مدرسة يمة
الثانوية
الزراعية

בי"ס
תיכון
חקלאי ימה

המגמה להנדסת אלקטרוניקה ומחשבים

מטלת הסיום

ארדוינו למתחילים תיאוריה וניסויים

עבודה זו מוגשת מטעם שני מגישים לשם עמידה בדרישות הקורס **כלים למורה המנחה עבודת גמר באלקטרוניקה** לצורך קבלת גמול ההשתלמות.

לכבוד : מר אהרון שחר

:מר יגאל שפירא

מגישים

תאריך : **29 לפברואר 2016**

הקדמה

תופעת הארדוינו הולכת ותופסת תשומת לבם של מגוון רחב של אנשים בעולם, ומספקת הזדמנויות חדשות ומיידיות למי שמעולם לא החזיק מלחם בידו או כתב שורה אחת של קוד בשפת תכנות. האקרים לחומרה ולמפתחי דפי אינטרנט, חובבי רובוטיקה עד למומחי התקנת תכניות, סטודנטים וסטודנטים למוזיקה: את רובם אפשר למצוא בקהילת הארדוינו. הרבגוניות של הפלטפורמה כוללת חומרה ותוכנה, בשילוב כלים לפיתוח, תפסה זה מכבר את דמיונם של עשרות אלפי מפתחים.

תחילת עבודה בארדוינו היא קלה יחסית וזאת בהתאם לכוונת הצוות האחראי לפיתוח הארדוינו. בין היתרונות הרבים של הארדוינו היא כמות המידע העצומה והזמינה בין בספרים או ברחבי האינטרנט.

הארדוינו הינו שילוב של שלושה אלמנטים קריטיים: חומרה, תוכנה, וקהילה. לשם השגת התועלת המרבית מהארדוינו, המשתמש נדרש להבנה בסיסית של כל שלושת המרכיבים הללו. יש לציין כי ועבור מרבית האנשים, האתגר הגדול ביותר מבין שלושה האלמנטים יהיה הבנת החומרה הרלוונטית. אחד ההיבטים המרכזיים להצלחה הגדולה של הארדוינו מתבטא בקהילה הצומחת סביבו בשל האופי הפתוח של הן של תוכנת ארדוינו והן של החומרה. הווה אומר, קל להסתגל בעבודה עם התוכנה והחומרה לצרכי המשתמש, ולאחר מכן לתרום את מה שעושים ומפתחים בחזרה לפרויקט ארדוינו.

אחד הדברים היפים בעיצוב פרויקטים בסביבת הארדוינו הוא שחלק גדול מהרכיבים האלקטרוניים הינם ברמה נמוכה ומתאימה לידע שהתלמידים צברו במהלך לימודם במקצוע אלקטרוניקה ומחשבים. במידה והתלמיד בכל מעט מיומנות בסיסית בתחום האלקטרוניקה, הדבר רק ישרתו היטב בבניית פרויקטים פשוטים וגם יאפשר לו להבין את מה שקורה מאחורי הקלעים.

כאמור לעיל, הארדוינו הינו כרטיס אלקטרוני המבוסס על מיקרו בקר עם מפתחי קלט / פלט פשוטים (I/O), וסביבת פיתוח המיישמת את עיבוד ההוראות שנכתבו לבקר. ניתן להשתמש בארדוינו לפתח פרויקטים אינטראקטיביים עצמאיים או יכול להיות מחובר לתוכנה במחשב שהמפתח/תלמיד מפתח.

במסגרת עבודה זו, נסקור ונאפיין את מבנהו החומרתי של אחד מלוחות הפיתוח של ארדוינו. בנוסף, אנו נציג את תכונותיו העיקריות ונסקור את חבילת התוכנה IDE המאפשרת תכנותו ביעילות. עיקר העבודה מתרכז בהצגה, והבנה של מספר רעיונות לפרויקטים משולבי חומרה ותוכנה בעזרת לוח ארדוינו.

מבוא

כרטיסי הפיתוח של הארדוינו זמינים בכמה דגמים שונים כמו :

Arduino Uno , Arduino Fio , Arduino Nano ו Arduino Mega.

השוני בין הדגמים הוא במספר הכניסות והיציאות בחומרה והלכה למעשה קובעות את מספר הרכיבים החיצוניים שניתן לשלוט עליהם בעזרת הארדוינו, וכן את מספר החיישנים שאפשר לחבר לכרטיס, כמו כן סוג המיקרו בקר ומהירותו משתנה מערכה לערכה.

עבודה הזאת מתמקדת בכרטיס הפיתוח ((Arduino Uno שזהו דגם בסיסי ויציב המהווה את הגרסה העדכנית ביותר של כרטיסי הפיתוח. כרטיס זה יצא לאור בחודש ספטמבר 2010.

להלן, המונח "לוח", "כרטיס", "ערכה" או "ארדוינו" יהיה מכוון לכרטיס Arduino Uno, אלא אם צוין אחרת.

איור 1: כרטיסי הפיתוח Arduino Uno

Arduino Uno הוא לוח אלקטרוני המבוסס על מיקרו בקר של ATMEL שמספרו ATmega328. הלוח כולל מספר רכיבים בסיסיים המבטיחים פעולה תקינה למיקרו בקר. כך הגביש בלוח מספק שעון למיקרו בקר כדי מנת לאפשר לו לפעול במהירות הנכונה. בנוסף על הלוח מוכלל מייצב מתח לינארי של 5 וולט, המספק לכרטיס מתחי ההזנה. לכרטיס ישנו גם חיבור USB המאפשר חיבור בין מחשב שולחני לכרטיס הארדוינו לשם טעינה/צריבה של קוד המכונה של התוכנית לתוך הזיכרון.

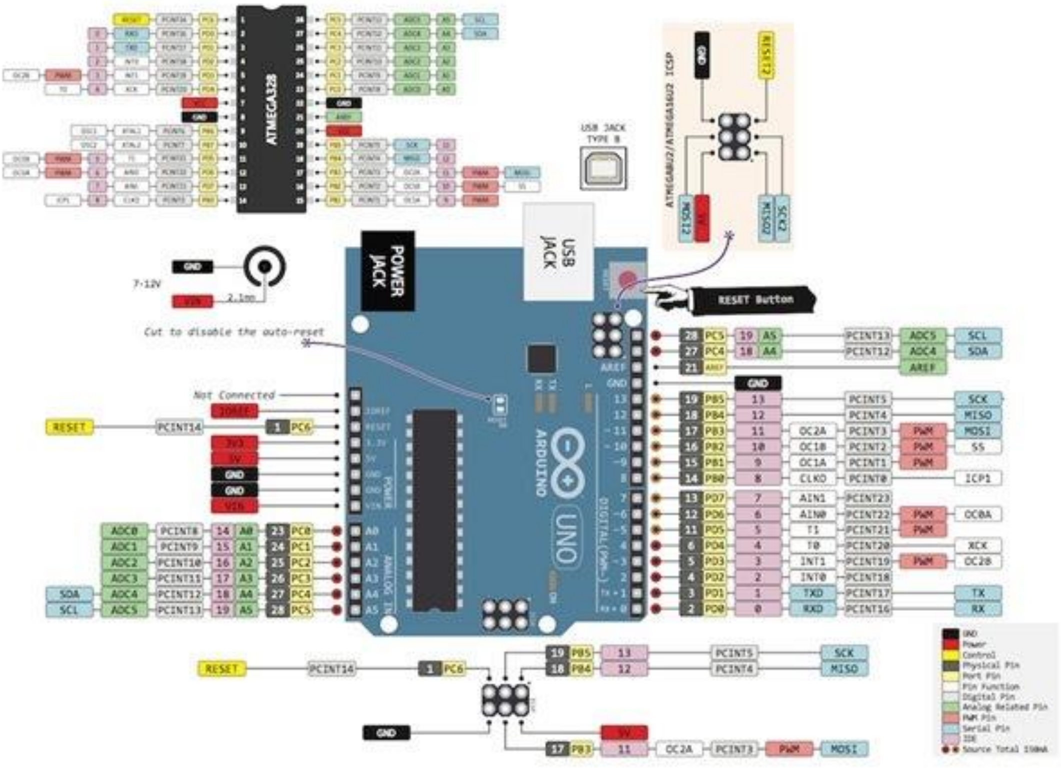
פרק -1 מבנה לוח ארדואינו ומאפיינים חשובים

תיאור הדקי לוח הארדוינו ואפיון כללי.

אפיון כללי

הלוח יכול להיות מופעל על ידי מתח המסופק מיציאת ה USB של המחשב , או ספק מתח חיצוני של 9 וולט. בזמן הפעלת הכרטיס, במידה ואין ספק מתח המחובר לשקע בכרטיס, החשמל יסופק מה-USB,. עם חיבור ספק כוח, הלוח משתמש בספק הזה באופן אוטומטי.

תדר העבודה של ה Arduino Uno הוא MHz 16 המסופק באמצעות גביש קראמי.



איור – 1: מבנה סכמתי של לוח הארדוינו

הדקי הארדוינו

לארדוינו 14 הדקים I/O דיגיטליים (הדקים 0 - 13)

אלה יכולים להיות כהדקי קלט או פלט, זאת בהתאם להגדרתם בקוד התוכנית הרלוונטית.

6 כניסות אנלוגיות (הדקים A0 - A5)

הדקים אלה מיועדים לתפקד ככניסות אנאלוגיות. כל הדק מסוגל לקבל ערך אנלוגי (לדוגמא, מתח מקריאת חיישן) ולהמירו למספר בין 0 ל 1023.

6 הדקים אנלוגיים (הדקים 3 , 5, 6, 9, 10, ו – 11)

אלה הם למעשה ששה מההדקים הדיגיטליים שניתן לתכנתם כפלט אנלוגי באמצעות הקוד הנצרב לבקר.

1.1 מבוא למיקרו בקר

בסעיף הזה נתמקד הן בחומרה של לוח הארדווינו והן במיקרו-בקר המהווה גורם מרכזי במערכת. השימושיות של המיקרו-בקרים במגוון רחב של רכיבים/מוצרים בחיינו הינה סיבה עיקרית לכך שהתלמידים צריכים ללמוד אודותם. עם זאת, ישנו גם סיפוק רב בעיצוב מעגלים אלקטרוניים תוך שימוש במיקרו-בקרים. המיקרו-בקרים משובצים ברוב המכשירים וההתקנים סביבנו. כל מי שרוצה הבנה אמתית על אופן עבודתם ופעולתם של כמה מוצרים מודרניים - מכוניות, טלפונים ניידים, צעצועים, מכשירים ביתיים וכו' -צריך לרכוש הבנה בסיסית במיקרו בקרים.

בעבר נדרש זמן לא מועט כדי לסיים ולהשלים מעגל אלקטרוני. כיום, התלמידים יכולים לקבל דברים מובנים ולחסוך זמן ניכר. בנוסף זאת מצטרפת הנאה רבה בתהליך העיצוב, הבניה, ההרכבה, כתיבת הקוד והפעלת המוצר המוגמר. לימודים בשילוב הנאה מאתגרת את התלמיד ומביאה למוטיבציה לימודית גבוהה. אנו יכולים להשתמש במיקרו-בקר במספר אינסופי של דרכים כדי לשפר את העבודה שלנו, הלמידה שלנו, התחביב שלנו, או את החיים החברתיים שלנו. הערך הגדול של מייקרו הבקרים נובע מזה שהם מאפשרים לנו להרחיב את היתרונות של המחשוב בעולם האמתי.

למה צריך מיקרו-בקר?

מחשבים שולחניים (מחשבים אישיים) הם מצוינים, הם באמת פלא של זמנינו. בתיאום עם האינטרנט, מחשב שולחני שאתנו קונים בחנות יכול לעשות כמעט כל דבר שאנו רוצים עם מידע דיגיטלי. מחשב שולחני הוא למעשה מעבד ומאחסן המידע באופן אמין בעולם מרושת. הוא יכול לעשות הרבה דברים בבת אחת (למשל, לבדוק הודעות הדואר האלקטרוני, ולעשות בדיקת וירוסים בזמן שאנחנו גולשים באינטרנט), כי הוא פועל במערכת הפעלה מורכבת, והוא מסוגל לבצע מספר משימות בו-זמנית. מה שאנחנו רואים על המסך הוא רק קצה הקרחון של העבודה המתרחשת בתוך לבה של המכונה.

במחשב שולחני מודרני יש מעבד מרכזי הפועל בקצב המתקרב ל- 3 מיליארד מחזורים בשנייה, ויש מעבדים רבים עם ארכיטקטורה מרובת ליבות, כלומר הם מסוגלים לעבד שניים, או אפילו ארבעה, סדרות של זרמי הוראות במהירות זו, בו זמנית. במונחי מחשוב, זה מכשיר עתיר משאבים. מפתחי המחשבים הראשונים לא היו יכולים לייחל לו יותר.

המחשב השולחני הוא מכונה לכל דבר לכן הם יחסית יקרים. את היכולות של המחשב השולחני הן עכשיו כל כך גבוהות שאנחנו יכולים להשתמש בו לכל אחת מהמשימות שהוזכרו קודם ולנהל את המידע ללא שום בעיה, אבל במצב ברירת המחדל שלו הוא בעצם דל מאד בממשק להתחברות עם העולם האמתי החיצוני, וכאן בדיוק מתהווה מקומו של המיקרו-בקר.

מקומו של המיקרו-בקר

האם ברצוננו לקבל הודעה כאשר התנור שלנו סיים את הבישול? האם רוצים לשלוט בחוכמה על המהירות של מנוע המסתובב מהר? האם רוצים ליישם מערכת בקרה להפקת חשמל מאור השמש על גג הבית?. כל מכונה מודרנית יכולה לספק תשובה לשאלות לעיל, אך במחיר של התוספות המתאימות. ועדיין יהיה למכונה הרבה כוח עיבוד אדיר לא מנוצל, זאת בנוסף למחירה הגבוהה, ועוד לא דיברנו על צריכת ההספק שלה. מכאן, ברור שבהרבה יותר משימות מחשוב שגרתיות לא צריכים לזה את העוצמה שבכוח העיבוד של המחשב השולחני.

מצד אחד, בהשוואה למחשב השולחני המיקרו-בקר נראה קטן ועלוב. המעבד שבו איטי בהרבה יחסית להמחשב השולחני. הוא חסר קיבולת זיכרון גדולה לעיבוד הנתונים. לרוב הוא חסר מנגנון מובנה לתמיכה בכוננים קשיחים, ואינו מתחבר לאינטרנט בפשטות.

מצד שני ניתן לרכוש מיקרו-בקר במחיר זול יחסית ולתכנן בעזרתו מערכת בקרה שלמה בעלות מאד קטנה. למיקרו-בקר ישנן מספר רב של כניסות ויציאות המותאמות לשימוש והתחברות עם מכשירים

בעולם אמיתי, ועם קצת מאמץ, הוא יכול לתקשר עם המחשב השולחני דרך היציאות הטוריות או USB.

לסיכומו של דבר, במחשב השולחני נשתמש לדברים גדולים ומטרות כלליות:אינטרנט, דואר אלקטרוני, הורדה ונגינת וידאו, עיבוד תמלילים, הדפסת חומר, מסרים מיידים, רשתות חברתיות, בניית ספריות תמונות ומוסיקה, עריכת תמונות. לעומת זאת, נשתמש במיקרו-בקר כיחידה מחשב עצמאית המבצעת משימה מסוימת ומוגדרת מראש. כמו שליטה בכמה אורות, מדידת הטמפרטורה, והעברת המידע והתוצאות למחשב השולחני.

מערכת מבוססות מיקרו שולטות ומבקרות הרבה מכשירים בעולם האמיתי שלעיתים נקראות מערכת חכמה, כי הן מופעלות ונשלטות על ידי תוכנה דבר שמאפשר למערכת למידה של הסביבה שבה היא מופעלת בה וגם הסתגלות בגבולות ותחום שנקבע מראש, וכך להגיב בהתאם ללמידה שנרכשה.

החיבור בין מחשב שולחני ומיקרו-בקר

מערכות מיקרו יכולות לקבל הוראות מהמחשב השולחני להפעיל ולכבות מכשירים אחרים. אך מערכת מיקרו לא חייבת להיות מחוברת למחשב כדי שתעבוד. היא יכולה לתפקד כמו מחשב בלתי תלוי ולבצע משימה פשוטה ומוגדרת מראש, באופן עצמאי וחכם.

1.2 משפחת ATMEL

בסעיף זה נציג ונתמקד בשבב הקטן שבלב הArduin- כרכיב בודד בתוך מערכת גדולה יותר. בתוך השבב הקטן יש עולם ומלואו של יחידות ומרכיבים.

על מנת לנצל את הפוטנציאל הגלום ב-Arduino בבניית פרויקטים מבוססי חומרה, ישנו צורך להבין טוב יותר איך השבב הקטן עובד ומתפקד. בעיקר יש להבין מה הן מגבלותיו, מה הוא יכול לעשות, וכיצד לגרום לו לעשות את זה. הכל מתחיל בתוך השבב הזה. ושוב יודגש, ה- Arduino הוא "רק לוח פיתוח של המיקרו בקר AVR".

משפחת AVR של bit-8 כוללת מגוון רכיבים המספקים תכונות מובנות מיוחדות ושונות. הרכיב ATmega328 שבלב ה- Arduino המודרני שייך למשפחת ATmega. באופן דומה, הרכיב ATmega2560 שהוא שבב המחשב של הארדוינו מגה " Arduino Mega ". ישנה גם משפחה קטנה ומשפחת AVR קלאסית. בנוסף, Atmel הציגה את המשפחה המשופרת XMega, עם מהירויות שעון גבוהה יותר ויותר פונקציונליות. Atmel הפיקה גם גרסת 32 סיביות בקו היצור של ה- AVR, בשםAVR32 .

כמו שהזכרנו מיקרו-בקר הנו מחשב מזערי. כדי שמערכת תוגדר כמחשב, עליה לכלול את שלושת המרכיבים הבאים.

זיכרון (RAM) להפעלת התוכנית

מעבד (CPU) שמעבד כל הנתונים ומפעיל הפורטים בהתאם

יחידות כניסה/יציאה (I/O) מקשרת הבקר עם העולם החיצוני.

כל מיקרו-בקר מורכב מכמה יחידות עיקריות:

פורטים (port) הקולטים מידע ומוציאים אותו החוצה.

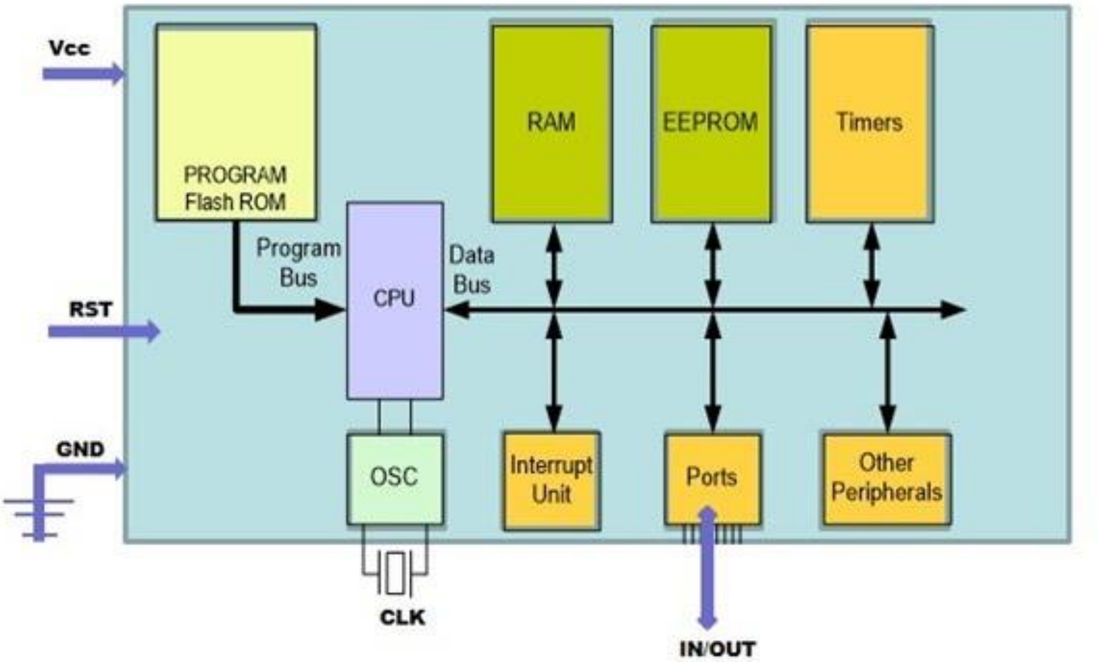
רגלי בקרה: איפוס, הזנת מתח,שעון.

