תוכן עניינים

הצעת פרויקט באלקטרוניקה

כיתה: י״ב

נושא: מכונת מכירת שתייה

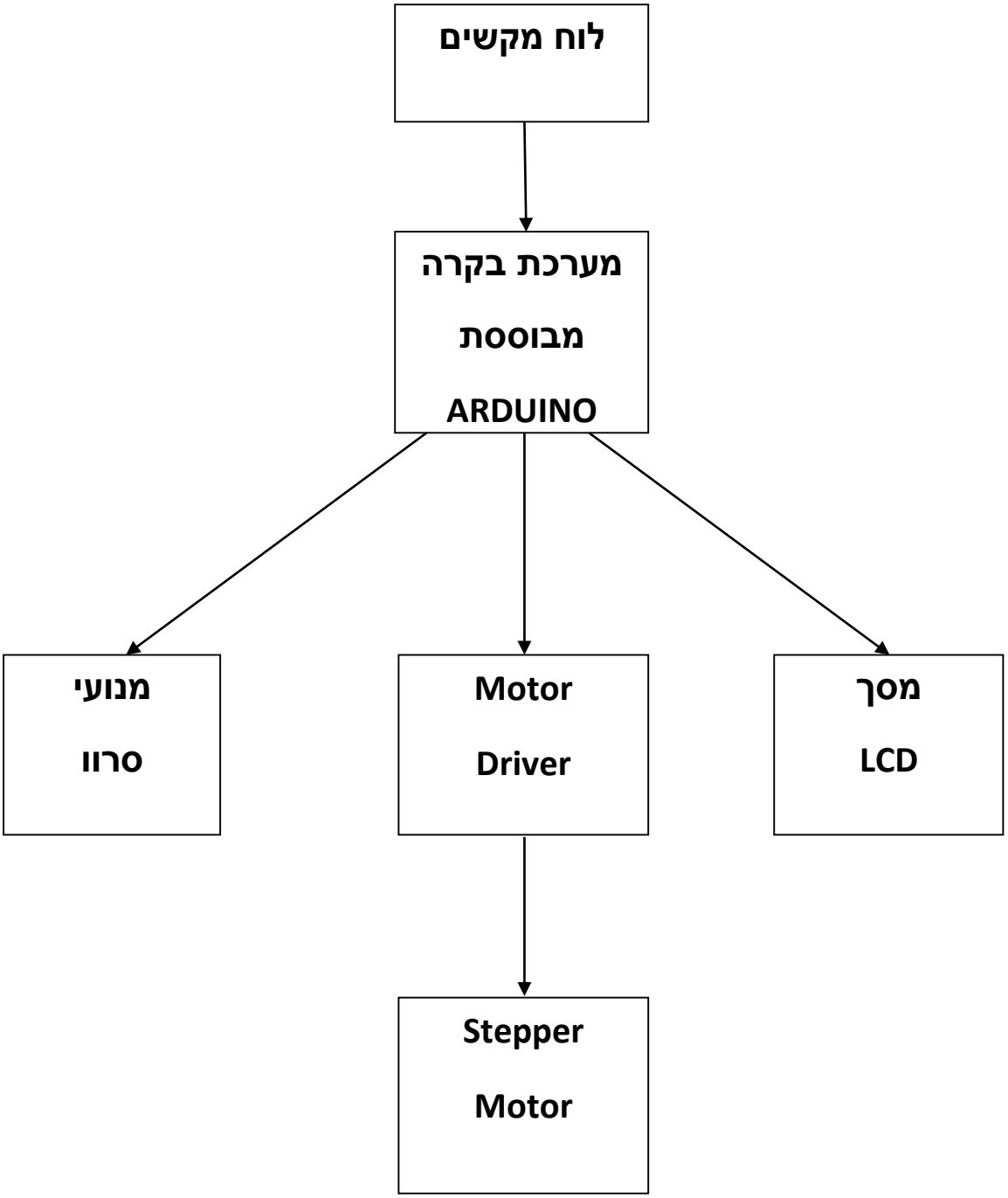
תיאור מפורט שלנושא העבודה או פרויקט

תכנון ובנייה של מערכת בקרה של מכונת מכירה מזערית מאפס באמצעות Arduino Mega, ארבעה servos , ואת LCD 2 X 16 . המכשיר, המכונה כולל לוח מקשים לבחירה. בין אם זה קולה, פאנטה , מים, או פריטים שונים, האפשרויות הן אינסופיות. פשוט, בחר מוצר וחזור.

הערות המורה:

- Arduino Mega board
- 4 X Micro Servo Motor 9g
- LCD 16 X 2
- Keypad
- Power supply

תרשים מלבנים:



הבעת תודה

בראש השבח לאל שהקל לנו נתיב המדע .

אני מודה להורים שלי שתמכו בי בדרך הקשה הזה

תביע תודה למנחה בסאם עבד אלקאדר על ליווי הפרויקט ועל סבלנותו הרבה, מסירותו לקידום הפרויקט והבעת נכונותו לעזור בכל עת, ושיתפה פעולה בצורה מלאה .

ולסיום תודתי מעומק הלב למנהל בית הספר שהוכן לנו האווירה המתאימה להשלים הפרויקט הזה .

מקדיש את הפרויקט הזה לכל מי שעזר תמך ושיתף אותי בדעותיו במהלך העבודה על הפרויקט.

הקדשה שלי

אנו אסירי תודה למנחה שלנו ,בסאם עבד אל קאדר , אשר לווה אותנו למשך כל הדרך ובעזרת הידע הרב שלו הנחה אותנו בדרך הנכונה והיעילה ביותר כדי להצליח לבנות ולתכנת את הפרויקט שלנו.

מערכת הבקרה

בסעיף זה נציג ונתמקד בשבב הקטן שבלב ה ***Arduino***- 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. כמו שהזכרנו מיקרו-בקר הנו מחשב מזערי. כדי שמערכת תוגדר כמחשב, עליה לכלול את שלושת המרכיבים הבאים.

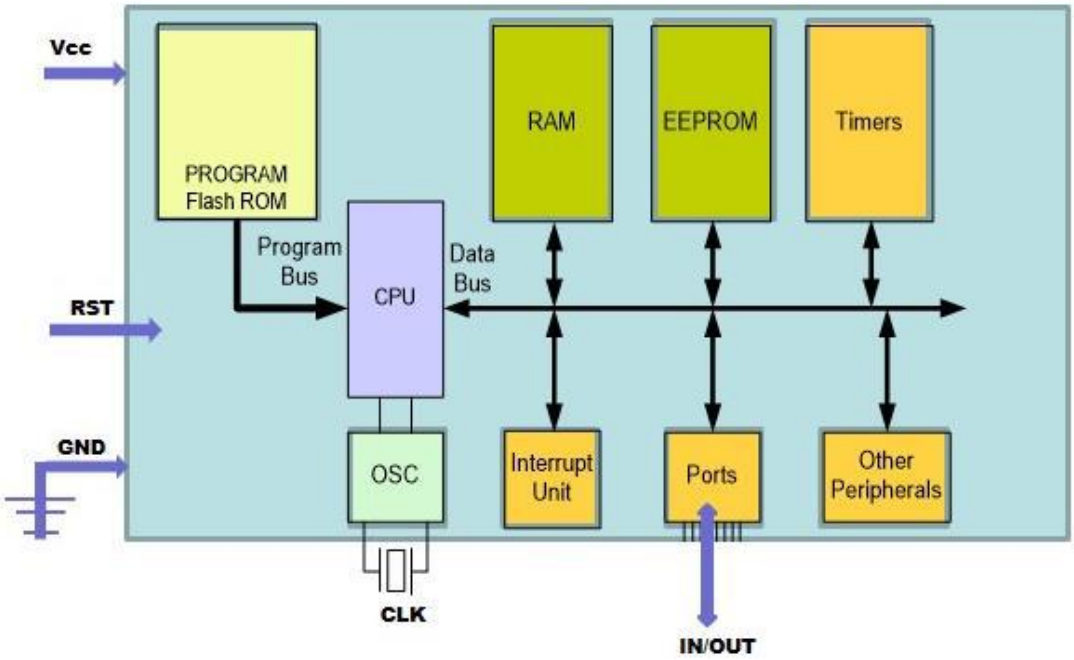
- זיכרון(RAM) להפעלת התוכנית.
- מעבד(CPU) שמעבד כל הנתונים ומפעיל הפורטים בהתאם.
- יחידות כניסה/יציאה(O/I) מקשרת הבקר עם העולם החיצוני.

שלשת המרכיבים הבסיסיים של מיקרו בקר/מחשב

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.

האיור הבא מציג תרשים מלבני פשוט של *AVR 8-bit* -



במרכז *AVR 8-bit* יש 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 5 V.5 ל 5 V.4מתח 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, זיכרונות, 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.

אתחול RESET

הדק ה ***RESET*** - מספק מנגנון לאיפוס /הפעלה מחדש של המיקרו-בקר. פונקצית ***RESET*** מתרחשת כאשר הדק זה נמצאת במצב נמוך. הדק מספר 30 ב - Atmega2560מהווה כניסה ייעודית המוקדשת לפעולת האיפוס.

