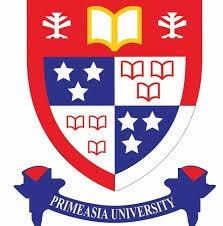
**PRIMEASIA UNIVERSITY**

**DEPARTMENT OF CSE**



**PROJECT REPORT**

**Computer Interfacing (LAB)**

**(CSE-402)**

**SUBMITTED BY**

**TANVIR ANJUM LABIR**

**191039042**

**MANZURUL ISLAM**

**191005042**

**SANTOSH MRIDHA**

**181048042**

**Atia Sikder**

**172024042**

**SUBMITTED TO**

**SALEHUZZAMAN EBON**

**LECTURER**

**DEPARTMENT OF CSE**

**DATE: December 21, 2021**

**INTRODUCTION**

**Biometric systems have overtime served as robust security mechanisms in various domains. Fingerprints are the oldest and most widely used form of biometric identification. The use of fingerprint for identification has been employed in law enforcement for about a century. A much broader application of fingerprint is for personal authentication, for instance to access a computer, a network, an ATM machine, a car or a home. Electronic lock using fingerprint recognition system is a process of verifying the fingerprint image to open the electronic lock. This project highlights the development of fingerprint verification. Verification is completed by comparing the data of authorized fingerprint image with incoming fingerprint image. Then the information of incoming fingerprint image will undergo the comparison process to compare with authorized fingerprint image. Fingerprint door lock incorporates the proven technology. Fingerprint reader scanning is the most mature and tested type of biometric technology. Recent studies on biometrics have shown that compared to the hand method, fingerprint is more accurate and cost-effective. The duplication of biometric fingerprint technology is virtually impossible, only one in one billionth of a chance. Biometric security guarantees a positive method of user identification with something that cannot be lost, replicated or stolen.**

**ARDUINO:**

**Arduino is an open-source electronic platform based on easy-to-use hardware and software. Arduino boards are able to read inputs – light on sensor, a finger on a button, or a Twitter message – and turn it into an output – activating a motor, turning on an LED, publishing some thing online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on writing), and the arduino software (IDE), based on processing. Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers students, hobbyists, artists, programmers, and professionals – has gathered around this open source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike. Arduino was born at the ivrea interaction design institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IOT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open –source, and it is growing through the contributions of users worldwide.**

****

**Arduino UNO R3**

**The power pins description +VIN:- The input voltage to the Arduino board when it’s using an external power source (as opposed to volt from the USB connector or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access through this pin. +5V:- The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN an on-board regulator, or be supplied but USB or another 5V supply. +3V3:- A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50mA. GND:- Ground pins.**

**FINGERPRINT SCANNER**

**A fingerprint is an impression left by the friction ridges of a human finger. The recovery of partial fingerprints from a crime scene is an important method of forensic science. Moisture and grease on a finger result in fingerprints on surfaces such as glass or metal. Deliberate impressions of entire fingerprints can be obtained by ink or other substances transferred from the peaks of friction ridges on the skin to a smooth surface such as paper. Fingerprint records normally contain impressions from the pad on the last joint of fingers and thumbs, although fingerprint cards also typically record portions of lower joint areas of the fingers. Human fingerprints are detailed, nearly unique, difficult to alter, and durable over the life of an individual, making them suitable as long-term markers of human identity. They may be employed by police or other authorities to identify individuals who wish to conceal their identity, or to identify people who are incapacitated or deceased and thus unable to identify themselves, as in the aftermath of a natural disaster.**

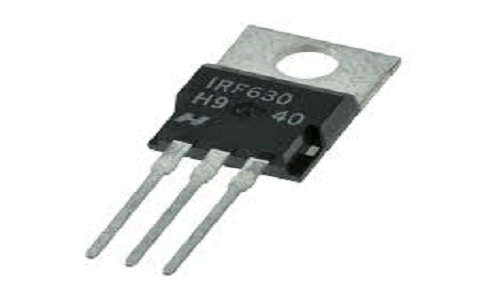
**Here in our project we use 4 pins of this scanner. Red which is Vcc connected to Arduino 5v, Black which is negative connected to GND of Arduino, Tx is connected to pin 2 and Rx is connected to pin 3 .**

