

רובוט דו גלגלי מיוצב ע"י גלגל תגובה ויכולת התקדמות ע"י עיבוד תמונה

פרויקט גמר הנדסי

ME - 12

הוקן לשם השלמת הדרישות לקבלת

תואר ראשון בהנדסה B.Sc.

מאט

שי שביט אלכס רז

ד"ר חנן גלעדי

הוגש למחלקה להנדסת מכונות

המכללה האקדמית להנדסה סמי שמעון

אוקטובר 2024

תשס"ד

SCE

המכללה האקדמית להנדסה סמי שמעון

תתחבר לרעונות גדולים

באר שבע | אשדוד | ממהנדדים

www.sce.ac.il

רובוט דו גלגלי מיוצב ע"י גלגל תגובה ויכולת התקדמות ע"י עיבוד תמונה

פרויקט גמר הנדסי

הוכן לשם השלמת הדרישות לקבלת תואר ראשון בהנדסה B.Sc.

מאת

אלכס רז

שי שביט

מנחה

ד"ר חן גלעדי

הוגש למחלקה להנדסת מכונות

המכללה האקדמית להנדסה ע"ש סמי שמעון

קמפוס אשדוד

אוקטובר 2024

תשורי תשפ"ד

רובוט דו גלגלי מוצב ע"י גלגל תגובה ויכולת התקדמות ע"י עיבוד תמונה

פרויקט גמר הנדסי

הוכן לשם השלמת הדרישות לקבלת תואר ראשון בהנדסה B.Sc.

מאת

אלכס רז

שי שביט

מנחה

ד"ר חן גלעדי

ט' ט' ט' ט' ט' ט' ט'

הוגש למחלקה להנדסת מכונות

המכללה האקדמית להנדסה ע"ש סמי שמעון

קמפוס אשדוד

תקציר

פרויקט זה עוסק בתכנון ובנית אב טיפוס של רובוט דו-גלגלי מסוג "סאגויי", המציע גישה חדשה ליצוב. בינויו למערכות מסורתיות המתבססות על מנועים בגלגליים התחרטוניים בלבד, מטרת הפרויקט היא לייצב את הרובוט באמצעות גלגל תגובה שלישי, הממוקם מעל ציר הסיבוב של בגלגליים התחרטוניים. בנוסף, הרובוט תוכנן לzechot אובייקטים באמצעות עיבוד תמונה ולהגיב בהתאם - להמשיך בתנועה או לעצור.

במהלך הפרויקט בוצעו שישה ניסויים מקיפים לאפיון ואופטימיזציה של המערכת. הניסוי הראשון התמקד בייצוב המערכת באמצעות גלגל תגובה בלבד, ללא מעורבות מנועים נוספים. ניסוי זה לא הצליח מומנט לא מספק מגלגל התגובה, מה שהוביל לחיפוש פתרונות חלופיים.

בניסוי השני, נעשה ניסיון לייצב את המערכת באמצעות הורדת מרכזו הכובד, תוך שימוש במשקלות. ניסוי זה בוצע ללא התערבות של מנועים, ורק עם חישון תאוצה למידית השינויים במרחב. למרות זאת, הניסוי לא הצליח להשיג יציבות סביבה חזותית הרצויה.

הניסוי השלישי שילב את מערכת המרستان עם מנגנון הייצוב של גלגל התגובה. אולם, גם ניסוי זה לא הצליח וඅף הגדיל את הסטייה מהחזותית הרצויה בהשוואה לניסוי הקודם.

הפריצה המשמעותית הגיעה בניסוי הרביעי, כאשר הוחלט להסיר את המסתה המרستان ולהשתמש במנועים התחרטוניים של המערכת יחד עם המומנט שנוצר מגלגל התגובה. ניסוי זה הוכתר בהצלחה, עם המערכת מצליחה להתייצב סביב חזותית הרצויה לאורך זמן. הניסוי החמישי, שבחן את המערכת רק עם המנועים התחרטוניים, הדגש את חשיבותו של גלגל התגובה לייצוב יעיל.

לבסוף, הניסוי השישי והמכריע שילב את העיקרון המוצלח ביותר מהניסויים הקודמים עם יכולת עיבוד תמונה. המערכת הצליחה להתקדם תוך שמירה על יציבותה בהתבסס על נתונים מעיבוד התמונה, ובכך הושגה מטרת הפרויקט במלואה: פותח רובוט דו-גלגלי המתייצב באמצעות גלגל תגובה ומסוגל להתקדם תוך שימוש בעיבוד תמונה, תוך שמירה על יציבות.

תודות

ברצוננו להביע את תודתנו והערכתנו העמוקה לכל מי שהושיט יד ותרם מזמן ומרצו להצלחת הפרויקט.
תודה מכרב לב לד"ר גיא בן חמו, ראש המחלקה להנדסת מכונות, על תמיכתו ויעידותו לאורך כל הדרך.

אנו מודים לylimור ונאור על נכונותם לשיער בכל עת ועל מענה מהיר לכל בקשה.

תודה מיוחדת שמורה למנהל הפרויקט, ד"ר חן גלעדי, על הדרכתו המסורה, הכוונתו המקצועית והליווו הצמוד
לכל אורך התהליך.

תוכן עניינים

תקציר	4
תודות	5
תוכן עניינים	6
רשימת סימנים	7
רשימת טבלאות	9
רשימת איורים	9
פרק 1 – מבוא	12
1.1 תיאור הפרויקט	12
1.2 מטרות הפרויקט	12
1.3 חשיבות העבודה וモוטיבציה	13
פרק 2 – סקר ספרות	14
2.1 בקורת מערכת סאגוי	14
2.2 מערכת גלגל תגובה	16
2.3 עיבוד תמונה	17
פרק 3 – מפרט דרישות המערכת	19
פרק 4 – ניתוח מתמטי פיזיקלי של המערכת	20
4.1 רכיבי המערכת	20
4.2 הנחות	20
4.3 משוואות התנועה	20
4.4 בקרה	23
פרק 5 – מתודולוגיית הניסויים	25
5.1 ניסוי 1 – בוחינת יציבות המערכת בעזרת גלגל תגובה בלבד	25
5.2 ניסוי 2 – בוחינת המערכת תחת השפעת מסה מרסנת ולא גלגל תגובה	34
5.3 ניסוי 3 – בוחינת יציבות המערכת בשילוב גלגל תגובה ומסה מרסנת	43
5.4 ניסוי 4 – בוחינת יציבות המערכת בשילוב גלגל תגובה וגלגים תחתוניים	53
5.5 ניסוי 5 – בוחינת יציבות המערכת בהפעלת מנועים תחתוניים בלבד	64
5.6 ניסוי 6 – בוחינת יציבות המערכת (גלגל תגובה + מנועים תחתוניים) בשילוב עיבוד תמונה להתקדמות	75
פרק 6 – סיכום ומסקנות	94
פרק 7 –ביבליוגרפיה	96
פרק 8 –נספחים	97

97	8.1 מודל המערכת
97	8.1.1 מודל המערכת לפי ניסויים ללא משקלות (קונפיגורציה 1)
98	8.1.2 מודל המערכת לפי ניסויים כולל שימוש במשקלות (קונפיגורציה 2)
99	8.2 מודל המערכת BOM
99	8.2.1 מודל BOM - ניסויים ללא משקלות
101	8.3 רכיבי המערכת :
101	8.3.1 רכיבים לא סטנדרטים – הדפסות ועיבוד
106	8.3.2 רכיבים אלקטרוניים וסטנדרטים
109	8.4 שרטוטי תכנן המערכת – רכיבים לא סטנדרטים

רשימת סימנים

סימן	הגדרה	יחידות
θ	זווית הנטייה של גוף הרובוט ביחס לאנך	Rad
$\dot{\theta}$	מהירות זוויתית של גוף הרובוט	Rad/s
$\ddot{\theta}$	תאוצה זוויתית של גוף הרובוט	Rad/s ²
ϕ	זווית גלגל התגובה ביחס לגוף הרובוט	Rad
$\dot{\phi}$	מהירות זוויתית של גלגל התגובה	Rad/s
$\ddot{\phi}$	תאוצה זוויתית של גלגל התגובה	Rad/s ²
x	הموقع האופקי של מרכז המסה של הרובוט	m
\dot{x}	מהירות הרובוט בכיוון אופקי	m/s
\ddot{x}	תאוצה של הרובוט בכיוון אופקי	m/s ²
m	מסה של גוף הרובוט	kg
I_b	מומנט התמד של גוף הרובוט סביב ציר	kg·m ²
I_{rw}	מומנט התמד של גלגל התגובה	kg·m ²

m	המרחק מנקודת הציר של המטוולת למרכז המסה של גוף הרובוט	r
kg	המסה הכוללת של הרובוט	M
m/s^2	תאוצת הכבידה	g
$N \cdot m$	מומנט גלגל תגובה	τ_{rw}
$N \cdot m$	מומנט שפיעיל כוח הכביד על המערכת	τ_g
$N \cdot m$	מומנט גלגלים ראשיים	τ_w
m	רדיוס גלגלים ראשיים	r_w
N	כוח הפועל מהגלגלים הראשיים	F_w
לא מוחלט	מקדם תיקון שגיאה פרופורציונלי בבקר ה PID	K_p
לא מוחלט	מקדם תיקון שגיאה אינטגרלי בבקר ה PID	K_i
לא מוחלט	מקדם תיקון שגיאה דיפרנציאלי בבקר ה PID	K_d
לא מוחלט	ערך ה K_p האולטימטיבי הדרוש ליציבות לפי שיטת ניקולס-זיגלר	K_u
s	זמן שלוקח למערכת להשלים מחזור אושילציה אחד כאשר היא רוטטת בצורה יציבה	T_u

רשימת טבלאות

108 טבלה 1 – השוואת מנועים למערכת

רשימת איורים

15.....	איור 1 - רובוט דו גלגלי
26.....	איור 2 - מערכת ניסוי 1
26.....	איור 3 - גוף בסיס המערכת וגלגל תגובה
27.....	איור 4 - גוף בסיס המערכת ללא גלגל תגובה
28.....	איור 5 - תיאור אופן פעולה המערכת, ניסוי 1
29.....	איור 6 – תכנית ארדואינו 1 חלק 1
30.....	איור 7 – תכנית ארדואינו 1 חלק 2
31.....	איור 8 - – תכנית ארדואינו 1 חלק 3
32.....	איור 9 - גרפ' זווית-זמן, ניסוי 1
35.....	איור 10 - – מערכת הניסוי באיטרציה השלישייה, מועמסת ב 1 ק"ג
35.....	איור 11 – – מערכת הניסוי באיטרציה השנייה, מועמסת ב 1 ק"ג
36.....	איור 12 – – מערכת הניסוי באיטרציה הראשונה, מועמסת ב 0.5 ק"ג
36.....	איור 13 – חלק 1, תושבת מרכזית
37.....	איור 14 – חלק 2, מתיקת תלילית משקלות
37.....	איור 15 - – חלק 3, משקלות של 0.5 ק"ג
38.....	איור 16 - – תיאור אופן פעולה המערכת, ניסוי 2
39.....	איור 17 – – תכנית ארדואינו 2 חלק 1
39.....	איור 18 – – תכנית ארדואינו 2 חלק 2
41.....	איור 19 - – גרפ' זווית-זמן, ניסוי 2
43.....	איור 20 – – מערכת ניסוי 3 : גלגל תגובה יחד עם מסה מרנסת של 1 ק"ג
45.....	איור 21 – – תיאור אופן פעולה מערכת, ניסוי 3
46.....	איור 22 – – תיאור אופן פעולה מערכת, ניסוי 3
47.....	איור 23 – – תכנית ארדואינו 3 חלק 2
48.....	איור 24 – – תכנית ארדואינו 3 חלק 3
50.....	איור 25 – – גרפ' זווית-זמן, ניסוי 3.1
50.....	איור 26 – – גרפ' זווית-זמן, ניסוי 3.2
51.....	איור 27 – – גרפ' זווית-זמן, ניסוי 3.3
53.....	איור 28 – – תיאור מערכת ניסוי 4 : גלגל תגובה יחד עם פעולות מנועים תחתונים
55.....	איור 29 – – תיאור אופן פעולה המערכת, ניסוי 4
56.....	איור 30 – – תכנית ארדואינו 4 חלק 1
57.....	איור 31 – – תכנית ארדואינו 4 חלק 2
58.....	איור 32 – – תכנית ארדואינו 4 חלק 3
59.....	איור 33 – – תכנית ארדואינו 4 חלק 4

איור 34 – גרפ' זווית-זמן, ניסוי 4.1	61
איור 35 – גרפ' זווית-זמן, ניסוי 4.3	62
איור 36 – תיאור מערכת ניסוי 5 : מערכת כל בסיס גללים תחומיים וגלל תגובה מנוטק	64
איור 37 – תיאור מערכת ניסוי 5 : תיעוד גלל תגובה מנוטק	65
איור 38 – תיאור אופן פעולה מערכת הניסוי	67
איור 39 – תכנית ארדואינו 5 חלק 1	68
איור 40 - תכנית ארדואינו 5 חלק 2	69
איור 41 – תכנית ארדואינו 5 חלק 3	70
איור 42 – תכנית ארדואינו 5 חלק 4	70
איור 43 – גרפ' זווית-זמן, ניסוי 5.1	72
איור 44 – גרפ' זווית-זמן, ניסוי 5.2	72
איור 45 – גרפ' זווית-זמן, ניסוי 5.3	73
איור 46 – תיאור מערכת ניסוי 6 : מערכת על בסיס מנועים תחומיים וגלל תגובה משולב	76
איור 47 – תיאור מערכת ניסוי 6 : סימונו עוצר לעצירת המערכת	76
איור 48 – תיאור מערכת ניסוי 6 : סימונו רכב להתקדמות המערכת	76
איור 49 – תיאור אופן פעולה המערכת, ניסוי 6	78
איור 50 – תכנית ארדואינו 6.1 חלק 1	79
איור 51 – תכנית ארדואינו 6.1 חלק 2	80
איור 52 – תכנית ארדואינו 6.1 חלק 3	81
איור 53 – תכנית ארדואינו 6.1 חלק 4	82
איור 54 – תכנית ארדואינו 6.2 חלק 1	83
איור 55 – תכנית ארדואינו 6.2 חלק 2	84
איור 56 – תכנית ארדואינו 6.2 חלק 3	85
איור 57 – תכנית ארדואינו 6.2 חלק 4	86
איור 58 – תכנית פיתון 6.1 חלק 1	87
איור 59 – תכנית פיתון 6.1 חלק 2	88
איור 60 – תכנית פיתון 6.2	89
איור 61 – תכנית פיתון 6.2 חלק 2	89
איור 62 – גרפ' זווית-זמן, ניסוי 6.1	91
איור 63 – גרפ' זווית-זמן, ניסוי 6.2	91
איור 64 – גרפ' זווית-זמן, ניסוי 6	92
איור 65 - מודל סאגוי לניסויים ללא משקלות	97
איור 66 - מודל סאגוי לניסויים עם משקלות	98
איור 67 – BOM - סאגוי ללא משקלות	99
איור 68 - BOM - סאגוי עם משקלות	100
איור 69 - שילדה	101
איור 70 - תושבת מנוע	102
איור 71 - מתאם מנוע לגלגל	102

103	איור 72 - גלגל תגובה
103	איור 73 - תושבת עץ ספייסר
104	איור 74 - תפסנית למשkolot
104	איור 75 תפסנית למשkolot-- אגוליזות חוץ
105	איור 76 - בסיס המשkolot
105	איור 77 - מחבר תפסנית לבסיס המשkolot
106	איור 78 – H-Bridge
106	איור 79 - Arduino Uno
108	איור 80 - מנוע 20Dx41L
108	איור 81 - מנוע XD-37GB555-600
108	איור 82 - מנוע JGB37-520 DC
109	איור 83 – שרטוט שילדה
110	איור 84 – שרטוט תושבת מנוע
110	איור 85 – שרטוט מתאם מנוע לגלגל
111	איור 86 – שרטוט גלגל תגובה
111	איור 87 – שרטוט תפסנית למשkolot
112	איור 88 - שרטוט בסיס משkolot
112	איור 89 – שרטוט מחבר תפסנית לבסיס המשkolot

פרק 1 – מבוא**1.1 תיאורuproיקט**

בשנים האחרונות המחבר על רובוטים דו-גלגליים מואזנים עצמאית כבר תאוצה, זאת בשל המערכת הדינמית הלא-لينארית והלא יציבה שלהם, שמקשה על יישום גישות רובוטיות קונבנציונליות לשיליטה. הרובוטים הללו מבוססים במקור על מודל מוטולת הפוכה ועגלת, אך הם נעים למרחב בעודם שומרים על איזון המוטולת, ונדרשת בקרת משוב טוביה לשימירה על יציבותם. מספר שיטות בקרה כמו בקר PID ולינאריזציה סביב נקודת עבודה נבחנו לשיליטה ברובוטים אלו.

בכך רוב המערכות הדו-גלגליות הרובוטיות מתיצבות ומונעות על ידי מנוע השולט בכל גלגל, דבר אשר מחייב

סנכרון בין הגלגלים לבין עצמם באמצעות הבקר.

בקרא זה ומונעים אלו, דורשים השקעה תקציבית והנדסית גדולה יחסית.

על אף זאת עדין יכולת התאיצבות שלהם אינה מוחלטת.

בפרויקט זה ניצור מודל לרובוט דו-גלגלי אשר מיוצב באמצעות מנוע אחד, זאת באמצעות מערכת גלגל תוגובה.

מערכת גלגל תוגובה הינה מורכבת מגלגל שמחובר למנוע DC שמייצר מומנט הנשלט על ידי בקר.

העיקרון הינו שימוש תנע זוויתי בין הגלגל לבין הגוף אליו הוא מחובר, ובכך יציבות הרובוט נשמרת.

בנוסף, המערכת תנותן למרחב באמצעות מצלמה שתוותקו על גביה, ופיענוח המידע המועבר דרך תיסיע.

בhimnuot ממושלים העומדים לפניה.

לפיכך, המערכת תנועה באמצעות גלגליים ממונעים מתחתית, תתייצב בעזרת גלגל התוגובה המבוקר ותנותן

באמצעות עיבוד תמונה.

1.2 מטרות הפרויקט

- **תכננו וייצור מודל רובוט דו-גלגלי המיוצב בעזרת גלגל תוגובה, ובבעל יכולת התקדמות באמצעות עיבוד תמונה**

- **תכננו מערכת בקרת התאיצבות הדורשת מנוע אחד בלבד.**

- **תכננו ויישום יכולת עיבוד תמונה, כך שהסאגויים יימנע מהמכשולים העומדים בדרך, תוך שמירה על יציבותו.**

1.3 חשיבות העבודה ומוטיבציה

- חשיבות העבודה נובעת משלושה היבטים מרכזיים: בטיחות,יעילות כלכלית וחדשנות טכנולוגית. מבחינות בטיחות, מערכת גלגל התגובה מציעה אפשרות לשיפור היציבות של מערכות רובוטיות דו-גלגליות, מה שעשויה להפחית משמעותית את הסיכון לתאונות ופגיעות גופו הנפוצות בעת שימוש במערכות אלו. היבט היעילות הכלכלית מתבטא בכך שהשימוש במערכת גלגל תגובה מאפשר שימוש במנועים ובקרים פשוטים וזולים יותר, תוך השגת תוצאות דומות או אף טובות יותר מאשר במערכות מסורתיות יקרות יותר.
- החדשנות הטכנולוגית של הפרויקט מתבטאת בשילוב של טכנולוגיות מתקדמות כמו עיבוד תמונה יחד עם מערכת גלגל תגובה, פתיחה אפשריות חדשות לפיתוחים בתחום הרובוטיקה הנידחת. המוטיבציה מאחוריה העבודה נובעת מהרצון לפתח פתרונות חדשניים ויעילים לאתגרים קיימים בתחום הרובוטיקה, תוך שימוש בטכנולוגיות זמניות ובעלן נוכחהיחסית. זאת, במטרה להרחיב את האפשרויות ליישומים מעשיים של רובוטים דו-גלגליים בתחוםים שונים, כגון תחבורה אישית, לוגיסטיקה, ואפילו ישומים ביתיים או תעשיוניים, תוך שיפור הבטיחות והיעילות הכלכלית של מערכות אלו.

פרק 2 - סקר ספרות

2.1 בקורת מערכת סאגויו [1]

בקורת מערכת סאגויו דו-גלגלי היא תחום בתחום הרובוטיקה והbakra, בעיקר בשל הצורך לשמר על איזון דינמי ולמנוע נפילות, תוך כדי תנועה ושמירה על יציבות. מבחינה תאורטית, המערכת פועלת על עקרונות של פיזיקה, בקרה ופיתוח טכנולוגי מתקדם. עקרונות מרכזיים המאפיינים למערכת סאגויו דו-גלגלי.

מטוטלת הפוכה (Inverted Pendulum)

המערכת הדו-גלגלית דומה מאוד לבסיס הבקרה הקלאסית של מטוטלת הפוכה. מטוטלת הפוכה היא מערכת שבה מרכז הכובד של הגוף נמצא מעל נקודת התמיכה שלו. בשל כך, המערכת אינה יציבה באופן טבעי, ויש צורך להפעיל כוח חיצוני כדי לשמר על איזון.

בקורת רובוטים דו-גלגליים, כמו סאגויו, דורשת שמירה על איזון על ידי שימוש במדדים של זווית המערכת, מהירות הסיבוב ותנועת הרובוט. הרובוט משתמש בבקורת משוב כדי להתאים את תנועת הגלגלים בהתאם למידת הנטייה של גוף הרובוט.

חישוני זווית ותאוצה (Gyroscopes & Accelerometers)

חישונים אלו משמשים למדידת זווית הנטיה (roll angle) והאצה של הרובוט. הגירוסקופים עוזרים למדוד את השינויים בזווית הנטיה של הרובוט, בעוד שחיישני תאוצה מודדים את הכוחות המופעלים על הרובוט. יחד, הם מספקים את הנתונים הנדרשים כדי לקבוע את מידת הסטיטה של הרובוט מהיציבות, כך שנitin יהיה לתקן זאת בזמן אמת באמצעות בקרת מנועים.

בקורת משוב (Feedback Control)

כדי לשמר על יציבות, מערכת הסאגויו משתמשת בבקורת משוב. העיקרונו הוא להכניס את ערכי השגיאה (ההבדל בין מצב הרובוט המוגדר במצב הנוכחי) למערכת, כך שהמנועים יפעלו לתיקון השגיאה. בקרות נפוצות שימושות במערכת סאגויו כוללות:

- **בקר – PID Controller (Proportional-Integral-Derivative Controller)**

היאISON ו מהירות. הוא פועל על פי שלושה מרכיבים :

- מרכיב יחסי (P) מתיקן את השגיאה ב מהירות לפייחס ישיר לגודל השגיאה.
- מרכיב אינטגרלי (I) מתיקן שגיאות מצטברות לאורך זמן.
- מרכיב דיפרנציאלי (D) צופה את השגיאה לפי קצב השינוי הנוכחי.

- **LQR (Linear-Quadratic Regulator)** - שיטה מתקדמת יותר שמיועדת ליעול השיטה באמצעות

חישוב פונקציה ריבועית המזערת את השגיאה והמאץ הנדרש מהמנועים. שימוש ב LQR – עוזר לשמר על יציבות לאורך זמן בצורה חלקה ויעילה.

מודלים מתמטיים (Mathematical Models)

בקרת המערכת מבוססת על מודלים מתמטיים המייצגים את הדינמיקה של מערכת דו-גלגלי. המודל המתמטי העיקרי של סאגווי הוא מערכת דיפרנציאלית המתארת את תנועת הגוף והגלגלים :

- **משוואת תנועה (Equation of Motion)** מותארת את הקשר בין הכוחות הפועלים על הרובוט, כולל מומנטים, חיכוך, ותנאי התחלת. משוואות אלו נגוזות לחוקי ניוטון (ניוטון השני) או מנוסחאות לגראנז'יאניות.
- **תורת הבקרה הקלאסית והמודרנית** – משתמשים במטריצות מרחב מצב (State Space) ואופטימיזציה כדי לייצר בקרות מותאיימות למערכת.

(Balance and Movement Control)

אחרת האתגרים הגדולים במערכות דו-גלגליות היא שילוב בין ניהול הרובוט לבין יכולת לנوع בצורה חלקה. כדי שהרובוט יוכל לנوع תוך כדי שמירה על יציבות, המערכת צריכה לسانכרן את תנועת הגלגלים עם מצב הגוף המרכזי (גוף האדם או המטען שנמצא על הרובוט). השימוש בברק PID או LQR מאפשר תיקון השגיאה באופן שוטף כך שהרובוט יישאר יציב בעת התנועה.

(Steering Control)

בנוסך לאיזון ותנועה קדימה או אחורה, רובוט דו-גלגלי כמו סאגווי צריך יכולת להסתובב ולהחליף כיוון. תנועה כזו מתאפשרת על ידי כך שכל גלגל מסתובב במחירות שונות, בדומה למערכת של טנקים, שם גלגל אחד נע קדימה והשני אחורה.



איור 1 - רובוט דו-גלגלי [5]

2.2 מערכת גלגל תגובה [2]

מערכת גלגל תגובה (Reaction Wheel) היא טכנולוגיה שמקורה בתעשייה החלל, שם היא משמשת בעיקר לשימרה על יציבות של לוויינים וחלליות. עם זאת, הרעיון יושם גם ברובוטים ורכבים אוטונומיים כמו סאגוי. מטרתה של מערכת גלגל תגובה היא לספק מומנטום זוויתי שיאפשר שליטה על תנועות הגוף, תוך כדי שימירה על יציבות ללא תלות בקרקע.

עקרונות מנהים**1. מומנטום זוויתי**

מערכת גלגל תגובה פועלת לפי עקרון שימור המומנטום הזוויתי (Angular Momentum) כאשר גלגל מסתובב במהירות בתוך גוף נתון, הוא מייצר מומנטום זוויתי. אם המערכת מרגישה נטייה לכיוון מסוים, הגלגל יכול לשנות את מהירותו או את כיוון הסיבוב כדי ליציר מומנט זוויתי בכיוון הפוך לנטייה, מה שמאזן את הגוף.

2. חוקי ניוטון ויחסי פעולה ותגובה

כשגלגל מסתובב בתוך הסAGO, הוא מייצר כוח שמנסה לשובב את הגוף של הסAGO בכיוון הפוך. על ידי שליטה במהירות הסיבוב, ניתן לשלוט בזווית הנטייה של הרובוט ובכך לשמור על יציבותו.

3. בקרת משוב (Feedback Control)

כמו מערכות אחרות שבסיסן על איזון דינמי, גם מערכת גלגל תגובה מבוססת על שימוש בחישנים (כמו גירוסקופים ותאוצות) שמודדים את השינויים בזווית ובמהירות הזוויתית של הרובוט. המידע הזה נכנס לבקרה משוב המפעילה את הגלגל בהתאם לדרישות.

4. מודלים מתמטיים

המודל המתמטי שמייצג את מערכת גלגל התגובה דומה למודל של סיבוב סביב ציר מרכזי, כולל את המשוואה אשר :

$$\ddot{\phi}_{rw} \cdot I_{rw} = \tau_{rw}$$

τ_{rw} הוא המומנט שמייצר גלגל התגובה. ○

I_{rw} הוא מומנט ההתמד של הגלגל. ○

$\ddot{\phi}_{rw}$ הוא התאוצה הזוויתית של הגלגל. ○

5. בקרה דינמית: הבקרה של גלגל התגובהחייבת להיות מהירה ומדויקת. מכיוון שמדובר במערכת לא-لينארית, לעיתים קרובות משתמשים בקרים מתקדמים כמו PID או אפילו שיטות בקרה לא-لينאריות מתקדמות יותר.

2.3 עיבוד תמונה [3]

עיבוד תמונה הוא תחום במדעי המחשב ובמתמטיקה העוסק בניתות, עיבוד ושיפור של תמונות באמצעות אלגוריתמים ממוחשבים. המטרה היא להפיק מידע שימושי מהתמונות או לשפר אותן בדרך שנותecessaria ל-goal. זה יוזמי, הבנה או פעולה מסוימת. עיבוד תמונה משמש בתחוםים רבים כגון רפואי, תחבורה אוטונומית, רובוטיקה, זיהוי פנים ומערכות מעקב, ועוד.

Haar Cascade הוא אחד האלגוריתמים הנפוצים ביותר בתחום עיבוד תמונה וזיהוי אובייקטים, ובפרט בתחום **זיהוי פנים**. האלגוריתם מבוסס על שימוש בфиילטרים שבוצעים חישובים מהירים אך יעילים של אזוריים בתמונה. כדי לקבוע אם מדובר באובייקט מסוים או לא.

איך Haar Cascade עובד?

1. מאפייני (Haar-like Features):

- **Haar-like features** – האלגוריתם מבוסס על תכונות של האזור בתמונה שנקראות כהן ותבניות אלו הן תבניות של אזוריים בהירים וכחולים בתמונה, לדוגמה, הבדלים בין אזוריים כהים ובHIRIM בעין האנושית, באף או בפה.
- לכל תבנית יש צורה מלכנית והיא נבדקת על אזוריים שונים בתמונה כדי לאתר את האובייקט המבוקש.

2. אינטגרל תמונה (Integral Image):

- כדי להציג את חישוב התכונות של כל אזור בתמונה, משתמש בטכניקה שנקראת אינטגרל **תמונה**. זה זוהי טבלה שמספקת דרך מהירה לסכום את ערכי הpixels באזוריים מלכניים. זה מאפשר לחשב את המאפיינים של Haar בצורה מהירה מאוד.

3. Cascade Classifier

- האלגוריתם משתמש במבנה הנקרא **Cascade Classifier** (סיווג במפל). המבנה הזה מחולק לשבים (Stages) כשבכל שלב האלגוריתם מבצע בדיקות של תכונות Haar על חלקים מההתמונה. ככל שהתמונה עוברת יותר שלבים, הבדיקה הופכת להיות מדויקת יותר אך גם סלקטיבית יותר.
- בשלב הראשון, האלגוריתם מסנן את רוב התמונה ומבודד שאין בה אובייקטים כלל. לאחר מכן, רק האזוריים שעברו את השבים הראשונים ממשיכים לשבים הבאים, שבהם יש בדיקות נוספות.

4. **אימון האלגוריתם**

- כדי לאמן את האלגוריתם, נדרש לספק לו מאגר של תמונות המכילות את האובייקט המבוקש (למשל, פרצופים). האלגוריתם לומד את התכונות שמייצגות את האובייקט ובונה מערך של סיוגים שיבצע את הזיהוי.

5. **יישומים:**

- **זיהוי פנים:** היישום הפופולרי ביותר של Haar Cascade הוא זיהוי פנים. האלגוריתם מזהה את תווי הפנים, כגון עיניים, אף, פה ועוד.
- **זיהוי עצמים אחרים:** ניתן להשתמש באותו עיקרונו גם לזיהוי אובייקטים נוספים כמו מכוניות, רמזורים ועוד, כל עוד הוא מאומן עם סט נתונים מתאימים.

פרק 3 - מפרט דרישות המערכת

עיקר הדרישות מהמערכת היא התיאצבותה בעזרת המומנט המיווצר מהמנוע שבגלגל התגובה. גלגל התגובה אמור לספק את היציבות, בין אם בעת התקדמות המערכת המבוצעת משני הגלגלים הממונעים הראשיים של הסאגורוי, ובין אם במצב עמידה ועכירה. בנוסף על המערכת באמצעות מצלמה ואלגוריתם של עיבוד תמונה לזהות ולפענח את סביבתה.

1. מנוע גלגל התגובה :

- יספק מומנט מינימלי של $0.04 \text{ ניוטון-מטר (m*N)}$ לייצוב המערכת כאשר זווית הסטייה (θ) קטנה מ-5 מעלות בערך מוחלט.

2. שלדת המערכת :

- תעמוד בעומסים של עד 2 ק"ג מבלי להיכשל, על מנת להבטיח את יציבות ועמידות המערכת.

3. מערכת הבקרה והחישנים :

- תהיה פשוטה וחסכונית, עם עלות כוללת שלא תעלה על 50 דולר.
- תותאם לשיטת בקרה מסוג PID (פרופורציונלי-אינטרגרלי-דיפרנציאלי) לשיליטה מדויקת ויעילה.

4. תוכנת הבקרה :

- תפותח באמצעות פלטפורמת Arduino.
- תאפשר שליטה על מהירות ואופן פעולה המנועים, וכן על קבועי הבקרה.
- תרוץ בתדרות עדכון מינימלית של 100 הרץ (Hz) להבטחת תגובה מהירה ויציבה של המערכת.

5. מצלמת המערכת :

- תספק איכות ורזולוציה מינימלית של 720p.
- תפעל בקצב של 30 פריימרים לשנייה (fps) לפחות.
- תאפשר זיהוי ברור של רכיבים המוצגים לפניה למרחק של עד 2 מטרים.

- על הסאגורוי ביכולתו להיות בעל מסה כוללת שלא תעלה על 2.5 ק"ג, כדי להקטין את המומנט הדורוש מהמנועים הראשיים להניע את המערכת ולהקטין את מומנט האינרציה של המערכת, כך שיידרש מומנט קטן יותר מגלגל התגובה על מנת לייצב אותו.

- על הסאגורוי באופן כולל, ועל מערכת גלגל התגובה בפרט, להיות ממוקמים בטווח של 10 ס"מ מציר הגלגלים הראשיים, זאת כדי להקטין את המומנט הנדרש מגלגל התגובה.

- המערכת הכוללת תהיה סימטרית עם סטייה מרבית של 0.5 ± 0.5 ס"מ לאורך הציר האנכי שלה, כדי לא ליזור מומנט נוסף במערכת שיגרום לחוסר יציבות.

פרק 4 – ניתוח מתמטי פיזיקלי של המערכת

תיאור הדינמיקה של רובוט מטוטלת עם גלגל תגובה עליון אשר נע על בסיס 2 גלגלים תחתונים. המשוואות שמתארות את דינמיקת המערכת כוללות את הדינמיקה של נטיית הרובוט ודינמיקת התנועה האופקית שלו.

4.1 רכיבי המערכת

1. גלגל תגובה – גלגל המותקן בחלק העליון של הרובוט, שתפקידו לשנות בזווית הנטייה של גוף הרובוט על ידי ייצור מומנט זוויתי.
2. גוף המטוטלת – גוף הרובוט הראשי שמתחנה כמטוטלת הפוכה.
3. גלגלים תחתונים – שני גלגלים שמניעים את הרובוט קדימה ואחורה ותורמים לשמירה על היציבות.

4.2 הנחות

1. הרובוט נעה במישור דו-ממדי (תנועה אופקית וזוויתית בלבד).
2. הגלגל התגובה משפיע רק על היציבות הזוויתית (מומנט זוויתי) ולא על הכוחות המניעים אותו קדימה ואחורה.
3. המסה של הרובוט מרוכזת בגוף המרכזי (המטוטלת) ובגלגל התגובה.
4. הגלגלים התחתונים אינם מחליקים – התנועה היא גלגול טהור.

4.3 משוואות התנועה

1. דינמיקת הנטייה (אינטראקציה בין הגלגל לגוף הרובוט)

גלגל התגובה מייצר מומנט שמייצב את זווית הנטייה של גוף הרובוט. שימור התנועה הזוויתית נותן :

$$(2) \quad \tau_{rw} = I_b \ddot{\theta} + mr^2 \ddot{\theta} + I_{rw} \ddot{\phi}$$

כאשר τ הוא המומנט שפועל על גלגל התגובה.
 המומנטים $I_b \ddot{\theta}$, $mr^2 \ddot{\theta}$ מתארים את דינמיקת גוף הרובוט.
 והגורם $I_{rw} \ddot{\phi}$ מתאר את דינמיקת גלגל התגובה.
 כוח הכבידה שיוצר מומנט שמחזיר את המטוטלת לאנך הוא :

$$(3) \quad \tau_g = mg r \sin(\theta)$$

לכן משווה התנועה עברו הנטייה -

$$(4) \quad (I_b + mr^2) \ddot{\theta} + I_{rw} \ddot{\phi} = -mg r \sin(\theta)$$

2. דינמיקת גלגל התגובה

מומנט המנוע שמאפעיל את גלגל התגובה מתואר על ידי

$$(1) \quad \tau = I_{rw} \ddot{\phi}$$

המשפיע גם על הנטייה של הרובוט כאשר τ_w הוא המומנט שמאפעיל מנוע גלגל התגובה.

