

---

# **Adafruit HT16K33 Library Documentation**

***Release 1.0***

**Radomir Dopieralski**

**Jan 25, 2020**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Installing from PyPI</b>	<b>5</b>
<b>3</b>	<b>Usage Example</b>	<b>7</b>
<b>4</b>	<b>Contributing</b>	<b>9</b>
<b>5</b>	<b>Documentation</b>	<b>11</b>
<b>6</b>	<b>Table of Contents</b>	<b>13</b>
6.1	Simple test . . . . .	13
6.2	adafruit_ht16k33.ht16k33 . . . . .	16
6.3	Matrix Displays . . . . .	17
6.4	Segment Displays . . . . .	18
<b>7</b>	<b>Indices and tables</b>	<b>21</b>
	<b>Python Module Index</b>	<b>23</b>
	<b>Index</b>	<b>25</b>



This is a library for using the I<sup>2</sup>C-based LED matrices with the HT16K33 chip. It supports both 16x8 and 8x8 matrices, as well as 7- and 14-segment displays.

- **Notes**

1. This library is intended for Adafruit CircuitPython's API. For a library compatible with MicroPython machine API see this [library](#).
2. This library does not work with the Trellis 4x4 LED+Keypad board. For that product use: [CircuitPython Trellis Library](#)



# CHAPTER 1

---

## Dependencies

---

This driver depends on:

- [Adafruit CircuitPython](#)
- [Bus Device](#)

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the [Adafruit library and driver bundle](#).





## CHAPTER 2

---

### Installing from PyPI

---

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally [from PyPI](#). To install for current user:

```
pip3 install adafruit-circuitpython-ht16k33
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-ht16k33
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-ht16k33
```



## CHAPTER 3

---

### Usage Example

---

```
# Import all board pins and bus interface.
import board
import busio

# Import the HT16K33 LED matrix module.
from adafruit_ht16k33 import matrix

# Create the I2C interface.
i2c = busio.I2C(board.SCL, board.SDA)

# Create the matrix class.
# This creates a 16x8 matrix:
matrix = matrix.Matrix16x8(i2c)
# Or this creates a 8x8 matrix:
#matrix = matrix.Matrix8x8(i2c)
# Or this creates a 8x8 bicolor matrix:
#matrix = matrix.Matrix8x8x2
# Finally you can optionally specify a custom I2C address of the HT16k33 like:
#matrix = matrix.Matrix16x8(i2c, address=0x70)

# Clear the matrix.
matrix.fill(0)

# Set a pixel in the origin 0,0 position.
matrix[0, 0] = 1
# Set a pixel in the middle 8, 4 position.
matrix[8, 4] = 1
# Set a pixel in the opposite 15, 7 position.
matrix[15, 7] = 1
matrix.show()

# Change the brightness
matrix.brightness = 8
```

(continues on next page)

(continued from previous page)

```
# Set the blink rate
matrix.blink_rate = 2
```

## CHAPTER 4

---

### Contributing

---

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.



## CHAPTER 5

---

### Documentation

---

For information on building library documentation, please check out [this guide](#).





---

## Table of Contents

---

### 6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/ht16k33\_matrix\_simpletest.py

```
1  # Basic example of clearing and drawing a pixel on a LED matrix display.
2  # This example and library is meant to work with Adafruit CircuitPython API.
3  # Author: Tony DiCola
4  # License: Public Domain
5
6  # Import all board pins.
7  import time
8  import board
9  import busio
10
11 # Import the HT16K33 LED matrix module.
12 from adafruit_ht16k33 import matrix
13
14
15 # Create the I2C interface.
16 i2c = busio.I2C(board.SCL, board.SDA)
17
18 # Create the matrix class.
19 # This creates a 16x8 matrix:
20 matrix = matrix.Matrix16x8(i2c)
21 # Or this creates a 16x8 matrix backpack:
22 # matrix = matrix.MatrixBackpack16x8(i2c)
23 # Or this creates a 8x8 matrix:
24 #matrix = matrix.Matrix8x8(i2c)
25 # Or this creates a 8x8 bicolor matrix:
26 #matrix = matrix.Matrix8x8x2(i2c)
27 # Finally you can optionally specify a custom I2C address of the HT16k33 like:
```

(continues on next page)

