Fingerprint Based Student Attendance System Using GSM Module

Abstract: The system includes terminal fingerprint acquisition module and attendance module. It can realize automatically such functions as information acquisition of fingerprint, processing, and wireless transmission, fingerprint matching and making an attendance report. After taking the attendance, this system sends the attendance of every student to their parent's mobile through GSM and also stored the attendance of respective student to calculate the percentage of attendance and alerts to class in charge. Attendance system facilitates access to the attendance of a particular student in a particular class. This system eliminates the need for stationary materials and personnel for the keeping of records and efforts of class in charge.

Equipment:

- Optical Fingerprint Sensor Module-FPM10A
- GSM SIM900A
- Arduino Uno
- Center Tapped Transformer(9V/1000mA)
- Buck converter
- Capacitor(25v/1000µf)
- Diodes
- Connectors

Designing System Hardware:

Optical Fingerprint Sensor Module-FPM10A: For securing 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 .We can also enroll new fingers directly up to 162 finger prints can be stored in the on board FLASH memory. There's a red LED in the lens that lights up during a photo so we know its working .We like this particular sensor because not only is easy to use, it also comes with fairly straight-forward Windows software that makes testing the module simple .We can even enroll using the software and see an image of the fingerprint on our computer screen.

Features:

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

Source code:

Fingerprint sensor: Firstly we have to upload this code on arduino uno .The code is given below:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
int fingerprintID = 0;
String IDname;
int led1=6;
int led2=7;
int led3=8;


void setup(){
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
  Serial.begin(9600);
  finger.begin(57600);
  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  }
  else {
    Serial.println("Did not find fingerprint sensor :(");
```

```
    while (1) { delay(1); }
  }
  Wire.begin();
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  displayMainScreen();
}

void loop(){
  displayMainScreen();
  fingerprintID = getFingerprintIDez();
  delay(50);
}
int getFingerprintIDez() {
  uint8_t 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)  return -1;
  Serial.print("Found ID #");
  Serial.print(finger.fingerID);
  Serial.print("\n");
  return finger.fingerID;
}

void displayMainScreen(){

  delay(2000);
  digitalWrite(led1,LOW);
  digitalWrite(led2,LOW);
  digitalWrite(led3,LOW);

}

void displayUser(String Name){
 delay(1000);
  fingerprintID = 0;
  digitalWrite(led1,HIGH);
}

void displayUser1(String Name){

  delay(1000);
  fingerprintID = 0;
  digitalWrite(led2,HIGH);
}
void displayUser2(String Name){
```

```
  delay(1000);
  fingerprintID = 0;
  digitalWrite(led3,HIGH);
}
```

Enrolling: First is we'll need to enroll fingerprints that means assigning ID #'s to each print so we can query them later. Once we've enrolled all our prints, we can easily 'search' the sensor, asking it to identify which ID (if any) is currently being photographed .The code is:

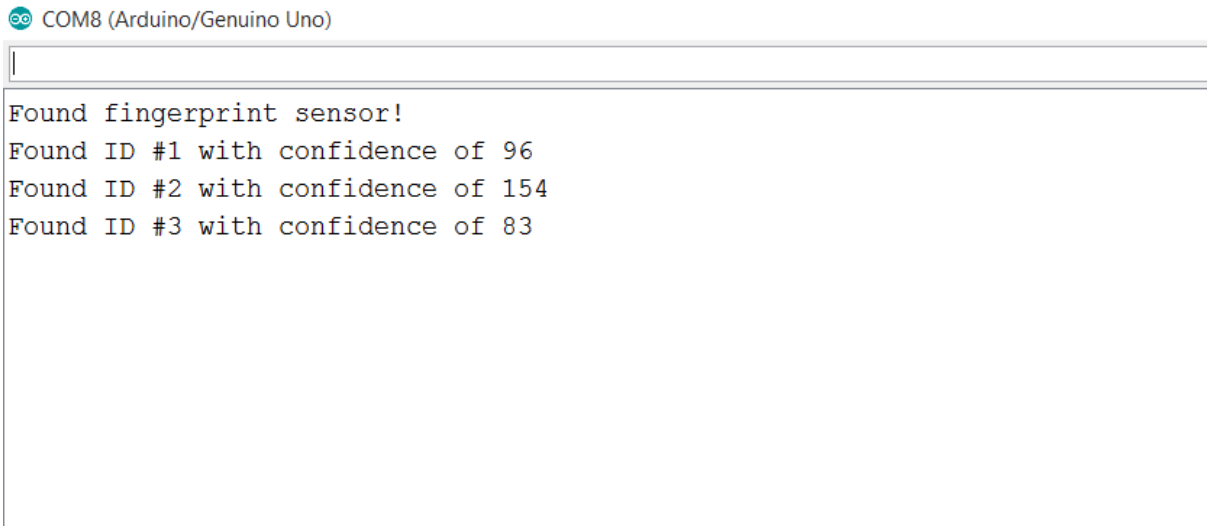
```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>

