**Ministerul Educației și Cercetării al Republicii Moldova**
**Universitatea Tehnică a Moldovei**
**Facultatea Calculatoare, Informatică și Microelectronică**
**Departamentul Ingineria Software și Automatică**

# Lucrare de laborator Nr. 1

Disciplina: Internetul lucrurilor (IOT)
Tema: Interacțiunea cu Utilizatorul

**A efectuat:** Popa Cătălin, gr. TI-211

**A verificat:** Cristian Lupan, asist. univ.

**Chișinău 2024**

**Problema propusă:**

1. Să se proiecteze o aplicație în bază de MCU care ar schimba starea unui LED la detectarea apăsării unui buton. De simulat funcționarea aplicației într-un simulator.
2. Să se configureze aplicația pentru lucrul cu librăria STDIO prin interfața seriala pentru schimbul de text prin terminal.
3. Să se proiecteze o aplicație în baza de MCU care ar recepționa comenzi de la terminal prin interfața serială pentru a seta starea unui LED.
    - ***led on*** pentru aprindere și ***led off*** pentru stingere. Sistemul trebuie sa răspundă cu mesaje text despre confirmarea comenzii.
    - pentru schimbul de text prin terminal a se utiliza librăria STDIO
4. Să se proiecteze o aplicație in baza de MCU pentru detectarea unui cod de la o tastatură 4x4, sa verifice codul si sa afișeze mesaj la un LCD.
    - pentru cod valid sa se aprindă un led de culoare verde, pentru cod invalid, un led de culoare roșie.
    - A se utiliza STDIO pentru scanarea tastaturii si afișare la LCD.

**Obiective**

1. Simularea aplicației într-un simulator de circuite.
2. Utilizarea bibliotecii STDIO pentru interacțiunea cu utilizatorul.
3. Proiectarea unei aplicații bazate pe microcontroler (MCU).

**Introducere**

Microcontrolerele permit controlul și monitorizarea dispozitivelor fizice, aducând tehnologia mai aproape de utilizator prin intermediul unei interfețe intuitive și simplificate. Un exemplu clasic de interacțiune între utilizator și dispozitive fizice este controlul unui LED prin apăsarea unui buton sau trimiterea de comenzi printr-o interfață serială. Această tehnologie este utilizată într-o gamă largă de aplicații, de la sisteme de iluminat inteligente până la dispozitive electronice complexe utilizate în automatizări industriale. Unul dintre avantajele microcontrolerelor este costul redus și flexibilitatea lor, permițând implementarea rapidă de prototipuri și teste pentru diverse aplicații IoT. Microcontrolerele precum cele din familia Arduino sau ESP sunt folosite frecvent pentru prototiparea sistemelor IoT datorită ecosistemului extins de biblioteci și comunități de dezvoltatori care susțin aceste platforme .

## Probleme actuale

**Complexitatea interacțiunii cu utilizatorul** – implementarea unor interfețe de control eficiente și intuitive, precum utilizarea unui terminal sau a unui ecran LCD, poate fi dificilă pentru începători.

**Integrarea hardware-software** – sincronizarea perfectă dintre partea hardware și cea software este esențială în asigurarea funcționării corecte a sistemului.

**Testarea și simularea** – înainte de a trece la dezvoltarea pe un circuit fizic, este esențial să se simuleze funcționarea circuitului.

## Studiu de caz

Un exemplu relevant este utilizarea unui microcontroler pentru controlul unui LED prin intermediul unui buton sau a unei interfețe seriale. Aceasta este o problemă comună pentru începători în domeniul IoT și este adesea întâlnită în laboratoarele educaționale. Prin utilizarea unei interfețe seriale (precum STDIO) pentru a primi comenzi, se poate dezvolta o aplicație simplă dar eficientă de control al unui LED, permițând astfel utilizatorilor să exploreze și să înțeleagă modul de comunicare și control între software și hardware .

## Materiale și metode

**Arduino UNO -** platformă de microcontroler open-source bazată pe microcontrolerul ATmega328P. Este ușor de utilizat și permite conectarea ușoară a componentelor externe prin pini digitali și analogici. Acesta este ideal pentru prototiparea de sisteme IoT și proiecte electronice interactive. Afișarea este în figura 1.1.

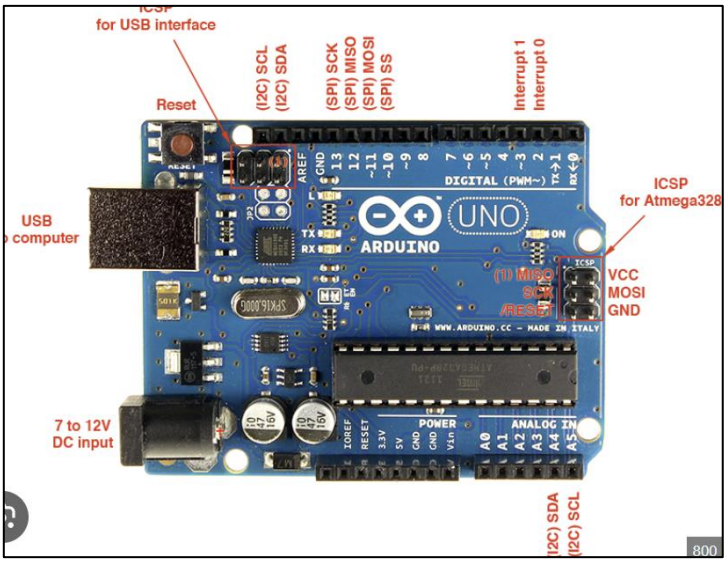



**Fig. 1.1. Arduino UNO**

- 14 pini digitali I/O (din care 6 sunt PWM);
- 6 intrări analogice;
- memorie flash de 32 KB;
- tensiune de operare: 5V;
- interfață de programare: Arduino IDE.

