# 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. Speaker (Peeps alarm)
6. Keypad 4x3
7. Three Buttons for interrupts

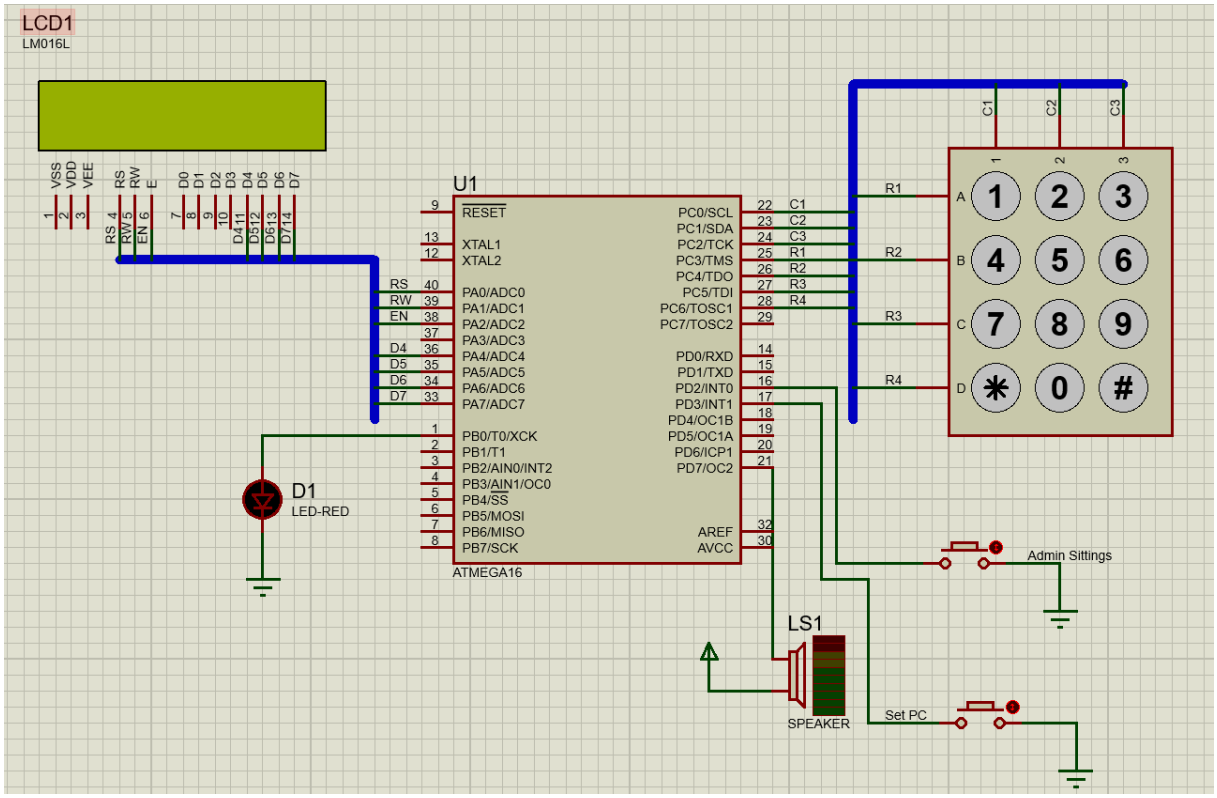


**Figure 1:** Hardware

## Main Program Flowchart

### Open Door Flowchart

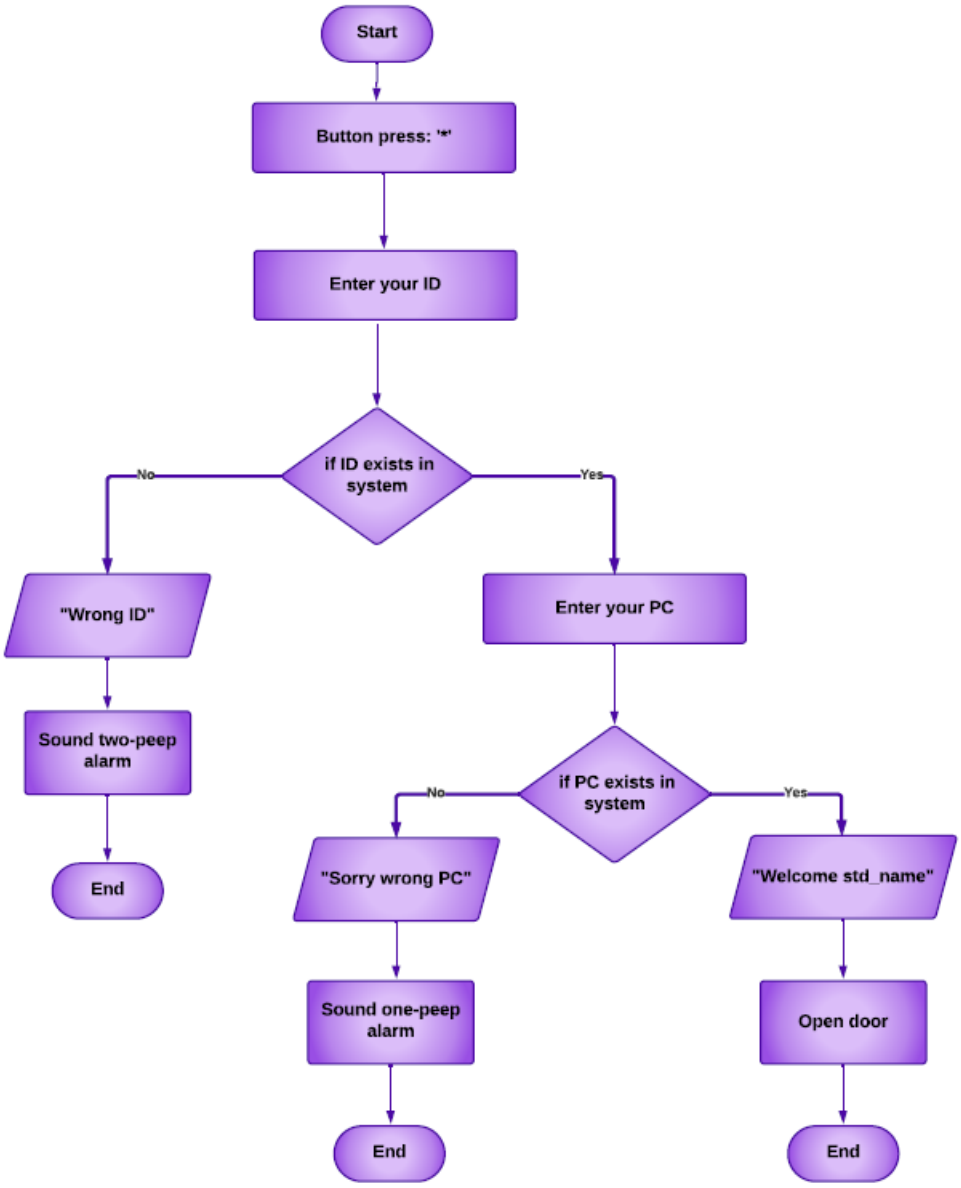


**Figure 2:** Open Door

## Set New PC Flowchart

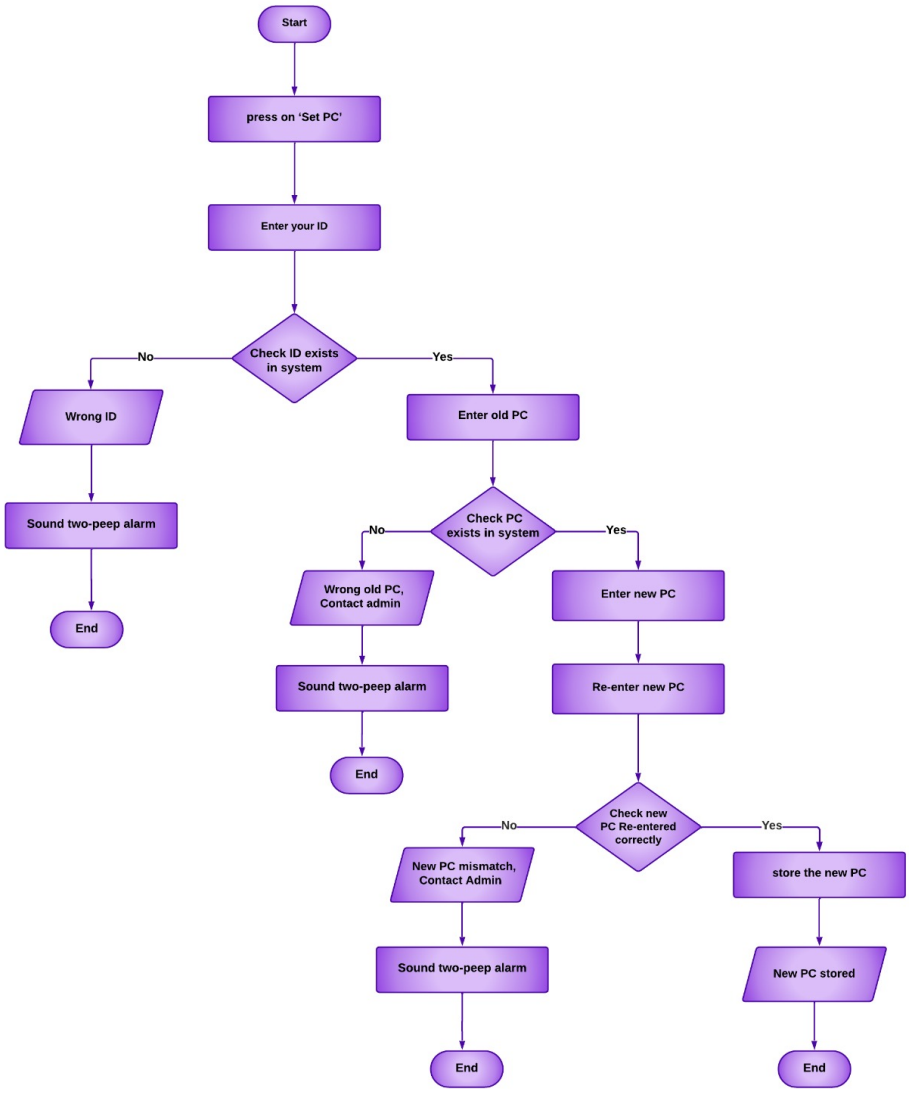


**Figure 3:** Set New PC

### Admin Sittings Flowchart

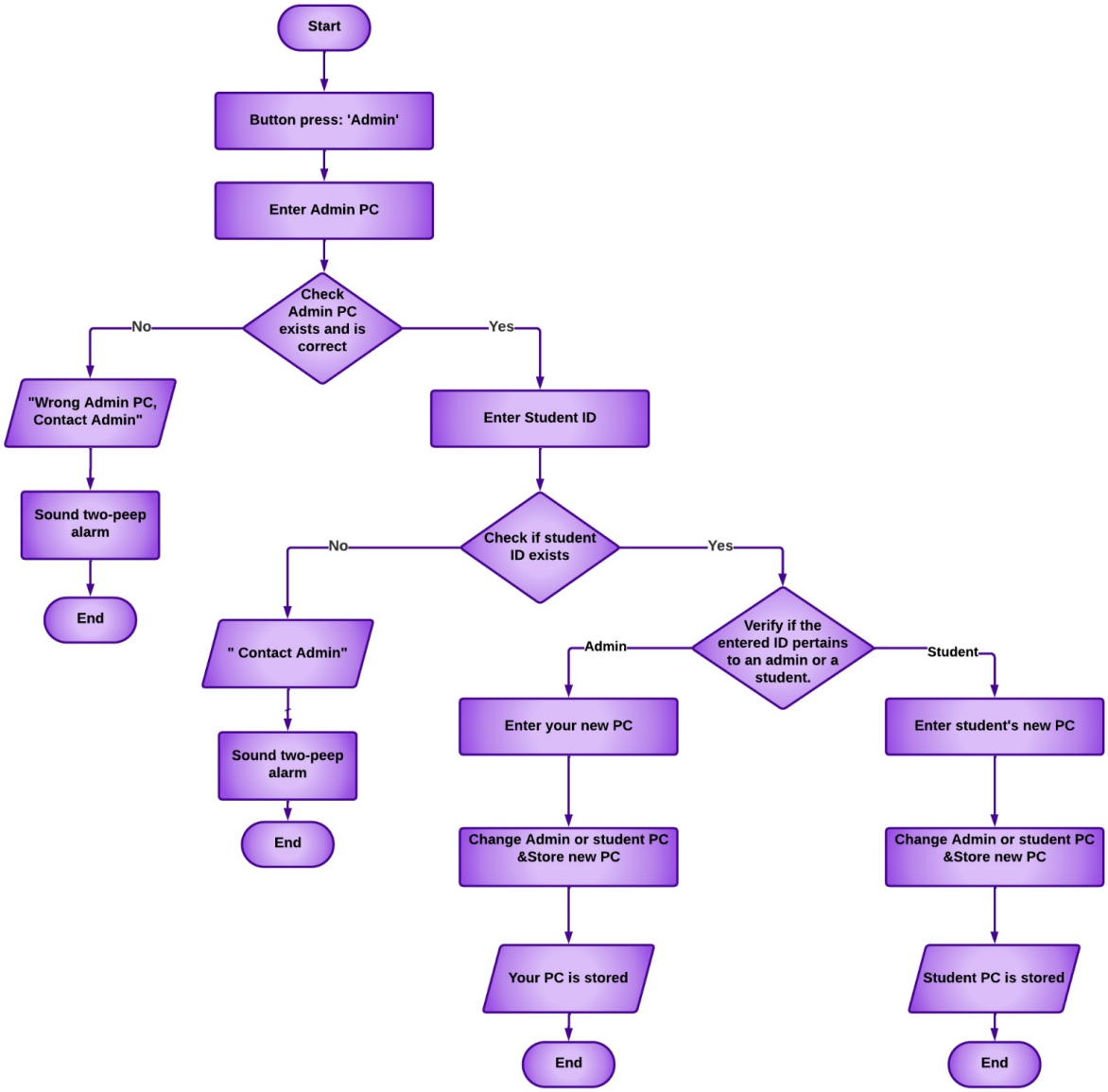


**Figure 4:** Admin Sittings

## 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 == '*')
            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

```
#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 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"},
};

```

```
47 // Function to initialize hardware components
48 void initializeHardware()
49 {
50     initializeKeypad();
51     lcd_init(16); // Initialize the LCD
52     initializeDoor();
53     initializeSpeaker();
54 }
55
56 // Function to initialize keypad
57 void initializeKeypad()
58 {
59     // Set keypad ports
60     DDRC = 0b00000111;  // 1 unused pin, 4 rows (input), 3 columns (
           output)
61     PORTC = 0b11111000; // pull-up resistance
62 }
63
64 // Function to initialize door
65 void initializeDoor()
66 {
67     DDRB .0 = 0; // Set the door as input (by default, the door is
           closed)
68     PORTB .0 = 1; // turn on pull-up resistance
69 }
70
71 // Function to initialize speaker
72 void initializeSpeaker()
73 {
74     DDRD .7 = 1; // Set the speaker as an output
75     PORTD .7 = 1; // Set it to 1 initially
76 }
77
78 // Function to initialize interrupts
79 void initializeIntrrupts()
80 {
81     DDRB .2 = 0;  // make button as input
82     PORTB .2 = 1; // turn on pull up resistance for INT2 intrrupt
83
84     // actual casue INT2
85     bit_set(MCUCSR, 6);
86
87     DDRD .2 = 0;  // make button as input
88     PORTD .2 = 1; // turn on pull up resistance for INT0 intrrupt
89
90     // actual casue (The falling edge of INT0)
91     bit_set(MCUCR, 1);
92     bit_clr(MCUCR, 0);
93
94     // actual casue (The falling edge of INT1)
95     bit_set(MCUCR, 3);
```

```
96      bit_clr(MCUCR, 2);
97
98      DDRD .3 = 0;  // make button SetPC as input
99      PORTD .3 = 1; // turn on pull up resistance
100
101     // Enable global interrupts
102 #asm("sei")
103
104     // GICR INT1 (bit no 7) , SetPC spacific enable
105     bit_set(GICR, 7);
106
107     // GICR INT2 (bit no 5) , open spacific enable
108     bit_set(GICR, 5);
109
110     // GICR INT0 (bit no 6) , admin spacific enable
111     bit_set(GICR, 6);
112 }
113
114 // Function: keypad
115 // Description: Reads the input from a 4x3 matrix keypad and returns
        the corresponding key value.
116 //              The keypad is connected to port C, and the function
        scans each row and column
117 //              combination to determine the pressed key.
118 // Returns: Character representing the pressed key.
119 char keypad()
120 {
121     while (1)
122     {
123         PORTC .0 = 0;
124         PORTC .1 = 1;
125         PORTC .2 = 1;
126
127         switch (PINC)
128         {
129         case 0b11110110:
130             while (PINC .3 == 0)
131                 ;
132             return 1;
133         case 0b11101110:
134             while (PINC .4 == 0)
135                 ;
136             return 4;
137         case 0b11011110:
138             while (PINC .5 == 0)
139                 ;
140             return 7;
141         case 0b10111110:
142             while (PINC .6 == 0)
143                 ;
144             return '*';
```

```
145         }
146
147         PORTC .0 = 1;
148         PORTC .1 = 0;
149         PORTC .2 = 1;
150
151         switch (PINC)
152         {
153         case 0b11110101:
154             while (PINC .3 == 0)
155                 ;
156             return 2;
157         case 0b11101101:
158             while (PINC .4 == 0)
159                 ;
160             return 5;
161         case 0b11011101:
162             while (PINC .5 == 0)
163                 ;
164             return 8;
165         case 0b10111101:
166             while (PINC .6 == 0)
167                 ;
168             return 0;
169         }
170
171         PORTC .0 = 1;
172         PORTC .1 = 1;
173         PORTC .2 = 0;
174
175         switch (PINC)
176         {
177         case 0b11110011:
178             while (PINC .3 == 0)
179                 ;
180             return 3;
181         case 0b11101011:
182             while (PINC .4 == 0)
183                 ;
184             return 6;
185         case 0b11011011:
186             while (PINC .5 == 0)
187                 ;
188             return 9;
189         case 0b10111011:
190             while (PINC .6 == 0)
191                 ;
192             return 11;
193         }
194     }
195 }
```

```
196
197 // Function to read from EEPROM
198 unsigned char EE_Read(unsigned int address)
199 {
200     while (EECR .1 == 1); // Wait till EEPROM is ready
201     EEAR = address; // Prepare the address you want to read from
202     EECR .0 = 1;    // Execute read command
203     return EEDR;
204 }
205
206 // Function to write to EEPROM
207 void EE_Write(unsigned int address, unsigned char data)
208 {
209     while (EECR .1 == 1); // Wait till EEPROM is ready
210     EEAR = address; // Prepare the address you want to read from
211     EEDR = data;    // Prepare the data you want to write in the
        address above
212     EECR .2 = 1;    // Master write enable
213     EECR .1 = 1;    // Write Enable
214 }
215
216 // Function to write a string to EEPROM
217 void EE_WriteString(unsigned int address, const char *str)
218 {
219     // Write each character of the string to EEPROM
220     while (*str)
221     EE_Write(address++, *str++);
222     // Terminate the string with a null character
223     EE_Write(address, '\0');
224 }
225
226 // Function to read a string from EEPROM
227 void EE_ReadString(unsigned int address, char *buffer, unsigned int
    length)
228 {
229     unsigned int i;
230     for (i = 0; i < length; ++i)
231     {
232         buffer[i] = EE_Read(address + i);
233         if (buffer[i] == '\0')
234             break;
235     }
236 }
237
238 // Function to initialize user data in EEPROM
239 void initializeUsers()
240 {
241     unsigned int address = 0;
242     int i;
243     for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
244     {
```

```
245         EE_WriteString(address, users[i].name);
246         address += sizeof(users[i].name);
247 
248         EE_WriteString(address, users[i].id);
249         address += sizeof(users[i].id);
250 
251         EE_WriteString(address, users[i].pc);
252         address += sizeof(users[i].pc);
253     }
254 }
255 
256 // Function to display a message on the LCD
257 void displayMessage(char *message, int delay_ms_value)
258 {
259     lcd_clear();
260     lcd_puts(message);
261     delay_ms(delay_ms_value);
262 }
263 
264 // Function to enter a value with the keypad
265 int enterValueWithKeypad(char *buffer)
266 {
267     buffer[0] = keypad() + '0';
268     lcd_putchar(buffer[0]);
269     buffer[1] = keypad() + '0';
270     lcd_putchar(buffer[1]);
271     buffer[2] = keypad() + '0';
272     lcd_putchar(buffer[2]);
273     buffer[3] = '\0'; // Null-terminate the string
274 
275     delay_ms(1000);
276 
277     return 1; // Return a non-zero value to indicate success
278 }
279 
280 // Function to generate a tone with speaker
281 void generateTone()
282 {
283     PORTD .7 = 1;
284     delay_ms(500);
285     PORTD .7 = 0;
286     delay_ms(500);
287     PORTD .7 = 1;
288 }
289 
290 // Interrupt functions
291 
292 // Function for admin mode
293 void adminMode()
294 {
295     char enteredPC[4];
```

```
296     char enteredStudentID[4];
297     char enteredNewPC[4];
298     User student;
299     User admin;
300     unsigned int adminPCAddress = 0;
301     unsigned int address = 0;
302     int userFound = 0;
303     int i;
304 
305     for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
306     {
307         EE_ReadString(address, admin.name, sizeof(users[i].name));
308         if (strcmp(admin.name, "Prof") == 0)
309         {
310             address += sizeof(users[i].name);
311             EE_ReadString(address, admin.id, sizeof(admin.id));
312             address += sizeof(users[i].id);
313             EE_ReadString(address, admin.pc, sizeof(admin.pc));
314             adminPCAddress = address;
315             break;
316         }
317         address += sizeof(users[i].pc);
318     }
319 
320     address = 0; // reset the address
321 
322     displayMessage("Enter Admin PC: ", 1000);
323     lcd_gotoxy(0, 1);
324 
325     if (enterValueWithKeypad(enteredPC))
326     {
327 
328         if (strcmp(admin.pc, enteredPC) == 0)
329         {
330             displayMessage("Enter Student ID: ", 1000);
331 
332             if (enterValueWithKeypad(enteredStudentID))
333             {
334                 int j;
335                 for (j = 0; j < sizeof(users) / sizeof(users[0]); ++j)
336                 {
337                     address += sizeof(users[j].name);
338                     EE_ReadString(address, student.id, sizeof(student.
                        id));
339                     address += sizeof(users[j].id);
340                     if (strcmp(student.id, enteredStudentID) == 0)
341                     {
342                         displayMessage("Enter student's new PC: ",
                            1000);
343                         if (enterValueWithKeypad(enteredNewPC))
344                         {
```

```
345                                 // Set the new pc for this student, address
                                        is for student PC
346                                 EE_WriteString(address, enteredNewPC);
347                                 displayMessage("Student PC is stored",
                                        3000);
348                                 userFound = 1;
349                                 break;
350                             }
351                         }
352                         else if (strcmp(admin.id, enteredStudentID) == 0)
353                         {
354                             displayMessage("Enter your new PC: ", 1000);
355                             lcd_gotoxy(0, 1);
356                             if (enterValueWithKeypad(enteredNewPC))
357                             {
358                                 // Set the new pc for this user (Admin),
                                        address is for admin PC
359                                 EE_WriteString(adminPCAddress, enteredNewPC
                                        );
360                                 displayMessage("Your PC is stored", 3000);
361                                 userFound = 1;
362                                 break;
363                             }
364                         }
365                         address += sizeof(users[i].pc);
366                     }
367                 }
368             }
369         }
370 
371         if (!userFound)
372         {
373             displayMessage("Contact Admin", 3000);
374             // Two peeps alarm
375             generateTone();
376             generateTone();
377         }
378         delay_ms(5000);
379         lcd_clear();
380     }
381 
382     // Function for set PC mode
383     void setPCMode()
384     {
385         char enteredID[5]; // Change data type to string
386         User currentUser;
387         unsigned int address = 0;
388         int userFound = 0;
389         int i;
390         char enteredNewPC[5];   // define enteredNewPC array to hold the
                new PC
```

```
391     char reenteredNewPC[5]; // define reenteredNewPC array to hold the
            Re-entered new PC
392 
393     lcd_clear();
394     displayMessage("Enter your ID:", 1000);
395     lcd_gotoxy(0, 1);
396     if (enterValueWithKeypad(enteredID))
397     {
398         char enteredOldPC[5];
399         // search for the entered ID in the user data
400         for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
401         {
402             address += sizeof(users[i].name);
403             EE_ReadString(address, currentUser.id, sizeof(currentUser.
                id)); // Read ID as a string
404 
405             if (strcmp(currentUser.id, enteredID) == 0)
406             {
407                 // ID found, verify the old PC
408                 address += sizeof(currentUser.id);
409                 EE_ReadString(address, currentUser.pc, sizeof(
                    currentUser.pc)); // Read PC as a string
410                 displayMessage("Enter old PC:", 1000);
411                 lcd_gotoxy(0, 1);
412 
413                 if (enterValueWithKeypad(enteredOldPC))
414                 {
415                     if (strcmp(currentUser.pc, enteredOldPC) == 0)
416                     {
417                         // Old PC verified
418                         displayMessage("Enter new PC:", 1000);
419                         lcd_gotoxy(0, 1);
