בית ספר סולם צור (סמל בית ספר)

**חממה הידרופונית חכמה**

פרויקט גמר במגמת הנדסת אלקטרוניקה ומחשבים בהתמחות מערכות אלקטרוניות

בחלופה:

סמל שאלון:

מאת: נוה ששוני

מספר תעודת זהות: 324835123

בהנחיית: אדי רוזנבאום וויקי סיגלר

שנה"ל תשפ"א

**הצהרת הלומד:**

שם התלמיד: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ מספר ת"ז: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

אני הח"מ, מצהיר בזאת כי פרויקט/עבודת הגמר וספר הפרויקט המצ"ב נעשו על ידי בלבד .

הפרויקט מסכם ידע, מיומנות והרגלים שלמדתי במסגרת לימודי ההתמחות במגמה ובאופן עצמאי.

הפרויקט וספר תיעוד הפרויקט נעשו על בסיס ההנחיות שקיבלתי מהמנחה שלי .

מקורות המידע בהם השתמשתי לביצוע פרויקט מצוינים ברשימת המקורות שבסוף הספר.

אני מודע לאחריות שהנני מקבל על עצמי על ידי חתימתי על הצהרה זו שכל הנכתב בה אמת.

חתימת התלמיד: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ תאריך: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**אישור מנחה הפרויקט/עבודת הגמר**

הריני מאשר שהפרויקט בוצע בהנחייתי, קראתי את ספר הפרויקט ומצאתי כי הוא ראוי להגשה.

שם המנחה: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ חתימה: \_\_\_\_\_\_\_\_\_\_\_\_\_ תאריך: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**אישור רכז המגמה**

הריני מאשר שדרישות הפרויקט ורמתו מתאימים לדרישות והנחיות משרד החינוך המפורסמים בחוזר המפמ"ר

ובאתר המגמה.

שם רכז המגמה: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ חתימה \_\_\_\_\_\_\_\_\_\_\_\_\_\_ תאריך: \_\_\_\_\_\_\_\_\_\_\_

# הקדשה/תודה:

אני רוצה להודות ל... דף להבעת שלמי תודה או הקדשה.

# תוכן עניינים

**°C°C°C°C**

# תקציר

הפרויקט שלי הוא מערכת הידרופונית מבוקרת. מערכת הידרופונית היא מערכת חקלאית לגידול צמחים, מערכת הידרופונית היא מערכת חקלאית לגידול צמחים, שלא מוזנת דרך אדמה דשנה, אלא על ידי צנרת שמעבירה את החומרים הדרושים כדי לגדל את הצמח (מים ודשן). המערכת נעזרת במשאבות בכדי להעביר את הנוזלים והחומרים הדרושים ברחבי המערכת, המים זורמים במעגל סגור. המערכת ההידרופונית שבניתי בנויה מצינורות מנוקבים, כל נקב הוא בעצם מקום לצמח. שיטה זו נקראת NFT. המערכת שלי מתייחדת בעובדה שהיא אוטונומית לגמרי, היא יודעת לטפל בעצמה מבחינה של ניקוז ומילוי מחדש של מים מהולים בדשן היא יכולה לפעול גם כשאין חשמל על ידי מצבר והיא מחוברת לאינטרנט לשרת שמקבל מידע לגבי המערכת, הנוזלים שבמערכת, הטמפרטורה, תנאי התאורה, העכירות של המים, המליחות של המים, גובה המים במיכל , החומציות והטמפרטורה במים. המערכת יודעת לתפעל ולבדוק הרבה מהרכיבים בתוכה וגם קיימת אפשרות לשליטה ידנית ומקומית. המערכת מיועדת בעיקר לקהל הביולוגי בחקר הצמחים, היא מהווה תשתית לביצוע של ניסויים שונים בתחום הביולוגיה על צמחים. באמצעות מערכות אוטונומיות וכלי מדידה חקלאים. לבסוף המערכת מאפשרת לבצע בקרה אוטומטית במערכת ההידרופונית לאסוף ולספק מידע על המצב של המערכת מבחינת הניסוי.

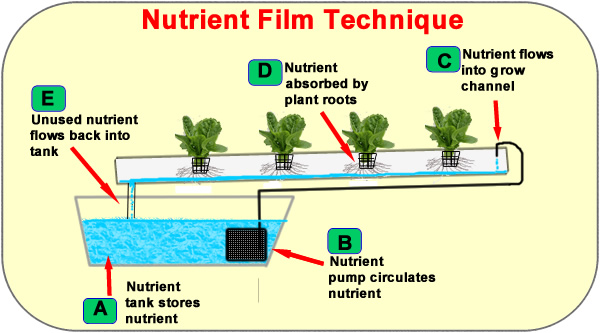
# תיאור הבעיה או הצורך

כיום בתחום חקר הצמחים בביולוגיה בשביל לבצע ניסוי במערכת הידרופונית צריך לבצע בקרה שותפת ולאסוף את המידע מתוך מספר חיישנים בתדירות שיכולה לפעמים להיות מאוד גבוהה. דבר הגורם לעומס על חוקר שצריך לעשות את הבקרה ואת הדגימה בצורה ידנית.

# תפקיד הפרויקט

תפקידו של הפרויקט הוא לנהל את המערכת ההידרופונית באופן כמעט אוטונומי לגמרי. לקחת דגימות ולשלוח את הדגימות למסד נתונים כדי לאפשר מעקב שוטף מכל מקום. לבצע פעולות תחזוקה שונות במערכת ההידרופונית באופן כמעט אוטונומי לגמרי. וגם להתריע על בעיות שיכולות לצוץ בחלק מרכיבי המערכת.

## מבנה הפרויקט

ראשית המערכת הינה מערכת מסוג NFT.

המיכל מכיל מים מהולים בדשן ומשאבה, תפקידה של המשאבה הוא להזרים מים מהמכל אל תחילתו של הצינור. הצינור הוא צינור חלול שבכל חור נמצא כלוב מחורר שבתוכו נמצא צמח ששואב את המים שזורמים דרך הצינור. בסופו של דבר המים מתנקזים בחזרה למיכל ושוב מוחזרים על ידי המשאבה כדי לעשות את אותו הסיבוב.

על מנת לאפשר מערכת ניסוי יעילה עבור תלמידי ביולוגיה הרחבנו את מערכת ה NFT להיות מבוקרת על ידי בקר Arduino. בכך אנחנו מורידים את כמות הזמן לבקרה ידנית ומורידים את הסיכוי לתקלות בזמן ניסוי ארוך טווח. בנוסף ה Arduino מסוגל לאסוף נתונים לאורך זמן ולשלוח אותם לשרת שאוגר אותם למנת ביצוע סטטיסטיקות בהמשך.

## תרשים המלבנים

### הסבר תרשים מלבנים

במערכת יש Arduino אחד מדגם Arduino Mega 2560 והוא מחובר למספר רכיבים:

* מסך מגע גדול שדרכו אפשר לבצע אינטראקציה עם המערכת.
* רכיב ESP8256 שבעזרתו אפשר להתחבר ל – WI-FI ולשלוח ולקבל נתונים דרך המרשתת. תפקידו במערכת הוא לשלוח נתונים שהמערכת אוספת לשרת ייעודי.
* מספר מנעולים אלקטרוניים (Relay) שתפקידם לבקר את הרכיבים שמחוברים למערכת במתח יותר גבוה ממנה, כמו: משאבה, ברז כדורי, סולנואיד.
* משאבה, שתפקידה להזרים מים במערכת.
* ברז כדורי, שתפקידו להזרים מים לתוך המערכת.
* סולנואיד., שתפקידו לנהל את הניקוז של המערכת.
* חיישני תאורה שתפקידם למדוד את עוצמת האור כחלק מהתיעודים שצריך לאסוף בניסוי.
* חיישני טמפרטורת ולחות אוויר שתפקידם למדוד את הטמפרטורה ואת הלחות שבאוויר גם כחלק מהתיעודים הדרושים לניסוי.
* חיישני גובה מים, מודדים את גובה המים בתוך המיכל הראשי של המערכת ומוודאים שגובה המים במערכת מתאים לפעולה תקינה של המשאבה.
* חיישן טמפרטורת מים, מודד את טמפרטורת הנוזלים במערכת גם חלק מהתיעודים הדרושים בניסוי.
* חיישני זרימת מים, מוודאים שמתבצעת זרימה של נוזלים ברחבי המערכת גם כדי לבדוק האם פעולתם של המשאבה והברזים תקינה.
* חיישני מתח למשאבה, מטרתם לבדוק האם עובר מתח בין המנעול האלקטרוני למשאבה כדי לוודא שהמשאבה מקבלת את המתח שהיא אמורה לקבל.
* חיישני מתח סוללה, בודקים את עוצמת המתח בסוללה גם כדי לדעת מהי רמת הטעינה של הסוללה (באחוזים). // לעבור על זה שוב ולהוסיף רכיבים במידת הצורך

