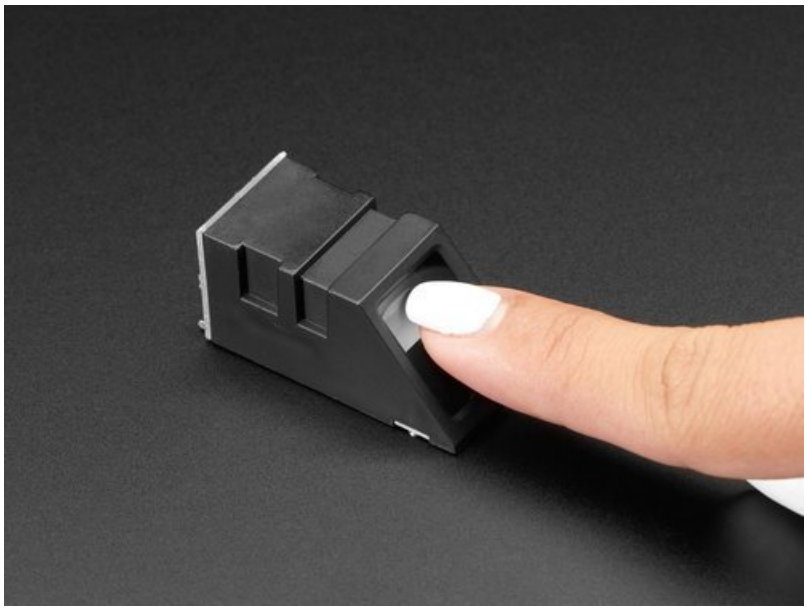
adafruit learning system

Adafruit Optical Fingerprint Sensor

Created by lady ada



Last updated on 2017-12-31 09:39:09 PM UTC

Guide 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. There's a red LED in the lens that lights up during a photo so you know its working.

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 *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() {}
```

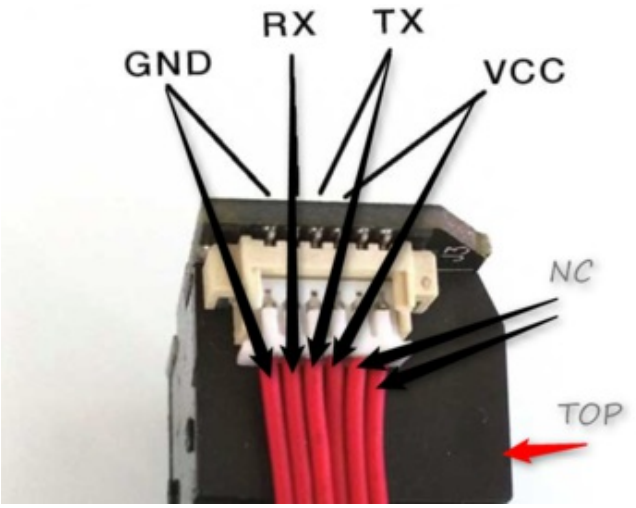
> The "blank" sketch won't work for "native USB" based Arduinos like the Leonardo, Micro, Zero, etc! Use the Leo_passthru sketch instead!

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());
}
```

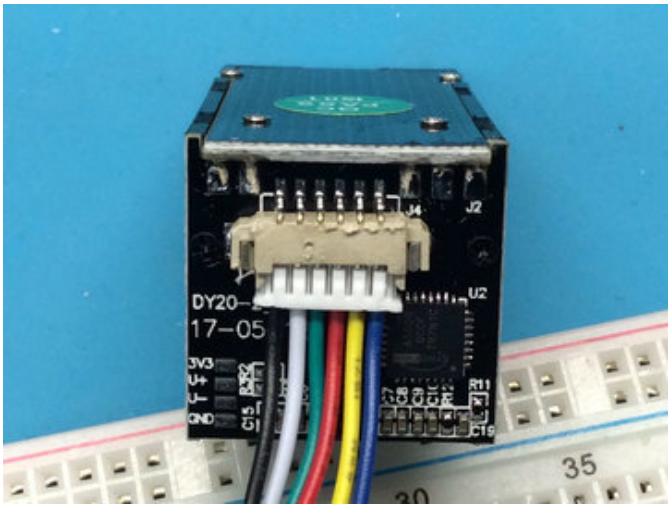
Wire up the sensor as described in the sketch comments **after** uploading the sketch. Since the sensor wires are so thin and short, we stripped the wire a bit and melted some solder on so it made better contact but you may want to solder the wires to header or similar if you're not getting good contact.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

LED blink to indicate the sensor is working.

