LABORATOR 7 - Tastatură

SCOPUL LUCRĂRII

Se va conecta o tastatura reală cu organizarea 4x4 la Atmega16. Tastatura este prezentată în figura 1 iar organizarea sa internă în figura 2.

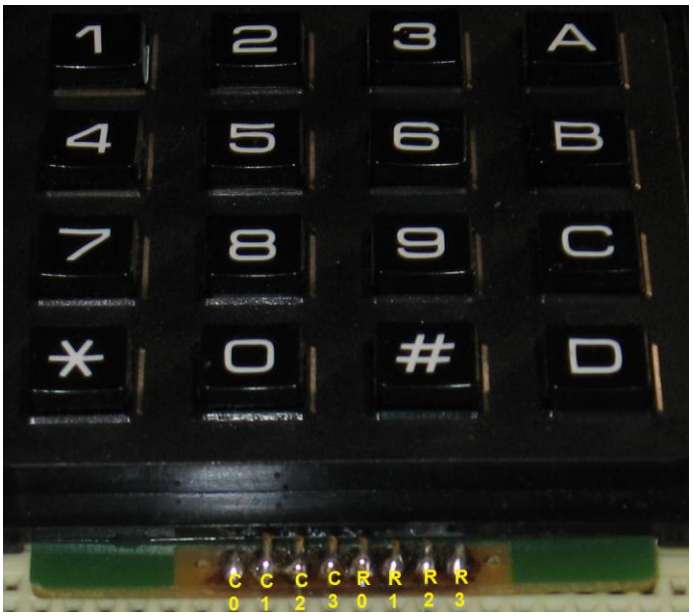


figura 1

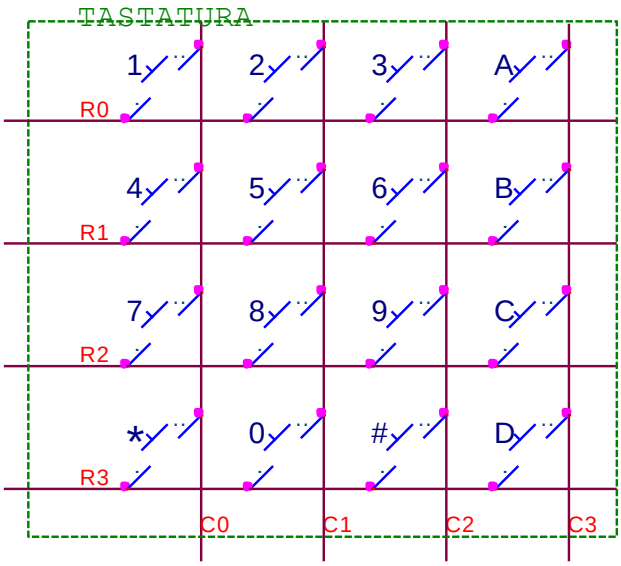
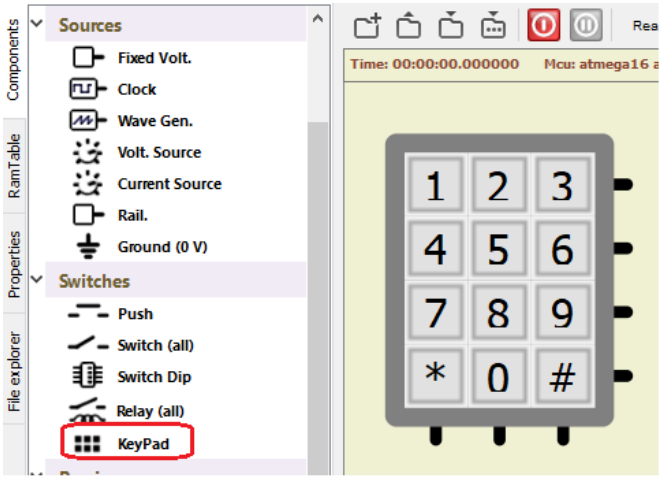


