

Ministry of Education, Culture and Research of the Republic of Moldova

Technical University of Moldova

Department of Software and Automatic Engineering

# REPORT

Laboratory Project nr.1  
*at Embedded Systems*

Done by:  
Gr.172-FAF

Ion Dodon

Checked by:

Andrei Bragarenco

Chişinău 2020

## Laboratory Project Nr.1

### Topic: User interactions

**Objective:** To gain skills on implementing input/output functions in order to facilitate user interactions.

**Domain:** The user interface consists of the features by which a user interacts with a computer system. This includes screens, pages, buttons, icons, forms, etc. The most obvious examples of user interfaces are software and applications on computers and smartphones.

A user interface doesn't necessarily require a screen, however. For example, a TV remote has a user interface that consists of various buttons, and devices such as Amazon Echo can be controlled with voice commands.

The relationship between hardware and software is a central matter in embedded systems programming. Embedded systems programmers like to think that what they do is special, even unique, in the software development world. User interaction is reflecting the real user experience after one interaction with the user interface. Common types of such events are mouse clicks, touches and keyboard events.

### Component description:

The used components are:

**Arduino UNO Rev3** - The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable.

### Switch Push Button Single Pole -

Number of pins: 2  
Mounting hole diameter: 0.500"  
Contact form: SPST off-(on)  
Contact rating: 3A @ 125VAC  
Dimensions: 0.82"(D) x 1.02"(T)  
Lead spacing: 0.20"  
Terminal type: Solder  
Button color: Green

**Light-emitting diode (LED)** - In a light emitting diode, the recombination of electrons and electron holes in a semiconductor produces light (be it infrared, visible or UV), a process called "electroluminescence". The wavelength of the light depends on the energy band gap of the semiconductors used. Since these materials have a high index of refraction, design features of the devices such as special optical coatings and die shape are required to efficiently emit light.

**VSM Virtual Terminal** - The VSM Virtual Terminal enables you to use the keyboard and screen of your PC to send and receive RS232 asynchronous serial data to and from a simulated microprocessor system. It is especially useful in debugging where you can use it to display debug/trace messages generated by the software which you are developing.

*The Virtual Terminal is specified as follows:*

Fully bi-directional - received serial data is displayed as ASCII characters whilst key presses are transmitted as serial ASCII data.

Simple two wire serial data interface: RXD for received data and TXD for transmitted data.

Simple two wire hardware handshake interface: RTS for ready-to-send and CTS for clear-to-send.

Baud rate from 300 to 57,600 baud.

7 or 8 data bits.

Odd, even or no parity.

0, 1 or 2 stop bits.

XON/XOFF software handshaking in addition to hardware handshaking.

Normal or inverted polarity for both RX/TX and RTS/CTS signals.

**74HC595** - an "8-bit serial-in, serial or parallel-out shift register with output latches; 3-state." In other words, you can use it to control 8 outputs at a time while only taking up a few pins on your microcontroller. You can link multiple registers together to extend your output even more.

