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

יהונתן ארמא 207938903
יובל יעקב סעיד 206921892

01.09.2024

מטרת הפרויקט

הפרויקט הזה הוא פרויקט סיכום לקורס "מבנה מחשבים ספרתיים", המשלב מגוון משימות ומאגד ידע רב שנצבר במהלך הסמסטר ובמהלך הפרויקט עצמו. במסגרת הפרויקט, נדרשנו לפתח מערכת בקרה למכונה המבוססת על מנוע צעד, אשר נשלטת ידנית באמצעות ג'ויסטיק וגם בשליטה מרחוק ממחשב אישי באמצעות תקשורת טורית.

הפרויקט מומש על גבי בקר מסוג MSP430G2553 בשפת C באמצעות תוכנת CCS, וכולל סט פעולות שיתואר בהמשך, יחד עם ממשק למחשב אישי שמאפשר שליטה בפעולות ומספק GUI שמחבר בין המשתמש לפעולות שהבקר מבצע.

תיאור הפרויקט

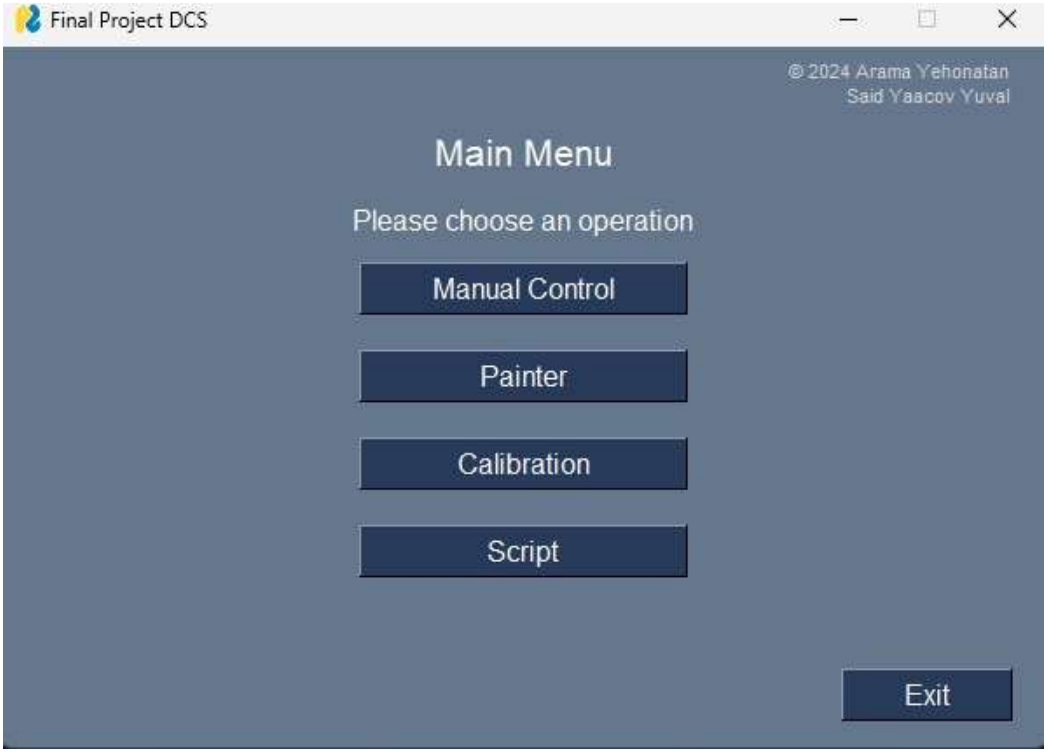
במהלך הפרויקט, המחשב והבקר צריכים לבצע סדרת פקודות בקורלציה משותפת, כאשר הארכיטקטורה התוכנית מבוססת על פרדיגמת תכנות FSM (מכונת מצבים סופית). ה-FSM מבצע קטע קוד השייך לאחד ממצבי המערכת (המפורט תחת "ביצועי חומרה ותוכנה"), בעקבות בקשת פסיקה שנשלחת על ידי המשתמש באמצעות תקשורת UART בין הבקר למחשב דרך ה-GUI.

דיאגרמת ה-FSM מוצגת בנספחים מטה.

ממשק משתמש – בקר

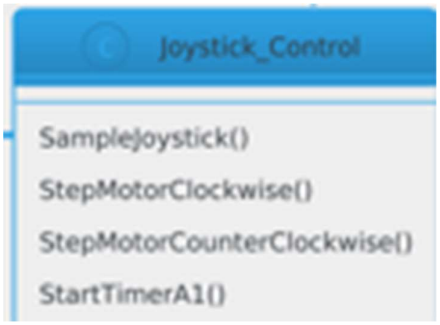
הממשק למשתמש בצד המחשב מתבצע באמצעות GUI המכיל כפתורים, כאשר כל כפתור מייצג פקודה שצריך לבצע. בלחיצה על כל כפתור, המחשב שולח באמצעות תקשורת UART אות המתאים לכפתור הרלוונטי. עם קבלת המידע בבקר, מתבצעת פסיקה והבקר משנה את מצבו ב-FSM, כך שיבצע את הפקודה הרלוונטית שנבחרה על ידי המשתמש.

