Sistem de preluare text folosind tastatura matricială

1. Obiective

2. Descrierea soluției

2.1. Schema

2.2. Descriere componente

2.3. Descriere algoritmi

3. Rezultate

4. Concluzii

**1.Obiective**

Sistemul va fi capabil să preia orice caracter alfanumeric de la tastatura matricială, folosind principiul apăsărilor multiple existent la telefoanele mobile.

Vor exista următoarele taste cu funcționalități bine definite, apelabile prin apăsare succesivă:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | E |
| 4 | 5 | 6 | < |
| 7 | 8 | 9 | X |
| - | 0 | + | = |

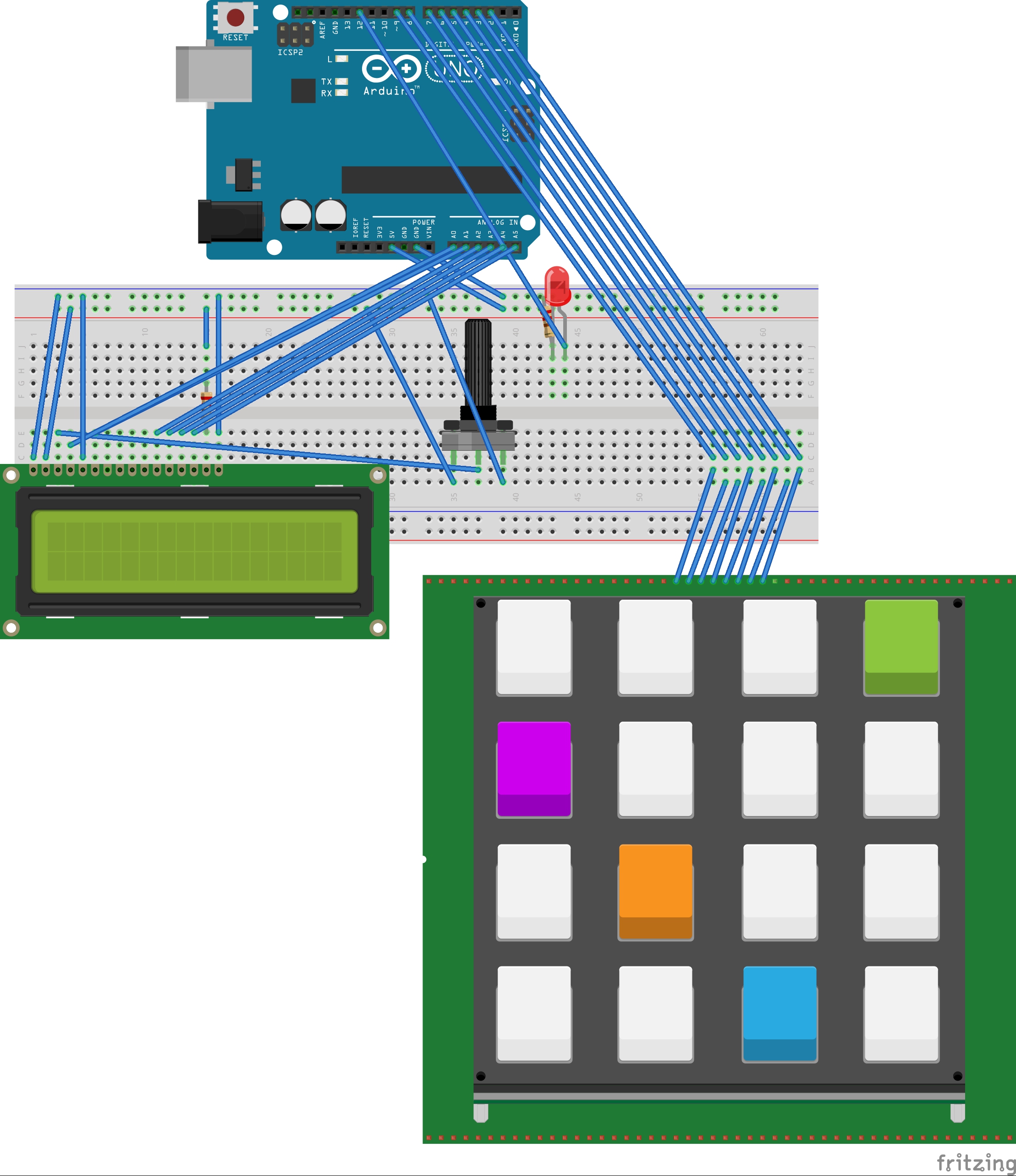
* Tasta 1: “.”, “,”, “?”, “!”, “1”.
* Tasta 2: “a”, “b”, “c”, “2”.
* Tasta 3: “d”, “e”, “f”, “3”.
* Tasta 4: “g”, “h”. “i”, “4”.
* Tasta 5: “j”, “k”, “l”, “5”.
* Tasta 6: “m”, “n”, “o”, “6”.
* Tasta 7: “p”, “q”, “r”, “s”, “7”.
* Tasta 8: “t”, “u”, “v”, “8”.
* Tasta 9: “w”, “x”, “y”, “z”, “9”.
* Tasta 0: “ ”, “0”, “#”; -- opțiunea # evaluează expresia matematică introdusă anterior la adevărat/fals, scriind un mesaj corespunzător -“corect” respectiv “wrong”- pe LCD, și aprinzând un LED de 2 ori în cazul unui răspuns corect, respectiv de 6 ori intermitent si apoi lung timp de 2 secunde, dacă răspunsul este greșit.
* Tasta E: enter – introducerea următorului caracter.
* Tasta <: backspace – ștergerea caracterului curent, și înapoierea cu un caracter.
* Tasta X: clear screen – ștergerea întregului ecran.
* Tasta –: “–”,“/” - operațiile matematice specifice.
* Tasta +: “+”,“\*” - operațiile matematice specifice.
* Tasta =: calculează operația precedentă, afișând rezultatul.

