



בית הספר להנדסת חשמל ומחשבים

דו"ח תיעוד פרויקט מסכם קורס "מבנה מחשבים ספרתיים"
36114191

**תכנון מערכת לגילוי מקורות אור וניטור אובייקטים
במרחב**

ادر שפירא 209580208

יהונתן דכה 211468582

01/09/2025

תוכן עניינים

2.....	תוכן עניינים
3.....	מטרת הפרויקט
3.....	תיאור הפרויקט
5.....	דיאגרמת FSM
5.....	מעגל אלקטרוני של חומרת הקצה המחברת ל – MCU
6.....	שאלות בנושא תכנון המערכת
6.....	גילוי אובייקטים למרחב
9.....	גילוי מקורות אוור במרחב
12.....	מד מרחק
12.....	מערכת קבצים
14.....	כיוול חומרה ובנית דרייברים להישנים ולרכיבי קצה
14.....	מנוע סרבו
14.....	חישון מרחק
15.....	חישון מרחק ממוקור אוור
16.....	מבנה נתונים ואלגוריתמים
19.....	מייפוי המשתנים בזיכרון
19.....	מסקנות והצעות לשיפורים

הגדרת ומטרת הפרויקט

מטרת הפרויקט היא לתוכנן ולממש מערכת משובצת (Embedded System) המבוססת על בקר מיקרו-קונט롤ר, אשר משלבת יכולות חישמה, בקרה ותקשורת. המערכת נדרשת לאפשר גילוי אובייקטים ומקורות אור במרחב באמצעות סריקה בסרבו, מדידת מרחק בעורת חישון אולטראוסוני, 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 מודול שכבות מלא: BSP, HAL, API, APP. המערכת מחוברת למנווע סרבו, חישון מרחק אולטראוסוני, שני חישוני אור (LDR), ותצוגה LCD. המשמש יכול להפעיל את המערכת במצב סריקה, במצב מדידה, ובמצב ניהול קבצים.

במצב File Mode ניתן לדפס 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 והעיבודים. מערכת הקבצים תומכת בקבלת קבצים מהמחשב דרך תקשורת טורית בפרוטוקול עוצר וחכה עם חישוב checksum לאיומות שלמות הנתונים.

המערכת מדגימה אינטגרציה מלאה של חומרה ותוכנה: ניהול משאבי MCU, טיפול באירועי משתמש, שמרות מידע ב-Flash, פרוטוקולי תקשורת, והפעלת אלגוריתמים פשוטים לחישמה ובקרה. תוכרי 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-קונטroller MSP430G2553 של Texas Instruments, אשר שולב במערכת האישית ששימשה אותנו במהלך הפרויקט. הבקר הוא ליבת 16 סיביות, בעל זיכרון פנימי, מיר ADC, טיימרים וממשק תקשורת מובנים. סביב הבקר חוברו מספר רכיבים פנימיים (המסופקים כחלק מהערכת הפיתוח) ורכיבי קצה חיצוניים שנוטפו בימוש הפרויקט.

רכיבים פנימיים בברker ובערכת הפיתוח

- תצוגת LCD (2*16) – משמשת להצגת נתונים, הודיעות מערכת ותוכן קבצים. ה-LCD מחובר לפורטים דיגיטליים של הבקר ונשלט דרך ממשק HAL תוכנתי.
- כפתורים (PB0, PB1) – מוגדרים כקלטי GPIO, מאפשרים דפוד בין קבצים (PB0) ובבחירה/כניתה (PB1).
- זיכרון Flash פנימי – משמש גם לאחסון הקוד וגם למשימם מערכת קבצים פשוטה. ספריית הקבצים נשמרת ב-Code Flash, בעוד תוכן הקבצים (טקסט/סקריפט) מאוחSEN בסגמנטים ייעודיים ב-Information Memory.

רכיבי קצה חיצוניים

- מנוע סרבו (SG90) – מחובר לפין PWM של Timer_A. אפשר סריקה בזווית שונה לצורך גילוי עצמים או מקורות אור.
- חיישן מרחק אולטראוסוני (HC-SR04) – מחובר ל-GPIO וטיימר Capture לממדית פולס Echo, ומאפשר מדידת מרחק בס"מ.
- חיישני אור (LDR ×2) – התנדותם משתנה כתלות בעוצמת האור. חוברו לכניסות ADC10 של הבקר ומשמשים לזיהוי כיוון מקור אור.

- תקשורת UART – מבוצעת דרך מודול A0 USCI הפנימי של הבקר, לצורך קבלת קבצים מהמחשב ושליחת ACK/NACK בהתאם ל프וטוקול “עוצר וחכה”.

