**TECHNICAL UNIVERSITY OF MOLDOVA**

**FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS**

**DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS**

**Report of laboratory work №1**

**Theme: User interactions**

**Fulfilled: st.gr. FAF-191 Boico Alexandr**

**Controlled: univ. lecturer Moraru Dumitru**

**Chișinău 2022**

**The task 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 lighting and led off for switching off, the system must respond with text messages about the order confirmation

- use STDIO library for terminal text exchange

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

- for valid code to light a green LED, for invalid code, a red LED.

- Use STDIO to scan the keyboard and display information on the LCD.

**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 libraries: “Button.h” and “LED.h”.

“Button.h” is used to rule the button component and the single function is “readButton()” (see in Appendix 1), that returns a boolean value. If the button is pressed, then it returns TRUE, else it returns FALSE.

**Appendix 1**

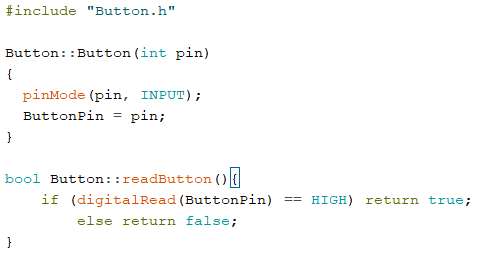


Fig 1.1 Button.cpp

LED.h is used to rule the LED component. It has 2 functions: “turnOn()” and “turnOff()”(see in Appendix 2). “turnOn()” function turns on the LED, but “turnOff()” respectively turns off.

**Appendix 2**

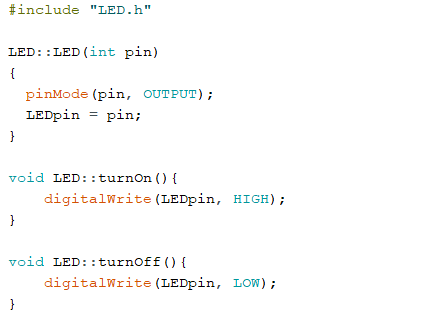


Fig 1.2 LED.cpp

In the Task1Main.cpp(see in Appendix 3) we have declaration of constants for button and LED pins, setting up of Button and LED classes and loop process, where is if statement, that asks the position of the button( if is pressed or not) and depending on the response it change the status of the LED.

