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

פרויקט גמר

קורס: מבנה מחשבים ספרתיים

361-1-4191

Control system of motor-based machine

תכנון ומימוש מערכת בקרה למכונה מבוססת

מנוע צעד בשליטה ידנית ומרחוק

מגישים: גיא כהן 207881004
ליאב בן אור 315909390

תוכן עניינים

מטרת הפרויקט

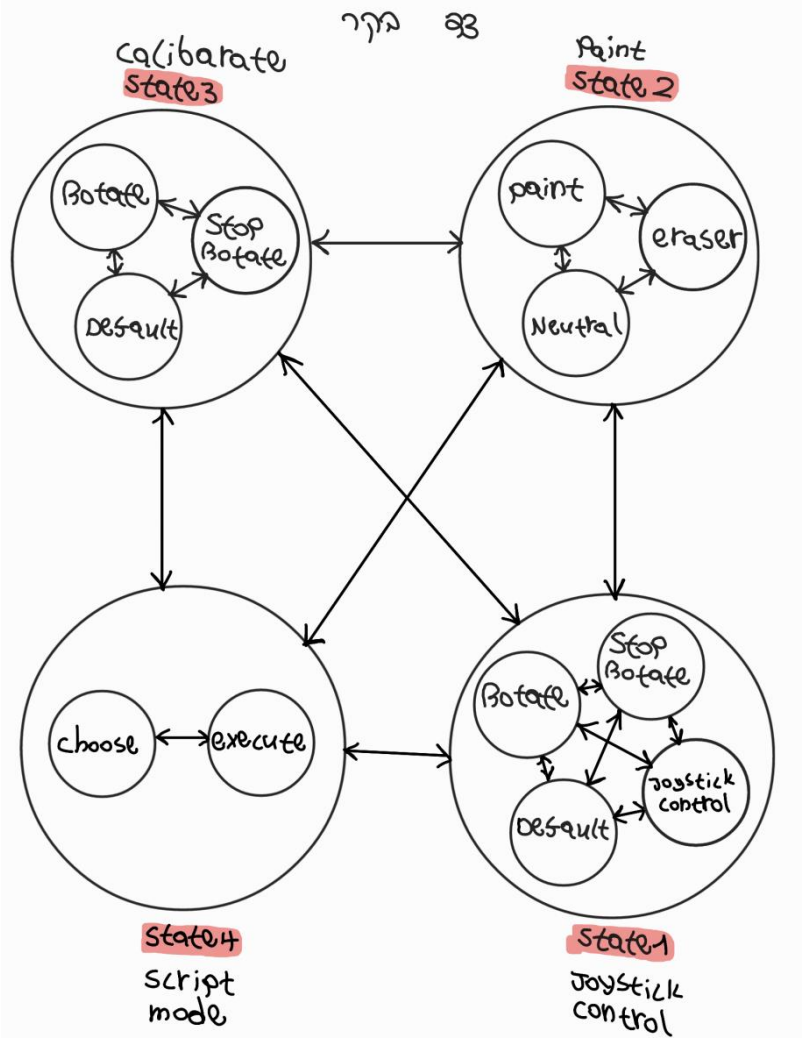
הפרויקט הינו פרויקט הגמר של הקורס "מבנה מחשבים ספרתיים" (DCS) . במהלכו נדרשנו לממש מערכת בקרה למכונה מבוססת מנוע צעד באמצעות שליטה ידנית על ידי ג'ויסטיק וגם בשליטה מרחוק ממחשב אישי דרך ערוץ תקשורת טורית.

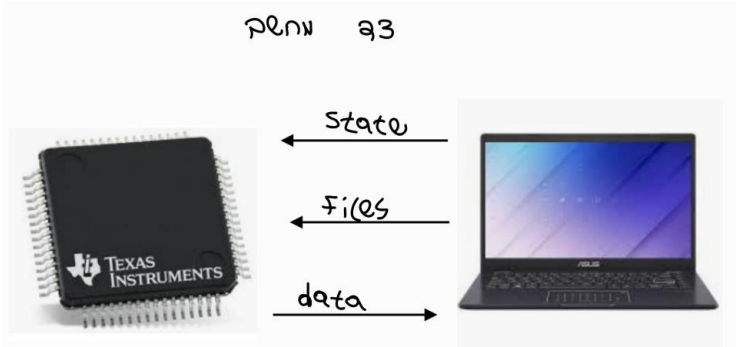
הפרויקט מומש על גבי בקר MSP430G2553 בשפת C בתוכנת CCS אשר עליו נממש את מגוון המשימות שהתבקשנו לממש, המחשב האישי ישלח את בקשות המשתמש דרך ממשק GUI, הבקר יקבל את הבקשות ע"י ערוץ התקשורת הטורית ויבצע את הדרוש.

דיאגרמת FSM (מצבי המערכת)

המערכת שבנינו צריכה לעבור בין המצבים השונים בצורה רובסטית, לפי תיאור המשימה בנינו דיאגרמת מצבים.

