



תוכנית הלימודים לתואר ראשון בהנדסת חשמל

שם הפרויקט: מערכת למדידה וניתוח של פולס אקריאו

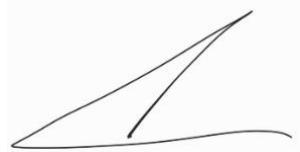
ספר הפרויקט

שם הסטודנט:	קצפר מרקס
שם הסטודנט:	גל בימן
שם המנחה:	מר' רפי גבעון
מציע הרעיון:	מר' רפי גבעון
אישור המנהה על בדיקת המסמך ואישור להגשתה:	אישור המנהה על בדיקת המסמך ואישור להגשתה:
תאריך ההגשה:	בהתאם 4.8.2024

2. אישור המנהה

מאשר להגיש את העבודה.

בברכה,
רפי גבעון



3. תודות

קודם כל נרצה להודות למנהל, מר' רפי גבעון, מציע הרעיון שליווה אותנו לאורך ביצוע הפרויקט, תרם מניסיונו וdag להנחות באזון קשbeta, עם ביקורת בונה ו贊micות לכל שאלה. שנית, נרצה להודות לפROYKTOR מר' אבי רבינוביץ', מר' ניר אלנברג, ומדור הפרויקטים של מכללת אפקה על ההזדמנויות ועל הדאגה לצרכים הלוגיסטיים בדרך לימוש הפרויקט.

4. תוכן עניינים

2.....	2. אישור המנהה.....
3.....	3. מודעות.....
4.....	4. תוכן עניינים.....
5.....	4.1 רשימת אירורים
6.....	4.2 רשימת טבלאות
6.....	4.3 רשימת משוואות
7.....	4.4 רשימת גרפים
8.....	5. תקציר מנהלים / <i>Executive summary</i>
8.....	5.1 תקציר מנהלים
10.....	5.2 Executive summary
12.....	6. הקדמה.....
12.....	6.1 מבוא
12.....	6.2 מיליון מונחים
13.....	6.3 מטרות, יעדים ומדדים
14.....	7. ניתוח חלופות
14.....	7.1 ניתוח חלופות מערכתי
15.....	7.2 ניתוח חלופות טכנולוגיות
22.....	8. סקר ספרות
22.....	8.1 Data Capture Via High-speed ADCs using FPGA
23.....	8.2 High Performance SW Architectures for Remote High-Speed DAQ
25.....	8.3 מודול ADC רכיב בעזרתו נדגום את הפולס האקראי
28.....	9. שיטות.....
28.....	9.1 תוכן ותוכנון המערכת
28.....	9.1.1 דיאגרמת בלוקים
29.....	9.1.2 רכיבים
33.....	9.1.3 תקשורת בין-ADC ל-RPi
35.....	9.1.4 ניהול זיכרון בזמן-אמת
36.....	9.1.5 תכנית ליזוח ועיבוד אות האקראי בזמן-אמת
38.....	9.1.6 חישובים
39.....	9.1.7 ניתוח הספקטרום
41.....	9.1.8 סימולציות
43.....	9.2 חלוקת העבודה בין השותפים
44.....	9.3 תוכנן הפרויקט
44.....	9.3.1 תכנון עבודה סופית
45.....	9.3.2 ריבוע שינויים
47.....	9.3.3 ניהול סיכונים
49.....	10. התוצר
49.....	10.1 פירוט המערכת

50.....	10.2 דוגמאות הרצה
54.....	10.3 בדיקות והערכתה
54	10.3.1 בדיקת דגימה ועיבוד
57	10.3.2 בדיקת איזות הדוחוי
59.....	7.7/11
59.....	11.1 הערכה כללית בהתקנות הפרויקט
59.....	11.2 ניתוח הבחרות הטכניות והמדועיות
60.....	11.3 השוואת התוצאות לתיאוריה
61.....	11.4 שיפורים ולקחים
61	11.4.1 שיפורים טכנולוגיים
61	11.4.2 לקחים מהפרויקט
62.....	11.5 הפטונציאל המסחרי והישומים העתידיים
63.....	12. סיכון ומסקנות
65.....	13. רשימת מקורות
67.....	14. נספחים
67.....	14.1 נספח א' – (Secondary Memory Interface) SMI
72.....	14.2 נספח ב' – (Direct Memory Access) DMA
79.....	14.3 נספח ג' – (C Library) FFTW
81.....	14.4 נספח ד' – פוסטר הפרויקט
82.....	14.5 נספח ה' – הוראות לשימוש במערכת
85.....	14.6 נספח ו' – קוד שימוש עבור ה-4.0 Teensy
86.....	14.7 נספח ז' – קוד שימוש עבור ה-RPi
98.....	14.8 נספח ח' – קוד שימוש לSIMULICITY Matlab

4.1 רשימת איורים

8.....	AIOR 4.0 :5.1 .Teensy 4.0 :
8.....	AIOR 5.2 .AD9226 Module :
9.....	AIOR 5.3 .Raspberry Pi 4 Model B :
10.....	Fig 5.4: Teensy 4.0.
10.....	Fig 5.5: AD9226 Module.
11.....	Fig 5.6: Raspberry Pi 4 Model B.
15.....	AIOR 7.1: המחשה של גנרטור פונקציות ומיקרו בקרה.
19.....	AIOR 7.2: המחשה של PCB של LPF.
22.....	AIOR 8.1: דיאגרמת בלוקים של ממשק תקשורת בין AD9222 לBIN FPGA
23.....	AIOR 8.2: דיאגרמת זמן של ההמרה עבור AD9222.
25.....	AIOR 8.3: דיאגרמה של ערוץ תקשורת המנצל שני בקרים DMA.

איור 8.4: דיאגרמת זמן של המירה עבורי ADS825.	26
איור 8.5: דיאגרמת זמן של המירה עבורי AD9226.	27
איור 9.1: בחירת מוצא WWM של ה-Teensy 4.0.	29
איור 9.2: המחשה של המירה בין חוטי נחושת לכבול SMA.	30
איור 9.3: המחשה של מסנן LPF עם תדר קיטוען של 10 MHz.	30
איור 9.4: המחשה של ה-Module AD9226 עם פירוט על החיבורים.	31
איור 9.5: אופן החיווט בין ה-AD9226 ל-RPi.	32
איור 9.6: קוויים זמינים ב-RPi עבור ה-SMI.	33
איור 9.7: פירוט קווי SMI ב-0/I של ה-RPi.	34
איור 9.8: דיאגרמת בלוקים הממחישה את פעולה המנגנונים השונים באlgorigthm.	36
איור 9.9: מבט 3D על ה-Layout של ה-PCB.	48
איור 10.1: צילום המערכת כולה (מבט על).	49

4.2 רישימת טבלאות

טבלה 7.1: מפרט של חלופות מערכתיות הקיימות בשוק.	14
טבלה 7.2: מפרט של מיקרו-בקרים במקור לפולסים.	16
טבלה 7.3: דירוג של מיקרו-בקרים במקור לפולסים.	16
טבלה 7.4: מפרט של ממיר D/A.	17
טבלה 7.5: דירוג של ממיר D/A.	18
טבלה 7.6: פירוט של ברטיסי FPGA שיכולים לשמש לצורכי DAQ-I.	19
טבלה 7.7: דירוג של ברטיסי FPGA שיכולים לשמש לצורכי DAQ-I.	20
טבלה 7.8: פירוט של מיקרו-בקרים שיכולים לשמש לצורכי DAQ-I.	20
טבלה 7.9: דירוג של מיקרו-בקרים שיכולים לשמש לצורכי DAQ-I.	21
טבלה 8.1: השוואה בין מערכות DAQ שונות.	24
טבלה 8.2: פירוט והשוואה של ארכיטקטורות ADC.	25
טבלה 9.1: חלוקת העבודה בין השותפים.	43
טבלה 9.2:GANUT העבודה על הפרויקט גמר.	44
טבלה 9.3: טבלת ריכוז שינויים שבוצעו בפרויקט.	45
טבלה 10.1: מדדים עברו פולס ברוחב סמ' 5.	55
טבלה 10.2: מדדים עברו פולס ברוחב סמ' 1.	56
טבלה 10.3: מדדים עברו פולס ברוחב סמ' 000.	56

4.3 רישימת משוואות

משוואה 9: חישוב זמן המחוור של AOS במונחי פעימות שעון.	38
--	----

38.....	משוואה 9.2: חישוב תדר הדגימה המדויק.
38.....	משוואה 9.3: חישוב תדר Nyquist לדגימה נבונה.
39.....	משוואה 9.4: חישוב רזולוציה בתדר.
39.....	משוואה 9.5: חישוב רוחב פולס מקסימלי.
40.....	משוואה 9.6: התמרת DFT.
40.....	משוואה 9.7: שינוי קנה מידת של האמפליטודות.
40.....	משוואה 9.8: חישוב ההספק של כל וכיב תדר.
41.....	משוואה 9.9: חישוב שונות המתח (תרמי) המתפתח על עומס לפי מודל A-J.
41.....	משוואה 10.9: חישוב רזולוציות הדגימה של ADC.

4.4 רישימת גרפים

18.....	גרף 7.1: הדגמה של תגובה לתדר של LPF מסחרי.
27.....	גרף 8.1: צילום ספקטרום של אות בתדר ZHM 31 הנקלט ב-AD9226 MADP היוצר.
41.....	גרף 9.1: תוצאת סימולציה עבור דיזה של פולס אקראי מלכני.
42.....	גרף 9.2: תוצאת סימולציה עבור דיזה של פולס אקראי מלכני בנסיבות רעש תרמי ורעש קוונטייזציה.
42.....	גרף 9.3: תוצאת סימולציה עבור דיזה של פולס אקראי מלכני חלש בנסיבות רעש תרמי ורעש קוונטייזציה.
44.....	גרף 9.4: גאנט העבודה על הפרויקט גמר.
46.....	גרף 9.5: גרף תגובה לתדר של מסנן LPF ללא מתאים.
46.....	גרף 9.6: גרף תגובה לתדר של מסנן LPF מתאים.
51.....	גרף 10.1: פולס ברוחב Δf , (1)-(2) ה- R_{Pi} , (4)-(3) מכשיiri מדידה ו-(6)-(5) סימולציה.
52.....	גרף 10.2: פולס ברוחב Δf , (1)-(2) ה- R_{Pi} , (4)-(3) מכשיiri מדידה ו-(6)-(5) סימולציה.
53.....	גרף 10.3: פולס ברוחב Δf , (2)-(1) ה- R_{Pi} , (4)-(3) מכשיiri מדידה ו-(6)-(5) סימולציה.
54.....	גרף 10.4: פרמטרי בדיקה למרחב התדר.
55.....	גרף 10.5: פרמטרי בדיקה למרחב הזמן.
57.....	גרף 10.6: (1) פולס ייחד עם jitter לא ידוע, (2) מרחב התדר של הפולס בנסיבות jitter.
58.....	גרף 10.7: הצגה של סף דיזה הוואדי של פולס אקראי.

5. תקציר מנהלים / Executive summary

5.1 תקציר מנהלים

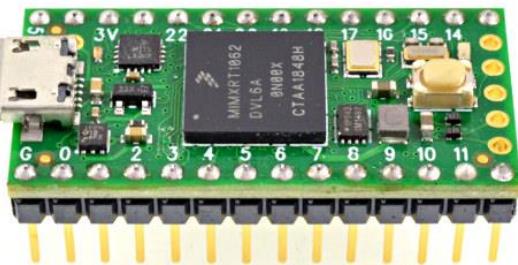
הרעיון לפרויקט הינו רעיון של המנחה רפואי גבעון, הצורך בפרויקט הotecג לנו בעזרת דוגמא מן התעשייה בה היו צרכים לאבחן פולסים אקראים אשר נוצרו מקרדיות אבק או לביר אשר נמצא על דיסק ויכלו לגרום לו לתקן. את האבק או הלביר ניתן להציג כפולס אקראי אשר צריך לקלוט אותו, לאבחן אותו, באשר הכל מתרחש בזמן אמיתי.

המערכת אינה המצאה חדשה אלא בכלל, הוזלה והקינה של מכשירי מדידה (לדוגמה Spectrum Analyzer, Oscilloscope) אשר קיימים כבר בשוק. מכשירים אלה משוכליים מאוד אך גם יקרים ומסורבלים מכיוון שאינם מיועדים לצורך הספציפי הזה אלא לשימושים רבים נוספים. במקרה מהעובדת על הפרויקט חקרונו את הדרכיהם הייעילות ביותר מבחינת חסכון במקום, תקציב מצומצם ומהירות עבודה לייצור אותן ולניטוחו.

הדרישות אשר הצבנו לעצמנו הן:

- הפקת פולס אקראי ברוחב של μs 1 בזורה חומרתית.
- תכנון מערכת אשר תקלוט, תנתח ותמיר את הפולס למשור התדר בזמן אמיתי.
- הצגת הפולס והתמרתו למשתמש בצורה אינטואטיבית.

לאחר מחשבה ושיקולי תכנון ובאים, החלנו להשתמש בマイקו-בקר (Teensy 4.0) לשם ייצור הפולס, על מנת ליצור את האות האקראי בצורה האינטואטיבית והפשטה ביותר.



.איור 5.1 : Teensy 4.0.

לפנינו שאננו מעבירים את האות ל-ADC (Analog to Digital Converter) נרצה להעבירו דרך מסנן - Low Pass אשר ישתמש אותו כAAF (Anti-Aliasing Filter) ויבטיח לנו שהאות יגיע ל-ADC בצורה נכונה ועם מינימום עיוותים. בפרט, האות נדגם ע"י ה-ADC (מודול AD9226) בעל 12 סיביות חולוציה שיוביל לפולטים בקצב ממוצע MSPS 65. הרכיב יעביר דגימה בודדת במצב המקביל שלו בכל ירידת שעון שתובנה לו. את קצב השעון יקבע הרכיב הבא ויעביר אותו למודול.



.איור 5.2 : AD9226 Module .

לקלייטה, עיבוד אותות והציג הנתונים בחרכו להשתמש במקירו בקר B. Rasperry Pi 4 Model B. רכיב זמין וזול שמציע מגננים מתאימים שיירשו לו לעמוד בדרישות המהירות של זיהוי פולס וניתוח בזמן-אמת. רכיב זה יכול לעמוד בכל הדרישות שלנו הן מבחינות יכולות עיבוד והן מבחינות זיכרון אשר יסייע לנו לשמר את הדגימות אשר הוא מקבל מה-ADC.



.איור 5.3 Raspberry Pi 4 Model B :

בסוף העבודה על הפרויקט אנחנו מצפים לקבל מערכת אחת שלמה המבילה את כל תת-המערכות אשר דיברנו עליהן, כל תת-מערכת תעמוד ביעדים אשר הצבענו והמערכת עצמה תהיה יכולה לשאר המערכות בשוק, קלת משקל ופשטה לשימוש.

Executive summary 5.2

The idea for the project comes from the supervisor, Rafi Givon. The need for the project was presented to us through an industry example where there was a need to diagnose random pulses generated by dust or dirt found on a disk that could cause damage. The dust or dirt can be represented as a random pulse that needs to be captured, diagnosed, and all this happening in real-time. The system is not an innovative invention but an enhancement, cost reduction, and miniaturization of existing measurement devices (e.g. Spectrum Analyzer, Oscilloscope) already available in the market. These devices are very sophisticated but also expensive and cumbersome as they are not designed for this specific need but for many other applications. As part of the project work, we researched the most efficient ways in terms of space-saving, limited budget, and work speed to generate and analyze the signal.

The requirements we set to ourselves are:

- Generating a random pulse of $1 \mu\text{s}$ using hardware.
- Designing a real-time system that will capture, analyze and transform the pulse into frequency domain.
- Presenting the pulse and its transformation to the user in high quality manner.

After much thought and many design considerations, we decided to use the Teensy 4.0 microcontroller for generating the pulse, to produce the random signal in the simplest and highest quality way possible.



Fig 5.4: Teensy 4.0.

Before the signal reaches the ADC (Analog to Digital Converter), we want to pass it through a Low-Pass filter that will serve as an Anti-Aliasing Filter (AAF) and ensure that the signal reaches the ADC correctly and with minimal distortions. Now, the signal is sampled by the ADC (AD9226 module) which has a 12-bit resolution and can output a maximum of 65 MSPS. The module will transmit a single sample at its parallel output at each clock drop provided to it.

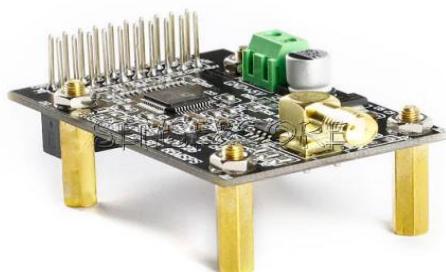


Fig 5.5: AD9226 Module.

For determining the sample rate, storing the samples, processing them, and displaying the data, we chose to use the Raspberry Pi 4 Model B microcontroller, a readily available and inexpensive component that offers suitable mechanisms that allow it to meet the speed requirements of pulse detection and analysis in real-time. This component can meet all our requirements both in terms of processing capabilities and memory which will assist us in preserving samples it receives from the ADC.



Fig 5.6: Raspberry Pi 4 Model B.

At the end of the project work, we expect to receive one complete system containing all the subsystems we discussed, each subsystem will meet the goals we set, and the system itself will be relatively cheap compared to other systems in the market, lightweight, and easy to use.

6. הקדמה

6.1 מבוא

בפרויקט זה מטרתנו לבנות מערכת אשר מייצרת אות חשמלי אקרוי, קולטת אותו, מבצעת עיבוד שלו ומציגה למשתמש ספקטרום איבוטי של האות במרחב התדר. המבנה יעשה באופן חומרתי, בשילוב של מערכות אנלוגיות ודיגיטליות הפועלות בתדר גובה מאוד (Very High Frequency). שלבי התכנון והביצוע ישלבו תחומים רבים מעולם החישמל, בעיקר מבנה מחשבים, עיבוד אותות ותקשורת.

שני החלקים של הפרויקט (בנייה המכשיר המייצר את האות האקרוי, ובנית מכשיר המדידה והציגה) בהגדرتם נבנאים לרשות מכשירים ספקטросקופים ומכלולאות, לאחר ומדובר ביצירה ומדידה של אותות חשמליים. כמובן, קיימים מכשירים רבים בשוק אשר מבצעים פעולה זהה למערכות שלנו, אך חום איןם נגישים למשתמש הממוצע בשל המחיר הגבוה, חשוב לציין שמדובר במקרה אחד העובודה והחומרה המקובצת. לצד העמידה בדרישות להן אנו מתחייבים בפרויקט זה, אנו רוצים לצמצם את העליות הכספיות בשלב של בחירת הרכיבים, כך שהפרויקט יוכל להיות מודל ל모ץ זול הזמן למשתמש הממוצע.

חשוב לציין כי בחרנו בפרויקט זה מותך רצון להביא לידי ביטוי את הבישורים השונים שרכשו במהלך התואר בתחוםים השונים (עיבוד אותות, תקשורת וחומרה מבוסנה), ולהראות בשור תכני ויצועי בבנייה מערכות מתאגרות. לאחר ומערכות זו בשמה אינה מהויה חדש טכנולוגי, מטרתנו העיקרית היא לנתח את המוצרים שדמים ברגע בשוק, לבדוק את הצדדים הפחות ייעילים שלהם, לתכנן את המערכת שלנו ביעלה יותר מהמורים הדמים, ואת החלקים הייעלים בהם, לעל עוד יותר.

6.2 מילון מונחים

- **Spectrum Analyzer** – מכשיר מעבדה למדידת ספקטרום התדרים של אותות. הרוב המוחלט היום הוא דיגיטלי אך ניתן לפגוש אנלוגיים. רענוןית הוא מודד אותות לא יזועים.
- **Oscilloscope** – מכשיר מעבדה למדידת אותות בזמן. יתכן מכשירים אנלוגיים/DIGITAL.
- **מיקרו-בקר** – מערכת מחשב קטנה ועצמאית המיועדת לבצע משימות ספציפיות ושלוט במכשירים אלקטרוניים. מיקרו בקרים הם בעצם מעגלים משלבים (IC) המשלבים מיקרו-מעבד (CPU), זיכרון, ציוד היקפי קלט/פלט (O/I) ורכיבים חיווניים אחרים לשכב אחד. מיקרו-בקרים נמצאים בשימוש נרחב ביישומים שונים, לרבות מוצרי אלקטронיקה, אוטומציה תעשייתית, רובוטיקה, מכשור רפואי, מערכות רכב ועוד. הם מספקים יכולות אינטיליגנציה גבוהה למכשירים אלו, ומאפשרים להם לבצע פונקציות ספציפיות או להגיב לבניוסות וגירויים שונים.
- **ממיר D/A** – דוגמ או ממיר אנלוגי לדיגיטלי, הוא התקן אלקטронיקי המבצע המרת של אות אנלוגי לאות דיגיטלי. דוגמה נפוצה לשימוש בדוגמ היא בהמרת אות אנלוגי שנקלט על ידי מיקרופון שמייצג קול, לאות דיגיטלי אשר ניתן לעיבוד וشمירה באמצעות ממוחשבים. המרת ברוכבה בקומוניציה של המידע הנכנס ולן בהברחה הוא מוסיף אלמנט של שגיאה. בנוסף, במקרים מסוימים ניתן לבצע רציפהADC, מבחן המרת ממיר A/D, ז"א ממיר דיגיטלי לאנלוגי.
- **FFT** – Fast-Fourier-Transform היא אלגוריתםיעיל לחישוב התמרת פורייה בדידה (DFT, Discrete-Fourier-Transform) וההתמרה ההופכית שלה.
- **מסנן LPF (אנלוגי)** – מדובר במסנן אנלוגי המעביר תדרים נמוכים (Low-Pass-Filter) והוא בעל יישומים רבים במגוון של מערכות. הוא יכול להיות מושם בمعالgi המרת DC-L-AC, DC-AC-L.

מערכות עיבוד אותות אנalogיות ובמיוחד במקלטי ומשדרי RF. יש סוגים רבים של מסנני LPF, כאשר מעבר לוחב הסרט, הן מתאפיינות ברמת הנichות, שטיחיות בסיס העובר, ליניאריות הפעזה, וכו'.

- **AAF** – Anti-Aliasing-Filter – הגדירה פונקציונליות בלבד עבור המשן LPF בטור מערכת דוגמת. הגדירה אומרת שمسנן זה מונע העברת של תדרים גבוהים מתדר הדגימה, לאחר והחידירה שלהם עלולה להתחזות תחת תדרים נמוכים ולשבש את המדייה שנלקחת ע"י המערכת.
- **PWM** – ראשי תיבות של Pulse Width Modulation בולם אפנון רוחב פולס. PWM זהו טכנית המשמשת לשיליטה בכמות הכוח המועברת למושיר אלקטרוני על ידי פירוק את הכוח לחלקים נפרדים. PWM ברוך בהדלקה ובכוי של הכוח למושיר בתדר גבוה, כאשר השונות של משך זמן ה-"הפעלה" לעומת זמן ה-"כיבוי" קובעת את כמות הכוח הנמסה.
- **GPIO** – ראשי תיבות של General Purpose Input/Output. GPIO זהה תכונה שנמצאת במיקרו-בקרים רבים ובמחשבים עםلوح ייחד המאפשרת להם אינטראקציה עם העולם החיצון על ידי שליטה באמצעות דיגיטליים (פלט) או קריית אוטות דיגיטליים (קלט). פיני GPIO יכולים להיות מוגדרים ככניות או יציאות וכיתן להשתמש בהם כדי להתmesh עם מושירים חיצוניים שונים כגון חיישנים, מצלמים, נוריות, מנועים ועוד.
- **מרוחב הזמן** – האופי בו רמת המתח משתנה בציר הזמן.
- **מרוחב התדר** – האופי בו רמת המתח משתנה בציר התדר.
- **MSPS** – ממד לבמות של מגה (10⁶) דגימות בשניה.

6.3 מטרות, יעדים ומדדדים המטרות והיעדים:

1. הפקת פולס חשמלי ברוחב בסדר גודל של [s] μ 1 באמצעות חומרה אנalogית / דיגיטלי.
2. תכנון והטמעה של מערכת ניתוח האות, אשר תבצע התמרת פורייה ותנתה את התוצאות.
3. תצוגה של הפרמטרים הרלוונטיים אשר נמדדדו.

מדדדים:

1. **יצור האות** – הפולס האקריא יהיה באורך של לא יותר מאשר 1 μ s.
2. **מערכת העיבוד** – תזהה בזמן אמיתי פולס חשמלי שמנגין בנקודה אקראית בזמן. למטרת הדיווי נדרש יכולת דגימה גבוהה מספיק, וכך לבצע זאת בצורה אינטואטיבית אנו מתכוונים לדגם מעלה הסטנדרט המקביל, לפחות MSPS 20. לצד זה, אנו נרצה לספק dynamic Range פרקטן של מתחים (-5/+5) עם רוחלチה גדולה ככל האפשר.
3. **הציג** – נציג את צורת הפולס אותה הצלחנו לקלוט במשורר הזמן ומישור התדר. בנוסף, נציג לתונינים רלוונטיים כגון עצמה מקסימלית, רוחב הפולס, וכו'.

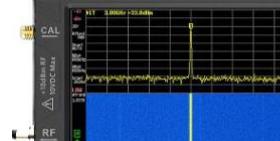
משמעותה של מערכת שלנו תיצור את האות, תנתה אותו ותציג אותו, נוכל לוודא את קיומם המדדים שלנו. מערכת הדיווי תהיה זו שתodium שהפולס נוצר במתוכן, והנתונים המוצגים למשתמש על הצג יהיו הוכחה לאיכות הדיווי והעיבוד.

7. ביטוח חלופות

7.1 ניתוח חלופות מערכתי

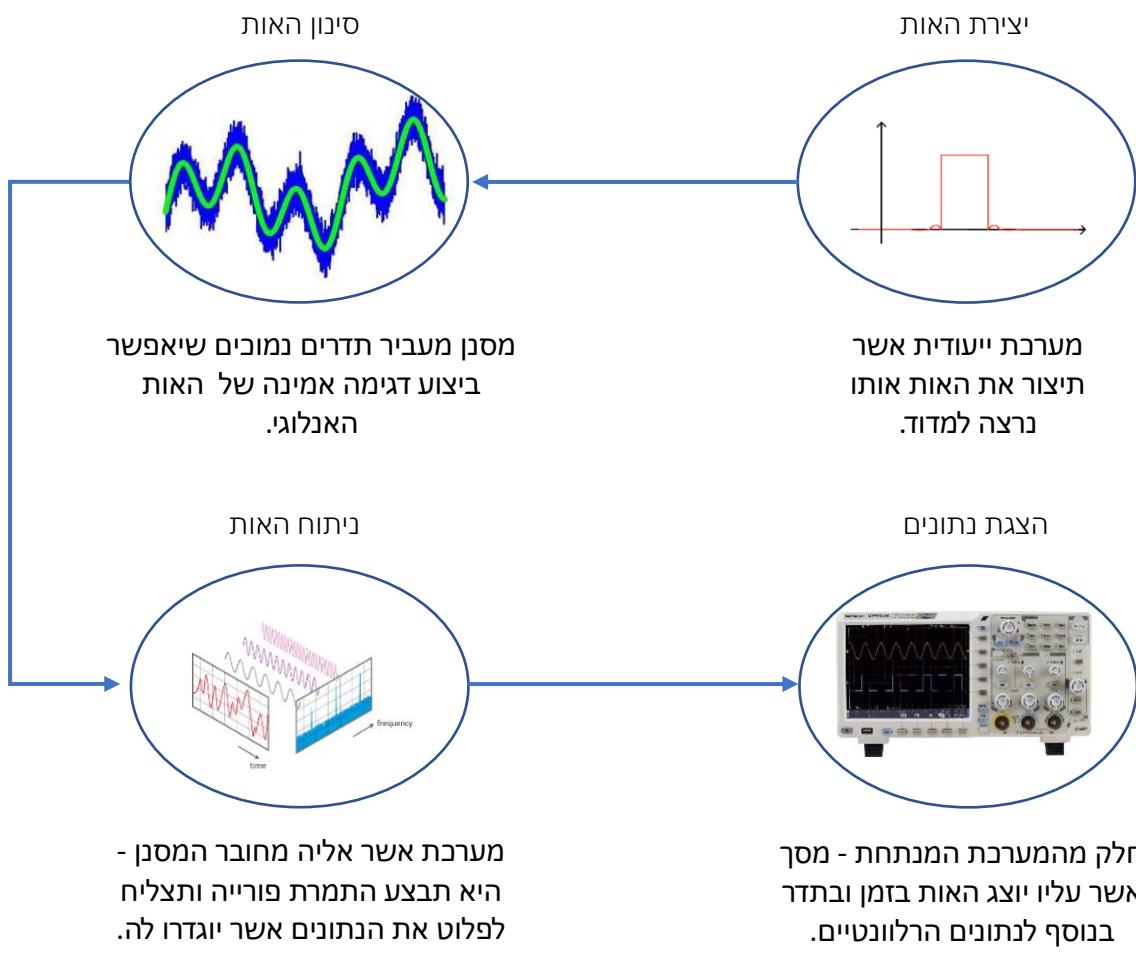
בבלה 7.1 מופיעות דוגמאות של מכשירי מדידה כמו Spectrum Analyzer's ו-Oscilloscope's דינמיים בשוק יחד עם פרמטרים מההפרט הטכני שלהם שלנו לשם השוואה. חלק מן המכשירים יקרים משמעויות ואינם אטרקטיביים במחיר עבור המסתמן המוצע או חובבן. הפקציונליות של כל אחד מן המכשירים דומה מאוד לו שאנו ממשיכם בפרויקט, וזה בניתו אוטומטית וידועים, ביחד ל-VNA (Vector Network Analyzer) שמנתח אוטומטית. המסר העיקרי שאנו רצים להביע דרך הבלה הוא שגובה המחוור מתלווה בעיקר ברוחב הشرط, רגולציה בתדר, מדר הדגימה והחסינות לרעיש.

עבלה 7.1: מפרט של חלופות מערכתיות הקיימות בשוק.

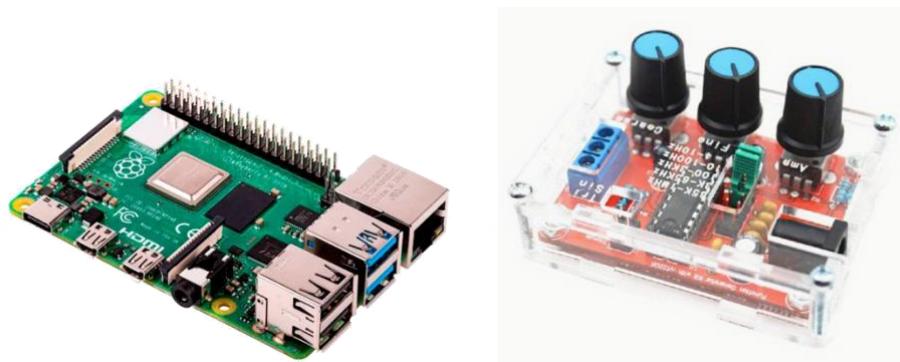
יצן וסוג	המשה תפקידו	המשה תפקידו	המשה תפקידו	המשה תפקידו
רשותם	תואר המתא	מחיר	המשה	המשה בולטים
Keysight (InfiniiVision 3000T X-Series)	Rigol (DSA705)	Triplet (Wi-Fi Hound™)	Kkmoon (TinySA Ultra)	
Oscilloscope and Spectrum Analyzer	Spectrum Analyzer	Spectrum Analyzer	Spectrum Analyzer	
רוחב פס של בין 100 .350 MHz[Hz] M-ל[Hz] M. מהירות דגימה של למעלה ל- 5.5 GSPS. בין 2 ל- 4 ערוצים אנלוגיים. רזולוציית תצוגה גבוהה מאוד. בעל פונקציות מתמטיות וברוט מובנות בצדורה נוחה על גבי צג המכשיר. בעל חיבורים רבים מסוג USB ו-LAN.	טוווח תדרים מ- 100 .500 kHz[Hz] k עד [Hz] 1 Hz[Hz]. רזולוציה של התדר 1 Hz[Hz]. טוווח מדידת אמפליטודה +20 dBm[ms] עד -100 dBm[ms]. קצב רענון של בין 10 ms[ms] ל-שניה. בעל חיבוריו USB ו-LAN. קומפקטי וקל במשקל דריש כיוול תקופתי	מכסה טוווח תדרים של 2.4 MHz[Hz]-G[Hz] 5, מסך תצוגה של 10 אינץ', מציג אותות אלחוטיים, מודד עצמות בין [dBm] 95- 0 dBm[ms] ברזולוציה של 0.5 ms[ms]. משתמשים בו בעיקר ליתוח ספקטרום וה-Audio. לצורך תחזוק Wifi ואופטימיזציה של רשותות אלחוטיות	טוווח תדרים של למעלה מ-6 GHz[Hz] מסך תצוגה בגודל 4 אינץ', הגישות מצוינות עם רעש ממוצע של -89-dBm[ms], בעל מסנני רזולוציה בטוווח של [Hz] 850 – 200. מציג סדרה של 450 נקודות על המסך, בעל קצב רענון של 165 s[ms], נייד וקומפקטי	
4,877\$	1,050\$	600\$	195\$	
				
איכות וקצב דגימה גבוהה מאוד, ערוצים רבים, שמאפשרים חישובים, דופרנציאלים ביןיהם, פעולות מתמטיות רבות	טוווח גודל של אמפליטודות כניסה, צירבת הספק קטנה, יכולת ניתוח רוחב פס גדול	מציג תוצאות בזמן אמיתי, יכולת סינון רעשים גבוהים מאוד, מכסה טוווח תדרים גדול, תצוגה נוחה וחווית משתמש.	ニイ, בעל טוווח תדרים רחב, קצב רענון מסך גבוה.	

7.2 ניתוח חלופות טכנולוגיות

תרשים ארבעת חלקיו הפרויקט:



בניתוח השיטות שביצענו במסמך הייזום אנו הפרדנו אותו לשני שלבים, לשיטות היוצרות פולס אקראי אותו נרצה למדוד ושיטות ליישום מערכת המבצעת עיבוד אותות והציג למשתמש. לאחר הציגת האלטרנטיבות האפשריות, התרשםנו מדרכי הפתרון האפשריות והחלטנו לצמצם את מגוון האופציות לדחק אחת של פתרון. קודם כל הייתה בפנינו אפשרות למשתמש ממקור הפולס האקראי באמצעות מיקרו-בלקרים, או גנרטורים של פונקציות (Function Generators) המופיעים באור 7.1.



איור 7.1: המachine של גנרטור פונקציות ומיקרו בקר.

שני סוגי הרכיבים מתאימים לייצור האות, אך הם בעלי תכונות שונות, ובהתאם אליהן אנו קיבלנו את ההחלטה עם איזה שיטה המשיך. המיקרו-בקרים בעלי יחידות PWM, DAC וכדומה, ומאפשרים לנו שליטה מאוד מדויקת על אות אנלוגי שאנו מפיקים מאות ה-GPIO. התממשקות שלנו עם הרכיב היא בעיקר בתכונות מה שעושה אותו מודיע לשלוניים בתרד האות והAMPLITUDE. בונסף לכך, מדר השעון הגבוה של הרוב המוחלט של המיקרו-בקרים, מאפשר רצולציה מאוד גבוהה בהפקת אות אנלוגי במקומות DAC. בטבלה 7.2 נעשה פירוט ודירוג בין מספר מיקרו-בקרים מסווג זה.

טבלה 7.2: מפרט של מיקרו-בקרים במקור לפולסים.

Arduino UNO	Teensy 4.0	STM32F103C8T6 (Blue Pill)	ESP32	מאפיין
16 MHz	600 MHz	72 MHz	Up to 240 MHz	מהירות שעון
16-bit (Timer1)	32-bit	Up to 32-bit	High-resolution timers	Timer Resolution
Arduino IDE	Arduino IDE with Teensyduino	STM32CubeIDE, Arduino IDE	ESP-IDF, Arduino IDE	התממשקות
Possible with direct manipulation of timer registers	Easily generates 1us pulses with precision	Capable of generating 1us pulses with detailed control	Well-suited for generating 1us pulses with precision	יכולת יצירת פולס
Wide library support, beginner-friendly	High performance, extensive memory and peripherals	Balance between performance and cost, detailed hardware control	Integrated Wi-Fi and Bluetooth, suitable for IoT applications	מאפיינים מיוחדים
\$20-\$25	\$20-\$30	\$2-\$5	\$5-\$10	מחיר

טבלה 7.3: דירוג של מיקרו-בקרים במקור לפולסים.

