

הפקולטה להנדסה  
המחלקה להנדסת חשמל ואלקטרוניקה

## אפליקציה מבוססת טכנולוגיית IOT

"בית חכם"

פרויקט גמר

מוגש על ידי:

אלומה אליה      ת.ז. 318405917      מייל alooma22@gmail.com

מנחה: דוקטור דניאל רוזבן

אחראי אקדמי: פרופסור אמיר אברמוביץ

תאריך: 3.1.22



## תקציר

ספר זה מציג את התכנון והמימוש של מערכת אוטומציה ביתית משולבת אפליקציה המבוססת על טכנולוגיית IOT.

במבוא הספר יוצג מהי מערכת "בית חכם", חשיבותה לאדם הפרטי, אלו מוצרים נכללים בה וכיצד התפתחות תחום ה-IOT מאפשרת אותה. בנוסף, תפורט סקירה של מודל התקשורת עליו מבוססת מערכת הבית החכם, הפרוטוקולים והספריות הנדרשים למימוש המערכת וממשק אפליקציה המאפשר גישה נוחה למשתמש.

מטרת הפרויקט היא בניית פלטפורמה המאפשרת שליטה וניטור של המכשירים הבייתיים מרחוק על מנת לשפר את איכות החיים של המשתמש, להגדיל את אורך החיים של המכשיר ולאפשר נגישות תוך תשומת דגש על ההיבט הכלכלי והבטיחותי.

בפרק תכנון ומימוש המערכת, תתואר חלוקת המערכת לשני חלקים מרכזיים ויוגדרו היחסים בין היחידות המרכיבות אותה. היחידות פועלות בסביבה הביתית וניתנות לשליטה באמצעות אפליקציה מכל מקום. עבור כל יחידה יפורטו דרישות התכנון ועבור כל מעגל חשמלי תוצג סימולציה שהורכבה באתר הסימולציות "Wokwi", תמונה של המעגל המעשי ותרשים זרימה עבור הקוד הצרוב על הבקר ביחידה.

יחידת האפליקציה תתואר באמצעות מסמך "אפיון אפליקציה" המתאר את רכיבי האפליקציה, פרטי המימוש שלהם והאופן בו הם מתקשרים אחד עם השני. בחלק המימוש עבור קוד האפליקציה יוצגו עבור כל מחלקה מטרת המחלקה, הספריות הנדרשות וממשק המחלקה.

בפרק הבדיקות, תופעל המערכת בשלמותה ויבדקו תגובות תתי המערכות מבחינת החומרה, תוכנת האפליקציה והשילוב ביניהן. תיבדק תגובת היחידות השונות במערכת אל מול בקשות המשתמש. בנוסף תיבדק יכולת הסנכרון של המערכת אל מול מספר משתמשים בו זמנית.

בפרק תקלות ופתרון יתוארו הקשיים שעלו במהלך בניית הפרויקט ודרך ההתמודדות איתם.

בפרק תוצאות ומסקנות יודגשו תובנות והסברים שעלו במהלך הפרויקט ולסיכום תימדד הצלחת הפרויקט עבור הקריטריונים שהוגדרו במטרת הפרויקט.

## הכרת תודה

"דברים גדולים נעשו על ידי סדרה של דברים קטנים שקובצו יחד"

וינסנט ואן גוך

בסימן זה, ארצה להודות לאנשים שתרמו מהידע ומהזמן שלהם על מנת לסייע בבניית פרויקט הגמר שלי.

לדוקטור דניאל רוזבן, על ההנחיה, הזמינות, והליווי האישי בתהליך.

לאבישי אליה, על התמיכה המקצועית והמעמיקה בנושא "רשתות מחשבים".

ליאיר סמדר, על הסיוע בזיהוי ופתרון בעיות בתחום התוכנה.

למשפחה שלי, על הכל.

בעבור כל זאת, ובעבור הדחיפה והאמון לאורך כל הדרך,

תודה,

אלומה.

## תוכן עניינים

7	1. מבוא	
7	1.1 בית חכם	
7	1.2 התפתחות	
8	1.3 IOT (Internet Of Things)	
8	1.4 המוצרים	
8	1.5 הצורך	
8	1.6 פרוטוקול תקשורת	
9	1.7 מודל ה- OSI ומודל ה- TCP/IP	
10	1.8 Port	
11	1.9 IP (Internet Protocol)	
11	1.10 MQTT (Message Queuing Telemetry Transport)	
12	1.11 מתווך (Broker)	
12	1.12 ספריות paho ו- pub sub-client	
12	1.13 One-wire protocol	
13	1.14 אפליקציה	
13	1.15 ממשק משתמש (user interface)	
13	1.16 Kivy	
13	1.17 מטרת הפרויקט	
14	2 תיאור המערכת	
14	3 תכנון ומימוש המערכת	
14	3.1 סכמת בלוקים	
16	3.2 חלק ראשון: תאורה	
21	3.3 חלק שני: דוד שמש	
26	3.4 אפליקציה	
26	3.4.1 תיאור כללי	
26	3.4.2 מטרותיה העיקריות של my smart home	
26	3.4.3 ממשק המשתמש	
32	3.4.4 קוד האפליקציה	

39	בדיקות המערכת	4
41	מיתוג מערכת התאורה על ידי המתג הידני	4.1
42	מיתוג מערכת התאורה על ידי חיישן התנועה	4.2
44	מיתוג מערכת התאורה על ידי האפליקציה	4.3
46	מיתוג מערכת הדוד על ידי המתג הידני	4.4
47	מיתוג מערכת הדוד על ידי האפליקציה	4.5
50	בדיקת תקינות יחידת חיישן הטמפרטורה	4.6
51	בדיקת סנכרון בין מספר מכשירים	4.7
52	תקלות ופתרון	5
53	תוצאות ומסקנות	6
54	סיכום	7
55	מקורות ספרותיים	8
55	רשימת איורים	9
57	רשימת טבלאות	10
57	נספחים	11
57	קוד עבור הבקרים	11.1
73	קוד עבור האפליקציה	11.2
87	רקע תיאורטי	11.3

## 1. מבוא

### 1.1 בית חכם

**בית "חכם"** הינו בית בעל מערכת שליטה במערכות הביתיות כגון: תאורה, תריסים, מיזוג, תקשורת, ואבטחה.

קיימות מספר אפשרויות ליצירת בסיס תקשורת עבור הבית ה-"חכם" :

1. מערכת בית חכם על גבי קווי תקשורת נפרדים המספקת שליטה מלאה על מוצרי החשמל בבית.
  2. מערכת בית חכם על גבי קווי החשמל בבית, בה מוחלפים המתגים הקיימים הסטנדרטיים למתגים חכמים והתקשורת עוברת על גבי גל קבוע של מערכת החשמל בבית.
  3. מערכת בית חכם על גבי רשת אלחוטית או גלי רדיו, בה מוחלפים המתגים הקיימים למתגים אלחוטיים המתחברים לרשת האלחוטית של הבית.
- התחום זכה להתפתחות מואצת החל מהעשור השני של המאה ה-21 בשל מהפכת האינטרנט של הדברים -IoT.

### 1.2 התפתחות

העיקרון של מכשיר שמחובר לרשת האינטרנט הומצא בשנת 1982, כאשר חיברו לרשת מכונת משקאות של קוקה קולה באוניברסיטת קרנגי מלון בפנסילבניה. המכונה ידעה לשלוח עדכונים לגבי המלאי שלה מכל מוצר, ועל הטמפרטורה של המשקאות שנמצאים במלאי. לאחר המצאה זו, חברות אחדות החלו לאמץ את רעיון ה-IoT (המונח Things of Internet נטבע רק יותר מאוחר יותר בשנת 1999 באוניברסיטת MIT).

התחום התחיל לתפוס תאוצה רק בשנת 1999, שבה ביל ג'וי (מדען מחשב אמריקני, אחד ממייסדי חברת - Microsystems Sun, אחת חברות המחשוב המובילות, שנקנתה ב-2010 ע"י חברת Oracle) הציג את הרעיון שלפיו מוצרים יכולים לתקשר עם מוצרים שונים דרך רשת האינטרנט.



המספרים מייצגים עניין בחיפוש ביחס לנקודה הגבוהה ביותר בתרשים באזור נתון ובזמן נתון. ערך של 100 מציין את רמת הפופולריות הגבוהה ביותר שהמונח הגיע אליה. ערך של 50 משמעותו שהפופולריות של המונח הינה מחצית מהשיא. ערך של 0 משמעותו שלא היו מספיק נתונים למונח

זה. על ידי התבוננות בגרף המציג את יחס כמות החיפושים במנוע החיפוש "גוגל" של המושג "Internet Of Things" ניתן לראות היטב את התאוצה שהמונח תפס בשנים האחרונות.

### 1.3 IOT (Internet Of Things)

רשת של חפצים פיזיים המשובצים בחיישנים ובתוכנה המאפשרים תקשורת מתקדמת בין החפצים ויכולות איסוף וניתוח מידע. ה-IoT -יוצר הזדמנויות לשילוב ישיר יותר של העולם הפיזי במערכות מבוססות מחשב, וכתוצאה מכך גדלה היעילות, משתפרת התועלת הכלכלית ומתקבלת הפחתה של מאמץ אנושי.

### 1.4 המוצרים

המוצרים עשויים להיות מוצרים אקטיביים/ פאסיביים. אקטיביים: מקבלים התראה ומפעילים משהו. למשל: נורה חכמה, דוד חכם, שלט רחוק. מוצרים אלו פועלים באמצעות אפליקציה לנייד שדרכה אפשר להפעיל/לשלוט עליהם. פאסיביים: תפקידם לאסוף מידע- חיישנים. למשל: חיישן תנועה/ חיישן טמפר' חיישן רטיבות. אי אפשר להפעיל אותם ותפקידם לדווח על כל שינוי במצב. על מנת להשתמש במוצרים אלו חייב להיות מכשיר נוסף – רכז, שמבין את השפה בה החיישן מתקשר.

### 1.5 הצורך

בכל בית מכשירי חשמל הדורשים את נוכחות המשתמש על מנת להפעיל אותם. עם ריבוי מכשירי החשמל הביתיים נדרשת מהאדם יכולת שליטה ובקרה תמידית על מכשירי החשמל. על המשתמש להיות בערנות מתמדת, לכבות את המכשירים כאשר אין בהם צורך ולעמוד על המשמר עבור מכשירים היכולים להוות סכנה כגון דוד מים שעשוי להתחמם יתר על המידה ולהתפוצץ. בנוסף קיים ההכרח להפעיל בצורה ידנית את המכשירים. כל אלו כובלים את האדם לביתו ואי היכולת של מרבית בני האדם להיות בשליטה תמידית יוצרת בזבז משאבים רב ומגדילה את הסיכוי לאסון. על כן נחשף הצורך לרתום את הטכנולוגיה עבור מערכת חכמה ואמינה שתנטר את מכשירי החשמל הביתיים. מערכת שתאפשר שליטה מרחוק על מכשירים ביתיים ותזמון נכון של פעילות המכשירים, מערכת שתזהה סכנה שיוצר מכשיר חשמלי, תתריע עליה למשתמש ואף תמנע אותה.

### 1.6 פרוטוקול תקשורת

פרוטוקול תקשורת הוא נוהל לתקשורת. כלומר, אוסף של כללים המגדירים את אופן בקשת וקבלת נתונים במערכת תקשורת מסוימת וכולל כללים לייצוג המידע, איתות, אימות, ותיקון שגיאות לצורך העברת המידע בערוץ תקשורת.



## **הבטחת קבלת מידע**

כדי להבטיח שהמידע אכן מגיע ליעדו, בפרוטוקולים רבים נהוג לאשר הגעה של חבילות מידע בדרכים שונות. אישור כזה נקרא לרוב Ack (Acknowledgements). חבילות מידע שלא מתקבל עליהן אישור בטרם נקבע timeout נשלחות שוב. בצורה זו מבטיחים שכל המידע מגיע לבסוף ליעדו.

## **הבטחת תקינות מידע**

כדי להבטיח שהמידע הגיע בדיוק כפי שנשלח, משתמשים במנגנוני תקינות מידע, כמו סיכום ביקורת, CRC, סיבית זוגיות. המידע הנוסף שיש לשלוח לביצוע בדיקות התקינות נשלח כחלק מהפתיחה (header). יש לציין שלעיתים ניתן לא רק לזהות שגיאות, אלא גם לתקן אותן. עם זיהוי שגיאה, פרוטוקולים מסוימים יבקשו שליחה חוזרת של המידע הפגום, בעוד שאחרים יתעלמו מהמידע הפגום.

## **1.7 מודל ה- OSI ומודל ה- TCP/IP**

מודל זה הוא מודל רב-שכבתי שמטרתו להסביר איך מתבצעת פעולה של העברת נתונים ממכשיר א' ל-ב'. המודל מספק הסבר על כל מרכיבי הרשת ומתייחס לחלק של החומרה ולחלק של התוכנה. המודל מפרק את הרשת ל-7 שכבות שבכל שכבה יש טיפול בחלק מהתהליך תקשורת. קיימים מצבים שטיפול בגורם מסוים מטופל בשכבה אחת או יותר. בנוסף, המודל מתייחס לשיתוף פעולה בין השכבות השונות ובין השכבות בצד השני. בתהליך שידור מתחילים משכבה 7 אל שכבה 1. בתהליך קליטה מתחילים משכבה 1 אל שכבה 7.

**שכבה 7 – שכבת היישום** – זו שכבה שמייצגת את התוכנה שאיתה אנו ניגשים לאינטרנט. המידע שאנו משדרים מכיל את המסר שאותו אנו רוצים לשלוח.

**שכבה 6 – שכבת התצוגה** – השכבה קובעת באיזו צורה יוצג המידע במסך. החשיבות של השכבה נובעת עקב ריבוי מערכות הפעלה ודפדפנים שונים.

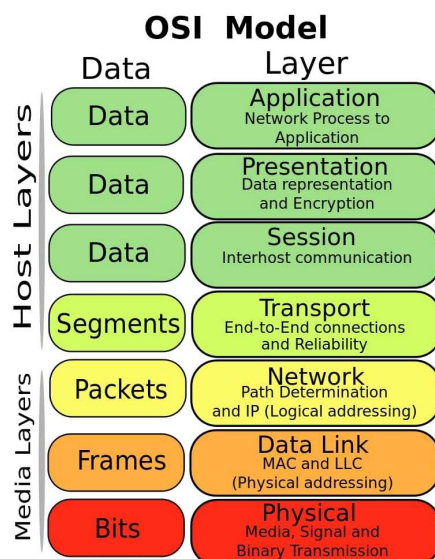
**שכבה 5 – שכבת השיחה** – אחראית על יצירת קשר לוגי בין מקור ויעד (לדוגמה התחברות לחשבון הבנק מתבצע רק לאחר שם משתמש וסיסמה נכונים).

**שכבה 4 – שכבת התעבורה** – אחראית על מעבר המידע בצורה אמינה ברשת. תפקידה לוודא שהמידע הגיע ליעדו בצורה ובסדר הנכון. שכבה זו אחראית גם על סוג השירות המבוקש, דבר המתבצע באמצעות מספרי הפורטים (Port).

**שכבה 3 – שכבת הרשת** – תפקידה למצוא מסלול מעבר המידע דרך כל רשת האינטרנט וגם הדרך אל מחשב היעד מחושב לפי כתובת לוגית (IP).

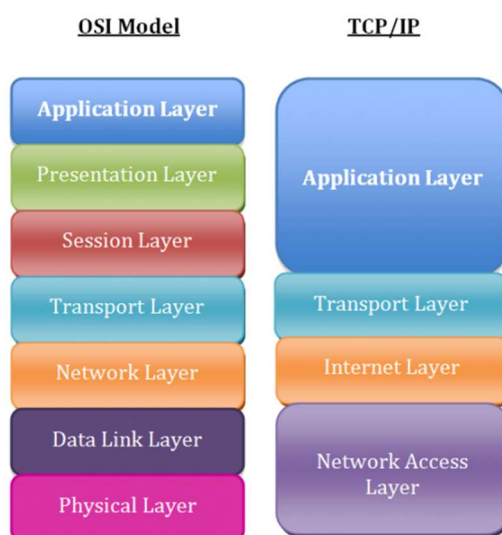
**שכבה 2 – שכבת הערוק** – אחראית על התקשורת בין שתי תחנות סמוכות. בשכבה זו מתבצעת תהליך בדיקת שגיאות. \*ברשתות מקומיות קיימת שיטה של תחרות על הקו.

**שכבה 1 – שכבה הפיזית** – מגדירה את כל ההתקנים הפיזיים שעוברים ברשת (לדוגמה: כל סוגי הכבלים, כל ציוד התקשורת, כל הסיבים האופטיים, כל ערוצי הרדיו... אלו הם השכבה הפיזית).



איור 1- מודל ה- OSI

מודל ה- TCP/IP מרכז את מודל ה- OSI ל-4 שכבות בצורה הבאה:



איור 2 מודל ה- TCP/IP בהשוואה למודל ה- OSI

## Port 1.8

פורט (port) הוא תהליך ספציפי שדרכו יכולות תוכנות להעביר נתונים באופן ישיר, במקום אמצעים אחרים כגון העברת קבצי נתונים. השימוש הנפוץ ביותר בפורט הוא בתקשורת מחשבים במסגרת הפרוטוקולים הנפוצים בשכבת התעבורה UDP/TCP. פורט מזוהה לכל כתובת או פרוטוקול מסוים על ידי מספר באורך 16 ביטים. כתובת זו נקראת "מספר הפורט".

## 1.9 IP (Internet Protocol)

פרוטוקול תקשורת המשמש להעברת נתונים ברשת האינטרנט וכתוצאה מכך הוא הפרוטוקול הנפוץ ביותר ברשתות המחשבים כיום. בפרוטוקול זה מוקצית כתובת IP לכל מחשב ברשת. ניתן להקביל כתובת IP לכתובת של דירה בעולם האמיתי, ולכן כתובת זו משמשת לצורך שליחת מידע ברשת.

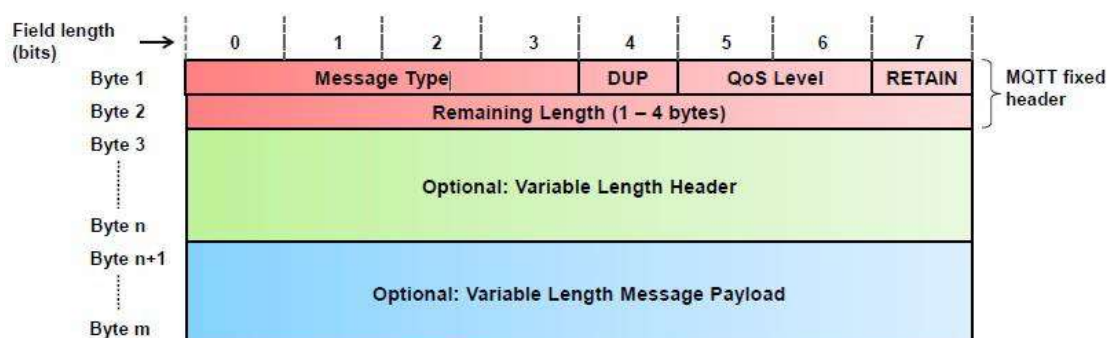
## 1.10 MQTT (Message Queuing Telemetry Transport)

MQTT הוא פרוטוקול הודעות פתוח קל משקל המספק דרך פשוטה להפיץ מידע ברוחב פס נמוך. הפרוטוקול משתמש בדפוס תקשורת פרסום/הרשמה (Publish/Subscribe). מכשיר יכול לפרסם הודעה בנושא (Topic) וכמו כן להיות מנוי לנושא מסוים כדי לקבל הודעות. בין המכשירים אין תקשורת ישירה אלא דרך המתווך (Broker) והוא מתחבר לכל הלקוחות בעזרת חיבור TCP. הפרוטוקול משמש לתקשורת מכונה למכונה (M2M) ובחיבורי אינטרנט של הדברים-IOT.

את איכות ובטיחות התקשורת ניתן לחלק ל-3 רמות:

- QoS 0 (אי ידיעה) - ההודעה נשלחת פעם אחת בלבד, ובמקרה של כישלון היא לא תימסר. משתמשים בו כאשר הוא לא קריטי.
  - QoS 1 (אשר): ההודעה תישלח פעמים רבות ככל שיידרש כדי להבטיח מסירה ללקוח. החיסרון הוא שהלקוח יכול לקבל את אותה הודעה מספר פעמים.
  - QoS 2 (מובטח) - בדומה לאמור לעיל, אך מובטח שיימסר רק פעם אחת. הוא משמש לעתים קרובות למערכות קריטיות יותר בהן יש צורך באמינות רבה יותר.
- הפקודות בפרוטוקול:

- פקודת Connect - פקודה שמטרתה ליצור תקשורת עם ה-broker ולהמתין עד שתגיע תגובה
- פקודת Disconnect - מחכה ל-broker שיסיים את התקשורת האחרונה, ומתחילה תהליך של סיום שיחה ב-TCP.
- פקודת Unsubscribe/Subscribe - בקשה של הלקוח להירשם לנושא מסוים או לכמה נושאים
- פקודת Publish - פרסום המידע מה-broker בנושא מסוים לכל הלקוחות הרשומים לנושא.



איור 3- פורמט הודעה MQTT

## 1.11 מתווך (Broker)

המכשיר או התוכנית המאפשרים ללקוח גם לפרסם וגם להירשם להודעות. שרת מקבל את חיבור הרשת של הלקוח, מקבל את הודעות הלקוח, מעבד בקשות הרשמה וביטול רישום, מעביר הודעות אפליקציה ללקוח וסוגר את חיבור הרשת של הלקוח.

**Eclipse Mosquitto** הוא מתווך הודעות בקוד פתוח (מורשה EPL/EDL) המיישם את פרוטוקול MQTT. Mosquitto הוא קל משקל ומתאים לשימוש בכל המכשירים ממחשבי לוח יחיד ועד שרתים מלאים.

**test.mosquitto.org** זהו שרת בדיקה ציבורי עבור הברוקר MQTT Mosquitto, MQTT. ניתן להשתמש בשרת הבדיקה על מנת לבדוק יישומים המשתמשים בפרוטוקול MQTT. השרת מסופק על ידי קרן Eclipse כשירות לקהילת MQTT.

## 1.12 ספריות paho ו-pub sub-client

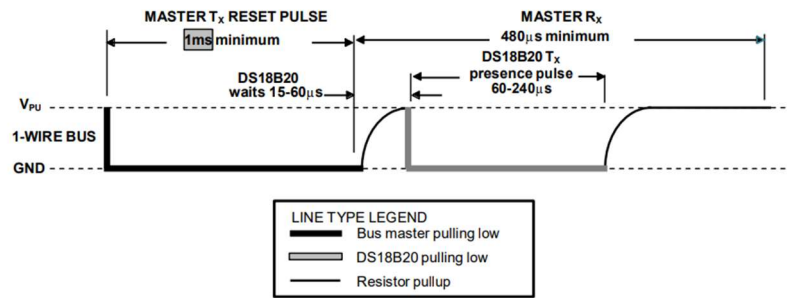
ספריות לקוח להעברת הודעות MQTT.

ספרות אלו מאפשרות לשלוח ולקבל הודעות MQTT. Paho עבור python | pub sub client עבור Arduino ide.

## 1.13 One-wire protocol

טכנולוגיית One-wire היא פרוטוקול של תקשורת טורית המשתמש בחוט אחד (החוט השני הוא אדמה) על מנת לשדר נתונים בין רכיבים. תקשורת כזו מאפשרת למיקרו בקרים להתחבר אל חיישנים כמו לחות וטמפרטורה, לג'וקים של קלט פלט מתוכנתים ולממסרים קטנים. בתקשורת One-wire קיים מאסטר (Master) אחד שמתחיל ושולט על תקשורת עם עבד (Slave) אחד או מספר עבדים. לכל אחד מרכיבי העבד יש מספר זיהוי ID ייחודי, שלא ניתן לשינוי, של 64 ביטים (8 ביטים) המשמש ככתובת הרכיב בפס Wire-1. הבתים מתארים את סוג הרכיב ותפקידו כאשר הביית האחרון. מבין ה-8 מתאר את סוג המשפחה – Code Family כמו זיהוי בלבד, רכיבי זיכרון, שמירת זמן, מדידות, חיישנים, רכיבי קלט פלט מתוכנתים או ממסרים קטנים וכו'.

