perplexity

Automated Locker Full Document

Date: Wednesday, May 14, 2025, 6:22 AM EET

1. Project Overview

The Automated Locker project is a secure access system that uses an Arduino microcontroller to control a physical locker. The locker is locked or unlocked by a servo motor and requires a 4-digit password entered via a 4×4 keypad. The system provides user feedback through a 20×4 I2C LCD and a buzzer for audio signals. It includes security features such as input masking, lockout after multiple failed attempts, and an alarm if the locker is left open too long.

2. Hardware Components and Connections

Component	Arduino Pin(s)	Description
Arduino UNO	-	Main controller
4×4 Keypad	2,3,4,5 (Rows)	Keypad rows connected to pins 2-5
	6,7,8,9 (Cols)	Keypad columns connected to pins 6-9
I2C LCD (20×4)	A4 (SDA)	I2C data line
	A5 (SCL)	I2C clock line
Servo Motor	10	Servo control signal
Buzzer	11	Buzzer output
Power & Ground	5V, GND	Power supply and ground connections

Notes:

- The keypad is a matrix with 4 rows and 4 columns, wired to digital pins 2–9.
- The LCD uses the I2C interface, connected to analog pins A4 (SDA) and A5 (SCL).
- The servo motor controls the locking mechanism by rotating between locked (0°) and unlocked (100°) positions.
- The buzzer provides audio feedback for success, error, and alarm conditions.

3. Software: Code Explanation

The code is written in Arduino C++ and organized into initialization, main loop, and several helper functions.

3.1 Library Inclusions and Definitions

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <Servo.h>

#define buzzer 11
```

- **Wire.h**: Enables I2C communication for the LCD.
- **LiquidCrystal_I2C.h**: Controls the LCD via I2C.
- **Keypad.h**: Manages keypad scanning and key detection.
- **Servo.h**: Controls the servo motor.
- `buzzer` is assigned to digital pin 11.

3.2 Hardware Objects and Keypad Setup

```
LiquidCrystal_I2C lcd(0x27, 20, 4);
Servo myservo;

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte rowPins[ROWS] = {2, 3, 4, 5};
byte colPins[COLS] = {6, 7, 8, 9};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
```

- Creates an LCD object at I2C address `0x27` with 20 columns and 4 rows.
- Creates a servo object.
- Defines keypad layout and maps physical pins for rows and columns.
- Initializes keypad scanning.

3.3 State Variables

```
String password = "1234";
String input = "";
byte maxPasswordLength = 4;
byte currentPasswordLength = 0;
byte wrongAttempts = 0;
bool doorlocked = true;
byte cursorPosition = 5;
unsigned long doorOpenTime = 0;
const unsigned long OPENALARM = 120000;
```

- `password`: The correct password.
- `input`: Stores user-entered digits.
- `maxPasswordLength`: Password length (4 digits).
- `currentPasswordLength`: Number of digits entered so far.
- `wrongAttempts`: Counts consecutive failed attempts.
- `doorlocked`: Tracks if door is locked (`true`) or unlocked (`false`).
- `cursorPosition`: LCD cursor position for password input masking.
- `doorOpenTime`: Timestamp when door was last opened.
- `OPENALARM`: Time in milliseconds (2 minutes) before open-door alarm triggers.

3.4 `setup()` Function

```
void setup() {
  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);
  myservo.attach(10);
  myservo.write(0);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Enter Password:");
}
```

- Initializes serial communication (optional for debugging).
- Sets buzzer pin as output.
- Attaches servo to pin 10 and moves it to locked position (0°).
- Initializes LCD and turns on backlight.
- Displays initial prompt "Enter Password:".

3.5 `loop()` Function

```
void loop() {
  if (!doorlocked && (millis() - doorOpenTime >= OPENALARM)) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Close Door!");
    lcd.setCursor(0, 1);
    lcd.print("Press # to lock");

    while (!doorlocked) {
      digitalWrite(buzzer, HIGH);
      delay(200);
      digitalWrite(buzzer, LOW);
      delay(200);

      char key = keypad.getKey();
      if (key == '#') {
        closeDoor();
      }
    }
  }

  char key = keypad.getKey();
  if (key != NO_KEY) {
    delay(60);
    if (key == '#') {
      if (!doorlocked) {
        closeDoor();
      }
    } else {
      processKey(key);
    }
  }
}
```

- Checks if the door is unlocked and open for more than 2 minutes.
- If so, displays alarm message and sounds buzzer repeatedly until the door is locked.
- Reads keypad input:
  - If `'#'` is pressed and door is unlocked, locks the door.
  - Otherwise, processes digit input for password.

3.6 `processKey(char key)` Function

