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

בית הספר להנדסת חשמל ומחשבים

דו"ח מסכם לפרויקט גמר קורס
"מבנה מחשבים ספרתיים" 361-1-4191

תכנון ומימוש מערכת בקרה למכונה מבוססת מנוע צעד בשליטה ידנית ומרחוק

מיכאל גרינדר 208839845
רועי כסלו 206917064

מדריך אחראי
18.08.2022

תוכן עניינים

מטרת הפרויקט

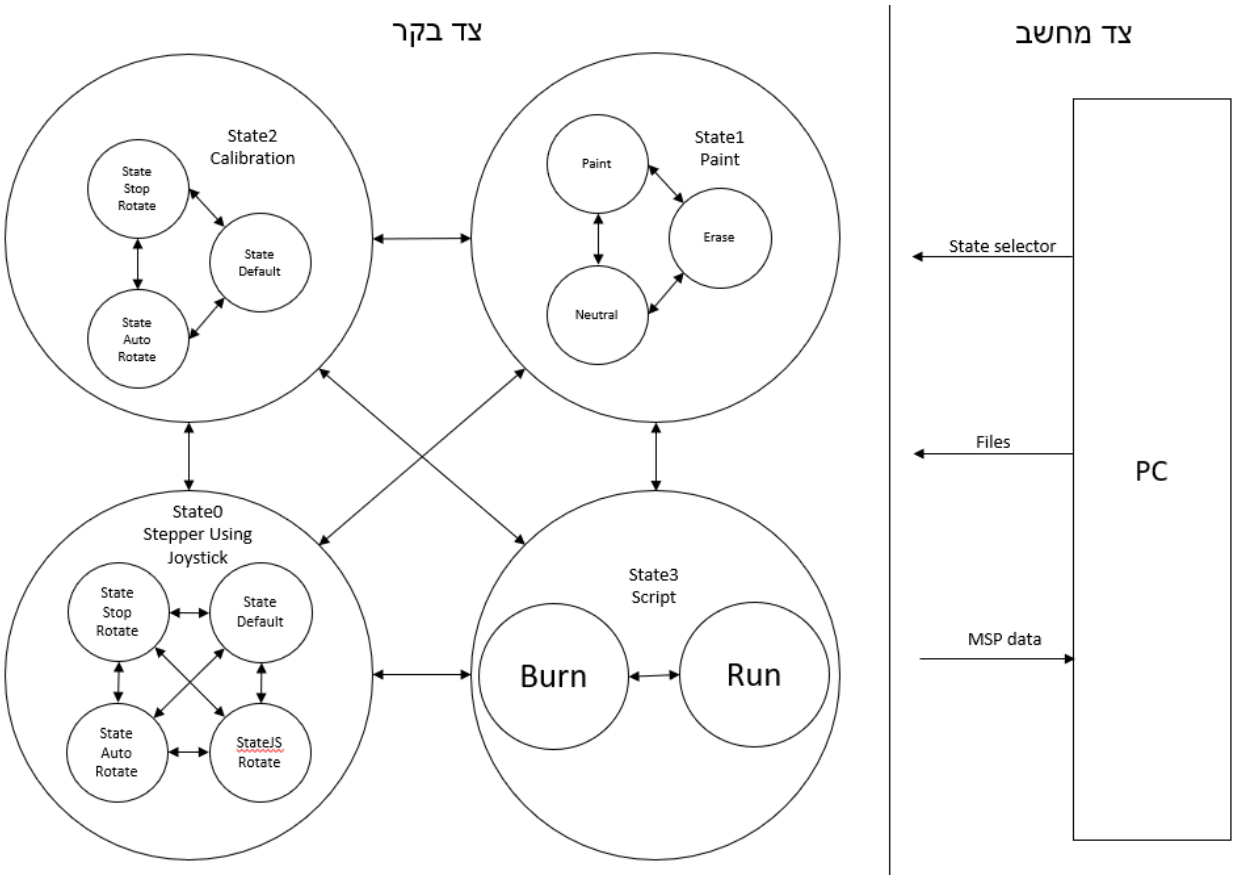
הפרויקט הינו פרויקט הסיכום של הקורס "מבנה מחשבים ספרתיים" אשר מאחד על גבי פרויקט אחד עם מגוון משימות, מגוון רב של ידע שצברנו במהלך הסמסטר וגם במהלך הפרויקט עצמו. במהלכו נדרשנו לממש מערכת בקרה למכונה מבוססת מנוע צעד באמצעות שליטה ידנית על ידי ג'ויסטיק וגם בשליטה מרחוק ממחשב אישי דרך ערוץ תקשורת טורית.