Arduino UNO	Teensy 4.0	STM32F103C8T6 (Blue Pill)	ESP32	מאפיין
2	10	6	8	מהירות שעון
3	9	7	8	Timer Resolution
8	8	6	7	התממשקות
4	10	7	9	יכולת יצירת הפולס
7	7	9	8	מחיר
4.8	8.8	7.0	8.0	משקל

טבלאות 7.2 ו-7.3 מראות את המגון בתדרי השעון, רוחancies הטימרים והמחירים של מיקרו-בקרים שנמצאים בטווח התקציב שלנו. אנו שמים לב שה-Uno Arduino הוא מיקרו בקר זמין וחסית שאפשר ליצור ליצר פולס [8][9]. הוא בעל IDE (Integrated Development Environment) נוח מאוד לשימוש, ומספר ערכיו PWM ו-DAC, אך מאוחר ותדר השעון הראשי הוא 16 MHz אז רוחב פולס צר מאוד מתקבל על ידי מניפולציה ישירה של אוגרי זיכרון. למעשה אנו מעמידים את ה-Blue Pill [10] שדורש תכנון זהיר כדי להפיק פולס ברוחב של מיקרו-שנייה.

מצד שני ישנים מיקרו-בקרים המיועדים למטרות של העברת דיגיטלית מהירה - PWM צרים במיוחד. יכולות אלה באות על חשבון במות הפריפרויות של המיקרו-בקר, מכאן המחיר הזול יחסית. מיקרו-בקרים מסוג זה הם למשל סדרת ה-Teensy 3.0/4.0/4.1 [7] וה-ESP32 [11].

השיטה נוספת לייצור הווידאו מייצרי פונקציות. הם יכולים להופיע בתור מכשירים מורכבים המתפרטים על מגוון רחב של צורות גל, תחום תדרים ואפיו אפנוניים. כאשר אנו מדברים על רכיבים בודדים הם מוגבלים בדרך כלל למספר פונקציות מוגבל בגון גל סינוס, מרובע ומשולש. שיפורים נוספים יכולים להיות מושפעות במחיה. מייצרי הפונקציות בתור רכיבים בודדים נשלטים באמצעות פוטנציאומטרים והם מפיקים אותן רציפים. כדי להפיק פולס בודד של צורת גל מסוימת (כפי שציינו לפני כן), ניתן לחבר מעגל ממוגן במאז Function Generator, וכך לקבל פולס בודד במצבה המרתקת. השיטה הזאת יכולה להיות עילאה עבור הפרויקט שלנו, אך לצערנו רוב מייצרי הפונקציות במחירים זולים, מוגבלים ברוחב סרט של 1 MHz בלבד [21].

בעת בעבר לשיטה בה ניתן לאגד את הפלט המופק באמצעות שיצינו לעיל, והמסקנה שמלווה אותנו עוד משלב כתביה מסמך היוזם היא שעילנו להשתמש ברכיב ADC "יעודי" לדגימה מהירה שיתקשר עם חומרה הקולעת את הדגימה. הסיבה שאנו פונים לפתרון מסוג זה הופיע לא מעט בסקר הספרות שעשינו בפרויקט, ובו ראיינו כי כל המערכות DAQ המtabססות על מחירים יחסית זולים, מתmeshקים עם ממיר נתונים חיצוני. בטבלאות 7.4 ו-7.5 נעשו פירוט ודירוג בין מספר מmirי נתונים אנלוגיים לדיגיטליים.

טבלה 7.4: מפרט של מmirי D/A.

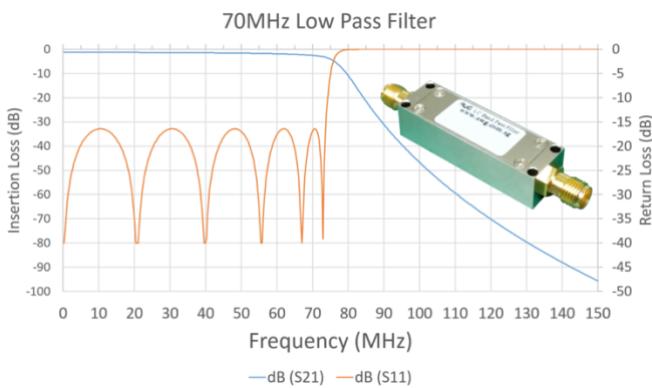
מופיע	MCP33131D-10 [15]	MAX11905 [14]	LTC2261-14 [13]	ADS4129 [12]	AD9226 [7]
יצן	Microchip Technology	Maxim Integrated	Linear Technology / Analog Devices	Texas Instruments	Analog Devices
רוחancies	16-bit	20-bit	14-bit	12-bit	12-bit
תדר דגימה מכיסימי	1 MSPS	1.6 MSPS	125 MSPS	250 MSPS	65 MSPS
רוחב סרט	N/A	N/A	N/A	550 MHz	N/A
Interface	SPI	SPI	LVDS	LVDS	Parallel
צריכת הספק	Low	Low	140 mW at 125 MSPS	185 mW at 250 MSPS	90 mW at 65 MSPS
מחיר	10\$	39\$	66\$	300\$	29\$

בטבלה 7.4 נעשו פירוט רחב של דוגמים שונים בעלי תדרי דגימה, מאפייני תקשורת ורמות רזולוציה שונות. אלו שמיים לב שטוחי המהירים משתנים בהתאם לפרמטרים האלה لكن הבחירה של ה-ADC המתאים צריכה להוות פשרה בין אותם פרמטרים.

טבלה 7.5: דירוג של ממיר D/A.

AD9226	ADS4129	LTC2261-14	MAX11905	MCP33131D-10	מאפיין
8	8	9	10	9	רזולוציה
8	9	10	4	3	תדר דגימה מקסימלי
10	2	5	7	8	מחיר
8.7	6.3	8.0	7.0	6.7	משקל

כפי שהסביר בפסקה י"ז, במערכת המבוצעת דגימה של אות רצף בזמן, אלו ממצאים עיבוד אות ומתחבים ספקטרום שכפולים של מרכיבי התדר המקוריים וזה מסכן את הספקטרום ב-Anti-Aliasing (AAF). ב כדי למנוע את ה-Aliasing אנו משתמש במסנן LPF המשמש כ-Filter) במבוא מערכת הדגימה. מדובר בחלק בלתי נפרד מהמערכת שלנו, שאנו צריכים לשלב בפרויקט, לאחר והספקטורומיים של הפלסים הם בדרך כלל אינטנסיביים, והוא בוודאות נתקנן ב-Aliasing. קיימות מספר שיטות למימוש של LPF עם תדר ברק באוזור 10 MHz שהוא המתאים ביותר למערכת שלנו. ישנו רכיבים מוכלים המבטיחים אופיין מאוד איזוטטי עם הנחתה מהירה ב-Stop Band אך מחירם יקר וכל אחד בנפרד בעל תדר ברק בודד שלא ניתן לכיוון. בגרף 7.1 מוצגת הדגמה של תגובה לתדר של מסנן זהה.



גרף 7.1: הדגמה של תגובה לתדר של LPF מסחרי.

האלטרונטיבה היא למש מסנן זהה באופן עצמאי והכוונה היא להשתמש במערכת CAD שבuzzرتה נתקן מעגל חשמלי (Schematic) ו-Layout של אחד מהטופולוגיות הנפוצות כמו אליפטית, Butterworth, Chebyshev וכדומה, ומסדר מתאים שמתאפשר על האחדות של the-pass Band והנחתת stop Band. לאחר מכן להליכים / לשЛОוח להדפסת המעגל. שיטה זו זולה יותר ביחס לקודמת, אך בו זמן מסכנת את לוח הזמנים של הפרויקט בשל התלות בהצעת הרכיבים, ומוסיפה הפסדים בהזנת האות לדוגם בגל אפקטים פרזיטיים. באIOR 7.2 מוצגת המחשה של מעגל מודפס זהה.



איו 2.7: המחשה של PCB של LPF.

במסגרת נושא החלופות האפשרות במימוש הפרויקט, אנו פוגשים מגוון רחב של ייחדות עיבוד אותן, והן באו לידי ביטוי בסקר הספרותי שביצענו במסמך היום [3]. בטבלאות 7.6 ו-7.7 נעשו פירוט ודרוג בין כמה ברטיסי פיתוח של FPGA שיכולים לשמש כיחידת עיבוד אותן.

טבלה 7.6: פירוט של כרטיסי FPGA שיכולים לשמש לצורכי DAQ-DSP.

Xilinx Spartan-7 XC7S25 [16]	Intel Cyclone IV EP4CE22F17C6N [17]	Lattice MachXO3L- 6900 [18]	Microsemi IGLOO2 M2GL050 [19]	מאפיין
23,360	22,320	6,900	50,000	Logic Cells/Elements
80	18	8	66	DSP Slices/Blocks
Integrated PLLs	PLLs	Integrated PLLs	High-performance PLLs	סוג שעון
150	154	335	125	I/O Pins
1.8 Mb	594 Kb	1.2 Mb	1.6 Mb	RAM
Low power, DSP applications	Cost-effective, low power	Instant-on, flexible I/O	Low power, secure, reliable	מאפיינים מיוחדים
260\$	125\$	200\$	800\$	מחיר

מתוך ההשוואה שנעשתה בטבלה 7.6 אנו מבינים שהיחידת FPGA היא השקעה כספית, אך כפי שהראנו בפרק הספרות, היא משלטת עבר דגימה בתדרים גבוהים מאוד וביצוע אנליה מהירה בעזרת העיבוד המקביל. لكن מיומוש של ייחידת עיבוד הолос הנקלט צריכה לבוא עם תכנון נכון וקפדי של כמה השערים כי אנו שמים לב שהמחרים של ברטיסי הפיתוח עלולים משמעותית בכל שכמות השערים ויחידות DSP ואיכות השעונים גדלה.

טבלה 7.7: דירוג של כרטיסי *FPGA* שיבולים למשתמש לצורכי *DAQ* ו-*DSP*.

Xilinx Spartan-7 XC7S25	Intel Cyclone IV EP4CE22F17C6N	Lattice MachXO3L-6900	Microsemi IGLOO2 M2GL050	מאפיין
7	7	4	9	Logic Cells/Elements
8	5	3	10	DSP Slices/Blocks
9	8	7	10	סוג שעון
7	6	5	9	RAM
3	6	5	1	מחיר
6.8	6.4	4.8	7.8	משקל

מצד שני, אנו יכולים למשתמש את יחידת העבודה בעזרת מיקרו-בקרים מוכרים שפירוטו ודירוג שלהם נמצא בטבלאות 7.8 ו-7.9 בהתאם.

טבלה 7.8: פירוט של מיקרו-בקרים שיבולים למשתמש לצורכי *DAQ* ו-*DSP*.

Raspberry Pi 4 Model B [22][23]	Odroid-XU4 [24]	BeagleBone Black [25]	Arduino MKR Zero [26]	מאפיין
Quad-core Cortex-A72 @ 1.5GHz	Samsung Exynos5422 Octa-core	AM335x 1GHz ARM Cortex-A8	32-bit ARM Cortex-M0+	CPU
2GB, 4GB, 8GB LPDDR4	2GB LPDDR3	512MB DDR3	32KB	RAM
MicroSD card slot	MicroSD slot, eMMC socket	4GB eMMC, MicroSD slot	256KB flash, MicroSD slot	Storage
GPIO, HDMI, USB, Ethernet, SPI	GPIO, HDMI, USB, Ethernet	GPIO, HDMI, USB, Ethernet	GPIO, SPI, I2C	I/O
4K, dual-display, Wi-Fi	Mali-T628 GPU, eMMC support	PRU for real-time processing	Low power, IoT focused	מאפיינים מיוחדים
Yes	Yes	Yes	No	תומך ב-DMA
68\$	100\$	82\$	54\$	מחיר

יש מגוון רחב במאפיינים של הרכיבים בטבלה, בין אם זה בתדר השעון הראשי, כמות הליביות, נפח הזיכרון ובמatters הפריפריות הecessary לשימוש, באשר יתרה של אחד מהפרמטרים האלה מגדיל את יכולות המיקרו-בקר. חלק מן המיקרו-בקרים בטבלה זמינים לרכישה במקללה, הם בעלי תיעוד רב

בAINTRNET והתמכשות נוכה. אנו שמים לב שביחס ליחידות FPGA אנו מדברים על רכיבים זולים ממשמעותית, שאמנים אינם בעלי יכולות קליטת מידע ועיבוד מהירות כמו-L-FPGA אך הן בהחלט מהוות פשרה. כפי שציינו בפרק הספרות, אנו יכולים להתגבר על העיבוד הטורי שנעשה במיקרו-בקרים ולצמצם את פערם אל ה-FPGA וזאת על ידי שימוש ב-DMA וכן החלטנו לציין בטבלה איזה מיקרו-בקרים בן תומכים במנגנון זה ואילו לא.

טבלה 7.9: דירוג של מיקרו-בקרים שיבולים לשימוש לצורכי DAQ ו-DSP.

Raspberry Pi 4 Model B	Odroid-XU4	BeagleBone Black	Arduino MKR Zero	מאפיין
9	10	7	4	CPU
9	8	6	2	RAM
9	9	8	6	I/O
10	10	10	1	DMA-ב
9	8	8	7	Interface/IDE
9.2	9.0	7.8	4.0	שקלול

המאפיינים הספציפיים של המיקרו-בקרים בטבלה 7.9 הם הפרמטרים עליהם נרצה לדעת לפני שנקבע איזה מיקרו-בקר ישמש אותנו בפרויקט, ואנו סבורים כי הם הći קרייטיים לתפקיד DAQ. ניתן לראות בטבלה כי שמו משקל רב על זמינות זיכרון-RAM, מגוון פריפריאות ויכולת התמכשות נוכה. לבסוף אנו בחרנו להמשיך עם ה-B Raspberry Pi 4 Model B מכיוון שהוא רכיב חזמין במקללה, הוא מציג מאפיינים מרשים למרות המחיר הזול יחסית, ובמובן הוא רכיב נפוץ מאוד בקרב מפתחים חובבים ומתקדמים וכן יש תיעוד רב ומקורות מידע עשירים על אופן פעולתו.

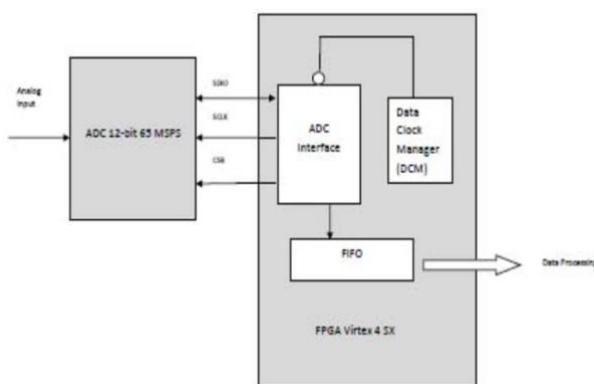
8. סקר ספרות

במסגרת הייזום עסקנו בעיקר בחקירת הדרכים האפשריות למימוש הפרויקט, את המידע הזה נתנו לנו המאמרים ומקורות מידע שעסקו בעיקר ב-Fast Data Acquisition במודון של שיטות. כולל שילבו בעבודה שלהם את שיטת הקליטה בעזרת FPGA וממשכו אותה להזנת העבודה (ברטיס פיתוח או מחשב) ותציגו למשתמש. כפי שצינו בפרק ההקדמה, לאחר התעמעקות בדפי רכיבים ותוכנו אופרטיבי הגיעו למסקנה כי علينا להתאפשר ולבחור בפתרונות את הרכיבים המרכזיים את המערכת. השאייפה שלנו במהלך הפרויקט היא לצמצם את כמות הסיכוןים בכך, لكن החלטנו למש את כולן בעזרת יחידת MCU חזקה המתאפשרת עם מודול ADC חיצוני. הניתוח הספרותי שביצעו במסגרת FPGA היא הייזום נתן לנו מבטعمוק לתוך עולם הדגימה והනיחות שלו. זיהינו כי התאפשרות בעזרת A/D היא שיטה נפוצה וחזקת מודולADC קלייט מדע מהירה, וניתן לנצל את העיבוד המקבילי שלה לטובות עיבוד אחרות בזמן אמת. מצד שני, כפי שנציג בהמשך, ישנה אלטרנטיביה של מימוש מערכת ניתוח האות בעזרת יחידות מיקרו-בקרים. בכל המאמרים שהוצעו במסגרת הייזום, הייתה התאפשרות ישירה של המערכות עם מודול ADC חזוני שביצע את הדגימה והעביר את המידע של הדגימה ליחידת הקליטה בברטיס הפיתוח או המחשב. חקר הנושא תרם לנו מאוד, לאחר ונתקטו במימוש של המערכת שלנו באמצעות השיטה, ועליה נפרט בהמשך. لكن, בסקירה זו אנו נשתף את תוכן הספרותי שמוביל אותנו במימוש הפרויקט כאשר הוא מתפרק על שני נושאים: התאפשרות של המיקרו בקר עם המודול ADC והבנה של ה-ADC עצמו מהՃדי יצרן.

Data Capture Via High-speed ADCs using FPGA 8.1

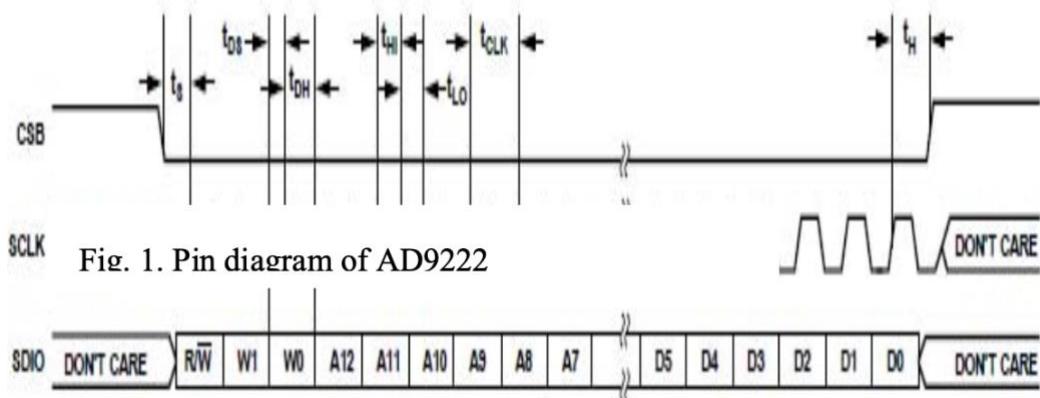
המאמר פורסם ב- International Conference on Micro Electronics and Telecommunication Engineering ב-2018, ובו ציינו שיטה של קליטה מידע מהויה בעזרת ADC's High Speed ADC בברטיס רכיב FPGA. המאמר מציע למשה רכיב AD9222 המסחרי שביכולתו לעמוד בקצב דגימה של כ- 65 MSPS וברחולוציה של 12 ביטים (4,096 רמות מתח) בטוחות מתחים של ± 2 dB. בעדרת ממיר זה, המאמר מראה שיטה לתקשורת יعلاה עם רכיב זה, בನיסיון של ניצול מלא הפוטנציאלי ברכיב, עם חסכון בחומרה וזכרון נוספת. הסיבה שמחברי המאמר הציגו את ה-ADC DDR (Dual Data Rate) והיא שיטה חסכונית בצריכת הספק סטטית ודינמית, אך החישרונו שליהם נובע מפעולות תזמון מורכבות, כאשר המידע נקלט גם בעליית ו גם בירידת השעון, ואז נוצר אתגר של התמודדות עם שגיאות קליטה.

ה-AD9222 הוא ממיר נתונים שתומך בשיטת ה-SPI (Serial Peripheral Interfacing) וזה שיטה שלמרות צריכת הספק המוגברת, מאפשרת תכנון וIMPLEMENTATION של תקשורת מול מעשנים שונים עם מעט חוווטים. מחברי המאמר מציעים למשם ממשק מבוסס FPGA המכיל בו ארכיטקטורה לתקשורת יعلاה יותר עם הרכיב על בסיס הדיאגרמה המוצגת באירור 8.1.



איור 8.1: דיאגרמת בלוקים של ממשק תקשורת בין AD9222 לבין FPGA.

כפי שניתן לראות בסכמה, הממשק שמנוהש בעזרת FPGA מתקשר עם ה-ADC בשלושה קוויים בלבד. כאשר הקו התיכון (CSB) מורה על קיראת מידע טורי ממור מאנלוגי דיגיטלי, ושני הקווים האחרים SCLK ו- SDIO משמשים להעברת סנסורן ומידע דו כיווני, בהתאם. התוצאה שמתבקשת לתוכה הממשק היא כפי דיאגרמת זמן באירור 8.2.



איור 8.2: דיאגרמת זמן של ההמרה עבור AD9222.

חשוב לציין כי בסוף קליטת בлок מידע אחד זה, יש בידינו מידע על דגימה אנלוגית שבוצעה ובוצעה לה קוונטייזציה. במערכת שאנו מתכוונים עבור קליטת הפלס האקריא אנו יכולים להשתמש בשיטה זו, ובסוף התרחיך שמתואר באירור זה, אנו יכולים להעביר את בлок המידע לשלב הבא שהוא עיבוד האות. דבר נוסף שאנו מסיקים ממאמר זה, הוא האינדיקציה שמיושם ממשקים בעזרת רכיב FPGA היא אופצייה נוספת שאנו יכולים לשקל בצד לביצוע תבניות יעיל וחסכוני של המערכת.

High Performance SW Architectures for Remote High-Speed DAQ 8.2

המאמר [1] פורסם על ידי צוות מהנדסים מאוניברסיטת ברגרד באתר MDPI הנותן גישה פתוחה למאמרים מדעיים. בעת אנו נתעמק בעולם-hData Acquisition המבוסס בマイקו-בקר, אשר הם נהיים חלק בלתי נפרד במערך עצום של יישומים, וזאת מתוך הורסיליות שלהם, פשוטות בתכנון ואדרטיביות. ברגע למערכות מבוססות FPGA,マイקו מעבדים מוכרים הרבה יותר בקרב המפתחים מה שעשו באותו בחירה מועדף למטרות כלליות. הדבר המשמעותי ביותר שמעביר את העילות של המיקרו בקרים הוא שימושם במערכות פעולה בזמן אמת (RTOS), Real-Time Operating Systems,RTOS, RTOS (Real-Time Operating Systems,RTOS,RTOS).

מאחר והן מפשטות את תהליכי התכנון ומאפשרות אמולציה יעילה של תהליכי מקבילים. במאמר זה הציגו מימושים שונים של מערכות DAQ (Data Acquisition) בתוכמים שונים. למשל, בתעשייה הצבאית, מערכת DAQ מנומשת בצד לוחות פגמים על ידי מדידת מתחי זורמי בהלחמות, והיא עשו את זה בעזרת מיקרו-בקר Cortex-M7 RTX Cortex-M7-RTX ו-RTOS Cortex-M3-RTOS ו-RTOS II OS/C II OS/C. בצד דומה, בעולם תחזוקת מכונות, מערכות מאובזרות בマイקו-בקר ביעילות שיאפשרו עיבוד מדויק. באופן דומה, המערכת מוגבהת באמצעות מודולים אינטגרטיביים במערכת. זו מציג ייעילות רבה של המערכות מסוג זה, למורות זיכרון אחסון ויכולות אනליזה מוגבלים.

בוגבי המאמר ביצעו ניתוח רחב בין מערכות DAQ שונות ויצינו גם כן מספר פקטורים שמוסמנים לקחת בחשבון כאשר בוחרים מערכת של DAQ. אנו צריכים קודם כל לשימם לב כמות העורכים המתאפשרת לדגימה, הרזולוציה של רכיב ה-ADC, מגבלת בmouth המידע שיכולה לעבור בזמן מסוים במערכת, הקצב בו המידע נקלט וכדומה. בטבלה 8.1 נעשה השוואה בין מספר מערכות DAQ מתוך המאמר.

טבלה 8.1: השוואה בין מערכות DAQ שונות.

Table 1. Overview and comparison of the reviewed DAQ systems.

Paper	Num. of Channels	Resolution [bit]	System Throughput [bps]	Acquisition Rate [Sps]	Visualization	Type of Storage	Remote	Comm. Technology
[11]	1	16	152.6 M	10 M	No	Local SDRAM	No	/
[12]	4	12	915.6 M ¹	20 M	No	Server	No	PCIe
[15]	10	12	921.6 k	500 k	Yes	Server	Yes	UART
[19]	2	16	31.25 k ²	1 k	Yes	Server	Yes	Ethernet
[22]	1	12	58.6 k ²	5 k	Yes, on small LCD	Local SRAM	No	/
[5,26]	8	24	1 M	16 k	No	Local and server	Yes	IEEE 802.15.4
[29]	8	16	312.5 k ²	20 k	Yes	Server	Yes	USB
[33]	1	12	23.5 k ²	2 k	No	Local and server	Yes	UART
[35]	16	16	6.4 M	25 k	Yes	Server	Yes	WiFi
[36]	64	32	47 M	24 k	No	Server	Yes	Ethernet
This work	1	16	51.2 M	3.2 M	Yes	Server	Yes	Ethernet

¹—when not factoring in compression; ²—estimation.

בטבלה 8.1 מוצג מגוון תכונות של מערכות DAQ עברו היישומים שהוצעו לפני כן. מנוקודת המבטו שלנו ברור לחילוץ כי לכל תבונה יש תחום של ערכיהם, וכך ברור כי ישנים מערכות שモותאמות לאופן מסוים של עבודה. למשל, אחת המערכות (12) ה특별ה בכך שהיא מסוגלת להעיבר 915.6 Mbps באמצעות אינטגרציה של FPGA ותקשורת PCIe מהירה, אך היא אינה תומכת בויזואליזציה ואנליה של האות בזמן אמיתי, ומיצעה רישום של האות בלבד. מצד שני, נשים לב כי מערכת מס' 35 קלטה בעודרת WiFi יוכלה להעביר מידע של 6.4 Mbps אשר מראה את יכולות של מערכת DAQ המבוססת מיקרו-בקר, כאשר מדובר רק בקצב העברת המידע נפוץ מאוד ביישומים מתקדמים. במובן שהקצב רחוק מאוד מזה של-hFPGA, אך אנו רואים בבירור כי המיקרו-בקר נותן לנו יכולת התאפשרות של דגימה והעברת מידע מהירה, לבין עיבוד אות בזמן אמיתי וויזואליזציה. חשוב לציין רק כי היה ניתן להכפיל את קצב העברת המידע כמעט פי 8 עבור מערכת המיקרו-בקר, כפי שעשו במערכת 36 והיא העבירה בקצב של 47 Mbps, אך תמבה הרבה פחות בתנהלות בזמן אמיתי.

בחיפוש השיטה הבי אופטימלית להשתמש במערכת המבוססת מיקרו-בקר לביצוע DAQ, מחברי המאמר הציעו להשתמש בגישה ישירה ל זיכרון, המוכר בתור DMA (Direct Memory Access) ומסבירים את הייחודה של השיטה זו ביעול תהליבי קליטת מידע מהירים. במאמר השתמשו לצורך הדוגמא במיקרו-בקר STM32H747AIHX6 אשר הוא עם תדר שעון של 480 MHz ובעל ממתקי ADC מהירים מאוד. DMA מאפשר העברת מידע ישירה מה-ADC אל הזיכרון ללא הענסה של מעבד הליבה של המיקרו-בקר. בשלב זה הוא קרייטי מאוד למערכות DAQ, לאחר מכן ואנו שואפים לעיל אותו ואת המהירות העברת מידע שלו, וככל שנינו בשלב זה, ישפייע על תפקודם של המרכיבים.

אחד היתרונות הביא גודלים של DMA היא יכולת להגדיל את כמות השבודה המקבילה. כאשר ליבת המיקרו-בקר עוסקת במשימות אחרות, DMA מאפשר העברת מידע רציפה בין-ה-ADC ו-Memory המתאימים. המקבילות ה兹וטות מתייעלות עוד יותר בעזרת צנרת דו-שלבית, המאפשרת לבקש ה-Buffers DMA לנהל את דגימת המידע ותתחליפו לעבוד במקביל. מדובר בשלב ייעול שלא רק ממכסם את קצב העברת המידע, אך גם מונע השהיות, מה שעושה אותו מאוד שימושי בתהליבי זמן אמיתי. בנוסף, קיימת אפשרות של Multibuffering, שיטה המאפשרת למערכת לבצע עיבוד על buffer אחד בלבד, ואפשר באותו הזמן buffer אחר מתמלא במידע ומאפשר קליטת מידע וביצוע עיבוד אות במקביל ללא התנגשות.

מעבר לכך, התכונון של הזרמת מידע רציפה לתוך הזיכרון המהיר ביותר של המערכת, בחיבור ישיר עם DMA, והשימוש מכפיל את רמת המהירות והיעילות. תכונון זה מתבצע מוקומית ומ恣מת משמעותית את הזמן הנדרש להעברת מידע, ומגביר את יכולות של המערכת לעמוד בקצבם העברת מידע גבויים מאוד. לצורך הדגמה, באIOR 8.3 מוצגת דיאגרמה של ערך תקשורת מהיר ויעיל המשתמש בשני בקרים DMA.

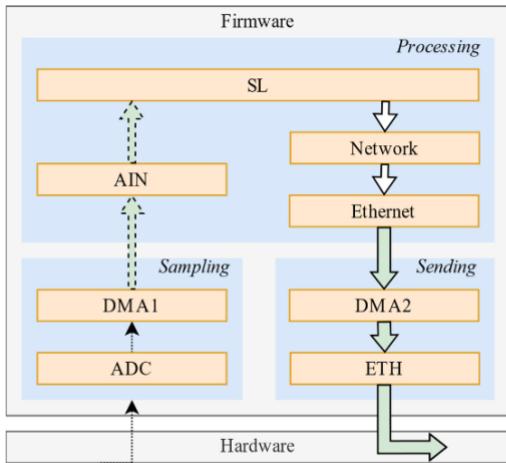


Figure 8. Samples stream path in firmware.

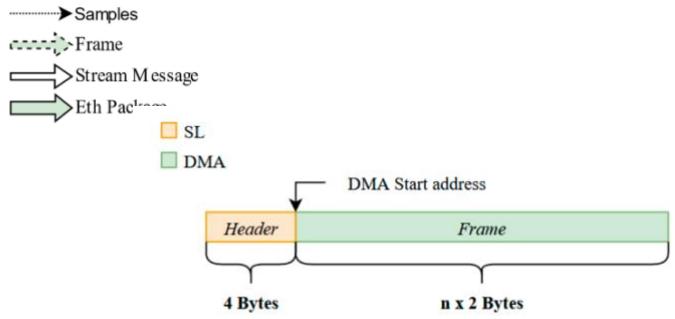


Figure 9. Stream buffer structure.

איור 8.3: דיאגרמה של ערוץ תקשורת המנצל שני בקרים DMA.

8.3 מודול ADC ורכיב בעזרתו נדגום את הפולס האקרוא'

בניתו הפטורתי שביצענו במסמך הייזום שמננו לב לנק' שלל מערכות DAQ (Data Acquisition) השואפות לדגום אוטומטיים מהסביבה בהתאם להשתמש בדרכם בברכיבים מוכלים או מודולים אשר מטרתן היחידה לדגום ולהמיר את האות דיגיטלי שימוש להמשך העיבוד. ישנו רכיבים רבים ושוק וראינו את מגוון המודולים בהן השתמשו כתובבי המאמר במסמך הייזום וכל אחד מהמודולים האלה היה בעל מאפיינים ייחודיים וצורות התממשקות שונות.

ראינו כי ישנו רכיב AD9222 [2] והוא ממיר נתונים שתומך בשיטת ה-SPI (Serial Peripheral Interfacing) וזו שיטה שמאפשרת תכנון וIMPLEMENTATION פשוט של תקשורת מול ממשקיים שונים עם מעת חוויה. מדובר ברכיב מסחרי בעל 12 סיביות רזולוציה עם טווח מתחים של 2Vpp [2] וקצב דגימה מבסימלי של MSPS 65. במאמר שהוזג [3] ניתן לראות שעבור ה-FPGA מדובר בהחלה בשיטה יעליה אף מומלצת, לאחר והיא מימושה בעזרת ה-Oscilloscope AD9222 בזמן אמת עם מספר שערים קטן. יחסית.

רכיבי ADC בין אם הם מתפקדים בתור רכיב חיצוני למערכת או רכיב אינטגרלי בתוך המערכת עצמה, היא יכולה להיות ממומשת במספר ארכיטקטורות. סוג הארכיטקטורה מותאם למערכת מטעמי מהירות המריה, רזולוציה, וועלות. בטבלה 8.2 מוצגים סוגים שונים של ארכיטקטורות [4], היתרונות והחסרונות של כל אחת והשימוש העיקרי שלהן.

טבלה 8.2: פירוט והשוואה של ארכיטקטורות ADC.

ADC TYPE	PROS	CONS	MAX RESOLUTION	MAX SAMPLE RATE	MAIN APPLICATIONS
Successive Approximation (SAR)	Good speed/resolution ratio	No inherent anti-aliasing protection	18 bits	10 MHz	Data Acquisition
Delta-sigma ($\Delta\Sigma$)	High dynamic performance, inherent anti-aliasing protection	Hysteresis on unnatural signals	32 bits	1 MHz	Data Acquisition, Noise & Vibration, Audio
Dual Slope	Accurate, inexpensive	Low speed	20 bits	100 Hz	Voltmeters
Pipelined	Very fast	Limited resolution	16 bits	1 GHz	Oscilloscopes
Flash	Fastest	Low bit resolution	12 bits	10 GHz	Oscilloscopes

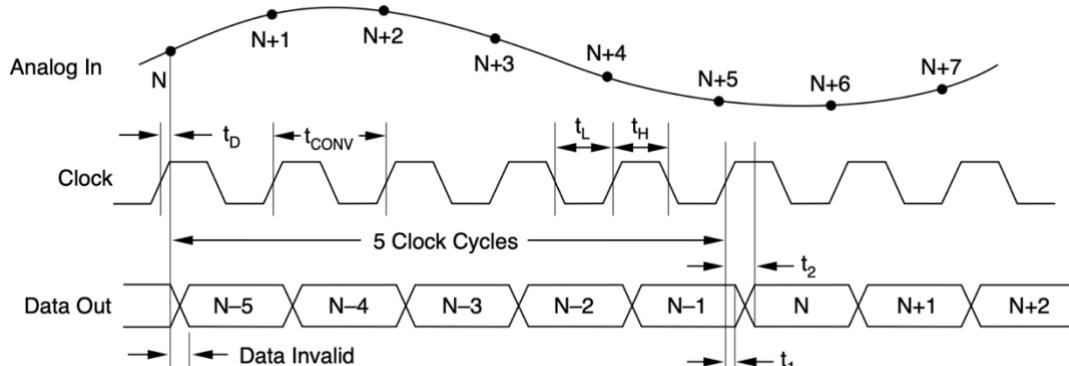
בחירה של ארכיטקטורה נבונה עבור מערכת ספציפית הוא שלב חשוב מאוד, וכן יכולים לראות מן הטבלה כי כאשר מהירות הדגימה גבוהה מאוד או Pipeline-Flash יהיה הבחירה הטובה ביותר כאשר

ה-Dual Slope Delta Sigma הינה הבחירה הגדולה ביותר. אם הרזולוציה היא הדבר החשוב ביותר במידע הנקלט אז אנו רואים שה-Delta Sigma הינה הבחירה הטובה ביותר, כאשר ה-Flash יהיה הבחירה נוספת.

באשר מדובר בממירים נטונים חיצוניים (דיגיטלי לאנלוגי או אנלוגי לדיגיטלי) יש מספר שיטות לתקשורת עםם, חלקם משתמשים בשיטת תקשורת אחת בלבד ויש כאלה שתומכים במספר שיטות מה שועשה אותם מאוד ורостиילים. ובמונון, כל שיטה מציגה את היתרונות והחסרונות הייחודיים שלה [5]. קודם כל קיימות שיטת הטרוירות, שהן למשל ה-I2C, SPI ו-UART. לצידם יש את השיטה המקבילית שמתבססת על העברת מידע בזוירה מקבילתית מה שגורר יותר חוווטים, אך מוגבר את מהירות העברת המידע משמעותית. נציין רק שקיימות גם שיטות התמסחרות נוספות כמו חיבור USB ותקשורת אלחוטית.

