CODE FOR ENROLLING FINGERPRINT

```
#include <Adafruit_Fingerprint.h>


#if (defined(__AVR__) || defined(ESP8266)) && !defined(__AVR_ATmega2560__)

SoftwareSerial mySerial(2, 3);

#else
// #0 is green wire, #1 is white
#define mySerial Serial1

#endif


Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

uint8_t id;

void setup()
{
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\nAdafruit Fingerprint sensor enrollment");

  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1) { delay(1); }
  }

  Serial.println(F("Reading sensor parameters"));
  finger.getParameters();
  Serial.print(F("Status: 0x")); Serial.println(finger.status_reg, HEX);
  Serial.print(F("Sys ID: 0x")); Serial.println(finger.system_id, HEX);
  Serial.print(F("Capacity: ")); Serial.println(finger.capacity);
  Serial.print(F("Security level: ")); Serial.println(finger.security_level);
  Serial.print(F("Device address: ")); Serial.println(finger.device_addr,
HEX);
  Serial.print(F("Packet len: ")); Serial.println(finger.packet_len);
  Serial.print(F("Baud rate: ")); Serial.println(finger.baud_rate);
```

```
}

uint8_t readnumber(void) {
  uint8_t num = 0;

  while (num == 0) {
    while (! Serial.available());
    num = Serial.parseInt();
  }
  return num;
}

void loop()                     // run over and over again
{
  Serial.println("Ready to enroll a fingerprint!");
  Serial.println("Please type in the ID # (from 1 to 127) you want to save
this finger as...");
  id = readnumber();
  if (id == 0) {// ID #0 not allowed, try again!
     return;
  }
  Serial.print("Enrolling ID #");
  Serial.println(id);

  while (!  getFingerprintEnroll() );
}

uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #"); Serial.println(id);
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      break;
    default:
      Serial.println("Unknown error");
```

```
    break;
  }
}

// OK success!

p = finger.image2Tz(1);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

Serial.println("Remove finger");
delay(2000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
  p = finger.getImage();
}
Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("Place same finger again");
while (p != FINGERPRINT_OK) {
  p = finger.getImage();
  switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image taken");
    break;
  case FINGERPRINT_NOFINGER:
    Serial.print(".");
    break;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
```

```
    break;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    break;
  default:
    Serial.println("Unknown error");
    break;
  }
}

// OK success!

p = finger.image2Tz(2);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK converted!
Serial.print("Creating model for #");  Serial.println(id);

p = finger.createModel();
if (p == FINGERPRINT_OK) {
  Serial.println("Prints matched!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
  Serial.println("Fingerprints did not match");
  return p;
} else {
  Serial.println("Unknown error");
```

```
    return p;
  }

  Serial.print("ID "); Serial.println(id);
  p = finger.storeModel(id);
  if (p == FINGERPRINT_OK) {
    Serial.println("Stored!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not store in that location");
    return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  return true;
}
```

FOR UNLOCKING DOORLOCK

```
#include <Adafruit_Fingerprint.h>

SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

#define RELAY_PIN       4
#define ACCESS_DELAY    3000 // Keep lock unlocked for 3 seconds
const int buzzer= 11;

void setup()
{
  // set the data rate for the sensor serial port
  finger.begin(57600);
  delay(5);
  if (finger.verifyPassword())
  {
  }
  else
  {
    while (1) { delay(1); }
```

```
  }

  pinMode(buzzer , OUTPUT);

  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, HIGH);   //Switch off relay initially. Relay is LOW
level triggered relay so we need to write HIGH.
}

void loop()
{
  if ( getFingerPrint() != -1)
  {

    digitalWrite(RELAY_PIN, LOW);
    delay(ACCESS_DELAY);
    digitalWrite(RELAY_PIN, HIGH);

  }
  else{

  }
  delay(50);            //Add some delay before next scan.
}

// returns -1 if failed, otherwise returns ID #
int getFingerPrint()
{
  int p = finger.getImage();
  if (p != FINGERPRINT_OK){

  return -1;
}

  p = finger.image2Tz();
  if (p != FINGERPRINT_OK)
  {
  return -1;
}

  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)
  {
  digitalWrite(buzzer,HIGH);
  delay(100);
  digitalWrite(buzzer,LOW);
  delay(100);
  digitalWrite(buzzer,HIGH);
```

```
delay(100);
digitalWrite(buzzer,LOW);
delay(100);
digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
delay(100);
digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
delay(100);
digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
delay(100);
digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
delay(100);
digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
delay(100);
digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
delay(100);
digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
delay(100);
digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
delay(100);
digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
delay(100);
digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);
delay(100);
digitalWrite(buzzer,HIGH);
delay(100);
digitalWrite(buzzer,LOW);

return -1;
```

```
}

  // found a match!
  digitalWrite(buzzer,HIGH);
  delay(500);
  digitalWrite(buzzer,LOW);
  return finger.fingerID;
}
```

FOR DELETING A FINGERPRINT

```
#include <Adafruit_Fingerprint.h>


#if (defined(__AVR__) || defined(ESP8266)) && !defined(__AVR_ATmega2560__)

SoftwareSerial mySerial(2, 3);

#else

#define mySerial Serial1

#endif


Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

void setup()
{
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\nDelete Finger");

  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1);
  }
}


uint8_t readnumber(void) {
  uint8_t num = 0;
```

```
  while (num == 0) {
    while (! Serial.available());
    num = Serial.parseInt();
  }
  return num;
}

void loop()                     // run over and over again
{
  Serial.println("Please type in the ID # (from 1 to 127) you want to
delete...");
  uint8_t id = readnumber();
  if (id == 0) {// ID #0 not allowed, try again!
     return;
  }

  Serial.print("Deleting ID #");
  Serial.println(id);

  deleteFingerprint(id);
}

uint8_t deleteFingerprint(uint8_t id) {
  uint8_t p = -1;

  p = finger.deleteModel(id);

  if (p == FINGERPRINT_OK) {
    Serial.println("Deleted!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not delete in that location");
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
  } else {
    Serial.print("Unknown error: 0x"); Serial.println(p, HEX);
  }

  return p;
}
```

s