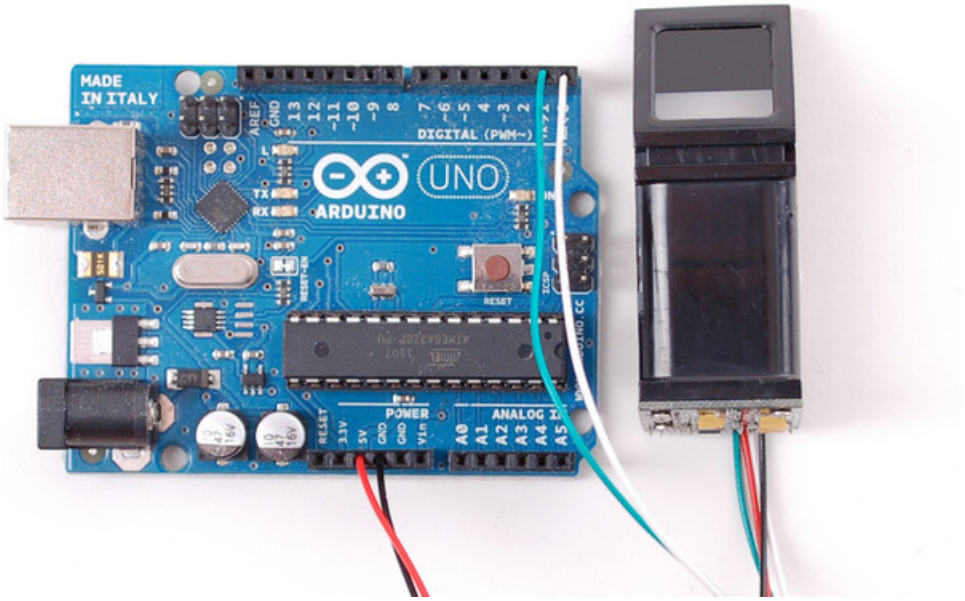


If your sensor has all the same-color wires, The first wire from the left is ground, then the two data pins, then power. You'll have to cut, strip and solder the wires.

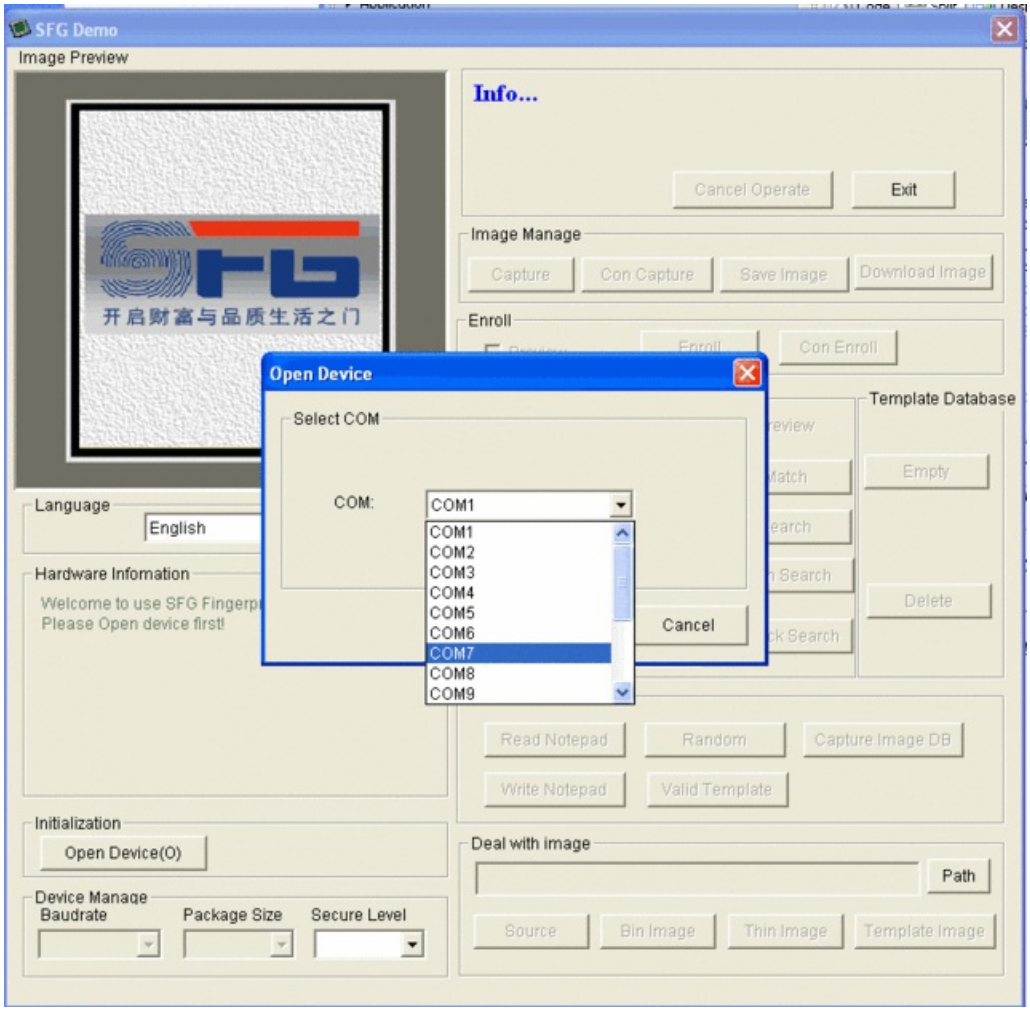
RX is the same as the White wire
TX is the same as the Green wire