**LED-ul -** (diodă emițătoare de lumină) este utilizată pentru a semnaliza starea sistemului în funcție de comenzile primite sau de validitatea codului introdus pe tastatură. Afișarea este în figura 1.2.

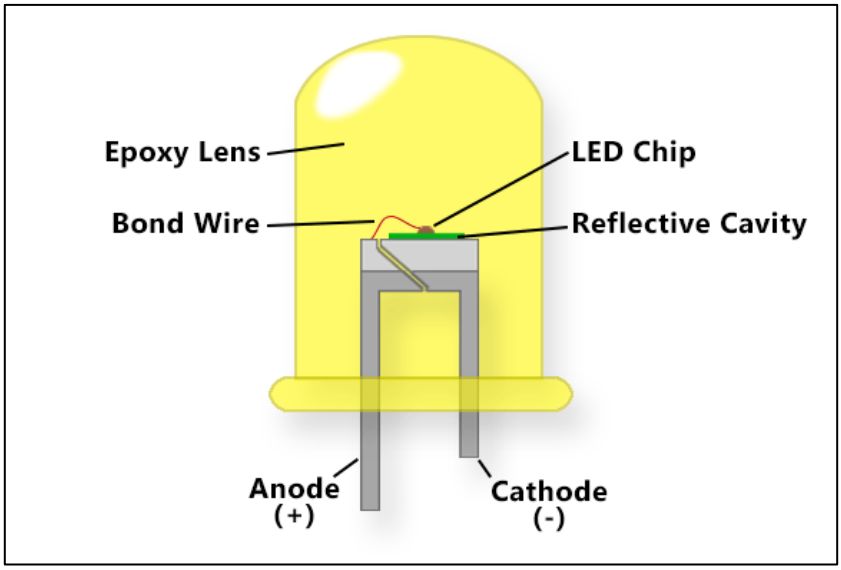



**Fig. 1.2. LED**

- tensiune de operare: 2V-3V;
- curent: ~20mA;
- LED verde pentru semnalizarea codurilor valide;
- LED roșu pentru semnalizarea codurilor invalide.

**Rezistor -** Rezistorul este utilizat pentru a limita curentul ce trece prin LED, prevenind arderea acestuia. Pentru a proteja LED-urile, se vor folosi rezistoare de aproximativ 220Ω. Afișarea este în figura 1.3.

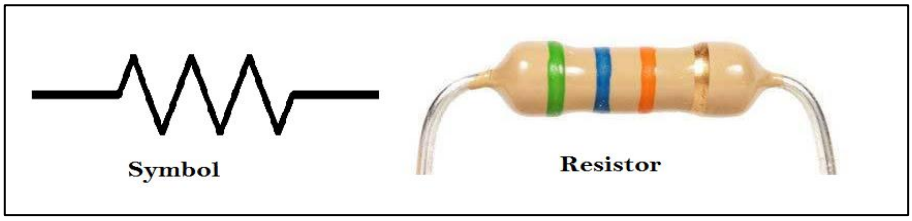



**Fig. 1.3. Rezistor**

**LCD(Liquid Crystal Display) -** LCD-ul va fi utilizat pentru a afișa mesajele text corespunzătoare comenzilor și interacțiunilor din sistem, inclusiv mesajele de confirmare a stării LED-urilor (aprins/oprit) sau pentru afișarea mesajelor în cazul validării unui cod. Afișarea este în figura 1.4.


**Fig. 1.4. LCD**

- display de 16x2 caractere;
- tensiune de operare: 5V;
- interfață paralelă (sau I2C, în funcție de model).

**Keypad 4x4** - Tastatura matricială 4x4 este formată din 16 butoane aranjate în 4 rânduri și 4 coloane, folosită pentru a introduce comenzi și coduri de acces. Microcontrolerul va scana rândurile și coloanele pentru a detecta care dintre butoane a fost apăsat. Afișarea este în figura 1.5.


**Fig. 1.5. Keypad 4x4**

- 16 butoane (4 rânduri x 4 coloane);
- interfață digitală (utilizează pini digitali pentru a citi intrările);
- utilizată pentru introducerea codurilor și trimite informațiile către Arduino pentru validare.

**Fire Jumper -** Firele jumper sunt utilizate pentru a face legături electrice între diverse componente și plăcuța Arduino. Acestea permit conectarea componentelor fără a fi necesară lipirea acestora. Afișarea este în figura 1.6.

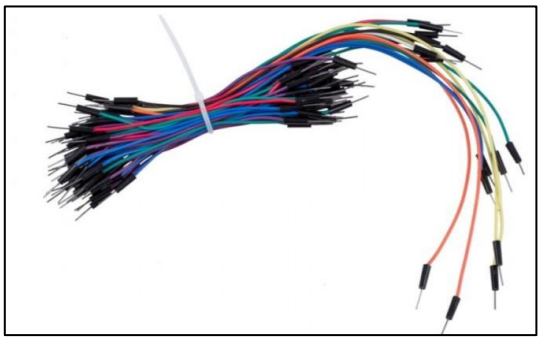


**Fig. 1.6. Fire Jumper**

**Buton -** În aplicații de acest gen, un buton (push button) este utilizat pentru a oferi utilizatorului o modalitate de a interacționa cu sistemul printr-un eveniment de apăsare. În această situație, ar putea fi utilizat pentru a reseta sistemul sau pentru a iniția o acțiune nouă. Afișarea este în figura 1.7.

**Fig. 1.7. Buton**

**Metode**

**Configurarea Arduino UNO:** Arduino va fi configurat folosind Arduino IDE. Se va implementa codul pentru a citi inputurile de la tastatura 4x4 și pentru a controla LED-urile și afișarea pe LCD.

**Interacțiunea cu utilizatorul:** Sistemul va primi comenzi fie prin terminalul serial, fie prin tastatura matricială, și va răspunde prin intermediul LCD-ului și al LED-urilor. Comenzile "led on" și "led off" vor aprinde și stinge LED-ul respectiv, iar codurile introduse vor fi validate, aprinzând LED-ul verde pentru coduri corecte și LED-ul roșu pentru coduri greșite.

**Validare în simulator:** Codul și conexiunile vor fi testate inițial într-un simulator de circuite, cum ar fi Tinkercad sau Proteus, pentru a verifica funcționalitatea corectă. După simulare, se va testa și pe un circuit real.

Schema ...

Pentru realizarea aplicației în baza MCU care ar schimba starea unui LED la dectarea apăsării unui buton, am avut nevoie de un simulator, led, buton și tranzitor. Mai jos în figura 1.7 este afișată implementarea sarciniii respective în simulator.

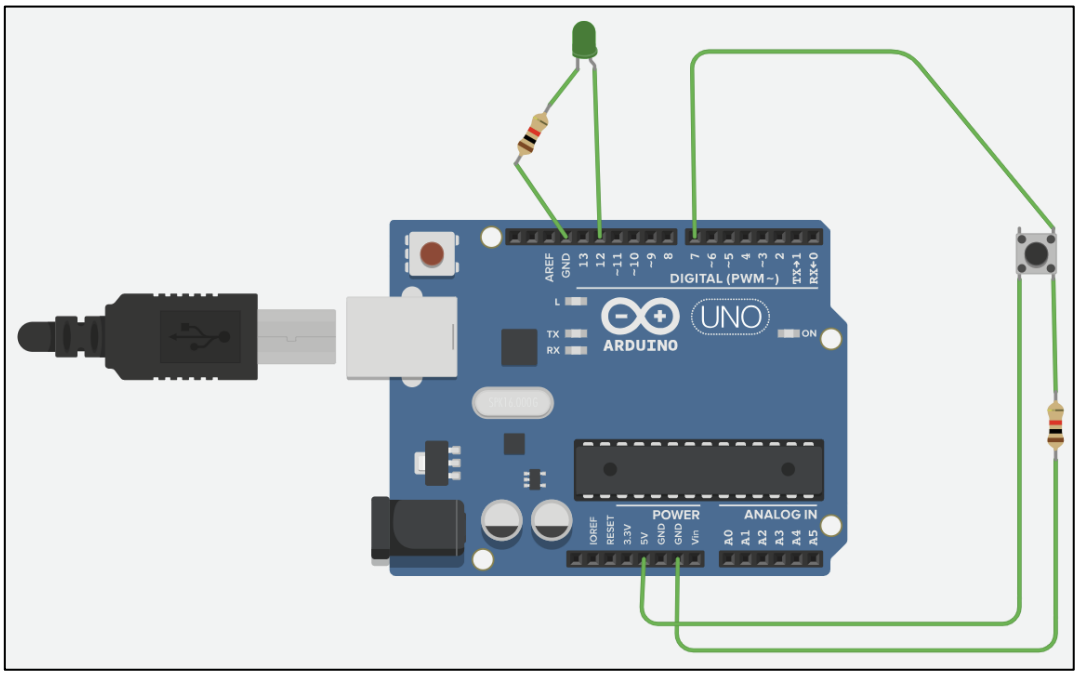



**Fig. 1.7. Schimb stare led la acțiune pe buton**

În partea de cod, mai întâi setăm modurile pentru pin. Ledul va fi de tip OUTPUT, iar butonul de tip INPUT.

```
void setup() {
pinMode(ledPin, OUTPUT);
pinMode(buttonPin, INPUT);
}
```

```
void loop() {
buttonState = digitalRead(buttonPin);
if (buttonState == HIGH && lastButtonState == LOW) {
ledState = !ledState;
digitalWrite(ledPin, ledState);
}
lastButtonState = buttonState;
delay(50);
}
```

În a doua metoda loop(), se va verifica mereu mai întâi care este starea la buton, prin digitalRead(). Apoi verificăm dacă starea curentă este activă și ultima stare a fost LOW, dacă da, atunci ledul se va activa și se va schimba starea la buton. La final avem un delay, pentru a nu forma sistemul.

În a doua sarcină, am creat un microcontroler care permite controlul unui LED conectat la pinul 12 prin comenzi seriale trimise dintr-un monitor serial.

