DESCRIPTION OF MODULES

- Configuring the R307 Fingerprint Module with Arduino and Bluetooth Module

We will be using the R307 sensor module that makes a serial communication with a controller such as Arduino to exchange data. First we set our finger print for R301T as a user and connect it to Arduino Tx and Rx pins. Followed by using a Bluetooth module called HC-05 and pair it with our devices one time. Now we will write a code for Arduino and say it when a user activate gadget turn the Bluetooth on.

- Circuit involved

Connect Enable and VCC pins of HC-05 together. Connect Tx and Rx Serial pins of Fingerprint Sensor to pins 2 and 3 of Arduino. For LED connection to VCC we will we requiring to use a resistor.

- Code

We must add the fingerprint sensor's library in the Arduino IDE and then upload the code to our Arduino Nano R3 by connecting it to our pc and setting the COM port in tools and port section.

- Circuit involved

Connect Enable and VCC pins of HC-05 together. Connect Tx and Rx Serial pins of Fingerprint Sensor to pins 2 and 3 of Arduino. For LED connection to VCC we will we requiring to use a resistor.

- Bluetooth Module

First, you should pair the Bluetooth module with your device. To pair with Win10, go to settings and search sign-in options and enable dynamic lock and pair your BT module from there. For Android smartphones, go to Settings > Display section > Lock screen then Enable Smart Lock from there and pair with BT. For IOS, Go to Settings, enter Touch ID and Passcode and make it from there.

- HC-05 Bluetooth serial wireless module

It is used for many applications like wireless headset, game controllers, wireless mouse, wireless keyboard and many more consumer applications. It has range up to <100m which depends upon transmitter and receiver, atmosphere, geographic & urban conditions. It is IEEE 802.15.1 standardized protocol, through which one can build wireless Personal Area Network (PAN). It uses frequency-hopping spread spectrum (FHSS) radio technology to send data over air.

- Arduino Nano R3

Arduino boards are widely used in robotics, embedded systems, automation, Internet of Things (IoT) and electronics projects. These boards were initially introduced for the students and non-technical users but nowadays Arduino boards are widely used in industrial projects. Arduino Nano is a small, complete, flexible, and breadboard-friendly Microcontroller board, based on ATmega328p, developed by Arduino.cc in Italy in 2008 and contains 30 male I/O headers, configured in a DIP30 style. Arduino Nano Pinout contains 14 digital pins, 8 analog pins, 2 Reset Pins & 6 Power Pins. It is programmed using Arduino IDE, which can be downloaded from Arduino Official site.

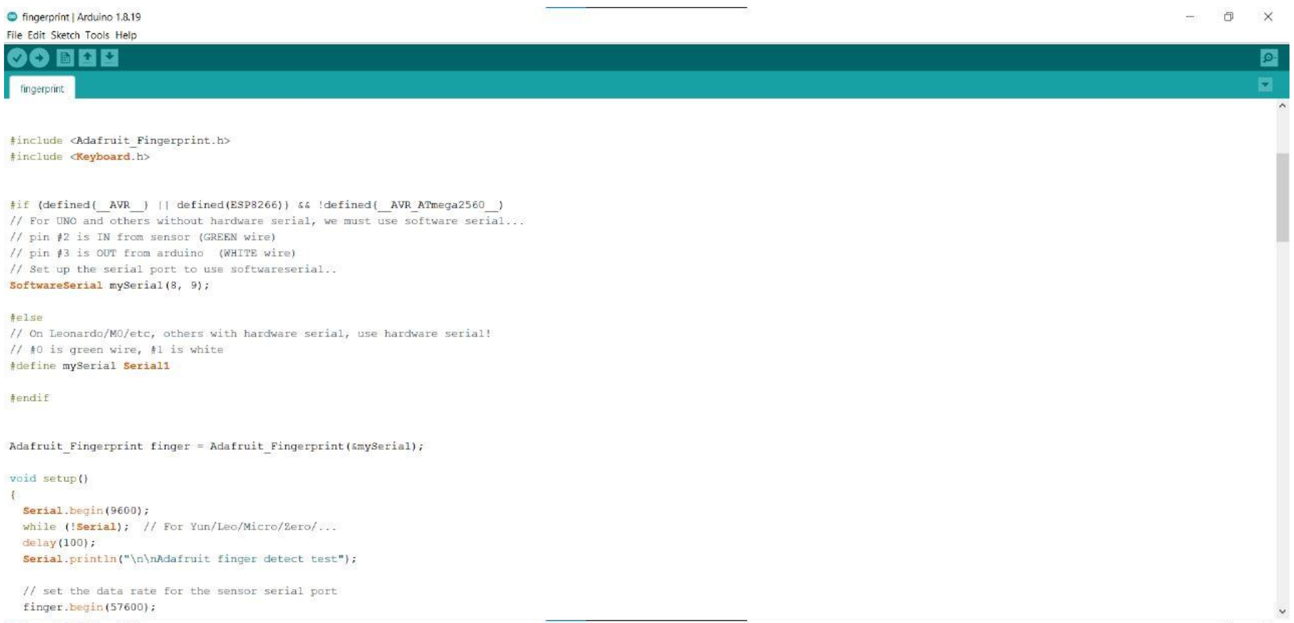
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
ICSP
ARDUINO.CC
ARDUINO NANO V3.0
GRAVITECH.US
USA
2009
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN

- 5mm Rgb Tri-Color 4 Pin Led

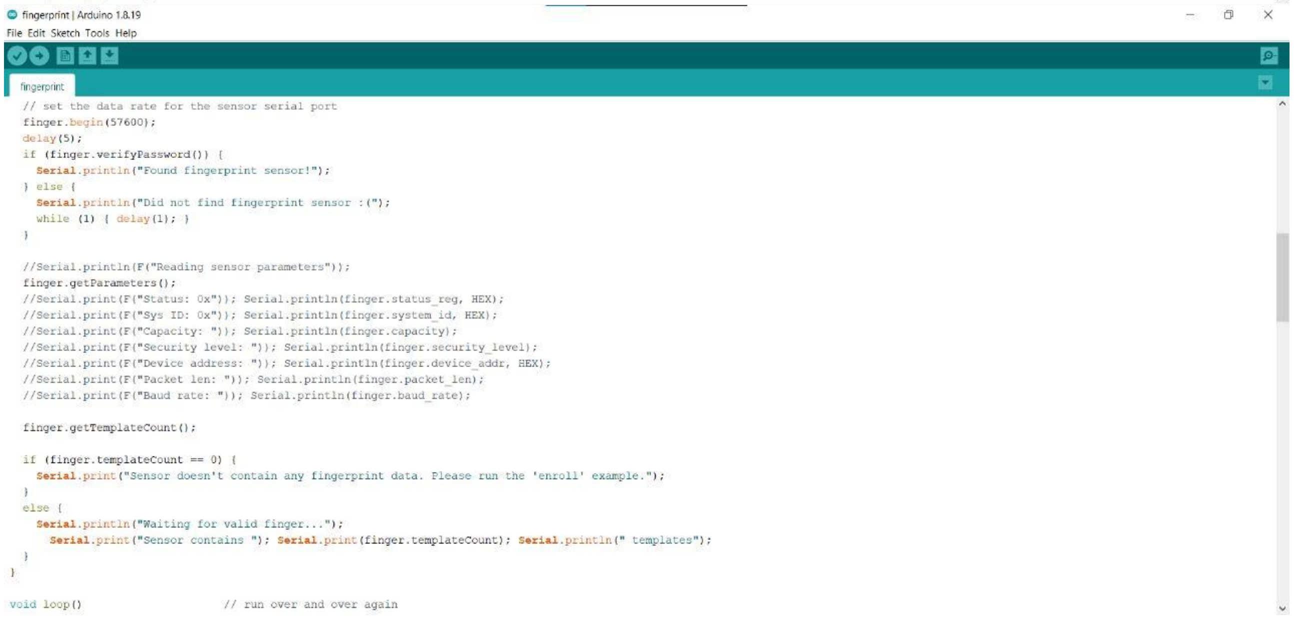
RGB 4-pin LED allows generation of colours across the visible spectrum. The 4-pin package is most seen on RGB (red-green-blue) LEDs.