XTAL2 ו XTAL1

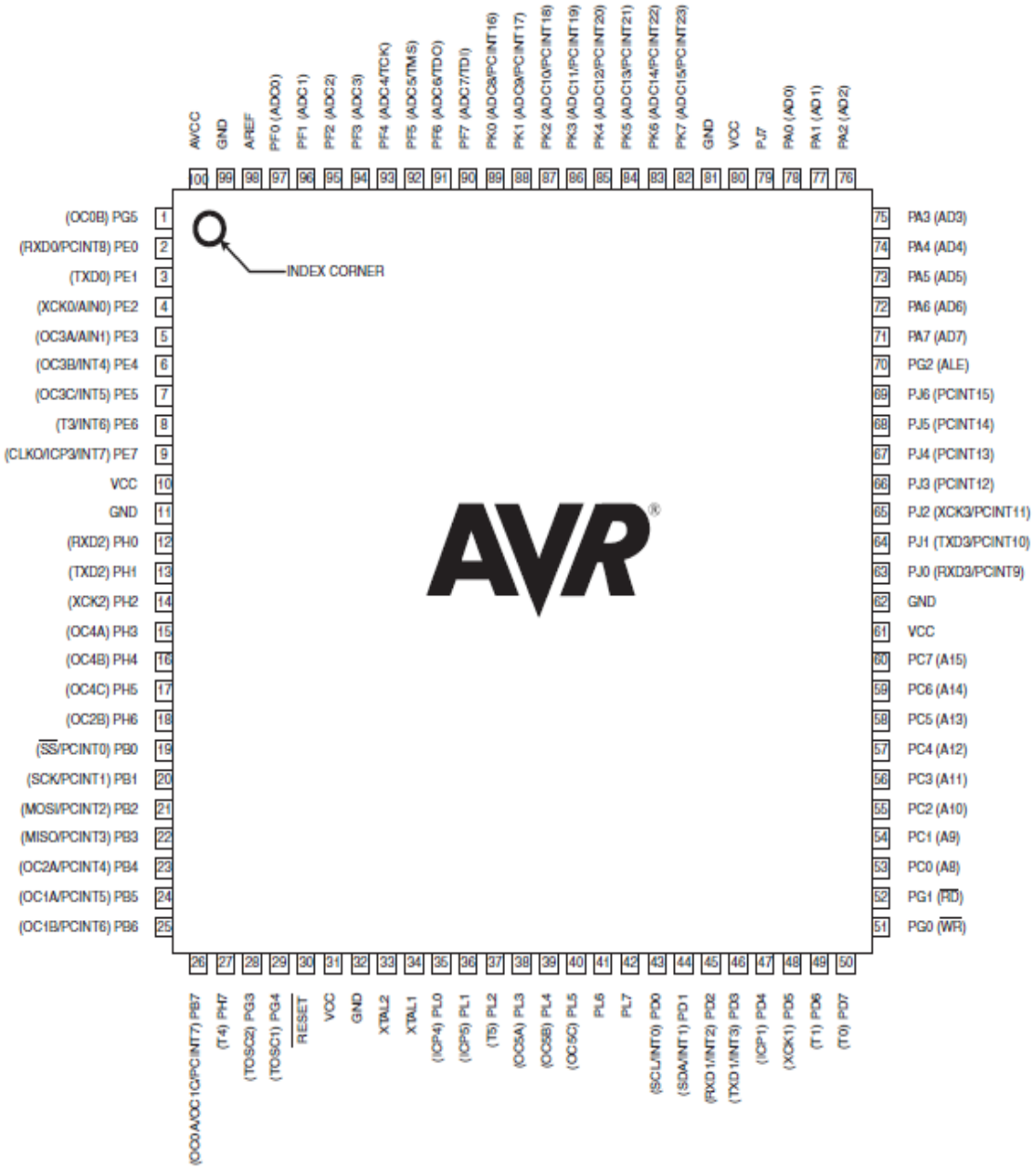
ל***Atmega2560*** - ישנם שני הדקים המוקדשים ל ***XTAL1*** ו***XTAL2*** . .לעומת זאת ב, ***Atmega328*** -הדקים אלה מרובבים עם הדקי הקלט/פלט הכלליים ***PB6*** , ו. ***PB7*** -הבחירה בין תפקוד ההדקים כקלט/פלט או הדקי שעון הזנה למערכת, נקבעת בתכנות על ידי קביעת תצורת נתיך בחירת השעון "3". ***CKSEL0-*** כאשר נבחרה האפשרות של גביש או מתנד קראמי ***PB6*** ,ו ***PB7*** 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.

הדקי הבקר

באיור הבא אפשר לראות את הדקי הבקר כפי שתוכנית פארד ואינו רואה.

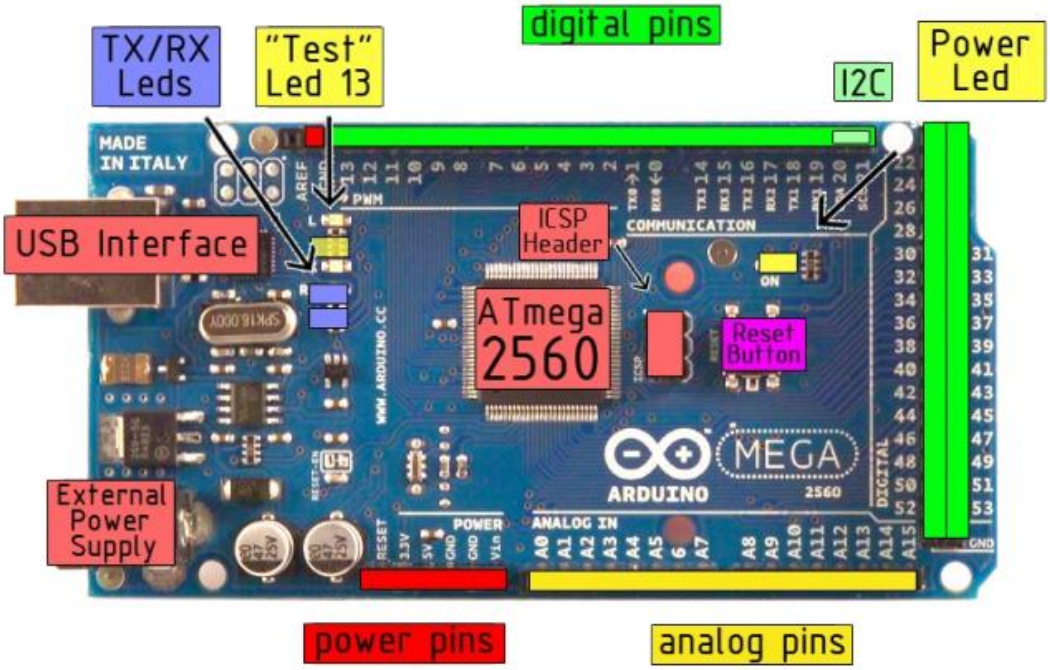

AVCC
GND
AREF
PF0 (ADC0)
PF1 (ADC1)
PF2 (ADC2)
PF3 (ADC3)
PF4 (ADC4/TCK)
PF5 (ADC5/TMS)
PF6 (ADC6/TDO)
PF7 (ADC7/TDI)
PK0 (ADC8/PCINT16)
PK1 (ADC9/PCINT17)
PK2 (ADC10/PCINT18)
PK3 (ADC11/PCINT19)
PK4 (ADC12/PCINT20)
PK5 (ADC13/PCINT21)
PK6 (ADC14/PCINT22)
PK7 (ADC15/PCINT23)
GND
VCC
PJ7
PA0 (AD0)
PA1 (AD1)
PA2 (AD2)
100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76
(OC0B) PG5 1
(RXD0/PCINT8) PE0 2
(TXD0) PE1 3
(XCK0/AIN0) PE2 4
(OC3A/AIN1) PE3 5
(OC3B/INT4) PE4 6
(OC3C/INT5) PE5 7
(T3/INT6) PE6 8
(CLKO/ICP3/INT7) PE7 9
VCC 10
GND 11
(RXD2) PH0 12
(TXD2) PH1 13
(XCK2) PH2 14
(OC4A) PH3 15
(OC4B) PH4 16
(OC4C) PH5 17
(OC2B) PH6 18
(SS/PCINT0) PB0 19
(SCK/PCINT1) PB1 20
(MOSI/PCINT2) PB2 21
(MISO/PCINT3) PB3 22
(OC2A/PCINT4) PB4 23
(OC1A/PCINT5) PB5 24
(OC1B/PCINT6) PB6 25
INDEX CORNER
AVR
75 PA3 (AD3)
74 PA4 (AD4)
73 PA5 (AD5)
72 PA6 (AD6)
71 PA7 (AD7)
70 PG2 (ALE)
69 PJ6 (PCINT15)
68 PJ5 (PCINT14)
67 PJ4 (PCINT13)
66 PJ3 (PCINT12)
65 PJ2 (XCK3/PCINT11)
64 PJ1 (TXD3/PCINT10)
63 PJ0 (RXD3/PCINT9)
62 GND
61 VCC
60 PC7 (A15)
59 PC6 (A14)
58 PC5 (A13)
57 PC4 (A12)
56 PC3 (A11)
55 PC2 (A10)
54 PC1 (A9)
53 PC0 (A8)
52 PG1 (RD)
51 PG0 (WR)
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
(OC0A/OC1C/PCINT7) PB7
(T4) PH7
(TOSC2) PG3
(TOSC1) PG4
RESET
VCC
GND
XTAL2
XTAL1
(ICP4) PL0
(ICP5) PL1
(T5) PL2
(OC5A) PL3
(OC5B) PL4
(OC5C) PL5
PL6
PL7
(SCL/INT0) PD0
(SDA/INT1) PD1
(RXD1/INT2) PD2
(TXD1/INT3) PD3
(ICP1) PD4
(XCK1) PD5
(T1) PD6
(T0) PD7

נתוני הבקר מובאים בטבלה הבאה :

Microcontroller	*Atmega2560*
Operating Voltage	*5V*
Input Voltage (recommended)	*7-12V*
Input Voltage (limits)	*6-20V*
Digital I/O Pins	*54 (of which 14 provide PWM output)*
Analog Input Pins	*16*
DC Current per I/O Pin	*40 mA*
DC Current for 3.3V Pin	*50 mA*
Flash Memory	*256 KB of which 8 KB used by bootloader*
SRAM	*8 KB*
EEPROM	*4 KB*
Clock Speed	*16 MHz*

בקר ארדואינו מיגא (MEGA):

זהו כרטיס מסדרת ארדואינו עם עם מיקרו בקר **Atmega2560** . בכרטיס 54 הדקים דיגיטאליים כשכל אחד מהם יכול להיות קלט או פלט לפי תכנות שלנו, ועוד 16 הדקים אנאלוגיים. חיבור **ICSP(In Circuit Serial Programming) , USB** , וכפתור איפוס.

