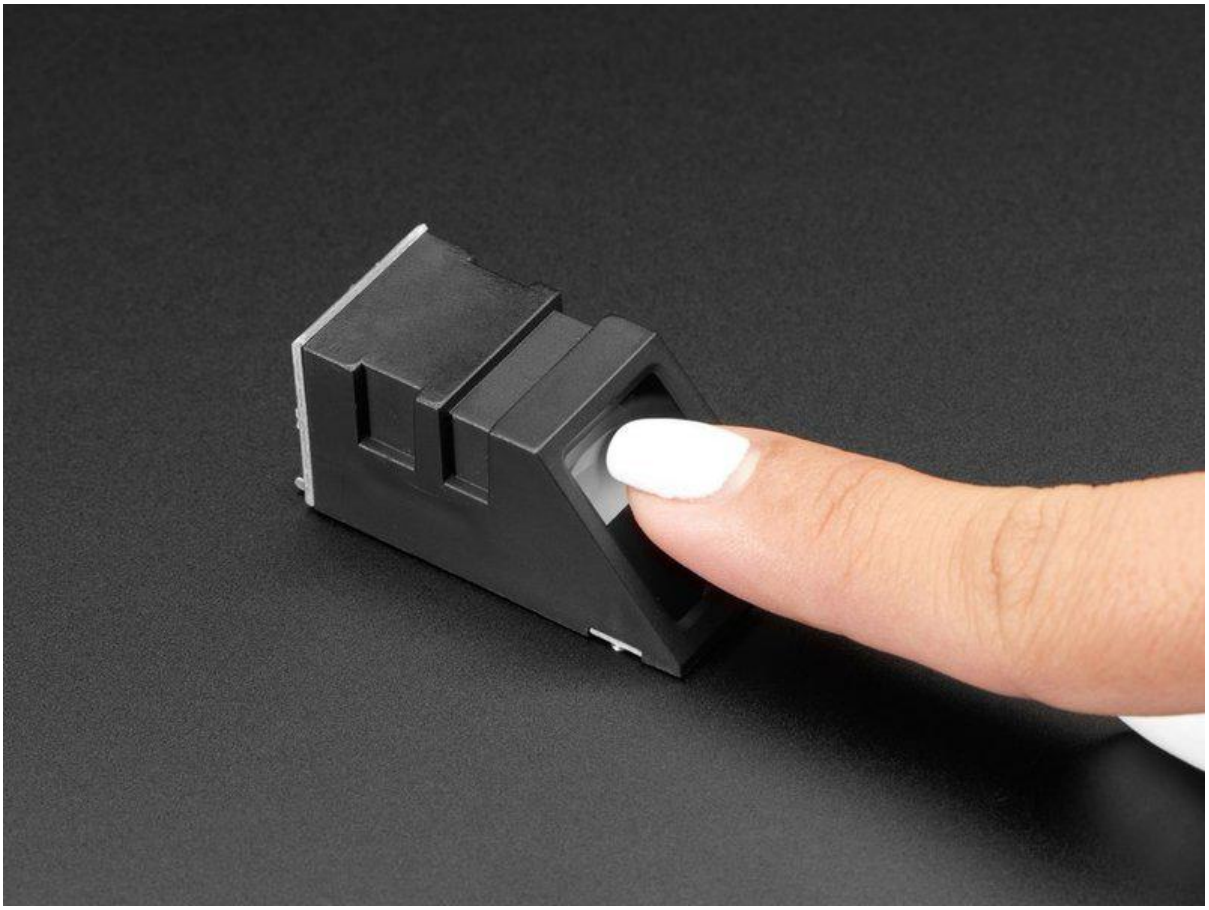
Adafruit Optical Fingerprint Sensor

Created by lady ada



https://learn.adafruit.com/adafruit-optical-fingerprint-sensor

Last updated on 2022-12-01 01:53:20 PM EST

Table of Contents

Overview

Secure your project with biometrics - this all-in-one optical fingerprint sensor will make adding fingerprint detection and verification super simple. These modules are typically used in safes - there's a high powered DSP chip that does the image rendering, calculation, feature-finding and searching. Connect to any microcontroller or system with TTL serial, and send packets of data to take photos, detect prints, hash and search. You can also enroll new fingers directly - up to 162 finger prints can be stored in the onboard FLASH memory.

We like this particular sensor because not only is it easy to use, it also comes with fairly straight-forward Windows software that makes testing the module simple - you can even enroll using the software and see an image of the fingerprint on your computer screen. But, of course, we wouldn't leave you a datasheet and a "good luck!" - we wrote a full Arduino library so that you can get running in under 10 minutes. The library can enroll and search so its perfect for any project (). We've also written a detailed tutorial on wiring and use (). This is by far the best fingerprint sensor you can get.

- Supply voltage: 3.6 - 6.0VDC
- Operating current: 120mA max
- Peak current: 150mA max
- Fingerprint imaging time: <1.0 seconds
- Window area: 14mm x 18mm
- Signature file: 256 bytes
- Template file: 512 bytes

- Storage capacity: 162 templates
- Safety ratings (1-5 low to high safety)
- False Acceptance Rate: <0.001% (Security level 3)
- False Reject Rate: <1.0% (Security level 3)
- Interface: TTL Serial
- Baud rate: 9600, 19200, 28800, 38400, 57600 (default is 57600)
- Working temperature rating: -20C to +50C
- Working humidy: 40%-85% RH
- Full Dimensions: 56 x 20 x 21.5mm
- Exposed Dimensions (when placed in box): 21mm x 21mm x 21mm triangular
- Weight: 20 grams

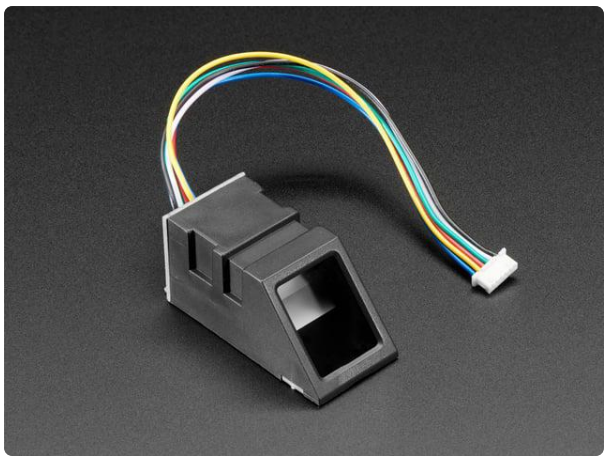
Enrolling vs. Searching

There are basically two requirements for using the optical fingerprint sensor. First is you'll need to enroll fingerprints - that means assigning ID #'s to each print so you can query them later. Once you've enrolled all your prints, you can easily 'search' the sensor, asking it to identify which ID (if any) is currently being photographed.

