Laboratory Work No. 1.2

User Interaction: STDIO - LCD + Keypad

Gurschi Gheorghe
Group: FAF-231
Technical University of Moldova

February 2, 2026

Contents

1 Domain Analysis

1.1 Description of Technologies Used

1.1.1 LCD Display (Liquid Crystal Display)

LCD is an electronic display module that uses liquid crystals to produce visible images. The LCD 20x4 used in this project can display 20 characters per line across 4 lines, providing sufficient space for user interface elements.

LCD 20x4 I2C Specifications:

- Display: 20 characters x 4 lines
- Controller: HD44780 compatible
- Interface: I2C (only 4 wires needed)
- Operating voltage: 5V
- I2C Address: 0x27 (or 0x3F depending on module)
- Backlight: Blue with white characters

I2C Communication Protocol:

I2C (Inter-Integrated Circuit) is a synchronous, multi-master, multi-slave serial communication protocol. It uses only two wires for communication:

- **SDA (Serial Data)** - bidirectional data line
- **SCL (Serial Clock)** - clock signal line

The I2C module attached to the LCD converts I2C signals to parallel data required by the HD44780 controller, significantly reducing the number of required connections from 16 pins to just 4 pins (VCC, GND, SDA, SCL).

1.1.2 Matrix Keypad

A matrix keypad is an input device that uses a combination of rows and columns to reduce the number of required pins. The 3x4 keypad used in this project has 12 keys arranged in 4 rows and 3 columns, requiring only 7 pins instead of 12.

3x4 Keypad Specifications:

- Keys: 0-9, *, #
- Rows: 4
- Columns: 3
- Required pins: 7 (4 rows + 3 columns)
- Interface: Matrix scanning

How Matrix Scanning Works:

The keypad library scans the matrix by setting each row LOW one at a time and reading the column pins. When a key is pressed, it creates a connection between its row and column, allowing the microcontroller to determine which key was pressed by identifying which row-column intersection is active.

1.1.3 STDIO Library for LCD Output

In this project, the STDIO library is configured to redirect output to the LCD display instead of the serial port. This is achieved by creating a custom putchar function that writes characters to the LCD and linking it to stdout using fdev_setup_stream(). This allows using standard printf() function to display formatted text on the LCD.

1.2 Hardware Components Used

- **Arduino Mega 2560** - microcontroller based on ATmega2560, with 54 digital I/O pins
- **LCD 20x4 I2C** - display module for visual output with I2C interface
- **Keypad 3x4** - matrix keypad for user input
- **2x LED** - light-emitting diodes for status indication (valid/invalid code)
- **2x 220Ω Resistor** - limits current through LEDs
- **Breadboard** - board for rapid prototyping
- **Jumper wires** - for connections

1.3 Software Components Used

- **Arduino IDE** - integrated development environment
- **STDIO Library** - for printf function redirected to LCD
- **Wire Library** - for I2C communication
- **LiquidCrystal_I2C Library** - for LCD control via I2C
- **Keypad Library** - for matrix keypad scanning

1.4 System Architecture

The system is organized in four levels:

1. **Application level** - PIN code verification logic
2. **STDIO level** - input/output abstraction (printf to LCD)
3. **Driver level** - LCD, Keypad, and LED control modules
4. **Hardware level** - I2C, GPIO control

1.5 Case Study

The developed application simulates a PIN-based access control system, similar to those used in door locks, ATM machines, and security systems. The user enters a 4-digit PIN code using the keypad, and the system verifies it against a stored code. Visual feedback is provided through the LCD display and LED indicators, making this a practical example of human-machine interaction in embedded systems.