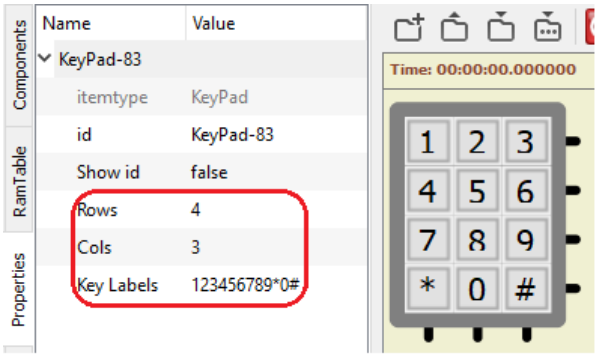
figura 2

Tastatura este tratată în prelegerea 4. Pentru acest laborator este obligatoriu să parcurgeți această prelegere.

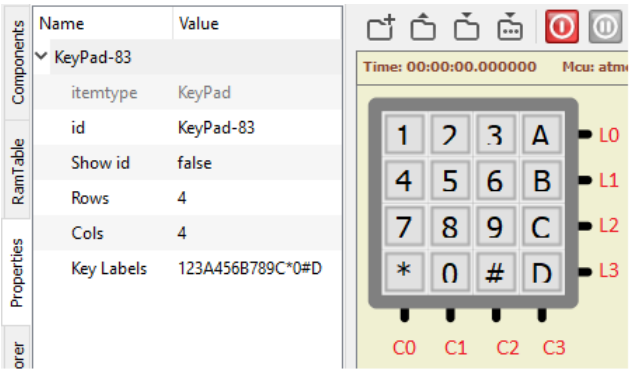
Pasul 1: Conectarea tastaturii



În simulare se va folosi componenta KeyPad din grupul Switches. Vezi figura alăturată.



a)



b)

figura 3

Inițial componenta KeyPad este configurată ca în figura 3. In continuare componenta KeyPad se va numi tastatură. În configurația inițială tastatura are 4 linii și 3 coloane iar pe taste sunt scrise caracterele din câmpul „Key Labels”. Vom reconfigură tastatura astfel încât să semene cu tastatura reală din figura 1. Numărul de coloane se va seta la 4 iar câmpul „Key Labels” se vas seta la valoarea din figura 3b. Tot în figura 3b se precizează semnificația terminalelor tastaturii: liniile L0..L3 și coloanele C0..C3

Conectarea tastaturii 4x4 din figura 1 la portul A de la ATmega16 se face conform schemei din figura 4:

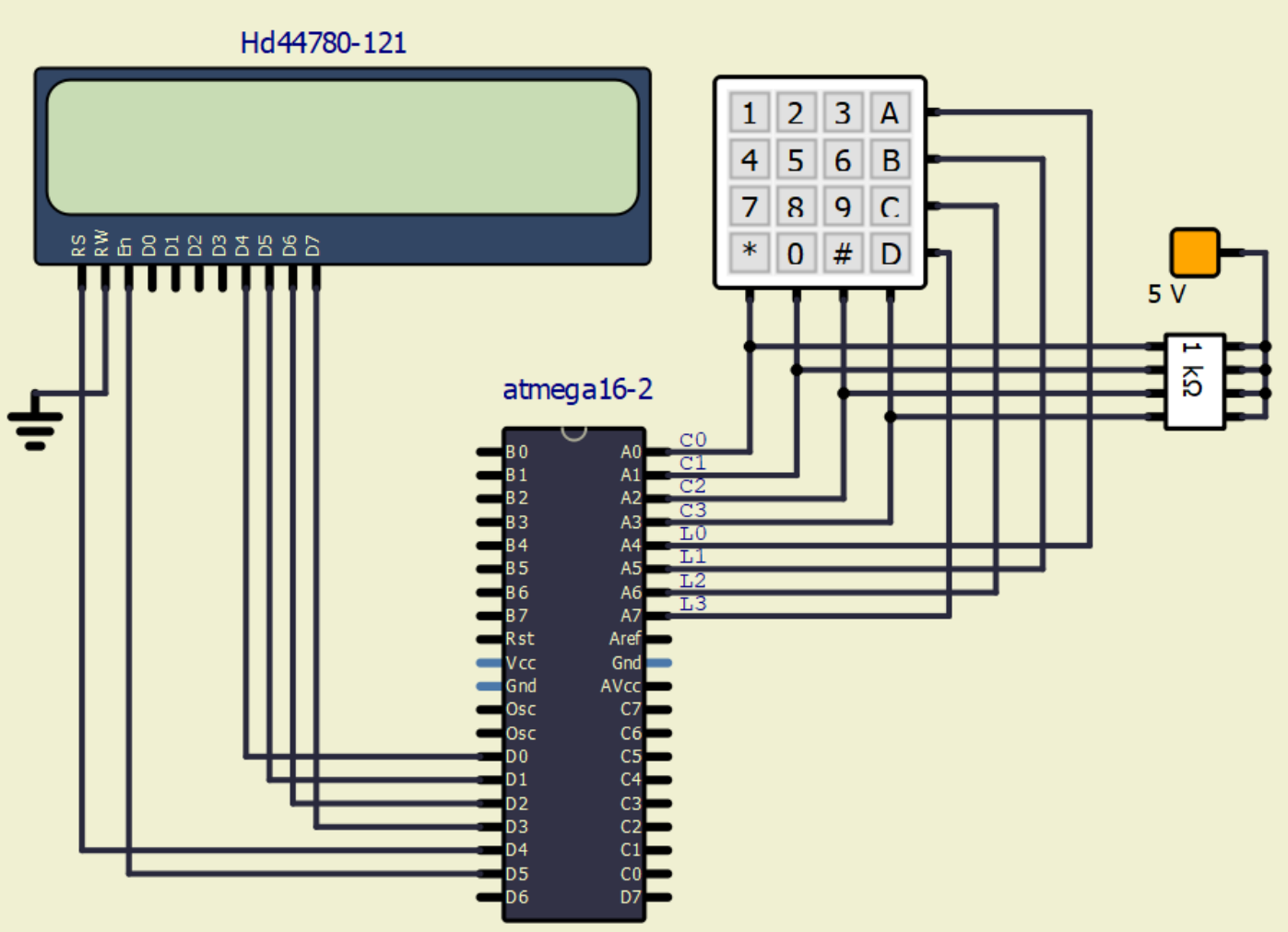


figura 4

Conectarea tastaturii la microcontroler ca în schema din figura 25, prelegerea 4. În schema din curs sunt 4 rezistențe prin care cele 4 coloane se conectează la Vcc. Aceste 4 rezistente stabilesc potențialul coloanelor atunci când nu este apăsată nici o tastă.

În schema din figura 4 cele patru rezistențe se înlocuiesc cu o rețea rezistivă. O astfel de rețea a fost folosită și în laboratorul 3. Numărul de rezistențe din rețea se setează prin intermediul proprietății Size. Setați Size la valoarea 4.

Pasul 2: kbscan()

- **Creaţi** un nou proiect numit **kbd**.
- **Copiați** în folderul proiectului **kbd** fișierele LCD.h și LCD.c din proiectul LCD.
- **Adăugați** la proiectul kbd fișierele LCD.h și LCD.c copiate anterior.
- **Redenumiți** fișierul main.c. Noul nume va fi kbd.c. Inițial conținutul fișierul kbd.c va fi:

```
#include <avr/io.h>
#include "LCD.h"

int main(){
   initLCD();

   while(1){

   }
}
```

- **Adăugați linia**

  ```
  #define NOKEY 0x7f
  ```

  în kbd.c

Mai întâi se va implementa funcția kbscan(), pornind de la specificațiile din prelegerea 4. Funcția va fi rezidentă în fișierul kbd.c.

