FINGERPRINT BASED BIOMETRIC ATTENDANCE SYSTEM USING ATMEGA32 MICROCONTROLLER

Abstract

Introduction

Requirements

Design

Code explanation

Output

Abstract

Every organization, be an educational institute or organization public or private, it has to

maintain accurate records of attendance of students or staff for effective working of

organization. As time has shown, both private and public sectors face high level of

mismanagement due to false records and impersonations. Employers and officials are

concerned over employee absenteeism in their human resources and the problems in

maintaining records of student attendance during lecture periods. Finger prints are the

patterns of ridges and valleys on everyone’s finger tips. They can be used as a type of

biometric identification which, like all personal features, are unique to a person and do not

change in the person’s lifetime. An attendance system based on fingerprint technology,

suitable to be used in a university environment, is presented in this thesis. The finger print

based attendance system was implemented with .atmega328 Microcontroller, R-305

fingerprint sensor and programmed in C++. It comprises of two

processes; enrollment and authentication. During enrollment, the finger print of the user is

captured and its unique defining features extracted to be stored in the flash memory along

with the other users identities as a template. Minutiae points, the unique feature, are extracted

using a method which extracts the ridge’s endings and bifurcation. During authentication, the

user’s fingerprint is scanned to be stored in one of the two buffers and the extracted features

compared with the template which is loaded to the other buffer to compare the match before

attendance is verified. The experimental result exhibits that the system developed is highly

efficient in verifying fingerprint with a high level of accuracy.

INTRODUCTION

1.1 Problem Statement

Designing and implementing a student attendance verification system based on finger print

recognition and is capable to manages records for attendance in classrooms, laboratories

seminars etc. in institute like NIT Rourkela.

Fingerprint Based Biometric Attendance System using AVR

According to researchers from Pen state University, humans are more likely to trust machines over people, which is likely evident from us revealing our ATM pin to a machine so easily. Today, in the world where AI, Machine learning, Chat bots, Smart Speakers, Robots etc are actively progressing, this synergy between humans and robots is only set to increase. Today, from bridge toll collectors to check-out cashiers everything around us is being replaced by machines to get the work done easier and more efficient. To keep up with the phase, in this project we will build a Bio-metric Attendance system using AVR microcontrollers to replace the manual procedure of taking attendance. This system will be more reliable and efficient since it would save time and avoid dodgers.

Fingerprint attendance systems are already readily available directly from the market, but what is more fun than building one? We have also built a wide variety of Attendance Systems earlier from a simple RFID based Attendance system to an IoT based biometric Attendance system using Arduino and Raspberry Pi. In this project, we have used fingerprint Module and AVR(atmega32) to register attendance. By using fingerprint sensor, the system will become more secure for the users. Following sections explains technical details of making a fingerprint-based Biometric Attendance System using AVR