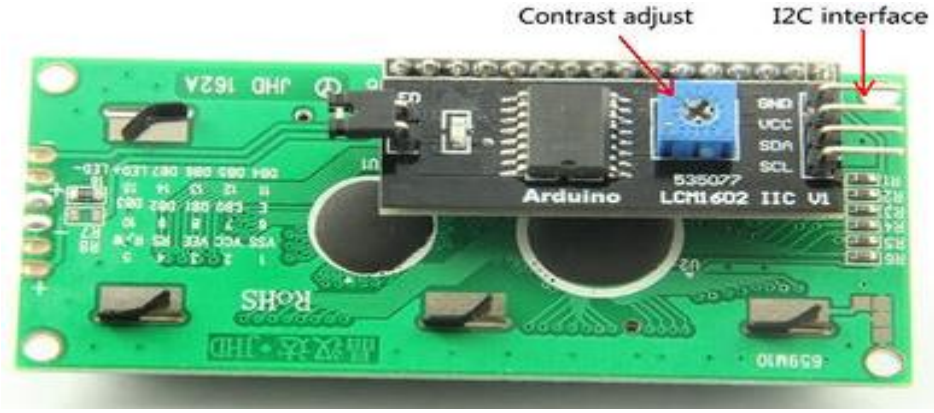


LCD1602 I2C

כפי שכולנו יודעים, אם כי ***LCD*** וכמה תצוגות אחרות מעשירים מאוד את האינטראקציה בין אדם למכונה, הם חולקים חולשה משותפת. כאשר הם מחוברים לבקר, מספר ***IO***s יהיה כבוש של הבקר אשר אין כל כך הרבה יציאות חיצוניות. כמו כן הוא מגביל פונקציות אחרות של הבקר. לכן, ***LCD1602*** עם אוטובוס ***I2C*** פותחה כדי לפתור את הבעיה.

אוטובוס ***I2C*** הוא סוג של אוטובוס טורי שהומצא על ידי **PHLIPS**. זהו אוטובוס סדרתי בעל ביצועים גבוהים אשר יש פסק האוטובוס גבוהה או נמוכה מהירות המכשיר סינכרון נדרש על ידי מערכת מרובת מארח. פואנטיומטר כחול על ***I2C*** ***LCD1602*** (ראה את הדמות להלן) משמש כדי להתאים את התאורה האחורית לתצוגה טובה יותר. ***I²C*** משתמשת רק בשני קווי ניקוז דו כיווני פתוח, קו נתונים טורי ***(SDA)*** ו קו שעון סידורי ***(SCL)***, עצר עם נגדים. מתחים אופייניים המשמשים הם +5 V או +3.3 V למרות מערכות עם מתחים אחרים מותרים.

התצוגה דורשת 16 סיכות כדי לשלוט ולכתוב למסך. בארדואינו יש מספר מוגבל של סיכות דיגיטליות ***I / O*** וחיבור התצוגה ישירות אינה מעשית. כדי להפוך את המכשיר שימושי, ספקים להוסיף הלוח המרחיב ***LCM1602***. לוח זה כולל שבב ***PCF8574***. זהו יציאת 8 סיביות מורחבות . הארדואינו שולח 8 סיביות, אחד בכל פעם, על ממשק טורי לשבב זה. שבב אוספת אותם ואז מכניס את כל 8 החוצה במקביל הסיכות המתאימות על המסך.



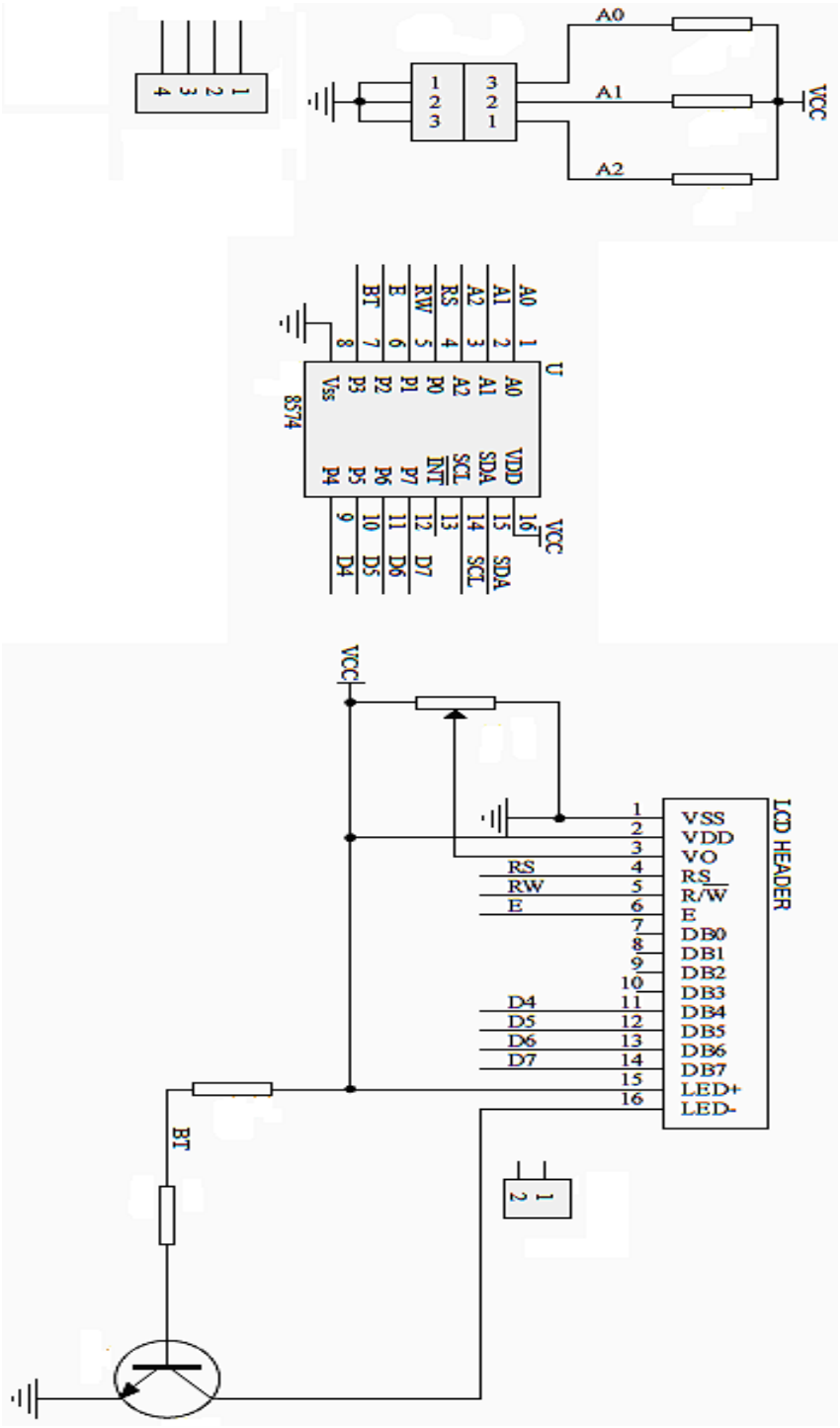
מאפיינים

- ממשק : I2C
- I2C כתובת : 0x27.
- פין הגדרה : **VCC, GND, SDA, SCL.**
- תאורה אחורית (ירוק עם צבע לבן).
- מתח אספקה : V5.
- גודל : 27.7mm X 42.6mm.
- ניגודיות כוונון : באמצעות פוטנציומטר.
- השתמש רק בשני ממשק קלט / פלט.

יתרונות ה I2c

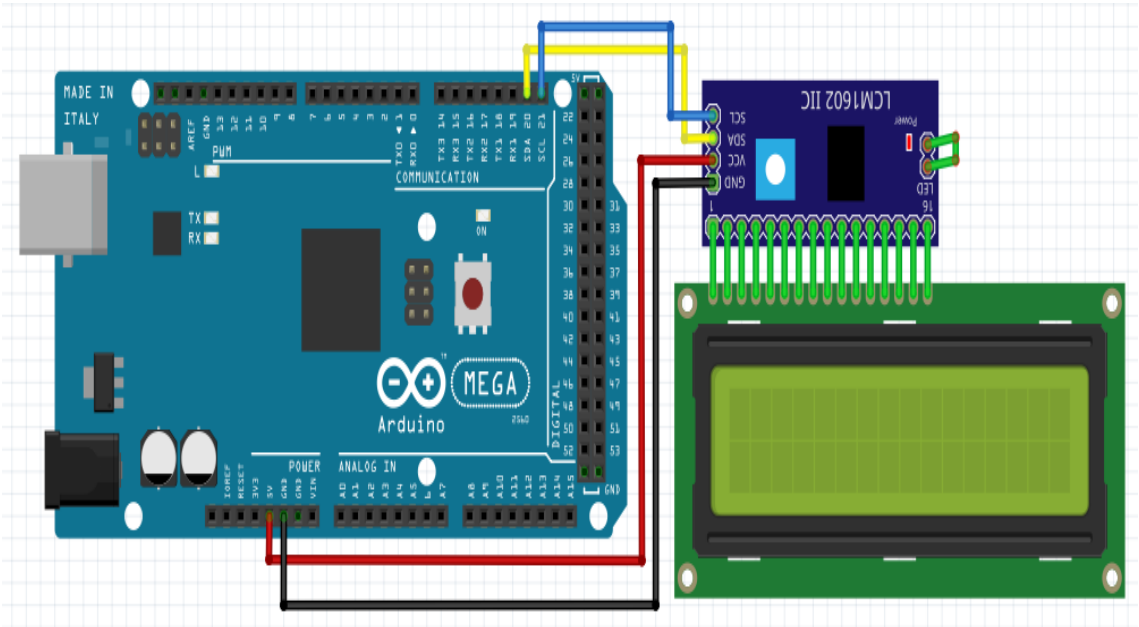
- מקטין את מספר החיבורים המינימאלי הנדרש מ -12 (עבור **LCD** עם תאורה אחורית) רק 3 או 4 ובכך לחסוך רבים קלט / פלט סיכות על המיקרו בקר .
- שליטה על **LCD** באמצעות כמעט כל, מיקרו או דרך ממשק **I2C** או באמצעות סדרתי חיבור **RX** (בחומרה או בתוכנה).
- מפשט את החיווט.
- שליטה דיגיטלית של בהירות התאורה האחורית של כבוי מ (0) עד מלא על (250) וכל רמה בין לבין.