להלן דיאגרמת הFSM –

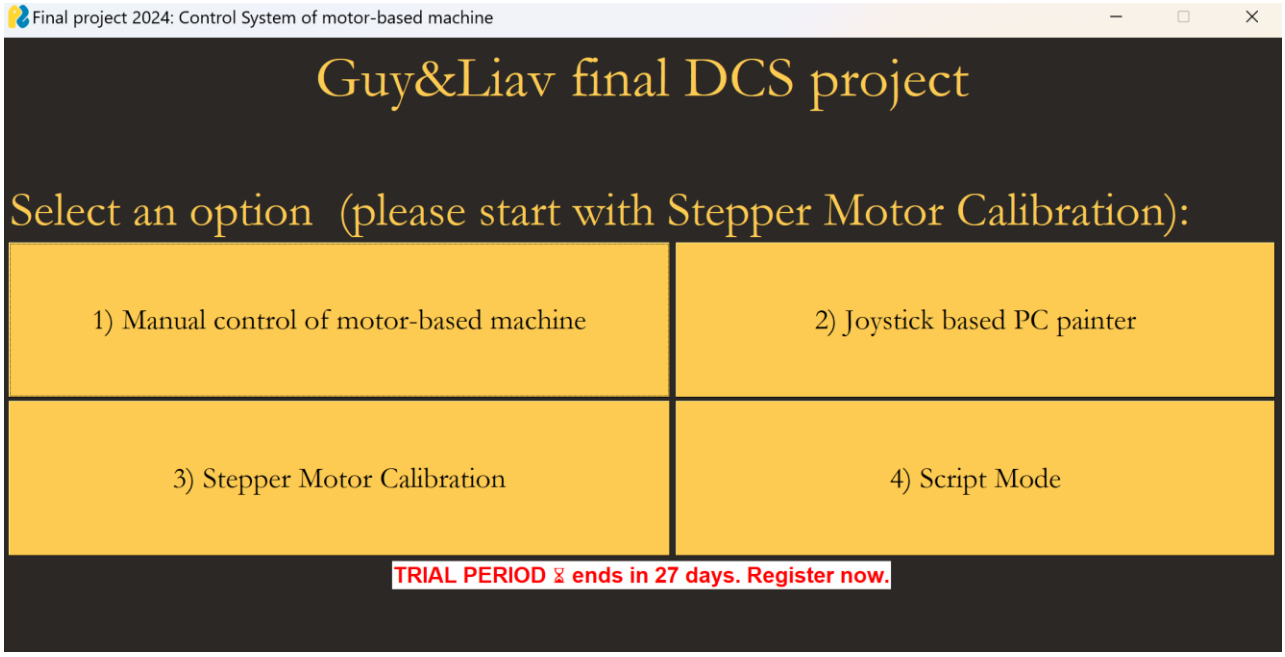




ממשק משתמש – בקר

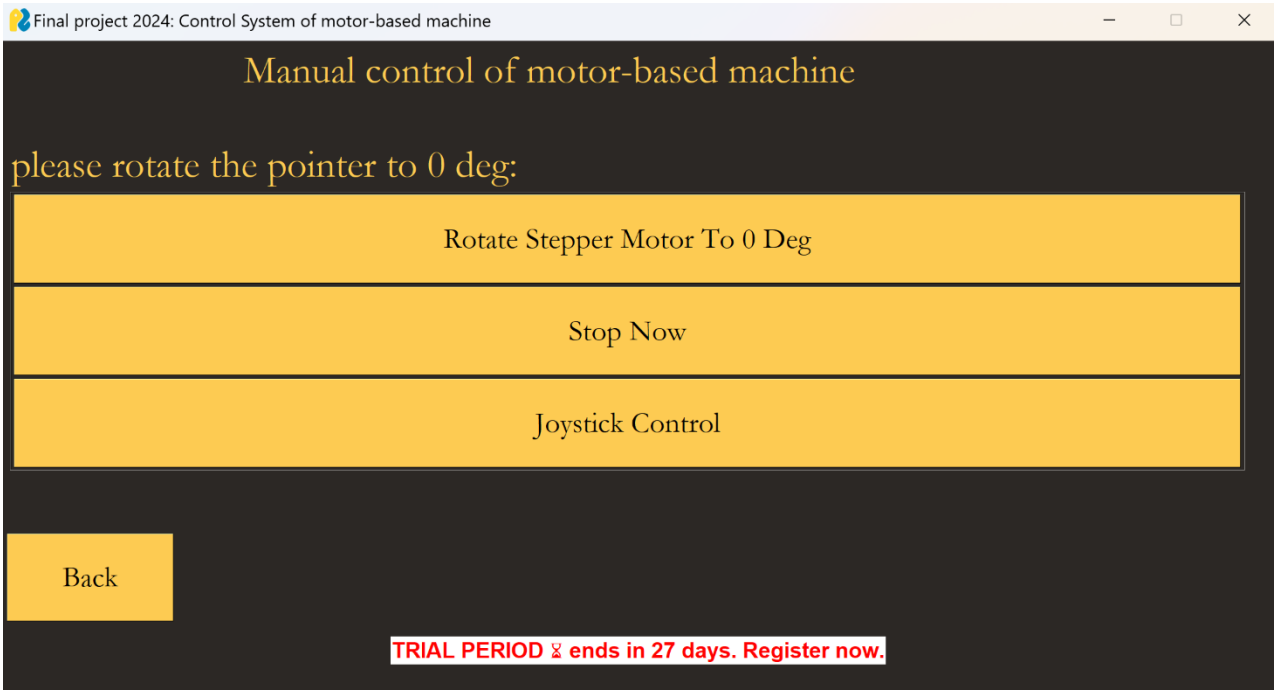
הממשק למשתמש יבוא לידי ביטוי בצד PC באמצעות GUI המכיל כפתורים כאשר כל כפתור מייצג פקודה אחרת שנדרש לבצע. עם הלחיצה על כל כפתור הPC ישלח באמצעות תקשורת UART אות מייצגת עבור הכפתור הרלוונטי. עם קבלת המידע בבקר תתקבל פסיקה ובהתאם הבקר ישנה את המצב שלו בFSM וכך הוא יבצע את הפקודה הרלוונטית ללחיצה מהצד של המשתמש.

