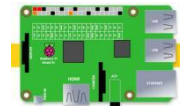
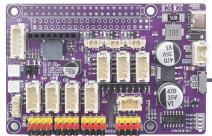



Lesson 13 Line Tracking

13.1 Overview

In this lesson, you'll learn to implement line - tracking functionality with a Raspberry Pi, Adeept Robot HAT V3.2, and a Line Tracking module. It covers components, principle, wiring, program demonstration, and code explanation.

13.2 Required Components

Components	Quantity	Picture
Raspberry Pi	1	
Adeept Robot HAT V3.2	1	
Line Tracking Module	1	

13.3 Principle Introduction

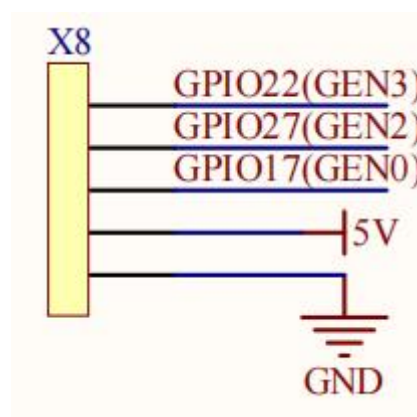
The Line Tracking module typically works based on the principle of light reflection. It often uses infrared - emitting diodes (IR LEDs) and phototransistors. When the IR LEDs emit infrared light, the light reflects differently from the line and the background. For example, if the line is black and the background is white, the black line absorbs more infrared light, while the white background reflects more.

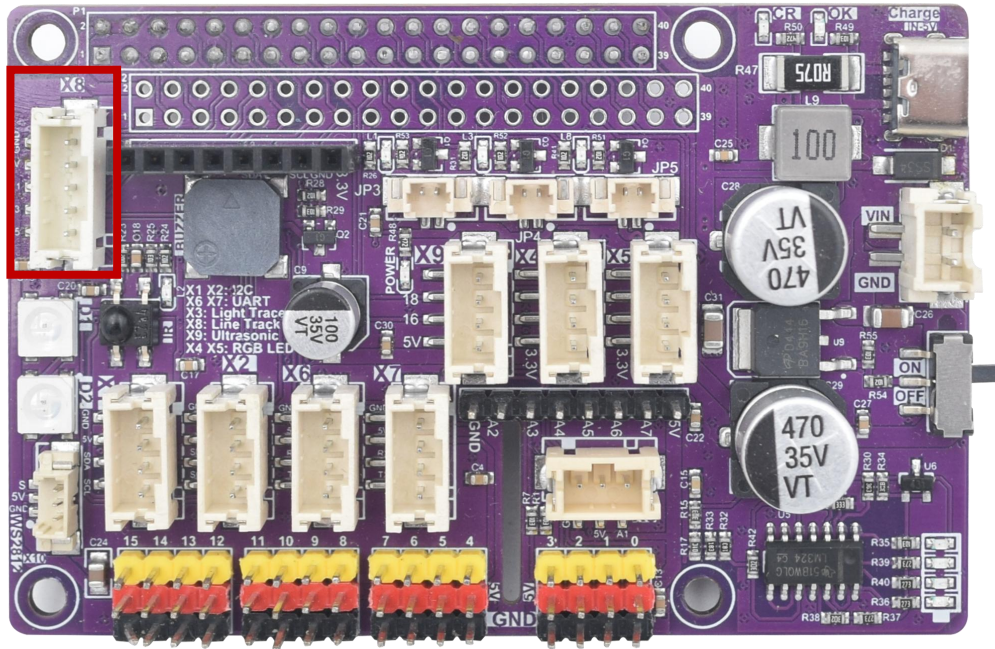
The phototransistors in the module detect the reflected light. When more light is reflected (from the background), the phototransistor conducts more current, and when less light is reflected (from the line), the phototransistor conducts less current. The Raspberry Pi reads the signals from these phototransistors through the Adeept Robot HAT V3.2. By analyzing the state of multiple phototransistors (usually arranged in an array), the Raspberry Pi can determine the position of the line relative to the module. For instance, if the left - most phototransistor detects less light, it indicates that the line is to the left of the module, and appropriate actions can be taken to adjust the direction of a vehicle or device following the line.