3. דינמיקת התנועה האופקית

התנועה האופקית של מרכז המסה של הרובוט תלויות בזווית הנטייה θ ומומנט שמאפעילים הגלגלים

$$(5) \quad M\ddot{x} = F_w$$

כאשר F_w הוא הכוח שהגלגלים מפעילים על הקruk, והוא תלוי במומנט שמאפעילים הגלגלים

$$(6) \quad F_w = M\ddot{x} = -\frac{\tau_w}{r_w}$$

כאשר r_w הוא רדיוס הגלגלים.

4. משווהה משולבת

על מנת לתאר את המערכת במלואה, יש לשלב את דינמיקת הנטייה ודינמיקת התנועה האופקית

משוואת הנטייה -

$$(7) \quad (I_b + mr^2)\ddot{\theta} + I_{rw} \ddot{\phi} = -mg r \sin(\theta)$$

5. לינארזציה לזוויות קטנות

במקרה של זוויות קטנות, ניתן לפשט את המשוואות:

משוואת הנטיה -

$$(8) \quad (I_b + mr^2)\ddot{\theta} + I_{rw}\ddot{\phi} = -mgr\theta$$

динמיקה גלגל התגובה:

$$(1) \quad I_{rw}\ddot{\phi} = \tau_{rw}$$

динמיקת התנועה האופקית:

$$(5) \quad M\ddot{x} = F_w$$

במערכת יש שני סוגי בקרים

- בקרת נטייה: (Tilt Control) מתייחסת לבקרת זווית הנטייה θ של הרובוט באמצעות גלגל התגובה.
- בקרת מהירות: (Velocity Control) מתייחסת לתנועה האופקית באמצעות הגלגלים התחרתוניים.

משוואות הנחוצות לבקרת PID

- משוואת התנועה הזוויתית של הרובוט :

$$(8) \quad (I_b + mr^2)\ddot{\theta} + I_w \ddot{\phi} = -mgr\theta$$

- משוואת בקרת התנועה האופקית :

$$(6) \quad F_w = M\ddot{x} = -\frac{\tau_w}{r_w}$$

תהליכי היונון ה PID

כדי לקבוע את המקדים הנכונים, יש מספר גישות נפוצות לביצוע היונון :

שיטת 1 - היונון ידני

1. נתחיל עם K_p בלבד :

- נגדיר את $0 = K_d = K_i$

- נתחיל להגדיל את K_p בהדרגה עד שהמערכת מגיבה בצורה יציבה אך לא תנודתית מדי.

2. נוסיף את K_d

- לאחר שהגענו ל K_p אשר נותן תגובה טובה, נוסיף את K_d כדי להקטין תנודות וליעיב את המערכת.

3. נוסיף את K_i

- לבסוף, נוסיף את K_i כדי לתקן שגיאות יציבות קטנות. נעשה זאת בזרירות, כדי לא לגרום לתנודות חדשות.

[4] Ziegler-Nichols שיטה 2 : ביוונון

זהי שיטה נפוצה יותר שימושה בניתוח של תגובת המערכת לשינויים חדים
 $K_i = K_d = K_p$
 מעלה את K_p עד שהמערכת מתחילה לרטוט בצורה יציבה.

נחשב את משך התנודות T_u

נשתמש בערכיים אלו כדי לחשב את מקדמי ה PID - בהתאם :

$$(9) \quad K_p = 0.6 * K_u$$

$$(10) \quad K_i = 1.2 * K_p / T_u$$

$$(11) \quad K_d = 0.075 * K_p * T_u$$

ביוונון עבור בקרת נטיה ובקרת תנועה

בקרת נטיה (Tilt Control)

נשתמש ב PID-כדי לשלוט על זווית הנטייה θ באמצעות גלגל התגובה. מטרתנו היא לשמור את הרובוט יציב סביב זווית 0 מעלות.

- K_p יגדיל את התגובה לשגיאה בזווית הנטייה.
- K_d יקטין את התנודות בזווית.
- K_i יפחית שגיאה מצטברת.

בקרת תנועה אופקית (Velocity Control)
 בקרת תנועה אופקית נשלטת על ידי הגלגלים התח托וניים, אשר עליהם לפצות על השגיאות בתנועה האופקית תוך שמירה על יציבות כללית.

בדיקות בפועל והתאמאה
 לאחר קביעת הערכיים הראשוניים, נצטרך לבצע ניסויים בפועל על הרובוט, למדוד את התגובה שלו, ובכך התאמות קטנות לפי הצורך עד שהמערכת תגיב בצורה יציבה ולא תנודות מוגזמות.

פרק 5 – מתודולוגיית הניסויים**5.1 ניסוי 1 – בוחנת יציבות המערכת בעזרת גלגל תגובה בלבד****מטרה**

בניסוי זה, הראשון מבין רצף הניסויים, אנו רוצחים לבדוק ולהראות מהי יכולת המנוע המרכזי, אשר מניע את גלגל התגובה, וזאת כדי להשיג את יציבות המערכת בעזרת גלגל התגובה בלבד. ניסוי זה חשוב מאוד מאחר והוא יקבע את אופן התקדמות הניסויים.

גלגל תגובה היא מערכת מכנית שנמצאת בשימוש רחב בלימודי וחקיר בקרה ורוביוטיקה. מערכת זו נועדה להציג את עקרונות בקרת התנועה ושמירה על יציבות. המערכת מבוססת על גוף ראשי המחבר לציר כך שהוא חופשי לנوع סיבוב ציר. כדי לשמר על הגוף במצב יציב, משתמשים בגלגל תגובה הממוקם בחלק העליון של המערכת. הגלגל מסתובב במהירות גבוהה ויוצר כוח תגובה שמשיע לשומר על יציבות הגוף הראשי. הרעיון המרכזי מאחריו מערכת זו הוא שהמומנט הנוצר על ידי שינוי מהירות הגלגל יגרום לתנועה נגדית של הגוף הראשי כדי לאזן את תנעתו. כאשר הגלגל מפסיק או מאט את סיבובו, הגוף הראשי ינוע בכיוונו הפוך כדי לשמר על שיווי משקל. חשוב להציג כי אין מצופה מהמערכת להצליח להתאושש מזוויות נתיחה גדולות $55^\circ \pm$ מעלות, אך כן מצופה כי תוכל לכל הפלות לתנוד סיבוב זווית 5° מעלות בדיזוק של $\pm 3^\circ$ מעלות.

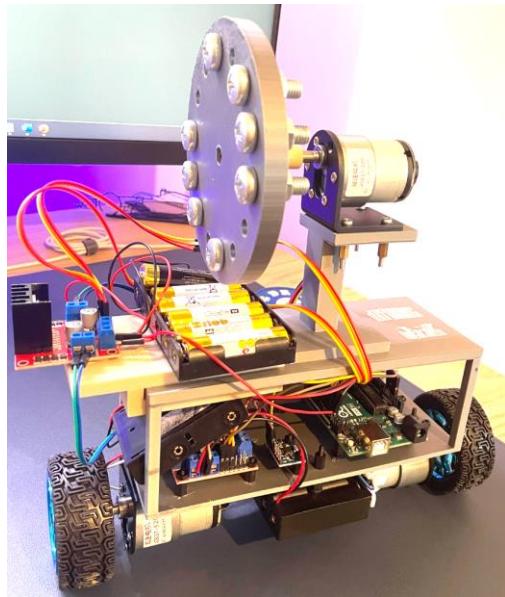
המטרה המרכזית בסדרת ניסויים אלו היא להוכיח כי ניתן לבנות מודול רובוט דו-גלגלי אשר מתייצב סיבוב זווית 0° מעלות בעזרת גלגל תגובה, וזאת באמצעות נגישים ורכיבים זולים. מרבית המאיצים והמחשבות הופנו לפתרית הבעה באמצעות בקרה ותוכנות.

תיאור המערכת

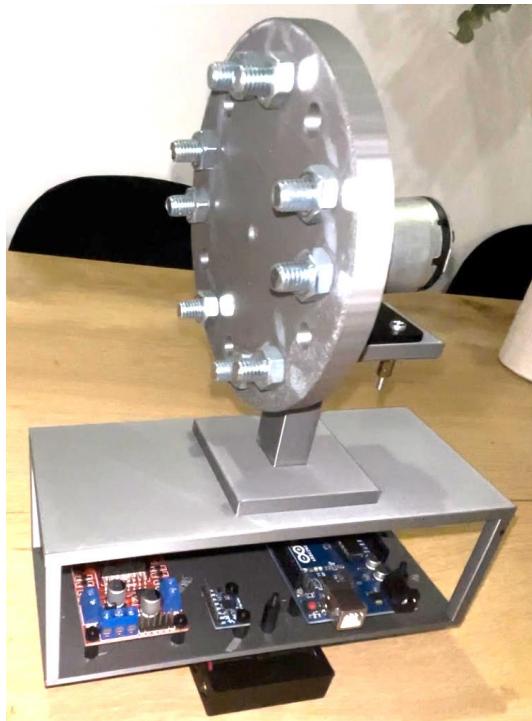
המערכת בנויה על בסיס גוף מודפס אשר עומד על 2 גלגלים. הגלגלים אינם מחוברים למקורה מתח ולבסוף נוע. על הגוף המודפס מחוברים כל רכיבי המערכת.

הגלגלים מהווים את הציר סיבבו המערכת מסתובבת. בחלק העליון של המערכת מחובר מנוע, אשר אליו מחובר גלגל התגובה. בהתאם למומנט המופעל על המערכת על ידי גלגל התגובה, המערכת נעה קדימה ואחוריה סיבוב ציר הסיבוב. המנוע אשר בחרנו לשמש בניסוי זה הוא בעל מהירות סיבוב גבוהה של 800 סל"ד ותפוקת מומנט של 0.4 Nm , אשר מתורגם ל- $0.04 \text{ m}^* \text{N}$. החישרונו בבחירה מנוע זה הוא תפוקת מומנט נמוכה יותר מאשר מנוע עם סל"ד נמוך יותר. הוויתור על מומנט כדי לקבל מהירות סיבוב הוא מתבקש מאחר我们知道 גלגל התגובה יגיב בחודות וכיitzlich להסתובב במהירות גבוהה.

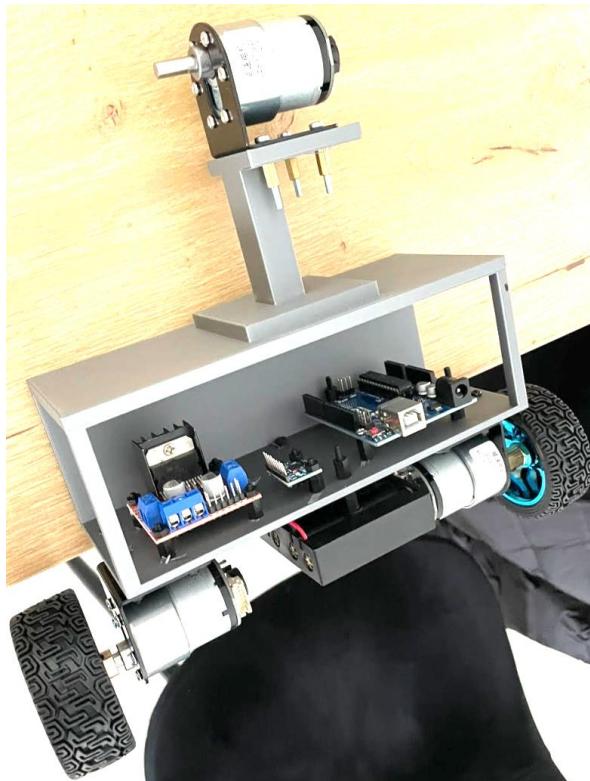
חשוב להציג כי מנוע זה הוא נגייל זול ולבן השיקול הכלכלי אינו גדול.



איור 2 - מערכת ניסוי 1



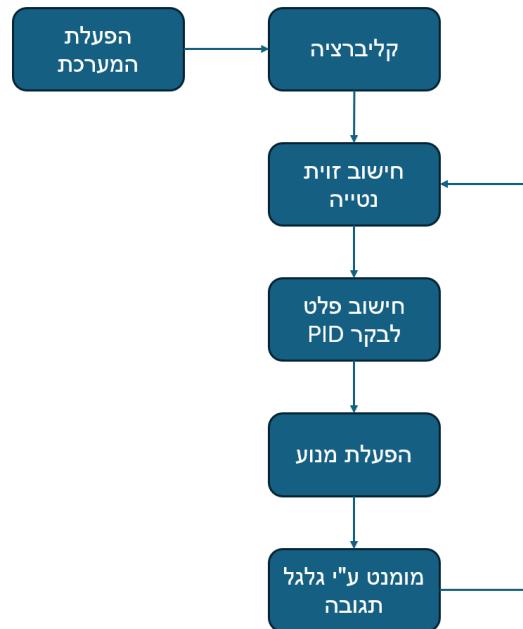
איור 3 - גוף בסיס המערכת וגלגל תגובה



איור 4 - גוף בסיס המערכת ללא גלגל תגובה

אופן פועלות המערכת

- המערכת מתחילה את פעילותה עם חיבורה למקור מתח (המחשב אליו מועברים ובו מנוטחים הנתונים). השלב הבא כולל קליברציה של זווית נטיית המערכת. שלב זה חשוב מאוד מאחר וסטייה גדולה מזוויות 0 מעלה תביא לחוסר איזון במערכת ואי יכלה להתיצב כלל.
- לאחר חישוב זוויות ההתחלה איתם תתחיל המערכת לפעול, מתחילה פועלות המערכת באופן לולאטי:
- דגימת זווית הנטיה וחישובה על ידי פילטר המשנן רעשים ותורם לדגימה איקונית ומדוקנת.
 - חישוב פלט בקר PID על ידי הגדרה מראש של מקדמי הבקר. שלב זה של הגדרת מקדמי הבקר חשוב מאוד מאחר ויש לו השפעה ישירה על תגובה המערכת.
 - שליחת אותן הנקודות לבקר למנוע. נרצה בתחום גבולות אלו מאחר ונרצה לקלוע לטוחה אותן אשר יפעיל מיידית את המנוע.
 - גלגל התגובה אשר מחובר למנוע, מסתובב בכיוון נגדי לכיוון נטיית המערכת. סיבוב זה מפעיל מומנט על המערכת סביב ציר הגלגלים התחתונים ובכך מתנגד לנפילה.



איור 5 - תיאור אופן פעולה המערכת, ניסוי 1

תכנית ארדזואינו

התוכנית נכתבה והופעלת ב Arduino IDE וכוללת את השלבים הבאים:

- הגדרת והפעלת חישון התאוצה.
- הגדרת חיבורים ללוח ארדזואינו.
- שלב קליברציה (Ấתחול זוויות התחליה).
- חישוב זווית הנטיה.
- בקרת PID – חישוב אות הפלט למנוע.
- בקרת מנוע – חישוב אופן הפעלת המנוע.
- הדפסת פלט לצורך ניתוח ומעקב.

```
1 #include <Wire.h>           // Includes the Wire library for I2C communication
2 #include <MPU6050_light.h>   // Includes the MPU6050 library for interacting with the MPU6050 sensor
3
4 // Initialize MPU6050 object with Wire (I2C)
5 MPU6050 mpu(Wire);
6
7 // Pin definitions for motor control
8 const int PWM_PIN = 5;      // PWM pin used to control motor speed (connected to motor driver)
9 const int IN1_PIN = 7;       // Pin to control motor direction (connected to motor driver)
10 const int IN2_PIN = 6;      // Pin to control motor direction (connected to motor driver)
11
12 // PID controller variables for maintaining balance
13 float Kp = 20000.0;        // Proportional gain (adjusts the motor power based on how far the system is from the setpoint)
14 float Ki = 10.0;           // Integral gain (helps eliminate steady-state error by considering the accumulated error over time)
15 float Kd = 50000.0;        // Derivative gain (reacts to the rate of change of the error, helping dampen oscillations)
16 float setpoint_angle = 0.0; // Desired angle of balance (setpoint) - 0 means upright
17 float lastAngleError = 0.0; // Used to store the previous error for the derivative term
18 float integral_angle = 0.0; // Used to accumulate the error over time for the integral term
19
20 // Timing variables for controlling the sample rate (time between updates)
21 unsigned long lastTime = 0; // Stores the last time the loop was updated
22 const unsigned long SAMPLE_TIME = 5; // Sample time for the loop in milliseconds (5ms = 200Hz)
23
24 // Timing variable for calculating delta time between PID updates
25 long prevT = 0;
26
27 void setup() {
28     Serial.begin(9600);    // Initialize serial communication at a baud rate of 9600 bits per second for debugging
29     Wire.begin();          // Initialize I2C communication with the MPU6050 sensor
30
31     // Initialize the motor control pins
32     pinMode(PWM_PIN, OUTPUT);
33     pinMode(IN1_PIN, OUTPUT);
34     pinMode(IN2_PIN, OUTPUT);
35
36     // Initialize the MPU6050 sensor
37     byte status = mpu.begin();
38     while (status != 0) { // If the MPU6050 fails to initialize, print an error message and retry
39         Serial.println("MPU6050 initialization failed. Please check your connections.");
40         delay(1000); // Wait 1 second before retrying
41     }
42
43     Serial.println("MPU6050 initialized successfully!"); // Confirm successful initialization
44
45     // Calibrate the MPU6050 to get accurate offsets for accelerometer and gyroscope
46     Serial.println("Calibrating MPU..."); 
47     mpu.calibrate(); // Automatically calculate the offsets for accurate readings
```

איו6 – תכנית ארדואינו 1 חלק 1

```
48   Serial.println("Calibration complete!");
49
50   // Print CSV header for data output (in this case, we only print the angle)
51   Serial.println("Time,Angle,MotorOutput"); // Commented out because we only print the angle in this version
52 }
53
54 void loop() {
55   // Get the current time
56   unsigned long currentTime = millis(); // millis() gives the number of milliseconds since the Arduino started running
57
58   // If enough time (SAMPLE_TIME) has passed since the last loop, proceed with the control
59   if (currentTime - lastTime >= SAMPLE_TIME) {
60     // Calculate the time that has passed since the last loop (delta time)
61     float deltaTime = (float)(currentTime - lastTime) / 1000.0; // Convert from milliseconds to seconds
62
63     // Update the MPU6050 sensor and calculate the current tilt angle
64     float angle = updateMPU(deltaTime);
65
66     // Calculate the required motor output using the PID controller
67     float motorOutput = calculatePID(angle, deltaTime);
68
69     // Set the motor speed and direction based on the PID output
70     setMotorSpeed(motorOutput);
71
72     // Output the current angle to the Serial Monitor (CSV format)
73     Serial.println(angle); // Only print the angle
74
75     // Update the last time the loop ran
76     lastTime = currentTime;
77   }
78 }
79
80 // Function to update MPU6050 and calculate the current tilt angle
81 float updateMPU(float deltaTime) {
82   mpu.update(); // Get updated sensor readings from the MPU6050
83   float rawAngle = mpu.getAngleX(); // Get the current angle around the X-axis (tilt angle)
84   return rawAngle; // Return the current tilt angle
85 }
86
87 // PID control function: calculates the required motor output based on the current angle
88 float calculatePID(float angle, float deltaTime) {
89   // Calculate the error (difference between the setpoint angle and the current angle)
90   float angleError = setpoint_angle - angle;
91
92   // Accumulate the error over time (integral term)
93   integral_angle += angleError * deltaTime;
```

איור 7 – תכנית ארדואינו 1 חלק 2

```

94     // Calculate the rate of change of the error (derivative term)
95     float derivative_angle = (angleError - lastAngleError) / deltaTime;
96
97     // Calculate the PID output (motor speed and direction)
98     float output = Kp * angleError + Ki * integral_angle + Kd * derivative_angle;
99
100    // Update the previous error for the next loop iteration
101    lastAngleError = angleError;
102
103    return output; // Return the motor control signal (output from the PID controller)
104}
105
106
107 // Function to control the motor speed and direction based on the PID output
108 void setMotorSpeed(float motorOutput) {
109     // If the motor output is positive, rotate the motor forward
110     if (motorOutput > 0) {
111         digitalWrite(IN1_PIN, HIGH); // Set motor direction to forward
112         digitalWrite(IN2_PIN, LOW);
113     }
114     // If the motor output is negative, rotate the motor in reverse
115     else {
116         digitalWrite(IN1_PIN, LOW); // Set motor direction to reverse
117         digitalWrite(IN2_PIN, HIGH);
118         motorOutput = -motorOutput; // Make the motor speed positive
119     }
120
121     // Set the motor speed using PWM (Pulse Width Modulation)
122     analogWrite(PWM_PIN, constrain(motorOutput, 0, 255)); // Constrain motor speed to the range 0-255 (valid for PWM)
123 }
```

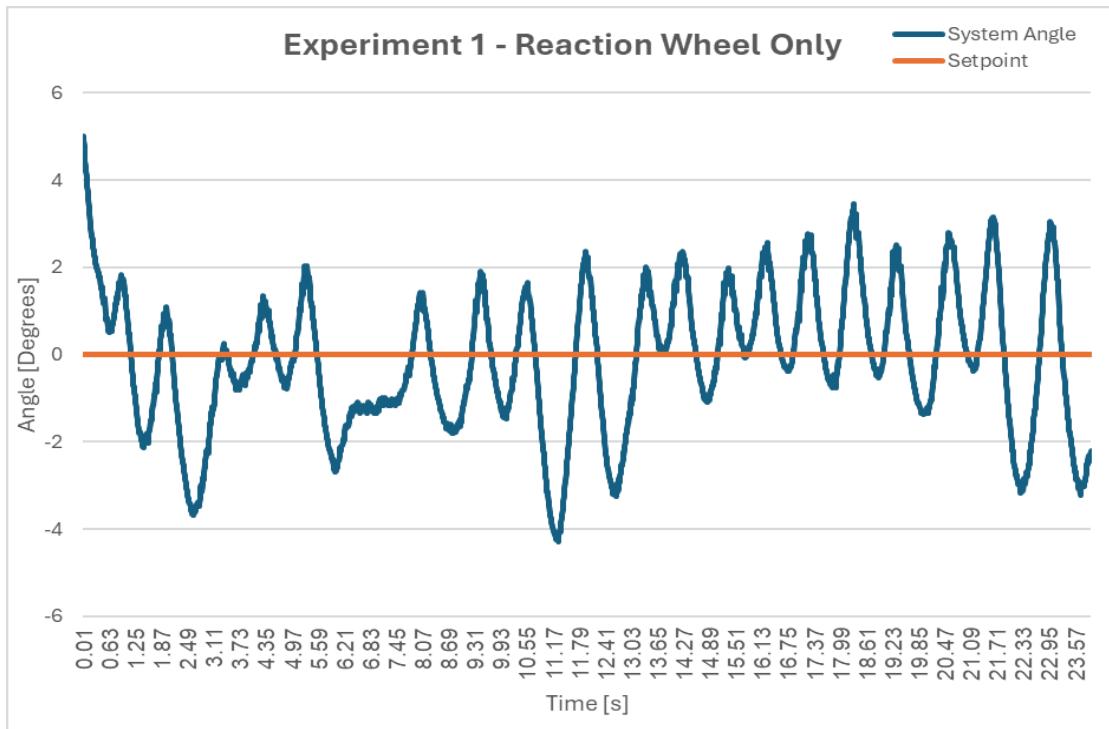
איור 8 תכנית ארדואינו 1 חלק 3

תוצאת הניסוי

ניתן לראות לפי הגרף מטה (אייר 9), כי המערכת תונדת סביב זווית 0 מעלות, והזווית אשר הוגדרה כזווית המטרה ואפיו מצלילה לעמוד בגבולות אותם הגדרנו במטרת הניסוי ($3 \pm$ מעלות).

למרות זאת, המנוע אשר משמש לשיבוב גלגל התגובה מתגלה כחלש מדי ואיינו מספק מומנט מספיק כדי לגרום למערכת לשמור באופן יציב על זווית 0 מעלות. דבר נוסף אשר הוכח הוא כי כמו שהשכנו בהתחלה, אין משמעות לנסות לגרום למערכת לצאת מזוויות נטיה הגדולות מ $10 \pm$ מעלות.

마וחר ונitin היה להסיק כבר מההתחלת כי עקב פשוטות המנוע ומחירו הזול, אין הדבר מפתיע שאי אפשר להשיג יציבות למערכת עם גלגל תגובה בלבד. נרצה לבחון שיטות עזר בין אם מכניות או תכניות כדי להגבר על מגבלות הרכיבים.



איור 9 - גרפ' זווית-זמן, ניסוי 1

דיון בתוצאות

תוצאות הניסוי הראשון מספקות תובנות חשובות לגבי היכולות והמוגבלות של מערכת הייצוב המבוססת על גלגל תגובה בלבד:

1. תנודות סיבב זווית המטרה: המערכת הצליחה לתנוד סיבב זווית 0 מעלות, שהוגדרה כזווית המטרה, ואך עדמה בגבולות שהוגדרו מראש של 3 ± 3 מעלות. זה הוכיח מעודדת המראה שעקרון הפעולה של גלגל התגובה אכן מסוגל להשפיע על יציבות המערכת.

2. חולשת המנוע: למרות ההצלחה החלקית, המנוע שנבחר (800 סל"ד, 0.04 m^3/N) התגלה כחלש מדי לשימורה על יציבות מלאה סיבב זווית 0 מעלות. זה מצביע על הצורך באיזון עדין בין מהירות סיבוב לבין תפוקת מומנט בבחירה המנוע.

3. מגבלות זווית הנטייה: הניסוי אישש את ההנחה המקדמת כי אין טעם לנסות לייצב את המערכת מזוויות נטיה הגדולות מ 10 ± 1 מעלות. זה הוכיח חשובה לתכנון מערכות בקרה עתידיות ולהגדלת גבולות הפעולה של המערכת.

4. יתרונות וחסרונות של רכיבים פשוטים: השימוש במנוע זול ונגיש אמן הגביל את ביצועי המערכת, אך סיפק נקודת תחילת טובת לניסויים ואפשר להבין את הנסיבות האמיתיות של המערכת.

5. כיוונים להמשך : התוצאות מצביעות על הצורך בבחינת שיטות עזר, הן מכניות והן תכניות, לשיפור יציבות המערכת. זה יכול לכלול שילוב של מנועים נוספים, שיפור אלגוריתמי הבדיקה, או שינויים במבנה המכני של המערכת.

6. השלכות על תכנון הניסויים הבאים : הניסוי הראשון סיפק בסיס חשוב להבנת התנагות המערכת וסייע בתכנוןיעיל יותר של הניסויים הבאים בסדרה, תוך התמקדות בשיפור נקודות החולשה שזוהו.

לסיכום, למרות שהניסוי לא הצליח להשיג יציבות מלאה באמצעות גלגל תגובה בלבד, הוא סיפק מידע חיוני על התנагות המערכת וסלל את הדרך לשיפורים ואופטימיזציות בניסויים הבאים.

5.2 ניסוי 2 – בוחינת המערכת תחת השפעת מסה מרסנת ולא גלגל תגובה

מטרה

בניסוי זה, אנו רוצים לבחון ולהראות מה קורה למערכת ללא גלגל התגובה מהניסוי הראשון. ראיינו כי השפעת גלגל התגובה על המערכת ניכרת בבירור אך לوكה בחסר מאחר והמנוע חלש מדי כדי לשולט באופן מוחלט בתונודת המערכת, והמערכת תונדת ללא התכונות לזרזות המטרה.

לצורך ניסוי זה, בנינו ו Tosf מסה מרסנת. מסה מרסנת היא מערכת מכנית המיועדת להפחית רuidות ותונודות מבנים, ומורכבת ממשה נעה המכוברת לבנייה הראשי. המטרה היא לספג את האנרגיה של התונודות ולהפחית את העומסים. המסה אשר נעה, מתוכננת כך שתנועה בכיוון נגדו לרuidות או התונודות, וכך נבלמת חלק מהאנרגיה של התונודות.

נדגיש כי בניסוי זה, גלגל התגובה אמנס קיים במערכת, אך איןנו מחובר למתח חשמלי ולכן איןנו פועל כלל ואיןנו משפיע על המערכת. נרצת למוד איך מתנהגת המערכת כאשר היא נתונה להשפעת מסה מרסנת בלבד. נתחל את הניסוי במסה של 0.5 ק"ג, ונתקדם בהפרשים של 0.5 ק"ג עד הגעה למסה כוללת של 1.5 ק"ג.

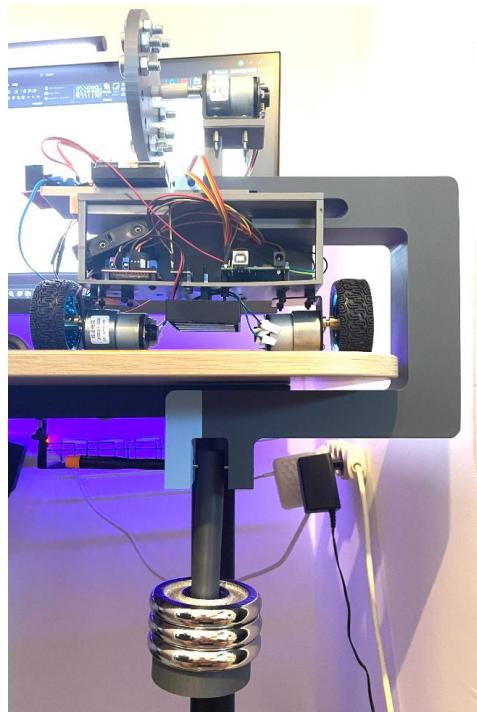
תיאור המערכת בניסוי

המערכת, כמו בניסוי הראשון, בנוייה על בסיס גוף מודפס אשר עומד על 2 גלגלים. הגלגלים אינם מחוברים למקור מתח ולן חופשיים לנوع. הגלגלים מהווים את הציר שביבו המערכת תונדת.

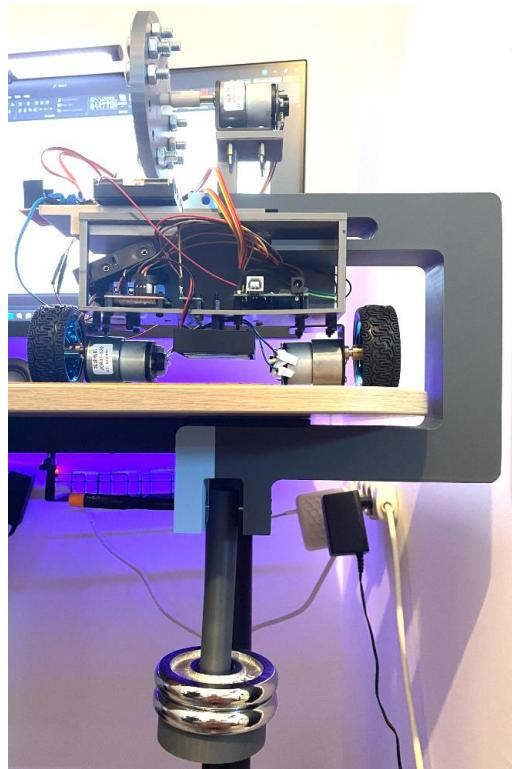
התונודה בניסוי זה מושגת על ידי חיבור מתקן אשר מסה מרסנת אשר מעצם קיומו מוריד את מרכז המסה של המערכת כלפימטה. הורידה זו של מרכז המסה אמורה לעזור להפחית את התונודות של המערכת.

מתקן המסה המרסנת מכיל 3 חלקים. החלק הראשון הוא התושבת אשר מתלבשת על המערכת מ从此 (איור 4). החלק השני הוא מתקן תלית המשקלות (איור 5). החלק השלישי הוא המשקלות (איור 6).

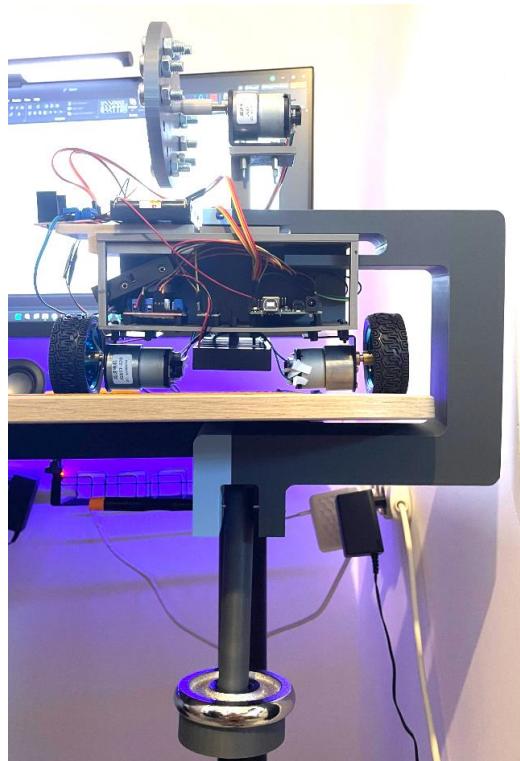
כדי לבחון מהימן את השפעת המסה המרסנת, כל הרכיבים מהניסוי הקודם נשארו במקום וזאת כדי לשמור על אופן פיזור המשקל במערכת.



איור 10 – מערכת הניסוי באיתרציה השלישית, מועמסת ב-1.5 ק"ג



איור 11 – מערכת הניסוי באיתרציה השנייה, מועמסת ב-1 ק"ג



איור 12 – מערכת הניסוי באיתרציה הראשונה, מועמסת ב-0.5 ק"ג



איור 13 – חלק 1, תוכבת מרכזית



איור 14 – חלק 2, מתקן תלוי משקלות



איור 15 – חלק 3, משקלות של 0.5 ק"ג

אופן פעולת המערכת

לפני תחילת הניסוי, נחבר את התושבת המרכזית (חלק 1) של מתקן המסה המרסנת למערכת. חיבור זה כורה מצד ימין של המערכת ולא מלפנים או מאחור כדי שמרכזי המסה הכלול של המערכת יישאר במקומו על ציר Z וرك יונמך על ציר Y.

לאחר חיבור התושבת המרכזית, נחבר את מתקן תלוי המשקלות (חלק 2) יחד עם המשקלות (חלק 3) בתחתית התושבת המרכזית. לאחר מכן נוכל להפעיל את המערכת ולציג את זווית הנטייה לאורך זמן של 30 שניות. בניסוי זה, הרכיבים היחידים אשר יפעלו במערכת הם לוח ארדיאנו וחישון תאוצה. הניסוי מתחילה כאשר המערכת מחוברת כשרורה למסה המרסנת ומתקן תלוי המשקלות (חלק 2) משוחרר בזווית של 5 מעלות.



איור 16 - תיאור אופן פעולה המערכת, ניסוי 2

תכנית ארדזואינו

התוכנית נכתבה והופעלה ב Arduino IDE וכוללת את השלבים הבאים:

- הגדרת ופעולת חיישן התאוצה.
- שלב קליברציה (אתחול זווית ההתחלת).
- חישוב זווית הנטיה.
- הדפסת פלט לצורך ניתוח ומעקב.

```

1 #include <Wire.h>           // Includes the Wire library for I2C communication
2 #include <MPU6050_light.h>   // Includes the MPU6050 library for interacting with the MPU6050 sensor
3
4 // Initialize MPU6050 object with Wire (I2C)
5 MPU6050 mpu(Wire);
6
7 // Timing variables for controlling the sample rate (time between updates)
8 unsigned long lastTime = 0; // Stores the last time the loop was updated
9 const unsigned long SAMPLE_TIME = 5; // Sample time for the loop in milliseconds (5ms = 200Hz)
10
11 // Timing variable for calculating delta time between PID updates
12 long prevT = 0;
13
14 void setup() {
15     Serial.begin(9600); // Initialize serial communication at a baud rate of 9600 bits per second for debugging
16     Wire.begin();        // Initialize I2C communication with the MPU6050 sensor
17
18
19 // Initialize the MPU6050 sensor
20 byte status = mpu.begin();
21 while (status != 0) { // If the MPU6050 fails to initialize, print an error message and retry
22     Serial.println("MPU6050 initialization failed. Please check your connections.");
23     delay(1000); // Wait 1 second before retrying
24 }
25
26 Serial.println("MPU6050 initialized successfully!"); // Confirm successful initialization
27
28 // Calibrate the MPU6050 to get accurate offsets for accelerometer and gyroscope
29 Serial.println("Calibrating MPU...");
30 mpu.calibrate(); // Automatically calculate the offsets for accurate readings
31 Serial.println("Calibration complete!");
32
33 }

```

אייר 17 – תכנית ארדואינו 2 חלק 1

```

34
35 void loop() {
36     // Get the current time
37     unsigned long currentTime = millis(); // millis() gives the number of milliseconds since the Arduino started running
38
39     // If enough time (SAMPLE_TIME) has passed since the last loop, proceed with the control
40     if (currentTime - lastTime >= SAMPLE_TIME) {
41         // Calculate the time that has passed since the last loop (delta time)
42         float deltaTime = (float)(currentTime - lastTime) / 1000.0; // Convert from milliseconds to seconds
43
44         // Update the MPU6050 sensor and calculate the current tilt angle
45         float angle = updateMPU(deltaTime);
46
47         // Output the current angle to the Serial Monitor (CSV format)
48         Serial.println(angle); // Only print the angle
49
50         // Update the last time the loop ran
51         lastTime = currentTime;
52     }
53 }
54
55 // Function to update MPU6050 and calculate the current tilt angle
56 float updateMPU(float deltaTime) {
57     mpu.update(); // Get updated sensor readings from the MPU6050
58     float rawAngle = mpu.getAngleX(); // Get the current angle around the X-axis (tilt angle)
59     return rawAngle; // Return the current tilt angle
60 }
61

```

אייר 18 – תכנית ארדואינו 2 חלק 2

תוצאת הניסוי

ניתן לראות באIOR מטה (AIOR 19), פلت של 3 גرافים לפי שלושת האיטרציות שעשינו למערכת. כל איטרציה עולה במשקל ב- 0.5 ק"ג.

האיטרציה הראשונה מועמסת עם משקל של 0.5 ק"ג, תונדת סביב 10- מעלות. תנודה זאת הגיונית מאחר שהמשקל אינו מספיק די כדי להוריד את מרכזו המסה של המערכת כלפי מטה. כמו כן המערכת אינה מתכנסת וה坦ודות המשיכות עוד זמן רב.

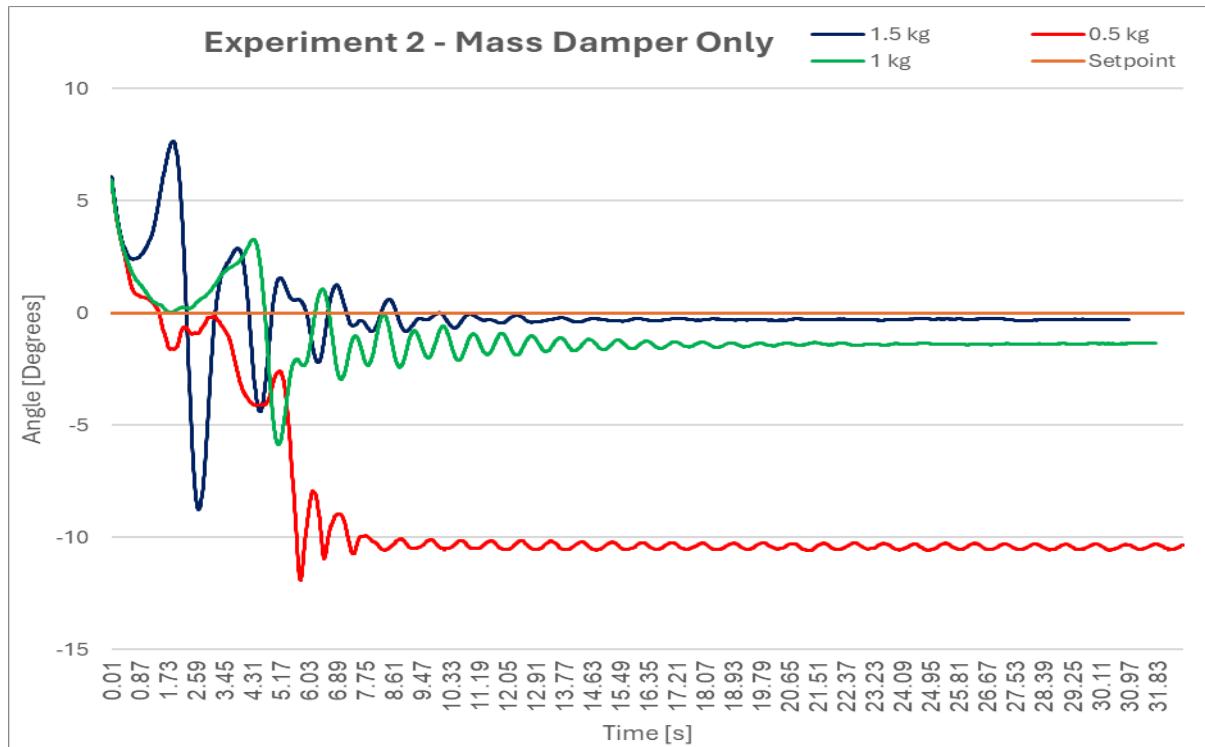
האיטרציה השנייה מועמסת עם משקל של 1 ק"ג, תונדת סביב 1.5- מעלות. ניתן לראות שיפור משמעותית בהתנהגות המערכת בעיקר בגלל התכונות מהירה יותר לזרזות אשר מתקרבת לזרזת המטרה של 0 מעלות. אפשר להסיק שהכפלת פי 2 של העומס מביאה לתוצאה פי 6 טוביה יותר בעניין ההתכנסות לזרזת המטרה.

האיטרציה השלישית מועמסת עם משקל של 1.5 ק"ג, תונדת סביב 0.5- מעלות. השיפור משמעותית עוד יותר וכמעט שמאפשר להתכנס לזרזת המטרה. אפשר ליחס את הסטייה מזרזת המטרה לעצם העובדה שמרכזו המסה של המערכת על ציר Z מלכתחילה אינו נמצא בראשית הצירים, אלא עם סטייה מוגברת כלפי הכיוון הקדמי בגלל אופן סיור וחיבור הרכבים.

ניתן להגדיר ניסוי זה כמושכל אחר וקיבלו מידע חדמשמעותי לגבי השפעת מסה מרנסת על יכולת המערכת להתכנס לזרזת המטרה כאשר מתקיימים התנאים הנכונים, אשר מושגים בעורת הורדת מרכזו המסה כלפי מטה על ציר Z.

דבר נוסף אשר ניתן להסיק מהניסוי הוא שהאיטרציה השלישית אשר הועמסה ב- 1.5 ק"ג, יוצרת עומס רב מדי בהסתכוות עתידית על יכולת המערכת לנوع קדימה ואחוריה.

אחר והאיטרציה הראשונה אינה משפיעה על תנודות המערכת, נבחר את האיטרציה השנייה עם עומס של 1 ק"ג להיות העומס אשר השתמש בו בניסוי הבא, והוא שילוב של יכולת גלגל התגובה יחד עם מסה מרנסת.



איור – 19 גרפ' זווית-זמן, ניסוי 2

דיון בתוצאות

ניסוי 2 הדגים באופן ברור את ההשפעה המשמעותית של המסה המרנסת על יציבות המערכת. עם הגדלת המסה מ-0.5 ק"ג ל-1.5 ק"ג, נצפה שיפור דרמטי ביכולת המערכת להתכנס לזוויות המטרה של 0 מעלות. התוצאות מראות כי מסה מרנסת יכולה להיות כלי יעיל לייצוב מערכת ללא צורך במנגנונים אקטיביים כמו גלגל תגובה.

האופטימיזציה של המסה הראתה תוצאות מעניינות. עם 0.5 ק"ג, המערכת תנעה סביב- 10 מעלות, מה שמייד על חוסר מספיק במסה להשפעה משמעותית. עם 1 ק"ג, נצפה שיפור דרמטי עם תנודות סביב- 1.5 מעלות, המהווה שיפור של פי 6 בהתקנות. המסה של 1.5 ק"ג הביאה את המערכת לכך קרוב לזוויות המטרה (סביב- 0.5 מעלות), אך עוררה חששות לגבי עומס יתר על המערכת.

למרות השיפור הניכר, המערכת עדין לא הגיעו לייצוב מושלים סביב- 0 מעלות, מה שמרמז על הצורך בשילוב עם מנגנונים נוספים. הسطיה הקליה מ-0 מעלות גם במשקל המרבי מדגישה את החשיבות של איזון מדויק ומיקום נכון של מרכז המסה. אלה הם גורמים קריטיים שיש לנקוט בחשבון בתכנון עתידי של המערכת.

הבחירה ב-1 ק"ג כמסה לניסויים הבאים מהויה פשרה טובה בין יציבות לבין ביצועים דינמיים. זהה החלטה חכמה המאזנת בין היתרונות של יציבות משופרת והחסרונות של עומס יתר על המערכת, ומספקת בסיס טוב להמשך הניסויים.

לסיכום, ניסוי זה סיפק תובנות חשובות לגבי השימוש במסה מרסנת לייצוב מערכות דו-גלגליות. התוצאות מצביעות על הפוטנציאל בשילוב מסה מרסנת עם מגנונים אקטיביים כמו גלגל התגובה בניסויים עתידיים. הניסוי הדגים את האיזון העדין הנדרש בין יציבות לביצועים, וספק בסיס מוצק להמשך פיתוח ואופטימיזציה של המערכת.

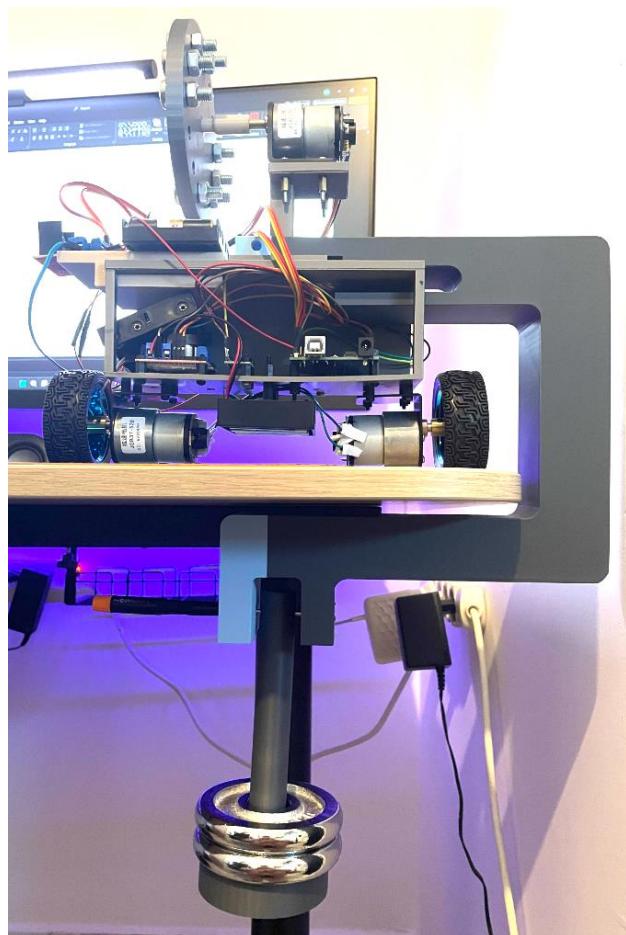
5.3 ניסוי 3 – בוחינת יציבות המערכת בשילוב גלגל תגובה ומסה מרסנת

מטרה

בהמשך לניסוי הקודם, בו חיברנו התקן מכני של מסה מרסנת למערכת, ובכך הקטנו את תנודות המערכת והגדלו את יכולת המערכת להתקנס לזרזית המטריה, נרצה להתקדם一步 ולבחון את יכולת הייצוב של המערכת בשילוב גלגל תגובה מניסוי מס' 1 עם מסה מרסנת מניסוי מס' 2. נרצה לבוחן בכמה משתפרת, אם בכלל, יכולת התכונות המרתקות לזרזית המטריה ואיזה מגבלות נוצרות בשילוב כזו.

תיאור המערכת בניסוי

המערכת מתבססת על המשך ישיר מןיסוי מס' 2, כאשר ההבדל המשמעותי הוא שהפעם כל הרכיבים החשמליים משתתפים, בפרט המנוע אשר שולט על גלגל התגובה. בניסוי זה השתמש במסה מרסנת משקל של 1 ק"ג לאחר והסקנו מןיסוי מס' 2 כי משקל זה מהוועה יחס עלות-תועלת גבוהה מבחינה משקל כולל למערכת, וייתן לנו את האפשרות להשיג את מטרות הניסוי ללא הפעלת עומס מיותר על המערכת בכללותה.



איור 20 – מערכת ניסוי 3 : גלגל תגובה יחד עם מסה מרסנת של 1 ק"ג

אופן פעולה המערכת

לפני תחילת הניסוי, לחבר את התושבת המרכזית (חלק 1) של מתקן המסה המרטנת למערכת. לאחר חיבורו, לחבר את מתקן תלוי המשקולות (חלק 2) יחד עם המשקולות (חלק 3) במשקל כולל של 1 ק"ג. לאחר חיבור מוצלח של רכיבי המסה המרטנת, המערכת מתחליה את פעילותה בדומה לניסוי מס' 1

- חיבור המערכת למקור מתח (מחשב, אליו מועברים נתונים הפלט לשם ניתוחם)
- השלב הבא כולל קליברציה של זווית נתית המערכת. שלב זה חשוב מאוד מאחר וסטיטה גדולה מזוינת 0 מעלות תביא לחוסר איזון במערכת ואי יכולת להתייצב כלל.

לאחר חישוב זווית ההתחלה איתם תחילת המערכת לפעול, מתחילה פעולה המערכת באופן לולאטי

- דגימת זווית הנתית וחישובה על ידי פילטר המסנן רעשים ותורם לדגימה איקוטית ומדויקת.
- חישוב פלט בקר PID על ידי הגדרה מראש של מקדמי הבקר. שלב זה של הגדרת מקדמי הבקר חשוב מאוד מאחר ויש לו השפעה ישירה על תגובת המערכת.
- שליחת אותן הנקודות הבקר למנוע. נרצה בתחום את גבולות אותן לאחר ונרצה לקבוע לטווח אותן אשר יפעיל מיידית את המנוע.
- גלגל התגובה אשר מחובר למנוע, מסתובב בכיוון נגדו לכיוון נתית המערכת. סיבוב זה מפעיל מומנט על המערכת סיבוב ציר הגלגלים התחתונים ובכך מתנגד לנפילה.

במשך כל זמן פעולה המערכת, המסה המרטנת תונדת קדימה ואחוריה כדי להקל על יכולת ההתקנסות לזרמת המטרה.



איור 21 – תיאור אופן פעולה מערכת, ניסוי 3

תכנות ארדזואינו

התוכנית זהה לאורך כל ניסוי מס' 3 למעט שינוי מקדמי הבקר אשר מפורטים בשלב התוצאות.
התוכנית נכתבה והופעלת ב Arduino IDE וכוללת את השלבים הבאים :

- הגדרת והפעלת חישון התאוצה.
- הגדרת חיבורים ללוח ארדזואינו.
- שלב קליברציה (אתחול זוויות התחלה).
- חישוב זווית הנטיה.
- בקרת PID – חישוב אותן הפלט למנוע.
- בקרת מנוע – חישוב אופן הפעלת המנוע.
- הדפסת פלט לצורך ניתוח ומעקב.

```
1 #include <Wire.h>           // Includes the Wire library for I2C communication
2 #include <MPU6050_light.h>   // Includes the MPU6050 library for interacting with the MPU6050 sensor
3
4 // Initialize MPU6050 object with Wire (I2C)
5 MPU6050 mpu(Wire);
6
7 // Pin definitions for motor control
8 const int PWM_PIN = 5;      // PWM pin used to control motor speed (connected to motor driver)
9 const int IN1_PIN = 7;       // Pin to control motor direction (connected to motor driver)
10 const int IN2_PIN = 6;      // Pin to control motor direction (connected to motor driver)
11
12 // PID controller variables for maintaining balance
13 float Kp = 20000.0;         // Proportional gain (adjusts the motor power based on how far the system is from the setpoint)
14 float Ki = 0.0;             // Integral gain (helps eliminate steady-state error by considering the accumulated error over time)
15 float Kd = 6000.0;          // Derivative gain (reacts to the rate of change of the error, helping dampen oscillations)
16 float setpoint_angle = 0.0; // Desired angle of balance (setpoint) - 0 means upright
17 float lastAngleError = 0.0; // Used to store the previous error for the derivative term
18 float integral_angle = 0.0; // Used to accumulate the error over time for the integral term
19
20 // Timing variables for controlling the sample rate (time between updates)
21 unsigned long lastTime = 0; // Stores the last time the loop was updated
22 const unsigned long SAMPLE_TIME = 8; // Sample time for the loop in milliseconds (5ms = 200Hz)
23
24 // Timing variable for calculating delta time between PID updates
25 long prevT = 0;
26
27 void setup() {
28     Serial.begin(9600);    // Initialize serial communication at a baud rate of 9600 bits per second for debugging
29     Wire.begin();          // Initialize I2C communication with the MPU6050 sensor
30
31     // Initialize the motor control pins
32     pinMode(PWM_PIN, OUTPUT);
33     pinMode(IN1_PIN, OUTPUT);
34     pinMode(IN2_PIN, OUTPUT);
35
36     // Initialize the MPU6050 sensor
37     byte status = mpu.begin();
38     while (status != 0) { // If the MPU6050 fails to initialize, print an error message and retry
39         Serial.println("MPU6050 initialization failed. Please check your connections.");
40         delay(1000); // Wait 1 second before retrying
41     }
42
43     Serial.println("MPU6050 initialized successfully!"); // Confirm successful initialization
44 }
```

איור 22 – תיאור אופן פועלת מערכת, ניסוי 3

```
45 // Calibrate the MPU6050 to get accurate offsets for accelerometer and gyroscope
46 Serial.println("Calibrating MPU...");
47 mpu.calcOffsets(); // Automatically calculate the offsets for accurate readings
48 Serial.println("Calibration complete!");
49
50 // Print CSV header for data output (in this case, we only print the angle)
51 Serial.println("Time,Angle,MotorOutput"); // Commented out because we only print the angle in this version
52 }
53
54 void loop() {
55     // Get the current time
56     unsigned long currentTime = millis(); // millis() gives the number of milliseconds since the Arduino started running
57
58     // If enough time (SAMPLE_TIME) has passed since the last loop, proceed with the control
59     if (currentTime - lastTime >= SAMPLE_TIME) {
60         // Calculate the time that has passed since the last loop (delta time)
61         float deltaTime = (float)(currentTime - lastTime) / 1000.0; // Convert from milliseconds to seconds
62
63         // Update the MPU6050 sensor and calculate the current tilt angle
64         float angle = updateMPU(deltaTime);
65
66         // Calculate the required motor output using the PID controller
67         float motorOutput = calculatePID(angle, deltaTime);
68
69         // Set the motor speed and direction based on the PID output
70         setMotorSpeed(motorOutput);
71
72         // Output the current angle to the Serial Monitor (CSV format)
73         Serial.println(angle); // Only print the angle
74
75         // Update the last time the loop ran
76         lastTime = currentTime;
77     }
78 }
79
80 // Function to update MPU6050 and calculate the current tilt angle
81 float updateMPU(float deltaTime) {
82     mpu.update(); // Get updated sensor readings from the MPU6050
83     float rawAngle = mpu.getAngleX(); // Get the current angle around the X-axis (tilt angle)
84     return rawAngle; // Return the current tilt angle
85 }
86
87 // PID control function: calculates the required motor output based on the current angle
88 float calculatePID(float angle, float deltaTime) {
89     // Calculate the error (difference between the setpoint angle and the current angle)
90     float angleError = setpoint_angle - angle;
91 }
```

איור 23 – תוכנית ארדואינו 3 חלק 2

```
92 // Accumulate the error over time (integral term)
93 integral_angle += angleError * deltaTime;
94
95 // Calculate the rate of change of the error (derivative term)
96 float derivative_angle = (angleError - lastAngleError) / deltaTime;
97
98 // Calculate the PID output (motor speed and direction)
99 float output = Kp * angleError + Ki * integral_angle + Kd * derivative_angle;
100
101 // Update the previous error for the next loop iteration
102 lastAngleError = angleError;
103
104 return output; // Return the motor control signal (output from the PID controller)
105 }
106
107 // Function to control the motor speed and direction based on the PID output
108 void setMotorSpeed(float motorOutput) {
109     // If the motor output is positive, rotate the motor forward
110     if (motorOutput > 0) {
111         digitalWrite(IN1_PIN, HIGH); // Set motor direction to forward
112         digitalWrite(IN2_PIN, LOW);
113     }
114     // If the motor output is negative, rotate the motor in reverse
115     else {
116         digitalWrite(IN1_PIN, LOW); // Set motor direction to reverse
117         digitalWrite(IN2_PIN, HIGH);
118         motorOutput = -motorOutput; // Make the motor speed positive
119     }
120
121     // Set the motor speed using PWM (Pulse Width Modulation)
122     analogWrite(PWM_PIN, constrain(motorOutput, 0, 255)); // Constrain motor speed to the range 0-255 (valid for PWM)
123 }
124 }
```

איור 24 – תכנית ארדואינו 3 חלק 3

תוצאת הניסוי

נתחיל ונאמר כי תוצאות ניסוי מס' 3 אינם עומדים במטרה שהגדכנו בתחילת, אך למדנו המונע על גלגל התגובה כאשר משולב יחד עם מסה מרסנת.

בגרף הראשון (איור 5) אנו רואים את פלט המערכת לאחר 30 שניות פעולה. ניתן לראות כי המערכת תנודת באופן הרמוני סביב 5 מעלות ואינה מתכנסת לזוויות המטרה. תוצאה זו מעניינת מהסיבה כי ניתן היה לחושב שמערכת מסה מרסנת בלבד אשר מתכנסת לזוויות של 1.5- מעלות, תצליח בעורת בקרה בחוג סגור של גלגל התגובה להתכנס עוד יותר לכיוון זווית המטרה.

מקדמי בקר PID אשר היה בהם שימוש בגרף זה הם $KD = 20000$, $KP = 6500$, $KI = 20000$.

בגרף השני (איור 6) אנו רואים את פלט המערכת לאחר 60 שניות פעולה והתאמאה של מקדמי בקר PID. המקדים הפעם הם $KD = 25000$, $KP = 6000$, $KI = 25000$. הגברנו מעט עם יכולת הבקר הפרופורציונלי והורדנו מעט את יכולת הבקר שאחראי לקצב השינוי.

גם בפעם זו נוכחנו לראות כי המערכת תנודת. אמן התנודת קטנה להיות סביב 3 מעלות אך עדין לא תוצאה רצואה.

כפי שניתן לראות בגרף השלישי (איור 7), המשכנו לכוון את מקדמי הבקר אך לא היה ניכר שינוי משמעותי. הפעם הוספנו גם בקר אינטגרלי כדי להתמודד עם הסחיפה בזמן של זווית הנטייה.

מקדמי הבקר בפעם זו הם $KD = 6200$, $KP = 12000$, $KI = 28000$. הוספה הבקר לא תרמה למערכת וזוויות הנטייה עדין התכנסה ל-3 מעלות.

תוצאות הניסוי מצביעות על כך שהמערכת המשולבת אינה מתנהגת כצפו, ובמקום לשפר את יכולת התכנס לזוויות המטרה, היא גורמת לנו להתרחק ממנה. ישנו מספר סיבות אפשריות:

1. סתירה בין דינמיקת המרשן לבין דינמיקת גלגל התגובה. יתכן שהמרשן והגלגל פועלים האחד כנגד השני.

המרשן מיועד להפחית תנודות על ידי תנועה בהתאם לנטיית המערכת, בעוד הגלגל מייצר תנע זוויתי שמתוקן את הנטייה על ידי סיבוב בכיוון הפוך. אם אין סyncron ראיו וגובהו בין המערכות, הן עלולות לייצר כוחות אשר סותרים אחד את השני.

המרשן עשוי למשוך את המערכת לכיוון אחד, בעוד שהגלגל מנסה לתקן בכיוון הפוך. דבר זה מוביל לתנודות מוגברות במקום שיכוך התנודות.

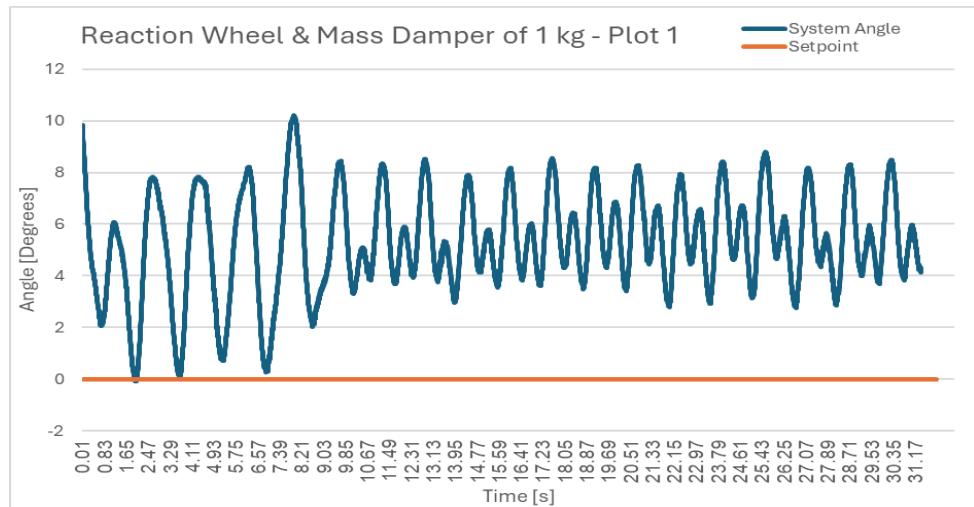
2. תדרים טבעיים של שתי המערכות אינם מכונים לעבוד בשילוב. אם הן אינם מכובנות כך שיישלימו אחד את השני, הם מגבירות תנודות המערכת הכללית.

3. בקרת גלגל התגובה אינה תואמת לפעולת משולבת עם המרשן כיון שאינו מתחשב בהשפעתו והדבר מוביל לתיקון ורישון יתר, מה שיגדיל את התנודות.

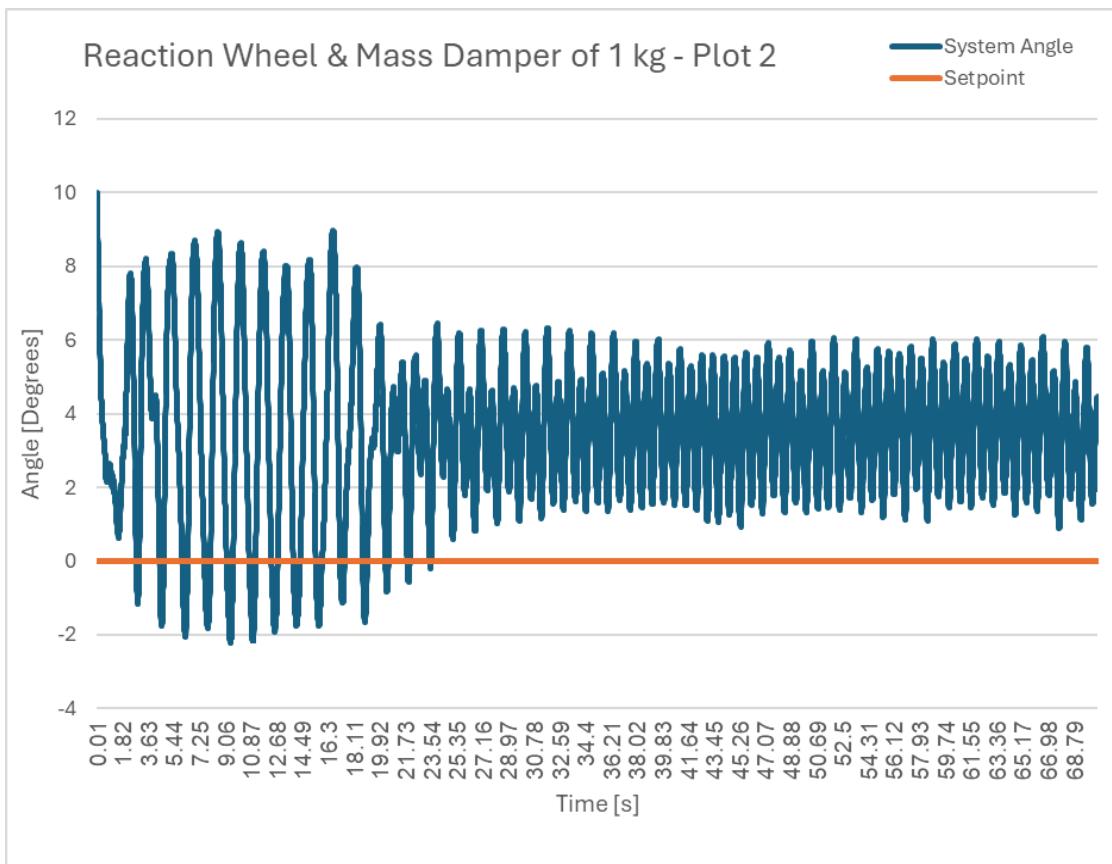
4. השפעות לא-لينאריות. המרשן והגלגל עשויים להכנס אי-لينאריות למערכת. יתכן שאחד מהרכיבים מתנהג בצורה לא-لينארית בתנאים מסוימים, והשילוב שלהם כולל לגורום לתנודות יתר בלתי צפויות.

לניסוי הבא, אנו רוצחים להכניס פתרונות לרוב המסקנות שהגענו אליהם בניסוי זה. הדבר אשר מוביל אותנו הוא פישוט המערכת מביצירת הבקרה אך כזה שיאפשר להשיג את מטרת הניסויים.

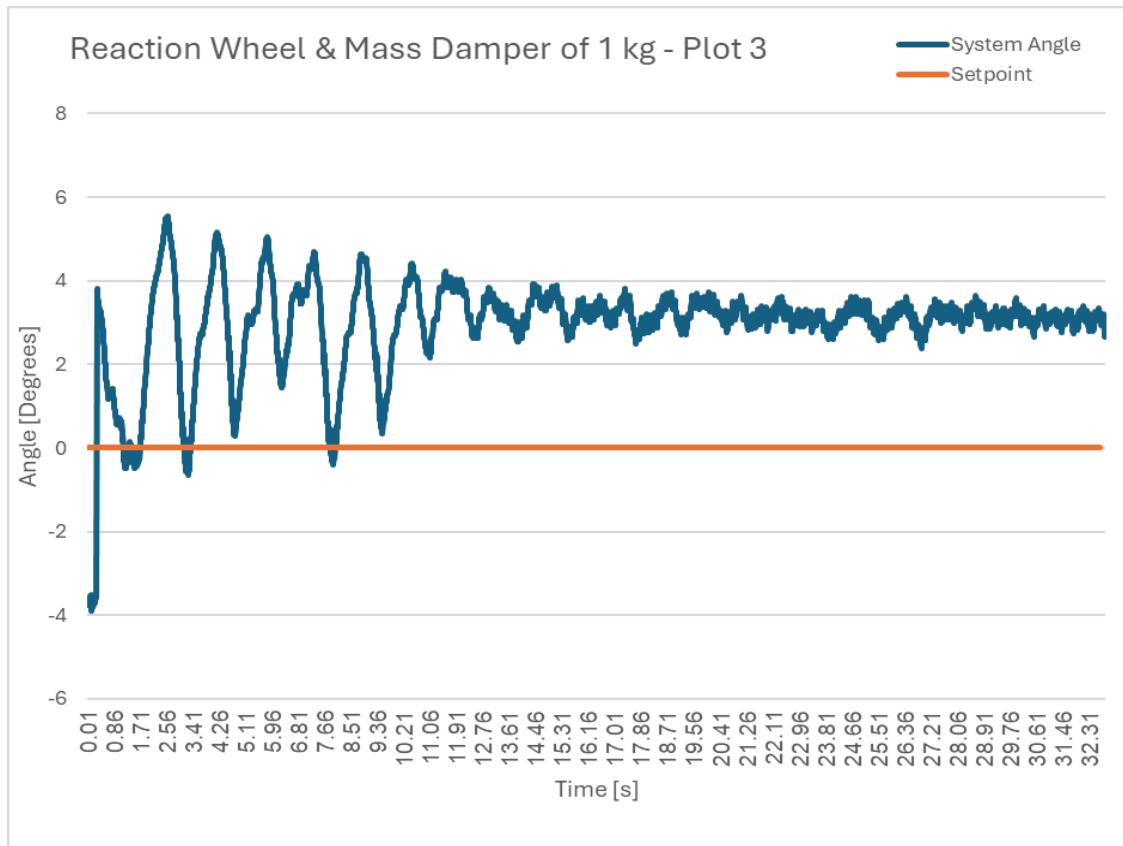
נרצה לקחת בחשבון את הגלגים עליהם עומדת המערכת ואשר משמשים כציר הסיבוב למערכת, ולגרום לה להתייצב בעורתם, בשילוב עם גלגל התגובה.



איור 25 – גרפ' זווית-זמן, ניסוי 3.1



איור 26 – גרפ' זווית-זמן, ניסוי 3.2



איור 27 – גראף זווית-זמן, ניסוי 3.3

דיון בתוצאות

1. תוצאות בLAT צפויות : ניסוי 3, אשר שילב את גלגל התגובה עם המסה המרנסת, הניב תוצאות מפתיעות ובלתי צפויות. בנגוד לציפיות המקוריות, השימוש לא הביא לשיפור ביציבות המערכת אלא דווקא הרחיק אותה מזווית המטרה. המערכת תנעה באופן חרמוני סביב 5 מעלות, מה שמהווה סטייה גדולה יותר מאשר בניסויים הקודמים. תוצאה זו מדגישה את המורכבות של שילוב מערכות בקרה שונות ואת הצורך בהבנה מעמיקה של האינטראקציות ביניהן.

2. השפעת מקדמי הבקר : ניסיונות לכוון את מקדמי בקר ה-PID הביאו לשיפור מסוים, אך לא מספק. הגדלת המקדם הפרופורציונלי (KP) והחטאת המקדם הדיפרנציאלי (KD) הצלחו להקטין את התנודות לשביות 3 מעלות, אך עדין רחוק מהיעד המקורי. הוספת המקדם האינטגרלי (KI) לא הביאה לשיפור נוסף, מה שמרמז על מגבלות בגישה הבקרה הנווכה ועל הצורך בחשיבה מחדש על אסטרטגיית הבקרה.

3. אינטראקציות מורכבות : התוצאות מצביעות על אינטראקציות מורכבות בין גלגל התגובה והמסה המרנסת. ייתכן שקיים מתאר בין דינמיקת המרנסן לבין דינמיקת גלגל התגובה, כאשר כל אחד מנסה לתכנן את המערכת בדרך שונה, מה שוביל לתנודות מוגברות. בנוסף, ייתכן שהתדרים הטבעיים של שתי המערכות אינם מסונכרנים, מה שגורם להגברת התנודות במקום להפחיתן.

4. השפעות לא ליניאריות: הניסוי חשף את האפשרות להשפעות לא ליניאריות במערכת. השימוש של המרנס והגלאל עשוי להכניס מרכיביות נוספות שאינן ניתנות לתיאור או לבקרה באמצעות מודל ליניארי פשוט. זה מדגיש את הצורך בגישה מתקדמת יותר לבקרה, אולי תוך שימוש בשיטות בקרה לא ליניאריות או אדפטיביות.

5. כיוונים להמשך: למרות שהניסוי לא השיג את מטרתו המקורית, הוא סיפק תובנות חשובות להמשך המחקר. הממצאים מובילים למסקנה כי יש צורך בפתרוט המבנה ובחשיבה מחדש על אסטרטגיית הבקרה. הרעיון לשלב את הגלגלים התתconjנים בתהליך הייצור, יחד עם גלגל התגובה, מציע כיוון מבטיח להמשך. זה עשוי לאפשר ניצול עיל יותר של כל מרכיבי המערכת ולהוביל לפתרון יציב יותר.

5.4 ניסוי 4 – בוחינת יציבות המערכת בשילוב גלגל תגובה וגלגלים תחתוניים

מטרה

בניסוי מס' 3 למדנו כי שילוב גלגל התגובה יחד עם מסה מרסנת איננו מביא לתוצאה הרצויה של התוכניות לזרזית המטרה.

בניסוי זה אנו רוצים לבדוק שילוב נוסף, של גלגל התגובה יחד עם המנועים התחתוניים של המערכת. לאחר והמנועים התחתוניים של המערכת משמשים כציר הסיבוב שלה, הפעלת מומנט בגלגליים אלו צריך להגדיל את ההשפעה על תנודות המערכת ולתרום להתקדמות לזרזית המטרה.

נרצה ללמידה אם שילוב שני הגורמים יכול לשמש כבסיס להתקדמות בנית המערכת לעבר המטרה המרכזית בפרויקט זה, בנית רובוט דו-גלגלי אשר מתיצב בעורת גלגל תגובה ומתקדם על ידי עיבוד תמונה.

תיאור המערכת בניסוי

המערכת מתבססת על המשך ישיר מניסוי מס' 1, כאשר ההבדל המרכזי הוא שהמנועים התחתוניים, אלה המניעים את הגלגלים התחתוניים עליהם עומדת המערכת ואשר משמשים כציר הסיבוב שלה, מחוברים למתחה חשמלי ותורמים את חלקם באופן פעולות המערכת.

마חר וצמד המנועים התחתוניים נתו עלומס משקלה הכלל של כל המערכת, נבחר במנועים בעלי מומנט גבוה יותר מאשר המנוע של גלגל התגובה. תפוקת המומנט של המנועים היא 166 סל"ד אשר מייצרים $1.7 \text{ kg} \cdot \text{cm}$, אשר מתורגמים ל $0.167 \text{ m} \cdot \text{N}$.

מצין כי אם במהלך הניסוי נלמד כי המנועים בעלי סל"ד של 166 יתקשו לבצע את המשימה, זאת אומרת שאינם יספקו את המומנט הנדרש כדי להגביל היטוב לתנודות המערכת ולעזר להגעה ליציבות או לכל הפחות להתקנות סביב זריזת המטרה, נחליף אותם במנועים עם סל"ד של 90, ותפוקת מומנט של $3.2 \text{ cm} \cdot \text{kg}$, המתורגמים ל $0.32 \text{ N} \cdot \text{m}$.

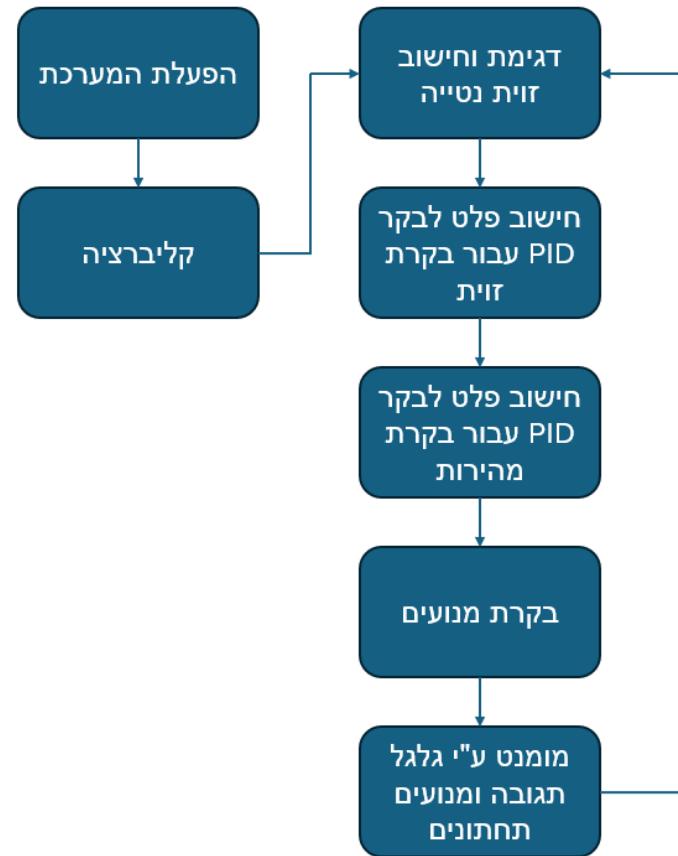


איור 28 – תיאור מערכת ניסוי 4 : גלגל תגובה יחד עם פעולות מנועים תחתוניים

אופן פעולות המערכת

המערכת מתחילה את פעילותה עם חיבורה למקור מתח (המחשב אליו מועברים ובו מנוטחים הנתונים). השלב הבא כולל קליברציה של זווית נטיית המערכת. שלב זה חשוב מאוד מאחר וטיטה גדולה מזוינת 0 מעלה תביא לחוסר איזון במערכת ואי יכלה להתיצב כלל.
לאחר חישוב זווית ההתחלה איתם תחילת המערכת לפעול, מתחילה פעולה המערכת באופן לולאטי:

- דגימת זווית הנטייה וחישובה על ידי פילטר המسان רעשים ותרום לדגימה איקוטית ומדוקת.
- חישוב פלט לבקר PID אשר מחשב את זווית הנטייה, על ידי הגדרה מראש של מקדמי הבקר. שלב זה של הגדרת מקדמי הבקר חשוב מאוד ויש לו השפעה ישירה על תגובת המערכת. בקר זה הוא הראשון מבין שניים ומשמש כລואה חיצונית. פלט בקרת הזווית נשלח ישירות להוות כאות הכניסה לבקרת המהירות אשר מגיעה בצעד הבא.
- חישוב פלט לבקר PID אשר מקבל בכניסה אתאות בקרת הזווית מהצעד הקודם ומחשב אתאות הבדיקה להפעלת המנועים.
- שליחתאות בקרת המהירות לבקרת המנועים. נרצה לתחום את גבולותאות אלו מאחר ונרצה לקבוע לטווח האותות אשר יפעיל מיידית את המנוע. בקרת המנועים שולטת במהירות הסיבוב, כיוון הסיבוב וחלוקת האותות בין מנוע גלגל התגובה לבין המנועים התחתונים.
- גלגל התגובה אשר מחובר למנוע, מסתובב בכיוון נגדו לכיוון נטיית המערכת. סיבוב זה מפעיל מומנט על המערכת סיבוב ציר הגלגלים התחתוניים ובכך מתנגד לנפילה. במקביל, המנועים התחתוניים גם הם מגיבים בהתאם לזוויות הנטייה אשר נוצרות. שילוב שלושת המנועים מפעיל מומנט כללי על המערכת.



איור 29 – תיאור אופן פעולה המערכת, ניסוי 4

תכנית ארדזואינו

התוכנית נכתבה והופעלת ב Arduino IDE וכוללת את השלבים הבאים :

- הגדרת והפעלת חישון התאוצה.
- הגדרת חיבורים ללוח ארדזואינו.
- שלב קליברציה (אתחול זוויות ההתחלה).
- חישוב זווית הנטיה.
- בקרת PID חיצונית – בקרת זווית הנטיה.
- בקרת PID פנימית – בקרת אותן המהירות למנועים.
- בקרת מנועים – חישוב אופן הפעלת המנועים.
- הדפסת פלט לצורך ניתוח ומעקב.

```
1 #include <Wire.h>           // Include Wire library for I2C communication
2 #include <MPU6050_light.h>   // Include MPU6050 library
3
4 MPU6050 mpu(Wire);         // Create an MPU6050 object using the Wire library
5
6 // Pin definitions for reaction wheel motor
7 const int PWM_PIN = 5;       // PWM pin for controlling speed
8 const int IN1_PIN = 7;        // Motor control pin 1 for direction
9 const int IN2_PIN = 6;        // Motor control pin 2 for direction
10
11 // Pin definitions for the bottom motors (left and right)
12 const int IN1_MOTOR1 = 8;     // Motor 1 control pin 1
13 const int IN2_MOTOR1 = 9;     // Motor 1 control pin 2
14 const int PWM_MOTOR1 = 10;    // PWM pin for Motor 1 speed control
15
16 const int IN1_MOTOR2 = 13;    // Motor 2 control pin 1
17 const int IN2_MOTOR2 = 12;    // Motor 2 control pin 2
18 const int PWM_MOTOR2 = 11;    // PWM pin for Motor 2 speed control
19
20 // PID controller variables (outer loop - tilt control)
21 float Kp_outer = 50.0;       // Proportional gain for outer loop
22 float Ki_outer = 0.0;        // Integral gain for outer loop
23 float Kd_outer = 0.0;        // Derivative gain for outer loop
24 float setpoint_angle = 0.0;   // Desired angle (setpoint)
25 float lastAngleError = 0.0;   // Previous angle error
26 float integral_angle = 0.0;  // Accumulated integral of angle error
27
28 // PID controller variables (inner loop - velocity control)
29 float Kp_inner = 80.0;       // Proportional gain for inner loop
30 float Ki_inner = 0.0;        // Integral gain for inner loop
31 float Kd_inner = 0.0;        // Derivative gain for inner loop
32 float lastVelocityError = 0.0; // Previous velocity error
33 float integral_velocity = 0.0; // Accumulated integral of velocity error
34
35 // Feedforward gain (not used in current code)
36 float feedforwardGain = 1.0;
37
38 // Motor control constants
39 const int MIN_SPEED = 150;    // Minimum PWM value to overcome motor's static friction
40 const int MAX_SPEED = 255;    // Maximum PWM value (max speed)
41
42 // Timing variables
43 unsigned long lastTime = 0;   // Last time control loop was executed
44 const unsigned long SAMPLE_TIME = 5; // Control loop sample time in milliseconds
45
46 void setup() {
47     Serial.begin(115200);      // Initialize serial communication at 115200 baud
```

אייר 30 – תכנית אודזיאנו 4 חלק 1

```
48  Wire.begin();           // Initialize I2C communication
49
50 // Initialize pins for reaction wheel motor
51 pinMode(PWM_PIN, OUTPUT);
52 pinMode(IN1_PIN, OUTPUT);
53 pinMode(IN2_PIN, OUTPUT);
54
55 // Initialize pins for the two bottom motors
56 pinMode(IN1_MOTOR1, OUTPUT);
57 pinMode(IN2_MOTOR1, OUTPUT);
58 pinMode(PWM_MOTOR1, OUTPUT);
59
60 pinMode(IN1_MOTOR2, OUTPUT);
61 pinMode(IN2_MOTOR2, OUTPUT);
62 pinMode(PWM_MOTOR2, OUTPUT);
63
64 // Initialize MPU6050 sensor
65 byte status = mpu.begin();
66 while (status != 0) {          // Check if MPU6050 is connected properly
67   Serial.println("MPU6050 initialization failed. Please check your connections.");
68   delay(1000);                // Wait 1 second before retrying
69 }
70
71 Serial.println("MPU6050 initialized successfully!");
72
73 // Calibrate MPU6050 sensor
74 Serial.println("Calibrating MPU6050...");
75 mpu.calcOffsets();           // Calculate offsets to improve accuracy
76 Serial.println("Calibration complete!");
77 }
78
79 void loop() {
80   unsigned long currentTime = millis(); // Get current time in milliseconds
81
82   // Execute control loop at specified sample time intervals
83   if (currentTime - lastTime >= SAMPLE_TIME) {
84     float deltaTime = (float)(currentTime - lastTime) / 1000.0; // Calculate elapsed time in seconds
85
86     // Update sensor readings and compute current angle
87     float angle = updateMPU(deltaTime);
88
89     // Outer PID loop to compute desired velocity based on angle error
90     float desiredVelocity = calculateOuterPID(angle, deltaTime);
91
92     // Inner PID loop to compute motor output for bottom motors based on desired velocity
93     float bottomMotorOutput = calculateInnerPID(desiredVelocity, deltaTime);
```

איור 31 – תכנית ארדואינו 4 חלק 2

```
95 // Compute reaction wheel output (could be modified separately)
96 float reactionWheelOutput = desiredVelocity;
97
98 // Set motor speeds for bottom motors and reaction wheel
99 setMotorSpeed(bottomMotorOutput, reactionWheelOutput);
100
101 Serial.println(angle);
102
103 // Update lastTime for next iteration
104 lastTime = currentTime;
105 }
106 }
107
108 // Helper function to update MPU6050 readings and compute angle
109 float updateMPU(float deltaTime) {
110     mpu.update(); // Update sensor data
111
112     // Complementary filter to combine accelerometer and gyroscope data
113     // Currently set to use only accelerometer data (adjust weights as needed)
114     return 0.0 * (mpu.getAngleX() + mpu.getGyroX() * deltaTime) + 1.0 * mpu.getAngleX();
115 }
116
117 // Calculate outer PID control (tilt control)
118 float calculateOuterPID(float angle, float deltaTime) {
119     float angleError = setpoint_angle - angle; // Compute angle error
120     integral_angle += angleError * deltaTime; // Update integral term
121     float derivative_angle = (angleError - lastAngleError) / deltaTime; // Compute derivative term
122
123     // Compute PID output
124     float output = Kp_outer * angleError + Ki_outer * integral_angle + Kd_outer * derivative_angle;
125
126     lastAngleError = angleError; // Update last angle error for next derivative calculation
127     return output; // Return desired velocity for inner loop
128 }
129
130 // Calculate inner PID control (velocity control)
131 float calculateInnerPID(float desiredVelocity, float deltaTime) {
132     // Assume actual velocity is zero (could be modified to include velocity measurements)
133     float velocityError = desiredVelocity; // Compute velocity error
134     integral_velocity += velocityError * deltaTime; // Update integral term
135     float derivative_velocity = (velocityError - lastVelocityError) / deltaTime; // Compute derivative term
136
137     // Compute PID output
138     float output = Kp_inner * velocityError + Ki_inner * integral_velocity + Kd_inner * derivative_velocity;
139
140     lastVelocityError = velocityError; // Update last velocity error for next derivative calculation
141     return output; // Return motor output for bottom motors
```

איור – 32 – תכנית ארדואינו 4 חלק 3

```
142 }
143
144 // Set motor speeds based on PID output for both motor types
145 void setMotorSpeed(float bottomOutput, float reactionOutput) {
146     // Map the PID output to PWM values within the motor speed range
147     int bottomMotorSpeed = map(abs(bottomOutput), 0, 180, MIN_SPEED, MAX_SPEED);
148     bottomMotorSpeed = constrain(bottomMotorSpeed, MIN_SPEED, MAX_SPEED);
149
150     int reactionWheelSpeed = map(abs(reactionOutput), 0, 180, MIN_SPEED, MAX_SPEED);
151     reactionWheelSpeed = constrain(reactionWheelSpeed, MIN_SPEED, MAX_SPEED);
152
153     // Control bottom motors (left and right)
154     if (bottomOutput > 0) {
155         // Set direction for forward motion
156         digitalWrite(IN1_MOTOR1, LOW);
157         digitalWrite(IN2_MOTOR1, HIGH);
158         digitalWrite(IN1_MOTOR2, LOW);
159         digitalWrite(IN2_MOTOR2, HIGH);
160     } else {
161         // Set direction for reverse motion
162         digitalWrite(IN1_MOTOR1, HIGH);
163         digitalWrite(IN2_MOTOR1, LOW);
164         digitalWrite(IN1_MOTOR2, HIGH);
165         digitalWrite(IN2_MOTOR2, LOW);
166     }
167     // Set PWM speed for bottom motors
168     analogWrite(PWM_MOTOR1, bottomMotorSpeed);
169     analogWrite(PWM_MOTOR2, bottomMotorSpeed);
170
171     // Control reaction wheel motor
172     if (reactionOutput > 0) {
173         // Set direction for reaction wheel rotation
174         digitalWrite(IN1_PIN, HIGH);
175         digitalWrite(IN2_PIN, LOW);
176     } else {
177         digitalWrite(IN1_PIN, LOW);
178         digitalWrite(IN2_PIN, HIGH);
179     }
180     // Set PWM speed for reaction wheel motor
181     analogWrite(PWM_PIN, reactionWheelSpeed);
182 }
183
```

איור 33 – תכנית ארדואינו 4 חלק 4

תוצאת הניסוי

ביצעונו 3 איטרציות לניסוי זה, כאשר כפי שניתן לראות בכל אחד משלשות הגרפים המצורפים מטה יש התנהגות מיטביתה של המערכת בכך בכל פעם יש התוכניות לזוויות המטרה.

בגרף הראשון (איור 5) אנו לומדים כי המערכת תונדת באופן יציב סביב זווית המטרה של 0 מעלות. ישנה קפיצה באמצעות הגרף אשר מוששת את ההבנה מניסוי מספר 1, והוא שאין למערכת יכולה להתאושש מזוויות הגדיות $55 \pm$ מעלות. במקרה זה אנו מגבלים את נתיות המערכת עד לערך של 8-9 מעלות. לאחר שהחזרנו את המערכת בקורס ידנית לטוחה יכולתה ($55 \pm$ מעלות), היא ממשיכה לתנוד בקורס יציבה סביב זווית 0 מעלות. את האיטרציה הראשונה עשינו עם מנועי 166 סל"ד.

בגרף השני (איור 6) החלפנו את המנועים התחתוניים במנועי 90 סל"ד. ניתן לראות כי ישנו שיפור בהתנהגות המערכת מאחר והתנודות קטנו לאורך כל זמן הפעולה ולמרות שעדיין קיימת קפיצה אחת שבה המערכת לא מצליחה להתאושש בקורס עצמאית, זמן החזרה לתנודה סביב 0 מעלות משתפר.

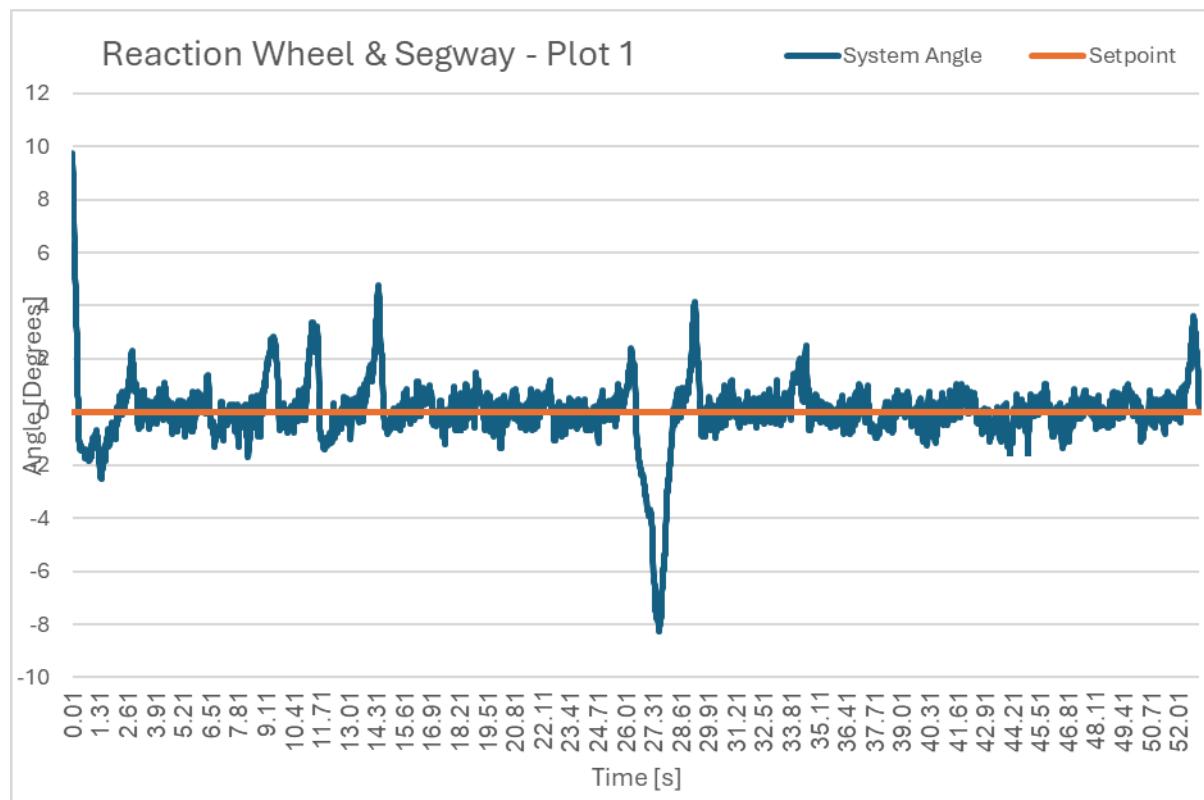
וחולט כי מנועי 90 סל"ד מספקים תמורה טובה יותר למערכת ולכן מעטה ואילך נמשיך איתם.

לצורך הודהות, הוחלט לבצע איטרציה נוספת, שלישית במספר זו זאת כדי לאשר כי אכן המערכת מתנהגת כמצופה. ניתן לראות בגרף השלישי (איור 7) כי המערכת מתכנסת בקורס פחות תונדת לזוויות המטרה ומצליחה לשומר על היציבות לאורך כל זמן הפעולה. גם באיטרציה זו קיימת קפיצה בהתנהגות המערכת וגם כאן החזרנו בקורס ידנית את המערכת לטוחה הזווית בהם היא יכולה להתאושש.

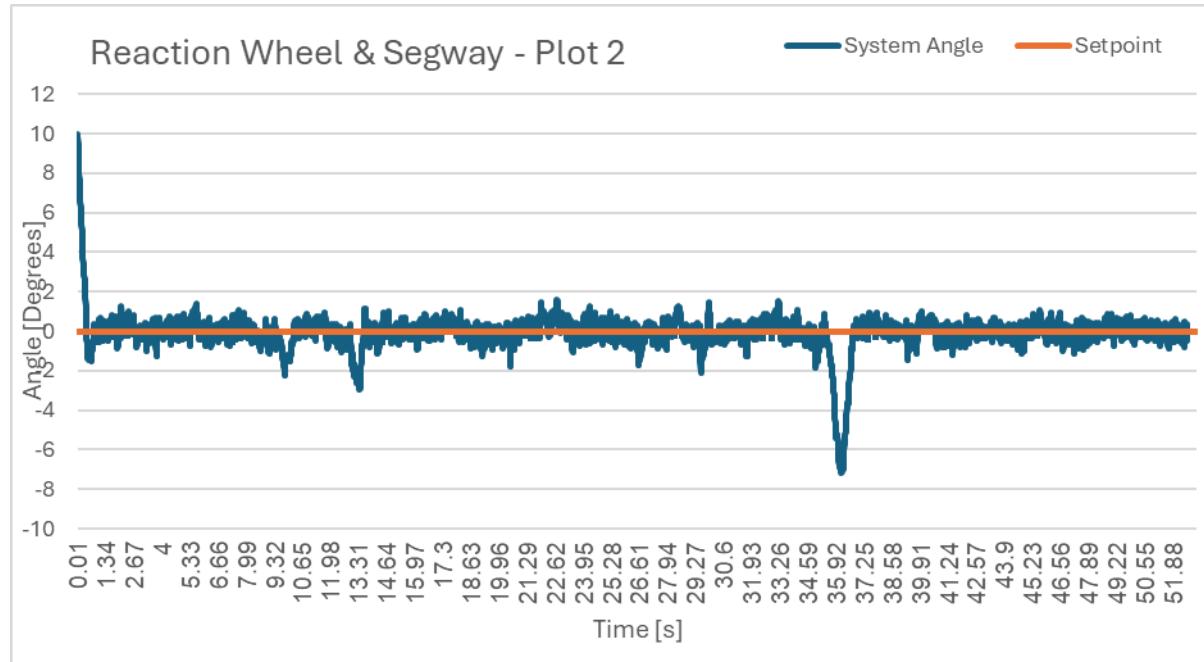
ניתן להגיד כי מטרת הניסוי הושגה בהצלחה ונתנה לנו מידע חשוב לגבי אופן פעולה המערכת כאשר משולבת בין גלגל התגובה למנועים התחתוניים.

בשלשות האיטרציות נתקלנו לפחות פעמיים אחד בזמן הפעלה בקפיצה לזוויות נתיות המערכת. הגורם המרכזי אליו אפשר ליחס קפיצה זו היא לפשטות המנוע אשר שולט בגלגל התגובה. מאחר וגלגל התגובה נמצא בחלק העליון של המערכת, נדרש ממנו לא שילוב של מהירות סיבוב גבוהה יחד עם מומנט מספק. מאחר ואנו דבקים בהחלה להמשיך את בניית הפרויקט באמצעות נגשים וזולים, נctrף לקבל כי קפיצה זו נדרשת ואני ניתנת למחיקה. מניסוי זה ניתן להסיק כי מאחר ונצליחים להתכנס לזוויות המטרה של 0 מעלות ושילוב גלגל התגובה יחד עם המנועים התחתוניים מהויה נקודת טוביה בה ניתן להמשיך להמשיך לשלב הבא של הפרויקט והוא הוספה של רובד עיבוד התמונה.

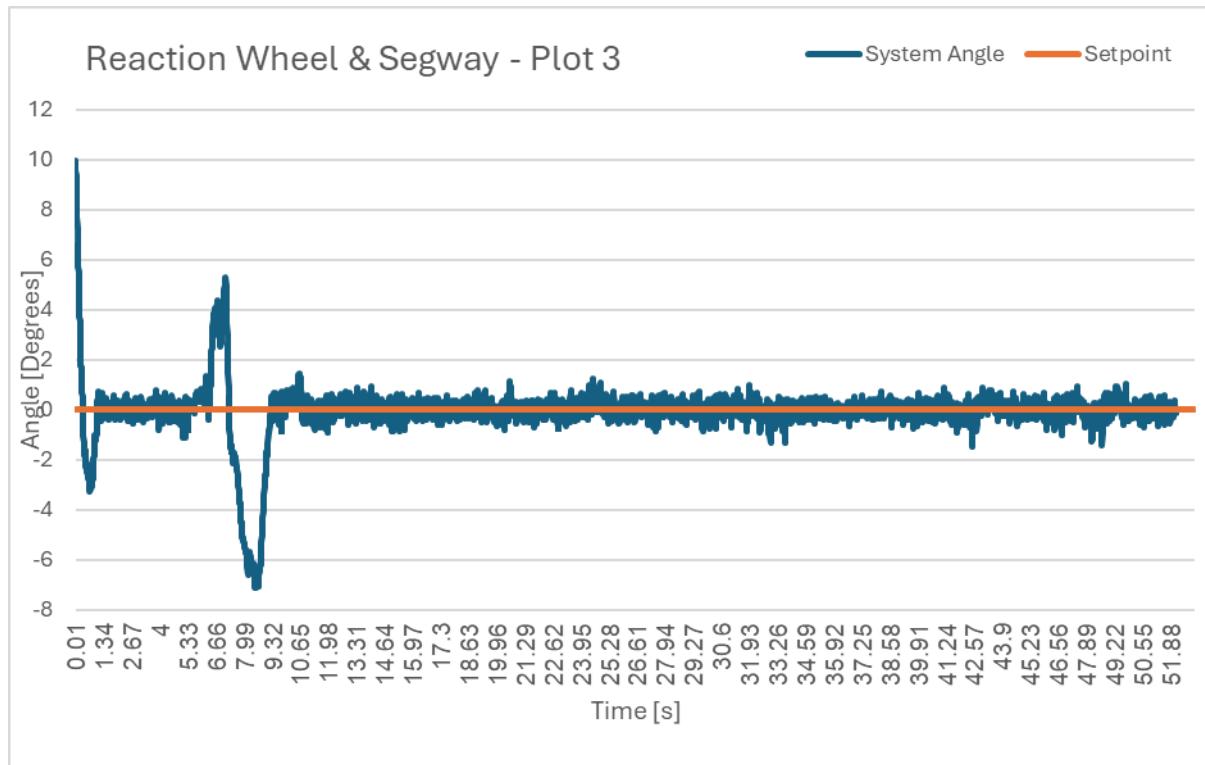
לפני שנמשיך לעיבוד התמונה, אנו רוצים לבצע ניסוי נוסף ובו נכבה את גלגל התגובה ונפעיל רק את המנועים התחתוניים. חשוב לנו להבין את המערכת בקורס מיטבית ולודא כי המערכת מצליחה להתאים היפך.



איור 34 – גרפ' זווית-זמן, ניסוי 4.1



איור 35 – גרפ' זווית-זמן, ניסוי 4.2



איור 35 - גרפ' זווית-זמן, ניסוי 4.3

דיון בתוצאות

1. ניסוי 4 מראה התקדמות משמעותית בפרויקט, שכן הוא בוחן את שילוב גלגל התגובה עם המנועים התחטוניים לצורך יצוב המערכת. בניגוד לניסוי הקודם, שבו שילוב גלגל התגובה עם המסה המרסטנת לא הביא לתוצאות הרצויה, הפעם הציפייה הייתה לשיפור התקנות לזרזת המטרה. מטרת הניסוי הייתה לבדוק אם שילוב זה יכול לשמש כבסיס להתקדמות לקראת המטרה הסופית של הפרויקט - בניית רובוט דו-גלגלי המתאים באמצעות גלגל תגובה ומסוגל להתקדם תוך שימוש בעיבוד תמונה.

2. המערכת בניסוי זה הتبססה על הניסוי הראשון, אך עם שינויים משמעותיים - חיבור המנועים התחטוניים למתח חשמלי. בחירת המנועים התחטוניים הייתה מרכיבת, כאשר נבחרו תחילת מנועים בעלי 166 סל"ד עם מומנט של $0.167 \text{ m}^* \text{N}$. עם זאת, החוקרים היו מוכנים להחליף אותם במנועים בעלי 90 סל"ד ומומנט של $0.32 \text{ m}^* \text{N}$ אם יתברר שהראשונים אינם מספקים. שילוב זה של מנועים עם גלגל התגובה נועד ליצור מערכת יציבה יותר, המסוגלת להגיב טוב יותר לתנודות ולהתכנס לזרזת המטרה.

3. תחילה הפעולה של המערכת כלל מספר שלבים מורכבים, החל מקליברציה של זווית הנטייה ועד לשימוש בשני בקרים PID - אחד לביקורת זווית הנטייה והשני לביקורת מהירות המנועים. המערכת פעל באופן לולאטי, כאשר בכל מחזור נדגמה זווית הנטייה, חושבו אותן הbakra, ונשלחו פקודות למנועים. גלגל התגובה פעל בכיוון הנגדי לנטיית המערכת, בעוד המנועים התחטוניים הגיבו גם הם לשינויים בזווית. שילוב זה של שלושת המנועים נועד ליצור מומנט כולל שיאן את המערכת.

4. תוצאות הניסוי היו מעודדות מאוד. בשלוש האיטרציות שבוצעו, המערכת הצליחה להתכנס לזוויות המטרה של 0 מעלות. באיטרציה הראשונה, עם מנועי 166 סל"ד, נצפתה תנודה יציבה סביבה זוויות המטרה, למעט קפיצה אחת שחרגה מוגבלות היכולת של המערכת. באיטרציה השנייה, עם החלפה למנועי 90 סל"ד, נראה שיפור נוסף עם תנודות קטנות יותר. האיטרציה השלישית אישרה את היציבות המשופרת של המערכת. למרות שבכל האיטרציות נצפתה לפחות קפיצה אחת בזווית הנטייה, המערכת הצלicha להתאושש ולהזור לתנודה יציבה סביבה זוויות המטרה.

5. המשקנות מניסוי זה היו חיוביות ומשמעותיות להמשך הפרויקט. השימוש של גלגל התגובה עם המנועים התchaptonim הוכיח את עצמו כפתרון יעיל לייצוב המערכת. למרות הקפיצות בזווית הנטייה, שיוחסו לפשטוות המנוע של גלגל התגובה, החוקרים החליטו לקבל מגבלה זו כחלק מהמחויבות לשימוש ברכיבים זולים ונגישים. הצלחה בהתקנסות לזוויות המטרה של 0 מעלות סימנה נקודת מפתח חשובה, המאפשרת התקדמותו של הפרויקט הבא של הפרויקט - שימוש עיבוד תמונה. עם זאת, לפני המעבר לשלב זה, הוחלט לבצע ניסוי נוסף ללא גלגל התגובה, כדי להבין לעומק את תרומתו הייחודית ליציבות המערכת.

5.5 ניסוי 5 – בוחינת יציבות המערכת בהפעלת מנועים תחטוניים בלבד.

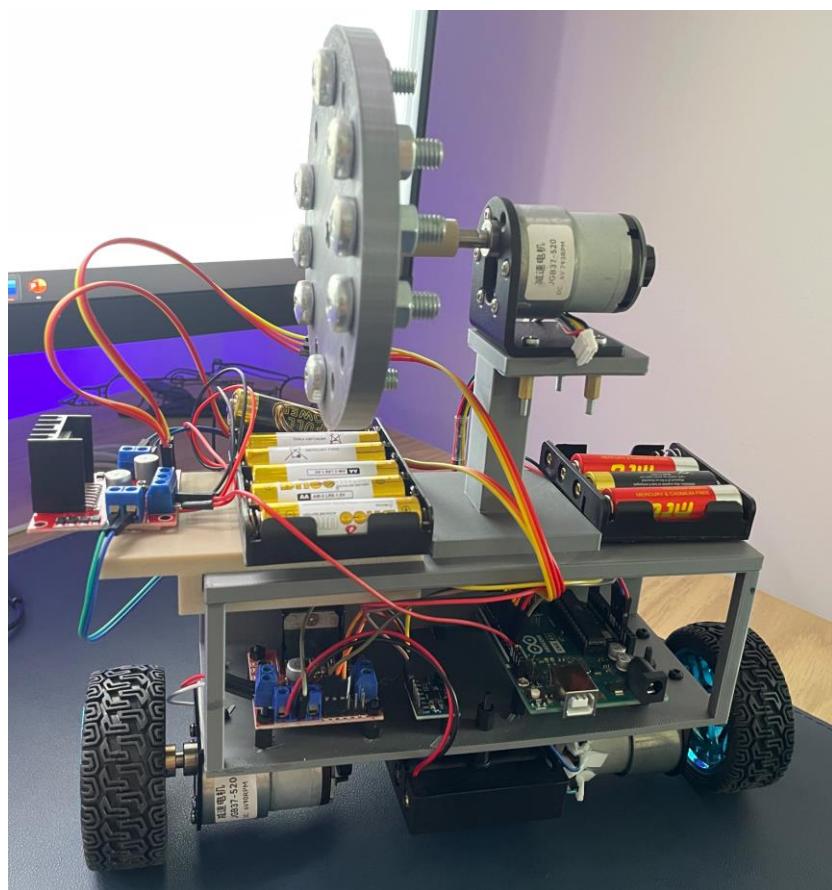
מטרה

בניסוי הקודם, ניסוי מס' 4, שילבנו במערכת את תתי-המערכות גלגל תגובה ומנועים תחטוניים כדי להשיג יציבות למערכת תוך התכנסות לזוויות המטריה שהינה 0 מעלות. ניסוי זה הוגדר כהצלחה מאחר והצלחנו לייצב את המערכת.

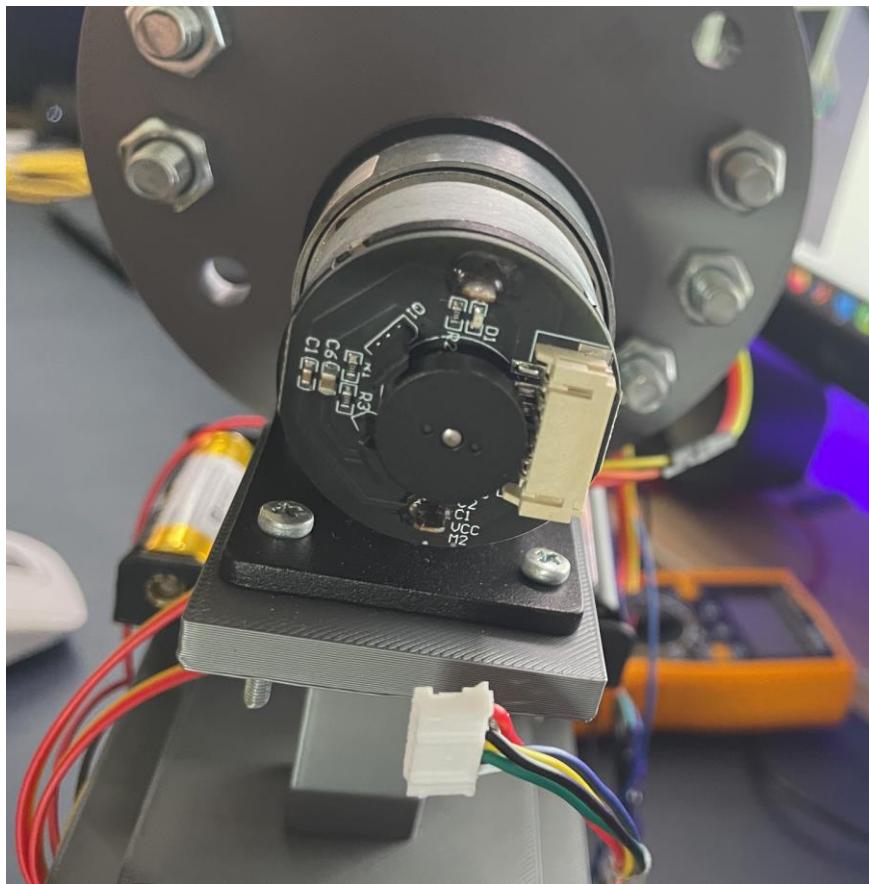
בניסוי זה אני רוצח לבחון את המערכת תחת השפעת המנועים התחטוניים בלבד ולהוריד את השפעת גלגל התגובה. מאחר ובניסוי קודם הגיעו לזוויות המטריה תוך קפיצות בזווית הנטייה, נרצה להבין, כמה אם בכלל, גלגל התגובה משפיע על המערכת מבחן התכנסות לזוויות המטריה ותרומה לשיכוך הקפיצות בזווית הנטייה.

תיאור המערכת בניסוי

המערכת מתבססת על המשך ישיר למערכת מניסוי 4, עם הבדל אחדמשמעותי והוא ניתוק מנוע גלגל התגובה. מעבר לניתוק המנוע, כל הרכיבים נשארים במקומות כולל סוללות ודראיבר מנוע. חשוב כי נשמר על פיזור המשקל של המערכת באופן זהה למערכת מהניסוי הקודם כדי שהמסקנות והתוצאות אליהם הגיעו בסוף הניסוי יהו מידיע מהימן אשר יהיה אפשר להשתמש בו בהמשך הפרויקט והמשך בניית המערכת.



איור 36 – תיאור מערכת ניסוי 5 : מערכת כל בסיס גלגליים תחטוניים וגלגל תגובה מנותק



איור 37 – תיאור מערכת ניסוי 5 : תיעוד גלגל תגובה מנוטק

אופן פועלות המערכת

אופן פועלות המערכת בניסוי זה זהה לאופן פועלות המערכת בניסוי הקודם, למעט העובדה שגלגל התגובה אינו משתתף בפעולת מכיוון שהוא מנוטק. ישנו שינוי באופן בקרת המנועים מאחר ואות הבדיקה מועבר רק למנועים התחטוניים ללא התניות של מנוע גלגל התגובה.

המערכת מתחילה את פעילותה עם חיבורה למקור מתח (המחשב אליו מועברים ובו מנוטחים הנתונים). השלב הבא כולל קליברציה של זווית נטיית המערכת. שלב זה חשוב מאוד מאחר וסטייה גדולה מזוויות 0 מעלה תביא לחוסר איזון במערכת ואי יכולת להתיצב כלל.

לאחר חישוב זווית ההתחלה אitem תחילת המערכת לפועל, מתחילה פועלות המערכת באופן לולאטי:

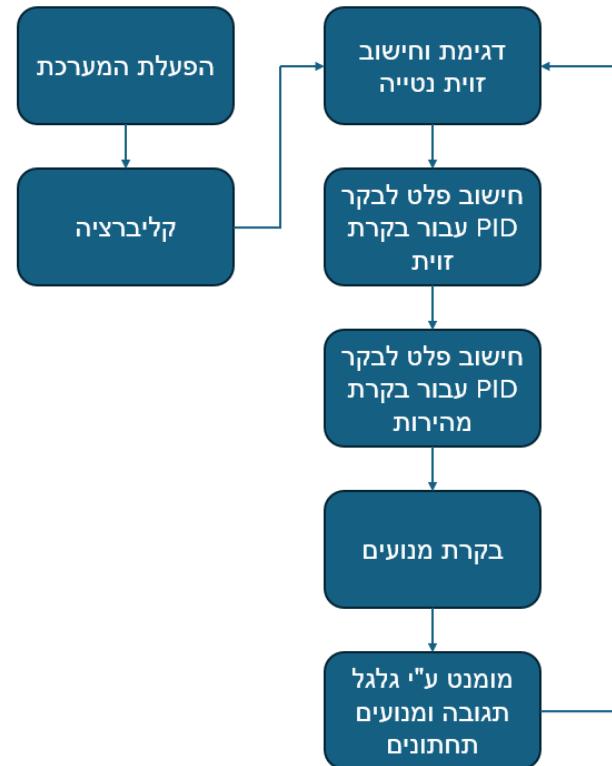
- דגימות זווית הנטייה וחישובה על ידי פילטר המשנן רעשים ותורם לדגימה איקוונית ומדויקת.

- חישוב פלט לבקר PID אשר מחשב את זווית הנטייה, על ידי הגדרה מראש של מקדמי הבקר. שלב זה של הגדרת מקדמי הבקר חשוב מאוד אחריו יש לו השפעה ישירה על תגובת המערכת. בקר זה הוא הראשון מבין שתים ומשמש כלולאה חיצונית. פלט בקרת הזווית נשלח ישירות להוות כאות הכניסה לבקרת המהירות אשר מגיעה בצד הבא.

- חישוב פלט לבקר PID אשר מקבל בכניסה את אותן בקרת הזווית מהצעד הקודם ומהчисב את אותן הבדיקה להפעלת המנועים.

- שליחת אותן בקרת המהירות לבקרת המנועים. נרצה לתchrom את גבולות אותן אלו מאחר ונרצה לקלוע לטוחה אותן אשר יפעיל מיידית את המנוע. בקרת המנועים שולטת במהירות הסיבוב, כיון הסיבוב של המנועים התחטוניים.

- המנועים התחטוניים מגיבים בהתאם לאות הבדיקה המחשב בהתאם לזרויות הנטייה אשר נוצרות ובכך מפעילים מומנט על המערכת כדי לגרום לה לרוץ את התנודות ולהתכנס לזריות המטרה.



איור 38 – תיאור אופן פעולה מערכת הניסוי

תכנית ארדואינו

התוכנית נכתבה והופעלת ב Arduino IDE וכוללת את השלבים הבאים :

- הגדרת והפעלת חישון התאוצה.
- הגדרת חיבורים ללוח ארדואינו.
- שלב קליברציה (אתחול זוויות התחליה).
- חישוב זווית הנתניה.
- בקרת PID חיצונית – בקרת זווית הנתניה.
- בקרת PID פנימית – בקרת אותן המהירות למנועים.
- בקרת מנועים – חישוב אופן הפעלת המנועים התחטוניים.
- הדפסת פלט לצורך ניתוח ומעקב.

```
1 #include <Wire.h>           // Include Wire library for I2C communication
2 #include <MPU6050_light.h>   // Include MPU6050 library
3
4 MPU6050 mpu(Wire);         // Create an MPU6050 object using the Wire library
5
6 // Pin definitions for the bottom motors (left and right)
7 const int IN1_MOTOR1 = 8;    // Motor 1 control pin 1
8 const int IN2_MOTOR1 = 9;    // Motor 1 control pin 2
9 const int PWM_MOTOR1 = 10;   // PWM pin for Motor 1 speed control
10
11 const int IN1_MOTOR2 = 13;   // Motor 2 control pin 1
12 const int IN2_MOTOR2 = 12;   // Motor 2 control pin 2
13 const int PWM_MOTOR2 = 11;   // PWM pin for Motor 2 speed control
14
15 // PID controller variables (outer loop - tilt control)
16 float Kp_outer = 40.0;      // Proportional gain for outer loop
17 float Ki_outer = 0.0;       // Integral gain for outer loop
18 float Kd_outer = 0.0;       // Derivative gain for outer loop
19 float setpoint_angle = 0.0; // Desired angle (setpoint)
20 float lastAngleError = 0.0; // Previous angle error
21 float integral_angle = 0.0; // Accumulated integral of angle error
22
23 // PID controller variables (inner loop - velocity control)
24 float Kp_inner = 80.0;      // Proportional gain for inner loop
25 float Ki_inner = 0.0;       // Integral gain for inner loop
26 float Kd_inner = 0.0;       // Derivative gain for inner loop
27 float lastVelocityError = 0.0; // Previous velocity error
28 float integral_velocity = 0.0; // Accumulated integral of velocity error
29
30 // Motor control constants
31 const int MIN_SPEED = 150;  // Minimum PWM value to overcome motor's static friction
32 const int MAX_SPEED = 255;  // Maximum PWM value (max speed)
33
34 // Timing variables
35 unsigned long lastTime = 0; // Last time control loop was executed
36 const unsigned long SAMPLE_TIME = 5; // Control loop sample time in milliseconds
37
38 void setup() {
39     Serial.begin(115200);      // Initialize serial communication at 115200 baud
40     Wire.begin();             // Initialize I2C communication
41
42     // Initialize pins for the two bottom motors
43     pinMode(IN1_MOTOR1, OUTPUT);
44     pinMode(IN2_MOTOR1, OUTPUT);
45     pinMode(PWM_MOTOR1, OUTPUT);
46
47     pinMode(IN1_MOTOR2, OUTPUT);
```

איור 39 – תכנית ארדואינו 5 חלק 1

```
48  pinMode(IN2_MOTOR2, OUTPUT);
49  pinMode(PWM_MOTOR2, OUTPUT);
50
51 // Initialize MPU6050 sensor
52 byte status = mpu.begin();
53 while (status != 0) {          // Check if MPU6050 is connected properly
54   Serial.println("MPU6050 initialization failed. Please check your connections.");
55   delay(1000);                // Wait 1 second before retrying
56 }
57
58 Serial.println("MPU6050 initialized successfully!");
59
60 // Calibrate MPU6050 sensor
61 Serial.println("Calibrating MPU6050...");
62 mpu.calibrate();              // Calculate offsets to improve accuracy
63 Serial.println("Calibration complete!");
64 }
65
66 void loop() {
67   unsigned long currentTime = millis(); // Get current time in milliseconds
68
69   // Execute control loop at specified sample time intervals
70   if (currentTime - lastTime >= SAMPLE_TIME) {
71     float deltaTime = (float)(currentTime - lastTime) / 1000.0; // Calculate elapsed time in seconds
72
73     // Update sensor readings and compute current angle
74     float angle = updateMPU(deltaTime);
75
76     // Outer PID loop to compute desired velocity based on angle error
77     float desiredVelocity = calculateOuterPID(angle, deltaTime);
78
79     // Inner PID loop to compute motor output for bottom motors based on desired velocity
80     float bottomMotorOutput = calculateInnerPID(desiredVelocity, deltaTime);
81
82     // Set motor speeds for bottom motors
83     setMotorSpeed(bottomMotorOutput);
84
85     Serial.println(angle);
86
87     // Update lastTime for next iteration
88     lastTime = currentTime;
89   }
90 }
91
92 // Helper function to update MPU6050 readings and compute angle
93 float updateMPU(float deltaTime) {
94   mpu.update(); // Update sensor data
```

איור 40 - תכנית ארדואינו 5 חלק 2

```

96     // Complementary filter to combine accelerometer and gyroscope data
97     // Currently set to use only accelerometer data (adjust weights as needed)
98     return 0.0 * (mpu.getAngleX() + mpu.getGyroX() * deltaTime) + 1.0 * mpu.getAngleX();
99 }
100
101 // Calculate outer PID control (tilt control)
102 float calculateOuterPID(float angle, float deltaTime) {
103     float angleError = setpoint_angle - angle; // Compute angle error
104     integral_angle += angleError * deltaTime; // Update integral term
105     float derivative_angle = (angleError - lastAngleError) / deltaTime; // Compute derivative term
106
107     // Compute PID output
108     float output = Kp_outer * angleError + Ki_outer * integral_angle + Kd_outer * derivative_angle;
109
110     lastAngleError = angleError; // Update last angle error for next derivative calculation
111     return output; // Return desired velocity for inner loop
112 }
113
114 // Calculate inner PID control (velocity control)
115 float calculateInnerPID(float desiredVelocity, float deltaTime) {
116     // Assume actual velocity is zero (could be modified to include velocity measurements)
117     float velocityError = desiredVelocity; // Compute velocity error
118     integral_velocity += velocityError * deltaTime; // Update integral term
119     float derivative_velocity = (velocityError - lastVelocityError) / deltaTime; // Compute derivative term
120
121     // Compute PID output
122     float output = Kp_inner * velocityError + Ki_inner * integral_velocity + Kd_inner * derivative_velocity;
123
124     lastVelocityError = velocityError; // Update last velocity error for next derivative calculation
125     return output; // Return motor output for bottom motors
126 }
127
128 // Set motor speeds based on PID output for both bottom motors
129 void setMotorSpeed(float bottomOutput) {
130     // Map the PID output to PWM values within the motor speed range
131     int bottomMotorSpeed = map(abs(bottomOutput), 0, 180, MIN_SPEED, MAX_SPEED);
132     bottomMotorSpeed = constrain(bottomMotorSpeed, MIN_SPEED, MAX_SPEED);
133
134     // Control bottom motors (left and right)
135     if (bottomOutput > 0) {
136         // Set direction for forward motion
137         digitalWrite(IN1_MOTOR1, LOW);
138         digitalWrite(IN2_MOTOR1, HIGH);
139         digitalWrite(IN1_MOTOR2, LOW);
140         digitalWrite(IN2_MOTOR2, HIGH);
141     } else {

```

איור 41 – תכנית ארדואינו 5 חלק 3

```

142     // Set direction for reverse motion
143     digitalWrite(IN1_MOTOR1, HIGH);
144     digitalWrite(IN2_MOTOR1, LOW);
145     digitalWrite(IN1_MOTOR2, HIGH);
146     digitalWrite(IN2_MOTOR2, LOW);
147 }
148 // Set PWM speed for bottom motors
149 analogWrite(PWM_MOTOR1, bottomMotorSpeed);
150 analogWrite(PWM_MOTOR2, bottomMotorSpeed);
151 }

```

איור 42 – תכנית ארדואינו 5 חלק 4

תוצאת הניסוי

בניסוי זה בוצעו 3 איטרציות עם תנאים זהים. ניתן להבחין כי תוצאות הגרפים די זהים אחד לשני ולבן מהימנים. אפשר לראות כי המערכת בכל איטרציה מצליחה להתכנס לזוויות המטרה ולשמור על יציבות המערכת.

בגרף הראשון (איור 5) אשר הינו האיטרציה הראשונה לניסוי, רואים כי המערכת מתנהגת באופן דומה למערכת מהניסוי הקודם, ומצליחה תוך זמן כמעט מיידי להתכנס לזוויות המטרה ובכך שומרת על יציבותה. כפי שהסבירנו בניסוי הקודם, קפיצות אלה הם בעצם יציאת המערכת מגבולות יכולת התפקיד שלה. הגדרנו בתחילת רצף הניסויים כי אנו נשמר על גבולות נטייה של 5 ± 5 מעלות. כל חריגה מטוחה מגבולות זה ניאלה להתרעב בפעולות המערכת ולווזר לה באופן ידני לחזור לטוחה הגבולות המקוריים.

בניגוד לניסוי הקודם, אשר כלל קפיצה אחת בלבד, ניתן להבחין כי ישנו 3 קפיצות בזוויות נטיית המערכת. בגלל שזויה איטרציה אחת מпоз' שלוש, עדין לא ניתן לקבוע באופן חד משמעי כי גידול מספר הקפיצות נובע מニアתוק גלגל התגובה.

בגרף השני (איור 6) אנו רואים את פעולות המערכת באיטרציה השנייה לניסוי. כמו באיטרציה הראשונה, גם כאן ניתן להבחין בבירור כי ישנו 3 קפיצות בזוויות נטיית המערכת וגם כאן ישנה התכנסות לזוויות המטרה ולהגעה ליציבות כללית של המערכת. חשוב להציג כי לא בוצע שום שינוי באופן פועלת המערכת או בהרכבה, لكن יש לנו סיבה טובה להאמין כי קפיצות אלו בגרף נובעות מニアתוק גלגל התגובה.

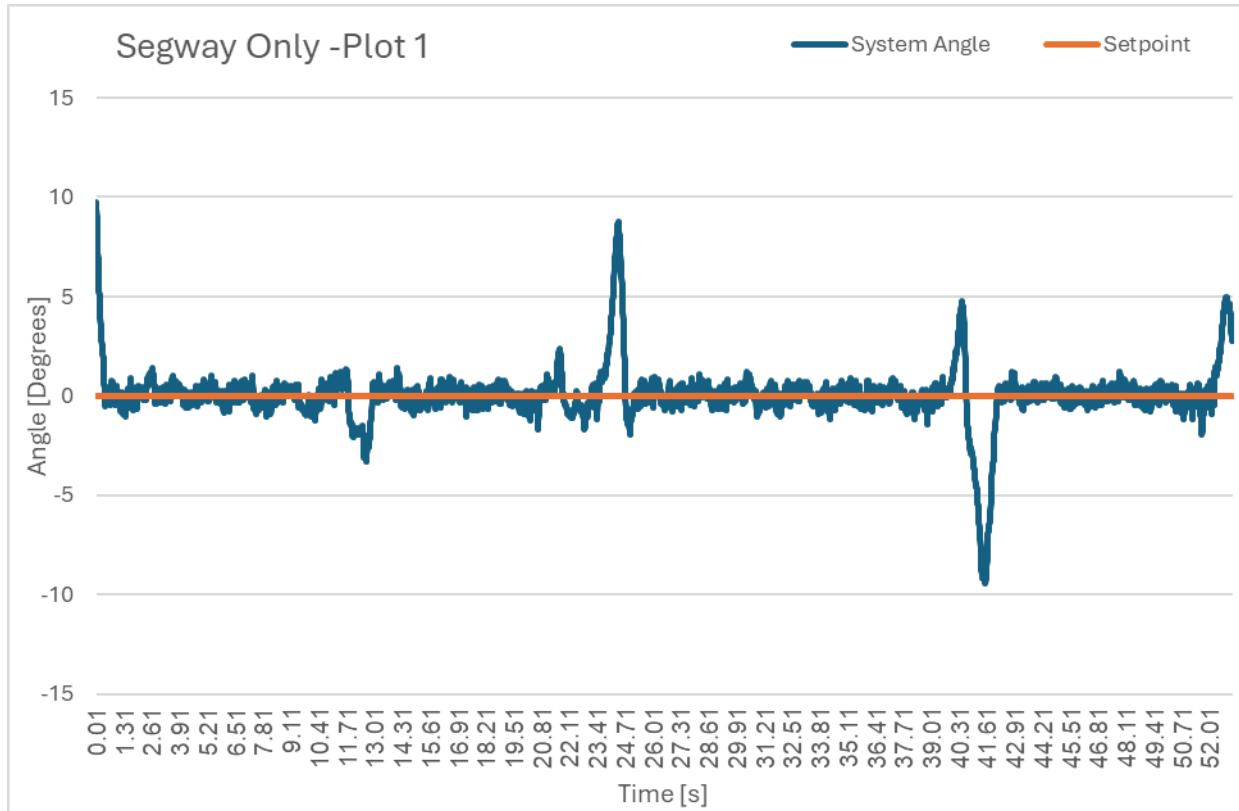
כדי לקבל אינדיקציה חד משמעית, ביצעו איטרציה נוספת ובחנו אותה.

בגרף השלישי (איור 7) מתוארת המערכת בעת ביצוע האיטרציה השלישית לניסוי. בדומה לאופן בו התנהגה המערכת בצד האיטרציות הקודם, גם כאן ישנה התכנסות לזוויות המטרה ושמירה על יציבות במהלך פעולה המערכת. דבר נוסף אשר זהה לאיטרציות הקודמות הוא שימוש קפיצות בזוויות הנטייה במספר דומה. ניתן להגיד באופן חד משמעי כי המערכת כאשר מופעלת בעזרת המנועים התחטוניים בלבד, מביאה תוצאה אשר מתאימה למטרת הניסוי. המערכת מצליחה להתכנס לזוויות המטרה של 0 מעלות ומצליחה לשמור על היציבות שלה.

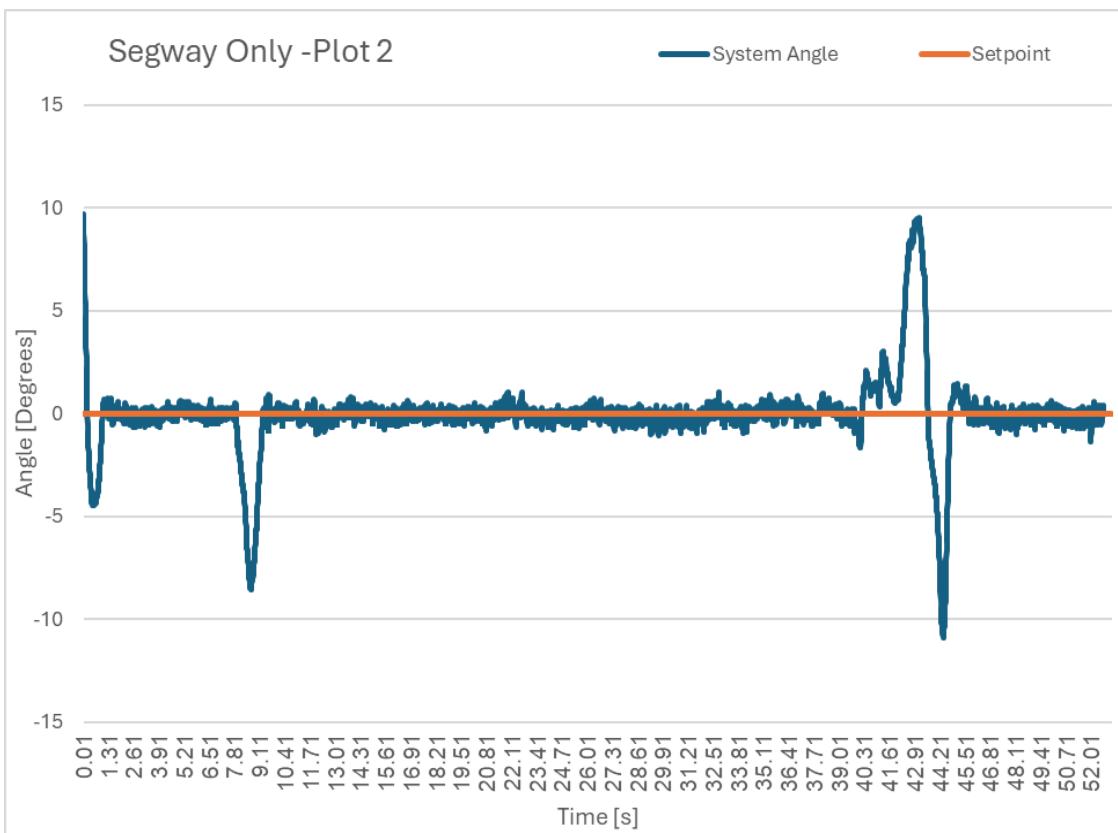
דבר נוסף ומשמעותי אשר נלמד בניסוי זה, הוא שלニアתוק גלגל התגובה ישנה חשיבות על ביצוע המערכת. רואים בבירור כי כאשר גלגל התגובה אינו פועל, המערכת מצליחה לצאת מיציבות יותר פעמים מאשר בזמן שגלגל התגובה פועל וועוזר ביצוב המערכת.

לפעולת המנועים התחטוניים יש חשיבות רבה בהשגת תוצאה רצואה בהינתן התנאים והמגבליות שהטלו על המערכת. לאחר ואנו מנסים לפטור את סוגיות יציבות המערכת עם רכיבים נגשים וזולים, אנו מבינים כי יש הכרח לחבר ושימוש במנועים התחטוניים.

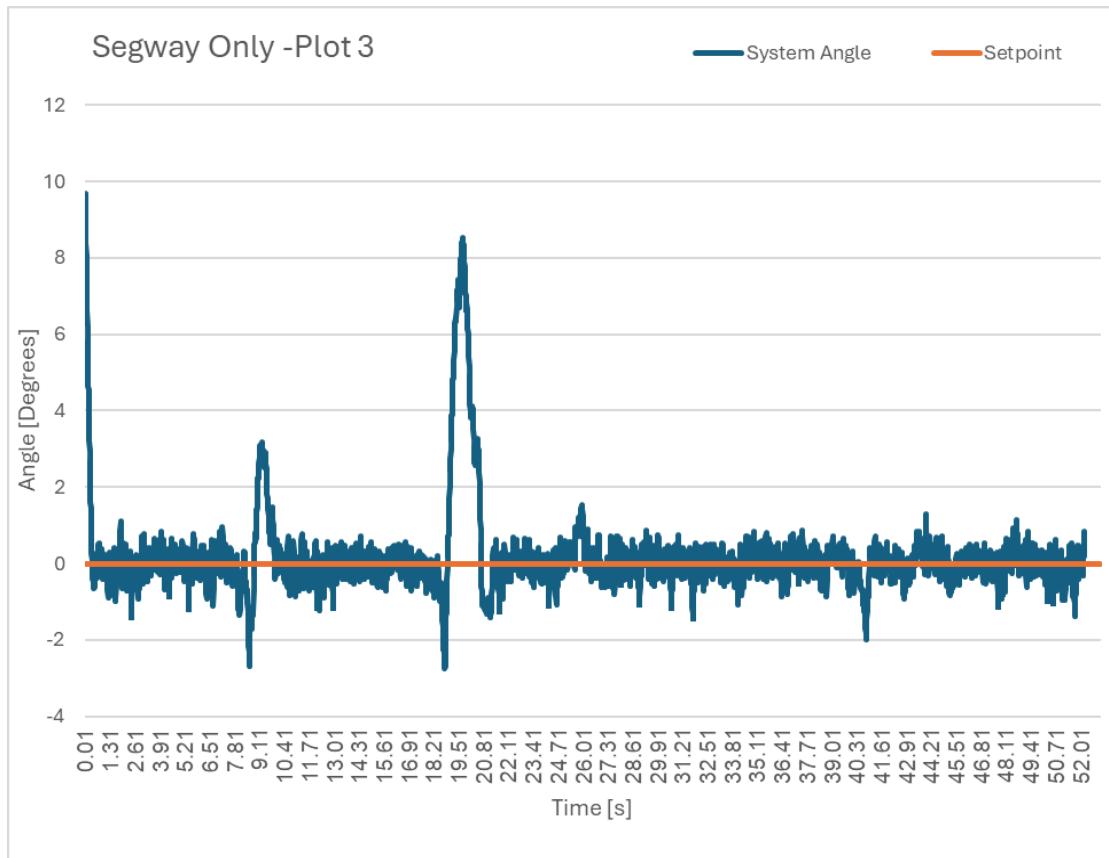
למהלך הניסוי הבא והآخرון, אנו משתמשים במערכת ייצוב אשר כוללת את הפעלת המנועים התחטוניים יחד עם שילוב הפעלת גלגל התגובה.



איור 43 – גרפ' זווית-זמן, ניסוי 5.1



איור 44 – גרפ' זווית-זמן, ניסוי 5.2



איור 45 – גרפ' זווית-זמן, ניסוי 5.3

דיון בתוצאות

ניסוי 5 היהוה שלב קרייטי בתחילת המבחן, שכן הוא בוחן את יכולת המערכת לשמר על יציבות ולהתכנס לזוויות המטרה תוך שימוש במנועים התחertosים בלבד, ללא סיוע מגלגל התגובה. תוצאות הניסוי סיפקו תובנות משמעותיות לגבי תפקודו של גלגל התגובה במערכת הכוללת ואפשרו הבנה מעמיקה יותר של דינמיקת המערכת.

אחד הממצאים המרכזיים של הניסוי היה שהמערכת הצליחה להתכנס לזוויות המטרה של 0 מעלות ולשמור על יציבות, גם ללא גלגל התגובה. זה הוכיח מעודדת המדגימה את היעילות של המנועים התחertosים בשימירה על יציבות המערכת. עם זאת, ההבדל המשמעותי בין ניסוי זה לניסוי הקודם הتبטא במספר הקפיצות בזוויות הנטייה. בעוד שבניסוי הקודם נפתחה קפיצה אחת בלבד, בניסוי הנוכחי נפתחו שלוש קפיצות בכל אחת מהאייטרציות.

העליה במספר הקפיצות מצביעה על תפקודו החשוב של גלגל התגובה בייצוב המערכת. נראה כי גלגל התגובה מסייע ברישום תנודות קיזוניות ומונע מהמערכת לצאת מטווח היציבות שלה. ללא גלגל התגובה, המערכת נוטה לחזור מגבולות היציבות שלה בתדרות גבוהה יותר, מה שמחייב התurbولات ידנית להחזרתה לטווח התפקיד הרצוי.

חשיבות לציין כי למטרות הعليיה במספר הקפיצות, המערכת עדין הצלילה להתכנס לזרזיות המטרה ולשמור על יציבות לאורך זמן. זה מדגיש את הייעולות של מערכת הבקרה PID ואת יכולתם של המנוועים התחטוניים לספק את המומנט הדורש לייצוב המערכת. התוצאות מראות כי המנוועים התחטוניים מהווים מרכיב קריטי ביציבות המערכת, ויש להם תפקיד מרכזי בהשגת היעד של התכנסות לזרזיות המטרה.

ההחלטה לשמר על כל שאר רכיבי המערכת במקום, כולל גלגל התגובה הלא פעיל, הייתה נconaה מבחינה מתודולוגית. זה אפשר השוואה ישירה בין הניסויים ואפשר לבדוק את ההשפעה של גלגל התגובה על ביצועי המערכת. תוצאות הניסוי מדגישות את החשיבות של שמירה על עקבות בתנאי הניסוי ומאפשרות הסחת מסקנות מהימנות.

הניסוי גם חשף את היתרונות והחסרונות של השימוש ברכיבים נגשים וזולים. מצד אחד, המערכת הצלילה להשיג יציבות סבירה גם עם רכיבים פשוטים יחסית. מצד שני, המוגבלות של רכיבים אלו באוט לידי ביטוי בפתרונות התכופות לזרזיות הנティיה. זה מדגיש את האתגר לפיתוח מערכות רובוטיות מתקדמות תוך שמירה על עליונות נמוכות.

לסיכום, ניסוי 5 הדגים כי ניתן להשיג יציבות סבירה גם ללא גלגל התגובה, אך חשף גם את תרומתו המשמעותית לביצועי המערכת. התוצאות מצביעות על כך שהשילוב של גלגל התגובה עם המנוועים התחטוניים מספק את הפתרון האופטימלי ליציבות המערכת. זה מחזק את החלטה להמשיך לניסוי הבא עם שילוב של שני המרכיבים הללו.

התובנות מניסוי זה יהיו קריטיות בהמשך פיתוח המערכת, במיוחד כאשר יתוסף רובד עיבוד התמונה. הבנת התפקיד של כל מרכיב במערכת מאפשר אופטימיזציה טובה יותר של הביצועים הכלולים וتسייע בהטבות עם האתגרים הצפויים בשלבים הבאים של הפרויקט.

5.6 ניסוי 6 – בוחינת יציבות המערכת (גלאג תגובה + מנועים תחכמוניים) בשילוב עיבוד תמונה להתקדמות

מטרה

בניסוי הקודם, ניסוי מס' 5, נמצא כי המערכת מגיבה היטב כאשר פועלות בשילוב של גלאג תגובה ומנועים תחכמוניים וכן הוחלט המשיך לניסוי האחרון ברכף הניסויים כאשר היא תורכב באותה מידה. בניסוי זה, נרצה לבחון את יכולת היציבות של המערכת המשולבת כאשר היא מקבלת פקודות מעיבוד תמונה. נבחן כיצד ניתן לקרוא ולפענה פקודות עצירה והתקדמות למערכת במקביל לשמירה על יציבותה. נעמיד במחן את יכולות המנועים ונבדוק עד כמה ניתן לדוחוף אותם עד שיגיעו לנצח גבול יכולתם ולא יוכלו לעמוד בעומס. לפני שנתחיל את הניסוי עם פקודות ההתקדמות, נרצה לבחון כיצד מגיבה המערכת ואלגוריתם עיבוד התמונה לפקודות עצירה. מאחר וזהו הניסוי האחרון, נרצה להתכנס לכדי כוורתה הפROYיקט ולהASIC מסקנות לגבי המערכת ביחס לניסוי זה כמו גם ביחס לניסוי הראשון.

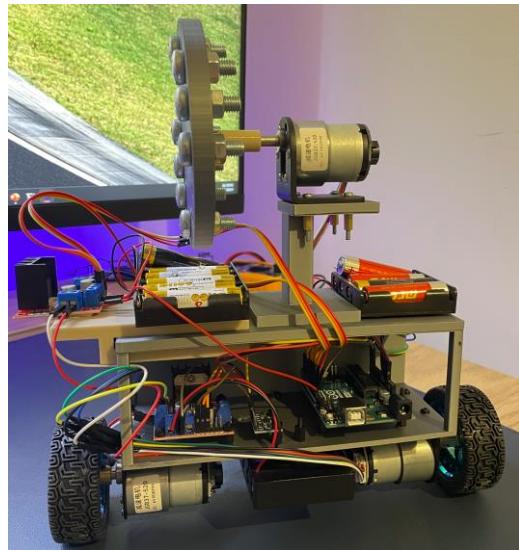
תיאור המערכת בניסוי

המערכת מتبוססת על המשך ישיר למערכת ניסוי 4: מערכת רובוט דו-גלאגי אשר נעזרת בשילובם של המנועים התחכמוניים יחד עם מנוע עליון אשר שלוט בגלאג תגובה כדי להתכנס לזוויות מטרה של 0 מעלות ולשמור על יציבותה.

השינויים במערכת בניסוי זה הוא העובדה כי נרצה לשלг למערכת יכולת לתקדם קדימה בזמן שמירה על יציבות, תוך כדי שהיא מקבלת בזמן אמיתי, פקודות מצלמת המחשב אשר קוראות ומפענחת סימנים קבועים מראש. אנו מתמקדים בצד סימנים אשרبعثת כיוונים למצלמת המחשב, יתבצעו רצף פעולות מוגדר מראש.

- סימן עזר (איור 1) לפקודות עצירה למערכת.
- סימן רכב (איור 2) לפקודות התקדמות למערכת.

הצלמה שנעשה בה שימוש בניסוי זה היא מצלמת המחשב, כאשר המחשב מחובר עם כבל ריען נתון למערכת ובכך מותבצעת העברת הנתונים ביניהם. המחשב משתמש בתוכנית בשפת Python בעורך VS Code. לצורך זה יהיו נכון ומיידי של הסימנים, נusb שימוש באלגוריתם עיבוד תמונה מסוג Haar Cascade.



איור 46 – תיאור מערכת ניסוי 6 : מערכת על בסיס מנועים תחטוניים וגלגל תגובה משולב



איור 47 – תיאור מערכת ניסוי 6 : סימון עצור לעצירת המערכת



איור 48 – תיאור מערכת ניסוי 6 : סימון רכב להתקרבות המערכת

אופן פעולות המערכת

פעולות המערכת מתחולקת ל-2 חלקים. החלק הראשון הוא תכנית הארדואינו אשר שולטת על מערכת הרובוט מבchinת חישוב זווית נתיחה, חישוב אותות לבקרים, והפעלת המנועים. לארדואינו לא מחוברת מצלמה שכן יש כורך בחלק נוסף.

החלק השני הוא תכנית פiyton אשר עשויה שימוש באלגוריתם עיבוד תמונה מסווג Haar Cascade, מפענחת באמצעות מודלים מאומנים מראש את הסימונים בהם אנחנו משתמשים, ושולחת חזרה לתוכנית ארדואינו חיוי לגבי מה סיומו שנקלט. כאשר הארדואינו מקבל את החיוי, הוא מחשב בהתאם ללוגיקת הבקרה איזה מהפעולות נדרשת, ואז מפעיל את המנועים בהתאם.

המערכת מתחילה את פעילותה עם חיבורה למקור מתח (המחשב אליו מועברים ובו מנוטחים הנתוניים). השלב הבא הוא תחילת פעילותה של תכנית הפiyton אשר מפעילה את מצלמת המחשב ויוצרת קו העברת נתונים בין לבני המערכת, לאחר מכן מחוברים בכבב. מרגע זה יש שימוש רציף במצלמת המחשב והיא פועלת בכל רגע נתון ברקע. המצלמה מחפשת כל הזמן את הסימונים שבחרנו.

ברקע פעולת המצלמה, המערכת מתחילה לפעול כאשר יותר מתח שמלי מספק ולאחר מכן עושה את שלב הקליברציה של זווית נתיחה המערכת.

לאחר חישוב זווית התחלה, תחילת המערכת לפעול לוואי כדי להגיע למצב יציב ולהתכנסות לזויה המטריה של 0 מעלות -

- דגימות זווית הנתיחה וחישובה על ידי פילטר המסנן רעשים ותורם לדגימה איקוטית ומדוקית.

- חישוב פלט לבקר PID אשר מחשב את זווית הנתיחה, על ידי הגדרה מראש של מקדמי הבקר. שלב זה של הגדרת מקדמי הבקר חשוב מאוד אחריו יש לו השפעה ישירה על ת gobot המערכת. בקר זה הוא הרראשון מבין שתיים ומשמש כולאה חיצונית. פלט בקרת הזווית נשלח ישירות להוות כאות הכניסה לבקרה מההירות אשר מגיעה בצעד הבא.

- חישוב פלט לבקר PID אשר מקבל בכניסה את אותן בקרת הזווית מהצעד הקודם ומהчисב את אותן הבקרה להפעלת המנועים.

- שליחת אותן בקרת המהירות לבקרת המנועים. נרצה בתחום את גבולות אותן אלו מאוחר ונרצה לקבוע אתו הטעות אשר יפעיל מיידית את המנוע. בקרת המנועים שולטת במהירות הסיבוב, כיוון הסיבוב וחלוקת אותן בין מנוע גלגל התגובה לבין המנועים התוחתוניים.

- גלגל התגובה אשר מחובר למנוע, מסתובב בכיוון נגדו לכיוון נתיחה המערכת. סיבוב זה מפעיל מומנט על המערכת סיבוב ציר הגלגלים התוחתוניים ובכך מתנגד לנפילה. במקביל, המנועים התוחתוניים גם הם מגיבים בהתאם לזוויות הנתיחה אשר נוצרות. שילוב שלושת המנועים מפעיל מומנט כללי על המערכת.

- בכל זמן נתון המערכת קשובה לכוון הנתונים מהצלמה וכאשר מתקבל אחד מהסימנים אשר הוגדרו, המערכת מבצעת את הדrhoosh :

אם זזה סימן עצור, המערכת תפסיק את פעילותה ותשבייה את המערכת.

אם זזה סימן רכב, המערכת מוציאה לפועל פונקציית התקדמות אשר בזמן שMRIה על יציבות נתונה אותן למנועים התוחתוניים להסתובב קדימה במשך זמן מוגדר מראש ובუכמלה מוגדרת מראש.

רצף פעולות אלה מתקיים כל עוד לא התקבלה פקודת יציאה לתוכנית פiyton וזו תסגור את המצלמה ותשבייה את המערכת.



איור 49 – תיאור אופן פעולה המערכת, ניסוי 6

תכנית ארדואינו ופיתון

תכנית הארדואינו נכתבה והופעלת ב Arduino IDE ומחולקת ל 2 תכניות : תכנית עבור עצירה (תכנית ארדואינו 6.1) ותכנית עבור התקדמות (תכנית ארדואינו 6.2).
 תכנית הפיתון נכתבה והופעלת ב VS Code ומחולקת ל 2 תכניות : תכנית עבור עצירה (תכנית פיתון 6.1) ותכנית עבור התקדמות (תכנית פיתון 6.2).

```
1 #include <Wire.h>
2 #include <MPU6050_light.h>
3
4 MPU6050 mpu(Wire); // Create an MPU6050 object
5
6 // Pin definitions for the motors
7 const int PWM_PIN = 5;
8 const int IN1_PIN = 7;
9 const int IN2_PIN = 6;
10 const int IN1_MOTOR1 = 8;
11 const int IN2_MOTOR1 = 9;
12 const int PWM_MOTOR1 = 10;
13 const int IN1_MOTOR2 = 13;
14 const int IN2_MOTOR2 = 12;
15 const int PWM_MOTOR2 = 11;
16
17 // PID controller variables
18 float Kp_outer = 50.0, Ki_outer = 0.0, Kd_outer = 0.0;
19 float Kp_inner = 80.0, Ki_inner = 0.0, Kd_inner = 0.0;
20 float setpoint_angle = 0.0, lastAngleError = 0.0, integral_angle = 0.0;
21 float lastVelocityError = 0.0, integral_velocity = 0.0;
22
23 unsigned long lastTime = 0;
24 const unsigned long SAMPLE_TIME = 5; // Control loop sample time
25 bool systemShutdown = false; // Flag to track if system is shut down
26
27 void setup() {
28     Serial.begin(115200); // Initialize serial communication
29     Wire.begin(); // Initialize I2C communication
30
31     // Initialize motor pins
32     pinMode(PWM_PIN, OUTPUT);
33     pinMode(IN1_PIN, OUTPUT);
34     pinMode(IN2_PIN, OUTPUT);
35     pinMode(IN1_MOTOR1, OUTPUT);
36     pinMode(IN2_MOTOR1, OUTPUT);
37     pinMode(PWM_MOTOR1, OUTPUT);
38     pinMode(IN1_MOTOR2, OUTPUT);
39     pinMode(IN2_MOTOR2, OUTPUT);
40     pinMode(PWM_MOTOR2, OUTPUT);
41
42     // Initialize MPU6050 sensor
43     byte status = mpu.begin();
44     while (status != 0) { // Check if MPU6050 is connected properly
45         Serial.println("MPU6050 initialization failed.");
46         delay(1000);
47     }
}
```

איור 50 – תכנית ארדואינו 6.1 חלק 1

```
48  Serial.println("MPU6050 initialized successfully!");
49  mpu.calcOffsets(); // Calibrate MPU6050
50  Serial.println("Calibration complete!");
51 }
52
53 void loop() {
54     // Check if system is shut down, and stop motors immediately if so
55     if (systemShutdown) {
56         stopMotors(); // Stop all motors
57         Serial.println("System is shutting down...");
58         while (true) {
59             // Keep the system halted indefinitely
60             delay(100);
61         }
62     }
63
64     // Check for serial input
65     if (Serial.available() > 0) {
66         String command = Serial.readStringUntil('\n'); // Read incoming command
67
68         if (command == "SHUTDOWN") {
69             systemShutdown = true; // Set shutdown flag to true
70         }
71     }
72
73     // Continue running the stabilization logic if not shut down
74     unsigned long currentTime = millis();
75     if (currentTime - lastTime >= SAMPLE_TIME) {
76         float deltaTime = (float)(currentTime - lastTime) / 1000.0;
77         maintainStability(deltaTime);
78         lastTime = currentTime;
79     }
80 }
81
82 // Function to maintain stability
83 void maintainStability(float deltaTime) {
84     mpu.update();
85     float angle = mpu.getAngleX(); // Get the tilt angle
86     Serial.print("Maintaining stability, Angle: ");
87     Serial.println(angle);
88     float desiredVelocity = calculateOuterPID(angle, deltaTime);
89     float bottomMotorOutput = calculateInnerPID(desiredVelocity, deltaTime);
90     setMotorSpeed(bottomMotorOutput, desiredVelocity); // Control motors
91 }
92
93 // Function to stop all motors
94 void stopMotors() {
```

איור 51 – תכנית ארדואינו 6.1 חלק 2

```
94 void stopMotors() {
95     analogWrite(PWM_MOTOR1, 0);
96     analogWrite(PWM_MOTOR2, 0);
97     analogWrite(PWM_PIN, 0);
98     digitalWrite(IN1_MOTOR1, LOW);
99     digitalWrite(IN2_MOTOR1, LOW);
100    digitalWrite(IN1_MOTOR2, LOW);
101    digitalWrite(IN2_MOTOR2, LOW);
102    digitalWrite(IN1_PIN, LOW);
103    digitalWrite(IN2_PIN, LOW);
104 }
105
106 // Outer PID control (tilt control)
107 float calculateOuterPID(float angle, float deltaTime) {
108     float angleError = setpoint_angle - angle;
109     integral_angle += angleError * deltaTime;
110     float derivative_angle = (angleError - lastAngleError) / deltaTime;
111     float output = Kp_outer * angleError + Ki_outer * integral_angle + Kd_outer * derivative_angle;
112     lastAngleError = angleError;
113     return output;
114 }
115
116 // Inner PID control (velocity control)
117 float calculateInnerPID(float desiredVelocity, float deltaTime) {
118     float velocityError = desiredVelocity;
119     integral_velocity += velocityError * deltaTime;
120     float derivative_velocity = (velocityError - lastVelocityError) / deltaTime;
121     float output = Kp_inner * velocityError + Ki_inner * integral_velocity + Kd_inner * derivative_velocity;
122     lastVelocityError = velocityError;
123     return output;
124 }
125
126 // Set motor speeds based on PID output
127 void setMotorSpeed(float bottomOutput, float reactionOutput) {
128     int bottomMotorSpeed = map(abs(bottomOutput), 0, 180, 150, 255);
129     bottomMotorSpeed = constrain(bottomMotorSpeed, 150, 255);
130     int reactionWheelSpeed = map(abs(reactionOutput), 0, 180, 150, 255);
131     reactionWheelSpeed = constrain(reactionWheelSpeed, 150, 255);
132
133     // Control bottom motors (left and right)
134     if (bottomOutput > 0) {
135         digitalWrite(IN1_MOTOR1, LOW);
136         digitalWrite(IN2_MOTOR1, HIGH);
137         digitalWrite(IN1_MOTOR2, LOW);
138         digitalWrite(IN2_MOTOR2, HIGH);
```

איור 52 – תכנית ארדואינו 6.1 חלק 3

```
139 } else {
140     digitalWrite(IN1_MOTOR1, HIGH);
141     digitalWrite(IN2_MOTOR1, LOW);
142     digitalWrite(IN1_MOTOR2, HIGH);
143     digitalWrite(IN2_MOTOR2, LOW);
144 }
145 analogWrite(PWM_MOTOR1, bottomMotorSpeed);
146 analogWrite(PWM_MOTOR2, bottomMotorSpeed);
147
148 // Control reaction wheel motor
149 if (reactionOutput > 0) {
150     digitalWrite(IN1_PIN, HIGH);
151     digitalWrite(IN2_PIN, LOW);
152 } else {
153     digitalWrite(IN1_PIN, LOW);
154     digitalWrite(IN2_PIN, HIGH);
155 }
156 analogWrite(PWM_PIN, reactionWheelSpeed);
157 }
```

איור 53 – תכנית ארדואינו 6.1 חלק 4

```
1 #include <Wire.h>
2 #include <MPU6050_light.h>
3
4 MPU6050 mpu(Wire); // Create an MPU6050 object
5
6 // Pin definitions for the motors
7 const int PWM_PIN = 5;
8 const int IN1_PIN = 7;
9 const int IN2_PIN = 6;
10 const int IN1_MOTOR1 = 8;
11 const int IN2_MOTOR1 = 9;
12 const int PWM_MOTOR1 = 10;
13 const int IN1_MOTOR2 = 13;
14 const int IN2_MOTOR2 = 12;
15 const int PWM_MOTOR2 = 11;
16
17 // PID controller variables
18 float Kp_outer = 50.0, Ki_outer = 0.0, Kd_outer = 0.0;
19 float Kp_inner = 80.0, Ki_inner = 0.0, Kd_inner = 0.0;
20 float setpoint_angle = 0.0, lastAngleError = 0.0, integral_angle = 0.0;
21 float lastVelocityError = 0.0, integral_velocity = 0.0;
22
23 unsigned long lastTime = 0;
24 const unsigned long SAMPLE_TIME = 5; // Control loop sample time
25 bool moveForward = false; // Flag to track if forward movement is happening
26 unsigned long forwardStartTime = 0; // Timer to track forward movement
27
28 void setup() {
29     Serial.begin(115200); // Initialize serial communication
30     Wire.begin(); // Initialize I2C communication
31
32     // Initialize motor pins
33     pinMode(PWM_PIN, OUTPUT);
34     pinMode(IN1_PIN, OUTPUT);
35     pinMode(IN2_PIN, OUTPUT);
36     pinMode(IN1_MOTOR1, OUTPUT);
37     pinMode(IN2_MOTOR1, OUTPUT);
38     pinMode(PWM_MOTOR1, OUTPUT);
39     pinMode(IN1_MOTOR2, OUTPUT);
40     pinMode(IN2_MOTOR2, OUTPUT);
41     pinMode(PWM_MOTOR2, OUTPUT);
42
43     // Initialize MPU6050 sensor
44     byte status = mpu.begin();
45     while (status != 0) { // Check if MPU6050 is connected properly
46         Serial.println("MPU6050 initialization failed.");
47         delay(1000);
```

איור 54 – תכנית ארדואינו 6.2 חלק 1

```
48 }
49 Serial.println("MPU6050 initialized successfully!");
50 mpu.calcOffsets(); // Calibrate MPU6050
51 Serial.println("Calibration complete!");
52 }
53
54 void loop() {
55 // Check for serial input
56 if (Serial.available() > 0) {
57     String command = Serial.readStringUntil('\n'); // Read incoming command
58
59     if (command == "FORWARD") {
60         moveForward = true; // Set flag to move forward
61         forwardStartTime = millis(); // Set the start time for moving forward
62         Serial.println("Move forward command received");
63     }
64 }
65
66 // Continue running the stabilization logic
67 unsigned long currentTime = millis();
68 if (currentTime - lastTime >= SAMPLE_TIME) {
69     float deltaTime = (float)(currentTime - lastTime) / 1000.0;
70
71     // If the move forward command was received, move forward for a short time
72     if (moveForward) {
73         if (currentTime - forwardStartTime <= 150) { // Move forward for 500ms (0.5 seconds)
74             moveForwardTinyStep(); // Move forward a little bit while maintaining balance
75         } else {
76             moveForward = false; // Stop moving forward after the time has elapsed
77             Serial.println("Completed forward movement, returning to balance");
78         }
79     } else {
80         maintainStability(deltaTime); // Maintain stability
81     }
82
83     lastTime = currentTime;
84 }
85 }
86
87 // Function to maintain stability
88 void maintainStability(float deltaTime) {
89     mpu.update();
90     float angle = mpu.getAngleX(); // Get the tilt angle
91     Serial.print("Maintaining stability, Angle: ");
92     Serial.println(angle);
93     float desiredVelocity = calculateOuterPID(angle, deltaTime);
94     float bottomMotorOutput = calculateInnerPID(desiredVelocity, deltaTime);
```

איור 55 – תכנית ארדואינו 6.2 חלק 2

```
95     setMotorSpeed(bottomMotorOutput, desiredVelocity); // Control motors
96 }
97
98 // Function to move forward for a short step
99 void moveForwardTinyStep() {
100    // Set a small speed for forward movement while maintaining stability
101    float forwardSpeed = 155; // Small constant speed
102
103    // Keep stabilizing while moving forward
104    mpu.update();
105    float angle = mpu.getAngleX();
106    float desiredVelocity = calculateOuterPID(angle, 0.05); // Use a small deltaTime for stabilization
107    setMotorSpeed(forwardSpeed, desiredVelocity); // Move forward with a small step while balancing
108 }
109
110 // Function to stop all motors
111 void stopMotors() {
112    analogWrite(PWM_MOTOR1, 0);
113    analogWrite(PWM_MOTOR2, 0);
114    analogWrite(PWM_PIN, 0);
115    digitalWrite(IN1_MOTOR1, LOW);
116    digitalWrite(IN2_MOTOR1, LOW);
117    digitalWrite(IN1_MOTOR2, LOW);
118    digitalWrite(IN2_MOTOR2, LOW);
119    digitalWrite(IN1_PIN, LOW);
120    digitalWrite(IN2_PIN, LOW);
121 }
122
123 // Outer PID control (tilt control)
124 float calculateOuterPID(float angle, float deltaTime) {
125    float angleError = setpoint_angle - angle;
126    integral_angle += angleError * deltaTime;
127    float derivative_angle = (angleError - lastAngleError) / deltaTime;
128    float output = Kp_outer * angleError + Ki_outer * integral_angle + Kd_outer * derivative_angle;
129    lastAngleError = angleError;
130    return output;
131 }
132
133 // Inner PID control (velocity control)
134 float calculateInnerPID(float desiredVelocity, float deltaTime) {
135    float velocityError = desiredVelocity;
136    integral_velocity += velocityError * deltaTime;
137    float derivative_velocity = (velocityError - lastVelocityError) / deltaTime;
138    float output = Kp_inner * velocityError + Ki_inner * integral_velocity + Kd_inner * derivative_velocity;
```

איור 56 – תכנית ארדואינו 6.2 חלק 3

```
139     lastVelocityError = velocityError;
140     return output;
141 }
142
143 // Set motor speeds based on PID output for both bottom motors and reaction wheel
144 void setMotorSpeed(float bottomOutput, float reactionOutput) {
145     // Map the PID output to PWM values within the motor speed range
146     int bottomMotorSpeed = map(abs(bottomOutput), 0, 180, 150, 255);
147     bottomMotorSpeed = constrain(bottomMotorSpeed, 150, 255);
148     int reactionWheelSpeed = map(abs(reactionOutput), 0, 180, 150, 255);
149     reactionWheelSpeed = constrain(reactionWheelSpeed, 150, 255);
150
151     // Control bottom motors (left and right)
152     if (bottomOutput > 0) {
153         digitalWrite(IN1_MOTOR1, LOW);
154         digitalWrite(IN2_MOTOR1, HIGH);
155         digitalWrite(IN1_MOTOR2, LOW);
156         digitalWrite(IN2_MOTOR2, HIGH);
157     } else {
158         digitalWrite(IN1_MOTOR1, HIGH);
159         digitalWrite(IN2_MOTOR1, LOW);
160         digitalWrite(IN1_MOTOR2, HIGH);
161         digitalWrite(IN2_MOTOR2, LOW);
162     }
163
164     analogWrite(PWM_MOTOR1, bottomMotorSpeed);
165     analogWrite(PWM_MOTOR2, bottomMotorSpeed);
166
167     // Control reaction wheel motor
168     if (reactionOutput > 0) {
169         digitalWrite(IN1_PIN, HIGH);
170         digitalWrite(IN2_PIN, LOW);
171     } else {
172         digitalWrite(IN1_PIN, LOW);
173         digitalWrite(IN2_PIN, HIGH);
174     }
175
176     analogWrite(PWM_PIN, reactionWheelSpeed);
177 }
```

איור 57 – תכנית ארדואינו 6.2 חלק 4

```
010.py > ...
1  import cv2
2  import serial
3  import time
4
5  # Initialize serial communication with Arduino
6  arduino = serial.Serial('COM4', 115200, timeout=1) # Adjust 'COM4' to the correct port
7  time.sleep(2) # Wait for Arduino to initialize
8
9  # Function to send commands to Arduino
10 def send_command(command):
11     arduino.write(f'{command}\n'.encode())
12     arduino.flush() # Ensure the command is sent immediately
13     print(f"Sent to Arduino: {command}")
14
15 # Function to log serial data received from Arduino
16 def log_serial_data():
17     while arduino.in_waiting > 0: # Check if there's data in the serial buffer
18         try:
19             response = arduino.readline().decode('utf-8', errors='ignore').strip() # Ignore invalid characters
20             if response: # If there is any data, print and log it
21                 print(f"Received from Arduino: {response}")
22         except UnicodeDecodeError:
23             print("UnicodeDecodeError: Could not decode response from Arduino.")
24
25 # Function to simulate sending a stop signal to the Arduino
26 def send_stop_signal():
27     send_command("SHUTDOWN")
28     time.sleep(1) # Give time for the Arduino to process
29     log_serial_data() # Log the response after sending the stop command
30
31 # Function to simulate sending a move forward signal to the Arduino
32 def send_move_forward_signal():
33     send_command("FORWARD")
34     time.sleep(1) # Move forward for 1 second
35     log_serial_data() # Log the response after sending the forward command
36     send_command("STOP")
37     log_serial_data() # Log the response after stopping
38
39 # Function to capture video from the webcam and detect stop and forward signs
40 def capture_video():
41     cap = cv2.VideoCapture(0) # Capture video from the default webcam
42
43     # Load the Haar Cascade classifiers for stop sign and forward sign detection
44     stop_sign_cascade = cv2.CascadeClassifier('stop_sign.xml')
45     forward_sign_cascade = cv2.CascadeClassifier('left-sign.xml') # Load forward sign Haar Cascade
46
47     stop_sign_detected = False # Flag to track if stop sign was detected
48     forward_sign_detected = False # Flag to track if forward sign was detected
```

איור 58 – תכנית פיתון 6.1 חלק 1

```
49
50    while True:
51        ret, frame = cap.read() # Read a frame from the video stream
52        if not ret:
53            break
54
55        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # Convert the frame to grayscale
56
57        # Detect stop signs in the frame
58        stop_signs = stop_sign_cascade.detectMultiScale(
59            gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30)
60        )
61
62        # Detect forward signs in the frame
63        forward_signs = forward_sign_cascade.detectMultiScale(
64            gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30)
65        )
66
67        # Process detected stop signs
68        for (x, y, w, h) in stop_signs:
69            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2) # Draw rectangle
70            send_stop_signal() # Send stop signal to Arduino
71            print("Stop sign detected! System shutting down.")
72            stop_sign_detected = True # Set flag to True to prevent further actions
73            break
74
75        # Process detected forward signs
76        if not forward_sign_detected:
77            for (x, y, w, h) in forward_signs:
78                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2) # Draw rectangle
79                send_move_forward_signal() # Send move forward signal to Arduino
80                forward_sign_detected = True # Set flag to prevent further detections
81                break
82
83        # Display the processed frame
84        cv2.imshow('Video Feed', frame)
85
86        # Continuously log any serial data from Arduino (non-blocking)
87        log_serial_data()
88
89        # Exit the loop if 'q' is pressed or stop sign is detected
90        if cv2.waitKey(1) & 0xFF == ord('q') or stop_sign_detected:
91            break
92
93        cap.release() # Release the video capture object
94        cv2.destroyAllWindows() # Close all OpenCV windows
95
96    if __name__ == "__main__":
97        capture_video()
```

איור 59 – תכנית פיתון 6.1 חלק 2

```

❶ 012.py > ...
1  import cv2
2  import serial
3  import time
4
5  # Initialize serial communication with Arduino
6  arduino = serial.Serial('COM4', 115200, timeout=1) # Adjust 'COM4' to the correct port
7  time.sleep(2) # Wait for Arduino to initialize
8
9  # Function to send commands to Arduino
10 def send_command(command):
11     arduino.write(f"{command}\n".encode())
12     arduino.flush() # Ensure the command is sent immediately
13     print(f"Sent to Arduino: {command}")
14
15 # Function to log serial data received from Arduino
16 def log_serial_data():
17     while arduino.in_waiting > 0: # Check if there's data in the serial buffer
18         try:
19             response = arduino.readline().decode('utf-8', errors='ignore').strip() # Ignore invalid characters
20             if response: # If there is any data, print and log it
21                 print(f"Received from Arduino: {response}")
22         except UnicodeDecodeError:
23             print("UnicodeDecodeError: Could not decode response from Arduino.")
24
25 # Function to simulate sending a move forward signal to the Arduino
26 def send_move_forward_signal():
27     send_command("FORWARD")
28     time.sleep(1) # Move forward for 1 second
29     log_serial_data() # Log the response after sending the forward command
30     send_command("STOP")
31     log_serial_data() # Log the response after stopping
32
33 # Function to capture video from the webcam and detect forward signs
34 def capture_video():
35     cap = cv2.VideoCapture(0) # Capture video from the default webcam
36
37     # Load the Haar Cascade classifier for forward sign detection
38     forward_sign_cascade = cv2.CascadeClassifier('cars.xml') # Load forward sign Haar Cascade
39
40     while True:
41         ret, frame = cap.read() # Read a frame from the video stream
42         if not ret:
43             break
44
45         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # Convert the frame to grayscale
46
47         # Detect forward signs in the frame
48         forward_signs = forward_sign_cascade.detectMultiScale(
49             gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30)
50         )

```

איור 60 – תכנית פיתון 2

```

51
52     # Process detected forward signs
53     for (x, y, w, h) in forward_signs:
54         cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2) # Draw rectangle
55         send_move_forward_signal() # Send move forward signal to Arduino
56         break # Break after detecting and processing the first forward sign
57
58     # Display the processed frame
59     cv2.imshow('Video Feed', frame)
60
61     # Continuously log any serial data from Arduino (non-blocking)
62     log_serial_data()
63
64     # Exit the loop if 'q' is pressed
65     if cv2.waitKey(1) & 0xFF == ord('q'):
66         break
67
68     cap.release() # Release the video capture object
69     cv2.destroyAllWindows() # Close all OpenCV windows
70
71     if __name__ == "__main__":
72         capture_video()
73

```

איור 61 – תכנית פיתון 6.2 חלק 2

תוצאת הניסוי

בניסוי זה בוצעו 3 איטרציות עם תנאים זהים. טרם תחילת איטרציות אלו, ועל מנת ללמידה איך להשתמש בעיבוד התמונה אשר יצרנו, נעשתה איטרציה קודם אשר עשתה שימוש בתוכנית ARDOAINO 6.1 ותכנית PIYTUN 6.6. תכניות אלו ממחפשות את סימן העзор אשר הותג בתחלת הניסוי. המערכת הפעלה והמצלמה נכנסה מיד לפועלה. לאחר הרמת זימן העזר נגד עיני המצלמה, הופסקה לצורך מיידית פועלות המערכת.

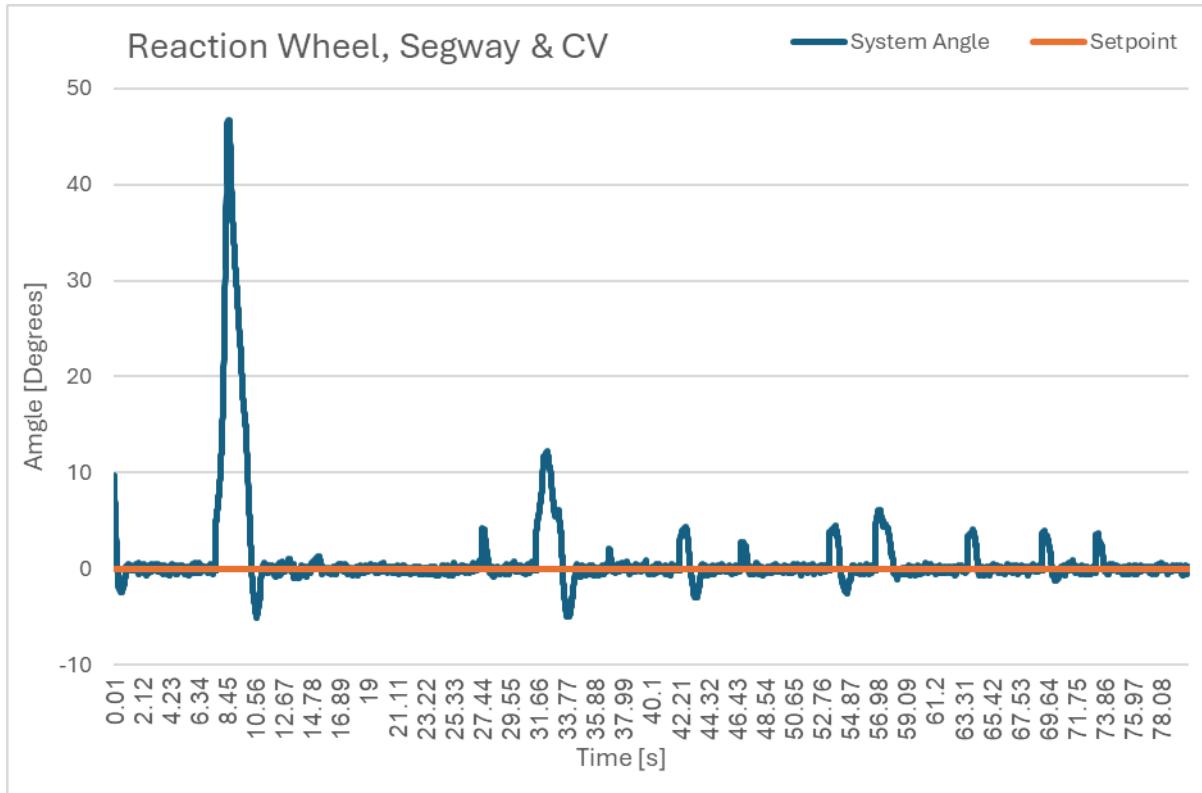
לאחר מכן עברנו לתוכנית ARDOAINO 6.2 ותכנית PIYTUN 6.2. תכניות אלו ממחפשות אחר סימן הרכב אשר מסמן למערכת להתקדם קדימה. הוגדר כי התקדמות קדימה תהיה סיבוב צמד המנועים התחתוניים בעוצה של 50 אחוז ממהירותם המרבית ולמשך שנייה אחת.

בגרף הראשון (איור 5) נעשתה האיטרציה הראשונה של הפעלת המערכת תוך כדי זיהוי סימן התקדמות ולאחר מכן התקדמות בהתאם להגדרה. הקפיצות אשר נראות בגרף הם תגובה המערכת לכל הצגת הסימן. כאשר זהה סימן התקדמות, המערכת מצליחה לענח אותו בהצלחה ולהפעיל את המנועים בהתאם. רואים כי הדבר משפייע על יציבות המערכת באופן מיידי ולאחר מכן מנסה לפצצות בחודות על הפעלת המנועים, מה שמצויא אותה מיציבות. עוד ניתן לראות כי למרות יציאה חזה מזווית המטרה של 0 מעלות, ברוב המקרים המערכת מצליחה להתאושש ולהזור לתנודה סיבוב זווית המטרה די במהירות. כמעט 2 מקרים בהם זווית הנטייה של המערכת יוצאת מגבולות יכולתה להציג ($5 \pm$ מעלות) ולכן אנחנו חייבים להתערב בפעולות המערכת ולתקן באופן ידני, ברוב המקרים המערכת מצליחה להציג לבדה ושמור על יציבות.

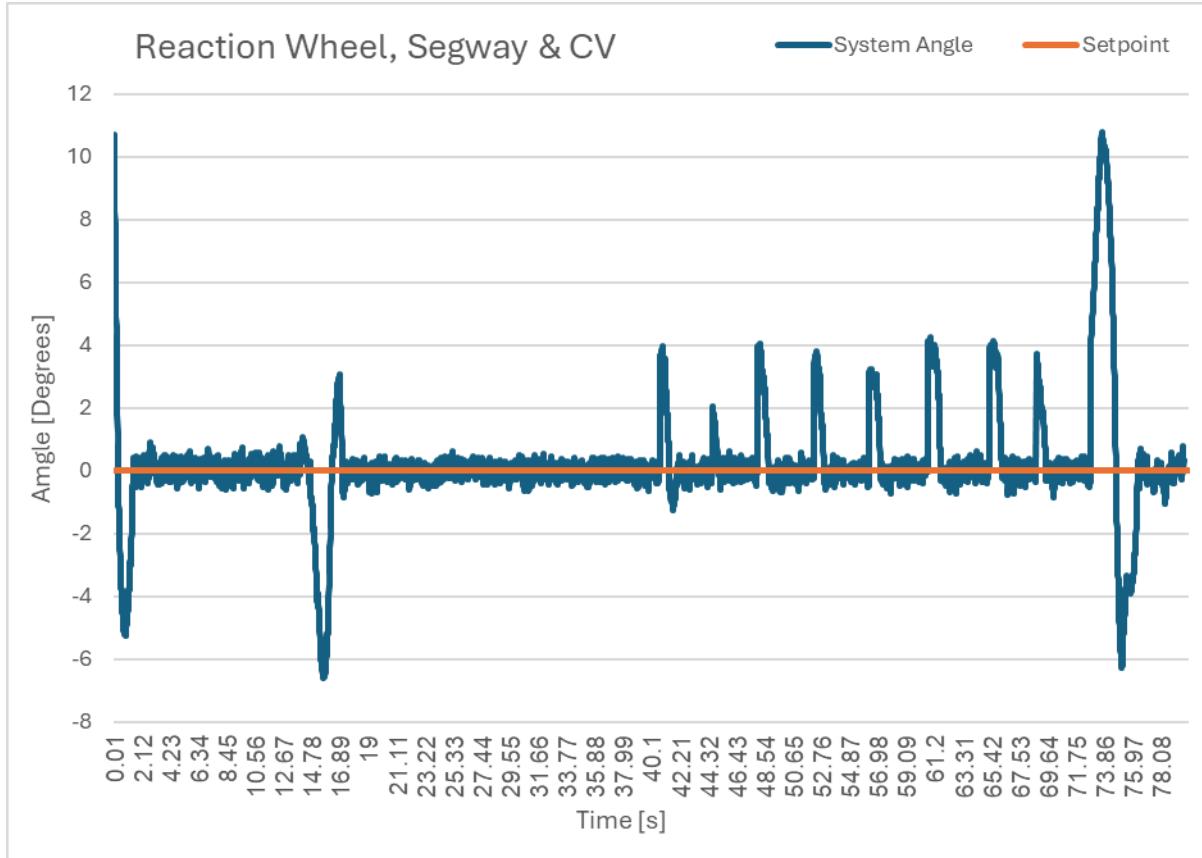
בגרף השני (איור 6) מתאפשרות תוצאות זהות לאלה אשר היו באיטרציה הראשונה, וגם כאן רואים כי המערכת הרוב המקרים מצליחה להכנס לזוויות המטרה ולשמור על יציבות.

בוצעה איטרציה נוספת כדי לקבל תוצאה חד משמעית לגבי יכולת המערכת. בגרף השלישי (איור 7) גם כן מתאפשרות תוצאות זהות לאיטרציות הקודמות.

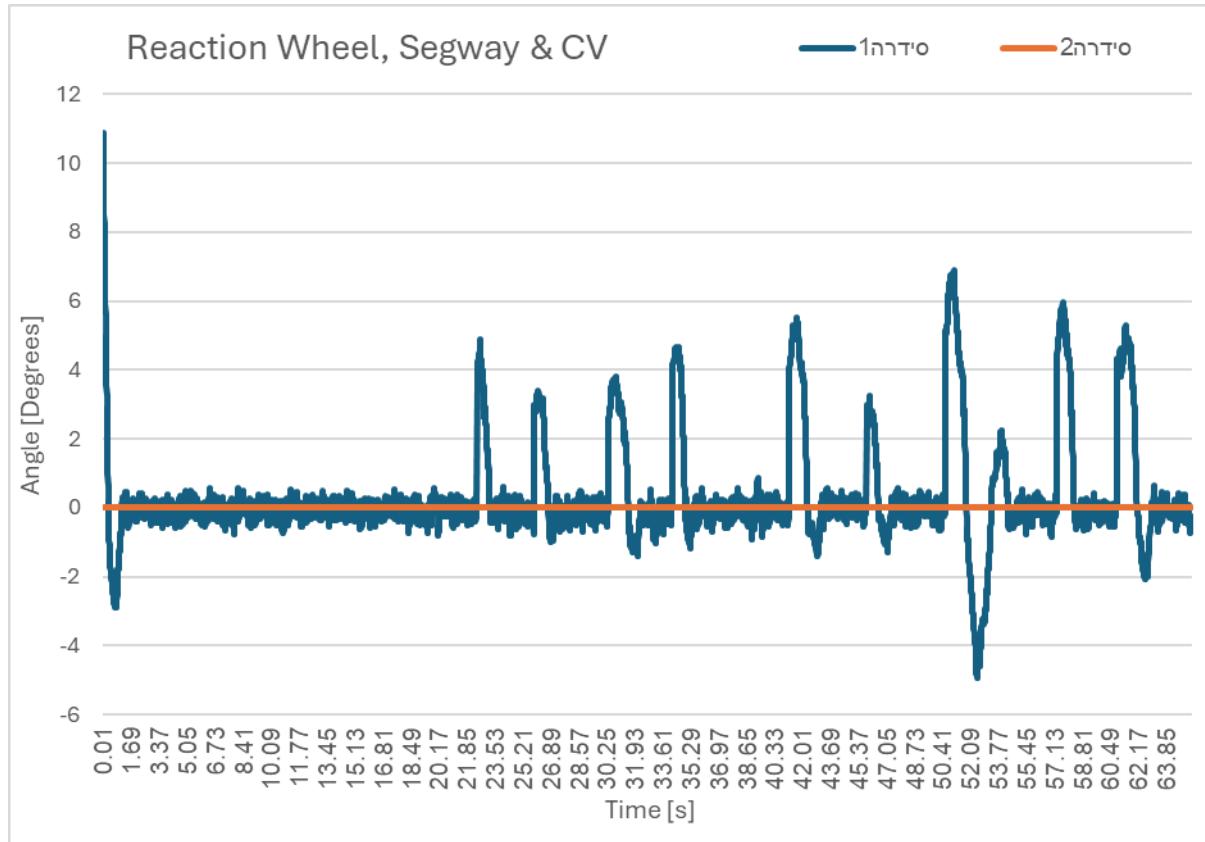
ניתן לומר באופן חד משמעי כי המערכת המשולבת יחד עם עיבוד התמונה מצליחה לעמוד במטרת הניסוי לאחר והיא מותאמת לזוויות המטרה של 0 מעלות במהירות, בנוסף לשומר על יציבות למרות הפוקודה להתקדם קדימה.



איור 62 – גרף זווית-זמן, ניסוי 6.1



איור 63 – גרף זווית-זמן, ניסוי 6.2



איור 64 – גרפ' זווית-זמן, ניסוי 6

דיון בתוצאות

ניסוי 6 מראה את השלב המסכם והמכרע בסדרת הניסויים, שכן הוא משלב את כל המרכיבים שנחקרו בניסויים הקודמים יחד עם יכולת חדשה של עיבוד תמונה. מטרת הניסוי הייתה לבחון את יציבות המערכת המשולבת (גלגלי תגובה ומנועים תחתונים) תוך כדי קבלת פקודות מעיבוד תמונה, ובכך להתקרב למטרה הסופית של הפרויקט – רובוט דו-גלגלי מיוצב עם יכולת תנועה מבוססת עיבוד תמונה.

אחד ההישגים המשמעותיים של הניסוי היה הצלחה בשילוב מערכת עיבוד התמונה עם מערכת הבקרה של הרובוט. השימוש באלגוריתם Haar Cascade ליזיהוי סימנים ויזואליים הוכיח את עצמו כפתרון יעיל ומדויק. היכולת של המערכת לזהות ולהגיב לסימני "עוצר" ו"התقدس" מדגימה את הפוטנציאל של שילוב ראייה ממוחשבת במערכות רובוטיות מסווג זה.

תוצאות הניסוי מראות כי המערכת הצליחה לשמור על יציבות המטרה של 0 מעלות, גם כאשר קיבלה פקודות תנועה. זה הוכיח שמודולות משמעותיות מהניסויים הקודמים, שכן היא מדגימה את היכולת של המערכת לא רק להתייצב, אלא גם לבצע פעולות דינמיות תוך שמירה על יציבות. עם זאת, הקפיצות שנצפו בזווית הנטייה בעת קבלת פקודות תנועה מדגימות את האתגר בשמירה על יציבות בזמן תנועה.

נקודה מעניינת היא ההבדל בין תגوبת המערכת לפקודת "עוצר" לעומת פקודת "התקדם". בעוד שפקודת העצירה הביאה להפסקה מיידית של פעולת המערכת, פקודת התקדמות גורמת לתגובה מורכבת יותר, כולל ניסיונות של המערכת לפצות על השינויים בזווית הנטיה. זה מדגיש את הצורך בתכנון זהיר של אלגוריתמי הבדיקה כדי להתמודד עם מצבים מורכבים.

חשוב לציין את הייעילות של מערכת הבדיקה PID המשולבת (חיצונית ופנימית) בהתקומות עם השינויים הדינמיים. היכולת של המערכת לחזור לתנודה סביבה זווית המטרה לאחר הפרעות מראה על חוסן ויציבות של אלגוריתם הבדיקה. עם זאת, העבודה שבמקרים מסוימים נדרשה התערבות ידנית מצביעה על הצורך בשיפור נוספים של מערכת הבדיקה לטיפול במצבים קיצוניים.

השימוש בשתי תוכנות נפרדות (Python ו-Arduino) לבקרה המערכת ולעיבוד התמונה מדגים גישה מודולרית ויעילה לפיתוח מערכות מורכבות. עם זאת, זה גם מעלה שאלות לגבי האפשרות לשלב את כל הֆונקציונליות במערכת אחת בעתיד, מה עשוי לשפר את זמני התגובה ואת הייעילות הכוללת.

לסיכום, ניסוי 6 מראה הוכחה מרשימה ליכולת שלב מערכות ייצוב מכניות עם יכולות עיבוד תמונה מתקדמות ברובוט דו-גלגלי. התוצאות מראות כי המערכת מסוגלת לשמור על יציבות תוך כדי ביצוע פעולות דינמיות בתגובה לגירויים ויזואליים. עם זאת, הניסוי גם חושף אתגרים הדורשים מחקר נוספת, כמו שיפור היציבות בזמן תגובה ופיתוח אלגוריתמים מתקדמים יותר לטיפול במצבים קיצוניים.

הפרויקט בכללו מדגים את הפטונצייאל של שילוב טכנולוגיות שונות (בדיקה, מכנית, וראיה ממוחשבת) לייצור מערכות רובוטיות מתקדמות. ההצלחה בשימוש ברכיבים זולים ונגישים מראה כי ניתן לפתח מערכות מורכבות גם עם משאבים מוגבלים, מה שפותח אפשרויות רבות למחקר ופיתוח בתחום הרובוטיקה.

פרק 6 – סיכום ומסקנות

פרויקט זה ה证实 בפיתוח רובוט דו-גלגלי מתקדם, המשלב יכולות מבוססות גלגל תגובה עם יכולת תנועה מונחתית עיבוד תמונה. סדרת הניסויים שבוצעה חשפה לתובנות מעמיקות לגבי דינמיקת המערכת ואפשרה אופטימיזציה הדרגתית של ביצועה. הניסוי הראשון, שבחן את השפעת גלגל התגובה לבדו, הדגיש את האתגרים בשימוש ברכיבים פשוטים וזולים. למרות שגלגל התגובה הצליח להשפיע על יכולות המערכת, הוא לבדו לא הספיק להשיג את רמת היציבות הנדרשת. תוצאה זו הובילה לחיפוש אחר פתרונות משלימים. הניסויים עם מסה מרנסת, הן לבד והן בשילוב עם גלגל התגובה, לא הניבו את התוצאות המצופות. למרות שהמסה הדרישה לא הפחיתה תנודות במידה מסוימת, היא לא סיימה את רמת היציבות הדרישה ואף יצרה אתגרים חדשים בשילוב עם גלגל התגובה. ממצאים אלו הובילו להחלטה לנוטש גישה זו ולהיפך פתרונות אלטרנטיביים. נקודת המפנה הגיעה בניסוי הרביעי, כאשר שולבו המנוונים התחתוניים עם גלגל התגובה. שילוב זה הוכיח את עצמו כפתרון אופטימלי, מאפשר התכנסות מהירה לזרימת המטרה תוך שמירה על יכולות ארוך זמן. תוצאה זו הדגישה את החשיבות של גישה הוליסטית בתכנון מערכות רובוטיות מורכבות. הניסוי החמישי, שבחן את המערכת ללא גלגל התגובה, חיזק את ההבנה לגבי תרומותו הקritisטיות של גלגל התגובה לציבות המערכת. למרות שההדרישה הדרישה לא שומרה על יכולות מסוימות גם ללא גלגל התגובה, ביצועה היו נחותים משמעותית בהשוואה למערכת המשולבת. תובנה זו הייתה מכרעת בהחלטה לשמר את השימוש של גלגל התגובה עם המנוונים התחתוניים בשלב הסופי של הפרויקט. הניסוי האחרון והמכריע הוכיח את יכולת המערכת לשומר על יכולות גם בתנאים דינמיים של תנוצה והאצה, תוך כדי תגובה לפקוודות המתקבלות מעיבוד תמונה. הצלחה זו מדגימה את הפוטנציאל של שימוש טכנולוגיות בקרה מתקדמות עם יכולות ראייה ממוחשבת ברובוטיקה.

לסיכום, הפרויקט השיג את מטרתו העיקרית: פיתוח רובוט דו-גלגלי המתייצב באמצעות גלגל תגובה ומסוגל לנوع בהתאם לפקוודות המתקבלות מעיבוד תמונה. למרות שהמטרה המקורית של ייצוב המערכת באמצעות מנוע אחד בלבד לא הושגה במלואה, הפרויקט הצליח למצוא פתרון עיל המשלב מספר מרכיבים בעלות נוכחותיחסית.

הממצאים מדגימים מספר נקודות חשובות:

1. **חשיבות האיזון בין פשוטות ויעילות:** השימוש ברכיבים זולים ונגישים הציב אתגרים, אך גם דחף לחדשות ויצירתיות בפתרונות.
2. **ערך הגישה האיטרטיבית:** כל ניסוי סייף תובנות שהובילו לשיפורים בניסוי הבא, מדגים את החשיבות של תהליך פיתוח מדורג ובסיס נתוניים.
3. **יתרונות השימוש הטכנולוגי:** האינטגרציה של מערכות מכניות, אלקטרוניות ותוכנה מתקדמת אפשרה יצירת מערכת מורכבת ויעילה.
4. **גמישות בתכנון:** יכולת לשנות את הגישה (למשל, יותר על רעיון המנוע היחיד) בהתאם לממצאים אפשרה להגיע לפתרון מיטבי.

פרויקט זה פותח אפקטים חדשים למחקר עתידי, כולל אופטימיזציה נוספת של אלגוריתמי הבקרה, שיפור יכולות עיבוד התמונה, והרחבת יכולות התנועה של הרובוט. ההצלחה בפיתוח מערכת מורכבת זו באמצעות מוגבלים מדגישה את הפוטנציאל ליישומים עתידיים בתחוםים כמו רובוטיקה ניידת, מערכות אוטונומיות, ואפיילו יישומים תעשייתיים וביתיים.

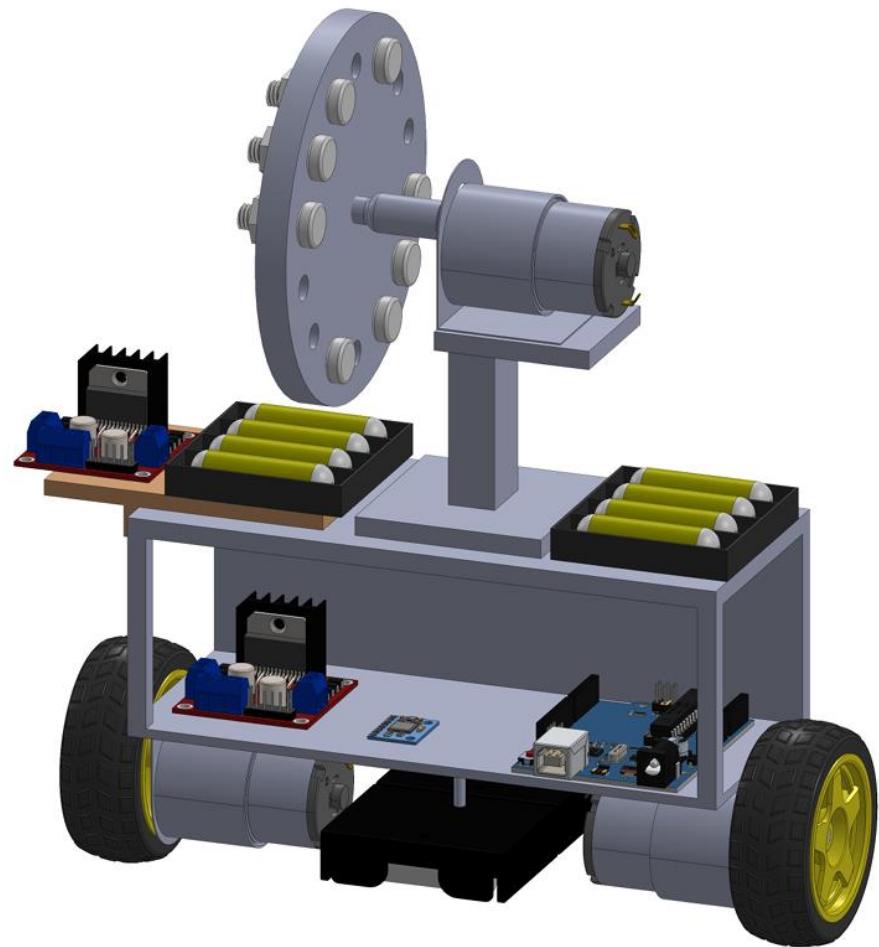
לבסוף, הפרויקט מדגיש את החשיבות של גישה רב-תחומי בפיתוח מערכות רובוטיות מתקדמות, משלב ידע מגוון תחומיים כולל מכנית, אלקטרוניקה, בקרה, ועיבוד תמונה. ההצלחה בשילוב כל אלה מהוות בסיס איתן להמשך מחקר ופיתוח בתחום הרובוטיקה המתקדמת.

פרק 7 – ביבליוגרפיה

- [1] Osama Jamil, Mohsin Jamil, Yasar Ayaz, Khubab Ahmad, 2014 International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE) Islamabad, Pakistan
- [2] Morgan & Claypool Publishers (2007) , **The Reaction Wheel Pendulum**
- [3] Luis Arreola, Gesem Gudi and Gerardo Flores , arXiv: 1903.03947v1 [cs.RO] , 2019
- [4] Neil Kuyvenhoven, Calvin College, ENGR. 315 2002
- [5] SYSTEMSSCIENCE&CONTROLENGINEERING: ANOPENACCESSJOURNAL
2018,VOL.6,NO.1,1–11
- [6] <https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf>
- [7] <https://datasheet.octopart.com/A000066-Arduino-datasheet-38879526.pdf>
- [8] <https://www.4project.co.il/product/metal-gearmotor-20d-6v-590rpm-extended-shaft>
- [9] <https://www.handsontec.com/dataspecs/MPU6050.pdf>
- [10] <https://www.amazon.com/JGB37-520-Torque-Motor-Electric-960RPM/dp/B0CCY8HX6S>
- [11] https://handsontec.com/dataspecs/motor_fan/XD-37GB555.pdf

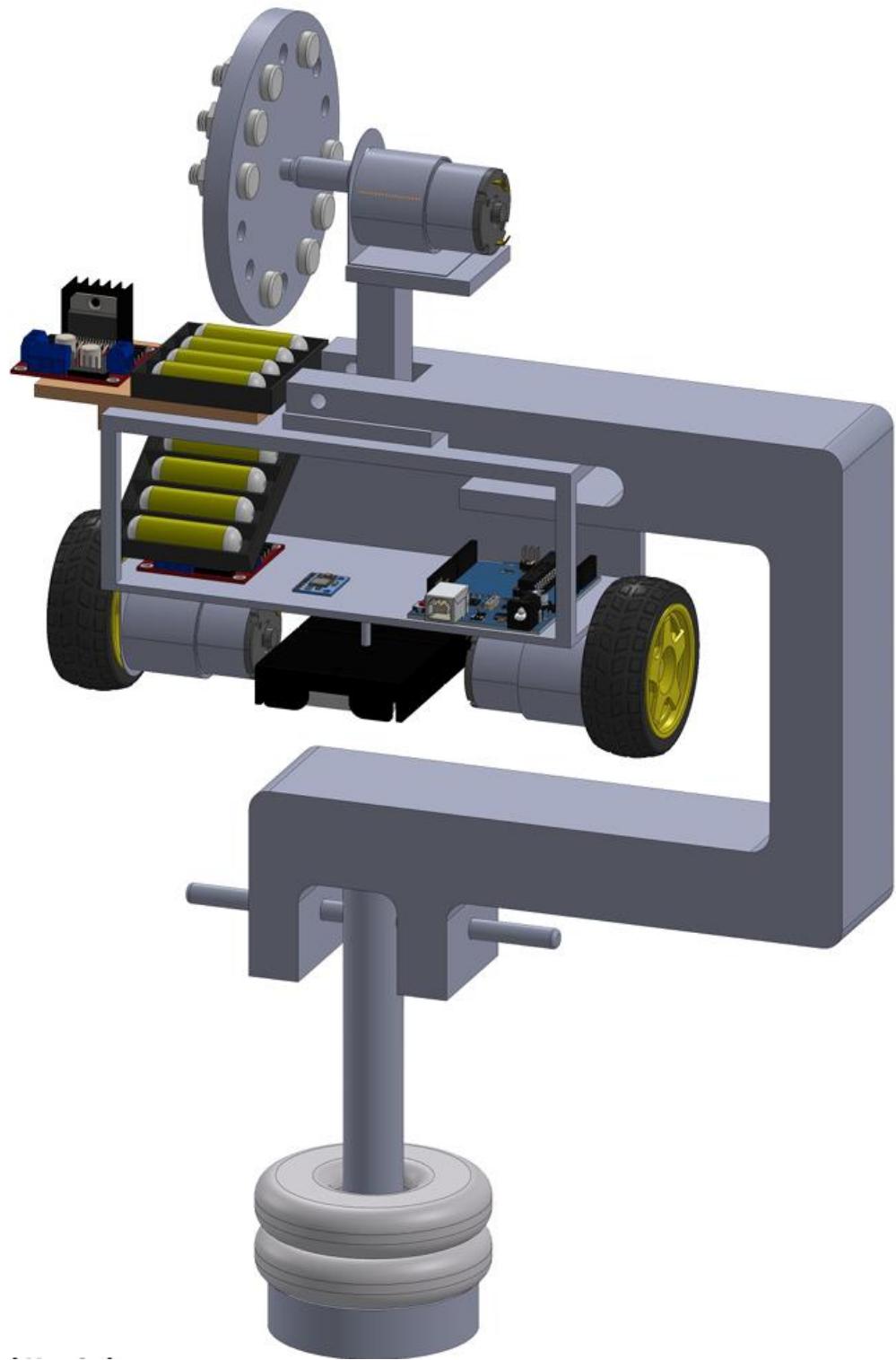
8.1 מודל המערכת

8.1.1 מודל המערכת לפי ניסויים ללא משקלות (קונFIGורציה 1)



איור 65 - מודל סאגוני לניסויים ללא משקלות

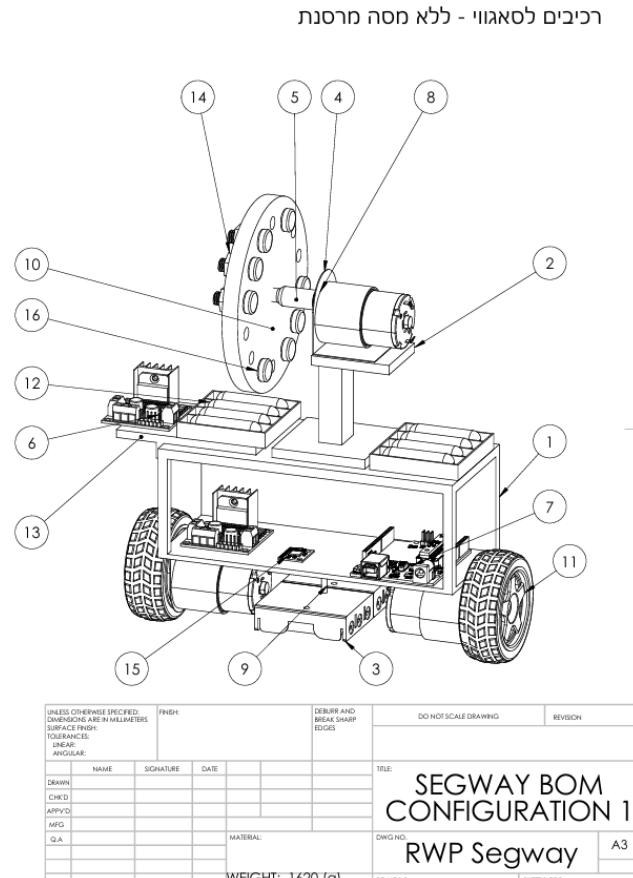
8.1.2 מודל המערכת לפי ניסויים כולל שימוש במקולות (קונפיגורציה 2)



איור 66 - מודל סאגוי לניסויים עם משקולות

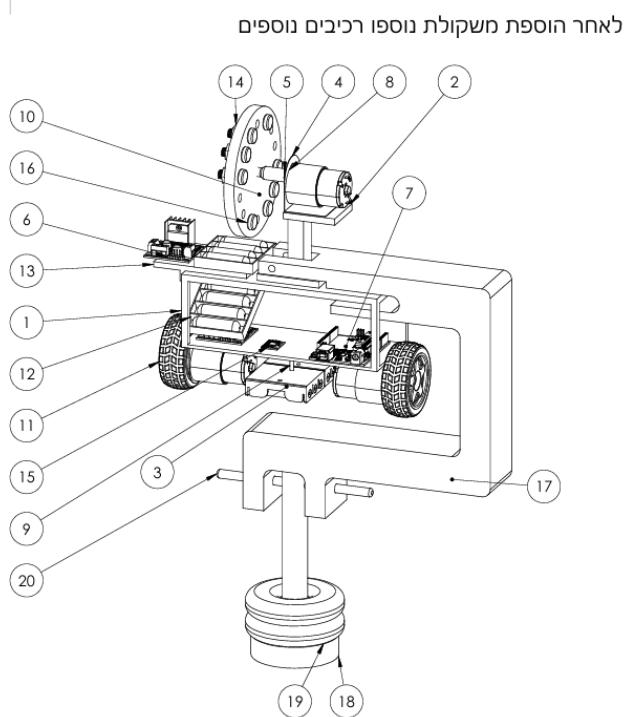
BOM 8.2 מודל המערכת**8.2.1 מודל BOM - ניסויים ללא משקלות**

ITEM NO.	PART NUMBER	DESCRIPTION	WEIGHT (g)	QTY.
1	Base for HW	בסיס שלדה	210	1
2	Reaction Wheel Bracket 02	בסיס שלדה מנע גלגל תגובה	50	1
3	3AA ASSY.step	בית סוללות 3 סוללות 1.5 [ז]	75	2
4	DC motor bracket	תשבץ מנע	40	3
5	Adaptor012	מתאם מנע לגלגל	3	1
6	L298N Motor Driver	דראיבר מנע	25	2
7	Arduino Uno	ברker ארדואינו אונו	50	1
8	JGB37-520.STEP	מנע	140	3
9	bolt M2	בורג החיבור בין השלדה לבית הסוללות החתומים	5	1
10	Reaction Wheel	גלגל תגובה	106	1
11	Wheel D65x25	גלגלים מחזוניים	35	2
12	4AA Batary	בית סוללות 4 סוללות 1.5 [ז]	100	2
13	wood chassis	תשבץ עץ פפייר	20	1
14	nut M8	אום לבורא M8	5	8
15	MPU6050 v2.step	חישון תאוצה	5	1
16	bolt M8	בורג M8	15	8

**איור 67 – BOM – סאגוי ללא משקלות**

8.2.2 מודל BOM - ניסויים בתוספת משקלות

ITEM NO.	PART NUMBER	DESCRIPTION	WEIGHT (g)	QTY.
1	Base for HW	בסיס שלדה	210	1
2	Reaction Wheel Bracket 02	בסיס שלדת מנוע גלגל תגובה	50	1
3	3AA ASSY.stp	בית סוליות 3 סוליות 1.5 [v]	75	2
4	DC motor bracket	תושבת מנוע	40	3
5	Adaptor012	מתאם מנוע לגלאג	3	1
6	L298N Motor Driver	דיזיר מנוע	25	2
7	Arduino Uno	בקר ארדואינו אונו	50	1
8	JGB37-520.STEP	מנוע	140	3
9	bolt M2	ברוג החיבור בין השלדה לבין הסוללות התחתון	5	1
10	Reaction Wheel	גלגל תגובה	106	1
11	Wheel D65x25	גלגל תחתוניים	35	2
12	4AA Batary	בית סוליות 4 סוליות 1.5 [v]	100	2
13	wood chassis	תושבת עץ ספיידר	20	1
14	nut M8	אום לבורג M8	5	8
15	MPU6050 v2.step	חיישן תנועה	5	1
16	bolt M8	בורג M8	15	8
17	Damper_Bracket	תפסונית למשקולת	1083	1
18	dumbbell_rod	בסיס המשקלות	165	1
19	weight plate	שקלנות	500	2
20	connector dumbbell	מחבר תפנוני לבסיס משקלות	20	1

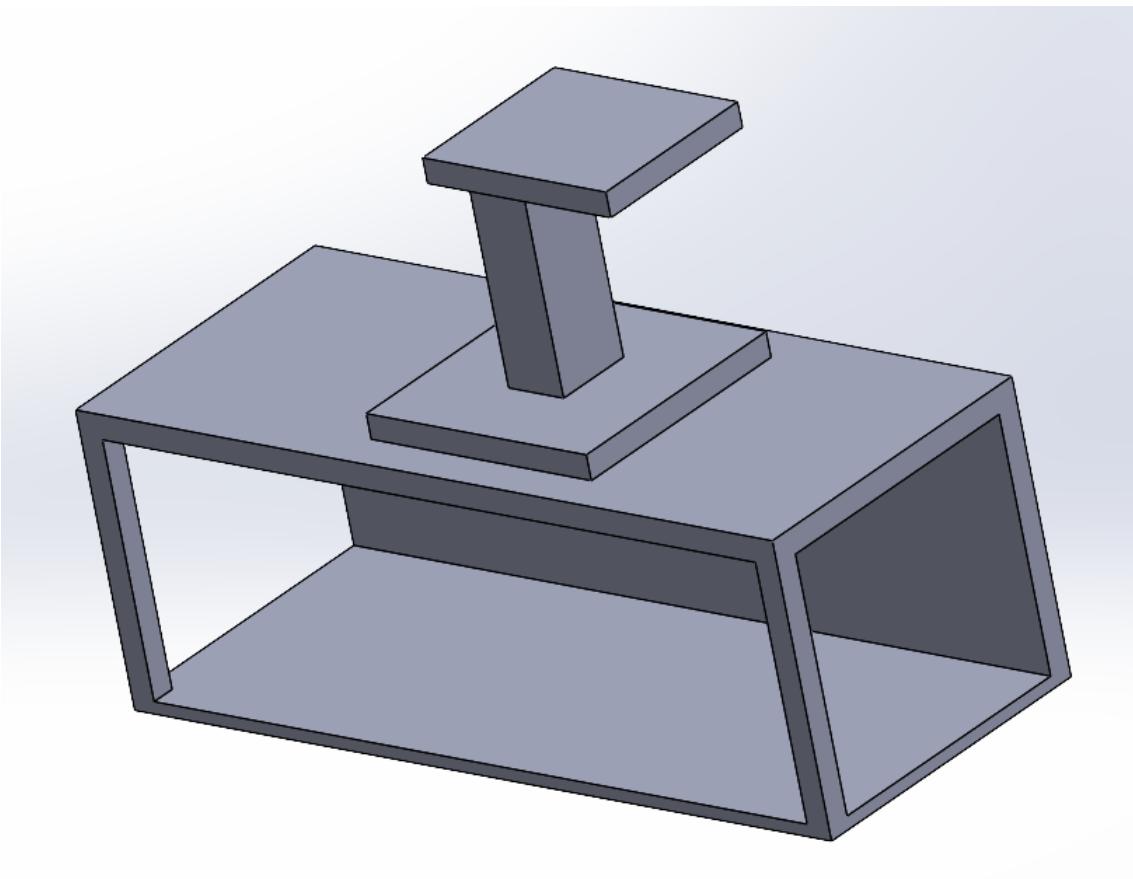


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DO NOT SCALE DRAWING	
DRAWN:	SIGNATURE:	DATE:	REVISION:
CHKD:			
APV/D:			
MFG:			
QA:	MATERIAL:	DWG NO:	A3
		WEIGHT: 3890 (g)	SCALE:1:1
			SHEET 2 OF 9

איור 68 - BOM - סאגוי עם משקלות

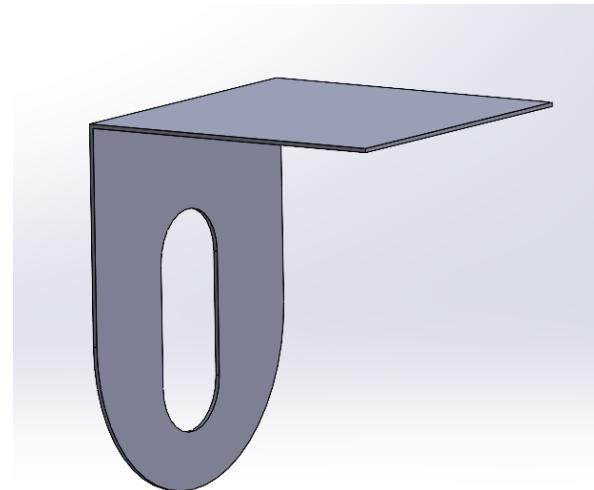
8.3 רכיבי המערכת :**8.3.1 רכיבים לא סטנדרטים – הזרשות ועיבוד****שילדה**

השילדזה מורכבת משני חלקים נפרדים בטבלת ה BOM , אך הזרשה באופן מאוחד בשבייל לשימור על חזק גבואה , ולא להיתר למערכת נקודות חולשה .
מורכבת מ PLA עם 60 % מיילוי חומר

**איור 69 - שילדזה**

תושבת מנוע

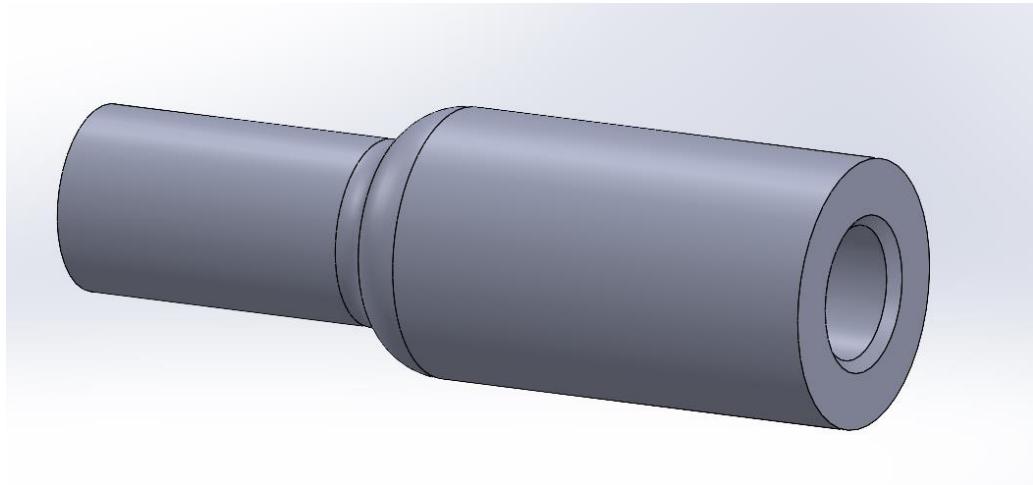
אחרαιות על קיבוע המנוע ולמניעת תזוזות לא רצויות בעת פעילותו



איור 70 - תושבת מנוע

מתאים מנוע לגלגל

עקב אורך ציר המנוע והחיש מהתקלות לא רציה בעת סיובו של הגלגל תגובה, ישנו צורך במתאים שירחיק מעט את הגלגל, ובכך לוודא שלא יהיו התנגדויות בין השלדה לגלגל בתנועתו

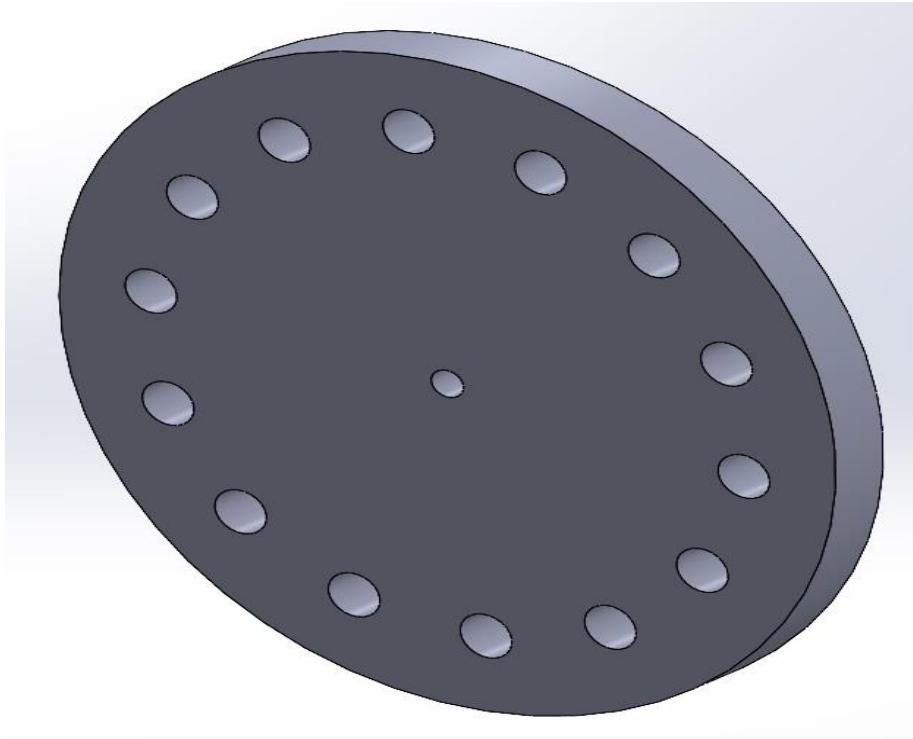


איור 71 - מתאים מנוע לגלגל

גלגל תגובה

הגלגל שאליו מתחבר המנוע ובאמצעות תנועתו נוצרת היציבות.

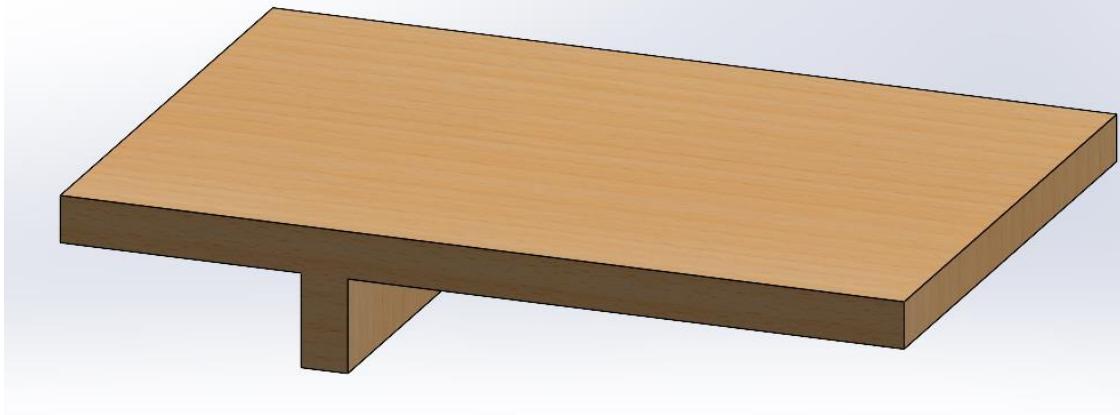
תוכנן עם 14 קדחים לברגיני M8 כך שתיהיה יכולה תמרון בעת הצורך בהוספת משקל או המעטתו.
הודפס ב PLA 60% מיולי .



איור 72 - גלגל תגובה

טושבת עץ ספייסר

בכדי לתת מקום נוספת לחישנים שלא בעלי מושב רב מדי ולהפריע לאיזון המערכת

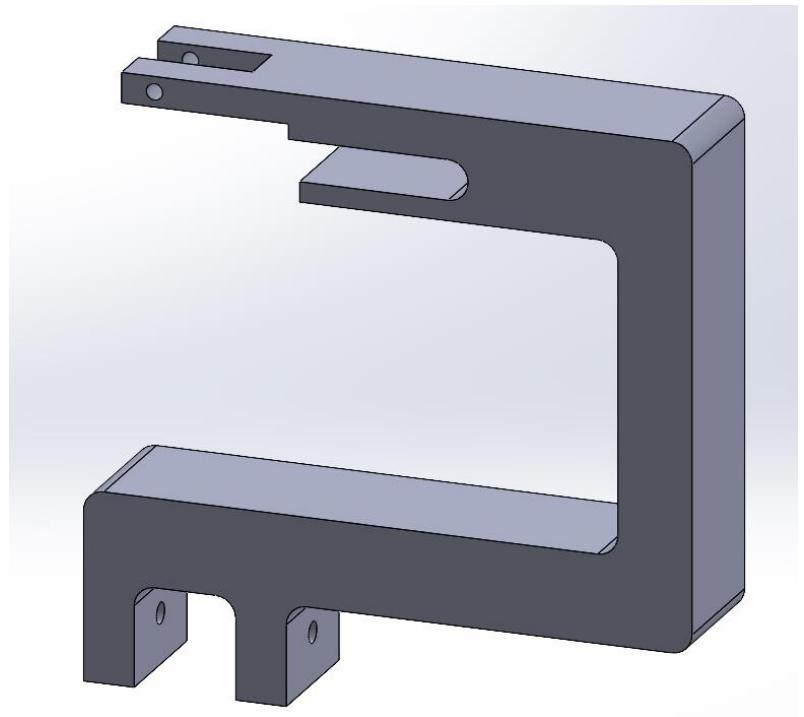


איור 73 - טושבת עץ ספייסר

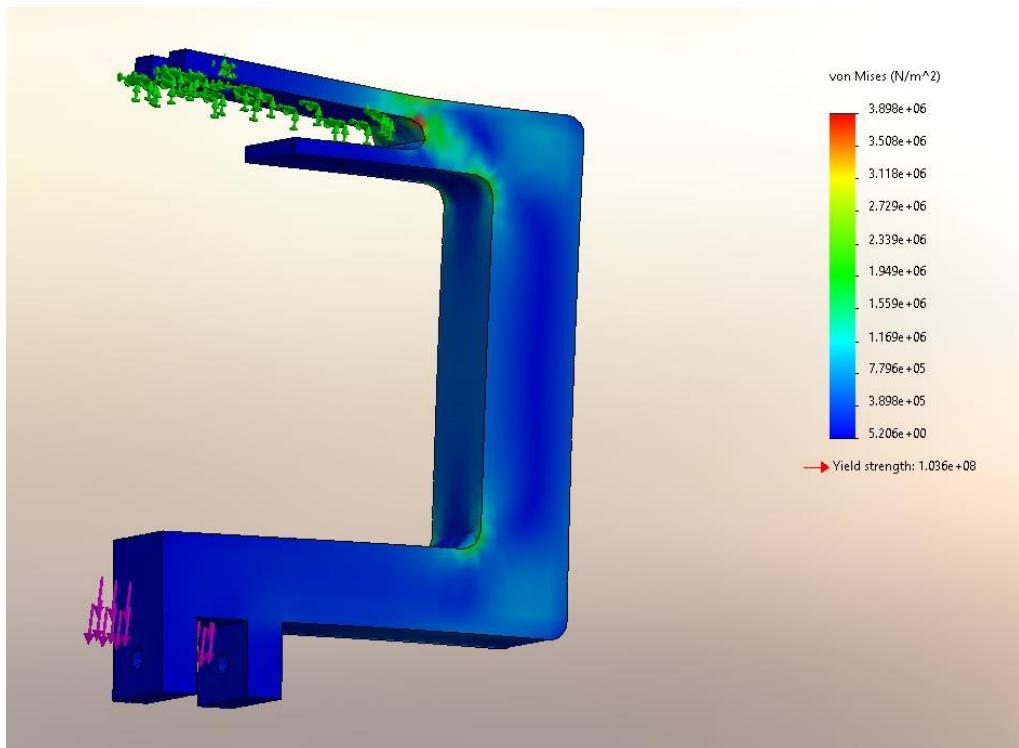
תפסנית למשקלות

הגוף המקשר בין המשקלות לרובוט.

מתוכנן על מנת לעמוד בעומסים הנוצרים עליו מהמשקלות ומגוף הסאגוי. מודפס בPLA 25% מיולי.



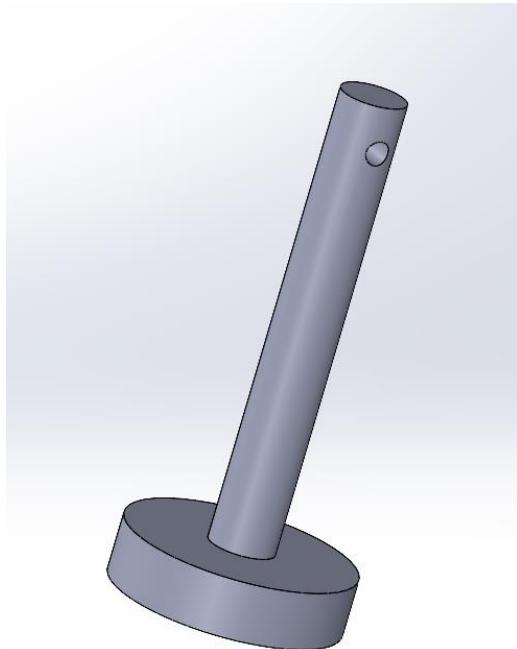
איור 74 - **תפסנית למשקלות**



איור 75 **תפסנית למשקלות-- אנליזות חוזק**

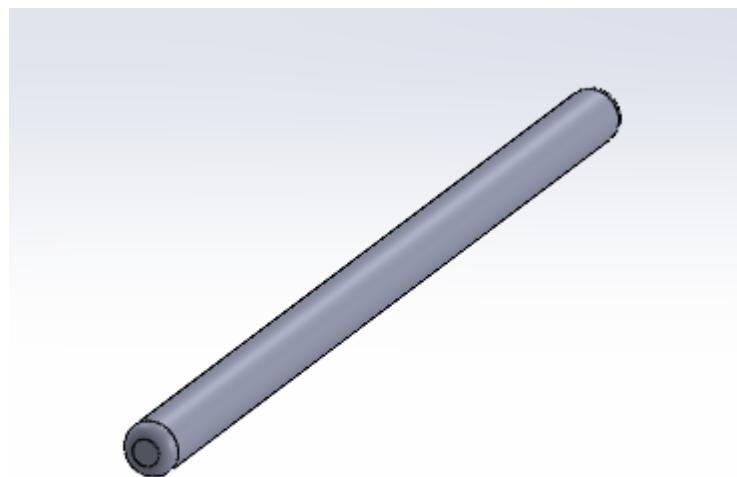
בסיס המשקולת

מקשר בין המשקולת לתפסנית



איור 76 - בסיס המשקולות

מחבר תפסנית לבסיס המשקולת



איור 77 - מחבר תפסנית לבסיס המשקולות

8.3.2 רכיבים אלקטרוניים וסטנדרטים

L298N Dual H-Bridge Motor Driver

Brief Data

- Input Voltage:** 3.2V~40Vdc
- Driver:** L298N Dual H Bridge DC Motor Driver
- Power Supply:** DC 5 V - 35 V
- Peak current:** 2 Amp
- Operating current range:** 0 ~ 36mA
- Control signal input voltage range:**

Low: $-0.3V \leq Vin \leq 1.5V$.

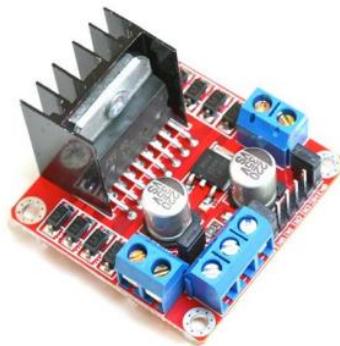
High: $2.3V \leq Vin \leq Vss$.

- Enable signal input voltage range:**

Low: $-0.3 \leq Vin \leq 1.5V$ (control signal is invalid).

High: $2.3V \leq Vin \leq Vss$ (control signal active).

- Maximum power consumption:** 20W (when the temperature $T = 75^{\circ}C$).
- Storage temperature:** $-25^{\circ}C \sim +130.0^{\circ}C$
- Size:** 3.4cm x 4.3cm x 2.7cm
- price:** 10 NIS



[6] H-Bridge - 78

Microcontroller - ATmega328

Operating Voltage - 5V

Input Voltage (recommended) - 7-12V

Input Voltage (limits) - 6-20V

Digital I/O Pins - 14 (of which 6 provide PWM output)

Analog Input Pins - 6

DC Current per I/O Pin - 40 mA

DC Current for 3.3V Pin - 50 mA

Flash Memory - 32 KB of which 0.5 KB used by bootloader

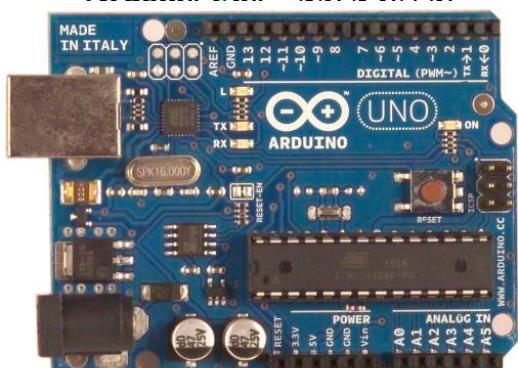
SRAM - 2 KB

EEPROM - 1 KB

Clock Speed - 16 MHz

Price - 100 NIS

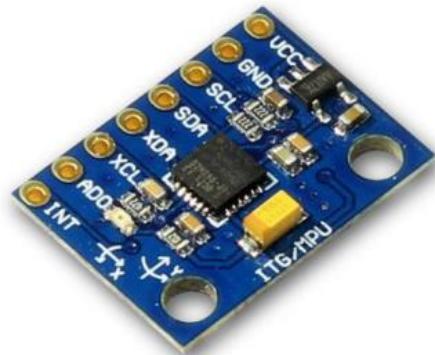
ארדואינו אונו – Arduino Uno



[7] Arduino Uno - 79

חישון תואכות – MPU 6050

- **I2C Digital-output of 6 or 9-axis Motion Fusion data in rotation matrix, quaternion, Euler Angle, or raw data format.**
- **Input Voltage:** 2.3 - 3.4V
- **Module dimensions:** 20x15mm (L x W).
- **price:** 5 NIS



[8] MPU6050 – 80

טבלה 1 – השוואת מנועים למערכת

XD-37GB555-600		JGB37-520 DC		20Dx41L		מנוע קריטריון
ציון נתון	ציון crit	ציון נתון	ציון crit	ציון נתון	_crit	
2/5	300	3/5	140	4/5	47	משקל (g)
1/5	12V	5/5	6V	5/5	6V	מתח עבודה (V)
3/5	1	2/5	0.4	5/5	1.6	מומנט (kg*cm)
3/5	90	5/5	60	1/5	169	מחיר (NIS)
9/20		15/20		15/20		ציון כולל

. לאחר בוחנה הבחירה והשלישית התקציבי, הוחלט לבחור במנוע JGB37-520 DC.



איור 82 - מנוע
[10]20Dx41L

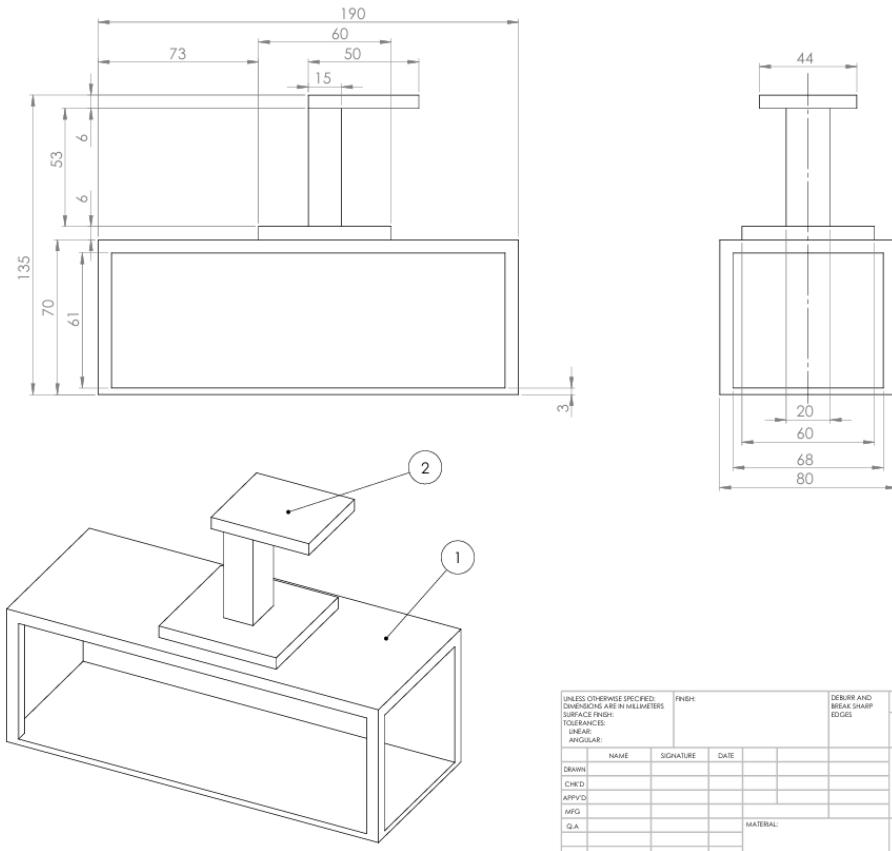


[9] XD-37GB555-600-20Dx41L



איור 80 - מנוע 80 DC [8]

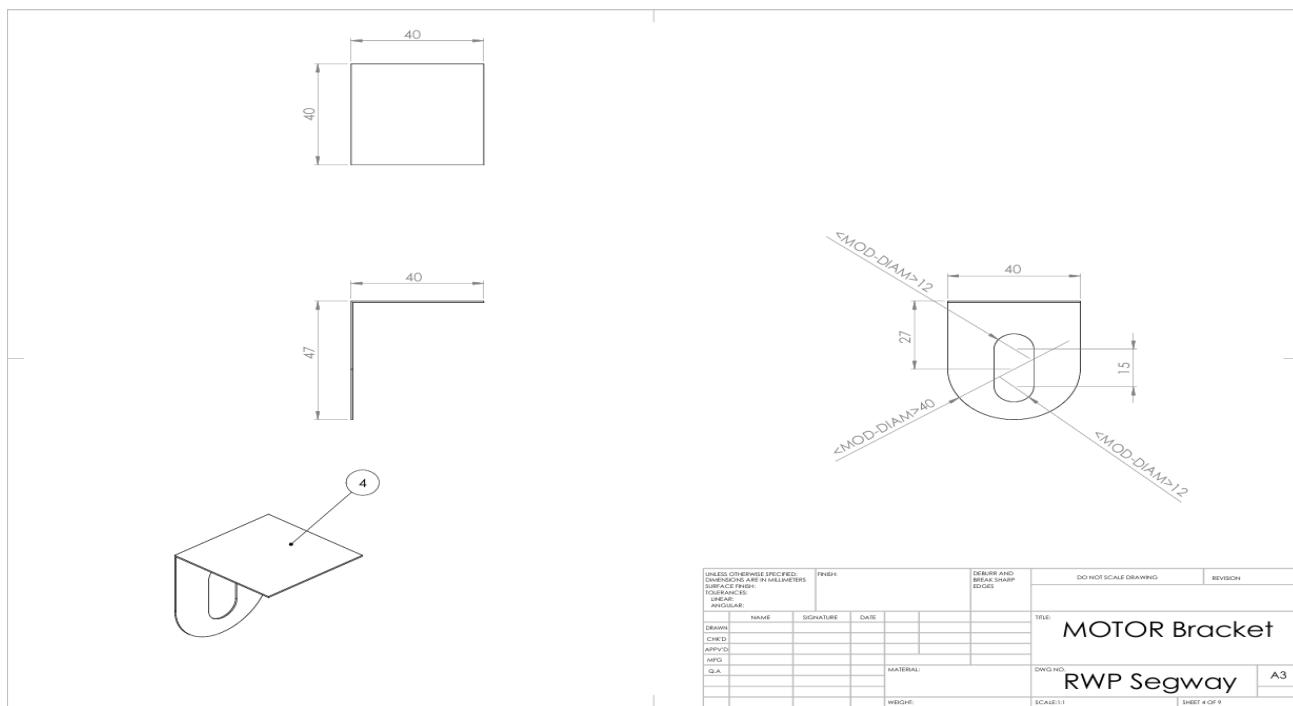
8.4 שרטוטי תכנון המערכת – רכיבים לא סטנדרטים



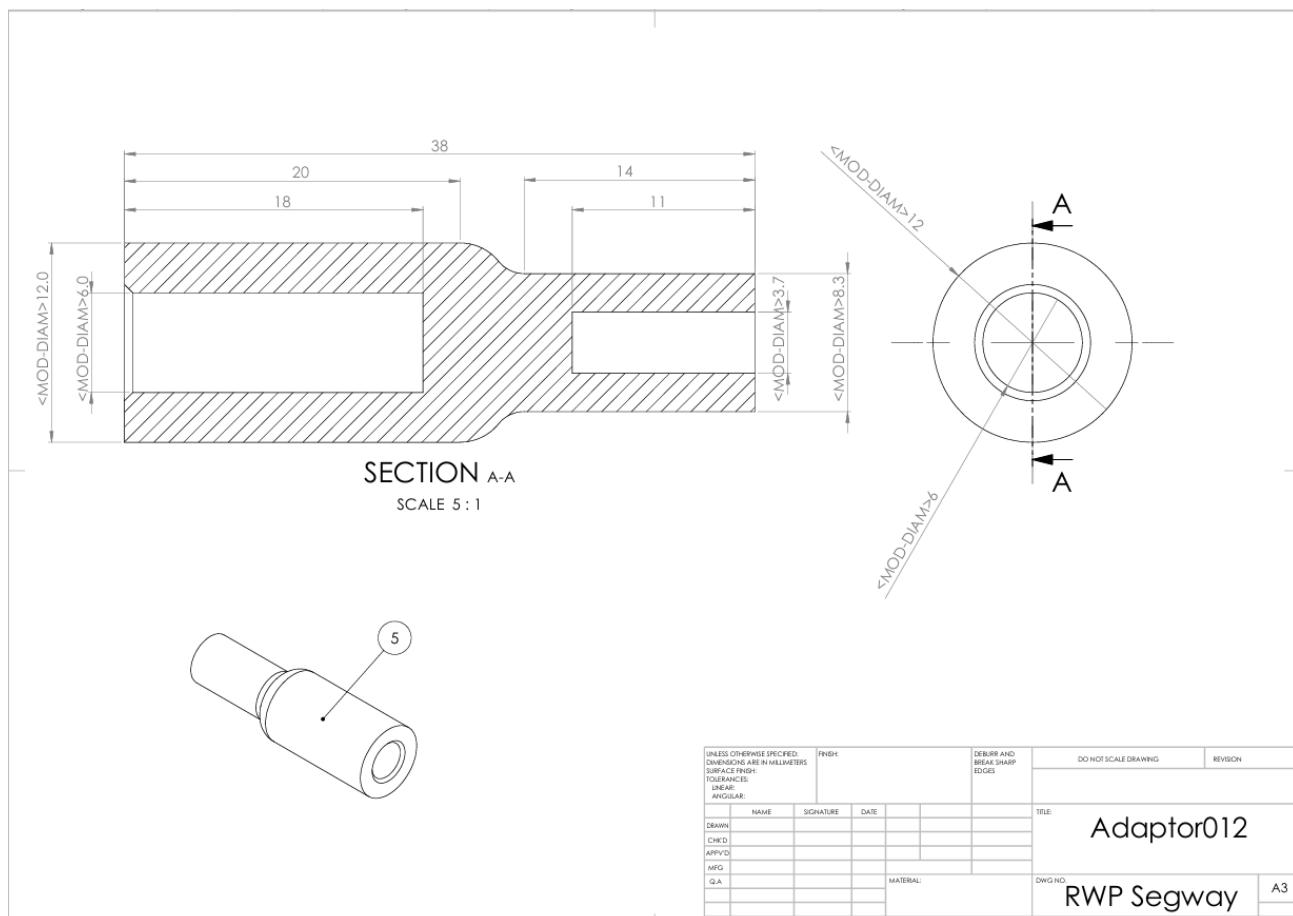
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DEBUR AND MILL SHARP EDGES:	DO NOT SCALE DRAWING	REVISION
DRAWN	NAME	SIGNATURE	DATE	
CHEK'D				
APPV'D				
MFG				
QA			MATERIAL:	DWG NO.
			WEIGHT:	SCALE:1:2
				A3
				SHEET 3 OF 9

TITLE: chassis
DWG NO: RWP Segway

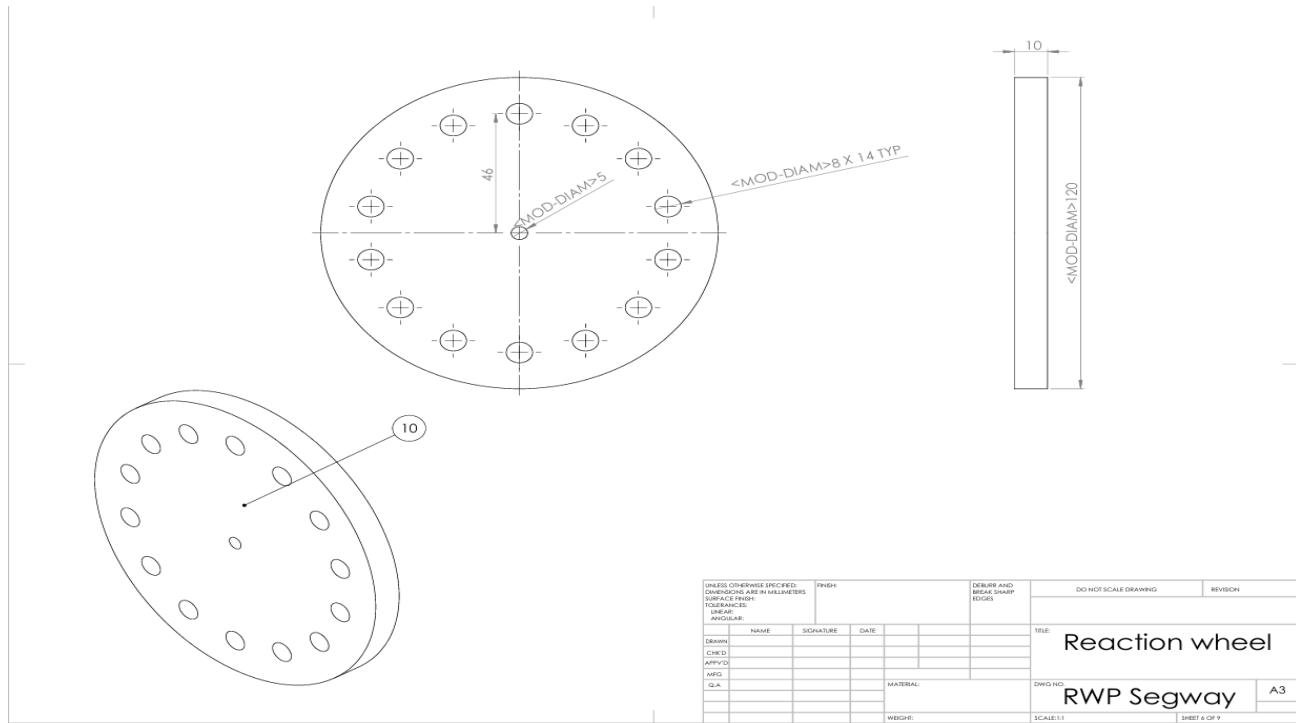
איור 83 – שרטוט שילדה



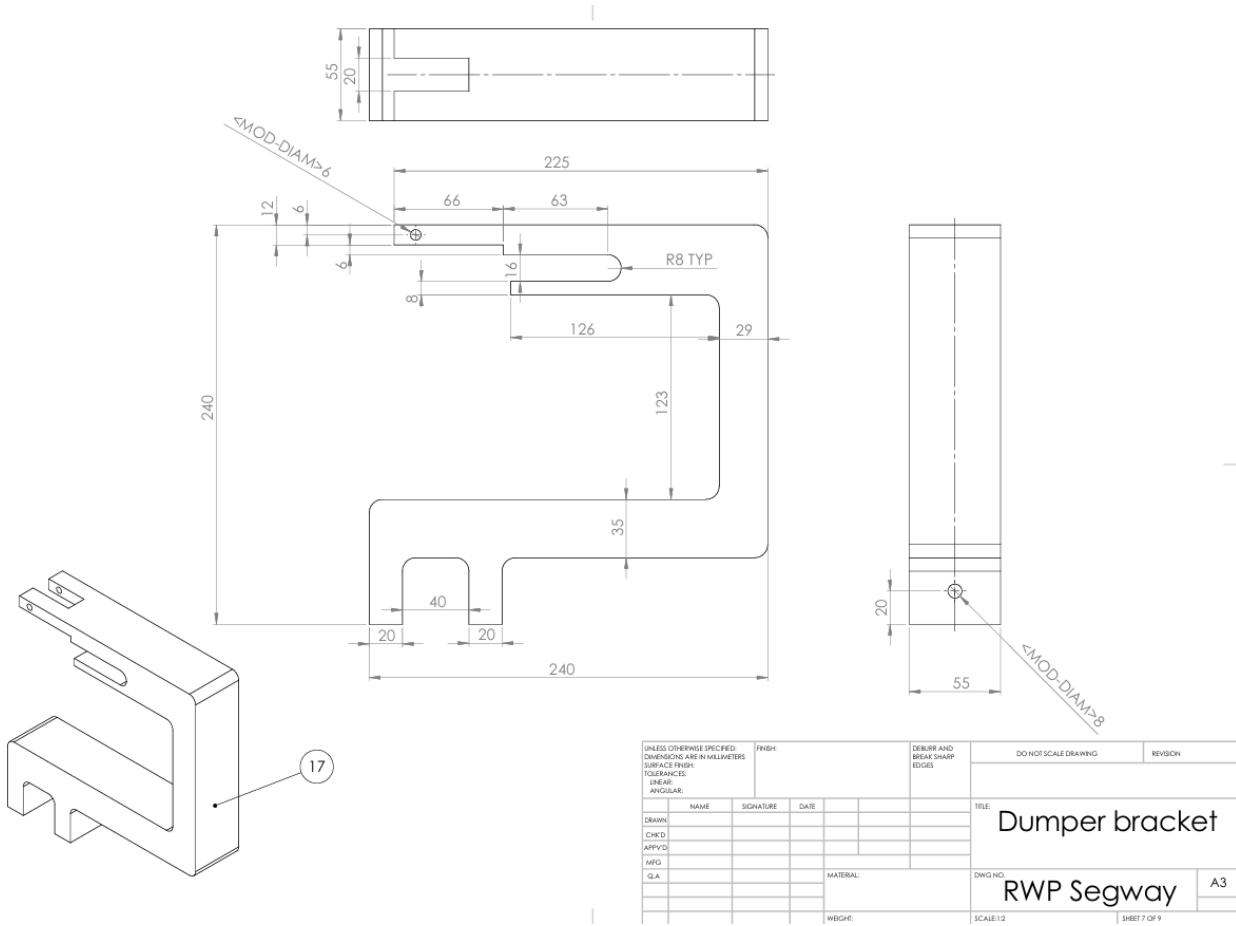
איור 84 –شرطוט תושבת מנוע



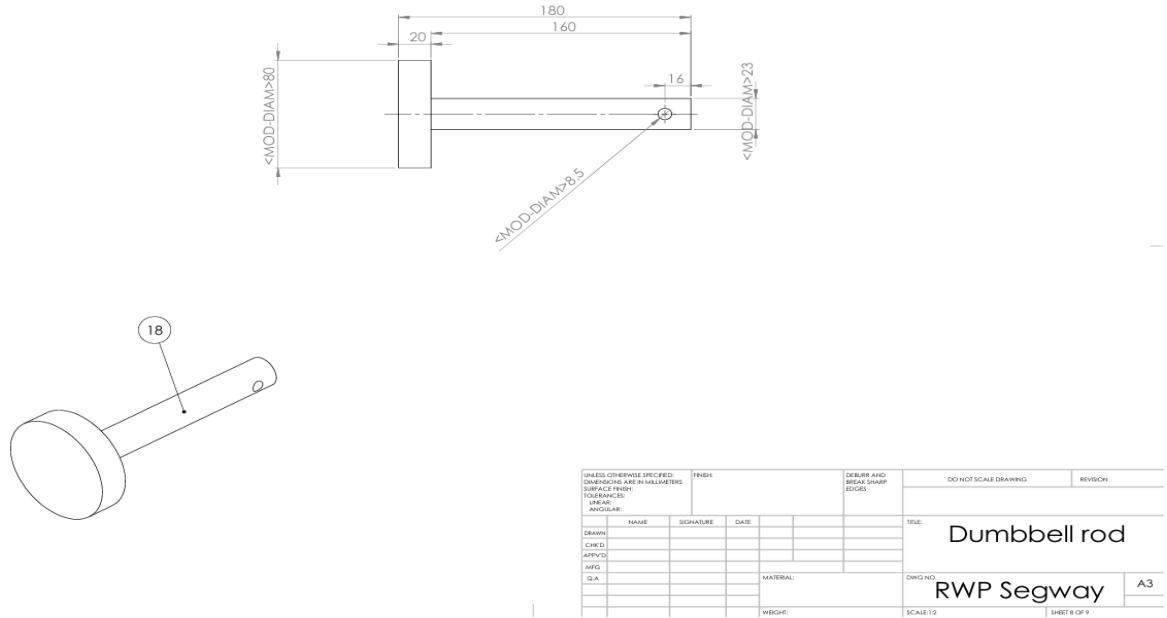
איור 85 –شرطוט מתאים מנוע על גלגל



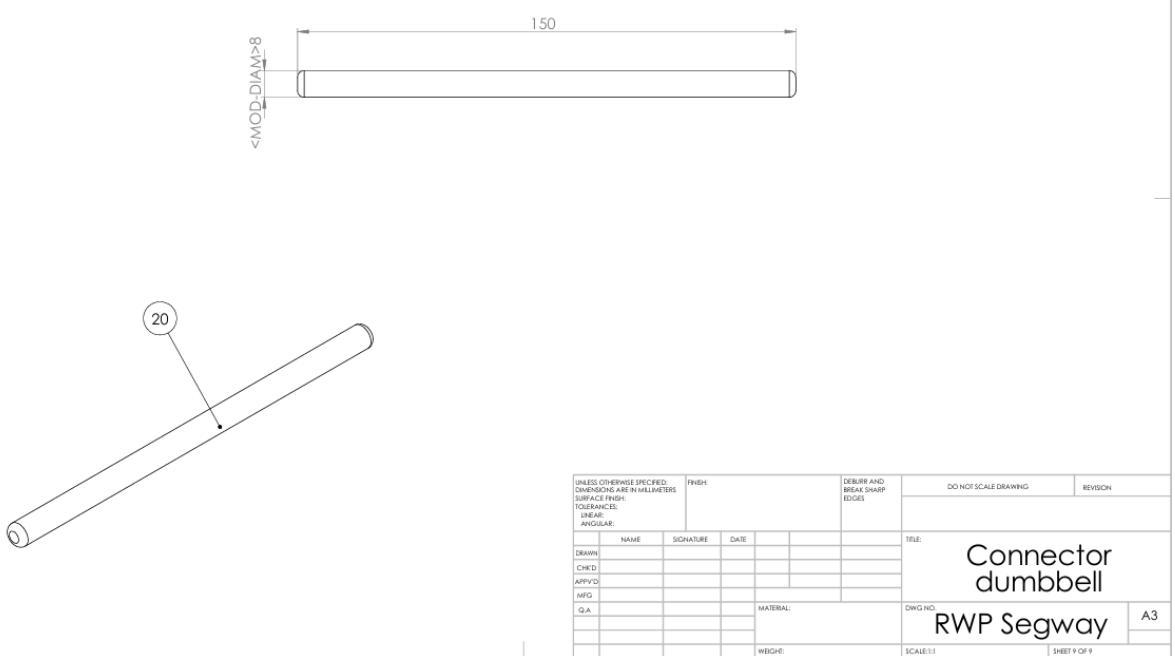
איור 86 – שרטוט גלגל תגובה



איור 87 – שרטוט תפסנית למשקולת



איור 88 שרטוט בסיס משקלות



איור 89 – שרטוט מחבר תפסנית לבסיס המשקלות

Abstract

This project focuses on the design and construction of a prototype for a two-wheeled "Segway"-type robot, offering an innovative approach to stabilization. Unlike traditional systems that rely solely on motors in the lower wheels, the aim of this project is to stabilize the robot using a third reaction wheel, located above the axis of rotation of the lower wheels. Additionally, the robot was designed to detect objects using image processing and respond accordingly—either to continue moving or to stop.

Throughout the project, six comprehensive experiments were conducted to characterize and optimize the system. The first experiment focused on stabilizing the system using only the reaction wheel, without involving additional motors. This experiment was unsuccessful due to insufficient torque from the reaction wheel, leading to the exploration of alternative solutions.

In the second experiment, an attempt was made to stabilize the system by lowering the center of gravity, using a weight. This experiment was performed without motor involvement, relying only on an accelerometer to measure spatial changes. However, the test failed to achieve stability around the desired angle.

The third experiment combined the damping mass system with the stabilization mechanism of the reaction wheel. However, this test also failed, increasing the deviation from the desired angle compared to the previous experiment.

A significant breakthrough occurred during the fourth experiment, where it was decided to remove the damping mass and use the system's lower motors in conjunction with the torque generated by the reaction wheel. This experiment was successful, with the system able to stabilize around the desired angle over time. The fifth experiment, which tested the system using only the lower motors, emphasized the importance of the reaction wheel for efficient stabilization.

Finally, the sixth and decisive experiment combined the most successful principle from the previous tests with image processing capability. The system was able to move while maintaining stability, based on data from the image processing, thus fully achieving the project's objective: the development of a two-wheeled robot that stabilizes using a reaction wheel and can move while utilizing image processing, all while maintaining balance.

A Two-Wheeled Robot Stabilized by a Reaction Wheel and Capable of Movement via Image Processing

Final Engineering Project

Prepared as part of the requirements for the degree of Bachelor of Science (B.Sc.) in Engineering

ME - 12

By

Alex Raz

Shai Shavit

Supervisor

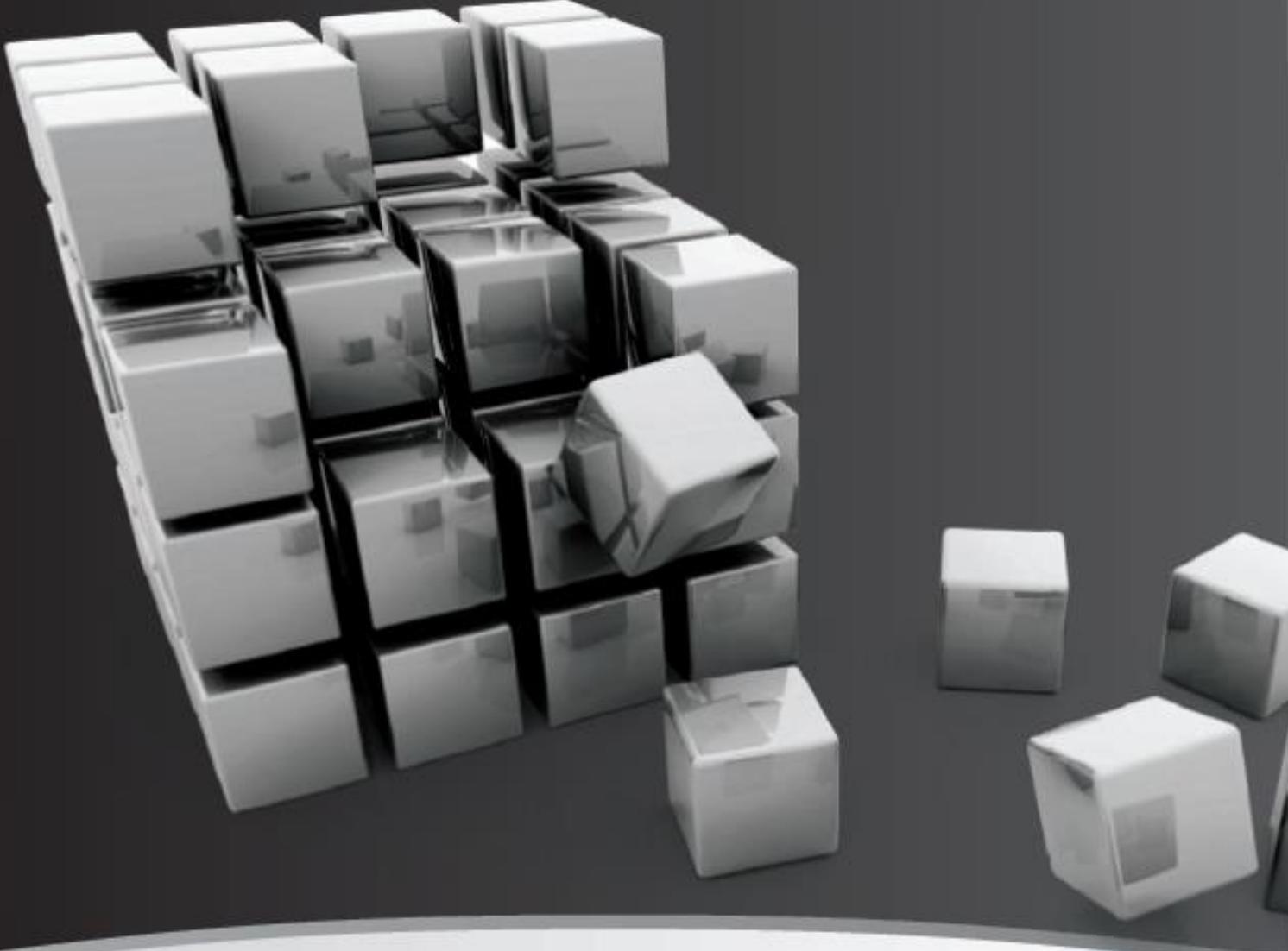
Dr. Chen Giladi

Submitted to the Department of Mechanical Engineering

The Academic College of Engineering - Sami Shamoon

Ashdod Campus

October 2024



A Two-Wheeled Robot Stabilized by a Reaction Wheel and Capable of Movement via Image Processing

Final Engineering Project

ME - 12

Prepared as part of the requirements for the degree of Bachelor of Science (B.Sc.) in Engineering
By

Alex Raz

Shai Shavit

Dr. Chen Giladi

Submitted to the Department of Mechanical Engineering

The Academic College of Engineering - Sami Shamoon

October 2024

sce

המכללה האקדמית להנדסה סמי שמעון

תתחבר לרעיונות גדולים

באר שבע ■ אשדוד ■ *מתקדם ■ www.sce.ac.il