#### 

# 



## 

## 

## המגמה: תוכנה נז'

הנושא:

MyGarden – Smart Irrigation System

**מערכת השקייה אוטונומית**

##### **המגיש/ה: יונתן הרנס**

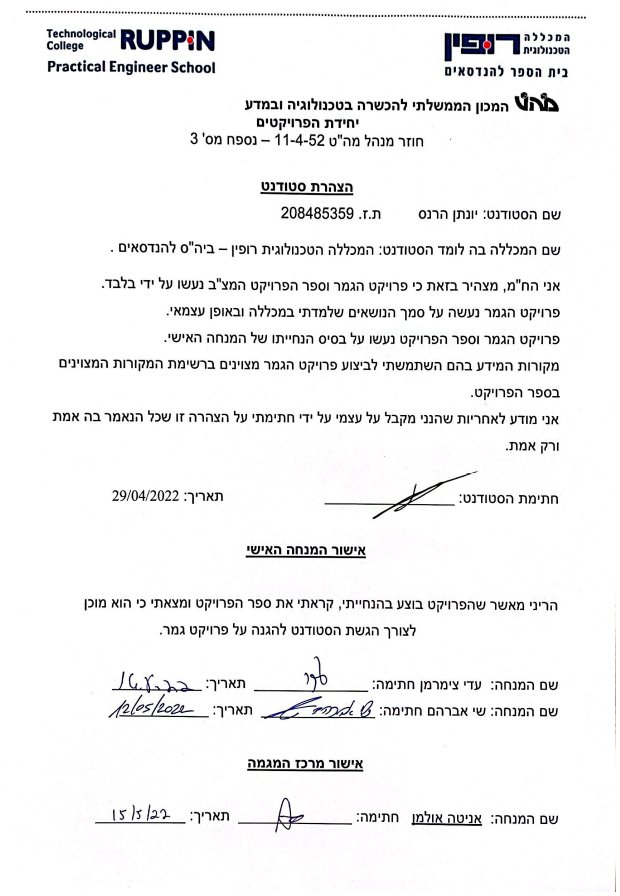
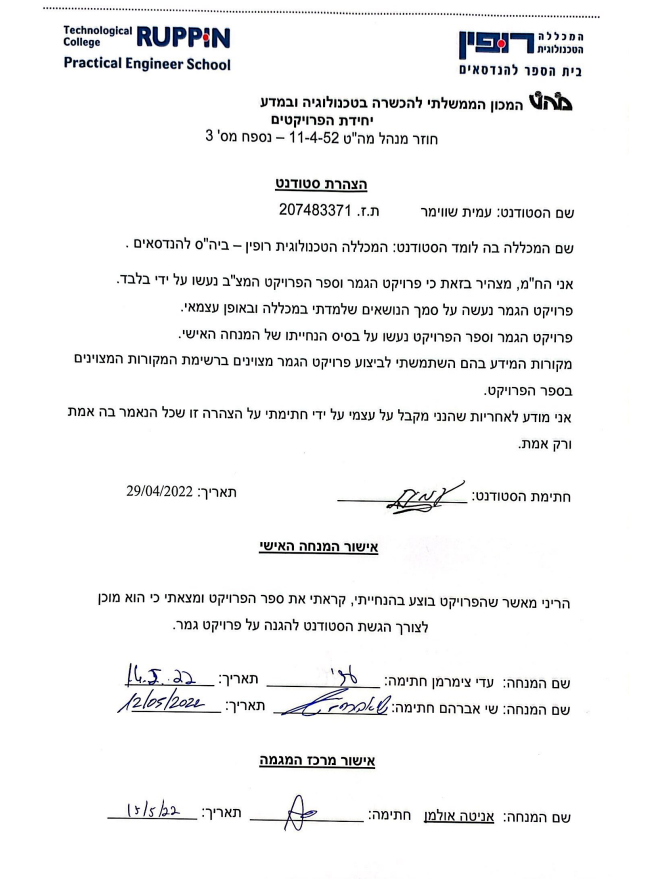
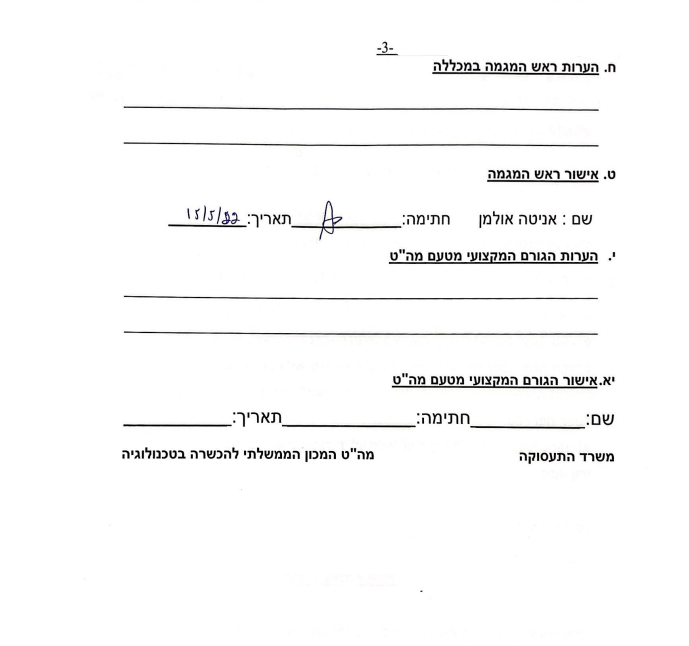
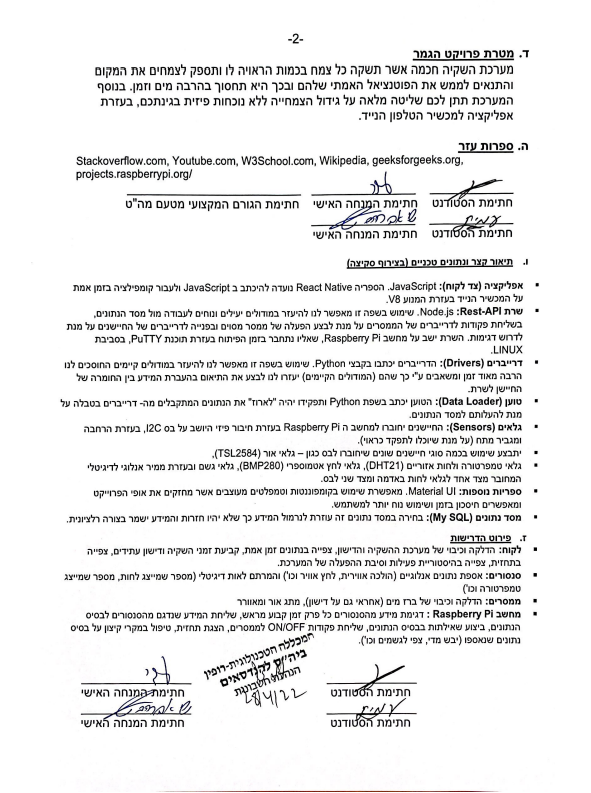
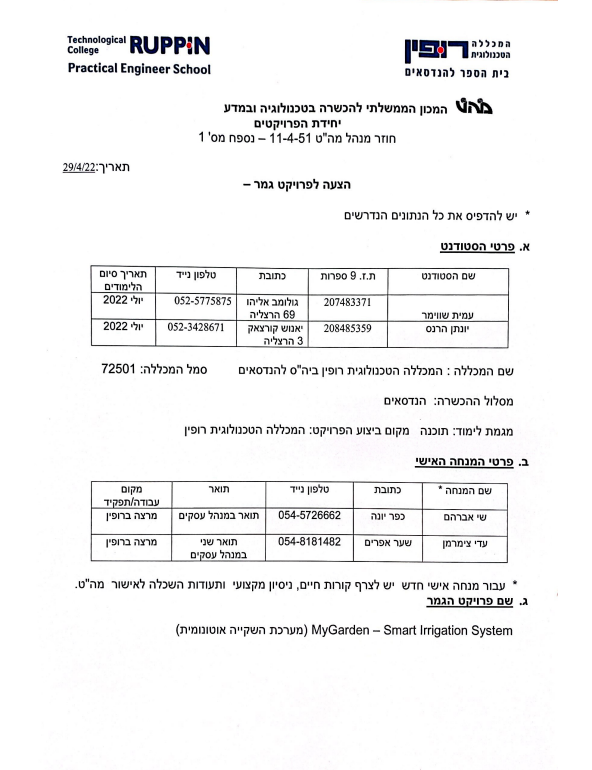
##### **ת.ז: 208485359**

##### **המגיש/ה: עמית שווימר**

##### **ת.ז: 207483371**

**המנחה: עדי צימרמן**

**המנחה: שי אברהם**



1. **שם הפרויקט**

**MyGarden**

**מערכת השקייה אוטונומית**

1. **רקע**
   1. **תיאור ורקע כללי**

כשמדברים על מערכת השקיה מתכוונים לרוב לצינור ראשי שיש לו הסתעפויות שמגיעות אל כל צמח, עץ או שיח בגינה. על ידי חיבור למחשב ניתן היום לשלוט על זמני ההשקיה של כל צמח וצמח ובכך למעשה ליצור מערכת השקיה נבונה, חכמה שחוסכת בעלויות רבות. באמצעות אוטומציה של כל התהליך, מערכת השקיה חכמה ויעילה תאפשר טיפוח הולם לגינה ותבטיח כמות מים קבועה ולא מבוזבזת. היום תוכלו למצוא מערכות השקיה בסיסיות ביותר ומערכות הרבה יותר מורכבות שיש להן שליטה באמצעות מחשב, ניתן להגדיר אותן מראש והן פועלות מעצמן. מערכת חכמה בגינה שהיא גם [מערכת השקיה חסכונית בגינה](https://www.hagarin.co.il/%D7%9E%D7%95%D7%A6%D7%A8%D7%99%D7%9D-1469-%D7%9E%D7%97%D7%A9%D7%91%D7%99%D7%9D-%D7%95%D7%9E%D7%A2%D7%A8%D7%9B%D7%95%D7%AA-%D7%94%D7%A9%D7%A7%D7%99%D7%94.aspx), תאפשר ביצוע של כמה משימות בו זמנית, כדי להגיע לתוצאה האיכותית ביותר. לפני הכל, מערכות השקיה תהיינה אחראיות לאספקה של מים לכל הצמחים במועד שתוכנן ונקבע מראש על ידי המחשב ההשקיה ובנוסף לכל זה היא גם תחסוך בכוח אדם וזמן

* 1. **מטרות המערכת**

מערכת השקיה חכמה אשר תשקה כל צמח בכמות הראויה לו ותספק לצמחים את המקום והתנאים לממש את הפוטנציאל האמתי שלהם ולכן היא  תחסוך לכם בהרבה מאוד מים וזמן. בנוסף המערכת תוכל לתת לכם שליטה מלאה על גידול הצמחייה ללא נוכחות פיזית בגינתכם

1. **המצב בשוק והבעיות**
   1. **סקירת מצב קיים בשוק**

ישנן מספר מערכות השקיה ביתיות חכמות בשוק כרגע, אך המוכרות ביותר בינהן הן של חברת Galcon וחברת GardenPro.

**מערכת השקיה של חברת Galcon:**

**יתרונות:**

* חווית משתמש נוחה ופשוטה שבה ניתן לאתחל את המערכת בהגדרות קבועות מראש.
* הציוד הנכלל בחבילה כולל את רוב האבזור הדרוש על מנת לספק למשתמש יכולת לגדל צמחים ללא נוכחות פיזית בגינה.
* הציוד שהחברה מספקת ללקוח הינו איכותי מאוד, עמיד בתנאי מזג אוויר קיצוני ללא צורך באמצעי הגנה נוספים מצד הלקוח (כגון כיסוי המערכת בעת גשמים, חשש להתחממות בימי הקיץ החמים וכו').

