }
```

קובץ מצורף מס 3- Motor_pwm

```
#include "motor_pwm.h"


/*
P1.3  CCAP0
P1.4  CCAP1
P1.5  CCAP2
P1.6  CCAP3
P1.7  CCAP4
*/


motors(int left,int right)//  יצירת מבנה הפקודה
{
motor_left(left);//P1.0,P1.5
motor_right(right);//P1.1,P1.7
}

motor_left(int pwm) //P1.0,P1.5
{
	CCAPM2=0X42; //enable pwm 2
	CCON=0X40;  // PCA counter run
	if(pwm>=0)//  בודק אם אחורה או קדימה
	{
		PHASE_MOTOR_LEFT=1;//P1.0- קדימה
		CCAP2H=pwm;  //P1.5- PWM כ 1.5 מגדיר את פורט
	}
	else
	{
		PHASE_MOTOR_LEFT=0; //P1.0- אחורה
		CCAP2H=pwm; //P1.5- PWMכ 1.5 מגדיר את פורט
	}
}


motor_right(int pwm)
{
	CCAPM4=0X42; //enable pwm 4
	CCON=0X40;   // PCA counter run
	if(pwm>=0) // בודק אם אחורה או קדימה
```

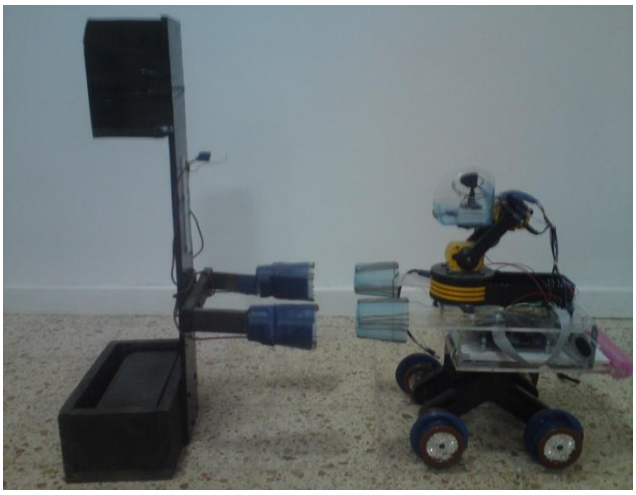
```
	{
		PHASE_MOTOR_RIGHT=1; //P1.1-קדימה
		CCAP4H=pwm; //P1.7 - PWMכ 1.7 מגדיר את פורט
	}
	else
	{
		PHASE_MOTOR_RIGHT=0; //P1.1-אחורה
		CCAP4H=pwm; //P1.7 - PWMכ 1.7 מגדיר את פורט
	}
}



 /*---------------motors zroa-------------------------------



motors_zroa(int zroa_a,int zroa_b)// יצירת מבנה הפקודה
{
motor_zroa_a(zroa_a);// p1.4,p1.2
motor_zroa_b(zroa_b);// p1.3,p1.6
}

motor_zroa_a(int pwm) // p1.4,p1.2
{
	CCAPM1=0X42; //enable pwm 1
	CCON=0X40;  // PCA counter run
	if(pwm>=0) // בודק אם אחורה או קדימה
	{
		ZROA_A=1;//P1.2 -קדימה
		CCAP1H=pwm;  //P1.4- PWMכ 1.4 מגדיר את פורט
	}
	else
	{
		ZROA_A=0; //P1.2 -אחורה
		CCAP1H=pwm; //P1.4- PWMכ 1.4 מגדיר את פורט
	}
}


motor_zroa_b(int pwm) // p1.3,p1.6
{
	CCAPM3=0X42; //enable pwm 3
	CCON=0X40;   // PCA counter run
	if(pwm>=0) // בודק אם אחורה או קדימה
	{
		ZROA_B=1; //P1.3 - קדימה
		CCAP3H=pwm; //P1.6
	}
	else
	{
		ZROA_B=0; //P1.3 - אחורה
		CCAP3H=pwm; //P1.6

	}
}
```

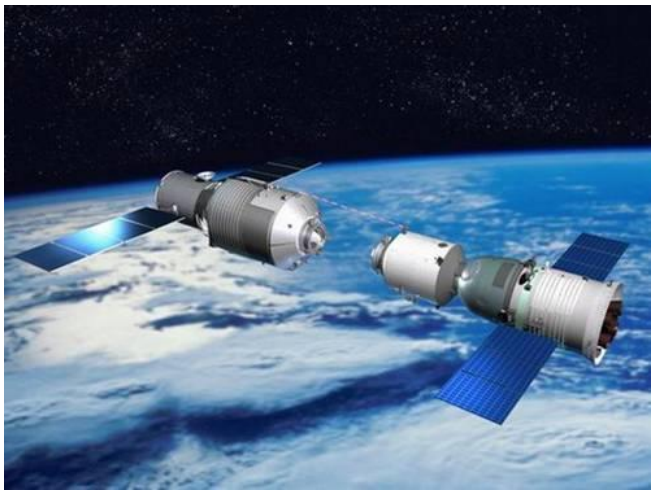
קובץ מצורף מס 4- uart

```
#include<reg_89c51.h>
//#include<at89c5131.h>
#include "uart.h"

 void plan_uart()       /* Function that plan the UART */
  {
  TR1=0;
  TH1=-(F_OSC/BAUD)/384;
  TMOD|=0x20;               /* planing the counter to autoreload   */
  SCON=0x52;                /* planing the UART to work with 8 bit */
  TR1=1;
  RI=0;
  }


 void plan_uart_timer2()       /* Function that plan the UART  for
timer2*/
  {
  TR2=0;
  RCAP2H=((-(F_OSC/BAUD)/32)&0XFF00)>>8;
  RCAP2L=(-(F_OSC/BAUD)/32)&0X00FF;
  T2CON=0X30;///////////////
  TR2=1;
  RI=0;
  }
```

עמדת הטעינה :

בראשית הפרויקט התעקשנו אנו רוצים שיהיה ניתן לטעון את הרובוט מרחוק... לאחר סיעור מוחות ..מצאנו פתרון בהשראת האופן שמתדלקים מטוסים ומעבורות חלל באוויר.





