פיתוח פרויקטם בסביבת Arduino

פיתוח פרויקט הוא תהליך בו התלמיד צריך ללמוד:

- הכרת הממשק של התוכנה.
- שפת תכנות.
- הכרה החומרה של הפרויקט.
- אופן החיבורים של החומרה למיקרו-בקר.
- כתיבת תוכנה להפעלת חומרה של כל רכיב בפרויקט.
- איתור תקלות בתוכנה באמצעות חיווי על מסך התקשורת הטורית או באמצעות נורות LED וכו.
- איתור תקלות בחומרה בדיקת מתחים וזרמים של הרכיבים, בדיקת חיווט הפעלת מיכשור מעבדה כגון : רב מודד, סקופ, logic Analazer.
- לבסוף כתיבת קוד תוכנית להפעלת כל הפרויקט.

תלמיד שיעבור את התהליך ידע להסביר את קוד התוכנית, החומרה ולספר על התהליך שעבר.

לדוגמה פרויקט לבקרת טמפרטורה הכולל:

- מסך גרפי העובד בפרוטוקול טורי.
- חיישן טמפרטורה אנלוגי או חיישן העובד בפרוטוקול טורי.
- מאוורר.
- רמקול .
- התקן Bluetooth העובד בפרוטוקול טורי UART.

מסך גרפי טורי

כתיבת קוד להפעלת מסך גרפי דורשת ניסיון רב בכתיבת קוד להפעלת החומרה. דפי הנתונים של המסכים יכול להגיע למאות עמודים. האתחול של המסך כולל מספר שורות קוד ולכן סביר להניח שהתלמיד ישתמש בספרייה מוכנה. שזה כל היופי ב- Arduino אחרת הוא לא היה יכול לחבר את המסך.

אך על התלמיד לדעת:

- האם המסך עובד בצורה מקבלית או באמצעות פרוטוקול טורי?
- במידה והמסך עובד בפרוטוקול טורי ניתן לשאול:
- האם הפרוטוקול טורי סינכרוני או אסינכרוני?
- במידה והפרוטוקול סינכרוני מה תדר השעון של הפרוטוקול? מי יוצר את השעון?
- מה מבנה הפרוטוקול:
  - ✓ כיצד יודעים שיש התחלת תשדורת? (Start Bit)?
  - ✓ כמה סיביות מידע שולחים או מקבלים בפרוטוקול ?
  - ✓ איזה סיבית מידע שולחים קודם ? MSB או LSB?
  - ✓ האם בודקים את המידע שהתקבל ? אם כן איזה בדיקה?
  - ✓ כיצד מסיימים את התקשורת?
- מה זה פיקסל? עומק הפיקסל (כמה סיביות צבע יש לכל פיקסל)?
- מה הרזולוציה של המסך? (מספר שורות מספר עמודות)
- פירוט הדקים של המסך?
- לבקש מהתלמיד להראות בסכימה החשמלית כיצה המסך מחובר למיקרו-בקר ?
- בשימוש עם ספרייה מוכנה התלמיד צריך להכיר את הפונקציות שהוא השתמש בהן ניתן לשאול:
  - ✓ איזה ספרייה צריך להצהיר על מנת להתשמש עם המסך?
  - ✓ כיצד מבצעים אתחול למסך?
  - ✓ ניתן לבקש מהתלמיד להראות כיצד הוא כותב את ערך הטמפרטורה על המסך?
  - ✓ ניתן לבקש מהתלמיד להראות כיצד הוא משנה את צבע הפונט וגודל הספרות על המסך?
  - ✓ כיצד מוחקים את המסך?
  - ✓ אם במידה והתלמיד מציג גרף של הטמפרטורה שיסביר את סדר הפעולות שהוא ביצע כדי להציג את הגרף.

חיישן טמפרטורה אנלוגי

- לספר איזה חיישן חיבר?
- האם היה שיקול מסויים בבחירת החיישן?
- מה רמת הדיוק של חיישן הטמפרטורה?
- התלמיד שצריך לדעת להסביר מה תפקיד רכיב ADC?
- מה הרזולוציה של הרכיב?
- לדעת להסביר על הדקי הרכיב?

- מה מתחי עבודה של הרכיב?
- לדעת להראות בסכימה החשמלית לאן הרכיב מחובר?
- אם במידה וחיבר Vref לדעת לאן הוא מחובר ומה חשיבות הדבר?
- באיזה פונקציה השתמש על מנת לקרוא את ערך הטמפרטורה?
- איזה המרה הוא ביצע על מנת להציג את המתח במעלות?
- שיראה בקוד התוכנית את השלבים מקריאת הטמפרטורה ועד להצגתה על גבי מסך הגרפי.

חיישן טמפרטורה דיגיטאלי (חיישן שמחובר בפרוטוקול טורי)

- האם הפרוטוקול טורי סינכרוני או אסינכרוני?
- במידה והפרוטוקול סינכרוני מה תדר השעון של הפרוטוקול? מי יוצר את השעון?
- מה מבנה הפרוטוקול:
    - כיצד יודעים שיש התחלת תשדורת? (Start Bit)?
    - כמה סיביות מידע שולחים או מקבלים בפרוטוקול ?
    - איזה סיבית מידע שולחים קודם ? MSB או LSB?
    - האם בודקים את המידע שהתקבל ? אם כן איזה בדיקה?
    - כיצד מסיימים את התקשורת?
- מה מתחי עבודה של הרכיב?
- לדעת להראות בסכימה החשמלית לאן הרכיב מחובר?
- להראות בדפי הנתונים של הרכיב איך קבע את כתובת הרכיב עם הרכיב עובד בפרוטוקול I^2C?
- אם הרכיב עובד בפרוטוקול טורי I^2C האם ניתן לחבר עוד חיישן טמפרטורה לאותם הדקים עם כתובת אחרת?
- מה רמת הדיוק של חיישן הטמפרטורה?
- שיראה בקוד התוכנית את השלבים מקריאת הטמפרטורה ועד להצגתה על גבי מסך הגרפי.

מאוורר

- מה מתחי עבודה של הרכיב?
- מה הזרם של המאוורר?
- איזה דוחף זרם הוא השתמש? מה השיקול בחיבור רכיב זה?
- לדעת להראות בסכימה החשמלית כיצד המאוורר והדוחף זרם מחוברים?
- האם ניתן לשנות את מהירות המאוורר? אם כן כיצד מבצעים זאת? (DAC או PWM) לדעת להסביר את השיטה בה הוא משתמש?
- מדוע לא חיבר את המאוורר ישירות להדקי ה- Arduino?
- שיראה בקוד התוכנית את השלבים מקריאת הטמפרטורה ועד להפעלת המאוורר.
- לבקש ממנו להראות היכן הוא צריך לשנות בקוד את ערך טמפרטורה הסף שממנה מפעילים את המאוורר.

רמקול

- אופן חיבור הרכיב
- כיצד משמעים צליל?
- איזה פונקציה משתמשים בשביל להפיק צלילים?
- לבקש ממנו להראות היכן הוא צריך לשנות בקוד התוכנית את הפרמטר שיפיק צליל אחר ברגע שהרמקול מופעל?

התקן Bluetooth העובד בפרוטוקול טורי UART

- מה זה Bluetooth?
- מה תדרי עבודה של הרכיב?
- כיצד מתחברים לרכיב באמצעות Smarphone?
- באיזה אפליקציה הוא השתמש?
- האם הוא כתב את האפליקציה הוא שהיא מוכנה והוא עשה לה התאמות לצרכים שלו?
- האם הפרוטוקול טורי סינכרוני או אסינכרוני?
- מה זה אומר פרוטוקול אסינכרוני?
- מה מבנה הפרוטוקול:
  - ✓ כיצד יודעים שיש התחלת תשדורת? (Start Bit)?
  - ✓ כמה סיביות מידע שולחים או מקבלים בפרוטוקול ?
  - ✓ איזה סיבית מידע שולחים קודם ? MSB או LSB?
  - ✓ האם בודקים את המידע שהתקבל ? אם כן איזה בדיקה?
  - ✓ כיצד מסיימים את התקשורת?
- שיראה בסכימה החשמלית את צורת החיבור של התקן?
- האם הוא חיבר ל- UART בתוכנה או בחומרה?
- שיראה בקוד התוכנית את השלבים מקבלת מידע מ- Bluetooth עד לקריאת המידע וביצוע פעולה מסויימת?

כמנחה פרויקטים ברצוני לשתף אותכם בכמה תובנות

- ❖ פרויקט הוא תהליך שמתחיל מתחילת הלימודים.
- ❖ פרויקט דורש מהתלמיד לעשות שילוב בין המקצעות הוא למד.
- ❖ פרויקט יכול לגרום לתלמיד להשקיע בלימודים.
- ❖ פרויקט דורש ממנחה המון שעות הכנה של חומרים ורעיונות.

Arduino הינה חברה איטלקית שהוקמה בשנת 2005 המפתחת סביבת פיתוח ולוחות שכוללים מיקרו-בקרים מסדרת AVR של חברת ATMEL.

להלן תמונות של לוחות:

Arduino Uno

Arduino Leonardo

Arduino Due

Arduino Yún

Arduino Tre

Arduino Micro

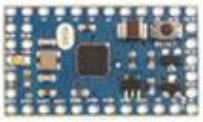
Arduino Robot

Arduino Esplora

Arduino Mega ADK

Arduino Ethernet

Arduino Mega 2560



Arduino Mini

Arduino Nano

Arduino Pro Mini

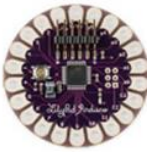
Arduino Pro

Arduino Fio

LilyPad Arduino USB

LilyPad Arduino Simple

LilyPad Arduino SimpleSnap



LilyPad Arduino

Arduino Shields

אלו לוחות שתוכננו לביצוע משימה ספציפית וניתן לחברם באופן ישיר על גבי לוח ה- Arduino באמצעות פינים מאורכים בחלק העליון.