תרשים חשמלי ל LCD



שיטת חיבור

I2C LCD1602	Arduino Mega
GND	GND
VCC	5V
SDA	20
SCL	21



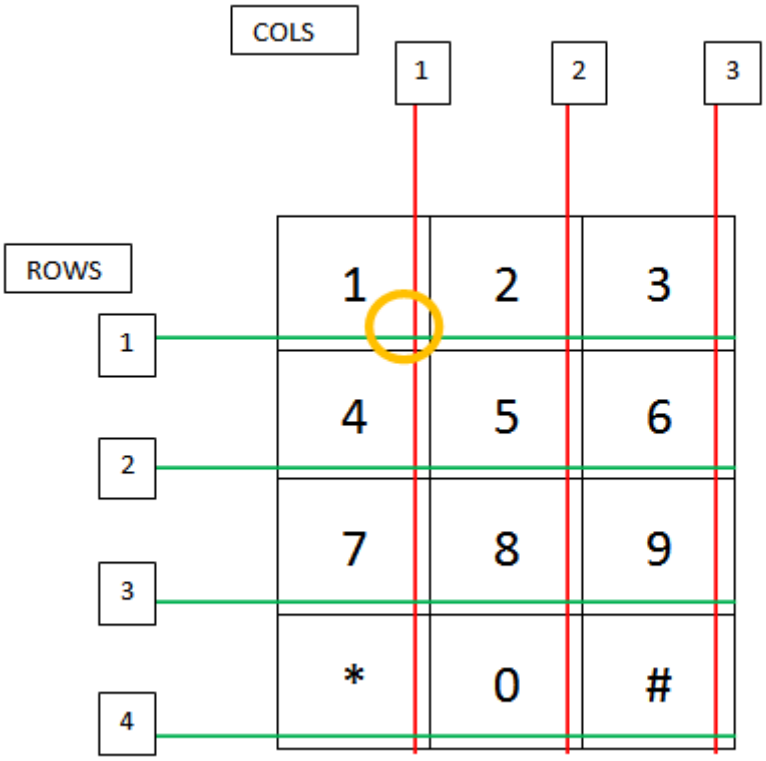
פונקציות ה LCD

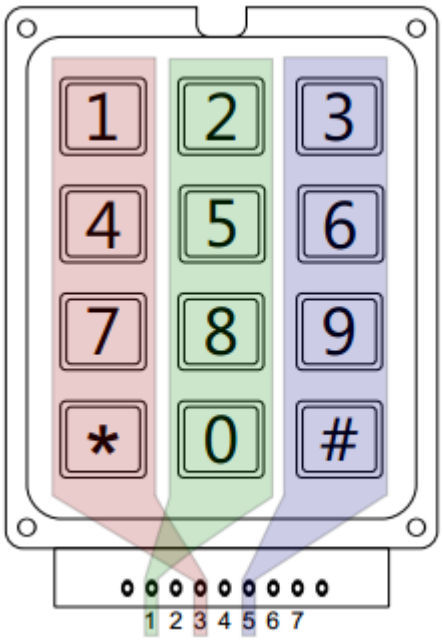
```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C  lcd(0x27,  16,  2);

//////////////////////////////////////

init();

clear();

print("");

home();

setCursor();

Begin();

Write();

Cursor();

noCursor();

blink();

noBlink();

display();

scrollDisplayLeft();

scrollDisplayRight();

autoscroll();

noAutoscroll();

leftToRight();

rightToLeft();

createChar();
```

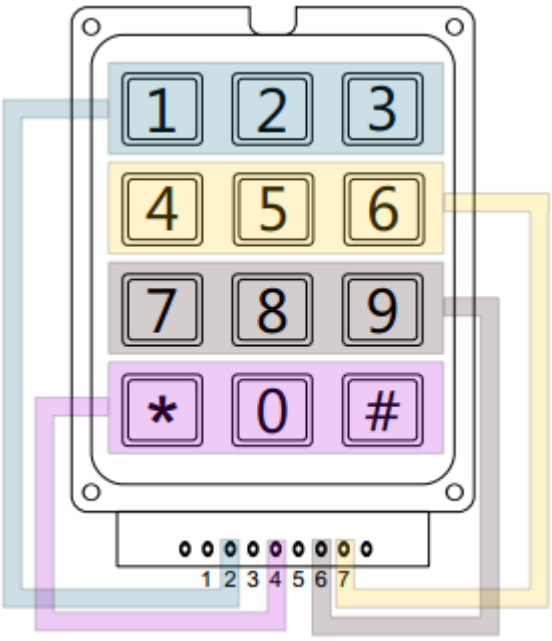
לוח מקשים Keypad

לוח מקשים נדרש לעתים קרובות כדי לספק קלט למערכת ***Arduino***, אנו מדגימים כיצד להשתמש בלוח מקשים נומרי בעל 12 לחצנים, בדומה למה שאתה עשוי למצוא בטלפון. לוח מקשים בעל 12 לחצנים כולל שלוש עמודות וארבע שורות. לחיצה על כפתור תהיה קצרה אחת מפלט השורות לאחת מפלט העמודות. ממידע זה, ***Arduino*** יכול לקבוע איזה כפתור היה לחוץ. לדוגמה, כאשר מקש 1 נלחץ, עמודה 1 ושורה 1 קצרים. ***Arduino*** יזהה את זה קלט 1 לתוכנית.

כיצד שורות ועמוד מסודרים בתוך לוח המקשים, מוצג באיור להלן.







קוד קבוע של לוח מקשים

```
#include<keypad.h>

Const byte  ROWS = 4;

Const byte COLS = 3;

Char keys [ROWS] [COLS] = {

{'1','2','3' },

{'4','5','6' },

{'7','8','9' },

{'*' ,'0','#' }

};

Byte rowPins [ROWS] = {5,4,3,2};

Byte colPins [COLS] = {8,7,6};

Keypad   keypad  = keypad ( makeKeymap(keys), rowPins, colPins ,ROWS , COLS);
```

מנוע סרוו Servo 9g

זעיר וקל משקל בעל הספק גבוה. סרוו יכול לסובב כ 180 מעלות (90 בכל כיוון), ועובד בדיוק כמו סוגים סטנדרטיים אבל קטן יותר. ניתן להשתמש בכל קוד סרוו, חומרה או ספריה לשלוט אלה סרוו. טוב למתחילים שרוצים לעשות דברים לזוז מבלי לבנות בקר המנוע עם המשוב ו תיבת הילוכים, במיוחד מאז זה יתאים במקומות קטנים. זה מגיע עם 3 קרניים (ידיים) וחומרה.

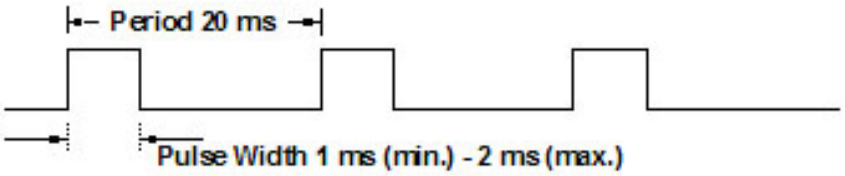
למנועי סרוו יש שלושה חוטים : כוח, אדמה, ואת האות. חוט החשמל הוא בדרך כלל אדום, והוא צריך להיות מחובר הסיכה V5 על לוח הארדואינו. חוט הקרקע הוא בדרך כלל שחור או חום ויש לחבר אותו על סיכת הקרקע על לוח הארדואינו. סיכה האות הוא בדרך כלל צהוב, כתום או לבן צריך להיות מחובר סיכה דיגיטלית על לוח הארדואינו . שים לב סירוו לצייר כוח רב, אז אם אתה צריך לנהוג יותר מאשר אחד או שניים, אתה כנראה צריך כוח מהם אספקה נפרדת (כלומר לא סיכה V5 + על הארדואינו שלך). הקפד לחבר את השטח של הארדואינו ואת אספקת החשמל החיצונית יחד.

מפרטים הסרוו

- משקל : g9.
- מדה : 22.2 * 11.8 * 31 mm .
- מהירות הפעלה : 0.1 60 / s מעלות.
- מתח הפעלה : (~5V) 4.8 V.
- רוחב פס מת : 10 µs .
- טווח טמפרטורות : 0 ºC – 55 ºC.

איך הסרוו עובד

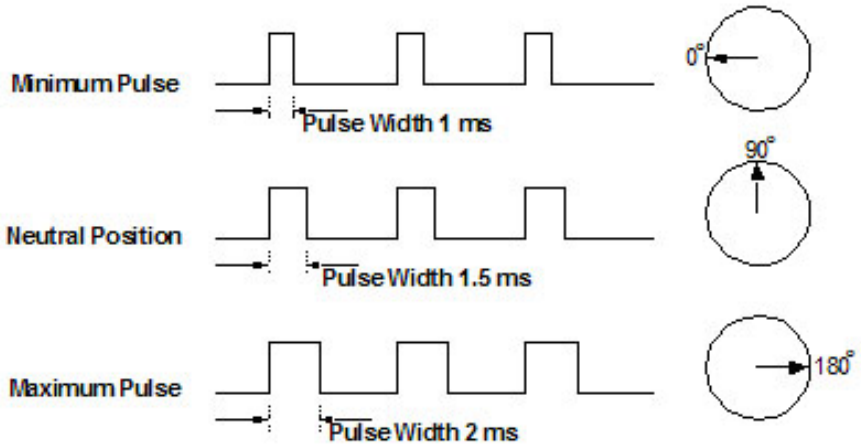
מנועי הסרוו נשלטים על ידי שליחת אותם הדופק של רוחב משתנה. חוט הבקרה משמש לשליחת הדופק. הפרמטרים עבור הדופק הזה הם שיש לו דופק מינימלי, דופק מקסימלי, ושיעור החזרה. בהתחשב באילוצי הסיבוב של הסרוו, נייטרלי מוגדר להיות המיקום שבו סרוו יש בדיוק את אותה כמות של סיבוב פוטנציאלי בכיוון השעון כפי שהוא עושה בכיוון השעון נגד. חשוב לציין כי מנועי הסרוו שונים יהיו אילוצים שונים על הסיבוב שלהם אבל לכולם יש עמדה נייטרלית, וכי המיקום הוא תמיד סביב 1.5 אלפיות השנייה (אלפיות השנייה).



