

מדריך תכנות לסטודנטים — RaspTankPro

מדריך זה מסביר כיצד לכתוב תוכניות לרובוט באמצעות **סביבת הסנדיוקס**. תשתמשו בפייטון וסט פשט 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. תוכניות לדוגמה
8. הרצת בדיקות החומרה
9. פתרון בעיות

1. סקירת החומרה

תיאור	רכיב
מנועי הנעה	שני מנועים (שמאל ימין) לשיליטה בתנועה ובפנייה
סרויים	חמש סרויים: סיבוב זרוע שמאל-ימין, זרוע למעלה-למטה, יד למעלה-למטה, אחיזה פתוח-סגור, הטית מצלמה למעלה-למטה
חיישן אולטרסוני	מודד מרחק למכשול הקרוב ביותר (מטרים)
 MPU6050	ג'ירוסקופ ומד-תאוצה מושלבים (מהירות זוויתית + תאוצה)
מצלמה	מצלמת Pi, Raspberry, משמשת לשידור הויזואו ולאודומטריה ויזואלית

2. תחילת עבודה

הפעלת הרובוט

1. חיבור מקלדת, עכבר וצג לרובוט.
2. הפעילו את הרובוט. ה-Pi Raspberry יתחל ויופיע שולחן העבודה.
3. נפתח אוטומטית עם **sandbox.py** טען — זו סביבת התכנות שלכם.

כתיבה והרצה של התוכנית

1. ערכו את פונקציית (**run**) **sandbox** בתחום **py**. כל מה שבתוכה (**run**) הוא התוכנית שלכם.
2. לחצו **F5** (או לחזו על כפתור-h-Run ב-Thonny) להרצה התוכנית.
 - שרת הבקרה המקורי מופסק אוטומטית כאשר לוחצים Run.
 - החומרה של הרובוט זמינה עכשו/apنן לתוכנית שלכם.

3. לחזו על כפתור **Stop** (או **F2**) בכל עת לעצירה בטוחה של הרובוט.
4. כדי לאותחל לשרת הבקרה המקורי (מצב הדגמה), הפעילו מחדש את הרובוט.

עצירת חירום

אם הרובוט בורח או מתרנה בצורה בלתי צפואה ואנים יכולים הגיעו לה-**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 נוער עד שתקראו ל-**robot.stop()**.

```
נושך קדימה 2 שניות במחזית מהירות #  
נושך אחורה שנייה אחת #  
(פנה שמאלה (גלגל ימין נוסע, שמאל עוזר #  
(פנה ימינה (גלגל שמאל נוסע, ימין עוזר #  
סתובב במקום נגד כיוון השעון #  
סתובב במקום עם כיוון השעון #  
robot.stop() # מיד  
robot.wait(1.5) (המתן 1.5 שניות (הרובוט נשאר במקום #
```

תיאור הפונקציות

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

נושך קדימה.

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

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

נושך אחורה. אותו פרמטרים כמו **forward**.

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

פנה שמאלה. גלגל ימינו נוער קדימה; גלגל שמאל עוזר.

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

פנה ימינה. גלגל שמאל נוער קדימה; גלגל ימין עוזר.

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)`

המתן למשר מספר השניות הנדרש. הרובוט נשאר במקום.

4. חישונים

חישון מרחק

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

`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 חד.

ג'יروسkop

```
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 לשניה כמיילן 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- $g \approx 9.81$ מ/ש² כמיון עם מפתחות 'x', 'y', 'z'. כאשר הרובוט שטוח ומנוחה, $z \approx 0$.
 $x \approx 0$ (ככזה) ו- $y \approx 0$.

מחזיר `{'z': 1, 'x': 0, 'y': 0}` אם החישון אינו מחויב.

5. סרוויסים וזרוע

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

