TABLE OF CONTENTS

ABSTRACT

From earlier times, security was and also till now is an issue of concern in our households and also in office, shops, etc. Everyone has a fear of unauthorized person entering to their home or office without their knowledge. The normal door can be fitted with locks which are capable of breaking with the use of an alternate key. Alternatives to this system can be found like the password or pattern system in the locks which again has the possibility of getting exposed and opening the lock. So, a solution to such problems can be by combining door lock with biometrics. Biometric verification is any means by which a person can be uniquely identified by evaluating one or more distinguishing biological traits.Unique identifiers include fingerprints, hand geometry, earlobe geometry, retina and iris patterns, voice waves, DNA, and signatures. Here we will use fingerprint for biometric verification as it is one such thing which is unique to every individual and the use of fingerprint as the key to door locks can overcome the security problem of unauthorized people trespassing to our homes, shops, offices, etc to a great extent as duplicacy in such key is not possible. Also, this system will not lead to problems like losing keys because we do not require carrying keys if this system is used instead of traditional locks. So, using arduino we will try to implement the system with features which will increase the security level**.**

Keywords: Arduino, IOT, Fingerprint, Security, Locks.

LIST OF FIGURES

CHAPCTER 1
INTRODUCTION

In this Paper we are trying to solve the problems which occurs related to the security in homes, shops and offices. These issues can be fixed by using traditional locks but here a possibility is may occurred of some unknown person will open the lock without breaking it by using duplicate keys. Using these locks also make problems if we lost keys of lock and we have to carry those keys with us. Again, using these patterns in the locks can improve security but again it can open and cracked if somehow someone guesses the passwords or patterns are known. The pattern which can or will used as key will be unique. Here, to implement this project successfully we will use fingerprint as the key. This Arduino UNO project will make use of different devices to the implementation of the advanced security lock where there are different features to maximize the security levels. II.

1.1BLOCK DIAGRAM OF DOOR UNLOCK USING FINGER PRINT

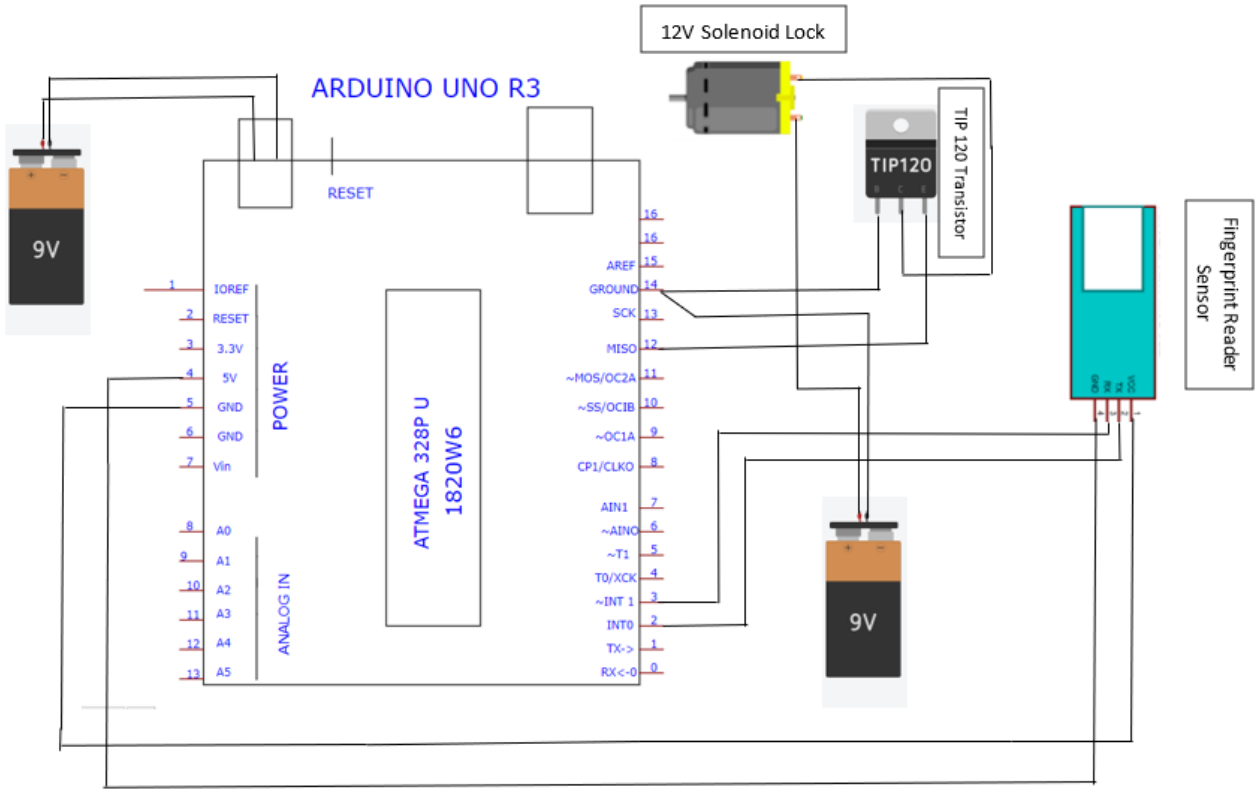

Figure 1.1. BLOCK DIAGRAM

1.2 ARDUINO UNO

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Its hardware products are licensed under a CC BY-SA license, while software is licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL),[1] permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially from the official website or through authorized distributors. Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards ('shields') or breadboards (for prototyping) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs. The microcontrollers can be programmed using the C and C++ programming languages, using a standard API which is also known as the Arduino language, inspired by the Processing language and used with a modified version of the Processing IDE. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) and a command line tool developed in Go.