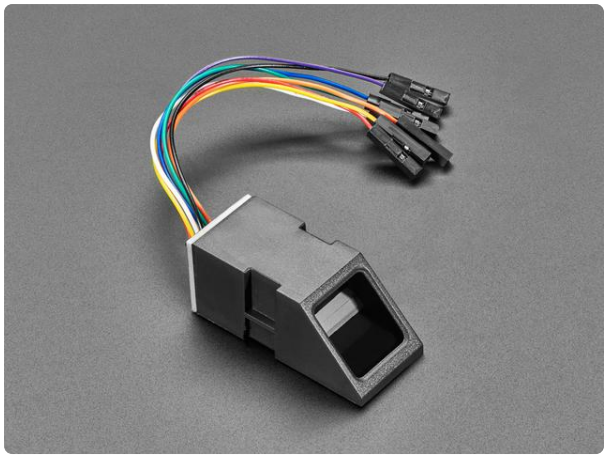
You can enroll using the Windows software (easiest and neat because it shows you the photograph of the print) or with the Arduino sketch (good for when you don't have a Windows machine handy or for on-the-road enrolling).

Enrolling New Users with Windows

The easiest way to enroll a new fingerprint is to use the Windows software. The interface/test software is unfortunately windows-only and the fingerprint image preview section only seems to work with these sensors:



Fingerprint sensor
Secure your project with biometrics - this all-in-one optical fingerprint sensor will make adding fingerprint detection and verification super simple. These modules are typically used...
https://www.adafruit.com/product/751



[Basic Fingerprint Sensor With Socket Header Cable](https://www.adafruit.com/product/4690)
Secure your project with biometrics - this all-in-one optical fingerprint sensor will make adding fingerprint detection and verification super simple. It's easy to use and more...
https://www.adafruit.com/product/4690

but you only need to use it once to enroll, to get the fingerprint you want stored in the module.

First up, you'll want to connect the sensor to the computer via a USB-serial converter. The easiest way to do this is to connect it directly to the USB/Serial converter in the Arduino. To do this, you'll need to upload a 'blank sketch' this one works well for "traditional" Arduinos, like the Uno and the Mega:

```
// this sketch will allow you to bypass the Atmega chip
// and connect the fingerprint sensor directly to the USB/Serial
// chip converter.

// Red connects to +5V
// Black connects to Ground
// White goes to Digital 0
// Green goes to Digital 1

void setup() {}
void loop() {}
```

The "blank" sketch won't work for "native USB" based Arduinos like the Leonardo, Micro, Zero, etc! Use the Leo_passthru sketch instead!

If you're using a Leonardo, Micro, Yun, Zero, or other native-USB device like ATSAMD21 or ATmega32U4-based controller, use the Leo_passthru sketch instead of the "blank" sketch.

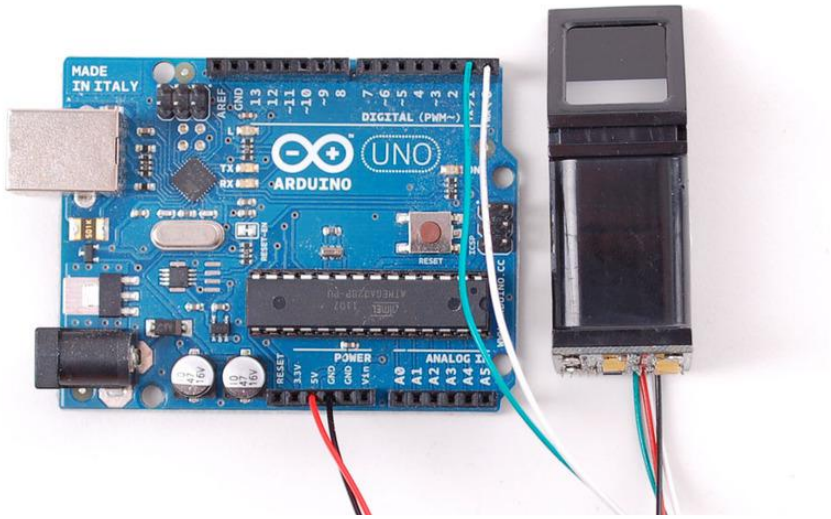
```
//Leo_passthru
// Allows Leonardo to pass serial data between fingerprint reader and Windows.
//
// Red connects to +5V
// Black connects to Ground
// Green goes to Digital 0
// White goes to Digital 1

void setup() {
  // put your setup code here, to run once:
  Serial1.begin(57600);
  Serial.begin(57600);
```

```
}

void loop() {

  while (Serial.available())
    Serial1.write(Serial.read());
  while (Serial1.available())
    Serial.write(Serial1.read());
}
```

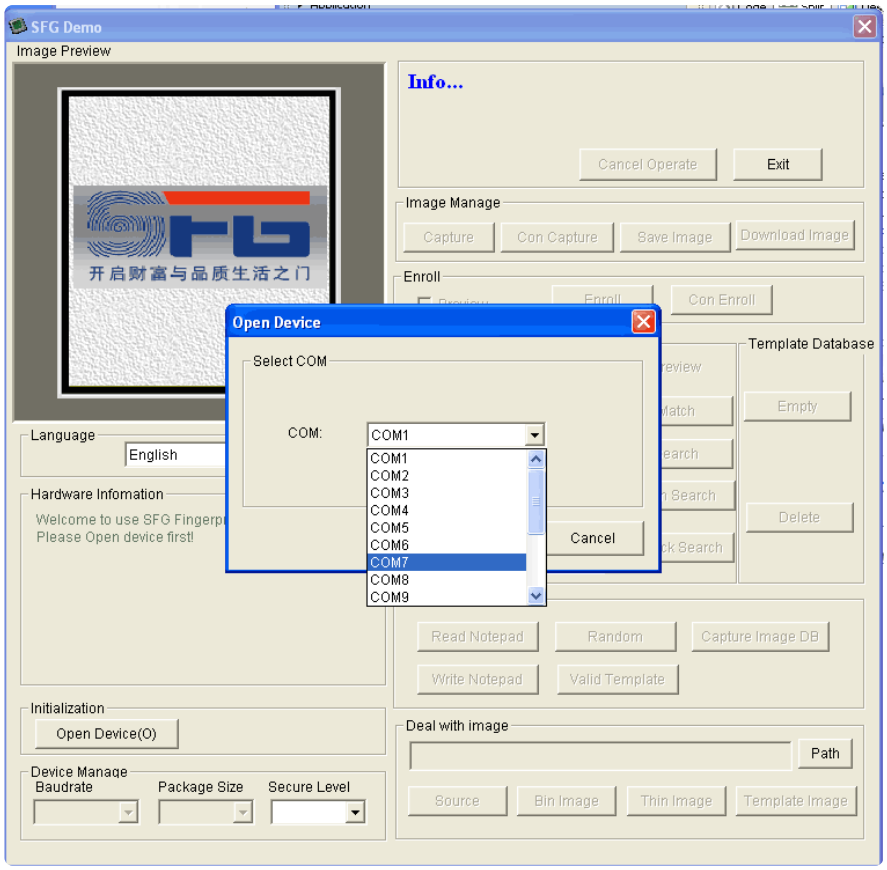
Wire up the sensor as described in the sketch comments after uploading the sketch. Since the sensor wires are so thin and short, we stripped the wire a bit and melted some solder on so it made better contact but you may want to solder the wires to header or similar if you're not getting good contact. When you plug in the power, you may see the LED blink to indicate the sensor is working.