**חסרונות:**

* המערכת אינה מותאמת לשימוש מרוחק (אין אפליקציה שנותנת שליטה מרחוק, אפשרות לשינוי זמני השקיה והפעלה מידית של מערכת ההשקיה). שינוי הגדרות המערכת מוכרח להתבצע פיזית מול הבקר שהחברה מספקת
* החבילה כוללת את ציוד ההשקיה בלבד ואינו כולל בקרי טמפרטורה ולחות שמאפשרים לנו לקבל מידע נוסף ולממש את פוטנציאל הצמח
* החברה מתעסקת בעיקר בשדות ופחות נותנת שירותים למשתמש הביתי (ישנם שירותים למערכת ביתית אך הם אינם פונים לכל בית), המערכות שהחברה משתמשת בהן הינן גדולות, תופסות המון מקום ומסורבלות מבחינת השינוע שלהן לבית הלקוח.
* הלקוח אינו יכול ללכת לחנות הגינון הקרובה לביתו ולקנות את המערכת, עליו להזמין צוות מקצועי לביתו (בתיאום זמן פנוי) על מנת שיגיעו להתקין את המערכת
* השימוש החודשי במערכת הינו בתשלום, כלומר נדרש מינוי ותשלום חודשי מצד המשתמש בנוסף לתשלום עבור ההתקנה והמוצר.

**מערכת השקיה של חברת GardenPro:**

**יתרונות:**

* מערכת ההשקיה הביתית הינה פשוטה להתקנה ואינה כוללת ידע נרחב בחומרה בכדי שתתחיל לבצע את עבודתה
* מחיר זול יחסית, זמינה לכל בוטן חובב בכל חנות המספקת אביזרים לגידול צמחים ולרווחת הגינה.
* בקר נייח, נוח ופשוט לשימוש, ללא פיצ'רים מבלבלים הנותן אפשרות נוחה לכל חובבן הנכנס לתחום להתחיל לגדל צמחים מידית.
* אינה דורשת חיבור לאינטרנט בכדי לבצע את עבודתה, חיבור חשמל ומים הינו התנאי היחיד להפעלה של המערכת

**חסרונות:**

* מפאת המחיר הנמוך של המוצר החומרים שמהם מיוצר המוצר אינם איכותיים (פלסטיק) ומעצם היותה מערכת השקיה המפרסמת את עצמה גם כמערכת "חוץ ביתית" ניתן להסיק שתוחלת החיים שלה מאוד נמוכה.
* מערכת ההשקיה אינה כוללת בתוכה קישור לאפליקציה – כלומר לא ניתן לשנות הגדרות מערכת "תוך כדי תנועה" ובכך לאפשר לצמח לממש את הפוטנציאל שלו. בנוסף מפאת חוסר חיבור בכלל לאינטרנט המערכת עצמה לא נחשבת "חכמה מספיק" בכך שאינה נותנת למשתמש אפשרות לקבלת נתונים חיים על הצמחים.
* הפונקציונאליות של המערכת לוקה בחסר, ניתן להגדיר רק זמני השקיה עתידיים ללא טיפול מידי (כגון במקרים של טמפרטורה גבוה, לחות נמוכה מדי וכו')
  1. **בעיות במצב הקיים**

לאחר ביצוע סקר השוק למערכות השקיה חכמות נוספות הסקתי שכרגע יש בעיה בכך שמערכות ההשקיה מתחלקות לשתיים.

סוג אחד של מערכות ההשקיה אינו פונקציונאלי מספיק. אינו מספק למשתמש מספיק מידע על התפתחות הצמח ואינו נותן למשתמש אפשרות לשנות הגדרות "תוך כדי תנועה". הסוג השני של מערכות ההשקיה הקיימות בשוק אינו מיועד בעיקר לשימוש ביתי אלא יותר לשימוש עסקי, כלומר מערכות השקיה לשדות אשר מפרסמות את עצמן לשימוש ביתי למרות היותן מסורבלות, דורשות משאבים רבים, תשלום חודשי מהמשתמש על מנת להשתמש בפונקציונאליות של המערכת, תיאום התקנה של צוות מקצועי מפאת האפשרות ללכת לחנות הקרובה לביתך ורכישה של המוצר וכמובן שהיא מקשה על המשתמש במידה והינו מעוניין לבצע שינויים בחזות הגינה (עליו לתאם לצוות מקצועי אשר יבוא ויבצע את ההזזה של המערכת למיקום הנדרש).

1. **מה הפרויקט אמור לחדש או לשפר**

* יכולת הצגת היסטוריית פעילות וסיבת הפעלת המערכת.
* מערכת מותאמת אישית לדרישת הלקוח.
* המערכת תספק לצמח את התנאים האופטימליים למימוש פוטנציאל הגדילה שלו
* המערכת מספקת יכולת הפעלה מרחוק של מרכיבי המערכת(הדלקת אור, דישון , מאוורר וכו').
* טיפול מידי ב"חריגים" (לדוגמא במקרה של חושך המערכת תדליק את התאורה במתחם).
* המערכת משתמשת בחיישנים הנמצאים בשטח ובתחזיות הנלקחות מהמרשתת.

1. **דרישות מערכת ופונקציונאליות**

**רשימת שחקנים:**

* לקוח (מגדל)
* סנסורים
* ממסרים
* מחשב Raspberry Pi
  1. **דרישות מערכת**
* טלפון חכם שמותקנת עליו האפליקציה
* חיבור של הטלפון לאינטרנט
* מחשב Raspberry Pi עם חיבור חשמל ואינטרנט קבועים
* גלאי אור
* גלאי טמפרטורה
* גלאי גשם
* גלאי לחץ אטמוספרי
* גלאי לחות קרקע
* ממסר אור
* ממסר מים
* ממסר דישון
* ממסר מאוורר
  1. **דרישות פונקציונאליות**
* **לקוח:** הדלקה וכיבוי של מערכת ההשקיה והדישון, צפייה בנתונים זמן אמת, קביעת זמני השקיה ודישון עתידים, צפייה בתחזית, צפייה בהיסטוריית פעילות וסיבת ההפעלה של המערכת, קביעת מצב מערכת, קביעת תנאי הפעלה למערכת.
* **גלאים:** אספת נתונים אנלוגיים (הולכה אווירית, לחץ אטמוספרי וכו') והמרתם לאות דיגיטלי (מספר שמייצג לחות, מספר שמייצג טמפרטורה וכו')
* **ממסרים:** הדלקה וכיבוי של ממסר מים, ממסר דישון, ממסר אור וממסר מאוורר
* **מחשב Raspberry Pi :** דגימת מידע מהסנסורים כל פרק זמן קבוע מראש, שליחת המידע שנדגם מהסנסורים לבסיס הנתונים, ביצוע שאילתות בבסיס הנתונים, שליחת פקודות ON/OFF לממסרים, הצגת תחזית, טיפול במקרי קיצון על בסיס נתונים שנאספו (יבש מדי, צפי לגשמים וכו')

1. **בעיות צפויות במהלך הפיתוח ופתרונות** 
   1. **תיאור הבעיות**

במהלך הפיתוח של פרויקט המשלב בתוכו חומרה, מספר הבעיות האפשריות גדל משמעותית. מספר בעיות שיכולות להופיע הינן:

* איך להעלות את המידע מהגלאים למסד הנתונים?
* איך לוודא שכל הגלאים קיימים ועובדים?
* מה לעשות במקרה שאחד הגלאים לא עובד כראוי?
* איך לדעת שהמשתמש החליט להפעיל ממסר מסוים?
* איך לדעת שהגיעה השעה להפעלה (בהתאם לקביעת משתמש)?
* מה יהיו "תנאי הקיצון" (תנאי מזג אוויר שיפעילו את המערכת)?
* שיטת הוצאת המידע ממסד הנתונים לאפליקציה בעמוד הניתוחים הסטטיסטים
  1. **פתרונות אפשריים**
* פתרון אחד לבעיה של **העלאת המידע מהגלאים** הוא שלכל גלאי נכתוב סקריפט פיתון ("דרייבר"), תפקידו יהיה לגשת לגלאי הפיזי, דרך הבס, להמיר ולכייל את המידע המתקבל (מספר ממשי כלשהו שאינו מייצג יחידת מידה כלשהי כגון טמפ’, אור וכדומה אלא מייצג את התנגדות ההולכה) ליחידת המידע המתאימה. לאחר שכל הגלאים, הדרייברים יהיו מוכנים נוכל "לעטוף" את כל הדרייברים בסקריפט נוסף שתפקידו יהיה להפעיל ולקבל את המידע המכויל מכל הגלאים ולהרכיב אובייקט מסוג ג'ייסון (בהתאם לפורמט המתאים של הטבלה בבסיס הנתונים). לאחר מכן נכתוב סקריפט נוסף שיפעיל ויקבל את האובייקט המוכן ותפקידו יהיה העלאת האובייקט לטבלה המתאימה במסד הנתונים. פתרון זה מאפשר לנו לאתר ולמנוע בעיות בקלות מפני שהפרדנו פעולות גדולה ומסובכת לתתי פעולות.
* בכדי **לוודא שכל הגלאים קיימים ועובדים** ישנם מספר פתרונות אפשריים. אך הפתרון המתאים ביותר למערכת שלנו הינו לוודא בכל פעם שאנו מפעילים את הסקריפט של יצירת אובייקט הג'ייסון המכיל את המידע מכל הגלאים, שכל הגלאים/ממסרים/ממירים (A/D) מזוהים ע"י השרת (מחשב הRaspberry Pi), ולעדכן קובץ לוג, עם תאריך ושעה בהתאם. במידה ותהיה חריגה כלשהי או מידע שחסר מגלאי מסוים, נוכל לזהות זאת בקלות רבה יותר מאשר בחיפוש ידני.
* במקרה **שאחד הגלאים לא עובד כראוי** (אבל ישנו מתח) יהיה עלינו לכיילו מחדש.

הכיול מתבצע בעזרת ההוראות של יצרן הגלאי (במידה ויש) או בעזרת גלאי אחר המספק את אותה יחידת מידה והשוואה ביניהם (פחות ממומלץ אך אפשרי). במקרה של חוסר מתח המקרה קצת יותר מסתבך ויהיה עלינו לברר אם הבעיה נמצאת בכל הגלאים (כלומר שהמגבר זרם לא מתפקד כראוי), האם הבעיה נמצאת בכבל של אותו גלאי, בחיבורים או בגוף הגלאי עצמו

* בכדי לדעת **האם המשתמש מעוניין להפעיל ממסר מסוים** (אור, השקיה, מאוורר או דישון) יהיה עלינו לבדוק כל פרק זמן מסוים, האם הטבלה המתאימה בבסיס הנתונים השתנתה (כלומר שהמשתמש בחר להפעיל/לכבות אותה) ולהפעיל / לכבות את הממסר בהתאמה. למען ייעול ונוחות מערכת ההפעלה לינוקס מספקת לנו אפשרות להפעיל סקריפט מסוים, כל פרק זמן שנבחר וזה יהיה תפקידו.
* העיקרון של לבדוק **ולהפעיל את המערכת בזמן עתידי** כלשהו מאוד דומה לעיקרון של הפעלה מרחוק (הפעלה ברגע זה), ולכן הפתרון גם הוא זהה. יהיה עלינו לבדוק כל פרק זמן מסוים את הטבלה המתאימה במסד הנתונים ולהפעיל את הממסר בהתאם למידע שבטבלה. גם כאן ניעזר במערכת ההפעלה לינוקס ובשירותים שהיא מספקת לנו.
* בהקשר **של קביעת תנאי הקיצון** ישנם מספר פתרונות אפשריים אך המוצלח ביניהם הוא גם הפשוט ביותר, לאפשר למשתמש להחליט באילו תנאי מזג האויר המערכת תפעלנה. פתרון זה מאפשר למשתמש להיות זה "שמחליט" בהתאם לסוג הצמחייה אותה בחר לגדל. אך בכל זאת נספק למשתמש "תנאי ברירת מחדל" לפי ממוצע מזג אוויר שנתי של המזרח התיכון הצפוני.
* **קבלת המידע ממסד הנתונים**, במיוחד לעמוד הניתוחים הסטטיסטים (Analytics), המכיל בפער את כמות המידע הכי גדולה הנשלפת ממסד הנתונים היא בעיה לא פשוטה מכיוון שאנו מאפשרים למשתמש לבחור לצפות במידע של שנה אחת אחורה. לכן עלינו לבחור האם כל פעם שהמשתמש משנה את תקופת הזמן של הנתונים, לקבל מחדש את הנתונים (דבר אשר יעמיס על מסד הנתונים) או האם בכניסה לעמוד, לקבל את כל המידע (קריאות של הגלאים משנה אחורה ועד העדכנית ביותר), דבר אשר יכול לעכב את טעינת הדף. הפתרון המתבקש במצב זה הוא קבלת הנתונים של הקריאות, משנה אחורה, מפני שאם כמה שהכמות מידע גדולה, מסד הנתונים שבחרנו מיועד לקריאות כאלה והמשתמש לא ירגיש כלל בעיכוב, במיוחד שכל המערכת יושבת על אותו שרת.