Rezultatul va fi afișat pe lcd în timpul tastării.   
Proiectul finalizat va cuprinde funcționalități multiple:

* introducerea, validarea, ștergerea, editarea unui text pe lcd;
* calcularea unor operații matematice de bază;
* evaluarea unei expresii matematice anterior introduse la T(true)/F(false).

**2.Descrierea soluției**

**2.1.Schema**



**\***în locul tastaturii din imagine se va folosi una matricială, cu 8 intrări.

**2.2.Descriere Componente**

Pentru realizarea proiectului se vor folosi următoarele componente:

- o plăcuță Arduino Uno

- un display LCD 16\*2

- tastatură matricială 4x4

- un potențiometru de 10K

- un bread board

- o rezistență de 220 ohmi

- o rezistență de 10K

- un LED

- fire pentru realizarea cuplajelor

Cuplaje:

*LCD – Arduino:*

- VSS - GND

- VDD - VCC

- V0 – pinul din mijloc al potentiometrului

- RS – A0

- RW - GND

- E – A1

- D4 – A2

- D5 – A3

- D6 – A4

- D7 – A5

- A – un pin al rezistentei de 10K ohmi

- K - Gnd

*Tastatura matricială - Arduino:*

- pinul 1 -> digital pin 9

- pinul 2 -> digital pin 8

- pinul 3 -> digital pin 7

- pinul 4 -> digital pin 6

- pinul 5 -> digital pin 5

- pinul 6 -> digital pin 4

- pinul 7 -> digital pin 3

- pinul 8 -> digital pin 2

*Potentiometru:*

- pin stanga - Vcc

- pin mijloc – LCD V0

- pin dreapta - Gnd

*Rezistență 10K:*

- capăt 1: LCD A (pin 15)

- capăt 2: Vcc

*Rezistență 220 ohmi:*

-capăt 1: capăt scurt LED

-capăt 2: Gnd

*LED:*

-capăt scurt: legat cu rezistența

-capăt lung: digital pin 12 Arduino

**2.3. Descriere Algoritmi**

În programul creat s-au folosit librăriile:

#include <LiquidCrystal.h>

#include <Key.h>

#include <Keypad.h>

S-a definit o instanță a clasei Keypad pentru a reprezenta intrările de la tastatura matricială:

const byte ROWS = 4;

const byte COLS = 4;

char keys[ROWS][COLS] =

{

{'1', '2', '3', 'E'},

{'4', '5', '6', '<'},

{'7', '8', '9', 'X'},

{'-', '0', '+', '='}

};

byte rowPins[ROWS] = { 9, 8, 7, 6 };

byte colPins[COLS] = { 5, 4, 3, 2 };

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

Multiplele valori ale unei taste au fost stocate în vectori de caractere:

char letters\_plus[2] = {'+', '\*'};

char letters\_minus[2] = {'-', '/'};

char letters\_0[3] = {' ', '0', '#'}; // #-verify eq;

char letters\_1[5] = {'.', ',', '?', '!', '1'};

char letters\_2[4] = {'a', 'b', 'c', '2'};

char letters\_3[4] = {'d', 'e', 'f', '3'};

char letters\_4[4] = {'g', 'h', 'i', '4'};

char letters\_5[4] = {'j', 'k', 'l', '5'};

char letters\_6[4] = {'m', 'n', 'o', '6'};

char letters\_7[5] = {'p', 'q', 'r', 's', '7'};

char letters\_8[4] = {'t', 'u', 'v', '8'};

char letters\_9[5] = {'w', 'x', 'y', 'z', '9'};

Mesajul de afișat pe ecranul LCD a fost stocat într-un șir de caractere, de lungime 32, apoi parsat în 2 linii a câte 16 caractere, corespunzător afișajului pe LCD:

char messageToPrint[32]= "";

char messageLine1[16];

char messageLine2[16];

În inițializarea programului, funcția setup:

- se declară prinul 12 ca pin de OUTPUT;

- se inițializează messageToPrint cu „caractere goale”(spații).

Funcția principală, loop:

- preia caracterul de la tastatură;

- verifică dacă acesta este un caracter valid, și dacă s-a efectuat o schimbare de stare a butonului (dacă s-a apăsat);

- află lungimea șirului de valori a caracterului tastat: majoritatea tastelor au 4 valori posibile (de unde initializarea la 4), tastele 7 și 9 au 5 valori, apoi urmând tastele cu funcționalități speciale, care pot avea 1, 2 sau 3 valori;

- se calculează a câta apăsare a butonului curent este, acesta urmând a fi un indice al poziției din șirul de valori posibile ale butonului;

- după aceasta se va putea alege, în funcția chooseFunctionality acțiunea de executat în funcție de caracterul găsit corespunzător apăsării butonului

- la final se va procesa mesajul – șirul de caractere, pentru a putea fi afișat pe ecranul LCD și apoi se va afișa.

void loop()

{

char key = keypad.getKey();

arrayLength = 4;

// Check for a valid key

if(key!=NO\_KEY && keypad.keyStateChanged())

{

// key's no. of letters/chars

arrayLength = findNoLetters(key);

// no. times pressed button

timesPressed(key);

chooseFunctionality(key);

parseMessageToDisplay();

lcd.setCursor(0,0);

lcd.print(messageLine1);

lcd.setCursor(0,1);

lcd.print(messageLine2);

}

}

Funcția *chooseFunctionality* :

- în cazul tastei X: va șterge tot ce a fost scris pe ecran, resetând poziția curentă

- tasta E: va trece la introducerea următorului caracter

- tasta < (backspace): va șterge caracterul curent, repoziționându-se pe caracterul anterior

- tasta =: va calcula expresia matematică introdusă (de forma: “a+b”, unde a, b sunt cifre)

- valoarea #, poate fi accesată prin apăsări succesive ale tastei 0, și reprezintă evaluarea proprietății de adevăr a expresiei introduse anterior, ce va fi de forma: “a\*b c”, programul verificând dacă rezultatul introdus de utilizator (în acest caz “c”) este corect.