להלן מסך התפריט הראשי –



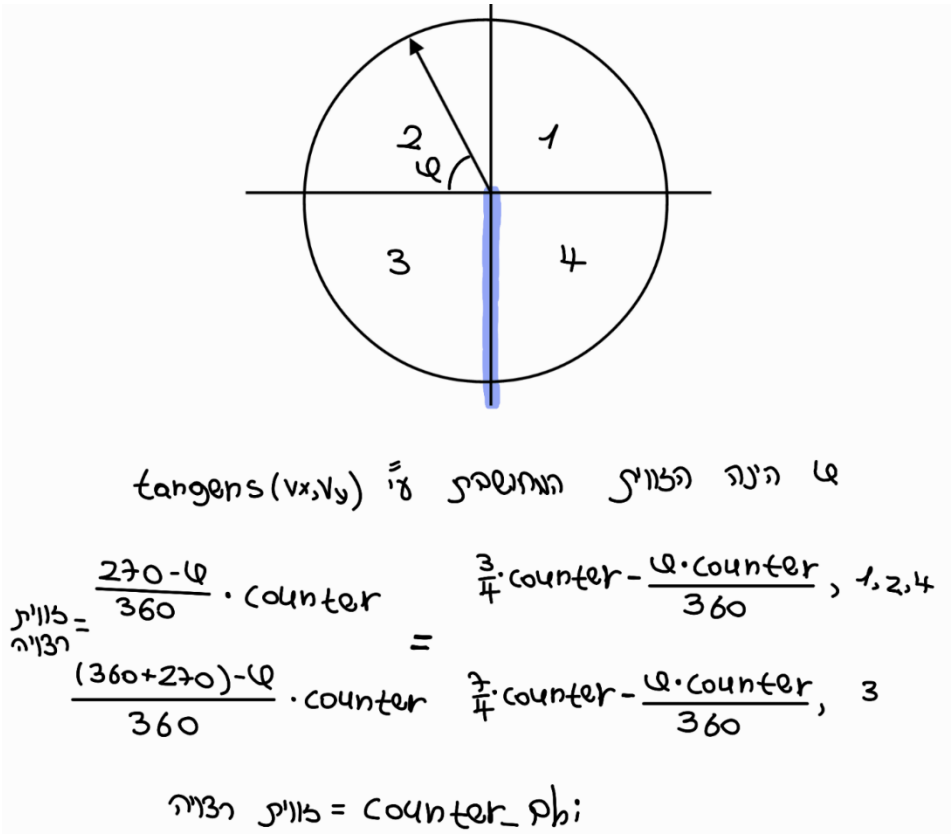
Manual control of motor-based machine

במשימה זו נדרשנו לגרם להזזת המצביע של מנוע הצעד לפי הזווית שעליה מצביע הג'ויסטיק, כפי שמצוין על חלון הGUI על המשתמש קודם כל להביא את המצביע לזווית 0 (על הפס הכחול שמסומן על גבי מנוע הצעד). כפי שיוסבר בהמשך, כאשר עושים שימוש ראשון במערכת על המשתמש לבצע כיול על מנת לחשב את הזווית המדוייקת אשר משתנה ממנוע אחד למנוע אחר



אלגוריתם למימוש המצב:

לאחר ביצוע הכיול ההתחלתי, משתנה counter שומר את מספר הצעדים עבור סיבוב שלם של המצביע, כלומר ע"י חלוקה ב360 נוכל למצוא את הזווית phi שהמצביע עובר בכל צעד.
המשתנה curr_counter מחזיק את הזווית הנוכחית של המצביע ביחידות של כמות צעדים בסיבוב שלם, עבור צעד בכיוון השעון נעלה אותו ב1 ונגד כיוון השעון נוריד ב1.



הזזת מנוע בצעד עם כיוון השעון – ביצענו הזזה כל עוד המצב הזה הוא הנבחר של ה*stepper motor* עם כיוון השעון יחידה אחת וגם קידום של הזווית הנוכחית ב1.

Rotate Stepper Motor To 0 Deg

עצירת המנוע – הגדרת הבקר להיות במצב שינה.

Stop Now

הזזת המנוע לפי הג'ויסטיק – במצב זה נאלצנו לעבוד בבקרה בחוג בפתוח על ידי כך שבכל איטרציה אנחנו יודעים את הזווית שכרגע המצביע נמצא ואת הזווית אליה נרצה להגיע. (ע"י החישוב שפירטנו מעלה) באמצעות בדיקה פשוטה של המקסימום בניהם, ניתן להסיק לאיזה כיוון על המצביע לנוע – אם המצביע קטן מהזווית הרצויה נזוז עם כיוון השעון. אחרת, ננוע נגד כיוון השעון.

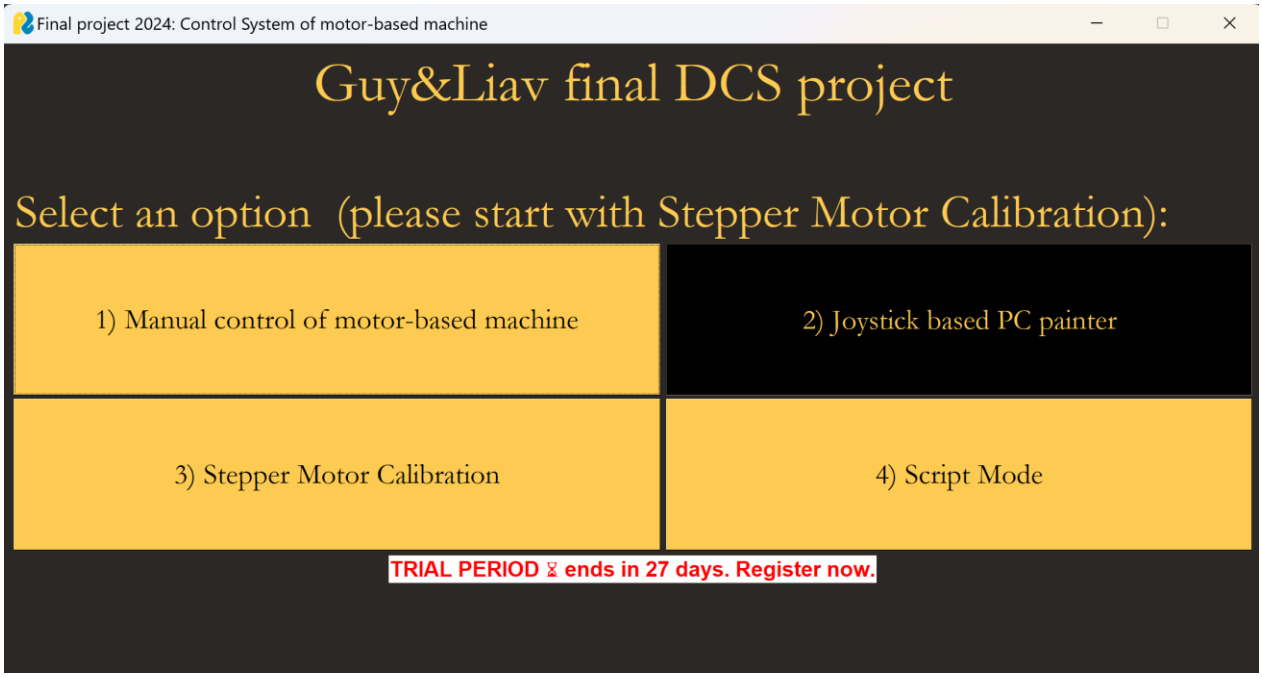
במידה והג'ויסטיק במצב רגיל ולא מוסט ממקומו הגדרנו לא תתבצע תזוזה של הפוינטר.