מה שעשינו זה בעצם לקחנו 4 קונוסים (כוסות פלסטיק) וכרכנו סביבם חוט נחושת..צורת קונוס דומה לחץ שמוצא את מקומו למטרה . לא צריך להתאמץ יותר מידיי בכדיי להכיוון את הרובוט לעמדת הטעינה מכיוון שברגע שהקונוס מספיק קרוב הוא נכנס פנימה לקונוס השני ונוצר סגירת מעגל לטעינה. על הרובוט מותקנים הקונוסים בצורת זכר ועל העמדת טעינה מותקנים קונוסים בצורת נקבה.

עמדת הטעינה היא מבנה נייח שבו מתוקן המטען שכפי שמופיע בתצלום מותקנים בוא הקונוסים בצורת הנקבה בגובה המתאים וברוחב המתאים בכדי שיוצר חיבור מוצלח לסגירת מעגל עם המטען.

תקלות ובעיות :

החלפת סוללות-

בתחילת העבודה בחרנו להשתמש בסוללות ליתיום לוחות של 12V/ 4800Ah .

לא היה לנו נוח להשתמש בסוללות אלו מכיוון שלאחר כמה שימושים, הסוללה התקלקלה והיא לא סיפקה את המתח המתאים.

לאחר מחשבה נוספת החלטנו כי יש צרך להחליף לסוללה אחרת שיהיה לנו יותר נוח לעבוד איתה

ובחרנו לעבוד עם סוללות ניקל מטל נטענות שזרם היציאה שלהם גדול בהרבה וגם יציבות עם מתחים וזרמים גבוהים.

החיסרון הוא משקל הסוללה שגדול יחסית למשקל סוללת הליתיום.

בעיית הדרייברים-

בעיה נוספת שנאלצנו להתמודד איתה הינה הדרייברים.

לאחר שחיברנו את הדרייברים- L293D למנועים ולבקר, גילינו, שהמערכת לא עובדת כמו שצריך.

לאחר תחקור ובדיקה מהי מהות הבעיה, ראינו כי הדרייברים לא יכולים לספק למנועים מעל 600mA .

וכמו כן כל מנוע דורש בעומס מלא 2.5A.

בעקבות כך החלטנו להחליף את הדרייברים לדרייברים מסוג TLE5206.

בעיות המשדר מקלט:

בתחילה שתמשנו במשדר מקלט בסיסי מסוג מקלט rr3 ומשדר rt4 אך אחוזי הרעש היו גדולים מכיוון שהמעגל בחלקו עם חיבורי וויראפ. פתרנו בעיה זו בעזרת מכמש איכותי ארוך טווח בעל יכולת סינון רעש.

סיכום ומסכנות:

- לאורך העבודה למדנו רבות על כל רכיב ורכיב בפרויקט ועל האפשריות עבודה שנמצאות בכל אחד ואחד מהם.
- למדנו להשתמש,למדוד ולשלב במערכת מיקרו בקר, למדנו על תכנותיו ושימושיו השונים.
- לאורך כל הפרויקט נאלצנו לקרוא דפי נתונים, דבר זה הקנה לנו מלבד הידע את הכלים לקריאת דפי נתונים בצורה מקיפה ולמידתם בדרך הטובה שיש.
- למדנו כי חשוב לעבוד על פי לוח זמנים, מכיוון שפרויקט מסוג זה כולל בתוכו מספר מרכיבים ויש חשיבות לסדר עבודה נכון בגלל התלות בין החלקים השונים.
  כמו כן סדר עבודה נכון מונע תקלות ובעיות רבות שיכלו לצוץ ולעלות במהלך הפרויקט.
  למרות זאת, חשוב לציין כי התכנון ולוח הזמנים חייבים להיות דינאמיים וגמישים לשינויים הנובעים מאילוצים שונים כגון תקלות ובעיות שונות שלא מחושבות מלכתחילה.
- אחת המסקנות החשובות שאליהם הגענו הוא שישנו הבדל ניכר בין מה שמצופה בתחילת הפרויקט לבין מה שקורה בפועל.
  ישנם מספר דברים שיכולים להשתנות וכתוצאה מכן להשפיע על התוצר הסופי.
- החקר הרב שעשינו בפרויקט נתן לנו כלים לעתיד, למדנו איך לנתח כל רכיב ורכיב ולקשר אותו לצורך שלנו.
- הצורך שעלה במהלך הפרויקט לפתור בעיות, גרם לנו להבין איך מגיעים לפתרונות דווקא בתחום האלקטרוניקה בצורה יעילה ומעמיקה .ע"י תחקור הבעיה בצורה מקיפה וחשיבה יצירתית מצאנו פתרונות לבעיות השונות
- מלבד הלמידה על הפרויקט עצמו, למדנו בצורה מעמיקה על כל רכיב ורכיב בפרויקט- מה שהקנה לנו ידע רב ובסיס להמשך דרכנו בתחום האלקטרוניקה.

