

# — מדריך תכנות לסטודנטים — RaspTankPro

מדריך זה מסביר כיצד לכתוב תוכניות לרובוט באמצעות **סביבת הסנדיוקס**. תשתמשו בפייטון וسط פשוט של פקודות — אין צורך בידע קודם בחומרה או אלקטרוני.

## תוכן העניינים

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

## 1. סקירת החומרה

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

## 2. תחילת עבודה

### הפעלת הרובוט

1. הרובוט מופעל אוטומטית כאשר הוא מחובר למקור חשמל.
2. חיבור מקלחת, עכבר וציג אם אתם צריכים לכתוב או לעורר תוכניות.
3. נפתח אוטומטית עם `sandbox.py` טען — זו סביבת התכנות שלכם.

## כתיבה והרצה של התוכנית

1. הגדרו את פונקציות העזר שאתם צריכים באזורי **הfonקציות** שמעל (`run`) — יש שם פונקציה לדוגמה שמרת את הפורטט.
2. כתבו את התוכנית הראשית שלכם בתחום פונקציית (`run`).
3. לחזו **F5** (או לחזו על כפתור Run ב-Thonny) להרצה התוכנית.
  - שרת הבקרה המקורי מופסק אוטומטית כאשר לחצים Run.
  - החומרה של הרובוט זמינה עכשו באופן בלעדי לתוכנית שלכם.
4. לחזו על כפתור **Stop** (או **F2**) בכל עת לעצירה בתוכה של הרובוט.

מעבר בין מצב שרת הבקרה המקורי למסך **Sandbox**

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

**מסך שרת הבקרה המקורי** (ברירת מחדל — שליטה ב Robbins דרך דף|פ|ן):

```
# 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
```

**מסך Sandbox** (הרובוט מרים 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 יתחל במצב שנבחר.

### עיצוב חירום

אם הרובוט בורח או מתרנה בצורה בלתי צפופה ואינכם יכולים להגיע אליו: Thonny

- לחזו פעמיים על האיקון **EMERGENCY STOP** בשולחן העבודה.
- פעולה זו מפסיקת מידית את התוכנית הרצה ועצרת את המנועים.

## 3. תנועה

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

```

נוט קדימה 2 שניות במחזית המהירות #  

נוט אחורה שנייה אחת #  

(פנה שמאל (גלגל ימין נוסע, שמאל עוצר #  

(פנה ימינה (גלגל שמאל נוסע, ימין עוצר#  

סובב במקום נגד כיוון השעון #  

סובב במקום עם כיוון השעון#  

עצור מיד #  

חסום 1.5 שניות — המנועים ממשיכים לרווק #
אם כבר פועלו

```

## תיאור הפקציות

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

נוט קדימה.

- עצמת מנוע, 0–100 (ברירת מחדל: 50) •
- שניות לנסעה; אם `None`, נסע עד לקריאה ל-`(stop)` •

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

נוט אחורה. אותו פרמטרים כמו `.forward`.

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

פנה שמאל. גלגל ימין נסע קדימה; גלגל שמאל עוצר.

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

פנה ימינה. גלגל שמאל נסע 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 מידית.

### `robot.wait(seconds)`

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

## 4. חישונים

### חישון מרחק

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

`robot.get_distance() → float`

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

---

### ג'ירוסקופ

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

`robot.get_gyro() → dict`

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

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

מחזיר `{'z': 0, 'y': 0, 'x': 0}` אם החישון 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- $\approx 9.81 \text{ g}$  ( $\text{m/s}^2$ ) כמילון עם מפתחות 'z', 'y', 'x'. כאשר הרובוט שטוח ובמנוחה,  $\approx 1.0$  (ככידה) ו- $x \approx y \approx 0$ .

