

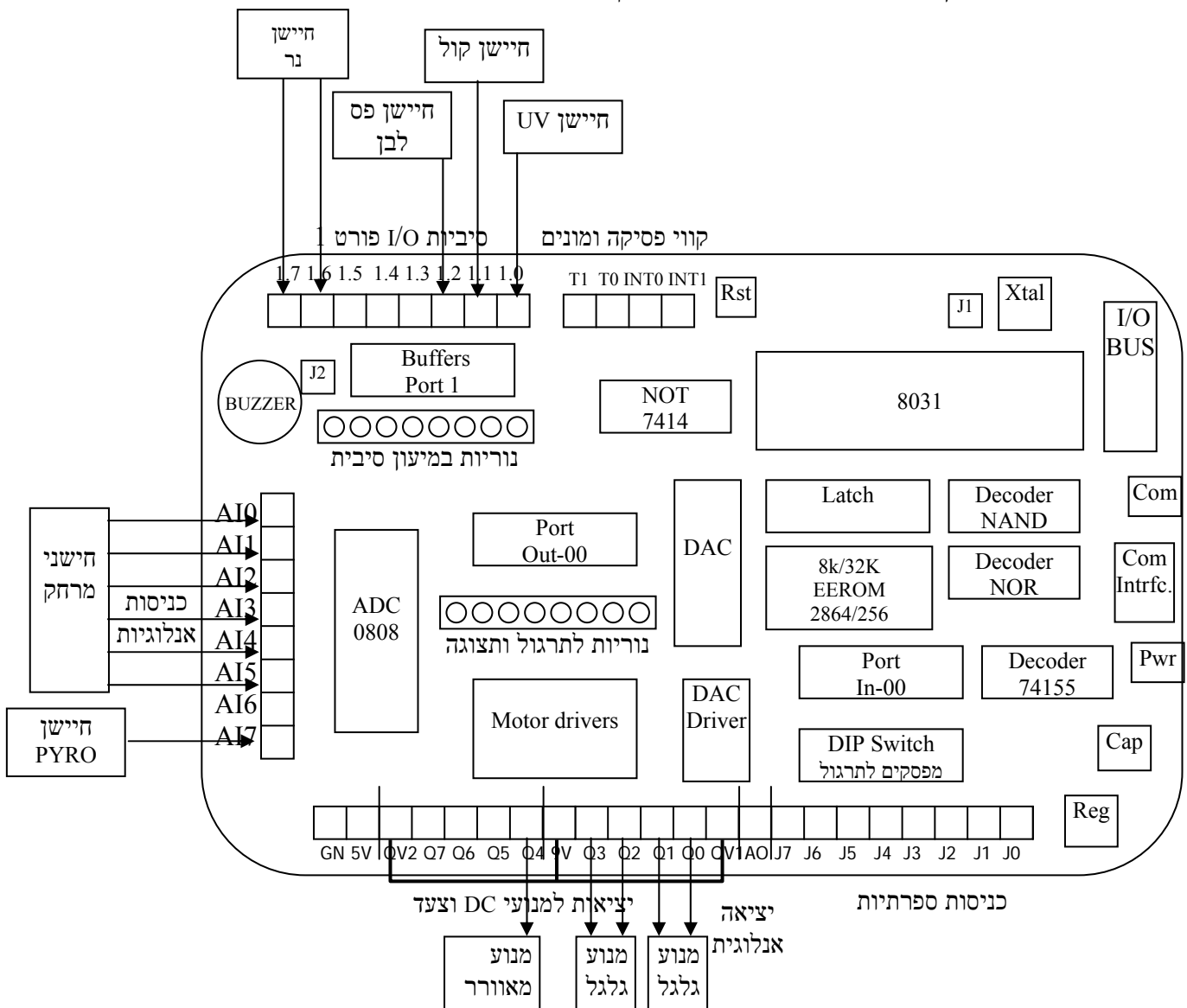


## דיאגרמת חיבורים חשמליים לרובונר

רובוט לתערוכת רובונר זקוק, בנוסף לכרטיס רובוטיקה ולמבנה, למרכיבים הבאים:

1. 2 מנועי DC לגלגלים
2. מנוע DC למאוורר
3. 6 חישני מרחק אנלוגיים
4. 1 חישן פס לבן דיגיטלי
5. 1 חישן קול דיגיטלי
6. 1 חישן UV דיגיטלי
7. 1 חישן אנלוגי PYRO

אופן חיבור המרכיבים לכרטיס הרבוטיקה DSM-2095 מתואר בדיאגרמה הבאה:



1. מנועי DC

כל מנוע מחברים לשתי יציאות של דרייבר (Q0-Q7). בצורה זו ניתן לשלוט על כיוון הסיבוב של המנוע. רצוי לחבר כל מנוע לזוג יציאות קרובות: Q1-Q0, Q3-Q2, Q5-Q4, Q6-Q5.

קיימות שתי אפשרויות לשליטה על מהירות המנוע. האחת על ידי חיבור כניסת QV של הדרייבר ליציאת ה-DAC של הכרטיס. שיטה זו פשוטה ביותר, אבל לא מאפשרת להגיע להספק מכסימלי ומחייבת ששני המנועים יופעלו תמיד באותה המהירות.

השיטה השנייה היא בשיטת PWM (Pulse Width Modulation). בשיטה זו אנו מספקים למנועים פולסים ברוחב דופן משתנה ורוחב הדפקים קובע את מהירות המנוע. בספר "רובוטיקה ומערכות ממוחשבות" יש התנסות העוסקת בבקרת מנועים.

נספח A מתאר תוכניות המאפשרות בקרת מהירות והפעלה לכל מנוע המחובר לזוג יציאות (כמתואר למעלה).

חשוב מאוד לזכור

אסור לשנות כיוון למנוע מבלי לעוצרו תחילה. שינוי כזה גורם להיווצרות מתח גבוה ביותר על הדרייבר ועל המנוע. הדרייברים בכרטיס מיועדים לזרמים גבוהים וכוללים דיודות הגנה אבל הם לא יוכלו לעמוד בפולסי מתח גבוהים כאלה.

2. חיישן מרחק

חיישן המרחק הוא רכיב הניזון ממתח של 5V ומוציא מתח אנלוגי כפונקציה של מרחקו מחפץ כלשהו. הרכיב שולח פולסי אור אינפרא אדום חזקים ומתייחס לפולס האור המוחזר.

החיישן מגיע עם ספר חיישן באנגלית. רצוי מאוד לקרוא היטב את ספר החיישן והוראות השימוש.

**רצוי מאוד לא לחבר את חיישני המרחק במקביל למתח של 5V של בקר הכרטיס. לרכיב יש צריכת זרם רגעית גבוהה מאוד. כמה חיישנים ביחד עלולים להפיל את מתח הבקר לזמן קצר ולשגע את המערכת. מומלץ לספק לחיישנים מתח 5V ממייצב נפרד של 5V עם קבל גבוה במוצא.**

ישנם חיישנים שהאות במוצא שלהם רועש יותר וישנם שרועש פחות. חיבור קבל סינון של 0.1uF מקטין במידה ניכרת את הרעשים. החברה ממליצה לחבר קבל של 10uF במקביל לכניסות המתח של החיישן.

הרעשים רוכבים על האות, כך שדגימת האות מספר פעמים ולקיחת הערך הנמוך, יכולה לשמש כסינון נוסף. כמות הרעש משתנה מחיישן לחיישן ואינה אחידה. מבדיקה שערכנו, לקיחת הערך הנמוך מתוך 8 עד 12 דגימות נותנת קריאה יציבה למדי.

חשוב לזכור:

הרכיב מוגבל לתחום מרחק מוגדר. צריך למנוע קבלת קריאות מטווח קטן מעשרה ס"מ כי הן עלולות להיחשב כמרחק גדול יותר.

הקריאות המתקבלות מחיישנים שונים אינן זהות. יש לכייל כל חיישן בנפרד.

הרכיב אינו ליניארי. אם הליניאריות חשובה, ניתן להשתמש בטבלת המרה על מנת להמיר את הקריאות לקריאות ליניאריות. בספר "יסודות הבקר הזעיר 8051" מתואר אופן השימוש בטבלת המרה.

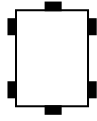
כדי לנוע לאורך קיר, מחברים חיישן לכל דופן של הרובוט. רצוי אפילו שניים לכל דופן.

רצוי להעביר את דגימת החיישנים לרוטינת פסיקה של קוצב הזמן, שתקרא כל חיישן מספר פעמים ותעביר את הערך הנמוך לתא בזכרון. התוכנית הראשית תתייחס רק לערכי החיישנים שנמצאים בתאי הזכרון.

נספח B מתאר אופן קריאת חיישנים על ידי תוכנית פסיקה כולל מציאת הערך הנמוך של מספר קריאות.

### 3. בקרת תנועה בעזרת חיישני מרחק

כדי לשלוט על תנועת הרובוט בצורה מלאה, מחברים 6 חיישני מרחק סביבו בצורה הבאה:

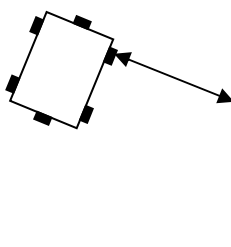


נסיעה ישר נעשית על ידי הנעת שני המנועים קדימה. פנייה ימינה נעשית על ידי הורדת מהירות גלגל ימין או עצירתו אם רוצים פנייה חדה. פנייה שמאלה נעשית על ידי הורדת מהירות גלגל שמאל או עצירתו אם רוצים פנייה חדה.

סבוב ימינה במקום נעשית על ידי הנעת גלגל שמאל קדימה וגלגל ימין אחורה. סבוב שמאלה במקום נעשית על ידי הנעת גלגל ימין קדימה וגלגל שמאל אחורה.