-- în cazul unei evaluări corecte, led-ul va pâlpâi de 2 ori, apoi se va afișa mesajul “corect”

-- în cazul unei evaluări incorecte, led-ul va pâlpâi intermitent de 6 ori și apoi lung timp de 2 secunde, afișându-se la final mesajul “wrong!”

void chooseFunctionality(char key)

{

switch(key){

case 'X':{

clearMessage();

letterPos=0;

};break;

case 'E':

positionLetter(); break;

case '<':{

backspace();

};break;

case '=':{

int res = equalsTo();

printResToMessage(res);

};break;

default: {

charToPrint = chooseChar(key);

if(charToPrint=='#'){

messageToPrint[letterPos] = 'e'; // evaluate

verifyEquation();

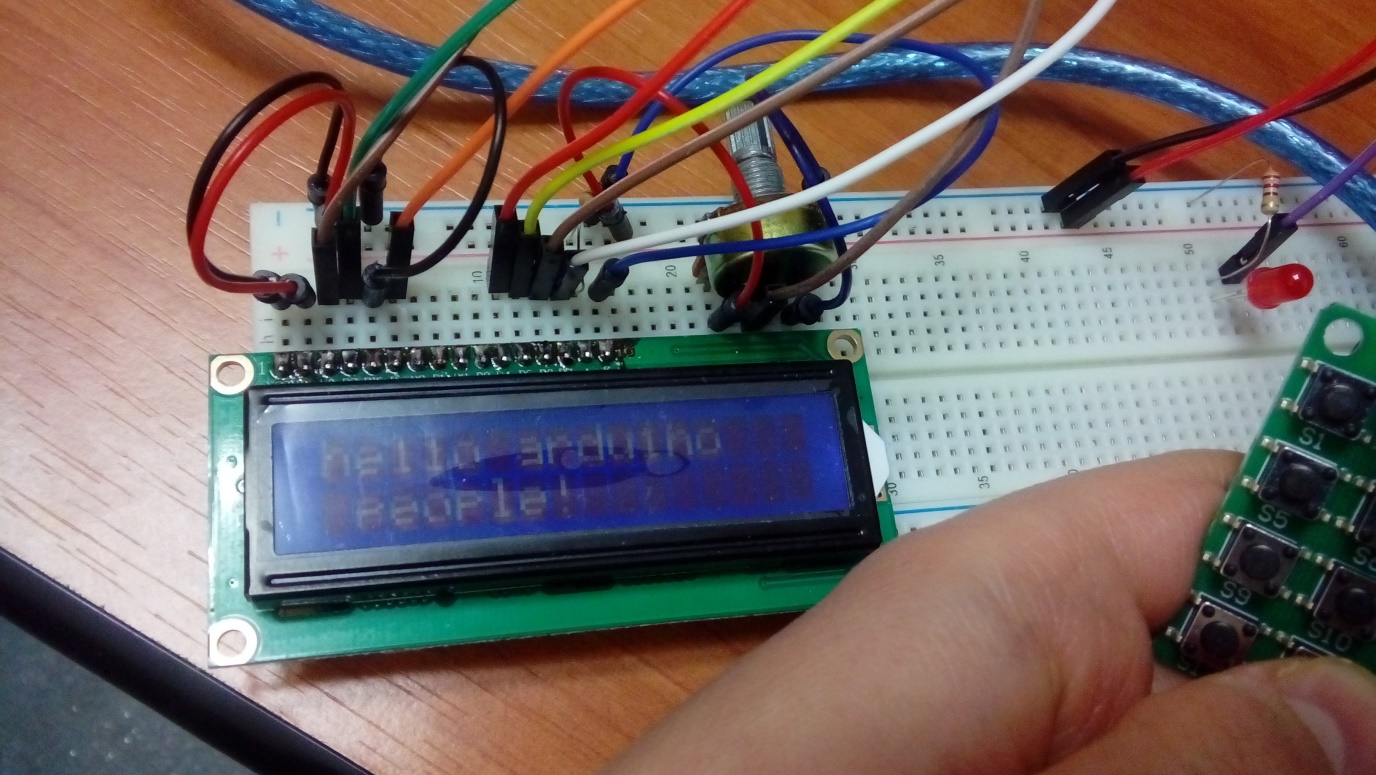
}else

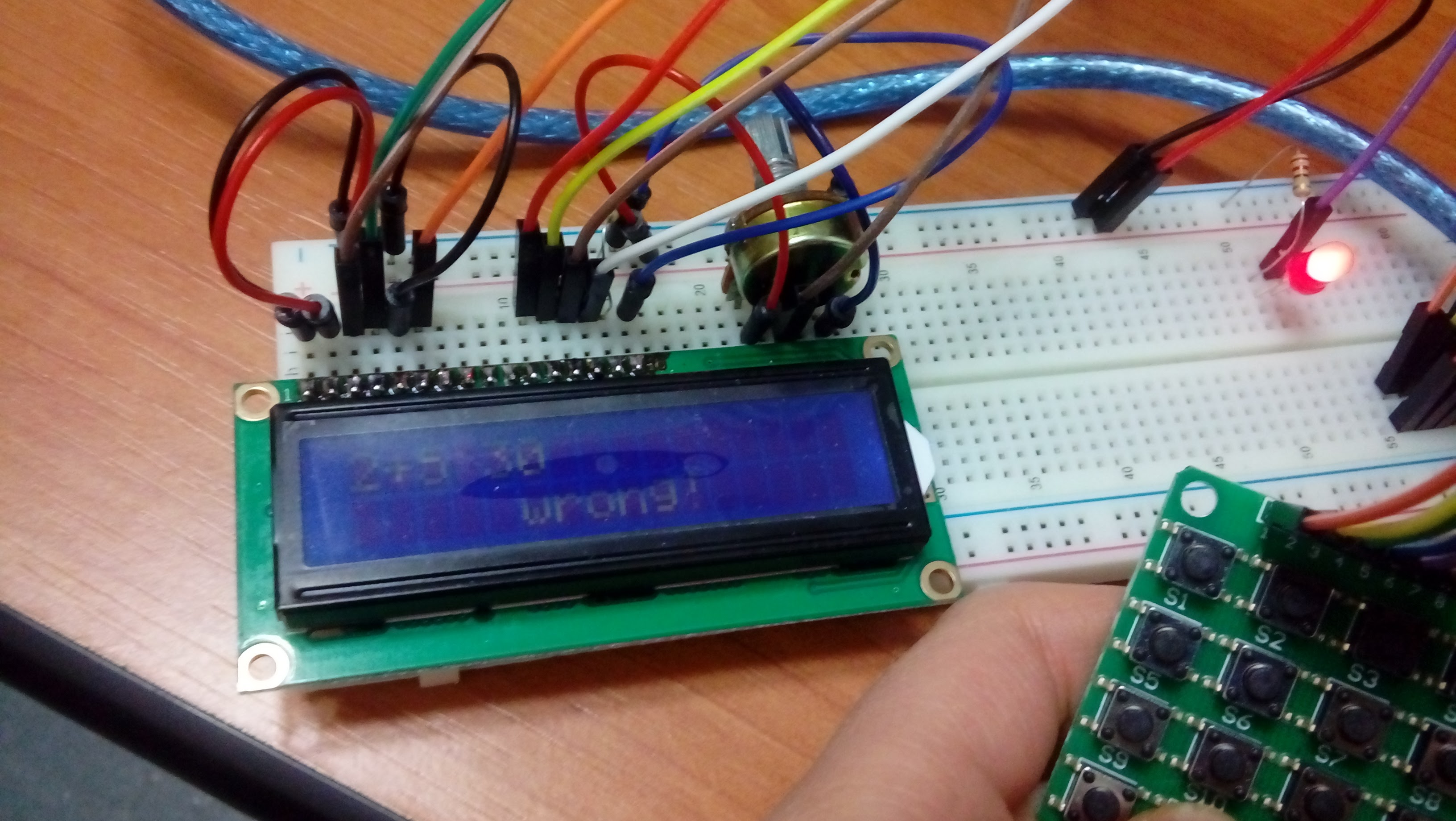
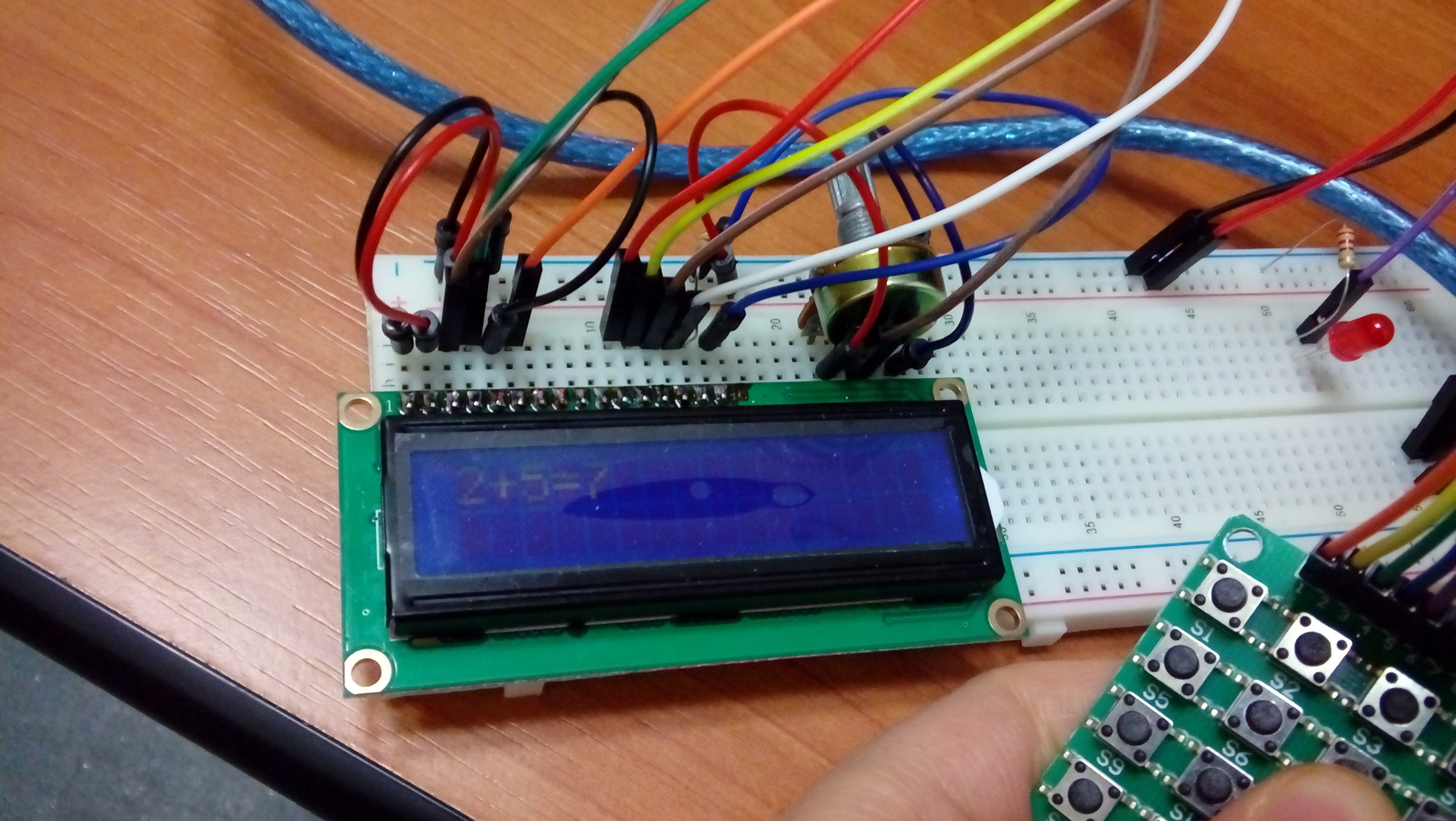
messageToPrint[letterPos] = charToPrint;

};break;

}

}

**3.Rezultate**



**4.Concluzii**

Sistemul va putea fi utilizat cu ușurință de către orice tip de utilizator, premițând o comunicare ușoară cu programul. Se vor putea trimite mesaje specifice către afișor, permițându-se, de asemenea, efectuarea unor calcule rapide, precum și evaluarea stării de adevăr a unor răspunsuri introduse de utilizator, într-un mod interactiv.