

המחלקה להנדסת
חשמל ואלקטרוניקה

אוניברסיטת
אריאל
בשומרון

הפקולטה להנדסה
המחלקה להנדסת חשמל ואלקטרוניקה

מערכת ניטור ששולטת על חיישנים

פרויקט גמר המהווה חלק מהדרישות לתואר B.Sc.

מוגש ע"י:

רמי מוטלק

ת.ז. : 209019538

מגד סעסע

ת.ז. : 318612314

אחראי אקדמי: דר דאניל רוזבן

דצמבר 2023

תשפ"ג

תקציר

מערכת ניטור תעשייתית מבוססת ענן זוהי תשתית תוכנה מבוססת יישומים, המאחסנת נתונים בענן אליהם ניתן לגשת דרך האינטרנט. במקום לשמור קבצים בכונן קשיח או בהתקן אחסון מקומי, מערכת מבוססת ענן מאפשרת לשמור אותם במסד נתונים מרוחק. כל עוד למכשיר אלקטרוני יש גישה לאינטרנט, ניתן לגשת בקלות לנתונים ולתוכנות להפעלה, כדי לעבוד בצורה יותר יעילה, חסכונית ומודרנית יש את הצורך בלהתקין מערכת חכמה עם חיישנים חכמים אשר ינטרו כל שינוי במערכת כמו טמפרטורה, לחות עוצמת אור וכמות מים וגם קבלת התראות בהתאם להגדרות שמתאימים לנו. בזכות מערכת ניטור נוכל לקבל התרעות על שינויים במשאבי המערכת כדי שנוכל להגיב ולטפל במהירות

כולנו יודעים שתקלות קורות כל הזמן לפיכך בכל מערכת יש צורך לשלוח בתפעול נכון על מנת ששגיאה אפשרית לא תשפיע בסופו של דבר על השירות שניתן למשתמש, כדי לזהות ולמנוע תקלות כדאי להשתמש בשירותי מערכת ניטור.

מערכת מורכבת מחיישנים שעושות ניטור ובקרה על בית עסק חממה לגידול שתילים כדי להגדיל שתילים בתנאים הכי טובים ובית העסק יצליח

יש חמישה חיישנים:

-**חיישן אור**: חיישן אור הוא חיישן שבודק מצב האור בחממה מהכי בהיר להכי כהה

-**חיישן לוחות** : חיישן שבודק אחוז הלוחות בחממה

-**חיישן טמפרטורה** : חיישן שבודק את הטמפרטורה בחממה

-**חיישן מרחק**: חיישן שבודק את המרחק מהרצפה לגג קדי לדעת מה האורך של השתילים

- **חיישן מפלס מים** : חיישן שבודק את רמת המים

החיישנים מחוברים ל Arduino Uno מיקרו בקר שמספק לחיישנים מתח ומפעיל אותם Arduino Uno מחובר ל (raspberry pi) ומעלה את כל התוצאות שמתקבלות מהחיישנים בזמן אמת

הכרת תודה

מגד סעסע: תודה למשפחתי שתמכה בי במהלך הלימודים האקדמיים, תודה לדר דניאל רוזבן מנחה הפרויקט שלימד, חניך ותרם מהידע והזמן שלו במהלך עשיית הפרויקט. בזכות הפרויקט, התרגשתי להיכנס לעולם ההנדסה והחשמל באופן ייחודי ומעשי. ישנם מספר נקודות שאני מעורר לך תודה במיוחד: ראשית ושוב, תודה על ההזדמנות להשתתף בפרויקט מעשי. זה הביא לי חוויות רבות והבנה עמוקה יותר של עולם ההנדסה. העבודה הפרקטית היא מקור ללמידה לא פחות חשוב מהשיעורים התיאורטיים, ואני שמח על הזדמנות זו שניתנה לי.

שנית, תודה על ההנחיה המקצועית שסיפקת לי ולשותף שלי. היכולת שלך להעביר את הידע בצורה ברורה ומובנת עזרה לנו להתמודד עם אתגרים מקצועיים ולהתקדם בפרויקט בצורה יעילה. לבסוף, רק רציתי להודות על ההשקעה שלך בכל שלב של הפרויקט. האנרגיה והמחויבות שלך הביאו לתוצאות טובות, ואני שמח לומר שלמדתי המון בתקופה הזו.

תודה רבה על ההזדמנות המדהימה ועל המקצועיות שלך. אני בטוח שהניסיון והידע שרכשתי ישמשו אותי גם בעתיד.

רמי מוטלק: אני, רמי, סטודנט לתואר הנדסת חשמל ואלקטרוניקה בשנה האחרונה, רוצה להביע את תודתי העמוקה והמרובה לך על ההדרכה המרתקת שלך במהלך הפרויקט שהושקע בו בשנת הלימודים האחרונה.

סיום הפרויקט הוא רגע מרגש ומשמעותי במהלך הלימודים שלי, ואני מרגיש חובה להודות לך על הליווי המקצועי והמופקד שלך. ההכוונה, ההתעקשות והידע העמוק שהבאת איתך היו הפקטורים המכריעים להצלחת הפרויקט ולתחושת ההתמודדות שלי עם אתגרים טכניים מורכבים.

אני מודה לך על הזמן והמחויבות שהקדשת לי ולמגד סעסע. כמרצה, השקפתך המקצועית והיכולת שלך להעביר את הידע והניסיון שלך בצורה ברורה ומעוררת השראה ניכרים בכל שלב של הפרויקט. השילוב בין התרסה לידע התיאורטי יצרו סביבת למידה מעניינת ומשדרת אתגר.

אני מרגיש שהתפתחתי כמהנדס וכאדם במהלך הפרויקט, ואני מכיר בהערכת עמיתים שהמצויה בי כיום מסתייעת בהצלחתי. התעריפותי לך על האנרגיה וההקפצה שביצעתי, ואני מקווה להמשיך לגדול ולהתפתח תחת נציבתך.

תודה על המסע המקצועי והאישי המרתק, ואני מצפה למצוא עצמי חוזר על ההילה המושגית שלך בעולם ההנדסה.

תוכן עניינים

6	אבולוציה של טכנולוגיית החיישנים	1.1.
6	מיקרו-בקר : Arduino Uno	1.2.
7	ארדואינו אונו	1.2.1.
8	Raspberry Pi	1.2.2.
9	שירותי ענן לניטור מרחוק	1.3.
9	היסטוריית החיישנים	1.4.
9	חיישן טמפרטורה	1.4.1.
9	חיישן מפלס מים	1.4.2.
10	חיישן אור	1.4.3.
10	חיישן מרחק (אולטרסאונד)	1.4.4.
10	חיישן לחות	1.4.5.
11	שליטה מרחוק	1.5.
11	HTTP(פרוטוקול היפרטקסט)	1.5.1.
12	MQTT(העברת טלמטריה בתור הודעות)	1.5.2.
15	מטרת הפרויקט	2.
15	תנאי התכנון	2.1.
16	תרשים זרימה	2.2.
17	תיאור המערכת	
18	הסבר כללי על המערכת	2.3.
19	תכנון החומרה	2.4.
19	מרכיבי עיצוב החומרה	2.4.1.
20	ממשק משתמש	2.4.2.
20	מארז והגנה	2.4.3.
20	מטריצה ובדיקות	2.4.4.
21	עיצוב PCB	2.4.5.
21	תכנון התוכנה	2.5.
21	עיצוב התוכנה של המערכת	2.5.1.
22	ממשק משתמש	2.5.2.
23	בדיקה של המערכת	2.5.3.
23	תיעוד	2.5.4.
23	מימוש המערכת	3.
27	מימוש החומרה	3.1.
36	מימוש התוכנה	3.2.
58	ניסויים ובדיקות	4.

64	מערך הניסוי	4.1.
64	מכשירי המדידה	4.1.1.
65	רכיבי המערכת	4.1.2.
66	תוצאות ומסקנות	5.
67	תוצאות הניסויים/תוצאות המדידות/תוצאות הפעלת המערכת	5.1.
68	מסקנות הפרויקט	5.2.
68	סיכום	5.3.
69	רשימות	6.
69	רשימת טבלאות	6.1.
69	רשימת איורים	6.2.
70	נספחים	7.
70	נספח א': נתוני החישובים	7.1.
71	מקורות ספרותיים	8.

1. מבוא

בעולם המקושר יותר ויותר, הדרישה למערכות בקרה וניטור חכמות עלתה משמעותית. השילוב של חיישנים, מיקרו-בקרים ושירותי ענן סלל את הדרך לפרויקטים מתוחכמים המשפרים את היכולת שלנו להבין ולתקשר עם הסביבה. פרויקט זה מתמקד בפיתוח של מערכת בקרה ניטור ושליטה מקיפה המשתמשת בטכנולוגיות מתקדמות, כולל Raspberry Pi ו-Arduino Uno, כדי לאסוף נתונים מחיישנים שונים. לאחר מכן, הנתונים שנאספו מועברים בצורה חלקה לשירותי ענן, ומציעים למשתמשים את הנוחות של גישה וניתוח בזמן אמת.

1.1. אבולוציה של טכנולוגיית החיישנים

טכנולוגיית החיישנים עברה התקדמות מדהימה במהלך השנים, ושינתה את הנוף של מערכות רכישת נתונים ובקרה. מצמדים תרמיים ונגדי פוטו ועד חיישנים קוליים מודרניים וגלאי לחות, האבולוציה מונעת על ידי הצורך בדיוקנות, אמינות ורבגוניות. חיישנים אלו מהווים את עמוד השדרה של הפרויקט שלנו, ומספקים מדידות מדויקות של פרמטרים סביבתיים מרכזיים.

1.2. מיקרו-בקר: Arduino Uno

הבחירה במיקרו-בקר ממלאת תפקיד מרכזי בפונקציונליות של הפרויקט שלנו. ה-Arduino Uno, הידוע בפשטותו וביכולותיו בזמן אמת, משמש להתממשקות עם החיישנים ולעיבוד נתונים ראשוני. מצד שני מחשב לוח, ה-Raspberry Pi, עם יכולות העיבוד החזקות שלו ומערכת ההפעלה שלו, משמש כרכזת מרכזית לצבירה של נתונים ותקשורת עם שירותי ענן. שילוב זה מאפשר ניצול סינרגטי של החוזקות של כל פלטפורמה.

1.2.1. ארדואינו אנו

Arduino Uno הוא לוח מיקרו-בקר המיועד לקלות שימוש ולרבגוניות בבניית אב טיפוס אלקטרוני ופרויקטי עשה זאת בעצמך. להלן תכונות ורכיבים מרכזיים של Arduino Uno:

1. מיקרו-בקר: הלב של Arduino Uno הוא המיקרו-בקר ATmega328, שאחראי על ביצוע התוכנית ואינטראקציה עם מכשירים מחוברים.
2. פינים דיגיטליים ואנלוגיים: Arduino Uno כולל קבוצה של פיני כניסה/פלט דיגיטליים ואנלוגיים המאפשרים התממשקות עם רכיבים אלקטרוניים שונים כגון חיישנים, מפעילים וצגים.
3. ספק כוח: ניתן להפעיל את הלוח באמצעות חיבור USB, ספק כוח חיצוני או סוללה. הוא כולל גם ווסת מתח המבטיח אספקת חשמל יציבה להתקנים המחוברים.
4. ממשק USB: Arduino Uno מצויד בממשק USB לתכנות ותקשורת עם מחשב. זה מפשט את התהליך של העלאת קוד וניפוי באגים.
5. מהירות שעון: ה-ATmega328 ב-Arduino Uno פועל ב-16 מגה-הרץ, ומספק כוח עיבוד מספיק למגוון רחב של יישומים.
6. פלטפורמת קוד פתוח: Arduino Uno היא חלק מפלטפורמת הקוד הפתוח Arduino, מה שאומר שהסכמטיקה וקוד המקור שלה זמינים באופן חופשי. פתיחות זו מעודדת שיתוף פעולה ופיתוח של מערכת אקולוגית עצומה של ספריות ופרויקטים שנתרמו מהקהילה.
7. סביבת פיתוח משולבת Arduino Uno (IDE): מתוכנת באמצעות Arduino IDE, סביבה ידידותית למשתמש התומכת בשפות התכנות C ו-C++. זה מפשט את תהליך הכתיבה, הקומפילציה והעלאת קוד ללוח.

Arduino Uno נמצא בשימוש נרחב על ידי חובבים, סטודנטים ואנשי מקצוע ליצירת אב טיפוס של פרויקטים אלקטרוניים בשל הפשטות, הנגישות והתמיכה הקהילתית הנרחבת שלו.

1.2.2 Raspberry Pi

Raspberry Pi היא סדרה של מחשבים עם לוח יחיד המיועדים למטרות חינוכיות, פיתוח מערכות משובצות ופרויקטים שונים של עשה זאת בעצמך. להלן תכונות ורכיבים מרכזיים של Raspberry Pi טיפוס:

1. Raspberry Pi System-on-Chip (SoC) מופעל על ידי Broadcom system-on-chip (SoC), המשלב את ה-CPU, GPU, RAM, ורכיבים אחרים על שבב בודד. דגמים שונים עשויים להכיל SoCs שונים.

2. כוח עיבוד: דגמי Raspberry Pi משתנים מבחינת כוח העיבוד, כאשר גרסאות מאוחרות יותר כוללות מעבדים מרובי ליבות. ל-Raspberry Pi 4 יש מעבד ARM Cortex-A72 ארבע ליבות.

3. זיכרון (RAM): לוחות Raspberry Pi מגיעים עם כמויות שונות של זיכרון RAM, בדרך כלל נע בין GB1 ל-GB8. זיכרון RAM מספיק מאפשר ריבוי משימות וטיפול ביישומים מורכבים יותר.

4. אחסון: Raspberry Pi מסתמך על כרטיסי microSD לאחסון. מערכות הפעלה ויישומים מותקנים על כרטיסים אלה, המספקים את שטח האחסון הדרוש למכשיר.

5. GPIO Pins (Input/Output למטרות כלליות): בדומה ל-Arduino, Raspberry Pi כולל פני GPIO המאפשרים למשתמשים להתחבר ולתקשר עם התקנים חיצוניים, מה שהופך אותו למתאים לממשקי חומרה ופרויקטים של IoT (Internet of Things).

6. יציאות USB: Raspberry Pi כולל יציאות USB לחיבור ציוד היקפי כגון מקלדות, עכברים, אחסון חיצוני והתקני USB אחרים.

7. יציאת HDMI: לרוב דגמי Raspberry Pi יש יציאות HDMI, המאפשרות חיבור של צגים לפלט וידאו.

8. רשת: לוחות Raspberry Pi מצוידים ביציאות Ethernet או ב-Wi-Fi מובנה, מה שמקל על קישוריות רשת לתקשורת עם מכשירים אחרים או גישה לאינטרנט.

9. תמיכה במערכת הפעלה: Raspberry Pi תומך במערכות הפעלה שונות המבוססות על לינוקס, כולל Raspbian (הידועה כיום כ-Raspberry Pi OS), אובונטו ואחרות.

המחיר הסביר של Raspberry Pi, הגודל הקומפקטי והרבגוניות הופכים אותו לבחירה פופולרית למטרות חינוכיות, אוטומציה ביתית, מרכזי מדיה ומגוון רחב של פרויקטי מחשוב משובצים.