Check the Arduino wiring page for the different wire colors for each kind of sensor ()

You'll connect to the hardware RX / TX pins on the microcontroller:

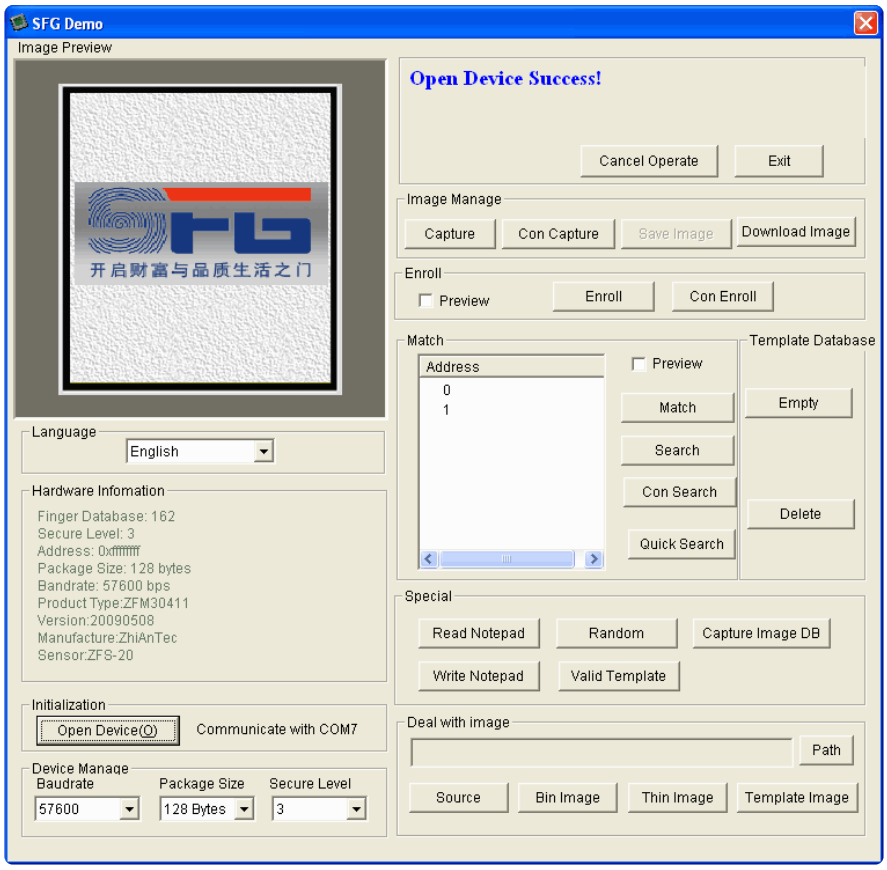


Start up the SFGDemo software and click Open Device from the bottom left corner. Select the COM port used by the Arduino.

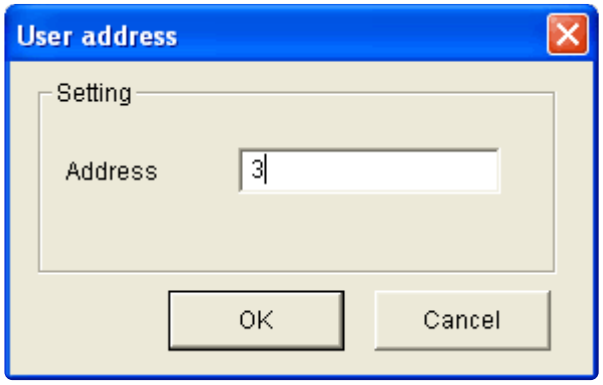


And press OK when done. You should see the following, with a blue success message and some device statistics in the bottom corner. You can change the baud rate in the bottom left hand corner, as well as the "security level" (how sensitive it is) but we suggest leaving those alone until you have everything running and you want to experiment. They should default to 57600 baud and security level 3 so set them if they're wrong

If you get an error when you Open Device, check your wiring, try swapping the RX and TX wires on the sensor, that's a common mixup!

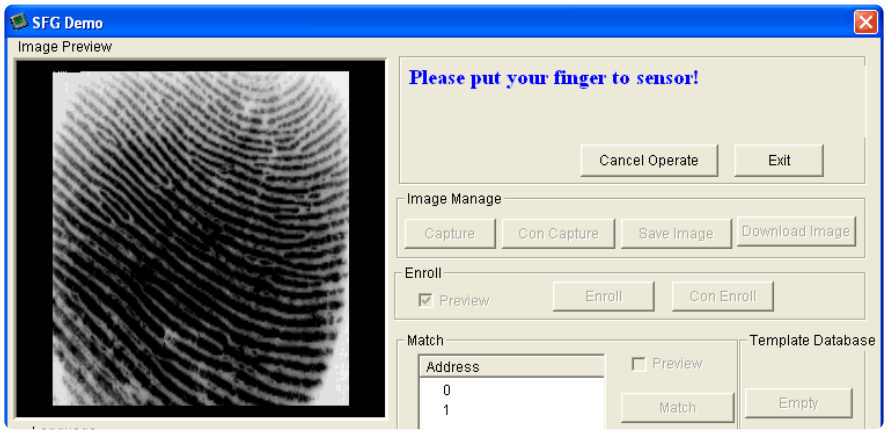


Lets enroll a new finger! Click the Preview checkbox and press the Enroll button next to it (Con Enroll means 'Continuous' enroll, which you may want to do if you have many fingers to enroll). When the box comes up, enter in the ID # you want to use. You can use up to 162 ID numbers.



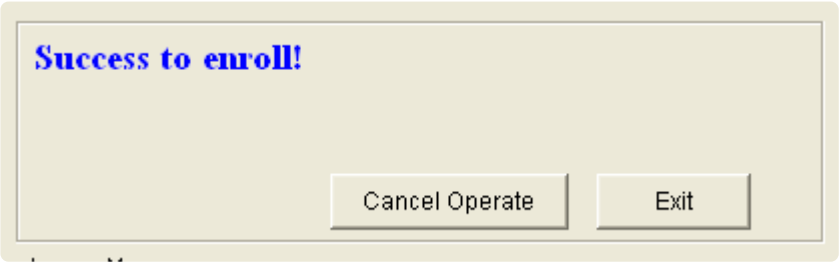
The software will ask you to press the finger to the sensor

You can then see a preview (if you cliecked the preview checkbox) of the fingerprint.



You will then have to repeat the process, to get a second clean print. Use the same finger!

On success you will get a notice.



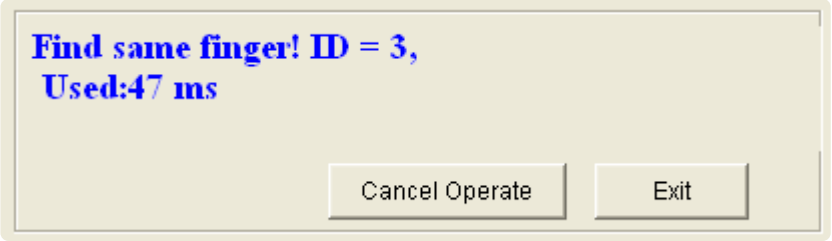
If there's a problem such as a bad print or image, you'll have to do it again.

Searching with the Software

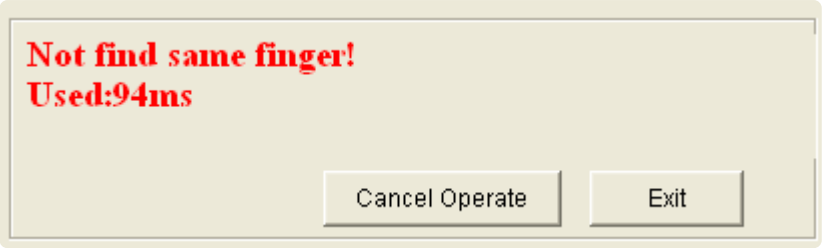
Once you have the finger enrolled, it's a good idea to do a quick test to make sure it can be found in the database. Click on the Search button on the right hand side.

When prompted, press a different/same finger to the sensor.

If it is the same finger, you should get a match with the ID #



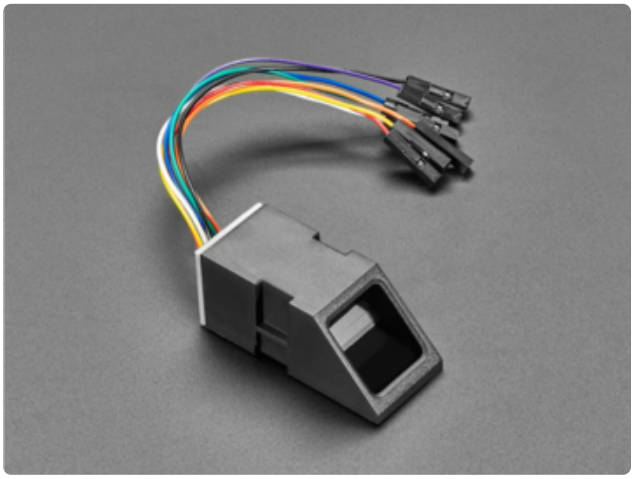
If it is not a finger in the database, you will get a failure notice.

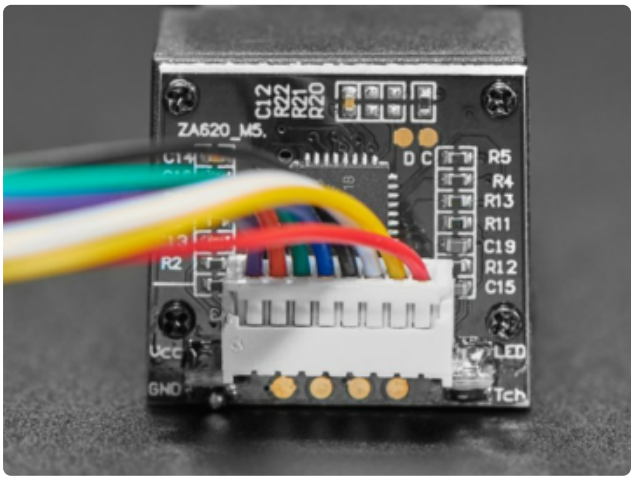


Wiring for use with Arduino

Once you've tested the sensor, you can now use it within a sketch to verify a fingerprint. We'll need to rewire the sensor. Disconnect the green and white wires and plug the green wire into digital 2 and the white wire to digital 3. (For ESP8266 use 4 & 5, for devices with Hardware UART use 0 & 1)

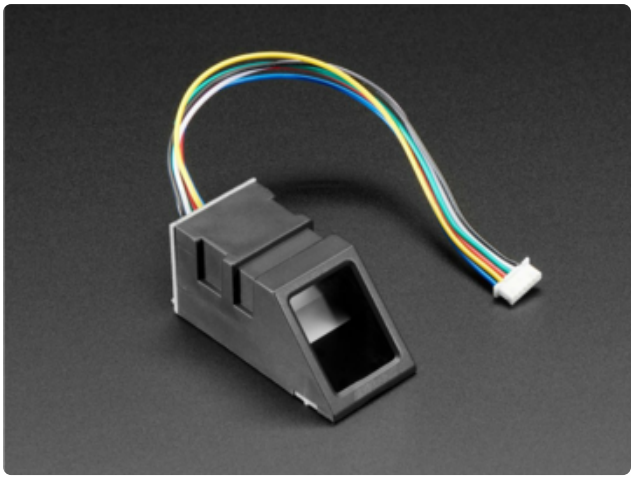
It is normal for the sensor to blink the LED quickly once powered, after that the LED may stay off until you've started to request data from it





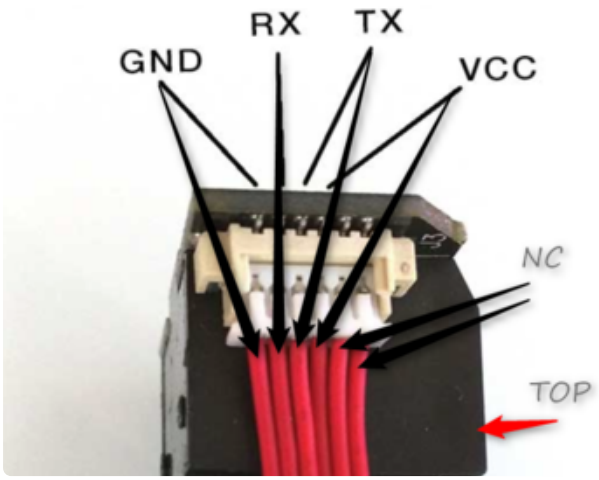
If your fingerprint sensor has individual socket wires (its this one) () then use the following wire setup:

Red Wire to 3.3V
Yellow wire is Serial TX
White wire is Serial RX
Black wire is ground



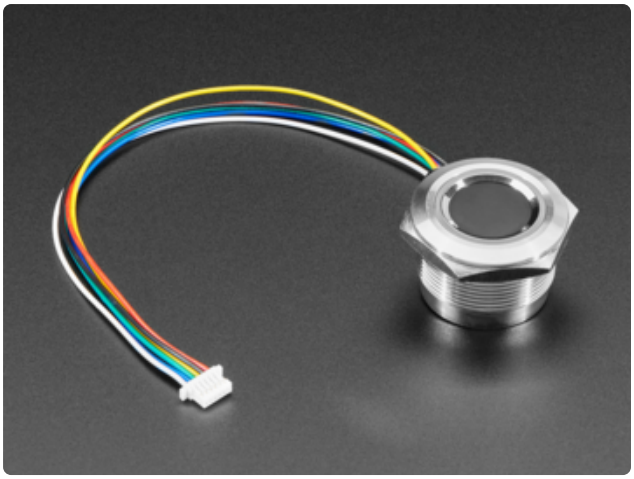
If your cable has a single slim connector on the end and has different color wires:

The first wire from the left should be the black wire ground
then the two data pins: Serial RX is the white wire
Serial TX is the green wire
Then the red power wire (3 or 5V)
You'll have to cut, strip and solder the wires.



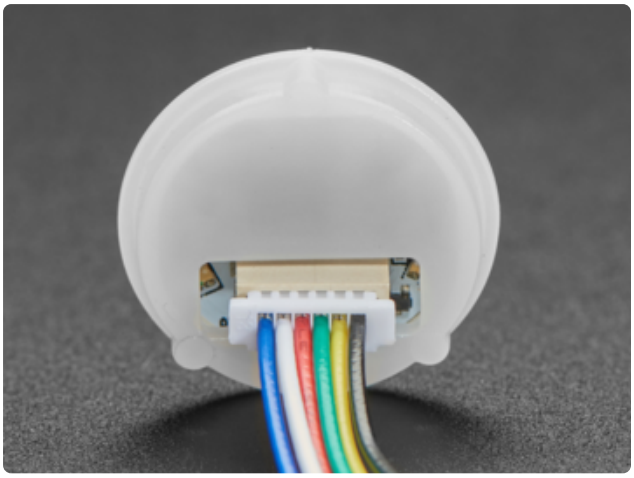
If your sensor is an older one and has all the same-color wires, The first wire from the left is ground, then the two data pins, then power. You'll have to cut, strip and solder the wires.

RX is the same as the White wire
TX is the same as the Green wire



On the "rugged" fingerprint sensor:

Red wire is connected to 3.3V for power
Black wire is connected to ground
Yellow to Microcontroller TX (data out from microcontroller)
Green to Microcontroller RX (data in from microcontroller)



On the 'slim' fingerprint sensor

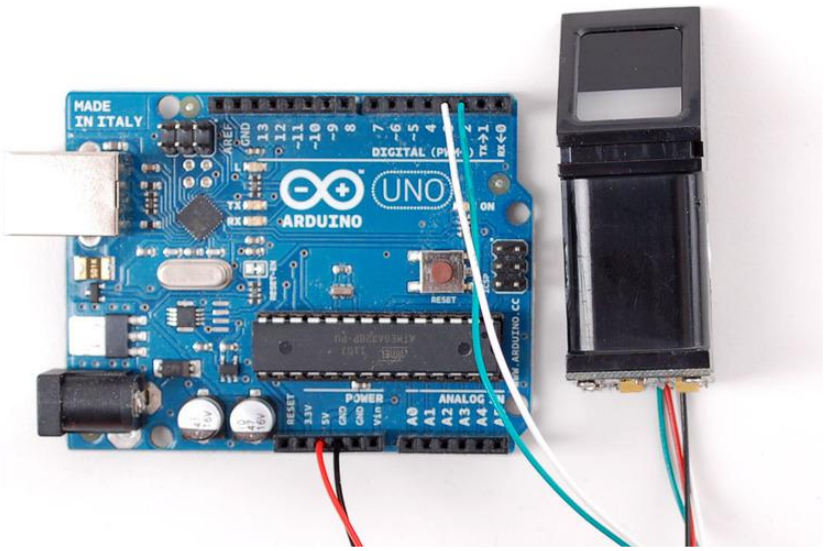
Black to GND
Yellow to Microcontroller TX (data out from microcontroller)
Green to Microcontroller RX (data in from microcontroller)
Red to 3.3V VIN

Arduino UNO & Compatible Wiring

This example sketch uses pins 2 and 3 for software serial (on ATmega328P type boards by default) - Not all boards support Software Serial on all pins so check board documentation! For example on ESP8266 we used 4 & 5

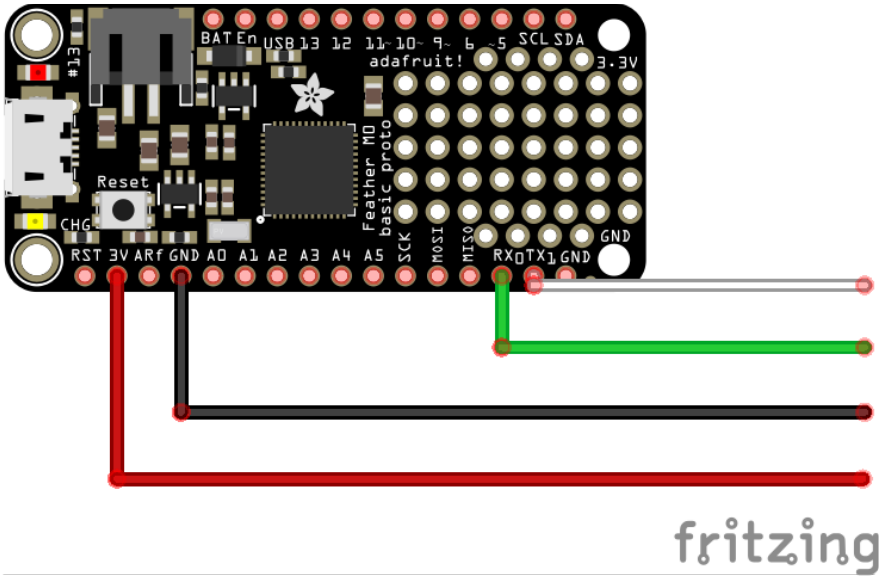
You can power the sensor from 3.3V or 5V - some sensors require 3.3V, see above!