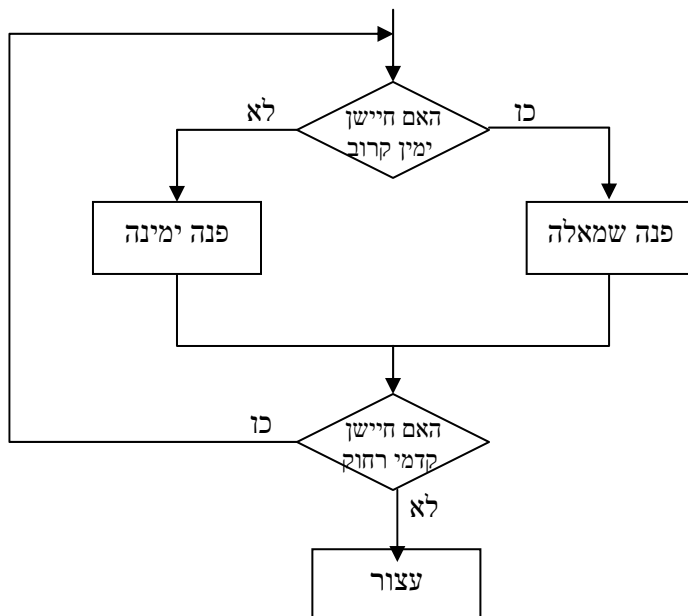
נסיעה לאורך קיר נעשית על ידי קריאת חיישן המרחק שבדופן וקבלת החלטות התנועה כל הזמן.

צריך לזכור שחיישן המרחק קורא את המרחק מהקיר מהקיר גם באלכסון. הוא אינו קורא את ההיטל אלא את האלכסון.



התוכנית הבאה היא תוכנית פשוטה ואמינה למדי, המניעה את הרובוט לאורך קיר ימין. תוכנית זו מבצעת פנייה (לא חדה) ימינה כאשר הרובוט רחוק מהקיר (או גם במרחק הרצוי מהקיר) ופנייה (לא חדה) שמאלה כאשר הרובוט קרוב לקיר. למעשה מבצע הרובוט כל הזמן פניות תוך כדי התקדמות.

התוכנית בנויה על האלגוריתם הבא:



נספח C מתאר תוכנית זו המשלבת עבודה עם PWM וקריאת חיישנים ברקע, כולל מציאת הערך הנמוך.

ניתן לשכלל את התוכנית לתנועה לפי שני חיישנים.

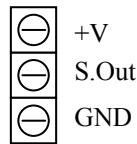
רצוי לנתח את תנועת הרובוט בעזרת סימולציות של מצבים ותגובות הרובוט למצבים אלה.

**4. חיישן קול**

הפעלת הרובוט בתחרות נעשית על ידי קירוב זמזם (buzzer) אל חיישן הקול.

חיישן הקול (SESPET-81) כולל מיקרופון, מגבר שמע ומחבר התחברות.

למחבר 3 נקודות התחברות:



לנקודות +V ו-GND מחברים את ספק הכוח. ל-GND מחברים את ההדק השלילי של ספק הכוח. מתח הספק צריך להיות בן 5V. ניתן לחבר את החיישן למחבר 5V של הכרטיס. הוא אינו צורך זרם ובזמן הפנייה אליו, הרובוט ומערכתיו אינן מופעלות.

את הנקודה S. Out (Signal Out) מחברים לכניסה דיגיטלית של הבקר. ניתן ורצוי לחבר אותו לכניסה של פורט 1. במידה והחיישן והבקר מחוברים לספקים שונים, אזי יש לחבר את הנקודה GND של החיישן לנקודה GND של הבקר.

האות ביציאה Sout, הוא גל ריבועי בתדירות הצליל.

רצוי מאוד לקבוע גליל קרטון קטן (עדיף בצורת חרוט) סביב המיקרופון כדי לשפר את רגישות החיישן.

רצוי לרשום תוכנית הסופרת את הזמן של מספר מוגדר של עליות וירידות. ככל שהזמן קצר יותר התדר גבוה יותר.

אפשרות אחרת היא לספור את מספר העליות והירידות המופיעות בלולאת מנייה מסוימת. ככל שהמספר גדול יותר, התדר גבוה יותר.

נספח D מתאר תוכנית כזו.

המערכת צריכה להגיב רק לצליל בתדירות מעל 2000 HZ.

הטיימרים של ה-8051 יכולים לשמש כמונים של אותות חיצוניים. כניסות הטיימרים מופיעים על הכרטיס DSM-2095 ומסומנים T0 ו-T1.

ניתן לחבר את יציאת חיישן הקול לכניסת אחד הטיימרים.

התוכנית הראשית מאתחלת את הטיימר לפעול כמונה. לאחר מכן, מאפסת התוכנית הראשית את המונה, מבצעת לולאת השהייה וקוראת את המונה. אם המספר קטן מהערך הרצוי (עלינו למצוא תדירות מעל 2000HZ), חוזרת התוכנית הראשית על התהליך (איפוס, השהייה, קריאה ובדיקה). רק כאשר מתקבלת מנייה גדולה מהערך הרצוי, ממשיכים בתוכנית.

נספח D מתאר גם תוכנית כזו.

זכור, זיהוי בבקרה ממוחשבת מבצעים לאחר בדיקה אם מספר גדול או קטן מערך מסוים ולא אם שווה לערך מסוים.

לסנן אותות בתוכנה עדיף על סינון בחומרה.

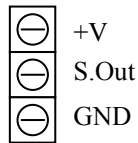
**5. זמזם**

מודול הזמזם (SESPET-02) הוא מודול המכיל מפסק לחצן, זמזם ומחבר לסוללה 9V.

לאחר חיבור סוללה של 9V לשקע הסוללה, לחיצה על המפסק תגרום לצפצוף של הזמזם.

חיישן פס לבן (SESPET-82) כולל LED עוצמה חזקה, חיישן אור ומחבר התחברות.

למחבר 3 נקודות התחברות:



לנקודות +V ו-GND מחברים את ספק הכוח. ל-GND מחברים את ההדק השלילי של ספק הכוח. מתח הספק צריך להיות 5V.

את הנקודה S. Out (Signal Out) מחברים לכניסה דיגיטלית או אנלוגית של הבקר.

במידה והחיישן והבקר מחוברים לספקים שונים, אזי יש לחבר את הנקודה GND של החיישן לנקודה GND של הבקר.

כאשר מספקים מתח למודול, נדלק ה-LED באור אדום.

כאשר מפנים את המודול לכיוון הקרקע, מוחזר אור אל החיישן. רקע שחור בולע את האור ורקע לבן מחזיר את האור. עוצמת האור חזקה במיוחד, כדי להקטין ככל האפשר את השפעות אור הסביבה המשתנה. למרות זאת, כדאי לקבוע את הכרטיסון מתחת לרובוט, כך שאור סביבה ישיר לא יגיע אליו. עדיף גם ליצור דפנות סביבו, כדי לחסום את אור הסביבה.

ל-SES שני חיישני פס לבן.

אחד כרטיס שחור בגודל 5 X 5 ס"מ והשני כרטיס ירוק קטן יותר.

הכרטיס השחור הוא חיישן אנלוגי (ניתן לחבר אותו גם לכניסה הדיגיטלית) שהמתח המוצא הוא פונקציה של האור המוחזר.

הכרטיס הירוק כולל גם מעגל הגברה מתוחכם המבוסס על שני מגברי שרת וכולל פוטנציומטר לכיוון רגישות החיישן. הכרטיס מוציא אות דיגיטלי, כך שעדיף לחברו לכניסה דיגיטלית.

שני הכרטיסים מתוכננים לקריאה טובה בגובה של בין 1.5 ל-3 ס"מ מהקרקע.

הדבק בריסטול לבן על בריסטול שחור. חבר את החיישן להדקי 5V של הכרטיס ואת יציאתו אל כניסה של פורט 1.

הפעל את כרטיס הרובוטיקה ולחץ RESET. העבר את החיישן מעל הבריסטולים. בדוק שהנורית הירוקה מגיבה לרקעים השונים שעל הקרקע. מצא את הגובה האופטימלי.

### **חשוב לדעת**

גם רקע שחור מחזיר אור. קיימים רקעים שמחזירים לא מעט אור.

ניתן גם לחבר את הכרטיס השחור לכניסה אנלוגית.

רשום והרץ תוכנית הקוראת את הערוץ האנלוגי שאליו חיברת את החיישן (ראה ספר רובוטיקה ומערכות ממוחשבות בהוצאת SES) ומוציאה את הקריאות לנוריות.

מצא את המספרים הבינאריים המתקבלים מעל רקע שחור ומעל פס לבן בהתאם למרחק המודול ברובוט מהקרקע. קבע לך את המספר שיגדיר זיהוי פס לבן.

בצע בדיקות גם עם הפרעות של אור סביבה.

זכור, זיהוי בבקרה ממוחשבת מבצעים לאחר בדיקה אם מספר גדול או קטן מערך מסוים ולא אם שווה לערך מסוים.

**7. חיישן UV**

חיישן זה כולל נורת UV אותה יש להרכיב על כרטיס בקר. החיישן מגיע עם ספר חיישן באנגלית. רצוי מאוד לקרוא היטב את ספר החיישן והוראות השימוש.

**שים לב:**

הנורה רגישה ללכלוך ושמוניות ואין להחזיק אותה בידיים.

את הכרטיס ניתן להזין במתח מיוצב 5V או במתח ישר לא מיוצב בתחום 10-30V. יש להתאים את הגשרים בכרטיס לפי הספר.

לכרטיס מחבר עם 5 נקודות חיבור: +, -, 1, 2, 3. את המתח מספקים לנקודות + ו- בהתאם.

נקודה 1 מספקת פולס דיגיטלי ברמת CMOS כאשר מזהה קרינת להבה.  
נקודה 2 מספקת אות הפוך לנקודה 1.  
נקודה 3 מספקת אות ביציאת Open Collector.

כאשר מזהה להבה, מתקבלים ביציאות דפקים ברוחב של 10ms. ניתן להרחיב את הדפקים על ידי הוספת קבל בנקודות המסומנות CX.

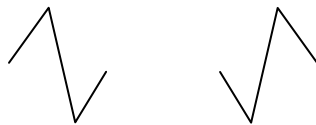
רצוי לא להגיב לפולס הראשון, אלא לספור מספר פולסים בזמן קצר כדי לוודא שאכן להבה דולקת.

מחברים את יציאת החיישן (נקודה 1) לכניסה של פורט 1.

בדיקת החיישן נעשית בעזרת הפקודה JB או JNB.

**8. חיישן PYRO**

חיישן זה הוא חיישן המזהה תנועה של גוף פולט חום. זהו חיישן הקולט קרינה אינפרא אדומה. החיישן מגיע עם ספר חיישן באנגלית. רצוי מאוד לקרוא היטב את ספר החיישן והוראות השימוש. חיישן זה מוציא אות אנלוגי כאשר גוף פולט חום חוצה אותו. כיוון החצייה קובע את צורת האות.