להלן כמה תמונות של לוחות המשמשים Shieldes

Arduino GSM Shield

Arduino WiFi Shield

Arduino Wireless SD Shield

Arduino Wireless Proto Shield

Arduino Ethernet Shield

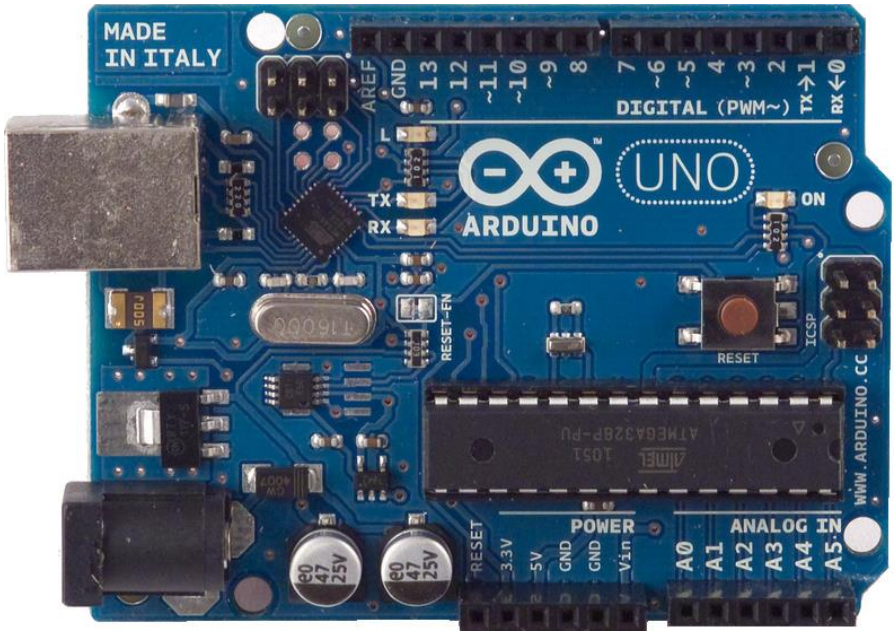
Arduino Wireless SD Shield

Arduino Motor Shield

Arduino Proto Shield

לוח Arduino Uno

בספר זה בחרנו להדגים את התוכניות על גבי לוח Arduino UNO.



להלן מפרט טכני של הלוח

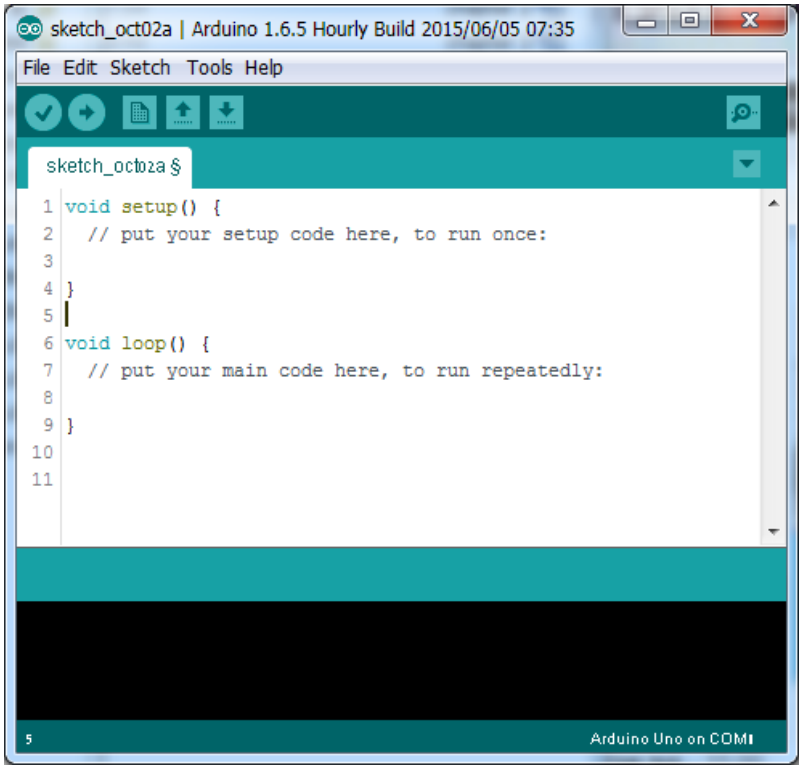
- מיקרו-בקר **Atmega328**
- מתח עבודה 5v למעבד (מתח אספקה לכרטיס נעים בתחום (12~ 7) .
- 20 כניסות ויציאות דיגיטאליות מתוכן 6 יציאות PWM ו- 6 כניסות אנלוגיות.
- זרם בהדקי I/O עד 40mA.
- זיכרון ROM מסוג Flash בגודל 32k Bytes .
- זיכרון RAM מסוג Sram בגודל 2k Bytes.

- זיכרון מסוג EEPROM בגודל 1Kbytes.
- תקשורת טורית מובנית מסוג UART .
- תקשורת טורית מובנית SPI
- תקשורת טורית מובנית I^2C.
- תדר שעון 16MHZ.
- לחצן Reset.
- 4 נורות LED המשמשות לחיווי הבא:
  - Led On משמשת לחיווי מתח הכניסה.
  - Led Tx משמשת לחיווי שליחה נתון בתקשורת טורית.
  - Led Rx משמשת לחיווי קבלת נתון בתקשורת טורית.
  - Led L מחוברת להדק 13.

סביבת פיתוח ב- Arduino

ל- Arduino יש סביבת פיתוח IDE (Integrated Development Environment) הכוללת עורך קוד (Editor), מהדר (Compiler) ובודק שגיאות Debugger שמטרתם ליצור סביבה נוחה לפיתוח ותכנת צריבה המובנת בסביבת העבודה.

להלן תמונה של מסך התוכנה:



להלן האייקונים שמופעים ב- Tools Bar. חלקו עליון של החלון:


(Verify) באמצעות איקון זה אנו מבצעים בדיקה לקוד התוכנית. במידה ובשורות תהיינה טעויות תחביריות הן תרשמנה בחלון התחתון.


upload באמצעות אייקון זה מבצעים קומפילציה מלאה לתוכנית ובנוסף צורבים/ טוענים את התוכנית למעבד.(המעבד כולל תוכנה פנימית המאפשרת קשר עם סביבת הפיתוח במחשב PC).


New באמצעות איקון זה ניתן לפתוח דף חדש ב- Editor (עורך) של התוכנה.


Open באמצעות איקון זה ניתן לפתוח תוכנית.


Save באמצעות איקון זה ניתן לשמור את התוכנית.


באמצעות אייקון זה ניתן לפתוח תוכנת Terminal לתקשורת טורית לשימוש כמוניטור.

כללים לכתיבת תוכנית בסביבת עבודה עם מעבד Arduino

- כל תוכנית חייבת לכלול פונקציה setup ופונקציה loop גם אם הפונקציות יישארו ריקות.
- **בפונקציה setup** מבצעים פעולות אתחול להדקי I/O (Input /Output), ליחידות הבקר, להתקנים חיצוניים ולכתיבת קוד. **פונקציה זו תתבצע פעם אחת בתחילת התוכנית**, לאחר חיבור המתח ללוח או לאחר פעולת Reset.
- **פונקציה loop** היא הפונקציה שהמיקרו-בקר מבצע לאחר פונקציה setup. פונקציה זו אינה מקבלת או מחזירה ערכים. כאשר עובדים עם חומרה עלינו לדאוג לכך שהתוכנית תעבוד כל הזמן עד לכיבוי המתח ולכן פונקציה זו היא אינסופית, קוד שייכתב בלולאה זו יתבצע אינסוף פעמים.
- רוב הפקודות מסתיימות בנקודה פסיק (;), רצוי לכתוב כל פקודה בשורה נפרדת.
- ניתן לכתוב הערות (comments) בשורת הקוד אחרי צמד התווים.. או בין התווים /* .....*/
- שימו לב השפה מבדילה בין אותיות גדולות לקטנות ולכן חשוב לשים לב לגודל האותיות בכל פקודה.
- אסור לכתוב פונקציה בתוך לולאת loop, המהדר אינו מאפשר פעולה זו, מכיוון שפעולה זו תיצור רקורסיה.

הכרת שפת Arduino

שפת Arduino היא שפה שדומה במידה רבה בתחביר שלה לשפת C ו ++C והיא משלבת בתוכה פקודות בשפת C ו- ++C. בפרק זה נלמד כמה פקודות בסיסיות, משפטי תנאי, לולאות ופונקציות. חומר זה נחוץ וישמש לכם בסיס לכתיבה תוכניות בסביבת עבודה Arduino.

הגדרת משתנה

הגדרת משתנה כוללת שני שדות: שדה להגדרת טיפוס המשתנה ושדה להגדרת שם המשתנה.

שם המשתנה	טיפוס המשתנה

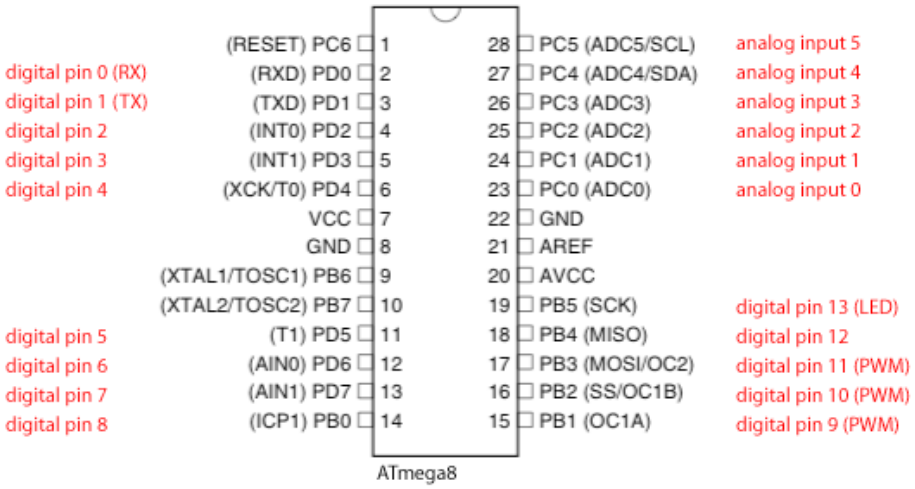
להלן טבלה גדלי טיפוסי המשתנים

Data Types	Bits	Bytes	Value Range
boolean	1	1	**0 ÷1(false or true)**
Byte	8	1	**0 ÷255**
Char	8	1	**-128 ÷127**
unsigned char	8	1	**0 ÷255**
Int	16	2	**-32768 ÷ 32767**
word	16	2	**0 ÷ 65535**
unsigned int	16	2	**0 ÷ 65535**
Long	32	4	**-214783648 ÷ 2147483647**
unsigned long	32	4	**0 ÷4294967295**
float	32	4	**±3. 4E-38 ÷ ±3.4E+38**

הדקי I\O (Input \ Output) של מיקרו-בקר Atmega328P

- באמצעות הדקים הנ"ל ניתן לחבר למעבד התקנים חיצונים הפועלים בצורה מקבילית או טורית. כגון חישנים, תצוגות LED, תצוגות LCD התקן Bluetooth ועוד.
- המיקרו-בקר מכיל שלושה פרוטים: פורט D, פורט B ופורט C.
- אין מעיון סיבית לבקר ניתן לפנות לכל הפורט.