**Appendix 3**



Fig 1.3 Task1Main.ino

* **Electrical scheme**

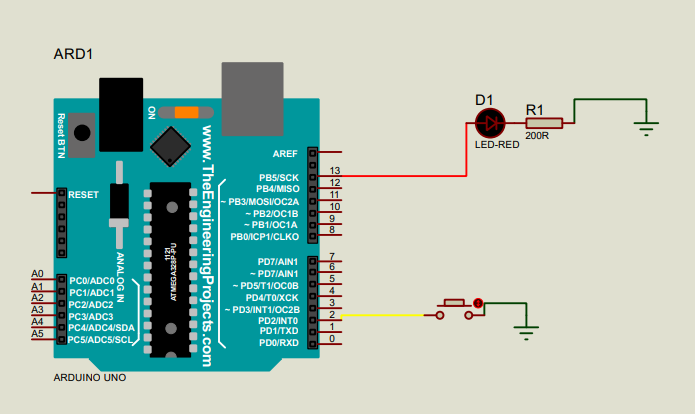
****

Fig 1.4 Electrical scheme for Task 1

* **Flowchart of the program**

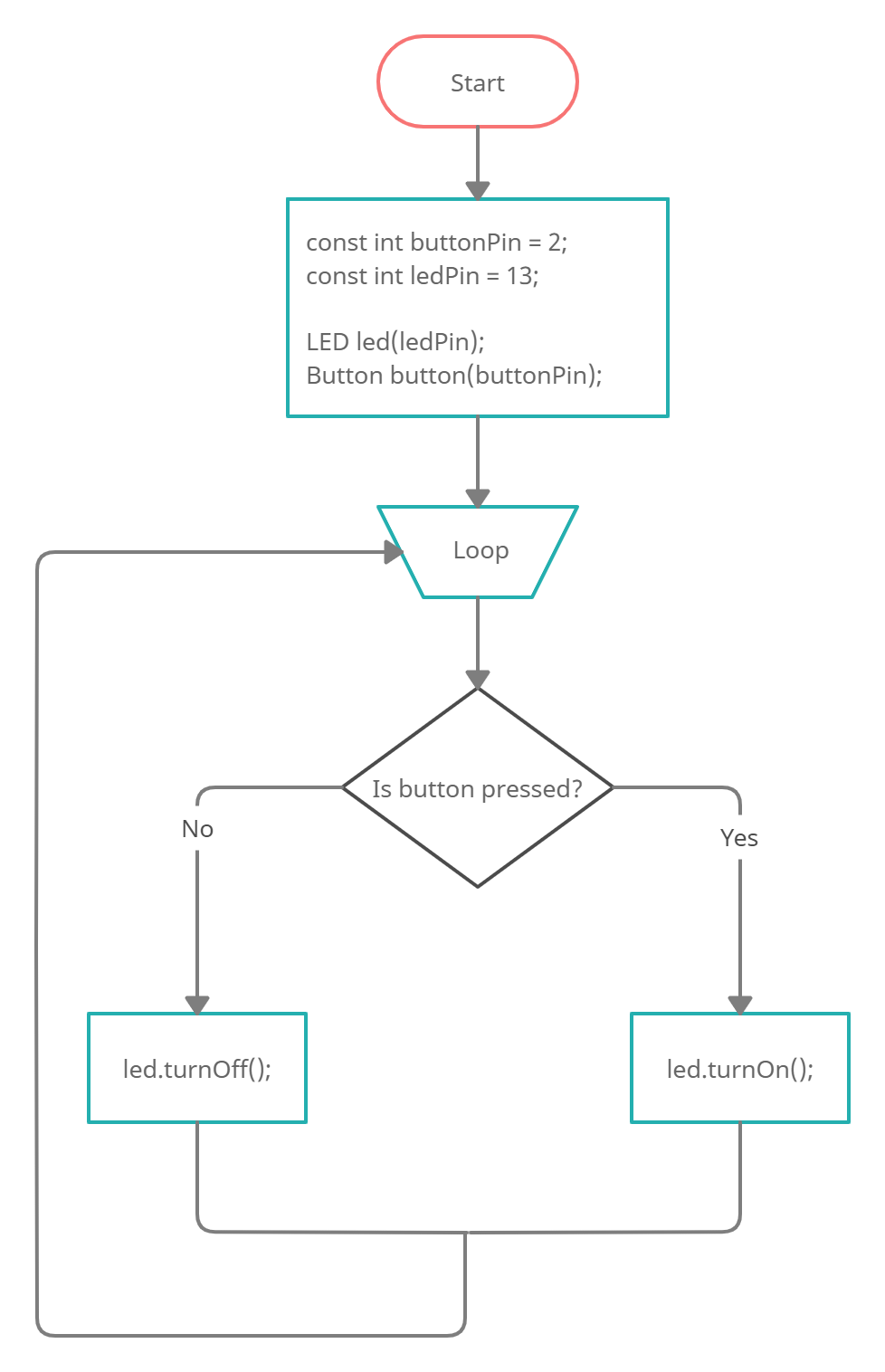


Fig 1.5 Flowchart of the program for Task 1 [5]

* [**Video of the real assembled equipment**](https://drive.google.com/file/d/1OtySxTULnPdXDlMz0aUPF7ruHhngHOoK/view?usp=sharing)

**Task 2**

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

To perform task 2, I have created the library “SerialIO.h” and have used “LED.h” from the previous task.

“SerialIO.h” is used to handle the Serial input. It has 2 functions: “readIn()” and “check(String input, String pattern)”. (see Appendix 4). “readIn()” function scans the Serial Port, if it contains something, it reads the input and returns it to the main program in the form of a string. “check(String input, String pattern)” function checks if the input string is equal to the pattern and returns to the main program if it is true or false.

