
AdafruitFingerprint Library Documentation

Release 1.0

ladyada

Dec 21, 2018

Contents

1	Dependencies	3
2	Usage Example	5
3	API Reference	7
3.1	<code>adafruit_fingerprint</code>	7
4	Contributing	9
5	Building locally	11
5.1	Sphinx documentation	11
6	Table of Contents	13
6.1	Simple test	13
7	Indices and tables	17
	Python Module Index	19

This library will let you use an Adafruit Fingerprint sensor on any UART to get, store, retrieve and query fingerprints! Great for adding bio-sensing security to your next build.

CHAPTER 1

Dependencies

This driver depends on:

- [Adafruit CircuitPython](#)

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

CHAPTER 2

Usage Example

See 'examples' folder for full usage demo!

3.1 adafruit_fingerprint

This library will let you use an Adafruit Fingerprint sensor on any UART to get, store, retrieve and query fingerprints! Great for adding bio-sensing security to your next build.

- Author(s): ladyada

3.1.1 Implementation Notes

Hardware:

- [Fingerprint sensor](#) (Product ID: 751)

Software and Dependencies:

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

class `adafruit_fingerprint.Adafruit_Fingerprint` (*uart, passwd=(0, 0, 0, 0)*)

UART based fingerprint sensor.

count_templates ()

Requests the sensor to count the number of templates and stores it in `self.template_count`. Returns the packet error code or OK success

create_model ()

Requests the sensor take the template data and turn it into a model returns the packet error code or OK success

delete_model (*location*)

Requests the sensor delete a model from flash memory given by the argument location. Returns the packet error code or OK success

`finger_fast_search()`

Asks the sensor to search for a matching fingerprint template to the last model generated. Stores the location and confidence in `self.finger_id` and `self.confidence`. Returns the packet error code or OK success

`get_image()`

Requests the sensor to take an image and store it memory, returns the packet error code or OK success

`image_2_tz(slot)`

Requests the sensor convert the image to a template, returns the packet error code or OK success

`read_templates()`

Requests the sensor to list of all template locations in use and stores them in `self.templates`. Returns the packet error code or OK success

`store_model(location)`

Requests the sensor store the model into flash memory and assign a location. Returns the packet error code or OK success

`verify_password()`

Checks if the password/connection is correct, returns True/False

CHAPTER 4

Contributing

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

CHAPTER 5

Building locally

To build this library locally you'll need to install the `circuitpython-build-tools` package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-fingerprint --
↳library_location .
```

5.1 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the `index.html` in your browser to view them. It will also (due to `-W`) error out on any warning like Travis will. This is a good way to locally verify it will pass.

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/fingerprint_simpletest.py

```
1 import time
2 import board
3 import busio
4 from digitalio import DigitalInOut, Direction
5 import adafruit_fingerprint
6
7 led = DigitalInOut(board.D13)
8 led.direction = Direction.OUTPUT
9
10 uart = busio.UART(board.TX, board.RX, baudrate=57600)
11
12 # If using with a computer such as Linux/RaspberryPi, Mac, Windows...
13 #import serial
14 #uart = serial.Serial("/dev/ttyUSB0", baudrate=57600, timeout=1)
15
16 finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)
17
18 #####
19
20
21 def get_fingerprint():
22     """Get a finger print image, template it, and see if it matches!"""
23     print("Waiting for image...")
24     while finger.get_image() != adafruit_fingerprint.OK:
25         pass
26     print("Templating...")
27     if finger.image_2_tz(1) != adafruit_fingerprint.OK:
```

(continues on next page)

(continued from previous page)

```

28     return False
29     print("Searching...")
30     if finger.finger_fast_search() != adafruit_fingerprint.OK:
31         return False
32     return True
33
34 # pylint: disable=too-many-branches
35 def get_fingerprint_detail():
36     """Get a finger print image, template it, and see if it matches!
37     This time, print out each error instead of just returning on failure"""
38     print("Getting image...", end="", flush=True)
39     i = finger.get_image()
40     if i == adafruit_fingerprint.OK:
41         print("Image taken")
42     else:
43         if i == adafruit_fingerprint.NO_FINGER:
44             print("No finger detected")
45         elif i == adafruit_fingerprint.IMAGE_FAIL:
46             print("Imaging error")
47         else:
48             print("Other error")
49     return False
50
51     print("Templating...", end="", flush=True)
52     i = finger.image_2_tz(1)
53     if i == adafruit_fingerprint.OK:
54         print("Templated")
55     else:
56         if i == adafruit_fingerprint.IMAGE_MESS:
57             print("Image too messy")
58         elif i == adafruit_fingerprint.FEATURE_FAIL:
59             print("Could not identify features")
60         elif i == adafruit_fingerprint.INVALID_IMAGE:
61             print("Image invalid")
62         else:
63             print("Other error")
64     return False
65
66     print("Searching...", end="", flush=True)
67     i = finger.finger_fast_search()
68     # pylint: disable=no-else-return
69     # This block needs to be refactored when it can be tested.
70     if i == adafruit_fingerprint.OK:
71         print("Found fingerprint!")
72         return True
73     else:
74         if i == adafruit_fingerprint.NOT_FOUND:
75             print("No match found")
76         else:
77             print("Other error")
78     return False
79
80 # pylint: disable=too-many-statements
81 def enroll_finger(location):
82     """Take a 2 finger images and template it, then store in 'location'"""
83     for fingerimg in range(1, 3):
84         if fingerimg == 1:

```

(continues on next page)

(continued from previous page)