In the diagrams below we show the wires plugged directly into the Arduino. However, this does not work well because the wires are so thin and they dont make contact. You should solder thicker solid core wires to each wire, to make good contact. NOTE: Both ends of the cable plug into the fingerprint sensor - verify that you're removing the correct end of it before cutting anything off. Your wiring order should match the image above with the fingerprint sensor upside down.



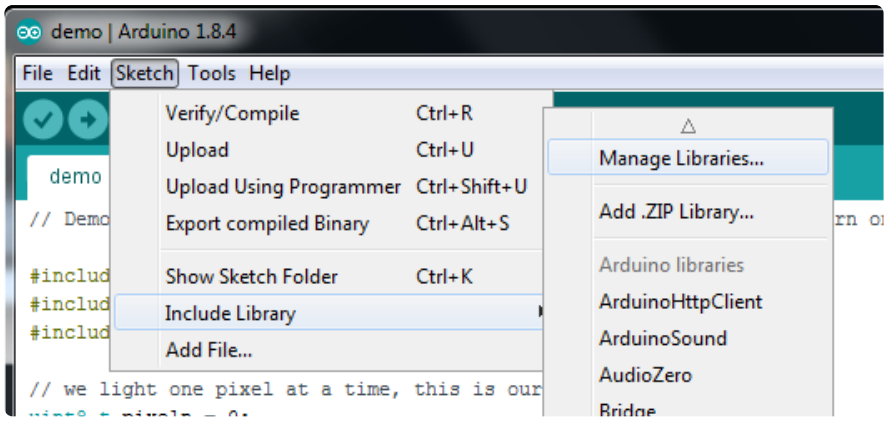
Hardware Serial Wiring

If you have a device with hardware serial, you should use that instead. Often this is pins #0 and #1

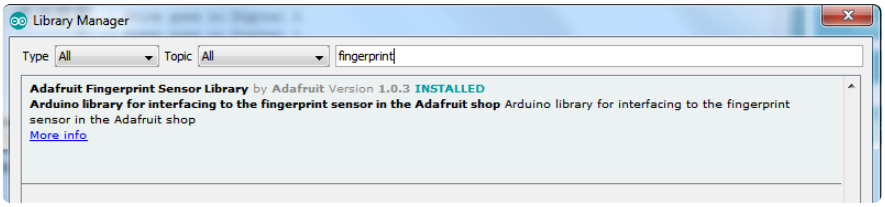


Next, you'll need to install the Adafruit Fingerprint sensor library (also available from github) ().

Open up the Arduino Library Manager:



Type in Fingerprint until you see the Adafruit Fingerprint library show up!



Click Install! That's it. Now you should be able to select the File→Examples→Adafruit_Fingerprint→fingerprint example sketch.

Soft & Hard Serial

By default the sketch uses software serial (Arduino UNO & compatibles). If you are using a device with Hardware Serial, e.g does not have a USB-Serial converter chip, use that instead! Usually those are on pins 0 & 1

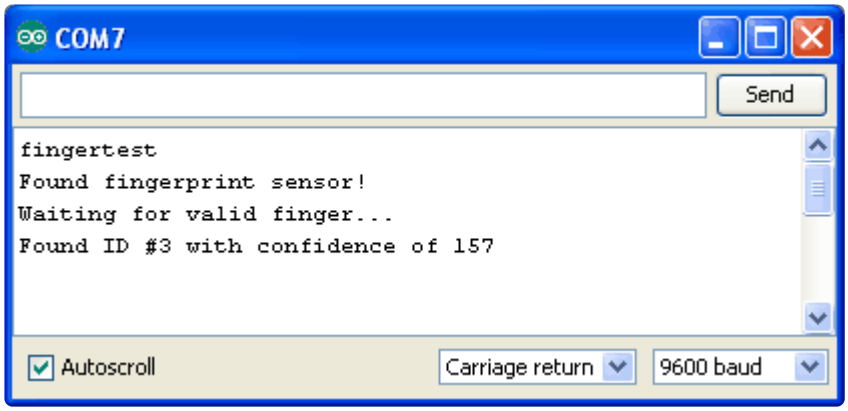
```
// On Leonardo/Micro or others with hardware serial, use those! #0 is green wire,
#1 is white
// uncomment this line:
#define mySerial Serial1

// For UNO and others without hardware serial, we must use software serial...
// pin #2 is IN from sensor (GREEN wire)
// pin #3 is OUT from arduino  (WHITE wire)
// comment these two lines if using hardware serial
//#include &lt;SoftwareSerial.h&gt;
//SoftwareSerial mySerial(2, 3);
```

If necessary, uncomment/comment lines for hardware serial support

Upload

Upload it to your Arduino as usual. Open up the serial monitor at 9600 baud and when prompted place your finger against the sensor that was already enrolled.

You should see the following:



If you can't get 'connected' to the sensor the library says "Did not find fingerprint sensor :(" Try swapping the RX and TX wires and resetting

The 'confidence' is a score number (from 0 to 255) that indicates how good of a match the print is, higher is better. Note that if it matches at all, that means the sensor is pretty confident so you don't have to pay attention to the confidence number unless it makes sense for high security applications.

Of course you have to have enrolled a fingerprint first! If you did this using the Windows program, that's good to go. If you have not yet, you should at least have gotten a `Found fingerprint sensor!` printout. You can go ahead to the next step to enroll fingerprints.

If you get `Did not find fingerprint sensor :(` check your wiring, maybe swap the RX and TX wire as that's a common issue

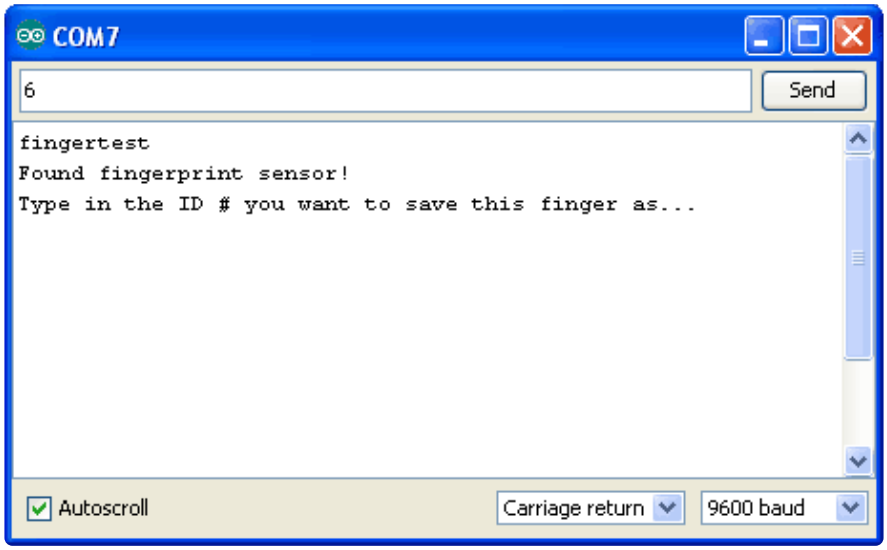
If you want to have a more detailed report, change the loop() to run getFingerprintID() instead of getFingerprintIDez() - that will give you a detailed report of exactly what the sensor is detecting at each point of the search process.

