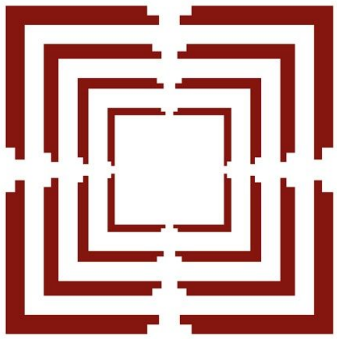


Ahmedabad University

Internet of Things (EVD250)

Biometric Attendance system

Jay Patel (1641011)
Maharsh Suryawala (1641029)
Parshwa Shah (1641068)
Nikhil Balwani (1641070)

Index

Motivation:

Few years back if you were to tell someone that the Geyser and bedroom lights in your home are connected to the internet, they would be baffled and might even criticize it as over-engineered products. But today with the advent of IoT, Smart cities etc the idea no longer sounds strange, we have devices around us that have become smarter by being able to communicate with the internet.

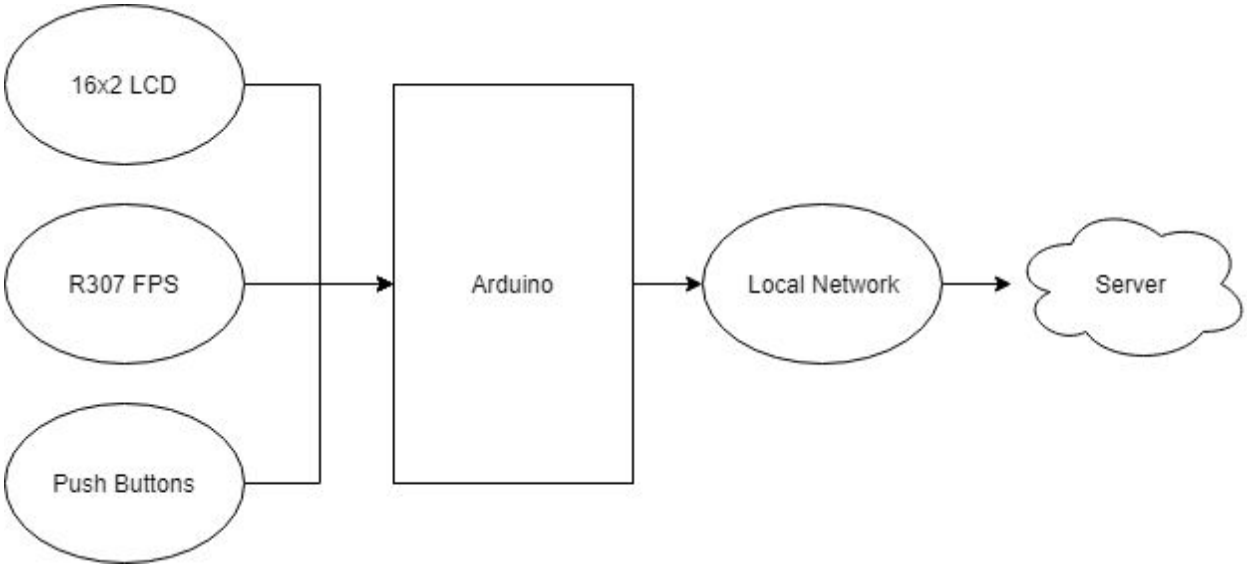
Attendance plays a major role in educational institutions. The most common means of taking attendance in the classroom is by calling out the roll numbers of students or asking the students to manually sign the attendance sheet, which is passed around during the lecture. The process of manually taking and maintaining the attendance records becomes highly cumbersome

Biometric systems have reached a sufficiently advanced stage wherein they can now be deployed in systems without hampering portability. With the recent development of various cloud based computing and storage systems, data can be securely stored and retrieved whenever required. Primarily, fingerprints and iris images are considered to be the most reliable for use in biometric systems.