****

**Fingerprint Sensor**

**MOSFET Transistor:**

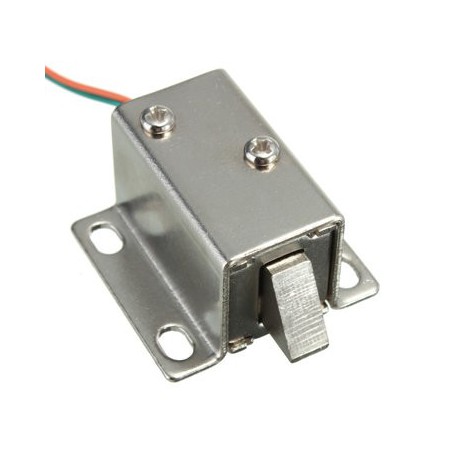
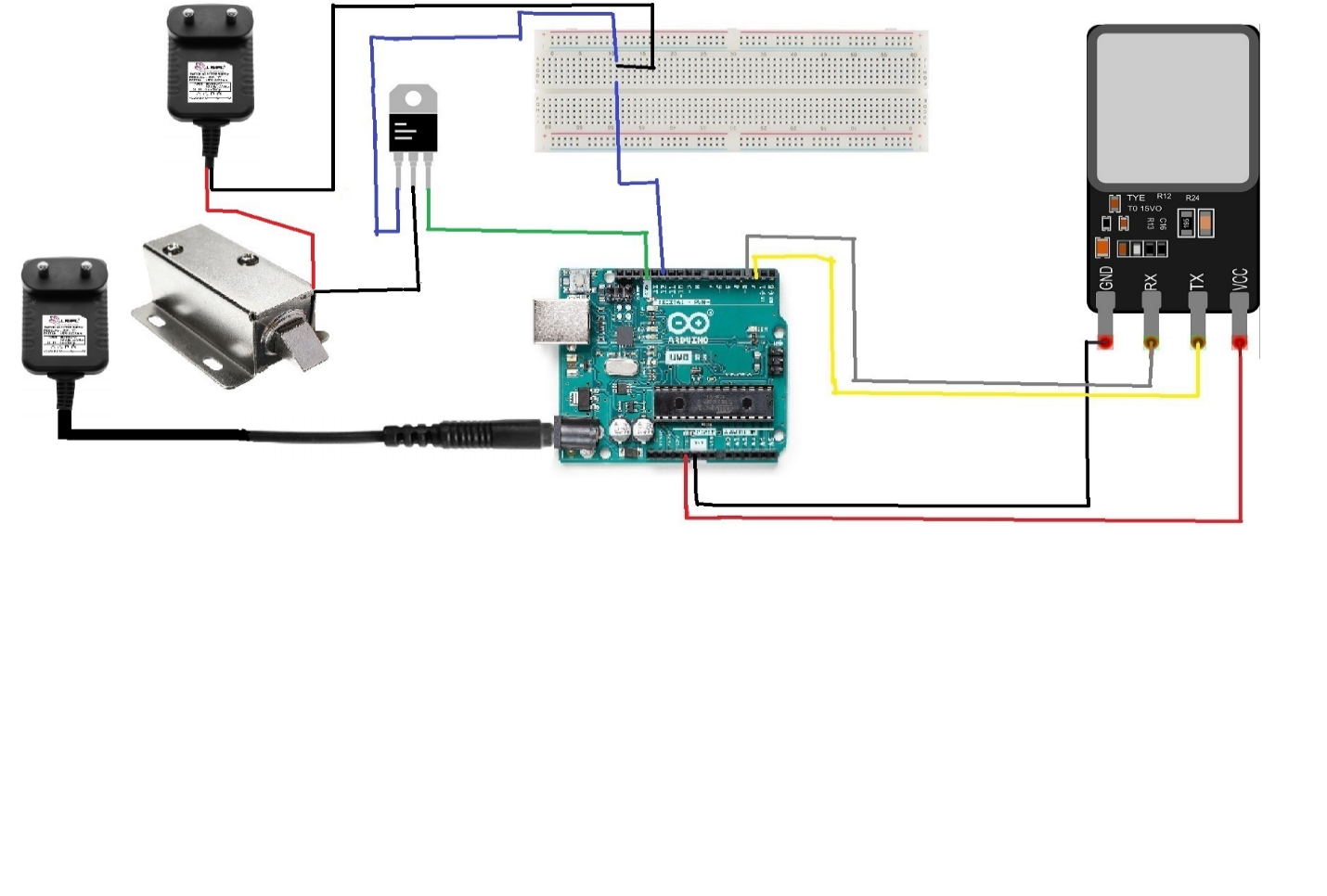
**MOSFET, in short, is a metal oxide semiconductor field-effect transistor used to switch or amplify voltages in circuits. Being part of the field-effect transistor family, it is a current-controlled device that is constructed with 3 terminals; Source Gain Drain The purpose of a MOSFET transistor is essentially to control voltage/current flow between the source and the drain. The working principle differs based on the type of MOSFET. Here in our project MOSFET is used as a switch. We connected doorlock negative wire to the emitter of the MOSFET. And base to ground and collector to pin no 12 of Arduino uno. While fingerprint matches Arduino make the pin 12 HIGH and the depletion region of MOSFET gets short to base and collector. As we know our base is ground , emitter is negative which needs to be grounded to open the lock so the lock opens . And when fingerprint doesnot matches MOSFET doesnot flow the free electrons and the lock remains locked.**