כיוול חומרה מול ערכיו יצירנו

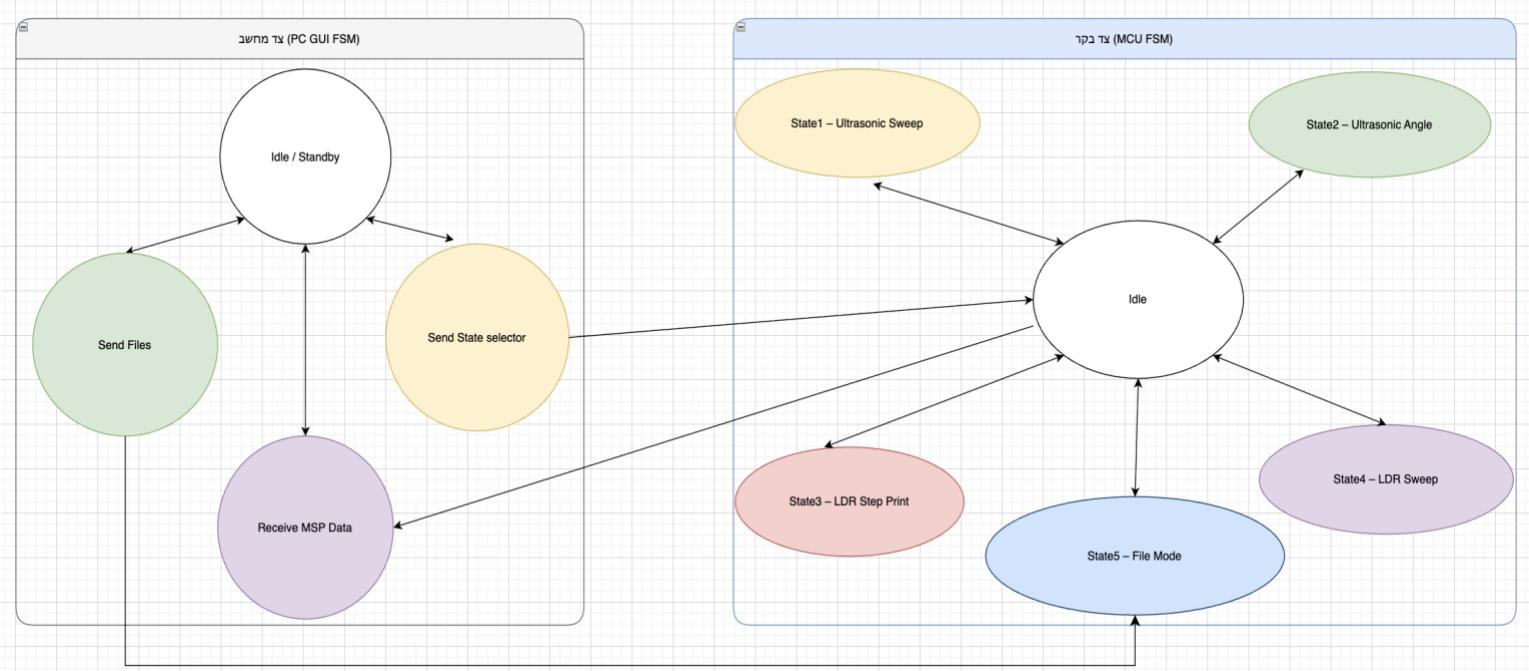
- סרבו – ערכי הייצרן מגדרים טווח עבודה של 0° – 180° עבור פולסים ברוחב 1–2ms. בוצע כיוול מעשי שבו נמדדה זווית בפועל עבור מספר ערכי PWM ונבנה מיפוי ליניארי. כדי למנוע עומס בקצבות, הוגדר טווח עבודה מצומצם (-10° – 170°).
- חיישן מרחק HC-SR04 – לפי הייצרן הטווח 2–400 ס"מ. בוצע כיוול בעזרת שליטה/מדידה ידנית במרחקים ידועים (60, 100, 30 ס"מ). התקבל גוף זמני Echo-> מרחק בפועל, והזנו מקדמי תיקון בתוכנה.
- LDR - נעשה כיוול מול מירור אוור בעוצמות ידועות (100, 200, 500 LUX). התקבל גוף התנגדות מול עצמאית אוור, ולפיו הוגדרו ספי השליטה במערכת.
- LCD וכפתורים – לא נדרש כיוול; השתמשנו בהגדרות בירית מחדל של מערכת הפיתוח.

חלוקת העבודה בין חומרה לתוכנה

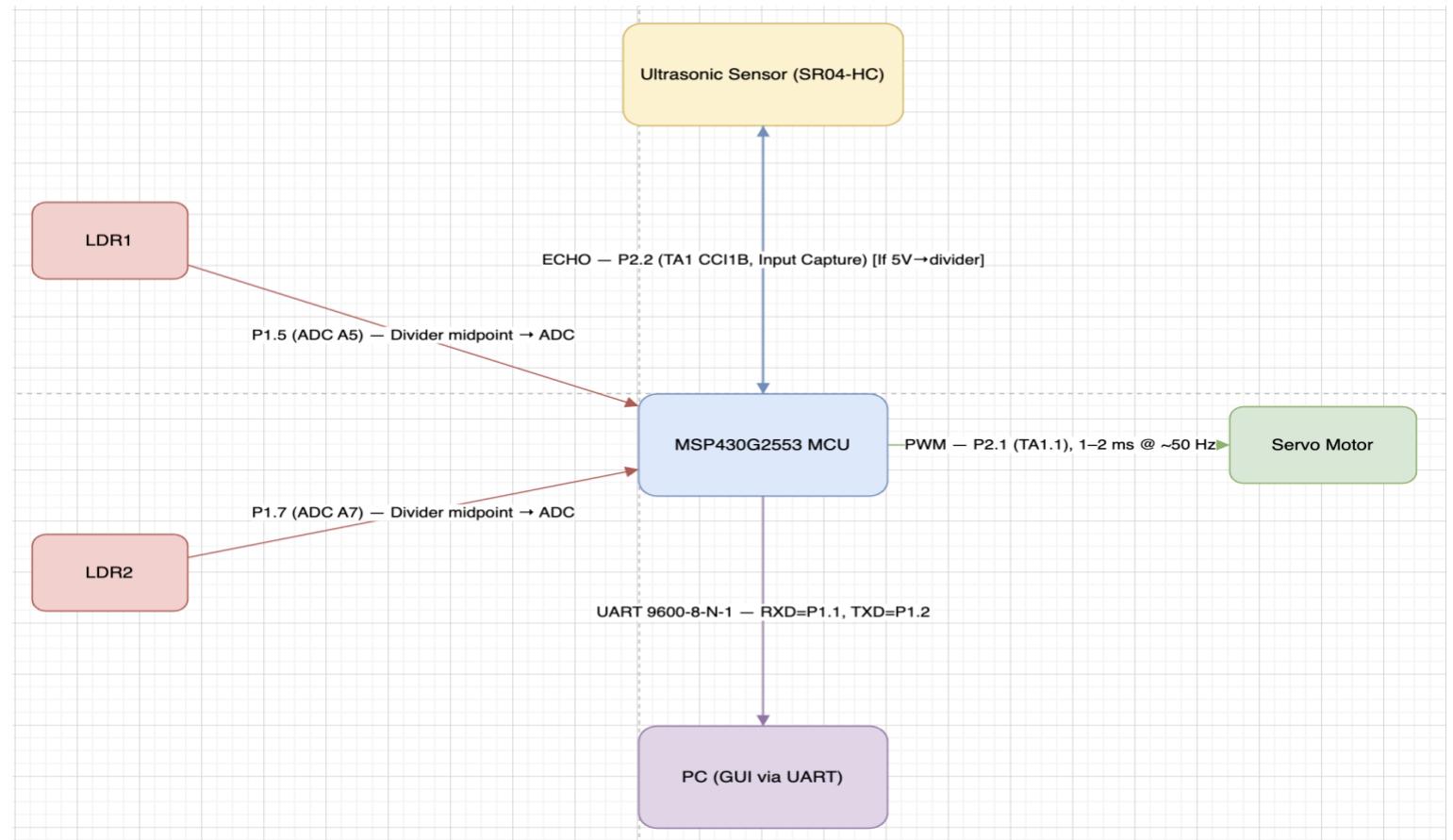
- חומרה – מספקת את האותות הגלמיים: פולס Echo מהחיישן האולטראוסוני, ערכי ADC מה-LDR, לחיצות כפתורים, הצגת תווים ב-LCD והפעלה החשמלית של הסרבו.
- תוכנה – אחראית לעיבוד המידע ולשליטה בرمמות גבוהות יותר:
 - atakhol שעונים, פורטים ויחידות תקשורת: BSP
 - HAL: מימוש פונקציות בסיסיות להפעלה LCD סרבו, חיישני ADC ו-UART.
 - API: ניהול מערכת הקבצים ב-Flash, תקשורת עם המחשב בפוטוקול עוצר וחכה, אימות ACK/NACK ושליחת checksum.
 - APP: מימוש FSM ראשי, File Mode, הרצה סקורייפים ווריאנט לוגיקה כללית.