1. **פתרון טכנולוגי נבחר**
   1. **טופולוגית הפתרון**

מערכת השקייה חכמה בעלת אפליקציה יעילה ונוחה מבוססת על React Native, אשר מותקנת על המכשיר הנייד של המשתמש. האפליקציה תקושר לשרת שאנו נתכנן ונוציא לפועל שישב על Raspberry Pi. השרתים מריצים סביבת Node.js. התקשורת בין צד לקוח לצד שרת תתבצע ע"י תקשורת אינטרנט. המידע ישמר על גבי שרתי MariaDB, אשר גם כן ישב על גבי שרת ה raspberry pi. בנוסף, על גבי שרת ה Raspberry Pi

יופעלו ע"י טיימר של מערכת ההפעלה, סקריפטים (python) ותפקידם יהיה להפעיל את הממסרים, לקרוא מידע מהגלאים ולהעלותו למסד הנתונים. מחשב (שרת) Raspberry Pi בעל מגבר מתח המקושר בעזרת הרחבה פיזית ל-בס I2C

* 1. **טכנולוגיות בשימוש**
* Expo – React Native
* Rest API – Node.js
* Data Loader – Python
* Drivers - Python
* Remote SSH - PuTTY
* Transact SQL - MariaDB
* SFTP – WinSCP
  1. **שפות הפיתוח**
* **אפליקציה (צד לקוח):** JavaScript. הספריה React Native נועדה להיכתב ב JavaScript ולעבור קומפילציה בזמן אמת על המכשיר הנייד בעזרת המנוע V8.
* **שרת Rest-API:** Node.js. שימוש בשפה זו מאפשר לנו להיעזר במודולים יעילים ונוחים לעבודה מול מסד הנתונים .שימוש ב Rest API,

מאפשר לנו למקם את שרת המשתמש והשרת במיקומים שונים ובעזרת סנכרון המידע ביניהם למונע תקלות. מאפשר למשתמשים לדרוש מידע ולבצע שאילתות ללא פגיעה בביצועים. השרת ישב על מחשב Raspberry Pi, שאליו נתחבר בזמן הפיתוח בעזרת תוכנת PuTTY, המאפשרת השתלטות על מחשב מרחוק ושימוש במערכת הפעלה LINUX.

* **דרייברים (Drivers):** הדרייברים יכתבו בקבצי Python. שימוש בשפה זו מאפשר לנו להיעזר במודולים קיימים החוסכים לנו הרבה מאוד זמן ומשאבים ע"י כך שהם (המודולים הקיימים) יעזרו לנו לבצע את התיאום בהעברת המידע בין החומרה של החיישן לשרת, מפשטים את השימוש בחומרה ומאפשרים לנו לבצע בדיקות על מנת לאמת את תקינות החומרה.
* **טוען (Data Loader):** הטוען יכתב בשפת Python ותפקידו יהיה "לארוז" את הנתונים המתקבלים מה- דרייברים בטבלה על מנת להעלותם למסד הנתונים. בעתיד התקשורת בין הדרייברים לטוען ובין הטוען למסד הנתונים תתבצע בסביבת ה – localhost, מפני שגם השרת, הטוען ומסד הנתונים יושבים על ה- Raspberry.
* **גלאים (Sensors):** החיישנים יחוברו למחשב הRaspberry Pi בעזרת חיבור פיזי היושב על בס I2C, בעזרת הרחבה ומגביר מתח (על מנת שיוכלו לתפקד כראוי). יתבצע שימוש בכמה סוגי חיישנים שונים שיחוברו לבס כגון –

גלאי אור (TSL2584), גלאי טמפרטורה ולחות אזוריים (DHT21), גלאי לחץ אטמוספרי (BMP280), גלאי גשם ובעזרת ממיר אנלוגי לדיגיטלי המחובר מצד אחד לגלאי לחות באדמה ומצד שני לבס.

* **מסד נתונים (MariaDB - MySQL):** בחירה במסד נתונים זה עוזרת לנרמול המידע כך שלא יהיו חזרות והמידע ישמר בצורה רלציונית.
* **ספריות נוספות:** Material UI. מאפשרת שימוש בקומפוננטות וטמפלטים מעוצבים אשר מחזקים את אופי הפרוייקט ומאפשרים חיסכון בזמן ושימוש נוח יותר למשתמש.
  1. **תיאור הארכיטקטורה הנבחרת**

Water

Light

Soil

Moist

senso

Soil

Moist

senso

4 Channel Relay

Rain

Sensor

BMP280

Atmospheric

Pressure Sens

DHT21

Temperature & Humidity

Sensor

TSL2584

Light Sensor

ADS 1115

AD 4 port 16 bit

HAL – python drive

Raspberry Pi 4

4GB RAM + 32 GB SD

REST API –

Node.js

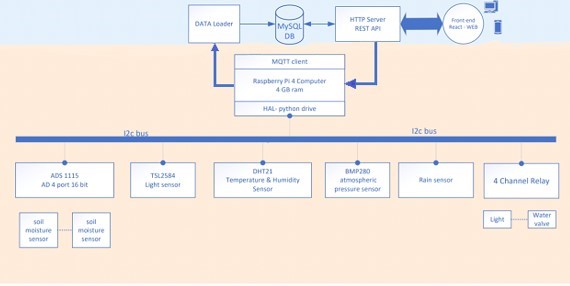
DATA

LOADER

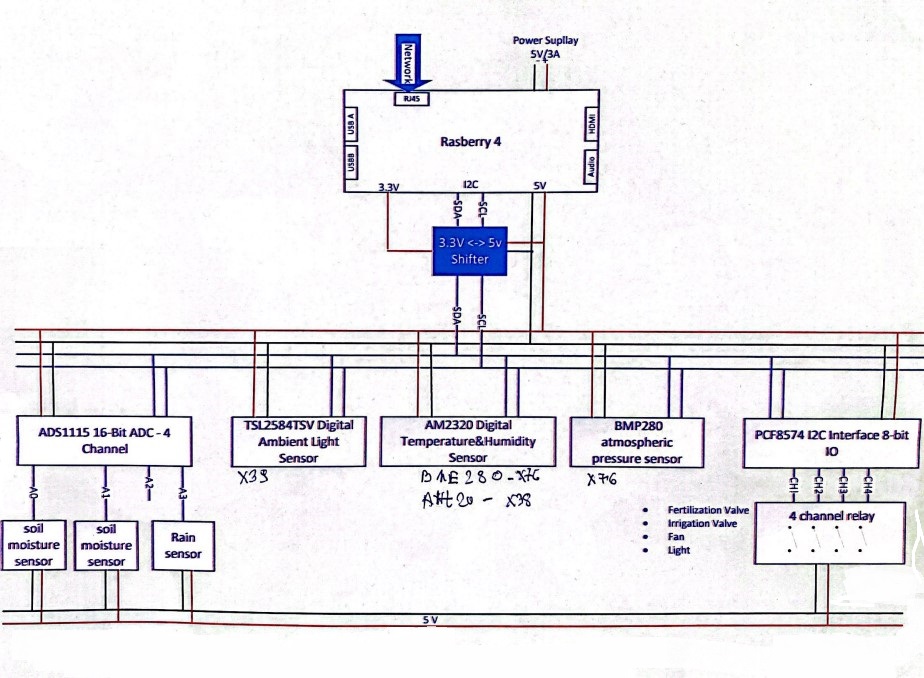
REST API –

Node.js

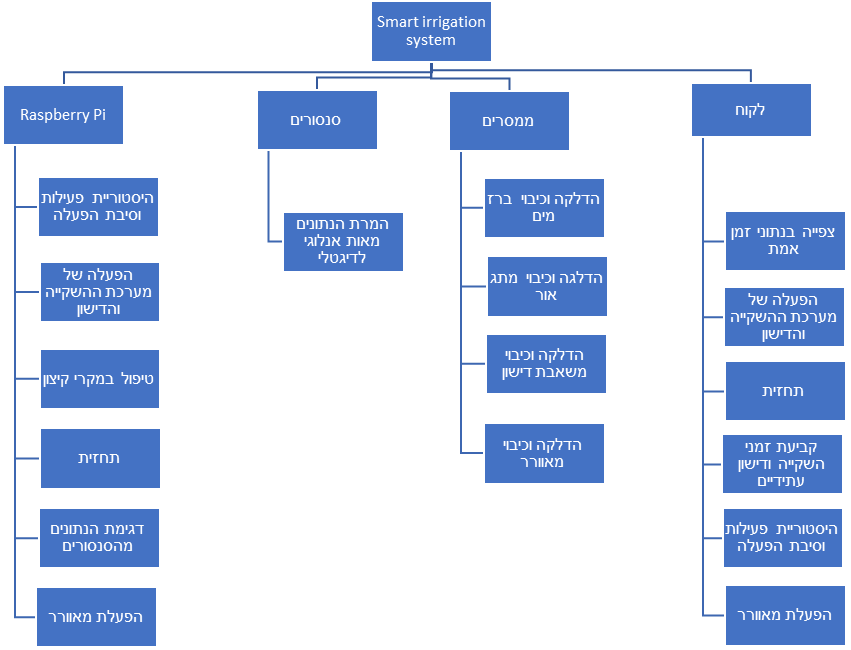
Front – React Native



**תיאור סכמת החומרה**



* 1. **חלוקה לתכניות ומודולים**



* 1. **סביבת השרת**

העבודה תתבצע מול שרת מקומי הנמצא בRaspberry Pi, בעזרתו נשלח פקודות, נבצע פניות למסד הנתונים, נקבל בקשות מהמשתמש נחזיר תשובות למשתמש. בנוסף לשרת תתבצענה בדיקה כל פרק זמן מסוים בעזרת Schedule שמספקת לנו מערכת ההפעלה, ותפקידו יהיה להפעיל/לכבות ממסרים לפי דרישה, ולקרוא מידע חדש מהגלאים ולהעלותם למסד הנתונים. מפני שהשרת הוא פרטי וייעודי לפרויקט זה, העבודה מולו תתבצע בעזרת תוכנה בשם PuTTY – המאפשרת שליטה מרחוק על מערכת ההפעלה בפרוטוקול SSH, והעברת קבצים לשרת וממנו בעזרת WinSCP – בפרוטוקול SFTP.

