**TASK\_21** \_\_*Digital door lock*

This project revolves around building a password-protected security system that uses an 8051 microcontroller. The system allows users to generate, store, and enter passwords, which are validated against an EEPROM-stored value. The correct password unlocks a door controlled by a servo motor, while incorrect attempts trigger warnings or system lockouts. It also includes features like changing the password and limiting the number of incorrect attempts.

Let’s break down the key components of this project:

**1. Components Overview:**

* **Microcontroller (8051)**: It is the brain of the system, handling all operations like reading the keypad, controlling the LCD, servo motor, and managing EEPROM data.
* **LCD Display**: Displays information to the user, such as prompts to enter the password, door status, error messages, etc.
* **Keypad**: A matrix keypad that allows the user to input passwords and other commands.
* **Servo Motor**: This is used to physically control the door, either locking or unlocking it based on the password verification.
* **EEPROM**: Non-volatile memory used to store the password so that it remains intact even when the system is powered off.
* **Buzzer**: Emits beeps for feedback, such as when a key is pressed, the password is incorrect, or the system is locked.

**2. Main Project Logic (MAIN.C):**

This file contains the main logic of the system and controls the interaction between components.

**State Machine**

The project uses a **state machine** to manage the different states of the system:

* **Generate Password (generate\_password)**: The system prompts the user to generate a new password. If no password is stored in the EEPROM (checked by reading the EEPROM value at a specific address), the system enters this state.
* **Enter Password (enter\_password)**: Once a password is stored in the EEPROM, the system prompts the user to enter the password for unlocking the door.
* **Wrong Password (wrong\_password)**: If the entered password is incorrect, the system informs the user and increments a counter to track the number of failed attempts.
* **New Password (new\_password)**: If the user wants to change the password, they first have to enter the old one correctly. If successful, the system allows them to generate a new password.

**Servo Motor Control (Door Lock Mechanism)**

The servo motor is used to simulate a door lock. The motor angle is adjusted based on the password verification result:

* **Unlock Door**: The servo motor rotates to 0 degrees, simulating unlocking the door.
* **Lock Door**: The servo motor rotates to 180 degrees, simulating locking the door.

The set\_duty\_cycle function controls the angle of the servo motor by adjusting the duty cycle of a PWM signal.

**Password Generation and Storage**

* When generating a password, the user inputs a 6-digit number using the keypad. This password is stored in the EEPROM at specific addresses, ensuring that it persists even if the system is powered off.
* The EEPROM stores each digit of the password, and the system later reads it back for comparison.

**Password Entry and Validation**

* The user is prompted to enter the password, and each digit is displayed on the LCD as it is input.
* The entered password is compared against the stored password in the EEPROM. If it matches, the door unlocks. Otherwise, the system tracks incorrect attempts, and after too many failures, the system disables itself.

**Error Handling and System Lockout**

* If the user enters the wrong password more than 5 times, the system disables further input to prevent tampering. The EEPROM flag for this is set, and the system displays a "System Disabled" message on the LCD.
* When entering a new password, the user must first enter the old password correctly. If the old password is incorrect, the user is given 3 attempts before the system locks.

**3. LCD Display Handling (LCD.C)**

The LCD is a 16x2 character display used to show prompts and information to the user.

* **Initialization (lcd\_init)**: The LCD is initialized with a set of commands to configure it for 2-line mode and to clear the display.
* **Commands**: The lcd\_command function sends specific commands to the LCD, such as clearing the display, setting the cursor position, and turning the display on/off.
* **Character and String Display**: The system can display individual characters (lcd\_char) or entire strings (lcd\_string) on the LCD. It can also display numbers by converting them to characters (lcd\_shownum).
* **Cursor Position**: The cursor can be positioned anywhere on the display using the lcd\_set\_cursor function, which calculates the correct memory address for each position.

**LCD Interaction Example:**

* When the user is prompted to enter a password, the LCD will display "Enter Password".
* As the user presses keys on the keypad, the corresponding characters (asterisks or numbers) are displayed on the screen.

**4. Keypad Input Handling (KEYPAD.C)**

The keypad is a 4x4 matrix used for user input.

* **Key Detection**: The rows and columns of the keypad are scanned to detect which button is pressed. This is done by grounding each row and checking the status of the columns. Based on the row-column combination, the pressed key is determined.
* **Feedback (Beep)**: When a key is pressed, the system provides an audible beep for feedback. The Beep function controls the duration of the beep.
* **Input Mapping**: The keys are mapped to digits (0-9) and some special characters like \* (used for cancel or reset) and #.

**5. EEPROM Handling (EEPROM.C)**

EEPROM is used for persistent storage of the password.

* **Writing to EEPROM**: The EEPROM\_write function takes an address and a value and writes the value to that address. This allows the system to store each digit of the password in separate addresses.
* **Reading from EEPROM**: The EEPROM\_read function retrieves the stored value from a specified address. This allows the system to read the stored password and compare it against the user input.
* **I2C Protocol**: The EEPROM is likely interfaced using the I2C protocol, with SDA (Serial Data) and SCL (Serial Clock) lines handling communication. Functions like start\_data, sendbyte, and acknowledge handle the low-level I2C communication protocol.

