בן כהן, 208685784
אורי אברהמי, 315416057

דו"ח HomeKit – מעבדה מתקדמת למערכות אוטונומיות

משימה 3 :

3.1 ביצענו החלפה של ערך LED_BUILTIN ל pin_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

בקובץ Fade.ino עדכנו את הקוד כמתואר וכן השתמשנו בפונקציית הmap לטובת השילוב עם הפוטנציומטר לערכים הדרושים. כמו כן, נעזרנו במקור המצורף לטובת הקוד[1]
כך שלמעשה באמצעות פונקציית map ביצענו את הדרוש בקוד.

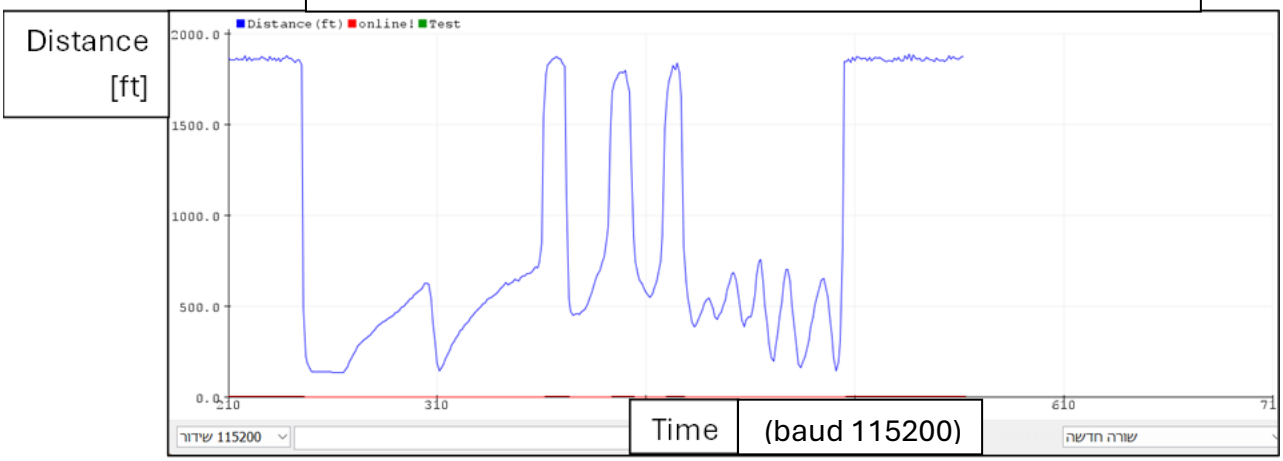
DC Motor

- בקובץ Motor_basics.ino מימשנו את הקוד הדרוש לביצוע התנועה למשך 5 שניות, עצירה ל5 שניות ולאחר מכן תנועה בכיוון השני. לטובת מימוש זה, נעזרנו בדיאגרמת הבלוקים של הKit אשר מופיע בקובץ ההנחיות כדי להתאים את הכניסות IN1, IN2 לפינים בהתאמה. כמו כן, על מנת לבצע את התנועות לכל כיוון השתמשנו בטבלה שצורפה וכן את הdelay באמצעות (5000)delay.
- בקובץ motor_by_potentiometer מימשנו את קוד פעולת המנוע ע"פ הפוטנציומטר וכן שילוב הלד כחלק מתנועת המנוע (בקוד משולב). על מנת להשתמש בפוטנציומטר ביצענו שלושה תנאים של if כך שאם הפוטנציומטר נמצא בחצי כלשהו אזי תנועת המנוע תסתובב לצד אחד ובהתאמה לצד האחר וכמו כן, עבור המרכז – ללא תנועה. כמו כן, השתמשנו בפין הled מס' 9 על מנת לעדכן בהתאם גם את הלד.

Distance sensor - VL53L4CD

- בקובץ distance_sensor.ino מימשנו את הקוד למדידת מהירות המנוע כתוצאה ממרחק הסנסור מהמטרה. השתמשנו בספרייה "SparkFun_VL53L1X.h" כמתואר בקובץ ההנחיות ואכן קיבלנו תלות בין המרחק לבין הפלט. נציג זאת גם בגרף של הפלוטר :

תרשים פונקציית מהירות המנוע כתוצאה ממרחק הסנסור מהמטרה



[1] https://arduinogetstarted.com/tutorials/arduino-potentiometer-fade-led

בן כהן, 208685784
אורי אברהמי, 315416057

- בחלק השני בקוד בשם non_blocking_distance_sensor.ino 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: $CPR = gear - ratio * 12$
  קיבלנו ספירה ידנית של 618 במחזור אחד ולכן הgear-ratio הינו 51.5 (הגיוני, שכן ראינו על הלוח שה50 = gear ratio)
- בקוד שמימשנו בשם Magnetic_Encoder.ino 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 השעון) וימינה RPM חיובי. כמובן שעבור ערך אבסולוטי ניתן לתחום עם abs().

