

LABORATOR 7 - Tastatură

SCOPUL LUCRĂRII

Se dorește conectarea unei mini tastaturi cu organizarea 4x4 la Atmega16. Tastatura este prezentată în figura 1 iar organizarea sa internă în figura 2.

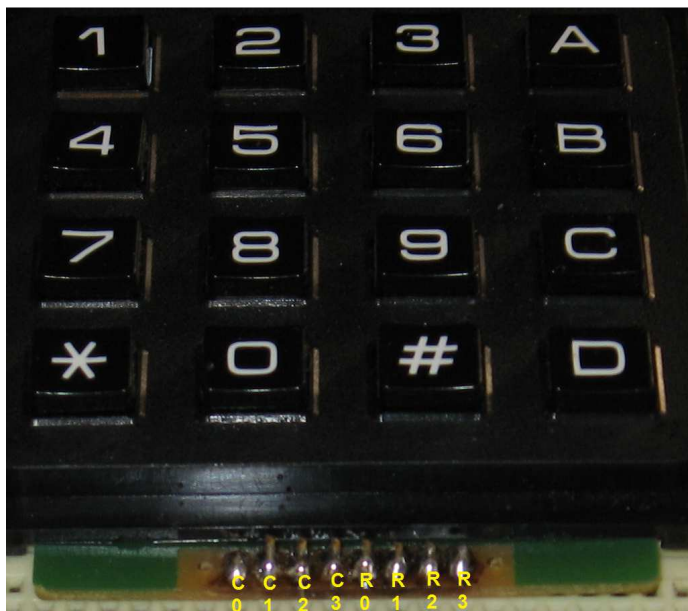


figura 1

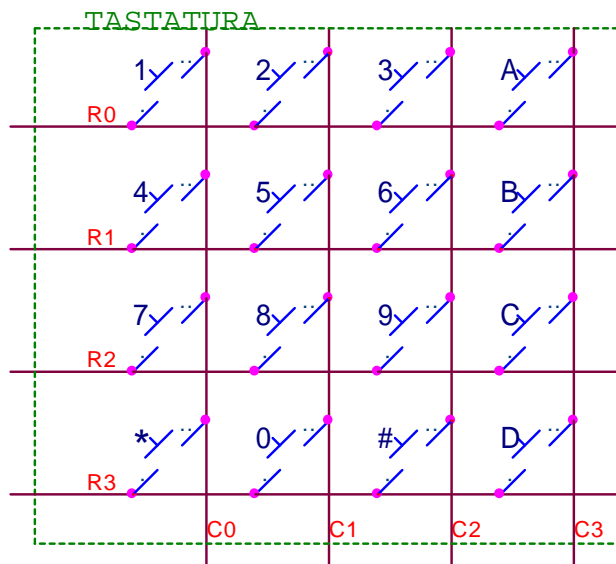


figura 2

Pentru a se conecta cu ușurință cu placa de dezvoltare, conectorul tastaturii este prevăzut cu pini, Conectarea se face ca în figura 3:

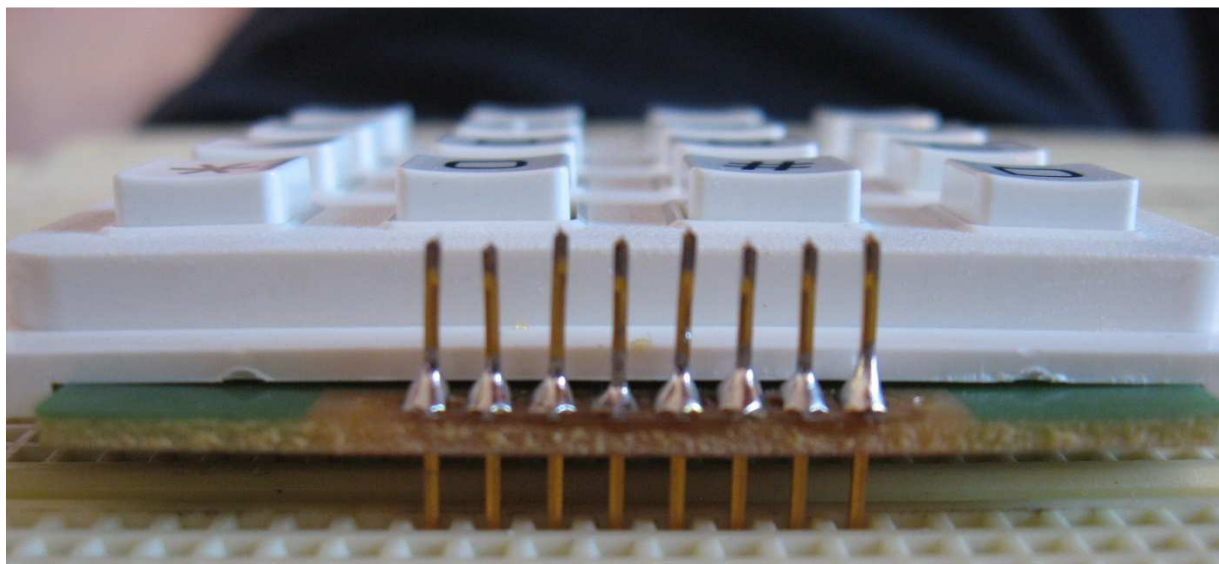


figura 3

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

În schemă apar 4 rezistențe care conectează cele 4 coloane la Vcc. Aceste 4 rezistențe stabilesc potențialul coloanelor atunci când nu este apăsată nici o tastă. Montarea celor 4 rezistențe pe placa de dezvoltare necesită 8 găuri (două per rezistență). Însă dacă ținem seama că toate rezistențele au un terminal conectat la Vcc am putea să construim un ansamblu precum cel din figura 6. În figură ansamblul s-a construit din 8 rezistențe, dar ideea funcționează la fel de bine și pentru 4 rezistențe. Ansamblul celor patru rezistențe va necesita numai 5 găuri și considerabil mai puțin spațiu.

Deoarece structura de n rezistențe conectate ca în figura 6 este foarte întâlnită, ea este realizată industrial și se numește **rețea rezistivă**. În figura 7 este prezentată varianta industrială pentru structura din figura 6. Se observă că poziția terminalului comun este marcată cu un **cerc alb**.

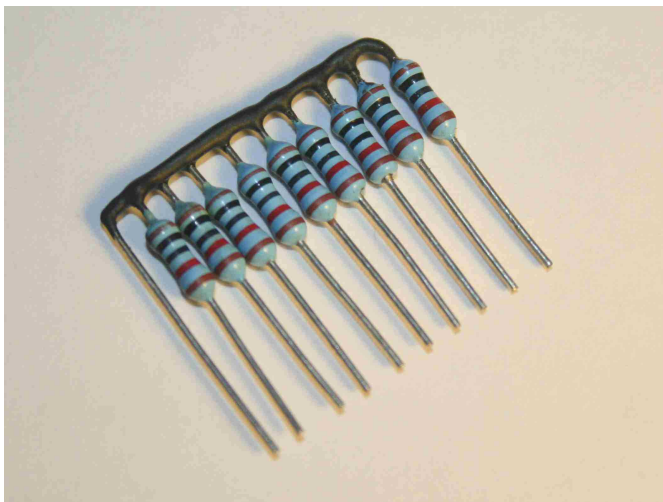


figura 6

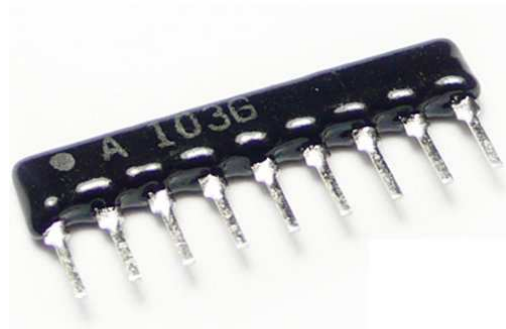


figura 7

Pentru a ocupa mai puțin spațiu cele 4 rezistențe ce țin coloanele la Vcc au fost înlocuite cu o rețea rezistivă. Aceasta are referință RN1 în schema din figura 4. În figura 5 poziția terminalului comun al rețelei este marcat cu **cerc alb**. Acesta este conectat la Vcc.

Conectați tastatura 4x4 conform schemei din figura 4.

Pasul 2: kbscan()

- **Creați** un nou proiect numit **kbd**.
- **Copiați** în folderul proiectului **kbd** fișierul **iofn.c** din proiectul LCD. În loc de proiectul LCD puteți folosi proiectul LCD2, numai dacă pasul **Wrap** a funcționat.
- **Adăugați** la proiectul **kbd** fișierul **iofn.c** copiat anterior.
- **Copiați conținutul** fișierului **LCDtest.c** din proiectul LCD în fișierul **kbd.c**
- **Ștergeți conținutul** buclei principale **while(1)** din **kbd.c**

- **Adăugați linia**

```
#define NOKEY 0x7f
```

în **kbd.c** și **iofn.c**

Mai întâi se va implementa funcția **kbscan()**, pornind de la specificațiile din prelegerea 4. Funcția se va scrie în C și va fi rezidentă în fișierul **IOfn.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 combinații biții 7:4

din PORTA se înscriu cu „0000”. Această valoare nu se mai modifică. Pentru ca numai un singur bit din 4 să 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 combinații folosite sunt centralizate în tabelul următor:

PORTA (7:4)	0000			
DDRA (7:4)	0001	0010	0100	1000
PA (7:4)	ZZZ0	ZZ0Z	Z0ZZ	0ZZZ

- Ceilalți patru pini din portul A, și anume PA(3:0), se vor folosi pentru a citi starea coloanelor.
- În funcția de inițializare a sistemului *sysinit*, 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 valoare ce se va trimite în DDRA7:4 este 0001. Urmează apoi 0010, 0100 și 1000. 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:

```
0001_0000 = 0x10   inițial
0010_0000 = 0x20   după prima deplasare la stânga
0100_0000 = 0x40   după a II-a deplasare la stânga
1000_0000 = 0x80   după a III-a deplasare la stânga
0000_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 lia=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 pushbuton și prin sincronizatorul din logica unuia din pinii PA de intrare. Structura porturilor A, B, C și D este detaliată în figura 12.2 de la pagina 50 din datele de catalog. Sincronizatorul introduce o întârziere de două cicluri de ceas.

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

```
nop; nop; nop; nop; nop; nop;
```

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 instrucțiuni trebuie detaliat pentru a ști ce anume se citește în variabila `cols`. În acest sens trebuie ținut seama de trei informații: valoarea scrisă în DDRA, valoarea scrisă în

PORTA și potențialul pinilor PA:

- Pentru exemplificare vom considera că DDRA are prima valoare generată de ciclul for început la P1, și anume 0001_0000,
- PORTA are valoarea constantă 0000_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 ciclul for(DDRA =1<<4..... început la pasul P1.

```
if(.....
```

5. Dacă NU, se termina iterația curentă a ciclului for-ul de la pasul P1.

```
//end for
```

6. Am terminat ciclului for(DDRA =1<<4;... Ajungem aici dacă ciclul s-a terminat în mod natural 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 se **determină linia activă** conform tabelului următor:

PORTA(7:0)	00000000			
DDRA(7:0)	00010000	00100000	01000000	10000000
PA(7:4)	ZZZ0	ZZ0Z	Z0ZZ	0ZZZ
Linia activă - lia	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 bazată 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 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 lia
```

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

- 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 unuri 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 coloana activă – cola.

```
... calculează cola
```

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

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

$$\text{Cod_intern} = \text{Linia_activă} * \text{Lungimea_liniei} + \text{Coloana_activă}$$

De exemplu, dacă configurația trimisă pe linii este ZZ0Z (DDRA=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:

$$\text{Cod_intern} = 1*4+3=7$$

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: L3 L2 L1 L0	Stare coloane: C3 C2 C1 C0
		Linia	Coloana		
,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
```

După ce ați terminat de scris funcția *kbscan()*, funcționarea acesteia se testează cu următorul program principal:

```
...
```

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

```
...
```

- Verificați alimentarea iar apoi alimentați.
- Rulați.
- Dacă rutina funcționează corect, atâta timp cât o tasta este apăsată afișajul va pâlpâi. Când tasta este eliberată, va apare codul tastei.

Pasul 3: kbhit și kbcod

Pentru a genera codul tastei o singură dată, la apăsare, se consideră două scanări succesive cu *kbscan* și se detectează secvență 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 timp mai mare decât durata instabilității. **Prezentarea detaliată a acestei metode plus codul aferent se găsesc în prelegerea 4 la pagina 10.**

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 NOKEY 0x7f
#define DELAY 200

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

    unsigned char kbhit = 0;
    char kbcode;

    unsigned char loop_cnt=0;

    sysinit(); //nu este in curs, dar este necesar

    while(1){ //bucula 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 afiseaza-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, **chetați profesorul pentru validare**. Dacă ați ajuns aici aveți nota 5.

Partea opțională

Pasul 4: Shift LED

Creați alt proiect. Fie numele acestui proiect **kbd2**.

În noul proiect modificați *kbscan()* astfel încât tastele * și # să îndeplinească rolul de SHIFT. Unul din biții nefolosiți ai portului B va controla un LED. Acest LED trebuie să fie aprins atunci când una din tastele cu rol de SHIFT este apăsată. Când SHIFT este eliberată, LED-ul se va stinge.

În această fază codul generat la apăsare nu se modifică.

Dacă funcționează, chemați profesorul pentru validare. Dacă ați ajuns aici aveți între 1 și 2 puncte.

Pasul 5: Shift Code

În noul proiect modificați *kbscan()* astfel încât această funcție să întoarcă codul conform tabelului următor:

Ce serie pe tastă	Dacă SHIFT nu este apăsat funcția întoarce:	Dacă SHIFT este apăsat funcția întoarce:
0, 1, 2,...,9	0, 1, 2,...,9	0, 1, 2,...,9
A	a	A
B	b	B
C	c	C
D	d	D

Dacă sunt apăstate numai taste SHIFT (una sau amândouă) *kbscan* întoarce 0x7f.

Sugestie (puteți să o ignorați dacă aveți o metodă mai bună):

1. Scanare se face la fel. După încheierea scanării tastele de pe linia 3 vor fi tratate special.
2. Pentru tastele de pe linia 3 identificați dacă este apăsată vreo tastă SHIFT. De asemenea identificați dacă este apăsată ,0' sau ,D'.
3. Identificați următoarele situații: numai taste non SHIFT apăstate, numai taste SHIFT apăstate și atât taste SHIFT cât și non SHIFT apăstate. Generați codul pentru fiecare situație în parte.

Dacă funcționează, chemați profesorul pentru validare. Dacă ați ajuns aici aveți între 1 și 3 puncte.

Pasul 6: Deconectare

La terminare executați următoarele operații:

1. Se oprește execuția programului cu **Break**,
2. Se oprește depanatorul cu **Stop**
3. Se apasă **Con** și apoi se șterge memoria de program cu **Erase**
4. Se închide AVR Studio.
5. Se oprește alimentarea JTAG ICE.
6. Se oprește sursa Hameg
7. Se desface montajul, numai componentele adăugate în acest laborator.