



UNIVERSITATEA TEHNICĂ A MOLDOVEI
FACULTATEA: CALCULATOARE,
INFORMATICĂ ȘI MICROELECTRONICĂ
DEPARTAMENTUL: INGINERIA
SOFTWARE ȘI AUTOMATICA

Laboratory work NR. 1.2

Interaction with the user: LCD, Keypad

Executed: Konjevic Alexandra, gr. FAF-213

Verified: Assistant Professor Moraru Dumitru

Chișinău – 2024

Task: Create an application for interaction with user through LCD and keypad using STDIO library for being able to use functions ``printf()`` and ``scanf()``.

Objectives:

Task 1. Configure application for working with the library STDIO through LCD and keypad for exchanging text through serial interface.

Task 2. Create an MCU-based application which validates a code (password) introduced in a 4x4 keypad, and displaying corresponding messages on LCD:

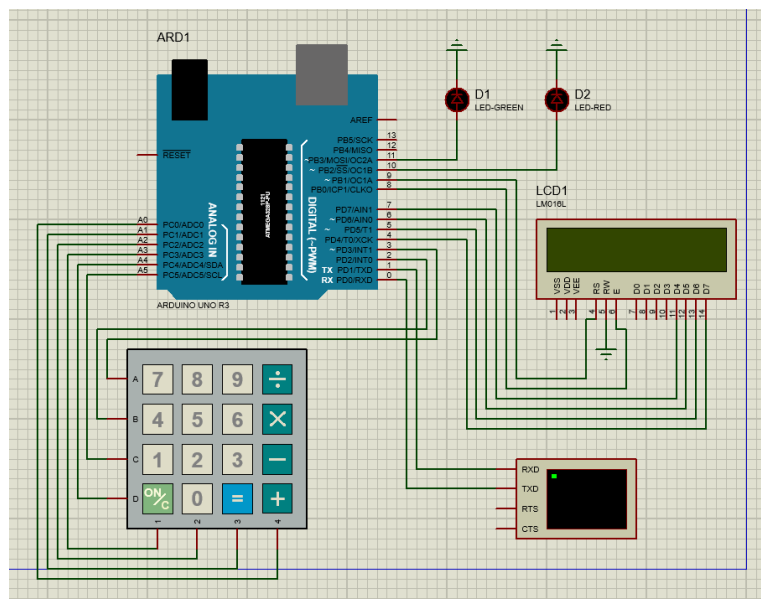
- for an invalid password turn a red led on, and for a valid password – a green led

Implementation:

Given the problem, we need first of all to investigate the LCD and keypad components.

The Arduino LCD screen is a backlit LCD screen with headers. You can draw text, images, and shapes to the screen with the 'Liquid Crystal' library. There is an onboard micro-SD card slot on the back of the screen that can, among other things, store bitmap images for the screen to display. The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

- A **register select (RS)** pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
- A **Read/Write (R/W)** pin that selects reading mode or writing mode
- An **Enable** pin that enables writing to the registers
- **8 data pins (D0 -D7)**. The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.



From the documentation of the `LiquidCrystal` library, we can read that the LCD can be controlled using either 4 or 8 data lines. For 4 lines of control, the ones that must be used are: D4, D5, D6, D7. The RW pin can be connected to ground (as I did in my circuit). We also need to connect to the Arduino the receiving (RS) and enable (E) pins of the LCD.

As a keypad, I used a calculator keypad 4x4 `KEYPAD-SMALLCALC` (because in Proteus there is no other 4x4 keypad). The interaction with the keypad can be implemented using the library `Keypad`. I connected the keypad with the digital and analog pins of the Arduino.

Now, let's analyze the code part by part:

1. Libraries used:

- `LiquidCrystal.h` - interaction with the LCD
- `Keypad.h` - interaction with the keypad
- `stdio.h` - printing log messages in the serial monitor

2. Constants and variables declaration:



```
1  const int LED_RED = 10;
2  const int LED_GREEN = 11;
3
4  const int BAUD_RATE = 9600;
5
6  const int DELAY = 1000;
7
8  const int PASSWORD_LENGTH = 5;
9
10 LiquidCrystal lcd(9, 8, 7, 6, 5, 4);
11
12 const int ROWS = 4;
13 const int COLS = 4;
14
15 byte rowPins[ROWS] = {3, 2, 19, 18};
16 byte colPins[COLS] = {17, 16, 15, 14};
17
18 char keys[ROWS][COLS] = {
19   {'7', '8', '9', '/'},
20   {'4', '5', '6', '*'},
21   {'1', '2', '3', '-'},
22   {'C', '0', '=', '+'}};
23
24 Keypad customKeypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
25
26 int keyCount = 0;
27
28 const char PASSWORD[PASSWORD_LENGTH] = "1234";
29 char introducedPassword[PASSWORD_LENGTH];
```