Arduino serial parser

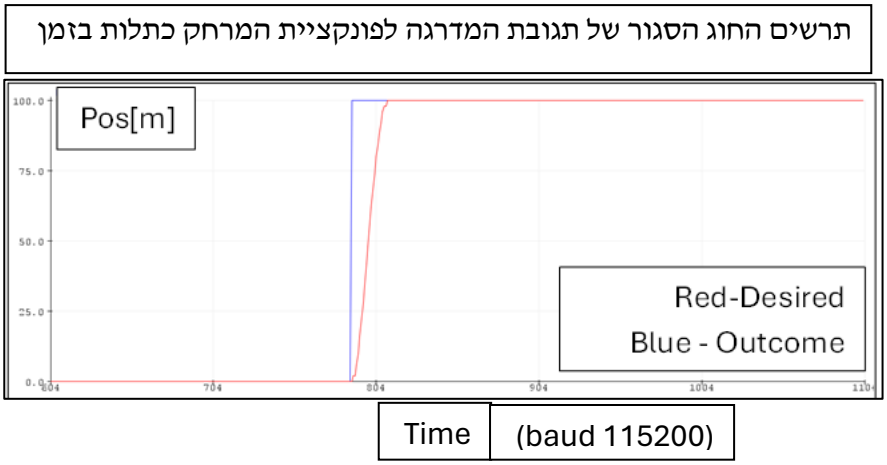
- מימשנו את הקוד המתקדם בשם serial_parser.ino המאפשר למשתמש להזין כקלט שלוש מחרוזות המופרדות ע"י delimeter כלשהו (בדוגמה שלנו זה ','(ובהתאם לגרום לרמת בהירות הלדים. (כמו כן, נציין שאם נרצה להרחיב את הקוד כך שימיר תו כלשהו באלפבית למספרי יכולנו להוסיף פונקציית תרגום לAscii, אך להבנתנו לא היה צורך לממש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

Close loop implementations

<u>Position control close loop</u>

- מימשנו קוד בשם closed_loop_location.ino 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 בחרנו Kp, Ki = 1.0 וקיבלנו תגובה רצויה הן מבחינת OS והן מבחינת זמן התייצבות כמופיע בגרפים ובקוד.

בן כהן, 208685784
אורי אברהמי, 315416057

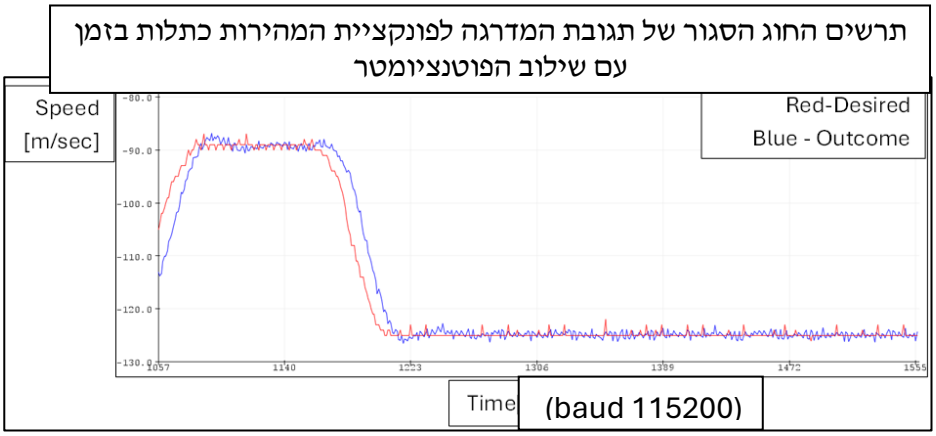


Velocity control close loop

- מימשנו קוד בשם close_loop_velocity.ino 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) וכדי למנוע מצב שהקלט הרצוי לא יסופק לעולם.

Potentiometer / Distance sensor feedback velocity control

- בדוגמה זו, ביצענו דבר זהה לקוד הקודם, למעט כך שהגדרנו שאת הקלט המשתמש יבצע מהפוטנציומטר (A0).