להלן מסך התפריט הראשי –

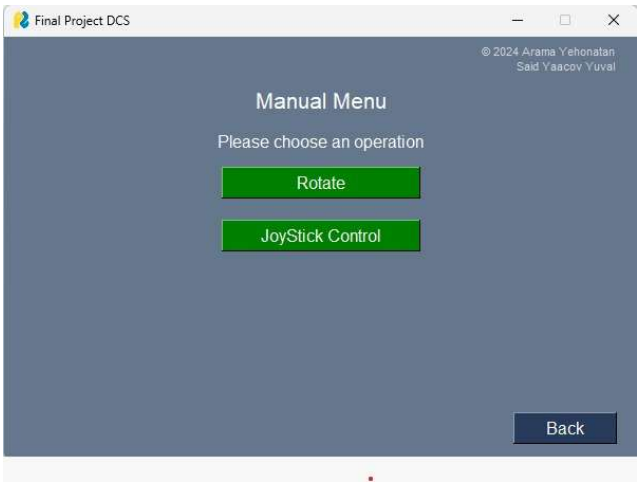


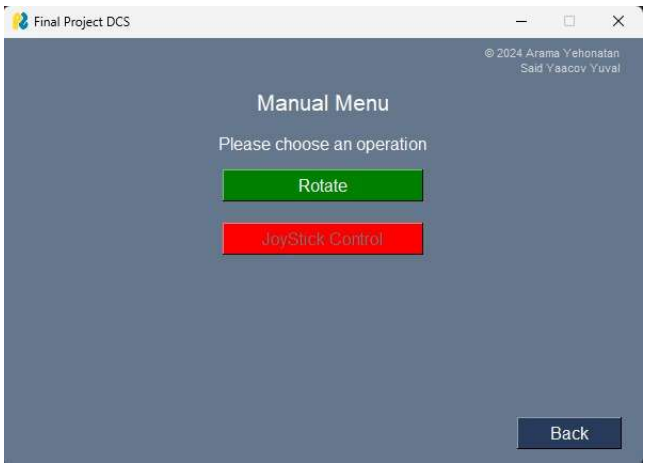
שליטה ידנית במכונה המבוססת על מנוע

במשימה זו, נרצה לשלוט במנוע הצעד באמצעות הג'ויסטיק כך שכאשר המשתמש יזיז את הג'ויסטיק לזווית מסוימת, מנוע הצעד ינוע גם הוא לאותה זווית. חשוב לציין שהציר של כל רכיב חומרה מותאם בהתאם –



המשימה כוללת 4 פקודות עיקריות: דגימת ערכי הג'וייסטיק והמרתם לערך בין 0-1023 ע"י ה -10ADC ומנוקה מרעשים ע"י מיצוע, סיבוב המנוע עם כיוון השעון ונגד כיוון השעון, ולצורך כך נצטרך לקבוע נידלים אשר נקבעים ע"י פקודת ה startTimerA1. כאשר נלחץ על התפריט " Manual control of motor-based machine," הוא יתעדכן לתת-תפריט רלוונטי למשימה זו המכיל 2 פקודות נוספות –





עכשיו נצלול לאופן המימוש: לאחר כיול התחלתי, אנו יודעים את כמות הצעדים הנדרשת למנוע לביצוע סיבוב שלם (N), ולכן ניתן לחשב את גודל זווית כל צעד על ידי חלוקה ל-360°, למעשה $\varphi = \frac{360}{N}$.

הגדרנו משתנה בשם `currentPos`, שמחזיק את הזווית הנוכחית של הבקר ביחידות של כמות הצעדים לסיבוב שלם. כאשר המנוע מסתובב יחידה אחת עם או נגד כיוון השעון, הוא גדל או קטן ב-1 בהתאמה אשר מתאפס בזמן כיול המערכת.

- הזזת מנוע הצעד עם כיוון השעון – המנוע יזוז עם כיוון השעון יחידה אחת, ם והזווית הנוכחית תגדל ב-1.

- הזזת מנוע הצעד נגד כיוון השעון – המנוע יזוז נגד כיוון השעון יחידה אחת, והזווית הנוכחית תקטן ב-1.

Rotate – המנוע מסתובב נגד כיוון השעון עד שלוחצים על מקש ה rotate שוב כדי להפסיק את פעולת המנוע האוטמטי.

- הזזת המנוע לפי הג'ויסטיק – במצב זה, מתבצעת בקרה בחוג פתוח; בכל איטרציה יודעים את הזווית הנוכחית ואת הזווית הרצויה. לאחר בדיקת מקסימום ביניהן, ניתן לקבוע לאיזה כיוון המנוע צריך לנוע: אם הזווית הנוכחית קטנה מהזווית הרצויה, נזוז עם כיוון השעון, אחרת נגד כיוון השעון. במצב ניטרלי (כאשר הג'ויסטיק באמצע בשני הצירים), המנוע יישאר בזווית הנוכחית.

