

אוניברסיטת בן-גוריון בנגב  
Ben-Gurion University of the Negev



בית הספר להנדסת חשמל ומחשבים

דו"ח תיעוד פרויקט מסכם קורס "מבנה מחשבים ספרתיים"

36114191

תכנון מערכת לגילוי מקורות אור וניטור אובייקטים  
במרחב

אדר שפירא 209580208

יהונתן דדכה 211468582

01/09/2025

## תוכן עניינים

2	תוכן עניינים.....
3	מטרת הפרויקט.....
3	תיאור הפרויקט.....
5	דיאגרמת FSM.....
5	מעגל אלקטרוני של חומרת הקצה המחוברת ל – MCU.....
6	שאלות בנושא תכנון המערכת.....
6	גילוי אובייקטים במרחב.....
9	גילוי מקורות אור במרחב.....
12	מד מרחק.....
12	מערכת קבצים.....
14	כיוול חומרה ובניית דרייברים לחיישנים ולרכיבי קצה.....
14	מנוע סרבו.....
14	חיישן מרחק.....
15	חיישן מרחק ממקור אור.....
16	מבנה נתונים ואלגוריתמים.....
19	מיפוי המשתנים בזיכרון.....
19	מסקנות והצעות לשיפורים.....

## הגדרת ומטרת הפרויקט

מטרת הפרויקט היא לתכנן ולממש מערכת משובצת (Embedded System) המבוססת על בקר מיקרו־קונטרולר, אשר משלבת יכולות חישה, בקרה ותקשורת. המערכת נדרשת לאפשר גילוי אובייקטים ומקורות אור במרחב באמצעות סריקה בסרבו, מדידת מרחק בעזרת חיישן אולטראסוני, וכן ניהול מערכת קבצים פנימית ב־Flash, הכוללת שמירה, קריאה והרצה של קבצי טקסט וקבצי סקריפט. בנוסף, המערכת כוללת ממשק משתמש בצד הבקר (כפתורים + LCD) ובצד המחשב (שליחת קבצים ופקודות דרך UART).

## תיאור תמציתי של הפרויקט (Abstract)

בפרויקט זה פותחה מערכת משובצת המיישמת מודל שכבות מלא: APP, BSP, HAL, API. המערכת מחוברת למנוע סרבו, חיישן מרחק אולטראסוני, שני חיישני אור (LDR), ותצוגת LCD. המשתמש יכול להפעיל את המערכת במצב סריקה, במצב מדידה, ובמצב ניהול קבצים.

במצב File Mode ניתן לדפדף בין שמות קבצים המאוחסנים ב־Flash, להציג קובץ טקסט על גבי ה־LCD, או להריץ קובץ סקריפט – שבו פקודות מקודדות (ISA) המפעילות את הסרבו, התצוגה והעיכובים. מערכת הקבצים תומכת בקבלת קבצים מהמחשב דרך תקשורת טורית בפרוטוקול עצור וחכה עם חישוב checksum לאימות שלמות הנתונים.

המערכת מדגימה אינטגרציה מלאה של חומרה ותוכנה: ניהול משאבי MCU, טיפול באירועי משתמש, שמירת מידע ב־Flash, פרוטוקולי תקשורת, והפעלת אלגוריתמים פשוטים לחישה ובקרה. תוצרי הפרויקט כוללים מימוש קוד מלא בשפת C, דיאגרמות FSM, תרשימי בלוקים ותיעוד הנדסי מלא.

## תיאור רכיבי חומרת הקצה וחלוקת העבודה בין החומרה לתוכנה

המערכת מבוססת על מיקרו־קונטרולר MSP430G2553 ממשפחת MSP430 של Texas Instruments, אשר שולב בערכת המעבדה האישית ששימשה אותנו במהלך הסמסטר. הבקר הוא ליבת 16 סיביות RISC, בעל זיכרון פנימי, ממיר ADC, טיימרים וממשקי תקשורת מובנים. סביב הבקר חוברו מספר רכיבים פנימיים (המסופקים כחלק מהערכת הפיתוח) ורכיבי קצה חיצוניים שנוספו במימוש הפרויקט.

## רכיבים פנימיים בבקר ובערכת הפיתוח

- תצוגת LCD (16\*2 תווים) – משמשת להצגת תפריטים, הודעות מערכת ותוכן קבצים. ה־LCD מחובר לפורטים דיגיטליים של הבקר ונשלט דרך ממשק HAL תוכנתי.
- כפתורים (PB0, PB1) – מוגדרים כקלטי GPIO, מאפשרים דפדוף בין קבצים (PB0) ובחירה/כניסה (PB1).
- זיכרון Flash פנימי – משמש גם לאחסון הקוד וגם למימוש מערכת קבצים פשוטה. ספריית הקבצים נשמרת ב־Information Memory, בעוד תוכן הקבצים (טקסט/סקריפט) מאוחסן בסגמנטים ייעודיים ב־Code Flash.

## רכיבי קצה חיצוניים

- מנוע סרבו (SG90) – מחובר לפין PWM של Timer\_A. מאפשר סריקה בזוויות שונות לצורך גילוי עצמים או מקורות אור.
- חיישן מרחק אולטראסוני (HC-SR04) – מחובר ל־GPIO וטיימר Capture למדידת פולס Echo, ומאפשר מדידת מרחק בס"מ.
- חיישני אור (LDR \*2) – התנגדותם משתנה כתלות בעוצמת האור. חוברו לכניסות ADC10 של הבקר ומשמשים לזיהוי כיוון מקור אור.

- תקשורת UART – מבוצעת דרך מודול USCI\_A0 הפנימי של הבקר, לצורך קבלת קבצים מהמחשב ושליחת ACK/NACK בהתאם לפרוטוקול “עצור וחכה”.

