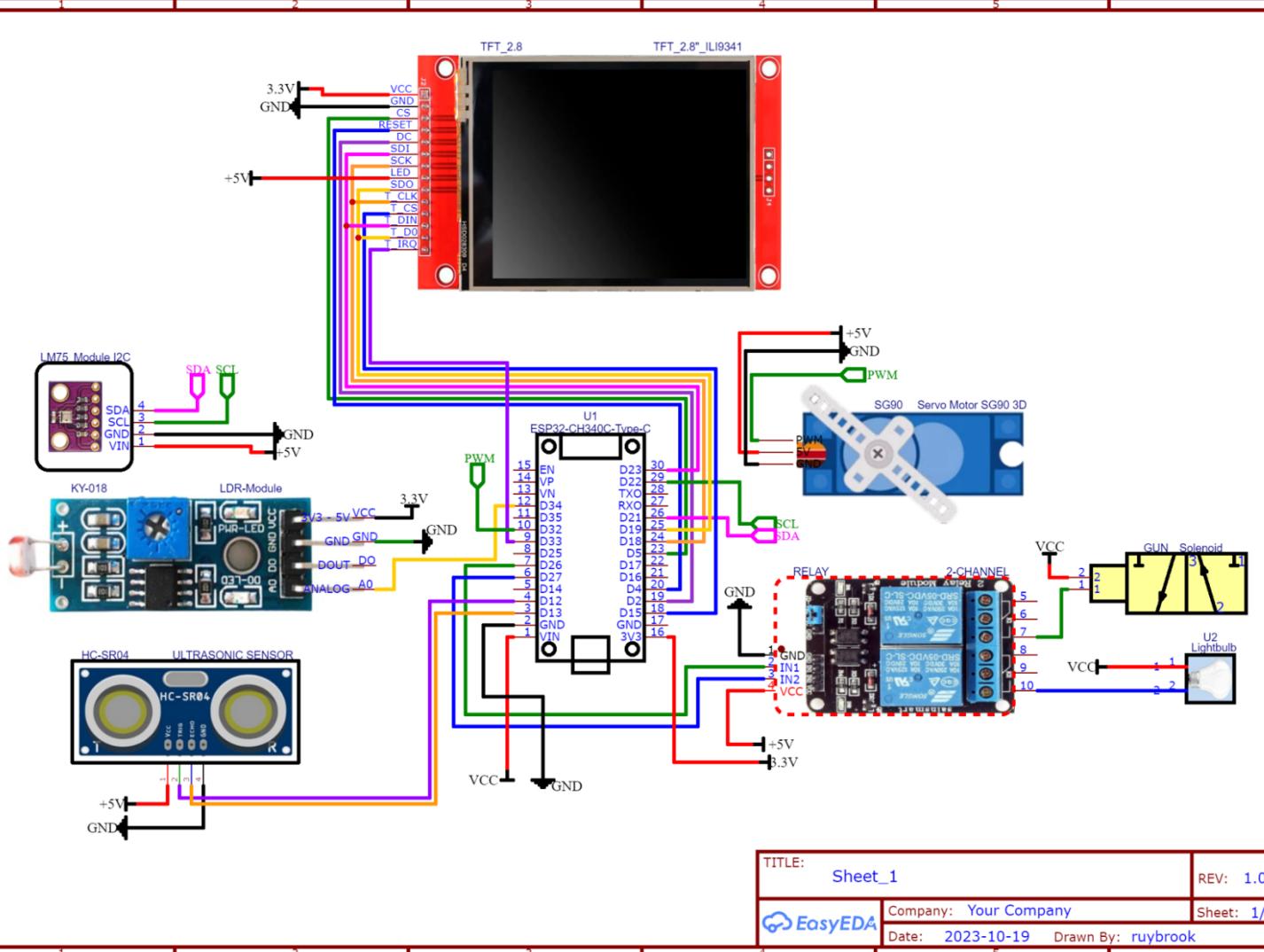
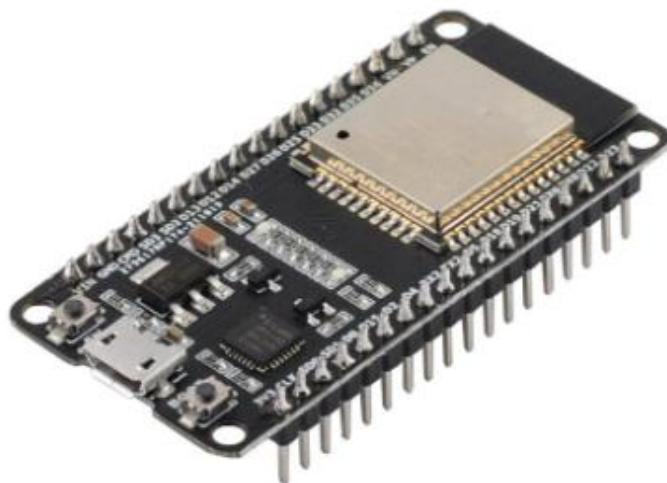


שרטוט חשמלי ראשוני בתוכנת EASYEDA



זה למעשה השרטוט הראשוני עבור המערכת אליה התchingיבתי בהצעת פרויקט. הפרויקט שעשייתי בפועל הוא יותר מרכיב ומחולק לשתי מערכות שמתקשרות ביניהן בעזרת פרוטוקול תקשורת אלחוטי הנקרא **ESP_NOW** אני משתמש בשני מיקרו בקרים מסוג **ESP32** ואתייחס לכך בשרטוט החשמלי הסופי שלי לפרויקט שם ניתן יהיה לראות בבירור את המערכות וכיידן מתחברות למערכת אחת גודלה.

מיקרו בקר ESP32



ה- **ESP32** הוא הרבה יותר מסתם מיקרו-בקר פשוט ; הוא מערכת על שבב (SoC) חזקה ורב-תכליתית, המשלבת את כל מה שצריך כדי לבנות פרויקט חכם, מוקשר וחסכוני באנרגיה. פרי-פיתוחה של חברת **Espressif Systems**, ESP32 בולט בזכות השימוש הייחודי שלו עצמת עיבוד חזקה, קישוריות אלחוטית וכיותה צריכת חשמל נמוכה.

תכונות טכניות עיקריות:

- .**1. עיבוד בעל ביצועים גבוהים** : מצויד בمعالג כפול-ЛИבָה במהירות של עד 240 MHz, ה- ESP32 מסוגל לבצע משימות מורכבות ולנהל מספר פעולות במקביל במהירות גבוהה.
- .**2. קישוריות אלחוטית מובנית** : ה- ESP32 מגיע עם **Wi-Fi** ו- **Bluetooth** מובנים, מה שהופך אותו לבחירה אידיאלית לפיתוח פרויקטים מחוברים ללא צורך ברכיבים חיצוניים נוספים.
- .**3. יעילות אנרגטית** : עם מצבי שינה מרובים, ה- ESP32 מתוכנן לפעול לטוווח ארוך על סוללה בודדת, מה שהופך אותו למושלם עבור יישומים המופעלים באמצעות סוללה.
- .**4. ממשקים קלט/פלט (I/O) גמישים** : ה- ESP32 מציע מגוון רחב של ממשקים היקפיים, כולל **GPIO, ADC, DAC, I2C, SPI**.
- .**5. תמיכת קהילה נרחבת** : עם אלפי מפתחים ברחבי העולם המשתמשים ב- ESP32, קיימים שפע של ספריות קוד, תיעוד ופרויקטי קוד פתוח הזמינים לתמיכת בפיתוח שלך.

הסוד שמאחורי ה-ESP32: יחידת העיבוד המרכזית (CPU)

ה-ESP32 מצויד בمعالג עצמאי Xtensa LX6 dual-core, הפועל במהירות מרשימה של עד 240MHz. אבל מה זה בעצם אומר?

- דמיינו שיש לנו שני מוחים עובדים במקביל: בזמן אחד מהם מטפל בקשריות-Wi-Fi, השני יכול לעבוד נתונים מהחישונים שלכם - הכל בבת אחת!
- **240MHz** זהה למהירות השעון של המعالج. לשם השוואה, המعالج של ה-ESP32 מסוגל **לבצע 240 מיליון פעולות בשנייה!**
- **ארQUITקטורת 32-ביט** זה אומר שהוא יכול לעבוד נתונים בכמותות גדולות יותר, מה שמאפשר חישובים מורכבים במהירות גבוהה.

הזיכרון: איפה כל המידע נשמר?

ה-ESP32 מגיע עם מערך מרשים של אפשרויות זיכרון:

- **SRAM (Static Random-Access Memory)**: זיכרון מהיר לשימוש מיידי של 520KB. זהו המקום שבו המידע שהمعالגעובד עליו ברגע נתון נשמר.
- **ROM (Read-Only Memory)**: זיכרון קבוע של 448KB המכיל את ההוראות הבסיסיות הנדרשות להפעלת המערכת.
- **Flash (זיכרון חיצוני)**: זיכרון של עד 16MB שבו נשמרות תוכניות המשמשותם והנתונים הקבועים.

לשם השוואה, הזיכרון הכלול של ה-ESP32 הוא כל כך גדול, שניתן לאחסן בו ספר אודיו שלם!

קשריות אלחוטית: החיבור לעולם:

ה-ESP32 הוא אלף התקשורות האלחוטית:

- **Wi-Fi**: תומך בטקן 802.11 n/g/b, מה שמאפשר חיבור כמעט ללא כל רשות אלחוטית מודרנית. הוא יכול לשמש גם כנקודת גישה ולספק חיבורם למכשירים אחרים.
- **Bluetooth Low Energy (BLE)**: עם תמיכה ב-Bluetooth Classic וגם ב- BLE, ה-ESP32 יכול לתקשר עם מגוון רחב של מכשירים, טלפונים חכמים ועוד חישונים צעירים.

ממשקים: חיבור לעולם הפיזי:

ה-ESP32 מציע מגוון רחב של אפשרויות חיבור כדי לתקשר עם העולם הפיזי:

- **GPIO (General Purpose Input/Output)**: עד 36 פינים, המשמשים כ"עצבים" של ה-ESP32, המאפשרים לו לחוש ושלוט בסביבה.
- **ממיר אנלוגי-דיגיטלי (ADC)**: ישנו 18 ערוצים המאפשרים ל-ESP32 ל לקרוא ערכים אנלוגיים מחישונים כמו טמפרטורה או עצמת אור.
- **ממיר דיגיטלי-אנלוגי (DAC)**: שני ערוצים המאפשרים לייצר אותן אוטות אנלוגיים כמו צלילים, או מתחים משתנים.
- **חישוני מגע (Touch Sensors)**: 10 פינים מגע מובנים המאפשרים ל-ESP32 לחוש מגע ישיר.

צריכת חשמל:יעילות אנרגטית מרשימה:

אחד ה יתרונות הגודלים ביותר של ה-ESP32 הוא יכולת שלו לחסוך באנרגיה:

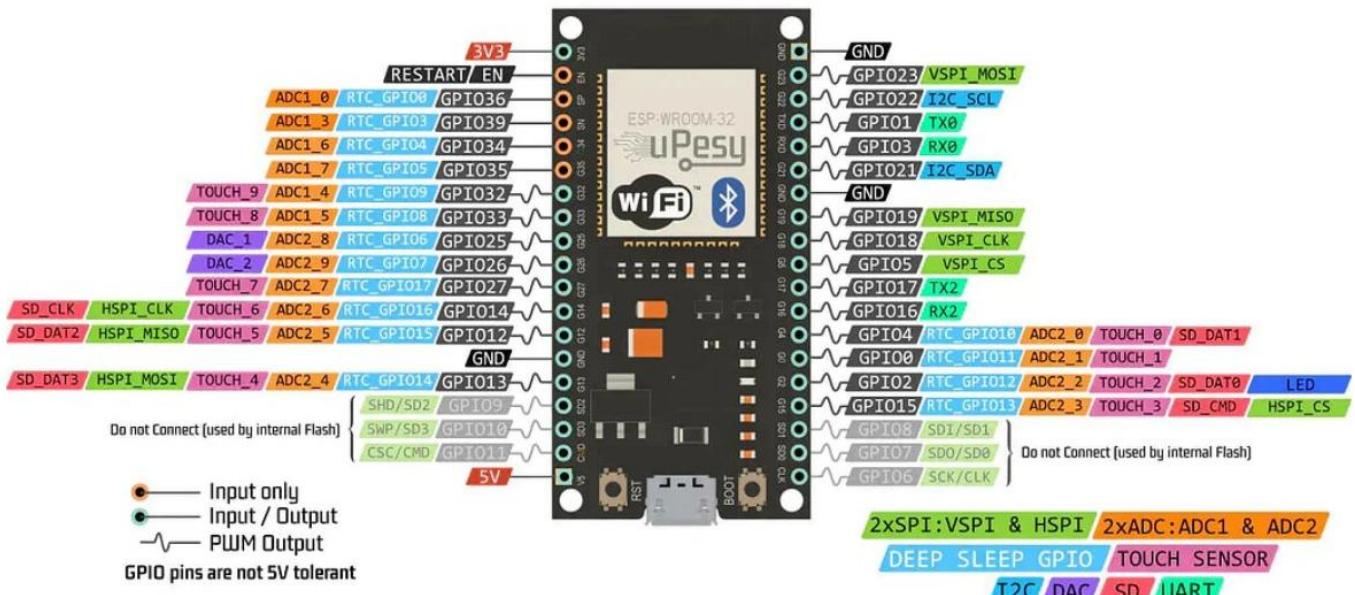
- **מצב פעיל מלא:** במצב זה, צריכת הזרם היא עד **.mA240**.

- **מצב שינה عمוקה:** במצב זה, הצריכה יורדת באופן דרמטי ל-**A10μ** בלבד!

זה אומר שגם עם סוללה קטנה, הפרויקט שלכם יוכל לפעול במשך שבועות ואף חודשים ארוכים.

כעת נראה את צורת הפנים במקרו בקר ESP32

ESP32 Wroom DevKit Full Pinout



כפי שכבר רأינו, ה-ESP32 מוקף בפינים רבים, שהם ה"שערים" דרכם הוא מתקשך עם העולם החיצוני. הבנה מעמיקה של פינים אלו היא המפתח לייצור פרויקטים מודרכים. בואו נצלול לעולם המרתך של פיני ה-ESP32:

סוגי הפינים העיקריים:

פיני (General Purpose Input/Output)

- אלו הם הפינים הרוב-תכלתיים של ה-ESP32.
- יכולים לשמש כקלט (לקראת מצבים) או כפלט (להפעלת רכיבים).
- ה-ESP32 מציע עד **34 פיני GPIO**, אך חלקם משמשים גם למטרות אחרות.
- **שימושים נפוצים:** חיבור נורות LED, לחצנים, מנועים ועוד.

:ADC (Analog to Digital Converter)

- אפשררים קראיה של מערכות אנלוגיות ומרתם למערכות דיגיטליים.
- ה-ESP32 מציע שני מגלי ADC עם סה"כ **18 ערוצים**.
- רזולוציה של 12 ביט** מאפשרת **4,096 רמות שונות** של קראיה.
- שימושים:** מדידת טמפרטורה, עצמת אור, לחות קרקע ועוד.

:DAC (Digital to Analog Converter)

- ממירים ערcis דיגיטליים לאוותות אנלוגיות.
- ה-ESP32 מציע שני ערוציו DAC ברזולוציה של **8 ביט**.
- **שימושים נפוצים:** יצירת צלילים, בקרת מתח ועוד.

:Touch

- ה-ESP32 מציע **10** חיישני מגע קפסיטיביים מובנים.
- אפשררים זיהוי מגע אנושי ישיר, ללא צורך ברכיבים נוספים.
- **שימושים נפוצים:** יצירת ממשקי משתמש מבוססי מגע, לחצנים וירטואליים ועוד.

:פיני תקשורת:

- **UART:** לתקשורת טורית. ה-ESP32 מציע **3 יחידות UART**.
- **SPI:** לתקשורת מהירה. ה-ESP32 תומך במספר חיבורו SPI במקביל.
- **I2C:** לתקשורת עם מגוון רחב של חיישנים ורכיבים. ה-ESP32 מציע שתי יחידות **I2C**.

:פינים מיוחדים:

- **פין BOOT:** משמש לכינסה למצב **כרייבת תוכנה (flashing mode)**.
- **פין (EN) (Enable):** משמשים להפעלה וכיבוי של השבב.
- **פיני אספקת מתח:** פינים המספקים מתחים של **5V, 3.3V** וכן **GND**.

:נקודות חשובות לגבי השימוש בפינים:

- **רמות מתח:** פיני ה-ESP32 עובדים ברמת מתח של **3.3V**. חיבור למתח גבוה יותר עלול לגרום נזק.
- **פינים מרובי תפקידים:** רבים מהפינים יכולים לשמש במספר מטרות. חשוב לתכנן את השימוש בהם בקפידה.
- **הגנה על הפינים:** מומלץ להשתמש ב נגדים מגבילים זרם בעת חיבור רכיבים חיצוניים, במיוחד עם נורות LED.
- **פינים מיוחדים:** חלק מהפינים GPIO (כמו 0, 2, 15) משמשים במהלך תהליך האתחול. שימוש לא נכון בהם עלול למנוע מה-ESP32 לפעול כראוי.
- **פולאף ופולאדיון (Pull-up and Pull-down):** חלק מהפינים כוללים גודי Pull-Up או Pull-Down פנימיים. זה יכול להוות שימוש חשוב, אך חשוב להיות מודעים לכך בעת תכנון המעגל.

הבנה מעמיקה של מבנה הפינים ותכונותיהם היא המפתח לייצרת פרויקטים מוצלחים ויעילים עם ESP32. זה מאפשר לנו לנצל את מלאה הפוטנציאלי של השבב העוצמתי הזה.

:הכוח האלחוטי של ESP32

אחד ה יתרונות הגודולים ביותר של ה-ESP32 הוא יכולתו המובנית לתקשורת באמצעות אלחוטי.

החיבור לעולם הרחב Wi-Fi

ה-ESP32 תומך בתקן **Wi-Fi 802.11 b/g/n**, מה שמאפשר לו להתחבר כמעט לכל רשת אלחוטית מודרנית.

מצבי פעולה עיקריים:

• מצב תחנה (Station Mode - STA)

- ה-ESP32 מתחבר לרשת Wi-Fi קיימת, כמו כל מכשיר קצה רגיל.
- שימושים נפוצים: התחברות לאינטרנט דרך הרouter הביתי, שליחת נתונים למסד נתונים מרוחק ועוד.

• מצב נקודת גישה (Access Point - AP):

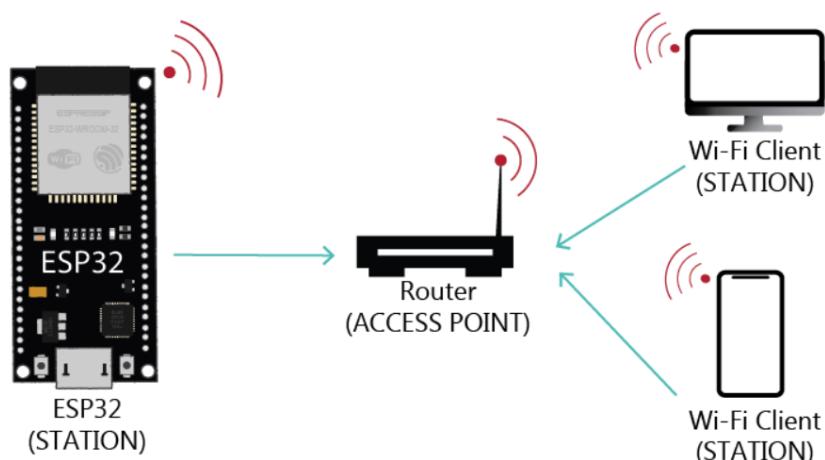
- ה-ESP32 יוצר רשת Wi-Fi משלו, אליה יכולים להתחבר מכשירים אחרים.
- מצוין לייצור פרויקטים שימושיים ממשק אינטרנט או תקשורת ישירה עם מכשירים ניידים.
- שימושים נפוצים במצבים בהם אין גישה לרשת Wi-Fi חיצונית.

• מצב כפול (Dual Mode):

- ה-ESP32 יכול לפעול במקביל גם כתחנה וגם כנקודת גישה בו-זמנית!
- אפשר ליצור "גשר" בין רשתות או אספקת גישה לאינטרנט דרך ה-ESP32.

יכולות מתקדמות של Wi-Fi:

- **שירות אינטרנט (Web Server):** ה-ESP32 יכול לשמש כשרת אינטרנט, מה שמאפשר יצירה ממשקי משתמש מרוחק ודף תצוגה נתונים.
- **MQTT:** תמייהה בפרוטוקול זה מאפשרת תקשורת יעילה בין מכשיר IoT ושרותים מרכזיים.
- **OTA (Over-The-Air) Updates:** עדכון תוכנה מרוחק, ללא צורך בחיבור פיזי למחשב.
- **ESP-NOW:** פרוטוקול תקשורת ייחודי של Espressif המאפשר תקשורת ישירה בין מכשירי ESP32 בזריכת אנרגיה נמוכה.



: תקשורת קצרת טווח עם יכולות ארכובות טווח Bluetooth

ה-ESP32 תומך הן ב-(BLE) Bluetooth Low Energy והן ב-(Classic), מה שפותח אפשרויות רבות:

:Bluetooth Classic

- מתאים לתקשורת עם מכשירים ישנים יותר או כנדרשת העברת נתונים בנפח גדול.
- שימושים נפוצים: הזרמת אודיו, העברת קבצים, ותקשורת עם מכשירים שאינם תומכים BLE.
- אפשר יצירה פרופילים שונים כמו **Serial Port Profile (SPP)** לתקשורת טורית אלחוטית.

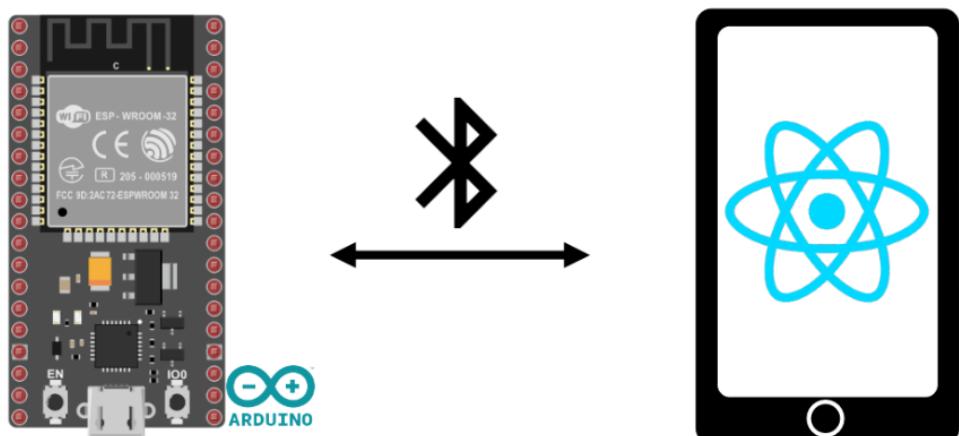
:Bluetooth Low Energy (BLE)

- צריכה אנרגיה נמוכה במיוחד, אידיאלי למכשירים המופעלים על סוללה.
- מתאים לשילוח נתונים קטנים באופן_TDיר, כמו נתונים חיישנים.
- תומך במודול **GATT (Generic Attribute Profile)** המאפשר יצירה שירותים ומאפיינים מותאמים אישית.
- אפשר יצירה "bijoux" לשיווק מידע ללא צורך בחיבור מלא.

שילוב Wi-Fi ו-Bluetooth: יתרונות מרתקים:

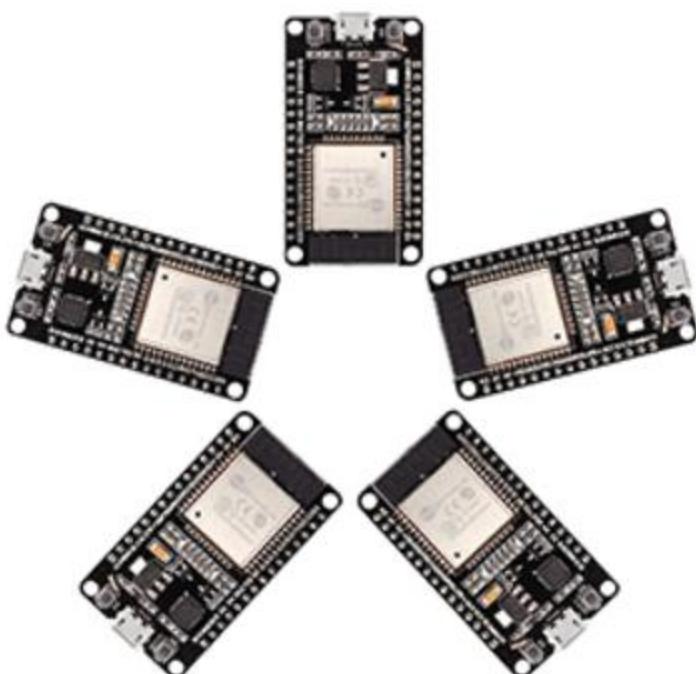
אחד היתרונות הגדולים של ה-ESP32 הוא יכולת להשתמש גם ב-Bluetooth וגם ב-Wi-Fi במקביל. זה פותח אפשרויות מרתקות:

- **שער (Gateway):** קבלת נתונים ממיכרי Bluetooth ושליחתם לענן דרך Wi-Fi.
- **שליטה מקומית ומרוחקת:** שליטה בפרויקט דרך Bluetooth כשלרבים, ודרך Wi-Fi כশמרוחקים.
- **רשתות היברידיות:** שימוש ב-BLE לתקשורת בין חיישנים קרובים, וב-Wi-Fi לשילוח נתונים רבים לשרת מרכזי.



תדרונות מרכזיים של ESP32:

- .**1. עוצמה חיובית**: עם מעבד כפול-ליבה ב מהירות של עד 240 MHz-ESP32-מאפשר ביצוע משימות מורכבות ועיבוד נתונים מהיר, מה שהופך אותו למתאים לפרויקט IoT מתקדמים.
- .**2. קישוריות אלחוטית מקיפה**: תמייהה מובנית ב- **Wi-Fi** ו- **Bluetooth** (כולל **BLE**) מאפשרת ייצור מגוון רחב של פרויקטים מחוברים, החל מממערכות בית חכם ועד לישומים תעשייתיים.
- .**3. גמישות בחיבורים**: מגוון רחב של ממשקים כמו **GPIO, ADC, DAC** ופרוטוקולי תקשורת כמו **I2C, UART, SPI** מאפשרים אינטגרציה קלה עם כמעט כל סוג של חישון או פעולה.
- .**4.יעילות אנרגטית**: מבצעי שינה מתקדמים וצריכת חשמל נמוכה מאפשרים ייצור מכשירים ניידים עם חיי סוללה ארוכים, מה שהופך אותו לאידיאלי למערכות ניטור מרוחקות.
- .**5. פיתוח נגיש**: תמייהה במגוון סביבות פיתוח כמו **ESP-IDF** ו- **Arduino IDE** וקהילת מפתחים גדולה ופעילה מקלים על תהליכי הפיתוח ומספקים שפע של ספריות תוכנה.
- .**6. יכולות IoT מתקדמות**: תמייהה בפרוטוקולים כמו **MQTT** ויכולות עדכון תוכנה מרוחק (OTA) מאפשרות הקמת שרתים **web** מקומיים וניהול התקנים מכל מקום בעולם.
- .**7. אבטחה משולבת**: תכונות אבטחה מובנות כמו הצפנה חזקה ו- **secure boot** חיוניות להגנה על מכשירי IoT ועל נתונים המשתמשים בעולם המחבר.
- .**8. חיישנים מיוחדים מובנים**: ה- ESP32 כולל חיישני מגע קפטייטיביים, המאפשרים ייצור ממשקים משתמש חדשניים ללא צורך ברכיבים נוספים.
- .**9. יכולות רשת מתקדמות**: תמייהה ביצירת רשתות **Mesh** ושימוש בפרוטוקול - **ESP-NOW** מאפשרות ייצור רשתות תקשורת מרובות ורב-כיווניות.
- .**10. התאמה למגוון יישומים**: מפרויקטים פשוטים ועד למערכות תעשייתיות מורכבות, ה- ESP32 מציע את הגמישות והעוצמה הדרושים למגוון רחב של יישומים.



ESP NOW

פרוטוקול ESP-NOW הוא טכנולוגיית תקשורת אלחוטית ללא חיבור (Connectionless) שפותחה על ידי חברת Espressif המאפשרת למספר מכשירים לתקשר זה עם זה ללא צורך בתשתיות Wi-Fi או ברואוטר. ה프וטוקול פועל בתדר 2.4 GHz ומאפשרן ביצירת הספק נמוכה, בדומה לקשרויות הקשורות בצד היקפי אלחוטי. על מנת ליצור תקשורת, נדרש ביצוע של תהליך צימוד (Pairing) בין המכשירים מראש. לאחר השלמת הצימוד, נוצר חיבור מאובטח בתצורת עמית-עלמית (Peer-to-Peer), אשר אינו מצריך תהליך "לחיצת יד" (Handshake). " בכל העברת נתונים. מאפיין זה מאפשר העברת כבilities מידע במהלך ייצואת דופן ובייעילות אנרגטית גבוהה, מה שהופך את ה프וטוקול לפתרון אידיאלי עבור ישומי IoT וחישוניים המופעלים על סוללות.

מאפייני פרוטוקול ESP-NOW

- הגדרה:** על פי אתר האינטרנט של Espressif, זהו פרוטוקול המאפשר למספר מכשירים לתקשר זה עם זה באמצעות Wi-Fi.
- טכנולוגיה:** ה프וטוקול דומה לקשרויות אלחוטיות עמית-עלמית (Peer-to-Peer), אשר אין צורך בתדר GHz.
- תהליכי הצימוד:** נדרש ביצוע צימוד (Pairing) בין המכשירים לפני תחילת התקשרות ביניהם.
- אבטחה וחיבור:** לאחר סיום הצימוד, החיבור הוא בטוח ובतצורת עמית-עלמית (Peer-to-Peer), כאשר אין צורך בתהליכי "לחיצת יד" (Handshake).
- רציפות התקשרות:** החיבור בין המכשירים הוא קבוע (Persistent). אם אחד הלוחות מאבד מתח או מתAES, הוא יתחבר מחדש באופן אוטומטי לעמידה שלו עם הפעלו החדש כדי להמשיך בתקשורת.

יכולות ה프וטוקול:

- תמייה בתקשורת Unicast מוצפנת ושאיתנו מוצפנת.
- תמייה בשילוב של מכשירי עמית מוצפנים ושאיתנו מוצפנים.
- יכולת נשיאת מטען נתונים (Payload) של עד 250 Bytes.
- אפשרות להגדרת פונקציית Callback לשיליחה, המיידעת את שכבת האפליקציה על הצלחה או כישלון של השידור.

מוגבלות טכניות:

- מוגבלות עמיתים מוצפנים:** תמייה ב-10 עמיתים מוצפנים לכל היוטר במצב Station ועד 6 עמיתים במצב SoftAP או מצב משולב (SoftAP + Station).
- סך עמיתים כללי:** ניתן לצמד מספר עמיתים שאינם מוצפנים, אך המספר הכללי של המכשירים (כולל המוצפנים) חייב להיות נמוך מ-20.
- נפח הודעה:** מטען הנתונים מוגבל ל 250 Bytes בלבד.

במילים פשוטות **ESP-NOW**, הוא פרוטוקול תקשורת מהיר שנitin להשתמש בו כדי להחליף הودעות קטנות (עד 250 בתים) בין לוחות ESP32.

ESP-NOW הוא ורטטייל מואוד וניתן לקים באמצעות תקשורת חד-כיוונית או דו-כיוונית בתצורות שונות.

תקשורת חד-כיוונית ב-ESP-NOW

לדוגמא, בתקשורת חד-כיוונית, ניתן לקיים תרחישים כגון :

- לוח ESP32 אחד השולח נתונים ללוח ESP32 אחר.

צורה זו קלה מאוד לישום והיא מצוינה לשילוח נתונים מלאח אחד למשנהו, כמו קריאות חיישנים או פקודות הפעלה (ON) וכיבוי (OFF) לשילטה פנימי ה-GPIO.



שליחה מלאה אחד למספר לוחות(One-to-Many):

לוח ESP32 אחד השולח פקודות זהות או שונות ללוחות ESP32 שונים. צורה זו אידיאלית לבניית מכשיר הדומה לשלט רחוק. ניתן להציב מספר לוחות ESP32 ברוחבי הבית הנשלטים על ידי לוח ESP32 מרכזי אחד.



כמה נקודות לציין בשימוש בתצורה זו:

- **שליטה מרכזית**: לוח אחד משמש כ"MASTER" (Master) ווהשאר כ"SLAVES" (Slaves).
- **גמישות**: ניתן לשלוח פקודה אחת לכלם בו-זמנית או פקודות ייעודיות לכל לוח בנפרד.

איסוף נתונים ממספר לוחות לוח אחד (Many-to-One)

תצורה זו אידיאלית אם ברצון לאסוף נתונים ממספר צמתי חישנים לוח ESP32 אחד. ניתן להציג לוח זה כשרת אינטרנט (Web Server) כדי להציג נתונים מכל שאר הלוחות, למשל.



תקשורת דו-כיוונית ב-ESP-NOW (Two-Way Communication)

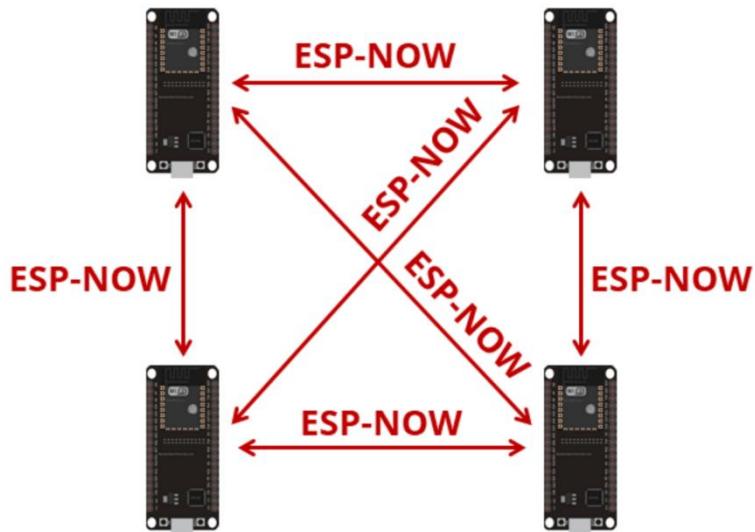
עם ESP-NOW כל לוח יכול לתפקיד שולח ו מקבל בו-זמנית. כך שניתן :

- **לקיום תקשורת דו-כיוונית בין שני לוחות:** לדוגמה, שני לוחות ESP32 יכולים להחליפה בינהם הודעות, קריאות חישנים או אישורי קבלה.
- **להציג רשות שבה כל לוח מתקשר עם האחרים:** בתצורה זו, כל יחידה מסוגלת לשולח נתונים ו גם להאזין לנדרונות המגיעים מלוחות אחרים ברשות.

תצורה זו שימושית במיוחד כאשר יש צורך במסוב – (Feedback) למשל, לוח אחד שולח פקודה להדילק אור, והלוח השני מחזיר אישור שהפעולה בוצעה בהצלחה.



ניתן להוסיף לוחות נוספים לתצורה זו וליצור מבנה הדומה לרשת) כל לוחות ה-ESP32 מתקשרים זה עם זה בתצורה זו, כל לוח יכול לשלח נתונים לכל לוח אחר בטוחה הקליטה, מה שמאפשר גמישות מרבית במערכות מבזורת או ביישומי בית חכם מורכבים.



השגת כתובת ה- MAC של הלוח

כדי לתקשר באמצעות פרוטוקול ESP-NOW עליך לדעת את כתובת ה MAC-של ה-ESP32-המקבל. כך תדע לאיזה מכשיר לשלח את הנתונים.

כל לוח ESP32 יש כתובת MAC ייחודית, וזה הדרך שבה אנו מזהים כל לוח כדי לשלח אליו נתונים באמצעות ESP-NOW.

תקשות טורית UART

הקדמה:

תקשות טורית (UART) היא שיטה נפוצה ו פשוטה להעברת מידע בצורה טורית (סידורית) בין שני התקנים דיגיטליים, כגון בין מיקרו-בקר לבין מחשב או מודם. זהו אחד מפרוטוקולי התקשורת הבסיסיים ביותר הנמצאים בשימוש בתחום האלקטרוניקה הדיגיטלית.

בבסיסה, תקשורת UART מורכבת וכוללת פרוטוקולים המאפשרים שידור וקליטה של מידע בצורה אמינה ויעילה. היא משתמשת בשני חוטי תקשורת עיקריים: "קבל (RX)" ו- "שלח (TX)".

מהי תקשורת UART?

תקשות טורית (UART) היא שיטה פשוטה ונוחה להעברת מידע בין שני התקנים דיגיטליים, למשל בין מיקרו-בקר כמו ארדואינו למחשב.

בקשות UART המידע מועבר בצורה **סיביות** (זרמים מתחלפים גבירות ונדירות - ראו תמונה בהמשך) וMOVBRORTS בחוותי תקשורת נפרדים בין שני המכנים. כל מקשר מחובר עם שני פינים לתקשורת - **קלט (RX)** ו**פלט (TX)**.

כדי לשדר מידע, המקשר (השולח) שולח את הביטים על גבי קו ה- TX. הצד השני, המתקבל מאזין לאותו קו ומזהה את המידע בכניסה ה- RX. שלו ופענה אותו בחזרה למידע דיגיטלי.

עקרונות חשובים ב-UART הם **קצב התקשורת** - baud rate - כמה ביטים בשנייה, (**אורץ קבוע** - 8 ביטים בשנייה, **בית התחלה** (סינכרון לתחילת העברת מידע), **בית סיום** (סימון סיום ההעברה).

היתרונות הגדולים של תקשורת טורית הם פשוטות וミニימליות בחומרת המקשר, מה שמאפשר להשתמש בה לישומים רבים עם מגבלות תוכנה וחומרה.

לכן UART היא אחת השיטות הנפוצות להעברת נתונים בין מיקרו-בקרים לבין מחשבים במגוון יישומים כמו התקני אינטרנט של הדברים, מערכות בקרה תעשייתית, רובוטיקה ועוד.

Łסיכום: תקשורת טורית מאפשרת לנו לשדר ולקלוט נתונים באופן אמין ופשוט יחסית בין כל סוגי ההתקנים הדיגיטליים.

מאפיינים תקשורת טורית:

- **תקשות סינכרונית** (סיראלית) - המידע מועבר כבית, בית אחרי בית, על גבי קו תקשורת יחיד.
- **א-סינכרונית** - אין שעון משותף, כל התקן פועל לפי שעונו הפנימי.
- **תקשות דו-כיוונית** - יכולה לשדר ולקלוט נתונים.
- **חד ציווני (Simplex)** - שידור רק בכיוון אחד.
- **חצי דו ציווני (Half-Duplex)** - בכל רגע נתון, רק צד אחד משדר והשני קולט ולהפך, זהו האופן הנפוץ ביותר.
- **שידור וקליטה בקצב זהה** (קבוע מראש לדוגמה 9,600 ביט לשנייה).
- **מבנה מוגדר היטב** של מסגרת הנתונים (פרימיטים) - סיביות התחלה, נתונים, סיביות עצירה.
- **דרישות חומרה פשוטות** - בדרך כלל 2 קווים בלבד כיוון (TX ו-RX).
- **תמך חומרה פשוט וסטנדרטי** (למשל 16550 UART במחשבים).