REQUIRED COMPONENTS

Atmega32 -1

Fingerprint module (r305) -1

Push Button or membrane buttons - 4

LEDs -2

1K Resistor -2

2.2K resistor -1

Power 12v adaptor

Connecting wires

Buzzer -1

16x2 LCD -1

PCB or Bread Board

RTC Module (ds1307 or ds3231)-1

LM7805 -1

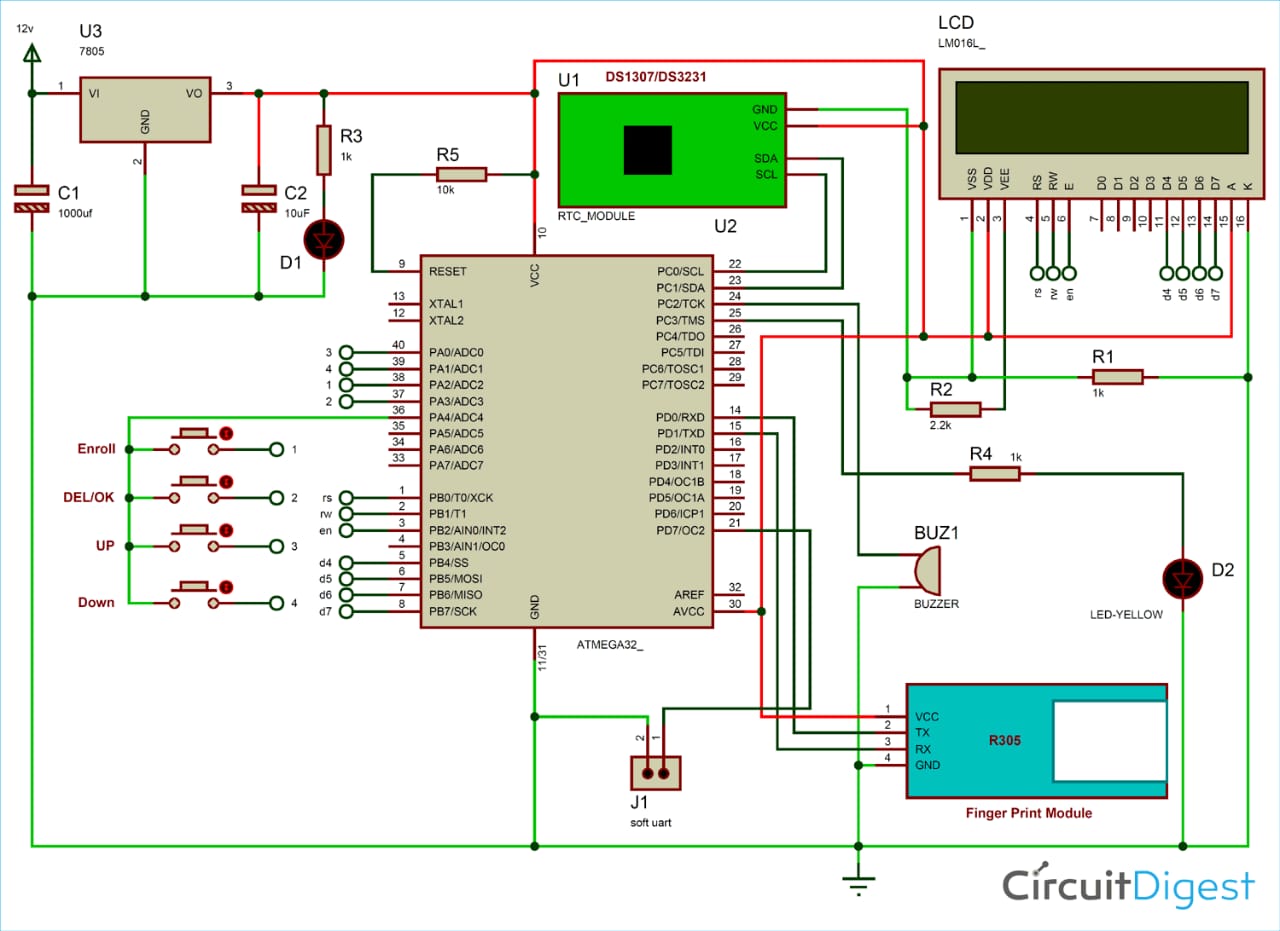
1000uf, 10uf capacitor -1

Burgstips male female

DC JACK (optional)

BC547 Transistor -1

DESIGN



As shown in the circuit diagram, push or membrane buttons are directly connected to pin PA2 (ENROL key 1), PA3(DEL key 2), PA0(UP key 3), PA1(DOWN key 4) of microcontroller with respect to the ground or PA4. And a LED is connected at pin PC2 of microcontroller with respect to ground through a 1k resistor. Fingerprint module’s Rx and Tx directly connected at Serial pin PD1 and PD3 of microcontroller. 5v supply is used for powering the whole circuit by using LM7805 voltage regulator which is powered by 12v dc adaptor. A buzzer is also connected at pin PC3. A 16x2 LCD is configured in 4-bit mode and its RS, RW, EN, D4, D5, D6, and D7 are directly connected at pin PB0, PB1, PB2, PB4, PB5, PB6, PB7 of microcontroller. RTC module is connected at I2Cpin PC0 SCL and PC1 SDA. And PD7 is used as soft UART Tx pin for getting the current time.

