**דו"ח HomeKit – מעבדה מתקדמת למערכות אוטונומיות**

**רקע תיאורטי:**

משימה 3:

3.1 ביצענו החלפה של ערך LED\_BUILTIN לpin\_number אחר (כאשר 9-אדום 10-ירוק 11-כחול כפי שמוצג בגרף).

3.2 שינוי הזמנים נעשה על ידי שינוי הערכים תחת delay().

3.3 ביצענו הוספה של לדים והגדרת זמני ההדלקה והכיבוי שלהם כאמור לעיל.

משימה 4:

**Serial Communication & Potentiometer**

עדכנו את הקוד כמתואר וכן השתמשנו בפונקציית הmap לטובת השילוב עם הפוטנציומטר לערכים הדרושים. כמו כן, נעזרנו במקור המצורף לטובת הקוד[[1]](#footnote-1)

**DC Motor**

* מימשנו את הקוד הדרוש לביצוע התנועה למשך 5 שניות, עצירה ל5 שניות ולאחר מכן תנועה בכיוון השני.
* מימשנו את קוד פעולת המנוע ע"פ הפוטנציומטר וכן שילוב הלד כחלק מתנועת המנוע (בקוד משולב).

**Distance sensor - VL53L4CD**

* מימשנו את הקוד למדידת מהירות המנוע כתוצאה ממרחק הסנסור מהמטרה.
* בחלק השני נעזרנו בChatGPT לטובת עריכת הקוד של דוגמה 6 לכך שיהיה non-blocking ויחליף את לולאת הwhile בתנאי if

**Magnetic Encoder - Interrupts**

* ראשית, נשים לב כי בשלב הSETUP הקוד מגדיר שני interrupts ככאלו המחוברים לפינים 2,3 – כך שאם מתרחש שינוי כלשהו בהם, אזי מופעלות הפונקציות encoderA,B בהתאמה. (ISR Functions)
* gear ratio חישבנו באמצעות חישוב ידני של כמות הספירות במחזור סיבוב אחד   
  לפי הנוסחה:   
  קיבלנו ספירה ידנית של 618 במחזור אחד ולכן הgear-ratio הינו 51.5 (הגיוני, שכן ראינו על הלוח שהgear ratio = 50)
* בקוד שמימשנו ביצענו עריכה של הקוד המקורי לטובת ביצוע שלושת המשימות:
  + עריכת הקוד כך שיציג את מספר הסיבובים שמבצע המנוע (מופיע בcomment) וחושב ע"י מספר הטיקים שמבצע כל סיבוב
  + שימוש בפונקציית millis() כך שנוכל למדוד את הRPM
  + שילוב הפוטנציומטר כך שהוא ישלוט בתנועת המנוע – נשים לב שתנועה "שמאלה" מהמרכז מבחינתנו הינה RPM שלילי (שכן מסובב נגד כיוון השעו) וימינה RPM חיובי. כמובן שעבור ערך אבסולוטי ניתן לתחום עם abs().

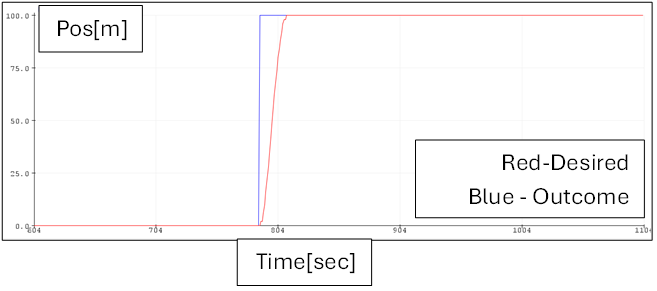
**Arduino serial parser**

* מימשנו את הקוד המתקדם המאפשר למשתמש להזין כקלט שלוש מחרוזות המופרדות ע"י delimeter כלשהו (בדוגמה שלנו זה ',') ובהתאם לגרום לרמת בהירות הלדים. (כמו כן, נציין שאם נרצה להרחיב את הקוד כך שימיר תו כלשהו באלפבית למספרי יכולנו להוסיף פונקציית תרגום לAscii**,** אך להבנתנו לא היה צורך לממש זאת, אלא להתייחס לקלט מספרי).

**Close loop implementations**

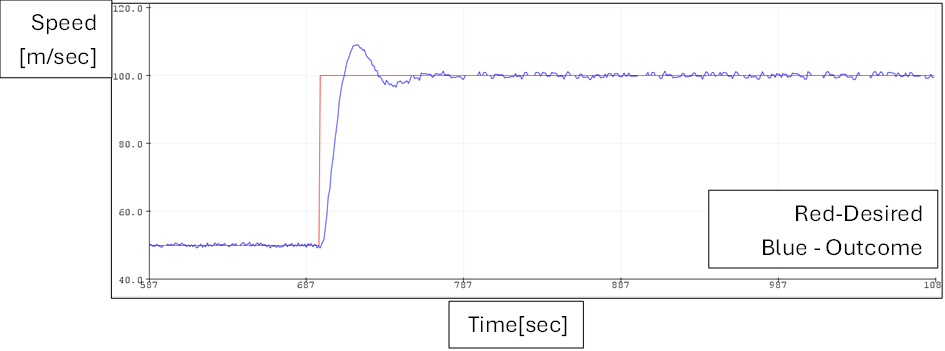
Position control close loop

* מימשנו קוד אשר מממש בקר PI כך שיסגור את החוג וכן לתגובת מדרגה. הבקר ממומש בתדר של 100Hz (לפיו בחרנו את המשתנה diff וכן את קצב הdelay).