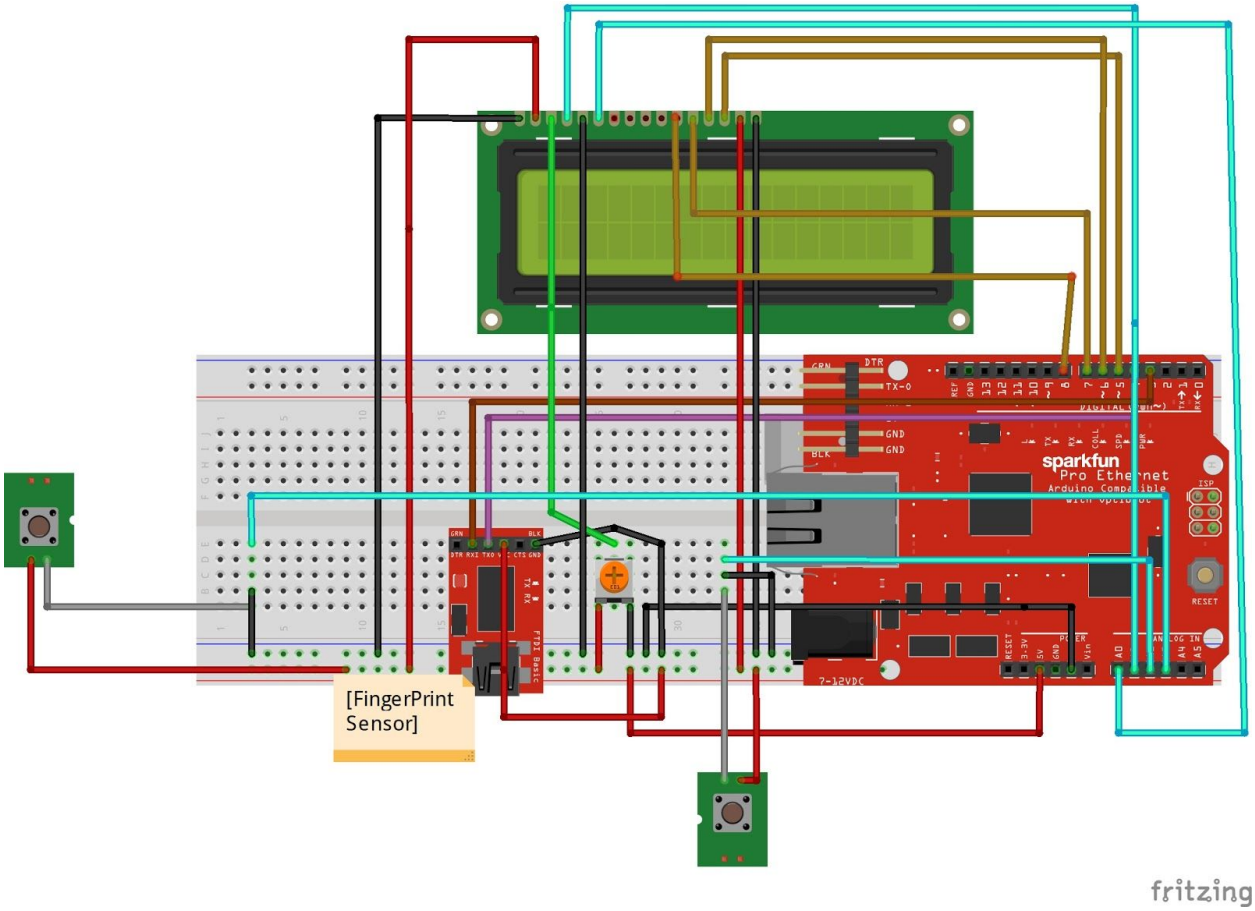
Description And Final Outcome:

In this project our aim is to leverage this IoT into the boring attendance system to make it smart and more effective. Most conventional attendance systems available today store the information over a micro SD card and have to be connected to software via a computer to access the information. Here, we will build a biometric attendance system using Arduino that scans for finger print and on successful identification of the person it will log the information to a cloud platform like ThingsBoard by using the ESP8266 Wi-Fi module. This information can then be displayed in the dashboard of ThingsBoard making it available for the required authorities to view and analysis information over the internet without having any direct physical access to the hardware. However the conventional Attendance system without involving IoT can also be built by following the link and Fingerprint sensor can be further used for many other biometric applications like Voting Machine, Security system etc.

Block Diagram:



Circuit Diagram:



Components Needed:

- Arduino UNO
- 16x2 LCD Display
- Ethernet Shield
- R307 Fingerprint sensor (FPS)
- Jumper wires
- Push Buttons
- Potentiometer
- Resistors

Operating Principle of Components:

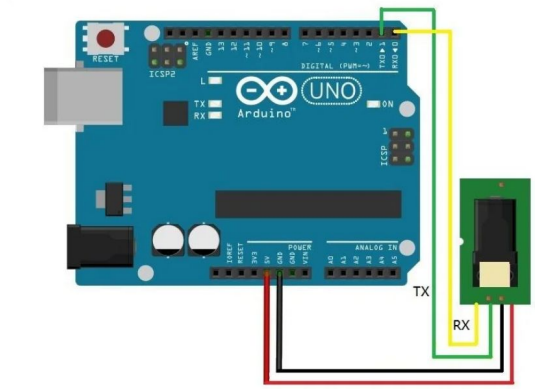
R307 Fingerprint Sensor Module:

The features of this sensor include the following.

- It includes image collection as well as chip algorithm.
- The fingerprint reader can perform lesser growth and can be fixed into a range of end products.
- Power use is low, excellent performance, small in size, and less cost.
- Optical technology which is used is professional, and exact module developed techniques.
- The capabilities of image processing are good, and can effectively capture pictures up to 500 dpi resolution.

