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

אוניברסיטת בן-גוריון בנגב

בית הספר להנדסת חשמל ומחשבים

דו"ח מכין לפרוייקט מסכם

קורס "מבנה מחשבים ספרתיים"

361-1-4191

שם הפרוייקט

Light source and object proximity detector system

מערכת לגילוי מקורות אור וניטור אובייקטים במרחב

מגישים

ארקדי גרסימוק 312960891

עדי שלמה 211619226

תאריך הגשה

01.09.25

מטרת הפרוייקט

תכנון ומימוש של מערכת משובצת מחשב מבוססת מיקרו-בקר (MCU) מסוג MSP430G2553, שתפקידה המרכזי הוא גילוי מקורות אור וניטור של אובייקטים במרחב. המערכת מבצעת סריקה בגזרה של 180 מעלות באמצעות מנוע סרבו ומסתמכת על מספר חיישנים: מד מרחק אולטראסוני וחיישני אור. בנוסף, נדרשנו לאפשר שליחת קבצי סקריפט וטקסט דרך ממשק המשתמש אל הבקר ושמירתם בזכרון הFLASH, והרצה שלהם בעת הצורך.

הפרוייקט כולל פיתוח קוד צד בקר בשפת C, המבוסס על ארכיטקטורת מכונת מצבים (FSM) ושכבות אבסטרקציה, וכן פיתוח אפליקציית צד-מחשב עם ממשק משתמש גרפי (GUI) בשפת פייתון. התקשורת בין הבקר למחשב מבוססת על פרוטוקול UART תקשורת טורית אסינכרונית, אשר ממומשת בתקן RS-232.

ממשק משתמש מסך ראשי:

תיאור רכיבי חומרת הקצה

- חיישן אולטראסוני HCSR04 – שימש אותנו כמד מרחק, למדידת המרחק מהאובייקטים בטווח של 2 ס"מ ועד 4.5 מטר.
- חיישני אור LDR – שני נגדים תלויי-אור המשמשים לגילוי מקורות אור וכיולם מאפשר להעריך את מרחקם.
- מנוע סרבו מדגם SG90 – משמש להזזת מערך החיישנים בזווית מבוקרת וביצוע סריקה של 180 מעלות.

- צג LCD – משמש להצגת מידע למשתמש בצד הבקר, משמש בעיקר במצב הקבצים, להצגה של שמות הקבצים והתוכן שלהם.
- לחצנים – משמשים לאינטראקציה עם המשתמש בצד הבקר, למשל לצורך כיול חיישני הLDR ודפדוף בין קבצי סקריפט וטקסט, לצורך בחירתם והצגתם מזכרון הFLASH.

אפיון המערכת

גילוי אובייקטים במרחב:

במצב זה מתבצעת סריקה מרחבית של $180°$ באמצעות מנוע הסרבו ובכל מעלה, דוגמים בעזרת חיישן האולטראסוניק את המרחב ולפי הגל שמתקבל חזרה מהאובייקט הכי קרוב שנמצא מול החיישן בכל דגימה, מחשבים את המרחק בין ציר סיבוב המנוע לבין האובייקט הנתון.

כדי לחשב את זמן הסריקה הכולל, ננתח את הפעולות שמבוצעות בכל צעד של מעלה, בפונקציה servo_scan.

עבור כל מעלה בסריקה, המערכת מבצעת את הפעולות הבאות:

- **הזזת המנוע:** השהיה של 250ms עבור התחלת התנועה לצורך התייצבות ראשונית במעלה הראשונה.
- **התייצבות המנוע:** השהיה נוספת של 50ms, כדי להבטיח הגעה לזווית הבאה.
- **מדידת מרחק:** משתמשים בפונקציה measure_distance_averaged, שמבצעת 3 מדידות עוקבות כאשר בין כל מדידה יש השהייה של 30ms בנוסף לזמן המדידה עצמו, סך הזמן הוא כ90ms.

הביטוי המתמטי שמתאים לזמן הסריקה הכולל:

נגדיר את זמן הסריקה הכולל כT_{scan} שיהווה סכום של הצעד הראשון ועוד 180 צעדים נוספים, כי סורקים מ0 עד 180 כולל – סה"כ 181 מדידות.

הנוסחאות יראו כך:

$$T_{step} = T_{move} + T_{settle} + T_{measure}$$

$$T_{initial-step} = 250ms + 50ms + 90ms = 390ms$$

$$T_{other-step} = 50ms + 90ms = 140ms$$

$$T_{scan} = 390ms + 180 * 140ms = 25590ms \approx 25.59s$$

כמובן שגם לביצוע הפקודות עצמן גם יש משקל בהשהיה מבחינת מחזורי השעון כפול מספר הפקודות, מה שמוסיף זמן לכל מחזור סריקה, לפי החישוב שביצענו מעלה קיבלנו מחזור סריקה של כ25.6 שניות, שזה

הזמן ללא התחשבות לזמן ביצוע הפקודות שכנראה מוסיף לנו מעט זמן, אבל עדיין עומדים בדרישות של החצי דקה לסריקה.

ולכן נקבל כי התדר הוא: $f = \frac{1}{T_{scan}} = \frac{1}{25.59} = 0.039Hz$

כל דגימה שנשלחת מהבקר למחשב במצב זיהוי אובייקטים מורכבת בסה"כ מ-3 בתים.

המידע שכל דגימה מייצגת:

- **בתים 1-2:** מייצגים את מרחק האובייקט, זהו ערך של 16 סיביות(int), המייצג את המרחק הממוצע בס"מ שחושב על ידי הבקר. הבית הראשון הוא הLSB והשני הוא הMSB.
- **בית 3:** מייצג את הזווית, ערך של 8 סיביות(char), שמתאים בדיוק כי הזווית היא ערך בין 0 ל180 ולכן הוא מדויק בגודלו. נותן לנו את הזווית הנוכחית שבה נלקחה הדגימה.

חלוקת העבודה בין הMCU לPC מבוססת על שיקול הנדסי של משאבים מוגבלים. במהלך הפרוייקט נתקלנו באתגר המרכזי בעבודה עם הMCU שהוא זכרון הRAM שלו, שהוא קטן מאוד וגודלו הוא בסה"כ 512 בתים, ויכולת העיבוד שלו מוגבלת ביחס למחשב שלו משאבים כמעט בלתי מוגבלים.

עיבוד בצד הMCU:

- **איסוף וסינון ראשוני:** הבקר מנהל את החומרה ברמה הנמוכה – מייצר אותות PWM לסרבו וTrig לחיישן האולטראסוני, מודד את פולס הEcho באמצעות טיימר, ומבצע סינון רעשים בסיסי על ידי מיצוע של 3 מדידות מרחק, עבור כל זווית עם 30ms בין הדגימות ודואג לtimeout: מתקבלות רק דגימות שנמצאות בתחום בין 0 ל30000 מיקרו-שניות, המיצוע מחושב רק מעל דגימות מוצלחות.
- **חישוב ראשוני:** מכיוון שלבקר זה אין יחידת FPU(Floating point unit), ופעולת חילוק במספר נקודה-צפה היא פעולה מאוד איטית ויקרה עבור הבקר, בחרנו לבצע אופטימיזציה ולעשות את הפעולה המתמטית באמצעות כפל וBit shift שהן פעולות מהירות בהרבה. הבקר ממיר את זמן הפולס הגולמי למרחק בס"מ באמצעות החישוב היעיל שמתחשב במהירות הקול:

$$((unsigned\ long)echo\_pulse\_time\_us * 1130) \gg 16$$

- **תפקיד:** מטרת הבקר, לאסוף ולהכין נתונים גולמיים אמינים ועם כמה שפחות רעשים לצד המחשב שיהיה נוח לניתוח וחישוב.

עיבוד בצד המחשב:

- **תצוגה גרפית:** המחשב מתפקד כממשק משתמש ומציג את התוצאות בצורה ויזואלית על גרפים ונתונים מספריים.
- **ניתוח נתונים מורכב:** מקבל את כל 180 הדגימות הנקיות, מריץ את אלגוריתם זיהוי האובייקט על כל המידע, תוך זיהוי קצוות אובייקטים וחישובים סופיים לדיוק.

- **חישובים מתקדמים:** אחראי לחשב את כל הפרמטרים הנדרשים במצב זה עבור כל אובייקט, מרחק ממוצע, זווית מרכזית ורוחב אובייקט באמצעות טריגונומטריה המסומנים כך לפי הגדרת הפרוייקט:

$$\rho - \text{מרחק}$$

$$\varphi - \text{זווית}$$

$$l - \text{רוחב}$$

נצילות התקשורת:

נצילות התקשורת מחושבת כיחס בין סיביות המידע השימושי לבין סך כל הסיביות המשודרות על הקו הפיזי. המידע השימושי הוא בעצם 24 ביטים, 16 ביטים עבור מרחק ו8 ביטים עבור זווית ולכן המידע שמועבר הוא 24 ביטים בData layer. בPhysical Layer, כל בית נשלח בUART עם start bit וstop bit, סה"כ 10 ביטים לכל בית. לכן, סך הביטים על הקו הוא 30 ביטים.

נקבל את הנוסחה הבאה:

$$\eta = \frac{24}{30} = 0.8 * 100\% = 80\%$$

נצילות המידע הנשלח בערוץ התקשורת לכל אחת מהדגימות הוא 80 אחוז.