הזווית נקבעת על ידי משך הדופק המוחל על חוט הבקרה. זה נקרא אפנון רוחב דופק. סרוו מצפה לראות דופק כל 20 אלפיות השנייה. אורך הדופק יקבע עד כמה המנוע יסתובב. לדוגמה, דופק 1.5 ms יעשה את הפנייה המנועית למצב 90 מעלות (מיקום נייטרלי).

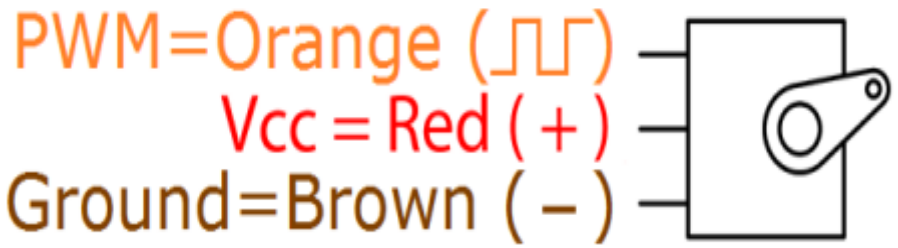
כאשר מנועי הסרוו אלה מצווים לנוע הם יעברו למצב והחזק את המיקום. אם כוח חיצוני דוחף כנגד סרוו בעוד סרוו הוא מחזיק עמדה, סרוו יתנגדו לצאת מהמקום הזה. כמות מקסימלית של כוח סרוו יכול להפעיל את דירוג המומנט של סרוו. מנועי הסרוו לא מחזיקים את עמדתם לנצח; יש לחזור על הדופק כדי להורות למכשיר להישאר במצב.

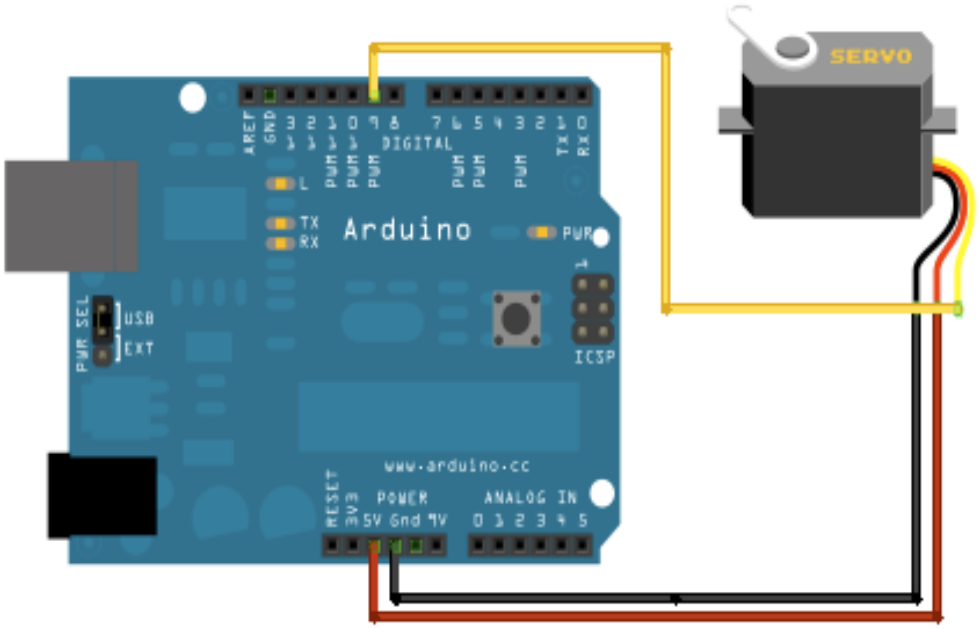
כאשר הדופק נשלח לסרוו כי הוא פחות מ -1.1 מילישניות סרוו מסתובבת למיקום ומחזיק פיר המוצא שלה כמה מעלות מעלות נגד כיוון השעון מנקודת נייטרלי.
כאשר הדופק הוא רחב יותר מ 1.5 ms ההיפך מתרחשת. רוחב מינימלי רוחב מרבי של הדופק כי הפקודה סרוו לפנות עמדה חוקית הן פונקציות של כל סרוו. מותגים שונים, ואפילו סרוו שונים של המותג אותו, יהיה מקסימום מינימום שונים. בדרך כלל הדופק המינימלי יהיה בערך 1 ms רחב ואת הדופק המרבי יהיה 2 ms רחב.



שיטת חיבור המנוע

Servo 9g	Arduino
(כתום) PWM	PWM pins
אדום	v5
חום/שחור	GND



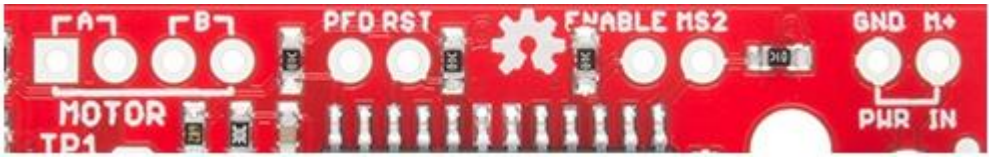


רכיב Easy driver

Easy driver מתוכנן סביב ***A3967 IC*** . ***IC*** זה מאפשר לך לנהוג מנועים ***stepper*** דו קוטבית כי הם 4, 6, או 8-תיל תצורות. הלוח יכול לעבוד עם מערכות ***V3.3*** או ***V5*** , מה שהופך אותו רב תכליתי. שני חורים הרכבה על הלוח לתת למשתמש את האפשרות לייצב מכנית את ***Easy driver*** .

החלק העליון של הלוח

אם אתה מסתכל על החלק העליון של הלוח, תראה מספר סיכות.



הם פועלים כדלקמן :

- **סליל +A**, H-Bridge 2 Output A, חצי נקודת חיבור עבור דו קוטבית, מנוע stepper, סליל A.
- **סליל -A** , H-Bridge 2 Output B , חצי נקודת חיבור עבור דו קוטבית, מנוע stepper, סליל A.
- **סליל +B** , H-Bridge 1 Output A , , חצי נקודת חיבור עבור דו קוטבית, מנוע stepper, סליל B.
- **סליל -B** , H-Bridge 1 Output B , , חצי נקודת חיבור עבור דו קוטבית, מנוע stepper, סליל B.
- **PFD** , מתח, קלט הבוחר במצב ריקבון זרם הפלט.
- **RST** , קלט לוגי. בעת הגדרת LOW, כל הפקודות STEP מתעלמות וכל פונקציונליות FET כבויה. יש למשוך את HIGH כדי לאפשר בקרת STEP .

- **ENABLE**_קלט לוגי. הפעלת פונקציונליות FET בתוך מנהל ההתקן. אם הוגדר ל- HIGH, ה- FETs יושבתו, וה- IC לא יניע את המנוע. אם מוגדר ל LOW, כל FETs יופעלו, המאפשרים שליטה במנוע.
- **MS2** קלט לוגי. ראה טבלת האמת להלן עבור פונקציונליות גבוהה / נמוכה.
- **GND , אדמה** .
- **M+** , ספק כוח. V6-30, אספקת A2.

<u>החלק התחתון של הלוח:</u>

יש גם סיכות על החלק התחתון של הלוח.

<u>הם פועלים כדלקמן :</u>

- **GND** , אדמה.
- **V5** , פלט . סיכה זו יכולה לשמש כוח מעגל חיצוני. נדרש מקסימום של mA 70 עבור פונקציונליות של 'מנהל התקן קל'.
- **SLP** , קלט לוגי. כאשר משך LOW, יציאות מושבתות וצריכת החשמל ממוזער.
- **MS1** , קלט לוגי. ראה טבלת האמת להלן עבור פונקציונליות גבוהה / נמוכה.
- **GND** , אדמה.
- **STEP** , קלט לוגי. כל מעבר על סיכה זו מ LOW ל HIGH יפעילו את המנוע צעד אחד קדימה. כיוון וגודל של צעד נשלט על ידי הגדרות DIR ו- MSX PIN. זה יהיה גם V0-5 או V0-3.3, על פי בחירת ההיגיון.

- **DIR** , קלט לוגי. סיכה זו קובעת את כיוון סיבוב המנוע. שינויים במצב מ HIGH ל LOW או LOW ל HIGH רק ייכנסו לתוקף על הקצה העולה הבא של הפקודה STEP. זה יהיה גם V0-5 או V0-3.3, על פי בחירת ההיגיון.

מגשרים הלחמה

ישנם שני jumpers הלחמה על הלוח. אלה מספקים למשתמש את התכונות הבאות:

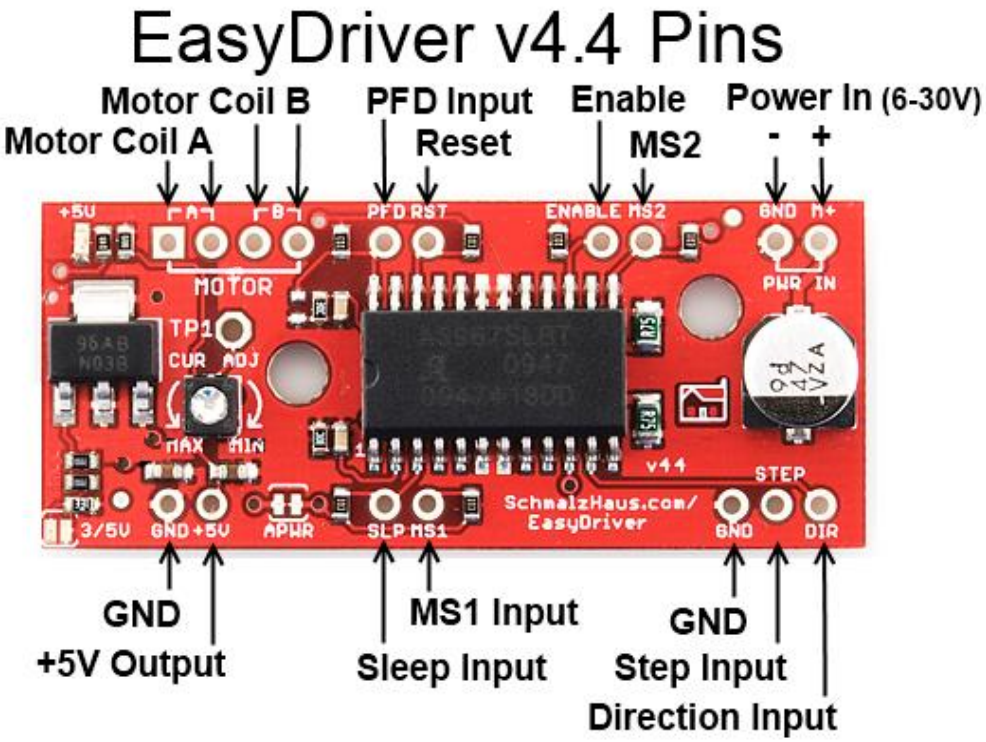
- **V5 / 3** - מגשר זה מאפשר למשתמש להגדיר את התצורה של VCC בין V3.3 או V5. עם מגשר פתוח, VCC יהיה V5. אם מגשר סגור, VCC הוא V3.3.