  Working Principle:

  The working principle of the fingerprint sensor mainly depends on the processing. The fingerprint processing mainly includes two elements namely enrollment and matching. In fingerprint enrolling, every user requires to place the finger twice. So that the system will check the finger images to process as well as to generate a pattern of the finger and it will be stored. When matching, a user places the finger using an optical sensor then the system will produce a pattern of the finger & compares it with the finger library templates. For 1:1 fingerprint matching, the system will evaluate the exits finger with a precise pattern which is selected within the module. Similarly, for 1: N matching, the scanning system will look for the complete finger records for the finger matching. In both situations, the scanning system will go back to the corresponding result, success otherwise crash.

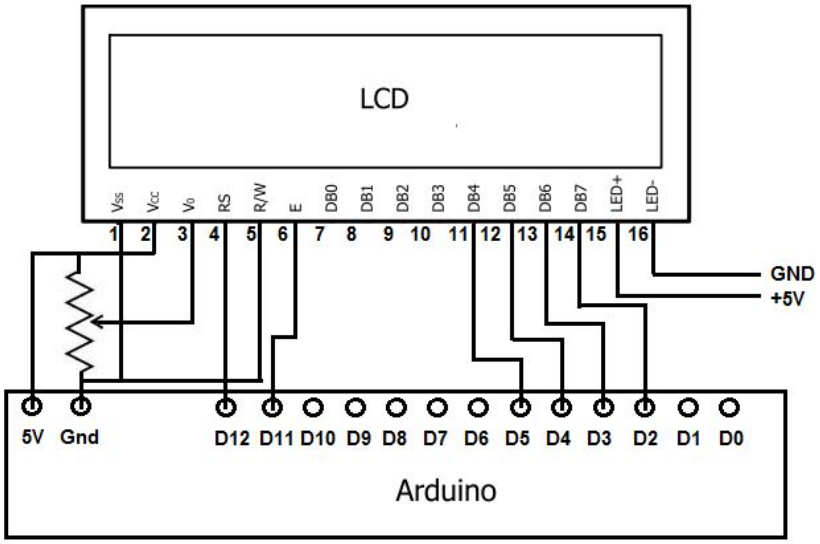


Fingerprint Sensor Interfacing using Arduino

16x2 LCD Display

Character lcd's come in many sizes 8x1, 8x2, 10x2, 16x1, 16x2, 16x4, 20x2, 20x4, 24x2, 30x2, 32x2, 40x2 etc . Many multinational companies like Philips, Hitachi, Panasonic make their own custom type of character lcd's to be used in their products. All character lcd's performs the same functions(display characters numbers special characters, ascii characters etc).Their programming is also the same and they all have the same 14 pins (0-13) or 16 pins (0 to 15).

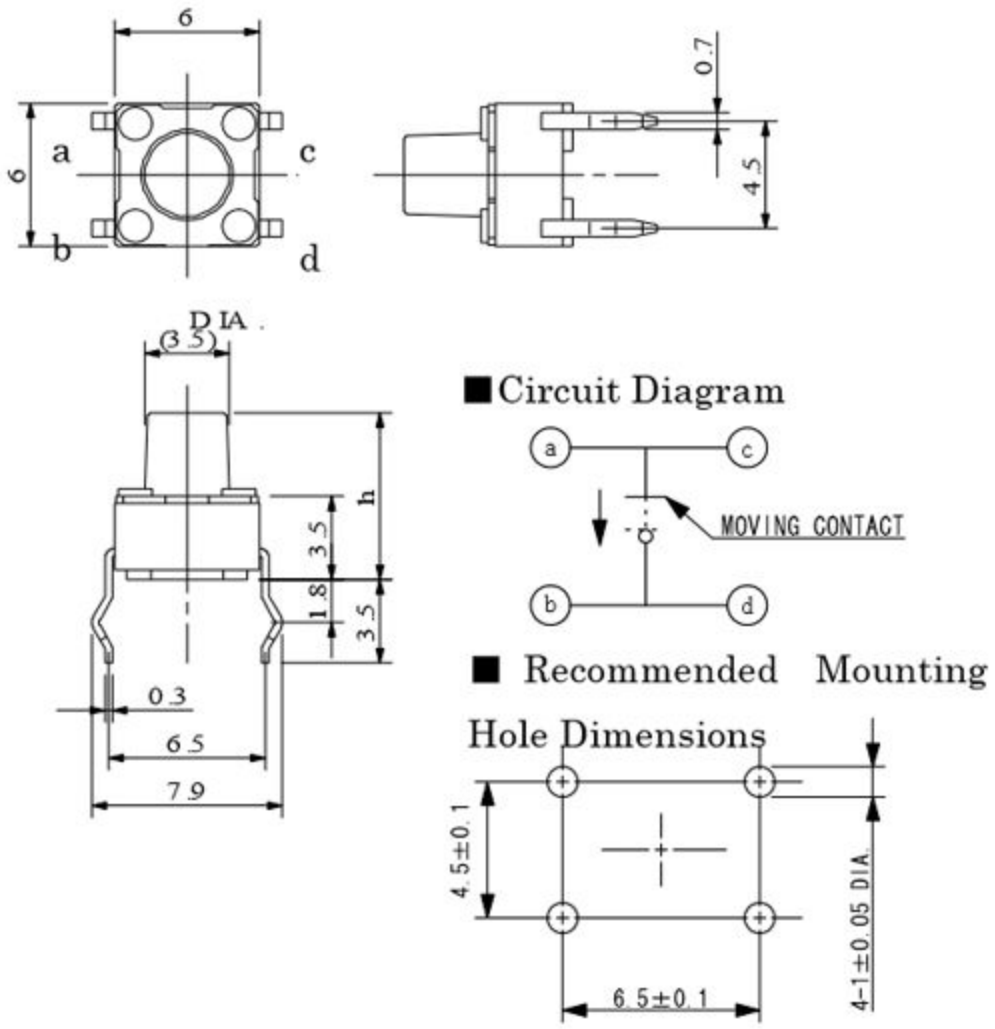
In an m x n lcd. M denotes number of coulombs and n represents number of rows. Like if the lcd is denoted by 16x2 it means it has 16 coulombs and 2 rows. On a character lcd a character is generated in a matrix of 5x8 or 5x7. Where 5 represents number of coulombs and 7/8 represent number of rows. Maximum size of the matrix is 5x8. You can not display character greater than 5x8 dimension matrix. Normally we display a character in 5x7 matrix and left the 8th row for the cursor. If we use the 8th row of the matrix for the character display, then their will be no room for cursor. The picture below shows the 5x8 dot matrix pixels arrangement.