כעת נותר למצוא את הזווית הרצויה אותה נמצא על ידי דגימה של ערכי רגלי הג'ויסטיק על ידי ה*ADC* בשני הצירים ולאחר קבלת הדגימות נבצע חישוב של הזווית על ידי פונקציה tangens אשר מחשבת את הזווית בהינתן 2 וקטורים ב$fixed\ point$ ובעלת דיוק של $\pm 1°$. נמצא את הזווית הרצויה ביחידות של כמות צעדים לסיבוב בדרך שפירטנו מעלה.

Joystick Control

Joystick based PC painter

במשימה זו נרצה לממש צייר על גבי מסך המחשב הנשלט על ידי ג'ויסטיק. כאשר המשתמש יכול לברור בין שלושה מצבים – כתיבה, מחיקה, מצב ניטרלי.

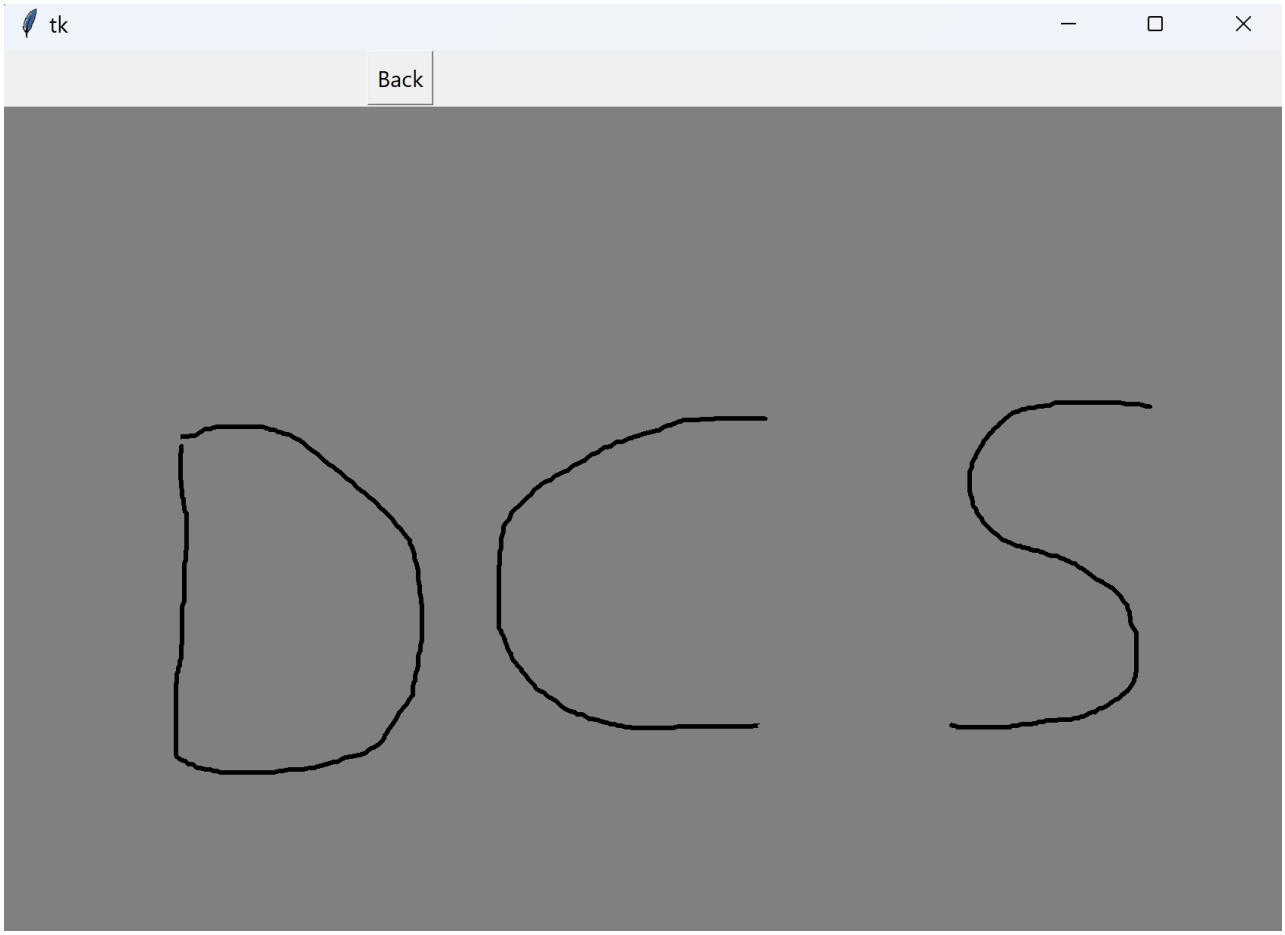


אלגוריתם למימוש המשימה:

באופן רציף ובקצב מקסימלי דגמנו את ערכי מיקום הג'ויסטיק ושלחנן את ערכי הדגימה לצד המחשב, כעת בצד המחשב נממש שסמן הצייר זז לפי תזוזת העכבר.