באופן זה, ישנה הפרדה ברורה: החומרה אוספת נתונים ומבצעת רכיבים, והתוכנה ממלמת את האלגוריתמיקה, הבדיקה והאינטרציה בין הרכיבים.

להלן דיאגרמת FSM של צד המחשב וצד הAKER:



מעגל אלקטרוני של חומרת הקצה המחברת ל-MCU –



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

- גילוי אובייקטים למרחב:

1. מספר הדגימות בכל סריקה מסומן ב- N ואצלנו הוא שווה ל-180. גודל צעד הסרבו במעלות לכל דגימה הינו θ , אצלנו צעד של מעלה אחת ולכן $\Delta\theta = 1$ מעלה. נסמן ב- ω_{servo} את מהירות סיבוב הסרבו במעלות לשניה נסמן ב- t_{settle} את זמן ההתייצבות לאחר כל צעד נסמן ב- t_{meas} את זמן מדידת החישון. אם מודדים מרחק מרבי d_{max} אז בקירוב $t_{meas} \approx \frac{2 \cdot d_{max}}{c_{sound}}$ כאשר $c_{sound} = \frac{m}{s} = 343$ נסמן ב- t_{proc} את זמן צורך קצר לעיבוד נתונים והדפסה נסמן ב- t_{return} את זמן החזרה של הסרבו לנקודת ההתחלת בין הסריקות, לרוב יהיה $\frac{180}{\omega_{servo}}$, ועוד מרובה התהייצבות קטן מהזור דגימה בודדת: $t_{sample} = \frac{180}{\omega_{servo}} + t_{settle} + t_{meas} + t_{proc}$ מהזור 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:
 $\omega_{servo} = 400 \frac{\circ}{s}$, $t_{settle} = 5ms$, $d_{max} = 2m \Rightarrow t_{meas} = 11.7ms$, $t_{proc} = 0.2ms$
 $t_{return} = 0.47ms$, $t_{sample} = 0.0197s$, $T_{scan} = 4.016s$, $f_{scan} = 0.249Hz$
 $f_{sample} = 44.8 \frac{samples}{s}$
2. במערכת, בכל צעד של סריקה הסרבו נאספת דגימה אחת אשר מכילה מידע משולב שלושה מקורות: זווית הסרבו, מדידת מרחק מהחישון האולטראסוני, ועוצמות האור הנקלטות על ידי זוג חיישני LDR. זווית הסרבו – נשמרת כערך בודד בתחום $0^\circ - 180^\circ$, כאשר דיוק של מעלה אחת מספק. די בכך לאחסן את הערך ב-Byte (8 ביט). מרחק מהחישון האולטראסוני – החישון מסוג HC-SR04 מודד מרחקים בתחום 2–400 ס"מ. ייצוג הערך דורש 16 ביט (Bytes 2), המאפשר טווח נוח עד 65,535 ס"מ. עוצמת אור מהחיישני LDR – כל חישון מחובר לכנית ADC10 של הבקר, המספקת ערך בן 10 ביט (0–1023). בפועל, נשמרים שני ערכים כאלה (שני חיישנים), וביחד נדרשים Bytes 4.

מכאן, סך כל דגימה בודדת תופסת הוא 7 Bytes.
 במהלך סריקה מלאה נאספות 180 דגימות, כך שגודל הנתונים הכללי לסריקה אחת הוא = 1260 Bytes = 1.23KB

משמעות הדגימה – כל דגימה מייצגת נקודת מדידה בודדת למרחב, המתארת את מצב הסביבה בכיוון שבו מוצב הסרבו: הזרות בה נלקחה המדידה, המרחק לעצם הקרוב ביותר באותו כיוון, ועוצמות האור שנקלטו בחישוני הדגימה. שילוב של 180 דגימות יוצר תמונה מרוחבית של סביבת המערכת הן במונחים מרוחקים והן במונחים מוקrovות או.