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

---

## 5. סרוויים וזרוע

כל פונקציית הסרוויו מקבלות **position** (מיקום) בין **-1.0** ל-**+1.0** (קצת אחד) כאשר **0.0** הוא המרכז.

```
robot.set_arm_rotation(-0.5) סובב זרוע צדי דרך שמאלה #
robot.set_arm(0.8) הרם זרוע 80% מהדרך למעלה #
robot.set_hand(0.5) הרם יד צדי דרך למעלה #
robot.set_gripper(-1.0) סגור אחיזה לחלווטין #
robot.set_gripper(1.0) פתוח אחיזה לחלווטין #
robot.set_camera_tilt(1.0) הטה מצלמה לחלווטין למעלה #
robot.reset_servos() ההדר את כל הסרוויים למרכז #
```

תיאור הפונקציות

### **robot.set\_arm\_rotation(position)**

סובב את הזרוע שמאליה או ימיןה (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** = לחלווטין למטה, **0.0** = מרכז, **+1.0** = לחלווטין למעלה

### **robot.set\_hand(position)**

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

- **-1.0** = לחלווטין למטה, **0.0** = מרכז, **+1.0** = לחלווטין למעלה

### **robot.set\_gripper(position)**

פתח או סגור את האחיזה.

- **-1.0** = סגור לחלווטין, **0.0** = מרכז, **+1.0** = פתוח לחלווטין

### **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 הסרוויים למיקום המרכזי שלהם (מיקום **0.0**).

## 6. אודומטריה ויזואלית (מעקב מיקום)

אודומטריה ויזואלית מעריכה את מיקום הרובוט על-ידי ניתוח התנועה בין פריים עוקבים של המצלמה. היא משתמשת ב**דיזייני נקודות FAST** ודריפה אופטית ל**ווקאס-קאנדה** כדי לעקוב כיצד הסצנה משתנה מפריים לפריים, ואז מחשבת את ההזזה התרלת-טימודית של המצלמה.

## מגבלות חשובות

**אי-ביהירות סקלה חד-ענית:** מצלמה בודדת אינה יכולה לקבוע מרחקים אמיתיים ללא מקור ייחוס חיצוני.

ערכים `z`, `y`, `x` המוחזרים על-ידי (`get_position()`) הם **ביחידות יחסיות**, לא מטרים. הסקלה תלויה ב- `absolute_scale` (ברירת מחדל: 1.0). אם דרישים לכם מרחקים אמיתיים, יש לכינן ערך זה ביחס למרחק ידוע.

**סחיפה:** האודומטריה היזואלית צוברת שגיאות קטנות עם הזמן. מסלולים ארוכים יראו סחיפה מהמיוקם האמיתי.

**התנגשות בשימוש במכשיר:** שרת הבקרה המקוון משתמש גם הוא במכשיר. (`start_odometry()` דוחש שהשתתת יהיה מופסק — הדבר קורה אוטומטית כאשר לוחצים F5 ב-*Thonny*).

## שימוש

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

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

# קריא את המיוקום הנוכחי
x, y, z = robot.get_position()
print(f"x={x:.2f} y={y:.2f} z={z:.2f}") # מיקום

# אפס נקודת מוצא למיוקום 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 להעיבר רק את הנדרשים. לדוגמה,  
`robot.start_odometry(scale=4.7)`

- `focal_length` — אורך מוקד המצלמה בפיקסלים (ברירת מחדל: 537.0 למצלמת Pi ב- $640 \times 480$ )
- `pp` — נקודת העיקרון (`cy, cx`) בפיקסלים (ברירת מחדל: (320.0, 240.0))
- `scale` — מקדם סקלה המוחל על כל שלב תראום (ברירת מחדל: 1.0)
- `show_debug` — אם `True`, פותח שני חלונות בזמן ריצת האודומטריה: תמונהchia מהמכשיר עם נקודות המיעקב (ירוק) ומפת מסלול דו-ממדית. שימושי לוידוא שהמכשיר עובדת והמסלול הנחשב הגיוני. ברירת מחדל: `False`

מחזיר אם לא ניתן לפתוח את המצלמה.