ביבליוגרפיה:

מקרו בקר :

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

http://www.mikroe.com/eng/chapters/view/65/chapter-2-8051-microcontroller-/architecture

דוחף למנוע :

http://www.datasheetcatalog.org/datasheet/infineon/1-tle5206.pdf

קיט המרכב :

http://www.sparkfun.com/datasheets/Robotics/Rover%205%20Introduction.pdf

http://letsmakerobots.com/node/24031

כרטיס ce2:

www.see-sys.com

חוצץ 74F244:

http://www.datasheetcatalog.com/datasheets_pdf/7/4/F/2/74F244.shtml

rs232

http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html

מעגל הגביש:

http://robots.eitan.ac.il/index.php?page=MechPElecMicro

pwm:

http://mop.ort.org.il/es/scripts/inner.asp?pc=802870619

מנוע D.C :

http://www.rczone.co.il/modules.php?name=Forums&file=viewtopic&p=559108

http://robots.eitan.ac.il/index.php?page=MotorsDC

מייצב מתח :

http://ecee.colorado.edu/~mcclurel/mc7800.pdf

מעגל RESET–

www.arikporat.com/projects/**reset**_circuit.doc

מעגל ה-LCD

ספרו של שי מלול : ספר לימוד מיקרו בקרים ממשפחת 8051 לפרויקטים בשפת אסמבלי.

הוצאת שורש .

מעגל ה- USB .

http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