מעבד (CPU)

זיכרון (RAM)

בקרת זיכרון, ספק מתח, בקרת איפוס, שעון ותזמון.

איור-2 מציג תרשים מלבני פשוט של AVR 8-bit



איור-2: תרשים מלבני פשוט של AVR 8-bit

במרכז AVR 8-bitיש את ליבת ה AVR המכילה את יחידת העיבוד המרכזית (CPU) וכל המרכבים החיוניים להפעלת ה AVR, כמו היחידה המתמטית (ALU) שמבצעת חישובים מתמטיים ולוגיים.

מקורות מתחי ההזנה

כמו כל רכיב אלקטרוני, משפחת ATmega דורשת מתחי אספקה כדי לפעול. מתח האספקה של הרכיבים נע בין V5.5 ל V4.5 מתח ישר (DC) . ישנם שני מעגלי כוח שונים בתוך את שבבי ATmega. אחד מהן הוא לאספקת המתח הדיגיטלי, המכונה Vcc . זהו המתח שמספק את ליבת המעבד, זכרונות, וציוד ההיקפי הדיגיטלי. המתח השני הוא אספקת מתח לחלקים האנלוגיים של השבב, כולל ADC ומשווה אנלוגי (AC). הדק האספקה למערכת האנלוגית נקרא AVcc. שני המתחים Vcc ו AVcc חייבים להיות מסופקים מאותו מתח.

מתח ייחוס אנלוגי (AREF)

מתח זה מחובר בתוך השבב לכניסת הייחוס של הממיר מאנלוג-לדיגיטל (ADC). מתח הייחוס קובע את הקצה העליון של טווח מדידת המתח האנאלוגי בכניסת הממיר. הקצה הנמוך של טווח מדידת המתח הינו, אדמה. הערך המספרי המתקבל מההמרה הוא יחסי בין ערך המתח בכניסת הממיר למתח הייחוס האנלוגי.

באמצעות שליטה בתוכנה, ה ADC יכול לבחור בין מספר מקורות מתח ייחוס אנלוגיים שהינם: מקור מתח חיצוני המחובר להדק AREF, מקור אספקת המתח האנלוגית, AVcc, או מתח יחוס פנימי.

מתח הייחוס הפנימי משתנה משבב לאחר, כך ב- ATmega328, הוא V1.1. וב- ATmega2560, ניתן לבוחרו מבין המתח V1.1 או V2.56.

אם משתמשים ב AVcc או מתח ייחוס פנימי, צריך לנתק את הדק AREF חיצונית על ידי חיבורה לאדמה דרך קבל כדי להגדיל את יציבות מתח הייחוס.

אפשרות נוספת היא להשתמש במקור מתח חיצוני כמתח ייחוס. באפשרות הזו נעשה שימוש כאשר נדרשת נקודת ייחוס מדויקת יותר, או נדרש מתח ספציפי. במקרה זה יש להיזהר ואסור לקבוע בתוכנה אף אחת מאפשרויות הייחוס האחרות, אחרת יוצא שאנו יוצרים חיבור ישיר בין מקורות המתח הפנימיים והחיצוניים.

אתחולRESET

הדק ה- RESET מספק מנגנון לאיפוס /הפעלה מחדש של המיקרו-בקר. פונקציית RESET מתרחשת כאשר הדק זה נמצאת במצב נמוך. הדק מספר 30 ב- ATmega2560 מהווה כניסה ייעודית המוקדשת לפעולת האיפוס. ב ATmega328, הדק PC6 המרובבת עם קלט/פלט כללי משמשת גם לאיפוס. הפונקציה של פין זה נקבעת בתכנות באמצעות ביטול/האתחול (RSTDISBL reset disable).

XTAL1 ו XTAL2

XTAL1 ו-XTAL2 הינם הדקי קלט/פלט ממגבר מתנד מהפך פנימי. XTAL1 משמש הדק הכניסה למגבר ו- XTAL2 משמש יציאת המגבר. להדקים אלה מחברים את הגביש החיצוני שנועד לקבוע את תדר העבודה. ניתן להשתמש בהדק XTAL1 גם כקלט למעגל השעון הפנימי במידה ואות שעון חיצוני

זמין.

ל- ATmega2560 ישנם שני הדקים המוקדשים ל XTAL1 וXTAL2 . לעומת זאת ב- ATmega328, הדקים אלה מרובבים עם הדקי הקלט/פלט הכלליים, PB6 ו-PB7 . הבחירה בין תפקוד ההדקים כקלט/פלט או הדקי שעון הזנה למערכת, נקבעת בתכנות על ידי קביעת תצורת הדק בחירת השעון "CKSEL0-3". כאשר נבחרה האפשרות של גביש או מתנד קראמי, PB6 וPB7 אינם יכולים כבר לשמש למטרות כלליות של קלט/פלט למערכת.

מקורות שעון

התזמון של כל הפונקציות הפנימיות בתוך ה Arduino נשלט על ידי שעון המערכת. מעגל Arduino סטנדרטי עושה שימוש בגביש קוורץ חיצוני או מעגל תהודה קרמי המספק את התדר היסודי שמניע את כלל המערכת. ב- AVR משולב מתנד פנימי שהתדר שלו נקבע על ידי גביש או מעגל תהודה חיצוני. ל AVR יש גם את האפשרות להתחבר לשעון חיצוני או מתנד RC אך זו אפשרות שאינה מאפשרת שעון מדויק בהשוואה לגביש קוורץ, ואף לא למעגל תהודה קרמי.

מרחבי כתובות

AVR מאפשרת גישה למספר מערכים של זיכרון והתקני קלט/פלט. ארכיטקטורת ה- AVR מבוססת על ארכיטקטורת הרווארד, שבה הנתונים והפקודות מאוחסנים בזיכרונות פיזיים נפרדים, זאת בניגוד לארכיטקטורת פון-נוימן, שבה שני הזיכרונות מתערבבים, חופפים, וניתנים להחלפה.

זיכרון התכנית

תכנית המחשב מקודדת לשפת המכונה אמורה להיות מאוחסנות בזיכרון הפקודות (תכנית) על מנת שהמעבד יוכל לקרוא ולבצע אותה.

ברכיבי AVR, זיכרון זה מיושם כמערך זיכרון פלאש של 16 סיביות. לטכנולוגיית זיכרון הפלאש המשמשת במוצרים של Atmel AVR, כמו גם ברכיבי מוליכים למחצה אחרים' יש את כל היתרונות של אי-נדיפות המידע המאוחסן עם יכולת לתכנתה מחדש בקלות, ללא צורך בסיוע חומרה מורכבת ומכשירים יקרים.

תוכנית ה- Arduino מנצלת את העובדה שרכיבי ה AVR הם בעלי תכונות עצמאית לזיכרון הפלאש ונעזרת בחלק הקוד המכונה " Bootloader " .

ה- Bootloader היא תוכנית קטנה ביותר כ-) 100(byte שבשלב הראשון יש לצרוב אותה למיקרו-בקר באמצעות צורב רגיל. בעת פעולת Reset (מתבצעת עם הזנת מתח לבקר) המיקרו-בקר מתחיל לקרוא פקודות מזיכרון הפקודות' החל מכתובת התחלתית מסוימת (בד"כ מתחילת הזיכרון). הוא מבצע את הפקודות לפי הסדר שבו הן מופיעות בזיכרון. במקרה בו יש הפנייה לכתובת אחרת, המעבד מבצע קפיצה לכתובת הזיכרון אליה מתייחסת ההפניה וממשיך בביצוע מהכתובת החדשה.

עקרון הפעולה של ה Bootloader הוא לכתוב בשורות הראשונות של הזיכרון פקודה שתגרום להפניה לאזור הזיכרון בו נמצאת התוכנית של ה-Bootloader . התוכנית עצמה, מנסה בתחילה לתקשר עם המחשב באמצעות הממשק הטורי) בחלק מהבקרים ישנה אפשרות גם (USB . במידה ונוצרת תקשורת שמתאימה לפרוטוקול אז ניתן להעביר מהמחשב באמצעות (תוכנה מסוימת או Terminal) קוד למקרו-בקר דרך תקשורת זו. תוכנית ה Bootloader מקבלת קוד זה ורושמת אותו לאזור אחר בזיכרון. לאחר סיום פעולת ה- Bootloader התוכנית יכולה לעבור לאזור הזיכרון בו נמצא הקוד החדש ולבצע אותו. במידה ובעת ריצת ה- Bootloader לא נוצרת תקשורת עם המחשב, התוכנה עוברת לבצע את התוכנית כרגיל.

זכרון הנתונים

זהו זיכרון המשמש לאחסון ערכי משתנים ונתונים המשתנים במהלך ביצוע התכנית במיקרו-בקר. זיכרון זה מיושם ב AVR כזכרון סטטי, בעל גישה אקראית (SRAM). מבחינה טכנית, כל מערכי הזיכרון בתוך המיקרו הם בעלי גישה אקראית, ואפשר לפנות ישירות לכל תא זיכרון במרחב מערך הזיכרון. מערך הזיכרון מבוסס על תאים סטטיים כל תא הוא סיבית אחת ולא דורש אות שעון כדי לשמור את התוכן שלהם, בניגוד לתאי זיכרון מסוג RAM הדינמיים הדורשים ריענון באופן מחזורי.

כידוע, זיכרון SRAM הוא זיכרון נדיף השומר על הנתונים בתוכו, רק כל עוד כוח חשמלי מחובר למערכת. כאשר מנתקים המערכת מהחשמל המידע השמור בתוך הזיכרון נעלם ותוכן הזיכרון הוא בלתי מוגדר. על כן, לא ניתן להניח כי תוכן ה SRAM נקבע לערך מיוחד (כמו הכל האפסים) לאחר חיבור מתח חשמלי למערכת.

ב- ATmega328 מכיל KB2 של זיכרון SRAM וה- ATmega2560 מכיל KB8 של זיכרון SRAM.

אוגרים

לכל שבב AVR על סוגיו ישנם אוגרים למטרות כלליות רחבות. ישנם 32 אוגרים, ממוספרים R0 לR31 . רוב ההוראות האריתמטיות והלוגיות יכולות לקרוא ולכתוב ישירות לאוגרים. לעיתים קרובות במחזור שעון אחד. לשישה מהאוגרים יש תפקיד מיוחד. הם יכולים לייצר שלושה קבוצות של 16 bit כדי לשמש מצביעים על מרחב הנתונים.

EEPROM

זיכרון EEPROM הוא זיכרון לקריאה בלבד הניתן למחיקה חשמלית. הוא מאפשר כתיבה ע"י צריבת ערכי משתנים תוך כדי כתיבת התוכנית. הבקר ATmega328 מכילKB1 של KB4.

ולמיקרו בקר ATmega2560 יש EEPROM בגודל KB4.

ה- EEPROM משמש זיכרון עזר מסוג Flash, לשם שמירת נתונים גם לאחר הכיבוי. ניתן לצרוב את הנתונים במהלך כתיבת התוכנית.

להלן סיכום הנקודות החשובות על זיכרון זה:

- מאפשר לבצע שמירת נתונים תוך כדי כתיבת תוכנית.
- שמירת נתונים מתבצעת ע"י צריבה במהלך עבודת המיקרו.
- ניתן לקרוא את הנתונים מהזיכרון במהלך התוכנית ולהשתמש בהם.
- הפניה אליו מתבצעת בעזרת אוגרים מיוחדים .

פרק 2 הכרת סביבת העבודה עם ערכת הפיתוח ("לוח ארדוינו")

כללי

כאשר ערכת ה- Arduino זמינה ניתן להתחיל וללמוד את מאפייני התוכנה של סביבת העבודה המותקנת במחשב האישי, ואשר בעזרתה מתפעלים את רכיבי החומרה ברמת התוכנה בערכת הפיתוח.

פרק זה מציג את סביבת העבודה ומסייע לקורא להבין וללמוד את סביבת הפיתוח של ערכת ה-Arduino. מעבר לכך, הפרק מביא לקורא את המידע הנדרש כדי להגדיר את סביבת עבודה באופן נכון לתחילת העבודה.

דרישות המינימום לעבודה תקינה הן: מחשב אישי עם חבילת IDE מותקנת, ערכת Arduino, כבל USB לחיבור ה- Arduino למחשב הביתי.

תוכנה ל Arduino

לוח ה- Arduino הוא רק לוח עם כמה רכיבים אלקטרוניים מולחמים עליו. כמו כל מחשב אחר, המיקרו-בקר צריך לקבל סדרת הוראות בשפת מכונה כדי לבצע עבודה שימושית/תכליתית. להשגת מטרה זו נדרשת תוכנת פיתוח ייעודית.

Arduino סיפקה את התוכנה והכלים החיוניים לכתיבת תוכניות וקידודן לשפת מכונה המתאימה להרצה בקושחה של הארדוינו. חבילת תוכנה זו זמינה להורדה בחינם באתר האינטרנט של חברת Arduino.

החבילה מספקת את כל הדרוש עבור תכנות ה Arduino , כולל מספר תוכניות דוגמה או סקיצות שמדגימות כיצד לחבר את הלוח, וכיצד הוא מתקשר עם רכיבים נפוצים , כגון LED, LCD, ועוד מספר חיישנים נפוצים. אי לכך, בסעיף זה נדון בהורדה, התקנה, ובדיקת התוכנה באמצעות IDE.

תוכנת ה Arduino זמינה עבור כל מערכות ההפעלה, Windows Mac OS X, ו Linux עם גרסה מתאימה לכל מערכת הפעלה.יש לוודא שמתבצעת הורדה לגרסה הנתמכת במערכת ההפעלה המותקנת על המחשב האישי שעושים בו שימוש.

סביבת הפיתוח נקראת(Integrated Development Environment) IDE שפרושה סביבת פיתוח משולבת. זו היא תכנית מיוחדת הפועלת במחשב האישי ומאפשרת כתיבה הוראות בשפה פשוטה ללוח הArduino . מערכת הפיתוח IDE זמינה להורדה באתר:

. www.arduino.cc/en/Main/Software

2.1 מושג הסקיצה ותהליך הפיתוח

ב-"עולמו" של Arduino, בלוק של קוד נקרא סקיצה " "Sketch, כאשר סקיצה מספקת ל- Arduino רשימה של הוראות והArduino- מוציא לפועל את הרעיון ששרטטנו לו. מכאן החשיבות להכיר לעומק את סביבת הפיתוח IDE, היות והיא הפלטפורמה שבה כותבים ומעצבים את קוד התוכנית.

ה- IDE מסתיר הלכה למעשה חלק גדול מהמורכבות החומרתית של ה-Arduino , וכך נהיה תהליך הפיתוח של הפרויקט לקל יותר בסביבת ה Arduino.

תהליך תכנות ה Arduino כולל את השלבים הבאים :

- חיבור כרטיס ה- Arduino ליציאת ה USB של המחשב האישי

- כתיבת סקיצה ש-"תביא" חיים ללוח.

- העלאת הסקיצה ללוח באמצעות חיבור ה-USB

- המתנה מספר שניות להפעלת הלוח מחדש.

- הלוח מבצע את הסקיצה שנכתבה עבורו.

לאחר התקנת IDE יש לחבר בין הבקר למחשב באמצעות כבל ה- USB. המחשב יבצע זיהוי לחומרה חדשה וידרוש התקנת דרייבר (תוסף) מתאים לשם יצירת תקשורת תקינה עם לוח ה .Arduino

תהליך ההתקנה וזיהוי הלוח מתחת למערכת Windows

להלן נתייחס כאמור לעבודה עם Arduino Uno בלבד, וכך לאחר חיבור בין המחשב והלוח, נבצע את הפעולות הבאות:

- המתנה לWindows לסיום התקנת הדרייבר המתאים. מנהל ההתקנים לא יצליח לזהות החומרה ותהליך ההתקנה ייכשל.

- ללחוץ על תפריט התחל וממנו לבחור את לוח הבקרה.

- לבחור את מערכת ותחזוקה ולאחר מכן לחיצה על מנהל ההתקנים.

- במנהל ההתקנים, אתר את Arduino Uno תחת לשונית יציאות (COM ו- LPT).

- לחץ לחיצה ימנית, ובחר באפשרות "עדכון מנהל התקן", ולבחור "עיין במחשב שלי".

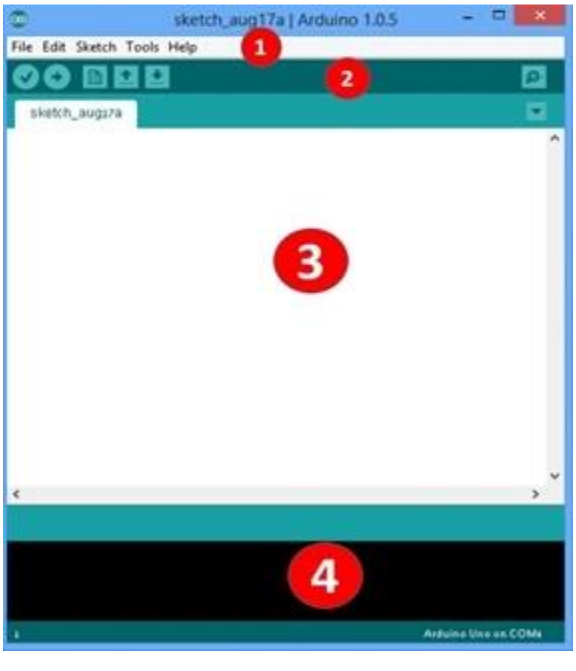
- נווט ובחר את קובץ הדרייבר בשם ArduinoUNO.inf הנמצא בתיקיית התוכנה

הכרת תוכנת IDE

בדיקת תקינות המערכת והתקשורת

לאחר שתוכנת ה Arduino והדרייבר הותקנו, ניתן ומומלץ לבצע בדיקה קצרה לוודא כי הכל פועל כשורה. מלבד זאת הדגמת הבדיקה יספק לנו היכרות מהירה עם תפריטי IDE החשובים ושלבי העבודה הננקטים לעיתים תכופות במהלך הפיתוח. לביצוע הבדיקה, יש להפעיל את התוכנה על ידי לחיצה על אייקון קיצור הדרך של התוכנה הנראה כדלהלן:

החלון שיפתח יראה כך:



איור-:3 חלון סביבת הפיתוח וחלקיו השונים

חלון סביבת הפיתוח מורכב מארבה חלקים (ממוספרים באיור הקודם):

1. בחלק הראשון מצוי התפריט הראשי

2. בחלק השני מצויים כפתורים לניווט מהיר

3. בחלק השלישי בו מצויים הקודים שכתבנו

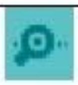
4. החלק הרביעי הוא חלון תצוגה להודעות המהדר, אזהרות ושגיאות וכן סוג הכרטיס ולאיזה פורט במחשב הוא מחובר.