**Appendix 4**

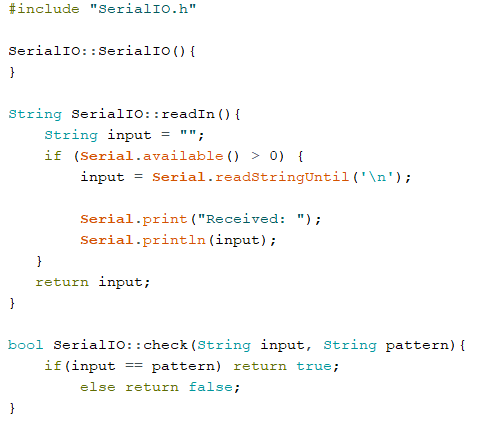


Fig 2.1 SerialIO.cpp

In the Task2Main.cpp(see in Appendix 5) we have declaration of constants for LED pin, patterns of strings, setting up of SerialIO and LED classes and loop process, where we ask for input string and checking if it corresponds to one of the patterns and depending on the response it change the status of the LED.

**Appendix 5**

****

Fig 2.2 Task2Main.ino

* **Electrical scheme**

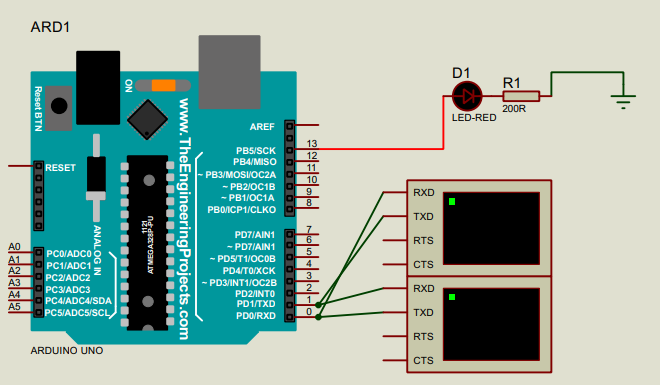
****

Fig 2.3 Electrical scheme for Task 2

* **Flowchart of the program**

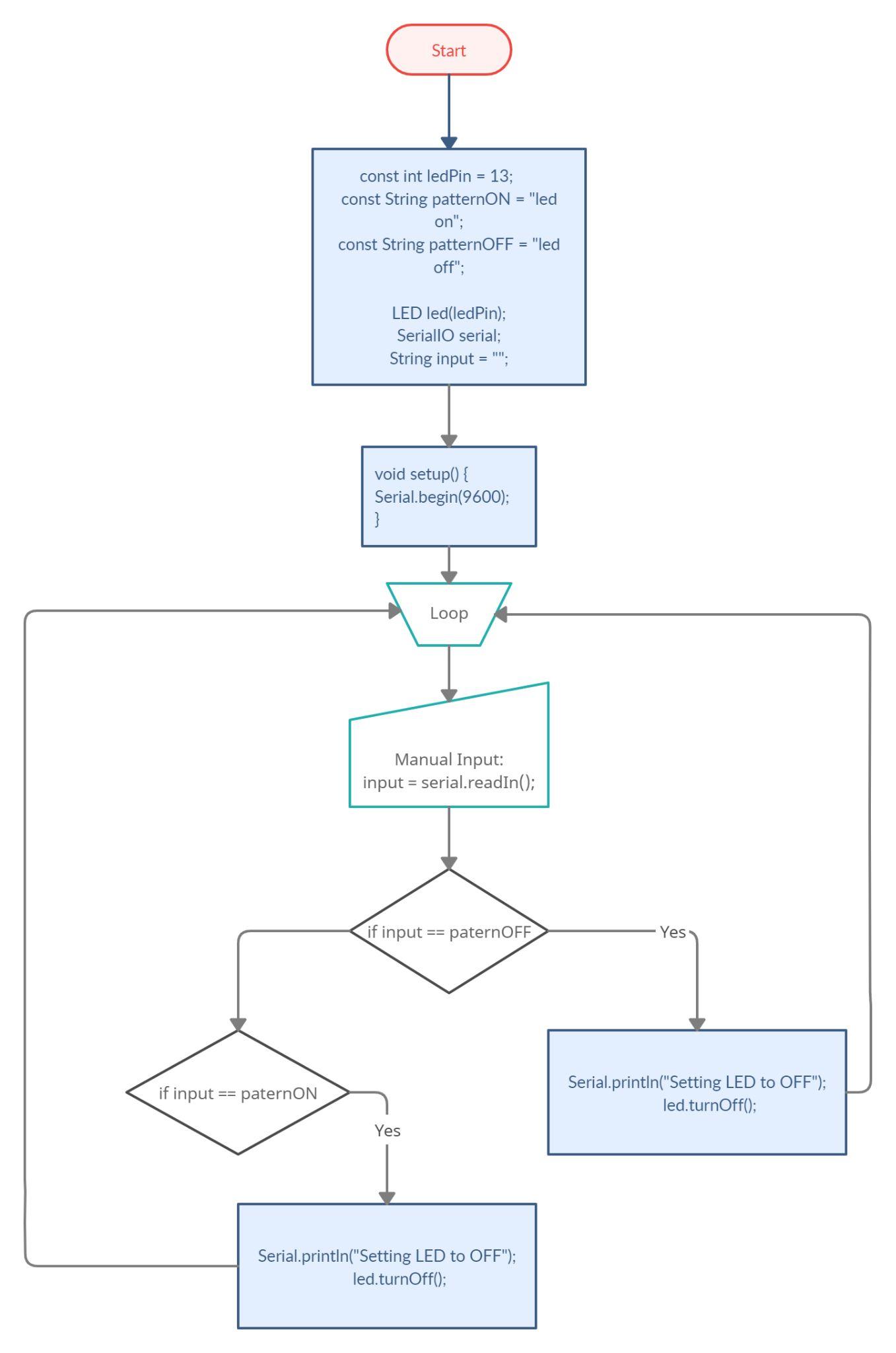
****

Fig 2.4 Flowchart for Task 2

* [**Video of real assembled equipment**](https://drive.google.com/file/d/1OlFAkAOXyIdmSRL5ViWxeb6uyjbmLe46/view?usp=sharing)

**Task 3**

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

To perform this task I have used 2 third-party libraries for LCD display: “LiquidCrystal i2c.h”[1]; and for keypad: “Keypad.h”[2]; also I have created my own library “SecuritySystem.h”.

“SecuritySystem.h” is a library that handles input from the keypad, gives commands to LCD to display the message, checks the password matching and etc. “SecuritySystem.h” has 2 ‘main’ functions: “initSetup()” and “initLoop()”.(see Appendix 6)

Appendix 6



Fig 3.1 “initSetup()” and “initLoop()” in “SecuritySystem.cpp”

“initSetup()” does all preparations on setting up the program, like defining pinMode for LEDs and initialization of LCD display. “initLoop()” is the function that repeats on each iteration of the loop in “Task3Main.ino”(see Appendix 7). It gets the input string from the keypad and handles it by sending to check if it is a special code for changing password, if not then it sends to check if it corresponds to the current password or not. If the input string is equal to password then the program displays message “Password accepted” and turns on green LCD, else it displays message “Password invalide” and turns on red LCD.