השראה ורעיונות.

/http://www.jbprojects.net/projects/wifirobot

נספחים :

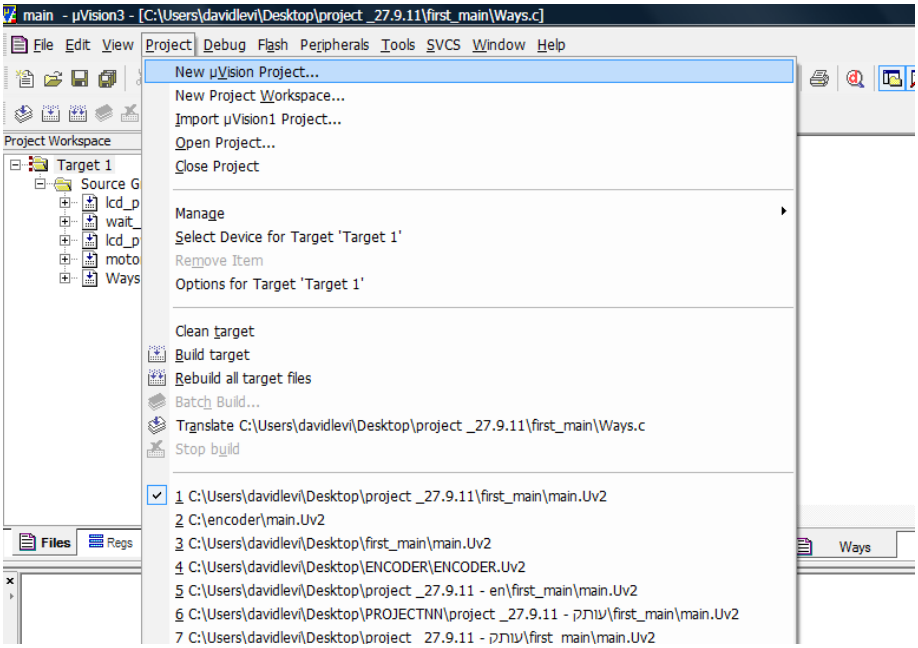
אופן פעולה לצריבה בבקר:

על מנת ליצור פרויקט חדש עם פונקציות שקיימות אני עובד על פי השלבים הבאים :

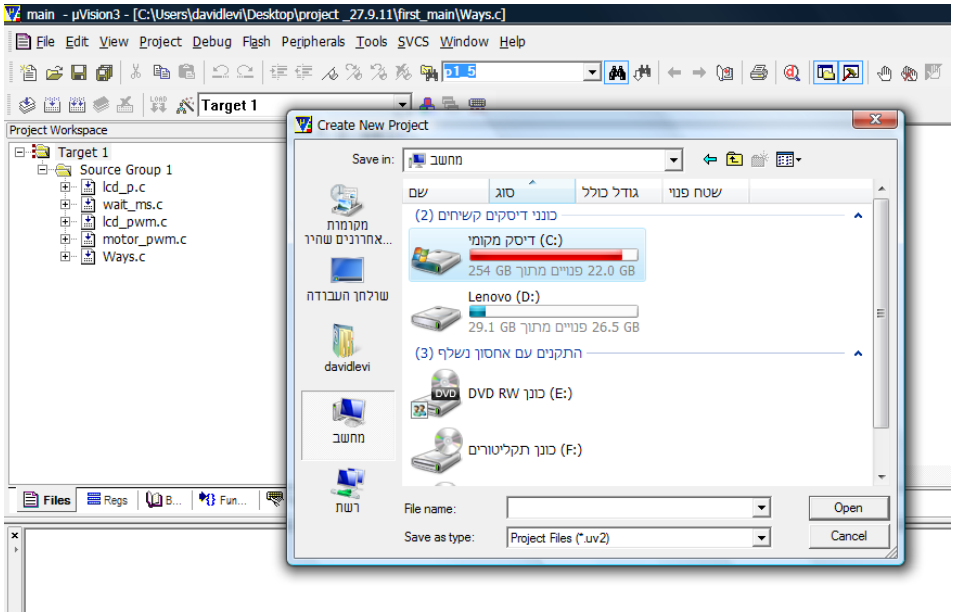
פותח את µVISION3

PROJECT

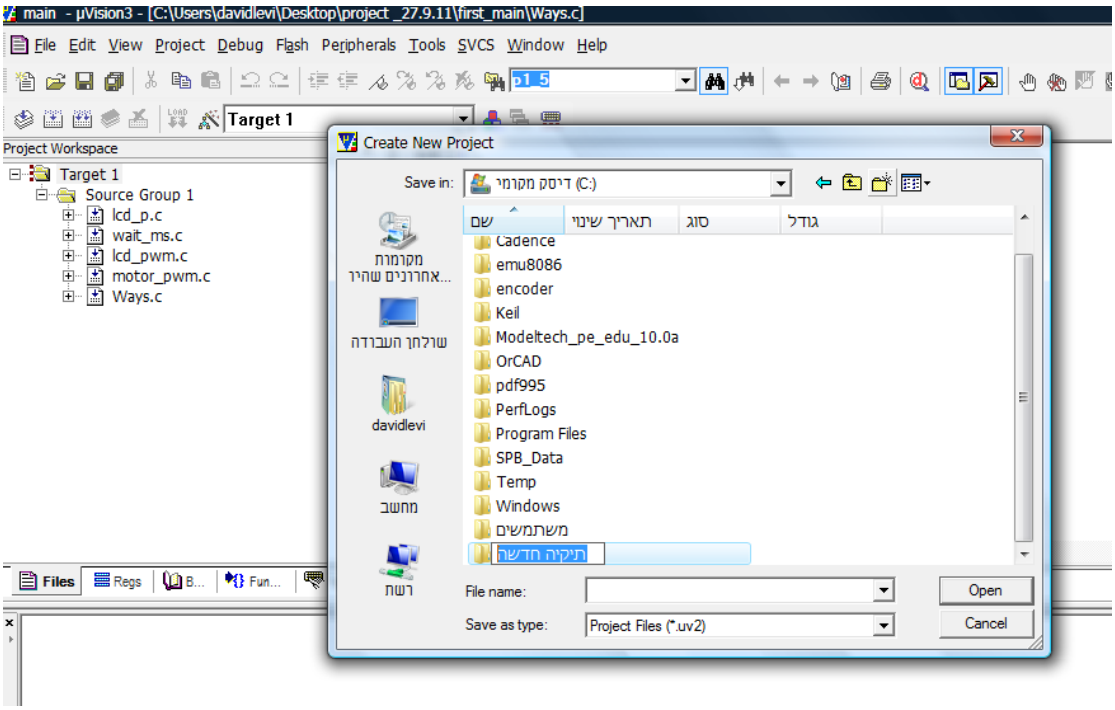
NEW µVISION PROJECT

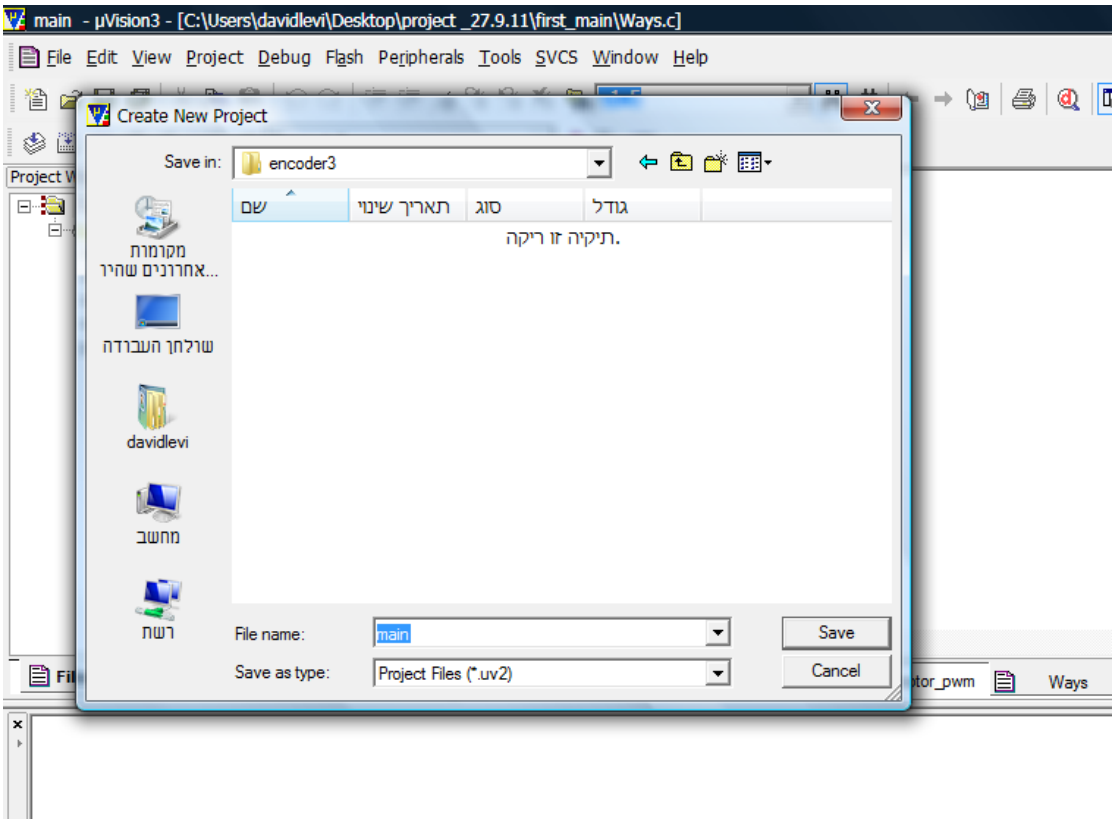


לאחר מכן אני בוחר איפה אני רוצה למקם את הפרויקט (עדיפות ב- C)

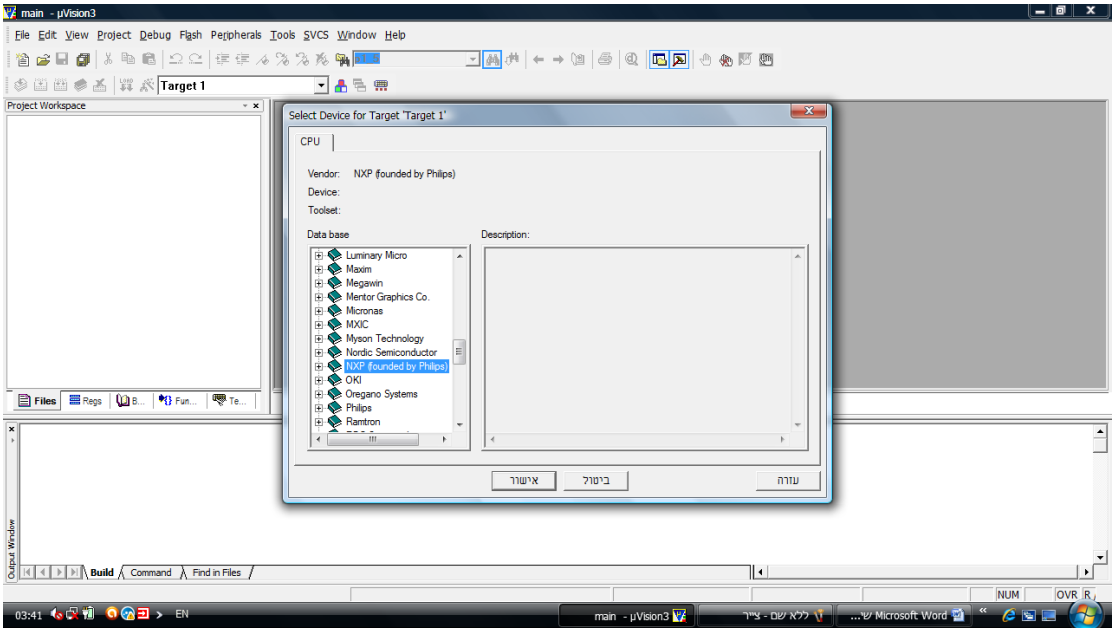


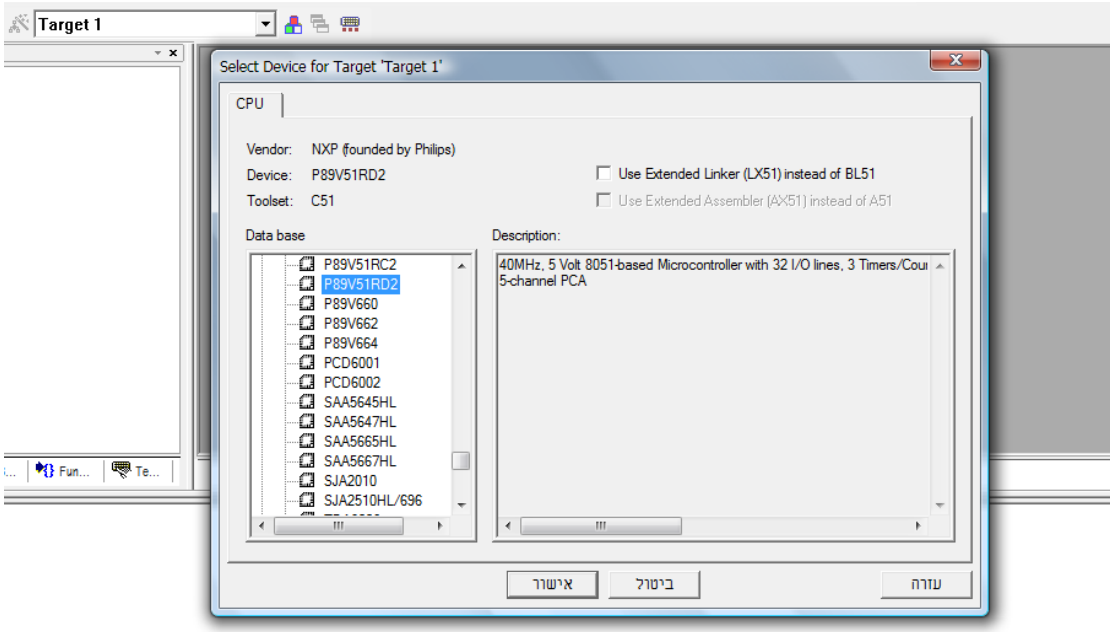
פותחים תיקיה חדשה ובוחרים שם .