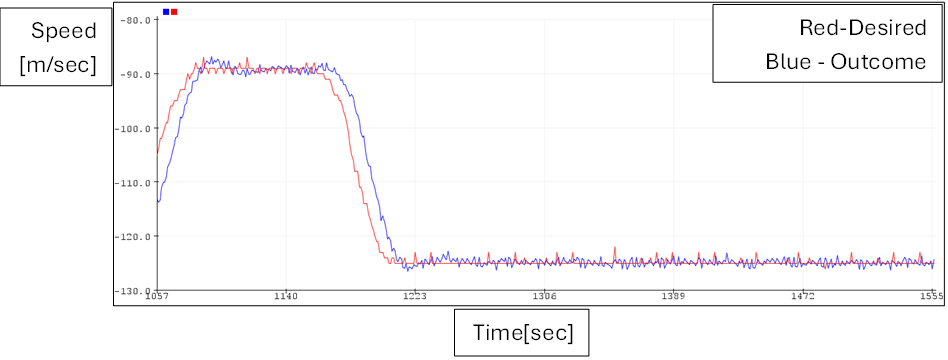


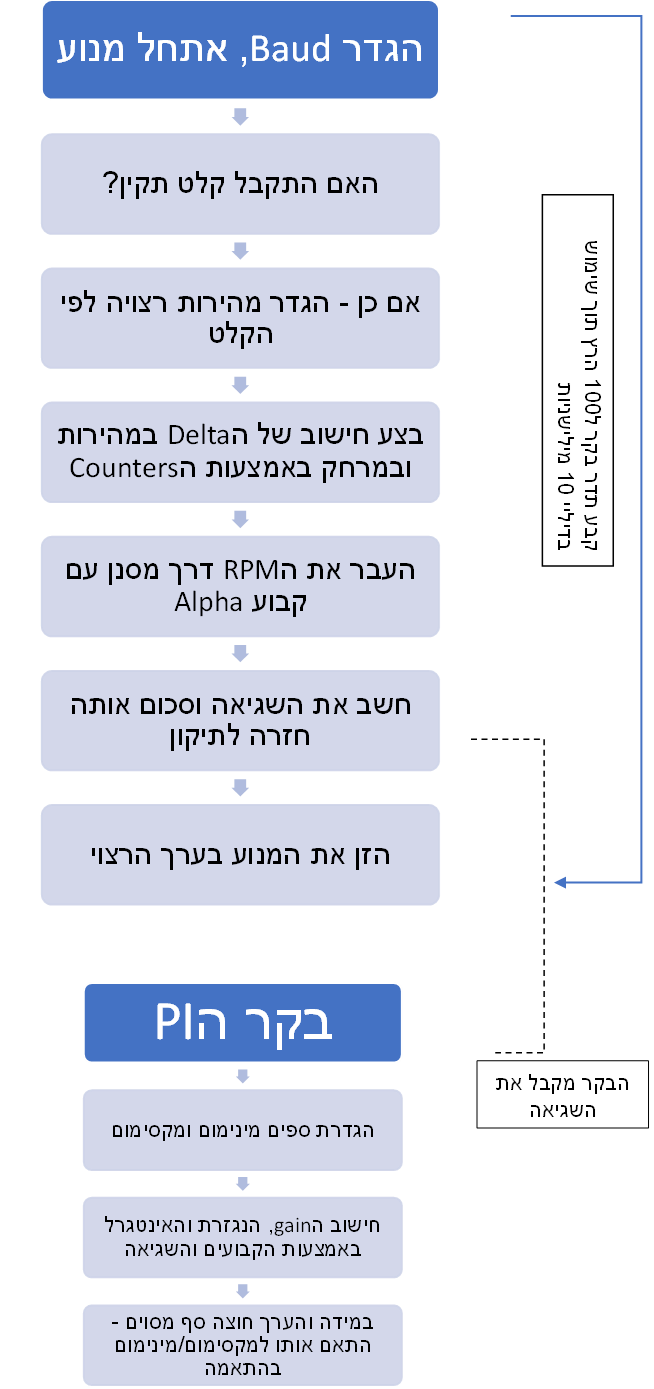
Velocity control close loop

* מימשנו קוד אשר מממש בקר PI כך שיסגור את החוג וכן לתגובת מדרגה. הבקר ממומש בתדר של 100Hz (לפיו בחרנו את המשתנה diff וכן את קצב הdelay).כמו כן, כמתואר בקוד, את הערכים של Ki, Kp בחרנו תוך הרצת סימולציות שונות לבחירת מהירות תגובה, OS וזמני התייצבות אפקטיביים עבורנו. כמו כן, גם מקדם הפלטור (Alpha) נבחר באופן דומה, על מנת למנוע "חיתוך" מהיר בין מעברי מצב.בנוסף, הגדרנו מהירות מקסימלית ומינימלית כדי לא לחרוג מהסף האפשרי (ע"פ מדידות שלנו)

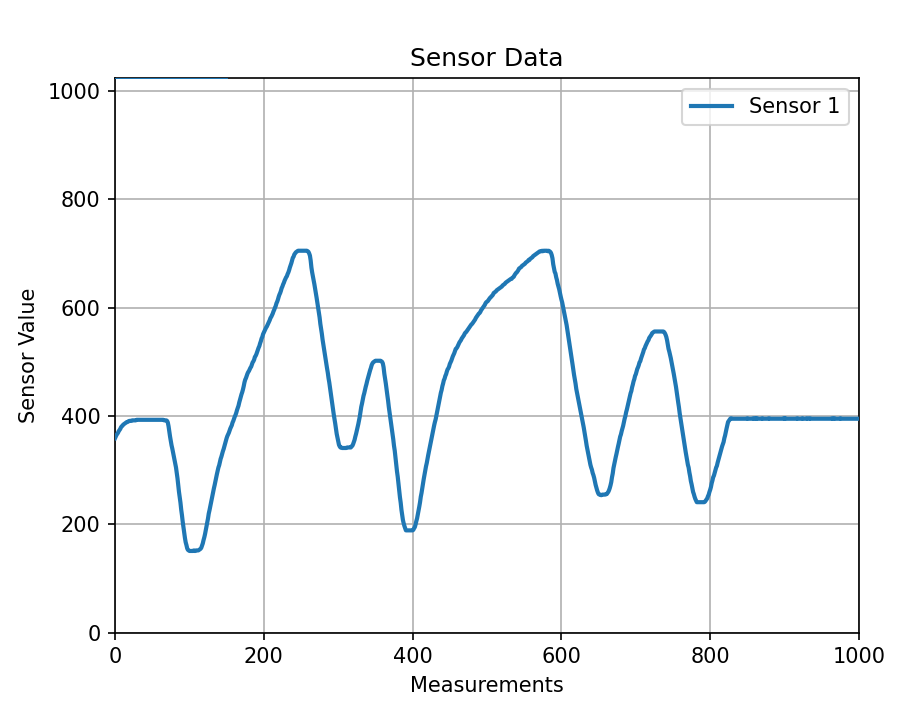


Potentiometer / Distance sensor feedback velocity control

* בדוגמה זו, ביצענו דבר זהה לקוד הקודם, למעט כך שהגדרנו שאת הקלט המשתמש יבצע מהפוטנציומטר וכן הגבלנו את הRPM אליו ניתן להגיע תוך התייחסות לתזוזת הפוטנציומטר בהתאם.