(continued from previous page)

```

28 #matrix = matrix.Matrix16x8(i2c, address=0x70)
29
30 # Clear the matrix.
31 matrix.fill(0)
32
33 # Set a pixel in the origin 0, 0 position.
34 matrix[0, 0] = 1
35 # Set a pixel in the middle 8, 4 position.
36 matrix[8, 4] = 1
37 # Set a pixel in the opposite 15, 7 position.
38 matrix[15, 7] = 1
39
40 time.sleep(2)
41
42 # Draw a Smiley Face
43 matrix.fill(0)
44
45 for row in range(2, 6):
46     matrix[row, 0] = 1
47     matrix[row, 7] = 1
48
49 for column in range(2, 6):
50     matrix[0, column] = 1
51     matrix[7, column] = 1
52
53 matrix[1, 1] = 1
54 matrix[1, 6] = 1
55 matrix[6, 1] = 1
56 matrix[6, 6] = 1
57 matrix[2, 5] = 1
58 matrix[5, 5] = 1
59 matrix[2, 3] = 1
60 matrix[5, 3] = 1
61 matrix[3, 2] = 1
62 matrix[4, 2] = 1
63
64 # Move the Smiley Face Around
65 while True:
66     for frame in range(0, 8):
67         matrix.shift_right(True)
68         time.sleep(0.05)
69     for frame in range(0, 8):
70         matrix.shift_down(True)
71         time.sleep(0.05)
72     for frame in range(0, 8):
73         matrix.shift_left(True)
74         time.sleep(0.05)
75     for frame in range(0, 8):
76         matrix.shift_up(True)
77         time.sleep(0.05)

```

Listing 2: examples/ht16k33\_segments\_simpletest.py

```

1 # Basic example of setting digits on a LED segment display.
2 # This example and library is meant to work with Adafruit CircuitPython API.
3 # Author: Tony DiCola

```

(continues on next page)

(continued from previous page)

```

4  # License: Public Domain
5
6  import time
7
8  # Import all board pins.
9  import board
10 import busio
11
12 # Import the HT16K33 LED segment module.
13 from adafruit_ht16k33 import segments
14
15 # Create the I2C interface.
16 i2c = busio.I2C(board.SCL, board.SDA)
17
18 # Create the LED segment class.
19 # This creates a 7 segment 4 character display:
20 display = segments.Seg7x4(i2c)
21 # Or this creates a 14 segment alphanumeric 4 character display:
22 #display = segments.Seg14x4(i2c)
23 # Finally you can optionally specify a custom I2C address of the HT16k33 like:
24 #display = segments.Seg7x4(i2c, address=0x70)
25
26 # Clear the display.
27 display.fill(0)
28
29 # Can just print a number
30 display.print(42)
31 time.sleep(2)
32
33 # Or, can print a hexadecimal value
34 display.print_hex(0xFF23)
35 time.sleep(2)
36
37 # Or, can set individual digits / characters
38 # Set the first character to '1':
39 display[0] = '1'
40 # Set the second character to '2':
41 display[1] = '2'
42 # Set the third character to 'A':
43 display[2] = 'A'
44 # Set the forth character to 'B':
45 display[3] = 'B'
46 time.sleep(2)
47
48 # Or, can even set the segments to make up characters
49 if isinstance(display, segments.Seg7x4):
50     # 7-segment raw digits
51     display.set_digit_raw(0, 0xFF)
52     display.set_digit_raw(1, 0b11111111)
53     display.set_digit_raw(2, 0x79)
54     display.set_digit_raw(3, 0b01111001)
55 else:
56     # 14-segment raw digits
57     display.set_digit_raw(0, 0x2D3F)
58     display.set_digit_raw(1, 0b0010110100111111)
59     display.set_digit_raw(2, (0b00101101, 0b00111111))
60     display.set_digit_raw(3, [0x2D, 0x3F])

```

(continues on next page)

(continued from previous page)

```
61 time.sleep(2)
62
63 #Show a looping marquee
64 display.marquee('Deadbeef 192.168.100.102... ', 0.2)
```

Listing 3: examples/ht16k33\_bicolor24\_simpletest.py

```
1  # Basic example of using the Bi-color 24 segment bargraph display.
2  # This example and library is meant to work with Adafruit CircuitPython API.
3  # Author: Carter Nelson
4  # License: Public Domain
5
6  import time
7
8  # Import board related modules
9  import board
10 import busio
11
12 # Import the Bicolor24 driver from the HT16K33 module
13 from adafruit_ht16k33.bargraph import Bicolor24
14
15 # Create the I2C interface
16 i2c = busio.I2C(board.SCL, board.SDA)
17
18 # Create the LED bargraph class.
19 bc24 = Bicolor24(i2c)
20
21 # Set individual segments of bargraph
22 bc24[0] = bc24.LED_RED
23 bc24[1] = bc24.LED_GREEN
24 bc24[2] = bc24.LED_YELLOW
25
26 time.sleep(2)
27
28 # Turn them all off
29 bc24.fill(bc24.LED_OFF)
30
31 # Turn them on in a loop
32 for i in range(24):
33     bc24[i] = bc24.LED_RED
34     time.sleep(0.1)
35     bc24[i] = bc24.LED_OFF
36
37 time.sleep(1)
38
39 # Fill the entire bargraph
40 bc24.fill(bc24.LED_GREEN)
```