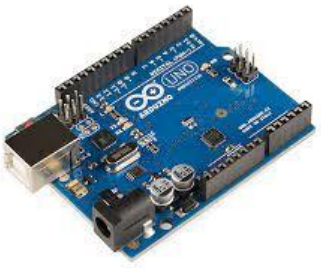

Figure 1.2. Arduino Uno

1.3FINGER PRINT SENSOR

The fingerprint sensor is one kind of sensor which is used in a fingerprint detection device. These devices are mainly inbuilt in the fingerprint detection module and it is used for computer safety. The main features of this device mainly include accuracy, better performance, robustness based on exclusive fingerprint biometric technology. Both fingerprint scanner otherwise reader are an extremely safe & suitable device for safety instead of a secret word. Because the password is easy to scan and also it is hard to keep in mind. There are different types of fingerprint modules available in the market like R305, R307. For a better understanding of this sensor, here we are going to discuss an overview of R305 fingerprint sensor module. The R305 is one kind of fingerprint sensor module used in biometrics for security in fingerprint detection as well as verification. These devices are mainly used in safes where there is a high-powered DSP chip used in the rendering of image, feature-finding, searching and calculation by connecting it to any microcontroller with the help of TTL serial, & send data packets to get photos, notice prints, search and hash. The enrollment of new fingers can be stored directly within the flash memory of on board.

Figure 1.3. Fingerprint Sensor Module(R307)

1.4 12V SOLENOID LOCK

Solenoids are basically electromagnets: they are made of a big coil of copper wire with an armature (a slug of metal) in the middle. When the coil is energized, the slug is pulled into the center of the coil. This makes the solenoid able to pull from one end. This solenoid in particular is nice and strong, and has a slug with a slanted cut and a good mounting bracket. It's basically an electronic lock, designed for a basic cabinet or safe or door. Normally the lock is active so you can't open the door because the solenoid slug is in the way does not use any power in this state. When 9-12VDC is applied, the slug pulls in so it doesn't stick out anymore and the door can be opened. The solenoids come with the slanted slug as shown above, but you can open it with the two Phillips-head screws and turn it around so its rotated 90, 180 or 270 degrees so that it matches the door you want to use it with. To drive a solenoid you will a power transistor and a diode, check this diagram for how to wire it to an Arduino or other microcontroller. You will need a fairly good power supply to drive a solenoid, as a lot of current will rush into the solenoid to charge up the electro-magnet, about 500mA, so don't try to power it with a 9V battery!

Figure 1.4. 12V Solenoid Lock

1.5 TIP 120 TRANSISTOR

The TIP120 is an NPN Power Darlington Transistor. It can be used with an Arduino to drive motors, turn lights on, and drive other high power gadgets. The TIP120 acts as a power broker or gatekeeper between the Arduino realm and the high power realm composed of the PC fan and its battery pack. The Arduino can tell the TIP120 how much power to pass from the external battery pack to the PC fan but the Arduino does not share any of its power or share pins with the PC fan or its batteries. The TIP120 is the go in between. The TIP120 has three pins. One is called Base, which we will connect to any of the Arduino PWM pins. Through the Base pin, the Arduino can tell the TIP120 how much power to supply to the motor from the external battery pack. That's it. The TIP120 does the heavy lifting while Arduino sits back and gives orders through one of its PWM pins to the TIP120 Base pin telling it how much power to pass to the motor. The poor TIP120 has to then pass the requested power from the external power to the motor based on Arduino's request

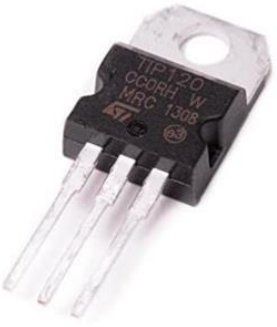

Figure 1.5. TIP 120 Darlington Transistor

1.6 RESULTS AND DISCUSSION

This IOT based Fingerprint door lock using Arduino, we are showing the components and connected them to the power supply. This system is based for improving the security which will register the owner's fingerprint into the Arduino using the fingerprint sensor, and this system we have given 5v power supply to Arduino through the code uploading wire. When you put your thumb on fingerprint sensor after registering yourself the lock will be unlocked and you repeat this process again then the solenoid lock will be got locked. The process of locking and unlocking requires less than 1 second so this is why the Solenoid lock is used inside this project.