אלגוריתם לזיהוי אובייקטים תמציתי בPseudo-Code:

```
FUNCTION detect_objects(samples[180]):
// הגדרת משתנים
Object_list = EMPTY_LIST
Current_samples_list = EMPTY_LIST
Detected = FALSE
start_angle = 0
// מרחק מקסימלי
THRESHOLD = 400
// לולאת הדגימות
for i from 0 to 180:
->distance = samples[i].distance
->angle = samples[i].angle
// בדיקת זיהוי חדש
->if distance < THRESHOLD && detected == FALSE:
→detected = TRUE
```

→ start_angle = angle

→ Current_samples_list.append(samples[i])

// בדיקה האם אנחנו באותו אובייקט

->else if distance< THRESHOLD && detected == TRUE:

→ Current_samples_list.append(samples[i])

// בדיקה האם הגענו לסוף הזיהוי

->else if distance>=THRESHOLD && detected == TRUE:

→detected = FALSE

// חישובי זווית, מרחק ורוחב

→End_angle = samples[i-1].angle

→phi = (start_angle + end_angle)/2

→delta_radians = RAD(end_angle - start_angle)

→rho = avg_dist(current_samples_list)

→width = 2 * rho * sin(delta_radians/2)

// הוספה לרשימת האובייקטים והחזרה של הרשימה

→object_list.append(phi, rho,width)

→ Current_samples_list = EMPTY_LIST

->End if

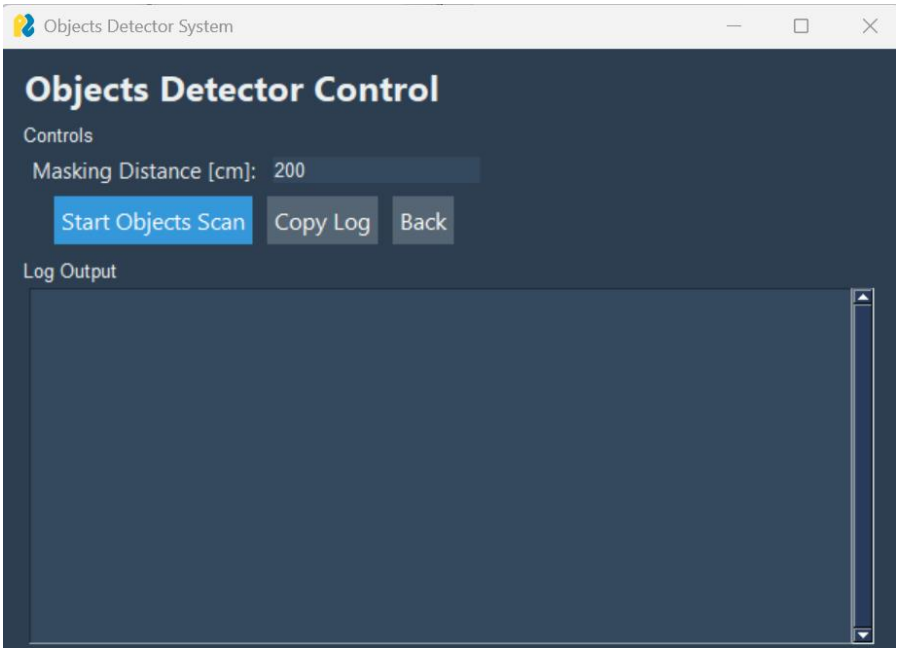
End for

Return object_list

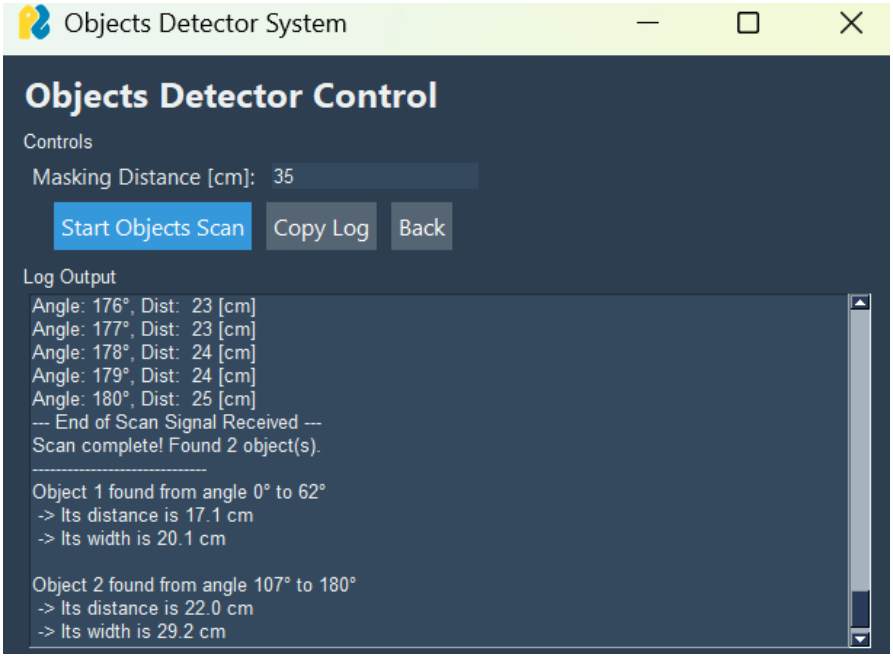
End FUNCTION detect_objects

ממשק משתמש:

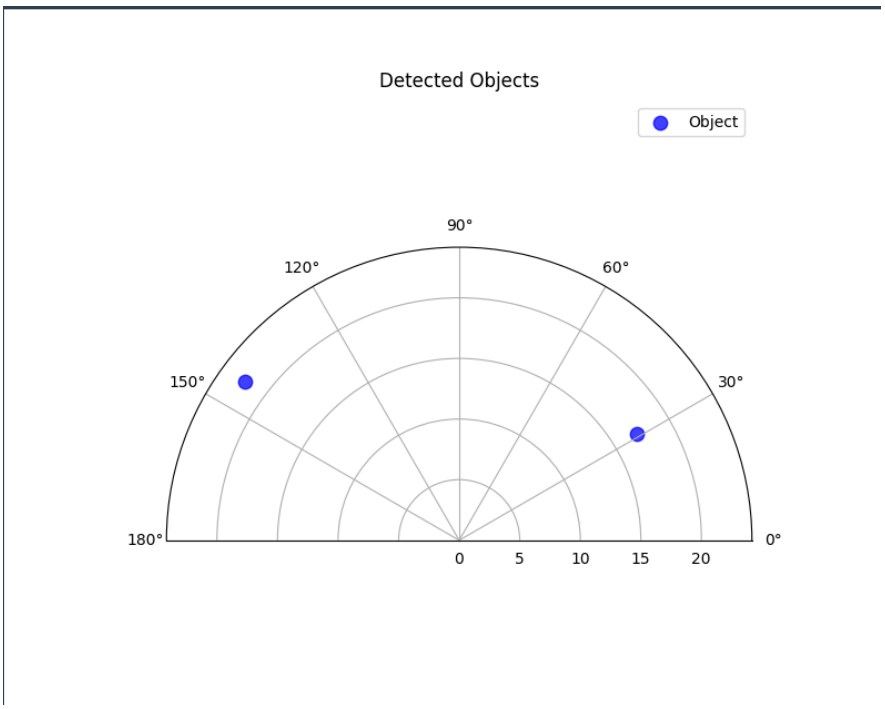
מסך כניסה למצב:



מסך בזמן הרצה:



הצגה גרפית:



גילוי מקורות אור במרחב

במצב זה מתבצעת סריקה מרחבית של 180° באמצעות מנוע הסרבו ובכל מעלה, דוגמים בעזרת חיישני הLDR, נגדים הרגישים לאור את המרחב ומודדים את מרחק מקור האור הנמצא מולו ברגע נתון.

לפני תחילת עבודה במצב זה, נדרש כיול של חיישני הLDR עבור מקורות האור הנמצאים במרחב הסריקה לצורך זיהוי אופטימלי וחישוב מדויק של הזווית והמרחק שבו מקורות האור נמצאים בהם.

את הכיול מבצעים לכל חיישן בנפרד, כאשר מתחילים מ5 ס"מ ומתקדמים כל פעם בקפיצות של 5 ס"מ, תוך לחיצה על כפתור PB0 עד כיול סופי ב50 ס"מ שהוא המרחק המקסימלי עבורם לזיהוי מקור אור. את המרחק הרלוונטי בכל שלב בכיול, ניתן לראות במסך הLCD שמציג באיזה מרחק לכייל בכל רגע נתון.

כדי לחשב את זמן הסריקה הכולל, ננתח את הפעולות שמבוצעות בכל צעד של מעלה, בפונקציה servo_scan.

עבור כל מעלה בסריקה, המערכת מבצעת את הפעולות הבאות:

- **הזזת המנוע:** השהיה של 250ms עבור התחלת התנועה לצורך התייצבות ראשונית במעלה הראשונה.
- **התייצבות המנוע:** השהיה נוספת של 50ms, כדי להבטיח הגעה לזווית הבאה.
- **השהייה לפני מדידת אור:** השהיה נוספת לפני מדידת האור של 50ms.
- **מדידת אור:** הפונקציה מבצעת שתי קריאות של ADC, וביניהן יש השהייה של 30ms, הזמן הכולל למדידה הוא כ30ms, כי זמן המרת הADC הוא זניח.

הביטוי המתמטי שמתאים לזמן הסריקה הכולל:

נגדיר את זמן הסריקה הכולל בT_{scan} שיהווה סכום של הצעד הראשון ועוד 180 צעדים נוספים, כי סורקים מ0 עד 180 כולל – סה"כ 181 מדידות.

הנוסחאות יראו כך:

$$T_{step} = T_{move} + T_{settle} + T_{pre\ light} + T_{light}$$

$$T_{initial-step} = 250ms + 50ms + 50ms + 30ms = 380ms$$

$$T_{other-step} = 50ms + 50ms + 30ms = 130ms$$

$$T_{scan} = 380ms + 180 * 130ms = 23780ms \approx 23.78s$$

כמובן שגם לביצוע הפקודות עצמן גם יש משקל בהשהיה מבחינת מחזורי השעון כפול מספר הפקודות, מה שמוסיף זמן לכל מחזור סריקה, לפי החישוב שביצענו מעלה קיבלנו מחזור סריקה של כ23.8 שניות, שזה הזמן ללא התחשבות לזמן ביצוע הפקודות שכנראה מוסיף לנו מעט זמן, אבל עדיין עומדים בדרישות של החצי דקה לסריקה.

ולכן נקבל כי התדר הוא: $f = \frac{1}{T_{scan}} = \frac{1}{23.78} = 0.042Hz$

כל דגימה שנשלחת מצד הבקר לצד המחשב במצב של זיהוי אור, מורכבת מ5 בתים.

