**המכללה הטכנולוגית של חיל אויר**

**באר שבע**

**פרויקט גמר הנדסאים**

**מגמת אלקטרוניקה**

**התמחות מערכות אלקטרוניקה**

# נושא: רובוט מונחה רשת נוירונים

# מגישה: צוקרוב אנה

# מנחה: זוניס ישראל

הצעת פרויקט:

הצהרת סטודנט:

**אני צוקרוב אנה, מספר תעודת זהות 320609241.**

החתום מטה, מצהיר בזאת שכל עבודת הפרויקט המוגשת בחוברת זו הנה פרי עבודתי בלבד, על בסיס הנחיתו של המנחה ותוך הסתמכות על מקורות הידע והמידע האחרים המצוינים בביבליוגרפיה המובאת בסיום ספר זה.

אני מודע/ת לאחריות שהנני מקבל על עצמי ע"י חתימתי על הצהרה זו שכל הנאמר בה הנו אמת ורק אמת.

**חתימת הסטודנט: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

אישור המנחה:

הריני מאשר חוברת זו להערכה.

**חתימת המנחה: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

שלמי תודה:

ברצוני להודות לכל אלו שמטוב ליבם עזרו לי והיו מוכנים לשבת שעות על גבי שעות על מנת ופרויקט זה יצליח.

ראשית כל אני רוצה להודות להורי אשר תמחו בי לא רק מבחינה מנטלית אלא גם מבחינה כספית. כמו כן אני רוצה להודות לחבר שלי שדחף אותי להתקדם בשעות שרציתי לעזוב הכל, מלבד זאת אני רוצה להודות לחברי לכיתה שעזרו לי לפתור בעיות שונות שבאו על הפרק. וכמובן אני רוצה להודות לצוות החינוכי שגם כן עזרו ותרמו רבות להצלחת הפרויקט.

תקציר:

בפרויקט זה פותח רובוט מונחה רשת נוירונים, רובוט זה בא בהשראת ניסוי פבלוב בו הוכיח כי ישנם שני סוגי אינסטינקטים: מולדים ומותנים.

בניסויו פבלוב יצר התניה אצל כלב בין האינסטינקט המולד שלו לאכול לבין צלצול של פעמון.

את מוח המערכת מהווה מעבד ARM Cortex M-3 LPC1769 שהוא בעצם מממש את פעולת רשת הנוירונים. בפרויקט אצור התניה בין שלט RF לבין אור ה-LED, רשת הנוירונים נמצאת בהתחלה במצב למידה. לאחר מספר תרגולים של הרשת, הרשת מגדילה את משקליה עד לערך מסוים ומחשבת את השגיאה של רשת הנוירונים עד אשר היא אינה יורדת מתחת לשגיאה המינימלית של המערכת. לאחר מכן הרשת בא לידי מצב עבודה בו הרובוט יכול להתקדם אחר האור ללא מדריך (שלט).

בפרויקט זה תפקידה של תצוגת ה-LCD הוא להוות בקרה לפעולתו התקינה של הרובוט, כלומר לבדוק אם כיוון ההתקדמות של הרובוט תואם למוצג על המסך.

**תוכן עניינים**

פרק 1: מבוא ................................................................................ 11

מבוא

פרק 2: מפרט טכני ........................................................................ 13

מפרט טכני

פרק 3: סכמה מלבנית ................................................................... 15

סכמה מלבנית

הסבר הסכמה המלבנית

הסבר יחידות הסכמה המלבנית

פרק 4: סכמה חשמלית ................................................................. 19

סכמה חשמלית

פרק 5: תוכנה .............................................................................. 33

פרק 6:חישובים ומדידות.............................................................. 54

פרק 7:זיווד.................................................................................. 57

פרק 8:סיכום ומסקנות.................................................................. 62

פרק 9: Reversing engineering .................................................. 64

פרק 10:ביבליוגרפיה..................................................................... 66

פרק 11: נספחים ......................................................................... 68

נספח א'

נספח ב'

נספח ג'

נספח ד'

נספח ה'

נספח ו'

נספח ז'

נספח ח'

נספח ט'

נספח י'

**פרק 1:**

**מבוא**

1.1 מבוא:

בפרויקט זה פותח רובוט מונחה רשת נוירונים, רובוט זה בא בהשראת ניסוי פבלוב בו הוכיח כי ישנם שני סוגי אינסטינקטים: מולדים ומותנים. כמו בניסוי פבלוב שיצר התניה אצל כלב בין האינסטינקט המולד שלו לאכול לבין צלצול של פעמון.

בפרויקט אני מדמה לאינסטינקט המולד את השלט, הרובוט יקבל פקודות מהמורה הפקודות יתבצעו באופן ישיר. כאינסטינקט המותנה אותו אני יוצרת עם עזרת השלט יהיה יצירת התניה עם האור אותו הרובוט ילמד בזמן פעולה הדדית בין המנחה לתלמיד.

את מוח המערכת מהווה מעבד ARM Cortex M-3 LPC1769 בעזרת תוכנה בשפת C, הוא בעצם מממש את פעולת רשת הנוירונים. בפרויקט אצור התניה בין שלט RF לבין אור ה-LED שיזוהה ע"י פוטוטרנזיסטורים, רשת הנוירונים נמצאת בהתחלה במצב למידה. לאחר מספר תרגולים של הרשת, הרשת מגדילה את משקליה עד לערך מסוים ומחשבת את השגיאה של רשת הנוירונים עד אשר היא אינה יורדת מתחת לשגיאה המינימלית של המערכת. לאחר מכן הרשת בא לידי מצב עבודה בו הרובוט יכול להתקדם אחר האור ללא מדריך (שלט).

בנשך זמן הלמידה ופעולת התוכנה נעשה שימוש במסך LCD ומספר LED צבעוניים לשם תצוגת שלבי הלמידה והודעה על פעילות הרובוט.

בסיס לרובוט משמשת פלטפורמה מכנית בעלת שני מנועי DC עם רדוקטורים ושני מקורות מתח DC (8.4v ו-7.2v) נפרדים (לביטול השפעת הרעשים של פעילות המנועים על פעילות המערכת).

**פרק 2:**

**מפרט טכני**

2.1 מפרט טכני:

* רובוט מונחה רשת נוירונים הממומשת בתוכנה באמצעות מעבד מסוג ARM Cortex-M3.
* הרובוט נשלט ממשדר RT4 או תחליפי reversing engineering של TX-2 למימוש השלט.
* לשם קליטת השדר נעשה שימוש במקלט RR3 או תחליפי reversing engineering של מקלט RX-2.
* לרובוט יכולת למידה.
* לצורך ניסוים ברשת נוירונים משתמשים בסימולציה ב-MATLAB.
* תיכתב תוכנה בשפת C להפעלת הרובוט.
* פלטפורמה מכנית מבוססת שני מנועי DC עם רדוקטורים להפחתת סיבובי מנוע עד 40 סל"ד.
* שני מקורות מתח DC נפרדים 8.4v למנועים (מנוע DC 6v ו-2v נופלים על גשר H), ומקור מתח 7.2v לפעולת הרכיבים האלקטרוניים. מקורות המתח מורכבים מסוללות AA.
* לרובוט מורכבת לקליטת אור המבוססת על ארבע פוטוטרנזיסטורי (שלושה מקדימה ואחד מאחור) TEPT5600.

**פרק 3:**

**סכמה מלבנית**

3.1 סכמה מלבנית:

LCD

Phototransistor

LED

RF

receiver

RF

transmitter

Buttery

Power supply

Drivers

Motors

מחשב

PC

Microcontroller

3.1.1 סבר הסכמה המלבנית:

המעבד מהווה את לב המערכת, הוא שולט במנועים לפי האותות המשודרים מהשלט. השלט (RF transmitter) משמש בתור מנחה ללימוד רשת הנוירונים אותה מממש המעבד. אור ה-LED נקלט ע"י הפוטוטרנזיסטורים אשר יוצר אותות כניסה לרשת הנוירונים.

3.2 הסבר יחידות הסכימה המלבנית:

3.2.1 Microcontroller:

המעבד ARM Cortex M-3 LPC1769 משמש בעצם כ"מוח" המערכת, התוכנית אותה אני בונה נשמרת בו. הוא עורך את כל החישובים ומבצעה את הלימוד של המערכת.

3.2.2 מחשב PC:

המחשב משמש לכתיבת התוכנית LPCXpresso v4.2.0 וצריבת ל-ARM. משגם הוא מממש הדמיה של רשת הנוירונים המלכותית ע"י תוכנית הנכתבת ב-MATLAB.

3.2.3 LCD:

משמש לבקרת פעולת התוכנית בזמן אמת. על המסך יופיע הכיוון המיועד לנסיעה משמשמש לבקרה והצגת מספר שלבי הלמידה. במידה וכיוון התקדמות הרובוט הינו בכיוון הרשום פעולתו תקינה.

3.2.4 LED:

משמש כגירוי המעורר את ה"אינסטינקט הנלמד" של רשת הנוירונים ברובוט.

3.2.5 RF transmitter:

משמש לשידור כיוון ההתקדמות של הרובוט, הינו מייצג את ה"אינסטינקט המולד" של הרובוט. משמש כמדריך ללימוד הרובוט.

3.2.6 RF receiver:

משמש לקליטת שידור ה-transmitter והעברת הפקודות למיקרו מעבד.

3.2.7 Phototransistor:

משמש לקליטת אור לעירור רשת הנוירונים.

3.2.8 Driver:

מפני שהמעבד אינו יכול לספק זרם הנדרש לפעולת המנועים אני משתמשת בגשר H.

3.2.9 Motors:

הינם מהווים את הפלטפורמה המכנית, נשתמש במנועי DC עם רדוקטור מובנה. ישנם שני מנועים אשר כל אחד מהם שולט על גלגל.

3.2.10 Buttery Power Supply:

מכיוון המערכת הינה מערכת ניידת יש להשתמש במקורות מתח המאפשרים זאת, סוללות AA, ישנם שני מקורות מתח שונים למנועים ולרכיבים אלקטרונים.

**פרק 4:**

**סכמה חשמלית**

4.1 סכמה חשמלית:





4.2 סכמה חשמלית שלט:



4.3 הסבר סכמה חשמלית מכונית:

4.3.1 מעגל הספקת המתח :

בבסיס מעגל זה עומד מייצב מתח בעת נפילה נמוכה L387 אשר מייצב את המתח הישר במבוא (בין 5.5v ל-35v) למתח ישר של 5v ±4%.

רגל 1 של הרכיב מתחברת למתח הכניסה אשר מסופק ע"י סוללות. רגל זו מתחבר גם דרך קבל בגודל 100nF לאדמה.

רגל 3 מתחברת לאדמה על מנת לשמש מתח יחוס להפרש פוטנציאלים.

רגל 5 מתחברת לפס ה-VCC המשותף אליו אני אחבר את שאר רכיבי המערכת (רשום בהמשך). רגל זו מתחברת לאדמה דרך קבל שגודלו 100µF.

תרשים מעגל ההספקה:



4.3.2 תצוגת גביש נוזלי LCD וחיבורה למיקרו מעבד:

התצוגה כוללת 14 הדקים מהם אני משתמשת ב-10 הדקים בלבד.

התצוגה משמשת בתור התקן פלט לבקרה על פעולתו התקינה של הרובוט.

8 הדקים (D0-D7) משמשים לקבלת נתונים אשר יופיעו על התצוגה. מכיוון ואני משתמשת בתצורת עבודה של ארבעה ביטים, מתוך שמונת ההדקים אני משתמשת רק בארבעת העליונים (D4-D7), הם מתחברים להדקי Port 2 של המיקרו מעבד. מכיוון וקודי הבקרה והתווים ה-ASCII הינם בגודל שמונה ביטים אנו נשלח קודם את ארבעת הסיביות העליונות של הנתון ולאחר מכן את ארבעת הסיביות התחתונות. הדקים אלו יחוברו להדקי P2.0-P2.3 של המיקרו מעבד.