- **APWR** - מגשר זה מאפשר למשתמש מקור VCC על GND5 / V סיכות לחומרה חיצונית.

פוטנציומטר

הפוטנציומטר על הלוח נכלל כדי לאפשר למשתמשים את היכולת לבחור את הזרם המקסימאלי שסופק המנוע. זה נע בין mA150 ל mA750.

איור הרכיב


EasyDriver v4.4 Pins
Motor Coil B
Motor Coil A
PFD Input
Reset
Enable
MS2
Power In (6-30V)
-
+
GND
+5V Output
MS1 Input
Sleep Input
GND
Step Input
Direction Input

מנוע Stepper

מנוע ***Stepper*** הוא מנוע חשמלי סינכרוני חסר מברשות הממיר פולסים דיגיטליים לסיבובי פיר מכניים. כל סיבוב של מנוע ***stepper*** מחולק למספר מוגדר של צעדים, לפעמים עד 200 צעדים. מנוע צעד צריך להישלח הדופק נפרד עבור כל צעד. מנוע צעד יכול לקבל רק דופק אחד לקחת צעד אחד בכל פעם וכל צעד חייב להיות באותו אורך. מאז כל הדופק תוצאות המנוע מסתובב זווית מדויקת - בדרך כלל 1.8 מעלות - אתה יכול לשלוט במדויק את המיקום של מנוע ***stepper*** ללא מנגנון משוב.

כמו הפולסים הדיגיטליים מ להגדיל את הבקר בתדירות, תנועת דריכה ממיר לתוך סיבוב רציף עם מהירות הסיבוב ישירות ביחס לתדירות של פעימות בקרה. מנועי ***Stepper*** נמצאים בשימוש נרחב בגלל העלות הנמוכה שלהם, אמינות גבוהה, מומנט גבוה במהירויות נמוכות. הבניה הקשוחה שלהם מאפשרת לכם להשתמש במנועי ***Stepper*** במגוון רחב של נושאים סביבתיים.

יתרונות Stepper motor

- מגוון רחב של מהירויות סיבוב יכול להיות מנוצל מאז המהירות של מנוע צעד הוא יחסי לתדירות של קלט פולסים מבקר שלך.
- ניתן לבצע בקרה מדויקת של תנוחת לולאה פתוחה באמצעות מנוע צעד, ללא מנגנון משוב.
- סיבוב מהירות נמוך מאוד אפשרי עם עומס כי הוא מצמידים ישירות את הפיר של מנוע צעד.
- מנוע stepper הוא אמין למדי, כי אין מברשות מגע. בדרך כלל, את החיים של מנוע stepper נקבע על ידי החיים של מנוע stepper הנושא.
- מנוע stepper הוא טוב מאוד בהתחלה, עצירה, היפוך כיוון.
- מנוע stepper מספק מיקום מדויק של הדירות.
- מנוע Stepper אנרגטי שומר מומנט מלא בתנוחת קיפאון.

סוגי Stepper motor

ישנם שלושה סוגים של מנועים צעד : מגנט קבוע, היברידית, ורתיעה משתנה. מנועי צעד היברידי מציעים את הרבגוניות ביותר לשלב את המאפיינים הטובים ביותר של אי רצון משתנה ומנועים קבועים מגנט stepper. מנועים היברידיים היברידיים בנויים עם מוטות רב מוטות סטטור רוטור מגנט קבוע. מנוע סטנדרטי היברידי סטפר יש 200 שיניים הרוטור מסתובב 1.8 מעלות לכל צעד. מנועי היברידי היברידיים מספקים מומנט סטטי ודינמי גבוה והם פועלים בקצב גבוה מאוד. יישומים עבור מנועי stepper היברידי לכלול כונני דיסק במחשב ושחקנים תקליטורים. מנועי היברידי היברידי הם גם בשימוש נרחב ביישומים תעשייתיים ומדעיים. מנועים צעד היברידי משמשים רובוטיקה, בקרת תנועה, חיתוך חוט אוטומטי, ואפילו במהירות גבוהה נוזלי מכשירי.

צעד מלא Full step

מנועים סטנדרטיים היברידית דריכה יש 200 צעדים מלאים לכל מהפכה. אם מחלקים את 200 המדרגות לתוך 360 מעלות של סיבוב אתה מקבל 200 1.8 מעלות צעד. בדרך כלל זה מושגת על ידי ממריץ שתי פיתולים תוך הפיכת לסירוגין הנוכחי, כלומר אחד הדופק מהנהג שווה צעד אחד מלא על המדרגה המנוע.

חצי צעד Half step

חצי צעד אומר כי מנוע דריכה הוא מסתובב ב 400 צעדים לכל מהפכה (0.9 מעלות מעלות 360 = 400 x מעלות). תחילה מתפתל אחד מתפתל ולאחר מכן שני פיתולים לחילופין אנרגטי. זה יגרום הרוטור של מנוע דריכה לנוע בחצי המרחק (0.9 מעלות). במצב חצי צעד, מנוע סטפר אופייני מספק כ -30% פחות מומנט, אבל הוא מספק תנועה חלקה יותר מאשר במצב מלא.

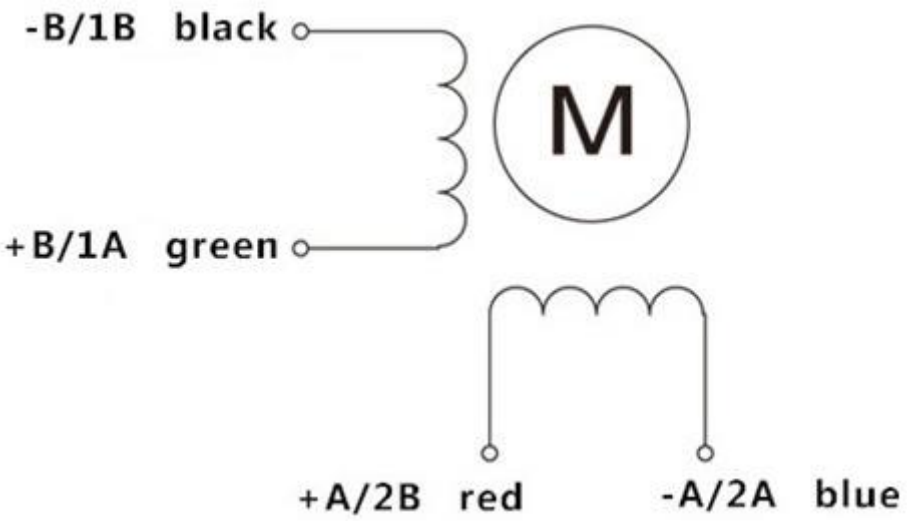
Stepper Motor Drivers

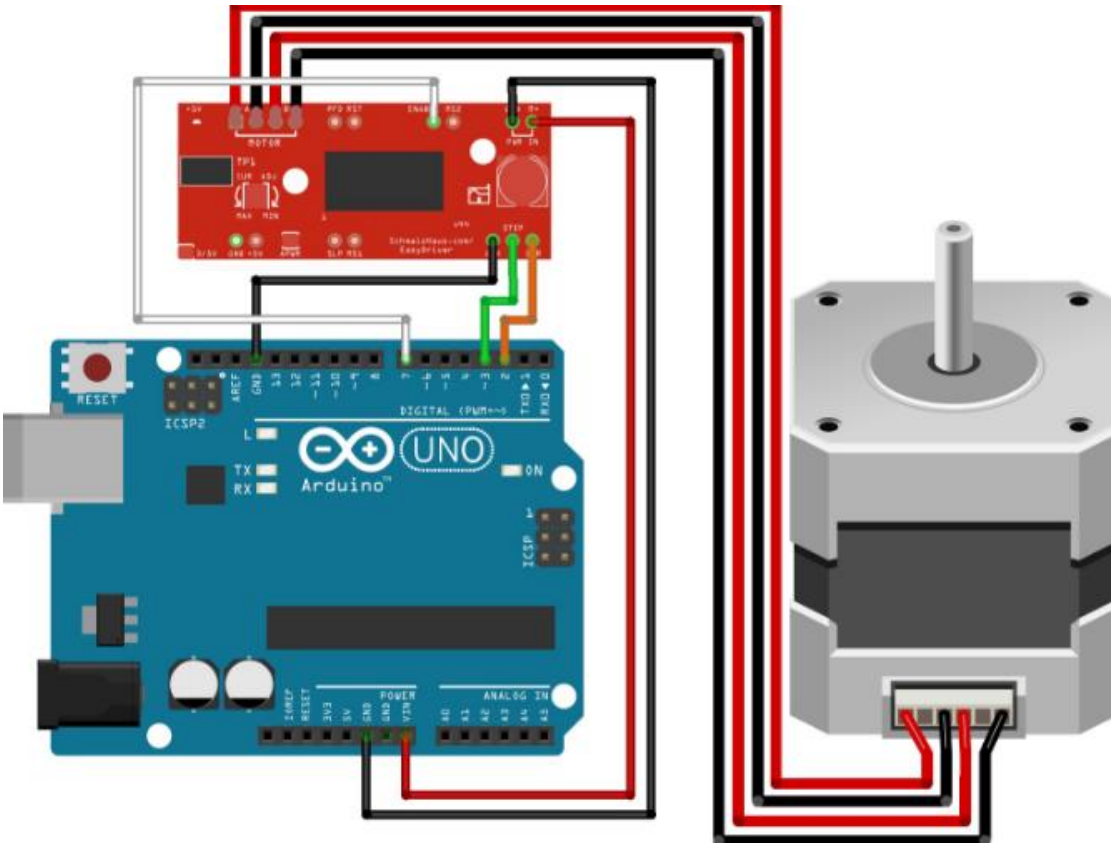
מנוע ***stepper*** נשלט על ידי לוח ***motor driver*** . ***motor driver*** מקבל אותות צעד והכוונה ממערכת בקרה, בדרך כלל מחשב, וממיר אותם אותות אלקטרוניים אשר מפעיל את המנוע צעד. דופק אחד נדרש עבור כל צעד של מוט המנוע ***stepper.*** במצב מלא, בהנחה שאתה משתמש במנוע סטנדרטי 200 צעד, 200 צעדים או פעימות משלים מהפכה אחת של מוט מנוע צעד. המהירות והסיבוב של מוט המנוע ***stepper*** הוא ביחס ישר לתדירות הדופק.

המהירות ואת מומנט של מנוע ***stepper*** נקבעת על ידי זרימת הנוכחי מן ***motor driver*** דורך על מנוע דריכה מתפתל. השראות מפחית את הזרימה או מגביל את הזמן הדרוש כדי הנוכחי כדי להמריץ את סלילה. רוב מנועי מנוע ***stepper*** המנוע נועדו לספק כמות גדולה יותר של מתח מאשר מתח מנוע מדורגת של צעד. ככל שהמתח המוצא גבוה יותר ***motor driver*** הסופר, כך גבוה יותר רמת המומנט לעומת המהירות. באופן כללי, מתח הפלט של מנוע הסוכר, הידוע גם כמתח האוטובוסים, צריך להיות מדורג פי חמש עד פי 10 ממדירוג המתח של מנוע הסופר. על מנת להגן על מנוע דריכה, הנוכחי של בקר המנוע הנוכחי צריך להיות מוגבל לדירוג המנוע הנוכחי צעד.

29

שיטת חיבור (עם Motor driver)





יומן עבודה

מפגש 1

המורה הסביר לנו על הרובוטיקה ועל העבודה במעבדה בדרך כלל.

מפגש 2

התחלקנו לקבוצות , והתחלנו לעסות ניסוים פשוטים .

כמו :

- הבהוב LED .
- חיבור מפסקים.

מפגש 3

התחלנו ללמוד על רכיבים אנו עשויים להשתמש בו בפרויקטים.

כמו :

- LCD
- Keypad
- Servo
- Stepper motor

מפגש 4

קיבלנו הצעות פרויקטים.

מפגש 5

קיבלנו את הרכיבים עם ערכת ארדואינו.

מפגש 6

התחלנו לעבוד בפרויקט .

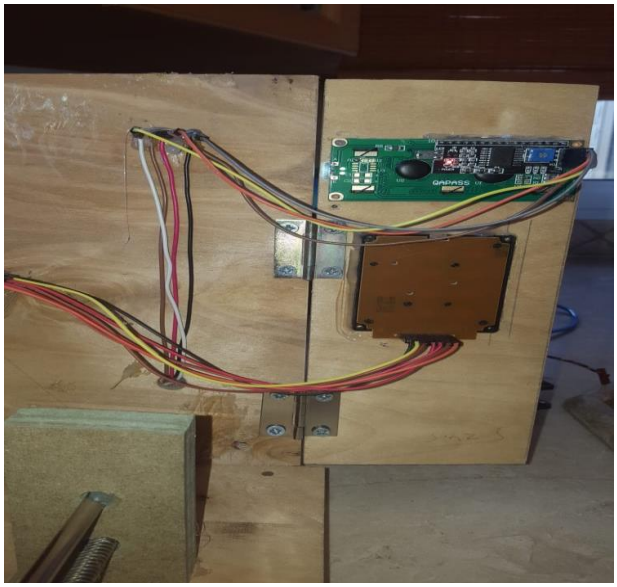
בנינו סטריאוסקופית הפרויקט .

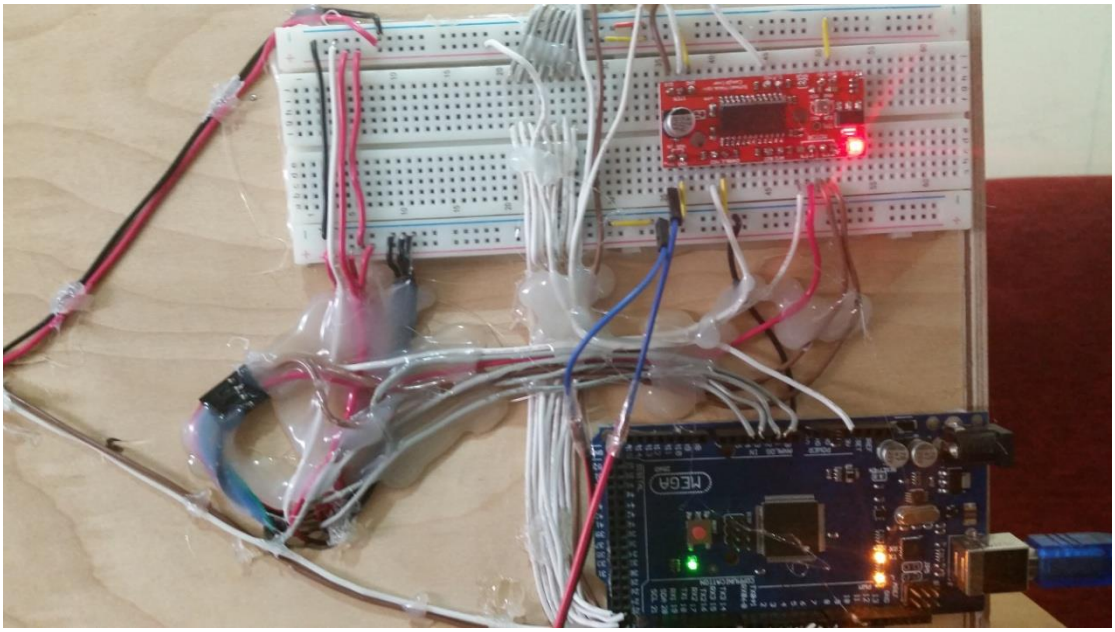
מפגש 7

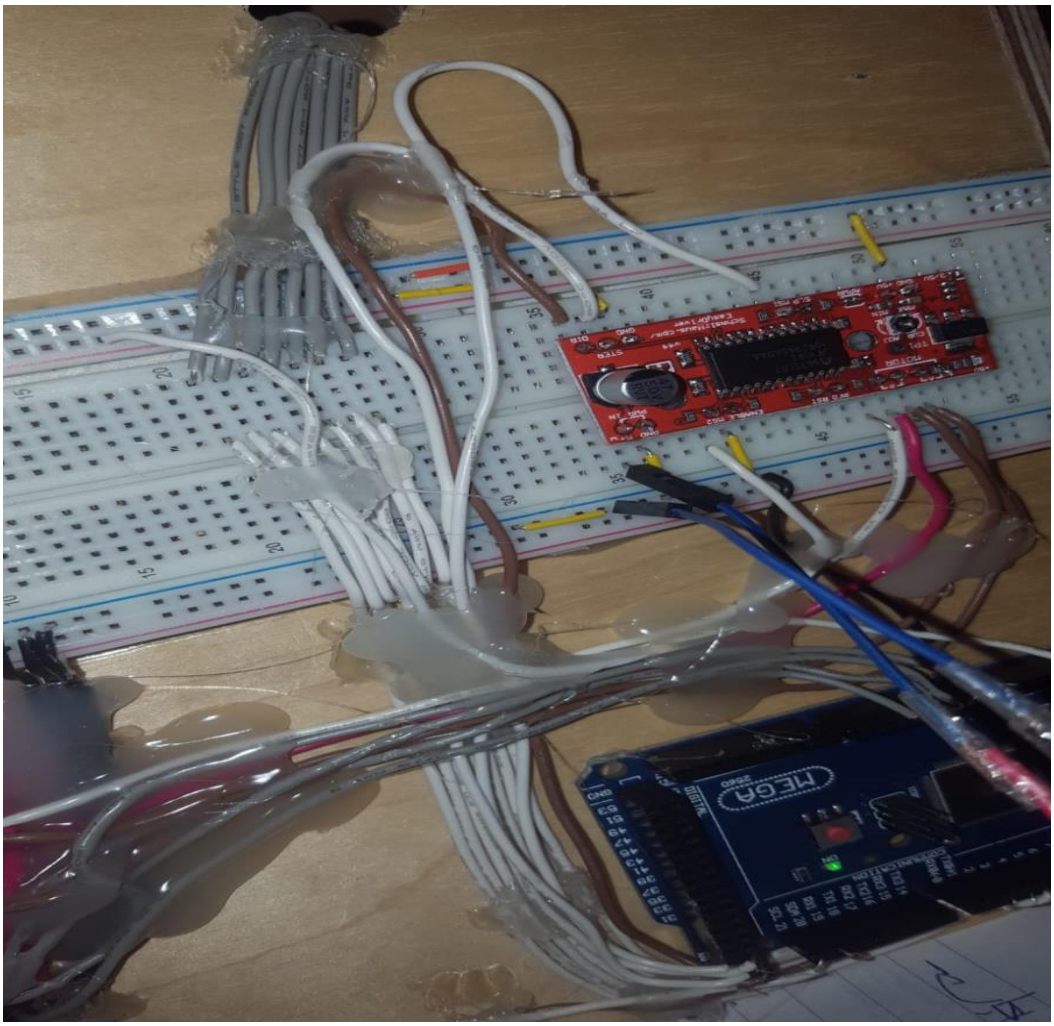
חיברנו את כל הרכיבים לארד ואינו ועשינו סימולציה לשיטת העבודה לפרויקט .