להלן מבנה הדקים של הרכיב:



בסכימה הנ"ל רואים סימנים הן לפורט והן לפונקציונאליות נוספת.

PORTD

פורט זה מכיל שמונה סיביות דיגיטליות הממוקמות מהדק 0 עד הדק 7. ניתן לקרוא או לכתוב לכל הסיביות בפקודה אחת.

(D Data Direction Register) DDRD

באמצעות פקודה זו ניתן להגדיר את הדקי הפורט לכתיבה או קריאה. כאשר ערך הסיבית הרצויה תהיה שווה ל- 0 ההדק יוגדר ככניסה כאשר ערך הסיבית הרצויה תהיה שווה ל- 1 ההדק יוגדר כיציאה. יש לכתוב את פקודה זו בתוך פונקציה setup.

```
DDRD = 0xff;// set PORTD (digital 7~0) to outputs
```

(D Data Register) PORTD

באמצעות פקודה זו ניתן לקרוא או לכתוב לכל פורט D. פעולת כתיבה לפורט תתבצע כאשר נבצע השמה ל- PORTD

לדוגמא

```
PORTD = B00110111;
```

(D Input Pins Register) PIND

באמצעות פקודה זו ניתן לקרוא את כל ערך הסיביות של פורט D.

```
 Val= PIND;
```

PORTB

פורט זה מכיל שמונה סיביות דיגיטליות במסגרתן שש סיביות הנמוכות של הפורט ממוקמות מהדק 8 עד הדק 13. שתי הסיביות הגבוהות של פורט זה (7- 6) משמשות לכניסות תדר הגביש של הבקר ולכן אין לא בשימוש. ניתן לקרוא או לכתוב לכל הסיביות בפקודה אחת.

(B Data Direction Register) DDRB

באמצעות פקודה זו ניתן להגדיר את הדקי הפורט לכתיבה או קריאה כאשר ערך הסיבית הרצויה תהיה שווה ל- 0, ההדק יוגדר ככניסה כאשר ערך הסיבית הרצויה תהיה שווה ל- 1, ההדק יוגדר כיציאה. יש לכתוב את פקודה זו בתוך פונקציה setup.

```
DDRB = 0x3f;  // set PORTB (digital 8~13) to outputs
```

(B Data Register) PORTB

באמצעות פקודה זו ניתן לקרוא או לכתוב לכל פורט B.

פעולת כתיבה לפורט תתבצע כאשר נבצע השמה ל- PORTB

לדוגמא

```
PORTB = B00110111;
```

(B Input Pins Register) PINB

באמצעות פקודה זו ניתן לקרוא את כל ערך הסיביות של פורט B.

```
Val= PINB;
```

PORTC

פורט זה מכיל שש סיביות אנלוגיות או דיגיטליות אשר ממוקמות מהדק A0 עד הדק A5. ניתן לקרוא או לכתוב לכל הסיביות בפקודה אחת.

DDRC (C Data Direction Register)

באמצעות פקודה זו ניתן להקצות את הדקי הפורט לכתיבה או קריאה. כאשר ערך הסיבית הרצויה תהיה שווה ל- '0' ההדק יוגדר ככניסה וכאשר ערך הסיבית הרצויה תהיה שווה ל- '1' ההדק יוגדר כיציאה. יש לכתוב פקודה זו בתוך הפונקציה setup.

```
DDRC = 0x3f; // set PORTC (A0 ~A5) to outputs
```

PORTC (C Data Register)

באמצעות פקודה זו ניתן לקרוא או לכתוב לכל פורט C. פעולת כתיבה לפורט תתבצע כאשר נבצע השמה ל- PORTC

לדוגמא

```
PORTC = B00110111;
```

PINC (C Input Pins Register)

באמצעות פקודה זו ניתן לקרוא את כל ערך הסיביות של פורט C.

```
Val= PINB;
```

דוגמא לתוכנית הדוגמת שישה מתגים. במידה והמתג מחובר ל- VCC יש להדליק נורת LED מתוך שורה של ששת הנורות המציינת את מיקום המתג. חברו את מתגי ה- Dip Switch להדקים 7 ~ 2 (ששת הסיביות הגבוהות של פורט D) ואת נורות ה-LED להדקים 13 ~ 8 (פורט B).

```
byte ValDipSwitch;
void setup(){
  DDRB=B111111;
  DDRD=B00000000;
 }//End setup

void loop(){
  ValDipSwitch= PIND;
  PORTB= ValDipSwitch>>2;
}//End of loop function
```

הגדרת כניסות ויציאות דיגיטאליות

ההדקים של המיקרו-בקר יכולים לשמש ככניסות או יציאות דיגיטאליות (יציאה המוציאה מתח של '1' לוגי או '0' לוגי). לכן, לפני השימוש בהם עלינו להגדירם למצב כניסה או יציאה. פעולה זו תתבצע באמצעות פונקציה pinMode, אשר מקבלת שני פרמטרים:

- פרמטר ראשון, מציין את מספר ההדק של לוח ה- Arduino .
- פרמטר שני, קובע את מצב ההדק לאחד משלושת המצבים הבאים:
  - INPUT - כניסה.
  - INPUT_PULLUP - כניסה עם נגד המחובר למתח 5V.
  - OUTPUT - יציאה.

להלן פורמט הפונקציה:

```
pinMode (מספר ההדק, מצב ההדק);
```

כתיבת להדק דיגיטאלי

פעולה זו מתבצעת באמצעות פונקציה digitalWrite, אשר מקבלת שני פרמטרים:

פרמטר ראשון קובע את מספר ההדק בלוח ה- Arduino.

פרמטר שני קובע את מצב הלוגי של ההדק, בשני אופנים:

- רמת לוגית של '0' לוגי = LOW = 0.
- רמת לוגית של '1' לוגי = HIGH = 1

ניתן לכתוב פונקציה זו בכל פונקציה.

להלן הפורמט של הפונקציה

```
digitalWrite(מספר הדק,מצב לוגי);
```

```
//=============================================
#define Led  2
void setup(){
  pinMode(Led,OUTPUT);
 }//End of setup function
void loop(){
    digitalWrite(Led,1);
    delay(1000);
    digitalWrite(Led,0);
    delay(1000);
    }//End of loop function
//=============================================
```

קריאה ערך הכניסה מהדק דיגיטאלי

פעולה זו מתבצעת באמצעות פונקציה digitalRead, אשר מקבלת פרמטר המציין את מספר ההדק אשר שאנו מעוניינים לקרוא. הפונקציה מחזירה ערך 0 או 1. לכן, כדאי להגדיר משתנה מטיפוס הכי קטן char/byte וזאת על מנת לחסוך בזיכרון משתנה זה ישמור את ערך המוחזר.

להלן פורמט הפונקציה:

```
ValInput=digitalRead(מספר הדק לקריאה);
```

דוגמא לתוכנית הדוגמת (קוראת) את ערך לחצן המחובר להדק 2 וכותבת את ערכו לנורת LED המחוברת דרך הדק 3 . יש להגדיר את מצב ההדק למצב INPUT.

```
//================================================
byte SW1 = 2; //The number of the pushbutton pin
byte Led = 3;
void setup(){
 pinMode( Led,OUTPUT);
 pinMode( SW1, INPUT);
}//End setup
void loop(){
   digitalWrite(Led,digitalRead(SW1));
  }//End of loop function
//================================================

//================================================
```

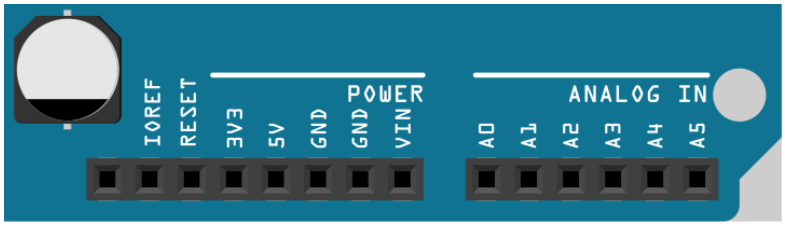
קריאה מכניסות אנלוגיות

פעולה זו מתבצעת בעזרתה של פונקציה analogRead, אשר מקבלת פרמטר המציין את הכניסה שאנו מעוניינים לקרוא. הפונקציה מחזירה את הערך הדיגיטאלי המייצג את המילה.

להלן הפורמט של הפונקציה:

```
analogRead(מספר הדק);
```

לוח Arduino Uno כולל שש כניסות אנלוגיות הממוקמות בין ההדקים A0 ~ A5 להלן תמונת ההדקים של הלוח:



לדוגמה

```
val = analogRead(A0);// read the input pin

val = analogRead(0);// read the input pin
```

דוגמה לשאלה על רזולוציה

יש לחשב את הערך הדיגיטאלי שמתקבל ממיר אנלוגי בגודל 10 סיביות שמחובר למתח ייחוס 5V כאשר ערך מתח הכניסה שווה ל- 4.3V?

פתרון

בשלב הראשון נחשב את ה- Resolution, את התוצאה שנקבל נציב בנוסחא לחישוב הערך הדיגיטאלי.

$$Resolution = \frac{5}{1024} = 4.88\text{mV}$$

$$\text{D}out = \frac{4.3}{0.00488} = 880.64$$

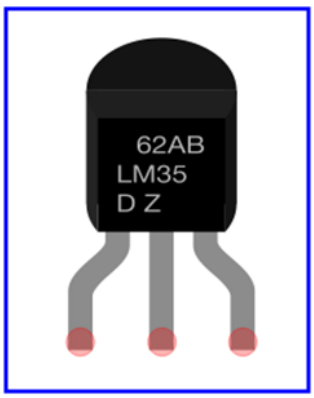
בהתאם לסוג הממיר התוצאה תהיה 880 או 881.