* 1. **ממשק המשתמש/לקוח – GUI**

|  |  |
| --- | --- |
| **מסך הבית** | **מסך הפעלה מרחוק** |

|  |  |
| --- | --- |
| **מסך היסטוריה** | **מסך העדפות** |

|  |  |
| --- | --- |
| **מסך נתונים אנליטיים** | **מסך נתונים אנליטיים** |

|  |
| --- |
| **מסך קביעת זמני הפעלה עתידיים** |

* 1. **ממשקים למערכות אחרות/API**
* Open Weather API
  1. **שימוש בחבילות תוכנה**

1. **Frontend**

* react-native-community/datetimepicker V6.3.2
* react-native-community/geolocation V2.1.0
* react-native-community/masked-view V0.1.11
* react-native-material/core V1.3.7
* react-navigation/bottom-tabs V6.3.3
* react-navigation/drawer V6.4.4
* react-navigation/native V6.0.12
* react-navigation/stack V6.2.3
* axios V0.27.2
* body-parser V1.20.0
* expo V~45.0.0
* expo-location V~14.2.2
* expo-status-bar V~1.3.0
* express V4.18.1
* moment V2.29.4
* mysql V2.18.1
* react V17.0.2
* react-dom V17.0.2
* react-native V0.68.2
* react-native-beautiful-timeline V0.1.1
* react-native-chart-kit V6.12.0
* react-native-dropdown-picker V5.4.2
* react-native-element-dropdown V2.3.0
* react-native-gesture-handler V2.6.0
* react-native-material-dropdown V0.11.1
* react-native-modal-datetime-picker V13.1.2
* react-native-numeric-input V1.9.1
* react-native-paper V4.9.2
* react-native-progress V5.0.0
* react-native-safe-area-context V4.3.1
* react-native-screens V3.17.0
* react-native-slider V0.11.0
* react-native-svg V12.3.0
* react-native-weather-api V3.0.2
* react-native-web V0.17.7
* victory-native V36.6.7

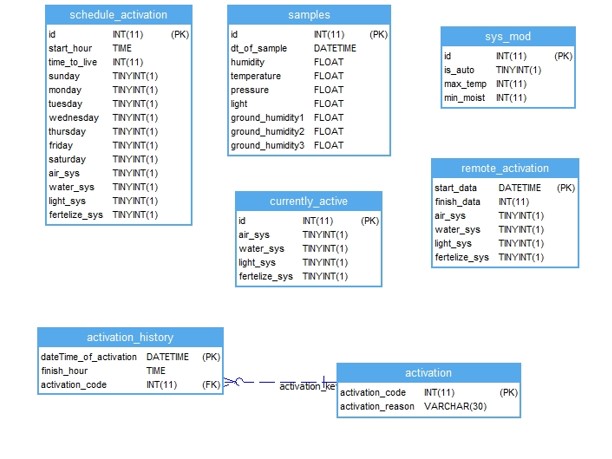
1. **REST API**

* express V4.18.1
* mysql2 v2.3.3

1. **Drivers**

* datetime
* sys
* json
* smbus
* mysql.connector
* time
* SMBus from smbus
* load\_workbook from openpyxl

1. **שימוש במבני נתונים וארגון קבצים** 
   1. **מבני הנתונים**



**DSD**

**מילון טבלאות**

**שם טבלה: סיבות הפעלה – activation**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| מספר מזהה ייחודי | Not Null, PK | int | activation\_code |
| תיאור סיבת ההפעלה | Not Null | VARCHAR (30) | activation\_reason |

**שם טבלה: היסטוריית דגימות – samples**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| מספר מזהה לדגימה | Not Null, PK, AUTO\_INC | INT | id |
| תאריך ושעת דגימה | Not Null | DATETIME | dt\_of\_sample |
| דגימת גלאי לחות אווירית | Not Null | FLOAT | humidity |
| דגימת גלאי טמפרטורה | Not Null | FLOAT | temperature |
| דגימת גלאי לחץ אטמוספרי | Not Null | FLOAT | pressure |
| דגימת גלאי אור | Not Null | FLOAT | light |
| דגימת גלאי לחות קרקע 1 | Not Null | FLOAT | ground\_humidity1 |
| דגימת גלאי לחות קרקע 2 | Not Null | FLOAT | ground\_humidity2 |
| דגימת גלאי לחות קרקע 3 | Not Null | FLOAT | ground\_humidity3 |

**שם טבלה: הפעלה ידנית – remote\_activation**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| שעת התחלה פעילות | Not Null, PK | DATETIME | start\_data |
| משך זמן הפעלה בדקות | Not Null | INT | finish\_data |
| הם להפעיל מערכת אוורור | Not Null | TINYINT(1) | air\_sys |
| האם להפעיל השקיה | Not Null | TINYINT(1) | water\_sys |
| האם להפעיל אור | Not Null | TINYINT(1) | light\_sys |
| האם להפעיל דישון | Not Null | TINYINT(1) | fertelize\_sys |

**שם טבלה: הפעלה נוכחית – currently\_active**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| מספר מזהה ייחודי | Not Null, PK, AUTO\_INC | INT | id |
| הם להפעיל מערכת אוורור | Not Null | TINYINT(1) | air\_sys |
| האם להפעיל השקיה | Not Null | TINYINT(1) | water\_sys |
| האם להפעיל אור | Not Null | TINYINT(1) | light\_sys |
| האם להפעיל מערכת דישון | Not Null | TINYINT(1) | fertelize\_sys |

**שם טבלה: מצב המערכת – sys\_mod**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| תאריך בו המערכת פעלה | Not Null, PK, AUTO\_INC | INT | id |
| האם מצב אוטומטית | Not Null | TINYINT(1) | is\_auto |
| טמפרטורת מקסימום | Not Null | INT | max\_temp |
| לחות מינימלית | Not Null | INT | min\_moist |

**שם טבלה: זמני הפעלה עתידיים – schedule\_activation**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| תאריך בו המערכת תפעל | Not Null, PK, AUTO\_INC | INT | id |
| שעת התחלה פעילות | Not Null, PK | TIME | start\_hour |
| משך זמן הפעלה בדקות | Not Null | INT | time\_to\_live |
| הם להפעיל אוורור | Not Null | TINYINT(1) | air\_sys |
| האם להפעיל השקיה | Not Null | TINYINT(1) | water\_sys |
| האם להפעיל אור | Not Null | TINYINT(1) | light\_sys |
| האם להפעיל דישון | Not Null | TINYINT(1) | fertelize\_sys |
| האם לפעול ביום ראשון | Not Null | TINYINT(1) | sunday |
| האם לפעול ביום שני | Not Null | TINYINT(1) | monday |
| האם לפעול ביום שלישי | Not Null | TINYINT(1) | tuesday |
| האם לפעול ביום רביעי | Not Null | TINYINT(1) | wednesday |
| האם לפעול ביום חמישי | Not Null | TINYINT(1) | thursday |
| האם לפעול ביום שישי | Not Null | TINYINT(1) | friday |
| האם לפעול ביום שבת | Not Null | TINYINT(1) | saturday |

**שם טבלה: היסטוריית פעילות activation\_history**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| תאריך בו המערכת פעלה | Not Null, PK | DATETIME | dateTime\_of\_activation |
| שעת סיום | Not Null | TIMR | finish\_data |
| קוד סיבת הפעלה | Not Null, FK | INT | activation\_code |

**8.2 שיטת האחסון**

בסיס נתונים טבלאי על גבי MariaDB.

שפת הפיתוח שבאמצעותה מועברות הפקודות למערכת היא Transact-SQL ‏ שהיא מימוש תקן ANSI של שפת SQL משמשת לתשאול וטיפול בנתונים, יצירת טבלאות והיחסים ביניהן ותחזוקת המערכת תוך שימוש בתוכניות שירות שונות.

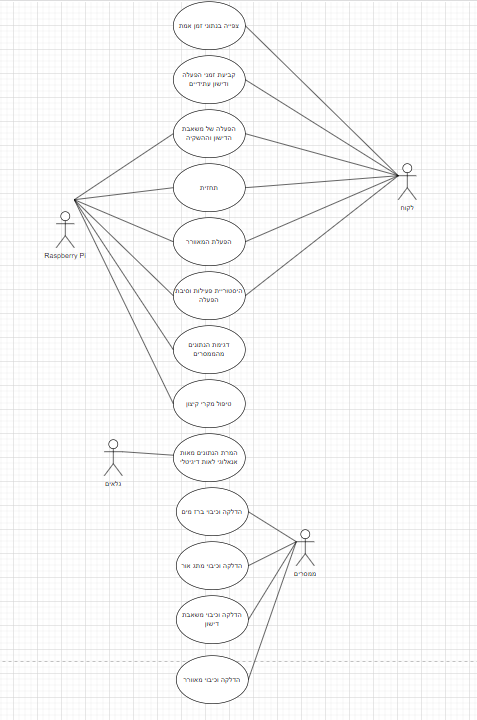
**8.3 מנגנוני התאוששות**

מספר כלים מאפשרים ליצור עותק של בסיס הנתונים לשפר את הזמינות וההתאוששות מתקלות, בנוסף לגיבוי שנחשב לאיטי. העותקים הנוצרים יכולים לשמש הן כגיבוי זמין במקרה של תקלה במערכת, הן לעבודה במערכות משנה המנותקות מהמערכת הראשית, והן לחלוקת עומסים כך שהעדכון יתבצע בבסיס הנתונים האמיתי והשליפות לצורכי דוחות ותחקור מההעתקים.

בין הכלים ניתן לציין את כלי הרפליקציה שמאפשרים ליצור עותק מסונכרן של בסיס הנתונים, Mirroring שמאפשר החזקת עותק של בסיס הנתונים והרצה של כל פקודה שמשנה את בסיס הנתונים המקורי, Log Shipping שמאפשר החזקת עותק מסונכרן באמצעות העברת קובצי הלוג וביצוע (בעותק) של השינויים שהצטברו בהם (במקור), וכלי ה-HADR שפותר בעיות שונות של ה-Mirroring מבחינת מהירות התגובה לתקלה.

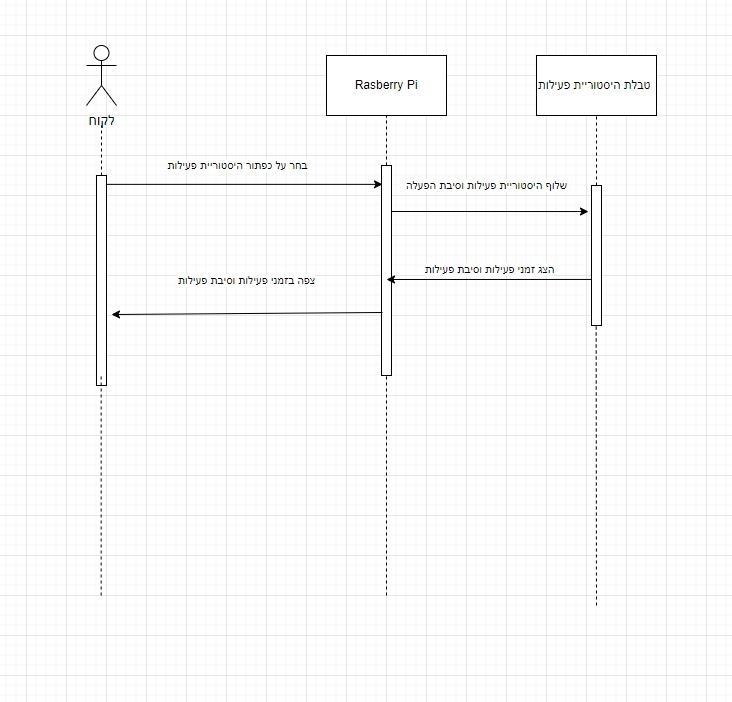
שימוש ב Transactions שיגנו על פעולות Delete \ Update \ Insert בבסיס הנתונים כדי למנוע בעיות של התאוששות.