הפרויקט מומש על גבי בקר MSP430G2553 בשפת C בתוכנת CCS אשר יבצע את סט הפעולות שיתואר בהמשך יחד עם ממשק של מחשב אישי שיכול לבוא לידי ביטוי גם בשליטה על הפעולות וגם באמצעות GUI שנותן ממשק בין המשתמש לפעולות שהבקר יעשה.

תיאור הפרויקט

כאמור, במהלך הפרויקט הבקר והמחשב יצטרכו לבצע בקורלציה מספר פקודות, כאשר ארכיטקטורת התוכנה שלהן מבוססת פרדיגמת תכנות מסוג FSM המבצעת קטע קוד השייך לאחד ממצבי המערכת (המתוארים תחת "ביצועי חומרה ותוכנה") בהינתן בקשת פסיקה על ידי המשתמש באמצעות תקשורת UART בין הבקר בPC דרך הGUI.

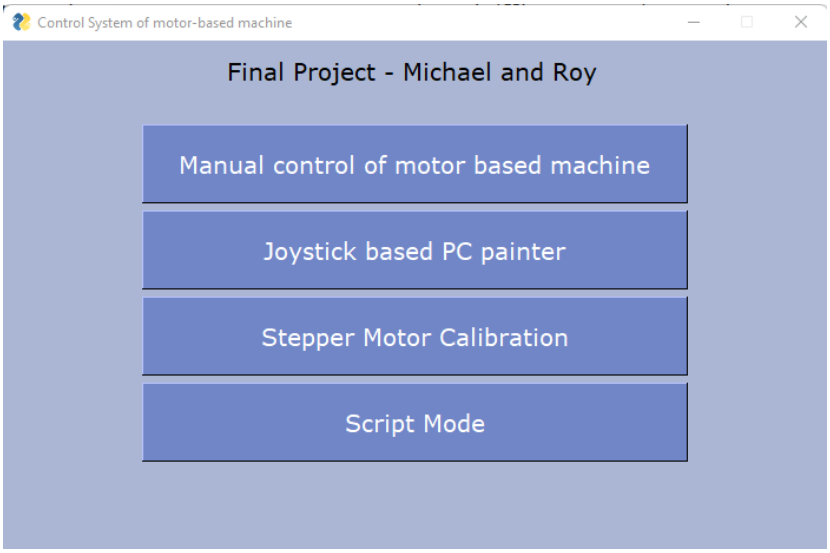
להלן דיאגרמת הFSM –



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.

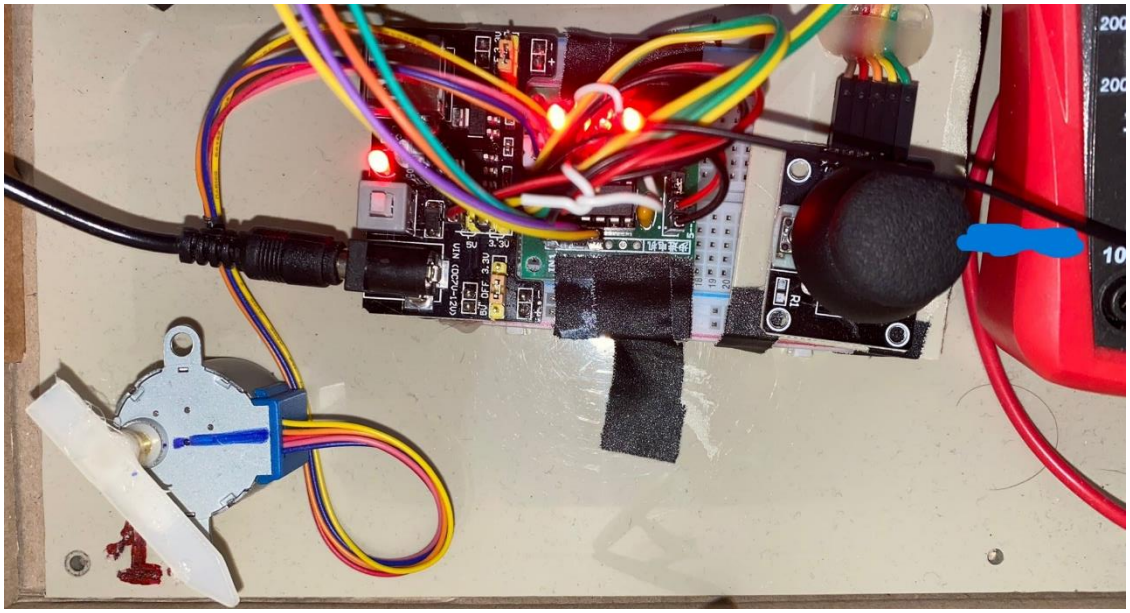
להלן מסך התפריט הראשי –



הערה –צורת התקשורת שהוסברה במשימה זו פועלת באותו האופן בכל המשימות גם אם מסך הGUI שונה.