בחיבור אנלוגי יש המון רכיבים שניתן לחבר. על התלמיד לדעת מה המרה שהתלמיד מבצע מערך ספרתי לערך הרצוי. לדוגמה דגימת חיישן טמפרטורה LM35.

חיישן טמפרטורה LM35

סדרת LM35 הינם חיישני טמפרטורה המשולבים במערכות לבקרת האקלים והטמפרטורה. קריאת הטמפרטורה בחיישן זה נוחה מאוד ופשוטה מכיוון שמתח המוצא שלו משתנה באופן ליניארי, בהתאם לשינוי הטמפרטורה במעלות צלזיוס (Celsius) ביחס של 10mV לכל מעלת צלסיוס. תחום הטמפרטורות שרכיב זה עובד נע בין -55°C ~ 150°C ופועל במתחי הספקה של 4V ~ 20V

להלן מבנה האריזה:



חישוב ערך הטמפרטורה עם רכיב ADC ברזולוציה של 10 סיביות

כדי לחשב את ערך הטמפרטורה עלינו להשתמש בנוסחא לחישוב ערך הדיגיטאלי במוסברת בתחילת הפרק **להלן נוסחא לחישוב ערך הדיגיטאלי**

$$\text{D}out = \frac{Vin}{\text{R}esolution}$$

Vin מתח הכניסה לרכיב ה- ADC מאחר וכל מעלה בטמפרטורה מתורגמת לערך של 10mV בחיישן. ניתן לומר שמתח הכניסה שווה ערך הטמפרטורה שנדגמת כפל 10mV.

$$Vin = temp \times 10mV$$

נציב את הערכים בנוסחא לחישוב ערך דיגיטאלי:

$$adcValue = \frac{temp \times 10mV}{\frac{5}{1204}}$$

ניתן לכתוב את הנוסחא בצורה הבאה:

$$adcValue = temp \times 10mV \times \frac{1024}{5}$$

מכיוון שמעוניינים למצוא את ערך הטמפרטורה נבצע שינוי בנוסחא למציאת adcValue ונבודד את פרמטר temp

להלן הנוסחא שמתקבלת למציאת ערך הטמפרטורה לאחר השינוי:

$$temp = \frac{adcValue}{10mV} \times \frac{5}{1024} = \frac{5 \times adcValue}{10.24} = 0.488xadcValue$$

דוגמא לחישוב ערך הדיגיטאלי וערך הטמפרטורה שמתקבלים כאשר הטמפרטורה שנמדדת שווה לערך 20°C.

לשם כך נשתמש בנוסחא למציאת adcValue:

$$adcValue = 20 \times 10mV \times \frac{1024}{5} \approx 40$$

נציב כעת את התוצאה שהתקבלה בנוסחא לחישוב ערך הטמפרטורה:

$$temp = \frac{adcValue}{10mV} \times \frac{5}{1024} = \frac{5 \times 40}{10.24} = 19.53$$

ניתן לראות שנקבל סטייה של בסביבות חצי מעלה.

```
temp = (adcValue / 10.24)*5;

temp = (adcValue / 1024)*500;
```

ניתן לחבר הרבה חיישנים אנלוגיים לפרויקט. התלמיד צריך להסביר על רכיב ADC ומה המרה שביצע מערך ספרתי לתוצאה שמציג על המסך או לשימוש בו הוא עושה.

נוח להשתמש עם פונקציה map

באמצעות פונקציה זו ניתן להמיר מספר מתחום המספרים הנוכחי לתחום היעד. פונקציה זו מקבלת חמישה פרמטרים ומחזירה ערך:

פרמטר 1- מייצג את המספר שאנו מעוניינים להמיר.

פרמטר 2- מייצג את הערך הנמוך של התחום הנוכחי.

פרמטר 3 – מייצג את הערך הגבוה של התחום הנוכחי.

פרמטר 4- מייצג את הערך הנמוך של תחום היעד.

פרמטר 5- מייצג את הערך הגבוה של תחום היעד.

הפונקציה מחזירה את המספר המומר.

להלן פורמט הפונקציה:

```
val = map(מספר, from Low, from High, to Low, to High);
```

לדוגמא, המרת מספר מתחום 0~1023 אל תחום 0~255

```
val = map(val, 0, 1023, 0, 255);
```

פונקציות ליצירת tones

באמצעות פונקציות אלו ניתן ליצור גל ריבועי בעל DC של 50% ביציאות הדיגיטאליות של הלוח. השימוש בפונקציה אלו יפריע לעבודה תקינה של יציאות PWM ביציאות בהדק 3 ובהדק 11, כמו כן לא ניתן ליצור תדר נמוך מ- 31Hz.

פונקציה tone

פונקציה זו מאפשרת יצירת צלילים באמצעות שתי אופציות:

אופציה ראשונה

הפונקציה מקבלת שני פרמטרים פרמטר הראשון, מציין את מספר ההדק המחובר לרמקול והפרמטר השני מציין את התדר.

להלן פורמט הפונקציה:

tone(מספר ההדק, תדר) ;

אופציה שניה

באופציה זו הפונקציה מקבלת שלושה פרמטרים פרמטר הראשון, מציין את מספר ההדק המחובר לרמקול, הפרמטר השני מציין את התדר והפרמטר השלישי מציין את משך השמעת התדר.

להלן פורמט הפונקציה:

tone(מספר ההדק, תדר, זמן השמעה) ;

פונקציה noTone

פונקציה זו מפסיקה את פעולת הפונקציה ton שנבחרה ומקבלת פרמטר המציין את מספר ההדק.

להלן פורמט הפונקציה:

noTone(מספר ההדק) ;

הגדרת יציאות אנלוגיות (PWM)

ניתן להפעיל יציאות אנלוגיות המסוגלות לשנות את מתח היציאה על ידי שימוש בשיטת בקרת הספק באמצעות אפנון רוחב פולס PWM. שפת Arduino כוללת פונקציות ליצירת יציאות אלו.

יציאה אנלוגית המשנה את מתח היציאה על ידי שינוי מספרי

על מנת לשנות את מתח היציאה יש להשתמש בשיטת אפנון רוחב פולס PWM (Pulse Width Modulation) .

בקרת הספק באמצעות אפנון רוחב פולס PWM

שיטה זו הינה שיטה חסכונית ונפוצה לשליטה על הספק חשמלי, הנמסר לצרכן בהפעלת מערכות חשמליות שונות לדוגמה: הפעלת נורות, מנועי DC, מנועי RC-Servo ועוד. בשיטה זו מחברים גל ריבועי מחזורי אל הצרכן ועל ידי שינוי D.C (Duty Cycle) של האות מספקים מתח המשתנה כתלות בזמן. משך הזמן במסגרתו המתח נמצא באפס וולט נקרא Toff ואילו משך הזמן במסגרתו המתח נמצא ברמה גבוהה (5V) נקרא Ton. המתח הממוצע שמקבל הצרכן נקבע על ידי מיתוג מקור מתח כפול ה-D.C. כאשר אנו עובדים עם רכיבים פריפריאליים, מתח הספק נע בדרך כלל בין 4.8 וולט ל - 6 וולט ונהוג לחברו למתח של 5 וולט.

להלן נוסחא לחישוב מתח ממוצע (Uave):

$$\mathrm{U}ave = Umax \cdot D.C(\text{Duty Cycle}) = Umax \cdot \frac{Ton}{T}$$

כתיבה ליציאה אנלוגית (PWM) בשפת Arduino

פעולה זו מתבצעת בעזרתה של פונקציה analogWrite, אשר מקבלת שני פרמטרים:

פרמטר ראשון, מציין את מספר הדק יציאת ה- PWM של המיקרו-בקר.

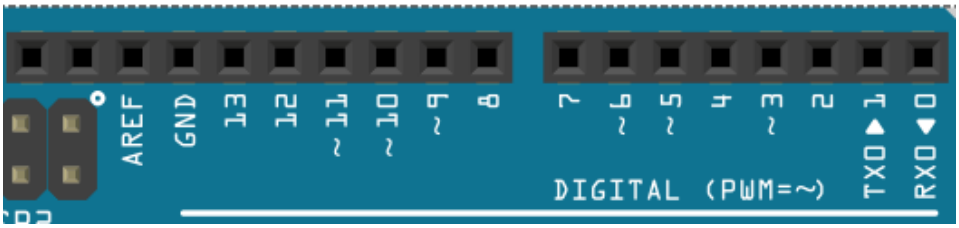
פרמטר שני, קובע את הערך המספרי של Ton שתחומו בין 255 ~ 0.

להלן הפורמט של הפונקציה.

```
analogWrite(מספר הדק,Ton(0~255));
```

ללוח Arduino Uno כולל שש יציאות אנלוגיות (PWM) יציאות אלו ממוקמות בין ההדקים 13 ~ 0 כאשר הן מסומנות על ידי הסמן (~ PWM)

להלן תמונה ההדקים של הלוח



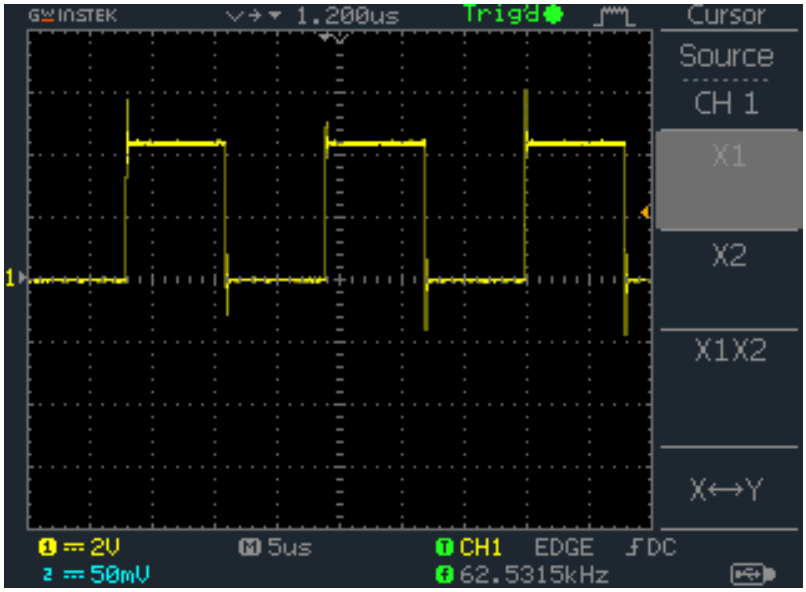
מהתמונה לעיל ניתן לראות את ששת יציאות ה- PWM הממוקמות בהדקים:

~ 3,~ 5, ~6 , ~9 , ~10 , ~11

להלן דוגמה להפעלת מוצאה של 3 למתח של 2.5V:

```
analogWrite(3,127);
```

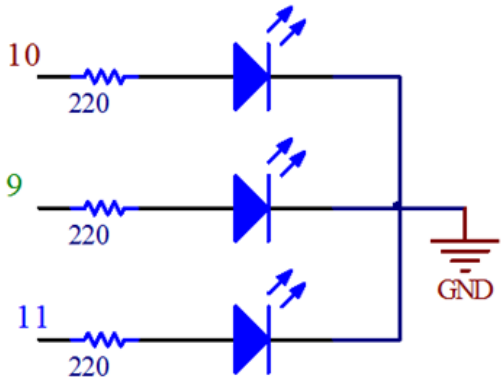
להלן דוגמה של אות מוצא של גל ריבועי בתדר של 62.5kHz בעל Duty Cycle של 50%.