IMPLEMENTATION

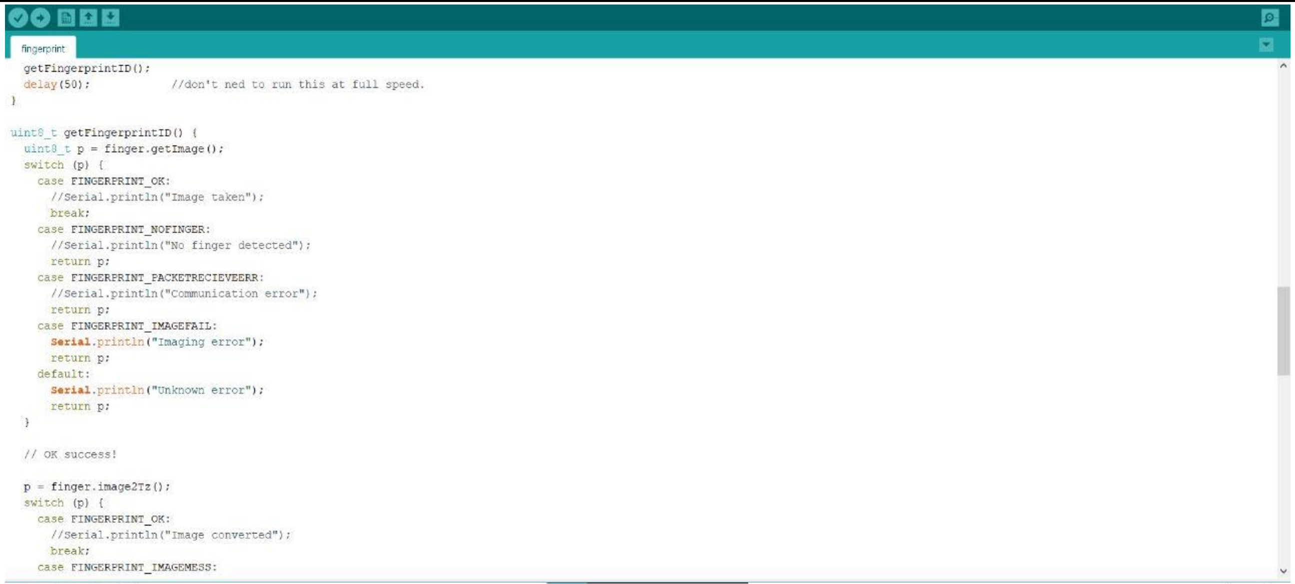
Code for Fingerprint sensor and unlocking and accessing the devices:



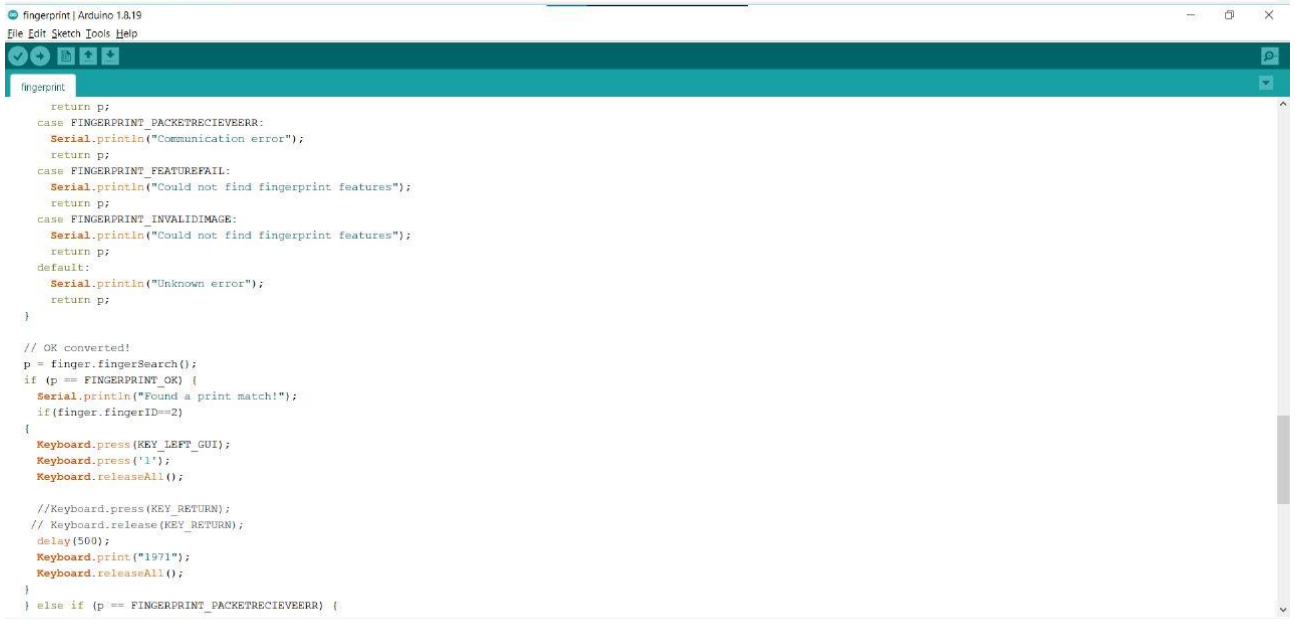
```
#include <Adafruit_Fingerprint.h>
#include <Keyboard.h>


#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(8, 9);

#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);
```