המידע שכל דגימה מייצגת:

- **בתים 1-2:** ערך ADC של 16 ביטים(int) מחיישן LDR1.
- **בתים 3-4**: ערך ADC של 16 ביטים(int) מחיישן LDR2.
- **בית 5:** הזווית, 8 ביטים(char) של מנוע הסרבו בעת המדידה.

חלוקת עיבוד המידע בין הMCU לבין הPC מבוססת על ניצול מיטבי של משאבי המערכת – הבקר מבצע את המשימות התלויות בזמן-אמת ברמה הנמוכה ואילו המחשב, מבצע את הניתוח המורכב הדורש זכרון וכוח עיבוד גדולים יותר.

עיבוד בצד הMCU:

- **תפקיד:** הבקר משמש כיחידת איסוף נתונים יעילה, שמעבירה את המידע המתקבל מהADC והחיישני אור הלאה.
- **איסוף נתונים:** הבקר שולט בתנועת מנוע הסרבו, מפעיל את יחידת הADC בזוויות המתאימות, וקורא את ערכי הADC הגולמיים(10 bit – מה שקיים בבקר הערכה הביתית), משני חיישני הLDR.
- **אריזת המידע:** הבקר לוקח את שני ערכי הADC ואת הזווית, אורז אותם לפריים מידע בגודל 5 בתים ושולח לצד המחשב.

עיבוד בצד המחשב:

- **שימוש בנתוני הכיול:** המחשב מקבל את ערכי הADC הגולמיים המגיעים מהבקר, משתמש בטבלאות הכיול(Lookup tables), שנשמרות מראש כדי להמיר כל ערך ADC למרחק בסנטימטרים.
- **אלגוריתם זיהוי מקור אור:** המחשב מריץ את האלגוריתם לזיהוי מקורות אור, שסורק את 180 הדגימות, מחפש "פסגות" של עוצמת אור, כלומר ערכי ADC נמוכים ומשתמש בנתונים משני החיישנים כדי לקבוע את מיקומו המדויק של האור, מבחינת זווית ומרחק.
- **חישוב:** חישוב סופי של הערכים הסופיים של הזווית והמרחק והצגת בצורה גרפית למשתמש.

נצילות התקשורת:

נצילות התקשורת מחושבת כיחס בין סיביות המידע השימושי לבין סך כל הסיביות המשודרות על הקו הפיזי. המידע השימושי הוא בעצם 28 ביטים, כל חיישן LDR, הוא ערך ADC ברזולוציה של 10 ביטים ולכן עבור כל דגימה נקבל 10 ביטים, יש לנו שני חיישנים ולכן 20 ביטים רק מדגימות אור ו8 ביטים עבור זווית ולכן המידע שמועבר הוא 5 בתים כפול 8 ביטים – סה"כ 40 ביטים בData layer. בPhysical Layer, כל בית נשלח בUART עם start bit וstop bit, סה"כ 10 ביטים לכל בית. לכן, סך הביטים על הקו הוא 50 ביטים.

נקבל את הנוסחה הבאה:

$$\eta = \frac{28}{50} = 0.56 * 100\% = 56\%$$

נצילות המידע הנשלח בערוץ התקשורת לכל אחת מהדגימות הוא 56 אחוז, שהוא אחוז נמוך לנצילות אבל זה קורה כי כל ערך של 10 ביטים(של כל חיישן) נשלח בתוך 2 בתים(16 ביטים).

אלגוריתם זיהוי מקורות אור Pseudo-code:

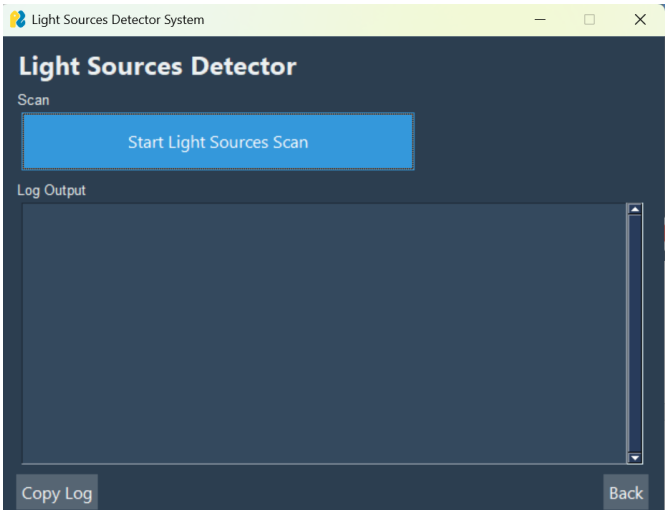
האלגוריתם מניח כי קיימות שתי טבלאות כיול הממפות את ערכי הADC למרחק.

```
FUNCTION light_sources(samples[180], cal_t1, cal t2):
// הגדרת משתנים
source_list = EMPTY_LIST
detected = FALSE
angle = 0
// מרחק זיהוי מקסימלי
THRESHOLD = GET_DARK_VAL(cal_t1)
// לולאת הדגימות
for i from 0 to 179:
```

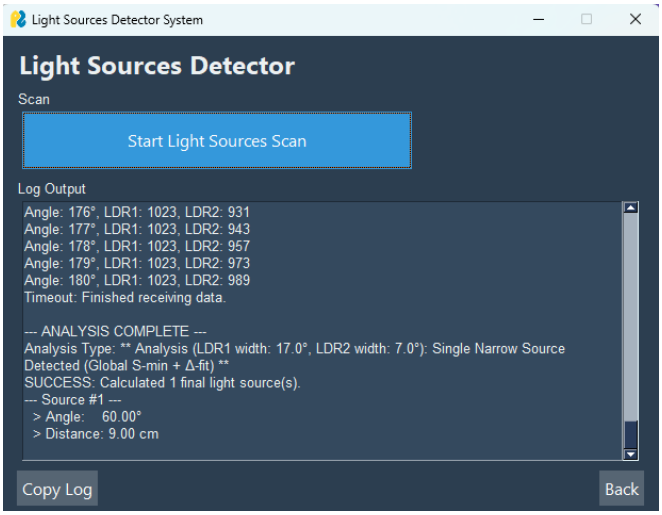
```
->ldr1_val = samples[i].ldr1_val
->ldr2_val = samples[i].ldr2_val
->angle = samples[i].angle
// בדיקת זיהוי מקור אור פוטנציאלי
->there_is_light = ldr1_val < THRESHOLD || ldr2_val < THRESHOLD
// התחלה של זיהוי מקור אור
->if there_is_light && detected == FALSE:
→detected = TRUE
→samples_list = EMPTY_LIST
// איסוף דגימות אור
->if detected == TRUE:
→samples_list.append(sample)
// סוף זיהוי
->if (!there_is_light || i== 179 ) && detected == TRUE
→detected = FALSE
// מציאת זווית מדויקת ועוצמת הארה מקסימלית
→peak_sample = MIN(samples_list)
→angle = peak_sample.angle
→min_adc = MIN(peak_sample.ldr1_val, best_sample.ldr2_val)
→rho = LOOKUP_TABLE(min_adc, cal_t1, cal_t2)
// הוספת מקור האור לרשימה
→ source_list.append(angle, rho)
->End if
End for
Return source_list
End FUNCTION light_sources
```

ממשק משתמש:

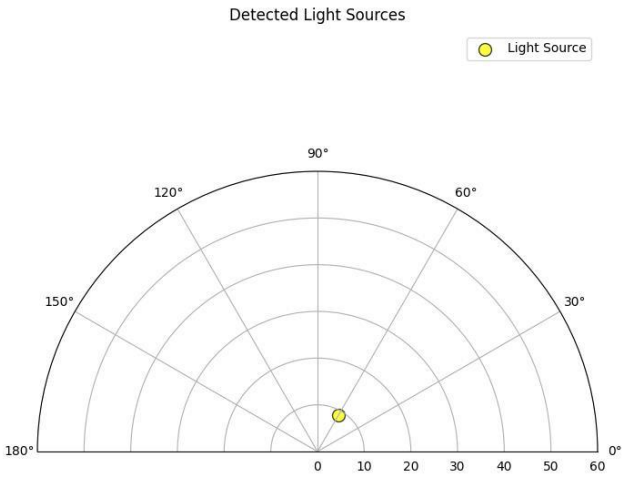
מסך כניסה למצב:



מסך בזמן הרצה:



הצגה גרפית:



גילוי אובייקטים ומקורות אור במרחב(סעיף בונוס)

במצב הסריקה המשולב (בונוס) מאחד את יכולות המערכת ומאפשר זיהוי סימולטני של אובייקטים פיזיים ומקורות אור בסריקה אחת של 180 מעלות. בכל צעד זוויתי, הבקר דוגם את כלל החיישנים – מבצע מדידת מרחק ממוצעת באמצעות החיישן האולטרסוני וקורא את ערכי הADC מחיישני הLDR. המידע המלא נארז ונשלח למחשב בפריים נתונים מורחב בגודל 7 בתים, המכיל את המרחק, שני ערכי האור, והזווית. צד המחשב אחראי לקבל את הפריים המורחב, להפעיל במקביל את אלגוריתם זיהוי האובייקטים על נתוני המרחק ואת אלגוריתם זיהוי מקורות האור על נתוני הLDR, ולהציג תמונה מאוחדת של כלל הגילויים על גבי התצוגה הגרפית.

עבור כל מעלה בסריקה, המערכת מבצעת את הפעולות הבאות:

- **הזזת המנוע:** השהיה של 250ms עבור התחלת התנועה לצורך התייצבות ראשונית במעלה הראשונה ולאחר מכן, השהיה של 25ms לכל מעלה.
- **התייצבות המנוע:** השהיה נוספת של 50ms לאחר עצירת הPWM, כדי להבטיח עצירה מוחלטת של המנוע לאחר הצעד הנתון.
- **מדידת מרחק:** משתמשים בפונקציה measure_distance_averaged, שמבצעת 3 מדידות עוקבות כאשר בין כל מדידה יש השהייה של 30ms בנוסף לזמן המדידה עצמו, סך הזמן הוא כ90ms.
- **מדידת אור:** הפונקציה מבצעת שתי קריאות של ADC, וביניהן יש השהייה של 30ms, הזמן הכולל למדידה הוא כ30ms, כי זמן המרת הADC הוא זניח.