הדק 1 VCC של התצוגה משמש לחיבור מתח הזנה לפס ה-VCC המשותף.

הדק 2 משמש לחיבור התצוגה לאדמה המשותפת.

הדק 3 משמש לבקרה על הארת המסך, מתחבר לרגל הבורר של הטרימר וע"י שינוי ההתנגדות אני מגדילה או מקטינה את עוצמת ההארה.

הדק 4, הדק ה-RS הינו הדק לברירה בין אוגר תצוגה לבין אוגר בקרה. כאשר הדק זה מקבל אפס לוגי ה-LCD עובר לאוגר בקרה, כלומר כל מידע שיתקבל בהדקי הנתונים כאשר הדק זה ב-'0' יהיו תווי בקרה, תווים אלו הינם קודים אשר קובעים את תצורת פעולת התצוגה. כמוכן אלו יכולים להיות קודים אשר באים לביצוע פעולה מסוימת כגון: מחיקה, ירידת שורה, הזזת הסמן, הבהוב הסמן וכד'... ואילו כאשר הדק זה ב-'1' ה-LCD עובר לאוגר תצוגה וכל מידע שיתקבל בהדקי הנתונים במצב זה יהווה תווי תצוגה. לתצוגה נשלח את המידע אותו נרצה להציג בתור תווים ב- ASCII. בדק זה מחובר להדק P2.4 של המיקרו מעבד.

הדק 5, הדק R/W המצביע על מצב קריאה או כתיבה לתצוגה. כאשר הדק זה באפס מתבצעת כתיבה וכאשר הינו נמצא באחד לוגי תתבצע קריאה מה-LCD. בפרויקט שלי הדק זה מחובר האופן קבועה לאדמה מכיוון ובפרויקט זה נכתוב אליו בלבד.

הדק 6, הדק E, הדק האפשור של התצוגה. אפס בהדק זה מציין כי התצוגה אינה מאופשרת, ואילו '1' בהדק זה מציין כי התצוגה מאופשר וניתן לעבוד אימה, כלומר כל נתונן שישלח אליה יקלט. הדק זה מתחבר להדק P2.5 של ה-ARM.

תרשים חיבור LCD למערכת:



4.3.3 מעגל לשליטה על מנועים מבוסס L298n גשר-H כפול מלא:

מכיוון והמעבד אינו יכול לספק מספיק זרם להפעלת המנועים, מעגל זה בא לפתור בעיה זו. L298 מכיל בתוכו שני גשרי H מלאים, פעולת גשר H מוסברת בנספח ד'. ל-L298n חמישה עשר הדקים:

הדק 1 והדק 15 מתחברים לאדמה דרך נגדים.

הדקים 2,3, 13 ו-14 הם הדקי מוצא של שני גשרי H, הם מתחברים לשני מנועים.

הדק 4 מתחבר למתח (בין 2.5v יותר מהמתח בכניסה להדקי המבוא לבין 46v) להפעלת המנועים.

הדקים 5, 7, 10 ו-12 הם הדקי כניסה לשליטה על גשרי H, מתחברים אל הדקים P0.4-P0.7.

הדקים 6 ו-11 הדקי אפשור של גשרי H, מתחברים אל הדקים P2.6 ו-P2.7.

הדק 8 מתחבר לאדמה.

הדק 9 מתחבר למתח ההזנה VCC (בין 4.5v לבין 7v).

תרשים חיבור מעגל שליטה על מנועים:



4.3.4 מיקרו מעבד LPC1769:

להדק ה-External power מתחבר הפס ה-VCC המשותף, זהו הדק ההזנה של המעבד.

הדקים P2.0-P2.5 הם הדקים שהגדרתי כמוצא של המעבד מתחברים להדקי הנתונים הגבוהים של תצוגת הגביש הנוזלי (D4-D7), להדק ה-RS ולהדק האפשור של ה-LCD.

הדקי P0.8 ו-P0.9 מחוברים ל-LED אדום וכחול אשר מהווים פלט המערכת.

הדקים P0.4-P0.7 מהווים את פלט המערכת לגשר H לשם שליטה על המנועים.

הדקים P2.6 ו-P2.7 מתחברים להדקי האפשור של L298n.

הדק P0.0 הינו הדק קליטה של ה-UART ואליו מתחבר המוצא של המקלט RR3.

הפוטוטרנזיסטורים מתחברים להדקים P0.25-P0.28 דרך נגד הגנה של 1kΩ.

הדקים P0.3 ו-P0.2 והדקים P0.23 ו-P0.24 מחוברים לשם תחליף למקלט RX-2 שעליו נעשה reversing engineering.

תרשים חיבור מיקרו מעבד של המכונית:



4.3.5 מקלט RR3-433.92:

זהו התקן קלט לתקשורת רדיו, מקלט AM שתדר גל הנושא שלו הינו 433.92MHz. על מנת לקלוט את אות המשדר אני משתמשת באנטנת מוט שאורכה רבע מאורך הגל.

למקלט חמישה עשר הדקים אשר חיבורם מתבצע עלפי דפי הנתונים.

הדקים 1, 10, 12 ו-15 מתחברים לפס ה-VCC המשותף. אלו הם הדקי הזנה של חלקי הרכיב, הם מספקים מתח לפעולת הגלאי והמקלט.

הדקים 4, 5, 6, 8 ו-9 אינם מתחברים לשום הדק.

הדקים 2, 7 ו-11 הם הדקי האדמה של הרכיב. אותם אני מחברת לאדמה המשותפת לשם יחוס רמות המתחים.

הדק 3 זהו הדק המתחבר לאנטנה המקבלת את האות.

הדק 13 זהו הדק לבדיקת פעולת הרכיב ובו אין שימוש בפרויקט.

הדק 14 זהו הדק אותו אני מחברת להדק הקליטה של התקשורת הטורית שאותו מחברים ל-P0.0 שהוא הדק הקליטה של המעבד בתקשורת UART.

תרשים חיבור המקלט:



4.3.6 פוטוטרנזיסטורים:

משתמשים בפוטוטרנזיסטורים TEPT5600 NPN הם מתחברים להדקי המעבד, ומהווים התקן קלט למערכת.

הם מחוברים להדקים P0.25-P0.28 הם מחוברים עם ההדק קולט דרך נגד של 1kΩ לצורך הגנת המיקרו מעבד מזרמים גבוהים. אותו ההדק דרך נגד של 100kΩ מתחבר ל-VCC המשותף. עם הדק הפולט הפוטוטרנזיסטור מתחבר לאדמה.

כאשר נקלט אור בבסיסו הוא עובר למצב פעיל, רוויה כך עליו נופל מתח נמוך מאוד המתפרש ע"י המעבד כ-'0'. ראו דיאגרמת ניסוי בפרק חישובים ומדידות.

תרשים חיבור TEPT5600:



4.3.7 LED להמקרה:

אלו LED בצבעים אדום וכחול, הם משמשים לבקרה על פעולתו של הרובוט ומתחברים, האדום להדק P0.8 והכחול להדק P0.9 של המיקרו מעבד.

תרשים חיבור ה-LED:



4.4 הסבר סכמת שלט RF:

4.4.1 מיקרו מעבד LPC1769:

הדק P0.1 מוגדר בעזרת תוכנה כהדק לתקשורת לשידור נתונים, הדק זה מתחבר להדק השידור של המשדר מקלט.

להדקי פורט 2 (P2.0-P2.3) מתחברים לארבעה לחצנים לשליטה על כיוון התקדמות הרובוט דרך נגד הגנה של 1kΩ.

את המתח למעגל מספק רכיב LM7805, הוא מייצב את המתח במבואו (7v-20v) למתח של 5v. הוא תחבר להדק External Power של המעבד.

LED מחובר ל- P2.4 כאשר הוא ידלוק בזמן לחיצה על לחצן.

תרשים חיבור המעבד בשלט:



4.4.2 משדר RT4-433.92:

משדר תקשורת רדיו AM המשמש לשידור אות השלט למקלט דרך האנטנה אשר אורכה מחושב, המשדר זה נעשה שימוש באנטנת מוט שאשר מהווה רבע מאורך בגל. תדר הגל הנושא הינו 433.92MHz אשר הוא מתואם עם אורך הגל הנושא של המקלט בו אני משתמשת.

למשדר ארבעה הדקים, אשר חיבורם נעשה לפי דפי הנתונים.

הדק ראשון הינו הדק ההזנה של הרכיב אותו אני מחברת למתח ההזנה של כל השלט.

הדק שני של הרכיב הינו הדק האדמה אשר מתחבר למטרת יחוס המתחים ברכים. לשם כך אני מחברת אותו לאדמה המשותפת.

ההדק השלישי הינו ההדק אליו אני מחברת את הדק השידור של התקשות הטורית של ה-LPC1769.

ההדק האחרון של המשדר, הדק רביעי של המשדר הינו הדק אליו מתחברת האנטנה. כלומר הינו הדק המוצא של המשדר.

תרשים חיבור המשדר:



4.4.3 לחצני השלט:

זהו ממשק קלט של השלט בעזרתו המשתמש שולט על כיוון נסיעת הרובוט.

אלו הם בעצם ארבעה לחצנים המתחברים להדקי פורט 2 (P2.0-P2.3) של המיקרו מעבד דרך נגד 1kΩ להגנת המיקרו מעבד. אשר לחיצה על לחצן שונה מעידה על כיוון שונה.

כאשר ילחץ לחצן הוא יקצר לאדמה את ההדק אליו הוא מחובר והמעבד יזהה את הלחיצה.

תרשים חיבור הלחצנים:



4.4.4 מעגל הספקת מתח לשלט:

מעגל זה מבוסס על מייצב מתח LM7805, הוא מייצב את המתח בכניסתו למתח DC של 5v.

רגל אחת של המייצב מתחברת למתח הכניסה, אני מחברת אותו לשם ניוד לסוללות. המתח המסופק לרכיב חייב להיות בין 7v לבין 20v.

רגל שתיים של הרכיב מתחברת לאדמה.

רגל שלישית של המייצב מתחברת לפס ה-VCC המשותף אליו מתחברים כל שאר רכיבי השלט.

תרשים חיבור LM7805:



4.4.5 חיבור ה-LED:

הדפ"א מתחברת להדק P2.4 של המיקרו מעבד והיא נדלקת כאשר נלחץ לחצן בשלט.

תרשים חיבור ה-LED:



**פרק 5:**

**תוכנה**

5.1 הקדמה:

בפרויקט זה נעשה שימוש בשפת C המיועדת למיקרו מעבד LPC1769.

נשתמש בשני מעבדים לשם בניית המכונית והשלט, לשם כך יכתבו שתי תוכנות בשפה זו.

התוכנית של המעבד הראשון תיכתב לשם קליטת אות רדיו מהמשדר, הפעלת המנועים ותהליך הלמידה. התוכנית השנייה נכתבת במיוחד לשם שליטה על כיוון התקדמותו של הרובוט בעזרת תקשורת רדיו עם אפנון AM.

