

RaspTankPro — מדריך תכנות לסטודנטים

מדריך זה מסביר כיצד לכתוב תוכניות לרובוט באמצעות **סביבת הסנדבוקס**. תשתמשו בפייתון וסט פשוט של פקודות — אין צורך בידע קודם בחומרה או אלקטרוניקה.

תוכן העניינים

- 1. סקירת החומרה
- 2. תחילת עבודה
- 3. תנועה
- 4. חיישנים
- 5. סרוויים וזרוע
- 6. אודומטריה ויזואלית (מעקב מיקום)
- 7. תצוגה ו-LEDs
- 8. תוכניות לדוגמה
- 9. הרצת בדיקות החומרה
- 10. פתרון בעיות

1. סקירת החומרה

תיאור	רכיב
	מנועי הנעה
שני מנועים (שמאל וימין) לשליטה בתנועה ובפנייה	
	סרוויים
חמישה סרוויים: סיבוב זרוע שמאל-ימין, זרוע למעלה-למטה, יד למעלה-למטה, אחיזה פתוח-סגור, הטיית מצלמה למעלה-למטה	
	חיישן אולטרסוני
מודד מרחק למכשול הקרוב ביותר (במטרים)	
	MPU6050
ג'ירוסקופ ומד-תאוצה משולבים (מהירות זוויתית + תאוצה)	
	מצלמה
מצלמת Raspberry Pi, משמשת לשידור הווידאו ולאודומטריה ויזואלית	
	רצועת LED
16 נורות RGB NeoPixel	
	תצוגת OLED
מסך SSD1306 קטן עם 6 שורות טקסט	

2. תחילת עבודה

הפעלת הרובוט

- 1. הרובוט מופעל אוטומטית כאשר הוא מחובר למקור חשמל.
- 2. חברו מקלדת, עכבר וצג אם אתם צריכים לכתוב או לערוך תוכניות.
- 3. **Thonny IDE** נפתח אוטומטית עם **sandbox.py** טעון — זו סביבת התכנות שלכם.

כתיבה והרצה של התוכנית

1. הגדירו את פונקציות העזר שאתם צריכים באזור **הפונקציות** שמעל `run()` — יש שם פונקציה לדוגמה שמראה את הפורמט.
2. כתבו את התוכנית הראשית שלכם בתוך פונקציית `run()`.
3. לחצו **F5** (או לחצו על כפתור ה-Run ב-Thonny) להרצת התוכנית.
 - שרת הבקרה המקוון מופסק אוטומטית כאשר לוחצים Run.
 - החומרה של הרובוט זמינה עכשיו באופן בלעדי לתוכנית שלכם.
4. לחצו על כפתור **Stop** (או **F2**) בכל עת לעצירה בטוחה של הרובוט.

מעבר בין מצב שרת הבקרה המקוון למצב Sandbox

הקובץ `~/startup.sh (/home/pi/startup.sh)` קובע מה הרובוט מריץ בעת האתחול. פתחו אותו בעורך טקסט (לדוגמה: `nano ~/startup.sh`) ושנו את ה-`comments:`

מצב שרת הבקרה המקוון (ברירת מחדל — שליטה ברובוט דרך דפדפן):

```
# Comment this line when working in sandbox:
sudo python3 /home/pi/adeept_rasptankpro/server/webServer.py

# Uncomment this line when working in sandbox:
#sudo python3 /home/pi/adeept_rasptankpro/server/sandbox.py
```

מצב Sandbox (הרובוט מריץ את פונקציית `run()` שלכם אוטומטית בעת האתחול):

```
# Comment this line when working in sandbox:
#sudo python3 /home/pi/adeept_rasptankpro/server/webServer.py

# Uncomment this line when working in sandbox:
sudo python3 /home/pi/adeept_rasptankpro/server/sandbox.py
```

שמרו את הקובץ ואתחלו מחדש. הרובוט יתחיל במצב שנבחר.

עצירת חירום

אם הרובוט בורח או מתנהג בצורה בלתי צפויה ואינכם יכולים להגיע ל-Thonny:

- לחצו פעמיים על האייקון **EMERGENCY STOP** בשולחן העבודה.
- פעולה זו מפסיקה מיידית את התוכנית הרצה ועוצרת את המנועים.