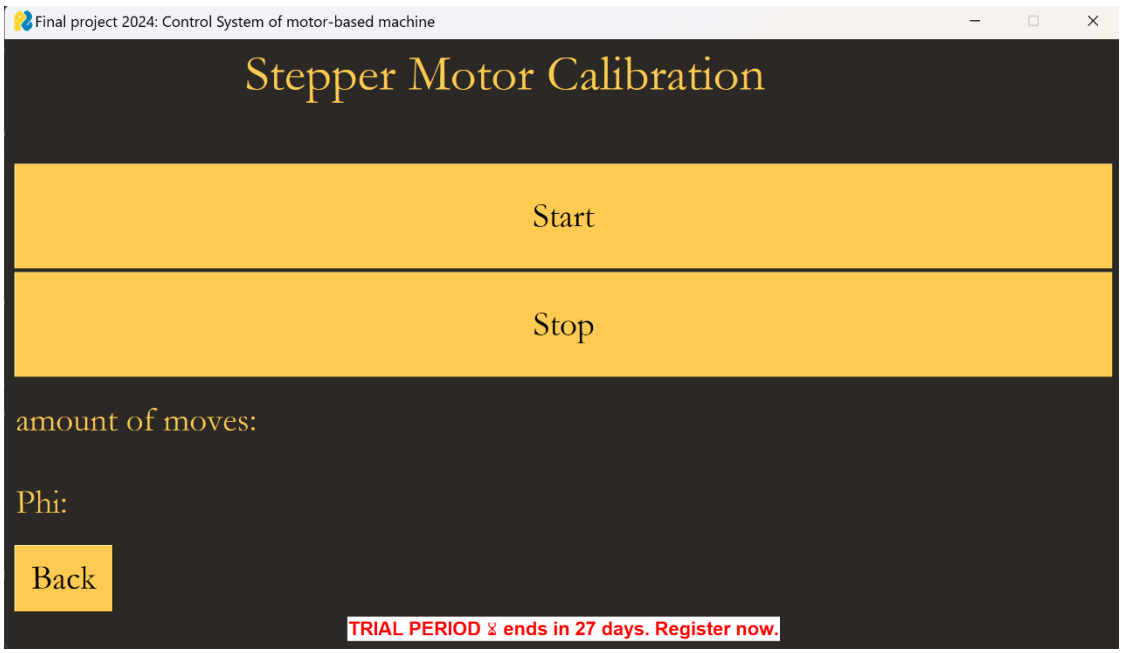
אנחנו רוצים שהעכבר יזוז לפי תזוזת הג'ויסטיק ולכן נחשב את ההפרש בין דגימות עוקבות ונזיז לפיכך את העכבר.

איתחלנו את המערכת להיות במצב ניוטרל בכניסה למצב paint, המצבים ישתנו ע"י לחיצה על הלחצן, ברוטינת הפסיקה של הלחצן נשלח לצד המחשב ערכי דגימות מחוץ לטווח הערכים ({1000,1000}) וכך צד המחשב יסיק שבוצע שינוי מצב ועליו להחליף tool בכתבן. מעבר בין מצבים יתבצע ע"י מודולו 3.



Stepper Motor Calibration

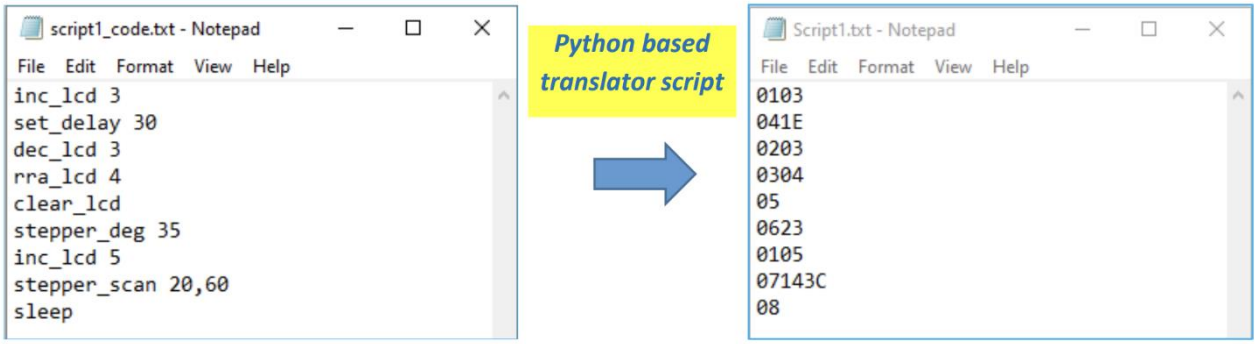
במשימה זו, עלינו לבצע כיול למנוע הצעד ולהציג על גבי מסך המחשב את כמות הצעדים בסיבוב וגודל זווית הצעד φ של מנוע הצעד. מפני שאנחנו עובדים בבקרה בחוג פתוח, מנוע הצעד נדרש לקבל חיווי מהמשתמש על נקודת התחלה שתקבע בתור נקודת $\varphi = 0$. כעת ניתן לפרוט את המשימה להכיל 2 פקודות – הזזת מנוע הצעד עם כיוון השעון (אין חשיבות לכיוון התזוזה), על המשתמש לעצור את במצביע על הקו הכחול (זווית 0) ולאחר מכן להתחיל סיבוב חדש, כאשר המשתמש יבחין על השלמת סיבוב מלא עליו לעצור בשנית את המצביע על הקו הכחול. צד הבקר יחשב את כמות הצעדים הנדרשים להשלמת סיבוב וע"י חלוקה ב360 נקבל את הזווית של מנוע הצעד.