Portul A este folosit după cum urmează (este explicat în curs și reluat în continuare):

- Patru pini din portul A, și anume PA7(7:4), se vor folosi pentru a trimite codul de scanare al liniilor: ZZZ0, ZZ0Z, Z0ZZ, 0ZZZ, unde Z înseamnă că 3-state. Pentru a obține aceste 4 combinaţii biţii 7:4 din PORTA se înscriu cu „0000”. Deoarece pinii PA(3:0) se folosesc pentru citire coloanelor valoarea scrisa în PORTA(3:0) nu contează. Pentru că totuși trebuie să scriem ceva în PORTA(3:0), vom scrie ‚0’. Astfel PORTA va fi 0b00000000 tot timpul. Această valoare nu se mai modifică.

  Pentru ca numai un singur bit din biții 7:4 fie zero iar ceilalți Z, trebuie ca un singur repetor 3–state din port să fie activ. Repetoarele 3-state sunt controlate de registrul I/O DDRA. Cele 4 perechi de combinaţii folosite sunt centralizate în tabelul următor:

PORTA(7:0)	00000000			
DDRA(7:0)	00010000	00100000	01000000	10000000
PA(7:0)	ZZZ0----	ZZ0Z----	Z0ZZ----	0ZZZ----

- Deoarece pinii PA(3:0) se folosesc pentru citire coloanelor aceștia trebuie să fie pini de intrare. De aceea in tabelul de mai sus DDRA(3:0) este „0000” și nu se modifică.
- Valorile din PORTA și DDRA pentru configurarea liniilor sunt scrise cu albastru în tabelul de mai sus.
- În secțiunea de inițializare se face `PORTA=0`. Valoarea scrisă în PORTA nu se modifică ulterior.

În continuare va detalia algoritmul de scanare al tastaturi prezentat în curs și se vor da câteva indicații de implementare. Scrieți funcția kbscan() conform indicațiilor de mai jos. Pentru aceasta adăugați și completați liniile scrise cu verde:

1. Prima valoarea ce se va trimite în DDRA este 0001_0000. Urmează apoi 0010_0000, 0100_0000 şi 1000_0000. Pentru a implementa în C trimiterea pe linii a celor 4 configurații din tabelul de mai sus, există mai multe posibilități. În continuare se va prezenta una dintre aceste posibilități.

   Cele 4 configurații ce se înscriu în DDRA se generează prin deplasare la stânga. Se pornește cu configurația 0001_0000. Prin deplasare la stânga a configurației inițiale se obțin configurațiile:

   ```
   0b0001_0000  = 0x10    inițial
   0b0010_0000  = 0x20    după prima deplasare la stânga
   0b0100_0000  = 0x40    după a II-a  deplasare la stânga
   0b1000_0000  = 0x80    după a III-a deplasare la stânga
   0b0000_0000  = 0x00    după a IV-a deplasare la stânga
   ```

Aceste 4 configurații se pot genera cu o instrucțiune for:

```
char kbscan(){
   unsigned char temp;
   unsigned char cols;

   unsigned char lina=0;  //linia activa
   unsigned char cola=0; //coloana activa

   unsigned char cod_intern;
   char          cod_extern;
   char          tabela_trans[] = "123A456B789C*0#D";

   for( DDRA = 1<<4; ... ) {
```

2. După s-a stabilizat potențialul liniilor trebuie citită starea coloanelor. Înainte de această operație trebuie să se aștepte apariția ‚0' –ului pe unul din pinii PA de ieșire, propagarea acestuia pe linie, prin pus buton și prin sincronizatorul din logica unuia din pinii PA de intrare. Sincronizatorul introduce o întârziere de două cicluri de ceas.