3. בצד הבקר (MCU – MSP430G2553)

- שכבה פיזית – חישנים מחוברים ישירות לפינים של הבקר: חישון האולטראוסוני מחובר לטיימר Capture, ADC10, PWM מטימר_A.
- חישוני ה-DTR מחוברים לכניות ה-ADC, והסربו נשלט ע"י מטימר_Echo, ADC, להיצות כפתורים) והמרתם לגדים עיבוד ראשוני – הבקר מבצע את קריאת הערכיים הגלומיים (FSM – לוגיקת התוכנה בברק מנהלת את הסirkיות (180 עצדים), שומרת את הדגימות בזיכרון, ומחליטה מתי להציג נתונים על ה-LCD או לשולח אותם למחשב.
- פיזיקליים: מרחק בס"מ, עוצמת אור ביחידות יחסיות.
- ניהול FSM – לוגיקת התוכנה בברק מנהלת את הסirkיות (180 עצדים), שומרת את הדגימות בזיכרון, ומחליטה מתי להציג נתונים על ה-LCD או לשולח אותם למחשב.
- חישון במשאים – כל דגימה מצומצמת ל-7 בתים בלבד (כפי שהושב בסעיף הקודם), כדי להתאים למוגבלת הזיכרון הפנימי (B RAM512 בלבד).

בצד המחשב (PC)

- המחשב מקבל מהברק את רצף הדגימות הגלומיות.
- בעיבוד ברמת PC ניתן לבצע פעולות חשוביות כבודות יותר, כגון:

 - שרטוט מפה מרוחבית של הסביבה (Polar Plot): זווית–מרחק–עוצמת אור.
 - עיבוד נתונים סטטיסטי (ሚוצע, סינון רעשים).
 - הצגה גרפית נוחה למשתמש.

- המחשב לא מתעסק בפעולות בזמן אמיתי (כמו שליטה בסרבו או חישוב מרחק מה Echo) – הוא מקבל תוצאה מוכנה לעיבוד.

המידע העובר בערוץ התקשות (UART):

- כל דגימה: 7 בתים (1 בית זווית, 2 בתים מרחק, 4 בתים אור).
- עברו סירקה מלאה (180 דגימות): 1.23KB.
- הערוץ מוגדר כ-UART בפרוטוקול עיצור וחכה, עם מגנון בדיקת שגיאות (Checksum ביט).
- בנוסף לנתוני הדגימה עצם, עוברים בערוץ גם מסגרות בקרה קצרות (ACK/NACK) לצורך סync'ון ואישור קבלת.

שיקול הנדסי לחלוקת העיבוד

ההגולטה להשאיר את החישוב הראשוני בברק (מדידת מרחק, קריאת ADC, שמירת דגימות קומפקטיבית) נובעת מمبرאות של זמן בזמן אמיתי: רק הברק יכול לבדוק את פולס הדס – בדיקת של מיקוד-שנויות ולקרוא ADC בסync'ון עם תנועת הסרבו.

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

4. בניית בייטוי כללי לניצולו 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 לבקרה שאיות (checksum C נסמך

בתים של אישור מהצד השני נסמך ב 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 φ בין 0° ל- 180° . כל דגימה מכילה

הزوית של הסרבו – φ

ערכי עצמות האור משני היישני R – L

המרחק הנמדד ביחסן האולטארסוני – ρ

שלבי האלגוריתם לזיהוי אובייקטים:

- איסוף דגימות – המערכת שומרת את כל 180 הדגימות בראשימה.
- סינון רעשים – מתחבע מיוצע על הלון קטן (למשל 3–5 דגימות) כדי להחלק סטיות הדות.
- אישור גבולות אובייקט – מוגדר סף החלטה: אם ρ קטן באופן משמעותי לעומת השכנים, מזהים התחלת של אובייקט; כאשר הערך חוזר גדול – זו סיום האובייקט.
- חישוב מיקום האובייקט – עבור כל אובייקט שנמצא, מחשבים את הזוויות המרכזיות φ , את המרחק הממוצע ρ , ואת ממוצע ערכי האור l .
- שבירת תוצאות – כל אובייקט מוצג כשלשיה (ρ, φ, l) .

```

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

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:
        # החלה 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 נתונים לתוך אובייקט
        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 מרכזים מיצגים
            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- ω_{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 קרייה, אצלנו שווה 2.
נסמן ב- t_{proc} אם צריך זמן קצר לעיבוד נתונים והדפסה
נסמן ב- t_{adc} זמן המירה של ערך יחיד ב-ADC.
נסמן ב- t_{return} את זמן החזרה לנקודת ההתחלת בין סריקות, אם הלווק חזר ניתן להציג 0.
מחזורי דגימה בזוזת: $t_{sample} = \frac{\Delta\theta}{\omega_{servo}} \cdot t_{settle} \cdot n_{ldr} \cdot t_{proc} \cdot t_{adc}$
מחזורי סריקה מלאה של 180 דגימות: $T_{scan} = N \cdot t_{sample} + t_{return}$
תדר סריקה: $f_{scan} = \frac{1}{T_{scan}}$
תדר הדגימה המומוצע לאורק הסריקה הוא: $f_{sample} = \frac{N}{T_{scan}}$

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

- זווית הסרבו (φ)
 - טווח $0^\circ - 180^\circ$
 - דיק של מעלה אחת
 - מוצג ב-1 בתים
- עצמת אור מהישן LDR שמאל
 - נקלט ע"י ADC 10, (10 ביטים, 0-1023)
 - מאוחסן ב-2 בתים
- עצמת אור מהישן 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 מצבי
- עצמת האור שנמדדה בכל אחד משני ה-LDR

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

3. בצד הבקר (MSP430G2553)

- שכבה פיזית – הסרבו מזוז צעד (1° לכל דגימה) בעזרת אות PWM מטיימר. בכל צעד, הבקר קורא את הערכים האנלוגיים משני חישוני ה-LDR דרך ממיר ה-ADC10 הפנימי.

- עיבוד ראשוני – הבקר שומר לכל זווית φ את זוג הערכיהם הדיגיטליים של ה-LDR (شمאלית וימנית), ומאנד אותם יחד עם הזווית לדגימה אחת.
- ניהול נתונים – הדגימות מאוחסנות בטבלה בזיכרון (180 שורות × 5 בתים כל אחת). אין חישוב כבד על הבקר, כדי להחסוך משאבים ולשמור על זמן תגובה בזמן אמת.

בצד המחשב (PC)

- המחשב מקבל את הדגימות הגולמיות מהבקר ובמוצע את העיבוד המתකדם:
- ציר גרפ' עצומות או כתלות בזווית (Profile) של 180 מעלות).
- חישוב כיוון מקור האור (הזווית שבה נמדדה עצמת האור המקסימלית).
- אפשרות למייעז או פילטר דיגיטלי כדי להפחית רעש.
- הבחירה להשאיר את החישוב למחשב מבוססת על שיקול הנדסי: למחשב אין מגבלת זיכרון או זמן תגובה, בעוד**>bakr mogenel malot (512 RAM 512 בתים בלבד).**

המידע העובר בערוץ התקשרות (UART)

- כל דגימה כוללת:
 - זווית: 1 בתים
 - LDR שמאלית: 2 בתים
 - LDR ימנית: 2 בתים
 - סה"כ: 5 בתים
- לש:right; לסריקה שלמה:

180 כפול 5, סה"כ 0.9 ק"ב

- בנוסף לנדרי המדייה עצם, עובר בערוץ גם מידע בקרה קצר (Header, Checksum, ACK/NACK) בפרוטוקול עצור ומחכה.
- כך מבטיחים שגם אם נופל שגיאת תקשורת, הדגימה תישלח מחדש ולא תאביד מידע.

шиיקול הנדסי מרכזי:

הברker אוסף את הנתונים הגולמיים בלבד ושולח אותם במבנה קומפקטי של 5 בתים לדגימה. המחשב, שלו משאבים חישוביים וזכרון כמעט בלתי מוגבלים, מבצע את כל האלגוריתמיקה לזריה מיוקו האור והציגו למשתמש. כך נשמר איזון נכון בין עבודה בזמן אמת על הברker לבין עיבוד כבד מצד המחשב.

- זהה לסעיף הקודם רק שכואן B_p שהוא 5 במקום 7.
גודל המטען השימושי בבתים הוא B_p אצלנו הוא שווה 5 בתים.
סה"כ בתים של זנה מסגרת לדגימה אחת H
סה"כ בתים של זנה מסגרת לדגימה אחת T
בתים לבקרה שגיאות (למשל checksum) נסמן C
בתים של אישור מהצד השני נסמן A הכוונה ל-ACK-
נסמן 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 φ
- עצמת אור מהישן LDR שמאלית
- עצמת אור מהישן LDR ימני

מטרת האלגוריתם היא להזות את מקורות האור במרחב, ולהשכUbOR כל מקור את ערכיו

שלבי האלגוריתם:

1. איסוף דגימות – קריית 180 הזווית וערך שני ה-LDR.
2. חישוב עצמה משולבת – ניתן לבחור בסכום שתי הקריאות או במקסימום ביניהן.
3. תיקון זווית – לפי ההפרש בין החישונים, מחשבים הטית 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 φ , העוצמה l , ובמידה ויש כיוול – המרחק r

```

קיט: samples[0..179] // כל דגימה
      sources[] // אוסף רשומות מקורות {rho, phi, l} (rho)

epsilon ← מספר קטן למינית חילוק באפס
k_diff ← קבוע כיוול לדזוזין
TH ← סף פיק לטזומה
sources [] ← []

// ייבוד 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)
    PhiStar[i] = phi_s
    Light[i] = l_i

// סינון רמשים
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 כינול כינול:
        rho_hat = sqrt(k_lux / l_hat)
    else:
        rho_hat = לא מוגדר