- שימוש נרחב במערכות מובנות (מחשב) ובתקשורת בין מכשירים.
- **פרוטוקולי חיבור** התקני קטן כמו מודמים, מדפסות, מודמים ועוד.
- **פרוטוקולים נפוצים הבנויים מעל UART :** RS-232, RS-422, RS-485 .

מבנה חיבור נתונים בתקשורת טורית:

כאשר מתכוון (למשל מיקרו-בקר) מעוניין לשדר נתונים, הוא בונה חיבור נתונים מבנה מוגדר הכולל מספר שדות :

- **בית התחלה (Start Bit)** - ירידת מ-1' ל-0' לוגי, המסמך את תחילת השידור.
- **נתונים** - המידע עצמו במסגרת של ביטים (בדרך כלל 8-5 ביטים).
- **בית זוגיות (Parity Bit)** - אופציונלי לבדיקת שגיאות.
- **בית עצירה (Stop Bit)** 1 לוגי, המסמך את סוף השידור.

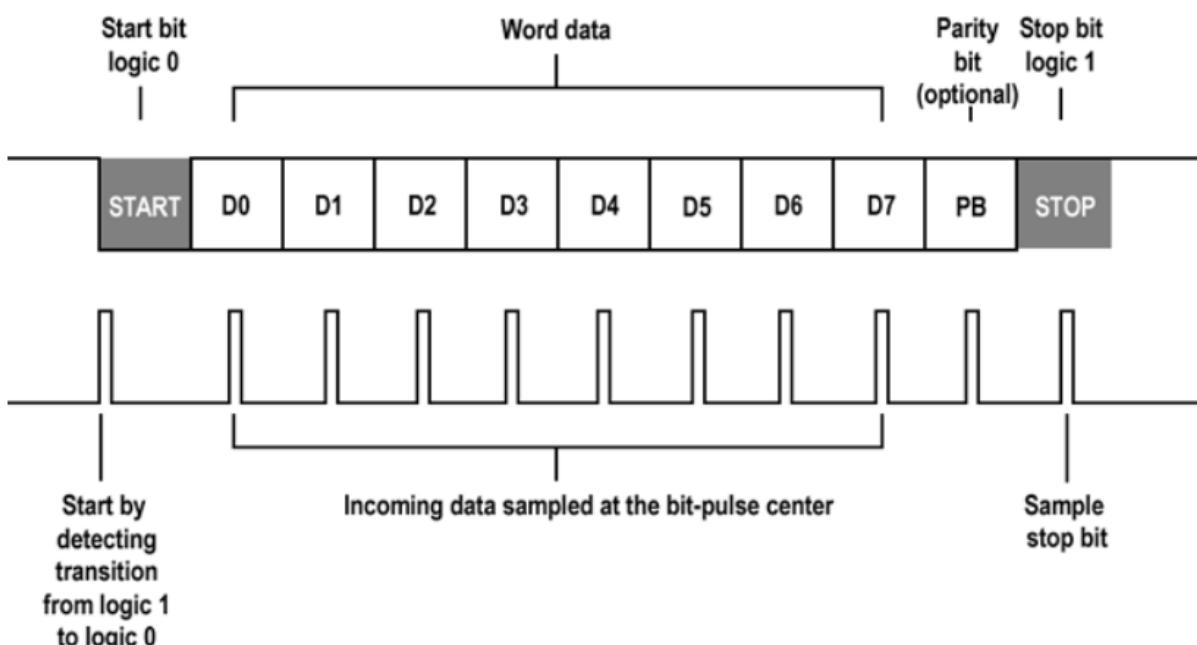
דוגמה:

במידה ונרצה לשЛОוח את התו F (המספר 63 בעשרוני) ייבנה מסר באורך 8 ביטים :

Start Bit (0) → 00111111 → Stop Bit(1)

כלומר, **בית התחלה אחד (0)**, אחריו (מימין לשמאל) **הביטים של הנתון (11111111)**, ולבסוף **בית עצירה אחד (1)**.

בין כל שידור של חיבור כזו יש מרווה זמן קצר לא שידור, כך שהצד מקבל יוכל לזהות היטב את בית ההתחלה הבא ובכך את תחילת הנתונים החדשים.



.**Parity bit (optional)** : בית זוגיות (אופציונלי).

.**Start bit logic 0** : בית התחלה.

.**Stop bit logic 1** : בית סיום.

.**Word Data** : המuid שאנו שולחים.

- תחילת שידור – מעבר מהמתה גובה 0 לוגי למתח הנמוך (0 לוגי) שמסמן את תחילת העברת המידע.**
- נתונים מגיעים בפולסים של ביטים.**
- סיום שידור.**

בית זוגיות (Parity Bit):

beit zogiyot meshem loziohi shgaiot basisi b'tekshoret torriah. hoa mosif b'it nosaf l'masgerat hantoinim, kach shmasper habitiim um urk 1 yihya tamid zogi (zogiyot zogiyah) au ai-zogi (zogiyot ai-zogiyah). am b'kliyut hamidu masper habitiim um urk 1 la mataim l'sog hzogiyot shnabu, haklal kol lehnih shairura shgaiyah b'shidur. um zat, b'it zogiyot la maafshar ltakan shgaiot ao lozohot shgaiot morobot b'itiim, l'kun hoa shiite basisit belbad looyidaa shlemot hantoinim.

תקשורת טוריית ב-ESP32:

ה-ESP32-בניגוד לארדואינו אונו, מצויד במספר יחידות תקשורת טוריית (UART). בדרך כלל, יש לו שלושה ממשיки UART, מה שהופך אותו למתאים במיוחד לפרויקטים הדורשים חיבור למספר התקנים חיצוניים כמו מודולי GPS, חיישני טמפרטורה, מסכי LCD ועוד.

הבדלים העיקריים בין תקשורת טוריית בארדואינו ל-ESP32:

- מספר יחידות UART : בעוד שלארדואינו אונו יש רק יחידת UART אחת (בserieות 0 ו- 1), ל- ESP32 יש שלושה ממשייק UART (UART0, UART1, UART2) זה מאפשר לפתח פרויקטים מורכבים יותר המשתמשים במספר רב של מכשירים הפעילים בשיטה טוריית.
- חופש בבחירה הפינים : ה- ESP32 מאפשר לתוכנת לבחור באופן חופשי את הפינים (GPIO) שימשו לכל יחידת UART, בזכות יכולת מולטייפקסינג הפינים. המשמעות היא שניתן להגדיר את פיני ה- TX ו- RX של כל UART לכל פין GPIO פניו בלוח, דבר המעניין גמישות רבה בתכנון החומרה.
- פונקציונליות : כל יחידת UART ב- ESP32 היא עצמאית לוחוטין וכוללת מאגרים (buffers) גדולים לקליטה ושידור. זה מבטיח שהמידע לא יאבז גם כאשר גם יש עומס גדול של נתונים, ומאפשר שימוש ב- UART0 (שמוחבר גם ל- USB) לצורך ניפוי באגים (debug) בתוכנה, תוך כדי שימוש ב- UART1 ו- UART2 לצורך תקשורת עם רכיבים אחרים.

קצב שליחת נתונים:

אנו מגדירים את קצב שליחת הנתונים בתקשורת טוריית בפקודה אחת בלבד בפונקציה void Setup באמצעות הפקודה :

Serial.begin(115200);

רשימת קצבי השידור הנפוצים בתקשורת טוירית (UART):

• 300 בית לשניה
• 600 בית לשניה
• 1,200 בית לשניה
• 2,400 בית לשניה
• 4,800 בית לשניה
• 9,600 בית לשניה
• 19,200 בית לשניה
• 38,400 בית לשניה
• 57,600 בית לשניה
• 115,200 בית לשניה
• 230,400 בית לשניה
• 460,800 בית לשניה
• 921,600 בית לשניה

ניתן לראות שיש מגוון רחב של קצבי שידור, החל מ-300 בית לשניה במערכות ישנות יותר, ועד למיליאוני ביטים לשניה במערכות טויריות מהיריות יותר.

הकצבים הנפוצים ביותר הם: 9,600 בית לשניה ו-115,200 בית לשניה.

קצב השידור קבוע מראש ומהיב להיות זהה בין שני ה התקנים המתקשרים כדי לאפשר קליטה נכונה של הנתונים.

רכיבים שעובדים בתקשורת טוירית:

- **מודולי בלוטוס** - HC-05, HC-06
- **מודולי WiFi** - ESP8266, ESP32
- **מודולי LoRaWAN/LoRa** - לתקשורת לטוחה רחוק
- **מודולי תקשורת סלולרית** - SIM900, SIM800L
- **מודולי GPS** - NEO-6M, UBLOX
- **צגי OLED ו-LCD** טוריים (I2C)
- **חיישני טמפרטורה, לחות, אור ותנוועה**
- **מודולי קוראי RFID/NFC**
- **מודולי מצלמות טויריות**
- **כרטיסי SD וקוראי כרטיסי SD**
- **מדפסות ומפענחים טויריים**

שליחת נתונים בתקשורת טורית:

ניתן לשולח נתונים בתקשורת טורית באמצעות הפקודה :

Serial.print();

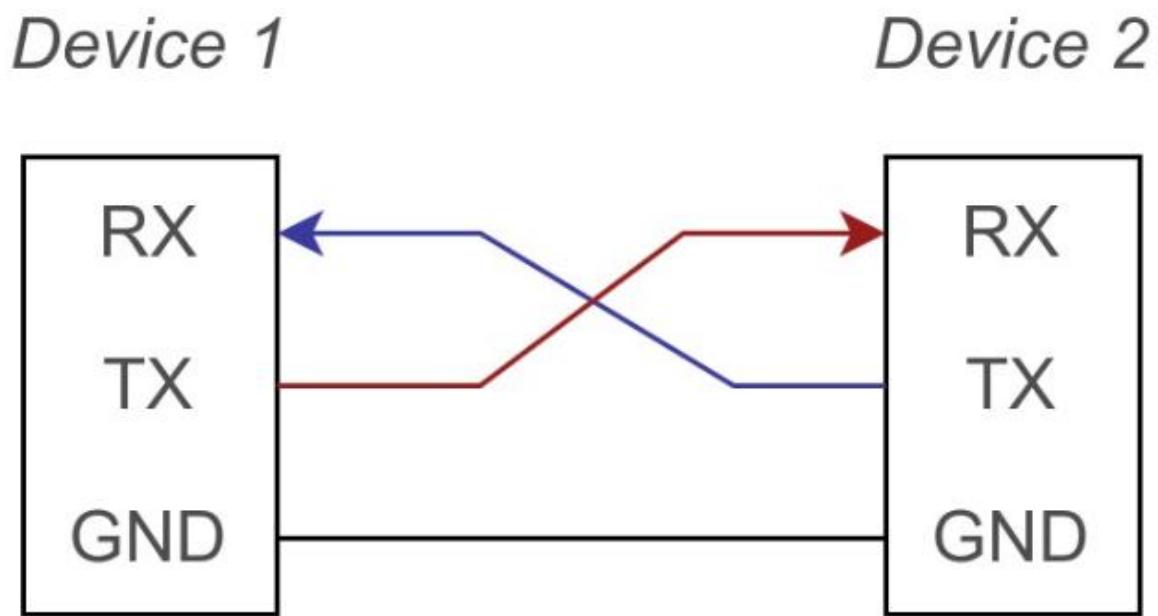
כל מה שנרשם בין הסוגרים () עם מראכות ישלח בתוUr ערך של משתנה. בפקודה זו אין ירידת שורה. במידה ונרצה לרדת שורה אנו צריכים להשתמש בפקודה :

Serial.println();

חיבור בין 2 רכיבים בתקשורת טורית:

בחיבור בין שני התקני UART נדרשים לפחות 3 חוטים - TX של צד אחד מתחבר ל-RX של הצד השני, וחוט האركה משותף (GND).

לעתים נדרש גם חיבור של מתח ההזנה (Vcc) אם אחד ההתקנים מספק כוח לשני. יש לוודא התאמה של רמות המתח בין שני הצדדים (3.3 או 5V).



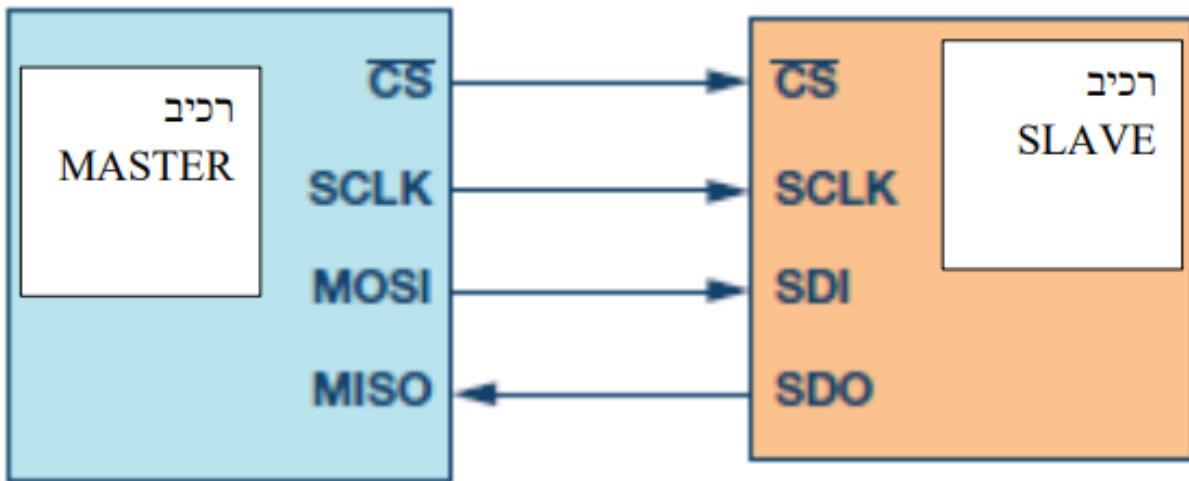
תקשורת SPI

התקשורת נוצרה על ידי חברת מוטורולה ב 1979 עם יציאת המיקרו פרוססור של חברת מוטורולה שנקרא 68000.

השם SPI הוא – Interface Peripheral Serial – ממשק טורי היקפי . זהה תקשורת טורית סינכרונית כי יש בה שעון שמסנכרן את הכניסה/ היציאה הנתונים.

דוגמאות לרכיבי SPI הם מתגים, זיכרון, רכיבי שמע להקלטה/השמעה , ממירים למיניהם (ADC), תצוגות לדיס, TFT ועוד.

באיור הבא מתואר חיבור של תקשורת טורית SPI בין רכיב MASTER ורכיב SLAVE.



באיור רואים שבתקשורת SPI יש 4 קווים:

- .1 : קו הנtauנו מהMASTER (המיקרו בקר) אל העבד (ברכיב העבד השם הוא Master out slave in – MOSI).
- .2 : קו הנtauנו הטורי מהעבד אל המMASTER. ברכיב העבד השם הוא Master in slave out – MISO . (SDO – Serial Data Out).
- .3 : שני אותות MOSI ו- MISO מסונכרנים בעזרת קו השעון הטורי .
- .4 : זהו קו נוסף שדרכו אנו בוחרים את העבד ה- slave , בעזרת קו זה המיקרו בקר מודיע לאיזה עבד הוא פונה. הקו פועל בנמוך – LOW ACTIVE . שמota נוספים לקו him – CS – בבחירה רכיב – Enable , אפסור וועוד .

תהליך התקשורת מתבצעת בשלבים הבאים:

- כאשר ה-MASTER מתחילה תקשורת עם ה-SLAVE הוא מוריד את קו בחירת הרכיב עבד (Select slave) ל-0.
- ה-MASTER שלוח בקו ה- MOSI בית אחרי בית, כאשר כל בית מסונכרן בעזרת פולס שעון – SCLK – שמייצר ה- MASTER לתוך ה- SLAVE .
- הביטים הנשלחים נמצאים ברגיסטר הזהה הנמצא בתוך ה- SLAVE .
- הסyncronו לתוך ה- SLAVE יכול להיות בעליית פולס שעון או בירידה שלו.

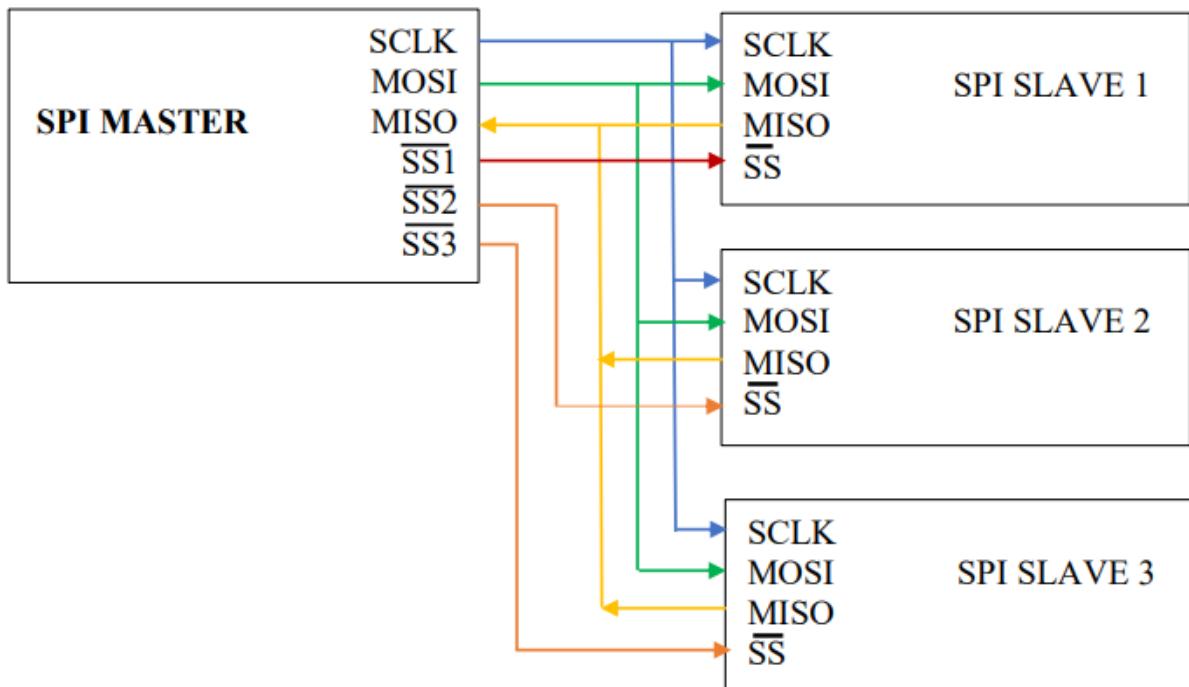
- תוך כדי שליחת כל ביט מה- SLAVE גם ה- MASTER מוציא בית אחריו בית נתון אל קו ה- MISO.
- גם הביטים מה- SLAVE מסונכרנים בעזרת פולסי השעון ב- SCLK והם נכנסים לרגיסטר הזהה ב- MASTER.
- בסיום שליחת הביטים ה- MASTER מעלה את קו בחירת רכיב עבד (Select slave) ל- 1 ומסיים תקשורת.
- בסיום התקשרות ה- MASTER לא מייצר פולסי שעון, והקו של פולסי השעון SCLK נמצא במצב IDLE מצב סרף. הוא יכול להיות ב- 0 או ב- 1.

חיבור מספר SLAVES אל ה- MASTER

ניתן לחבר ל- SLAVES אחד מספר MASTER, מה שיקרא למעשה שהקווים המשותפים לכל ה- SLAVES הם SLCK, MISO, MOSI. קו שנבדל בין כולם הוא הקו לבחירת רכיב העבד (Slave select).

כל ה- SLAVES מקבלים במקביל את פולסי השעון ואת קווי ה- MOSI ו- MISO. קו ה- Slave select כל רכיב יהיה ב- 1 (כלומר הרכיב לא נבחר) והוא MASTER יוריד את קו בחירת רכיב העבד ל- 0 רק עבור אותו ה- SLAVE שהוא רוצה לתקשר אליו.

נראה את האיור הבא שמחיש לנו חיבור של 3 SLAVES ל- MASTER :



חיבור של MATER אחד לשוליה SLAVES שונים דרך תקשורת טורית SPI.

באיור זה נוכל לראות שלכל עבד יש קו (Slave select) משלו. ל- MASTER יש אותו מספר קוויים בדומה למספר ה- SLAVES. לכל רכיב עבד יתחבר קו מסויר אחר, באյור הם נקראים :

(Slave select 1, Slave select 2, Slave select 3)

הס מסומנים עם קו מעליים כדי להציג שהם פעילים בנמוך (Active Low). דוגמה אם ה- MASTER יחולט לתקשר רק עם רכיב 1 SPI SLAVE הוא יוריד את קו (Slave select 1) ל- 0 ואז רק רכיב העבד הראשון ידע שהוא MASTER מתקשר אליו. שאר העבדים יודעים שהמייקרו בקר אינו מתקשר איתם.

כמוות ה- SLAVES שנitinן לחבר אל ה- MASTER תלויים בכמות הבדיקות הפנויים שיש לו במערכת בה הוא נמצא ובקשר של דחיפה הזרם שלו. אפשרויות מתקדמות יותר על מנת לחסוך בחדקים של המיקרו בקר הן להתחבר אל הדקי Slave select של כל רכיב SPI בעזרת מנגנון או מפלג DEMUX עם כניסה אחת, 3 הדקי בחרה ו- 8 יציאות שונות.

אפשרויות העבודה עם תקשורת SPI:

במערכת ה- SPI (גם ה- MASTER וגם ה- SLAVE) יש 2 רגיסטרים, האחד הוא הרגיסטר הזזה המקבל את הדגימות ומזיז את הנטוון, עוד רגיסטר נתון שבסיום העברת הנתון שהתקבל נמצא בו.

בՓולס שעון יש 2 מעברים (מגבוהה לנמוך ולהפך) הכוללים גם עלייה וגם ירידה ויש לעשות 2 דברים :

- .א. במעבר מ-0 ל-1 גם המסטר ו גם העבד מוצאים את בית הנטוון לקו הנתון (I_S MOSI בMASTER MISO בעבד).
- .ב. במעבר השני מתבצעת הזזה של הנתון ברגיסטר ההזזה שנמצא גם בMASTER וגם בעבד.

שני פרמטרים/מאפיינים חשובים בתקשורת SPI:

- .1 CPOL – קוטביות השעון הקובעת מה מצב הקו (נמוך או גבוה) במצב סריך – IDLE כאשר אין פולסי שעון (לפני התחלת תקשורת ובסוף), כאשר CPOL=0 בקו יש 0 לוגי וכשהר CPOL=1 בקו יש 1 לוגי.
- .2 CPHA – פאוזת השעון הקובעת באיזה מעבר (האם מ-0 ל-1 או מ-1 ל-0) יוצא בית הנטוון בקו ה- MOSI ובקו ה- MISO ובאיזה מעבר הוא מזוז ברגיסטר הזזה גם בMASTER וגם בעבד. פרמטר נוסף חשוב הוא באיזה מצב נמצא קו השעון SCLK בסיום התקשורת (האם ב 0 או ב 1).

מבנה העברת של נתון עבור 0=CPHA:

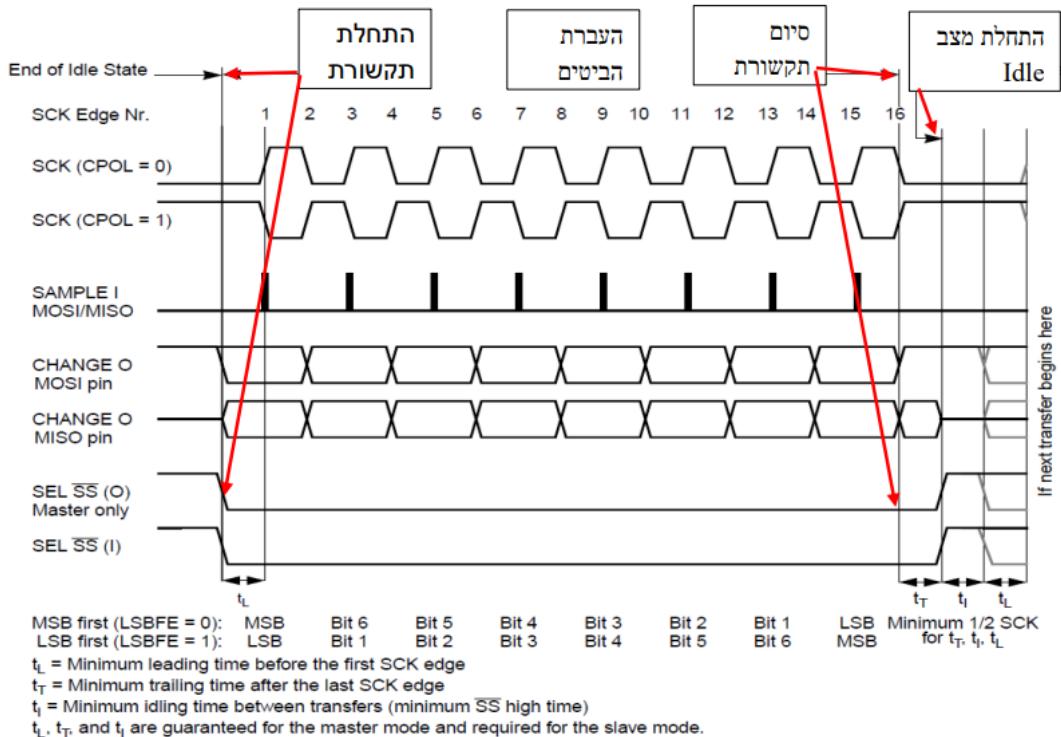
הקו Slave select פעיל בנמוך, ניתן לרשום אותו גם כך : NSS כדי שלא יהיה צורך לרשום גם SS. האירור הבא מתאר את תבנית העברת עם צורות הגל של תקשורת SPI כאשר CPHA=0 ועם פרמטרים של זמן אופיניים.

בחלק התחתיו רואים את התחלת התקשורת כאשר קו NSS (Slave select) יורד ל-0 ואת סיומו כאשר קו זה עולה ל-1.

קו השעון SCK מתואר על ידי 2 צורות גלים . העליונה ביותר מתארת את פולסי השעון כאשר CPOL = 0 (כאשר אין תקשורת יש 0 בקו פולסי השעון) והפולס הראשון הוא עלייה וצורת הגל שמתחתייה כאשר CPOL = 1 הוא 1 לוגי בקו פולסי השעון והפולס הראשון הוא ירידה. הראשון הוא ירידה.

במקומות הרושים באירור O הכוונה ל- Output (יציאה) ובמקומות שרשום I הכוונה ל- Input (כניסה). הביטים של הנתון הטורי בהדק הבדיקה MOSI ו- MISO נדגמים ומתבצעת הזזה ברגיסטרים של ההזמנה באחת מ- 2 אפשרויות. או בעליית השעון או בירידת פולסי השעון ואת זה נקבע לפי נתוני הרכיב שאליו מתחברים.

איור: צורת גל בתקשות SPI כאשר $CPHA = 0$.



המעבר הראשון של קו SCK (מ-0 ל-1 או מ-1 ל-0 תלוי ב-CPOL) משמש להכנסת בית הנטוון הראשון של העבד אל המスター (בקו ה-MISO) ואת בית הנטוון הראשון של המスター אל העבד (בקו ה-MOSI). באירור ניתן לראותו במרכזו את הקו שנקרא SEMPLE והוא משמש ככניסה גם למスター בקו ה-MISO וגם לעבד בקו ה-MOSI.

במעבר השני, בחצי המחוור הבא, מופיע מעבר נוסף בקו SCK ואוז הערך שנಡג מהעבד בקו MISO נכנס לתוך רגיסטר ההזזה שבMASTER. אחרי המעבר הזה משודר הביט הבא של המスター על קו ה-MOSI התהיליך נמשך עד 16 מעברים בקו SCK כאשר נטען **נעל אל המスター במעברים איזוגיים ומועבר אל העבד במעברים זוגיים**.

אחרי מעבר מס' 16 הנטוון יהיה ברגיסטר ה-SPI של המスター צריך להיות ברגיסטר הנטוון של העבד והנתוון יהיה ברגיסטר הנטוון של העבד צריך להיות בMASTER.

באיור מופיעים זמינים בגוון:

- t_L המראה מה הזמן המינימאלי מרגע הורדת קו NSS (Slave select) ל-0 ועד להתחלה פולסי השעון.
- t_T הוא הזמן המינימאלי בין פולס השעון האחרון והעלאת קו NSS (Slave select) ל-0.
- T_i הוא זמן הסרק-IDLE – בין העלאת קו NSS (Slave select) ל-1 ועד שידור נתון חדש על ידי הורדת NSS (Slave select) ל-0.

מתוחת לצורות הגלים ניתן לראות את הנטוון מביט ה-LSB אל ה-MSB או להיפך.

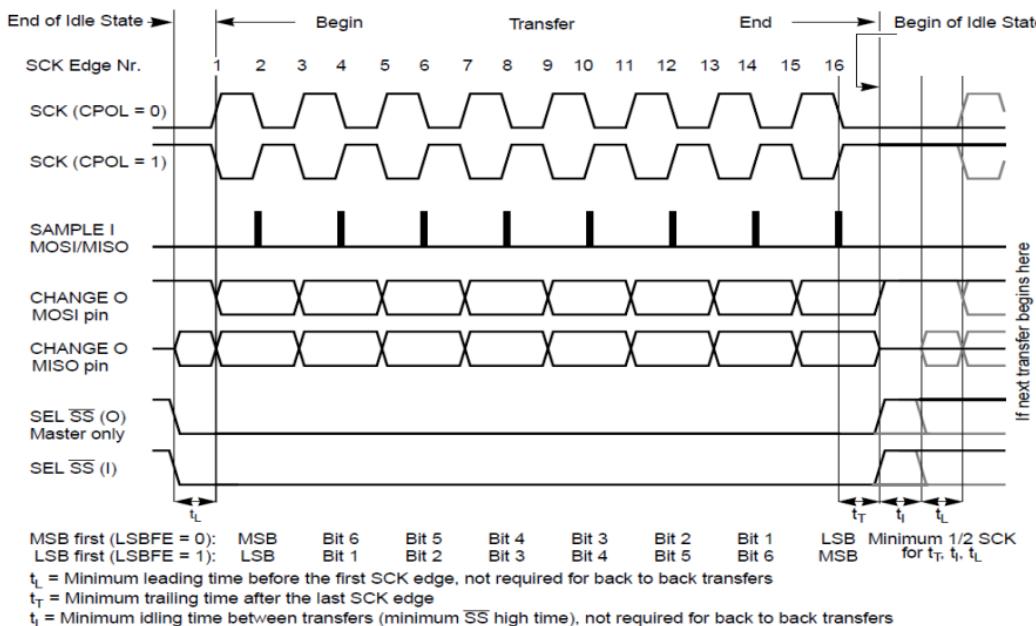
קיימים ביט LSB First Enable (LSBFE) אשר LSB First Enable (LSBFE) שקבע מי נכנס ראשון. כאשר הוא 0 נכנס ביט ה-MSB ראשון וכאשר הוא 1 נכנס ה-LSB ראשון.

קיבולת הנטוון מתבצעת בשני שלבים. היא מזוזת לרגיסטר ההזזה של ה-SPI בזמן ההעברה ומועברת לרגיסטר הנטוון של ה-SPI כאשר הבית האחרון הווז פנימה.

תבנית העברת של נתון עבור 1 : CHPA

קייםים רכיבים שצרכיהם את המעבר של פולס השעון הראשון לפני שניתן יהיה לגשת לביט הנתון הראשוני בכו היציאה של הנתון. המעבר השני מכניס את הנתון למערכת. פורמט זה מתקבל על ידי השמה של A-L-CPHA.

האיור הבא מ_tAר את התקשורות עבור 1 .CPHA =

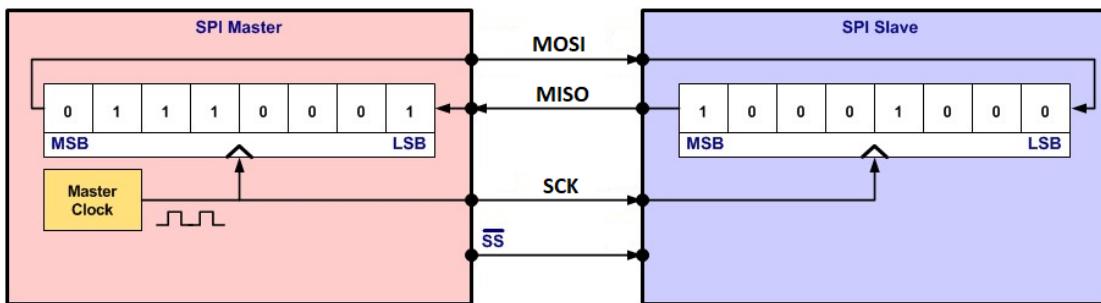


באיור רואים שהמעבר הראשוני משמש כהשיות סync'רו וואמר לעבוד להוציא נתון בכו ה- MISO. בחצי המחוור הבא, במעבר השני, יש געילה של בית הנתון גם בMASTER וגם בעבד. כאשר מגיע המעבר השלישי מועבר הביט שננעל במעבר השני לתוך ה- MSB או ה- LSB של רגיסטר החזזה (כתלות בבית LSBFE).

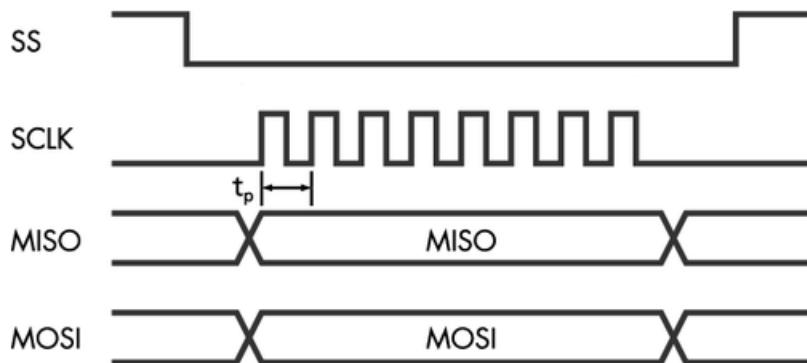
אחרי מעבר זה יוצא בית הנתון הבא של המסטר בכו ה- MOSI. התהליך חוזר על עצמו עבור כל 16 מעברים, כאשר הנתון ננעל במערכות הזוגיים וההזזה קוראת במערכות האי זוגיים.

המידע מועבר בו זמינות בשני הכוונים בכל עליית שעון באמצעות אוגרי הזזה .

בדוגמה הבאה: לאחר 8 מחזורי שעון מידע מוחלף בין ה- Master ל- Slave .



שליחת מילת בקרה של 8 סיביות.



בתחילת התקשרות, הדק SS יורדת ל- 0 . לאחר מכן מועבר המידע מה- Master דרך הדק מה- Slave . Master יכול לקרוא את המידע מה- Slave דרך הדק MISO . בסיום התקשרות הדק SS חוזרת ל- 1 lagi .

t_p – זמן מחזור של השעון .

תקשורת I2C

הקדמה:

פרוטוקול I2C הוא שיטת תקשורת סריאלית נפוצה בין התקני אלקטרונייקה וחישנים. הוא מאפשר העברת נתונים בין מספר התקנים, כאשר יתרונו הגדול הוא שהוא דורש רק שני חוטים לתקשורת – החוט לאותות השעון והחוט לאות הנתונים.

מהו פרוטוקול I2C:

פרוטוקול I2C (אי-שטיים-סי) הוא שיטה סריאלית להעברת נתונים בין התקנים, שפותחה על ידי חברת פיליפס בשנת 1980.

בניגוד לתקשורת UART שדורשת שני חוטי תקשורת לכל כיוון, ב-I2C יש רק שני חוטים משותפים לשני הכוונים: SDA לנtones ו-SCL לאות שעון. שני חוטים אלו מחוברים לכל החתקנים בטופולוגיה של BUS.

כל התקן ברשות I2C מזוהה בכמות ייחודית. כאשר מתokin אחד רוצה לתקשר, הוא שולח הודעה עם הכמות של המתokin היחיד. רק המתokin עם אותה כמות "יורם את השפופרת".

ההעברה הנתונים עצמה נעשית לפי עקרון מסטר/עבד (לרוב מסטר עובד), מפעיל את אות השעון ושולט מתי לשדר ומתי לקבל נתונים, ואילו המתקנים המשניים הם עובדים פסיביים.

ב-I2C כל בית נשלח ברכז אחורי הבית הקודם, והוא תוקף כל עוד אות השעון SCL גבוהה. ירידת SCL מהווה אינטואיטיבית לבית חדש.

יתרונותיו הגדולים של פרוטוקול I2C הן הפשטות והחיסכון בכמות החוטים לתקשורת, ניתן לחבר עשרות מתקנים קצה, וכל יחסית ליישום בマイקו-בקרים.

לכן זהו פרוטוקול נפוץ לתקשורת עם אלקטרונייקה כמו חישנים, מסכי LCD, גרפים, זיכרונות וכו'.

מאפיינים פרוטוקול I2C:

- **תקשורת סיידרטייט (סריאלית)** - המידע מעובר בטור, בית אחורי בית, על גבי קו התקשורת SDA.
- **סינכרוניות** - אות השעון SCL מסנכרון בין כל המכשירים.
- **תקשורת דו כיוונית** בין מסטר למספר סלייבים.
- **שני קווי תקשורת** משותפים לכל המכשירים SDA ו-SCL.
- **קצב תקשורת** גבוה יחסית - עד 5 מגה ואף יותר.
- **כבות מוגדרות** מראש לכל רכיב (10/7 בית כמות).
- **ממשק חומרה פשוט וסטנדרטי.**
- **מתאים במיוחד** לתקשורת בין מעבד להתקני קצה כמו חישנים, מסכיים וזיכרון.
- **מצמצם מאוד** בכמות החיווט לעומת UART.

מבנה חיבור (מסגרת) נתונים בתקשורת טורית:

כאשר מתקן מסטר (כמו מיקרו-בקר) מעוניין לשדר נתונים למתקן SLAVE, הוא בונה את החיבור במבנה מוגדר הכלל מספר שדות:

- **התחלת העברה (START)** - ירידה באות SDA מ-1 ל-0 בזמן שאות SCL נשאר גבוה, מה שמהווה "אות התחלת".
- **כתובת SLAVE** - 7 או 10 ביטים עם הכתובת הייחודית של המתקן SLAVE היעד. רק אותו מתקן ייריב את השופורת.
- **נתונים** - המידע עצמו בסדרת ביטים הנשלחים על ידי המאסטר או שמתקבלים מהמתקן SLAVE.
- **אישור 9 (ACK)** - פעולות שעון שבוחן ה-SLAVE מאשר קיבל תקינה של העברת 8 ביטים בקו SDA.
- **סיום העברה (STOP)** - עלייה באות SDA מ-0 ל-1 בזמן שקו SCL נשאר גבוה, מה שמהווה "אות סיום".

דוגמה: שליחת בית ב프וטוקול I2C:

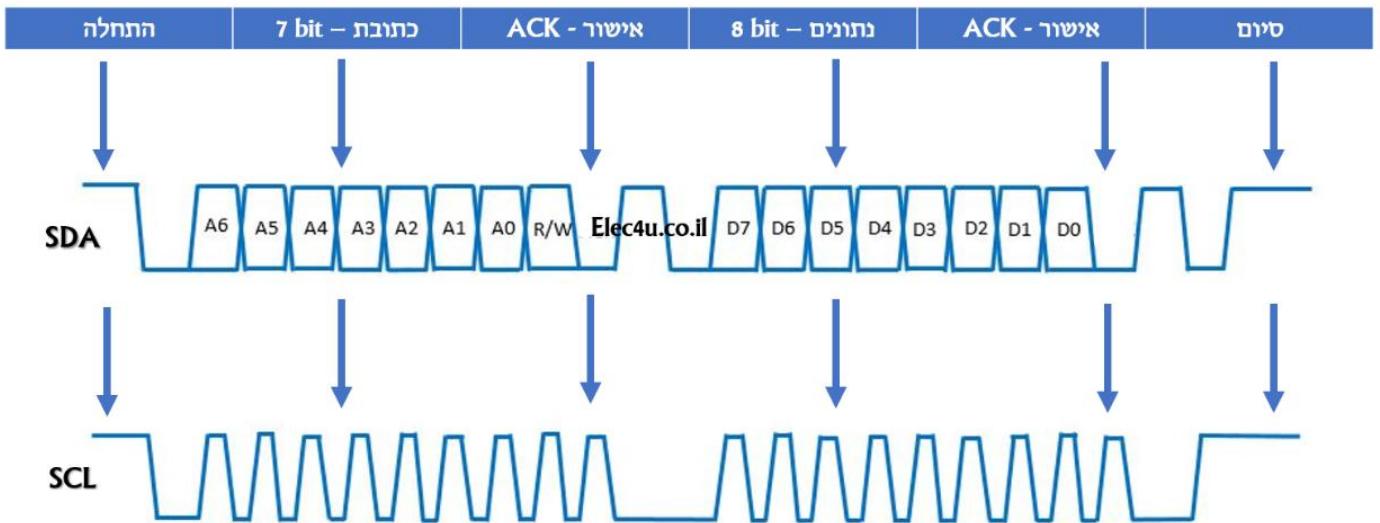
הנה פירוט דוגמה לשילוח הביטי $0x2F$ (מספר 47 עשרוני) למתקן בכתובת $0x38$ (מספר 56 עשרוני):

- $00101111 \rightarrow ACK \rightarrow STOP \rightarrow 0x38$ (כתובת) $\rightarrow START \rightarrow 0111000$

פירוט שלבי התקשורת:

- **START** : מצין על התחלת העברה חדשה. נוצר על ידי ירידה של קו הנתונים SDA מ-1 ל-0, בעוד קו השעון SCL נשאר גבוה.
- **0x38** (כתובת) : שבעת הביטים הבאים מצינים את כתובת ה-SLAVE (במקרה זה 38 או 56 עשרוני). הם נשלחים מהביט הנמוך (LSB) לביט הגבוה ביותר (MSB).
- **STOP** : שמונה הביטים הבאים מצינים את הנתונים שאנו שולחים ל-SLAVE (במקרה זה 47 עשרוני או $0x2F$). שוב, מ-LSB ל-MSB.
- **ACK** : לאחר הנתונים, ה-SLAVE חייב לשЛОוח בית "אישור" אחד בחזרה לMASTER כדי לאשר שקלט את הנתונים כראוי.
- **STOP** : סיום תנועת ה-I2C. נעשה על ידי העלאת קו הנתונים SDA בחזרה מ-0 ל-1, בעוד קו השעון SCL נשאר גבוה.

נתבונן באיור הבא:



בחבילת נתונים של פרוטוקול I₂C ישנה אפשרות לשЛОח מסגרת אחת או מספר מסגרות ברכז. באוטה החבילה. מסגרת נתונים אחת מכילה את כמות המידע שאנו רוצים לשЛОח באותו הרגע. הפרוטוקול מאפשר לנו לשLOB מספר מסגרות נתונים ברכז בתוך אותה החבילה, ללא צורך לשLOW חדשה עבור כל מסגרת. זאת על ידי שליחת אישור (ACK) מהתקן מקבל בין מסגרת למסגרת. כך ניתן לשLOW ביעילות נתונים ארוכים יותר באותו החבילה I₂C.

יתרונות מרכזיים של פרוטוקול I₂C :

- **חסכוני בכמות החיווט** - משתמשים רק בשני קווי תקשורת משותפים לכל התקנים - קו SDA לנ נתונים וקו SCL לאוט השעון.
- **תקשרות דו-כיוונית** - מאפשר הרן למאסטר לשLOW נתונים והן ל-SLAVE להחזיר נתונים.
- **פשוט לחיבור התקני קטן ורבים** - ניתן לחבר עשרות התקנים שונים על אותה רשת I₂C.
- **ממשק תקשורת פשוט וסטנדרטי** - כל רכיבי החומרה מתוכננים לתמוך בפרוטוקול.
- **מהירות תקשורת גבוהה** - עד 5 מגה-הרץ, מותאים להעברת נתונים מחישנים, שליטה ועוד.
- **אמינות גבוהה** - שימוש ב-ACK-ו מנגנון זיהוי שגיאות.
- **נתבות מוגדרות מראש** - מאפשר תקשורת מכוונת בין התקנים הרשא.

כמויות רכיבים שנייתן לחבר לפרטוקול I2C:

בפרטוקול I2C ניתן לחבר מספר רב של **רכיבים**, אך ישנה הגבלה על הכמות המקסימלית. בפועל, ניתן לחבר עד 128 **רכיבי קצה (SLAVES)** לכל רשות I2C. הסיבה להגבלת 128 היא **שימוש ב-7 ביטים לייצוג כתובת הריבב בפרטוקול**, מה שמאפשר טווח כתובות של 0 עד 127 (כלומר 128 כתובות שונות). כמות זו של 128 רכיבים נחשבת כבואה יחסית ומספריקה לרוב המערכות והיישומים הנוכחיים.

הרחבה ל-10 ביטים:

ישנה אפשרות להשתמש גם בכתובות עם 10 ביטים, מה שמרחיב את טווח הכתובות ומאפשר חיבור של מספר רב יותר של רכיבים. עם זאת, השימוש ב-10 ביטים מורכב יותר ופחות נפוץ.

הגבלות נוספות:

מלבד מגבלת הכתובות, קיימות גם **מגבליות פיזיות** שאלוולות להשפעה על כמות הרכיבים המקסימלית שנייתן לחבר בפועל. מגבלות אלו כוללות:

- **קיבוליות (Capacitance)**: ככל שמוסיפים יותר רכיבים, כך גדלה הקיבוליות על קווי התקשורת SCL ו-SDA. קיבוליות גבוהה מדי יכולה לעוות את האותות החשמליים ולשבש את התקורת.
- **אורך החוטים**: ככל שהחוטים ארוכים יותר, כך גדלים הסיכון לרעש חשמלי ולירידת מתח.
- **זיהום סביבתי**: סבيبة רועשת מבינה חשמלית עלולה להשפיע על האותות.

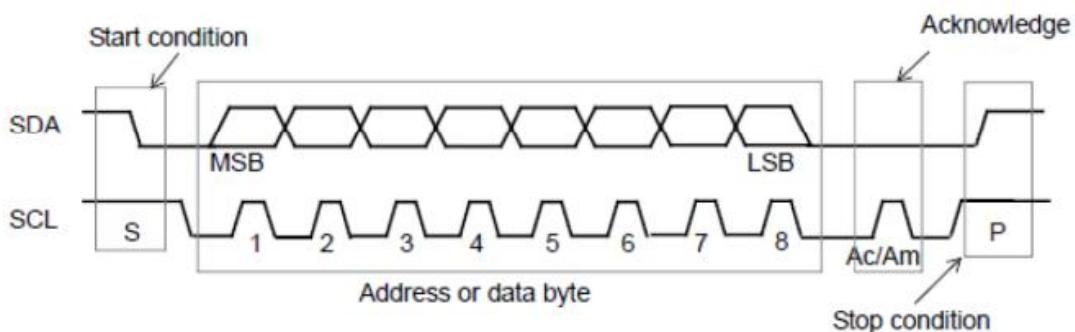
בפועל, למروת התיאוריה של 128 כתובות, יש לנקוט בחשבון את המגבליות הללו, ולרוב רשותות I2C פשוטות מוגבלות למספר קטן יותר של רכיבים.

פעולות בסיסיות בפרטוקול התקורת I2C:

כאשר שני האותות SDA ו- SCL, נמצאים במצב 1 לוגי ה- BUS במנוחה. השידור עצמו נעשה בשיטת MSB – First.

- האות SDA חייב להיות יציב כשהאות SCL במצב 1 לוגי.
- האות SDA יכול להשתנות רק כשהאות SCL במצב 0 לוגי.

נראה תמונה שמתארת העברת מידע בפרטוקול בדומה לתמונה מהעמוד הקודם:



ניתן לראות שפעולות START מתבצעת כאשר SDA בירידת ו- SCL נמצא ב-1 לוגי. לעומת זאת פעולת STOP מתבצעת כאשר SDA בעלייה ו-SCL כבר ב-1 לוגי.

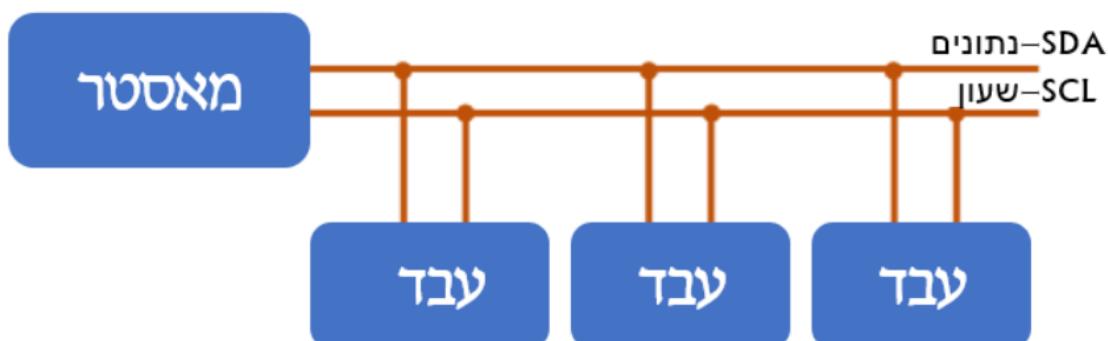
הרכיב שמקבל את המידע (ה-MASTER) או ה-SLAVE מקבלים Acknowledge (SLAVE) מחזקירים (MASTER) תמיד להשיב באוטו Acknowledge לפניהו לכתובות שליהם.

מצבים בהם רכיבי SLAVE לא מגיבים ב- Acknowledge :
רכיבי SLAVE יכולים שלא להגיב בתגובה Acknowledge (כלומר להגיב ב- NACK), במקרים הבאים :

- ה-MASTER עסוק (במקרה כזה הוא יכול גם להאריך את ה-SCL).
- ה-SLAVE קיבל מידע לא חוקי.

לאחר קבלת NACK ה-MASTER חייב לבצע פעולה STOP. כאשר ה-MASTER הוא ה-Receiver הוא יכול שלא להגיב באוטו Acknowledge (כלומר להגיב ב- NACK) כדי לסמן שהגיע ל- Byte הסופי של המידע. מערכות שפועלות ב프וטוקול I2C יכולות לבצע פעולה Byte Acknowledge גם ברמת ה- Byte.

הסביר כיצד ה프וטוקול I2C פועל:



פרוטוקול I2C הוא שיטת תקשורת טורית בין מיקרו-בקר לרכיבי ציוד היקפי, המשמשת בשני קווים בלבד.

- רgel - (SDA) Serial Data אחראית על שליחת וקבלת המידע בין המכניםים.
- רgel - (SCL) Serial Clock אחראית על אותן השעון, המسانדרן את התקשרות בין כל רכיב, כך שהם "מדברים" יחד ובזמן הנכון.

הארדוואינו/ESP32, המכונה "MASTER", והוא שולט על הקווים, והשאר הם ה"עבדים" (SLAVES). כאשר הארדואינו/ESP32 רוצה לתקשר עם אחד החישנים, הוא שולח אליו את הכתובת הייחודית שלו. לדוגמה :

- תצוגת LCD 2X16 עובדת בכתובת 0x27.
- חיישן טמפרטורה LM75 עובד בכתובת 0x07.

מסך TFT (ST7735) – 1.8



זהו המסך בו השתמשתי על מנת להציג את הרadiator בפרויקט שלי. מסך קטן וקומפקטי שיוצר תמונה ברורה וטובה עבור הרadiator. אשר מציג את המרחק החל מ 0 ועד 100 סנטימטרים. בנוסף ניתן לראות את הקו שמשתובב 180 מעלות החל מצידו הימני/הشمالي של המסך ועד לצידו השני ומציע נקודה בהתאם למרחק ככל שהאובייקט קרוב יותר נראה שצבעה של הנקודה הוא אדום, וככל שהאובייקט רחוק יותר הנקודה תהיה בצבע צהוב.

1. סוג המסך:

- דגם : ST7735
- גודל : 1.8 אינץ'.
- רזולוציה : 128×160 פיקסלים.
- צבעים : צבעוני (RGB565).
- ממשק : SPI (Serial Peripheral Interface) מהיר ומדויק.

2. חיבור פיניים (SPI):

המסך מתקשר עם ה- ESP32 דרך SPI. בפועל בקורס שלי :

- MOSI (Master Out Slave In) – GPIO 23 (נתונים מה- ESP32 למסך).
- SCLK (Clock) – GPIO 18 (שעון SPI).
- CS (Chip Select) – GPIO 5 (בחירה התקן SPI פעיל*).
- DC (Data/Command) – GPIO 2 (מספר למסר “פוקודה או נתון”).
- RESET – GPIO 4 (Ấתחול המסך).
- VCC / GND - מתח והארקה.

הערה: SCLK ו-MOSI – משותפים אם יש יותר מסך אחד CS ו- DC - חיבורים להיות שונים לכל מסך.

3. תאורת רקע (Backlight)

- לרוב המטכים יש **LED backlight** פנימי.
- חיבור פיזי: - BL / LED+ / LED .
- אם מחברים ל- 5V → Backlight .
- ניתן לכבות או לשנות בעוצמה דרכן טרנזיסטור PWM / כדי לחסוך צריכה חשמלית.
- כיבוי Backlight חיסכון משמעותי בזרם. (50–300 mA).

4. ספריות ותפקודים בקוד שלבי:

- הספרייה Ucglib שולחת פקודות SPI למסך.
- פונקציות עיקריות בקוד:
 - ucg.begin() - אתחול המסך.
 - setRotate90() - סיבוב המסך לאופקי.
 - setColor(...) - הגדרת צבע לצירוף.
 - drawLine, drawBox, drawDisc, drawCircle - ציור צורות.
 - setFont(...), setPrintPos(...), print(...) - כתיבת טקסט.
 - cls() - ניקוי המסך באמצעות ריבועים שחורים.

5. זרם ומתח

- VCC : רוב המודולים מקבלים 5V חלק תומכים גם ב-3.3V.
- צורך טיפוסית :

 - backlight: 10–60 mA ללאם
 - backlight: 50–300 mA עםם

מה המסך מציג בקוד שלבי:

- רקע גрафי :** עיגולים, קוויים, רקע ירוק-שחור (גרדיינט)
 - טקסט :** שם הפרויקט “Mini Radar”, מרחקים בקנה מידה (...) (25cm, 50cm...).
 - נקודות :** מיקום אובייקטים לפי החישון האולטראשוני (>100cm <100cm, אדום <צהוב).
 - קוויים :** קו סריקה שמסתובב עם הסרוון, כמו קרן רdar.
- בקיצור, המסך הוא **המסך הגרפי של הרdar** – מציג מידע בזמן אמת בצורה צבעונית.

הסביר על תוצאות הרadar:

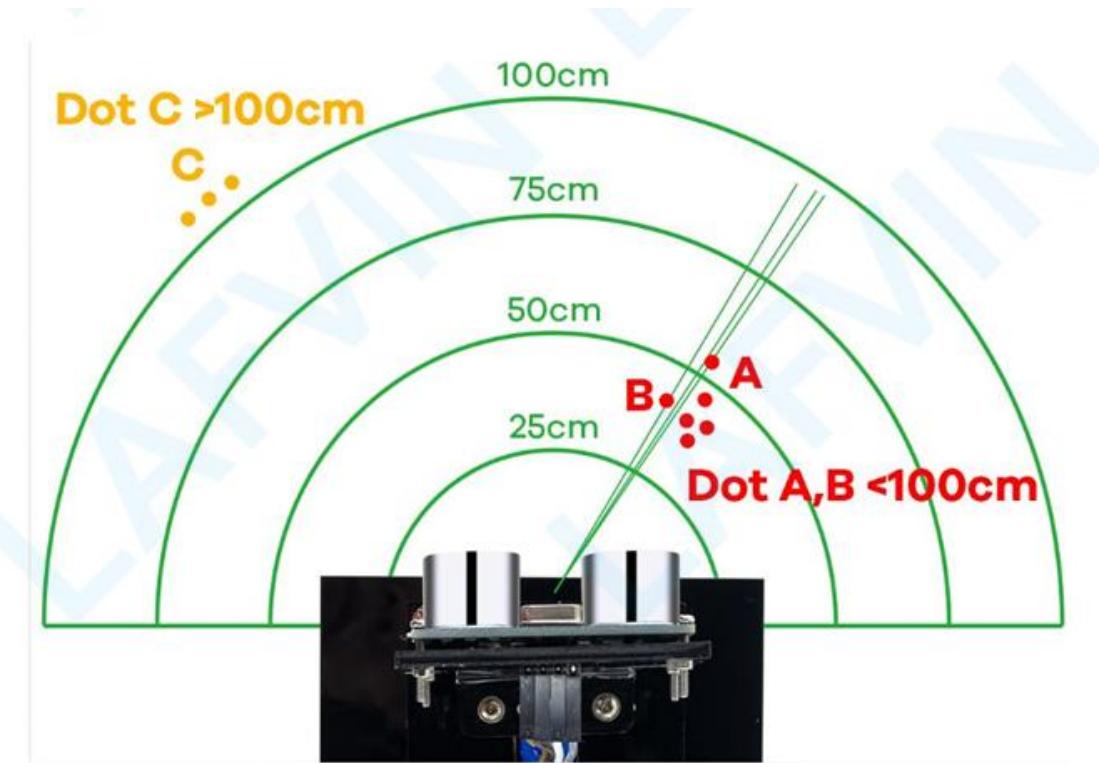
איך למשה המרצת עובדת מה אנחנו רואים על המסך?

אוначיל בזזה שהרוביוט השומר שלנו צריך לשמור על סביבה שנספרת 180 מעלות.

הפעולה של הסריקה עצמה מתבצעת באמצעות שני רכיבים חשובים מאוד במערכת radar: הרכיב הראשוני הוא המנוע סרו שבכלל מאפשר לנו את כל העניין של התזוז של חצי הסיבוב החל מ0 מעלות ועד הגעתו ל180 מעלות. הרכיב השני שלנו שלמעשה מבצע את המידה זה בחישון האולטרא-סוני שמבצע בדיקה של האובייקטים בסביבה בתווך של 100 פלוס סנטימטרים ומחזיר לנו "מיען" שלמעשה מתאפיין בנקודה על radar.

כל המערכת מוצגת באופן חד וברור על המסך שלו ואנו יכולים לדעת האם האובייקט קרוב אלינו או רחוק מיתנו, כאשר האובייקט נמצא בתחום בין 0 ל- 100 סנטימטרים נראה שהנקודה היא קבוע אדום. כאשר האובייקט נמצא בתחום של 100 סנטימטרים או הנקודה תהיה קבוע צהוב.

נראה תמונה שמחישה לנו באופן ברור את מה שתרחש במסך בזמן פעולה מערכת radar:



כמו שאמרתי קודם לכן, ניתן לראות מההמונה שכל עוד אנו נמצאים בתחום בין 0-100 סנטימטרים固定 הנקודות האו אדום וברגע שאנו עברים תחום זה固定 הנקודות הוא צהוב.

מסך גרפי טאץ' TFT 9341

מסך גרפי 9341 TFT מאפשר הצגת תווים, מספרים, צורות גרפיות ותמונות על המסך.

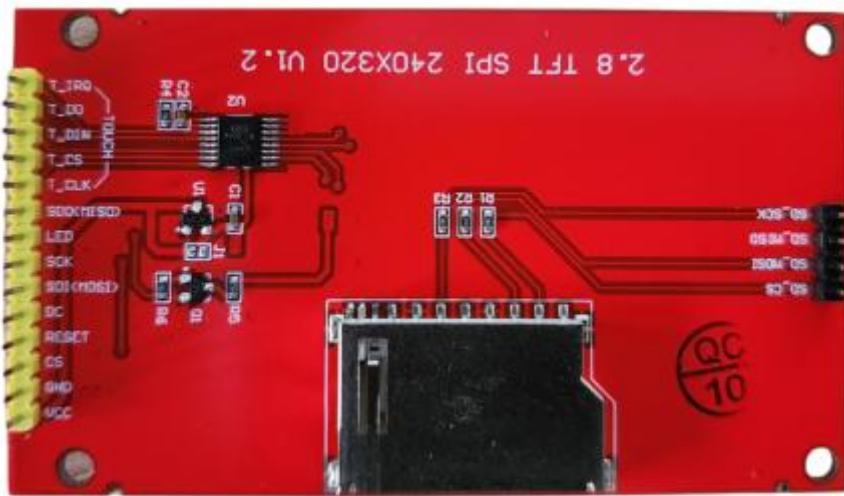


מפורט טכני של מסך גרפי:

- .**1. סוג תצוגה :** מסך גרפי צבוני מגע (Touch).
- .**2. פרוטוקול עבודה :** SPI (Serial Peripheral Interface).
- .**3. גודל פיזי 8 :** אינץ'.
- .**4. רזולוציה 240x320 :** פיקסלים (נקודות).
- .**5. ציר X רוחב :** (מכיל 320 פיקסלים).
- .**6. ציר Y גובה :** (מכיל 240 פיקסלים).
- .**7. מתח הפעלה :** 3V.
- .**8. תכונת אחסון :** תצוגה כוללת קורא כרטיסי SD המאפשר שמירת קבצי תמונה בנפח זיכרון גדול.

הערה : אם תעדיף שהסדר יהיה רוחב ואז גובה (כפי שמקובל לעתים ברזולוציה), ניתן לשנות את סעיפים 5-6 בהתאם לצורה שבה תרצה להציג את זה במסך. הרשימה הנוכחית משקפת את המידע שסבירת, כאשר ציר X הוא 320 וציר Y הוא 240.

תיאור ותפקיד פיני חיבור – מסך TFT/Touch :



ההמשך הגרפי דורש מספר חיבורים מרכזיים כדי לאפשר תקשורת נתונים מהירה (SPI) ובקרה מגע. כל פין משרת מטרת ייחודית:

- 1. VCC מתח אספקה :** זהו פין אספקת המתח החיובי של התצוגה, הנדרש להפעלת הבקר והתאורה האחוריית. עבור מסך זה, **המתח הנדרש הוא 3.3V**.
- 2. GND הארקה :** (חיבור הארקה המשותף, סוגר את המעגל החשמלי).
- 3. CS בחירת רכיב - (Chip Select) :** משמשת לשיליטה על שבב התצוגה (LCD). כאשר הפין זהה במצב נמוך (LOW) לוח הבקרה (כمو ESP32) מסמן לבקר התצוגה (ILI9341) שהוא הרכיב העיקרי לקבל או לשלוח נתונים דרך דרכ קו-ה-SPI המשותף.
- 4. DC נתונים/פקודה (Data/Command) :** – פין קריטי בבקרה. הלוח משתמש בו כדי להבדיל בין סוג המידע הנשלח: אם הפין במצב גבוה (HIGH) המידע הנשלח הוא **נתונים** (צבע פיקסלם, תמונהו); אם הפין במצב נמוך (LOW) המידע הוא **פקודה** (הוראות לבקר לשנות רזולוציה או אתחול).
- 5. SDI – Master Out Slave In (MOSI) :** קו **כניסת הנתונים הראשי** בפרוטוקול SPI. באמצעות פין זה, לוח הבקרה שולח את כל הנתונים הגרפיים ופקודות אל התצוגה.
- 6. CLK שעון - (Clock) :** רgel השעון. מספקת את הקצב והתיזמון הנדרשים לסינכרון תהליך העברת הנתונים ב-SPI. כל פיקסל או בית נשלח בתיאום עם דופק השעון.
- 7. SDO – Master In Slave Out (MISO) :** קו **יציאת הנתונים** מפרוטוקול SPI. דרכ פין זה, לוח הבקרה **מקבל נתונים** חזירים מהמסך (למשל, קריית סטטוס או זיהוי הבקר).
- 8. CS בחירת שבב מגע :** (זהו פין Chip Select נפרד המשמש לשיבב בקר המגע לרוב (XPT2046) הואאפשר ללוח הבקרה לתקשר עם שבב המגע באמצעות רשת SPI, בנפרד מבקר התצוגה הראשי.
- 9. T-IRQ_F – פסיקת מגע Touch Interrupt :** רgel זו נכנסת לפעולה כאשר **נגיעה מזויה** על פני המסך. היא מאפשרת ללוח לעבד אירועי מגע באופן יעיל ומידי (mbased פסיקה), מבלי צורך לבדוק באופן קבוע אם התרחשנו נגעה.

הסבר מקייף על פונקציות התצוגה (Adafruit ILI9341): בה אני משתמש בפרויקט

1. הכללת הספריות (Headers):

בתחילת כל סקיצה (קוד) עליך לכלול את הספריות הבאות, המגדירות את כל הפקודות הגרפיות והחומרתיות:

- `#include <Adafruit_GFX.h>` : זהה ליבת הציור הגרפי (GFX Core). היא מספקת את כל הפקודות הגרפיות לצור צורות, קוים וטקסט, ואני תליה בברcker המסך הגרפי.
- `#include <Adafruit_ILI9341.h>` : זהה הדרייבר הגרפי התומך בברcker הפיזי ILI9341 של המסך שלו. הוא מתרגם את הפקודות הגרפיות של GFX לפקודות בرمאה נמוכה (SPI) שהחומרה מבינה.

2. אתחול ותחילת עבודה:

- פונקציות אלו נראות לרוב רק פעם אחת בתחום פונקציית `setup()`:
- `() tft.begin()` : זהה הפקודה הראשונה והקריטית ביותר. היא שולחת את כל רצף האתחול הנדרש לבקר ILI9341 כדי להביא אותו במצב עבודה וכיוול ראשי. (מקביל ל- `lcd.begin()`)
 - `() tft.setRotation(0)` : כיוון (פקודה זו קובעת את **כיוון התצוגה** (אורינטציה). ישנו ארבעה מצבים אפשריים (0, 1, או 2, או 3), כאשרם יגדיר את המסך לרוחב (320x240) והآخر לאורץ (240x320).
 - `() tft.fillScreen(BLACK)` : צבע (זויה הפקודה **לנקיי המסך**. היא ממלאת את כל שטח התצוגה בצבע אחד, ומשמשת לרוב לקביעת צבע הרקע (למשל, lcd.clear() מחליף `tft.fillScreen(BLACK)` במערכות אחרות).

3. עבודה עם טקסט ומיקום:

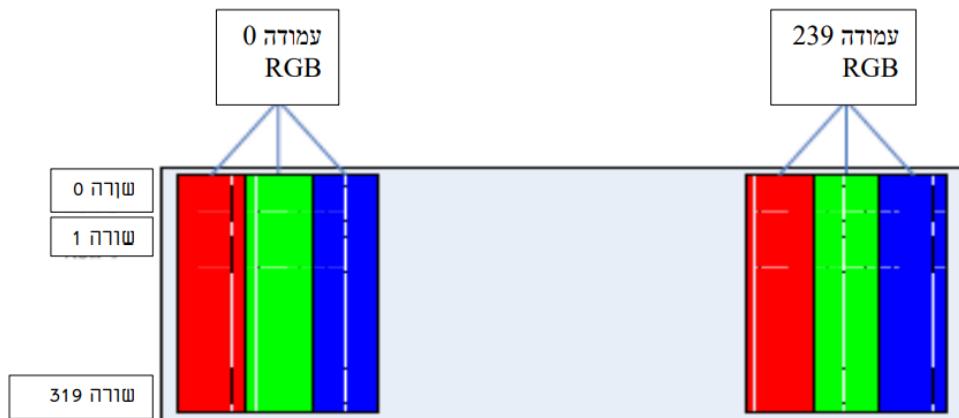
- פונקציות אלו משמשות להציג נתונים מסוימים ומספריים ומילוליים על המסך:
- `(y) tft.setCursor(x, y)` : מגדירה את מיקום הסמן שבו הטקסט הבא יודפס. ה- X וה- Y מציננים את הנקודה השמאלית העליונה של האות הראשונה. (מקביל ל- `lcd.goto(x, y)`)
 - `() tft.setTextSize(size)` : קובעת את גודל ה폰ט. גודל 1 הוא הפוןט הבסיסי, וגדלים גדולים יותר מכפילים את הפיקסלים (לדוגמה, גודל 3 הוא פי 3 מהגודל הבסיסי).
 - `() tft.setTextColor(color)` : צבע (קובעת את **צבע הטקסט** (צבע החזית) שיוצג. ניתן להוסיף ארגומנט שני אופציוניaldi כדי להגדיר את צבע הרקע סביב הטקסט.
 - `() tft.print(str)` : מבצעת את **הדפסת הטקסט, המספרים או המשתנים** אל המסך במקום הסמן הנוכחי (mosaic מוסיפה ירידת שורה). (מקביל ל- `lcd.print(str)`)

4. פונקציות מגע (Touch):

- הפונקציות הללו מטפלות בברcker המגע (ככל הנראה XPT2046 הנמצא במכשיר):
- `() ts.begin()` : אתחול שבב המגע (כאשר ts הוא אובייקט של ספריית המגע הנפרדת, כמו XPT2046_TouchScreen). זהו המקביל לאתחול בקרת המגע.
 - `() ts.getTouch()` : פונקציות ספציפיות בספרייה המגע, (כגון: ts.getTouch()) מאפשרות **lezehot at negiha** ולתקבל את קואורדינטות ה- X וה- Y של נקודת הלחיצה על המסך.

תצוגות - TFT טרנזיסטור סרט זק – הן תצוגות אקטיביות:

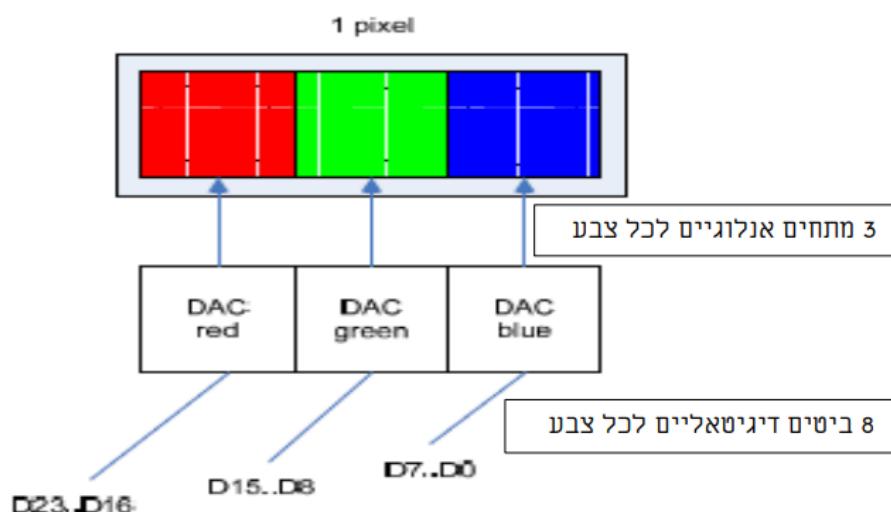
כל סגמנט בתצוגה מאפשר החזרה או העברה של אור. גם כאן יש מקור אור המקרין אוור מאחרוי התצוגה אל החזיות. בעזרת הטרנזיסטור קבועים האור יוקרן החזiosa או שהוא ייחסס ולא יועבר. כדי ליצור צבעים צריך 3 סגמנטטים. 3 סגמנטטים אלו מעבירים אוור דרך מסננת אדומה, יroxה או כחולה. כך נוצר פיקסל RGB. כלומר פיקסל מורכב מ-3 סגמנטטים. למעשה כל פיקסל מורכב מ-2 מסננות עם קיטוב, 3 מסננות צבע ו-2 שכבות יישור וכך נקבע בדיקוק כמה אוור עברו ובאיזה צבע. באирו רואים את סידור העמודות והשורות של תצוגת RGB של 320 שורות ו-240 עמודות.



צירת צבע בתצוגה TFT (Thin Film Transistor)

تצוגות TFT יכולות לדוחוף 3 סגמנטטים (פיקסל אחד) בפולס שעון אחד עם שדה חשמלי משתנה חזק. גודל המתח האנלוגי בכל צבע קובע את עצמת הצלב המסוים. תצוגה כזו תומכת בצבעים רבים. רמות הצלב נקבעות על ידי מספר קווי הנטוון בפנל התצוגה ובמספר קווי יציאות הנטוון של הבקר. האפשרויות הן: 24 בית לפיקסל (bits per pixel), 15bpp, 16bpp, 18bpp, 8bpp. במשך מקבילי נדרשים 320 פולסי שעון ל-320 פיקסלים.

ניתן לראות בתמונה מושך של 24 ביטים לפיקסל בשעון אחד:

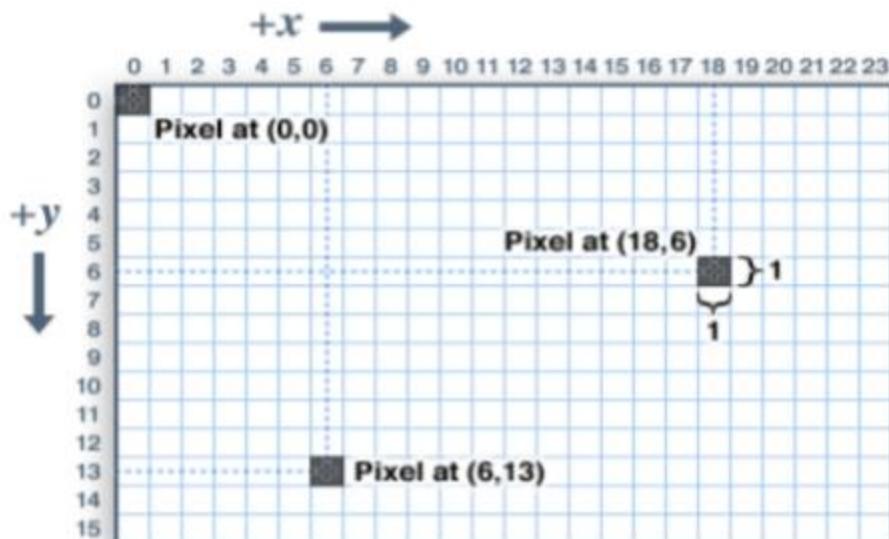


תמונה זו מותארת תצוגה המתחברת בMUX של 24 ביט.

כל 8 ביטים מתחברים ל DAC הקובע את גודל המתח שיגיע לכל צבע בפיקסל מסוים. הצלע הכחול נשלט על ידי 8 הביטים , D0~D7 הצלע הירוק על ידי הביטים D8~D15 והצלע האדום על ידי D16~D23. 8 הביטים של המספר הדיגיטלי קובעים את גודל המתח האנלוגי שיוצא מהוואיתו את עצמת הצלע בכל אחד מ 3 צבעי ה-RGB.

התמונה בתצוגת TFT מורכבת מפיקסלים. תצוגת TFT זו מורכבת ממכפלת של 2 המספרים 320 × 240 (הנותנת 76800 פיקסלים). ניתן לשולט על הצלע של כל פיקסל ומכאן ניתן להבין כי ניתן להציג תצוגה מנוקודה בודדת ועד תמונה מורכבת.

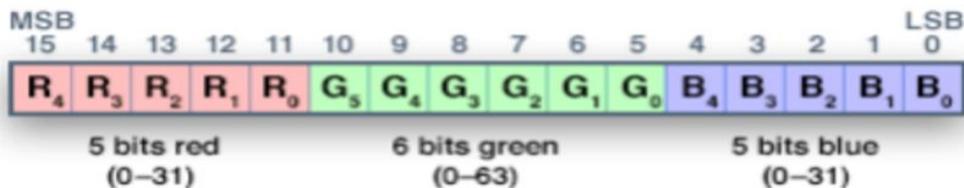
לכל פיקסל יש קואורדינטות (קואורדינטות הן קבוצת מספרים המציינות את מיקומו של גוף במרחב כלשהו) של x ושל y שלו. מערכת הקואורדינטות ממקמת את הראשית (0,0) (בפינה השמאלית העליונה כאשר x גדול לכיוון ימין ו- y כלפי מטה, ניתן לראות באירוע).



בהתוצאות התומכות בצבע, הצבע נרשם כערך בן 16 סיביות לא מסומן – (unsigned) כמו כן בחלוק מההתוצאות ניתן לקבל יותר צבעים ובחולק פחות – יותר או פחות סיביות בערך. רוב הספריות שעובדות עם תוצאות כאלה עובדות עם 16 סיביות. גם לארדואינו קל לעובוד עם צבעים בני 16 ביטים. שלושת צבעי היסוד -RGB- אדום (red), ירוק (green), כחול (blue) נוכנים בתוך ה-16 סיביות بصورة הבאה:

מחיוך רואים:

5 הביטים הגבוהים הם של הצבע האדום והם בני 5 ביטים (0-31).



6 הביטים הבאים הם של הצבע הירוק (0-63).

5 הביטים הנמוכים הם של הצבע הכחול (0-31).

צבע הירוק יש 6 ביטים כי העין של האדם רגישה יותר לצבע הירוק,

לדוגמה, אם נרצה לצבע פיקסל בצבע ירוק נשלח את הערך $00000\ 11111\ 00000 = 07E0H$ הערך: $F800H = 1111100000000000$ יציב את הפיקסל באדום והערך FFFFH יציב את הפיקסל לבן.

הצבעים הנפוצים הם:

- #define BLACK 0x0000 • שחור //
- #define BLUE 0x001F • כחול //
- #define RED 0XF800 • אדום //
- #define GREEN 0x07E0 • ירוק //
- #define CYAN 0x07FF • טורקיז //
- #define MAGENTA 0XF81F • גוון אדום //
- #define YELLOW 0xFFE0 • צהוב //
- #define WHITE 0xFF • לבן //

```

1. // Color definitions
2. #define BLACK      0x0000      // שחור
3. #define BLUE       0x001F      // כחול
4. #define RED        0XF800     // אדום
5. #define GREEN      0x07E0     // ירוק
6. #define CYAN       0x07FF     // כחול ירוק
7. #define MAGENTA    0XF81F     // גוון אדום
8. #define YELLOW     0xFFE0     // צהוב
9. #define WHITE      0xFF      // לבן

```

בPRIOSHOM הקואורדינטות , קודם נרשמת הנקודה בציר ה X של הפיקסל ולאחריה בציר ה Y. הקואורדינטות מבוטאות ביחידות של פיקסל ולא בגודלים של אורך (מ"מ או ס"מ אינץ'ים וכו'). גודל מלבן שנציג בהמשך יהיה במספר פיקסלים שלו בציר ה X ומספר הפיקסלים בציר Y ולא באורך של מ"מ או ס"מ. ניתן למדוד את אורך ורוחב המסך ואז לחלק בכמות הפיקסלים בכל ציר ולדעת את הגודל של כל פיקסל, אם כי הדבר פחות שימושי. ניתן לשנות את האוריינטציה של התמונה כך שהאורך יהיה רוחב והרוחב יהיה אורך.



נקבל תמונה כזו :

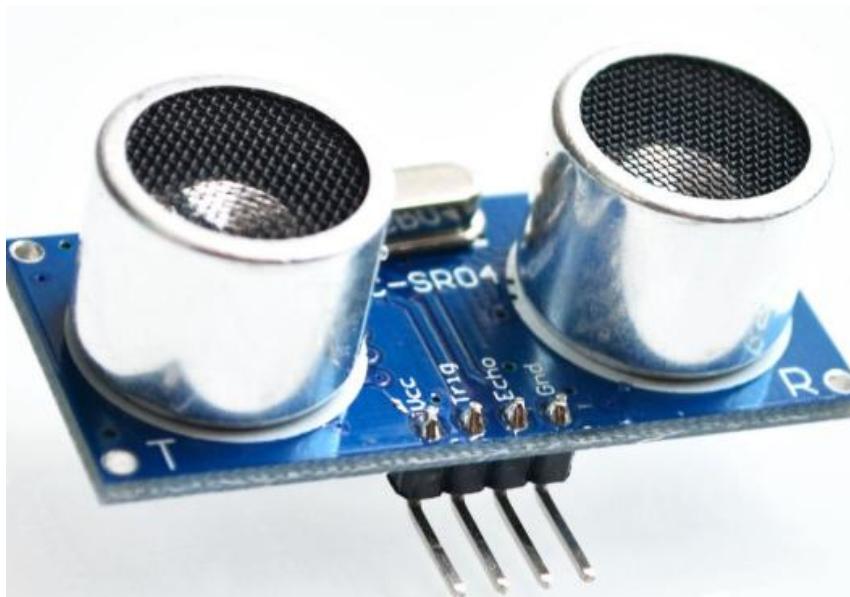


כלומר ממצב של תמונה כזו :

התצוגה הימנית נקראת פורמט landscape (נוף) והשמאלית portrait (דיוקן). ראשית הציריים היא הנקודה השמאלית למעלה. ניתן לקבל 4 סוגים אוריינטציה, ככלומר איזו אחת מ 4 הפינות של המלבן תהיה ראשית הציריים (0,0) .

פונקציות הגרפיקה לכל תצוגת TFT יש בקר תצוגה פנימי המנהל את התצוגה עצמה ואת הקשר עם העולם שמחוץ לתצוגה שבו יש בדרך כלל מיקרו בקרים השולחים פקודות להפעלת התצוגה. לכל בקר יש פונקציית אתחול משלו. כדאי לראות בדף הנתונים של התצוגה מי הבקר המפעיל אותה. במידה ולא יודעים אז בפונקציית.

חישון מרחק – HC-SR04 :



ה-HC-SR04 זה החישון בו השתמשתי למדידת מרחק בפרויקט שלי. זה חישון מאד נפוץ בפרויקטים של רובוטיקה, חישוני חניה, ומכשורים חכמים. חישון זה נפוץ בגלל מספר דברים, כמו למשל: הוא נחשב לזול, פשוט לשימוש ומדויק יחסית בטעחים קצריים ובינוניים.

איך החישון עובד?

העיקרון מאחורי ה-HC-SR04 מבוסס על גלי קול על-קוליים – חישון משדר גל קול בתדר גבוה, שאיןנו נשמע לאוזן האנושית (כ-40 קילו-הרץ). הגל הזה מתרחב בסביבה, וכאשר הוא פוגע במכשול או חפץ כלשהו, הוא חוזר אל החישון. החישון מודד את הזמן שלקח לגל להגעה חזרה. בעזרה מהירות הקול באוויר, ניתן לחשב את המרחק למכשול.

הנוסחה שמאפשרת לחשב את המרחק היא:

$$\text{מרחק} = \frac{\text{זמן החזרה} \times \text{מהירות הקול}}{2}$$

החלק החשוב הוא חלוקה ב-2, שכן הזמן שנמדד הוא הזמן הכלול של ההלך והחזרה של הגל. **מהירות הקול באוויר אינה קבועה**, והיא תלולה בעיקר בטמפרטורה. ככל שהטמפרטורה גבוהה יותר, המולקولات באוויר נעות מהר יותר ומעבירות את גל הקול ביעילות הרבה יותר. כתוצאה לכך, מהירות הקול עולה.

נוסחה לחישוב מהירות הקול

ניתן לחשב את מהירות הקול באוויר (במטרים לשנייה) בצורה מוקorbit באמצעות הנוסחה הבאה :

$$\text{מהירות} = T \times 331.4 + 0.6$$

כאשר:

- T היא הטמפרטורה במעלות צלזיאוס ($^{\circ}\text{C}$)
- $331.4 \text{ מ}/\text{s}$ היא מהירות הקול באוויר בטמפרטורה של 0°C .
- $0.6 \text{ מ}/\text{s}$ היא הקירוב לשינוי במהירות הקול עברו כל עלייה של מעלה אחת בטמפרטורה.

השפעה על הדיווק

אם לא מתחשבים בטמפרטורה, חישוב המרחק יכול להיות לא מדויק.
המערכות כה חשובות כגון כיפות ברזל או רובוט שומר בהתחשב בפרויקט שלי, יש צורך להתחשב בטמפרטורה הסביבתית על מנת להבין כיצד למרחק שמתקבל על פי החישון האולטרסוני.

מבנה החישון:

- HC-SR04 מגיע עם ארבעה פינים עיקריים :
- VCC - מתח אספקה של 5 וולט.
 - GND - חיבור לאדמה.
 - Trig - פין טריגר, אליו שולחים פולס קצר (בדרכן כל 10 מיקרו-שניות) כדי להפעיל את שליחת הגל האולטרסוני.
 - Echo - פין החזרה, שופיק פולס שמתממש לאורך זמן החזרה של הגל. אורך הפולס זהה מאפשר לחשב את המרחק.

מגבלות ויתרונות

מצוין למדידה בטוחים של 2 ס"מ עד 400 ס"מ. עם זאת, יש לו מגבלות :
משטחים קטנים או סופגים עלולים להקשوت על קריית המרחק, כמו גם רעים חזקים באוויר שמספריים לגלים האולטרסוניים. בנוסף, הוא רגיש לזווית החזרה של הגל – החישון עובד היטב כאשר המשטח ניצב אליו.

1.דרישות פולס הטריגר וה毛泽מון (Timing):

אף שצוין הצורך בפולס של $\geq 10\text{ }\mu\text{s}$ בפין Trig, חשוב להציג את סדר הפעולות המדוייק ואת התזמונים הקרייטיים לתקשורת תקינה:

- **שליחת הפולס :** יש להකפיד שהפולס יהיה $\geq 10\text{ }\mu\text{s}$ (או ארוך יותר, אך $\geq 10\text{ }\mu\text{s}$ זה המינימום הדרושים) כדי להפעיל את המשדר האולטראסונייק.
- **זמן המתנה פנימי (Internal Delay) :** לאחר קבלת פולס הטריגר, המודול ממתין כ- $500\text{ }\mu\text{s}$ לפני שהוא מושדר בפועל את סדרת **שמונה פולסים** 40kHz אלחוטיים.
- **פין ה-Echo :** פין זה משמשנו למצב **HIGH** (גובה) ברגע שהשידור מתחילה, ונשאר במצב זה עד לקליטת החזר או עד זמן ה- "Time-Out" הפנימי (בסביבות 38ms). המיקרו-בקר שלך (כמו ארדואינו) צריך לדוד את משך הזמן שבו פין ה-Echo נמצא במצב **HIGH**.

2.רזולוציה ודיוק מעשי (Resolution and Practical Accuracy)

מעבר לתלות בטופרטורה, ישנן מגבלות נוספות המשפיעות על הדיוק:

- **רזולוציה התאורטית (HC-SR04) :** משדר ומודד את הזמן. אם נניח מהירות קול של 340m/s , ($0.034\text{cm}/\mu\text{s}$), הפולס של 40kHz נמשך $\geq 25\text{ }\mu\text{s}$. אורך גל בודד הוא כ- 8.5mm . כדי לקבל רזולוציה של 3mm (0.3 ס"מ), נדרש יכולת מדידת זמן מדויקת מאוד.
- **מעשית, הרזולוציה של המודול היא כ-3 מילימטר.**
- **היסט מיניימי (Blind Zone) :** הטווח המיניימי הוא כ- 2cm . הסיבה לכך היא שהחישון זוקק בזמן מיניימי להשלמת הפעולה של שליחת, מכבי והתחלה הקליטה, לפני שהחץ של גל הקול הראשון חוזר.
- **פיצול אלומת הקול (Beam Divergence) :** החישון אינו פועל כ"קרכן לייזר" צרה. הוא משדר בקונוס (זווית פתיחה) של כ- 15° החישון יזהה את האובייקט הקרוב ביותר שנמצא בתחום קונוס זה. שימושה הדבר היא שטדיית המרחק משקפת את הנקודה הקרובה ביותר ביותר על פני האובייקט ולא דורך את המרכז. ככל שהמרחק גדול, שטח הכנפי של הקונוס גדול, מה שמחזק את הדיוק המקומי (Local Accuracy).

3.מודל הנהיגה הפנימי ומתח הלוגי (Driving & Logic Level)

- **מתח פעולה ובקרה (VCC) :** החישון דורש 5V בפין ה-VCC והוא מתאים לעבודה עם מיקרו-בקרים של 5V כמו ארדואינו אונו (ישנם רגילים מתאימים עבור חישונים העובדים ב-5 וולט גם במיקרו בקר ESP32).
- **תאיימות לוגית עם 3.3V :** כאשר מחברים את החישון למיקרו-בקרים של 3.3V כמו ESP32, (פין ה-Trig יכול להיות מופעל ישירות) מכיוון שה- 3.3V נחשב למצב HIGH(מספיק), אך פין ה-Echo מוציא פולס של 5V . חיבור 5V ישירות לפין קלט של 3.3V עלול לגרום לנויה. לכן, נדרש שימוש במחalker מתח (Voltage Divider) או ממיר רמות לוגיות (Logic Level Shifter) בפין ה-Echo.
- **צריict זרם :** החישון צורך זרם נמוך במצב רגיל (פחות mA-mA) אך בזמן שידור הוא יכול לצורך עד כ- 15mA .

חישון טמפרטורה LM75



חישון הטמפרטורה **LM75** אינו רק רכיב אלקטרוני פשוט אלא הוא מערכתי-על-שਬב (SoC) דיגיטלי קומפקטי שנועד למדוד טמפרטורה ולהמיר אותה לאוטות שמיקו-בקרים כמו הארדואינו או ה-ESP32 יכולים להבין בקלות. בנגוד לחישנים אנלוגיים (כמו ה-**LM35**) שמוסכאים מתח משתנה, ה-**LM75**- מבצע את ההמרה לאות דיגיטלי בתוכו, מה שפשט משמעותית את החיבור והתקנות.

מאפיינים וдиוק טכני:

- **טוחה מדידה :** החישון מתוכנן למדוד טמפרטורות בטווח רחב של $55^{\circ}\text{C} + \text{ עד } 125^{\circ}\text{C}$. טוחה זה הופך אותו למתאים ליישומים תעשייתיים, רפואיים וביתיים.
- **רזולוציה ודיוק :** החישון מספק קריאה ברזולוציה של **9 ביט**, המקבילה לדיוק 0.5°C . דגמים מתקדמים יותר כמו ה-**LM75B** מציעים רזולוציה של **11 ביט**, מה שמספק דיוק גובה יותר של 0.125°C (למרות שהדיוק המוחלט עדיין תלוי בטווח הטמפרטורה).
- **צריכת אנרגיה :** יתרון משמעותי של ה-**LM75** הוא ייעילותו האנרגטית. הוא צורך זרם נמוך מאוד במצב פעולה, וככלל מצב **כיבוי** (Shutdown mode) בו הוא צורך זרם של מיקרו-אמפרים בלבד (בסביבות $4\mu\text{A}$), מה שהופך אותו לאידיאלי ליישומים המופעלים על סוללות.
- **מתח עבודה :** החישון יכול לפעול בתחום שנו בין 2.7V ל- 5.5V , מה שהופך אותו לתואם למגוון רחב של מיקרו-בקרים, כולל ארדואינו (5V) ו-ESP32 (3.3V).

עקרון הפעולה והתקשות הדיגיטלית (I2C): (כפי שהוסבר בתחילת הספר)

ה-**LM75** מתקשר עם המיקרו-בקר באמצעות פרוטוקול, (Inter-Integrated Circuit, I2C) הידוע גם בשם TWI (Two-Wire Interface).

- **SDA (Serial Data Line) :** קו דו-כיווני להעברת נתונים.
- **SCL (Serial Clock Line) :** קו שעון המסנכרן את התקשרות בין המיקרו-בקר לחישון.

לכל רכיב I₂C יש כתובות ייחודית של 7 ביט. ה-LM75-מציע שלוש רגלי כתובות (A0, A1, A2) שנitin לחבר ל-VCC-או ל-GND, מה שמאפשר להגדיר עד שמונה **חישני LM75** שונים על אותו אוטובוס, I₂C ללא הפרעות.

כיצד זה עובד?

המיicro-בקר (הMASTER) שולח בקשה לקריאת נתונים לכתובת הספציפית של ה-LM75-(העבד). החישן מגיב נתונים המאוחסנים בזיכרון הפנימי שלו, שם הטמפרטורה נשמרת באופן רצוף לאחר המרה.

פונקציונליות "תרמוסטט- (Over-temperature :

אחד התכונות המייחדות והשימושיות של ה-LM75 היא יכולתו לפעול כתרמוסטט דיגיטלי עצמאי. לחישן יש יציאה ייודית OS (Over-temperature Shutdown), שיכולה לשמש בשני מצבים :

- **מצב השוואה (Comparator Mode):** המicro-בקר יכול להגדיר שני ספי טמפרטורה
 - בחישן עצמו :
 - TOS (Temperature Over-limit) : טמפרטורה מקסימלית. כאשר הטמפרטורה עולה מעלה מערך זה, יציאת OS-פעילה פلت דיגיטלי.
 - THYST (Hysteresis) : טמפרטורה שבה הפלט חוזר למצב רגיל. הדבר מונע הפעלה וכיבוי חזריים ונשנים (נדון) של היציאה כאשר הטמפרטורה נמצאת קרוב ל██.
- **מצב פסיקה (Interrupt Mode) :** במצב זה, יציאת OS- יכולה לייצר פסיקה (interrupt) למicro-בקר, להתריע בפניו על חריגת מהטמפרטורה, ללא צורך בסקרים מתמידים של החישן.

פונקציה זו שימושית במיוחד לבניית מערכות פשוטות של בקרת טמפרטורה, כמו מפוח שמופעל אוטומטית כאשר הטמפרטורה עולה על סף מסוים.

שלבי חיבור בסיסיים:

- .1 **VCC** (מתח) ו-**GND** (הארקה) : חיבור למקור מתח של 5V או 3.3V.
- .2 **SCL** ו-**SDA** : חיבור לפיני I₂C המתאימים בארדואינו (ברוב הלוחות : A4 ו-A5 בהתאם) או ל-ESP32.
- .3 **נגד פול-אף :** יש צורך בנגדי פול-אף (בדרך כלל 4.7Ω) על קווי ה-SCL ו-SDA כדי למשוך את הקווים למתח גבוה כאשר הם אינם פעילים. במודולים רבים, הנגדים הללו כבר מובנים על הלוח.

חיבור החומרה ב-ESP32

חיבור הגרפי של חישן LM75 ל-ESP32 הוא פשוט מאוד, כיון ששניהם תואמים למתח עבודה של 3.3V.

- .1 **VCC**(מתח) של ה-LM75-מחבר לפין 3.3V ב-ESP32.
 - .2 **GND**(הארקה) של ה-LM75-מחבר לפין GND ב-ESP32.
 - .3 **SDA**(קו נתונים) של ה-LM75-מחבר לפין GPIO21 ב-ESP32.
 - .4 **SCL**(קו שעון) של ה-LM75-מחבר לפין GPIO22 ב-ESP32.
- פיני GPIO21 ו- GPIO22 הם בירית המחלל של I₂C בלוחות ESP32 רבים, מה שמנשט את החיבור.

שלבי תכונות (פסאודו קוד):

- .1. **הכללת ספרייה:** השתמש בספרייה ייעודית לחישון (למשל, Adafruit LM75) כדי לפשר את התקשרות.
- .2. **Ấתחלול:** בפונקציית ()`setup`, אתחול את החישון.
- .3. **קריאת נתוניים:** בפונקציית ()`loop`, קרא את ערך הטמפרטורה מהחישון. הספרייה כבר תמיר את הערך הדיגיטלי לטמפרטורה בצלזיוס.
- .4. **הצגת הנתוניים:** הדפס את הנתוניים לצג טורי (Serial Monitor) או לצג LCD.

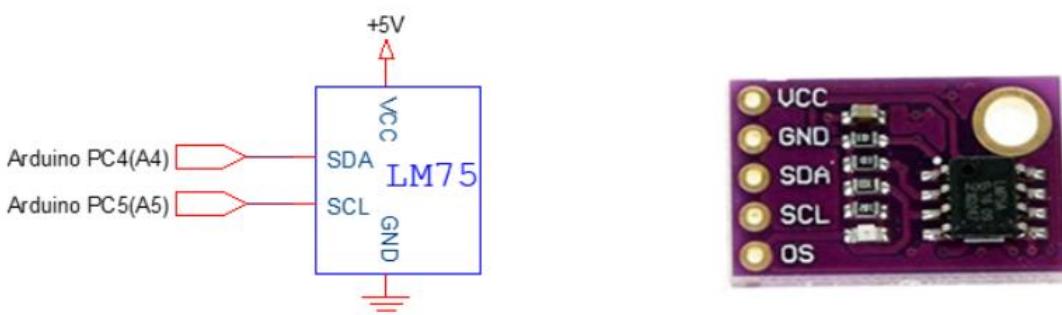
ישומים נפוצים:

השילוב הייחודי של דיוק סביר, צריכה נמוכה ומשק דיגיטלי פשוט הפך את ה-LM75 לבחירה פופולרית ב מגוון רחב של תחומיים :

- **מערכות קירור:** בקרת טמפרטורה בצד אלקטронני, שרתים ומחשבים כדי למנוע התחלומות יתר.
- **טרמוסטטים:** בניית מערכות בקרת אקלים פשוטות לבית או למשרד.
- **מכשירים ניידים:** שימוש במכשירים המופעלים על סוללות, כמו שעונים חכמים או מערכות ניטור אישיות.
- **מדידת טמפרטורה כללית:** ישומים מדעיים, תעשייתיים וחינוכיים שבהם נדרשת מדידת טמפרטורה אמינה ולא יקרה.

לxicom, ה-LM75 מספק פתרון אלגנטי וחסכוני למדידת טמפרטורה דיגיטלית. הוא משריר את המיקרו-בקר ממשימת המרת האות האנלוגי, מציע גמישות בשימוש בזכות I₂C ותכונות ה-"טרמוסטט" המובנות, מה שהופך אותו לרכיב יסוד בפרויקטים רבים של IoT ואלקטרונית.

נראות ומאפיינים עיקריים של החישון:



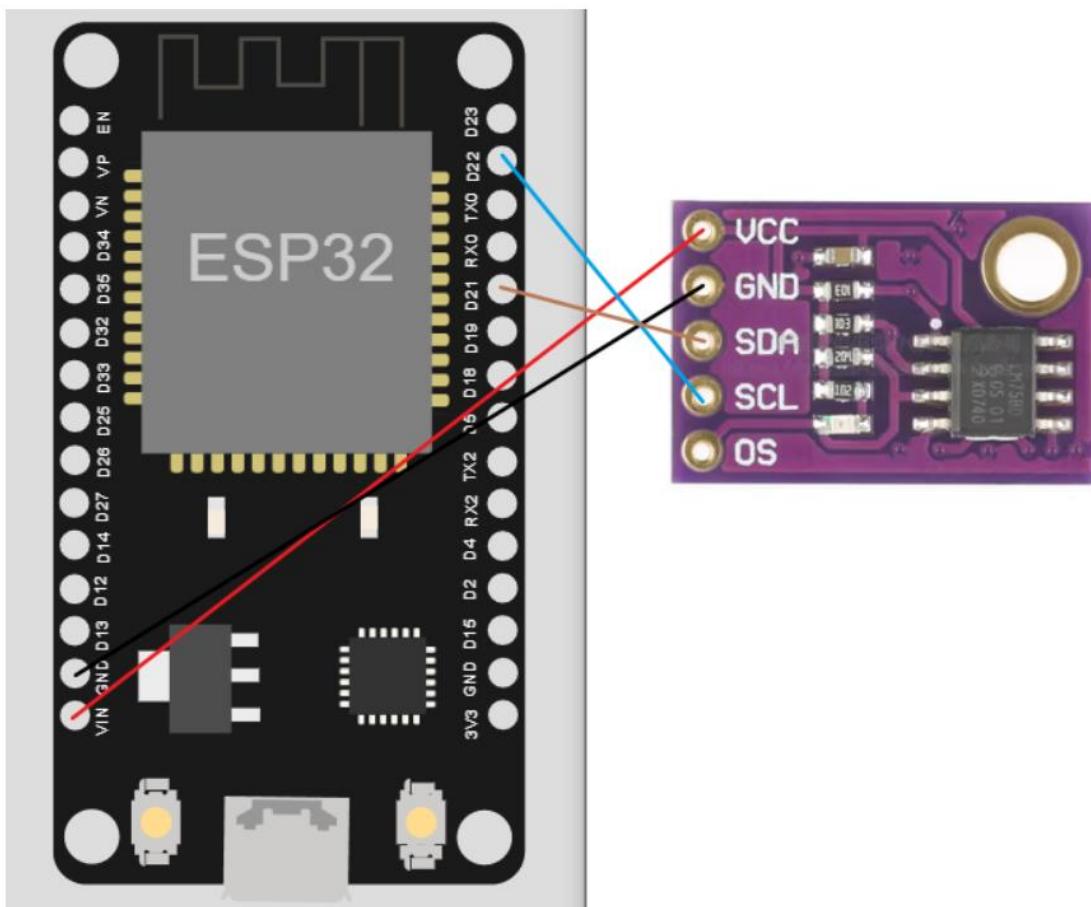
הסבר מדוע החישון מחובר באמצעות ה프וטוקול I2C

I2C הוא פרוטוקול תקשורת שמאפשר לחבר התקנים שונים לוח הארדואינו באופן פשוט יחסית. במקרה שלנו, החישון **LM75** מחובר דרך פיני ה- SCL ו- SDA של הלוח. פינים אלה משמשים לתקשורת I2C דרך הפינים האלה, הארדואינו ESP32 והחישון "מדוברים", באמצעות פרוטוקול I2C.

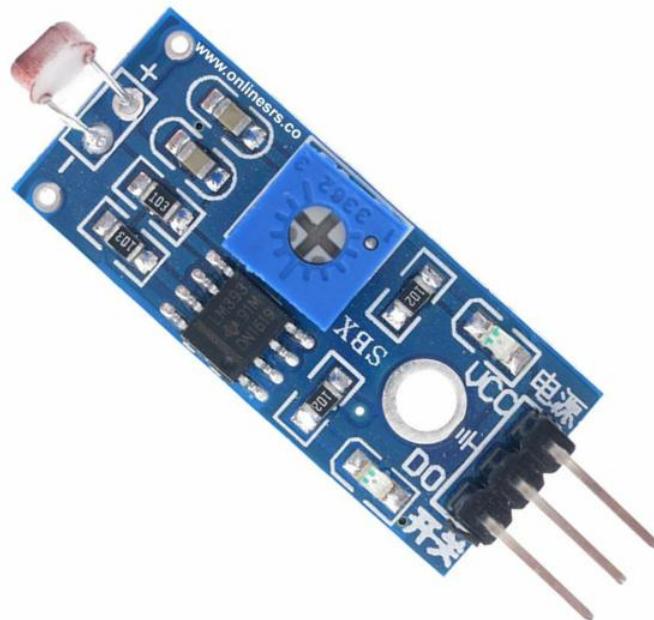
כשאנו רוצים לקרוא את הטמפרטורה, הארדואינו/ESP32 שולח בקשה לחישון דרך פיני ה- I2C. החישון קורא את הטמפרטורה, ושולח חזרה את הערך דרך אותם פינים. כך אנחנו יכולים לקרוא את הטמפרטורה בקלות רבה, ללא צורך בمعالג מורכב.

I2C מאפשר לנו לחבר מספר התקנים על אותם פינים SCL ו- SDA, ולתCKER איתם באופן עצמאי

נראה חיבור של החישון LM75 אל מיקרו בקר ESP32:



מודול חיישן האור KY-018



מודול חיישן האור (KY-018 או "LDR Module") הוא רכיב אידיאלי למדידת עצמת אור ובקרת תאורה אוטומטית באמצעות ה-ESP32. המודול משלב את החישון הבסיסי עם רכיבי בקרה, מה שמאפשר שתי דרכי פעולה: קריאה מדוקט או קריאה בינארית פשוטה.

1. המבנה והרכיבים העיקריים:

המודול בנוי סבב שלושה רכיבים מרכזיים:

1. **נגד תלוי-אור (LDR):** זהו החישן הראשי שאחראי על המדידה. כשהאור הפוגע בו גדל, החתנוגדות שלו יורדת. המודול מכיל מעגל פנימי שממיר את שינוי החתנוגדות הזה לשינוי מתח.
2. **קונפרטור מתח (LM393):** זהו הцип האלקטרוני הראשי. תפקידו הוא להשוות את המתח המשנה של ה-LDR מול מתח ייחוס קבוע (הסף). השוואה זו מניבה את הפלט הדיגיטלי (D0).
3. **פוטנציאומטר כחול:** זהו נגד משתנה שמאפשר למשתמש **לכינול ידנית** את מתח הייחס הקבוע של ה-LM393 ובכך לקבוע באופן חומרתי את סף ההדרקה/כיבוי של היציאה הדיגיטלית.

2. דרכי השימוש במודול (היציאות):

הגמישות של המודול נובעת משתי יציאות הפלט שלו:

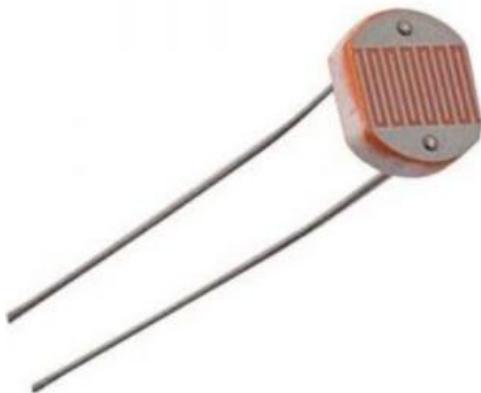
א. יציאה אנלוגית - (A) למדידה מדויקת

ב. יציאה דיגיטלית - (D0) להחלטה בינארית

- **תפקיד:** פין זה הוא הפלט של ה-LM393. הוא מספק רק שתי אפשרויות: HIGH או LOW.
 - **היתרון:** השיטה היא "הכנס והפעל". ה-LM393 מבצע את ההחלטה האם חסוך או מואר עבורה, בהתאם לכיוול שקבעת באמצעות הפוטנצימטר הכהול.
 - **חיבור:** חיבור את D0 לפין **GPIO דיגיטלי** רגיל ב-ESP32.

:LDR - Light Dependant Resistor

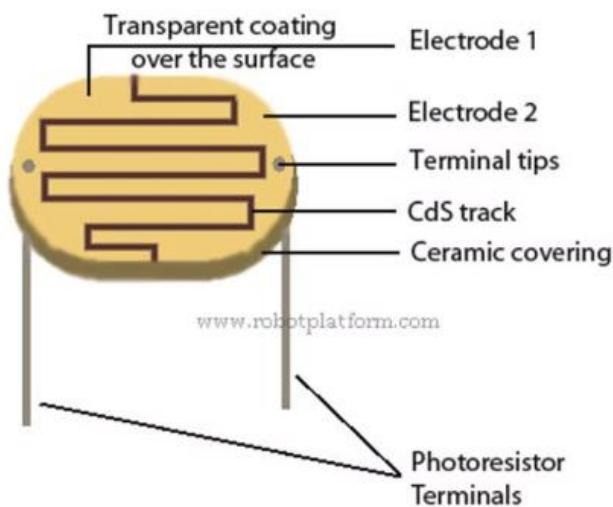
חישון היכול למדוד את עצמות האור הנמצאת בסביבתו. שמו באנגלית מסגיר את אופן פועלתו - נגד תלוי אור, יכולומר נגד שמשנה את התנגדותו בתלות באור אליו הוא נחשף.



מבנה חישון א/or:

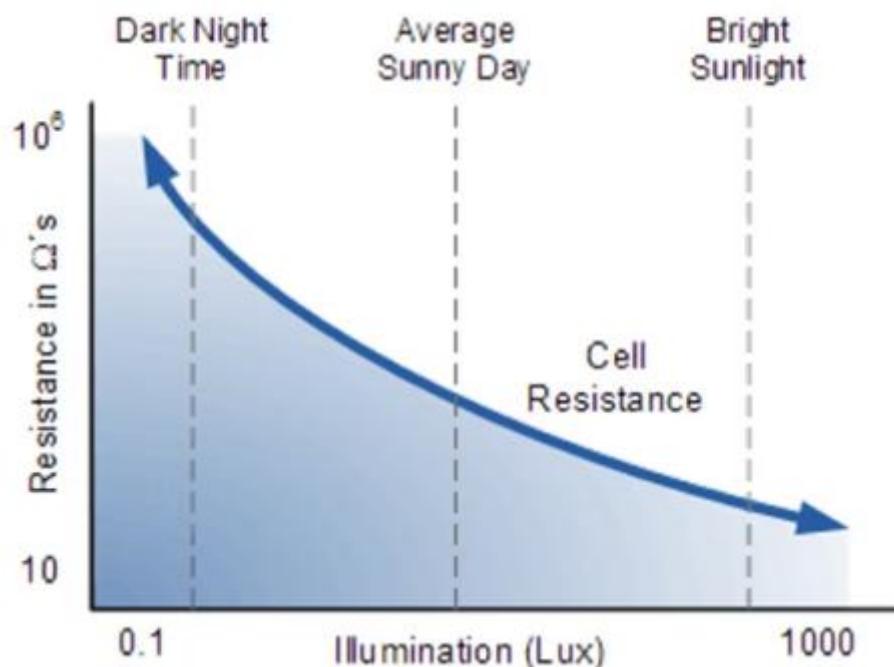
הHIGH-TECH מושג הושג באמצעות תרכובת כבש-סולפיד צדדי (CdS). תרכובת כבש-סולפיד צדדי (CdS) היא תרכובת כבש-סולפיד צדדי (CdS) המורכבת מ- 10% של תרכובת כבש-סולפיד צדדי (CdS) ו- 90% של תרכובת כבש-סולפיד צדדי (CdS).

- בחושך (התנודות גבואה):** במצב זה, האלקטרונים בחומר ה- CdS קשורים חזק לאטומים שלהם. כתוצאה מכך, ישנים מעט מאוד נושאי מטען חופשיים שימושיים לשאת זרם חשמלי, ולכן החישון מציג התנודות גבואה מאוד (עשרות קילו-אוום עד מגה-אוום).
 - באור (התנודות נמוכה):** כאשר אור (המורכב מפוטונים) פוגע בשטח הפנים של ה- CdS, הפוטונים מעבירים אנרגיה לאלקטרונים ומשחררים אותם מבננה האטום. שחרור זה יוצר מספר רב של אלקטרונים חופשיים.
 - תוצאה חשמלית:** ככל שנוצרים יותר אלקטرونים חופשיים, המוליכות של החומר גדלה, וההתנודות החשמליות של החישון יורדת באופן דרמטי.



אופין שינוי התנודות:

החישון משנה את התנודות בהתאם לעוצמת האור אליה הוא נחשף. כאשר החישון בחושך מוחלט איז התנודותו מאד גבוהה (ניתן לראות קטע מדף נתוני הייצור), ואילו כאשר הוא מואר התנודותו יורדת באופן מהיר. עוצמת תאורה נמדדת ביחידות מידת הנקראות Lux.



אוף המדידה של מודול חישון האור KY-018 מtabסס על עקרון **מחלק המתח**. הליבה של החישון היא **נגד תלוי אור (LDR)**, אשר התנודותו משתנה באוף הפוך לעוצמת האור. המודול מחובר ל-ESP32 דרך אספקת מתח של (VCC) 3.3V המודול עצמו כולל **נגד קבוע** בטור עם ה- LDR כדי ליצור מחלק מתח.

כחותאה משינוי התנודות של ה-LDR, המתח בנקודה המשותפת (המסומנת A0) משתנה. בקורס ESP32 מודד את המתח המשנה זהה באמצעות הפין האנלוגי (A0) ומתרגם אותו לערך מספרי בטוחה של **עד 4095**, בהתאם לרזולוציית 12bit של הממיר האנלוגי-לדיגיטלי (ADC) שלו.

מנוע SERVO

הסביר כללי על המנוע SERVO:

מנוע SERVO הוא מנוע שבו יש את המרכיבים הבאים :

- **מנוע זרם ישיר (DC Motor)**
- **מערכת תמסורת פנימית של גלגלי שיניים**
- **בקשה אלקטרוניות על מיקום המנוע.**

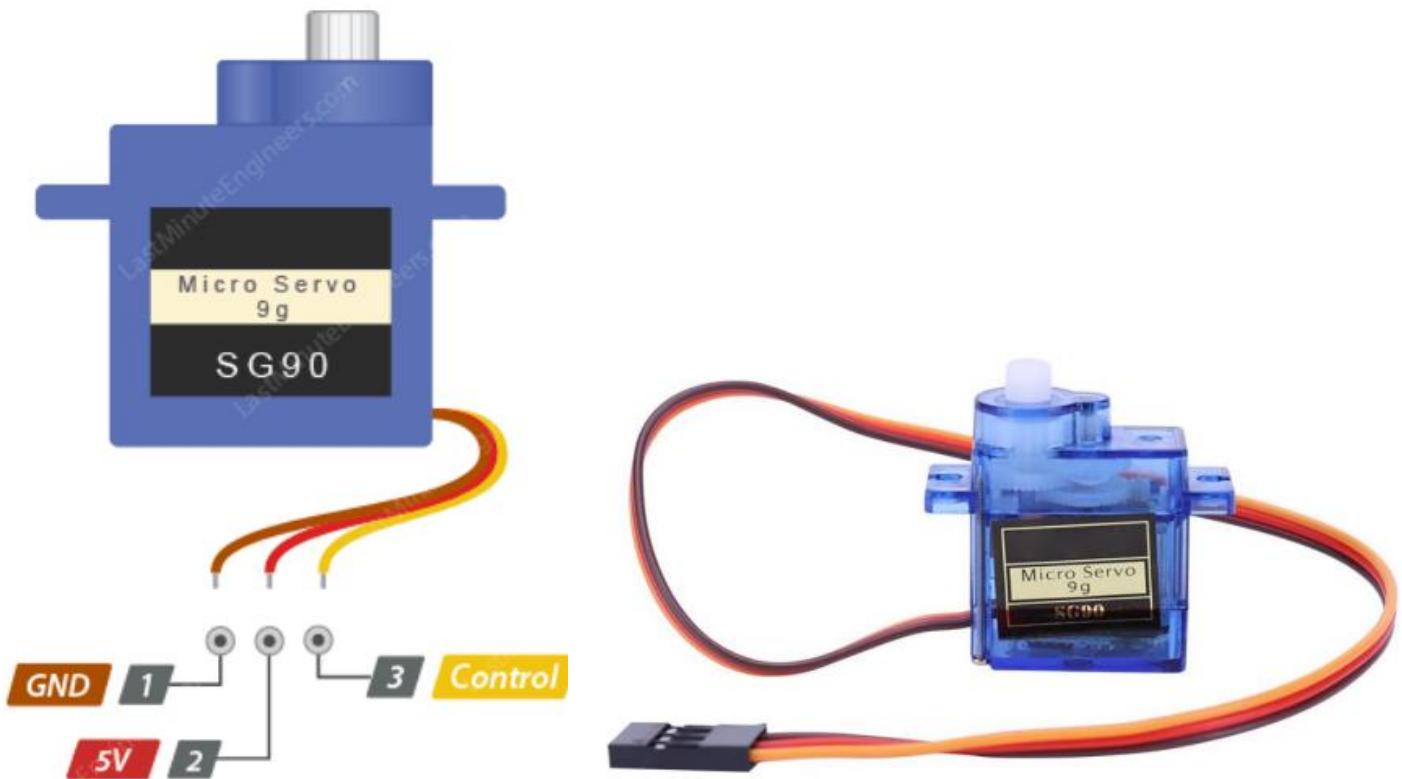
מנוע SERVO לא מסתובב בצורה חופשית כמו מנועי DC אלא נע עם פי זווית – לרוב בין 0 ל- 180 מעלות (אם כי יש מנועי SERVO שמסתובבים גם עד 360 מעלות).

מנועי SERVO נפוצים מאוד ומשמשים באפליקציות רבות. הסיבות לכך הן : יכולת השיליטה על **מנועי SERVO**, דרישות האנרגיה הנמוכות) **יעילות**, (ה**כוח החזק יחסית** של המנוע, **רמת המתוח** המשמשים להפעלו, **הגודל, המשקל והמחיר** הנמוכים שלו.

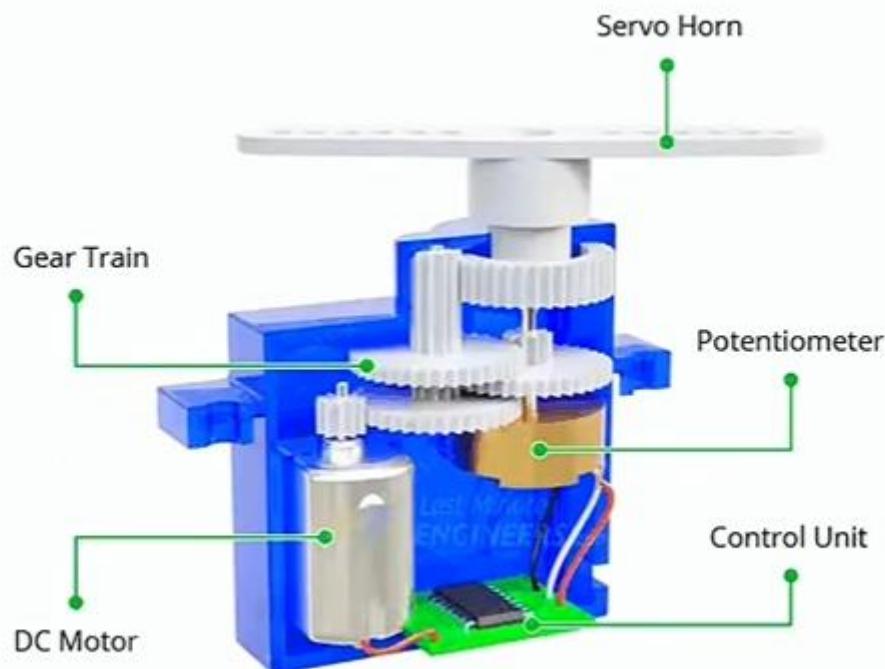
מנועי SERVO פועלם **בתוך מעגל סגור**, כלומר יש להם מערכת בקרה על מיקום המנוע והם יכולים לתקן הפרשיות מהמיוקם הרצוי.

האיור הבא מתאר מנוע SERVO שנitinן להשיג באינטרנט במחירים נמוכים יחסית ואת החיבורים שלו המנוע נקרא : **SG90**.

נראה כיצד נראה המנוע SERVO:



מנוע SERVO מכיל מנוע DC קטן המחבר לציר היציאה דרך גלגלי השיניים. על הציר מחובר פוטנציאומטר המחזיר משוב למערכת הבקרה על המיקום הנוכחי בו הוא נמצא.



עקרון פעולה של מנוע SERVO

מנוע SERVO מופעל בדרך כלל על ידי **מיקום בקר** הגורם לו להגיע למיקום רצוי. פקודת התנועה מתקבלת בקר מטורמת למתח PWM שMOVEDר ליחידת הבקרה הנמצאת בתוך המנוע. ערך זה הוא **הערך הרצוי** של המנוע.

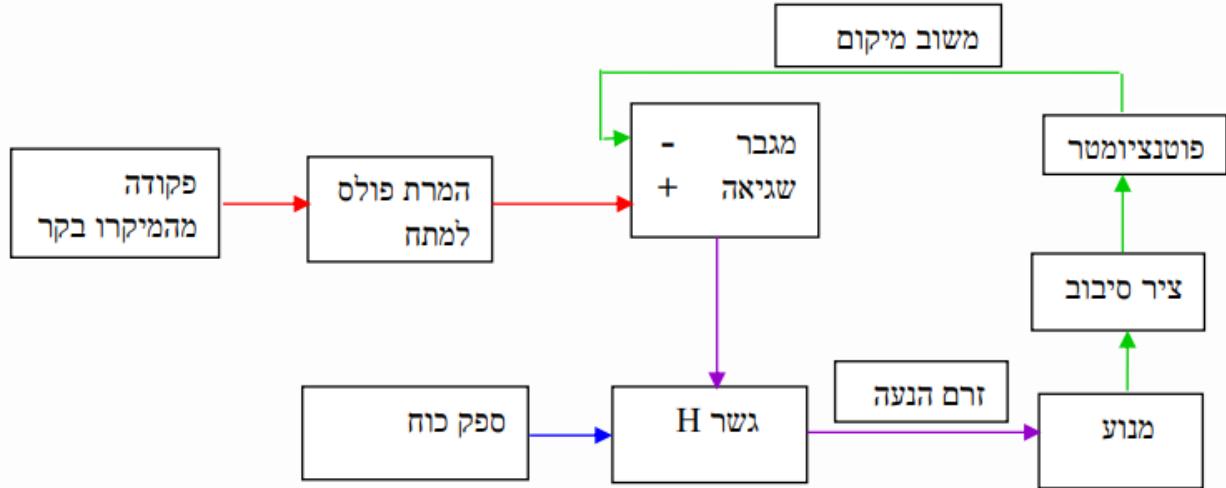
הערך המצוין של המנוע, נמדד **באמצעות פוטנציאומטר** המחבר לציר המנוע (הציר שיוצא מחלקו העליון של המנוע ואליו מחברים את המנגנון שיש להניע). סיבוב הפוטנציאומטר גורם לשינוי ההתנגדות שלו ולשימושים של שינוי מתח המתח על רגלו של הפוטנציאומטר. מתח זה הוא

הערך המצוין שימושו לברkar הסגור. כך נוצר **שי פרש** נקרא מתח שגיאה (בין המתח המצוין ולבין המתח הרצוי). מנוע ה-DC-SERVO, יושיק לנوع בכיוון המתאים.

סיבוב מנוע ה-DC מתרחש יחד עם **סיבוב גלגלי השיניים** - שמהווים תמסורת מפחיתה) מאטה את מהירות הסיבוב ומגדילה את מומנט הכוח. (גלגלי השיניים מניעים את ציר היציאה ואת הפוטנציאומטר. ברגע שהפרש המתחים יהיה קטן מערך סף (קרוב ל-0), ייחידת הבקרה של המנוע תנתק מתח הנחוץ למנוע ה-DC והוא יחולן לנوع. עצירת המנוע תהיה בדיקות בזווית שנשלה מהתוכנית בימיקרו בקר'.

אם התוכנית שולחת למנוע פקודה לנوع לזווית מעבר ל-180 מעלות, המנוע יתחיל לרעוז מכיוון שהוא ינסה לsegor את השגיאה בין **הערך הרצוי** למוציא ולא יצליה. כמו כן יש ציר יציאה שמסומם מכני מהמנוע לעبور את זווית הסיבוב של 180 מעלות.

נראה בעת כיצד נראה מערכת הבקרה של מנוע SERVO



מעולה, הנה גם הטקסט מהתמונה השלישית, מוקן להעתקה לוורד :

בצד שמאל מגיעו את הפקודה מהמיקרו בקר על הזרווית הרצוי שנקרא **הערך הרצוי**. זהו פולס עם **Duty Cycle** כאשר הזמן בין ה-ON ל-OFF קובע את הזרווית הרצוי. אותן זה נקרא גם **מיקום מטרה**.

معالג "מרת פולס למתח" מעביר למגבר השגיאה מתח שמצוין מה-**Duty Cycle** של אותן הפקודה.

מגבר השגיאה מקבל מהפוטנציאומטר מתח היחסי למיקום ציר המנוע, מתח זה הוא **המיקום הנוכחי של המנוע** ונקרא **ערך מצוי**.

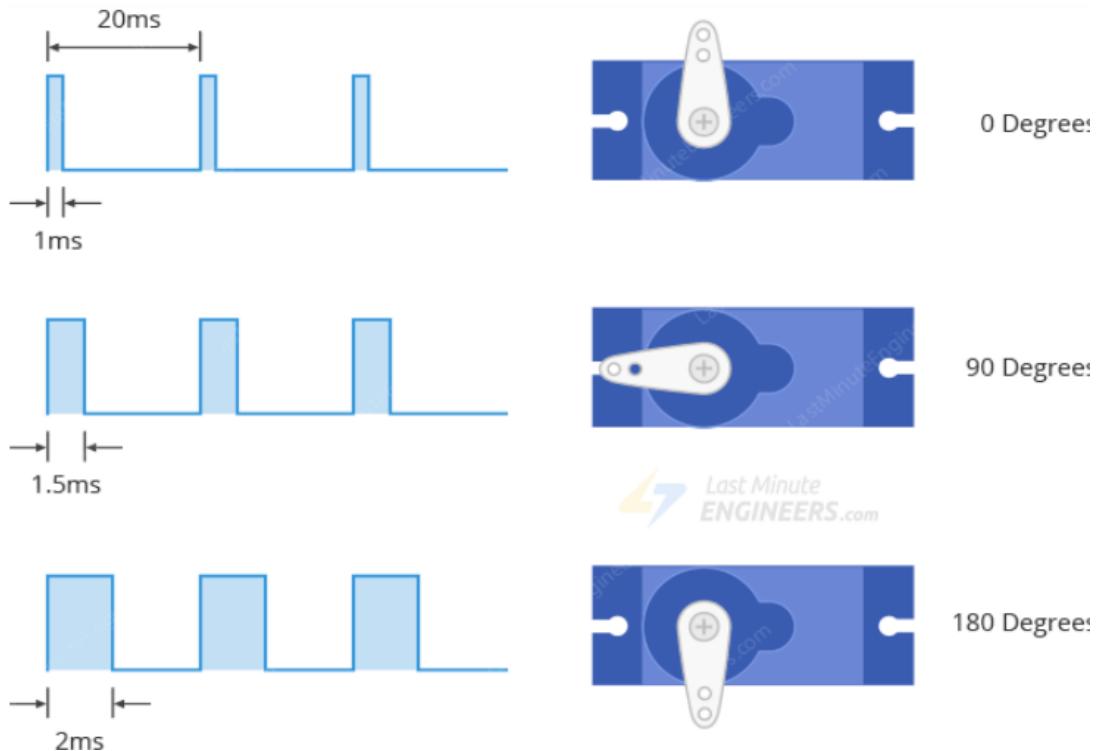
מגבר השגיאה מגביר את **הפרש המתח** בין הרצוי למתח המזמין. יציאת המגבר מחוברת לגשר - **H Bridge** שמעביר זרם המסובב את המנוע.

כאשר המנוע מסתובב הוא **מסובב את הפוטנציאומטר** שמעביר שוב מתח שלילי אל **הכניסה המהפקת** (המינוס של המגבר) של המגבר. ככל שהמנוע מסתובב וציר המנוע מתקרב אל הזרווית הרצוייה **הפרש המתח הולך וקטן**. כאשר מגיעים לזרווית הרצוייה, **הפרש המתח למגבר הוא 0** ואין איינו מוציא פקודה לסייעת המנוע והמנוע עצמן.

כיצד עובד מנוע סרוו?

עבודה עם פולסים:

אפשר לשלוט על מנוע SERVO על ידי שליחת סדרה של פולסים אליו. מנוע SERVO טיפוסי מחייב לפולס כל 20 אלףות השנייה (כלומר, תדר האות שנשלחה למיקרו בקר ציריך להיות 50HZ. רוחב הדופק קובע את מיקום המנוע - SERVO. האירור הבא מראה את הזוויות של המנוע עבור רוחבי שונים. Duty Cycles



מיקום המנוע כתלות ברוחב הדופק של ה-DUTY CYCLE.

מתיאור רוחב הפולס:

- **פולס קצר ברוחב של 1 אלףות שנייה או פחות מסובב את מנוע הסervo ל-0-מעלות.**
- **פולס ברוחב של 1.5 אלףות שנייה מסובב את מנוע הסervo ל-90-מעלות.**
- **פולס ברוחב של 2 אלףות שנייה בערך מסובב את מנוע הסervo ל-180-מעלות.**

פולסים בטוחה של 1 ms עד 2 ms יסובבו את הסervo למיקום פרופורצionalי לרוחב הפעימה.

יש מנועי servo המסוגלים להסתובב 360 מעלות. רוחב הפולס במנועים אלו יתן את הזוויות הבאות:

- **0.5 ms נוthen זווית של 0 מעלות.**
- **1 ms נוthen זווית של 45 מעלות.**
- **1.5 ms נוthen זווית של 90 מעלות.**
- **2 ms נוthen זווית של 135 מעלות.**
- **2.5 ms נוthen זווית של 180 מעלות.**

הערה: אין תאום מדויק בין רוחב הפולסים למיקום ולכון יהיה علينا לשנות את התכונות שלנו כדי להתאים לטוווח של מנוע-SERVO שלנו. כמו כן, **משך/רוחב הדופק** יכול להשנות בין יצרני מנועי SERVO שונים; לדוגמה, זה יכול להיות 0.5ms עד 2.5 ms עבור 180 מעלות ו- 1ms עבור 0 מעלות.

הטבלה הבאה מאפיינת את המנוע SERVO:

טיואר	עברית	אנגלית
23.2 x 12.5 x 22 mm	מידות	Dimensions
9 g	משקל	Weight
0.12sec/60degrees (4.8V) 0.10sec/60degrees (6V)	פעולה מהירות	Operating Speed
1.3kg.cm/18.09oz.in (4.8V) 1.5kg.cm/20.86oz.in(6V)	מומנט מעכב	Stall Torque
4.8V~6V	מתח פעולה	Operating Voltage
Analog	מערכת בקרה	Control System
CCW	כיוון	Direction
120degrees	זווית פעולה	Operating Angle
900us-2100us	הפולס הנדרש	Required Pulse
None	סוג מיסב	Bearing Type
Metal	סוג גלגל השיניים	Gear Type
Plastic	סוג המנוע	Motor Type
20 cm	אורך חוטי החיבור	Connector Wire Length

עבודה עם מנוע סרו ומיקרו בקר ESP32:

העבודה עם מיקרו בקר ESP32 בסביבת AEDUINO IDE דומה מאוד לעבודה עם כרטיסי מיקרו בקר ארדואינו (אונו, נאנו, מגה וכו').

התכונות כמעט זהה לתכונות עבור הארדואינו.

הערה חשובה: יש לשים לב כי המתח של מנוע ה-SERVO הוא בין 4.8V עד 6V ולא את מתח 3.3V שברכיבים.

מודול ממser כפול 5 וולט



מודול ממser כפול 5 וולט (2 Channel 5V Relay Module)

מודול ממser כפול 5 וולט הוא רכיב אלקטרוני חיווני המשמש כמתג אלקטرومגנטי. מטרתו העיקרית היא לאפשר לכם לשלוט במכשורים בעלי מתח וזרם גבוהים) כמו נורות ביתיות בעקבות 220V או מנועים (באמצעות מעגל בקרה בעל מתח נמוך ובטוח של **3.3 וולט כמו ה-ESP32**)

עקרון הפעולה של המסר:

בלב כל ממser נמצא **אלקטромגנטי** - סליל תיל. כאשרם מפעילים מתח של 5V על הסליל, הוא הופך למגנט רב עצמה. המגנט הזה מושך אליו **מagnet cusp** (Armature). תנעת המשך גורמת **למגעים החשמליים** של המפסק להיסגר או להיפתח, ובכך היא סוגרת או מנתקת את המעגל החזק שבו אתם רוצים לשלוט. ככלומר, זרם קטן בצד הבקרה משמש כמנוף להפעלת זרם גדול בצד העומס. המודול הזה הוא **כפוף** מכיוון שהוא מכיל שני ממserים כאלה, הפעלים **באופן עצמאי** אחד מהשני. זה מאפשר לשלוט בשני מכשורים שונים באמצעות מודול אחד.

התאמת ממser 5V ל-(3.3V):

השינוי העיקרי נובע מכך שה-ESP32 פועל ברמת מתח לוגי של 3.3V. בצד סלילי הממסרים במודול דורשים 5V להפעלה אמינה. הנה שני השיקולים המרכזיים וכייד להתמודד איתם:

1. אספקת כוח מופצלת הפרצת VCC ו-(JD-VCC)

הבעיה הנגדולה ביותר היא שסלילי הממסר צורכים זרם גבוה (בדרכן כל 100-70 מיליאמפר לכל ממסר). ה-ESP32 לא יכול לספק את הזרם הזה מהיציאות שלו, ואפילו לא מפני ה-5V שלו) אם הוא מופעל מ-USB (הפרטון הוא להפריד את אספקת הכוח).

- **בדיקה קרייטית:** רוב מודולי הממסר מגיעים עם מגשר (Jumper Cap) קטן המחבר בין הפינים (VCC ל-JD-VCC או JD-VCC ל-VCC) הפעולה הנדרשת:

1. יש להסיר את המגשר הזה! פעולה זו מפרידה בין מעגל השיטה (VCC) לבין מעגל הכוח של הסלילים (JD-VCC).

2. **VCC** הצד הבקרה : יש לחבר את VCC של המודול ל-3.3V של ה-ESP32 זה נותן מתח לעיני הבקרה האופטיות (Optocouplers) שבמודול.

3. **JD-VCC** הצד הכוח : יש לחבר את JD-VCC למקור מתח חיצוני וישיב של 5V. זה מספק את הכוח הנדרש לסלילי הממסרים.

4. **GND** : יש לחבר את הארץ (GND) של ה-ESP32 ואת ה-GND של ספק ה-5V החיצוני ביחד ל-GND של מודול הממסר.

2. רמת מתח השליטה V (3.3) מול V

פנוי הבקרה IN1 ו-IN2 מצפים בתיאוריה לאות V כדי להיות מופעלים/מכובים. למזלנו, ברוב המודולים החדשניים יותר (שמכילים אופטוקופלים) :

- אות (HIGH) כיבוי : (מתוך של V 3.3 הוצאה מה-ESP32) בדרך כלל מספק גובה כדי להיחשב כ"אות" HIGH ולכבות את הממסר (בנהנזה שהמודול הוא "הדק רמה נמוכה").
- אות (LOW) הפעלה : (מתוך של V 0 מה-ESP32) תמיד ייחשב כ"אות" LOW ויפעל את הממסר.

ברוב המקרים, ה-ESP32-עובד ישירות עם מודול V 5 ללא צורך במיר רמות מתח נוספים, בתנאי שהפרצת הכוח (סעיף 1) בוצעה כהלכה.

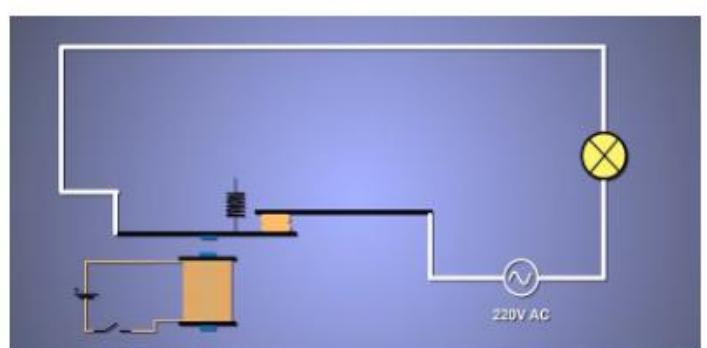
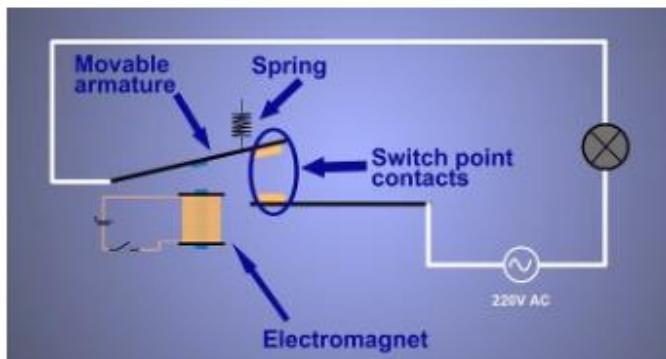
יתרונות מריציים

השימוש במודול זה מספק שני יתרונות קריטיים :

- בידוד ובטיחות** : הממסר יוצר **בידוד חשמלי מלא** בין מעגל הבקרה הרגיש והבטוח (V5) לבין מעגל הכוח המשוכן (כgon V220). זה מגן על הבקר שלכם מפני קפיצות מתח ו מבטיחה שהשליטה תתבצע בביטחון.
- יכולת מיתוג** : הוא מאפשר למעגל אלקטוריוני פשוט לשנות על זרמים גבוהים שהבקר לבודו לא יכול היה לשאת, בדרך כלל עד **10 אמפר** (A) ב- AC 250V או DC 30V (בהתאם למפרט הממסר הספציפי).

תיאור פשוט של אופן פעולה הרכיב

המסר הוא רכיב חשמלי הבניוי משני חלקים : סליל אלקטרו-מגנטי ומפסק (מtag). כאשר מועבר זרם חשמלי בסליל, נוצר שדה מגנטי המושך את המגעים ופותח או סגור את המעגל, דבר המוביל לניטוק או הפעלת המכונה/מערכת בהתאם.



הסבר פנימי של אופן פעולה הרכיב:

מצבי המגעים במסר

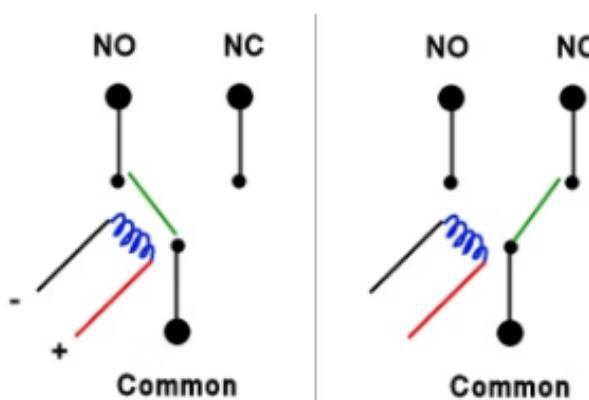
במסר לרוב אפשר להשתמש בשני מצבים עיקריים :

- מגע פתוח** : (NO) מצב שבו כשהמסר לא מופעל המגעים פתוחים ולכן לא עבר זרם במערכת (המעגל הנשלט פועל כשהמסר דולק).
- מגע סגור** : (NC) מצב שבו כשהמסר לא מופעל המגעים סגורים וועבר זרם במערכת (המעגל הנשלט פועל כשהמסר כבוי).

במסרים רבים, כמו זה אני עשית בו שימוש, למתכוון/ת המערכת יש אפשרות לבחור לחבר את המסר בתצורת NO או NC בהתאם לצרכים שלו/ה.

עקרון הפעולה האלקטרומגנטי:

מהסרטוט משמאל אפשר להבין פחות או יותר מה קורה בתוך המ מסר עצמו ואיך הוא פועל.



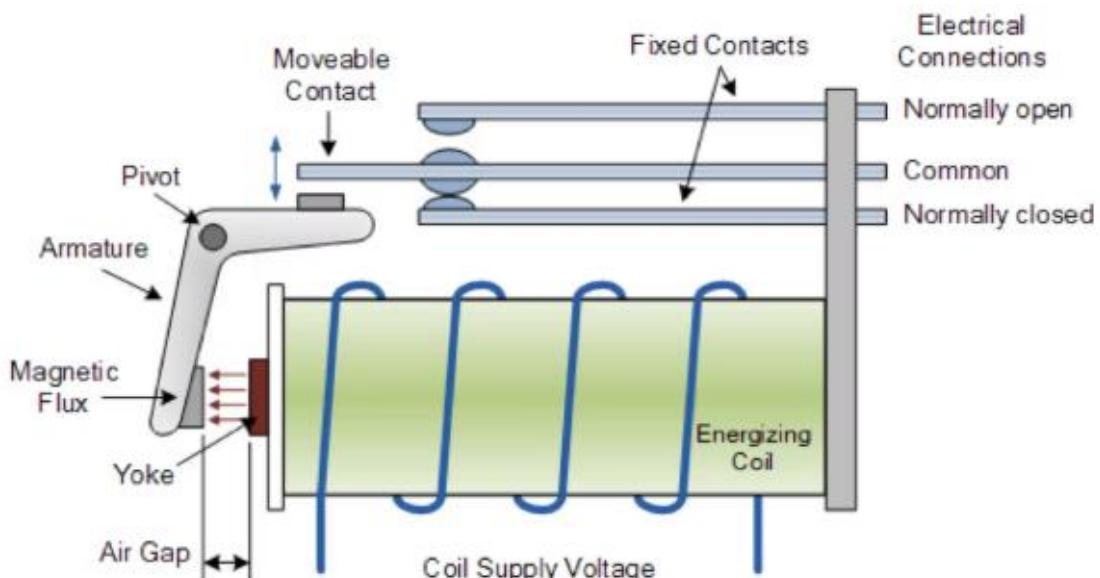
- **הסליל הכהול מייצג את מעגל השליטה הלווי, כשהוא מופעל על ידי הבקר ESP32.**

כאשר מעבירים דרכו זרם נוצר שדה מגנטי שמוסך את לשונית המתכת (בירוק).

- **הלשונית סוגרת מעגל בין רגלי הכניסה COM - common (COM) לרגל ה-NO.**

כאשר זרם, הלשונית (בעזרת קפיז) חוזרת למצבה המקורי ו**סוגרת מעגל** בין הכניסה לרגל ה-NC.

נבחן בשרטוט הבא:



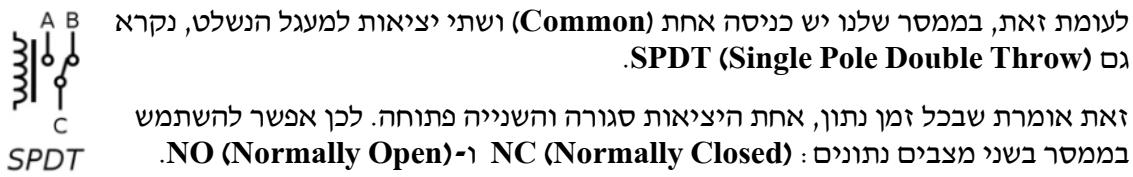
בשרטוט מועל ניתן לראות כיצד המ מסר האלקטרו-מכני הנפוץ נראה מבפנים ומאליו חלקיים הוא בינוי. **סליל אלקטרו מגנטי** שכادر עובר בו זרם מושך את המגנטי ובכך מזיז את לשונית ה-**Common** לסגור מעגל עם לשונית ה-**NO** (Normally Open), וכשבסליל לא עבר זרם הלשונית סוגרת מעגל עם רגלי ה-**NC** (Normally Closed).



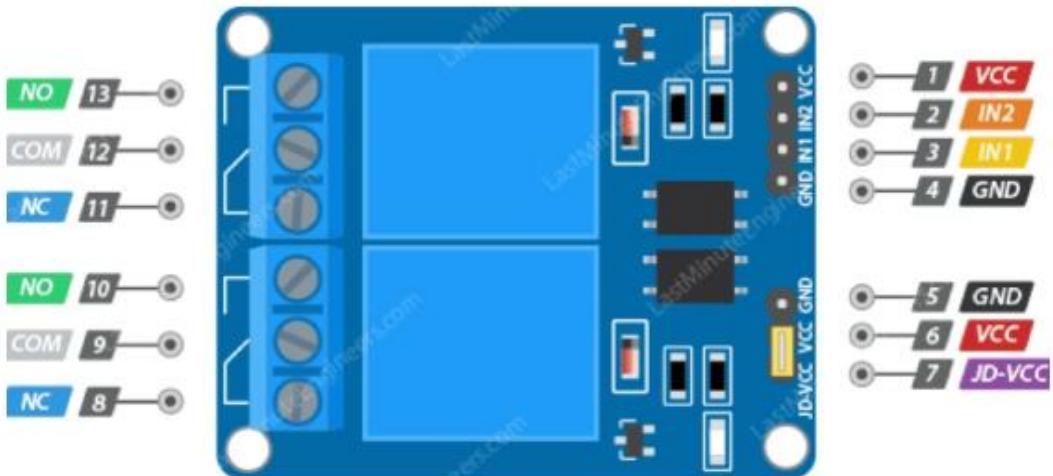
כדי להבין איך המ מסר עובד מבפנים נח Sob על המ פסק ה כי פשו ט שאפער לדמיין.



لم פסק כזה יש רק שני מצבים, פתוח או סגור. לצורה זו קוראים SPST - single pole single throw והמשמעות היא שיש לו כניסה אחת ויציאה אחת.



נתבונן באיוור הבא ונסביר אותו:



מבנה וחיבורו מודול MMCRC פול 7V:

מודול MMCRC הכפול מכיל שני MMCRC עצמאיים. החיבורים שלו מתחולקים לשני אזוריים עיקריים: כניסה בקרה ויציאות כוח.

1. כניסה בקרה (צד ה - ESP32 / אודיאינו)

אלו הפינים המקבלים מתח ושליטה מהבקר כגון ESP32 הפינים הנפוצים הצד זה הם :

- **VCC** : חיבור למתח האספקה הלוגי של המודול 5V (במקור, או 3.3V המשמשים ב-ESP32).
- **GND** : חיבור להארקה המשותפת של המערכת.
- **IN1** : כניסה דיגיטלית לשיליטה בMMCRC הראשון.
- **IN2** : כניסה דיגיטלית לשיליטה בMMCRC השני.

2. יציאות הכוח (צד המעגל הנשלט)

יציאות אלו מחברות את MMCRC למכשיר החשמלי הנשלט. לכל אחד משני MMCRC יש סט של שלושה חיבורים (בורגים) זהים :

- **(COM Common) משותף** : (זהי נקודת הכניסה המשותפת של MMCRC, אליה מחברים קצה אחד של המעגל הנשלט).
- **(NO Normally Open) פועל רגיל** : (זהו מגע שפותוח במצב מנוחה של MMCRC. המעגל נסגר רק כשהMMCRC מופעל).
- **(NC Normally Closed) סגור רגיל** : (זהו מגע ששגור במצב מנוחה של MMCRC. המעגל נפתח רק כשהMMCRC מופעל).

תיאור כללי של הרכיב:

הממסר הוא רכיב חשמלי המורכב משני חלקים: **סליל אלקטرومגנטי וmpsak (מוגע)**. מטרתו היא להעביר או לנתק את הזרם בין שתי נקודות במעגל חשמלי באופן נשלט. כאשר מועבר זרם חשמלי בסליל, נוצר שדה מגנטי המושך ופותח/סגור את המגעים, ובכך מפעיל או מנתק את המערכת בהתאם.

סקירת סוגי מסרים נוספים

מסמר מצב מזוק (SSR - Solid State Relay):

במסמר זה אין שימוש בחלקים מכניים, והוא עובד בצורה שונה מהמסר האלקטרו-מכני שאנו מכירים (EMR). הוא משתמש ברכיבים אלקטронיים מוליכים למחצה, כמו טרנזיסטורים וטיריסטורים, כדי לחבר ולנתק את הזרם בין המעגלים (השלט והנשלט) בצורה מבודדת. הוא מחוווט בצורה של (SPST כניסה אחת וכיינאה בלבד).

יתרונות:

- **עמידות גבוהה :** לעומת מסמר אלקטרו-מכני (EMR), חלקיו אינם נשחקים ומתבלמים מהר, ולכן הוא עמיד ליותר זמן.
- **מהירות :** מתאפשרה החלפה מהירה יותר בין מצביו המפסק.
- **שקט :** אין רעש בעת קיום הפעולות.

מסמר עלה (Reed Relay):

מסמר זה, כמו ה-EMR, משתמש בסליל אלקטромגנטי לפעולה, אך הפעם המוליך מושפע ישירות מהשדה המגנטי. המוליך הוא דק מאוד, ולכן הזרם הדרוש להפעלתו קטן יותר. במסמר זה שני חלקים המתג הם מגעים מוליכים דמיי "עליה". בזמן שהמסר מופעל, הסליל יוצר שדה מגנטי באזורי המגעים. כתוצאה נחוצים לבני קטים מגנטיים הפוכים כך שהם נמשכים זה לזה וסגורים את המעגל. כשהמסר לא פועל, אין שדה מגנטי והמגעים חוזרים למצבם המקורי (מצב פתוח).

גם במסמר זה הוא מסוג SPST.

יתרונות:

- **חסכון בחשמל :** דורש מעט חשמל להפעלה.
- **מהירות :** כמות מופחתת של חלקים, גודלם הקטן והזרם המועט הדרוש מאפשרים לו להחליף מצבים בקצב מהיר יחסית.
- **שיקפה מופחתת :** ה"עלים" נמצאים בתוך שפופרת זכוכית אטומה שמלאה בגז אציל (או מרוקנת), מה שמקטין את הקורוזיה (חולודה) ומאורך את חיי המסמר.

חסרונות:

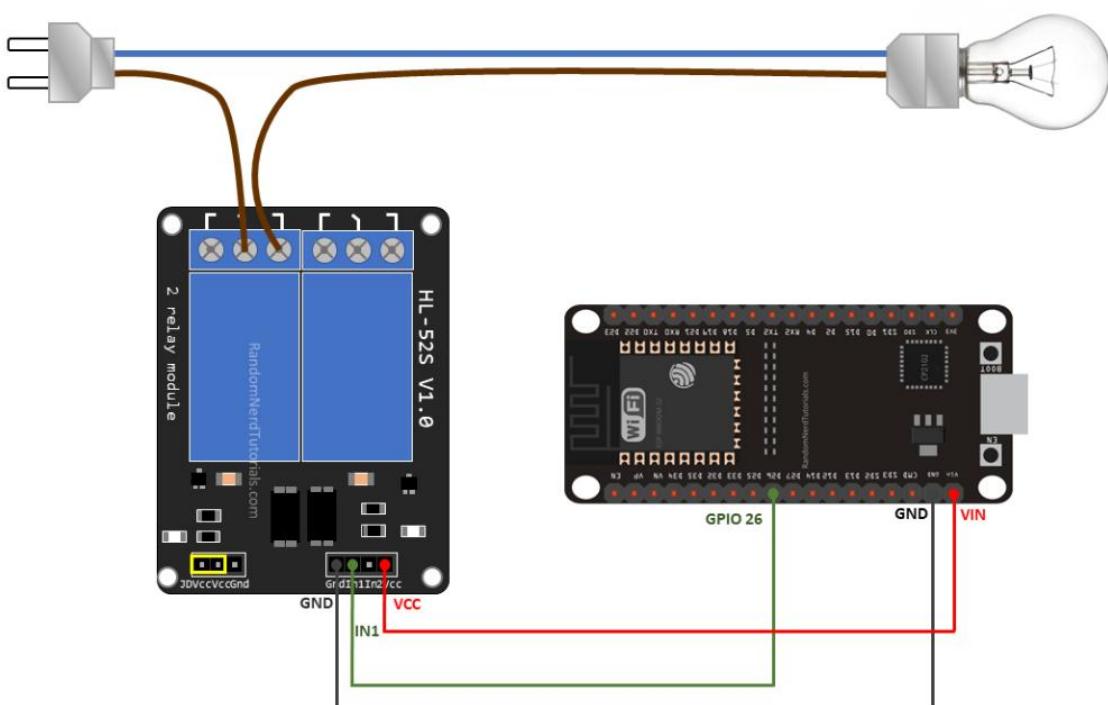
- **רגישות להפרעות :** הפרעות חיצונית עלולות להשפיע על תזוזת המגעים במסמר.
- **עדינות מכנית :** את המגעים מקיפה לרוב מעתפת זכוכית עדינה שעלולה להישבר מלחץ רב.

שגיונות שכדי להימנע מהן וטיפים מועילים:

כמובן שאלו רק שתி דוגמאות למסרים נוספים שאפשר למצוא בנוסח לממסר אלקטרוני-מכני.
ישנם סוגים רבים נוספים של מסרים הפעילים באופןים שונים שבמציעים פעולות דומות.

- **ברירת מחדל :** חשבו היטב כיצד אתם רוצים שברירת המחדל של המעגל הנשלט שלכם תיראה, ככה תוכלו לדעת לאן לחבר אותו לממסר (-**NO** או -**NC**)
- **קליק חזק :** כשהמסר מופעל שומעים קליק חזק. לא לפחות, זה סימן שהוא עובד.
- **מקור מתח נפרד :** המעגל הנשלט לא מקבל מתח מהבקר, (**ESP32**) لكن צריך לדאוג שהוא יהיה מחובר למקור מתח נפרד! (אפשר לראות דוגמה בהסביר על החיווט).
- **שימוש כפול :** אפשר להשתמש באותו מסר לשני מעגים שונים אם רוצים שהם אף פעם לא יופעלו במקביל ושיהיו תלויים באותו משתנים.
- **למשל :** לחבר שני לדיסטריפים בעלי מקור מתח משותף, אחד לרجل של ה-**NC**- והשני לרجل של ה-**NO**, וכך כשהמסר יפעיל – לדיסטריפ אחד יAIR והשני יהיה CABIN, וכשהמסר ייכבה – הראשון ייכבה והשני יפעיל.
- **דוגמאות לשימוש :** מנורתليلת שנדלקת כשהחדר חשוך, מאורור שמופעל כשהטמפרטורה גבוהה מדי.
- **מגבליות מתח :** בסדנה אנו משתמשים בדרך כלל במסר לשיליטה על מעגים בעלי מתח של עד **24V**, אבל בפועל הוא מסוגל להחזיק עד **10 אמפר** במתה של **250V**.
- **התאמת לבקר :** המסר דורש יחסית מעט מתח להפעלה, לכן הוא מצוין לשימוש ב벅רים כמו ה- **ESP32** שמסוגל להעביר לו את המתח הלוגי הדורש (3.3V) לשיליטה. מומלץ לשים לב עם אילו מתחים המסר שילך פועל כדי לוודא שהוא יעבוד עם הבקר.

נראה חיבור של מסר כפול המಡליק מנורה דרך מיקרו בקר ESP32:



הסבר מפורט על החיבורים בסכימה בדף הקודם:

הסכמה מחולקת לשולשה חלקים עיקריים : בקר ה-ESP32 (המוח), מודול הממסר הכפול (הmps) ומעגל העומס (הנורה).

1. חיבור צד הבקרה (ESP32 למודול הממסר)

חיבור זה משתמש במתח נמוך (V3.3V) כדי לשЛОט על הממסר.

- אספקת מתח לוגית (VCC) : הפין VCC במודול הממסר מחובר לפין VIN (Voltage In) ב-ESP32. במקרה זה, ה-VIN בדרך כלל מספק 5V למודול (אם ה-ESP32 מופעל מ-USB), אך חשוב לציין שרמת המתח הלוגית של ה-ESP32 היא 3.3V.
- האركה (GND) : הפין GND במודול הממסר מחובר ל-GND ב-ESP32.
- כניסה שליטה (IN1) : הפין IN1 במודול (השולט על הממסר הראשון) מחובר לפין GPIO 26 ב-ESP32. זהו האות הדיגיטלי (LOW/HIGH) שבו ה-ESP32 משתמש כדי להפעיל או לכבות את הממסר (בשיטת "הדק רמה נמוכה").
- חיבור JD-VCC : ניתן לראות שהחיבורים GND ו-JDVCC בצד המודול נשארים מנוקקים (או שנמחברים לכוח 5V חיצוני באמצעות ספק כוח נפרד), וזהו החיבור הנכוון והמומלץ עבור ESP32, שmbטיח שהסליל יקבל את הכוח החדשנו לו (5V) ולא יפגע בבקר.

2. חיבור צד העומס (הנורה למודול הממסר)

חיבור זה משתמש בממסר כדי לחבר או לנתק את זרם ה-AC מהנורה.

- מעגל ה-AC : כבל המתח (במקרה זה, כבל הפאזה, בדרך כלל חום/כחול) מהתקע נחתק.
- חיבור ל-COM : קצה אחד של הcabל החתוּך (הבא מהתקע) מחובר לפין COM (Common) של הממסר הראשון.
- חיבור ל-NO : הקצה השני של הcabל החתוּך (המוביל אל הנורה) מחובר לפין NO (Normally Open) של אותו מסר.

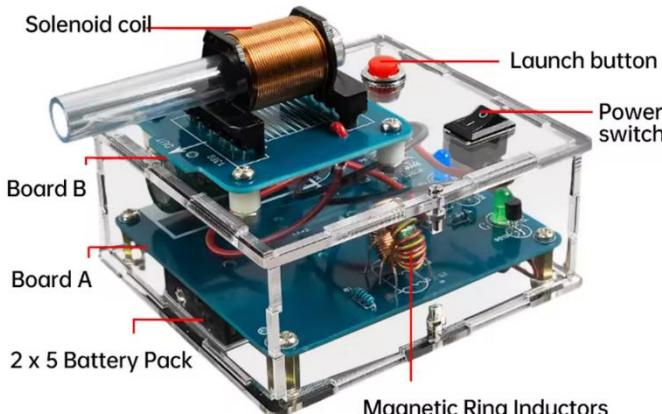
3. משמעות התצורה

התצורה שנבחרה היא (NO) (Normally Open), שפירושה :

- ברירת מחדל : כאשר ה-ESP32 כבוי או שולח אותן HIGH (V3.3V), הממסר נמצא במצב מנוחה, המגעים פתוחים, והנורה כבוייה.
- הפעלה : כאשר ה-ESP32 שולח אותן LOW (0V) לפין 26 GPIO, הממסר מופעל, סוגר את המעגל בין COM ל-NO, והנורה נדלקת.

לסיכום, הסכמה מראה כיצד ה-ESP32 שולט בביטחון במקשר חשמלי (נורה) באמצעות מודול ממסר כפול, תוך ניצול העיקרי האלקטרומגנטי לניתוק או חיבור של זרם ה-AC.

רובה אלקטרוני (Electromagnetic gun)



תיאור כללי:

המערכת המתוארת בתמונה היא רובה אלקטרוני שנוצר ע"י סליל (Solenoid Coil), המאיצ' גוף מתכתית דרך קנה פלסטי. הרובה נשלט ידנית באמצעות כפתור שיגור, ומתקבלת אספקת החשמל ממארז סוללות פנימי.

מטרת המערכת:

שיגור גופים מתכתיים גליליים קטנים (הנקראים כאן "Bombs") באמצעות כוח מגנטי – לצורכי הדגמה פיזיקלית, חינוכית או ניסיונית.

טבלת הסבר של התמונה:

רכיב

תיאור

Solenoid Coil סליל **אלקטרומגנטי**

סליל נחושת דרכו עובר זרם גבוה. יוצר שדה מגנטי חזק ברגע הפעלה ומושך את הגוף המתכתית לתוךו ב מהירות גבוהה.

Launch Button כפתור **שיגור**

Power Switch מתג הפעלה

מתג הפעלה/כיבוי ראשי של המערכת. מנתק ומחבר את אספקת המתח לכל המעגלים.

Board B לוח עליון

מכיל את הסליל, קונקטורים, כפתור הפעלה, נורות בקרה, והחיבורים הפיזיים.

Board A לוח תחתון

מכיל את הרכיבים החשמליים המרכזיים – קבלים, נגדים, מיצבים, לולאות השראה וכו'. אחראי על ויסות המתח וההגנה על המערכת.

2x5 Battery Pack מארז סוללות

שתי סדרות של 5 סוללות המספקות מתח כולל גבוה (לרוב 18–24 וולט) אחראי על אספקת האנרגיה לסליל.

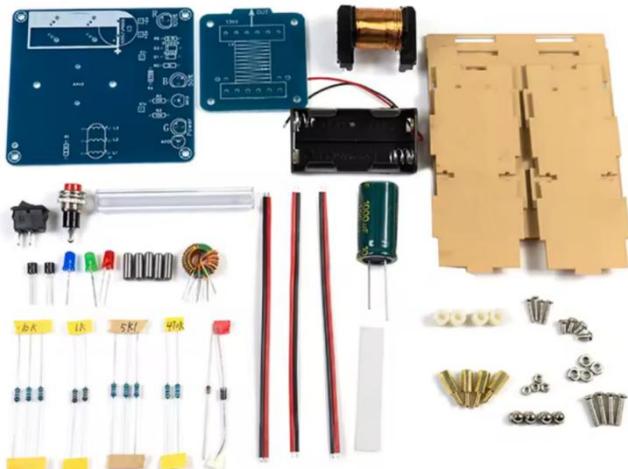
Magnetic Ring Inductors **חיישוקי השראה מגנטיים**

משמשים להחלהת הזרם, הפחיתה רעשים אלקטרוניים, ואחסון אנרגיה זמנית בمعالג.

Bombs כדוריות מתכתיים

גופים גליליים מתכת אשר מוכנסים לקנה. בזמן שיגור הם נמשכים דרך הסליל ויוצאים ב מהירות מהקצתה השנייה.

תמונה של הרכיבים ברובה האלקטרומגנטי:



עקרון פעולה:

- .1. המשמש מפעיל את מתג ההפעלה.
- .2. בלחיצה על כפתור השיגור, זרם מוזרם מהסולולות לסולנויד.
- .3. נוצר שדה מגנטי חזק בתוך הסליל.
- .4. גוף המטען נמשך פנימה וሞץ לאורך הקנה.
- .5. לאחר מכן, הגוף י יצא במהירות מהקנה – ללא חלקיים נעים מכניים.

הטבלה הבאה מציגה את כל רכיבי הערכה לרובה האלקטרומגנטי כפי שמופיעים בתמונה לעיל, כולל תיאור תפקודו של כל רכיב המערכת:

רכיב	תיאור תפקודי
לוח PCB ראשי	לוח המעל המרכזית שליטה מולחמים כל רכיבי האלקטרוניקה, כולל נגדים, קבלים וסלילים.
לוח PCB שני	לוח עזר – לרוב משמש לקישוריות בין סליל הירוי לבין הלוח הראשי או כקונקטור.
סולנויד (Solenoid Coil)	סליל נחושת דרכו עובר זרם ליצור שדה מגנטי חזק המאיץ את הקליע לאורך הקנה.
מחזיק סולולות (2x5)	מחזיק שני סטים של 5 סולולות – AA מספק מתח גבוה (לרוב 1.5V–24V) להפעלת הסולנויד.
פלטות מארז (Housing Panels)	חלקי הגוף המערכת – חיתוך לייזר מחומר מוקשה / (MDF) אקריליק (להרכבת המארז).
כפתור שיגור (Launch Button)	כפתור אדום להפעלת השיגור – מזרים זרם מיידי לסליל לצורך ירי.
מתג הפעלה (Power Switch)	mpsok הראשי להפעלה או כיבוי המתח הכללי של המערכת.
נורת לד	נורת חיוי – מדיליקה כשהמערכת פועלה או מוכנה לשיגור.
צינור פלסטיק (קנה)	משמש כמיסילה לירוי – דרכו עובר הקליע במהלך השיגור.
קליעים מתקטיים (Bombs)	גופים גליליים ממתכת אשר נמשכים אל תוך הסליל ויוצאים במהירות – משמשים להדגמה.

טורואידים / סילילים טבעתיים	רכיבי השראה מגנטיים לאגירת אנרגיה ולSHIPOR יציבות הזרם.
קבל אלקטROLITY	משמש לאגירת אנרגיה קצחה והעברת זרם חזק מיידי בעת השיגור.
נגדים (Resistors)	רכיבים להגבלת זרם, קביעת מתח, או תיאום בין רכיבים שונים במעגל.
חותמים ומוליכים	חותמים אדומיים/שחורים ל קישוריות חשמלית בין הלוחות, הסוללות, הסולנואיד והכפטורים.
ברגים, אומים ומרוחחים	חלקים מכניים להרכבת הלוחות, קיבוע הסוללה, בניית המסגרת וחיבור כללי של המערכת.

יתרונות המערכת:

יתרון

שימוש בטכנולוגיה אלקטرومגנטית מתקדמת

מודגים את עקרונות ההנעה המגנטית ללא צורך במכאניקה מסובכת.

**אין חלקים נעים מכניים
פעול פשוט**

מקטין בלבד, שקט יותר וב吐וח יותר (יחסית).
כפטור אחד לשיגור – מותאים למדידה, הדוגמה וניסויים.

גודל קומפקטי ונידות

ניתן להפעיל את המערכת בכל מקום ולשאת אותה בקלות.

רב פעימות

ניתן לשגר מספר פעמים עם אותו "פיצוץ".

הסבר

הסבר

עוצמת שיגור מוגבלת

בשל מגבלות מתח, סליל יחיד וגוף קל – העוצמה וה מהירות יחסית נמוכות.

יעילות אנרגטית נמוכה

חלק גדול מהאנרגיה מתבזבז כחום ולא תורם להאצת הגוף.

חימום רכיבים

שימוש רצוף עלול לגרום להתחממות יתר ולפגיעה במעגלים.

אין בקרת תזמון מתקדמת

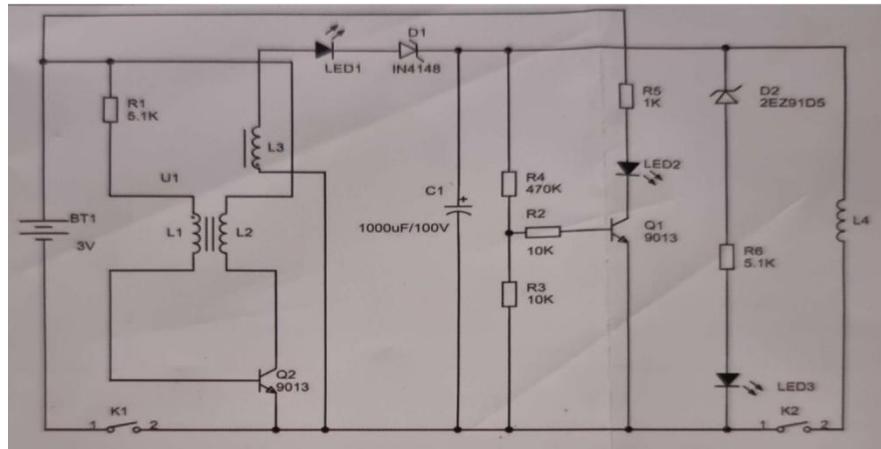
לא חייני מיקום – אין ניתוק מדויק של השדה ולכן תיתכן האטה בשלב מסוים.

טעינה איטית (אם משתמש בקבלים)

אם קיים שלב טעינת קבלים, יידרש זמן בין שיגור לשיגור.

רובה אלקטромגנטי הוא מערכת מותתקת ומשולבת – מדגים פיזיקה, חשמל, אלקטרוניקה ומכניקה בסיסית. הוא מהוות פלטפורמה חינוכית מצוינת להבנת עקרונות השראה מגנטית, בקרה, ואנרגיה. מבנהו הקומפקטי והשימוש הנגיש הופכים אותו לכלי לימוד ולמעבדה ניידת לכל דבר.

הסבר על מעגל חשמלי - רובה אלקטромגנטי



הסבר כללי:

המעגל משתמש ברכיבים כמו קבלים, סילילים (L) טרנזיסטורים, דיודות, ונוורות LED כדי לטען קובל בתמוך גבוה ואז לפרוק אותו דרך סליל אלקטромגנטי לצורך יצירת דחף (למשל, שיגור פרויקטיל מותכתי קטן).

הסבר לפיזורים:

אזור הספק:

BT1 - סוללה (V) 3 - מספקת את האנרגיה למעגל.
K1 - מתג הפעלה ראשי שמחבר את החשמל.

חלק טעינת הקובל:

C1 - קובל μF 1000 – 100V נתון בהדרגה דרך המעגל, ואוגר אנרגיה שתshawרר בבatteriah אחת לשיגור.
D1 - דיודה 1 – (N4148) מגינה על המעגל מזרם חוזר.
LED1 - מצבן שהקובל נתען.
L1, L2, L3 - סילילים כנראה חלק ממעגל מתנד/רוזוננס שמעביר אנרגיה לקובל.

חלק בקרה:

Q1, Q2 - טרנזיסטורים מסוג 9013 – מפעילים או חוסמים את מעבר הזרם.
R1-R6 - נגדים – קובעים את זרימת הזרם לבסיס הטרנזיסטורים, וכן שלוטים עליהם.
LED2 - חיובי נוסף למצב הפעלה של השלב הבא.

חלק הפעלת סליל השיגור:

D2 - דיודה Zener או – SCR (2EZ91D5) מפעילה שחרור מהיר של האנרגיה מהקובל.
L4 - סליל השיגור – כאשר הקובל נפרק דרכו, נוצר שדה מגנטי חזק שיכל לדחוף אובייקט מותכני קטן.
K2 - מתג הפעלה לשיגור.
LED3 - נדלק כאשר מתחבץ השיגור.

טבלת רכיבים:

רכיב	תיאור	תפקיד
BT1	סוללה 3V	מקור מתח
K1, K2	מתגים	K1-הפעלה, K2-шиגור
Q1, Q2	טרנזיסטורים 9013	מגברים/מתגים לברכה
C1	קбл μF 1000 - 100V	אגירת אנרגיה חשמלית
D1	דיודה N41481	הגנה מזרם חוזר
D2	דיודה Zener / SCR	פריקה פתאומית של הקבל
R1-R6	נדדים	קביעת זרמים לבקרה
LED1-LED3	נוריות חיומי	מראה מצבים פעולה שונות
L1-L4	סלילים	יצוב, תדר, שיגור

פירוט רכיבי המעגל ותפקידם האלקטרוני:

C1 : קבל האגירה (μF -100V-1000):

- **תפקיד ראשי:** המאגר הראשי של האנרגיה הפוטנציאלית החשמלית. הקבל נתען באופן יזום למתח נבוה (עד 100V) ומחזיק את האנרגיה זו עד לשלב השיגור.
- **ניתוח מקצעי:**
 - **קיבול :** ($C = 1000 \mu F$) הקיבול נמדד בפאראדים (Farads) וקובע את כמות המטען (Q) הנדרשת כדי להגיע למתח נתון.
 - **מתח נקוב :** (100V) זה המתח המקסימלי הבטיחותי שהקבל יכול לעמוד בו. במעגל דחף, חשוב שהמתח הנספג במהלך הטעינה לא עלה על ערך זה.
- **אנרגייה אగורה:** האנרגיה הפוטנציאלית המקסימלית האגורה בקבל נתונה על ידי הנוסחה

$$E = \frac{1}{2} CV^2$$

עבור קבל זה בטעינה מלאה (100V) :

$$E = \frac{1}{2} \cdot (1000 \cdot 10^{-6} F) \cdot (100 V)^2 = \frac{1}{2} \cdot 0.001 \cdot 10,000 = 5 \text{ Joules}$$

זהו אנרגיית השיגור המקסימלית העומדת לרשות המערכת.

D1: דיוידת הגנה (1N4148)

- **תפקיד ראשי: חסימת זרם חוזר (Reverse Current Blocking).** הדיוידה מודדת שהזרם יזרום רק מהמעגל המגביר המעגל שמשתמש ב- (L1,L2,L3) אל הקבל (C1).
- **ניתוח מקצעי:**
 - **הגנה מפני פריקה:** בזמן השיגור (כasher הקבל נפרק), המעגל חוויה שינוי דרמטי במתח וזרם גדול. הדיוידה מונעת מהזרם הפורק להזיק לרכיבי המגבר (כגון טרנזיסטורים) שעולמים להיות עדינים.
 - **דגם הדיוידה :** (1N4148) זהה לדיויד מיתוג מהירה וקטנה (Small Signal Switching Diode, עם זרם קדיימה מקסימלי נמוך יחסית (כ- 150mA) לכן, יש להניח שהיא ממוקמת **במעגל הטעינה בלבד**, היכן שהזרם קטן יחסית (שכן הקבל נתען בהדרגה, ולא חלק מעגל הפריקה של ה-J5-שים נדרש דיוידה חזקה בהרבה (כגון דיוידת UF4007 או FR107).

נוירית חיוי LED1

- **תפקיד ראשי: אינדיקציה חזותית ל乓בוב הטעינה.**
- **ניתוח מקצעי:**
 - נוירית זו מעידה כי המתח על הקבל (V) הגיע לרמה מספקת או מלאה. היא מחוברת בדרך כלל **במקביל** לקבל בטור עם נגד הגבלת זרם מתאים. כיוון שמתהה הטעינה הוא 100V, יש צורך בנגד הספק גובה או במנגנון מיתוג (כגון נוירית ניאוון קטנה או טרנזיסטור) כדי להבטיח שהיא לא תישרף בגל המתהה הגבוה.

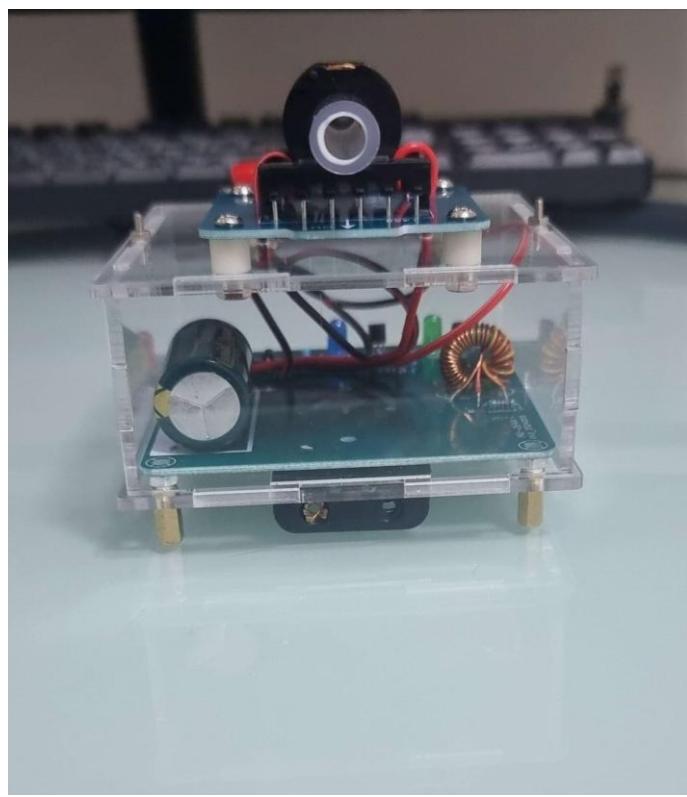
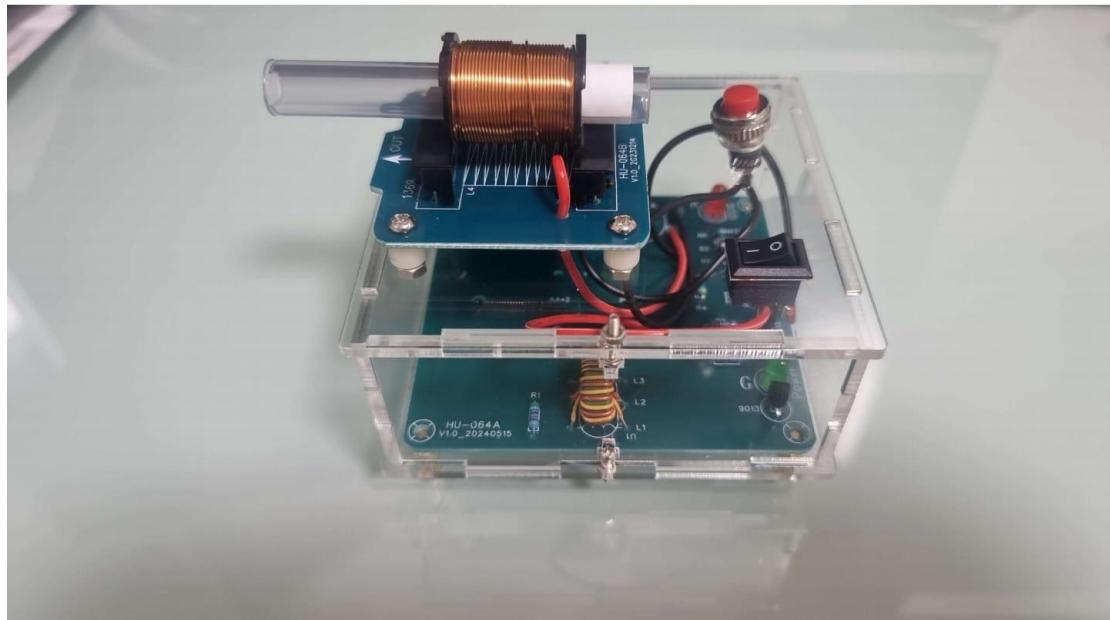
L1, L2, L3 : סילילים (Inductors/Coils)

- **תפקיד ראשי: רכיבים אלו הם ליבת **מעגל המגביר** (Boost Converter) או **ממיר מתח מורם** (Flyback Converter). תפקידם הוא להפוך מתח כניסה נמוך (כגון סוללה של 12V – 3V למתח גובה של 100V)**
- **ניתוח מקצעי:**
 - **ממיר DC-DC :** סילילים אלה משמשים ליצור שדה מגנטי. על ידי מיתוג מהיר של זרם דרך סליל ראשוני (L1) או (L2) באמצעות טרנזיסטור, נוצרת השראה הדזית שמייצרת מתח גובה בסליל משני L2 או (L3).
 - **דחית זרם ישיר :** המעגל המגביר פועל על העיקרונו של **שינוי מהיר בשטף המגנטי**, שיוצר מתח גובה על פי חוק פאראדי. המתח הגובה הזה מיושר באמצעות דיוידה (אולטי, D1, אם כי D1 היא חלשה מדי, או דיוידה נוספת) ונשלח לטיענת C1.

סיכום ומסקנה (הקשר המערבי):

- המערכת יכולה מתארת **מחולל דחף אלקטромגנטי**, העובר את שלבי הפעולה הבאים :
1. **הגברה :** המתח הנמוך מוגבר ל- 100V באמצעות מעגל המתנד/הממיר (L1,L2,L3).
 2. **טעינה :** המתח המוגבר עובר דרך דיוידת הגנה (D1) ונטען לתוך **קבל האנרגיה** (C1).
 3. **МОכנים :** כאשר C1 מגיע ל- 100V (אוגר J5), נוירית החיווי (LED1) נדלקת.
 4. **שיגור :** בשלב זה, באמצעות מתג מיוחד שאינו מצוי בראשימה, כגון טרנזיסטור SCR או MOSFET בעל זרם גבוה), משוחררת האנרגיה האגורה ב- C1 **בבת אחת** לתוך סליל שיגור (העומס), מה שמייצרת את הדחף האלקטרומגנטי הדרוש לשיגור.

תמונות של התוצאה הסופית של הרובה:



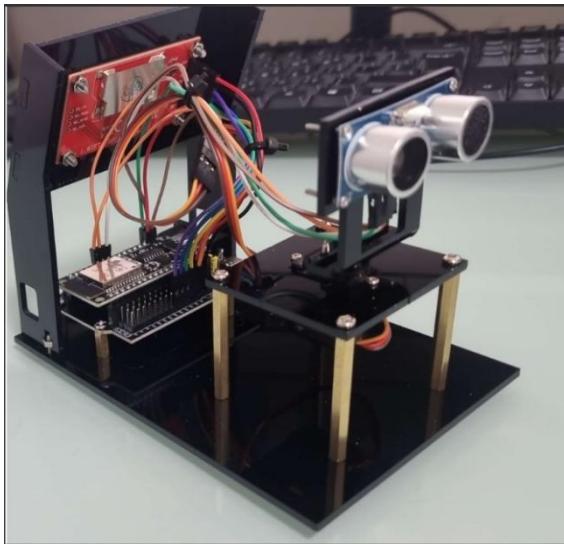
קישור לסרטון תיעוד בו אני מרכיב את המערכת של הרובה האלקטרומגנטי ומדגים את אופן
פעולתו וכייד מtbody היריה (יש לחוץ CTRL + על קישור של הסרטון)

<https://youtu.be/5IMt-HXeClk?feature=shared>

מיני רdar (Mini Radar)

המטרה והעקרון:

המטרה של הפרויקט הייתה ליצור מערכת שתסורך את הסביבה הקרובה שלה, תזהה מכשולים ותציג אותם על גבי מסך, בדיקות כמו מכ"ם אמייתי.



הרכיבים והתפקידים שלהם:

כמו בכל פרויקט מוצלח, גם המערכת זו מורכבת מכמה חלקים שפועלים יחד :

- המוח : זהו המיקרו-בקר, הלב והמוח של המערכת. הוא שולט בכל הרכיבים ומבצע את החישובים המורכבים בזמן אמיתי.
- העיניים : החישון האולטרה-סוני, שמצויר קצר שתי עיניים קטנות. עין אחת משדרת פולס קולי שאנו חזו לא יכולים לשם, והשנייה קולטת את החזר השוחרר. הנתון הקרייטי שאנו מקבלים הוא הזמן שלקח לפולס הזה לצאת ולהזoor.
- הזרוע : זהו מנוע השרו. הוא מאפשר לחישון שלנו לנעו מצד לצד, מה שנutanן למערכת את היכולת לסרוק שטח שלם ולא רק נקודה אחת.
- התוכנה : כל החלקים הפיזיים לא שווים כלום בלי הקוד שכתבתי. התוכנה היא זו שמנחה את מנוע השרו לנעו בزواיות שונות (למשל, 5 מעליות בכל פעם), ובכל עזירה היא מורה לחישון למדוד את המרחק לעצם. לאחר מכן, הקוד שולח את נתוני המרחק והزاوية למסך חיצוני או למחשב.



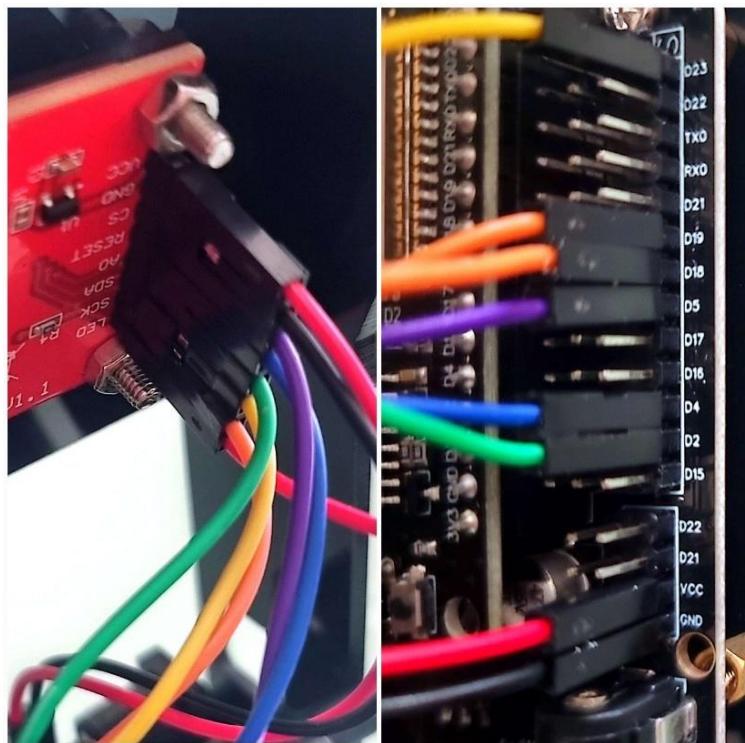
איך זה עובד בפועל?

התהליך פשוט ואלגנטי: המנוע מסתובב בזווית מסוימת, החישון מודד את המרחק לעצם הקרוב מזונים לקוד. התוכנה לוקחת את הנתונים – **הזווית והמרחק** – ביותר, ושני הנתונים האלו ומציגו אוטומטית ויזואלית על מסך. ככל שהמכשול קרוב יותר, כך הנקודה שתופיע על ה"מכוון" תהייה קרובה יותר למרחוק. על ידי חזרה על התהליך בכל זווית לאורך הקשת של 180 מעלות, אנחנו מקבלים תמונה שלמה של הסביבה שנמצאת מולנו.

הפרויקט הזה מדגים שימוש פשטוט של רכיבים יכול ליצור מערכת מורכבת ויעילה, והוא פותח את הדלת ליישומים מרחוקים כמו רובוטים אוטונומיים, מערכות אבטחה ביתית ועוד.

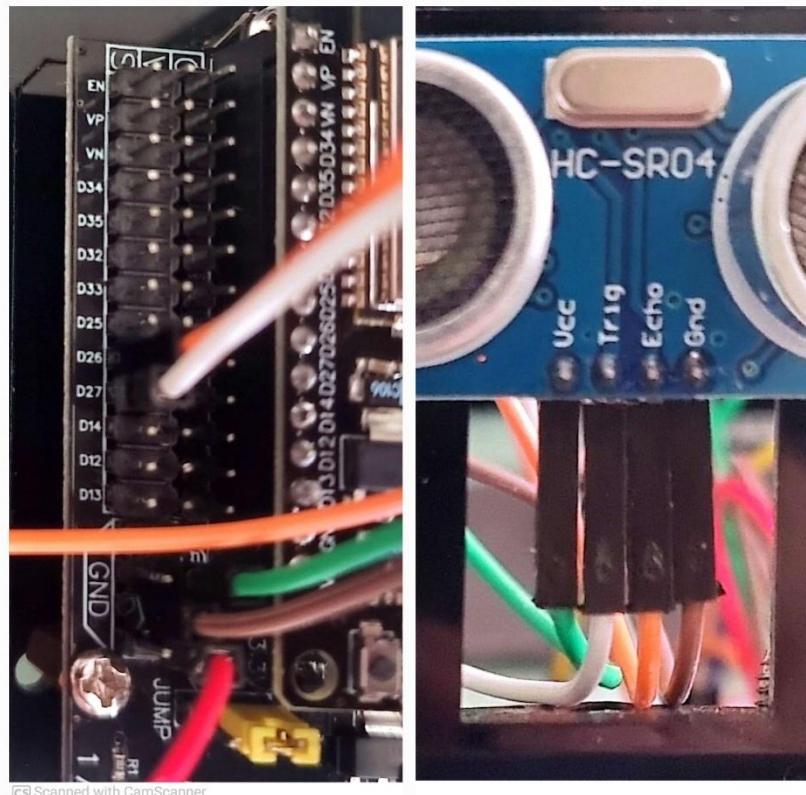
נראה את אופן צורת החיבור של המძק והחישון האולטרסוני אל המיקרו בקר בזורה ברורה ומקורבת יותר:

[המסך הקטן מחובר באופן הבא:](#)



- ניתן לראות ש-VCC במסך מחובר ל-VCC שבמיקרו בקר.
- היפין GND שבמסך מחובר ל-GND שבמיקרו בקר.
- רgel CS שמוצגת במסך מחוברת לפין D5 שבמיקרו בקר.
- רgel ה-RESET שבמסך שלגנו מחוברת לפין D4 שבמיקרו בקר.
- רgel 5 A יכולה להיות מחוברת לרgel אנלוגית או>Dיגיטלית תליי בפקודות שנרצה להשתמש, אני חיבורתי רgel זאת לרgel>Dיגיטלית/Sifرتית 2 שבמיקרו בקר.
- רgel SDA שהיא למעשה הרgel SPI מאחר והמסך מתקשר עם המיקרו בקר דרך תקשורת טורית SPI, ישנו בלבול משום שהוא רגליים שרגל SDA שימושית בתקשורת I2C אך יבואנים סיניים החלו לייצר מסכים ולרשום גם במסכים המתזרים דרכם תקשורת SPI את השם של הרgel SDA והוא מאפיינת את התקשרות שבין המסך למיקרו בקר.
- רgel SCK מחוברת לפין D18 (החוות השני הוא של המנוע סרבו שמחובר לפין D19).
- למעשה רgel SCK זהה הרgel שמעבירה את אותן הש�ון.
- רgel ה-LED מחוברת ל-3.3V בצד השני של המיקרו בקר.

נראה את החיבור של החיבורן האולטרסונייק:



- רgel ה-TRIG מחוברת לפין D27 שבמיקרו בקר.
- רgel ה-ECHO מחוברת לפין D26 שבמיקרו בקר.
- המתח VCC שבחישון מתחבר לפין 5V שבמיקרו בקר משומש לחיבור זה בדומה למונע סרווו שלנו עובדים עם 5 וולט והמיקרו בקר ESP32 עובד על 3.3V.
- הארקה GND מחובר כמובן לאזמה GND שבמיקרו בקר שלנו.

קישור לסרטון בו אני מרכיב את המערכת של המיני רадאר. (יש ללחוץ Ctrl + אוז על הקישור):

<https://youtu.be/TOxmNhN4LiU>

תיעודים-בדיקות

תיעוד של אופן פעולה החישון האולטרסוני ומדידת מתחים 5.9.25:

בתיעוד זה נוכל לראות את אופן פעולה החישון האולטרסוני, מדדתי את המתח שהספק מקבל מהמיקרו בקר ESP32 ובנוסך בדكتי כמה מתח המיקרו בקר מוצאי מהפין של ה- 3.3V האם הוא באמת מוציא מתח זה או שלא, בבדיקות קיבلت שחקל תקין וחישון קיבל טיפה פחותה מתח 5 וולט וזה עלול להיגרם ממספר סיבות הבאות :

מתח ה- USB הסטנדרטי מוגדר להיות 5.0V, אך יש לו **סבירות** (Tolerance). קריאה של 4.5V נובעת ככל הנראה מהשילוב של הגורמים הבאים :

1. מפל מתח בcabbel ה-

זהו הגורם העיקרי :

- **התנגדות הקabel:** לכל cabbel, גם הקצר והaicotti ביותר, יש התנגדות חשמלית. כאשר ה- ESP32 מחוברים וצורכים זרם (הס **העומס**) נוצר **מפל מתח** לאוריך הקabel, לפי חוק אוהם : $V_{drop} = I \times R$.
- **cabbel USB דקים וארכיים:** cabbelים דקים יותר (בעלי מס' AWG גובה יותר, כמו AWG 28 במקום AWG 20 עבה יותר) או ארוכים, הם בעלי התנגדות גבוהה יותר, מה שմגביר את מפל המתח.
- **הקריאה שלך נушה תחת עומס:** אם מזנת 4.5V בזמן 4.5V, חישון האולטרסוני ומעגל ה- Wi-Fi שלו פועלן, המתח יורך עקב צריכת הזרם.

2. סבירות תקן ה-USB:

תקן USB 2.0 מאפשר למתח לרדת עד 4.4V בחיבור ל"Hub Port"- (SKU פחות חזק) או לכל הפחות 4.75V בחיבור לשקע ורגיל, עד לצריכה המקסימלית המותרת. ערך של 4.5V נופל בטוויה הסביר, במיוחד אם אתה משתמש בcabbel ארוך או ביציאת USB של מחשב נייד או רכזת(Hub).

3. מפל מתח ברכיבי לוח ה-ESP32:

המתח עבר דרך מספר רכיבים על לוח ה- ESP32 שלך :

- **הגנת זרם יתר:** (Over Current Protection) לוחות רבים מכילים נתיק הנitinן לאיפוס או רכיב ההגנה אחר שמוסיף התנגדות קטנה וגורם למפל מתח קל.
- **דיודת הגנה:** (Protection Diode) דיודה הגנה בכניסה ה-USB- יכולה לגרום למפל מתח של כ- 0.2V עד 0.7V תלוי בסוג.

לסיום, הקריאה של 4.5V היא תוצאה של הזרם שצורך ה- ESP32 (והחישונים המתחברים אליו) בשילוב ההתנגדות של cabbel ה- USB ורכיבים שעל הלוח. אם הלוח עובד יcitיב, המתח הזה תקין לחלוtin עבורה.

את המדידות שייצאו ניתן לראות בהסבר על החישון האולטרסוני בספר זה איפה שאינה מסביר על החישון ואופן פועלתו בסוף ההסבירים הנסטי תחת הכותרת "מדידות מתח ואופן פעולה" מצרף כאן סרטון מיפוי בו אתיודתי את אופן הפעולה ואת המדידות וכמוון אכנים את הסרטון לאתר שלי תחת כתורת בדיקות.

קישור לסרטון: <https://youtu.be/qJxPFXFTNug>

מדידות מתח ואופן פעולה:

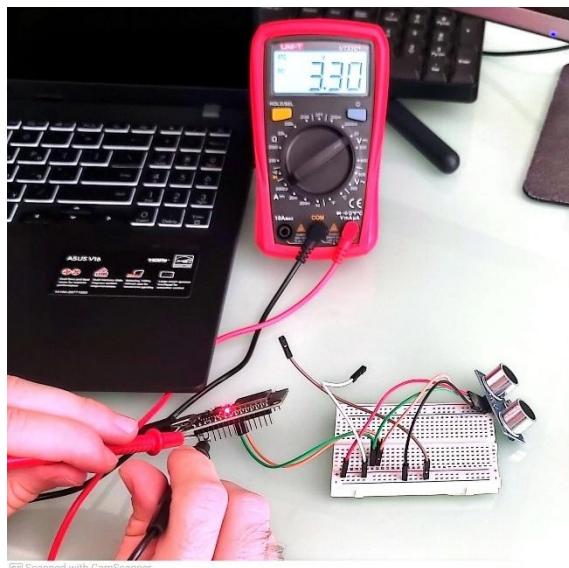
מדידת מתחים של החיבור האולטרסונייק אל המיקרו בקר ESP32, אראה את המדידה אל המתח אליו אני לחבר את החיבור עצמו אל המיקרו בקר. בנוסף לכך אראה את המתח שMOVEDה הפין של ה- 3.3V, האם הוא באמת מוצריה מתח זה:

נראה תמונות:

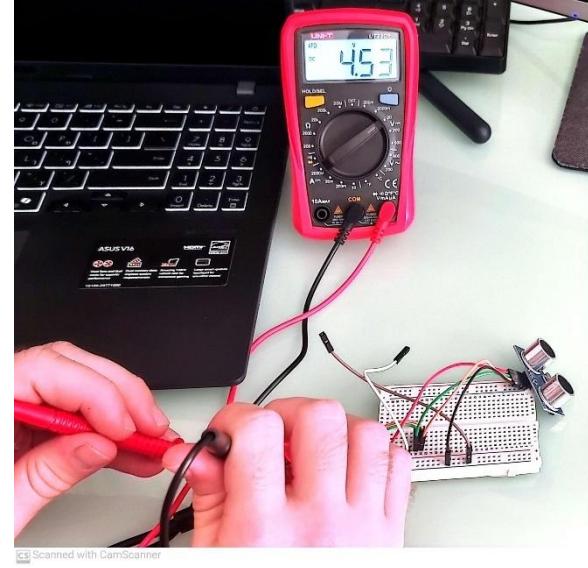
```

1 // חיבור תזוזאת בסיריאל HC-SR04-ESP32
2 // פולס-Trig פולס-Echo פין ח
3 #define TRIG_PIN 27
4 // חזרה את חווילס פין ח
5 #define ECHO_PIN 26
6 // משגנה לשינוי הזמן של גזען לפולס גזודור
7 long duration;
8 // המרחק בcentimeters
9 int distance;
10
11 void setup() {
12     // 初始化 תקשורת סיריאלית ליצירת_BTוויזוטאות
13     Serial.begin(115200);
14
15     // גדרת פינים
16     pinMode(TRIG_PIN, OUTPUT);
17     pinMode(ECHO_PIN, INPUT);
18
19     Serial.println(" מדידת מרחק עם חיישן אולטרסונייק ESP32");
20 }
21
22 void loop() {
23     // ל פולס קוצר פולס-Trig
24     digitalWrite(TRIG_PIN, LOW);
25     delayMicroseconds(2);
26     digitalWrite(TRIG_PIN, HIGH);
27     delayMicroseconds(18);
28     digitalWrite(TRIG_PIN, LOW);
29
30     // כריהת מתח Echo HIGH
31     duration = pulseIn(ECHO_PIN, HIGH);
32
33     // חישוב מרחק בס"מ
34     distance = duration * 0.034 / 2;
35
36     // הדפסת המרחק במסך הסיריאלי
37     Serial.print("המץ: ");
38     Serial.print(distance);
39     Serial.println(" ס' );
40
41     // המתנה קצרה בין מדידות
42     delay(500);
43 }
```

המתח שMOVEDה המיקרו בקר מהפין V_{DD}:



המתח שאני מספק לחיפוי:



תיעוד של אופן הפעולה 20.9.25:



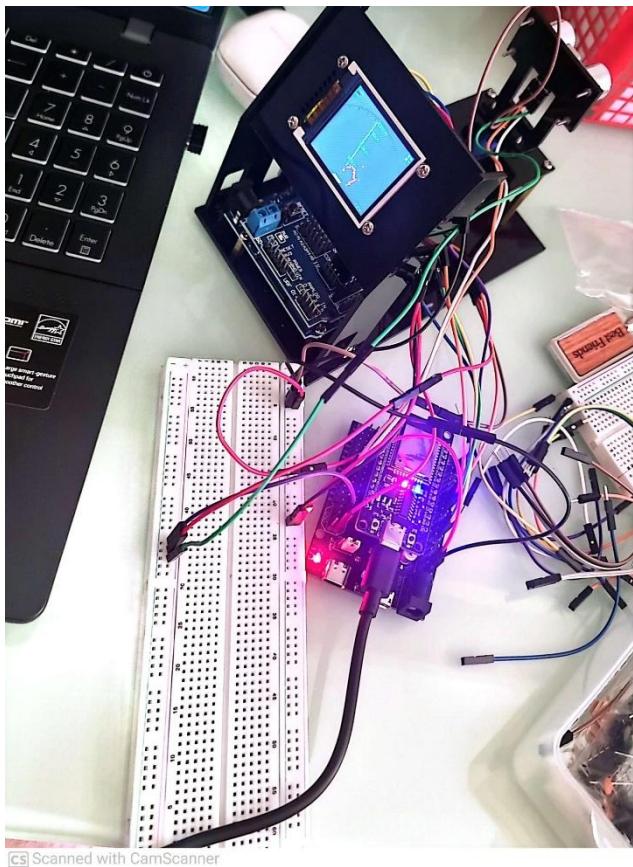
Scanned with CamScanner

אחד הביעות הגדולות שיצא לי להתמודד בפרויקט זה היה להפעיל את מערכת הרdar.

נתקלתי בבעיות כמו למשל ספירות לא תואמות הגדרת פורטים לא נכונים. בעיות אלו פתרתי על ידי חקירה על הפונקציונליות של המיקרו בקר ובנוסף על בסיסו הפורט בעזרת הטרמינל במחשב, אלו היו הפוצדרות המסבירות איך המערכת בוחרת מאחר והן לימדו אותי להתמודד עם בעיות כאלו לפעם הבאות במידה ואתקל בהן.

נוסף על כך אחד הביעות שיצא לי להתמודד היה הפינים שהוגדרו להיות "בעיתיים" לשימוש, הכוונה שבמיקרו בקר אי אס פין פינים בהם כדי להשתמש משום שנוחים יותר עבר פעולות מסויימות ויישם ככלו שמדובר בעובדה ספציפית יותר. לדוגמה יכול לומר שהמנוע שלי לא עבד כאשר חיבורתי אותו לפין 15 ואז שיחקנתי קצת עם הפינים ומתי שחברתי SERVO לפין 19 הוא התחליל לעבוד והסתنصرן בעובדה יחד עם המשך והחישון אולטרא סוני שלמעשה נמצא עליו. בנוסף אחד הביעות הגדולות ביוטר היה להפעיל את המסק. גרפיקה זה צעד גדול בפרויקט ויחסית מסוובך, לא כל הפונקציות תואמות למיקרו בקרים ספציפיים וצריך לנשות כמה פונקציות על מנת למצוא את המתאימה ביוטר עבור אופן הפעלה שלו.

마וחר ואני יוצר תמונה הרdar שפועל בזמן אמת אז עלי לא להשתמש רק בתצוגה רגילה אלא ממש בתצוגת עיצוב גרפי חדשני ממעני להשתמש בפונקציות ייחודיות עבור פעולות אלו. בסופה של דבר עם קצת כישלונות ולמידה הצלחתית להבין יכון טעיתי בכל הניסיונות הללו ולמדתי מזה המונ.



Scanned with CamScanner

גם כאן בתמונה ניתן לראות את הניסיון הפעלה על מיקרו בקר אי אס פי.

בהתחלת חיבורתי הכל דרך מטריצה ולבסוף ארגנטית את זה על המערכת עצמה כדי לשפר את הנראות והנוחות במהלך השימוש בה.

בתמונה קצת קשה לראות את הרדאר שנוצר על המסך וכמובן שהתמונה לא משדרת את התזוזה של המנוע סרוו ואת הקליטה של החישון אולטרסוני אך כדי להמחיש זאת אני צירפתי לכטוטון שלי לבדוק את אופן הפעולה של המערכת.

בסרטון אציג את הבניה כולה של המערכת ואת אופן פועלתה בסוף הסרטון כדי שנitin יהיה לראות ולעקוב אחר השלבים צורת הבניה של המערכת.

בנוסף הסרטון יעלה לאתר שניtin לגשת אליו דרך ספר זה בקישור שאציג שייהה תחת הכותרת **"קישור לאתר"**

תיעוד של אופן פעולה הרדאר 21.9.25 :

בניסיון זה בדקתי את סימון הנקודות הצהובות שהן מופיעות עברו אובייקטים רחוקים מ-100 סנטימטר. הנקודות האדומות שופיעו על הרדאר אלו נקודות שמאפיינות אובייקטים הנמצאים בתחום של 0 עד 100 סנטימטרים מהחישון אולטרסוני.

תיעדתי ניסיון זה והוספתי סרטון שמראה כיצד בדקתי זאת הסרטון טיפול הפוך החלק של הבדיקה לקראת הסוף אבל זה לא פרק זמן ארוך וניתן להבחן בכל זאת את הנקודות הצהובות. אני מצרף כאן קישור לסרטון שלי שנמצא ביוטיוב וגם הכנסתי את הסרטון תחת הקטגוריה "בדיקות" בתוך האתר שיצרתי שיופיע בעמודים האחרונים של הספר פרויקט שלי.

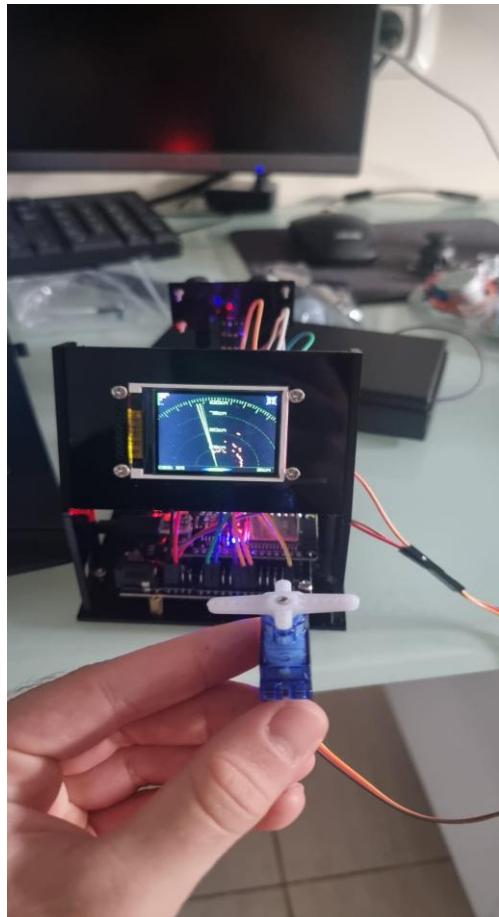
קישור לסרטון תיעוד של ניסיון יצירט ובධיקה של הנקודות הצהובות שמדוברות אובייקטים שנמצאים למרחק מעל 100 סנטימטרים כולל מהחישון האולטרסוני:

[קישור הסרטון:](https://youtu.be/8BiuvFxFUo20)

תיעוד של אופן הפעלה כאשר חיבורתי סרוו נוסף שהסיבוב של התותח 23.9.25:

בניסוי זה הקפנתי צעד יחסית מושם בפרויקט שלי משומש להצלחתו לבצע סנכרון בין שני המנועי SERVO שיש לי במערכת שלי. ה-SERVO הראשון אחראי על לסובב את החישון האולטרסוני שסורק את הסביבה בטוחה של 180 מעלות החל מ-0 מעלות ועד ל-180 מעלות. המנוע ה-SERVO השני שלי אחראי למשעה על הסיבוב של המשטח עליו לחבר את הרובה האלקטרומגנטי שלי. ברגע שמתגלו נקודות אדומות שבמשמעותן שהאובייקט נמצא בתוך של המנוע O SERVO השני או המנוע ה-SERVO השלישי ישירות לפועל מהותו המוקם בו 100 סנטימטר ופחות מזה המנוע O SERVO הראשון שמסוגל את החישון האולטרסוני נמצא מתחת וברגע שייתגלת שוב נקודה אדומה שמסמנת על אובייקט או תבצעי את הנסכרון של המנוע ה-SERVO השני עם הראשון יחד. מתקדמיים יותר כרגע בצעתי את הנסכרון של המנוע O SERVO השני עם הראשון יחד.

נראה תמונה ואסביר את החיבורים הנוספים שבעטי למיקרו בקר על מנת שהמנוע O SERVO השני יתחליל להסתובב:



זאת למעשה התמונה בה אני מחזק את המנוע O SERVO השני, למעשה אי אפשר להבין מה מערכת אכן עובדת מתוך פוטו ולכן חשבתי איך סרטון ביוטיוב וקיים בתוך הספר וכמוון שאכנים את התיעוד לקטגוריה הבדיקה שבאתר שלי.

המנוע SERVO מחובר לרgel D25 שבמיקרו בקר ESP32 ואת שאר החוטים שבמנוע SERVO צירפתי את החוט האדום ל-GND ואת החוט החום חיבורתי לאדמה VCC.

בקוד כל מה שעלי היה להוסיף זה את ההגדרה של ה-SERVO הנוסף ולאיזו רגל הוא מחובר במיקרו בקר, ובנוסף לכך היה עלי להוסיף בתנועה הלוֹק וה坦ועה חזור בתוך התנאי AiFa שרשום המרחק קטן מ-100 שזה לעומת הנקודות אדמות מה שאומר שהאובייקט נמצא בטוחה של 100 סנטימטרים ומטה SERVO השני או במידה והמרחק נמצא פקודה שתפעיל את המנוע O של המנוע O SERVO השני פוסקת והמנוע עצם.

מצרך קישור לסרטון בערוץ היוטיוב שלו:

<https://youtu.be/eeDQquCGN5I>

תיעוד של אופן פועלות המסך טאץ' המסך השני בו אני משתמש בפרויקט 27.9.25

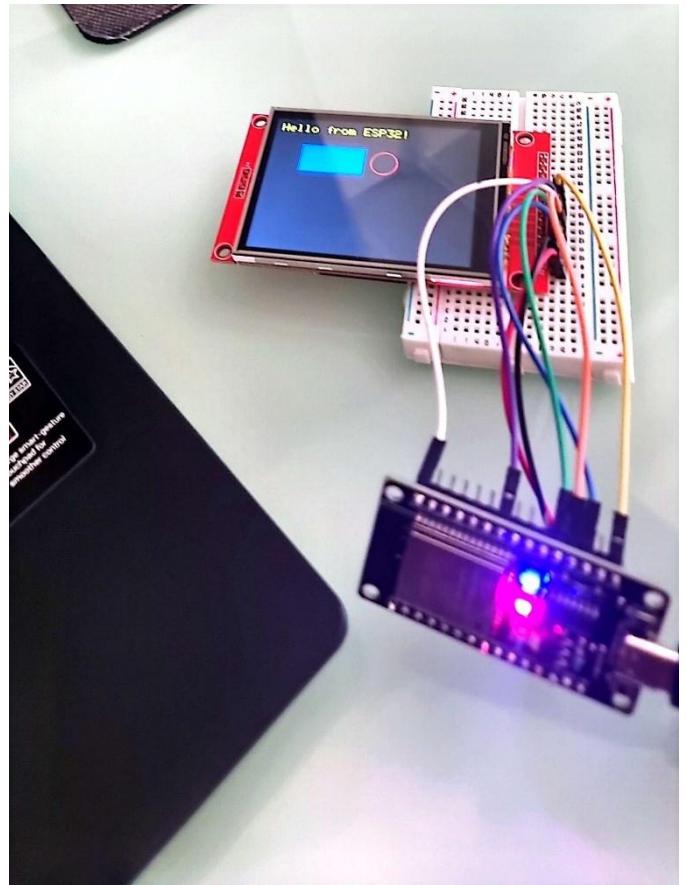
בתיעוד זה אציג את אופן פועלות המסך השני זהו מסך טאץ' בו משתמש כדי להראות את מצב התאורה והטמפרטורה ואוכל לעבור בין מצבים שונים שאחלייט לצרף על המסך.

הקישים איתם התמודדתי על מנת לצרף את המסך השני עיביר ייחד עם פועלות המסך הראשון היי בעיקר מאופיינים בהגדרת ספריות אחרות בחן המסך שלו לא תמק, לאחר זמן ממושך וחקירה לגבי הנושא מצאתי את הספריות המתאימות שעלייהן הרחבתי והסבירתי בספר זה כאשר הסברתי על המסך טאץ'.

מצרף קישור לסרטון בו אני מבצע מדידת מתחים יחד עם אופן פועלות המסך השני. הקוד שצרכתי זהו קוד בסיסי שמדפיס את המשפט Hello from ESP32 ומתחתיו יש מלבן שכלי כחול ועיגול שהיקפו אדום והפניהם הוא במצב צבע (בצבע שחור). זהו אותו הקוד שהציגתי בהסביר על המסך טאץ' (המסך השני בפרויקט שלי), لكن ניתן לראותו בצורה מלאה וברורה יותר בספר זה תחת הכותרת והסבירים של "מסך טאץ' TFT 9341".

נראה כיצד אופן פועלות המסך ומדידת המתחים בפנים אליהם אני לחבר את המסך:

```
1 #define TFT_CS 15 // Chip Select (CS)
2 #define TFT_DC 2 // Data/Command (DC)
3 #define TFT_RST 4 // Reset (RST)
4 // הכללת הספריות
5 #include <Adafruit_GFX.h>
6 #include <Adafruit_ILI9341.h>
7 // יזרור אובייקט המסך
8 // אנו מנהירים לו את הגדרות הפיינרים הנדרשים
9 Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_RST);
10
11 void setup() {
12     Serial.begin(115200);
13     // אתחול המסך
14     tft.begin();
15
16     // הגדרת כיוון המסך (0, 1, או 3)
17     tft.setRotation(1);
18
19     // ניירוי המסך ומיצורי בזבוב שחזור
20     tft.fillRect(0, 0, 160, 160, ILI9341_BLACK);
21
22     // הגדרת גודל וצבע הטקסט
23     tft.setTextSize(2); // 2 גודל טקסט
24     tft.setTextColor(ILI9341_YELLOW); // צבע טקסט: צהוב
25
26     // (x, y) = (10, 10) הדפסת טקסט במיקום
27     tft.setCursor(10, 10);
28     tft.print("Hello from ESP32!");
29
30     // (רוחב, גובה, צבע, x, y) צירור מלמן
31     tft.fillRect(50, 50, 100, 50, ILI9341_BLUE);
32
33     // (רוחב, גובה, צבע, x, y) צירור מעגל
34     tft.drawCircle(180, 75, 20, ILI9341_RED);
35 }
36 void loop() {
37 }
```



מתמונה זו ניתן לראות שהדפסתי על המסך דברים בסיסיים על מנת לראות שהוא פועל יחד עם המיקרו בקר ESP32 מה שניתן לראות למעשה על המסך מה שמודפס עליו זה Hello from ESP32 בנוסף מלמן בזבוב כחול ומעגל שرك ההיקף שלו צבוע בזבוב אדום ואני צבעו מבפנים.

הקוד הוא פשוט ובסיסי רק כדי להמיץ את אופן הפעולה של המסך יחד עם המיקרו בקר ולראות שהוא מדפיס ואכן מתבצעת תקשורת בין המסך למיקרו בקר.

כעת נראה את מדידת המתחים:

מדידת מתח הנכנס לפין LED:



Scanned with CamScanner



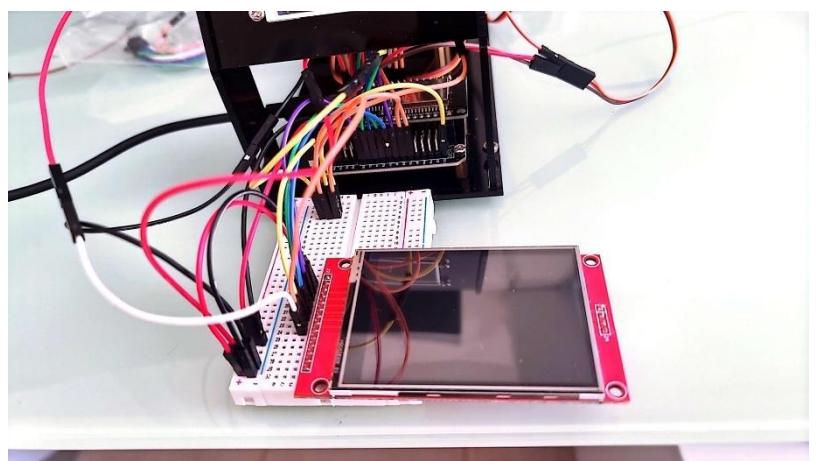
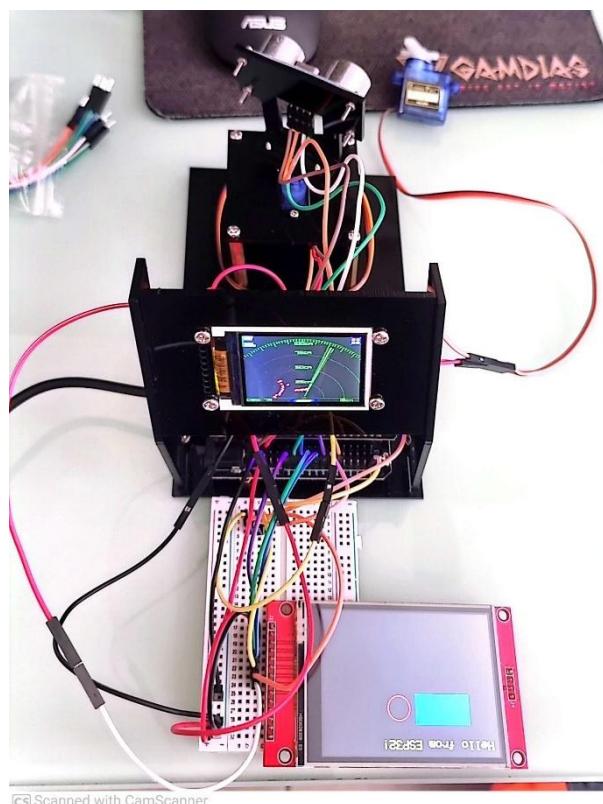
Scanned with CamScanner

ניתן לראות הסביר מדוע המתח אינו 5 וולט אלא בקרוב 4.5 וולט. בחישון אולטרסונייק כאשר ביצעתנו מדידה של המתחים מהפינים האלו במקירו בקר אליו נכנסים הפינים GND ו- VCC . ברגע שמדדתי בfin - VN קיבلتني גם מתח דומה למתח זה ובfin ה- 3.3V קיבلتני גם בקרוב למתח 3.3V חקרתי מדוע תופעה זאת קורת והסבירתי בטעוד הראשן תחת הכותרת "תיעודים- בדיקות" בתיאוריך 5.9.25 , הסברתי מדוע תופעה זאת עלולה לקרות ומה הסיבות שגורמות ל"אייבוד" המתח בכינסה VN .

קישור לסרטון: <https://youtube.com/shorts/a2xaMos41Qk?feature=share>

תיעוד של חיבור המסך השני למערכת כולה ואופן פועלות המערכת 28.9.25:

כأن תיעדי את אופן החיבור של המסך השני למערכת כולה, המסך מציג בניסיון זה את אותו הדבר כמו שהראה בתייעוד הקודם אך ניסוי זה היה מדרגה משמעותית עברית בהתקדמות הפרויקט יצא את אופן פועלות המערכת בסרטון נוסף וכמוון יצא מס' טרנומות להמחשה כדי שנייתן יהיה לראות זאת. הסרטון מראה בצורה ברורה ומקצועית יותר את אופן הפעולה, ניתן לראות את התזוזה של המנוע SERVO ואת ריצת הרזאר במסך השני.

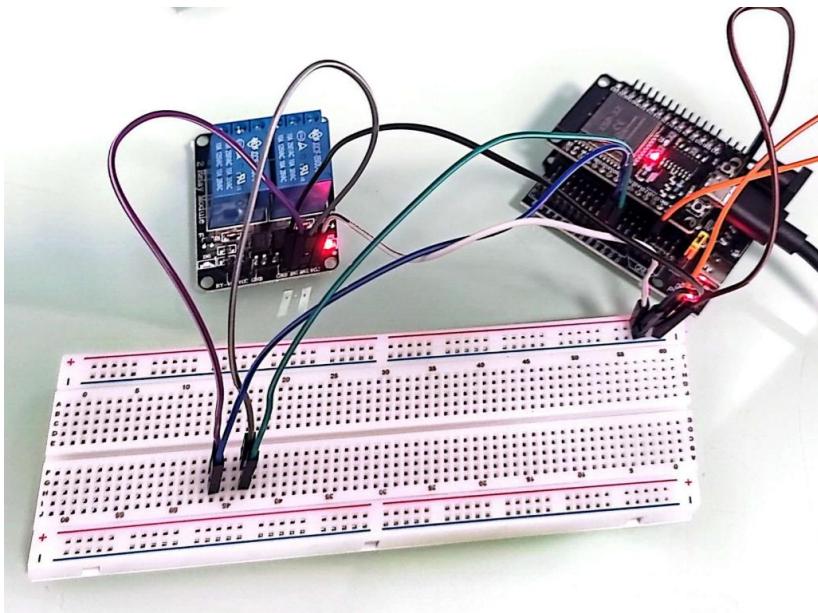


שני המסכים עובדים ב프וטוקול SPI لكن היו לי שני רגילים משותפות בשני המסכים. בעזרתו שני הרגילים האלו המסכים מתקשרים בפרוטוקול SPI עם המיקרו בקר. ניתן לראות הסבר מורחב בנוגע לפרוטוקול זה בספר שלי תחת הכותרת "פרוטוקול SPI"

[קישור לסרטון:](https://youtube.com/shorts/rToKhwszY90?feature=share)

תיעוד של אופן פעולה של הממסר ההפוך 3.10.25:

בניסוי זה חיבורתי ממser 5 וולט כפול בעזרת מטריצה אל המיקרו בקר ESP32. נתתי קוד בסיסי שמדפיס לי במניטור שהמסר הראשוני דולק ואז לאחר זמן קצר נכבה, ובנוסף המסר השני גם כן פועל לזמן מסוים ולאחר מכן נכבה. ההודעות מופיעות ומוצגות במניטור. כמו כן, ניתן לראות שהלדים פועלים על המסר וattiיחס לכך גם במהלך הניסויים הבאים.



כאן ניתן לראות את החיבור שביצעתך ייחד עם המטריצה אל המיקרו בקר.

המטריצה שמשה ככלי עזר על מנת לסדר את חיבור החוטים בין המסר ההפוך אל המיקרו בקר ESP32.

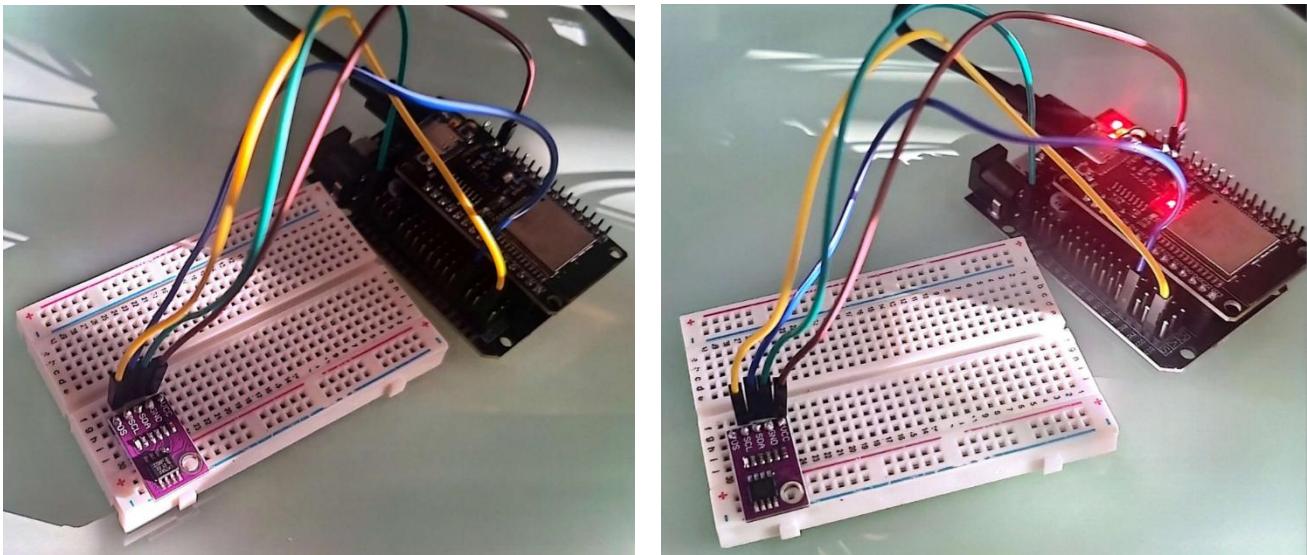
הקוד והפלט:

```
Turning RELAY 1 OFF
Turning RELAY 2 ON
Turning RELAY 2 OFF
Turning RELAY 1 ON
Turning RELAY 1 OFF
Turning RELAY 2 ON
Turning RELAY 2 OFF
Turning RELAY 1 ON
Turning RELAY 1 OFF
Turning RELAY 2 ON
Turning RELAY 2 OFF
Turning RELAY 1 ON
Turning RELAY 1 OFF
Turning RELAY 2 ON
Turning RELAY 2 OFF
```

קישור לסרטון:
<https://youtu.be/T7QlbvXVhSo>

```
1 const int RELAY_PIN_1 = 26;
2 const int RELAY_PIN_2 = 27;
3
4 const int DELAY_TIME = 2000;
5
6 void setup() {
7     pinMode(RELAY_PIN_1, OUTPUT);
8     pinMode(RELAY_PIN_2, OUTPUT);
9
10    digitalWrite(RELAY_PIN_1, HIGH);
11    digitalWrite(RELAY_PIN_2, HIGH);
12
13    Serial.begin(115200);
14    Serial.println("Dual Relay Test Started...");
15}
16
17 void loop() {
18     Serial.println("Turning RELAY 1 ON");
19     digitalWrite(RELAY_PIN_1, LOW);
20     delay(DELAY_TIME);
21
22     Serial.println("Turning RELAY 1 OFF");
23     digitalWrite(RELAY_PIN_1, HIGH);
24     delay(DELAY_TIME);
25
26     Serial.println("Turning RELAY 2 ON");
27     digitalWrite(RELAY_PIN_2, LOW);
28     delay(DELAY_TIME);
29
30     Serial.println("Turning RELAY 2 OFF");
31     digitalWrite(RELAY_PIN_2, HIGH);
32     delay(DELAY_TIME);
33 }
```

תיעוד של אופן פעולה החישון LM75 תאריך : 10.10.25



ניתן לראות את אופן החיבור של החישון טמפרטורה LM75 הפעול בפרוטוקול תקשורת I2C. מה שלמעשה שונה בחישון זה האופן חיבור המיעוד שלו למיקרו בקר ESP32, יש שתי רגליים אליו חיבורתי את הפינים שאחרים על התקשרות של החישון עם המיקרו בקר. הרגליים על המיקרו בקר הם 22,21 כאשר פין 21 מס' מתחבר לפין SCL בחישון הטמפרטורה והפין 22 מתחבר למספר 22 לפין SDA שבחישון. על מנת שאני אוכל לתקשר בפרוטוקול תקשורת I2C אני רושם Wire.begin לאחר מכן אני רשאי למשה לתקשר בפרוטוקול תקשורת זה ולהגידו אפיו פנויים אחרים ולא דוחק את הפינים 22 ו 21. אם ארצה לשנות רגליים נראה זאת בעורת הפקודה .Wire.begin(32, 33); // SDA=32, SCL=33

```

1 #include <Wire.h>
2
3 #define LM75_ADDR 0x48 // של הזיהישן I2C כתובות //
4
5 void setup() {
6     Wire.begin();
7     Serial.begin(9600);
8     Serial.println("...בזמן אמת LM75 קריית טמפרטורה ...");
9 }
10
11 void loop() {
12     Wire.beginTransmission(LM75_ADDR);
13     if (Wire.endTransmission() == 0) { // אם הזיהישן מגיב
14         Wire.requestFrom(LM75_ADDR, 2); // מבקש 2 בתים
15         if (Wire.available() == 2) {
16             byte msb = Wire.read();
17             byte lsb = Wire.read();
18
19             int temp = ((msb << 8) | lsb) >> 5; // המרת בתים
20             float celsius = temp * 0.125;
21
22             Serial.print("טמפרטורה: ");
23             Serial.print(celsius);
24             Serial.println(" °C");
25         }
26     } else {
27         Serial.println("LM75 נמצא!");
28     }
29
30     delay(200); // מseg
31 }
```

ניראה את התוצאות המודפסות ב serial monitor

```
15:05:01.370 -> LM75 לא נמצא.  
15:05:03.381 -> LM75 לא נמצא.  
15:05:05.370 -> LM75 נמצא!  
15:05:05.370 -> 29.75 °C טמפרטורה:  
15:05:07.364 -> LM75 נמצא!  
15:05:07.364 -> 29.00 °C טמפרטורה:  
15:05:09.363 -> LM75 נמצא!  
15:05:09.363 -> 28.50 °C טמפרטורה:  
15:05:11.358 -> LM75 נמצא!  
15:05:11.358 -> 28.25 °C טמפרטורה:  
15:05:13.364 -> LM75 נמצא!  
15:05:13.364 -> 28.00 °C טמפרטורה:  
15:05:15.346 -> LM75 נמצא!  
15:05:15.346 -> 27.75 °C טמפרטורה:  
15:05:17.368 -> LM75 נמצא!  
15:05:17.368 -> 27.75 °C טמפרטורה:  
15:05:19.346 -> LM75 נמצא!  
15:05:19.346 -> 27.50 °C טמפרטורה:  
15:05:21.346 -> LM75 נמצא!  
15:05:21.391 -> 27.75 °C טמפרטורה:  
15:05:23.368 -> LM75 נמצא!  
15:05:23.368 -> 28.00 °C טמפרטורה:  
15:05:25.376 -> LM75 נמצא!  
15:05:25.376 -> 28.12 °C טמפרטורה:  
15:05:27.346 -> LM75 נמצא!  
15:05:27.346 -> 28.25 °C טמפרטורה:
```

```
! נמצא!  
°C טמפרטורה: 23.00  
לא נמצא.  
לא נמצא.  
לא נמצא.  
! נמצא!  
! נמצא!  
°C טמפרטורה: 23.25  
! נמצא!  
°C טמפרטורה: 23.12  
! נמצא!  
°C טמפרטורה: 23.00  
! נמצא!  
°C טמפרטורה: 23.12  
! נמצא!  
°C טמפרטורה: 23.00  
! נמצא!  
°C טמפרטורה: 23.00  
! נמצא!  
°C טמפרטורה: 22.87
```

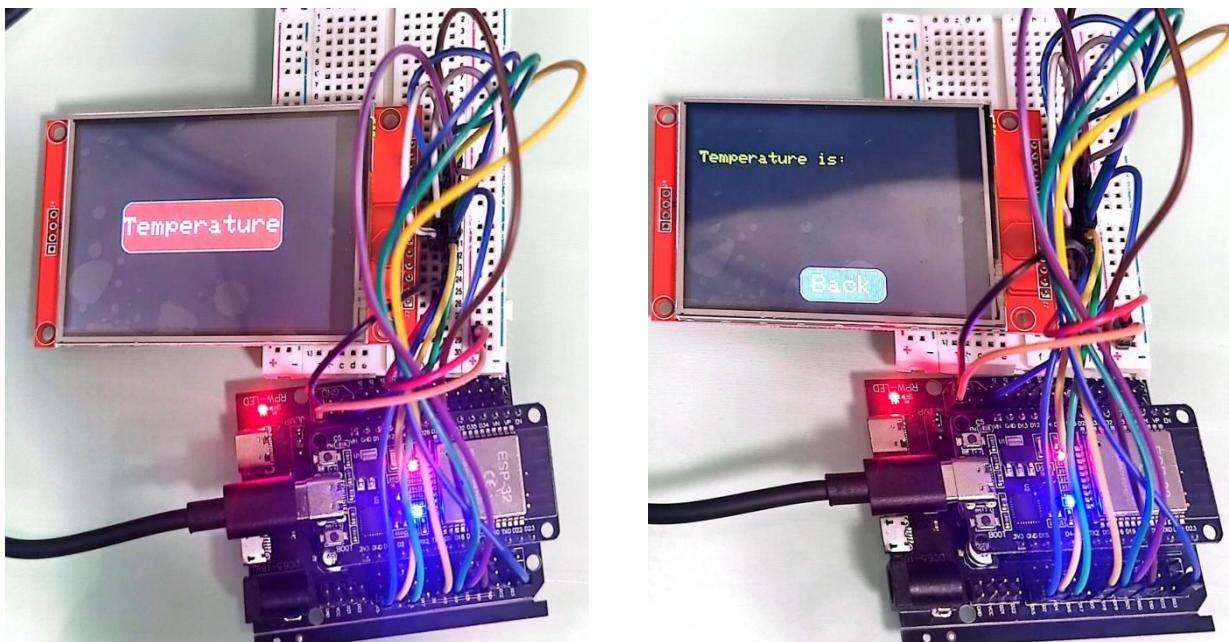
ניתן לראות את ההדפסות המתקבלות ב- serial monitor. "המצא" או "לא נמצא" זה כאשר אני מכסה את החישון וברגע שמכסים אז ניתן לראות שזה לא מצוי את החישון וברגע שלא מסתירים אז החישון נמצא. בנוסף ברגע שנגעתי בחישון אז הטמפרטורה שהוא בדק עלתה בעקבות החום גופי שלי ואני ניתן לראות את זה ממש בהדפסות למסך.

[קישור לסרטון אופן פעולה של הרכיב](https://youtube.com/shorts/kOMxanI2hQE):

תיעוד עבור טאץ במסך TFT 9431 : 17.10.25

כעת אבנה את המערכת השנייה בפרויקט שלי. הפרויקט הוא פרויקט גדול וCompanyId העומס שיש על כל רכיב/ מסך בקורס עלול לגרום להפרשי זמנים עצומים בין פעולתם. לכן החלטתי להפריד את הפרויקט לשתי מערכות נפרדות, כאשר לכל מערכת יש מיקרו בקר משלה והמיקרו בקרים יתקשרו ביניהם באמצעות הпрוטוקול WiFi_NOW.

כעת נראה את אופן הפעולה של המסך טאץ כיצד יצרתית את הטאץ ומיצאיו למעשה את הקואורדינטות עבור כל כפטור בצורה ידנית. לחצתי בעוזרת קוד פשוט לחישוב קואורדינטות ובדיקת טאץ באופן כללי, רשמתי את הקואורדינטות בקורס חדש שמשלב יצירה תצוגה לכפתורים ואדגים בעוזרת תמונות וסרטון בעמוד יווטיוב שלי שיוכנס גם לאתר שלי שבניתי עבור הפרויקט ונitin יהיה ממש לראות את האופן פועלה של המסך טאץ בצורה היקפית וברורה יותר מאשר בתמונות.



בתמונות אלו ניתן לראות את האופן בו אחיברתי את המסך. לאחר והמסך מוגדר כברירת מחדל להדפסה בצורה אנכית אז על מנת שאראה הכל בצורה אופקית נדרש לשובב את התצוגה בעוזרת 90 routation מעלות שזה למעשה סיבוב פעמי, כל 90 מעלות זה סיבוב 1 נוספת.

ESP32	מסך TFT
3.3V	VCC
GND	GND
GPIO 15	CS
GPIO 2	DC
GPIO 4	RST
GPIO 23	MOSI (DIN / SDA)
GPIO 19	MISO
GPIO 18	SCK / CLK
3.3V	LED / BL
ESP32	Touch
GPIO 5	T_CS
GPIO 21	T_IRQ
GPIO 23	D_IN
GPIO 19	D_OUT
GPIO 18	SCK

נראה את הקוד:

```

1 #include <TFT_eSPI.h>
2 #include <SPI.h>
3 #define ILI9341_DRIVER
4 #define TFT_MOSI 23
5 #define TFT_MISO 19
6 #define TFT_SCLK 18
7 #define TFT_CS 15
8 #define TFT_DC 2
9 #define TFT_RST 4
10 #define TOUCH_CS 5
11 #define TOUCH_IRQ 21
12 TFT_eSPI tft = TFT_eSPI();
13 struct Button {
14     int x, y, w, h;
15     String label;
16     uint16_t fillColor;
17     uint16_t textColor;
18 };
19 // Temperature Button
20 Button btnTemp = {0,0,200,60,"Temperature",TFT_RED,TFT_WHITE};
21 // Back Button
22 Button btnBack = {tft.width()-110, 193, 100, 40, "Back", TFT_DARKGREY, TFT_WHITE};
23 enum Screen { MAIN, TEMP };
24 Screen currentScreen = MAIN;
25 // Initialize Buttons
26 void drawButton(const Button &btn, bool pressed=false){
27     uint16_t color = pressed ? TFT_ORANGE : btn.fillColor;
28     tft.fillRoundRect(btn.x, btn.y, btn.w, btn.h, 12, color);
29     tft.drawRoundRect(btn.x, btn.y, btn.w, btn.h, 12, TFT_WHITE);
30     tft.setTextColor(btn.textColor);
31     tft.setTextSize(3);
32     int txtW = btn.label.length() * 6 * 3; // רוחב הטקסט
33     int txtH = 8 * 3; // גובה הטקסט
34     int cx = btn.x + (btn.w - txtW)/2;
35     int cy = btn.y + (btn.h - txtH)/2;
36     tft.setCursor(cx, cy);
37     tft.print(btn.label);
38 }
39 // Initialize Screen
40 void drawText(String text, int x, int y, uint16_t color, int size){
41     tft.setTextColor(color);
42     tft.setTextSize(size);
43     tft.setCursor(x, y);
44     tft.print(text);
45 }
46 // Draw Main Screen
47 void drawMainScreen(){
48     tft.fillScreen(TFT_BLACK);
49     // Temperature Button
50     btnTemp.x = (tft.width() - btnTemp.w)/2;
51     btnTemp.y = (tft.height() - btnTemp.h)/2;
52     drawButton(btnTemp);
53 }
54 // Draw Temp Screen
55 void drawTempScreen(){
56     tft.fillScreen(TFT_BLACK);
57     drawText("Temperature is:", 10, 50, TFT_YELLOW, 2);
58     drawButton(btnBack); // Back button, same size as the temperature button
59 }
60 void setup(){
61     Serial.begin(115200);
62     tft.init();
63     tft.setRotation(1); // 90° ימינה
64     pinMode(TOUCH_IRQ, INPUT);
65     drawMainScreen();
66 }
67 void loop(){
68     uint16_t tx=0, ty=0;
69
70 if(digitalRead(TOUCH_IRQ) == LOW){
71     if(tft.getTouch(&tx,&ty,600)){
72         Serial.print("Touch X: "); Serial.print(tx);
73         Serial.print(" | Y: "); Serial.println(ty);
74         // MAIN screen - touch here to change Temperature
75         if(currentScreen == MAIN){
76             if(tx >= btnTemp.x && tx <= btnTemp.x + btnTemp.w &&
77                 ty >= btnTemp.y && ty <= btnTemp.y + btnTemp.h){
78                 drawButton(btnTemp, true); // Update button
79                 delay(100);
80                 currentScreen = TEMP;
81                 drawTempScreen();
82                 while(tft.getTouch(&tx,&ty,600)); // Wait for release
83             }
84         }
85         // TEMP screen - touch here to go back
86         else if(currentScreen == TEMP){
87             if(tx >= btnBack.x && tx <= btnBack.x + btnBack.w &&
88                 ty >= btnBack.y && ty <= btnBack.y + btnBack.h){
89                 drawButton(btnBack, true); // Update button
90                 delay(100);
91                 currentScreen = MAIN;
92                 drawMainScreen();
93                 while(tft.getTouch(&tx,&ty,600)); // Wait for release
94             }
95         }
96     }
97 }
98 }
99 }
100 }

```

הקוד למעשה יוצר לנו שני כפתורים כאשר הכפתור הראשון Temperature הוא כפתור אדום וברגע שאלחץ עליו עברו לחילונית חדשה במסך שתכיל את הטקסט Temperature is: וגם כפתור Back שיחזיר אותנו לישירות לחילונית עם הפתור טמפרטורה. הלוח שאני בוחר בהרצאת התוכניות זה

ESP32 DEV Module

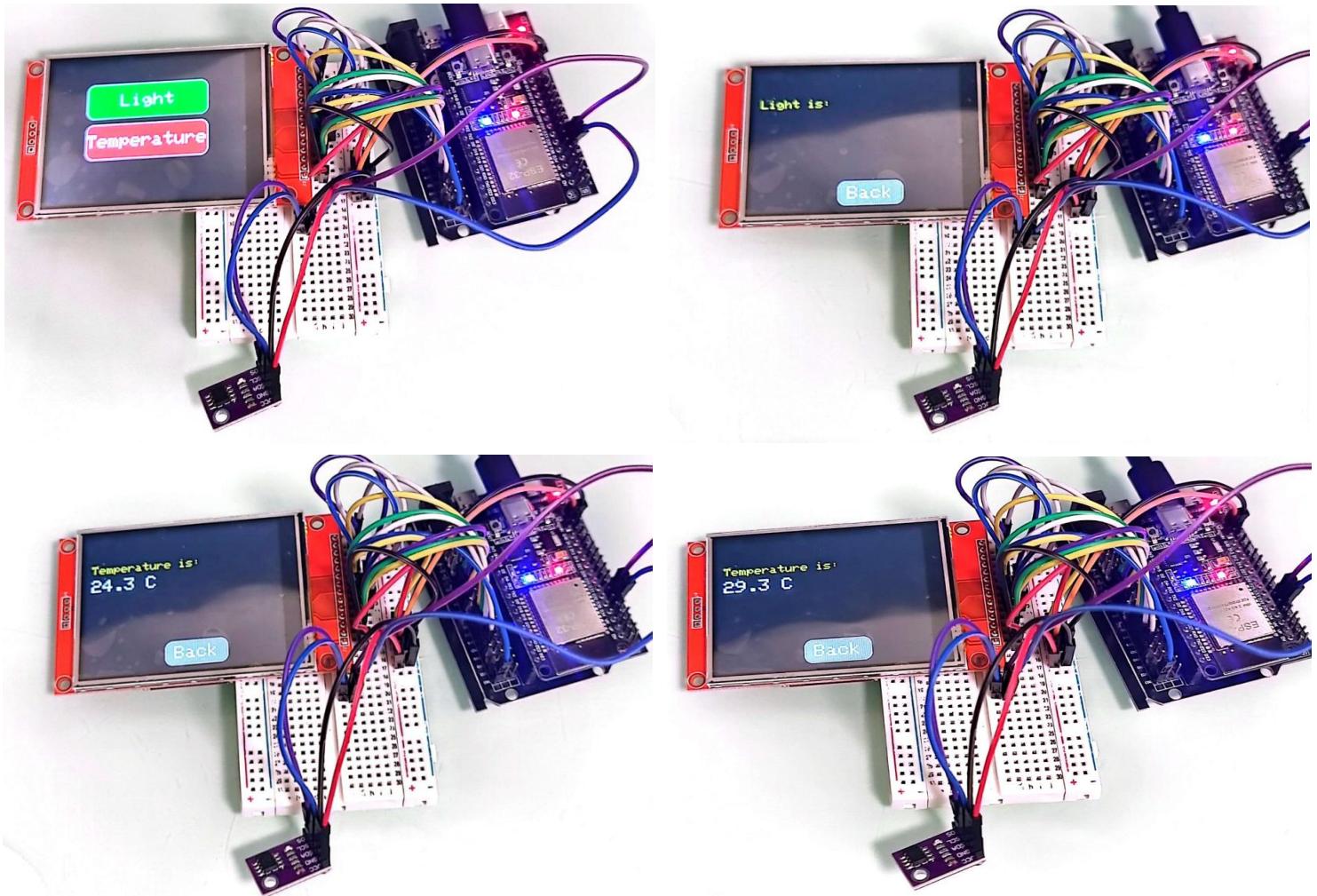
ניתן לראותו בודאי מציג את זה באופן ברור ולהבחן באופן הפעולה של המסך יחד עם הטאץ. את הסרטון אוסף גם ישירות לאתר שלי שעובד אחר הפוך.