```
robot.set_arm_rotation(-0.5) # סובב זרוע צדי דרך שמאליה
robot.set_arm(0.8) # הרם זרוע 80% מהדרך למעלה
robot.set_hand(0.5) # הרם יד צדי 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() # הזרע את כל הסרוויסים למרכז
```

תיאור הפונקציות

`robot.set_arm_rotation(position)`

סובב את הזרוע שמאליה 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.

- $-1.0 =$ לחלווטין למיטה, $0.0 =$ מרכז, $+1.0 =$ לחלווטין למעלה

`robot.set_hand(position)`

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

- $-1.0 =$ לחלווטין למיטה, $0.0 =$ מרכז, $+1.0 =$ לחלווטין 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 הservo'ים למיקום המרכזית שלהם (מיקום 0.0).

6. אודומטריה ויזואלית (מעקב מיקום)

אודומטריה ויזואלית מעריצה את מיקום הרובוט על-ידי ניתוח התנועה בין פרויינים עוקבים של המצלמה. היא משתמשת ב**דיזייני נקודות FAST** ו**זרימה אופטית לוקאס-קאנדה** כדי לעקוב כיצד הסצנה משתנה מפריים למפריים, ואז מחשבת את ההזזה התרلت-מידית של המצלמה.

מגבליות חשובות

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

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

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

שימוש

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

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

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

# אפס נקודת מוצא למיקום הנוכחי #
robot.reset_position()

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

תיאור הפקציות

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

התחל מעקב מיקום ברקע.

- `focal_length` — אורך מוקד המצלמה בפיקסלים (ברירת מחדל: 537.0 למצלמת Pi ב- 640×480)
- `pp` — נקודת העיקרון (cx, cy) בפיקסלים (ברירת מחדל: (320.0, 240.0))
- `scale` — מקדם סקאללה המוחל על כל שלב תרגום (ברירת מחדל: 1.0)
- אם `True`, פותח שני חלונות בזמן ריצת האודומטריה: תמונה חייה מהמצלמה עם נקודות המיעקב (ירוק) ומפת מסלול דו-ממדית. שימושו ליזוא 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 נקרה.

`robot.reset_position()`

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

`robot.stop_odometry()`

עצור מעקב מיקום ושחרר את המצלמה.

7. תוכניות לדוגמה

דוגמה 1 — نوع קדימה ועצור לפני מכשול

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

    robot.forward(speed=40) # המהלך לנסוט – duration התחילה לנסוט (לא #)

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

        if distance < SAFE_DISTANCE:
            robot.stop()
            print("זזהה מכשול! עוזר")
            break

    robot.wait(0.05) # ההשניה קצרה בין קרייאות
```

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 שהсерו יתני צב
        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"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  מטר={dist:.2f}\n" +
                  f"gyro_z={gyro['z']:.2f}")
            time.sleep(0.1)

```

`print("בגילינו אלקטרוני נניתוח הנתונים CSV-סיום. פתחו אתקובץ ה")`

8. הרצת בדיקות החומרה

כדי לוזודא שכל החומרה עוכצת כראוי:

```
# בקוד הבא run() הפעיל את תוכן run_all_tests()
from robot_test import run_all_tests

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

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

9. פתרון בעיות

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

המצלמה בשימוש על-ידי תהילך אחר (בדרך כלל שרת הבקרה המקורי). וידאו שהרצתם את התוכנית דרך F5 (F5), שמספיק את השרת אוטומטית לפני התחלה הקוד.

שגיאות C2 / חישון בהפעלה

סרוויים לא זרים

בקר הסרו משותב ב-C2a (אותו אפיק כמו הג'ירוסקופ). בדקנו את החיבורים ו-ש-C2a מופעל. אם רק חלק מהסרואים נשלמים, יתכן שיש בעיה בחיבור ערוץ ה-PWM.

הרכות לא עוזר כאשר לוחצים על

Thonny שולח SIGTERM לסקריפט הרץ sandbox.py קולט אותו זה וקורא ל-robot.cleanup(). שעוצר את המנגנונים. אם הרובוט עדין נע, השתמשו בקיצור הדזר EMERGENCY STOP בשולחן העבודה.

ازהרות GPIO בהפעלה

פירושו שה-GPIO לא שוחרר כראוי בריצה קודמת. הרובוט עדין יעבד בהתאם.

המנועים מסתובבים ארכ הרובוט לא נסע ישר

לשני המנוועים עשוי להיות יעילות שונה מעט. השתמשו בעבר **speed** גובה יותר, או שנו את הפקטור **radius** דרך **(move, move)** ישירות לשילטה מדויקת יותר (שימוש מתקדם).