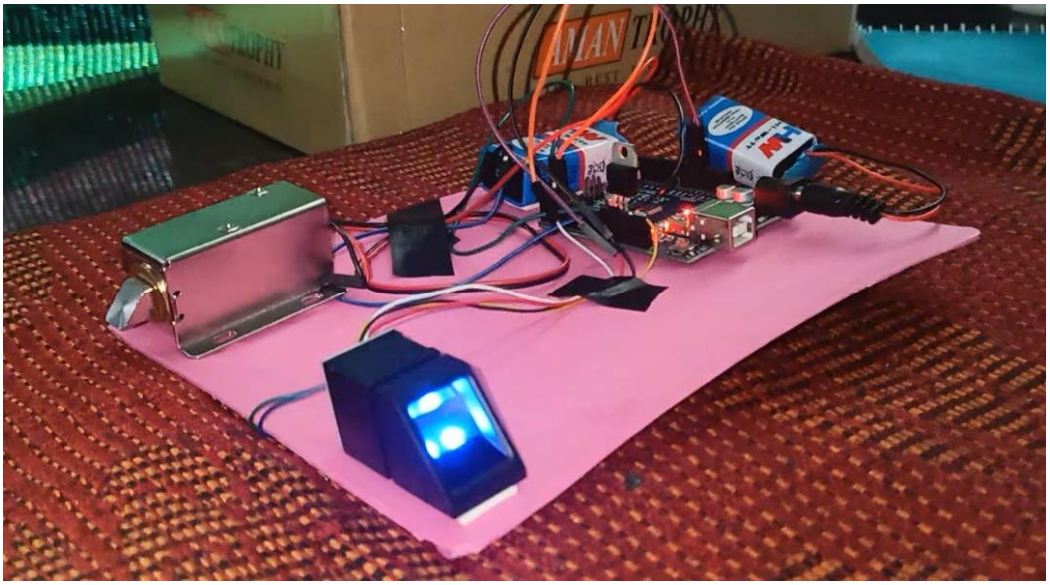

Figure:1.6:Implementation of Project

1.7 Problem Statement

Problem statements are the problems identified by observing the existing door access system. Through observation, there are three problems that have been identified in order to develop this project which are:-

1- Obsolete components

The existence of new technologies comes from the creation of new components. This fact shows that once the old devices are obsolete, there are no spare components could replace it. Therefore new system with new components has to be created.

2- Upgrading issues

It would be difficult with PIC because every time upgrading needs to be done, the stand-alone PIC needs to be pulled out from the circuit and burned again.

3- Complicated programming

Since most of control access system nowadays uses PIC, many engineers will face difficulties if problem occur on the device. The reason being is because PIC is not an open-source IDE compared to Arduino. Therefore if problem occur, more time is needed for the engineers to create new coding for the device.

1.8 Objective

1. To create a door access system using Arduino approach.

2. To test the accuracy of this device.

1.9 Scope Of Study

1. To understand about door access system.

2. To understand about keypad.

3. To understand about biometric fingerprint scanner.

4. To study about Arduino and its application.

1.10 Relevancy / Significant / Contribution Of The Project

The Door Access System Arduino Based is proposed mainly for Tricubes Computer Sdn. Bhd. This company produces product and services for Enterprise Mobility and Identity Authentication. The presence of the Door Access System will help the personnel of this to have access to the building. It is also designed to help the engineers in the future to perform maintenance and upgrading work easily.

1.11 Feasibility Of The Project Within The Scope & Time Frame

The Door Access System-Arduino Based is designed to be technically feasible. It is designed to be simpler compared to the existing door access system. Since most of buildings nowadays implement the use of door access system, this project is economically feasible as it can be marketed around the world. It is also feasible within the scope and time frame because Arduino is an open-source IDE which is suitable to be used within the time frame which leads to less time consumption.

CHAPTER 2

2.1 LITERATURE REVIEW

Door access system has been widely used around the world. It is a type of security system that is created to help in securing people and assets in a building from unwanted cases such as burglaries and kidnapping. To develop a door access system, features or hardware such as keypad, smart card, RFID card and biometric are implemented. Besides hardware, software is also included in developing the door access system as it helps in interfacing the hardware and to have the desired system flow. Types of software or programming language being used in this system are PIC language programming, Matlab, Microsoft Visual C++, Arduino and many more.

In this project, keypad and fingerprint scanner are implemented in developing the Door Access System – Arduino Based. These two features were chosen because of its user-friendly, smart and high security system compared to other features such as face-verification, smart card, RFID card and many more.

Fingerprint scanner is a type of biometric sensors whereby it senses the human fingerprint for identification. Biometric consist of many types such as voice-recognition, face-recognition, fingerprint-recognition and other identification that consist of human body parts.

Based on a journal written by *Wheeler et.al.* (2000) on face-verification system, this system is time-consuming to build because the users need to have more than one images to separate ID and non-ID images for identification data storage purposes. Despite of its advantages of identifying intra and interdependencies, it is proven that this

system is inefficient because it took 6 seconds to make decision while the aim is within 2 seconds .

The fact of time-consuming system has also been said in a journal written by *Ibrahim et. al.* (2011) on face-recognition system. Face-recognition is difficult to build as there are alot of factor needs to be considered during image capturing which are illumination, distance and an individuals head orientation. This system is also sensitive to aging and facial expression. It is also troublesome during experimental work as many faces need to be taken at nine different angles .

Another access control system project by using voice-recognition system done by *Rashid et. al.* (2008) is also a sensitive system as it will reject the voice input if there are background noise.

Cui et. al. (2009) agrees the fact of voice-recognition system being difficult. The reason being is because it needs to build up a speech model whereby the users have to pronounce the text according to the stated ones. Despite of its lacking in efficieny, this project done by Bo Cui and Tongze Xue has its advantages compared to other projects with similar feature by using a technology to filter low frequency disturbing.

The benefit of using fingerprint scanner was also said by *Zhu et. al*. (2011) in their journal that fingerprint-verification overcome the issue of losing ID card where in their project is car keys. Another advantage is that the ownership can never be passed to other people . An optical fingerprint scanner is implemented in this project compared to other biometric features is because it has more advantages as stated in Table 2.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Project Process Flow

The project process flow is illustrated as flow chart in Figure 3.1. Firstly the user will insert the password that has already been set by the system. If the entered password is correct, the user may proceed by entering their ID. Otherwise the user will be given three attempts to enter password. If wrong password is entered at the third attempt, alarm will be activated.