```
void processKey(char key) {
  if (currentPasswordLength < maxPasswordLength && doorlocked) {
    lcd.setCursor(cursorPosition, 1);
    lcd.print(key);
    delay(500);
    lcd.setCursor(cursorPosition, 1);
```

```
    lcd.print("*");
    cursorPosition++;
    if (cursorPosition > 10) cursorPosition = 5;

    input += key;
    currentPasswordLength++;

    if (currentPasswordLength == maxPasswordLength) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Checking...");
      delay(1000);
      checkPassword();
    }
  }
}
```

- Accepts digit input only if door is locked and password length is not exceeded.
- Displays the digit briefly, then replaces it with an asterisk to mask input.
- Advances cursor position on LCD.
- When 4 digits entered, clears LCD and calls `checkPassword()`.

3.7 `checkPassword()` Function

```
void checkPassword() {
  if (input == password) {
    digitalWrite(buzzer, HIGH); delay(200); digitalWrite(buzzer, LOW);
    openDoor();
    wrongAttempts = 0;
  } else {
    digitalWrite(buzzer, HIGH); delay(1000); digitalWrite(buzzer, LOW);
    wrongAttempts++;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Wrong Password!");
    lcd.setCursor(0, 1);
    lcd.print("Try Again...");
    delay(2000);

    if (wrongAttempts >= 3) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Too many fails!");
      lcd.setCursor(0, 1);
      lcd.print("Wait 10 seconds");
      digitalWrite(buzzer, HIGH); delay(10000); digitalWrite(buzzer, LOW);
      delay(10000);
      wrongAttempts = 0;
    }
    resetPassword();
    showEnterPassword();
```

```
  }
}
```

- Compares entered password with stored password.
- On success:
  - Sounds short beep.
  - Calls `openDoor()`.
  - Resets failed attempts.
- On failure:
  - Sounds long beep.
  - Increments failed attempts.
  - Displays error message.
  - After 3 failed attempts, enforces 10-second lockout with buzzer alarm.
  - Resets input and prompts for password again.

3.8 `openDoor()` Function

```
void openDoor() {
  myservo.write(100);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Door Opened!");
  lcd.setCursor(0, 1);
  lcd.print("Press # to lock");
  doorlocked = false;
  doorOpenTime = millis();
}
```

- Moves servo to 100°, unlocking the door.
- Updates LCD to inform the user.
- Sets door state to unlocked.
- Records current time for open-door alarm.

3.9 `closeDoor()` Function

```
void closeDoor() {
  myservo.write(0);
  lcd.setCursor(0, 0);
  lcd.print("Door Locked!");
  delay(1000);
  doorlocked = true;
  resetPassword();
```

```
  showEnterPassword();
}
```

- Moves servo to 0°, locking the door.
- Displays confirmation message.
- Sets door state to locked.
- Resets password input.
- Shows password prompt.

3.10 `resetPassword()` Function

```
void resetPassword() {
  input = "";
  currentPasswordLength = 0;
  cursorPosition = 5;
}
```

- Clears user input buffer.
- Resets input length and LCD cursor position.

3.11 `showEnterPassword()` Function

```
void showEnterPassword() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Enter Password:");
}
```

- Clears the LCD and displays the password entry prompt.

4. Security and Privacy Measures

- **Password Authentication:** Only users with the correct 4-digit password can unlock the locker.
- **Input Masking:** Password digits are masked on the LCD to prevent onlookers from seeing the password.
- **Lockout Mechanism:** After 3 failed attempts, the system locks out input for 10 seconds and sounds an alarm to deter brute-force attempts.
- **Open-Door Alarm:** If the door remains open for over 2 minutes, the system sounds an alarm and prompts the user to lock the door.
- **Physical Locking:** The servo motor physically secures the locker, preventing unauthorized access.

5. Summary

This project combines hardware and software to create a secure, user-friendly automated locker system. It demonstrates integration of input devices (keypad), output devices (LCD, buzzer, servo), and embedded logic to enforce access control and user feedback.

If you need further assistance with modifications, expansions, or troubleshooting, feel free to ask!