3. תנועה

כל פונקציות התנועה מקבלות `speed` (מהירות, 0–100) ו-`duration` אופציונלי בשניות. אם `duration` לא מצוין, הרובוט ימשיך לנוע עד שתקראו ל-`robot.stop()`.

```

robot.forward(speed=50, duration=2.0) # נוע קדימה 2 שניות במחצית המהירות
robot.backward(speed=50, duration=1.0) # נוע אחורה שנייה אחת
robot.turn_left(speed=50, duration=0.5) # פנה שמאלה (גלגל ימין נוסע, שמאל עוצר)
robot.turn_right(speed=50, duration=0.5) # פנה ימינה (גלגל שמאל נוסע, ימין עוצר)
robot.spin_left(speed=40, duration=1.0) # סתובב במקום נגד כיוון השעון
robot.spin_right(speed=40, duration=1.0) # סתובב במקום עם כיוון השעון
robot.stop() # עצור מיד
robot.wait(1.5) # חסום 1.5 שניות – המנועים ממשיכים לרוץ
    אם כבר פעלו

```

תיאור הפונקציות

robot.forward(speed=50, duration=None)

נוע קדימה.

- **speed** — עוצמת מנוע, 0–100 (ברירת מחדל: 50)
- **duration** — שניות לנסיעה; אם **None**, נוסע עד לקריאה ל-**stop()**

robot.backward(speed=50, duration=None)

נוע אחורה. אותם פרמטרים כמו **forward**.

robot.turn_left(speed=50, duration=None)

פנה שמאלה. גלגל ימין נוסע קדימה; גלגל שמאל עוצר.

robot.turn_right(speed=50, duration=None)

פנה ימינה. גלגל שמאל נוסע קדימה; גלגל ימין עוצר.

robot.spin_left(speed=50, duration=None)

סתובב נגד כיוון השעון במקום. שני הגלגלים נעים באותה מהירות בכיוונים מנוגדים.

robot.spin_right(speed=50, duration=None)

סתובב עם כיוון השעון במקום.

robot.stop()

עצור את כל המנועים מיידית.

robot.wait(seconds)

חסום ביצוע למשך מספר השניות הנתון — התוכנית לא עושה דבר בזמן זה. **כל מנוע שכבר פעל ימשיך לפעול.**
השתמשו ב-**robot.stop()** לפני או אחרי **wait()** אם רוצים שהרובוט יעמוד במקום.

4. חיישנים

חיישן מרחק

```
distance = robot.get_distance()
print(f"מטר מהרובוט {distance:.2f} המכשול נמצא")
```

`robot.get_distance()` → float

מחזיר את המרחק (במטרים) למכשול הקרוב ביותר ישירות מול החיישן. ערכים מעל 2 מטר עשויים להיות לא מדויקים. מחזיר 0 אם לא התקבל הד.

ג'ירוסקופ

```
gyro = robot.get_gyro()
print(f"סיבוב - x: {gyro['x']:.2f} y: {gyro['y']:.2f} z: {gyro['z']:.2f} °/s")
```

`robot.get_gyro()` → dict

מחזיר מהירות זוויתית במעלות לשנייה כמילון עם מפתחות 'x', 'y', 'z'.

ציר	משמעות
x	גלגול (הטייה לצדדים)
y	הצעדה (הטייה קדימה/אחורה)
z	פנייה (סיבוב שמאל/ימין)

מחזיר {`'x': 0`, `'y': 0`, `'z': 0`} אם החיישן אינו מחובר.

מד-תאוצה

```
accel = robot.get_accel()
print(f"תאוצה - x: {accel['x']:.2f} y: {accel['y']:.2f} z: {accel['z']:.2f} g")
```

`robot.get_accel()` → dict

מחזיר תאוצה ליניארית ב- g ($1 g \approx 9.81$ מ/ש²) כמילון עם מפתחות 'x', 'y', 'z'. כאשר הרובוט שטוח ובמנוחה, $z \approx 1.0$ (כבידה) ו- $x \approx y \approx 0$.