Next, each authorized user have a template of their own fingerprint saved in the memory. In order to enter the building, the users need to scan their fingers by placing their finger onto the fingerprint scanner.

During storing process for fingerprint scanner, the storing starts by enrolling the users ID. Once the users place their finger, the fingerprint scanner will capture the fingerprint. It will then extract the minutiae and store the output in memory.
Next the process continues by finding the match. For fingerprint, it uses the stored minutiae with the ones that was recently capture.

If the users minutiae matches their own minutiae which has already been stored before, the users may access the door as the magnetic door lock will be deactivated. Otherwise, the users have to enter their ID number again. The users will be given three attempts. If the users fail at the third attempt, alarm will be activated indicating the presence of intruders.

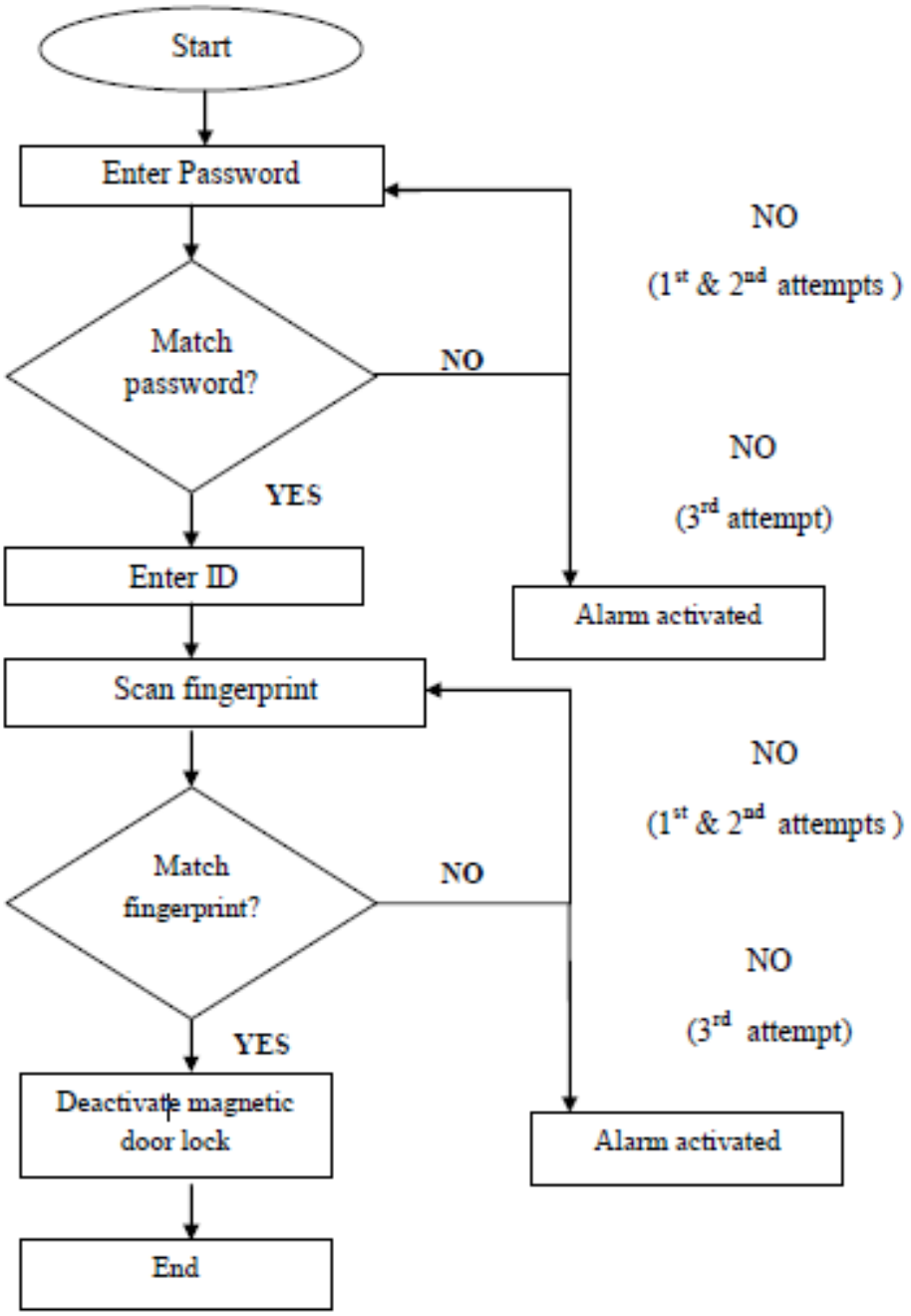

Figure:3.1 Project Process Flow

3.2 Design circuit and identify I/O pins to Arduino board

The circuit diagram (Appendix A) of this door access system is designed by using Fritzing software which is a type of designing tools that supports Arduino. The connections of the features to Arduino I/O pins are shown in Fig 3.2

		Arduino I/O pins
Keypad	1	23
	2	25
	3	27
	4	29
	5	31
	6	33
	7	35
	8	37
Fingerprint scanner	Red	5V
	Green	12
	White	13
	Black	GND
LCD display	1	GND
	2	5V
	3	150Ω
	4	40
	5	42
	6	44
	7	NC
	8	NC
	9	NC
	10	NC
	11	46
	12	48
	13	50
	14	52
	15	5V
	16	GND
Magnetic Switch	1 end	11
	1 end	GND
Siren	Positive	9
	Negative	GND
Indicator	Red	11
	Green	10