Manual control of motor-based machine

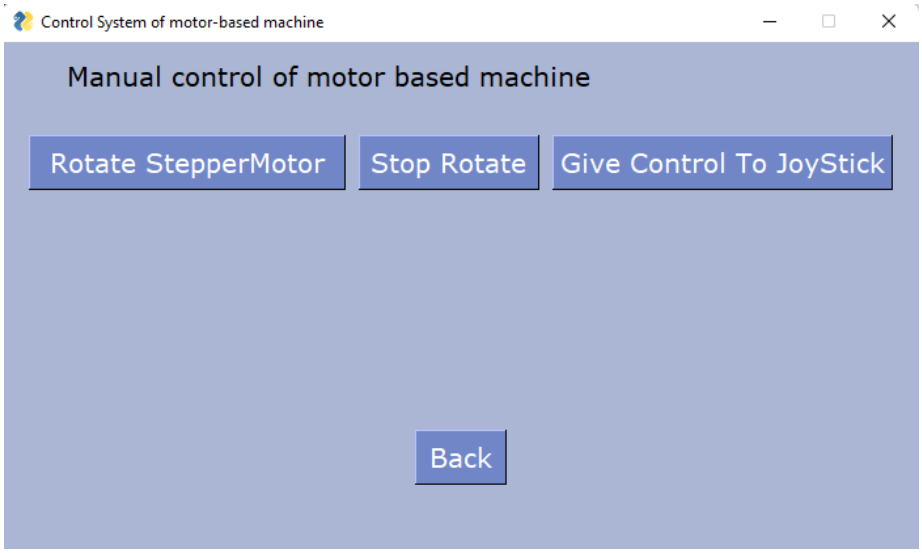
במשימה זו נרצה לשלוט במנוע הצעד על ידי הג'ויסטיק על ידי כך שאם המשתמש יזיז את הג'ויסטיק לזווית φ אזי מנוע הצעד ינוע לזווית φ גם כן. נדגיש כי ציר הφ של כל רכיב חומרתי הוא על פי ההתאמה הבאה –



כאשר φ היא הזווית בין המיקום הנוכחי לבין הקו הכחול מצויר בג'ויסטיק וגם הקו הכחול ב$stepper\ motor$.

נדגיש כי מאחר ומדובר בבקרה בחוג פתוח על המנוע צעד לנוע תחילה אל נקודת ה$\varphi = 0$ (היכן שהפס הכחול סומן) וביחס לכיול זה מנוע הצד ינוע לזווית φ המבוקשת על ידי הג'ויסטיק (ראה פירוט על כיול מנוע הצעד בפרק "Stepper Motor Calibration").

כעת ניתן לפרוט את המשימה להכיל 3 פקודות – הזזת מנוע הצעד על ידי הג'ויסטיק, הזזת מנוע הצעד עם כיוון השעון ועצירתו. לכן, עם הלחיצה בתפריט על ה Manual control of motor-based machine התפריט יתעדכן לתת-תפריט רלוונטי למשימה הזאת המכיל 3 פקודות נוספות –



כעת נצלול לאופן המימוש כאשר נדגיש שבהינתן כיול התחלתי, אנחנו יודעים את כמות הצעדים הנדרשת עבור הStepper motor לביצוע סיבוב שלם, לכן ניתן לגזור מכך את גודל זווית כל צעד על ידי חלוקה ב$360°$.
לכן הגדרנו משתנה בשם $curr\_counter$ המחזיק את הזווית בה הבקר נמצא כרגע ביחידות של כמות הצעדים לסיבוב שלם. לכן אם מסתובבים יחידה אחת עם כיוון השעון ונגד כיוון השעון הוא גדל ב1 וקטן ב1 בהתאמה.