המטרה היא ליצור סנכרון בין המסטר לעבד. לדוגמה רכיב אנלוגי אשר משתנה באופן רציף כל הזמן יכול לתת ערכים ללא הפסקה ולגרום למסטר לקרוא מידע לא נכון. ולכן בפרוטוקול Wire-One המאסטר שולח בקו המשותף 0 או 1 לוגי (בהתאם לחיישן ולצורת חיבור Pull-up/Pull-down) למשך זמן המוגדר מראש כאיתחול. החיישן מתאפס ומתחיל לשלוח מידע והמסטר קולט את המידע מהביט הראשון וכך הם מסונכרנים והמידע לא נאבד או מתעבת.



איור 4- דיאגרמת זמנים של קו התקשורת בין מאסטר לעבד

## 1.14 אפליקציה

אפליקציה היא יישום לטלפון חכם, לדפדפן או לטאבלט. אפליקציה סלולרית – יישום מחשב המיועד לשימוש בטלפונים חכמים, מחשבי לוח ומכשירים ניידים אחרים. יישום מחשב – תת-מחלקה של תוכנות מחשב אשר מנצל את יכולות המחשב ישירות ובאופן יסודי לביצוע משימות אותם המשתמש מבקש לבצע.

## 1.15 ממשק משתמש (user interface)

חלקה של מערכת החשוף למשתמש בה, כך שדרכו מתקיים הקשר בין המשתמש לבין המערכת. "מערכת", בהקשר זה היא תוכנה, מכונה, מכשיר או חפץ כלשהו, שנועדו לשימוש של אדם.

## Kivy 1.16

קיווי הוא מסגרת לפיתוח קוד פתוח ורב-פלטפורמה המאפשרת לפתח יישומים עם פונקציות מורכבות, ממשק משתמש ידידותי ועם מאפייני ריבוי מגע, כל זאת מכלי אינטואיטיבי, המכוון ליצירת אבות טיפוס במהירות ובעיצובים יעילים המסייעים לקודים לשימוש חוזר וקלים לפריסה.

## 1.17 מטרת הפרויקט

בניית פלטפורמה המאפשרת שליטה וניטור של המכשירים הבייניים מרחוק על מנת לשפר את איכות החיים של המשתמש, להגדיל את אורך החיים של המכשיר ולאפשר נגישות תוך תשומת דגש על ההיבט הכלכלי והבטיחותי.

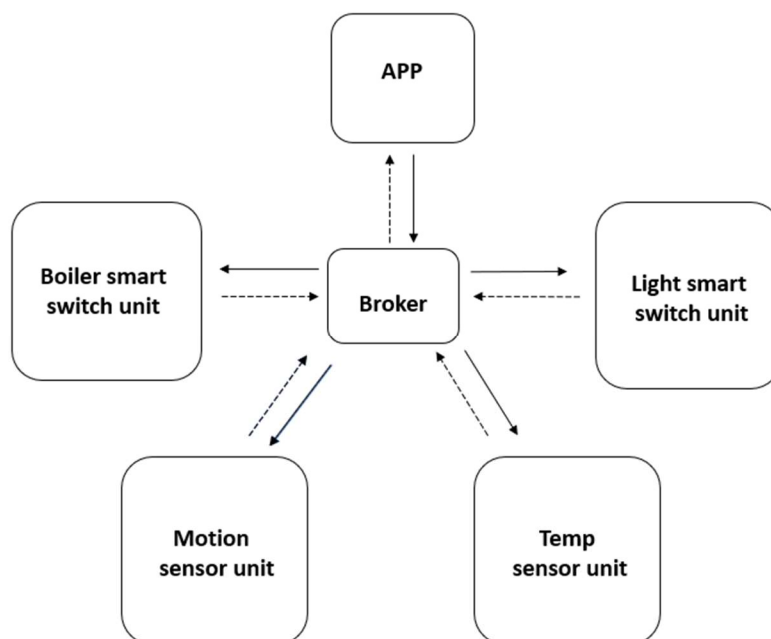
## 2 תיאור המערכת

### הסבר כללי על המערכת

מערכת הבית החכם הינה מערכת משובצת הנשלטת על ידי מספר חיישנים ואפליקציה. ניתן לחלק את המערכת לשני חלקים, חלק ראשון הוא שליטה על התאורה בחדר והחלק השני הוא קבלת מידע על טמפרטורת המים בדוד ושליטה על כיבוי והדלקת הדוד. המערכות פועלות בסביבה הביתית וניתנות לשליטה באמצעות האפליקציה מכל מקום.

## 3 תכנון ומימוש המערכת

### 3.1 סכמת בלוקים



איור 5- סכמת בלוקים

מימוש הבית החכם בוצע באמצעות חלוקה לתתי מערכות, כאשר כל מערכת תוכננה בנפרד וניתן להשתמש בה כיחידה עצמאית.

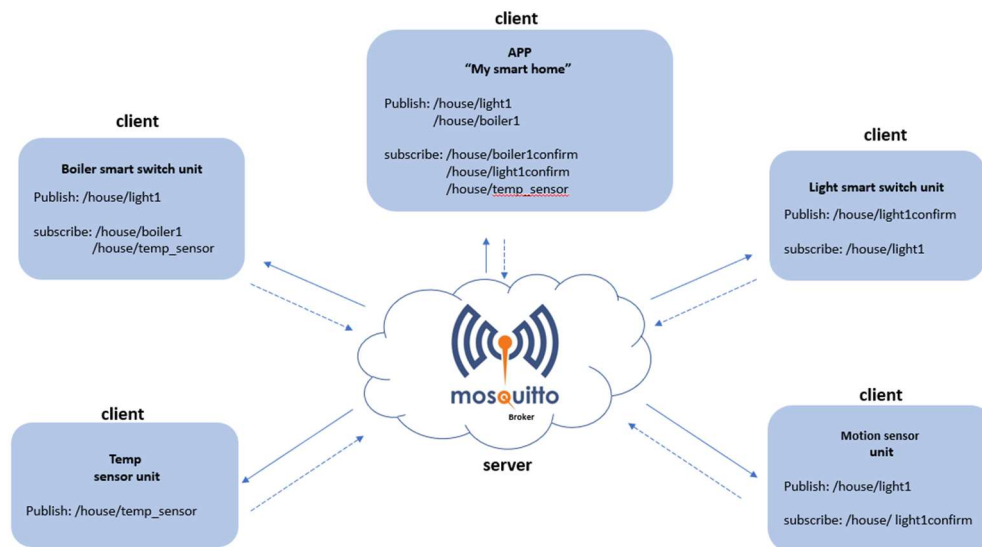
כפי שניתן לראות בסכמת הבלוקים ישנן 5 תת מערכות המתקשרות אחת עם השנייה באמצעות המתווך. תת המערכות הן יחידות החיישנים- תנועה וטמפרטורה, יחידות המתגים- תאורה ודוד השמש ויחידת האפליקציה.

יחידת חיישן התנועה מתקשרת עם יחידת מתג התאורה החכם, יחידת חיישן הטמפרטורה מתקשרת עם יחידת מתג הבוילר החכם ואילו יחידת האפליקציה מקיימת אינטראקציה עם כל תת המערכות. הפרויקט ימומש באמצעות הפרוטוקול MQTT המותאם לפרויקטים בתחום בתים חכמים הדורשים אינטראקציה מורכבת בין מכשירי הבית. הפרוטוקול ייושם באמצעות המתווך "mosquitto" דרך שרת הבדיקה [test.mosquitto.org](https://test.mosquitto.org).

החיבורים נוצרים באמצעות TCP / IP והמתווך ישמור תיעוד של הלקוחות המחוברים. כברירת מחדל, ההתקנים ישתמשו ביציאות תקשורת (port) מספר 1883 המזוהה עם הפרוטוקול.

**תכנות הבקרים:** תכנות הבקרים יעשה בשפת C++ בסביבת העבודה Arduino ide. מימוש הפרוטוקול יתבצע באמצעות ספריית הלקוח "pubSubClient" המאפשרת שליחת וקבלת הודעות MQTT.

**תכנות האפליקציה:** תכנות האפליקציה יעשה בשפת Python בסביבת העבודה PyCharm. מימוש הפרוטוקול יתבצע באמצעות ספריית הלקוח "Paho".



איור 6-מימוש פרוטוקול ה-MQTT במערכת

מוגדרים במערכת ה- topics הבאים:

- /house/light1 – אחראי על בקשות למתג האור.
- /house/light1confirm – אישור בשינוי מצב מתג האור.
- /house/temp\_sensor – מידע על הטמפרטורה הנקלטת מחיישן הטמפרטורה.
- /house/boiler1 – אחראי על בקשות למתג הדוד.
- /house/boiler1confirm – אישור בשינוי מצב מתג האור.

### יחידת המתג החכם:

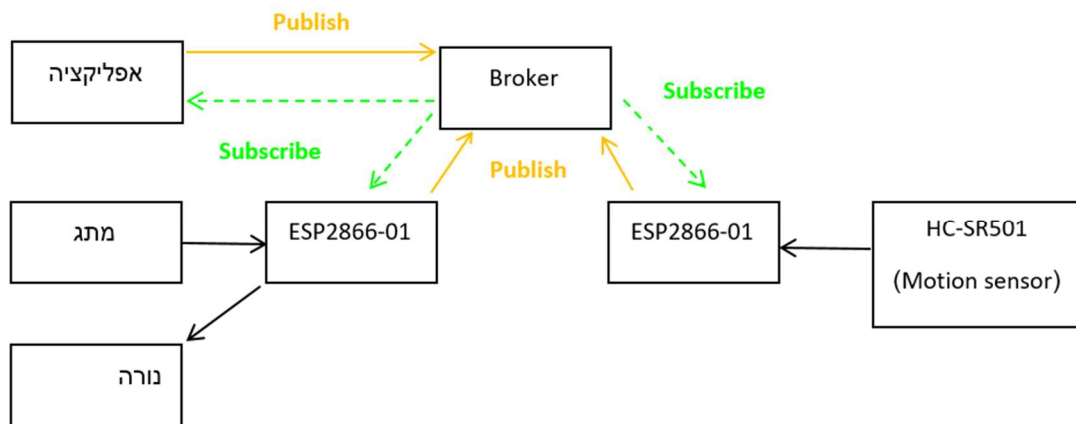
יחידת המיתוג החכמה תהיה אחראית על סנכרון בקשות ההדלקה והכיבוי של המתג וביצוען. היחידה תכיל מתג פיזי עבור כיבוי/הדלקה ובמקביל תאזין לבקשות כיבוי/ הדלקה מהמתווך. על המערכת לשלוח עדכון למנויים בכל שינוי של מצב המתג.

### יחידת החיישן:

יחידת החיישן תהיה אחראית על קריאת נתונים מהחיישן, ניהולם, ופרסומם למנויים.

## 3.2 חלק ראשון: תאורה

### 3.2.1 תכנון



איור 7- רכיבים במערכת התאורה

מערכת התאורה מורכבת מיחידת חיישן התנועה ויחידת המתג החכם. כל יחידה מורכבת מרכיב Wi-Fi – ESP8266-01 המאפשר את התקשורת הפיזית עם המתווך (Broker) וניהול של היחידה. את קטעי הקוד הצרובים על הבקרים נתן למצוא בנספחים.

#### יחידת חיישן התנועה:

על יחידה זו לקיים את התנאים הבאים:  
במידה ויש תנועה בחדר והאור כבוי תשלח בקשה ליחידת המיתוג להדליק את האור. במידה והאור דלוק ואין תנועה בחדר במשך זמן אשר הוגדר מראש תישלח הודעה ליחידת המתג לכבות את האור.

#### יחידת המתג החכם:

כמתואר בתכנון הכללי.

### 3.2.2 מימוש

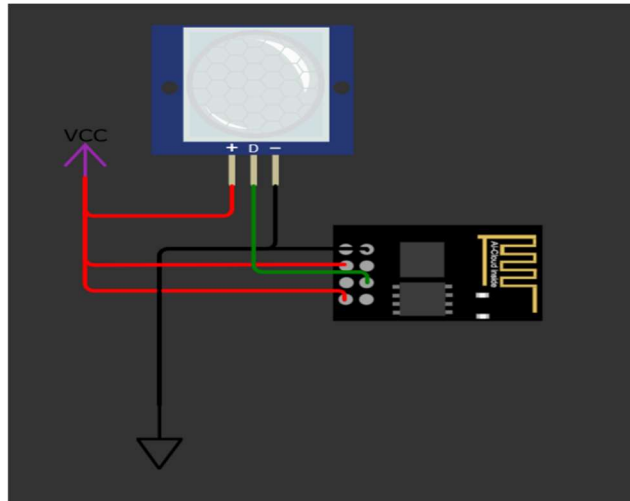
#### רכיבים

- שני רכיבי Wi-Fi esp8266-01.
- שתי מטריצות.
- שני ממסרים.
- בטריות ומתאם לבטרייה.
- חיישן תנועה- HC-SR501



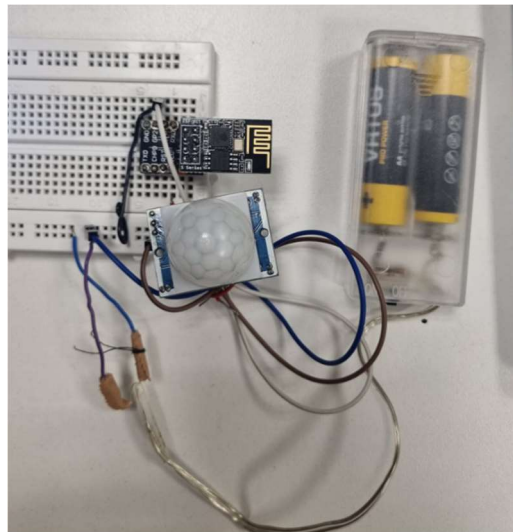
### 3.2.2.1 יחידת חיישן התנועה:

תחילה תוכנן והורכב המעגל באתר הסימולציות "Wokwi":



איור 8 - סימולציה יחידת חיישן תנועה

ולאחר מכן הורכב המעגל בפועל:



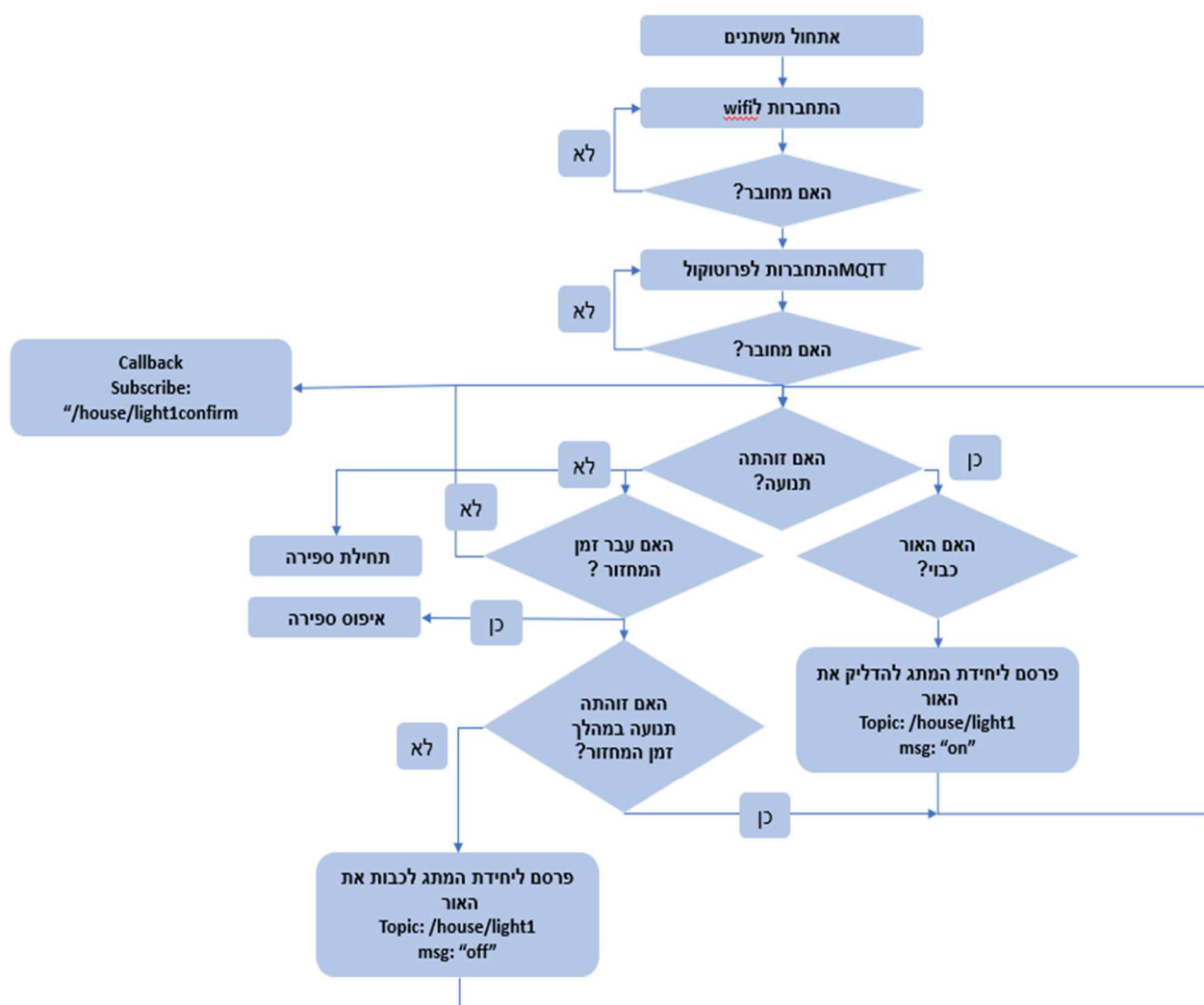
איור 9 - מימוש מעגל יחידת חיישן התנועה

המעגל מורכב מחיישן תנועה ורכיב ה-Wi-Fi ESP8266-01 – המאפשר את התקשורת הפיזית עם המתווך (Broker) ואת ניהול המעגל. על מנת לאפשר ניידות ליחידה מקור המתח של המעגל הינו בטריות.

החיישן HC-SR501 הינו מעגל המורכב מחיישן (רכיב אנלוגי) ו-BISS0001 בקר חיישן חום מדויק במיוחד כך שמוצא המעגל הוא 1 לוגי או 0 לוגי. ולכך החיישן HC-SR501 הוא חיישן דיגיטלי. הוא מגיב לשינוי במידה וקיים בחום המוצא יהפוך ל-1 לוגי ובמידה ואין המוצא יהפוך ל-0 לוגי. חיישן התנועה HC-SR501 מניב מידע באופן שוטף.

## קוד חיישן התנועה

תרשים זרימה:



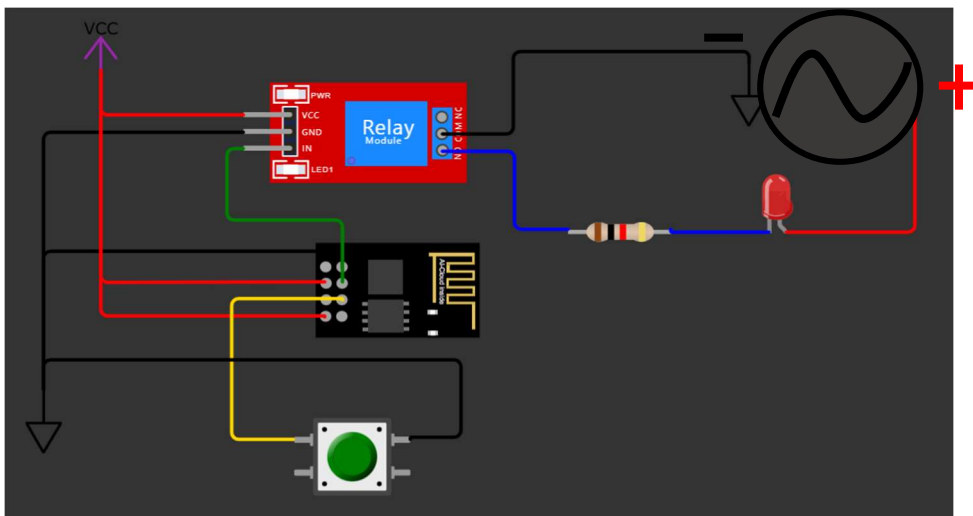
איור 10- תרשים זרימה עבור יחידת חיישן התנועה

לאחר התחברות מוצלחת לרשת ה-Wi-Fi ולפרוטוקול הרכיב נרשם כמנוי ל: `"/house/light1confirm"` והמערכת לזיהוי וניטור התנועות בחדר מתחילה לפעול. במידה וזוהתה תנועה והאור כבוי תפורסם הודעה ליחידת המיתוג להדליק את האור. במידה ולא זוהתה תנועה, תתחיל ספירה של זמן המחזור הרצוי. המשתנה `"isMove"` מקבל ערך "אמת" כאשר מזוהה תנועה בחדר, באמצעות משתנה זה ניתן לבדוק בתום זמן המחזור האם נקלטה תנועה. במידה ולא זוהתה תנועה תפורסם הודעה ליחידת המיתוג לכבות את האור.

ההודעה נשלחת ל- `topic: "/house/light1"` באמצעות המתווך. במידה ומתקבלת הודעה מ- `"/house/light1confirm"` (הנושא המאשר שינוי במצב הנורה) יעודכן המשתנה המעיד על מצב האור.