לחצני הניווט המהיר

הלחצנים בסרגל הכלים שבחלק השני, (ראה איור -3) מספקים גישה נוחה לפונקציות הנפוצות ביותר בתוך התפריט .

איור-4: לחצני ניווט מהיר

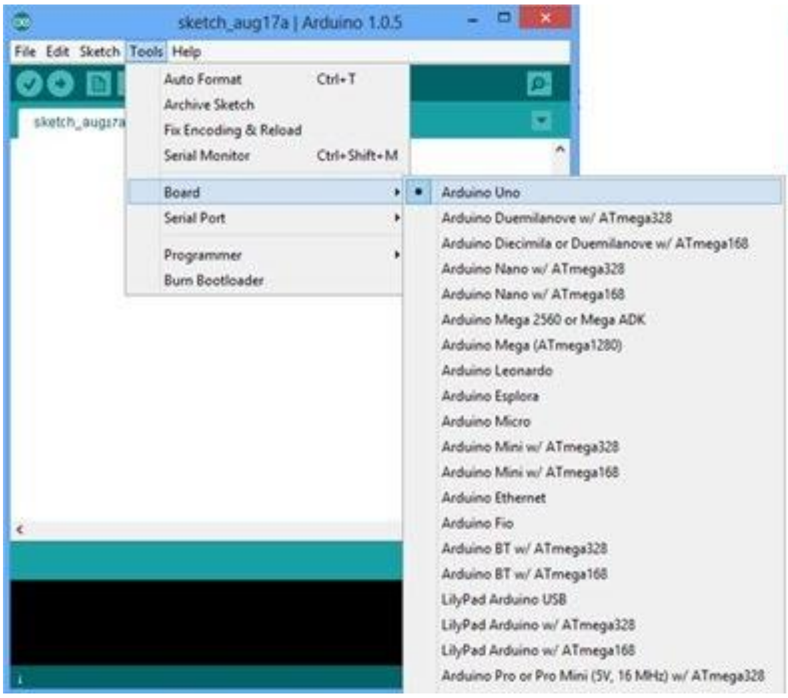
הטבלה שלהלן מפרטת תפקיד כל הלחצנים הנמצאים בסרגל הכלים.

סמל	תיאור
	Verify : מאפשר בדיקת שגיאות בקוד
	Upload: מהדר את הקוד שנכתב ו"מעלה" (צורב" אותו ללוח ה Arduino.
	New : פותח עמוד " Sketch " חדש
	Open: מציג תפריט של כל התוכניות הנמצאים בתקיה. לחיצה על אחד תפתח אותו . בתוך החלון נוכחי.
	Save: שומר את ה Sketch שנכתב.
	**Serial Monitor**: ממשק בסיסי לתקשורת טורית דו-כיוונית עם לוח Arduino.

טבלה-2: תיאור לחצני ניווט מהיר

הצעד הראשון והמתבקש שחובה לבצעו לאחר חיבור לוח ה Arduino, הוא בחירה בתוכנה ללוח ה- Arduino המחובר אליה דרך המחשב ושאיתו עובדים.

לשם כך, בתפריט כלים " Tools ", בחר Board, ומתוך הרשימה המופיעה בוחרים את סוג הלוח הנכון. במקרה שלנו, מסמנים את Arduino Uno ראה איור-5.



איור-5: בחירת סוג הארדוינו בתוך IDE

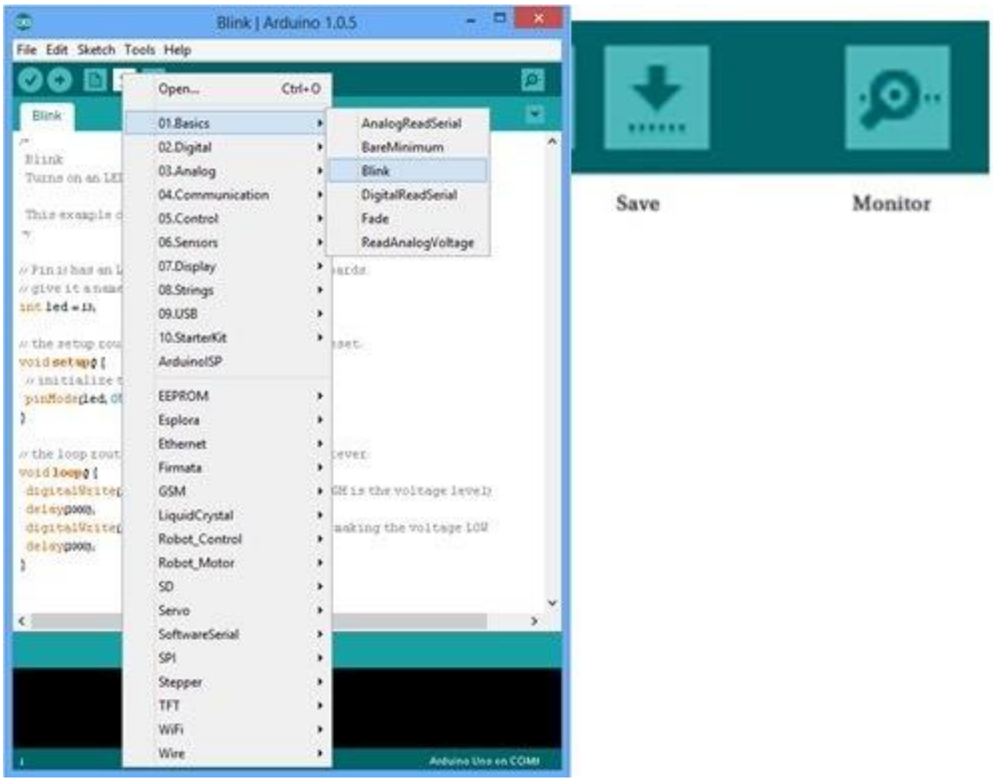
השלב הבא: יש לבחור לאיזו יציאה com ((של המחשב חובר כבל התקשורת בין ערכת ה Arduino לבין המחשב . זאת מבצעים על ידי לחיצה על תפריט כלים " Tools ", ומתוכו בחר Serial Port ואז לבחור ביציאת ה com הנכונה.

שים לב: שורת הטקסט המופיעה בפינה הימנית התחתונה של ה IDE מציגה לנו איזה לוח Arduino ואיזו יציאה טורית בחרנו, ראה איור-4. במידה והמוצג בשורה אינו מתאים לבחירתנו, ניתן לשנותם בהתאם לשלבים לעיל.

בדיקת ה- Arduino

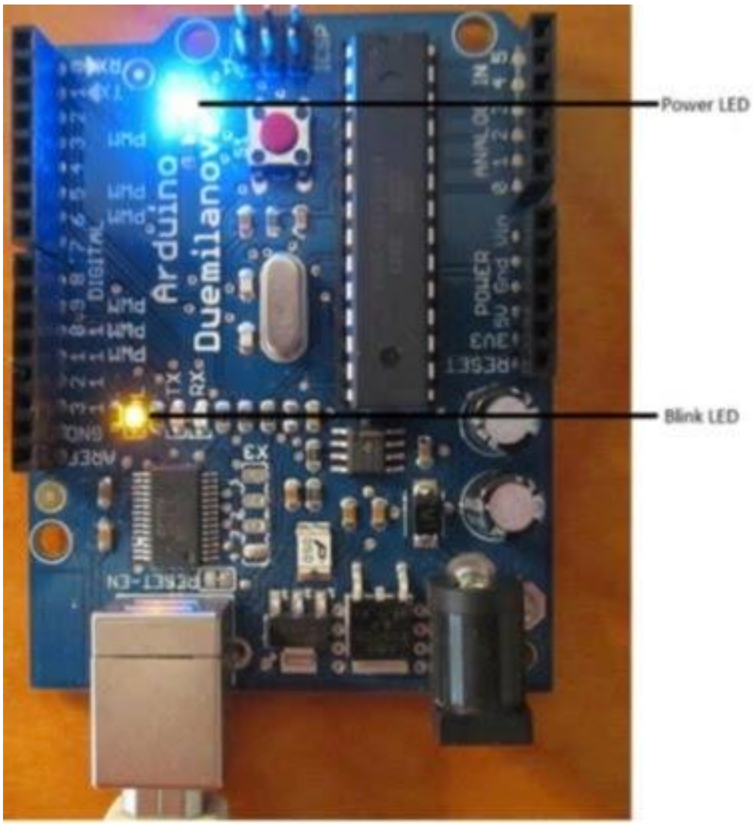
בתוך ה- Arduino מובנה LED המחובר להדק מספר 13, כך שלמעשה ניתן לבדוק את הלוח ללא כל חומרה נוספת. בין הדוגמאות הרבות הכלולות ב- Arduino IDEקיימת תוכנית דוגמה הנקראת blink, אשר גורמת ל- LED המובנה להבהב בכל שנייה. בהמשך בסעיפים הבאים נוכל ללמוד על עוד הדקים, לידים, וקודים ל Arduino. כעת, אנו נשתמש בדוגמה זו במטרה לבחון האם ה- Arduino מתקשר כראוי עם המחשב ?, והאם ניתן להעלות את הsketches ללוח ?.

שלב ראשון: יש לטעון את קוד תוכנית הדוגמא לזיכרון. לשם כך, נלחץ על כפתור פתח בסרגל הכלים ונווט ל- 1. Basics >> Blink, כפי שמוצג באיור- 6. בשלב הבא יש לטעון את הסקיצה אל המעבד על ידי לחיצה על כפתור Upload . לאחר מספר שניות, נקבל בשורת המצב הודעה שאומרת " טעינה בוצע ", וה- LED יתחיל להבהב בכל שנייה. אם זה הצליח ,סימן שה Arduino מוגדר בצורה נכונה.



איור-6: לשונית IDE ל טעינת התוכנית

איור-7 מציג בפינה השמאלית התחתונה את ה LED האמור להבהב בהתאם לתוכנית הדוגמא. ה- LED בחלק העליון משמש למתן אינדיקציה לכך שמתח חשמלי מחובר ללוח.



איור-7: תמונת LED לאחר הפעלת תוכנית Blink.

2.2 חמשת שלבי פיתוח התכנית

שפת Arduino מבוססת על שפות התכנות .C++/C רוב הקודים שנכתוב עבור Arduino, כולל התחביר שלהם "syntax", מבנה הקוד, אופרטורים, הוראות בקרה, ופונקציות, יישארו ביסודם ובפונקציונאליות כמו בשפת C. בנוסף לכך, שפת Arduino גם מספקת פונקציות מוכנות מראש לביצוע פעולות נפוצות כמו קלט, פלט וכו, שחלקן נציג בהמשך. עם זאת, ישנם מספר הבדלים בסיסיים בגישה המהותית לפיתוח הכולל.

הבדלים אלה משקפים בצורה נכונה את ההבדלים בין סביבת הפיתוח של Arduino לבין תוכנות יישומיות קלסיות. בסיס ההבדלים הוא בכך שנדרשים חלקים נוספים בתוכנית לשם הכנת הערכה לעבודה ולשם טיפול בקלט/פלט לתוכנית.

כל תכנית ניתן לתארה באמצעות חמישה אלמנטים בסיסים או צעדים. הדבר מחייב חשיבה מוקדמת בשלב התכנון של הקוד במונחים של אותם 5 אלמנטים ("סוף מעשה במחשבה תחילה"). אלמנטים אלה יתוארו להלן:

1. שלב האתחול

מטרתו העיקרית היא לבצע פעולות נכנה לסביבה שבה התכנית תפעל. לדוגמה, בגלשה באינטרנט, דפדפני האינטרנט מאפשרים לנו להגדיר את דף בית. או לקבוע מדפסת ברירת מחדל.

מה שמאפיין את האתחול הוא השימוש בנתונים הנטענים ממקום כלשהו כמו, קובץ נתונים, זיכרון, EEPROM, או אוגרים, ונתונים אלה משמשים להקמה קו התחלה בסיסי שיאפשר סביבה להרצה נכונה של התכנית. במילים אחרות, בשלב האתחול נעשות כל ההכנות ברקע כדי שהתוכנית תתבצע נכונה ותבצע את המשימה העיקרית שלה.

2. שלב הקלט

בכל תוכנית מחשב (במרבית המקרים) המשימה המבוצעת נועדה לקבל מצב קיים של מידע או נתוני מציאות, לעבד אותו לפי דרישה, ובהמשך ליצור מצב חדש של מידע זה. לדוגמא, אם אנחנו עומדים לכתוב תוכית למערכת גילוי אש, המערכת אמורה לקלוט את המידע שנמסר מחיישני האש, לפענח את מצבם הנוכחי, ובמידה ויש אש, לעשות משהו בקשר לזה (להתריע או להפעיל מערכת התזה).

התכנית עשויה להדרש ולבצע קריאות חדשות של המידע בפרקי זמן קבועים מראש ולהחליט אם חובה לבצע פעולה מתקנת או לא. התהליך כולו תלוי בהזנת נתונים חדשים מהחיישנים הרלוונטיים. שלב הקלט אם כן הוא רצף של הצהרות בתוכנית הנחוצות לשם אפשור השגת המידע בדייקנות מהמציאות ואשר בעזרתו ניתן לפתור את המשימה שבלב התוכנית וכך לבצע את ההחלטה הנדרשת

3. שלב התהליך

שלב התהליך אחראי על קליטת חבילה של נתונים ועיבודם כדי להפיק חבילה חדשה של נתונים. בהמשך לדוגמא של תכנית גילוי האש, ההליך יהיה חלק הקוד האחראי ביצוע הפעולות המובילות לקבלת ההחלטה האם המידע שנתקבל בקלט מהחיישנים, מוגדר כמצב גילו אש או לא. החלטה זו תתקבל על סמך רצף לוגי של צעדים (לפי אלגוריתם) בהתאם להגדרה מראש של מצב גילוי איש. בנוסף לכך ההליך יהיה אחראי על נקיטת פעולות מסוימות ככל שאלה נדרשות בהגדרת המשימה (הפעלת אזעקה למשל). יודגש כי תוכנית מסוימת עשויה להכיל מספר תהליכים, זאת בהתאם לרמת המורכבות של המשימה.

במילים אחרות, המתח המופק מחיישן החום(כלומר, טמפרטורה) על התוכית לקרוא (בפעולת קלט), לאחר מכן, על התוכנית לפרש/לעבד את הנתונים כדי לקבוע את המצב הנוכחי בהתאם למידע מהחיישנים החיישנים ולפעול בהתאם.

4. שלב הפלט

שלב הפלט הוא חלק הקוד בתוכנית האחראי על שימוש בתוצאות שלב התהליך.אופן השימוש משתנה ממקרה לאחר, יכול להיות הצגת הנתונים החדשים על התקן תצוגה או העברת הערך החדש שהתקבל משלב התהליך לתכנית אחרת (או הליך אחר) לשם המשך עיבוד . בחזרה לדוגמא של גילוי האש שלנו אזי: שלב הפלט יכול לגרום ל- LED השייך לחיישן מסוים להמשיך ולדלוק בצבע ירוק בתנאים רגילים. אם המערכת גילתה אש , המערכת תפעל לכך שידלקLED אדום.

5. שלב הסיום

שלב הסיום מכיל את חלק הקוד בתוכנית האחראי על ביצוע הפעולות הנחוצות לסיום תקין של התוכנית לאחר תום ביצוע משימתה.

יוער כי, יישומי מיקרו בקרים רבים, לא נועדו להסתיים. כך לדוגמא, מערכת גילוי אש נועדה להמשיך ולהתבצע לנצח אלא במצב של הפסקה יזומה או קריסתה עקב תקלות אחרות. במצב דברים זה ייתכן כי תהליך הסיום מנטרל את מערכת האזעקה לפני הכיבוי והתחזוקה או מתעד כמו קופסא שחורה את הנתונים לפני הכיבוי.

לסיכום: חמשת שלבי התכנות יכולים לשמש לשם גיבוש פתרון תכנותי כולל לפתרון בעיה נתונה. למרות שאלגוריתמים רבים קשורים באופן הדוק יותר לשלבים 2 ו -3 (כלומר, קלט ועיבוד) ,חמש, שלבי התכנות אמורים לסייע בניסוח אלגוריתם לפתרון משימה כלשהי.

2.3 מבנה של סקיצה

סקיצה טיפוסית מורכבת משני חלקים או פונקציות (שגרות): החלק הראשון הוא שגרת האתחול הנקראת setup ,והחלק השני הוא שגרה הנקראת loop, המכילה לרוב את עיקר הקוד של התוכנית. בסעיפים הבאים נתייחס בפירוט אודות שתי שגרות אלה .

תת השגרה setup()

כשאנחנו נערכים לצאת לריצה, ישנם צעדי הכנה שאנו נוקטים בהם: נועלים נעליים, דואגים לבקבוק מים ומבצעים מספר מתיחות.

צעדי הכנה דומים יש לנקוט עם Arduino לפני שממש ניגשים לקוד העיקרי של התוכנית. כלומר הארדוינו חייב להיות מוכן ומוגדר לפני תחילת המסע למשימה העיקרית. צעדי הכנה חיוניים אלה, כולם כלולים בתוך שגרת האתחול או פונקציה בשם setup.

הדברים האופייניים הנעשים ב setup הם הגדרת ההדקים ואתחול ההדקים הדיגיטליות כקלט או פלט, וקביעת קצב השידור לתקשורת טורית . להלן דוגמא לשגרה זו:

```
void setup()
{
pinMode(13,OUTPUT);
Serial.begin(9600);
}
```

הקוד לעיל בתוך תת השגרה setup מבצע את פעולות ההכנה הבאות:

- pinMode() – מגדיר את ההדק הדיגיטלי שמספרו 13 כהדק כפלט
- Serial.begin() -מגדירה את קצב שידור התקשורת הטורית ל 9600 סיבית/שניה

- ההגדרה void לפני השם setup, כיאות לקוד C, קובעת שהפונקציה אינה מחזירה ערך למפעיל אותה.

יש להדגיש כי גם במקרה ולא נדרשות פעולות אתחול, עדיין חובה לרשום ולצרף תת השגרה setup אחרת תבוא הודעת שגיאה בעת העלאת הסקיצה. הפונקציה הריקה תראה כך:

```
void setup(){
// nothing to setup
}
```

תת השגרה loop() - הלולאה האינסופית

עד מתי נמשיך בריצת הערב שלנו ?. או עד שנתעייף או לפי הזמן שקבענו מראש. באותו האופן יש לנהוג עם ה- Arduino. בברירת המחדל, הארדוינו ממשיך לבצע באופן אינסופי את הפקודות בתת-שגרה או פונקציה בשם loop(), והכל עד שהארדוינו פוגש תנאי עצירה בתוכנית או פשוט מנתקים את מתח ההזנה ומכבים אותו. להלן דוגמא לגוף פונקצית loop אשר גורמת להבהוב ה LED באופן אינסופי עם השהיה של 2000 מילישניות בין הבהוב למשנהו.

```
void loop()
{
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13,LOW);
delay(1000);
}
```

הקוד לעיל בתוך תת השגרה loop מבצע את פעולות הבאות:

- קובע ההדק הדיגיטלי שמספרו 13 כערך HIGH וכך מדליק LED – digitalWrite()

- מבצע השהיה של 1000 מילישניות בעזרת הפונקציה delay(1000)

· קובע ההדק הדיגיטלי שמספרו 13 כערך LOW וכך מכבה digitalWrite() - LED

רישום הערות בגוף הקוד – לא פחות חשוב מהקוד עצמו

כשכותבים קטע קוד גדול אנו יכולים להוסיף הערות לתוכניתנו, הערות אלה ישמשו אותנו ואת כל מי שיקרא את התוכנית שכתבנו. השימוש בהערות מומלץ בעיקר לתיעוד אופייה של פונקציה או של קטע קוד על מנת להפוך את התוכנית הופכת למובנת יותר וקלה יותר לתחזוקה ושינוי בעתיד. המהדר יתעלם להערות וידלג עליהן .

