AdafruitFingerprint Library Documentation

Release 1.0

ladyada

Aug 25, 2018

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

Usage Example

See ‘examples’ folder for full usage demo!

CHAPTER 3

API Reference

3.1 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

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.

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/fingerprint_simpletest.py

```
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...
#import serial
#uart = serial.Serial("/dev/ttyUSB0", baudrate=57600, timeout=1)

finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)

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


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:
```

(continues on next page)

(continued from previous page)

```
        return False
    print("Searching...")
    if finger.finger_fast_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")
        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:
```

(continues on next page)

(continued from previous page)

```
            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
            elif 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:
            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")
```

(continues on next page)

(continued from previous page)

```
        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")
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index