Se setează inițial ledPin,  fiind ca ieșire prin pinMode. Configurăm comunicarea serială la o viteză de 9600 baud și configurăm funcțiile Serial_write și Serial_read pentru a redirecționa I/O-ul serial prin funcțiile standart de I/O.

```
void setup() {
    pinMode(ledPin, OUTPUT);
    Serial.begin(9600);
    fdevopen(Serial_write, Serial_read);
     }
```

În funcția loop() verificăm dacă sunt disponibile date prin conexiunea serială. Se citește comanda trimisă până la un caracter newline, apoi elimină spațiile goale. Dacă comanda este *led on*, LED-ul este arpins, iar dacă comanda este *led off*, LED-ul este stins, altfel se va afișa un mesaj de eroare.

```
void loop() {
    if (Serial.available() > 0) {
        String command = Serial.readStringUntil('\n');
        command.trim();
        if (command.equals("led on")) {
```

```
            digitalWrite(ledPin, HIGH);
            Serial.println("LED aprins.");
        } else if (command.equals("led off")) {
            digitalWrite(ledPin, LOW);
            Serial.println("LED stins.");
        } else {
            Serial.println("Comanda necunoscuta.");
        }
    }
    }
```

Simularea microcontrolerului este afișată în figura 1.9.

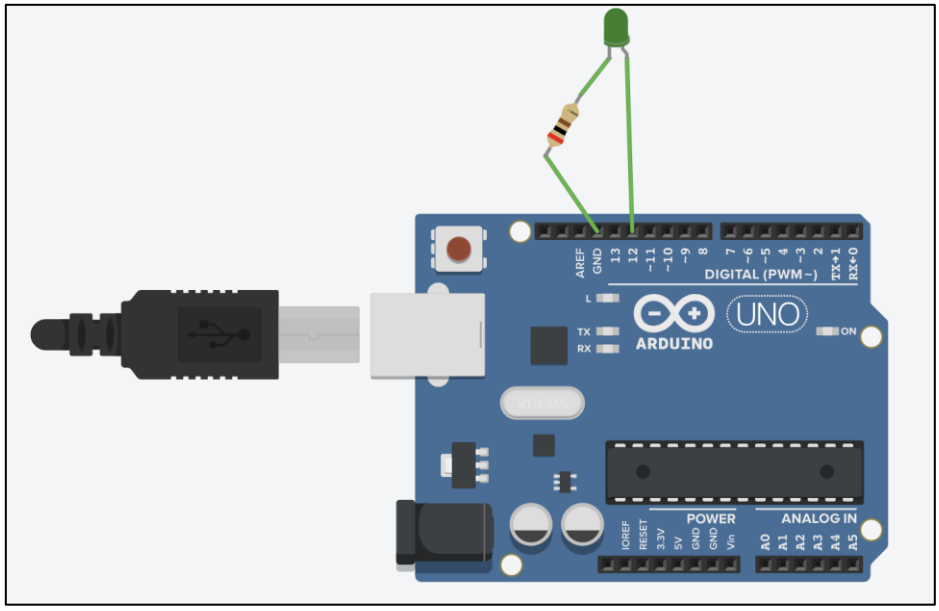



**Fig. 1.9. Schimbare stare LED prin comandă**

În ultimul item, am simulat o verificare a corectitudinii codului de acces introdus. Inițial includem biblioteca pentru controlul ecranului LCD Adafruit_LiquidCrystal și cea pentru keypad. Configurăm pinii pentru LED-ul verde și LED-ul roșu. Apoi definim o hartă de taste pentru keypad și configurăm conexiunile rândurilor și coloanelor.

În setup setăm pinii pentru LED-uri ca ieșire. Inițializăm și afișăm mesajul cu Introdu parola pentru 1,5 secunde, apoi ecranul este golit.

```
void setup()
    {
```

```
   pinMode(ledGreen, OUTPUT);
   pinMode(ledRed, OUTPUT);
   lcd_1.begin(16, 2);
   lcd_1.print("Introdu Parola");
   delay(1500);
   lcd_1.clear();
   }
```

În bucla principală loop, mai întâi are loc citirea tastelor apăsate de utilizator folosind funcția readKeypad. Dacă este apăsat butonul # , se va începe verificarea parolei introduse. Dacă parola este corectă, se va afișa pe LCD textul cu parola corectă și se va aprinde LED-ul verde, dacă parola este greșită se va aprinde LED-ul roșu. Dacă se apasă butonul *, parola introdusă este resetată și ecranul este golit.

```
   void loop()
   {
   readKeypad();
   if (keypressed == '#') {
  if (pad == password ) {
      digitalWrite(ledGreen, HIGH);
   lcd_1.setCursor(0, 1);
         lcd_1.print("Parola corecta");
  } else {
   lcd_1.setCursor(0, 1);
         lcd_1.print("Parola gresita");
    digitalWrite(ledRed, HIGH);
  }
} if (keypressed == '*') {
   pad = "";
  lcd_1.clear();
}
lcd_1.setCursor(0, 0);
lcd_1.print(pad);
delay(10);
```

```
  digitalWrite(ledGreen, LOW);
  digitalWrite(ledRed, LOW);
     }
```

Simularea microcontrolerului este afișată în figura 2.

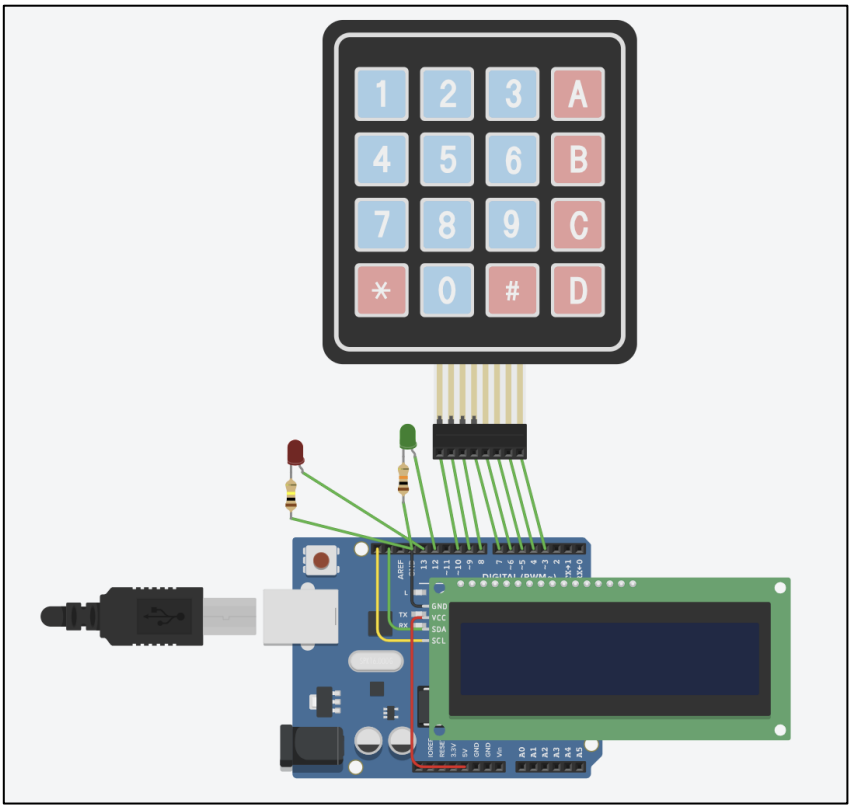



**Fig. 2. Verificare parolă**

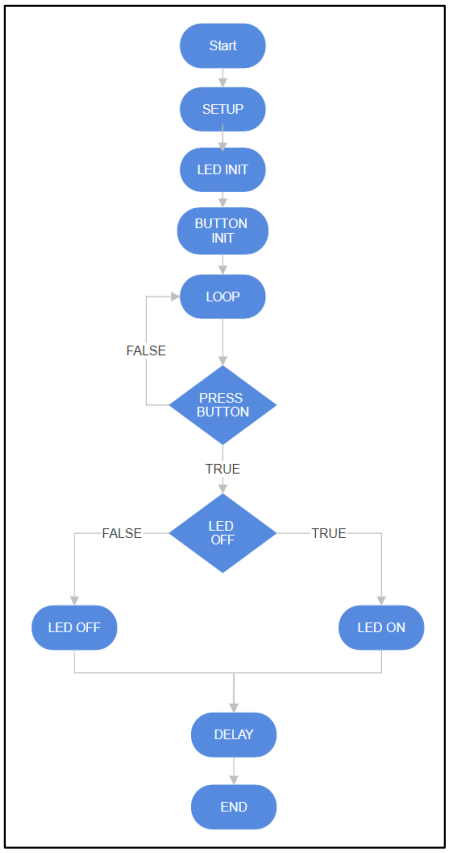
## Concluzii

Obiectivul principal a fost explorarea comunicării dintre utilizator și microcontroler, atât prin comenzi text, cât și printr-o tastatură fizică, iar rezultatele au demonstrat o interacțiune eficientă între hardware și software. De asemenea, s-a reușit simularea și testarea corectă a funcționalității LED-urilor, oferind utilizatorului un răspuns imediat la fiecare comandă. Această lucrare subliniază importanța microcontrolerelor în dezvoltarea de prototipuri IoT și oferă o experiență nouă pentru înțelegerea principiilor fundamentale de interacțiune. În rezultat, s-a creat o viziune clară cum sunt create microcontrolerele și metodele de utilizare pentru a rezolva o problemă reală.

**Anexă**

1.0

Schema bloc pentru led cu buton.



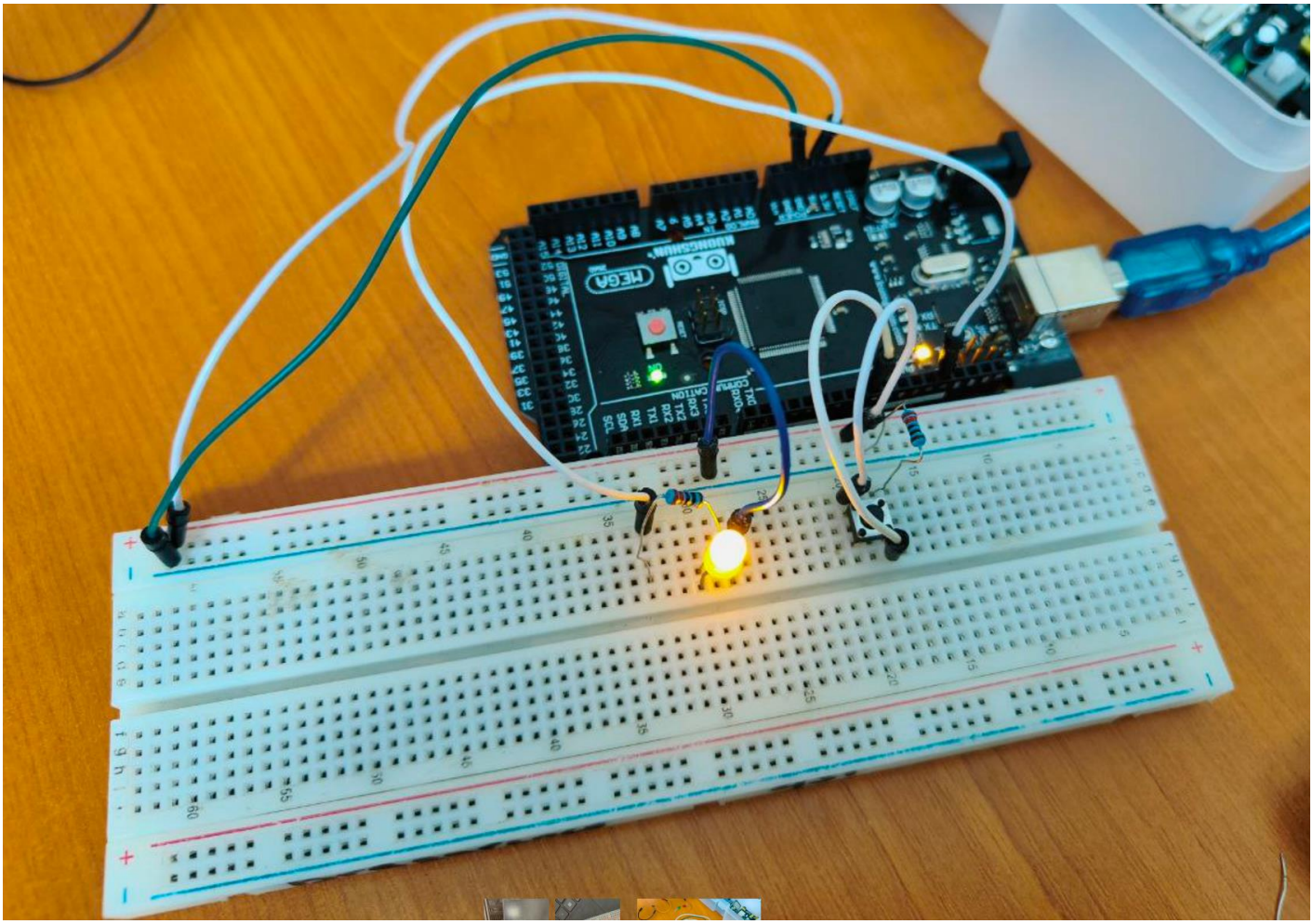


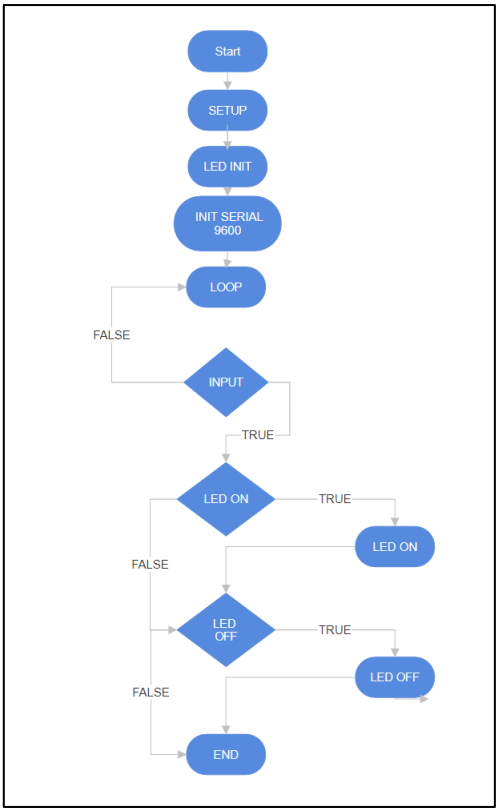
Fig1.1 – Sarcina 1

```
int ledPin = 12;
int buttonPin = 7;
int ledState = LOW;
int buttonState;
int lastButtonState = LOW;
void setup() {
pinMode(ledPin, OUTPUT);
pinMode(buttonPin, INPUT);
}
void loop() {
buttonState = digitalRead(buttonPin);
if (buttonState == HIGH && lastButtonState == LOW) {
```

```
ledState = !ledState;
digitalWrite(ledPin, ledState);
}
lastButtonState = buttonState;
delay(50);
}
```

1.1

Schema bloc pentru activare LED prin comanda.



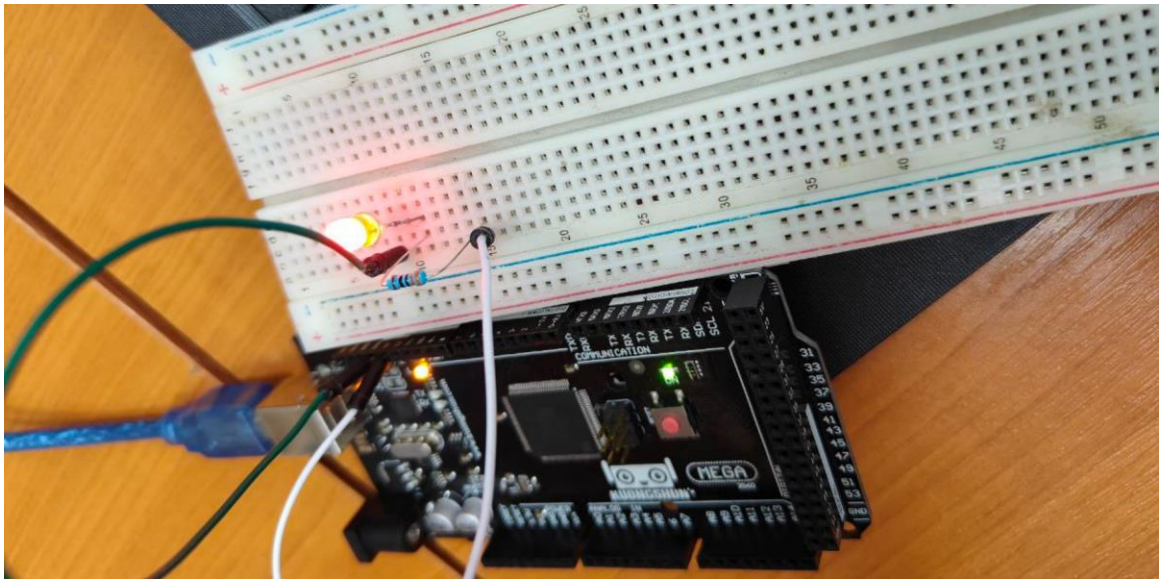


Fig.1.2 – Sarcina 2

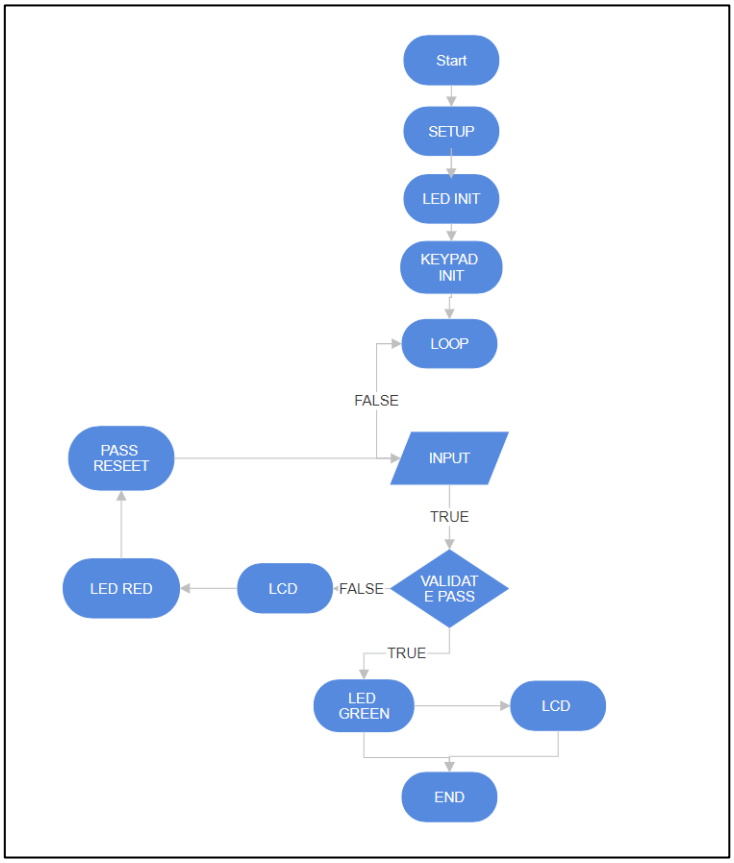
```
  #include <stdio.h>
  const int ledPin = 12;
 void setup() {
pinMode(ledPin, OUTPUT);
Serial.begin(9600);
fdevopen(Serial_write, Serial_read);
 }

 void loop() {
if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command.equals("led on")) {
        digitalWrite(ledPin, HIGH);
        Serial.println("LED aprins.");
    } else if (command.equals("led off")) {
        digitalWrite(ledPin, LOW);
        Serial.println("LED stins.");
    } else {
        Serial.println("Comanda necunoscuta.");
```

```
    }
}
 }
 int Serial_write(char c, FILE* stream) {
Serial.write(c);
return 0;
 }
 int Serial_read(FILE* stream) {
while (!Serial.available());
return Serial.read();
 }
```

1.2

Schema bloc pentru verificarea parolei.