שיטת ה-SPI פועלת לפי מבנה Master-Slave שבו רכיב Master שולט על התקשרות עם אחד או יותר Slaves. מדובר בשיטה שפותча בהעברת מידע מהיר, והיא עיליה בתדרי שעון גבוהים ומאפשרת אפיקו תקשורת מקבילתית שבה מתבצעת קריאה וכתיבה בו זמנית. מצד שני, אין לשיטת התקשרות זו מבנה ליזוי שגיאות, וכך לא באפשרות תוכנה או חומרה נוספת. בדומה אליה, יש את שיטת ה-I2C שהיא שיטת תקשורת המשמשת בשני קווי תקשורת שכילים ליצוק קומפלקס של Masters ו-Slaves על אותו bus ומאפשר מימוש של מערכות תקשורת מורכבות, אך היא שיטה איטית יותר יחסית ל-SPI. היתרון המהותי של ה-I2C הוא בעל מבנה ליזוי שגיאות ושמירה על מהימנות המידע המועבר.

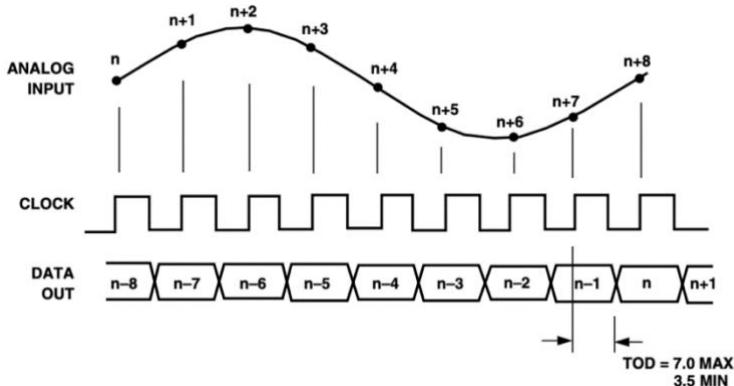
רכיב נוסף שפגשנו במאמר سابق במסמך ייזום הוא ה-ADS825 [6], ותוכנותיו דומות מאוד לרכיב הקודם כאשר מהירות הדגימה היא 40 MSPS, אך השוני המהותי הוא אופן התקשרות והוצאה המידע. רכיב זה מעביר דגימה ב-Parallel Bus של 10 סיביות רזולוציה, והוא מעביר אותה בעלילית שעון שהוא מקבל מהמייקרו-בקרה. באIOR 8.4 מוצגת דיאגרמת זמן לתיאור אופן ההמרה של הרכיב.



איור 8.4: דיאגרמת זמן של ההמרה עבור ADS825.

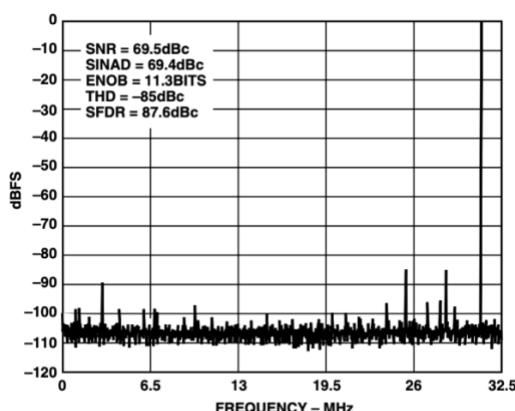
לאחר סקירה מעמיקה שביצענו בנושא הממירים של אותות אנלוגיים לדיגיטליים, מצאנו מודול יחסית זול, בעל הספק צריכה נמוך מאוד, 12 סיביות רזולוציה ויכולת דגימה של עד 40 MSPS. מדובר במודול של AD9226 [7].

קודם כל, בעזרת הארכיטקטורה של הצנרת הרב-שלבית הדיפרנציאלית, הרכיב דוגם את אות הכניסה אפיקו בתדר המביסימי ואינו מאבד מהסיביות הנמוכות של הרזולוציה (שהן בדרך כלל הביקשות למיניות). הרזולוציה היא של 12 סיביות لكن הוא מאפשר 4,096 רמות מתח על טווח מתחים של ±5 V, שכן על 10 kpt [7]. נשים לב כי מדובר ברכיב בעל טווח מתחים גדול בהרבה מזה של ה-AD9222. אופן הדגימה שמתבצע ברכיב זה הוא S&H (Sample and Hold) (ונזכר רק כי הרעיון שועמד מאוחר יותר) והdagima הזיה על פני דגימות אחרות (כמו טבנית, רכבות הלמים וכו'), היא שהדוגם מחזק למשך אינטראול קצר מאוד (hold stage) את מתח האות האנלוגי ומאפשר לדוגם לביצוע דגימה מדוייקת יותר. שימוש באופן דגימה זהה לטובת דגימה של אות אנלוגי בתדרים גבוהים הוא מובן מאחר ודגימה ישירה של אותות בתדר גבוה היא בעייתית ולכך אנו נרצה לבדוק כמה שיוור, ובנוסף, אנו מצמצמים את אפשרויות Aliasng והשפעתו במקרה של תטא דגימה. באIOR 8.5 מוצגת דיאגרמת זמן לתיאור אופן ההמרה של הרכיב.



איור 8.5: דיאגרמת זמן של המרمرة עברו AD9226.

נשים לב שרכיב זה בניגוד לרכיב ADS825 מעביר את המידע על האות הדגום בכל רידית שעון שהוא מקבל מהמיוקו-בקה. לצורך ההדגמה, בגרף 8.1 נתנו צילום של הספקטרום של אות דגם בתדר 31 MHz (תדר גודל בהרבה מהפולס האקריאי שאנו מסמליצים).



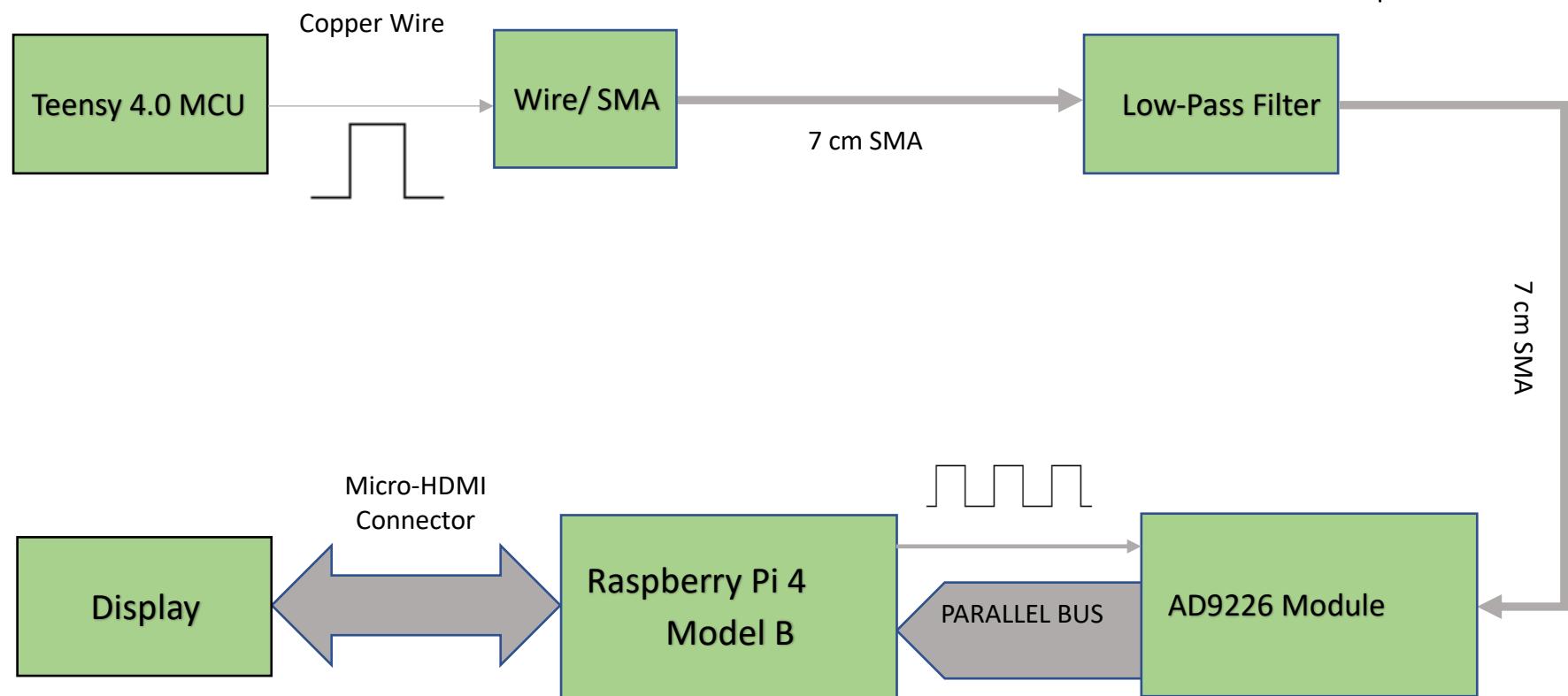
TPC 3. Single-Tone 8K FFT with $f_{IN} = 31 \text{ MHz}$

גרף 8.1: צילום ספקטרום של אות בתדר 31 MHz הנקלט ב-AD9226 מופיע הימן.

מתוך הצילום של הספקטרום בדף ייצן אנו יכולים להסיק מספר מסקנות, קודם כל אנו שמים לב כי ישנה רזולוציה גבוהה מאוד שנובעת מכמות הדגימות, אך מאוחר ואנו דוגמים את האות של 31 MHz אנו נמצאים על הגבול של תדר Nyquist ולמן אנו מצפים חילגתו להרמוניות צדדיות. אנו רואים אותן נוכחים בספקטרום אך הן dB 10 מעל גובה הרעש ובכמעט dB 85 מתחתאות המידע. זה מראה על כך שאפשרי לקבל דגימה איבוטית של האות בניסחה, אפילו אם נעמיד את הרכיב במצב של תחת דגימה. לרכיב זה יש תכונה מאוד שימושית, וזה ההשניה המצוומצמת של הרכיב במצב של שינוי תדר השעון שהוא מבניםים לרכיב שmagdir ישירות את קצב הדגימה. במידה ואנו נctrar לשנות את תדר הדגימה של ה-ADC לשם שינוי רזולוציה בספקטרום, אנו למעשה מבניםים את ה-AD9226 במצב של "שעון לא יציב" והוא דורש לפחות 100 פעימות שעון חדשות כדי להתייצב על תדר הדגימה החדש.

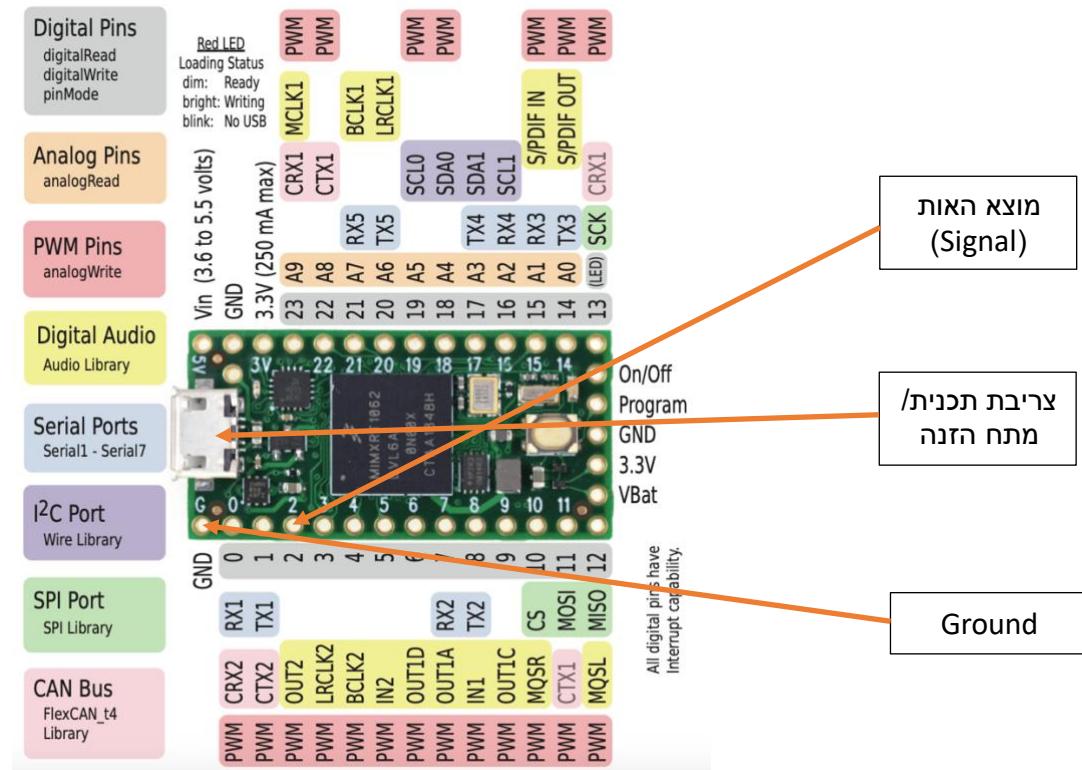
9. שיטות

9.1 תכנן ותבנן המערכת
9.1.1 דיאגרמת בלוקים



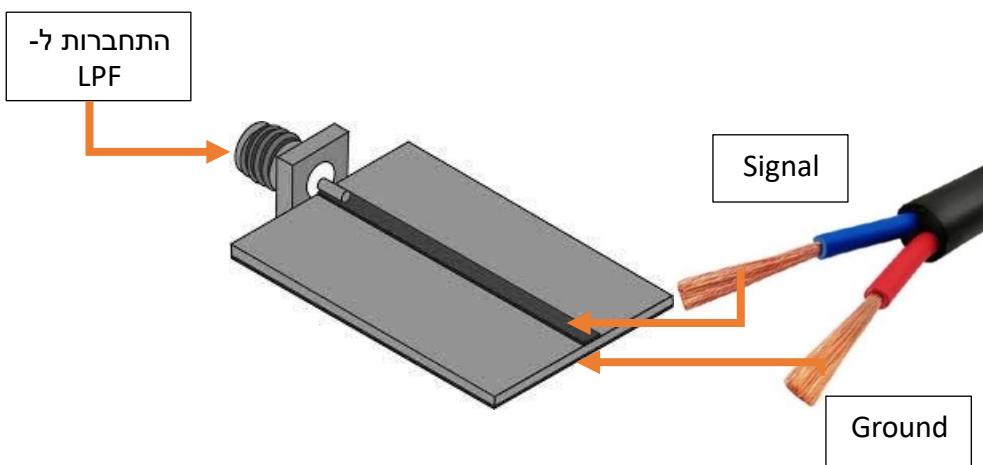
9.1.2 רכיבים

בפרויקט זה אנו מתכוונים לבצע אינטגרציה של מספר רכיבים בוודדים שמתוכננים לפועלות יצירת האות, העברתו למערכת הדגימה והעיבוד, והציגו למשתמש. לשם יצירת הолос האקריאי אנו מתחווים להשתמש במקרו-בקר 4.0 Teensy 4.0 ולהשתמש באחת מהיציאות הדיגיטליות שלו לשם ייצור הолос. הסיבה שבחרנו דווקא ברכיב זה ולא באחרים היא מתוך הבטחון ביכולת יצירת פולס הרצוי, ללא צורך בהשקעה בספיט מופחת ביחס לשאר האופציונות. -Teensy הוא רכיב החוץ הספק נמוך – הוא מוזן במתח של [v] 5 ובעת פעולה מעבד מקסימלית של 600 MHz הוא מדרים בערך 250 mA. את הפעולה של מיקרו-בקר זה ניתן לכתוב בתוכנית שפת C, שניצבת אל המיקרו-בקר בעזרת IDE של Arduino. הצריבה נעשית בעזרת כבל USB-Micro USB שמתחבר למחשב ומצדיו השני אל ה- Teensy. את יצירת הолос אנו מתחווים לבצע על ידי בתיבת תוכנית פשוטה שתשתמש באחת ממוצאי ה-PWM (ראה איור 9.1) של המיקרו-בקר, ותשדר דרכו פולסים בוודדים ברוחב של μm 1 באנטרוולים מוגדרים.



איור 1.9: בחירת מוצא PWM של ה-Teensy 4.0.

בכדי למנוע הפסדים בהעברת האות מモצא ה-0LPF Teensy 4.0 אל ה-SMA, אנו מתקונים להשתמש בכבלים של SMA באורך של 7 ס"מ. הבחירה שלנו בסוג זה של כבל נובעת מהרצן להתאים את ה-Front של מערכת הדגימה למדרי RF ולאפשר התחברות ישרה וקלה של מכשורם/ מעגלים המזינים אות אל המערכת שאגם הם עם חיבור SMA. חשוב לציין כי לטובות חיבור כבל SMA למערכת, אנו צריכים להושיב את חיבור ה-SMD של ה-SMA על גבי לוח הלחמה/ לוח נחושת ולה לחבר אליו את הcab�ים מה-Teensy כפי שמסומנים באירור 9.2. ניתן לראות המחברה של המרא בין החיבורים כאשר אות המוצא (ה-Signal) מחובר לפס microstrip נחושת שמחובר לציר הפנימי של ה-SMA, וה-Ground מחובר למשטח נחושת מצדדיו השני של הלוח ובראילץ החיצוני של ה-SMA.



איור 9.2: המחשה של המרת בין חוטי נחושת לכבל SMA.

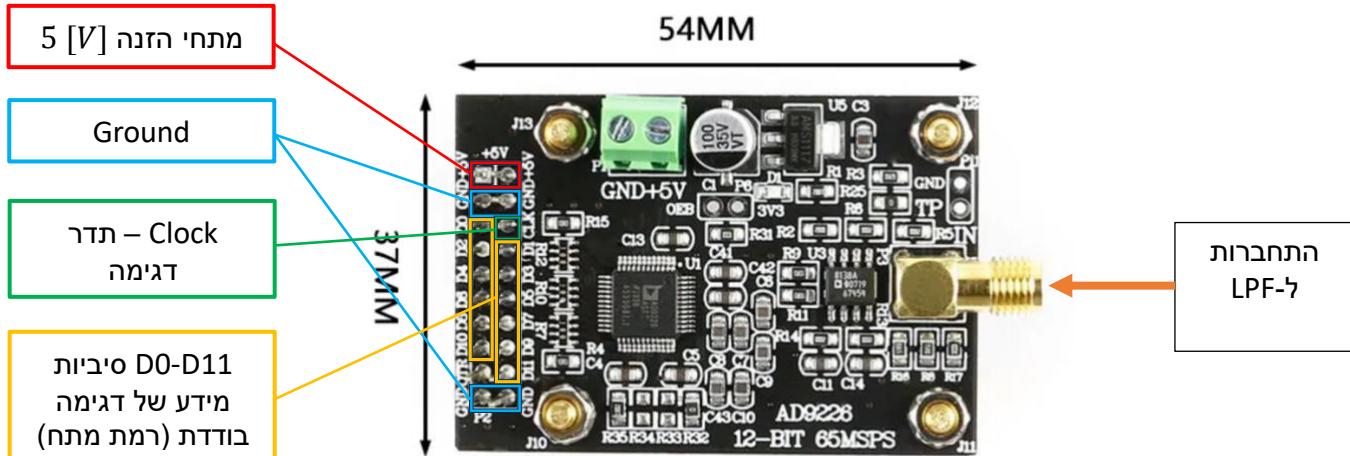
כפי שצוין בפרק החłówות, אנו יכולים לשלב את המסנן LPF במספר שיטות, אך אחריו שיקולים פנימיים החלטנו לא לבצע תכון משלנו למסנן ולפי המלצת המנחה – להציג מסנן מוכן שיתאים לצרכינו. תפקידו של המסנן LPF הוא לשמש במסנן AAF, لكن הוא מבצע דיכוי הרמוניות גבוחות הקיימות בפולס האקריאי ואנו רוצים להעבירו ל-ADC בצוות האיכותי ביותר. לכן, אנו מתכוונים לרכוש עבור הפרויקט מסנן LPF עם תדר קיטוען שאנו נציג בפרק 9.1.6 מזור חיישובים וסימולציות ראשונות, ולהתחבר אליו עם כבל SMA. מעבר לכך, מדובר ברכיב פסיבי, שאנו לא מתכוונים להזין לו מתחי הדנה, لكن יהי הפדיון בכניסה (Insertion Loss) שאנו נדע מראש מתוך מתקות מתאימות, ואנו נשתדל להתגבר עליה בהתאם בשלה עיבוד האות. פרמטרים קריטיים שאנו רוצים לשים אליהם לב הם Flatness כמה שייתור גבואה ב-Band Pass Band עם סדר מסנן כמה שייתור גבואה כדי לקבל הנחתה מהירה ככל האפשר ל-Stop Band, צרכים אלה נובעים מנעימת עיוותים בפולס וڌחיה של הרמוניות מעל לתדר Nyquist. דוגמא למסנן מסוג זה ניתן לראות באירור 9.3.



איור 9.3: המחשה של מסנן LPF עם תדר קיטוען של 10 MHz.

בחלק מהפרויקט אנו מתבקשים לבצע דגימה אינטואטיבית של הפולס האקריאי ולהציג את הספקטרום שלו. אנו יודעים מהקורסים התאורטיים במבוא לאותות ומערכות-DSP כי אנו דוגמים למעשה את החלק בעל אינסוף הרמוניות لكن לא מציאות להראות את כלו, אך אנו בהחלט יכולים להציג את החלק המשמעותי ביותר. עמידה בתנאי Nyquist היא דגימה בתדר של לפחות 20 MHz (עבור פולס ברוחב 1 μs) אך לטובת הצגה אינטואטיבית אנו נשאף לתדר דגימה של לפחות 20 MHz 20 שגרור MSPS. עבור דגימה אינטואטיבית והשקרה בספיית סבירה אנו החלטנו על מודול 12 סיביות רזולוציה שיוביל לדגם בתדר מקסימלי של 65 MSPS, لكن יתאים לצרכים שלנו. המודול בעצמו עם חיבור SMA מובנה וכן התחברות אליו פשוטה, והוא יסונכרן עם יחידת עיבוד אותות בעזרת שעון מוקצה על ידי היחידה, ויעביר בכל רזיה שעון דגימה חדש של אותות בכניסה למודול. הדגימה עוברת מה-ADC באמצעות

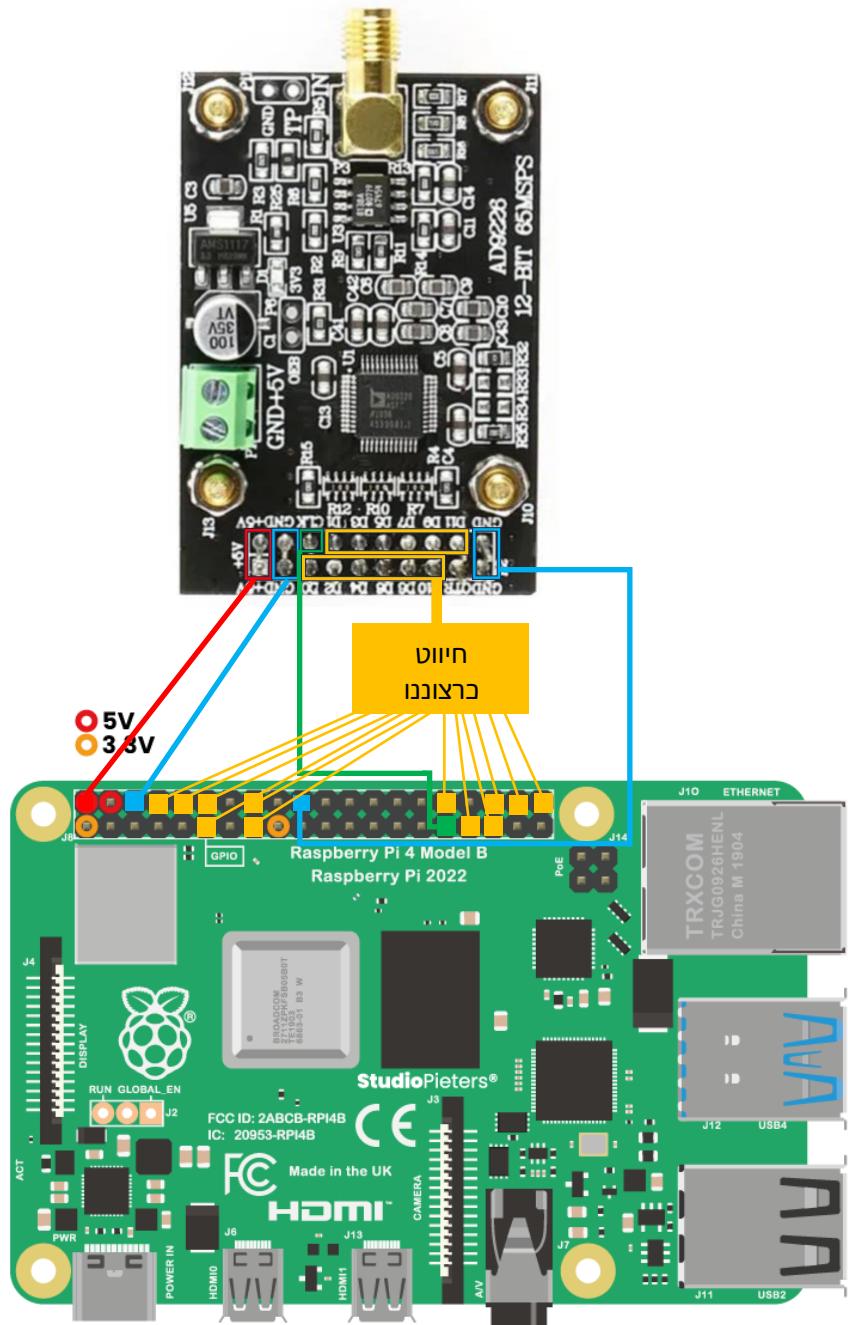
GPIO (במotaות סיביות לפי רמת רזולוציה שנבחרה) ומתחברת אל יחידת העבודה בכניסות ה-Parallel Bus שלה. בנוסף להכל, הוא מוזן על ידי מתח של [v] 5 לkn מדבר ברכיב הצורף הספק נמוך. באIOR 9.4 ניתן לראות המחשה של ה-ADC המדובר אשר מצד אחד יש התחברות ל-LPF עם חיבור SMA, ומצדו השני פינים הזנה (באדום וכחול) ופינים דיגיטליים (בירוק וצהוב).



איור 9.4: המחשה של ה-AD9226 Module עם פירוט על החיבורים.

לא הכרחי שיחידת העבודה (ה-RPi) תספק הזנה עבור ה-ADC אך אנו נשאף לכך מאחר זה ישמר אינטגרציה יותר פשוטה של כל המערכת. הפינים CLK ו-D0-D11 יהיו העורך תקשורת בין יחידת העבודה לבין ה-ADC, ובשלב הניסויים אנו נבצע את החיבורים בעזרת Wires Jumper, ובאשרנו נוודא שהמייתוג הזה עובד עבור הפרויקט אנו נרצה שהחומר יהיה יותר מסודר והולם על ידי פיסת PCB.

הרכיב החשוב בשרשראת, זה שיבצע את עיבוד האות ויתכן שגם את הציגה אל המשתמש הוא Raspberry Pi 4 Model Bn. הסיבה לכך שאנו בחרנו להשתמש במיקרו-בקר ולא ביחידת FPGA נובעת מכך שאנו רצינו לצמצם סיבוכים טכניים וסיכון לו"ז שיכולים לבבוע מכתיבת תוכנית VHDL. מאוחר וביצוע עיבוד האות בפרויקט הוא ביכולות גם של ה-FPGA וגם של המיקרו-בקרים, העדפנו לבוחר בדרך הנוחה לנו גם מבחינת התאמימות, מציאות שגיאות ודימינוט. הסיבה שבחרנו דווקא ב-Raspberry Pi 4 Model B מבין המיקרו-בקרים שצינו בטללה בפרק על החלופות הטכנולוגיות היא הזמיןות של הרכיב במקללה, ב모ות ה-GPIO, תדר השעון המרכזי ונפח זיכרון-h-SRAM מספקים שבורנו. רכיב זה הוא בעל מערכת הפעלה של AxPin וההتمmensיות אליו היא כמו עם מחשב נייח סטנדרטי כאשר מחברים לו מסך תצוגה, עכבר ומקלדת. בכך להפעיל את המיקרו-בקר אנו צריכים מתח הזנה של [v] 5 על [A] 3 [22]. אנו מוצפים כי שלב קליטת המידע יכלול הפעלה של מגנומן זיכרון שדרושים תקשורת עם החומרה עצמה, וכן אנו מתכוונים לכתוב תוכנית בשפת C שתהווה את ליבת העבודה. מאוחר והמערכת הפעלה של AxPin בדף כל אינה מתאימה לפעול בזמן אמיתי [27], אנו נרצה לוודא את ההצלחה של קליטת מסר גדוול של דגימות בקצב העובר את היכולות של המעבד. אנו יכולים לספק את זה על ידי מגנומים זמינים ב-RPi שאינם תלויים במעבד כמו ה-DMA וה-SMI, יפורט בפרק 9.1.3, ויכולים לתחזק את התקשרות עם ה-ADC בצורה אמינה. זו גישה שתנתן לנו את האפשרות לקלוט כמות דגימות מסוימת שבעזרתה נוכל להציג את הפלט האקריאי במשור התדר ובמשור הזמן. ניתן לראות את אופן החיבור באIOR 9.5 כאשר פירוט על בחירת הפינים המתאים ב-RPi נמצא בפרק 9.1.3, אך באופן כללי אין ממשות לסדר בו נחבר את הסיביות D0-D11 (מאחר והכל יוגדר תוכניתית בהתאם), כל עוד יחויבו למקומות המסומנים על ה-RPi.



איור 9.5: אופן החיבור בין ה-AD9226 ל-RPi.

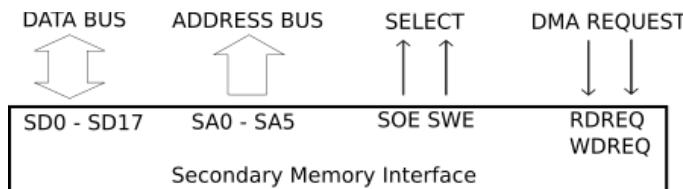
לבסוף, כאשר יהיה בידינו את כל המידע הרצוי על הפולס האקרוי, אנו מתכוונים להציג אותו גרפית על מסך שבשלב ההתחלתי יהיה צג מחשב ובמידה ויהי לנו מספיק זמן אנו נשדרג את המערכת עם מסך תצוגה המחברת ל-RPi. צוואר הביקון בהוספה צג המחברת ל-RPi הוא זמיןות נקודות האזנה ופינים פנויים עבור התקשרות בין ה-RPi למסך.

9.1.3 תקשורת בין ה-ADC ל-RPi

בכדי לעורר את העברת הדגימות דרך הקווים D0-D11 (מידע על הדגימה ב-ADC) אנו צריכים לספק ל-Clock ADC שווייה יציב ורציף במשך כל זמן הדגימה שאנו מעוניינים בו. בנוסף לכך, אנו צריכים לקרוא את המידע בקווים בהתאם ל-Clock ללא השהיות והתנשויות ב-RPi. שיכולות להוביל לפספוס של דגימות ובשל כך גם לדגימה בזמן אמת. בבר בשלב בתיבת הדו"ח ההנדסי הגענו למסקנה כי השילטה על התקשרות בקווים ואחסון הדגימות אינה יכולה להיות בידי המעבד בגל מזרי הדגימה הגבוהים, ולכן הפנו את תשומת לבנו לשני מנגןונים [28][29][30] שבシיתוף פעולה יבצעו את משימות אלה. המנגנון הראשון הוא SMI (Secondary Memory Interface) וכפי שניתן להסביר ממשו הוא ממשה, שכן בכל הגרסאות של RPi, שמאפשר התממשקות של O/I בצורה מקבילית עם הדיזיין המקומי. מטרתו של ה-SMI היא לאחסן את הדגימות המתקבלות מקבילים בקווים D0-D11 בהתאם ל-Clock. בהנחה ופעלת המעבד הייתה מספק מהירה והינו יכולם לשמור עליו בהעתיקת Batch של דגימות בזמןים מוגדרים ללא הפרעות מזיכרון uncached, הינו מסתפקים במנגנון זה בלבד, אך לאחר התיעצויות ובדיקות שעשינו עם ה-SMI (בຕיבת מקבילית למוצאי RPi) הבנו שהוא לא אפשרי. זה מוביל אותנו אל DMA (Direct Memory Access), שהוא מנגנון מהיר ויעיל להעברת מידע ב-RPi. הוא מתפקיד חופשי לגמרי מהמעבד ומבצע העברות זיכרון O/I במהירות גבוהה מאוד. מצאנו לנכון להתגבר על צוואר הבקבוק שמציבה לנו מהירות המעבד ולהשתמש ב-DMA כדי לבצע העברת זיכרון אוטומטית ומהירה מהזיכרון אליו ה-SMI מאחסן את הדגימות, אל זיכרון uncached, שמיידית המעבד יוכל לנתח אותן. בעת אנו נפרט בהרחבה על כל אחד מהמנגןונים ואופן הפעולה שלהם.

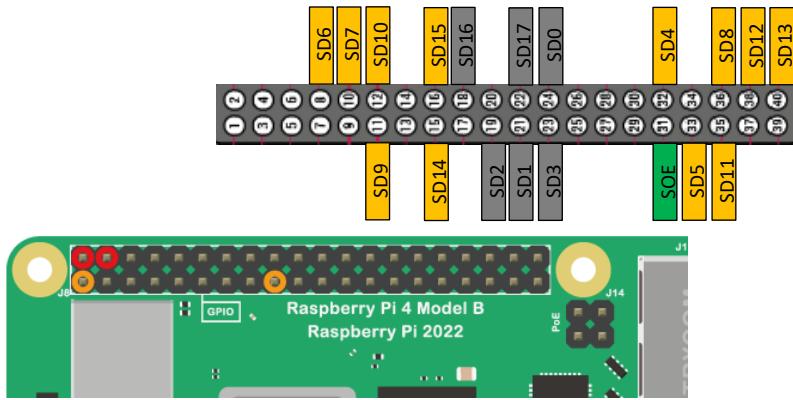
(Secondary Memory Interface) SMI

כפי שצינו לפני כן, ה-SMI [31][32] יתחזק את התקשרות הישירה מול הקווים D0-D11 וזה מתאפשר בגל צמינות רוחבה של אפשרות עברת ה-SMI, והם 18 קווים מקבילים, 6 בתיבות הקצהה בזיכרון, קו קריאה וכתיבה, קווי תקשורת עם ה-DMA. איור 9.6 ממחיש זאת.



איור 9.6: קווים זמינים ב-RPi עבור ה-SMI.

חשיבות לציין שהקווים SD0-SD17 הם קווים מחווטים ישירות למוצאי ה-RPi והפרישה שלהם מוצגת באיוור 9.7. זאת הסיבה מדוע בפרק 9.1.2 החלטנו לבצע את החיבור בין ה-ADC-L-RPi לבין RPi. הנה הלה, החלטנו להשתמש ב-SD4-SD15 (12 סיביות סה"כ) כי הן שייכות ל-SMI ומשאירות קווים פנויים עבור חיבור צג של RPi (אופציונלי). לכן הקווים D0-D11 יוחברו לפינים SD4-SD15 ודרכם יעברו סיביות המידע על דגימה בודדת. כਮובן הסדר בו אנו נחליט להזמין אותן ממשמעותי, אך אם נשמר את הלוגיקה ש-D11 הוא LSB D0-MSB SD4-SD15 יירשם בזיכרון כ-MSB-LSB.



איור 9.7: פירוט קווי SMI ב-*O/I* של ה-RPi.

