AdafruitFingerprint Library Documentation

Release 1.0

ladyada

Oct 25, 2021

Contents

This library will let you use an Adafruit Fingerprint sensor on any UART to get, store, retreive 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

Documentation

API documentation for this library can be found on Read the Docs.

CHAPTER 5

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 6

Documentation

For information on building library documentation, please check out this guide.

CHAPTER 7

Table of Contents

7.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/fingerprint_simpletest.py

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import busio
from digitalio import DigitalInOut, Direction
import adafruit_fingerprint

led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

uart = busio.UART(board.TX, board.RX, baudrate=57600)

# If using with a computer such as Linux/RaspberryPi, Mac, Windows with USB/serial␣
↪converter:
# import serial
# uart = serial.Serial("/dev/ttyUSB0", baudrate=57600, timeout=1)

# If using with Linux/Raspberry Pi and hardware UART:
# import serial
# uart = serial.Serial("/dev/ttyS0", baudrate=57600, timeout=1)

finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)

##################################################

```

(continues on next page)

(continued from previous page)

```

def get_fingerprint():
    """Get a finger print image, template it, and see if it matches!"""
    print("Waiting for image...")
    while finger.get_image() != adafruit_fingerprint.OK:
        pass
    print("Templating...")
    if finger.image_2_tz(1) != adafruit_fingerprint.OK:
        return False
    print("Searching...")
    if finger.finger_search() != adafruit_fingerprint.OK:
        return False
    return True


# pylint: disable=too-many-branches
def get_fingerprint_detail():
    """Get a finger print image, template it, and see if it matches!
    This time, print out each error instead of just returning on failure"""
    print("Getting image...", end="", flush=True)
    i = finger.get_image()
    if i == adafruit_fingerprint.OK:
        print("Image taken")
    else:
        if i == adafruit_fingerprint.NOFINGER:
            print("No finger detected")
        elif i == adafruit_fingerprint.IMAGEFAIL:
            print("Imaging error")
        else:
            print("Other error")
        return False

    print("Templating...", end="", flush=True)
    i = finger.image_2_tz(1)
    if i == adafruit_fingerprint.OK:
        print("Templated")
    else:
        if i == adafruit_fingerprint.IMAGEMESS:
            print("Image too messy")
        elif i == adafruit_fingerprint.FEATUREFAIL:
            print("Could not identify features")
        elif i == adafruit_fingerprint.INVALIDIMAGE:
            print("Image invalid")
        else:
            print("Other error")
        return False

    print("Searching...", end="", flush=True)
    i = finger.finger_fast_search()
    # pylint: disable=no-else-return
    # This block needs to be refactored when it can be tested.
    if i == adafruit_fingerprint.OK:
        print("Found fingerprint!")
        return True
    else:
        if i == adafruit_fingerprint.NOTFOUND:
            print("No match found")
```

(continues on next page)

(continued from previous page)

```
        else:
            print("Other error")
        return False


# pylint: disable=too-many-statements
def enroll_finger(location):
    """Take a 2 finger images and template it, then store in 'location'"""
    for fingerimg in range(1, 3):
        if fingerimg == 1:
            print("Place finger on sensor...", end="", flush=True)
        else:
            print("Place same finger again...", end="", flush=True)

        while True:
            i = finger.get_image()
            if i == adafruit_fingerprint.OK:
                print("Image taken")
                break
            if i == adafruit_fingerprint.NOFINGER:
                print(".", end="", flush=True)
            elif i == adafruit_fingerprint.IMAGEFAIL:
                print("Imaging error")
                return False
            else:
                print("Other error")
                return False

        print("Templating...", end="", flush=True)
        i = finger.image_2_tz(fingerimg)
        if i == adafruit_fingerprint.OK:
            print("Templated")
        else:
            if i == adafruit_fingerprint.IMAGEMESS:
                print("Image too messy")
            elif i == adafruit_fingerprint.FEATUREFAIL:
                print("Could not identify features")
            elif i == adafruit_fingerprint.INVALIDIMAGE:
                print("Image invalid")
            else:
                print("Other error")
            return False

        if fingerimg == 1:
            print("Remove finger")
            time.sleep(1)
            while i != adafruit_fingerprint.NOFINGER:
                i = finger.get_image()

    print("Creating model...", end="", flush=True)
    i = finger.create_model()
    if i == adafruit_fingerprint.OK:
        print("Created")
    else:
        if i == adafruit_fingerprint.ENROLLMISMATCH:
            print("Prints did not match")
        else:
```

(continues on next page)

(continued from previous page)

```
            print("Other error")
        return False

    print("Storing model #%d..." % location, end="", flush=True)
    i = finger.store_model(location)
    if i == adafruit_fingerprint.OK:
        print("Stored")
    else:
        if i == adafruit_fingerprint.BADLOCATION:
            print("Bad storage location")
        elif i == adafruit_fingerprint.FLASHERR:
            print("Flash storage error")
        else:
            print("Other error")
        return False

    return True


##################################################


def get_num():
    """Use input() to get a valid number from 1 to 127. Retry till success!"""
    i = 0
    while (i > 127) or (i < 1):
        try:
            i = int(input("Enter ID # from 1-127: "))
        except ValueError:
            pass
    return i


while True:
    print("----------------")
    if finger.read_templates() != adafruit_fingerprint.OK:
        raise RuntimeError("Failed to read templates")
    print("Fingerprint templates:", finger.templates)
    print("e) enroll print")
    print("f) find print")
    print("d) delete print")
    print("----------------")
    c = input("> ")

    if c == "e":
        enroll_finger(get_num())
    if c == "f":
        if get_fingerprint():
            print("Detected #", finger.finger_id, "with confidence", finger.
↪confidence)
        else:
            print("Finger not found")
    if c == "d":
        if finger.delete_model(get_num()) == adafruit_fingerprint.OK:
            print("Deleted!")
        else:
            print("Failed to delete")
```

7.2 adafruit_fingerprint

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

- Author(s): ladyada

7.2.1 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 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index