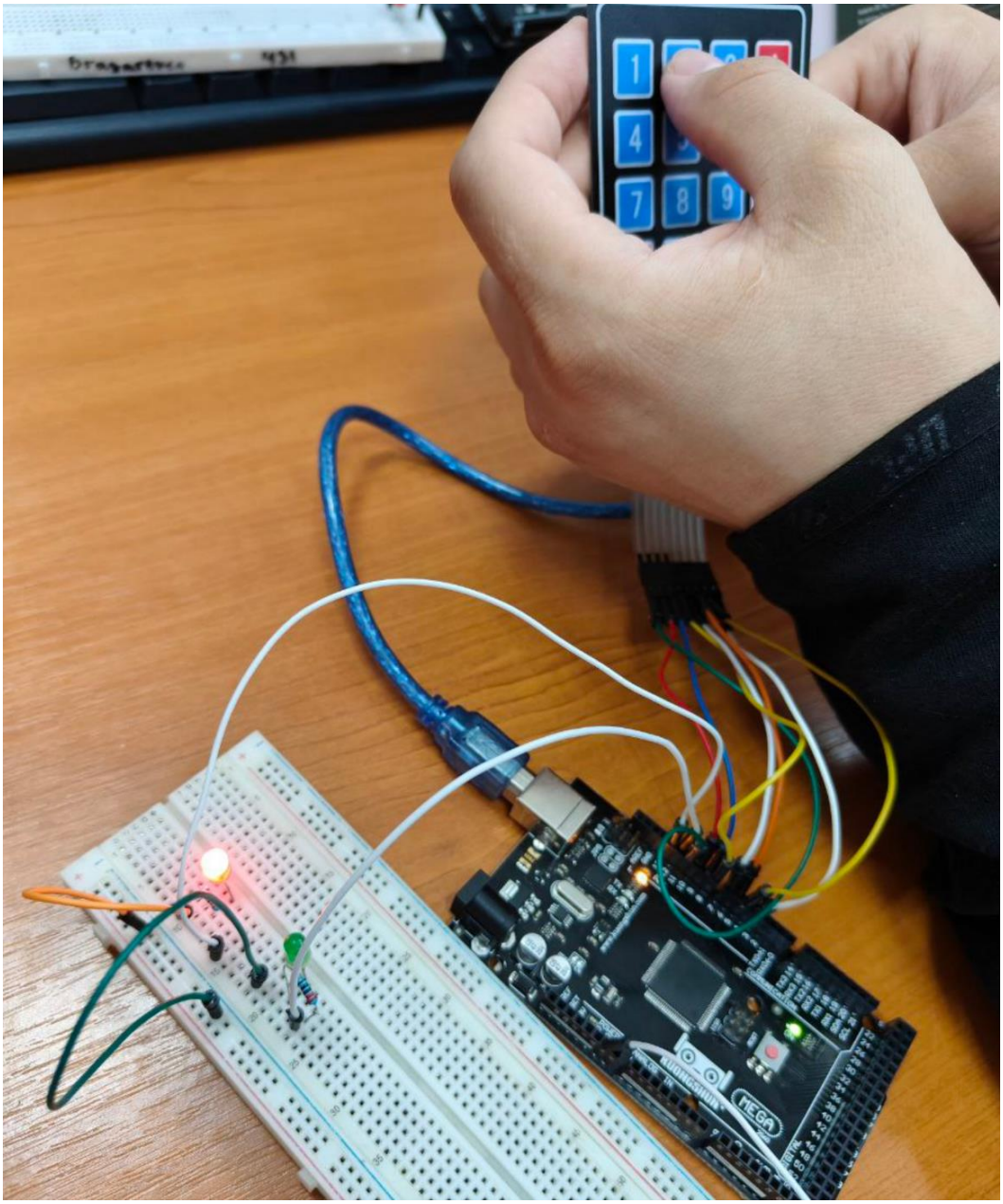



Fig1.3 – Sarcina 3

```
// C++ code
//
#include <Adafruit_LiquidCrystal.h>
#include <Keypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>


Adafruit_LiquidCrystal lcd_1(0);
```

```
int ledGreen = 12;
int ledRed = 13;
String pad;
const byte numRows = 4;
const byte numCols = 4;
String password = "1452";
char keypressed;
char keymap[numRows][numCols] =
{
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};
byte rowPins[numRows] = {10, 9, 8, 7};
byte colPins[numCols] = {6, 5, 4, 3};
Keypad myKeypad = Keypad(makeKeymap(keymap), rowPins, colPins, numRows, numCols);
void setup()
{
pinMode(ledGreen, OUTPUT);
pinMode(ledRed, OUTPUT);
lcd_1.begin(16, 2);
lcd_1.print("Introdu Parola");
delay(1500);
lcd_1.clear();
}

void loop()
{
readKeypad();
if (keypressed == '#') {
if (pad == password ) {
```

```
        digitalWrite(ledGreen, HIGH);
     lcd_1.setCursor(0, 1);
            lcd_1.print("Parola corecta");

    } else {
     lcd_1.setCursor(0, 1);
            lcd_1.print("Parola gresita");
      digitalWrite(ledRed, HIGH);
    }
  } if (keypressed == '*') {
     pad = "";
    lcd_1.clear();
  }
  lcd_1.setCursor(0, 0);
  lcd_1.print(pad);
  delay(10);
  digitalWrite(ledGreen, LOW);
  digitalWrite(ledRed, LOW);
     }

     void readKeypad() {
     keypressed = myKeypad.getKey();
     if (keypressed != '#') {
     String pass = String(keypressed);
     pad += pass;
     }
     }
```

1.2.1

```
int ledPin = 12;

int buttonPin = 4;
```

```
int ledState = LOW;
int buttonState;
int lastButtonState = LOW;
bool buttonPressed = false;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {

  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH && lastButtonState == LOW && !buttonPressed) {
    ledState = !ledState;
    digitalWrite(ledPin, ledState);
    buttonPressed = true;
  }

  if (buttonState == LOW) {
    buttonPressed = false;
  }
  lastButtonState = buttonState;
}
```