הדבר החשוב שנוצר להגדר עבור ה-SMI כדי שיכל לקיים את התקשרות מול ה-ADC ולאחסן את הדגימות, היא לספק Clock בتردد רצוי מה-RPi שיוביל את ה-ADC וימקם את סיביות הדגימה בקווים בכל ירידת השעון. חקרנו מספר דרכיים לבצע זאת זה בעזרת PWM וטימרים נוספים של המערכת, אך חלום לא חופשיים לחלוין לעבוד ללא השגחת המעבד או שקשה לסנן אליהם את קרייאותה-*SMI*. הפתרון הפשטן ביותר לדעתנו הוא לנצל את אחד מקוווי הכתיבה של ה-SMI והוא SOE (Output Enable). לאחר וההתממשקות של ה-SMI היא גם עבר קריאה וגם עבר כתיבה בקווים, אנו ננצל את קו הכתיבה של ה-SMI בתור ה-Clock של המערכת. היתרון הגדול הוא שיש לנו שליטה מלאה על צורת השעון והتردد שלו ברוחוציה של ננו-שניות, ושליטה על זמני *Setup*, *Strobe* ו-*Hold*. חישובים מפורטים עבור קבלת תדר Clock מתאים נמצאים בפרק 9.1.6. מידע מלא על האוגרים המפעילים את ה-SMI ואתחלם נמצאים בנספח א'.

(Direct Memory Access) DMA

ב-RPi 4, צ'יפ BCM2711, רוב הצנרת החומרנית והפריפריות הם masters bus, מה שמאפשר להם העברת מידע עיליה ולדאוג לעצם בהיבטי זיכרון [33]. זה עושה את ה-DMA סביבה יותר נוחה לעובדה, בנוסף לכך הוא יכול לתקשר עם פריפרויות פשוטות, ואף להעביר את בלוק הזיכרון הרצוי למטען 2 (במצב קריאה בלבד). ב-RPi יש עד 16 ערוצים של DMA, כאשר כל אחד מהם מופעל על ידי טעינה של Control Block (CB) שהוא data structure שמודעתך מהזיכרון לאוגר הרלוונטי ב-DMA. כל פעולה DMA מתואמת על ידי ה-CB והשדות שלו, ואף אפשר ליצור שרשרת של פעולות כאשר ה-CB מסמן הבא לפעולה בעת סיוםו. התרומה הגדולה של ה-DMA לפרויקט שלנו נובעת מזה שאנו יכולים לעשות לו (בעזרת טעינת 's CB מתאים לערך ה-DMA), לבצע העברת בלוק דגימות מוקן על ידי ה-SMI למקום שנבחר ובו המעבד יוכל לבצע את ניתוח המידע מתי שייתפנה. במקרים אחרות, הוא מאפשר לנו לפעול בקשה מפריפריה ללא צורך בהמתנה מצד המעבד, מה שמאפשרים סיכון תזמון (בגלל תדרי הדגימה הגבוהים) ושמירת העבודה בזמן-אמת של המערכת. אם אנו לא נdag לפועל מסודרת בהעברת הבלוקים של הדגימות אנו יכולים להגיש במצב מסוכן בו בלוק דגימות יגרס וגע לפני שאנו נרצה לקרוא ממנו, וחשוב לזכור שה-Clock שMOVED ע"י ה-SMI עוצר והדges ממשיכות להיכתב בבלוק זיכרון. הסבר טכני ועמוק על ה-DMA ועל האוגרים שלו ושל ה-s' CB נמצאים בנספח ב'.

9.1.4 ניהול זיכרון בזמן-אמת

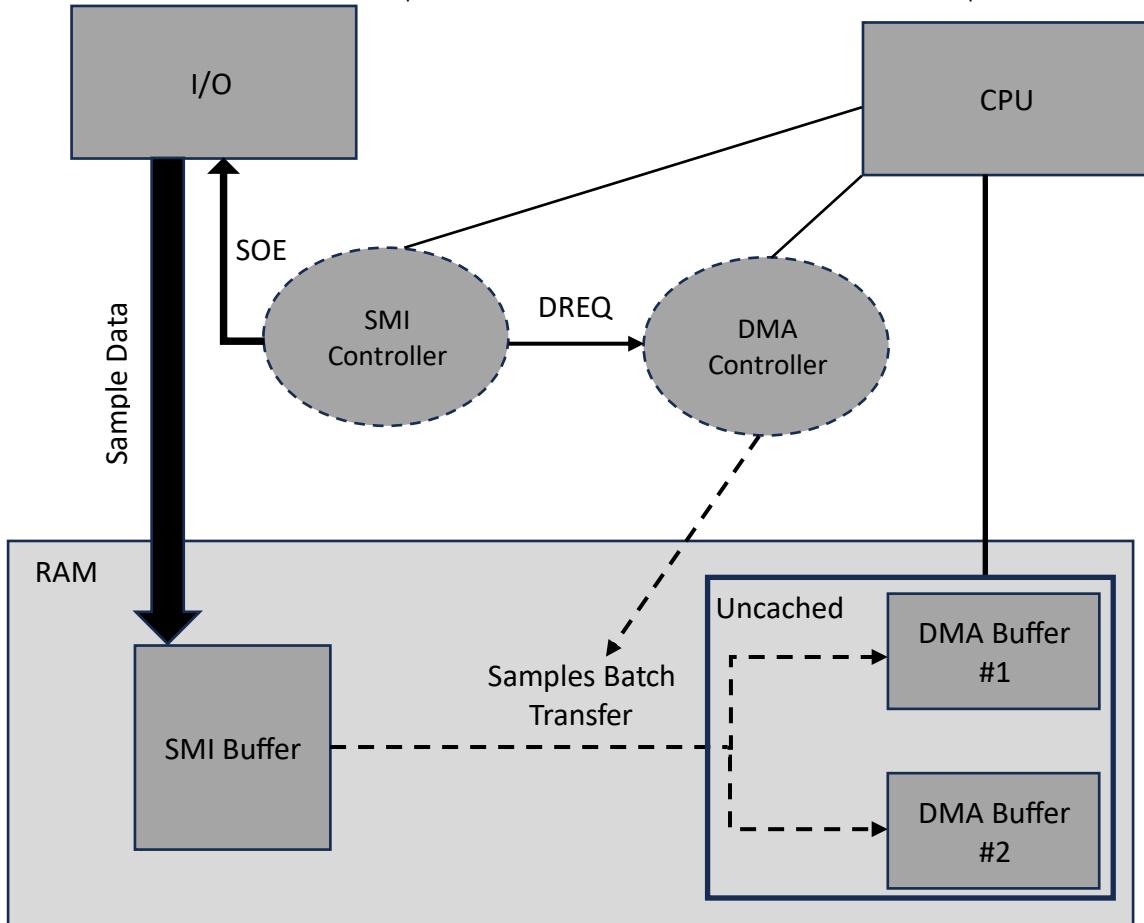
בכל החלק של עיבוד האות מתבצע ב-RPi בתוצאה מהפעלה של תכנית C שאנו נרשום. כאשר אנו מרכיבים תכנית היא משתמשת במרחבי זיכרון וירטואלי, כידוע לנו מקורס מבנה מחשבים, הכתובות לזכרון שאנו משתמשים בהם בדרך כלל פיקציה מיוצרת ע"י מערכת הפעלה למשך ריצת התכנית. זה מאפשר למערכת הפעלה להשתמש במקסימום RAM הזמין לה, אך לעיתים (תלי בעומס פקודות על המעבד) התכנית יכולה להזורך למקום אחר לגמרי, מוחץ-L[30][34]. כאשר בדרך כלל אנו לא מיחסים לכך חישובות, ברגע ואנו רצחים לתחזק תקשורת של פריפריה שתקרה מידע בזמנים מוגדרים ותחסן דגימות ולבסוף שdagmoths אלה יעברו במידי לזכרון הזמן למעבד, אנו צריכים שככל דבר יהיה במקומ תחوت תחובת ידועה. אותו מקום נקרא memory bus. אנו רצחים להגיד ולאחרל DMA-SMI במרחבי המשמש, כך שמאזתו הרגע הן פעולות באופן בלתי תלוי (עד לעצירה על ידי התכנית). כל הכתובות שאנו משתמשים בהגדרת-h DMA-SMI צרכות להיות addresses bus, אך לאחר מכן תכנית רצה באופן וירטואלי, אין לנו גישה ישירה לפריפריות. כדי לקבל גישה לקריאה/ כתיבה, אנו צריכים לבקש גישה מהמערכת הפעלה יחד עם physical address של הפריפריה הרצiosa. את הבקשה למערכת הפעלה אנו מפעילים דרך פונקציית mmap (בתוך header file שזמן בכל RPi) והוא לוקחת את physical address של הפריפריה ופותחת חלון בזיכרון הוירטואלי שאנו יכולים להשתמש בו, וכל קריאה וכתיבה אליו מועברת ישירות לפריפריה. לסיכום, יש לנו שלושה מרחבי זיכרון שאנו צריכים לפעול בהם:

- Virtual Address – כתובות במרחבי המשמש (תכנית C).
- Bus Address – כתובות על-h-sus שמאפשר גישה לזכרון וגם לפריפריות.
- Physical Address – כתובות כפי שהיא בחומרה.

באשר המעביר מה-Virtual ל-Physical / Bus אינו טריוויאלי (הסביר לפני כן), המעביר מ-Bus ל-Physical, ולהפך, הוא עניין של הוספת קבוע (משתנה בין גרסאות של RPi). לאחר חקר שעשינו בנושא, הבנו שלא מספיק לפתח גישה דרך mmap, ויש נתק בין המידיע שאנו רושמים בתכנית לבין הערךם שיועברו ל-DMA-SMI (אללא אם נשים שהוויות משמעויות בקוד) וזה נבע מה-Caching שמבצע המעבד. כדי לנו מקורס מבנה מחשבים, המעבד מבצע Caching כדי לגשת יערכים מהר יותר, ולכן יערכים שנכתבם לזכרון בקוד עלולים לא להגיע באותו הרגע למקומות בזיכרון DMA-SMI ו-SMI מקבלים הוראות וمبرירים בלקים של דגימות, ועלולים לקבל יערכים לא רצויים. האפשרות לתקלה כזו תליה בעומס המעבד, ועל סמך מקורות שבדקנו השהיה כזו עד לרגע שבו מבוצע Uncaching יכול לחתך באזורי-h 1 [34]. כדי להתמודד עם הבעיה זו, מצאנו פתרון שימושה זמינות של הערכים שלנו ב-Uncached memory וזה על ידי שימוש ביחידה גרפית של-h RPi, VideoCore [28][35][36]. מסתבר כי במקרה הזה יש עדיפות עלינה בתפקוד של-h RPi, וכן העברת בזיכרון לזכרון Uncached דרך-h VideoCore, עבור הערכים שאיתם אנו נעבד בתכנית C, מוגדר זמינותם עבור הפריפריות.

9.1.5 תכנית ליזוי ועיבוד אות האקראי בזמן-אמת

בעת, כאשר יש בידינו תקשורת יציבה בין ה-ADC ל-RPi ותקשורת ישירה של מרחב המשמש עם הדרישות, אנו נציג את האלגוריתם אותו נமמש ליזוי האות האקראי. באור 9.8 ניתן לראות דיאגרמת בלוקים המתארת את האינטגרציה של המנגנונים מפרק 9.1.3.



אior 8.9: דיאגרמת בלוקים הממחישה את פעלת המנגנונים השונים באלגוריתם.

מתוך הדיאגרמה זו ניתן לראות כי המעבד (התכנית C) בעל גישה ישירה ל-SMI Controllers של ה-ADC ולחיבורו ה-DMA וליחסון ה-Uncached שאננו פתחנו אליו גישה. אתחול ושליטה על ביצוע הדגימה ב-ADC ושליטה על אחסון בזיכרון, מתבצעים דרך ה-SMI Controllers. בעת, אנו נציג את השיטה בעדרתה אנו מתכוונים ללהות את הפולס האקראי בין הדגימות, ובסיוף נציג את האלגוריתם המלא היכול את השיטה.

בדי לאפשר זרימה בלתי פוסקת של דגימות מה-ADC לחיבורו של ה-RPi ולהות בינהן פולס בנקודות זמן שרירותיות ולא ידועה, אנו נדרשם לבדוק כל דגימה שמתקבלת, ולא ליצור חלונות זמן בהם אנו מפספסים דגימות. לשם כך אנו מתוכננים להשתמש בשיטת פינג-פונג בין שני מאגרי זיכרון (buffers) שמתבססת על סריקת מקבץ שלם של דגימות מ-#1 buffer לאחר מכן במקביל מתמלא ה-#2 buffer, ולאחר שלא מצאנו סימן לפולס בין הדגימות, אנו ממתינים ש-#2 buffer יסתティם להתמלא ונחליף את תפקידם של שני ה-buffers. תהליך זה ימשיך באופן אינסופי עד לרגע שבו נזהה פולס, ובאשר הוא יגיע אנו ניתן הוראה ל-SMI Controllers לעצור את התהילך של הדגימה רק ברגע שהוא ה-buffer שבודוק קלט דגימות. כפי שניתן לראות באור ל-CPU יש גישה ישירה לשני ה-buffers ואנו צריכים לדאוג שיש מספיק זמן למעבד לבצע את הסריקה על ה-buffer המסייע לפני שה-buffer השני

יתמלא, לאחרת תבוצע גירסה של הערכים. יש קשר ישיר בין נפח ה-buffers לבין זמן הנדרש לעיבוד הדגימות, וכן נציג את החישובים המפורטים בפרק 9.1.6. נותר לנו להגדיר מהו פולס שאנו מתכוונים להיות במסגרת התכנית, בתיאום עם המנחה אנו הגדרנו את הפולס האקראי כפולס העובר את סף המתח שਮוגדר כ-10% מתחן הטווח מתחים שמסוגל ליהות ה-ADC. במקרה שלנו מדובר בטוווח [-5,5] לבן כל פולס בעל אמפליטודה מעל [+] 1 יוגדר בתור פולס אקראי. משך הפולס הוא גם דבר שאנו צריכים לנקח בחשבון, וכן מבינים שהשיטה של הפינג-פונג מציבה לנו גבול למשך הפולס שנוכל לדגום והוא למעשה חייב להיות מהדגימה בה הוא נקלט, ועד סוף ה-buffer שמתמלא במקביל. זה יהיה אחד השיקולים לבחירת גדי ה-buffers, מאחר והם מגדרים לנו את ה-"חלון זמן" שאנו משתמשים על הפולס.

האלגוריתם הכללי ליהוי הפולס האקראי:

- I. הגדרת מקום בזיכרון CB[0] buffer #2 ו-buffer #1 uncached.
- II. אתחול ה-SMI והגדרת האוגרים שלו:
 - a. משך הדגימה.
 - b. תדר הדגימה.
 - c. גודל הדגימות.
 - d. הקצת זיכרון.
 - e. הגדרת פינים של O/I.
- III. הגדרת CB's של DMA:
 - a. CB #0+1: הפניה של הפינים O/I ל-SMI.
 - b. CB #2: ביצוע מספר דגימות סרקל.
 - c. CB #3: העברת הדגימות ל-buffer #1.
 - d. CB #4: העברת הדגימות ל-buffer #2 (+שרשור בחזרה ל-CB #3).
- IV. הפעלת ה-SMI (תחילת הדגימה).
- V. כל עוד DMA פעול:
 - a. ברגע והסתיים להتمלא buffer (התקבל DREQ), נסroxק אותו בחיפוש אחר פולס.
 - b. אם נמצא:
 1. נגדיר CB #5: החזרה של O/I למצבם המקורי וסיום DMA.
 2. נשרשר דינמיית את ה-CB הנוכחי ל-#5.
- VI. ביבוי ה-SMI.
- VII. סידור הדגימות משנה ה-buffers תחת מערך אחד.
- VIII. ביצוע חישוב FFT.
- IX. הצגת תוצאות הדגימה למשתמש.
- X. פינוי זיכרון ה-uncached.

אלגוריתם זה הוא הצגה בלילית של התכנית שאנו נציג בפרק התוצרים, חשוב רק לשים לב במספר דברים. קודם כל בנקודה III, ה-CB's משורשרים זה לזה לפי סדר המספרים וכאשר מסתיימת פעולה זו מפנה למספר CB הבא, למעט CB #4 המשורשר בחזרה ל-CB #3 כדי לשמור על מגנון הפינג-פונג. בנוסף לכך, מימוש ה-FFT יבוצע על ידי ספרייה קיימת ב-C הנקראת FFT שפירוט עליה [37] ניתן בנספח ג'.

9.1.6 חישובים תדר דגימה (SOE)

כפי שצינו בפרק 9.1.3, ה-Clock שאנו נאכיל ל-ADC הוא קו ה-SOE ששיך ל-SPI. יש לנו שליטה מלאה על הוצאה של השעון ב-SOE והוא מוגדר על ידי שלושה מרכיבים:

- Setup – הזמן שלוקח לפירפירה לפענה את הערכים.
- Strobe – רוחב הpollo לשימוש הערכים בקורסים לאחר סיום ההערכה.
- Hold – משך הזמן לשימור הערכים בקורסים על ידי השעון הראשי ב-RPi למימוש מחזור אחד של SOE.

לפי משווהה 1.9, ניתן לתאר את במתות פעימות השעון הנדרשת על ידי השעון הראשי ב-RPi למימוש מחזור אחד של SOE.

$$N_{cycles} = width \cdot (setup + strobe + hold)$$

משווהה 1.9: חישוב זמן המחוור של SOE במונחי פעימות שעון.

לפי משווהה 9.2 ניתן לחשב את תדר הדגימה.

$$f_s = \frac{f_{CPU}}{N_{cycles}} = \frac{f_{CPU}}{width \cdot (setup + strobe + hold)}$$

משווהה 2.9: חישוב תדר הדגימה המדוייק.

חישוב לצורך שהתדר שעון הראשי בגרסת B הוא $f_{CPU} = 1.5 \text{ GHz}$ Raspberry Pi 4 Model B ולהתחשב בכך שאנו מעדיפים צורת שעון סימטרית, אנו נעדר ש-*strobe* = *setup* + *hold*. לאחר ביצוע מספר בדיקות, אנו ראיינו שהתדר המקסימלי היוצר אליו הגיעו ללא יצירת התנגשויות ב-bus הוא 25 MHz . $f_s = 25 \text{ MHz}$. את התדר חישבנו באמצעות הבאה:

$$f_s = \frac{1.5 \text{ GHz}}{4 \cdot (3 + 8 + 4)} = 25 \text{ MHz}$$

마חר וה-ADC בו אנו משתמשים דורש זמן מעבר בין שינוי תדר דגימה, אנו החלפנו לא לנתקם באלגוריתם שלנו בכל סוג של שינוי תדר דגימה, ולהשאיר את במתות הדגימות בתור בדרגת חופש אותה ננצל לשם שינוי הרוחולציה בתדר Δf . נשים לב כי באלגוריתם שהוצע בפרק 9.1.5 בסעיף b.III אנו הגדכנו CB שלם לטובת ביצוע מספר דגימות סרק, זאת מאחר ואנו מפעילים את ה-ADC מנצח מנוחה לתדר דגימה כלשהו, אך יש תופעת מעבר קרצה (בדיקות הרואו שב��ביבות ה-30 דגימות) שאנו יכולים להתעלם מהן מאחר והן דגימות סרק.

תדר קיטוען של ה-LPF

במהשך לסעיף הקודם, תדר הדגימה שאותו החלפנו לא לשנות נוון לנו דרישת חד משמעות לגבי המסנן AAF שנמצא במבוא ל-ADC. מתוך תנאי Nyquist לדגימה (משווהה 9.3) המורה על מניעת תופעת ה-Aliasing החל מתרד:

$$f_{cutoff} < \frac{f_s}{2}$$

משווהה 3.9: חישוב תדר Nyquist לדגימה נכונה.

אך אנו מסיקים כי במקרה Aliasing, אנו צריכים מסנן עם תדר קיטוען במקסימום בתדר $f_{cutoff} = 12.5 \text{ MHz}$. כדי להתאפשר בין רוחב סרף לבין תדר קיטוען מקסימלי, אנו נבחר להשתמש ב-LPF עם תדר קיטוען של $f_{cutoff} = 5 \text{ MHz}$ במקסימום.

נפח ה-buffers

הבנו בפרק 9.1.5 על החשיבות של גודל ה-buffers ובעת נחשב את גודלם כך שיישמשו את זהוי הפלס האקראי בצורה אופטימלית. אם התוצאה הסופית של האלגוריתם הם שני buffers המלאים בדגימות, אז למעשה הרזולוציה בתדר שאנו נקבל ניתנת לחישוב בעזרת משווה 9.4.

$$\Delta f = \frac{f_s}{2 \cdot N_{buffer}}$$

משווה 9.4: חישוב רזולוציה בתדר.

כאשר N_{buffer} היא כמות הדגימות שמכיל buffer בודד. מתוך בדיקות שביצענו על כמות הזמן שלוקח למבצע לסרוק buffer בודד, ומתוך דאגה לרזולוציה גבוהה בתדר, ומתוך מגבלות של גודל אחסון של ה-SMI, אנו העדפנו לעשות כל buffer בגודל של f_s או במילים אחרות $= 2^{14} = 16,384 Samples$ או $2^{15} = 32,768 Bytes$. הסיבה לכך שאנו משתמשים בחזקות של 2 נובעת מתוך ייעול ביצוע ה-FFT, וכך שההתוצאות ייצגו למשתמש מהר ככל האפשר. מכאן אנו מקבלים רזולוציה של:

$$\Delta f = \frac{25 MHz}{2 \cdot 16,384} = 762.9 Hz$$

מסקנה נוספת מתוך הבחירה זו הוא רוחב הפולס המקסימלי שאנו מסוגלים ללהות, והוא יכול לעמוד בין דגימה בודדת לבין רוחב של שני buffers שלמים, החישוב המדויק הוא לפי משווה 9.5.

$$T_{max} = \frac{2 \cdot N_{buffer}}{f_s} = \frac{32,768}{25 MHz} = 1.31 ms$$

משווה 9.5: חישוב רוחב פולס מаксימלי.

9.1.7 ניתוח הספקטרום

המידע העיקרי שאנו רוצים להציג על הפולס האקראי במסגרתuproject הם רכיבי התדר המרכיבים אותו פולס. המערכת דוגמת את האות שמגיע ל-ADC ושמוררת את הדגימות עבור המשך העיבוד. יש מספר שיטות בעזרתן אנו יכולים לקבל מידע על רכיבי התדר הנוכחים באוט שנדגם, אך רובן המוחלט מיושם בשיטת החלונות (Windowing). השיטה בה נבחר תשפיע ישירות על הספקטרום של הפולס, וכן על המסקנות שניתן להסיק עליו. בעת, אנו נציג את השיטה בה בחרנו ואת השיקולים מאחוריהם.

קודם כל חשוב להבין שיחסוב הספקטרום אינו דבר חד משמעי ונitin לבצע התמരות לאות בזמן במספר רב של שיטות, שכן אנו עושים שימוש להסביר על השיטה שלמן, לאחר והיא תהיה שימושית בשלב הבדיקה של תוצר הפרויקט. בשלב הבדיקה אנו נבצע השוואה לסייעות ולתוצאות המתקבלות ב-Spectrum Analyzer שכן חישוב שימושאים אלה יהיה ברורים.

בעת נסביר שיקולים טכניים ביצוע ההתמרה שנובעים מתוכן הפרויקט ויחסוב לקחת אותם לתשומת לב, אנו נתבסס על אנליזת פוריה שפירוט מעמיק ותאורטי שלו נמצא במקורות [38][39]. בפרק 9.1.6 ראיינו שבדורך בה תכננו את מערכת הדגימה אנו יוצרים חלון ברוחב של ms 1.31. כדי להראות תפקוד נכון של המערכת אנו מזהים באותו חלון צורת פולס אקראית בנקודות זמן אקראיות ברוחב של $\Delta t = 1$, וכן מנקודות מבט החalon בו – הפלס יהיה אלא הלם. כאשר אנו מחשבים DFT של אות אנלגי דגם אנו מבצעים בטור בירית מחדל למשה חalon מלכני על האות הדגם. רווח החalon משפיע ישירות על חישוב הספקטרום בגליל 茲יגזג ספקטרליות (Spectral Leakage) שיכולה להופיע. שינוי של צורת החולן (לדוגמא Blackman ,Hanning ,Hamming וכו') יכולה במקרים רבים לצמצם את אותן 茲יגזג,

אך כדי לעשות את זה אנו צריכים רקע מסוים על אופי האות שאנו מצפים לדגום כמו תדרים ספציפיים חשובים, צורת האות, מטרפה, SNR וכדומה. בפרויקט שלנו, אנו בוחרים להשתמש בחילון מלכני מאחר והפרמטרים שצויינו לפני כן הם אקראים מבחינתנו, החישוב יהיה יותר פשוט ובנוסף אנו לא צריכים להפחית מהחולציה בתדר הסבר נוסף לכך הוא שאם אנו במודע נרצה למדוד פולס ברוחב של Δf אחד חילון של ms 1.31 ניתן חולציה מספיק גובהה כך שה贊יות הספקטוריות יהיו מוגשות ורק בקנה מידת בערך של Hz 1, ומבוחנתנו יהיו לא רלוונטיות. מכשורי Real-Time FFT נבדלים זה מזה באופן בו הם מבצעים את התמרת ובסוג החלון בו הם משתמשים, ובහלט יש מקום לשגיאה כאשר אנו נבצע השוואה בין ספקטורים במכשורים שונים. אך בסופו של דבר באשר מתיקלת חולציה מספיק גובהה בתחום התדרים הרצוי – ניתן לבצע את ההשוואה בצורה נאמנה (ROLONETI לפרק 10).

בעת אנו נציג את החישובים בעורთם אנו מתחווים לחשב את הספקטורים של הפולס האקראי X_k , בהינתן תדר הדגימה שלנו f_s , כמות הדגימות N והדגימות עצמן x_n . נתחיל מביוזו התמרת FFT (שתיוים ע"י ספריית FFTW) לפי משווה 9.6.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-j2\pi \frac{k}{N} n}$$

משווה 9.6: התמרת DFT.

בפרויקט שלנו, בדומה למכשורי מדידה אחרים, אנו מתחווים בעיקר באמצעות רכיבי התדר ומתחומים מהפaza שלhn. כדי שהAMPLITUDE של רכיבי התדר יהיה מוקור להשוואה בשלב הבדיקה אנו צריכים שהן יהיו בקנה מידת מתאימים, אך אנו נdag שAMPLITUDE התדרים A יהיו לפי רמת המתה שלhn, ולשם זה נשתמש במשווה 9.7. הפקטור 2 במשווה נובע מתוך התייחסות רק לחצי ספקטורים החשובים.

$$A = \frac{2|X_k|}{N}$$

משווה 9.7: שינוי קנה מידתAMPLITUDE.

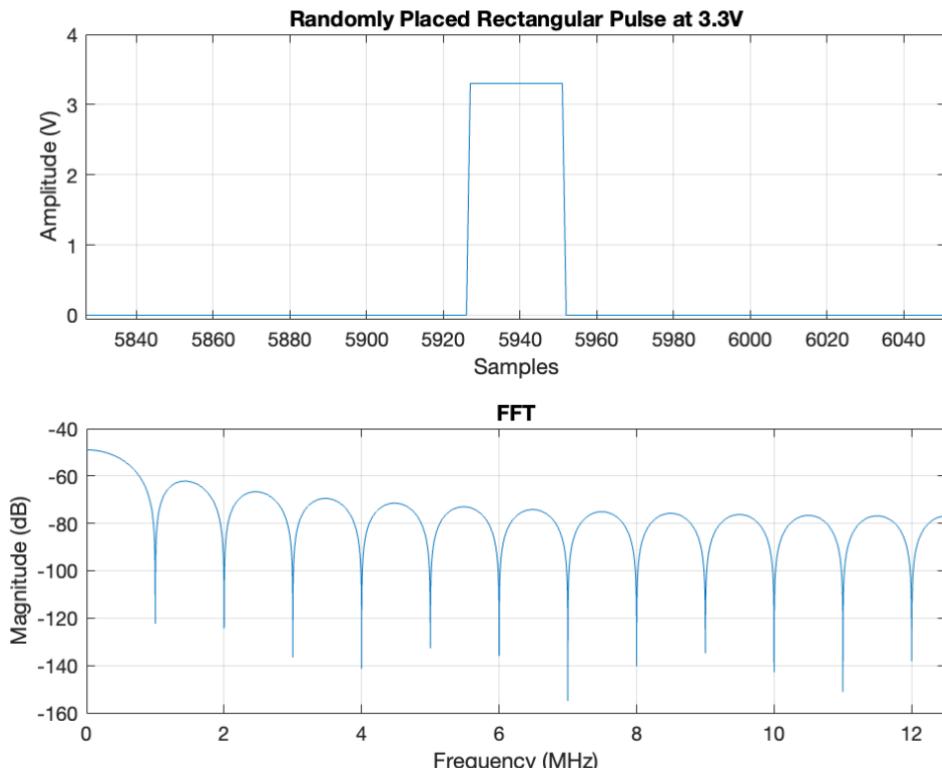
לצורך ההשוואה עם McMOS מדידה אחרים נעדיף להציג את ספקטורים ההספק. רכיבי ההספק יוחשבו לפי משווה 9.8 כאשר אנו נעדיף להציג אותן ב-BP.

$$P_k = \frac{2|X_k|^2}{N^2}$$

משווה 9.8: חישוב ההספק של כל רכיב תדר.

9.1.8 סימולציות

להתחשב בחישובים שביצענו בתת פרק 9.1.6 ובאלגוריתם שהוצג לפני כן, אנו יכולים לבצע סימולציה תאורטית של תוצאה שיכולה להיות מוצגת בהסתמך על הנתונים שהגדכנו, כמו מדר דגימה וככמויות דגימות. את הסימולציה אנו מבצעים באמצעות תכנתה Matlab ונקוד שלא מצורף בנספח ח'. בגרף 9.1 ניתן לראות הדוגמה של פולס אקריאי ברוחב Δf , באמפליטודה של V_{pp} 3.3 שנדגם במדר $f_s = 25 \text{ MHz}$ עם 32,768 דגימות.



גרף 9.1: תוצאה סימולציה עבור זיהוי של פולס אקריאי מלבד.

בעת אנו נציג סימולציה יותר מתקדמת שמשלבת מודל של רעש תרמי הצפוי להיות במבוא ל-ADC. מדובר במודל מוכבר של Johnson-Nyquist שמתאר את שונות המתח התרמי על עומס כלשהו בהינתן:

$$1.38 \cdot 10^{-23} \left[\frac{J}{K} \right] K_B \bullet \\ .300^\circ T \bullet \\ .50 \Omega R \bullet \\ .5 \text{ MHz} B \bullet \\ 9.9 \text{ noise voltage} \bullet$$

$$V_{noise} = \sqrt{4 \cdot K_B \cdot T \cdot R \cdot B} = 2 \mu[V]$$

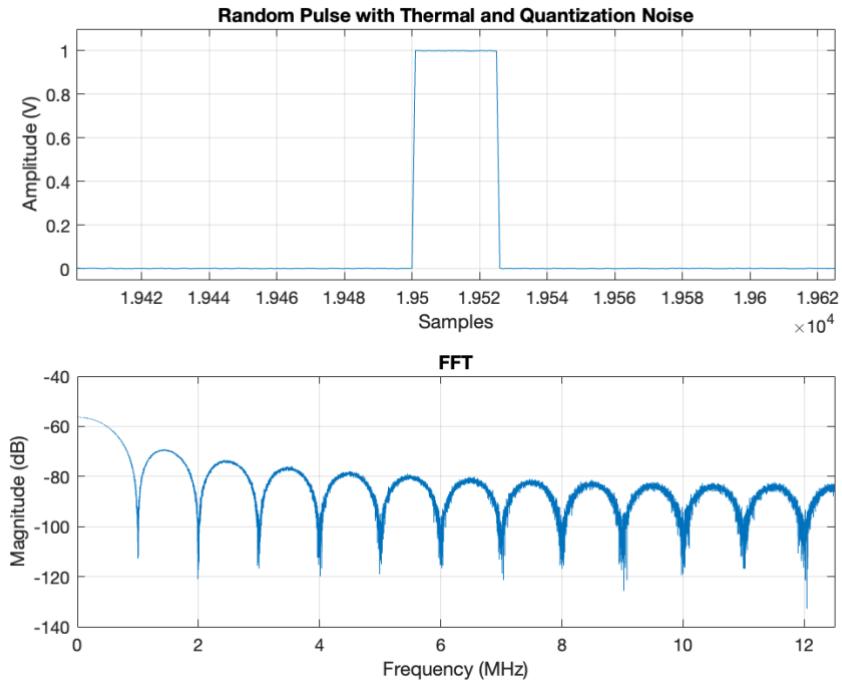
משוואה 9.9: חישוב שונות המתח (תרמי) המתפתח על עומס לפי מודל N-J.

בנוסף לכך, אנו מוסיפים רעש קוונטיצציה שנובע מתוך מגבלת הרזולוציה של הדוגימה. רזולוציות ה-ADC היא 12 סיביות, לכן ניתן לחשב את רזולוציות הדוגימה לפי משוואה 10.9.

$$\Delta V_{res} = \frac{V_{pp}}{2^n} = \frac{10}{2^{12}} = 2.44 \text{ mV}$$

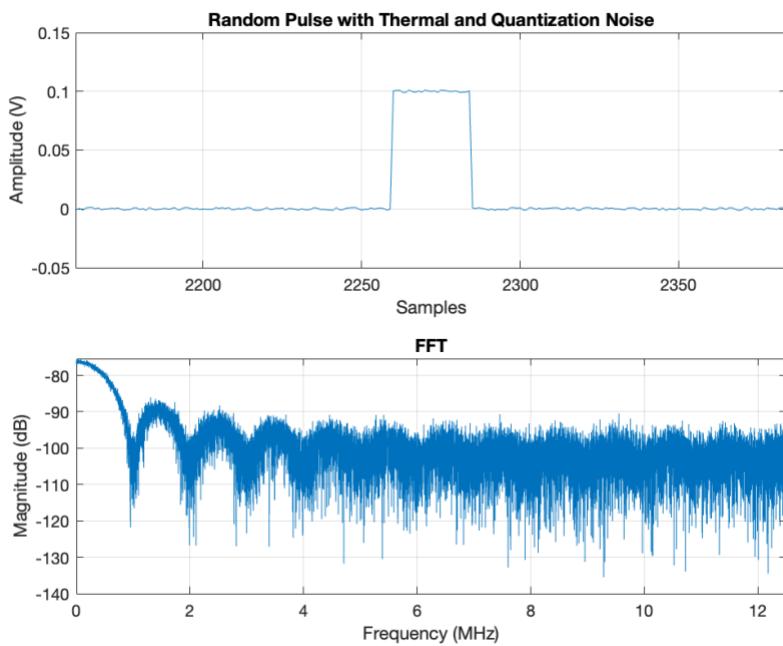
משוואה 10.9: חישוב רזולוציות הדוגימה של ADC.

שילוב של הרעש התרמי והרעש קוונטיזציה מהוות קירוב של המציאות ומקור להשוואה בין המעשית לתאוריה.



गण. 9.2: תוצאה סימולציה עבור דיאו של פולס אקראי מלכני במכוחות רעש תרמי ורעש קוונטיזציה.

אנו יכולים לראות בגרף 9.2 שלמרות הוספת הרעש לא ניתן לבדוק בשינויים במעט בכלל באוט בזמן, אך במרחב התדר ה- Hz alto במדדים מעל ל- 6 MHz מושפעים מאוד על ידי ריצף הרעש. כדי להמחיש את השפעת ריצף הרעש על הדגימות של הפולס, אנו הקטנו את האמפליטודה של אות הבנינה ל- 0.1 V .



गण. 9.3: תוצאה סימולציה עבור דיאו של פולס אקראי מלכני חלש במכוחות רעש תרמי ורעש קוונטיזציה.