באמצעות PWM ניתן לשנות עוצמת ההארה של נורת LED

```
//================================================
#define RedLed 10
#define GreenLed 9
#define BlueLed 11
void setup() {
  }//End of setup function
void loop(){
int i;
for(i=0 ;i <  256; i++){
    analogWrite(RedLed   ,  0);
    analogWrite(GreenLed,  i);
    analogWrite(BlueLed ,  0);
    delay(150);
   }
}//End of loop function
//================================================
```

להלן אופן החיבור:



שאלה שניתן לשאול את התלמיד

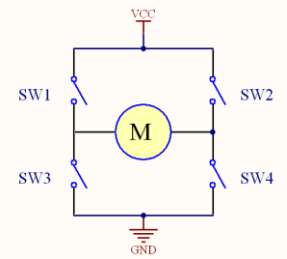
מה מתח ההולכה של נורת ה - LED?

אם ידוע שמתח ההולכה של ה LED הירוק הוא 3.3V. כאשר אני עובד 50% D.C מתקבל מתח של 2.5V האם נורת ה- LED תפעל?

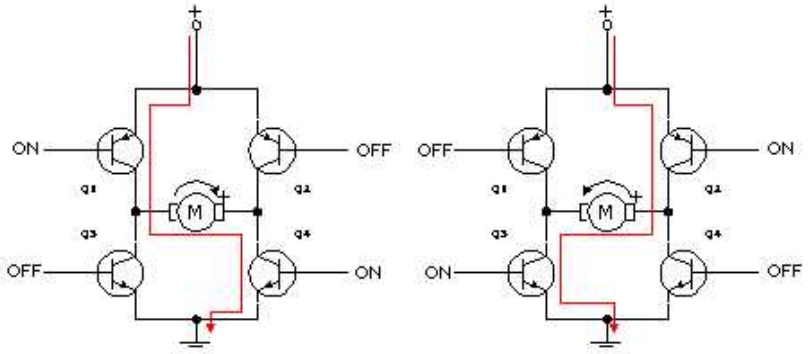
מנוע DC

מנוע חשמלי שפועל על מתח ישיר (DC), בדרך כלל מסוללות. להפעלה פשוטה של המנוע דרוש רכיב שיספק את הזרם למנוע עם בקרת שליטה על החלפת קוטביות. החלפת הקוטביות תתבצע באמצעות גשר H- ארבעה מתגים (או טרנזיסטורים) המסודרים בצורה שדומה לאות H, כל H- גשר יכול לשלוט בשני מנועים.

להלן מבנה של ארבעה מתגים המסודרים בצורה H ומחוברים למנוע DC:

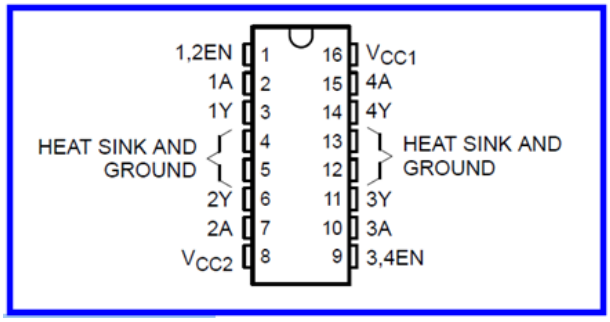


כעת נדגים את מבנה של ארבעה טרנזיסטורים המסודרים בצורה H ומחוברים למנוע DC

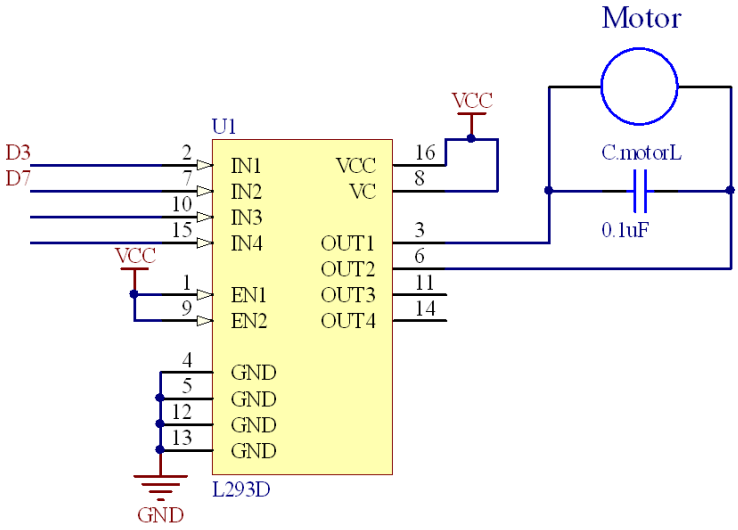


בפרק זה נשתמש ברכיב L293D המספק זרם וכולל גשר H.

להלן מבנה האריזה:



להלן הסכימה לחיבור המנוע למיקרו- בקר :



שאלות

מה הזרם של המנוע ?

מדוע הוא משתמש הרכיב נוסף ולא מחבר את יציאת ההדק מהבקר ישירות להדק המנוע ?

לבקר למנוע?

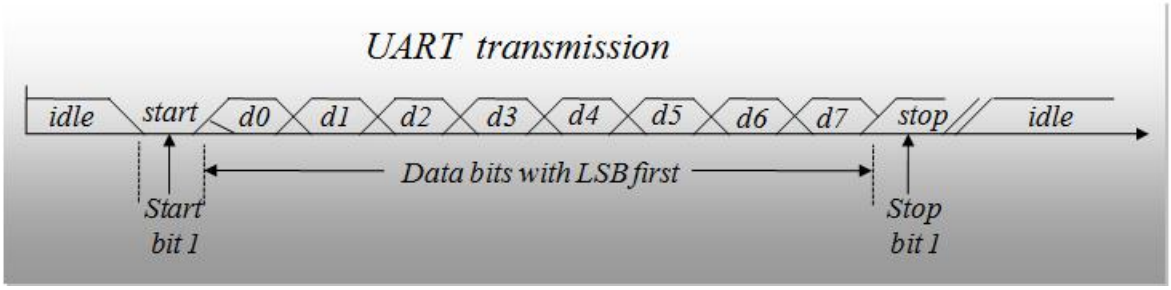
עיקרון פעולת גשר H?

תראה לי את השלבים מקביעת הדקים ועד לשינוי מהירות / הפעלת המנוע בקוד התוכנית?

תקשורת טורית UART (Universal **A**synchronous **R**eceiver **T**ransmitter**)**

תקשורת זו היא תקשורת אסינכרונית – כלומר תקשורת נעשית ללא העברת אות שעון משותף בין שתי היחידות. תקשורת זו מאפשרת למיקרו-בקר לתקשר עם התקנים חיצונים כגון מחשב PC, מודם, התקן Bluetooth ועוד.

מבנה התשדורת הטורית של UART



התחלת התשדורת הטורית נעשית באמצעות סיבית התחלה (Start Bit), המסנכרנת בין יחידות התקשורת ומסמנת ליחידה המקבלת על התחלת התשדורת. לאחר שידור סיבית התחלה משודרות סיביות התו (סיביות התו מכילות שמונה סיביות, כאשר מתחילים בסיבית הנמוכה (LSB) ומסיימים בסיבית הגבוהה (MSB). כל סיבית משודרת למשך הזמן הקצוב שנקבע מראש (ברוחב זהה לרוחב סיבית ההתחלה). בפרק זה בחרנו

לעבוד בקצב השידור המוסכם המקובל של 9600 Bits Per Second כלומר הזמן המוקצה לכל סיבית הוא $\frac{1}{9600} \cong 104\mu sec$) ואחריהן משודרת סיבית עצירה (Stop Bit).

במהלך התשדורת מקצים לכל סיבית זמן קבוע ומוסכם מראש. זמן זה נקבע על ידי קצב ההעברה (Baud Rate) של ה – UART. קצבי ההעברה של ה- UART יכולים להיות כלשהם, אך קצבי ההעברה הבאים הם הקצבים המקובלים ביותר:

- קצב של 2400 Bits Per Second
- קצב של 4800 Bits Per Second
- קצב של 9600 Bits Per Second
- קצב של 14400 Bits Per Second
- קצב של 19200 Bits Per Second
- קצב של 38400 Bits Per Second
- קצב של 57600 Bits Per Second
- קצב של 115200 Bits Per Second

תשדורת הסיביות הכולל: סיבית התחלה, סיביות מידע של התו, סיבית בדיקה וסיבית עצירה נקרא מסגרת (Frame).

כדי לתרגל את הפקודות נשתמש בחלון התקשורת הטורית המשמש אותנו כמסך פלט. בשפת C בעבודה עם מחשב PC שליחת נתונים למסך הפלט מתבצע על ידי פונקציות שמוגדרות בספרייה stdio הכוללת את פונקציה printf. בסביבת Arduino השימוש בחלון התקשורת הטורית דורש הפעלה של יחידת התקשורת הטורית. בשלב זה נלמד כיצד להפעיל את היחידה ולהשתמש בפונקציות הבסיסיות שלה.