****

**Mosfet Transistor**

**SOLENOID LOCK:**

**The solenoid lock denotes a latch for electrical locking and unlocking. It is available in unlocking in the power-on mode type, and locking and keeping in the power-on mode type, which can be used selectively for situations. The power-on unlocking type enables unlocking only while the solenoid is powered on.**

**Circuit Diagram:- **

**SOFTWEAR DESCRIPTION ARDUINO IDE (R3):**

**The open-source Arduino software (IDE) makes is easy to write code and upload it to the board. It runs on windows, MAX OS X, and Linux. The environment is written in java and based on processing and other open-source software. The Arduino integrate Development Environment or Arduino software (IDE) contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common function and a series of menus. It connects to the Arduino and Genuine hardware to upload the program and communication with them.**

****

**Source Code:-**

**Enroll**

|  |
| --- |
| #include <Adafruit\_Fingerprint.h> |
|  |  |
|  |  |
|  | #if (defined(\_\_AVR\_\_) || defined(ESP8266)) && !defined(\_\_AVR\_ATmega2560\_\_) |
|  | // 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) |
|  | // Set up the serial port to use softwareserial.. |
|  | SoftwareSerial mySerial(2, 3); |
|  |  |
|  | #else |
|  | // On Leonardo/M0/etc, others with hardware serial, use hardware serial! |
|  | // #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; |
|  | } |