Figure 2. Constants and variables declaration


First of all, there are the pins with which the LEDs are connected, after that, the baud rate for the serial monitor; delay (1000 ms) and the length of the password. After that, I create a variable `lcd` of the type `LiquidCrystal` - in the constructor are declared the values of the Arduino pins at which

are connected the RS pin of the LCD, the Enable pin of the LCD, and the data pins of it. After that, the constants that represent the number of the rows and columns on the keypad (4x4). I declared after that two arrays that contain the pins of the Arduino to which are connected the pins of the keypad. Also, after that, I declared a two-dimensional array, with the values of the keypad.

Next, there is instantiated a variable of type Keypad – in the constructor we need to call the `makeKeymap` function, to transform the array `keys` into a map that the library understands. Also, we need to pass the pins of rows and columns, and the number of the rows and columns.

`keyCount` is a value used as an iterator, to count the number of the characters pressed by the user. After that, I declare the actual password, and another variable, `introducedPassword`, which is holding the password introduced by the user through the keypad.

3. `putChar()` function:




```
1  int putChar(char c, FILE *fp)
2  {
3      if (c == '\n')
4      {
5          Serial.write('\n');
6          Serial.write('\r');
7          return 0;
8      }
9      return !Serial.write(c);
10 }
```

Figure 3. `putChar()` function

This is a function that I used to be able to use the `stdio.h` library's function `printf`. This function specifies a target for STDOUT. If a device such as a terminal needs special handling, it is in the domain of the terminal device driver to provide this functionality. Thus, a simple function suitable as `put()` for `fdevopen()` that talks to a UART interface might look like the one that I implemented: `putChar()`. The function passed as put shall take two arguments, the first a character to write to the device, and the second a pointer to FILE, and shall return 0 if the output was successful.

4. `ClearKeypad()` function:

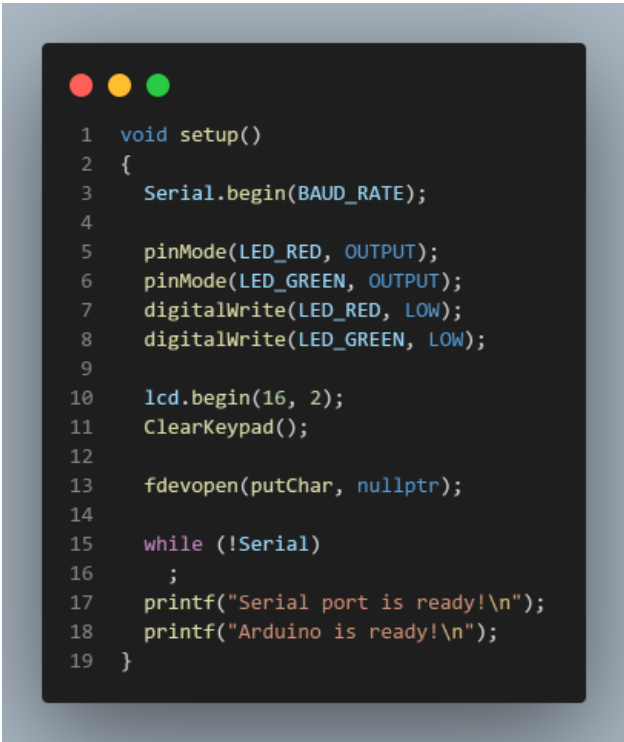
A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains the following C++ code:

```
1 void ClearKeypad()
2 {
3     lcd.clear();
4     lcd.print("Password: ");
5     lcd.setCursor(0, 1);
6     lcd.cursor();
7 }
```

Figure 4. `ClearKeypad()` function

This is a helper function, to clear the keypad after some inputs introduced by the user. It clears the whole keypad, and prints the string “Password: ”, after that, sets the cursor on the next line, and actually displays the cursor.

5. Setup function:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains the following C++ code:

```
1 void setup()
2 {
3     Serial.begin(BAUD_RATE);
4
5     pinMode(LED_RED, OUTPUT);
6     pinMode(LED_GREEN, OUTPUT);
7     digitalWrite(LED_RED, LOW);
8     digitalWrite(LED_GREEN, LOW);
9
10    lcd.begin(16, 2);
11    ClearKeypad();
12
13    fdevopen(putChar, nullptr);
14
15    while (!Serial)
16        ;
17    printf("Serial port is ready!\n");
18    printf("Arduino is ready!\n");
19 }
```

Figure 5. Setup function

In the setup function, first of all, I entered the command `Serial.begin()` that starts the serial communication. After that, I set the LEDs as outputs. The method `lcd.begin(16,2)`, is needed to declare the LCD with dimensions 16x2. I call the function `ClearKeypad()`, and the “Password: ” is shown on the LCD. After that, I use the function `fdevopen()` so declare a stream that redirects the STDOUT in a specified file. And after that, I added some log messages in the serial monitor: a message that says that the serial port is ready, and another one to say that the Arduino is ready.