   Pentru ca toate evenimentele descrise anterior să aibă loc, trebuie așteptat cu:

   ```
   _delay_us(4);
   ```

3. Se citește starea coloanelor. Este preferabil să inversăm valoarea citită, din motive care vor deveni evidente mai târziu. Citirea stării coloanelor se face cu:

   ```
   cols=~PINA;
   ```

   Efectul acestei instrucțiunii trebuie detaliat pentru a ști ce anume se citește în variabila `cols`. În acest sens trebuie ţinut seama de trei informații: valoarea scrisă în DDRA, valoarea scrisă în PORTA și potențialul pinilor PA:

   - Pentru exemplificare vom consideră că DDRA are prima valoare generată de ciclul for început la P1, și anume 0b0001_0000,
   - PORTA are valoarea constantă 0b0000_0000
   - Pinii PA3:0 sunt conectați la coloanele 3:0.

   Prin combinarea acestor trei informații obținem:

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DDRA:	0	0	0	1	0	0	0	0
PORTA:	0	0	0	0	0	0	0	0
PA:	z	z	z	0	c3	c2	c1	c0
PINA:	x	x	x	0	c3	c2	c1	c0

   Stării pinilor PA3:0 este ușor de determinat deoarece aceștia sunt conectați la coloanele c3:0.

   Starea lui PA4 este simplu de determinat deoarece DDRA4 este ‚1', ceea ce face ca PA4=PORTA4. Astfel PA4 este ‚0'.

   Starea pinilor PA7:5 fluctuează deoarece DDRA-ul corespunzător este ‚0' și nu există rezistență sau driver extern. Citirea unui potențial fluctuant duce la rezultate imprevizibile. Din acest motiv nu putem ști starea lui PA7:5.

   Deoarece starea coloanelor se citește pe biții 3:0 din PINA, iar 3 din biții 7-4 au valori aleatoare, biții 7:4 trebuie mascați la ‚0':

   ```
   cols=...;
   ```

4. Testează dacă există cel puțin un ‚1' în variabila `cols`, adică dacă `cols` este diferită de zero. Dacă DA, există cel puțin o tastă apăsată pe linia pe care s-a trimis ‚0' la pasul P3 și vom trece la P7.

Pentru aceasta întrerupeți ciclu `for(DDRA =1<<4…….` început la pasul P1.

```
if(……
```

5. Dacă NU, se termina iterația curenta a ciclului *for* de la pasul P1.

```
}//end for
```

6. Am terminat ciclului `for(DDRA =1<<4;…` Ajungem aici dacă ciclul s-a terminat normal sau dacă la pasul P4 s-a găsit o coloană activă și ciclul a fost întrerupt. Pentru a determina cum s-a terminat for-ul se testează fie `cols`, fie `DDRA`.

```
if(cols... //sau if(DDRA...
```

De exemplu, dacă `cols` conține cel puțin un ‚1' înseamnă că s-a ieșit prin P4. În acest caz vom continua cu pasul P7.

În caz contrar nu este nici o tastă apăsată și vom termina funcția cu

```
return NOKEY;
```

7. Execută conversia de la codurile de scanare la codul tastei apăsate. Pentru a afla codul liniei active trebuie executate patru calcule. Se va calcula:
   - numărul liniei active,
   - numărul coloanei active,
   - codul intern
   - codul extern.

Mai întâi s**e determină linia activă** conform tabelului următor:

PORTA(7:0)	00000000			
DDRA(7:0)	**0001**0000	**0010**0000	**0100**0000	**1000**0000
Linia activă - lina	0	1	2	3

Trebuie aflată poziția zeroului în configurația trimisă pe linii. Aceasta funcționalitate se poate implementa fie cu `if`-uri, fie cu `switch(DDRA)`, fie prin deplasare dreapta și contorizare. Vom exemplifica varianta bazata pe deplasare deoarece se poate aplica și atunci când numărul de coloane este mare.