רישום ההערות זהה לשפת C סטנדרטית, הווה אומר,או כשורה בודדת או כבלוק.

לשורה אחת של הערות משתמשים בלוכסן כפול (/ /) בתחילת השורה.

במקרה בלוק של הערה בתוך הקוד, מתחילים את הבלוק עם /* ומסתיים ב */ .

להלן דוגמא לשתי האופציות:

// This is a single-line comment

/* And this is a block carried over

a couple of lines*/

מיקום ההערות בסקיצה

בכל סקיצה מומלץ לרשום תיעוד לגביה בבלוק הערות בתחילתה., בבלוק זה נספק תיאור של פעולות הסקיצה העיקרית, מי כתב אותה, את התאריך, ואת מספר הגרסה. להלן דוגמא לכך:

/*

Code to blink LED

Turns on an LED on for one second, then off for one second, repeatedly.

Author: Nasser & Morad

Date created : 1st August 2013

Version 1.0

*/

הערות של שורה אחת פורשים ברחבי הסקיצה כדי לאפשר הבנה ותזכורת מהירה מה כל חלק בקוד מבצע. אין צורך לרישום הערות בכל חלק שבקוד, אלא רק במקומות שבהם זה ישרת אותנו או ישרת מישהו אחר, בשלב מאוחר יותר, להבין את כוונתנו המקורית למה שהקוד אמור לבצע . להלן דוגמא לכך:

// Pin 13 has an LED connected on most Arduino boards.

// give it a name:

```
int led = 13;
digitalWrite(led, HIGH);   // turn the LED On
delay(1000);               // wait for a second
digitalWrite(led, LOW);    // turn the LED off
```

2.4 סיכום

מלת סיכום ביניים

בפרק הראשון הצגנו מאפיינים כליים של הארדוינו ובהמשך הבאנו את פריסת ההדקים ותיאורם. למדנו על המרכיבים העיקריים של ה Arduino. את החומרה, המוח, מתחים, מהירות, הדקים דיגיטליים ואנלוגיים וזיכרון.

בפרק השני, הכרנו את סביבת העבודה, בדקנו את הכרטיס , הכרנו את מושג הסקיצה והדגמנו כתיבת סקיצה ראשונה המביאה את ה- Arduino ל"התעורר לחיים" ולבצע פעולות.

בחנו בפירוט את תוכנת ה Arduino IDE , את המרכיבים של הסקיצה, את תת השגרת setup() ותת השגרה loops(), ועמדנו על חשיבות ההערות בגוף הקוד.

לאחר שרכשנו הבנה בסיסית אודות הארדוינו, התוכנה הרלוונטית ותוכנית הדוגמא, זו העת להעמקה ולפתח פרויקטים משלנו.

פרק זה הינו מעין הדרכה מכוונת והדרגתית תוך העמקה של האופן שבו יש לפתח ולהשלים פרויקט בסיוע הארדוינו. נדגיש כאן את סוגיית פיתוח פרויקטים בסביבת ה- Arduino המנצלים את הכניסות והיציאות הדיגיטאליות שהוא מספק למפתח התוכנה. לשם כך נדגים פיתוח פרויקט הגורם להפעלה ושליטה על מנועים, כולל הסבר לגבי המרכיבים השונים וחשיבותם בפרויקט.

פרק 3 ארדוינו בתנועה

בפרק הזה נסתפק בלהציג את העקרונות לתחילת העבודה ושליטה על מנועים, בתקווה שעניין זה יזכה לפיתוח והרחבה בהמשך. כך ניתן יהיה לנצל את הארדוינו כדי לשלוט במנועי DC קטנים המניעים רובוטים מסוגים שונים.

תיאוריה רלוונטית למנועים

מנועי DC משמשים לרוב לשליטה ברובוטים קטנים. מנוע קטן לזרם ישר ניתן למצוא במגוון רחב של מכשירים, בין היתר, בתוך מכוניות הנשלטות באמצעות שלט רחוק , חלונות מכונית החשמלית, נגני DVD, ומאווררים חשמליים קטנים ועוד.

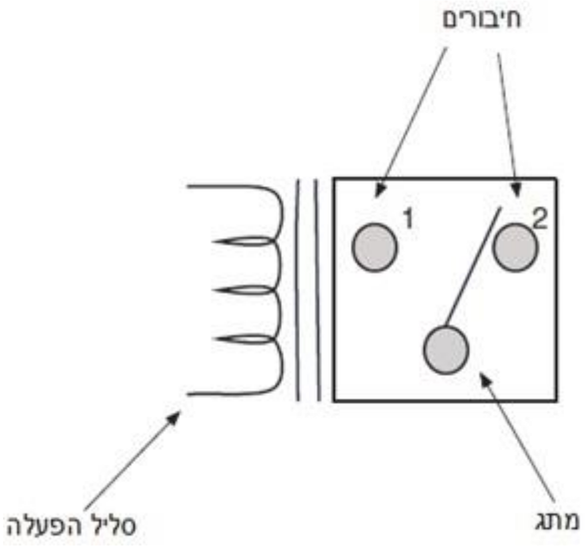
מתח מנועי DC קטנים נע בין V1.5 ל V30 המסופק דרך שני חוטים; יצרן המנוע מספק למשתמש רמות מתח המומלצות להפעלתו. הזנת מתח עודף מהמומלץ עלולה להביא לשריפת המנוע; אספקת מתח קטן מדי עלולה למנוע מהמנוע להסתובב.

על מנת להפוך כיוון הסיבוב של המנוע, לרוב מספיק רק להפוך את שני החוטים המחוברים אליו. הארדוינו מסוגל לספק ערכים נמוכים של זרם שאינה מספיקה כדי לסובב את המנוע. לכן, לעיתים קרובות, אנו נדרשים להשתמש באספקת מתח חיצונית כאשר הארדוינו ישמש kשליטה על פעילות המנוע)הפעלה\גיבוי\מהירות).

עצירת והפעלת המנוע

הדרך הפשוטה ביותר לעצור ולהפעיל מנוע היא להשתמש במתג מתג השולט על כך עם סגירתו והפעלתו.

הארדוינו יכול לשמש סוג של מתג חשמלי אוטומטי כדי להפעיל מנוע ולהפסיק מנוע. דרך נוספת היא להשתמש בממסר כמתג חשמלי. ממסרים כאלה זמינים במספר סוגי וחבילות. נשתמש בסוג ממסר הנקרא (SPDT), יכול להזרים זרם של 2 אמפר או יותר עם סליל הפעלה של 5 וולט. איור 4-1 מציג ממסר SPDT. על הממסר נשלוט בעזרת הארדוינו.

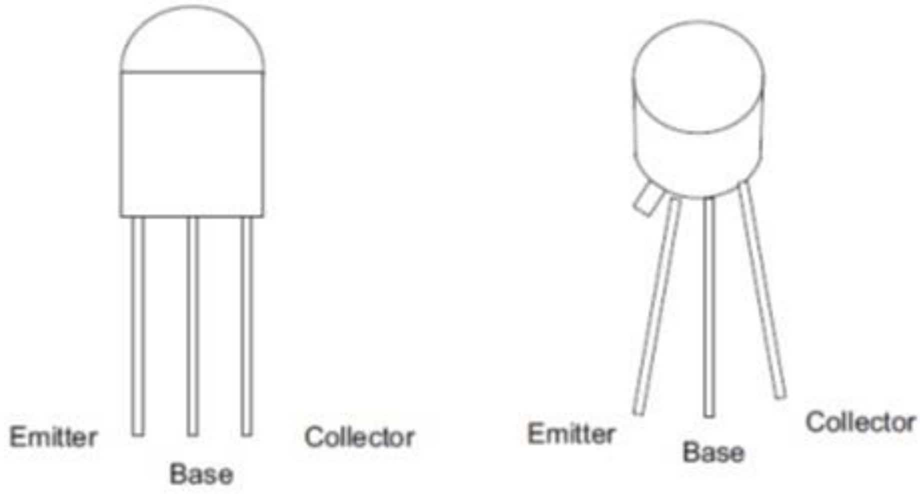


איור 4-1: מבנה הממסר מבפנים

לממסר ישנו סליל הפעלה הנוצר בתוכו מגנט כאשר הוא מחובר למתח. המגנט גורם לחיבור לנקודה 1 וכאשר אין מתח חוזר החיבור לנקודה 2.

כאמור, הארדוינו לא יכול לספק מספיק זרם, ולכן נצטרך להשתמש בטרנזיסטור כדי לשלט על הפעלת סליל הממסר. נשתמש בטרנזיסטור שמספרו N22222 מסוגNPN . הטרנזיסטור יתפקד בתחום הרוויה והקטעון וכך ישמש כמתג השולט על הפעלת סליל הממסר או עצירתו. מבנה עקרוני של הטרנזיסטור מוצג באיורים 4-1, 4-2.

איור 4-2: אריזות שונות לטרנזיסטור



4-3: סרטוט סכמתי של טרנזיסטור NPN

כאשר לא מחובר מתח חשמלי לבסיס של הטרנזיסטור, הטרנזיסטור נמצא במצב קטעון והמתג פתוח (בין קולט לפולט). כאשר בבסיס הטרנזיסטור יסופק מתח של מהארדוינו מתח זה גורם לטרנזיסטור לכנס לתחום הרוויה ויהווה כמתג בין הקולט לפולט.

3.1 הפעלת והפסקת מנוע DC קטן

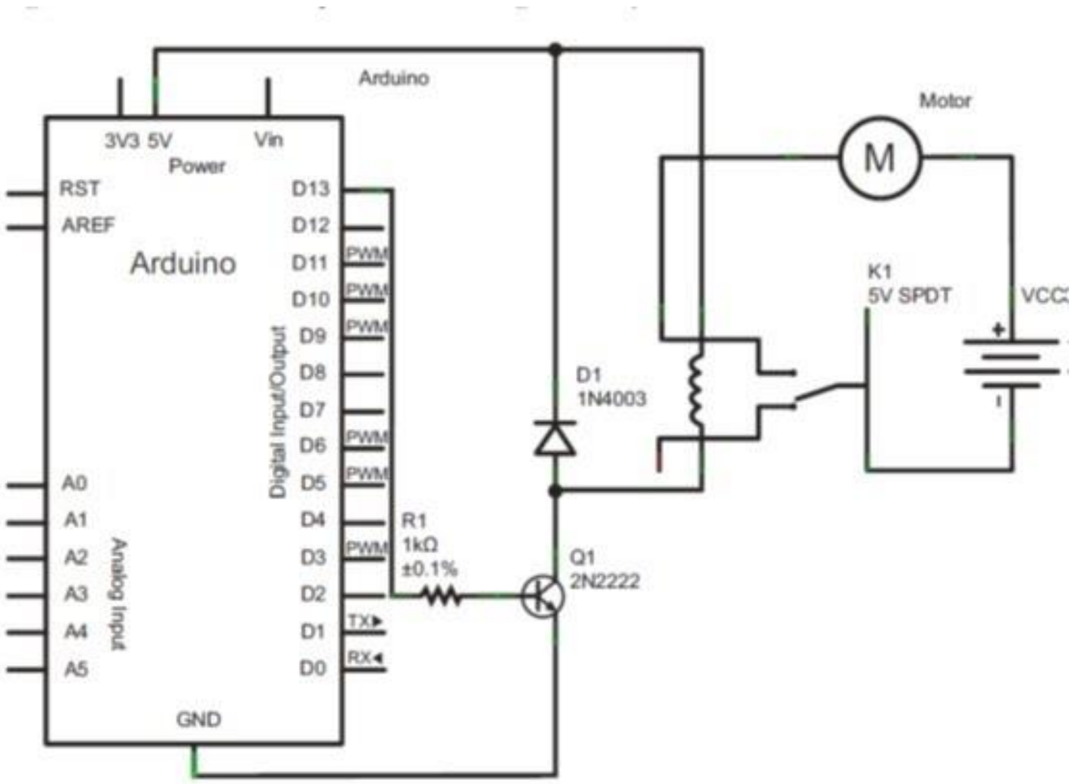
בפרויקט הזה נפעיל מנוע DC קטן המחובר לממסר .

מעגל חשמלי רלוונטי

באיור 4-4 מתואר הסרטוט החשמלי להפעלת מנוע זרם ישר ועצירתן.

ה Arduino שולח אות גבוהה לבסיס הטרנזיסטור, זה גורם לטרנזיסטור לכנס לתחום הרוויה שלו וזה בתורו מזרים זרם לסליל הממסר, אשר מעביר את המגעים בתוכו וכך נוצר מעגל חשמלי סגור למנוע שגורם לו להסתובב.

כאשר הארדוינו שולח אות נמוכה לבסיס הטרנזיסטור, הוא מכבה אותו, וגרם להפסקת הזרם בסליל הממסר שמעביר את המגים בתוכו ופותח את המעגל החשמלי של המנוע דבר שמפסיק את פעולתו.



איור 4-4: סרטוט חשמלי למעגל המנוע

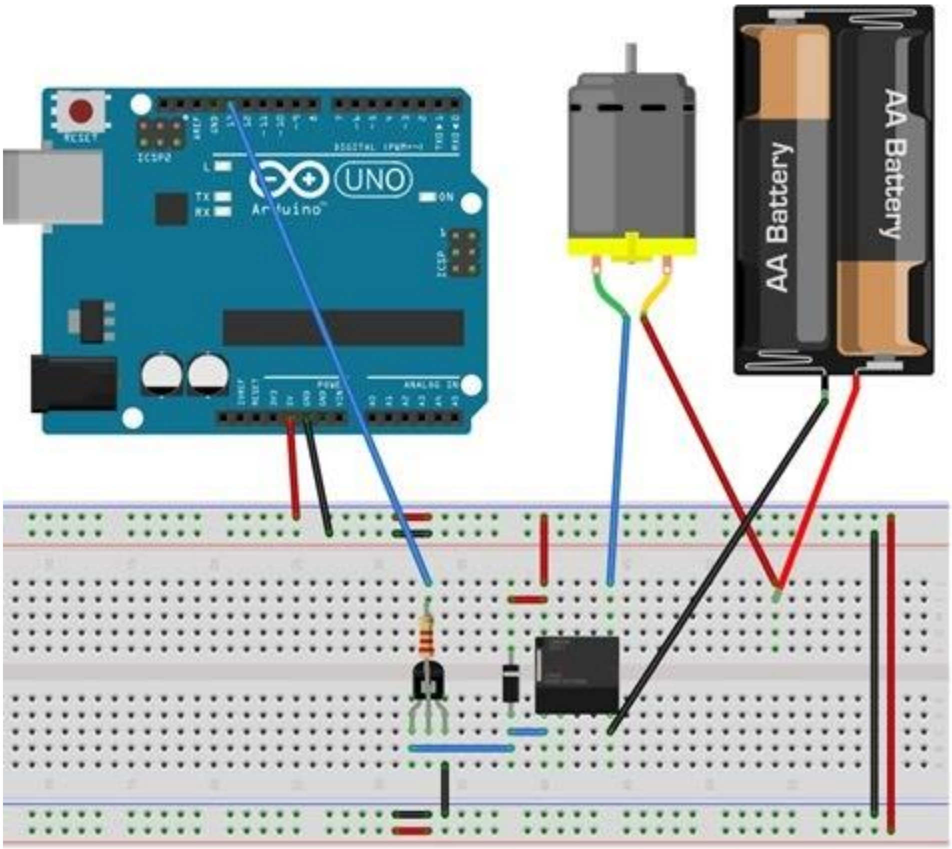
בשרטוט המעגל החשמלי המתואר , דיודה D1 מהווה דיודת הגנה מפני ספייקים.

ניתוח פעולת המעגל ודיודת הגנה על המנוע

כאשר הטרנזיסטור מפסיק לספק זרם לעומס ההשראתי, נוצר על הסליל מתח שמתנגד לשינוי בזרם, זאת בהתאם לחוקי לנץ ופארדיי. המתח שנוצר על הסליל יכול להיות גדול יותר ממתח המקור שמזין את המנוע הפרש המתחים בין שני צידי הסליל יוצר זרם השואף לחזור מהסליל דרך הטרנזיסטור ובחזרה למקור. הדבר עלול לשרוף את הטרנזיסטור.

לצורך התמודדות עם התופעה, נחבר דיודות הגנה. כל זמן שהפרש המתחים תקין משני צידי הטרנזיסטור, הדיודות תפעל במצב של נתק. כאשר מפסיקים להזרים זרם לסליל נוצר מתח הפוך וזה גורם שהדיודה תיפרץ ותהפוך לקצר.

הרעיון של ההגנה של הדיודות מתבססת על העובדה שזרם יעדיף תמיד לזרום דרך המסלול עם ההתנגדות הנמוכה ביותר. כידוע, במוצא הטרנזיסטור יש התנגדות מסוימת שהינה יחסית גדולה בעוד שהדיודות, כשהם נפרצות, מהוות קצר חסר התנגדות. מכאן יוצא שהזרם החוזר מהמנוע יעדיף לעבור דרך הדיודות בחזרה למקור במקום לעבור דרך הטרנזיסטור שבמוצא הדוחף.



איור 4-5: סרטוט חשמלי למעגל המנוע

קוד רלוונטי לפרויקט - 4

כדי להפעיל את פרויקט המנוע יש להרכיב את המעגל באיור 17 ולטעון את הקוד הבא לערכת הארדוינו. הסקיצה הבאה מפעילה את המנוע לסירוגין למשך חמש שניות ולאחר מכן מפסיקה אותו גם למשך חמש שניות.

```
;int trn = 13
/***********************************************
()void setup
}
 ;(pinMode(trn, OUTPUT
{
/***********************************************/

()void loop
```

```
{
digitalWrite(trn, LOW);
delay(5000);
digitalWrite(trn, HIGH);
delay(5000);
}

/*********************************************/
```

הסבר לקוד הפרויקט

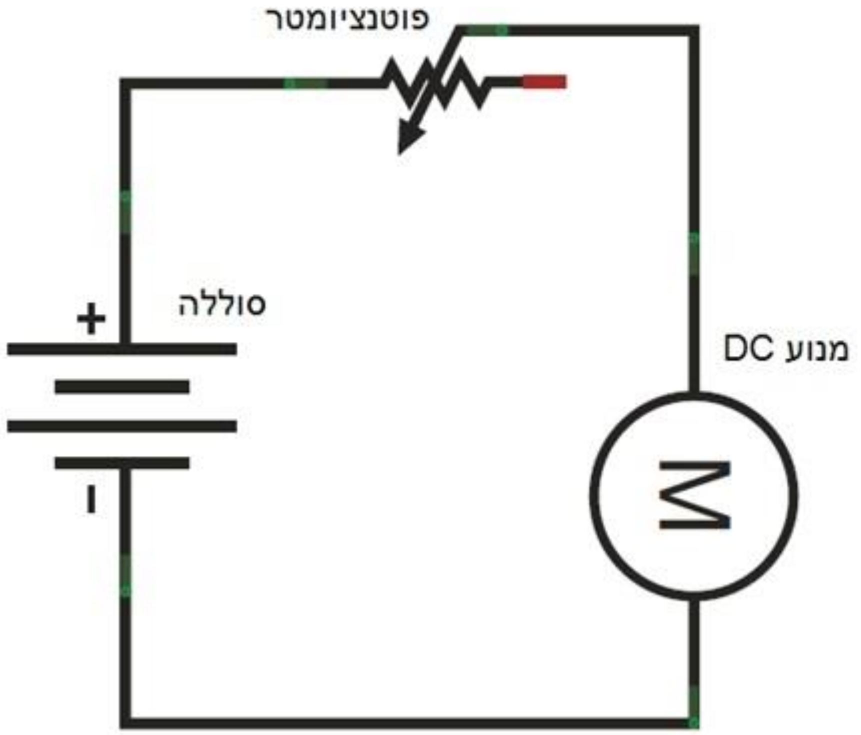
הסקיצה שולטת על הטרנזיסטור שבסיסו מחובר להדק הדיגיטלי מס' 13 בארדוינו. שימוש בטרנזיסטור וממסר הוא דרך מקובלת למיתוג מנוע לסירוגין. לעיתים נרצה לשלוט במהירות ושל המנוע וגם בכיוון הסיבוב שלו. עניין זה מודגם בפרויקט הבא.