[קישור לסרטון:](https://youtube.com/shorts/wPKWHskcr4U)

חיבור חיישן טמפרטורה למערכת השנייה והוספת כפטור עבור החישון תוארה, 25.10.25 :

כאנ אציג את ההתקדמות שעשיתני, הוסף כפטור נוסף שיציג את מה שהחישון תוארה יציג את ערכיו, כתע לא חיבורתי אותו עדיין لكن בניסוי זה רק נראה את הכפטור ומה שקרה לאחר שלוחצים עליו. הבדיקה העיקרית בניסוי זה החיבור של החישון טמפרטורה LM75 אל המערכת יכולה. אצרף למעשה תמונה המשלבota את החיבור של החישון טמפרטורה ואציג שני סרטונים בפנרד אחד עבור עיצוב המשך יחד עם הכפטור השני, וסרטון נוסף עבור חיבור של החישון עם המערכת השנייה.

נראה את החיבורים:



חברתי את החישון טמפרטורה את המתח ל-3.3 וולט את האדמה לאדמה ואת SDA לפין 32 ואת SCL לפין 33. ניתן לראות את התצוגה החדשה על המשך שלי יחד עם הכפטור הירוק החדש שרשום עליו תוארה באנגלית וכאשר לחוצים עליו אז ניתן לראות בתמונה מעלה מכך ימין שזה מעביר אותו לחילונית כזאת ומשם ניתן לחזור ולראות את הערכיהם רק שמכיוון שלא חיבורתי את החישון עדיין אז לא קורא שום ערכיהם וכרגע אי אפשר לראותו.

סרטון ראשון של הוספת כפטור נוסף:

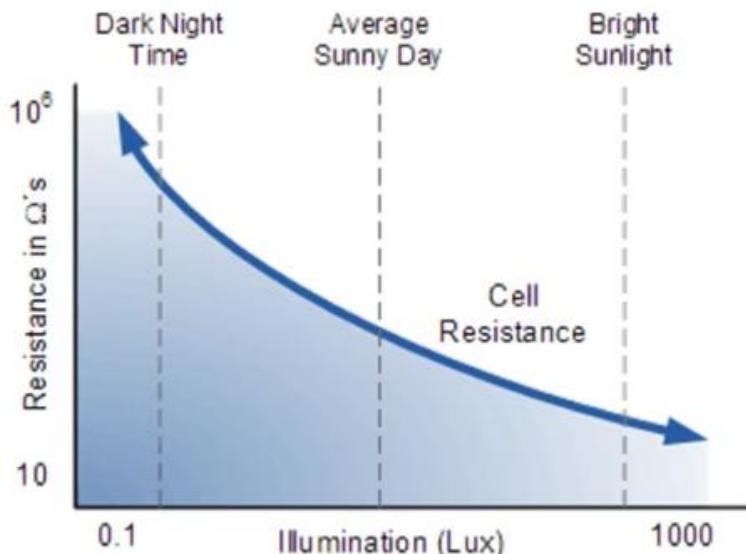
<https://youtube.com/shorts/AhsRNy-44W4?feature=share>

סרטון שני של הוספת חיישן טמפרטורה : LM75
<https://youtube.com/shorts/1TjnN75Hobc?feature=share>

תיעוד של אופן פעולה חיישן LDR עם המודול חיישן האור 018-KY:

בניסוי זה בדקתי למשה את אופן פעולה החישן LDR בתוך המודול שאני משתמש בו עבור פרויקט זה. למעשה אני זוקק לבדוק תאוריה מסוימת שאמם מדובר בתנאי חושך עלי לפעיל משווה שידמה מקור אור על מנת להגדיל את שדה הראיה ואת השטח מסביב למערכת.

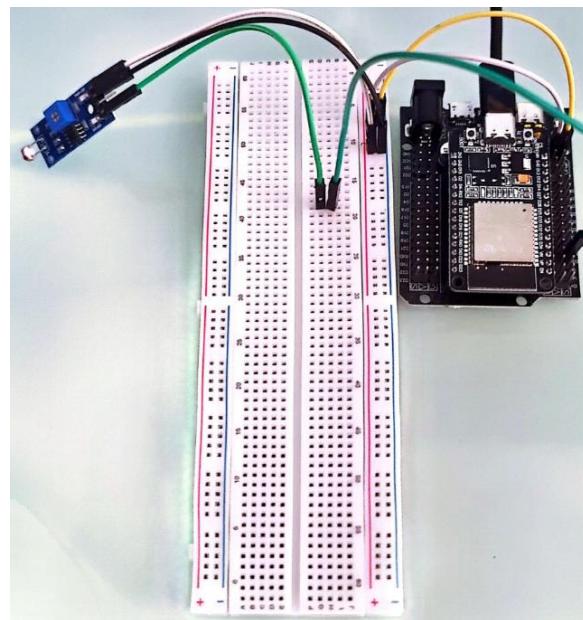
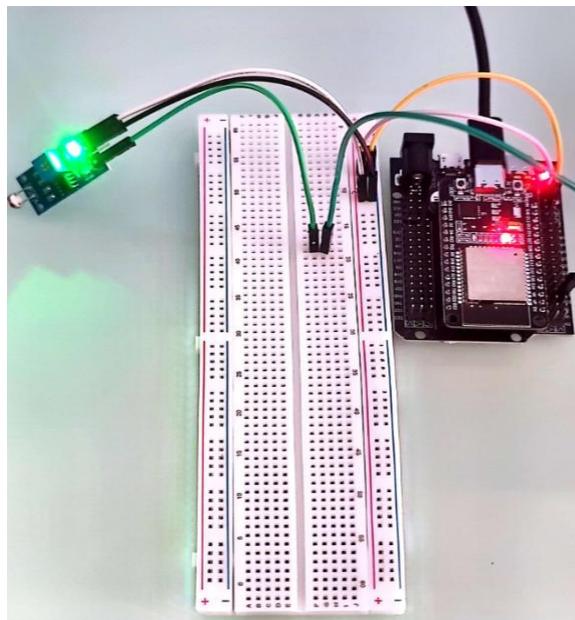
בניסוי זה ראייתי שככל שיש יותר אור בסביבת החישן LDR הערך האנלוגי צונח ומנגד כאשר אני מכסה את החישן עם האכבעות אז הערך האנלוגי גדול. אוכל לשמש בבדיקה ערכאים אלו ולהציג ערך סף שמננו אדליק או אכבה את ה"מקור אור" בפרויקט שלי.



כאן לנעש ניתן לראות את הגרפ' שהוא בוצרה הקספוננציאלית. ככל שייתר חשוך בסביבה נראה שהערכים גדלים מושום שהה Tangendot גדלה ככל שייתר חשוך בסביבה.

לעומת זאת כאשר בהיר בסביבה של החישן LDR נראה שהה Tangendot קטנה וכן גם הערכים שמציד החישן קטנים.

נראה את אופן החיבור :



החיבורים שעשייתי:

חיברתי את המתח ל-3.3 וולט ואת האדמה לאדמה את הפין A0 חיבורתי לפין 34 במקרו בקר.

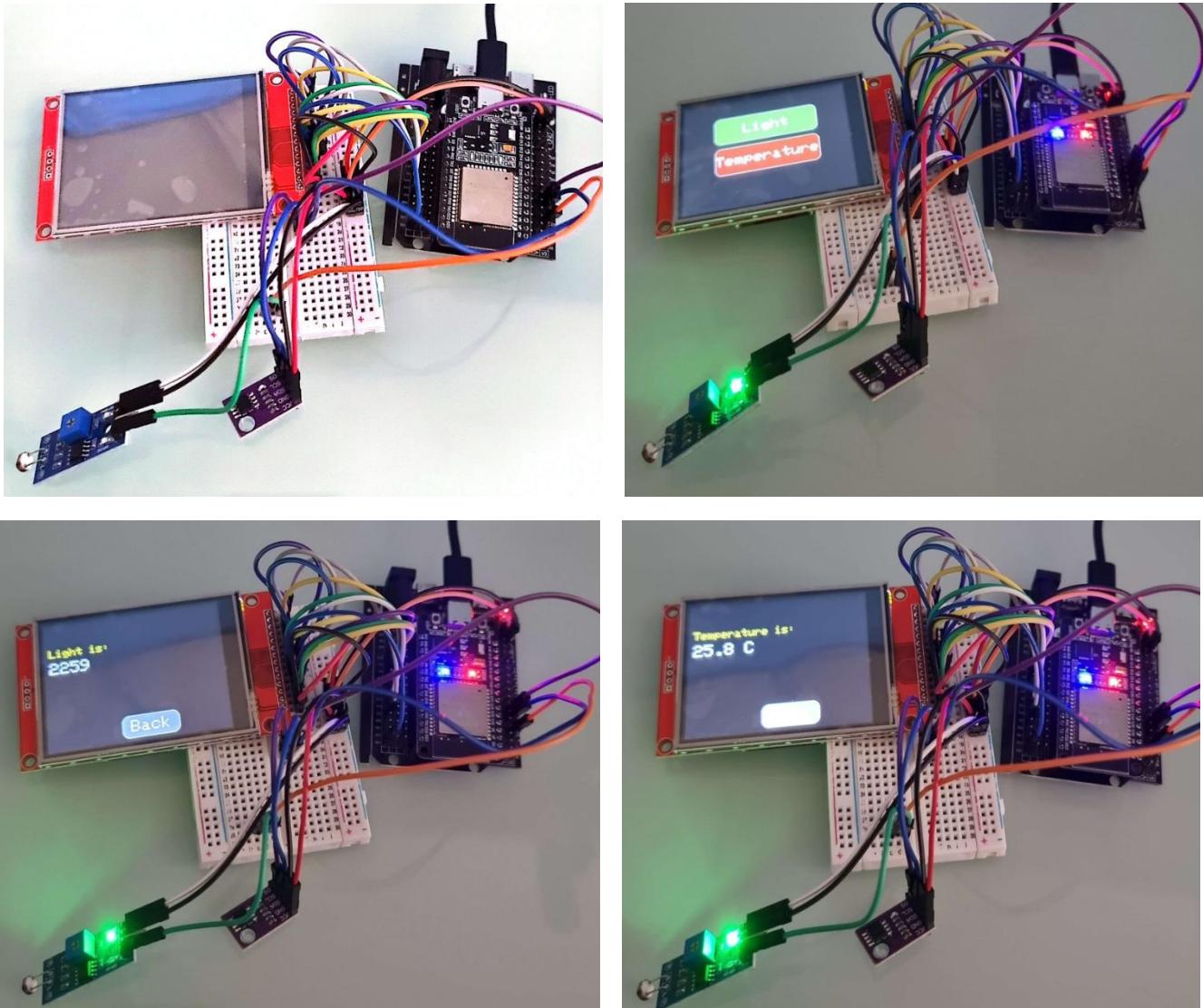
סרטון בו אני מציג את אופן הפעולה:

<https://youtube.com/shorts/BH0AUaHqhAI?feature=share>

תיעוד של חיבור החישון למערכת השניה עם המ██ ו החישון טמפרטורה, 8.11.25:

בניסוי זה למשה בדקתי האם המערכת עם הבקר השני יכולה המשך יחד עם כל החישונים. לאחר שחברתי את החישון LDR עם המודול אל המערכת יכולה שיניתי את הקוד כך שייציג לי את הערכים האנלוגיים שמנגלה החישון LDR הערכים מוצגים בחלוןנית ישר לאחר שלוחצים על הכפתור LIGHT וזה מעביר לחלונית נוספת שם יוצאה הערך והכפתור חוזר לתפריט הראשי.

נראה את החיבור של המערכת ומצבים שונים :



ניתן לראות את המ מצבים השונים המשך פתיחה ואת החלונית הנפתחות בהתאם לכל כפתור שנלחץ. את אופן הפעולה קל יותר לראות בתורה ברורה בסרטון שאצרף ממש כאן עבור בדיקה זו. וכמוון שאצרף גם לאתר שלי עם כוונת מתאימה שיהיה קל לעקוב אחרי התקדמות הפרויקט.

קישור לסרטון בערוץ יוטיוב שלי:
<https://youtube.com/shorts/ooBv1BanwWM?feature=share>

תיעוד של תקשורת בין שני המיקרו בקרים ומציאת כתובת MAC של המקלט

בניסוי זה בדקתי את הדבר החשוב ביותר התקשורת בין שני המיקרו בקרים. למעשה הפרויקט שלי הוא פרויקט גדול והחלטי חלק אותו לשתי מערכות נפרדות המתקשרות זו עם זו בקשר ESP_NOW. תקשורת זו עדיפה על פני תקשורת טורית רגילה המשמשת לא-חוותית הנקראת UART. מושם שמאפשרת לנו מרחק רחב יותר של שימוש ואך יותר מיקרו בקרים לתקשורת.

נראה את הקודים והפלטים בסריאל מוניטור שייצאו לי:

בצד השולח:

```

1 #include <WiFi.h>
2 #include <esp_now.h>
3 uint8_t receiverAddress[] = {0x6C, 0xC8, 0x40, 0x4E, 0x31, 0x78}; // כל כתובת MAC צריכה להיות
4 typedef struct {
5     uint8_t value;
6 } Message;
7 void setup() {
8     Serial.begin(115200);
9     delay(1000);
10    Serial.println("Sender setup started...");
11    WiFi.mode(WIFI_STA);
12    if (esp_now_init() != ESP_OK) {
13        Serial.println("Error initializing ESP-NOW");
14        while(true);
15    } else {
16        Serial.println("ESP-NOW initialized");
17    }
18    esp_now_peer_info_t peerInfo = {};
19    memcpy(peerInfo.peer_addr, receiverAddress, 6);
20    peerInfo.channel = 0;
21    peerInfo.encrypt = false;
22    if (esp_now_add_peer(&peerInfo) != ESP_OK) {
23        Serial.println("Failed to add peer");
24    } else {
25        Serial.println("Peer added successfully");
26    }
27 }
28 void loop() {
29     Message msg;
30     msg.value = random(0, 256);
31     esp_err_t result = esp_now_send(receiverAddress, (uint8_t *)&msg, sizeof(msg));
32     if (result == ESP_OK) {
33         Serial.print("Sent value: ");
34         Serial.println(msg.value);
35     } else {
36         Serial.println("Send failed");
37     }
38     delay(1000); // שולח פוד בשנייה
39 }
40 
```

פלט בסריאל מוניטור:

```

19:33:05.014 -> Sent value: 140
19:33:06.004 -> Sent value: 39
19:33:07.014 -> Sent value: 208
19:33:08.031 -> Sent value: 201
19:33:09.007 -> Sent value: 244
19:33:10.015 -> Sent value: 225
19:33:11.011 -> Sent value: 88
19:33:12.020 -> Sent value: 177

```

בצד המקלט:

```

1 #include <WiFi.h>
2 #include <esp_now.h>
3 typedef struct {
4     uint8_t value;
5 } Message;
6 // Callback הודהה
7 void onDataRecv(const esp_now_recv_info_t *info, const uint8_t *incomingData, int len) {
8     Message msg;
9     memcpy(&msg, incomingData, sizeof(msg));
10    Serial.print("Received value: ");
11    Serial.println(msg.value);
12 }
13 void setup() {
14     Serial.begin(115200);
15     delay(1000);
16     Serial.println("Receiver setup started...");
17     WiFi.mode(WIFI_STA);
18     if (esp_now_init() != ESP_OK) {
19         Serial.println("Error initializing ESP-NOW");
20         while(true);
21     } else {
22         Serial.println("ESP-NOW initialized");
23     }
24     esp_now_register_recv_cb(onDataRecv);
25 }
26 void loop() {
27     // כלום, ההודאות מתקבלות ב callback
28 }

```

פלט בסריאל מוניטור:

```

19:33:05.012 -> Received value: 140
19:33:06.002 -> Received value: 39
19:33:07.012 -> Received value: 208
19:33:08.029 -> Received value: 201
19:33:09.005 -> Received value: 244
19:33:10.013 -> Received value: 225
19:33:11.010 -> Received value: 88
19:33:12.018 -> Received value: 177

```

ניתן לראות שהערכים הנשלחים מהצד של הבקר השולח אלו בדיקת הערכים שאנו מקבלים בצדו של התקיבר המקלט.

ESP-NOW הוא פרוטוקול תקשורת אלחוטי שנבנה בתוך שבבי ESP8266 ו-ESP32 ומאפשר למיקרו-בקרים לתקשר זה עם זה ישירות באמצעות Wi-Fi, ללא צורך בנטב, רשת אלחוטית או חיבור לאינטרנט. ברגע של Wi-Fi וגל שבו יוצרים רשת, מתחברים ל-SSID ומעבירים נתונים בפרוטוקולים כמו TCP/IP, UDP, ESP-NOW המשיע נשלח בצורה של חבילות קצורות (packets) ישירות בין מיקרו-בקרים על בסיס כתובות החומרה שלהם. היתרון הגדול של ESP-NOW הוא מהירות גבוהה, השהייה נמוכה מאוד, צרכית חשמל נמוכה, ופשטות – מה שהופך אותו לאידיאלי לפROYקטים של חישנים, רובוטיקה, רדיארים ומערכות בזמן אמת כמו בפרויקט שלך.

במערכת של קיימים שני תפקדים ברורים: משדר (Sender) ומקבל (Receiver). המשדר מודד ערך כלשהו (ערך שמייצג מצב או מרחק), אוزو אותו בתוך מבנה נתונים (struct) ושולח אותו למקבל באמצעות WiFi. המקלט אינו "מבקש" את המידע אלא מקבל אותו אוטומטית

ברגע שהוא נשלח, באמצעות מנגנון שנקרא `callback`. זהו רעיון חשוב: הפונקציה `onDataRecv` אינה רצתה בוללה (loop), אלא מופעלת אוטומטית על ידי המערכת בכל פעם שהביילט נתונים הגיעו מהאוויר.

בקוד של המקלט רואים שהוא מוגדר מבנה בשם `Message`, שבתוכו שדה אחד מסוג `uint8_t bytes`. זהו מבנה הנתונים שמייצג את הודעה שנשלחת. כאשר מתאפשרת הודעה, המידע הגולמי (bytes) מועתק לתוך המבנה הזה באמצעות `memcpy` וכן אפשר לעובוד עם הנתונים בצורה מסודרת וברורה. לאחר מכן הקוד מדפיס לסדריאל את הערך שהתקבל, דבר שמאפשר בדיקה וניטור של התקשרות בזמן אמת. חשוב להבין שב-`ESP-NOW` אין "חיבור מתמשך" כמו ב-`Bluetooth`, אלא כל הודעה היא חביבה עצמאית שמניעה, מעובדת, ונגמרה.

בצד המשדר מתבצע תהליך הפוך. גם כאן מוגדר אותו מבנה `Message`, כדי שני הצדדים "ידברו באותה שפה". המשדר מתחילה את `ESP-NOW`, ואז מוסיף מקבל (`peer`) לרשימת היעדים שלהם מותר לשולח נתונים. ההוספה זו מתבצעת באמצעות כתובות ה-`MAC` של המקלט. לאחר שה-`peer` נוסף בהצלחה, המשדר יכול לשולח הודעות באמצעות הפונקציית `esp_send_now`. בדוגמה הזאת, ככל שניתנה נשלח ערך אקראי, והמשדר מדפיס לסדריאל האם השילוח הצליפה. זה שלב חשוב מאוד בפיתוח, כי כך ניתן לדעת שהבעיה אינה בתקשורת אלא בלוגיקה של התוכנית.

כתובות ה-`MAC` היא מזזה חומרה ייחודי של כל רכיב, Wi-Fi בדומה למספר שלדיה של רכב. כל `ESP32` מגע מהפעל עם כתובות `MAC` ייחודית, והוא משתמש את `ESP-NOW` כדי לדעת למי לשולח את הנתונים. פרוייקט שלנו מצאנו את כתובות ה-`MAC` של המקלט על ידי הדפסתה בסדריאל מוניטור באמצעות פונקציה כמו `(WiFi.macAddress())` או על ידי קריית הפלט שמופיע בעת צירבת הקוד (כפי שניתן לראות בלוג של `esptool`). לאחר שמצאנו את הכתובות, העתקנו אותה לצורה של מערך בתים (hexadecimal) אל קוד המשדר, למשל

{`0x78, 0x31, 0x40, 0xC8, 0x6C, 0x05`}

זהו שלב קרייטי: אם כתובות ה-`MAC` אינה נכונה, הנתונים פשוט לא יגיעו, גם אם הקוד תקין לחלוון.

לxicom `ESP-NOW`, אפשר לנו לבנות מערכת מבוססת מיקרו-בקר אחד מודד ומעבד נתוניים, ומיקרו-בקר אחר מציג או מגיב אליהם, לצורה מהירה ואמינה ולא תלות בתשתיות חיצונית. השימוש ב-`struct`, `callback`, לקבלת נתונים, וכתוות `MAC` לויה הייד, יוצר מערכת תקשורת נקייה, מודולרית ומקצועית.

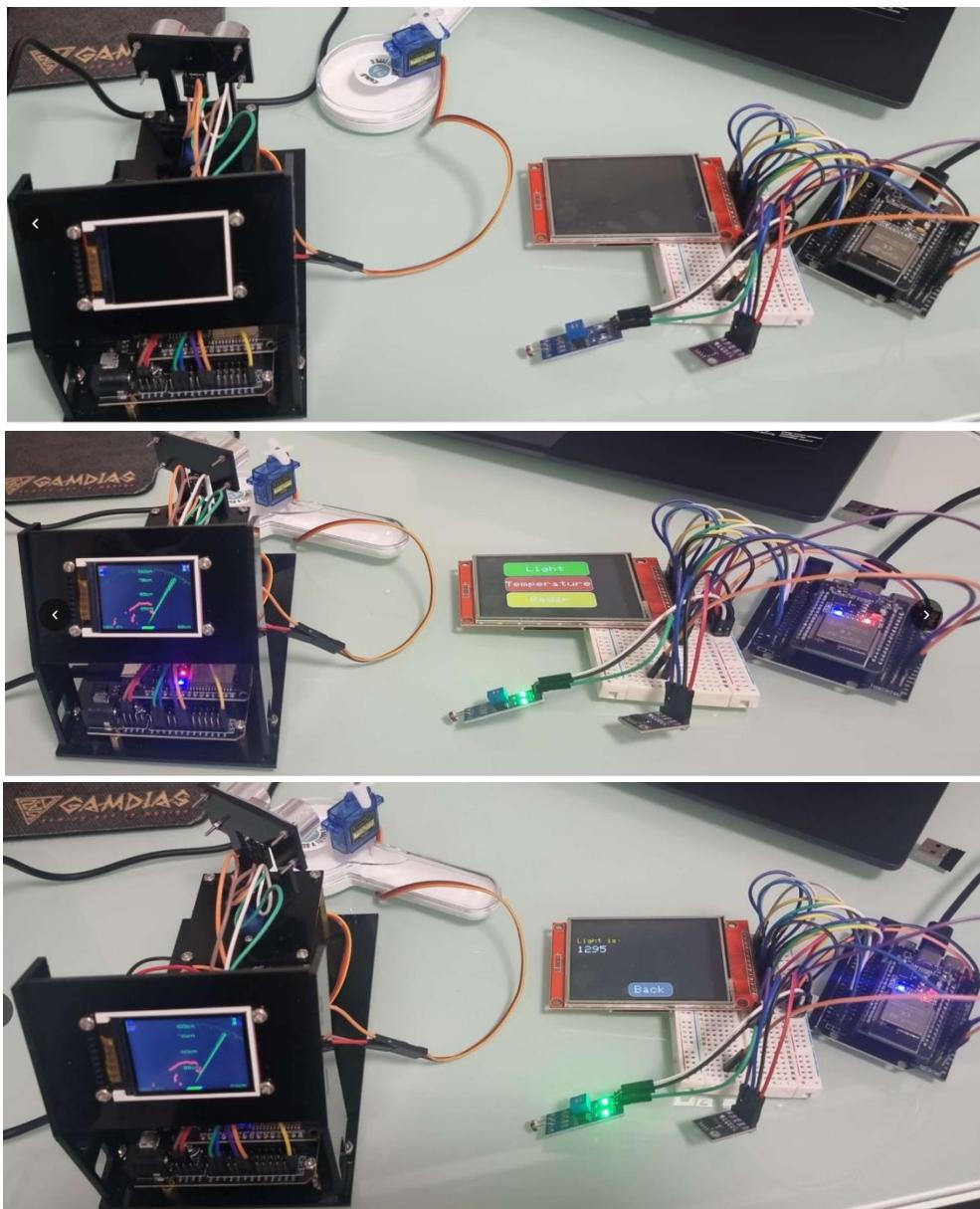
[קישור לסרטון של אופן תקשורת בין שני הבקרים:](#)

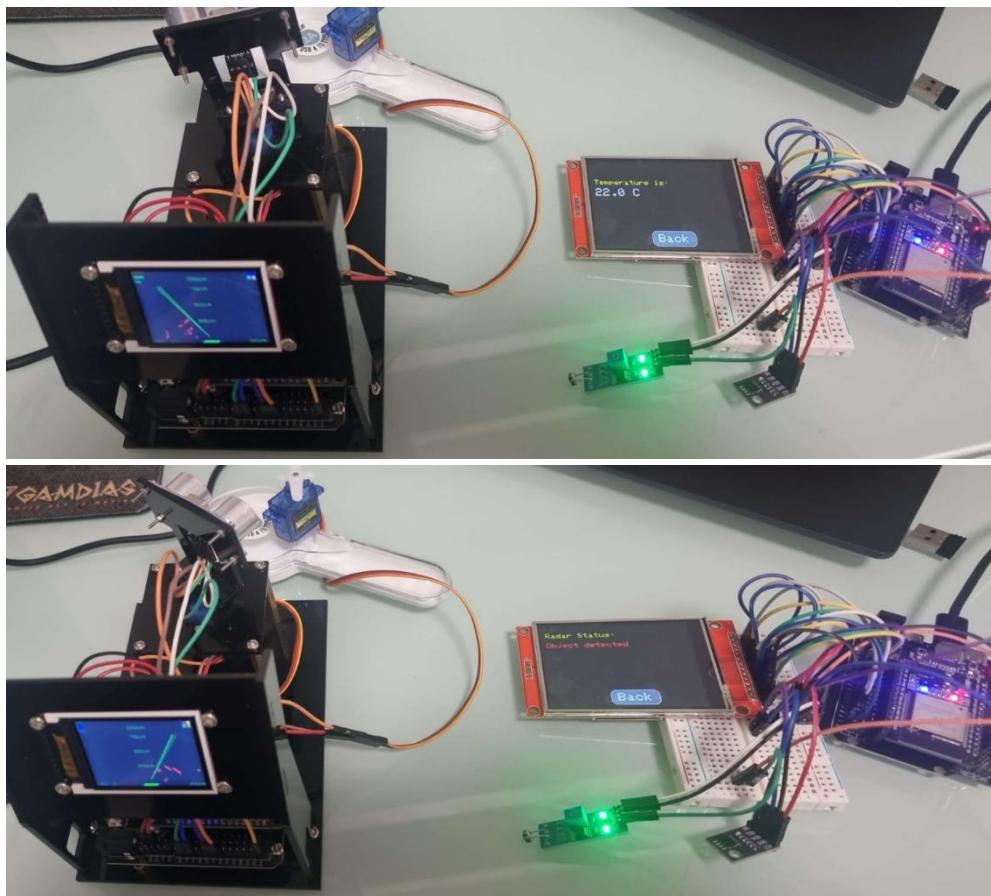
https://youtu.be/oa2_R1PLNHI

תיעוד של אופן התקשרות בין שתי המערכות, 22.11.25:

בניסוי זה החלטתי לבדוק את האופן בו המערכות פועלות כל אחת בפני עצמה ובנוסף לכך את התקשרות בין שני המיקרו בקרים. כיצד עשית זאת? למעשה חיבורתי כל מערכת לפורט אחר במחשב דרך חיבור USB לאחר מכן בחרתי את ה- COM המתאים עבור כל מערכת והרצתי אותה בתוכניות. מה שימושה עשית זו הוספה במערכת של המערכת עם המשך תצוגה הגדול והחישונים, הוספה כפתרון נוספת לצבע צחוב ובו רשות RADAR לאחר מכן כאשר לוחצים עליו זה מעביר אותנו לחולנית נפרדת שבה רואים האם התגללה אובייקט במידה והמערכת הראשונה עם הרadar גילתה אובייקט והנוקודה היא אדומה אז במערכת השנייה רשום לנו שהתגללה אובייקט לצבע אדום. במידה ולא התגללה אובייקט שזה 1000 סנטימטר ומעלה ויש נוקודה צהובה במערכת הראשונה של הרadar אז המערכת השנייה מראה שאין אובייקט. נראה איך הכל עובד ואיך סרטון שמרת החול בזורה ברורה ומקצועית יותר. כמובן שככל הסרטונים עולים לאתר שלי שמהווה חלק בלתי נפרד לתצפית על התקדמות הפרויקט שלי.

נראות חיבורים:





ניתן לראות את אופן הפעולה של התצוגה ואת המכבים השונים לאחר שלוחצים על כל כפתור. מאחר ואלו תמונות קצט קשה להבין כיצד הדברים פועלים בפועל لكن אצרף סרטון להמחשה ברורה שיציג את אופן הפעולה של המערכת והפרויקט. בנוסף ניתן לראות את המכבים השונים של הרadiator שכאשר יש נקודות אדומות אז שום שהתגלה אובייקט וזה רשום בצד אדום וכאשר יש נקודות צהובות רשום שאין אובייקט בטווח של 100 סנטימטר.

נבחן גם בסריאל מוניטור של כל אחד מהמיקרו בקרים. צירפתי גם הדרשות בסריאל מוניטור על מנת להבטיח תקשורת בטוחה ולראות שכאן מתקיימת תקשורת. למעשה השולח שזאת המערכת של הרaadär מעבירה את המרחק שהראaadär מדפיס על המסך והחישן אולטרסונייק קולט. לאחר מכן אנו נראה בהדפסה הסריאלית של המיקרו בקר מקבל את אותם הערכים שנשלחו אליו.

המקבל :	השלוח :
22:09:34.887 -> Object detected: 35	22:09:34.887 -> Object detected: 35
22:09:34.922 -> Object detected: 36	22:09:34.922 -> Object detected: 36
22:09:34.988 -> Object detected: 34	22:09:34.988 -> Object detected: 34
22:09:35.116 -> Object detected: 16	22:09:35.115 -> Object detected: 16
22:09:35.162 -> Object detected: 25	22:09:35.161 -> Object detected: 25
22:09:35.196 -> Object detected: 30	22:09:35.196 -> Object detected: 30
22:09:35.278 -> Object detected: 41	22:09:35.278 -> Object detected: 41
22:09:35.359 -> Object detected: 39	22:09:35.358 -> Object detected: 39
22:09:35.445 -> Object detected: 38	22:09:35.445 -> Object detected: 38
22:09:35.492 -> Object detected: 36	22:09:35.491 -> Object detected: 36
22:09:35.654 -> Object detected: 37	22:09:35.653 -> Object detected: 37
22:09:35.740 -> Object detected: 36	22:09:35.740 -> Object detected: 36
22:09:35.858 -> Object detected: 37	22:09:35.857 -> Object detected: 37
22:09:35.903 -> Object detected: 36	22:09:35.903 -> Object detected: 36

ניתן להבחין שישנם הפרשים קטנים בזמני שליחה וקבלת וכך אנו למשה מבחנים מי מהם מי. השולח שלח את ההודעה והמקבל יכול לקבל את ההודעה בעיקוב ממש קטן ניתן להבחין בזאת בכמה מקומות כמו למשל 22:09:35.857 בצדו של השולח, והמקבל מקבל את זה ב 22:09:35.858.

קישור לסרטון:

<https://youtube.com/shorts/cFMXjWujSgI?feature=share>

ביבליוגרפיה:

במהלך הפרויקט שלי נזורתי במספר מקומות באינטרנט כדי לרכז ולסכם את המידע בצורה הכי פרקטית ומקוצרת שאפשר. נזורתי במספר אתרים.

האתרים שיצא לי ללמוד מהם ולרכז מידע מדויק שעזר ליקדם את הספר פרויקט היו אתרים שנזורתי בהם בכלל תקופת הלימודים שלי במהלך התואר הטכנולוגי (יג-יד) שעשיתי.

באטרים אלו ישנו סיכומים רבים ו מבחני תרגול שהייתי מתכוון דרכם ומתפתח באופן כללי כדי לחזק ולשפר את הידע שלי לגבי חישנים למיניהם או פרוטוקולים שונים.

לדעתי גם במהלך התואר ובאופן כללי בחיים יש צורך להתפתח מחו למסגרת הלימודים הקבועה לנו וללקחת קורסים עצמאיים כדי לקבל ידע ממקורות נוספים.

האתרים העיקריים בהם הם:

- האתר לאלקטרוניקה ומחשבים אתר גדור עם סיכומים וחבבים בכל התחומיים. דרך האתר זה למדתי עוד בלימודי במהלך התיכון והוא עזר לי המון כדי לחזור ואףקדם את הידע שלי בכל תחום עלילות האלקטרוניקה, התכונות והמחשבים, קישור לאתר :
<https://www.elec4u.co.il>
- אתר נוסף שעזר לי לגבי הפרויקט שלי גם במהלך הפרויקט בסוף לימודי בתיכון במקצוע אלקטרוניקה ומחשבים וגם בלימודי בתואר הטכנולוגי במהלך זהו האתר של פורט, יש לו חומר לימודי, סיכומים, דוגמאות ועוד הרבה דברים שימושיים שיכולים לעזור להבנת הנושאים בכל עלילות האלקטרוניקה, התכונות ומחשבים. הקישור לאתר שליהם :
<https://www.arikporat.com>
- ספר טוב מאוד שמננו לקחתי חלק מההטבות ויש בו Data-sheets I2C בנושא זה- ודברים טובים ו שימושיים להבנת הפרוטוקול תקשורת I2C. הקישור בספר זה שנזורתי בו הוא <https://www.st.com/resource/en/datasheet/vl53l8cx.pdf>
- מקום נוסף שהשתמשתי בו על מנת לסכם מידע ולקיים חלק מההטבות בנוגע למנוע SERVO זהו הספר עם הקישור הבא :
<https://www.arikporat.com/wp-content/uploads/2024/03/%D7%9E%D7%A0%D7%95%D7%A2-SERVO.pdf>
- נזורתי בדף הבא על מנת לסכם לגבי החישן אור שהוא חלק מן המודול חישן LDR שבו אני משתמש בפרויקט שלי, הקישור לאתר הוא :
<https://gabbyshimoni.wixsite.com/arduino-programming/ldr>
- כדי להעלות את האתר שלי השתמשתי בשני תוכנות הראשונות GitHub שבה עשיתי העלית את התקינה שבה נמצאים הקודם CSS+HTML בנוסח לכך שם נמצאים התמונות שקיישרתי לאתר לסרטונים הראשיים בעמוד הבית. ובנוסח הקובץ PDF של הספר שלי, על מנת שנitin יהיה לפותח אותו באתר או Netlify צריך להזות את מיקום קובץ ה-PDF ולכן הכנסתי אותו לתיקייה הראשית שמנה אני מעלה את כל המסמכים שלו ל-GitHub וב-Netlify אני למשה מקשר את התיקייה שמאוכסנת ב-GitHub ולאחר מכן יוצר את הדומין של האתר (ה קישור) שדרכו למשה אני יכול להכנס אל הדף האינטרנט שלי.
- לאתר הבא שאציג נזורתי בו כדי לסכם את החומר בנוגע למודול מסר כפול 75. קישור לאתר :
<https://www.handasiya.co.il/aboutus>
- באתר הבא נזורתי בהציגת תमונות החיבור של מיקרו בקר ESP32 יחד עם המمسטר הכפול בספר פרויקט שלי, הקישור להרבה מיקרו בקר ESP32 יחד עם הממסטר להרבה מיקרו בפרקtes ESP8266-
<https://randomnerdtutorials.com/micropython-relay-web-server-esp32-esp8266/>
- אתר נוסף שנזורתי בו על מנת לסכם מידע על פרוטוקול תקשורת NOW_ESP, הקישור לאתר הוא :
<https://randomnerdtutorials.com>

קישור לאתר שלי : <https://raybrook-radar-system.netlify.app>