מכיוון שהנר ניח, עלינו לסובב את החיישן מצד לצד, עד לזיהוי האות הנ"ל. האות מתקבל כאשר החיישן מול מקור החום.

עדיף לבצע סריקה כפולה – מימין לשמאל ומשמאל לימין.

מומלץ על ידי חברת Acroname לבנות שופר קרטון מסביב לחיישן והכולל עדשה מיוחדת. הדבר מאפשר התמקדות ברורה וחדה יותר על מיקום מקור החום.

שים לב שיש חשיבות לאופן מיקום החיישן בכרטיס ביחס לתנועת הרובוט. לחיישן יש בליטת זיהוי.

**9. חיישן נר SES**

חיישן חדש זה הוא חיישן נר דיגיטלי בעל מבנה מיוחד. הוא כולל שלושה חיישני אינפרא אדום. שני חיישנים נמצאים בקו אחד ומופרדים בעזרת מחיצה אטומה. אור הנר נופל על שני החיישנים רק כאשר נמצא הרובוט מול הנר. כאשר נמצא הרובוט בזווית לנר, נופל אור הנר רק על אחד החיישנים.

החיישן השלישי נמצא מתחת לשני החיישנים ונועד לבטל את אור הסביבה. עוצמת המתח המתקבלת מכל חיישן משווית עם המתח המתקבל בחיישן השלישי. לכל חיישן פוטנציומטר לכיולו.

למודול החיישן שתי יציאות דיגיטליות אותן מחברים לכניסות דיגיטליות של הבקר. זיהוי הנר וכיוונו פשוט ביותר:

- 00 - אין נר דולק בסביבה
- 01 - יש נר דולק משמאל למודול
- 10 - יש נר דולק מימין למודול
- 11 - יש נר דולק ממול למודול.

**10. מאורר**

יחידה זו כוללת מאורר. ניתן להשתמש במאוררים שונים.

המאורר מיועד למתח של 12V.

מתקן המאורר מאפשר התקנתו על משטח כלשהו.

את המאורר ניתן לחבר לשתי יציאות של הדרייבר (Q0-Q7) או בין יציאה ל-12V.

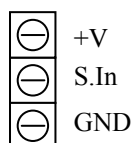
**11. דרייבר חד כיווני**

מודול זה (SESPET-08) אינו חיישן אלא רכיב הנעה.

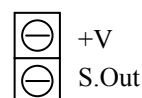
משתמשים במודול זה כאשר כל יציאות הדרייברים שבכרטיס DSM-2095 נוצלו ואנו זקוקים להפעיל מנוע נוסף או מנורה. מודול זה מאפשר להפעיל ולנתק את המנוע אבל לא לשלוט על כיוון הסיבוב.

המודול כולל טרנזיסטור הספק, מחבר מבוא ומחבר מוצא.

מחבר מבוא



מחבר מוצא



את הצרכן (המנוע או המנורה) מחברים בין נקודות ההתחברות +V ו-GND שבמחבר המוצא.

לנקודות +V ו-GND שבמחבר המבוא, מחברים את ספק הכוח. ל-GND מחברים את ההדק השלילי של ספק הכוח. מתח הספק צריך להיות בין 9V ל-12V.

את הנקודה S. In (Signal In) מחברים לאחת מיציאות פורט 1 של הכרטיס DSM-2095. הפעלת סיבית זו בעזרת הפקודה

SETB P1.2

למשל, תפעיל את הצרכן.

CLR P1.2

איפוס סיבית זו בעזרת הפקודה

למשל, תנתק את הצרכן.

**תוכנית ברקע:**

הדרך הטובה ביותר לבצע בקרת מנוע בעזרת PWM היא בעזרת קוצב זמן. אנו מפעילים בתוכנית הראשית את קוצב הזמן ודואגים שיתן פסיקה כל פרק זמן מוגדר. לאחר ביצוע מהלך זה משוחררת התוכנית הראשית לביצוע התפקידים הנוספים של הבקר ותוכנית הפסיקה של קוצב הזמן פועלת ברקע במקביל ודואגת לבקרת מהירות המנוע.

**התקשורת בין התוכנית הראשית לתוכנית הפסיקה של קוצב הזמן:**

כדי שהתוכנית הראשית תוכל להעביר לתוכנית הפסיקה את אופן הפעלת המנוע (ON או OFF וכיוון) ואת המהירות הדרושה של המנוע, אנו מגדירים שני תאי זכרון המשמשים לתקשורת בין התוכנית הראשית לתוכנית הפסיקה.

בתא אחד, שנכנה אותו בשם IMAGE (בבואת הפורט – Port Image), תכניס התוכנית הראשית את המספר אותו אנו רוצים להוציא לפורט המוצא. מספר זה יקבע אם המנועים יופעלו ולא יזהו כיוון.

בתא שני, שנכנה אותו בשם PWM1 (PWM עבור מנוע 1 שמחובר לערוצים Q0 ו-Q1), תכניס התוכנית הראשית מספר בתחום 0-255 (00-FF). מספר זה יקבע את משך ה-ON של המנוע. ככל שהמספר גדול יותר, משך ה-ON ארוך יותר.

**פעולת תוכנית הפסיקה:**

תוכנית הפסיקה משתמשת בשני תאים נוספים. תא אחד נקרא CNTR. תא זה גדל ב-1 בכל פסיקה. לאחר שהוא מגיע ל-FF ומתקבלת פסיקה נוספת הוא מתאפס.

תוכנית הפסיקה משווה בין המספר שבתא PWM1 למספר שבתא CNTR. אם CNTR קטן מ-PWM1, היא מפעילה את המנוע. אם CNTR גדול או שווה ל-PWM1, מכבה תוכנית הפסיקה את המנוע. היא תפעיל אותו שוב לאחר שה-CNTR יתאפס ושוב יהיה קטן מ-PWM1.

**תוכנית 1:**

התוכנית הבאה מבצעת בקרת מנוע אחד, המחובר לערוצי המוצא Q1, Q0.

התוכנית הראשית קוראת את מצב המפסקים ומאחסנת אותו בתא PWM1. כלומר, מצב המפסקים קובע את מהירות המנוע.

התוכנית מורכבת מתוכנית ראשית ותוכנית פסיקה של Timer0.

התוכנית הראשית מתחילה באיתחול הטיימר ואיפוס ה-CNTR. לאחר מכן יכולה התוכנית הראשית להתעלם מבקרת המנוע.

כדי להפעיל את המנוע, על התוכנית הראשית לכתוב מספר בתא IMAGE. 01 יסובב את המנוע לכיוון אחד, 02 יסובב את המנוע לכיוון השני ו-00 יעצור אותו. הכתיבה נעשית פעם אחת בכל פעם שנדרשים הפעלה או כיבוי של המנוע.

בכל פעם שנדרש שינוי במהירות המנוע, על התוכנית הראשית לכתוב מספר בין 00 ל-FF בתא PWM1. מספר זה יקבע את מהירות המנוע.



```

PWM1 EQU 41H ;determines Q0,Q1 motor speed
CNTR EQU 45H
IMAGE EQU 46H
PORTA EQU 0FF00H
TINTR EQU 00H ;for 10ms timer interrupts
; ;increase this number for shorter intervals
;
; ORG 200BH
; LJMP TMR0
;
; ORG 2100H
MAIN: MOV TMOD,#01H ;mode1, C/T'=0,GATE=0
MOV TH0,#0FFH
MOV TL0,#TINTR
SETB EA ;enables all interrupts
SETB ET0 ;enables timer0 interrupt
SETB TR0 ;starts the timer
MOV CNTR,#0
MOV DPTR,#PORTA
MOV IMAGE,#01 ;motor Q1-Q0 ON
MAI2: MOVX A,@DPTR ;variable speed according to
MOV PWM1,A ;switches to motor1
SJMP MAI2
;
TMR0: CLR TR0 ;stops the timer
PUSH ACC
PUSH DPH
PUSH DPL
PUSH PSW
;
MOV DPTR,#PORTA
INC CNTR
MOV A,PWM1
CLR C
SUBB A,CNTR ;compare PWM1 with CNTR
JNC TMR2 ;jump if CNTR is smaller than PWM1
MOV A,IMAGE ;CNTR is bigger than PWM1
CLR ACC.0 ;so it stops the motor
CLR ACC.1
SJMP TMR3
TMR2: MOV A,IMAGE ;PWM1 is still bigger than CNTR
TMR3: MOVX @DPTR,A ; so it outputs IMAGE
;
MOV TH0,#0FFH ;to create the next timer interrupt
MOV TL0,#TINTR
SETB TR0
POP PSW
POP DPL
POP DPH
POP ACC
RETI

```

חשוב מאד להבין תוכנית זו, כי היא הבסיס לכל העבודה עם תוכניות ברקע ושימוש בקוצב זמן לבקרה בזמן אמת.

```

#include "8052.h"

unsigned char at 0xff00 xdata IOPORT;
unsigned char PWM1;
unsigned char PWM_CNTR;
unsigned char OPORT_IMAGE;
unsigned char TEMP_IMAGE;

unsigned char TINTH=0xFF;
unsigned char TINTR=0;

void tmr0() interrupt 1 using 2
{
    TR0=0;                //stops the timer
    TH0=TINTH;            //to create the next timer interrupt
    TL0=TINTL;
    TR0=1;                //enables the timer

    TEMP_IMAGE = OPORT_IMAGE;
    PWM_CNTR = PWM_CNTR + 4;
    if (PWM_CNTR > PWM) //compare PWM counter with PWM1 value
        TEMP_IMAGE = TEMP_IMAGE & 0xfc; //and stop motor if bigger
    IOPORT = TEMP_IMAGE;
}

void main(void)
{
    TMOD=0x21;           //keep timer1 for comm. and timer0 in mode1
    TH0=TINTH;           //timer0 interval
    TL0=TINTR;
    PWM_CNTR=0;
    ET0=1;
    EA=1;
    TR0=1 ;              //start timer0
    OPORT_IMAGE = 01;
    while (1)
    {
        PWM1 = IOPORT;
    }
}

```

הערך TINTR קובע את הזמן שיעבור בין פסיקה לפסיקה. ככל שנגדיל מספר זה, נקבל זמנים קצרים יותר ותנועת המנוע תהיה חלקה יותר. מצד שני, הדבר יעסיק את ה-CPU יותר בתוכנית הפסיקה ופחות בתוכנית הראשית. המספר 00 מאפשר לנו לחוש את מיתוגי המנוע. בפועל עדיף להגדילו ל-80H.