#### **4X4 matrix membrane keypad –**

16 Keys configured into a 4x4 matrix

Arduino and Raspberry Pi compatible

Switch pad has 8 pin female header on end of flat cable

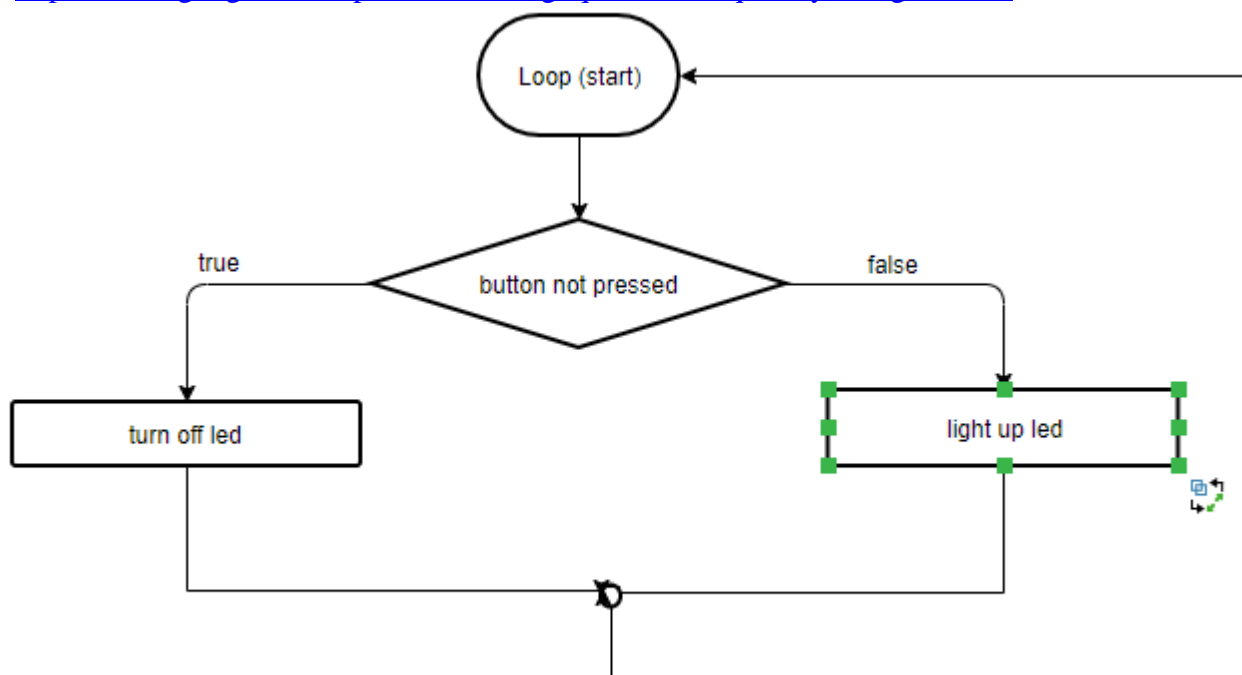
White sticker on back can be peeled off for adhesive mounting

**LM0161** - The Hitachi HD44780 LCD controller is an alphanumeric dot matrix liquid crystal display (LCD) controller developed by Hitachi. The character set of the controller includes ASCII characters, Japanese Kana characters, and some symbols in two 28 character lines. Using an extension driver, the device can display up to 80 characters.

#### **Implementation:**

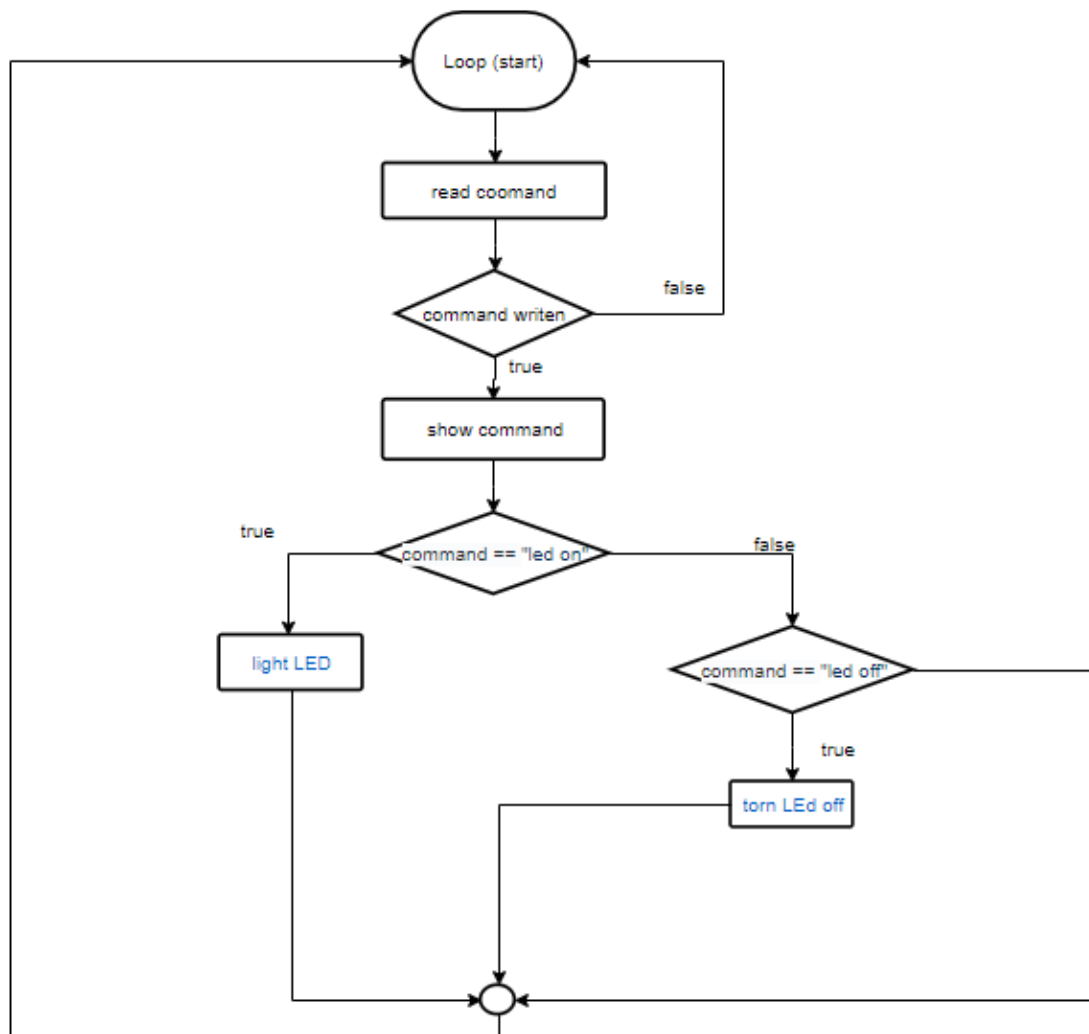
- a) Turn a led on and off using a button