CODE EXPLANATION

define F\_CPU 8000000ul

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

/\*\*MACROS/

#define USART\_BAUDRATE 9600

#define BAUD\_PRESCALE (((F\_CPU / (USART\_BAUDRATE \* 16UL))) - 1)

#define uchar unsigned char

#define uint unsigned int

#define LCDPORTDIR DDRB

#define LCDPORT PORTB

#define rs 0

#define rw 1

#define en 2

#define RSLow (LCDPORT&=~(1<<rs))

#define RSHigh (LCDPORT|=(1<<rs))

#define RWLow (LCDPORT&=~(1<<rw))

#define ENLow (LCDPORT&=~(1<<en))

#define ENHigh (LCDPORT|=(1<<en))

#define KeyPORTdir DDRA

#define key PINA

#define KeyPORT PORTA

After this, we have declared some variables and arrays for fingerprint command and response. We have also added some functions for fetching and setting data to RTC.

void RTC\_stp()

{

TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWSTO); //stop communication

}

void RTC\_read()

{

TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN);

while((TWCR&0x80)==0x00);

TWDR=0xD0; //RTC write (slave address)

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

TWDR=0x00; //RTC write (word address)

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN); //start RTC communication again

while ((TWCR&0x80)==0x00);

TWDR=0xD1; // RTC command to read

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

Then we have some functions for LCD which are responsible to drive the LCD. LCD driver function is written for 4-bit mode drive. Followed by that we also have some UART driver functions which are responsible for initializing UART and exchanging data between fingerprint sensor and microcontroller.

void serialbegin()

{

UCSRC = (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1);

UBRRH = (BAUD\_PRESCALE >> 8);

UBRRL = BAUD\_PRESCALE;

UCSRB=(1<<RXEN)|(1<<TXEN)|(1<<RXCIE);

sei();

}

ISR(USART\_RXC\_vect)

{

char ch=UDR;

buf[ind++]=ch;

if(ind>0)

flag=1;

//serial1Write(ch);

}

void serialwrite(char ch)

{

while ((UCSRA & (1 << UDRE)) == 0);

UDR = ch;

}

void serialprint(char \*str)

{

while(\*str)

{

serialwrite(\*str++);

}

}

Now we have some more UART function but they are software UART. It is used for transferring saved data to the computer via serial terminal. These functions are delay-based and don’t use any type of interrupt. And for UART only tx signal will work and we have hardcoded baud rate for soft UART as 9600.

void SerialSoftWrite(char ch)

{

PORTD&=~(1<<7);

\_delay\_us(104);

for(int i=0;i<8;i++)

{

if(ch & 1)

PORTD|=(1<<7);

else

PORTD&=~(1<<7);

\_delay\_us(104);

ch>>=1;

}

PORTD|=(1<<7);

\_delay\_us(104);

}

void SerialSoftPrint(char \*str)

{

while(\*str)

{

SerialSoftWrite(\*str);

str++;

}

}

Followed by that we have functions that are responsible for displaying the RTC time in the LCD. The below given functions are used for writing attendance data to EEPROM and reading attendance data from EEPROM.

int eeprom\_write(unsigned int add,unsigned char data)

{

while(EECR&(1<<EEWE));

EEAR=add;

EEDR=data;

EECR|=(1<<EEMWE);

EECR|=(1<<EEWE);

return 0;

}

char eeprom\_read(unsigned int add)

{

while(EECR & (1<<EEWE));

EEAR=add;

EECR|=(1<<EERE);

return EEDR;

}

Code

#define F\_CPU 8000000ul

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

/\*\*MACROS/

#define USART\_BAUDRATE 9600

#define BAUD\_PRESCALE (((F\_CPU / (USART\_BAUDRATE \* 16UL))) - 1)

#define uchar unsigned char

#define uint unsigned int

#define LCDPORTDIR DDRB

#define LCDPORT PORTB

#define rs 0

#define rw 1

#define en 2

#define RSLow (LCDPORT&=~(1<<rs))

#define RSHigh (LCDPORT|=(1<<rs))

#define RWLow (LCDPORT&=~(1<<rw))

#define ENLow (LCDPORT&=~(1<<en))

#define ENHigh (LCDPORT|=(1<<en))