### כיול חומרה מול ערכי יצרן

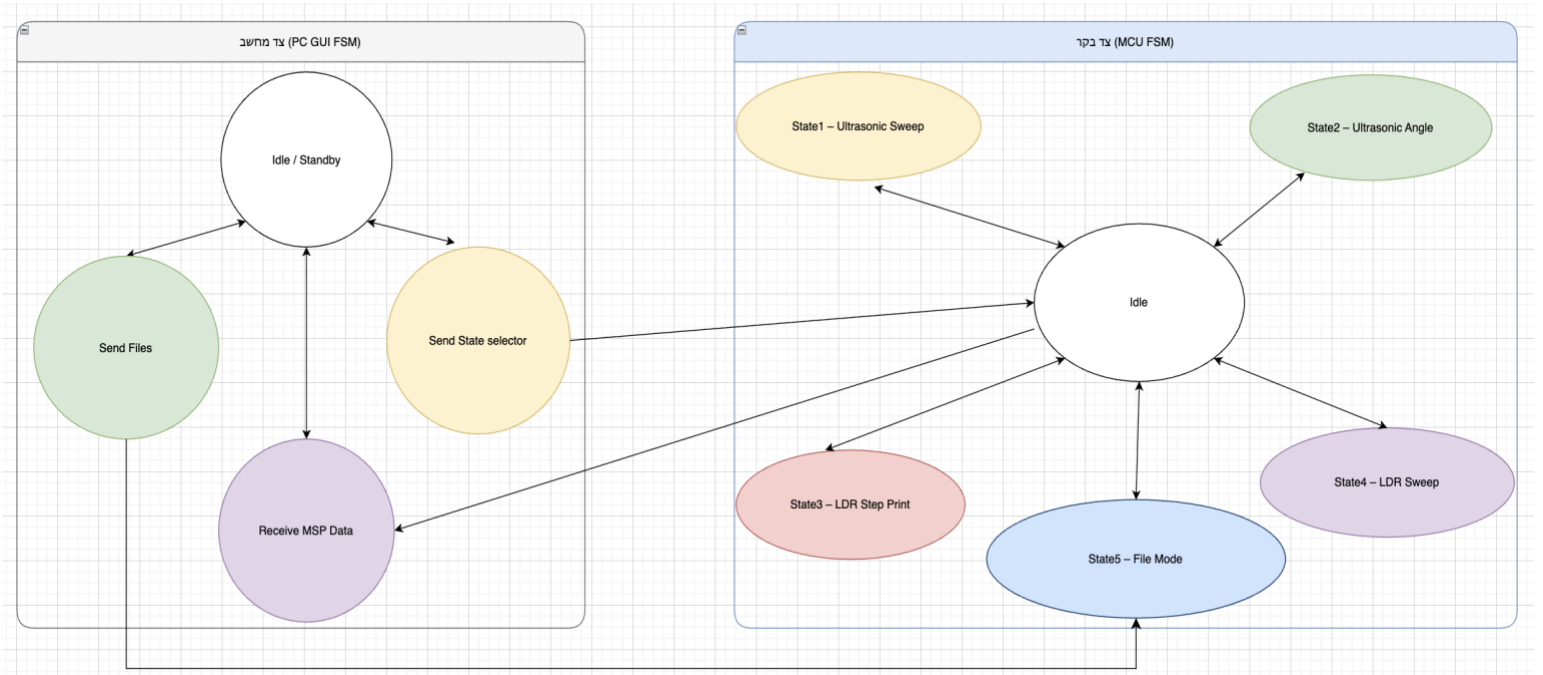
- סרבו – ערכי היצרן מגדירים טווח עבודה של  $0^{\circ}$ – $180^{\circ}$  עבור פולסים ברוחב ms2–1. בוצע כיול מעשי שבו נמדדה זווית בפועל עבור מספר ערכי PWM ונבנה מיפוי ליניארי. כדי למנוע עומס בקצוות, הוגדר טווח עבודה מצומצם ( $10^{\circ}$ – $170^{\circ}$ ).
- חיישן מרחק HC-SR04 – לפי היצרן הטווח 2–400 ס"מ. בוצע כיול בעזרת שליט/מדידה ידנית במרחקים ידועים (30, 60, 100 ס"מ). התקבל גרף זמן-Echo < מרחק בפועל, והוזנו מקדמי תיקון בתוכנה.
- LDR - נעשה כיול מול מקור אור בעוצמות ידועות (100, 200 ו-LUX 500). התקבל גרף התנגדות מול עוצמת אור, ולפיו הוגדרו ספי החלטה במערכת.
- LCD וכפתורים – לא נדרש כיול; השתמשנו בהגדרות ברירת מחזל של ערכת הפיתוח.

### חלוקת העבודה בין חומרה לתוכנה

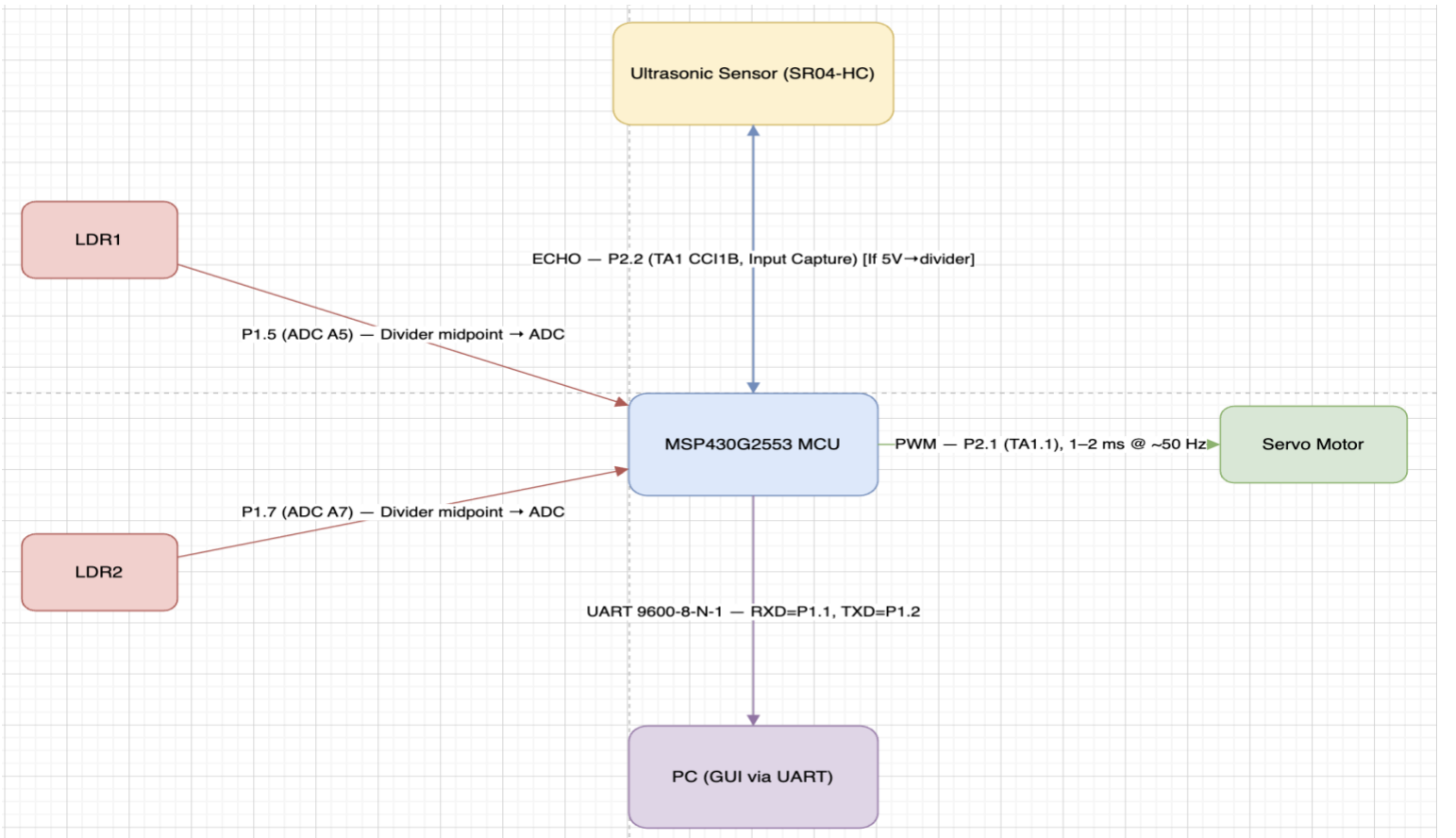
- חומרה – מספקת את האותות הגולמיים: פולס Echo מהחיישן האולטראסוני, ערכי ADC מה-LDR, לחיצות כפתורים, הצגת תווים ב-LCD והפעלה חשמלית של הסרבו.
- תוכנה – אחראית לעיבוד המידע ולשליטה ברמות גבוהות יותר:
  - BSP: אתחול שעונים, פורטים ויחידות תקשורת.
  - HAL: מימוש פונקציות בסיסיות להפעלת LCD, סרבו, חיישני ADC ו-UART.
  - API: ניהול מערכת הקבצים ב-Flash, תקשורת עם המחשב בפרוטוקול עצור וחכה, אימות checksum ושליחת ACK/NACK.
  - APP: מימוש FSM ראשי, File Mode, הרצת סקריפטים וזרימת לוגיקה כללית.