Figure 3.2: Connections to Arduino I/O pins

3.3 Software

3.3.1 Arduino programming

Arduino is an open-source microcontroller. It can be edited and loaded to the microcontroller by using a USB cable. Arduino is an inexpensive type of single-microprocessor prototyping platform. The software consists of standard programming language, which is similar to C/C++. To run the program, Arduino comes with a bootloader that runs on the board. The reason of implementing Arduino in this project is because it has many advantages compared to other microcontrollers such as PIC. It is comes with an ATMEGA2560 microcontroller whereby the program stored in ATMEGA2560 can be edited in the future for maintenance purposes. Figure 3.3.1 shows the development cycle of Arduino IDE whereby it consist of editing the coding, compile, upload and run.

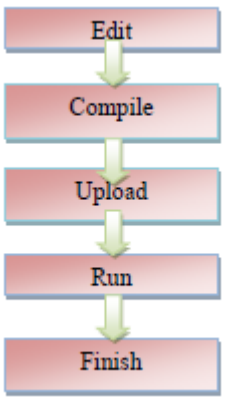

Figure:3.3.1 Arduino Programming

CHAPTER 4

RESULTS & DISCUSSIONS

According to project activities flow chart Figure 17 above, after constructing the circuit, testing was done. Figures 36 to 47 shows the results obtained from each feature and the interfacing all feature with Arduino. During testing, the outputs obtained were displayed through serial monitor. The LCD display is added afterwards to replace the serial monitor.

4.1 Fingerprint scanner

4.1.1 Enrolling

The function of fingerprint scanner is to scan the users fingerprint. During scanning the fingerprint scanner will capture and store the users minutiae. To have a database of users fingerprint minutiae, the user needs to enroll their fingerprint. Figure 4.1.1 shows the result of an enrolled fingerprint. Once Arduino detected fingerprint scanner, the users will insert the ID that will be saved together with their fingerprint minutiae. According to figure above, the users fingerprint minutiae is enrolled with the ID:2. It is then stored in the fingerprint scanner onboard flash memory

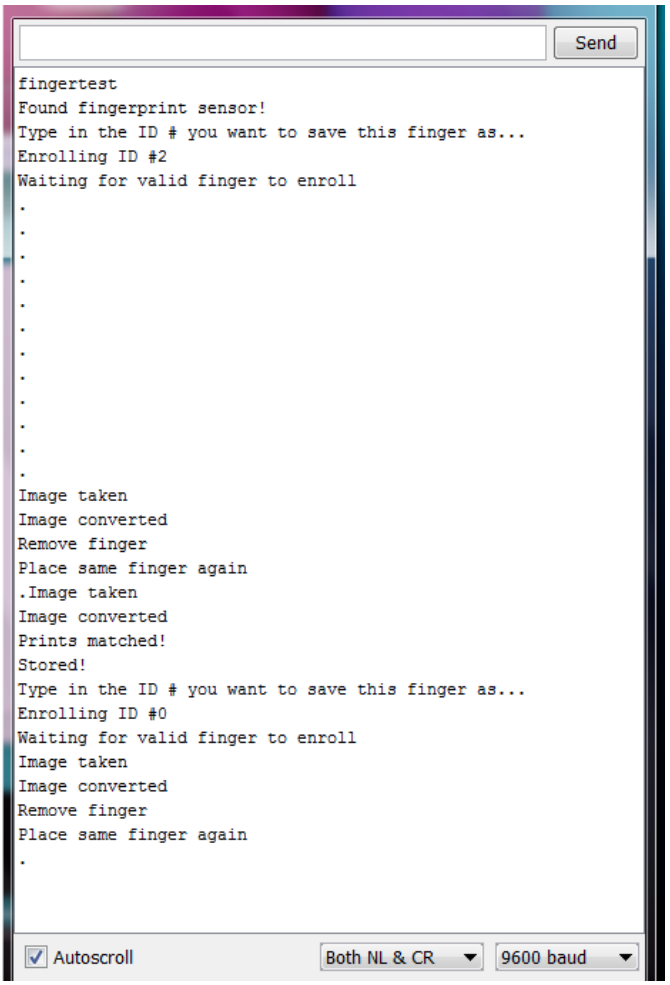

Figure:4.1.1 Fingerprint enrolling output at serial monitor

4.1.2 Finger test and matching

After enrolling the fingerprint, the stored minutiae need to be tested to test the accuracy of the fingerprint scanner. Figure 4.1.2 shows the finger test output at serial monitor. The user ID is found to be ID: 2. It states the confidence which actually

measures the accuracy of the current scanned fingerprint and the ones stored in memory. The finger print scanner has a level of confidence from 0 to 255 which indicates from less accurate to very accurate. The output indicates the current scanned fingerprint is matched with the stored minutiae with ID: 2.