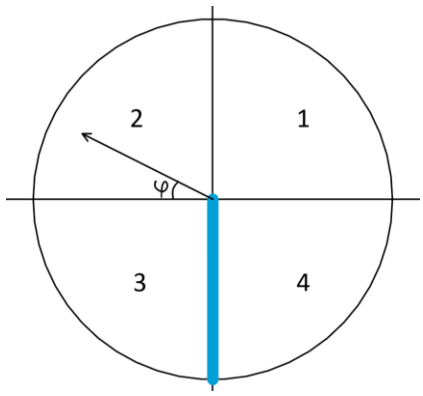
את המשימה הזו, כאמור פיצלנו ל3 תתי משימות ולכן עשינו $switch - case$ נוסף הבורר את הפקודה המתאימה מסט הפקודות שצוין.

הזזת מנוע הצעד 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.

עצירת המנוע – הגדרת הבקר להיות במצב שינה.

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. 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. הערה – כאשר הג'ויסטיק במצב ניטרלי (כלומר באמצע בשני הצירים) הגדרנו שהמנוע יישאר בזווית הנוכחית אליה הגיע.

כעת נותר למצוא את הזווית הרצויה אותה נמצא על ידי דגימה של ערכי רגלי הג'ויסטיק על ידי הADC בשני הצירים ולאחר קבלת הדגימות נבצע חישוב של הזווית על ידי פונקציה $atan2\_fp$ אשר מחשבת את הזווית בהינתן 2 וקטורים ב$fixed\ point$ ובעלת דיוק של $\pm 1°$. נמצא את הזווית הרצויה ביחידות של ה$stepper$ –

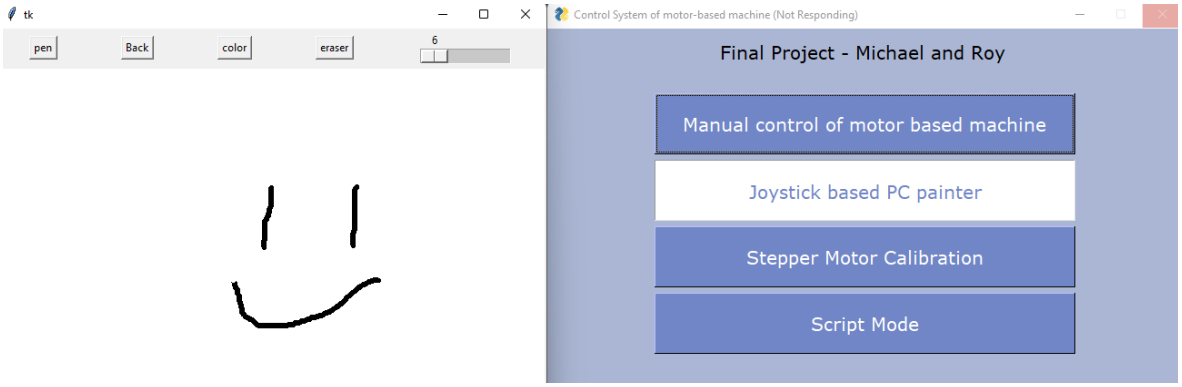


כאשר φ היא הזווית המחושבת על ידי ה$atan2$. נמצא את הזווית הרצויה לפי החישוב הבא –

$$wanted = \begin{cases} \dfrac{270-\varphi}{360}\cdot counter \\ \dfrac{(360+270)-\varphi}{360}\cdot counter \end{cases} = \begin{cases} \dfrac{3}{4}counter - \dfrac{\varphi\cdot counter}{360}, & 1,2,4 \\ \dfrac{7}{4}counter - \dfrac{\varphi\cdot counter}{360}, & 3 \end{cases}$$

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 ומצב ״ניוטרל״ (רק מזיז את העכבר ללא ביצוע פעולה על מסך הצייר עצמו).



מימשנו את האפליקציה בצד מחשב ורצינו שהג׳ויסטיק הנמצא תחת צד הבקר ישלוט על הסמן של הצייר. לשם כך, רצינו לדגום את ערכי רגלי הג׳ויסטיק ולשדרם לצד מחשב בכדי שישנה את הסמן בצייר בהתאם לכיוון שהג׳ויסטיק נע אליו. מפני שרצינו שמשימה זו תתקיים ב$hard\ real\ time$ רצינו שהמידע יעבור בקצב מקסימלי ולכן החלטנו לשדר את ערכי דגימות הג׳ויסטיק בצורה רציפה על ידי הבקר (כמובן בהנחה והוא קיבל הנחיה להיות במצב של הצייר) ללא התערבות הPC. המחשב ידגום גם כן בצורה רציפה את המידע המגיע אליו ויפענח את המידע לפי סדר – ציר X, ציר Y.