420                         enterValueWithKeypad(enteredNewPC);
421 
422                         lcd_clear();
423                         displayMessage("Re-enter new PC:", 1000);
424                         lcd_gotoxy(0, 1);
425                         enterValueWithKeypad(reenteredNewPC);
426 
427                         if (strcmp(enteredNewPC, reenteredNewPC) == 0)
428                         {
429                             // If new PC entered correctly, store it
430                             EE_WriteString(address, enteredNewPC);
431                             displayMessage("New PC stored", 1000);
432                         }
433                         else
434                         {
435                             displayMessage("New PC mismatch, Contact
                                admin", 1000);
436                             generateTone();
437                             generateTone();
```

```
438                         }
439                     }
440                     else
441                     {
442                         displayMessage("Wrong old PC,   Contact admin",
                                1000);
443 
444                         generateTone();
445                         generateTone();
446                     }
447                 }
448 
449                 userFound = 1;
450                 break;
451             }
452 
453             address += sizeof(users[i].id);
454             address += sizeof(users[i].pc);
455         }
456 
457         if (!userFound)
458         {
459             displayMessage("Wrong ID", 1000);
460             generateTone();
461             generateTone();
462         }
463         delay_ms(5000);
464         lcd_clear();
465     }
466 }
467 
468 // Function for open/close door mode
469 void openCloseDoorMode()
470 {
471     char enteredID[4]; // Change data type to string
472     User currentUser;
473     unsigned int address = 0;
474     int userFound = 0;
475     int i;
476 
477     displayMessage("Enter your ID: ", 1000);
478     lcd_gotoxy(0, 1);
479 
480     if (enterValueWithKeypad(enteredID))
481     {
482         char enteredPC[4];
483         for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
484         {
485             EE_ReadString(address, currentUser.name, sizeof(users[i].
                name));
486             address += sizeof(users[i].name);
```

```
487             EE_ReadString(address, currentUser.id, sizeof(currentUser.
                    id)); // Read ID as a string
488 
489             if (strcmp(currentUser.id, enteredID) == 0)
490             {
491 
492                 address += sizeof(users[i].id);
493                 EE_ReadString(address, currentUser.pc, sizeof(
                        currentUser.pc)); // Read PC as a string
494 
495                 displayMessage("Enter your PC: ", 1000);
496                 lcd_gotoxy(0, 1);
497 
498                 if (enterValueWithKeypad(enteredPC))
499                 {
500                     if (strcmp(currentUser.pc, enteredPC) == 0)
501                     {
502                         lcd_clear();
503                         lcd_puts("Welcome, ");
504                         lcd_puts(currentUser.name);
505                         // Open the door
506                         DDRB .0 = 1;
507                     }
508                     else
509                     {
510                         displayMessage("Sorry wrong PC", 1000);
511                         // one peep alarm
512                         generateTone();
513                     }
514                 }
515                 userFound = 1;
516                 break;
517             }
518 
519             address += sizeof(users[i].id);
520             address += sizeof(users[i].pc);
521         }
522     }
523 
524     if (!userFound)
525     {
526         displayMessage("Wrong ID", 1000);
527         // Two peeps alarm
528         generateTone();
529         generateTone();
530     }
531     delay_ms(5000);
532     // close the door and clear lcd
533     DDRB .0 = 0;
534     lcd_clear();
535 }
```