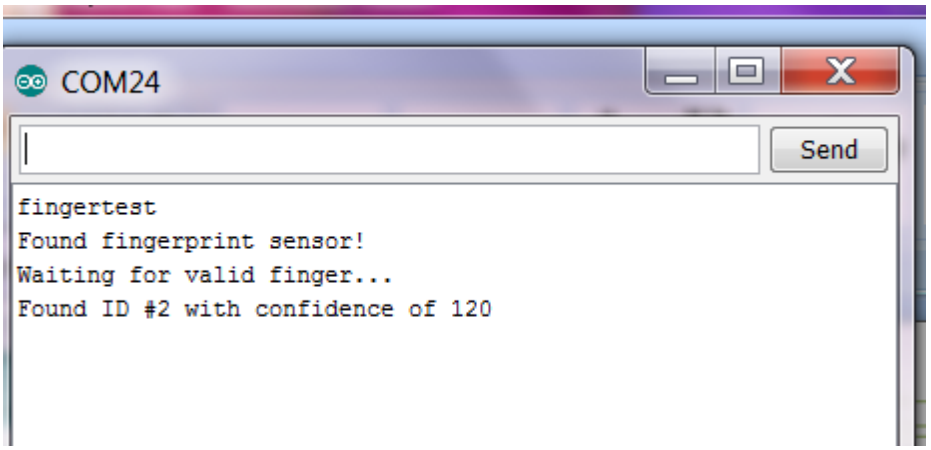

Figure:4.1.2 Finger test output at serial monitor

4.1.3 PROJECT CODE:

```
 his is a library for our optical Fingerprint sensor

  Designed specifically to work with the Adafruit Fingerprint sensor
  ----> http://www.adafruit.com/products/751

  These displays use TTL Serial to communicate, 2 pins are required to
  interface
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  BSD license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_Fingerprint.h"
#if defined(__AVR__) || defined(ESP8266)
   #include <SoftwareSerial.h>
#endif

//#define FINGERPRINT_DEBUG

#if ARDUINO >= 100
  #define SERIAL_WRITE(...) mySerial->write(__VA_ARGS__)
#else
  #define SERIAL_WRITE(...) mySerial->write(__VA_ARGS__, BYTE)
#endif

#define SERIAL_WRITE_U16(v) SERIAL_WRITE((uint8_t)(v>>8)); SERIAL_WRITE((uint8_t)(v & 0xFF));

#define GET_CMD_PACKET(...) \
  uint8_t data[] = {__VA_ARGS__}; \
  Adafruit_Fingerprint_Packet packet(FINGERPRINT_COMMANDPACKET, sizeof(data), data); \
  writeStructuredPacket(packet); \
  if (getStructuredPacket(&packet) != FINGERPRINT_OK) return FINGERPRINT_PACKETRECIEVEERR; \
  if (packet.type != FINGERPRINT_ACKPACKET) return FINGERPRINT_PACKETRECIEVEERR;

#define SEND_CMD_PACKET(...) GET_CMD_PACKET(__VA_ARGS__); return packet.data[0];

/***************************************************************************
 PUBLIC FUNCTIONS
 ***************************************************************************/


#if defined(__AVR__) || defined(ESP8266)
/**************************************************************************/
/*!
   @brief  Instantiates sensor with Software Serial
   @param  ss Pointer to SoftwareSerial object
   @param  password 32-bit integer password (default is 0)
*/
/**************************************************************************/
Adafruit_Fingerprint::Adafruit_Fingerprint(SoftwareSerial *ss, uint32_t password) {
  thePassword = password;
  theAddress = 0xFFFFFFFF;

  hwSerial = NULL;
  swSerial = ss;
```

```
  mySerial = swSerial;
}
#endif

/**************************************************************************/
/*!
    @brief  Instantiates sensor with Hardware Serial
    @param  hs Pointer to HardwareSerial object
    @param  password 32-bit integer password (default is 0)

*/
/**************************************************************************/
Adafruit_Fingerprint::Adafruit_Fingerprint(HardwareSerial *hs, uint32_t password) {
  thePassword = password;
  theAddress = 0xFFFFFFFF;

#if defined(__AVR__) || defined(ESP8266)
  swSerial = NULL;
#endif
  hwSerial = hs;
  mySerial = hwSerial;
}

/**************************************************************************/
/*!
    @brief  Initializes serial interface and baud rate
    @param  baudrate Sensor's UART baud rate (usually 57600, 9600 or 115200)
*/
/**************************************************************************/
void Adafruit_Fingerprint::begin(uint32_t baudrate) {
  delay(1000);  // one second delay to let the sensor 'boot up'

  if (hwSerial) hwSerial->begin(baudrate);
#if defined(__AVR__) || defined(ESP8266)
  if (swSerial) swSerial->begin(baudrate);
#endif
}

/**************************************************************************/
/*!
    @brief  Verifies the sensors' access password (default password is 0x0000000). A good way to also check if the
sensors is active and responding
    @returns True if password is correct
*/
/**************************************************************************/
boolean Adafruit_Fingerprint::verifyPassword(void) {
  return checkPassword() == FINGERPRINT_OK;
}

uint8_t Adafruit_Fingerprint::checkPassword(void) {
  GET_CMD_PACKET(FINGERPRINT_VERIFYPASSWORD,
                 (uint8_t)(thePassword >> 24), (uint8_t)(thePassword >> 16),
                 (uint8_t)(thePassword >> 8), (uint8_t)(thePassword & 0xFF));
  if (packet.data[0] == FINGERPRINT_OK)
    return FINGERPRINT_OK;
  else
    return FINGERPRINT_PACKETRECIEVEERR;
}

/**************************************************************************/
/*!
    @brief   Ask the sensor to take an image of the finger pressed on surface
    @returns <code>FINGERPRINT_OK</code> on success
```

```
   @returns <code>FINGERPRINT_NOFINGER</code> if no finger detected
   @returns <code>FINGERPRINT_PACKETRECIEVEERR</code> on communication error
   @returns <code>FINGERPRINT_IMAGEFAIL</code> on imaging error
*/
/**************************************************************************/
uint8_t Adafruit_Fingerprint::getImage(void) {
  SEND_CMD_PACKET(FINGERPRINT_GETIMAGE);
}

/**************************************************************************/
/*!
   @brief   Ask the sensor to convert image to feature template
   @param slot Location to place feature template (put one in 1 and another in 2 for verification to create model)
   @returns <code>FINGERPRINT_OK</code> on success
   @returns <code>FINGERPRINT_IMAGEMESS</code> if image is too messy