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 the wires.

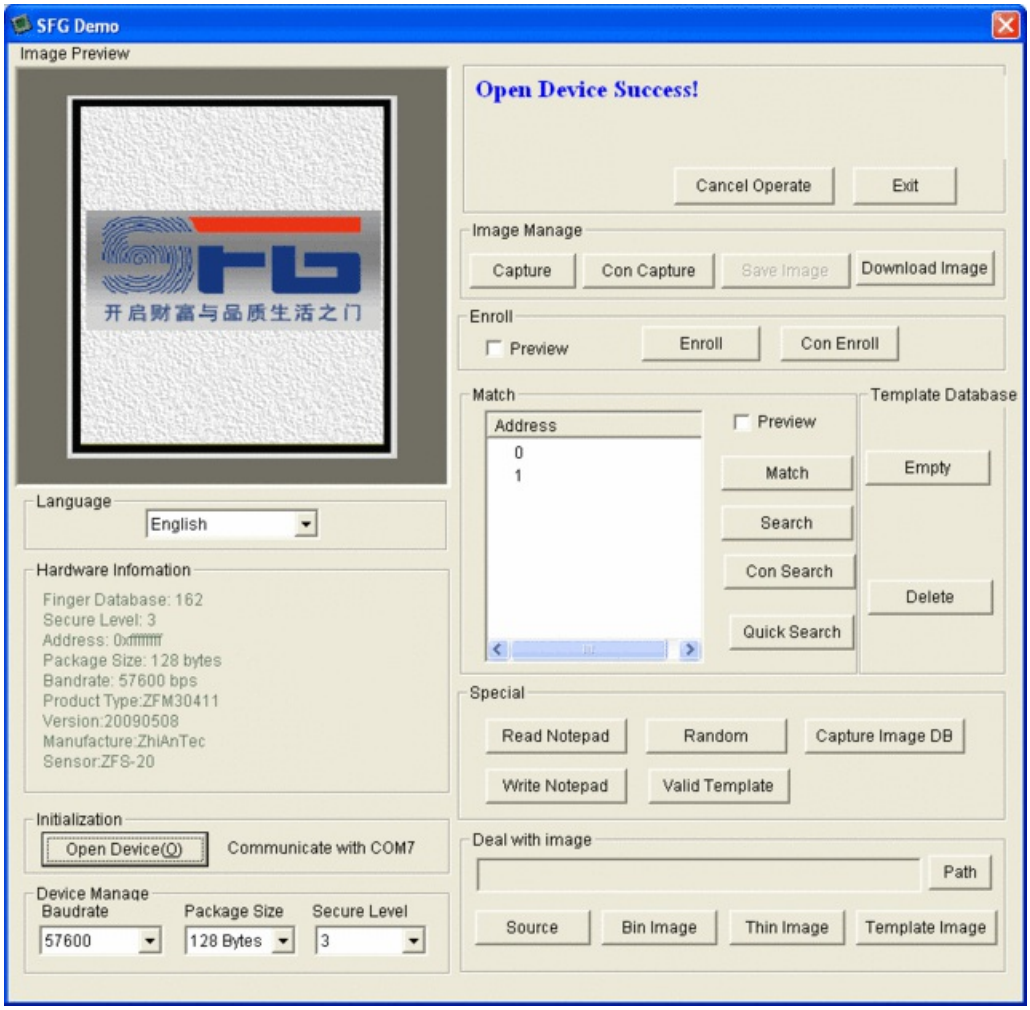


Start up the SFGDemo software and click **Open Device** from the bottom left corner. Select the **COM port** used by the Arduino.

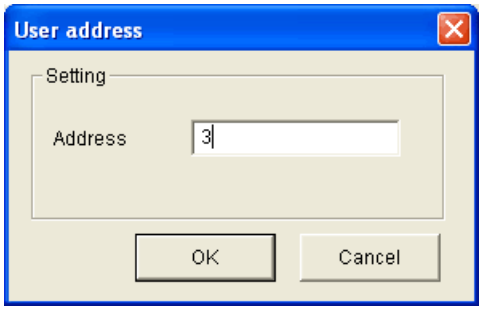


And press OK when done. You should see the following, with a blue success message and some device statistics in the bottom corner. You can change the baud rate in the bottom left hand corner, as well as the "security level" (how sensitive it is) but we suggest leaving those alone until you have everything 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.

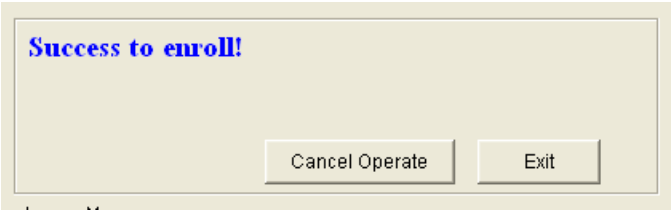