הביטוי המתמטי שמתאים לזמן הסריקה הכולל:

נגדיר את זמן הסריקה הכולל כT_{scan} שיהווה סכום של הצעד הראשון ועוד 180 צעדים נוספים, כי סורקים מ0 עד 180 כולל – סה"כ 181 מדידות.

הנוסחאות יראו כך:

$$T_{step} = T_{move} + T_{settle} + T_{distance} + T_{light}$$

$$T_{initial-step} = 250ms + 50ms + 90ms + 30ms = 420ms$$

$$T_{other-step} = 50ms + 90ms + 30ms = 170ms$$

$$T_{scan} = 420ms + 180 * 170ms = 31020ms \approx 31.02s$$

כמובן שגם במצב זה, הפקודות עצמן מוסיפות משקל בהשהיה מבחינת מחזורי השעון כפול מספר הפקודות. השהייה זו, מוסיפה זמן לכל מחזור סריקה, לפי החישוב שביצענו מעלה קיבלנו מחזור סריקה של

כ31 שניות, שזה הזמן ללא התחשבות לזמן ביצוע הפקודות שכנראה מוסיף לנו מעט זמן. בשילוב שני המצבים הגיוני שזמן הסריקה יהיה מעט ארוך יותר אך עדיין מתבצע בזמן מינימלי ומהיר.

ולכן נקבל כי התדר הוא: $f = \frac{1}{T_{scan}} = \frac{1}{31.02} = 0.032Hz$

כל דגימה שנשלחת מצד הבקר לצד המחשב במצב המשולב, מורכבת מ7 בתים והיא הגדולה מבין המצבים כי משלבת את שניהם.

המידע שכל דגימה מייצגת:

- **בתים 1-2:** מייצגים את מרחק האובייקט, זהו ערך של 16 סיביות(int), המייצג את המרחק הממוצע בס"מ שחושב על ידי הבקר. הבית הראשון הוא הLSB והשני הוא הMSB.
- **בתים 3-4**: ערך ADC של 16 ביטים(int) מחיישן LDR1.
- **בתים 5-6**: ערך ADC של 16 ביטים(int) מחיישן LDR2.
- **בית 7:** הזווית, 8 ביטים(char) של מנוע הסרבו בעת המדידה.

העקרון ההנדסי נשאר זהה: חלוקת עיבוד המידע בין הMCU לבין הPC מבוססת על ניצול מיטבי של משאבי המערכת – הבקר מבצע את המשימות התלויות בזמן-אמת ברמה הנמוכה ואילו המחשב, מבצע את הניתוח המורכב הדורש זכרון וכוח עיבוד גדולים יותר.

עיבוד בצד הMCU:

בכל צעד זוויתי, הבקר מבצע את כל המדידות הנדרשות (מרחק ואור), אורז את כל שבעת בתי המידע לפריים אחד ושולח אותו למחשב.

עיבוד בצד המחשב:

המחשב מקבל את הפריים המורחב, מפרק אותו לנתוני מרחק ונתוני אור, ומריץ את שני האלגוריתמים (זיהוי אובייקטים וזיהוי מקורות אור) על אותו סט נתונים. לבסוף, הוא מציג את כל התוצאות יחד על אותה תצוגה גרפית.

נצילות התקשורת:

נצילות התקשורת מחושבת כיחס בין סיביות המידע השימושי לבין סך כל הסיביות המשודרות על הקו הפיזי. המידע השימושי הוא בעצם 44 ביטים, כל חיישן LDR, הוא ערך ADC ברזולוציה של 10 ביטים, המרחק מהחיישן האולטראסוני הוא 16 ביטים ו8 ביטים עבור זווית ולכן המידע שמועבר הוא סה"כ 44 ביטים בData layer. בPhysical Layer, כל בית נשלח בUART עם start bit וstop bit, סה"כ 10 ביטים לכל בית. לכן, סך הביטים על הקו הוא 70 ביטים.

נקבל את הנוסחה הבאה:

$$\eta = \frac{44}{70} = 0.628 * 100\% = 62.8\%$$

נצילות המידע הנשלח בערוץ התקשורת לכל אחת מהדגימות הוא 63 אחוז בקירוב, שהוא אחוז נמוך לנצילות אבל זה קורה כי כל ערך של 10 ביטים(של כל חיישן) נשלח בתוך 2 בתים(16 ביטים).

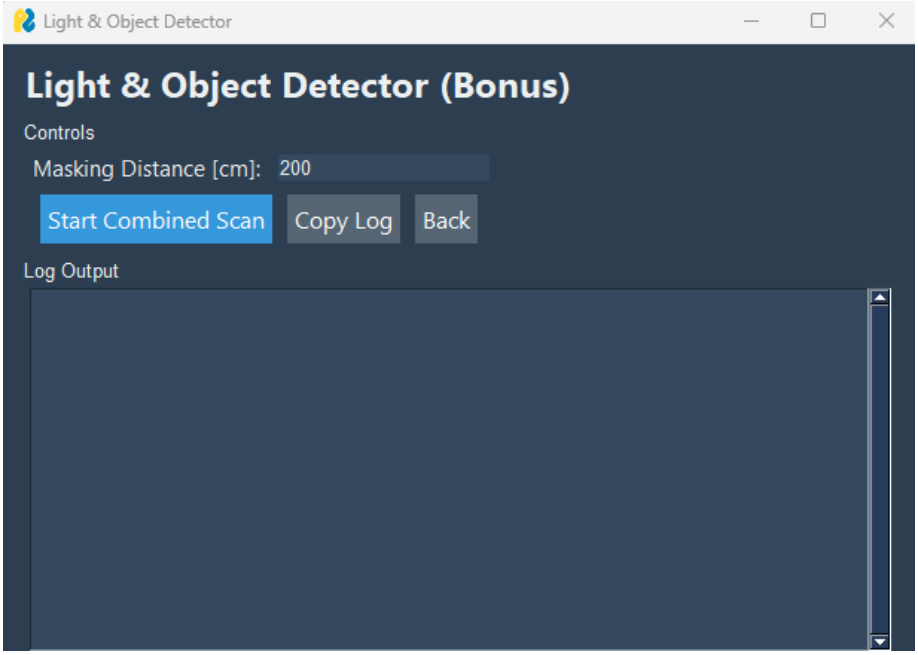
אלגוריתם זיהוי משולב:

האלגוריתם הוא שילוב של שני האלגוריתמים הקודמים:

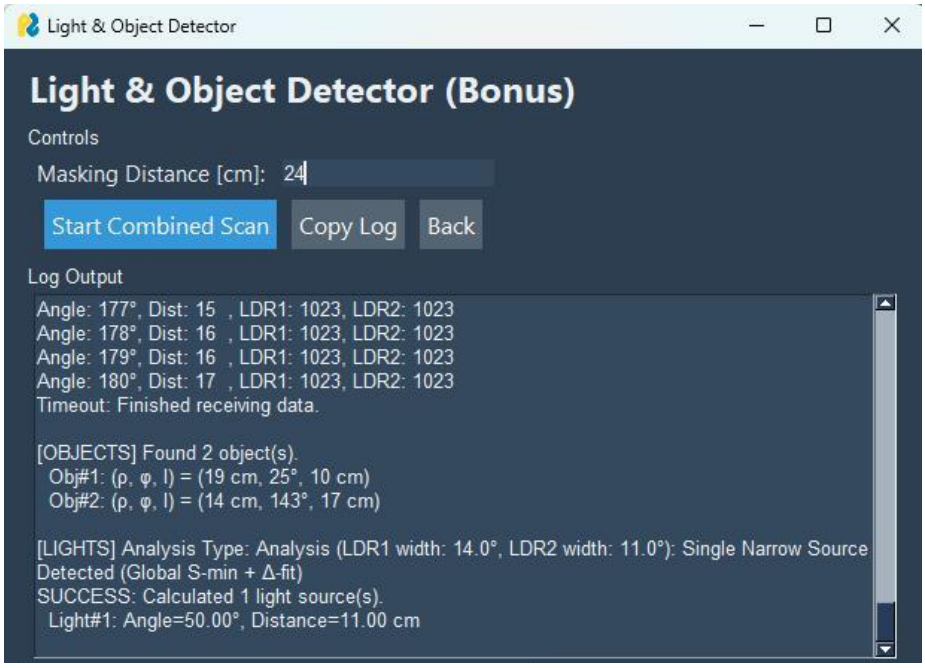
1. לאחר קבלת כל 180 הדגימות, התוכנה מריצה את אלגוריתם זיהוי האובייקטים על רכיבי המרחק והזווית של הנתונים.
2. במקביל, מריצה התוכנה את אלגוריתם זיהוי מקורות האור על רכיבי הLDR והזווית, תוך שימוש בטבלאות הכיול.
3. שתי רשימות התוצאות מאוחדות ומוצגות יחד על התצוגה הגרפית הסופית.

ממשק משתמש:

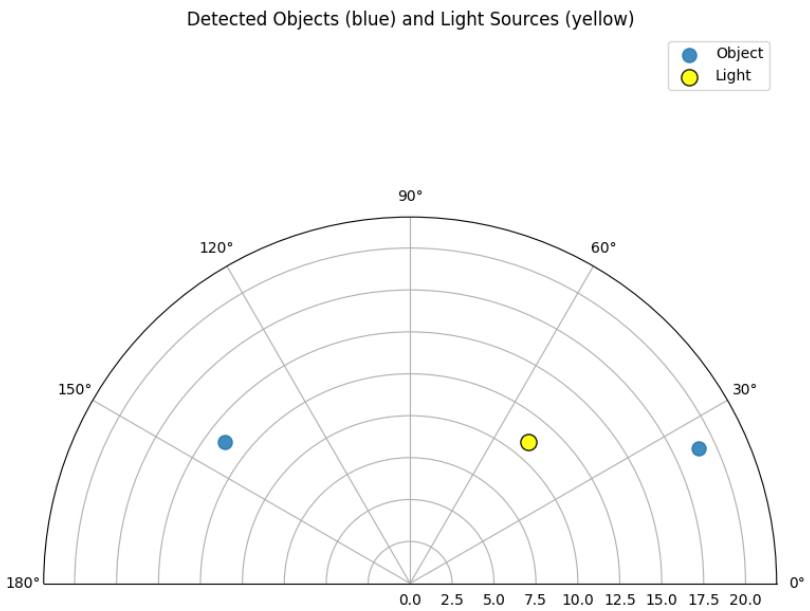
מסך כניסה למצב:



מסך בזמן הרצה:



תצוגה גרפית:



מד מרחק – טלמטר

במצב זה המנוע שם את המערכת בזווית לשנבחרה על ידי המשתמש, מציג באופן רציף על גבי הממשק את המרחק של האובייקט מהחיישן האולטראסוני באותה זווית לאחר הגעה אליה, עד למעבר למצב אחר.

