הטכניון – מכון טכנולוגי לישראל
הפקולטה להנדסת מכונות

רובוט לסיוע בחקלאות

חלק ב'

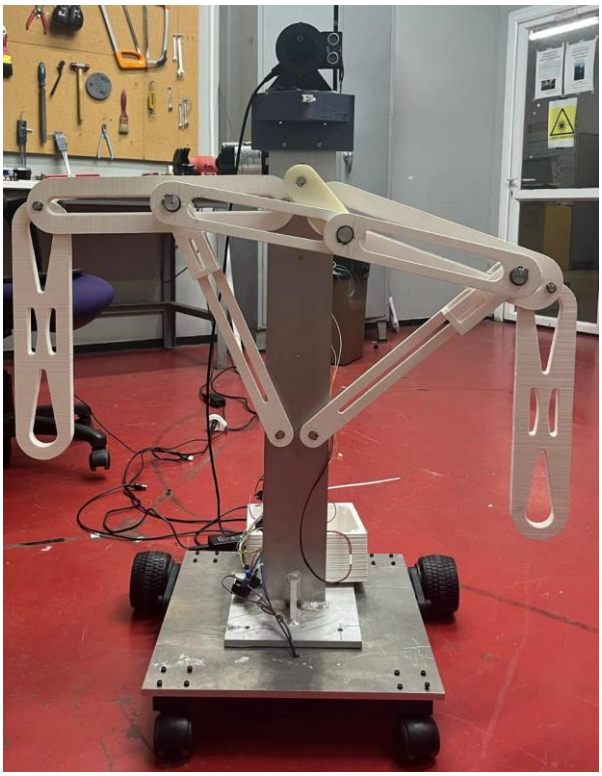
מגישים:

איתן מוסייב - 206215550

eitanmu@campus.technion.il

ראובן רוסטם שחפזוב - 314568593

rechahpazov@campus.technion.il



תוכן עניינים:

1.הקדמה:

בפרויקט עבדנו על רובוט מסוג "דחליל" שמטרתו לסייע לחקלאים להגן על היבול שלהם מפני בעלי כנף ומזיקים.

בעלי כנף יכולים לגרום לנזק בחקלאות באופן משמעותי. פעילותם המזיקה כוללת אכילת התוצר החקלאי, הם עשויים לפגוע בצמחים על ידי קיטוע עלים, גרירת שורשים והעברת מחלות מצמח לצמח. כל אלה ועוד מובילים להפסדים ביצורי המזון ולהפחתת הביצועים החקלאיים. ניהול הנזק של בעלי כנף בחקלאות חיוני על מנת להפחית את ההשפעה השלילית שלהם על ענפי החקלאות ועל ייצור המזון. הרובוט שלנו ידע לזהות את בעלי הכנף ולבצע תנועות שיבריחו אותם, במידת הצורך הרובוט יוכל לרדוף אחרי המזיקים עד שלא יהבו איום על היבול.

בחלק הראשון של הפרויקט התעסקנו בתכנון מערכת ההנעה ובבדיקת קוד עיבוד התמונה. וכעט, בחלק השני, הרכבנו את המערכת עם מעט שינויים שבצענו בתכן ובמערכת ההנעה, המשכנו בפיתוח מערכת עיבוד התמונה, וצרבנו קוד לארדואינו לניהול הבקרה על המערכת, התעסקנו בחיווט ובהשלמת כתיבת הקוד שגורם לדחליל לעמוד במשימה. ובנוסף, חיברנו את הראספברי פאי, מצלמה סטריאו מדגם Intel Realsense L515, וחיישן אולטרסוני למדידת המרחק.

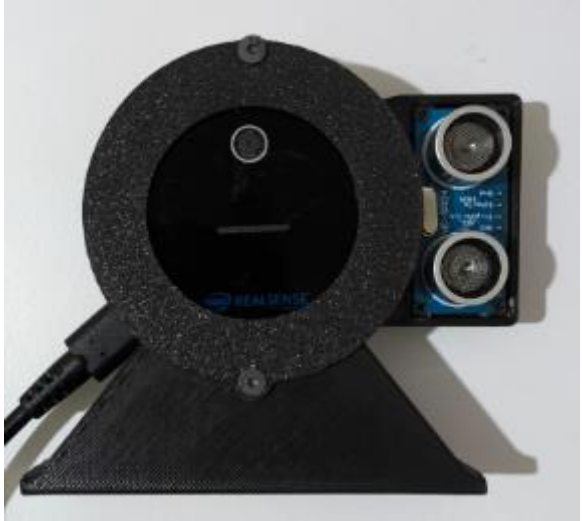
2.הרכבת הרובוט והשינויים במערכת ההנעה

2.1 השינויים שביצענו במערכת ההנעה והרכבתם:

מטרת המערכת היא לאפשר את תנועת הרובוט במרחב כך שהרובוט יוכל לעקוב ולרדוף אחר מזיקים עד להברחתם, מערכת זו נשלטת על ידי חוג הבקרה. למערכת שני מנועים בחלק האחורי של המרכב ושני גלגלים פסיביים על מנת שנוכל לשלוט בהיגוי הדחליל בצורה טובה וחלקה יותר. לצורך הבקרה, חיברנו שני בקרי מהירות למנועים, שאחראים על ניהול אספקת המתח למנועים בהתאם לפקודות הבקרה.

2.2 חיבור המצלמה וחיישן אולטרסוני:

התקנו את המצלמה על הראש של הרובוט וחיברנו אותה למחשב באמצעות כבל USB, תוך שימוש במצלמת Intel Realsense L515 שמאפשרת זיהוי עצמים בטווח של עד 6 מטרים בדיוק גבוה.
בנוסף, חיברנו חיישן אולטרסוני על מנת לאמוד מרחקים ולספק עוד נתונים לחוג הבקרה.



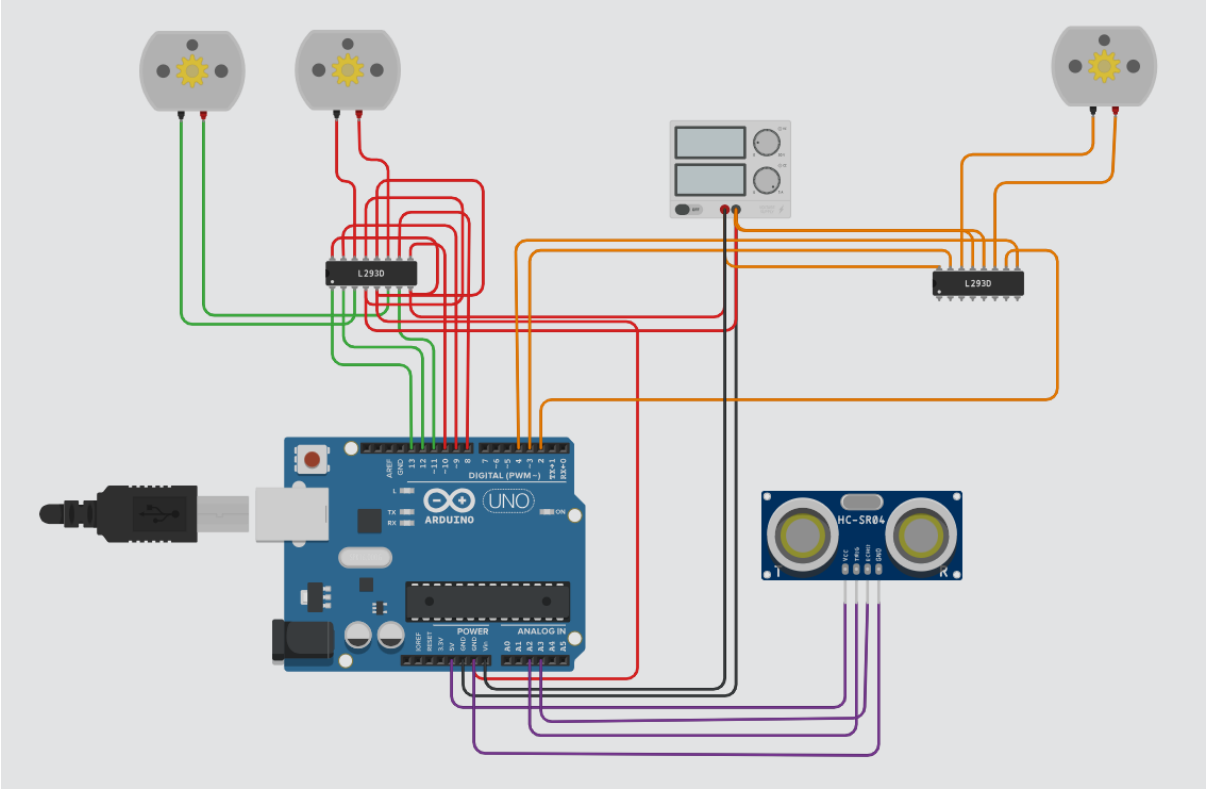
איור 1: המצלמה וחיישן מרחק מימין.