The software will ask you to press the finger to the sensor

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.



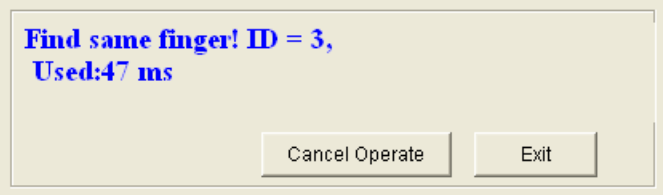
If there's a problem such as a bad print or image, you'll have to do it again.

Searching with the Software

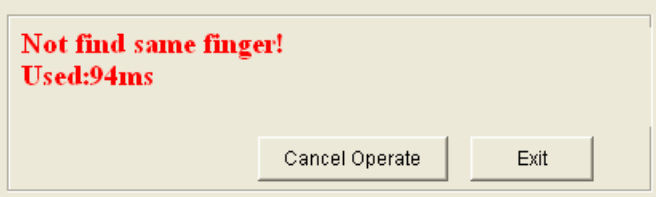
Once you have the finger enrolled, it's a good idea to do a quick test to make sure it can be found in the database. Click on the **Search** button on the right hand side.

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

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



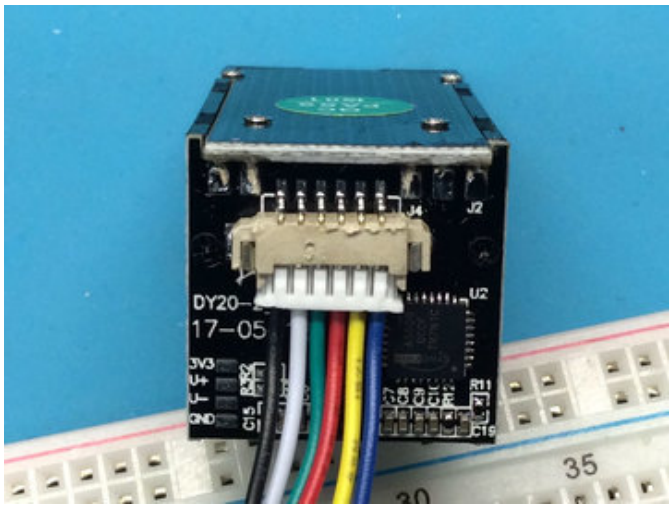
If it is not a finger in the database, you will get a failure notice.



