

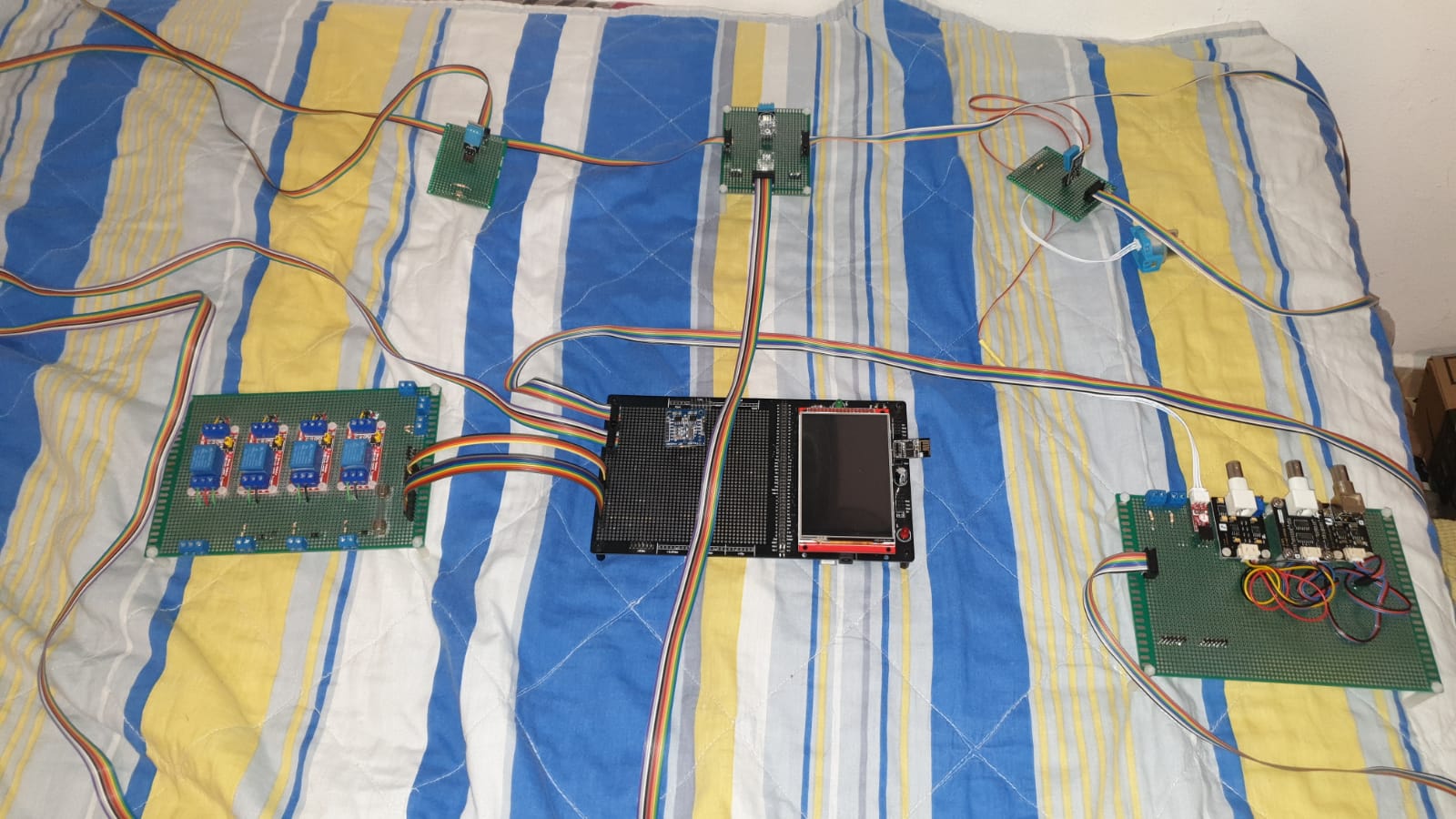
מגמת הנדסת אלקטרוניקה ומחשבים

סמל ביה"ס: 260331

כתובת: גשר הזיו 22815

טלפון: 04-9825272

**חממה הידרופונית חכמה**



פרויקט גמר במגמת הנדסת אלקטרוניקה ומחשבים

בהתמחות מערכות אלקטרוניות

סמל שאלון: 841589

מאת: נוה ששוני

מספר תעודת זהות: 324835123

בהנחיית: אדי רוזנבאום וויקי סיגלר

שנה"ל תשפ"א

**הצהרת הלומד:**

שם התלמיד: נוה ששוני מספר ת"ז: 324835123

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.

**°C°C°C°C**

תוכן עניינים

[תקציר 7](#_Toc74206490)

[תיאור הבעיה או הצורך 7](#_Toc74206491)

[תפקיד הפרויקט 7](#_Toc74206492)

[מבנה הפרויקט 8](#_Toc74206493)

[תרשים המלבנים 9](#_Toc74206494)

[הסבר תרשים מלבנים 9](#_Toc74206495)

[טיוטה לשרטוט חשמלי 10](#_Toc74206496)

[מפרט טכני ורשימת רכיבים 11](#_Toc74206497)

[שרטוט חשמלי סופי 12](#_Toc74206498)

[הסבר התרשים 13](#_Toc74206499)

[ארדואינו – Arduino Mega 13](#_Toc74206500)

[רכיב IOT – ESP8266 13](#_Toc74206501)

[מנעול אלקטרוני – Relay 13](#_Toc74206502)

[רכיב – RTC 14](#_Toc74206503)

[מסך מגע – TFT Lcd 14](#_Toc74206504)

[ברז חשמלי – סולנואיד 14](#_Toc74206505)

[מצבר 12v (סוללת גיבוי למערכת) 15](#_Toc74206506)

[חיישן טמפרטורה ולחות – DHT11 15](#_Toc74206507)

[חיישן עכירות מים 15](#_Toc74206508)

[חיישן תאורה 16](#_Toc74206509)

[פרוטוקולים 17](#_Toc74206510)

[פרוטוקול UART 17](#_Toc74206511)

[פרוטוקול SPI 18](#_Toc74206512)

[פרוטוקול I2C 20](#_Toc74206513)

[תאור כללי 20](#_Toc74206514)

[אופן העברת מידע 20](#_Toc74206515)

[כתיבה מה-Master ל-Slave 21](#_Toc74206516)

[פרוטוקול מותאם אישית לאנטנת WI-FI – ESP8266 22](#_Toc74206517)

[הפקודות האפשריות 22](#_Toc74206518)

[פקודת התחברות לרשת: 22](#_Toc74206519)

[פקודת התנתקות מהרשת: 22](#_Toc74206520)

[הפעלת מצב ניפוי באגים (Debug): 22](#_Toc74206521)

[כיבוי מצב ניפוי באגים (Debug): 22](#_Toc74206522)

[קבלת כתובת ה – IP של הרכיב: 22](#_Toc74206523)

[שליחת פקודת get לכתובת url: 22](#_Toc74206524)

[שליחת פקודת post לכתובת url: 22](#_Toc74206525)

[קבלת פלט מהרכיב: 22](#_Toc74206526)

[השוואה בין הפרוטוקולים 23](#_Toc74206527)

[רכיבים 24](#_Toc74206528)

[מבוא לארדואינו 24](#_Toc74206529)

[רכיב IOT – ESP8266 25](#_Toc74206530)

[צורת חיבור 25](#_Toc74206531)

[אופן השימוש 25](#_Toc74206532)

[שסתום סולנואיד 26](#_Toc74206533)

[חיבור 26](#_Toc74206534)

[ברז כדורי CWX-15Q 27](#_Toc74206535)

[חיבור 27](#_Toc74206536)

[חיישן עכירות 28](#_Toc74206537)

[חיבור 28](#_Toc74206538)

[מעגל השעון -  RTC DS1307 29](#_Toc74206539)

[תאור כללי 29](#_Toc74206540)

[אופן התקשורת עם הרכיב 30](#_Toc74206541)

[מבנה האוגרים: 30](#_Toc74206542)

[כתיבה לרכיב מתבצעת בצורה הבאה דרך פרוטוקול I2C: 30](#_Toc74206543)

[קריאה מהרכיב מתבצעת בצורה הבאה: 30](#_Toc74206544)

[אתחול הזמן 31](#_Toc74206545)

[קריאת הזמן 31](#_Toc74206546)

[חיישן טמפרטורה ולחות DHT11 32](#_Toc74206547)

[חיבור 32](#_Toc74206548)

[ספריה 32](#_Toc74206549)

[מסך מגע – ILI9488 33](#_Toc74206550)

[מבוא 33](#_Toc74206551)

[מסך גרפי 33](#_Toc74206552)

[מסך מגע  33](#_Toc74206553)

[חיבור 34](#_Toc74206554)

[פונקציות המסך 34](#_Toc74206555)

[תיעוד 35](#_Toc74206556)

[יום ראשון 10/5/2020 35](#_Toc74206557)

[יום ראשון 7/9/2020 35](#_Toc74206558)

[יום שני 12/10/2020 36](#_Toc74206559)

[מימוש תוכנה גרסה סופית 46](#_Toc74206560)

[פירוט פונקציות 46](#_Toc74206561)

[פונקציות בקוד של הארדואינו 46](#_Toc74206562)

[פונקציות בקוד של המשאבה 46](#_Toc74206563)

[פונקציות בקוד של הפוסט הנורמלי (nfosset): 52](#_Toc74206564)

[פוסט אוסמוזי 56](#_Toc74206565)

[פונקציות של המסך מגע 60](#_Toc74206566)

[הקוד של מד העכירות 62](#_Toc74206567)

[פונקציות בקוד של ה – esp (שנצרב ב – Arduino) 63](#_Toc74206568)

[פונקציות בקוד של השרת 66](#_Toc74206569)

[פונקציות של הברז הכדורי – Drainage 68](#_Toc74206570)

[מד גובה מים/חיישן גובה מים 70](#_Toc74206571)

[חיישן טמפרטורה 71](#_Toc74206572)

[חיישן תאורה 71](#_Toc74206573)

[רכיב RTC 71](#_Toc74206574)

[קוד מלא של המערכת 72](#_Toc74206575)

[Hydroponic 72](#_Toc74206576)

[Battery 80](#_Toc74206577)

[Drainage 82](#_Toc74206578)

[Espwifi 85](#_Toc74206579)

[Hydroponic.h 92](#_Toc74206580)

[lcd screen 103](#_Toc74206581)

[Nfosset 113](#_Toc74206582)

[Ofosset 122](#_Toc74206583)

[Pump 129](#_Toc74206584)

[Server 138](#_Toc74206585)

[Temperature 143](#_Toc74206586)

[Turbidity 150](#_Toc74206587)

[Turbidity 152](#_Toc74206588)

[אלגוריתם 158](#_Toc74206589)

[רפלקציה על התהליך ועל התוצר 159](#_Toc74206590)

# תקציר

הפרויקט שלי הוא מערכת הידרופונית מבוקרת. 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.

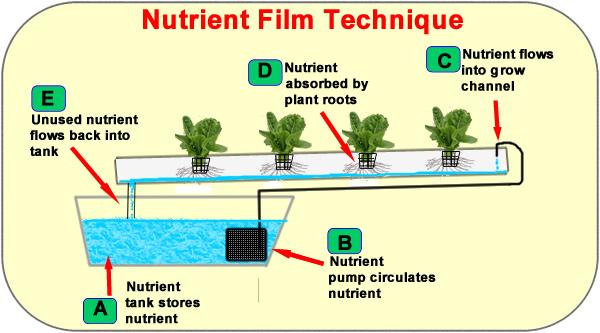
## תיאור הבעיה או הצורך

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.

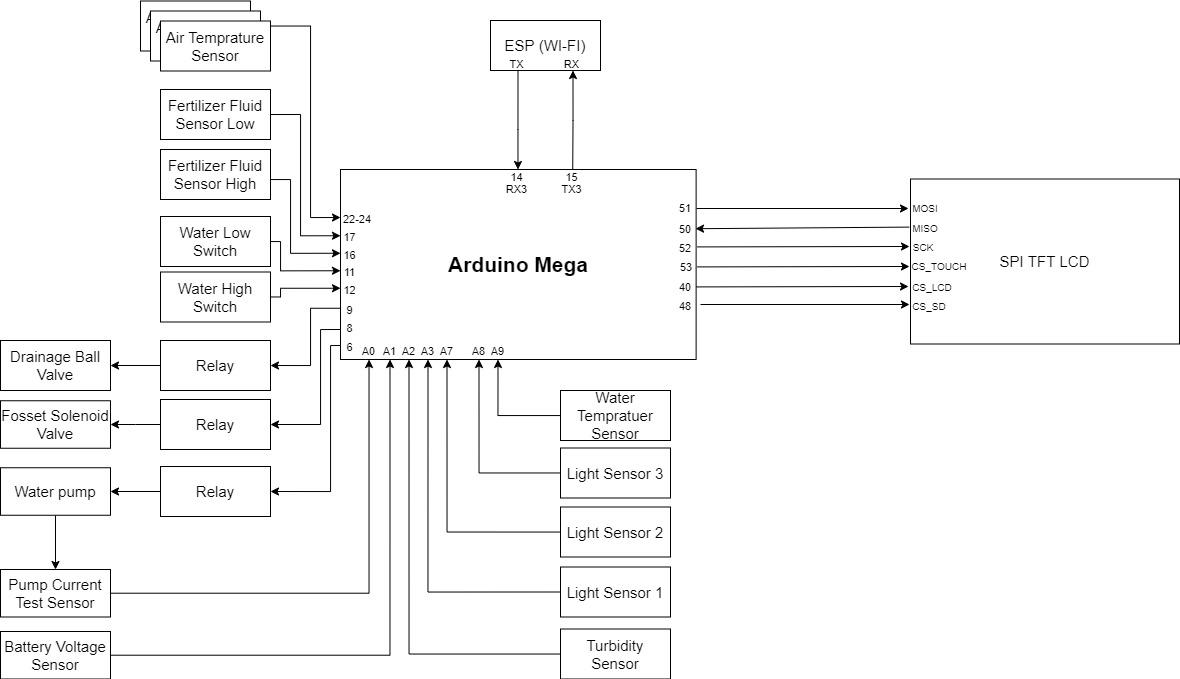
## מבנה הפרויקט

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

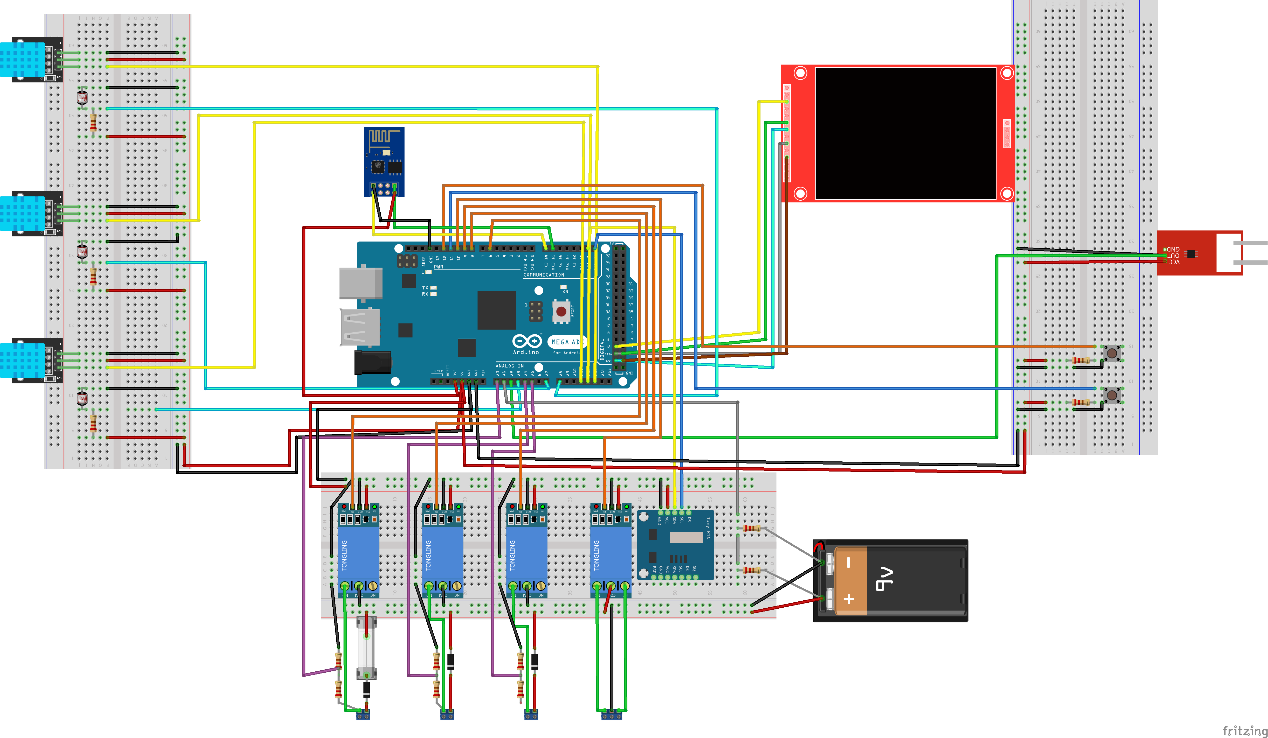


### הסבר תרשים מלבנים

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.
* 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).

## טיוטה לשרטוט חשמלי



## 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 | טווח לחצים: 0 – 10bar  טמפרטורת נוזל: -15 - 90 **°C**  הספק: 5w | 9 – 27v DC  85 – 250 AC/DC |  |
| ברז סולנואיד | 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 |  |
| מד גובה מים | צריכת זרם: |  |  |
| מד עכירות | צריכת זרם: 30mA  טמפרטורה: -30 - 80**°C** | 5v | פין אנלוגי |
| חיישן טמפרטורה ולחות (אוויר) DHT11 | צריכת זרם: 2.5mA  ניטרלי – 60uA  בשימוש – 0.3mA  טווח מדידה טמפרטורה: 0°C ~ 50°C דיוק של 2 מעלות  טווח מדידה לחות: 20% - 80% דיוק של 5%  רזולוציה: 16-bit  קצב דגימה 1Hz | 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) |  |  |  |

# שרטוט חשמלי סופי

## 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

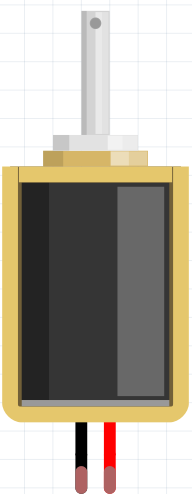
רכיב ה – rtc הוא רכיב שתפקידו לקבל ולשמור את הזמן הנוכחי או הזמן שהוזן לו, הוא מהווה כשעון אלקטרוני הרכיב מחובר לפינים sda ו scl והוא צורך מתח של 5v.

#### מסך מגע – TFT Lcd

Shape, rectangle

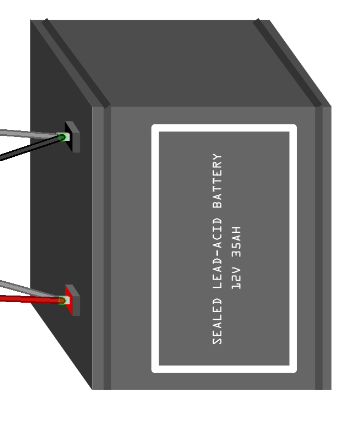
Description automatically generatedזהו מסך מסוג lcd התומך במגע (התנגדותי) מסך זה מחובר לארדואינו בפינים 52,50,51,40,53,43 הוא משמש לשליטה ישירה והעברת מידע באופן מקומי (כמו הגדרות בטלפון) המסך מתקשר עם הארדואינו בפרוטוקול spi.

#### ברז חשמלי – סולנואיד



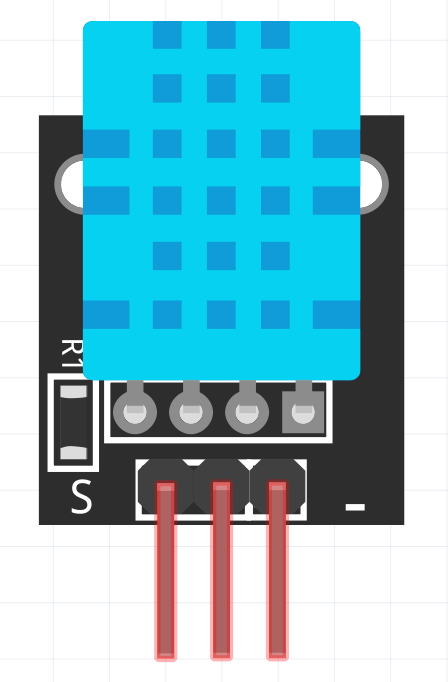
ברז זה תפקידו לאפשר או לא לאפשר זרימת מים למערכת הוא נשלט על ידי מנעול אלקטרוני (Relay) וקיימים שני רכיבים כאלה במערכת.

#### מצבר 12v (סוללת גיבוי למערכת)



מצבר זה מעביר מתח של 12v עם זרם מקסימלי של 7A ותפקידו לשמש כמתח גיבוי למערכת במידה ואספקת החשמל תיפסק.

#### חיישן טמפרטורה ולחות – DHT11



חיישן זה הוא חיישן טמפרטורה ולחות דיגיטלי שמטרתו היא למדוד את הטמפרטורה והלחות באוויר במערכת קיימים שלושה חיישנים כאלה והם מחוברים לפינים A11,A12,A13 (פינים אנלוגיים בארדואינו יכולים לשמש גם כפינים דיגיטליים). (לבדוק פרוטוקול תקשורת)

#### חיישן עכירות מים



חיישן זה הוא חיישן שמודד את עכירות המים החיישן שולח אות אנלוגי לארדואינו עם ערך דיגיטלי שמומר לאחוזים חיישן זה מחובר לפין A2. (לבדוק פרוטוקול תקשורת)

#### חיישן תאורה



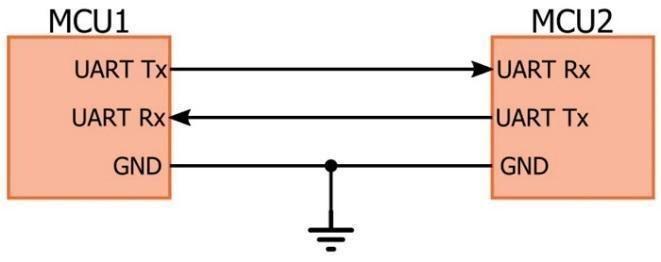
חיישן זה הוא חיישן עוצמת אור LDR הוא בעצם נגד שמשתנה על פי עוצמת האור. התנגדותו גדלה ככל שעוצמת האור גדלה הוא מחובר לפינים A3, A7,A8 והוא שולח אות אנלוגי שמומר לאחוזים.