LCD Interfacing using Arduino

Push button

A Push Button is a type of switch works on a simple mechanism called “Push-to-make”. Initially, it remains in off state or normally open state but when it is pressed, it allows the current to pass through it or we can say it makes the circuit when pressed. Normally their body is made up of plastic or metal in some types.



The working concept of Push Button is given above, till the button pressed it conducts current through it or make the circuit. As the button released it break the circuit again.

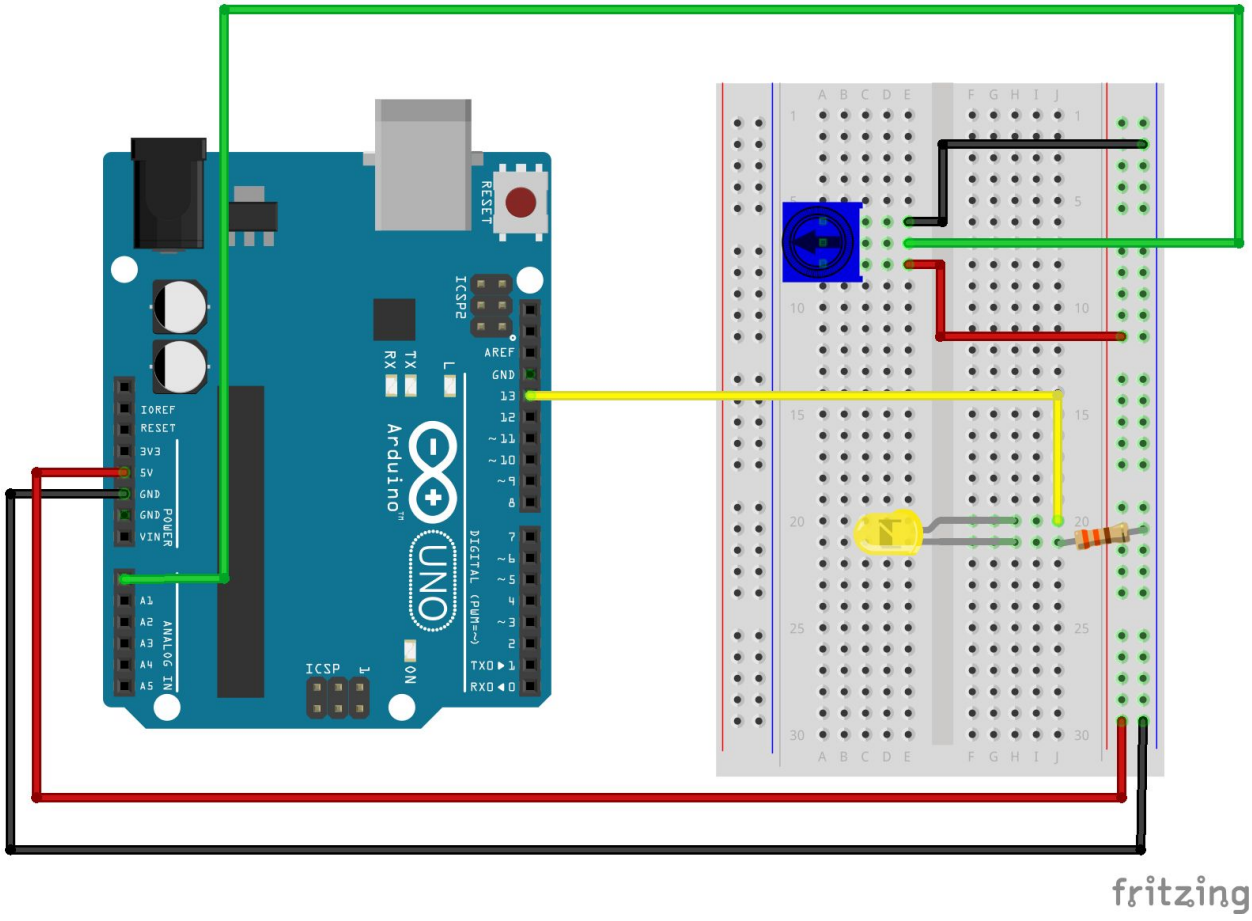
Potentiometer

A potentiometer is an instrument for measuring voltage by comparison of an unknown voltage with a known reference voltage. If a sensitive indicating instrument is used, very little current is drawn from the source of the unknown voltage. Since the reference voltage can be produced from an accurately calibrated voltage divider, a potentiometer can provide high precision in measurement. The method was described by Johann Christian Poggendorff around 1841 and became a standard laboratory measuring technique.

Working Principle:

The principle of a potentiometer is that the potential dropped across a segment of a wire of uniform cross-section carrying a constant current is directly proportional to its length. The potentiometer is a simple device used to measure the electrical potentials(or compare the e.m.f of a cell). One form of potentiometer is a uniform high-resistance wire attached to an insulating support, marked with a linear measuring scale. In use, an adjustable regulated voltage source E, of greater magnitude than the potential to be measured, is connected across the wire so as to pass a steady current through it.

Between the end of the wire and any point along it will be a potential proportional to the length of wire to that point. By comparing the potential at points along the wire with an unknown potential, the magnitude of the unknown potential can be determined. The instrument used for comparison must be sensitive, but need not be particularly well-calibrated or accurate so long as its deflection from zero position can be easily detected.



Ethernet Shield

The W5100 is a versatile single-chip network interface chip with a 10/100Mbs Ethernet controller integrated internally. It is mainly used in a high integrated, high stable, high performance and low cost embedded system. The W5100 enables you to connect to the Internet without operation system. It is also compatible with IEEE802.3 10BASE-T and 802.3u 100BASE-TX.