**Fingerprint**

|  |
| --- |
| #include <Adafruit\_Fingerprint.h> |
|  |  |
|  |  |
|  | #if (defined(\_\_AVR\_\_) || defined(ESP8266)) && !defined(\_\_AVR\_ATmega2560\_\_) |
|  | // 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) |
|  | // Set up the serial port to use softwareserial.. |
|  | SoftwareSerial mySerial(2, 3); |
|  |  |
|  | #else |
|  | // On Leonardo/M0/etc, others with hardware serial, use hardware serial! |
|  | // #0 is green wire, #1 is white |
|  | #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\nAdafruit finger detect test"); |
|  |  |
|  | // set the data rate for the sensor serial port |
|  | finger.begin(57600); |
|  | delay(5); |
|  | 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); |
|  |  |
|  | finger.getTemplateCount(); |
|  |  |
|  | if (finger.templateCount == 0) { |
|  | Serial.print("Sensor doesn't contain any fingerprint data. Please run the 'enroll' example."); |
|  | } |
|  | else { |
|  | Serial.println ("Waiting for valid finger..."); |
|  | Serial.print("Sensor contains "); Serial.print(finger.templateCount); Serial.println(" templates"); |
|  | } |
|  | } |
|  |  |
|  | void loop() // run over and over again |
|  | { |
|  | getFingerprintID(); |
|  | Delay (50); //don't ned to run this at full speed. |
|  | } |
|  |  |
|  | uint8\_t getFingerprintID() { |
|  | uint8\_t p = finger.getImage(); |
|  | switch (p) { |
|  | case FINGERPRINT\_OK: |
|  | Serial.println("Image taken"); |
|  | break; |
|  | case FINGERPRINT\_NOFINGER: |
|  | Serial.println("No finger detected"); |
|  | return p; |
|  | case FINGERPRINT\_PACKETRECIEVEERR: |
|  | Serial.println("Communication error"); |
|  | return p; |
|  | case FINGERPRINT\_IMAGEFAIL: |
|  | Serial.println("Imaging error"); |
|  | return p; |
|  | default: |
|  | Serial.println("Unknown error"); |
|  | return p; |
|  | } |
|  |  |
|  | // OK success! |
|  |  |
|  | p = finger.image2Tz(); |
|  | 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! |
|  | p = finger.fingerSearch(); |
|  | if (p == FINGERPRINT\_OK) { |
|  | Serial.println ("Found a print match!"); |
|  | } else if (p == FINGERPRINT\_PACKETRECIEVEERR) { |
|  | Serial.println("Communication error"); |
|  | return p; |
|  | } else if (p == FINGERPRINT\_NOTFOUND) { |
|  | Serial.println("Did not find a match"); |
|  | return p; |
|  | } else { |
|  | Serial.println("Unknown error"); |
|  | return p; |
|  | } |
|  |  |
|  | // found a match! |
|  | Serial.print("Found ID #"); Serial.print(finger.fingerID); |
|  | Serial.print(" with confidence of "); Serial.println(finger.confidence); |
|  |  |
|  | return finger.fingerID; |
|  | } |
|  |  |
|  | // returns -1 if failed, otherwise returns ID # |
|  | 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; |
|  |  |
|  | // found a match! |
|  | Serial.print("Found ID #"); Serial.print(finger.fingerID); |
|  | Serial.print(" with confidence of "); Serial.println(finger.confidence); |
|  | return finger.fingerID; |
|  | } |
| **Fingerprint Door Lock:**   |  | | --- | | /\* PINOUT | |  | \* Fingerprint Sensor Arduino | |  | \* Vcc(red)----------------> 5v | |  | \* Gnd(Black)--------------> Gnd | |  | \* Tx(Yellow)---------------> Pin 2 | |  | \* Rx(Blue)---------------> Pin 3 | |  | \*/ | |  |  | |  | #include <Adafruit\_Fingerprint.h> | |  |  | |  | // pin #2 is IN from sensor (YELLOW wire) | |  | // pin #3 is OUT from arduino (BLUE wire) | |  | SoftwareSerial mySerial(2, 3); | |  |  | |  | Adafruit\_Fingerprint finger = Adafruit\_Fingerprint(&mySerial); | |  |  | |  | int door\_lock = 12; //connect the door lock to pin 12 | |  | int lock\_delay = 1500; //change the delay from here | |  |  | |  | void setup() | |  | { | |  | pinMode(door\_lock, OUTPUT); | |  | digitalWrite(door\_lock, LOW); | |  | Serial.begin(9600); | |  | finger.begin(57600); | |  | } | |  |  | |  | void loop() | |  | { | |  | getFingerprintIDez(); | |  | delay(50); | |  | } | |  |  | |  | uint8\_t getFingerprintID() { | |  | uint8\_t p = finger.getImage(); | |  | switch (p) { | |  | case FINGERPRINT\_OK: | |  | Serial.println("Image taken"); | |  | break; | |  | case FINGERPRINT\_NOFINGER: | |  | Serial.println("No finger detected"); | |  | return p; | |  | case FINGERPRINT\_PACKETRECIEVEERR: | |  | Serial.println("Communication error"); | |  | return p; | |  | case FINGERPRINT\_IMAGEFAIL: | |  | Serial.println("Imaging error"); | |  | return p; | |  | default: | |  | Serial.println("Unknown error"); | |  | return p; | |  | } | |  |  | |  | // OK success! | |  |  | |  | p = finger.image2Tz(); | |  | 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! | |  | p = finger.fingerFastSearch(); | |  | if (p == FINGERPRINT\_OK) { | |  | Serial.println ("Found a print match!"); | |  | } else if (p == FINGERPRINT\_PACKETRECIEVEERR) { | |  | Serial.println("Communication error"); | |  | return p; | |  | } else if (p == FINGERPRINT\_NOTFOUND) { | |  | Serial.println("Did not find a match"); | |  | return p; | |  | } else { | |  | Serial.println("Unknown error"); | |  | return p; | |  | } | |  |  | |  | // found a match! | |  | Serial.print("Found ID #"); Serial.print(finger.fingerID); | |  | Serial.print(" with confidence of "); Serial.println(finger.confidence); | |  | digitalWrite(door\_lock, HIGH); | |  | delay(lock\_delay); | |  | digitalWrite(door\_lock, LOW); | |  | return finger.fingerID; | |  | } | |  |  | |  | // returns -1 if failed, otherwise returns ID # | |  | 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; | |  |  | |  | // found a match! | |  | Serial.print("Found ID #"); Serial.print(finger.fingerID); | |  | Serial.print(" with confidence of "); Serial.println(finger.confidence); | |  | digitalWrite(door\_lock, HIGH); | |  | delay(lock\_delay); | |  | digitalWrite(door\_lock, LOW); | |  | return finger.fingerID; | |  | } | |  |

**CONCLUSION:-**

**Fingerprint door locks are great investment for home or business. It provides great security by providing restrictions to unwanted access. This device increases level of security by adding unique biological features of authorized person. For anyone who wants more security to their homes, fingerprint door locks are best choice.**