# פרוטוקולים

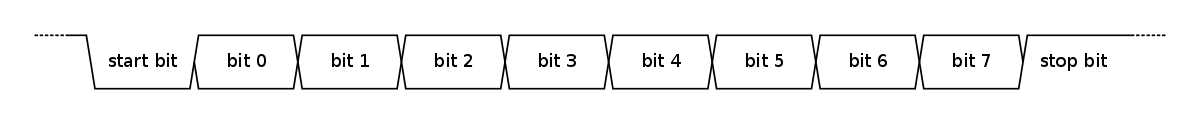
## פרוטוקול UART

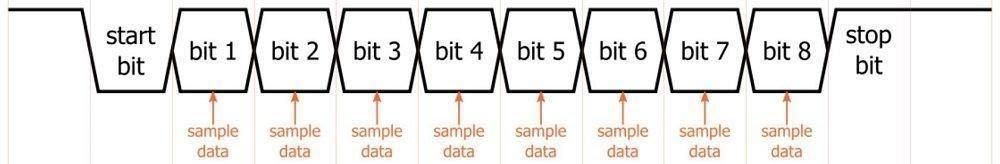
תקשורת UART  היא תקשורת טורית אסינכרונית, בניגוד לתקשורת SPI ו-I2C שעובדות עם שעון. תקשורת אסינכרונית עובדת ללא שעון, אבל יש צורך לתאם קצב העברת ביטים בין הרכיבים. כמו כן רוחב כל סיבית צריך להיות זהה כדי ששני הצדדים יידעו לקרוא נכון את המידע. קצב מקובל בארדואינו הוא 960Bps.

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).



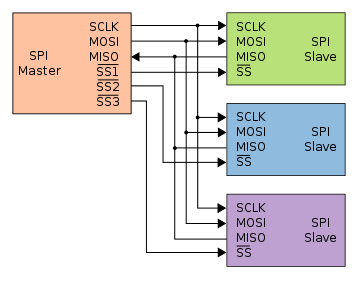
בפרוטוקול זה המידע מועבר ביחידות של בית (8 ביטים), כל פעם בית אחד, ה-LSB  קודם. כל בית מוקף בביט התחלה ובביט סיום. מצב של חוסר תקשורת הו מצב של "1" לוגי. ברגע שהקצב מתואם בין הצדדים, הדגימה של הביט נעשית באמצע הביט, וכך אין אי וודאות לגבי הערך של הביט.





## פרוטוקול SPI

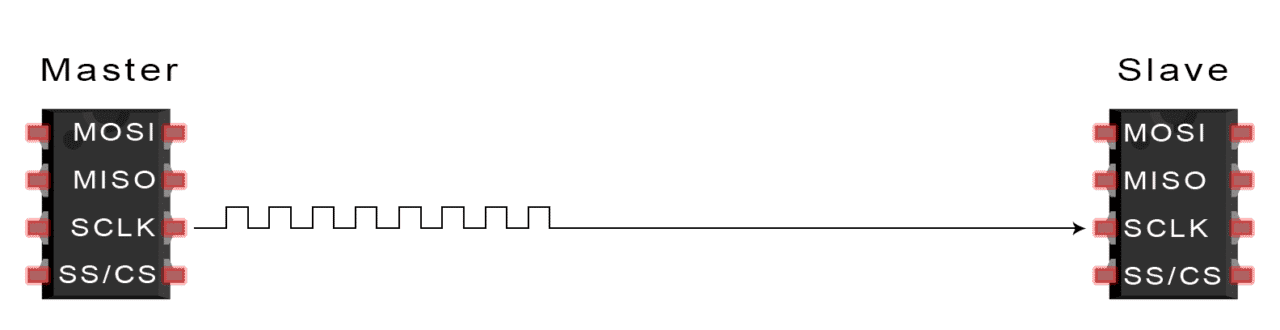
פרוטוקול SPI הינו פרוטוקול סינכרוני – כלומר כל הרכיבים יושבים על קווים משותפים ועובדים על שעון משותף. מכיוון שהוא פרוטוקול סינכרוני, המהירות שלו גבוהה (מהירות –MHz), מה שגורם לקצב העברת מידע גבוה. התקשורת של הפרוטוקול הינה תקשורת טורית, כלומר תקשורת דו כיוונית באותו זמן, ויש לו למעשה 4 קווי תקשורת. בפרוטוקול 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: 

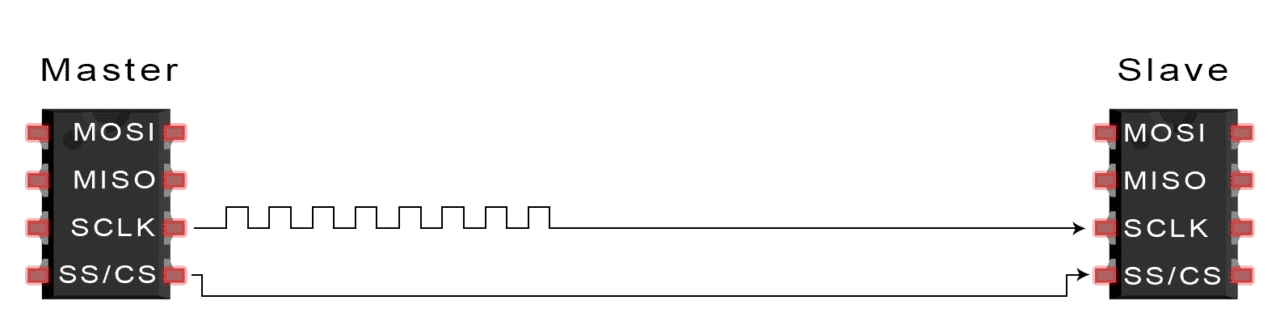
|  |  |  |
| --- | --- | --- |
| **שם הפין** | **כיוון** | **תפקיד** |
| SCLK | Master -> Slave | שעון שמסנכרן את כל היחידות |
| MOSI | Master -> Slave | Master Out Slave In  מעבר מידע מה-Master  ל-Slave |
| MISO | Slave - > Master | Master In Slave Out   מעבר מידע מה-Slave  ל-Master |
| CS או SS | Master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

תקשורת SPI

שלב 1 - ה-Master מוציא שעון.

****

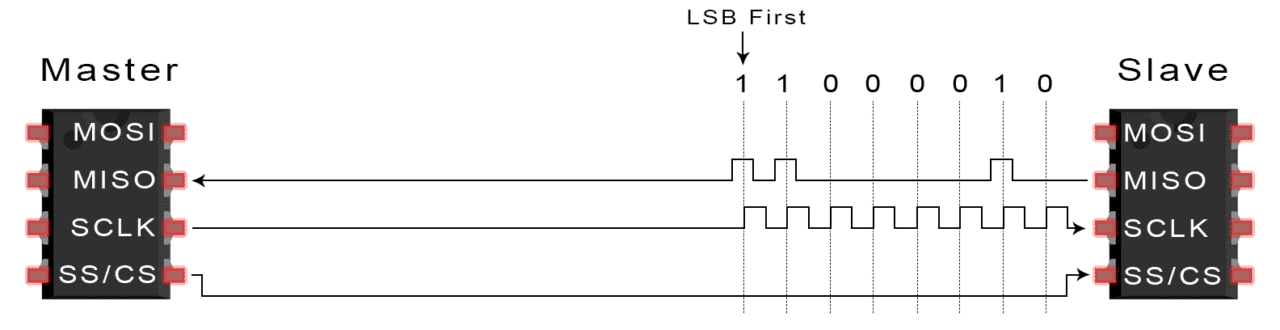
שלב 2 - ה-Master מוריד את הCS ל-Slave  הנבחר. כל עוד ה-CS אפס התקשורת מתקיימת.

****

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.

****

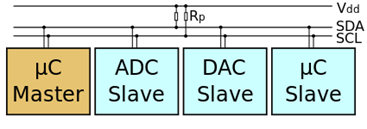
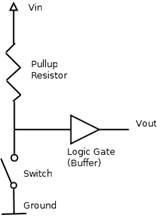
## 

## פרוטוקול I2C

### תאור כללי

הפרוטוקול נקרא I2C - Inter-Integrated Circuit מכיוון שהוא מתקשר בין שני רכיבים שונים. הפרוטוקול חסכוני  ובעל שני חוטים בלבד. הדבר מוזיל עלויות ייצור בשל מיעוט הקווים לתקשורת, וגודל הרכיב קטן שנדרש למימושו. שני הקווים נושאים גם את הכתובת וגם את המידע. הפרוטוקול עובד ב-Half-Duplex כלומר לא ניתן לכתוב ולקרוא באותו זמן. פין אחד משמש שעון שמסנכרן את המידע בין שני הרכיבים, ופין שני משמש להעברת המידע בצורה טורית.

הפרוטוקול I2C דורש שרכיב אחד יוגדר כ-Master וכל השאר כ-Slaves. ה-Master הוא זה שדוחף את השעון, ויוזם פנייה. ה-Slaves יכולים להיות יותר איטיים ולהאט את השעון של ה-Master ע"י משיכתו ל-0 כמה שצריכים.

הפינים הם מסוג open drain ולכן יש צורך לחבר אותם באמצעות . pull-up resistors: הנגד משמש להגדיר מצב ידוע של סיגנל. כאשר המתג של הטרנזיסטור פתוח, המתח נשאר Vcc, כאשר המתג סגור, היציאה היא 0 (מחוברת לאדמה). לכן כאשר ה-Slaves לא דוחפים מידע הפין נשאר ב-1 קבוע ונמשך ל-0 כאשר אחד מהרכיבים פעיל ורוצה לדחוף 0 לוגי.

### 

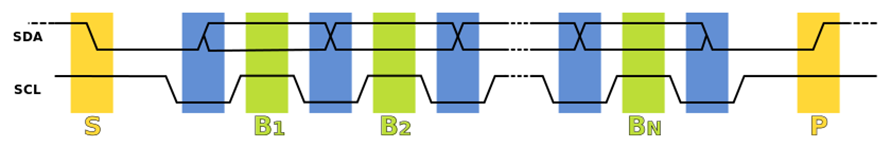
### אופן העברת מידע

מצב של חוסר תקשורת הוא מצב ששני הפינים נמצאים ב-1 לוגי. ה-Master  כאמור יוזם את התקשורת ע"י הורדת השעון לאפס, פעולה שמתפרשת כתחילת תקשורת (Start). פעולת סיום התקשורת (stop) היא פעולה שבה ה-Master מושך חזרה את השעון ל-1.

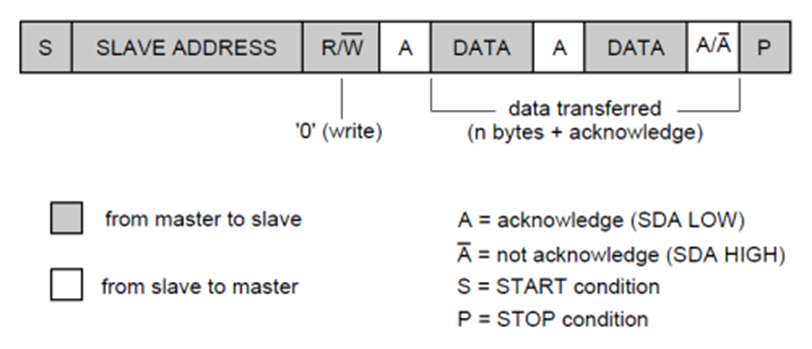
השידור מעשה בשיטת MSB . בהעברת המידע חייבים להתקיים הכללים:

* SDA צריך להיות יציב כש-SCL ב-1 לוגי (ירוק)
* SDA יכול להשתנות רק כאשר SCL ב-0 לוגי (כחול)

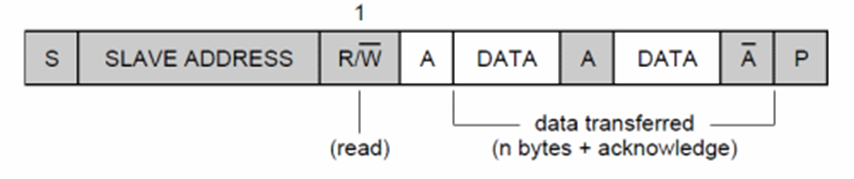
הרכיב המקבל את המידע (Master  או Slave) חייבים להגיב ב-Acknowledge באמצעות הורדת ה-SDA ל-0 לוגי. רכיב חייב להחזיר Ack מלבד כאשר הרכיב עסוק, או הרכיב קיבל מידע בלתי חוקי (שגיאה). פעולה שלא מחזירה Ack נקראת Nack, ובסיומה ה-Master חייב לבצע פעולת Stop.



### כתיבה מה-Master ל-Slave

המידע מועבר לפי הסדר של MSB תחילה. 

קריאה מה-Slave



## 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 | ~ |

### 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 |

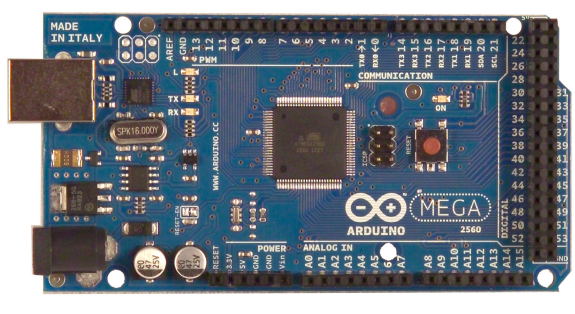
# רכיבים

## 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.



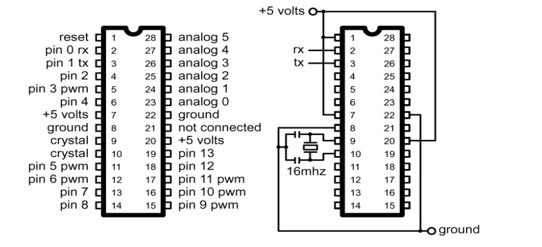
כרטיסי הפיתוח של הארדואינו זמינים במספר צורות ודגמים שונים, בפרויקט אני מתמקד בכרטיסArduino Mega ובלוח פיתוח.

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.

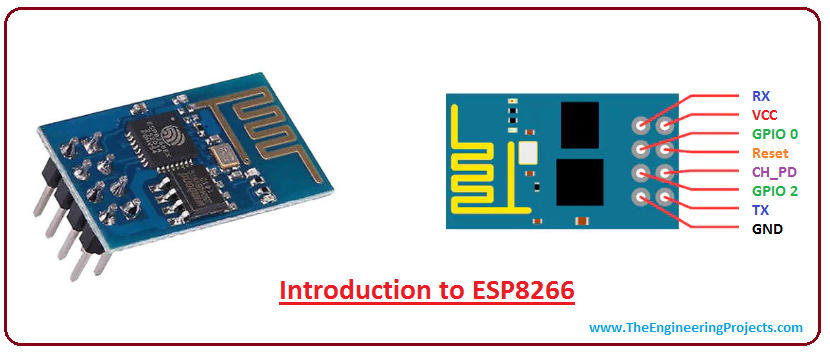
המעבד ATmega2560 תומך בפרוטוקולי תקשורת I2C ו SPI ו-UART.



## 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.

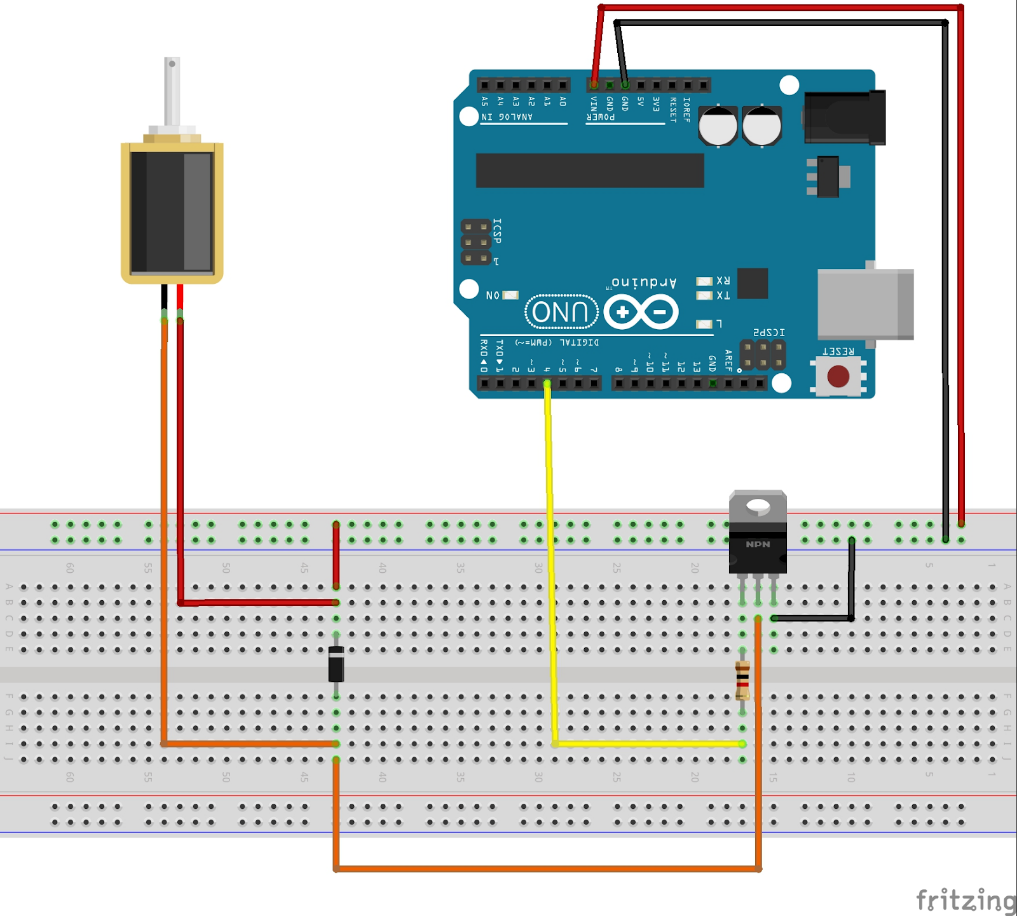
רכיב הIOT מתחבר דרך הWiFi אל השרת בעזרת פרוטוקול תקשורת HTTP(להוסיף פרוטוקול) פרוטוקול זה משמש מערכות אלקטרוניות דומות המבוססות על רשת, כדי להעביר מידע לשרת וגם כדי להחזיר מידע מהשרת.

## 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).

### חיבור

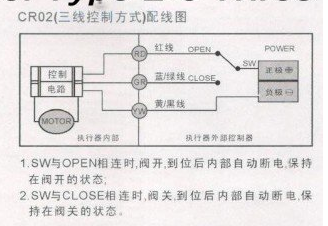
יש לחבר את הסולנואיד עם דיודה השומרת על המעגל מקפיצות במתח. הטרנזיסטור משמש כמגבר זרם. את הסולנואיד מחברים לפין דיגיטלי השולט בפתיחה וסגירה של הברז.

## 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.

### חיבור

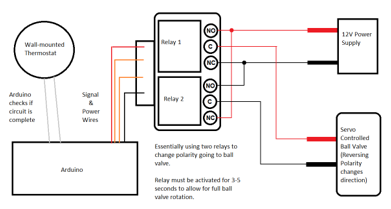
|  |  |  |
| --- | --- | --- |
| חוט | משמעות | חיבור |
| צהוב | אדמה | GND |
| אדום | ברז פתוח | Relay 1 |
| כחול | ברז סגור | Relay2 |

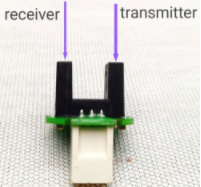


|  |  |  |
| --- | --- | --- |
| אדום | כחול | משמעות |
| 0 | 0 | עצירה |
| 1 | 0 | ברז פתוח |
| 0 | 1 | ברז סגור |
| 1 | 1 | עצירה? |

## 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.

### חיבור

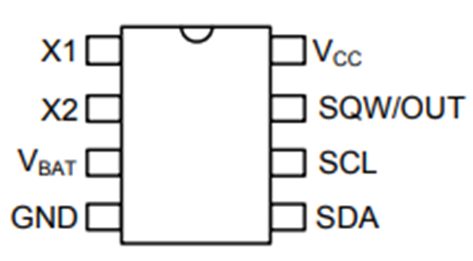
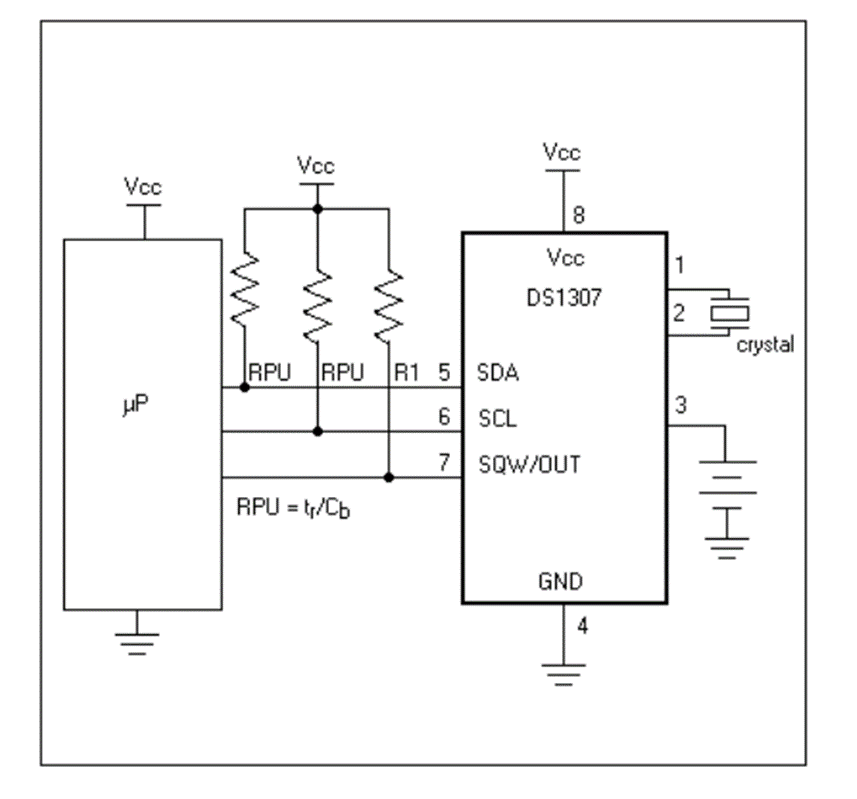
|  |  |  |
| --- | --- | --- |
| רגל | משמעות | חיבור לארדואינו |
| VCC | מתח 5V | VCC |
| GND | אדמה | GND |
| OUT | רמת עכירות | כל פין אנלוגי A0-A6 |

## 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.

### חיבור

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

### ספריה

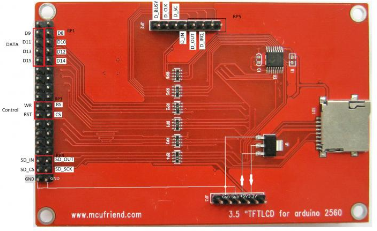
לתקשורת עם הרכיב נשתמש בספריה dht.h.

|  |  |
| --- | --- |
| **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

מסך 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. A picture containing diagram