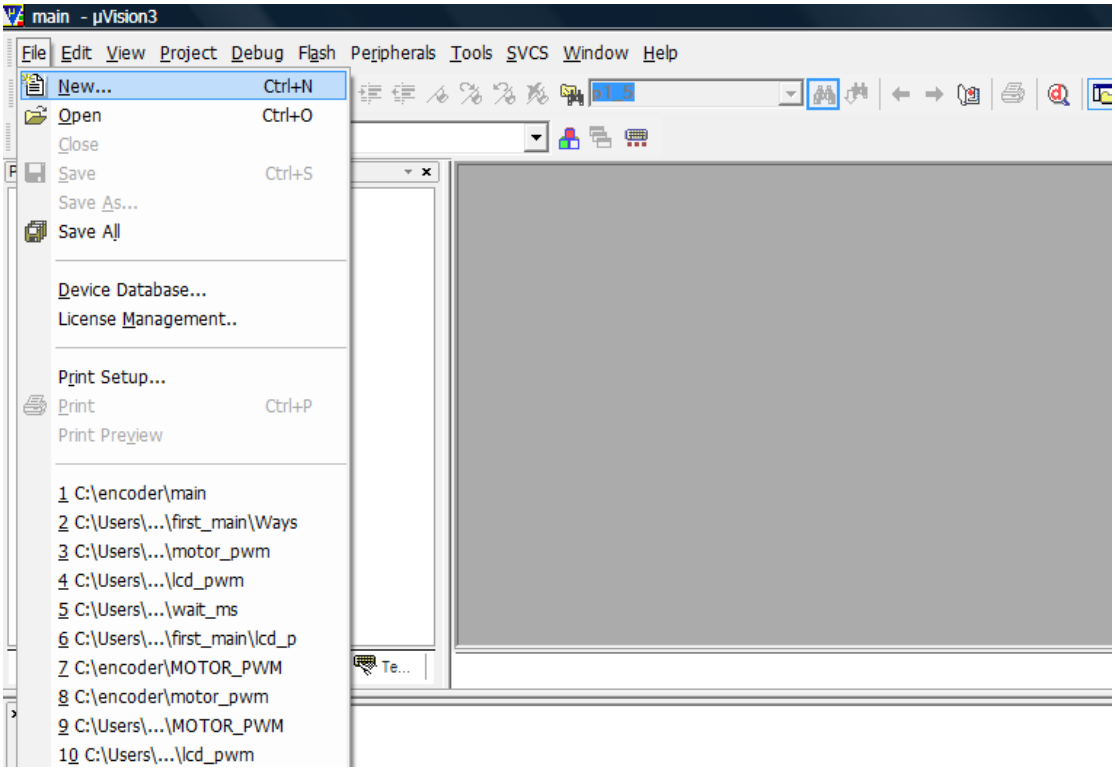


לאחר מכן בוחרים את הבקר

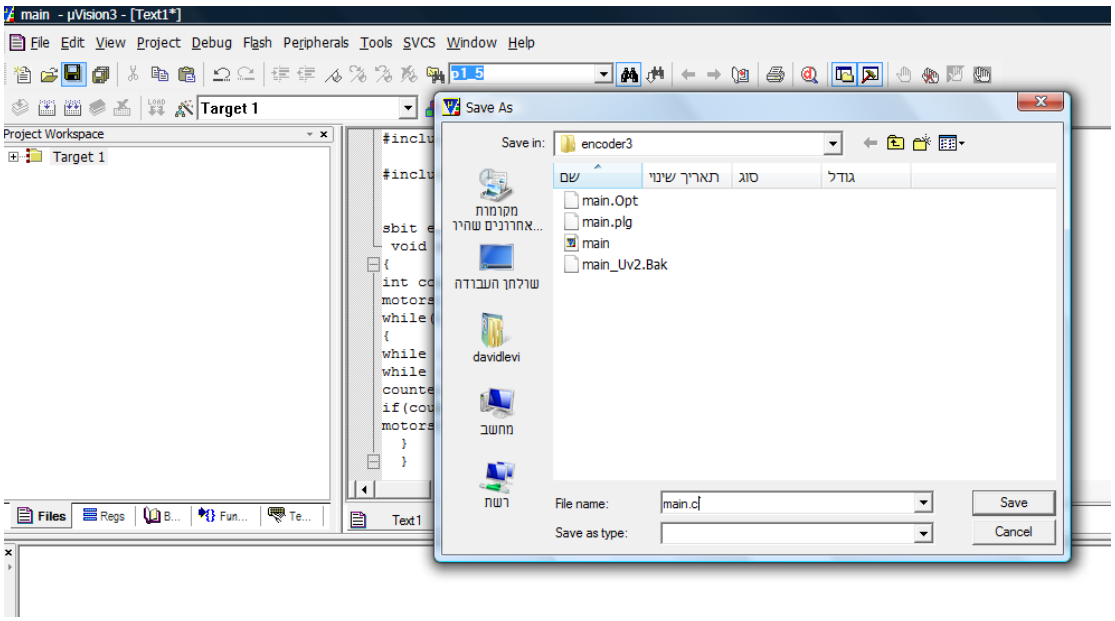




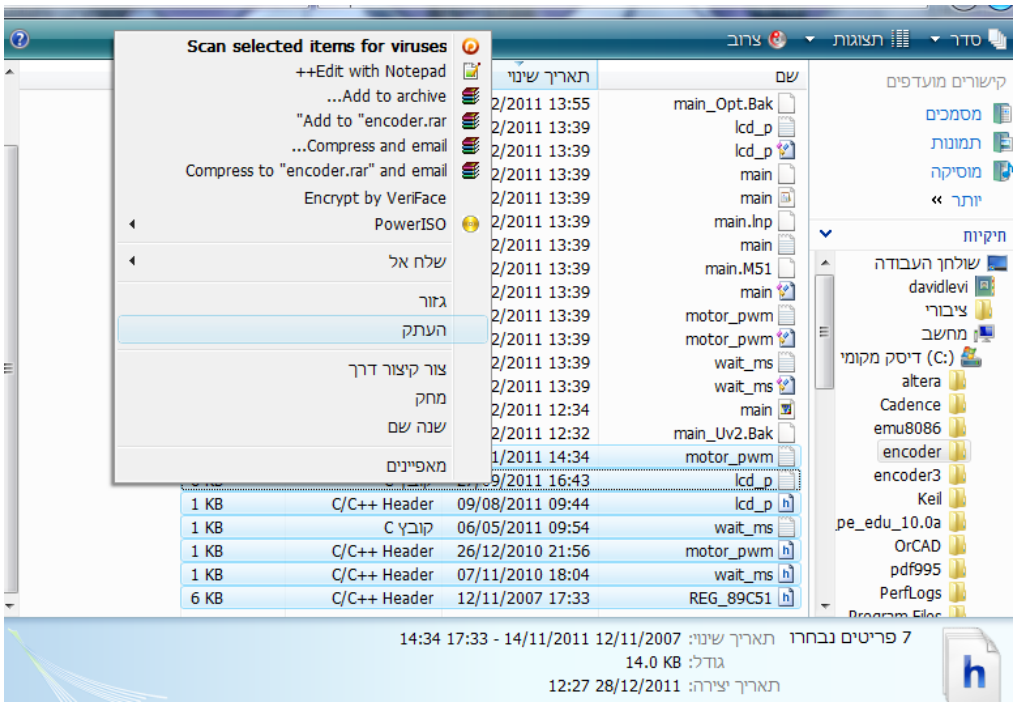
לאחר מכן פותחים NEW FILE

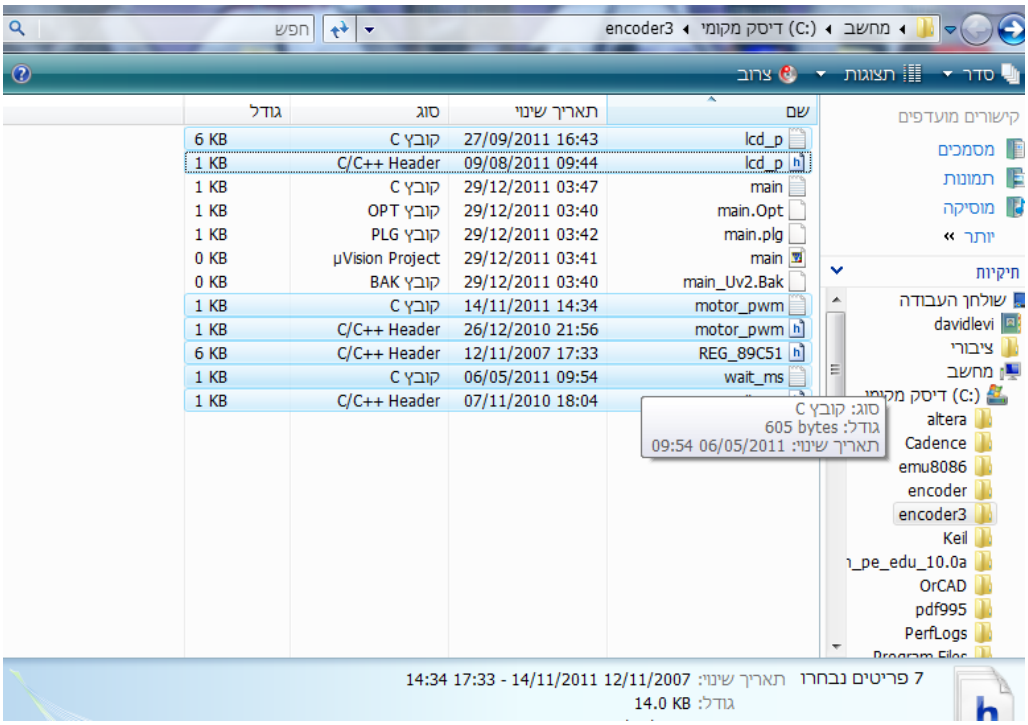