לסיום, נמצא את הזווית הרצויה על ידי דגימת ערכי רגלי הג'ויסטיק באמצעות ה-ADC בשני הצירים, ולאחר קבלת הדגימות נחשב את הזווית בעזרת פונקציית מפה אשר מקבלת את ערכי ה- X,Y שנגדמים לאחר מכן מהכיול בו אנו מקבלים של אמצע הג'וייסטיק נקבל ערכים (נומינליים) של [512,512-] אותם בעצם נחלק ל-8 על גבי ציר קרטזי ובעצם נמיין בכל 45 מעלות(כל שמינית מהציר) להיכן בידיוק שווה הזווית כאשר נחלק ל-15 מקרים (בעצם דיוק של 3 מעלות, נבחין שבכל שמינית החישוב של הזוויות הוא זהה עד כדי מראה – בתכונה זו של מעגל השתמשנו בשביל למחזר קוד) כך שלפי יחס של Y ו-X נקבע לאיזו זווית הם שווים (השתמשנו בפונקציית האש שממירה בין יחס לזווית - לפי חישוב מחשבון של arctag של היחס).

צייר מבוסס ג'ויסטיק למחשב האישי

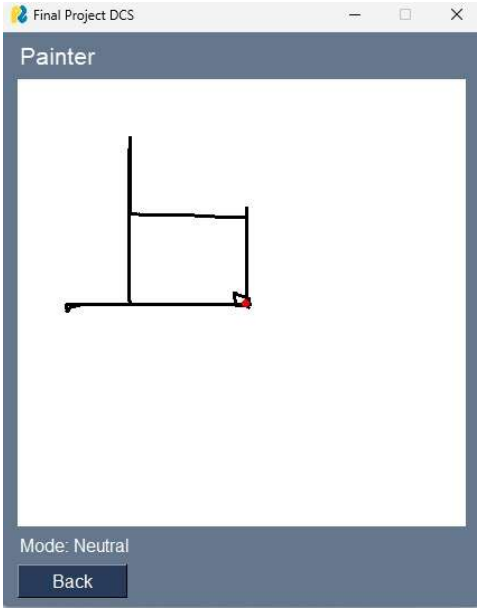
במשימה זו נרצה לממש צייר על מסך המחשב הנשלט על ידי ג'ויסטיק, כשהמשתמש יכול לבחור בין שלושה מצבים: כתיבה, מחיקה ומצב "ניטרלי" (רק הזזת העכבר ללא פעולה בצייר).

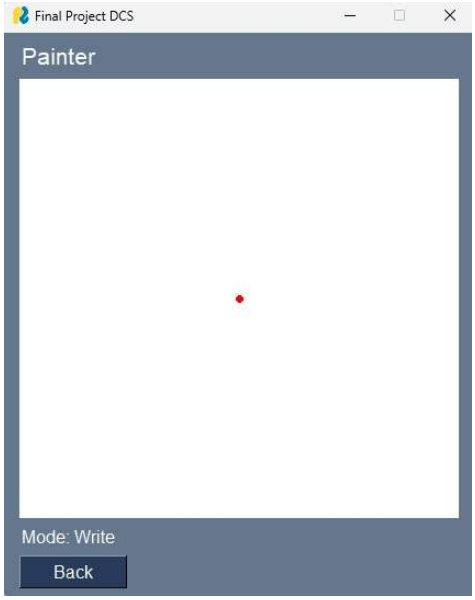
האפליקציה נבנתה בצד המחשב כאשר הג'ויסטיק תחת שליטת הבקר שולט על הסמן של הצייר. לשם כך, נבצע דגימה של ערכי רגלי הג'ויסטיק ונשדר אותם למחשב כדי שהסמן בצייר יזוז בהתאם לכיוון תנועת הג'ויסטיק. משימה זו מתבצעת בזמן אמת (hard real time) ולכן יש להעביר את המידע בקצב מקסימלי, ושידור ערכי הג'ויסטיק יתבצע באופן רציף מצד הבקר (בהנחה שקיבל הנחיה להיות במצב של הצייר) ללא התערבות המחשב.

בנוסף, על מנת להעביר את המידע בקצב מקסימלי ביצענו קידוד של המידע הנשלח. וביצענו שליחה של משתנה מסוג unsigned int ב-2 שליחות של UART.

המחשב ידגום באופן רציף את המידע הנכנס ויפענח אותו לפי סדר: ציר X, ציר Y. הזזת הסמן על אפליקציית הצייר תתבצע בהתאם להפרש בין דגימות עוקבות על שני הצירים.

סוג הצייר יקבע ע"י הלחיצות ויקבעו את מצב הצייר: כתיבה, מחיקה או ניוטרל בהתאם. שמאותחל למצב כתיבה. מצב הצייר ישתנה לפי לחצן מהערכה אל המחשב ואז הוא מסיק שהבקר שלח מידע על לחיצה ומקדם את משתנה המצב שלו (במודולו 3).