**Appendix 7**

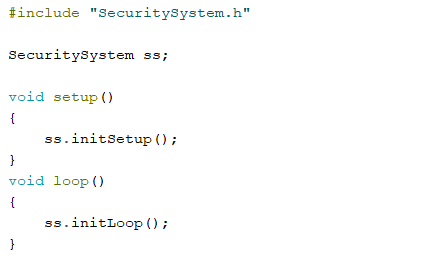


Fig 3.2 Task3Main.ino

* **Electrical scheme**

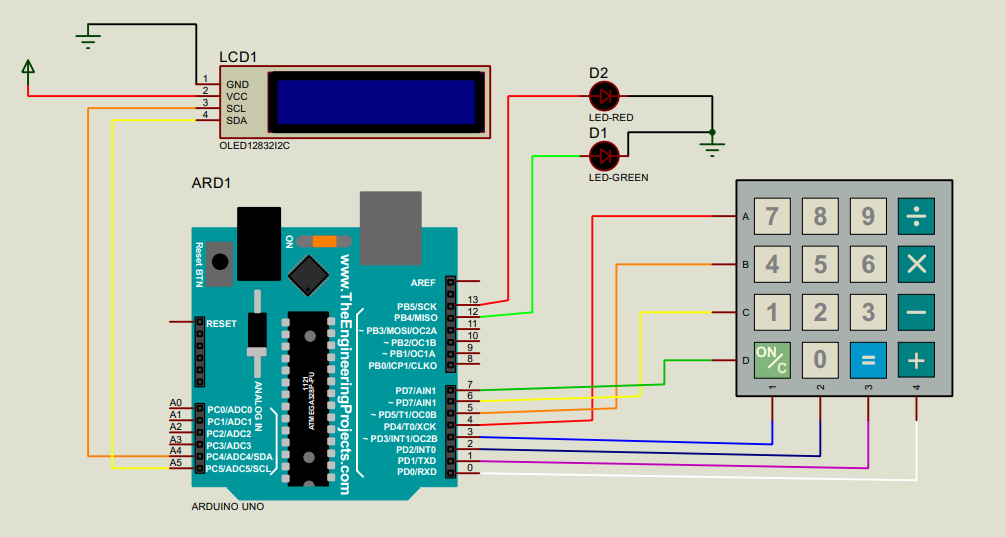
****

Fig 3.3 Electrical scheme for Task 3

* **Flowchart of the program**

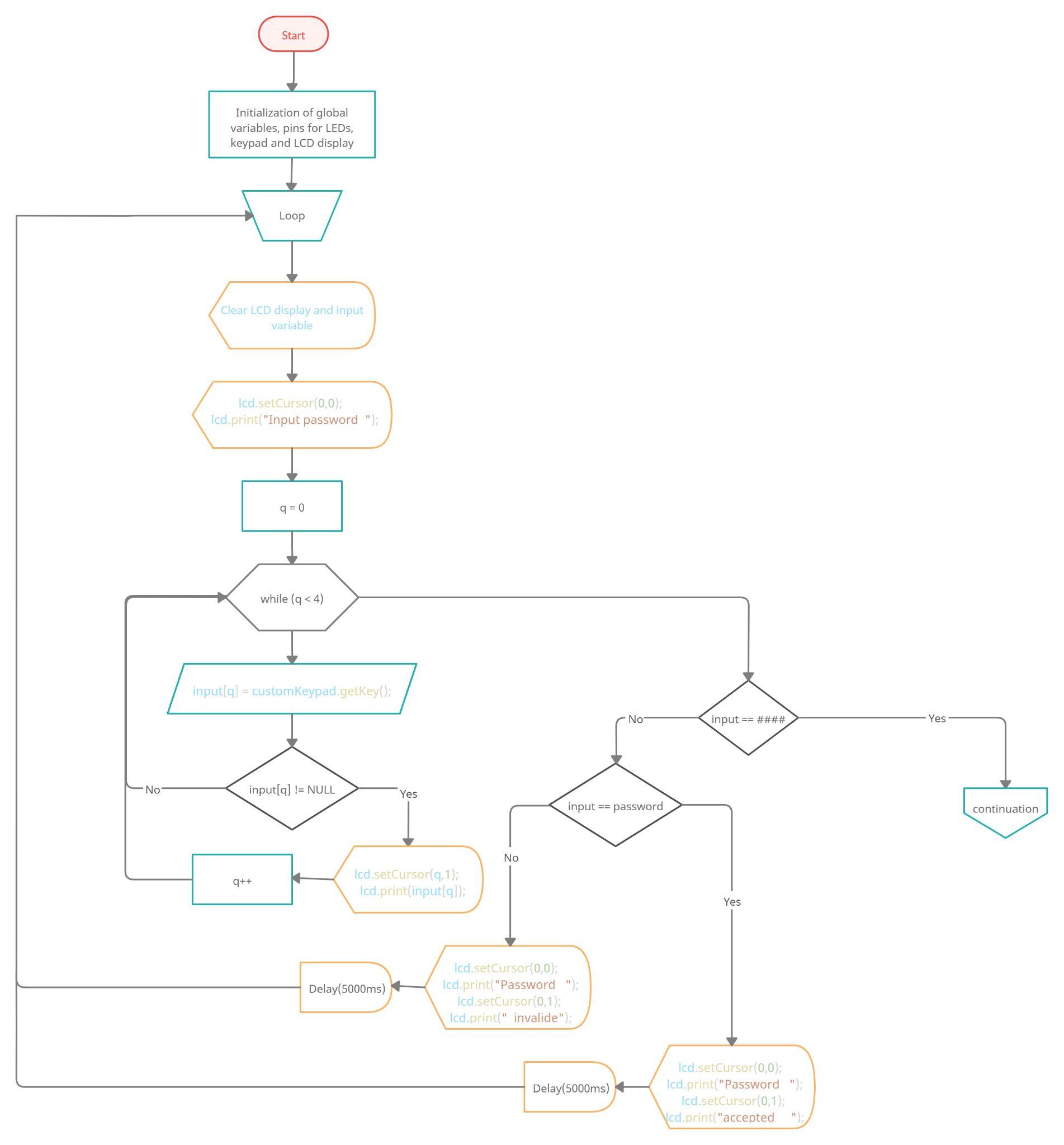
****

Fig 3.4 Flowchart for Task 3 part 1

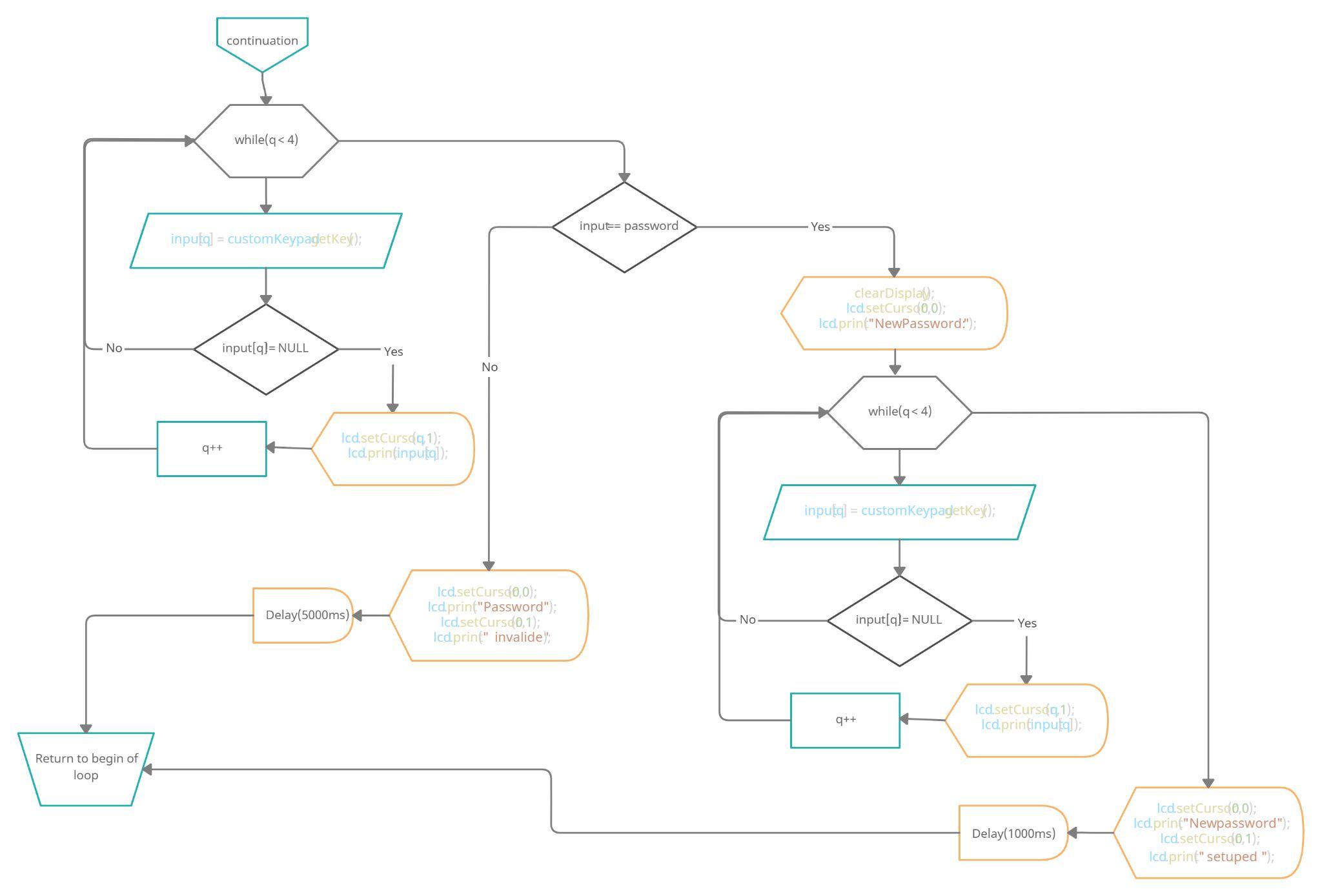


Fig 3.5 Flowchart for Task 3 part 2

* [**Video of real assembled equipment**](https://drive.google.com/file/d/1OqTrJHj3kUqhi4f7xtRRjEJhKCOD_hfS/view?usp=sharing)

**Conclusion**

During this laboratory work I restored in my memory the basic knowledge of programming on Arduino. Also I have learnt about such simulation software like Proteus, which I used for testing and designing electrical schemes for tasks of laboratory work.

**BIBLIOGRAPHY**

[1] LiquidCrystal I2C: official documentation page of the library. Access link: [LiquidCrystal I2C - Arduino Reference](https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/)

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

**Appendix**

* **Task1Main.ino**

#include "LED.h"