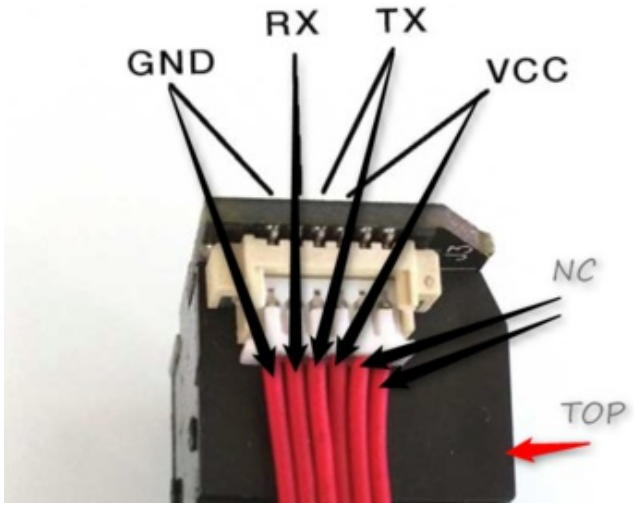
Wiring for use with Arduino

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 will be off until you've started to request data from it



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 the 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

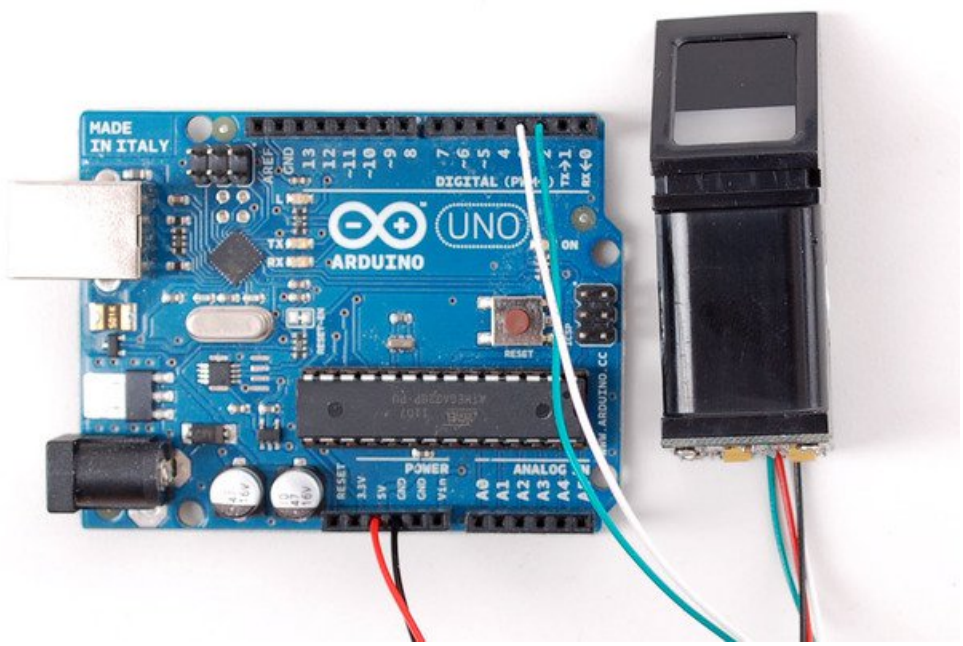
Arduino UNO & Compatible Wiring

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**

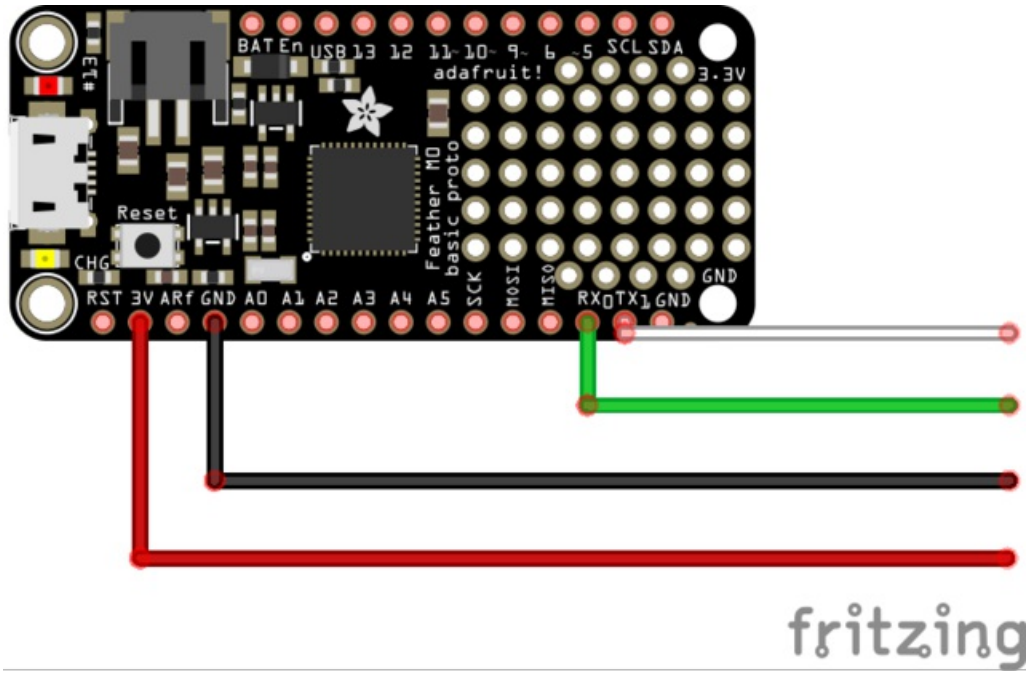