## 6.2 adafruit\_ht16k33.ht16k33

- Authors: Radomir Dopieralski & Tony DiCola for Adafruit Industries

**class** adafruit\_ht16k33.ht16k33.**HT16K33** (*i2c, address=112, auto\_write=True*)  
The base class for all displays. Contains common methods.

**Parameters**

- **address** (*int*) – The I2C address of the HT16K33.
- **auto\_write** (*bool*) – True if the display should immediately change when set. If False, *show* must be called explicitly.

**auto\_write**

Auto write updates to the display.

**blink\_rate**

The blink rate. Range 0-3.

**brightness**

The brightness. Range 0-15.

**fill** (*color*)

Fill the whole display with the given color.

**show** ()

Refresh the display and show the changes.

## 6.3 Matrix Displays

```
class adafruit_ht16k33.matrix.Matrix16x8 (i2c, address=112, auto_write=True)
```

The matrix wing.

```
pixel (x, y, color=None)
```

Get or set the color of a given pixel.

```
class adafruit_ht16k33.matrix.Matrix8x8 (i2c, address=112, auto_write=True)
```

A single matrix.

**columns**

Read-only property for number of columns

```
image (img)
```

Set buffer to value of Python Imaging Library image. The image should be in 1 bit mode and a size equal to the display size.

```
pixel (x, y, color=None)
```

Get or set the color of a given pixel.

**rows**

Read-only property for number of rows

```
shift (x, y, rotate=False)
```

Shift pixels by x and y

**Parameters rotate** – (Optional) Rotate the shifted pixels to the left side (default=False)

```
shift_down (rotate=False)
```

Shift all pixels down

**Parameters rotate** – (Optional) Rotate the shifted pixels to top (default=False)

```
shift_left (rotate=False)
```

Shift all pixels left

**Parameters rotate** – (Optional) Rotate the shifted pixels to the right side (default=False)

**shift\_right** (*rotate=False*)

Shift all pixels right

**Parameters** **rotate** – (Optional) Rotate the shifted pixels to the left side (default=False)

**shift\_up** (*rotate=False*)

Shift all pixels up

**Parameters** **rotate** – (Optional) Rotate the shifted pixels to bottom (default=False)

**class** adafruit\_ht16k33.matrix.**Matrix8x8x2** (*i2c, address=112, auto\_write=True*)

A bi-color matrix.

**fill** (*color*)

Fill the whole display with the given color.

**image** (*img*)

Set buffer to value of Python Imaging Library image. The image should be a size equal to the display size.

**pixel** (*x, y, color=None*)

Get or set the color of a given pixel.

**class** adafruit\_ht16k33.matrix.**MatrixBackpack16x8** (*i2c, address=112, auto\_write=True*)

A double matrix backpack.

**pixel** (*x, y, color=None*)

Get or set the color of a given pixel.

## 6.4 Segment Displays

**class** adafruit\_ht16k33.segments.**BigSeg7x4** (*i2c, address=112, auto\_write=True*)

Numeric 7-segment display. It has the same methods as the alphanumeric display, but only supports displaying a limited set of characters.

**ampm**

The AM/PM indicator.

**bottom\_left\_dot**

The bottom-left dot indicator.

**top\_left\_dot**

The top-left dot indicator.

**class** adafruit\_ht16k33.segments.**Colon** (*disp, num\_of\_colons=1*)

Helper class for controlling the colons. Not intended for direct use.

**class** adafruit\_ht16k33.segments.**Seg14x4** (*i2c, address=112, auto\_write=True*)

Alpha-numeric, 14-segment display.

**marquee** (*text, delay=0.25, loop=True*)

Automatically scroll the text at the specified delay between characters

**Parameters**

- **text** (*str*) – The text to display
- **delay** (*float*) – (optional) The delay in seconds to pause before scrolling to the next character (default=0.25)
- **loop** (*bool*) – (optional) Whether to endlessly loop the text (default=True)

**print** (*value*)

Print the value to the display.

**print\_hex** (*value*)

Print the value as a hexadecimal string to the display.

**scroll** (*count=1*)

Scroll the display by specified number of places.