#include "Button.h"

const int buttonPin = 2; //pin for button

const int ledPin = 13; //pin for LED

LED led(ledPin);

Button button(buttonPin);

void setup() {}

void loop(){

if (button.readButton()) { //if button is pressed, than LED turns on

led.turnOn();

}

else { //else turns off

led.turnOff();

}

}

* Task2Main.ino

#include "LED.h"

#include "SerialIO.h"

const int ledPin = 13; //pin for LED

const String patternON = "led on";

const String patternOFF = "led off";

LED led(ledPin);

SerialIO serial;

String input = "";

void setup() {

Serial.begin(9600);

}

void loop() {

input = serial.readIn();

if (serial.check(input, patternOFF)) {

Serial.println("Setting LED to OFF");

led.turnOff();

}else if (serial.check(input, patternON)) {

Serial.println("Setting LED to ON");

led.turnOn();

}

}

* Task3Main.ino

#include "SecuritySystem.h"

SecuritySystem ss;

void setup()

{

ss.initSetup();

}

void loop()

{

ss.initLoop();

}

* Button.h

#ifndef Button\_h

#define Button\_h

#include <Arduino.h>

class Button

{

public:

Button(int pin);

bool readButton();

private:

int ButtonPin;

};

#endif

* Button.cpp

#include "Button.h"

Button::Button(int pin)

{

pinMode(pin, INPUT);

ButtonPin = pin;

}

bool Button::readButton(){

if (digitalRead(ButtonPin) == HIGH) return true;

else return false;

}

* LED.h

#ifndef LED\_h

#define LED\_h

#include <Arduino.h>

class LED

{

public:

LED(int pin);

void turnOn();

void turnOff();

private:

int LEDpin;

};

#endif

* LED.cpp

#include "LED.h"

LED::LED(int pin)

{

pinMode(pin, OUTPUT);

LEDpin = pin;

}

void LED::turnOn(){

digitalWrite(LEDpin, HIGH);

}

void LED::turnOff(){

digitalWrite(LEDpin, LOW);

}

* SerialIO.h

#ifndef SerialIO\_h

#define SerialIO\_h

#include <stdio.h>

#include <Arduino.h>

class SerialIO

{

public:

SerialIO();

String readIn();

bool check(String input, String pattern);

};

#endif

* SerialIO.cpp

#include "SerialIO.h"

SerialIO::SerialIO(){

}

String SerialIO::readIn(){

String input = "";

if (Serial.available() > 0) {

input = Serial.readStringUntil('\n');

Serial.print("Received: ");

Serial.println(input);

}

return input;