#define KeyPORTdir DDRA

#define key PINA

#define KeyPORT PORTA

#define OK 3

#define UP 0

#define DOWN 1

#define DEL 3

#define MATCH 1

#define ENROL 2

#define enrol (key & (1<<ENROL)) // key 1

#define match (key & (1<<MATCH)) // key 4

#define delet (key & (1<<DEL)) // key 2

#define up (key & (1<<UP)) // key 3

#define down (key & (1<<DOWN)) // key 4

#define ok (key & (1<<OK)) // key 2

#define LEDdir DDRC

#define LEDPort PORTC

#define LED 3

#define BUZ 2

#define LEDHigh (LEDPort += (1<<LED))

#define LEDLow (LEDPort &= ~(1<<LED))

#define BUZHigh (LEDPort += (1<<BUZ))

#define BUZLow (LEDPort &= ~(1<<BUZ))

#define HIGH 1

#define LOW 0

#define PASS 0

#define ERROR 1

#define check(id) id=up<down?++id:down<up?--id:id;

#define maxId 5

#define dataLenth 6

#define eepStartAdd 10

/variable/

uchar buf[20];

uchar buf1[20];

volatile uint ind;

volatile uint flag;

uint msCount=0;

uint g\_timerflag=1;

volatile uint count=0;

uchar data[10];

uint id=1;

int s,a,b,c;

const char passPack[]={0xEF, 0x1, 0xFF, 0xFF, 0xFF, 0xFF, 0x1, 0x0, 0x7, 0x13, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1B};

const char f\_detect[]={0xEF, 0x1, 0xFF, 0xFF, 0xFF, 0xFF, 0x1, 0x0, 0x3, 0x1, 0x0, 0x5};

const char f\_imz2ch1[]={0xEF, 0x1, 0xFF, 0xFF, 0xFF, 0xFF, 0x1, 0x0, 0x4, 0x2, 0x1, 0x0, 0x8};

const char f\_imz2ch2[]={0xEF, 0x1, 0xFF, 0xFF, 0xFF, 0xFF, 0x1, 0x0, 0x4, 0x2, 0x2, 0x0, 0x9};

const char f\_createModel[]={0xEF,0x1,0xFF,0xFF,0xFF,0xFF,0x1,0x0,0x3,0x5,0x0,0x9};

char f\_storeModel[]={0xEF,0x1,0xFF,0xFF,0xFF,0xFF,0x1,0x0,0x6,0x6,0x1,0x0,0x1,0x0,0xE};

const char f\_search[]={0xEF, 0x1, 0xFF, 0xFF, 0xFF, 0xFF, 0x1, 0x0, 0x8, 0x1B, 0x1, 0x0, 0x0, 0x0, 0xA3, 0x0, 0xC8};

char f\_delete[]={0xEF,0x1,0xFF,0xFF,0xFF,0xFF,0x1,0x0,0x7,0xC,0x0,0x0,0x0,0x1,0x0,0x15};

//const char f\_readNotepad[]={0xEF,0x1,0xFF,0xFF,0xFF,0xFF,0x1,0x0,0x4,0x19,0x0,0x0,0x1E};

//char f\_writeNotepad[]={0xEF,0x1,0xFF,0xFF,0xFF,0xFF,0x1,0x0,0x24};

int timeStamp[7],day;

enum

{

CMD=0,

DATA,

};

void buzzer(uint);

void lcdwrite(char ch,char r)

{

LCDPORT=ch & 0xF0;

RWLow;

if(r == 1)

RSHigh;

else

RSLow;

ENHigh;

\_delay\_ms(5);

ENLow;

\_delay\_ms(10);

LCDPORT=ch<<4 & 0xF0;

RWLow;

if(r == 1)

RSHigh;

else

RSLow;

ENHigh;

\_delay\_ms(5);

ENLow;

\_delay\_ms(10);

}

void lcdprint(char \*str)

{

while(\*str)

{

lcdwrite(\*str++,DATA);

//\_\_delay\_ms(20);

}

}

void lcdbegin()

{

uchar lcdcmd[5]={0x02,0x28,0x0E,0x06,0x01};

uint i=0;

for(i=0;i<5;i++)

lcdwrite(lcdcmd[i], CMD);

}

void serialbegin()

{

UCSRC = (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1);

UBRRH = (BAUD\_PRESCALE >> 8);

UBRRL = BAUD\_PRESCALE;

UCSRB=(1<<RXEN)|(1<<TXEN)|(1<<RXCIE);

sei();

}

ISR(USART\_RXC\_vect)

{

char ch=UDR;

buf[ind++]=ch;

if(ind>0)

flag=1;

[11:52 AM, 4/24/2022] Sindhu M T: serial1Write(ch);

}

void serialwrite(char ch)

{

while ((UCSRA & (1 << UDRE)) == 0);

UDR = ch;

}

void serialprint(char \*str)

{

while(\*str)

{

serialwrite(\*str++);

}

}

void serialprintln(char \*str)

{

serialprint(str);

serialwrite(0x0d);

serialwrite(0x0a);

}

void serialFlush()

{

for(int i=0;i<sizeof(buf);i++)

{

buf[i]=0;

}

}

void SerialSoftWrite(char ch)

{

PORTD&=~(1<<7);

\_delay\_us(104);

for(int i=0;i<8;i++)

{

if(ch & 1)

PORTD|=(1<<7);

else

PORTD&=~(1<<7);

\_delay\_us(104);

ch>>=1;

}

PORTD|=(1<<7);

\_delay\_us(104);

}

void SerialSoftPrint(char \*str)

{

while(\*str)

{

SerialSoftWrite(\*str);

str++;

}

}

void SerialSoftPrintln(char \*str)

{

SerialSoftPrint(str);

SerialSoftWrite(0x0D);

SerialSoftWrite(0x0A);

}

int bcdtochar(char num)

{

return ((num/16 \* 10) + (num % 16));

}

void RTC\_start()

{

TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN);

while((TWCR&0x80)==0x00);

}

void RTC\_stp()

{

TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWSTO); //stop communication

}

void RTC\_read()

{

TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN);

while((TWCR&0x80)==0x00);

TWDR=0xD0; //RTC write (slave address)

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

TWDR=0x00; //RTC write (word address)

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN); //start RTC communication again

while ((TWCR&0x80)==0x00);

TWDR=0xD1; // RTC command to read

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void sec\_init(unsigned char d)

{

TWDR=d; //second init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void min\_init(unsigned char d)

{

TWDR=d; //minute init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void hr\_init(unsigned char d)

{

TWDR=d; //hour init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void day\_init(unsigned char d)

{

TWDR=d; //days init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void date\_init(unsigned char d)

{

TWDR=d; //date init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void month\_init(unsigned char d)

{

TWDR=d; //month init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void yr\_init(unsigned char d)

{

TWDR=d; //year init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

int sec\_rw()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC second read

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int min\_rw()

{

TWCR|=(1<<TWINT); //RTC minute read

TWCR|=(1<<TWEA);

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}[11:59 AM, 4/24/2022] Sindhu M T: void yr\_init(unsigned char d)

{

TWDR=d; //year init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

int sec\_rw()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC second read

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int min\_rw()

{

TWCR|=(1<<TWINT); //RTC minute read

TWCR|=(1<<TWEA);

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int hr\_rw()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC hour read

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int day\_rd()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC day read

while((TWCR&0x80)==0x00);

return bcdtochar(TWDR);

}

int date\_rw()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC date read

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int month\_rw()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC month read

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int yr\_rw()

{

TWCR|=(1<<TWINT); //RTC year read

TWCR&=(~(1<<TWEA));

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

void device()

{

TWDR=0xD0; //RTC write (slave address)

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

TWDR=0x00; // word address write

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void RTCTimeSet()

{

RTC\_start();

device();

sec\_init(0);

min\_init(0x47);

hr\_init(0x22);

day\_init(0x03);

date\_init(0x23);

month\_init(0x08);

yr\_init(0x19);

RTC\_stp();

}

void show()

{

char tem[20];

sprintf(tem,"%d",timeStamp[0]);

lcdwrite(0x80,CMD);

lcdprint("Time:");

lcdprint(tem);

lcdwrite(':',DATA);

sprintf(tem,"%d",timeStamp[1]);

lcdprint(tem);

lcdwrite(':',DATA);

sprintf(tem,"%d",timeStamp[2]);

lcdprint(tem);

lcdprint(" ");

lcdwrite(0xc0,CMD);

lcdprint("Date:");

sprintf(tem,"%d",timeStamp[3]);

lcdprint(tem);

lcdwrite('/',DATA);

sprintf(tem,"%d",timeStamp[4]);

lcdprint(tem);

lcdwrite('/',DATA);

sprintf(tem,"%d",timeStamp[5]);

lcdprint("20");

if(timeStamp[5]<10)

lcdwrite('0',DATA);

lcdprint(tem);

lcdprint(" ");

}

void RTC()

{

RTC\_read();

timeStamp[2]=sec\_rw();

timeStamp[1]=min\_rw();

timeStamp[0]=hr\_rw();

day=day\_rd();

timeStamp[3]=date\_rw();

timeStamp[4]=month\_rw();

timeStamp[5]=yr\_rw();

RTC\_stp();

show();

}

int eeprom\_write(unsigned int add,unsigned char data)

{

while(EECR&(1<<EEWE));

EEAR=add;

EEDR=data;

EECR|=(1<<EEMWE);

EECR|=(1<<EEWE);

return 0;

}

char eeprom\_read(unsigned int add)

{

while(EECR & (1<<EEWE));

EEAR=add;

EECR|=(1<<EERE);

return EEDR;

}

void saveData(int id)

{

uint cIndex= eeprom\_read(id);

if(cIndex == 0)

cIndex=1;

uint cAddress= (cIndex\*6) + (id-1)\*48;

for(int i=0;i<6;i++)

eeprom\_write(cAddress+i,timeStamp[i]);

eeprom\_write(id,cIndex+1);

}

int sendcmd2fp(char \*pack, int len)

{

int res=ERROR;

serialFlush();

ind=0;

\_delay\_ms(100);

for(int i=0;i<len;i++)

{

serialwrite(\*(pack+i));

}

\_delay\_ms(1000);

if(flag == 1)

{

if(buf[0] == 0xEF && buf[1] == 0x01)

{

if(buf[6] == 0x07) // ack

{

if(buf[9] == 0)

{

uint data\_len= buf[7];

data\_len<<=8;

data\_len|=buf[8];

for(int i=0;i<data\_len;i++)

data[i]=0;

//data=(char \*)calloc(data\_len, sizeof(data));

for(int i=0;i<data\_len-2;i++)

{

data[i]=buf[10+i];

}

res=PASS;

}

else

{

res=ERROR;

}

}

}

ind=0;

flag=0;

return res;

}

return res;

}

uint getId()

{

uint id=0;

lcdwrite(1, CMD);

while(1)

{

//check(id);

if(up == LOW)

{

id++;

buzzer(200);

}

else if(down == LOW)

{

id--;

if(id==0)

id=0;

buzzer(200);

}

else if(ok == LOW)

{

buzzer(200);

return id;

}

lcdwrite(0x80, CMD);

(void)sprintf((char \*)buf1,"Enter Id:%d ",id);

lcdprint((char \*)buf1);

\_delay\_ms(200);

}

}

void matchFinger()

{

// lcdwrite(1,CMD);

// lcdprint("Place Finger");

// lcdwrite(192,CMD);

// \_delay\_ms(2000);

if(!sendcmd2fp((char \*)&f\_detect[0],sizeof(f\_detect)))

{

if(!sendcmd2fp((char \*)&f\_imz2ch1[0],sizeof(f\_imz2ch1)))

{

if(!sendcmd2fp((char \*)&f\_search[0],sizeof(f\_search)))

{

LEDHigh;

buzzer(200);

uint id= data[0];

id<<=8;

id+=data[1];

uint score=data[2];

score<<=8;

score+=data[3];

(void)sprintf((char \*)buf1,"Id: %d",(int)id);

lcdwrite(1,CMD);

lcdprint((char \*)buf1);

saveData(id);

\_delay\_ms(1000);

lcdwrite(1,CMD);

lcdprint("Attendance");

lcdwrite(192,CMD);

lcdprint("Registered");

\_delay\_ms(2000);

LEDLow;

}

else

{

LEDHigh;

lcdwrite(1,CMD);

lcdprint("Not Found");

buzzer(5000);

LEDLow;

}

}

else

{

LEDHigh;

lcdwrite(1,CMD);

lcdprint("Not Found");

buzzer(2000);

LEDLow;

}

}

else

{

//lcdprint("No Finger");

}

//\_delay\_ms(200);

}

void enrolFinger()

{

lcdwrite(1,CMD);

lcdprint("Enroll Finger");

\_delay\_ms(2000);

lcdwrite(1,CMD);

lcdprint("Place Finger");

lcdwrite(192,CMD);

\_delay\_ms(1000);

for(int i=0;i<3;i++)

{

if(!sendcmd2fp((char \*)&f\_detect[0],sizeof(f\_detect)))

{

//lcdprint("Finger Detected");

//\_\_delay\_ms(1000);

if(!sendcmd2fp((char \*)&f\_imz2ch1[0],sizeof(f\_imz2ch1)))

{

lcdwrite(192,CMD);

lcdprint("Finger Detected");

\_delay\_ms(1000);

// lcdwrite(1,CMD);

// lcdprint("Tamplate 1");

// \_\_delay\_ms(1000);

lcdwrite(1,CMD);

lcdprint("Place Finger");

lcdwrite(192,CMD);

lcdprint(" Again ");

\_delay\_ms(2000);

if(!sendcmd2fp((char \*)&f\_detect[0],sizeof(f\_detect)))

{

if(!sendcmd2fp((char \*)&f\_imz2ch2[0],sizeof(f\_imz2ch2)))

{

lcdwrite(1,CMD);

lcdprint("Finger Detected");

\_delay\_ms(1000);

if(!sendcmd2fp((char \*)&f\_createModel[0],sizeof(f\_createModel)))

{

id=getId();

f\_storeModel[11]= (id>>8) & 0xff;

f\_storeModel[12]= id & 0xff;

f\_storeModel[14]= 14+id;

if(!sendcmd2fp((char \*)&f\_storeModel[0],sizeof(f\_storeModel)))

{

buzzer(200);

lcdwrite(1,CMD);

lcdprint("Finger Stored");

(void)sprintf((char \*)buf1,"Id:%d",(int)id);

lcdwrite(192,CMD);

lcdprint((char \*)buf1);

\_delay\_ms(1000);

}

else

{

lcdwrite(1,CMD);

lcdprint("Finger Not Stored");

buzzer(3000);

}

}

else

lcdprint("Error");

}

else

lcdprint("Error");

}

else

i=2;

}

break;

}

if(i==2)

{

lcdwrite(0xc0,CMD);

lcdprint("No Finger");

}

}

\_delay\_ms(2000);

}

void deleteFinger()

{

id=getId();

f\_delete[10]=id>>8 & 0xff;

f\_delete[11]=id & 0xff;

f\_delete[14]=(21+id)>>8 & 0xff;

f\_delete[15]=(21+id) & 0xff;

if(!sendcmd2fp(&f\_delete[0],sizeof(f\_delete)))

{

lcdwrite(1,CMD);

sprintf((char \*)buf1,"Finger ID %d ",id);

lcdprint((char \*)buf1);

lcdwrite(192, CMD);

lcdprint("Deleted Success");

}

else

{

lcdwrite(1,CMD);

lcdprint("Error");

}

\_delay\_ms(2000);

}

void lcdinst()

{

lcdwrite(0x80, CMD);

lcdprint("1-Enroll Finger");

lcdwrite(0xc0, CMD);

lcdprint("2-delete Finger");

\_delay\_ms(10);

}

void buzzer(uint t)

{

BUZHigh;

for(int i=0;i<t;i++)

\_delay\_ms(1);

BUZLow;

}

/function to show attendence data on serial moinitor using softserial pin PD7/

void ShowAttendance()

{

char buf[128];

lcdwrite(1,CMD);

lcdprint("Downloding....");

SerialSoftPrintln("Attendance Record");

SerialSoftPrintln(" ");