Description automatically generatedDiagram, schematic

Description automatically generated

### 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.

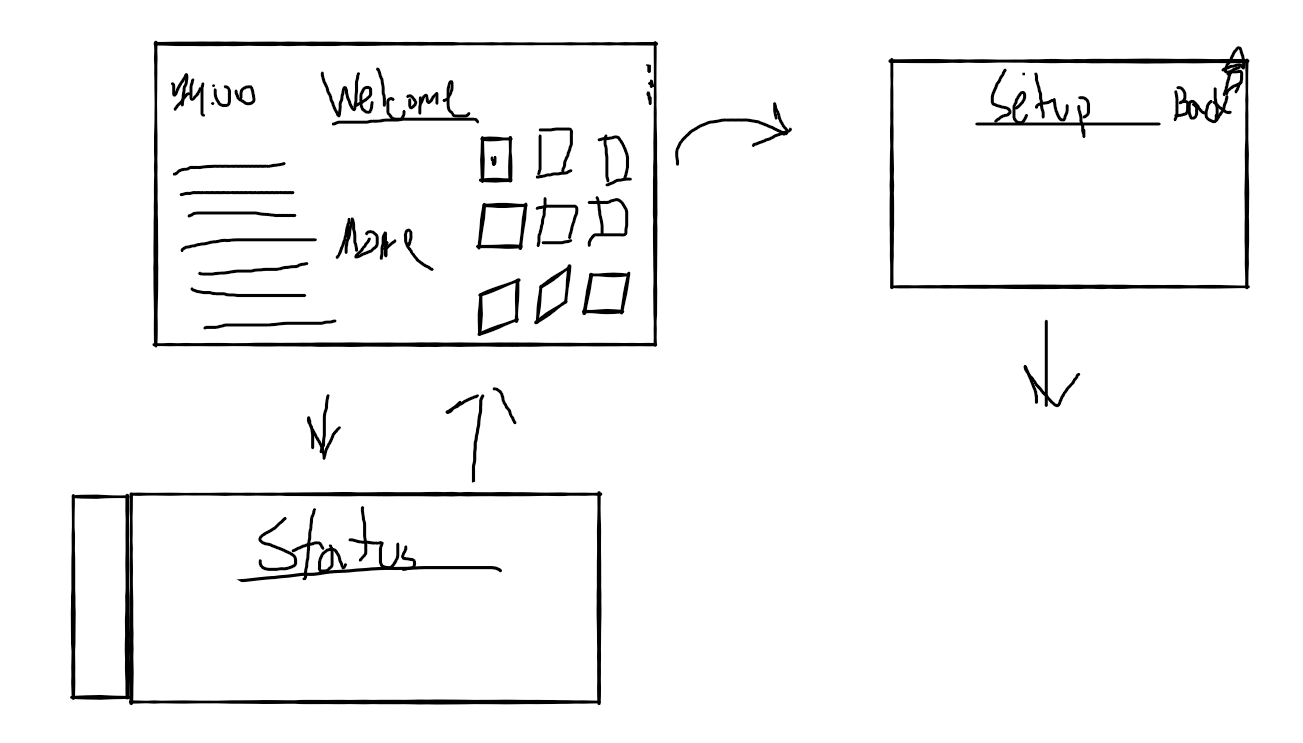




# תיעוד

## יום ראשון 10/5/2020

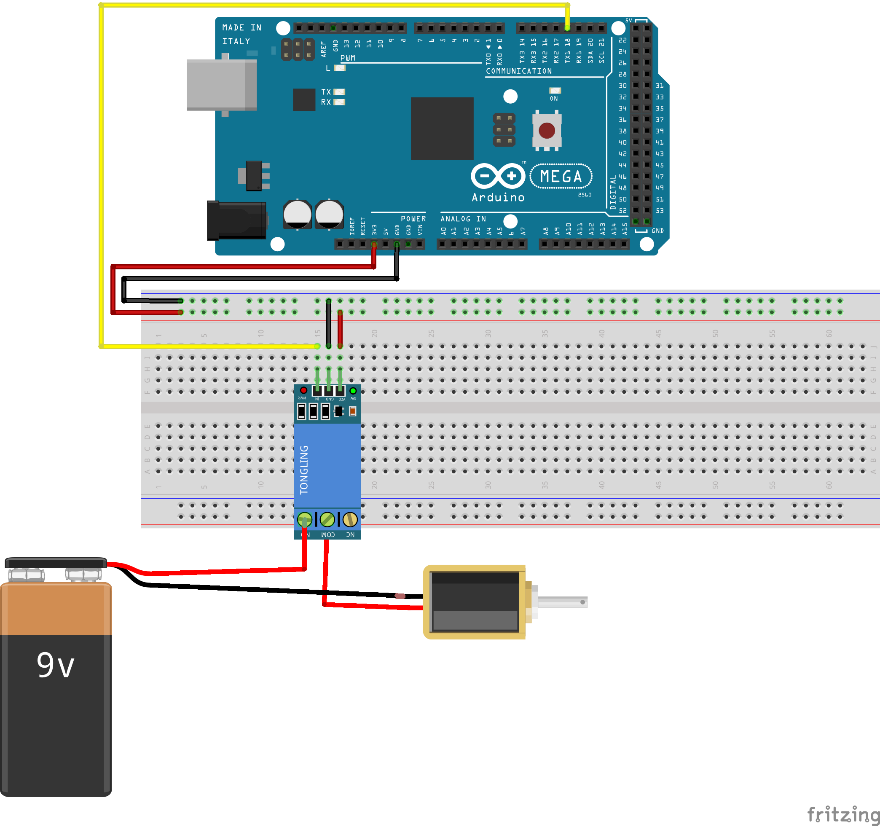
היום שוב עשינו פגישה דרך הזום לצערי הרב. עקב חוסר היכולת להגיע לרכיבים הפיזיים המשכנו לעבוד על הקוד. היום התחלנו לעבוד על תצוגת ה Lcd בנתיים תכננו איך הממשק ייראה וכבר עיצבנו את מסך הבית.



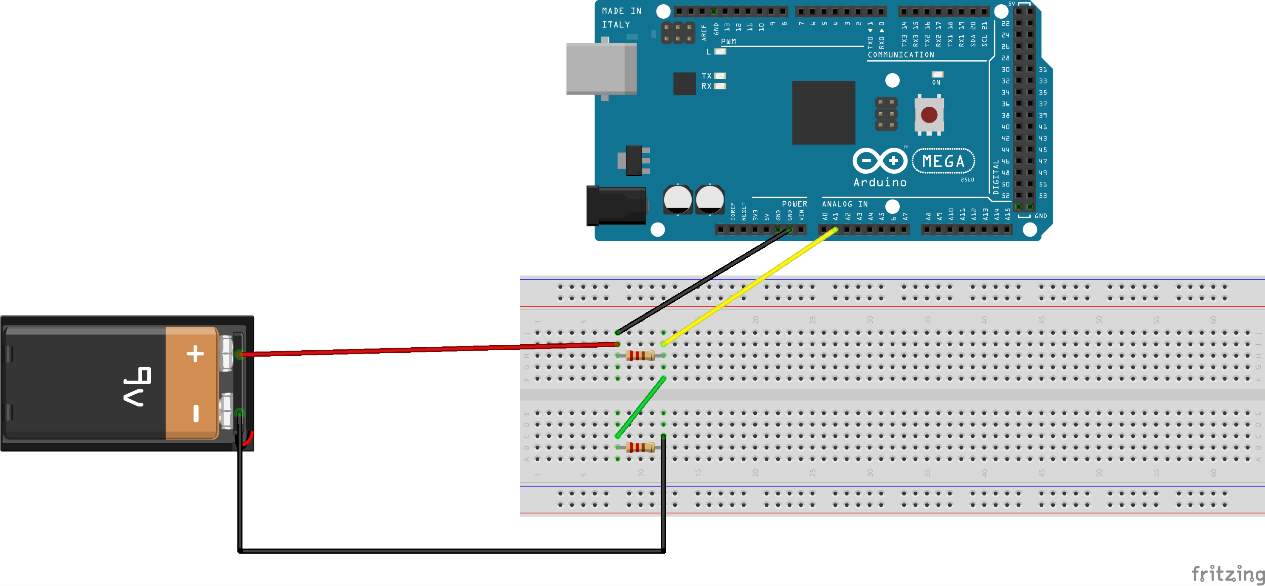
## יום ראשון 7/9/2020

היום התקדמנו בסידור של מסך הבית בממשק של המערכת והצגת האינפורמציה לגביי סטטוס החיישנים ובנינו פונקציות לבדיקת סטטוס הברזים במערכת (מבחינת זרם ומבחינת זרימה של מים).

## יום שני 12/10/2020

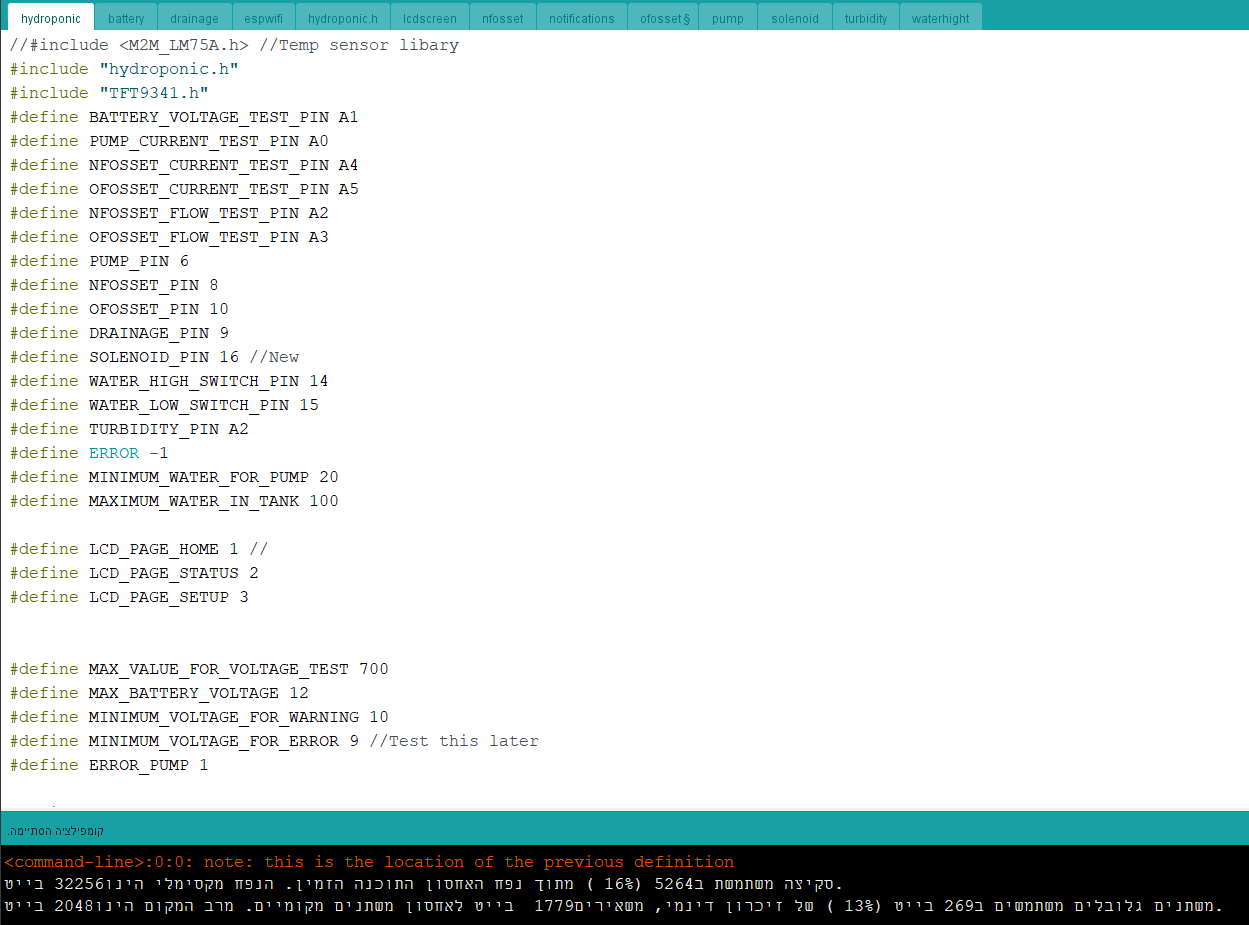
היום עבדתי על הסולנואיד, למדתי איך להפעיל אותו התנסיתי איתו וגם יצרתי מחלקה בשבילו בקוד הראשי של הפרוייקט.

בנוסף גם התקדמתי בבניית החיישן שמודד את מצב הסוללה (אחוזי סוללה) ויצרתי בשבילו מחלקה בקוד הראשי של הפרוייקט.



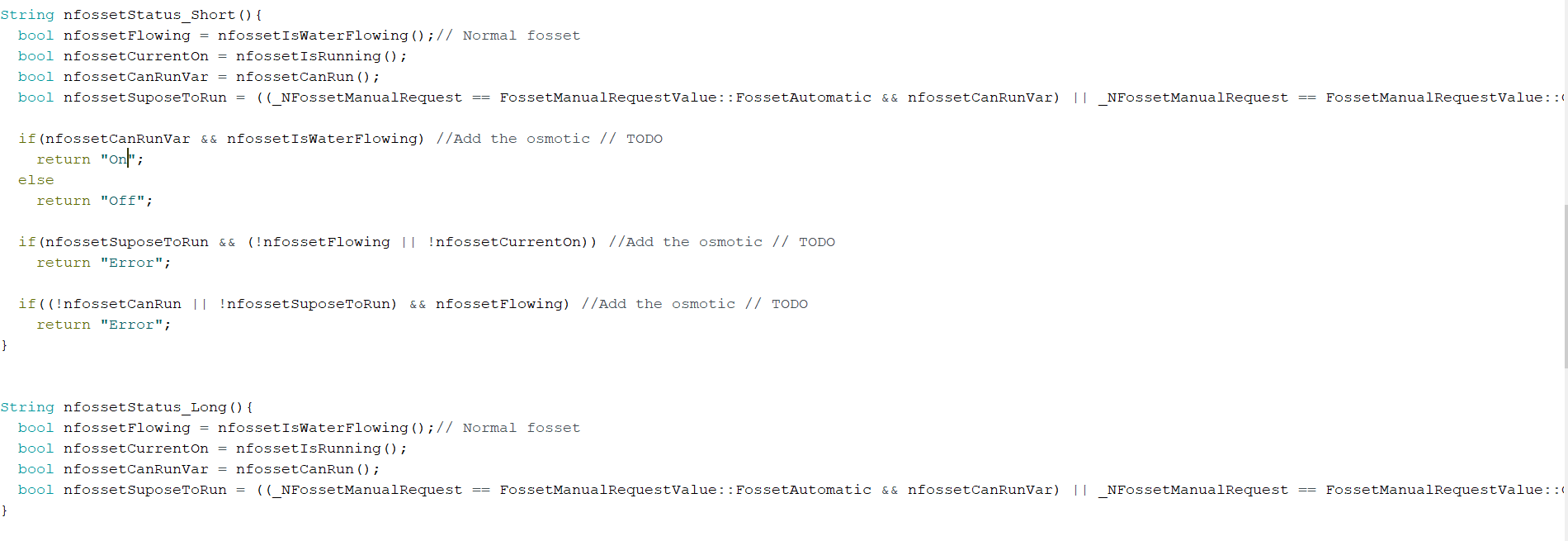
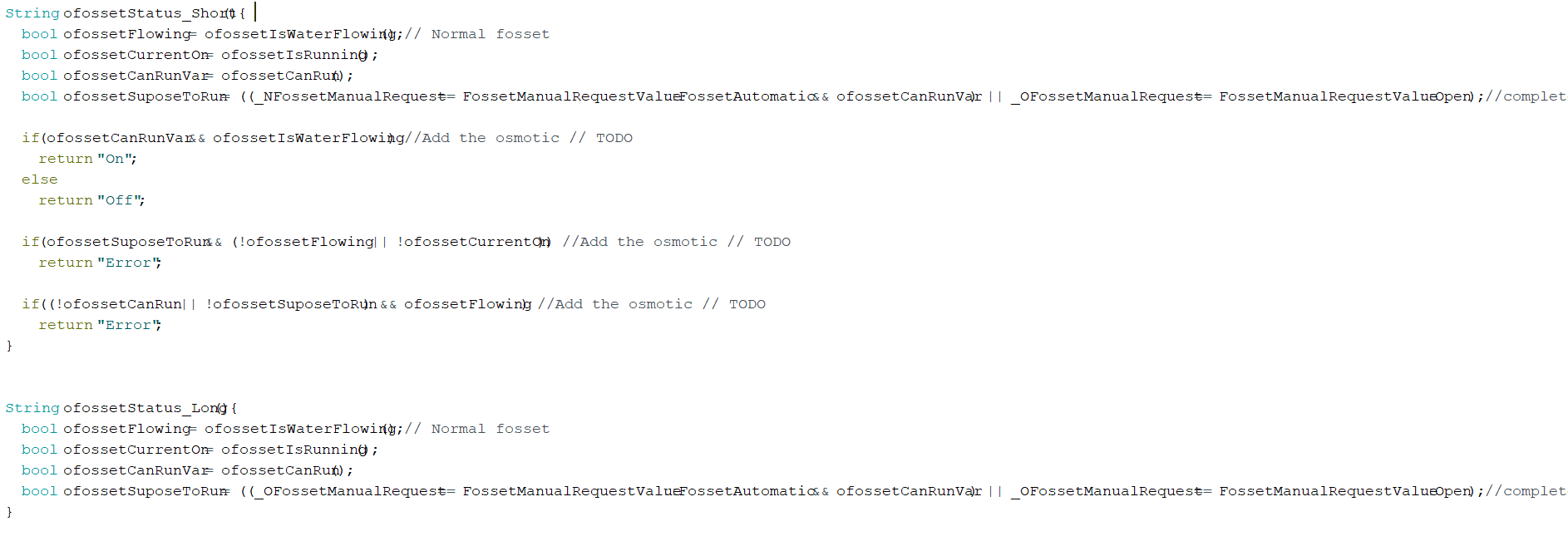
יום חמישי 15/10/2020

היום התמקדנו בלדאוג שהקוד יצליח להתקמפל. במהלך העבודה שלנו עם המסך גילינו באג מוזר, הפונקציות שהגדרנו בחלק התחתון יותר של התוכנה לא נקלטו ולכן החלטנו לשנות את הסדר של הקוד. יצרנו קבצים שונים לכל רכיב ב Arduino והגדרנו את כולם בקובץ h אחד כך שהבעיה לא תחזור על עצמה ואכן הצלחנו.



יום שני 19/10/2020

היום התקדמתי בפונקציות שבודקות את סטטוס הרכיבים (היום הספקתי רק הסולנואיד והאוסמוזה) בניתי לרכיבים האלה שתי פונקציות אחת שמוציאה סטטוס קצר שיוצג במסך הבית ואחת שמוציאה סטטוס ארוך ומפורט עם הודעות שגיאה מפורטות במידה ויש שאלה יוצגו בתפריט הסטטוס בממשק שנבנה במסך של ה Arduino.

יום ראשון 8/11/2020

היום התקדמתי בפונקציה הקצרה שבודקת את סטטוס הרכיבים ומציגה אותו על המסך במסך הבית בתפריט של המערכת. היום למדתי פונקציה חדשה בשם switch שמטרתה לבדוק מצבים ב enum ולשלוח פקודת בדיקה במידה והמערכת נמצאת באותו מצב. הספקנו לסיים את הפקודה של הnfosset + ofosset (2 הסולנואידים) והספקנו גם על המשאבה.

הדגמה של הפקודה ב Nfosset:



יום שני 9/11/2020

היום המשכתי להתקדם בסטטוסים ברכיבים אחרים ובדקתי והשלמתי את הסטטוסים שעשיתי בשיעור קודם בנוסף בנינו מפת עבודה לקראת השיעורים הבאים כי ככל הנראה אנחנו נחזור לבית ספר עד אז.

יום ראשון 15/11/2020

היום המשכתי לעבוד על מסך הבית (שוב) בשביל להציג את סטטוס הרכיבים במסך הבית עם צבעים!!! בנוסף סידרתי את הקוד כך שיחזיר צבע ותוצאה שצריך להציג.

יום שני 14/12/2020

היום התקדמנו עם בניית הפרוייקט בנינו את המעגל החיצוני של המערכת בו נמצאים ה relays שמחוברים לשאר המערכת בנוסף יצרתי מחבר בין המעגל המרכזי של הMega למעגל החיצוני החדש מאוחר יותר ביצעתי בדיקה של הנגדים כדי ליצור את מודד המתח של הרכיבים ושל מקור המתח של המערכת.

יום שני 21/12/2020

היום התקדמתי במעגל החיצוני חיברתי את הנגדים למערכת אחרי הבדיקה ובנוסף התחלתי בחיבור ה relays למחבר שיוצא לכיוון המעגל של ה Mega בנוסף ביצעתי בדיקה של המערכת עם אחד מהרכיבים והבדיקה עברה בהצלחה.

יום שישי 22/1/2021

היום התקדמנו עם התקשורת של ה esp לאחר החיבור לארדואינו היום ננסה ליצור תקשורת עם השרת בעזרת הכתובת שניתנה לנו וה api שניתן לנו <https://docs.google.com/document/d/1w8L4jMxFrutquqafYReikdoszqZtQo8MWmbC8jpTd0A/edit?ts=5fff1a39>

היום ננסה לבצע התחברות ראשונית לשרת ובנוסף ביצוע ניסיון של העברת מידע.

הפקודות להתחברות לרשת

//connect

//~c<SSID>$<password>$<timeout>$

הפקודות להתנתקות מהרשת

//disconnect

//~d

הפקודות לקבלת בקשה מהשרת

//GET request

//~g<request url>$

הפקודה לקבלת בקשות POST מהשרת

//POST request

//~p<request url>$<data>$

הפקודה לקבלת IP מהשרת

//get IP address

//~a

הפקודה לקבלת סטטוס של החיישן

//get status

//~s

הפקודה לבקשת שם של רשת wifi

יום ראשון 24/1/2021

היום התקדמתי עם ההגדרה של ה ESP ב – Arduino בחרתי בנתיים לוותר על הפקודה של ה getip למען סידור בנוסף הוספתי הערות של פתיחה וסגירה של הודעות כדי שהארדואינו יוכל לבדוק האם החיישן שלח לו הודעה מלאה. בהמשך אני אנסה ליצור כבר קשר עם השרת.

יום שני 25/1/2021

היום התקדמתי עם בניית קוד שמתקשר עם ה esp מהארדואינו כדי לבקש מהחיישן להתחבר לרשת לשרת וגם כדי לקבל ממנו נתונים ולוודא שהוא מחובר היום גם סוף סוף בנינו את הפונקציה שמכניסה את הכתובת של האתר על פי המידע שהוא צריך להעביר בלוג ובהעברת מידע על חיישן.

יום ראשון 31/1/2021

<https://www.instructables.com/Arduino-String-Manipulation-Using-Minimal-Ram/> - קביעת מחרוזות קבועות לעומת מחרוזות משתנות (דינאמיות). מפני ששימוש ב – class של string מוצא מחדש זיכרון (מה שגורם לחורים בזיכרון) מצאנו את הפתרון הזה.

יום חמישי 4/2/2021

היום ניסינו להפעיל את ה esp אחרי שסיימנו עם הקוד וניסינו להתחבר ל wifi לשרת (עדיין לא לשרת שלנו) וניסינו לשלוח פקודות get ו Post הניסיון עבר בהצלחה לאחר מכן פתחנו תפריט debug יותר נכון יצרנו כדי לצמצם את הפלט שיוצא מהחיישן כך שהארדואינו יקרא רק את מה שהוא צריך לקרוא לאחר בדיקה חוזרת ראינו שהכל עבד הפעלנו debug mode וכל הפלט התקבל וכשכיבינו אותו קיבלנו רק את מה שאנחנו צרכים לקבל.

יום ראשון 7/2/2021

היום התקדמנו עם החיבור של ה esp לארדואינו הראשי לאחר כמה תקלות שתיקנתי הצלחתי לשלוט על ה esp דרך הארדואינו וגם להתחבר לשרת מסויים כדי לבדוק בהמשך אני אנסה להתחבר לשרת הראשי של הפרוייקט. הקוד מתקמפל והכל רץ חלק.

