
AdafruitFingerprint Library Documentation

Release 1.0

ladyada

Mar 09, 2021

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_fingerprint	17
6.2.1	Implementation Notes	17
7	Indices and tables	19
	Python Module Index	21
	Index	23

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

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-fingerprint
```

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

```
sudo pip3 install adafruit-circuitpython-fingerprint
```

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-fingerprint
```


CHAPTER 3

Usage Example

See 'examples' folder for full usage demo!

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.

Listing 1: examples/fingerprint_simpletest.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import time
5  import board
6  import busio
7  from digitalio import DigitalInOut, Direction
8  import adafruit_fingerprint
9
10 led = DigitalInOut(board.D13)
11 led.direction = Direction.OUTPUT
12
13 uart = busio.UART(board.TX, board.RX, baudrate=57600)
14
15 # If using with a computer such as Linux/RaspberryPi, Mac, Windows with USB/serial_
16 ↪ converter:
17 # import serial
18 # uart = serial.Serial("/dev/ttyUSB0", baudrate=57600, timeout=1)
19
20 # If using with Linux/Raspberry Pi and hardware UART:
21 # import serial
22 # uart = serial.Serial("/dev/ttyS0", baudrate=57600, timeout=1)
23
24 finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)
25
26 #####

```

(continues on next page)

(continued from previous page)

```

27
28 def get_fingerprint():
29     """Get a finger print image, template it, and see if it matches!"""
30     print("Waiting for image...")
31     while finger.get_image() != adafruit_fingerprint.OK:
32         pass
33     print("Templating...")
34     if finger.image_2_tz(1) != adafruit_fingerprint.OK:
35         return False
36     print("Searching...")
37     if finger.finger_search() != adafruit_fingerprint.OK:
38         return False
39     return True
40
41
42 # pylint: disable=too-many-branches
43 def get_fingerprint_detail():
44     """Get a finger print image, template it, and see if it matches!
45     This time, print out each error instead of just returning on failure"""
46     print("Getting image...", end="", flush=True)
47     i = finger.get_image()
48     if i == adafruit_fingerprint.OK:
49         print("Image taken")
50     else:
51         if i == adafruit_fingerprint.NO_FINGER:
52             print("No finger detected")
53         elif i == adafruit_fingerprint.IMAGEFAIL:
54             print("Imaging error")
55         else:
56             print("Other error")
57         return False
58
59     print("Templating...", end="", flush=True)
60     i = finger.image_2_tz(1)
61     if i == adafruit_fingerprint.OK:
62         print("Templated")
63     else:
64         if i == adafruit_fingerprint.IMAGEMESS:
65             print("Image too messy")
66         elif i == adafruit_fingerprint.FEATUREFAIL:
67             print("Could not identify features")
68         elif i == adafruit_fingerprint.INVALIDIMAGE:
69             print("Image invalid")
70         else:
71             print("Other error")
72         return False
73
74     print("Searching...", end="", flush=True)
75     i = finger.finger_fast_search()
76     # pylint: disable=no-else-return
77     # This block needs to be refactored when it can be tested.
78     if i == adafruit_fingerprint.OK:
79         print("Found fingerprint!")
80         return True
81     else:
82         if i == adafruit_fingerprint.NOTFOUND:
83             print("No match found")

```

(continues on next page)

(continued from previous page)