[https://drive.google.com/open?id=1YVGg\\_qiDB0r7ZXq3CPIJy9kvugsLch1oI](https://drive.google.com/open?id=1YVGg_qiDB0r7ZXq3CPIJy9kvugsLch1oI)



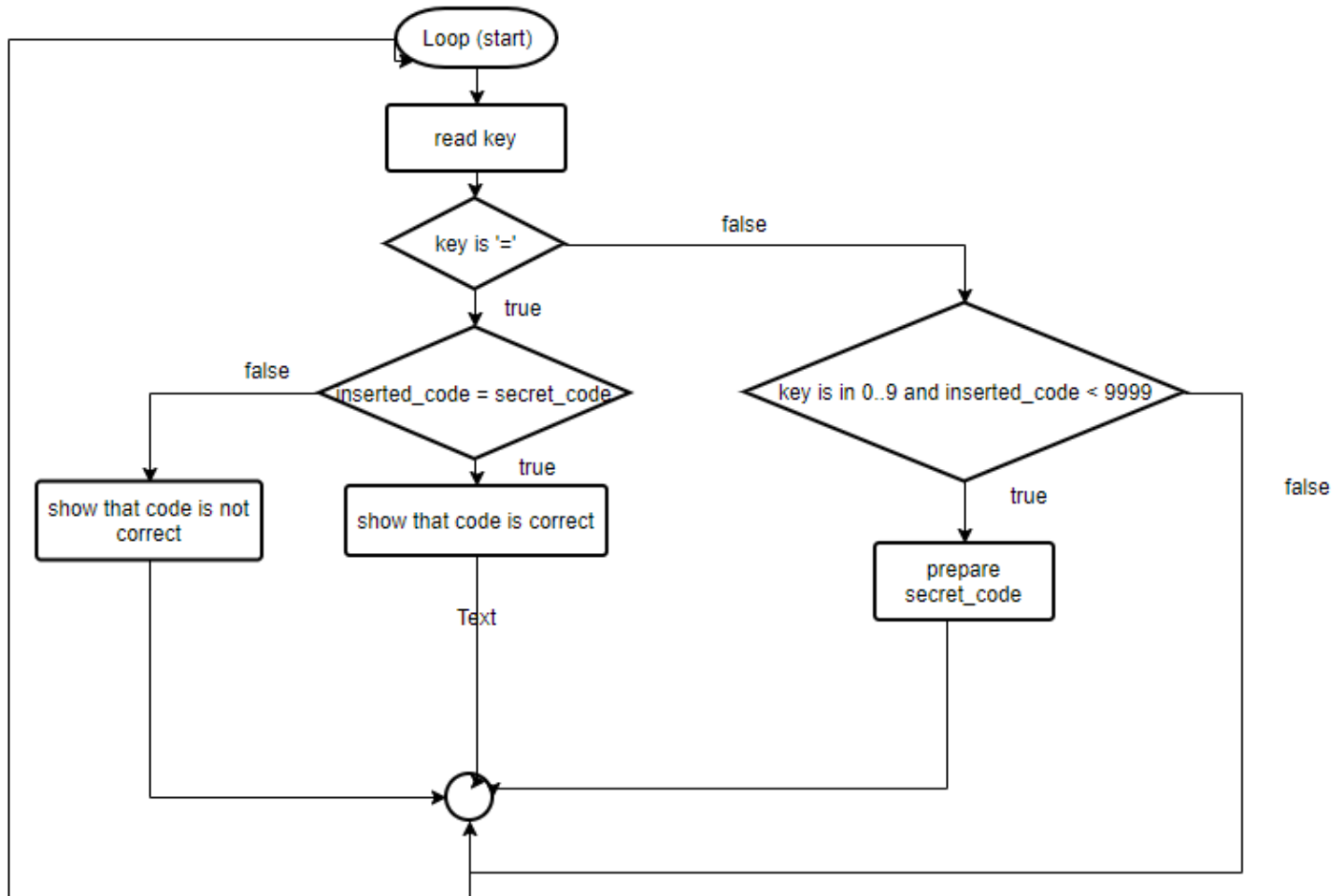
b) Turn a led on and of using terminal commands

<https://drive.google.com/open?id=13iXdfUzcB53ouB9tAPsrEBcbhsnPYPyA>



- c) Check a pin code and if the inserted code is correct light a green led, else light a red led. Also, at each verification show status on a LCD.

[https://drive.google.com/open?id=1XkUMcbKPdfl4QjQh0i0rBuQizIhdjS\\_R](https://drive.google.com/open?id=1XkUMcbKPdfl4QjQh0i0rBuQizIhdjS_R)



**Conclusions:** During this laboratory work I learned how to implement different types of user interactions for an Android Uno. The used user interface hardware components are LCD, 4x4 KeyPad, Virtual Terminal and a Button. I have learned how use Proteus simulator, how to connect different components to Arduino and how to program it.

## Annex:

- a) Turn a led on and off using a button

```
#include "button.h"
#include "led.h"

Button *btn;
Led *led;

void setup() {
    btn = new Button(4);
    led = new Led(7);
}

void loop() {
    if(btn->pressed()) {
        led->turnOn();
    } else {
        led->turnOff();
    }
}

Button::Button(int digitalPin) {
    this->digitalPin = digitalPin;
    pinMode(digitalPin, INPUT_PULLUP);
}

bool Button::pressed() {
    if(digitalRead(this->digitalPin) == LOW) {
        return true;
    }
    return false;
}

Led::Led(int digitalPin) {
    this->digitalPin = digitalPin;
    pinMode(digitalPin, OUTPUT);
}

void Led::turnOn() {
    digitalWrite(this->digitalPin, HIGH);
}

void Led::turnOff() {
    digitalWrite(this->digitalPin, LOW);
}

#ifdef BUTTON_HEADER
#define BUTTON_HEADER

#include <Arduino.h>

class Button {
private:
    int digitalPin;
public:
    Button(int digitalPin);
    bool pressed();
};

#endif

#ifdef LED_HEADER
#define LED_HEADER

#include <Arduino.h>

class Led {
private:
    int digitalPin;
public:
    Led(int digitalPin);
    void turnOn();
    void turnOff();
};

#endif
```