יום ראשון 7/2/2021

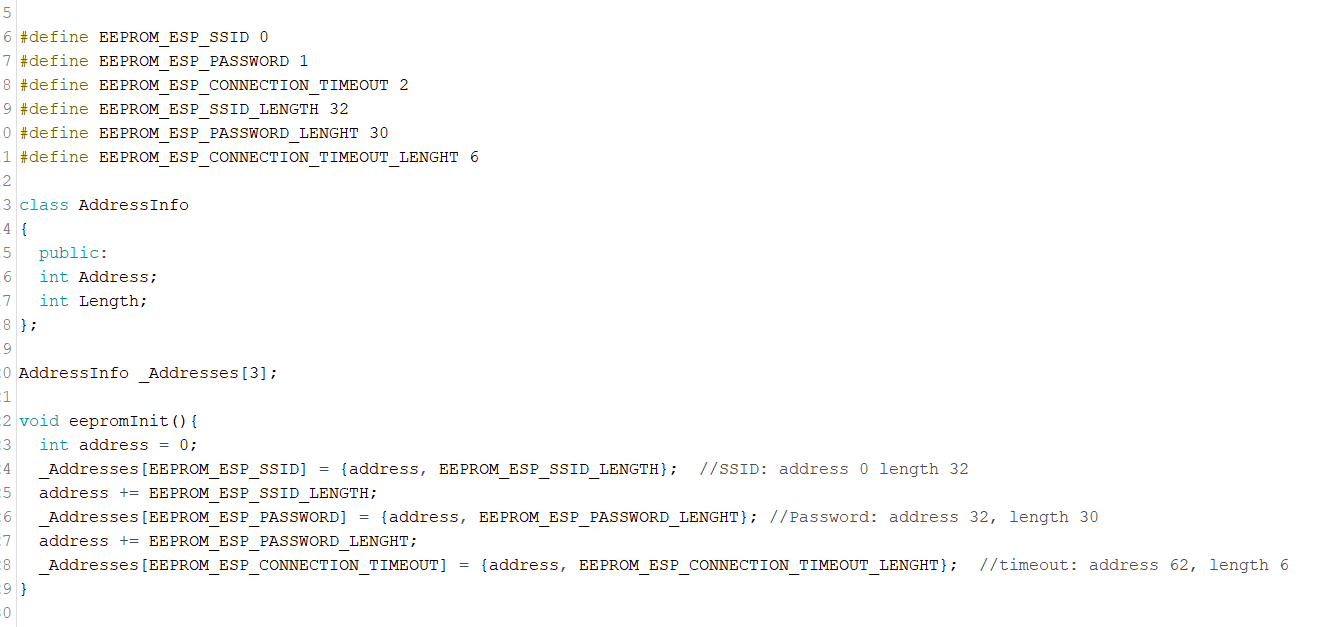
היום התקדמנו עם החיבור של ה esp לארדואינו הראשי לאחר כמה תקלות שתיקנתי הצלחתי לשלוט על ה esp דרך הארדואינו וגם להתחבר לשרת מסויים כדי לבדוק בהמשך אני אנסה להתחבר לשרת הראשי של הפרוייקט. הקוד מתקמפל והכל רץ חלק.

יום חמישי 11/2/2021

היום התחלנו לעבוד עם זיכרון ה eeprom של הארדואינו כדי לאחסן נתוני התחברות ל wifi זאת אומרת נגיד ואני רוצה להגדיר רשת חדשה במקום להגדיר אותה דרך הקוד אני שולח לארדואינו בקשה לשמור את הפרטי התחברות לרשת וברגע שהוא ירצה להתחבר הוא ימשוך אותם מהזיכרון

יום ראשון 14/2/2021

היום התקדמתי קצת עם הזיכרון של ה Arduino (eeprom) המשכתי לבנות את הפונקציות שלו ועדכנתי אותן בקוד של ה – esp המטרה בעצם היא לקבל מהמשתמש את פרטי ההתחברות לרשת ולשמור אותם בזיכרון וגם לעדכן במידת הצורך.



יום שני 14/2/2021

היום התקדמתי עם ההלחמות הלחמתי את הנגדים שמודדים את מעבר המתח לרכיבים החיצוניים ובנוסף גם התקדמתי עם חיבור ה relay לארדואינו דרך הלוח החיצוני. בנוסף גם הלחמתי את רכיב ה RTC שהוא בעצם השעון של הארדואינו למערכת. (להוסיף תמונה).

יום ראשון 21/2/2021

היום התעסקתי בטיפול בקוד עקב תקלה שבגללה אני לא מקבל פלט מה esp זה קורה רק בקוד הראשי כאשר אני מנסה קבצים אחרים הכל עובד לא הספקתי לתקן את הבעיה באותו היום אתקן בשיעור הבא.

יום שני 22/2/2021

תיקנתי את התקלה עם ה esp ועדכנתי את הקוד כדי להימנע מהתקלה היינו צרכים לחזור ל string בפקודות של הזיכרון של הארדואינו eeprom זאת הייתה הבעיה.

יום שני 1/3/2021

ה api של השרת התעדכן ולכן גם הייתי צריך לעדכן גם את הקוד של ה esp הקוד עודכן והצלחתי להתחבר לשרת בהצלחה בנוסף המשכתי לעבוד על החיישן טמפרטורה.

יום ראשון 7/3/2021

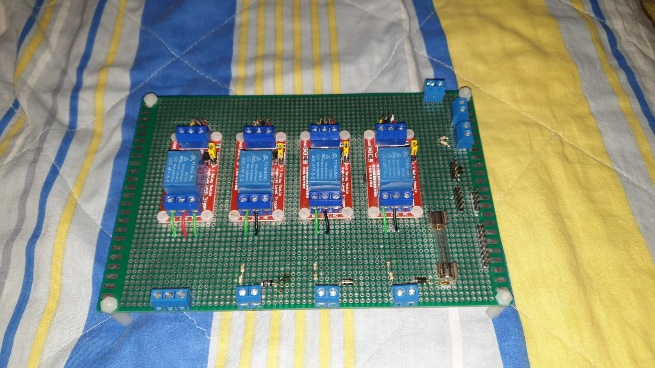
היום חזרתי בבניית המעגל החיצוני של הפרוייקט שבו מחוברים כל ה relays (\*לצרף תמונה) של הפרוייקט בגלל שמועד ההגשה מתקרב תיקנתי חלק מההלחמות והתקדמתי בהשלמת המעגל.

יום שני 8/3/2021

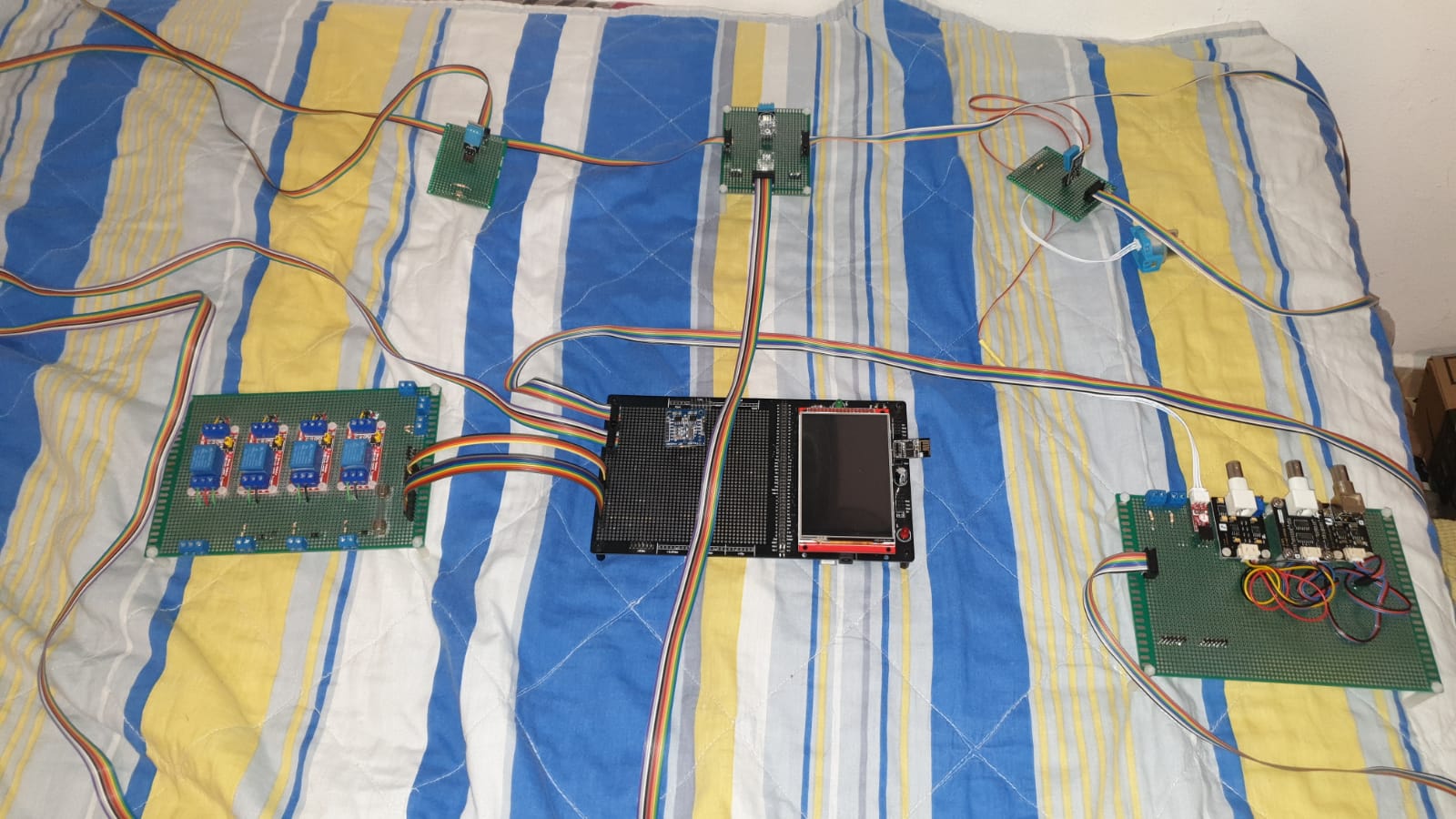
היום המשכתי עם בניית המעגל החיצוני וכמעט סיימתי עם ההלחמות.

יום שני 15/3/2021

היום סיימתי את בניית המעגל החיצוני שאליו מחוברים כל הרכיבים שצורכים מתח גבוה בפרוייקט (משאבה, ברזים וסולנואיד). ביצעתי בדיקה עם המשאבה ונתקלתי בבעיה עם הסוללה, היא לא מספקת מספיק מתח מה שגורם למשאבה ולשאר הרכיבים לעבוד בצורה לא יעילה. (להוסיף תמונה של המעגל החיצוני למשאבה והברזים). בסופו של דבר תפקידו של המעגל החיצוני הוא להפעיל את כל ה relays שאליהם מחוברים רכיבים כמו המשאבה ה – fosset והברז החשמלי.

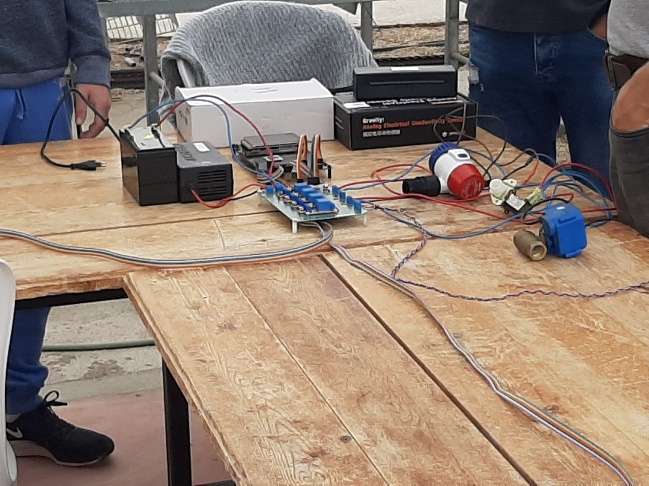


יום ראשון 21/3/2021

היום סיימתי את הבנייה של המערכת. בניתי שלושה לוחות חיצוניים בשביל חיישני הטמפרטורת אוויר ואור (להוסיף תמונה) בנוסף גם את המעגל החיצוני לחיישני גובה המים והעכירות. (להוסיף תמונה). מחר אני אפגש עם אנשי התחזוקה של בית הספר כדי לדון בבניה של המערכת ושילוב המערכות האלקטרוניות במערכת ההידרופונית שכבר מחכה בחממה. 

יום שני 22/3/2021

היום התכנסתי בחממה ביחד עם כל אנשי התחזוקה כדי לעדכן אותם לגביי הבינויים והתוספות שצריך להוסיף כדי לסיים את הבנייה של החממה בנוסף גם הסברתי להם איך הפרוייקט עובד את האוטומציה ובנוסף גם הסברתי לנציג מביולוגיה שהוא גם בסופו של דבר הלקוח איך כל המערכת עובדת המבנה של החיישנים והרכיבים ברחבי המערכת.



יום רביעי 31/3/2021

היום נפגשתי עם אדי כדי לעבור על המבנה של המערכת ולעדכן אותו בספר בנוסף התקדמנו גם בקוד עם חיישן הטמפרטורה וגם בדקנו את חיישני התאורה וגובה המים. לצערי חיישני גובה המים התגלו כבעיתיים ואנחנו מחפשים כרגע פתרון אולי חיישן חלופי.



יום שבת 3/4/2021

היום התקדמתי בבית עם בניית השרטוטים של הפרוייקט וכתיבת הפרקים בספר הפרוייקט.

יום שישי 9/4/2021

היום התקדמתי עם אדי על הקוד של הארדואינו, החלטנו סופית שהזיכרון eepro של הארדואינו כבר לא רלוונטי כי הוא יוצר המון תקלות ולכן הוצאנו אותו מהפרוייקט. בנוסף גם המשכנו לעבוד על הקוד של המסך עם הספרייה החדשה וקבענו את המבנה של התפרטים במסך

יום ראשון 11/4/2021

היום התקדמתי בכתיבה של הספר פרויקט, סיימתי את החלק הראשון שלו (לתאר את החלק בהמשך). והתחלתי לעדכן את תרשים הבלוקים של הפרוייקט מפני שהוא לא סופי. בנוסף אני ממשיך לאסוף את החלקים שצריך להכניס לפריצינג.

יום שני 12/4/2021

היום המשכתי להתקדם עם ספר הפרוייקט אך גם נזכרתי בחיבור של הארדואינו למקור מתח מתוך המערכת. הלחמתי חיבור שיצא מהמעגל החיצוני בעל המתח הגבוה ישירות לארדואינו אך בעת הבדיקות נתקלנו בבעיה, המתח הוא גבוה מדיי בשביל הארדואינו מפני שלפעמים הסוללה יכולה להיות טעונה במתח שהוא גבוה יותר מ 12V בנתיים אנחנו מחפשים פתרון לזה.

\*התיעוד לא מושלם

# 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() | PUMP\_PIN |  | void pumpInit(){  pinMode(PUMP\_PIN, OUTPUT);  } |
| להפעיל את המשאבה | void pumpStart() | PUMP\_PIN |  | void pumpStart(){  digitalWrite(PUMP\_PIN, HIGH);  \_IsPumpRunning = true;  } |
| לכבות את המשאבה | void pumpStop() | PUMP\_PIN |  | void pumpStop(){  digitalWrite(PUMP\_PIN, LOW);  \_IsPumpRunning = false;  } |
| לבצע בדיקה במשאבה על ידי הפעלה וכיבוי שלה | void pumpTest() |  |  | void pumpTest(){  pumpStart();  delay(500);  pumpStop();  delay(500);  } |
| לאפשר מעקף של האוטומציה והפעלה ידנית של המשאבה | void pumpManualControl(PumpManualRequestValue value) | PumpManualRequestValue |  | 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() | waterHightRead() | 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() | \_IsPumpRunning |  | bool pumpIsSignalOn(){  if(\_IsPumpRunning == 1){  return true;  }  else{  return false;  }  } //להעתיק את זה מחדש |
| לבדוק האם המשאבה אמורה לעבוד | bool pumpSuposeToRun() | PumpManualRequestValue  pumpCanRun() | האם המשאבה יכולה לרוץ | bool pumpSuposeToRun(){  return((\_PumpManualRequest == PumpManualRequestValue::PumpAutomatic && pumpCanRun()) || \_PumpManualRequest == PumpManualRequestValue::PumpStart);  } |
| לספק את הסטטוס של המשאבה | void pumpCheckStatus() | pumpIsWaterFlowing()  pumpIsCurrentOn()  pumpIsSignalOn()  pumpSuposeToRun() | 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() | PUMP\_CURRENT\_TEST\_PIN |  | void pumpCurrentInit(){  pinMode(PUMP\_CURRENT\_TEST\_PIN, INPUT);  } |
| לבצע בדיקה בחיישן זרם של המשאבה | void pumpCurrentTest() | pumpCurrentRead() |  | void pumpCurrentTest(){  Serial.print("Pump current:");  Serial.println(pumpCurrentRead());  } |
| בודקת האם המשאבה עובדת | bool pumpIsRunning() |  | 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.  } |
| לאתחל את מד המתח של המשאבה | pumpCurrentInit() | את הפין שאליו מחובר מד המתח של המשאבה | כלום |  |
| לבצע בדיקה האם יש מתח במשאבה | bool pumpIsCurrentOn() | את עוצמת המתח של המשאבה | האם המשאבה פועלת או לא (לפי המתח) |  |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| לאתחל את הפוסט | nfossetInit | את הפין שאליו הפוסט מחובר  NFOSSET\_PIN |  | void nfossetInit(){  pinMode(NFOSSET\_PIN, OUTPUT);  } |
| לפתוח את הפוסט הנורמלי | nfossetOpen | NFOSSET\_PIN |  | void nfossetOpen(){  digitalWrite(NFOSSET\_PIN, HIGH);  } |
| לסגור את הפוסט הנורמלי | nfossetClose | NFOSSET\_PIN |  | void nfossetClose(){  digitalWrite(NFOSSET\_PIN, LOW);  } |
| לבצע בדיקה בפוסט הנורמלי על ידי הפעלה וכיבוי שלו | nfossetTest |  |  | void nfossetTest(){  nfossetOpen();  delay(500);  nfossetClose();  delay(500);  } |
| לאפשר שליטה ידנית על הפוסט הנורמלי | nfossetManualControl | nfossetManualControl(FossetManualRequestValue value |  | 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 | nfossetCanRun | waterHightRead() | bool | 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 הפוסט הנורמלי | nfossetCheckStatus |  |  | 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 | nfossetCurrentInit | NFOSSET\_CURRENT\_TEST\_PIN |  | void nfossetCurrentInit(){  pinMode(NFOSSET\_CURRENT\_TEST\_PIN, INPUT);  } |
| לבצע בדיקה בחיישן הזרם של הפוסט הנורמלי | nfossetCurrentTest | nfossetCurrentRead() |  | void nfossetCurrentTest(){  Serial.print("nfosset current:");  Serial.println(nfossetCurrentRead());  } |
| לבדוק האם הפוסט הנורמלי עובד או לא | nfossetIsRunning | NFOSSET\_CURRENT\_TEST\_PIN | bool | 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 | ofossetInit | OFOSSET\_PIN |  | void ofossetInit(){  pinMode(OFOSSET\_PIN, OUTPUT);  } |
| פתיחה של הפוסט האוסמוזי | ofossetOpen | OFOSSET\_PIN |  | void ofossetOpen(){  digitalWrite(OFOSSET\_PIN, HIGH);  \_OfossetSignalOn = 1;  } |
| סגירה של הפוסט האוסמוזי | ofossetClose | OFOSSET\_PIN |  | void ofossetClose(){  digitalWrite(OFOSSET\_PIN, LOW);  \_OfossetSignalOn = 0;  } |
| לבצע בדיקה בפוסט האוסמוזי על ידי הפעלה וכיבוי שלו | ofossetTest |  |  | void ofossetTest(){  ofossetOpen();  delay(500);  ofossetClose();  delay(500);  } |
| לאפשר שליטה ידנית על הפוסט האוסמוזי | ofossetManualControl | FossetManualRequestValue value) |  | 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 | fossetCanRun | MAXIMUM\_WATER\_IN\_TANK  waterHightRead() | bool | 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;  } |
| לבדוק האם הפוסט האוסמוזי קיבל בקשה לפעול | ofossetIsSignalOn |  | bool | bool ofossetIsSignalOn(){  if(\_OfossetSignalOn == 1){  return true;  }  else  return false;  } |
| להוציא סטטוס קצר של הפוסט האוסמוזי | ofossetStatus\_Short | ofossetIsWaterFlowing()  ofossetIsCurrentOn()  ofossetIsSignalOn()  ofossetSuposeToRun() | StatusResult | void ofossetCheckStatus(){  bool ofossetWaterFlowing = ofossetIsWaterFlowing();  bool ofossetCurrentOn = ofossetIsCurrentOn();  bool ofossetSignalOn = ofossetIsSignalOn();  bool ofossetSuposeToRunVar = ofossetSuposeToRun();  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(ofossetSignalOn){  \_OfossetStatus.SignalOn = StatusResult("Error: Fosset signal pin on while water levle too high", STATUS\_ERROR, PRIORITY\_HIGH);  \_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_OfossetStatus.SignalOn = StatusResult("Off: Fosset signal off", STATUS\_OK, PRIORITY\_LOW);  \_OfossetStatus.Short = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);  }  }  else{  if(ofossetSuposeToRun){  if(!ofossetSignalOn){  if(h > MINIMUM\_WATER\_FOR\_PUMP){  \_OfossetStatus.SignalOn = StatusResult("Warning: Fosset signal pin off while water levle is low", STATUS\_WARNING, PRIORITY\_MEDIUM);  \_OfossetStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_MEDIUM);  }  else{  \_OfossetStatus.SignalOn = StatusResult("Error: Fosset signal pin off while water levle is under pump minimum", STATUS\_ERROR, PRIORITY\_HIGH);  \_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  }  else{  \_OfossetStatus.SignalOn = StatusResult("On: Fosset signal pin on", STATUS\_OK, PRIORITY\_LOW);  \_OfossetStatus.Short = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);  }  }  }      if(ofossetSignalOn){  if(!ofossetWaterFlowing){  if(h > MINIMUM\_WATER\_FOR\_PUMP){  \_OfossetStatus.WaterFlowing = StatusResult("Warning: Water not flowing while signal pin is on", STATUS\_WARNING, PRIORITY\_MEDIUM);  \_OfossetStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_MEDIUM);  }  else{  \_OfossetStatus.WaterFlowing = StatusResult("Error: Water not flowing while water levle is under pump minimum", STATUS\_ERROR, PRIORITY\_HIGH);  \_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  }  else{  \_OfossetStatus.WaterFlowing = StatusResult("On: Water flowing", STATUS\_OK, PRIORITY\_LOW);  if(\_OfossetStatus.Short.Status == STATUS\_OK){  \_OfossetStatus.Short = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);  }  }  if(!ofossetCurrentOn){  if(h > MINIMUM\_WATER\_FOR\_PUMP){  \_OfossetStatus.CurrentOn = StatusResult("Warning: No current while signal pin is on", STATUS\_WARNING, PRIORITY\_MEDIUM);  if(\_OfossetStatus.Short.Status != STATUS\_ERROR){  \_OfossetStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_MEDIUM);  }  }  else{  \_OfossetStatus.CurrentOn = StatusResult("Error: No current while water levle is under pump minimum", STATUS\_ERROR, PRIORITY\_HIGH);  \_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  }  else{  \_OfossetStatus.CurrentOn = StatusResult("Current on", STATUS\_OK, PRIORITY\_LOW);  if(\_OfossetStatus.Short.Status == STATUS\_OK){  \_OfossetStatus.Short = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);  }  }  }  else{  if(ofossetWaterFlowing){  \_OfossetStatus.WaterFlowing = StatusResult("Error: Water flowing while signal pin is off", STATUS\_ERROR, PRIORITY\_HIGH);  \_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_OfossetStatus.WaterFlowing = StatusResult("No water flowing", STATUS\_OK, PRIORITY\_LOW);  if(\_OfossetStatus.Short.Status == STATUS\_OK){  \_OfossetStatus.Short = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);  }  }  if(ofossetCurrentOn){  \_OfossetStatus.CurrentOn = StatusResult("Error: Fosset has current while signal pin is off", STATUS\_ERROR, PRIORITY\_HIGH);  \_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_OfossetStatus.CurrentOn = StatusResult("No current", STATUS\_OK, PRIORITY\_LOW);  if(\_OfossetStatus.Short.Status == STATUS\_OK){  \_OfossetStatus.Short = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);  }  }  }  } |