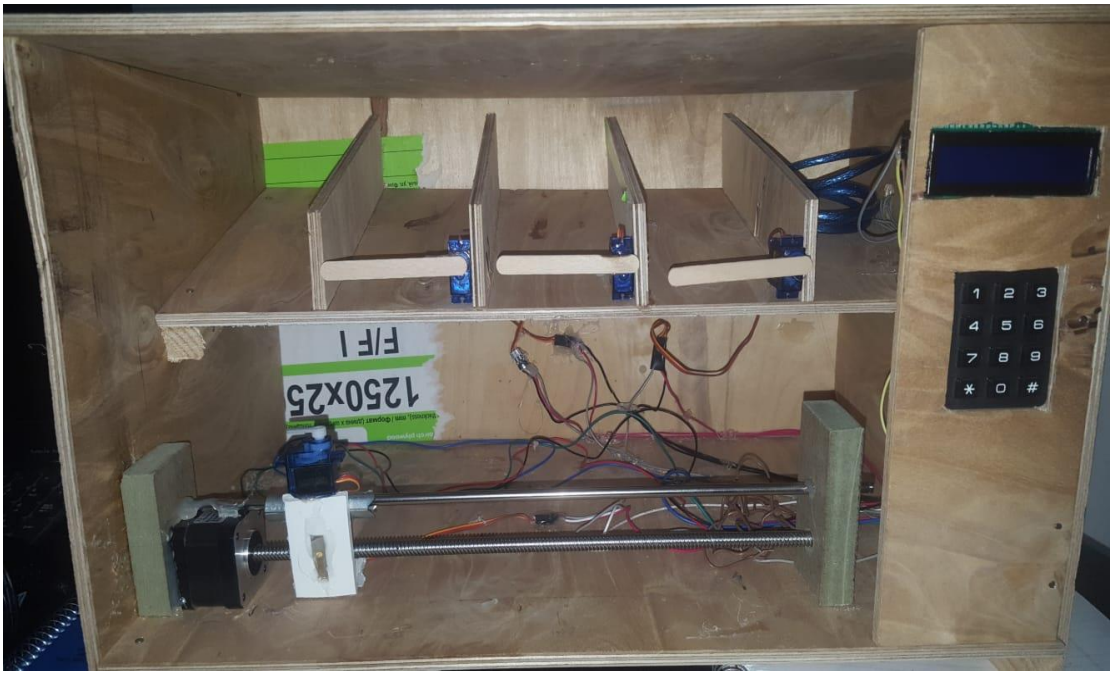
מפגש 8

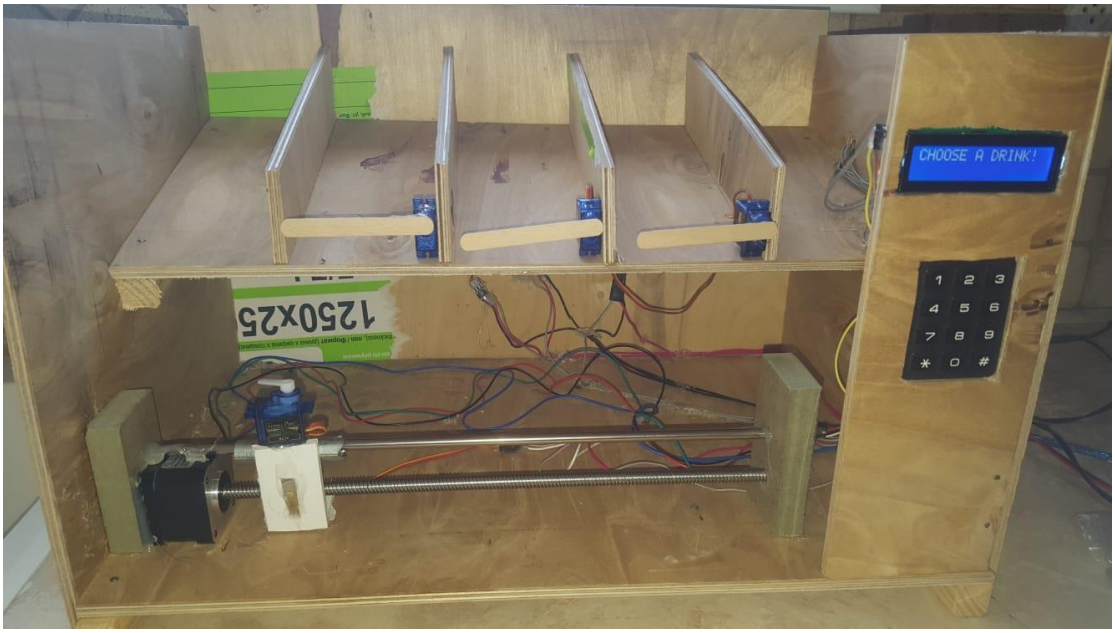
שמנו את הרכיבים המחוברים על סטריאוסקופית הפרויקט .








1250x25
F/F 1
1 2 3
4 5 6
7 8 9
* 0 #


1250x25
CHOOSE A DRINK!
1 2 3
4 5 6
7 8 9
* 0 #

העלאת את הקוד לארד ואינו .

```
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <Wire.h>
#include <Servo.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

const byte ROWS = 4;
const byte COLS = 3;
char hexaKeys[ROWS][COLS] = {
{'1','2','3'},
{'4','5','6'},
{'7','8','9'},
};
byte rowPins[ROWS] = {5, 4, 3, 2};
byte colPins[COLS] = {8, 7, 6};
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins,
colPins, ROWS, COLS);

Servo myservo1;
Servo myservo2;
Servo myservo3;
Servo myservoR;

int pos = 0;
```

```
char customKey;
#define step_pin 23   // Define pin 3 as the steps pin
#define dir_pin 22   // Define pin 2 as the direction pin
#define MS1 25    // Define pin 5 as "MS1"
#define MS2 24     // Define pin 4 as "MS2"

#define MOTOR_SPEED 1500
#define MOTOR_ROTATE_LOOP 500
Int  lookup [8] = { B01000, B01100, B00110, B00110, B00010, B00011,
B00001, B01001};
void setup()
{
 lcd.init();
 lcd.backlight();
 lcd.setCursor(0, 0);
 lcd.print("CHOOSE A DRINK");
 myservo1.attach(A1);
 myservo1.write(90);
 myservo2.attach(A2);
 myservo2.write(90);
 myservo3.attach(A3);
 myservo3.write(90);
 myservoR.attach(A4);
 myservoR.write(0);
 Serial.begin(9600);
 pinMode(MS1, OUTPUT);    // Configures "MS1" as output
 pinMode(MS2, OUTPUT);    // Configures "MS2" as output
```

```
  pinMode(dir_pin, OUTPUT);    // Configures "dir_pin" as output
  pinMode(step_pin, OUTPUT);    // Configures "step_pin" as output
  digitalWrite(MS1, LOW);      // Configures the steps division (see above(
  digitalWrite(MS2, LOW);    // Configures the steps division (see above(
  digitalWrite(dir_pin, LOW);   // Sense (HIGH = anti-clockwise / LOW =
clockwise) - It can be also changed
}

void loop()
{
  lcd.setCursor(0, 0);
  lcd.print("CHOOSE A DRINK");
  customKey = customKeypad.getKey();
  Serial.println(customKey);
  if (customKey == '1')
  {
    lcd.setCursor(0, 0);
    lcd.print("YOU'VE CHOSED  ");
    lcd.setCursor(1, 5);
    lcd.print("COLA");
    Serial.println(customKey);
    moveRight(1100);
    myservo1.write(10);
    delay(3000);
    myservo1.write(90);
    delay(1000);
```

```
    moveLeft(1100);

    swepClock();
    delay(3000);

    swepAntiClock() ;
    lcd.clear();

  }

   if (customKey == '2')
   {
    lcd.setCursor(0, 0);
    lcd.print("YOU'VE CHOSED"  );
    lcd.setCursor(1, 4);
    lcd.print("ORANGE");
    Serial.println(customKey);

    moveRight(3200);
    myservo2.write(10);

    delay(3000);
    myservo2.write(90);
    delay(1000);

    moveLeft(3200);
```

```
    swepClock();
    delay(3000);

    swepAntiClock();

    lcd.clear();
  }

  if (customKey == '3')
   {
    lcd.setCursor(0, 0);
    lcd.print("YOU'VE CHOSED");
    lcd.setCursor(1, 4);
    lcd.print("FANTA");
    Serial.println(customKey);

    moveRight(5500);

    myservo3.write(10);
    delay(3000);
     myservo3.write(90);
    delay(1000);

    moveLeft(5500);

    swepClock();
    delay(3000);
```

```
    swepAntiClock();

    lcd.clear();

   }

   customKey = '5';



 }

//////////////////////////////////////////////////
void moveRight(int  loopsNum)
{
  digitalWrite(dir_pin, HIGH);  // move in the LOW direction
  startmotor(loopsNum);
}
//////////////////////////////////////////////////////////
void moveLeft(int loopsNum)
{
  digitalWrite(dir_pin, LOW);  // move in the LOW direction
  startmotor(loopsNum);
}

void startmotor(int loopsNum)
{
```

```
 for (int j = 0 ; j < loopsNum;  ++j)
{
   digitalWrite(step_pin, LOW);
   delay(1);
   digitalWrite(step_pin, HIGH);
   delay(1);

  }
}
//////////////////////////////////////////////////////
void swepClock ()
{
 for (pos = 0;  pos <= 90;  pos += 1) // goes from 0 degrees to 180 degrees
{
   myservoR.write(pos);          // tell servo to go to position in variable 'pos'
   delay(30);                    // waits 15ms for the servo to reach the position
}
}

void swepAntiClock ()
 for (pos = 90;  pos >= 0;  pos -= 1) // goes from 0 degrees to 180 degrees
{
   myservoR.write(pos);            // tell servo to go to position in variable
'pos'
   delay(30);                    // waits 15ms for the servo to reach the position
  }
}
```

תרשים חשמלי