## טיוטה לשרטוט חשמלי

Diagram

Description automatically generated

## מפרט טכני ורשימת רכיבים:

|  |  |  |  |
| --- | --- | --- | --- |
| רכיב | מאפיינים | מתח הפעולה | פרוטוקול תקשורת |
| מסך מגע | גודל המסך: 3.5inch  רזולוציה: 480x320  עומק סיביות הצבע: 24 | 3.3v – 5v | SPI |
| IOT ESP8266 | צריכת זרם: 80mA  פרוטוקול wifi: 802.11 b/g/n  טווח תדירויות: 2.4G – 2.5G  פרוטוקלי רשת: IPv4  TCP/UDP/HTTP/FTP  טמפרטורה: -40 - +125 | 3.3v – 5v | פרוטוקול מותאם אישית |
| Arduino Mega | זיכרון Flash: 256KB  זיכרון Ram: 8KB  מספר רגלי GPIO: | 5v | I2C, UART, SPI |
| משאבה טבולה | צריכת זרם: 1.6A  קצב עבודה: 500GPH (גלונים לשעה)  דורשת פיוז של 4A | 12v |  |
| ברז כדורי חשמלי | צריכת זרם: |  |  |
| ברז סולנואיד | צריכת זרם: 0.5A | 12V |  |
| מד גובה מים | צריכת זרם: |  |  |
| מד עכירות | צריכת זרם: 30mA  טמפרטורה: -30 - 80**°C** | 5v | פין אנלוגי |
| חיישן טמפרטורה ולחות (אוויר) DHT11 | צריכת זרם:  ניטרלי – 60uA  בשימוש – 0.3mA  טווח מדידה טמפרטורה: 0°C ~ 50°C  טווח מדידה לחות: 20% - 90%  רזולוציה: 16-bit | 3.5v – 5v | one-wire |
| חיישן טמפרטורה (מים) |  |  |  |
| חיישן עוצמת אור LDR | צריכת זרם:  טווח מדידה: |  | Analog |
| ספק כוח |  |  |  |
| סוללה | הספק: |  |  |
| RTC | צריכת זרם:  טמפרטורה: -40 - +85  סוללה: 3v | 5v | I2C |
| מנעול אלקטרוני (Relay) |  |  |  |

# טיוטה לשרטוט חשמלי סופי

//פריצינג בתהליך

# שרטוט חשמלי סופי

Diagram

Description automatically generated

## הסבר התרשים

#### ארדואינו – Arduino Mega

A screenshot of a computer

Description automatically generated with medium confidence

זהו בקר הארדואינו, הבקר המרכזי במערכת שבו אני מתכנת ומריץ את הקוד. אל הבקר גם מחוברים כל הרכיבים והחיישנים במערכת.

#### רכיב IOT – ESP8266

Graphical user interface, application

Description automatically generated

זהו רכיב ה – IOT לרכיב זה יש יכולת להתחבר לאינטרנט אלחוטי (WI-FI) ולשלוח בקשות http לשרתים. הרכיב מתחבר לפינים rx3,tx3 ופועל על מתח של 3.3v ומתקשר בפרוטוקול UART.

#### מנעול אלקטרוני – Relay

Chart

Description automatically generated

במערכת יש 3 מנעולים אלקטרוניים והם מהווים כמתג שמעביר מתח גבוה לרכיבים שדורשים זאת (משאבה, 2 ברזים וברז כדורי) והוא מחובר לפינים 8,9,10,12.

#### רכיב – RTC

A picture containing graphical user interface

Description automatically generated

רכיב ה – rtc הוא רכיב שתפקידו לקבל ולשמור את הזמן הנוכחי או הזמן שהוזן לו, הוא מהווה כשעון אלקטרוני הרכיב מחובר לפינים sda ו scl והוא צורך מתח של 5v.

#### מסך מגע – TFT Lcd

Shape, rectangle

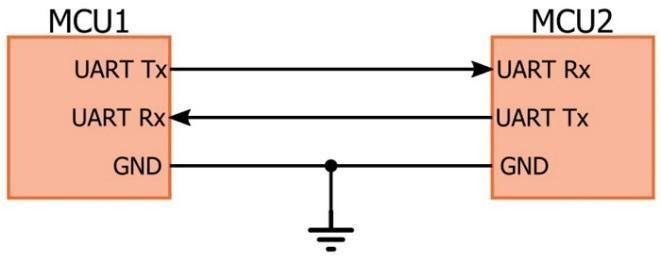
Description automatically generatedזהו מסך מסוג lcd התומך במגע (התנגדותי) מסך זה מחובר לארדואינו בפינים 52,50,51,40,53,43 הוא משמש לשליטה ישירה והעברת מידע באופן מקומי (כמו הגדרות בטלפון) המסך מתקשר עם הארדואינו בפרוטוקול spi.

# פרוטוקולים

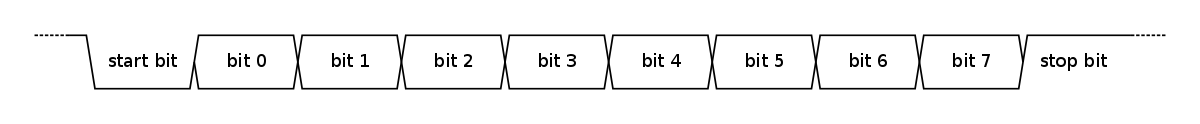
## פרוטוקול UART

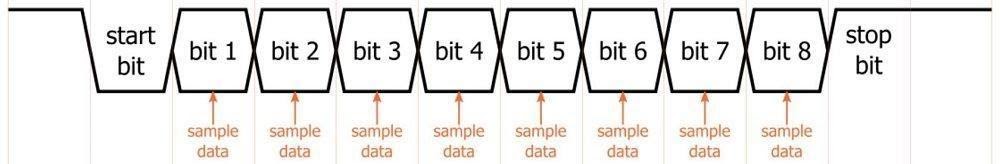
תקשורת UART  היא תקשורת טורית אסינכרונית, בניגוד לתקשורת SPI ו-I2C שעובדות עם שעון. תקשורת אסינכרונית עובדת ללא שעון, אבל יש צורך לתאם קצב העברת ביטים בין הרכיבים. כמו כן רוחב כל סיבית צריך להיות זהה כדי ששני הצדדים יידעו לקרוא נכון את המידע. קצב מקובל בארדואינו הוא 960Bps.

הפרוטוקול מתקשר דרך שני קווים (RX/TX), אחד מקבל מידע ואחד קולט. החיבור הוא נקודה לנקודה, כלומר בין כל שני רכיבים יהיה חיבור ייעודי, אין שיתוף של קווים בין רכיבים. צד אחד שולח דרך קו TX שלו (Transmit), והצד השני יתחבר עם הדק RX  שלו כדי לקבל את המידע. מכיוון שכל חוט הוא חד-כיווני, בפרוטוקול זה מתאפשרת תקשורת דו כיוונית (Full-duplex).



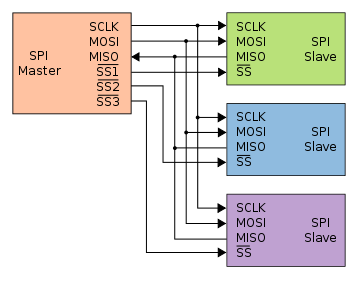
בפרוטוקול זה המידע מועבר ביחידות של בית (8 ביטים), כל פעם בית אחד, ה-LSB  קודם. כל בית מוקף בביט התחלה ובביט סיום. מצב של חוסר תקשורת הו מצב של "1" לוגי. ברגע שהקצב מתואם בין הצדדים, הדגימה של הביט נעשית באמצע הביט, וכך אין אי וודאות לגבי הערך של הביט.





## פרוטוקול SPI

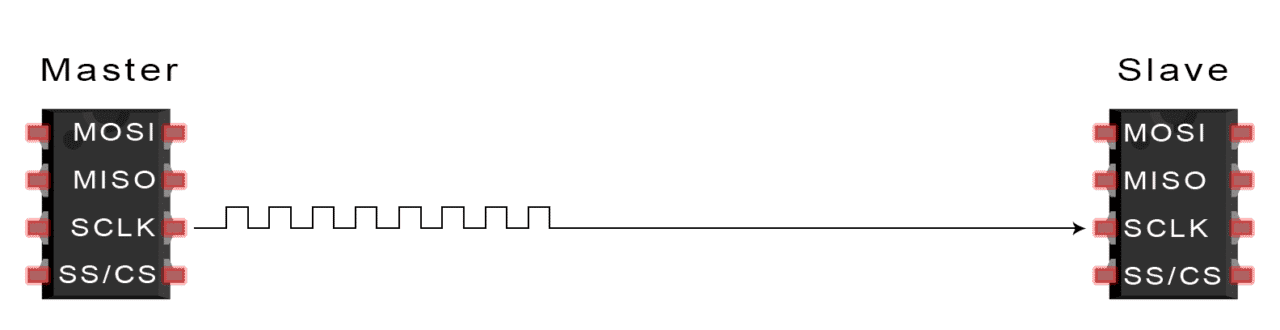
פרוטוקול SPI הינו פרוטוקול סינכרוני – כלומר כל הרכיבים יושבים על קווים משותפים ועובדים על שעון משותף. מכיוון שהוא פרוטוקול סינכרוני, המהירות שלו גבוהה (מהירות –MHz), מה שגורם לקצב העברת מידע גבוה. התקשורת של הפרוטוקול הינה תקשורת טורית, כלומר תקשורת דו כיוונית באותו זמן, ויש לו למעשה 4 קווי תקשורת. בפרוטוקול הזה רכיב אחד יפעל כ-Master וכל השאר כ-Slave. כל הקווים משותפים מלבד הקו של ה-CS שבוחר איזה Slave פעיל.

הפרוטוקול עובד עם 4 פינים: 

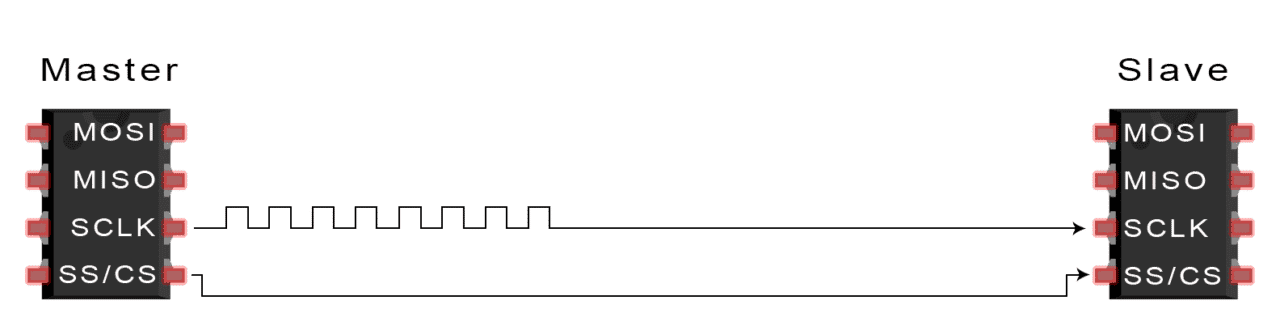
|  |  |  |
| --- | --- | --- |
| **שם הפין** | **כיוון** | **תפקיד** |
| SCLK | Master -> Slave | שעון שמסנכרן את כל היחידות |
| MOSI | Master -> Slave | Master Out Slave In  מעבר מידע מה-Master  ל-Slave |
| MISO | Slave - > Master | Master In Slave Out   מעבר מידע מה-Slave  ל-Master |
| CS או SS | Master -> Slave | Chip select or Slave Select  בחירת ה-Slave  שאיתו מתנהלת התקשורת |

תקשורת SPI

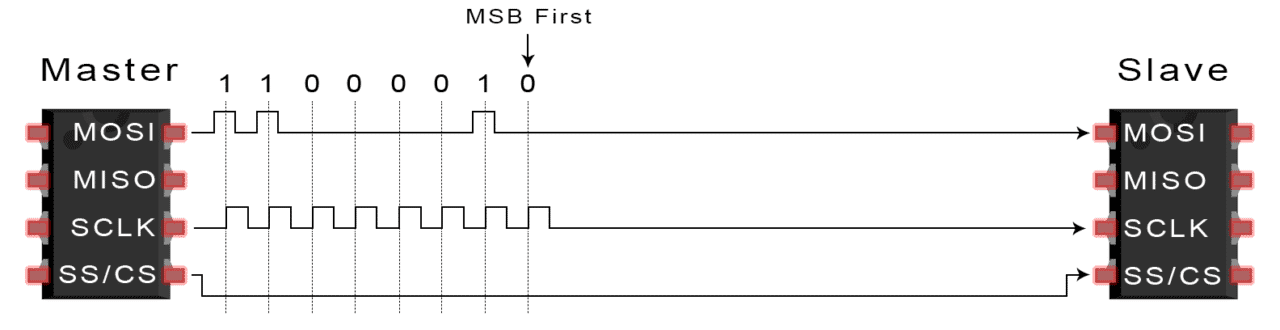
שלב 1 - ה-Master מוציא שעון.

****

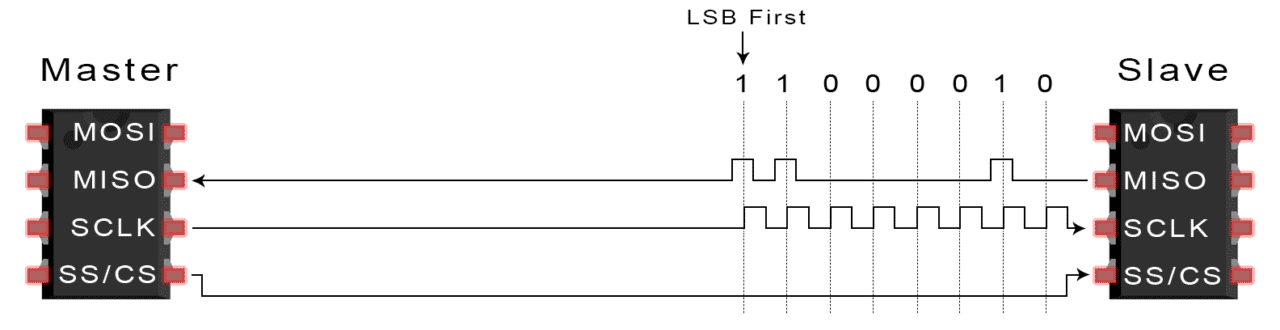
שלב 2 - ה-Master מוריד את הCS ל-Slave  הנבחר. כל עוד ה-CS אפס התקשורת מתקיימת.

****

שלב 3 - ה-Master  שולח את המידע על קו ה-MOSI  עם ה-MSB ראשון בד"כ.  
ה-Slave קורא כל ביט בזמן שנשלח.

****

שלב 4 - אם יש צורך בתשובה ה-Slave שולח את המידע עם הLSB  ראשון בד"כ.  
ה-Master  קורא כל ביט כשהוא מתקבל.

****

## פרוטוקול מותאם אישית לאנטנת WI-FI – ESP8266

בניתי פרוטוקול מותאם אישית לתקשורת בין הארדואינו ל – ESP. הפרוטוקול בנוי מעל פרוטוקול ה – UART והוא נעזר בו כדי להעביר מידע בין הארדואינו לרכיב ה – ESP.

פקודה שנשלחת מ – Arduino ל – esp בנויה כך:

|  |  |  |
| --- | --- | --- |
| parameters | command | start |
| param saperated by $ | one char | ~ |

### להלן הפקודות האפשריות:

#### פקודת התחברות לרשת:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | timeout |  | password |  | ssid | command | start |
| $ |  | $ |  | $ |  | c | ~ |

#### פקודת התנתקות מהרשת:

|  |  |
| --- | --- |
| command | start |
| d | ~ |

#### הפעלת מצב ניפוי באגים (Debug):

|  |  |
| --- | --- |
| command | start |
| b | ~ |

#### כיבוי מצב ניפוי באגים (Debug):

|  |  |
| --- | --- |
| command | start |
| e | ~ |

#### קבלת כתובת ה – IP של הרכיב:

|  |  |
| --- | --- |
| command | start |
| a | ~ |

#### שליחת פקודת get לכתובת url:

|  |  |  |  |
| --- | --- | --- | --- |
|  | url | command | start |
| $ |  | g | ~ |

#### שליחת פקודת post לכתובת url:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | data |  | url | command | start |
| $ |  | $ |  | p | ~ |

#### קבלת פלט מהרכיב:

|  |  |
| --- | --- |
| end | data |
| \0 | g |

## השוואה בין הפרוטוקולים

|  |  |  |  |
| --- | --- | --- | --- |
|  | SPI | I2C | UART |
| שם מלא | Serial Peripheral Interface | Inter-Integrated Circuit | Universal Asynchronous Receiver/Transmitter |
| מס' קווים לתקשורת | 4  SCLK, CS, MOSI, MISO | 2  SCL, SDA | 2  TX, RX |
| סנכרון | סינכרונית | סינכרונית | אסינכרונית |
| קצב העברת המידע (bandwidth). | 10 – 20 Mbps | 1 – 100 kbps | 2 – 460 kbps |
| מס' המסטרים | אחד | אחד או יותר | לא רלוונטי, חיבור נקודה לנקודה |
| סיבוכיות חומרה | בינונית | גבוהה | נמוכה |
| בחירת Slave | סינגל ייחודי (CS) | כתובת | אחד לאחד |
| אימות | לא | כן | לא |
| בדיקת שגיאה | לא | כן | אפשרי |
| דו כיווניות | שני כיוונים בו זמנית  Full duplex | כיוון אחד  Half duplex | Simplex, Half duplex, Full duplex |

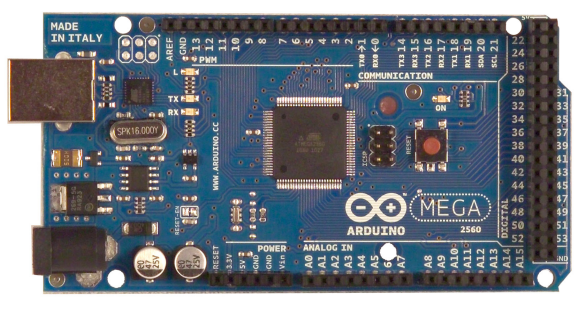
# רכיבים

## מבוא לארדואינו

ארדואינו הנה פלטפורמה/ כרטיס פיתוח המבוסס על מיקרו בקר מסדרת AVR של חברת ATMEL, עם מפתחי קלט/ פלט פשוטים וסביבת פיתוח גמישה בעלת קוד פתוח IDE. פלטפורמה זו מאפשרת שילוב של חומרה ותוכנה והיא קלה לשימוש.

ניתן להשתמש בארדואינו לפיתוח פרויקטים אינטראקטיביים עצמאיים או פרויקטים מקושרים עם תוכנה הפועלת במחשב.

הארדואינו הוא שילוב של שלושה אלמנטים קריטיים: חומרה, תוכנה וקהילה.



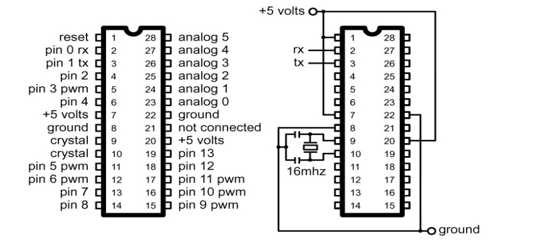
כרטיסי הפיתוח של הארדואינו זמינים במספר צורות ודגמים שונים, בפרויקט אני מתמקד בכרטיסArduino Mega ובלוח פיתוח.

את הכרטיס אנחנו מפעילים על ידי מתח המסופק מיציאת ה USB של המחשב. //לבדוק מה נסגר עם זה באמת.

כרטיס הArduino Mega מבוסס על המיקרו בקר ATmega2560

למיקרו בקר זה יש 54 הדקי קלט/פלט, 40 מהם הדקי קלט/פלט דיגיטליים, כאשר 14 מהם יכולים לשמש כיציאת PWM. בנוסף, כלולים בו עוד 15 הדקים אנלוגיים.

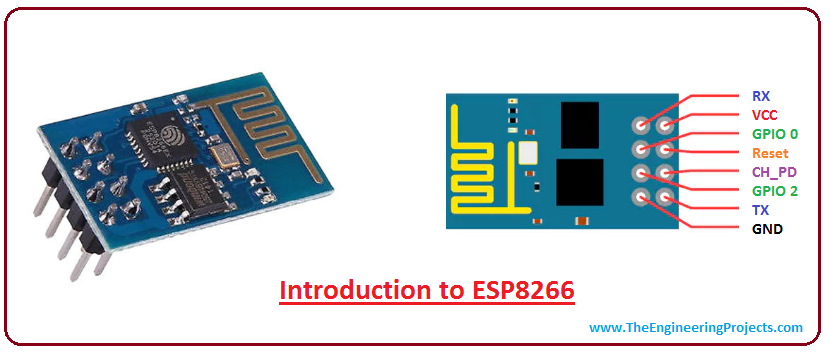
המעבד ATmega2560 תומך בפרוטוקולי תקשורת I2C ו SPI ו-UART.



## רכיב IOT – ESP8266

רכיב הIOT הינו לוח, מבוסס שבב ESP8266. השבב הזה הוא בעל יכולות התחברות לWiFi. הוא יכול לשמש כלקוח, כלומר להתחבר אל מחשבים אחרים, להיות כמו דפדפן, והוא בעצמו יכול להיות גם שרת, כלומר לתת למכשירים אחרים ברשת להתחבר אליו ואז להגיב לפי דרישות המפתחים.

### **צורת חיבור**



|  |  |  |
| --- | --- | --- |
| **רגל** | **חיבור** | **משמעות** |
| RX | TX | המידע שהESP מקבל |
| VCC | 3.3V | רגל אספקת המתח החיובית |
| GPIO 0 | לא מחובר | משמשת לתכנות הרכיב |
| RESET | 3.3V | ריסטרט לרכיב, פועל כאשר המתח נמוך |
| CH\_PD / EN | 3.3V | איפשור ריצת התוכנה על הרכיב |
| GPIO 2 | לא מחובר | משמשת לתכנות הרכיב |
| TX | RX | המידע שהESP שולח |
| GND | GND | רגל האדמה |

לרגליים RX,TX יש המרה מובנת ממתח 5 וולט למתח 3.3 וולט, ולכן ניתן לחבר אותן ישירות לארדואינו.

### אופן השימוש

הרכיב עצמו מכיל מודול חומרתי, שיכול להתחבר לרשתות WIFI שקיימות.

רכיב הIOT מתחבר דרך הWiFi אל השרת בעזרת פרוטוקול תקשורת HTTP(להוסיף פרוטוקול) פרוטוקול זה משמש מערכות אלקטרוניות דומות המבוססות על רשת, כדי להעביר מידע לשרת וגם כדי להחזיר מידע מהשרת.

### ספרית EspSerial

//להזין את הספריה של ה esp

## שסתום סולנואיד

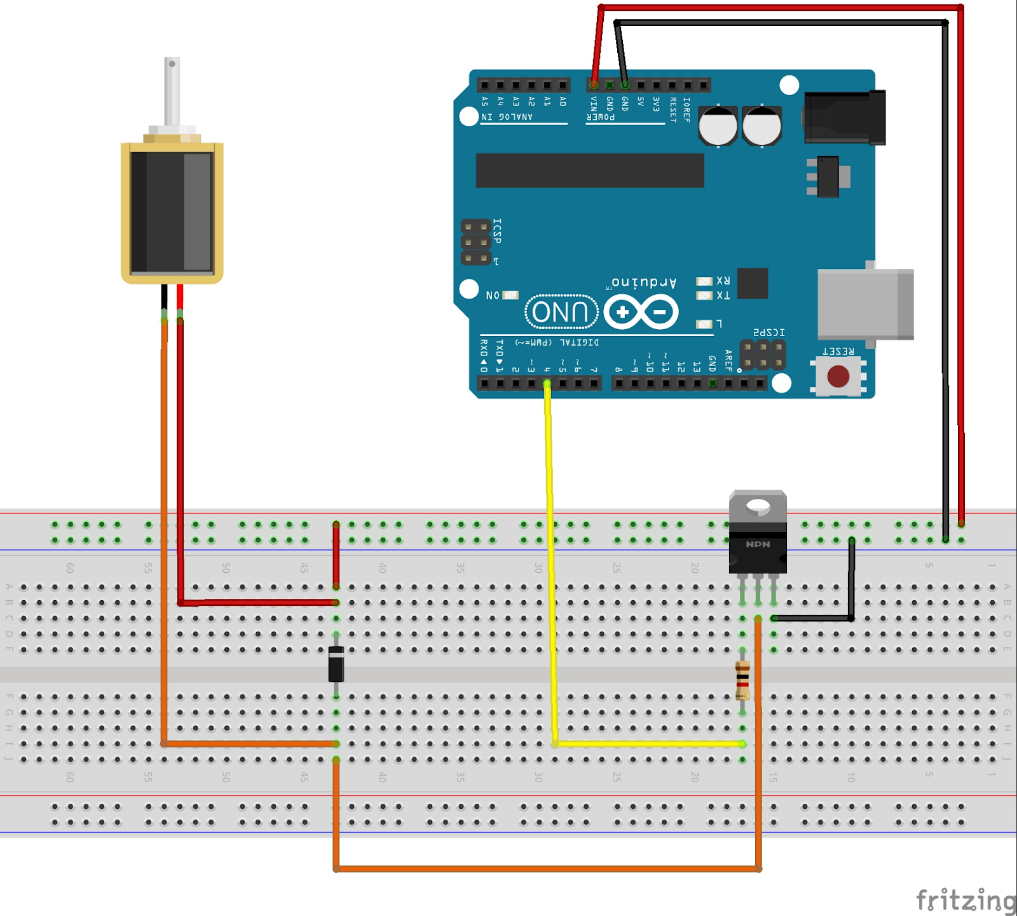
הסולנואיד הוא אלקטרומגנט, ההופך אנרגיה מגנטית לתנועה קוית. הסולנואיד מורכב מסליל מלופף, וכשזה מקבל מתח חשמלי ונוצר זרם, נוצר במרכז הסליל שגה מגנטי. שדה זה מומר לתנועה הפותחת או סוגרת את הברז.

הסולנואיד שלנו הוא סגור בדרך כלל (Normally Closed).

### מפרט טכני

|  |  |
| --- | --- |
| תכונה | מתח פעולה |
| הספק 5w  טמפרטורת סביבה 0 – 40°C  לחות באוויר קטנה מ – 95%  טמפרטורת מים 0 - 55°C  לחץ מים 0.02 – 0.8 Mpa  אורך חיים עד 100,000 פעמים | 12v |

### חיבור

יש לחבר את הסולנואיד עם דיודה השומרת על המעגל מקפיצות במתח. הטרנזיסטור משמש כמגבר זרם. את הסולנואיד מחברים לפין דיגיטלי השולט בפתיחה וסגירה של הברז.

## ברז כדורי CWX-15Q

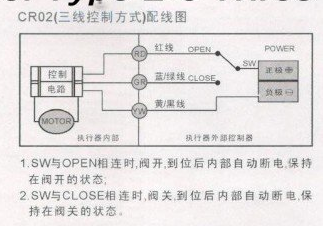
ברז כדורי חשמלי מורכב מברז מכני כדורי שסוגר ופותח את זרם הנוזל ע"י סיבוב הכדור החלול בהתאם לעוצמת הזרם הרצויה. השליטה על הברז היא חשמלית. הברז יכול להעביר נוזלים וגז ללא זליגה.

### מפרט טכני

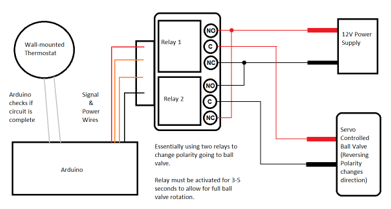
|  |  |
| --- | --- |
| תכונה | מתח פעולה |
| טווח לחצים: 0 – 10bar  טמפרטורת נוזל: -15 - 90 **°C**  הספק: 5w | 9 – 27v DC  85 – 250 AC/DC |

### חיבור

|  |  |  |
| --- | --- | --- |
| חוט | משמעות | חיבור |
| צהוב | אדמה | GND |
| אדום | ברז פתוח | Relay 1 |
| כחול | ברז סגור | Relay2 |



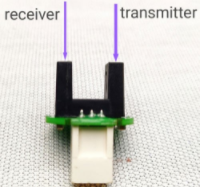
|  |  |  |
| --- | --- | --- |
| אדום | כחול | משמעות |
| 0 | 0 | עצירה |
| 1 | 0 | ברז פתוח |
| 0 | 1 | ברז סגור |
| 1 | 1 | עצירה? |



## חיישן עכירות

חיישן עכירות בודק נדרש כדי לבדוק איכות המים, כלומר עכירותם. עכירות המים נקבעת ע"י כמות החלקיקים הנראים והבלתי נראים בתוך המים. העכירות היא ביחס ישר לכמות החלקיקים הנמצאים בנוזל. כאשר אור פוגע בנוזל עכור, האור מתפזר בגלל כמות החלקיקים, לכן אם נבדוק את פיזור האור, נוכל לגלות את רמת עכירות המים. עכירות נמדדת ביחידות של NTU או JTLJ.

החיישן בנוי ממשדר אור ומקלט, וביניהם נמצא הנוזל. כאשר העכירות גבוהה, נקבל כמות נמוכה של אור במקלט.

חיישן העכירות מחובר דרך רכיב הגבר   driver אשר מגביר את האות של האור המתקבל ומעביר אות אנלוגי לארדואינו היחסי ישר לכמות העכירות במים.

|  |  |  |
| --- | --- | --- |
| תכונה | מתח פעולה | פרוטוקול |
| זרם מקסימלי 30mA  טמפרטורה: -30 - 60°C | 5V | פין אנלוגי |

### מפרט טכני

### חיבור

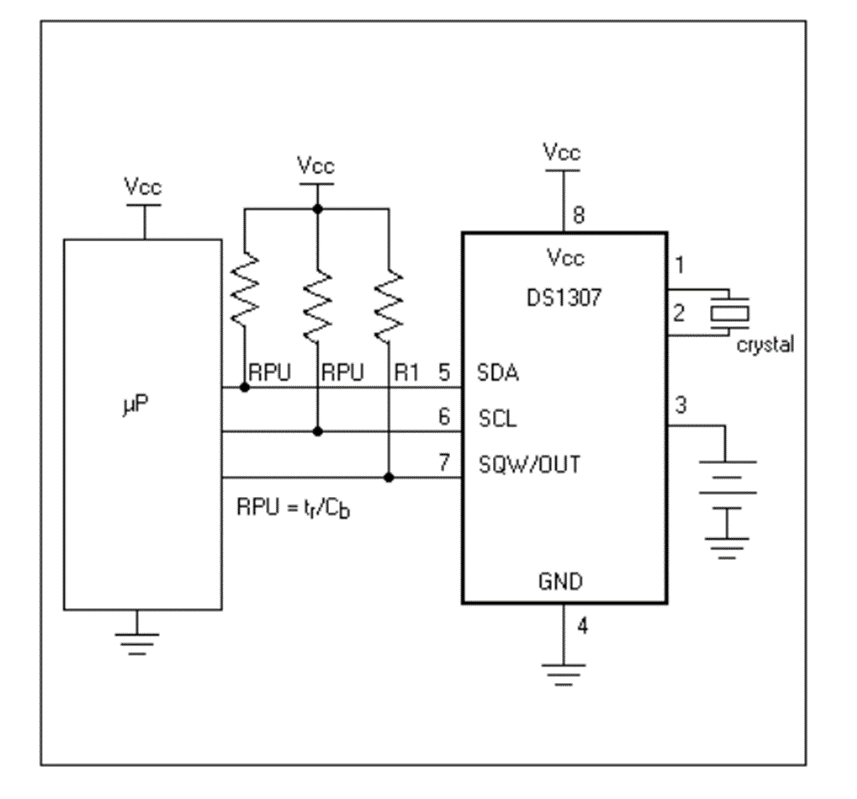
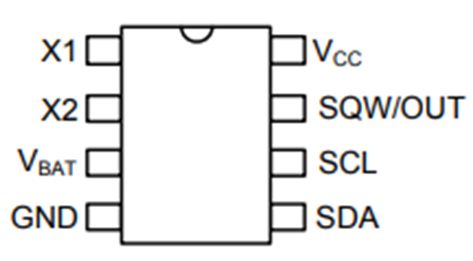
|  |  |  |
| --- | --- | --- |
| רגל | משמעות | חיבור לארדואינו |
| VCC | מתח 5V | VCC |
| GND | אדמה | GND |
| OUT | רמת עכירות | כל פין אנלוגי A0-A6 |

## מעגל השעון -  RTC DS1307

### תאור כללי

רכיב RTC הוא מעגל משולב הבנוי מגביש (חומר קריסטלי) המשמש ליצירת תדר פנימי שבאמצעותו ניתן יהיה לייצר שעון. שעון יודע לבצע ספירה החל מרזולוציה של שנייה ועד ספירה של ימים. ניתן לתכנת את השעון במחזור של 12 שעות או של 24 שעות. אתחול השעון מתבצע פעם אחת בהרצה הראשונית והרכיב מסוגל להמשיך ולהציג תאריך יום ושעה גם כאשר המעגל מנותק ממקור המתח כאשר קיימת סוללת מתח לגיבוי פנימי. קריאת השעון תיעשה על ידי סריקה או באמצעות פסיקה.

הרכיב מתקשר באמצעות פרוטוקול תקשורת I2C. כתובת הרכיב היא  0x68.

|  |  |  |  |
| --- | --- | --- | --- |
| **מספר ההדק** | **שם הדק** | **תיאור ההדק** | **חיבור** |
| 1 | XTAL1 | חיבור של הגביש | גביש |
| 2 | XTAL2 | חיבור של הגביש | גביש |
| 3 | BAT | חיבור סוללת גיבוי | הדק חיובי של סוללה |
| 4 | GND | אדמה | GND |
| 5 | SDA | הדק data של פרוטוקול I2C | A4 ארדואינו |
| 6 | SCL | הדק clock של פרוטוקול I2C | A5 ארדואינו |
| 7 | SQW/OUT | יציאה של גל ריבועי בתדר ניתן לתכנות | לא מחובר |
| 8 | VCC | מתח אספקה של 5V | VCC |

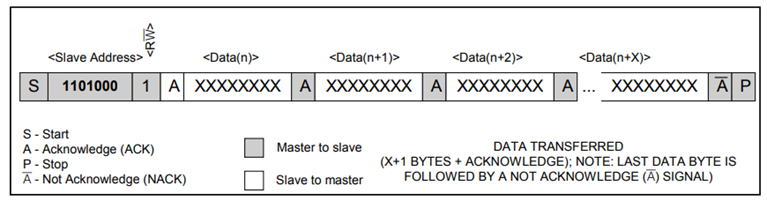
### אופן התקשורת עם הרכיב

הרכיב מכיל אוגרים בכתובות המתחילות ב-0x00. האוגרים מעודכנים כל הזמן עם השעה והתאריך הנוכחי, וניתן לכתוב אליהם, ובכך לאתחל את השעה והתאריך לתאריך ידוע. שלושת האוגרים הראשונים משמשים לשמירת השניות, הדקות והשעה. ביט 7 בכתובת 0x00 מאפשר את הפעלת השעון. האוגר בכתובת 0x03 מכיל את היום בשבוע (בין 1 ל-7), ושלושת האוגרים הבאים מכילים את תאריך. אוגר בכתובת 0x07 הוא אוגר בקרה. בנוסף יש לרכיב זכרון RAM בהמשך הכתובות בגודל 56X8 ביטים.

#### מבנה האוגרים:

#### כתיבה לרכיב מתבצעת בצורה הבאה דרך פרוטוקול I2C:

#### קריאה מהרכיב מתבצעת בצורה הבאה:



### אתחול הזמן

|  |  |
| --- | --- |
| **משמעות** | **פקודה** |
| תחילת תקשורת כתיבה לרכיב הRTC (כתובת 0x68) | Wire.beginTransmission(0x68); |
| המילה הראשונה היא הכתובת אליה רוצים לכתוב – רגיסטר 0 של השניות | Wire.write(0x0); |
| כתיבת השניות, הדקות והשעה אחת אחרי השניה | Wire.write(decToBcd(second)); |
| Wire.write(decToBcd(minute)); |
| Wire.write(decToBcd(hour)); |
| סיום הכתיבה | Wire.endTransmission(); |

אתחול הזמן מתבצע ע"י כתיבה לאוגרים בכתובות מ-0x00-0x06. תחילת התקשורת את כתובת 0x68. השתמשנו בספריית Wire המבצעת כתיבה בפרוטוקול I2C.

### קריאת הזמן

|  |  |
| --- | --- |
| **משמעות** | **פקודה** |
| תחילת תקשורת כתיבה לרכיב הRTC (כתובת 0x68) | Wire.beginTransmission(0x68); |
| המילה הראשונה היא הכתובת אליה רוצים לכתוב – רגיסטר 0 של השניות | Wire.write(0x0); |
| סיום הכתיבה | Wire.endTransmission(); |
| בקשת קריאה מ- Slave לפי הכתובת וכמות הנתונים שרוצים לקרוא בבתים | Wire.requestFrom(0x68,3); |
| אם לא נקראו כל ההודעות מה-slave קוראים כל פעם מילה לתוך משתנה מסוג byte | if(Wire.available()){          second = bcdToDec(Wire.read());          minute = bcdToDec(Wire.read());          hour =bcdToDec(Wire.read()&0b111111);      } |
|  |
|  |

הקריאה מתבצעת ע"י קריאה מהאוגרים מהכתובת הרצויה.

## חיישן טמפרטורה ולחות DHT11

החיישן הינו רכיב המודד טמפרטורת סביבה ולחות בצורה מאוד מדויקת. החיישן מודד לחות באמצעות מדידת ההתנגדות בין שתי אלקטרודות. ההתנגדות נמצאת ביחס הפוך ללחות, ככל שהלחות יותר גבוהה ההתנגדות יותר נמוכה. הטמפרטורה נמדדת ברזולוציה של מעלה, בין 0  לבין 50 מעלות. לצורך מדידת הטמפרטואה משתמשים בטרמיסטור, כלומר נגד המשנה את ההתנגדות בהתאם לטמפרטורה. גם כאן ההתנגדות ביחס הפוך לטמפרטורה. הרכיב משתמש בפרוטוקול ייחודי של חוט אחד, הדורש תזמון מדויק.

### מפרט טכני

|  |  |
| --- | --- |
| **תכונה** | **מתח פעולה** |
| טווח טמפרטורות 0-50C דיוק של 2 מעלות  טווח לחות 20%-80% דיוק של 5%  קצב דגימה 1Hz  זרם מקסימלי 2.5mA | 3-5V |

### חיבור

|  |  |  |
| --- | --- | --- |
| **רגל** | **משמעות** | **חיבור בארדואינו** |
| VCC | מתח אספקה | VCC |
| GND | אדמה | GND |
| DATA | מידע - פרוטוקול קו אחד | פין דיגיטלי |

### ספריה

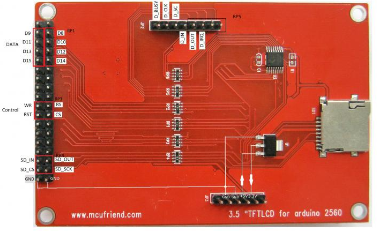
לתקשורת עם הרכיב נשתמש בספריה dht.h.

|  |  |
| --- | --- |
| **פקודה** | **משמעות** |
| **#include <dht.h>** | קריאה לספריה |
| dht DHT; | הגדרת שם לרכיב |
| DHT.read11(dataPin) | קריאה של מידע מהרכיב |
| DHT.temperature | ערך הטמפרטורה (מסוג float) |
| DHT.humidity | ערך הלחות (מסוג float) |

## מסך מגע – ILI9488

### מבוא

בלוח (שם של הלוח של המגה) בו אנו משתמשים מורכב לוח מסך שהוא גם מסך גרפי וגם מסך מגע התנגדותי. המסך הגרפי משמש כ – Output ומציג את מה שהבקר מבקש ממנו להציג. מסך המגע ההתנגדותי הוא רכיב נפרד, והוא משמש כ – Input, כלומר מחזיר לבקר מידע היכן נלחץ המסך.



### מסך גרפי

המסך שנמצא על הלוח ארדואינו הינו מסוג: TFT LCD –  Thin Film Transistor Liquid CrystalCristal Display. כדי לייצג תווים, מספרים, צורות גרפיות ותמונות יש להדליק ולכבות כל נקודה ונקודה בתצוגה (פיקסלים) . איכות התמונה נקבעת ע"י הרזולוציה. הרזולוציה (כושר הפרדה) בתמונה נקבעת ע"י כמות הפיקסלים בציר ה-X וכמות הפיקסלים בציר ה-Y.

המסך שלנו הוא בעל רזולוציה של 480X320 עומק סיביות הצבע של התמונה הוא כמות הסיביות שמייצגות צבעים בתמונה. ככל שהעומק מיוצג ע"י מספר גבוה יותר של סיביות, הדיוק בצבע יהיה יותר גדול. במסך שלנו יש 24 סיביות: 8 סיביות לכל צבע (אדום , ירוק וכחול)

### מסך מגע

Diagram, schematic

Description automatically generatedA picture containing diagram

Description automatically generatedמסך המגע ההתנגדותי הוא  הבסיסי, הזול והנפוץ ביותר. המסך בנוי משתי שכבות מוליכות המופרדות זו מזו עם מרווח ביניהן. שכבות אלו נקראות שכבת ה-X ושכבת ה-Y    
כאשר לוחצים על נקודה במסך השכבה העליונה נלחצת והמרווח בנקודה הזאת מצטמצם והשכבות נפגשות בנקודה זו.  
ישנו ממיר אנלוגי לדיגיטלי שממיר את המתח למיקום X, Y.

### חיבור

|  |  |
| --- | --- |
| מיפוי הפינים | מס׳ הפין |
| SCK | 52 |
| MOSI | 51 |
| MISO | 50 |
| CS מסך גרפי | 40 |
| CS מסך מגע | 53 |
| CS כרטיס SD | 48 |

### פונקציות המסך

עבור המסך השתמשנו בספריה “ili9488.h”, המתרגמת את הפקודות לפרוטוקול SPI, הפרוטוקול שבו הבקר מתקשר עם המסך. להלן הפקודות של הספריה.

//להעביר את התמונה של טבלת הפקודות.

# תיעוד

מתוך יומן הפרוייקט

# מימוש תוכנה גרסה סופית

## פירוט פונקציות

### פונקציות בקוד של הארדואינו

#### פונקציות בקוד של המשאבה:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| לאתחל את המשאבה | void pumpInit() | את הפין שה Relay של המשאבה מחובר אליו | כלום | void pumpInit(){  pinMode(PUMP\_PIN, OUTPUT);  } |
| להפעיל את המשאבה | void pumpStart() | את הפין שה Relay של המשאבה מחובר אליו | כלום | void pumpStart(){  digitalWrite(PUMP\_PIN, HIGH);  \_IsPumpRunning = true;  } |
| לכבות את המשאבה | void pumpStop() | את הפין שה Relay של המשאבה מחובר אליו | כלום | void pumpStop(){  digitalWrite(PUMP\_PIN, LOW);  \_IsPumpRunning = false;  } |
| לבצע בדיקה במשאבה על ידי הפעלה וכיבוי שלה | void pumpTest() | כלום (לבדוק את זה) | כלום | void pumpTest(){  pumpStart();  delay(500);  pumpStop();  delay(500);  } |
| לאפשר מעקף של האוטומציה והפעלה ידנית של המשאבה | void pumpManualControl(PumpManualRequestValue value) | את גובה המים במכל (לבדוק אם משאירים את זה) | כלום (לבדוק את זה) | void pumpManualControl(PumpManualRequestValue value){  switch (value)  {  case PumpManualRequestValue::Start:{  int h = waterHightRead();  if(h > MINIMUM\_WATER\_FOR\_PUMP){  \_PumpManualRequest = value;  pumpStart(); //Pump can run  }  break;  }  case PumpManualRequestValue::Stop:{  \_PumpManualRequest = value;  pumpStop();  break;  }  case PumpManualRequestValue::PumpAutomatic:{  \_PumpManualRequest = value;  break;  }  }  } |
| בודקת האם המשאבה יכולה לעבוד | bool pumpCanRun() | את גובה המים במכל (לבדוק את זה) | True או false של האם המשאבה יכולה לעבוד | bool pumpCanRun(){  int h = waterHightRead();  if(h > MINIMUM\_WATER\_FOR\_PUMP) //If the water levle is not the maximum the fosset will be open.  return true;  else  return false;  } |
| לבדוק מה הפקודה שהמשאבה קיבלה (דולק או כבוי) | pumpIsSignalOn() | את הערך שהגדרתי ושינית כל פעם ששלחתי פקודה למשאבה |  | bool pumpIsSignalOn(){  if(\_IsPumpRunning == 1){  return true;  }  else{  return false;  }  } //להעתיק את זה מחדש |
| לבדוק האם המשאבה אמורה לעבוד | bool pumpSuposeToRun() | את המצב של השליטה הידנית ואת הערך של האם המשאבה אמורה לרוץ | האם המשאבה יכולה לרוץ | //להעתיק |
| לספק את הסטטוס של המשאבה | void pumpCheckStatus() | את הערכים השונים על המצב של המשאבה | סטטוס קצר או ארוך (במידת הצורך של המשאבה) | void pumpCheckStatus(){  bool pumpWaterFlowing = pumpIsWaterFlowing();  bool pumpCurrentOn = pumpIsCurrentOn();  bool pumpSignalOn = pumpIsSignalOn();  bool pumpSuposeToRunVar = pumpSuposeToRun();  int h = waterHightRead();  //Check Signal On  if(\_PumpManualRequest == PumpManualRequestValue::PumpAutomatic){  if(h <= MINIMUM\_WATER\_FOR\_PUMP){  if(pumpSignalOn){  \_PumpStatus.SignalOn = StatusResult("Error, Signal on (automatic) when water levle is too low", STATUS\_ERROR, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_PumpStatus.SignalOn = StatusResult("Warning, Signal off (automatic) Water levle is low", STATUS\_WARNING, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_HIGH);  }  }  else{  if(pumpSignalOn){  \_PumpStatus.SignalOn = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);  \_PumpStatus.Short = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);  }  else{  \_PumpStatus.SignalOn = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);  \_PumpStatus.Short = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);  }  }  }  else if(\_PumpManualRequest == PumpManualRequestValue::PumpStart){  if(h <= MINIMUM\_WATER\_FOR\_PUMP){  if(pumpSignalOn){  \_PumpStatus.SignalOn = StatusResult("Error, Signal on (manual) when water levle is too low", STATUS\_ERROR, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_PumpStatus.SignalOn = StatusResult("Warning, Signal off (manual) Water levle is low", STATUS\_WARNING, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_HIGH);  }  }  else{  if(pumpSignalOn){  \_PumpStatus.SignalOn = StatusResult("On (manual)", STATUS\_OK, PRIORITY\_LOW);  \_PumpStatus.Short = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);  }  else{  \_PumpStatus.SignalOn = StatusResult("Error, Signal off (manual) while manual request is on ", STATUS\_ERROR, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  }  }  else{ //Manual stop  if(pumpSignalOn){  \_PumpStatus.SignalOn = StatusResult("Error, Signal on while manual is off", STATUS\_ERROR, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else {  if(h <= MINIMUM\_WATER\_FOR\_PUMP){  \_PumpStatus.SignalOn = StatusResult("Warning, Signal off (manual) Water levle is low", STATUS\_WARNING, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_HIGH);  }  else {  \_PumpStatus.SignalOn = StatusResult("Off (manual)", STATUS\_OK, PRIORITY\_LOW);  \_PumpStatus.Short = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);  }  }  }  //Check can run  if(h <= MINIMUM\_WATER\_FOR\_PUMP){  \_PumpStatus.CanRun = StatusResult("Error: Not enough water for pump", STATUS\_ERROR, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_PumpStatus.CanRun = StatusResult("Ok", STATUS\_OK, PRIORITY\_LOW);  }  //Check manual request  if(\_PumpManualRequest == PumpManualRequestValue::PumpAutomatic){ //((\_PumpManualRequest == PumpManualRequestValue::PumpAutomatic && pumpCanRun()) || \_PumpManualRequest == PumpManualRequestValue::PumpStart);  if(h <= MINIMUM\_WATER\_FOR\_PUMP){  \_PumpStatus.SuposeToRun = StatusResult("No (automatic), Not enough water", STATUS\_WARNING, PRIORITY\_HIGH);  }  else{  \_PumpStatus.SuposeToRun = StatusResult("Yes (automatic)", STATUS\_OK, PRIORITY\_LOW);  }  }  else if(\_PumpManualRequest == PumpManualRequestValue::PumpStart){  if(h <= MINIMUM\_WATER\_FOR\_PUMP){  \_PumpStatus.SuposeToRun = StatusResult("Error, Yes (manual), Not enough water", STATUS\_ERROR, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_PumpStatus.SuposeToRun = StatusResult("Yes (manual)", STATUS\_OK, PRIORITY\_LOW);  }  }  else{ //Manual stop  if(h <= MINIMUM\_WATER\_FOR\_PUMP){  \_PumpStatus.SuposeToRun = StatusResult("No (manual), Not enough water", STATUS\_WARNING, PRIORITY\_HIGH);  }  else{  \_PumpStatus.SuposeToRun = StatusResult("Yes (manual)", STATUS\_OK, PRIORITY\_LOW);  }  }  //Check current on  if(pumpSignalOn){  if(!pumpCurrentOn){  \_PumpStatus.CurrentOn = StatusResult("Error, No current when signal is on", STATUS\_ERROR, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_PumpStatus.CurrentOn = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);  }  }  else{  if(!pumpCurrentOn){  \_PumpStatus.CurrentOn = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);  }  else{  \_PumpStatus.CurrentOn = StatusResult("Error, Current while signal off", STATUS\_ERROR, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  }  //Check Water Flowing  if(pumpSignalOn){  if(!pumpWaterFlowing){  \_PumpStatus.CurrentOn = StatusResult("Error, Water not flowing while signal is on", STATUS\_ERROR, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_PumpStatus.CurrentOn = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);  }  }  else{  if(!pumpWaterFlowing){  \_PumpStatus.CurrentOn = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);  }  else{  \_PumpStatus.CurrentOn = StatusResult("Error, Water flowing while signal is off", STATUS\_ERROR, PRIORITY\_HIGH);  \_PumpStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  }  } |
| אתחול של חיישן זרם של המשאבה | void pumpCurrentInit() | את הפין שחיישן זרם של המשאבה מחובר אליו | כלום | void pumpCurrentInit(){  pinMode(PUMP\_CURRENT\_TEST\_PIN, INPUT);  } |
| לבצע בדיקה בחיישן זרם של המשאבה | void pumpCurrentTest() | את הערך שחיישן זרם של המשאבה קרא | מדפיסה את הזרם שחיישן הזרם קרא | void pumpCurrentTest(){  Serial.print("Pump current:");  Serial.println(pumpCurrentRead());  } |
| בודקת האם המשאבה עובדת | bool pumpIsRunning() | את הקריאה מהפין האנלוגי שהחיישן מחובר אליו | True או false של האם המשאבה עובדת או לא. | bool pumpIsRunning(){  int c = analogRead(PUMP\_CURRENT\_TEST\_PIN);  return c > 0; //Cheeks if the current is higer than 0. If its higer than 0 its true if not false.  } |