באופן זה, ישנה הפרדה ברורה: החומרה אוספת נתונים ומפעילה רכיבים, והתוכנה מממשת את האלגוריתמיקה, הבקרה והאינטגרציה בין הרכיבים.

## להלן דיאגרמת FSM של צד המחשב וצד הבקר:



## מעגל אלקטרוני של חומרת הקצה המחוברת ל – MCU:



## שאלות בנושא תכנון המערכת (תכנון מבוסס אופטימיזציה של ביצועים) (סיבוכיות זמן), נפתח קוד (סיבוכיות מקום), וצריכת הספק):

### • גילוי אובייקטים במרחב:

- מספר הדגימות בכל סריקה מסומן ב-N ואצלנו הוא שווה ל-180. גודל צעד הסרבו במעלות לכל דגימה הינו  $\theta$ , אצלנו צעד של מעלה אחת ולכן  $\Delta\theta = 1$  מעלות. נסמן ב- $\omega_{servo}$  את מהירות סיבוב הסרבו במעלות לשניה נסמן ב- $t_{settle}$  את זמן ההתייצבות לאחר כל צעד נסמן ב- $t_{meas}$  את זמן מדידת החיישן. אם מודדים מרחק מרבי  $d_{max}$  אז בקירוב  $t_{meas} \approx \frac{2 \cdot d_{max}}{c_{sound}}$  כאשר  $c_{sound} = 343 \frac{m}{s}$  נסמן ב- $t_{proc}$  את זמן עיבוד נתונים והדפסה נסמן ב- $t_{return}$  את זמן החזרה של הסרבו לנקודת ההתחלה בין הסריקות, לרוב יהיה  $\frac{180}{\omega_{servo}}$ , ועוד מרווח התייצבות קטן מחזור דגימה בודדת:  $t_{sample} = \frac{180}{\omega_{servo}} + t_{settle} + t_{meas} + t_{proc}$  מחזור סריקה מלאה של 180 דגימות:  $T_{scan} = N \cdot t_{sample} + t_{return}$  תדר סריקה:  $f_{scan} = \frac{1}{T_{scan}}$  תדר הדגימה הממוצע לאורך הסריקה הוא:  $f_{sample} = \frac{N}{T_{scan}}$  ניקח את הערכים של הערכה:  $\omega_{servo} = 400 \frac{^\circ}{s}$ ,  $t_{settle} = 5ms$ ,  $d_{max} = 2m \Rightarrow t_{meas} = 11.7ms$ ,  $t_{proc} = 0.2ms$  נקבל,  $t_{return} = 0.47ms$ ,  $t_{sample} = 0.0197s$ ,  $T_{scan} = 4.016s$ ,  $f_{scan} = 0.249Hz$ ,  $f_{sample} = 44.8 \frac{samples}{s}$
  - במערכת, בכל צעד של סריקת הסרבו נאספת דגימה אחת אשר מכילה מידע משולב משלושה מקורות: זווית הסרבו, מדידת מרחק מהחיישן האולטראסוני, ועוצמות האור הנקלטות על ידי זוג חיישני LDR.
    - זווית הסרבו – נשמרת כערך בודד בתחום  $0-180^\circ$ , כאשר דיוק של מעלה אחת מספק. די בכך לאחסן את הערך ב-Byte יחיד (8 ביט).
    - מרחק מחיישן האולטראסוני – החיישן מסוג HC-SR04 מודד מרחקים בתחום 2–400 ס"מ. ייצוג הערך דורש 16 ביט (2 Bytes), המאפשר טווח נוח עד 65,535 ס"מ.
    - עוצמת אור מחיישני LDR – כל חיישן מחובר לכניסת ADC10 של הבקר, המספקת ערך בן 10 ביט (0–1023). בפועל, נשמרים שני ערכים כאלה (שני חיישנים), וביחד נדרשים 4 Bytes.
- מכאן, סך כל דגימה בודדת תופסת הוא 7 Bytes. במהלך סריקה מלאה נאספות 180 דגימות, כך שגודל הנתונים הכולל לסריקה אחת הוא 1260 Bytes = 1.23KB.

משמעות הדגימה – כל דגימה מייצגת נקודת מדידה בודדת במרחב, המתארת את מצב הסביבה בכיוון שבו מצביע הסרבו: הזווית בה נלקחה המדידה, המרחק לעצם הקרוב ביותר באותו כיוון, ועוצמות האור שנקלטו בחיישני ה-LDR. שילוב של 180 דגימות יוצר תמונה מרחבית של סביבת המערכת הן במונחי מרחקים והן במונחי מקורות אור.

### 3. בצד הבקר (MCU – MSP430G2553)

- שכבה פיזית – חיישנים מחוברים ישירות לפינים של הבקר: חיישן האולטראסוני מחובר לטיימר Capture, חיישני ה-LDR מחוברים לכניסות ה-ADC10, והסרבו נשלט ע"י PWM מטיימר A\_.
- עיבוד ראשוני – הבקר מבצע את קריאת הערכים הגולמיים (ADC, Echo, לחיצות כפתורים) והמרתם לגדלים פיזיקליים: מרחק בס"מ, עוצמת אור ביחידות יחסיות.
- ניהול FSM – לוגיקת התוכנה בבקר מנהלת את הסריקות (180 צעדים), שומרת את הדגימות בזיכרון, ומחליטה מתי להציג נתונים על ה-LCD או לשלוח אותם למחשב.
- חיסכון במשאבים – כל דגימה מצומצמת ל-7 בתים בלבד (כפי שחושב בסעיף הקודם), כדי להתאים למגבלת הזיכרון הפנימי (512 B RAM בלבד).

### בצד המחשב (PC)

- המחשב מקבל מהבקר את רצף הדגימות הגולמיות.
- בעיבוד ברמת PC ניתן לבצע פעולות חישוביות כבדות יותר, כגון:
  - שרטוט מפה מרחבית של הסביבה (Polar Plot: זווית-מרחק-עוצמת אור).
  - עיבוד נתונים סטטיסטי (מיצוע, סינון רעשים).
  - הצגה גרפית נוחה למשתמש.
- המחשב לא מתעסק בפעולות בזמן אמת (כמו שליטה בסרבו או חישוב מרחק מה Echo) – הוא מקבל תוצאה מוכנה לעיבוד.

### המידע העובר בערוץ התקשורת (UART):

- כל דגימה: 7 בתים (1 בית זווית, 2 בתים מרחק, 4 בתים אור).
- עבור סריקה מלאה (180 דגימות): 1.23KB.
- הערוץ מוגדר כ-UART בפרוטוקול עצור וחכה, עם מנגנון בדיקת שגיאות (8 Checksum ביט).
- בנוסף לנתוני הדגימה עצמם, עוברים בערוץ גם מסגרות בקרה קצרות (ACK/NACK) לצורך סנכרון ואישור קבלה.

### שיקול הנדסי לחלוקת העיבוד

ההחלטה להשאיר את החישוב הראשוני בבקר (מדידת מרחק, קריאת ADC, שמירת דגימות קומפקטית) נובעת ממגבלות של תזמון בזמן אמת: רק הבקר יכול למדוד את פולס ה-Echo בדיוק של מיקרו-שניות ולקרוא ADC בסנכרון עם תנועת הסרבו.