מחזיר {`'x': 0`, `'y': 0`, `'z': 1`} אם החיישן אינו מחובר.

5. סרבוים וזרוע

כל פונקציות הסרבוים מקבלות **position** (מיקום) בין -1.0 (קצה אחד) ל- $+1.0$ (הקצה השני), כאשר 0.0 הוא המרכז.

```
robot.set_arm_rotation(-0.5) # סובב זרוע חצי דרך שמאלה
robot.set_arm(0.8)          # הרם זרוע 80% מהדרך למעלה
robot.set_hand(0.5)         # הרם יד חצי דרך למעלה
robot.set_gripper(-1.0)     # סגור אחיזה לחלוטין
robot.set_gripper(1.0)      # פתח אחיזה לחלוטין
robot.set_camera_tilt(1.0)  # הטה מצלמה לחלוטין למעלה
robot.reset_servos()        # החזר את כל הסרבוים למרכז
```

תיאור הפונקציות

robot.set_arm_rotation(position)

סובב את הזרוע שמאלה או ימינה (סיבוב הגוף).

- -1.0 = שמאל מלא, 0.0 = מרכז, $+1.0$ = ימין מלא

robot.set_arm(position)

הזז את מפרק הזרוע למעלה או למטה.

- -1.0 = לחלוטין למטה, 0.0 = מרכז, $+1.0$ = לחלוטין למעלה

robot.set_hand(position)

הזז את היד (פרק כף היד / אמה) למעלה או למטה.

- -1.0 = לחלוטין למטה, 0.0 = מרכז, $+1.0$ = לחלוטין למעלה

robot.set_gripper(position)

פתח או סגור את האחיזה.

- -1.0 = סגור לחלוטין, 0.0 = מרכז, $+1.0$ = פתוח לחלוטין

robot.set_camera_tilt(position)

הטה את המצלמה למעלה או למטה.

- -1.0 = למטה מלא, 0.0 = מרכז, $+1.0$ = למעלה מלא

robot.reset_servos()

החזר את כל הסרבוים למיקום המרכזי שלהם (מיקום 0.0).

6. אודומטריה ויזואלית (מעקב מיקום)

אודומטריה ויזואלית מעריכה את מיקום הרובוט על-ידי ניתוח התנועה בין פריימים עוקבים של המצלמה. היא משתמשת בזיהוי נקודות FAST וזרימה אופטית לוקאס-קאנדה כדי לעקוב כיצד הסצנה משתנה מפריים לפריים, ואז מחשבת את ההזזה התלת-מימדית של המצלמה.

מגבלות חשובות

אי-בהירות סקאלה חד-עינית: מצלמה בודדת אינה יכולה לקבוע מרחקים אמיתיים ללא מקור ייחוס חיצוני. ערכי x, y, z המוחזרים על-ידי `get_position()` הם ביחידות יחסיות, לא מטרים. הסקאלה תלויה ב-`absolute_scale` (ברירת מחדל: 1.0). אם דרושים לכם מרחקים אמיתיים, יש לכייל ערך זה ביחס למרחק ידוע.

סחיפה: האודומטריה הויזואלית צוברת שגיאות קטנות עם הזמן. מסלולים ארוכים יראו סחיפה מהמיקום האמיתי.

התנגשות בשימוש במצלמה: שרת הבקרה המקוון משתמש גם הוא במצלמה. `start_odometry()` דורש שהשרת יהיה מופסק — הדבר קורה אוטומטית כאשר לוחצים F5 ב-Thonny.

שימוש

```
# התחל מעקב מיקום
robot.start_odometry()

# ... הזז את הרובוט ...
robot.forward(speed=40, duration=3.0)

# (ביחידות יחסיות x, y, z) קרא את המיקום הנוכחי
x, y, z = robot.get_position()
print(f"מיקום: x={x:.2f} y={y:.2f} z={z:.2f}")

# אפס נקודת מוצא למיקום הנוכחי
robot.reset_position()

# עצור מעקב כשסיימת
robot.stop_odometry()
```

תיאור הפונקציות

`robot.start_odometry(focal_length=537.0, pp=(320.0, 240.0), scale=1.0, show_debug=False)`

התחל מעקב מיקום ברקע. לכל הפרמטרים יש ערכי ברירת מחדל — ניתן להעביר רק את הנדרשים. לדוגמה, `robot.start_odometry(scale=4.7)` תקין לחלוטין.