   @returns <code>FINGERPRINT_PACKETRECIEVEERR</code> on communication error
   @returns <code>FINGERPRINT_FEATUREFAIL</code> on failure to identify fingerprint features
   @returns <code>FINGERPRINT_INVALIDIMAGE</code> on failure to identify fingerprint features
*/
uint8_t Adafruit_Fingerprint::image2Tz(uint8_t slot) {
  SEND_CMD_PACKET(FINGERPRINT_IMAGE2TZ,slot);
}

/**************************************************************************/
/*!
   @brief   Ask the sensor to take two print feature template and create a model
   @returns <code>FINGERPRINT_OK</code> on success
   @returns <code>FINGERPRINT_PACKETRECIEVEERR</code> on communication error
   @returns <code>FINGERPRINT_ENROLLMISMATCH</code> on mismatch of fingerprints
*/
uint8_t Adafruit_Fingerprint::createModel(void) {
  SEND_CMD_PACKET(FINGERPRINT_REGMODEL);
}


/**************************************************************************/
/*!
   @brief   Ask the sensor to store the calculated model for later matching
   @param   location The model location #
   @returns <code>FINGERPRINT_OK</code> on success
   @returns <code>FINGERPRINT_BADLOCATION</code> if the location is invalid
   @returns <code>FINGERPRINT_FLASHERR</code> if the model couldn't be written to flash memory
   @returns <code>FINGERPRINT_PACKETRECIEVEERR</code> on communication error
*/
uint8_t Adafruit_Fingerprint::storeModel(uint16_t location) {
  SEND_CMD_PACKET(FINGERPRINT_STORE, 0x01, (uint8_t)(location >> 8), (uint8_t)(location & 0xFF));
}

/**************************************************************************/
/*!
   @brief   Ask the sensor to load a fingerprint model from flash into buffer 1
   @param   location The model location #
   @returns <code>FINGERPRINT_OK</code> on success
   @returns <code>FINGERPRINT_BADLOCATION</code> if the location is invalid
   @returns <code>FINGERPRINT_PACKETRECIEVEERR</code> on communication error
*/
uint8_t Adafruit_Fingerprint::loadModel(uint16_t location) {
  SEND_CMD_PACKET(FINGERPRINT_LOAD, 0x01, (uint8_t)(location >> 8), (uint8_t)(location & 0xFF));
}

/**************************************************************************/
/*!
   @brief   Ask the sensor to transfer 256-byte fingerprint template from the buffer to the UART
```

```
   @returns <code>FINGERPRINT_OK</code> on success
   @returns <code>FINGERPRINT_PACKETRECIEVEERR</code> on communication error
*/
uint8_t Adafruit_Fingerprint::getModel(void) {
  SEND_CMD_PACKET(FINGERPRINT_UPLOAD, 0x01);
}

/**************************************************************************/
/*!
   @brief   Ask the sensor to delete a model in memory
   @param   location The model location #
   @returns <code>FINGERPRINT_OK</code> on success
   @returns <code>FINGERPRINT_BADLOCATION</code> if the location is invalid
   @returns <code>FINGERPRINT_FLASHERR</code> if the model couldn't be written to flash memory
   @returns <code>FINGERPRINT_PACKETRECIEVEERR</code> on communication error
*/
uint8_t Adafruit_Fingerprint::deleteModel(uint16_t location) {
  SEND_CMD_PACKET(FINGERPRINT_DELETE, (uint8_t)(location >> 8), (uint8_t)(location & 0xFF), 0x00,
0x01);
}

/**************************************************************************/
/*!
   @brief   Ask the sensor to delete ALL models in memory
   @returns <code>FINGERPRINT_OK</code> on success
   @returns <code>FINGERPRINT_BADLOCATION</code> if the location is invalid
   @returns <code>FINGERPRINT_FLASHERR</code> if the model couldn't be written to flash memory
   @returns <code>FINGERPRINT_PACKETRECIEVEERR</code> on communication error
*/
uint8_t Adafruit_Fingerprint::emptyDatabase(void) {
  SEND_CMD_PACKET(FINGERPRINT_EMPTY);
}

/**************************************************************************/
/*!
   @brief   Ask the sensor to search the current slot 1 fingerprint features to match saved templates. The matching
location is stored in <b>fingerID</b> and the matching confidence in <b>confidence</b>
   @returns <code>FINGERPRINT_OK</code> on fingerprint match success
   @returns <code>FINGERPRINT_NOTFOUND</code> no match made
   @returns <code>FINGERPRINT_PACKETRECIEVEERR</code> on communication error
*/
/**************************************************************************/
uint8_t Adafruit_Fingerprint::fingerFastSearch(void) {
  // high speed search of slot #1 starting at page 0x0000 and page #0x00A3
  GET_CMD_PACKET(FINGERPRINT_HISPEEDSEARCH, 0x01, 0x00, 0x00, 0x00, 0xA3);
  fingerID = 0xFFFF;
  confidence = 0xFFFF;

  fingerID = packet.data[1];
  fingerID <<= 8;
  fingerID |= packet.data[2];

  confidence = packet.data[3];
  confidence <<= 8;
  confidence |= packet.data[4];

  return packet.data[0];
}

/**************************************************************************/
/*!
   @brief   Ask the sensor for the number of templates stored in memory. The number is stored in
<b>templateCount</b> on success.
```

```
   @returns <code>FINGERPRINT_OK</code> on success
   @returns <code>FINGERPRINT_PACKETRECIEVEERR</code> on communication error