באיור 1 ניתן לראות jig שהודפס בהדפסת תלת מימד. ה jig תופס את המצלמה והחיישן בצורה שניתן יהיה למקם אותם על גבי הרובוט.

3.תכנות ובקרת מערכת

3.1 קוד הארדואינו:

דיאגרמת החיווט של הארדואינו למנועים, דרייברים והחיישן:



דיאגרמה זו מתארת את החיבורים של המנועים והחיישן לדרייברים ולארדואינו.

ניתן לראות שלושה מנועים, שני מנועים האחראים על תנועת הרובוט במרחב, מחוברים לשני גלגלים על ידי תמסורת גלגלי שיניים ורצועה ומנוע אחד מאחראי לתנועת הזרוע של הרובוט. שני מנועי התנועה מחוברים לדרייבר ומשם לארדואינו המנוע השלישי מחובר לדרייבר אחר וממנו לארדואינו.

קוד הארדואינו אחראי על שליטה במנועים של הרובוט, קבלת פקודות מהקוד בפייתון דרך תקשורת טורית (Serial), ומדידת מרחקים באמצעות חיישן אולטרסוני לצורך הימנעות ממכשולים.

1. הגדרת פינים:

בתחילת הקוד מוגדרים כל הפינים שבהם משתמשים לשליטה במנועים ולחיבור החיישן:

לדוגמה:

- IN1, IN2, ENA שליטה על מנוע הזרוע.
- IN5, IN6, ENC שליטה על מנוע קדמי-אחורי.
- TRIG_PIN, ECHO_PIN פינים לחיישן האולטרסוני.
- SAFE_DISTANCE = 50 מרחק סף לעצירה אוטומטית (ב־ס״מ).

2.פונקציית :setup()

מאתחלת את החיבורים הפיזיים:

- כל הפינים מוגדרים כ־ OUTPUT או INPUT לפי הצורך.
- מופעלת תקשורת סידרתית. (Serial.begin(9600))

3.פונקציית loop() :

- רצה כל הזמן ובודקת האם התקבלה פקודה מ־ Python דרך החיבור הסידרתי:

```
if (Serial.available()) {
    String command = Serial.readStringUntil('\n');
}
```

- אם מגיעה פקודה – מתבצעת פעולה בהתאם (קדימה, אחורה, סיבוב, עצירה).
- כל כמה שניות נמדד גם מרחק מהחיישן:

אם המרחק קטן מ SAFE_DISTANCE עצירה אוטומטית של הרובוט.

4. פונקציה getDistance() :

שיטת הפעולה:

1. שליחת פולס של 10 מיקרושניות ל־TRIG.
2. המתנה לקבלת ההד דרך ECHO.
3. חישוב זמן השהייה.
4. חישוב המרחק לפי מהירות הקול (340 מטר לשנייה):

```
distance_cm = (duration * 0.034) / 2;
```

5. אינטראקציה עם קוד הפייתון

- הקוד בפייתון שולח מחרוזות פקודה כמו:

  "FORWARD", "LEFT", "RIGHT", "STOP"

- הארדואינו מקבל את הפקודה, מפרש אותה ומפעיל את הפונקציה הרלוונטית.
- כל התקשורת מבוססת על USB/Serial.

3.2 קוד עיבוד התמונה בפייתון:

קוד הפייתון משמש כמרכז העיבוד החכם של המערכת. תפקידו לזהות מזיקים (כגון ציפורים) באמצעות מצלמה ומודל בינה מלאכותית מסוג YOLOv8, ולשלוח פקודות תנועה לרובוט בהתאם למיקום וסוג האובייקט שזוהה.