**6. Timer and Delay Functions**

Timers are used for managing time delays and controlling the PWM signal for the servo motor.

* **Delays**: Functions like Delay1 and Timmer\_Delay are used to introduce time delays in the system. These are needed for various operations like debouncing the keypad or controlling the servo motor.
* **Timer Initialization**: The timer\_init function configures the microcontroller’s timer for generating PWM signals.

**7. System Workflow**

1. **Power On**: When powered on, the system checks if a password is stored in the EEPROM.
2. **Password Generation**: If no password exists, the system enters the password generation mode and prompts the user to input a 6-digit password.
3. **Password Entry**: If a password exists, the system prompts the user to enter it. The entered password is compared against the stored password in EEPROM.
4. **Door Control**: If the password is correct, the door unlocks for 10 seconds. If incorrect, the system warns the user and tracks failed attempts.
5. **New Password**: The user can enter a new password by first correctly inputting the old password.
6. **System Lockout**: After too many incorrect attempts, the system locks out, and the user is unable to input further passwords.

**8. Error Handling**

* **Invalid Input**: If the user presses an invalid key or the wrong key, the system provides feedback through the LCD and buzzer.
* **Wrong Password**: If the user enters the wrong password, they are given up to 5 attempts before the system disables further input.
* **System Disabled**: If the system detects too many failed password attempts, it sets a flag in the EEPROM and prevents further use until reset.

**Conclusion**

This project effectively demonstrates how to design an embedded system using the 8051 microcontroller for password-protected access control. It integrates various peripherals such as an LCD, keypad, servo motor, and EEPROM. The system is designed with a user-friendly interface, error handling, and security features like limiting incorrect password attempts and allowing for password changes. This kind of system can be applied in various security applications such as door locks, safes, and restricted access areas.

**Main Features:**

1. **Password Generation and Validation**:
   * The system generates a random 6-digit password, which is saved in EEPROM.
   * The user must enter this password to unlock the door.
   * The system compares the entered password with the one stored in EEPROM.
2. **EEPROM Memory**:
   * The system uses EEPROM to store the generated password persistently, even after power-off.
   * The password is saved at specific addresses, and the user input is compared against it.
3. **Servo Motor Control**:
   * A servo motor (connected to P2^0) is controlled using PWM to simulate door lock/unlock operations.
   * The door opens when the correct password is entered and closes after a set delay.
4. **State Machine**:
   * The system follows a state machine model, with states like generate\_password, enter\_password, wrong\_password, and new\_password.
5. **Beep Notification**:
   * The system uses a beep sound for feedback when buttons are pressed or an error (wrong password) occurs.
6. **Keypad Input**:
   * The system takes input via a 4x4 matrix keypad. It reads digits entered by the user and processes them to form the password.
7. **LCD Display**:
   * The system uses an LCD to display prompts like "Generate Pin," "Enter Password," and error messages such as "Wrong Password" or "System Disabled."

**Key Functionalities:**

1. **Password Generation**:
   * randompassword() generates and displays a password on the LCD using the keypad. This password is stored in EEPROM for future validation.
2. **Password Entry**:
   * In the enter\_password state, the system prompts the user to enter the password using the keypad. The entered password is compared to the one stored in EEPROM.
3. **Password Validation**:
   * If the entered password matches, the door unlocks. If not, the system transitions to the wrong\_password state, allowing up to 5 incorrect attempts before disabling the system.
4. **Password Reset**:
   * In the new\_password state, the user is prompted to enter the old password to reset it. After successful validation, the system generates a new password.

**Summary of Code Files:**

* **MAIN.C**: Implements the main state machine and overall system logic for password handling and servo control.
* **LCD.C and LCD.H**: Handles the LCD display for displaying messages, numbers, and user inputs.
* **KEYPAD.C and KEYPAD.H**: Implements keypad input handling and a buzzer for user feedback.
* **EEPROM.C and EEPROM.H**: Provides EEPROM read/write functions to store and retrieve passwords persistently.
* **REG51.H**: Defines special function registers (SFRs) for the 8051 microcontroller.

**Flow of Operations:**

1. **Generate a New Password**:
   * If no password exists, the system generates a new 6-digit password using the keypad.
   * The password is saved in EEPROM.
2. **Enter the Password**:
   * The user is prompted to enter the password via the keypad.
   * The entered password is compared with the stored password in EEPROM.
3. **Unlock the Door**:
   * If the password is correct, the servo motor is triggered to unlock the door.
   * The system waits for 10 seconds before locking the door again.
4. **Handle Wrong Password**:
   * If the entered password is incorrect, the system increments a failure counter.
   * After 5 wrong attempts, the system disables itself, requiring a reset to regain functionality.
5. **Reset Password**:
   * The user can reset the password by entering the old one. If valid, a new password is generated.

This system is designed to provide security with a simple password-protected door lock controlled by an 8051 microcontroller.