SerialSoftPrintln("S.No ID1 ID2 Id3 ID4 ID5 ");

//serialprintln("Attendance Record");

//serialprintln(" ");

//serialprintln("S.No ID1 ID2 Id3 ID4 ID5");

for(int cIndex=1;cIndex<=8;cIndex++)

{

sprintf((char \*)buf,"%d "

"%d:%d:%d %d/%d/20%d "

"%d:%d:%d %d/%d/20%d "

"%d:%d:%d %d/%d/20%d "

"%d:%d:%d %d/%d/20%d "

"%d:%d:%d %d/%d/20%d ",

cIndex,

eeprom\_read((cIndex\*6)),eeprom\_read((cIndex\*6)+1),eeprom\_read((cIndex\*6)+2),eeprom\_read((cIndex\*6)+3),

eeprom\_read((cIndex\*6)+4),eeprom\_read((cIndex\*6)+5),

eeprom\_read((cIndex\*6)+48),eeprom\_read((cIndex\*6)+1+48),eeprom\_read((cIndex\*6)+2+48),

eeprom\_read((cIndex\*6)+3+48),eeprom\_read((cIndex\*6)+4+48),eeprom\_read((cIndex\*6)+5+48),

eeprom\_read((cIndex\*6)+96),eeprom\_read((cIndex\*6)+1+96),eeprom\_read((cIndex\*6)+2+96),

eeprom\_read((cIndex\*6)+3+96),eeprom\_read((cIndex\*6)+4+96),eeprom\_read((cIndex\*6)+5+96),

eeprom\_read((cIndex\*6)+144),eeprom\_read((cIndex\*6)+1+144),eeprom\_read((cIndex\*6)+2+144),

eeprom\_read((cIndex\*6)+3+144),eeprom\_read((cIndex\*6)+4+144),eeprom\_read((cIndex\*6)+5+144),

eeprom\_read((cIndex\*6)+192),eeprom\_read((cIndex\*6)+1+192),eeprom\_read((cIndex\*6)+2+192),

eeprom\_read((cIndex\*6)+3+192),eeprom\_read((cIndex\*6)+4+192),eeprom\_read((cIndex\*6)+5+192));

SerialSoftPrintln(buf);

//serialprintln(buf);

}

lcdwrite(192,CMD);

lcdprint("Done");

\_delay\_ms(2000);

}

void DeleteRecord()

{

lcdwrite(1,CMD);

lcdprint("Please Wait...");

for(int i=0;i<255;i++)

eeprom\_write(i,10);

\_delay\_ms(2000);

lcdwrite(1,CMD);

lcdprint("Record Deleted");

lcdwrite(192,CMD);

lcdprint("Successfully");

\_delay\_ms(2000);

}

int main()

{

LEDdir= 0xFF;

LEDPort=0x03;

KeyPORTdir=0xF0;

KeyPORT=0x0F;

LCDPORTDIR=0xFF;

DDRD+=1<<7;

PORTD+=1<<7;

serialbegin();

SerialSoftPrint("Circuit Digest");

//serialprint("Saddam Khan");

buzzer(2000);

lcdbegin();

lcdprint("Attendance Systm");

lcdwrite(192,CMD);

lcdprint("Using AVR and FP");

\_delay\_ms(2000);

if(down == LOW)

ShowAttendance();

else if(delet == LOW)

DeleteRecord();

ind=0;

while(sendcmd2fp((char \*)&passPack[0],sizeof(passPack)))

{

lcdwrite(1,CMD);

lcdprint("FP Not Found");

\_delay\_ms(2000);

ind=0;

}

lcdwrite(1,CMD);

lcdprint("FP Found");

\_delay\_ms(1000);

lcdinst();

\_delay\_ms(2000);

lcdwrite(1,CMD);

//RTCTimeSet();

while(1)

{

RTC();

// if(match == LOW)

// {

matchFinger();

// }

if(enrol == LOW)

{

buzzer(200);

enrolFinger();

\_delay\_ms(2000);

// lcdinst();

}

else if(delet == LOW)

{

buzzer(200);

getId();

deleteFinger();

\_delay\_ms(1000);

}

}

return 0;

}