W5100 integrates a market-proven full-hardware TCP/IP protocol stack inside, Ethernet media access control (MAC) layer and physical layer (PHY). The hardware TCP/IP protocol stack supports the following protocols: TCP, UDP, IPV4, ICMP, ARP, IGMP and PPoE, which have been tested by the market for years in many fields. In addition, W5100 also integrates 16KB memory for data transmission. For W5100, you don't need to consider the control of the Ethernet; what you need to do is socket programming.

The W5100 has three interfaces: direct parallel bus, indirect parallel bus and SPI bus. It is easy to connect it with an MCU, just like accessing an external memory. Here we use the Ethernet library to apply W5100 more easily and conveniently.

Explanation

There are two push buttons one for check-in and other for check out. On pressing check in button and putting finger on sensor, fingerprint of person is detected and verified. Time corresponding to check in will be inserted in database. At the time of check-out person is identified by his fingerprint and corresponding time will be inserted in database.

Arduino code for interfacing:

R307 Fingerprint Module

enroll.ino

```
#include <Adafruit_Fingerprint.h>



SoftwareSerial mySerial(2, 3);

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 Adafruit 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); }
  }
}

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;
  }
}
```

fingerprint.ino

```
#include <Adafruit_Fingerprint.h>

SoftwareSerial mySerial(2, 3);
```

```
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

void setup()
{
  Serial.begin(9600);
  while (!Serial);
  delay(100);
  Serial.println("\n Adafruit finger detect test");

  finger.begin(57600);

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

  finger.getTemplateCount();
  Serial.print("Sensor contains ");
Serial.print(finger.templateCount); Serial.println(" templates");
  Serial.println("Waiting for valid finger...");
}

void loop()
{
  getFingerprintID();
  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;
}


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;
}

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;
  }

  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of ");
Serial.println(finger.confidence);

  return finger.fingerID;
}

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(" with confidence of ");
Serial.println(finger.confidence);
  return finger.fingerID;
}
```

16x2 LCD Display

```
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 10, d5 = 9, d6 = 8, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("Hello, World!");
}

void loop() {
  // Turn off the display:
  lcd.noDisplay();
  delay(500);
  // Turn on the display:
  lcd.display();
  delay(500);
}
```