לעומת זאת, המחשב חופשי ממשאבים וזמן אמת, ולכן מתאים לביצוע עיבוד כבד וגרפי שלא ניתן לבצע על ה-MSP430 עקב מגבלות זיכרון ומהירות.

### 4. נבנה ביטוי כללי לנצילות המידע לכל דגימה אחת בערוץ הטורי עם עצור וחכה. הגדרות:

גודל המטען השימושי בבתים הוא  $B_p$  אצלנו הוא שווה 7 בתים.

סה"כ בתים של ראש מסגרת לדגימה אחת H

סה"כ בתים של זנה מסגרת לדגימה אחת T

בתים לבקרת שגיאות (למשל checksum) נסמן C

בתים של אישור מהצד השני נסמן A הכוונה ל-ACK

נסמן ב-F פקטור מסגרת

$$\eta = \frac{B_p}{F \cdot (B_p + H + T + C + A)}$$

5. במהלך סריקה מלאה נאספות 180 דגימות – לכל זווית  $\varphi$  בין  $0^\circ$  ל- $180^\circ$ . כל דגימה מכילה

הזווית של הסרבו  $\varphi$

ערכי עוצמות האור משני חיישני ה-LDR  $l$

המרחק הנמדד בחיישן האולטראסוני  $\rho$

שלבי האלגוריתם לזיהוי אובייקטים:

- איסוף דגימות – המערכת שומרת את כל 180 הדגימות ברשימה.
- סינון רעשים – מתבצע מיצוע על חלון קטן (למשל 3–5 דגימות) כדי להחליק סטיות חדות.
- איתור גבולות אובייקט – מוגדר סף החלטה: אם  $\rho$  קטן באופן משמעותי לעומת השכנים, מזהים התחלה של אובייקט; כאשר הערך חוזר לגדול – זו סיום האובייקט.
- חישוב מיקום האובייקט – עבור כל אובייקט שנמצא, מחשבים את הזווית המרכזית  $\varphi$ , את המרחק הממוצע  $\rho$ , ואת ממוצע ערכי האור  $l$ .
- שמירת תוצאות – כל אובייקט מיוצג כשלישייה  $(\rho, \varphi, l)$ .

```
Input: samples[0..179] // כל דגימה מכילה {phi, rho, l}
Output: objects[] // רשימת אובייקטים {rho, phi, l}

objects = []
in_object = false
current_obj = {}

for i from 0 to 179:
    phi = samples[i].phi
    rho = samples[i].rho
    l = samples[i].l

    // סינון רעשים בסיסי
    rho = average(rho, samples[i-1].rho, samples[i+1].rho)

    if rho < THRESHOLD and not in_object:
        // התחלת אובייקט חדש
        in_object = true
        current_obj.start_phi = phi
        current_obj.rho_values = [rho]
        current_obj.l_values = [l]

    else if in_object:
        // ממשיכים לאסוף נתוני האובייקט
        current_obj.rho_values.append(rho)
        current_obj.l_values.append(l)

    if rho >= THRESHOLD or i == 179:
        // סיום אובייקט
        in_object = false
        current_obj.end_phi = phi

        // חישוב ערכים מיוצגים
        current_obj.phi = (current_obj.start_phi + current_obj.end_phi)/2
        current_obj.rho = average(current_obj.rho_values)
        current_obj.l = average(current_obj.l_values)

        objects.append(current_obj)
        current_obj = {}

return objects
```



• גילוי מקורות אור במרחב:

1. מספר הדגימות בכל סריקה מסומן ב-N ואצלנו הוא שווה ל-180.

גודל צעד הזווית במעלות לכל דגימה הינו  $\Delta\theta$   
 נסמן ב- $\omega_{servo}$  את מהירות הסרבו במעלות לשניה  
 נסמן ב- $t_{settle}$  את זמן ההתייצבות המכני לאחר כל צעד  
 נסמן ב- $n_{ldr}$  את מספר קריאות LDR בכל קריאה, אצלנו שווה 2.  
 נסמן ב- $t_{proc}$  אם צריך זמן עדכון קצר לעיבוד נתונים והדפסה  
 נסמן ב- $t_{adc}$  זמן המרה של ערך יחיד ב-ADC.  
 נסמן ב- $t_{return}$  את זמן החזרה לנקודת ההתחלה בין סריקות, אם הלוך חוזר ניתן להציב 0.  

$$t_{sample} = \frac{\Delta\theta}{\omega_{servo}} \cdot t_{settle} \cdot n_{ldr} \cdot t_{proc} \cdot t_{adc}$$
 מחזור דגימה בודדת:  

$$T_{scan} = N \cdot t_{sample} + t_{return}$$
 מחזור סריקה מלאה של 180 דגימות:  

$$f_{scan} = \frac{1}{T_{scan}}$$
 תדר סריקה:  

$$f_{sample} = \frac{N}{T_{scan}}$$
 תדר הדגימה הממוצע לאורך הסריקה הוא:

2. בכל זווית סרבו נאספת דגימה אחת. כל דגימה מכילה:

- זווית הסרבו ( $\varphi$ )
  - טווח  $0^\circ-180^\circ$
  - דיוק של מעלה אחת
  - מיוצג ב-1 בתים
- עצמת אור מחיישן LDR שמאלי
  - נקלט ע"י ADC 10, (10 ביטים, 0-1023)
  - מאוחסן ב-2 בתים
- עצמת אור מחיישן LDR ימני
  - כנ"ל – עוד 2 בתים

סה"כ גודל דגימה אחת הינה 5 בתים וגודל של 180 דגימות בסריקה אחת הינו 0.9KB.  
 כל דגימה מייצגת נקודת מידע אחת במרחב:

- הזווית בה הסרבו מצביע
- עצמת האור שנמדדה בכל אחד משני ה-LDR

באופן זה, רצף של 180 דגימות יוצר פרופיל מרחבי של עוצמות האור מכל כיוון. ההשוואה בין שני החיישנים מאפשרת להעריך האם מקור האור ממוקם יותר בצד שמאל או ימין, ובשילוב עם הזווית ניתן לאמוד את כיוון מקור האור.

3. בצד הבקר (MSP430G2553)

- שכבה פיזית – הסרבו מוזז צעד צעד ( $1^\circ$  לכל דגימה) בעזרת אות PWM מטיימר. בכל צעד, הבקר קורא את הערכים האנלוגיים משני חיישני ה-LDR דרך ממיר ה-ADC10 הפנימי.

- עיבוד ראשוני – הבקר שומר לכל זווית  $\varphi$  את זוג הערכים הדיגיטליים של ה-LDR (שמאלי וימני), ומאגד אותם יחד עם הזווית לדגימה אחת.
- ניהול נתונים – הדגימות מאוחסנות בטבלה בזיכרון (180 שורות  $\times$  5 בתים כל אחת). אין חישוב כבד על הבקר, כדי לחסוך משאבים ולשמור על זמן תגובה בזמן אמת.

#### בצד המחשב (PC)

- המחשב מקבל את הדגימות הגולמיות מהבקר ומבצע את העיבוד המתקדם:
  - ציור גרף עוצמות אור כתלות בזווית (Profile של 180 מעלות).
  - חישוב כיוון מקור האור (הזווית שבה נמדדה עוצמת האור המקסימלית).
  - אפשרות למיצוע או פילטר דיגיטלי כדי להפחית רעשים.
- הבחירה להשאיר את החישוב למחשב מבוססת על שיקול הנדסי: למחשב אין מגבלת זיכרון או זמן תגובה, בעוד הבקר מוגבל מאוד (RAM 512 בתים בלבד).

#### המידע העובר בערוץ התקשורת (UART)

- כל דגימה כוללת:
  - זווית: 1 בתים
  - LDR שמאלי: 2 בתים
  - LDR ימני: 2 בתים
  - סה"כ: 5 בתים
- לסריקה שלמה:

180 כפול 5, סה"כ 0.9 ק"ב

- בנוסף לנתוני המדידה עצמם, עובר בערוץ גם מידע בקרה קצר (Header, Checksum, ACK/NACK) בפרוטוקול עצור וחכה.
- כך מבטיחים שגם אם נופל שגיאת תקשורת, הדגימה תישלח מחדש ולא תאבד מידע.

#### שיקול הנדסי מרכזי:

הבקר אוסף את הנתונים הגולמיים בלבד ושולח אותם במבנה קומפקטי של 5 בתים לדגימה. המחשב, שלו משאבים חישוביים וזיכרון כמעט בלתי מוגבלים, מבצע את כל האלגוריתמיקה לזיהוי מקור האור והצגתו למשתמש. כך נשמר איזון נכון בין עבודה בזמן אמת על הבקר לבין עיבוד כבד בצד המחשב.

4. זהה לסעיף הקודם רק שכאן  $B_p$  שווה 5 במקום 7.
- גודל המטען השימושי בבתים הוא  $B_p$  אצלנו הוא שווה 5 בתים.
- סה"כ בתים של ראש מסגרת לדגימה אחת H
- סה"כ בתים של זנה מסגרת לדגימה אחת T
- בתים לבקרת שגיאות (למשל checksum) נסמן C
- בתים של אישור מהצד השני נסמן A הכוונה ל-ACK
- נסמן ב-F פקטור מסגרת

$$\eta = \frac{B_p}{F \cdot (B_p + H + T + C + A)}$$

5. תיאור האלגוריתם:  
במהלך סריקה מלאה נאספות 180 דגימות. כל דגימה מכילה:

- זווית הסרבו  $\varphi$
  - עוצמת אור מחיישן LDR שמאלי
  - עוצמת אור מחיישן LDR ימני
- מטרת האלגוריתם היא לזהות את מקורות האור במרחב, ולחשב עבור כל מקור את ערכיו

שלבי האלגוריתם:

1. איסוף דגימות – קריאת 180 הזוויות וערכי שני ה-LDR.
2. חישוב עוצמה משולבת – ניתן לבחור בסכום שתי הקריאות או במקסימום ביניהן.
3. תיקון זווית – לפי ההפרש בין החיישנים, מחשבים הטיית זווית קטנה
4. סינון רעשים – ביצוע מיצוע נע קצר על רצף הדגימות.
5. איתור מקורות אור – זיהוי פיקים בעוצמה מעל סף החלטה TH. כל פיק מייצג מקור אור.
6. חישוב ערכים – לכל פיק מחושבים הזווית המרכזית  $\varphi$ , העוצמה  $l$ , ובמידה ויש כיוול – המרחק  $\rho$

```
// קלט: samples[0..179] // כל דגימה: {phi, L_left, L_right}
// אופציונלי (rho, phi, l) - אור רשימת מקורות // sources[]

epsilon ← מספר קטן למניעת חילוק באפס
k_diff ← קבוע כיוול לזווית
TH ← סף פיק לעוצמה
sources ← []

// עיבוד ראשוני
for i = 0 .. 179:
    Lsum = samples[i].L_left + samples[i].L_right + epsilon
    D = (samples[i].L_right - samples[i].L_left) / Lsum
    delta = k_diff * D
    phi_s = samples[i].phi + delta
    l_i = max(samples[i].L_left, samples[i].L_right)
    שמור PhiStar[i] = phi_s
    שמור Light[i] = l_i

// סינון רעשים
PhiStar = moving_average(PhiStar, window=3)
Light = moving_average(Light, window=3)

// איתור מקורות אור
peaks = find_local_maxima(Light, min_prominence=TH)
for p in peaks:
    i0 = center_index_of_peak(p)
    phi_hat = PhiStar[i0]
    l_hat = Light[i0]

    if כיוול זמין:
        rho_hat = sqrt(k_lux / l_hat)
    else:
        rho_hat = לא מוגדר

    sources.append({rho: rho_hat, phi: phi_hat, l: l_hat})

החזר sources
```

## • מד מרחק

1. החלטת ערך המרחק של האובייקט הנמדד מתקבלת על סמך מדידת רוחב פולס ה־Echo מהחיישן האולטראסוני, כאשר המרחק מחושב לפי:  $\rho = \frac{c \cdot t_{echo}}{2}$ . כאשר c זה מהירות הקול ו  $t_{echo}$  זה זמן הפולס הנמדד. כדי להתמודד עם רעשים במדידה, המערכת מבצעת מספר קריאות רצופות ומיישמת סינון פשוט: ערכים חריגים (מחוץ לטווח העבודה או שונים משמעותית מהממוצע) נדחים, והערך הסופי נקבע כממוצע של המדידות התקינות. באופן זה מתקבלת החלטה יציבה ואמינה יותר על המרחק הנמדד, תוך הפחתת השפעת רעשים והחזרים שגויים.
2. חיישן המרחק האולטראסוני פועל בשיטת Time of Flight: שליחת פולס קול והמתנה לקבלת ההחזר. הזמן הנדרש למדידה אחת תלוי במרחק המקסימלי האפשרי  $\rho_{max}$ , משום שיש להמתין עד שההד החוזר יגיע (או עד שתסתיים תקופת הזמן המקסימלית אם אין החזר). ביטוי כללי למחזור מדידה:  $T_{meas} = \frac{2 \cdot \rho_{max}}{c}$  כאשר תדר הדגימה שווה  $f_{meas} = \frac{1}{T_{meas}}$ . התדר מוגבל פיזיקלית ע"י זמן המעבר של גלי הקול – לפני שניתן לשלוח פולס חדש חייבים להמתין לסיום האפשרי של הפולס הקודם. לכן ככל שהמרחק הנמדד קטן יותר אפשר להעלות את תדר הדגימה בפועל, אך עבור טווח מרבי יש להסתפק בסדר גודל של עשרות הרץ בלבד.
3. החלטה על שינוי המרחק הפיזי של האובייקט הנמדד מתקבלת על ידי השוואה בין ערכי המדידה הנוכחיים לערכים הקודמים שנשמרו. כאשר ממוצע המדידות האחרונות (לאחר סינון רעשים) שונה באופן מובהק מערך קודם ביותר מסף החלטה מוגדר מראש  $\Delta \rho_{th}$  המערכת מסיקה כי האובייקט זו. במילים אחרות, אם  $\Delta \rho_{th} < |\rho_{new} - \rho_{old}|$  אזי המערכת קובעת שהתרחש שינוי במרחק הפיזי. אם ההפרש קטן מהסף אזי ההבדל מיוחס לרעש או לשגיאת מדידה, והמערכת שומרת על ערך המרחק הקודם. באופן זה נמנעים זיהויים שגויים ומתקבלת החלטה יציבה לגבי שינוי אמיתי במיקום האובייקט.