3.2 בקרת מהירות והפיכת כיוון סיבוב המנוע

בפרויקט זה נדגים שיטות שונות ורכיבים שונים לשליטה במהירות וכיוון הסיבוב של המנוע. כידוע, כיוון הזרם בסליל המנוע קובע את כיוון הסיבוב, מכאן, הפיכת קוטביות החיבור תגרום להפיכת כיוון הסיבוב. פתרון זה ישים כל עוד שהמנוע אינו מותקן במקום מרוחק כמו על רובוט נייד בתנועה. במקרה האחרון, החלפת החוטים אינה ניתנת לביצוע וכך יוצא שהשליטה על מהירות המנוע תהיה מסובכת. מהירות המנוע יכולה להיות נשלטת על ידי שינוי המתח המופעל עליו. הורדה ברמת ההזנה תביא לאיטיות בפעולתו. כך גם הגברת המתח תביא למהירות מנוע גבוהה יותר.

הערה: בהקשר זה יש לשים לב כי הפעלת מנוע במתח גבוה מדי עלולה לגרום להתחממות יתר, וכתוצאה מכך לניזק בלתי הפיך למנוע.

אחת הדרכים לשינוי המתח על המנוע היא להיעזר בפוטנציומטר כדי לייצר התנגדות משתנה כפי שמודגם באיור 4-6.



איור 4-6: שימוש בפוטנציומטר כדי לשלוט במהירות המנוע

יתרון השיטה הוא בפשטותה מחד, מאידך, חסרונה הבולט היא יעילותה הנמוכה עקב התחממות מהירה של הפוטנציומטר.

הגישה המומלצת היא להשתמש בשיטה הנקראת אפנון רוחב הדופק (PWM).

עקרון PWM

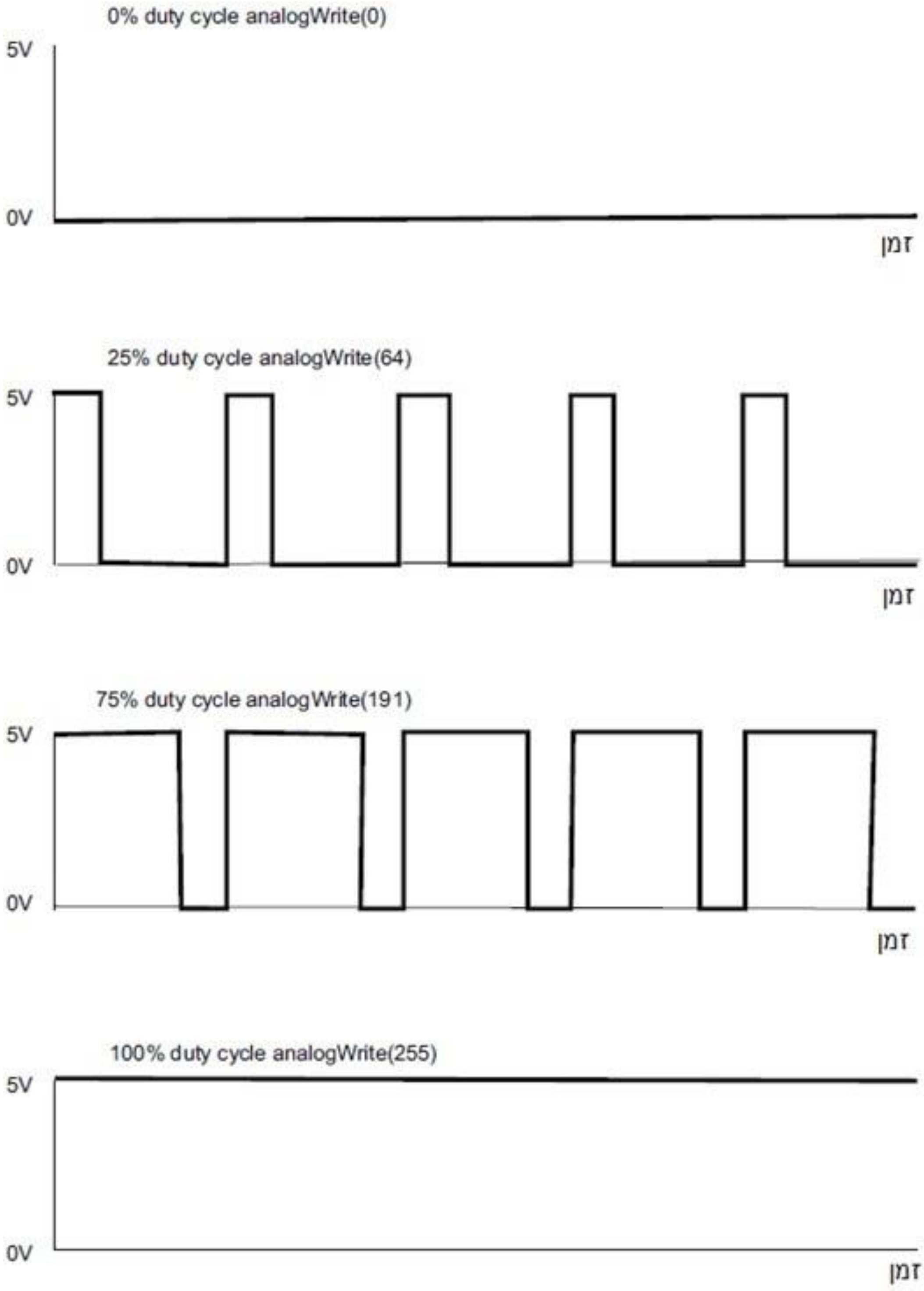
PWM היא הדרך היעילה ביותר להפעלת המנוע, שכן באמצעותה מספקים פולסים טוריים של מתח למנוע, וכך הוא יעבוד ביעילות גבוהה יותר.

לצורך יישום הרעיון הנ"ל אנו נעזר בפונקציות ספריה מוכנות של הארדוינו.

הראשונה, פונקצית analogRead() אשר פגשנו בפרויקט 4. הפונקציה מבצעת קריאה של מתח אנלוגי מהכניסה המתאימה, ובתורה, ממירה אותו למספר דיגיטלי באמצעות ממיר אנלוגי לדיגיטלי הפנימי אשר קיים בארדוינו (ADC) .

הפונקציה analogWrite () מבצעת פעולה הפוכה לפונקצית analogRead(). היא מוציאה מתח אנלוגי בבהתאם לערך דיגיטלי המועבר אליה כפרמטר. כך למעשה, פונקציית analogWrite() מייצרת פלט PWM.

איור 4-7 מדגים ייצוג גרפי של פלט הארדוינו כאשר משתמשים בפונקציית analogWrite()



איור 4-7 פלט פונקציית analogWrite()

הדגשי חומרה ותוכנה חשובים לתפעול מנועים

שפת התכנות של ה Arduino עושה את השימוש ב- PWM נוח וקל יחסית . בתחילתו של כל מקטע קוד כלשהו, התוכנה של ה Arduino קובעת תצורה אוטומטית עבור כל הטיימרים הזמינים במעבד הארדוינו. טיימרים אלה המא האחראים להפקת סיגנל PWM בהתאם לפרמטרים הניתנים על ידי המשתמש.

הפונקצייה (analogWrite(pin, duty cycle אחראית על הפקת גל PWM ב- duty cycle מסוים ובפין מסוים שמספרו pin. שני הפרמטרים הנ"ל מספיקים לאפיון הגל הרצוי. הפרמטר duty cycle אמור לקבל ערך בין 0 עד 255 , ואילו pin זה מספר הקובע אחד מהדקי ה PWM בארדוינו.

בארדוינו הסטנדרטי' הדקי ה PWM הם 3, 5, 6, 9, 10 או 11, או ההדקים מהדק מספר 2 עד 13 בארדוינו Mega

3.4 בקרת מנוע באמצעות H-Bridge

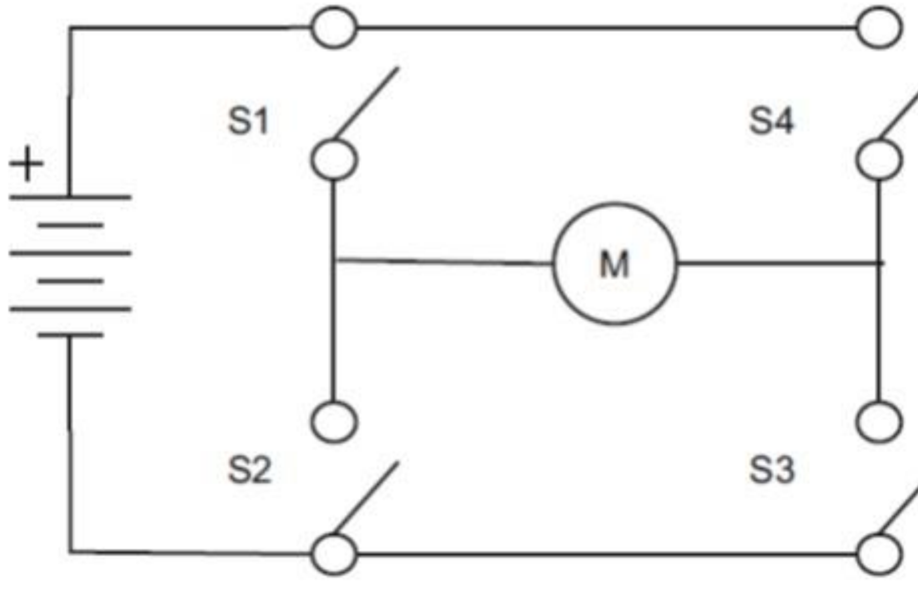
הארדוינו איננו מסוגל להניע את המנועים ישירות, כיוון שאיננו יכול לעמוד בדרישת הזרם הנצרך על ידי המנוע בדרך כלל. מן העבר האחר, אפשר לנצל את ה- Arduino כדי לשלוט על המהירות וכיוון הסיבוב.

כדי להתגבר על בעיית דחיפת הזרם למנועים, מספקים את הזרם באמצעות בקר המנועים ייעודי, המשמש כחוצץ בין הארדוינו והמנוע. במקרה זה מעבד הארדוינו יתפקד רק כמעביר אותות שליטה ובקרה לחוצץ זה. בקרים אופייניים של מנועי DC משתמשים בטופולוגיה בסיסית הקרויה H – גשר (H-bridge). בקצירת האומר, מדובר למעשה ב- 4 מפסקים המחוברים בצורה דמוית האות H למנוע.

H-bridge היא שיטה מקובלת ונפוצה לשליטה במהירות וכיוון מנועDC h קטנים. להלן

נדגים את השימוש בשיטה זו בשני אופנים. תחילה נשתמש ב H-bridge להפעלה והפסקת מנוע לסירוגין תוך שליטה גם על כיוון הסיבוב. בהמשך ניישם P-WM כדי לשלוט על מהירות המנוע.

איור 4-8 מביא תיאור ארכיטקטורה פשוטה של מעגל בקרת מנועיH-bridge .



איור 4-8: H-bridge המורכב מארבעה מפסקים

הטבלה הבאה מתארת את אופן תגובת מעגל H-bridge לכל מצב מפסיקים ובהתאם, את השפעת החיבור על כיוון המנוע.

S1	S2	S3	S4	פעולת המנוע
סגור	פתוח	סגור	פתוח	סיבוב עם כיוון השעון
פתוח	סגור	פתוח	סגור	סיבוב נגד כיוון השעון
פתוח	פתוח	פתוח	פתוח	חופשי
סגור	פתוח	פתוח	סגור	עצירה
פתוח	סגור	סגור	פתוח	עצירה

טבלה 4-1 מיקום ארבעת המתגים ופעולת המנוע

שתי השורות האחרונות מתארות מצב שבו קצוות המנוע מקוצרות יחדיו, דבר זה גורם למנוע זרם הישר לפעול כמו גנרטור, הווה אומר נגד עצמו. במצב הזה סיבוב המנוע יוצר מתח שמנסה לכפות על המנוע להפוך את כיוון הסיבוב. הדבר מביא לעצירה מהירה של המנוע. מצב זה נקרא "בלימה". כאשר המפסקים פתוחים, המנוע יסתובב חופשי לל בלימה, עד לעצירה כתוצאה מכוחות החיכוך המשפעים עליו. דומה הדבר לרכב הנוסע בהילוך סרק.

המתגים לעיל, מופעלים תמיד בזוגות, או המתג השמאלי העליון S1 והימני התחתון S3, או שמאלי תחתון S2 וימני עליון S4. לעולם לא יופעלו שני המתגים באותו "הצד" של הגשר S1 ו S2 או S4 ו S3. אם שני המתגים בצד אחד של הגשר מופעלים הדבר יביא ליצירת קצר בין הקוטב החיובי והשלילי של הסוללה, דבר שבתורו יביא להתרוקנות מהירה של הסוללה, ואף לשריפת המעגל החשמלי.

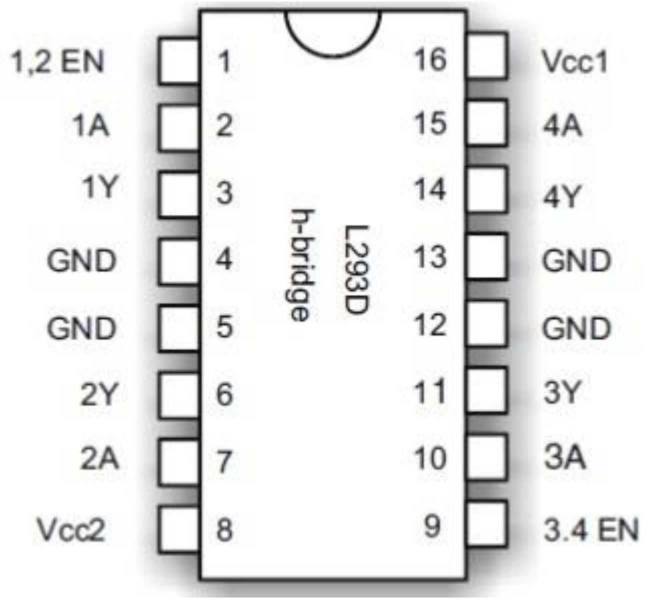
מבחינה עקרונית, ניתן ליישם- את ה H-bridge בחומרה באמצעות טרנזיסטורים אשר מחליפים את המתגים. אבל עבור קלות שימוש ומהירות ההרכבה וחיבורים נשתמש במעגל משולב (IC) המכיל בתוכו גשר H.

בפרקטיקה, המעגל המשולב L293D, משמש להפעלת מנועי זרם ישר קטנים ובהפעלת רובוטים קטנים גם. רכיב זה מיישם דוחף זרם בקונפיגורציית H-bridge. הוא מכיל בתוכו ארבעה דוחפי זרם המאפשרים הפעלת מנוע בשני הכיוונים , שליטה על מהירות הסיבוב, בלימה או סיבוב חופשי. הרכיב מאפשר לשלוט על המנוע בעזרת רמות TTL 5V.

הרכיב L293D עם שני H-bridge

הרכיב המשולב L293D יכול להפעיל שני מנועי זרם ישר בו-זמנית. ניתן בעזרתו לקבוע את כיוון הסיבוב של כל אחד מהמנועים, באופן בלתי תלוי בכיוון הסיבוב של המנוע האחר.

הרכיב L293D מגיע באריזה של 16 הדקים. איור 4-9 מביא תיאור היציאות והכניסות של רכיב זה.



איור 4-9: תרשים הפינים של L293D

טבלה 4-2 מתארת את תפקיד כל אחד מהפינים של רכיב זה.

הדק מספר	שם ההדק	תיאור
1	1,2 EN	אפשור חצי מה H-bridge דוחף 1 ו 2
2	1A	כניסת חצי מה H , דוחף 1
3	1Y	יציאת חצי מה H , דוחף 1
4, 5, 12, 13	GND	חיבור לאדמה
6	2Y	יציאת חצי מה H , דוחף 2
7	2A	כניסת חצי מה H , דוחף 2
8	Vcc2	אספקת מתח למנוע בין 4.5–36V
9	3,4 EN	אפשור חצי מה H-bridge דוחף 3 ו 4
10	3A	כניסת חצי מה H , דוחף 3
11	3Y	יציאת חצי מה H , דוחף 3
14	4Y	כניסת חצי מה H , דוחף 4
15	4A	יציאת חצי מה H , דוחף 4
16	Vcc1	5V מתח הזנה לרכיב

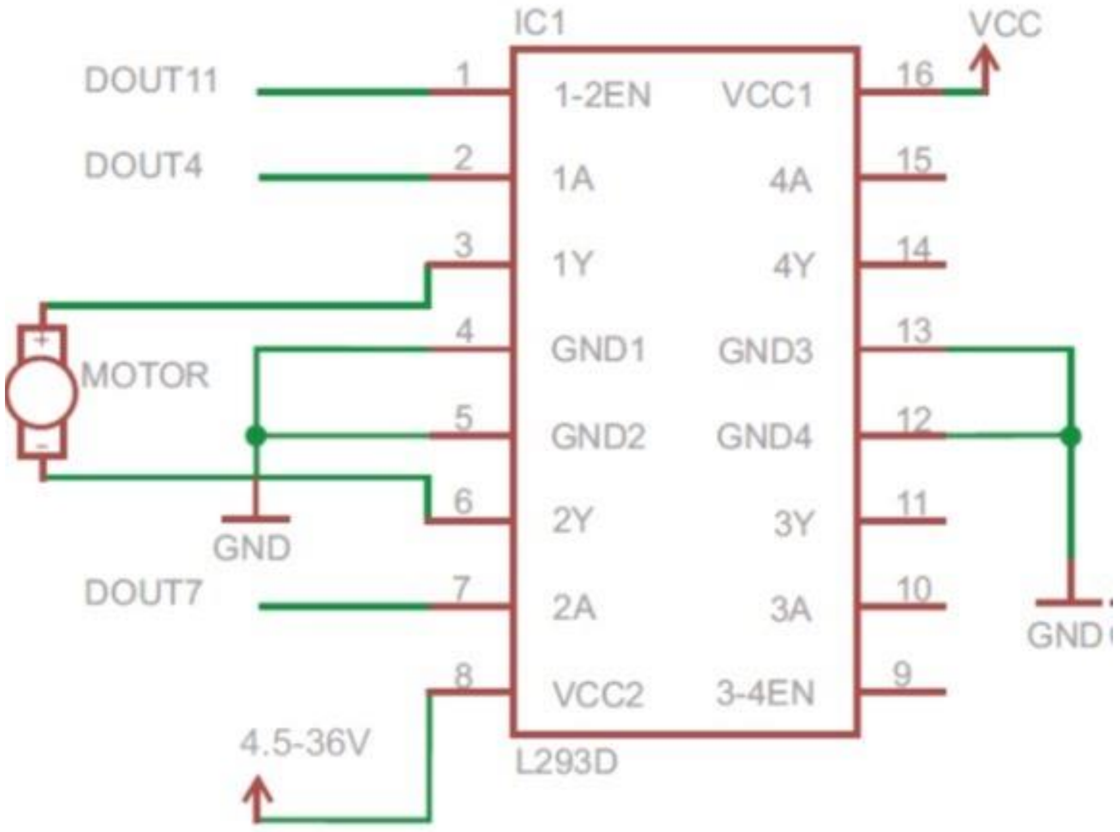
טבלה 4-2: הדקי L293D

פרויקט 5 בקרת מהירות והפיכת כוון המנוע – שימוש ב- L293D

חיבור חשמלי :