*/
/**************************************************************************/
uint8_t Adafruit_Fingerprint::getTemplateCount(void) {
  GET_CMD_PACKET(FINGERPRINT_TEMPLATECOUNT);

  templateCount = packet.data[1];
  templateCount <<= 8;
  templateCount |= packet.data[2];

  return packet.data[0];
}

/**************************************************************************/
/*!
   @brief   Set the password on the sensor (future communication will require password verification so don't forget
it!!!)
   @param   password 32-bit password code
   @returns <code>FINGERPRINT_OK</code> on success
   @returns <code>FINGERPRINT_PACKETRECIEVEERR</code> on communication error
*/
/**************************************************************************/
uint8_t Adafruit_Fingerprint::setPassword(uint32_t password) {
  SEND_CMD_PACKET(FINGERPRINT_SETPASSWORD, (password >> 24), (password >> 16), (password >> 8),
password);
}

/**************************************************************************/
/*!
   @brief   Helper function to process a packet and send it over UART to the sensor
   @param   packet A structure containing the bytes to transmit
*/
/**************************************************************************/

void Adafruit_Fingerprint::writeStructuredPacket(const Adafruit_Fingerprint_Packet & packet) {
  SERIAL_WRITE_U16(packet.start_code);
  SERIAL_WRITE(packet.address[0]);
  SERIAL_WRITE(packet.address[1]);
  SERIAL_WRITE(packet.address[2]);
  SERIAL_WRITE(packet.address[3]);
  SERIAL_WRITE(packet.type);

  uint16_t wire_length = packet.length + 2;
  SERIAL_WRITE_U16(wire_length);

  uint16_t sum = ((wire_length)>>8) + ((wire_length)&0xFF) + packet.type;
  for (uint8_t i=0; i< packet.length; i++) {
    SERIAL_WRITE(packet.data[i]);
    sum += packet.data[i];
  }

  SERIAL_WRITE_U16(sum);
  return;
}

/**************************************************************************/
/*!
   @brief   Helper function to receive data over UART from the sensor and process it into a packet
   @param   packet A structure containing the bytes received
   @param   timeout how many milliseconds we're willing to wait
   @returns <code>FINGERPRINT_OK</code> on success
   @returns <code>FINGERPRINT_TIMEOUT</code> or <code>FINGERPRINT_BADPACKET</code> on
```

```
failure
*/
/**************************************************************************/
uint8_t Adafruit_Fingerprint::getStructuredPacket(Adafruit_Fingerprint_Packet * packet, uint16_t timeout) {
  uint8_t byte;
  uint16_t idx=0, timer=0;

  while(true) {
    while(!mySerial->available()) {
      delay(1);
      timer++;
      if( timer >= timeout) {
#ifdef FINGERPRINT_DEBUG
                                              Serial.println("Timed out");
#endif
                                              return FINGERPRINT_TIMEOUT;
      }
    }
    byte = mySerial->read();
#ifdef FINGERPRINT_DEBUG
    Serial.print("<- 0x"); Serial.println(byte, HEX);
#endif
    switch (idx) {
      case 0:
        if (byte != (FINGERPRINT_STARTCODE >> 8))
                                                continue;
        packet->start_code = (uint16_t)byte << 8;
        break;
      case 1:
        packet->start_code |= byte;
        if (packet->start_code != FINGERPRINT_STARTCODE)
    return FINGERPRINT_BADPACKET;
        break;
      case 2:
      case 3:
      case 4:
      case 5:
        packet->address[idx-2] = byte;
        break;
      case 6:
      packet->type = byte;
      break;
      case 7:
       packet->length = (uint16_t)byte << 8;
      break;
      case 8:
       packet->length |= byte;
        break;
      default:
        packet->data[idx-9] = byte;
        if((idx-8) == packet->length)
          return FINGERPRINT_OK;
        break;
    }
    idx++;
  }
  // Shouldn't get here so...
  return FINGERPRINT_BADPACKET;
```

CONCLUSION

Thus Door unlocking System using Arduino" is a modern successor of the conventional door locking system. The conclusion of the discussion of Lock using Arduino is the innovation created from the lock system with no more direct contact between the user and the lock. This system is very cost-effective and easy to install. In conclusion, it was discovered that the project performed according to specification and can be implemented. The use of the Arduino UNO microcontroller in this project allows for design simplicity, hence, the project can be achieved in lesser time compared to other techniques previously employed. we have tried to solve the security matters in door locks by bringing the conceptions of biometric fingerprint with the door lock. For that purpose, we are using fingerprint as a rare key to implement device so as to lock or unlock a door lock. This project is going to implemented in multiple applications and this concept is still in use in the banks but those systems are specially built for those organizations to secure the whole organization and the assured will be very high and other companies and houses.