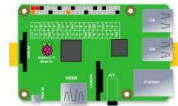
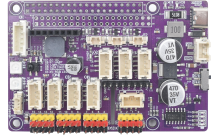



Lesson 15 LED Matrix Display

15.1 Overview

This lesson focuses on the use of a 16*8 LED matrix display. It details the necessary components, working principles, wiring methods, demonstration procedures, and code implementation. Through this lesson, users can learn how to control the LED matrix to display specific patterns, enhancing their understanding of electronic components and programming applications.

15.2 Required Components

Components	Quantity	Picture
Raspberry Pi	1	
Adept Robot HAT V3.2	1	
LED matrix module	1	

15.3 Principle Introduction

A LED matrix is a rectangular display module that consists of a uniform grid of LEDs. The following is an 8X8 monochrome LED matrix containing 64 LEDs (8 rows by 8 columns).



In order to facilitate the operation and reduce the number of ports required to drive this component, the positive poles of the LEDs in each row and negative poles of the LEDs in each column are respectively connected together inside the LED matrix module, which is called a common anode. There is another arrangement type. Negative poles of the LEDs in each row and the positive poles of the LEDs in each column are respectively connected together, which is called a common cathode. The default address of LED matrix is 0x70.

The principle of 8*16 LED matrix:

a byte has 8 bits, each bit is 0 or 1. When a bit is 0, turn off LED and when a bit is 1, turn on LED. Thereby, one byte can control the LED in a columns of dot matrix, so 16 bytes can control 16 columns of led lights, that is, 8*16 dot matrix.

We divide the LED matrix into two sides and display "+" on the left and "o" on the right. As shown below, yellow stands for lit LED while other colors represent the OFF LED.

			1	1						1	1				
			1	1						1			1		
	1	1	1	1	1	1			1					1	
	1	1	1	1	1	1			1					1	
			1	1						1			1		
			1	1							1	1			

Below, the table on the left corresponds to the "+" above, and the table on the right corresponds to the "o" above.

Columns	Binary	Hexadecimal	Columns	Binary	Hexadecimal
1	0000 0000	0x00	9	0000 0000	0x00
2	0001 1000	0x18	10	0001 1000	0x18
3	0001 1000	0x18	11	0010 0100	0x24
4	0111 1110	0x7e	12	0100 0010	0x42
5	0111 1110	0x7e	13	0100 0010	0x42
6	0001 1000	0x18	14	0010 0100	0x24
7	0001 1000	0x18	15	0001 1000	0x18
8	0000 0000	0x00	16	0000 0000	0x00

PINS of Raspberry Pi	OLED
GPIO03(SCL)	SCL
GPIO02(SDA)	SDA
VCC	VCC
GND	GND

15.4 Wiring Diagram

If you want to use the LED matrix module, you need to connect the IIC interface on the Adeept Robot HAT V3.2 driver board, as shown in the figure below:

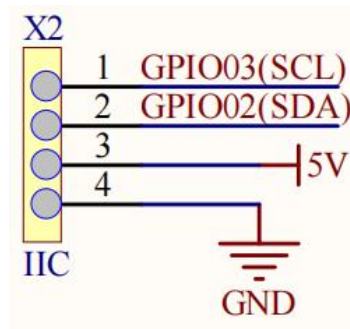
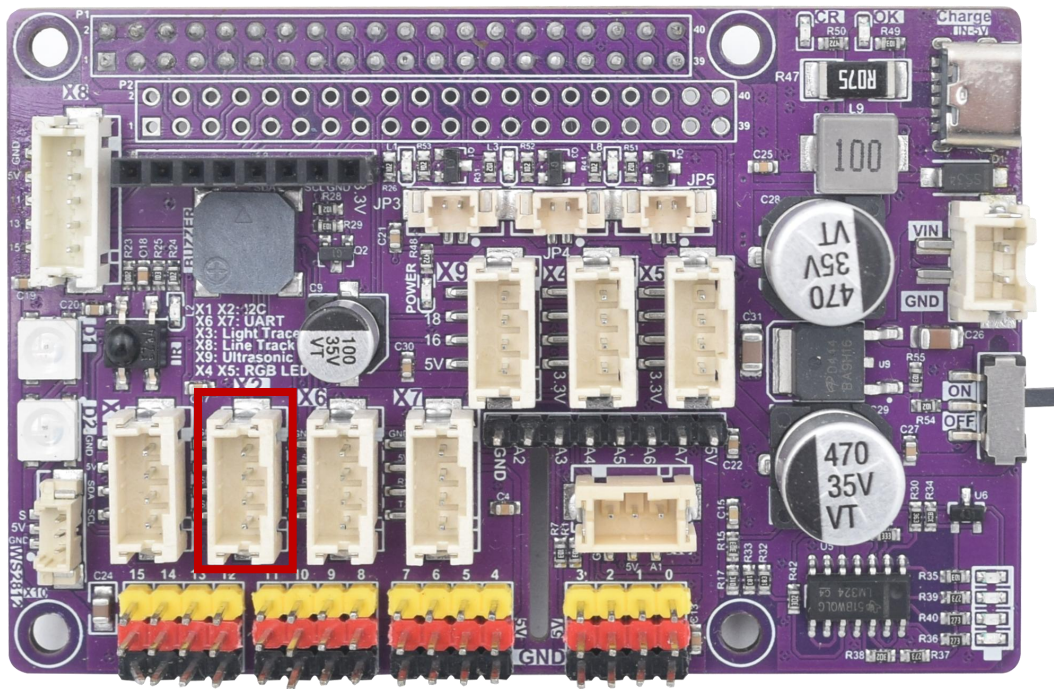


Figure as below :



15.5 Demonstration

Run the code

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_4WD_Smart_Car_for_RPi/Examples/09_LED_Matrix/
```

```
pi@raspberrypi:~ $ cd Adeept_4WD_Smart_Car_for_RPi/Examples/09_LED_Matrix/  
pi@raspberrypi:~/Adeept_4WD_Smart_Car_for_RPi/Examples/09_LED_Matrix $
```

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 "**DynamicsMatrix.py**" and "**StaticMatrix.py**" file is present:

```
ls
```

```
pi@raspberrypi:~/Adeept_4WD_Smart_Car_for_RPi/Examples/09_LED_Matrix $ ls  
DynamicsMatrix.py StaticMatrix.py
```

4. **Run the Program:**Demonstrate how to display a static image.

```
sudo python3 StaticMatrix.py
```

```
pi@raspberrypi:~/Adeept_4WD_Smart_Car_for_RPi/Examples/09_LED_Matrix $ sudo python3 StaticMatrix.py
```

5. **Observation and Termination:**After successfully running the program, You will see the pattern defined by the 'smile_array' array data on the 16*8 matrix LED display. When you want to terminate a running program, you can press the **Ctrl+C** shortcut key on the keyboard.

6. **Run the Program:**Demonstrate how to display a dynamics image.

```
sudo python3 DynamicsMatrix.py
```

```
pi@raspberrypi:~/Adeept_4WD_Smart_Car_for_RPi/Examples/09_LED_Matrix $ sudo python3 DynamicsMatrix.py
```

7. **Observation and Termination:**After successfully running the program, You will see the patterns defined by the data in the 'smile_array1' and 'smile_array2' arrays displayed alternately on the 16*8 matrix LED display. When you want to terminate a running program, you can press the **Ctrl+C** shortcut key on the keyboard.

15.6 Code

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

```
01 #!/usr/bin/env/python
02 # File name   : StaticMatrix.py
03 # Website    : www.Adeept.com
04 # Author     : Adeept
05 # Date      : 2025/07/24
06 import time
07 import board
08 import busio
09 from adafruit_ht16k33 import matrix
10
11 i2c = busio.I2C(board.SCL, board.SDA)
12
13 matrix_display = matrix.Matrix16x8(i2c, address=0x70)
14
15 # Define the pixel data of the smiling face
16 smile_array = bytes([
17     0x00,0x00,0x04,0x02,0x02,0x24,0x40,0x40,
18     0x40,0x40,0x24,0x02,0x02,0x04,0x00,0x00,
19 ])
20
21 # initialization function
22 def setup():
23     matrix_display.fill(0) # Clear the matrix
24     matrix_display.brightness = 1.0 # set brightness (0.0-1.0)
25
26 # Display smiley face function
27 def display_array(array):
28     for i in range(8):
29         matrix_display._buffer[i*2+1] = array[i]
30         matrix_display._buffer[i*2+2] = array[i+8]
31     matrix_display.show()
32
33 def loop():
34     try:
35         while True:
36             display_array(smile_array)
37             time.sleep(1)
38     except KeyboardInterrupt:
39         matrix_display.fill(0)
40         print("\nThe LED matrix has safely stopped")
41
42 if __name__ == "__main__":
43     setup()
44     loop()
```

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

```
01 #!/usr/bin/env/python
02 # File name   : DynamicsMatrix.py
```

```
03 # Website      : www.Adeept.com
04 # Author       : Adeept
05 # Date        : 2025/07/24
06 import time
07 import board
08 import busio
09 from adafruit_ht16k33 import matrix
10
11 i2c = busio.I2C(board.SCL, board.SDA)
12
13 matrix_display = matrix.Matrix16x8(i2c, address=0x70)
14
15 # Define the pixel data of the smiling face
16 smile_array1 = bytes([
17     0x00,0x00,0x04,0x02,0x02,0x24,0x40,0x40,
18     0x40,0x40,0x24,0x02,0x02,0x04,0x00,0x00,
19 ])
20 smile_array2 = bytes([
21     0x00,0x08,0x4,0x02,0x04,0x08,0x00,0x00,
22     0x00,0x00,0x08,0x4,0x02,0x04,0x08,0x00,
23 ])
24
25 # initialization function
26 def setup():
27     matrix_display.fill(0) # Clear the matrix
28     matrix_display.brightness = 1.0 # set brightness (0.0-1.0)
29
30 # Display smiley face function
31 def display_array(array):
32     for i in range(8):
33         matrix_display._buffer[i*2+1] = array[i]
34         matrix_display._buffer[i*2+2] = array[i+8]
35     matrix_display.show()
36
37 def loop():
38     try:
39         while True:
40             display_array(smile_array1)
41             time.sleep(1)
42             display_array(smile_array2)
43             time.sleep(1)
44     except KeyboardInterrupt:
45         matrix_display.fill(0)
46         print("\nThe LED matrix has safely stopped")
47
48 if __name__ == "__main__":
49     setup()
50     loop()
```

Code explanation

StaticMatrix.py

Initialization Stage:

Connect LED matrix module via I2C (address 0x70). Clear the matrix and set the brightness of the display screen to the maximum value of 1.0.

Loop Control Process:

The current pattern data is written to the LED matrix display screen.

[DynamicsMatrix.py](#)

Initialization Stage:

Connect LED matrix module via I2C (address 0x70). Clear the matrix and set the brightness of the display screen to the maximum value of 1.0.

Loop Control Process:

First display the first smiley face, then delay for 1 second, and then display the second smiley face, alternating between them.