#### פונקציות של המסך מגע

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| אתחול של מסך ה – LCD | lcdInit |  |  | void lcdInit(){  lcd.Init\_LCD();  lcdT.TP\_Init();  lcd.Fill\_Screen(WHITE);  } |
| בדיקה של מסך ה – LCD | lcdTest |  |  | void lcdTest(){  lcdShowHomeScreen();  lcdShowHomeScreenStatus();  } |
| להציג את מסך הבית של המערכת על גבי המסך | lcdShowHomeScreen |  |  | void lcdShowHomeScreen(){  if (\_Lcd\_Status == LCD\_PAGE\_HOME){  return;  }  \_Lcd\_Status = LCD\_PAGE\_HOME;  lcd.Set\_Rotation(1);  lcd.Set\_Text\_Size(4);  lcd.Set\_Text\_colour(BLACK);  lcd.Set\_Text\_Back\_colour(BLACK);  lcd.Print\_String("Home Screen", 105, 4);  lcd.Set\_Draw\_color(BLACK);  lcd.Draw\_Fast\_HLine(105,35,265); //make a line on the screen  lcd.Set\_Text\_Size(2);  lcd.Print\_String("Status", 5, 45);  lcd.Draw\_Rectangle(5,60,230,315);  lcd.Print\_String("Nfosset:", 10, 65);  lcd.Print\_String("Ofosset:", 10, 85);  lcd.Print\_String("Pump:", 10, 105);  lcd.Print\_String("Drainage:", 10, 125);  lcd.Print\_String("Battery:", 10, 145);  lcd.Print\_String("Water:", 10, 165);  lcd.Print\_String("Drainage:", 10, 185);  lcd.Print\_String("Temprature:", 10, 205);  lcd.Print\_String("Humidity:", 10, 225);  lcd.Print\_String("WI-FI:", 10, 245);  lcd.Print\_String("Server:", 10, 265);  //Buttons  lcd.Set\_Text\_Size(4);  lcd.Draw\_Rectangle(240,60,470,130);  lcd.Print\_String("Status", 280,83);    lcd.Draw\_Rectangle(240,140,470,210);  lcd.Print\_String("Config", 280,160);    lcd.Draw\_Rectangle(240,220,470,290);  lcd.Print\_String("About", 280,240);  lcd.Set\_Text\_Size(2);    lcdHandleTouch();  } |
| לבדוק האם התבצע מגע באחד הכפתורים במסך ובמידה והתבצע לעבור לתפריט המבוקש | lcdHandleTouch |  |  | void lcdHandleTouch(){  if(lcdT.TP\_Scan()){  int x = lcdT.getX();  int y = lcdT.getY();  if(\_Lcd\_Status == LCD\_PAGE\_HOME){  if(x >= 240 && x <= 470 && y >= 60 && y <= 130){ //Status Screen button 240,60,470,130  Serial.println("Status");  lcdShowStatusScreen();  }  if(x >= 240 && x <= 470 && y >= 140 && y <= 210){ //Configuration Screen Button 240,140,470,210  Serial.println("Config");  lcdShowConfigScreen();  }  if(x >= 240 && x <= 470 && y >= 220 && y <= 290){ //About Screen Button 240,220,470,290  Serial.println("About");  lcdShowAboutScreen();  }  }  } |
| לעדכן את הסטטוס על גבי מסך הבית ב – LCD | lcdShowHomeScreenStatus |  |  | void lcdShowHomeScreenStatus(){  lcd.Print\_String(" ", 105, 65);  lcdWriteStatus(105, 65, \_NfossetStatus.Short);  lcd.Print\_String(" ", 105, 85);  lcdWriteStatus(105, 85, \_OfossetStatus.Short);  lcd.Print\_String(" ", 105, 105);  lcdWriteStatus(100, 105, \_PumpStatus.Short);  lcd.Print\_String(" ", 105, 125);  lcdWriteStatus(100, 125, \_DrainageStatus.Short);  lcd.Print\_String(" ", 105, 145);  lcdWriteStatus(100, 145, \_BatteryStatus.Short);  lcd.Print\_String(" ", 105, 165);  lcdWriteStatus(100, 165, \_WaterHightStatus.Short);    } |
| תבנית לכתיבת סטטוס קצר על גבי מסך הבית | lcdWriteStatus | int x, int y, StatusResult res |  | 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 | lcdChangeColor | int priority |  | void lcdChangeColor(int priority)  {  switch (priority)  {  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 | turbidityInit | TURBIDITY\_PIN |  | void turbidityInit(){  pinMode(TURBIDITY\_PIN, INPUT);  } |
| בדיקה של מד העכירות | turbidityTest | turbidityReadPercent() |  | void turbidityTest(){  int value = turbidityReadPercent();  Serial.println(value);  } |
| המרת הערך שחוזר ממד העכירות לאחוזים | turbidityReadPercent | TURBIDITY\_PIN | int | 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) | turbidityStatus\_Short | tubidityIsWorking() | String | 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 | turbidityStatusLong | tubidityIsWorking() | String | 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 | espInit |  |  | void espInit(){  Serial3.begin(9600);  } |
| לבצע בדיקה לרכיב ה – esp | espTest | readFromEsp() |  | void espTest(){  Serial.print("This is a test");  espSetDebugMode(true);  ConnectToWifi();  String content;  do {  content = readFromEsp();  if (content != NO\_DATA)  Serial.println(content);  }  while(content.indexOf("CONNECTION\_STATUS:CONNECTED") == -1);  byte sensorId = 4;  long timeStamp = 0;  int value = 6900;  // serverSendSensorData(sensorId, timeStamp, value);  //espSendGetRequest("http://hamama.ew.r.appspot.com/board?cmd=measure&sid=4&time=0&value=6900"); //TODO Make a get request for the server on espSendGetRequest...  do{  content = readFromEsp();  if (content != NO\_DATA)  Serial.println(content);  }  while(content.indexOf("RESAULT\_STATUS:") == -1);  }  byte \_WifiConnectionStatus;  long \_WifiConnectionStatusTime;  void espSetDebugMode(bool mode){  if(mode == true){  Serial3.print("~b");  }  else{  Serial3.print("~e");  }  } |
| להתחבר מחדש לרשת ה wifi | espReconnect | ConnectToWifi() |  | void espReconnect(){  switch(\_WifiConnectionStatus){  case CONNECTION\_STATUS\_CONNECTED:  return;  case CONNECTION\_STATUS\_WAITING:  ConnectToWifi();  case CONNECTION\_STATUS\_CONNECTING:  return;  case CONNECTION\_STATUS\_TIMEOUT:  case CONNECTION\_STATUS\_ERROR:  if(millis() - \_LastTimeConnectRequest > ESP\_TIME\_TO\_RECONNECT){  ConnectToWifi();  }  else{  return;  }  case CONNECTION\_STATUS\_MISSING\_INFO:  return;  }  } |
| לקרוא את המידע שהesp שולח לארדואינו | readFromEsp | \_SerialReadBuffer | String | String readFromEsp(){  if(Serial3.available()){  String a1 = Serial3.readString();  Serial.print(a1);  int index = a1.indexOf(END\_OF\_RETRUN\_STRING);  if(index >= 0 ){  String temp = a1.substring(0, index) + \_SerialReadBuffer;  \_SerialReadBuffer = a1.substring(index,a1.length());  return temp;  }  else{  \_SerialReadBuffer = \_SerialReadBuffer + a1;  return NO\_DATA;  }  }  return NO\_DATA;  } |
| להתחבר ל wifi | ConnectToWifi | espGetWifiSSID()  espGetWifiPassword()  espGetConnectionTimeout() |  | void ConnectToWifi(){  String ssid;//[EEPROM\_ESP\_SSID\_LENGTH];  String password;//[EEPROM\_ESP\_PASSWORD\_LENGHT];  ssid = espGetWifiSSID();  password = espGetWifiPassword();  int timeout = espGetConnectionTimeout();  espConnectToWifi(ssid, password, timeout);  } |
| לשלוח ל esp בתקשורת סריאלית בקשה להתחבר ל wifi | espConnectToWifi | String ssid, String password, int connectionTimeout |  | void espConnectToWifi(String ssid, String password, int connectionTimeout){  Serial3.print("~c");  Serial3.print(ssid);  Serial3.print(END\_OF\_COMMAND\_STRING);  Serial3.print(password);  Serial3.print(END\_OF\_COMMAND\_STRING);  Serial3.print(connectionTimeout);  Serial3.print(END\_OF\_COMMAND\_STRING);  \_LastTimeConnectRequest = millis();  } |
| לשלוח פקודה ל esp להתנתק מהרשת | espDisconnectFromWifi |  |  | void espDisconnectFromWifi(){  Serial3.print("~d");  } |
| לשלוח בקשה ל esp להעביר את כתובת ה ip שהוא קיבל | espGetIp |  |  | void espGetIp(){  Serial3.print("~a");  } |
| שליחת בקשת get לשרת עם כתובת http | espSendGetRequest | String url |  | void espSendGetRequest(String url){  //~g<request url>$  Serial3.print("~g");  Serial3.print(url); Serial3.print(END\_OF\_COMMAND\_STRING);  } |
| שליחת בקשת post לשרת עם כתובת http | espSendPostRequest | long reqid, String url, String data |  | void espSendPostRequest(long reqid, String url, String data){  //~p<request url>$<data>$  Serial3.print("~p");  Serial3.print(reqid);  Serial3.print(END\_OF\_COMMAND\_STRING);  Serial3.print(url);  Serial3.print(END\_OF\_COMMAND\_STRING);  Serial3.print(data);  Serial3.print(END\_OF\_COMMAND\_STRING);  } |

#### פונקציות בקוד של השרת

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| לקבל את המזהה של הבקשה | getRequestID | \_RequestID | long | long getRequestID(){  long reqID = \_RequestID;  \_RequestID++;  return reqID;  } |
| לבצע בדיקה בשרת | serverTest | //פקודה זמנית |  | void serverTest(){  serverRandomTest();  } |
| שליחת לוג מהארדואינו לשרת | serverSendLog | byte sensorId, long timeStamp, byte priority, String message |  | void serverSendLog(byte sensorId, long timeStamp, byte priority, String message){  \_ServerRequestType = SERVER\_REQUEST\_LOG;  String url = "http://"+serverGetAdress()+"/board?";  url += "cmd=log";  String data = "";  if(sensorId != 0){  data += "sid=" + sensorId;  data += "&";  }  data += "time=";  data += timeStamp;  data += "&";  data += "priority=";  data += priority;  data += "&";  data += "message=";  data += message;  espSendPostRequest(getRequestID(), url, data);  /// Testing with get request ///  //espSendGetRequest(url);  //url = url + "?" + data;  } |
| לשלוח מידע לשרת על חיישן ספציפי | serverSendSensorData | byte sensorId, long timeStamp, int value |  | void serverSendSensorData(byte sensorId, long timeStamp, int value){  //TODO: add request to queue and send data from queue periodically. delete data from queue only after receiving confirmation that it arived on server.  \_ServerRequestType = SERVER\_REQUEST\_SENSOR\_DATA;  String url = "http://"+serverGetAdress()+"/board";  String data = "cmd=measure";  data += "&";  data += "sid=" + String(sensorId);  data += "&";  data += "time=";  data += String(timeStamp);  data += "&";  data += "value=";  data += String(value);  espSendPostRequest(getRequestID(), url, data);  ///// Testing with get request /////  //url = url + "?" + data;  //espSendGetRequest(requestId, url);  } |
| לשלוח פינג לשרת | serverSendPing |  |  | void serverSendPing(){  \_ServerRequestType = SERVER\_REQUEST\_PING;  String url = "http://"+serverGetAdress()+"/board?";  url += "cmd=ping";  \_ServerConnectionStatusTime = millis();  } |
| להתחבר מחדש לשרת | void serverReconnect |  |  | void serverReconnect(){  if(\_WifiConnectionStatus!=CONNECTION\_STATUS\_CONNECTED){  return;  }  if(\_ServerConnectionStatus!=SERVER\_CONNECTION\_STATUS\_CONNECTED){  serverSendPing();  }  else if(millis() - \_ServerConnectionStatusTime > SERVER\_TIME\_TO\_RECONNECT){  serverSendPing();  }  } |
| לשלוח הודעת שגיאה לשרת | void serverNotifyError(int sensorId, int errorType, String errorMsg, String timeStamp) | את המזהה של הרכיב את סוג השגיאה את הודעת השגיאה את חתימת הזמן של השגיאה | כלום | void serverNotifyError(int sensorId, int errorType, String errorMsg, String timeStamp){  \_ServerRequestType = SERVER\_REQUEST\_LOG;  String url = "http://"+serverGetAdress()+"/board?";  url += "cmd=log";  String data = "";  if(sensorId != 0){  data += "sid=" + sensorId;  data += "&";  }  data += "time=";  data += timeStamp;  data += "&";  data += "priority=";  data += PRIORITY\_HIGH;  data += "&";  data += "errorType=";  data += errorType;  data += "&";  data += "message=";  data += errorMsg;  espSendPostRequest(getRequestID(), url, data);  /// Testing with get request ///  //espSendGetRequest(url);  //url = url + "?" + data;  } |

#### פונקציות של הברז הכדורי – Drainage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| לבצע בדיקה בברז הכדורי | drainageTest |  |  | void drainageTest(){  drainageOpen();  delay(500);  drainageClose();  delay(500);  } |
| לאתחל את הברז הכדורי | drainageInit |  |  | void drainageInit(){  pinMode(DRAINAGE\_PIN, OUTPUT);  drainageClose();  } |
| לפתוח את הברז הכדורי | drainageOpen |  |  | void drainageOpen(){  digitalWrite(DRAINAGE\_PIN, HIGH);  \_DrainageIsOpen = true;  } |
| לסגור את הברז הכדורי | drainageClose |  |  | void drainageClose(){  digitalWrite(DRAINAGE\_PIN, LOW);  \_DrainageIsOpen = false;  } |
| לבצע בדיקה האם הברז הכדורי קיבל פקודה להיפתח או להיסגר | drainageIsOpen |  | bool | bool drainageIsOpen(){  return \_DrainageIsOpen;  } |
| לבצע שליטה ידנית בברז הכדורי | drainageManualControl | DrainageManualRequestValue value |  | void drainageManualControl(DrainageManualRequestValue value){  switch (value)  {  case DrainageManualRequestValue::DrainageClose:{  \_DrainageManualRequest = value;  drainageClose();  break;  }  case DrainageManualRequestValue::DrainageOpen:{  \_DrainageManualRequest = value;  drainageOpen();  break;  }  case DrainageManualRequestValue::DrainageAutomatic:{  \_DrainageManualRequest = value;  break;  }  }  } |
| לקבל סטטוס קצר או ארוך של הברז הכדורי (הוא מהווה גם ניקוז במערכת) | drainageCheckStatus |  |  | void drainageCheckStatus(){  bool drainageSignal = drainageIsOpen();  if(DrainageManualRequestValue::DrainageOpen){  if(drainageSignal){  \_DrainageStatus.SignalOn = StatusResult("Open (manual)", STATUS\_OK, PRIORITY\_LOW);  \_DrainageStatus.Short = StatusResult("Open", STATUS\_OK, PRIORITY\_LOW);  }  else{  \_DrainageStatus.SignalOn = StatusResult("Error, signal off while manual request is on", STATUS\_ERROR, PRIORITY\_HIGH);  \_DrainageStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  }  else if(DrainageManualRequestValue::DrainageClose){  if(drainageSignal){  \_DrainageStatus.SignalOn = StatusResult("Error, signal on while manual request is off", STATUS\_ERROR, PRIORITY\_HIGH);  \_DrainageStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);  }  else{  \_DrainageStatus.SignalOn = StatusResult("Close (manual)", STATUS\_OK, PRIORITY\_LOW);  \_DrainageStatus.Short = StatusResult("Close", STATUS\_OK, PRIORITY\_LOW);  }  }  else{ //Automatic  if(drainageSignal){  \_DrainageStatus.SignalOn = StatusResult("Open (automatic)", STATUS\_OK, PRIORITY\_LOW);  \_DrainageStatus.Short = StatusResult("Open", STATUS\_OK, PRIORITY\_LOW);  }  else{  \_DrainageStatus.SignalOn = StatusResult("Close (automatic)", STATUS\_OK, PRIORITY\_LOW);  \_DrainageStatus.Short = StatusResult("Close", STATUS\_OK, PRIORITY\_LOW);  }  }  } |

#### מד גובה מים/חיישן גובה מים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| לבצע בדיקה בחיישן גובה המים | waterHightTest | WATER\_LOW\_SWITCH\_PIN  WATER\_HIGH\_SWITCH\_PIN |  | void waterHightTest(){  //Sensor test  Serial.print("Water low sensor:");  Serial.println(digitalRead(WATER\_LOW\_SWITCH\_PIN));  Serial.print("Water high sensor:");  Serial.println(digitalRead(WATER\_HIGH\_SWITCH\_PIN));  delay(1000);  } |
| לאתחל את חיישן גובה המים | waterHightInit) | WATER\_LOW\_SWITCH\_PIN  WATER\_HIGH\_SWITCH\_PIN |  | void waterHightInit(){  pinMode(WATER\_HIGH\_SWITCH\_PIN, INPUT);  pinMode(WATER\_LOW\_SWITCH\_PIN, INPUT);  } |
| לקרוא את גובה המים מהחיישן | int waterHightRead() | WATER\_LOW\_SWITCH\_PIN  WATER\_HIGH\_SWITCH\_PIN | int | int waterHightRead(){  bool waterInLow = !digitalRead(WATER\_LOW\_SWITCH\_PIN); //\* off/down = 1, on/up = 0  bool waterInHigh = !digitalRead(WATER\_HIGH\_SWITCH\_PIN); //\* true 1 , false 0.  if(!waterInLow){  return 0;  }  else if(waterInHigh){  return 100;  }  else if (!waterInHigh && waterInLow){  return 50;  }  else {  return ERROR;  }  } |

#### חיישן טמפרטורה

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
| אתחול חיישן הטמפרטורה | tempratureSensorInit |  |  | void tempratureSensorInit(){  dht.begin();  } |
| בדיקת חיישן הטמפרטורה | tempratureTest |  |  | void tempratureTest(){  float humi1 = dht1.readHumidity();  float tempC1 = dht1.readTemperature();  float humi2 = dht2.readHumidity();  float tempC2 = dht2.readTemperature();  float humi3 = dht3.readHumidity();  float tempC3 = dht3.readTemperature();  Serial.print("Humidity: ");  Serial.print(humi1);  Serial.print("%");  Serial.print(" | ");  Serial.print("Temperature: ");  Serial.print(tempC1);  Serial.print("°C ~ ");  } |

#### חיישן תאורה

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
|  |  |  |  |  |

\*עדיין בתהליך

#### רכיב RTC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תפקידה של הפונקציה | שם הפונקציה | מה הפונקציה מקבלת | מה הפונקציה מחזירה | מימוש הפונקציה |
|  |  |  |  |  |

\*עדיין בתהליך

# קוד מלא של המערכת

## Hydroponic

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

/////// Smart Hydroponic Greenhouse - Made by Nave Sasoni 2020-2021 ///////

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

#include "DHT.h"

#include <SoftwareSerial.h>

#include "hydroponic.h"

#include "ili9488.h"

#define BATTERY\_VOLTAGE\_TEST\_PIN A1

#define PUMP\_CURRENT\_TEST\_PIN A0

#define PUMP\_FLOW\_TEST\_PIN A6

#define NFOSSET\_CURRENT\_TEST\_PIN A4

#define OFOSSET\_CURRENT\_TEST\_PIN A5

#define NFOSSET\_FLOW\_TEST\_PIN A2

#define OFOSSET\_FLOW\_TEST\_PIN A3

#define PUMP\_PIN 6

#define NFOSSET\_PIN 8

#define OFOSSET\_PIN 9

#define DRAINAGE\_PIN 10

#define SOLENOID\_PIN 16 //New

#define WATER\_HIGH\_SWITCH\_PIN 12

#define WATER\_LOW\_SWITCH\_PIN 11

#define TURBIDITY\_PIN A2

#define TEMPERATURE\_SENSOR1\_PIN A10

#define TEMPERATURE\_SENSOR2\_PIN A11

#define TEMPERATURE\_SENSOR3\_PIN A12

#define LIGHT\_SENSOR1\_PIN A3

#define LIGHT\_SENSOR2\_PIN A7

#define LIGHT\_SENSOR3\_PIN A8

//ESP TX Pin 14, ESP RX Pin 15

#define WATER\_TANK\_TOTAL\_HIGHT 50 //50 cm water tank height

#define ERROR -1

#define MINIMUM\_WATER\_FOR\_PUMP 20 //MINIMUM\_WATER\_IN\_TANK

#define MAXIMUM\_WATER\_IN\_TANK 90

#define LCD\_PAGE\_HOME 1 //

#define LCD\_PAGE\_STATUS 2

#define LCD\_PAGE\_CONFIG 3

#define LCD\_PAGE\_ABOUT 4

#define STATUS\_SUBPAGE\_MAIN 0

#define STATUS\_SUBPAGE\_NFOSSET 1

#define STATUS\_SUBPAGE\_OFOSSET 2

#define STATUS\_SUBPAGE\_PUMP 3

#define STATUS\_SUBPAGE\_DRAINAGE 4

#define STATUS\_SUBPAGE\_BATTERY 5

#define STATUS\_SUBPAGE\_WATER 6

#define STATUS\_SUBPAGE\_TEMPERATURE 7

#define STATUS\_SUBPAGE\_HUMIDITY 8

#define STATUS\_SUBPAGE\_TURBIDITY 9

#define STATUS\_SUBPAGE\_WIFI 10

#define STATUS\_SUBPAGE\_SERVER 11

#define MAX\_VALUE\_FOR\_VOLTAGE\_TEST 700

#define MAX\_BATTERY\_VOLTAGE 12

#define MINIMUM\_VOLTAGE\_FOR\_WARNING 10

#define MINIMUM\_VOLTAGE\_FOR\_ERROR 9 //Test this later

#define ERROR\_PUMP 1

#define PRIORITY\_HIGH 1

#define PRIORITY\_MEDIUM 2

#define PRIORITY\_LOW 3

#define STATUS\_OK 1

#define STATUS\_WARNING 2

#define STATUS\_ERROR 3

#define STATUS\_NORMAL\_COLOR BLACK

#define STATUS\_OK\_COLOR GREEN

#define STATUS\_WARNING\_COLOR YELLOW

#define STATUS\_ERROR\_COLOR RED

#define MINIMUM\_TEMPERATURE\_FOR\_WARNING 7

#define MINIMUM\_TEMPERATURE\_FOR\_ERROR 2

#define MAXIMUM\_TEMPERATURE\_FOR\_WARNING 32

#define MAXIMUM\_TEMPERATURE\_FOR\_ERROR 40

#define MINIMUM\_HUMIDITY\_FOR\_WARNING 50

#define MINIMUM\_HUMIDITY\_FOR\_ERROR 30

#define MAXIMUM\_HUMIDITY\_FOR\_WARNING 90

#define MAXIMUM\_HUMIDITY\_FOR\_ERROR 95

#define TIMEOUT\_FOR\_LAST\_VALUD\_WATER\_HEIGHT 20000

enum DrainageManualRequestValue{

DrainageOpen,

DrainageClose,

DrainageAutomatic

};

enum FossetManualRequestValue{

FossetOpen,

FossetClose,

FossetAutomatic

};

enum PumpManualRequestValue{

PumpStart,

PumpStop,

PumpAutomatic

};