Poziția ‚1'-lui din DDRA poate fi găsită prin deplasare la dreapta. Pentru a nu altera valoarea din DDRA (este necesara în caz de debug) vom copia vom face o copie în variabila *temp*:

```
temp=DDRA;
```

Oricare ar fi valoarea lui `temp`, după un număr de deplasări la dreapta se va ajunge la valoarea `0001_0000`. De exemplu, dacă `temp=0001_0000` nu mai trebuie făcută nici o deplasare iar `lia` – linia activă – va fi 0. Dacă `temp =0010_0000b` după o deplasare dreapta se ajunge la valoarea `0001_0000`. Numărul de deplasări dreapta după care se ajunge la combinația `0001_0000` identifică linia activă.

```
... calculează lina
```

Dacă varianta cu deplasări vi se pare complicată, puteți implementa cu 4 if-uri sau cu switch.

8. **Se determină coloana activă**. Este posibil ca utilizatorul să fi apăsat mai multe taste simultan, astfel încât ne putem aștepta să existe mai multe zerouri în configurația citită. Zerourile citite de pe coloane devin unu-ri în variabila `cols`. În acest caz este necesară o regulă care să spună cum se tratează acest caz. De exemplu, se poate defini o prioritatea a coloanelor: coloana zero are prioritatea cea mai mare iar coloana trei prioritatea cea mai mică. Conform acestei reguli poziția zeroului în configurația citită

de pe coloane se determină conform tabelului următor:

Variabila cols				
bit_3	**bit_2**	**bit_1**	**bit_0**	**Coloana activă - cola**
x	x	x	1	0
x	x	1	0	1
x	1	0	0	2
1	0	0	0	3

Coloana activă se poate determina prin deplasarea la dreapta a lui `cols`, la fel ca în cazul liniei. Deplasați la dreapta variabila `cols` și numărați după câte deplasări se ajunge la valoarea care are un unu pe poziția 0. Numărul de deplasări făcut este numărul coloanei active – `cola`.

```
... calculează cola
```

Dacă varianta cu deplasări vi se pare complicată, puteți implementa cu 4 if-else-uri.

9. În general, pentru orice tastatură se poate determina cu ușurință **codul intern** al tastei. Acest cod se calculează cu formula:

*Cod_intern = Linia_activă * Lungimea_liniei + Coloana_activă*

De exemplu, dacă configurația trimisă pe linii este ZZ0Z (DDRA(7:4)=0010) iar configurația citită pe coloane este 0111 (cols=1000), atunci este apăsată tasta ‚B'. Linia activă este 1 iar coloana activă este 3. Atunci codul intern al tastei este:

Cod_intern = 1*4+3=7

Calculează codul intern:

```
... calculează cod_intern
```

Codul generat în exterior este de regulă codul caracterului ASCII înscris pe tastă și se obține prin prelucrarea codului intern:

Cod_extern=f(Cod_intern)

În curs situația este particulară: *Cod_extern=Cod_intern*, fapt care în general nu este valabil. Pentru tastatura cu care se lucrează în cadrul acestui laborator, tabelul codurilor interne și externe este prezentat în continuare:

Codul extern	Codul intern	Tasta conectează:		Pe linii se trimite:	Stare coloane:
		Linia	Coloana	L3 L2 L1 L0	C3 C2 C1 C0
‚1'	0000	$0_{10}=00_2$	$0_{10}=00_2$	Z Z Z 0	x x x 0
‚2'	0001	$0_{10}=00_2$	$1_{10}=01_2$	Z Z Z 0	x x 0 x
‚3'	0010	$0_{10}=00_2$	$2_{10}=10_2$	Z Z Z 0	x 0 x x
‚A'	0011	$0_{10}=00_2$	$3_{10}=11_2$	Z Z Z 0	0 x x x
‚4'	0100	$1_{10}=01_2$	$0_{10}=00_2$	Z Z 0 Z	x x x 0
‚5'	0101	$1_{10}=01_2$	$1_{10}=01_2$	Z Z 0 Z	x x 0 x
‚6'	0110	$1_{10}=01_2$	$2_{10}=10_2$	Z Z 0 Z	x 0 x x
‚B'	0111	$1_{10}=01_2$	$3_{10}=11_2$	Z Z 0 Z	0 x x x
‚7'	1000	$2_{10}=10_2$	$0_{10}=00_2$	Z 0 Z Z	x x x 0
‚8'	1001	$2_{10}=10_2$	$1_{10}=01_2$	Z 0 Z Z	x x 0 x
‚9'	1010	$2_{10}=10_2$	$2_{10}=10_2$	Z 0 Z Z	x 0 x x
‚C'	1011	$2_{10}=10_2$	$3_{10}=11_2$	Z 0 Z Z	0 x x x
‚*'	1100	$3_{10}=11_2$	$0_{10}=00_2$	0 Z Z Z	x x x 0
‚0'	1101	$3_{10}=11_2$	$1_{10}=01_2$	0 Z Z Z	x x 0 x
‚#'	1110	$3_{10}=11_2$	$2_{10}=10_2$	0 Z Z Z	x 0 x x

‚D'	1111	$3_{10}=11_2$	$3_{10}=11_2$	0 Z Z Z	0 x x x

Deoarece există mari diferențe între codul intern și cel extern, codul extern se generează cu metoda LUT. În tabelă codul intern se folosește ca index:

```
   cod_extern= tabela_trans[cod_intern];
   return cod_extern;
}//end kbscan
```

```
...
   while(1){
      temp=kbscan();
      if(NOKEY != temp){
         clrLCD();
         putchLCD(temp);
      }
   }
...
```

- Rulați.
- Dacă rutina *kbscan* funcționează corect, atâta timp cât o tasta este apăsată afișajul va pâlpâi. Când tasta este eliberată, va apare codul tastei. Dacă funcționează, apelați profesorul pentru înregistrarea progresului.

Pasul 3: kbhit şi kbcod

Pentru a genera codul tastei o singură dată, la apăsare, se consideră două scanări succesive cu *kbscan* și se detectează secvența tastă neapăsată – tastă apăsată. Pentru tastă neapăsată *kbscan* va întoarce 0x7f iar pentru tastă apăsată ceva diferit de 0x7f. La fel ca în cazul unui singur contact, cele două scanări se fac la un interval de tip mai mare decât durata instabilității. **Prezentarea detaliată a acestei metode plus codul aferent se găsesc în prelegerea 4 la pagina 19**.

Valoarea constantei **DELAY este 200**. Generarea codului o singură dată, la apăsare, face cu un program similar cu cel din curs. Acest program este reprodus în continuare deoarece din curs nu se poate face copy - paste:

```
...
#define DELAY 200

int main(){
   char code_ante = NOKEY;
   char code_now  = NOKEY;

   unsigned char kbhit = 0;
   char kbcode;

   unsigned char loop_cnt=0;

   //initializare LCD, kbd, etc.

   while(1){  //bucla principala
      //…
      if(loop_cnt==DELAY){ //citirile se fac din 15 ms in 15 ms
         loop_cnt=0;

         code_ante = code_now;
         code_now  = kbscan();
        if( code_ante == NOKEY && code_now != NOKEY){
            kbhit=1;
            kbcode=code_now;
```

```
            }
        }
        //……
        //consuma codul
        if(kbhit){
            kbhit=0;
            //prelucreaza kbcode; de exemplu afişeaza-l
        }
        //……
        loop_cnt++;
    } //end while
} //end main
```

Adaptați programul de mai sus!

Dacă implementarea funcționează corect, ar trebui să se afișeze codul tastei, o singură dată, la apăsare. Dacă funcționează, apelați profesorul pentru validare.