## • מערכת קבצים

1. קוד המערכת מאוחסן בסגמנטים של Main Flash בזיכרון ה־Flash של הבקר, בטווח הכתובות 0xC000-0xFFFF. הקוד עצמו נטען לאזור זה, בעוד שהכתובות העליונות (0xFFE0-0xFFFF) שמורות לטבלאות וקטורי הפסיקה.
2. קבצי הטקסט והסקריפטים נשמרים בזיכרון ה־Flash של הבקר באופן הבא:  
ספריית הקבצים נשמרת בסגמנטים של Information Memory בטווח הכתובות 0x1000-0x10FF.  
תוכן הקבצים נשמר בסגמנטים פנויים של Main Flash בטווח הכתובות הגבוהות 0xC000-0xFFFF, בצד הקוד, אך באזור ייעודי שהוגדר לאחסון נתונים.
3. בבקר MSP430G2553 קיים אזור ייעודי קטן בשם Information Memory, בנפח 256 בתים (ארבעה סגמנטים בגודל B64 כל אחד):

- Segment A → 0x10C0–0x10FF •
- Segment B → 0x1080–0x10BF •
- Segment C → 0x1040–0x107F •
- Segment D → 0x1000–0x103F •

במערכת שלנו הוחלט לשמור את מבנה הנתונים לניהול מערכת הקבצים (כלומר טבלת הספרייה – שמות קבצים, סוג קובץ, מצביע לאזור ה-Main Flash שבו נשמר התוכן) דווקא ב-Information Memory, מכיוון ש:

- זהו אזור נפרד מה-Main Flash ולכן לא מתנגש עם קוד או עם תוכן הקבצים עצמם.
- הנפח הקטן (B64 לכל סגמנט) מתאים מאוד לאחסון טבלאות קטנות.
- ניתן למחוק ולעדכן סגמנטים בודדים בלי לפגוע בשאר הזיכרון.

#### 4. מבנה הנתונים וניהולו במערכת קבצים (נפח KB2)

מבנה נתונים (Directory):

- טבלת ספרייה קטנה נשמרת ב-Information Memory.
- כל רשומת ספרייה כוללת: שם קובץ, סוג (טקסט/סקריפט), גודל, offset ב-Main Flash, checksum, דגל .valid
- בנוסף נשמר מצביע next\_free לכתובת הפנויה הבאה במאגר.

ניהול אחסון:

- הוספת קובץ – כתיבה ל-Main Flash החל מ-`next_free`, יצירת רשומה בטבלה ועדכון המצביע.
- קריאת קובץ – חיפוש רשומה תקפה לפי שם וקריאת התוכן מה-Flash.
- מחיקה – סימון הרשומה כלא תקפה (מחיקה לוגית).
- אריזה מחדש – בעת חוסר מקום, העתקת קבצים תקפים לתחילת המאגר ועדכון ה-`offset`ים.

```
def write_file(name, buf, n):
    if n > POOL_SIZE - dir.next_free:
        compact()
    write_flash_at(POOL_BASE + dir.next_free, buf)
    add_entry(name, size=n, offset=dir.next_free, valid=1)
    dir.next_free += n

def read_file(name):
    e = find_entry(name)
    if e.valid:
        return flash_read(POOL_BASE + e.offset, e.size)

def delete_file(name):
    e = find_entry(name)
    e.valid = 0
```

## כיוול חומרה ובניית דרייברים לחיישנים ולרכיבי קצה:

### • מנוע סרבו

1. עבור 0 מעלות Ton יהיה שווה 1ms ועבור 180 מעלות הוא יהיה שווה 2ms.
2. הפקת אות הבקרה לסרבו

אות הבקרה למנוע הסרבו הוא אות PWM עם מחזור של כ-20ms (תדר  $\approx 50\text{Hz}$ ) ורוחב פולס משתנה בין 1-2ms לפי זווית הסרבו הרצויה. בבקר ה-MSP430 הפקת האות נעשית באמצעות:

- I. Timer-A פועל במצב Up Mode:
  - מונה מ-0 ועד ערך CCR0 שמוגדר כך שתקופת הספירה שווה ל-20ms.
  - כאשר המונה מגיע ל-CCR0, הוא מתאפס ומתחיל מחדש – וכך מתקבלת מחזוריות קבועה.
- II. יחידת Comapre (למשל CCR1) – מוגדרת במצב Set/Reset (או Reset/Set):
  - בתחילת המחזור הפלט מוגדר ל-1 (High).
  - כשהמונה מגיע לערך CCR1, הפלט מתאפס ל-0 (Low).
  - התוצאה היא פולס ברוחב Ton, שניתן לשלוט עליו על ידי שינוי הערך ב-CCR1.

אופן העבודה של כל טיימר

- Timer-A במצב Up Mode קובע את מחזור אות ה-PWM (20ms).
- יחידת Comapre (למשל CCR1) – קובע את זמן ההשהיה שבו הפלט יורד ל-0 בתוך כל מחזור, ובכך שולט על זווית הסרבו.

3. אות הבקרה של מנוע הסרבו מופק באופן רציף וקבוע, גם כאשר לא מתבצעת תנועה.

- מנוע סרבו סטנדרטי (כגון SG90) זקוק לאות PWM רציף (מחזור של 20ms) כדי לשמור על המיקום.
- אם מפסיקים את האות, המנוע עובר למצב "חופשי" (Free Run), ואז הזרוע יכולה לזוז כתוצאה מחיכוך, כבידה או רעידות.
- לכן הבקר ממשיך לייצר את אות ה-PWM כל הזמן, כאשר רוחב הפולס (Ton) מייצג את הזווית הרצויה:

1. אם אין דרישה לתנועה → הערך ב-CCR1 נשאר קבוע.
2. אם נדרשת תנועה → הבקר משנה את ערך CCR1, וכך משתנה רוחב הפולס והמנוע נע לזווית החדשה.

### • חיישן מרחק

1. גרעיניות המדידה (Granularity):

- טיימר A\_ בבקר MSP430G2553 הוא טיימר בן 16 ביט.
- כאשר מגדירים אותו עם מקור שעון  $SMCLK = 1\text{MHz}$ , מתקבלת רזולוציה של  $1\mu\text{s}$  לכל טיקט של הטיימר.
- כלומר, הגרעיניות של מדידת רוחב הפולס (Echo) היא מיקרו-שניה אחת.

הערך הגולמי המתקבל:

- במצב Capture Input, הטיימר לוכד את ערך המונה ברגע קצה עולה וברגע קצה יורד של אות ה-Echo.
- ההפרש בין שני הערכים הוא  $t_{echo}$  – משך הזמן שבו האות היה פעיל.
- הערך הגולמי שמתקבל הוא מספר צעדים של הטיימר ביחידות של 1  $\mu s$ .

קשר בין הערך למרחק:

$$\rho = \frac{c \cdot t_{echo}}{2} = \frac{c \cdot (N \cdot T_{tick})}{2}$$

כאשר N זה מספר הטיקים שנמדד,  $1\mu s = T_{tick}$

2. אות ה-Trigger שמפעיל את חיישן המרחק מופק על ידי יצירת פולס מרובע קצר באורך של כ-10 מיקרו-שניות על רגל ה-Trigger של החיישן. הפולס הזה מופק בעזרת יחידות הטיימר הפנימיות של הבקר.