//M2M\_LM75A \_Lm75a; //Temp Sensor

DrainageManualRequestValue \_DrainageManualRequest;

FossetManualRequestValue \_OFossetManualRequest;

FossetManualRequestValue \_NFossetManualRequest;

PumpManualRequestValue \_PumpManualRequest;

bool \_IsPumpRunning;

bool \_PumpError;

bool \_VoltageError;

//int temperatureGet();

int \_Lcd\_Status; // LCD\_PAGE\_HOME, LCD\_PAGE\_STATUS, LCD\_PAGE\_CONFIG, LCD\_PAGE\_ABOUT.

int \_Lcd\_Status\_Subpage;

void setup(){

serialInit();

temperatureSensorInit();

ofossetInit();

nfossetInit();

pumpInit();

drainageInit();

turbidityInit();

pumpCurrentInit();

waterHightSensorInit();

lcdInit();

espInit();

eepromInit();

lcdShowHomeScreen();

pumpManualControl(PumpManualRequestValue::PumpAutomatic);

ofossetManualControl(FossetManualRequestValue::FossetAutomatic);

nfossetManualControl(FossetManualRequestValue::FossetAutomatic);

}

void loop(){

checkStatus();

lcdHandleTouch();

if(\_Lcd\_Status == LCD\_PAGE\_HOME){

lcdShowHomeScreenStatus();

}

else if(\_Lcd\_Status == LCD\_PAGE\_STATUS){

lcdShowStatusScreenStatus();

}

//\_PumpStatus.Short.print();

//waterHightSensorTest();

/\*

Serial.print("--");

Serial.println(waterHightRead());

delay(2000);

\*/

//turbidityTest();

//lcdHandleTouch();

//getTemperatureStatusResult(int 1 bool true);

//Serial.print(\_Lcd\_Status);

//test();

mainLogic(); //Automation

//tempratureSendSensorData();

}

void mainLogic(){

if(Serial.available()){

char c = Serial.read();

if(c = 's'){

\_NfossetStatus.print();

\_OfossetStatus.print();

\_PumpStatus.print();

\_DrainageStatus.print();

\_BatteryStatus.print();

\_WaterHightStatus.print();

\_TemperatureStatus.print();

\_HumidityStatus.print();

//\_TurbidityStatus.print();

}

}

espReconnect();

serverReconnect();

int h = waterHightRead();

if(h < MINIMUM\_WATER\_FOR\_PUMP){ //The water is bellow the minimum - pump stop and the fosset is open

pumpStop();

if (\_OFossetManualRequest == FossetManualRequestValue::FossetAutomatic) ofossetOpen(); //Manual control on the fosset - FossetManualRequestValue::FossetAutomatic

}

else if(h >= MINIMUM\_WATER\_FOR\_PUMP){ // The water is above the minimum ammount to start the pump again.

if (\_PumpManualRequest == PumpManualRequestValue::PumpAutomatic) pumpStart(); //Manual control on the pump - PumpManualRequestValue::PumpAutomatic)

}

if(h >= MAXIMUM\_WATER\_IN\_TANK){ //The water levle has reached the maximum, the fosset is closing (the pump keeps working)

ofossetClose();

}

//Cheeks if the pump is really running when he gets the command to start.//

if(pumpIsSignalOn() && !pumpIsCurrentOn()){

pumpStop();

}

/\*

float v = batteryGetVoltage();

float p = batteryGetPercent();

if(v < MINIMUM\_VOLTAGE\_FOR\_WARNING){

pumpStop();

ofossetClose();

nfossetClose();

drainageClose();

}

//check battery volateg

float b = batteryGetVoltage();

if(b < MINIMUM\_VOLTAGE\_FOR\_ERROR){

notifyLowBatteryError(b);

}

else(b < MINIMUM\_VOLTAGE\_FOR\_WARNING){

pumpStop();

ofossetClose();

drainageClose();

notifyLowBatteryWarning(b);

}

\*/

}

void test()

{

// TEMPERATURETest();

// ofossetTest();

// nfossetTest();

//

// pumpTest();

// drainageTest();

// turbidityTest();

// waterHightTest();

// pumpCurrentTest();

// lcdTest();

// espTest();

}

void checkStatus(){

nfossetCheckStatus();

ofossetCheckStatus();

pumpCheckStatus();

drainageCheckStatus();

batteryCheckStatus();

temperatureCheckStatus();

humidityCheckStatus();

lcdHandleTouch();

waterHightCheckStatus();

}

void serialInit(){

Serial.begin(9600);

}

## Battery

void batteryInit(){

pinMode(BATTERY\_VOLTAGE\_TEST\_PIN, INPUT);

}

void batteryTest(){

}

float batteryGetVoltage(){

int d = analogRead(BATTERY\_VOLTAGE\_TEST\_PIN);

return map(d, 0, MAX\_VALUE\_FOR\_VOLTAGE\_TEST, 0, MAX\_BATTERY\_VOLTAGE);

}

float batteryGetPercent(){

int d = analogRead(BATTERY\_VOLTAGE\_TEST\_PIN);

return map(d, 0, MAX\_VALUE\_FOR\_VOLTAGE\_TEST, 0, 100);

}

void batteryCheckStatus(){

int batteryVoltage = batteryGetVoltage();

int batteryPercent = batteryGetPercent();

String percent = String(batteryPercent);

char buffer[80];

if(batteryVoltage < MINIMUM\_VOLTAGE\_FOR\_ERROR){

sprintf(buffer, "Error, %dv(%d\%) battery levle is too low", batteryVoltage, batteryPercent);

\_BatteryStatus.Voltage = StatusResult(String(buffer), STATUS\_ERROR, PRIORITY\_HIGH);

sprintf(buffer, "%dv(%d%%)", batteryVoltage, batteryPercent);

\_BatteryStatus.Short = StatusResult(String(buffer), STATUS\_ERROR, PRIORITY\_HIGH);

}

else if(batteryVoltage < MINIMUM\_VOLTAGE\_FOR\_WARNING){

sprintf(buffer, "Warning, %dv(%d\%) battery levle is low", batteryVoltage, batteryPercent);

\_BatteryStatus.Voltage = StatusResult(String(buffer), STATUS\_WARNING, PRIORITY\_MEDIUM);

sprintf(buffer, "%dv(%d%%)", batteryVoltage, batteryPercent);

\_BatteryStatus.Short = StatusResult(String(buffer), STATUS\_WARNING, PRIORITY\_MEDIUM);

}

else{ //Battery levle is normal

sprintf(buffer, "%dv(%d%%)", batteryVoltage, batteryPercent);

\_BatteryStatus.Voltage = StatusResult(String(buffer), STATUS\_OK, PRIORITY\_LOW);

\_BatteryStatus.Short = StatusResult(String(buffer), STATUS\_OK, PRIORITY\_LOW);

}

}

## Drainage

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

/// DRAINAGE ///

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

bool \_DrainageIsOpen;

void drainageTest(){

drainageOpen();

delay(500);

drainageClose();

delay(500);

}

void drainageInit(){

pinMode(DRAINAGE\_PIN, OUTPUT);

drainageClose();

}

void drainageOpen(){

digitalWrite(DRAINAGE\_PIN, HIGH);

\_DrainageIsOpen = true;

}

void drainageClose(){

digitalWrite(DRAINAGE\_PIN, LOW);

\_DrainageIsOpen = false;

}

bool drainageIsOpen(){

return \_DrainageIsOpen;

}

void drainageManualControl(DrainageManualRequestValue value){

switch (value)

{

case DrainageManualRequestValue::DrainageClose:{

\_DrainageManualRequest = value;

drainageClose();

break;

}

case DrainageManualRequestValue::DrainageOpen:{

\_DrainageManualRequest = value;

drainageOpen();

break;

}

case DrainageManualRequestValue::DrainageAutomatic:{

\_DrainageManualRequest = value;

break;

}

}

}

void drainageCheckStatus(){

bool drainageSignal = drainageIsOpen();

if(DrainageManualRequestValue::DrainageOpen){

if(drainageSignal){

\_DrainageStatus.SignalOn = StatusResult("Open (manual)", STATUS\_OK, PRIORITY\_LOW);

\_DrainageStatus.Short = StatusResult("Open", STATUS\_OK, PRIORITY\_LOW);

}

else{

\_DrainageStatus.SignalOn = StatusResult("Error, signal off while manual request is on", STATUS\_ERROR, PRIORITY\_HIGH);

\_DrainageStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);

}

}

else if(DrainageManualRequestValue::DrainageClose){

if(drainageSignal){

\_DrainageStatus.SignalOn = StatusResult("Error, signal on while manual request is off", STATUS\_ERROR, PRIORITY\_HIGH);

\_DrainageStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);

}

else{

\_DrainageStatus.SignalOn = StatusResult("Close (manual)", STATUS\_OK, PRIORITY\_LOW);

\_DrainageStatus.Short = StatusResult("Close", STATUS\_OK, PRIORITY\_LOW);

}

}

else{ //Automatic

if(drainageSignal){

\_DrainageStatus.SignalOn = StatusResult("Open (automatic)", STATUS\_OK, PRIORITY\_LOW);

\_DrainageStatus.Short = StatusResult("Open", STATUS\_OK, PRIORITY\_LOW);

}

else{

\_DrainageStatus.SignalOn = StatusResult("Close (automatic)", STATUS\_OK, PRIORITY\_LOW);

\_DrainageStatus.Short = StatusResult("Close", STATUS\_OK, PRIORITY\_LOW);

}

}

}

## Espwifi

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

/// ESP-WIFI NETWORK ///

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

#define NO\_DATA "!@@#@@!" //used for ReadFromESP when there is no data yet

//used in the espReconnect for the \_WifiConnectionStatus - Updating the connection status of the ESP.

#define CONNECTION\_STATUS\_WAITING 0

#define CONNECTION\_STATUS\_CONNECTING 1

#define CONNECTION\_STATUS\_CONNECTED 2

#define CONNECTION\_STATUS\_TIMEOUT 3

#define CONNECTION\_STATUS\_ERROR 4

#define CONNECTION\_STATUS\_MISSING\_INFO 5

#define ESP\_TIME\_TO\_RECONNECT 30000

#define END\_OF\_RETRUN\_STRING '\0'

#define END\_OF\_COMMAND\_STRING '$'

long \_LastTimeConnectRequest;

void espInit(){

Serial3.begin(9600);

}

void handleEspRead(){

String content;

content = readFromEsp();

if (content != NO\_DATA)

Serial.println(content);

int index = content.indexOf("CONNECTION\_STATUS:");

if(index == -1){

content.substring(index + 18, content.lenght() - 1);

\_WifiConnectionStatus = millis;

switch (content.substring(index + 18, content.lenght() - 1)){

case "WAITING":

\_WifiConnectionStatus = CONNECTION\_STATUS\_WAITING;

\_WifiConnectionStatusTime = millis;

break;

case "CONNECTING":

\_WifiConnectionStatus = CONNECTION\_STATUS\_CONNECTING;

\_WifiConnectionStatusTime = millis;

break;

case "CONNECTED":

\_WifiConnectionStatus = CONNECTION\_STATUS\_CONNECTED;

\_WifiConnectionStatusTime = millis;

break;

case "TIMEOUT":

\_WifiConnectionStatus = CONNECTION\_STATUS\_TIMEOUT;

\_WifiConnectionStatusTime = millis;

break;

case "ERROR":

\_WifiConnectionStatus = CONNECTION\_STATUS\_ERROR;

\_WifiConnectionStatusTime = millis;

break;

case "MISSING\_INFO":

\_WifiConnectionStatus = CONNECTION\_STATUS\_MISSING\_INFO;

\_WifiConnectionStatusTime = millis;

break;

}

}

}

void espCheckConnection(){

if(millis - \_LastConnectionTimestamp <

}

void espTest(){

Serial.print("This is a test");

espSetDebugMode(true);

ConnectToWifi();

String content;

do {

content = readFromEsp();

if (content != NO\_DATA)

Serial.println(content);

}

while(content.indexOf("CONNECTION\_STATUS:CONNECTED") == -1);

byte sensorId = 4;

long timeStamp = 0;

int value = 6900;

// serverSendSensorData(sensorId, timeStamp, value);

espSendGetRequest("http://hamama.ew.r.appspot.com/board?cmd=measure&sid=4&time=0&value=6900"); //TODO Make a get request for the server on espSendGetRequest...

do{

content = readFromEsp();

if (content != NO\_DATA)

Serial.println(content);

}

while(content.indexOf("RESAULT\_STATUS:") == -1);

}

byte \_WifiConnectionStatus;

long \_WifiConnectionStatusTime;

void espSetDebugMode(bool mode){

if(mode == true){

Serial3.print("~b");

}

else{

Serial3.print("~e");

}

}

void espReconnect(){

switch(\_WifiConnectionStatus){

case CONNECTION\_STATUS\_CONNECTED:

return;

case CONNECTION\_STATUS\_WAITING:

ConnectToWifi();

case CONNECTION\_STATUS\_CONNECTING:

return;

case CONNECTION\_STATUS\_TIMEOUT:

case CONNECTION\_STATUS\_ERROR:

if(millis() - \_LastTimeConnectRequest > ESP\_TIME\_TO\_RECONNECT){

ConnectToWifi();

}

else{

return;

}

case CONNECTION\_STATUS\_MISSING\_INFO:

return;

}

}

String \_SerialReadBuffer;

String readFromEsp(){

if(Serial3.available()){

String a1 = Serial3.readString();

Serial.print(a1);

int index = a1.indexOf(END\_OF\_RETRUN\_STRING);

if(index >= 0 ){

String temp = a1.substring(0, index) + \_SerialReadBuffer;

\_SerialReadBuffer = a1.substring(index,a1.length());

return temp;

}

else{

\_SerialReadBuffer = \_SerialReadBuffer + a1;

return NO\_DATA;

}

}

return NO\_DATA;

}

void ConnectToWifi(){

String ssid;//[EEPROM\_ESP\_SSID\_LENGTH];

String password;//[EEPROM\_ESP\_PASSWORD\_LENGHT];

ssid = espGetWifiSSID();

password = espGetWifiPassword();

int timeout = espGetConnectionTimeout();

espConnectToWifi(ssid, password, timeout);

}

void espConnectToWifi(String ssid, String password, int connectionTimeout){

//~c<SSID>$<password>$<timeout>$

/\* Serial.print("~c");

Serial.print(ssid);

Serial.print(END\_OF\_COMMAND\_STRING);

Serial.print(password);

Serial.print(END\_OF\_COMMAND\_STRING);

Serial.print(connectionTimeout);

Serial.print(END\_OF\_COMMAND\_STRING);

\*/

Serial3.print("~c");

Serial3.print(ssid);

Serial3.print(END\_OF\_COMMAND\_STRING);

Serial3.print(password);

Serial3.print(END\_OF\_COMMAND\_STRING);

Serial3.print(connectionTimeout);

Serial3.print(END\_OF\_COMMAND\_STRING);

\_LastTimeConnectRequest = millis();

}

void espDisconnectFromWifi(){

Serial3.print("~d");

}

void espGetIp(){

Serial3.print("~a");

}

void espSendGetRequest(String url){

//~g<request url>$

Serial3.print("~g");

Serial3.print(url);

Serial3.print(END\_OF\_COMMAND\_STRING);

}

void espSendPostRequest(long reqid, String url, String data){

//~p<request url>$<data>$

Serial3.print("~p");

Serial3.print(reqid);

Serial3.print(END\_OF\_COMMAND\_STRING);

Serial3.print(url);

Serial3.print(END\_OF\_COMMAND\_STRING);

Serial3.print(data);

Serial3.print(END\_OF\_COMMAND\_STRING);

}

void espSetWifiSSID(String value){

eepromWrite(EEPROM\_ESP\_SSID,value);

}

void espSetWifiPassword(String value){

eepromWrite(EEPROM\_ESP\_PASSWORD,value);

}

void espSetConnectionTimeout(int value){

eepromWrite(EEPROM\_ESP\_CONNECTION\_TIMEOUT,value);

}

String espGetWifiSSID(){

return "Sulam";

//strcpy(buffer, "NAVE");

}

String espGetWifiPassword(){

return "2508572511";

//strcpy(buffer, "0524424546");

}

int espGetConnectionTimeout(){

return "45000";

}

## Hydroponic.h

struct StatusResult{

String StatusText;

byte Status; //STATUS\_OK, STATUS\_WARNING, STATUS\_ERROR

byte Priorty; //PRIORITY\_HIGH, PRIORITY\_MEDIUM, PRIORITY\_LOW

bool Sent;

StatusResult()

{

StatusText = "";

Status = 0;

Priorty = 0;

}

StatusResult(String statusText, int status, int priorty)

{

StatusText = statusText;

Priorty = priorty;

Status = status;

}

void print(){

Serial.print(StatusText);

Serial.print(" priority:");

Serial.print(Priorty);

Serial.print(" Status:");

Serial.println(Status);

}

};

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

/// Normal Fosset Status ///

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

class nfossetStatus{

public:

StatusResult WaterFlowing;

StatusResult CurrentOn;

StatusResult SignalOn;

StatusResult CanRun;

StatusResult SuposeToRun;

StatusResult Short;

void print(){

Serial.println("------------------");

Serial.println("Normal Fosset");

Serial.print("WaterFlowing:");

WaterFlowing.print();

Serial.print("CurrentOn:");

CurrentOn.print();

Serial.print("SignalOn:");

SignalOn.print();

Serial.print("CanRun:");

CanRun.print();

Serial.print("SuposeToRun:");

SuposeToRun.print();

Serial.println("------------------");

Serial.println();

}

};

nfossetStatus \_NfossetStatus;

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

/// Osmotic Fosset Status ///

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

class ofossetStatus{

public:

StatusResult WaterFlowing;

StatusResult CurrentOn;

StatusResult SignalOn;

StatusResult CanRun;

StatusResult SuposeToRun;

StatusResult Short;

void print(){

Serial.println("------------------");

Serial.println("Osmotic Fosset");

Serial.print("WaterFlowing:");

WaterFlowing.print();

Serial.print("CurrentOn:");

CurrentOn.print();

Serial.print("SignalOn:");

SignalOn.print();

Serial.print("CanRun:");

CanRun.print();

Serial.print("SuposeToRun:");

SuposeToRun.print();

Serial.println("------------------");

Serial.println();

}

};

ofossetStatus \_OfossetStatus;

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

/// Pump Status ///

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

class PumpStatus{

public:

StatusResult WaterFlowing;

StatusResult CurrentOn;

StatusResult SignalOn;

StatusResult CanRun;

StatusResult SuposeToRun;

StatusResult Short;

void print(){

Serial.println("------------------");

Serial.println("Pump");

Serial.print("WaterFlowing:");

WaterFlowing.print();

Serial.print("CurrentOn:");

CurrentOn.print();

Serial.print("SignalOn:");

SignalOn.print();

Serial.print("CanRun:");

CanRun.print();

Serial.print("SuposeToRun:");

SuposeToRun.print();

Serial.println("------------------");

Serial.println();

}

};

PumpStatus \_PumpStatus;

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

/// Drainage Status ///

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

class DrainageStatus{

public:

StatusResult SignalOn;

StatusResult Short;

void print(){

Serial.println("------------------");

Serial.println("Drainage");

Serial.print("SignalOn:");

SignalOn.print();

Serial.println("------------------");

Serial.println();

}

};

DrainageStatus \_DrainageStatus;

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

/// Battery Status ///

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

class BatteryStatus{

public:

StatusResult Voltage;

StatusResult Short;

void print(){

Serial.println("------------------");

Serial.println("Battery");

Serial.print("Voltage:");

Voltage.print();

Serial.println("------------------");

Serial.println();

}

};

BatteryStatus \_BatteryStatus;

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

/// Water Hight Status ///

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

class WaterHightStatus{

public:

StatusResult WaterLevle;

StatusResult Short;

void print(){

Serial.println("------------------");

Serial.println("Water Hight");

Serial.print("WaterLevle:");

WaterLevle.print();

Serial.println("------------------");

Serial.println();

}

};

WaterHightStatus \_WaterHightStatus;

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

/// Temperature Status ///

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

class TemperatureStatus{

public:

StatusResult Temperature1;

StatusResult Temperature2;

StatusResult Temperature3;

StatusResult Short;

void print(){

Serial.println("------------------");

Serial.println("Temperature");

Serial.print("Temperature1:");

Temperature1.print();

Serial.print("Temperature2:");

Temperature2.print();

Serial.print("Temperature3:");

Temperature3.print();

Serial.println("------------------");

Serial.println();

}

};

TemperatureStatus \_TemperatureStatus;

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

/// Humidity Status ///

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

class HumidityStatus{

public:

StatusResult Humidity1;

StatusResult Humidity2;

StatusResult Humidity3;

StatusResult Short;

void print(){

Serial.println("------------------");

Serial.println("Temperature");

Serial.print("Humidity1:");

Humidity1.print();

Serial.print("Humidity2:");

Humidity2.print();

Serial.print("Humidity3:");

Humidity3.print();

Serial.println("------------------");

Serial.println();

}

};

HumidityStatus \_HumidityStatus;

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

/// Turbidity Status ///

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

class TurbidityStatus{

public:

StatusResult IsWorking;

StatusResult Short;

};

TurbidityStatus \_TurbidityStatus;

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

// function declarations //

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

void batteryInit();

void batteryTest();

float batteryGetVoltage();

void serialInit();

void test();

void turbidityTest();

void turbidityInit();

int turbidityReadPercent();

bool tubidityIsWorking();

void drainageTest();

void drainageInit();

void drainageOpen();

void drainageClose();

void espInit();

void espTest();

void espNotifyError(int errorID, String error, int priorty);

void lcdInit();

void lcdTest();

void lcdNotifyError(String error);

void lcdShowState();

void lcdShowHomeScreen();

void lcdShowHomeScreenStatus();

void lcdShowStatusScreen();

void lcdShowSetupScreen();

void lcdHandleTouch();

void nfossetTest();

void nfossetInit();

void nfossetOpen();

void nfossetClose();

void nfossetManualControl();

bool nfossetCanRun();

void nfossetFlowInit();

void nfossetFlowTest();

bool nfossetIsWaterFlowing();

void nfossetCurrentInit();

void nfossetCurrentTest();

bool nfossetIsRunning();

int \_NfossetSignalOn;

void notifyPumpError();

void notifyDrainageError();

void notifyLowBatteryWarning(float voltage);

void notifyLowBatteryError(float voltage);

void showBatteryPrecent(byte percent);

void ofossetTest();

void ofossetInit();

void ofossetOpen();

void ofossetClose();

void ofossetManualControl();

bool ofossetCanRun();

void ofossetFlowInit();

void ofossetFlowTest();

bool ofossetIsWaterFlowing();

void ofossetCurrentInit();

void ofossetCurrentTest();

bool ofossetIsRunning();

void pumpTest();

void pumpInit();

void pumpStart();

void pumpStop();

void pumpManualControl();

void pumpCurrentInit();

void pumpCurrentTest();

bool pumpIsRunning();

void solenoidInit();

void solenoidTest();

void solenoidOpen();

void solenoidClose();

void waterHightTest();

void waterHightInit();

int waterHightRead();

void espInit();

void waterHightSensorInit();

void waterHightSensorTest();

## lcd screen

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

/// LCD SCREEN ///

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

#include "ili9488.h"

#include "ili9488\_touch.h"

void lcdInit(){

lcd.Init\_LCD();

lcdT.TP\_Init();

lcd.Fill\_Screen(WHITE);

}

void lcdTest(){

lcdShowHomeScreen();

lcdShowHomeScreenStatus();

}

void lcdNotifyError(String error){

}

void lcdShowState(){ //All the lcd screen stuff (idk).