```

85     print("Place finger on sensor...", end="", flush=True)
86 else:
87     print("Place same finger again...", end="", flush=True)
88
89 while True:
90     i = finger.get_image()
91     if i == adafruit_fingerprint.OK:
92         print("Image taken")
93         break
94     elif i == adafruit_fingerprint.NOFINGER:
95         print(".", end="", flush=True)
96     elif i == adafruit_fingerprint.IMAGEFAIL:
97         print("Imaging error")
98         return False
99     else:
100         print("Other error")
101         return False
102
103 print("Templating...", end="", flush=True)
104 i = finger.image_2_tz(fingerimg)
105 if i == adafruit_fingerprint.OK:
106     print("Templated")
107 else:
108     if i == adafruit_fingerprint.IMAGEMESS:
109         print("Image too messy")
110     elif i == adafruit_fingerprint.FEATUREFAIL:
111         print("Could not identify features")
112     elif i == adafruit_fingerprint.INVALIDIMAGE:
113         print("Image invalid")
114     else:
115         print("Other error")
116     return False
117
118 if fingerimg == 1:
119     print("Remove finger")
120     time.sleep(1)
121     while i != adafruit_fingerprint.NOFINGER:
122         i = finger.get_image()
123
124 print("Creating model...", end="", flush=True)
125 i = finger.create_model()
126 if i == adafruit_fingerprint.OK:
127     print("Created")
128 else:
129     if i == adafruit_fingerprint.ENROLLMISMATCH:
130         print("Prints did not match")
131     else:
132         print("Other error")
133     return False
134
135 print("Storing model #%d..." % location, end="", flush=True)
136 i = finger.store_model(location)
137 if i == adafruit_fingerprint.OK:
138     print("Stored")
139 else:
140     if i == adafruit_fingerprint.BADLOCATION:
141         print("Bad storage location")

```

(continues on next page)

(continued from previous page)

```

142     elif i == adafruit_fingerprint.FLASHERR:
143         print("Flash storage error")
144     else:
145         print("Other error")
146     return False
147
148     return True
149
150
151 #####
152
153 def get_num():
154     """Use input() to get a valid number from 1 to 127. Retry till success!"""
155     i = 0
156     while (i > 127) or (i < 1):
157         try:
158             i = int(input("Enter ID # from 1-127: "))
159         except ValueError:
160             pass
161     return i
162
163
164 while True:
165     print("-----")
166     if finger.read_templates() != adafruit_fingerprint.OK:
167         raise RuntimeError('Failed to read templates')
168     print("Fingerprint templates:", finger.templates)
169     print("e) enroll print")
170     print("f) find print")
171     print("d) delete print")
172     print("-----")
173     c = input("> ")
174
175     if c == 'e':
176         enroll_finger(get_num())
177     if c == 'f':
178         if get_fingerprint():
179             print("Detected #", finger.finger_id, "with confidence", finger.
180             ↪confidence)
181         else:
182             print("Finger not found")
183     if c == 'd':
184         if finger.delete_model(get_num()) == adafruit_fingerprint.OK:
185             print("Deleted!")
186         else:
187             print("Failed to delete")

```

CHAPTER 7

Indices and tables

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

a

`adafruit_fingerprint`, [7](#)

A

Adafruit_Fingerprint (class in adafruit_fingerprint), 7
adafruit_fingerprint (module), 7

C

count_templates() (adafruit_fingerprint.Adafruit_Fingerprint
method), 7
create_model() (adafruit_fingerprint.Adafruit_Fingerprint
method), 7

D

delete_model() (adafruit_fingerprint.Adafruit_Fingerprint
method), 7

F

finger_fast_search() (adafruit_fingerprint.Adafruit_Fingerprint
method), 7

G

get_image() (adafruit_fingerprint.Adafruit_Fingerprint
method), 8

I

image_2_tz() (adafruit_fingerprint.Adafruit_Fingerprint
method), 8

R

read_templates() (adafruit_fingerprint.Adafruit_Fingerprint
method), 8

S

store_model() (adafruit_fingerprint.Adafruit_Fingerprint
method), 8

V

verify_password() (adafruit_fingerprint.Adafruit_Fingerprint
method), 8