#### פונקציות בקוד של הפוסט הנורמלי (nfosset):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| לאתחל את הפוסט | void nfossetInit() | את הפין שאליו ה Relay של המשאבה מחובר | כלום | void nfossetInit(){  pinMode(NFOSSET\_PIN, OUTPUT);  } |
| לפתוח את הפוסט הנורמלי | void nfossetOpen() | את הפין שאליו ה Relay של המשאבה מחובר | כלום | void nfossetOpen(){  digitalWrite(NFOSSET\_PIN, HIGH);  } |
| לסגור את הפוסט הנורמלי | void nfossetClose() | את הפין שאליו ה Relay של המשאבה מחובר | כלום | void nfossetClose(){  digitalWrite(NFOSSET\_PIN, LOW);  } |
| לבצע בדיקה בפוסט הנורמלי על ידי הפעלה וכיבוי שלו | void nfossetTest() | כלום | כלום | void nfossetTest(){  nfossetOpen();  delay(500);  nfossetClose();  delay(500);  } |
| לאפשר שליטה ידנית על הפוסט הנורמלי | void nfossetManualControl(FossetManualRequestValue value) | את הערך שהוחזר מהפונקציה שבודקת האם הפוסט הנורמלי יכול לעבוד | כלום (לבדוק את זה) | void nfossetManualControl(FossetManualRequestValue value){  switch (value)  {  case FossetManualRequestValue::Close:{  \_NFossetManualRequest = value;  nfossetClose();  break;  }  case FossetManualRequestValue::Open:{  if(nfossetCanRun()){ //If the water levle is not the maximum the nfosset will be open.  \_NFossetManualRequest = value;  nfossetOpen();  }  break;  }  case FossetManualRequestValue::FossetAutomatic:{  \_NFossetManualRequest = value;  break;  }  }  } |
| לבדוק האם הפוסט הנורמלי יכול לעבוד | bool nfossetCanRun() | את גובה המים במכל | True או false של האם הפוסט הנורמלי יכול לעבוד | bool nfossetCanRun(){  int h = waterHightRead();  if(h < MAXIMUM\_WATER\_IN\_TANK) //If the water levle is not the maximum the fosset will be open.  return true;  else  return false;  } |
| לספק סטטוס של הברז הרגיל | void nfossetCheckStatus() | את הערכים של כל כלי הבדיקה שמחוברים לברז הנורמלי | סטטוס קצר או ארוך של הברז הנורמלי (לפי בחירה) | void nfossetCheckStatus(){  bool nfossetWaterFlowing = nfossetIsWaterFlowing();  bool nfossetCurrentOn = nfossetIsCurrentOn();  bool nfossetSignalOn = nfossetIsSignalOn();  bool nfossetSuposeToRunVar = nfossetSuposeToRun();  int h = waterHightRead();  if(h >= MAXIMUM\_WATER\_IN\_TANK){ //If the water levle is the maximum levle for the tank, the fosset will be closed.  if(nfossetSignalOn){  \_NfossetStatus.SignalOn = StatusResult("Error: Fosset signal pin on while water levle too high", STATUS\_ERROR, PRIORITY\_HIGH);  \_NfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_NfossetStatus.SignalOn = StatusResult("Off: Fosset signal off", STATUS\_OK, PRIORITY\_LOW);  \_NfossetStatus.Short = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);  }  }  else{  if(nfossetSuposeToRun){  if(!nfossetSignalOn){  if(h > MINIMUM\_WATER\_FOR\_PUMP){  \_NfossetStatus.SignalOn = StatusResult("Warning: Fosset signal pin off while water levle is low", STATUS\_WARNING, PRIORITY\_MEDIUM);  \_NfossetStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_MEDIUM);  }  else{  \_NfossetStatus.SignalOn = StatusResult("Error: Fosset signal pin off while water levle is under pump minimum", STATUS\_ERROR, PRIORITY\_HIGH);  \_NfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  }  else{  \_NfossetStatus.SignalOn = StatusResult("On: Fosset signal pin on", STATUS\_OK, PRIORITY\_LOW);  \_NfossetStatus.Short = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);  }  }  }      if(nfossetSignalOn){  if(!nfossetWaterFlowing){  if(h > MINIMUM\_WATER\_FOR\_PUMP){  \_NfossetStatus.WaterFlowing = StatusResult("Warning: Water not flowing while signal pin is on", STATUS\_WARNING, PRIORITY\_MEDIUM);  \_NfossetStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_MEDIUM);  }  else{  \_NfossetStatus.WaterFlowing = StatusResult("Error: Water not flowing while water levle is under pump minimum", STATUS\_ERROR, PRIORITY\_HIGH);  \_NfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  }  else{  \_NfossetStatus.WaterFlowing = StatusResult("On: Water flowing", STATUS\_OK, PRIORITY\_LOW);  if(\_NfossetStatus.Short.Status == STATUS\_OK){  \_NfossetStatus.Short = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);  }  }  if(!nfossetCurrentOn){  if(h > MINIMUM\_WATER\_FOR\_PUMP){  \_NfossetStatus.CurrentOn = StatusResult("Warning: No current while signal pin is on", STATUS\_WARNING, PRIORITY\_MEDIUM);  if(\_NfossetStatus.Short.Status != STATUS\_ERROR){  \_NfossetStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_MEDIUM);  }  }  else{  \_NfossetStatus.CurrentOn = StatusResult("Error: No current while water levle is under pump minimum", STATUS\_ERROR, PRIORITY\_HIGH);  \_NfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  }  else{  \_NfossetStatus.CurrentOn = StatusResult("Current on", STATUS\_OK, PRIORITY\_LOW);  if(\_NfossetStatus.Short.Status == STATUS\_OK){  \_NfossetStatus.Short = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);  }  }  }  else{  if(nfossetWaterFlowing){  \_NfossetStatus.WaterFlowing = StatusResult("Error: Water flowing while signal pin is off", STATUS\_ERROR, PRIORITY\_HIGH);  \_NfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_NfossetStatus.WaterFlowing = StatusResult("No water flowing", STATUS\_OK, PRIORITY\_LOW);  if(\_NfossetStatus.Short.Status == STATUS\_OK){  \_NfossetStatus.Short = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);  }  }  if(nfossetCurrentOn){  \_NfossetStatus.CurrentOn = StatusResult("Error: Fosset has current while signal pin is off", STATUS\_ERROR, PRIORITY\_HIGH);  \_NfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_NfossetStatus.CurrentOn = StatusResult("No current", STATUS\_OK, PRIORITY\_LOW);  if(\_NfossetStatus.Short.Status == STATUS\_OK){  \_NfossetStatus.Short = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);  }  }  }  } |
| אתחול של חיישן זרם של הפוסט הנורמלי | void nfossetCurrentInit() | הפין שאליו מחובר חיישן הזרם של הפוסט הנורמלי | כלום | void nfossetCurrentInit(){  pinMode(NFOSSET\_CURRENT\_TEST\_PIN, INPUT);  } |
| לבצע בדיקה בחיישן הזרם של הפוסט הנורמלי | void nfossetCurrentTest() | את הקריאה של חיישן הזרם של הפוסט הנורמלי | מדפיסה את הפלט של החיישן במסך הסיראלי (או כלום) | void nfossetCurrentTest(){  Serial.print("nfosset current:");  Serial.println(nfossetCurrentRead());  } |
| לבדוק האם הפוסט הנורמלי עובד או לא | bool nfossetIsRunning() | את הפין של חיישן הזרם של הפוסט הנורמלי | True או false של האם הפוסט הנורמלי עובד | bool nfossetIsRunning(){  int c = analogRead(NFOSSET\_CURRENT\_TEST\_PIN);  return c > 0; //Cheeks if the current is higer than 0. If its higer than 0 its true if not false.  } |