Arduino Ethernet Shield

```
#include <SPI.h>
#include <Ethernet.h>

// Enter a MAC address for your controller below.
// Newer Ethernet shields have a MAC address printed on a sticker on
the shield
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

// if you don't want to use DNS (and reduce your sketch size)
// use the numeric IP instead of the name for the server:
//IPAddress server(74,125,232,128);  // numeric IP for Google (no
DNS)
```

```
char server[] = "www.google.com";     // name address for Google
(using DNS)

// Set the static IP address to use if the DHCP fails to assign
IPAddress ip(192, 168, 0, 177);
IPAddress myDns(192, 168, 0, 1);

// Initialize the Ethernet client library
// with the IP address and port of the server
// that you want to connect to (port 80 is the default for HTTP):
EthernetClient client;

// Variables to measure the speed
unsigned long beginMicros, endMicros;
unsigned long byteCount = 0;
bool printWebData = true;  // set to false for better speed
measurement

void setup() {
  // You can use Ethernet.init(pin) to configure the CS pin
  //Ethernet.init(10);  // Most Arduino shields
  //Ethernet.init(5);   // MKR ETH shield
  //Ethernet.init(0);   // Teensy 2.0
  //Ethernet.init(20);  // Teensy++ 2.0
  //Ethernet.init(15);  // ESP8266 with Adafruit Featherwing Ethernet
  //Ethernet.init(33);  // ESP32 with Adafruit Featherwing Ethernet

  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
      ; // wait for serial port to connect. Needed for native USB
port only
  }

  // start the Ethernet connection:
  Serial.println("Initialize Ethernet with DHCP:");
  if (Ethernet.begin(mac) == 0) {
      Serial.println("Failed to configure Ethernet using DHCP");
```

```
    // Check for Ethernet hardware present
    if (Ethernet.hardwareStatus() == EthernetNoHardware) {
    Serial.println("Ethernet shield was not found.  Sorry, can't
run without hardware. :(");
    while (true) {
    delay(1); // do nothing, no point running without Ethernet
hardware
    }
    }
    if (Ethernet.linkStatus() == LinkOFF) {
    Serial.println("Ethernet cable is not connected.");
    }
    // try to configure using IP address instead of DHCP:
    Ethernet.begin(mac, ip, myDns);
  } else {
    Serial.print("  DHCP assigned IP ");
    Serial.println(Ethernet.localIP());
  }
  // give the Ethernet shield a second to initialize:
  delay(1000);
  Serial.print("connecting to ");
  Serial.print(server);
  Serial.println("...");

  // if you get a connection, report back via serial:
  if (client.connect(server, 80)) {
    Serial.print("connected to ");
    Serial.println(client.remoteIP());
    // Make a HTTP request:
    client.println("GET /search?q=arduino HTTP/1.1");
    client.println("Host: www.google.com");
    client.println("Connection: close");
    client.println();
  } else {
    // if you didn't get a connection to the server:
    Serial.println("connection failed");
  }
  beginMicros = micros();
```

```
}

void loop() {
  // if there are incoming bytes available
  // from the server, read them and print them:
  int len = client.available();
  if (len > 0) {
      byte buffer[80];
      if (len > 80) len = 80;
      client.read(buffer, len);
      if (printWebData) {
      Serial.write(buffer, len); // show in the serial monitor (slows
some boards)
      }
      byteCount = byteCount + len;
  }

  // if the server's disconnected, stop the client:
  if (!client.connected()) {
      endMicros = micros();
      Serial.println();
      Serial.println("disconnecting.");
      client.stop();
      Serial.print("Received ");
      Serial.print(byteCount);
      Serial.print(" bytes in ");
      float seconds = (float)(endMicros - beginMicros) / 1000000.0;
      Serial.print(seconds, 4);
      float rate = (float)byteCount / seconds / 1000.0;
      Serial.print(", rate = ");
      Serial.print(rate);
      Serial.print(" kbytes/second");
      Serial.println();

      // do nothing forevermore:
      while (true) {
      delay(1);
      }
```

```
  }
}
```

Final Program

```
#include <SPI.h>
#include <Ethernet.h>
#include <LiquidCrystal.h>
#include <EEPROM.h>
#include <Ethernet.h>
#include <Adafruit_Fingerprint.h>

SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
uint8_t id;
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = {192, 168, 0, 100}; //Enter the IP of ethernet shield
byte serv[] = {192, 168, 0, 102} ; //Enter the IPv4 address
LiquidCrystal lcd(A1, A0, 8, 7, 6, 5);
EthernetClient cliente;

void setup() {
  lcd.begin(16, 2);
  lcd.clear();
  Serial.begin(9600); //setting the baud rate at 9600
  Ethernet.begin(mac, ip);
  pinMode(A0, OUTPUT);

  Serial.begin(9600);
  while (!Serial);
  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!");
      lcd.println("Found sensor!");
  } else {
      Serial.println("Did not find fingerprint sensor :(");
      lcd.println("Not found :(");
      /*while (1) { delay(1); }*/
      delay(10000);
  }
}

uint8_t readnumber(void) {
  uint8_t num = 0;

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

}

String readstring(void) {
  String str = "";

  while (str == "") {
      while (! Serial.available());
      str = Serial.readString();
  }
  return str;

}

void loop() {

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Fingerprint");
```

```
  lcd.setCursor(0, 1);
  lcd.print("System");


  if (digitalRead(A3) == HIGH) {
      int finger_id = getFingerprintID();

      Serial.print(finger_id);

      if (finger_id != 254) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Welcome!" + finger_id);



      if (cliente.connect(serv, 80)) { //Connecting at the IP address
and port we saved before
      Serial.println("connected");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Connected");

      String query = "GET /fingerprint-database/data.php?";
      query += "id=";
      query += finger_id;
      query += "&in_out=";
      query += 1;
      Serial.println(query);
      cliente.println(query); //Connecting and Sending values to
database



      cliente.stop(); //Closing the connection
      Serial.print("Closed");
      delay(5000);
      }
```