ניתן לראות בגרף 9.3 יש השפעה גדולה הרבה יותר של הרעש על צורת הפולס למרחב הזמן. בנוסף לכך, החל מ- 4 MHz העונותצד של ה- c -חוס מתחזקות כמעט באופן מוחלט עם ריצף הרעש ולא ניתן

לקבל מידע אמין על מרכיבי התדר של הפולס מעבר לתדר הנ"ל. המסקנה שלנו מתחום הסימולציות היא שבמידה ואנו נציג את ה-threshold במערכת מעל V_{m0} 100 אנו יכולים לקבל מספר הרמוניות ראשונות של הפולס, ובכל שanno נרים אותו, כך נקבל מידע אמין יותר על התדרים הגבוהים ביותר ברווח הסרט. חשוב לציין שלא רקחנו באן בחשבו את התגובה לתדר של LPF שנמצא במبدأ ל-ADC, לאחר והוא מסנן סיבתי לא אידיאלי הוא משפיע על הפלס העובר דרכו. למרות זאת, בהנחה Insertion Loss בסדר גודל של עד dB 0.5.

9.2 חלוקת העבודה בין השותפים

טבלה 9.1: חלוקת העבודה בין השותפים.

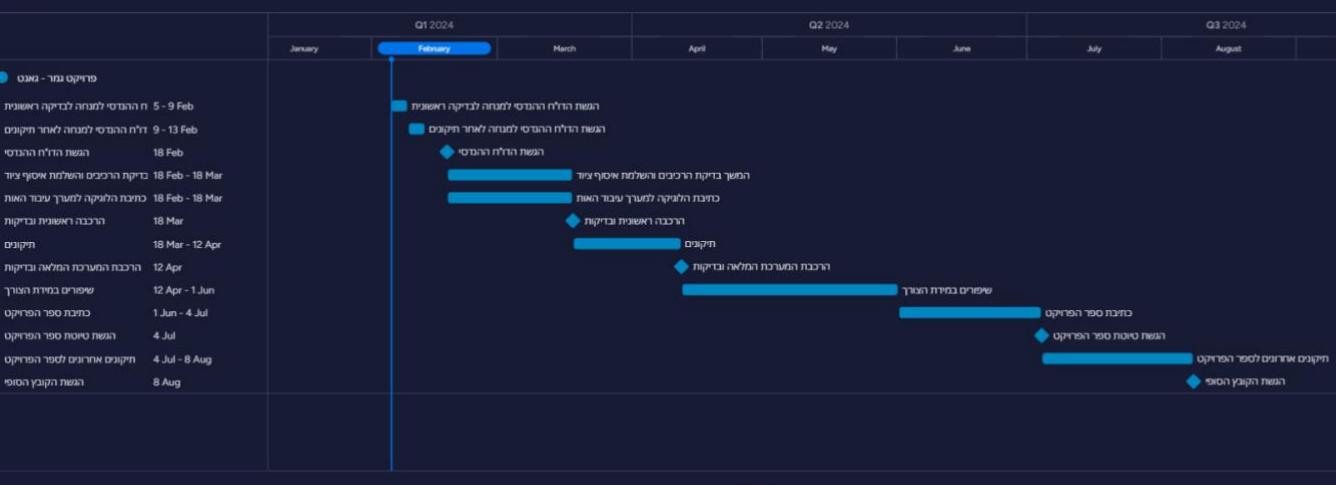
אחריות	נושא
גלאן וקצפר	אלגוריתמייקה
קצפר	תכנות מערכת הדגימה
גלאן	תכנות ממש משתמש ותצוגה
גלאן	תכנות Teensy
קצפר	בנייה PCB חיוווט בין ADC ל-RI
גלאן וקצפר	אינטגרציה וחיבורם בין הרכיבים
גלאן	בנייה מארץ והזמנת רכיבים

9.3 תבנון הפרויקט

9.3.1 תכנית עבודה סופית

Gantt

05 February, 2024 | 17:30:26



נוף 4: גאנט העבודה על הפרויקט גמר.

טבלה 2: גאנט העבודה על הפרויקט גמר.

משימה	תאריך
הגשת הדוח הכספי לבדיקה ראשונה אצל המנהה	09.02.24
הגשת הדוח הכספי פעם נוספת למנחה לאחר תיקונים	13.02.24
הגשת הדוח הכספי	18.02.24
המשך בדיקת הרכיבים והשלמת איסוף הציוד	18.03.24 – 18.02.24
בתיבת הלוגיקה למערך עיבוד אותות	18.03.24 – 18.02.24
הרכבה ראשונית ובדיקות	18.03.24
תיקונים	12.04.24 – 18.03.24
רכבת המערכת המלאה ובדיקות	12.04.24
שיפורים במידה הצורך	01.06.24 – 12.04.24
בתיבת ספר הפרויקט	04.07.24 – 01.06.24
הגשת טווחת ספר הפרויקט	04.07.24

משימה	תאריך
תיקונים אחרונים לספר הפרויקט	08.08.24 – 04.07.24
הגשת הקובץ הסופי	08.08.24

9.3.2 ריכוז שינויים

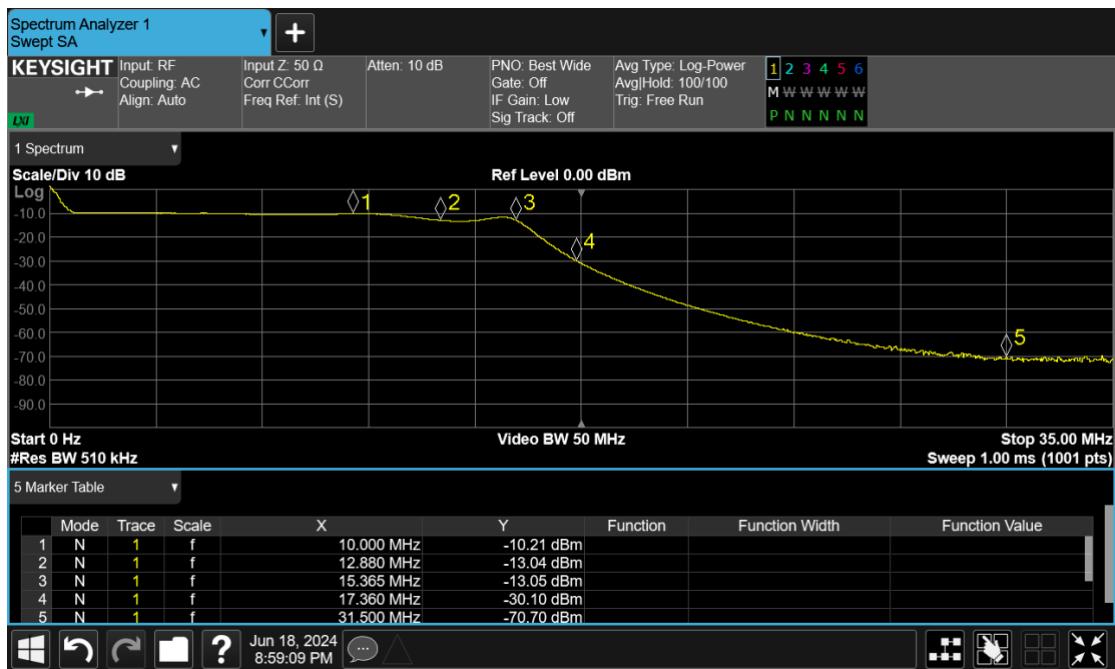
טבלה 9.3: טבלת ריכוז שינויים שבוצעו בפרויקט.

שינוי	באיזה חלק השינוי	מתי קרה השינוי	יוזם השינוי
Line Driver	פונקציונלי	03/2024	סטודנטים
מערכת נידת	פונקציונלי	05/2024	סטודנטים
$f_c = 10 \text{ MHz}$ עם LPF	פונקציונלי	06/2024	סטודנטים

באשר הצגנו למנהל הפרויקט את הכיוון הכללי בו אנו נרצה להתמודד עם המשימה, הוא הפנה את תשומת ליבנו לבעה שעוללה להופיע ברגע בו אנו נרצה להוביל פולס, שרוב הסיבוכים יהיה בהספק נמוך, ולאחר שרשרת של ממירם וחיבורם פסיביים הוא נתן להפסדים משמעותיים. בדרך התמודדות, המנהה המליק לנו לשימוש בדרגת הגבר (Line Driver) שתגדיל את עצמת הפולס. לאחר ביצוע בדיקות על האב טיפוס של התוצר הסופי רأינו שאנו מפיקים מה-0.40 Teensy 4.0 אות בהספק גובה מספיק ליזוי (בסביבות -20dBm) והפסדים מינימליים לאחר והשתמשנו בכבליים של SMA באורך 15 ס"מ ו-LPF עם הפסד מינימלי ב-band pass. בשל כך החלנו לא לשלב מגבר ישירות במוואה מקור הפולסים.

בשלב של בתיבת מסמר היוזם רצינו מאד לאפשר למערכת-h-ADC+RPi+RPI להיות נידת למגמי ולהסתמך על סוללה נידת של [7] 5. מדובר בשדרוג שהיה יכול לעשות את המערכת ככלי בדיקה נוח לשימוש בהרבה יותר מצבים, אך לאור לחץ בלוח זמינים החלנו לוותר על הרעיון.

בתחילת הפרויקט רצינו שתדר הקיטוען של ה-LPF יהיה $f_c = 10 \text{ MHz}$ ולכן הזמן רכיב בזה מסין. לאחר ביצוע בדיקה התבדר שתדר קיטוען שלו לא היה בהכרח בתדר $f_c = 10 \text{ MHz}$ אלא דווקא ב- $f_c \approx 15.3 \text{ MHz}$. כפי שניתן לואות בגרף (9.5 dBm = P_{in}).



גרף 9.5: גוף תגובה לתדר של מסנן LPF הלא מתאים.

מדובר בשגיאה שאינה מקובלת, מאחר והוא מסכן את המערכת בקיולי תדר שעולים להתרחש בתדרים מעל $f = 12.5 \text{ MHz}$ (נובע מתנאי Nyquist לדגימה). בשל כך החלטנו להזמין מסנן חדש מסין שעל הניר יהיה עם תדר קיטוע $f_c = 5 \text{ MHz}$. בgraf 9.6 ניתן לראות את התגובה לתדר של המسان החדש, שמראה שיפור ניכר ביצועים ודיוק הרבה יותר גבוה בהשוואה למסנן הקודם. מכאן והלאה החלטנו להשתמש במסנן החדש בלבד.



גרף 9.6: גוף תגובה לתדר של מסנן LPF מתאים.

9.3.3 ניהול סיכונים

הימנעות מ-*Bus Conflicts*

כאשר אנו מפעילים את ה-*I/O* של ה-*i-RPi* יחד עם ה-*I/O* של ה-AD9226 בתדרי העברת גבויים, אנו מסתכנים בהתנגשויות ב-*bus* כאשר ה-*SMI* וה-*ADC* פועלים במקביל על הקווים. זה מצב שעלול לקרות כאשר אנו דוגמים בקצב מהיר מדי, ולא מספיקה להגיע הוראה מה-*DMA* לבבות את ה-*SMI* בפניהם של ה-*I/O* בהרף זמן זה יש זרם גבוה בקווים. לפיכך היצרנו של ה-AD9226, העומס שהוא מציג בקווים הוא נמוך מאוד ($\Omega = 100$) ובמידה והקונפליקט ב-*bus* יימשך זמן רב מדי, יכול להיווצר נזק בלתי הפיך ל-*ADC*. כדי להתמודד עם הבעיה החלחנו למתן את תדר הדגימה לקצב בו לא אמורה להתרחש התנגשות כזו ($f_s = 25 \text{ MHz}$), ובמידה והיא תתרחש אז למן קצר מאוד שאינו יגרום נזק.

רכיבים לא אידאליים

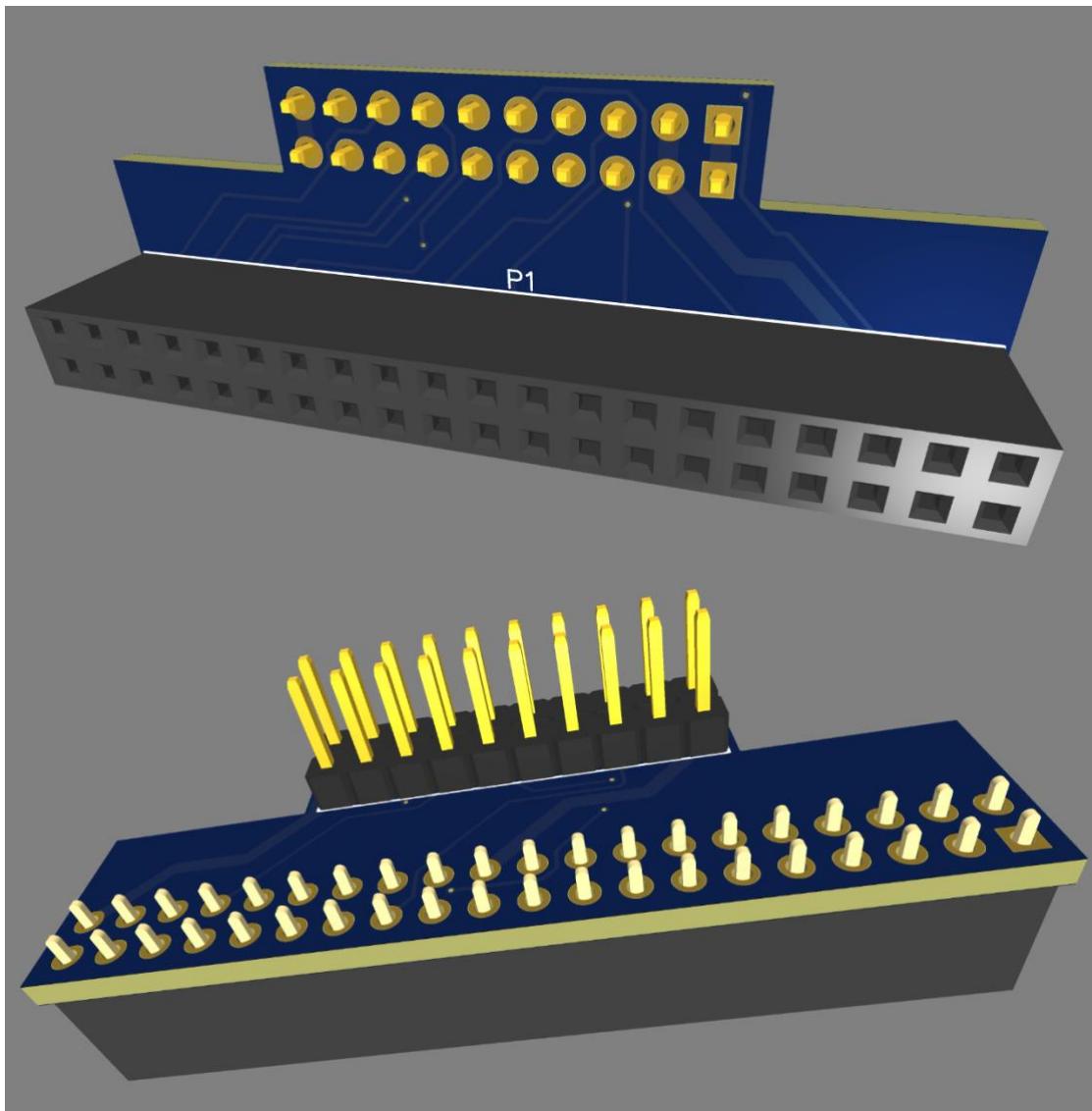
בהמשך לבעה הקודמת, אנו רואים שלמרות המחיר הזול של ה-AD9226 הוא לא רכיב אידיאלי וההתנגדות קטנה מידי במציאות קווים Data וביידה והיא הייתה יותר גדולה אז זה היה משפר את ביצועי המערכת כי איזה היה לנו אפשרות לדוגם בתדר גבוי יותר. מעבר לכך, כפי שהראנו בפרק 9.3.2, ה-*LPF* שהזמננו מסין אכן עוזם לבדוק בפרמטרים הנדרשים לפעולת תקינה של המערכת, וכן אכן הזמן מסנן עם תדר קיטוען קטן יותר, כדי שבחן עם השגיאות הקיימות בו הוא יהיה מסנן עם הפרמטרים הרצויים.

פספוס של פולס אקראי

אחר ומהות הפרויקט היא ללהות כל פולס חשמלי (תחת הגדרות של הפרויקט) שנכנס למערכת הקולטת, אנו לא יכולים להשרות לעצמנו לפספוס אפלו batch בודד של דגימות שמרוכז על ידי ה-*SMI*. למעשה כאשר ה-*SMI* משלים את מילוי ה-*batch* בדגימות, הוא מורה על כך ל-*DMA* וה-*DMA* מידית מתրיע על כך למרחב המשתמש. ה-*i-RPi* פועל על מערכת הפעלה של אונחון ובגדירות ברירת המחדל אין אפשרות של הגדרת פסקה (Interrupt) המופעלת ישירות על ידי פריפריה, אלא אם נרשום Kernel Driver ייעודי (לא אפשר בשל לוח זמן לחוץ שמנוע לקבל וሩע בתחום). פסקה מאפשרת להפנות את תשומת הלב של המעבד במידי לאחר שהתקבל batch של דגימות ליחרונו ה-*uncached* על ידי ה-*DMA*. כאשר היא אינה קיימת, אנו משתמשים על זמינות המעבד, וכך השאלגוריתם יפעל במתובן אנו צריכים לספק למעבד חלון זמן גדול מספיק בכך להתפנות לתכנית C ולסרק את ה-*buffer* עד שה-*buffer* השני יתמלא בדגימות. שיטה זו נקראת Polling, והוא אינה איזאילית מאחר והוא תליה בזמןות המעבד, אך מתרעם בדיקה שלנו יצא שאפלו בעומס פעולות, הוא מתפנה לאחר $\text{ms} = 100$ לתוכית המשתמש. כדי להתמודד עם הבעיה אנחנו מגדירים את ה-*batch*-ים בגודל של 16,384 דגימות (2^{14}), וכן אנחנו יוצרים חלון זמן רחב מאוד עבור ה-*Polling* ($\text{ms} = 670$). עד כה לא היה מקרה בו שיטה זו לא עבדה.

Delay בקווים *Data* בין ה-*i-RPi* ל-*ADC*

בפעמים הראשונות בהם ביצענו דגימה של אותן במערכת, אנו השתמשנו ב-*Jumper Wires* באורך של 15 ס"מ לחיבור בין ה-*ADC* ל-*i-RPi*. מעבר לכך שהחיבור רופף ולא הולם לתוצר הסופי, חיווט זה הראה לנו פעמים רבות דגימות מעוותות בנקודות זמן מסוימות, במיוחד במקרה אשר אנחנו את תדר הדגימה. אנו לא בטוחים לסיבת המדיוקת הגורמת לעביעות אלה אך בהכרח מדובר בקibiliים פרזיטיים וחיבור לא איזוטטי רופף של הcabliers. כדי להתמודד עם הבעיה אנחנו תכננו פיסת PCB המתוישבת על ה-socket של ה-*i-RPi* ונכנשת ל-socket של ה-AD9226 מצידו השני כפי שניתן לראות באIOR 9.9.

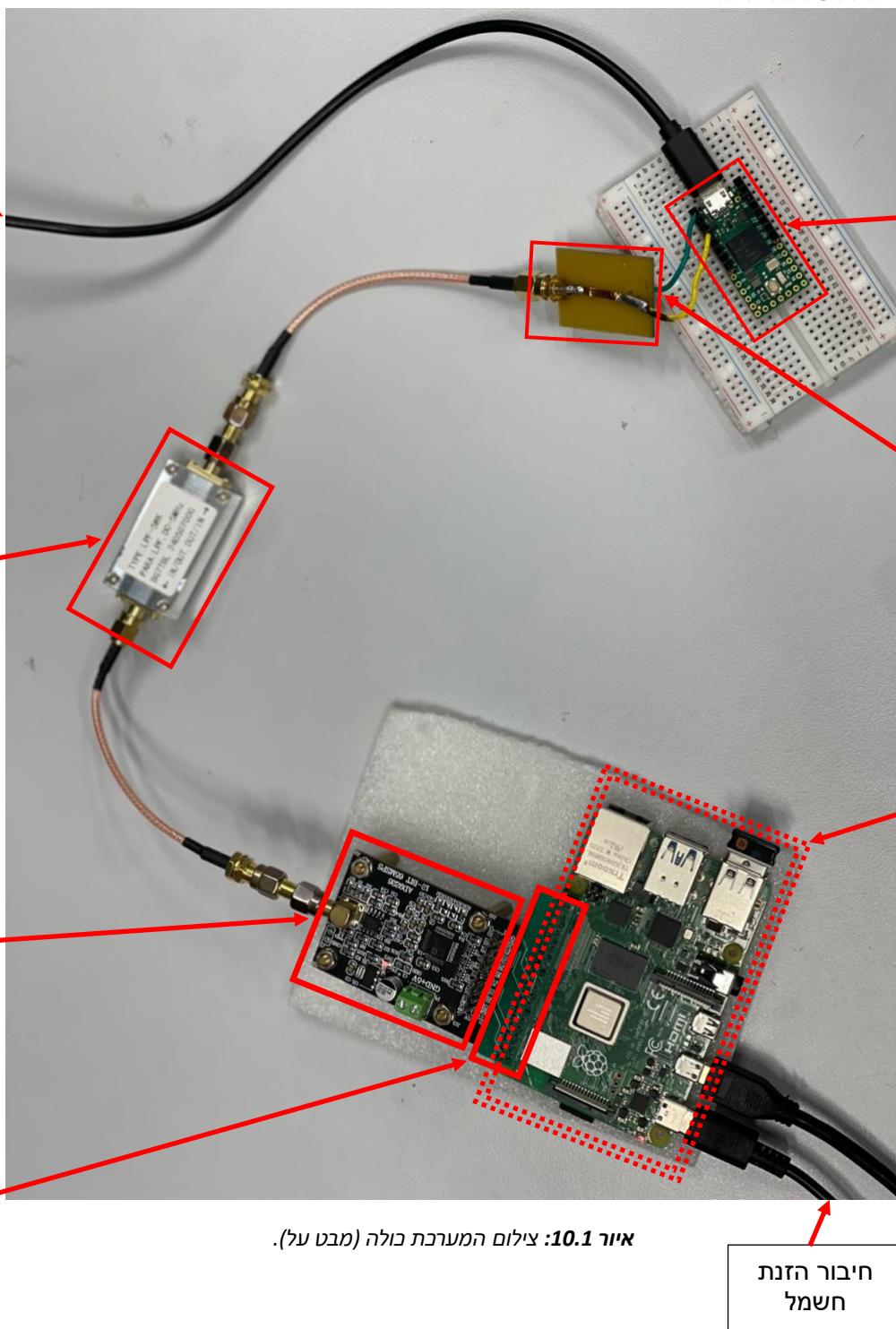


איור 9.9: מבט 3D על ה-*Layout* של ה-PCB.

בעזרת זה אנו קיצרנו את אורכי הקווים ל-2 ס"מ במקסימום, עם הושבה יציבה ואסטטית. בנוסף לכך, דאגנו שהקווים הזנה יהיו עבים יותר בגלל הגדלים ומיניעת התחרמותם ה-PCB. כאשר חיברנו את ה-PCB ל-AD9226 וה-RPi הבעיות שהופיעו לפני כן נעלמו לחולstein.

10. התוצר

10.1 פירוט המערכת



הצלחנו לבנות אב טיפוס ראשון הכלול את כל התתי מערכות שפורטו בפרק 9, באשר ניתן לעיין בפירוט ויזואלי של המערכת באירור 10.1.

חומרה:

- השאלנו מהמכללה את ה-RPi ורכיביו הנלוויים.
- רכשנו חיצונית את ה-Teensy 4.0, מסנן ה-LPF, כבל SMA ומתחאים.
- הרכבנו בידינו ממיר מחוטים רגילים ל-SMA.
- עיצבנו בתוכנת CAD ואז הזמנו במיזוח מיוחד PCB ייעודית לחיבור בין ה-AD9226 ל-RPi.
- חיברנו את ה-Teensy 4.0 למחשב נייד לצוריכת קוד ושליטה על גע שליחת הפולס. בנוסף, חיברנו את ה-RPi למסך מחשב, מקלדת ועכבר, כדי להריץ את תכנית הזיהוי ולראות תוצאות.

תוכנה:

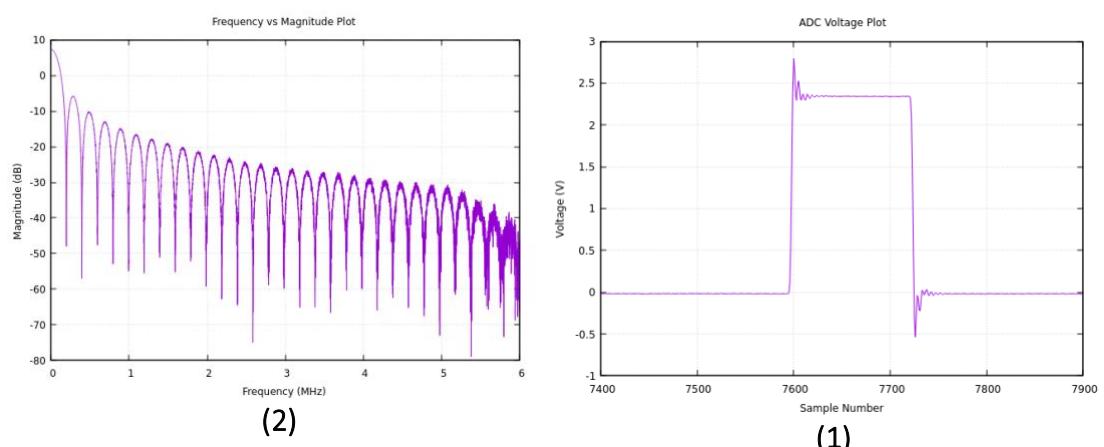
- בtabנו תכנית עבור ה-Teensy 4.0 ששולח פולס בודד בעט לחיצת המסתמש.
- בtabנו תכנית עבור ה-RPi שמתקשר עם מודול הדגימה, מזהה את הפולס האקריאי ומציג את נתוני ביצ'ר הזמן ויצ'ר התדר.
- בtabנו תכנית Debugging ייעודית עבור ה-RPi לבדיקת המערכת.

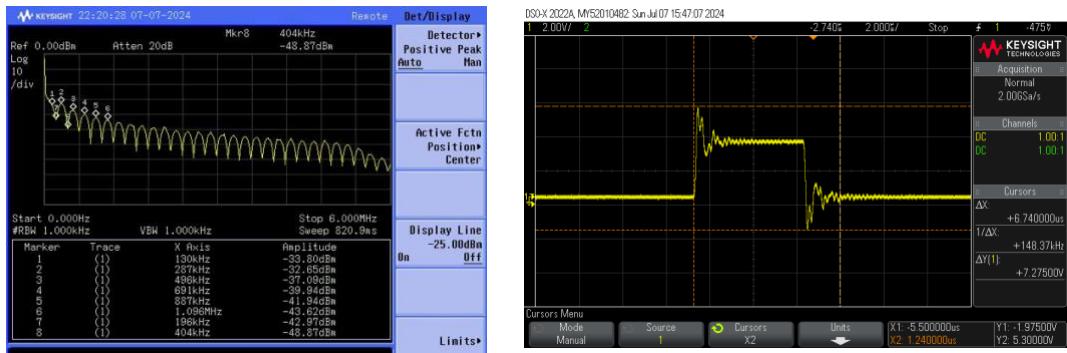
העשיה שמתוארת לעיל דאגה שנעמוד ביעדים ובמדדים שהוצעו ופורטו במסמך ה-WoS. בפרק 10.2 אנו מציגים דוגמאות הריצה של זיהוי פולסים אקריאים ובפרק 10.3 אנו מנתחים את התוצאות ומראים את מידת העמידה ביעדים שהוצבו לפרוייקט.

10.2 דוגמאות הריצה

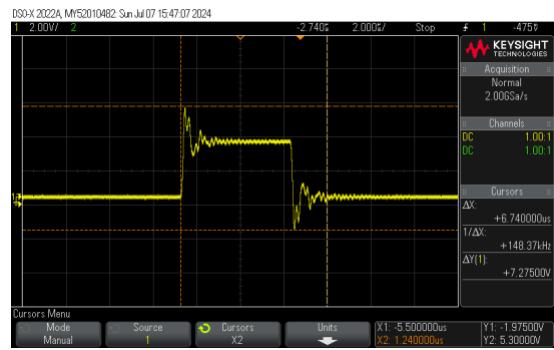
מדוריך המפורט שלב שלב כיצד לפעול את המערכת ולהריץ את התכנית נמצוא בנספח ה'. כדי לקבל רושם מלא על תפקוד המערכת ראיינו לנוכח לחת 3 מקרים של פולסים, כאשר כל אחד ברוחב שונה. כל פולס צזה נשלח על ידי ה-Teensy 4.0 וה-RPi מזהה אותו ומדפיס על המסך מחשב את צורותם ביצ'ר הזמן ויצ'ר התדר. דאגנו לקחת את צילומי הפלסים בסימולציה, ב-Spectrum Analyzer, כדי לקבל רושם ומקור להשוואה ביצ'ר התדר וב-Oscilloscope כדי לקבל רושם ומקור להשוואה ביצ'ר הזמן.

פולס ברוחב 5μ

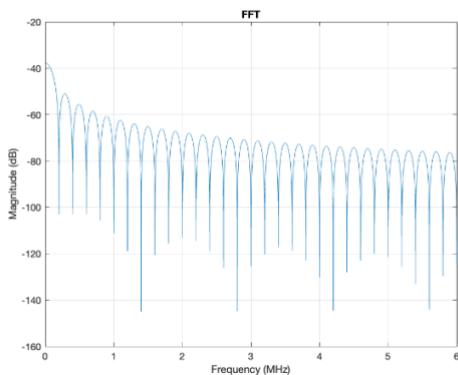




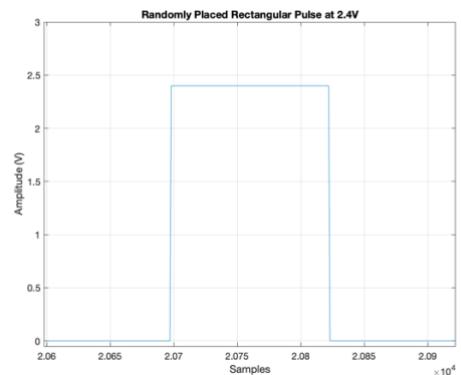
(4)



(3)



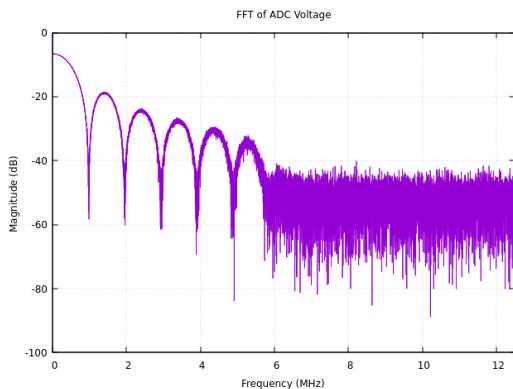
(6)



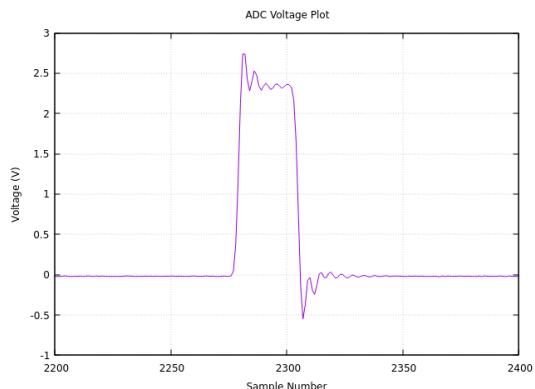
(5)

נוף 10.1: פולס ברוחב μs (1)-(2), RPi-ה (3)-(4), מכשירי מדידה -(5)-(6) סימולציה.

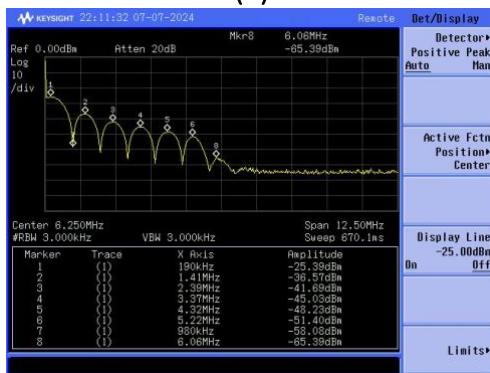
פולס ברוחב μs 1



(2)



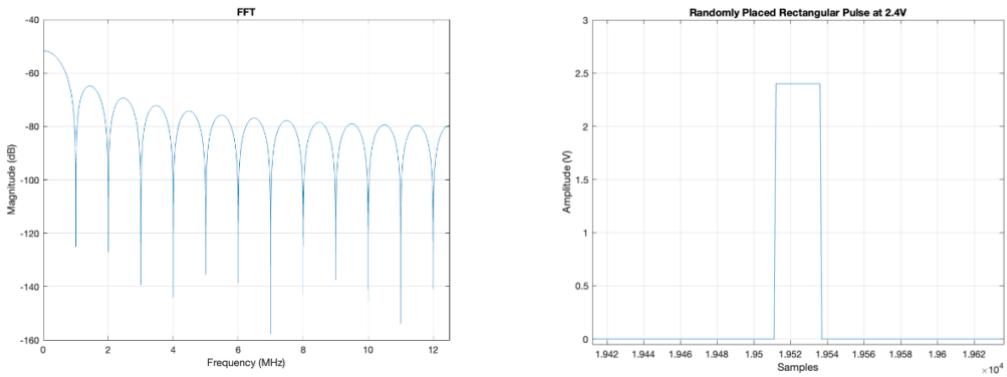
(1)



(4)



(3)

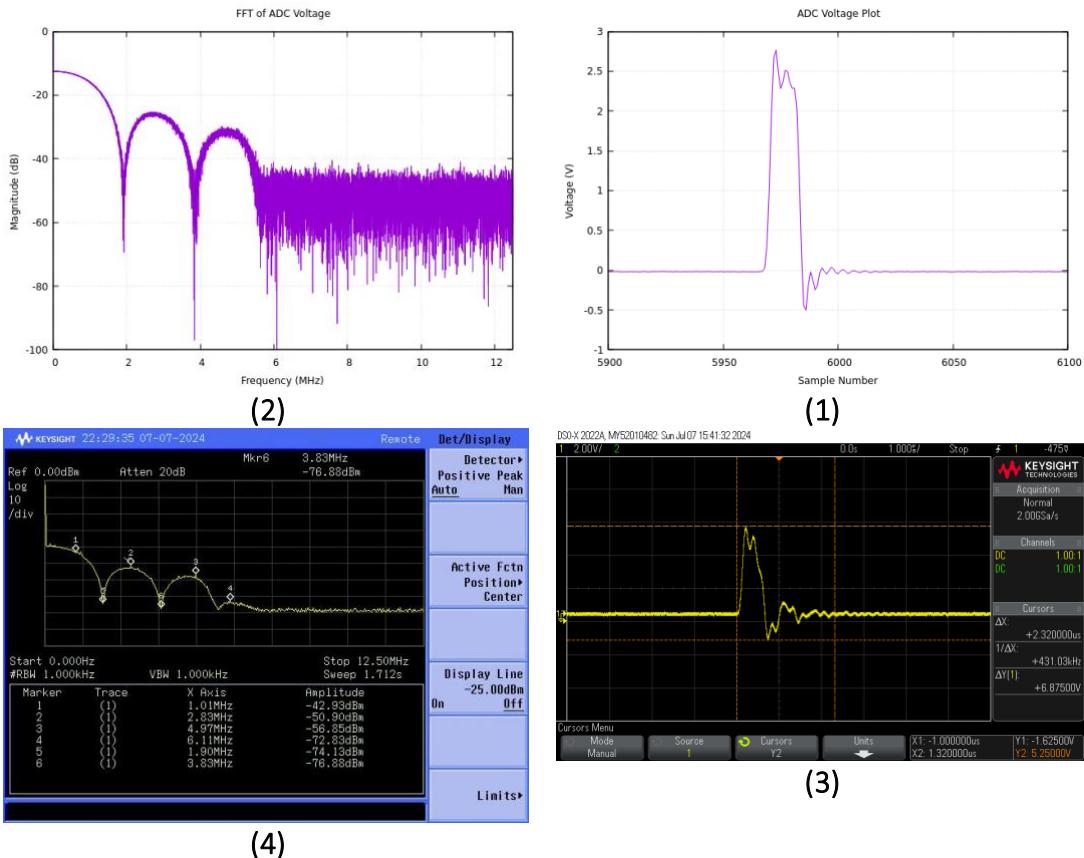


(6)

(5)

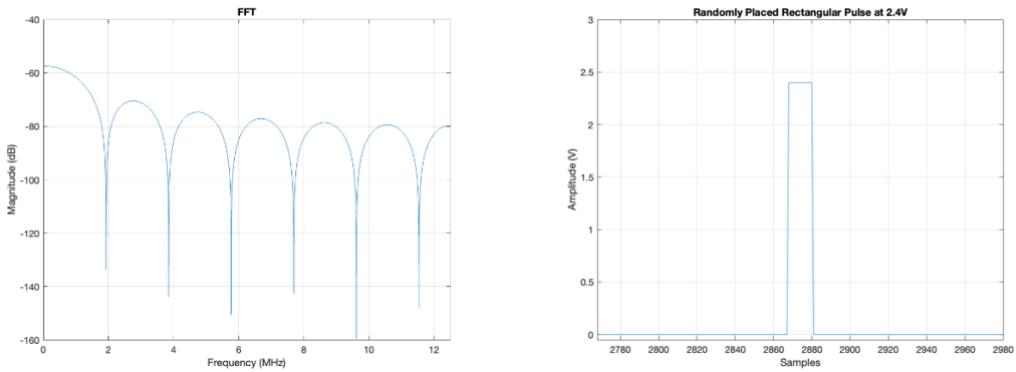
נץ 2: פולס ברווח μs (1)-(2), (3)-(4) מכשורי מדידה ו-(6)-(5) סימולציה.