`robot.get_position() → (x, y, z)`

מחזיר את הערךת המיקום העדכנית C-tuple של שלושה מספרים עשרוניים. נקודת המוצא היא מיקום הרובוט כאשר נקרא `.start_odometry()` או `RuntimeError 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 ו歇רר את המצלמה.

## 7. תצוגה LEDs

### רצועת LED

לרובוט 16 נורות RGB. הגדרו את כלן לכל צבע עם שלושה ערכים בתווות 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 בתווות 0–255

`robot.led_off()`

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

### 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'מץ: {distance:.2f}')
robot.show_display(3, f'מהירות: {speed}')
robot.clear_display()               # נקה את כל השורות
```

**robot.show\_display(line, text)**

כתב טקסט לשורה בתצוגה.

- **line** — 1 עד 6 (למטה)
- **text** — כל ערך; מספרים מומרים אוטומטית למחוזות

**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)    # המהלך לנסוע – duration (לאן)

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

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

    robot.wait(0.05) # הושהייה קרצה בין קריאות
```

**דוגמה 2** — סריקת סיבוב זרוע עם קריאות מרחוק

```
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 שהсерו יתתייצב
        distance = robot.get_distance()
        angle_label = f"{int(pos * 90)}°"
        print(f"  {angle_label}: {distance:.2f} מ'")

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

### דוגמה 3 — רישום נתונים חישוניים לקובץ

```

def run():
    import csv
    import time

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

    print(f"רושם נתונים {DURATION} → {LOG_FILE}")

    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("בגלאיון אלקטרוני לניטוח הנתונים CSV-סיוום. פתחו את קובץ ה-")

```

## 9. הרצת בדיקות החומרה

כדי לוודא שכל החומרה עובדת כראוי:

```

# בקובץ הבא run() הצליפו את תוכן
from robot_test import run_all_tests

def run():
    run_all_tests(robot)

```

סקריפט הבדיקה יבדוק כל רכיב ברכז' ויציג [ PASS ] או [ FAIL ] לכל אחד. הבדיקה המלאה נמשכת כ-30 שניות ומציה את הרובוט פיזית, لكن וודאו שיש מקום סבירו.

## 10. פתרון בעיות

**start\_odometry()** בעת קריאה ל-

המצלמה בשימוש על-ידי תחילה אחר (בדרך כלל שרת הבדיקה המקורי). וודאו שהרכבתם את התוכנית דרך F5 (F5), שmpsik את השרת אוטומטית לפני התחלת הקוד.

שגיאות I2C / חישון בהפעלה

המד-תאוצה/ג'ירוסקופ MPU6050 מתקשר דרך I2C. אם האתחול נכשל, **(get\_accel() - get\_gyro())** מוחזרת ערכוAPS במקום לקרים. בדקו שהחישון מחובר ו-I2C מופעל (I2C → Interface Options → I2C מופעל).

סרוויים לא זזים

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

הרובוט לא עוצר כאשר לוחצים F2

Thonny שלוח SIGTERM לסקריפט הרץ. Thonny sandbox.py קולט אותן וקורא ל-**robot.cleanup()**, שעוצר את המנועים. אם הרובוט עדין נע, השתמשו בקיצור הדרך **EMERGENCY STOP** בשולחן העבודה.

ازהרות GPIO בהפעלה

ה-GPIO של Raspberry Pi עשוי להציג **RuntimeWarning: This channel is already in use**. זה לא מזיק — פירשו שהוא-GPIO לא שוחרר כראוי בritchא קודמת. הרובוט עדין יעבד כהכלמה.

המנועים מסתובבים אך הרובוט לא נושא ישך

לשני המנועים עשויה להיות עליות שונות מעט. השתמשו בערך **speed** גובה יותר, או שנו את הParmeter **radius** דרך move.move()