הטיימרים המעורבים ואופן עבודתם:

- טיימר מסוג Timer\_A מוגדר במצב Up Mode, כך שהמונה סופר מאפס עד לערך ב-CCR0.
- ערוץ השוואה CCR1 מוגדר במצב Set/Reset. בתחילת מחזור המנייה הפלט עולה ל-High, וכשהמונה מגיע לערך CCR1 הפלט יורד ל-Low.
- בחירה של CCR1 = 10 (כאשר תדר השעון הוא 1MHz, כלומר כל טיק = 1  $\mu s$ ) יוצרת פולס ברוחב 10  $\mu s$ .

## • חיישן מרחק ממקור אור

1. עשרים הדגימות לכיול חיישן מרחק ממקור אור נשמרות באזור Information Memory של ה-Flash הקצאת הכתובות מתבצעת באופן הבא:

- חיישן LDR שמאלי – הדגימות נשמרות ב-Segment D, בטווח הכתובות 0x1000-0x103F.
- חיישן LDR ימני – הדגימות נשמרות ב-Segment C, בטווח הכתובות 0x1040-0x107F.

לכל חיישן נשמרות עשר דגימות רצופות ניתן לשמור כל דגימה כ מילה של ששה עשר סיביות וכן להשאיר מספר בתים לסימון גרסה ובקרת שלמות.

2. עיקרון הפעולה:

- במהלך הכיול נאספות 10 דגימות לכל חיישן LDR (ערך ADC שנמדד עבור מרחק ידוע).
- הדגימות מייצרות זוגות נקודות (d,ADC) לאורך טווח המרחקים.
- כאשר נדרש ערך ביניים עבור מרחק שלא נמדד בפועל, מבוצעת אינטרפולציה ליניארית בין שתי הנקודות הקרובות:

$$ADC(d) = ADC(d_i) + \frac{d - d_i}{d_{i+1} - d_i} (ADC(d_{i+1}) - ADC(d_i))$$

בצד בו מתבצעת ההשלמה:

- ההשלמה מתבצעת בצד המחשב (PC), ולא בבקר.
- הסיבה: לבקר יש משאבים מוגבלים מאוד (זיכרון Flash, RAM 512 בתים קטן), ולכן הוא רק שולח את ערכי הדגימה הגולמיים.
- המחשב, בעל יכולת חישובית גבוהה, מבצע את החישוב והאינטרפולציה כדי לבנות את גרף המרחק המלא.

שמירת הגרף:

- הגרף המלא שנבנה לאחר אינטרפולציה נשמר במחשב.
- בבקר נשמרים אך ורק ערכי הדגימות הגולמיים ב־ Information Memory (10 דגימות לכל חיישן).

3. חיישן ה־LDR משנה את התנגדותו בהתאם לעוצמת האור הפוגע בו. ככל שהמקור רחוק יותר, כמות האור המגיעה קטנה יותר והתנגדות של ה־LDR משתנה בהתאם.

- קריאת ערך אנלוגי – המתח על ה־LDR מחובר לכניסת ה־ADC של הבקר (ADC10). הבקר מבצע המרה אנלוגית-לדיגיטל ומקבל ערך מספרי ADC בטווח 0–1023.
- שימוש בגרף הכיול – במהלך הכיול נשמרו 10 נקודות מדידה לכל חיישן, המתארות את ההתאמה בין ערך ADC לבין מרחק פיזי ידוע.
- אינטרפולציה – כאשר מתקבל ערך ADC חדש, המחשב משווה אותו לנקודות הכיול הקרובות ביותר ומבצע אינטרפולציה ליניארית כדי לאמוד את המרחק המתאים.

ביטוי מתמטי:

$$\rho = d_i + \frac{ADC - ADC(d_i)}{ADC(d_{i+1}) - ADC(d_i)} (d_{i+1} - d_i)$$

כאשר:  $\rho$  - המרחק הנמדד בס"מ

$d_i, d_{i+1}$  – נקודות המרחק מהכיול ובהתאם  $ADC(d_i), ADC(d_{i+1})$  ערכי ה־ADC שנמדדו בהתאם לערכי הכיול

## • מבנה נתונים ואלגוריתמים

1. מבנה הנתונים לקליטת פריים בערוץ RX – הפריים שמתקבל בערוץ התקשורת כולל את המרכיבים הבאים:

- שדה Header – משמש כסימן פתיחה ומאפשר למקלט לזהות את תחילת המסגרת.
- שדה Length – מציין את גודל המטען בביתים ומונע קריאה מעבר לסיום.
- שדה Payload – מכיל את תוכן המידע עצמו, למשל דגימות או פקודות.
- שדה Checksum – משמש לאימות שלמות המידע; אם החישוב אינו תואם, המסגרת נפסלת.
- שדה Tail – משמש כסימן סיום ומוודא שהמסגרת הסתיימה במקום הנכון.

2. גודל הפריים אינו קבוע אלא משתנה. הסיבה לכך היא שהמטען (Payload) של המסגרת יכול להכיל כמויות מידע שונות – למשל דגימה אחת, מספר דגימות, או מקטע מקובץ. כדי להתמודד עם שונות זו, בתוך כל פריים משולב שדה Length המציין במדויק את גודל המטען בביתים. בזכות שדה זה המערכת יודעת היכן מסתיים



הפריים גם כאשר כמות הנתונים שונה מפריים לפריים, מה שמאפשר העברת מידע באופן דינאמי ויעיל מבלי לבזבז מקום על מבנים קבועים וגדולים מדי.

3. מבנה הנתונים לשליחת פריים בערוץ TX - הפריים המשודר דרך ערוץ התקשורת כולל את המרכיבים הבאים:

- שדה Header – משמש כסימן פתיחה ומאפשר לצד המקבל לזהות את תחילת המסגרת.
- שדה Length – מציין את גודל המטען בבתים ומספק מידע על אורך הפריים.
- שדה Payload – מכיל את תוכן המידע עצמו, לדוגמה דגימות או פקודות.
- שדה Checksum – משמש לאימות שלמות המידע; במקרה של אי-התאמה הפריים נפסל.
- שדה Tail – משמש כסימן סיום ומוודא שהמסגרת הסתיימה בצורה תקינה.

4. גודל הפריים המשודר דרך ערוץ התקשורת TX הוא משתנה. הסיבה לכך היא שחלק מהשדות בפריים (Header, Length, Checksum, Tail) הם בעלי גודל קבוע מראש, אבל השדה המרכזי – Payload – יכול להשתנות בהתאם לכמות המידע שיש להעביר. אם נדרשת שליחה של דגימה בודדת, המטען יהיה קצר, ואם יש צורך בשליחה של מספר דגימות או מקטע מקובץ, המטען יהיה ארוך יותר. כדי שהמקבל ידע להתמודד עם שינוי זה, בכל פריים יש שדה Length שמציין את אורך המטען המדויק, וכך ניתן לזהות את סיום המסגרת גם כאשר גודל המטען משתנה.

5. המערכת תומכת בגילוי שגיאות בהעברת מידע דרך ערוץ התקשורת, וזאת באמצעות מנגנון Checksum בגודל 8 ביט (מודולו 256).

אופן המימוש בצד הבקר (TX):

- בעת יצירת פריים, הבקר מחשב את סכום כל בתים במטען (Payload).
- הסכום מחושב מודולו 256 (כלומר רק שמונה הביטים התחתונים נשמרים).
- ערך ה-Checksum נוסף לשדה הייעודי בפריים לפני שליחתו.