כיול מנוע הצעד

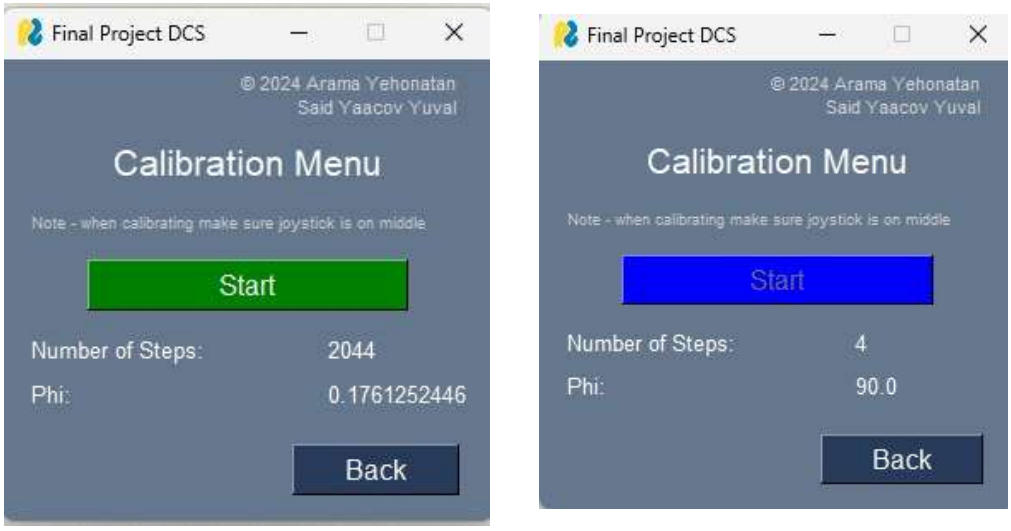


Fig 3: calibration

Right side – During Calibration. Left side After Calibration.

במשימה זו, יש לבצע כיול למנוע הצעד ולהציג על מסך המחשב את כמות הצעדים בסיבוב ואת גודל זווית הצעד של מנוע הצעד. מכיוון שעובדים בבקרה בחוג פתוח, המנוע נדרש לקבל חיווי מהמשתמש על נקודת התחלה שתוגדר כנקודת 0. בנוסף הוספנו מצב בו אנו מחזירים את המתח באיזור הערך של 512 כאשר הג'וייסטיק באמצע(מצב זה עוזר לנו בהזזת המנוע בעזרת הג'וייסטיק ובציור).

המשימה כוללת 2 פקודות: הזזת מנוע הצעד (בחרנו עם כיוון השעון) ועצירתו כשהמשתמש רואה שהמנוע השלים סיבוב שלם מהפס הכחול. בעת לחיצה בתפריט על "Calibration ", התפריט יתעדכן לתת-תפריט למשימה זו עם 2 הפקודות (back) (start), יחד עם הדפסת מספר הצעדים שנעשו בסיבוב שלם וזווית הצעד שתחושב כמות הצעדים לחלק ל-360 מעלות.

מצב Script

במשימה זו, עלינו לבצע פעולות שונות בבקר בהתאם לקובץ script שמכיל פקודות high level שהוגדרו מראש. המשימה כוללת תמיכה בשליחה וקבלה של עד שלושה קבצים, ובחירה להפעיל אחד מהם בנפרד ובאופן בלתי תלוי.

קבצי ה-script יצרבו לרכיב ה-FLASH בבקר. כל script יצרב לסגמנט מתאים ב-FLASH (הראשון מתחיל בכתובת 0x1000), תוך שמירה על כך שגודל כל סגמנט הוא 64 בתים והנחה שגודל כל script אינו חורג מ-64 בתים. נגדיר משתנה מסוג `struct` שיכיל את כמות הקבצים, מערך שמות קבצים, מערך מצביעים לתחילת כל קובץ ומערך גדלי הקבצים.

התפריט שלנו מכיל מצב של חיפוש קובץ במחשב, והצגת תוכנו ב-GUI.

בזמן זה גם מתבצע לו תרגום לקובץ שאותו נשלח לבקר.

השליחה מתבצעת על ידי בחירת Slot לטעינה (מתוך 3 אפשריים) ואז אפשר גם להריץ את הסקריפט על ידי לחיצה כל Execute ב- Slot הרצוי.

התפריט יציג לנו כאשר הקובץ נטען בהצלחה.

יש לציין שבזמן פתיחה של מצב הסקריפט הבקר שולח למחשב איזה סקריפטים טעונים לאיזה סלוטים. ואותם ניתן להריץ ללא טעינה.שמרנו את ההדרים בסגמנט D בכדי שנוכל לדעת מה הנתונים של מה ששמור בפלאש גם אחרי כיבוי המתח ובמצב כניסה לסקריפט אנו קוראים את המידע הזה ומעדכנים את ה-pc אילו סקריפטים שמורים אצלנו כבר בזיכרון כאשר את שאר הסקריפטים אנו שומרים בסקריפטים A-C .

Fig 4: scripts

הגדרות חומרה ותוכנה

לביצוע משימות הפרויקט השתמשנו במודולי חומרה שונים בצד הבקר ובנוסף ברכיבי תוכנה בצד המחשב. נפרט את אופן מימוש המשימות תוך התייחסות לקורלציה בין החומרה לתוכנה.

חומרה

בפרויקט השתמשנו בבקר MSP430 המכיל מודולי חומרה שונים:

ADC10 -

השתמשנו ב-ADC כדי לדגום את ערכי המתח של הג'ויסטיק למשימות השונות. לג'ויסטיק יש 2 צירים ולכן נרצה לבצע דגימות לשני הצירים לשני משתנים שונים

TIMER -

השימוש בטיימר בפרויקט נועד להשהיות בתהליכים שונים. מימוש זה מבוצע על ידי הפעלת הטיימר לפרק הזמן הנדרש והכנסת הבקר למצב שינה מיד לאחר מכן. לאחר פרק הזמן הרצוי, הטיימר ייצור פסיקה שבה הבקר ייצא ממצב שינה וימשיך את התוכנית מהמקום שבו עצר.

UART -

במהלך הפרויקט היינו צריכים לפעול בקורלציה בין הבקר למחשב ולכן השתמשנו בתקשורת UART ביניהם עם הפרמטרים: 9600 BPS, 8-bits, 1 Start, 1 Stop, ללא פריטי.

צד המחשב – בנינו פונקציה ששולחת מחרוזת מהצד המחשב לצד הבקר באמצעות שליחה של אות-אות. בנוסף, בנינו פונקציה שקולטת מידע מהבקר.

צד הבקר – בבקר קיים מודול UART שמקבל פסיקות מסוג RX וגם TX. לכן בצד הבקר ביצענו ISR לכל אחד מסוגי הפסיקות.

- ג'ויסטיק

רכיב חומרה זה מקבל מתח של V3.3 ובעל 3 רגלי יציאה – יציאת ציר X, יציאת ציר Y ויציאת לחיצה. ערכי היציאות מופקים על ידי חלוקת מתח בצורה ליניארית (באמצעות פוטנציומטרים פנימיים), כך שעבור ערכים חיוביים בכל ציר, טווח ערכי המתח יהיה בין [Vcc/2, Vcc] ועבור ערכים שליליים, הטווח יהיה בין [0, Vcc/2].

- Stepper Motor

מנוע הצעד מקבל מתח של V5 ובעל כניסה של 4 פאזות, השולטות בסיבוב שלו (יוסבר בהמשך). את המנוע ניתן להפעיל על ידי סיבוב המוט שלו לזווית ϕ שתמצא במשימת הכיול. המנוע יכול להסתובב לזווית ϕ או ϕ/2 בכיוונים של clockwise או counter-clockwise על ידי קביעת סדר הפאזות שלו בהתאם לתצורה שתוסבר בהמשך. יש לציין שהמנוע מוגבל על ידי תדר מקסימלי בין הפאזות שעבורו כל תדר גבוה ממנו יפסיק את פעולתו וגודל זווית הצעד משתנה בין הרכיבים.

- FLASH

בוצע שימוש בפלאש על מנת להיות מסוגלים לשמור על הקבצים (בדומה ל-FileSystem במחשב ביתי) גם לאחר הורדת מתח.

תוכנה

- GUI

במהלך הפרויקט היינו צריכים לממש מספר רב של פונקציות, שלחלקן יש תתי פונקציות. בעקבות כמות המידע שעל המשתמש לנהל, החלטנו להשתמש ב-GUI שמציג את כל המצבים שעל המשתמש להחליט לגביהם בכל רגע נתון.

ה-GUI נבנה בעזרת ספריית PySimpleGUI שמאפשרת יצירת פריסות שונות לכל משימה וניהולן בהתאם. לתפריט הראשי ולכל משימה (ותת-משימה) עשינו פריסה הכוללת כפתורים, טקסט וכדומה בהתאם לחלון המשימה הרלוונטית. ה-GUI עובד בממשק ישיר עם מודול ה-UART בצד המחשב על ידי שליחה לבקר וקבלת מידע מהבקר עם לחיצה על כל כפתור שהותאמה לו פונקציה זו.