> In the diagrams below we show the wires plugged directly into the Arduino. However, this does not work well because the wires are so thin and they dont make contact. You should solder thicker solid core wires to each

wire, to make good contact



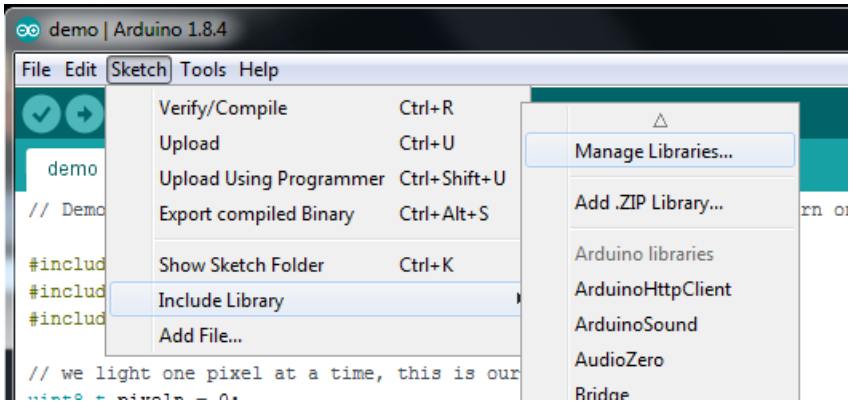
Hardware Serial Wiring

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

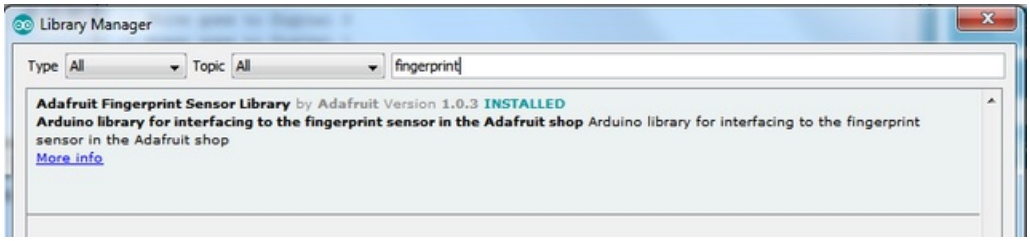


Next, you'll need to install the Adafruit Fingerprint sensor library (also available from github).