הסבר על הקוד:

1. חיבור למצלמה

- המצלמה מחוברת ל־ Raspberry Pi או מחשב.
- הקוד משתמש ב OpenCVכדי ללכוד פריימים מהמצלמה בזמן אמת.
- מוגדרת רזולוציית צילום של 640X480 פיקסלים:

```
cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
```

2. טעינת מודלYOLO

- הקוד טוען את מודל YOLOv8n דרך ספריית ultralytics.
- הקובץ yolov8n.pt הוא מודל מאומן שיכול לזהות סוגים שונים של עצמים.

```
from ultralytics import YOLO
model = YOLO("yolo-Weights/yolov8n.pt")
```

3. זיהוי אובייקטים בזמן אמת

- כל פריים מהמצלמה עובר ניתוח על ידי YOLO .
- המודל מחזיר את מיקום האובייקטים(bounding boxes).
- הקוד מחשב את מרכז האובייקט כדי לקבוע את כיוון התנועה של הרובוט.

4.קבלת החלטות תנועה

באופן כללי אלגוריתם קבלת החלטות מבצע את הדברים הבאים:

- אם האובייקט (למשל ציפור) נמצא במרכז התמונה – נשלחת פקודת "FORWARD" .
- אם האובייקט בצד ימין – נשלחת פקודת "RIGHT" .
- אם בצד שמאל – נשלחת פקודת "LEFT" .
- אם אין אובייקט – מופעל מצב סריקה (SCAN) שבו הרובוט מבצע סיבוב של 180° ימינה ושמאלה עד שיזהה מזיק.
- ברגע שהרובוט מגיע למרחק שהגדרנו מהמטרה הרובוט עוצר ומזיז את הידיים.

כעת נסביר את האלגוריתם בפירוט:

לאחר שזוהה אובייקט בתמונה באמצעות YOLO, הקוד בודק האם הוא אחד מהאובייקטים הרלוונטיים למשל ("bird") ואם רמת הביטחון בזיהוי (confidence) גבוהה מ־0.5.

```
if class_name in relevant_objects and confidence > 0.5:
```

חישוב מיקום האובייקט בציר X

הקוד מחשב את **מרכז התיבה (bounding box)** של האובייקט ואת מרכז הפריים (התמונה) בציר האופקי (X) :

```
center_x = (x1 + x2) // 2
frame_center_x = img.shape[1] // 2
```

קבלת החלטה לאיזה כיוון לזוז

לאחר מכן מבוצעת השוואה בין מיקום האובייקט לבין מרכז הפריים:

```
if center_x < frame_center_x - 50:
    detected_command = "LEFT"
elif center_x > frame_center_x + 50:
    detected_command = "RIGHT"
else:
    detected_command = "FORWARD"
```

- אם האובייקט נמצא ב־50 פיקסלים שמאלה ממרכז המסך פקודת "LEFT"
- אם הוא ב־50 פיקסלים ימינה "RIGHT"
- אם הוא במרכז המסך פחות או יותר "FORWARD"

5. שליחת פקודות לארדואינו

- הקוד מתחבר לארדואינו באמצעות serial.Serial.
- הפקודות נשלחות כטקסט בפורמט:

```
arduino.write((command + '\n').encode())
```

- הקוד בודק שהחיבור פעיל לפני שליחה, ומדפיס הודעת שגיאה אם יש בעיה.

4.סיכום

השלמנו את הרכבת הרובוט והצלחנו לשלב את רכיבי החומרה והתוכנה, ולפתח מערכת שמזהה מזיקים ומגיבה אליהם בזמן אמת. ביצענו בדיקות שונות שאישרו את פעולתו התקינה של הרובוט וראינו כי הרובוט מצליח לעקוב בצורה טובה וחלקה אחרי עצמים שאנו מגדירים לו ללא התערבותנו ובצורה עצמאית לגמרי. הרובוט מבצע את כל המשימות שהגדרנו: זיהוי, רדיפה עד להגעה למרחק שהגדרנו מהמטרה וההזזת הזרועות "להפחדה". הפרויקט משמש דוגמה לשימוש בטכנולוגיות חכמות לשיפור החקלאות ומניעת נזק ליבול ומאפשר פיתוח של מערכת מורכבת יותר המבוססת על העקרונות של מערכת זו. כמו כן אנו רואים אפשרויות רבות לשימוש ברובוט מסוג שכזה גם לאפליקציות אחרות כגון רובוט שמירה, פטרולים וכדומה.