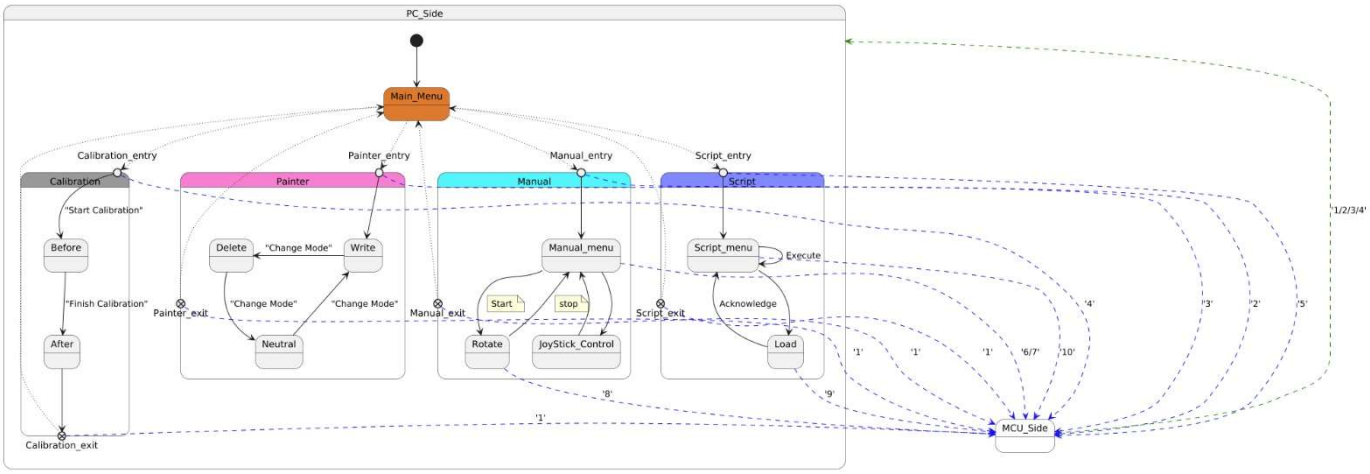
Painter -

אופן הציור נקבע על ידי משתנה גלובלי בשם `state`, המכיל את מצב הפעולה (כתיבה, מחיקה, ניטרלי), שמשתנה עם קבלת פסיקה מהבקר (ראה פרק "צייר מבוסס ג'ויסטיק למחשב האישי"). מימוש הכתיבה נעשה על ידי קביעת הצבע לסמן השחור, המחיקה ללבן, והמצב הניטרלי ללא צבע.

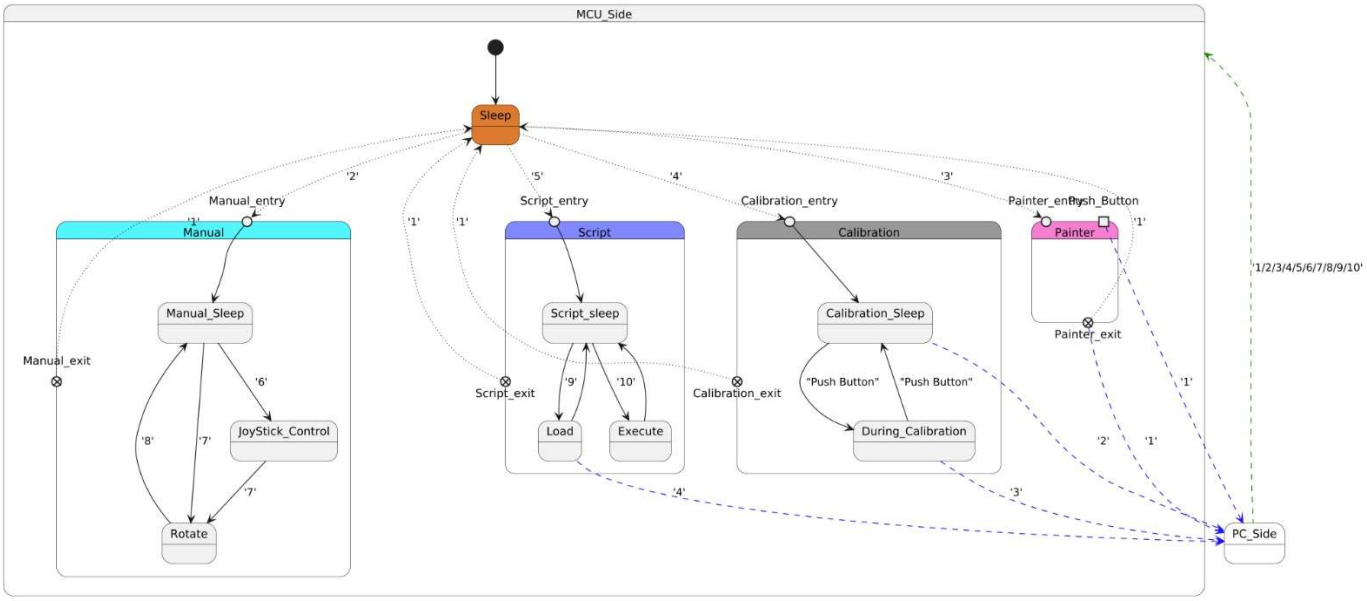
הערה, על מנת שתוכנית הפייתון תרוץ ותוכל לקבל תשדורות UART במקביל לריצת התוכנית ובאופן א-סינכרוני השתמשנו בספריית Threading לביצוע הקליטות.

גרפים ותמונות:

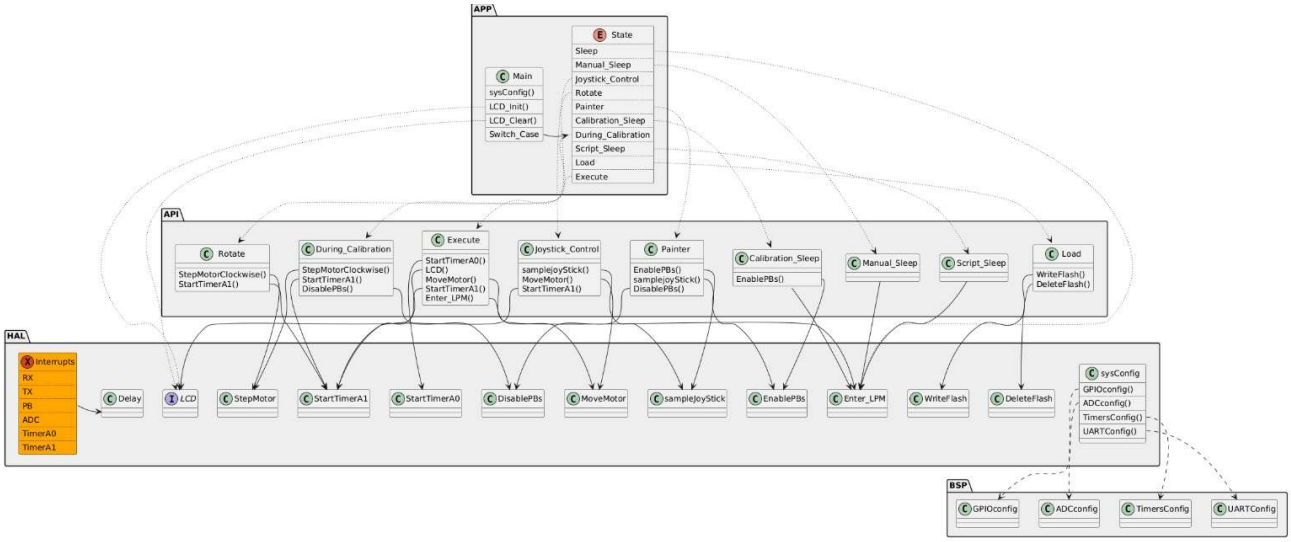
pc side to MCU



MCU to pc side

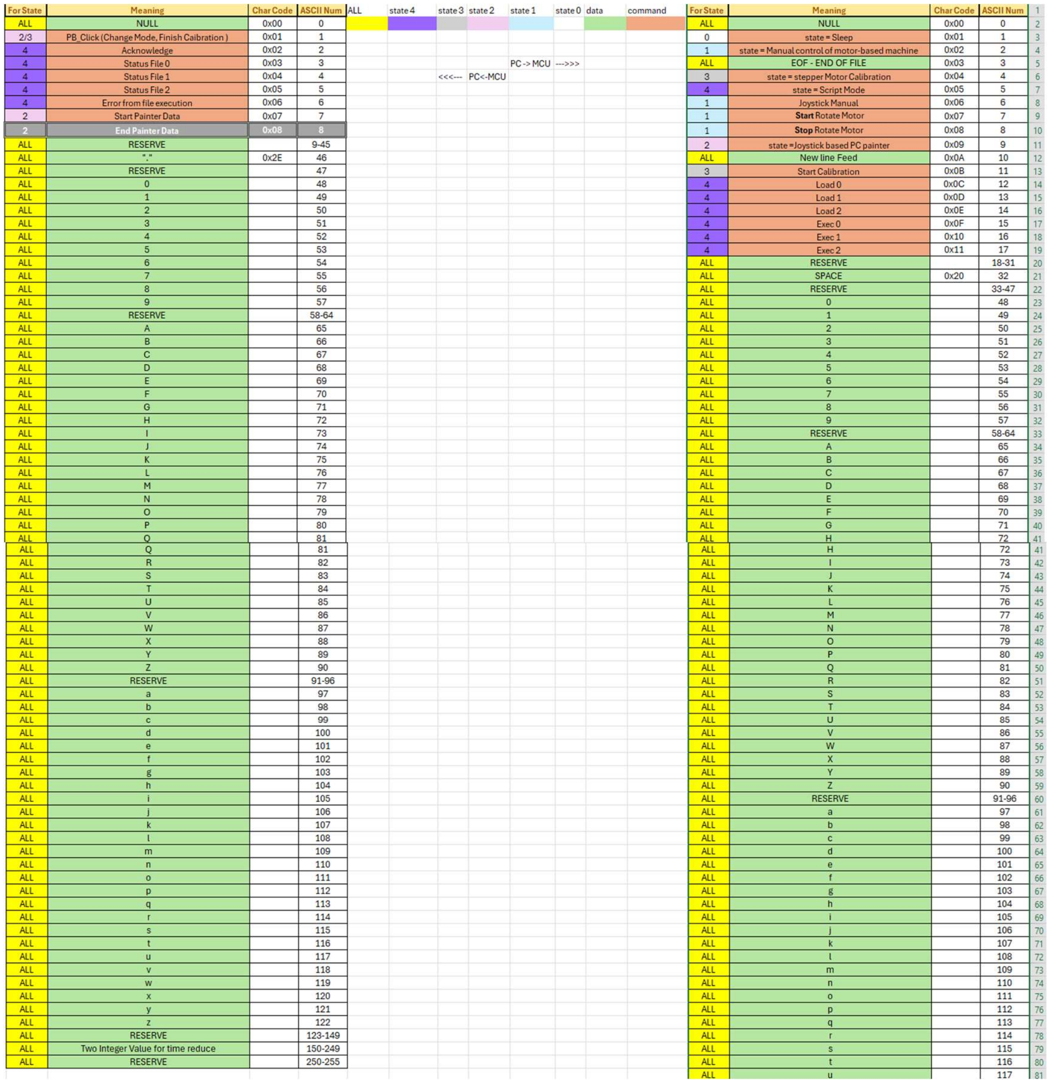


Drivers



הערות	Driver	1
	SysConfig	2
	Enter LMP	3
	RX Interrup Routine	4
	TX Interrup Routine	5
	Push Button Interrup Routine	6
	ADC Interrup Routine	7
	Timer A0 Interrup Routine	8
	Timer A1 Interrup Routine	9
	Delay (for debounce)	10
	EnablePBInt	11
	DisablePBInt	12
מקבל ארגומנט של הערך שבמנוע כרגע, משנה אותו בהתאם	StepMotorClockwise(val)	13
""	StepMotorCounterClockwise(val)	14
מקבל את הזמן לישון במילי שניות (מבצע המרה)	TimerA0Start(Time)	15
מקבל את הזמן לישון במילי שניות (מבצע המרה)	TimerA1Start(Time)	16
מקבל שני ערכים בעזרת ADC	SampleJoystick(Xval , Yval)	17
כמה צעדים עם כיוון השעון (משתמש בפונקציות קודמות)	MotorClockwise(val)	18
כמה צעדים נגד כיוון השעון (משתמש בפונקציות קודמות)	MotorCounterClockwise(val)	19
	LCD Drivers	20
כתיבת תוכן לסגמנט	FlashWrite(Seg,Content)	21
מחיקת סגמנט	FlashDelete(Seg)	22

Codes:



For State	Meaning	Char Code	ASCII Num
ALL	NULL	0x00	0
2/3	PB_Click (Change Mode, Finish Caibration)	0x01	1
4	Acknowledge	0x02	2
4	Status File 0	0x03	3
4	Status File 1	0x04	4
4	Status File 2	0x05	5
4	Error from file execution	0x06	6
2	Start Painter Data	0x07	7
2	**End Painter Data**	**0x08**	**8**
ALL	RESERVE		9-45
ALL	"."	0x2E	46
ALL	RESERVE		47
ALL	0		48
ALL	1		49
ALL	2		50
ALL	3		51
ALL	4		52
ALL	5		53
ALL	6		54
ALL	7		55
ALL	8		56
ALL	9		57
ALL	RESERVE		58-64
ALL	A		65
ALL	B		66
ALL	C		67
ALL	D		68
ALL	E		69
ALL	F		70
ALL	G		71
ALL	H		72
ALL	I		73
ALL	J		74
ALL	K		75
ALL	L		76
ALL	M		77
ALL	N		78
ALL	O		79
ALL	P		80
ALL	Q		81
ALL	Q		81
ALL	R		82
ALL	S		83
ALL	T		84
ALL	U		85
ALL	V		86
ALL	W		87
ALL	X		88
ALL	Y		89
ALL	Z		90
ALL	RESERVE		91-96