Script Mode

במשימה זו, היה עלינו לבצע מספר פעולות שונות בבקר בהתאם לקובץ script המכיל פקודות high level המוגדרות מראש. עלינו לתמוך בשליחה וקבלה של עד שלושה קבצים ולבחור להפעיל אחד מהם בנפרד ובאופן בלתי תלוי מהשאר.

לפני שליחת הקבצים מצד המחשב לצד הבקר, עלינו לבצע פיענוח של הפקודות כמתואר להלן:

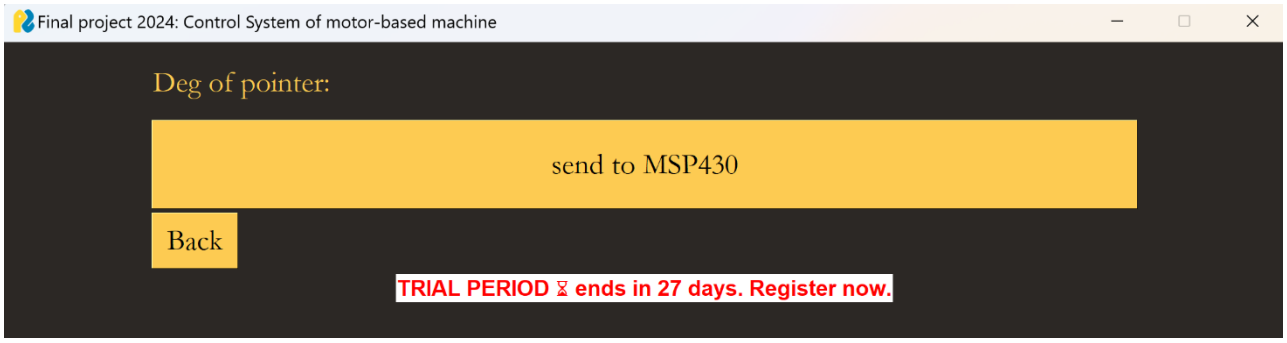


את קבצי הscript נצרוב לרכיב הFLASH הנמצא בבקר, כל סקריפט נצרוב לסגמנט מתאים בפלאש. נגדיר משתנה מסוג struct אשר יכיל את השדות הבאים: כמות קבצים קיימים, מערך המכיל את שם הקובץ, מערך מצביע לתחילת כל קובץ ומערך המכיל את גדלי -הקבצים.

את הצריבה של כל קובץ נבצע לפי אינדקס הבחירה מהתפריט בGUI.

עבור מימוש התפריט המשני: התפריט המשני של המשימה הנ"ל יכיל בתוכו אופציה לבחירת תיקייה מהמחשב ממנה ניתן לברור בין קבצי טקסט, בנוסף לאחר בחירת הקובץ הרצוי, על מנת לצרוב אותו לבקר נלחץ על הכפתור המתאים לצריבה (choose). את הקבצים שנצרבו ניתן יהיה לראות בתפריט הימני שיכיל את הקבצים ופירוט גודלם והתוכן שלהם.

עם לחיצה על *Execute* נפתח חלון חדש –



חיבורי חומרה:

UART: תקשורת טורית מול המחשב- p1.1, p1.2.

LCD: קו הבקרה E לרגל p1.0, קו הבקרה RS לרגל p1.6 וקו הבקרה RW לרגל p1.7.
ארבעת קווי המידע של הLCD מחוברים לרגליים p2.4-p2.7.

Vrx: p1.3.

Vry: p1.4.

PB: לחצן של הג'ויסטיק p1.5.

STEPMOTOR: ארבע פאזות של מנוע צעד p2.0-p2.3.

הגדרות חומרה ותוכנה

לצורך ביצוע משימות הפרויקט השתמשנו במודולים חומרתיים שונים בצד הבקר ובנוסף לכך רכיבי תוכנה בצד המחשב. נפרט את אופן מימושן:

חומרה

בפרויקט כאמור השתמשנו בבקר $MSP430$ המכיל מודולי חומרה שונים שנגענו בהם בפרויקט.

ADC10

השתמשנו בADC רק בכדי לדגום את ערכי מתחי הג'ויסטיק לביצוע המשימות השונות. לג'ויסטיק 2 צירים ולכן נרצה לבצע דרך הADC דגימות בצורה מערכית בגודל 2 לתוך מערך Vr כך שערך הדגימה הראשונה והשנייה ישמרו במערך באינדקס הראשון והשני בהתאמה.

TIMER

השימוש בטיימרים בפרויקט בא לידי ביטוי בהשהיות שנעשו בתהליכים שונים בפרויקט. המימוש הוא על ידי הפעלת הטיימר לפרק הזמן שנרצה להכניס אותו לפי חישוב והכנסת הבקר מיד אחרי למצב שינה. לאחר פרק הזמן הרצוי הטיימר יגיע לפסיקה בה הוא יוצא ממצב שינה וממשיך את התוכנית מהמצב בו עצר.

UART

במהלך הפרויקט היינו צריכים לפעול בקורלציה בין הבקר למחשב ולכן השתמשנו בתקשורת UART בניהם עם הפרמטרים – 9600 BPS , 8-bits , 1 Start , 1 Stop , (none) No parity. כעת נסביר על תשתית הUART בכל אחד מהצדדים.

צד מחשב – בנינו פונקציה ששולחת string מהצד מחשב לצד בקר באמצעות שליחה של אות–אות. בנוסף, בנינו פונקציה שקולטת מידע מהבקר.