כותבים את התוכנית ואז שומרים .

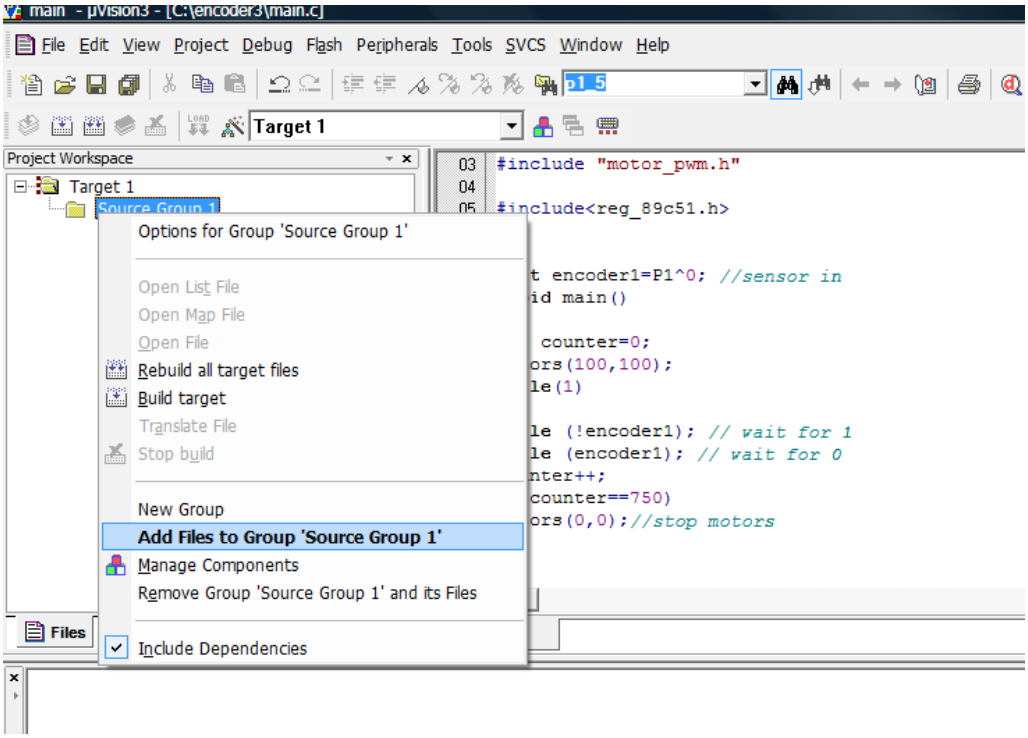


לאחר מכן הולכים אל תיקיה שרוצים להעתיק לצרף ממנה קבצים ומעתיקים לתיקיה שפתחנו את ההדרים את ה- REG ואת ה- C לתיקיה החדשה שפתחנו .