2 Design

2.1 Electrical Schematic

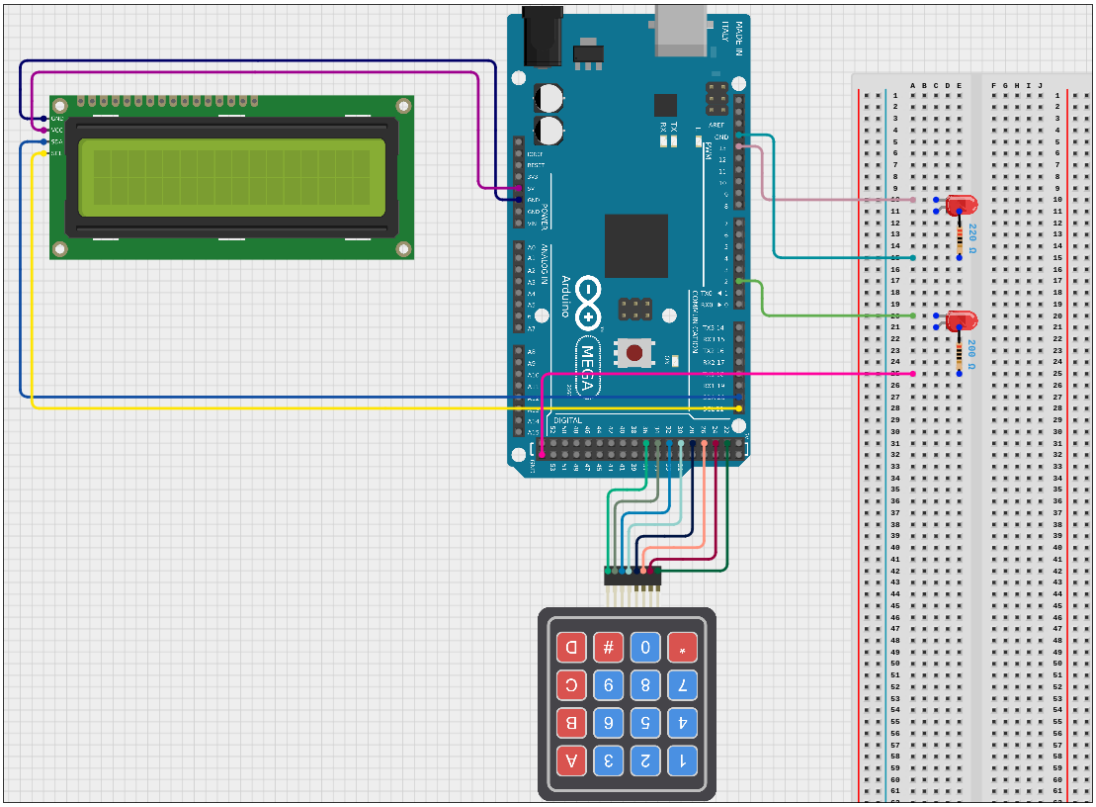

Figure 1: Circuit electrical schematic

LCD I2C Connections:

- **GND** → Arduino **GND**
- **VCC** → Arduino **5V**
- **SDA** → Arduino **Pin 20** (SDA)
- **SCL** → Arduino **Pin 21** (SCL)

Keypad 3x4 Connections:

- **Row 1** → Arduino **Pin 22**
- **Row 2** → Arduino **Pin 24**
- **Row 3** → Arduino **Pin 26**
- **Row 4** → Arduino **Pin 28**
- **Col 1** → Arduino **Pin 30**
- **Col 2** → Arduino **Pin 32**

- **Col 3** → Arduino **Pin 34**

LED Connections:

- **LED 1 (Valid):** Arduino **Pin 13** → LED Anode → LED Cathode → 220Ω Resistor → GND
- **LED 2 (Invalid):** Arduino **Pin 2** → LED Anode → LED Cathode → 220Ω Resistor → GND

2.2 Project Structure

```
Lab2_LCD_Keypad/
 Lab2_LCD_Keypad.ino
     LCD Module (lcdInit, lcdPrint, lcdClear)
     LED Module (ledInit, ledValidOn, ledInvalidOn, ledAllOff)
     Keypad Module (keypadGetKey)
     STDIO Module (lcdPutchar, stdioInit)
     Main Module (setup, loop, resetCode, checkCode)
```

2.3 Modular Implementation

2.3.1 LCD Module

Functions for LCD control:

- `lcdInit()` - initializes LCD, turns on backlight
- `lcdPrint()` - prints text at specified position
- `lcdClear()` - clears the display

2.3.2 LED Module

Functions for LED control:

- `ledInit()` - initializes LED pins as OUTPUT
- `ledValidOn()` - turns on valid LED, turns off invalid LED
- `ledInvalidOn()` - turns on invalid LED, turns off valid LED
- `ledAllOff()` - turns off both LEDs

2.3.3 Keypad Module

Functions for keypad input:

- `keypadGetKey()` - returns pressed key or null if no key pressed

2.3.4 STDIO Module

STDIO library configuration for LCD output:

- `lcdPutchar()` - sends a character to LCD
- `stdioInit()` - links stdout to LCD via fdev_setup_stream()

2.3.5 Main Module

The main logic functions:

- `resetCode()` - clears entered code and resets display
- `checkCode()` - verifies entered code against correct code
- `setup()` - initializes all modules
- `loop()` - handles keypad input and processes commands

3 Results Presentation

3.1 System Default State

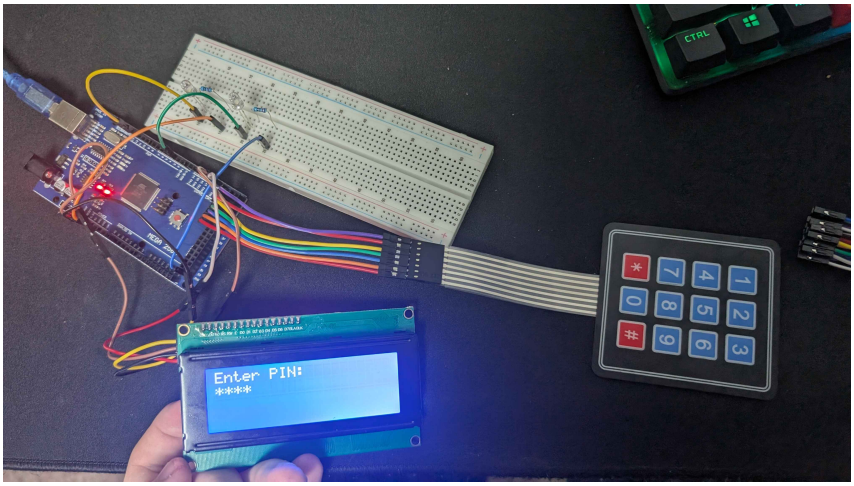

Figure 2: LCD displaying "Enter PIN:" prompt - system ready for input

3.2 PIN Entry

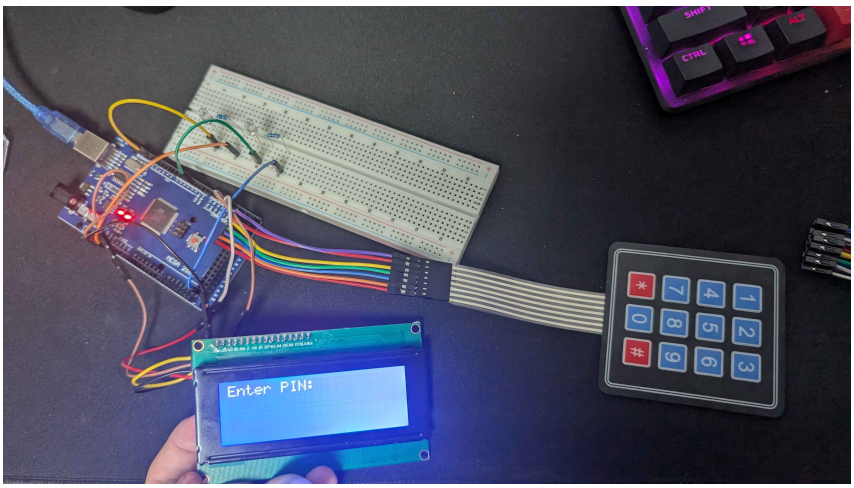

Figure 3: LCD showing masked PIN entry (****) for security

3.3 Testing Valid Code

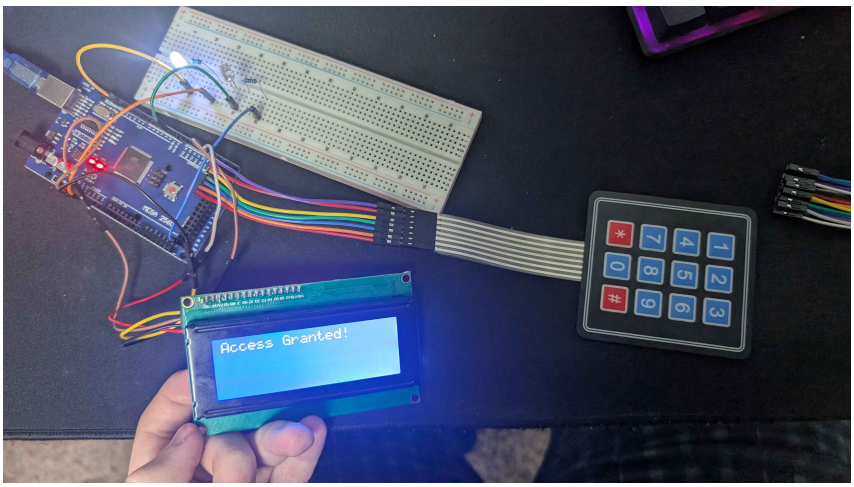

Figure 4: LCD showing "Access Granted!" - correct PIN entered, valid LED on

3.4 Testing Invalid Code

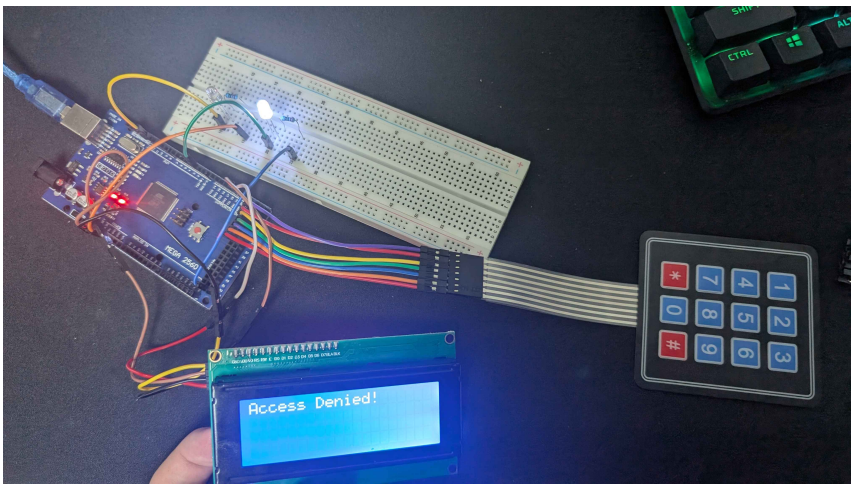

Figure 5: LCD showing "Access Denied!" - incorrect PIN entered, invalid LED on

3.5 Functionality Demonstration

The system was successfully tested:

- The keypad correctly detects all key presses
- The LCD displays masked input (****) for security
- The correct PIN code (1234) triggers "Access Granted" and valid LED
- Incorrect PIN codes trigger "Access Denied" and invalid LED
- The "*" key successfully clears input
- The "#" key successfully submits the code for verification

4 Conclusions

The developed system successfully meets all the laboratory requirements, demonstrating effective integration of multiple peripherals through a modular software architecture. The LCD display provides clear visual feedback to the user, while the matrix keypad enables intuitive input of PIN codes. The STDIO library abstraction allows using familiar printf() syntax for LCD output, simplifying the code and improving readability.

The I2C communication protocol significantly reduces wiring complexity for the LCD, requiring only 4 wires instead of 16. The matrix keypad similarly optimizes pin usage, needing only 7 pins for 12 keys. This efficient use of microcontroller resources demonstrates important embedded systems design principles.

The PIN verification system works reliably, correctly distinguishing between valid and invalid codes and providing appropriate visual feedback through both the LCD and LED indicators. The modular code structure separates concerns between different peripherals, making the code maintainable and reusable for future projects.

This laboratory work demonstrates practical skills in human-machine interface design, combining input devices (keypad), output devices (LCD, LEDs), and embedded software to create a functional access control system. These concepts are directly applicable to real-world applications such as door locks, security systems, and various IoT devices.

Note on AI Tools Usage

During the preparation of this report, the author used **Claude (Anthropic)** for content generation and consolidation. The resulting information was reviewed, validated, and adjusted according to laboratory requirements.