החלטת ערך המרחק וסינון רעשים:

ההחלטה על המרחק הסופי מתבססת על שתי שיטות השולבות יחד בפונקציה measure_distance_averaged:

- **מיצוע –** זוהי השיטה העיקרית לסינון רעשים. במקום להסתמך על מדידה בודדת שעלולה להיות שגויה או רועשת, המערכת מבצעת 3 מדידות עוקבות והערך הסופי שמוחזר הוא הממוצע של שלושת דגימות אלו, מה שמייצב את הקריאה ומפחית את ההשפעה של חריגות אקראיות.
- **סינון –** לפני שכל מדידה נכנסת לחישוב הממוצע, הקוד בודק האם היא תקינה, מוודא שרוחב פולס הEcho נמצא בטווח הגיוני ובכך מסנן דגימות לא תקינות. מדידה עם זמן פולס 0, שמעידה על timeout או זמן ארוך מדי, שמעיד על מרחק גדול מטווח העבודה של החיישן, נזרקת ולא משתתפת בחישוב הממוצע.

תדר הדגימה:

במצב מד מרחק, תדר הדגימה הוא כ - 1.69Hz, הוא נקבע בלוגיקה במצב tele_action בFSM של מצב הTelemeter. בתחילת כל מחזור של המצב, ישנה השהייה קבועה של 500ms, וכל מדידה מבצעת מיצוע של 3 מדידות עוקבות, כך שיש עוד תוספת של 90ms למדידה. ורק אחריה מתבצעת מדידת המרחק הבאה.

לכן, המערכת מבצעת דגימה אחת כל חצי שניה, זמן המחזור מחושב כך:

$$T_{delay} = \frac{1}{f} \rightarrow f = \frac{1}{0.59[s]} \approx 1.69\ Hz$$

תדר זה נבחר כשיקול הנדסי: הוא מהיר מספיק כדי להציג תחושה של עדכון דינמי ורציף למשתמש, אך איטי כדי שהערכים במסך לא יתחלפו מהר מדי ויהיו גם קריאים עבורו.

מרחק פיזי:

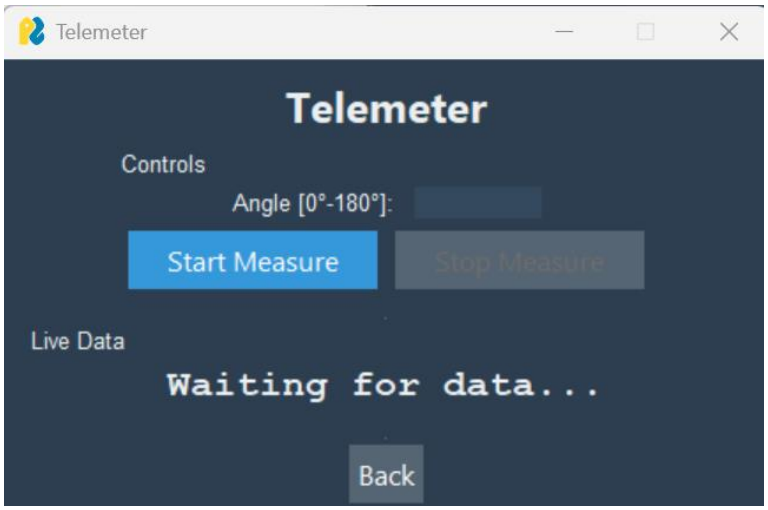
הMCU, והמערכת כולה, לא מקבלת פקודה אקטיבית על שינוי המרחק.

הארכיטקטורה עובדת במצב רציף בלי שום משהו מתוחכם – הבקר נכנס ללולאה קבועה במצב של המד מרחק, כל מחזור יש השהייה של 500ms, שבסופה הוא מודד מרחק ובאופן מיידי שולח למחשב את התוצאה, בין אם היא זהה או שונה מהקודמת, אין השוואה או שמירה של המדידה הקודמת בצד הבקר.

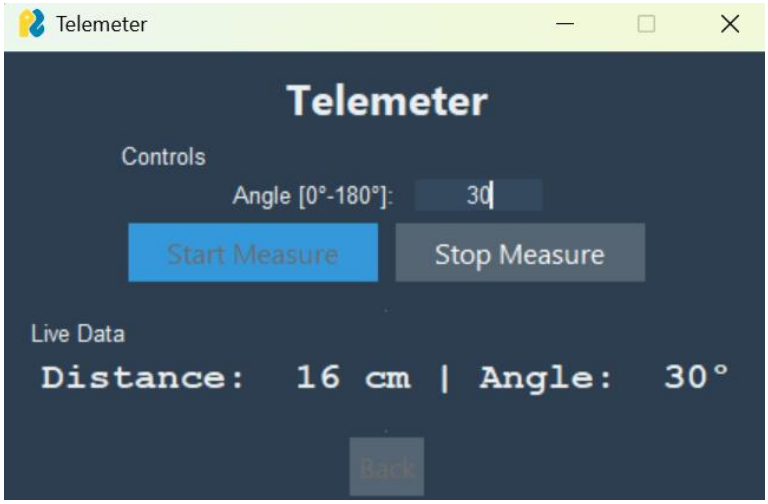
ההחלטה של השינוי, מתבצעת כולה בצד המשתמש, הוא רואה בממשק משתמש את הערך המתעדכן באופן דינמי בצג הGUI שבמחשב וכך הוא מבין את המרחק הפיזי של האובייקט, הבקר רק מספק זרם נתונים רציף בזמן אמת.

ממשק משתמש:

מסך כניסה למצב:



מסך בזמן הרצה:



מערכת קבצים

מצב זה מממש יכולת אחסון וניהול של קבצים על גבי זכרון הFlash הבלתי נדיף של המיקרו-בקר.
המערכת תומכת באחסון דינמי של עד 10 קבצים בגודל משתנה, בתוך מרחב אחסון ייעודי של 2KB.
הקבצים שבהם המערכת תומכת מחולקים לשני סוגים: קובצי טקסט, המיועדים לקריאה על גבי צג הLCD המובנה, וקובצי סקריפט, המכילים פקודות high-level, להפעלת פונקציות המערכת באופן אוטומטי.
הארכיטקטורה מבוססת על מבנה נתונים(File Entry), השמור באזור נפרד בזכרון הFlash, אשר מנהל את המטא-דאטה של כל קובץ – כולל שמו, סוגו, גודלו וכתובת ההתחלה שלו באזור התוכן. ניהול דינמי של הזכרון מתאפשר באמצעות מעקב אחר מספר הקבצים והכתובת הפנויה הבאה, מה שמאפשר הוספת קבצים באופן רציף. המשתמש יכול להעלות קבצים מהמחשב, לדפדף ברשימת הקבצים ולקרוא את תוכנם באמצעות הלחצנים הפיזיים, ולהפעיל את הסקריפטים מרחוק דרך ממשק הGUI.

בדומה לשאר חלקי המערכת, גם כאן חלקות העבודה בין הMCU לצד המחשב, מבוססת על השיקול ההנדסי של משאבים מוגבלים. במהלך הפרוייקט, במיוחד במצב זה של המערכת, האתגר המרכזי בעבודה עם הMCU היה זכרון הRAM הקטן, שגודלו 512 בתים בלבד, ויכולת העיבוד המוגבלת שלו ביחס למחשב, שלו משאבים בלתי מוגבלים.

סגמנטים וטווח כתובות של קוד המערכת:

קוד המערכת, מאוחסן בחלק המרכזי של זכרון הFlash, המכונה Main Memory.

- **טווח כתובות –** טווח זה מתחיל בדרך כלל מכתובת 0xC000 ומתפרס כלפי מעלה. גודלו הסופי תלוי בסיבוכיות ובגודל הקוד שנכתב.
- **סגמנטים –** הקוד תופס את הסגמנטים הגבוהים ביותר בזכרון, החל מ-Segment 31 ומתקדם כלפי מטה לSegment 29, Segment 30 וכו', ככל שנדרש מקום. בקצה העליון ביותר של הזכרון שמורה טבלת פסיקות (Interrupt Vector Table), בטווח הכתובות 0xFFE0 עד 0xFFFF.

סגמנטים וטווח כתובות של הקבצים:

תוכן הקבצים עצמם מאוחסן באזור ייעודי שהוגדר בקוד, בקובץ app.h ניתן לראות זאת.

- **טווח כתובות –** הטווח לאחסון תוכן הקבצים הוא בין הכתובות 0xF600 ל-0xFDFF. טווח זה מספק נפח אחסון של 2KB, כנדרש.
- **סגמנטים –** טווח כתובות זה מתפרס על פני ארבעה סגמנטים של זכרון הFlash, גודל כל סגמנט הוא 512 בתים – סה"כ 2KB.

סגמנט וטווח אחסון מבנה הנתונים לניהול הקבצים:

מבנה הנתונים משמש לניהול מערכת הקבצים, המטא-דאטה של הקבצים, מאחוסן בכתובת נפרדת.

- **כתובת התחלה –** הגדרנו את מבנה הנתונים להיות שמור בזכרון הFlash, החל מכתובת 0xF400, יצרנו מאקרו בשם FILE_METADATA_ADDR בקובץ app.h לצורך זה.
- **סגמנט –** כתובת זו נמצאת ב4 Segment, של זכרון הMM.