1.3. שירותי ענן לניטור מרחוק

שילוב שירותי הענן מציג מימד חדש לפרויקט, המאפשר ניטור מרחוק ואחסון נתונים. באמצעות מינוף פלטפורמות כמו, AWS, Azure Thingsboard או Google Cloud, משתמשים יכולים לגשת לנתונים ולשלוט על המערכת בזמן אמת מהחיישנים מכל מקום בעולם. תשתית הענן מבטיחה אבטחת נתונים, מדרגיות ומאפשרת תקשורת חלקה בין המערכת המקומית לממשק המשתמש.

1.4. היסטוריית החיישנים

ניתן לייחס את שורשי הפרויקט שלנו להתפתחויות המוקדמות בטכנולוגיית החיישנים ולהופעת המיקרו-בקרים. באמצע המאה ה-20, הופעתם של הצמדים התרמיים והחיישנים הפוטואלקטריים הראשונים סימנה את תחילתו של עידן חדש במכשור. ככל שהמיקרו-בקרים התפתחו בשנות ה-70 וה-80, השילוב שלהם ביישומים שונים סלל את הדרך למערכות בקרה מתוחכמות יותר.

הזמינות הנרחבת והזמינות של מיקרו-בקרים, כגון Arduino Uno, במאה ה-21 אפשרה את הגישה לפיתוח מערכות משובצות. במקביל, ה-Raspberry Pi, שהוצג ב-2012, הביא את הכוח של מחשב מן המניין לתחום הפרויקטים המשובצים. אבני דרך אלו הציבו את הבמה ליצירת מערכות מורכבות וחכמות, כמו זו המוצעת בפרויקט זה.

1.4.1. חיישן טמפרטורה

חיישן הטמפרטורה הוא מרכיב חיוני המודד את טמפרטורת הסביבה של הסביבה. תוך שימוש בעקרונות כמו צמדים תרמיים או תרמיסטורים, החיישן ממיר שינויים בטמפרטורה לאותות חשמליים. ה-Arduino Uno מעבד את האותות הללו, ומספק קריאות טמפרטורה מדויקות לניתוח נוסף.

1.4.2. חיישן מפלס מים

חיישן מפלס המים משתמש במוליכות חשמלית משתנה או בטכנולוגיה קולית למדידת עומק המים. כאשר המים באים במגע עם החיישן, המוליכות משתנה, או שגלים קוליים משתקפים בצורה שונה, מה שמאפשר ל-Arduino Uno לקבוע את מפלס המים במדויק.

1.4.3. חיישן אור

חיישן האור, המבוסס לרוב על התנגדות פוטו, מזהה את עוצמת האור הסביבתי. ההתנגדות המשתנה של הנגד הפוטו בתגובה לשינויי האור מומרת לאות מתח, מה שמאפשר ל-Arduino Uno לכמת את רמות האור בסביבה.

1.4.4. חיישן מרחק (אולטראסאונד)

חיישן המרחק האולטראסוני משתמש בגלי קול כדי למדוד את המרחק בין החיישן לעצם. על ידי פליטת פולסים קוליים ומדידת הזמן שלוקח להד לחזור, ה-Arduino Uno מחשב את המרחק במדויק, ומספק מידע על הסביבה.

1.4.5. חיישן לחות

חיישן הלחות מודד את תכולת הלחות באוויר. שימוש בחומר רגיש ללחות, שינויים בתכונות החשמליות מתרחשים על סמך רמות הלחות. ה-Arduino Uno מפרש את השינויים הללו, ומייצר קריאות לחות מדויקות לניטור סביבתי מקיף.

השילוב של חיישנים אלו עם Arduino Uno ו-Raspberry Pi, המחוברים לשירותי ענן, מקים מערכת בקרה ניטור ושליטה חזקה. גישה בינתחומית זו, המשלבת טכנולוגיית חיישנים, יכולות מיקרו-בקר ומחשוב ענן, פותחת אפיקים לבקרה וניתוח סביבתיים חכמים ויעילים. הפרויקט מתיישב עם מסלול ההתקדמות הטכנולוגית, ותורם לאבולוציה של מערכות חכמות לעולם מחובר. הפיתוח של מערכת הבקרה והניטור המוצגת בפרויקט זה נותן מענה לכמה צרכים ופערים קריטיים במערכות קיימות, תורם להתקדמות בחישה סביבתית, נגישות לנתונים וניטור מרחוק.

1.5. שליטה מרחוק

ניהול וניטור מרחוק של נתונים מחיישנים באמצעות שירותי ענן. זה כולל איסוף נתונים בזמן אמת מחיישנים, שליחתם לענן לעיבוד וניתוח, ואפשרות שליטה ותגובה מהירה דרך ממשק ווב או אפליקציה ניידת. המערכת מאפשרת ניהול יעיל ומדויק של הנתונים, ומתן תגובה מהירה לאירועים והתראות.

1.5.1 HTTP (פרוטוקול היפרטקסט)

HTTP - הוא הפרוטוקול הבסיסי לתקשורת נתונים ב World Wide Web- פותח על ידי Tim Berners-Lee ב CERN- בשנת 1989, הוא מהווה את הבסיס לכל חילופי נתונים באינטרנט והוא פרוטוקול המיועד להעברת מסמכי היפרטקסט, שהם אבן היסוד של יישומי אינטרנט.

HTTP - פועל על מודל שרת-לקוח, שבו לקוח (כמו דפדפן אינטרנט) מבקש מידע ושרת מגיב לבקשה זו.

-הוא משתמש בפרוטוקול חסר מצב, כלומר כל זוג בקשה-תגובה הוא עצמאי; השרת אינו שומר על מצב כלשהו בין בקשות שונות מאותו לקוח.

-הודעות HTTP מורכבות משורת בקשה שיטה, כתובת URL, גרסת פרוטוקול, כותרות בקשה או תגובה וגוף.

בפרויקטים שלנו HTTP, משמש בדרך כלל לניטור ובקרה מרחוק של מכשירים באמצעות ממשקי אינטרנט. לדוגמה, שרת HTTP יכול להיות משולב במכשיר, ומאפשר למשתמשים לתקשר איתו או לאחזר נתונים באמצעות דפדפני אינטרנט סטנדרטיים.

-ממשקי API של HTTP חיוניים לבניית יישומי IoT (האינטרנט של הדברים) שבהם מכשירים צריכים לתקשר עם שירותי ענן לצורך ניתוח נתונים, אחסון או עיבוד נוסף.

1.5.2 MQTT (העברת טלמטריה בתור הודעות)

- MQTT הוא פרוטוקול הודעות קל משקל, אידיאלי עבור יישומי IoT. הוא תוכנן על ידי IBM בשנת 1999 עבור מצבים שבהם נדרשת טביעת רגל קטנה של קוד ו/או רוחב הפס של הרשת מוגבל.

- MQTT פועל על מודל פרסום-הרשמה, מה שהופך אותו להרחבה. לקוחות מפרסמים הודעות לנושא, ולקוחות אחרים נרשמים לנושאים כדי לקבל הודעות.

- ליבה של MQTT היא הברוקר, שרת שמרכז את קבלת ההודעות מהלקוחות ומפנה אותן למנויים המתאימים.

- הוא תומך בשלוש רמות של איכות שירות, (QoS) המבטיח מסירת הודעות בהתאם לצרכים של אפליקציה מסוימת.

- MQTT נמצא בשימוש נרחב בהנדסת חשמל עבור רשתות חיישנים, מערכות אוטומציה ויישומי IoT בשל יעילותו ושימוש ברוחב הפס הנמוך, הוא שימושי במיוחד בתרחישים שבהם מכשירים צריכים לפעול על סוללה לתקופות ממושכות או שבהם תנאי הרשת גרועים.

- היכולת שלו לספק עדכונים בזמן אמת הופכת את MQTT לאידיאלי לניטור ובקרה של מערכות חשמל מבזרות, כגון רשתות חכמות או חיישנים מרוחקים.

בהקשר לפרויקט שלנו גם HTTP וגם MQTT ממלאים תפקידים קריטיים. השימוש הנרחב והסטנדרטיזציה של HTTP הופכים אותו לאידיאלי עבור ממשקי בקרה מבוססי אינטרנט וחילופי נתונים, בעוד שהיעילות והמדרג של MQTT מושלמות עבור יישומי IoT בזמן אמת ורשתות חיישנים. הבנה ומינוף של פרוטוקולים אלה היא המפתח לפרקטיקות הנדסת חשמל מודרניות, במיוחד בתחומי אוטומציה, ניטור מרחוק ו-IoT.

להלן סיבות מרכזיות מדוע פרויקט כזה הוא חיוני:

ניטור סביבתי בזמן אמת:

צורך: מערכות ניטור מסורתיות לרוב חסרות יכולות בזמן אמת, ומספקות נתונים במרווחי זמן תקופתיים.

פתרון: הפרויקט המוצע משלב חיישנים עם מיקרו-בקרים ושירותי ענן כדי להציע ניטור בזמן אמת של טמפרטורה, מפלס מים, אור, מרחק ולחות. זה מבטיח שמשתמשים יכולים לגשת לנתונים העדכניים ביותר לקבלת החלטות בזמן.

נגישות נתונים וממשק משתמש:

צורך: למערכות ניטור רבות יש נגישות מוגבלת, הדורשת נוכחות פיזית לאחזור נתונים. פתרון: על ידי חיבור המערכת לשירותי ענן, משתמשים יכולים לגשת לנתונים מרחוק דרך יישומי אינטרנט או ממשקים ניידים. זה נותן מענה לצורך בגישה נוחה וידידותית לנתונים סביבתיים מכל מקום בעולם.

שילוב חיישנים מרובים:

צורך: מערכות קיימות מתמקדות לרוב בפרמטר בודד, ומגבילות את היקף הניטור הסביבתי. פתרון: הפרויקט משלב מגוון רחב של חיישנים, כולל טמפרטורה, מפלס מים, אור, מרחק ולחות. גישה מקיפה זו מאפשרת למשתמשים לאסוף נתונים רב-צדדיים, ומציעה תצוגה הוליסטית של הסביבה המנוטרת.

מערכות משובצות חסכוניות:

צורך: פיתוח פתרונות ניטור מותאמים אישית יכול להיות יקר עבור יישומים בקנה מידה קטן יותר. פתרון: השימוש ב-Arduino Uno וב-Raspberry Pi מספק מערכות משובצות חסכוניות אך חזקות. פלטפורמות אלו מציעות איזון בין סבירות לפונקציונליות, מה שהופך את הפרויקט לנגיש עבור מגוון רחב של יישומים.

שלט רחוק ואוטומציה:

צורך: פונקציונליות בקרה מוגבלת במערכות קיימות, במיוחד במקומות מרוחקים או בסביבות לא נגישות.

פתרון: הפרויקט מאפשר למשתמשים לא רק לנטר אלא גם לשלוט בסביבה מרחוק. לדוגמה, התאמת הגדרות הטמפרטורה או הפעלת מערכות השקיה המבוססות על נתונים בזמן אמת הופכים לאפשריים, מה שמגביר את הפוטנציאל לאוטומציה.

מדרגיות ויכולת הסתגלות:

צורך: מערכות ניטור רבות חסרות יכולת מדרגיות ומתקשות להסתגל לדרישות המשתנות. פתרון: ארכיטקטורה מבוססת ענן משפרת את המדרגיות, ומאפשרת למערכת להכיל חיישנים או תכונות נוספים בצורה חלקה. יכולת הסתגלות זו מבטיחה שמערכת הניטור יכולה להתפתח עם הצרכים המשתנים וההתקדמות הטכנולוגית.

אבטחת נתונים משופרת:

צורך: חששות לגבי אבטחת מידע במערכות ניטור מסורתיות. פתרון: אינטגרציה עם שירותי ענן בעלי מוניטין מבטיח אמצעי אבטחת נתונים חזקים, שמירה על מידע רגיש. פרטוקולי הצפנה ומנגנוני גישה מאובטחים תורמים לסביבת נתונים אמינה ומאובטחת.

לסיכום, הפרויקט מטפל בפערים במערכות ניטור סביבתיות קיימות על ידי מתן גישה לנתונים בזמן אמת, ממשק ידידותי למשתמש, מגוון מגוון של שילובי חיישנים, מערכות משובצות חסכוניות, יכולות שליטה מרחוק, מדרגיות ואבטחת מידע משופרת. התקדמות אלו תורמים יחד לפתרון יעיל יותר, מותאם ונגיש יותר עבור מגוון יישומים.

2. מטרת הפרויקט

מטרת הפרויקט היא לפתח מערכת בקרה ניטור ושליטה מתקדמת, המשתמשת בטכנולוגיות חדישות של חיישנים ותקשורת ענן. המערכת מיועדת לשפר את היכולת לניטור וניהול יעיל של מגוון רחב של פרמטרים ומקרים באמצעות אגזוז מידע ישיר לענן. כל חיישן מתקן ומעניק מידע רך ודינמי, מאפשר למערכת להגיב בזמן אמת לשינויים ולתנאים שונים.

השילוב של טכנולוגיות אלו מספק יתרון נוסף בכך שהנתונים נשלחים ונאכסנים בענן, מאפשרים ניהול מרחוק, ניתוח מתקדם של נתונים והגברת היכולת לקבל החלטות מבוססות על מידע מקיף. בנוסף, ההעלאה לענן משמשת ככלי חירום ושחזור מהיר במקרי פיקוח וניטור, כדי להבטיח יכולת פעולה רצינית גם במקרים חריגים.

הפרויקט יצטרף למהפכה הטכנולוגית בתחום, תוך שמציע פתרונות יעילים וחדשים לתחומים רבים, כגון ניהול אנרגיה, בטיחות תעשייתית, וניטור תנועה וסביבה. דרך חדשנית זו יכולה לשפר את היציבות, הביצועים והיכולת ההגנה במגוון תחומים, ולספק פתרונות חכמים ומתקדמים שמסייעים להפוך את האובדן למציאות מינימלית ואת הפעולה למרבית היכולת.

2.1. תנאי התכנון

- חיבור החיישנים ביחד במעגל חשמלי בתוך חממת הגידול למשל
- חיבור החיישנים למקרו בקר שמספק להם מתח ושולט עליהם
- המקרו בקר מחובר לראספרי פאי ושולח לו המידע מהחיישנים בזמן אמת
- דרך הראספרי פאי מעלים את הנתונים לענן

תיאור המערכת

מערכת ניטור זו יכולה לספק מעקב ובקרה על מגוון חיישנים דינמיים באמצעות טכנולוגיית IoT (אינטרנט של הדברים). המערכת מתבצעת באמצעות חיבור החיישנים למערכת ענן מתקדמת, שבאמצעותה ניתן לנטר את הנתונים, וקבלת דוחות פרונטליים ומתקדמים על המצב והתפקוד של המערכת והסביבה הניטורית.

רכיבי המערכת:

Arduino כמרכז ניטור:

חיישנים שונים לדוגמה (טמפרטורה, לחות, זיהוי מרחק, זיהוי מפלס המים, אור) מחוברים ללוח Arduino לאיסוף הנתונים על ידי קוד שמאפשר קלטת הנתונים מהחיישנים ושליטה על פעולתם

ראספרי פאי כמרכז תקשורת והעלאת נתונים לענן :

קוד בשפת פיתון הרצה על ראספרי פאי אשר מקבלת נתונים Arduino ומפעילה יכולת התקשורת נעשה באמצעות ממשק סריאלי דרך חיבור USB השילוב בין ראספרי פאי ו Arduino

תקשורת עם הענן :

ראספרי פאי מתקשר עם הענן דרך תקשורת אלחוטית ושימוש בכמה פרוטוקולים פופולריים להעלאת הנתונים לענן.