#### פוסט אוסמוזי

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| אתחול של הפוסט האוסמוזי | void ofossetInit() | את הפין של ה Relay שהפוסט האוסמוזי מחובר אליו | כלום | void ofossetInit(){  pinMode(OFOSSET\_PIN, OUTPUT);  } |
| פתיחה של הפוסט האוסמוזי | void ofossetOpen() | את הפין של ה Relay שהפוסט האוסמוזי מחובר אליו | כלום | void ofossetOpen(){  digitalWrite(OFOSSET\_PIN, HIGH);  } |
| סגירה של הפוסט האוסמוזי | void ofossetClose() | את הפין של ה Relay שהפוסט האוסמוזי מחובר אליו | כלום | void ofossetClose(){  digitalWrite(OFOSSET\_PIN, LOW);  } |
| לבצע בדיקה בפוסט האוסמוזי על ידי הפעלה וכיבוי שלו | void ofossetTest() | כלום | כלום | void ofossetTest(){  ofossetOpen();  delay(500);  ofossetClose();  delay(500);  } |
| לאפשר שליטה ידנית על הפוסט האוסמוזי | void ofossetManualControl(FossetManualRequestValue value) | את הערך של האם הפוסט האוסמוזי יכול לעבוד | כלום (לבדוק את זה) | void ofossetManualControl(FossetManualRequestValue value){  switch (value)  {  case FossetManualRequestValue::Close:{  \_OFossetManualRequest = value;  ofossetClose();  break;  }  case FossetManualRequestValue::Open:{  if(ofossetCanRun()){ //If the water levle is not the maximum the fosset will be open.  \_OFossetManualRequest = value;  ofossetOpen();  }  break;  }  case FossetManualRequestValue::FossetAutomatic:{  \_OFossetManualRequest = value;  break;  }  }  } |
| לבדוק האם הפוסט האוסמוזי יכול לעבוד | bool ofossetCanRun() | את גובה המים במכל | True או false של האם הפוסט האוסמוזי יכול לעבוד | bool ofossetCanRun(){  int h = waterHightRead();  if(h < MAXIMUM\_WATER\_IN\_TANK) //If the water levle is not the maximum the fosset will be open.  return true;    else  return false;  } |
| להוציא סטטוס קצר של הפוסט האוסמוזי | StatusResult ofossetStatus\_Short() | האם יש זרימה של מים דרך הפוסט האוסמוזי והאם הפוסט האוסמוזי יכול לפעול | סטטוס קצר של הפוסט האוסמוזי (עובד, לא עובד ותקלה) | StatusResult ofossetStatus\_Short(){  bool ofossetFlowing = ofossetIsWaterFlowing();// osmotic fosset  bool ofossetCanRunVar = ofossetCanRun();    switch (\_OFossetManualRequest) {  case FossetManualRequestValue::FossetAutomatic:  {  if(ofossetFlowing == true)  {  if(ofossetCanRunVar == true)  return StatusResult ("On", STATUS\_OK);  else  return StatusResult ("Error", STATUS\_ERROR);  }  else  {  return StatusResult ("Off", STATUS\_OK); //a  }  break;  }  case FossetManualRequestValue::Open:  {  if(ofossetFlowing == true)  return StatusResult ("On", STATUS\_OK);  else  return StatusResult ("Error", STATUS\_ERROR);  break;  }  case FossetManualRequestValue::Close:  if(ofossetFlowing == false)  return StatusResult ("Off", STATUS\_OK);  else  return StatusResult ("Error", STATUS\_ERROR);  break;  default:  return StatusResult ("Unknown", STATUS\_WARNING);    }  } |

#### פונקציות של המסך מגע

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| אתחול של מסך ה – LCD | void lcdInit() | כלום (לא בטוח) | כלום | void lcdInit(){  lcd.Init\_LCD();  lcd.Fill\_Screen(BLACK);  } |
| בדיקה של מסך ה – LCD | void lcdTest() | כלום | כלום | void lcdTest(){  lcdShowHomeScreen();  lcdShowHomeScreenStatus();  } |
| להציג את מסך הבית של המערכת על גבי המסך | void lcdShowHomeScreen() | את המסך שאותו ה – LCD מציג לפני שהוא מקבל את הפקודה | מציגה את מסך הבית על גבי המסך (כלום) | void lcdShowHomeScreen(){    if (\_Lcd\_Status == LCD\_PAGE\_HOME){  return;  }  \_Lcd\_Status = LCD\_PAGE\_HOME;  lcd.setFont(4);  lcd.gotoxy(30,4); //take this text to the center  lcd.print("Home Screen");  lcd.drawHLine(25,35,265);  lcd.setFont(2);  lcd.gotoxy(5,40);  lcd.print("Status");  lcd.drawRect(5,60,150,180); //Status show area  lcd.setFont(2);  lcd.gotoxy(10,65);  lcd.print("NFosset:");  lcd.gotoxy(10,84);  lcd.print("OFosset:");  lcd.gotoxy(10,103);  lcd.print("Pump:");  lcd.gotoxy(10,122);  lcd.print("Drainage:");  lcd.gotoxy(10,141);  lcd.print("Battery:");  lcd.gotoxy(10,160);  lcd.print("Water:");  } |
| לעדכן את הסטטוס על גבי מסך הבית ב – LCD | void lcdShowHomeScreenStatus() | מקבל את הסטטוס הקצר מכל אחד מהרכיבים | הסטטוס הקצר במיקום שהוא צריך להיות על גבי מסך הבית | void lcdShowHomeScreenStatus(){  lcdWriteStatus(100, 65, nfossetStatus\_Short());  lcdWriteStatus(100, 84, ofossetStatus\_Short());  lcdWriteStatus(100, 103, pumpStatus\_Short());  lcdWriteStatus(100, 122, drainageStatus\_Short());  lcdWriteStatus(100, 141, batteryStatus\_Short());  } |
| תבנית לכתיבת סטטוס קצר על גבי מסך הבית | void lcdWriteStatus(int x, int y, StatusResult res) | מקבל את המיקום שבו הסטטוס צריך להיות X,Y ואת הסטטוס עצמו | את הסטטוס ביחד עם המיקום שהוא צריך להופיע במסך עם הצבע לפי המצב | void lcdWriteStatus(int x, int y, StatusResult res)  {  lcdChangeColor(res.Priorty);  print(x, y, res.StatusText);  } |
| לשנות את צבע הסטטוס לפי רמת העדיפות שלו/מצב | void lcdChangeColor(int priorty) | את רמת העדיפות של הסטטוס | את הצבע שצריך להופיע על אותו סטטוס | void lcdChangeColor(int riority)  {  switch (riority)  {  case STATUS\_OK:  lcd.setColor(STATUS\_OK\_COLOR);  break;  case STATUS\_WARNING:  lcd.setColor(STATUS\_WARNING\_COLOR);  break;  case STATUS\_ERROR:  lcd.setColor(STATUS\_ERROR\_COLOR);  break;  default:  lcd.setColor(STATUS\_NORMAL\_COLOR);  break;  }  } |

#### הקוד של מד העכירות

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| אתחול של מד העכירות | void turbidityInit() | את הפין שמד העכירות מחובר אליו | כלום | void turbidityInit(){  pinMode(TURBIDITY\_PIN, INPUT);  } |
| בדיקה של מד העכירות | void turbidityTest() | את הערך שמד העכירות מחזיר | מדפיס את הערך במסך הסיראלי | void turbidityTest(){  int value = turbidityReadPercent();  Serial.println(value);  } |
| המרת הערך שחוזר ממד העכירות לאחוזים | int turbidityReadPercent() | את הערך הדיגיטלי הישיר שחוזר ממד העכירות | את העכירות באחוזים | int turbidityReadPercent(){  int val = analogRead(TURBIDITY\_PIN); //Return status in %  return map(val, 0, 1024, 0, 100);  // return val \* (5.0 / 1024.0);  } |
| להוציא סטטוס קצר של מד העכירות (עובד + ערך, לא עובד ותקלה) | String turbidityStatus\_Short() | האם מד העכירות עובד או לא, הערך באחוזים של העכירות | את הסטטוס של מד העכירות (עובד + ערך, לא עובד ותקלה) | String turbidityStatus\_Short(){ //TURBIDITY STATUS FOR THE HOME SCREEN (AR&AP)  bool IsWorking = tubidityIsWorking(); //change it to the minimum and the maximum values, with % .  int turbidityValue = turbidityReadPercent();    if(IsWorking == true)  return String(turbidityValue) + "%";  else  return "Error";  } |
| להוציא סטטוס ארוך של מד העכירות | String turbidityStatusLong() | חיובי או שלילי לגביי האם מד העכירות עובד ואת הערך הסופי שחוזר ממד העכירות | את הסטטוס הארוך של מד העכירות + פירוט רחב יותר של השגיאה במידה ויש. | String turbidityStatusLong(){ //The shortest long status for now...  bool isWorking = tubidityIsWorking();  int turbidityValue = turbidityReadPercent();    if(isWorking == true)  return "Active" + turbidityValue;  else  return "Error, Cheek the sensor";  } |

#### פונקציות בקוד של ה – esp (שנצרב ב – Arduino)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| לאתחל את ה – esp | void espInit() | כלום | כלום | void espInit(){  Serial3.begin(9600);  } |
| לבצע בדיקה לרכיב ה – esp | void espTest() | כלום (לבדוק את זה) | את הפלט שה – esp מחזיר לאחר ביצוע הבדיקה |  |
|  |  |  |  |  |

# רפלקציה על התהליך ועל התוצר

//בתהליך