1. **תרשימי מערכת מרכזיים** 
   1. **Use Case**

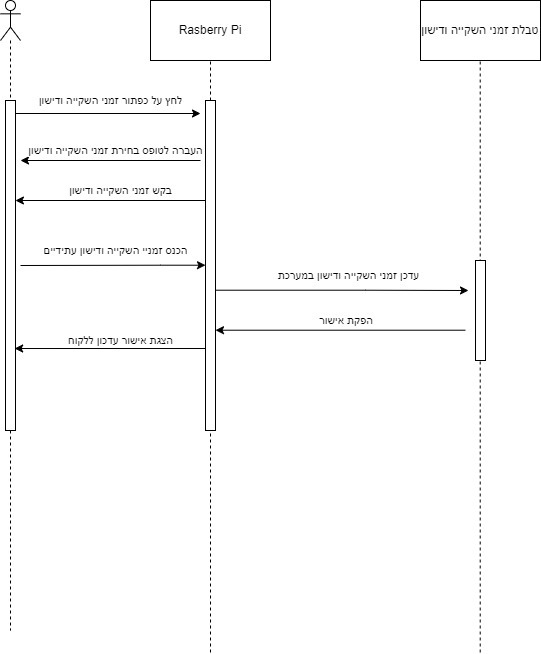


* 1. **Sequence Diagram**

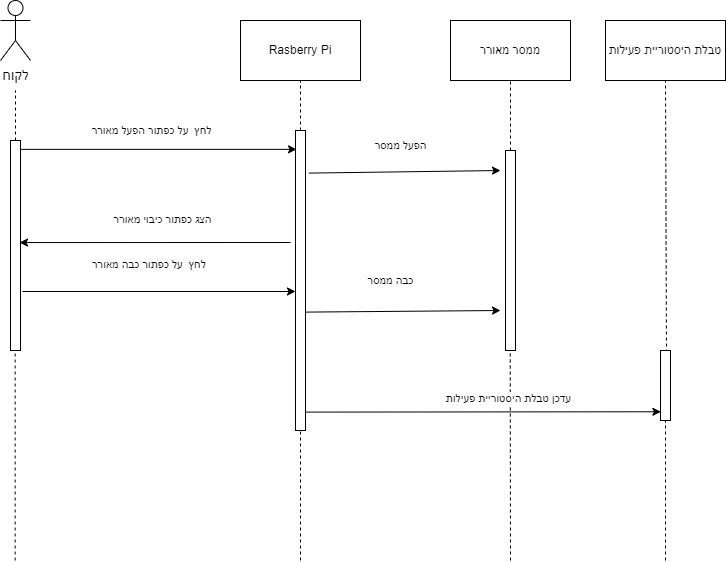
**צפייה בהיסטוריית פעילות וסיבת הפעלה**



**עדכון זמני השקייה ודישון עתידיים**



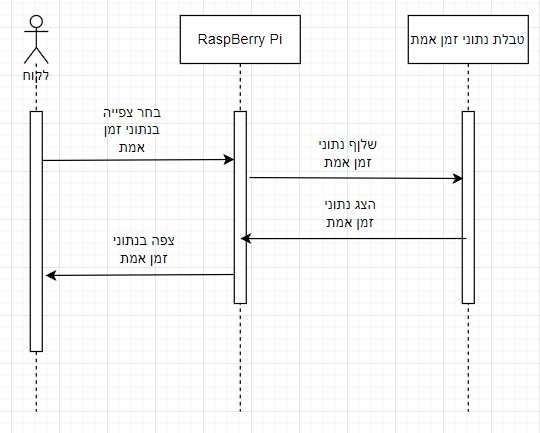
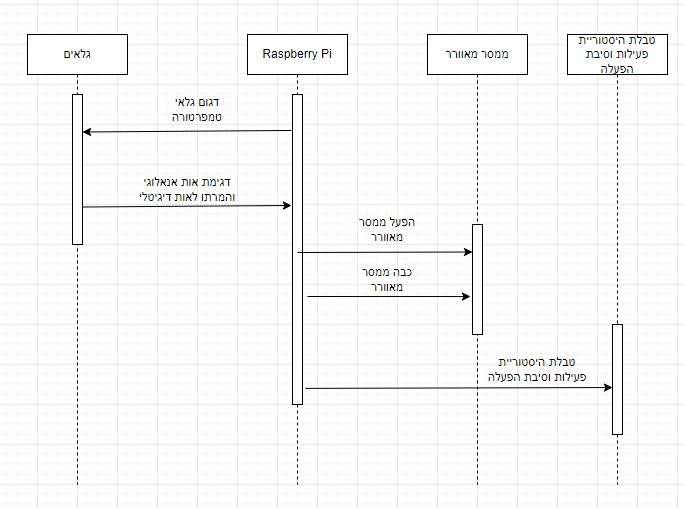
**הפעלת המאוורר**



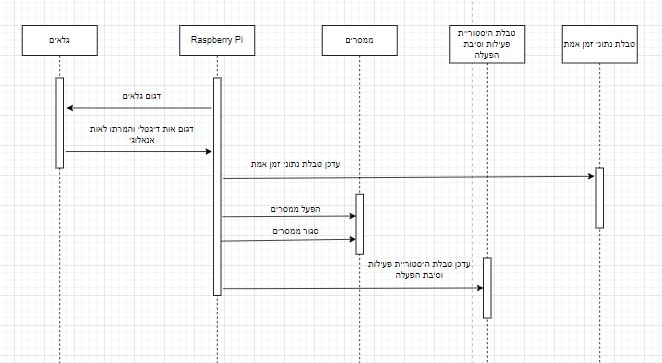
**.**

**דגימה חדשה והפעלת ממסר**

**צפייה בנתוני זמן אמת**



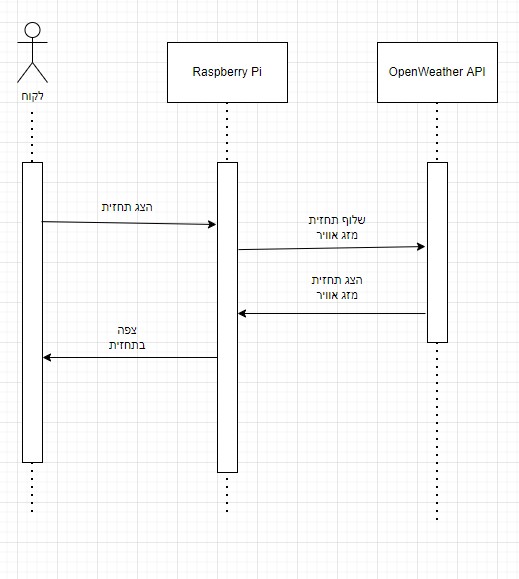
**הפעלת ממסר כללי**



**הפעלה מרחוק (זמן אמת) של המערכת**

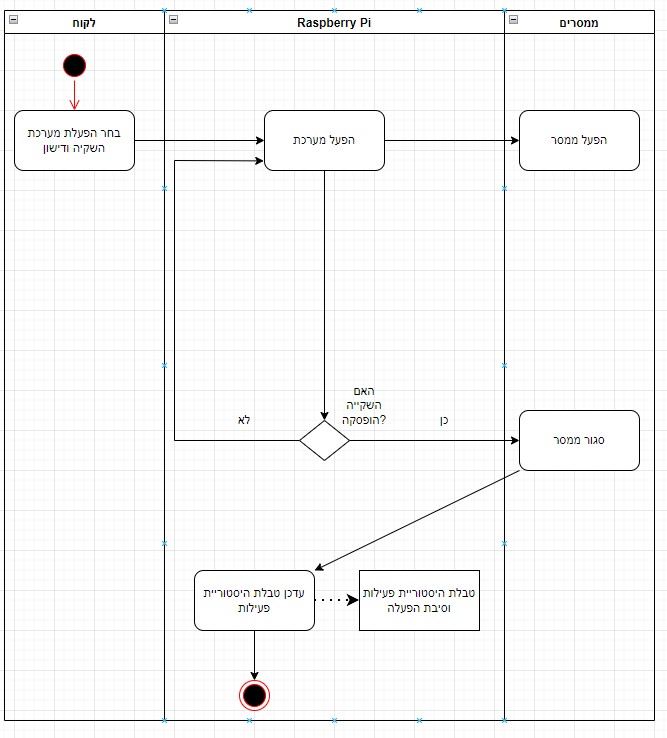


**שליפת תחזית מזג אוויר מAPI**



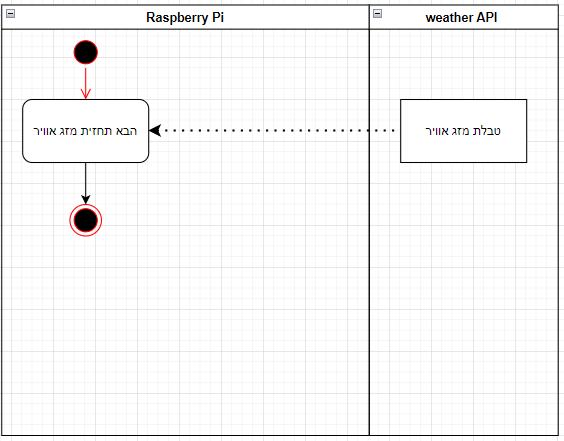
**9.3Data Flow**

**תרשים זרימה – הפעלה מרחוק**



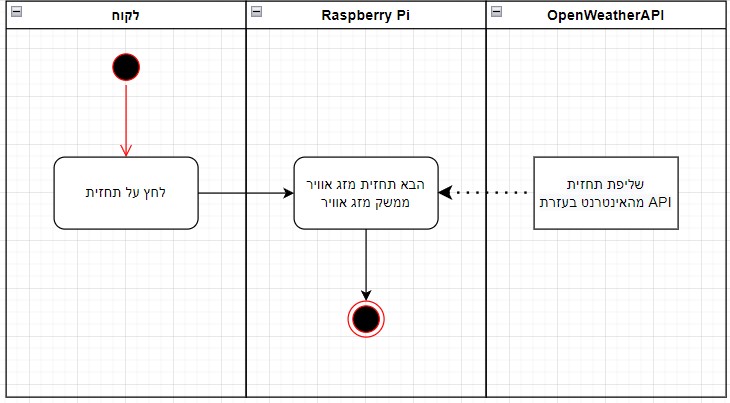
**תרשים זרימה – שליפת תחזית מ- בסיס הנתונים**

טבלת דגימות



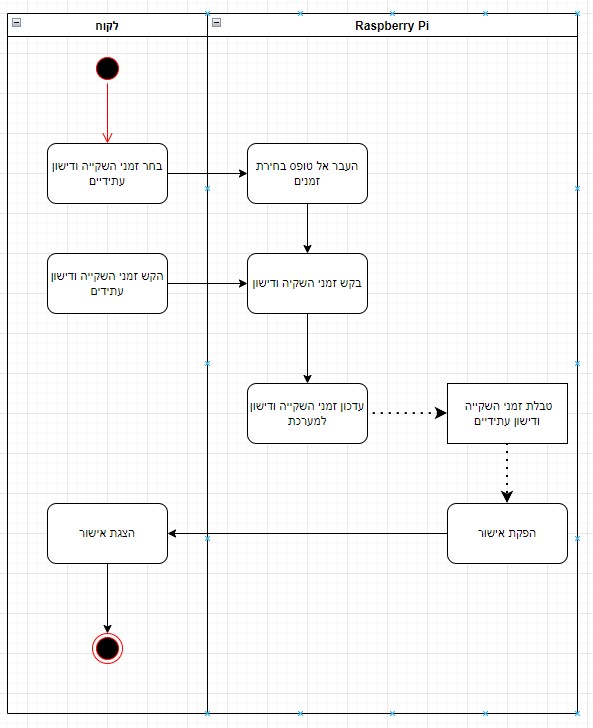
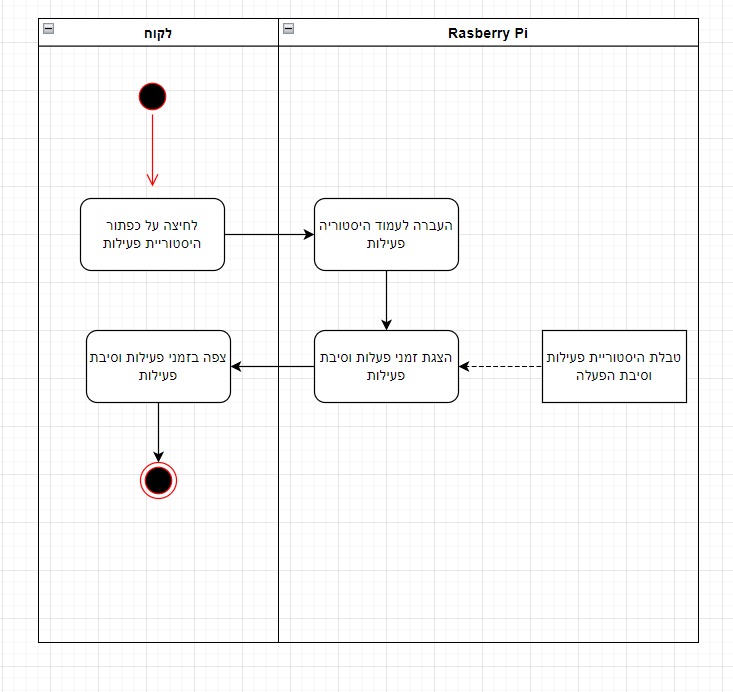
**תרשים זרימה – שליפת תחזית מ- API חיצוני**