מחזור PWM מסתיים כל 256 פסיקות, כלומר, מחזור אחד אורך 77ms בערך (0.08 שניה) השווה ל-12 מחזורים בשניה. הדבר אטי במקצת (אנו נחוש את מיתוגי המנועים). אם אנו מבקשים לבצע בקרת תנועה מהירה, נקבל תגובה איטית של המערכת.

הגדלת הערך TINTR יקצר את הזמן בין הפסיקות אבל יקציב פחות זמן לתוכנית הראשית. המעבד יהיה עסוק הרבה זמן בתוכנית הפסיקה. במקום זאת הגדלה כפולה של תא CNTR בכל מחזור פסיקה, מקצרת את מחזור PWM לחצי, מבלי להשפיע על דיוק ה-PWM. ארבע הגדלות בכל מחזור פסיקה, מקצרת את מחזור PWM לרבע. ההגדלה נעשית על ידי הוספת ההוראה INC CNTR אחרי השורה המסומנת בתוית TMR1 שבתוכנית. המנוע זרם ישר של Micro Motors, המסופק על ידי SES, מיועד למחזור PWM של 50 עד 100 מחזורים בשניה. הגדלה של CNTR ב-4 בכל מחזור פסיקה תיתן לנו את הקצב הרצוי.

כאשר אנו רוצים לבקר יותר ממנוע אחד ולאפשר מהירות שונה לכל אחד מהמנועים, אנו זקוקים לתאים נוספים. נכנה תאים אלה (תא אחד לכל מנוע) PWM1, PWM2, PWM3 ו-PWM4.

מנוע 1 מתחבר לערוצים Q0, Q1, מנוע 2 לערוצים Q2, Q3, מנוע 3 לערוצים Q4, Q5, ומנוע 4 לערוצים Q6, Q7.

תוכנית הפסיקה משווה בין ערך ה-CNTR לכל אחד מהתאים PWM1-PWM4 ומפעילה או מכבה את המנוע בהתאם.

PWM1	EQU	41H	;determines Q0,Q1 motor speed
PWM2	EQU	42H	
PWM3	EQU	43H	
PWM4	EQU	44H	
CNTR	EQU	45H	
IMAGE	EQU	46H	
PORTA	EQU	0FF00H	
TINTR	EQU	00H	;for 10ms timer interrupts
			;increase this number for shorter intervals
			;
	ORG	200BH	
	LJMP	TMR0	
			;
	ORG	2100H	
MAIN:	MOV	TMOD,#01H	;mode1, C/T'=0,GATE=0
	MOV	TH0,#0FFH	
	MOV	TL0,#TINTR	
	SETB	EA	;enables all interrupts
	SETB	ET0	;enables timer0 interrupt
	SETB	TR0	;starts the timer
	MOV	CNTR,#0	
	MOV	PWM1,#0FFH	;maximum speed to motor1
	MOV	PWM2,#0C0H	;fast speed to motor2
	MOV	PWM3,#80H	;medium speed to motor3
	MOV	PWM3,#40H	;slow speed to motor 4
MAI2:	MOV	DPTR,#PORTA	
	MOVX	A,@DPTR	;motors ON according to switches
	MOV	IMAGE,A	
	SJMP	MAI2	
			;
TMR0:	CLR	TR0	;stops the timer
	PUSH	ACC	
	PUSH	DPH	
	PUSH	DPL	
	PUSH	PSW	
			;
	MOV	DPTR,#PORTA	
	INC	CNTR	
	MOV	A,PWM1	
	CLR	C	
	SUBB	A,CNTR	;compare PWM1 with CNTR
	JNC	TMR2	;jump if CNTR is smaller than PWM1

	MOV	A,IMAGE	;CNTR is bigger than PWM1
	CLR	ACC.0	;so stop the Q1,Q0 motor
	CLR	ACC.1	
	SJMP	TMR3	
TMR2:	MOV	A,IMAGE	;do not change image
TMR3:	MOV	IMAG2,A	;save image in a temporary cell
	;		
	MOV	A,PWM2	
	CLR	C	
	SUBB	A,CNTR	;compare PWM2 with CNTR
	JNC	TMR4	;jump if CNTR is smaller than PWM2
	MOV	A,IMAG2	;CNTR is bigger than PWM2
	CLR	ACC.2	;so stop the Q3,Q2 motor
	CLR	ACC.3	
	MOV	IMAG2,A	;update imag2
	;		
TMR4:	MOV	A,PWM3	
	CLR	C	
	SUBB	A,CNTR	;compare PWM3 with CNTR
	JNC	TMR5	;jump if CNTR is smaller than PWM3
	MOV	A,IMAG2	;CNTR is bigger than PWM3
	CLR	ACC.4	;so stop the Q5,Q4 motor
	CLR	ACC.5	
	MOV	IMAG2,A	;update imag2
	;		
TMR5:	MOV	A,PWM4	
	CLR	C	
	SUBB	A,CNTR	;compare PWM4 with CNTR
	JNC	TMR6	;jump if CNTR is smaller than PWM4
	MOV	A,IMAG2	;CNTR is bigger than PWM4
	CLR	ACC.6	;so stop the Q7,Q6 motor
	CLR	ACC.7	
	MOV	IMAG2,A	;update imag2
TMR6:	MOV	A,IMAG2	
	MOVX	@DPTR,A	;output imag2
	;		
	MOV	TH0,#0FFH	;to create the next timer interrupt
	MOV	TL0,#TINTR	
	SETB	TR0	
	POP	PSW	
	POP	DPL	
	POP	DPH	
	POP	ACC	
	RETI		

כדי לא להשפיע על המספר, שהתוכנית הראשית מוציאה לפורט (IMAGE), משתמשים בתא זכרון נוסף IMAG2.

בתא IMAGE כותבת התוכנית הראשית מספר בהתאם למנועים אותם היא רוצה להפעיל. נתון זה מועבר לתא IMAG2 בתחילת כל מחזור פסיקות. לדוגמא, המספר 85H (1000101) יפעיל את המנועים Q0,Q1 ו-Q2,Q3 בכיוון אחד, יעצור את מנוע Q4,Q5 ויסובב את מנוע Q6,Q7 לכיוון השני.

מחזור פסיקות מתחיל כאשר תא CNTR מאופס. כל פסיקה מעלה תא זה ב-1. הוא סופר מ-00 עד FF וחוזר חלילה.

כאשר ה-CNTR מאופס, אנו מעבירים את תוכן IMAG2 לתא IMAG2. בכל פעם שתוכנית הפסיקה מוצאת שיש לכבות את אחד המנועים, היא מאפסת את הסיביות הקשורות למנוע זה.

בסוף רוטינת הפסיקה (לאחר ההשוואות), מוציאה התוכנית את תוכן IMAG2 לפורט המוצא.

לשם המחשה, קובעת התוכנית למנוע 1 מהירות מכסימלית, למנוע 2 מהירות גבוהה, למנוע 3 מהירות בינונית ולמנוע 4 מהירות נמוכה.

קביעת המנועים המופעלים וכיוון סיבובם נעשה בעזרת המפסקים.

```

#include "8052.h"
unsigned char at 0xff00 xdata OPORT;
unsigned char PWM1;
unsigned char PWM2;
unsigned char PWM3;
unsigned char PWM4;
unsigned char PWM_CNTR;
unsigned char OPORT_IMAGE;
unsigned char TEMP_IMAGE;

unsigned char TINTH=0xFF;
unsigned char TINTR=0;

void tmr0() interrupt 1 using 2
{
    TR0=0;                //stops the timer
    TH0=TINTH;            //to create the next timer interrupt
    TL0=TINTR;
    TR0=1;                //enables the timer

    TEMP_IMAGE = OPORT_IMAGE;
    PWM_CNTR = PWM_CNTR + 4;
    if (PWM_CNTR > PWM1) //compare PWM counter with PWM1 value
        TEMP_IMAGE = TEMP_IMAGE & 0xfc; //and stop motor if bigger
    if (PWM_CNTR > PWM2) //compare PWM counter with PWM2 value
        TEMP_IMAGE = TEMP_IMAGE & 0xf3; //and stop motor if bigger
    if (PWM_CNTR > PWM3) //compare PWM counter with PWM3 value
        TEMP_IMAGE = TEMP_IMAGE & 0xcf; //and stop motor if bigger
    if (PWM_CNTR > PWM4) //compare PWM counter with PWM4 value
        TEMP_IMAGE = TEMP_IMAGE & 0x3f; //and stop motor if bigger

    OPORT = TEMP_IMAGE;
}

void main(void)
{
    TMOD=0x21; //keep timer1 for comm. and timer0 in mode1
    TH0=TINTH; //timer0 interval
    TL0=TINTL;
    PWM_CNTR=0;
    ET0=1;
    EA=1;
    TR0=1 ; //start timer0

    PWM1 = 255;
    PWM2 = 128;
    PWM3 = 255;
    PWM4 = 32;
    while (1)
    {
        OPORT_IMAGE = OPORT;
    }
}

```

## נספח B – קריאת חיישני מרחק ברקע

כדי לשחרר את התוכנית הראשית מפנייה אל חיישני המרחק, נשתמש גם כאן בקוצב הזמן. בכל פסיקה של קוצב הזמן אנו דוגמים חיישן אחד ומכניסים אותו לתא בזכרון. ניתן כמובן לדגום את כל החיישנים, אבל אז תוכנית הפסיקה תהיה ארוכה מדי ותשפיע על מהירות התוכנית הראשית.

השיטה שבה אנו דוגמים כל פעם חיישן אחר מאפשרת לנו לוותר על ההמתנה לאות EOC (End Of Conversion).

בכל פסיקה אנו קוראים ערוץ אחד ויוצרים SOC לערוץ הבא. הזמן בין הפסיקות צריך להיות גדול מזמן ההמרה.

התוכנית הבאה דוגמת את שמונת ערוצי ה-ADC ומאחסנת אותם בשמונה תאי זכרון ADC0-ADC7.

התוכנית הראשית מוציאה לפורט FF00 את מצב ערוץ AI0 ולפורט 1 את מצב ערוץ AI7.

### תוכנית 3:

```

ADC0 EQU 40H
ADC1 EQU 41H
ADC2 EQU 42H
ADC3 EQU 43H
ADC4 EQU 44H
ADC5 EQU 45H
ADC6 EQU 46H
ADC7 EQU 47H
INDEX EQU 48H
PORT0 EQU 0F00H
PAI0 EQU 0FF08H
TINTR EQU 00H ;for 10ms timer interrupts
; ;increase this number for shorter intervals
;
ORG 200BH
LJMP TMR0
;
ORG 2100H
ADC: MOV TMOD,#01H ;mode1, C/T'=0,GATE=0
MOV TH0,#0FFH
MOV TL0,#TINTR
SETB EA ;enables all interrupts
SETB ET0 ;enables timer0 interrupt
SETB TR0 ;starts the timer
MOV INDEX,#0
MOV DPTR,#PAI0 ;creates SOC for ADC0
MOVX @DPTR,A
ADC2: MOV A,ADC0
MOV DPTR,#PORT0
MOVX @DPTR,A
MOV P1,ADC7
SJMP ADC2

```

```

;
TMR0:   CLR     TR0           ;stops the timer
        PUSH   ACC
        PUSH   DPH
        PUSH   DPL
        PUSH   PSW
;
        MOV    R0,#ADC0
        MOV    DPTR,#PAI0
        MOV    A,INDEX
        CJNE   A,#0,TMR2
        SJMP   TMR3
TMR2:   INC     DPTR
        INC     R0
        DJNZ   ACC,TMR2
TRM3:   MOVX   A,@DPTR       ;reads analog channel
        MOV    @R0,A        ;and saves it
        INC    DPTR
        INC    INDEX
        MOV    A,INDEX
        CJNE   A,#8,TMR4
        MOV    DPTR,#PAI0
        MOV    INDEX,#0
TMR4:   MOVX   @DPTR,A       ;creates SOC for next channel
        MOV    TH0,#0FFH    ;to generate the next timer interrupt
        MOV    TL0,#TINTR
        SETB   TR0
        POP    PSW
        POP    DPL
        POP    DPH
        POP    ACC
        RETI

```

```

#include "8052.h"
unsigned char at 0xff00 xdata OPORT;
unsigned char TINTH=0xFF;
unsigned char TINTL=0;
unsigned char TINTR=0;

unsigned char at 0xff08 xdata ADC0;
unsigned char at 0xff09 xdata ADC1;
unsigned char at 0xff0a xdata ADC2;
unsigned char at 0xff0b xdata ADC3;
unsigned char at 0xff0c xdata ADC4;
unsigned char at 0xff0d xdata ADC5;
unsigned char at 0xff0e xdata ADC6;
unsigned char at 0xff0f xdata ADC7;
unsigned char ANALOG_IN0;
unsigned char ANALOG_IN1;
unsigned char ANALOG_IN2;
unsigned char ANALOG_IN3;
unsigned char ANALOG_IN4;
unsigned char ANALOG_IN5;
unsigned char ANALOG_IN6;
unsigned char ANALOG_IN7;
unsigned char INDEX;

void tmr0() interrupt 1 using 2
{
    TR0=0;                //stops the timer
    TH0=TINTH;            //to create the next timer interrupt
    TL0=TINTL;
    TR0=1;                //enables the timer

    switch (INDEX)
    {
    case 0:
        ANALOG_IN0 = ADC0;    //read channel 0
        ADC1 = 0;            //create soc for channel 1
        INDEX = 1;
        break;

    case 1:
        ANALOG_IN1 = ADC1;    //read channel 1
        ADC2 = 0;            //create soc for channel 2
        INDEX = 2;
        break;

    case 2:
        ANALOG_IN2 = ADC2;    //read channel 2
        ADC3 = 0;            //create soc for channel 3
        INDEX = 3;
        break;

    case 3:
        ANALOG_IN3 = ADC3;    //read channel 3
        ADC4 = 0;            //create soc for channel 4
        INDEX = 4;
        break;
    }
}

```



```

case 4:
    ANALOG_IN4 = ADC4; //read channel 4
    ADC5 = 0;          //create soc for channel 5
    INDEX = 5;
    break;

case 5:
    ANALOG_IN5 = ADC5; //read channel 5
    ADC6 = 0;          //create soc for channel 6
    INDEX = 6;
    break;

case 6:
    ANALOG_IN6 = ADC6; //read channel 6
    ADC7 = 0;          //create soc for channel 7
    INDEX = 7;
    break;

case 7:
    ANALOG_IN7 = ADC7; //read channel 7
    ADC0 = 0;          //create soc for channel 0
    INDEX = 0;
    break;
}
}

void main()
{
    TMOD=0x21;          //keep timer1 for comm. and timer0 in mode1
    TH0=TINTH;         //timer0 interval
    TL0=TINTL;
    ET0=1;
    EA=1;
    TR0=1              ; //start timer0
    INDEX = 0;

    while (1)
    {
        OPORT = ANALOG_IN0;
        P1 = ANALOG_IN7;
    }
}

```

התוכנית הבאה מבצעת 8 דגימות של כל חיישן ומעדכנת את התא שלו עם הערך הנמוך מתוך 8 הדגימות. התוכנית משתמשת במונה נוסף בשם SAMPLE, המאופס בשלב האיתחול בתוכנית הראשית ובתא בשם TEMP לאיחסון זמני של הקריאה הנמוכה במשך הדגימות. אנו מכניסים לתא זה FF בכל פעם שמתחילים בסדרת דגימות חדשה.

#### תוכנית 4:

```

ADC0 EQU 40H
ADC1 EQU ADC0+1
ADC2 EQU ADC1+1
ADC3 EQU ADC2+1
ADC4 EQU ADC3+1
ADC5 EQU ADC4+1
ADC6 EQU ADC5+1
ADC7 EQU ADC6+1
INDEX EQU ADC7+1
SAMPLE EQU INDEX+1
TEMP EQU SAMPLE+1
;
PORT0 EQU 0FF00H
PAI0 EQU 0FF08H
TINTR EQU 00H ;for 10ms timer interrupts
; ;increase this number for shorter intervals
ORG 200BH
LJMP TMR0
;
ORG 2100H
ADC: MOV TMOD,#01H ;mode1, C/T'=0,GATE=0
MOV TH0,#0FFH
MOV TL0,#TINTR
SETB EA ;enables all interrupts
SETB ET0 ;enables timer0 interrupt
SETB TR0 ;starts the timer
MOV INDEX,#0
MOV SAMPLE,#0
MOV TEMP,#0FFH
MOV DPTR,#PAI0 ;creates SOC for ADC0
MOVX @DPTR,A
ADC2: MOV A,ADC0 ;read analog channel 0
MOV DPTR,#PORT0
MOVX @DPTR,A
MOV P1,ADC7
SJMP ADC2

```

```

;
TMR0:  CLR    TR0           ;stops the timer
        MOV    TH0,#0FFH   ;to generate the next timer interrupt
        MOV    TL0,#TINTR
        SETB   TR0
        PUSH   ACC
        PUSH   DPH
        PUSH   DPL
        PUSH   PSW
;
        MOV    R0,#ADC0
        MOV    DPTR,#PAI0
        MOV    A,INDEX
        CJNE   A,#0,TMR2
        SJMP   TMR3
TMR2:  INC    DPTR
        INC    R0
        DJNZ   ACC,TMR2
TMR3:  MOVX   A,@DPTR      ;read analog channel
        MOV    B,A
        CLR    C
        SUBB   A,TEMP      ;compare with TEMP
        JC     TMR4        ;if lower than
        MOV    TEMP,B      ;replace TEMP with the new number
TMR4:  INC    SAMPLE
        MOV    A,SAMPLE
        CJNE   A,#8,TMR5
        MOV    A,TEMP      ;after 8 samples
        MOV    @R0,A       ;save it in its cell
        MOV    SAMPLE,#0
        MOV    TEMP,#0FFH
        INC    DPTR        ;next channel
        INC    INDEX
        MOV    A,INDEX
        CJNE   A,#8,TMR5
        MOV    DPTR,#PAI0  ;next channel is AI0
        MOV    INDEX,#0
TMR5:  MOVX   @DPTR,A     ;creates SOC for next sample
        POP    PSW
        POP    DPL
        POP    DPH
        POP    ACC
        RETI

```

```

#include "8052.h"
unsigned char at 0xff00 xdata OPORT;
unsigned char TINTH=0xFF;
unsigned char TINTL=0;
unsigned char TINTR=0;

unsigned char at 0xff08 xdata ADC0;
unsigned char at 0xff09 xdata ADC1;
unsigned char at 0xff0a xdata ADC2;
unsigned char at 0xff0b xdata ADC3;
unsigned char at 0xff0c xdata ADC4;
unsigned char at 0xff0d xdata ADC5;
unsigned char at 0xff0e xdata ADC6;
unsigned char at 0xff0f xdata ADC7;
unsigned char ANALOG_IN0;
unsigned char ANALOG_IN1;
unsigned char ANALOG_IN2;
unsigned char ANALOG_IN3;
unsigned char ANALOG_IN4;
unsigned char ANALOG_IN5;
unsigned char ANALOG_IN6;
unsigned char ANALOG_IN7;
unsigned char INDEX;
unsigned char SAMPLE;
unsigned char MIN;
unsigned char TEMP

void tmr0() interrupt 1 using 2
{
    TR0=0;                //stops the timer
    TH0=TINTH;            //to create the next timer interrupt
    TL0=TINTL;
    TR0=1;                //enables the timer

    switch (INDEX)
    {
        case 0:
            TEMP = ADC0;
            if TEMP < MIN
                MIN = TEMP;
            SAMPLE++;
            if SAMPLE > 7
            {
                ANALOG_IN0 = MIN;
                MIN = 0xFF;
                SAMPLE = 0;
                ADC1 = 0;        //create soc for channel 1
                INDEX = 1;
            }
            break;

        case 1:
            TEMP = ADC1;
            if TEMP < MIN
                MIN = TEMP;
            SAMPLE++;
            if SAMPLE > 7
            {
                ANALOG_IN1 = MIN;
                MIN = 0xFF;
                SAMPLE = 0;
                ADC2 = 0;        //create soc for channel 2
                INDEX = 2;
            }
            break;
    }
}

```

```
case 2:
    TEMP = ADC2;
    if TEMP < MIN
        MIN = TEMP;
    SAMPLE++;
    if SAMPLE > 7
    {
        ANALOG_IN2 = MIN;
        MIN = 0xFF;
        SAMPLE = 0;
        ADC3 = 0;           //create soc for channel 3
        INDEX = 3;
    }
    break;

case 3:
    TEMP = ADC3;
    if TEMP < MIN
        MIN = TEMP;
    SAMPLE++;
    if SAMPLE > 7
    {
        ANALOG_IN3 = MIN;
        MIN = 0xFF;
        SAMPLE = 0;
        ADC4 = 0;           //create soc for channel 4
        INDEX = 4;
    }
    break;

case 4:
    TEMP = ADC4;
    if TEMP < MIN
        MIN = TEMP;
    SAMPLE++;
    if SAMPLE > 7
    {
        ANALOG_IN4 = MIN;
        MIN = 0xFF;
        SAMPLE = 0;
        ADC5 = 0;           //create soc for channel 5
        INDEX = 5;
    }
    break;

case 5:
    TEMP = ADC5;
    if TEMP < MIN
        MIN = TEMP;
    SAMPLE++;
    if SAMPLE > 7
    {
        ANALOG_IN5 = MIN;
        MIN = 0xFF;
        SAMPLE = 0;
        ADC6 = 0;           //create soc for channel 6
        INDEX = 6;
    }
    break;
```

```

case 6:
    TEMP = ADC6;
    if TEMP < MIN
        MIN = TEMP;
    SAMPLE++;
    if SAMPLE > 7
        {
            ANALOG_IN6 = MIN;
            MIN = 0xFF;
            SAMPLE = 0;
            ADC7 = 0;           //create soc for channel 7
            INDEX = 7;
        }
    break;
case 7:
    TEMP = ADC7;
    if TEMP < MIN
        MIN = TEMP;
    SAMPLE++;
    if SAMPLE > 7
        {
            ANALOG_IN7 = MIN;
            MIN = 0xFF;
            SAMPLE = 0;
            ADC0 = 0;           //create soc for channel 0
            INDEX = 0;
        }
    break;
}
}

void main()
{
    TMOD=0x21;           //keep timer1 for comm. and timer0 in mode1
    TH0=TINTH;           //timer0 interval
    TL0=TINTL;
    ET0=1;
    EA=1;
    TR0=1 ;              //start timer0
    INDEX = 0;
    SAMPLE = 0;
    MIN = 0xFF;

    while (1)
    {
        OPORT = ANALOG_IN0;
        P1 = ANALOG_IN7;
    }
}

```

## נספח C – בקרת תנועה עם חיישני מרחק

התוכנית הבאה מניעה את הרובוט לאורך קיר ימין בהתאם לאלגוריתם המתואר בפסקה 3. התוכנית מניחה שמנוע ימין מחובר ל- Q1,Q0 ומנוע שמאל מחובר ל- Q3,Q2. בחיבור כזה המספר 5 (0000101) מניע את הרובוט קדימה. הפניות מתבצעות על ידי שינוי המהירות של המנועים. התוכנית מניחה שחיישן קדמי מחובר לערוץ AI0 וחיישן ימין מחובר לערוץ AI1. המרחק הרצוי מקיר ימין מתקבל עם הקריאה 30H. מרחק העצירה מהקיר הקדמי מתקבל עם הקריאה 40H. שים לב: ככל שהמספר גדול יותר, החיישן קרוב יותר לקיר.

### תוכנית 5

PWM1	EQU	41H	;determines Q1,Q0 motor speed – right motor
PWM2	EQU	PWM1+1	;determines Q3,Q2 motor speed – left motor
PWM3	EQU	PWM2+1	;determines Q5,Q4 motor speed
PWM4	EQU	PWM3+1	;determines Q7,Q6 motor speed
CNTR	EQU	PWM4+1	
IMAGE	EQU	CNTR+1	
ADC0	EQU	IMAGE+1	
ADC1	EQU	ADC0+1	
ADC2	EQU	ADC1+1	
ADC3	EQU	ADC2+1	
ADC4	EQU	ADC3+1	
ADC5	EQU	ADC4+1	
ADC6	EQU	ADC5+1	
ADC7	EQU	ADC6+1	
INDEX	EQU	ADC7+1	
SAMPLE	EQU	INDEX+1	
TEMP	EQU	SAMPLE+1	
;			
PORT0	EQU	0FF00H	
PAI0	EQU	0FF08H	
TINTR	EQU	00H	;for 0.3ms timer interrupts
;			
	ORG	200BH	
	LJMP	TMR0	
;			
	ORG	2100H	
MAIN:	MOV	TMOD,#01H	;mode1, C/T'=0,GATE=0
	MOV	TH0,#0FFH	
	MOV	TL0,#TINTR	
	SETB	EA	;enables all interrupts
	SETB	ET0	;enables timer0 interrupt
	SETB	TR0	;starts the timer
	MOV	CNTR,#0	
	MOV	IMAGE,#05H	;Q3-Q2, Q1-Q0 motors ON
	MOV	INDEX,#0	
	MOV	SAMPLE,#0	
	MOV	TEMP,#0FFH	
	MOV	DPTR,#PAI0	;creates SOC for ADC0
	MOVX	@DPTR,A	
MAI2:	MOV	A,ADC1	;read right sensor
	CLR	C	
	SUBB	A,#30H	;compare with the required distance
	JC	MAI3	;if close or equal turn left, if far turn right
	MOV	PWM2,#0FFH	;far - high speed to motor2
	MOV	PWM1,#0A0H	;slow speed to motor1
	LJMP	MAI4	
MAI3:	MOV	PWM2,#0A0H	;close or equal - slow speed to motor2
	MOV	PWM1,#0FFH	;high speed to motor1
MAI4:	MOV	A,ADC0	;read front sensor
	CLR	C	
	SUBB	A,#40H	;compare with the required distance
	JNC	MAI2	;if far than do another loop
	MOV	IMAGE,#0	;if close than stop
MAI5:	LJMP	MAI5	;wait for reset

```

;
TMR0:   CLR     TR0           ;stop the timer
        MOV     TH0,#0FFH   ;update it
        MOV     TL0,#TINTR
        SETB    TR0         ;enable the timer for the next timer interrupt
        PUSH   ACC
        PUSH   DPH
        PUSH   DPL
        PUSH   PSW
;
        MOV     DPTR,#PORT0
TMR1:   INC     CNTR
        INC     CNTR
        INC     CNTR
        INC     CNTR
        MOV     A,PWM1
        CLR     C
        SUBB   A,CNTR       ;compare PWM1 with CNTR
        JNC    TMR2       ;jump if CNTR is smaller than PWM1
        MOV     A,IMAG1     ;CNTR is bigger than PWM1
        CLR     ACC.0       ;so stop the Q1,Q0 motor
        CLR     ACC.1
        SJMP   TMR3
TMR2:   MOV     A,IMAG1     ;do not change image
TMR3:   MOV     IMAG2,A     ;save image in a temporary cell
;
        MOV     A,PWM2
        CLR     C
        SUBB   A,CNTR       ;compare PWM2 with CNTR
        JNC    TMR4       ;jump if CNTR is smaller than PWM2
        MOV     A,IMAG2     ;CNTR is bigger than PWM2
        CLR     ACC.2       ;so stop the Q3,Q2 motor
        CLR     ACC.3
        MOV     IMAG2,A     ;update imag2
;
TMR4:   MOV     A,PWM3
        CLR     C
        SUBB   A,CNTR       ;compare PWM3 with CNTR
        JNC    TMR5       ;jump if CNTR is smaller than PWM3
        MOV     A,IMAG2     ;CNTR is bigger than PWM3
        CLR     ACC.4       ;so stop the Q5,Q4 motor
        CLR     ACC.5
        MOV     IMAG2,A     ;update imag2
;
TMR5:   MOV     A,PWM4
        CLR     C
        SUBB   A,CNTR       ;compare PWM4 with CNTR
        JNC    TMR6       ;jump if CNTR is smaller than PWM4
        MOV     A,IMAG2     ;CNTR is bigger than PWM4
        CLR     ACC.6       ;so stop the Q7,Q6 motor
        CLR     ACC.7
        MOV     IMAG2,A     ;update imag2
TMR6:   MOV     A,IMAG2
        MOVX   @DPTR,A     ;output imag2
;
        MOV     R0,#ADC0
        MOV     DPTR,#PA10
        MOV     A,INDEX
        CJNE   A,#0,TMR2
        SJMP   TMR8
TMR7:   INC     DPTR       ;adapt DPTR
        INC     R0         ;and R0 to INDEX
        DJNZ   ACC,TMR7

```



```

TMR8:  MOVX  A,@DPTR      ;read analog channel
        MOV   B,A
        CLR   C
        SUBB  A,TEMP      ;compare with TEMP
        JC    TMR9        ;if lower than
        MOV   TEMP,B      ;replace TEMP with the new number
TMR9:  INC    SAMPLE
        MOV   A,SAMPLE
        CJNE  A,#8,TMR10
        MOV   A,TEMP      ;after 8 samples
        MOV   @R0,A       ;save it in its cell
        MOV   SAMPLE,#0
        MOV   TEMP,#0FFH
        INC   DPTR        ;next channel
        INC   INDEX
        MOV   A,INDEX
        CJNE  A,#8,TMR10
        MOV   DPTR,#PAI0  ;next channel is AI0
        MOV   INDEX,#0
TMR10: MOVX  @DPTR,A     ;creates SOC for next sample
;
        POP   PSW
        POP   DPL
        POP   DPH
        POP   ACC
        RETI

```

בדקו תוכנית זו. ייתכן ותמצאו טעויות הקלדה.  
לתוכנית זו אין גרסה בשפת C.

## נספח D – תגובה לחיישן קול והמתנה לצליל

### תוכנית 6

תוכנית זו מבצעת המתנה לפולס ראשון. לאחר מכן מבצעת FFFF לולאות השהייה תוך כדי ספירת הפולסים המתקבלים. מספר הפולסים שהתקבלו נאגר בצובר A.

```

sndb equ    p1.0    ;assuming that the sensor output
                  ;is connected to p1.0
plss equ    40      ;minimum number of pulses to be
                  ;counted in an interval of 0.1 second
;
    org     2100h
snd:  mov    r7,#0ffh
      mov    r6,#0ffh    ;to create 0.1 second delay
      mov    a,#0        ;pulse counter
      clr    f0          ;checking flag
      jnb   sndb,snd     ;wait for the first pulse
;
snd2: jb     sndb,snd3    ;wait for pulse dropping
      jnb   f0,snd4      ;don't count if flag=0
      inc   a            ;count if f0=1 and sndb=0
      clr   f0
      ljmp  snd4
snd3: setb   f0          ;when sndb=1, raise flag
snd4: djnz   r6,snd2
      mov   r6,#0ffh
      djnz  r7,snd2
      clr   c
      subb  a,#plss
      jc    snd
;
;   the program continue from here
;
    end

```

**תוכנית 7**

תוכנית זו קובעת את TIMER0 לשמש כמונה, מתוך הנחה שיציאת חיישן הקול מחוברת לכניסת הטיימר. התוכנית מאפסת את המונה ולאחר מכן מבצעת לולאת השהייה. בסיום לולאת ההשהייה בודקת את המספר שנמצא במונה. אם המספר גדול מהערך המבוקש, סימן שהתקבל הצליל הדרוש. היא מבצעת תהליך זה שוב לשם בטחון. אם המספר קטן מהערך הדרוש, היא חוזרת על התהליך.

```

    ORG    2000H
    LJMP   MAIN
;
    ORG    2100H
MAIN:  MOV    A,#255
      MOV    P1,A          ; TO INDICATE STATUS
;
;
      CLR    ET0          ; DISABLE TIMER0 INTERRUPT
      MOV    TMOD,#25H    ; TIMER0 16 BIT COUNTER FROM T0 PIN
                          ; TIMER1 IN MODE 2 TO ENABLE TO RETURN
                          ; TO THE DEBUGGER
;
WAIT_BUZ:
      CLR    TR0
      CLR    TF0
      MOV    TH0,#0FFH
      MOV    TL0,#255-100 ; TO COUNT 155 PULSE UNTIL OVEFLOW
      SETB   TR0          ; ENABLE COUNTER
      LCALL  DLY_80MS
      JNB    TF0,WAIT_BUZ ; BUZZER WILL CAUSE COUNTER SET T1 FLAG
;
      CLR    TR0          ;ONCE AGAIN
      CLR    TF0
      MOV    TH0,#0FFH
      MOV    TL0,#255-100 ; TO COUNT 155 PULSE UNTIL OVEFLOW
      SETB   TR0          ; ENABLE COUNTER
      LCALL  DLY_80MS
      JNB    TF0,WAIT_BUZ ; BUZZER WILL CAUSE COUNTER SET T1 FLAG
;
      MOV    A,#55H       ;TO INDICATE RECEIVING THE PITCH
      MOV    P1,A
      CLR    TR0
      LCALL  103H         ;RETURN TO THE DEBUGGER

DLY_80MS:
      MOV    R7,#144
DLY2:  MOV    R6,#0FFH
DLY3:  DJNZ   R6,DLY3
      DJNZ   R7,DLY2
      RET

```

```
#include "8052.h"

int I;

void main(void)
{
    P1 = 0xFF;
    ET0 = 0;
    TMOD=0x25;
    do
    {
        TR0 = 0;
        TF0 = 0;
        TH0 = 0xFF;
        TL0 = 255-100;
        TR0 = 1;
        for (I = 0; I < 1000; I++);
    }
    while (TF0 = 0);
    P1 = 0x55;
    TR0 = 0;

    while (1)
    {
    }
}
```

### "סוף מעשה במחשבה תחילה"

הרצון הטבעי של כל אחד הוא ליצור, לפעול, לבצע. התכנון לפרטיו נתפס כחלק משעמם וכדרישה מעצבנת של המנחה או המורה. הרבה פעמים אנו מתחילים בבנייה של מערכת מורכבת על פי איזה שהוא רעיון ראשוני וסקיצה בסיסית ביותר. לכל אלה מובטח באחריות מלאה, שהפתוח יהיה ארוך, מלווה בשינויים רבים ומפחי נפש.

השקעה בתכנון מפורט לפני תחילת הבצוע, נראית דרך ארוכה אבל זו תמיד הדרך המהירה ביותר. עבור אלה שמוכנים ללכת בדרך זו נרשמו את ההערות הבאות.

1. רשמו לכם אפיון של המוצר ושרטוטים עבורכם ועבור חבריכם. עצם הרישום גורם לחשיבה מסודרת וניתוח מסודר של המוצר. גם דיון מול מסמכים כתובים הופך לדיון מסודר, ברור וממצה.
2. עבדו בגישה של 'מהמערכת אל המרכיבים'. אם לא תתכננו נכון, תמצאו שחסר לכם מקום לפריט מסוים, או שהמערכת לא יציבה וכדומה.
3. רשמו את האפיון בעזרת מעבד תמלילים על מנת שתוכלו לערוך אותו ולשפרו כל הזמן. הנכם יכולים להכין עותקים נוספים לחבריכם לצוות.
4. התרגלו להמיר הערות שרשמתם על הדפים בכתב יד לחלק מהטקסט המודפס.
5. פרקו את האפיון להגדרת המוצר, מטרות ובצועים, מרכיבים, מאפייני מבנה, מאפייני חומרה, מאפייני תוכנה.
6. השתמשו במשפטים קצרים ממוספרים בצורה מדורגת. לדוגמא:

1. שם המוצר

2. תיאור המוצר

3. מטרות ובצועים

3.1. ...

3.2. ...

4. מרכיבים

4.1. ...

4.2. ...

5. מבנה

5.1. ...

5.2. ...

6. חומרה

6.1. ...

6.2. ...

7. תוכנה

7.1. ...

7. שרטטו דיאגרמת מלבנים של המערכת ומרכיביה.
8. שרטטו דיאגרמת מלבנים של התוכנה. תכנון מערכת משובצת מיקרו-מחשב כרוך בפתרונות המשלבים תוכנה וחומרה.
9. שינויים קשים הרבה יותר לבצוע מאשר בנייה שיטתית ומסודרת. ככל שתשקיעו זמן וחשיבה באפיון, פחות תדרשו לשינויים במהלך הביצוע.
10. אל תתחילו לבנות או לתכנת לפני שברורה לכם כל תמונת המוצר ומרכיביה. הכוונה לאפיון ולא לחלקים המושלמים, אחרת לא תתחילו לעבוד לעולם.
11. בעבודה בצוות מקבל כל אחד משימה. ככל שהאפיון של המרכיבים יהיה ברור יותר, שלב האינטגרציה יהיה פשוט יותר.
12. נהלו תיק מסודר שבו כל דפי העזר, הטיטות (אל תזרקו טיטות כי הן תזכרנה לכם את אופן החשיבה שלכם), הערות וסכומי דיון, דפי נתונים, פרוספקטים, רשימות תוכנה, מחירים וכדומה. תיק זה ילווה אתכם עד לסיום הפרוייקט.
13. חלקו תיק זה למדורים בצורה מסודרת.

1. תכננו ושרטטנו כל חלק מכני.
2. רשמו את רשימת הרכיבים, בדקו את הימצאותם לפני שתתחילו בבנייה. דאגו שלא תישארו עם מברג ביד ללא בורג.
3. חפשו והשתמשו באמצעים קיימים. הימנעו ככל האפשר 'מהמצאת הגלגל'. למרות זאת אל תניחו שהכל הומצא כבר. ייתכן שתעלו על רעיון טוב וחדשני. חפשו רכיבי הנעה, תמסורת.
4. זכרו – פשוט זה אמין, יפה וחזק.
5. השתמשו בכלים מתאימים ונכונים.
6. **בטיחות מעל הכל.** הקפידו על כללי הבטיחות.
7. גם תכנון קפדני לא פטור מטעויות. לעתים גם נוצר צורך או עולים רעיונות לשינוי או לשיפור. לכן חשבו כל הזמן שהנכם הולכים לייצר סדרת ייצור של המוצר ולא מוצר בודד, גם אם הנכם מתכננים לייצר מוצר אחד בלבד.
8. לכל חלק הכינו שרטוט מסודר.
9. הכינו שבלונת חיתוך לחומר הייצור.
10. הכינו שבלונת קידוחים לכל סדרת קידוחים עם קדחים קטנים (2-1 מ"מ) במקומות המתאימים. ליד כל קדח רשמו את הקוטר של הקדח שיהיה בחומר האמיתי. לפני כל קידוח בחומר האמיתי, קדחו חור קטן בעזרת השבלונה.
11. בנו דגם מחומר רך לפני השימוש בחומר האמיתי, כדי לראות כיצד יראה המוצר ולחזות את הקשיים הצפויים בתהליך הייצור. קרטון ביצוע הוא אמצעי עזר מצוין לשבלונות ולדגמים.
12. אל תסמכו על הזיכרון. רשמו כל הערה, רעיון, מידות על השרטוטים והשבלונות. "**ההפך מלזכור זה לרשום**". ככל שתשרמו יותר ותנסו לזכור פחות, תעשו פחות טעויות.
13. האמינו לי. עבודה עם דגמים ושבלונות מקצרת את זמן הייצור. היא גם מאפשרת לייצר חלק שנפגם במהירות רבה. שלא לדבר על ייצור של מוצר נוסף.
14. אחרי שמדדתם וסימנתם, שתו כוס מים ומדדו שוב בעיניים בודקות. אין דבר מתסכל יותר מזה שקדח אחד או כיפוף אחד מתוך כל העבודה לא במקום או לא מדויק.
15. חומר נוח למבנה הוא PVC מוקצף. חומר זה הוא חומר קשיח למדי, אבל ניתן לחיתוך וכיפוף בקלות.
16. לאחר בניית הדגם, סמנו בעזרתו את קווי החיתוך על פלטת ה-PVC. חתכו בזהירות את החומר.
17. סמנו את קווי הכיפוף בחומר.
18. תפסו בעזרת מלחציים את החומר לאורך קו כיפוף. אם המלחציים קצרים מדי, השתמשו בסרגלי מתכת או בפרופילי אלומיניום כדי להתאימם לקו הכיפוף.
19. חממו את החומר משני צדדיו לאורך קו הכיפוף. השתמשו במייבש שיער חשמלי (FAN) או בגוף חימום.
20. ברגע שתחושו שהחומר רך וניתן לכיפוף, כופפו אותו בעזרת פלטת עץ כלשהי. פלטת העץ תשמור על ידיכם וגם תיצור כיפוף ישר ואחיד. המשיכו ללחוץ על החומר עד להתקררותו.
21. זכרו שנוצר רדיוס כיפוף מסוים בקו הכיפוף.