5.2 מבוא לתוכנית ב-C ל-ARM Cortex M-3:

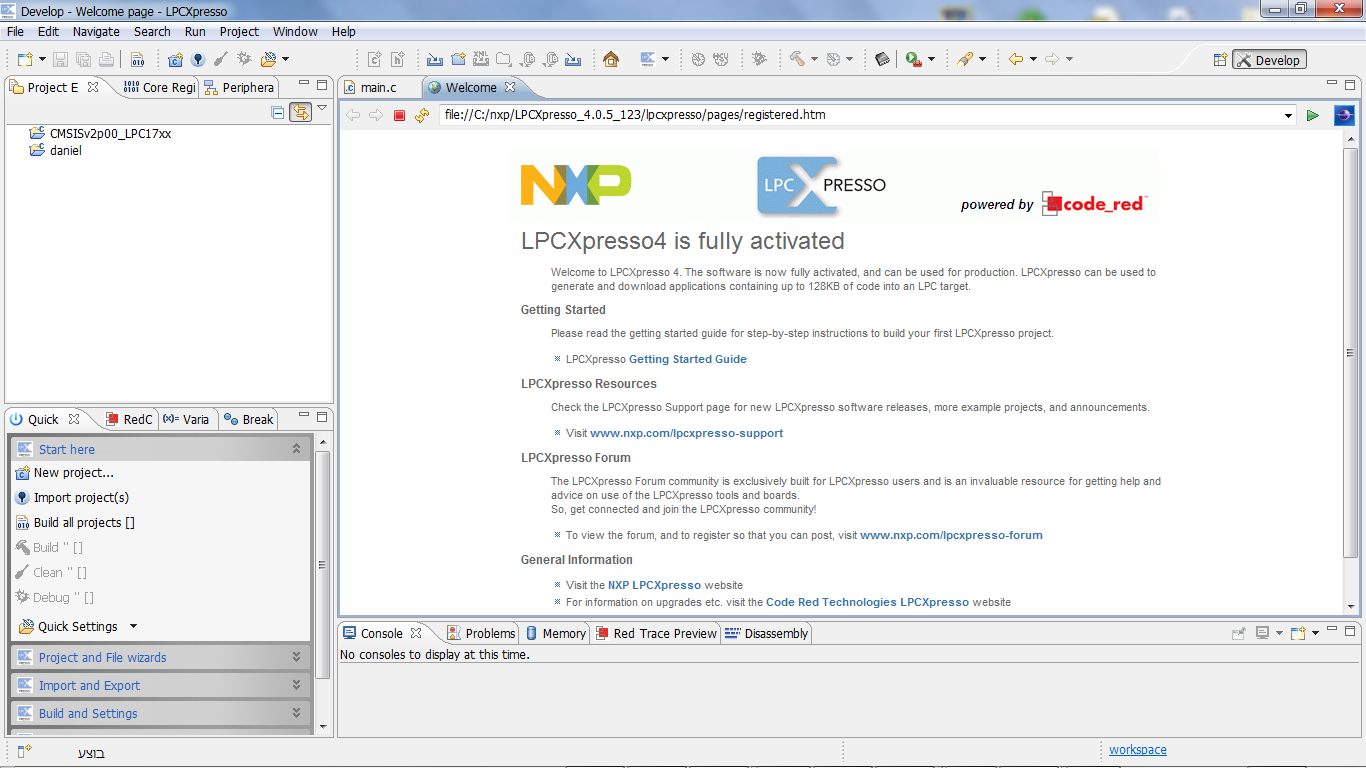
בשפה זו דרך מיוחדת לפנות לכל הדק, והפעולות המיוחדות של ההדקים כגון UART או PWM.

בשפה זו יש לקבוע את תצורת החיבור הפנימי של הדק המבוא, יש לקבוע האם הדק זה הוא מבוא או מוצא ויש לקבוע אם זהו הדק בתצורת GPIO או הדק PWM לדוגמה.

כל זה נעשה ע"י LPC\_GIOPx, LPC\_PINCON->PINSEL, LPC\_PINCON->PINMODE.

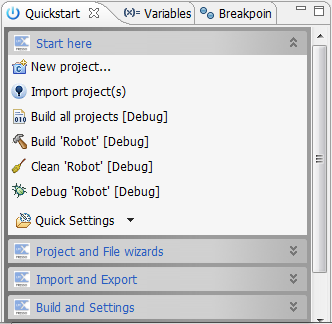
5.3 סביבת עבודה של code-red NXP

LPCXpresso v4.2.0



סביבת עבודה זו משמשת כתיבת תוכנות למעבדים מסוג ARM מתוצרת NXP ולא רק לכך אלה גם ל-Debug של התוכנות הנכתבות.

לסביבת העבודה ניתן להוסיף דוגמאות של פרויקטים וספריות לכתיבת התוכנות.



בעזרת Import progect אנו מיבאים דוגמאות של תוכנות וספריות.

5.4 תרשים זרימה של המעבד המתלמד:

מגדירים כניסות ויציאות

לולאה ראשית

תשאול הכניסות

מגדירים משתנים וקונסטרוקטורים

האטת לולאה להדפסה על –LCD

לימוד רשת נוירונים

הפעלת מנועים

חישובי רשת נוירונים

התפסה על המסך של שלב הלימוד או הודעה על סיום הלימוד

5.5 תרשים זרימה של ה-LPC המשדר (שלט):

מגדירים כניסות ויציאות

לולאה ראשית

תשאול הכניסות והדלקת דפ"א

מגדירים משתנים

שליחת נתונים למשדר

5.6 הסבר קבצי C ו-Headers:

LCD.c:

קובץ זה מכיל פונקציות לעבודה עם ה-LCD.

פונקציות לאתחול התצוגה והדפסה.

LCD.h:

קובץ זה הינו קובץ המכיל define הקשורים ב-LCD.

robot.h:

מכיל את ה-define של ההדקים הקשורים בהפעלת המנועים ואת הסטוקטורה של רשת נוירונים.

main.c:

מכיל את התוכנית הראשית כולל רשת נוירונים. כולל קלט ופלט של המיקרו מעבד ולימוד רשת הנוירונים. ופונקציה להפעלת המנועים.

5.7 תוכנה של בסיס המכונית:

Main.c

**int** **main**(**void**)

{

/\* SystemClockUpdate() updates the SystemFrequency variable \*/

SystemClockUpdate();

UARTInit(192000); /\* baud rate setting \*/

SysTick\_Config(SystemFrequency / 1000000);

LPC\_PINCON->PINSEL4 &= ~(0xFFFF); // 1111111111111111

LPC\_GPIO2->FIODIR0 |= 0xFF; // 11111111 | Init LCD pin

LPC\_PINCON->PINMODE0 &= ~(0xFF); // 11111111

LPC\_PINCON->PINSEL0 &= ~(0xFFFFFF); // 111111111111111111111111

LPC\_GPIO0->FIODIRL |= 0xFF0; // 111111110000

LPC\_PINCON->PINMODE1 &= ~(0xFF); // 11111111

LPC\_PINCON->PINSEL1 &= ~(0xFFFFFFFF); // 11111111111111111111111111111111

LPC\_GPIO0->FIODIRH &= ~(0x1F80); // 1111110000000

lcd\_init();

lcd\_print("I love you!!!");

**unsigned** **char** right, left, forword, backword;

**unsigned** **char** pht\_right, pht\_left, pht\_forword, pht\_backword;

LED\_RED\_OFF;

LED\_BLUE\_OFF;

delay\_us(1000000);

MOTOR1EN1;

MOTOR2EN1;

//init neural network

**int** i;

**float** a=0.1;

**for**(i=0;i<4;i++){

neo[i].w[0]=0;

neo[i].w[1]=0;

neo[i].w[2]=0;

neo[i].w[3]=0;

neo[i].ch=1;

neo[i].y=0;

}

**unsigned** **int** slowdownDelay = 1000000;

**int** counter = 0;

**while**(1) {

**if** (UART3Count==0)

{

LPC\_UART3->IER = IER\_THRE | IER\_RLS; /\* Disable RBR \*/

**if**(LPC\_UART3->RBR=='L')

left=1;

**else** **if**(LPC\_UART3->RBR=='F')

forword=1;

**else** **if**(LPC\_UART3->RBR=='R')

right=1;

**else** **if**(LPC\_UART3->RBR=='B')

backword=1;

LPC\_UART3->IER = IER\_THRE | IER\_RLS | IER\_RBR; /\* Re-enable RBR \*/

}

right = 'R' ? 1 : 0;

left = 'L' ? 1 : 0;

forword = 'F' ? 1 : 0;

backword = 'B' ? 1 : 0;

pht\_right = ((LPC\_GPIO0->FIOPIN & (1 << 25)) >> 25) ? 1 : 0;

pht\_left = ((LPC\_GPIO0->FIOPIN & (1 << 27)) >> 27) ? 1 : 0;

pht\_forword = ((LPC\_GPIO0->FIOPIN & (1 << 26)) >> 26) ? 1 : 0;

pht\_backword = ((LPC\_GPIO0->FIOPIN & (1 << 28)) >> 28) ? 1 : 0;

**if**(pht\_right == 0 || pht\_left == 0 || pht\_forword == 0 || pht\_backword == 0)

{

LED\_RED\_ON;

}

**else**

LED\_RED\_OFF;

// Slowdown

**if**(slowdownDelay == 0)

{

slowdownDelay = 1000000;

**int** x[4],c[4];

x[0] = ~pht\_right;

x[1] = ~pht\_left;

x[2] = ~pht\_forword;

x[3] = ~pht\_backword;

**for**(i=0;i<4;i++)

Neuron(x,i);

**if**(learning == 1)

{

**int** j,d[4];

d[0] = right;

d[1] = left;

d[2] = forword;

d[3] = backword;

**for**(i=0;i<4;i++)

**for**(j=0;j<4;j++)

neo[i].w[j]=neo[i].w[j] + a \* (d[i] - neo[i].y) \* x[j];

counter++;

**int** count;

**float** E=0,E1=1,Err;

count=counter;

**for**(i=3;i>=0;i--){

c[i]=count%10+0x30;

count=count/10;

}

lcd\_clr();

lcd\_print(c);

E1=0;

**for**(i=0;i<4;i++){

E1+=**pow**(d[i]-neo[i].y,2);

}

E+=E1; // /4

Err=**sqrt**(E);

**if** (Err<0.5)

{

learning=0;

lcd\_clr();

lcd\_print("All OK!!!");

}

**else**

learning=1;

}

}

**else**

slowdownDelay--;

**if**(neo[0].y == 1)

{

MOTORR\_F1;

MOTORR\_B0;

}

**else** **if**(neo[1].y == 1)

{

MOTORL\_F1;

MOTORR\_F1;

MOTORL\_B0;

MOTORR\_B0;

}

**else** **if**(neo[2].y == 1)

{

MOTORL\_F1;

MOTORL\_B0;

}

**else** **if**(neo[3].y == 1)

{

MOTORL\_B1;

MOTORR\_B1;

MOTORL\_F0;

MOTORR\_F0;

LED\_RED\_ON;

}

**else**

{

MOTORL\_F0;

MOTORR\_F0;

MOTORL\_B0;

MOTORR\_B0;

}

}

**return** 0;

}

Robot.h

**#ifndef** ROBOT\_H\_

**#define** ROBOT\_H\_

**#define** LED\_RED\_ON LPC\_GPIO0->FIOSETL |= (1 << 8);

**#define** LED\_RED\_OFF LPC\_GPIO0->FIOCLRL |= (1 << 8);

**#define** LED\_BLUE\_ON LPC\_GPIO0->FIOSETL |= (1 << 9);

**#define** LED\_BLUE\_OFF LPC\_GPIO0->FIOCLRL |= (1 << 9);

**#define** MOTORL\_F1 LPC\_GPIO0->FIOSETL |= (1 << 7);

**#define** MOTORL\_B1 LPC\_GPIO0->FIOSETL |= (1 << 6);

**#define** MOTORR\_F1 LPC\_GPIO0->FIOSETL |= (1 << 5);

**#define** MOTORR\_B1 LPC\_GPIO0->FIOSETL |= (1 << 4);

**#define** MOTORL\_F0 LPC\_GPIO0->FIOCLRL |= (1 << 7);

**#define** MOTORL\_B0 LPC\_GPIO0->FIOCLRL |= (1 << 6);

**#define** MOTORR\_F0 LPC\_GPIO0->FIOCLRL |= (1 << 5);

**#define** MOTORR\_B0 LPC\_GPIO0->FIOCLRL |= (1 << 4);

**#define** MOTOR1EN1 LPC\_GPIO2->FIOSETL |= (1 << 6);

**#define** MOTOR1EN0 LPC\_GPIO2->FIOCLRL |= (1 << 6);

**#define** MOTOR2EN1 LPC\_GPIO2->FIOSETL |= (1 << 7);

**#define** MOTOR2EN0 LPC\_GPIO2->FIOCLRL |= (1 << 7);

**typedef** **struct** {

**float** w[4];

**int** ch;

**int** y;

}neuron;