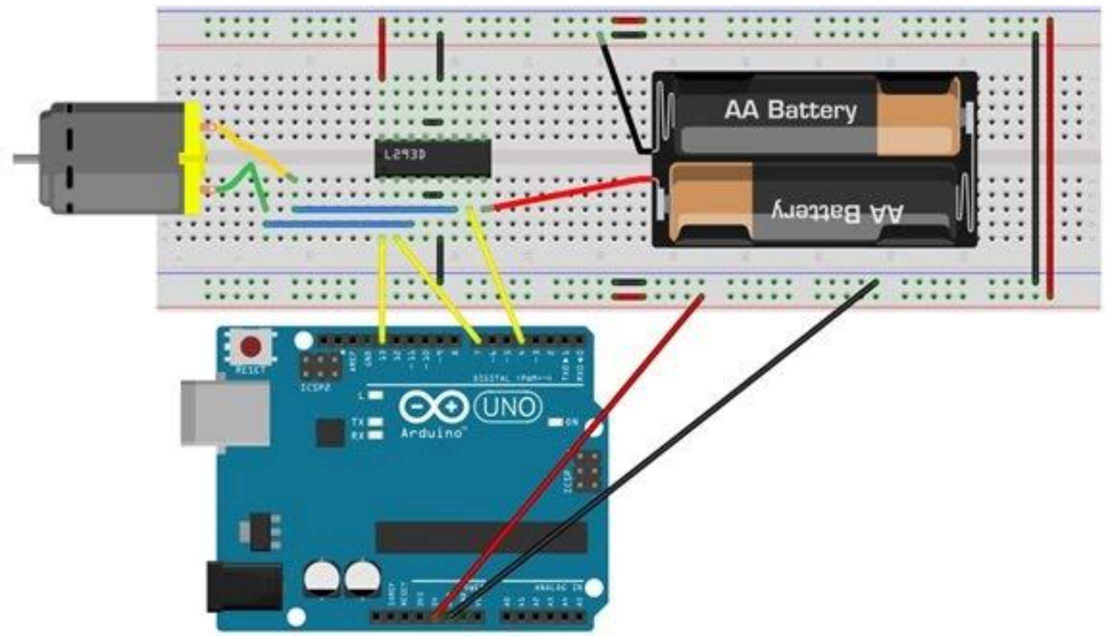
הרכיב L293D מסוגל להפעיל שני מנועי DC. בפרויקט זה אנו מדגים איך להפעיל מנוע אחד. באותה הטכניקה, ניתן בקלות להשתמש ולהפעיל שני מנועים וכך גם להרחיב את הפרויקט במידת הצורך.

איור 4-10, מתאר תרשים המעגל החשמלי של הפרויקט המוצע ובו מפורטים החיבורים בין המנוע, H-bridge והארדוינו.



איור 4-10: חיבור מנוע DC לרכיב L293D

אחת האפשריות של המעגל הכולל מודגמת באיור 4-11 שבה הורכב הפרויקט על גבי מטריצת חיבורים.



איור 4-11: מטריצת חיבורים לפרויקט בקרת מנוע DC בעזרת L293D

קוד רלוונטי לפרויקט - בקרת מנוע DC באמצעות L293D

בתרשים 3-10, המנוע מחובר להדקים 3 ו 6 של L293D שהם היציאות של ה H-bridg. ההדקים 1, 2 ו 7 מחוברים להדקי הארדוינו D11 D4 ו D7 בהתאמה. בהתאם לכך, טבלה 4-3 מספקת את האינפורמציה הנדרשת להפעלת המנוע ולכתיבת הסקיצה המתאימה לפרויקט .

מנוע	2A	1A	1,2 EN
סיבוב עם כיוון השעון	גבוה	נמוך	גבוה
סיבוב נגד כיוון השעון	נמוך	גבוה	גבוה
מצב בלימה	נמוך	נמוך	גבוה
מצב בלימה	גבוה	גבוה	גבוה
מצב בלימה	Φ	Φ	נמוך

טבלה 4-3: טבלת האמת לרכיב L293D

מטבלת 4-3, ניתן לאות שהדק (EN 1,2) חייב להיות בגבוהה כדי שהמנוע יסתובב

ואילו ההדקים A1 ו A2 הן השולטות על כיוון הסיבוב. בהסתמכות על מידע זה, להלן הסקיצה הרלוונטית לפרויקט.

```
int enablePin = 11;
int in1A = 4;
int in2A = 7;
/*******************************************************************************************/
void setup()
{
 pinMode(enablePin, OUTPUT);
 pinMode(in1A, OUTPUT);
 pinMode(in2A, OUTPUT);
 digitalWrite(enablePin, LOW);
}
*******************************************************************************************/
/
void loop()
{
 digitalWrite(in1A, HIGH);
 digitalWrite(in2A, LOW);
 digitalWrite(enablePin, HIGH);
 delay(5000);
 digitalWrite(enablePin, LOW);
 delay(2000);
 digitalWrite(in1A, LOW);
 digitalWrite(in2A, HIGH);
 digitalWrite(enablePin, HIGH);
 delay(5000);
 digitalWrite(enablePin, LOW);
 delay(2000);
}
/*******************************************************************************************/
```

סיבוב המנוע עם כיוון השעון ← digitalWrite(in1A, HIGH);

בלימה ← digitalWrite(enablePin, LOW);

סיבוב המנוע נגד כיוון השעון ← digitalWrite(enablePin, HIGH);

הסבר לסקיצה

בתחילה אנו קובעים את הדק האפשור בנמוך (EN 1,2) וכך למעשה אנו משבתים את ה H-bridge. במהלך פונקציה loop, מאפשרים את ה H-bridge על ידי העלאת הדק האפשור (EN 1,2) לגבוה.

הדק in1A ו in2A המחוברות להדקים 2 (A1) ו 7 (A2) ברכיב L293D , משתנים בתוך הפונקציה loop כדי שהמנוע יסתובב בכיוון אחד למשך חמש שניות. לאחר מכן סיבוב בכיוון ההפוך למשך חמש שניות נוספות, במרווח של שתי שניות השהיה בין כיוון למשנהו.

בסקיצה הנ"ל ראינו כיצד ניתן להשתמש ברכיב L293D כדי לשלוט במנוע אחד. אם ברצוננו לשלוט במנוע נוסף, יש רק לשכפול המעגל ולהשתמש ב H-bridge השני שברכיב L293D .

עדכון הפרויקט - שינוי מהירות המנוע

בפרק זה כבר הצגנו כיצד ניתן לשלוט על מהירות המנוע באמצעות PWM . יישום מעשי לעניין ניתן לבצע בעזרת המעגל כאשר אות ה PWM יסופק למנוע דרך הדק האפשור (EN 1,2), וכך למעשה לאפשר ולבטל את ה H-bridge.

להלן הסקיצה החדשה ששולטת במהירות המנוע.

```
int enablePin = 11;
int in1 = 4;
int in2 = 7;
/**********************************************************************************/
void setup()
{
pinMode(enablePin, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
digitalWrite(enablePin, LOW);
}
/**********************************************************************************/
```

```
void loop()
{
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);
digitalWrite(enablePin, HIGH);
for(int i = 0 ; i <= 255; i++) {
analogWrite(enablePin, i);
delay(50);
}
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
for(int i = 0 ; i <= 255; i++) {

analogWrite(enablePin, i);
delay(50);
}
}

/***********************************************************************************/
```

הקוד לעיל דומה לסקיצה הקודמת, ההבדל המרכזי הוא השימוש בלולאת for אשר משנה את ערך ה PWM מ 0 עד 255. בהתחלה המנוע מסתובב בכיוון אחד ואחר כך לכיוון האחר עם עליה הדרגתית קבועה במהירות הסיבוב ממצב מנוחה עד למהירות מלאה.

הערה : עם הרצת הסקיצה אנו נבחין בהשהיה בין כל התחלת סיבוב המנוע. הדבר נובע מכך שהמנוע דורש מתח מינימלי להתחלת הסיבוב. מתח מינימלי פירושו, ערך מינימילני ל PWM. וזה תלוי במאפייני המנוע.

גם בסקיצה הזאת שלטנו רק במנוע אחד, אבל כמו בסקיצה הקודמת אפשר לשלוט בעוד מנוע על ידי שיכפול המעל והקוד והתאמתו לחיבורים.

באותה השיטה ניתן לשלוט במנועי DC גדולים, אך הדבר מחייב רכיבים מיוחדים העומדים בהספקים גדולים.

פרק 4 שימוש ב EEPROM הפנימי

אחסון נתונים ב EEPROM המובנה ב Arduino

המשתנים המוגדרים בתוך סקיצת הארדוינו מאוחסנים בתוך זיכרון ה RAM והם נעלמים כאשר מאפסים את הארדוינו או מכבים אותו. השאלה הנשאלת כאן מה אם רוצים לשמור ערכים ונתונים בתוך הארדוינו לשימוש עתידי, כגון שמירת ושינוי קוד סודי של כספת.

כאן נכנס לשימוש זיכרון ה EEPROM (electrically erasable read-only memory). זהו זיכרון לקריאה בלבד והניתן רק חשמלית למחיקה . המשתנים השמורים בזיכרון ה EEPROM הבנוי בתוך המיקרו בקר ATmega328 אינם נאבדים כאשר מנתקים הארדוינו מהמתח.

זיכרון ה EEPROM בארדוינו יכול לאחסן 1,024 משתנים בגודל בייט כל משתנה. משתנים אלה מאוחסנים במערך זיכרון הממוספר מ -0 עד 1,023. נזכיר כאן כי תא זיכרון בגודל בייט יכול לאחסן מספר שלם עם ערך בין 0 ל 255.

כדי להשתמש בזיכרון הEEPROM הפנימי של הארדוינו בסקיצות שלנו, אנו חייבים קודם כל לקרוא לספריית ה EEPROM .

כלולה בתוך ה Arduino IDE באמצעות ההנחיה הבאה :

```
#include <EEPROM.h>
```

כדי לכתוב ערך לזיכרון ה משתמשים בפונקציה הבאה

```
EEPROM.write(a, b); .
```

כאשר a מצביע על כתובת תא הזיכרון (ערך בין 0 ל 1023) שבתוכה ישמר המשתנה b שהוא בגודל בייט אחד.

כדי לקרוא ערך מזיכרון ה EEPROM , משתמשים בפונקציה הבאה

```
value = EEPROM.read(position);
```

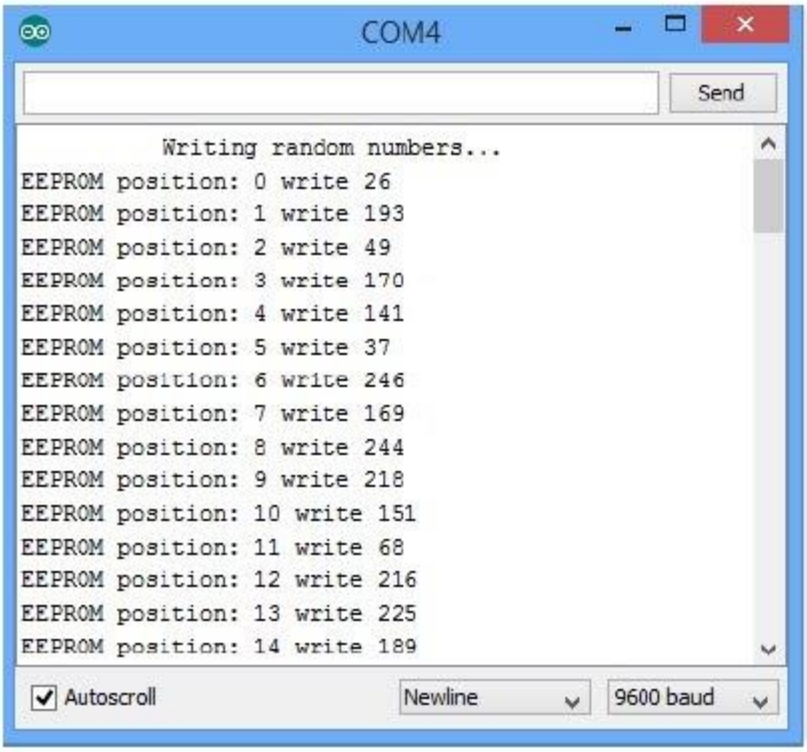
פונקציה זו קוראת את הנתון מתוך תא הזיכרון המוצבע על ידי המשתנה position ושומרת אותו בתוך המשתנה value.

סקיצה לקריאה וכתיבה בתוך ה EEPROM הפנימי

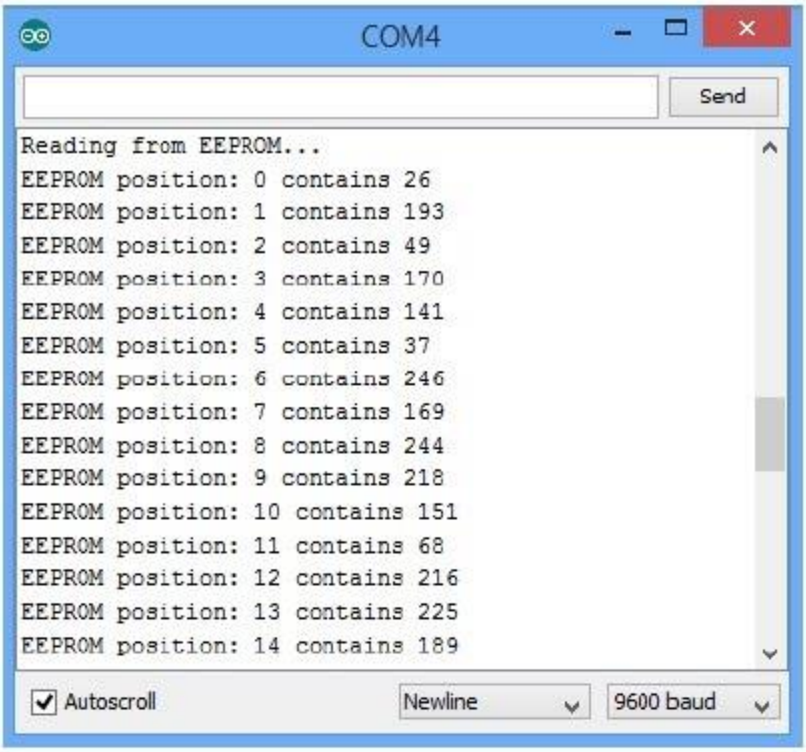
מס' שורה	הוראה	תיאור קצר
1	#include <EEPROM.h>	יבוא ספריות הזיכרון EEPROM
2	int zz;	
3	/****************************/	
4	void setup()	
5	{	
6	Serial.begin(9600);	קביעת קצב העברת הנתונים
7	randomSeed(analogRead(0));	הפקת מספר אקראי
8	}	
9	/****************************/	
10	void loop()	
11	{	
12	Serial.println("Writ random number")	
13	for (int i = 0; i < 1024; i++)	הדק האפשור של ה slave בנמוך
14	{	בחירת אחד מהאוגר בתוך הרכיב
15	zz = random(255);	מספר אקראי בין 0 ל 255
16	EEPROM.write(i, zz);	שמירת המספר בתוך הזיכרון הפנימי
17	}	
18	Serial.println();	
19	for (int a = 0; a < 1024; a++)	
20	zz = EEPROM.read(a);	קריאת נתון מתא הזיכרון הפנימי
21	Serial.print("EEPROM position: ");	
22	Serial.print(a);	
23	Serial.print(" contains ");	
24	Serial.println(zz);	הצגת המספר על המוניטור הטורי
25	delay(25);	
26	}	
27	}	

הסבר קצר לקוד הפרויקט

לאחר טעינה והרצת הסקיצה מספרים אקראיים יוצגו במוניטור הטורי כפי שמודגם באיור 5-5. המספרים ישמרו בתוך תאי בזיכרון של ה EEPROM הפנימי, החל מתא מס' 0 עד התא האחרון 1023. לאחר שמירת הנתונים, הסקיצה קוראת בחזרה את הנתונים מתאי הזיכרון ושולחת אותם לתצוגה במוניטור הטורי, כפי שודגם באיור 5-6.



איור 5-5: כתיבת לזיכרון



איור 5-6: קריאה מהזיכרון

פרק 5 מתנה

Fritzing

לעיתים קרובות כשאנו בונים מעגלים אלקטרוניים בארדוינו אנו משתמשים בארדוינו שלנו, מטריצה לחיבורים ומגוון רחב של רכבים אלקטרוניים. מחברים הכל בעזרת מגשרים וחוטים. באופן הזה אנו הופכים את האב טיפוס הפיזי למוצר בפועל.

לצורך פיתוח אב טיפוס לארדוינו, אם זה במישור התוכנה או החומרה נצטרך תוכנה שתעזור לנו באופן מהיר ויעיל. כאן, המקום המתאים לתוכנה Fritzing שהיא תוכנת קוד פתוח חינמית שבאה לעזור למפתחים, מעצבים, חוקרים וחובבי אלקטרוניקה לתכנת ולייצר פרויקטים מבוססים ארדוינו.

Fritzing היא כלי לשרטוט מעגלים חשמליים, לתעד אותם, לייצר תרשים למיקום וחיווט הרכיבים ביחד וגם לעריכת המעגל (PCB Layout) במידת הצורך. Fritzing משתמשת במטאפורה של מטריצה, כך שיהיה יותר קל להעביר סקיצת החומרה שבנינו לתוכנה. תכנות סקיצה מתאימה מכניסה חיים למעגל האלקטרוני שבנינו.

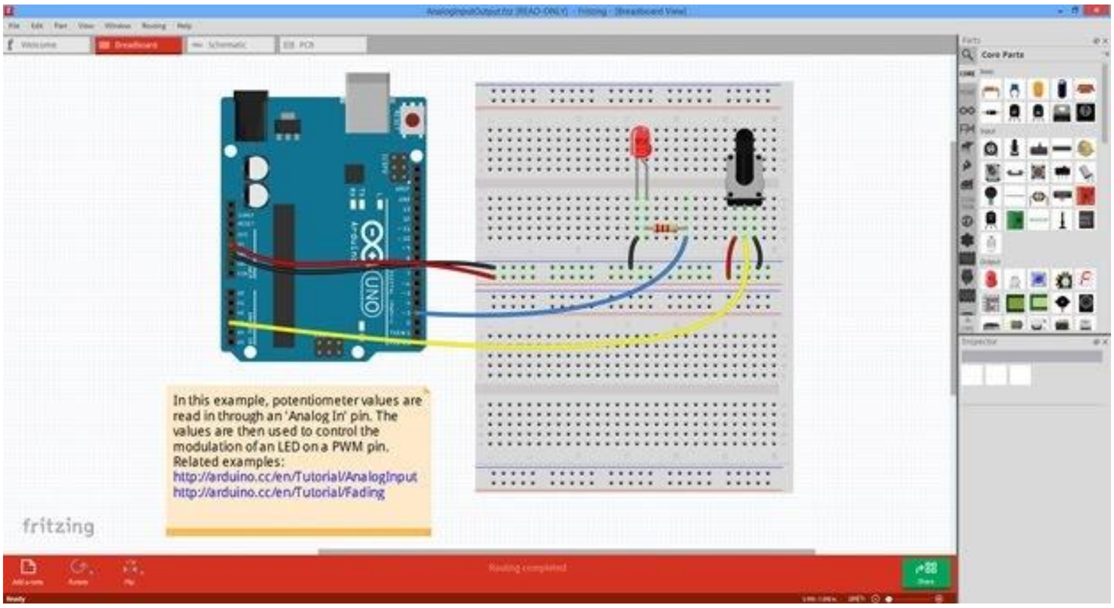
משטח העבודה של Fritzing

משטח העבודה של Fritzing פשוט ונוח לשימוש, והוא עושה את עיצוב המעגלים לאינטואיטיבי ומהיר.