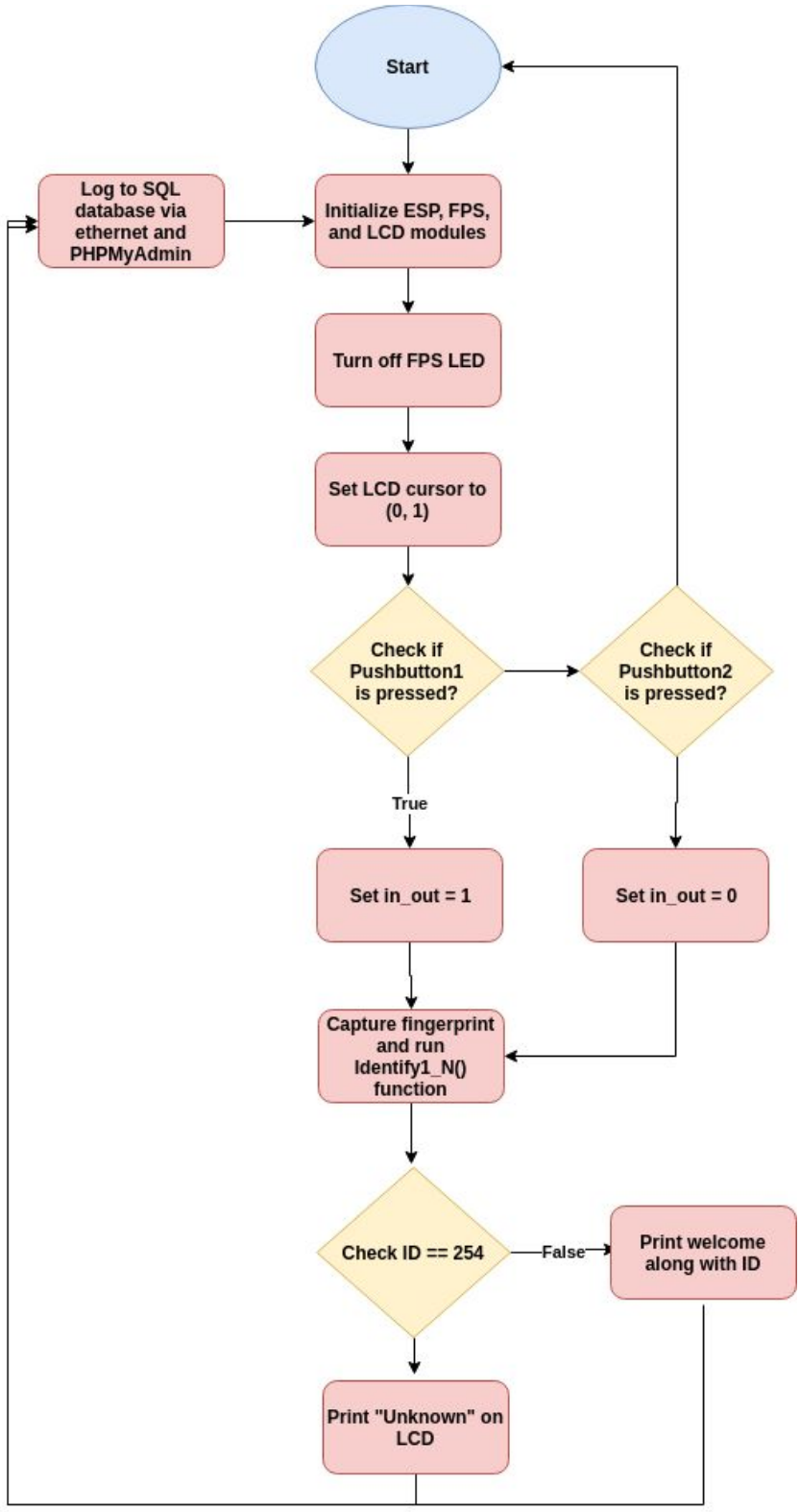
```
      else {
      // if you didn't get a connection to the server:
      Serial.println("connection failed");
  }

      }

      delay(5000);
  }


      if (digitalRead(A2) == HIGH) {
      int finger_id = getFingerprintID();

      Serial.print(finger_id);

      if (finger_id != 254) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Welcome!" + finger_id);


      if (cliente.connect(serv, 80)) { //Connecting at the IP address
and port we saved before
      Serial.println("connected");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Connected");

      String query = "GET /fingerprint-database/data.php?";
      query += "id=";
      query += finger_id;
      query += "&in_out=";
      query += 0;
      Serial.println(query);
      cliente.println(query); //Connecting and Sending values to
database
      cliente.stop(); //Closing the connection
```

```
      Serial.print("Closed");
      delay(5000);
      }
      else {
      // if you didn't get a connection to the server:
      Serial.println("connection failed");
  }

      }
      delay(5000);
  }

}


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;
  }


  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;
}
int flag = 0;

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 254;

    return p;
} else {
    Serial.println("Unknown error");
    return p;
}

Serial.print("Found ID #"); Serial.print(finger.fingerID);
```

```
  Serial.print(" with confidence of ");
Serial.println(finger.confidence);

  return finger.fingerID;
}

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(" with confidence of ");
Serial.println(finger.confidence);
  return finger.fingerID;
}
```