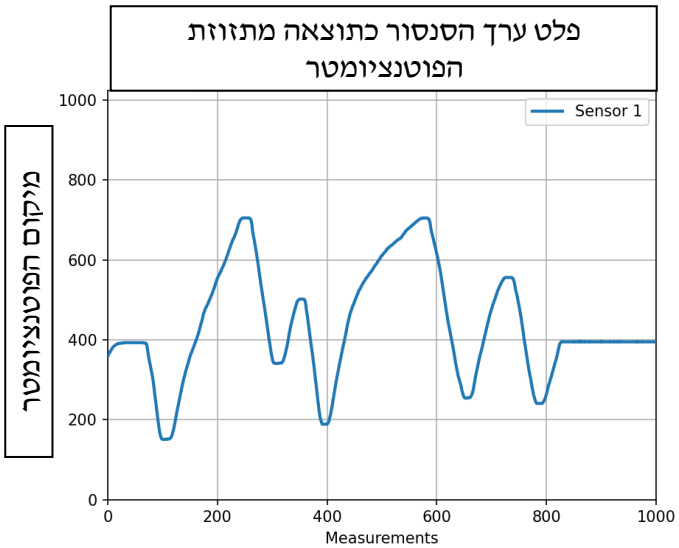
בן כהן, 208685784
אורי אברהמי, 315416057

<u>דיאגרמת בלוקים:</u>

בן כהן, 208685784
אורי אברהמי, 315416057

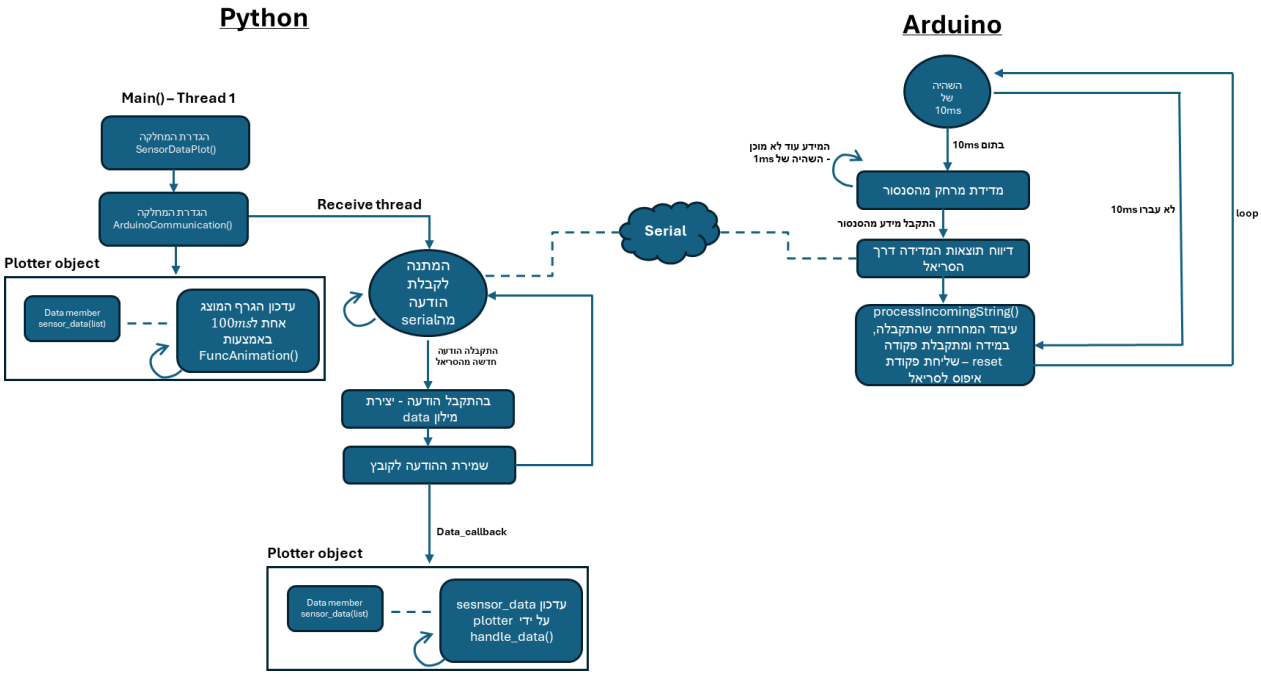
Python arduino communication

- שילבנו את קטעי הקוד (python_arduino.ino, main.py) לפי ההנחיות המצורפות וקיבלנו פלט של תזוזת הפוטנציומטר:



- ראשית, נסביר כיצד קטעי 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:



בן כהן, 208685784
אורי אברהמי, 315416057

- על מנת להוציא כפלט את המרחק מהסנסור, ביצענו שילוב של הקוד מהחלק של הdistance_sensor וקיבלנו את הרצוי. זאת נעשה בקבצים:
python_arduino_distance, main.py כמו כן, ביצענו התאמות לערך הY המקסימלי בפלט הPython שיהיה כ-2500. מצורף הפלט: