Embedded Lock System Documentation

Table of Contents

Overview

The Embedded Lock System is a collaborative project designed to implement a secure lock system. The system utilizes Proteus 8 Professional for simulation and CodeVisionAVR Evaluation for programming the ATmega16 microcontroller. Written in the C programming language, the system encompasses three main functionalities: opening the door, setting a new passcode (PC), and accessing administrative features. The project is organized into three distinct parts, with each part expertly handled by different contributors.

Features

- Password-based Access Control
- LCD Display for User Interaction
- Audible Alarms for Incorrect Entries

Define interrupts priorities:

(Recommended):

- Press the Open button to open the door, triggering button '*'.
- Prioritize the Admin button by associating it with interrupt INT0.

- Set the PC configuration with the Set PC button, utilizing interrupt INT1.

> The project favors this Option because it assigns higher priority to the Admin button, followed logically by the Set PC button, and then the Open button.

Getting Started

Prerequisites

Ensure you have the following tools and components:

- Proteus 8 Professional
- CodeVisionAVR Evaluation
- ATmega16 Microcontroller
- Other necessary components (LCD, DC Motor, Buzzer, Keypad)

Installation

1. Clone the repository:

```
git clone https://github.com/Hussein119/lock-system.git
cd lock-system
```

2. Open the project in CodeVisionAVR.
    - Launch CodeVisionAVR and open the project file (`\Code\Project #1 lock system.prj`).
    - Customize project settings if necessary.
3. Simulate in Proteus.
    - Open Proteus 8 Professional.
    - Load the simulation file (`\Simulation\Project #1 lock system.pdsprj`) and run the simulation.
4. Hardware Implementation.
    - Connect the ATmega16 to the necessary components.
    - Program the microcontroller using CodeVisionAVR.

Usage

Test the lock system with the predefined password, verify LED indicators, and explore other functionalities.

Proteus Simulation

Hardware Components

1. ATmega16 Microcontroller
2. LCD Display
3. Keypad 4x3
4. Red light (for door simulation)
5. Motor (for door simulation)
6. Speaker (Peeps alarm)
7. Keypad 4x3
8. Three Buttons for interrupts

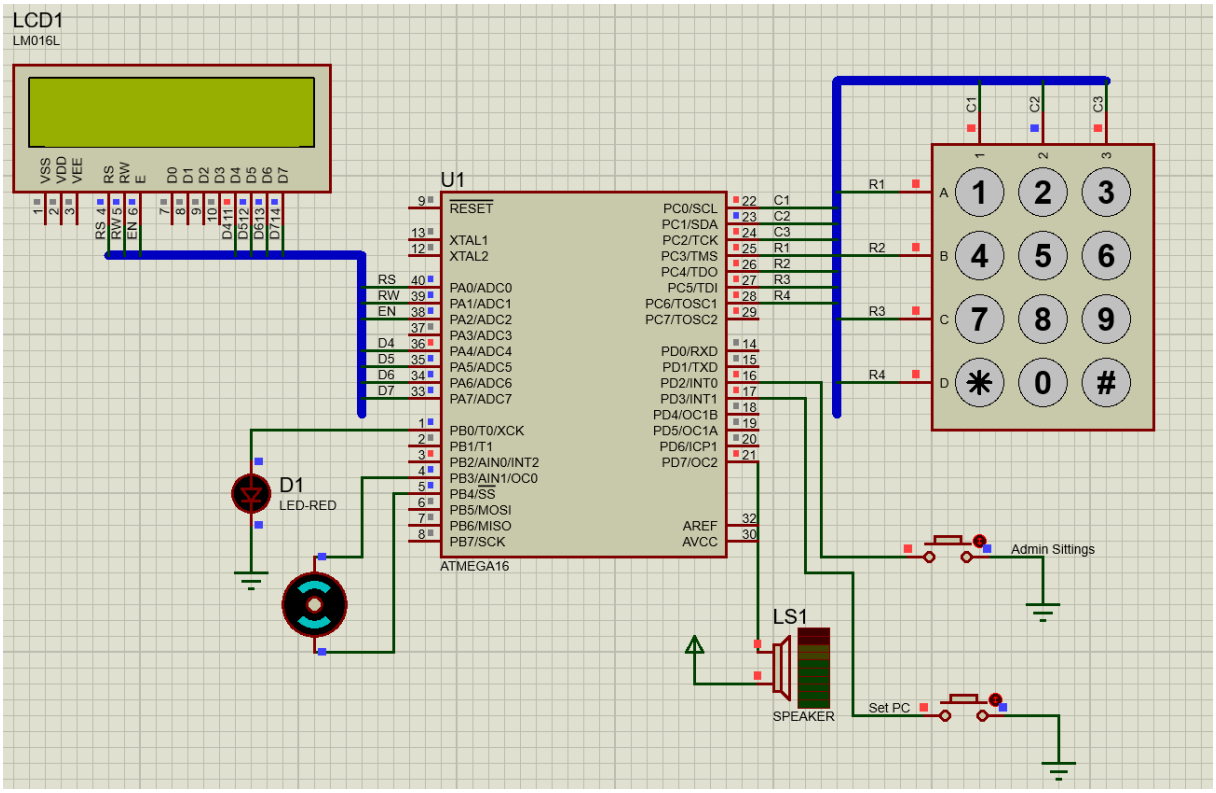

Figure 1: Hardware

Main Program Flowchart

Open Door Flowchart

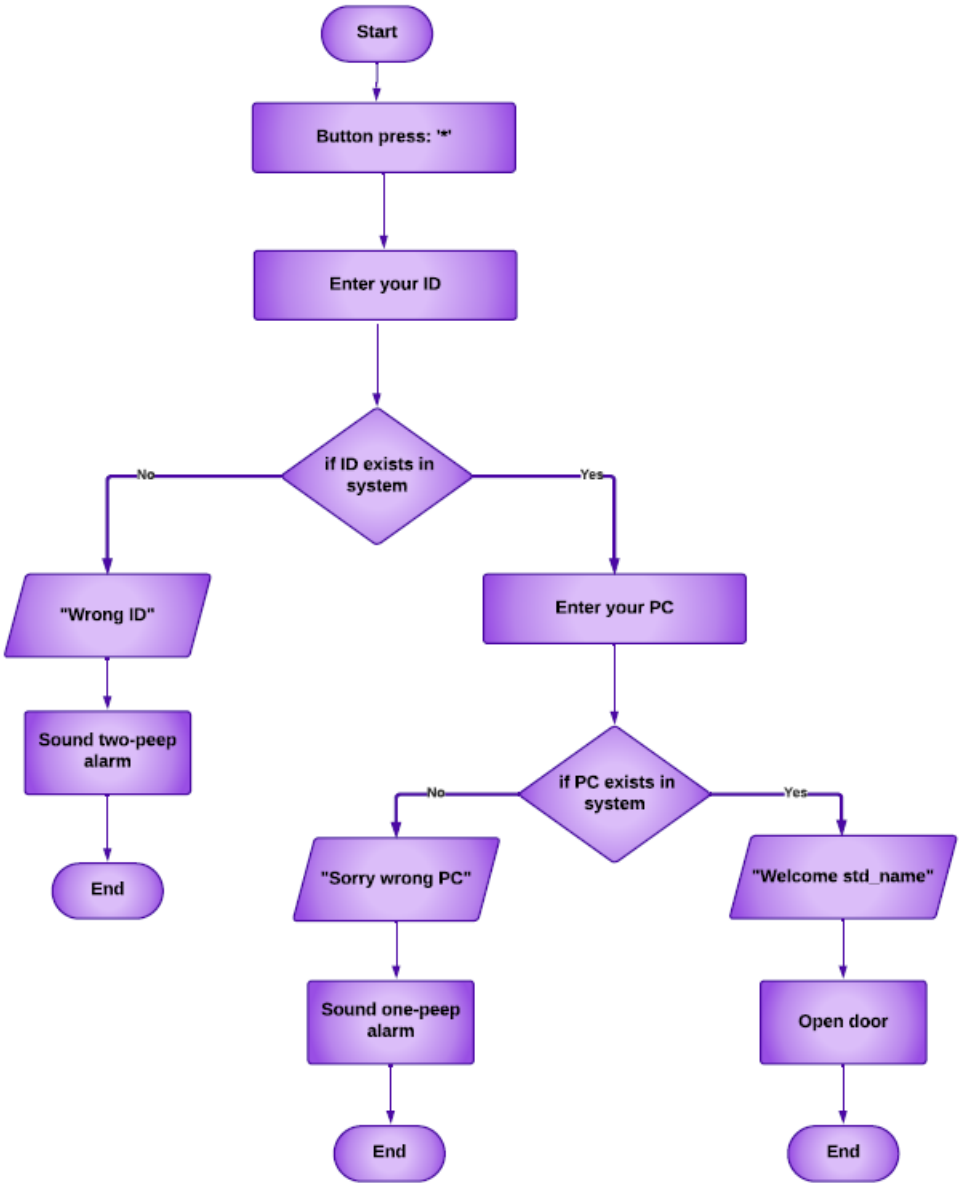

Figure 2: Open Door

Set New PC Flowchart

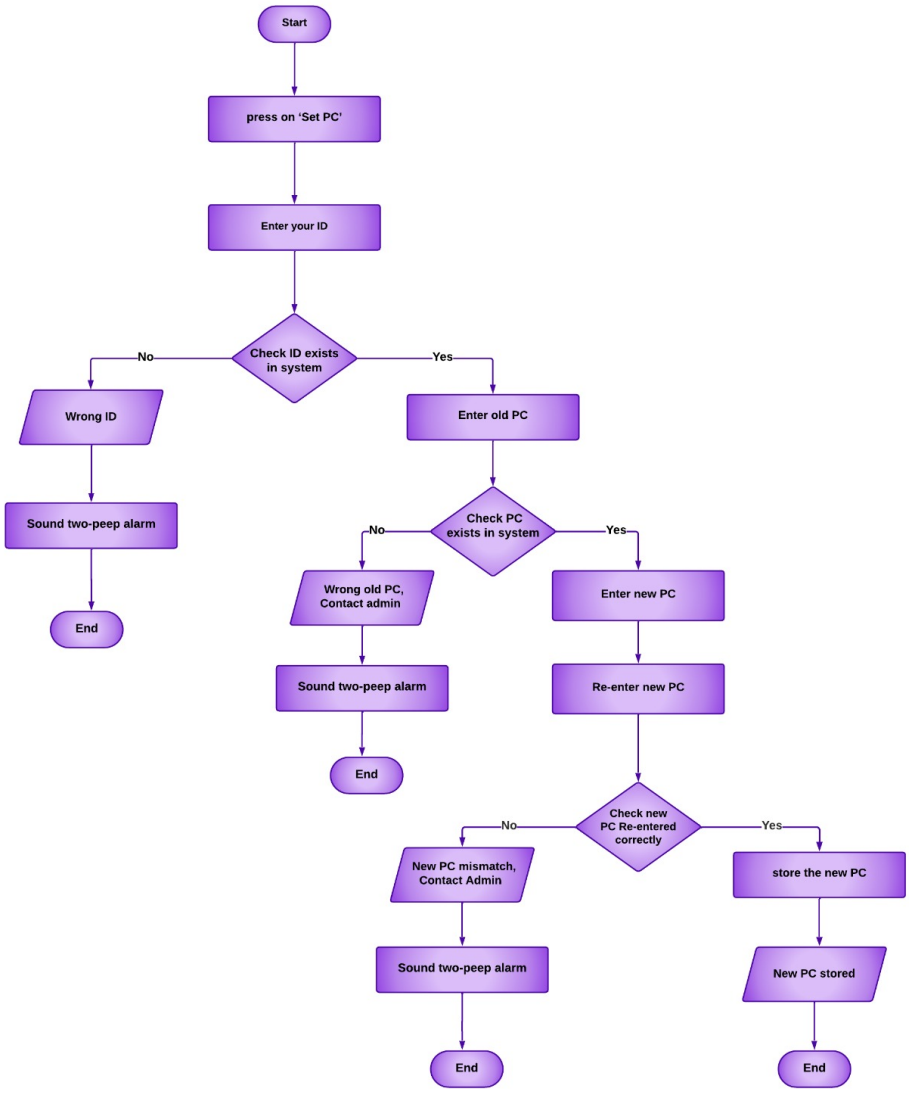

Figure 3: Set New PC

Admin Sittings Flowchart

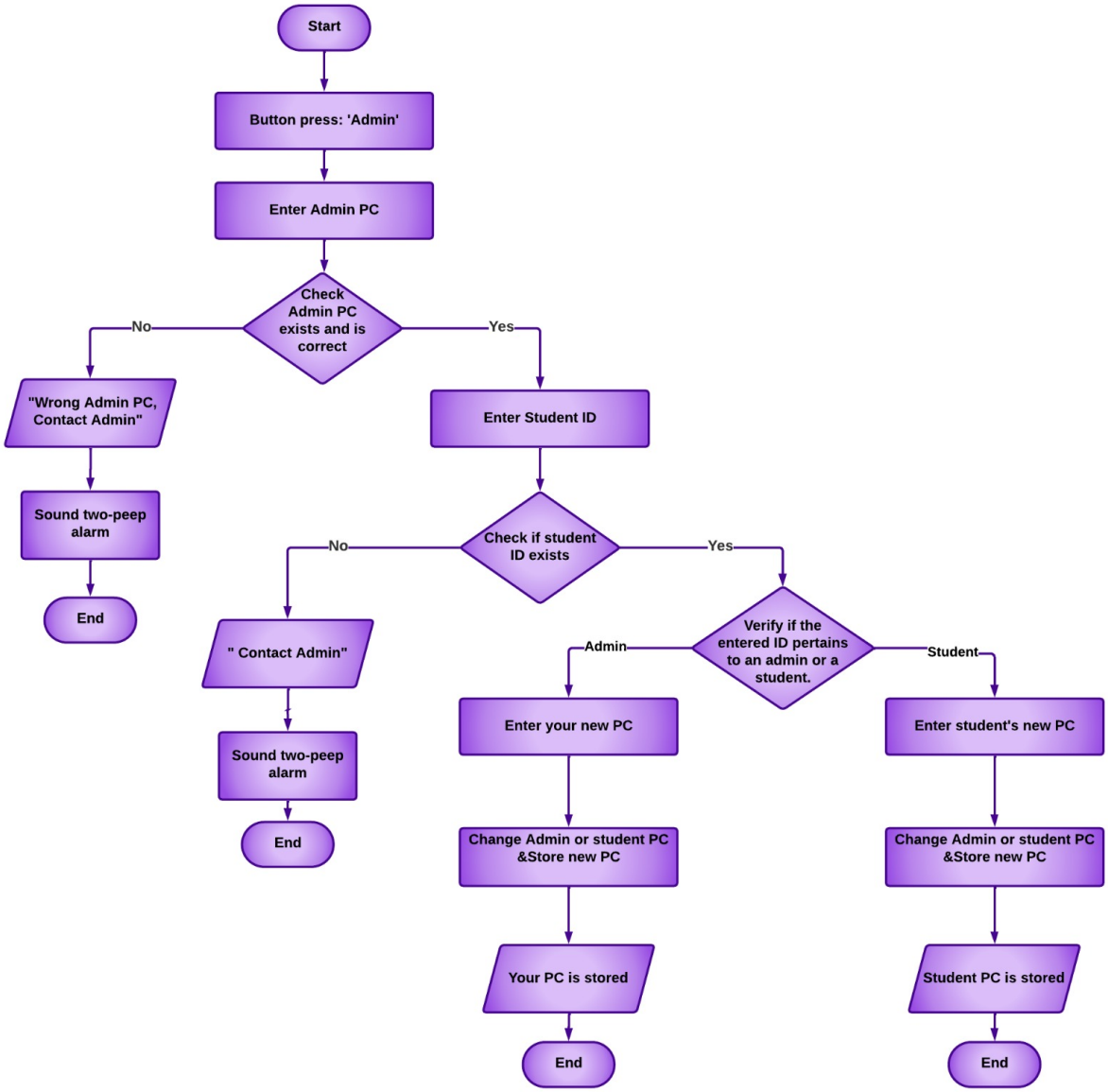

Figure 4: Admin Sittings

Developers

- Islam AbdElhady Hassanein
- Ahmed Hesham Fathall Farag
- Elsherif Shapan Abdelmageed

- Hussein AbdElkader Hussein
- Enas Ragab AbdEllatif
- Mariam Tarek Saad

Main Function

- LockSystem.c

```c
/*
 * Project #1 lock system.c
 *
 * Created: 12/16/2023 1:47:34 AM
 * Author: Hos10
 */

#include "lockSys.h"

void main(void)
{
    char input;

    // Initialize Hardware
    initializeHardware();

    // Initialize user data in EEPROM
    initializeUsers();

    // Initialize interrupts for various modes
    initializeIntrrupts();

    // If user need to open the door must press '*' on the keypad
    while (1)
        {
        input = keypad();
        if (input == 10) // 10 is '*' in keypad
            openCloseDoorMode();
        }
}

interrupt[3] void setPC(void) //  vector no 3 -> INT1
{
    setPCMode();
}

interrupt[2] void admin(void) // vector no 2 -> INT0
{
    adminMode();
}
```

Header File

- lockSys.h

```h
#include <mega16.h>
#include <alcd.h>
#include <delay.h>
#include <string.h>

// Macros for setting and clearing bits in a register
#define bit_set(r, b) r |= 1 << b
#define bit_clr(r, b) r &= ~(1 << b)

// Function prototypes
void initializeHardware();
void initializeKeypad();
void initializeDoor();
void initializeSpeaker();
void initializeIntrrupts();
char keypad();
unsigned char EE_Read(unsigned int address);
void EE_Write(unsigned int address, unsigned char data);
void EE_WriteString(unsigned int address, const char *str);
void EE_ReadString(unsigned int address, char *buffer, unsigned int
    length);
void initializeUsers();
void displayMessage(char *message, int delay_ms_value);
int enterValueWithKeypad(char *buffer);
void generateTone();
void openDoor();
void closeDoor();
void adminMode();
void setPCMode();
void openCloseDoorMode();

// User structure to store user data
typedef struct
{
    char name[6];
    char id[4];
    char pc[4];
} User;
// Array of user data
User users[] =
{
    // name  ID   PC
    {"Prof", "111", "203"},
    {"Ahmed", "126", "129"},
    {"Amr", "128", "325"},
    {"Adel", "130", "426"},
    {"Omer", "132", "079"},
```

```
};

// Function to initialize hardware components
void initializeHardware()
{
    initializeKeypad();
    lcd_init(16); // Initialize the LCD
    initializeDoor();
    initializeSpeaker();
}

// Function to initialize keypad
void initializeKeypad()
{
    // Set keypad ports
    DDRC = 0b00000111;  // 1 unused pin, 4 rows (input), 3 columns (
       output)
    PORTC = 0b11111000; // pull-up resistance
}

// Function to initialize door
void initializeDoor()
{
    // Set the motor pins as output
    DDRB |= (1 << DDB3) | (1 << DDB4);
    // Set the red LED pin as output
    DDRB |= (1 << DDB0);
}

// Function to initialize speaker
void initializeSpeaker()
{
    // Set the speaker as an output
    DDRD .7 = 1;
    PORTD .7 = 1; // Set it to 1 initially
}

// Function to initialize interrupts
void initializeIntrrupts()
{
    DDRB .2 = 0;  // make button as input
    PORTB .2 = 1; // turn on pull up resistance for INT2 intrrupt

    // actual casue INT2
    bit_set(MCUCSR, 6);

    DDRD .2 = 0;  // make button as input
    PORTD .2 = 1; // turn on pull up resistance for INT0 intrrupt

    // actual casue (The falling edge of INT0)
    bit_set(MCUCR, 1);
```

```
97      bit_clr(MCUCR, 0);
98
99      // actual casue (The falling edge of INT1)
100     bit_set(MCUCR, 3);
101     bit_clr(MCUCR, 2);
102
103     DDRD .3 = 0;  // make button SetPC as input
104     PORTD .3 = 1; // turn on pull up resistance
105
106     // Enable global interrupts
107 #asm("sei")
108
109     // GICR INT1 (bit no 7) , SetPC spacific enable
110     bit_set(GICR, 7);
111
112     // GICR INT2 (bit no 5) , open spacific enable
113     bit_set(GICR, 5);
114
115     // GICR INT0 (bit no 6) , admin spacific enable
116     bit_set(GICR, 6);
117 }
118
119 // Function: keypad
120 // Description: Reads the input from a 4x3 matrix keypad and returns
        the corresponding key value.
121 //              The keypad is connected to port C, and the function
        scans each row and column
122 //              combination to determine the pressed key.
123 // Returns: Character representing the pressed key.
124 char keypad()
125 {
126     while (1)
127         {
128         PORTC .0 = 0;
129         PORTC .1 = 1;
130         PORTC .2 = 1;
131
132         switch (PINC)
133             {
134             case 0b11110110:
135                 while (PINC .3 == 0)
136                     ;
137                 return 1;
138             case 0b11101110:
139                 while (PINC .4 == 0)
140                     ;
141                 return 4;
142             case 0b11011110:
143                 while (PINC .5 == 0)
144                     ;
145                 return 7;
```

```
146             case 0b10111110:
147                 while (PINC .6 == 0)
148                     ;
149                 return 10;
150             }
151 
152         PORTC .0 = 1;
153         PORTC .1 = 0;
154         PORTC .2 = 1;
155 
156         switch (PINC)
157             {
158             case 0b11110101:
159                 while (PINC .3 == 0)
160                     ;
161                 return 2;
162             case 0b11101101:
163                 while (PINC .4 == 0)
164                     ;
165                 return 5;
166             case 0b11011101:
167                 while (PINC .5 == 0)
168                     ;
169                 return 8;
170             case 0b10111101:
171                 while (PINC .6 == 0)
172                     ;
173                 return 0;
174             }
175 
176         PORTC .0 = 1;
177         PORTC .1 = 1;
178         PORTC .2 = 0;
179 
180         switch (PINC)
181             {
182             case 0b11110011:
183                 while (PINC .3 == 0)
184                     ;
185                 return 3;
186             case 0b11101011:
187                 while (PINC .4 == 0)
188                     ;
189                 return 6;
190             case 0b11011011:
191                 while (PINC .5 == 0)
192                     ;
193                 return 9;
194             case 0b10111011:
195                 while (PINC .6 == 0)
196                     ;
```

```
                return 11;
            }
        }
}

// Function to read from EEPROM
unsigned char EE_Read(unsigned int address)
{
    while (EECR .1 == 1)
        ;           // Wait till EEPROM is ready
    EEAR = address; // Prepare the address you want to read from
    EECR .0 = 1;    // Execute read command
    return EEDR;
}

// Function to write to EEPROM
void EE_Write(unsigned int address, unsigned char data)
{
    while (EECR .1 == 1)
        ;           // Wait till EEPROM is ready
    EEAR = address; // Prepare the address you want to read from
    EEDR = data;    // Prepare the data you want to write in the
       address above
    EECR .2 = 1;    // Master write enable
    EECR .1 = 1;    // Write Enable
}

// Function to write a string to EEPROM
void EE_WriteString(unsigned int address, const char *str)
{
    // Write each character of the string to EEPROM
    while (*str)
        EE_Write(address++, *str++);
    // Terminate the string with a null character
    EE_Write(address, '\0');
}

// Function to read a string from EEPROM
void EE_ReadString(unsigned int address, char *buffer, unsigned int
   length)
{
    unsigned int i;
    for (i = 0; i < length; ++i)
        {
        buffer[i] = EE_Read(address + i);
        if (buffer[i] == '\0')
            break;
        }
}

// Function to initialize user data in EEPROM
```

```
void initializeUsers()
{
    unsigned int address = 0;
    int i;
    for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
        {
        EE_WriteString(address, users[i].name);
        address += sizeof(users[i].name);

        EE_WriteString(address, users[i].id);
        address += sizeof(users[i].id);

        EE_WriteString(address, users[i].pc);
        address += sizeof(users[i].pc);
        }
}

// Function to display a message on the LCD
void displayMessage(char *message, int delay_ms_value)
{
    lcd_clear();
    lcd_puts(message);
    delay_ms(delay_ms_value);
}

// Function to enter a value with the keypad

int enterValueWithKeypad(char *buffer)
{
    int buffer2[3];

    buffer2[0] = keypad();
    if (buffer2[0] == 10)
        lcd_putchar('*');
    else if (buffer2[0] == 11)
        lcd_putchar('#');
    else
        lcd_putchar(buffer2[0] + '0');

    buffer2[1] = keypad();
    if (buffer2[1] == 10)
        lcd_putchar('*');
    else if (buffer2[1] == 11)
        lcd_putchar('#');
    else
        lcd_putchar(buffer2[1] + '0');

    buffer2[2] = keypad();
    if (buffer2[2] == 10)
        lcd_putchar('*');
    else if (buffer2[2] == 11)
```

```
297         lcd_putchar('#');
298     else
299         lcd_putchar(buffer2[2] + '0');
300 
301     buffer[0] = buffer2[0] + '0';
302     buffer[1] = buffer2[1] + '0';
303     buffer[2] = buffer2[2] + '0';
304     buffer[3] = '\0';
305 
306     delay_ms(1000);
307 
308     return 1;
309 }
310 
311 // Function to generate a tone with speaker
312 void generateTone()
313 {
314     PORTD .7 = 1;
315     delay_ms(500);
316     PORTD .7 = 0;
317     delay_ms(500);
318     PORTD .7 = 1;
319 }
320 
321 // Function to open the door (motor and redled)
322 void openDoor()
323 {
324     // Turn on the red LED light
325     PORTB |= (1 << PORTB0);
326 
327     // Motor movement for smooth opening
328     PORTB &= ~(1 << PORTB3);
329     delay_ms(500);
330     PORTB |= (1 << PORTB4);
331     delay_ms(1000);
332     PORTB &= ~(1 << PORTB4);
333 }
334 // Function to open the door (motor and redled)
335 void closeDoor()
336 {
337     // Turn off the red LED light
338     PORTB &= ~(1 << PORTB0);
339 
340     // Motor movement for smooth closing
341     PORTB |= (1 << PORTB3);
342     delay_ms(500);
343     PORTB |= (1 << PORTB4);
344     delay_ms(1000);
345     PORTB &= ~(1 << PORTB4);
346     PORTB &= ~(1 << PORTB3);
347 
```

```
    // Return to initial position
    PORTB |= (1 << PORTB3);
    delay_ms(500);
    PORTB &= ~(1 << PORTB3);
}

// Interrupt functions

// Function for admin mode
void adminMode()
{
    char enteredPC[4];
    char enteredStudentID[4];
    char enteredNewPC[4];
    User student;
    User admin;
    unsigned int adminPCAddress = 0;
    unsigned int address = 0;
    int userFound = 0;
    int i;

    for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
        {
        EE_ReadString(address, admin.name, sizeof(users[i].name));
        if (strcmp(admin.name, "Prof") == 0)
            {
            address += sizeof(users[i].name);
            EE_ReadString(address, admin.id, sizeof(admin.id));
            address += sizeof(users[i].id);
            EE_ReadString(address, admin.pc, sizeof(admin.pc));
            adminPCAddress = address;
            break;
            }
        address += sizeof(users[i].pc);
        }

    address = 0; // reset the address

    displayMessage("Enter Admin PC: ", 1000);
    lcd_gotoxy(0, 1);

    if (enterValueWithKeypad(enteredPC))
        {

        if (strcmp(admin.pc, enteredPC) == 0)
            {
            displayMessage("Enter Student ID: ", 1000);

            if (enterValueWithKeypad(enteredStudentID))
                {
                int j;
```

```
399                 for (j = 0; j < sizeof(users) / sizeof(users[0]); ++j)
400                     {
401                     address += sizeof(users[j].name);
402                     EE_ReadString(address, student.id, sizeof(student.
                            id));
403                     address += sizeof(users[j].id);
404                     if (strcmp(student.id, enteredStudentID) == 0)
405                         {
406                         displayMessage("Enter student's new PC: ",
                                1000);
407                         if (enterValueWithKeypad(enteredNewPC))
408                             {
409                             // Set the new pc for this student, address
                                    is for student PC
410                             EE_WriteString(address, enteredNewPC);
411                             displayMessage("Student PC is stored",
                                3000);
412                             userFound = 1;
413                             break;
414                             }
415                         }
416                     else if (strcmp(admin.id, enteredStudentID) == 0)
417                         {
418                         displayMessage("Enter your new PC: ", 1000);
419                         lcd_gotoxy(0, 1);
420                         if (enterValueWithKeypad(enteredNewPC))
421                             {
422                             // Set the new pc for this user (Admin),
                                    address is for admin PC
423                             EE_WriteString(adminPCAddress, enteredNewPC
                                    );
424                             displayMessage("Your PC is stored", 3000);
425                             userFound = 1;
426                             break;
427                             }
428                         }
429                     address += sizeof(users[i].pc);
430                     }
431                 }
432             }
433         }
434
435     if (!userFound)
436         {
437         displayMessage("Contact Admin", 3000);
438         // Two peeps alarm
439         generateTone();
440         generateTone();
441         }
442     delay_ms(5000);
443     lcd_clear();
```

```
444 }
445
446 // Function for set PC mode
447 void setPCMode()
448 {
449     char enteredID[5]; // Change data type to string
450     User currentUser;
451     unsigned int address = 0;
452     int userFound = 0;
453     int i;
454     char enteredNewPC[5];   // define enteredNewPC array to hold the
        new PC
455     char reenteredNewPC[5]; // define reenteredNewPC array to hold the
        Re-entered new PC
456
457     lcd_clear();
458     displayMessage("Enter your ID:", 1000);
459     lcd_gotoxy(0, 1);
460     if (enterValueWithKeypad(enteredID))
461         {
462         char enteredOldPC[5];
463         // search for the entered ID in the user data
464         for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
465             {
466             address += sizeof(users[i].name);
467             EE_ReadString(address, currentUser.id, sizeof(currentUser.
                id)); // Read ID as a string
468
469             if (strcmp(currentUser.id, enteredID) == 0)
470                 {
471                 // ID found, verify the old PC
472                 address += sizeof(currentUser.id);
473                 EE_ReadString(address, currentUser.pc, sizeof(
                    currentUser.pc)); // Read PC as a string
474                 displayMessage("Enter old PC:", 1000);
475                 lcd_gotoxy(0, 1);
476
477                 if (enterValueWithKeypad(enteredOldPC))
478                     {
479                     if (strcmp(currentUser.pc, enteredOldPC) == 0)
480                         {
481                         // Old PC verified
482                         displayMessage("Enter new PC:", 1000);
483                         lcd_gotoxy(0, 1);
484                         enterValueWithKeypad(enteredNewPC);
485
486                         lcd_clear();
487                         displayMessage("Re-enter new PC:", 1000);
488                         lcd_gotoxy(0, 1);
489                         enterValueWithKeypad(reenteredNewPC);
490
```

```
491                             if (strcmp(enteredNewPC, reenteredNewPC) == 0)
492                                 {
493                                 // If new PC entered correctly, store it
494                                 EE_WriteString(address, enteredNewPC);
495                                 displayMessage("New PC stored", 1000);
496                                 }
497                             else
498                                 {
499                                 displayMessage("New PC mismatch, Contact
                                        admin", 1000);
500                                 generateTone();
501                                 generateTone();
502                                 }
503                             }
504                         else
505                             {
506                             displayMessage("Wrong old PC,   Contact admin",
                                    1000);
507 
508                             generateTone();
509                             generateTone();
510                             }
511                         }
512 
513                     userFound = 1;
514                     break;
515                     }
516 
517                 address += sizeof(users[i].id);
518                 address += sizeof(users[i].pc);
519                 }
520 
521             if (!userFound)
522                 {
523                 displayMessage("Wrong ID", 1000);
524                 generateTone();
525                 generateTone();
526                 }
527             delay_ms(5000);
528             lcd_clear();
529             }
530 }
531 
532 // Function for open/close door mode
533 void openCloseDoorMode()
534 {
535     char enteredID[4]; // Change data type to string
536     User currentUser;
537     unsigned int address = 0;
538     int userFound = 0;
539     int i;
```

```
540
541     displayMessage("Enter your ID: ", 1000);
542     lcd_gotoxy(0, 1);
543
544     if (enterValueWithKeypad(enteredID))
545         {
546         char enteredPC[4];
547         for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
548             {
549             EE_ReadString(address, currentUser.name, sizeof(users[i].
                name));
550             address += sizeof(users[i].name);
551             EE_ReadString(address, currentUser.id, sizeof(currentUser.
                id)); // Read ID as a string
552
553             if (strcmp(currentUser.id, enteredID) == 0)
554                 {
555                 address += sizeof(users[i].id);
556                 EE_ReadString(address, currentUser.pc, sizeof(
                    currentUser.pc)); // Read PC as a string
557
558                 displayMessage("Enter your PC: ", 1000);
559                 lcd_gotoxy(0, 1);
560
561                 if (enterValueWithKeypad(enteredPC))
562                     {
563                     if (strcmp(currentUser.pc, enteredPC) == 0)
564                         {
565                         lcd_clear();
566                         lcd_puts("Welcome, ");
567                         lcd_puts(currentUser.name);
568                         openDoor();
569                         delay_ms(2000); // Wait for 2 seconds with the
                            door open
570
571                         closeDoor();
572                         delay_ms(2000); // Wait for 2 seconds with the
                            door closed
573                         }
574                     else
575                         {
576                         displayMessage("Sorry wrong PC", 1000);
577                         // one peep alarm
578                         generateTone();
579                         }
580                     }
581                 userFound = 1;
582                 break;
583                 }
584
585             address += sizeof(users[i].id);
```

```
586                 address += sizeof(users[i].pc);
587                 }
588             }
589 
590     if (!userFound)
591         {
592         displayMessage("Wrong ID", 1000);
593         // Two peeps alarm
594         generateTone();
595         generateTone();
596         }
597     lcd_clear();
598 }
```