**set\_digit\_raw** (*index, bitmask*)

Set digit at position to raw bitmask value. Position should be a value of 0 to 3 with 0 being the left most character on the display.

bitmask should be 2 bytes such as: 0xFFFF If can be passed as an integer, list, or tuple

**class** `adafruit_ht16k33.segments.Seg7x4` (*i2c, address=112, auto\_write=True*)

Numeric 7-segment display. It has the same methods as the alphanumeric display, but only supports displaying a limited set of characters.

**scroll** (*count=1*)

Scroll the display by specified number of places.

**set\_digit\_raw** (*index, bitmask*)

Set digit at position to raw bitmask value. Position should be a value of 0 to 3 with 0 being the left most digit on the display.





## CHAPTER 7

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### **a**

`adafruit_ht16k33.ht16k33`, [16](#)  
`adafruit_ht16k33.matrix`, [17](#)  
`adafruit_ht16k33.segments`, [18](#)



## A

adafruit\_ht16k33.ht16k33 (module), 16  
adafruit\_ht16k33.matrix (module), 17  
adafruit\_ht16k33.segments (module), 18  
ampm (adafruit\_ht16k33.segments.BigSeg7x4 attribute), 18  
auto\_write (adafruit\_ht16k33.ht16k33.HT16K33 attribute), 17

## B

BigSeg7x4 (class in adafruit\_ht16k33.segments), 18  
blink\_rate (adafruit\_ht16k33.ht16k33.HT16K33 attribute), 17  
bottom\_left\_dot (adafruit\_ht16k33.segments.BigSeg7x4 attribute), 18  
brightness (adafruit\_ht16k33.ht16k33.HT16K33 attribute), 17

## C

Colon (class in adafruit\_ht16k33.segments), 18  
columns (adafruit\_ht16k33.matrix.Matrix8x8 attribute), 17

## F

fill() (adafruit\_ht16k33.ht16k33.HT16K33 method), 17  
fill() (adafruit\_ht16k33.matrix.Matrix8x8x2 method), 18

## H

HT16K33 (class in adafruit\_ht16k33.ht16k33), 16

## I

image() (adafruit\_ht16k33.matrix.Matrix8x8 method), 17  
image() (adafruit\_ht16k33.matrix.Matrix8x8x2 method), 18

## M

marquee() (adafruit\_ht16k33.segments.Seg14x4 method), 18  
Matrix16x8 (class in adafruit\_ht16k33.matrix), 17  
Matrix8x8 (class in adafruit\_ht16k33.matrix), 17  
Matrix8x8x2 (class in adafruit\_ht16k33.matrix), 18  
MatrixBackpack16x8 (class in adafruit\_ht16k33.matrix), 18

## P

pixel() (adafruit\_ht16k33.matrix.Matrix16x8 method), 17  
pixel() (adafruit\_ht16k33.matrix.Matrix8x8 method), 17  
pixel() (adafruit\_ht16k33.matrix.Matrix8x8x2 method), 18  
pixel() (adafruit\_ht16k33.matrix.MatrixBackpack16x8 method), 18  
print() (adafruit\_ht16k33.segments.Seg14x4 method), 18  
print\_hex() (adafruit\_ht16k33.segments.Seg14x4 method), 19

## R

rows (adafruit\_ht16k33.matrix.Matrix8x8 attribute), 17

## S

scroll() (adafruit\_ht16k33.segments.Seg14x4 method), 19  
scroll() (adafruit\_ht16k33.segments.Seg7x4 method), 19  
Seg14x4 (class in adafruit\_ht16k33.segments), 18  
Seg7x4 (class in adafruit\_ht16k33.segments), 19  
set\_digit\_raw() (adafruit\_ht16k33.segments.Seg14x4 method), 19  
set\_digit\_raw() (adafruit\_ht16k33.segments.Seg7x4 method), 19  
shift() (adafruit\_ht16k33.matrix.Matrix8x8 method), 17

`shift_down()` (*adafruit\_ht16k33.matrix.Matrix8x8*  
*method*), [17](#)  
`shift_left()` (*adafruit\_ht16k33.matrix.Matrix8x8*  
*method*), [17](#)  
`shift_right()` (*adafruit\_ht16k33.matrix.Matrix8x8*  
*method*), [17](#)  
`shift_up()` (*adafruit\_ht16k33.matrix.Matrix8x8*  
*method*), [18](#)  
`show()` (*adafruit\_ht16k33.ht16k33.HT16K33* *method*),  
[17](#)

## T

`top_left_dot` (*adafruit\_ht16k33.segments.BigSeg7x4*  
*attribute*), [18](#)