**#endif** /\* ROBOT\_H\_ \*/

LCD.c

**#ifdef** \_\_USE\_CMSIS

**#include** "LPC17xx.h"

**#endif**

**#include** "LCD.h"

**void** **SysTick\_Handler**(**void**);

**void** **delay\_us**(uint32\_t);

**static** **volatile** uint32\_t usTicks = 0;

**void** **SysTick\_Handler**(**void**){

usTicks++; }

**void** **delay\_us**(uint32\_t us){

uint32\_t startTicks;

startTicks = usTicks;

**while**((usTicks - startTicks) < us); }

**void** **delay**(**unsigned** **int** i)

{

**while**(i--);

}

**void** **lcd\_clr**(**void**)

{

lcd\_command(0x01);

delay\_us(5000);

}

**void** **lcd\_init**(**void**)

{

delay\_us(40000);

lcd\_init\_write(0x3);

delay\_us(5000);

lcd\_command(0x30);

delay\_us(5000);

lcd\_command(0x30);

delay\_us(5000);

lcd\_command(0x20);

delay\_us(5000);

lcd\_command(0x28);

delay\_us(1000);

lcd\_command(0x06);

delay\_us(1000);

lcd\_command(0x0F);

delay\_us(1000);

lcd\_command(0x01);

delay\_us(5000);

}

**void** **lcd\_init\_write**(**unsigned** **char** a)

{

RS0;

LPC\_GPIO2->FIOCLR0 = 0xF;

LPC\_GPIO2->FIOSET0 = a;

EN1;

delay\_us(100);

EN0;

}

**void** **lcd\_write4**(**unsigned** **char** value)

{

LPC\_GPIO2->FIOCLR0 = 0xF;

LPC\_GPIO2->FIOSET0 = (value & 0xF);

EN1;

delay\_us(100);

EN0;

}

**void** **lcd\_write**(**unsigned** **char** value)

{

lcd\_write4(value >> 4);

lcd\_write4(value);

}

**void** **lcd\_command**(**unsigned** **char** value)

{

RS0;

lcd\_write(value);

}

**void** **lcd\_print**(**char** chap[])

{

**int** count=0;

**while**(chap[count]!='\0')

{

RS1;

lcd\_write(chap[count]);

}

}

LCD.h

**#ifndef** LCD\_H\_

**#define** LCD\_H\_

**#define** RS0 LPC\_GPIO2->FIOCLR0 |= 0x10;

**#define** RS1 LPC\_GPIO2->FIOSET0 |= 0x10;

**#define** EN0 LPC\_GPIO2->FIOCLR0 |= 0x20;

**#define** EN1 LPC\_GPIO2->FIOSET0 |= 0x20;

**void** **delay**(**unsigned** **int**);

**void** **lcd\_init**(**void**);

**void** **lcd\_write4**(**unsigned** **char**);

**void** **lcd\_write**(**unsigned** **char**);

**void** **lcd\_init\_write**(**unsigned** **char**);

**void** **lcd\_command**(**unsigned** **char**);

**void** **lcd\_print**( **char** []);

**void** **lcd\_clr**(**void**);

**#endif** /\* LCD\_H\_ \*/

Uart.c

**#include** "lpc17xx.h"

**#include** "type.h"

**#include** "uart.h"

**volatile** uint32\_t UART3Status;

**volatile** uint8\_t UART3TxEmpty = 1;

**volatile** uint8\_t UART3Buffer[BUFSIZE];

**volatile** uint32\_t UART3Count = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Function name: UART0\_IRQHandler

\*\*

\*\* Descriptions: UART0 interrupt handler

\*\*

\*\* parameters: None

\*\* Returned value: None

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **UART3\_IRQHandler** (**void**)

{

uint8\_t IIRValue, LSRValue;

uint8\_t Dummy = Dummy;

IIRValue = LPC\_UART0->IIR;

IIRValue >>= 1; /\* skip pending bit in IIR \*/

IIRValue &= 0x07; /\* check bit 1~3, interrupt identification \*/

**if** ( IIRValue == IIR\_RLS ) /\* Receive Line Status \*/

{

LSRValue = LPC\_UART0->LSR;

/\* Receive Line Status \*/

**if** ( LSRValue & (LSR\_OE|LSR\_PE|LSR\_FE|LSR\_RXFE|LSR\_BI) )

{

/\* There are errors or break interrupt \*/

/\* Read LSR will clear the interrupt \*/

UART3Status = LSRValue;

Dummy = LPC\_UART0->RBR; /\* Dummy read on RX to clear

interrupt, then bail out \*/

**return**;

}

**if** ( LSRValue & LSR\_RDR ) /\* Receive Data Ready \*/

{

/\* If no error on RLS, normal ready, save into the data buffer. \*/

/\* Note: read RBR will clear the interrupt \*/

UART3Buffer[UART3Count] = LPC\_UART3->RBR;

UART3Count++;

**if** ( UART3Count == BUFSIZE )

{

UART3Count = 0; /\* buffer overflow \*/

}

}

}

**else** **if** ( IIRValue == IIR\_RDA ) /\* Receive Data Available \*/

{

/\* Receive Data Available \*/

UART3Buffer[UART3Count] = LPC\_UART3->RBR;

UART3Count++;

**if** ( UART3Count == BUFSIZE )

{

UART3Count = 0; /\* buffer overflow \*/

}

}

**else** **if** ( IIRValue == IIR\_CTI ) /\* Character timeout indicator \*/

{

/\* Character Time-out indicator \*/

UART3Status |= 0x100; /\* Bit 9 as the CTI error \*/

}

**else** **if** ( IIRValue == IIR\_THRE ) /\* THRE, transmit holding register empty \*/

{

/\* THRE interrupt \*/

LSRValue = LPC\_UART3->LSR; /\* Check status in the LSR to see if

valid data in U3THR or not \*/

**if** ( LSRValue & LSR\_THRE )

{

UART3TxEmpty = 1;

}

**else**

{

UART3TxEmpty = 0;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Function name: UARTInit

\*\*

\*\* Descriptions: Initialize UART port, setup pin select,

\*\* clock, parity, stop bits, FIFO, etc.

\*\*

\*\* parameters: UART baudrate

\*\* Returned value: true

installed to the

\*\* VIC table

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint32\_t **UARTInit**(uint32\_t baudrate )

{

uint32\_t Fdiv;

uint32\_t pclkdiv, pclk;

LPC\_PINCON->PINSEL0 &= ~0x0000000F;

LPC\_PINCON->PINSEL0 |= 0x0000000A; /\* Enable RxD1 P0.1, TxD1 P0.0 \*/

/\* By default, the PCLKSELx value is zero, thus, the PCLK for

all the peripherals is 1/4 of the SystemFrequency. \*/

/\* Bit 8,9 are for UART3 \*/

pclkdiv = (LPC\_SC->PCLKSEL0 >> 8) & 0x03;

**switch** ( pclkdiv )

{

**case** 0x00:

**default**:

pclk = SystemFrequency/4;

**break**;

**case** 0x01:

pclk = SystemFrequency;

**break**;

**case** 0x02:

pclk = SystemFrequency/2;

**break**;

**case** 0x03:

pclk = SystemFrequency/8;

**break**;

}

LPC\_UART3->LCR = 0x83; /\* 8 bits, no Parity, 1 Stop bit \*/

Fdiv = ( pclk / 16 ) / baudrate ; /\*baud rate \*/

LPC\_UART3->DLM = Fdiv / 256;

LPC\_UART3->DLL = Fdiv % 256;

LPC\_UART3->LCR = 0x03; /\* DLAB = 0 \*/

LPC\_UART3->FCR = 0x07; /\* Enable and reset TX and RX FIFO. \*/

NVIC\_EnableIRQ(*UART3\_IRQn*);

LPC\_UART3->IER = IER\_RBR | IER\_THRE | IER\_RLS; /\* Enable UART3 interrupt \*/

**return** (0);

}

Uart.h

**#ifndef** \_\_UART\_H

**#define** \_\_UART\_H

**#define** IER\_RBR 0x01

**#define** IER\_THRE 0x02

**#define** IER\_RLS 0x04

**#define** IIR\_PEND 0x01

**#define** IIR\_RLS 0x03

**#define** IIR\_RDA 0x02

**#define** IIR\_CTI 0x06

**#define** IIR\_THRE 0x01

**#define** LSR\_RDR 0x01

**#define** LSR\_OE 0x02

**#define** LSR\_PE 0x04

**#define** LSR\_FE 0x08

**#define** LSR\_BI 0x10

**#define** LSR\_THRE 0x20

**#define** LSR\_TEMT 0x40

**#define** LSR\_RXFE 0x80

**#define** BUFSIZE 0x40

uint32\_t **UARTInit**( uint32\_t Baudrate );

**void** **UART3\_IRQHandler**( **void** );

**#endif** /\* end \_\_UART\_H \*/

5.8 הסבר קבצי C ו-Headers:

Main.c:

קובץ זה מכיל את התוכנית של השלט.

Uart.c:

קובץ זה מכיל פונקציות להפעלת ה-UART.

Uart.h:

מכיל את ה-define של UART.

5.9 תוכנת השלט:

Main.c

**#include** <cr\_section\_macros.h>

**#include** <NXP/crp.h>

// Variable to store CRP value in. Will be placed automatically

// by the linker when "Enable Code Read Protect" selected.