שירותי ענן לניהול וניטור:

שירות ענן מתקדם אשר מקבל ומאחסן את הנתונים ומספק ממשק משתמש לניהול ובקרה ואפשרות לתכנן ולקבוע פרמטרים ניטור, התראות , וקבלת דוחות , ושליטה על המערכת מרחוק כמו Things board.

מערכת התראות ופעולה אוטומטית ושליטה :

מערכת מתקדמת של התראות באמצעות הגדרות שמותאמת אישית או באופן עצמי על ידי המשתמש.

2.3. הסבר כללי על המערכת

מערכת ניטור השולטת בחיישנים היא פתרון מקיף שנועד לאסוף, לעבד ולנהל נתונים בזמן אמת מחיישנים שונים בסביבה נתונה. המערכת בנויה לניטור פרמטרים ספציפיים, כגון טמפרטורה, לחות, מרחק, אוויר, כמות מים, והיא כוללת את היכולת להעלות הנתונים הנרכשים מחיישנים אלו לענן ולשלוט במערכת באמצעות ממשק קל למשתמש.

מרכיבי המפתח של המערכת:

חיישנים:

חיישנים סביבתיים: חיישנים אלו נפרסים למדידה ולניטור תנאים בסביבה, כגון טמפרטורה, לחות, מרחק, אוויר וכמות המים.

יחידת בקרה ועיבוד:

מיקרו-בקר או יחידת עיבוד: אחראי על איסוף נתונים מהחיישנים, עיבודם וקבלת החלטות על סמך כללים מוגדרים מראש או תצורות משתמש.

תקשורת והעברת נתונים:

תקשורת פנימית: מאפשרת תקשורת בין החיישנים ליחידת הבקרה.
העברת נתונים למערכת מרכזית: המערכת עשויה להעביר נתונים למערכת ניהול מרכזית, בדרך כלל דרך האינטרנט, לאחסון, ניתוח ניטור ושליטה מרחוק.

מערכת ניהול מרכזית:

פלטפורמה או שרת מבוססי ענן: פועלת כמרכז מרכזי לאחסון, ניתוח וניהול נתונים.
ממשק משתמש: מספק ידידותי למשתמש למשתמשים לנטר, להגדיר ולשלוט במערכת מרחוק.
התראות והתראות: מייצר התראות והתראות בזמן אמת על סמך ספים מוגדרים מראש או פרמטרים שהוגדרו על ידי המשתמש.

היתרונות של מערכת ניטור מבוקרת חיישנים:

ניטור בזמן אמת: המערכת מספקת מידע עדכני על תנאי הסביבה. **אוטומציה והתראות מתקדמות:** מאפשר למשתמשים להגדיר התראות אוטומטיות על סמך נתוני חיישנים.

ניהול יעיל: מאפשר בקרה וניהול מרכזי של מספר רב של חיישנים ממקום אחד. **יעילות אנרגטית:** מספקת את היכולת לשלוט במכשירים או לנקוט בפעולות כדי לחסוך באנרגיה בהתבסס על נתוני חיישנים.

תקשורת מרחוק: מאפשר ניטור ושליטה מרחוק באמצעות ממשק משתמש או אפליקציה סלולרית. לסיכום, מערכת ניטור השולטת בחיישנים מציעה פתרון מתוחכם וחכם לניהול יעיל של הסביבה, תוך מינוף הבנה מדויקת של נתוני הסביבה כדי לקבל החלטות מושכלות ולהפוך תגובות לאוטומטיות.

2.4. תכנון החומרה

תכנון חומרה למערכת השולטת בחיישנים כרוך בבחירת רכיבים מתאימים, הגדרת הארכיטקטורה ויצירת מערכת חזקה לרכישת נתונים, עיבוד ובקרה. להלן מתווה כללי לתכנון מערכת כזו:

2.4.1. מרכיבי עיצוב החומרה

מיקרו-בקר/מעבד:

בחירת מיקרו-בקר או מעבד: נבחר מיקרו-בקר מתאים (Arduino, Raspberry Pi).

חיישנים:

נעשה זיהוי לסוגי החיישנים: נבחר חיישנים על סמך הפרמטרים שהיה צריך לנטר (טמפרטורה, לחות, אור, זיהוי מרחק, זיהוי כמות מים).

ווידוי תאימות: נבדק שהחיישנים תואמים למיקרו-בקר/מעבד ויכולים לתקשר ביעילות.

ממשק תקשורת:

בחירת פרוטוקול תקשורת: נבחר פרוטוקול לתקשורת בין המיקרו-בקר והחיישנים

תקשורת אלחוטית: משיקולי נוחות נבחר להשתמש בתקשורת אלחוטית (Wi-Fi) להעברת נתונים מרחוק.

ספק כוח:

לקבע דרישות הספק: חשב את דרישות ההספק עבור המיקרו-בקר, החיישנים וכל רכיב אחר. **נבחר ספק כוח:** נבחרה שיטת אספקת חשמל מתאימה (הוחלט להשתמש במתח AC, וסולארי להמשך פיתוח עתידי עם אפשרות גיבוי כמו סוללת גיבוי) בהתבסס על היישום והסביבה המיועדים של המערכת.

2.4.2. ממשק משתמש

לשלב ממשק משתמש: רכיבים לאינטראקציה עם המשתמש, כגון לחצנים, נוריות, ממשק משתמש לשליטה או ניטור מקומיים דרך מסך ה Arduino IDE ו Thony .

2.4.3. מארז והגנה

נבחר מארז מתאים: נבחר מארז שמגן על החומרה מפני גורמים סביבתיים (אבק, לחות) ומספק אוורור נאות כמו dry cabinet שהוא בעצם מארז שמגן מפני אבק ולחות.

שקול הגנה על הסביבה: נלקח בחשבון שהמערכת מתוכננת לעמוד בסביבת ההפעלה המיועדת לשם כך צריך לבחור במארז שיעמוד בתנאי הסביבה של המערכת.

2.4.4. מטריצה ובדיקות

נבנה מעגל על המטריצה : הרכיבים הורכבו על מטריצה לבדיקה ראשונית. **לבדוק את הפונקציונליות:** כדי לבדוק את המערכת וכדי לוודא שהחיישנים מתממשקים כהלכה, הנתונים נרכשים בצורה מדויקת ולוגיקת הבקרה פועלת כמתוכנן נעשו שורה של בדיקות ואימות לתוצאות.

2.4.5. עיצוב PCB

נוצר לוח מעגלים מודפסים (PCB): לתכנון עתידי להמשך הפיתוח נדרש הגדרה קבועה יותר, לכן יש צורך לעצב לוח מודפס מותאם אישית על סמך המעגל המורכב על המטריצה .

2.5. תכנון התוכנה

תכנון התוכנה למערכת השולטת בחיישנים כרוך בפיתוח הקוד המאפשר למיקרו-בקר או למעבד ליצור אינטראקציה עם החיישנים, לעבד את הנתונים ולבצע פעולות בקרה.

2.5.1. עיצוב התוכנה של המערכת

ארכיטקטורת תוכנה:

קשוחה של מיקרו-בקר/מעבד:

נבחר שפת תכנות: נבחרה שפת תכנות מתאימה עבור המיקרו-בקר נבחרה השפה C ו Python עבור ה Raspberry Pi).

פתח קשוחה: נכתב קוד קשוחה כדי לאתחל את המיקרו-בקר, להגדיר ממשקי תקשורת ולהגדיר את לולאת התוכנית הראשית.

שילוב חיישנים:

בוצע מנהלי התקנים של חיישנים: נפתח דרייברים או משתמשים בספריות קיימות כדי להתממשק עם כל סוג של חיישן.

קריאת נתוני חיישנים: נכתב קוד לקריאת נתונים מהחיישנים באמצעות פרוטוקול התקשורת הנבחר (למשל, MQTT ו HTTP).

2.5.2. ממשק משתמש

שליטה דרך ממשק משתמש:

MQTT (Message Queuing Telemetry Transport):

הוא פרוטוקול קל משקל להעברת הודעות בשימוש נרחב באינטרנט של הדברים (IoT) לתקשורת בין מכשיר למכשיר. כשמדובר בשליטה בפיני GPIO (General Purpose Input/Output) במכשיר כמו Raspberry Pi או Arduino באמצעות MQTT, הרעיון הוא לשלוח פקודות דרך רשת למכשירים אלה כדי לשלוט ברכיבים פיזיים (כמו מנורות LED וחיישנים) מחוברים לפיני ה-GPIO שלהם.

שימוש ב-MQTT לשליטה ב-GPIO:

שליטה על LED המחובר לפיני GPIO ב-Raspberry Pi באמצעות MQTT. כאשר ה-Raspberry Pi מקבל הודעה, השתמש בספריית GPIO כדי לשנות את מצב פינת ה-GPIO (הפעל או כבה את הנורית).

HTTP:

קריאת נתונים מיציאה סריאלית ולאחר מכן שליחתם לפלטפורמה כמו ThingsBoard כרוכה במספר שלבים.

ThingsBoard:

היא פלטפורמת IoT בקוד פתוח המספקת ניהול מכשירים, איסוף נתונים, עיבוד והדמיה עבור פרויקטי IoT.

חומרה:

מכשיר עם יציאה טורית (כמו מיקרו-בקר וחיישן) המחובר למחשב.

תוכנה:

סביבת תכנות Python.

ספרייה לתקשורת טורית (כמו pyserial ב-Python).

ספריית לקוח HTTP לשליחת נתונים ל-ThingsBoard (כמו בקשות ב-Python).

2.5.3. בדיקה של המערכת

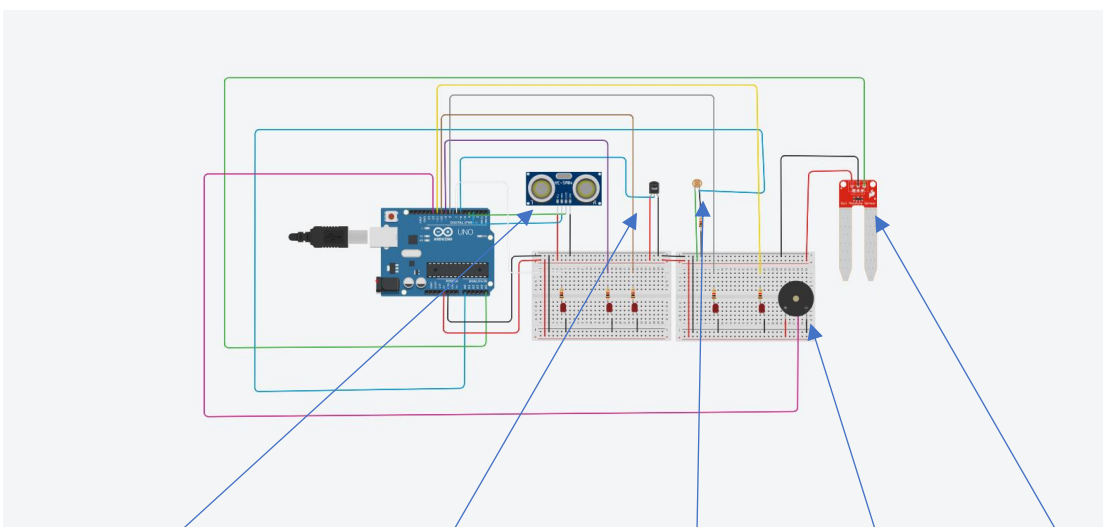
בדיקת יחידות: נערך בדיקות יחידות עבור רכיבים בודדים, כולל חיישנים, מפעילים ומודולי תקשורת.

בדיקת אינטגרציה: לבדוק את המערכת המשולבת כדי להבטיח שכל הרכיבים עובדים יחד בצורה חלקה.

2.5.4. תיעוד

תיעוד קוד: הערה יסודית בקוד לקריאה והבנה טובה יותר. והפרטים הספציפיים של עיצוב התוכנה יהיו תלויים בסוג המיקרו-בקר או המעבד שבו נעשה שימוש, החיישנים המשולבים, והדרישות הכוללות של המערכת הנשלטות על ידי חיישנים. שקול תמיד גורמים כמו עיבוד בזמן אמת, יעילות ומדרגיות במהלך תהליך עיצוב התוכנה.

3. מימוש המערכת



איור 1: המעגל החשמלי של המערכת

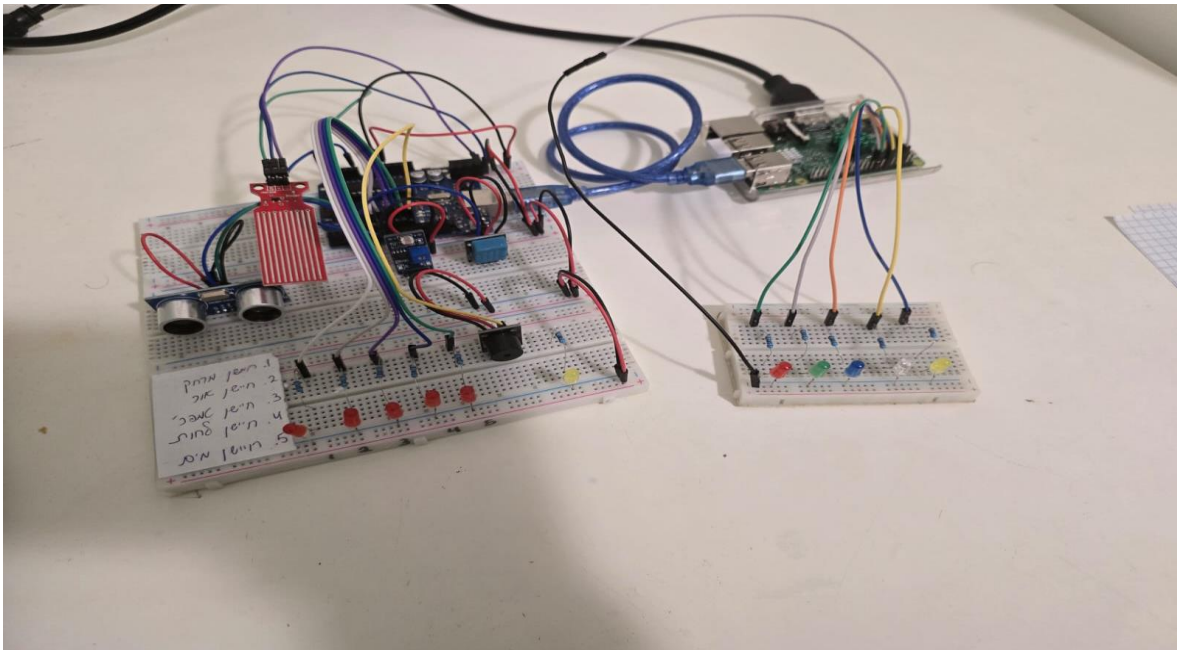
חיישן מרחק

חיישן טמפרטורה
ולוחות

חיישן אור

מצפצף

חיישן מפלס המים



איור 2: המעגל החשמלי בנוי

בהרשאת המודל התלת ממדי נעשה חיבור לחיישנים על המטריצה ובאמצעות קבלים לארדוינו שמספק להם מתח ושולט עליהם דרך קטע קוד בשפת C ומחברים הארדוינו לראספברי פאי שמקבל את המדע מארדוינו על כל חיישן בזמן אמת ומעלה את זה לענן