פולס ברווח μs 500



(4)

(3)



(6)

(5)

גוף 3.10.3: פולס ברווח **ns 500**, (1)-**RPi**, (2)-(3) מכשירי מדידה ו-(4)-(5) סימולציה.

חשיבות לציין כי הסימולציה שנלקחה עבורי כל אחד מהפולסים מוקהה אידיאלי, ללא שום רעשיהם, כאשר הפולס הוא פולס מלכני אידיאלי שננטפס במקומם אקרראית בתוך ה-FFT של ה-32,768 דגימות (כמו ב-**RPi**), שכן הספקטורים המוחשב ע"י Matlab הוא תואם את התאוריה, ובפרט מיישם את דרכי החישוב שהסבירו בפרק 9.1.7. בנוסף לכך, אנו לא כולים בסימולציה את השפעת המשן, וזאת לאחר ואנו נרצה להבחן בפרק 10.3 בהשפעת המשן על האות.

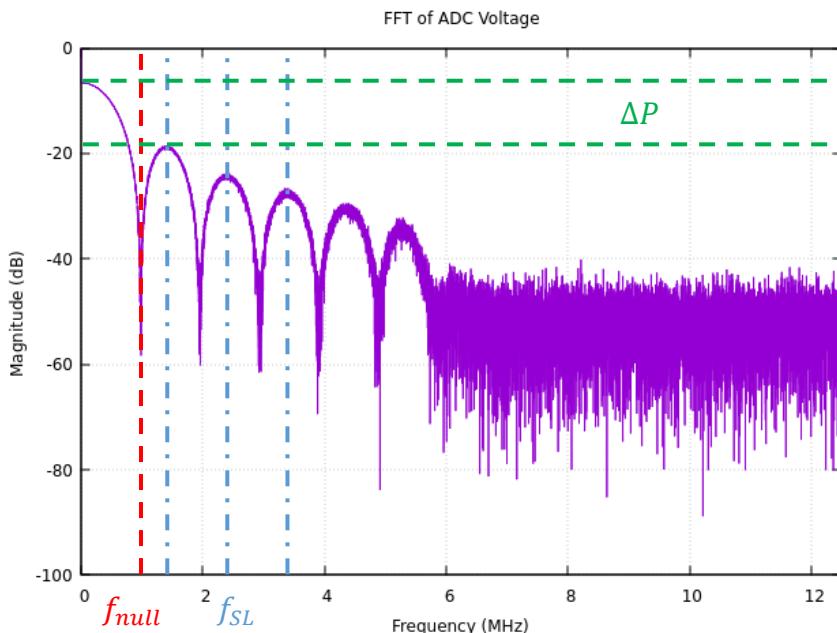
10.3 בדיקות והערכתה

10.3.1 בדיקת דגימה ועיבוד

כדי לקבל ממדד ממשי על איזיות הפרויקט ומערכות היזיהו והעיבוד בפרט, אנו נשווה את קודם כל המאפיינים השונים של הפלסים שזווחו בפרק 10.2 והוצאו בגרפים 10.1-3. גם, בעת אנו עוסקים באיזיות הדגימה והעיבוד ובמה שארנו נערך את איזיות היזיהו של המערכת. נזכיר כי לצורך השוואת ממדדים את אותם הפלסים שזווחו על ידי המערכת גם במקשיי מדידה הזמינים במכלול וגם ביצענו סימולציה שלהם. המאפיינים שראינו לנכון לבחון אותם, הם אלה שמאפיינים פולסים, לדוגמה התדרים המרכזיים של אוננות הצד (side lobes). החלטנו שגורם הטיב בבדיקה של המערכת תהיה רמת הסטיאו של לנו במאפיינים בין מכשיי המדידה המשחררים והיקרים הזמינים במכלול.

מרוחב התדר:

- תדר של האפס (null) הראשון (f_{null}).
- תדרים מרכזיים של אוננות הצד (f_{SL}).
- הפרשים בהספק בין נקודות המקסימום באוננות (ΔP).

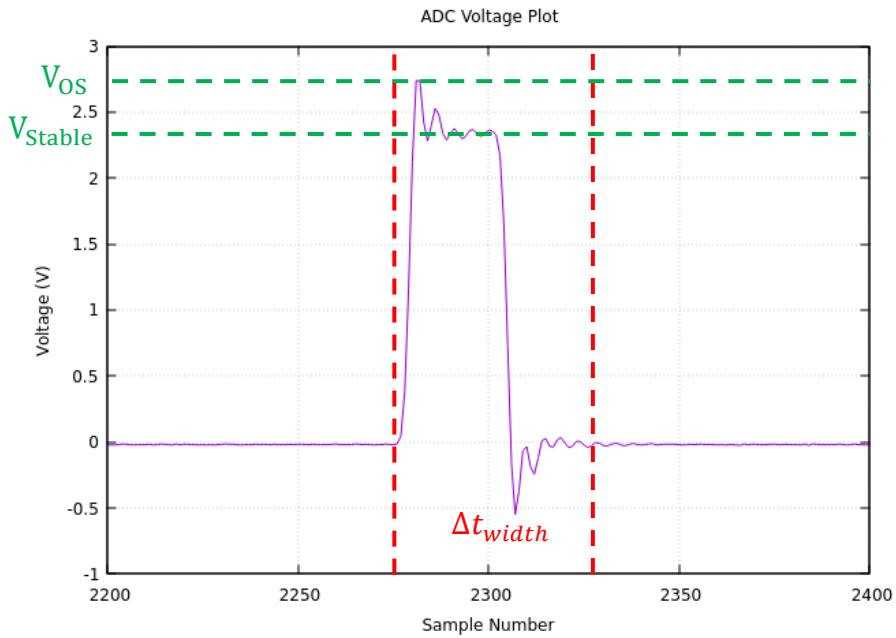


graf 4.10.4: פרמטרי בדיקה במרוחב התדר.

חשוב לציין כי עוצמת magnitude בספקטרום אינה משמעותית ויכול להיות שונה לכל מערכת מדידה בהתאם לנקודת הייחוס. הפרט העיקרי הוא ההפרשנים בmagnitude.

מרוחב הזמן:

- רוחב אבסולוטי של הפלס, מרגע העלייה עד החזרה והתייצבויות כמעט מוחלטת (Δt_{width}).
- רמת מתח בנקודות overshoot ו/או רמת מתח היציבה (V_{OS} ו- V_{stable}).



גרף 10.5: פרמטרי בדיקה בזמן.

טבלה 10.1: מדדים עבור פולס ברוחב $5 \mu s$.

V_{stable} [V]	V_{OS} [V]	Δt_{width} μs	ΔP [dB]	f_{SL} M[Hz]	f_{null} M[Hz]	
2.344	2.795	5.44	13.65	0.2838	0.1991	מערכת שלנו
			4.37			
			2.85			
3.32	5.3	6.74	-	0.287	0.1960	մեթոդ մետրի
			4.44			
			2.84			
2.4	-	5	13.3	0.291	0.1998	סימולציה
			4.53			
			3.16			

טבלה 10.2: מדדים עבור פולס ברוחב $\Delta t = 1 \mu s$.

V_{stable} [V]	V_{OS} [V]	Δt_{width} μs	ΔP [dB]	f_{SL} M[Hz]	f_{null} M[Hz]	
2.3	2.738	1.48	12.02	1.392	0.985	מערכת שלנו
			5.38			
			3.06			
-	5.3	2.3	-	1.41	0.98	մեթիր մազակ
			5.12			
			3.34			
2.4	-	1	13.22	1.458	1	Սիմուլացիա
			4.48			
			2.82			

טבלה 10.3: מדדים עבור פולס ברוחב $\Delta t = 500 ns$.

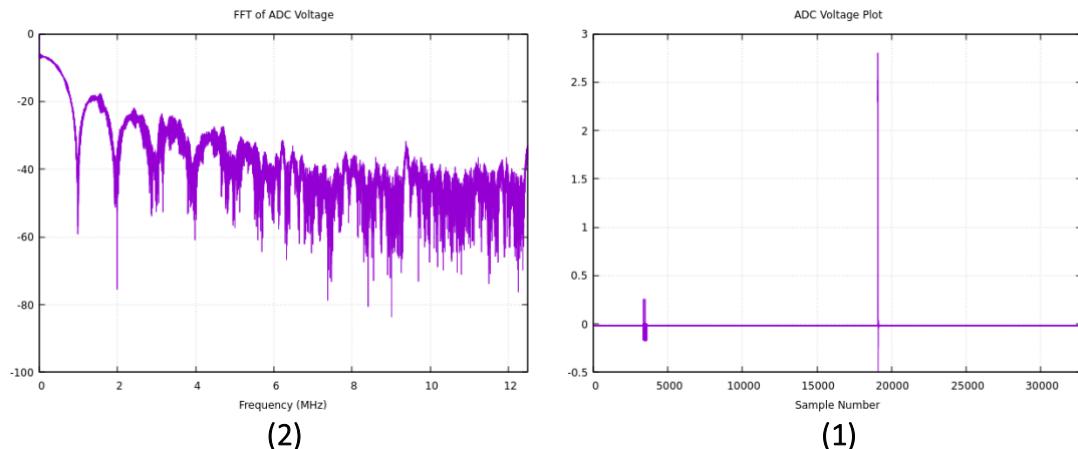
V_{stable} [V]	V_{OS} [V]	Δt_{width} μs	ΔP [dB]	f_{SL} M[Hz]	f_{null} M[Hz]	
-	2.765	0.88	13.01	2.83	1.922	מערכת שלנו
			5.3			
			-			
-	5.25	1.15	-	2.83	1.9	մեթիր մազակ
			5.95			
			15.98			
2.4	-	0.5	13.1	2.76	1.923	Սիմուլացիա
			4.2			
			2.43			

מ托וך התוצאות הכתובות בטבלה 10.1-3 נובע יישורות כי הפולס שמצויה במערכת שלנו הוא בעוצמה של $V \sim 2.3$ ~ כאשר בմեթיר המזיהה הם $V \sim 3.3$. מדובר בהפסדים בקווים ובממשק ההמרה לבבל ה-SMA, לאחר וכאשר חיברנו מחולל אותן יישורות למערכת הזיהוי אז לא חווינו הפסדים כלל. הפסד כפי שהוא הוא 30% ברמת המתח, ולמרות זאת אנחנו רואים דמיון רב בצורת הפסדים בזמנו בין גרפים 10.1.1 ו- 10.1.3 ו- 10.3.1 ו- 10.3.2. אנו צריכים לזכור כי ה- Oscilloscope שמדדנו בעזרתו את הפולס, יכול לדגום בקצב של GSPS 2 עם יותר רמות קוונטיזציה, מכיוון שאם נסתכל בגרפים 10.2.1 ו- 10.2.3 אז נראה צורה קצרה שונה של הפולס (במיוחד במצב היציב) מאחר והמערכת שלנו דוגמה בסביבות ה-40 דגימות של הפולס, כאשר מטבחי המזיהה ללחוץ הרבה יותר וגם ברזולוציה יותר גבוהה. דבר דומה ניתן להגיד גם על רוחב הפולס שנמדד, מאחר ויתכןנו תנודות נוספות באות שהמערכת שלנו לא מסוגלת לחתות מה-Oscilloscope בן מסוגל.

דבר ראשון שאנו שמים אליו לב כאשר אנו מנהנים את התוצאות של הספקטרום שהתקבל (במיוחד בגרפים 10.2.2 ו-10.3.2), אנו רואים הנחתה חזקה של המסנן LPF בתדרים מעל ~ 5 MHz, מה שמצוור על פעולה אמינה כ-AAF. בידע לנו מהתאוריה מדר האפס הראשון של ה- c_{chos} הוא ההופכי של רוחב הפולס. אנו יכולים לראות בתוצאות של f_{null} ב_TBLAOOT 10.1-10.3 כי הסטייה בין המערכת שלנו, מבשרי המדידה והסימולציה היא מינורית ומושפעת בעיקר מה-RBW של המערכת שלנו וה- f_{LPF} , אשר דומה ניתנת להגדיל על גבולה הרובה. דבר זה מוכיח שפונקציית f_{null} מושפעת מהתוצאות של ה- c_{chos} . אנו מקבלים תוצאות מספקות בין מערכת המדידה שלנו לבין ה-SA, לאחריהם הם אחת התכונות העיקריות של ה- c_{chos} ומהוות פרמטר חשוב בניתוח הפולסים. הסטייה העיקרית ב- P_{Δ} מתאפשר כאשר אנו משווים את ערכי המערכת שלנו ושל ה-SA לטימומצילה, וזה לא מפתיע אותנו מאחר והסימולציה מנהנת פולס מלכני מושלם (אות לא סיבתי), لكن יש מקום לשגיאה. לסיום, היכולת שלנו בניתוח ספקטרום הפולס שמצוור על ידי המערכת היא מספקת.

10.3.2 בדיקת איכות הדיזיין

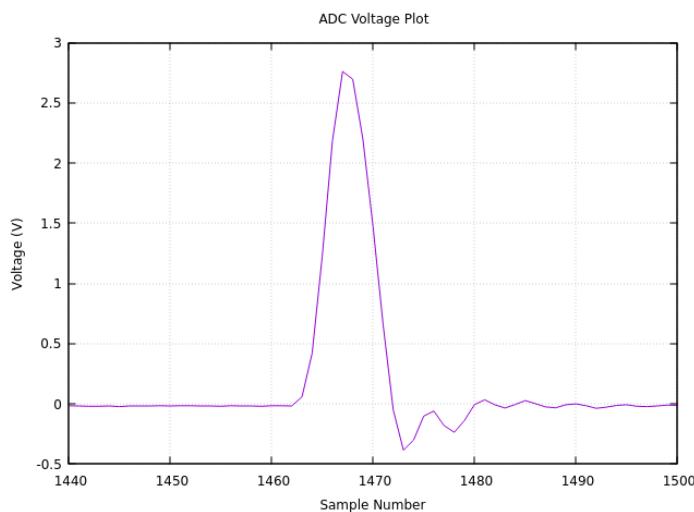
כדי לבדוק את יכולות של מערכת הדיזיין מבחינת היכולת לתפוס פולס אקרים, אנו עשינו מספר בדיקות שיישרתו את שולרי המערכת שלנו. קודם כל, חשוב לציין שלאור כל הפעולה שלנו עם ה-DMA ו-SMI, כאשר עשינו אופטימיזציה של הפרמטרים שלהם בשיטת ניסוי וטעיה, לא נפגשנו עם מצב בו המערכת פספה דגימה או חלה בה בעיות זמן. ישנה תופעה אחת שקיים במערכת שלנו, והיא לעיתים מסוגלת לגרום ל-FA (False Alarm) בדיזיין הפולס. דוגמא לתופעה זו ניתן לראות בגרף 10.6.



גרף 10.6: (1) פולס ייחודי עם jitter לא ידוע, (2) מרחב התדר של הפולס במערכות jitter.

כאשר אנו רואים בדגם 19,000 בגרף 10.6.1 את הפולס הרצוי ברוחב ~ 1 , אך בערך בדגם 4,000 אנו רואים jitter לא ידוע שבහלט משפייע על חישוב הספקטרום של התוצאות בחולון זמן שהתקבל וניתן לאות זאת בגרף 10.6.2. התופעה זו אינה שכיחה, אך כן הבחננו בה ודאגנו להימנע מזיהוי שווה על ידי בהגדרת המתח סף זיהוי ~ 1 , כאשר ניתן עוד להנמיך אותן. בנוסף לתופעה זו, יש תופעה נוספת אף הרבה יותר פחות שכיחה והיא מתאפיינת בדגם בודדת שמתאפשרת בתור spike מואוד חזק, כאשר שאר הדגימות מסביבה הן ברמת הרעש. כדי להימנע מזיהוי שווה בגל תופעה זו, אנו הגדרנו בקוד פרמטר שמאפשר להציג רוחב מינימלי של פולס מעל ה-threshold. אנו הגדרנו את הפרמטר בקוד בין 5-2, ונמנענו מהתופעה זו לゲמרי. אנו לא מודעים למקור התופעות הללו, אך אנו מודעים אליהם ונתקטו באמצעות מנגנונים מתאימים.

בהתאם לפרמטר אותו ציינו לעיל, אנו למשה מגדרים את רוחב הפולס המערכת שלנו מסוגלת לזהות. לאחר בדיקת רוחבי פולס שונים, אנו ראיינו שהסף של זיהוי jitter של כל פולס שנשלח אליו מופיע ברוחב של ~ 200 , כפי שניתן לאות בגרף 10.7.



גרף 7: הציגה של ספַּדְזָהוּ הוֹוְדָאִי שֶׁל פּוֹלֵס אֲקָרָאי.

אנו יכולים לראות ויזואלית בגרף 10.7 כי הדגימה כבר אינה איבוטית כמו עבור פולסים היותר רחבים, ואני בהחלט מפספסים לא מעט פרטים על הפולס. לאחר בדיקות שביצענו, רוחב יותר צר $m\text{-}\tau = 200$, איןנו וודאי שנצליח לזהות את הפולס במערכת שלנו. חשוב להזכיר חישוב שבוצע בפרק 9.1.6, שקבע את רוחב הפולס המקסימלי שהמערכת מסוגלת לזהות, והוא לנו כי הרוחב המקסימלי הוא $ms = 1.31$, لكن המערכת שלנו מסוגלת לזהות בוודאות פולסים אקראים בתחום רחב ומספק של ארבעה סדר גודל.

11. דין

11.1 הערכת כלילת התחנלות הפרויקט

היעד המרכזי היה לפתח מערכת למדידה וניתוח של פולס אקראי שתעמדו ביעדים כמו יצירת פולס אקראי ברוחב של מיקרו-שנייה, ניתוח האות בזמן אמת והצגה ברמה גבוהה של האותות. במהלך הפרויקט נתקלנו באתגרים טכניים ותיאורטיים שככלו אופטימיזציה של החומרה לעיבוד הפלסים בזמן אמת, פיתוח התוכנה שתוכנן לתמוך עם עיבוד האותות בזמן אמת תוך ניצול משאבי המערכת ביעילות. חלק מהאתגרים היו גם כיצד להפחית את עלות המערכת תוך שימוש על ביצועים טובים ואמינות ברמה גבוהה.

תכנון והגדרת יעדים

ההחלטות שהתקבלו בשלב התבונן התרבשו על שילוב של מחקר תאורטי ובדיקות מעשיות.

- מיקרו-בקר 4.0 : נבחר בשל ביצועו הגבוה והיכולת שלו להתמודד עם יצירת פולסים בצורה מדויקת.
- ADC AD9226 : נבחר בשל יכולות הדגימה המהירות שלו ודיקן הרזולוציה, מה שמאפשר מדידה ברזולוציה של 12 סיביות בתדר מקסימלי של [MSPS] 65.
- מסנן LP : נדרש למניעת Aliasing ולהבטחת איכות האות הנמדד.
- B Model 4 RP : נבחר בשל זמינותו של מגנונים וממשקים חומרתיים המאפשרים דיזיין ועיבוד הפולס.

ביצוע ומעקב

שלב הביצוע היו כמה שלבים מרכזיים:

- איסוף הרכיבים: הזמן ובידקה הרכיבים (זמןה ובידקה הרכיבים (LPF, Teensy 4.0, ADC AD9226 וכו'), כל רכיב נבדק ונבחר על פי קритריונים כמו עלות, ביצועים, גודל ויכולת התאמה למערכת הכלולית.
- בדיקות מעבדה: בדיקות ראשונית של כל רכיב בנפרד ולאחר מכן בדיקות אינטגרציה בין הרכיבים כדי לוודא את תפקודם המשותף.
- פיתוח תוכנה: יצירת תוכנה בשפת C שתפקידה לעבד את האותות הנקלטים ב-Raspberry Pi 4 Model B וbijouterie האנליה בזמן אמת.

11.2 ניתוח הבחירה הטכנית והמדעית

בחירה החומרה

מיקרו-בקר 4.0

- יתרונות – תדר שעון גבוה של ZHMH00600, גודל קטן ועלות נמוכה.
- שימוש – מתאים במיוחד לצירת פולסים אקראיים בשל יכולתו לשימוש בדיקון רב על מוצא ה-PWM, ובכך ליצור פולסים ברוחב של מיקרו שנייה ומעבר.

- יתרונות – רזולוציה של 12 סיביות, קצב דגימה גבוה של [MSPS] 65, אינטגרציה קלה עם מערכות מבוססות מיקרו-בקה.
- שימוש – דוגם את הפלסים הנקלטים וממיר אותם לאוטות דיגיטליים בצורה מדויקת, התומכת בניתוח תדרים ברמה גבוהה.

Raspberry Pi 4 Model B

- יתרונות – עצמת עיבוד גבוהה, זמינות של 40 פיני GPIO, תמייה ב- DMA ו- I2S, ויכולת לעיבוד נתונים בזמן אמיתי.
- שימוש – משמש כיחידת הדיזי ועיבוד המרבי לצורך ניתוח האותות והציגתם.

בחירה מדעית

בפרויקט נעשה שימוש בשיטות מתקדמות לעיבוד אותות:

- התמרת פורייה (FFT) – מיושמת לצורך ניתוח התדרים של הפלסים הנקלטים, מאפשרת לראות את הספקטרום בצורה ברורה.
- פילטרים – מסנן LP המשמש כAAF למניעת Aliasing ושימור איותאות.
- זהויות – עבדה עם פולסים אקראים מחיבת יירה של מנגןונים זהוי עבור מגוון של רוחבי פולסים ועוצמות.
- תדר Nyquist – דגש על עמידה בתנאיNyquist ב כדי למנוע קיפולי תדר (Aliasing).

11.3 השוואת התוצאות לתיאוריה

בכדי לקבל ממד על איותות מערכת זהוי ועיבוד אותות ביצענו השוואה בין תוצאות המערכת שלנו לבין התיאוריה ולמדידות במערכות שונות הקיימות בשוק (מודגם בהרחבה בפרק 10.2 ופרק 10.3).

בפרק 10.2, ניתן לראות השוואה בין תוצאות המערכת לבין התוצאות אשר קיבלנו כאשר מددנו את אותם האותות ב-Spectrum Analyzer אשר נמצא במקללה. ניתן לראות שם שאנו מקבלים בעזרת המערכת שלנו תוצאות דומות מאוד כאשר מודדים ומעבדים פולסים ברוחב [ns] 500 ועד [ns] 5, המערכת שלנו יכולה לקבל את אותה רמת דיוק גם עבור פולסים בגודל [ns] 200 ועד פולסים ברוחב [ms] 1.3, ככלומר, המערכת שלנו יכולה להוות בודדות פולסים אקראים בטוחה של ארבעה סדר גודל.

בහיבט של יכולות הדגימה, ניתן לראות בטבלאות 10.1, 10.2 ו- 10.3 שההשוואה למדידות במכשירי המדידה שבמקללה, קיבלנו תוצאות מציניות אשר משקפות בצורה טובת מאוד את תוצאות הסימולציה. בנוסף, ניתן לראות בפרק 10.3.2 שבחשבדנו את איותות הזהוי של המערכת ראיינו שבמספר רב של דגימות שביצענו, לא נפגשנו במצב בו המערכת פספסה דגימה או שהיה בה טעויות תחזמן, מה שמראה על דיוק רב ועל בחירת הרכיבים ופרמטריהם בצורה מדויקת.

11.4 שיפורים ולקחים

11.4.1 שיפורים טכנולוגיים

במהלך הפרויקט, חלק מהתכнולוגיות שנבחרו עמדו ביציפיות, אך הופיעו גם אתגרים שדרשו פתרונות יצירתיים. ניתן לשפר את המערכת בכמה אופנים:

- **יעילות העבודה –** שיפור האלגוריתמים שימושיים לניטוח האותות יכול לקטר את זמני העבודה ולהגדיל את יעילות המערכת.
- **חומרה משופרת –** החלפת רכיבי החומרה ברכיבים מתקדמים יותר, כגון ADC עם רזולוציה גבוהה יותר, עשויה לפחות את דיקן המדידות ותפקיד המערכת.
- **ממשק משתמש –** שיפור המשטמש יכול להקל על התפעול ולהגביר את נגישות המערכת למשתמשים לא מקצועים.

11.4.2 ליקויים מהפרויקט

התאמות והתמודדות עם בעיות טכניות:

- **אתגרי אינטגרציה –** אחד אתגרים שנתקלנו בהם היה השימוש הנכון של המיקרו-ברקר, ה-ADC וה-RPi. לא היה ברור לנו מלכתחילה כיצד לבצע את החיבורים בצורה הטובה ביותר, נתקלנו בבעיות טכניות במיוחד בתחום החיווט והסנכרון בין הרכיבים. חווינו בעיות של השתקפות של אותות ורעשים בתדרים הגבוהים, שהצריכו חיווט ושימוש בכבלים מתאימים יותר.
- **פיתוח תוכנה –** פיתוח התוכנה לעיבוד האותות הוכח כמאתגר מהצפוי. התמודדנו עם בעיות הקשורות ליברנון, היה צורך באופטימיזציות רבות בקוד בכספי להשיג עיבוד איבוטי בזמן אמיתי. תקשורת בין הרכיבים – גילינו שהתקשרות בין ה-ADC ו-SPI, שהיא אמורה להיות פשוטה יחסית, נתקלה בעקבות תזמון, במיוחד כאשר ה-ADC נדרש להעיר מידע במתוירות גבוהה ל-RPi. הפתרון היה שימוש נרחב יותר ב-DMA שהפחית את העומס על המעבד המרכזי וסייע ביצועים.

ניהול הפרויקט ותדרוניים:

- **תיכון מוקדם –** גילינו שהעבודה שתכננו מראש זמן רב לכל שלב בנייסוי עדר לנו מאוד, הצלחנו לעמוד במשימות בצורה שרצינו בצורה טובאה, מה שהפרק את הפרויקט לטוב יותר ופחות מלחץ. לדוגמה, לתהיליך רכישת הרכיבים הקಡשנו זמן רב בתכנון, אך הרכיבים הגיעו מהר מהצפוי מה שאפשר לנו להתחיל לבדוק אותם ולבצע ניסויים מוקדם מהצפוי.
- **חשיבות בדיקות מוקדמות –** במהלך העבודה על המערכת, בדקנו את הרכיבים שהזמננו וקיבלו מהמכללה בנפרד בשלב מוקדם ולפניהם שמערכת הורכבה למערכת אחת אינטגרלית. עבדה זו אפשרה לנו להחליף רכיבים לפי הצרכים שלנו בהתאם לתוצאות שקיבלנו ובכך מנענו תקלות לא צפויות שיובילו לקרוות.

11.5 הפוטנציאל המסחרי והיישומים העתידיים

לפרויקט יש פוטנציאל ממשוני בשוק בזכות הפיתוח של מערכת יעילה וזולה לניטוח פולסים אקראיים.

ישומים תעשייתיים:

המערכת שפיתחנו יכולה לשמש לבדיקות בתעשייה שבנה נדרש זהה מדויק של אורות חשמליים, כגון בתחום המוצרים האלקטרוניים, רכב ובתעשייה הchlל. היכולת להזות פולסים אקראים בזמן אמת תוך שימוש בטכנולוגיה זולה ויעילה יכולה להוביל לשיפור בקרת האיכות וליהו מוקדם של תקלות.

פוטנציאל מסחרי:

על מנת להבין את הפוטנציאל המסחרי, ניתן לשקל שיתופי פעולה עם חברות בתעשייה האלקטרונית, אשר יכולים להטמע את הטכנולוגיה במוצרים שלהם. היכולת להזות ולנתח פולסים עדינים ומהירים יכולה גם להיות שימושית ביישומים רפואיים. למשל, במעקב אחריו פעילות הלב או המוח. הטכנולוגיה זו יכולה להתאים גם למכשירי ניתור חדשניים שדורשים זהה מהיר ומדויק של שינויים בפעולות החשמלית של גוף האדם.

12. סיכום וסיכום

מבוא

הפרויקט נולד מ הצורך לפתח כלי זמין ויעיל לאבחן פולסים אקראים בתעשייה שומנת. ישנו פתרונות שונים בתעשייה להתקודדות עם פולסים אקראים, חלון חומרתיות וחלאן בשימוש בתוכנה. בפרויקט זה תכננו ובנינו מערכת אשר משתמשת הן בחומרה והן בתוכנה על מנת ליצור, לדגום ולנתה פולסים בלבד.

תיאור המערכת

המערכת שפיתחנו מבוססת על שילוב של חומרה ותוכנה שנבחרו ונבנו בקפידה. בצד החומרתי כלנו RPi 4 Model B שנבחר בשל גמישותו וממשקיו ה-I/O הרים, את מודול ה-ADC מדגם AD9226, בחרנו בשל היכולת להפיק דגימות ב מהירות וברזולוציה גבוהה. לצורך שימוש באלגוריתמים לצורך זיהוי וניתוח מהיר של הפולסים שהתקבלו בתבוננו קוד בשפת C, תוך שימוש מתקדמים אשר פותחו במיוחד לצורכי הפרויקט.

שלבי הפרויקט והאתגרים

במהלך הפרויקט, עברנו שלבים שונים שככלו תכנון, בניה, ניסויים ושיפורים. אחד האתגרים העיקריים איתם התמודדנו היה השגת רזולוציית דגימה גבוהה תוך שמירה על זמן תגובה מהיר, רכיבים זולים ונגישים ומדדים קטנים יחסית. פתרונו זאת באמצעות תוכנה אופטימלית אשר יכולה להתמודד עם כמות גדולה של נתונים בזמן אמת.

הישגים ותוצאות

המערכת שפותחה הוכיחה את עצמה ככלי יעיל ומדויק לזיהוי פולסים אקראים. בבדיקות שביצענו ראיינו שהמערכת מסוגלת לזהות אותן בرمאות רעש שונות ולהפיק מהם נתונים מדויקים וموעילים לתהליכי בקרה ותחזקה בתעשייה.

- **הפקת פולס** – בתחילת הפרויקט התchingינו ליצר פולס ברוחב של 5nm. בפועל המערכת שלנו הצליחה לעמוד בדרישה זו בדיק גבוה מאוד. הבחירה-b-Teensy הייתה בחירה מצוינת, הרכיב נותן לנו גמישות רבה בהפקת האות וביעילות ודיוק גבוהים מאוד.
 - **ניתוח והתרמה** – התchingינו ליצר מערכת אשר תנתה את האות, תבצע התרמת פוריה ונתנה את התוצאות. המערכת שביצינו הוכיחה יכולות דגימה, עיבוד וניתוח מעולות, המערכת מבצעת התרמת פוריה בדיק גבוה. האלגוריתמים שפותחו הצליחו לנתח את האותות בזמן אמת ולהציגם בצורה ברורה ומובנת לשימוש.
 - **הצגת האותות** – המערכת שפיתחנו בעלת משקל משתמש אינטואיטיבי ונוח המציג את התוצאות בצורה ברורה ואיכותית. הציגת כוללת גרפים ונתונים חשובים של האותות הן בזמן זהן בתדר בצורה פשוטה להבנה לשימוש.
- לסיום, ניתן לראות מהשואות המטרות לنتائج בפועל שהפרויקט לא רק הצליח לעמוד במידדים שהגדרנו, אלא אף עבר את הציפיות בכל חלק מהאפקטים. המערכת הסופית היא בעלי חזק ומדויק לניתוח פולסים אקראים, המספקת פתרון יעיל ונגיש למשתמשים בתעשייה ובמחקר.

לקחים ותוצאות

העבודה על הפרויקט לימדה אותנו על החשיבות העצומה של גמישות בתכנון והבנה לשינויים במהלך פיתוח פרויקטים מורכבים.

הפקנו תובנות חשובות לאורך של שלבי הפרויקט, בעיקר לגבי כיצד>Create פיתוח צריך להיות מוכן ומסוגל להתמודד עם אתגרים בעלי צפויים. הפתרון לכך הוא תכנון מערכות גמישות, כך שגם אם יוחלט תוך כדי פיתוח הפרויקט על שינוי מהותי בכל شيء, שאר חלקי המערכת יהיו מתוכננים כך שיידועו לעבוד בצורה אינטגרלית עם החלק שונה מבלי שהוא צריך בתכנון מחדש של כל חלק הפרויקט. דברים נוספים שיכולים לעזור בעיות מסווג זה הם חסיבה יצירתיות, תכנון מוקדם של חלופות והיכולת לשנות ולהתאים את תהליכי העבודה בהתאם לנרטונים משתנים.

מעבר לכך, הבנו עד כמה חשוב השקיע בשלבי התכנון המוקדמים. התובנה שרכשנו היא שעבודה יסודית בשלבים אלו יכולה למנוע בעיות רבות בעtid ולחסוך זמן יקר במהלך הפרויקט. לכן, בפרויקטים עתידיים נדע להקדיש זמן רב יותר לפיתוח אמצעים ושיטות עבודה שיאפשרו לנו להתמודד בצורה טובה יותר עם שינויים ודרישות שונות שעשוות להופיע במהלך העבודה.

בקשר זה, אנו גם מבינים בעת את חשיבות המעקב והביקורת השוטפת על התקדמות הפרויקט. שיטות כגון בדיקות תקופתיות ושימוש בכלים טכנולוגיים לניהול פרויקטים הם מפתח להצלחה בפרויקטים מסובכים הכוללים שלבים רבים ותת מערכות שונות.