לפני שנוכל להתחיל ולהשתמש בתוכנה, אנו חייבים להוריד אותה מהאתר הרשמי של Fritzing

. http://fritzing.org/download

באתר מובא גם הסבר כללי על אופן ההתקנה המתאימה לכל מערכת הפעלה. באיור הבא מתואר אחד מהפרויקטים של Fritzing(ראה להלן איור 1).



איור 1: משטח העבודה של Fritzing

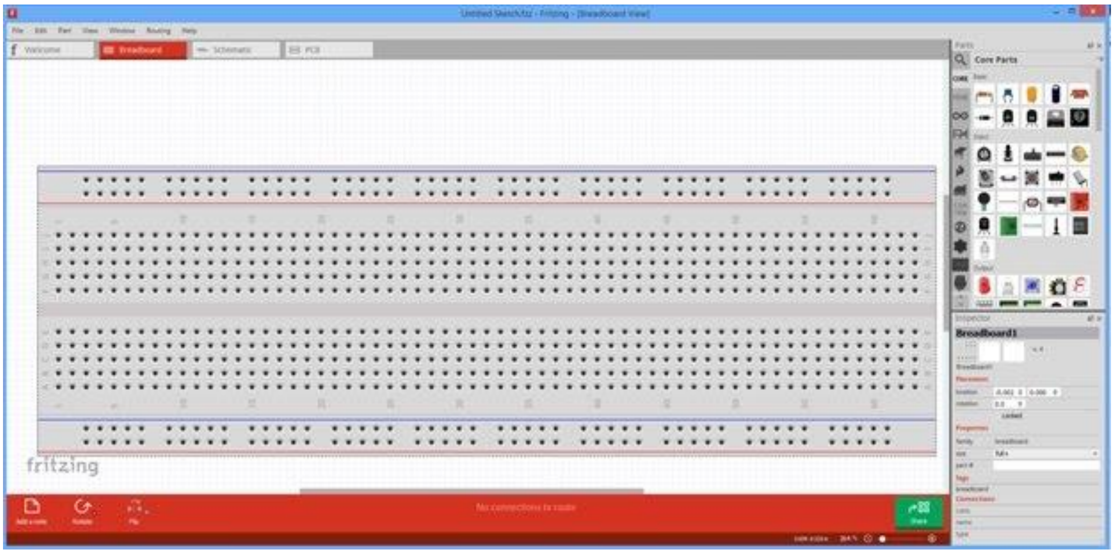
מאיור 1 אפשר לראות כמה מהרכיבים שבאמצעותם נוכל להרחיב ולתכנן כל מעגל שנרצה. נמצא גם השחקן העיקרי, לוח הארדוינו, המחובר דרך חוטים צבעוניים למטריצה שעליה תקועים כמה רכיבים אלקטרוניים, כמו נגד, לד אדום ונגד משתנה המחוברים ביניהם גם בעזרת חוטים צבעוניים. לצורך הסבר פעולת הרכיבים או פעולת המעגל אנחנו יכולים לכתוב הערות קטנות בצורה הדומה לפתקים צהובים.

הבניה או הרחבת המעגל האלקטרוני מתרחשת באמצעות גרירה ושחרור הרכיב הנבחר מאחת הספריות. פשוט לגרור את הרכיב המבוקש, למקם אותו במקום הנכון ואחר כך לשחרר אותו.

בספריות נמצא שמות הרכיבים העיקריים שהם הבסיס לפיתוח כל מעגל חשמלי כמו נגדים, דיודות, לדים, טרנזיסטורים, קבלים ועוד הרבה רכיבים. חוץ מזה נמצא שמות וכותרות מיוחדים, שמהווים את כל לוחות הארדוינו הקיימות.

השימוש במטריצה

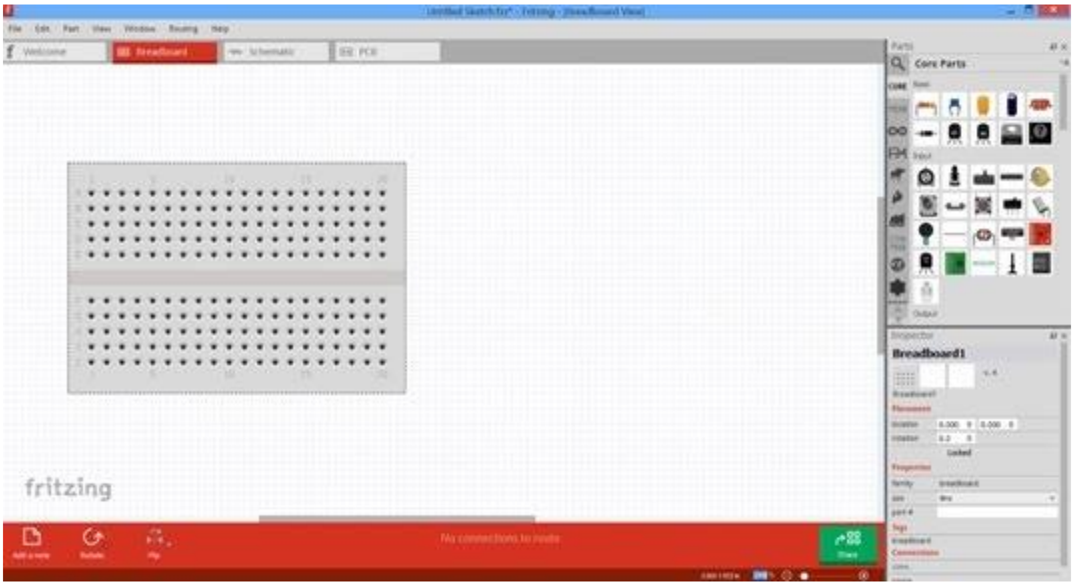
אפשר לבצע את בניית המעגל החשמלי על גבי לוח מטריצה. אחרי הפעלת תוכנת Fritzing נמצא לפנינו לוח מטריצה שעליו אפשר לגרור ולמקם רכיבי המעגל האלקטרוני. לוח המטריצה קיים בגדלים שונים, אפשר לבחור בגודל המתאים בהתאם לצרכים שלנו.



איור 2: המטריצה בגודל הסטנדרטי

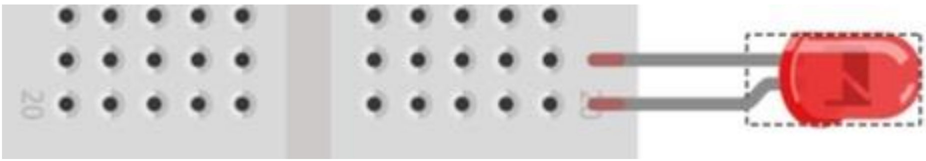
בהסבר הזה אציג בניית פרויקט קטן השולט בלד בעזרת לחצן. לשם כך נצמצם את גודל המטריצה ל גודל Tiny ונוציא לד אדום מהספרייה Basic .

כדי לראות את המאפיינים של המטריצה או כל רכיב אחר, פשוט נלחץ עם העכבר פעם אחת על הרכיב שברצוננו להציג את מאפייניו ראה איור 2. ברגע שלחצנו על הרכיב חלון ה inspector הוא יראה את המאפיינים השייכים לרכיב הנבחר.



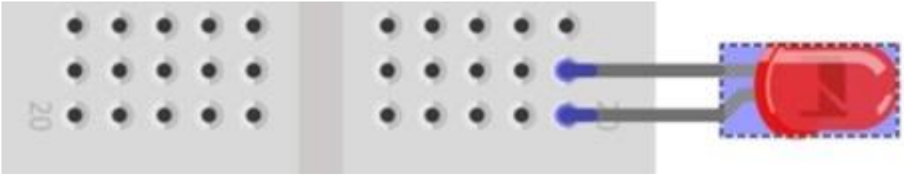
איור 3 : הספרייה Basic ומאפייני המטריצה

שני הדקי הלד חייבים לבוא במגע עם החורים של המטריצה, כדי שנוכל לחבר אליהם חוטים. כאשר אין מגע בין הדקי הלד וחורי המטריצה ההדקים יהיו בצבע אדום. ראה איור 4.



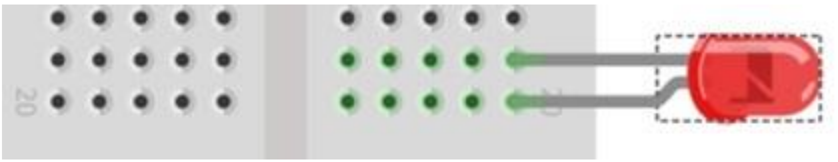
איור 4: לד אדום לחיבור

בעזרת העכבר גוררים את הלד לחורים שברצוננו לתקוע אותו בהם, ואז צבע קצה ההדקים הופך מאדום לסגול. ראה איור 5.



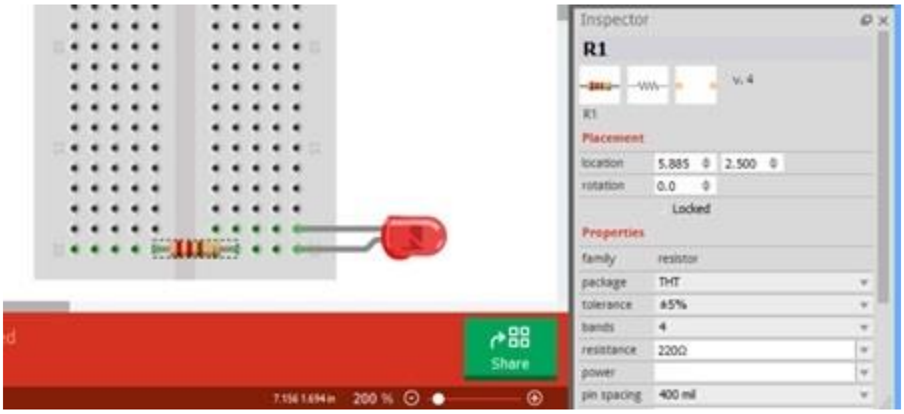
איור 5: ליד ממוקם נכון בחורים

אם נשחרר את הלד, קצה ההדקים מחליפים צבע לירוק וגם כל שורות החורים המחוברים יחד מוארים בצבע ירוק, באופן הזה נוצר חיבור בין הדקי הלד וכל חור נצבע בירוק. ראה איור 6.



איור 6 : לד מחובר ומוכן

באותה דרך נוכל למקם ולחבר את כל הרכיבים הדרושים לפרויקט שלנו. לדוגמה חיבור נגד טורי ללד, אחרי שמוציאים את הנגד מהספרייה Basic וממקמים אותו על המטריצה, משנים את הערך שלו לגודל המתאים לחישובים שלנו, במקרה הזה נבחר בנגד של . שינוי ערך הנגד או כל רכיב אחר נעשה בחלון מאפייני הרכיב שנבחר. ראה איור 7.



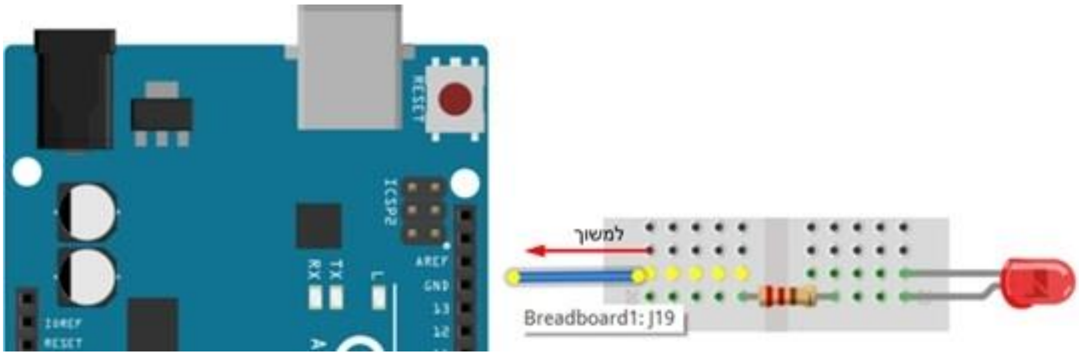
איור 7: מאפייני הרכיב שנבחר " נגד "

שים לב כאשר ממקמים את הנגד במקומו כל החורים הנמצאים במגע עם הנגד ועם כל רכיב אחר מואר בצבע ירוק. לבסוף נציג לכם את אופן החיבור החשמלי בין המטריצה ללוח הארדוינו על ידי העברת מוליך בין המטריצה ללוח הארדוינו. לשם כך נלחץ על הלחצן השמאלי של העכבר כאשר מצביע העכבר נמצא מעל החור שבמטריצה שברצוננו לחבר אותו לארדוינו.

כאשר לוחצים על לחצן העכבר השמאלי כל שורת החורים המחוברים ביחד במטריצה מוארים בצבע צהוב. ראה איור 8 .

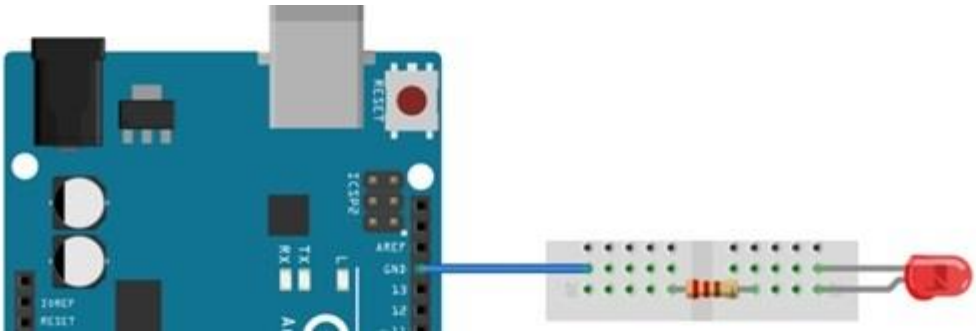
איור 8: חיבור חוטים מוליכים למטריצה

ממשיכים ללחוץ על העכבר ופשוט מושכים את החוט היוצא מהחור עד למקום הנכון בלוח הארדוינו.
ראה איור 9 .



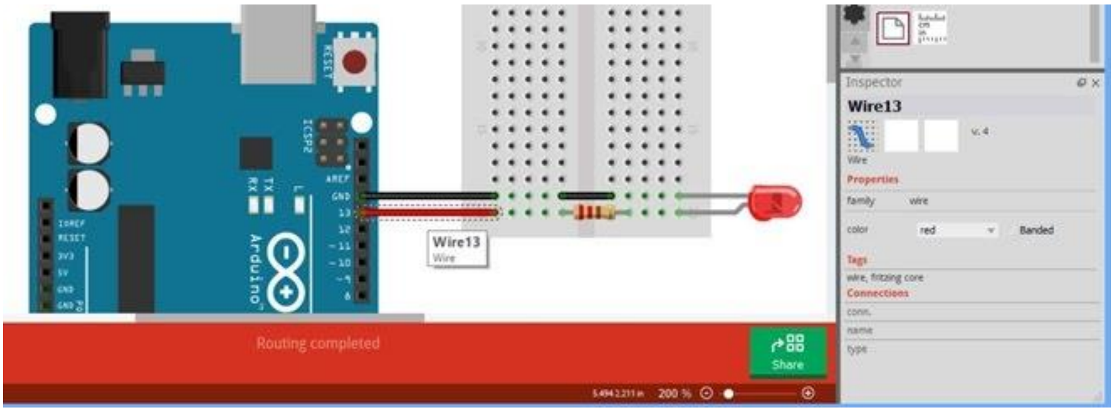
איור 9:החיבור החשמלי בין המטריצה ולוח הארדוינו נעשה

אחרי שהחוט הגיע למקום הנכון שאליו רוצים לחבר את החוט משחררים את לחצן העכבר.ראה איור
.10



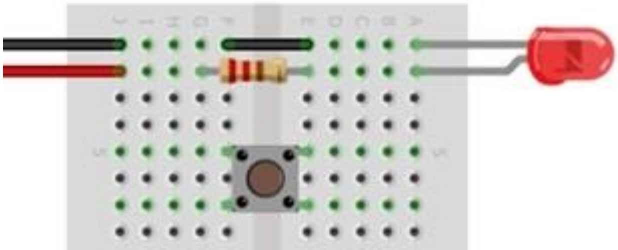
איור 10 :חיבור חשמלי בין המטריצה ללוח הארדוינו

באיור 10 אפשר לראות את חיבור הלד למטריצה. מהאיור אפשר לראות ששינינו את צבע החוטים והתאמנו אותם לסוג החיבור, כך שהאדמה בצבע שחור, וצבע הבקרה אדום. שינוי הצבע מתרחש בחלון ה inspector כאשר מופיעה בחלון זה המילה " Wire " . ראה איור 11.



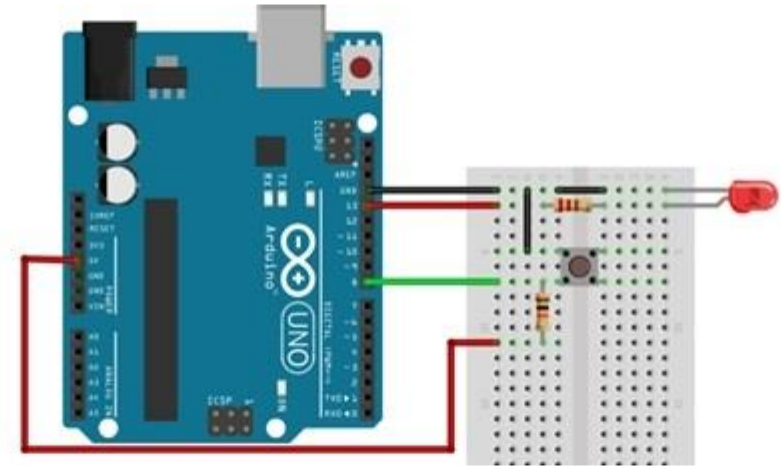
איור 11: חיבור הלד עם הארדוינו

אחרי חיבור הלד לארדוינו נחבר את הלחצן עם נגד מושך למעלה לאחת מכניסות הארדוינו. אפשר למצוא את הלחצן, שסימנו Pushbutton גם בספריה Basic תחת הקטגוריה Core. מושכים את הלחצן מתוך הספריה וממקמים אותו על המטריצה. ראה איור 12 ושים לב לחורים שבצבע ירוק.



איור 12: מיקום הלחצן על המטריצה

אחרי מיקום הלחצן במקומו מחברים אליו את הנגד המושך למעלה. לשם כך גוררים נגד מספרית Basic החוצה ונותנים לו את הערך של K10   מחלון inspector ולבסוף מחברים את הלחצן לערכת הארדוינו בעזרת צבעים מתאימים למוליכים. בצורה הזאת הכנו את המעגל החשמלי על גבי המטריצה ראה איור 13.

