
AdafruitTrellis Library Documentation

Release 1.0

Michael Schroeder

Jul 09, 2020

Contents

1	Dependencies	3
2	Installing from PyPI	5
3	Usage Example	7
4	Contributing	9
5	Documentation	11
6	Table of Contents	13
6.1	Simple test	13
6.2	adafruit_trellis - Adafruit Trellis Monochrome 4x4 LED Backlit Keypad	14
6.2.1	Implementation Notes	15
7	Indices and tables	19
	Python Module Index	21
	Index	23

This library will allow you to control the LEDs and read button presses on the [Adafruit Trellis Board](#). It will work with a single Trellis board, or with a matrix of up to 8 Trellis boards.

For more details, see the [Adafruit Trellis Learn Guide](#).

CHAPTER 1

Dependencies

This driver depends on:

- [Adafruit CircuitPython 2.0.0+](#)
- [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-trellis
```

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

```
sudo pip3 install adafruit-circuitpython-trellis
```

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-trellis
```


CHAPTER 3

Usage Example

See `examples/trellis_simpletest.py` for full usage example.

```
import time
import busio
from board import SCL, SDA
from adafruit_trellis import Trellis

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

# Create a Trellis object for each board
trellis = Trellis(i2c) # 0x70 when no I2C address is supplied

# Turn on every LED
print('Turning all LEDs on...')
trellis.led.fill(True)
time.sleep(2)

# Turn off every LED
print('Turning all LEDs off...')
trellis.led.fill(False)
time.sleep(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](#).

6.1 Simple test

Ensure your device works with this simple test.

Single Adafruit Trellis Board

Listing 1: examples/trellis_simpletest.py

```
1  # Basic example of turning on LEDs and handling Keypad
2  # button activity.
3
4  # This example uses only one Trellis board, so all loops assume
5  # a maximum of 16 LEDs (0-15). For use with multiple Trellis boards,
6  # see the documentation.
7
8  import time
9  import busio
10 from board import SCL, SDA
11 from adafruit_trellis import Trellis
12
13 # Create the I2C interface
14 i2c = busio.I2C(SCL, SDA)
15
16 # Create a Trellis object
17 trellis = Trellis(i2c) # 0x70 when no I2C address is supplied
18
19 # 'auto_show' defaults to 'True', so anytime LED states change,
20 # the changes are automatically sent to the Trellis board. If you
21 # set 'auto_show' to 'False', you will have to call the 'show()'
22 # method afterwards to send updates to the Trellis board.
23
24 # Turn on every LED
25 print("Turning all LEDs on...")
```

(continues on next page)

(continued from previous page)

```

26 trellis.led.fill(True)
27 time.sleep(2)
28
29 # Turn off every LED
30 print("Turning all LEDs off...")
31 trellis.led.fill(False)
32 time.sleep(2)
33
34 # Turn on every LED, one at a time
35 print("Turning on each LED, one at a time...")
36 for i in range(16):
37     trellis.led[i] = True
38     time.sleep(0.1)
39
40 # Turn off every LED, one at a time
41 print("Turning off each LED, one at a time...")
42 for i in range(15, 0, -1):
43     trellis.led[i] = False
44     time.sleep(0.1)
45
46 # Now start reading button activity
47 # - When a button is depressed (just_pressed),
48 #   the LED for that button will turn on.
49 # - When the button is released (released),
50 #   the LED will turn off.
51 # - Any button that is still depressed (pressed_buttons),
52 #   the LED will remain on.
53 print("Starting button sensory loop...")
54 pressed_buttons = set()
55 while True:
56     # Make sure to take a break during each trellis.read_buttons
57     # cycle.
58     time.sleep(0.1)
59
60     just_pressed, released = trellis.read_buttons()
61     for b in just_pressed:
62         print("pressed:", b)
63         trellis.led[b] = True
64     pressed_buttons.update(just_pressed)
65     for b in released:
66         print("released:", b)
67         trellis.led[b] = False
68     pressed_buttons.difference_update(released)
69     for b in pressed_buttons:
70         print("still pressed:", b)
71         trellis.led[b] = True

```

6.2 adafruit_trellis - Adafruit Trellis Monochrome 4x4 LED Backlit Keypad

CircuitPython library to support Adafruit's Trellis Keypad.

- **Author(s):** Limor Fried, Radomir Dopieralski, Tony DiCola, Scott Shawcroft, and Michael Schroeder

6.2.1 Implementation Notes

Hardware:

- Adafruit Trellis Monochrome 4x4 LED Backlit Keypad (Product ID: 1616)

Software and Dependencies:

- Adafruit CircuitPython firmware (2.2.0+) for the ESP8622 and M0-based boards: <https://github.com/adafruit/circuitpython/releases>
- Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice

class adafruit_trellis.**Trellis** (*i2c*, *addresses=None*)
Driver base for a single Trellis Board

Parameters

- **i2c** (*I2C*) – The `busio.I2C` object to use. This is the only required parameter when using a single Trellis board.
- **addresses** (*list*) – The I2C address(es) of the Trellis board(s) you're using. Defaults to `[0x70]` which is the default address for Trellis boards. See Trellis product guide for using different/multiple I2C addresses. <https://learn.adafruit.com/adafruit-trellis-diy-open-source-led-keypad>

Listing 2: Usage Example