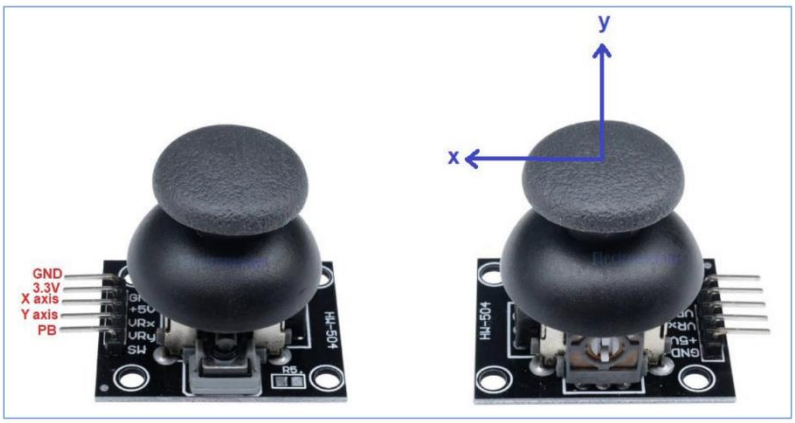
צד בקר – בבקר קיים מודול UART אשר יודע לקבל פסיקות מסוג RX וגם TX כנלמד במהלך הקורס. לכן בצד הבקר ביצענו ISR לכל אחד מסוג הפסיקות.

ג'ויסטיק

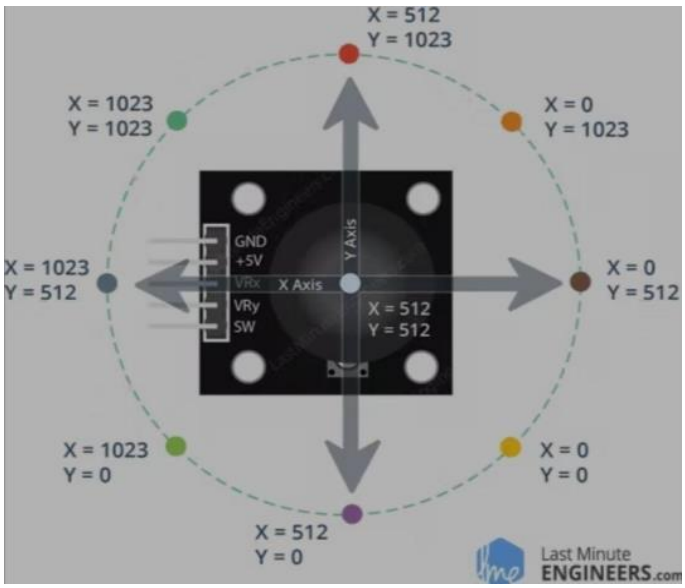
מוט היגוי joystick:

כמפורט לעיל, לצורך שליטה על זווית המיקום של מוט מנוע stepper נשתמש במוט היגוי (joystick) הפועל בצורה הבאה:

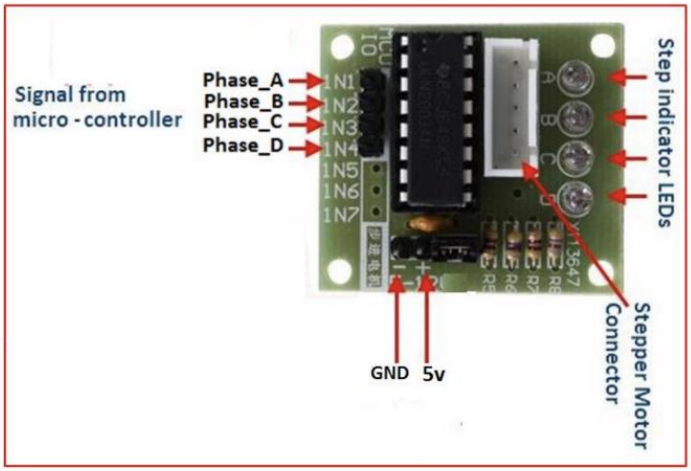
- כאשר המוט במצב המקורי (לא מוטה) המתח במוצא הרגליים Vrx , Vry הוא $\frac{V_{cc}}{2} = \frac{3.3v}{2} = 1.65v$
- בהטיית המוט לכיוון y,x המתחים Vrx , Vry בהתאמה יעלו בצורה יחסית בטווח [1.65v,3.3v]
- בהטיית המוט לכיוון y,-x- המתחים Vrx , Vry בהתאמה ירדו בצורה יחסית בטווח [0,1.65v]
- לחיצה אנכית על ראש מוט ההיגוי מהווה לחיצת לחצן (בחיבור לרגל בקר MSP430 משפחה 2 השתמשו ברגיסטר PxREN לצורך הגדרת הלחצן בקונפיגורציה של Pullup/Pulldown)



את ערכי הג'ויסטיק נדגום על ידי רכיב ה*ADC* ונקבל את הדגימות בהתאם לאיור הבא –



מנוע צעד, זהו מנוע שניתן להפעילו כך שמוט הסיבוב שלו, יסתובב בכל איטרציה בזווית φ הנקראת צעד.
מנוע צעד מהווה עומס, המחובר לכרטיס ממשק המתווך בין הבקר לעומס. מצד אחד נחבר לכרטיס הממשק את ה- MCU המספק מידע בהספק נמוך, מצד שני נחבר את המנוע המהווה עומס וצורך הספק גבוה.
יש צורך לחבר לכרטיס מתח הפעלה של 5v ברגל המיועדת לכך (ראה תצלום הבא).