חומרה

1. בדקו שהשרטוט האלקטרוני של המוצר ברור לכם. רצוי כי האלקטרונאים שבחבורה יבינו את אופן פעולת המעגל האלקטרוני לפרטיו, על מנת שיוכלו לאתר תקלות במעגל עצמו וברכיביו.
2. לאנשי המכניקה והתוכנה, אין חובה להבין את המעגל האלקטרוני, אבל חשוב מאוד להבין את מרכיביו, אופן הפנייה אליהם, הנתונים העוברים דרכם, והפרמטרים והביצועים שלהם.
3. חשוב להבין מה תפקיד כל נקודת התחברות לרכיבים החיצוניים ומגבלותיה (מתח, זרם וכדומה).
4. לפני בניית המעגל הכינו רשימת כל רכיבי המעגל ובדקו התאמתם לשרטוט ולמיקומם במעגל. בדקו שיש ברשותכם את כל הרכיבים הדרושים.
5. השתמשו במלחם תקין, בכך למלחם ובספוג רטוב לניקוי המלחם בזמן ההלחמה.
6. דאגו ששולחן העבודה שלכם נקי ומסודר ושיש לכם מספיק מרחב.
7. השתמשו בבדיל לאלקטרוניקה המכיל פלקס ניקוי בתוכו. בדיל זה ישפר את איכות ההלחמה שלכם. הפלקס מושך אל מחוץ לבדיל לכלוך ושמונויות שנמצאים בנקודת ההלחמה.

תוכנה

1. סיימו את אפיון התוכנה והגדרות הביצועים לפני שתתחילו בכתיבת התוכנה. תוכנה הכתובה בטלאים, תגרום לבאגים ותגרום לכם לבזבוז זמן נוראי.
2. בנו את התוכנית הראשית כך שתשתמש בתוכניות המשנה.
3. תוכנית מסודרת מתחילה ברוטינות הבסיסיות ורק לאחר מכן התוכנית הראשית. אנו שואפים שכל רוטינה תקרא ותשתמש ברוטינות שמעליה (שהוגדרו, נכתבו ונבדקו כבר).
4. ניתן לכתוב את התוכנית הראשית בחלקים כך שהיא תשמש לבדיקת הרוטינות הבסיסיות.
5. רשמו לעצמכם את סדר הפיתוח והבדיקות. בצורה כזו תוכלו להתקרב לעמידה בלוח הזמנים של פיתוח התוכנה.
6. הגדירו את רוטינות העזר הבסיסיות שבהם תשתמש התוכנית הראשית (קדימה, אחורה, ימינה, שמאלה, חיפוש וכדומה). רוטינות אלה נקראות BIOS – Basic Input Output Subroutines.
7. לכל רוטינה רשמו אפיון קצר – **שם, בצועים, משתנים שבהם היא משתמשת, ערכים שהיא מחזירה, רוטינות עזר שבהן היא משתמשת.**
8. תיעוד התוכנית והרוטינות לא מיועד למורה, הוא מיועד לכם.
9. ככל שתסמכו פחות על הזיכרון שלכם, תרוצו מהר יותר בפתוח.
10. בנו כל רוטינה בנפרד ובדקו אותה בנפרד. לעולם, אל תכתבו תוכנית שלמה ותבדקו אותה כמקשה אחת.
11. לכל רוטינה הכינו נוהל בדיקה. זכרו, כל באג בתוכנה מעלה את זמן איתור התקלות בצורה מעריכית (בחזקה) ולא בצורה ליניארית. נוהל בדיקה גורם לנו לחשוב על כל רוטינה כעל תוכנית נפרדת ולנפות את הבאגים מתוכה בצורה אמינה.
12. לעתים בהמשך הפיתוח, אנו נדרשים לשנות רוטינה קודמת. ביצוע שינוי, גם מה שנראה פשוט למדי, יוצר באגים חדשים. אם ברשותנו נוהל בדיקה כתוב, אין אנו צריכים לסמוך על הזיכרון. בדיקת הרוטינה לאחר השינוי תהיה אמינה ומהירה.

## תוכנה SOFTWARE DSM-2095

תוכנת המוניטור כוללת שגרות שונות המשמשות אותן. שגרות אלה יכולות לשמש גם את המשתמש בכרטיס כשגרות עזר. כדי להקל על המשתמש ועל התייעוד, נכתבו בתחילת תוכנת ה-Debugger טבלאות הפנייה לשגרות השונות. הפניה לשגרת עזר נעשית על-ידי פניה לכתובת בטבלת ההפנייה. שם קיימת פקודת דילוג לשגרה הדרושה. בצורה כזו, גם אם משתנה מיקום השגרה בגרסאות מתקדמות, כתובת הכניסה לשגרה נשארת קבועה.

התקשורת של הכרטיס DSM-2095 עם המחשב נעשית בעזרת רכיב התקשורת (נקרא UART) שנמצא בתוך המעבד. המוניטור של הכרטיס קובע אותו באופן (MODE) העבודה הדרוש. הרכיב משתמש ב-TIMER1 ב-MODE2.

כאשר רוצים להשתמש ברוטינות הבאות, יש לדאוג שהתוכנית אינה משנה את אתם TIMER1 MODE. TMOD צריך לקבל את המספר 2X (ראו כדוגמא את תוכנית 6).

כל שגרה היא שגרת משנה המסתיימת בפקודה RET (חזור מתוכנית משנה).

### 4.1 שגרות עזר של ה-On Line Debugger:

#### 4.4.1 WARM (כניסה "חמה") לתוכנת ה-Debugger – 0103:

קריאה לשגרה זו כאשר מריצים תוכניות תחת ה-Debugger, גורמת לחזרת התוכנית ל-Debugger המדפיס נקודה במסך וממתין לפקודה מהמשתמש. קריאה זו רושמים בסוף התוכנית והיא משמשת מעין משפט END. שגרה זו מאחסנת את האוגרים השונים. עם החזרה ל-Debugger אפשר לתת את הפקודה <ENTER> R ואז יוצגו האוגרים במסך.

קריאה לשגרה זו (LCALL 0103), יכולה לשמש כנקודת שבירה בתוכנית, כאשר אין רוצים להשתמש בריצה עם נקודת שבירה, המאיטה את עבודת ה-CPU.

#### 4.4.2 CCI - קליטת תו מלוח המקשים – 0106:

שגרה זו ממתינה ללחיצה על מקש בלוח המקשים. ברגע שמקש נלחץ, מתבצעת חזרה לתוכנית הראשית. צופן ASCII של המקש שנלחץ יופיע בצובר. השגרה אינה משדרת הד (Echo) למסוף. התוכנית משנה את ה-ACC בלבד.

#### 4.4.3 CCO - הדפסת תו במסך – 0109:

שגרה זו מדפיסה בתצוגה תו, שצופן ASCII שלו נמצא ב-ACC. לאחר ההדפסה מתבצעת חזרה לתוכנית הראשית בתנאי שמקש S^ לא נלחץ. אם נלחץ מקש S^, ממתינה התוכנית ללחיצה על @^.

נתוני הכניסה של השגרה הם ה-ACC.

התוכנית אינה משנה כל אוגר.



#### **4.4.4 INKEY – קליטת תו מהמסוף ללא המתנה – 010C:**

שגרה זו בודקת האם התקבל תו מהמסוף. אם לא התקבל תו מתבצעת חזרה לתוכנית הראשית, כאשר בצובר המספר FF. אם התקבל תו, יופיע תו זה ב-ACC. התוכנית משנה את ה-ACC בלבד.

#### **4.4.5 TEXT – הדפסת טקסט במסוף – 010F:**

שגרה זו מדפיסה במסוף טקסט בצופן ASCII הנמצא באיזור ה-Program. שורת הטקסט חייבת להסתיים ב-byte 00. כתובת תחילת הטקסט צריכה להימצא ב-DPTR לפני הקריאה לשגרה. התוכנית משנה את ה-ACC וה-DPTR. התוכנית קוראת ל-CCO.

#### **4.4.6 INLINE – קליטת שורה מהמסוף – 0112:**

שגרה זו קולטת שורה של תוים מלוח המקשים. כאשר נלחץ מקש <ENTER>, מתבצעת חזרה לתוכנית הראשית. אוסף התוים שהוקשו ימצא ב-Buffer, הנמצא ב-RAM החיצוני החל מכתובת BFE0H. השגרה מגיבה גם למקש <BS> (Back Space).

התוכנית משנה את האוגרים ACC, DPTR.

התוכנית קוראת לשגרות CCI, CCO.

#### **4.4.7 BIN2ASC – המרת מספר בינארי ל-2 ספרות בצופן ASCII – 0115:**

שגרה זו מפרקת מספר בינארי, הנמצא בצובר, לשתי ספרות ומתרגמת ל-ASCII שתי ספרות אלה. הספרות המתורגמות ל-ASCII תימצאנה באוגרים R2, R3. שגרה זו מאפשרת שידור של מספר בינארי למסוף. נתון הכניסה של השגרה הוא ה-ACC - המספר להמרה.

נתוני היציאה הם R2 ו-R3. R2 יכול את הספרה המשמעותית יותר ו-R3 את הספרה הפחות משמעותית. התוכנית משנה את האוגרים ACC, R2, R3, R4.

#### **4.4.8 ASC2BIN – המרת שתי ספרות בצופן ASCII למספר בינארי – 0118:**

שגרה זו ממירה לבינארי שתי ספרות, הכתובות בצופן ASCII, ומאחדת אותן למספר בינארי אחד. שתי הספרות להמרה, צריכות להימצא באוגרים R2 ו-R3. תוצאת ההמרה תימצא בצובר.

נתוני הכניסה של השגרה הם R2 ו-R3. R2 צריך להכיל את הספרה המשמעותית יותר ו-R3 את הספרה הפחות משמעותית.

נתוני היציאה של השגרה הוא ה-ACC, המכיל את תוצאת ההמרה. התוכנית משנה את הצובר בלבד.

#### **4.4.9 AOUT – הדפסת תוכן הצובר במסוף – 011B:**

שגרה זו ממירה לשתי ספרות בצופן ASCII את המספר הנמצא בצובר ומשדרת אותן אל המסוף בזה אחר זה. קודם את הספרה המשמעותית יותר.

נתון הכניסה של השגרה הוא הצובר, המכיל את המספר לשידור.

התוכנית משנה את האוגרים ACC, R2, R3, R4.

התוכנית קוראת ל-BIN2ASC ול-CCO.