}

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

//-Home Screen-//

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

void lcdShowHomeScreen(){

if (\_Lcd\_Status == LCD\_PAGE\_HOME){

return;

}

\_Lcd\_Status = LCD\_PAGE\_HOME;

lcd.Fill\_Screen(WHITE);

lcd.Set\_Rotation(1);

lcd.Set\_Text\_Size(4);

lcd.Set\_Text\_colour(BLACK);

lcd.Set\_Text\_Back\_colour(BLACK);

lcd.Print\_String("Home Screen", 105, 4);

lcd.Set\_Draw\_color(BLACK);

lcd.Draw\_Fast\_HLine(105,35,265); //make a line on the screen

lcd.Set\_Text\_Size(2);

lcd.Print\_String("Status", 5, 45);

lcd.Draw\_Rectangle(5,60,230,315);

lcd.Print\_String("Nfosset:", 10, 65);

lcd.Print\_String("Ofosset:", 10, 85);

lcd.Print\_String("Pump:", 10, 105);

lcd.Print\_String("Drainage:", 10, 125);

lcd.Print\_String("Battery:", 10, 145);

lcd.Print\_String("Water:", 10, 165);

lcd.Print\_String("Temprature:", 10, 185);

lcd.Print\_String("Humidity:", 10, 205);

lcd.Print\_String("Turbidity:", 10, 225);

lcd.Print\_String("WI-FI:", 10, 245);

lcd.Print\_String("Server:", 10, 265);

lcd.Print\_String("Light:", 10, 285);

//Buttons

lcd.Set\_Text\_Size(4);

lcd.Draw\_Rectangle(240,60,470,130);

lcd.Print\_String("Status", 280,83);

lcd.Draw\_Rectangle(240,140,470,210);

lcd.Print\_String("Config", 280,160);

lcd.Draw\_Rectangle(240,220,470,290);

lcd.Print\_String("About", 280,240);

lcd.Set\_Text\_Size(2);

}

void lcdShowHomeScreenStatus(){ //Status text blinking fix this TO:DO

lcd.Print\_String(" ", 105, 65);

lcdWriteStatus(105, 65, \_NfossetStatus.Short);

lcd.Print\_String(" ", 105, 85);

lcdWriteStatus(105, 85, \_OfossetStatus.Short);

lcd.Print\_String(" ", 75, 105);

lcdWriteStatus(75, 105, \_PumpStatus.Short);

lcd.Print\_String(" ", 120, 125);

lcdWriteStatus(120, 125, \_DrainageStatus.Short);

lcd.Print\_String(" ", 105, 145);

lcdWriteStatus(105, 145, \_BatteryStatus.Short);

lcd.Print\_String(" ", 105, 165);

lcdWriteStatus(105, 165, \_WaterHightStatus.Short);

lcd.Print\_String(" ", 140, 195);

lcdWriteStatus(140, 185, \_TemperatureStatus.Short);

lcd.Print\_String(" ", 115, 205);

lcdWriteStatus(115, 205, \_HumidityStatus.Short);

lcd.Print\_String(" ", 105, 225);

//lcdWriteStatus(105, 225, ); Convert turbidity to statusresult (not working right now)

}

void lcdHandleTouch(){

if(lcdT.TP\_Scan()){

int x = lcdT.getX();

int y = lcdT.getY();

/// Buttons on the home screen ///

if(\_Lcd\_Status == LCD\_PAGE\_HOME){

if(x >= 240 && x <= 470 && y >= 60 && y <= 130){ //Status Screen button 240,60,470,130

//Serial.println("Status");

lcdShowStatusScreen();

\_Lcd\_Status = LCD\_PAGE\_STATUS;

}

if(x >= 240 && x <= 470 && y >= 140 && y <= 210){ //Configuration Screen Button 240,140,470,210

//Serial.println("Config");

lcdShowConfigScreen();

\_Lcd\_Status = LCD\_PAGE\_CONFIG;

}

if(x >= 240 && x <= 470 && y >= 220 && y <= 290){ //About Screen Button 240,220,470,290

//Serial.println("About");

lcdShowAboutScreen();

\_Lcd\_Status = LCD\_PAGE\_ABOUT;

}

}

/// Buttons on the status screen ///

if(\_Lcd\_Status == LCD\_PAGE\_STATUS){

//\*back to home and back to status buttons configuration

if(\_Lcd\_Status\_Subpage == STATUS\_SUBPAGE\_MAIN){

if(x >= 4 && x <= 60 && y >= 5 && y <= 30){ //Back to home button

//Serial.println("Back to home button");

lcdShowHomeScreen();

\_Lcd\_Status = LCD\_PAGE\_HOME;

}

}

else{

if(x >= 4 && x <= 60 && y >= 5 && y <= 30){ //Back to status button

//Serial.println("Back to home button");

lcdShowStatusScreen();

\_Lcd\_Status = LCD\_PAGE\_STATUS;

}

}

//\*Full status for sensors buttons

if(x >= 4 && x <= 60 && y >= 5 && y <= 30){ //Nfosset status button

}

}

/// Buttons on configuration screen ///

if(\_Lcd\_Status == LCD\_PAGE\_CONFIG){

if(x >= 4 && x <= 60 && y >= 5 && y <= 30){ //Status Screen button 240,60,470,130

//Serial.println("Back to home button");

lcdShowHomeScreen();

\_Lcd\_Status = LCD\_PAGE\_HOME;

}

}

/// Buttons on about screen ///

if(\_Lcd\_Status == LCD\_PAGE\_ABOUT){

if(x >= 4 && x <= 60 && y >= 5 && y <= 30){ //Status Screen button 240,60,470,130

//Serial.println("Back to home button");

lcdShowHomeScreen();

\_Lcd\_Status = LCD\_PAGE\_HOME;

}

}

}

}

void lcdShowStatusScreen(){

\_Lcd\_Status = LCD\_PAGE\_STATUS;

lcd.Fill\_Screen(WHITE);

lcd.Set\_Rotation(1);

lcd.Set\_Text\_colour(BLACK);

lcd.Set\_Text\_Back\_colour(BLACK);

//\*Back to home button

lcd.Set\_Text\_Size(2);

lcd.Draw\_Rectangle(4,5,60,30);

lcd.Print\_String("Home", 10,10);

lcd.Set\_Text\_Size(4);

lcd.Set\_Text\_colour(BLACK);

lcd.Set\_Text\_Back\_colour(BLACK);

lcd.Print\_String("Status Screen", 90, 4);

lcd.Set\_Draw\_color(BLACK);

lcd.Draw\_Fast\_HLine(90,35,310); //make a line on the screen

lcd.Set\_Text\_Size(2);

//Short status for now

lcd.Print\_String("Nfosset:", 10, 65);

lcd.Print\_String("Ofosset:", 10, 85);

lcd.Print\_String("Pump:", 10, 105);

lcd.Print\_String("Drainage:", 10, 125);

lcd.Print\_String("Battery:", 10, 145);

lcd.Print\_String("Water:", 10, 165);

lcd.Print\_String("Temprature:", 10, 185);

lcd.Print\_String("Humidity:", 10, 205);

lcd.Print\_String("Turbidity:", 10, 225);

lcd.Print\_String("WI-FI:", 10, 245);

lcd.Print\_String("Server:", 10, 265);

lcd.Print\_String("Light:", 10, 285);

/// Status buttons on the status screen ///

//\*Nfosset Button

/\*

lcd.Set\_Text\_Size(3);

lcd.Set\_Text\_colour(BLACK);

lcd.Set\_Text\_Back\_colour(BLACK);

lcd.Draw\_Rectangle(10,45,180,85);

lcd.Print\_String("Nfosset", 15, 55);

//\*Ofosset button

lcd.Set\_Text\_Size(3);

lcd.Set\_Text\_colour(BLACK);

lcd.Set\_Text\_Back\_colour(BLACK);

lcd.Draw\_Rectangle(10,65,180,125);

lcd.Print\_String("Ofosset", 15, 75);

//\*Pump Button

lcd.Set\_Text\_Size(3);

lcd.Set\_Text\_colour(BLACK);

lcd.Set\_Text\_Back\_colour(BLACK);

lcd.Draw\_Rectangle(10,65,180,125);

lcd.Print\_String("Pump", 15, 75);

//\*Drainage Button

lcd.Set\_Text\_Size(3);

lcd.Set\_Text\_colour(BLACK);

lcd.Set\_Text\_Back\_colour(BLACK);

lcd.Draw\_Rectangle(10,65,180,125);

lcd.Print\_String("Drainage", 15, 75);

\*/

}

void lcdShowStatusScreenStatus(){

lcd.Set\_Text\_Size(2);

lcd.Print\_String(" ", 105, 65);

lcdWriteStatus(105, 65, \_NfossetStatus.Short);

lcd.Print\_String(" ", 105, 85);

lcdWriteStatus(105, 85, \_OfossetStatus.Short);

lcd.Print\_String(" ", 75, 105);

lcdWriteStatus(75, 105, \_PumpStatus.Short);

lcd.Print\_String(" ", 120, 125);

lcdWriteStatus(120, 125, \_DrainageStatus.Short);

lcd.Print\_String(" ", 105, 145);

lcdWriteStatus(105, 145, \_BatteryStatus.Short);

lcd.Print\_String(" ", 105, 165);

lcdWriteStatus(105, 165, \_WaterHightStatus.Short);

lcd.Print\_String(" ", 140, 195);

lcdWriteStatus(140, 185, \_TemperatureStatus.Short);

lcd.Print\_String(" ", 115, 205);

lcdWriteStatus(115, 205, \_HumidityStatus.Short);

lcd.Print\_String(" ", 105, 225);

}

void lcdShowConfigScreen(){

\_Lcd\_Status = LCD\_PAGE\_STATUS;

lcd.Fill\_Screen(WHITE);

lcd.Set\_Text\_Size(4);

lcd.Set\_Text\_colour(BLACK);

lcd.Set\_Text\_Back\_colour(BLACK);

lcd.Print\_String("Configuration", 90, 4);

lcd.Set\_Draw\_color(BLACK);

lcd.Draw\_Fast\_HLine(90,35,310); //make a line on the screen

lcd.Set\_Text\_Size(3);

}

void lcdShowAboutScreen(){

\_Lcd\_Status = LCD\_PAGE\_STATUS;

lcd.Fill\_Screen(WHITE);

lcd.Set\_Text\_Size(4);

lcd.Set\_Text\_colour(BLACK);

lcd.Set\_Text\_Back\_colour(BLACK);

lcd.Print\_String("About", 90, 4);

lcd.Set\_Draw\_color(BLACK);

lcd.Draw\_Fast\_HLine(90,35,310); //make a line on the screen

lcd.Set\_Text\_Size(3);

}

void lcdWriteStatus(int x, int y, StatusResult res){

lcd.Set\_Text\_Back\_colour(WHITE);

//Serial.println(res.StatusText);

lcdChangeColor(res.Status);

print(x, y, res.StatusText);

}

void lcdChangeColor(int status){

switch (status)

{

case STATUS\_OK:

lcd.Set\_Text\_colour(STATUS\_OK\_COLOR);

//Serial.print("Ok");

break;

case STATUS\_WARNING:

lcd.Set\_Text\_colour(STATUS\_WARNING\_COLOR);

//Serial.print("Warning");

break;

case STATUS\_ERROR:

lcd.Set\_Text\_colour(STATUS\_ERROR\_COLOR);

//Serial.print("Error");

break;

default:

lcd.Set\_Text\_colour(STATUS\_NORMAL\_COLOR);

//Serial.print("Normal");

break;

}

}

void print(int x, int y, String text){

lcd.Print\_String(text, x, y);

}

## Nfosset

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

/// NORMAL FOSSET /// - Water in...

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

/\*

class nfossetStatus{

public:

StatusResult WaterFlowing;

StatusResult CurrentOn;

StatusResult SignalOn;

StatusResult CanRun;

StatusResult SuposeToRun;

StatusResult Short;

String getShortString(){

}

};

nfossetStatus \_NfossetStatus;

\*/

void nfossetTest(){

nfossetOpen();

delay(500);

nfossetClose();

delay(500);

}

void nfossetInit(){

pinMode(NFOSSET\_PIN, OUTPUT);

}

void nfossetOpen(){

digitalWrite(NFOSSET\_PIN, HIGH);

\_NfossetSignalOn = 1;

}

void nfossetClose(){

digitalWrite(NFOSSET\_PIN, LOW);

\_NfossetSignalOn = 0;

}

void nfossetManualControl(FossetManualRequestValue value){

switch (value)

{

case FossetManualRequestValue::FossetClose:{

\_NFossetManualRequest = value;

nfossetClose();

break;

}

case FossetManualRequestValue::FossetOpen:{

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;

}

}

}

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;

}

//int \_NfossetSignalOn;

bool nfossetIsSignalOn(){

if(\_NfossetSignalOn == 1){

return true;

}

else

return false;

}

/\*

StatusResult nfossetStatus\_Short(){ //// Short Status - old code ////

bool nfossetFlowing = nfossetIsWaterFlowing();// Normal fosset

bool nfossetCanRunVar = nfossetCanRun();

switch (\_NFossetManualRequest) {

case FossetManualRequestValue::FossetAutomatic:

{

if(nfossetFlowing == true)

{

if(nfossetCanRunVar == true)

return StatusResult ("On", STATUS\_OK);

else

return StatusResult ("Error", STATUS\_ERROR);

}

else

{

return StatusResult ("Off", STATUS\_OK);

}

break;

}

case FossetManualRequestValue::Open:

{

if(nfossetFlowing == true)

return StatusResult ("On", STATUS\_OK);

else

return StatusResult ("Error", STATUS\_ERROR);

break;

}

case FossetManualRequestValue::Close:

if(nfossetFlowing == false)

return StatusResult ("Off", STATUS\_OK);

else

return StatusResult ("Error", STATUS\_ERROR);

break;

default:

return StatusResult ("Unknown", STATUS\_WARNING);

}

}

\*/

bool nfossetCanRunVar = nfossetCanRun();

bool nfossetSuposeToRun(){

return ((\_NFossetManualRequest == FossetManualRequestValue::FossetAutomatic && nfossetCanRunVar) || \_NFossetManualRequest == FossetManualRequestValue::FossetOpen);//complete by the enum //c//

}

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);

}

}

}

}

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

/// NORMAL FOSSET FLOW SENSOR ///

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

void nfossetFlowInit(){

pinMode(NFOSSET\_FLOW\_TEST\_PIN, INPUT);

}

void nfossetFlowTest(){

//TODO

}

bool nfossetIsWaterFlowing(){ //Not completed yet (for now its on while the current and the signal is on)

if(nfossetIsCurrentOn() && nfossetIsSignalOn()){

return true;

}

else{

return false;

}

}

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

/// NORMAL FOSST CURRENT TEST ///

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

void nfossetCurrentInit(){

pinMode(NFOSSET\_CURRENT\_TEST\_PIN, INPUT);

}

void nfossetCurrentTest(){

Serial.print("nfosset current:");

Serial.println(analogRead(NFOSSET\_CURRENT\_TEST\_PIN));

}

bool nfossetIsCurrentOn(){

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.

}

## Ofosset

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

/// OSMOTIC FOSSET /// - Water in...

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

int \_OfossetSignalOn;

void ofossetTest(){

ofossetOpen();

delay(500);

ofossetClose();

delay(500);

}

void ofossetInit(){

pinMode(OFOSSET\_PIN, OUTPUT);

}

void ofossetOpen(){

digitalWrite(OFOSSET\_PIN, HIGH);

\_OfossetSignalOn = 1;

}

void ofossetClose(){

digitalWrite(OFOSSET\_PIN, LOW);

\_OfossetSignalOn = 0;

}

void ofossetManualControl(FossetManualRequestValue value){

switch (value)

{

case FossetManualRequestValue::FossetClose:{

\_OFossetManualRequest = value;

ofossetClose();

break;

}

case FossetManualRequestValue::FossetOpen:{

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;

}

}

}

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;

}

bool ofossetIsSignalOn(){

if(\_OfossetSignalOn == 1){

return true;

}

else

return false;

}

bool ofossetSuposeToRun(){

return ((\_OFossetManualRequest == FossetManualRequestValue::FossetAutomatic && ofossetCanRun()) || \_OFossetManualRequest == FossetManualRequestValue::FossetOpen); //complete by the enum

}

void ofossetCheckStatus(){

bool ofossetWaterFlowing = ofossetIsWaterFlowing();

bool ofossetCurrentOn = ofossetIsCurrentOn();

bool ofossetSignalOn = ofossetIsSignalOn();

bool ofossetSuposeToRunVar = ofossetSuposeToRun();

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(ofossetSignalOn){

\_OfossetStatus.SignalOn = StatusResult("Error: Fosset signal pin on while water levle too high", STATUS\_ERROR, PRIORITY\_HIGH);

\_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);

}

else{

\_OfossetStatus.SignalOn = StatusResult("Off: Fosset signal off", STATUS\_OK, PRIORITY\_LOW);

\_OfossetStatus.Short = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);

}

}

else{

if(ofossetSuposeToRun){

if(!ofossetSignalOn){

if(h > MINIMUM\_WATER\_FOR\_PUMP){

\_OfossetStatus.SignalOn = StatusResult("Warning: Fosset signal pin off while water levle is low", STATUS\_WARNING, PRIORITY\_MEDIUM);

\_OfossetStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_MEDIUM);

}

else{

\_OfossetStatus.SignalOn = StatusResult("Error: Fosset signal pin off while water levle is under pump minimum", STATUS\_ERROR, PRIORITY\_HIGH);

\_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);

}

}

else{

\_OfossetStatus.SignalOn = StatusResult("On: Fosset signal pin on", STATUS\_OK, PRIORITY\_LOW);

\_OfossetStatus.Short = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);

}

}

}

if(ofossetSignalOn){

if(!ofossetWaterFlowing){

if(h > MINIMUM\_WATER\_FOR\_PUMP){

\_OfossetStatus.WaterFlowing = StatusResult("Warning: Water not flowing while signal pin is on", STATUS\_WARNING, PRIORITY\_MEDIUM);

\_OfossetStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_MEDIUM);

}

else{

\_OfossetStatus.WaterFlowing = StatusResult("Error: Water not flowing while water levle is under pump minimum", STATUS\_ERROR, PRIORITY\_HIGH);

\_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);

}

}

else{

\_OfossetStatus.WaterFlowing = StatusResult("On: Water flowing", STATUS\_OK, PRIORITY\_LOW);

if(\_OfossetStatus.Short.Status == STATUS\_OK){

\_OfossetStatus.Short = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);

}

}

if(!ofossetCurrentOn){

if(h > MINIMUM\_WATER\_FOR\_PUMP){

\_OfossetStatus.CurrentOn = StatusResult("Warning: No current while signal pin is on", STATUS\_WARNING, PRIORITY\_MEDIUM);

if(\_OfossetStatus.Short.Status != STATUS\_ERROR){

\_OfossetStatus.Short = StatusResult("Warning", STATUS\_WARNING, PRIORITY\_MEDIUM);

}

}

else{

\_OfossetStatus.CurrentOn = StatusResult("Error: No current while water levle is under pump minimum", STATUS\_ERROR, PRIORITY\_HIGH);

\_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);

}

}

else{

\_OfossetStatus.CurrentOn = StatusResult("Current on", STATUS\_OK, PRIORITY\_LOW);

if(\_OfossetStatus.Short.Status == STATUS\_OK){

\_OfossetStatus.Short = StatusResult("On", STATUS\_OK, PRIORITY\_LOW);

}

}

}

else{

if(ofossetWaterFlowing){

\_OfossetStatus.WaterFlowing = StatusResult("Error: Water flowing while signal pin is off", STATUS\_ERROR, PRIORITY\_HIGH);

\_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);

}

else{

\_OfossetStatus.WaterFlowing = StatusResult("No water flowing", STATUS\_OK, PRIORITY\_LOW);

if(\_OfossetStatus.Short.Status == STATUS\_OK){

\_OfossetStatus.Short = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);

}

}

if(ofossetCurrentOn){

\_OfossetStatus.CurrentOn = StatusResult("Error: Fosset has current while signal pin is off", STATUS\_ERROR, PRIORITY\_HIGH);

\_OfossetStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);

}

else{

\_OfossetStatus.CurrentOn = StatusResult("No current", STATUS\_OK, PRIORITY\_LOW);

if(\_OfossetStatus.Short.Status == STATUS\_OK){

\_OfossetStatus.Short = StatusResult("Off", STATUS\_OK, PRIORITY\_LOW);

}

}

}

}

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

/// OSMOTIC FOSSET FLOW SENSOR ///

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

void ofossetFlowInit(){

pinMode(OFOSSET\_FLOW\_TEST\_PIN, INPUT);

}

void ofossetFlowTest(){

//TODO

}

bool ofossetIsWaterFlowing(){ //Not completed yet (for now its on while the current and the signal is on)

if(ofossetIsCurrentOn() && ofossetIsSignalOn()){

return true;

}

else{

return false;

}

}

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

/// OSMOTIC FOSSET CURRENT TEST ///

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

void ofossetCurrentInit(){

pinMode(OFOSSET\_CURRENT\_TEST\_PIN, INPUT);

}

void ofossetCurrentTest(){

//Serial.print("ofosset current:");

//Serial.println(ofossetCurrentRead());

}

bool ofossetIsCurrentOn(){

int c = analogRead(OFOSSET\_CURRENT\_TEST\_PIN);

return c > 0; //Cheeks if the current is higer than 0. If its higer than 0 its true if not false.

}

## Pump

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

/// PUMP /// //Liquids to the system...

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

void pumpTest(){

pumpStart();

delay(500);

pumpStop();

delay(500);

}

void pumpInit(){

pinMode(PUMP\_PIN, OUTPUT);

}

void pumpStart(){

digitalWrite(PUMP\_PIN, HIGH);

\_IsPumpRunning = 1;

}

void pumpStop(){

digitalWrite(PUMP\_PIN, LOW);

\_IsPumpRunning = 0;

}

void pumpManualControl(PumpManualRequestValue value){

switch (value)

{

case PumpManualRequestValue::PumpStart:{

int h = waterHightRead();

if(h > MINIMUM\_WATER\_FOR\_PUMP){

\_PumpManualRequest = value;

pumpStart(); //Pump can run

}

break;

}

case PumpManualRequestValue::PumpStop:{

\_PumpManualRequest = value;

pumpStop();

break;

}

case PumpManualRequestValue::PumpAutomatic:{

\_PumpManualRequest = value;

break;

}

}

}

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;

}

bool pumpIsSignalOn(){

if(\_IsPumpRunning == 1){

return true;

}

else{

return false;

}

}

bool pumpSuposeToRun(){

return((\_PumpManualRequest == PumpManualRequestValue::PumpAutomatic && pumpCanRun()) || \_PumpManualRequest == PumpManualRequestValue::PumpStart);

}

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);

}

}

}

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

/// PUMP CURRENT SENSOR ///

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

void pumpCurrentInit(){

pinMode(PUMP\_CURRENT\_TEST\_PIN, INPUT);

// pinMode( //no pin yet

}

void pumpCurrentTest(){

//Serial.print("Pump current:");

//Serial.println(pumpCurrentRead());

}

bool pumpIsCurrentOn(){

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.