}

bool SerialIO::check(String input, String pattern){

if(input == pattern) return true;

else return false;

}

* SecuritySystem.h

#ifndef LED\_h

#define LED\_h

#include <Arduino.h>

#include <Keypad.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include "LED.h"

class SecuritySystem

{

public:

SecuritySystem();

void initSetup();

void initLoop();

private:

void clearDisplay();

void correct();

void invalide();

void getInput();

bool checkFeature();

void changePassword();

void inputPassword();

bool checkPassword();

void clearInput();

void setPassword();

};

#endif

* SecuritySystem.cpp

#include "SecuritySystem.h"

//#include "LED.cpp"

LiquidCrystal\_I2C lcd(0x27,20,4);

const byte ROWS = 4;

const byte COLS = 4;

const byte RedLED = 13;

const byte GreenLED = 12;

char hexaKeys[ROWS][COLS] =

{

{'1','2','3','a'},

{'4','5','6','b'},

{'7','8','9','c'},

{'\*','0','#','d'}

};

byte rowPins[ROWS] = {4,5,6,7};

byte colPins[COLS] = {3,2,1,0};

Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

char input[4] = {' ', ' ', ' ', ' '};

char password[4] = {'1', '3', '3', '7'};

/\*LED RedLed(RedLED);

LED GreenLed(GreenLED);\*/

SecuritySystem::SecuritySystem(){

}

void SecuritySystem::initSetup(){

pinMode(RedLED, OUTPUT);

pinMode(GreenLED, OUTPUT);

lcd.init();

lcd.backlight();

}

void SecuritySystem::initLoop(){

clearDisplay();

lcd.setCursor(0,0);

lcd.print("Input password ");

getInput();

if(checkFeature())

{ //if true -> change password

changePassword();

}

else //if false -> check password

{

if (checkPassword())

{

correct();

}

else

{

invalide();

}

}

}

void SecuritySystem::getInput(){

byte q = 0;

while(q < 4){

input[q] = customKeypad.getKey();

if (input[q] != NULL)

{

lcd.setCursor(q,1);

lcd.print(input[q]);

q++;

}

}

}

bool SecuritySystem::checkFeature(){

if (input[0] == '#' && input[1] == '#' && input[2] == '#' && input[3] == '#'){ //#### - special code for password changing

return true;

//changePassword();

}

else return false;

}

void SecuritySystem::changePassword(){

clearDisplay();

lcd.setCursor(0,0);

lcd.print("Old Password:");

//To setup new password we need to introduce old one

getInput();

if(checkPassword())

{

setPassword();

}

else

{

invalide();

}

}

bool SecuritySystem::checkPassword(){

lcd.setCursor(0,0);

lcd.print("Checking our ");

lcd.setCursor(0,1);

lcd.print(" password");

delay(300);

if (input[0] == password[0] && input[1] == password[1] && input[2] == password[2] && input[3] == password[3])

return true;

else

return false;

}

void SecuritySystem::setPassword(){

clearDisplay();

lcd.setCursor(0,0);

lcd.print("New Password: ");

getInput();

password[0] = input[0];

password[1] = input[1];

password[2] = input[2];

password[3] = input[3];

lcd.setCursor(0,0);

lcd.print("New password ");

lcd.setCursor(0,1);

lcd.print(" setuped ");

delay(1000);

}

void SecuritySystem::correct(){

lcd.setCursor(0,0);

lcd.print("Password ");

lcd.setCursor(0,1);

lcd.print("accepted ");

digitalWrite(GreenLED, HIGH);

delay(5000);

digitalWrite(GreenLED, LOW);

//opening();

}

void SecuritySystem::invalide()

{

lcd.setCursor(0,0);

lcd.print("Password ");

lcd.setCursor(0,1);

lcd.print(" invalide");

digitalWrite(RedLED, HIGH);

delay(5000);

digitalWrite(RedLED, LOW);

}

void SecuritySystem::clearDisplay(){

//clear display

lcd.setCursor(0,0);

lcd.print(" ");

lcd.setCursor(0,1);

lcd.print(" ");

//clear input

input[0] = ' ';

input[1] = ' ';

input[2] = ' ';

input[3] = ' ';

}