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

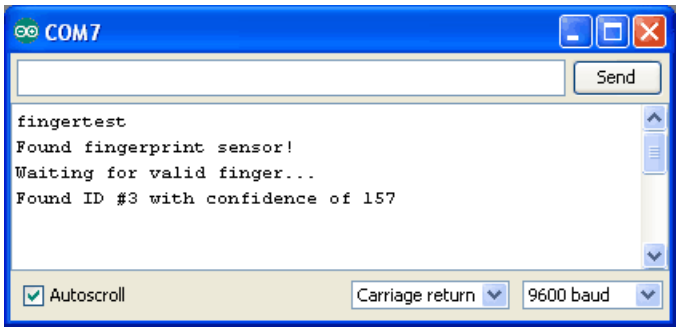
```
// On Leonardo/Micro or others with hardware serial, use those! #0 is green wire, #1 is white
// uncomment this line:
#define mySerial Serial1

// 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 <SoftwareSerial.h>
//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:



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

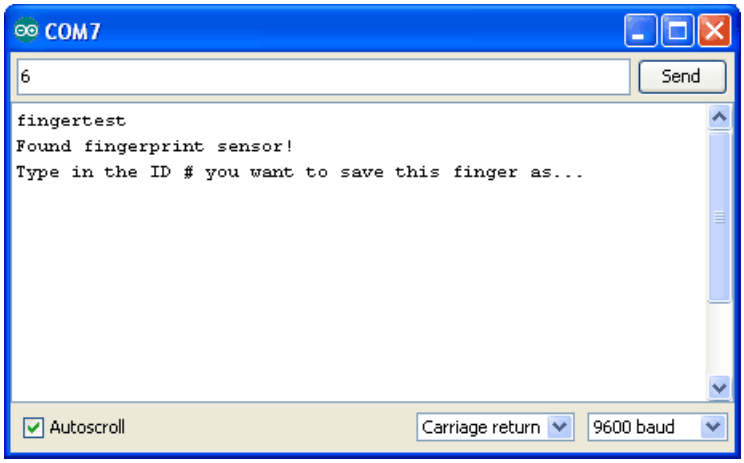
If you want to have a more detailed report, change the **loop()** to run **getFingerprintID()** instead of **getFingerprintIDez()** - that will give you a detailed report of exactly what the sensor is detecting at each point of the search process.

Enrolling with Arduino

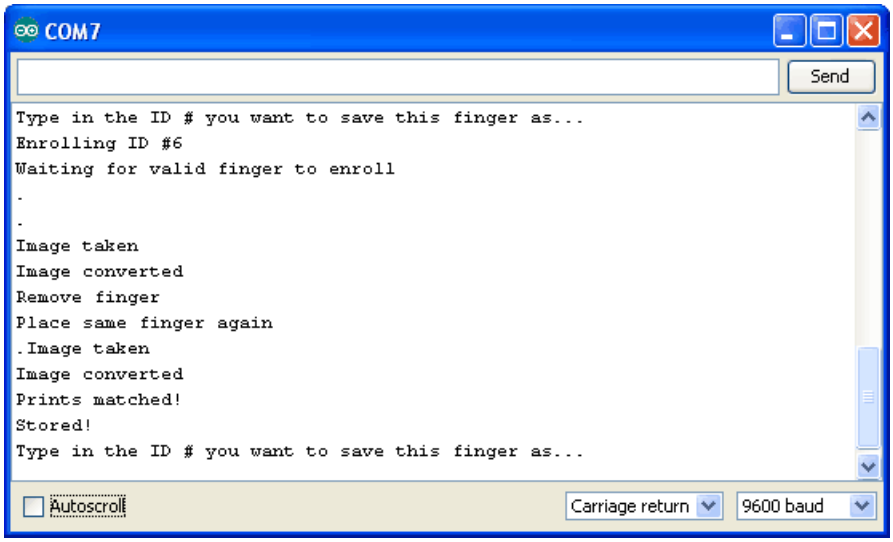
We did put together a simple sketch for enrolling a new finger via Arduino - its not as easy to use as the Windows program but it does work!

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.

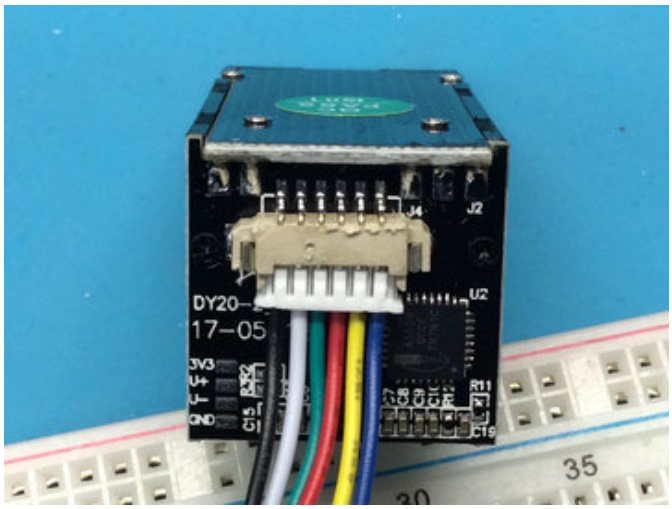


Then go through the enrollment process as indicated. When it has successfully enrolled a finger, it will print **Stored!**

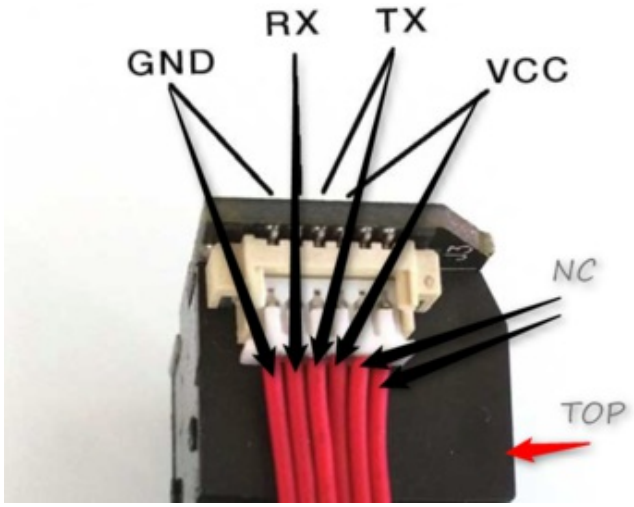


Don't forget to do a search test when you're done enrolling to make sure its all good!

CircuitPython

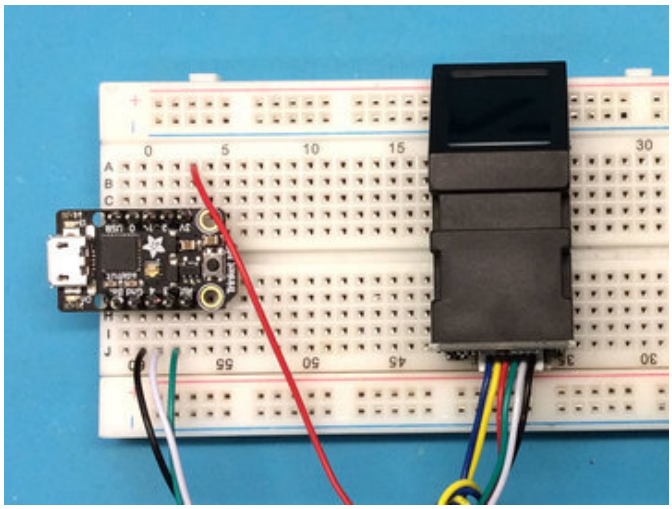


If your sensor has different wires, The first wire from the left should be the black wire ground, then the two data pins, RX is the white wire, TX is the green wire then the red power wire. You'll have to cut, strip and solder the wires.



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



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, its a common mistake

Installing Library

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.

Remember for non-express boards like the, you'll need to manually install the necessary libraries from the bundle:

- **adafruit_fingerprint.mpy**

You can also download the **adafruit_fingerprint.mpy** from its releases page on Github.

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_fingerprint.mpy** file copied over.

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

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.

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

```
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)

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_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="")
    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()
    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
            elif 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 **f**ind 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
----------------
```

Downloads

- Arduino interface library on github
- User Manual
- Datasheet (its not really a great datasheet and its Chinese but its better than nothing)
- English version of the User Manual
- "SFGDemo" Windows-only test software