Loop function:

```
1 void loop()
2 {
3   char customKey = customKeypad.getKey();
4
5   if (customKey == 'C')
6   {
7     ClearKeypad();
8     keyCount = 0;
9   }
10
11  if (customKey == '=')
12  {
13    if (strncmp(introducedPassword, PASSWORD, keyCount) == 0)
14    {
15      ClearKeypad();
16      digitalWrite(LED_GREEN, HIGH);
17      lcd.print("Door Unlocked!");
18      delay(DELAY);
19      ClearKeypad();
20      digitalWrite(LED_GREEN, LOW);
21    }
22    else
23    {
24      ClearKeypad();
25      digitalWrite(LED_RED, HIGH);
26      lcd.print("Invalid Password!");
27      delay(DELAY);
28      ClearKeypad();
29      digitalWrite(LED_RED, LOW);
30    }
31    keyCount = 0;
32  }
33
34  if (customKey && (keyCount < 4) && (customKey != '=') && (customKey != 'C'))
35  {
36    lcd.print('*');
37    introducedPassword[keyCount] = customKey;
38    keyCount++;
39  }
40 }
```

Figure 6. Loop function

First of all, the program reads the value that is introduced by the user in the keypad, using the method `getKey()`. After that, it checks whether the key pressed is “C”, and if so, the keypad resets and so does the counter of the characters `keyCount`. After that, it checks if the key is “=” and if so, it compares if the introduced string is the needed password or not. If the password is correct, the words “Door Unlocked!” are printed on the LCD, and the green LED is turned on for a second. After this second passes, the LCD screen is cleared, and the green LED is turned off. The same goes for the incorrect password, but the RED led is turned on, an error message is displayed on the LCD.

Simulation:

Figure 7. Initial state of the circuit

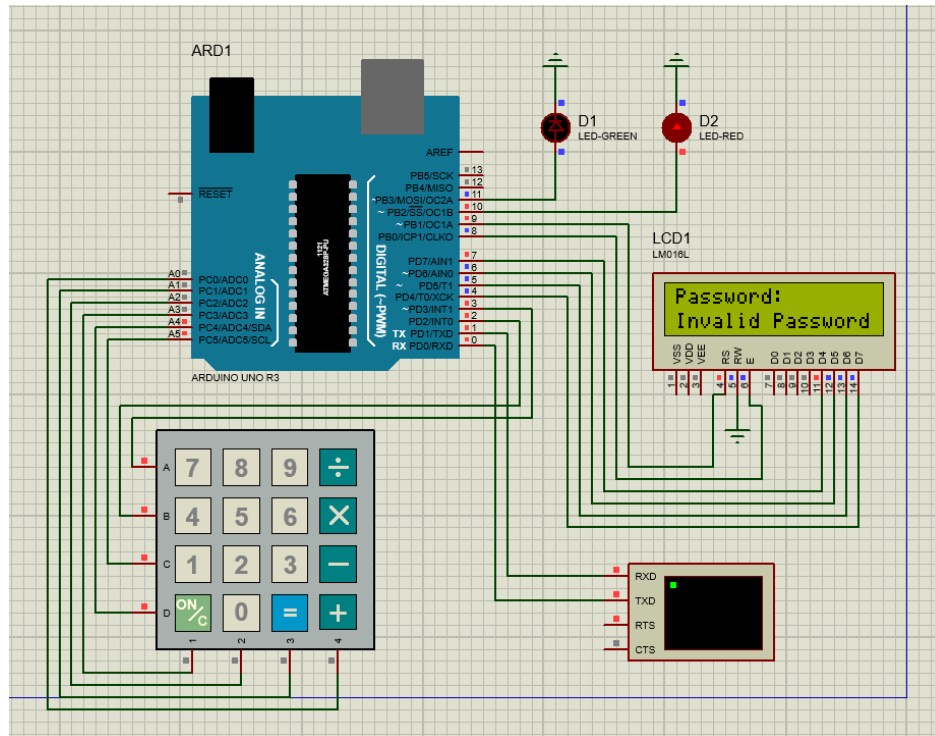


Figure 8. Invalid password introduced

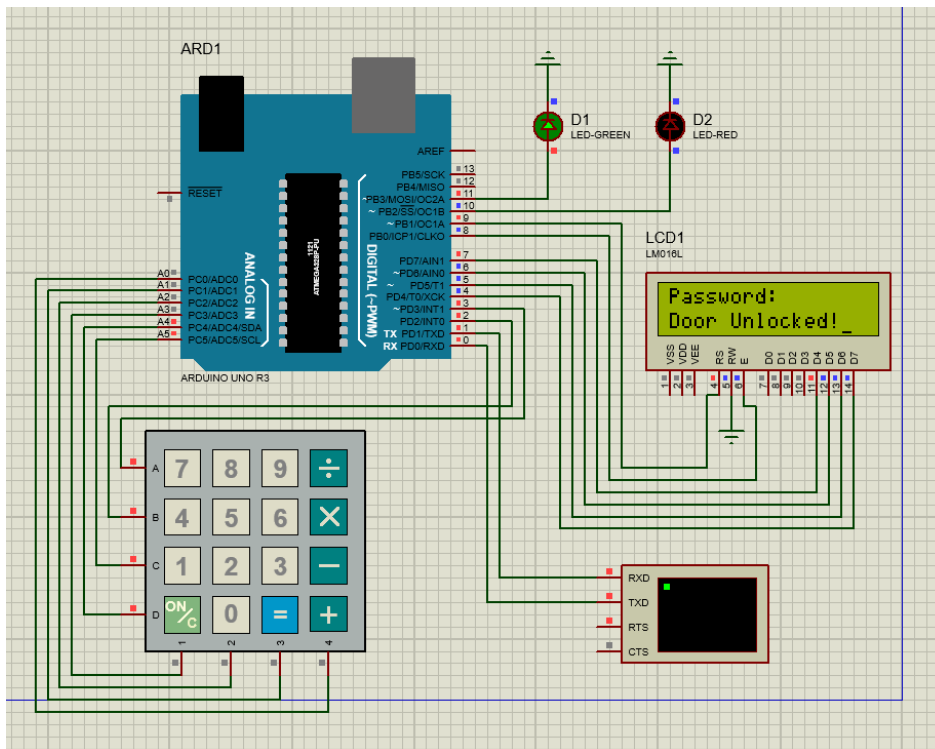


Figure 9. Valid password introduced

Conclusion:

In conclusion, the execution of the laboratory work involving the creation of a door lock system using Arduino Uno, an LCD screen, and a 4x4 keypad has provided valuable insights into the practical application of microcontroller-based projects. Through the implementation of this circuit, I gained hands-on experience in designing and programming a simple yet effective security mechanism.

The integration of the Arduino Uno, LCD screen, and keypad 4x4 allowed for the development of a user-friendly interface for the door lock system. The LCD screen served as a visual display, providing real-time feedback and information, while the keypad facilitated user input for password authentication. This combination not only demonstrated the versatility of Arduino in interfacing with various components but also highlighted the potential for customization and scalability in future projects.

The successful completion of the laboratory work emphasized the importance of careful circuit design, coding, and troubleshooting in the development process. It required a comprehensive understanding of both hardware and software aspects, encouraging a holistic approach to problem-solving. Additionally, the project underscored the significance of effective communication between the Arduino and external devices, showcasing the fundamental principles of embedded systems.

Through this hands-on experience, I have not only strengthened my proficiency in Arduino programming but also gained a deeper appreciation for the practical applications of microcontrollers in real-world scenarios. This project has not only enhanced my technical skills but also instilled a sense of confidence in my ability to tackle complex electronic projects.

In future endeavors, I anticipate applying the knowledge and skills acquired from this laboratory work to more advanced projects, further exploring the capabilities of Arduino and expanding my expertise in the realm of embedded systems. Overall, the door lock system project has been a rewarding learning experience, equipping me with valuable practical insights that will undoubtedly contribute to my growth in the field of electronics and programming.

BIBLIOGRAPHY

1. LCD Screen: Arduino LCD Screen. Arduino official site, ©2024 [quote 02.02.2024]. Access link: <https://docs.arduino.cc/retired/other/arduino-lcd-screen/>
2. LCD Display: Liquid Crystal Displays (LCD) with Arduino. Arduino official site, ©2024 [quote 02.02.2024]. Access link: <https://docs.arduino.cc/learn/electronics/lcd-displays/>
3. `LiquidCrystal` documentation: LiquidCrystal - LiquidCrystal(). Arduino official site, ©2024 [2024]. Access link: <https://www.arduino.cc/reference/en/libraries/liquidcrystal/liquidcrystal/>
4. Analog pins as digital: How to define analog pins. Arduino official forum, @2020 [quote 10.10.2010] Access link: <https://forum.arduino.cc/t/how-can-i-define-the-analog-pin-2-and-digital-pin/49434/12>