פונקציה begin

באמצעות פונקציה זו מאפשרים עבודה עם תקשורת טורית. בנוסף פונקציה זו מקבלת פרמטר מטיפוס long אשר משמש להגדרת קצב תשדורת הנתונים (Baud Rate) בשלב זה נעבוד קצב של 9600 Bits Per Second.

להלן דוגמא להגדרת תקשורת טורית לקצב העברה של 9600BPS.

```
void setup() {
Serial.begin(9600);// open the serial port at 9600 bps:
}//End of setup function
```

פונקציה print

באמצעות פונקציה זו ניתן לשלוח מחרוזות מהמיקרו-בקר של לוח Arduino אל מחשב ה- PC וניתן להשתמש בה בשתי אופציות:

1. Serial.print(Val);
2. Serial.print(Val , format);

אופציה שניה לפונקציה Serial.print היא קריאה לפונקציה ושליחת שני פרמטרים לפונקציה. כאשר הפרמטר הראשון מייצג מחרוזת והפרמטר השני את הבסיס (Radix) אותו נרצה להציג את המספר.

להלן טבלת ייצוג בסיסים בשפת Arduino.

ייצוג הבסיס בשפת Arduino	בסיס
BIN	בינארי / בסיס 2
OCT	בסיס אוקטי / בסיס 8
DEC	בסיס עשרוני/ בסיס 10
HEX	בסיס הקסהדצימלי/ בסיס 16

פונקציה println

מבחינה פוקציונלית פונקציה זו מבצעת את הפעולה של פונקציה print ובנוסף מורידה את הסמן לתחילת השורה הבאה (\n).

פונקציה end

באמצעות פונקציה זו עוצרים את התקשורת הטורית בין המיקרו-בקר ליחידת התקשורת.

להלן פורמט הפונקציה:

```
Serial.end();
```

פונקציה write

באמצעות פונקציה זו ניתן לשלוח נתונים מהמיקרו-בקר למחשב. הנתונים יכולים להיות מספרים נומרים או התווים בקוד- ASCII ניתן להשתמש בפונקציה זו בשלוש אופציות:

1. Serial.write(byte);
2. Serial. write (str);
3. Serial. write (buf,len);

פונקציות קלט

פונקציות אלו משמשות לקליטת נתונים בתקשורת טורית ממחשב ה-PC או מכל התקן העובד בתקשורת טורית UART. ישנם מספר פונקציות הקשורות לקבלת נתונים. בפרק זה נדגים כמה שימושים בפונקציות אלו.

ליחידת הקלט FIFO (First In Fist Out)

פונקציה available (זמין)

פונקציה זו מחזירה את מספר התווים שהתקבלו בתקשורת טורית. התווים שהתקבלו מאוחסנים ב-Buffer העובד עם יחידת FIFO (First In Fist Out) וגודלו 64Bytes. במידה וב- buffer לא מאוחסן נתון הפונקציה תחזיר את הערך 0.

לדוגמא:

```
if (Serial.available() > 0)
```

פונקציה read

באמצעות פונקציה זו קוראים נתונים מאוחסנים ב- buffer.

להלן דוגמא לקריאת נתון ושמירתו במשתנה

```
SBUF = Serial.read();
}//End of loop function
```

פונקציה flush

שולפת את כל הנתונים שמאוחסנים ב- Buffer של יחידת התקשורת הטורית.

להלן פורמט הפונקציה:

```
Serial.flush();
```

פונקציה readBytes

פונקציה זו מאפשרת לקרוא מספר נתונים מטיפוס byte אשר מאוחסנים ב- Buffer של יחידת התקשורת הטורית ומקבלת שני פרמטרים: פרמטר ראשון מציין את כתובת המערך מטיפוס byte ופרמטר שני מציין את מספר הנתונים שמעוניינים לקרוא.

להלן פורמט הפונקציה:

```
Serial.readBytes(buffer, length);
```

פונקציה readString

באמצעות פונקציה זו ניתן לקרוא מחרוזת המאוחסנת במחסנית התקשורת הטורית אל מחרוזת יעד.

להלן פורמט הפונקציה:

```
Serial.readString();
```

פונקציה readStringUntil

באמצעות פונקציה זו ניתן לקרוא מחרוזת המאוחסנת במחסנית התקשורת הטורית אל מחרוזת יעד עד למציאת התו אשר נשלח לפונקציה.

להלן פורמט הפונקציה:

```
Serial.readStringUntil(char);
```

פונקציה find

באמצעות פונקציה זו ניתן לבדוק האם מחרוזת המכילה תו או מספר תווים נמצאת ברצף של נתונים שהתקבלו בתקשורת טורית. הפונקציה מחזירה true (אמת) במידה ויש התאמה ו- false (שקר) במידה ולא.

להלן פורמט הפונקציה:

```
Serial.find(target);
```

פונקציה setTimeout

פונקציה זו מאפשרת למיקרו-בקר לצאת ממצב המתנה בתקשורת טורית כשלא מתקבלים נתונים בתקשורת טורית. הפונקציה מקבלת פרמטר המייצג מספר מטיפוס long כאשר זמן ההמתנה הוא במילי שניות וברירת מחדל נקבעה ל- 1000msec.

להלן פורמט הפונקציה:

```
Serial.setTimeout(time);
```

פונקציה parseInt

פונקציה זו מחזירה מספר מטיפוס int שנמצא ברצף נתונים אשר התקבלו בתקשורת טורית ומדלגת על תווים שאינם ספרות או סימן מינוס. הפונקציה תחזיר את הערך 0 אם רצף התווים שמאוחסנים ב- buffer לא ייצגו ספרות. במידה ולא נשתמש בפונקציה Serial.available. חזרה מהפונקציה תתבצע באמצעות הפונקציה Serial.setTimeout.

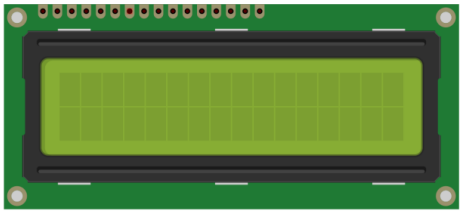
להלן פורמט הפונקציה:

```
Serial.parseInt();
```

תרגול

דוגמא לתוכנית המממשת מונה בגודל 6Bits (תחום 0 ~ 64) התוחמת את הערך העליון והתחתון של המונה. קידום ערך המונה באחד יתבצע באמצעות שליחת התו '+' בתקשורת טורית. חיסור באחד מערך המונה יתבצע באמצעות שליחת התו '-' בתקשורת טורית . את יציאות המונה יש לחבר לנורות ה- LED דרך פורט B

מבוא לתצוגה LCD אלפאנומרית



תצוגת LCD, Liquid Crystal Display

להלן פירוט הדקי רכיב אריזת התצוגה:

Pin No.	Name	Function
1	GND	Ground supply voltage
2	VCC	+5 Supply voltage
3	VEE	Contrast control pin
4	RS	Register (Data) Select control input
5	R/W	Read / Write (active low) control input
6	E	Enable control input
7	D0	Data bit 0 of I/O
8	D1	Data bit 1 of I/O
9	D2	Data bit 2 of I/O
10	D3	Data bit 3 of I/O
11	D4	Data bit 4 of I/O
12	D5	Data bit 5 of I/O
13	D6	Data bit 6 of I/O
14	D7	Data bit 7 of I/O

ספריית LiquidCrystal

ספרייה זו מאפשרת ללוחות Arduino לשלוט על תצוגות LCD אלפאנומריות המבוססות על מיקרו- בקר Hitachi HD44780 (או תואם), הנמצא על רוב צגי LCD אלפאנומרים. הספרייה תומכת בעבודה עם רוחב BUS של ארבע סיביות או רוחב BUS של שמונה סיביות קווי בקרה. כפי שציינו בחרנו לעבוד עם רוחב BUS של ארבע סיביות ושני קווי בקרה SR ו- E.

מכיוון שבפרק זה נדגים רק כתיבה לתצוגה חיברנו את קו הבקרה $R/\overline{W}$ ל- GND.

בעת שימוש בספרייה זו יש להצהיר על קובץ LiquidCrystal. h כאשר משתמשים בספרייה זו.

פונקציה LiquidCrystal

```
LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
```

פונקציה begin

```
lcd.begin(cols, rows);
```

פונקציה print

1. lcd.print(data);
2. lcd.print (data , BASE);

פונקציה write

באמצעות פונקציה זו כותבים תווים על מסך התצוגה.

להלן פורמט הפקודה:

```
lcd.write(data);
```

פונקציה clear

באמצעות פונקציה זו ניתן למחוק את התווים שמופעים על מסך התצוגה. בסיום פעולת המחיקה סמן התצוגה יחזור לתחילת התצוגה.

להלן פורמט הפקודה:

```
lcd.clear();
```

פונקציה home

פונקציה זו מעבירה את סמן התצוגה לתחילת התצוגה. להלן פורמט הפקודה

```
lcd.home();
```

פונקציה cursor

פונקציה זו מאפשרת הצגת הסמן על גבי מסך התצוגה.

להלן פורמט הפקודה:

```
lcd.cursor();
```

פונקציה noCursor

פונקציה זו מבטלת את הצגת הסמן על גבי מסך התצוגה.

להלן פורמט הפקודה:

```
lcd.noCursor();
```

פונקציה setCursor

פונקציה זו ממקמת את הסמן על גבי מסך התצוגה.

להלן פורמט הפקודה

```
lcd.setCursor(col ,rows);
```

ספרייה Software Serial

מממשת את פרוטוקול UART בתוכנה ובאמצעותה ניתן לתקשר עם התקן העובד בתקשורת טורית בהדקים דיגיטאליים.

הספרייה מכילה פונקציות שכדי להשתמש בהן יש להצהיר על שימוש בקובץ כותרת הבא:

```
#include < SoftwareSerial.h>
```

פונקציה SoftwareSerial

באמצעות פונקציה זו מגדירים את שם האובייקט של הספרייה ואת ההדקי ה- UART בתוכנה. הפונקציה מקבלת שני פרמטרים: פרמטר ראשון מציין את מספר ההדק שישמש כהדק Rx ופרמטר שני מציין את מספר ההדק שישמש כהדק Tx. יש לקרוא לפונקציה בתחילת התוכנית ובנוסף יש לקבוע את שם לאובקייט שאיתו נעבוד.

דוגמא לפורמט הפונקציה:

```
SoftwareSerial mySerial = SoftwareSerial(2, 3);
```

בדוגמא הנ"ל ניתן לראות ששם של האובייקט שנבחר הוא mySerial (שם שרירותי) נמשיך להשתמש עם שם זה במהלך כל הפרק. בנוסף ניתן שהפונקציה מקבלת שני פרמטרים: פרמטר הראשון מציין שהדק 2 מחובר Rx והפרמטר השני מציין שהדק 3 מחובר Tx.

מומלץ להשתמש בהנחיית define על מנת לתת שמות להדקים לדוגמא:

```
#define rxPin 2
#define txPin 3
SoftwareSerial mySerial =  SoftwareSerial(rxPin, txPin);
```

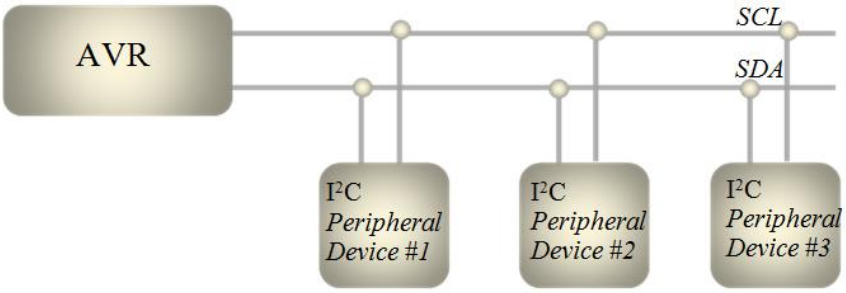
שימו לב! פונקציות של ספרייה זו זהות מבחינת פונקצונילות לפוקציה ספרייה של serial בחומרה.

פרוטוקול I^2C

פרוטוקול I^2C (Inter-Integrated Circuit) הוא פרוטוקול של BUS, בעל שני חוטים בלבד לתקשורת עם נתונים בקצב איטי עד בינוני.

ה - BUS הטורי של I^2C פועל ב - Half-Duplex, כלומר אי אפשר לכתוב ולקרוא באותו הזמן. ה – BUS כולל שני חוטים:

- אות שעון טורי – SCL (Serial Clock) שתפקידו לסנכרן את המידע המועבר בין שני רכיבים.
- אות מידע טורי – SDA (Serial Data) שתפקידו להעביר מידע בין שני רכיבים.



האותות המופעים לעיל הם דו כיוניים וכל אחד מהם מחובר ברכיב כיציאות מסוג Open-Drain. בנוסף, האותות הנ"ל מהודקים באופן חיצוני למתח VCC באמצעות Pull-Up Resistors.

הרכיב שיוזם את התקשורת ומספק את אות השעון נקרא Master והרכיב שאליו פונים נקרא ה – Slave. לרוב ה – Master הוא מיקרו-בקר וה – Slaves הם רכיבים פריפריאליים כמו חיישנים, פוטנציומטרים, רכיבי המרה אנלוגיים (DAC ו- ADC), רכיבי תצוגה והתקני זיכרון. רכיב ה- Slave שאליו פונים איטי והוא מסוגל להשאיר את אות השעון במצב Low (Clock stretching) עד שהוא מוכן להמשיך, לפיכך ניתן לחבר רכיבים איטיים ל – Master מהיר.

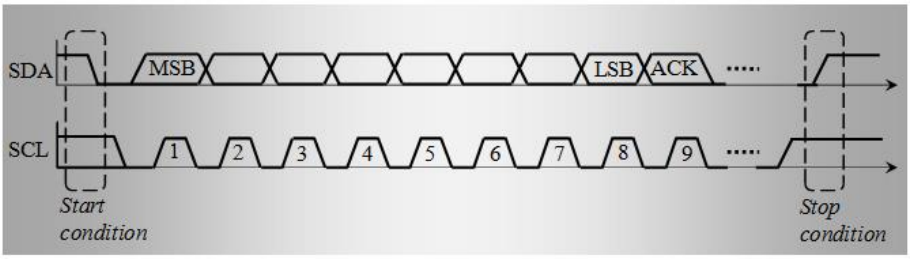
פעולות בסיסיות ברמת הביטים

כאשר שני האותות, SCL ו- SDA נמצאים במצב '1' לוגי, ה- BUS מצוי במנוחה.

התשדורת נעשית בשיטת ה – MSB – First.

בהעברת המידע חייבים להתקיים הכללים הבאים:

1. האות SDA חייב להיות יציב כשהאות SCL נמצא במצב '1' לוגי.
2. האות SDA יכול להשתנות רק כשהאות SCL נמצא במצב '0' לוגי.



שני היוצאים מן הכלל מתרחשים כאשר ה – Master מבצע את הפעולות הבאות:

- פעולת Start – ירידה של SDA, כאשר SCL נמצא ב – '1' לוגי.
- פעולת Stop – עליה של SDA, כאשר SCL נמצא ב – '1' לוגי.

ספריית Wire שמובנית בסביבת העבודה של Arduino

ספריה זו מפעילה את יחידת TWI של הבקר ההופכת את השימוש בפרוטוקל זה לנוח ומשמשת לתקשורת בין רכיבים העובדים בפרוטוקול $I^2C\backslash TWI$. הספרייה מכילה פונקציות. כדי לעבוד איתן יש להצהיר על שימוש בקובץ כותרת (HEADER FILE) קובץ כותרת הינו קובץ המכיל הצהרות לפונקציות/משתנים גלובליים וכדומה. ומקבל סיומת Wire.h. כפי שציינו הדק A5 ישמש כהדק SCL והדק A4 ישמש כהדק SDA.

פונקציה Wire.begin

זוהי פונקציה אשר באמצעותה מאתחלים את יחידת $I^2C\backslash TWI$ של המעבד לעבודה בפרוטוקול $I^2C\backslash TWI$ וקובעים שהמיקרו-בקר ישמש כרכיב Master. מכיוון שפונקציה זו משמשת לאתחול היחידה יש לכתוב אותה בפונקציה setup.

להלן פורמט הפונקציה:

```
Wire.begin();
```

פונקציה Wire.begin(Address)

זוהי פונקציה אשר באמצעותה מאתחלים את יחידת $I^2C\backslash TWI$ של המעבד לעבודה בפרוטוקול זה.

המיקרו-בקר יכול לשמש כרכיב Master או כרכיב Slave, כאשר הפונקציה מקבלת פרמטר יחידת $I^2C\backslash TWI$ של **המעבד מתפקדת כרכיב Slave והפרמטר שמתקבל קובע את כתובת רכיב ה- Slave**.

מכיוון שהפונקציה משמשת לאתחול היחידה יש להשתמש בה פעם אחת ולכן נכתוב אותה בפונקציה setup.

להלן פורמט הפונקציה:

```
Wire.begin(address);
```

פונקציה Wire.beginTransmission(address)

זוהי פונקציה אשר באמצעותה מתחילים את התקשורת בין המיקרו-בקר עם רכיב ה- Slave. הפרמטר שנשלח לפונקציה משמש ככתובת הרכיב. כשלמעשה הפונקציה יוצרת את פעולת ה- Start ושולחת לרכיב את הכתובת הכוללת רק שבע סיביות. כאשר סיבית ה- LSB של הכתובת הקובעת את פעולת הכתיבה או הקריאה אינה נכללת בפרמטר שנשלח.

להלן פורמט הפונקציה:

```
Wire.beginTransmission(PCF8574Adderess);
```

פונקציה Wire.endTransmission()

באמצעות פונקציה זו עוצרים את התקשורת בין המיקרו-בקר (המשמש כ- Master) ורכיב ה- Slave.

להלן פורמט הפונקציה:

```
Wire.endTransmission();
```

פונקציה Write(Parameters)

באמצעות פונקציה זו כותבים מהמיקרו-בקר אל לרכיב ה- Slave. הפונקציה יכולה לקבל את הפרמטרים הבאים:

Wire.write(value) - פרמטר מסוג byte.

Wire.write(string)– כתובת מחרוזת.

Wire.write(data, length)- כתובת המערך והאורך שמעוניינים לכתוב ממערך.

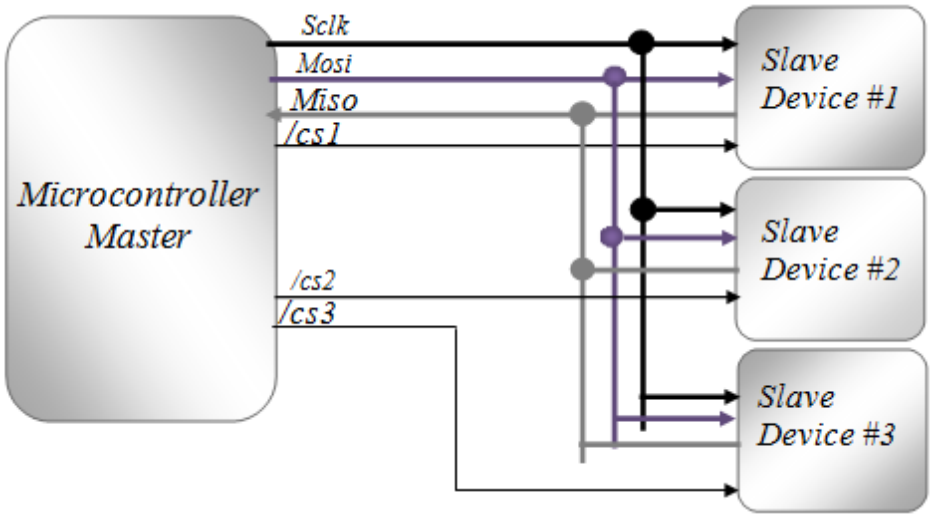
יש לכתוב את פונקציה הזו אחרי פונקציה Wire.beginTransmission(address). בסיום פעולת הכתיבה יש לכתוב את הפונקציה write.endTarnsmission המשמשת לעצירת התקשורת בין הרכיבים.

פרוטוקול SPI (Serial Peripheral Interface)

התקן משתמש בשלושה סוגי אותות חד כיווניים:

- SCLK – Serial Clock (sometimes called clk).
- MOSI – Master Output Slave Input (may be called SI, SDI on slave device).
- MISO – Master Input Slave Output (may be called SO, SDO on slave device).
- CSN – Chip Select (on ore many & usually active low).