```

84         else:
85             print("Other error")
86         return False
87
88
89 # pylint: disable=too-many-statements
90 def enroll_finger(location):
91     """Take a 2 finger images and template it, then store in 'location'"""
92     for fingerimg in range(1, 3):
93         if fingerimg == 1:
94             print("Place finger on sensor...", end="", flush=True)
95         else:
96             print("Place same finger again...", end="", flush=True)
97
98         while True:
99             i = finger.get_image()
100             if i == adafruit_fingerprint.OK:
101                 print("Image taken")
102                 break
103             if i == adafruit_fingerprint.NO_FINGER:
104                 print(".", end="", flush=True)
105             elif i == adafruit_fingerprint.IMAGE_FAIL:
106                 print("Imaging error")
107                 return False
108             else:
109                 print("Other error")
110                 return False
111
112         print("Templating...", end="", flush=True)
113         i = finger.image_2_tz(fingerimg)
114         if i == adafruit_fingerprint.OK:
115             print("Templated")
116         else:
117             if i == adafruit_fingerprint.IMAGE_MESS:
118                 print("Image too messy")
119             elif i == adafruit_fingerprint.FEATURE_FAIL:
120                 print("Could not identify features")
121             elif i == adafruit_fingerprint.INVALID_IMAGE:
122                 print("Image invalid")
123             else:
124                 print("Other error")
125             return False
126
127         if fingerimg == 1:
128             print("Remove finger")
129             time.sleep(1)
130             while i != adafruit_fingerprint.NO_FINGER:
131                 i = finger.get_image()
132
133         print("Creating model...", end="", flush=True)
134         i = finger.create_model()
135         if i == adafruit_fingerprint.OK:
136             print("Created")
137         else:
138             if i == adafruit_fingerprint.ENROLL_MISMATCH:
139                 print("Prints did not match")
140             else:

```

(continues on next page)

(continued from previous page)

```

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

```

6.2 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

6.2.1 Implementation Notes

Hardware:

- [Fingerprint sensor](#) (Product ID: 751)
- [Panel Mount Fingerprint sensor](#) (Product ID: 4651)

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.

check_module ()

Checks the state of the fingerprint scanner module. Returns OK or error.

close_uart ()

close serial port

compare_templates ()

Compares two fingerprint templates in char buffers 1 and 2. Stores the confidence score in `self.finger_id` and `self.confidence`. Returns the packet error code or OK success

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

empty_library ()

Requests the sensor to delete all models from flash memory. 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

finger_search ()

Asks the sensor to search for a matching fingerprint starting at slot 1. Stores the location and confidence in `self.finger_id` and `self.confidence`. Returns the packet error code or OK success

get_fpdata (*sensorbuffer='char', slot=1*)

Requests the sensor to transfer the fingerprint image or template. Returns the data payload only.

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=1*)

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

load_model (*location, slot=1*)

Requests the sensor to load a model from the given memory location to the given slot. Returns the packet error code or success

read_sysparam ()

Returns the system parameters on success via attributes.

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

send_fpdata (*data, sensorbuffer='char', slot=1*)

Requests the sensor to receive data, either a fingerprint image or a character/template data. Data is the payload only.

set_led (*color=1, mode=3, speed=128, cycles=0*)

LED function – only for R503 Sensor. Parameters: See User Manual for full details color: 1=red, 2=blue, 3=purple mode: 1-breathe, 2-flash, 3-on, 4-off, 5-fade_on, 6-fade-off speed: animation speed 0-255 cycles: numbe of time to repeat 0=infinite or 1-255 Returns the packet error code or success

set_sysparam (*param_num, param_val*)

Set the system parameters (param_num)

soft_reset ()

Performs a soft reset of the sensor

store_model (*location, slot=1*)

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 7

Indices and tables

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

a

`adafruit_fingerprint`, [16](#)

A

Adafruit_Fingerprint (class
 adafruit_fingerprint), 17
adafruit_fingerprint (module), 16

C

check_module() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17
close_uart() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17
compare_templates()
 (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17
count_templates()
 (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17
create_model() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17

D

delete_model() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17

E

empty_library() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17

F

finger_fast_search()
 (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17
finger_search() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17

G

get_fpdata() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17
get_image() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17

I

in_image_2_tz() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 17

L

load_model() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 18

R

read_sysparam() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 18
read_templates() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 18

S

send_fpdata() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 18
set_led() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 18
set_sysparam() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 18
soft_reset() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 18
store_model() (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 18

V

verify_password()
 (*adafruit_fingerprint.Adafruit_Fingerprint*
 method), 18