בסיס:

מטריצת חיבורים משתמשים כדי לחבר את הרכיבים על המטריצה ונחבר בניהם באמצעות קבלים ליצר את המעגל שתכננו כי לא נוח לחבר את הרכיבים ישירות לארדוינו

חיישנים:

- חיישן אור: חיישן אור הוא חיישן שבודק מצב האור בחממה מהכי בהיר להכי כהה
- חיישן לוחות : חיישן שבודק אחוז הלוחות בחממה
- חיישן טמפרטורה : חיישן שבודק את הטמפרטורה בחממה
- חיישן מרחק: חיישן שבודק את המרחק מהרצפה לגג קדי לדעת מה האורך של השתילים
- חיישן מפלס מים : חיישן שבודק את רמת המים

מיקרו בקר (Arduino) :

ארדואינו היא פלטפורמת קוד פתוח המשמשת לבניית פרויקטים אלקטרוניים הארדואינו מורכב ממיקרו בקר פיזי לתכנות והן מתוכנה המשתמשת לכתיבה והעלאת קוד ללוח הפיזי מטרתו היא לבקר על תהליכים ולווסט רגלי כניסות ויצאות ברכיב אלקטרוני בודד במקרה שלנו חיישנים הלוחות מצוידים ברגלי קלט\פלט דיגיטלית ואנאלוגית חיבור USB ומייצב מתח

ראספרי פאי (Raspberry Pi) :

הנו מחשב קטן, אך אינו סמארטפון או תחנת עבודה שנועדה לשימוש נייד. מדובר בכרטיס מחשב שגודלו בגודל של כרטיס אשראי: אחדות בודדות של סנטימטרים לכל כיוון. הכרטיס הזה יכול להתחבר למקלדת ולמסך, ובכך להיות מחשב לכל דבר ועניין. מדובר במחשב – אשר בדומה לאייפד ולסמארטפון משתייך לקבוצה מיוחדת אשר רק אפליקציות מסוימות יתאימו לו

שירותי ענן Thingsboards :

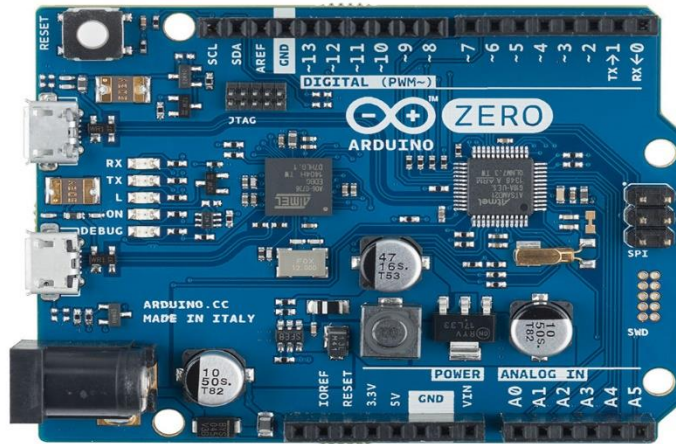
ThingsBoard היא פלטפורמת IoT (Internet of Things) בקוד פתוח המאפשרת ניהול, ניטור ובקרה של מכשירים מחוברים. הוא נועד להקל על פיתוח יישומי ופתרונות IoT. להלן מדריך כללי כיצד ניתן להשתמש ב-ThingsBoard לשליטה ובקרה על מערכת הכוללת חיישנים: התחל בהתקנת ThingsBoard בשרת או במופע הענן שלך.

הגדר את ThingsBoard על ידי הגדרת משתמשים, התקנים ופרמטרים נחוצים אחרים רשום את החיישנים שלך באמצעות ThingsBoard. זה כולל יצירת ישויות עבור כל חיישן בפלטפורמה.

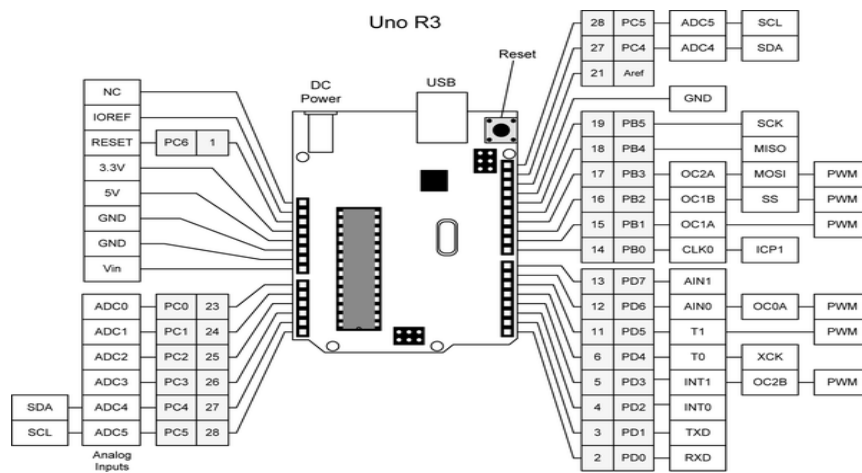
ThingsBoard תומך בפרוטוקולי קישוריות מכשירים שונים כגון HTTP, CoAP, MQTT ואחרים. ThingsBoard מספק REST API ופרוטוקולי MQTT/CoAP עבור מכשירים לפרסום נתוני טלמטריה.

עוצבו לוחות מחוונים ב-ThingsBoard כדי להמחיש ולנטר את נתוני הטלמטריה הנכנסים. יצרנו ווידג'טים ותרשימים כדי לייצג את נתוני החיישן בצורה ידידותית למשתמש. שולב בין ThingsBoard עם מערכת הבקרה כדי לאפשר תקשורת דו-כיוונית. זה מאפשר למערכת הבקרה לשלוח פקודות בחזרה לחיישנים או להתקנים. נעשה שימוש בתכונת CONTROL PANEL ב-ThingsBoard כדי לבצע פקודות במכשירים מחוברים.

3.1. מימוש החומרה



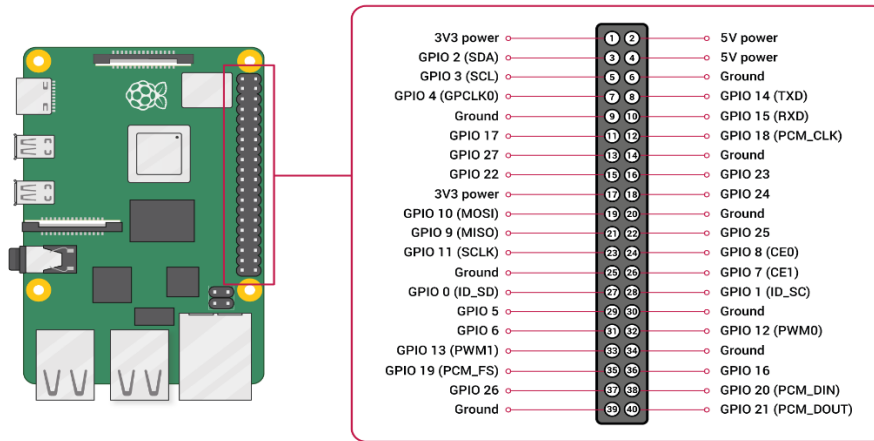
איור 3: ארדוֹינוֹ



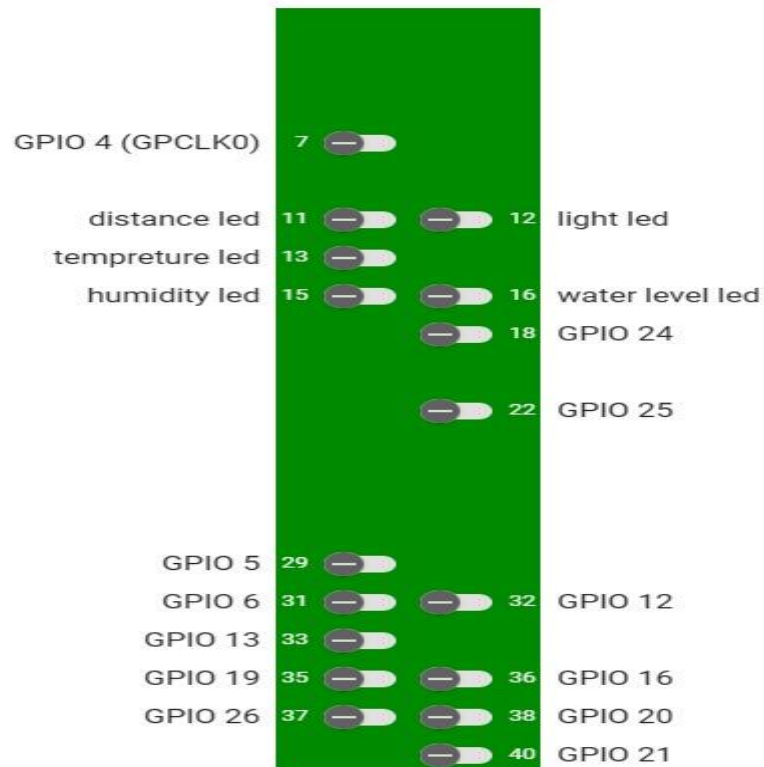
איור 4: סכימה חשמלית של ארדוֹינוֹ

קטגוריית הרגל	שם הרגל	מספר רגל על פי איור	פירוט
Power	Vin,3.3v,5v,GND	4,29,27,17,30	Vin: כניסת מתח לארדואינו בשימוש מקור מתח חיצוני של 6-12v 5v : כניסת מתח מבוקר לשימוש המיקרו מעבד ורכיבים אחרים 3.3V : מתח מוגבל לא נעשה בו שימוש (50mA) GND : רגלי אדמה
Reset	Reset	3	איפוס מעבד
Analog pins	A0-A7	19-26	נועד למדוד או לספק מתח אנלוגי בתחום 0-5V רגל 21 מחוברת לחיישן זיהוי המים ורגל 26 מחוברת לחיישן האור
Digital pins – input/output pins	D2-D13	5-16	משמש כרגלי כניסה או רגלי מוצא כאשר 0v מתח נמוך ו 5v מתח גבוה רגלים מ 7-12 מחוברות לילידים שנדלקים להתראה רגל 2 ו 3 מחוברות לחיישן המרחק ורגל 5 מחוברת לחיישן טמפרטורה ולחות
serial	Rx,Tx	1,2	מיועד לקבלה ושידור מסוג TTL
AREF	AREF	18	מתח יחוס למתח כניסה

טבלה 1: תיאור כניסות ויציאות של ארדואינו



איור 5: סכימה של ראספרי פאי

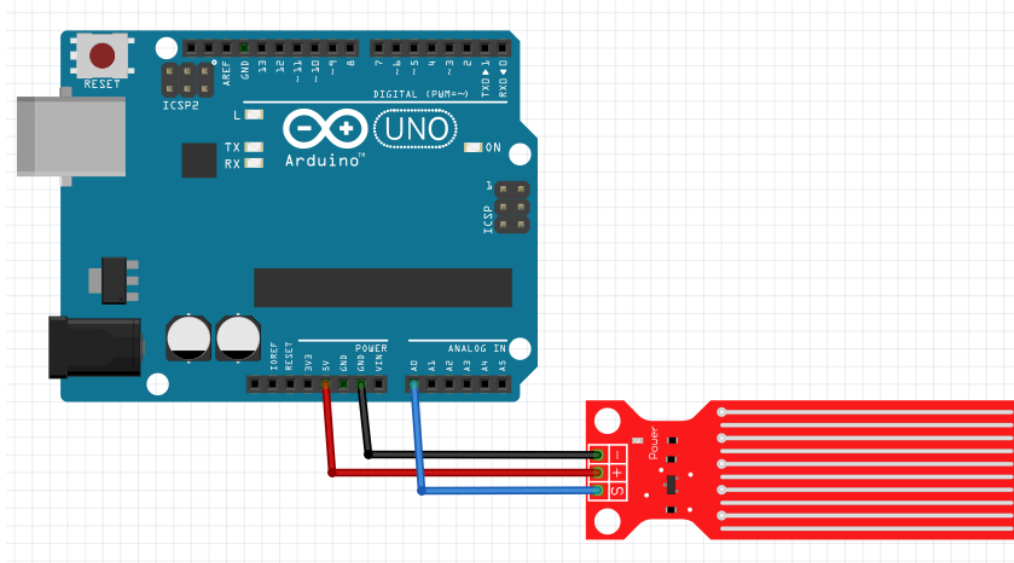


איור 6: ההדקים של GPIO ב Thingsboard

שם הרגל	מספר רגל על פי איור	פירוט
GND	6	אדמה
GPIO 17	11	קורה מתח או 0V או 5V דרך שליטה מ thingsboard ונדלק ליד שמטפל בחיישן המרחק
GPIO 18	12	קורה מתח או 0V או 5V דרך שליטה מ thingsboard ונדלק ליד שמטפל בחיישן האור
GPIO 27	13	קורה מתח או 0V או 5V דרך שליטה מ thingsboard ונדלק ליד שמטפל בחיישן הטמפרטורה
GPIO 22	15	קורה מתח או 0V או 5V דרך שליטה מ thingsboard ונדלק ליד שמטפל בחיישן הלחות
GPIO 23	16	קורה מתח או 0V או 5V דרך שליטה מ thingsboard ונדלק ליד שמטפל בחיישן מפלס המים

טבלה 2: תיאור החיבורים לראספרי פאי

חיישן זיהוי רמת המים :

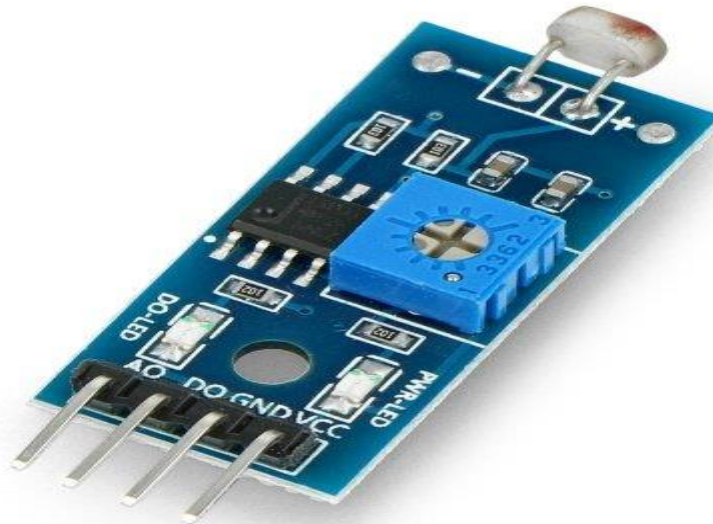


איור 7: חיישן גילוי מים

חיישן מפלס מים הוא מכשיר שנועד למדוד ולציין את מפלס המים במיכל או במיקום מסוים. הוא נמצא בשימוש נפוץ ביישומים שונים, כולל תהליכים תעשייתיים, חקלאות, ניטור סביבתי ואוטומציה ביתית. המטרה העיקרית של חיישן מפלס המים היא לספק מידע על מפלס המים, ולאפשר בקרה וניטור יעילים של מערכות הקשורות למים.