}

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

/// PUMP FLOW SENSOR ///

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

void pumpFlowInit(){

pinMode(PUMP\_FLOW\_TEST\_PIN, INPUT);

}

void pumpFlowTest(){

//Not ready for now

}

bool pumpIsWaterFlowing(){ //Not completed yet (for now its on while the current and the signal is on)

if(pumpIsCurrentOn() && pumpIsSignalOn()){

return true;

}

else{

return false;

}

}

## Server

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

/// SERVER ///

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

//used in the serverReconnect for the \_ServerConnectionStatus - Cheek if the ESP is connected to the Server.

#define SERVER\_CONNECTION\_STATUS\_CONNECTED 1

#define SERVER\_CONNECTION\_STATUS\_DISCONNECTED 0

#define SERVER\_TIME\_TO\_RECONNECT 30000

//used to define the type of request from the Server.

#define SERVER\_REQUEST\_NONE 0

#define SERVER\_REQUEST\_PING 1

#define SERVER\_REQUEST\_SENSOR\_DATA 2

#define SERVER\_REQUEST\_LOG 3

long \_RequestID = 0;

long getRequestID()

{

long reqID = \_RequestID;

\_RequestID++;

return reqID;

}

void serverInit(){

}

void serverTest(){

//serverRandomTest();

}

byte \_ServerConnectionStatus;

long \_ServerConnectionStatusTime;

byte \_ServerRequestType;

long rtc\_GetTimeStamp()

{

return millis();

}

void serverSendResault(StatusResult stat, byte sensorId){

if (!stat.Sent && !stat.Status == STATUS\_OK){

long timeStamp = rtc\_GetTimeStamp();

serverSendLog(sensorId, timeStamp, stat.Priorty, stat.StatusText);

}

}

void serverSendLog(byte sensorId, long timeStamp, byte priority, String message){

\_ServerRequestType = SERVER\_REQUEST\_LOG;

String url = "http://"+serverGetAdress()+"/board?";

url += "cmd=log";

String data = "";

if(sensorId != 0){

data += "sid=" + sensorId;

data += "&";

}

data += "time=";

data += timeStamp;

data += "&";

data += "priority=";

data += priority;

data += "&";

data += "message=";

data += message;

espSendPostRequest(getRequestID(), url, data);

/// Testing with get request ///

//espSendGetRequest(url);

//url = url + "?" + data;

}

void serverSendSensorData(byte sensorId, long timeStamp, int value){

//TODO: add request to queue and send data from queue periodically. delete data from queue only after receiving confirmation that it arived on server.

\_ServerRequestType = SERVER\_REQUEST\_SENSOR\_DATA;

String url = "http://"+serverGetAdress()+"/board";

String data = "cmd=measure";

data += "&";

data += "sid=" + String(sensorId);

data += "&";

data += "time=";

data += String(timeStamp);

data += "&";

data += "value=";

data += String(value);

espSendPostRequest(getRequestID(), url, data);

///// Testing with get request /////

//url = url + "?" + data;

//espSendGetRequest(requestId, url);

//Serial.print(url);

//Serial.println(data);

}

void serverSendPing(){

\_ServerRequestType = SERVER\_REQUEST\_PING;

String url = "http://"+serverGetAdress()+"/board?";

url += "cmd=ping";

\_ServerConnectionStatusTime = millis();

}

void serverReconnect(){

if(\_WifiConnectionStatus!=CONNECTION\_STATUS\_CONNECTED){

return;

}

if(\_ServerConnectionStatus!=SERVER\_CONNECTION\_STATUS\_CONNECTED){

serverSendPing();

}

else if(millis() - \_ServerConnectionStatusTime > SERVER\_TIME\_TO\_RECONNECT){

serverSendPing();

}

}

String serverGetAdress(){

//TODO: Add a server adress read function, add server eeprom read function

return "hamama.ew.r.appspot.com";

}

String serverSetAdress(){

//Idk if i need that but if i am, to do add a eeprom write function for the server address.

}

void serverSetAdress(String value){

//TODO: save Adress to EEPROM memory

}

void serverNotifyError(int sensorId, int errorType, String errorMsg, String timeStamp){

\_ServerRequestType = SERVER\_REQUEST\_LOG;

String url = "http://"+serverGetAdress()+"/board?";

url += "cmd=log";

String data = "";

if(sensorId != 0){

data += "sid=" + sensorId;

data += "&";

}

data += "time=";

data += timeStamp;

data += "&";

data += "priority=";

data += PRIORITY\_HIGH;

data += "&";

data += "errorType=";

data += errorType;

data += "&";

data += "message=";

data += errorMsg;

espSendPostRequest(getRequestID(), url, data);

/// Testing with get request ///

//espSendGetRequest(url);

//url = url + "?" + data;

}

void serverRandomTest(){

Serial2.print("https://httpbin.org/status/200");

}

## Temperature

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

/// TEMPERATURE sensor ///

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

#define DHTTYPE DHT11

DHT\* \_TempHumiSensors[3];

void temperatureSensorInit(){

\_TempHumiSensors[0] = new DHT(TEMPERATURE\_SENSOR1\_PIN, DHTTYPE);

\_TempHumiSensors[1] = new DHT(TEMPERATURE\_SENSOR2\_PIN, DHTTYPE);

\_TempHumiSensors[2] = new DHT(TEMPERATURE\_SENSOR3\_PIN, DHTTYPE);

\_TempHumiSensors[0]->begin();

\_TempHumiSensors[1]->begin();

\_TempHumiSensors[2]->begin();

}

void temperatureTest(){

for(int i = 1; i <= 3; i++){

float humi = getHumidity(i);

float tempC = getTemperature(i);

Serial.println("Humidity: ");

Serial.print(humi);

Serial.print("%");

Serial.print(" | ");

Serial.print("Temperature: ");

Serial.print(tempC);

Serial.print("°C ~ ");

}

}

float getTemperature(byte sensorId){

return \_TempHumiSensors[sensorId-1]->readTemperature();

}

float getHumidity(byte sensorId){

return \_TempHumiSensors[sensorId-1]->readHumidity();

}

void temperatureCheckStatus(){

int temperature1 = getTemperature(1);

int temperature2 = getTemperature(2);

int temperature3 = getTemperature(3);

float avgTemperature = (temperature1 + temperature2 + temperature3) / 3;

String oldStatus1 = \_TemperatureStatus.Temperature1.StatusText;

bool oldSent1 = \_TemperatureStatus.Temperature1.Sent;

\_TemperatureStatus.Temperature1 = getTemperatureStatusResult(temperature1,false);

if (oldStatus1 != \_TemperatureStatus.Temperature1.StatusText)

\_TemperatureStatus.Temperature1.Sent = false;

else

\_TemperatureStatus.Temperature1.Sent = oldSent1;

String oldStatus2 = \_TemperatureStatus.Temperature2.StatusText;

bool oldSent2 = \_TemperatureStatus.Temperature2.Sent;

\_TemperatureStatus.Temperature2 = getTemperatureStatusResult(temperature2,false);

if (oldStatus2 != \_TemperatureStatus.Temperature2.StatusText)

\_TemperatureStatus.Temperature2.Sent = false;

else

\_TemperatureStatus.Temperature2.Sent = oldSent2;

String oldStatus3 = \_TemperatureStatus.Temperature3.StatusText;

bool oldSent3 = \_TemperatureStatus.Temperature3.Sent;

\_TemperatureStatus.Temperature3 = getTemperatureStatusResult(temperature3,false);

if (oldStatus3 != \_TemperatureStatus.Temperature3.StatusText)

\_TemperatureStatus.Temperature3.Sent = false;

else

\_TemperatureStatus.Temperature3.Sent = oldSent3;

//\_TemperatureStatus.Temperature2 = getTemperatureStatusResult(temperature2,false);

// \_TemperatureStatus.Temperature3 = getTemperatureStatusResult(temperature3,false);

\_TemperatureStatus.Short = getTemperatureStatusResult(avgTemperature, true);

}

StatusResult getTemperatureStatusResult(int value, bool shortStatus){

char buffer[80];

//Temperature printing not working -- fix this

if(shortStatus){

sprintf(buffer, "%dC", value);

}

if (value > MAXIMUM\_TEMPERATURE\_FOR\_ERROR){

if(!shortStatus) sprintf(buffer, "Error, Temperature %dC is too high", value);

return StatusResult(String(buffer), STATUS\_ERROR, PRIORITY\_HIGH);

}

else if(value > MAXIMUM\_TEMPERATURE\_FOR\_WARNING){

if(!shortStatus) sprintf(buffer, "Warning, Temperature %dC is high", value);

return StatusResult(String(buffer), STATUS\_WARNING, PRIORITY\_MEDIUM);

}

else if(value < MINIMUM\_TEMPERATURE\_FOR\_ERROR){

if(!shortStatus) sprintf(buffer, "Warning, Temperature %dC is high", value);

return StatusResult(String(buffer), STATUS\_ERROR, PRIORITY\_HIGH);

}

else if(value < MINIMUM\_TEMPERATURE\_FOR\_WARNING){

if(!shortStatus) sprintf(buffer, "Warning, Temperature %dC is low", value);

return StatusResult(String(buffer), STATUS\_WARNING, PRIORITY\_MEDIUM);

}

else{

return StatusResult(String(buffer), STATUS\_OK, PRIORITY\_LOW);

}

}

void humidityCheckStatus(){

int humidity1 = getHumidity(1);

int humidity2 = getHumidity(2);

int humidity3 = getHumidity(3);

int avgHumidity = (humidity1 + humidity2 + humidity3) / 3;

String oldStatus1 = \_HumidityStatus.Humidity1.StatusText;

bool oldSent1 = \_HumidityStatus.Humidity1.Sent;

\_HumidityStatus.Humidity1 = getHumidityStatusResult(humidity1,false);

if (oldStatus1 != \_HumidityStatus.Humidity1.StatusText)

\_HumidityStatus.Humidity1.Sent = false;

else

\_HumidityStatus.Humidity1.Sent = oldSent1;

String oldStatus2 = \_HumidityStatus.Humidity2.StatusText;

bool oldSent2 = \_HumidityStatus.Humidity2.Sent;

\_HumidityStatus.Humidity2 = getHumidityStatusResult(humidity2,false);

if (oldStatus2 != \_HumidityStatus.Humidity2.StatusText)

\_HumidityStatus.Humidity2.Sent = false;

else

\_HumidityStatus.Humidity2.Sent = oldSent2;

String oldStatus3 = \_HumidityStatus.Humidity3.StatusText;

bool oldSent3 = \_HumidityStatus.Humidity3.Sent;

\_HumidityStatus.Humidity3 = getHumidityStatusResult(humidity3,false);

if (oldStatus3 != \_HumidityStatus.Humidity3.StatusText)

\_HumidityStatus.Humidity3.Sent = false;

else

\_HumidityStatus.Humidity3.Sent = oldSent1;

//\_HumidityStatus.Humidity1 = getHumidityStatusResult(humidity1, false);

//\_HumidityStatus.Humidity2 = getHumidityStatusResult(humidity2, false);

//\_HumidityStatus.Humidity3 = getHumidityStatusResult(humidity3, false);

\_HumidityStatus.Short = getHumidityStatusResult(avgHumidity, true);

}

StatusResult getHumidityStatusResult(int value, bool shortStatus){

char buffer[80];

if(shortStatus) sprintf(buffer, "%d%", value);

if (value > MAXIMUM\_HUMIDITY\_FOR\_ERROR){

if(!shortStatus) sprintf(buffer, "Error, Humidity %d% is too high", value);

return StatusResult(String(buffer), STATUS\_ERROR, PRIORITY\_HIGH);

}

else if(value > MAXIMUM\_HUMIDITY\_FOR\_WARNING){

if(!shortStatus) sprintf(buffer, "Warning, Humidity %d% is high", value);

return StatusResult(String(buffer), STATUS\_WARNING, PRIORITY\_MEDIUM);

}

else if(value < MINIMUM\_HUMIDITY\_FOR\_ERROR){

if(!shortStatus) sprintf(buffer, "Warning, Humidity %d% is high", value);

return StatusResult(String(buffer), STATUS\_ERROR, PRIORITY\_HIGH);

}

else if(value < MINIMUM\_HUMIDITY\_FOR\_WARNING){

if(!shortStatus) sprintf(buffer, "Warning, Humidity %d% is low", value);

return StatusResult(String(buffer), STATUS\_WARNING, PRIORITY\_MEDIUM);

}

else{

return StatusResult(String(buffer), STATUS\_OK, PRIORITY\_LOW);

}

}

void tempratureSendSensorData(){

int temp1 = getTemperature(1);

int temp2 = getTemperature(2);

int temp3 = getTemperature(3);

serverSendSensorData(4, 0, temp1);

serverSendSensorData(5, 0, temp2);

serverSendSensorData(6, 0, temp3);

}

## Turbidity

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

/// TURBIDITY ///

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

void turbidityTest(){

int value = turbidityReadPercent();

Serial.println(value);

}

void turbidityInit()

{

pinMode(TURBIDITY\_PIN, INPUT);

}

int turbidityReadPercent(){

int val = analogRead(TURBIDITY\_PIN); //Return status in %

return map(val, 0, 821, 0, 100);

}

bool turbidityIsWorking(){

int val = analogRead(TURBIDITY\_PIN);

if(val != 0){

return true;

}

else{

return false;

}

}

void turbidityCheckStatus(){ //Finish the turbidity Check Status function today.

int turbidityPercent = turbidityReadPercent();

//bool turbidityIsWorking = turbidityIsWorking();

}

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";

}

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";

}

## Turbidity

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

/// WATER HIGHT ///

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

SoftwareSerial mySerial(12,11); // RX, TX

unsigned char data[4]={};

float distance;

void waterHightSensorInit(){

Serial.begin(57600);

mySerial.begin(9600);

}

void waterHightSensorTest(){

do{

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

{

data[i]=mySerial.read();

}

}while(mySerial.read()==0xff);

mySerial.flush();

if(data[0]==0xff)

{

int sum;

sum=(data[0]+data[1]+data[2])&0x00FF;

if(sum==data[3])

{

distance=(data[1]<<8)+data[2];

if(distance>30)

{

Serial.print("distance=");

Serial.print(distance/10);

Serial.println("cm");

}else

{

Serial.println("Below the lower limit");

}

}else Serial.println("ERROR");

}

delay(100);

}

int \_LastValidWaterHightValue = -2;

long \_LastValidWaterHightTimestamp= 0;

int waterHightRead(){

int value = ultrasonicReadDistance();

if (value == -1) //too close

{

\_LastValidWaterHightValue = 100;

\_LastValidWaterHightTimestamp = millis();

}

else if (value != -2 && value > WATER\_TANK\_TOTAL\_HIGHT\*10) //bigger then the tank - cannot be possible

{

\_LastValidWaterHightValue = -2;

\_LastValidWaterHightTimestamp = millis();

}

else if (value != -2)

{

\_LastValidWaterHightValue = ((WATER\_TANK\_TOTAL\_HIGHT\*10 - value)\*10)/WATER\_TANK\_TOTAL\_HIGHT; //calculate percentage

\_LastValidWaterHightTimestamp = millis();

}

if (millis() - \_LastValidWaterHightTimestamp > TIMEOUT\_FOR\_LAST\_VALUD\_WATER\_HEIGHT)

{

\_LastValidWaterHightValue = - 2;

}

return \_LastValidWaterHightValue;

}

int ultrasonicReadDistance(){

mySerial.flush();

mySerial.write(HIGH);

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

{

data[0]=mySerial.read();

if (data[0] == 0xff)

{

for(int i=1;i<4;i++)

{

data[i]=mySerial.read();

}

break;

}

}

/\*

do{

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

{

data[i]=mySerial.read();

}

}while(mySerial.read()==0xff);

\*/

mySerial.flush();

if(data[0]==0xff)

{

int sum;

sum=(data[0]+data[1]+data[2])&0x00FF;

if(sum==data[3])

{

distance=(data[1]<<8)+data[2];

if(distance>30)

return (distance);

else

return -1; //too close

}

else

{

return -2; //error

}

}

}

/\*

void waterHightTest(){

//Sensor test

Serial.print("Water low sensor:");

Serial.println(digitalRead(WATER\_LOW\_SWITCH\_PIN));

Serial.print("Water high sensor:");

Serial.println(digitalRead(WATER\_HIGH\_SWITCH\_PIN));

delay(1000);

//Function test

//Serial.print("Water hight test:");

//Serial.println(waterHightRead());

}

void waterHightInit(){

pinMode(WATER\_HIGH\_SWITCH\_PIN, INPUT);

pinMode(WATER\_LOW\_SWITCH\_PIN, INPUT);

}

int waterHightRead(){

return 15;

}

int waterHightRead(){

bool waterInLow = !digitalRead(WATER\_LOW\_SWITCH\_PIN); //\* off/down = 1, on/up = 0

bool waterInHigh = !digitalRead(WATER\_HIGH\_SWITCH\_PIN); //\* true 1 , false 0.

if(!waterInLow){

return 0;

}

else if(waterInHigh){

return 100;

}

else if (!waterInHigh && waterInLow){

return 50;

}

else {

return ERROR;

}

}

\*/

void waterHightCheckStatus(){

int WaterHight = waterHightRead();

Serial.println(WaterHight);

String WaterLevle = String(WaterHight);

//char buffer[80]

if(WaterHight == -2){

\_WaterHightStatus.WaterLevle = StatusResult("Error, check the water hight sensor", STATUS\_ERROR, PRIORITY\_HIGH);

\_WaterHightStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);

}

else if(WaterHight >= MAXIMUM\_WATER\_IN\_TANK){

\_WaterHightStatus.WaterLevle = StatusResult("Error, water levle is too high", STATUS\_ERROR, PRIORITY\_HIGH);

\_WaterHightStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);

}

else if(WaterHight <= MINIMUM\_WATER\_FOR\_PUMP){

\_WaterHightStatus.WaterLevle = StatusResult("Error, water levle too low for pump", STATUS\_ERROR, PRIORITY\_HIGH);

\_WaterHightStatus.Short = StatusResult("Error", STATUS\_ERROR, PRIORITY\_HIGH);

}

else{

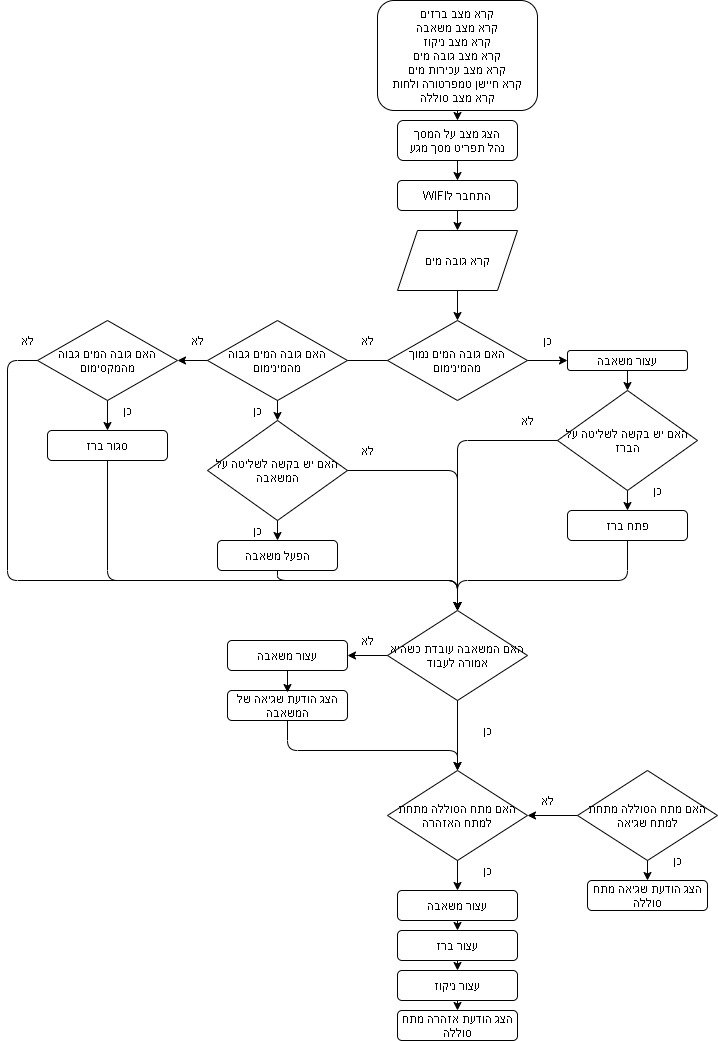
\_WaterHightStatus.WaterLevle = StatusResult(WaterLevle, STATUS\_OK, PRIORITY\_LOW);

\_WaterHightStatus.Short = StatusResult(WaterLevle, STATUS\_OK, PRIORITY\_LOW);

}

}

# אלגוריתם



# רפלקציה על התהליך ועל התוצר

במהלך ביצוע הפרויקט נהניתי בעיקר מתהליך הבנייה והתפעול שלו אך בנוסף גם נהניתי מהאתגרים השונים שהוצבו לי, אתגרים כמו ללמוד על רכיבים שלא למדתי עליהם בכיתה, להרחיב את הידע שלי בשפת התכנות כדי לבצע את הפרויקט ועוד אתגרים שונים שהוצבו לי בדרכי.(אולי להרחיב עוד על האתגרים).

במהלך ביצוע הפרויקט נתקלתי בקשיים שונים בניהם הקושי העיקרי היה להתקדם בו בתקופת הקורונה בגלל שרוב הזמן נאלצתי להתקדם בפרויקט בבית בלי אפשרות להתייעץ עם מורה במידת הצורך למרות שהיו לי הרבה מפגשים עם המנחה שלי עדיין היו מצבים שבהם לא יכולתי להתקדם עד שקיבלתי את הסיוע שלו. בנוסף, בגלל העומס שנוצר עם שאר המקצועות לא הצלחתי לארגן לעצמי לוז כדי גם לעבוד על הפרויקט וגם להתקדם בדברים האחרים.

במידה ואעבוד על פרויקט מהסוג הזה בעתיד אני אדאג לנהל לוח זמנים שיעבור על כל שלבי הפרויקט לוח זמנים שיהיה פתוח לשינויים ושמטרתו העיקרית היא להגיש את הפרויקט בזמן בלי לסיים את הכל בדקה ה – 90.

במהלך ביצוע הפרויקט למדתי על עצמי שיש לי יכולת ללמוד הרבה נושאים שלא תמיד באים מתוך בית הספר. למשל להרחיב את יכולות התכנות שלי ב – C++ למידה על פונקציות חדשות וגם פתרון בעיות באופן עצמאי בלי עזרה של מורה. בנוסף למדתי שאני יכול לשבת על פרויקט במשך שעות ארוכות לא משהו שחשבתי שאני אצליח לפני שהתחלתי אותו.