אופן המימוש בצד המחשב (RX):

- לאחר קבלת פריים, המחשב מחשב מחדש את סכום כל בתים במטען (מודולו 256).
- אם הערך המחושב תואם לערך שנשלח בשדה ה-Checksum, הפריים נחשב תקין.
- במקרה של אי-התאמה, הפריים נפסל והמחשב יכול לבקש שליחה חוזרת (Stop and Wait).

6. המערכת אכן תומכת בפרוטוקול Stop-and-Wait Automatic Repeat Request, כך שניתן להבטיח שהמידע המשודר יגיע באופן אמין.

אופן המימוש בצד הבקר (Transmitter):

- לאחר יצירת פריים ושליחתו, הבקר נכנס למצב המתנה ל-ACK.
- מוגדר טיימר פנימי לזמן מקסימלי (Timeout).
- אם מתקבלת הודעת ACK תקינה מהמחשב  $\Rightarrow$  הבקר מתקדם לפריים הבא.
- אם מתקבלת הודעת NACK (כלומר זוהתה שגיאה בפריים) או שהטיימר פג מבלי לקבל תגובה  $\Rightarrow$  הבקר שולח מחדש את אותו פריים.
- תהליך זה נמשך עד קבלת ACK תקין או עד שמספר ניסיונות החזור יסתיים.

אופן המימוש בצד המחשב (Receiver):

- לאחר קבלת פריים דרך ערוץ RX, המחשב מבצע חישוב Checksum (מודולו 256).
- אם המידע תקין  $\leq$  נשלחת חזרה הודעת ACK לבקר.
- אם התגלה פגם ב־Checksum או במבנה המסגרת  $\leq$  נשלחת הודעת NACK לבקר.
- כך הבקר יודע אם לשמור על רצף השליחה או לחזור על פריים שנפגם.

## מיפוי המשתנים בזיכרון:

סקירת חלוקת זיכרון - בקר MSP430G2553 מחלק את הזיכרון למספר אזורים:

- זיכרון תכנית (Flash) – משמש לאחסון קוד התוכנה ולמערכת הקבצים במצב File Mode.
- זיכרון עבודה (RAM) – משמש לאחסון משתנים מקומיים, משתנים גלובליים וערכי stack.
- רגיסטרים ייחודיים (SFR) – כתובות מיוחדות לשליטה ברכיבי החומרה כמו UART, GPIO, ADC וטיימרים.
- כתובות פריפריה – אזור במרחב הזיכרון שממופה ישירות לרכיבי חומרה פנימיים (Timer\_A, ADC10, ) (USCI).

קובץ main.c תפקידו לממש את מכונת המצבים הראשית (FSM) ואת נקודת הכניסה הראשית:

- מערך line באורך שישים וארבע תווים – buffer לקלט UART, נשמר ב־RAM.
- משתנים len ו־ticks – מונים פשוטים ב־RAM.
- משתנה cm מסוג uint16\_t – תוצאה של מדידת מרחק מחיישן אולטראסוניק, נשמר ב־RAM.
- פונקציות עזר tx\_line, tx\_write, tx\_putc ו־delay\_ms – הקוד נשמר בזיכרון Flash.

קובץ api.c שכבת השירות ברמה הגבוהה:

- פונקציה api\_servo\_set\_angle – מקבלת ערך זווית, משתמשת במשתנים מקומיים ב־RAM.
- פונקציה api\_ultra\_get\_cm – מחזירה תוצאה דרך מצביע ל־RAM.
- פונקציה api\_ldr\_step\_print – קוראת ערכים מחיישני LDR ושומרת תוצאות זמניות ב־RAM.
- פונקציות סריקה (sweep) – מפעילות את שכבת HAL, אינן מחזיקות משתנים גלובליים.

קובץ hal.c שכבת ה־HAL לגישה ישירה לחומרה:

- אין שימוש במשתנים גדולים, אלא גישה ישירה לרגיסטרים.
- דוגמאות: UCA0TXBUF, IFG2, P1DIR, ADC10CTL0 – כולם ממופים ל־SFR ולא נשמרים ב־RAM.

קובץ flash.c שכבת ניהול זיכרון Flash עבור File Mode:

- מבנה file\_entry – כולל שם קובץ, גודל וכתובת, נשמר ב־Flash.
- טבלת קבצים – מאוחסנת ב־Flash Segments.
- נתוני הקבצים עצמם – נשמרים ב־Flash Program Memory.
- בעת קריאה התוכן מועתק ל־RAM buffers זמניים.

קובץ bsp.c: שכבת BSP:

- שימוש במאקרו מסוג define עבור מיפוי פינים – אינו תופס זיכרון, מתורגם בזמן קומפילציה.
- פונקציות clock\_init, uart\_init ו-sysConfig – פועלות ישירות על רגיסטרים כמו DCOCTL, BCSCTL1, UCA0CTL1.

קובצי header: קובצי bsp.h, hal.h, api.h, app.h ו-flash.h.

- תפקידם להצהיר על פונקציות, משתנים וקבועים.
- אינם מקצים זיכרון בפועל.
- מספקים הפשטה והפרדה בין שכבות התוכנה.

קובץ GUI.py - קובץ פייתון בצד המחשב האישי.

- מנהל ממשק גרפי ושולח פקודות דרך UART.
- כל המשתנים כאן מאוחסנים בזיכרון המחשב, ללא השפעה על זיכרון ה-MSP430.

## מסקנות והצעות לשיפורים:

### מסקנות:

- ניהול הזיכרון במערכת בוצע בצורה יעילה בהתחשב במגבלות הבקר MSP430G2553.
- השימוש ב-RAM ממוקד במשתנים זמניים, buffers ומוני עזר, תוך הקפדה על חסכון עקב מגבלת חמש מאות ושתיים בייט.
- זיכרון ה-Flash מנוצל לא רק לאחסון קוד אלא גם למימוש File Mode, מה שמגדיל את הגמישות של המערכת.
- רגיסטרים (SFR) מנוצלים ישירות דרך שכבת ה-HAL ללא צורך במשתנים נוספים ב-RAM.
- חלוקת השכבות (BSP → HAL → API → APP) מייעלת את ניהול הזיכרון ומאפשרת ניידות גבוהה של הקוד בין בקרי MSP430 שונים.
- עם זאת, המערכת אינה כוללת מנגנון מובנה לניהול שגיאות בזיכרון או ב-File Mode, מה שעשוי לגרום לאיבוד מידע במקרים של תקלה בכתיבה ל-Flash.

### הצעות לשיפורים:

- ניתן להוסיף מנגנוני איתור ותיקון שגיאות (כגון CRC או checksum) עבור קבצים המאוחסנים ב-Flash, לשם שיפור אמינות File Mode.
- מומלץ ליישם מצב Error State ב-FSM שיטפל במקרים של שגיאת קריאה/כתיבה או overflow ב-buffer.
- שימוש ב-circular buffer ל-UART RX/TX יכול להפחית עומס על RAM ולמנוע איבוד נתונים בתקשורת.
- ניתן לשקול כיווץ משתנים (לדוגמה שימוש ב-bitfields במבני נתונים) כדי לחסוך RAM.
- הרחבת File Mode באמצעות ניהול טבלת קבצים דינמית או תמיכה במספר גדול יותר של קבצים תגדיל את יכולת השימוש.
- בצד ה-GUI ניתן להוסיף תצוגות גרפיות (גרפים בזמן אמת) שיאפשרו ניטור וניתוח טוב יותר של נתוני ה-MCU, מבלי להעמיס על משאבי הזיכרון בבקר עצמו.