לשים לב שפיני IO אנלוגיים פועלים בצורה שונה מזו של פיני IO דיגיטליים. פינים אנלוגיים יספקו ערך מספרי (על טווח מספרי) המשויך לאות הנקרא בפין. טווח זה יכסה את כל המספרים השלמים מ-0 עד 1023. ה-0 המקביל לאות 0 וולט, ו-1023 המתאים לאות 5.0 וולט. לעומת זאת, הסיכה הדיגיטלית יכולה לספק רק שני ערכים 0 או 1 (כבוי או מופעל). זה מאפשר לך לנקוט בפעולה במקרה שאתה רוצה תרחישי הפעלה שונים

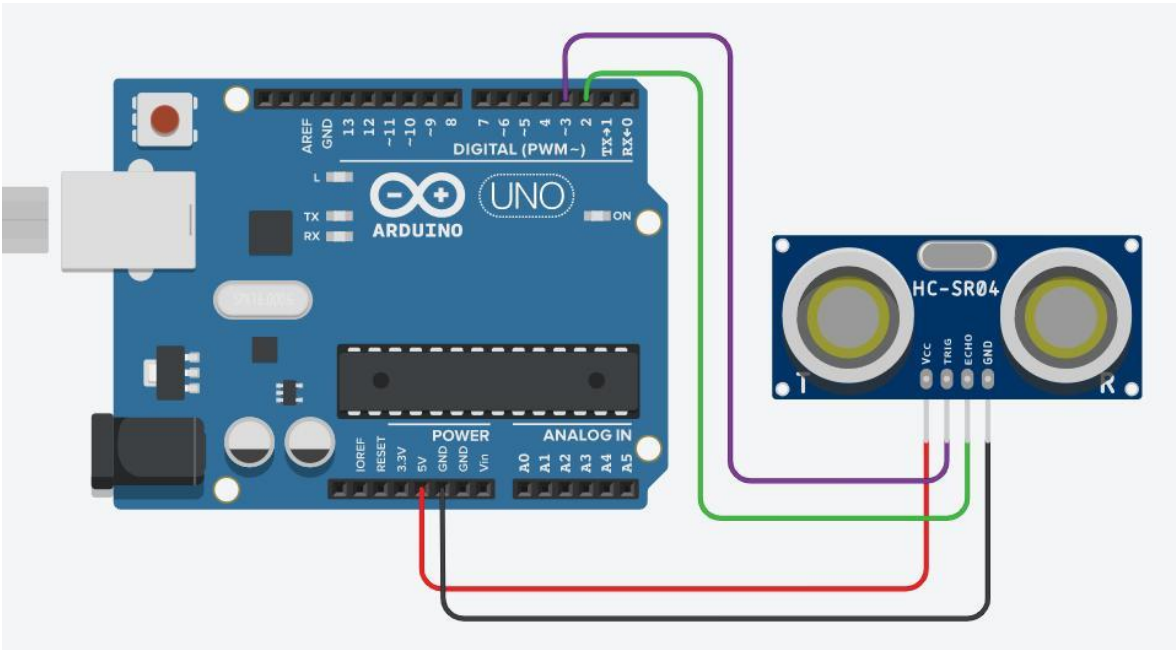
חיישן אור:



איור 8: חיישן אור

מודול חיישן האור הדיגיטלי/אנלוגי רגיש מאוד לאור הסביבה. משמש בדרך כלל לזיהוי בהירות ועוצמת האור הסביבתי. ניתן לכוון את הרגישות של זיהוי האור באמצעות פוטנציומטר. LDR (נגד תלוי אור) פירושו נגד תלוי אור או נגד פוטו, זהו אלמנט פסיבי של מעגל עם נגד בעל התנגדות משתנה בהתאם לעוצמת האור. יש לו 4 פינים ו-2 נוריות LED המציגות את מצב הדיגיטלית אספקת חשמל ופלט. בניית המודול מבוססת על מעגל LM393. החיישן מופעל מ-3.3 וולט עד 5 וולט.

חיישן זיהוי המרחק:



איור 9: חיישן המרחק

חיישנים אולטראסוניים מודדים מרחק על ידי שליחת וקליטה של הגל האולטראסוני. לחיישן האולטראסונד יש שולח כדי לפלוט את הגלים האולטראסוניים ומקלט לקליטת הגלים האולטראסוניים. הגל הקולי המועבר עובר באוויר ומוחזר על ידי פגיעה באובייקט. מחשב Arduino מחשב את הזמן שלוקח גל הדופק האולטראסוני להגיע למקלט מהשולח.

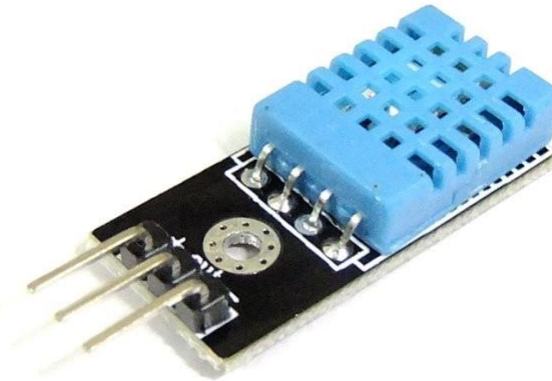
אנו יודעים שמהירות הקול באוויר היא כמעט 344 מטר לשנייה, אז, הפרמטרים הידועים הם זמן ומהירות (קבוע). באמצעות פרמטרים אלה, נוכל לחשב את המרחק שעבר גל הקול.

$$\text{נוסחה: מרחק} = \text{מהירות} * \text{זמן}$$

בקוד, המשתנה "משך" מאחסן את הזמן שלוקח גל הקול שעובר מהפולט למקלט. זה כפול מהזמן להגיע לאובייקט, בעוד שהחיישן מחזיר את הזמן הכולל כולל השולח לאובייקט וחפץ למקלט. לאחר מכן, הזמן שלוקח להגיע לאובייקט הוא מחצית מהזמן שלוקח להגיע למקלט.

$$\text{מרחק} = \text{מהירות הקול באוויר} * \frac{\text{זמן נלקח}}{2}$$

חיישן טמפרטורה ולחות:



איור 10: חיישן טמפרטורה ולחות

ה-DHT11 הוא חיישן טמפרטורה ולחות דיגיטלי בסיסי ונפוץ. הוא נמצא בשימוש נרחב בפרויקטים ויישומים שונים בשל הפשטות, העלות הנמוכה וקלות השימוש שלו. להלן כמה מאפיינים ומאפיינים עיקריים של חיישן DHT11:

תכונות עיקריות:

מדידת טמפרטורה:

ה-DHT11 יכול למדוד טמפרטורה בטווח של 0 עד 50 מעלות צלזיוס (32 עד 122 מעלות פרנהייט) עם דיוק של ± 2 מעלות צלזיוס.

מדידת לחות:

זה יכול למדוד לחות בטווח של 20% עד 80% עם דיוק של $\pm 5\%$.

יציאה דיגיטלית:

החיישן מספק פלט אות דיגיטלי, מה שמקל על התממשק עם מיקרו-בקרים ומעגלים דיגיטליים.
תקשורת חוטית אחת:

תקשורת עם DHT11 מושגת באמצעות ממשק דיגיטלי חוטי יחיד, המפשט את החיווט.
צריכת חשמל נמוכה:

ה-DHT11 פועל בהספק נמוך, מה שהופך אותו מתאים ליישומים המופעלים על ידי סוללה.
תנור חימום מובנה:

החיישן כולל תנור חימום מובנה המאפשר לו להתאושש מעיבוי או ערפל.
ל-DHT11 יש בדרך כלל שלושה פינים:

VCC (כוח): מתחבר לאספקת החשמל (בדרך כלל V3.3 או V5).

נתונים: מתחבר לפין דיגיטלי במיקרו-בקר לתקשורת נתונים.

GND (הארקה): מתחבר לאדמה של ספק הכוח.

התממשקות עם מיקרו-בקר:

התממשקות ה-DHT11 עם מיקרו-בקר כרוכה בקריאת האות הדיגיטלי מסיכת הנתונים של החיישן.
פרוטוקול התקשורת פשוט יחסית, וזמינות ספריות שונות עבור פלטפורמות מיקרו-בקר פופולריות
כמו Raspberry Pi ו-Arduino.

3.2. מימוש התוכנה

נעשה שימוש בשפת Python ב-Raspberry Pi. כדי לשאוב מידע מה Arduino וכדי להעלות לענן מכיוון ששפת התכנות Python מצטיינת בפופולריות רבה בתחום תכנון מערכות בקרה וניטור, וכן בהעלאת נתונים לשירותי ענן, קוד פתוח והמגוון הרב של ספריות וכלים מסייעים למתכנתים לפתח מערכות יעילות, גמישות ויעילות. השפה מספקת גם תמיכה מובנית בפרוטוקולים וסטנדרטים נפוצים, מה שהופך אותה לבחירה ידידותית ומתקדמת למגוון יישומים בתחום התקשורת והשליטה.

```
#!/usr/bin/env python3
import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
import json
import serial
import time
import requests
import threading

# MQTT setup
THINGSBOARD_HOST = 'thingsboard.cloud'
ACCESS_TOKEN = 'MChdaEdIOWeGZSxmiFS3'

# Serial setup
SERIAL_PORT = '/dev/ttyACM0'
SERIAL_BAUDRATE = 9600
thingsboard_url = 'https://thingsboard.cloud/api/v1/MChdaEdIOWeGZSxmiFS3/telemetry'
ser = serial.Serial(SERIAL_PORT, SERIAL_BAUDRATE, timeout=1.0)

# GPIOs
gpio_state = {7: False, 11: False, 12: False, 13: False, 15: False, 16: False, 18: False, 22: False, 29: False,
              31: False, 32: False, 33: False, 35: False, 36: False, 37: False, 38: False, 40: False}

# MQTT Callbacks
def on_connect(client, userdata, flags, rc):
    print('Connected with result code ' + str(rc))
    client.subscribe('v1/devices/me/rpc/request+')

def on_message(client, userdata, msg):
    print('Topic: ' + msg.topic + '\nMessage: ' + str(msg.payload))
    data = json.loads(msg.payload)
```

```

if data['method'] == 'getGpioStatus':
    client.publish(msg.topic.replace('request', 'response'), get_gpio_status(), 1)
elif data['method'] == 'setGpioStatus':
    set_gpio_status(data['params']['pin'], data['params']['enabled'])
    client.publish(msg.topic.replace('request', 'response'), get_gpio_status(), 1)
    client.publish('v1/devices/me/attributes', get_gpio_status(), 1)

# GPIO functions
def get_gpio_status():
    return json.dumps(gpio_state)

def set_gpio_status(pin, status):
    GPIO.output(pin, GPIO.HIGH if status else GPIO.LOW)
    gpio_state[pin] = status

# Serial reading and uploading to Thingsboard
def read_serial_data():
    while True:
        data = ser.readline().decode().strip()
        if data:
            try:
                sensor_values = [float(value) for value in data.split('x')]
                if len(sensor_values) >= 5:
                    upload_to_thingsboard(*sensor_values[:5])
            except ValueError:
                print("Insufficient values received")
        except ValueError:
            print(f"Invalid data received: {data}")

def upload_to_thingsboard(distance, humidity, temperature, light, waterlevel):
    payload = {
        "ts": int(time.time() * 1000),
        "values": {
            "distance": distance,
            "humidity": humidity,
            "temperature": temperature,
            "light": light,
            "waterlevel": waterlevel
        }
    }

```

```

response = requests.post(thingsboard_url, json=payload)
if response.status_code == 200:
    print("Data uploaded to Thingsboard successfully.")
else:
    print(f"Failed to upload data to Thingsboard. Status code: {response.status_code}, response:
{response.text}")

# Main setup
if _name_ == '_main_':
    # GPIO setup
    GPIO.setmode(GPIO.BOARD)
    for pin in gpio_state:
        GPIO.setup(pin, GPIO.OUT)

    # MQTT client setup
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.username_pw_set(ACCESS_TOKEN)
    client.connect(THINGSBOARD_HOST, 1883, 60)

    # Starting MQTT client in a separate thread
    threading.Thread(target=client.loop_forever).start()

    try:
        read_serial_data()
    except KeyboardInterrupt:
        ser.close()
        GPIO.cleanup()
        print("Serial and GPIO cleaned up.")

```

מבוא והסבר:

סקריפט Python זה נועד לקרוא נתוני חיישנים מציאה טורית, לעבד את הנתונים ולהעלות אותם לפלטפורמת Thingsboard. Thingsboard היא פלטפורמת IoT בקוד פתוח המאפשרת למשתמשים לאסוף, להמחיש ולנתח נתונים ממכשירי האינטרנט של הדברים (IoT). נפרט את מרכיבי המפתח והפונקציונליות של הקוד:

```
#!/usr/bin/env python3
```

שורה זו מציינת את האינטרפרטר שיש להשתמש בו כאשר הסקריפט יופעל, במקרה זה, זה מציין שיש להפעיל את הסקריפט באמצעות האינטרפרטר של Python 3.

ייבוא ספריות:

```
import paho.mqtt.client as mqtt
```

```
import RPi.GPIO as GPIO
```

```
import json
```

```
import serial
```

```
import time
```

```
import requests
```

```
import threading
```

שורות אלה מייבאות ספריות Python הדרושים:

serial: לתקשורת טורית, משמש לתקשורת עם התקנים המחוברים ליציאה הטורית.

time: לעבודה עם חותמות זמן ופונקציות הקשורות לזמן.

requests: לביצוע בקשות HTTP, שימשו לשליחת נתונים לפלטפורמת Thingsboard.

json: לטיפול בנתוני JSON, מכיוון שה-Thingsboard API מצפה לנתונים בפורמט JSON.

paho.mqtt.client: ספרייה זו משמשת לחיבור ותקשורת עם מתווך MQTT.

RPi.GPIO: מספק גישה לסיכות GPIO ב-Raspberry Pi לשליטה באלקטרוניקה.

threading: מאפשר הפעלת חלקים שונים של הקוד בו-זמנית, כמו טיפול בהודעות MQTT בזמן

קריאת נתוני חיישן.

הגדרת HTTP ו MQTT:

```
# MQTT setup
THINGSBOARD_HOST = 'thingsboard.cloud'
ACCESS_TOKEN = 'MChdaEdlOWeGZSxmiFS3'

# Serial setup
SERIAL_PORT = '/dev/ttyACM0'
SERIAL_BAUDRATE = 9600
thingsboard_url = 'https://thingsboard.cloud/api/v1/MChdaEdlOWeGZSxmiFS3/telemetry'
ser = serial.Serial(SERIAL_PORT, SERIAL_BAUDRATE, timeout=1.0)
```

THINGSBOARD_HOST, ACCESS_TOKEN: אלו הן תצורות לחיבור ל Thingsboard MQTT.

SERIAL_PORT, SERIAL_BAUDRATE: הגדרות לתקשורת טורית.
SERIAL_BAUDRATE = 9600 ההצהרה מגדירה את קצב ההחזרה ל-9600 באוד. זוהי מהירות נפוצה עבור מכשירים טוריים רבים ונחשבת לקצב העברת נתונים איטי יחסית אך יציב ואמין.