נספחים:

קוד הפייתון:

```
from ultralytics import YOLO
import cv2
import math
import serial
import time

# התחברות לארדואינו
try:
    arduino = serial.Serial(port='COM9', baudrate=9600, timeout=.1)
    print("Connected to Arduino on COM8")
except Exception as e:
    print(f"Error connecting to Arduino: {e}")
    arduino = None

# פונקציה לשליחת פקודה לארדואינו
def send_command(command):
    """שליחת פקודה לארדואינו דרך Serial"""
    if arduino:
        try:
            print(f"Sending command to Arduino: {command.strip()}")
            arduino.write((command + "\n").encode())
            time.sleep(0.1)  # השהייה ממושכת יותר
        except Exception as e:
            print(f"Error sending command to Arduino: {e}")
    else:
        print("Arduino is not connected. Command not sent.")

# אתחול YOLO
model = YOLO("yolo-Weights/yolov8n.pt")
```

```
# הפעלת מצלמה
cap = cv2.VideoCapture(2)
cap.set(3, 640)  # רוחב הפריים
cap.set(4, 480)  # גובה הפריים

# רשימת אובייקטים שניתן לזהות
classNames = ["person", "bicycle", "car", "motorbike", "aeroplane", "bus", "train", "truck",
              "boat", "traffic light", "fire hydrant", "stop sign", "parking meter", "bench",
              "bird", "cat", "dog", "horse", "sheep", "cow", "elephant", "bear", "zebra", "giraffe",
              "backpack", "umbrella", "handbag", "tie", "suitcase", "frisbee", "skis", "snowboard",
              "sports ball", "kite", "baseball bat", "baseball glove", "skateboard", "surfboard",
              "tennis racket", "bottle", "wine glass", "cup", "fork", "knife", "spoon", "bowl",
              "banana", "apple", "sandwich", "orange", "broccoli", "carrot", "hot dog", "pizza",
              "donut", "cake", "chair", "sofa", "pottedplant", "bed", "diningtable", "toilet",
              "tvmonitor", "laptop", "mouse", "remote", "keyboard", "cell phone", "microwave",
              "oven", "toaster", "sink", "refrigerator", "book", "clock", "vase", "scissors",
              "teddy bear", "hair drier", "toothbrush"]

# הגדרת אובייקטים רלוונטיים למעקב
relevant_objects = ["bird"]

# הגדרת משתנה למעקב אחרי אם זוהה אדם
person_detected = False
last_command = ""
```

```python
while True:
    success, img = cap.read()

    if not success:
        print("Error reading frame from camera.")
        break

    results = model(img, stream=True)
    detected_command = "STOP"
    person_detected = False

    for r in results:
        boxes = r.boxes
        for box in boxes:
            x1, y1, x2, y2 = box.xyxy[0]
            x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
            confidence = math.ceil((box.conf[0] * 100)) / 100
            cls = int(box.cls[0])
            class_name = classNames[cls]
            print(f"Detected: {class_name}, Confidence: {confidence}")

            if class_name in relevant_objects and confidence > 0.5:
                person_detected = True
                center_x = (x1 + x2) // 2
                frame_center_x = img.shape[1] // 2

                if center_x < frame_center_x - 50:
                    detected_command = "LEFT"
                elif center_x > frame_center_x + 50:
                    detected_command = "RIGHT"
                else:
                    detected_command = "FORWARD"
```

```python
                org = (x1, y1)
                font = cv2.FONT_HERSHEY_SIMPLEX
                fontScale = 1
                color = (255, 0, 0)
                thickness = 2
                cv2.putText(img, f"{class_name} {confidence}", org, font, fontScale, color, thickness)
                cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 255), 3)

    if not person_detected:
        if time.time() % 2 < 1:
            detected_command = "RIGHT"
        else:
            detected_command = "LEFT"

    if detected_command != last_command:
        send_command(detected_command)
        last_command = detected_command

    cv2.imshow('Webcam', img)
    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        print("Exiting program...")
        send_command("STOP")
        break

cap.release()
cv2.destroyAllWindows()
```