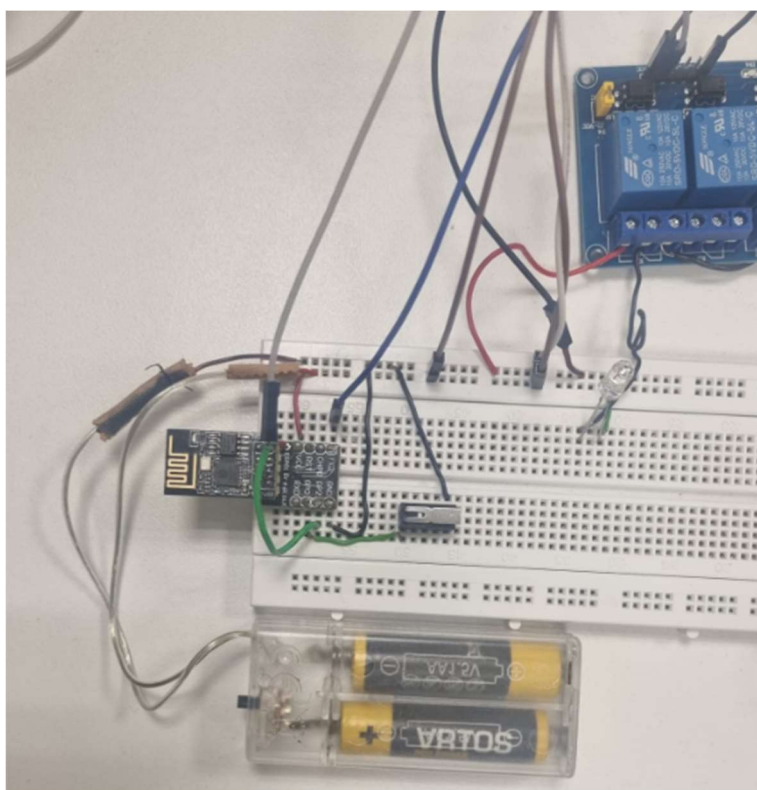
### 3.2.2.2 יחידת מיתוג האור:

תחילה תוכנן והורכב המעגל באתר הסימולציות "Wokwi":



איור 11- סימולציית יחידת מיתוג האור

ולאחר מכן הורכב המעגל בפועל:

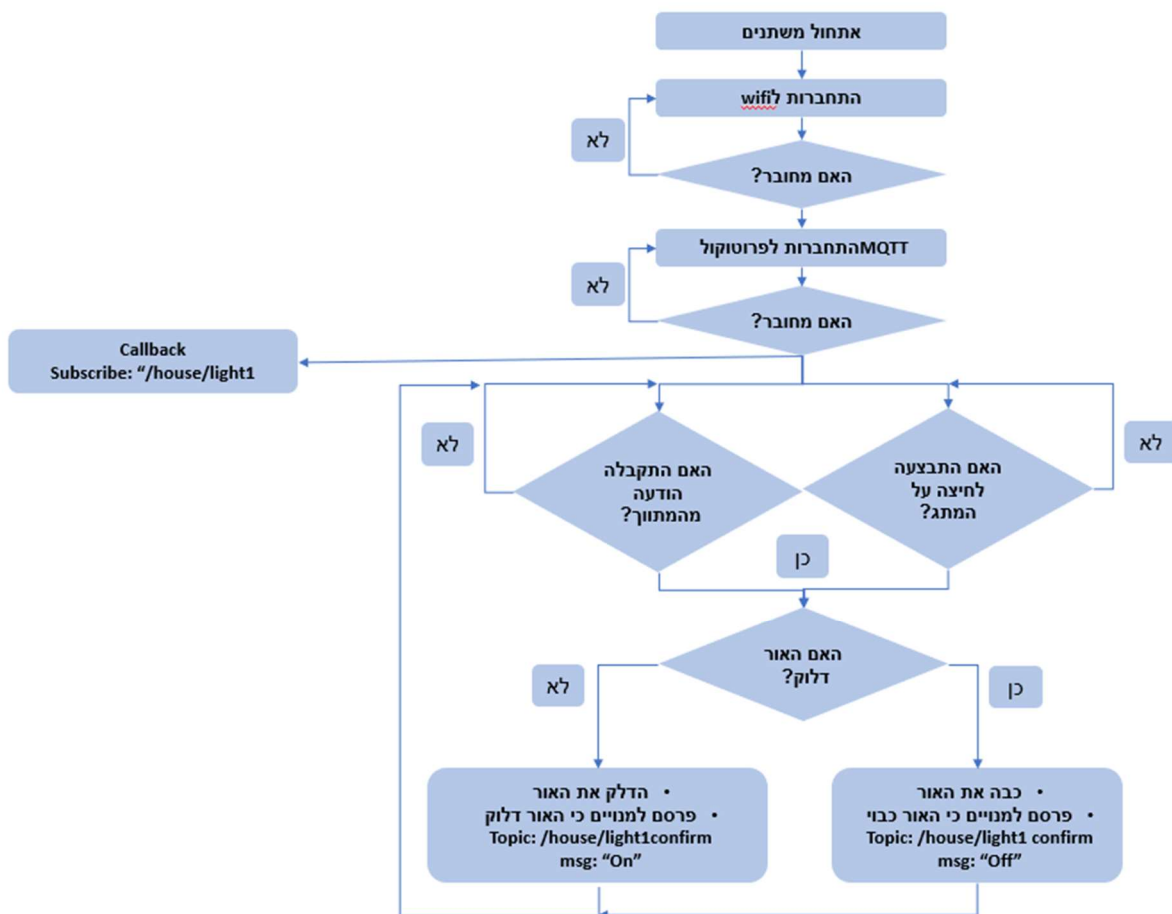


איור 12- מימוש מעגל יחידת מיתוג האור

המעגל אחראי על ניהול של הדלקה/כיבוי האור. במעגל זה משולב מתג לחיצה ידני הנגיש למשתמש, ממסר הממתג את הזרם שמגיע לנורה ורכיב ה-Wi-Fi - ESP8266-01 המאפשר את התקשורת הפיזי עם המתווך (Broker) ואת ניהול המעגל.

## קוד יחידת מיתוג האור

תרשים זרימה



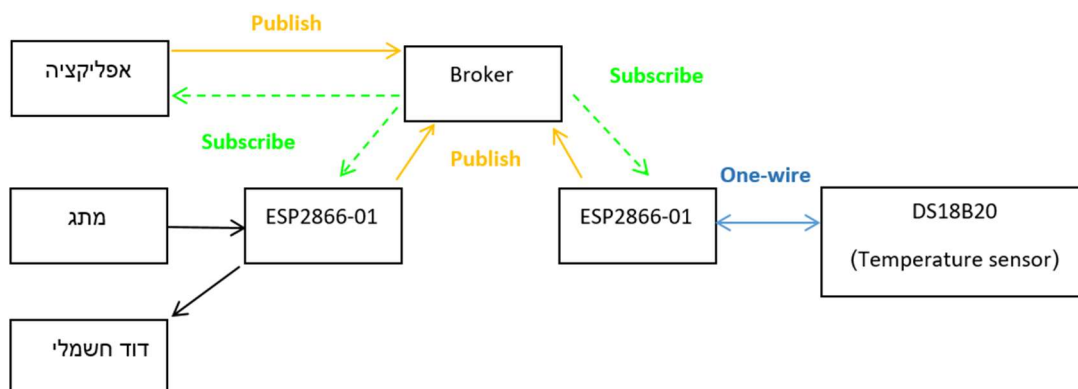
איור 13- תרשים זרימה יחידת מיתוג האור

לאחר התחברות מוצלחת לרשת ה Wi-Fi ולפרוטוקול הרכיב נרשם כמנוי ל: "/house/light1". המערכת מחכה ללחיצה על המתג או להודעה מהמתווך בנושא זה. במידה ומתקבל אחד מהשניים, יופעל הממסר והנורה תשנה את מצבה. בנוסף תפורסם הודעה למנויים ל "/house/light1confirm" על מצב האור. ההודעה נשלחת ל- "house/light1confirm" topic: באמצעות המתווך.

בקוד יש שימוש בספריית "Bounce2" המייצבת את האות המתקבל מהלחצן ומונעת קלט שווא. כאשר המשתמש לוחץ על כפתור או מתג, שני חלקי מתכת באים במגע אבל לפני ביצוע החיבור היציב בפועל חלקי המתכת מתחברים ומתנתקים מספר פעמים. דבר זה עלול לגרום להפעלה כוזבת או להפעלה מרובה כמו לחיצה על הכפתור מספר פעמים ועל כן יש להשתמש במערכת שתדע להתעלם מחיבור לא יציב. (דוגמא דומה היא הפלת כדור טניס מגובה הממשיך לקפוץ על פני השטח, עד שהוא מגיע למנוחה.)

### 3.3 חלק שני: דוד שמש

#### 3.3.1 תכנון



איור 14- דיאגרמת מלבנים של דוד שמש

מערכת הדוד מורכבת מיחידת חיישן הטמפרטורה ויחידת המתג החכם כך שהמידע עובר ביניהם באמצעות המתווך.

כל יחידה מורכבת מרכיב ה Wi-Fi - ESP8266-01 המאפשר את התקשורת הפיזית עם המתווך (Broker) וניהול של היחידה. את קטעי הקוד הצרובים על הבקרים נתן למצוא בנספחים.

#### יחידת חיישן הטמפרטורה:

כמתואר בתכנון הכללי.

על חיישן הטמפרטורה למדוד את הטמפרטורה בדוד ולשלוח אותה למנויים.

#### יחידת המתג החכם:

כמתואר בתכנון הכללי.

הדוד החשמלי יידלק ויכבה בלחיצה על מתג או בעת בקשה מהאפליקציה וניתן יהיה לשלוט במשך זמן ההדלקה באמצעות משתנה ייעודי.

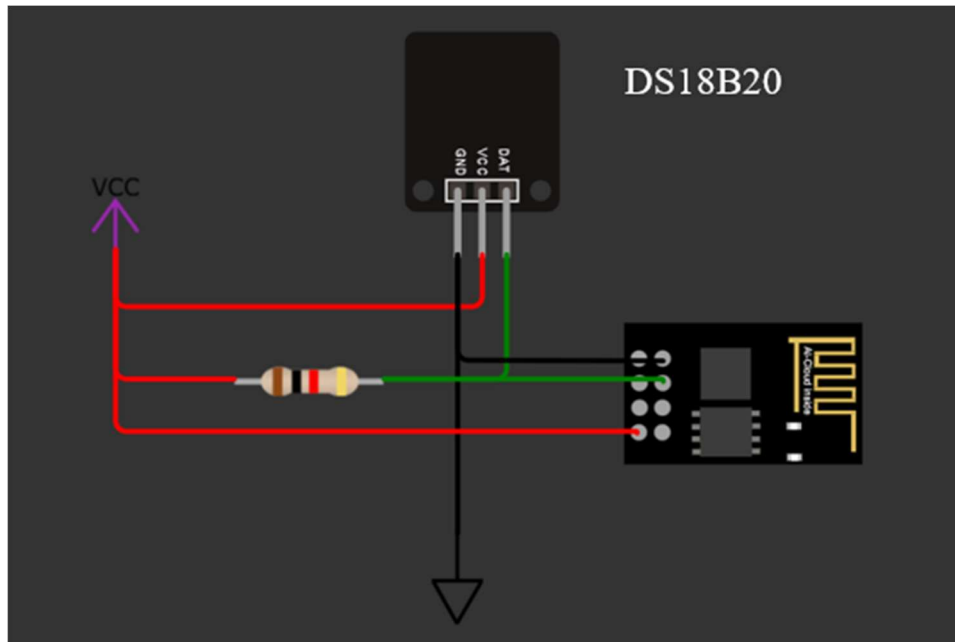
#### 3.3.2 מימוש

##### רכיבים

- שני רכיבי Wi-Fi esp8266-01.
- שתי מטריצות.
- שני ממסרים.
- בטריות ומתאם לבטרייה.
- חיישן טמפרטורה לדוד שמש- DS18B20.

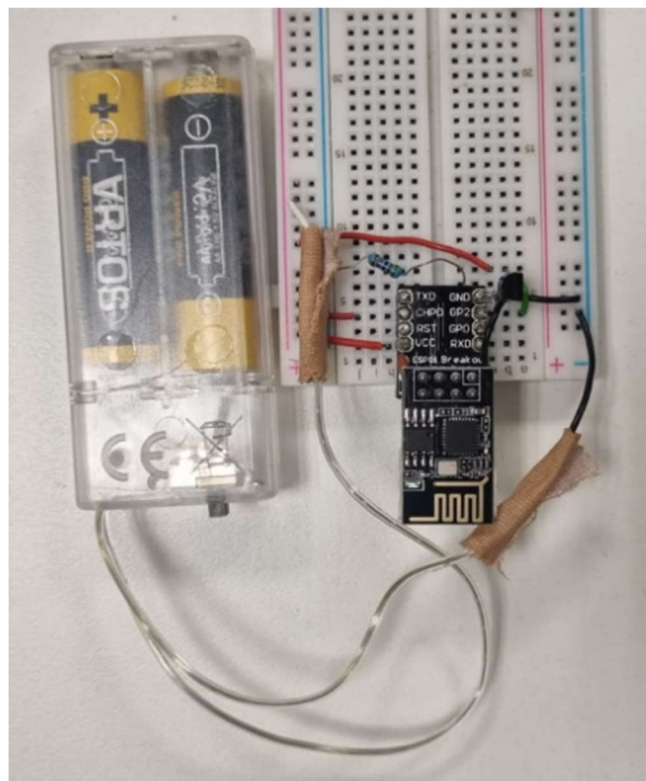
### 3.3.2.1 יחידת חיישן הטמפרטורה:

תחילה תוכנן והורכב המעגל באתר הסימולציות "Wokwi":



איור 15-סימולצית רכיב ESP8266-01 וחיישן טמפרטורה.

ולאחר מכן הורכב המעגל בפועל:



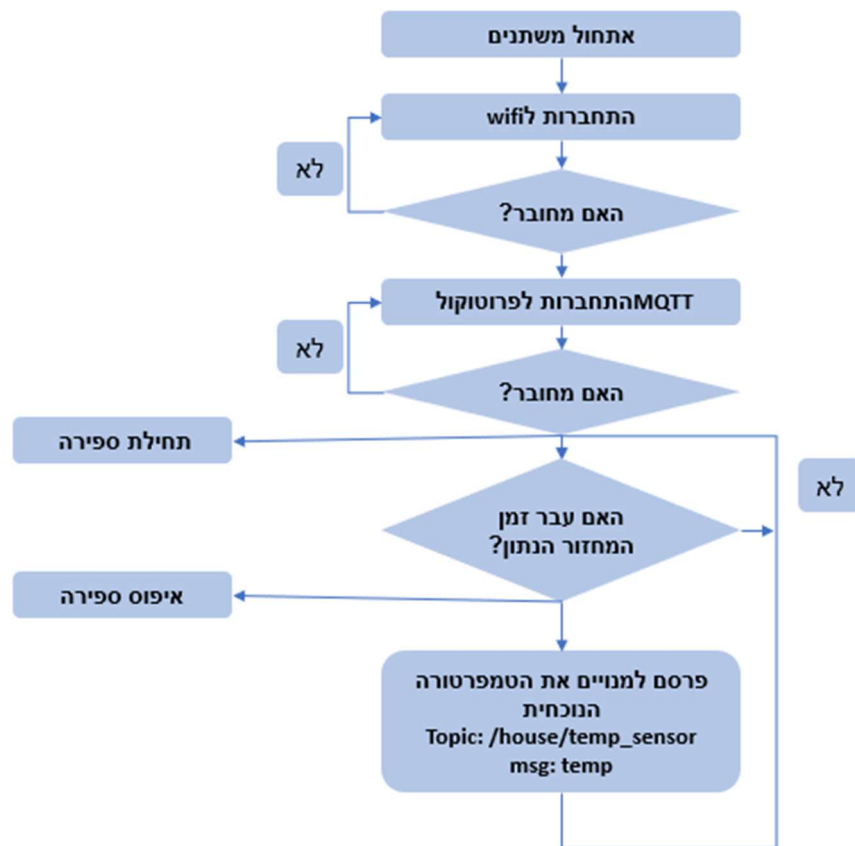
איור 16- מימוש מעגל יחידת חיישן הטמפרטורה

המעגל מורכב מחיישן טמפרטורה ורכיב Wi-Fi – ESP8266-01 המאפשר את התקשורת הפיזית עם המתווך (Broker) ואת ניהול המעגל.

הרכיבים מחוברים באמצעות נגד לספק מתח, כדי שבקו התקשורת בין החיישן ל-ESP8266-01 לפני התחלת התקשורת ימצא הקו במתח גבוה (VPUP- Up Pull). תחילת הפעולה היא כאשר המאסטר מוריד את הקו מ VPUP מתחת למתח סף VTL (מתח סף הנחשב נמוך) לרמה של '0' כדי לעבור מצב קריאה מהחיישן.

## קוד יחידת חיישן הטמפרטורה

תרשים זרימה

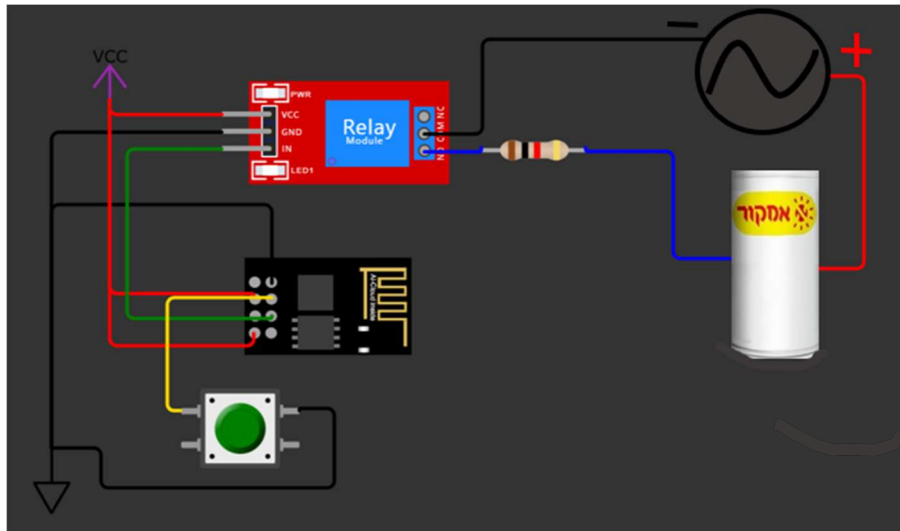


איור 17-תרשים זרימה יחידת חיישן טמפרטורה

לאחר התחברות מוצלחת לרשת ה Wi-Fi ולפרוטוקול, חיישן הטמפרטורה מתחיל לקלוט ולשדר נתונים. הנתונים נקלטים מחיישן הטמפרטורה באמצעות פרוטוקול one-wire. הטמפרטורה נשלחת למנוי ה- "/house/temp\_sensor" topic בקצב הנבחר מראש. ההודעה נשלחת באמצעות המתווך.

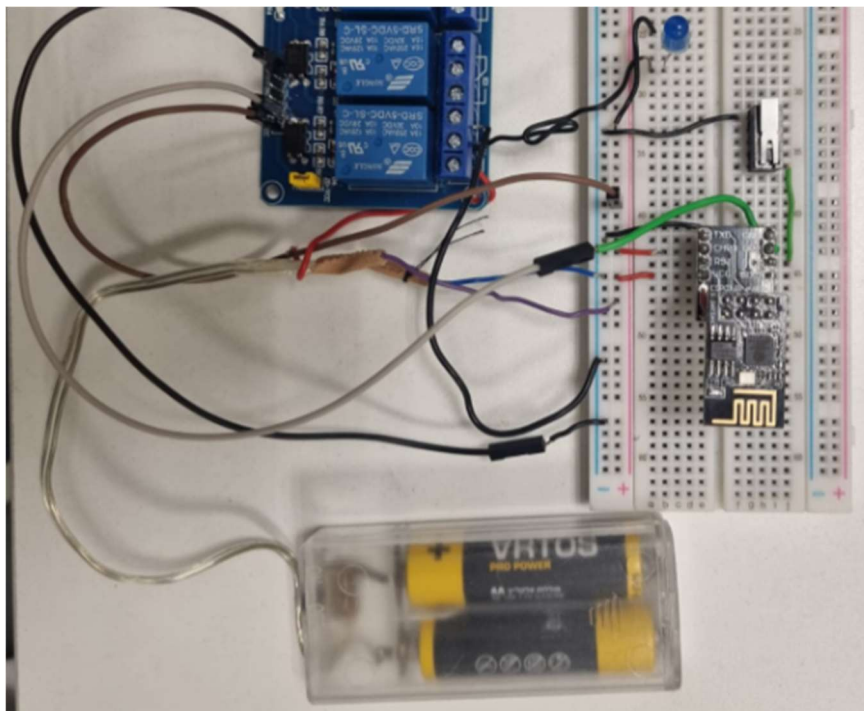
### 3.3.2.2 יחידת מיתוג דוד השמש

תחילה תוכנן והורכב המעגל באתר הסימולציות "Wokwi":



איור 18- סימולצית רכיב ESP8266-01, הממסר והמתג

ולאחר מכן הורכב המעגל בפועל:



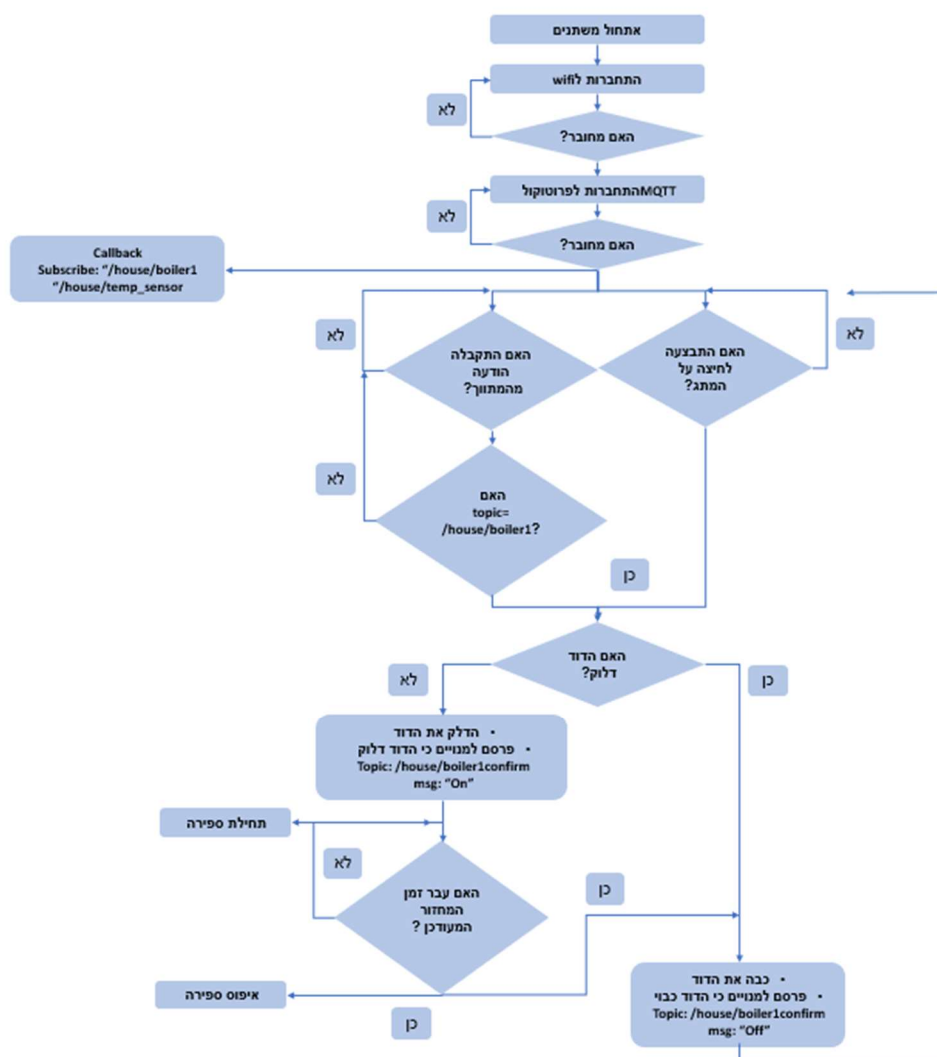
איור 19- מימוש מעגל יחידת מיתוג הדוד

המעגל אחראי על ניהול של הדלקה/כיבוי הדוד. במעגל זה משולב מתג לחיצה ידני הנגיש למשתמש, ממסר הממתג את הזרם שמגיע לדוד השמש ורכיב ה-Wi-Fi ESP8266-01 המאפשר את התקשורת הפיזית עם המתווך (Broker) ואת ניהול המעגל. המערכת תאזין לבקשות כיבוי/ הדלקה מהמתווך.



## קוד יחידת מיתוג דוד השמש

תרשים זרימה



איור 20- תרשים זרימה יחידת מיתוג דוד השמש

לאחר התחברות מוצלחת לרשת ה Wi-Fi ולפרוטוקול הרכיב נרשם כמנוי ל: `"/house/boiler1"` ול- `"/house/temp_sensor"`.

המערכת מחכה ללחיצה על המתג או להודעה מהמתווכ בנושא `"/house/boiler1"`. במידה ומתקבל אחד מהשניים, המערכת בודקת האם הדוד דלוק או כבוי. במידה והדוד דלוק, הוא יכבה. במידה והדוד כבוי- יופעל הממסר והדוד יידלק. משך הזמן בו יישאר הדוד דלוק נקבע באמצעות מכפלת זמן המאותחל בהתחלה במשתנה `numOfShowers`. ערך המשתנה `numOfShowers` נקבע באמצעות המידע המתקבל מהנושא `"/house/boiler1"` או באופן דיפולטיבי במידה והמשתמש לא שלח נתון. בנוסף תפורסם הודעה למנויים ל `"/house/boiler1confirm"` על מצב הדוד. ההודעה נשלחת ל- `topic: "house/boiler1confirm"` באמצעות המתווכ.

## 3.4 אפליקציה

### 3.4.1 תיאור כללי

My smart home הינה מערכת המאפשרת אוטומציה ביתית כך שהמשתמשים יכולים לשלוט במכשירי חשמל בביתם מכל מקום ובכל זמן באמצעות טלפון חכם. לאנשי הבית תהיה אפשרות לשלוט בהפעלה/ כיבוי האורות והפעלה/ כיבוי דוד המים למשך זמן שיקבע באופן דיפולטיבי או על פי מספר מתקלחים המוגדר על ידי המשתמש בכל הפעלה. כמו כן, יקבל המשתמש חיווי עבור מצב הפעילות (דלוק/ כבוי) של האור והדוד ועבור טמפרטורת הדוד העדכנית ובעת פתיחת האפליקציה, האפליקציה תתעדכן במצבי המתגים הפיזיים. האפליקציה קלה להתקנה ובעלת ממשק משתמש פשוט ונח.

### 3.4.2 מטרותיה העיקריות של my smart home

- ניהול המכשירים הבייתיים ממקום אחד.
- חיסכון בזמן ובכסף
- ממשק משתמש פשוט ונח
- ניתור וניהול המכשירים הבייתיים מכל מקום ובכל זמן

### 3.4.3 ממשק המשתמש

#### 3.4.3.1 עמוד "התחברות"

העמוד הראשי של האפליקציה. העמוד כולל:

■ קלט אימייל - חובה

- צריך להכיל את התווים '@' ו- ' '.
- יכולות להיות בשפה האנגלית בלבד

■ קלט סיסמה - חובה

- לפחות תו אחד (לכל היותר 30)
- יכולות להיות בשפה האנגלית בלבד

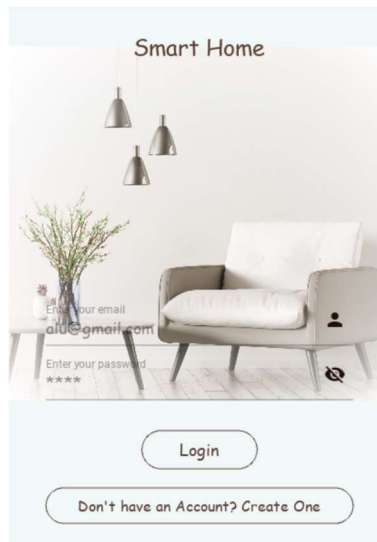
■ כפתור התחברות

• בלחיצה על כפתור ההתחברות המערכת תחפש התאמה במאגר הנתונים. במידה ולא תימצא התאמה, תוצג בחלון קופץ ההודעה "invalid username or password". לסגירת החלון יש להקיש בחלל המסך.

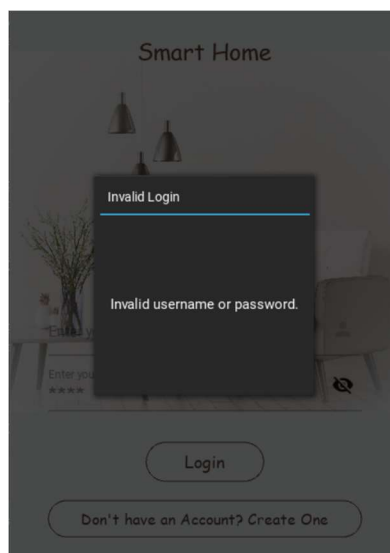
• במידה והאימות עבר בהצלחה, יועבר המשתמש אל עמוד הבית והפרטים המתאימים ממאגר הנתונים ישמרו לוקאלית ויוצגו בעת הצורך.

■ כפתור הרשמה

• המשתמש יועבר אל עמוד ההרשמה



איור 21- עמוד התחברות



איור 22- התראה על מידע לא תקין

### 3.4.3.2 עמוד "הרשמה"

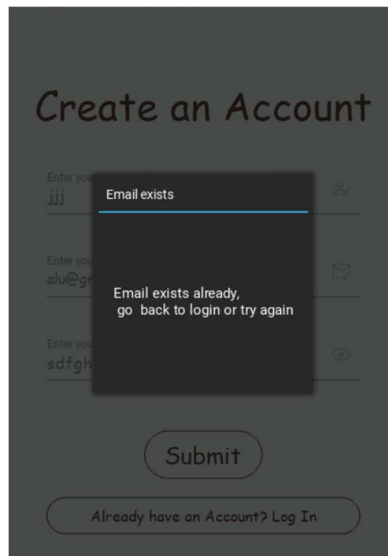
בעמוד זה יזין המשתמש את פרטיו אישיים לצורך זיהוי. העמוד כולל:

- קלט כתובת אימייל - חובה
- אותן המגבלות כמו בעמוד ההתחברות
- קלט סיסמא - חובה
- אותן המגבלות כמו בעמוד ההתחברות
- קלט שם מלא – חובה
- יכיל אותיות בשפה האנגלית בלבד
- כפתור הרשמה
- בלחיצה על כפתור ההרשמה תבוצע בדיקה על התנאים שהוגדרו עבור כל שורות הקלט.
- כמו כן תבוצע השוואה של כתובת האימייל למאגר הנתונים.

- במקרה של חריגה מהכללים תוצג בחלון קופץ ההודעה "please fill in all inputs with valid information." לסגירת החלון יש להקיש בחלל המסך.
- במקרה בו זיהתה המערכת כי כתובת המייל כבר קיימת תוצג בחלון קופץ ההודעה "Email existed already, go back to login or try again" לסגירת החלון יש להקיש בחלל המסך.
- לאחר הרשמה מוצלחת, יועבר המשתמש אל עמוד ההתחברות.
- במאגר המידע ישמרו הפרטים הבאים:
  - שם משתמש
  - כתובת אימייל
  - סיסמא
  - תאריך ההרשמה
  - כפתור התחברות
- המשתמש יועבר אל עמוד ההתחברות.

איור 23- עמוד הרשמה

איור 24- התראת מידע לא תקין בהרשמה

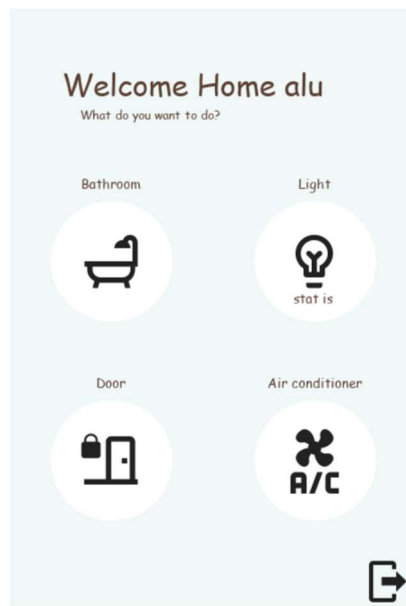


איור 25- התראת אימייל קיים

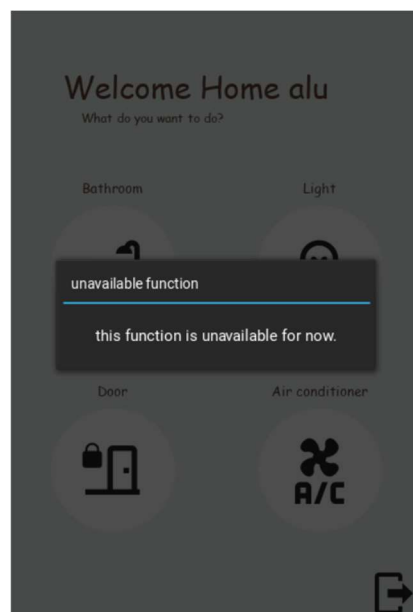
## עמוד "הבית"

מוצג רק כאשר המשתמש מחובר. העמוד כולל:

- טקסט "welcome home <current username>" עם שם המשתמש הנוכחי, והשאלה "what do you want to do?"
- כפתור "תאורה"
- הכפתור ממוקם בצידו הימני העליון של המסך.
- חיווי: על גבי הכפתור ישנם אייקון וכיתוב המציינים את מצב הנורה דלוק/ כבוי, המתעדכנים בהתאם למצב הנורה.
- בלחיצה על הכפתור נשלחת בקשה למתווך לשינוי מצב הנורה.
- כפתור "מקלחת"
- הכפתור ממוקם בצידו השמאלי העליון של המסך.
- בלחיצה על הכפתור יועבר המשתמש לעמוד "מקלחת".
- כפתור "AC" וכפתור "דלת"
- הכפתורים ממוקמים בחלקו התחתון של המסך.
- כפתורים אלו אינם זמינים למשתמש. מטרת הכפתורים היא להשלים את העיצוב של עמוד הבית.
- בלחיצה על כל כפתור תוצג בחלון קופץ ההודעה "this function is unavailable for now".
- לסגירת החלון יש להקיש בחלל המסך.
- כפתור "התנתקות"
- הכפתור ממוקם בפינה הימנית התחתונה של המסך.
- בלחיצה על הכפתור יועבר המשתמש לעמוד ההתחברות וינותק מהמערכת.



איור 26-עמוד הבית



איור 27- התראת פונקציה לא זמינה

### 3.4.3.3 עמוד "מקלחת"

בעמוד זה יוכל המשתמש לצפות בטמפרטורת דוד השמש, להפעיל ולנהל אותו. העמוד כולל:

- מד טמפרטורה מופיע בחלקו המרכזי העליון של המסך.
- ספינר חישוק המשתנה בהתאם לטמפרטורת הדוד באופן יחסי בטווח שבין 15-70 . במרכז הספינר מופיע ערך הטמפרטורה במעלות צלזיוס.
- מד זה מתעדכן על פי המידע שמתקבל מחיישן הטמפרטורה הנשלח כל 5 שניות.
- תיבת בחירה "מספר מקלחות" – לא חובה
- התיבה ממוקמת בצידו השמאלי התחתון של המסך.

- בלחיצה על התיבה תיפתח רשימה המכילה את האופציות הבאות לבחירה: "1, 2, more".
  - במידה והמשתמש מעוניין כי משך הדלקת הדוד יקבע על פי מספר המקלחות הרצוי, עליו לבחור באחת האפשרויות מהרשימה.
- כפתור "דוד שמש"
- הכפתור ממוקם בצידו הימני התחתון של המסך.
- חיווי: על גבי הכפתור ישנם אייקון וכיתוב המציינים את מצב הנורה דלוק/ כבוי. הכיתוב מתעדכן בהתאם למצב הנורה.
  - בלחיצה על הכפתור נשלחת בקשה למתווך לשינוי מצב הנורה הכוללת את המידע בתיבת הבחירה.
- כפתור "התנתקות"
- הכפתור ממוקם בפינה הימנית התחתונה של המסך.
- בלחיצה על הכפתור יועבר המשתמש לעמוד ההתחברות וינותק מהמערכת.
- כפתור "בית"
- הכפתור ממוקם משמאל לכפתור ה"התנתקות".
- בלחיצה על הכפתור יועבר המשתמש לעמוד הבית.



איור 28- עמוד "אמבטיה"



איור 29-עמוד "אמבטיה" רשימת בחירה

#### 3.4.4 קוד האפליקציה

שפת הפיתוח: Python

גרפיקת ממשק המשתמש: kivy

הקוד מורכב מ- 4 חלקים כך שכל חלק ממלא תפקיד בצורה בלתי תלויה.

ארבעת החלקים הם: Database האחראי על ניהול פרטי המשתמשים, צד לקוח המטפל בתקשורת עם השרת, ממשק המשתמש המנהל את הממשק וגרפיקת הממשק האחראית על העיצוב הוויזואלי של האפליקציה הגלויה למשתמש.

##### **data – Database 3.4.4.1**

ספריות: datetime

מחלקה: Database

המחלקה אחראית על ניהול פרטי המשתמשים.

הפרטים נשמרים בקובץ text.



שם הפעולה	תפקיד
Init (self, filename)	הפעולה בונה database ששמו הוא הפרמטר filename ומגדירה את המשתנים הבאים: תוכן המאגר וקובץ. הפעולה קוראת לפעולה load().
Load (self)	הפעולה פותחת את הקובץ בשם המשתנה filename לקריאה, מגדירה את תצורת הכתיבה של תוכן הקובץ באמצעות המשתנים הבאים: אימייל, סיסמא, שם, תאריך ולאחר מכן סוגרת את הקובץ.
get_user (self, email)	הפעולה מקבלת כפרמטר אימייל, במידה והאימייל קיים בקובץ הפעולה מחזירה את פרטי המשתמש במידה ולא הפעולה מחזירה 1-.
add_user (self, email, password, name)	הפעולה מקבלת כפרמטרים: אימייל, סיסמא ושם. במידה והאימייל קיים בקובץ הפעולה תחזיר 1- ותדפיס הודעה כי האימייל קיים. אחרת – הפעולה מעדכנת את פרטי המשתמש ואת התאריך וקוראת לפעולה save()
Validate (self, email, password)	הפעולה מקבלת כפרמטר אימייל וסיסמא. במידה והאימייל אינו קיים בקובץ או אינו תואם לסיסמא הפעולה מחזירה false, אחרת הפעולה מחזירה true.
save (self)	הפעולה פותחת את הקובץ לכתיבה ושומרת את פרטי המשתמש והתאריך בקובץ.
get_date ()	הפעולה מחזירה את התאריך הנוכחי.

### server - Mqtt\_p 3.4.4.2

מחלקה: Mqtt\_service

המחלקה אחראית על ניהול התקשורת בין ממשק המשתמש לשרת.

ספריות: paho, datetime, kivy

שם הפעולה	תפקיד
Init ()	הפעולה בונה mqtt_service ומגדירה עבורו Mqtt client . הפעולה קוראת לפעולה .mqtt_up()
on_message (self, mqttc, obj, msg)	פונקציית callback . מקבלת כפרמטרים: client נוכחי, obj, ו msg הכולל את ה- topic ואת תוכן ההודעה. הפעולה פונה לפעולות הבאות בהתאמה ל- topic שהתקבל ושולחת כפרמטר את תוכן ההודעה המפוענח: updateUI_light(self, status), updateUI_temp(self, temp), updateUI_boiler(self, status)
mqtt_up(self)	הפעולה מקשרת את הפעולה on_message לזו של client, שולחת בקשה להתחברות למתווך (broker) ונרשמת (subscribe) להאזנה לtopic הבאים: אישור עדכון התאורה, אישור עדכון דוד השמש וחיישן הטמפרטורה. בנוסף, קוראת לפעולה "update_request" על מנת לעדכן את נתוני המערכת.
publishMSG_light (self, msg)	הפעולה מקבלת כפרמטר הודעה, ומפרסמת (publish) למתווך (broker) הודעה לשנות את מצב הנורה.
publishMSG_boiler (self, msg)	הפעולה מקבלת כפרמטר הודעה, ומפרסמת (publish) למתווך (broker) הודעה לשנות את מצב הדוד.
updateUI_light (self, status)	הפעולה מקושרת לפעולה בשם זהה בקובץ main
updateUI_temp (self, temp),	הפעולה מקושרת לפעולה בשם זהה בקובץ main
updateUI_boiler (self, status)	הפעולה מקושרת לפעולה בשם זהה בקובץ main
Update_request (self)	הפעולה מפרסמת (publish) למתווך (broker) הודעה "update".

### Main 3.4.4.3

אחראי על ניהול ממשק המשתמש

ספריות: kivy, kivyMD

#### מחלקה: CreateAccountWindow

המחלקה אחראית על מימוש חלון הרישום לאפליקציה.

שם הפעולה	תפקיד
Submit (self)	הפעולה בודקת את תקינות פרטי המשתמש מוסיפה אותו למערכת ומעדכנת את מנהל החלונות למסך ההתחברות. במידה והפרטים אינם תקינים הפעולה מציגה הודעה על פי השגיאה.
Login (self)	הפעולה קוראת לפעולה reset() ומעדכנת את מנהל החלונות למסך ההתחברות.
reset ()	הפעולה מאתחלת את ערכי המשתנים הבאים: שם, סיסמא ואימייל.
emailExists ()	הפעולה פותחת חלון קופץ עם הודעת שגיאה עבור מייל קיים

**מחלקה: LoginWindow**

המחלקה אחראית על מימוש חלון ההתחברות לאפליקציה.

שם הפעולה	תפקיד
loginBtn (self)	הפעולה מעדכנת את מנהל החלונות למסך הבית. במידה ופרטי ההתחברות שגויים המסך לא יתעדכן והפעולה תקרא לפונקציה invalidLogin()
createBtn (self)	הפעולה קוראת לפעולה reset() ומעדכנת את מנהל החלונות למסך הרישום.
reset ()	הפעולה מאתחלת את ערכי המשתנים הבאים: סיסמא ואימייל.
emailExists ()	הפעולה פותחת חלון קופץ עם הודעת שגיאה עבור מייל קיים

**מחלקה: ControlWindow**

המחלקה אחראית על מימוש חלון הבית.

שם הפעולה	תפקיד
logOut (self)	הפעולה מעדכנת את מנהל החלונות למסך ההתחברות.
on_enter (self)	הפעולה שומרת את פרטי המשתמש הנוכחי במשתנים: סיסמא, שם ותאריך ומעדכנת את הודעת הברכה של העמוד בהתאם לשם.
reset ()	הפעולה מאתחלת את ערכי המשתנים הבאים: סיסמא ואימייל.
emailExists(self)	הפעולה פותחת חלון קופץ עם הודעת שגיאה עבור מייל קיים
light_pressed(self)	הפעולה קוראת לפעולה (status) light_update במחלקה EventDispatcher ושולחת כפרמטר את מצב האור הנוכחי (דלוק/כבוי).
Unavailable(self)	הפעולה פותחת חלון קופץ עם הודעת שגיאה עבור פונקציה לא קיימת.

**מחלקה: BathRoom**

המחלקה אחראית על מימוש חלון המקלחת.

שם הפעולה	תפקיד
boiler_pressed (self)	הפעולה קוראת לפעולה (status) boiler_update במחלקה EventDispatcher ושולחת כפרמטר את מצב הבویلר הנוכחי (דלוק/כבוי).

**מחלקה: WindowManager**

המחלקה מאפשרת את מנהל החלונות. אין תוכן.

**מחלקה: MyEventDispatcher**

המחלקה מאפשרת שמירה על פרטיות של המשתנים בממשק המשתמש ומחוצה לו ועדכון רק במחלקה ובקובץ הייעודים. הקובץ מאורגן כך שכל הבקשות היוצאות והבקשות הנכנסות מממשק המשתמש יעברו דרך מחלקה זו.

שם הפעולה	תפקיד
init (self)	הפעולה מגדירה את הevents הנדרשים.
light_update(self, status)	הפעולה מבצעת dispatch לפונקציה on_light ושולחת כמשתנה את הפרמטר status
on_light	הפעולה מקושרת לפונקציה my_callback_light (self, value, *args) במחלקה Main
lightUI_button (self, status, msg)	הפעולה מבצעת dispatch לפונקציה on_lightUI_button ושולחת כמשתנה את הפרמטר status
on_lightUI_button (self, *args)	הפעולה מעדכנת את האייקון והטקסט של כפתור הנורה במסך הבית בהתאם לפרמטר שהתקבל.
boiler_update(self, status)	הפעולה מבצעת dispatch לפונקציה on_boiler ושולחת כמשתנה את הפרמטר status
on_boiler	הפעולה מקושרת לפונקציה my_callback_boiler (self, value, *args) במחלקה Main
boilerUI_button (self, status)	הפעולה מבצעת dispatch לפונקציה on_boilerUI_button ושולחת כמשתנה את הפרמטר status
on_lightUI_button (self, *args)	הפעולה מעדכנת את הטקסט של כפתור הדוד במסך הבית בהתאם לפרמטר שהתקבל.
show_temp (self, temp)	הפעולה מבצעת dispatch לפונקציה on_show_temp (self, temp) ושולחת כמשתנה את הפרמטר temp.
on_show_temp (self, temp)	הפעולה מעדכנת את הפרמטרים של הטמפרטורה במסך המקלחת לפי הפרמטר שהתקבל.
fit_to_scale(self, value, minimum, maximum)	הפעולה מקבלת כפרמטרים את הטמפרטורה ואת ערכי הטמפרטורה המקסימלית והמינימלית ומחזירה את מיקום ערך הטמפרטורה על הספינר.

**מחלקה: MyMainApp**

המחלקה הראשית. אחראית על ניהול המחלקות השונות וגישור לקובץ mqtt\_p המתקשר עם הפרוטוקול.

שם הפעולה	תפקיד
build (self)	הפעולה מוסיפה את כל המסכים למנהל החלונות, יוצרת אובייקט מסוג Mqtt_service ומקשרת בין הפונקציות הרלוונטיות. הפעולה מחזירה את מנהל החלונות.
My_callback_light(self, value, *args)	הפעולה קוראת לפונקציה publishMsg_light (self, msg) בקובץ ה- mqtt_p ושולחת כפרמטר את הנתון שקיבלה [args[0].
updateUI_light (self, status, msg)	הפעולה מקבלת כפרמטר את מצב הנורה והודעה ומעדכנת את משתנה מצב הנורה בהתאם. הפעולה קוראת לפונקציה lightUI_button() במחלקה EventDispatcher ושולחת כפרמטר את status שקיבלה.
My_callback_boiler(self, value, *args)	הפעולה קוראת לפונקציה publishMsg_boiler (self, msg) בקובץ ה- mqtt_p ושולחת כפרמטר את הנתון שקיבלה [args[0].
updateUI_boiler (self, status, msg)	הפעולה מקבלת כפרמטר את מצב הדוד והודעה ומעדכנת את משתנה מצב הדוד בהתאם. הפעולה קוראת לפונקציה boilerUI_button() במחלקה EventDispatcher ושולחת כפרמטר את status שקיבלה.
updateUI_temp (self, temp)	הפעולה מקבלת כפרמטר את הטמפרטורה וקוראת לפונקציה show_temp(temp) במחלקה EventDispatcher ושולחת כפרמטר את הטמפרטורה שקיבלה.

#### פעולות כלליות:

שם הפעולה	תפקיד
invalidateLogin()	הפעולה פותחת חלון קופץ עם הודעת שגיאה עבור פרטי משתמש
invalidateForm()	הפעולה פותחת חלון קופץ עם הודעת שגיאה עבור אי מילוי פרטים הכרחיים.

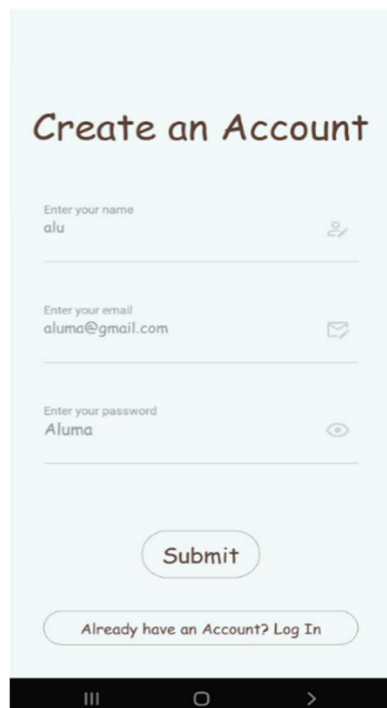
#### Kv 3.4.4.4

קובץ kv מאפשר את הפרדת גרפיקת ממשק המשתמש מהממשק עצמו. את קטעי הקוד ניתן למצוא בנספחים.

## 4 בדיקות המערכת

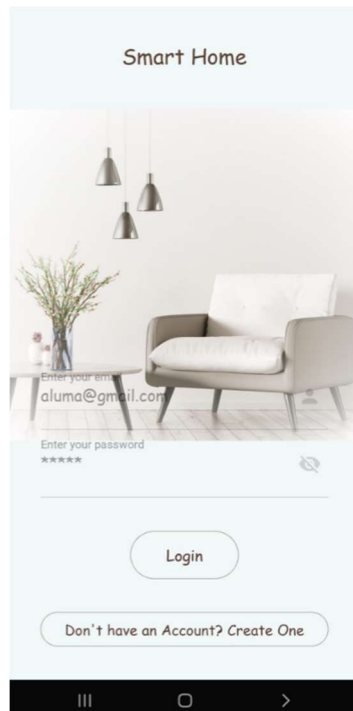
הבדיקות נעשו בסביבה הביתית, כאשר רשת האינטרנט עליה פועלת האפליקציה (הן במחשב והן בפלאפון נייד) שונה מן הרשת עליה פועלות היחידות השונות, על מנת לאפשר בדיקה מהימנה של הפעלת המערכת מרחוק. היחידות במערכת מיתוג האור והן במערכת מיתוג הדוד מורכבות על גבי אותה מטריצה, אך הן מנותקות לחלוטין ואין כל תקשורת קווית כל או מקור מתח משותף. בנוסף בדיקות הטמפרטורה מבוצעות עבור טמפרטורות נמוכות (סביבות טמפ' החדר) על מנת לבדוק את תקינות החיישן (ולא עבור טמפרטורות גבוהות יותר המתאימות לדוד חשמלי), ובדיקות הזמנים מבוצעות גם הם בזמנים קצרים (שניות בודדות) על מנת לאפשר בחינה מהירה של המערכת. כל זאת לא סותר את יכולת המערכת להתמודד עם זמנים ממושכים, וטמפרטורות גבוהות.

### כניסה לאפליקציה והרשמה



איור 30- הרשמה לאפליקציה

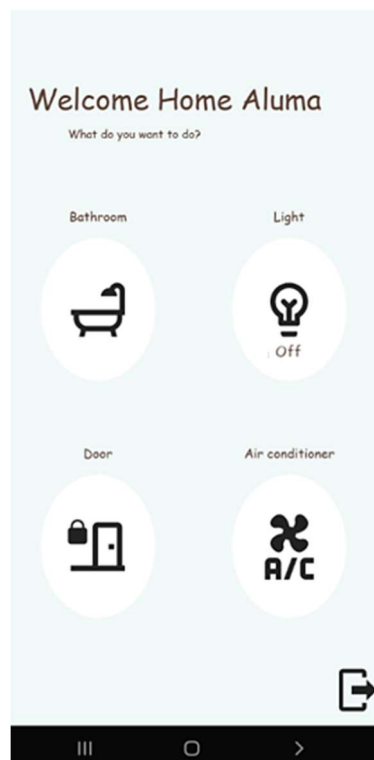
תחילה הוכנסו הפרטים למערכת ובלחיצה על כפתור ה"submit" ההרשמה התבצעה בהצלחה ומסך הפתיחה הופיע כנדרש.



איור 31- מסך הפתיחה

בהכנסת פרטי המשתמש ובלחיצה על כפתור ה – "login" הופיע מסך הבית כנדרש. ניתן להסיק מכך כי פרטי המשתמש נשמרו בהצלחה.

מעבר לעמוד הבית של האפליקציה ובדיקת תקינות מערכת האור



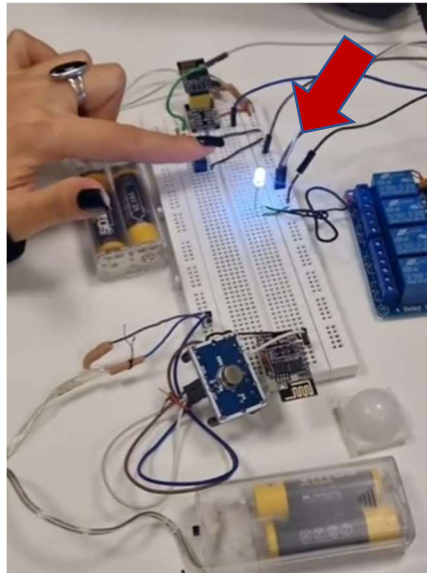
איור 32- מצב חיווי נורת הלד "off" בעקבות פתיחת האפליקציה



ניתן לראות כי בפתיחת מסך הבית חיווי מצב מתג האור התעדכן בהתאם למצב הנורה (כעת כבויה) "Off".

#### 4.1 מיתוג מערכת התאורה על ידי המתג הידני

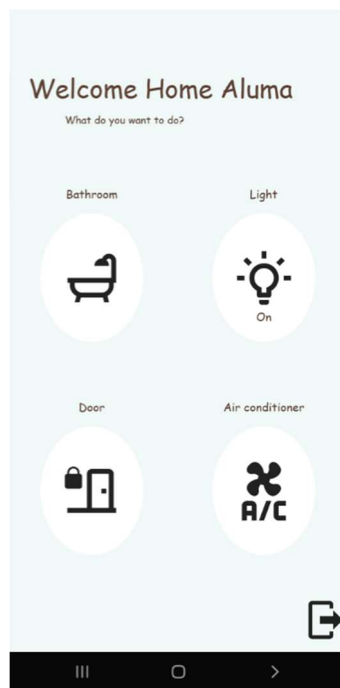
בשלב זה תיבדק יכולת מיתוג האור על ידי המתג הידני המורכב במעגל מיתוג האור. ראשית, נלחץ על המתג הידני המחובר במעגל המיתוג:



איור 33- לחיצה על המתג במערכת התאורה

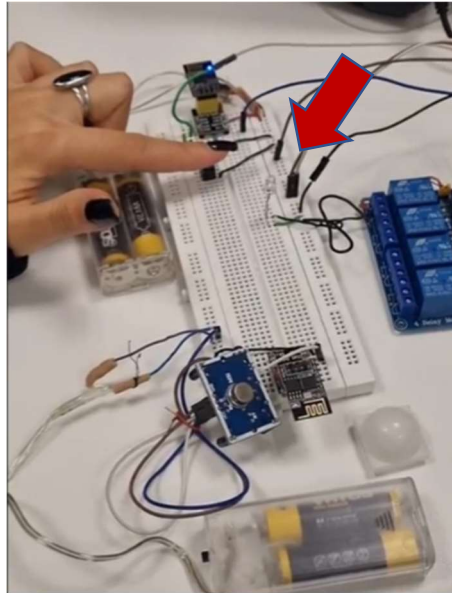
ניתן לראות כי בעת לחיצה על המתג הידני במעגל החשמלי הופעל הממסר ונדלקה מנורת הלד המחוברת אליו.

במקביל, ניתן לראות באפליקציה כי לחצן חיווי מצב נורת האור השתנה ל"On".



איור 34- מסך הבית, מצב הנורה "On"

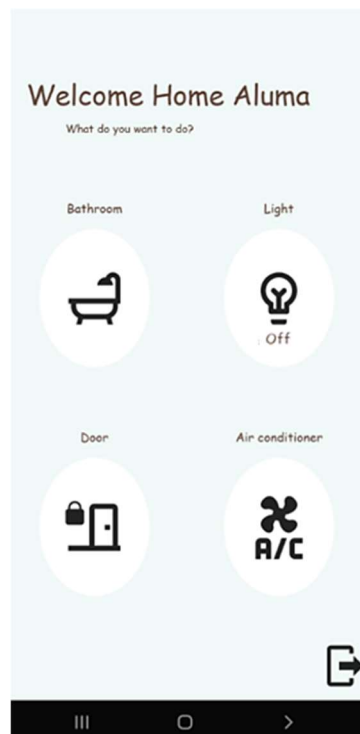
נלחץ שוב על המתג הידני במעגל מיתוג האור:



איור 35 - לחיצה נוספת על המתג במעגל מיתוג האור

ניתן לראות כי בעת לחיצה חוזרת על המתג הידני במעגל זה הופעל הממסר ונכבתה נורת הלד המחוברת אליו.

במקביל, ניתן לראות באפליקציה כי חיווי מצב נורת האור השתנה ל"Off".

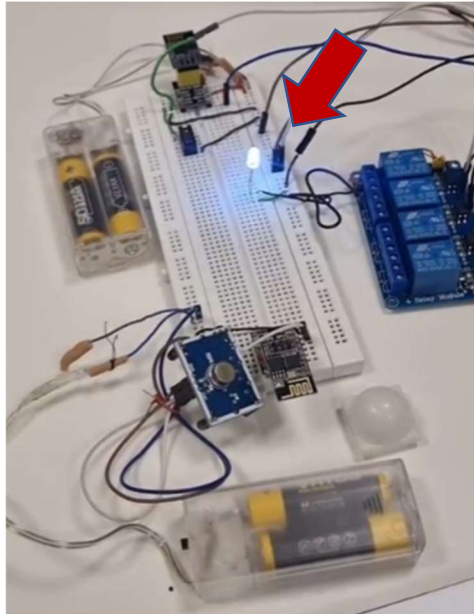


איור 36- מצב חיווי נורת הלד "off" בעקבות לחיצה חוזרת

## 4.2 מיתוג מערכת התאורה על ידי חיישן התנועה

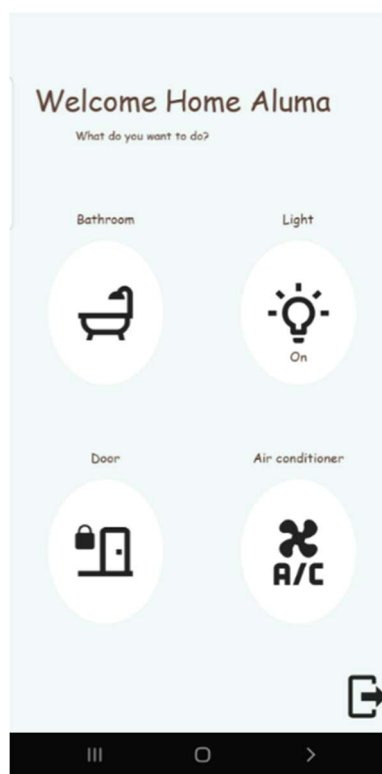
בשלב זה תיבדק יכולת מיתוג האור על ידי חיישן התנועה.

ראשית, נעביר חפץ באזור חיישן התנועה כאשר נורת הלד כבויה:



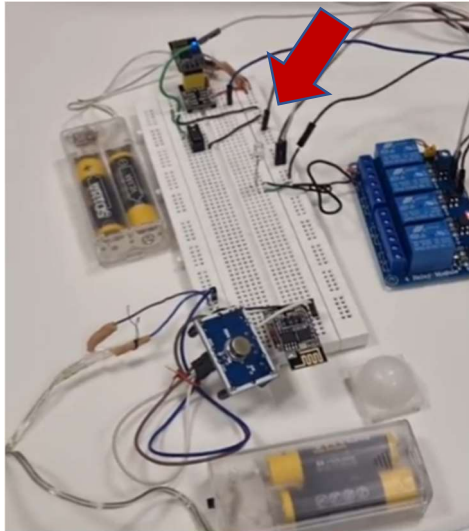
איור 37- נורת הLED דלוקה לאחר העברת חפץ באזור חיישן התנועה

ניתן לראות באיור כי החיישן זיהה את התנועה והנורה נדלקה כמצופה. במקביל, ניתן לראות כי בעקבות כך לחצן חיווי מצב נורת האור באפליקציה השתנה ל"On".



איור 38- מסך הבית, מצב הLED "On"

לאחר מכן נבדק תזמון הכיבוי האוטומטי כאשר לא התבצעה תנועה באזור חיישן התנועה

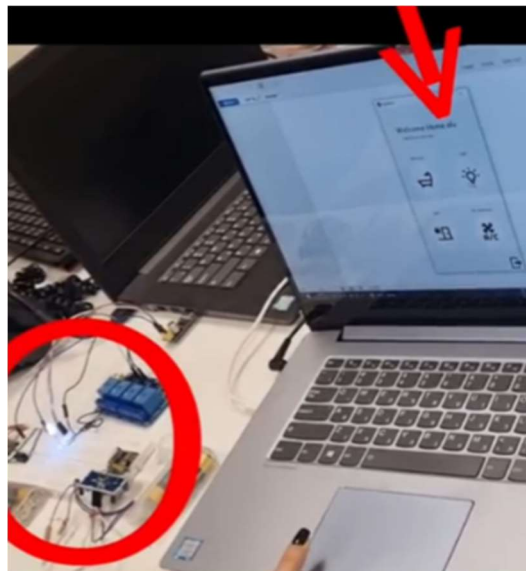


איור 39- מערכת התאורה לאחר 8 שניות ללא תנועה באזור

נורת הLED כבתה לאחר 8 שניות (כפי שהוגדר כברירת המחדל) בהם לא זוהתה תנועה באזור חיישן התנועה. במקביל, חיווי לחצן מצב נורת האור באפליקציה השתנה ל "Off".

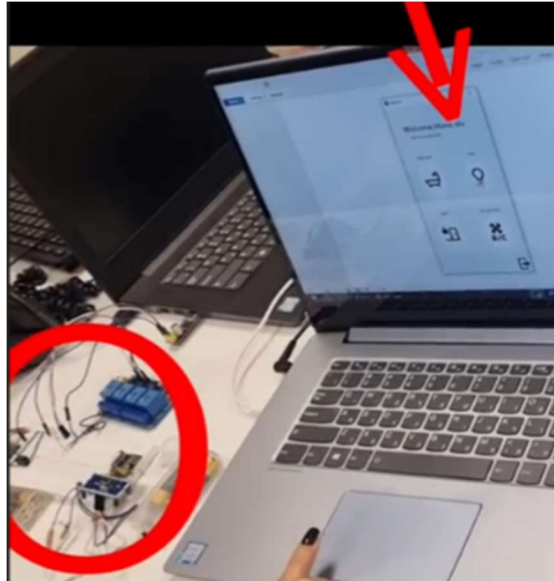
### 4.3 מיתוג מערכת התאורה על ידי האפליקציה

בשלב האחרון נבדקה יכולת מיתוג האור בעת הקשה על לחצן האור באפליקציה. המצב הנכחי: Off, בעת לחיצה על הלחצן:



איור 40- הקשה על לחצן האור באפליקציה הפתוחה במחשב

ניתן לראות כי בעת הקשה על לחצן האור באפליקציה נורת הLED נדלקה, וכעת מופיע האיור המתאר נורה דלוקה והחיווי "On".  
בלחיצה נוספת:



איור 41- הקשה על לחצן האור באפליקציה הפתוחה במחשב

ניתן לראות כי בעת הקשה על לחצן האור באפליקציה נורת הLED כבתה, וחיווי מצב הנורה באפליקציה השתנה בהתאם וכעת מופיע האיור המתאר נורה כבויה והחיווי "Off".

מעבר לעמוד "אמבטיה" ובדיקת תקינות מערכת הדוד:

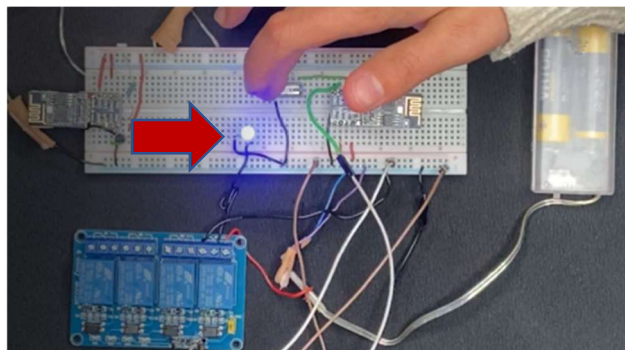


איור 42- עמוד "אמבטיה"

ניתן לראות כי בפתיחת מסך ה"אמבטיה" חיווי מצב מתג הדוד התעדכן בהתאם למצב הדוד (כעת כבוי) "Off", וכי הטמפרטורה התעדכנה גם כן.

#### 4.4 מיתוג מערכת הדוד על ידי המתג הידני

בשלב זה תיבדק יכולת מיתוג דוד החשמל על ידי המתג הידני המורכב במעגל מיתוג הדוד. ראשית, נלחץ על המתג הידני המחובר למעגל המיתוג כאשר הנורה כבויה:



איור 43-לאחר לחיצה על המתג הידני הנורה דלוקה

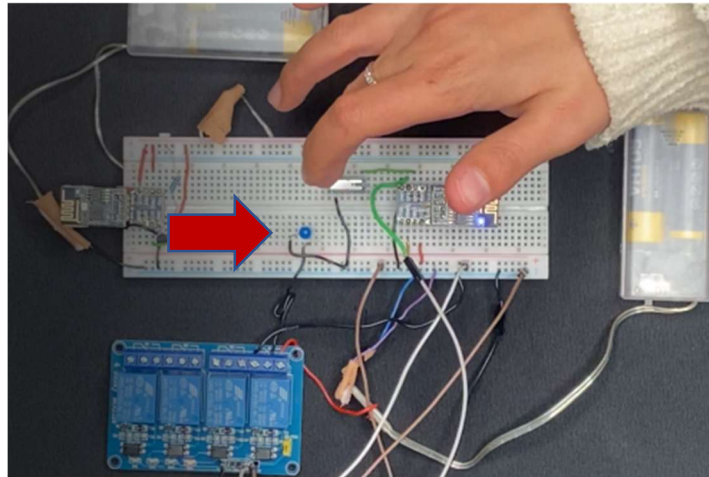
ניתן לראות כי בעת לחיצה על המתג הידני במעגל החשמלי הופעל הממסר ונדלקה מנורת הלד המחוברת אליו המייצגת את הפעלת הדוד החשמלי.

במקביל, ניתן לראות באפליקציה כי חיווי מצב הדוד השתנה ל"On".



איור 44- לחצן חיווי דוד "on"

נלחץ שוב על המתג הידני במעגל מיתוג הדוד:



איור 45- לאחר לחיצה נוספת על המתג הידני, הנורה כבויה

ניתן לראות כי בעת לחיצה חוזרת על המתג הידני במעגל זה הופעל הממסר ונכבתה נורת הלד המחוברת אליו.

במקביל, ניתן לראות באפליקציה כי חיווי מצב הדוד השתנה ל"Off".



איור 46- לחצן חיווי דוד "off"

#### 4.5 מיתוג מערכת הדוד על ידי האפליקציה

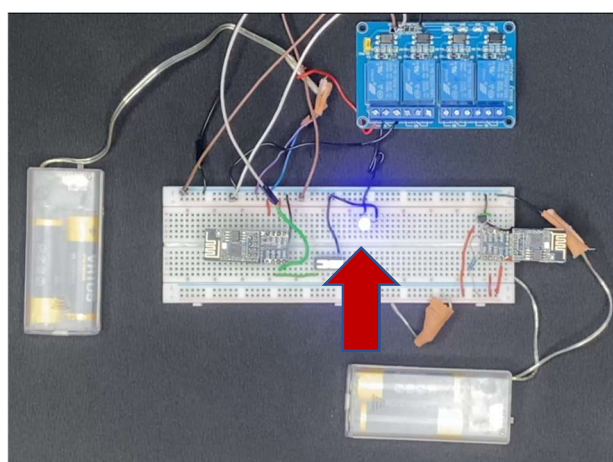
בשלב זה נבדקה יכולת מיתוג הדוד באמצעות הקשה על לחצן הדוד באפליקציה. האפליקציה מציעה למשתמש מספר אפשרויות למשך הדלקת הדוד באמצעות ספינר בחירה עבור מספר המתקלחים. לפני לחיצה על מתג הדוד ניתן לבחור אחת מהאפשרויות או לא לבחור כלל.

בלחיצה על לחצן הדוד ללא בחירת "מספר המקלחות":



איור 47- לחצן חיווי דוד "on"

ניתן לראות באפליקציה כי חיווי מצב מתג הדוד השתנה ל"On".



איור 48- נורת הלד דלוקה לאחר לחיצה באפליקציה

במקביל, ניתן לראות באיור כי בעת ההקשה על לחצן הדוד באפליקציה הממסר הופעל והנורה המייצג את מצב מתג הדוד נדלקה.

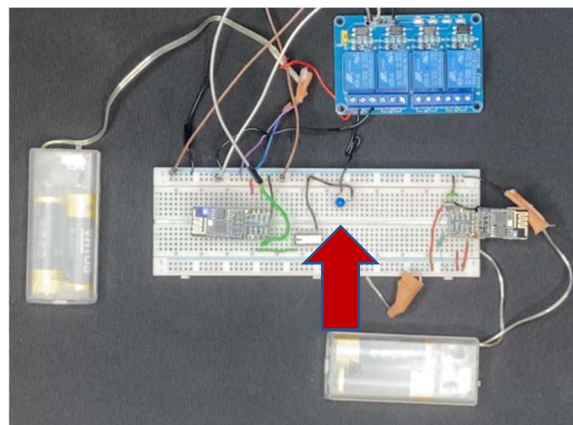
כעבור 6 שניות כבתה הנורה המייצגת את מצב המתג.





איור 49- לחצן חיווי דוד "off" כעבור 6 שניות

ניתן לראות באיור כי לחצן חיווי הדוד "off".



איור 50- נורת הלד כבוייה לאחר 6 שניות

ניתן לראות באיור כי הנורה כבתה.  
 המערכת דלקה למשך 6 שניות. זמן זה תואם לזמן שהוגדר למערכת כברירת מחדל כאשר אינה מקבלת נתון עבור "מספר המקלחות".  
 לאחר מכן נבדקו התוצאות עבור האפשרויות השונות בספינר הבחירה. על מנת להשתמש באפשרויות אלה, נבחרה אופציה מסוימת ולאחר מכן נלחץ לחצן ה"Boiler".  
 בעת לחיצה על הלחצן הופעל הממסר ונדלקה נורת החיווי. עבור כל אפשרות הנורה כבתה לאחר פרק זמן שונה.  
 התוצאות מובאות בטבלה הבאה:

טבלה 10 - משך הזמן עד לכיבוי הדוד כתלות בבחירת מספר המקלחות

משך הזמן עד לכיבוי (Second)	בחירת אפשרות ברירה "number of showers"
2	1
4	2
6	more

ניתן לראות כי עבור מקלחת אחת, משך הזמן בו הדוד דלוק הוא 2 שניות, עבור שתי מקלחות, 4 שניות ועבור 3 מקלחות ויותר משך הזמן הוא 6 שניות. משך הזמן בו הדוד פועל נגזר מתוך ערך קבוע, שהוגדר ביחידת המיתוג, העומד על 2s ומוכפל בנתון המתקבל מהאפליקציה. עבור אפשרות הבחירה "more" מוכפל הערך ב3. על פי הגדרה זו ניתן לראות כי הנתונים שהתקבלו בבדיקה תואמים למצופה מהמערכת.

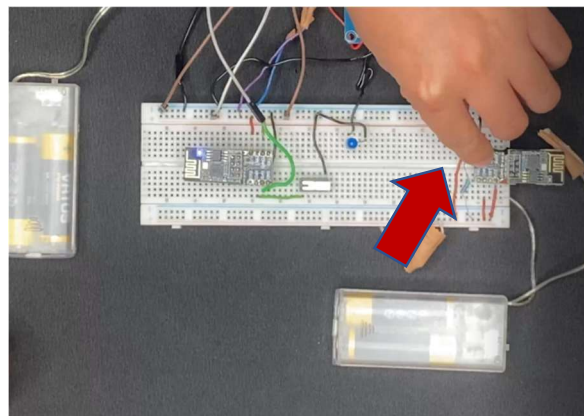
#### 4.6 בדיקת תקינות יחידת חיישן הטמפרטורה

ניתן לראות באיורים הקודמים כי האפליקציה מעודכנת לטמפרטורה הנוכחית העומדת על 24 מעלות. הנוסחה עבור נרמול הטמפרטורה לציר הגלגל כפי שמתוארת בקוד האפליקציה הינה:

$$\frac{\text{value} - \text{minimum}}{\text{maximum} - \text{minimum}} * 300 - 150$$

בהתבסס על כך, ניתן לראות כי ציר ה- "גלגל" הנע בטווח בין מינימום של 15 למקסימום של 70 מעלות מעודכן באופן הגיוני על פי הטמפרטורה.

נקרר את חיישן הטמפרטורה באמצעות מגע בקרח



איור 51-קירור חיישן הטמפרטורה

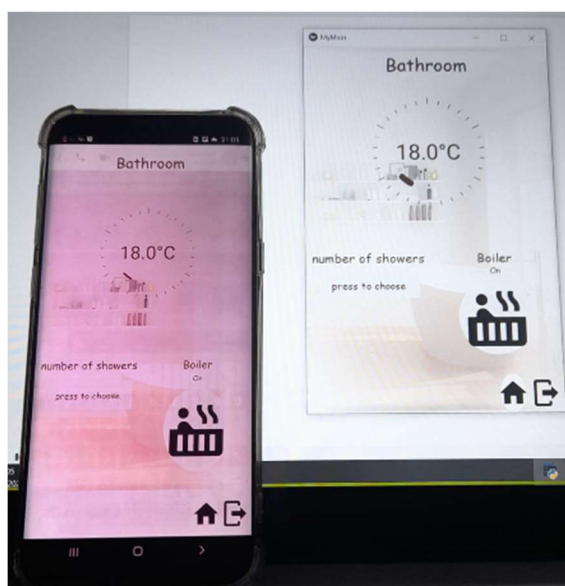


איור 52- חיווי טמפרטורה 18 מעלות

ניתן לראות כי בעקבות מגע של החיישן במקור קר, ירדה הטמפרטורה המופיעה באפליקציה ל 18 מעלות. מכאן ניתן להסיק כי חיישן הטמפרטורה תקין, וכי המידע מיחידת חיישן הטמפרטורה עובר באופן תקין לאפליקציה. בנוסף ניתן לראות כי ציר ה- "גלגל" התעדכן בהתאם.

#### 4.7 בדיקת סנכרון בין מספר מכשירים

בדיקה זו נועדה לבדוק את יכולת הסנכרון של האפליקציה בעדכון מספר מכשירים ממוחשבים שונים. על מנת לבצע בדיקה זו נפתח את האפליקציה במחשב האישי ובמקביל בפלאפון הנייד. כפי שנכתב בתחילת הפרק, האפליקציה הפתוחה במחשב משתמשת ברשת ה-Wi-Fi המקומית ואילו האפליקציה הפתוחה בפלאפון הנייד משתמשת ב"נתונים ניידים" של המכשיר כך שאין שום קשר ישיר בין המכשירים.



איור 53- מסכי האפליקציה במחשב האישי ובפלאפון הנייד

ניתן לראות בתמונה כי האפליקציה הפתוחה במחשב האישי והאפליקציה הפתוחה בפלאפון הנייד מציגות נתונים זהים עבור הטמפרטורה (18 מעלות) ומצב הדוד ("On"). הסנכרון נבדק באופן דומה עבור הפעלת מתג הדוד/ מתג האור מכל אחת מהאפליקציות והן בלחיצה על המתגים הפיזיים ובכולן התקבלו מצגים זהים.

## 5 תקלות ופתרון

### 1. הכנת הבקר לצריבת התוכנית

הרכיב esp8266-01 הינו רכיב בסיסי ועל מנת להכניס את ההרכיב למוד צריבה יש לבצע את השלבים הבאים :

1. ניתוק ה- ESP מאספקת המתח
2. חיבור gpio0 לGND
3. חיבור ה- ESP לאספקת המתח
4. ניתוק gpio0 מה- GND

**2. WDT reset (Watchdog Timer reset)** – טיימר, שכאשר אינו מאופס לפני שפג תוקפו, מפעיל את איפוס המערכת המנטרת (במקרה זה המערכת הינה מיקרו בקר esp8266-01).

הreset מתבצע כאשר המערכת מזהה שגיאת חומרה או תוכנה.

התקלה: כאשר חוברו שני מעגלים חשמליים ביחד למקור המתח יחיד- מתאם USB למחשב, מיד הרכיב איפס את עצמו בloop, וביחד עם הנתונים של פעולת הreset, הופיע בתיבת serial screen הודעת שגיאה WDT reset.

הפתרון: לאחר אבחון התקלה, למידה עליה מהרשת וביצוע בדיקות נוספות לחומרה, הוחלט להפריד את מקור המתח של שני המעגלים ולספק לכל מעגל מקור מתח אישי. באמצעות פעולה זו, הזרם לרכיבים הגיע בעצמה הנדרשת והמעגלים פעלו כראוי.

### 3. תקלה: מגעים בחומרה

פתרון: החלפת מטריצה שמסיבה מסוימת הייתה מקוצרת לכל אורכה, בדיקות מעבר מתח לרכיבים שבוצעה בהם הלחמה, בדיקת חוטי הקצר.

### 4. תקלה: עדכונים ובקשות של ממשק המשתמש באופן עקיף ולא ישיר מהממשק.

פתרון: שימוש בEvent Dispatcher – מספק ממשק עקבי לרישום ומניפולציה של events.

## 6 תוצאות ומסקנות

- לאור הבדיקות שנעשו, המערכת עובדת באופן תקין ורציף בסביבה הביתית ומחוצה לה. כל חלקיה מתעדכנים כנדרש בעת שינוי במצב המתגים, המידע מהחיישנים מתקבל כנדרש ובקשות המשתמש מהאפליקציה נשלחות גם כאשר האפליקציה מחוברת לרשת אינטרנט שונה.
- שימוש ברכיב esp8266-01 מאפשר ניהול זול, יעיל ואמין של מערכת פשוטה ומסוגל להתמודד באמצעות פרוטוקולים גם עם רכיבים אנלוגיים כגון חיישן טמפרטורה.
- שימוש בפרוטוקול ה-IP מאפשר לאפליקציה להתחבר מכל רשת אינטרנט למתווך וכך מתקבלת אפשרות שליטה מכל מקום למערכת הביתית.
- על מנת לדבג באופן יעיל יותר מערכות משובצות חומרה ואפליקציה המבוססות על פרוטוקול MQTT ניתן להשתמש באפליקציה MQTT- lens. האפליקציה "מבקשת" את הנתונים הרלוונטיים (כתובת IP ומספר הפורט), מתחברת למתווך וניתן לשלוח ולקבל דרכה מידע לפי topics הרלוונטיים. באמצעות כך, ניתן לראות האם המידע נשלח/ מתקבל ולדייק את מיקום התקלה חומרה/ תוכנה.
- שימוש בפרוטוקול MQTT מאפשר סנכרון נכון ויעיל של מספר יחידות ממוחשבות.
- תכנון יחידת המתג החכם כיחידה קומפקטית ופשוטה מאפשרת את חיבורה הנח למתג ביתי קיים.
- קביעת מקור המתח של החיישנים כבטריות מאפשר ליחידות גמישות בהחלטה עבור מיקומם הנכון בסביבה הביתית.
- הפרטת המערכת הגדולה כך שכל יחידה אחראית על נושא מסוים מספקת עצמאות לכל יחידה, איתור תקלות ממוקד ויעיל ומאפשרת שינויים ועדכונים בצורה יסודית, מדויקת ומינימלית המצמצמת את האפשרות לשגיאות.
- עלות הפרויקט עומדת על פחות מ-100 שקלים.

רכיב	מחיר שוקלים	מחיר סופי
ESP8266-01	4	16
ESP8266-Programmer	1	3
DS18B20	1	3
HC-SR501	1	3
נגדים	100 בשקית	2
חוטי קצר	4 צבעים	3
מטריצה	2	14
בתי סוללות	4	4
סוללות	8	24
מתג	2	4
ממסרים	2	5
ESP8266-Adapter	1	4
לדים	2	2
מחיר סופי		87

איור 54-חישוב עלות הפרויקט

## 7 סיכום

פרויקט זה עוסק בפיתוח מערכת אוטומציה ביתית זולה, קלה להתקנה ונגישה למשתמש. באמצעות מתגים חכמים, חיישנים ואפליקציית אנדרואיד מקבל המשתמש את האפשרות לשלוט על יחידת התאורה ויחידת הדוד בביתו באופן פעיל או סביל מכל מקום ובכל זמן. המערכת מאפשרת סנכרון ועדכון בזמן אמת של המתגים, החיישנים, ואפליקציית האנדרואיד באופן רציף גם כאשר נמצאת בשימוש אצל מספר משתמשים בו זמנית.

## 8 מקורות ספרותיים

1. רוזנבוים, ע'. גונן, ב'. והוד ש. (2020) 'רשתות מחשבים'. ישראל: המרכז לחינוך סייבר.  
[/http://www.cyber.org.il](http://www.cyber.org.il)
2. [/https://www.instructables.com/Getting-Started-With-the-ESP8266-ESP-01](https://www.instructables.com/Getting-Started-With-the-ESP8266-ESP-01)
3. [/https://kivy.org](https://kivy.org)
4. <https://microchipdeveloper.com/8bit:wdt>
5. [/https://test.mosquitto.org](https://test.mosquitto.org)

## 9 רשימת איורים

- איור 1- מודל ה- OSI..... 10
- איור 2 מודל ה TCP/IP בהשוואה למודל ה- OSI..... 10
- איור 3- פורמט הודעה MQTT..... 11
- איור 4- דיאגרמת זמנים של קו התקשורת בין מאסטר לעבד..... 13
- איור 5- סכמת בלוקים..... 14
- איור 6-מימוש פרוטוקול ה- MQTT במערכת..... 15
- איור 7- רכיבים במערכת התאורה..... 16
- איור 8- סימולציה יחידת חיישן תנועה..... 17
- איור 9 - מימוש מעגל יחידת חיישן התנועה..... 17
- איור 10- תרשים זרימה עבור יחידת חיישן התנועה..... 18
- איור 11- סימולציית יחידת מיתוג האור..... 19
- איור 12- מימוש מעגל יחידת מיתוג האור..... 19
- איור 13- תרשים זרימה יחידת מיתוג האור..... 20
- איור 14- דיאגרמת מלבנים של דוד שמש..... 21
- איור 15-סימולצית רכיב ESP8266-01 וחיישן טמפרטורה..... 22
- איור 16- מימוש מעגל יחידת חיישן הטמפרטורה..... 22
- איור 17-תרשים זרימה יחידת חיישן טמפרטורה..... 23
- איור 18- סימולצית רכיב ESP8266-01, הממסר והמתג..... 24
- איור 19- מימוש מעגל יחידת מיתוג הדוד..... 24
- איור 20- תרשים זרימה יחידת מיתוג דוד השמש..... 25
- איור 21- עמוד התחברות..... 27
- איור 22- התראה על מידע לא תקין..... 27
- איור 23- עמוד הרשמה..... 28
- איור 24- התראת מידע לא תקין בהרשמה..... 28
- איור 25- התראת אימייל קיים..... 29
- איור 26-עמוד הבית..... 30

30.....	איור 27- התראת פונקציה לא זמינה.....
31.....	איור 28- עמוד "אמבטיה".....
32.....	איור 29-עמוד "אמבטיה" רשימת בחירה .....
39.....	איור 30- הרשמה לאפליקציה.....
40.....	איור 31- מסך הפתיחה.....
40.....	איור 32- מצב חיווי נורת הלבד "off" בעקבות פתיחת האפליקציה .....
41.....	איור 33- לחיצה על המתג במערכת התאורה .....
41.....	איור 34- מסך הבית, מצב הנורה "On".....
42.....	איור 35 - לחיצה נוספת על המתג במעגל מיתוג האור.....
42.....	איור 36- מצב חיווי נורת הלבד "off" בעקבות לחיצה חוזרת .....
43.....	איור 37- נורת הלבד דלוקה לאחר העברת חפץ באזור חיישן התנועה .....
43.....	איור 38- מסך הבית, מצב הלבד "On".....
44.....	איור 39- מערכת התאורה לאחר 8 שניות ללא תנועה באזור .....
44.....	איור 40- הקשה על לחצן האור באפליקציה הפתוחה במחשב .....
45.....	איור 41- הקשה על לחצן האור באפליקציה הפתוחה במחשב .....
45.....	איור 42- עמוד "אמבטיה".....
46.....	איור 43-לאחר לחיצה על המתג הידני הנורה דלוקה .....
46.....	איור 44- לחצן חיווי דוד "on".....
47.....	איור 45- לאחר לחיצה נוספת על המתג הידני, הנורה כבויה .....
47.....	איור 46- לחצן חיווי דוד "off".....
48.....	איור 47- לחצן חיווי דוד "on".....
48.....	איור 48- נורת הלבד דלוקה לאחר לחיצה באפליקציה .....
49.....	איור 49- לחצן חיווי דוד "off" כעבור 6 שניות.....
49.....	איור 50- נורת הלבד כבויה לאחר 6 שניות.....
50.....	איור 51-קירור חיישן הטמפרטורה.....
51.....	איור 52- חיווי טמפרטורה 18 מעלות.....
51.....	איור 53- מסכי האפליקציה במחשב האישי ובפלאפון הנייד.....
53.....	איור 54-חישוב עלות הפרויקט .....
87.....	איור 55- סכמת בלוקים של החיישן בטמפרטורה.....
88.....	איור 56- רגלי הרכיב DS18B20 .....
89.....	איור 57- סכמת המעגל. HC-SR501.....
89.....	איור 58- כניסות ויציאות הרכיב HC-SR501 .....
90.....	איור 59- רגלי הממסר .....
91.....	איור 60 -כניסות ויציאות הרכיב ESP8266-01 .....



## 10 רשימת טבלאות

33.....	טבלה 1- ממשק מחלקה Database
33.....	טבלה 2- ממשק מחלקה Mqtt_p
35.....	טבלה 3- ממשק מחלקה CreateAccountWindow
35.....	טבלה 4 - ממשק מחלקה LoginWindow
36.....	טבלה 5- ממשק מחלקה ControlWindow
36.....	טבלה 6- ממשק מחלקה BathRoom
37.....	טבלה 7- ממשק מחלקה MyEventDispatcher
38.....	טבלה 8- ממשק מחלקה MyMainApp
38.....	טבלה 9- פעולות כלליות
50.....	טבלה 10 - משך הזמן עד לכיבוי הדוד כתלות בבחירת מספר המקלחות

## 11 נספחים

### 11.1 קוד עבור הבקרים

יחידת חיישן התנועה

[code]

```
// Define pins for LEDs
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
#define MQTT_SERVER "91.121.93.94"
const char* ssid = "Aluma";
const char* password = "aluma1234";
// Callback function header
void callback(char* topic, byte* payload, unsigned int length);
// millis
unsigned long currentMillis;
unsigned long startMillis;
const unsigned long period= 8000; // how much time?
bool isOn=false;
bool isMove= false;
//motion sensor status
int detectedLED ;
// Input from Motion Sensor on ESP8266 GPIO2
const int pirPin = 2;
```

```

// Input from Light Sensor on ESP8266 GPIO0
const int ldrPin = 0;
// Variable to store value from PIR
int pirValue;
// Variable to store value from PIR
int ldrValue;
//topic to publish to for controlling the other ESP module
const char* lightTopic = "/house/light1";
//topic to subscribe to the light
const char* lightConfirmTopic = "/house/light1confirm";
WiFiClient wifiClient;
PubSubClient client(MQTT_SERVER, 1883, callback, wifiClient);
void setup() {
    // initial start time
    startMillis=millis();
    // Setup PIR as Input
    pinMode(pirPin, INPUT);
    // Setup LDR as Input
    pinMode(ldrPin, INPUT);
    // Initial 1 Minute Delay to stabilize sensor
    digitalWrite(detectedLED, LOW);
    Serial.println("wait");
    // delay(600);
    Serial.println("ready");
    //start the serial line for debugging
    Serial.begin(115200);
    delay(100);
    //start wifi subsystem
    WiFi.begin(ssid, password);
    //attempt to connect to the WIFI network and then connect to the MQTT server
    reconnect();
    //wait a bit before starting the main loop
    delay(200);
}
void loop() {
    //reconnect if connection is lost
    if (!client.connected() && WiFi.status() == 3) {reconnect();}
    //maintain MQTT connection
    client.loop();
}

```

```

    checkMotion();
    //MUST delay to allow ESP8266 WIFI functions to run
    delay(10);
}
//MQTT callback
void callback(char* topic, byte* payload, unsigned int length) {
    //convert topic to string to make it easier to work with
    String topicStr = topic;
    //Print out some debugging info
    Serial.println("Callback update.");
    Serial.print("Topic: ");
    Serial.println(topicStr);
    payload[length] = 0;
    String recv_payload = String((char *) payload);
    //switch is turn on
    if(recv_payload == "On"){
        isOn=true;
    }
    //switch is turn off
    else if (recv_payload == "Off"){
        isOn=false;
        isMove= false;
    }
}
void checkMotion(){
    // Get value from Light sensor
    ldrValue = digitalRead(ldrPin);
    if (ldrValue == 0){

        // Get value from motion sensor
        pirValue = digitalRead(pirPin);

        // See if motion Detected for the first time or the user turn the switch off
        if (pirValue == 1){
            isMove=true;
            if (isOn==false) {
                client.publish(lightTopic, "On"); //publish movment to the broker,
            }
        }
        if (pirValue == 0) {

```

```

    currentMillis = millis(); //get the current "time" (the number of milliseconds since the
program started)
    if (currentMillis - startMillis >= period){ //test whether the period has elapsed
        if(isMove==false){
            //publish no movment to the broker,
            client.publish(lightTopic, "Off");
        }
        startMillis = currentMillis; //IMPORTANT to save the start time of the current LED state.
    }
}
}
}
}

//networking functions
void reconnect() {
    //attempt to connect to the wifi if connection is lost
    if(WiFi.status() != WL_CONNECTED){
        //debug printing
        Serial.print("Connecting to ");
        Serial.println(ssid);
        //loop while we wait for connection
        while (WiFi.status() != WL_CONNECTED) {
            delay(500);
            Serial.print(".");
        }
        //print out some more debug once connected
        Serial.println("");
        Serial.println("WiFi connected");
        Serial.println("IP address: ");
        Serial.println(WiFi.localIP());
    }
    //make sure we are connected to WIFI before attempting to reconnect to MQTT
    if(WiFi.status() == WL_CONNECTED){
        // Loop until we're reconnected to the MQTT server
        while (!client.connected()) {
            Serial.print("Attempting MQTT connection...");
            // Generate client name based on MAC address and last 8 bits of microsecond counter
            String clientName;
            clientName += "esp8266-";
            uint8_t mac[6];
            WiFi.macAddress(mac);

```

```

    clientName += macToStr(mac);
    //if connected, subscribe to the topic(s) we want to be notified about
    if (client.connect((char*) clientName.c_str())) {
        Serial.print("\tMTQQ Connected");
        client.subscribe(lightConfirmTopic);
    }
    //otherwise print failed for debugging
    else{Serial.println("\tFailed."); abort();}
}
}
}
//generate unique name from MAC addr
String macToStr(const uint8_t* mac){
    String result;
    for (int i = 0; i < 6; ++i) {
        result += String(mac[i], 16);
        if (i < 5){
            result += ':';
        }
    }
    return result;
}
[/code]

```

יחידת מיתוג האור

```

[code]
//ESP8266 Simple MQTT light controller
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
#include <Bounce2.h>
//LED on ESP8266 GPIO2
const int lightPin = 2;
//button on ESP8266 GPIO0
const int buttonPin = 0;
//topic to subscribe to the light
const char* lightTopic = "/house/light1";
//topic to publish to confirm that the light has been turned on for the python script to log
const char* lightConfirmTopic = "/house/light1confirm";
//create an instance of the bounce class
Bounce myButton = Bounce();

```

```

//EDIT THESE LINES TO MATCH YOUR SETUP
#define MQTT_SERVER "91.121.93.94"
const char* ssid = "Hodaya";
const char* password = "12345678";
static boolean isOn = false; //static var to keep track of the intended current boiler state
// Callback function header
void callback(char* topic, byte* payload, unsigned int length);
WiFiClient wifiClient;
PubSubClient client(MQTT_SERVER, 1883, callback, wifiClient);
void setup() {
    //initialize the button pin as an input
    pinMode(buttonPin, INPUT);
    myButton.attach(buttonPin);
    myButton.interval(5);
    //initialize the light as an output and set to LOW (off)
    pinMode(lightPin, OUTPUT);
    digitalWrite(lightPin, LOW);
    //start the serial line for debugging
    Serial.begin(115200);
    delay(100);
    //start wifi subsystem
    WiFi.begin(ssid, password);
    //attempt to connect to the WIFI network and then connect to the MQTT server
    reconnect();
    //wait a bit before starting the main loop
    delay(200);
}
void loop(){
    //reconnect if connection is lost
    if (!client.connected() && WiFi.status() == 3) {reconnect();}
    //maintain MQTT connection
    client.loop();
    //monitor the button
    checkButton();
    //MUST delay to allow ESP8266 WIFI functions to run
    delay(10);
}
//MQTT callback
void callback(char* topic, byte* payload, unsigned int length) {

```

```

//convert topic to string to make it easier to work with
String topicStr = topic;
payload[length] = 0;
String recv_payload = String(( char *) payload);
//Print out some debugging info
Serial.println("Callback update.");
Serial.print("Topic: ");
Serial.println(topicStr);
if(topicStr.equals(lightTopic)){
  if (recv_payload == "update"){
    if (isOn == true)
    {
      client.publish(lightConfirmTopic, "On");
    }
    else
    {
      client.publish(lightConfirmTopic, "Off");
    }
  }
  else
  {
    lightSwitchUpdate();
  }
}
}

void checkButton(){
  //static boolean isOn = false; //static var to keep track of the intended current light state
  if(myButton.update() && myButton.read() == HIGH){
    //update the button and check for HIGH or LOW state
    lightSwitchUpdate();
  }
}

void lightSwitchUpdate(){
  //on false, the light is off so tell it to turn on and set the internal var to true,
  if(isOn == false){
    digitalWrite(lightPin, HIGH);
    client.publish(lightConfirmTopic, "On"); //publish to the confirmation topic so the python
script can log it
    Serial.println("light on");
    isOn = true;
  }
}

```

```

    }
    //else (on true), the light is on so tell it to turn off and set the internal var to false
    else{
        digitalWrite(lightPin, LOW);
        client.publish(lightConfirmTopic, "Off"); //publish to the confirmation topic so the python
script can log it
        Serial.println("light off");
        isOn = false;
    }
}

//networking functions
void reconnect() {
    //attempt to connect to the wifi if connection is lost
    if(WiFi.status() != WL_CONNECTED){
        //debug printing
        Serial.print("Connecting to ");
        Serial.println(ssid);
        //loop while we wait for connection
        while (WiFi.status() != WL_CONNECTED) {
            delay(500);
            Serial.print(".");
        }
        //print out some more debug once connected
        Serial.println("");
        Serial.println("WiFi connected");
        Serial.println("IP address: ");
        Serial.println(WiFi.localIP());
    }

    //make sure we are connected to WIFI before attempting to reconnect to MQTT
    if(WiFi.status() == WL_CONNECTED){
        // Loop until we're reconnected to the MQTT server
        while (!client.connected()) {
            Serial.print("Attempting MQTT connection...");
            // Generate client name based on MAC address and last 8 bits of microsecond counter
            String clientName;
            clientName += "esp8266-";
            uint8_t mac[6];
            WiFi.macAddress(mac);
            clientName += macToStr(mac);

```



```

//if connected, subscribe to the topic(s) we want to be notified about
if (client.connect((char*) clientName.c_str())) {
    Serial.print("\tMTQQ Connected");
    client.subscribe(lightTopic);
}
//otherwise print failed for debugging
else{Serial.println("\tFailed."); abort();}
}
}
}
//generate unique name from MAC addr
String macToStr(const uint8_t* mac){
    String result;
    for (int i = 0; i < 6; ++i) {
        result += String(mac[i], 16);
        if (i < 5){
            result += ':';
        }
    }
    return result;
}
[/code]

```

## יחידת חיישן הטמפרטורה

```

//ESP8266 MQTT temp sensor node
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
//create 1-wire connection on pin 2 and connect it to the dallasTemp library
OneWire oneWire(2);
DallasTemperature sensors(&oneWire);
//EDIT THESE LINES TO MATCH YOUR SETUP
#define MQTT_SERVER "91.121.93.94"
const char* ssid = "maya";
const char* password = "0545446444";
//topic to publish to for the temperature
char* tempTopic = "/house/temp_sensor";
char currentTemp[2];
float previousTempFloat;