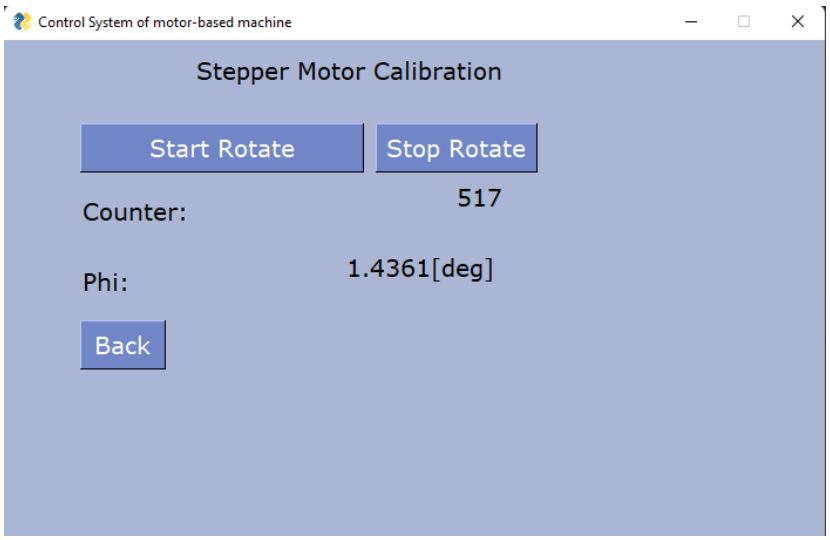
כעת בהינתן הדגימות של הג׳ויסטיק, נממש את הזזת הסמן על אפליקציית הצייר על ידי כך שסמן הצייר זז לפי תזוזת העכבר. מפני שנרצה שהג׳ויסטיק יממש שליטה אוטומטית על העכבר, נמצא את ההפרש בין דגימות עוקבות ונזיז העכבר בהתאם להפרש בשני הצירים.

את ברירת המצבים נקבע לפי משתנה $state$ אשר מאותחל להיות במצב 2 – מצב ניוטרל. מצב הצייר ישתנה לפי לחצן מהערכה על ידי שידור של קוד שלא יכול להיות להגיע על פי הדגימות של הבקר וכאשר המחשב יקבל קוד זה הוא יסיק כי הבקר שלח לו שהתקבלה לחיצה ולכן המחשב יקדם את ערך משתנה המצב שלו (במודולו 3).

הערה – אופן ביצוע כל מצב מפורט בסעיף הגדרות חומרה ותוכנה, *Painter*.

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 (בחרנו עם כיוון השעון, זה לא משנה) ועצירתו כאשר המשתמש מבחין שהמנוע סיים סיבוב שלם מהפס הכחול.
לכן, עם הלחיצה בתפריט על ה Stepper Motor Calibration התפריט יתעדכן לתת-תפריט רלוונטי למשימה הזאת המכיל את 2 הפקודות שפורטו קודם לכן יחד עם הדפסת מספר הצעדים שנעשה בסיבוב שלם וגם הזווית φ של מנוע הצעד שתחושב על פי כמות הצעדים לחלק ל360 מעלות.



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.

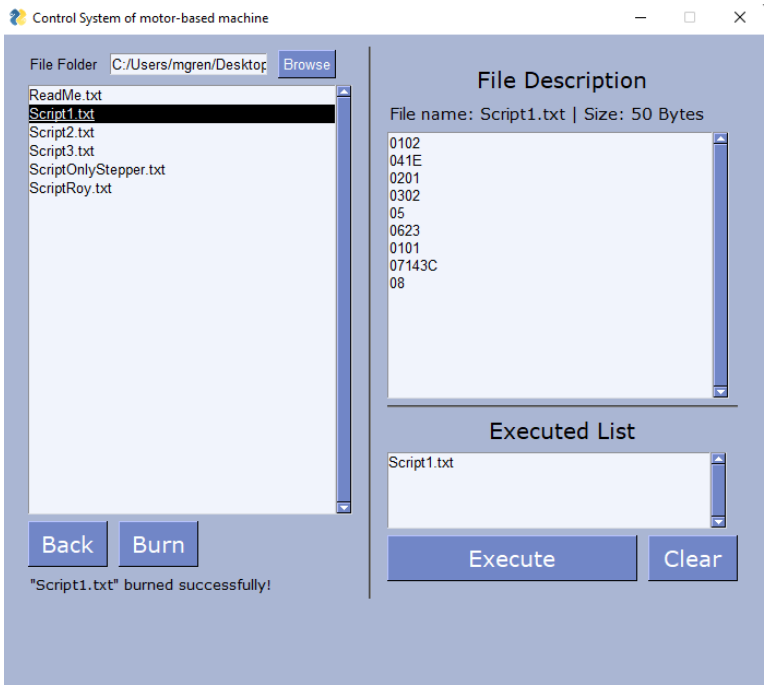
את קבצי הscript נצרוב לרכיב הFLASH הנמצא בבקר. כל סקריפט נצרוב לסגמנט מתאים בפלאש (כאשר הראשון מתחיל בכתובת 0x1000), נזכור כי גודל כל סגמנט הוא 64 בתים וההנחה היא שגודל כל סקריפט לא חורג מ64 בתים. נגדיר משתנה מסוג struct אשר יכיל את השדות הבאים: כמות קבצים קיימים, מערך המכיל את שם הקובץ, מערך מצביע לתחילת כל קובץ ומערך המכיל את גדלי -הקבצים.