Enrolling with Arduino

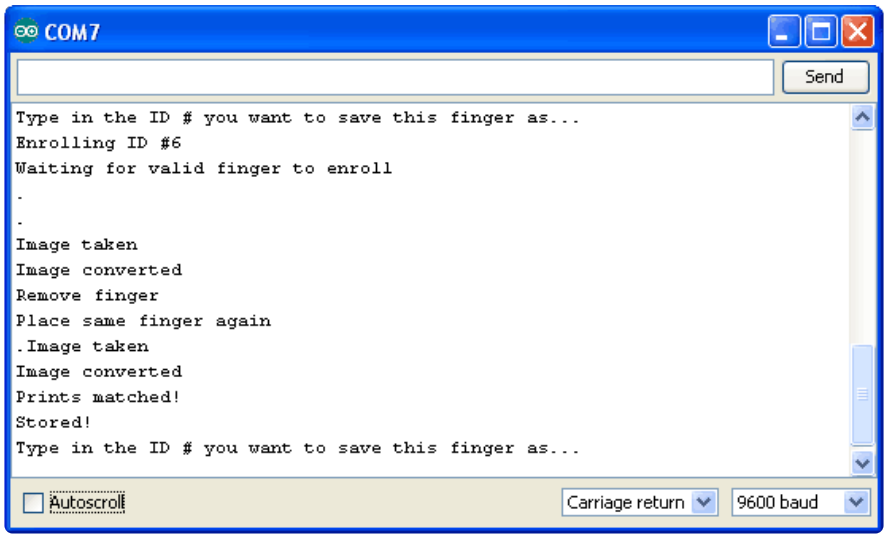
We did put together a simple sketch for enrolling a new finger via Arduino - its not as easy to use as the Windows program but it does work!

Run the File→Examples→Adafruit_Fingerprint→enroll sketch and upload it to the Arduino, use the same wiring as above.

When you open up the serial monitor, it will ask for you to type in the ID to enroll - use the box up top to type in a number and click Send.



Then go through the enrollment process as indicated. When it has successfully enrolled a finger, it will print Stored!



Don't forget to do a search test when you're done enrolling to make sure its all good!

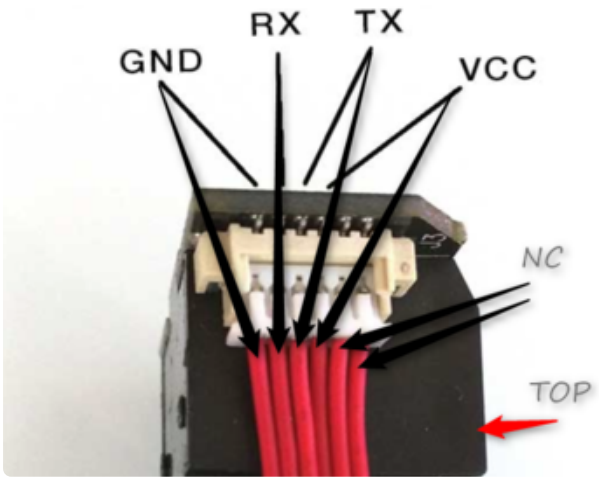
Python & CircuitPython

It's easy to use the optical fingerprint sensor with Python and CircuitPython, and the Adafruit CircuitPython Fingerprint () module. This module allows you to easily write Python code that reads, enrolls or deletes fingerprints.

You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library ().

Sensor Wiring

If your sensor has different wires, The first wire from the left should be the black wire ground, then the two data pins, RX is the white wire, TX is the green wire then the red power wire. You'll have to cut, strip and solder thicker gauge wires to the current wires.



If your sensor has all the same-color wires, The first wire from the left is ground, then the two data pins, then power. You'll have to cut, strip and solder the wires.

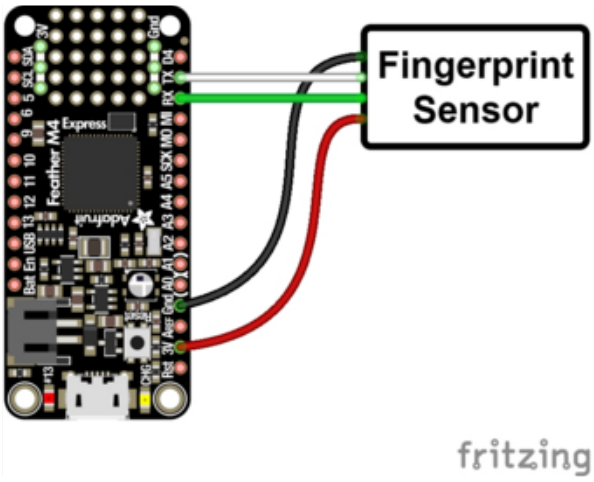
RX is the same as the White wire
TX is the same as the Green wire

In the diagrams below we show the wires plugged directly into the Trinket. However, this does not work well because the wires are so thin and they dont make contact. You should solder thicker solid core wires to each wire, to make good contact. NOTE: Both ends of the cable plug into the fingerprint sensor - verify that you're removing the correct end of it before cutting anything off. Your

wiring order should match the image above with the fingerprint sensor upside down.

CircuitPython Microcontroller Wiring

First wire up the fingerprint sensor to your board exactly as shown on the previous pages for Arduino. Here's an example of wiring a Feather M4 to the sensor with UART:



Board GND to sensor GND (black wire)
Board TX to sensor RX (white wire)
Board RX to sensor TX (green wire)
Board 3.3v to sensor VCC (red wire)

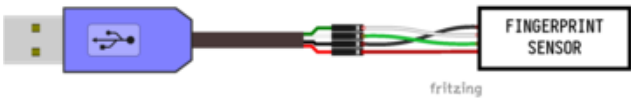
Every CircuitPython board has a hardware UART. Check the product page or look for RX and TX written on the board. Remember that the RX from the sensor goes to the TX on the board! If you have problems try swapping them, it's a common mistake.

