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. Two Buttons for interrupts

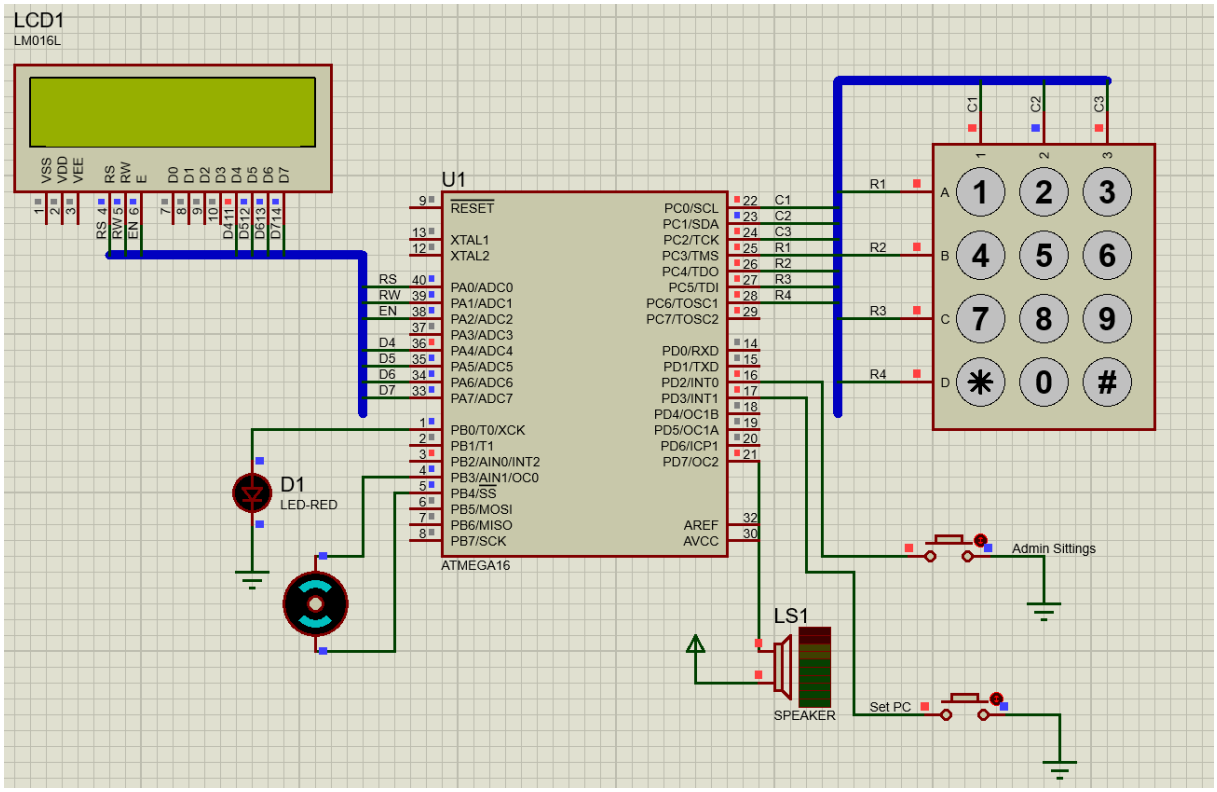

Figure 1: Hardware

Main Program Flowchart

Open Door Flowchart

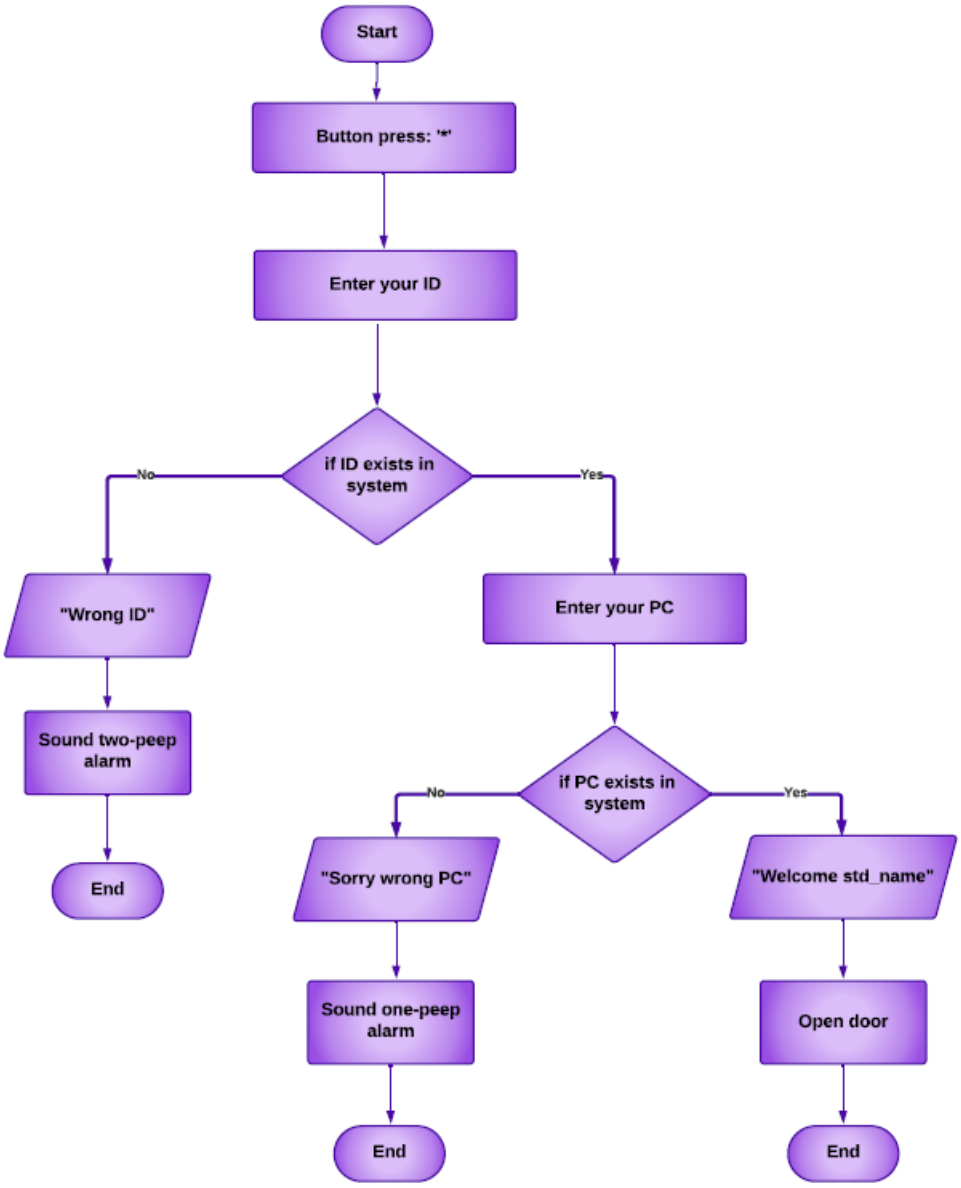

Figure 2: Open Door

Set New PC Flowchart

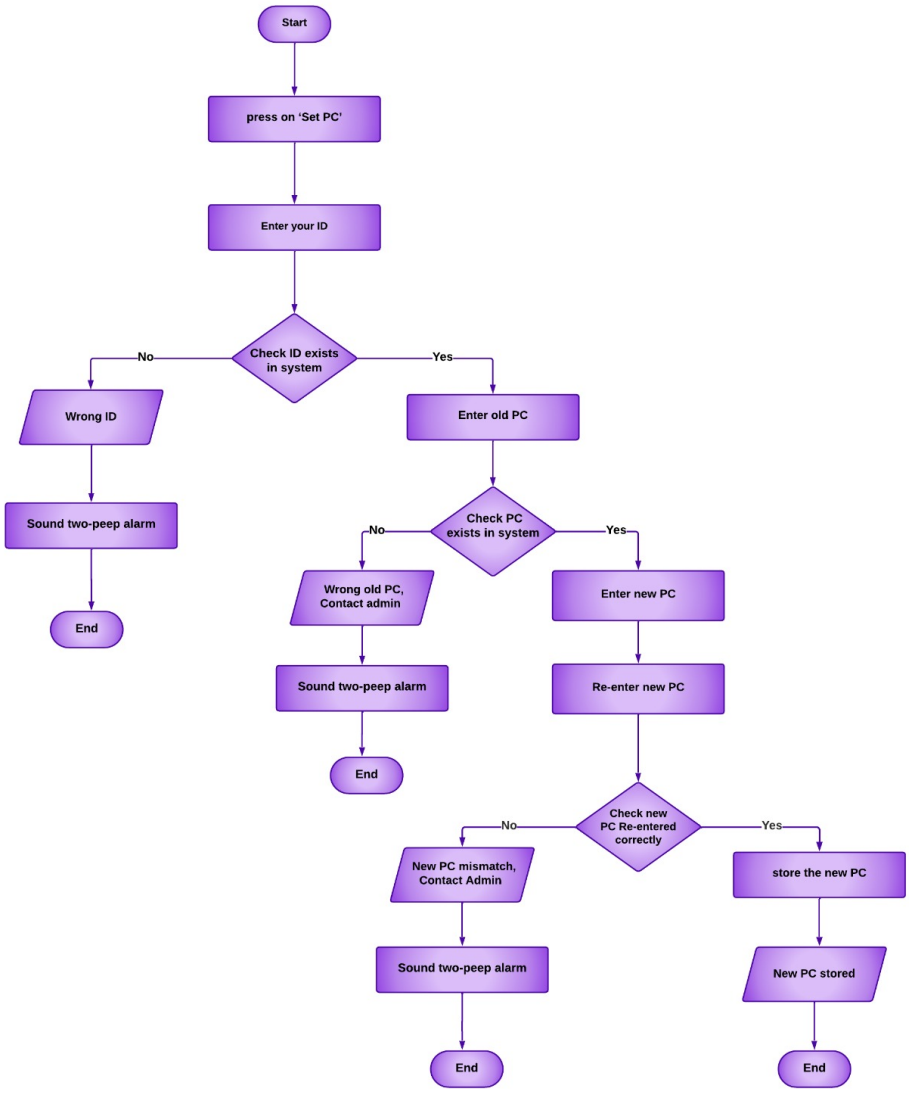

Figure 3: Set New PC

Admin Sittings Flowchart

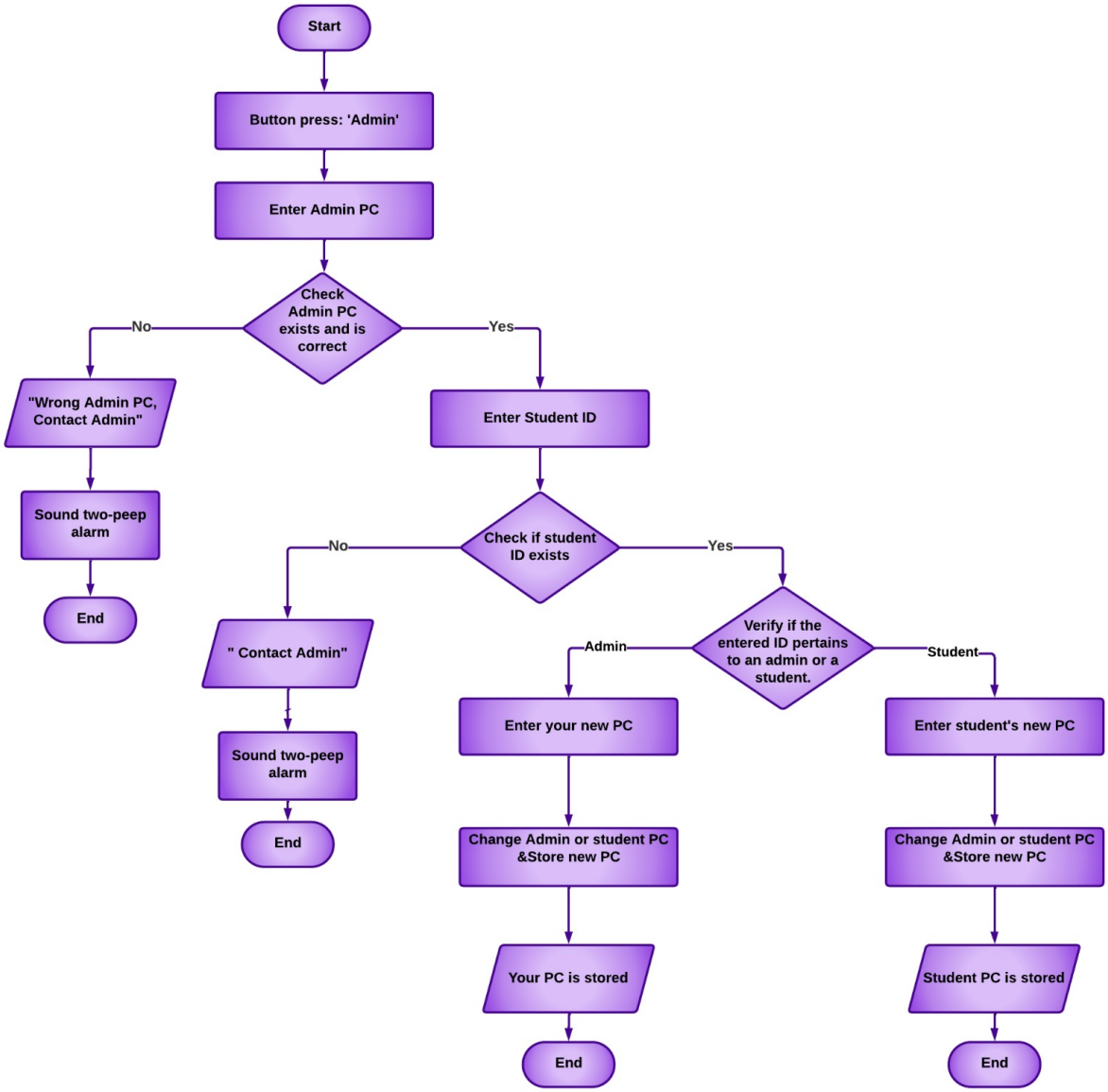

Figure 4: Admin Sittings

Default Users in System

Name	ID	PC
Prof	111	203
Ahmed	126	129
Amr	128	325
Adel	130	426
Omer	132	079

Developers

- Islam AbdElhady Hassanein
- Ahmed Hesham Fathall Farag
- Elsherif Shapan Abdelmageed
- Hussein AbdElkader Hussein
- Enas Ragab AbdEllatif
- Mariam Tarek Saad

Main File

LockSystem.c

- LockSystem.c - Main file for the embedded lock system project.
- This file serves as the main entry point for the lock system project. It includes necessary header files
- and contains the main function that initializes hardware, user data, and interrupts. The main loop continuously checks for a specific input (‘*’) on the keypad to trigger the door open/close mode.

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

#include "lockSysInit.h"
#include "lockSysMode.h"

```

```c
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 Files

lockSysInit.h

- lockSysInit.h - Header file containing initialization functions for the embedded lock system.
- This file includes functions for initializing various hardware components such as the keypad, LCD, door motor, speaker, and interrupts. Additionally, it defines macros for setting and clearing bits in a register.
- The file also includes the definition of the User structure and initializes user data in EEPROM.

```c
#include <mega16.h>
#include <alcd.h>
#include "lockSysReadWrite.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();
char keypad();
void initializeDoor();
void initializeSpeaker();
void initializeIntrrupts();
void initializeUsers();

// 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: keypad
// Description: Reads the input from a 4x3 matrix keypad and returns
   the corresponding key value.
```

```
55 //              The keypad is connected to port C, and the function
       scans each row and column
56 //              combination to determine the pressed key.
57 // Returns: Character representing the pressed key.
58 char keypad()
59 {
60     while (1)
61     {
62         PORTC .0 = 0;
63         PORTC .1 = 1;
64         PORTC .2 = 1;
65 
66         switch (PINC)
67         {
68         case 0b11110110:
69             while (PINC .3 == 0)
70                 ;
71             return 1;
72         case 0b11101110:
73             while (PINC .4 == 0)
74                 ;
75             return 4;
76         case 0b11011110:
77             while (PINC .5 == 0)
78                 ;
79             return 7;
80         case 0b10111110:
81             while (PINC .6 == 0)
82                 ;
83             return 10;
84         }
85 
86         PORTC .0 = 1;
87         PORTC .1 = 0;
88         PORTC .2 = 1;
89 
90         switch (PINC)
91         {
92         case 0b11110101:
93             while (PINC .3 == 0)
94                 ;
95             return 2;
96         case 0b11101101:
97             while (PINC .4 == 0)
98                 ;
99             return 5;
100        case 0b11011101:
101            while (PINC .5 == 0)
102                ;
103            return 8;
104        case 0b10111101:
```

```
105                 while (PINC .6 == 0)
106                     ;
107                 return 0;
108             }
109 
110             PORTC .0 = 1;
111             PORTC .1 = 1;
112             PORTC .2 = 0;
113 
114             switch (PINC)
115             {
116             case 0b11110011:
117                 while (PINC .3 == 0)
118                     ;
119                 return 3;
120             case 0b11101011:
121                 while (PINC .4 == 0)
122                     ;
123                 return 6;
124             case 0b11011011:
125                 while (PINC .5 == 0)
126                     ;
127                 return 9;
128             case 0b10111011:
129                 while (PINC .6 == 0)
130                     ;
131                 return 11;
132             }
133         }
134 }
135 
136 // Function to initialize door
137 void initializeDoor()
138 {
139     // Set the motor pins as output
140     DDRB |= (1 << DDB3) | (1 << DDB4);
141     // Set the red LED pin as output
142     DDRB |= (1 << DDB0);
143 }
144 
145 // Function to initialize speaker
146 void initializeSpeaker()
147 {
148     // Set the speaker as an output
149     DDRD .7 = 1;
150     PORTD .7 = 1; // Set it to 1 initially
151 }
152 
153 // Function to initialize interrupts
154 void initializeIntrrupts()
155 {
```

```
156     DDRB .2 = 0;  // make button as input
157     PORTB .2 = 1; // turn on pull up resistance for INT2 intrrupt
158 
159     // actual casue INT2
160     bit_set(MCUCSR, 6);
161 
162     DDRD .2 = 0;  // make button as input
163     PORTD .2 = 1; // turn on pull up resistance for INT0 intrrupt
164 
165     // actual casue (The falling edge of INT0)
166     bit_set(MCUCR, 1);
167     bit_clr(MCUCR, 0);
168 
169     // actual casue (The falling edge of INT1)
170     bit_set(MCUCR, 3);
171     bit_clr(MCUCR, 2);
172 
173     DDRD .3 = 0;  // make button SetPC as input
174     PORTD .3 = 1; // turn on pull up resistance
175 
176     // Enable global interrupts
177 #asm("sei")
178 
179     // GICR INT1 (bit no 7) , SetPC spacific enable
180     bit_set(GICR, 7);
181 
182     // GICR INT2 (bit no 5) , open spacific enable
183     bit_set(GICR, 5);
184 
185     // GICR INT0 (bit no 6) , admin spacific enable
186     bit_set(GICR, 6);
187 }
188 
189 // Function to initialize user data in EEPROM
190 void initializeUsers()
191 {
192     unsigned int address = 0;
193     int i;
194     for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
195     {
196         EE_WriteString(address, users[i].name);
197         address += sizeof(users[i].name);
198 
199         EE_WriteString(address, users[i].id);
200         address += sizeof(users[i].id);
201 
202         EE_WriteString(address, users[i].pc);
203         address += sizeof(users[i].pc);
204     }
205 }
```

lockSysReadWrite.h

- lockSysReadWrite.h - Header file containing functions for reading and writing data to EEPROM.
- This file includes functions for reading and writing individual bytes as well as strings to EEPROM.
- The EEPROM operations are crucial for storing and retrieving persistent data such as user information, PC details, and other configuration settings.

```
#include <mega16.h>

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);

// 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;
    }
}
```

lockSysMode.h

- lockSysMode.h - Header file containing functions for different modes of the Embedded Lock System.
- This file includes functions for admin mode, setting PC mode, and open/close door mode.
- Each mode serves a specific purpose in the functionality of the lock system, such as managing user data, updating user PC information, and controlling the door's open and close operations.

```
#include <string.h>
#include "lockSysHelp.h"

void adminMode();
void setPCMode();
void openCloseDoorMode();

// 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));
```

```
30              address += sizeof(users[i].id);
31              EE_ReadString(address, admin.pc, sizeof(admin.pc));
32              adminPCAddress = address;
33              break;
34          }
35          address += sizeof(users[i].pc);
36      }
37
38      address = 0; // reset the address
39
40      displayMessage("Enter Admin PC: ", 0);
41      lcd_gotoxy(0, 1);
42
43      if (enterValueWithKeypad(enteredPC))
44      {
45
46          if (strcmp(admin.pc, enteredPC) == 0)
47          {
48              displayMessage("Enter Student ID: ", 0);
49
50              if (enterValueWithKeypad(enteredStudentID))
51              {
52                  int j;
53                  for (j = 0; j < sizeof(users) / sizeof(users[0]); ++j)
54                  {
55                      address += sizeof(users[j].name);
56                      EE_ReadString(address, student.id, sizeof(student.
                            id));
57                      address += sizeof(users[j].id);
58                      if (strcmp(student.id, enteredStudentID) == 0)
59                      {
60                          displayMessage("Enter student's new PC: ", 0);
61                          if (enterValueWithKeypad(enteredNewPC))
62                          {
63                              // Set the new pc for this student, address
                                    is for student PC
64                              EE_WriteString(address, enteredNewPC);
65                              displayMessage("Student PC is stored",
                                    1000);
66                              userFound = 1;
67                              break;
68                          }
69                      }
70                      else if (strcmp(admin.id, enteredStudentID) == 0)
71                      {
72                          displayMessage("Enter your new PC: ", 0);
73                          lcd_gotoxy(0, 1);
74                          if (enterValueWithKeypad(enteredNewPC))
75                          {
76                              // Set the new pc for this user (Admin),
                                    address is for admin PC
```

```
                                EE_WriteString(adminPCAddress, enteredNewPC
                                    );
                                displayMessage("Your PC is stored", 1000);
                                userFound = 1;
                                break;
                            }
                        }
                        address += sizeof(users[i].pc);
                    }
                }
            }
        }

        if (!userFound)
        {
            displayMessage("Contact Admin", 0);
            // Two peeps alarm
            generateTone();
            generateTone();
        }
        delay_ms(5000);
        lcd_clear();
}

// Function for set PC mode
void setPCMode()
{
        char enteredID[5]; // Change data type to string
        User currentUser;
        unsigned int address = 0;
        int userFound = 0;
        int i;
        char enteredNewPC[5];   // define enteredNewPC array to hold the
            new PC
        char reenteredNewPC[5]; // define reenteredNewPC array to hold the
            Re-entered new PC

        lcd_clear();
        displayMessage("Enter your ID:", 0);
        lcd_gotoxy(0, 1);
        if (enterValueWithKeypad(enteredID))
        {
            char enteredOldPC[5];
            // search for the entered ID in the user data
            for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
            {
                address += sizeof(users[i].name);
                EE_ReadString(address, currentUser.id, sizeof(currentUser.
                    id)); // Read ID as a string

                if (strcmp(currentUser.id, enteredID) == 0)
```

```
124             {
125                 // ID found, verify the old PC
126                 address += sizeof(currentUser.id);
127                 EE_ReadString(address, currentUser.pc, sizeof(
                        currentUser.pc)); // Read PC as a string
128                 displayMessage("Enter old PC:", 0);
129                 lcd_gotoxy(0, 1);
130 
131                 if (enterValueWithKeypad(enteredOldPC))
132                 {
133                     if (strcmp(currentUser.pc, enteredOldPC) == 0)
134                     {
135                         // Old PC verified
136                         displayMessage("Enter new PC:", 0);
137                         lcd_gotoxy(0, 1);
138                         enterValueWithKeypad(enteredNewPC);
139 
140                         lcd_clear();
141                         displayMessage("Re-enter new PC:", 0);
142                         lcd_gotoxy(0, 1);
143                         enterValueWithKeypad(reenteredNewPC);
144 
145                         if (strcmp(enteredNewPC, reenteredNewPC) == 0)
146                         {
147                             // If new PC entered correctly, store it
148                             EE_WriteString(address, enteredNewPC);
149                             displayMessage("New PC stored", 1000);
150                         }
151                         else
152                         {
153                             displayMessage("New PC mismatch, Contact
                                    admin", 1000);
154                             generateTone();
155                             generateTone();
156                         }
157                     }
158                     else
159                     {
160                         displayMessage("Wrong old PC,   Contact admin",
                                1000);
161 
162                         generateTone();
163                         generateTone();
164                     }
165                 }
166 
167                 userFound = 1;
168                 break;
169             }
170 
171             address += sizeof(users[i].id);
```

```
172             address += sizeof(users[i].pc);
173         }
174 
175         if (!userFound)
176         {
177             displayMessage("Wrong ID", 0);
178             generateTone();
179             generateTone();
180         }
181         delay_ms(5000);
182         lcd_clear();
183     }
184 }
185 
186 // Function for open/close door mode
187 void openCloseDoorMode()
188 {
189     char enteredID[4]; // Change data type to string
190     User currentUser;
191     unsigned int address = 0;
192     int userFound = 0;
193     int i;
194 
195     displayMessage("Enter your ID: ", 0);
196     lcd_gotoxy(0, 1);
197 
198     if (enterValueWithKeypad(enteredID))
199     {
200         char enteredPC[4];
201         for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
202         {
203             EE_ReadString(address, currentUser.name, sizeof(users[i].
                name));
204             address += sizeof(users[i].name);
205             EE_ReadString(address, currentUser.id, sizeof(currentUser.
                id)); // Read ID as a string
206 
207             if (strcmp(currentUser.id, enteredID) == 0)
208             {
209 
210                 address += sizeof(users[i].id);
211                 EE_ReadString(address, currentUser.pc, sizeof(
                    currentUser.pc)); // Read PC as a string
212 
213                 displayMessage("Enter your PC: ", 0);
214                 lcd_gotoxy(0, 1);
215 
216                 if (enterValueWithKeypad(enteredPC))
217                 {
218                     if (strcmp(currentUser.pc, enteredPC) == 0)
219                     {
```

```
220                         lcd_clear();
221                         lcd_puts("Welcome, ");
222                         lcd_puts(currentUser.name);
223                         openDoor();
224                         delay_ms(2000); // Wait for 2 seconds with the
                               door open
225 
226                         closeDoor();
227                         delay_ms(2000); // Wait for 2 seconds with the
                               door closed
228                     }
229                     else
230                     {
231                         displayMessage("Sorry wrong PC", 1000);
232                         // one peep alarm
233                         generateTone();
234                     }
235                 }
236                 userFound = 1;
237                 break;
238             }
239 
240             address += sizeof(users[i].id);
241             address += sizeof(users[i].pc);
242         }
243     }
244 
245     if (!userFound)
246     {
247         displayMessage("Wrong ID", 1000);
248         // Two peeps alarm
249         generateTone();
250         generateTone();
251     }
252     lcd_clear();
253 }
```

lockSysHelp.h

- lockSysHelp.h - Header file containing helper functions for the Embedded Lock System.
- This file includes functions for displaying messages on the LCD, entering values with the keypad,
- generating tones with the speaker, and controlling the motor to open and close the door.

```
1 #include <mega16.h>
2 #include <alcd.h>
3 #include <delay.h>
4 
5 void displayMessage(char *message, int delay_ms_value);
```

```
int enterValueWithKeypad(char *buffer);
void generateTone();
void openDoor();
void closeDoor();

// 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)
        lcd_putchar('#');
    else
        lcd_putchar(buffer2[2] + '0');

    buffer[0] = buffer2[0] + '0';
    buffer[1] = buffer2[1] + '0';
    buffer[2] = buffer2[2] + '0';
    buffer[3] = '\0';

    delay_ms(1000);

    return 1;
```

```
}

// Function to generate a tone with speaker
void generateTone()
{
    PORTD .7 = 1;
    delay_ms(500);
    PORTD .7 = 0;
    delay_ms(500);
    PORTD .7 = 1;
}

// Function to open the door (motor and redled)
void openDoor()
{
    // Turn on the red LED light
    PORTB |= (1 << PORTB0);

    // Motor movement for smooth opening
    PORTB &= ~(1 << PORTB3);
    delay_ms(500);
    PORTB |= (1 << PORTB4);
    delay_ms(1000);
    PORTB &= ~(1 << PORTB4);
}
// Function to open the door (motor and redled)
void closeDoor()
{
    // Turn off the red LED light
    PORTB &= ~(1 << PORTB0);

    // Motor movement for smooth closing
    PORTB |= (1 << PORTB3);
    delay_ms(500);
    PORTB |= (1 << PORTB4);
    delay_ms(1000);
    PORTB &= ~(1 << PORTB4);
    PORTB &= ~(1 << PORTB3);

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