את הצריבה של כל קובץ נבצע לפי אינדקס הבחירה מהתפריט בGUI(כלומר נשמור את האינדקס בו הקובץ סקריפט נמצא בתפריט הגלילה) לאחר צריבת הקבצים, נממש כפתור שייקח אותנו לחלון חדש בו נבצע את ריצת הסקריפט שנצרב על הבקר בהתאם לאינדקס שבו הוא נמצא בתפריט הגלילה.

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

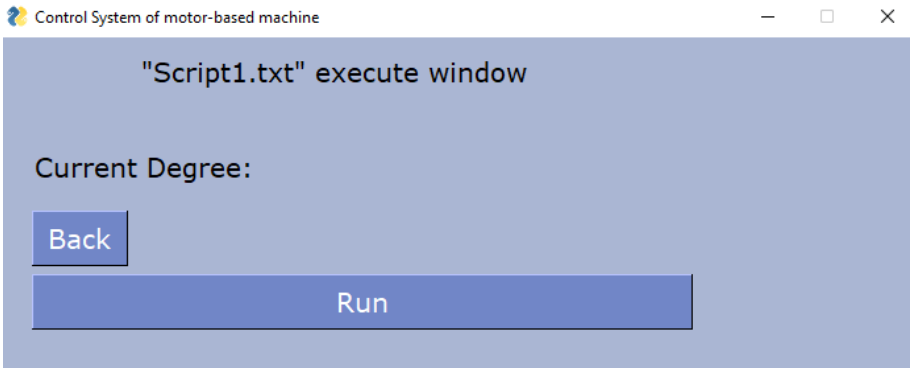
התפריט יכיל בנוסף כפתור הרצה (run) אשר יפתח תת תפריט חדש בו נבצע הרצה של הסקריפט המתאים ונדפיס לתפריט את ערך הזווית אשר נדרש לחלק מהפקודות high level שמבוצעות בסקריפט.

תפריט המשימה –



כאשר בתפריט זה ניתן לראות כי קיים הטקסט – $"Script1.txt"\ burned\ successfully!$, כלומר ניתן לראות כי בעת צילום התפריט הנוכחי בדיוק הסתיימה צריבה של $Script1.txt$ לבקר. ניתן לראות זאת גם על ידי הימצאותו של הקובץ ברשימת ה$Execured\ List$.

עם לחיצה על $Execute$ נפתח חלון חדש –



כאשר עם לחיצה על Run יתחיל ביצוע ה$File$ הרלוונטי (אותו ניתן לראות בכותרת).

הגדרות חומרה ותוכנה

לצורך ביצוע משימות הפרויקט השתמשנו במודולים חומרתיים שונים בצד הבקר ובנוסף לכך רכיבי תוכנה בצד המחשב. נפרט את אופן מימוש המשימות תוך נגיעה קורלציה של החומרה והתוכנה.

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

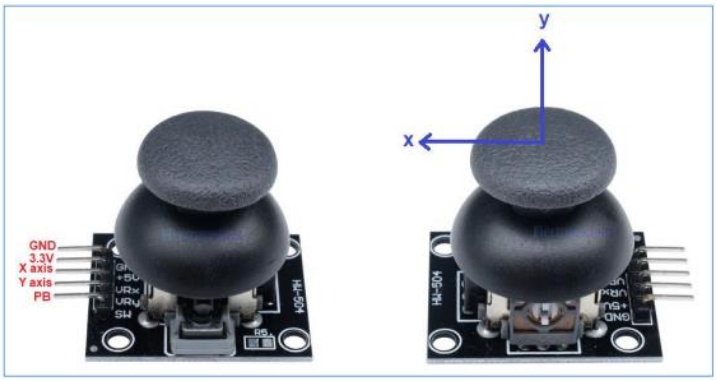
במהלך הפרויקט היינו צריכים לפעול בקורלציה בין הבקר למחשב ולכן השתמשנו בתקשורת UART בניהם עם הפרמטרים – No parity (none) , 1 Stop , 1 Start , 8-bits , 9600 BPS. כעת נסביר על תשתית הUART בכל אחד מהצדדים (צד המחשב וצד הבקר).