5 Bibliography

1. Arduino Reference - LiquidCrystal Library. `https://www.arduino.cc/en/Reference/LiquidCrystal`
2. Arduino Reference - Keypad Library. `https://playground.arduino.cc/Code/Keypad/`
3. HD44780 LCD Controller Datasheet. `https://www.sparkfun.com/datasheets/LCD/HD44780.pdf`
4. I2C Bus Specification. `https://www.nxp.com/docs/en/user-guide/UM10204.pdf`
5. AVR Libc Reference Manual - Standard IO facilities. `https://www.nongnu.org/avr-libc/user-manual/group__avr__stdio.html`

6 Appendix - Complete Source Code

Listing 1: Lab2_LCD_Keypad.ino

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

//============================================
// LCD MODULE
//============================================
LiquidCrystal_I2C lcd(0x27, 20, 4);

void lcdInit(void) {
    lcd.init();
    lcd.backlight();
    lcd.clear();
}

void lcdPrint(const char* text, int col, int row) {
    lcd.setCursor(col, row);
    lcd.print(text);
}

```

```
22 void lcdClear(void) {
23     lcd.clear();
24 }
25
26 //============================================
27 // LED MODULE
28 //============================================
29 #define LED_VALID_PIN 13
30 #define LED_INVALID_PIN 2
31
32 void ledInit(void) {
33     pinMode(LED_VALID_PIN, OUTPUT);
34     pinMode(LED_INVALID_PIN, OUTPUT);
35     digitalWrite(LED_VALID_PIN, LOW);
36     digitalWrite(LED_INVALID_PIN, LOW);
37 }
38
39 void ledValidOn(void) {
40     digitalWrite(LED_VALID_PIN, HIGH);
41     digitalWrite(LED_INVALID_PIN, LOW);
42 }
43
44 void ledInvalidOn(void) {
45     digitalWrite(LED_VALID_PIN, LOW);
46     digitalWrite(LED_INVALID_PIN, HIGH);
47 }
48
49 void ledAllOff(void) {
50     digitalWrite(LED_VALID_PIN, LOW);
51     digitalWrite(LED_INVALID_PIN, LOW);
52 }
53
54 //============================================
55 // KEYPAD MODULE
56 //============================================
57 const byte ROWS = 4;
58 const byte COLS = 3;
59
60 char keys[ROWS][COLS] = {
61     {'1', '2', '3'},
62     {'4', '5', '6'},
63     {'7', '8', '9'},
64     {'*', '0', '#'}
65 };
66
67 byte rowPins[ROWS] = {22, 24, 26, 28};
68 byte colPins[COLS] = {30, 32, 34};
69
70 Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS,
      COLS);
71
```

```
72  char keypadGetKey(void) {
73      return keypad.getKey();
74  }
75
76  //============================================
77  // STDIO MODULE
78  //============================================
79  static int lcdPutchar(char c, FILE *stream) {
80      lcd.write(c);
81      return c;
82  }
83
84  FILE lcdStream;
85
86  void stdioInit(void) {
87      fdev_setup_stream(&lcdStream, lcdPutchar, NULL,
            _FDEV_SETUP_WRITE);
88      stdout = &lcdStream;
89  }
90
91  //============================================
92  // MAIN MODULE
93  //============================================
94  #define CODE_LENGTH 4
95  #define CORRECT_CODE "1234"
96
97  char enteredCode[CODE_LENGTH + 1] = "";
98  uint8_t codeIndex = 0;
99
100 void resetCode(void) {
101     memset(enteredCode, 0, sizeof(enteredCode));
102     codeIndex = 0;
103     ledAllOff();
104     lcdClear();
105     lcd.setCursor(0, 0);
106     printf("Enter␣PIN:");
107     lcd.setCursor(0, 1);
108 }
109
110 void checkCode(void) {
111     if (strcmp(enteredCode, CORRECT_CODE) == 0) {
112         lcdClear();
113         lcd.setCursor(0, 0);
114         printf("Access␣Granted!");
115         ledValidOn();
116     } else {
117         lcdClear();
118         lcd.setCursor(0, 0);
119         printf("Access␣Denied!");
120         ledInvalidOn();
121     }
```

```
122     delay(2000);
123     resetCode();
124 }
125 
126 void setup() {
127     Serial.begin(9600);
128     lcdInit();
129     ledInit();
130     stdioInit();
131     resetCode();
132 }
133 
134 void loop() {
135     char key = keypadGetKey();
136 
137     if (key) {
138         if (key == '#') {
139             checkCode();
140         }
141         else if (key == '*') {
142             resetCode();
143         }
144         else if (codeIndex < CODE_LENGTH) {
145             enteredCode[codeIndex] = key;
146             codeIndex++;
147             enteredCode[codeIndex] = '\0';
148 
149             lcd.setCursor(0, 1);
150             for (int i = 0; i < codeIndex; i++) {
151                 printf("*");
152             }
153         }
154     }
155 }
```