©Hanan Ribo

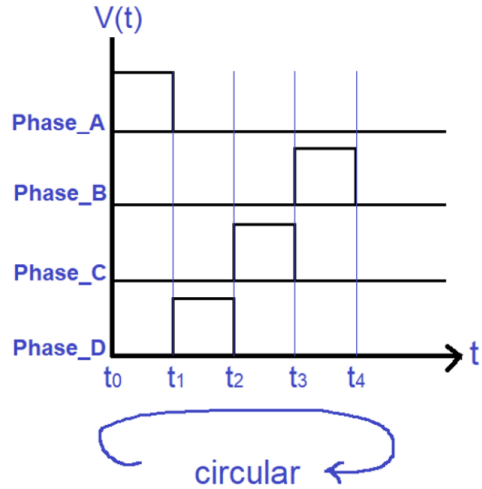
<u>סיבוב המנוע בצעד / חצי צעד :</u>

- <u>סיבוב המנוע בצעד מלא בכיוון clockwise:</u>

כדי להפעיל את המנוע **בצעד בודד** (סיבוב אחד בזווית **נומינאלית** $\varphi = 0.088°$) **בכיוון השעון**, נדרש להכתיב מהבקר (MCU) מתח לארבע רגליים Phase_A, Phase_B, Phase_C, Phase_D <u>לפי הטבלה הבאה:</u>

כדי לבצע הזזה רציפה כרצוננו, נצטרך לבצע את המתואר בטבלה בצורה מחזורית (***בשימוש פסיקות Timer בלבד***). קצב השינוי (בתחום של 5Hz-50Hz) יקבע את מהירות הסיבוב של מוט המנוע.

רגל בכרטיס הממשק המחוברת לבקר	t_0	t_1	t_2	t_3
Phase_A	1	0	0	0
Phase_B	0	0	0	1
Phase_C	0	0	1	0
Phase_D	0	1	0	0



סיבוב המנוע בצעד בכיוון counter clockwise:

כדי להפעיל את המנוע **בצעד בודד** (סיבוב אחד בזווית **נומינאלית** $\varphi = 0.088°$) **בכיוון נגד כוון השעון**,

נדרש להכתיב מהבקר (MCU) מתח לארבע רגליים Phase_A, Phase_B, Phase_C, Phase_D

נפעיל את הטבלה הנ"ל בכיוון הפוך.

רגל בכרטיס הממשק המחוברת לבקר	t_0	t_1	t_2	t_3
Phase_A	0	1	0	0
Phase_B	0	0	1	0
Phase_C	0	0	0	1
Phase_D	1	0	0	0

→ t

FLASH

עבור המשימה הרביעית היה עלינו לממש הפעלת קובץ SCRIPT המכיל פקודות high level שמבצעות את הפעולות הבאות:

OPC (first Byte)	Instruction	Operand (next Bytes)	Explanation
0x01	inc_lcd	x	Count up from zero to **x** with delay d onto LCD
0x02	dec_lcd	x	Count down from **x** to zero with delay d onto LCD
0x03	rra_lcd	x	Rotate **right** onto LCD from pixel index 0 to pixel index 31 a single char **x** (ASCII value) with delay d
0x04	set_delay	d	Set the delay **d** value (***units of 10ms***)
0x05	clear_all_leds		Clear LCD
0x06	stepper_deg	p	Points the stepper motor pointer to degree p and show the degree (dynamically) onto PC screen
0x07	stepper_scan	l,r	Scan area between left l angle to right r angle (once) and show the start and final degrees (right on time the motor pointer reaches them) ***onto LCD screen***
0x08	sleep		Set the MCU into sleep mode

עלינו לתמוך בשליחה וקבלה של עד שלושה קבצים. על מנת לבצע את המשימה עלינו לצרוב את הקבצים לתוך רכיב הFLASH של הבקר. את בחירת הקבצים וצריבתם מימשנו בצד המחשב דרך הGUI בתפריט המתאים למשימה.
בהתחלה נצרוב את כל הקבצים הנדרשים לבקר ולאחר מכן נקבל ACK מהבקר שהצריבה הסתיימה בהצלחה. את הצריבה ביצענו באמצעות מצביע שיצביע לכתובת סגמנט הרלוונטית, ותקדם אותו בכל כתיבת בייט חדש.

עבור הרצת הסקריפטים, נבחר את הסקריפט הרצוי שנרצה להריץ ובעזרת כפתור הExecute ייפתח לנו חלון חדש שבוא נוכל להתחיל להריץ את הסקריפט בעזרת כפתור choose ובבקר נרוץ על ה flash ונמשוך את הערכים שברכיב הפלאש באמצעות מצביע לכתובת הסגמנט הרלוונטית לקובץ שצרוב בו.

GUI

במהלך הפרויקט, היינו צריכים ליישם מגוון רחב של פונקציות, כולל תתי פונקציות רבות. כדי להתמודד עם כמות המידע הגדולה שהמשתמש נדרש לנהל, החלטנו להשתמש ב GUI(ממשק גרפי למשתמש) שיציג את כל האפשרויות והמצבים שהמשתמש צריך לקבל החלטות לגביהם בכל רגע. ה GUI נוצר באמצעות ספריית pySimpleGUI, אשר סיפקה לנו כלים נוחים לעיצוב תצוגות שונות לכל משימה ולניהול שלהן. לכל חלון בתפריט הראשי וגם לכל משימה ותת-משימה, יצרנו תצוגה (layout) הכוללת כפתורים, טקסט ועוד, בהתאם לצרכים של כל חלון משימה. ה GUI-פעל במקביל למודול ה UART-שבצד המחשב, על ידי שליחה וקבלה של מידע מהבקר בכל לחיצה על כפתור, בהתאם לפונקציות שהותאמו.

Painter

במשימה השנייה, שהייתה דורשת ציור באמצעות ג'ויסטיק, היינו צריכים לפתח אפליקציית ציור שתציג את הציורים בזמן אמת. לשם כך השתמשנו בספריית tkinter ליצירת class בשם Painter, המכיל את הפונקציות הנדרשות למשימה. אופן הציור נקבע על פי משתנה גלובלי בשם state,המייצג את מצב הפעולה (כתיבה, מחיקה, ניוטרל), ומשתנה לפי הפקודות שמתקבלות מהבקר . ביצענו ציור על ידי הגדרת הסמן לצבע שחור לכתיבה, לבן למחיקה ובלי צבע למצב ניוטרל.

הערה :מכיוון שה GUI וה Painter הם תהליכים אינסופיים הפועלים בו זמנית, השתמשנו בספריית threading והקצנו לכל אחת מהאפליקציות thread נפרד, כך שהן פועלות במקביל.