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.

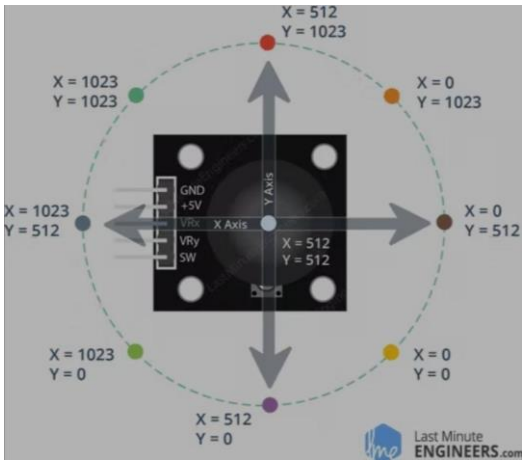
ג׳ויסטיק

רכיב חומרתי זה מקבל הזנת מתח ממקור של $3.3V$ ובעל 3 רגלי יציאה – יציאת ציר X, ציר Y ולחצן.



ערכי יציאות רגלי הצירים יוצאים על ידי חלוקת מתח בצורה ליניארית (בין פוטנציומטרים פנימיים) כך שעבור ערכים חיוביים בכל ציר טווח ערכי המתחים יהיו בתחום $\left[\frac{Vcc}{2}, Vcc\right]$ בעוד שעבור ערכים שליליים טווח ערכי המתחים יהיו בתחום $\left[0, \frac{Vcc}{2}\right]$.

את ערכי הג'ויסטיק נדגום על ידי רכיב הADC ונקבל את הדגימות בהתאם לאיור הבא –



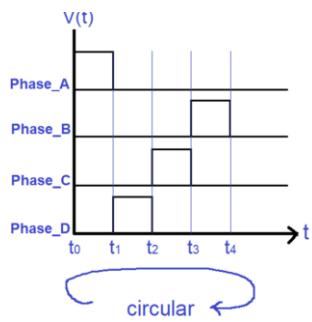
Stepper motor

מנוע הצעד מקבל הזנה על ידי מקור חיצוני של $5V$ ובעל כניסה של 4 פאזות הקובעות את הסיבוב שלו (יוסבר בהמשך). את המנוע ניתן להפעיל על ידי סיבוב מוט הסיבוב שלו על ידי זווית φ אותה נמצא על ידי משימת הכיול. המנוע יכול להסתובב לגודל φ או $\varphi/2$ בכיווניות של $clocwise$ או $counter-clockwise$ על ידי קביעת סדר הפאזות שלו בהתאם לתצורה שתוסבר בהמשך. נדגיש כי מהגבלות מכניות למנוע קיים תדר מקסימלי בין הפאזות שעבור כל תדר גבוה ממנו המנוע מפסיק לעבוד וגם גודל זווית הצעד משתנה בין הרכיבים. התצורות –

סיבוב המנוע בצעד אחד עם כיוון השעון –

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

$\longrightarrow t$



סיבוב המנוע בצעד אחד נגד כיוון השעון –

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

$\longrightarrow t$

סיבוב המנוע בחצי צעד עם כיוון השעון –

רגל בכרטיס הממשק המחוברת לבקר	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
Phase_A	0	0	0	0	0	1	1	1
Phase_B	0	0	0	1	1	1	0	0
Phase_C	0	1	1	1	0	0	0	0
Phase_D	1	1	0	0	0	0	0	1

$\longrightarrow t$

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	blink_rgb	x	Blink RGB LED in series **x** times with delay d
0x02	rlc_leds	x	Rotate **left** circularly a single lighted LED in 8-LED array **x** times with delay d
0x03	rrc_leds	x	Rotate **right** circularly a single lighted LED in 8-LED array **x** times with delay d
0x04	set_delay	d	Set the delay **d** value (***units of 10ms***)
0x05	clear_all_leds		Clear all LEDs (RGB and 16-LED array)
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 pinter reaches them) onto PC screen
0x08	sleep		Set the MCU into sleep mode

Note: The default delay d value is 50 (***units of 10ms***)

עלינו לתמוך בשליחה וקבלה של עד שלושה קבצים. על מנת לבצע את המשימה עלינו לצרוב את הקבצים לתוך רכיב הFLASH של הבקר. ההנחה היא שגודל כל קובץ הוא כגודל כל סגמנט שזה 64 בתים. כתובת הסגמנט הראשונה מתחילה ב0x1000 השנייה ב-0x1040 והשלישית ב-0x1080 (קפיצות של 64 כלומר 40 בהקסה) את ברירת בחירת הקבצים וצריבת מימשנו בצד המחשב דרך הGUI בתפריט המתאים למשימה. בהתחלה נצרוב את כל הקבצים הנדרשים לבקר ולאחר מכן נקבל ACK מהבקר שהצריבה הסתיימה בהצלחה ונדפיס למחשב. 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 RUN ובבקר נרוץ על הבקר ונמשוך את הערכים שברכיב הפלאש באמצעות מצביע לכתובת הסגמנט הרלוונטית לקובץ שצרוב בו.

Script1.txt

```
0102
041E
0201
0302
05
0623
0101
07143C
08
```

Script2.txt

```
0105
0203
0419
0306
05
0600
070A50
07143C
071E32
0628
08
```

Script3.txt

```
0106
044B
0208
0419
0305
05
0628
071E32
07143C
070A50
0600
08
```

```
0x1000  3130 3230 3430 4531 3230 3130
0x100C  3330 3230 3530 3630 3332 3130
0x1018  3130 3730 3431 4333 3830 FFFF
0x1024  FFFF FFFF FFFF FFFF FFFF FFFF
0x1030  FFFF FFFF FFFF FFFF FFFF FFFF
0x103C  FFFF FFFF 3130 3530 3230 3330
0x1048  3430 3931 3330 3630 3530 3630
0x1054  3030 3730 4130 3035 3730 3431
0x1060  4333 3730 4531 3233 3630 3832
0x106C  3830 FFFF FFFF FFFF FFFF FFFF
0x1078  FFFF FFFF FFFF FFFF 3130 3630
0x1084  3430 4234 3230 3830 3430 3931
0x1090  3330 3530 3530 3630 3832 3730
0x109C  4531 3233 3730 3431 4333 3730
0x10A8  4130 3035 3630 3030 3830 FFFF
0x10B4  FFFF FFFF FFFF FFFF FFFF FFFF
```

תוכנה

GUI

במהלך הפרויקט היינו צריכים לממש מספר רב של פונקציות כאשר לחלקן יש תתי פונקציות. כתוצאה מהאינפורמציה הרבה שעל המשתמש לנהל החלטנו להשתמש בGUI אשר יציג את כל המצבים שעל המשתמש להחליט לגביהן בכל רגע נתון. הGUI נעשה על ידי ספריית pySimpleGUI אשר באה עם יכולות נוחות למשתמש ליצור layoutים שונים לכל המשימות ולניהולם בהתאם. לתפריט הראשי ולכל משימה (ותת-משימה) עשינו layout המכיל כפתורים, טקסט וכדומה בהתאם לחלון המשימה הרלוונטית.

הGUI עבד בממשק ישיר עם מודול הUART של צד המחשב על ידי שליחה אל הבקר וקבלת מידע מהבקר עם לחיצה על כל כפתור שהתאמנו את הפונקציונליות הזו.

Painter

לצורך המשימה השנייה במהלכה נדרשנו לצייר באמצעות הג'ויסטיק, היינו צריכים ליצור אפליקציית צייר שעליה נציג את הציור ב$hard\ real\ time$. לשם כך השתמשנו בספריית tkinter ובעזרתה יצרנו class של Painter המכיל פונקציונליות בהתאם למשימה. אופן הציור נקבע על ידי משתנה גלובאלי בשם state המכיל את מצב הפעולה (כלומר כתיבה–מחיקה–ניוטרל), המשתנה עם קבלת פסיקה מהבקר (ראה פרק Joystick based PC painter). מימשנו כתיבה על ידי קביעה של הסמן להיות בצבע שחור, מחיקה כלבן וניוטרל ללא צבע.

הערה – כתוצאה מכך שהGUI הוא פעולה אינסופית שקוראת בו זמנית לPainter שגם הוא אינסופי, השתמשנו בספריית threading אשר הקצנו לכל אחת מהאפליקציות thread שונה וכך מימשנו אותם במקביל.