- `focal_length` — אורך מוקד המצלמה בפיקסלים (ברירת מחדל: 537.0 למצלמת Pi ב-480×640)
- `pp` — נקודת העיקרון (cx, cy) בפיקסלים (ברירת מחדל: (240.0, 320.0))
- `scale` — מקדם סקאלה המוחל על כל שלב תרגום (ברירת מחדל: 1.0)
- `show_debug` — אם `True`, פותח שני חלונות בזמן ריצת האודומטריה: תמונה חיה מהמצלמה עם נקודות המעקב (ירוק) ומפת מסלול דו-ממדית. שימושי לזיהוי שהמצלמה עובדת והמסלול הנחשב הגיוני. ברירת מחדל: `False`.

מחזיר `RuntimeError` אם לא ניתן לפתוח את המצלמה.

`robot.get_position() → (x, y, z)`

מחזיר את הערכת המיקום העדכנית כ-tuple של שלושה מספרים עשרוניים. נקודת המוצא היא מיקום הרובוט כאשר נקרא `start_odometry()` או `reset_position()`. מחזיר `RuntimeError` אם לא נקרא `start_odometry()`.

`robot.reset_position()`

אפס את המיקום הנוכחי ל-(0, 0, 0). מחזיר `RuntimeError` אם לא נקרא `start_odometry()`.

`robot.stop_odometry()`

עצור מעקב מיקום ושחרר את המצלמה.

7. תצוגה ו-LEDs

רצועת LED

לרובוט 16 נורות RGB. הגדירו את כולן לכל צבע עם שלושה ערכים בטווח 0–255 (אדום, ירוק, כחול).

```
robot.set_led_color(255, 0, 0)    # אדום
robot.set_led_color(0, 255, 0)   # ירוק
robot.set_led_color(0, 0, 255)   # כחול
robot.set_led_color(255, 165, 0) # כתום
robot.led_off()                  # כיבוי
```

`robot.set_led_color(r, g, b)`

הגדר את כל הנורות לצבע RGB.

• `r, g, b` — ערכי אדום, ירוק, כחול, כל אחד בטווח 0–255

`robot.led_off()`

כבה את כל הנורות (שקול ל-`set_led_color(0, 0, 0)`).

תצוגת OLED

לרובוט מסך OLED קטן מסוג SSD1306 עם 6 שורות טקסט (שורה 1 למעלה, שורה 6 למטה).

```
robot.show_display(1, 'שלום!')
robot.show_display(2, f'מרחק: {distance:.2f} מ')
robot.show_display(3, f'מהירות: {speed}')
robot.clear_display()           # נקה את כל השורות
```

`robot.show_display(line, text)`

כתוב טקסט לשורה בתצוגה.

- `line` — 1 (למעלה) עד 6 (למטה)
- `text` — כל ערך; מספרים מומרים אוטומטית למחרוזות

`robot.clear_display()`

נקה את כל שש שורות התצוגה.

8. תוכניות לדוגמה

דוגמה 1 — נוע קדימה ועצור לפני מכשול

```
def run():
    SAFE_DISTANCE = 0.30 # מטרים

    robot.forward(speed=40) # (ללא) duration - התחל לנסוע

    while True:
        distance = robot.get_distance()
        print(f"מרחק: {distance:.2f} מ")

        if distance < SAFE_DISTANCE:
            robot.stop()
            print("זוהה מכשול! עוצר.")
            break

    robot.wait(0.05) # השהייה קצרה בין קריאות
```