תיאור מבנה הנתונים:

מערכת הקבצים מבוססת על שני רכיבים עיקריים:

1. **מערך מטא-דאטה בFlash:**

בכתובת 0xF400 מוקצה מקום למערך של 10 רשומות מסוג FileEntry, שזה מבנה הנתונים, גודלה של כל רשומה הוא 32 בתים שמתארת קובץ בודד ומכילה את המשתנים הבאים:

- **name[17]:** שם הקובץ.
- **type:** סוג הקובץ, האם הוא טקסט או סקריפט.
- **size:** גודל הקובץ בבתים.
- **address:** כתובת הזכרון בFlash שבה מתחיל תוכן הקובץ.
- **padding[8]:** מהשיקול ההנדסי של יעילות ופשטות בחישוב הכתובות הוספנו ריפוד באפסים, הוספה של 8 בתים ריקים כדי להביא את הגודל הכולל של המבנה לגודל קבוע, אחיד ונוח לעבודה(חזקה של 2).

2. **משתני ניהול בRAM:**

בRAM, קיימים שני משתנים גלובליים אשר מנהלים את מצב המערכת:

- **g_num_files:** מונה את מספר הקבצים השמורים ברגע נתון בזכרון.
- **g_next_free_content_addr:** מצביע לכתובת הפנויה הבאה באזור אחסון התוכן, שמבטיח שהקבצים נכתבים באופן רציף ללא חפיפות ביניהם.

תיאור תהליך הניהול:

תהליך הניהול, הוא בעצם תהליך הוספת הקובץ. כדי לתמוך באחסון דינאמי, כאשר משתמש מעלה קובץ חדש, המערכת מבצעת את הפעולות הבאות:

1. בודקת אם יש מקום פנוי במערך המטא-דאטה, אם קטן מ10 קבצים.
2. בודקת אם יש מספיק מקום פיזי באזור שבו מאוחסן התוכן, על ידי בדיקה אם גודל הקובץ החדש בתוספת עם הכתובת הפנויה הבאה, לא חורג מהכתובת הסופית.

3. כותבת את תוכן הקובץ לזכרון הFlash, החל מהכתובת הפנויה הבאה שהמצביע של ניהול הזכרון מצביע עליה.
4. יוצרת רשומה חדשה(FileEntry), עם פרטי הקובץ וכתובת ההתחלה שלו, כמו שמפורט מעלה.
5. כותבת את הרשומה החדשה למקום הפנוי הבא במערך המטא-דאטה בFlash.
6. מעדכנת את מונה ניהול הזכרון בRAM, והמצביע מקודם לכתובת בסוף הקובץ החדש.

בנוסף, המערכת כוללת פונקציית אתחול (FileSystem_Init) הניתנת להפעלה דרך ממשק הGUI, פונקציה זו מבצעת מחיקה מלאה של כל הסגמנטים בזיכרון הפלאש המשמשים לאחסון קבצים (גם המטא-דאטה וגם התוכן), ומאפסת את משתני הניהול בRAM. יכולת זו מאפשרת "לנקות" את המערכת ולהכינה לאחסון קבצים חדשים.

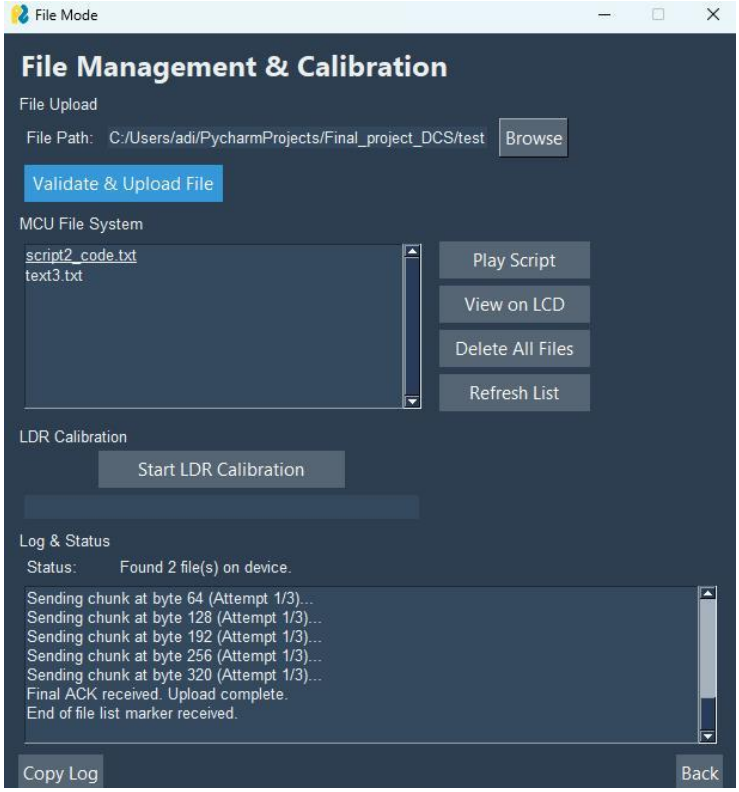
Pseudo-Code לניהול דינאמי של הוספת קובץ:

```
FUNCTION new_file_entry(meta_data, content):
// בדיקת תקינות הקובץ
->if g_num_files >= MAX_FILES:
→return “Error, file limit reached”

->if g_next_free_content_addr + meta_data.size > END_ADDR:
→return “Error, there is not enough space for this file”
// יצירה וכתיבת התוכן של הקובץ לזכרון הFlash
->WRITE_TO_FLASH(g_next_free_content_addr, content)
->new_file.name = metadata.name
->new_file.type = metadata.type
->new_file.size = metadata.size
->new_file.address = g_next_free_content_addr

->metadata_addr = FILE_METADATA_ADDR + (g_num_files * sizeof(FileEntry))
->WRITE_METADATA_TO_FLASH(metadata_addr, new_file)
// עדכון משתני הניהול בRAM
->g_num_files ++
->g_next_free_content_addr += metadata.size
```

```
return TRUE
END FUNCTION new_file_entry
```

ממשק משתמש:

מסך טעינה והעלאת הקבצים:

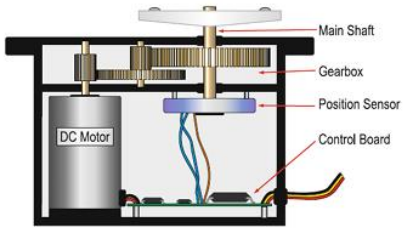


חלוקת העבודה בין החומרה לתוכנה

תפקידי החומרה:

מנוע סרבו

מבצע סיבוב פיזי של מוט סיבוב המנוע, לזווית הרצויה ובכך משמש אותנו כדי להזיז את המערכת כולה, ב180 מעלות במטרה לסרוק את המרחב על ידי החיישנים הרלוונטיים המפורטים מטה. קביעת זווית המנוע מתבצעת על ידי אות PWM, והDuty cycle שלו קובע את הזווית הנוכחית.



ההגדרות הבאות, מציינות את הPWM המינימאלי עבור מיקום זוויתי של 0 מעלות וPWM מקסימלי עבור מיקום זווית של 180 מעלות, כאשר טווחי הביניים הם שיתנו את תחום הזוויות בין 0 ל180.

Servo Motors (Nominal Values)

$f_{max} = 40Hz \rightarrow \ T_{min} = 25msec$

$T_{on} = 0.6msec$ **זווית של 0 מעלות**

$T_{on} = 2.5msec$ **זווית של 180 מעלות**

ערכי כיול המנוע:

ערכי הכיול בפועל, כפי שנקבעו בקוד:

- **0 מעלות** - $T_{on} = 490\mu s = 0.49ms$ ערך זה נגזר מהמאקרו PWM_FOR_CMD_0 490 שהגדרנו. כאשר כל יחידה מייצגת מיקרו-שנייה אחת בהתבסס על שעון טיימר של 1MHz.
- **90 מעלות** - $T_{on} = 1360\mu s = 1.36ms$ ערך זה נגזר מהמאקרו PWM_FOR_CMD_90 1360.
- **180 מעלות** - $T_{on} = 2370\ \mu s = 2.37ms$ באותו אופן, נגזר מהמאקרו המוגדר PWM_FOR_CMD_180 2370.

תחילה ערכי הכיול היו עבור 0 ו180 מעלות, אך משיקול הנדסי וצורת העבודה של המנוע, זיהינו כי ההתנהגות היא לא בדיוק ליניארית ושנצטרך למצוא את ערך הביניים הנכון עבור 90 מעלות ובכך לכייל שונה כל מניפת זוויות. ערכים אלו הם תוצאה של כיול המנוע הספציפי בפרוייקט, הם קרובים לערכים הנומינליים המפורטים מעלה.

אות הבקרה וטיימרים:

אות הבקרה של המנוע הוא אות PWM, המופק על ידי טיימר A1 של הבקר.

אופן הפעולה:

- הטיימר מוגדר לעבוד במצב של Up-mode.
- רגיסטר TA1CCR0 – קובע את זמן המחזור הכולל של האות(תדר).
- רגיסטר TA1CCR2 – קובע את רוחב הפולס(T_{on}), משך הזמן שהאות יהיה במצב של "High". שינוי בערך זה, משנה את הDuty cycle של האות ובהתאם משנה את הזווית של המנוע.
- מצב הפלט מוגדר לOUTMOD7, שזה Reset/Set – המצב הסטנדרטי ליצירת PWM בטיימרים אלו.

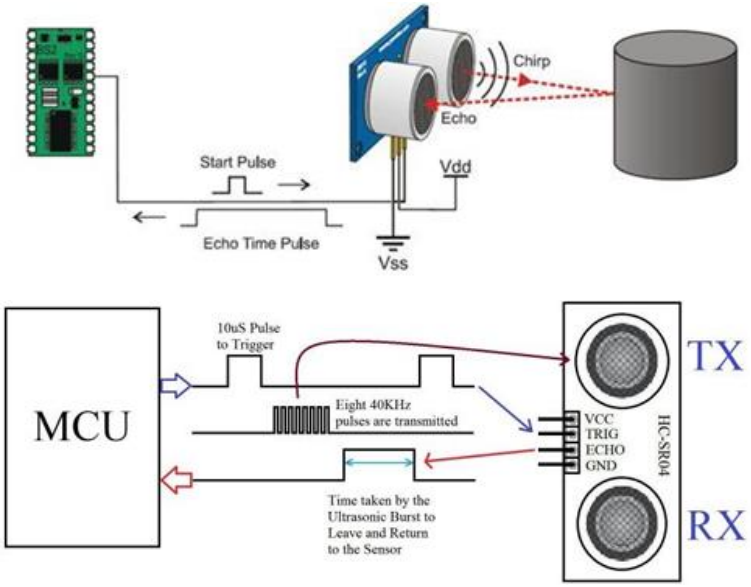
פעולת אות הבקרה:

אות הבקרה של המנוע אינו פעיל באופן רציף, אלא מופעל באופן מבוקר רק למסך הסריקה.