```

1  # Basic example of turning on LEDs and handling Keypad
2  # button activity.
3
4  # This example uses only one Trellis board, so all loops assume
5  # a maximum of 16 LEDs (0-15). For use with multiple Trellis boards,
6  # see the documentation.
7
8  import time
9  import busio
10 from board import SCL, SDA
11 from adafruit_trellis import Trellis
12
13 # Create the I2C interface
14 i2c = busio.I2C(SCL, SDA)
15
16 # Create a Trellis object
17 trellis = Trellis(i2c) # 0x70 when no I2C address is supplied
18
19 # 'auto_show' defaults to 'True', so anytime LED states change,
20 # the changes are automatically sent to the Trellis board. If you
21 # set 'auto_show' to 'False', you will have to call the 'show()'
22 # method afterwards to send updates to the Trellis board.
23
24 # Turn on every LED
25 print("Turning all LEDs on...")
26 trellis.led.fill(True)
27 time.sleep(2)
28
29 # Turn off every LED
30 print("Turning all LEDs off...")
31 trellis.led.fill(False)
32 time.sleep(2)

```

(continues on next page)

(continued from previous page)

```

33
34 # Turn on every LED, one at a time
35 print("Turning on each LED, one at a time...")
36 for i in range(16):
37     trellis.led[i] = True
38     time.sleep(0.1)
39
40 # Turn off every LED, one at a time
41 print("Turning off each LED, one at a time...")
42 for i in range(15, 0, -1):
43     trellis.led[i] = False
44     time.sleep(0.1)
45
46 # Now start reading button activity
47 # - When a button is depressed (just_pressed),
48 #   the LED for that button will turn on.
49 # - When the button is released (released),
50 #   the LED will turn off.
51 # - Any button that is still depressed (pressed_buttons),
52 #   the LED will remain on.
53 print("Starting button sensory loop...")
54 pressed_buttons = set()
55 while True:
56     # Make sure to take a break during each trellis.read_buttons
57     # cycle.
58     time.sleep(0.1)
59
60     just_pressed, released = trellis.read_buttons()
61     for b in just_pressed:
62         print("pressed:", b)
63         trellis.led[b] = True
64     pressed_buttons.update(just_pressed)
65     for b in released:
66         print("released:", b)
67         trellis.led[b] = False
68     pressed_buttons.difference_update(released)
69     for b in pressed_buttons:
70         print("still pressed:", b)
71         trellis.led[b] = True

```

auto_show

Current state of sending LED updates directly the Trellis board(s). True or False.

blink_rate

The current blink rate as an integer range 0-3.

brightness

The current brightness as an integer range 0-15.

led = None

The LED object used to interact with Trellis LEDs.

- `trellis.led[x]` returns the current LED status of LED x (True/False)
- `trellis.led[x] = True` turns the LED at x on
- `trellis.led[x] = False` turns the LED at x off
- `trellis.led.fill(bool)` turns every LED on (True) or off (False)

read_buttons()

Read the button matrix register on the Trellis board(s). Returns two lists: 1 for new button presses, 1 for button releases.

show()

Refresh the LED buffer and show the changes.

CHAPTER 7

Indices and tables

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

a

adafruit_trellis, [14](#)

A

`adafruit_trellis` (*module*), 14

`auto_show` (*adafruit_trellis.Trellis attribute*), 16

B

`blink_rate` (*adafruit_trellis.Trellis attribute*), 16

`brightness` (*adafruit_trellis.Trellis attribute*), 16

L

`led` (*adafruit_trellis.Trellis attribute*), 16

R

`read_buttons()` (*adafruit_trellis.Trellis method*), 16

S

`show()` (*adafruit_trellis.Trellis method*), 17

T

`Trellis` (*class in adafruit_trellis*), 15