// See crp.h header for more information

\_\_CRP **const** **unsigned** **int** CRP\_WORD = CRP\_NO\_CRP ;

**#include** "lpc17xx.h"

**#include** "type.h"

**#include** "uart.h"

**extern** **volatile** uint32\_t UART3Count;

**extern** **volatile** uint8\_t UART3Buffer[BUFSIZE];

**int** **main** (**void**)

{

/\* SystemClockUpdate() updates the SystemFrequency variable \*/

SystemClockUpdate();

LPC\_PINCON->PINSEL4 &=~(0xF);

LPC\_GPIO2->FIODIR0 &=~(0xF);

UARTInit(192000); /\* baud rate setting \*/

**while** (1)

{ /\* Loop forever \*/

**if** ( UART3Count != 0 )

{

**if** (LPC\_GPIO2->FIOPIN0 & (1 << 0)>>0)

{

LPC\_UART3->IER = IER\_THRE | IER\_RLS; /\* Disable RBR \*/

UARTSend( 'L' );

UART3Count = 0;

LED\_ON;

LPC\_UART3->IER = IER\_THRE | IER\_RLS | IER\_RBR; /\* Re-enable RBR \*/

}

**else** **if** (LPC\_GPIO2->FIOPIN0 & (1 << 1)>>1)

{

LPC\_UART3->IER = IER\_THRE | IER\_RLS; /\* Disable RBR \*/

UARTSend( 'F' );

UART3Count = 0;

LED\_ON;

LPC\_UART3->IER = IER\_THRE | IER\_RLS | IER\_RBR; /\* Re-enable RBR \*/

}

**else** **if** (LPC\_GPIO2->FIOPIN0 & (1 << 2)>>2)

{

LPC\_UART3->IER = IER\_THRE | IER\_RLS; /\* Disable RBR \*/

UARTSend( 'R' );

UART3Count = 0;

LED\_ON;

LPC\_UART3->IER = IER\_THRE | IER\_RLS | IER\_RBR; /\* Re-enable RBR \*/

}

**else** **if** (LPC\_GPIO2->FIOPIN0 & (1 << 3)>>3)

{

LPC\_UART3->IER = IER\_THRE | IER\_RLS; /\* Disable RBR \*/

UARTSend( 'B');

UART3Count = 0;

LED\_ON;

LPC\_UART3->IER = IER\_THRE | IER\_RLS | IER\_RBR; /\* Re-enable RBR \*/

}

**else**

LED\_OFF;

}

}

**return** 0;

}

Uart.c

**#include** "lpc17xx.h"

**#include** "type.h"

**#include** "uart.h"

**volatile** uint32\_t UART3Status;

**volatile** uint8\_t UART3TxEmpty = 1;

**volatile** uint8\_t UART3Buffer[BUFSIZE];

**volatile** uint32\_t UART3Count = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Function name: UART3\_IRQHandler

\*\*

\*\* Descriptions: UART3 interrupt handler

\*\*

\*\* parameters: None

\*\* Returned value: None

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **UART3\_IRQHandler** (**void**)

{

uint8\_t IIRValue, LSRValue;

uint8\_t Dummy = Dummy;

IIRValue = LPC\_UART0->IIR;

IIRValue >>= 1; /\* skip pending bit in IIR \*/

IIRValue &= 0x07; /\* check bit 1~3, interrupt identification \*/

**if** ( IIRValue == IIR\_RLS ) /\* Receive Line Status \*/

{

LSRValue = LPC\_UART3->LSR;

/\* Receive Line Status \*/

**if** ( LSRValue & (LSR\_OE|LSR\_PE|LSR\_FE|LSR\_RXFE|LSR\_BI) )

{

/\* There are errors or break interrupt \*/

/\* Read LSR will clear the interrupt \*/

UART3Status = LSRValue;

Dummy = LPC\_UART3->RBR; /\* Dummy read on RX to clear

interrupt, then bail out \*/

**return**;

}

**if** ( LSRValue & LSR\_RDR ) /\* Receive Data Ready \*/

{

/\* If no error on RLS, normal ready, save into the data buffer. \*/

/\* Note: read RBR will clear the interrupt \*/

UART3Buffer[UART3Count] = LPC\_UART0->RBR;

UART3Count++;

**if** ( UART3Count == BUFSIZE )

{

UART3Count = 0; /\* buffer overflow \*/

}

}

}

**else** **if** ( IIRValue == IIR\_RDA ) /\* Receive Data Available \*/

{

/\* Receive Data Available \*/

UART3Buffer[UART3Count] = LPC\_UART3->RBR;

UART3Count++;

**if** ( UART3Count == BUFSIZE )

{

UART3Count = 0; /\* buffer overflow \*/

}

}

**else** **if** ( IIRValue == IIR\_CTI ) /\* Character timeout indicator \*/

{

/\* Character Time-out indicator \*/

UART3Status |= 0x100; /\* Bit 9 as the CTI error \*/

}

**else** **if** ( IIRValue == IIR\_THRE ) /\* THRE, transmit holding register empty \*/

{

/\* THRE interrupt \*/

LSRValue = LPC\_UART3->LSR; /\* Check status in the LSR to see if

valid data in U0THR or not \*/

**if** ( LSRValue & LSR\_THRE )

{

UART3TxEmpty = 1;

}

**else**

{

UART3TxEmpty = 0;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Function name: UARTInit

\*\*

\*\* Descriptions: Initialize UART port, setup pin select,

\*\* clock, parity, stop bits, FIFO, etc.

\*\*

\*\* parameters: UART baudrate

\*\* Returned value: true

installed to the

\*\* VIC table

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint32\_t **UARTInit**( uint32\_t baudrate )

{

uint32\_t Fdiv;

uint32\_t pclkdiv, pclk;

LPC\_PINCON->PINSEL0 &= ~0x0000000F;

LPC\_PINCON->PINSEL0 |= 0x0000000A; /\* Enable RxD1 P0.1, TxD1 P0.0 \*/

/\* By default, the PCLKSELx value is zero, thus, the PCLK for

all the peripherals is 1/4 of the SystemFrequency. \*/

/\* Bit 8,9 are for UART1 \*/

pclkdiv = (LPC\_SC->PCLKSEL0 >> 8) & 0x03;

**switch** ( pclkdiv )

{

**case** 0x00:

**default**:

pclk = SystemFrequency/4;

**break**;

**case** 0x01:

pclk = SystemFrequency;

**break**;

**case** 0x02:

pclk = SystemFrequency/2;

**break**;

**case** 0x03:

pclk = SystemFrequency/8;

**break**;

}

LPC\_UART3->LCR = 0x83; /\* 8 bits, no Parity, 1 Stop bit \*/

Fdiv = ( pclk / 16 ) / baudrate ; /\*baud rate \*/

LPC\_UART3->DLM = Fdiv / 256;

LPC\_UART3->DLL = Fdiv % 256;

LPC\_UART3->LCR = 0x03; /\* DLAB = 0 \*/

LPC\_UART3->FCR = 0x07; /\* Enable and reset TX and RX FIFO. \*/

NVIC\_EnableIRQ(*UART3\_IRQn*);

LPC\_UART3->IER = IER\_RBR | IER\_THRE | IER\_RLS; /\* Enable UART3 interrupt \*/

**return** (TRUE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Function name: UARTSend

\*\*

\*\* Descriptions: Send data to the UART 3 port based

\*\*

\*\*

\*\* parameters: buffer

\*\* Returned value: None

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **UARTSend**( uint8\_t BufferPtr )

{

/\* THRE status, contain valid data \*/

**while** ( !(UART3TxEmpty & 0x01) );

LPC\_UART3->THR = BufferPtr;

UART3TxEmpty = 0; /\* not empty in the THR until it shifts out \*/

**return**;

}

Uart.h

**#ifndef** \_\_UART\_H

**#define** \_\_UART\_H

**#define** IER\_RBR 0x01

**#define** IER\_THRE 0x02

**#define** IER\_RLS 0x04

**#define** IIR\_PEND 0x01

**#define** IIR\_RLS 0x03

**#define** IIR\_RDA 0x02

**#define** IIR\_CTI 0x06

**#define** IIR\_THRE 0x01

**#define** LSR\_RDR 0x01

**#define** LSR\_OE 0x02

**#define** LSR\_PE 0x04

**#define** LSR\_FE 0x08

**#define** LSR\_BI 0x10

**#define** LSR\_THRE 0x20

**#define** LSR\_TEMT 0x40

**#define** LSR\_RXFE 0x80

**#define** BUFSIZE 0x40

**#define** LED\_ON LPC\_GPIO2->FIOSET0|=0x10;

**#define** LED\_OFF LPC\_GPIO2->FIOCLR0|=0x10;

uint32\_t **UARTInit**( uint32\_t Baudrate );

**void** **UART0\_IRQHandler**( **void** );

**void** **UART1\_IRQHandler**( **void** );

**void** **UARTSend**( uint8\_t BufferPtr );

**#endif** /\* end \_\_UART\_H \*/

**פרק 6:**

**חישובים ומדידות**

חישוב אורך האנטנה המינימלי שלפי דפי הנתונים של המקלט ושל המשדר אמור להיות רבע מאורך הגל.

f- תדר גל הנושא

λ- אורך הגל

c- מהירות האור

Lmin- האורך המינימלי של האנטנה.

ניסוי רגישות הפוטוטרנזיסטור

נעשה ניסוי על מנת לגלות את ההתנגדות הטובה ביותר לרגישות המתאימה ביותר לשליטה ע"י פנס LED.