כניסה לאפליקציה



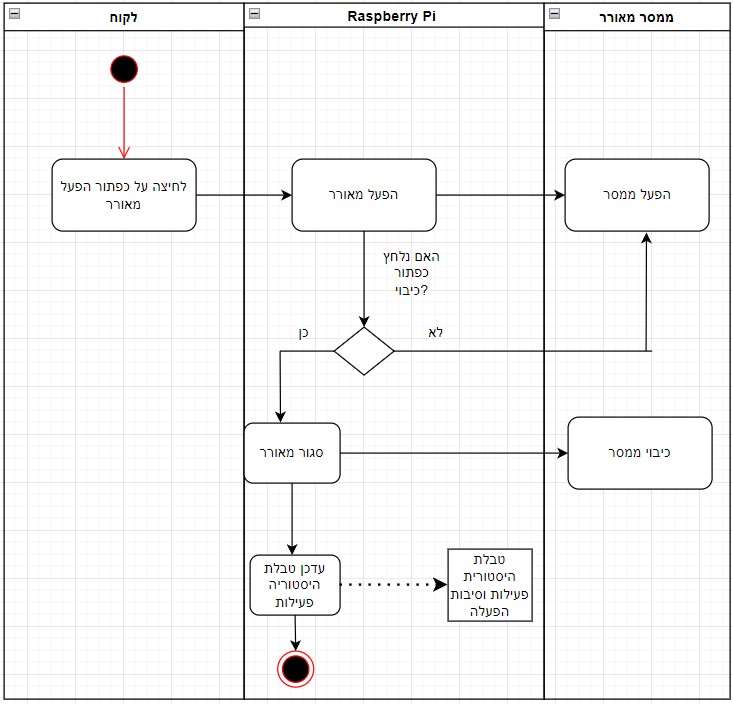
**תרשים זרימה – צפייה בהיסטוריית פעילות**

**תרשים זרימה – עדכון זמני השקיה עתידיים**

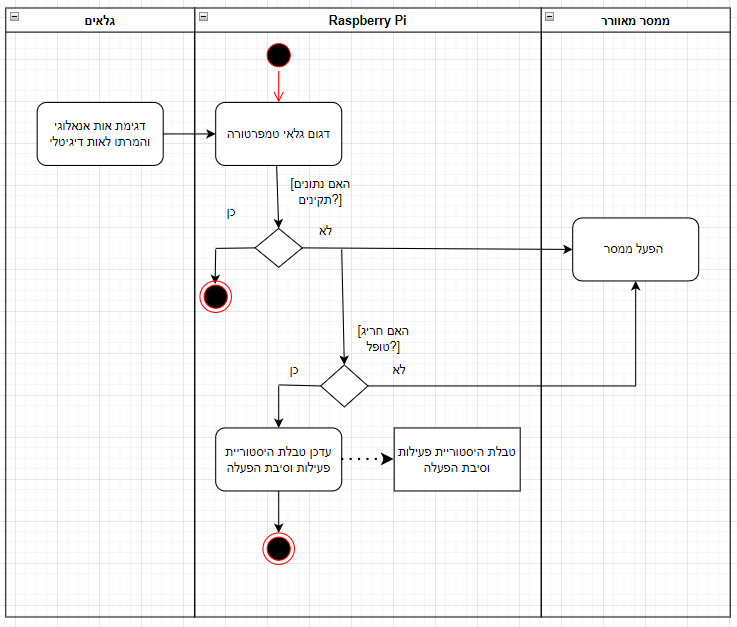


**תרשים זרימה – הפעלה מרחוק מאוורר**

איפוס



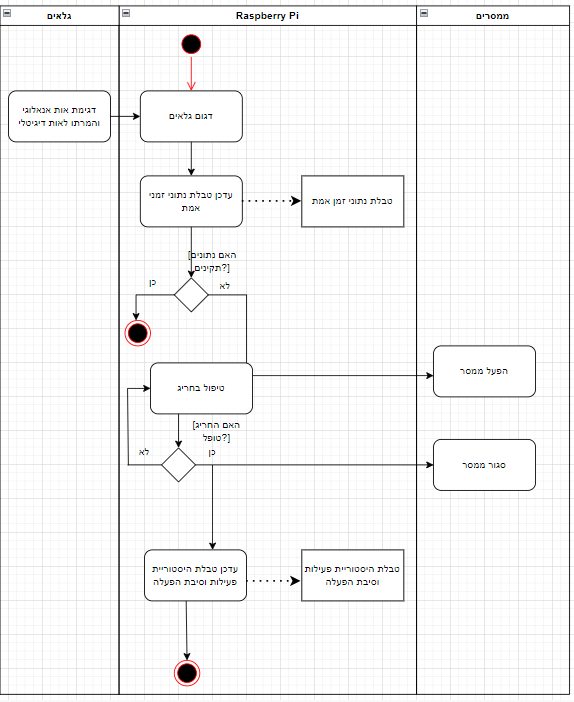
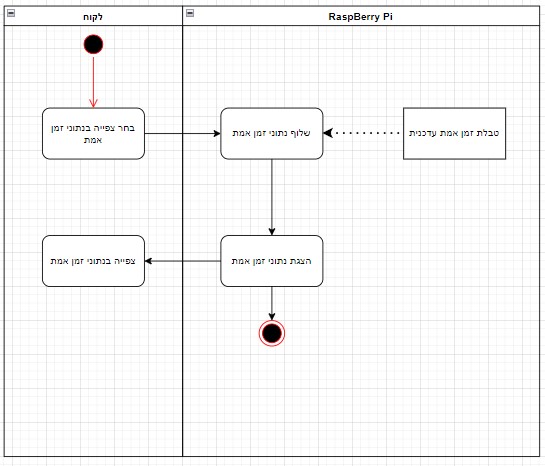
**תרשים זרימה – טיפול בדגימה טמפ' לא תקינה**



**תרשים זרימה – צפייה במערכות שפועלות תרשים זרימה – תרשים כולל של דגימה לא תקינה**

ממסרים פעילים יסומנו כך

בחר הפעלה מרחוק



1. **תיאור המרכיב האלגוריתמי – חישובי**
   1. **הבעיה שבא לפתור**

* מימוש פוטנציאל הגדילה של הצמח – מקנה לצמח את התנאים הביולוגים הרצויים על מנת לממש את הפוטנציאל שלו.
* חסכון בזמן – המערכת אוטומטית ומופעלת באמצעות האפליקציה בזמן אמת, ואינה מצריכה מהמשתמש לבצע פעולה פיזית.
* תיעוד של היסטוריית ההפעלות של המערכת וסיבת ההפעלה.
* טיפול במקרי קיצון (טמפ' גבוהה, יובש וכו') בצורה אוטונומית ללא התערבות המשתמש.
* הפעלה בזמנים קבועים בהתאם לבחירת המשתמש.
  1. **איסוף מידע וניתוחים סטטיסטיים**
* גרף שיציג את נתוני דגימות האור ב-שעה / יום / שבוע / חודש / שנה האחרונים
* גרף שיציג את נתוני דגימות הלחות ב-שעה / יום / שבוע / חודש / שנה האחרונים
* גרף שיציג את נתוני דגימות הלחץ האטמוספרי ב-שעה / יום / שבוע / חודש / שנה האחרונים
* גרף שיציג את נתוני דגימות הטמפרטורה ב-שעה / יום / שבוע / חודש / שנה האחרונים
* הממסר שהופעל הכי הרבה פעמים (מאוורר, לחות איזורית, דישון, השקייה וכו')

1. **אבטחת מידע**

* **שרת:** זיהוי המשתמש לפי שם משתמש וסיסמא ייחודיים, המורכבים מרצפים חוקיים של תווים. סיסמא לגישה מרחוק בעזרת PuTTY ל-Raspberry Pi
* **בקרת גישה:** מתן הרשאות ל- Administrator ולמשתמשי האפליקציה.
* **מסד נתונים:** מתן הרשאת גישה ולעדכון נתונים בטבלאות ל- Administrator.

1. **משאבים הנדרשים לפרויקט**

**12.1 מספר שעות המוקדש לפרויקט, חלוקת עבודה בין חברי הצוות**

* 900 שעות
* **יונתן הרנס** –Backend + Hardware + Database + Frontend
* **עמית שווימר** – Frontend + Database

**12.2 ציוד נדרש**

**רשימת ציוד:**

* ADS115 – A/D 4 port 16 bit
* Analog sensor – גלאי לחות קרקע
* TSL2584TSV – גלאי אור
* DHT21 – גלאי לחות וטמפרטורה
* BMP280-3.3 – גלאי לחץ אטמוספרי
* Rain Sensor – גלאי גשם
* Relay module – ממסר 4 ערוצים
* I2c Expander – בס i2c
* Raspberry Pi 4 4g – בקר\מחשב רספברי כולל קייס וכבלים
* Breadboard
* Wires

**12.3 תוכנות נדרשות**

* Visual Studio 2022
* Visual Studio Code
* MySQL
* Python MQTT
* Node.js + Express
* PuTTY
* Python
* Linux operating system
* Notepad ++
* Heidi SQL

**12.4 ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט**

* React native hooks
* Hardware (I2C, Voltages, IoT)
* Linux
* Node.js
* PuTTY
* Python drivers
* WinSCP

**12.5 ספרות ומקורות מידע**

* Stackoverflow.com
* Youtube.com
* W3School.com
* Wikipedia
* geeksforgeeks.org
* [projects.raspberrypi.org/](https://projects.raspberrypi.org/)

1. **תכנית עבודה ושלבים למימוש הפרויקט**

**אפיון-** ימשך עד 10/06/2022

**עיצוב המערכת וקידוד-** ימשך עד סוף 31/08/2022

**בדיקות ותיקונים-** במהלך חודש 09/2022

**הגשת הפרויקט-** עד לתאריך 10/09/2022

**הגנה על הפרוייקט-** במהלך 10/2022

1. **בדיקות**

**שם הבדיקה: עדכון זמן הפעלה עתידי**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **מס"ד** | **שם הבדיקה** | **תיאור הבדיקה** | **תוצאה רצויה** | **תוצאה מתקבלת** |
| 1 | טעינת מסך הפעלה עתידית | הצג מסך הפעלה עתידית, אם יש כבר לו"ז הצג אותו במקומות הרלוונטיים | הצגת מסך הפעלה עתידית ואם כבר יש לו"ז, הצג אותו בהתאם | הצגת מסך הפעלה עתידית עם כפתורים פעילים בהתאם ללו"ז הנוכחי |
| 2 | שינוי ימי הפעלה עתידים | מתן אפשרות להחלפת ימי ההפעלה העתידיים של המערכת ושינוי צבעם בהתאם | שינוי צבע הכפתור בהתאם לבחירת המשתמש (אדום פעיל וירוק לא פעיל) | צבע הכפתור שונה בהתאם להחלטת המשתמש |
| 3 | שינוי ממסרי הפעלה עתידים | מתן אפשרות להחלפת הממסרים שיפעלו | שינוי מצב הכפתור בהתאם לבחירת המשתמש (ירוק פעיל אפור לא פעיל) | מיקום וצבע הכפתור שונו בהתאם להחלטת המשתמש |
| 4 | שינוי זמן ההתחלה העתידי | פתיחת מקלדת להחלפת זמן ההתחלה ועדכון מלל הכפתור | פתיחת מקלדת ושינוי מלל הכפתור לזמן שנבחר בסיום | מקלדת נפתחה ומלל הכפתור עודכן |
| 5 | שינוי משך זמן ההפעלה העתידי | שינוי מיקום הכפתור על גבי הציר משנה את כמות הזמן שהמערכת תפעל | הזזת הכפתור לימין תעלה את כמות הזמן ולשמאל תקטין | משך הזמן שבו המערכת תפעל משתנה |
| 6 | לחיצה כל כפתור שמירה ועדכון הלו"ז | לחיצה על כפתור שמירה כאשר לא כל הנתונים הוכנסו כראוי לטופס | הצג התראה בהתאם לתא הרלוונטי שאינו מלא כהכרחי | הצגת התראה עם שם הקלט שלא הוכנס כראוי |
| לחיצה על כפתור שמירה כאשר כל הנתונים הוכנסו כראוי לטופס | הצג הודעת אישור ולאחר מכן העברה לדף הבית | הצגת הודעת האישור ומעבר לדף הבית |
| 7 | טעינת מסך הפעלה עתידית | הצג מסך הפעלה עתידית, עם לו"ז ההפעלה העדכני | הצג מסך הפעלה עתידית, עם הכפתורים המתאימים – פעילים, בהתאם ללו"ז ההפעלה העדכני | הצגת מסך הפעלה עתידית, עם הכפתורים המתאימים – פעילים, בהתאם ללו"ז ההפעלה העדכני |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **עדכון זמן הפעלה עתידי** | | | | | |
| **סוג בדיקה** | **מס"ד** | **TR** | **שם הבדיקה** | **תיאור הבדיקה** | **תוצאה רצויה** | **תוצאה מתקבלת** |
| פונקציונליות | 2.1 | 2.1.1 | לחיצה על כפתורי הימים | לחיצה על כפתור המייצג יום יציג את עצמו בצבע ירוק(יום ללא הפעלה) או אדום (יום הפעלה) | שינוי כפתורי הימים עפ"י הצבעים אדום לעובד וירוק ללא עובד | במעבר בין קביעת זמן ההפעלה הכפתור ישנה את צבעו עפ"י הפעלה או ללא הפעלה |

**שם הבדיקה: הצגת מסך היסטורית פעילות**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **מס"ד** | **שם הבדיקה** | **תיאור הבדיקה** | **תוצאה רצויה** | **תוצאה מתקבלת** |
| 1 | לחיצה על כפתור היסטורית פעילות במסך הבית | לחיצה על כפתור היסטוריה פעילות במסך הבית תעביר אותנו למסך היסטורית פעילות | מעבר למסך היסטוריית פעילות | מעבר למסך היסטוריית פעילות |
| 2 | טעינת מסך היסטורית פעילות | קבלת רשימה מעוצבת המכילה בתוכה את הפעמים שהמערכת פעלה ואת סיבת ההפעלה | תוצג רשימה של שעות פעילות המערכת וסיבת הפעלתה | הצגת רשימה מעוצבת עם זמני פעילות וסיבה |
| 3 | צפייה בכל היסטורית הפעילות | אפשרות לגלילה במסך על מנת לאפשר צפייה בכל זמני הפעילות וסיבת של המערכת | אפשרות צפייה בכל ההיסטורית פעילות ע"י גלילה בעמוד | גלילה כלפי מטה מציגה זמנים קודמים שהמערכת פעלה |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **צפייה בכל היסטוריית פעילות: TC1** | | | | | |
| **סוג בדיקה** | **מס"ד** | **TR** | **שם הבדיקה** | **תיאור הבדיקה** | **תוצאה רצויה** | **תוצאה מתקבלת** |
| פונקציונליות | 3.1 | 3.1.1 | צפייה בכל היסטוריית הפעילות | אפשרות לגלילה במסך על מנת לאפשר צפייה בכל זמני הפעילות וסיבה של המערכת | אפשרות צפייה בכל ההיסטורית פעילות ע"י גלילה בעמוד | גלילה כלפי מטה מציגה זמנים קודמים שהמערכת פעלה |
| 3.1.2 | אי יכולת צפייה בהיסטוריית פעילות | אי יכולת גלילה או צפייה בהיסטוריית פעילות בשל אי היסטוריית פעילות | הצגת מסך לבן למשתמש ללא יכולת גלילה | מסך לבן ללא היסטוריית פעילות ויכולת גלילה |

**שם הבדיקה : טעינת מסך הבית**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **מס"ד** | **שם הבדיקה** | **תיאור הבדיקה** | **תוצאה רצויה** | **תוצאה מתקבלת** |
| 1 | הרשאה למיקום | בעת כניסה למערכת המשתמש התבקש לתת למערכת הרשאה למיקום | הקפץ הודעה למשתמש על מתן אישור למיקום למערכת להמשך תפעול האפליקציה | הודעת הרשאת מיקום מהמשתמש להמשך תפעול האפליקציה |
| 2 | הצגת כותרת בהתאם לזמן | ברכת הכותרת תשתנה עפ"י שעת היממה בהתאם למיקום המשתמש | טען ברכת המשתמש עפ"י הזמן הנוכחי כגון : בוקר טוב, ערב טוב וכו' | ברכת Good Morning במהלך שעות הבוקר |
| 3 | הצגת טמפרטורה עפ"י המיקום | טמפרטורה תואמת לטמפרטורה עפ"י מיקום המשתמש | הצג טמפרטורה רצויה עפ"י מיקום משתמש | טמפרטורה מוצגת לפי המיקום היחסי של המשתמש |
| 4 | הצגת אייקון עפ"י תנאי מזג האוויר | הצגת אייקון תואם למזג האוויר עפ"י מיקום המשתמש | הצג את האייקון המתאים למזג האוויר עפ"י מיקום המשתמש | הצגת אייקון המייצג את מזג האוויר עפ"י מיקום המשתמש |
| 5 | הצגת כפתורי מעבר לעמודים עפ"י מצב המערכת | הצגת ארבעת כפתורי מעבר בין דפים צבעים ירוק או אפור עפ"י מצב המערכת (אוטומטי או ידני) | הצג את כל הכפתורים בירוק במידה והמערכת אינה על מצב אוטומטי ואפשרי לעבור לדפי ההפעלה מרחוק או קביעת זמני הפעלה , הצגה באפור במידה והמערכת במצב אוטומטי ולא ניתן לעבור בין העמודים | בזמן שהמערכת במצב אוטומטי כפתור הפעלה וקביעת הזמנים בצבע אפור ואינם עובדים, במצב רגיל הכפתורים ירוקים ונותנים גישה למעבר בין עמודים |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **הרשאה למיקום: TC1** | | | | | |
| **סוג בדיקה** | **מס"ד** | **TR** | **שם הבדיקה** | **תיאור הבדיקה** | **תוצאה רצויה** | **תוצאה מתקבלת** |
| פונקציונליות | 1.1 | 1.1.1 | הרשאה למיקום | הקפץ הודעה למשתמש על מתן אישור למיקום למערכת להמשך תפעול האפליקציה ומתן אישור | הקפץ הודעה למשתמש על מתן אישור למיקום למערכת להמשך תפעול האפליקציה ומתן אישור להפעלה | הודעת הרשאת מיקום מהמשתמש להמשך תפעול האפליקציה ומתן אישור  האפליקציה עובדת עם כל הפיצ'רים הנוגעים למיקום |
| 1.1.2 | אי הרשאה למיקום | הקפץ הודעה למשתמש על מתן אישור למיקום למערכת להמשך תפעול האפליקציה ואי מתן אישור | הקפץ הודעה למשתמש על מתן אישור למיקום למערכת להמשך תפעול האפליקציה | הודעת הרשאת מיקום מהמשתמש להמשך תפעול האפליקציה , ואי מתן אישור  האפליקציה עובדת ללא כל הפיצ'רים הנוגעים למיקום |

1. **בקרת גרסאות**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **שם רכיב** | **סוג רכיב** | **גירסה** | **תאריך** | **שינויים** |
| מסמך אפיון | מסמך | 1.0 | 29/4/2022 | מסמך מקור |
| אפליקצייתMyGarden | קוד | 1.0 | 31/08/2022 | קוד מקור |
| מסמך אפיון | מסמך | 2.0 | 30/09/2022 | עדכון פונקציונליות בפרק 5 ואיפה שנדרש באפיון  החלפת עיצובי מסכים 0בצילומי מסך אמיתיים  פרק 7.7 -  עדכון מבנה בסיס הנתונים פרק 8 -  עדכון פונקציונליות בתרשימים  פרקים 9, 7.5  הוספת פרקים 14, 15, 16 |
| אפליקצייתMyGarden | קוד | 2.0 | 10/10/2022 | תיקון באגים |

1. **קטעי קוד**

* קבלת המידע מהדרייבר ושמירתו בקובץ xlsx שממנו נאסוף את המידע מכל הגלאים:

# Import all libraries we need!  
from smbus import SMBus  
from bme280 import BME280  
import time  
import datetime  
from datetime import date  
from openpyxl import load\_workbook  
  
# Initialise the BME280  
bus = SMBus(1)  
bme280 = BME280(i2c\_dev=bus)  
  
# Take first reading and disgard it to avoid garbage first row  
temperature = bme280.get\_temperature()  
pressure = bme280.get\_pressure()  
humidity = bme280.get\_humidity()  
time.sleep(1)   
  
# Load the workbook and select the sheet  
wb = load\_workbook('/home/pi/Python\_Code/weather.xlsx')  
sheet = wb['Sheet1']  
  
try:  
 while True:  
   
 # Read the sensor and get date and time  
 temperature = round(bme280.get\_temperature(),1)  
 pressure = round(bme280.get\_pressure(),1)  
 humidity = round(bme280.get\_humidity(),1)  
 today = date.today()  
 now = datetime.datetime.now().time()  
  
 # Inform the user!  
 print('Adding this data to the spreadsheet:')  
 print(today)  
 print(now)  
 print('{}\*C {}hPa {}%'.format(temperature, pressure, humidity))  
   
 # Append data to the spreadsheet  
 row = (today, now, temperature, pressure, humidity)  
 sheet.append(row)  
   
 #Save the workbook  
 wb.save('/home/pi/Python\_Code/weather.xlsx')  
  
 # Wait for 10 minutes seconds (600 seconds)  
 time.sleep(600)  
  
finally:  
 # Make sure the workbook is saved!  
 wb.save('/home/pi/Python\_Code/weather.xlsx')  
   
 print('Goodbye!')

* הסקריפט הראשי שיפעיל את כל הדרייברים ויחזיר לנו קובץ json בפורמט של טבלת הדגימות. סקריפט זה גם מוודא שה Raspberry מזהה את כל הכתובות שעל הבס – כלומר שכל הגלאים תקינים ופועלים ועדכון קובץ LOG
* import smbus  
  import json  
  import sys  
  from datetime import datetime  
  sys.path.append('/home/pi/irrsys/drivers')  
  sys.path.append('/home/pi/irrsys/loaders')  
  import bme280 # Temperature & Humidity & Pressurs sensor driver  
  #import aht20 # Temperature & Humidity sensor driver  
  import tsl2584 # light sensor driver  
  #import bh1750 # light sensor driver  
  import ads1115 # analog to digtal driver   
  import pcf8574 # io driver  
  import samplesLoader  
  import relaysLoader  
    
  def logtofile(mesg):  
   now = datetime.now()  
   timestemp = now.strftime("%d-%m-%Y %H:%M:%S")  
   line = timestemp + ',' + mesg  
   #print(line)  
   with open("/home/pi/irrsys/logfile.txt", "a", newline='\n') as myfile:  
   myfile.write(line + '\n')  
   myfile.close()  
    
  def getsensordata():  
   msg={}  
   (temperature,pressure,humidity) = bme280.readBME280All()  
   print ("BME280 Temperature : ", temperature, "C")  
   print ("BME280 Pressure : ", pressure, "hPa")  
   print ("BME280 Humidity : ", humidity, "%")  
   msg["temperature"]=temperature  
   msg["humidity"]=humidity  
   msg["pressure"]= pressure  
   #(temperature,humidity) = aht20.readAHT20()  
   #print ("AHT20 Temperature : ", temperature, "C")  
   #print ("AHT Humidity : ", humidity, "%")  
    
    
   (ch0,ch1) = tsl2584.readTSL2584()  
   print ("Full Spectrum(IR + Visible) :%d lux" %ch0)  
   print ("Infrared Value :%d lux" %ch1)  
   print ("Visible Value :%d lux" %(ch0 - ch1))  
   msg["IR+Visible"]=ch0  
   msg["IR"]=ch1  
   msg["light"]=ch0-ch1  
   msg["ground\_humidity1"] = 0  
   msg["ground\_humidity2"] = 0  
   msg["ground\_humidity3"] = 0  
    
   #luminance=bh1750.readBH1750();  
   #print ("Ambient Light luminance : %.2f lux" %luminance)  
    
   #(raw\_adc,raw\_vlt)=ads1115.readADS1115(ain=0)  
   #print ("Digital Value of Analog Input : %d" %raw\_adc)  
   #print ("Analog Value of Analog Input : %.4f Volt" %raw\_vlt)  
    
    
   #msg["Digital-IN0"]=raw\_adc  
   #msg["Analog-IN0"]=raw\_vlt  
   #pcf8574.RealyTestOn(pcf8574.REL1)  
   #pcf8574.RealyTestOff(pcf8574.REL1)  
   return msg;  
    
    
  def get\_relays\_status():  
   relays\_status = pcf8574.GetRealyStatus()  
   print ("realay status = ",relays\_status)  
   return relays\_status;  
    
    
  def load():  
   samplesLoader.load\_new\_sample(getsensordata())  
   relaysLoader.load\_currently\_active(get\_relays\_status())  
    
  def i2cdetect():  
   global bus,i2cdevices  
   print('in i2cdetect...')  
   alldev=i2cdevices.keys()  
   print("device list:",alldev)  
   dev=[]  
   for device in range(128):  
   try:  
   bus.read\_byte(device)  
   #print(hex(device))  
   dev.append(hex(device))  
   except: # exception if read\_byte fails  
   pass  
   print('device found=',dev)  
   rslt=list(set(alldev)-set(dev))  
   if len(rslt)==0:  
   print('I2C devices found')  
   return True  
   else:  
   print('I2C devices not found:',sorted(rslt))  
   for x in rslt:  
   print(x,":",i2cdevices[x])  
   return False  
    
  def I2cSetup():  
   global bus,i2cdevices  
   bus=smbus.SMBus(1) # /dev/i2c-1  
   i2cdevices={  
   "0x20" : "PCF8574 io relay",  
   #"0x38" : "AHT20 Temperature&Humidity",  
   "0x39" : "TSL2584 light",  
   #"0x48" : "ADS1115 A/D 4 Channel",  
   "0x76" : "BME280 Temp&Hum&pressure"  
   }  
   #if not i2cdetect():  
   # return False  
    
    
  ############################################################  
  ######################## MAIN ##############################  
  ############################################################  
  #logtofile("start program")  
  def main():  
   logtofile("start program")  
   I2cSetup()  
   i2cdetect()  
   load()  
   logtofile("end program")  
    
    
  if \_\_name\_\_ == '\_\_main\_\_':  
   main()  
  #logtofile("after i2cdetect")  
  #getsensordata()  
  #logtofile("end program")  
  #initvar()  
  #SpiSetup()  
  #GpioSetUp()  
  #HbridgeSetup()  
  #I2cSetup()  
    
  #i2cdetect()  
  #print(vars())  
    
  #cmd = '{"CMD":"RunMotor","DIR":"c","SPEED":c}'  
    
  #logtofile('Start of program')  
  #runserver()
* REST – API, כל השירותים שניתנים מטבלת היסטורית פעילות

יושב תחת MyGarden/Server/Services/scheduleActivation.js -

const db = require("./db");

const helper = require("../helper");

const config = require("../config");

/\*Get latest schedule \*/

async function getScheduleActivation() {

  const rows = await db.query(`SELECT \*

  FROM schedule\_activation order by id Desc Limit 1`);

  const data = helper.emptyOrRows(rows);

  return data;

}

/\*Create a new schedule. parameter data is json object which consists of all

the data nesseccery for schedule\_activation table in DB\*/

async function create(data) {

  /\*the query for posting a new schedule, unfortunately couldn’t use a SQL proc

And needed to write it by hand…\*/

  const result = await db.query(

     `INSERT INTO schedule\_activation (start\_hour, time\_to\_live,sunday,monday,

        tuesday,wednesday,thursday,friday,saturday,air\_sys,water\_sys,light\_sys,fertelize\_sys) VALUES ('${data.startTime}', ${data["timeToLive"]},

    ${data["weekSchedule"]["sunday"]}, ${data["weekSchedule"]["monday"]}, ${data["weekSchedule"]["tuesday"]},

     ${data["weekSchedule"]["wednesday"]}, ${data["weekSchedule"]["thursday"]}, ${data["weekSchedule"]["friday"]},

      ${data["weekSchedule"]["saturday"]}, ${data["systemsToActivate"]["air\_sys"]},

       ${data["systemsToActivate"]["water\_sys"]}, ${data["systemsToActivate"]["light\_sys"]},

        ${data["systemsToActivate"]["fertelize\_sys"]});`

  );

  let message = "Error in creating a new schedule\_activation";

  /\*if there was no error\*/

  if (result.affectedRows) {

    message = "1 Row Added to schedule\_activation.";

  }

  return { message };

}

module.exports = {

  getScheduleActivation,

  create,

};

* REST – API, יושב תחת –

MyGarden/Server/routes/scheduleActivation.js

אחראי על ניתוב הבקשות לurl - המתאים

const express = require("express");

const router = express.Router();

const schedule\_activation = require("../services/scheduleActivation"); //importing all the services

/\* GET latese schedule \*/

/\*first param is route url, and second is the async function that posts.

the function req is the body (if needed) and the query(if needed)\*/

router.get("/", async function (req, res, next) {

  try {

    res.json(await schedule\_activation.getScheduleActivation(req.query));

  } catch (err) {

    console.error(`Error while getting schedule activation `, err.message);

    next(err);

  }

});

/\*Post new schedule\*/

/\*first param is route url, and second is the async function that posts.

the function req is the body (if needed) and the query(if needed)\*/

router.post("/", async function (req, res, next) {

  try {

    res.json(await schedule\_activation.create(req.body));

  } catch (err) {

    console.error(

      `Error while creating schedule\_activation new data`,

      err.message

    );

    next(err);

  }

});

module.exports = router;

* REST – API, קונפיגורציה של שרת ה- REST

const config = {

  db: {

    host: "192.168.1.157",

    user: "irruser",

    password: "irrpass",

    database: "mygarden",

  },

  listPerPage: 10,

};

module.exports = config;

* REST – API, יושב תחת –

MyGarden/Server/index.js

מייבא את כל ה- routes ומפעיל את השרת

const express = require("express");

const app = express();

const port = 3000;

const samplesRouter = require("./routes/samples");

const activationHistoryRouter = require("./routes/activationHistory");

const currentlyActiveRouter = require("./routes/currentlyActive");

const remoteActivationRouter = require("./routes/remoteActivation");

const scheduleActivationRouter = require("./routes/scheduleActivation");

const SysModRouter = require("./routes/SysMod");

app.use(express.json());

app.use(

  express.urlencoded({

    extended: true,

  })

);

app.get("/", (req, res) => {

  res.json({ message: "ok" });

});

app.use("/samples", samplesRouter);

app.use("/activationHistory", activationHistoryRouter);

app.use("/currentlyActive", currentlyActiveRouter);

app.use("/remoteActivation", remoteActivationRouter);

app.use("/scheduleActivation", scheduleActivationRouter);

app.use("/sysMod", SysModRouter);

/\* Error handler\*/

app.use((err, req, res, next) => {

  const statusCode = err.statusCode || 500;

  console.error(err.message, err.stack);

  res.status(statusCode).json({ message: err.message });

  return;

});

app.listen(port, () => {

  console.log(`server listening at http://localhost:${port}`);

});

* Frontend – מודול שמציגה בצורה דינאמית את הגרפים במסך Analytics

import moment from "moment/moment";

import { HStack } from "@react-native-material/core";

import Chart from "../Components/Chart";

import DropDown from "../Components/DropDown";

import { StyleSheet } from "react-native";

/\*data to be rendered in flatlist\*/

export const flatlistData = (timeTypeText, samples) => [

  {

    timeName: timeTypeText,// "Hour/day/week..."

    samples: samples,// the last year samples

    dataKey: "light",//the name of the col in the db

    dataName: "Light",

  },

  {

    timeName: timeTypeText,

    samples: samples,

    dataKey: "humidity",

    dataName: "Humidity",

  },

  {

    timeName: timeTypeText,

    samples: samples,

    dataKey: "temperature",

    dataName: "Temperature",

  },

  {

    timeName: timeTypeText,

    samples: samples,

    dataKey: "pressure",

    dataName: "Air Pressure",

  },

];

/\*X axis format, each time choosen has a diff format\*/

export const xTicksFormat = (timeName) => {

  let res = "";

  switch (timeName) {

    case "Hour":

    case "Day":

      res = (t) => `${moment(t).format("HH:mm")}`;

      break;

    case "Week":

    case "Month":

      res = (t) => `${moment(t).format("DD/MM")}`;

      break;

    case "Year":

      res = (t) => `${moment(t).format("MM/YY")}`;

  }

  return res;};

/\*text for the appropriate measurement unit\*/

export const measurementUnit = (dataName) => {

  let res = "";

  switch (dataName) {

    case "Light":

      res = "Lm";

      break;

    case "Humidity":

      res = "%";

      break;

    case "Temperature":

      res = "C";

      break;

    case "Air Pressure":

      res = "Mb";

      break;

  }

  return res;

};

/\*JSX element to be rendered for each col in flatlist\*/

export const Item = ({ timeName, samples, dataKey, dataName }) => (

  <HStack fill center>

    <Chart

      timeName={timeName}

      samples={samples}

      dataKey={dataKey}

      dataName={dataName}

    />

  </HStack>

);

/\*sending data to the element to be rendered\*/

export const renderItem = ({ item }) => (

  <Item

    timeName={item.timeName}

    samples={item.samples}

    dataKey={item.dataKey}

    dataName={item.dataName}

  />

);

/\*static header the only renders one time in flatlist\*/

export const header = (timeTypeText, timeTypeToPapa) => {

    return (

      <HStack fill center style={styles.header}>

        <DropDown state={timeTypeText} timeTypeToPapa={timeTypeToPapa} />

      </HStack>

    );

  };

  const styles = StyleSheet.create({

    header:{

        paddingBottom:30

    }

  })

* Frontend – יצירה של גרפים באופן דינאמי באמצעות Flatlist

  return samples ? (

    <>

        <FlatList

          keyExtractor={(item) => item.dataKey}

          ListHeaderComponent={header(timeTypeText, timeTypeToPapa)}

          data={flatlistData(timeTypeText, samples)}

          renderItem={renderItem}

        />

    </>

  ) : (

    <Text>Loading</Text>

  );