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

הגדה sources

```

• מד מרחק

1. החלטת ערך המרחק של האובייקט הנמדד מתקבלת על סמך מדידת רוחב פולס ה-Echo מהחישון האולטראוסוני, כאשר המרחק מחושב לפי: $\rho = \frac{c \cdot t_{echo}}{2}$. כאשר c זה מהירות הקול ו- t_{echo} זה זמן הפולס הנמדד. כדי להתחמוד עם רעשים במדידה, המערכת מבצעת מספר קריאות רצופות ומישמת סינון פשוט: ערכיהם הריגים (מחוץ לטווח העבודה או שונים ממשמעותית מהמוצע) נדחים, והערך הסופי נקבע כממוצע של המדידות התקינות. באופן זה מתקבלת החלטה יציבה ואmine יותר על המרחק הנמדד, תוך הפחחת השפעת רעשים והחזרים שגויים.

2. חישון המרחק האולטראוסוני פועל בשיטת Time of Flight: שליחת פולס קול והמתנה לקבלת החזר. הזמן הנדרש למדידה אחת תלוי במרקם המקסימלי האפשרי ρ_{max} , משומם שיש להמתין עד שהחזר יגיע (או עד שתסתהים תקופה $f_{meas} = \frac{2 \cdot \rho_{max}}{c}$ כאשר תדר הדגימה שווה $= \frac{1}{T_{meas}}$ הזמן המקסימלי אם אין החזר). ביטוי כללי למוחזר מדידה: $\Delta\rho_{th}$ המרכיב מסיקה כי האובייקט זו. במקרים אחרים, אם $|\rho_{new} - \rho_{old}| < \Delta\rho_{th}$ אז המערכת מוגדר מראש מראש - $\Delta\rho_{th}$ המרכיב מסיקה כי האובייקט זה. במקרה אחר, אך עבור טווח מרבי יש להסתפק בסדר גודל של עשרות הרץ בלבד.

3. החלטה על שינוי המרחק הפיזי של האובייקט הנמדד מתקבלת על ידי השוואת בין ערכי המדידה הנוכחיים לערכים הקודמים שנשמרו. כאשר ממוצע המדידות לאחררונות (לאחר סינון רעשים) שונה מובהק מוקדם ביותר מסף החלטה מוגדר מראש $\Delta\rho_{th}$ המרכיב מסיקה כי האובייקט זו. במקרים אחרים, אם $|\rho_{new} - \rho_{old}| < \Delta\rho_{th}$ אז המערכת קובעת שהתרחש שינוי במרקם הפיזי. אם הפרש קטן מהספ אזי ההבדל מיוחס לרעש או לשגיאת מדידה, והמערכת שומרת על ערך המרחק הקודם. באופן זה מנעים 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, בטוווח 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: ספרירית הקבצים נשמר בסגמנטים של Information Memory בטוווח הכתובות 0x1000-0x10FF. תוכן הקבצים נשמר בסגמנטים פנויים של Main Flash בטוווח הכתובות הגבוהות 0xC000-0xFFFF, הצד הקוד, אך באזור ייעודי שהוגדר לאחסון נתונים.
3. בAKER MSP430G2553 קיימים אזורי ייעודי קטן בשם Information Memory, בנפח 256 בתים (ארבעה סגמנטים בגודל כל אחד):

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

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

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

4. מבנה הנתונים וניהולו במערכת קבצים (נפח KB2)

מבנה נתונים (Directory):

- טבלת ספרייה קטנה נשמרה ב-Information Memory.
- כל רשומה ספרייה כוללת: שם קובץ, סוג (טקסט/סקרייפט), גודל, offset ב-Flash, checksum (טקסט/סקרייפט).
- בנוסף נשמר מצביע next_free לכתובת הפנوية הבאה במאגר.

ניהול אחסון:

- הוספה קובץ – כתיבה ל-Flash Main החל מ-offset next_free, יצרת רשומה בטבלה ועדכון המצביע.
- קריאת קובץ – חיפוש רשומה תקפה לפי שם וקריאת התוכן מה-Flash.
- מחיקה – סימון הרשומה ללא תקפה (מחיקה לוגית).
- אריזה מחדש – בעת חוסר מקום, העתקת קבצים תקפים לתחילה המאגר ועדכון offset_offset.

```
jonathan@jonathan: ~ Downloads > -> pagedc.py > ...
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

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

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

כיוול חומרה ובנית דרייברים לחישנים ולרכיבי קצה:

• מנוע סרבו

1. עברו 0 מעלות Ton יהיה שווה 1ms ומעבר 180 מעלות הוא יהיה שווה 2ms.
2. הפקת אותן הבקרה לסרבו

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

- I. Timer-A פועל במצב Up Mode:
 - מונה מ-0 ועד ערך CCR0 שמדובר כך שתקופת הספירה שווה ל-20ms.
 - כאשר המונה מגיע ל-CCR0, הוא מתאפס ומתחילה מחדש – וכן מתקבלת מחזריות קבועה.
- II. יחידת Comapre (למשל CCR1) – מוגדרת במצב (Reset/Set) או (Set/Reset):
 - בתחלת המבחן הפלט מוגדר ל-1.(High).
 - כשהמונה מגיע לערך CCR1, הפלט מתאפס ל-0.(Low).
 - התוצאה היא פולס ברוחב Ton, שניתו לשולט עליון על ידי שינוי הערך ב-CCR1.

אופן העבודה של כל טיימר

- ב מצב Up Mode קובע את מחזוראות הד'-PWM (20ms).
- יחידת Comapre (למשל CCR1) – קובע את זמן ההשניה שבו הפלט יורד ל-0 בתוך כל מחזור, ובכך שולט על זווית הסרבו.

3. אותן הבקרה של מנוע הסרבו מופק באופן רציף וקבוע, גם כאשר לא מטבחעת תנועה.
 - מנוע סרבו סטנדרטי (כגון SG90) זוקק לאות PWM רציף (מחזור של ms20) כדי לשמור על המיקום.
 - אם מפסיקים את אותן, המנוע עובר במצב "חופשי" (Free Run), ואז הזרוע יכולה לזרז כתוצאה מחיכוך, כבידה או רעדות.
 - לכן הבקר ממשיך לייצר את אותן PWM כל הזמן, כאשר רוחב הפולס (Ton) מייצג את הזווית הרצiosa:

1. אם אין דרישת לתנועה → הערך ב-CCR1 נשאר קבוע.
2. אם נדרשת תנועה → הבקר משנה את ערך CCR1, וכך משתנה רוחב הפולס ומהנו עז לזרזת החדשה.

• חישון מרחוק

1. גרעיניות המדיידה (Granularity):
- טיימר_A בבקר MSP430G2553 הוא טיימר בן 16 ביט.
 - כאשר מדירים אותו עם מקור שעון SMCLK = 1 MHz, מקבלת רזולוציה של 1 μs לכל טיקט של הטיימר.
 - קלומר, הגרעיניות של מדידה רוחב הפולס (Echo) היא מיקרו-שניה אחת.