b) Turn a led on and of using terminal commands

```
#include "mystdio.h"
#include "ledd.h"

Mystdio io;
Ledd *ledd = new Ledd(12);

void setup() {
    Serial.begin(9600);
}

void loop() {
    String command = io.readStr();

    if(command.length()) {
        Serial.println(command);
        if(command == "led on") {
            ledd->turnOn();
            printf("LED is on\r");
        } else
        if(command == "led off") {
            ledd->turnOff();
            printf("LED is off\r");
        }
    }
}

#include "ledd.h"

Ledd::Ledd(int digitalPin) {
    this->digitalPin = 12;
    pinMode(this->digitalPin, OUTPUT);
    this->on = false;
}

bool Ledd::isOn() {
    return this->on;
}

void Ledd::turnOn() {
    digitalWrite(this->digitalPin, HIGH);
    this->on = true;
}

void Ledd::turnOff() {
    digitalWrite(this->digitalPin, LOW);
    this->on = false;
}

#ifdef LEDD_HEADER
#define LEDD_HEADER

#include <Arduino.h>

class Ledd {
private:
    int digitalPin;
    bool on;

public:
    Ledd(int digitalPin);
    void turnOn();
    void turnOff();
    bool isOn();
};

#endif

#ifdef MUSTDIO_HEADER
#define MUSTDIO_HEADER

#include <stdio.h>
#include <Arduino.h>

static FILE *f;

class Mystdio {
public:
    Mystdio();
    String readStr();
    void writeStr(String str);
    static int putChar(char c, FILE *stream);
    static char getChar(FILE *stream);
};

#endif
```

```

#include "mystdio.h"

static int Mystdio::putChar(char c, FILE *stream)
{
    Serial.write(c) ;
    return 0 ;
}

static char Mystdio::getChar(FILE *stream)
{
    char c;
    while(Serial.available()) {
        c = Serial.read();
        printf("%c", c);
    }
    return c;
}

String Mystdio::readStr() {
    char c;
    String str;
    do {
        scanf("%c", &c);
        if(c != 0 && c != '\r'){
            str.concat(c);
        }
    }while(c!= '\r');

    return str;
}

void Mystdio::writeStr(String str) {
    for(int i = 0; i < str.length(); i++) {
        printf("%c", str[i]);
    }
}

Mystdio::Mystdio() {
    f = fdevopen(Mystdio::putChar, Mystdio::getChar);
    stdout = f;
    stdin = f;
}

```



- c) Check a pin code and if the inserted code is correct light a green led, else light a red led. Also, at each verification show status on a LCD

```
#include "global.h"

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
LiquidCrystal lcd(10);

void setup() {
    // set up the LCD's number of columns and rows:
    lcd.begin(16, 2);
    pinMode(LED_RED, OUTPUT);
    pinMode(LED_GREEN, OUTPUT);
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_GREEN, LOW);
    // Print a message to the LCD.
    // lcd.cursor();
}

void loop() {
    char key = keypad.getKey();

    if(key) {
        if(key == 61) {
            lcd.setCursor(1, 1);
            if(inserted_code == secret_code) {
                lcd.print("Correct");
                digitalWrite(LED_GREEN, HIGH);
                digitalWrite(LED_RED, LOW);
            } else {
                lcd.print("Incorrect");
                digitalWrite(LED_GREEN, LOW);
                digitalWrite(LED_RED, HIGH);
            }
            lcd.display();
            delay(1000);
            lcd.clear();
            inserted_code = 0;
        } else if(key >= 48 && key <= 57 && inserted_code * 10 < 9999) {
            inserted_code = inserted_code * 10 + (key - 48);
            lcd.setCursor(1, 1);
            lcd.print(String(inserted_code));
            lcd.display();
        }
    }
}
```

```

#ifndef HEADER_FILE
#define HEADER_FILE

#include<Keypad.h>
#include <SPI.h>
#include "LiquidCrystal.h"

#define LED_RED 15
#define LED_GREEN 14

const byte ROWS = 4; // four rows
const byte COLS = 4; // four columns

char keys[ROWS][COLS] = {
  {'7', '8', '9', '/'},
  {'4', '5', '6', '*'},
  {'1', '2', '3', '-'},
  {'c', '0', '=', '+'}
};

byte rowPins[ROWS] = {2, 3, 4, 5}; // connect to rows pinouts of the keypad
byte colPins[COLS] = {6, 7, 8, 9}; // connect to column pinouts of the keypad

int i = 0;
char arr[10];

int secret_code = 1234;
int inserted_code = 0;

- - -

```