uint8_t getFingerprintEnroll(int id);


// pin #2 is IN from sensor (GREEN wire)
// pin #3 is OUT from arduino  (WHITE wire)
SoftwareSerial mySerial(2, 3);

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

void setup()
{
  Serial.begin(9600);
  Serial.println("fingertest");

  finger.begin(57600);

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

void loop()
{
  Serial.println("Type in the ID # you want to save this finger as...");
  int id = 0;
  while (true) {
    while (! Serial.available());
    char c = Serial.read();
    if (! isdigit(c)) break;
    id *= 10;
    id += c - '0';
  }
  Serial.print("Enrolling ID #");
```

```
  Serial.println(id);

  while (!  getFingerprintEnroll(id) );
}

uint8_t getFingerprintEnroll(int id) {
  int p = -1;
  Serial.println("Waiting for valid finger to enroll");
  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;
    }
  }


  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();
}

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;
  }
}

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;
  }

  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;
  }
}
```

Output:



GSM SIM900A MODULE: GSM Module-RS232 is built with Dual Band GSM/GPRS engine- SIM900A, works on frequencies 900/ 1800 MHz. The Module is coming with RS232 interface, which allows us connect PC as well as microcontroller with RS232 Chip(MAX232). The baud rate is configurable from 9600-115200 through AT command. The GSM/GPRS Module is having internal TCP/IP stack to enable us to connect with internet via GPRS. It is suitable for SMS, Voice as well as DATA transfer application in M2M interface. The on board Regulated Power supply allows us to connect wide range unregulated power supply . Using this modem, we can make audio calls, SMS, Read SMS, attend the incoming calls and internet through simple AT commands.

Features:

- Dual-Band GSM/GPRS 900/ 1800 MHz.
- RS232 interface for direct communication with computer or MCU kit.
- Configurable baud rate.
- Power controlled using 29302WU IC.
- ESD Compliance.
- Enable with MIC and SPeaker socket.
- With slid in SIM card tray.
- With Stub antenna and SMA connector.
- Input Voltage: 12V DC.

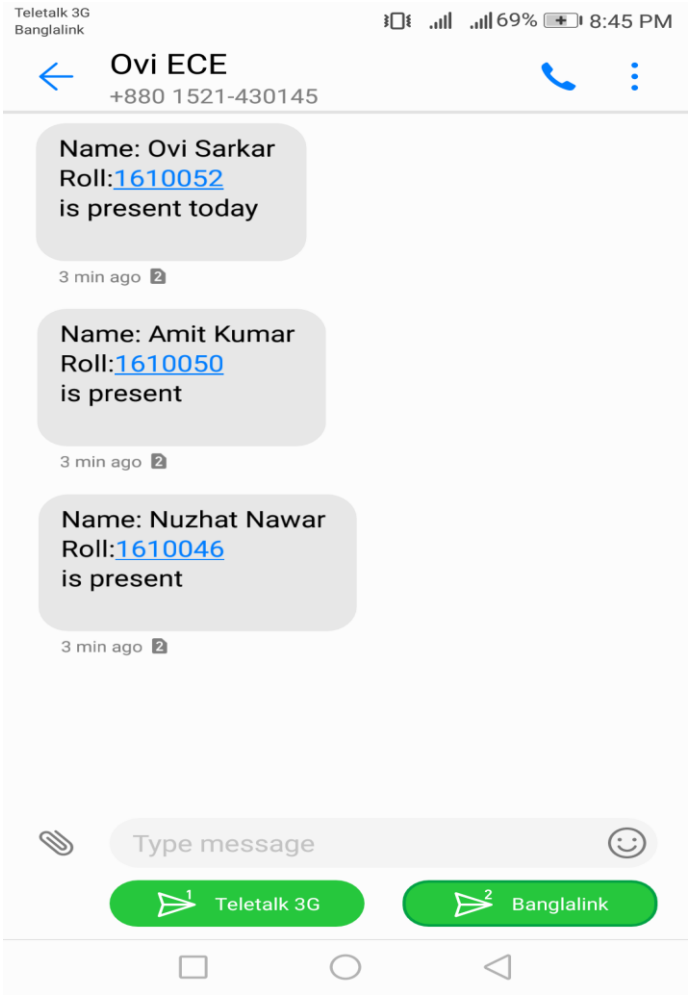
Source Code:

```
#include <SoftwareSerial.h>
SoftwareSerial GPRS(2,3);
boolean state, lastState;
boolean state1, lastState1;
boolean state2, lastState2;
```

```
void setup()
{
  pinMode(6, INPUT_PULLUP);
     pinMode(7, INPUT_PULLUP);
        pinMode(8, INPUT_PULLUP);
  state = digitalRead(6);
  lastState = state;
      state1 = digitalRead(7);
  lastState1 = state1;
            state2 = digitalRead(8);
  lastState2 = state2;

  GPRS.begin(9600);
  Serial.begin(9600);

  GPRS.println("AT+CMGF=1");

  delay(1000);
}

void loop()
{
  while(GPRS.available()) {
    Serial.write(GPRS.read());
  }

  lastState = state;
     state = digitalRead(6);
        lastState1 = state1;
              state1 = digitalRead(7);
               lastState2 = state2;
                    state2 = digitalRead(8);

  if ( state != lastState ) {
    sendSMS();
  }

      if ( state1 != lastState1 ) {
    sendSMS1();
  }
          if ( state2 != lastState2 ) {
            sendSMS2();
          }


  delay(500);
}
```

```
void sendSMS() {
  GPRS.println("AT+CMGS=\"+8801922029625\"\r");
  delay(1000);
  GPRS.println("Name: Ovi Sarkar\nRoll:1610052\nis present today");
  delay(100);
   GPRS.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
  //GPRS.write( 0x1a ); // ctrl+Z character

  delay(500);
}

void sendSMS1() {
  GPRS.println("AT+CMGS=\"+8801922029625\"\r");
 delay(1000);
  GPRS.println("Name: Nuzhat Nawar\nRoll:1610046\nis present today");
  delay(100);
   GPRS.println((char)26);// ASCII code of CTRL+Z
  delay(100);
  //GPRS.write( 0x1a ); // ctrl+Z character

  delay(500);
}

void sendSMS2() {
  GPRS.println("AT+CMGS=\"+8801922029625\"\r");
  delay(1000);
  GPRS.println("Name: Amit Kumar Paul\nRoll:1610050\nis present today");
  delay(100);
   GPRS.println((char)26);
  delay(100);
  delay(500);
}
```

Output:



Discussions: The main purpose of this project is to monitor the student attendance in lecture, tutorial and laboratory sessions in more efficient way and send this attendance to their parents. This system resists students from bunking classes through SMS sending feature to parents. Biometrics has been used effectively for more than a decade for time and attendance system. Fingerprint attendance system is a cost effective simplified system that uses fingerprints for identification.