קוד הארדואינו:

```
// ------------------- הגדרת פינים למנועים -------------------
const int IN1 = 2;  // מנוע של הזרוע
const int IN2 = 3;   // מנוע של הזרוע
const int ENA = 4;

const int IN5 = 10;  // מנוע 3 - קדימה
const int IN6 = 9;   // מנוע 3 - אחורה
const int ENC = 8;   // בקרת מהירות מנוע 3 (PWM)

const int IN7 = 13;  // מנוע 4 - קדימה
const int IN8 = 12;  // מנוע 4 - אחורה
const int END = 11;  // בקרת מהירות מנוע 4 (PWM)

// ------------------- חיישן אולטרסוני (HC-SR04) -------------------
#define TRIG_PIN A2
#define ECHO_PIN A3
#define SAFE_DISTANCE 50  // מרחק בטוח מינימלי (ס"מ)

// ------------------- הצהרות לפונקציות -------------------
float getDistance();
void stopMotors();
void moveForward();
void moveBackward();
void turnLeft();
void turnRight();

void setup() {
    Serial.begin(9600);

    // הגדרת כל הפינים כיציאות
    pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);
    pinMode(IN5, OUTPUT); pinMode(IN6, OUTPUT); pinMode(ENC, OUTPUT);
    pinMode(IN7, OUTPUT); pinMode(IN8, OUTPUT); pinMode(END, OUTPUT); pinMode(ENA, OUTPUT);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);

    stopMotors();  // התחלה עם מנועים כבויים
}
```

```
// ------------------- פונקציה לקריאת המרחק מהחיישן -------------------
float getDistance() {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);

    long duration = pulseIn(ECHO_PIN, HIGH);
    float distance = duration * 0.034 / 2;  // חישוב מרחק בס"מ

    if (duration == 0) {
        return -1;  // אם אין החזרה מהאולטרסוניק
    }

    return distance;
}

// ------------------- פונקציות לשליטה במנועים -------------------
void moveForward() {
    Serial.println(" תנועה קדימה");
    digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
    digitalWrite(IN5, LOW); digitalWrite(IN6, HIGH);
    digitalWrite(IN7, LOW); digitalWrite(IN8, HIGH);
    analogWrite(ENC, 255); analogWrite(END, 255);analogWrite(ENA, 0);
}
```

```
void moveBackward() {
    Serial.println("תנועה אחורה");
    digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
    digitalWrite(IN5, HIGH); digitalWrite(IN6, LOW);
    digitalWrite(IN7, HIGH); digitalWrite(IN8, LOW);
    analogWrite(ENC, 255); analogWrite(END, 255);analogWrite(ENA, 0);
}

void turnLeft() {
    Serial.println(" פנייה שמאלה");
    digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
    digitalWrite(IN5, LOW); digitalWrite(IN6, HIGH);
    digitalWrite(IN7, HIGH); digitalWrite(IN8, LOW);
    analogWrite(ENC, 100); analogWrite(END, 100);analogWrite(ENA, 0);
}

void turnRight() {
    Serial.println(" פנייה ימינה");
    digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
    digitalWrite(IN5, HIGH); digitalWrite(IN6, LOW);
    digitalWrite(IN7, LOW); digitalWrite(IN8, HIGH);
    analogWrite(ENC, 100); analogWrite(END, 100);analogWrite(ENA, 0);
}

void stopMotors() {
    Serial.println(" עצירת מנועים");
    digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);
    digitalWrite(IN5, LOW); digitalWrite(IN6, LOW);
    digitalWrite(IN7, LOW); digitalWrite(IN8, LOW);
    analogWrite(ENC, 0); analogWrite(END, 0);analogWrite(ENA, 255);
}
```

```
// ------------------- לולאת התוכנית הראשית -------------------
float lastDistance = -1;

void loop() {
    // קריאת המרחק מהחיישן
    float distance = getDistance();

    // שליחת המרחק רק אם הוא השתנה ביותר מ-5 ס"מ או קטן מ-50 ס"מ
    if (distance < SAFE_DISTANCE) {
        Serial.print("DISTANCE: ");
        Serial.println(distance);
        lastDistance = distance;
    }

    // עצירה אוטומטית אם המרחק קטן מ-50 ס"מ
    if (distance > 0 && distance < SAFE_DISTANCE) {
        Serial.println("⚠ עצירה - קרוב מדי!");
        stopMotors();
        delay(500);
    }

    // אם יש פקודה חדשה מהפייתון והמרחק בטוח
    else if (Serial.available() > 0 && distance >= SAFE_DISTANCE) {
        String command = Serial.readStringUntil('\n');
        command.trim();

        Serial.print("COMMAND: ");
        Serial.println(command);

        if (command == "FORWARD") {
            moveForward();
        }
        else if (command == "BACKWARD") {
            moveBackward();
        }
        else if (command == "LEFT") {
            turnLeft();
        }
        else if (command == "RIGHT") {
            turnRight();
        }
        else if (command == "STOP") {
            stopMotors();
        }
    }

    delay(100);  // מניעת קריאות יתר
}
```