דיאגרמת בלוקים:

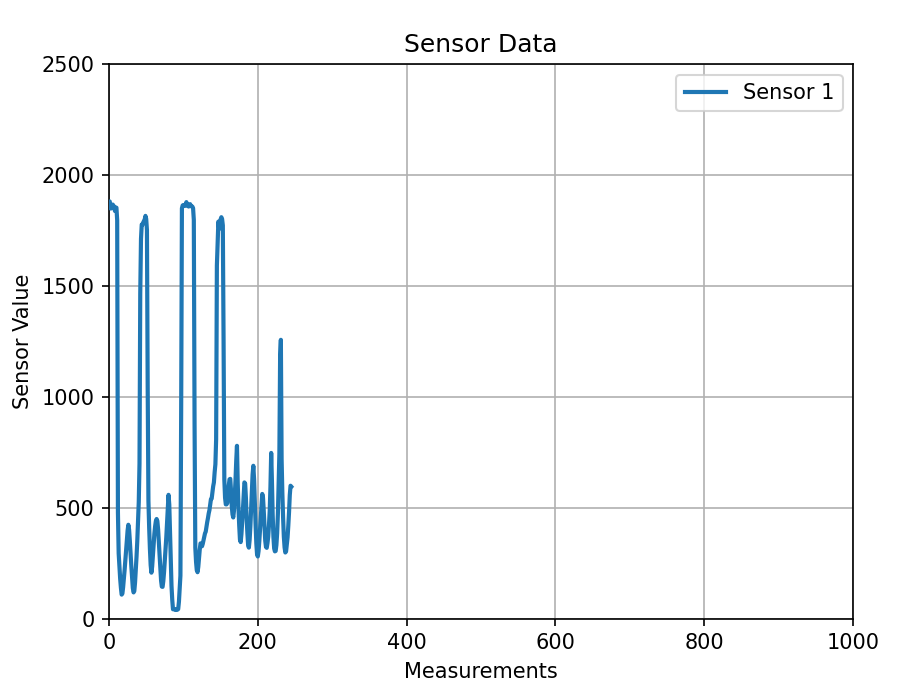
**Python arduino communication**

* שילבנו את קטעי הקוד לפי ההנחיות המצורפות וקיבלנו פלט של תזוזת הפוטנציומטר:
* ראשית, נסביר כיצדקטעי הקוד עובדים - סקריפט הPython הנתון יוצר תקשורת סדרתית עם לוח הArduino (עם קצב העברת נתונים של 115200 לפי הBaud), מקבל נתונים, שומר אותם בקובץ ומשרטט את הנתונים בזמן אמת באמצעות Matplotlib.

בפונקציה הראשית נוצרות מחלקות ArduinoCommunication ו-SensorDataPlot (ביטוי ה- With מבטיח שהפונקציה close(self), נקראת בסוף הסקריפט).

ה- ArduinoCommunication מטופל בחוט שבודק באופן רציף את הSerial עבור הודעות וכותב אותם לקובץ. פונקציית הdata\_callback מוכרזת כשיטת handle\_data של הפלוטר, אשר גורם לכך שלאחר כל הודעה המתקבלת מהSerial, פונקציית הhandle\_data נקראת עם הנתונים שקיימים.פונקציית הhandle\_data של הפלוטר, מקבלת ערך מהמילון "data" (בדוגמה שניתנה, המפתח הוא "סנסור 1"), ולאחר מכן מוסיף אותו לסוג "Deque" (מבנה דמוי רשימה). הפונקציה update\_plot נקראת שוב ושוב על ידי animation.FuncAnimation(), אשר מעדכנת את נתוני צירי ה-X וה-Y**.**

* דיאגרמת בלוקים:
* על מנת להוציא כפלט את המרחק מהסנסור, ביצענו שילוב של הקוד מהחלק של הdistance\_sensor וקיבלנו את הרצוי. כמו כן, ביצענו התאמות לערך הY המקסימלי בפלט הPython שיהיה כ-2500. מצורף הפלט:



**Appendix**

1. https://arduinogetstarted.com/tutorials/arduino-potentiometer-fade-led [↑](#footnote-ref-1)