Flowchart:


Start
Log to SQL database via ethernet and PHPMyAdmin
Initialize ESP, FPS, and LCD modules
Turn off FPS LED
Set LCD cursor to (0, 1)
Check if Pushbutton1 is pressed?
Check if Pushbutton2 is pressed?
True
Set in_out = 1
Set in_out = 0
Capture fingerprint and run Identify1_N() function
Check ID == 254
False
Print welcome along with ID
Print "Unknown" on LCD

Timeline:

	15/08/19	20/08/19	24/08/19	27/08/19	30/08/19	01/09/19
Report 1	X					
Coding Modules		X				
Hardware Implementation			X			
Server Side Implementation				X		
Final Implementation and testing					X	
Final Report						X

References:

1. Shoewu, O., and O. A. Idowu. "Development of attendance management system using biometrics." *The Pacific Journal of Science and Technology* 13.1 (2012): 300-307.
2. Taxila, Punjab. "Development of academic attendance monitoring system using fingerprint identification." *IJCSNS* 9.5 (2009): 164.
3. Janiak, Martin, et al. "Biometric authentication device for use with token fingerprint data storage." U.S. Patent Application No. 09/683,049
4. https://www.microcontroller-project.com/16x2-lcd-working.html
5. https://www.elprocus.com/fingerprint-sensor-working-and-applications/

Appendix:

16x2 LCD

1. 5x8 dots with cursor
2. 16characters *2lines display
3. 4-bit or 8-bit MPU interfaces
4. Built-in controller (ST7066 or equivalent)
5. Display Mode & Backlight Variations
6. ROHS Compliant

LCD type	☐TN			
	☐FSTN	☑FSTN Negative		
	☐STN Yellow Green	☐STN Gray	☐STN Blue Negative	
View direction	☑6 O'clock	☐12 O'clock		
Rear Polarizer	☐Reflective	☐Transflective	☑Transmissive	
Backlight Type	☑LED	☐EL	☐Internal Power	☑3.3V Input
		☐CCFL	☑External Power	☐5.0V Input
Backlight Color	☑White	☐ Blue	☐ Amber	☐Yellow-Green
Temperature Range	☑Normal	☐Wide	☐Super Wide	
DC to DC circuit	☐Build-in	☑Not Build-in		
Touch screen	☐With	☑Without		
Font type	☑English-Japanese	☐English-Europen	☐English-Russian	☐other

R307 Fingerprint Sensor

Power	DC 3.6V-6.0V	**Interface**	UART(TTL logical level)/ USB 1.1
Working current	Typical: 100mA Peak: 150mA	**Matching Mode**	1:1 and 1:N
Baud rate	(9600*N)bps, N=1～12 (default N=6)	**Character file size**	256 bytes
Image acquiring time	<0.5s	**Template size**	512 bytes
Storage capacity	256	**Security level**	5 (1, 2, 3, 4, 5(highest))
FAR	<0.001%	**FRR**	<0.1%
Average searching time	< 1s (1:1000)	**Window dimension**	18mm*22mm
Working environment	Temp: -10℃- +40℃	**Storage environment**	Temp: -40℃- +85℃
	RH: 40%-85%		RH: <85%
Outline Dimention	Split type	Module: 32*23*7mm Sensor:56*20*21.5mm	
	Integral type	54.5*20.6*23.8mm	

Ethernet Shield

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage Plug(limits)	6-18V
Input Voltage PoE (limits)	36-57V
Digital I/O Pins	14 (of which 4 provide PWM output)
Arduino Pins reserved:	
	10 to 13 used for SPI
	4 used for SD card
	2 W5100 interrupt (when bridged)
Analog Input Pins	6
DC Current per I/O Pin	40 mA

DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
W5100 TCP/IP Embedded Ethernet Controller	
Power Over Ethernet ready Magnetic Jack	
Micro SD card, with active voltage translators	