13. רשימה מקורה

- [1] Turkmanovic, H., Karlicic, M., Rajovic, V., Popovic, I. (2023) High Performance Software Architectures for Remote High-Speed Data Acquisition. *MDPI*.
- [2] AD9222 datasheet, Analog to Digital Converter, Analog Devices, 2017.
- [3] Gupta, S., Kumar, S. (2018) Data Capture Via High-Speed ADC's using FPGA. IEEE 2nd International Conference on Micro-Electronics and Telecommunication Engineering
- [4] DEWESoft. Types of ADC Converters [Internet]. DEWESoft. Available from: <https://dewesoft.com/blog/types-of-adc-converters> [Accessed 13 February 2024]
- [5] NewHaven Display International. Serial vs Parallel Communication [Internet]. NewHaven Display International. Available from: <https://newhavendisplay.com/blog/serial-vs-parallel-communication> [Accessed 13 February 2024]
- [6] ADS825 datasheet, Analog to Digital Converters, Texas Instruments, 2002.
- [7] AD9226 datasheet, Analog to Digital Converter, Analog Devices, 2000.
- [8] Arduino Docs. (2023) Secrets of Arduino PWM [Internet]. Arduino. Available from: <https://docs.arduino.cc/tutorials/generic/secrets-of-arduino-pwm> [Accessed 4 February 2024]
- [9] Arduino Docs. Arduino Uno R3 Product Reference Manual [Internet]. Arduino. Available from: <https://docs.arduino.cc/hardware/uno-rev3> [Accessed 4 February 2024]
- [10] STM32-base. Blue Pill [Internet]. STM32-base. Available from: <https://stm32-base.org/boards/STM32F103C8T6-Blue-Pill.html> [Accessed 14 February 2024]
- [11] ESP32 datasheet, ESP32 Series, Espressif Systems, 2024.
- [12] ADS4129 datasheet, Analog to Digital Converters, Texas Instruments, 2011.
- [13] LTC2261-14 datasheet, Analog to Digital Converters, Linear Technology, 2014.
- [14] MAX11905 datasheet, Analog to Digital Converters, Maxim Integrated, 2019.
- [15] MCP33131D datasheet, Analog to Digital Converters, Microchip Technology Inc., 2018.
- [16] Digilent. Arty S7 Reference Manual [Internet]. Digilent. Available from: <https://digilent.com/reference/programmable-logic/arty-s7/reference-manual?redirect=1> [Accessed 14 February 2024]
- [17] DE0-Nano User Manual, Altera.
- [18] MachXO Family datasheet, Lattice Semiconductor Corporation, 2006.
- [19] IGLOO2 FPGA Evaluation Kit, Quickstart Card, Microsemi.
- [20] PJRC. Teensy 4.0 Development Board [Internet]. PRJC. Available from: <https://www.pjrc.com/store/teensy40.html> [Accessed 4 February 2024]
- [21] XR-2206 datasheet, Monolithic Function Generator, EXAR, 1997
- [22] RaspberryPi. Raspberry Pi 4 Model B Datasheet [Internet]. RaspberryPi. Available from: <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf> [Accessed 4 February 2024]
- [23] RaspberryPi. BCM2835 ARM Peripherals [Internet]. RaspberryPi. Available from: <https://www.raspberrypi.org/app/uploads/2012/02/BCM2835-ARM-Peripherals.pdf> [Accessed 5 February 2024]
- [24] Odroid-XU4 User Manual, HardKernel.
- [25] Beagleboard.org, BeagleBone Black [Internet]. Beagleboard.org. Available from: <https://www.beagleboard.org/boards/beaglebone-black> [Accessed 14 February 2024]
- [26] Arduino Docs. Arduino MKR Zero Product Reference Manual [Internet]. Arduino. Available from: <https://docs.arduino.cc/hardware/mkr-zero> [Accessed 14 February 2024]

- [27] Eduardo Barbieri (2024). Real-Time Linux vs RTOS – Part 2 [Internet]. Ubuntu. Available from: <https://ubuntu.com/blog/real-time-linux-vs-rtos-2> [Accessed 20 March 2024]
- [28] Jeremy P. Bentham (2020). Raspberry Pi DMA programming in C [Internet]. Lean 2, Embedded Systems without the bloat. Available from: <https://iosoft.blog/2020/05/25/raspberry-pi-dma-programming/> [Accessed 10 February 2024]
- [29] Jeremy P. Bentham (2020). Fast data capture with the Raspberry Pi [Internet]. Lean 2, Embedded Systems without the bloat. Available from: <https://iosoft.blog/2020/06/11/fast-data-capture-raspberry-pi/> [Accessed 10 February 2024]
- [30] Jeremy P. Bentham (2020). Raspberry Pi Secondary Memory Interface (SMI) [Internet]. Lean 2, Embedded Systems without the bloat. Available from: <https://iosoft.blog/category/secondary-memory-interface/> [Accessed 10 February 2024]
- [31] Jeremy P. Bentham (2020). Raspberry Pi Secondary Memory Interface (SMI) [Internet]. Lean 2, Embedded Systems without the bloat. Available from: <https://iosoft.blog/category/secondary-memory-interface/> [Accessed 10 February 2024]
- [32] fenlogic (2017). Project to talk to IDE interface using Raspberry-Pi GPIO pins in SMI [Internet]. Github. Available from: https://github.com/fenlogic/IDE_trial/tree/master [Accessed 10 February 2024]
- [33] BCM2711 ARM Peripherals documentation, Chapter 4, Raspberry Pi Ltd, 2022.
- [34] Paul J. Drongowski (2013). Memory hierarchy and access time [Internet]. Sand, software and sound, Electronics and computing for the fun of it. Available from: <http://sandsoftwaresound.net/raspberry-pi/raspberry-pi-gen-1/memory-hierarchy/> [Accessed 13 March 2024]
- [35] Macoy Madson (2023). Mailbox property interface [Internet]. Github. Available from: <https://github.com/raspberrypi/firmware/wiki/Mailbox-property-interface> [Accessed 13 March 2024]
- [36] Jake Sandler (Unknown). The Mailbox Peripheral [Internet]. Building an operating system for the Raspberry Pi. Available from: <https://jsandler18.github.io/extra/mailbox.html> [Accessed 13 March 2024]
- [37] Frigo, M., Johnson, S.G. (2020). FFTW . Massachusetts Institute of Technology.
- [38] Mike Perkins (Unknown). Power Measurements with the DFT [Internet]. CardinalPeak, Signal Processing. Available from: <https://www.cardinalpeak.com/blog/16-2> [Accessed 28 April 2024]
- [39] Brian Douglas (2024). Understanding Power Spectral Density and the Power Spectrum [Internet]. Matlab, Mathworks, Videos and Webinars. Available from: <https://www.mathworks.com/videos/understanding-power-spectral-density-and-the-power-spectrum-1707740400025.html> [Accessed 29 April 2024]

14. נספחים

14.1 נספח א' – SMI (Secondary Memory Interface)

מקור המידע העיקרי יהיה לנו על אופן הפעולה של ה-SMI ב-RPi ועל האוגרים המגדירים אותו הוא מסמך ה-[BCM2711 Peripherals](#) ותיעודים אינטראקטיביים נוספים. האוגרים של ה-SMI הם:

- .Control & Status – CS •
- .Transfer Length – L •
- .Address & Device Number – A •
- .Data FIFO – D •
- .DMA Control – DMC •
- .Device Read Settings – DSR •
- .Device Write Settings – DSW •
- .Direct Control & Status – DCS •
- .Direct Control Address & Device Number – DCA •
- .Direct Data Control – DCD •

בכל הגרסאות של ה-RPi מיקום האוגרים של ה-SMI נמצא בהזזה של 0x600000 מה-[PHYS_REG_BASE](#) (מיקום בחומרה). במקורה של ה-B-RPi 4 Model + 0xFE000000, זה נמצא ב- 0x60000000 ואנו נגדר את זה מכאן והילך כ-[SMI_BASE](#). בטבלה מוצגים המיקומים המדויקים של כל אוגר.

SMI_BASE + 0x##	Register	תפקיד
0x00	CS	Control & Status
0x04	L	Transfer Length
0x08	A	Address & Device Number
0x0c	D	Data FIFO
0x10	DSR0	Read Settings for Device 0
0x14	DSW0	Write Settings for Device 0
0x18	DSR1	Read Settings for Device 1
0x1c	DSW1	Write Settings for Device 1
0x20	DSR2	Read Settings for Device 2
0x24	DSW2	Write Settings for Device 2
0x28	DSR3	Read Settings for Device 3
0x2c	DSW3	Write Settings for Device 3

תפקיד	Register	SMI_BASE + 0x##
DMA Control	DMC	0x30
Direct Control & Status	DCS	0x34
Direct Control Address & Device Number	DCA	0x38
Direct Data Control	DCD	0x3c
FIFO Buffer Debug	FD	0x40

בעת אנו נציג בפירוט את כל השדות של כל אחד מהאגרים.

CS – Control & Status		
תפקיד	Register	SMI_BASE + 0x00+0x##
RX FIFO full	CS_RXFF_FULL	0x80000000
TX FIFO not empty	CS_TXFF_EMPTY	0x40000000
RX FIFO holds data	CS_RXFF_DATA	0x20000000
TX FIFO can accept data	CS_TXFF_SPACE	0x10000000
RX FIFO \geq 3/4 full	CS_RXFF_HIGH	0x08000000
TX FIFO $<$ 1/4 full	CS_TXFF_LOW	0x04000000
AXI FIFO underflow/overflow	CS_AFERR	0x02000000
External DREQ received	CS_EDREQ	0x00008000
Pixel mode on	CS_PXLDAT	0x00004000
Setup reg. written & used error	CS_SETERR	0x00002000
Pixel Valve mode on	CS_PVMODE	0x00001000
Receive interrupt	CS_RXIRQ	0x00000800
Transmit interrupt	CS_TXIRQ	0x00000400

CS – Control & Status		
SMI_BASE + 0x00+0x##	Register	תפקיד
0x00000200	CS_DONIRQ	Done interrupt
0x00000100	CS_TEEN	Tear enable
0x000000C0	CS_BYTPAD	Number of PAD byte
0x00000020	CS_WRITE	1 = Write to ext. devices 0 = Read from ext. devices
0x00000010	CS_FFCLR	Write to 1 to clear FIFO
0x00000008	CS_START	Write to 1 to start transfer(s)
0x00000004	CS_BUSY	Set if transfer is taking place
0x00000002	CS_DONE	Set if transfer has finished
0x00000001	CS_ENABLE	Set to enable SMI

A- Address & Device Number		
SMI_BASE + 0x08+0x##	Register	תפקיד
0x00000300	ADRS_DEV_MSK	Which device timing to use
0x00000000	ADRS_DEV0	Use read0/write0 timing
0x00000100	ADRS_DEV1	Use read1/write1 timing
0x00000200	ADRS_DEV2	Use read2/write2 timing
0x00000300	ADRS_DEV3	Use read3/write3 timing
0x0000003F	ADRS_MSK	Address bits 0-5 to use

DMC – DMA Control		
תפקיד	Register	SMI_BASE + 0x30+0x##
DMA enable	DMAC_ENABLE	0x10000000
D16/17 are resp DMA_REQ/DMA_ACK	DMAC_DMAP	0x01000000
Read Panic threshold	DMAC_RXPANIC_MSK	0x00FC0000
Write Panic threshold	DMAC_TXPANIC_MSK	0x0003F000
Read DMA threshold	DMAC_RXTHRES_MSK	0x00000FC0
Write DMA threshold	DMAC_TXTHRES_MSK	0x0000003F

DSR/DSW – Device Read/Write Settings		
תפקיד	Register	SMI_BASE + 0x10-0x2c+0x##
Data width mask	RW_WIDTH_MSK	0xC0000000
Data width 8 bits	RW_WID8	0x00000000
Data width 16 bits	RW_WID16	0x40000000
Data width 18 bits	RW_WID18	0x80000000
Data width 9 bits	RW_WID9	0xC0000000
Setup cycles (6 bits)	RW_SETUP_MSK	0x3F000000
Run cycle motorola mode	RW_MODE68	0x00800000
Run cycle intel mode	RW_MODE80	0x00000000
READ! Setup only for first cycle	READ_FSETUP	0x00400000
Write! swap pixel data	WRITE_SWAP	0x00400000

DSR/DSW – Device Read/Write Settings		
SMI_BASE + 0x10- 0x2c+0x##	Register	תפקיד
0x003F0000	RW_HOLD_MSK	Hold cycles (6 bits)
0x00008000	RW_PACEALL	Apply pacing always
0x00007FO0	RW_PACE_MSK	Pace cycles (7 bits)
0x00000080	RW_DREQ	Use DMA req on read/write
0x0000007F	RW_STROBE_MSK	Strobe cycles (7 bits)

DCS – Direct Control & Status		
SMI_BASE + 0x34+0x##	Register	תפקיד
0x00000008	DIRCS_WRITE	Write to ext. devices =1
0x00000004	DIRCS_DONE	Set if transfer has finished
0x00000002	DIRCS_START	Write to 1 to start transfer(s)
0x00000001	DIRCS_ENABLE	Set to enable SMI 0 = Read from ext. devices

DCA – Direct Control Address & Device Number		
SMI_BASE + 0x38+0x##	Register	תפקיד
0x00000300	DIRADRS_DEV_MSK	Which device timing to use
0x00000000	DIRADRS_DEV0	Use read0/write0 timing
0x00000100	DIRADRS_DEV1	Use read1/write1 timing
0x00000200	DIRADRS_DEV2	Use read2/write2 timing

DCA – Direct Control Address & Device Number		
SMI_BASE + 0x38+0x##	Register	תפקיד
0x000000300	DIRADRS_DEV3	Use read3/write3 timing
0x0000003F	DIRADRS_MSK	Address bits 0-5 to use

כפי שהסבירנו בפרק 9.1.3, ל-SMI יש אפשרות לשלוט על קו ה-O/I של RPi באופן בלתי תלוי באמצעות טיימרים מסוימים ושמור און הדגימות ב-buffer שלו ולהעבירן ישירות למרחיב המשתמש. שיטה זו נקראת Direct Mode והאוגרים שצורך להגדיר עבור אונן בעודה זהה הם DCD, DCS ו- DCA. בפרויקט שלנו אנו לא משתמשים בשיטה זו בגלל תדר דגימה גבוה שגורר בעיות תזמון עם המעבד. לעומת זאת אנו משתמשים בבקשה ישירה ל-DMA שיבצע העברת batch דגימות כל פעם כאשר ה-FIFO מ מלא.

אנו מתחילה את ה-SMI ע"י הגדרת כל האוגרים הרלוונטיים, וחשוב לשים לב כי יש לסיים להגדיר את כל השודות טרם הגדרת ה-'1'='CS_ENABLE. כדי להתחיל הרצה של הדגימות מספיק לנו נגידר '1'='CS_START, ואז ה-SMI Controller יבצע את מספר ההפניות בהתאם לעורק באוגר L ובהתאם לפרמטרי הטיימר שהצבנו ב-DSR (לקריאיה) או DSW (לכתיבה). אם ברגע מסוים אנו נרצה לעצור את ה-SMI מספיק להגדיר '0'='CS_ENABLE. מומלץ גם לבצע '1'='CS_FFCLR לאחר הגדרת CS_ENABLE, כדי לגרוס כל תוכן סרק שיכל להיות לפני כן ב-FIFO.

14.2 נספח ב' – DMA (Direct Memory Access)

בפרק 9.1.3 אנו הצגנו את הרעיון הכללי של DMA, ללא העמeka באופן הגדרתו ובאגרים שמנגדירים אותו. בעת, אנו נציג שלב שלב מי המ אוגרים הרלוונטיים, מה השודות ומשמעותם, וכיצד אנו השתמשנו בו מעשית בפרויקט. מקור המידע העיקרי שהוא לנו על אונן הפעולה של DMA ב-RPi וועל האוגרים המגדירים אותו הוא מסמר ה-BCM2711 Peripherals ותיעודים אינטראקטיביים נוספים. ל-RPi יש יתרון גדול בכך שהוא מציע זמינות של עד 16 ערוצים בלא תילויים של DMA (כמפורט חלקים שמורים למערכת אך לא ידוע בהכרח איזה מאחר וזה משתנה בין עדכוני מערכת). בכל הגרסאות של RPi מיקום האוגרים של DMA נמצא בהזזה של 0x007000 מה-PHYS_REG_BASE (מיקום של DMA Channel 0 Model B-RPi 4). הערוצים השונים של DMA נמצאים לפני המיקומים בטבלה.

DMA_BASE + 0x###	DMA Channel
0x000	DMA Channel 0 Register Set
0x100	DMA Channel 2 Register Set
0x200	DMA Channel 3 Register Set
0x300	DMA Channel 4 Register Set
...	...

DMA_BASE + 0x###	DMA Channel
0xE00	DMA Channel 14 Register Set

באופן מפתח רק ערוץ 16 אינו שייר לרצף זה, והוא נמצא ב-0xE000000 + 0x05000. בפרויקט אנו השתמשנו בערך הראשון בלבד. כאשר אנו רצים להגדיר פעולה כלשהי עבור ה-DMA, אנו צריכים ליצור Control Block של Data Structure של CB (Control Block), ולטען אותו לערוץ הרצוי. כל CB בניו מ-8 מילימ (256 סיביות) כאשר כל אוגר הוא מילה (32 סיביות) והם:

- .Transfer Information – TI (Register 0) •
- .Source Address – SOURCE_AD (Register 1) •
- .Destination Address – DEST_AD (Register 2) •
- .Transfer Length – TXFR_LEN (Register 3) •
- .2D Mode Stride – STRIDE (Register 4) •
- .Next Control Block Address – NEXTCONBK (Register 5) •
- .Reserved, set to zero – N/A (Register 6-7) •

כאשר ייצרנו Data Structure כזה, אנו ממלאים את השדות הרלוונטיים שמתוארים בטבלאות הבאות.

TI – Transfer Information			
Bits	Name	Description	Type
31:27	Reserved	-	-
26	NO_WIDE_BURSTS	Don't do wide writes as a 2 beat burst	RW
25:21	WAITS	Add Wait Cycles	RW
20:16	PERMAP	Peripheral Mapping	RW
15:12	BURST_LENGTH	Burst Transfer Length	RW
11	SRC_IGNORE	Ignore Reads	RW
10	SRC_DREQ	Control Source Reads with DREQ	RW
9	SRC_WIDTH	Source Transfer Width	RW
8	SRC_INC	Source Address Increment	RW
7	DEST_IGNORE	Ignore Writes	RW

TI – Transfer Information			
Bits	Name	Description	Type
6	DEST_DREQ	Control Destination Writes with DREQ	RW
5	DEST_WIDTH	Destination Transfer Width	RW
4	DEST_INC	Destination Address Increment	RW
3	WAIT_RESP	Wait for a Write Response	RW
2	Reserved	-	-
1	TDMODE	2D Mode	RW
0	INTEN	Interrupt Enable	RW

SOURCE_AD – Source Address			
Bits	Name	Description	Type
31:0	S_ADDR	DMA Source Address	RW

DEST_AD – Destination Address			
Bits	Name	Description	Type
31:0	D_ADDR	DMA Destination Address	RW

TXFR_LEN – Transfer Length			
Bits	Name	Description	Type
31:30	Reserved	-	-

TXFR_LEN – Transfer Length			
Bits	Name	Description	Type
29:16	YLENGTH	When in 2D mode, This is the Y transfer length, indicating how many xlength transfers are performed. When in Linear mode this becomes the top bits of the XLENGTH	RW
15:0	XLENGTH	Transfer Length in bytes	RW

STRIDE – 2D Stride			
Bits	Name	Description	Type
31:16	D_STRIDE	Destination Stride (2D Mode)	RW
15:0	S_STRIDE	Source Stride (2D Mode)	RW

NEXTCONBK – Next Control Block Address			
Bits	Name	Description	Type
31:0	ADDR	Address of next CB for chained DMA operations	RW

בעת אנו נציג את האוגרים של הערוץ הראשון (ערוץ הראשן וכן 0 (DMA Channel 0 Control & Status

DMA_BASE + 0x000 + 0x###	Name	Description
0x000	CS	DMA Channel 0 Control & Status
0x004	CONBLK_AD	DMA Channel 0 Control Block Address

DMA_BASE + 0x000 + 0x###	Name	Description
0x008	TI	DMA Channel 0 CB Register 0 (Transfer Information)
0x00c	SOURCE_AD	DMA Channel 0 CB Register 1 (Source Address)
0x010	DEST_AD	DMA Channel 0 CB Register 2 (Destination Address)
0x014	TXFR_LEN	DMA Channel 0 CB Register 3 (Transfer Length)
0x018	STRIDE	DMA Channel 0 CB Register 4 (2D Stride)
0x01c	NEXTCONBK	DMA Channel 0 CB Register 5 (Next CB Address)
0x020	DEBUG	DMA Channel 0 Debug

להלן השדות הנוטרים של האוגרים בערוץ.

CS – Control & Status			
Bits	Name	Description	Type
31	RESET	DMA Channel Reset	W1SC
30	ABORT	Abort DMA	W1SC
29	DISDEBUG	Disable debug pause signal	RW
28	WAIT_FOR_OUTSTANDING_WRITES	Wait for outstanding writes	RW
27:24	Reserved	-	-
23:20	PANIC_PRIORITY	AXI Panic Priority Level	RW
19:16	PRIORITY	AXI Priority Level	RW
15:9	Reserved	-	-
8	ERROR	DMA Error	RO

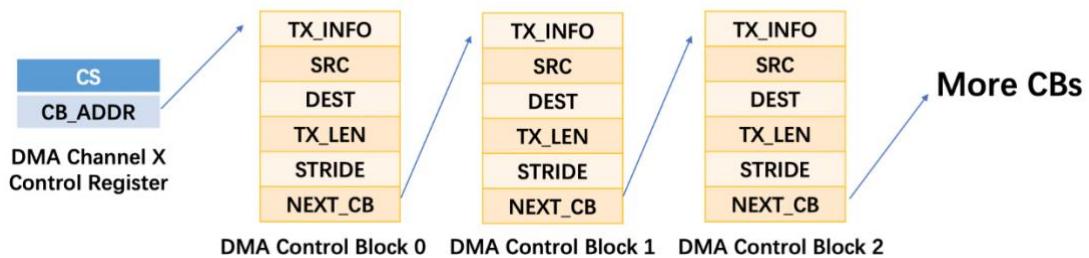
CS – Control & Status			
Bits	Name	Description	Type
7	Reserved	-	-
6	WAITING_FOR_OUTSTANDING_WRITES	Waiting for outstanding writes	RO
5	DREQ_STOPS_DMA	DMA Paused by DREQ State	RO
4	PAUSED	DMA Paused State	RO
3	DREQ	DREQ State	RO
2	INT	Interrupt Status	W1C
1	END	DMA End Flag	W1C
0	ACTIVE	Activate the DMA	RW

DEBUG – Debug register			
Bits	Name	Description	Type
31-29	Reserved	-	-
28	LITE	DMA Lite	RO
27:25	VERSION	DMA Version	RO
24:16	DMA_STATE	DMA State Machine State	RO
15:8	DMA_ID	DMA ID	RO
7:4	OUTSTANDING_WRITES	DMA Outstanding Writes Counter	RO
3	Reserved	-	-
2	READ_ERROR	Slave Read Response Error	W1C

DEBUG – Debug register			
Bits	Name	Description	Type
1	FIFO_ERROR	FIFO Error	W1C
0	READ_LAST_NOT_SET_ERROR	Read Last Not Set Error	W1C

באשר אנו רוצים להוביל את ה-CB לעורץ, אנו צריכים לטעון את הכתובת של ה-CB Structure לאוגר CONBLK_AD, ולאחר מכן להפעיל את סיבית ACTIVE ב-CS. זה אוטומטיות יטען CB מהזיכרון וימלא את השדות הרלוונטיים בעורץ. חשוב לציין כי רק שלושה מתחום סה"כ האוגרים הנמצאים בעורץ ניתנים לביצוע ישירה בזמן ריצה, והם CS, DEBUG CONBLK_AD ו-DECODE CONBLK_AD. למטרות שאנו לא מסוגלים לעורץ בזמן ריצה את שאר האוגרים בעורץ, אנו יכולים לשנות דינמיות את פעולה ה-DMA על ידי טעינה חדשה של CB אחר.

באשר אנו מפעילים את סיבית ACTIVE ה-DMA טוען את ה-CB ומתחיל לפעול לפי הוראותיו. כאשר CONBLK_AD השלים את ה-Transfer (במילויים אחרים, length = 0), ה-DMA יעדכן את ה-CB ייחודה עם הכתובת שרשומה ב-NEXTCONBK, יטען את האוגרים בעורץ בתוכן של ה-CB הבא בתוכו, ויחזור שוב על התהילן. DMA יעצור (ויאפס את סיבית ACTIVE) כאשר הוא סיים לבצע את ה-Transfer של ה-CB הנוכחי וה-NEXTCONBK מצביע לכתובת 0x0000_0000. בתובות זו נתענת ל-DMA ונעצר. באירור ניתן לדאות המכחשה של שרשרת פעולות על ידי שרשור של CB's.



בפרויקט שלנו אנו ניצלנו את העובדה שיש אפשרות של תקשורת ישירה בין ה-DMA ל-SMI, כאשר ה-SMI (בתוכו פריפריה) יכול לבקש DMA DREQ מה-DMA DMA. ביצוע העברת דגימות ממוקם אחד בזיכרון לזכרון ה-uncached. אם נחזיר לאלגוריתם שמתואר בפרק 9.1.5, אנו נראה שה-DMA ממתין לבצע את ההעברה של batch הדגימות עד לרגע בו הוא קיבל DREQ ישירות מה-SMI. ברגע שההעברה הסתיימה, אנו מקבלים INTEN ב奧גר INTINTEN ואוגר TI ב-TI של CB'ו. ההתרעה זו ניתנת לקריאה בזמן אמת על ידי המעבד במוחב המשמש והוא מפעילה סריקה על הדגימות, בחיפוש אחר סימנים של פולס כלשהו. במידה ופולס לא זהה, ה-DMA ממשך לפעול בollowה בין שני CB'ים שמוצעים פינגן-פונג על מיקום הדגימות בזיכרון. במידה זהה פולס, אז אנו יוצרים CB חדש ובו NEXTCONBK = 0x0000_0000, טוענים אותו דינמיות NEXTCONBK, מתבצעת החזרה של I/O של ה-RPi למצב גigel, וכן DMA נעצה.

14.3 נספח ג' – FFTW (C Library)

FFTW – Fastest Fourier Transform in the West היא ספרייה הקיימת בשפה C המשמשת לחישוב התמרת פורייה מהירה (FFT – Fast Fourier Transform) אשר פותחה ע"י מתאו פריגו וסטיבן ג'יונסון ב-MIT. הספרייה מספקת ממושך תכנית יעיל ונוח לביצוע פעולות חישוביות על נתונים בתדרים גבוהים. FFTW יודעה בgmt שמה וביבולתה לתמוך במערכות נתונים בכל גודל, תוך אפשרות לחשב התמורות רב ממדיות.

שימוש בספרייה

לאחר התקנה פשוטה, באמצעות include של <fftw3.h>, כל פונקציות FFT נגישות לשימוש. בעזרת קוד בשפת C ניתן להפעיל פונקציות לצירוף תוכניות FFT, הזרת נתונים וביצוע חישובים. FFTW תומכת בשימוש במצביים ומאפשרת חישובים על מערכים דינמיים של נתונים.

בפרויקט שלנו השתמשנו בספרייה לעיבוד נתונים בזמן אמת והרתום בתחום התדר. הנתונים מודדים ממשק ה-IMU של ה-RPi דרך חייזרים שמבצעים דיגיטציה של אותות אנלוגיים. התוכנית מקבלת את הנתונים בצורה של דגימות בזמן, מעבדת אותם לקבלת מידע על התדר ועוצמת האותות.

דוגמת קוד

- קראיה בספרייה בתחילת הקוד:

```
#include <fftw3.h>
```

- הקצת זיכרון – השתמשנו בפונקציה "fftw_malloc" להקצת זיכרון בצורה בטוחה עבור נתונים הזמן (הקלט) והוצאות (הפלט):

```
// Prepare for FFT
```

```
in = (double*) fftw_malloc(sizeof(double) * FFT_NSAMPLES);
out = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * (FFT_NSAMPLES/2 + 1));
p = fftw_plan_dft_r2c_1d(FFT_NSAMPLES, in, out, FFTW_ESTIMATE);
```

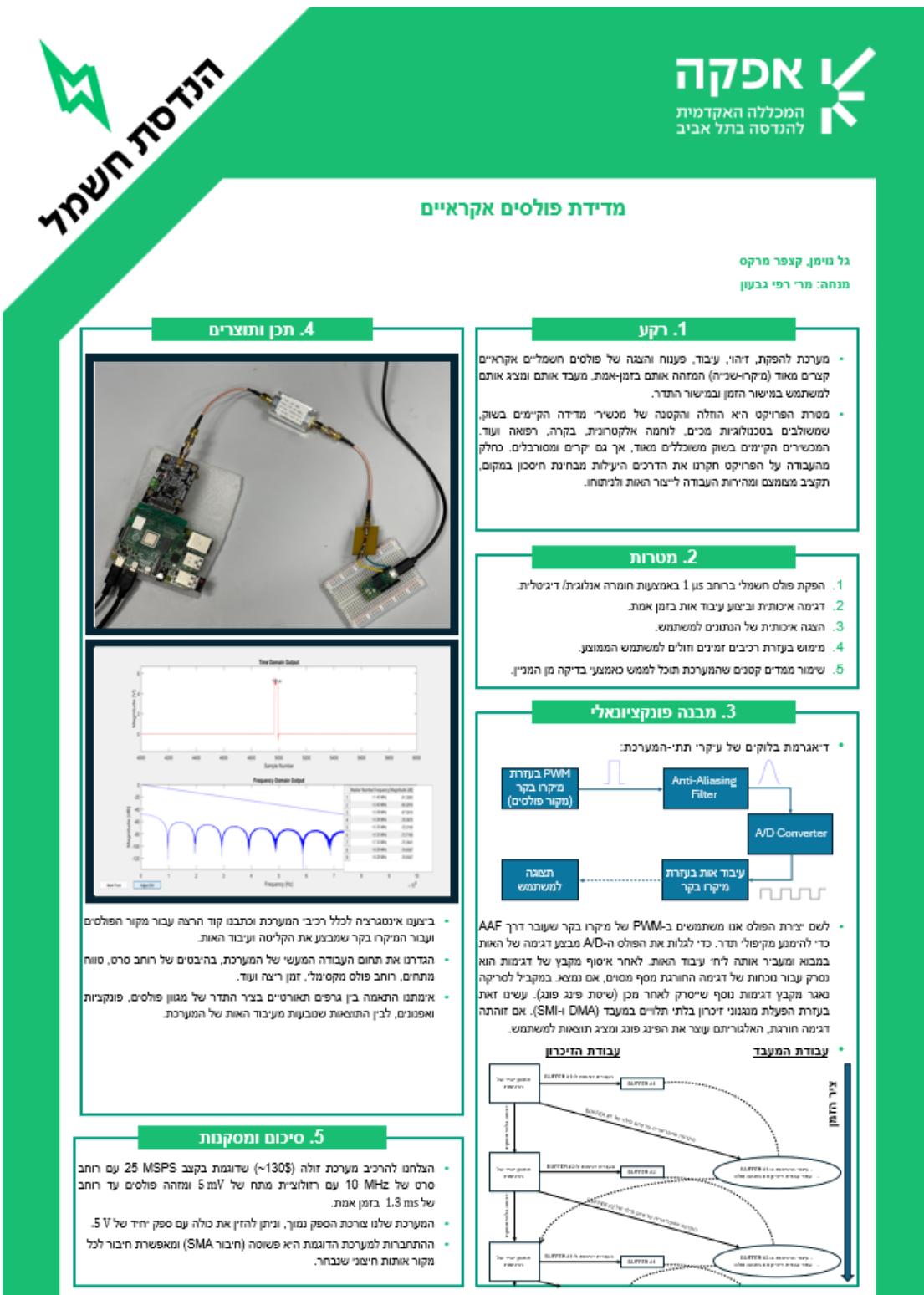
- יצירת תוכנית FFT – השתמשנו בפונקציה "fftw_plan_dft_1d" ליצירת התוכנית:
fftw_plan p;
p = fftw_plan_dft_r2c_1d(FFT_NSAMPLES, in, out, FFTW_ESTIMATE);
הפונקציה מקבלת את מספר הדגימות, מערכי הקלט והפלט ודגל שמצוין את סוג התכנית (במקרה שלנו זה תכנון מהיר ללא שינוי נתונים).

- הזרת הנתונים וביצוע התמורה:

```
// Populate input array and perform FFT
for (i = 0; i < data->nSamples; i++) {
    in[i] = val_volts(time_data[i]);
}
fftw_execute(p);
```

- ניתוח והציגת התוצאות – לאחר ההתמורה, המערך "out" מכיל את התוצאות במרחב הזמן:

```
// Plotting FFT data
gp2 = popen("gnuplot -persistent", "w");
fprintf(gp2, "set title 'FFT of ADC Voltage'\n");
fprintf(gp2, "set xlabel 'Frequency (MHz)'\n");
fprintf(gp2, "set ylabel 'Magnitude (dB)'\n");
fprintf(gp2, "set yrange [-150:0]\n");
fprintf(gp2, "set xrange [0:%.1f]\n", (Fs / 2) / 1e6);
fprintf(gp2, "set grid\n");
fprintf(gp2, "plot '-' using 1:2 with lines notitle\n");
for (i = 0; i <= FFT_NSAMPLES / 2; i++) {
    fprintf(gp2, "%f %f\n", data->frequency[i] / 1e6, data->magnitude[i]);
}
fprintf(gp2, "e\n");
pclose(gp2);
```



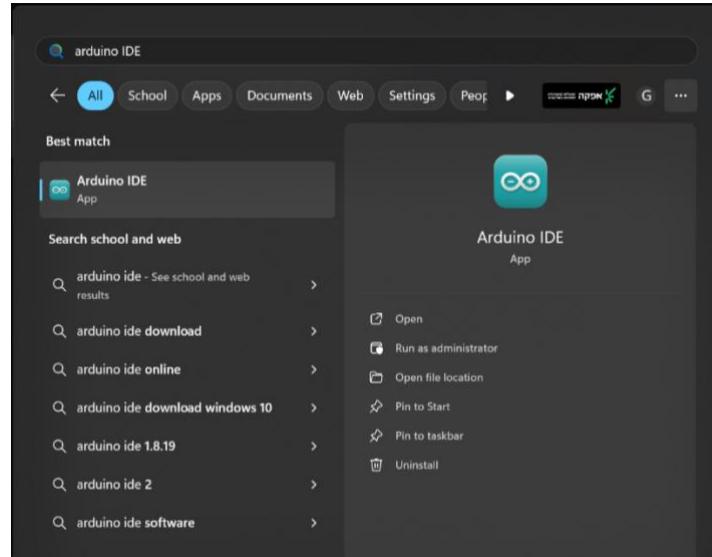
14.5 נספח ה' – הוראות לשימוש במערכת

להלן, הנקודות המפורטות כיצד יש להשתמש במערכת.

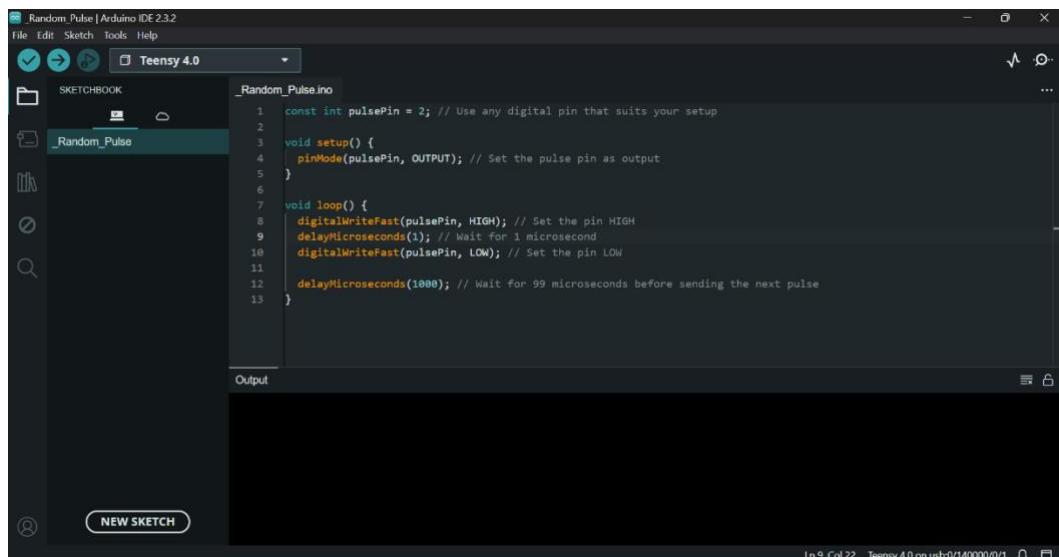
הפקת האות החשמלי:

להפקת הפולס בעזרת הרכיב Teensy נשתמש בתוכנה "Arduino IDE", להלן ההוראות כיצד להשתמש בתוכנה:

1. נפתח את התוכנה "Arduino IDE":



2. מzn את הקוד הבא ב-SketchBook חדש:



3. נבחר מבין המכשירים הזמינים את רכיב ה- "Teensy" שלנו:



4. בעת נלחץ על "Verify" ונכחה שתפתח לנו חלונית השליטה ברכיב:

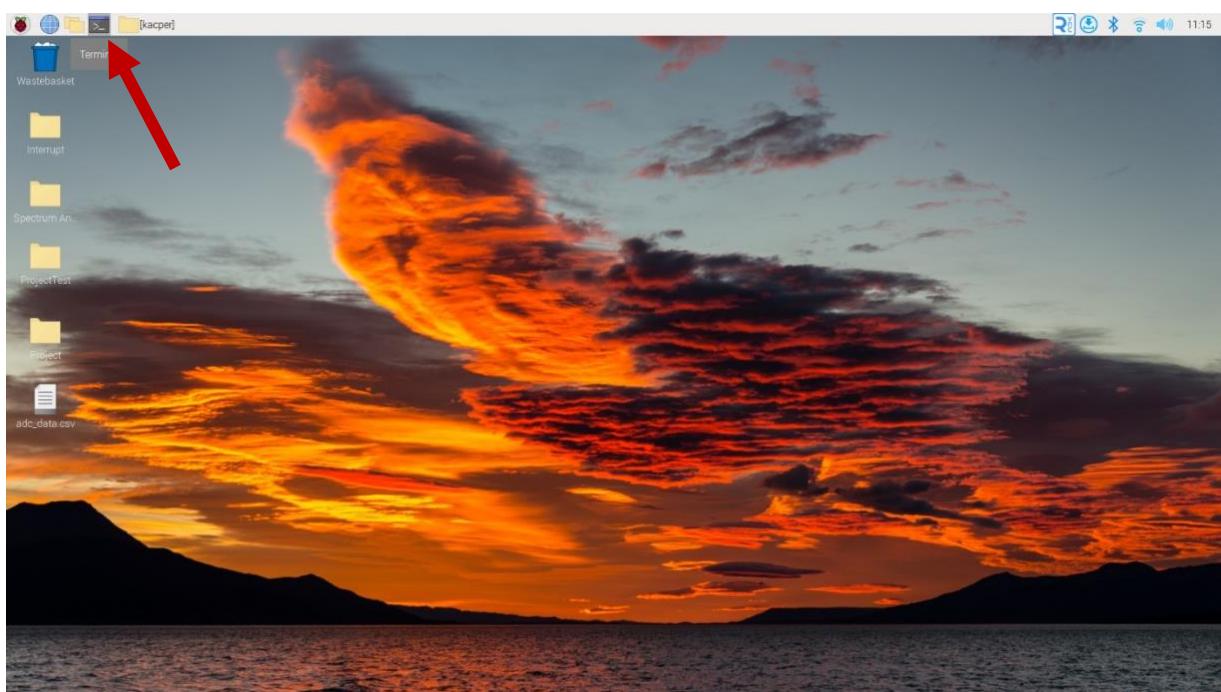


בעת התוכנה מזינה פולס ברווח של 1 מיקרו שנייה בזמנים אקראיים.