PINS of Raspberry Pi	Line Tracking Module
GPIO17	S1
GPIO27	S2
GPIO22	S3
VCC	VCC
GND	GND

13.4 Wiring Diagram

The wiring between the Line Tracking module and the Adeept Robot HAT V3.2 is crucial for proper operation. As shown in the diagrams:





13.5 Demonstration

1. **Remotely log:** Remotely log in to the Raspberry Pi terminal.
2. **Navigate to the Program Folder:** Enter the following command in the terminal and press Enter to access the folder where the program is located:

```
cd Adeept_PiCar-Pro/Examples/07_Line_Tracking/
```

```
pi@raspberrypi:~ $ cd Adeept_PiCar-Pro/Examples/07_Line_Tracking/
pi@raspberrypi:~/Adeept_PiCar-Pro/Examples/07_Line_Tracking $
```

3. **View Directory Contents:** Type "ls" in the terminal and press Enter. This will display all the files in the current directory, ensuring that the "**LineTracking.py**" file is present:

```
ls
```

```
pi@raspberrypi:~/Adeept_PiCar-Pro/Examples/07_Line_Tracking $ ls
LineTracking.py
```

4. **Run the Program:** Enter the command and press **Enter** to run the program:

```
sudo python3 LineTracking.py
```

```
pi@raspberrypi:~/Adeept_PiCar-Pro/Examples/07_Line_Tracking $ sudo python3 LineTracking.py
left: 0   middle: 0   right: 0
left: 0   middle: 0   right: 0
left: 0   middle: 0   right: 0
left: 0   middle: 0   right: 0
left: 1   middle: 1   right: 1
left: 1   middle: 1   right: 1
```

5. **Observation and Termination:** After the program runs successfully, you will see the detected information printed on the console. When a black object is detected or no object is detected, it is recorded as 1, while when a white object is detected, it is recorded as 0. This information can be used to control the steering of the small car and perform other functions. When you want to terminate the running program, you can press the "**Ctrl + C**" shortcut key on the keyboard.

13.6 Code

Complete code refer to [LineTracking.py](#).

```
01  #!/usr/bin/env/python
02  # File name   : LineTracking.py
03  # Website    : www.Adeept.com
04  # Author     : Adeept
05  # Date      : 2025/03/7
06
07  import time
08  from gpiozero import InputDevice
09
10  line_pin_left = 22
11  line_pin_middle = 27
12  line_pin_right = 17
13
14  left = InputDevice(pin=line_pin_left)
15  middle = InputDevice(pin=line_pin_middle)
16  right = InputDevice(pin=line_pin_right)
17
18  def run():
19      status_right = right.value
20      status_middle = middle.value
21      status_left = left.value
22      print('left: %d   middle: %d   right: %d' %(status_left,status_middle,status_right))
23
24
25  if __name__ == '__main__':
```

```
26     try:
27         while 1:
28             run()
29             time.sleep(0.3)
30     except KeyboardInterrupt:
31         pass
```

Code explanation

LineTracking.py

Initialization Stage:

Defining and Configuring Pins: Define S1 = 17, S2 = 27, and S3 = 22 as the pins connected to the sensors on the line - tracking module. Use GPIO.setup to configure these pins as input pins for reading electrical signals.

Loop Control Process:

After entering an infinite loop, execute the following steps in sequence:

Stage 1: Obtaining Sensor Status: In each loop, call the read_sensors function to read the statuses of pins S1, S2, and S3, and determine the position of the line.

Stage 2: Outputting Data and Delaying: Unpack the sensor status returned by the function into left, middle, and right variables, print them to the terminal, and use time.sleep(0.1) to introduce a 0.1 - second delay to prevent oversampling.

Create a loop effect by cyclically reading the electrical signals from the line inspection module and printing them to the terminal.