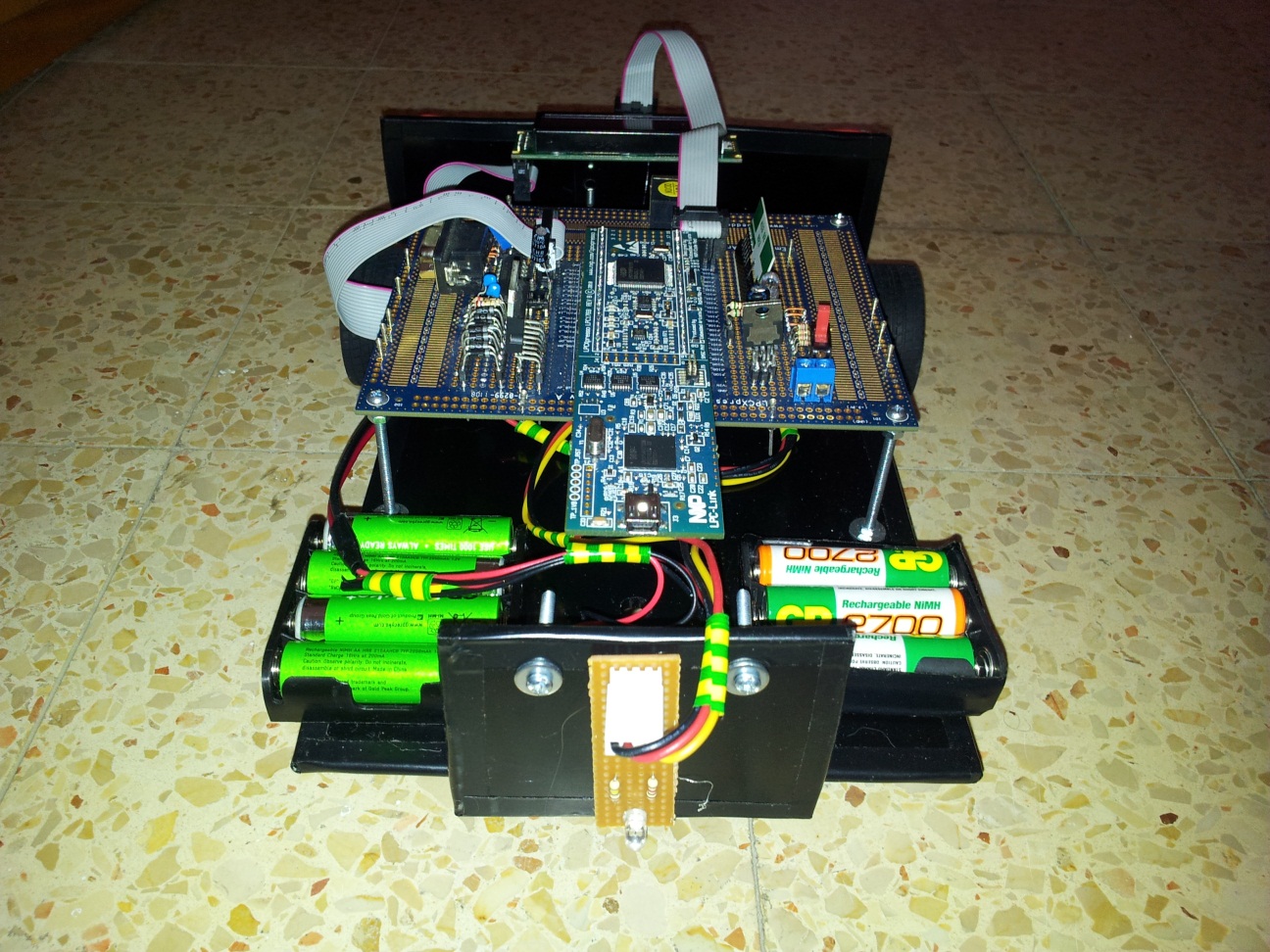
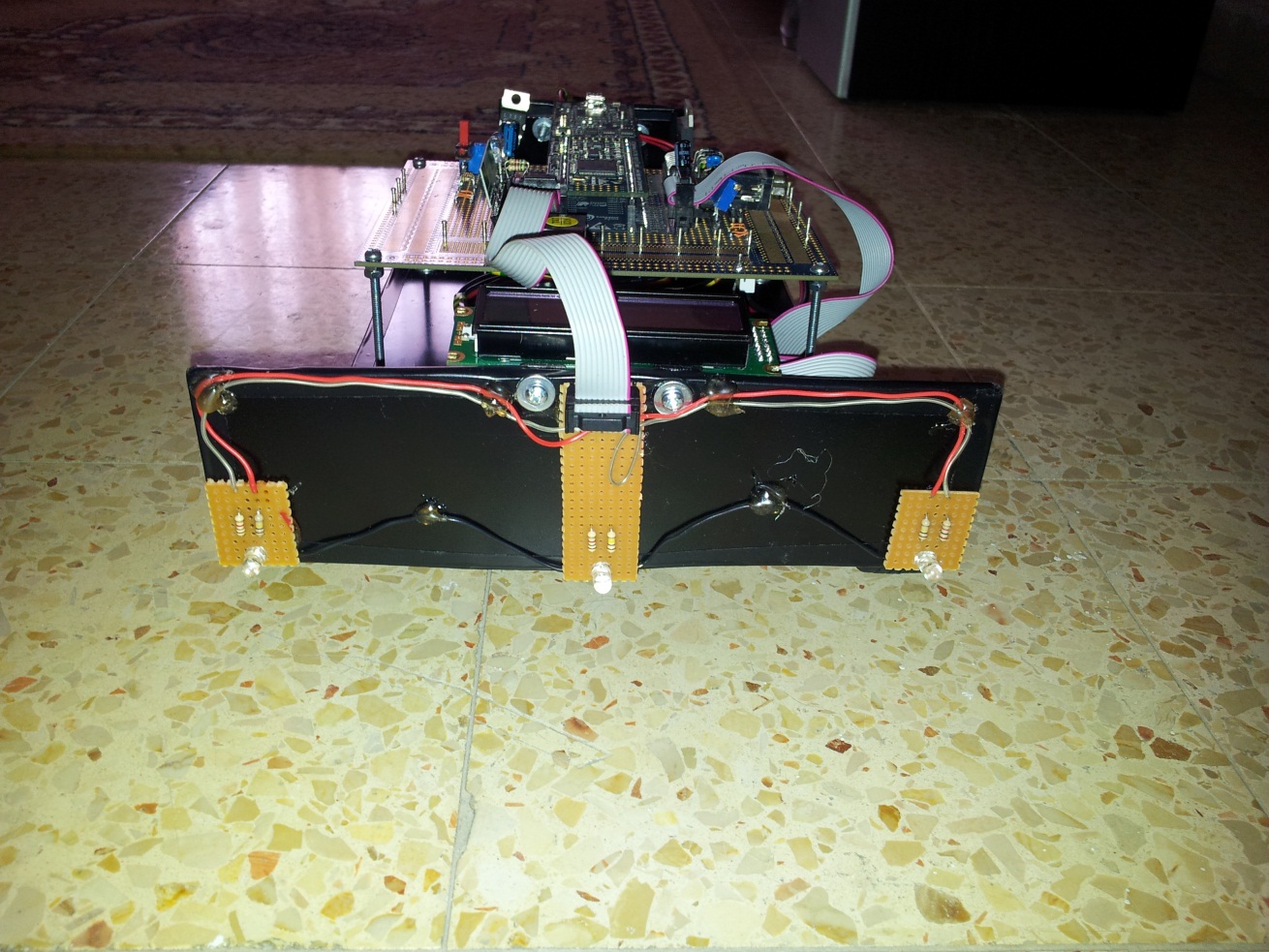
הניסוי נעשה עם פנס ממרחק 20 ס"מ.

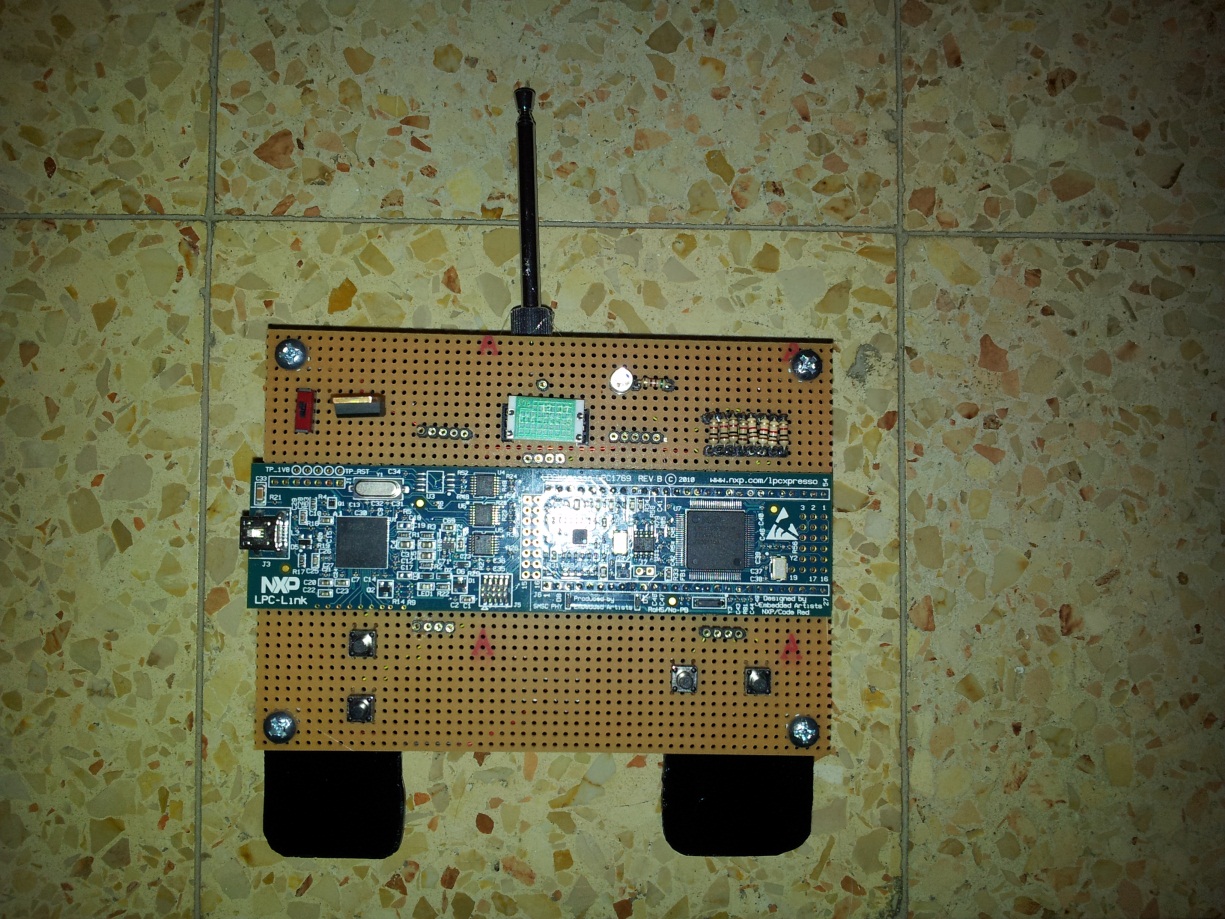
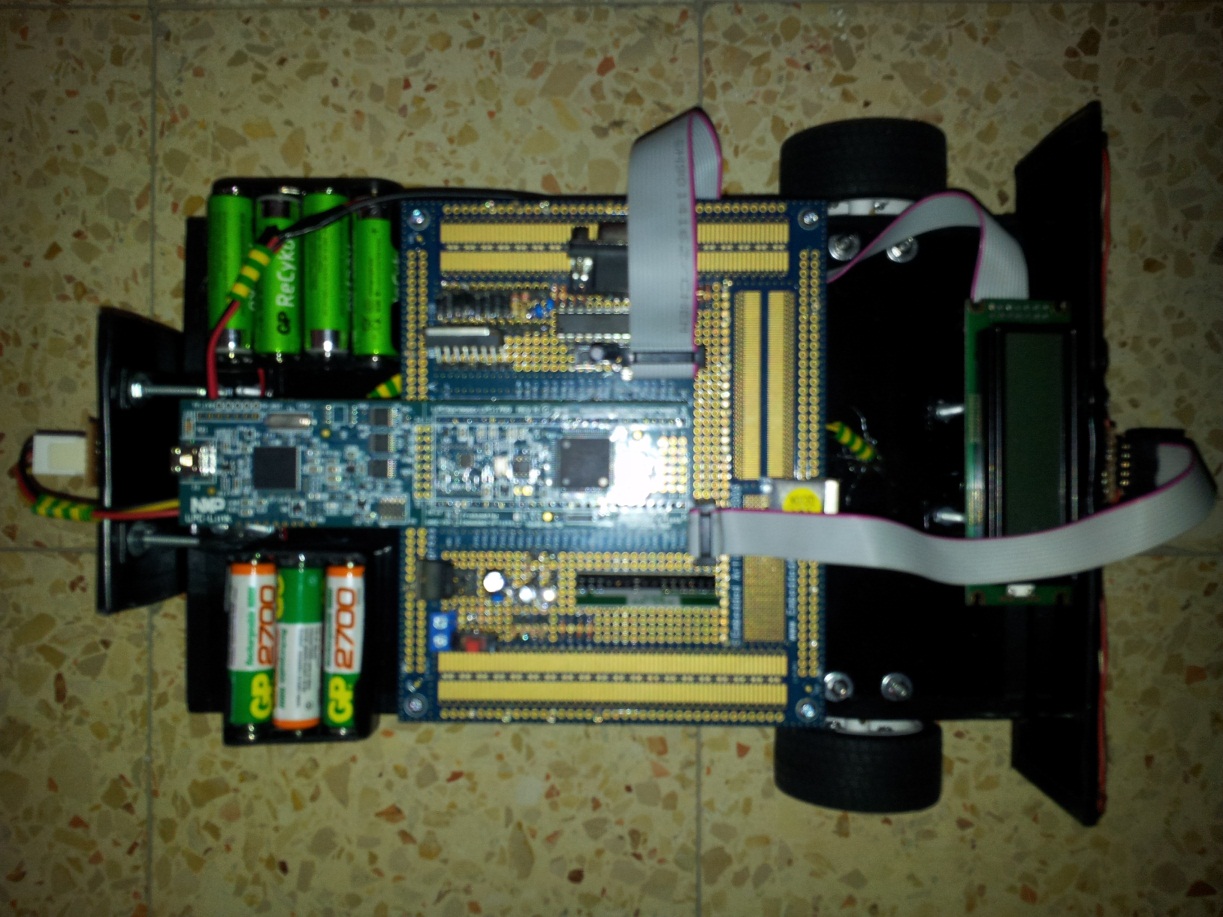
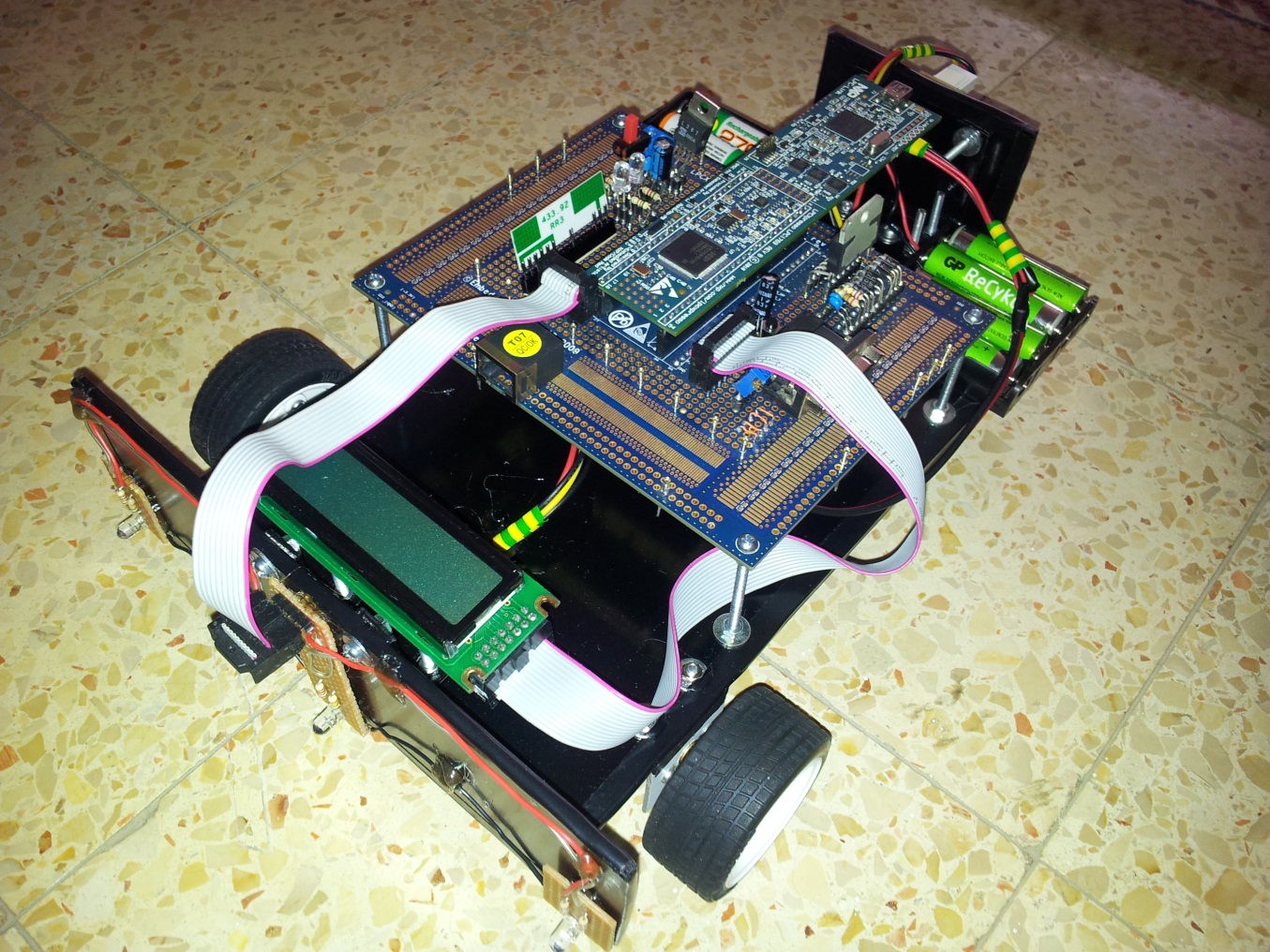
לניסוי השתמשתי הנגד משתנה עד 1MΩ, נגד של 1KΩ ופוטוטראנזיסטור tept5600. הניסוי נעשה במתח של 4.5v.