- **אופן הפעולה –** אות הPWM, השולט בזווית המנוע, מופעל בתחילת תהליך הסריקה ונשאר פעיל ברציפות לאורכה. בעל צעד זוויתי, התוכנה מעדכנת את רוחב הפולס(Duty Cycle), כדי שהמנוע יזוז למעלה הבאה, איך אינה מכבה את האות בין הצעדים. רק לאחר השלמת הסריקה כולה, 181 מדידות, קוראים לפונקציה שעוצרת את הטיימר A1 ומכבה את יציאת הPWM.
- **השיקול ההנדסי(Trade-off) –** גישה זו נבחרה כדי להשיג סריקה חלקה ומהירה יותר, על ידי הימנעות מכיבוי והפעלה מחדש של הטיימר בכל אחד מהצעדים של המנוע. גישה זו עלולה להגביר מעט את צריכת ההספק ואת הרעש החשמלי במהלך הסריקה עצמה, אך היא מפחיתה באופן משמעותי את זמן הסריקה הכולל ומונעת "קפיצות" אפשריות של המנוע שהיו נגרמות מהפעלה וכיבוי תכופים.

חיישן המרחק

בהוצאת פולס דרך הבקר, המהווה טריגר דרך רגל הTrigger של החיישן, בסיום הפולס חיישן המרחק "יורה" גל קול באורך שמונה מחזורים בתדר של 40kHz לכיוון האובייקט וקולט את ההחזרים שמגיעים ממנו. המעגל החשמלי שנמצא בחיישן, ממיר את החזרי גל הקול לפולס היוצא מרגל הEcho של החיישן באורך הזמן שעבר מרגע שידור גל הקול ועד לקבלת ההחזרים מהאובייקט הנמצא מול החיישן, פולס זה שיוצא מרגל החיישן, נכנס לרגל בקר בעלת יכולת פסיקה וטווח המדידה שלו הוא מ2cm ועד 450cm.



מדידת המרחק מתבצעת בעזרת מימוש Input Capture בלבד והיא נראת כך:

$$\boldsymbol{Range[cm] \cong Echo\_high\_level\_time[\mu sec] \cdot \frac{34,000\left[\frac{cm}{sec}\right]}{2}}$$

$$when: c = 34{,}000\left[\frac{cm}{sec}\right] \; is\ the\ speed\ of\ sound$$

כאשר מהירות הקול באוויר תלויה בטמפרטורה, אך ההנחה בפרויקט שלנו היא שהטמפרטורה היא טמפרטורת חדר של 25 מעלות צלזיוס, מכיוון שאין לנו חיישן טמפרטורה ואין צורך בכך בהגדרת הפרויקט.

החיישן הספציפי שאיתו עבדנו הוא מסוג HC-SR04 וזה המפרט שלו:

	HC-SR04
Working Voltage	5 VDC
Static current	< 2mA
Output signal:	Electric frequency signal, high level 5V, low level 0V
Sensor angle	< 15 degrees
Detection distance (claimed)	2cm-450cm
precision	~3 mm
Input trigger signal	10us TTL impulse
Echo signal	output TTL PWL signal
Pins	1. VCC 2. trig(T) 3. echo(R) 4. GND

גרעיניות המדידה והערך הגולמי:

- **גרעיניות(Granularity) –** גרעיניות המדידה של הטיימר במצב Input capture היא 1 מיקרו שנייה. הדבר נובע מכך שטיימר A1, המשמש למדידה, מקבל את אות השעון שלו מSMCLK, בתדר של 1MHz, ללא חלוקה. לכן, כל "טיק" של הטיימר מייצג $1\mu s$.
- **הערך הגולמי –** הערך הגולמי המייצג את המרחק הוא הפרש הזמנים, בין העליה לירידה של אות הEcho, שהוא למעשה רוחב הפולס ביחידות של מיקרו-שניות.

אות פולס הTrigger:

אות הTrigger, אינו מופק על ידי הטיימר אלא מופק על ידי שליטה ישירה על פין הI/O הרגיל, P2.6.

אופן הפעולה:

הפונקציה Ultrasonic_start_measurement מבצעת את הרצף הבא:

1. מעלה את פין P2.6 למצב "High".
2. ממתינה 15 מיקרו-שניות באמצעות השיית תוכנה.
3. מורידה את פין P2.6 למצב "Low".

חיישני הLDR

מערכת גילוי מקורות האור מבוססת על שני חיישני LDR (נגד תלוי-אור) המשמשים לזיהוי מיקומם והערכת מרחקם של מקורות אור במרחב הנסרק. כל חיישן מהווה חלק ממעגל מחלק-מתח, כאשר המתח האנלוגי המשתנה ביחס ישר להתנגדות (ובאופן הפוך לעוצמת האור) נדגם על ידי ממיר אנלוגי-לדיגיטלי (ADC) מובנה בבקר.

על מנת להבטיח מדידה מדויקת לתנאי תאורה משתנים, המערכת כוללת מנגנון כיול ייעודי. בתהליך הכיול, נאספות 10 דגימות ADC מכל חיישן במרחקים ידועים (עד 50 ס"מ), ונתוני כיול גולמיים אלו נשמרים בזיכרון הFlash, הבלתי נדיף של הבקר. בצד המחשב, נתונים אלו משמשים ליצירת גרף המרה מלא באמצעות אינטרפולציה לינארית. בזמן סריקה, הבקר שולח את ערכי ה ADC-הגולמיים למחשב, אשר משתמש בגרף הכיול כדי להמיר את הערכים למרחק בסנטימטרים ולקבוע את מיקומו של מקור האור.

סגמנטים וטווח כתובות של ערכי הכיול:

20 דגימות הכיול, 10 לכל חיישן מאוחסנות בזכרון הInformation Memory של הFlash.

- **LDR1 –** מאחוסנות החל מכתובת 0x1040, המהווה את תחילתו של Segment C.
- **LDR2 –** מאחוסנות החל מכתובת 0x1080, המהווה את תחילתו של Segment B.

אינטרפולציה:

השלמת 10 הדגימות הכיול לגרף של 50 נקודות מתבצעת באמצעות אינטרפולציה ליניארית. מכיוון שיש לנו דגימה כל 5 ס"מ, הערכים עבור כל ס"מ בודד ביניהן מחושבים בהנחה שהתנהגות החיישן היא קו ישר בין שתי נקודות הדגימה. הנוסחה לחישוב ערך ADC(y), במרחק x בין שתי נקודות ידועות: (x_1, y_1) ו(x_2, y_2) היא:

$$y = y_1 + (x - x_1) * \frac{y_2 - y_1}{x_2 - x_1}$$

צד ביצוע האינטרפולציה הוא כולו בצד המחשב. הבקר רק אוסף את 10 הדגימות הגולמיות עבור כל חיישן ושומר אותן בFlash.

שמירת הגרף, טבלת המרה של 50 ערכים, נשמרת בזכרון הRAM, של המחשב כחלק ממצב הריצה של אפליקציית הGUI.

קביעת המרחק:

לאחר שהגרף המלא נוצר ונשמר בזכרון המחשב, קביעת המרחק מתבצעת באופן הבא:

1. הבקר דוגם את המתח על הLDR, שולח למחשב את ערך הADC הגולמי.
2. בצד המחשב, התוכנה מבצעת חיפוש בטבלת 50 הערכים כדי למצוא את המיקום הקרוב ביותר לערך הADC שהתקבל.
3. האינדקס של התא בטבלה שבו נמצא הערך הקרוב ביותר, הוא המרחק הנמדד בסנטימטרים.

תפקידי התוכנה:

- **יצירת אותות הבקרה**: התוכנה בבקר אחראית לייצר את פולס הTrigger עבור החיישן האולטראסוני ואת אות הPWM המדויק לשליטה בזווית מנוע הסרבו.
- **מדידה וקריאת נתונים**: התוכנה משתמשת בטיימר במצב Input Capture כדי למדוד את רוחב פולס הEcho שמגיע מהחיישן האולטראסוני וביחידת הADC, כדי למדוד את המתח האנלוגי הנופל על חיישני הLDR.
- **ניהול לוגיקת המערכת**: מימוש ה – FSM, המעבר בין מצבי הפעולה השונים בהתאם לפקודות המתקבלות מצד המחשב.

- **עיבוד ואלגוריתמיקה**: הפעלת האלגוריתמים לזיהוי אובייקטים ומקורות אור על בסיס הנתונים הגולמיים מהחיישנים.
- **ניהול תקשורת**: שליחה וקבלה של פריימים של מידע דרך הUART אל רכיב הRS-232.
- **ניהול מערכת קבצים**: כתיבה, קריאה וניהול של קבצים המאחוסנים בזכרון הFlash של הבקר.

מבנה נתונים ואלגוריתמים:

מבנה הנתונים לקליטת הפריים(Rx):

מבנה נתונים לקליטת מידע מהמחשב, הוא Circular Buffer שמבוסס על שיטת FIFO.

המימוש מתבצע בשכבת הHAL, באמצעות מערך גלובלי g_rx_buffer, בגודל 65 בתים שיתאים לגודל של chunk מידע(64 בתים) ותוספת בית אחד עבור חישוב הchecksum, ושני מצביעים – head וtail.

- **תהליך הקליטה –** פסיקת הUART, פועלת כיצרן ומוסיפה כל בית שמתקבל לראש הבאפר, על ידי שימוש במצביע הhead.
- **תהליך העיבוד –** הלולאה הראשית בתוכנית, בקובץ המain, פועלת כצרכן וקוראת בתים מה"זנב" של הבאפר, באמצעות המצביע tail, לצורך עיבוד.

