Assiut University

Faculty of Computers & Information

Smart lock

EMPYDED SYSTEM PROJECT
ACADEMIC YEAR 2023-2024

Smart lock

The project proposes implementing a security system using the ATmega16 microcontroller for student identification and access control in a research lab.

Project Team

عبدالرحمن صلاح الدين سعد محمود IT - 162020343

رامز ناصر فايز بخيت CS - 162020217

مينا صلاح ذكي ويصا CS - 162020654

أندرو سمير بخيت غالي CS - 162020138

وعد أحمد شعبان عبدالله CS - 162020721

أسماء محمد بيومي السيد CS - 162020114

Supervisor

أ.د/ خليل اسماعيل

2023

Project Description:

The project involves designing a security system using the ATmega16 microcontroller for a research lab lock and student identification. Each student has a unique three-digit ID and a three-digit pass-code stored in EEPROM. The system handles entry requests, pass-code changes for students, and administrative tasks for the lab professor.

Assume any missing data or actions:

1-Missing Data:

- **Student Data:**
  - It is assumed that additional information, such as full names of students .
  - Additional details about each student, like their date of birth or any other relevant information that aids in distinguishing between students.
- **Additional Admin Information:**
  - We need to add more than one admin and add their full names and information about their date of birth and place of residence.

2-Missing Actions:

- The process of adding admin names.
- The process of adding student names.

- Enable the admin to delete a student.
- Enable the admin to modify the student code and new data that may be added later, such as student names.
- Enabling the student to modify his data, not just the password.
- Search for the student's name by his identification code.

Describe the needed hardware items:

1. **ATmega16 Microcontroller**
2. **Keypad (4x3)**
3. **LCD (16x2)**
4. **Push-Button for Admin**
5. **Push-Button for User**
6. **Door Locking Motor**
7. **Peep Alarm**
8. **EEPROM**

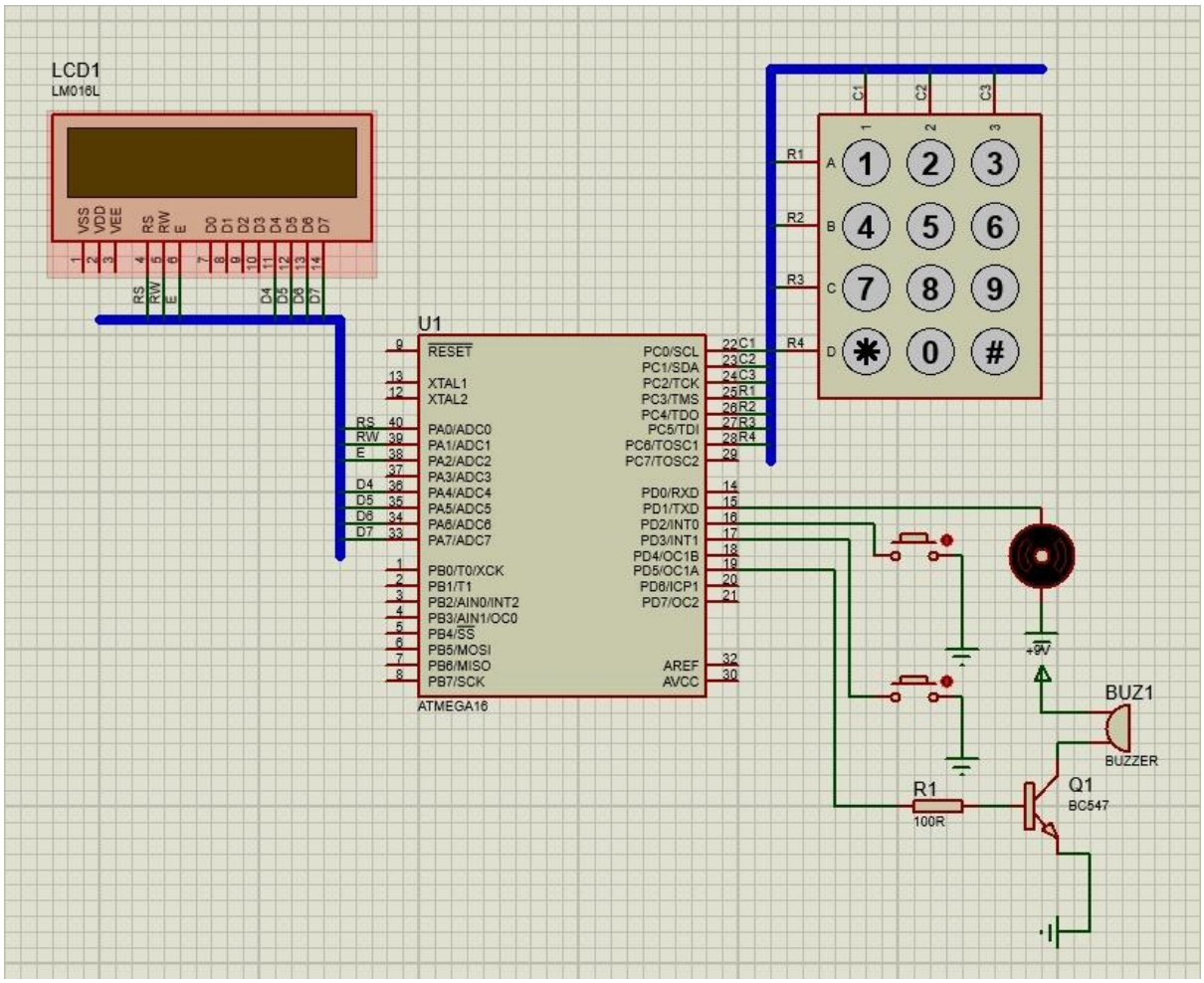
Table Connection:

Items	*ports*
Keypad (4 x 3)	*C*
Lcd (16 x 2)	*A*
Push-Button for Admin	*PD2 (INT0)*
Push-Button for User	*PD3 (INT1)*
Door Locking Motor	*PD1*
Peep Alarm	*PD5*

Draw the system operation flow chart:

Uploaded to a file alone next to this file under the name Smart lock flow chart.

Draw the system test-bench and interface circuits:



Write down the needed code (justify by comments)

```
#include <mega16.h>    // Include header file for ATmega16 microcontroller

#include <delay.h>     // Include header file for delay functions
#include <alcd.h>      // Include header file for alphanumeric LCD functions
#include <stdlib.h>    // Include standard library functions
#include <string.h>    // Include string manipulation functions
#include <stdio.h>     // Include standard I/O functions

#define bit_set(r, b) r |= (1 << b)    // Macro to set a bit in a register
#define bit_clr(r, b) r &= ~(1 << b)   // Macro to clear a bit in a register
char keypad();                          // Function prototype for reading
keypad input
void ChangePasswordUser();              // Function prototype for changing
user password
void ChangePasswordAdmin();             // Function prototype for changing
admin password
void EE_Write(unsigned int add, unsigned char data);    // Function prototype
to write to EEPROM
unsigned char EE_Read(unsigned int add);          // Function prototype to
read from EEPROM
unsigned int storedPassword = 0;                 // Variable to store password
unsigned int OldPassword = 0;                    // Variable to store old
password
unsigned int NewPassword = 0;                    // Variable to store new
password
unsigned int ReenterNewPassword = 0;             // Variable to re-enter new
password
unsigned int ChangeAdminPasswords = 0;           // Variable for changing
admin passwords
unsigned int pass1;             // Variable for password digit 1
unsigned int pass2;             // Variable for password digit 2
unsigned int pass3;             // Variable for password digit 3
unsigned int password;          // Variable to store entered password
unsigned int id1;               // Variable for ID digit 1
unsigned int id2;               // Variable for ID digit 2
unsigned int id3;               // Variable for ID digit 3
unsigned int enteredID;         // Variable to store entered ID
unsigned int NewID = 0 ;        // Variable to store new ID
void initializeEEPROM()
{
 // Function to initialize EEPROM with default values
 // Default passwords
```

```
    unsigned int defaultPassword1 = 203;
    unsigned int defaultPassword2 = 129;
    unsigned int defaultPassword3 = 700;
    unsigned int defaultPassword4 = 426;
    unsigned int defaultPassword5 = 79;
        // Writing default passwords to specific EEPROM addresses
    EE_Write(111, defaultPassword1 % 255);
    EE_Write(112, defaultPassword1 / 255);
    EE_Write(126, defaultPassword2 % 255);
    EE_Write(127, defaultPassword2 / 255);
    EE_Write(128, defaultPassword3 % 255);
    EE_Write(129, defaultPassword3 / 255);
    EE_Write(130, defaultPassword4 % 255);
    EE_Write(131, defaultPassword4 / 255);
    EE_Write(132, defaultPassword5 % 255);
    EE_Write(133, defaultPassword5 / 255);
    // Add more passwords if needed
}

void main(void)
{
 // Setting Port C for keypad input and output configurations
    DDRC = 0b00000111; // 1 pin unused, 4 rows (input), 3 columns (output)
    // Setting internal pull-up resistances for keypad pins
    PORTC = 0b11111000; // pull-up  resistance  to avoid floating  for
keypad
     // Setting direction and initial state for different pins on Port D
    DDRD.2 = 0;      // Configuring INT0 (Admin) as input
    PORTD.2 = 1;    // Enabling pull-up resistor for INT0
    DDRD.3 = 0;     // Configuring INT1 (Set PC for user) as input
    PORTD.3 = 1;    // Enabling pull-up resistor for INT1
    DDRD.1 = 1;    // Configuring Motor pin as output
    PORTD.1 = 0;   // Setting Motor pin to LOW initially
    DDRD.5 = 1;   // Configuring Alarm pin as output
    PORTD.5 = 0;  // Setting Alarm pin to LOW initially
  // Setting up External Interrupt 0 (INT0)
    bit_set(MCUCR, 1);  //MCUCR |= (1<<1)
    bit_clr(MCUCR, 0);  //MCUCR &= ~(1<<0)
   // Setting up External Interrupt 1 (INT1)
    bit_set(MCUCR, 3);
    bit_clr(MCUCR, 2);
#asm("sei");      // Set Enable Interrupt (Global Interrupt Enable)
    bit_set(GICR, 6);  // Enable external interrupt 0 (INT0)
    bit_set(GICR, 7);   // Enable external interrupt 1 (INT1)
    lcd_init(16);      // Important to initialize the LCD, Give it the number
of characters per line
```

```
    initializeEEPROM(); // Initialize EEPROM with default values  (call only
once for  initializing)
    while (1)
        {
        // Application code loop
        //Please write your application code here

        // Display message prompting for '*' key entry
        lcd_clear();
        lcd_printf("Press * to enter");
         // Wait until '*' key is pressed
        while (keypad() != 10);
        lcd_clear();
          // Clear LCD and display "Entered ID:"
        lcd_printf("Entered ID:");
        // Reading ID digits from keypad input
        id1 = keypad();
        id2 = keypad();
        id3 = keypad();
        enteredID = id3 + (id2 * 10) + (id1 * 100);
        lcd_gotoxy(0, 1);
          // Display entered ID on the LCD
        lcd_printf("%u", enteredID);
        delay_ms(1000);
        //lcd_clear();
         // Check entered ID and process accordingly
         // Check if the enteredID matches predefined IDs
        if (enteredID == 111 || enteredID == 126 || enteredID == 128 ||
enteredID == 130 || enteredID == 132 )    // enteredID == 111 || enteredID ==
222 || enteredID == 333 || enteredID == 444 || enteredID == 555
            {
            // Clear the LCD and prompt for password entry
            lcd_clear();
            lcd_printf("Enter Password:\n");
            lcd_gotoxy(0, 1);
            // Read the three digits of the password from the keypad
            pass1 = keypad();
            pass2 = keypad();
            pass3 = keypad();
               // Combine the entered password digits into a single password
value
            password = (pass1 * 100) + (pass2 * 10) + (pass3 * 1);
            lcd_printf("%u", password);
                // Retrieve stored password from EEPROM based on enteredID
            storedPassword = EE_Read(enteredID);
            storedPassword = storedPassword + (EE_Read(enteredID + 1) * 255);
            //lcd_printf("%u", storedPassword);
```

```
            delay_ms(1000);
              // Check if the entered password matches the stored password for
a specific enteredID
            if (password == storedPassword && enteredID == 111)
                {
                 // Display admin authentication message
                lcd_clear();
                lcd_printf("You are Admin");
                delay_ms(1000);
                lcd_clear();
                lcd_gotoxy(5, 0);
                 // (Code for displaying admin welcome message and actions)
                lcd_printf("Welcome!");
                lcd_gotoxy(7, 1);
                lcd_printf("Prof");
                delay_ms(1000);
                lcd_clear();
                PORTD.1 = 1;
                lcd_printf("Door is opening");
                lcd_gotoxy(0, 1);
                lcd_clear();
                lcd_printf("Press * to enter");
                lcd_gotoxy(0, 1);
                lcd_printf("Press # to Exit");
                 delay_ms(1000);
                    if(  keypad() == 10){

                           lcd_clear();
                    ChangePasswordAdmin();
                    } else{
                      lcd_clear();
                      PORTD.1 = 0;
                          continue;
                    }
                }
            else if (password == storedPassword && enteredID == 126)
                {
                 // Display user authentication message for ID 126
                lcd_clear();
                lcd_gotoxy(5, 0);
                // (Code for displaying user welcome message and actions for
ID 126)
                lcd_printf("Welcome!");
                lcd_gotoxy(7, 1);
                lcd_printf("Mina");
                delay_ms(1000);
                lcd_clear();
```

```
            lcd_printf("Door is opening");
            PORTD.1 = 1;
            lcd_clear();
            lcd_printf("Press * to enter");
            lcd_gotoxy(0, 1);
            lcd_printf("Press # to Exit");
             delay_ms(1000);
                if(  keypad() == 10){
                       lcd_clear();
                        ChangePasswordUser()  ;

                } else{
                  lcd_clear();
                  PORTD.1 = 0;
                      continue;
                }
            }
        else if (password == storedPassword && enteredID == 128)
            {
            lcd_clear();
            lcd_gotoxy(5, 0);
            lcd_printf("Welcome!");
            lcd_gotoxy(7, 1);
            lcd_printf("Abdo");
            delay_ms(1000);
            lcd_clear();
            lcd_printf("Door is opening");
            PORTD.1 = 1;
              lcd_clear();
            lcd_printf("Press * to enter");
            lcd_gotoxy(0, 1);
            lcd_printf("Press # to Exit");
             delay_ms(1000);
                if(  keypad() == 10){
                       lcd_clear();
                        ChangePasswordUser()  ;

                } else{
                  lcd_clear();
                  PORTD.1 = 0;
                      continue;
                }
            }
        else if (password == storedPassword && enteredID == 130)
            {
            lcd_clear();
            lcd_gotoxy(5, 0);
```

```
            lcd_printf("Welcome!");
            lcd_gotoxy(7, 1);
            lcd_printf("Salah");
            delay_ms(1000);
            lcd_clear();
            lcd_printf("Door is opening");

            PORTD.1 = 1;
              lcd_clear();
            lcd_printf("Press * to enter");
            lcd_gotoxy(0, 1);
            lcd_printf("Press # to Exit");
             delay_ms(1000);
                if(  keypad() == 10){
                       lcd_clear();
                        ChangePasswordUser()  ;

                } else{
                  lcd_clear();
                  PORTD.1 = 0;
                      continue;
                }
            }
        else if (password == storedPassword && enteredID == 132)
            {
            lcd_clear();
            lcd_gotoxy(5, 0);
            lcd_printf("Welcome!");
            lcd_gotoxy(7, 1);
            lcd_printf("Zaki");
            delay_ms(1000);
            lcd_clear();
            lcd_printf("Door is opening");
            PORTD.1 = 1;
              lcd_clear();
            lcd_printf("Press * to enter");
            lcd_gotoxy(0, 1);
            lcd_printf("Press # to Exit");
             delay_ms(1000);
                if(  keypad() == 10){
                   lcd_clear();
                   ChangePasswordUser()  ;

                } else{
                  lcd_clear();
                  PORTD.1 = 0;
                      continue;
```

```
                    }
                }
            else
                {
                lcd_clear();
                lcd_printf("Wrong password");
             // Activate alarm
                PORTD.5 = 1;
                delay_ms(1000);// Wait for 1 second
                PORTD.5 = 0;
                continue;   // Restart the loop to re-enter a valid password
                }
            }
        else
            {
            // Handling the case of an invalid ID
            lcd_clear();
            lcd_printf("Invalid ID");
            // Activate alarm in a specific pattern
           // (Code for activating alarm for an invalid ID)
            PORTD.5 = 1;
            delay_ms(1000); // Wait for 1 second
            PORTD.5 = 0;
            delay_ms(1000);
            PORTD.5 = 1;
            delay_ms(1000); // Wait for 1 second
            PORTD.5 = 0;
            delay_ms(1000);
            continue;       // Restart the loop to re-enter a valid ID
            }


        }


}

char keypad()
{
    while (1)  // Infinite loop to continuously check keypad input
        {
         // Activate column 1 and deactivate other columns
        PORTC.0 = 0; // column 1 is activated by 0
        PORTC.1 = 1; // column 2 is inactive by 1
        PORTC.2 = 1; // column 3 is inactive by 1
        // Switch case to check the row values based on the pressed key in
column 1
```

```
        switch (PINC)
            {
             // Check for specific row combinations in column 1
            // 0bxrrrrccc
            case 0b11110110:  // If row combination matches, indicating a
keypress
                while (PINC.3 == 0);// Wait for key release
                return 1;  // Return the value 1 corresponding to the pressed
key
                break;     // Exit the switch case
            case 0b11101110:
                while (PINC.4 == 0);
                return 4;
                break;
            case 0b11011110:
                while (PINC.5 == 0);
                return 7;
                break;
            case 0b10111110:
                while (PINC.6 == 0);
                return 10; // '*' corresponds to number 10
                break;
            }
            // Deactivate column 1 and activate column 2
        PORTC.0 = 1; // column 1 is inactive by 1
        PORTC.1 = 0; // column 2 is activated by 0
        PORTC.2 = 1; // column 3 is inactive by 1
        switch (PINC)
            {
            // 0bxrrrrccc
            case 0b11110101:
                while (PINC.3 == 0);
                return 2;
                break;
            case 0b11101101:
                while (PINC.4 == 0);
                return 5;
                break;
            case 0b11011101:
                while (PINC.5 == 0);
                return 8;
                break;
            case 0b10111101:
                while (PINC.6 == 0);
                return 0;
                break;
            }
```

```
            // Deactivate column 2 and activate column 3
        PORTC.0 = 1; // column 1 is inactive by 1
        PORTC.1 = 1; // column 2 is inactive by 1
        PORTC.2 = 0; // column 3 is activated by 0
        switch (PINC)
            {
            // 0bxrrrrccc
            case 0b11110011:
                while (PINC.3 == 0);
                return 3;
                break;
            case 0b11101011:
                while (PINC.4 == 0);
                return 6;
                break;
            case 0b11011011:
                while (PINC.5 == 0);
                return 9;
                break;
            case 0b10111011:
                while (PINC.6 == 0);
                return 11;
                break;
            }
        }
}

unsigned char EE_Read(unsigned int add)
{
    while(EECR.1 == 1);     //Wait till EEPROM is ready
    EEAR = add;         //Prepare the address you want to read from

    EECR.0 = 1;             //Execute read command

    return EEDR;   // Return the data read from the EEPROM

}

void EE_Write(unsigned int add, unsigned char data)
{
    while(EECR.1 == 1);     //Wait till EEPROM is ready
    EEAR = add;         //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
```

```
}

void ChangePasswordUser()
{
    lcd_clear();  // Clear the LCD display
    lcd_printf("Enter_ID");  // Display "Enter_ID" on the LCD
    NewID = (keypad() * 100) + (keypad() * 10) + keypad();  // Collect a new
ID from the keypad
    lcd_gotoxy(0, 1);  // Set cursor to the second line of the LCD
    lcd_printf("%u", NewID);  // Display the entered ID on the LCD
    delay_ms(1000);  // Delay for 1 second

    if (NewID == 126 || NewID == 128 || NewID == 130 || NewID == 132) {  //
Check if the entered ID is valid
        lcd_clear();  // Clear the LCD display
        lcd_printf("Enter Old-PC");  // Display "Enter Old-PC" on the LCD
        lcd_gotoxy(0, 1);  // Set cursor to the second line of the LCD
        OldPassword = 0;  // Initialize the variable for the old password
        OldPassword = (keypad() * 100) + (keypad() * 10) + keypad();  //
Collect the old password from the keypad
        storedPassword = EE_Read(NewID);  // Read the stored password from
EEPROM at the given ID
        storedPassword = storedPassword + (EE_Read(NewID + 1) * 255);  // Read
the second byte of the stored password
        lcd_printf("%u", OldPassword);  // Display the entered old password on
the LCD
        delay_ms(1000);  // Delay for 1 second

        if (OldPassword == storedPassword) {  // Check if the entered old
password matches the stored one
            // Prompt to enter the new password
            lcd_clear();
            lcd_printf("Enter New-PC");
            lcd_gotoxy(0, 1);

            NewPassword = (keypad() * 100) + (keypad() * 10) + keypad();  //
Collect the new password from the keypad
            lcd_printf("%u", NewPassword);  // Display the entered new
password on the LCD
            delay_ms(1000);  // Delay for 1 second

            lcd_clear();  // Clear the LCD display
            lcd_printf("Re-enter PC");  // Prompt to re-enter the new password
            lcd_gotoxy(0, 1);

            ReenterNewPassword = (keypad() * 100) + (keypad() * 10) +
keypad();  // Collect the re-entered new password
```

```
            lcd_printf("%u", ReenterNewPassword);  // Display the re-entered
new password on the LCD
            delay_ms(1000);  // Delay for 1 second

            if (ReenterNewPassword == NewPassword) {  // Check if the re-
entered new password matches the new password
                lcd_clear();  // Clear the LCD display
                // Write the new password to EEPROM
                EE_Write(NewID, NewPassword % 255);
                EE_Write(NewID + 1, NewPassword / 255);
                lcd_printf("Change");  // Display "Change" on the LCD
                lcd_gotoxy(0, 1);
                lcd_printf("Successfully");  // Display "Successfully" on the
LCD
                delay_ms(1000);  // Delay for 1 second
                lcd_clear();  // Clear the LCD display
                lcd_printf("Press * to enter");  // Prompt to press '*' to
enter
            } else {
                lcd_clear();  // Clear the LCD display
                lcd_printf("Wrong password");  // Display "Wrong password" on
the LCD
                delay_ms(1000);  // Delay for 1 second
                PORTD.5 = 1;  // Activate alarm
                delay_ms(1000);
                PORTD.5 = 0;  // Deactivate alarm
                delay_ms(1000);
                lcd_clear();  // Clear the LCD display
                lcd_printf("Press * to enter");  // Prompt to press '*' to
enter
                delay_ms(1000);  // Delay for 1 second
            }
        } else {
            lcd_clear();  // Clear the LCD display
            lcd_printf("Wrong password");  // Display "Wrong password" on the
LCD
            delay_ms(1000);  // Delay for 1 second
            PORTD.5 = 1;  // Activate alarm
            delay_ms(1000);
            PORTD.5 = 0;  // Deactivate alarm
            delay_ms(1000);
            lcd_clear();  // Clear the LCD display
            lcd_printf("Press * to enter");  // Prompt to press '*' to enter
            delay_ms(1000);  // Delay for 1 second
        }
    } else {
        lcd_clear();  // Clear the LCD display
```

```
        lcd_printf("Invalid ID");  // Display "Invalid ID" on the LCD
        PORTD.5 = 1;  // Activate alarm
        delay_ms(1000);  // Delay for 1 second
        PORTD.5 = 0;  // Deactivate alarm
        delay_ms(1000);  // Delay for 1 second
        PORTD.5 = 1;  // Activate alarm
        delay_ms(1000);  // Delay for 1 second
        PORTD.5 = 0;  // Deactivate alarm
        delay_ms(1000);  // Delay for 1 second
        lcd_clear();  // Clear the LCD display
        lcd_printf("Press * to enter");  // Prompt to press '*' to enter
        delay_ms(1000);  // Delay for 1 second
    }
}

void ChangePasswordAdmin()
{
    lcd_clear();  // Clear the LCD display
    lcd_printf("You are Admin");  // Display "You are Admin" on the LCD
    delay_ms(1000);  // Delay for 1 second
    lcd_clear();  // Clear the LCD display
    lcd_printf("Enter PC: ");  // Prompt to enter the password
    lcd_gotoxy(0, 1);  // Set cursor to the second line of the LCD
    OldPassword = (keypad() * 100) + (keypad() * 10) + keypad();  // Collect
the entered password
    storedPassword = EE_Read(111);  // Read the stored password for Admin from
EEPROM
    storedPassword = storedPassword + (EE_Read(112) * 255);  // Read the
second byte of the stored password
    lcd_printf("%u", OldPassword);  // Display the entered password on the LCD
    delay_ms(1000);  // Delay for 1 second

    if (storedPassword == OldPassword) {  // Check if the entered password
matches the stored Admin password
        lcd_clear();  // Clear the LCD display
        lcd_printf("Entered ID:");  // Prompt to enter the ID
        lcd_gotoxy(0, 1);  // Set cursor to the second line of the LCD
        id1 = keypad();  // Read the first digit of the ID
        id2 = keypad();  // Read the second digit of the ID
        id3 = keypad();  // Read the third digit of the ID
        NewID = id3 + (id2 * 10) + (id1 * 100);  // Calculate the new ID from
the entered digits

        if (NewID == 111 || NewID == 126 || NewID == 128 || NewID == 130 ||
NewID == 132) {  // Check if the new ID is valid
            lcd_printf("%u", NewID);  // Display the new ID on the LCD
            delay_ms(1000);  // Delay for 1 second
```

```
            lcd_clear();  // Clear the LCD display
            lcd_printf("Enter-new PC: ");  // Prompt to enter the new password
            lcd_gotoxy(0, 1);  // Set cursor to the second line of the LCD
            ChangeAdminPasswords = (keypad() * 100) + (keypad() * 10) +
keypad();  // Collect the new password
            lcd_printf("%u", ChangeAdminPasswords);  // Display the entered
new password on the LCD
            delay_ms(1000);  // Delay for 1 second
            lcd_clear();  // Clear the LCD display
            EE_Write(NewID, ChangeAdminPasswords % 255);  // Write the lower
byte of the new password to EEPROM
            EE_Write(NewID + 1, ChangeAdminPasswords / 255);  // Write the
upper byte of the new password to EEPROM
            lcd_printf("Change");  // Display "Change" on the LCD
            lcd_gotoxy(0, 1);  // Set cursor to the second line of the LCD
            lcd_printf("Successfully");  // Display "Successfully" on the LCD
            delay_ms(1000);  // Delay for 1 second
            lcd_clear();  // Clear the LCD display
            lcd_printf("Press * to enter");  // Prompt to press '*' to enter
        } else {
            lcd_clear();  // Clear the LCD display
            lcd_printf("Invalid ID");  // Display "Invalid ID" on the LCD
            PORTD.5 = 1;  // Activate alarm
            delay_ms(1000);  // Delay for 1 second
            PORTD.5 = 0;  // Deactivate alarm
            delay_ms(1000);  // Delay for 1 second
            PORTD.5 = 1;  // Activate alarm
            delay_ms(1000);  // Delay for 1 second
            PORTD.5 = 0;  // Deactivate alarm
            delay_ms(1000);  // Delay for 1 second
            lcd_clear();  // Clear the LCD display
            lcd_printf("Press * to enter");  // Prompt to press '*' to enter
            delay_ms(1000);  // Delay for 1 second
        }
    } else {
        lcd_clear();  // Clear the LCD display
        lcd_printf("Wrong password");  // Display "Wrong password" on the LCD
        delay_ms(1000);  // Delay for 1 second
        PORTD.5 = 1;  // Activate alarm
        delay_ms(1000);  // Delay for 1 second
        PORTD.5 = 0;  // Deactivate alarm
        delay_ms(1000);  // Delay for 1 second
        lcd_clear();  // Clear the LCD display
        delay_ms(1000);  // Delay for 1 second
        ChangePasswordAdmin();  // Call the function recursively to re-enter
the password
    }
```

```
}

// Admin  High priority

interrupt [2] void ext0(void)
{
    ChangePasswordAdmin();   // Call the function to change the password for
Admin when Interrupt 0 (external interrupt 0) occurs
}
// Set PC
interrupt [3] void  ext1(void)
{
    ChangePasswordUser( ); // Call the function to change the password for
User when Interrupt 1 (external interrupt 1) occurs
}
```

What do you think about the priorities of used interrupts ? Should these remain as mentioned? Why?

Interrupt Priorities:

1. **INT0 (Push-Button for Admin):**
    - **Priority:** High
    - **Justification:** Admin operations are typically critical and may involve security-related tasks. Assigning a high priority ensures quick response and execution of admin functions.
2. **INT1 (Push-Button for User):**
    - **Priority:** Medium
    - **Justification:** User operations, while important, may not be as time-critical as admin tasks. A moderate priority allows for

timely response without compromising admin-related functions.

Additional Considerations:

- **Admin vs. User Priority:**
    - Admin-related interrupts usually take precedence over user-related interrupts due to the critical nature of admin functions, especially in a security system.
- **Testing and Adjustments:**
    - Priorities may need to be adjusted based on system testing and real-world performance. Regular testing helps ensure that interrupt priorities meet the system's responsiveness requirements.
- **Security and Safety:**
    - Prioritizing admin functions with a higher priority level helps maintain the security and integrity of the system. It ensures that admin commands are processed promptly.
- **User Experience:**
    - Prioritizing user-related interrupts appropriately ensures a responsive and user-friendly interface, providing a seamless experience for individuals interacting with the system.