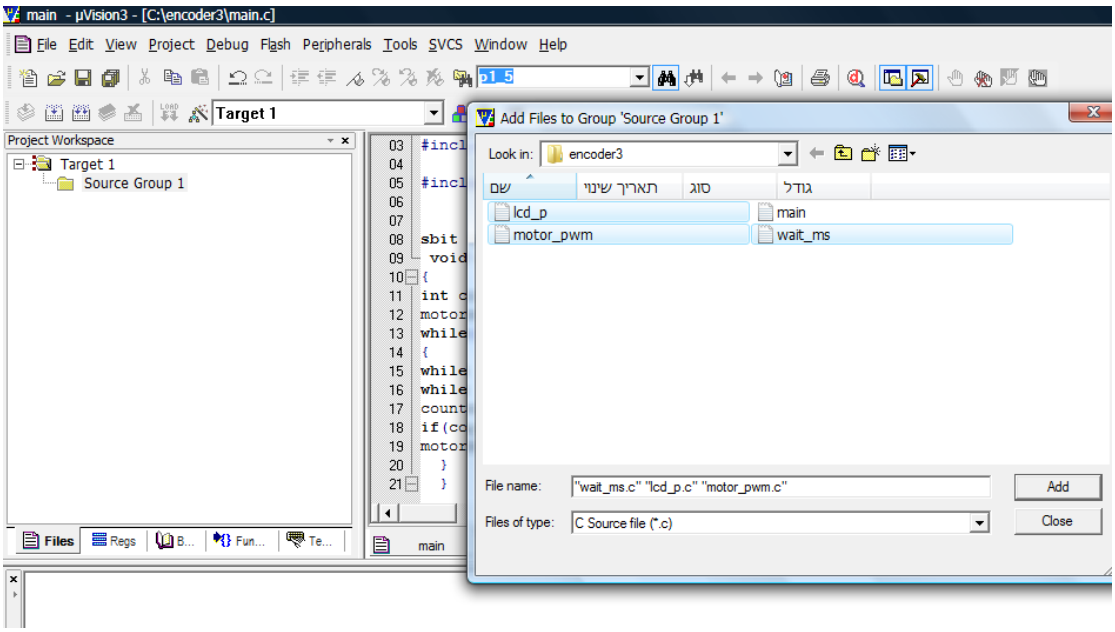




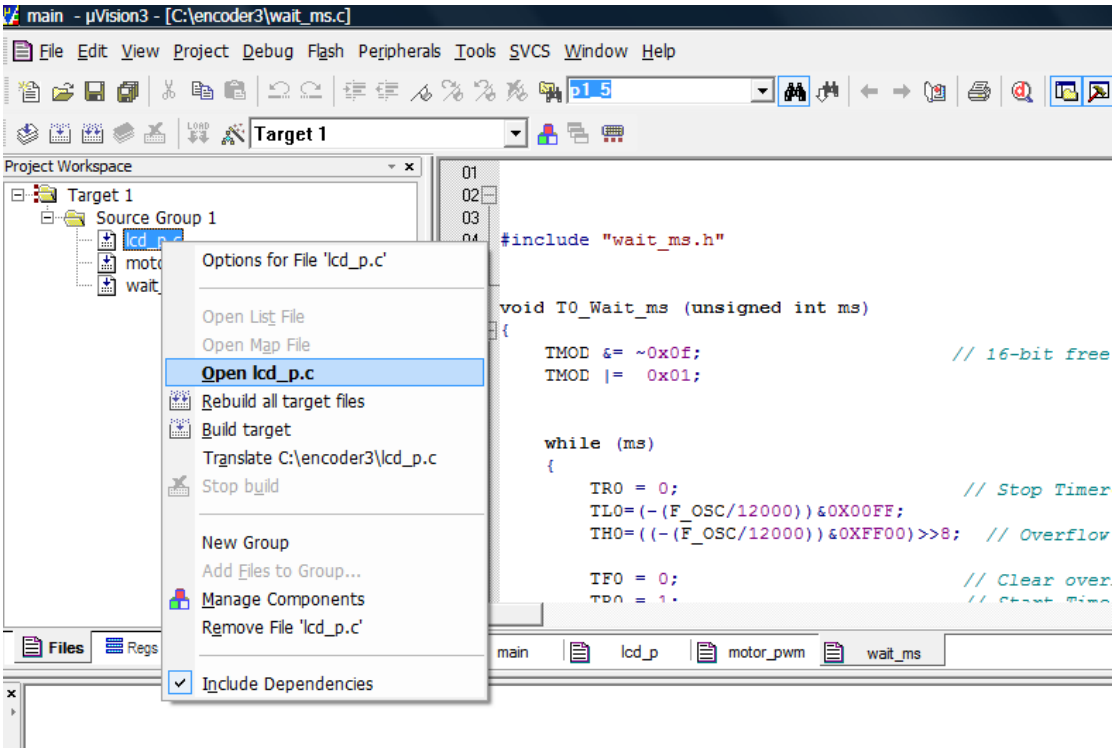
לאחר מכן מצרפים את הקבצים כדי לשמור ולעביר קומפילציה



מצרפים את הכול



ואז פותחים את מה שצרפנו



לאחר מכן שומרים ומקמפלים ואז אפשר לצרוב לבקר.