ALL	a		97
ALL	b		98
ALL	c		99
ALL	d		100
ALL	e		101
ALL	f		102
ALL	g		103
ALL	h		104
ALL	i		105
ALL	j		106
ALL	k		107
ALL	l		108
ALL	m		109
ALL	n		110
ALL	o		111
ALL	p		112
ALL	q		113
ALL	r		114
ALL	s		115
ALL	t		116
ALL	u		117
ALL	v		118
ALL	w		119
ALL	x		120
ALL	y		121
ALL	z		122
ALL	RESERVE		123-149
ALL	Two Integer Value for time reduce		150-249
ALL	RESERVE		250-255

ALL	state 4	state 3	state 2	state 1	state 0	data	command
🟨	🟪	▨	▨	▨	☐	🟩	🟧

PC -> MCU --->>>

<<<--- PC<-MCU

For State	Meaning	Char Code	ASCII Num
ALL	NULL	0x00	0
0	state = Sleep	0x01	1
1	state = Manual control of motor-based machine	0x02	2
ALL	EOF - END OF FILE	0x03	3
3	state = stepper Motor Calibration	0x04	4
4	state = Script Mode	0x05	5
1	Joystick Manual	0x06	6
1	**Start** Rotate Motor	0x07	7
1	**Stop** Rotate Motor	0x08	8
2	state =Joystick based PC painter	0x09	9
ALL	New line Feed	0x0A	10
3	Start Calibration	0x0B	11
4	Load 0	0x0C	12
4	Load 1	0x0D	13
4	Load 2	0x0E	14
4	Exec 0	0x0F	15
4	Exec 1	0x10	16
4	Exec 2	0x11	17
ALL	RESERVE		18-31
ALL	SPACE	0x20	32
ALL	RESERVE		33-47
ALL	0		48
ALL	1		49
ALL	2		50
ALL	3		51
ALL	4		52
ALL	5		53
ALL	6		54
ALL	7		55
ALL	8		56
ALL	9		57
ALL	RESERVE		58-64
ALL	A		65
ALL	B		66
ALL	C		67
ALL	D		68
ALL	E		69
ALL	F		70
ALL	G		71
ALL	H		72
ALL	H		72
ALL	I		73
ALL	J		74
ALL	K		75
ALL	L		76
ALL	M		77
ALL	N		78
ALL	O		79
ALL	P		80
ALL	Q		81
ALL	R		82
ALL	S		83
ALL	T		84
ALL	U		85
ALL	V		86
ALL	W		87
ALL	X		88
ALL	Y		89
ALL	Z		90
ALL	RESERVE		91-96
ALL	a		97
ALL	b		98
ALL	c		99
ALL	d		100
ALL	e		101
ALL	f		102
ALL	g		103
ALL	h		104
ALL	i		105
ALL	j		106
ALL	k		107
ALL	l		108
ALL	m		109
ALL	n		110
ALL	o		111
ALL	p		112
ALL	q		113
ALL	r		114
ALL	s		115
ALL	t		116
ALL	u		117