```

```

// Callback function header
void callback(char* topic, byte* payload, unsigned int length);
WiFiClient wifiClient;
PubSubClient client(MQTT_SERVER, 1883, callback, wifiClient);
void setup() {
    //null terminate the temp string to be published
    currentTemp[1] = '\0';
    previousTempFloat= 0;
    //start the serial line for debugging
    Serial.begin(115200);
    delay(100);
    //start wifi subsystem
    WiFi.begin(ssid, password);
    //attempt to connect to the WIFI network and then connect to the MQTT server
    reconnect();
    //start the temperature sensors
    sensors.begin();
    //wait a bit before starting the main loop
    delay(2000);
}
void loop(){
    // Send the command to update temperatures
    sensors.requestTemperatures();
    //get the new temperature
    float currentTempFloat = sensors.getTempCByIndex(0);
    currentTemp[0] = currentTempFloat;
    //if (abs(currentTempFloat - previousTempFloat) > 1 ){
    //publish the new temperature
    client.publish(tempTopic, currentTemp);
    previousTempFloat= currentTempFloat;
    Serial.println("tmp is:");
    Serial.println(currentTempFloat);
    delay(5000);
    // }
    //reconnect if connection is lost
    if (!client.connected() && WiFi.status() == 3) {reconnect();}
    //maintain MQTT connection
    client.loop();
    //MUST delay to allow ESP8266 WIFI functions to run
    delay(5000);
}

```

```

}
//MQTT callback
void callback(char* topic, byte* payload, unsigned int length) {}
//networking functions
void reconnect() {
    //attempt to connect to the wifi if connection is lost
    if(WiFi.status() != WL_CONNECTED){
        Serial.print("Connecting to ");
        Serial.println(ssid);
        //loop while we wait for connection
        while (WiFi.status() != WL_CONNECTED) {
            delay(500);
            Serial.print(".");
        }
    }
    //make sure we are connected to WIFI before attempting to reconnect to MQTT
    if(WiFi.status() == WL_CONNECTED){
        // Loop until we're reconnected to the MQTT server
        while (!client.connected()) {
            Serial.print("Attempting MQTT connection...");
            // Generate client name based on MAC address and last 8 bits of microsecond counter
            String clientName;
            clientName += "esp8266-";
            uint8_t mac[6];
            WiFi.macAddress(mac);
            clientName += macToStr(mac);
            //if connected, subscribe to the topic(s) we want to be notified about
            if (client.connect((char*) clientName.c_str())) {
                Serial.print("\tMTQQ Connected");
                //subscribe to topics here
            }
            //otherwise print failed for debugging
            else{Serial.println("\tFailed."); abort();}
        }
    }
}
//generate unique name from MAC addr
String macToStr(const uint8_t* mac){

    String result;

```

```

for (int i = 0; i < 6; ++i) {
    result += String(mac[i], 16);
    if (i < 5){
        result += ':';
    }
}
return result;
}

```

## יחידת מיתוג הדוד

```

[code]
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
#include <Bounce2.h>
//LED pin for temperature monitor
const int boilerPin = 2;
//button on ESP8266 GPIO0
const int buttonPin = 0;
//topic to subscribe to for the temperature
char* tempTopic = "/house/temp_sensor";
//topic to publish to confirm that the boiler has been turned on for the python script to log
char* boilerConfirmTopic = "/house/boiler1confirm";
char* boilerTopic = "/house/boiler";
//create an instance of the bounce class
Bounce myButton = Bounce();
//EDIT THESE LINES TO MATCH YOUR SETUP
#define MQTT_SERVER "91.121.93.94"
const char* ssid = "Hodaya";
const char* password = "12345678";
char headerByte;
String topicStr;
char byteToSend;
float byteToSend_float;
int numOfShowers;
static boolean isOn = false; //static var to keep track of the intended current boiler state
// millis
unsigned long currentMillis;
unsigned long startMillis;
const unsigned long period= 2000; // how much time?

```

```

// Callback function header
void callback(char* topic, byte* payload, unsigned int length);
WiFiClient wifiClient;
PubSubClient client(MQTT_SERVER, 1883, callback, wifiClient);
void setup() {
  // initial start time
  startMillis=millis();
  //initialize the button pin as an input
  pinMode(buttonPin, INPUT);
  myButton.attach(buttonPin);
  myButton.interval(5);
  //setup pin as output
  pinMode(boilerPin, OUTPUT);
  digitalWrite(boilerPin, LOW);
  //start the serial line for debugging
  Serial.begin(115200);
  delay(100);
  //start wifi subsystem
  WiFi.begin(ssid, password);
  //attempt to connect to the WIFI network and then connect to the MQTT server
  reconnect();
  //wait a bit before starting the main loop
  delay(2000);
}
void loop(){
  //reconnect if connection is lost
  if (!client.connected() && WiFi.status() == 3) {reconnect();}
  //maintain MQTT connection
  client.loop();
  //monitor the button
  checkButton();
  timer();
  //MUST delay to allow ESP8266 WIFI functions to run
  delay(10);
}
//MQTT callback
void callback(char* topic, byte* payload, unsigned int length) {
  //convert topic to string to make it easier to work with
  topicStr = topic;
  byteToSend = 0;

```

```

numOfShowers=3;
//handle tempTopic updates
if(topicStr.equals(tempTopic)){
    byteToSend = char(*payload); //set byte to send to the payload
    byteToSend_float= byteToSend;
    Serial.println(byteToSend_float);
}
if(topicStr.equals(boilerTopic)){
    byteToSend = char(*payload); //set byte to send to the payload
    byteToSend_float= byteToSend;
    payload[length] = 0;
    String recv_payload = String(( char *) payload);
    Serial.println(recv_payload);
    if (recv_payload == "update") {
        if (isOn == true) {
            client.publish(boilerConfirmTopic,"On");
            Serial.println("publish on");
        }
        else {
            client.publish(boilerConfirmTopic,"Off");
            Serial.println("publish off");
        }
    }
    else {
        if (recv_payload == "1"){
            numOfShowers = 1;
            Serial.println(numOfShowers);
        }
        else if (recv_payload == "2"){
            numOfShowers = 2;
            Serial.println(numOfShowers);
        }
        else if (recv_payload == "more"){
            numOfShowers = 3;
            Serial.println(numOfShowers);
        }
    }
    boilerSwitchUpdate();
}
}
}

```

```

void checkButton(){

    if(myButton.update() && myButton.read() == HIGH){ //update the button and check for HIGH
or LOW state
        Serial.println("button pressed");
        numOfShowers=3;
        boilerSwitchUpdate();
    }
}

void timer(){
    if (isOn == true) {
        currentMillis = millis(); //get the current "time" (the number of milliseconds since the
program started)
        if (currentMillis - startMillis >= period*numOfShowers){ //test whether the period has
elapsed
            Serial.print("the boiler was on for: ");
            Serial.println(currentMillis - startMillis);
            startMillis = millis();
            boilerSwitchUpdate();
        }
    }
}

void boilerSwitchUpdate(){
    //on false, the boiler is off so tell it to turn on and set the internal var to true
    if(isOn == false){
        digitalWrite(boilerPin, HIGH);
        client.publish(boilerConfirmTopic,"On");
        startMillis = millis(); //IMPORTANT to save the start time of the current LED state.
        isOn = true;
    }
    //else (on true), the light is on so tell it to turn off and set the internal var to false
    else{
        digitalWrite(boilerPin, LOW);
        client.publish(boilerConfirmTopic,"Off");
        isOn = false;
    }
}

//networking functions
void reconnect() {

```

```

//attempt to connect to the wifi if connection is lost
if(WiFi.status() != WL_CONNECTED){
  //loop while we wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  //debug printing
  Serial.print("Connecting to ");
  Serial.println(ssid);
}
//make sure we are connected to WIFI before attempting to reconnect to MQTT
if(WiFi.status() == WL_CONNECTED){
  // Loop until we're reconnected to the MQTT server
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Generate client name based on MAC address and last 8 bits of microsecond counter
    String clientName;
    clientName += "esp8266-";
    uint8_t mac[6];
    WiFi.macAddress(mac);
    clientName += macToStr(mac);
    //if connected, subscribe to the topic(s) we want to be notified about
    if (client.connect((char*) clientName.c_str())) {
      Serial.print("\tMTQQ Connected");
      client.subscribe(boilerTopic);
      client.subscribe(tempTopic);
    }
    //otherwise print failed for debugging
    else{Serial.println("\tFailed."); abort();}
  }
}
}

//generate unique name from MAC addr
String macToStr(const uint8_t* mac){
  String result;
  for (int i = 0; i < 6; ++i) {
    result += String(mac[i], 16);
    if (i < 5){

```



```

        result += ':';
    }
}
return result;
}
[/code]

```

## 11.2 קוד עבור האפליקציה

Database.py

```

import datetime

class DataBase:
    def __init__(self, filename):
        self.filename = filename
        self.users = None
        self.file = None
        self.load()

    def load(self):
        self.file = open(self.filename, "r")
        self.users = {}

        for line in self.file:
            email, password, name, created = line.strip().split(";")
            self.users[email] = (password, name, created)

        self.file.close()

    def get_user(self, email):
        if email in self.users:
            return self.users[email]
        else:
            return -1

    def add_user(self, email, password, name):
        if email.strip() not in self.users:
            self.users[email.strip()] = (password.strip(), name.strip(), DataBase.get_date())
            self.save()
            return 1
        else:
            print("Email exists already")
            return -1

    def validate(self, email, password):
        if self.get_user(email) != -1:
            return self.users[email][0] == password
        else:
            return False

    def save(self):
        with open(self.filename, "w") as f:
            for user in self.users:
                f.write(user + ";" + self.users[user][0] + ";" + self.users[user][1] + ";" +
self.users[user][2] + "\n")

    @staticmethod

```

```
def get_date():
    return str(datetime.datetime.now()).split(" ")[0]
```

Main.py

```
from kivy.app import App
from kivymd.app import MDApp
from kivy.lang import Builder
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.properties import NumericProperty, ObjectProperty, StringProperty
from kivy.uix.popup import Popup
from kivy.uix.label import Label
from database import DataBase
from kivy.core.window import Window
from kivymd.uix.textfield import MDTextField
from kivymd.uix.button import MDRoundFlatButton
from kivymd.uix.button import MDIconButton
from kivy.event import EventDispatcher
from mqtt_p import Mqtt_service
```

```
class MyEventDispatcher(EventDispatcher):
```

```
    def __init__(self, **kwargs):
        # self.register_event_type('on_loginBtn')
        self.register_event_type('on_light')
        self.register_event_type('on_lightUI_button')
        self.register_event_type('on_boiler')
        self.register_event_type('on_show_temp')
        self.register_event_type('on_boilerUI_button')
        super(MyEventDispatcher, self).__init__(**kwargs)

    def light_update(self, status):
        # when light_status is called, the 'on_light' event will be
        # dispatched with the value
        self.dispatch('on_light', status)

    def on_light(self, *args):
        print("message sent to the broker", args[0])

    def lightUI_button(self, status, msg):
        self.dispatch('on_lightUI_button', status)

    def on_lightUI_button(self, *args):
        print("update sent to the ui", args)
        if args[0] == "On":
            App.get_running_app().root.screens[2].ids.light_text.text = args[0]
            App.get_running_app().root.screens[2].ids.light_icon.icon = "lightbulb-on-outline"

            if args[0] == "Off":
                App.get_running_app().root.screens[2].ids.light_text.text = args[0]
                App.get_running_app().root.screens[2].ids.light_icon.icon = "lightbulb-outline"

    def boiler_update(self, status):
        self.dispatch('on_boiler', status)

    def on_boiler(self, *args):
        print("message sent to the broker", args)
```

```

def boilerUI_button(self, status):
    self.dispatch('on_boilerUI_button', status)

def on_boilerUI_button(self, *args):
    print("update sent to the ui", args)
    if args[0] == "On":
        App.get_running_app().root.screens[3].ids.boiler_text.text = args[0]

    if args[0] == "Off":
        App.get_running_app().root.screens[3].ids.boiler_text.text = args[0]

def show_temp(self, temp):
    self.dispatch('on_show_temp', temp)

def on_show_temp(self, temp):
    App.get_running_app().root.screens[3].val = str(temp)
    App.get_running_app().root.screens[3].scale = self.fit_to_scale(temp, 15, 70)

def fit_to_scale(self, value, minimum, maximum):
    # scales the given value between the set max and min limits
    return ((value - minimum) / (maximum - minimum)) * 300 - 150

class CreateAccountWindow(Screen):
    namee = ObjectProperty(None)
    email = ObjectProperty(None)
    password = ObjectProperty(None)

    def submit(self):
        if self.namee.text != "" and self.email.text != "" and self.email.text.count(
            "@" ) == 1 and self.email.text.count(".") > 0:
            if self.password.text != "":
                if db.add_user(self.email.text, self.password.text, self.namee.text) == 1:
                    self.reset()
                    sm.current = "login"
                else:
                    self.emailExists()
            else:
                invalidForm()
        else:
            invalidForm()

    def login(self):
        self.reset()
        sm.current = "login"

    def reset(self):
        self.email.text = ""
        self.password.text = ""
        self.namee.text = ""

    def emailExists(self):
        pop = Popup(title='Email exists',
                    content=Label(text='Email exists already,\n go back to login or try again'),
                    size_hint=(None, None), size=(250, 250))
        pop.open()

class LoginWindow(Screen):
    email = ObjectProperty(None)
    password = ObjectProperty(None)

```

```

def loginBtn(self):
    if db.validate(self.email.text, self.password.text):
        ControlWindow.current = self.email.text
        self.reset()
        sm.current = "control"
    else:
        invalidLogin()
        # ev.loginBtn()

def createBtn(self):
    self.reset()
    sm.current = "create"

def reset(self):
    self.email.text = "alu@gmail.com"
    self.password.text = "1234"

class ControlWindow(Screen):
    n = ObjectProperty(None)
    created = ObjectProperty(None)
    email = ObjectProperty(None)
    light_text = ObjectProperty(None)
    light_icon = ObjectProperty(None)
    current = ""

    def logOut(self):
        sm.current = "login"

    def on_enter(self, *args):
        password, name, created = db.get_user(self.current)
        self.n.text = "Welcome Home " + name

    def light_pressed(self): # handle publish

        if MyMainApp.light_s == "On":
            ev.light_update("On")

        if MyMainApp.light_s == "Off":
            ev.light_update("Off")

    def unavailable(self):
        pop = Popup(title='unavailable function',
                    content=Label(text='this function is unavailable for now.'),
                    size_hint=(0.8, 0.2))

        pop.open()

class BathRoom(Screen):
    numOfShowers = ObjectProperty(None)
    boiler_text = ObjectProperty(None)
    boiler_icon = ObjectProperty(None)
    scale = NumericProperty(150)
    val = StringProperty('--')
    unit = StringProperty("°C")

    def boiler_pressed(self):
        if MyMainApp.boiler_s == "On":
            ev.boiler_update('On')

```

```

        if MyMainApp.boiler_s == "Off":
            ev.boiler_update("Off")

class WindowManager(ScreenManager):
    pass

def invalidLogin():
    pop = Popup(title='Invalid Login',
                content=Label(text='Invalid username or password.'),
                size_hint=(0.8, 0.2))
    pop.open()

def invalidForm():
    pop = Popup(title='Invalid Form',
                content=Label(text='Please fill in all inputs \nwith valid information.'),
                size_hint=(0.8, 0.2))

    pop.open()

kv = Builder.load_file("my.kv")
sm = WindowManager()
db = DataBase("users.txt")
ev = MyEventDispatcher()

class MyMainApp(MDApp):
    service = None
    light_s = "Off"
    boiler_s = "Off"

    def build(self):
        # self.theme_cls.theme_style = "Dark"
        self.theme_cls.primary_palette = "Brown"
        self.theme_cls.primary_hue = "700"
        screens = [LoginWindow(name="login"), CreateAccountWindow(name="create"),
                    ControlWindow(name="control"), BathRoom(name="bathroom")]

        for screen in screens:
            sm.add_widget(screen)

        self.service = Mqtt_service()
        # ev.bind(on_loginBtn=self.connection)
        ev.bind(on_light=self.my_callback_light)
        ev.bind(on_boiler=self.my_callback_boiler)
        # self.service.connection_failed = self.connection_failed
        self.service.updateUI_light = self.updateUI_light
        self.service.updateUI_temp = self.updateUI_temp
        self.service.updateUI_boiler = self.updateUI_boiler

        sm.current = "login"

    return sm

def my_callback_light(self, value, *args):
    self.service.publishMsg_light(args[0])

def updateUI_light(self, status, msg):
    self.light_s = status
    ev.lightUI_button(status, msg)
    print("Main: " + msg)

```

```

def my_callback_boiler(self, value, *args):
    num = self.root.screens[3].ids.numOfShowers.text
    self.service.publishMsg_boiler(num, args[0])

def updateUI_boiler(self, status):
    self.boiler_s = status
    ev.boilerUI_button(status)
    print(status)

def updateUI_temp(self, temp):
    ev.show_temp(temp)
    print(temp)

#ad can

if __name__ == "__main__":
    MyMainApp().run()

```

Mqtt\_p.py

בנוסף למפורט בפרויקט, הקוד הבא נותן אינדיקציה עבור משך זמן בו האור היה דלוק ושומר את הנתונים בקובץ.

```

import sys
import os
import paho.mqtt.client as mqtt
import string
import datetime
import time
import logging
# import kivy module
import kivy

kivy.require("1.9.1")
import kivy.uix.button as kb
from kivy.app import App
from kivy.uix.widget import Widget
from kivy.clock import mainthread
import struct

# keeps track of when we last turned the light on
onStartTime = 0

#####

# Create and set up the logging subsystem
logger = None

logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)

# create a file handler
timeFormat = "%a %b %d %Y %H.%M.%S"
today = datetime.datetime.today()
timestamp = today.strftime(timeFormat)
logFile = r'logs/logs' + timestamp + '.log'

```

```

handler = logging.FileHandler(logFile)
handler.setLevel(logging.INFO)

# create a logging format
formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
handler.setFormatter(formatter)

# add the handlers to the logger
logger.addHandler(handler)

class Mqtt_service(App):

    def __init__(self, **kwargs):
        # create our MQTT client
        self.mqttc = mqtt.Client()
        self.mqtt_up()

    def on_message(self, mqttc, obj, msg):
        # define our global vars for logger and the start time tracker
        global onStartTime
        global logger

        # get the local time in an easy to read format
        localtime = time.asctime(time.localtime(time.time()))

        if msg.topic == "/house/light1confirm":
            print(f"{msg.topic} - {msg.payload.decode('UTF-8')}")
            # to do if the message said that we turned the light On
            if msg.payload.decode('UTF-8') == "On":
                # take note of when we turned the light on
                onStartTime = time.time()
                # log the light on time and print
                logMessage = "Light turned on at: " + localtime
                self.updateUI_light(msg.payload.decode('UTF-8'), logMessage)
                print(logMessage)
                logger.info(logMessage)

            # to do if the message said that we turned the light Off
            if msg.payload.decode('UTF-8') == "Off":
                # take note of the total run time
                runTime = time.time() - onStartTime

                # log & print when the light turned off
                logMessage = "Light turned off at: " + localtime
                self.updateUI_light(msg.payload.decode('UTF-8'), logMessage)
                logger.info(logMessage)

                # log & print the total time the light was on for
                logMessage = "The light was on for a total of " + str(int(runTime)) + " seconds"
                self.updateUI_light(msg.payload.decode('UTF-8'), logMessage)
                logger.info(logMessage)

        if msg.topic == "/house/temp_sensor":
            temp = float(int.from_bytes(msg.payload, byteorder='big'))
            print(temp)
            self.updateUI_temp(temp)

```

```

if msg.topic == "/house/boiler1confirm":
    print(f'{msg.topic} - {msg.payload.decode('UTF-8')}')
    self.updateUI_boiler(msg.payload.decode('UTF-8'))
    print(msg.payload.decode('UTF-8'))

def mqtt_up(self):
    # tell it what to do when we receive a message, bind callback function
    self.mqttc.on_message = self.on_message

    # connect to the broker
    self.mqttc.connect("91.121.93.94", 1883, 60)
    # start the MQTT client loop in a separate thread
    self.mqttc.loop_start()
    # subscribe to our topics
    self.mqttc.subscribe("/house/light1confirm", 0)
    self.mqttc.subscribe("/house/temp_sensor", 0)
    self.mqttc.subscribe("/house/boiler1confirm", 0)

    self.update_request()

def update_request(self):
    msg = "update"
    self.mqttc.publish("/house/light1", msg.encode())
    self.mqttc.publish("/house/boiler", msg.encode())

def publishMsg_light(self, msg):
    self.mqttc.publish("/house/light1", msg.encode())
    print("ok")

def publishMsg_boiler(self, num, msg):
    self.mqttc.publish("/house/boiler", num.encode())

def updateUI_light(self, status, msg):
    pass

def updateUI_boiler(self, status):
    pass

def updateUI_temp(self, temp):
    pass

```

Main.kv

```

<CreateAccountWindow>:
    name: "create"

    namee: namee
    email: email
    password: password

    FloatLayout:
        cols:1
        canvas.before:
            Color:
                rgba: (238/255,248/255,247/255,0.8)
            Rectangle:
                pos: self.pos
                size: self.size

```



Label:  
 text: "Create an Account"  
 size\_hint: 0.3, 0.15  
 pos\_hint: {"center\_x":0.5, "top": 0.9}  
 font\_size: self.width/3  
 font\_name: "Comic"  
 color: (92/255,64/255,51/255,1)

MDTextField:  
 id: namee  
 hint\_text: 'Enter your name'  
 font\_size: self.width/20  
 font\_name: "Comic"  
 multiline: False  
 pos\_hint: {"center\_x":0.5 , "y":0.6}  
 size\_hint: 0.8, 0.12  
 icon\_right: "account-edit-outline"

MDTextField:  
 id: email  
 hint\_text: 'Enter your email'  
 font\_size: self.width/20  
 font\_name: "Comic"  
 multiline: False  
 pos\_hint: {"center\_x":0.5 , "y":0.45}  
 size\_hint: 0.8, 0.12  
 icon\_right: "email-edit-outline"

MDTextField:  
 id: password  
 hint\_text: 'Enter your password'  
 font\_size: self.width/18  
 font\_name: "Comic"  
 multiline: False  
 pos\_hint: {"center\_x":0.5 , "y":0.3}  
 size\_hint: 0.8, 0.12  
 icon\_right: "eye-outline"

MDRoundFlatButton:  
 pos\_hint:{"center\_x":0.5, "y":0.15}  
 size\_hint: 0.3, 0.07  
 text: "Submit"  
 font\_size: self.width/5  
 font\_name: "Comic"  
 on\_release:  
   root.manager.transition.direction = "left"  
   root.submit()

MDRoundFlatButton:  
 pos\_hint:{"center\_x":0.5,"y":0.05}  
 size\_hint: 0.8, 0.05  
 font\_size: self.width/20  
 font\_name: "Comic"  
 text: "Already have an Account? Log In"  
 on\_release:  
   root.manager.transition.direction = "left"

```

        root.login()

<LoginWindow>:
  name: "login"
  email: email
  password: password

  canvas.before:
    Color:
      rgba: (238/255,248/255,247/255,0.8)
    Rectangle:
      pos: self.pos
      size: self.size

  FloatLayout:

    Image:
      source: 'Home5.png'
      keep_ratio: True
      allow_stretch: True
      pos_hint: {'center_x': .5,'center_y':0.6 }

    Label:
      text:"Smart Home"
      font_size: self.width/5
      font_name: "Comic"
      pos_hint: {"center_x":0.5, "top":1}
      size_hint: 0.3, 0.15
      color: (92/255,64/255,51/255,1)

    MDTextField:
      id:email
      text: "alu@gmail.com" #just for the practice
      hint_text: 'Enter your email'
      font_size: self.width/18
      font_name: "Comic"
      multiline: False
      pos_hint: {"center_x":0.5 , "y":0.35}
      size_hint: 0.8, 0.12
      icon_right: "account"

    MDTextField:
      id:password
      text: "1234"
      hint_text: 'Enter your password'
      font_size: self.width/18
      font_name: "Comic"
      multiline: False
      password: True
      pos_hint: {"center_x":0.5, "y":0.25}
      size_hint: 0.8, 0.12
      icon_right: "eye-off"

    MDRoundFlatButton:
      pos_hint:{'center_x':0.5,"y":0.15}
      size_hint: 0.3, 0.07
      font_size: self.width/7
      font_name: "Comic"
      text: "Login"
      on_release:

```

```

root.manager.transition.direction = "up"
root.loginBtn()

```

MDRoundFlatButton:

```

pos_hint:{"center_x":0.5,"y":0.05}
size_hint: 0.8, 0.05
font_size: self.width/20
font_name: "Comic"
text: "Don't have an Account? Create One"
on_release:
    root.manager.transition.direction = "right"
    root.createBtn()

```

<ControlWindow>:

```

name: "control"
n: n
light_text: light_text
light_icon: light_icon

```

canvas.before:

```

Color:
    rgba: (238/255,248/255,247/255,0.8)
Rectangle:
    pos: self.pos
    size: self.size

```

FloatLayout:

```

cols: 1
Label:
    id: n
    font_size: self.width/4
    font_name: "Comic"
    pos_hint:{"x": 0.3, "top":0.95}
    size_hint:0.3, 0.15
    text: "Welcome Home "
    color: (92/255, 64/255, 51/255, 1)

```

Label:

```

font_size: self.width/10
font_name: "Comic"
pos_hint:{"x": 0.2, "top":0.9}
size_hint:0.3, 0.15
text: "What do you want to do? "
color: (92/255, 64/255, 51/255, 1)

```

MDIconButton:

```

pos_hint:{"center_x":0.25,"center_y":0.25}
size_hint_x: 0.3
size_hint_y: 0.2
icon: "door-closed-lock"
icon_size: "64sp"
md_bg_color: (1, 1, 1, 1)
on_release:
    root.unavailable()

```

Label:

```

font_size: self.width/9
font_name: "Comic"

```

```
pos_hint:{"center_x":0.25,"center_y":0.38}
size_hint:0.3, 0.15
text: "Door"
color: (92/255, 64/255, 51/255, 1)
```

```
MDIconButton:
    pos_hint:{"center_x":0.75,"center_y":0.25}
    size_hint_x: 0.3
    size_hint_y: 0.2
    icon: "air-conditioner"
    icon_size: "64sp"
    md_bg_color: (1, 1, 1, 1)
    on_release:
        root.unavailable()
```

```
Label:
    font_size: self.width/9
    font_name: "Comic"
    pos_hint:{"center_x":0.75,"center_y":0.38}
    size_hint:0.3, 0.15
    text: "Air conditioner"
    color: (92/255, 64/255, 51/255, 1)
```

```
MDIconButton:
    pos_hint:{"center_x":0.25,"center_y":0.58}
    size_hint_x: 0.3
    size_hint_y: 0.2
    icon: "shower"
    icon_size: "64sp"
    md_bg_color: (1, 1, 1, 1)
    on_release:
        app.root.current = "bathroom"
        root.manager.transition.direction = "right"
```

```
Label:
    font_size: self.width/9
    font_name: "Comic"
    pos_hint:{"center_x":0.25,"center_y":0.71}
    size_hint:0.3, 0.15
    text: "Bathroom"
    color: (92/255, 64/255, 51/255, 1)
```

```
MDIconButton:
    id: light_icon
    pos_hint:{"center_x":0.75,"center_y":0.58}
    size_hint_x: 0.3
    size_hint_y: 0.2
    icon: "lightbulb-variant-outline"
    icon_size: "64sp"
    md_bg_color: (1, 1, 1, 1)
    on_release:
        root.light_pressed()
```

```
Label:
    font_size: self.width/9
    font_name: "Comic"
    pos_hint:{"center_x":0.75,"center_y":0.71}
    size_hint:0.3, 0.15
    text: "Light"
    color: (92/255, 64/255, 51/255, 1)
```

```

Label:
    id: light_text
    font_size: self.width/9
    font_name: "Comic"
    pos_hint:{"center_x":0.75,"center_y":0.52}
    size_hint:0.3, 0.15
    text: "stat is "
    color: (92/255, 64/255, 51/255, 1)

```

```

MDIconButton:
    pos_hint:{"center_x":0.93,"center_y":0.05}
    size_hint_x: 0.1
    size_hint_y: 0.1
    icon: "logout"
    icon_size: "50sp"
    #md_bg_color: (1, 1, 1, 1)
    on_release:
        app.root.current = "login"
        root.manager.transition.direction = "down"

```

```

<BathRoom>
    name: "bathroom"
    numOfShowers: numOfShowers
    boiler_text: boiler_text
    boiler_icon: boiler_icon

```

```

canvas.before:
    Color:
        rgba: (238/255,248/255,247/255,0.8)
    Rectangle:
        pos: self.pos
        size: self.size

```

```

FloatLayout:
    cols: 1

```

```

Image:
    source: 'bath4.png'
    keep_ratio: True
    allow_stretch: True
    pos_hint: {'center_x': .5,'center_y':0.5 }

```

```

Label:
    font_size: self.width/4
    font_name: "Comic"
    pos_hint:{"center_x": 0.5, "y":0.8}
    size_hint:0.3, 0.3
    text: "Bathroom"
    color: (92/255, 64/255, 51/255, 1)

```

```

Label:
    id: numOfShowers
    text: 'number of showers'
    font_size: self.width/8
    font_name: "Comic"
    pos_hint: {"center_x":0.25 , "y":0.35}
    size_hint: 0.4, 0.12
    color: (92/255, 64/255, 51/255, 1)

```

Spinner:

```
id: numOfShowers
text: "press to choose"
values: ["1", "2", "more"]
pos_hint: {"center_x":0.25 , "y":0.3}
size_hint: 0.4, 0.07
font_size: self.width/10
font_name: "Comic"
background_normal: "
background_color: 238/255,248/255,247/255,0.8
color: (92/255, 64/255, 51/255, 1)
```

MDIconButton:

```
id: boiler_icon
pos_hint:{"center_x":0.75,"center_y":0.25}
size_hint_x: 0.3
size_hint_y: 0.2
icon: "hot-tub"
icon_size: "110sp"
md_bg_color:238/255,248/255,247/255,0.8
```

```
on_release:
    root.boiler_pressed()
```

Label:

```
text: 'Boiler'
font_size: self.width/8
font_name: "Comic"
pos_hint: {"center_x":0.75 , "y":0.35}
size_hint: 0.4, 0.12
color: (92/255, 64/255, 51/255, 1)
```

Label:

```
id: boiler_text
font_size: self.width/9
font_name: "Comic"
pos_hint:{"center_x":0.75,"y":0.3}
size_hint:0.3, 0.15
text: "stat is "
color: (92/255, 64/255, 51/255, 1)
```

Button:

```
id: temp
text: root.val + root.unit
font_size: self.width/5
pos_hint:{"center_x":0.5, "center_y":0.7}
size_hint:0.45, 0.45
color: (92/255, 64/255, 51/255, 1)
background_normal: "
background_color: 0,0,0,0
canvas:
    Color:
        rgba: 92/255, 64/255, 51/255, 1
    Line:
        circle:(self.center_x, self.center_y, self.width/3, -150, root.scale)
        width: 5
        cap: 'round'
```

Image:

```
source: 'round-360-degree-scale-removebg.png'
```

```

keep_ratio: True
allow_stretch: True
pos_hint: {'center_x': .5, 'center_y': 0.7 }
size_hint: 0.5, 0.5
background_normal: "
background_color: 238/255, 248/255, 247/255, 0.8

```

```

MDIconButton:
pos_hint:{"center_x":0.8, "center_y":0.05}
size_hint_x: 0.1
size_hint_y: 0.1
icon: "home"
icon_size: "50sp"
md_bg_color: (1, 1, 1, 1)
on_release:
    app.root.current = "control"
    root.manager.transition.direction = "up"

```

```

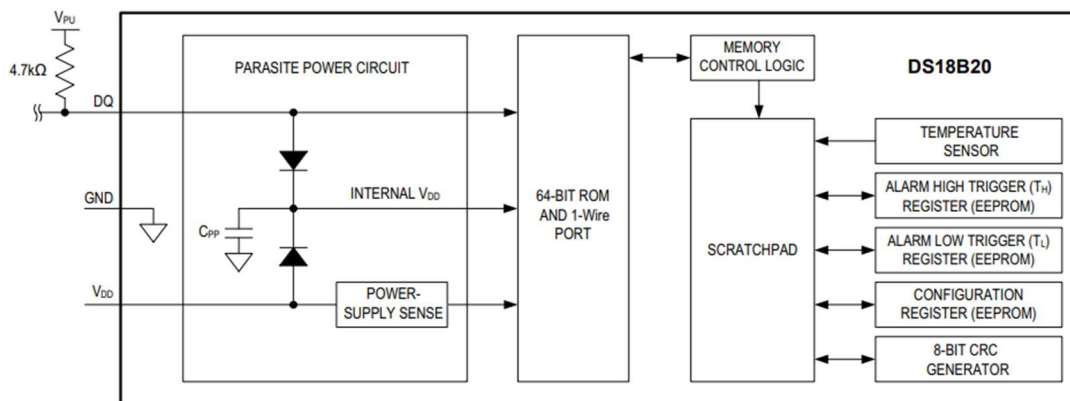
MDIconButton:
pos_hint:{"center_x":0.93,"center_y":0.05}
size_hint_x: 0.1
size_hint_y: 0.1
icon: "logout"
icon_size: "50sp"
#md_bg_color: (1, 1, 1, 1)
on_release:
    app.root.current = "login"
    root.manager.transition.direction = "down"

```

### 11.3 רקע תיאורטי

חיישן טמפרטורה

DS18B20 הוא מדחום דיגיטלי בעל 9 ל-12 סיביות המתורגמות למעלות צלזיוס.



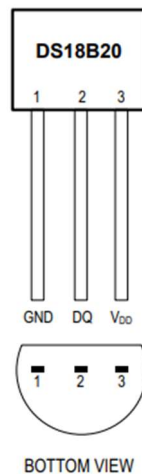
איור 55- סכמת בלוקים של החיישן בטמפרטורה

ניתן לראות באיור זה את זיכרון ה-ROM של 64 סיביות מאחסן את הקוד הסידורי הייחודי של המכשיר. ה-Scratchpad מכיל את הטמפרטורה של 2 בתים. האוגר מאחסן את הפלט הדיגיטלי מחיישן הטמפרטורה. בנוסף, Scratchpad מספק גישה ל-אוגרים העליונים והתחתונים של 1 בייט (TH ו-TH).

TL) ואוגר התצורה של 1 בתיים. אוגר התצורה מאפשר למשתמש להגדיר את הרזולוציה של המרת טמפרטורה לדיגיטל ל-9, 10, 11 או 12 סיביות.

אוגרי ה-TH, TL והקונפיגורציה יציבה (EEPROM), כך ישמרו נתונים כאשר המכשיר מושבת. ה-DS18B20 משתמש בפרוטוקול תקשורת 1-Wire המאפשר תקשורת בעזרת קו תקשורת יחיד. קו הבקרה דורש נגד Pullup. המיקרו-מעבד (המאסטר)

מכשיר) מזהה ומתייחס לחיישנים בקו באמצעות קוד 64 סיביות ייחודי של כל מכשיר. כי כל אחד למכשיר יש קוד ייחודי, מספר המכשירים שניתן לטפל בקו אחד הוא כמעט בלתי מוגבל. תכונה נוספת של ה-DS18B20 היא היכולת לפעול ללא ספק כוח חיצוני. כוח הוא במקום מסופק דרך הנגד 1-Wire pullup דרך ה-DQ כאשר קו במתח גבוה. גם אות הקו הגבוה מטעין קבל פנימי (CPP), אשר לאחר מכן מספק מתח למכשיר כאשר הקו נמוך. שיטה זו של הפקת כוח מהקו 1-Wire מכונה "כוח טפיל". כחלופה, DS18B20 עשוי להיות גם כן מופעל על ידי ספק חיצוני על VDD.



איור 56- רגלי הרכיב DS18B20

#### רגלי חיישן הטמפרטורה:

- 1- חיבור הרכיב לאדמה.
- 2- מוצא הרכיבה (Data).
- 3- חיבור הרכיב למתח אספקה בין 3V עד 5.5 V.

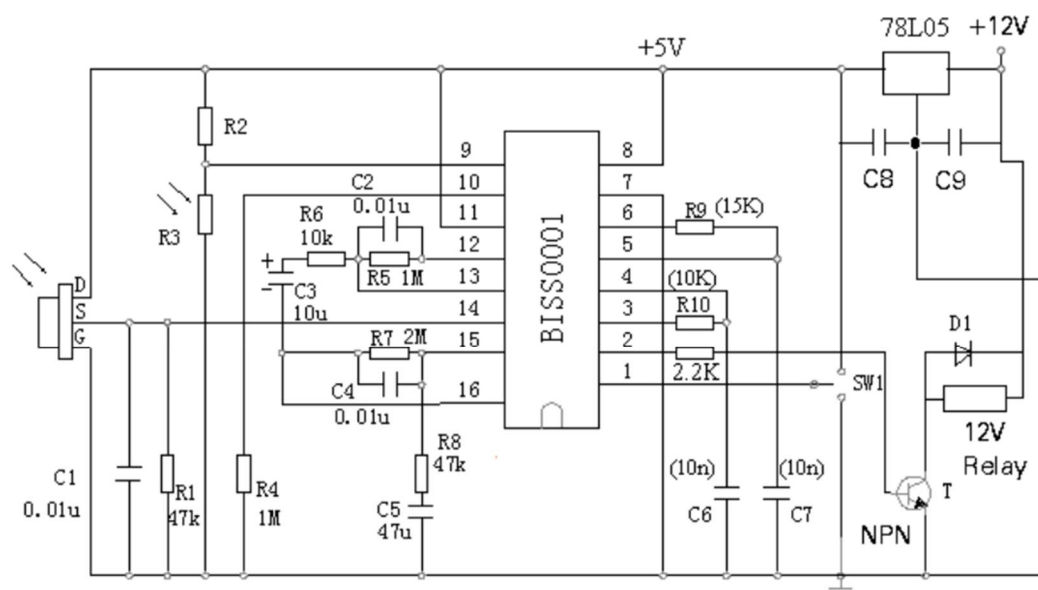
דף נתונים:

<https://www.digikey.co.il/en/products/detail/analog-devices-inc-maxim-integrated/DS18B20/420071>



## חיישן תנועה

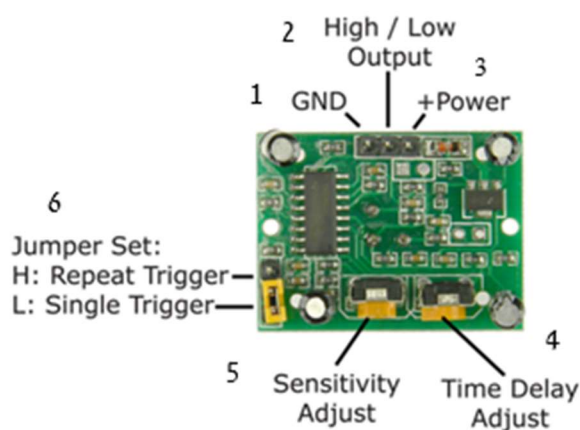
HC-SR501 מבוסס על טכנולוגיית אינפרה אדום, מודול בקרה אוטומטי, רגישות ואמינות גבוהים, מצב הפעלה במתח נמוך במיוחד, בשימוש נרחב בציוד חשמלי חישה אוטומטית.



$$T_x \approx 24576 \times R_{10} \times C_6 ; \quad T_i \approx 24 \times R_9 \times C_7. \quad (\text{ref to schematic})$$

איור 57- סכמת המעגל. HC-SR501.

מתג אינפרה אדום פירואלקטרי הוא מתג אינפרה אדום פסיבי המורכב מ BISS0001, חיישני אינפרה אדום פירואלקטריים וכמה רכיבים חיצוניים. BISS0001 הוא בקר PIR (אינפרה אדום פסיבי), חסין נגד רעש. הבקר מחובר לחיישן האינפרה אדום ומפרש את ערכי המתח המתקבלים ל-1 או 0 בינאריים. כך שציאתו של החיישן HC-SR501 היא ערך דיגיטלי של 1 או 0 לוגים.



איור 58- כניסות ויציאות הרכיב HC-SR501

חיבור המעגל לאדמה.

1- מוצא המעגל- 1 כאשר מזהה תנועה ו0 כאשר אינה מזהה תנועה.

- 2 חיבור המעגל למתח אספקה בין 4 V לבין 12 V.
- 3 ווסת delay, delay המעגל הוא בין 3 שניות ל200.
- 4 ווסת הרגישות, החיישן יכול לקלוט תנועה במרחק של עד 7 מטר.
- 5 ג'אמפר שהנחתו משפיע על תגובת המעגל לתנועה. תגובה לכל תנועה ותגובה לתנועה במידה ומוצא הרכיב הוא 0 לוגי.

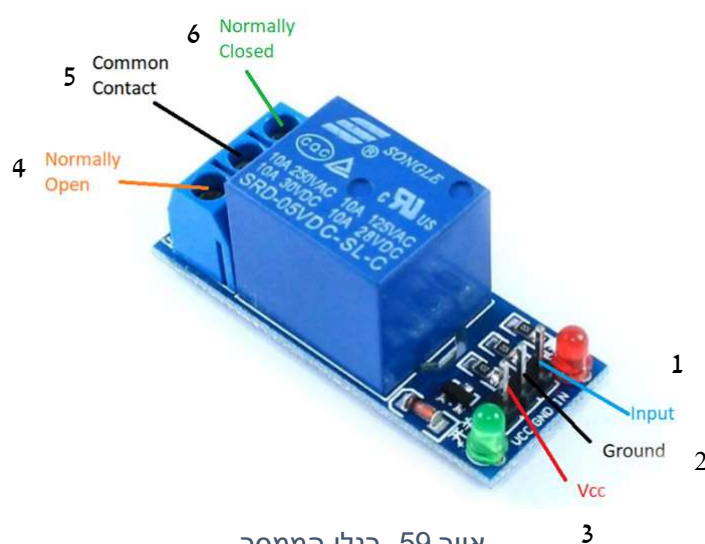
## דף נתונים:

<https://datasheetspdf.com/pdf/775435/ETC/HC-SR501/1>

## ממסר- SRD-05VDC-SL-C

מסר הוא רכיב השולט בזרם חשמלי וממתג אותו. הוא דו-מצבי בלבד ON או OFF. הוא אינו מסוגל לשלוט בעוצמת הזרם. אלא רק לחסום אותו או לתת לו לעבור במלואו. שנית, ממסרים מבודדים את המעגלים השונים: אין GND משותף לכל הרכיבים משני צדי הממסר, כך ששימוש בממסר הוא בטוח יותר ואין חשש שמיקרו-בקר (רכיב רגיש) יפגע. בהתאם למפרט של הממסר הספציפי – ממסר עובד במתחים וזרמים גבוהים.

כאשר מעבירים דרך הסליל זרם נוצר שדה מגנטי, שמושך לשונית מתכת. הלשונית הזו סוגרת מעגל בין רגל הכניסה (Common) לאחת מרגלי היציאה (Normally Open) כשאין זרם, הלשונית חוזרת למקומה המקורי וסוגרת את המעגל בין הכניסה לרגל היציאה השנייה (Normally open). כאשר הלשונית יוצרת מגע פיזי עם יציאה כלשהי נשמע קליק חד ואופייני.



איור 59- רגלי הממסר

## רגלי הממסר:

- 1 רגל כניסת הממסר .
- 2 חיבור הממסר לאדמה
- 3 חיבור בממסר למתח האספקה 5 V.
- 4 רגל ה-Normally open.

-5 רגל המשותפת Common.

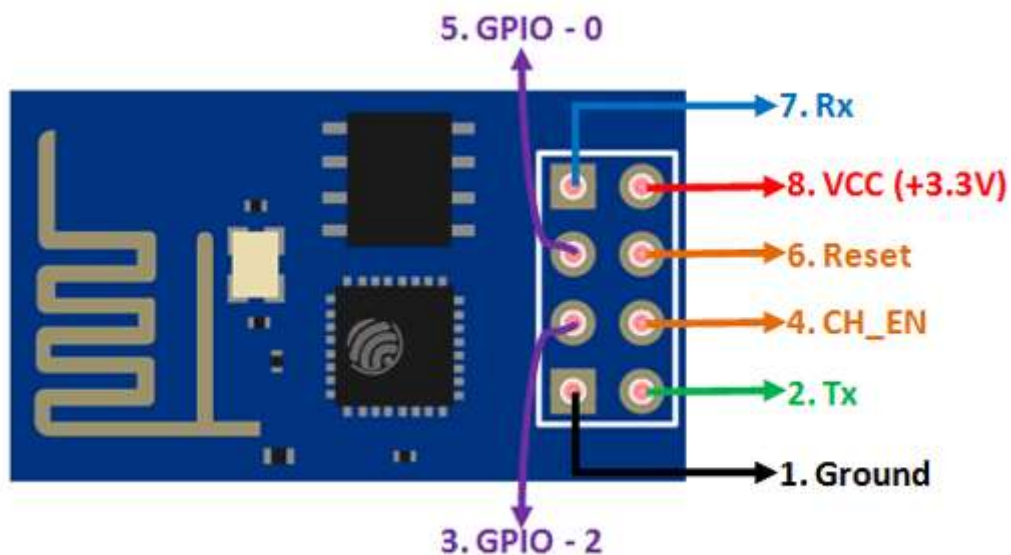
-6 רגל ה-Normally closed.

דף הנתונים:

<https://components101.com/switches/5v-single-channel-relay-module-pinout-features-applications-working-datasheet>

### ESP8266-01

ה-**ESP8266-01** הוא מיקרו שבב Wi-Fi בעל יכולות של מיקרו בקר, המתקשר באמצעות פרוטוקול TCP/IP או פרוטוקולים הבנויים עליו. רכיב זה מאוד ידידותי למשתמש. הוא יכול לעבוד גם כנקודת גישה (יכול ליצור נקודה חמה) וגם כתחנה (יכול להתחבר ל-Wi-Fi), מכאן שהוא יכול בקלות לקבל נתונים ולהעלות אותם לאינטרנט מה שהופך את ה- IOT (Internet of Things) לקל ככל האפשר. ניתן לקחת נתונים מהאינטרנט באמצעות API (Application Programming Interface), כך ניתן לגשת לכל מידע שזמין באינטרנט. תכונה נוספת של מודול זה היא שניתן לתכנת אותו באמצעות Arduino IDE מה שהופך אותו להרבה יותר ידידותי למשתמש. עם זאת, לגרסה זו של המודול יש רק 4 פני GPIO.



איור 60 -כניסות ויציאות הרכיב ESP8266-01

### פירוט כניסות ויציאות הרכיב ESP8266-01:

- 1 חיבור המעגל לאדמה.
- 2 יכול לשמש כרגל העברת מידע (TX) וכרגל כניסה ויציאה.
- 3 רגל יציאה/כניסה של המעגל.

- 4 רגל האפשר של הרכיב, פעיל בגבוהה.
- 5 רגל יציאה/כניסה של המעגל. בנוסף בקבלת מתח נמוך בחיבור ראשוני מאפשרת צריבה על הרכיב.
- 6 רגל האתחול, בפילה בנמוך.
- 7 יכול לשמש כרגל קליטת מידע (RX) וכרגל כניסה ויציאה.
- 8 חיבור המעגל למתח אספקה של 3.3 V.

דף נתונים:

[https://docs.ai-thinker.com/\\_media/esp8266/docs/esp-01e\\_product\\_specification\\_en.pdf](https://docs.ai-thinker.com/_media/esp8266/docs/esp-01e_product_specification_en.pdf)