SERIAL_PORT = '/dev/ttyACM0' השורה מגדירה את היציאה הספציפית המשמשת לתקשורת טורית. במערכות מבוססות לינוקס כמו מערכת ההפעלה של Raspberry Pi, יציאות טוריות נקראות בדרך כלל /dev/ttyXXXX, כאשר XXXX יכול להשתנות בהתאם לחומרה ולאופן שבו מערכת ההפעלה מזהה אותה.

thingsboard_url: כתובת אתר לשליחת נתונים ל Thingsboard.
 ser: מאתחל את החיבור הטורי עם ההגדרות שצוינו.

:GPIO

```
gpio_state = {7: False, 11: False, 12: False, 13: False, 15: False, 16: False, 18: False, 22: False,
29: False,
31: False, 32: False, 33: False, 35: False, 36: False, 37: False, 38: False, 40: False}
```

מעקב אחר המצבים (True/False) של פני GPIO שונים.

פונקציות התקשרות חוזרת של MQTT:

```
# MQTT Callbacks
def on_connect(client, userdata, flags, rc):
    print('Connected with result code ' + str(rc))
    client.subscribe('v1/devices/me/rpc/request/+')

def on_message(client, userdata, msg):
    print('Topic: ' + msg.topic + '\nMessage: ' + str(msg.payload))
    data = json.loads(msg.payload)
    if data['method'] == 'getGpioStatus':
        client.publish(msg.topic.replace('request', 'response'), get_gpio_status(), 1)
    elif data['method'] == 'setGpioStatus':
        set_gpio_status(data['params']['pin'], data['params']['enabled'])
        client.publish(msg.topic.replace('request', 'response'), get_gpio_status(), 1)
        client.publish('v1/devices/me/attributes', get_gpio_status(), 1)
```

on_connect: מופעל כאשר לקוח MQTT מתחבר למתווך. הוא נרשם לנושא לקבלת בקשות להתקשרות מרחוק (RPC).

on_message: מטפל בהודעות MQTT נכנסות. זה יכול לאחזר או לשנות את מצב ה-GPIO ולתקשר עם שרת Thingsboard.

פונקציות GPIO

```
# GPIO functions
def get_gpio_status():
    return json.dumps(gpio_state)

def set_gpio_status(pin, status):
    GPIO.output(pin, GPIO.HIGH if status else GPIO.LOW)
    gpio_state[pin] = status
```

`get_gpio_status`: מחזירה את המצב הנוכחי של כל פני GPIO בפורמט JSON.

`set_gpio_status`: מעדכן את המצב של פין GPIO שצוין ומשקף את השינוי במילון `gpio_state`.

קריאה סדרתית והעלאת נתונים

```
# Serial reading and uploading to Thingsboard
def read_serial_data():
    while True:
        data = ser.readline().decode().strip()
        if data:
            try:
                sensor_values = [float(value) for value in data.split('x')]
                if len(sensor_values) >= 5:
                    upload_to_thingsboard(*sensor_values[:5])
            except ValueError:
                print("Insufficient values received")
            print(f"Invalid data received: {data}")

def upload_to_thingsboard(distance, humidity, temperature, light, waterlevel):
    payload = {
        "ts": int(time.time() * 1000),
        "values": {
            "distance": distance,
            "humidity": humidity,
            "temperature": temperature,
            "light": light,
            "waterlevel": waterlevel
```

```

    }
}

response = requests.post(thingsboard_url, json=payload)
if response.status_code == 200:
    print("Data uploaded to Thingsboard successfully.")
else:
    print(f"Failed to upload data to Thingsboard. Status code: {response.status_code}, response: {response.text}")

```

`read_serial_data`: קורא נתונים ברציפות מהיציאה הטורית, מנתח אותם, ואם חוקי, מעלה אותם ל-Thingsboard.

`upload_to_thingsboard`: שולח נתוני חיישן ל-Thingsboard באמצעות בקשת HTTP POST.

הגדרה ראשית

```

# Main setup
if __name__ == '__main__':
    # GPIO setup
    GPIO.setmode(GPIO.BOARD)
    for pin in gpio_state:
        GPIO.setup(pin, GPIO.OUT)

    # MQTT client setup
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.username_pw_set(ACCESS_TOKEN)
    client.connect(THINGSBOARD_HOST, 1883, 60)

    # Starting MQTT client in a separate thread
    threading.Thread(target=client.loop_forever).start()

try:
    read_serial_data()
except KeyboardInterrupt:
    ser.close()
    GPIO.cleanup()
    print("Serial and GPIO cleaned up.")

```

1. ``__if __name__ == '__main__':``

שורה זו בודקת אם הסקריפט מופעל כתוכנית הראשית. זהו ניב פייתון סטנדרטי כדי להבטיח שקוד מסוים ירוץ רק כאשר הסקריפט מבוצע ישירות (ולא כשהוא מיובא כמודול בסקריפט אחר).

2. ``GPIO.setmode(GPIO.BOARD)``

מגדיר את מערכת מספור הפינים. ``GPIO.BOARD`` אומר שאתה מתייחס לסיכות לפי מיקומן הפיזי על הלוח.

3. ``gpio_state: GPIO.setup(pin, GPIO.OUT)`` - עבור PIN

לולאה זו חוזרת דרך כל פין GPIO המוגדר במילון ``gpio_state`` ומגדירה אותם כסיכות פלט. זה הכרחי לשליטה במכשירים כמו נוריות LED או מנועים המחוברים לפינים אלה.

4. ``client = mqtt.Client`()`

יוצר מופע לקוח MQTT חדש.

5. ``client.on_connect = on_connect``

מגדיר את פונקציית ההתקשרות `'on_connect'`, שתיקרא כאשר לקוח MQTT יתחבר למתווך.

6. ``client.on_message = on_message``

מגדיר את פונקציית ההתקשרות `'on_message'` לטיפול בהודעות MQTT נכנסות.

7. ``client.username_pw_set(ACCESS_TOKEN)``

מגדיר את הלקוח עם אסימון גישה לאימות עם מתווך MQTT.

8. ``client.connect(THINGSBOARD_HOST, 1883, 60)``

מחבר את לקוח MQTT למארח שצוין ('THINGSBOARD_HOST') ביציאה 1883 (יציאת MQTT רגילה) עם מרווח של 60 שניות Keepalive.

9. `threading.Thread(target=client.loop_forever).start``

מפעיל את לולאת הלקוח MQTT בשרשור נפרד. זה מאפשר ללקוח לעקוב באופן רציף אחר הודעות ולשמור על החיבור בחיים מבלי לחסום את השרשור הראשי.

10. `try: read_serial_data``

מתחיל לקרוא נתונים מהיציאה הטורית בתוך בלוק 'נסה'. זוהי הפעולה העיקרית שבה התסריט קורא נתוני חיישן.

11. `Keyboard Interrupt``

בלוק 'למעט' זה תופס חריג 'הפסקת מקשים' (נוצר בדרך כלל על ידי לחיצה על Ctrl+C). הוא משמש כאן לכיבוי בחן של התסריט.

12. `ser.close``

סוגר את החיבור הטורי.

13. `GPIO.cleanup``

מנקה את ה-GPIO על-ידי איפוס המצב של פני ה-GPIO המשמשים את הסקריפט.

14. `print`("סדרתי ו-GPIO נוקו.")``

מדפיס הודעה המציינת שהניקוי הצליח.

החלטנו להשתמש בשפת התכנות C ב Arduino כדי לשאוב מידע מהחיישנים מכיוון שפת התכנות C הופכת לבחירה מועדפת בקרב מתכנתי מערכות בקרה וניטור כמו Arduino Uno. היא מציינת בידודיותה למתכנתים, ביכולת לספק ביצועים מהירים. C מספקת פתרונות יעילים ויכולת לנהל משאבים בצורה יעילה, ובכך הופכת לכלי אידיאלי לפיתוח מערכות בקרה יציבות ויעילות.

```
int WaterLevel = 0; // holds the value
int Waterpin = A5; // sensor pin used
const int trigPin = 3; // ultrasonic sensor pin used
const int echoPin = 4; // ultrasonic sensor pin used
long duration; // ultrasonic defines variables
int distance; // ultrasonic defines variables
#include "DHT.h" //library for temp and humidity sensor
DHT dht;
int buzzer=12;
void setup()
{
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT); // ultrasonic Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // ultrasonic Sets the echoPin as an Input
  dht.setup(6); // temp hum sensor :set pin for data communication
  pinMode(buzzer,OUTPUT);
  pinMode(7,OUTPUT); //diodes for sensors warning !!
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
}
```

```
void loop()
{
  // begin light sensor
  // reads the input on analog pin A0 (value between 0 and 1023)
  int analogValue = analogRead(A0);
  //Serial.print("Analog reading = ");
  //Serial.print(analogValue); // the raw analog reading

  delay(1000);
  // thresholds, qualitatively determined
  if (analogValue < 100)
  {
    // Serial.println(" - Very bright");
    noTone(buzzer);
    digitalWrite(8,LOW);
  }
  else if (analogValue < 200)
  {
    // Serial.println(" - Bright");
    noTone(buzzer);
    digitalWrite(8,LOW);
  }
}
```

```
else if (analogValue < 500)
{
  // Serial.println(" - Light");
  noTone(buzzer);
  digitalWrite(8,LOW);
}
else if (analogValue < 800)
{
  // Serial.println(" - Dim");
  noTone(buzzer);
  digitalWrite(8,LOW);
}

else
{
  //Serial.println(" - Dark");
  tone(buzzer,2000,1500);
  digitalWrite(8,HIGH);
}
delay(1000);

// end light sensor
```



```
// begin WaterLevel sensor
WaterLevel = analogRead(Waterpin); //Read data from analog pin and store it to
WaterLevel variable

if (WaterLevel < 100)
{
  // Serial.println("water not detected");
  noTone(buzzer);
  digitalWrite(11,LOW);
}
else
{
  // Serial.println("water detected");
  tone(buzzer,2000,2000);
  digitalWrite(11,HIGH);
}

// end water level sensor

delay(1000);

// begin ultrasonic sensor
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
```

```

digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance = duration * 0.034 / 2;
// Prints the distance on the Serial Monitor
//Serial.print("Distance: ");
//Serial.print(distance);
//Serial.println(" cm");

if (distance < 10)
{
  tone(buzzer,2000,1500);
  digitalWrite(7,HIGH);
}
else
{
  noTone(buzzer);
  digitalWrite(7,LOW);
}
//end ultrasonic sensor
delay(1000);

//begin temperature & humidity sensor
//Serial.println();
//Serial.println("Status\tHumidity (%)\tTemperature (C)\t(F)");
delay(dht.getMinimumSamplingPeriod()); /* Delay of amount equal to sampling
period */
float humidity = dht.getHumidity(); /* Get humidity value */
float temperature = dht.getTemperature(); /* Get temperature value */

```

```

//Serial.print(dht.getStatusString());    /* Print status of communication */
//Serial.print("\t");
//Serial.print(humidity, 1);
// Serial.print("\t\t");
// Serial.print(temperature, 1);
// Serial.print("\t\t");
//Serial.println(dht.toFahrenheit(temperature), 1); /* Convert temperature to
Fahrenheit units */

```

```

if (humidity > 70 || humidity <40 )

```

```

{
    tone(buzzer,2000,1500);
    digitalWrite(10,HIGH);
}

```

```

else

```

```

{
    noTone(buzzer);
    digitalWrite(10,LOW);
}

```

```

if (temperature > 30 || temperature < 10 )

```

```

{
    tone(buzzer,2000,1500);
    digitalWrite(9,HIGH);
}

```

```

else

```

```

{

```

```

    noTone(buzzer);
    digitalWrite(9,LOW);
  }
//end temperature & humidity sensor


//print values onto serial monitor x is used as a delimiter
//Serial.println();
//Serial.println("----- ");
Serial.print(distance);
Serial.print('x');
Serial.print(humidity);
Serial.print('x');
Serial.print(temperature);
Serial.print('x');
Serial.print(analogValue);
Serial.print('x');
Serial.print(WaterLevel);
//Serial.println();
//Serial.println("----- ");


    delay(2000);
  }

```

קוד Arduino זה מיועד למערכת הכוללת חיישנים שונים לניטור תנאי סביבה שונים. החיישנים שבהם נעשה שימוש הם:

חיישן אור (מחובר לפין אנלוגי A0)

חיישן מפלס מים (מחובר לפין אנלוגי A5)

חיישן קולי (מחובר לפינים דיגיטליים 3 ו-4)

חיישן טמפרטורה ולחות (חיישן DHT, מחובר לפין דיגיטלי 6)

המערכת כוללת גם זמזם להתראות שמע ומספר דיודות (LED) להתרעות חזותיות. הקוד קורא ברציפות נתונים מחיישנים אלו, מעבד אותם ומייצר התראות על סמך תנאים מסוימים.

```
int WaterLevel = 0; // משתנה לאחסון ערך רמת המים
```

```
int Waterpin = A5; // פין החיישן של רמת המים
```

```
const int trigPin = 3; // פין של חיישן אולטרסוניק
```

```
const int echoPin = 4; // פין של חיישן אולטרסוניק
```

```
long duration; // משתנה לאחסון זמן תגובת החיישן
```

```
int distance; // משתנה לאחסון מרחק
```

```
#include "DHT.h" // ספריה לחיישן טמפרטורה ולחות
```

```
DHT dht;
```

```
int buzzer = 12;
```

```

void setup() {
  Serial.begin(9600); // אתחול התקשורת הסריאלית בקצב 9600
  pinMode(trigPin, OUTPUT); // הגדרת פין של החיישן אולטרסוניק כפלט
  pinMode(echoPin, INPUT); // הגדרת פין של החיישן אולטרסוניק כקלט
  dht.setup(6); // הגדרת פין לתקשורת נתונים עבור חיישן הטמפרטורה והלחות
  pinMode(buzzer, OUTPUT); // הגדרת פין של הזמזם כפלט
  pinMode(7, OUTPUT); // הגדרת פני דיודות להתראות
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
}

```

הקטע הזה מגדיר משתנים, פונים ומגדיר את החיישנים השונים המחוברים למערכת. כל פין משמש לחיבור חיישן או מכשיר שונה.

```

void loop()
{
  // קריאת ערך אנלוגי מחיישן האור // begin light sensor
  int analogValue = analogRead(A0);
  delay(1000); // השהיית קריאות נתונים למניעת רעש
  if (analogValue < 100) {
    noTone(buzzer);
    digitalWrite(8, LOW);
  } else if (analogValue < 200) {
    noTone(buzzer);
    digitalWrite(8, LOW);
  } else if (analogValue < 500) {
    noTone(buzzer);
  }
}

```

```

    digitalWrite(8, LOW);
} else if (analogValue < 800) {
    noTone(buzzer);
    digitalWrite(8, LOW);
} else {
    tone(buzzer, 2000, 1500);
    digitalWrite(8, HIGH);
}
delay(1000);
// end light sensor

```

```

// begin WaterLevel sensor      // קריאת ערך אנלוגי מחיישן רמת המים
WaterLevel = analogRead(Waterpin);
if (WaterLevel < 100)
{
    // Serial.println("water not detected");
    noTone(buzzer);
    digitalWrite(11,LOW);
}
else
{
    // Serial.println("water detected");
    tone(buzzer,2000,2000);
    digitalWrite(11,HIGH);
}
// end water level sensor

delay(1000);

```

```

// begin ultrasonic sensor
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = duration * 0.034 / 2;
if (distance < 10) {
    tone(buzzer, 2000, 1500);
    digitalWrite(7, HIGH);
} else {
    noTone(buzzer);
    digitalWrite(7, LOW);
}
// end ultrasonic sensor

delay(1000);

// begin temperature & humidity sensor
delay(dht.getMinimumSamplingPeriod());
float humidity = dht.getHumidity();
float temperature = dht.getTemperature();
if (humidity > 70 || humidity < 40) {
    tone(buzzer, 2000, 1500);
    digitalWrite(10, HIGH);
}

```



```

else {
    noTone(buzzer);
    digitalWrite(10, LOW);
}
if (temperature > 30 || temperature < 10) {
    tone(buzzer, 2000, 1500);
    digitalWrite(9, HIGH);
} else {
    noTone(buzzer);
    digitalWrite(9, LOW);
}
// end temperature & humidity sensor
Serial.print(distance);
Serial.print('x');
Serial.print(humidity);
Serial.print('x');
Serial.print(temperature);
Serial.print('x');
Serial.print(analogValue);
Serial.print('x');
Serial.print(WaterLevel);

delay(2000);
}

```

החלק הזה הוא הלולאה הראשית של הקוד. בכל פעם, הקוד קורא נתונים מכל החיישנים והמכשירים השונים ומעבד אותם. כל חלק מתרגם את הנתונים למצב של התראה, והקוד מדפיס את התוצאה למוניטור הסריאלי. הערכים מופרדים באמצעות התו. 'x'

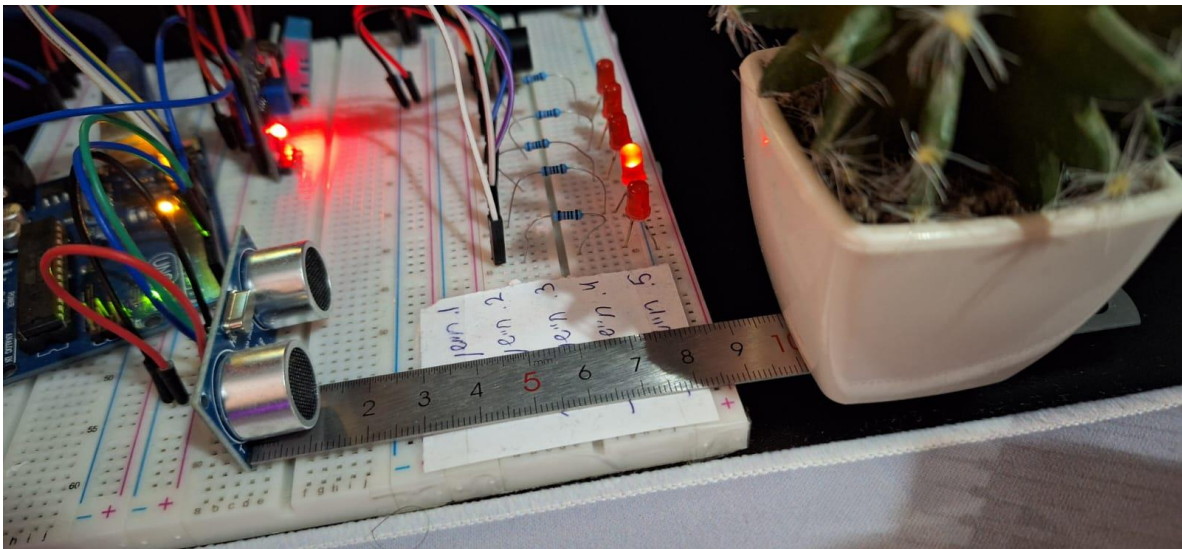
4. ניסויים ובדיקות

מטרת הניסויים והבדיקות היא להעריך את הפונקציונליות והביצועים של מערכת הבקרה והניטור של החיישנים המעלים את הנתונים לענן באמצעות raspberry pi ו-Arduino uno. הניסויים והבדיקות כוללים מדידה והשוואה של התוצאות המתקבלות מהמערכת, כגון קריאות החיישנים, קצב העברת הנתונים, צריכת החשמל וממשק המשתמש. הניסויים והבדיקות נערכים בתרחישים ובתנאים שונים, כגון בבית ובחוץ, ביום ובלילה, לחות גבוהה ונמוכה, במרחק קרוב ורחוק ומפלסי מים שונים. הניסויים והבדיקות נועדו לוודא שהמערכת פועלת כראוי ועומדת ביעדי הפרויקט.

sensors					
Timeseries table 🔍 ⬇️ ✕					
🕒 Realtime - last minute					
Timestamp ⬇️	distance	humidity	light	temperature	waterlevel
2024-01-12 08:58:33	12 m	71 g/cm³	924 lx	17 °C	0.0
2024-01-12 08:58:26	12 m	71 g/cm³	933 lx	17 °C	0.0
2024-01-12 08:58:19	12 m	71 g/cm³	891 lx	17 °C	0.0
2024-01-12 08:58:12	12 m	71 g/cm³	906 lx	17 °C	0.0
2024-01-12 08:58:05	12 m	71 g/cm³	906 lx	17 °C	0.0
2024-01-12 08:57:58	12 m	71 g/cm³	890 lx	17 °C	0.0
2024-01-12 08:57:51	12 m	71 g/cm³	895 lx	17 °C	0.0
2024-01-12 08:57:44	12 m	71 g/cm³	898 lx	17 °C	0.0
2024-01-12 08:57:37	12 m	71 g/cm³	908 lx	17 °C	0.0

טבלה 3: תוצאות מערכת בענן

חיישן המרחק:



איור 11: בדיקת חיישן המרחק

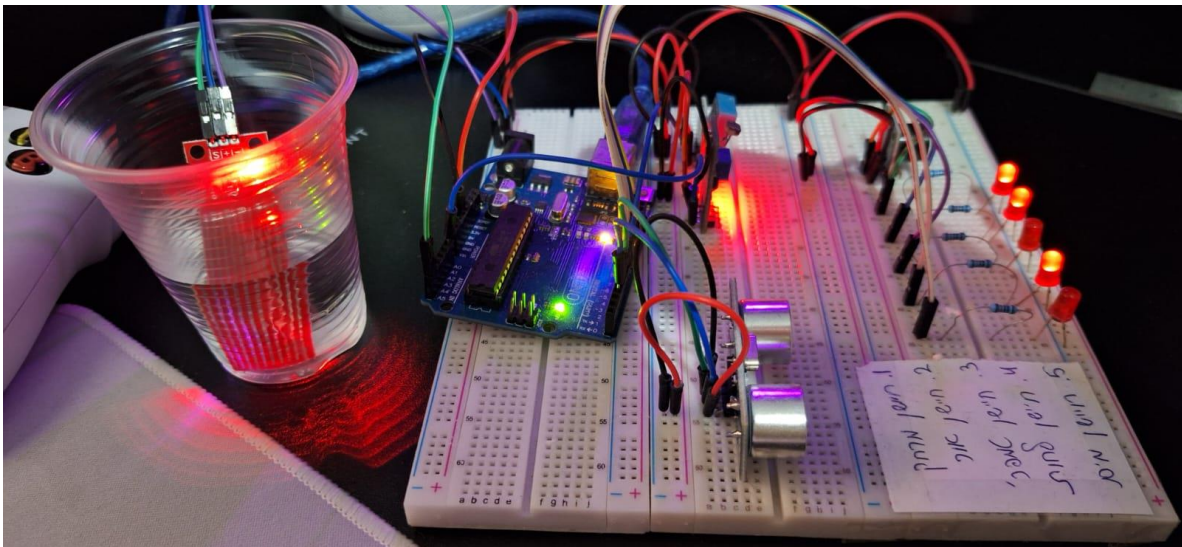
Timeseries table 🔍 ⬇️ ✕

🕒 Realtime - last minute

Timestamp ⬇️	distance	humidity	light	temperature	waterlevel
2024-01-12 09:29:07	10 cm	71 g/m³	964 lx	17 °C	0.0
2024-01-12 09:29:00	10 cm	71 g/m³	949 lx	17 °C	0.0
2024-01-12 09:28:53	10 cm	71 g/m³	952 lx	17 °C	0.0
2024-01-12 09:28:46	10 cm	71 g/m³	952 lx	17 °C	0.0
2024-01-12 09:28:39	10 cm	71 g/m³	950 lx	17 °C	0.0
2024-01-12 09:28:32	10 cm	71 g/m³	950 lx	17 °C	0.0
2024-01-12 09:28:25	10 cm	71 g/m³	951 lx	17 °C	0.0
2024-01-12 09:28:18	10 cm	71 g/m³	950 lx	17 °C	0.0
2024-01-12 09:28:11	10 cm	71 g/m³	953 lx	17 °C	0.0

טבלה 4: תוצאות של חיישן המרחק

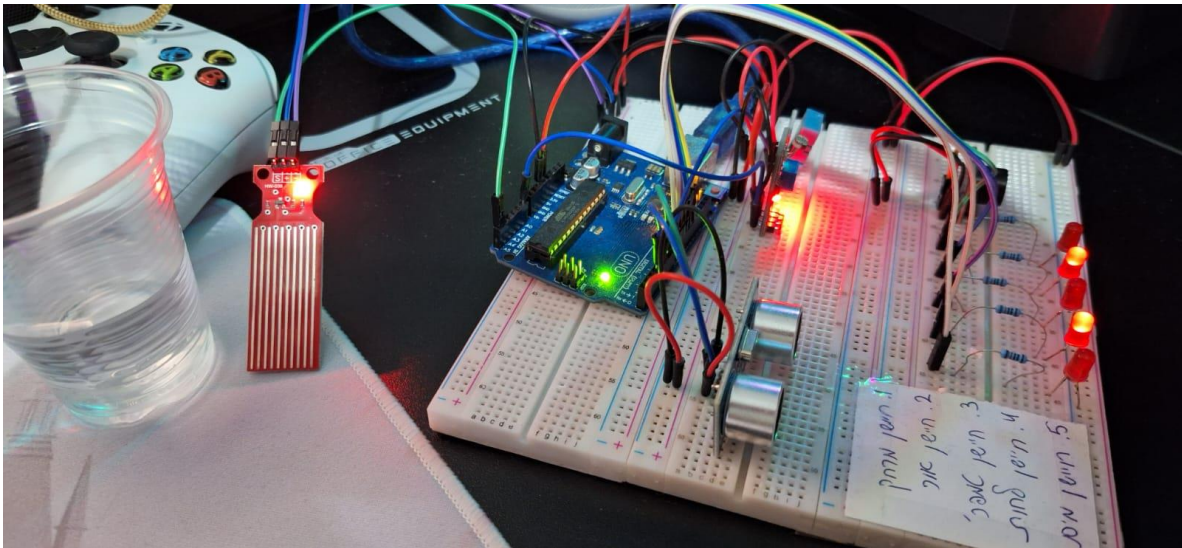
חיישן מפלס המים :



איור 12 : בדיקת חיישן גילוי המים במים

sensors					
Timeseries table					
⌚ Realtime - last minute					
Timestamp ↓	distance	humidity	light	temperature	waterlevel
2024-01-12 09:34:30	24 cm	72 g/m³	909 lx	17 °C	601.0
2024-01-12 09:34:23	24 cm	72 g/m³	902 lx	17 °C	607.0
2024-01-12 09:34:16	14 cm	72 g/m³	964 lx	17 °C	538.0
2024-01-12 09:34:09	16 cm	72 g/m³	962 lx	17 °C	542.0
2024-01-12 09:34:02	16 cm	72 g/m³	971 lx	17 °C	546.0
2024-01-12 09:33:55	16 cm	72 g/m³	963 lx	17 °C	551.0
2024-01-12 09:33:48	16 cm	72 g/m³	961 lx	17 °C	558.0
2024-01-12 09:33:41	16 cm	72 g/m³	957 lx	17 °C	561.0
2024-01-12 09:33:34	16 cm	72 g/m³	959 lx	17 °C	565.0

טבלה 5: תוצאות של חיישן גילוי המים במים







איור 13 : בדיקת חיישן גילוי המים בלי מים





Timeseries table					
🕒 Realtime - last minute					
Timestamp ↓	distance	humidity	light	temperature	waterlevel
2024-01-12 09:36:09	25 cm	72 g/m³	900 lx	17 °C	0.0
2024-01-12 09:36:02	120 cm	72 g/m³	901 lx	17 °C	0.0
2024-01-12 09:35:55	25 cm	72 g/m³	908 lx	17 °C	0.0
2024-01-12 09:35:48	120 cm	72 g/m³	904 lx	17 °C	0.0
2024-01-12 09:35:41	25 cm	72 g/m³	931 lx	17 °C	0.0
2024-01-12 09:35:33	25 cm	72 g/m³	975 lx	17 °C	0.0
2024-01-12 09:35:26	25 cm	72 g/m³	985 lx	17 °C	0.0
2024-01-12 09:35:19	25 cm	72 g/m³	909 lx	17 °C	0.0

טבלה 6: תוצאות של חיישן גילוי המים בלי מים

חיישן אור:

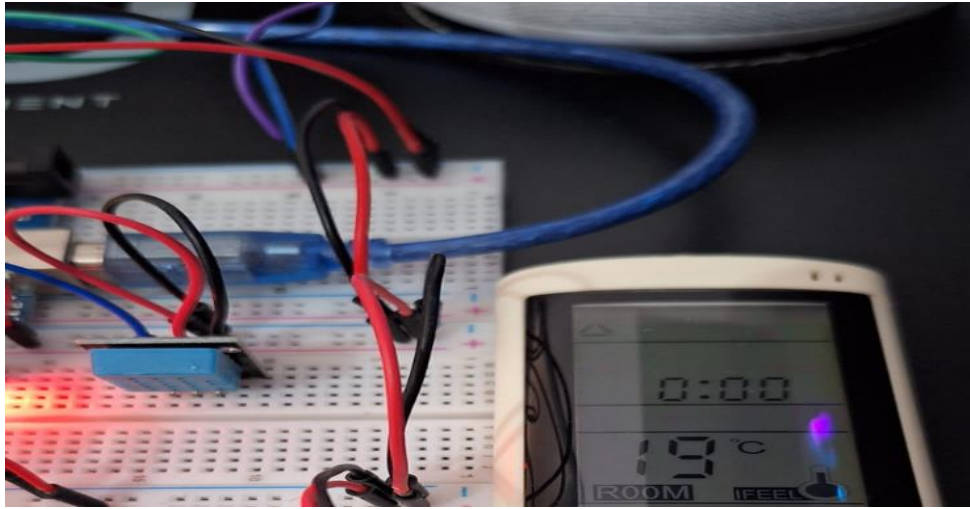
sensors					
Timeseries table   					
 Realtime - last minute					
Timestamp ↓	distance	humidity	light	temperature	waterlevel
2024-01-12 09:40:00	27 cm	72 g/m³	257 lx	17 °C	0.0
2024-01-12 09:39:53	53 cm	72 g/m³	356 lx	17 °C	0.0
2024-01-12 09:39:46	42 cm	72 g/m³	392 lx	17 °C	0.0
2024-01-12 09:39:39	42 cm	72 g/m³	480 lx	17 °C	0.0
2024-01-12 09:39:32	42 cm	72 g/m³	447 lx	17 °C	0.0
2024-01-12 09:39:25	34 cm	72 g/m³	557 lx	17 °C	0.0
2024-01-12 09:39:18	33 cm	71 g/m³	545 lx	17 °C	0.0
2024-01-12 09:39:11	42 cm	72 g/m³	614 lx	17 °C	0.0