הערך הגלמי המתkeletal:

- במצב Capture Input, הטימר לוד את ערך המונה ברגע קצה עולה וברגע קצה יורד של אוטה ה-Echo.
- ההפרש בין שני הערכים הוא t_{echo} – משך הזמן שבו האות היה פעיל.
- הערך הגלמי שמתkeletal הוא מספר צעדים של הטימר ביחידות של 1 μs.

קשר בין הערך למרחק:

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

כאשר N זה מספר הטייקים שנמדד, T_{tick}

- אות הד-Trigger שפעיל את החישון המרחק מופק על ידי יצירת פולס מרובע קצר באורך של כ-10 מיקרו-שניות על רג'ל הד-Trigger של החישון. הפולס הזה מופק בעוררת ייחוזת הטימר הפנימיות של הבקר.

הטימרים המעורבים ואופן העבודהם:

- טימר מסוג Timer_A מוגדר במצב Up Mode, כך שהמונה סופר מפעם עד לערך ב-CCR0.
- ערוץ השוואה CCR1 מוגדר במצב Set/Reset. בתחלת מחזור המניה הפלט עולה ל-High, וכשהמונה מגיע לערך CCR1 הפלט יורד ל-Low.
- בחירה של $T_{tick} = 10 \mu s$ (כאשר תדר השעון הוא 1MHz, כלומר כל טיק = $1 \mu s$) יוצרת פולס ברוחב $10 \mu s$.

חישון מרחק ממוקור אוֹר

- עשרים הדגימות לכיוול החישון ממוקור אוֹר נשמרות באזורי Information Memory של ה-Flash הקצתה הכתובת מתבצעת באופן הבא:

- חישון LDR שמאלי – הדגימות נשמרות ב-Address D, Segment 0x1000-0x103F, בטוויה הכתובות.
 - חישון LDR ימני – הדגימות נשמרות ב-Address C, Segment 0x1040-0x107F, בטוויה הכתובות.
- לכל חישון נשמרות עשר דגימות רצופות ניתן לשומר כל דגימה כ מילה של ששה עשר סיביות וכן להשאיר מספר בתים לסימון גרסה ובקרה של מילות.

2. עיקנון הפעולה:

- במהלך הכיוול נאספות 10 דגימות לכל החישון LDR (ערך ADC שנמדד עבור מרחק ידוע).
- הדגימות מייצרות זוגות נקודות (d,ADC) לאורך טווח המרחקים.
- כאשר נדרש ערך ביןים עבור מרחק שלא נמצא בפועל, מובוצעת אינטראפולציה 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 512 RAM בתים קטן), ולכן רק שולח את ערכיו הדגימה הגולמיים.
- המחשב, בעל יכולת חישוב גבוהה, מבצע את החישוב והאינטרפולציה כדי לבנות את גրף המרחק המלא.

שמירת הגרפ:

- הגרפ המלא שנבנה לאחר אינטראפולציה נשמר במחשב.
- בברק נשמרים אך ורק ערכיו הדגימות הגולמיים ב-**Information Memory** (10) דגימות לכל חישון).
- 3. חישון ה-DTR משנה את התנגדות בהתאם לעוצמת האור הפוגע בו. ככל שהמקור רחוק יותר, כמות האור המגיעה קטנה יותר וההתנגדות של ה-DTR משתנה בהתאם.
- קריית ערך אנלוגי – המתוח על ה-DTR מוחבר לכינסת ה-ADC של הבקר (ADC10). הבקר מבצע המורה אנלוגית-דיגיטל ומקבל ערך מספרי ADC בטווח 0–1023.
- שימוש בגרף הכיוול – במהלך נסמן 10 נקודות מדידה לכל חישון, המתארות את ההתאמות בין ערך ADC לבין מרחק פיזי ידוע.
- אינטראפולציה – כאשר מתקבל ערך ADC חדש, המחשב משווה אותו לנקודות הכיוול הקרובות ביותר וմבצע אינטראפולציה 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: ρ - המרחק הנמדד בס"מ

$ADC(d_{i+1}), ADC(d_i)$ – נקודות המרחק מהכיוול ובהתאם ADC שנמדדו בהתאם לערך הכיוול

• מבנה נתונים ואלגוריתמים

1. מבנה הנתונים לקליטת פריים בערוץ RX - הפריים שמתקבל בערוץ התקשרות כולל את המרכיבים הבאים:

- שדה Header – משמש כסימן פתיחה ומאפשר למקלט לזהות את תחילת המסגרת.
- שדה Length – מצין את גודל המטען בbytes 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 ומבודד שהמסגרת הסתיימה במקומות הנכונים.

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

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

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

- שדה Header – משמש כסימן פתיחה ומאפשר לצד המקלט להזות את תחילת המסגרת.
- שדה Length – מצין את גודל המטען בתים ומספק מידע על אורק הפריים.
- שדה Payload – מכיל את תוכן המידע עצמו, לדוגמה דגימות או פקודות.
- שדה Checksum – משמש לאיות שלמות המידע; במקרה של איזה ת妾המה הפריים נפסל.
- שדה Tail – משמש כסימן סיום ומבודא שהמסגרת הסתיימה בצורה תקינה.

4. גודל הפריים המשודר דרך ערוץ התקשרות TX הוא משתנה. הסיבה לכך היא שחלק מהשדות בפריים (Header, Length, Checksum, Tail) הם בעלי גודל קבוע מראש, אבל השדה המרכז – Payload – יכול להשתנות בהתאם לכמות המידע שיש להעביר. אם נדרשת שליחת של דגימה בודדת, המטען יהיה קצר, ואם יש צורך בשליחת של מספר דגימות או מقطع מסוים, המטען יהיה ארוך יותר. כדי שהמקלט ידע להתמודד עם שינוי זה, בכל פריים יש שדה Length שמצוין את אורק המטען המדויק, וכן ניתן להזות את סיום המסגרת גם כאשר גודל המטען משתנה.