דוגמה 2 — סריקת סיבוב זרוע עם קריאות מרחק

```
def run():
    print("סורק...")
    positions = [-1.0, -0.5, 0.0, 0.5, 1.0]

    for pos in positions:
        robot.set_arm_rotation(pos)
        robot.wait(0.5) # המתן עד שהסרנו יתייצב
        distance = robot.get_distance()
        angle_label = f"{int(pos * 90):+d}°"
        print(f" {angle_label}: {distance:.2f} מ")

    robot.reset_servos()
    print("הסריקה הושלמה.")
```


דוגמה 3 — רישום נתוני חיישנים לקובץ

```
def run():
    import csv
    import time

    LOG_FILE = '/home/pi/sensor_log.csv'
    DURATION = 10.0 # שניות

    print(f"שניות {DURATION} → {LOG_FILE} רישום חיישנים למשך")

    with open(LOG_FILE, 'w', newline='') as f:
        writer = csv.writer(f)
        writer.writerow(['time_s', 'distance_m',
                        'gyro_x', 'gyro_y', 'gyro_z',
                        'accel_x', 'accel_y', 'accel_z'])

    start = time.time()
    while time.time() - start < DURATION:
        t = time.time() - start
        dist = robot.get_distance()
        gyro = robot.get_gyro()
        accel = robot.get_accel()

        writer.writerow([
            round(t, 3), dist,
            gyro['x'], gyro['y'], gyro['z'],
            accel['x'], accel['y'], accel['z'],
        ])
        print(f"t={t:.1f}s מרחק={dist:.2f}m "
              f"gyro_z={gyro['z']:.2f}")
        time.sleep(0.1)

    print("בגיליון אלקטרוני לניתוח הנתונים CSV-סיום. פתחו את קובץ ה")
```

9. הרצת בדיקות החומרה

כדי לוודא שכל החומרה עובדת כראוי:

```
# בפקוד הבא run() החליפו את תוכן, sandbox.py ב-
from robot_test import run_all_tests

def run():
    run_all_tests(robot)
```

סקריפט הבדיקה יבדוק כל רכיב ברצף ויציא [PASS] או [FAIL] לכל אחד. הבדיקה המלאה נמשכת כ-30 שניות ומזיזה את הרובוט פיזית, לכן וודאו שיש מקום סביבו.

10. פתרון בעיות

"Cannot open camera" בעת קריאה ל-`start_odometry()`

המצלמה בשימוש על-ידי תהליך אחר (בדרך כלל שרת הבקרה המקוון). וודאו שהרצתם את התוכנית דרך Thonny (F5), שמפסיק את השרת אוטומטית לפני התחלת הקוד.

שגיאות I2C / חיישן בהפעלה

המד-תאוצה/ג'ירוסקופ MPU6050 מתקשר דרך I2C. אם האתחול נכשל, `get_gyro()` ו-`get_accel()` מחזירות ערכי אפס במקום לקרוס. בדקו שהחיישן מחובר ו-I2C מופעל (`sudo raspi-config` → Interface Options → I2C).

סרוויים לא זזים

בקר הסרוו משתמש ב-I2C (אותו אפיק כמו הג'ירוסקופ). בדקו את החיבורים ו-I2C מופעל. אם רק חלק מהסרוויים נכשלים, ייתכן שיש בעיה בחיבור ערוץ ה-PWM.

הרובוט לא עוצר כאשר לוחצים F2

Thonny שולח SIGTERM לסקריפט הרץ. `sandbox.py` קולט אות זה וקורא ל-`robot.cleanup()`, שעוצר את המנועים. אם הרובוט עדיין נע, השתמשו בקיצור הדרך **EMERGENCY STOP** בשולחן העבודה.

אזהרות GPIO בהפעלה

ה-GPIO של Raspberry Pi עשוי להציג `RuntimeWarning: This channel is already in use`. זה לא מזיק — פירושו שה-GPIO לא שוחרר כראוי בריצה קודמת. הרובוט עדיין יעבוד כהלכה.

המנועים מסתובבים אך הרובוט לא נוסע ישר

לשני המנועים עשויה להיות יעילות שונה מעט. השתמשו בערך `speed` גבוה יותר, או שנו את הפרמטר `radius` דרך `move.move()` ישירות לשליטה מדויקת יותר (שימוש מתקדם).