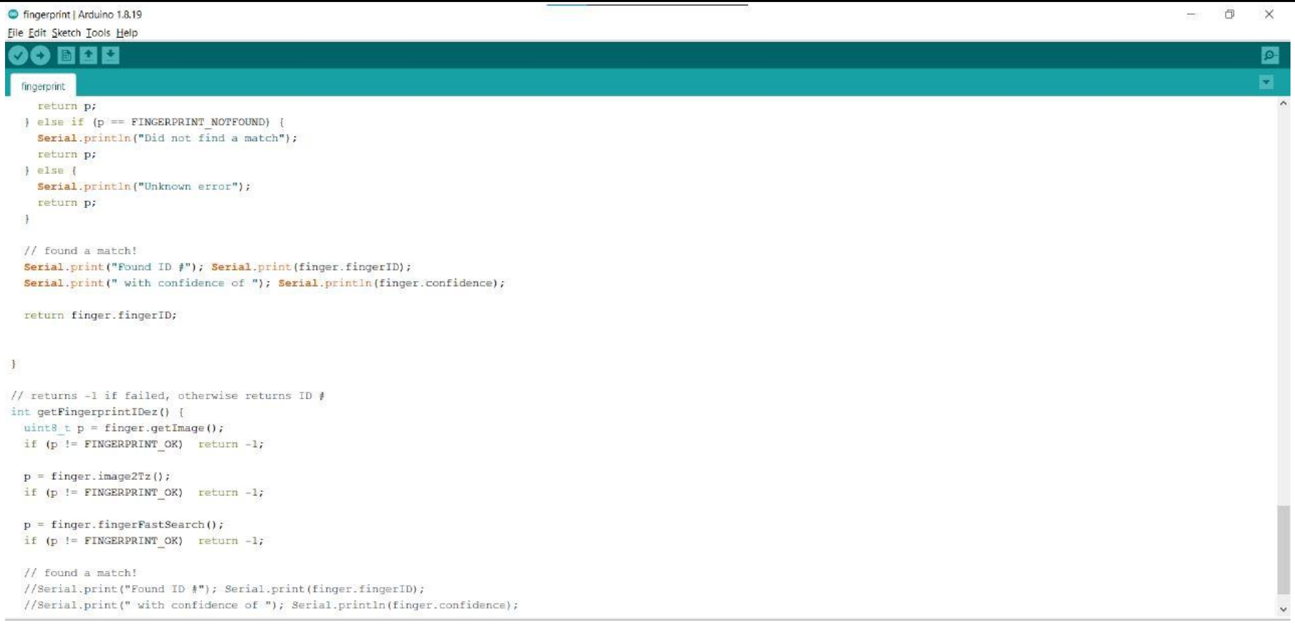
```
  // set the data rate for the sensor serial port
  finger.begin(57600);
  delay(5);
  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1) { delay(1); }
  }

  //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:
```



```
      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!");
    if(finger.fingerID==2)
  {
    Keyboard.press(KEY_LEFT_GUI);
    Keyboard.press('1');
    Keyboard.releaseAll();

    //Keyboard.press(KEY_RETURN);
   // Keyboard.release(KEY_RETURN);
    delay(500);
    Keyboard.print("1971");
    Keyboard.releaseAll();
  }
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
```



```
    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);
```

TESTING (ALL SCREEN SNAPSHOTS)