האיור הבא מדגים את צורת החיבור של Master אחד לכמה רכיבי Slave:



מצבי עבודה של הפרוטוקול

קיימים ארבעה מצבי עבודה של הפרוטוקול שהשוני בניהם הוא פאזה (מופע) וקוטביות של אות השעון.

מצבי עבודה אלו נקבעים באמצעות שני פרמטרים שנקראים CPHA ו CPOL:

פרמטר CPHA (Clock Phase)

קובע את פאזה (מופע) אות השעון (Clock Phase). כאשר **CPHA='0' הדגימה של אותות מידע תעשה בעליית אות השעון**. כאשר **CPHA='1' הדגימה של אותות מידע תעשה בירידת אות השעון**.

פרמטר CPOL (Clock Polarity)

קובע את הקוטביות של הדק אות השעון (Clock Polarity) **במצב idle (מצב מנוחה). כאשר 'CPOL='0 אות השעון במצב idle יהיה במצב '0' לוגי.**

כאשר **'CPOL='1 אות השעון במצב idle יהיה במצב '1' לוגי.**

טבלה המרכזת את מצבי עבודה בהתאם לפרמטר CPHA ו- CPOL:

SPI MODE	CPOL (Clock Polarity)	CPHA (Clock Phase)
SPI_MODE0	0	0
SPI_MODE1	0	1
SPI_MODE2	1	0
SPI_MODE3	1	1

יחידת SPI (Serial Peripheral Interface) של הבקר.

מיקרו-בקר ATmega328P מכיל יחידת SPI מובנת אפשר הופכת את השימוש בפרוטוקול זה לנוח.

להרחבת הידע הנושא יש לקרוא את **פרק SPI** בספר מיקרו-בקר ממשפחת AVR ליישומים לסביבת ARDUINO חלק ב'.

ישנן שלושה הדקים במיקרו-בקר שאליהם נחבר את כל הרכיבים שמשמשים כ- Slave במערכת.

- הדק SCLK (Serial CLock) שתפקידו לסנכרן את המידע המועבר בין שני רכיבים ממוקם בהדק D13 (B5).
- הדק MISO (Master Input Slave Output) שתפקידו להעביר מידע מרכיב ה- Slave לרכיב ה- Master ממוקם בהדק D12 (B4).
- הדק MOSI (Master Output Slave Input) שתפקידו להעביר מידע מרכיב ה- Master לרכיב ה- Slave ממוקם בהדק D11 (B3).
- הדק SS (Slave Select) שתפקידו לאפשר את הגישה לרכיב ה- Slave ממוקם לרוב בהדק D10 (B2) ניתן לבחור בכל הדק אחר.

טבלה ריכוז הדקים בעבודה בפרוטוקול SPI:

Arduino Board	SCLK	MISO	MOSI	SS(slave)
Uno/Nano	13	12	11	**10**

ספריית SPI שמובנית בסביבת העבודה של ARDUINO

ספריה זו מפעילה את יחידת SPI של הבקר והופכת את השימוש בפרוטוקל זה לנוח משמשת לתקשרות בין רכיבים העובדים בפרוטוקול SPI. הספרייה מכילה פונקציות שכדי להשתמש בהן יש להצהיר על שימוש בקובץ כותרת הבא:

```
#include <SPI.h>
```

פונקציה SPI.begin

זוהי פונקציה אשר באמצעותה מאתחלים את יחידת SPI של המעבד לעבודה בפרוטוקול SPI. בנוסף הפונקציה מאתחלת פונקציה זו מאתחלת את סיביות הפורט האמורות מלבד סיבית SS שצריכה הגדרה ידנית.

להלן פורמט הפונקציה:

```
SPI.begin();
```

פונקציה SPI.transfer

זוהי פונקציה אשר באמצעותה כותבים וקוראים בפרוטוקול SPI פעולת הכתיבה וקריאה מתבצעים בו זמנית. הפונקציה מקבלת פרמטר בגודל 8Bits לכתיבה ומחזירה ערך בגודל 8Bits שנקרא מהרכיב ה-Slave. במקרים שבהם מעוניינים רק לכתוב לרכיב Slave אין צורך לקרוא את הערך המוחזר. במקרים שבהם מעוניינים רק לקרוא מידע מרכיב Slave יש לשלוח כפרמטר את הערך 0 או 0xff.

להלן פורמט הפונקציה:

```
receivedVal=SPI.transfer(data);
```

פונקציה SPI.transfer16

זוהי פונקציה אשר באמצעותה כותבים וקוראים בפרוטוקול SPI פעולת הכתיבה וקריאה מתבצעים בו זמנית. הפונקציה מקבלת פרמטר בגודל 16Bits לכתיבה ומחזירה ערך בגודל 16Bits שנקרא מהרכיב ה-Slave. במקרים שבהם מעוניינים רק לכתוב לרכיב Slave אין צורך לקרוא את הערך המוחזר. במקרים שבהם מעוניינים רק לקרוא מידע מרכיב Slave יש לשלוח כפרמטר את הערך 0 או 0xffff.

להלן פורמט הפונקציה:

```
receivedVal16 = SPI.transfer16(val16);
```

פונקציה SPI.endTransaction

זוהי פונקציה אשר באמצעותה מסיימים את עבודה בפרוטוקול SPI.

להלן פורמט הפונקציה:

```
SPI.endTransaction() ;
```

הגדרת מצב הדק SS.

מכיוון שהדק SS מאפשר את התחלת התקשורת בין הרכיבים בפרוטוקול SPI. לפני פנייה לרכיב ה- Salve יש לאפס או לטעון להדק זה את הערך '1' לוגי בהתאם ל-MODE העבודה של הרכיב איתו או עובדים.

בסיום עבודה עם רכיבי יש לטעון '1' לוגי או לאפס את ערך הדק זה. לרוב לפני תחילת התשדורת מאפסים את ערך סיבית זו ובסיום טוענים לה את הערך '1' לוגי. פעולות אלו יתבצעו באמצעות הפונקציה digitalWrite כאשר יש להגדיר את ההדק כיציאה.

להלן דוגמא לכתיבת הערך 0x55 לרכיב Slave שהדק האפשור שלו מחובר להדק 10.

```
#include <SPI.h>
#define SS 10
void setup() {
  pinMode (SS, OUTPUT);
  SPI.begin();
  digitalWrite(SS,0);
  SPI.transfer(0x55);
  digitalWrite(SS,1);
}//End of setup function
```

פונקציה SPI.setDataMode

זוהי פונקציה אשר באמצעותה קובעים את MODE העבודה של הפרוטוקול על ידי שליחה פרמטר לפונקציה.

טבלה המרכזת את מצבי עבודה בהתאם לפרמטר CPHA ו- CPOL:

SPI MODE	CPOL (Clock Polarity)	CPHA (Clock Phase)
SPI_MODE0	0	0
SPI_MODE1	0	1
SPI_MODE2	1	0
SPI_MODE3	1	1

שימו לב! במידה ולא נגדיר את SPI_MODE ברירת המחדל היא SPI_MODE0.

להלן פורמט הפונקציה:

```
SPI.setDataMode(mode);
```

דוגמא לשימוש בפונקציה לקביעת MODE העבודה ל- MODE1:

```
SPI.setDataMode(SPI_MODE1);
```

פונקציה SPI. setClockDivider

זוהי פונקציה אשר באמצעותה מגדירים את קצב תדר השעון והוא נקבע כחלוקה של התדר שעון החיצוני שמחובר למיקרו-בקר (במקרה שלנו 16MHz) חלקי מספר קבוע. ערך מספר זה יכול להיות: 2, 4, 8, 16, 32, 64 ו-128.

ריכוז תדרי השעון בפרוטוקול SPI בהתאם ליחס החלוקה:

תדר השעון	קבוע
8MHz	SPI_CLOCK_DIV2
4MHz	SPI_CLOCK_DIV4
2MHz	SPI_CLOCK_DIV8
1MHz	SPI_CLOCK_DIV16
500KHz	SPI_CLOCK_DIV32
250KHz	SPI_CLOCK_DIV64
125KHz	SPI_CLOCK_DIV128

שימו לב! במידה ולא נגדיר את קצב תדר השעון ברירת המחדל היא 4MHz.

להלן פורמט הפונקציה:

```
SPI.setClockDivider(divider);
```

דוגמא לשימוש בפונקציה לקביעת קצב תדר השעון ל- 2MHz:

```
SPI.setClockDivider(SPI_CLOCK_DIV8);
```

פונקציה SPI.setBitOrder

זוהי פונקציה אשר באמצעותה מגדירים את כיוון התשדורת של הסיביות בפרוטוקול כאשר קיימות שתי אופציות: אחת שידור המידע יתחיל מסיבית LSB ועד סיבית ה-MSB. שנייה שידור המידע יתחיל מסיבית MSB ועד סיבית ה-LSB.

כדי לקבוע שכיוון התשדורת יהיה מסיבית LSB אל סיבית ה- MSB יש לשלוח את המילה LSTFIRST בעת קריאה לפונקציה. כדי לקבוע שכיוון התשדורת יהיה מסיבית ה- MSB אל סיבית ה- LSB יש לשלוח את המילה MSBFIRST בעת קריאה לפונקציה.

להלן פורמט הפונקציה:

```
SPI.setBitOrder(order);
```

דוגמא לשימוש בפונקציה להגדרה שכיוון שידור המידע יתחיל מסיבית LSB ועד סיבית ה-MSB.

```
SPI.setBitOrder(LSBFIRST);
```

שימו לב! במידה ולא נגדיר את כיוון שידור המידה ברירת המחדל מסיבית MSB ועד סיבית ה-LSB.

פונקציה SPISettings

זוהי פונקציה אשר מקבלת שלושה פרמטרים: פרמטר ראשון קובע את קצב השעון של יחידת SPI,
פרמטק שני קובע את כיוון התשדורת של הסיביות ופרמטר שלישי קובע את MODE העבודה של היחידה
על מנת להשתמש בפונקציה זו יש לכתוב אותה בתוך פונקציה ( )SPI.beginTransaction
להלן פורמט הפונקציה:

```
SPI.beginTransaction(SPISettings(speedMax, dataOrder, dataMode));
```