Be aware that SAMD21 and other very-low-memory devices will not be able to run this example.

Python Computer Wiring

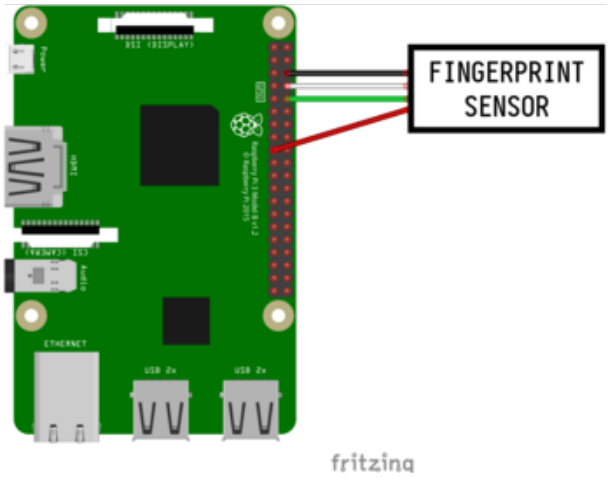
Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported ().

Here you have two options: An external USB-to-serial converter, or the built-in UART on the Pi's TX/RX pins. Here's an example of wiring up the USB-to-serial converter ():



Sensor VCC (red wire) to USB 5V or 3V (red wire on USB console cable)
Sensor GND (black wire) to USB Ground (black wire)
Sensor RX (white wire) to USB TX (green wire)
Sensor TX (green wire) to USB RX (white wire)

Here's an example using the Pi's built-in UART:



Pi GND to sensor GND (black wire)
Pi TX to sensor RX (white wire)
Pi RX to sensor TX (green wire)
Pi 3.3v to sensor VCC (red wire)

If you want to use the built-in UART, you'll need to disable the serial console and enable the serial port hardware in raspi-config. See the UART/Serial section of the CircuitPython on Raspberry Pi guide () for detailed instructions on how to do this.

Note SBC boards other than the Raspberry Pi may or may not have a hardware UART available on thee I/O pins or may have them permanently mapped to the console, etc. Refer to your board's documentation to see if there is a hardware UART, where it is located, how to enable it, and if it is available for programming with a serial device other than the console.

CircuitPython Fingerprint Library Installation

To use the Fingerprint sensor you'll need to install the Adafruit CircuitPython Fingerprint () library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython () for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (). Our introduction guide has a great page on how to install the library bundle () for both express and non-express boards.

Copy the necessary file from the library bundle to the lib folder on your CIRCUITPY drive:

- adafruit_fingerprint.mpy

Before continuing make sure your board's lib folder has the adafruit_fingerprint.mpy fi le copied over.

Next connect to the board's serial REPL ()so you are at the CircuitPython >>> prompt.

Python Installation of Fingerprint Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling the hardware UART on your platform (see red note above) and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready ()!

Once that's done, from your command line run the following command:

- `sudo pip3 install adafruit-circuitpython-fingerprint`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython & Python Usage

To demonstrate the usage of the sensor, we'll use the example python script included with the library. This sensor is fairly complex so its hard to run it just from the REPL.

CircuitPython Microcontroller Usage

Once you've installed the library, run this code.py example on your CircuitPython board.

Linux/Computer/Raspberry Pi with Python

If you're running fingerprint_simpletest.py on the Raspberry Pi (or any computer), you'll have to make some changes.

On the Raspberry Pi, comment out the `uart = busio.UART(...)` line, and uncomment the applicable `import serial` and `uart = serial.Serial(...)` lines, depending on whether you're using USB serial or hardware UART. Now you can run the program with the following command:

```
python3 fingerprint_simpletest.py
```

Example Code

Remember, SAMD21 and other very-low-memory devices will not be able to run the following example.

```
# 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)

##################################################
```

```
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="")
    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="")
    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="")
    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:
            print("Place finger on sensor...", end="")
        else:
```

```
            print("Place same finger again...", end="")

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

        print("Templating...", end="")
        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="")
    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="")
    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")
```

It's fairly long but it will help you set-up and test your sensor!

When you first start up, you should get something like this:

```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
main.py output:
----------------
Fingerprint templates: [2]
e) enroll print
f) find print
d) delete print
----------------
>
```

If you get an error like `RuntimeError: Failed to read data from sensor` it means something went wrong - check your wiring and baud rate!

This menu system is fairly simple, you have three things you can do

- Enroll print - you will use your finger to take images and 'store' the model in the sensor
- Find print - determine whether a fingerprint is known and stored
- Delete print - clear out a model

Enrolling Prints

Enrolling a finger print is easy. Type e to start the process. You'll need to select a location. The sensor can store up to 127 print locations. Pick a valid number, then place your finger twice to enroll.

```
Fingerprint templates: [2]
e) enroll print
f) find print
d) delete print
----------------
> e
Enter ID # from 1-127: 0
Enter ID # from 1-127: 199
Enter ID # from 1-127: 5
Place finger on sensor.................Image taken
Templating...Templated
Remove finger
Place same finger again....Image taken
Templating...Templated
Creating model...Created
Storing model #5...Stored
----------------
Fingerprint templates: [2, 5]
```

Note that after success, the Fingerprint templates: [...] printout will include the new template id.

If an error occurs, the sensor will give you an error, such as if the two prints don't match, or if it failed to store or generate a model:

```
> e
Enter ID # from 1-127: 4
Place finger on sensor..............Image taken
Templating...Templated
Remove finger
Place same finger again......Image taken
Templating...Templated
Creating model...Prints did not match
```

Finding Prints

Once you've enrolled fingerprints you can then test them. Run the find command, and try various fingers! Once the fingerprint id identified it will tell you the location number, in this case #5

```
----------------
> f
Waiting for image...
Templating...
Searching...
Finger not found
----------------
Fingerprint templates: [2, 5]
e) enroll print
f) find print
d) delete print
----------------
> f
Waiting for image...
Templating...
Searching...
Detected # 5 with confidence 102
```

Deleting Fingerprints

If you made a mistake you can remove fingerprint models from the database. For example, here's how to delete #5. Note the Fingerprint templates: [...] printout changes!

```
----------------
Fingerprint templates: [2, 5]
e) enroll print
f) find print
d) delete print
----------------
> d
Enter ID # from 1-127: 5
Deleted!
----------------
Fingerprint templates: [2]
e) enroll print
f) find print
d) delete print
----------------
```

Python Docs

Python Docs ()

Downloads

- Arduino interface library on github ()
- User Manual ()
- Datasheet (its not really a great datasheet and its in Chinese, but its better than nothing) ()

- English version of the User Manual ()
- "SFGDemo" Windows-only test software ()