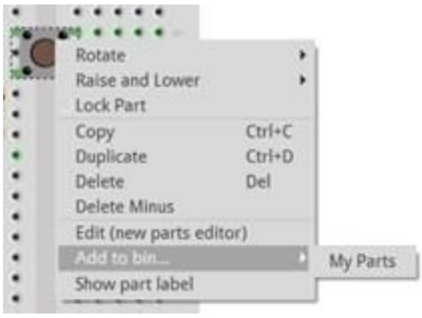


איור 13 : חיבור הלחצן לארדוינו

טיפ מעניין לעבודה עם Fritzing, כאשר אנחנו עומדים להשתמש באותם חלקים אלקטרוניים בפרויקטים עתידיים עדיף לשמור אותם בספריה מסוימת. בשיטה הזאת אנו יכולים למצוא את הרכיבים במהירות ובנוחיות גדולה במקום שתמיד נחפש אותם בספריה הנמצאת תחת קטגוריה מסוימת. בצד ימין בתחום הקטגוריות השונות יש קטגוריה מיוחדת בשם Mine ובתוכה יש את הספרייה My Parts. בתוך הספרייה הזאת אנחנו יכולים לשמור את כל הרכיבים אשר נשתמש בהם לעתים קרובות. ראה איור 14.

איור 14 : הספרייה My Parts

אופן שיוך רכיב מסוים לספרית My Parts הוא פשוט מאד: בעזרת העכבר בוחרים את אחד הרכיבים ולחיצה על הלחצן הימני של העכבר תפתח תפריט ההקשר " context menu" שבעזרתו אנחנו יכולים לערוך את הרכיב כמו לסובב אותו,להעתיק, למחוק וגם להוסיף אותו לספריית My Parts. ראה איור 15.

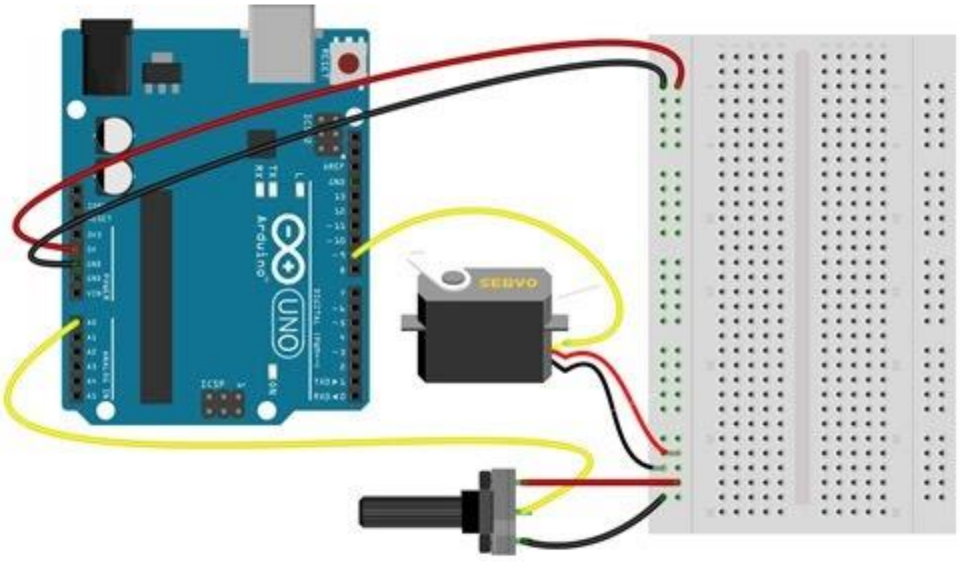


איור 15: הוספת רכיב לספרייה My Parts

איור 16 מראה את ספריית My Parts אחרי הוספת הרכיבים לתוכה.

איור 16 : ספריית My Parts

ב Fritzing יש דרך מעניינת מאד בהעברת החוטים בין הרכיבים. עד עכשיו העברנו או החוטים בדרך ישרה ועם זוויות. על מנת שהעסק ייראה קרוב יותר למציאות, Fritzing מציע דרך נוספת להעברת החוטים בין הרכבים בצורת קשת . הדבר ייראה כמו באיור 17.

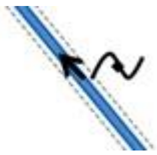


איור 17: חיבור בצורת קשת

בדרך כלל אם רוצים לעשות עיקול במקום מסוים בחוט, מקבל העיקול צורת מפרק, לצורך זה אנו מעבירים את העכבר על המקום המבוקש בחוט עד שמצביע העכבר מראה את הסימן המוצג באיור 18 ואחרי זה מזיזים את העכבר לכיוון מסוים כאשר הלחצן השמאלי שלו לחוץ.

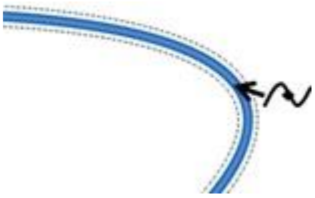
איור 18: מצביע העכבר עם סימן המפרק

כדי לקבל חוט בצורת קשת, נעביר את העכבר על המקום המבוקש בחוט ונחזיק את המקש Ctrl לחוץ עד שמצביע העכבר מציג את הסימן המוצג. ראה איור 19.



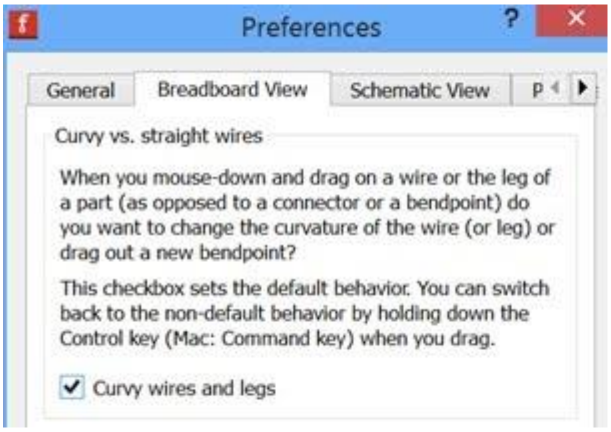
איור 19: מצביע העכבר עם סימן העיקול

כדי לקבל קשת יש למשוך את העכבר לכיוון מסוים כאשר הלחצן השמאלי שלו לחוץ איור 20.



איור 20 : חוט בצורת קשת

אם ברצוננו להשתמש תמיד בפונקציה הזאת, אפשר לעשות את זה דרך חלון מאפיינים "Preferences" הנמצא בתפריט Edit ראה איור 21. כאשר הריבוע של Curvy wires and legs מסומן, הפונקציה של חוט עקום תהיה מופעלת תמיד.



איור 21: הפעלת פונקציית עיקום החוט

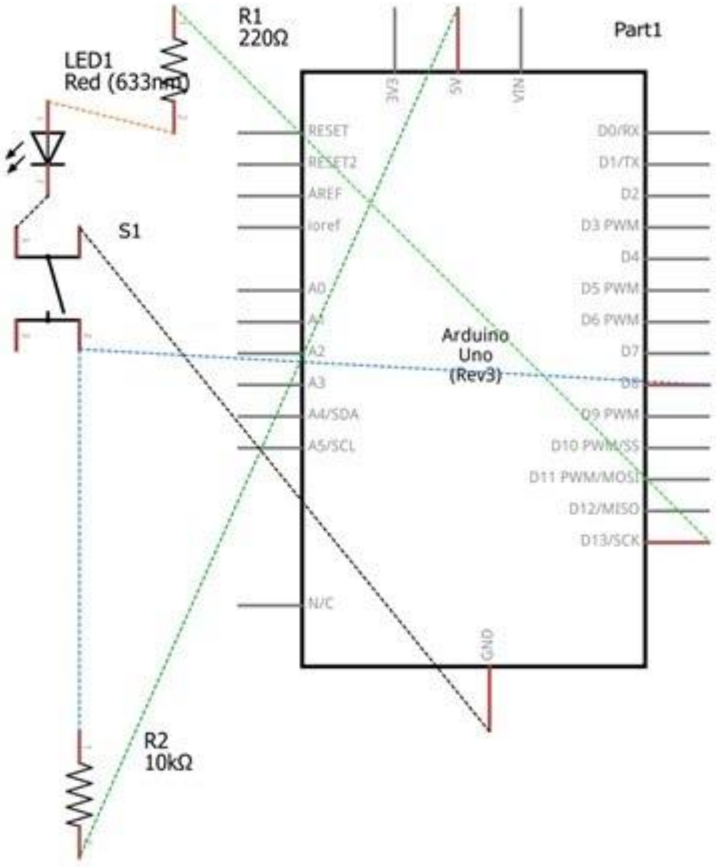
הסכימה החשמלית

מתחת לתפריט הראשי של התוכנה נמצא סרגל כלים העוזר לנו לנווט בין חלקי התוכנה השונים. ראה איור 22.

איור 22: סרגל הכלים

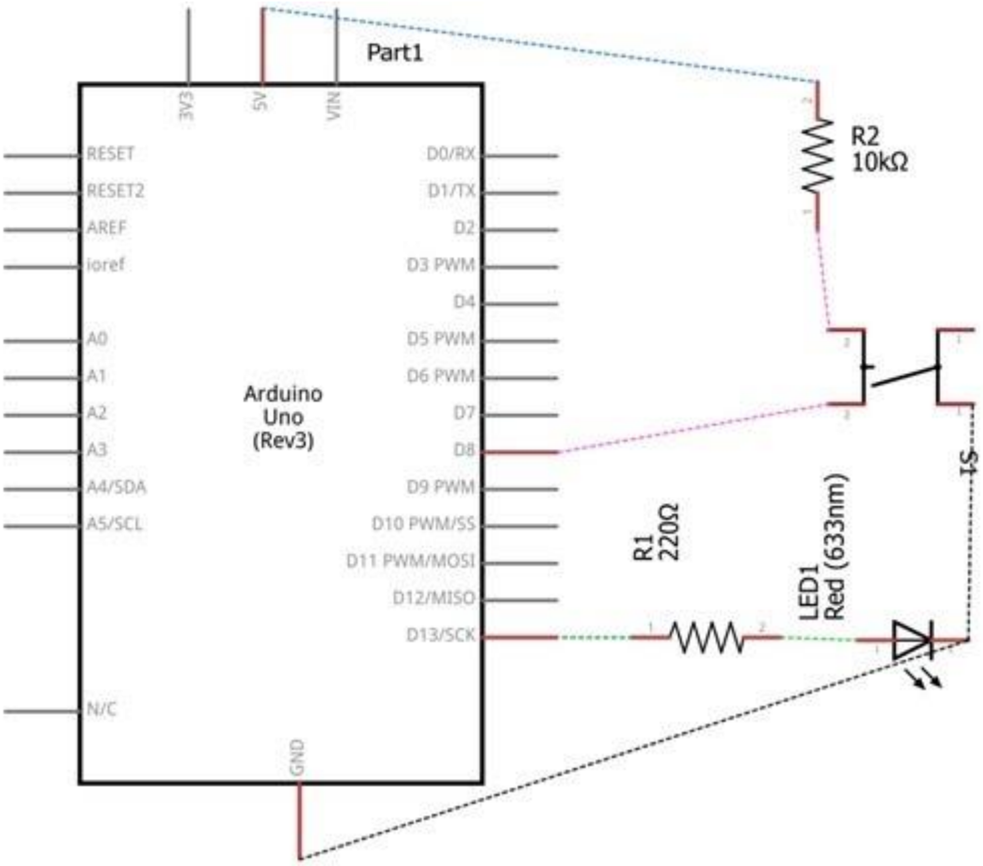
לחיצה על האייקון Schematic מעבירה אותנו לתרשים הסכמתי של המעגל שבנינו על המטריצה.

התרשים הסכמתי מציג בדיוק את החיבורים בין הרכיבים שבנינו אותם על לוח המטריצה. בהתחלה, החיבורים מצטלבים אחד עם השני ולא מסודרים. ראה איור 22. זה הינו דבר טבעי בתוכנה וצריך לסדר ולשנות את החיבורים והחיווטים.



איור 23: התרשים החשמלי ההתחלתי של התוכנה

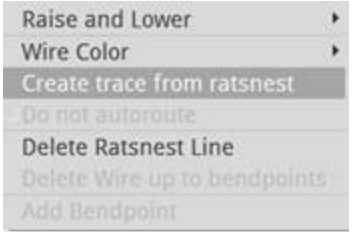
סידור הרכיבים ומיקומם נעשה באופן הגיוני וברור. עדיף בהתחלה למקם את הרכיבים ליד הדקי המיקרו בקר אליהם הם קשורים. ראה איור 24 .



איור 24: תרשים חשמלי ברור

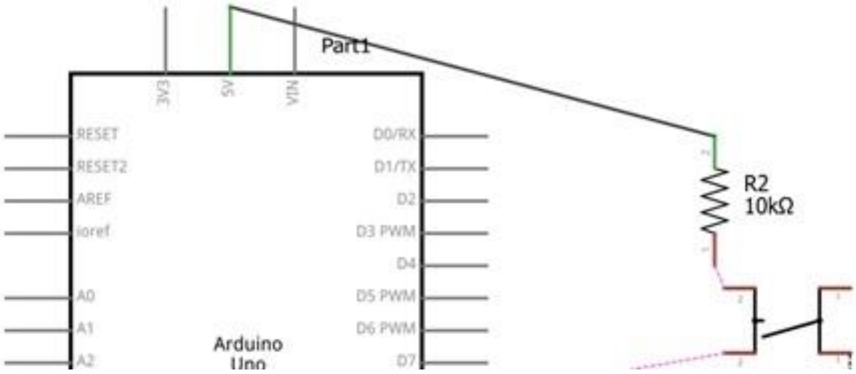
ערכי הנגדים ושמות הרכיבים לא מסודרים באופן קריא וברור. כדי לפתור בעיה זו ניתן לסדר אותם על ידי הזזת המיקום שלהם וגם סיבובם כך שיהיו מסודרים באופן ברור ומשויכים נכון לרכיבים.

הקווים המקווקווים הם למעשה החיבורים הקצרים ביותר בין הרכיבים הרלוונטיים. למעשה אלה הם קווי אוויר. לחיצה ימנית בעכבר על אחד מקווי האוויר האלה פותחת לפנינו חלון עם תפריט קישורים המתואר באיור 25.



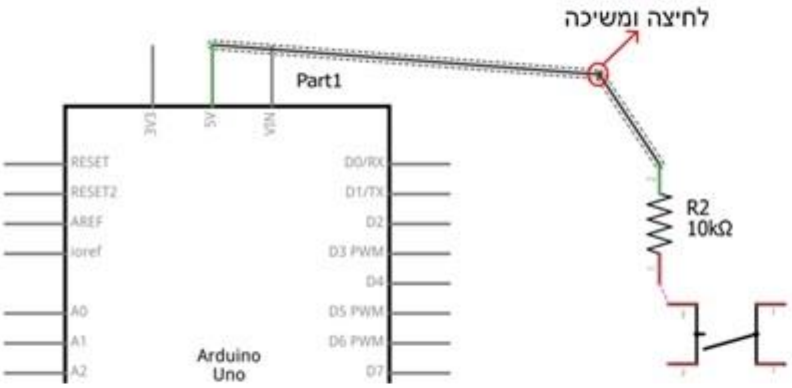
איור 25: תפריט הקישורים של קוו מקווקו "אוויר"

לחיצה על Create trance from ratsnest מחליפה את הקו מקוו מקווקו לקו רגיל המראה על חיבור הרכיבים ביחד. נדגים כאן את החלפת הקו האווירי של הלחצן והנגד המושך למעלה. ראה איור 26.



איור 26: החלפת קוו אווירי למוליך

מוליכים המצוירים בתרשים חשמלי חייבים שיהיו מועברים בקווים ישרים וזוויות ישרות גם בלי הצטלבויות, את זה אנחנו יכולים לעשות על ידי לחיצת הכפתור השמאלי של העכבר על נקודה באחד הקווים. הלחיצה על נקודה בקו מייצרת נקודת כיפוף שבקלות אפשר להזיז אותה בעזרת העכבר כל עוד הכפתור השמאלי לחוץ איור 27.

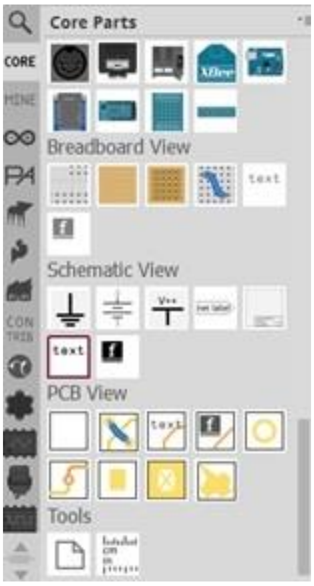


איור 27: יצירת נקודת כיפוף

אחרי עזיבת הנקודה קל מאד אחר כך לשנות את המיקום של המוליך. את שאר הקווים עושים באותה דרך על ידי נקודת הכיפוף ומשיכתה לכיוון הרצוי.

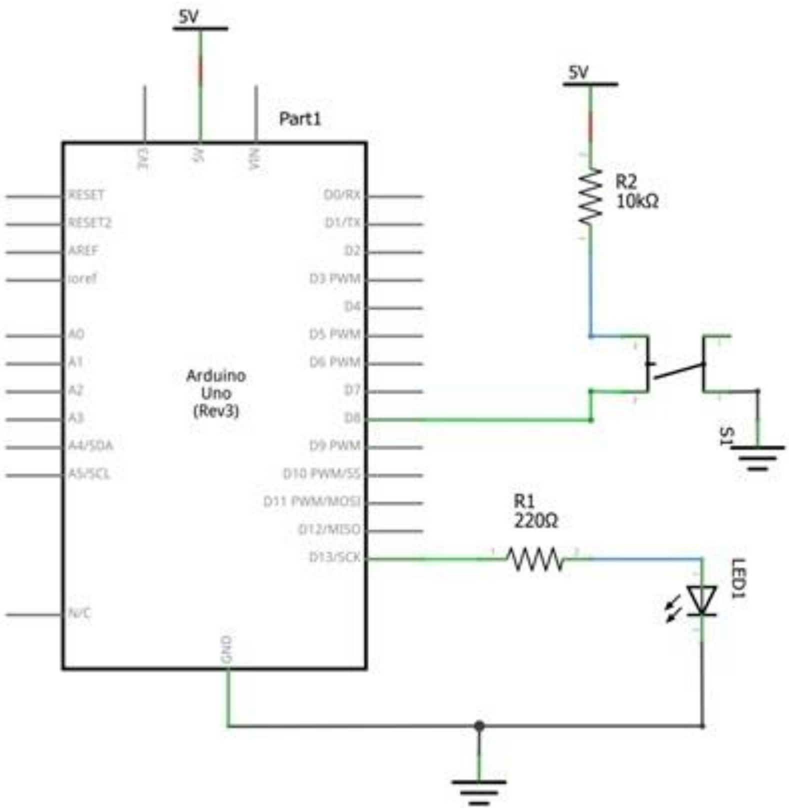
איור 28: סמל האדמה וסמל מתח ה Vcc

את הסמלים האלה ועוד, מופעלים רק כשעובדים במוד העבודה Schematic , ונמצאים בקטגוריה Core תחת הספרייה Schematic View . ראה איור 29.



איור 29: הספרייה Schematic View

אחרי מיקום הקווים, צביעתם וסידור הרכיבים, המעגל יראה יותר ברור מסודר ומובן איור 30.



איור 30: התרשים החשמלי הסופי

לדבר הזה קוראים " Physical computing ". במובן הרחב ביותר, פירושו בניית מערכות פיזיות אינטראקטיביות על ידי השימוש בתוכנה וחומרה שיכולות לחוש ולהגיב לעולם האנלוגי בכדי להבין את מערכת היחסים שבין בני האדם לעולם הדיגיטלי. המונח מתאר שימוש מעשי בחיישנים ובקרים זעירים המתרגמים קלט אנלוגי למערכת מחשבים דיגיטליים , או לבקרת התקנים אלקטרו מכאנית כגון מנועים, servos, תאורה או חומרה אחרת.