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

אוף המימוש הצד הבקר (TX):

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

אוף המימוש הצד המחשב (RX):

- לאחר קבלת פריים, המחשב מחשב מחדש את סכום כל בתים במטען (מודולו 256).
- אם הערך המחשב תואם לערך שנשלח בשדה `Checksum`, הפריים נחשב תקין.
- במקרה של איזה ת妾המה, הפריים נפסל והמחשב יכול לבקש שליחת חזרה (Stop and Wait).

6. המערכת אכן תומכת בפרטוקול Stop-and-Wait Automatic Repeat Request, כך שניתנו להבטחה שהמידע המשודר הגיע באופן אמין.

אוף המימוש הצד הבקר (Transmitter):

- לאחר ייצירת פריים ושליחתו, הבקר נכנס למצב המתנה ל-ACK.
- מוגדר טיימר פנימי לזמן מקסימלי (Timeout).
- אם מתקבלת הודעה ACK תקינה מהמחשב => הבקר מתקדם לפריים הבא.
- אם מתקבלת הודעה NACK (כלומר זהותה שגיאה בפריים) או שהטיימר פג מבלי לקבל תגובה => הבקר שולח מחדש את אותו פריין.
- תהליך זה נמשך עד קבלת ACK תקין או עד שמספר ניסיונות החזרה יסתתיים.

אOPEN המימוש בצד המחשב (Receiver) :

- לאחר קבלת פריים דרך ערוץ RX, המחשב מבצע חישוב Checksum (מודולו 256).
- אם המידע תקין => נשלחת חזרה הודעת ACK לבקר.
- אם התגללה פגם בChecksum או במבנה המsgרת => נשלחת הודעת NACK לבקר.
- כך הבקר יודע אם לשמר על רצף השילחה או לחזור על פריים שנפגם.

מייפוי 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 לשילטה ברכיבי החומרה כמו UART, GPIO, ADC.
- וטיימרים.
- כתובות פריפריה – אוצר במרחב הזיכרון שמומופה ישירות לרכיבי חומרה פנימיים (Timer_A, ADC10, .).
- (USCI).

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

- מערך line באורך שיטים וארבעתוווים – buffer לקלט UART, נשמר ב-RAM.
- משתנים len ו-ticks – מונים פשוטים ב-RAM.
- משתנה cm מסווג uint16_t – תוצאה של מדידת מרחוק מהיישן אולטראסוני, נשמר ב-RAM.
- פונקציות עזר tx_putc, tx_write, tx_line – הקוד נשמר בזיכרון Flash.

קובץ api.c שכבת השירותים 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 ערכיהם מהיישני LDR ושומרת תוצאות זמניות ב-RAM.
- פונקציות סריקה (sweep) – מפעילות את שכבת HAL, אין מחזיקות במשתנים גלובליים.

קובץ hal.c שכבת 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 עצם – נשמרים ב-Flash Program Memory.
- בעת קריאה התוכן מועתק ל-RAM buffers זמניים.

קובץ bsp.c: שכבת BSP:

- שימוש בmacro מוגן define עבור מיפוי פינים – אין תופס זיכרון, מתרגם בזמן קומpile'ציה.
- פונקציות sysConfig_clock_init, uart_init וכו' – פועלות ישירות על רגיסטרים כמו .BCSCTL1, .UCA0CTL1

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

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

קובץ GUI.py - קובץ פיתון לצד המחשב האישי.

- מנהל משך גרפי ושולח פקודות דרך UART.
- כל המשתנים כאן מאוחסנים בזכרון המחשב, ללא השפעה על זיכרון הד-EEPROM.

مسקנות והצעות לשיפורים:

מסקנות:

- ניהול הזיכרון במערכת בוצע בצורה ייילה בהתחשב במוגבלות הבקר MSP430G2553.
- השימוש ב-DRAM מוקד במשתנים זמינים, buffers ומוני עוזר, תוך הקפדה על חסכון עקב מגבלת חמש מאות ושתים בית.
- זיכרון הד-Flash מנוצל לא רק לאחסון קוד אלא גם למימוש File Mode, מה שגדיל את הגמישות של המערכת.
- רגיסטרים (SFR) מנוצלים דרך שכבת ה-HAL ללא צורך במשתנים נוספים ב-DRAM.
- חלוקת השכבות (APP → HAL → BSP) מייעלת את ניהול הזיכרון ומאפשרת ניידות גבוהה של הקוד בין בקרים MSP430 שונים.
- עם זאת, המערכת אינה כוללת מגנון מובנה לניהול שנויות בזכרון או ב-DRAM, מה שעשו לגורום לאיבוד מידע במקרים של תקלת כתיבה לד-Flash.

הצעות לשיפורים:

- ניתן להוסיף מגנוני איתור ותיקון שגיאות (כגון CRC או checksum) עבור קבצים המאוחסנים ב-Flash, לשם שיפור אמינות File Mode.
- מומלץ לישם מצב Error State ב-FSM שיטפל במקרים של שגיאת קריאה/כתיבה או overflow ב-buffer.
- שימוש ב-DRAM circular buffer ל-TX/RX UART יכול להפחית עומס על RAM ולמנוע איבוד נתונים בתקשורת.
- ניתן לשקלן כיוזם משתנים (לדוגמה שימוש bitfields במבנה נתונים) כדי לחסוך RAM.
- הרחבת File Mode באמצעות ניהול טבלת קבצים דינמית או תמייה במספר גדול יותר של קבצים תגדיל את יכולת השימוש.
- בצד הד-GUI ניתן להוסיף חצוגות גרפיות (גרפים בזמן אמת) שיאפשרו ניתוח וניתוח טוב יותר של נתוני הד-MCU, מוביל להעמים על משאבי זיכרון בברק עצמו.