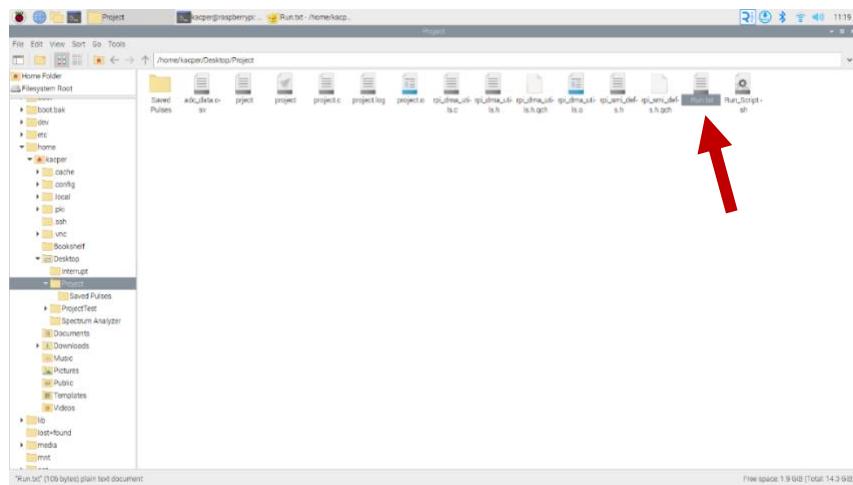
המשך השימוש במערכת:

לאחר שהדלקנו את המערכת והפכנו פולסים אקראיים כפי שנאמר, בעת נוכל לבצע כמה פעולות פשוטות אשר יתחלו לטרוק אחר פולסים, ברגע שייזכר פולס, המערכת תציג תרגום אותו, תבצע לו התרמה פוריה ותציג לנו את התוצאות.

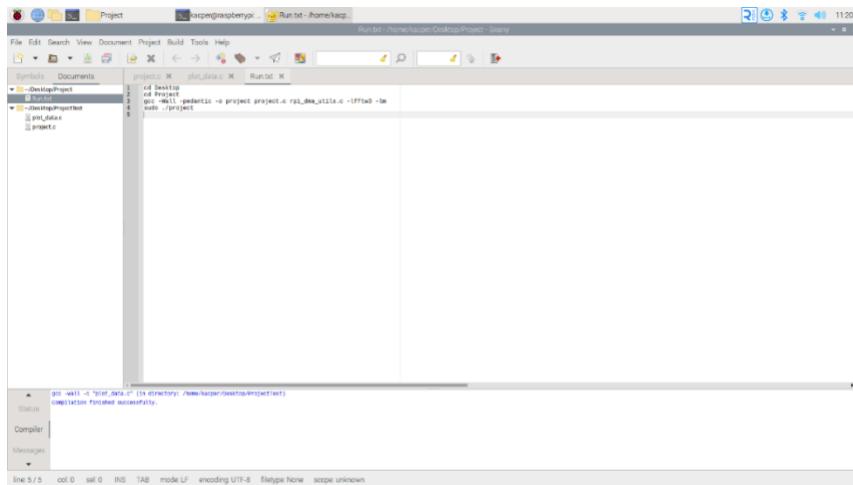
1.פתיחה הטרמינל בשולחן העבודה של ה-RPi:



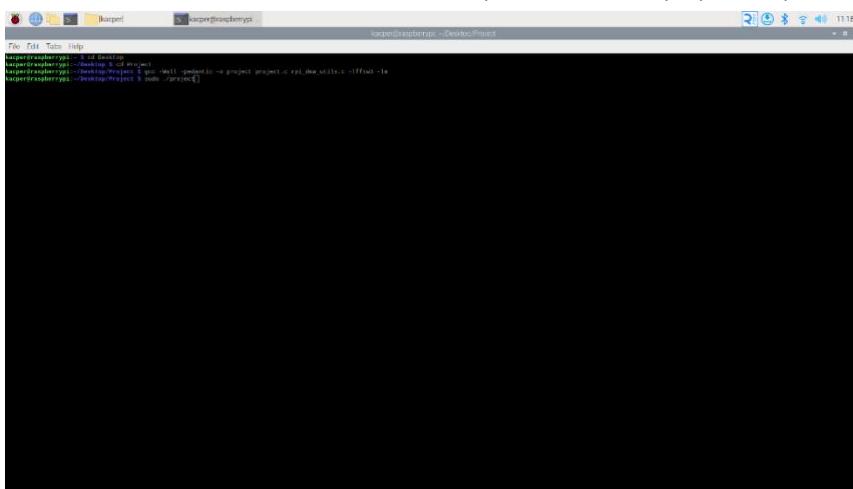
2. נפתח את הקובץ אשר מכיל את פקודות ההרצה של הפרויקט:



3. יפתח לנו הקובץ הבא:



4. את התוכן של הקובץ Run.txt נעתיק ישירות לטרמינל אשר פתוחה:



ובעת הקוד ירוץ. במידה וזהה פולס, יפתחו שתי חלונות ובכל אחד מהם יוצג האות במורחב הדגימה ובשני האות במורחב התדר.

14.6 נספח ו' – קוד שמומש עבור ה-Teensy 4.0 גרסה א' – פולסים משודרים באינטראול קבוע

```
const int pulsePin = 2; // Define the output pin

void setup() {
    pinMode(pulsePin, OUTPUT); // Set the pulse pin as output
}

void loop() {
    digitalWriteFast(pulsePin, HIGH); // Set the pin HIGH
    delayNanoseconds(1000); // Wait for 1 microsecond
    digitalWriteFast(pulsePin, LOW); // Set the pin LOW

    delayMicroseconds(1500); // Wait for 1.5 miliseconds (2 batch sizes) before sending the next pulse
}

גראסה ב' – פולס בודד משודר בעת לחיצת משתמש

const int pulsePin = 2; // Define the output pin

void setup() {
    pinMode(pulsePin, OUTPUT); // Set the pulse pin as output
    Serial.begin(9600); // Initialize serial communication at 9600 baud
    while (!Serial) {
        // Wait for the serial connection to be established
    }
    Serial.println("Press any key to send the pulse.");
}

void loop() {
    if (Serial.available() > 0) {
        char c = Serial.read(); // Read the input character
        sendPulse(); // Call the function to send the pulse
        Serial.println("Pulse sent. Press any key to send another pulse.");
    }
}

void sendPulse() {
    digitalWriteFast(pulsePin, HIGH); // Set the pin HIGH
    delayNanoseconds(1000); // Wait for 1 microsecond
    digitalWriteFast(pulsePin, LOW); // Set the pin LOW
```

```
delayMicroseconds(1500); // Wait for 1.5 miliseconds before the next possible pulse  
}
```

נספח ז' – קוד שימוש עבור ה-RPi 14.7

~/Desktop/Random_Pulse_Detection.c

```
1 // Code for Random Pulse Capturing and Analyzing, using AD9226 ADC in parallel
2 // connection with Raspberry Pi.
3 // EE Afeka Final Project 2024
4 // Kacper Marks & Gal Neumann
5
6 #include <stdint.h>
7 #include <stdio.h>
8 #include <signal.h>
9 #include <stdint.h>
10 #include <stdlib.h>
11 #include <string.h>
12 #include <unistd.h>
13 #include <time.h>
14 #include "rpi_dma_utils.h"
15 #include "rpi_smi_defs.h"
16 #include <fftw3.h>
17 #include <math.h>
18
19 // Number of samples in a batch, and samples for ADC transient
20 #define NSAMPLES      16384
21 #define PRE_SAMP       50
22 #define NCYCLES        50000 //Approx. 32 seconds pulse searching
23 #define FFT_NSAMPLES   (2*NSAMPLES)
24
25 // SMI cycle timings
26 #define SMI_NUM_BITS   SMI_16_BITS
27 #define SMI_TIMING     SMI_TIMING_25M
28
29 // Timing of SMI, width, setup, strobe & hold
30 #define SMI_TIMING_25M 4, 3, 8, 4 // 25 MS/s
31
32 // No. of bytes for ADC sample
33 #define SAMPLE_SIZE    2
34
35 // Voltage calibration & pulse width
36 #define ADC_ZERO        2034
37 #define ADC_SCALE       406.5
38 #define VOL_THRESH      1.0
39 #define PULSE_NSAMPS    5
40
41 // GPIO's pin no.
42 #define ADC_D0_PIN      12
43 #define ADC_NPINS       12
44 #define SMI_SOE_PIN     6
45 #define SMI_SWE_PIN     7
46 #define SMI_DREQ_PIN    24
47 #define TEST_PIN        25
48
```

```

49 // DMA request threshold
50 #define REQUEST_THRESH 4
51
52 // Set zero for single value, non-zero for block read
53 #define USE_DMA 1
54 // Use test pin in place of GPIO mode setting (to check timing)
55 #define USE_TEST_PIN 0
56
57 // SMI register names for diagnostic print
58 char *smi_regs[] = {
59     "CS", "LEN", "A", "D", "DSR0", "DSW0", "DSR1", "DSW1",
60     "DSR2", "DSW2", "DSR3", "DSW3", "DMC", "DCS", "DCA", "DCD", ""
61 };
62
63 // SMI CS register field names for diagnostic print
64 #define STRS(x) STRS_(x) ","
65 #define STRS_(...) #__VA_ARGS__
66 char *smi_cs_regs[] = STRS(SMI_CS_FIELDS);
67
68 // Structures for mapped I/O devices, and non-volatile memory
69 extern MEM_MAP gpio_regs, dma_regs, clk_regs;
70 MEM_MAP vc_mem, smi_regs;
71
72 // Pointers to SMI registers
73 volatile SMI_CS_REG *smi_cs;
74 volatile SMI_L_REG *smi_l;
75 volatile SMI_A_REG *smi_a;
76 volatile SMI_D_REG *smi_d;
77 volatile SMI_DMC_REG *smi_dmc;
78 volatile SMI_DSR_REG *smi_dsr;
79 volatile SMI_DSW_REG *smi_dsw;
80 volatile SMI_DCS_REG *smi_dcs;
81 volatile SMI_DCA_REG *smi_dca;
82 volatile SMI_DCD_REG *smi_dcd;
83
84 // Buffer for captured samples
85 uint16_t sample_data[2*NSAMPLES];
86
87 // Non-volatile memory size
88 #define VC_MEM_SIZE(nsamp) (PAGE_SIZE + ((nsamp)+4)*SAMPLE_SIZE)
89
90 void map_devices(void);
91 void fail(char *s);
92 void terminate(int sig);
93 void smi_start(int nsamples, int packed);
94 uint32_t *DMA_Start(MEM_MAP *mp, int nsamp, uint32_t *data, DMA_CB *cbs, uint32_t *modes);
95 int Convert_Samples(void *buff, uint16_t *data, int nsamp);
96 void init_smi(int width, int ns, int setup, int hold, int strobe);
97 void disp_smi(void);
98 void mode_word(uint32_t *wp, int n, uint32_t mode);

```

```

99 float val_volts(int val);
100 int GPIO_val(void);
101 void disp_reg_fields(char *regstrs, char *name, uint32_t val);
102 void dma_wait(int chan);
103 int detect_trigger(void *buff, int nsamps, float threshold);
104 void last_block(MEM_MAP *mp, DMA_CB *cbs, uint32_t *modes, volatile uint32_t *p);
105 void plot_data(int order);
106 void save_to_csv(uint16_t *data, int nsamp);
107
108
109
110 int main(int argc, char *argv[])
111 {
112     void *buffer; //Pointer to the uncached buffers
113     int i, flag = 0, order = 0;
114
115     signal(SIGINT, terminate);
116     map_devices(); //Memory mapping of DMA, SMI and GPIO's
117     for (i=0; i<ADC_NPINS; i++) //Defining I/O pins
118         gpio_mode(ADC_D0_PIN+i, GPIO_IN);
119     gpio_mode(SMI_SOE_PIN, GPIO_ALT1);
120     init_smi(SMI_NUM_BITS, SMI_TIMING); //Setting up SMI (No activation yet)
121     map_uncached_mem(&vc_mem, VC_MEM_SIZE(2*NSAMPLES+PRE_SAMP)); //Allocating uncached memory page for our user-space DMA, SMI and GPIO's registers
122     DMA_CB *cbs= (&vc_mem)->virt; //Creating DMA CB data structure
123     uint32_t *data=(uint32_t *) (cbs+5), *modes=data+0x10;
124     smi_dmc->dmaen = 1;
125     smi_cs->enable = 1; //SMI enabled (No activation yet)
126     smi_cs->clear = 1;
127     buffer = DMA_Start(&vc_mem, NSAMPLES, data, cbs, modes); //Setting up the DMA channel and CB's
128     volatile uint32_t *p=REG32(dma_regs, DMA_REG(DMA_CHAN_A, DMA_CS)); //Real-Time, dynamic DMA Control & Status Register
129
130     smi_start(NCYCLES*NSAMPLES+PRE_SAMP, 1); //SMI activated, starts sampling and requesting DMA CB's operations
131     while (dma_active(DMA_CHAN_A) && flag != 1) //Pulse Capturing Loop
132     {
133         if (*p & (1 << 2)) { //Check interrupt (either Buffer #1 or Buffer #2 is filled)
134             *REG32(dma_regs, DMA_REG(DMA_CHAN_A, DMA_CS)) = 5; // Clear 'INT' flag
135             flag = detect_trigger(buffer,NSAMPLES,VOL_THRESH); //Check for Pulse
136             if(flag){ //If pulse was detected
137                 printf("Success! %d\n",flag);
138                 last_block(&vc_mem, cbs, modes, p); //Exit the ping-pong loop by chaining new CB
139                 if(MEM_BUS_ADDR((&vc_mem), &cbs[3]) == *REG32(dma_regs, DMA_REG(DMA_CHAN_A, DMA_CONBLK_AD))){
140                     order = 1; //Check the order between Buffer #1 and Buffer #2

```

```

141             printf("Order = %d\n",order);
142         }
143     }
144 }
145 }
146 usleep(1000); // 1ms Halt for DMA disabaling
147 Convert_Samples(buffer, sample_data, 2*NSAMPLES); //Converting 2 byte
samples to voltage levels
148 smi_cs->enable = smi_dcs->enable = 0; //Closing SMI
149 save_to_csv(sample_data, 2*NSAMPLES); // Save data to CSV
150 plot_data(order); //Graph plotting of Time & Freq domains
151 terminate(0);
152 return(0);
153 }
154
155 void plot_data(int order)
156 {
157     uint16_t data[2*NSAMPLES];
158     FILE *gp1, *gp2; // GNUploat pipes for two separate plots
159     int i;
160     double *in;
161     fftw_complex *out; //FFTW register
162     fftw_plan p; //plan setting up
163     double Fs = 25e6, freq, magnitude, bin_val; // Sampling frequency in Hz
164     if(order){ //Re-ordering Buffer #1 and Buffer
#2 samples accordingly
165         for (i = 0; i < 2*NSAMPLES; i++){
166             if (i < NSAMPLES){
167                 data[i] = sample_data[NSAMPLES+i];
168             }
169             else{
170                 data[i] = sample_data[i-NSAMPLES];
171             }
172         }
173     }
174     else{
175         for (int i = 0; i < 2*NSAMPLES; i++) {
176             data[i] = sample_data[i];
177         }
178     }
179     in = (double*) fftw_malloc(sizeof(double) * FFT_NSAMPLES);
180     out = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * (FFT_NSAMPLES/2 +
1));
181     p = fftw_plan_dft_r2c_1d(FFT_NSAMPLES, in, out, FFTW_ESTIMATE); //FFT execu-
tion
182     // Open two separate GNUploat sessions
183     gp1 = popen("gnuplot -persistent", "w");
184     gp2 = popen("gnuplot -persistent", "w");
185     if (gp1 == NULL || gp2 == NULL) {
186         fprintf(stderr, "Could not open pipe to GNUploat.\n");
187     }
188     for (i = 0; i < FFT_NSAMPLES; i++) {

```

```

189     in[i] = Volts(data[i]);
190 }
191 fftw_execute(p);
192
193 // Plot Time-domain data in the first window
194 fprintf(gp1, "set title 'ADC Voltage Plot'\n");
195 fprintf(gp1, "set xlabel 'Sample Number'\n");
196 fprintf(gp1, "set ylabel 'Voltage (V)'\n");
197 //fprintf(gp1, "set yrange [-2:7]\n");
198 fprintf(gp1, "set xrange [0:%d]\n", FFT_NSAMPLES-1);
199 fprintf(gp1, "set grid\n");
200 fprintf(gp1, "plot '-' with lines notitle\n");
201 for (i = 0; i < FFT_NSAMPLES; i++) {
202     fprintf(gp1, "%d %1.3f\n", i, Volts(data[i]));
203 }
204 fprintf(gp1, "e\n");
205 //Save Freq domain in CSV
206 FILE *csvFile = fopen("freq.csv", "w");
207 if (!csvFile) {
208     perror("Failed to open file");
209 }
210 fprintf(csvFile, "Frequency (MHz),Magnitude (dB)\n";
211
212 // Plot FFT data in the second window
213 fprintf(gp2, "set title 'FFT of ADC Voltage'\n");
214 fprintf(gp2, "set xlabel 'Frequency (MHz)'\n");
215 fprintf(gp2, "set ylabel 'Magnitude (dB) '\n");
216 fprintf(gp2, "set yrange [-100:0]\n");
217 fprintf(gp2, "set xrange [0:.1f]\n", (Fs / 2) / 1e6);
218 fprintf(gp2, "set grid\n");
219 fprintf(gp2, "plot '-' using 1:2 with lines notitle\n");
220 for (i = 0; i <= FFT_NSAMPLES / 2; i++) {
221     freq = (double)i * Fs / FFT_NSAMPLES;
222     bin_val = 2*(out[i][0] * out[i][0] + out[i][1] * out[i]
223 [1])/(FFT_NSAMPLES^2);
224     magnitude = 10*log10(bin_val);
225     fprintf(gp2, "%f %f\n", freq / 1e6, magnitude);
226     fprintf(csvFile, "%f,%f\n", freq / 1e6, magnitude);
227 }
228 fclose(csvFile);
229 fprintf(gp2, "e\n");
230 pclose(gp1); // Close the first GNUpot pipe
231 pclose(gp2); // Close the second GNUpot pipe
232 fftw_destroy_plan(p);
233 fftw_free(in);
234 fftw_free(out);
235 }
236
237 int detect_trigger(void *buff, int nsamps, float threshold) {
238     uint16_t *array = (uint16_t *)buff; // Cast void* to uint16_t*

```

```

239     uint16_t t = (uint16_t)threshold;      // Convert float threshold to uint16_t
240     int consecutiveCount = 0;
241     t = (t*ADC_SCALE)+ADC_ZERO;
242     t = t<<4;
243     for (int i = 0; i < nsamps; i++) {
244         if (array[i] >= t) {
245             consecutiveCount++; // Increment counter when the condition is met
246             if (consecutiveCount == PULSE_NSAMPS) {
247                 return 1; // Return 1 if the counter reaches the width
248             }
249         }
250     }
251     return 0; // Return 0 if no value meets the threshold
252 }
253 void last_block(MEM_MAP *mp, DMA_CB *cbs, uint32_t *modes, volatile uint32_t *p)
254 {
255     // Control block 5: disable SMI I/O pins
256     cbs[5].ti = DMA_CB_SRCE_INC | DMA_CB_DEST_INC;
257     cbs[5].tfr_len = 3 * 4;
258     cbs[5].srce_ad = MEM_BUS_ADDR(mp, &modes[3]);
259     cbs[5].dest_ad = REG_BUS_ADDR(gpio_regs, GPIO_MODE0);
260     cbs[5].next_cb = 0x00000000;
261     *REG32(dma_regs, DMA_REG(DMA_CHAN_A, DMA_NEXTCONBK)) = MEM_BUS_ADDR(mp,
262     &cbs[5]);
263 }
264
265 // Map GPIO, DMA and SMI registers into virtual mem (user space)
266 // If any of these fail, program will be terminated
267 void map_devices(void)
268 {
269     map_periph(&gpio_regs, (void *)GPIO_BASE, PAGE_SIZE);
270     map_periph(&dma_regs, (void *)DMA_BASE, PAGE_SIZE);
271     map_periph(&clk_regs, (void *)CLK_BASE, PAGE_SIZE);
272     map_periph(&smi_regs, (void *)SMI_BASE, PAGE_SIZE);
273 }
274
275 // Initialise SMI, given data width, time step, and setup/hold/strobe counts
276 // Step value is in nanoseconds: even numbers, 2 to 30
277 void init_smi(int width, int ns, int setup, int strobe, int hold)
278 {
279     int divi = ns / 2;
280
281     smi_cs = (SMI_CS_REG*)REG32(smi_regs, SMI_CS);
282     smi_l = (SMI_L_REG*)REG32(smi_regs, SMI_L);
283     smi_a = (SMI_A_REG*)REG32(smi_regs, SMI_A);
284     smi_d = (SMI_D_REG*)REG32(smi_regs, SMI_D);
285     smi_dmc = (SMI_DMC_REG*)REG32(smi_regs, SMI_DMC);
286     smi_dsr = (SMI_DSR_REG*)REG32(smi_regs, SMI_DSR0);
287     smi_dsw = (SMI_DSW_REG*)REG32(smi_regs, SMI_DSW0);
288     smi_dcs = (SMI_DCS_REG*)REG32(smi_regs, SMI_DCS);

```

```

289     smi_dca = (SMI_DCA_REG*)REG32(smi_regs, SMI_DCA);
290     smi_dcd = (SMI_DCD_REG*)REG32(smi_regs, SMI_DCD);
291     smi_cs->value = smi_l->value = smi_a->value = 0;
292     smi_dsr->value = smi_dsw->value = smi_dcs->value = smi_dca->value = 0;
293     if (*REG32(clk_regs, CLK_SMI_DIV) != divi << 12)
294     {
295         *REG32(clk_regs, CLK_SMI_CTL) = CLK_PASSWD | (1 << 5);
296         usleep(10);
297         while (*REG32(clk_regs, CLK_SMI_CTL) & (1 << 7));
298         usleep(10);
299         *REG32(clk_regs, CLK_SMI_DIV) = CLK_PASSWD | (divi << 12);
300         usleep(10);
301         *REG32(clk_regs, CLK_SMI_CTL) = CLK_PASSWD | 6 | (1 << 4);
302         usleep(10);
303         while ((*REG32(clk_regs, CLK_SMI_CTL) & (1 << 7)) == 0);
304         usleep(100);
305     }
306     if (smi_cs->seterr)
307         smi_cs->seterr = 1;
308     smi_dsr->rsetup = smi_dsw->wsetup = setup;
309     smi_dsr->rstrobe = smi_dsw->wstrobe = strobe;
310     smi_dsr->rhold = smi_dsw->whold = hold;
311     smi_dmc->panicr = smi_dmc->panicw = 8;
312     smi_dmc->reqr = smi_dmc->reqw = REQUEST_THRESH;
313     smi_dsr->rwidth = smi_dsw->wwidth = width;
314 }
315
316
317 // Start SMI, given number of samples, optionally pack bytes into words
318 void smi_start(int nsamples, int packed)
319 {
320     smi_l->len = nsamples;
321     smi_cs->pxldat = (packed != 0);
322     smi_cs->enable = 1;
323     smi_cs->clear = 1;
324     smi_cs->start = 1;
325 }
326
327
328 // Start DMA for SMI ADC, return Rx data buffer
329 uint32_t *DMA_Start(MEM_MAP *mp, int nsamp, uint32_t *data, DMA_CB *cbs, uint32_t
*modes)
330 {
331     uint32_t *modep1=data+0x18, *modep2=modep1+1,i;
332     uint32_t *buffer=data+0x20;
333
334     // Get current mode register values
335     for (i=0; i<3; i++)
336         modes[i] = modes[i+3] = *REG32(gpio_regs, GPIO_MODE0 + i*4);
337     // Get mode values with ADC pins set to SMI
338     for (i=ADC_D0_PIN; i<ADC_D0_PIN+ADC_NPINS; i++)

```

```

339     mode_word(&modes[i/10], i%10, GPIO_ALT1);
340 // Copy mode values into 32-bit words
341 *modep1 = modes[1];
342 *modep2 = modes[2];
343 enable_dma(DMA_CHAN_A);
344 // Control blocks 0 and 1: enable SMI I/O pins
345 cbs[0].ti = DMA_SRCE_DREQ | (DMA_SMI_DREQ << 16) | DMA_WAIT_RESP;
346 cbs[0].tfr_len = 4;
347 cbs[0].srce_ad = MEM_BUS_ADDR(mp, modep1);
348 cbs[0].dest_ad = REG_BUS_ADDR(gpio_regs, GPIO_MODE0+4);
349 cbs[0].next_cb = MEM_BUS_ADDR(mp, &cbs[1]);
350 cbs[1].tfr_len = 4;
351 cbs[1].srce_ad = MEM_BUS_ADDR(mp, modep2);
352 cbs[1].dest_ad = REG_BUS_ADDR(gpio_regs, GPIO_MODE0+8);
353 cbs[1].next_cb = MEM_BUS_ADDR(mp, &cbs[2]);
354 // Control block 2: Pre-Sampling
355 cbs[2].ti = DMA_SRCE_DREQ | (DMA_SMI_DREQ << 16) | DMA_CB_DEST_INC;
356 cbs[2].tfr_len = PRE_SAMP * SAMPLE_SIZE;
357 cbs[2].srce_ad = REG_BUS_ADDR(smi_regs, SMI_D);
358 cbs[2].dest_ad = MEM_BUS_ADDR(mp, buffer);
359 cbs[2].next_cb = MEM_BUS_ADDR(mp, &cbs[3]);
360 // Control block 3: #1st Data-Buffer
361 cbs[3].ti = DMA_SRCE_DREQ | (DMA_SMI_DREQ << 16) | DMA_CB_DEST_INC | DMA_TI_INFINITE_TEN;
362 cbs[3].tfr_len = nsamp * SAMPLE_SIZE;
363 cbs[3].srce_ad = REG_BUS_ADDR(smi_regs, SMI_D);
364 cbs[3].dest_ad = MEM_BUS_ADDR(mp, buffer+2*PRE_SAMP);
365 cbs[3].next_cb = MEM_BUS_ADDR(mp, &cbs[4]);
366 // Control block 4: #2nd Data-Buffer
367 cbs[4].ti = DMA_SRCE_DREQ | (DMA_SMI_DREQ << 16) | DMA_CB_DEST_INC | DMA_TI_INFINITE_TEN;
368 cbs[4].tfr_len = nsamp * SAMPLE_SIZE;
369 cbs[4].srce_ad = REG_BUS_ADDR(smi_regs, SMI_D);
370 cbs[4].dest_ad = MEM_BUS_ADDR(mp, buffer+2*(nsamp+PRE_SAMP));
371 cbs[4].next_cb = MEM_BUS_ADDR(mp, &cbs[3]);
372 buffer+=PRE_SAMP/2;
373 start_dma(mp, DMA_CHAN_A, &cbs[0], 0);
374 return(buffer);
375 }
376
377
378 // ADC DMA is complete, get data
379 int Convert_Samples(void *buff, uint16_t *data, int nsamp)
380 {
381     uint16_t *bp = (uint16_t *)buff;
382     int i;
383
384     for (i=0; i<nsamp; i++)
385     {
386         *data++ = bp[i] >> 4;
387     }

```

```

388     return(nsamp);
389 }
390
391 // Display SMI registers
392 void disp_smi(void)
393 {
394     volatile uint32_t *p=REG32(smi_regs, SMI_CS);
395     int i=0;
396
397     while (smi_regs[i][0])
398     {
399         printf("%4s=%08X ", smi_regs[i++], *p++);
400         if (i%8==0 || smi_regs[i][0]==0)
401             printf("\n");
402     }
403 }
404
405 // Get GPIO mode value into 32-bit word
406 void mode_word(uint32_t *wp, int n, uint32_t mode)
407 {
408     uint32_t mask = 7 << (n * 3);
409
410     *wp = (*wp & ~mask) | (mode << (n * 3));
411 }
412
413 // Convert ADC value to voltage
414 float Volts(int val)
415 {
416     return((-1*ADC_ZERO + val) / ADC_SCALE);
417 }
418
419 // Return ADC value, using GPIO inputs
420 int GPIO_val(void)
421 {
422     int val = *REG32(gpio_regs, GPIO_LEV0);
423     return((val>>ADC_D0_PIN)&((1 << ADC_NPINS)-1));
424 }
425
426 // Display bit values in register
427 void disp_reg_fields(char *regstrs, char *name, uint32_t val)
428 {
429     char *p=regstrs, *q, *r=regstrs;
430     uint32_t nbits, v;
431
432     printf("%s %08X", name, val);
433     while ((q = strchr(p, ':')) != 0)
434     {
435         p = q + 1;
436         nbits = 0;
437         while (*p>='0' && *p<='9')
438             nbits = nbits * 10 + *p++ - '0';

```

```

439     v = val & ((1 << nbits) - 1);
440     val >>= nbits;
441     if (v && *r!= '_')
442         printf(" %.*s=%X", q-r, r, v);
443     while (*p==' ' || *p==' ')
444         p = r = p + 1;
445     }
446     printf("\n");
447 }
448 void save_to_csv(uint16_t *data, int nsamp)
449 {
450     FILE *fp = fopen("time.csv", "w");
451     if (fp == NULL) {
452         fprintf(stderr, "Failed to open file for writing.\n");
453         return;
454     }
455
456     fprintf(fp, "index,voltage\n");
457     for (int i = 0; i < nsamp; i++) {
458         fprintf(fp, "%d,% .3f\n", i, val_volts(data[i]));
459     }
460
461     fclose(fp);
462 }
463 // Catastrophic failure in initial setup
464 void fail(char* s)
465 {
466     printf(s);
467     terminate(0);
468 }
469
470 // Free memory segments and exit
471 void terminate(int sig)
472 {
473     int i;
474
475     printf("Closing\n");
476     if (gpio_regs.virt)
477     {
478         for (i = 0; i < ADC_NPINS; i++)
479             gpio_mode(ADC_D0_PIN + i, GPIO_IN);
480     }
481     if (smi_regs.virt)
482         *REG32(smi_regs, SMI_CS) = 0;
483     stop_dma(DMA_CHAN_A);
484     unmap_periph_mem(&vc_mem);
485     unmap_periph_mem(&smi_regs);
486     unmap_periph_mem(&dma_regs);
487     unmap_periph_mem(&gpio_regs);
488     exit(0);

```

489 | }

490 |

14.8 נספח ח' – קוד שמאמש לSIMOLCITY Matlab

```
% Parameters
pulseWidth = 0.5e-6; % Pulse width
fs = 25e6; % Sampling frequency
batchSize = 32768; % Number of samples in a batch

% Generate a rectangular pulse within the batch
pulseStart = randi([1, batchSize - round(pulseWidth * fs)], 1, 1); % Random start point
pulseEnd = pulseStart + round(pulseWidth * fs) - 1; % End point
signal = zeros(1, batchSize); % Initialize signal vector
signal(pulseStart:pulseEnd) = 2.4; % Set pulse values to 0.1v

% Plot the signal
figure;
plot(signal);
xlabel('Samples');
ylabel('Amplitude (V)');
title('Randomly Placed Rectangular Pulse at 2.4V');
ylim([-0.05 3]);
xlim([pulseStart-100, pulseEnd+100]); % Adjust xlim to focus on the pulse
grid on;

figure;
% Compute and plot FFT
fftSignal = fft(signal);
fftMag = 2*(abs(fftSignal).^2)/batchSize^2;
fftMag = 10*log10(fftMag);
freq = (0:batchSize-1) * (fs/batchSize) / 1e6; % Frequency vector in MHz
plot(freq(1:batchSize/2), fftMag(1:batchSize/2)); % Only plot up to Nyquist frequency
xlabel('Frequency (MHz)');
ylabel('Magnitude (dB)');
title('FFT');
```

```
xlim([0 12.5]);
```

```
grid on;
```

```
% Constants
```

```
k = 1.38e-23; % Boltzmann's constant in J/K
```

```
T = 300; % Temperature in Kelvin
```

```
R = 50; % Resistance in ohms
```

```
B = 5e6; % Bandwidth in Hz, cutoff at 5 MHz
```

```
nBits = 12; % Resolution of ADC
```

```
vRange = 10; % Voltage range from -5 to 5 volts
```

```
% Calculate noise voltage
```

```
V_noise = sqrt(4 * k * T * R * B);
```

```
% Generate Gaussian noise (thermal noise) with calculated standard deviation
```

```
noise_thermal = V_noise * randn(1, batchSize);
```

```
% Calculate quantization step size
```

```
quantStep = vRange / (2^nBits);
```

```
% Generate uniform quantization noise
```

```
noise_quantization = quantStep * (rand(1, batchSize) - 0.5);
```

```
% Add both noises to the signal
```

```
signal_noisy = signal + noise_thermal + noise_quantization;
```

```
% Plot the noisy signal
```

```
figure;
```

```

subplot(2,1,1);
plot(signal_noisy);
xlabel('Samples');
ylabel('Amplitude (V)');
title('Random Pulse with Thermal and Quantization Noise');
ylim([-0.05 6]);
xlim([pulseStart-100, pulseEnd+100]); % Adjust xlim to focus on the pulse
grid on;

% Compute FFT of the noisy signal
fftSignal = 2*fft(signal_noisy);
fftMag = 20 * log10(abs(fftSignal)/batchSize); % Convert to dB
freq = (0:batchSize-1) * (fs/batchSize) / 1e6; % Frequency vector in MHz

% Plot FFT
subplot(2,1,2);
plot(freq(1:batchSize/2), fftMag(1:batchSize/2)); % Only plot up to Nyquist frequency
xlabel('Frequency (MHz)');
ylabel('Magnitude (dB)');
title('FFT');
xlim([0 12.5]);
grid on;

```