השיקול ההנדסי והיתרון במימוש זה שמבנה זה מונע איבוד נתונים ומפריד בין תזמון של קבלת המידע, שנקבע על ידי המחשב לבין תזמון העיבוד שלו, שנקבע על ידי לוגיקת המערכת מה שחיוני למערכת שעובדת בגישת Interrupt-driven.

גודל הפריים הנקלט(Rx):

גודל הפריים הנקלט מהמחשב הוא משתנה. הפרוטוקול תומך בסוגי מסרים שונים באורכים שונים, מה שהופך אותו ליעיל ביותר.

- **פקודות של בית אחד –** רוב הפקודות לניווט התפריט הראשי, הן תו בודד.
- **פקודות של שני בתים –** הפעלת מד המרחק, דורשת שני בתים: גם את התו הבודד וגם את הבית שלאחריו שמייצג את הזווית.
- **אורך משתנה –** העלאת קובץ כוללת פריים של מטא-דאטה באורך קבוע של 20 בתים ולאחריו תוכן הקובץ שהוא באורך משתנה, גם קבלת סקריפט להרצה היא באורך משתנה שמסתיים בתו מוגדר מראש "\n".

מבנה הנתונים לשליחת הפריים(Tx):

במימוש הנוכחי, בצד השידור, לא נעשה שימוש במבנה נתונים כמו Circular Buffer.

השידור מתבצע בגישה של Blocking שמבוססת על Polling. פונקציות השידור שולחות בית אחר בית ישירות לחומרת הUART. לפי כל שידור בית, הקוד ממתין בלולאה עד שבאפר השידור של החומרה מתפנה, זוהי שיטה פשוטה למימוש.

גודל הפריים המשודר(Tx):

גודל הפריים המשודר מהבקר למחשב הוא משתנה. סוג המידע שנשלח מכתיב את גודל הפריים.

- **פריים של בית אחד –** Ack/Nack, בתהליך העלאת הקבצים נשלחים כתו בודד.
- **פריים של 3 בתים –** במצב של מד מרחק(Telemeter) ובסריקת אובייקטים, מצבים 1 ו2 של המערכת – נשלח פריים קבוע של 3 בתים, 2 עבור מדידת מרחק ואחד עבור זווית.
- **פריים של 5 בתים –** בסריקת מקורות אור, נשלח פריים קבוע של 5 בתים, 2 לכל LDR ו1 עבור הזווית.
- **אורך משתנה –** שליחת רשימת הקבצים למחשב כוללת שליחת מספר שמות קבצים באורכים שונים.

תמיכה במימוש Checksum:

כן ,המערכת תומכת בגילוי שגיאות באמצעות מנגנון Checksum 256, מומש באופן מודע ומכוון, רק עבור מצב העלאת הקבצים, ולא עבור שליחת נתוני הסריקה בזמן-אמת. החלטה זו נבעה משיקול הנדסי(Trade-off), המבדיל בין דרישות שני סוגי התקשורת, בהעלאת הקבצים, שלמות המידע היא קריטית, כאשר כל שגיאה, אפילו של תו בודד יכולה להרוס קובץ שלם ולהפוך אותו ללא רלוונטי ולכן, קריטי ליישם כאן מנגנון של חישוב ושליחת Checksum ושליחה חוזרת במקרה של שגיאה. מצד שני, בשליחת נתוני סריקה בזמן אמת, עמידה בזמנים וקצב רציף חשובים יותר משלמות של חבילת מידע בודדת, אם חבילה אחת מתוך 180 תגיע משובשת, הנזק למערכת הכוללת הוא זניח וניתן עדיין לדייק על ידי אלגוריתם בצד המחשב את זיהוי האובייקט או מקור האור, הוספת מנגנון Checksum במצב זה הייתה מאטה את קצב הסריקה ומוסיפה סיבוכיות מיותרת ללא תועלת משמעותית.

- **אופן המימוש בצד המחשב (השולח):**
  1. לפני שליחת חבילת מידע (כמו המטא-דאטה של הקובץ או "chunk" של תוכן), קוד הפייתון קורא לפונקציה calculate_checksum.
  2. פונקציה זו סוכמת את כל ערכי הבתים בחבילת המידע, מחשבת את המשלים ל2 של הסכום(מודולו 256), ומחזירה את בית הchecksum.
  3. בית הChecksum, מצורף לסוף חבילת המידע והחבילה המלאה(מידע + Checksum), נשלחת לצד הבקר.

- **אופן המימוש בצד הבקר (המקבל):**

1. הבקר, בפונקציה Handle_upload_Byte, אוסף את כל הבתים של החבילה הנכנסת לתוך באפר זמני, g_upload_buffer.
2. לאחר קבלת החבילה, קורא לפונקציה is_checksum_valid, פונקציה זו סוכמת את כל הבתים בחבילה שהתקבלה, כולל בית הChecksum.
3. אם החבילה תקינה, התוצאה של הסכום (בחשבון של 8 סיביות) חייבת להיות אפס. אם התוצאה אכן אפס, הפונקציה מחזירה TRUE והבקר ממשיך בתהליך ושולח Ack. אם התוצאה שונה מאפס, זוהי אינדיקציה לשגיאת שידור, הפונקציה מחזירה FALSE, הבקר שולח Nack ומבקש שליחה חוזרת.

תמיכה בStop-and-wait ARQ:

כן המערכת תומכת בS&W ARQ באופן מלא בתהליך העלאת הקבצים מהמחשב לבקר.

פרוטוקול זה מבטיח העברת מידע אמינה על ידי דרישה לאישור (Ack) מהמקבל לפני שהשולח ממשיך לחבילת המידע הבאה.

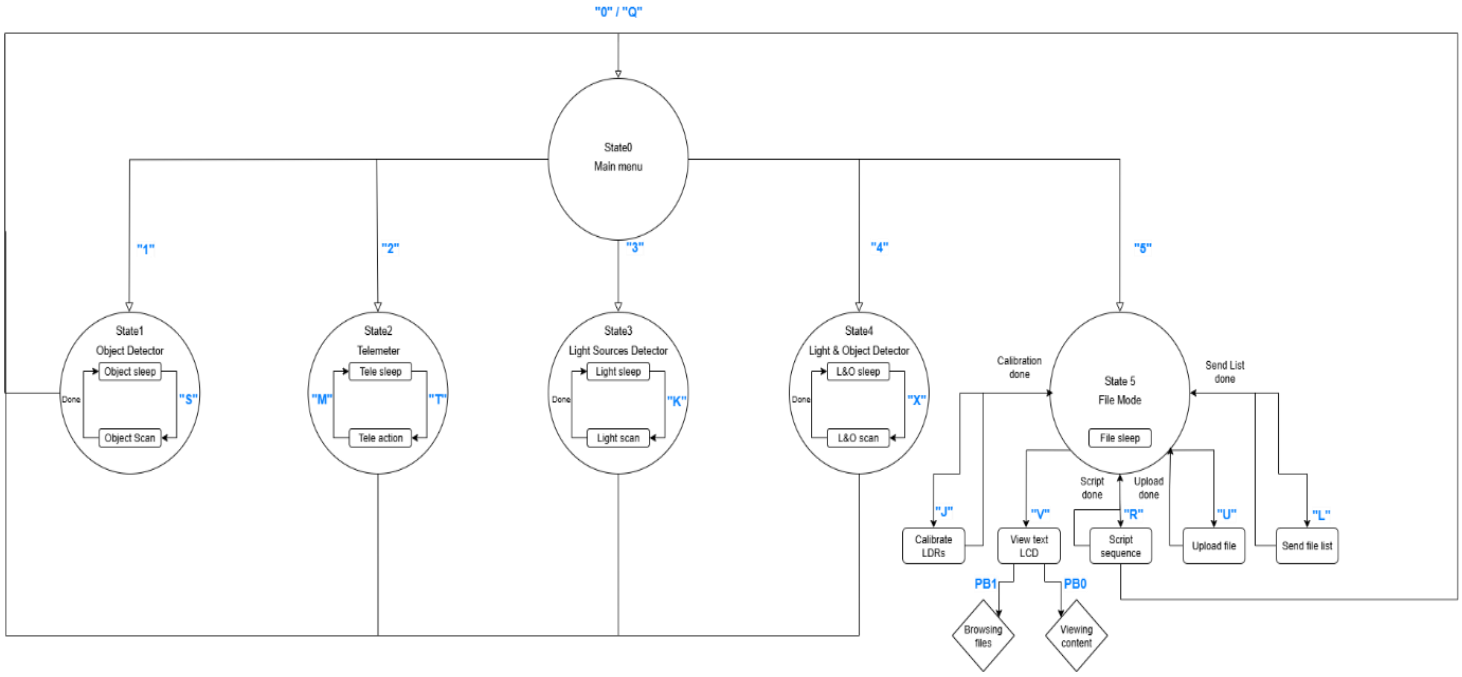
אופן המימוש בבקר:

1. **קבלת חבילה ואימות Checksum –** הבקר מקבל חבילת מידע(מטא-דאטה או "chunk" של תוכן), אוסף אותה לבאפר זמני. מיד לאחר קבלת החבילה המלאה, הוא מפעיל את פונקציית is_checksum_valid כדי לוודא את תקינותה על ידי חישוב סכום כל הבתים במודולו 256.
2. **שליחת משוב –**

- **במקרה של הצלחה –** כאשר הChecksum תקין, הבקר שולח Ack – תו אישור ייעודי עבור מטא-דאטה הוא "B", "K" עבור chunk ו"S" לסיום. מה שמאותת למחשב להמשיך לחבילה הבאה.
- **במקרה של כשלון –** כאשר הChecksum שגוי, הבקר שולח בחזרה Nack – תו "N" או תו שגיאה אחר "F" עבור חוסר מקום, המורה למחשב לשלוח את אותה החבילה שוב.

לוגיקת הTimeout והשליחה החוזרת (Retransmission) ממומשת כולה בצד המחשב (השולח), כנדרש מהפרוטוקול. הבקר מספק את כל המשוב הנדרש, המבוסס על אימות הChecksum, כדי לאפשר למחשב לנהל תהליך העלאה אמין ומבוקר.

גרף מצבים FSM צד MCU



מעגל חשמלי MCU