**TECHNICAL UNIVERSITY OF MOLDOVA FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS**

**DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS**

Report of laboratory work №1

# Theme: User interaction

## Fulfilled: st. gr. FAF-202 Popa Eugeniu

**Controlled: univ. lecturer Moraru Dumitru**

# Chișinău 2023

## THE TASKS OF THE LABORATORY WORK

**Task 1:** To design an MCU-based application that would change the status of an LED when a button press is detected.

**Task 2:** To design an MCU-based application that would receive commands from the terminal via the serial interface to set the status of an LED.

* *led on* for switching on and *led off* for switching off…, the system must respond with text messages about the order confirmation
* STDIO library for terminal text exchange has to be used

**Task 3:** To design an MCU-based application that would detect code from a 4x4 keyboard, verify it, and display a message on an LCD.

* for valid code to light a green LED, for invalid code, a red LED
* STDIO library to scan the keyboard and display information on the LCD has to be used

## THE PROGRESS OF THE WORK

**Task 1**

## Description of the main functions used to perform the task

To perform task 1, I have created 2 classes: “Button” and “Led”.

“Button.h” is used to work with the button component. It has a constructor, and several functions (Fig 1.1).

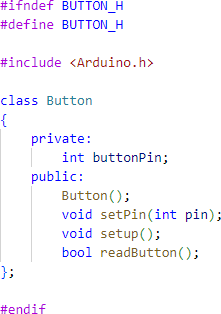


Fig 1.1 Button.h – The button custom library

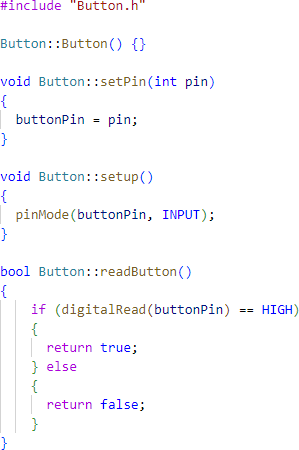
They are implemented in “Button.cpp” (Fig 1.2). “void setPin (int pin)” sets the pin of the button, “void setup()” sets the pin mode, and “bool readButton()” returns a boolean value. Is the button is pressed, it returns true. Else it returns false.

Fig 1.2 Button.cpp – Implementation of the button custom library

“Led.h” is used to work with the led component. It has a constructor, and several functions (Fig 1.3).

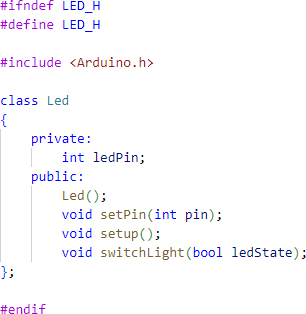


Fig 1.3 Led.h – The led custom library

They are implemented in “Led.cpp” (Fig 1.4). “void setPin (int pin)” sets the pin of the led, “void setup()” sets the pin mode, and “bool switchLight(bool ledState)” turns the led on or off depending on the parameter.

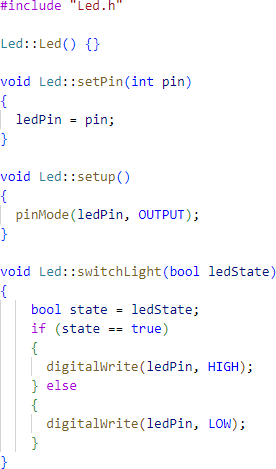


Fig 1.4 Led.cpp - Implementation of the led custom library

The main file “main.cpp” (Fig 1.5) creates the objects, sets them up, and runs the loop to switch the led by clicking the button.

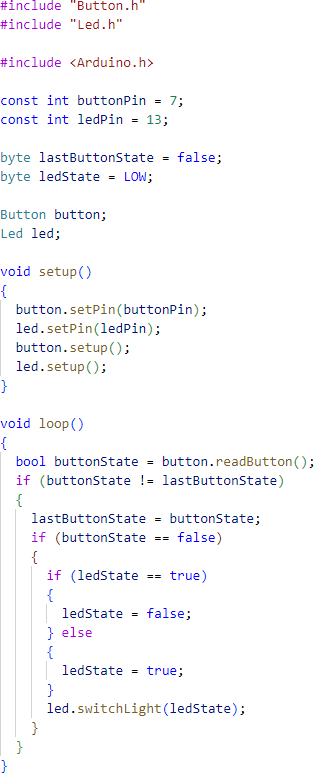


Fig 1.5 main.cpp – The main file of the program

## Electrical scheme made in Proteus

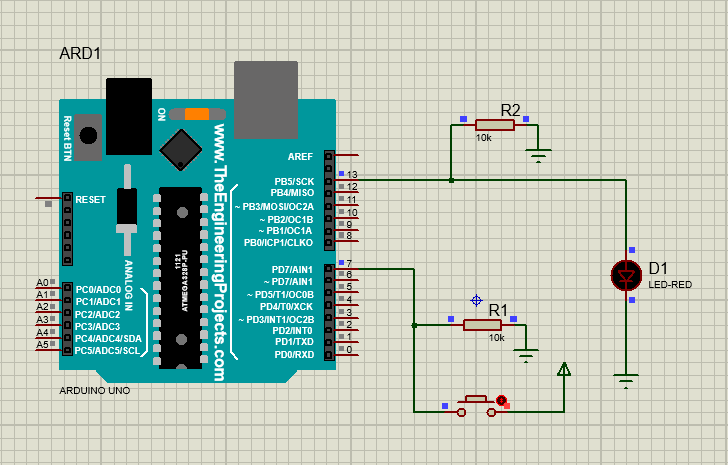


Fig 1.6 Electrical scheme for Task 1

## Block diagram of the program

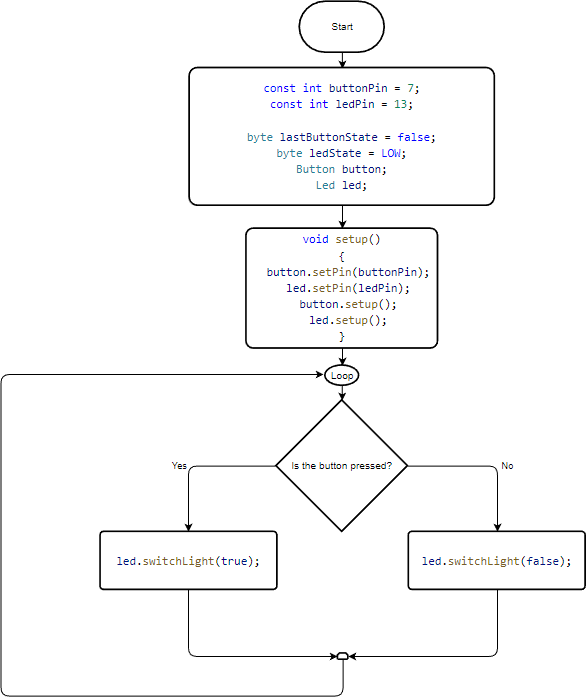


Fig 1.7 Flowchart of the program for Task 1

## Task 2

1. **Description of the main functions used to perform the task**

To perform task 2, I have created 2 classes: “SInterface” and “Led”. “Led” is from the previous task. “SInterface.h” (Fig 1.1) is used to handle the serial input, to give commands to the led, and to output messages about the led state and message receival. It has a constructor and some functions.

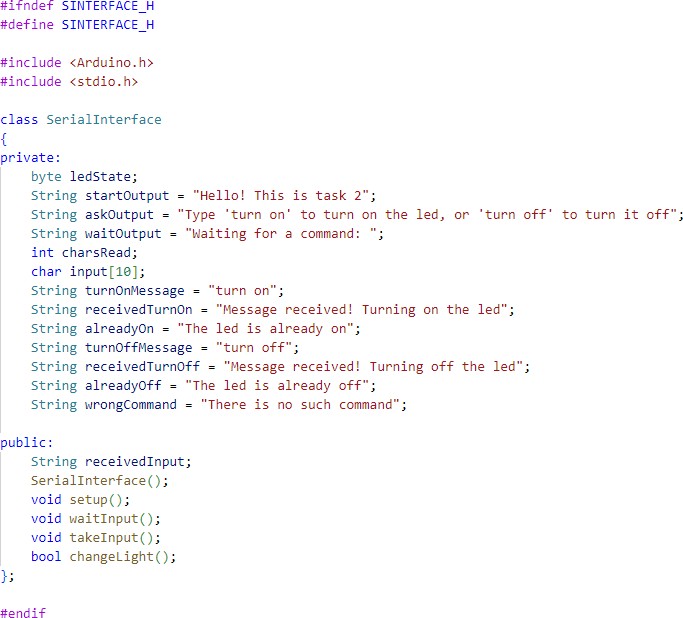


Fig 2.1 SInterface.h - The serial interface custom library

These functions are implemented in “SInterface.cpp” (Fig. 1.2). The constructor sets the led state to low by default. “setup()” sets the interface. “waitInput()” waits for the input from the interface and prints it if there is one. “takeInput()” takes the input of max.10 characters, than prints the “receivedInputMessage”. “bool changeLight()” returns true if the ledState is false and sets the state to true, and vice versa.



Fig 2.2 SInterface.cpp- Implementation of the serial interface custom library

The main file “main.cpp” (Fig 2.3) creates the objects, sets them up, and runs the loop to switch the led by the return value of the “changeLight()” function.

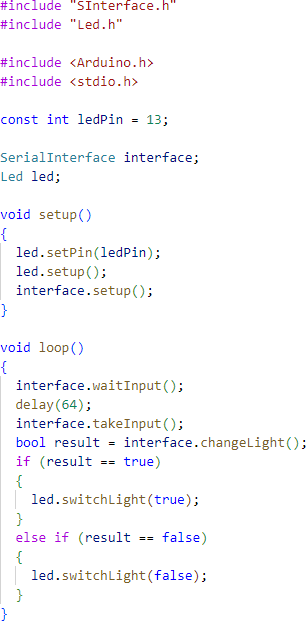


Fig 2.3 main.cpp - The main file of the program

## Electrical scheme made in Proteus

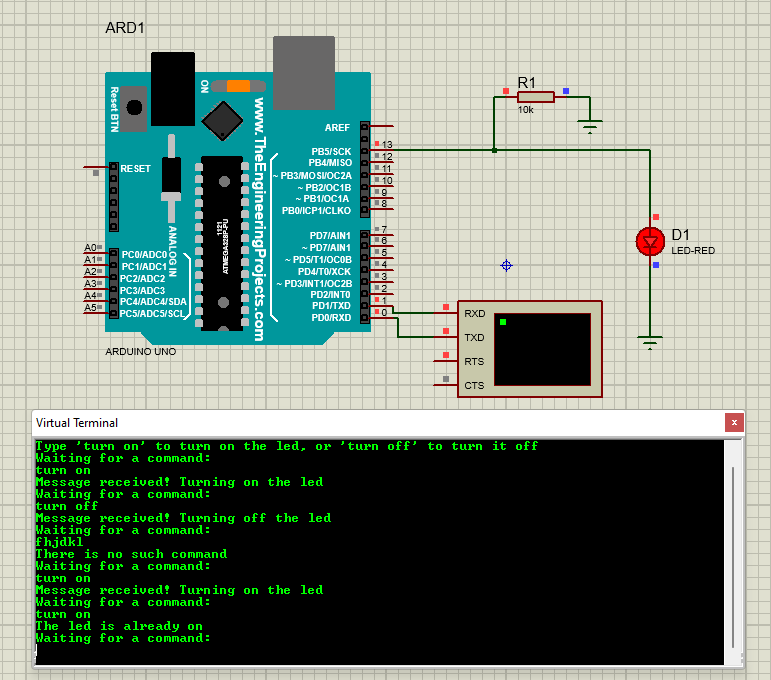


Fig 2.4 Electrical scheme for Task 2

## Block diagram of the program

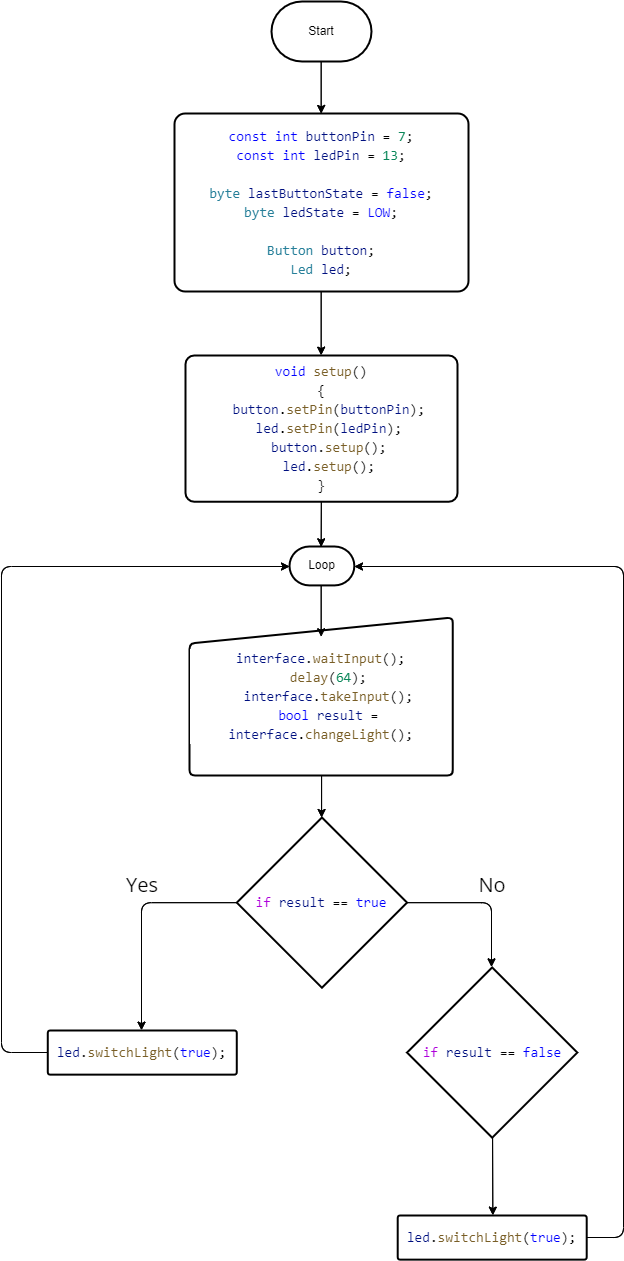


Fig 2.5 Flowchart for Task 2

## Task 3

1. **Description of the main functions used to perform the task**

To perform this task, one third-party library for the keypad is used: “Keypad.h” [1]; Also, I have created my own library “CustomKeypad”.

“CustomKeypad.h” (Fig 3.1) is a library that handles the input from the keypad, checks the password matching, gives commands to the leds, outputs messages to the terminal. “SInterface” and “Led” are from the previous tasks.

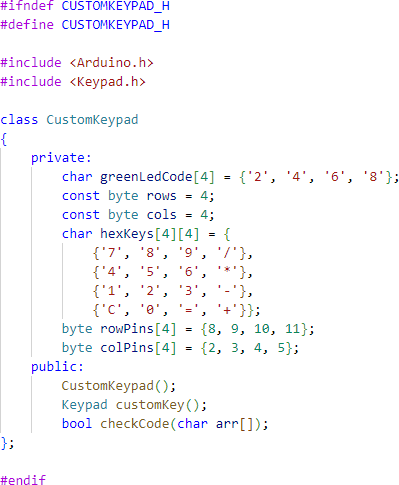


Fig 3.1 CustomKeypad.h - The keypad custom library

The functions are implemented in “CustomKeyboard.cpp” (Fig 3.2). “customKey()” initialized the keyboard from library “Keypad.h”. “checkCode()” is used to check the input array of numbers. It checks if the input numbers are equal to the secret code, and returns true if yes, otherwise false. In the main function this result will turn on the green or the red light.



Fig 3.2 CustomKeypad.cpp - Implementation of the keypad custom library

“Led” library has one more function “switchLightTimer()” (Fig 3.3) which turns the led for a specified period of time.

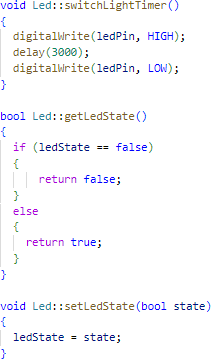


Fig 3.3 Function from Led.cpp - Implementation of the led custom library

The main file “main.cpp” (Fig 3.4) creates the objects, sets them up, and runs the loop where the keyboard waits for the input, takes an array of for numbers, then checks the code. If the code is correct, it lights the green led, otherwise the red. It also outputs messages to the terminal if the code is correct or not.

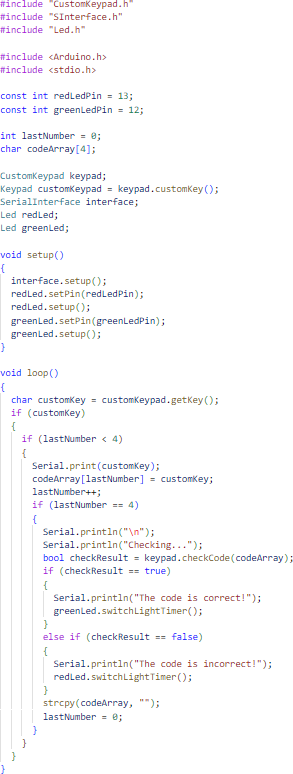


Fig 3.4 main.cpp - The main file of the program

## Electrical scheme

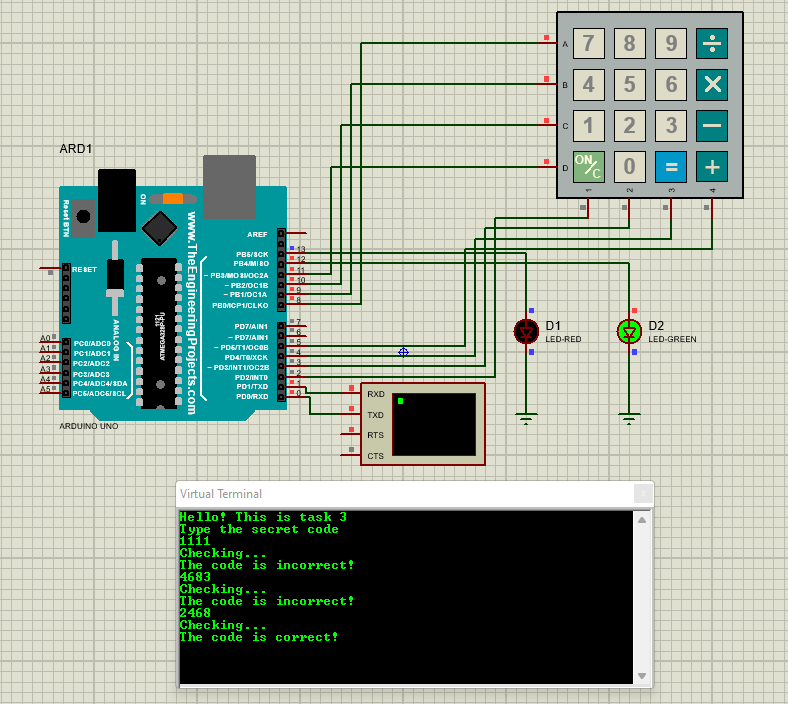


Fig 3.5 Electrical scheme for Task 3

## Block diagram of the program

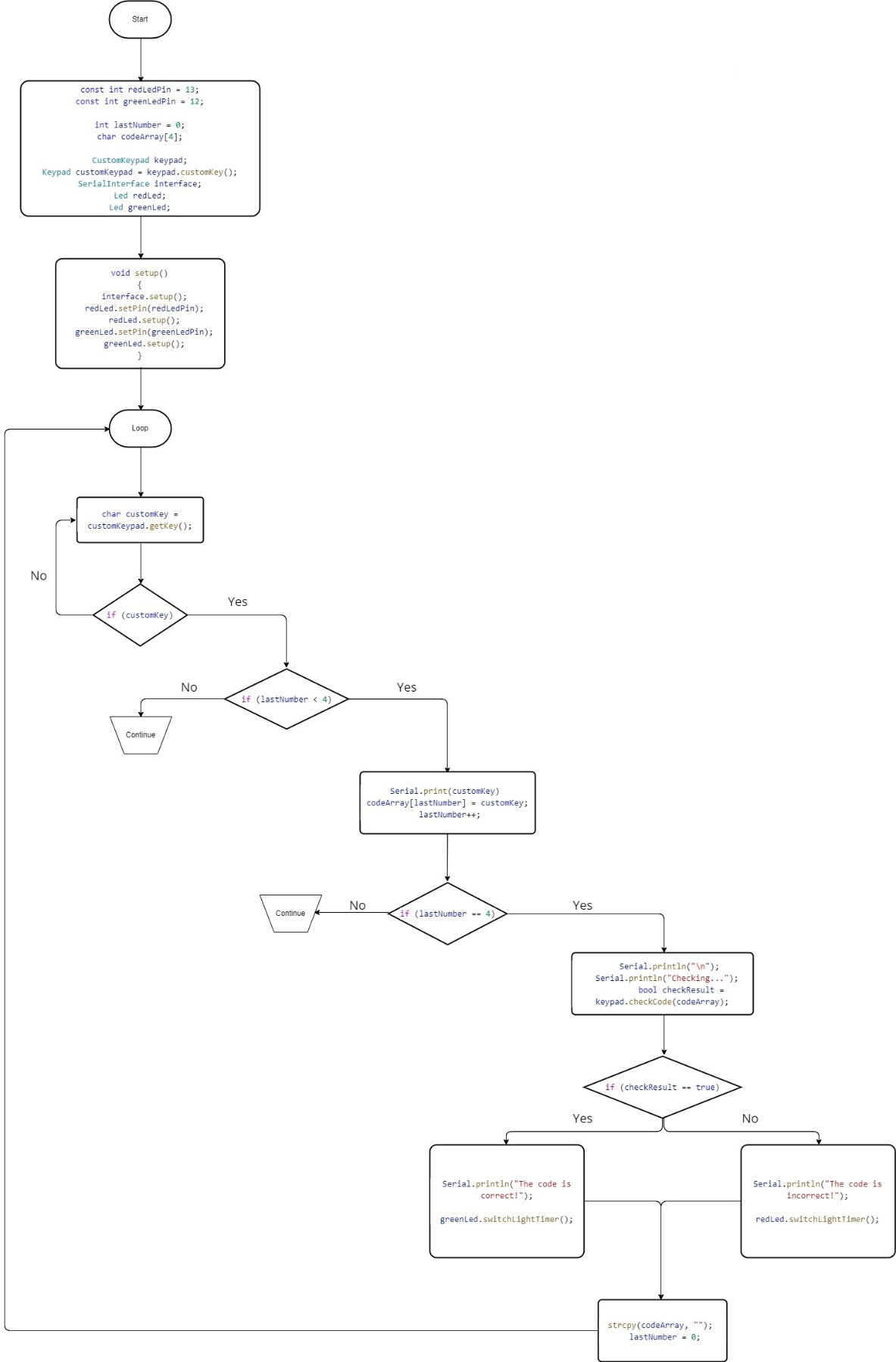


Fig 3.6 Flowchart for Task 3

# CONCLUSIONS

In this laboratory work, I have learned the basics of programming on C++ for Arduino. Also, I have learned about simulating hardware in Proteus, which was used for testing and designing electrical schemes.

It has been a valuable learning experience. I have gained hands-on experience in designing and implementing microcontroller-based applications using various components such as LEDs, buttons, serial interface, keyboard, etc. The tasks have allowed me to apply the theoretical concepts that I have learned in the classroom to real-world applications and have helped me to develop my coding and problem-solving skills. I have learned how to use different libraries and programming techniques to control and monitor various physical devices and interfaces. The tasks have also taught me how to work with multiple components and communication interfaces, such as the serial interface, which has given me a deeper understanding of microcontroller-based systems. Overall, this laboratory work has been a challenging but rewarding experience. I feel more confident in my ability to design and implement microcontroller-based applications and I am grateful for the opportunity to apply my knowledge in a practical setting.

# Bibliography

[1] Keypad: official documentation page of the library. Access link: [Arduino Playground - Keypad Library](https://playground.arduino.cc/Code/Keypad/)

# Tables

**Table 1** Arduino Uno technical specifications

|  |  |
| --- | --- |
| **Numele parametrului** | **Valoarea** |
| Microcontroller | ATmega328P |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (ATmega328P) of which 0.5 KB  used by bootloader |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| LED\_BUILTIN | 13 |
| Length | 68.6 mm |
| Width | 53.4 mm |
| Weight | 25 g |