טבלה 7: תוצאות חיישן האור בהתחלה שאין אור עד אור חלש

sensors					
Timeseries table   					
 Realtime - last minute					
Timestamp ↓	distance	humidity	light	temperature	waterlevel
2024-01-12 09:45:03	42 cm	77 g/m³	989 lx	18 °C	0.0
2024-01-12 09:44:56	42 cm	76 g/m³	997 lx	18 °C	0.0
2024-01-12 09:44:49	42 cm	76 g/m³	989 lx	18 °C	0.0
2024-01-12 09:44:42	43 cm	75 g/m³	990 lx	18 °C	0.0
2024-01-12 09:44:35	42 cm	74 g/m³	988 lx	17 °C	0.0
2024-01-12 09:44:28	42 cm	74 g/m³	980 lx	17 °C	0.0
2024-01-12 09:44:20	85 cm	74 g/m³	992 lx	17 °C	0.0
2024-01-12 09:44:13	33 cm	73 g/m³	986 lx	17 °C	0.0
2024-01-12 09:44:06	32 cm	72 g/m³	985 lx	17 °C	0.0

טבלה 8: תוצאות חיישן האור שיש אור

חיישן טמפרטורה ולחות:



איור 14: בדיקה של חיישן הטמפרטורה ולחות

sensors

Timeseries table

🕒 Realtime - last minute

Timestamp ↓	distance	humidity	light	temperature	waterlevel
2024-01-12 10:02:03	41 cm	76 g/m³	741 lx	19 °C	0.0
2024-01-12 10:01:56	37 cm	76 g/m³	770 lx	19 °C	0.0
2024-01-12 10:01:49	30 cm	76 g/m³	766 lx	19 °C	0.0
2024-01-12 10:01:42	29 cm	75 g/m³	777 lx	18 °C	0.0
2024-01-12 10:01:35	33 cm	74 g/m³	777 lx	18 °C	0.0
2024-01-12 10:01:28	33 cm	73 g/m³	774 lx	18 °C	0.0
2024-01-12 10:01:21	34 cm	72 g/m³	763 lx	17 °C	0.0
2024-01-12 10:01:14	35 cm	71 g/m³	809 lx	17 °C	0.0

טבלה 9: תוצאות של חיישן טמפרטורה ולחות

4.1. מערך הניסוי

מערך הניסויים מורכב מהשלבים הבאים:

- הגדרת רכיבי המערכת וחיבורם לחשמל ולאינטרנט
- הגדרת שירות הענן ולוח המחוונים האינטרנטי כדי לקבל ולהציג את נתוני החיישנים
- כיול החיישנים והבטחה שהם פועלים כהלכה
- הצבת החיישנים בסביבות שונות והתאמת הפרמטרים, כגון רמת האור, רמת הלחות, הטמפרטורה, המרחק ומפלס המים
- מדידת קריאות החיישנים באמצעות לוח המחוונים האינטרנטי והקלטתן
- מדידת אותם פרמטרים באמצעות התקני הייחוס או השיטות והקלטתם
- השוואת קריאות החיישנים עם מדידות הייחוס וחישוב השגיאה והדיוק
- מדידת קצב העברת הנתונים, צריכת החשמל ואיכות ממשק המשתמש באמצעות הכלים והשיטות המתאימים והקלטתם
- ניתוח התוצאות והסקת מסקנות

4.1.1. מכשירי המדידה

- מכשירי המדידה הם המכשירים או השיטות המשמשים למדידת אותם פרמטרים כמו החיישנים באותה סביבה ולהשוות אותם עם קריאות החיישנים. מכשירי המדידה הם כדלקמן:
- אפליקציית מד אור לוקס בסמארטפון למדידת רמת עוצמת הארה בלוקס בתנאי אור שונים, כגון בהיר, עמום וחשוך
 - מד לחות למדידת הלחות היחסית באחוזים בסביבות שונות, כגון חדר יבש וחדר אמבטיה לאחר מקלחת
 - מדחום למדידת טמפרטורת הסביבה במעלות צלזיוס עם ובלי מזגן
 - סרגל למדידת המרחק מהחיישן לעצם בסנטימטרים בטווחים שונים, כגון 10, 20, 30 ו-40 ס"מ
 - מדידת מפלס המים ברמות שונות

4.1.2. רכיבי המערכת

רכיבי המערכת הם רכיבי החומרה והתוכנה המרכיבים את מערכת הבקרה והניטור של החיישנים המעלים את הנתונים לענן באמצעות raspberry pi ו-Arduino uno.
רכיבי המערכת הם כדלקמן:

- 4 raspberry pi A דגם B כבקר הראשי ומעבד הנתונים
- לוח Arduino uno כממשק החיישן ומשדר הנתונים
- סט חיישנים כגון: אור, לחות, טמפרטורה, מרחק (ultrasonic) ומפלס מים, המחוברים ללוח Arduino uno באמצעות פינים אנלוגיים ודיגיטליים
- כבל USB המחבר את לוח ה-Arduino uno ל-raspberry pi דרך יציאה טורית
- מודול Wi-Fi המחבר את ה-raspberry pi לאינטרנט
- שירות ענן thingsboard שמקבל ומאחסן את נתוני החיישן מה-raspberry pi
- נגדים ($1K\Omega$)
- לידיים
- באזיר

5. תוצאות ומסקנות

מערכת הבדיקות והניסוי הופעלה בסדרת בדיקות בתנאים שונים , בוצע סדרה של ניסויים לבדיקת התוצאות ואימותן עבור סוגי חיישנים שונים: אור, לחות, טמפרטורה, מרחק ומפלס מים. עבור כל חיישן, נעשה שימוש במכשיר ייחוס או בשיטה כדי למדוד את אותו פרמטר באותה סביבה ולהשוות אותו עם קריאת החיישן. עבור חיישן האור, בוצעה בדיקה באפליקציית מד אור LUX בסמארטפון כדי למדוד את רמת עוצמת ההארה בLUX בתנאי אור שונים, כגון בהיר, עמום וחשוך, והשווה עם פלט החיישן. עבור חיישן הלחות, נבדק החיישן בסביבות שונות עם רמות לחות משתנות, כגון חדר יבש וחדר אמבטיה לאחר מקלחת, והשווה עם תפוקת החיישן. עבור חיישן הטמפרטורה, נעשה שימוש במדחום כדי למדוד את טמפרטורת הסביבה במעלות צלזיוס עם וברי מיזוג אוויר, והשווה עם תפוקת החיישן. עבור חיישן המרחק, נעשה שימוש בחיישן ultrasonic כדי למדוד את המרחק מהחיישן לעצם בסנטימטרים בטווחים שונים, כגון 10, 20, 30 ו-40 ס"מ, והשווה עם פלט החיישן באמצעות מדידה עם סרגל. עבור חיישן מפלס המים, בדקתי רמות מים שונות במיכל , כגון כוס ריק או מלא ממים, והשווה עם תפוקת החיישן.

5.1 תוצאות הניסויים/תוצאות המדידות/תוצאות הפעלת המערכת

כדי לבדוק את מערכת הבקרה והניטור של החיישנים, בוצע סדרה של ניסויים לבדיקת התוצאות ואימותן עבור סוגי חיישנים שונים: אור, לחות, טמפרטורה, מרחק ומפלס מים. עבור כל חיישן, נעשה שימוש במכשיר ייחוס או בשיטה כדי למדוד את אותו פרמטר באותה סביבה ולהשוות אותו עם קריאת החיישן. עבור חיישן האור, נעשה שימוש באפליקציית מד אור LUX בסמארטפון כדי למדוד את רמת עוצמת ההארה ב-LUX בתנאי אור שונים, כגון בהיר, עמום וחשוך, והשווה עם פלט החיישן. עבור חיישן הלחות, נבדק את החיישן בסביבות שונות עם רמות לחות משתנות, כגון חדר יבש וחדר אמבטיה לאחר מקלחת, והשווה עם תפוקת החיישן. עבור חיישן הטמפרטורה, נעשה שימוש במדחום כדי למדוד את טמפרטורת הסביבה במעלות צלזיוס עם וברי מיזוג אוויר, והשווה עם תפוקת החיישן. עבור חיישן המרחק, נעשה שימוש בחיישן ultrasonic כדי למדוד את המרחק מהחיישן לעצם בסנטימטרים בטווחים שונים, כגון 10, 20, 30 ו-40 ס"מ, והשווה עם פלט החיישן באמצעות מדידה עם סרגל. עבור חיישן מפלס המים, נבדקה רמות מים שונות במיכל, כגון כוס ריק או מלא ממים, והשווה עם תפוקת החיישן. התוצאות הראו כי קריאות החיישנים היו עקביות ומדויקות עם מדידות הייחוס, עם מרווח שגיאה בטווח המקובל. התוצאות גם הראו שמערכת הבקרה והניטור של החיישנים הצליחה להעלות את הנתונים לענן באמצעות raspberry pi ו-Arduino uno, ולהציג את הנתונים באינטרנט. התוצאות אימתו שהמערכת עבדה כשורה ועמדה ביעדי הפרויקט.

5.2. מסקנות הפרויקט

פרויקט זה סובב סביב תכנון ויישום מוצלחים של מערכת בקרה ניטור ושליטה חיישנים הממנפת את היכולות המשולבות של Raspberry Pi ו-Arduino Uno. באמצעות סדרה של ניסויים שנערכו בקפידה, נבדקה פונקציונליות המערכת ביסודיות, ונמדדו ביצועיה. נבדקו פרמטרים סביבתיים שונים והתוצאות הושוּוּ באופן שיטתי עם התוצאות הצפויות. הסינרגיה בין Raspberry Pi כיחידת העיבוד המרכזית לבין ממשק חיישני הטיפול ב-Arduino Uno התבררה כשילוב מנצח. המערכת סיפקה באופן עקבי נתונים מדויקים ואמינים, והדגימה את יעילותה ביישומים בעולם האמיתי. כעדות לאמינותה, התוצאות הנמדדות תאמו באופן הדוק עם הערכים הצפויים, מה שמאשר את מיומנות המערכת ברכישת נתונים. ביצוע מוצלח של ניסויים אלה מדגיש את חוסנה של המערכת, ומציין אבן דרך משמעותית בפיתוח רשתות חיישנים ופתרונות ניטור המחוברים לענן.

5.3. סיכום

שיאו של פרויקט זה לא רק מעיד על מימוש מוצלח של מערכת בקרה ניטור ושליטה חיישנים אלא גם מדגיש את כדאיותה המעשית. תהליך הניסוי הקפדני, שכלל מדידה מדוקדקת והשוואת תוצאות, סיפק עדות אמפירית לאמינות המערכת ולדיוקה. האינטגרציה החלקה של Raspberry Pi ו-Arduino Uno הוכיחה את עצמה כמשמעותית בהשגת יעדי הפרויקט. יש לציין שהמערכת עמדה בציפיות באופן עקבי, ואישרה את יכולתה לנטר ולהעביר נתונים סביבתיים בצורה מדויקת לענן. תוצאה מוצלחת זו מאמתת את האפקטיביות של רכיבי החומרה שנבחרו ואת ארכיטקטורת המערכת הכוללת. כתוצאה מכך, פרויקט זה תורם לנוף ההולך וגדל של יישומי IoT וחיישנים, תוך שימת דגש על הפוטנציאל של מערכות המחוברות לענן עבור תרחישי ניטור בעולם האמיתי. השגת יעדי הפרויקט מאשרת את הפונקציונליות התקינה של המערכת וממצבת אותה כנכס בעל ערך עבור יישומים מגוונים הדורשים חिשה ושליטה מרחוק.

6. רשימות

6.1 רשימת טבלאות

טבלה 1 :	תיאור כניסות ויציאות של ארדואינו	26
טבלה 2:	תיאור החיבורים לראספרי פאי	28
טבלה 3 :	תוצאות מערכת בענן	51
טבלה 4 :	תוצאות של חיישן המרחק	52
טבלה 5 :	תוצאות של חיישן המים במים	53
טבלה 6 :	תוצאות של חיישן המים בלי מים	54
טבלה 7 :	תוצאות חיישן האור שאין אור	55
טבלה 8 :	תוצאות חיישן האור שיש אור	55
טבלה 9 :	תוצאות של חיישן טמפרטורה ולחות	56

6.2 רשימת איורים

איור 1 :	המעגל החשמלי של המערכת	22
איור 2 :	המעגל החשמלי בנוי	23
איור 3 :	ארדואינו	25
איור 4 :	סכמה חשמלית של ארדואינו	25
איור 5:	סכימה של ראספרי פאי	27
איור 6:	ההדקים של GPIO ב Thingsboard	27
איור 7 :	חיישן מפלס מים	29
איור 8 :	חיישן אור	30
איור 9 :	חיישן מרחק	31
איור 10 :	חיישן טמפרטורה ולחות	32
איור 11 :	בדיקת חיישן המרחק	52
איור 12 :	בדיקת חיישן גילוי המים במים	53
איור 13 :	בדיקת חיישן גילוי בלי מים	54
איור 14 :	בדיקה של חיישן הטמפרטורה ולחות	56

7. נספחים

7.1. נספח א': נתוני החיישנים

חיישנים	סימן	יחידות
חיישן אור	Light	[lx]
חיישן מרחק	Distance	[cm]
חיישן טמפרטורה	Temperature	[C]
חיישן לחות	Humidity	[g/m ³]
חיישן מפלס מים	Water level	אין יחידות

8. מקורות ספרותיים

1. Arduino Water Level Sensor - DIY Engineers
2. Light sensor LDR resistive for Arduino - Okystar Botland - Robotic Shop
3. ThatsEngineering - YouTube
4. Welcome To Roles Academy (youtube.com)
5. ROS2 Tutorial - ROS2 Humble 2H50 [Crash Course] (youtube.com)
6. Learn To Code Fast!! (youtube.com)
7. Distance measurement using Ultrasonic sensor and Arduino - GeeksforGeeks
8. "Key Sensor Technology Components: Hardware and Software Overview"⁴
9. "Smart Sensors, Measurement and Instrumentation"⁵
10. "Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications"⁶
11. "Modeling, Programming and Simulations Using LabVIEW™ Software"⁷
12. "Cloud Services For Dummies, IBM Limited Edition"¹
13. 2. "Mastering Cloud Storage: Navigating cloud solutions, data security, and cost optimization for seamless digital transformation"² 1.
"Arduino Cookbook, 3rd Edition, by Michael Margolis"⁸
14. "Getting Started with Arduino"⁸
15. "Arduino Workshop: A Hands-on Introduction with 65 Projects"⁹
16. "Exploring Arduino"⁹
17. "Science and Engineering Projects Using the Arduino and Raspberry Pi"^[10]
18. "PLC Programming with the Raspberry Pi and the OpenPLC Project: ModbusRTU and ModbusTCP Examples with the Arduino Uno and ESP8266"¹¹
19. Getting Started With Raspberry Pi, 4th Edition By Shawn Wallace, Matt Richardson, Wolfram Donat