**פרק 7:**

**זיווד**





**פרק 8:**

**סיכום ומסקנות**

8.1 סיכום:

בפרויקט נבנה רובוט בעל אינטלגנציה מלאכותית עם יצירת התניה בין האור המוקרן עליו לבין שלט RF השולט בו תמיד.

8.2 מסקנות:

* ה-ARM Cortex-M3 LPC1769 מבזבז הרבה אנרגיה חשמלית, לכן הסוללות מתרוקנת מהר. ה-ARM ישמש הרבה יותר טוב כבסיס למערכת נייחת המוזנת ממערכת החשמל ולא לניידת המוזנת מסוללות כמו המערכת אותה בניתי.
* התנגדות הנדר נמצאת בפרופורציה ישרה לרגישות הפוטוטרנזיסטור, כלומר ככל שהתנגדות הנגד גדלה, רגישות הפוטוטרנזיסטור גדלה.

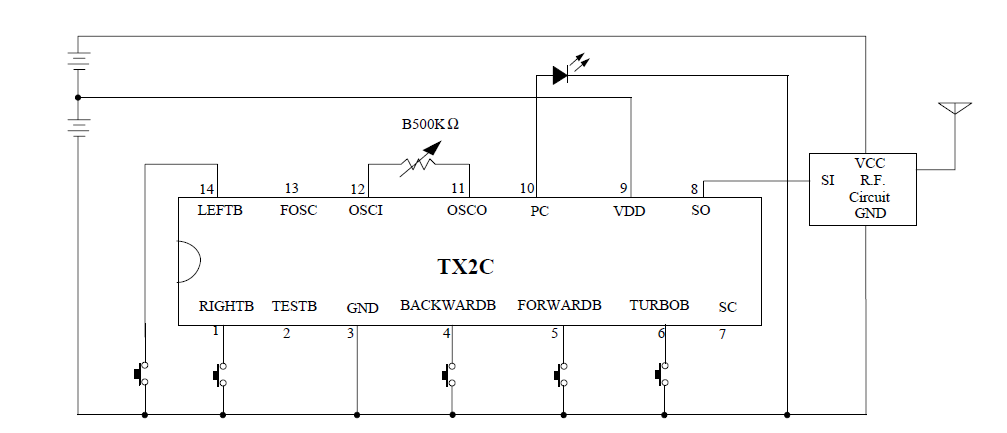
**פרק 9:**

**Reversing engineering**

9.1 מבוא ל-Reversing engineering:

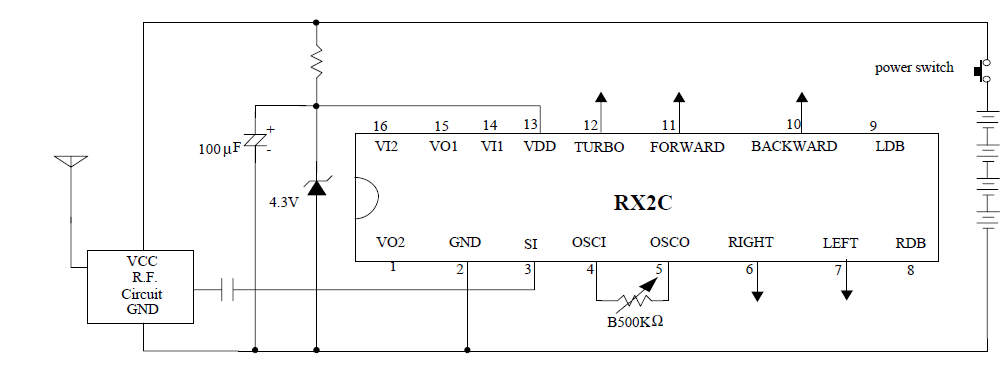
כתחליף לשידור דרך ה-UART בעזרת המקלט והמשדר( RT4 ו-RR3) משתמשים ב-RX-2 וב-TX-2, עליהם נעשה Reversing engineering.

9.2 משדר TX-2:



המשדר הינו משדר אות סיפרתי עם קוד שונה לכל לחצן. כאשר מתקבלת לחיצה משתנה התנגדות הפוטנציומטר וכך גם האות המשודר.

9.3 מקלט RX-2:



המקלט קולט את האות המשודר ומשנה את ההתנגדות של הפוטנציומטר בהתאם, לפי ההתנגדות המתקבלת הוא קובע את המוצא.

**פרק 10:**

**ביבליוגרפיה**

* האתר של NXP.
* ויקיפדיה.
* האתר של code red.
* פורום של NXP.

**פרק 11:**

**נספחים**

נספח א'

:ARM Cortex-M3 LPC1769

ה-LPC1769 הינו מעבד מבוסס ARM Cortex-M3 מתוצרת NXP בעל שלושה סוגי זיכרון: Flash ו-RAM.

למעבד חמישה פורטים ברי תכנות כל אחד לפעולה שונה בגודל 32bit.

את שאר התיאור ניתן למצוא בקבצים המצורפים בדיסק.

נספח ב'

מנועים:

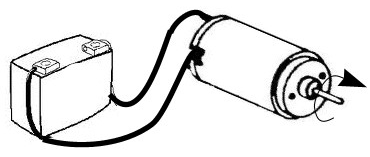
מנוע הינו בא להמיר אנרגיה חשמלית לאנרגיה מכנית, הוא מבוסס על עיקרון האלקטרומגנטיות שלפיו זרם חשמלי הזורם במוליך יוצר סביבו שדה מגנטי.

המנוע בנוי מרוטור וסטטור:

רוטור הינו ציר העובר בסטטור והוא חופשי להסתובב. עליו המלופפים סלילים.

סטטור הינו ליבה מוליכה אשר סביבה מלופפים סלילים והיא מקובעת.

כאשר מוזרם זרם דרך הסלילים הם מייצרים שדה חשמלי, כאשר נוצרים שני שדות חשמליים מנוגדים הרוטור מסתובב.



נספח ג'

LCD:

תצוגת גביש נוזלי בעלת 14 הדקים, התצוגה הבנויה ממספר מקטעים בצבעים שונים. ישנן מולקולות גביש נוזלי אשר עם שינוי המתח מסתובבים והפיקסל נראה.

אני משתמשת בתצוגה של 16X2. התצוגה מופעלת בעזרת תווי בקרה בגודל שמונה סיביות, וקודים ASCII לתצוגה.

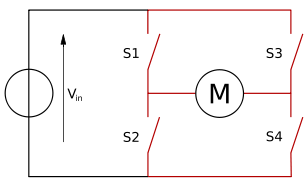
הדק E הוא הדק לאפשור פעולה של בתצוגה.

נספח ד'

גשר H:

זהו מעגל אלקטרוני המאפשר נפילת מתח בכיוונים שונים, מעגל זה יכול לשמש להרצת מנועי DC. הוא יכול להוות מגל משולב או להיבנות מרכיבים בדידים.

גשר H בעיקרו בנוי מארבעה מתגים, כאשר נסגרים המתגים S1 ו-S4 הזרם יזרום דרך העומס בכיוון אחד ואילו כאשר מתגים S2 ו-S3 נסגרים כיוון זרימת הזרם בעומס משתנה.

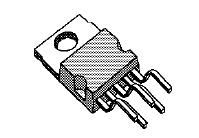


גשר H משמש בדרך כלל להחלפת כיוון הרצת המנוע, אך יכול לשמש גם לעצירתם בהתאם לטבלה הבאה לדוגמה הנ"ל.

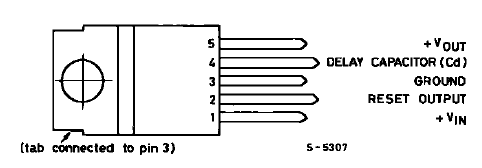
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| כיוון סיבוב מנוע | S4 | S3 | S2 | S1 |
| שמאל | 0 | 1 | 1 | 0 |
| ימין | 1 | 0 | 0 | 1 |
| עוצר | 0 | 0 | 0 | 0 |
| עוצר | 1 | 1 | 1 | 1 |

נספח ה'

L387:



זהו מייצב מתח עם נפילה נמוכה בעל חמש הדקים.

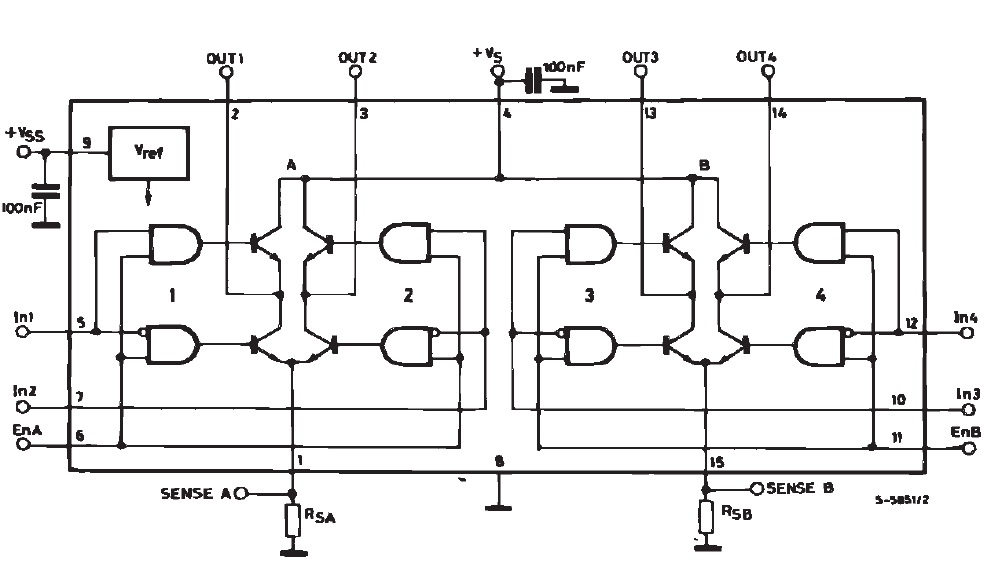


הוא מייצב את המתח בכניסה ל-5v ±4% במוצא.

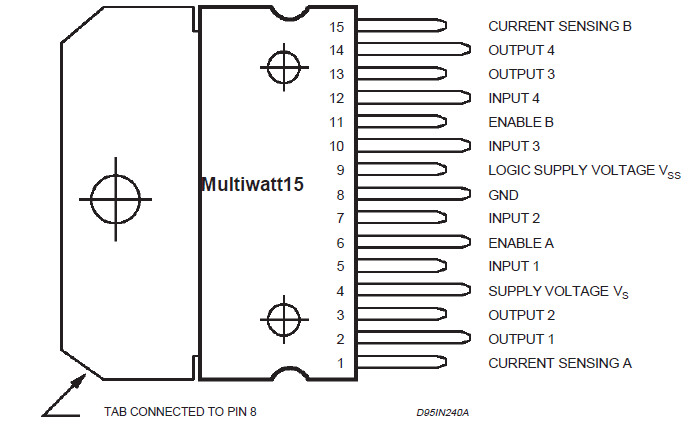
נספח ו'

L298n:

L298 הוא מעגל בעל 15 הדקים, שני גשרי H הבנויים בצורה הבאה:



L298 נראה כך:



לרכיב יסופק מתח לפעולת הרכיבים הלוגיים של המעגל (בין 4.5v לבין 7v, מתח טיפוסי 5v). המתח יסופק להדק 9 של הרכיב.

להדק 4 מסופק מתח ליצירת זרם אשר יפעיל את המנועים שזה מתח בין מתח במבוא בהדקי הכניסה ±2.5v לבין 46v.

הדק 8 מהווה אדמה משותפת של הרכיב.

נספח ז'

RT4-433.92:

זהו משדר עם אפנון AM עם גל הנושא בתדר של 433.92MHz. המשדר בעל 4 הדקים. האנטנה המתחברת אליו צריכה להיות באורך של רבע מאורך הגל של גל הנושא.

מתח ההזנה שלו הינו בין 2v לבין 14v, הוא מתחבר להדק מספר אחד של המשדר.

אורכה במינימלי של מחושב כרבע אורך גל של גל הנושא, והיא מתחברת להדק השלישי של המשדר.

נספח ח'

RR3-433.92:

זהו התקן קלט לתקשורת רדיו AM קולט בתדר 433.92MHz

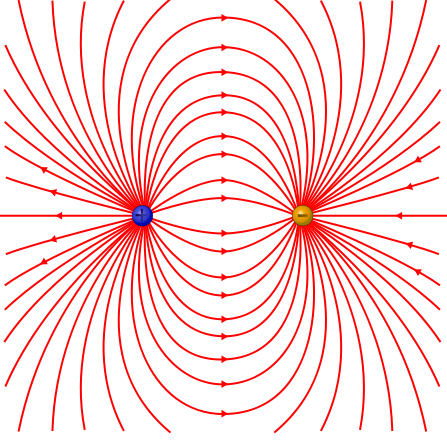
נספח ט'

אנטנות:

האנטנה הינה רכיב חשמלי המיועד ל

קיימים מספר סוגים של אנטנות:

אנטנת דיפול - אנטנה הבנויה משני מוטות הנמצאים בציר אחד. עקום הקרינה של האנטנה זו הוא כלל כיווני, במישור הניצב לאנטנה. אורך האנטנה הוא חצי אורך הגל.



אנטנת מונופול - בנויה ממוט מתכת, הינה מהווה מחצית מאנטנת דיפול. אורכה הינו רבע מאורך הגל, עקום הקרינה שלה הינו זהה לזה של דיפול.

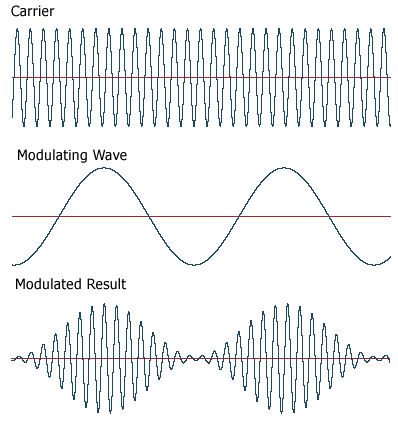
אנטנת יאגי - משמשת בעיקר לשידור תדרים נמוכים" היא מורכבת אלמנטים הנקראים דירקטורים ורפלקטורים, אלו אלמנטים הדומים לאנטנת מונופול ואנטנת דיפול מכופפת כלפי פנימה.



נספח י'

אפנון AM:

אפנון AM הינו שינוי באמפליטודת גל הנושא בתלות באמפליטודת גל המידע.



נספח יא'

נוירון:

נוירון הינו התא החשוב ביותר בגוף האדם אשר תפקידו לעבד ולהעביר מידע בצורת אימפולסים חשמליים לאורך גוף האדם. בגוף האדם ישנם יותר מ- 100000000 נוירונים.

תא זה בנוי משלוש יחידות עיקריות שהן:

אקסון- מעביר את המידע/פקודה לתא הבא אחריו.

דנדריטים- קולט מידע מהתא הקודם לו.

גוף התא- מעבד את המידע הנקלט ע"י הדנדריטים. הגוף הנוי מציטופלסמה וגרעין עטוף בממברנה.

סינפסה:

המרחב בין שני נוירונים המשמש להעברת האימפולסים בין הנוירונים.

היסטוריה:

בתחילת המאה העשרים המדענים ניסו להבין את פעולה המוח האנושי. בעקבות כך נוצרו שתי תאוריות האחת של סנטיאגו רמון אי קחאל הטוענת כי יחידות העיבוד הן תאים העלי גרעין בודד והן מתקשרות ביניהן דרך צמתים מיוחדים, ואילו השניה הינה של קמילו גולג'י האומרת כי כל יחידת עיבוד בנויה ממספר גרעיני תאים בתוך אותה ממברנה. שניהם קיבלו פרס נובל לרפואה ופיזיולוגיה על תרומתם לחקר מערכת העצבים.

בשנת 1943 McCulloch and Pitts פרסמו מאמר בו הם הגדירו את המושג "רשת נוירונים" ואת מודל פעולתו של הנוירון. בהמשך נוברט ווינר פרסם את עבודתו על קיברנטיקה בה הוא מתרגם את המערכות הביולוגיות המסובכות למודל מתמטי. ב1949 Hebb מציאה את האלגוריתם הראשון ללמידת הנוירון.

ניסוי פבלוב:

לכל בעלי החוליות קיימים אינסטינקטים מולדים כגון: לישון לאכול לברוח, ישנם אף אינסטינקטים נלמדים כפי שהוכיח פבלוב בניסויו.

לכלב ישנו אינסטינקט שכאשר הוא מבחין בדבר מה אכיל או מריח מוחו אומר ליצור ריר, פבלוב קישר את הגשת האוכל לכלב בצלצול פעמון לאחר מספר פעמים הכלב החל לייצר ריר עם צלצול הפעמון. כך הוכיח פבלוב את קיום האינסטינקט הנלמד.

תהליך הלמידה מתבצע על ידי שינוי מספר הקולטנים על כל דנדריט, ככל שהמידע מתקבל יותר מספר הקולטנים גדל תהליך זה מתבצע גם בהתמכרות, הגדלה או הקטנה של מספר הקולטנים הינו שינוי במשקלו של כל דנדריט. לפי כך עם נשנה את המשקל של כל אות מבוא נשנה את אות המוצא של הנוירון, שינוי המשקל של אות המבוא הינו תהליך הלמידה.

רשת נוירונים מלאכותית:

כללי:

לפתרון משימות רבות נדרשו להשתמש ברשתות נוירונים מלאכותיות, כגון: זיהוי תמונות, חיזוי הבורסה, תפעול רובוטים וכו'... רשתות הנוירונים מתחלקות לרשתות חד דרגתיות ורב דרגתיות, לדוגמה: XOR ניתן לממש רק ע"י רשת רב דרגתית.

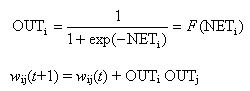
מודל הנוירון ברשת נוירונים מלאכותית מתבסס על תא העצב בגוף ועקרון פעולתו הינו קבלת סכום של כל אחד מהאימפולסים הנכנסים וכפולתם פי משקלן של הסינפסות, הסכום המתקבל נבדק בעזרת פונקציית ההפעלה.

לפי מודל McCulloch-Pitts יחידת נוירון מבצעת פעולת שער לוגי ע"י חיבור משוקלל של האותות והמשקלים במבואות ובמידה הערך המתקבל גדול או שווה לערך הסף במוצא יהיה '1' ובמידה ולא יהיה '0'. ככל שמשקלו של מבוא מסוים גדול יותר כך הקשר חזק יותר כלומר המבוא משפיע יותר.

u=\sum_{i=1}^{n} w_i x_i + w_0 x_0

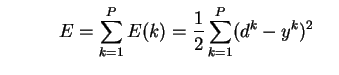
ישנן מספר שיטות למידה: למידה עם מדריך, למידה עצמית ולמידה משולבת. בניסוי של פבלוב בא לידי ביטוי למידה עם מדריך, מכיוון והפרויקט שלי מתבסס על ניסוי זה.

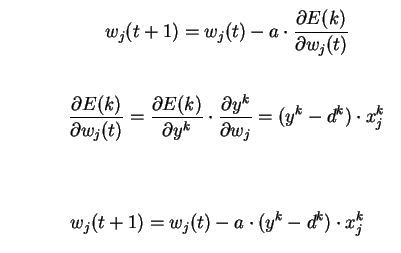
Hebb טען כי: לשם לימוד רשת נוירונים עלינו לשנות את משקל הסינפסות בהתאם לתיאום בין האותות המתקבלים בכניסה כמו ברשת נוירונים פיזיולוגית בה נוצרים אסוציאטיביים. כלומר רשת הנוירונים תשנה את המשקלים עד למצב הרצוי.



לימוד רשת נוירונים מתבצע בשיטת ווידרו הופף עם נוהל הדלטא שלו.

החישובים נעשים בעזרת הפונקציות הבאות:





מימוש שארים לוגיים ע"י נוירונים:

