Universitatea Tehnică din Cluj-Napoca
Departamentul Calculatoare

Proiectarea cu Micro-Procesoare

Lector: Mihai Negru

An 3 – Calculatoare și Tehnologia Informației

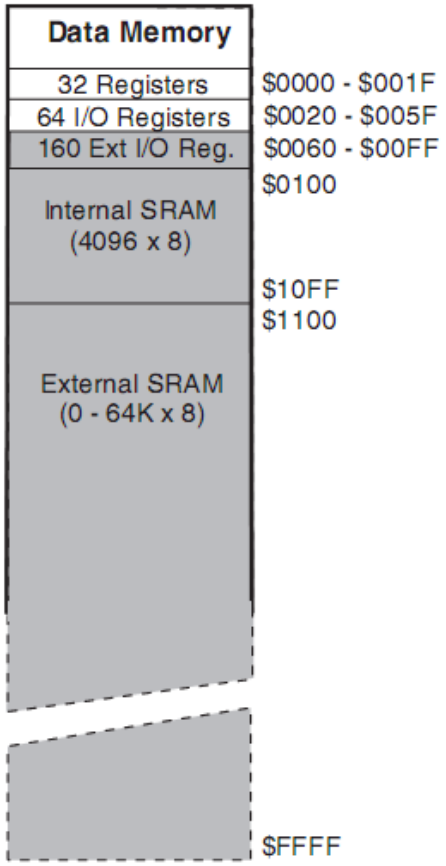
Seria B

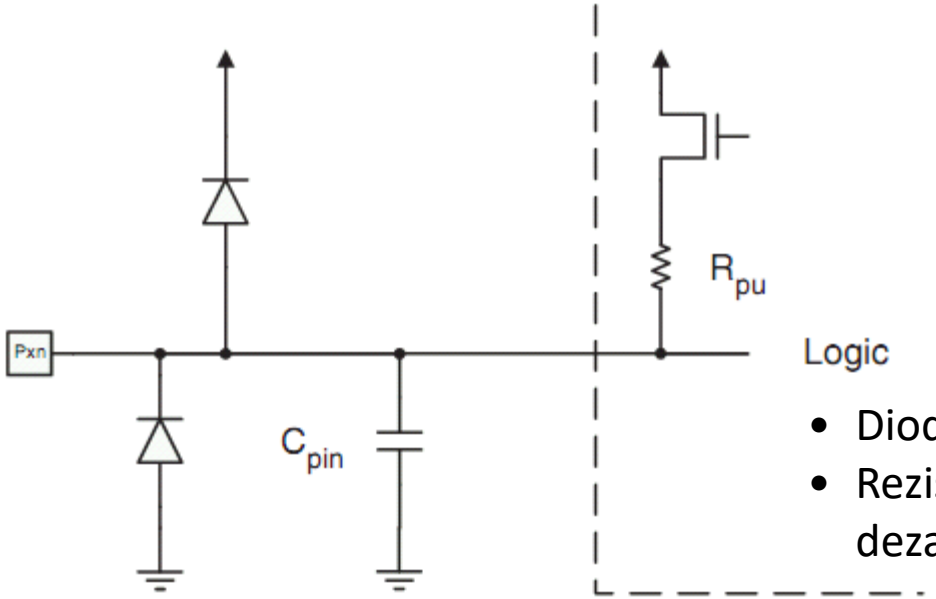
Curs 2: Intrare / Ieșire

https://mihai.utcluj.ro/

Intrare / Ieșire

- Porturi intrare/iesire:
  - ATmega 328P (UNO): **PORT B, C, D**
  - ATmega 2560 (MEGA): **PORT A, B, C, D, E, F, G, H, J, K, L**
- PORTA – PORTE: accesate prin instrucțiuni dedicate **in**, **out**
- PORTF – PORTL: doar prin **ld**, **st** (spațiul de adrese I/O extins)
- Registrul de direcție DDRx – fiecare bit din fiecare port poate fi configurat ca intrare / ieșire
- Scrierea portului se face prin registrul PORTx





- Diode de protecție, împotriva electricității statice
- Rezistenta “pull-up”, care poate fi activată / dezactivată prin logica

Intrare / Ieșire

- Exista trei addrese de memorie pentru fiecare port de intrare / ieșire port x (A… L):
  - Data Register – PORTx,
  - Data Direction Register – DDRx
  - Port Input Pins – PINx

Exemplu: PORTA

PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x02 (0x22)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x01 (0x21)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

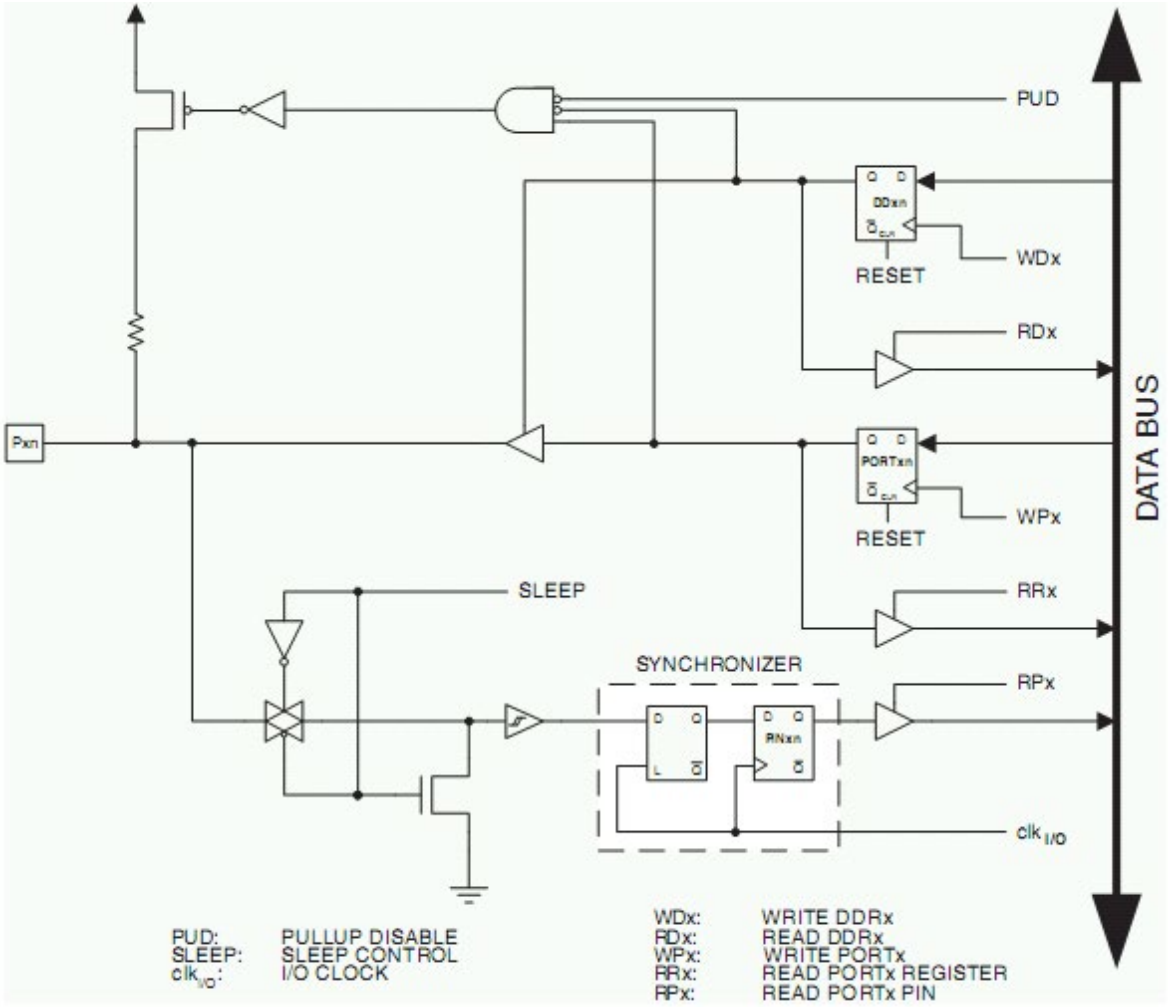
PINA – Port A Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x00 (0x20)	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Notație: PORTxn = pin n de la PORTx (ex: PORTB3 – bitul 3 din portul B).

Intrare / Ieșire

- Schema generală pentru 1 bit dintr-un port I/O



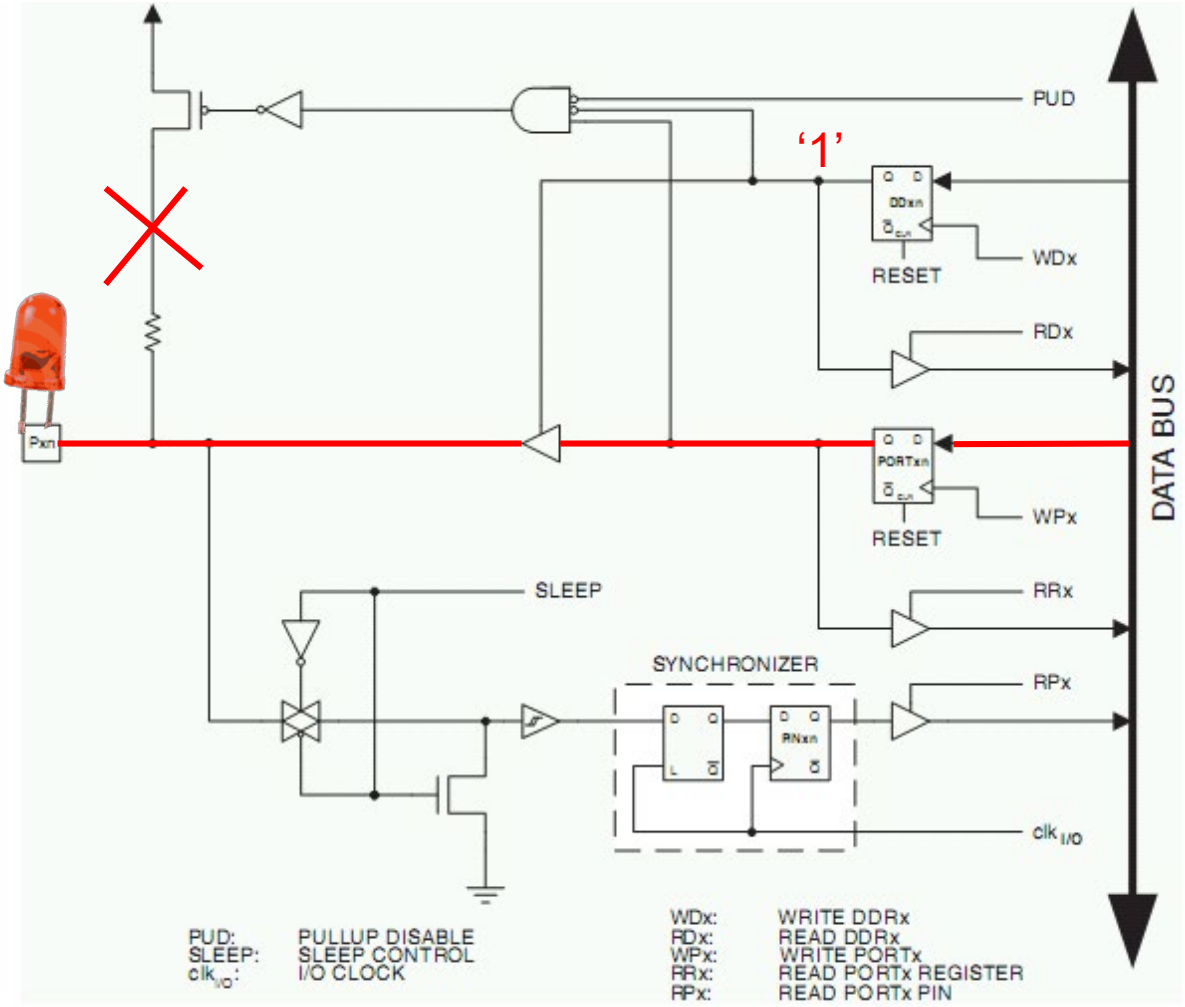
Control direcție

Datele ce vor fi trimise la ieșire

Datele citite de pe intrare

Intrare / Ieșire

- Configurația pentru ieșire

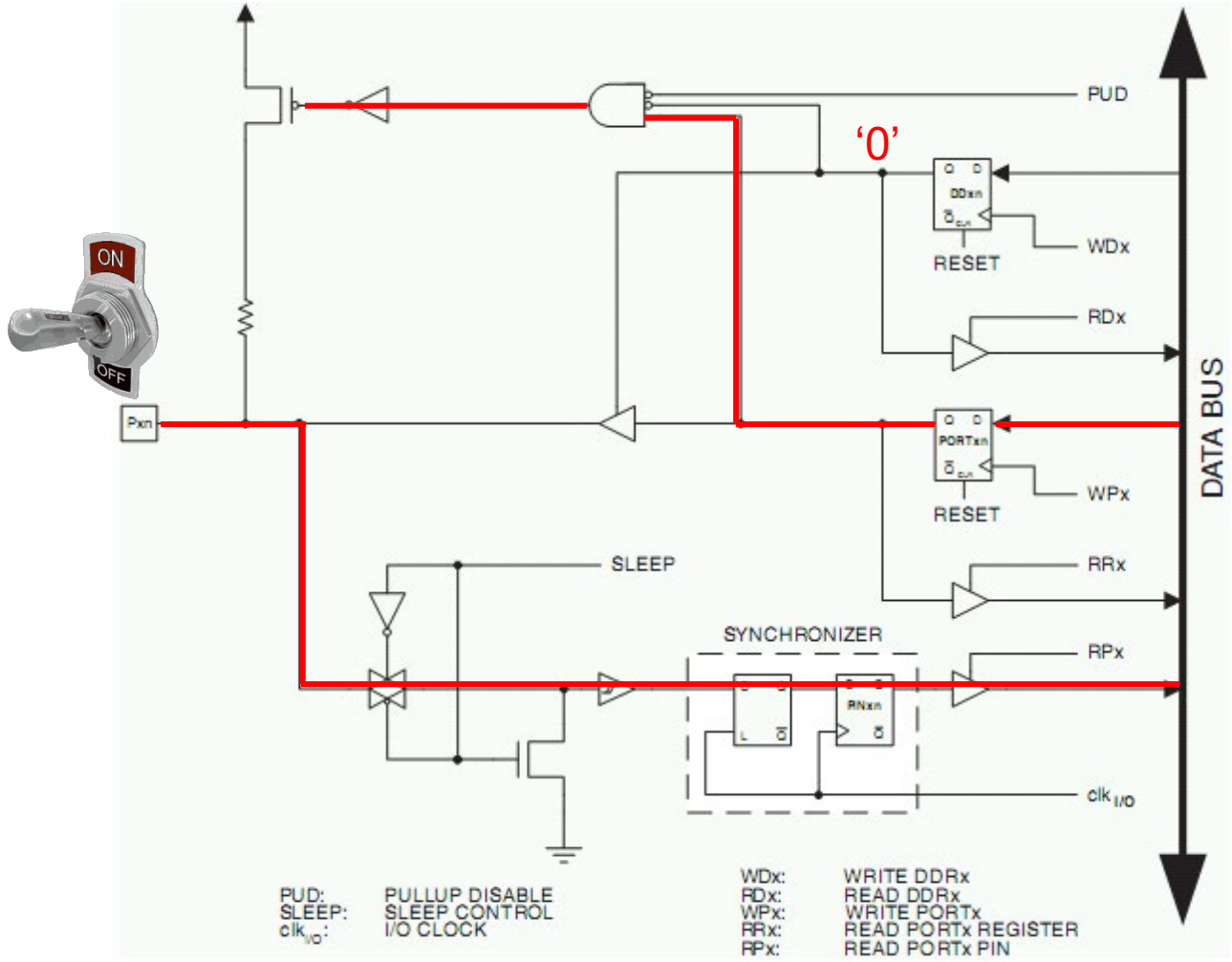


Direcție = 1

Datele scrise în PORTx sunt trimise la ieșire

Intrare / Ieșire

- Configurația pentru intrare



Direcție = 0

'1' scris in PORTx activează rezistenta pull-up

Datele devin disponibile la PINx

Intrare / Ieșire

- Stări posibile ale pinilor I/O

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

- PUD – Pull Up Disable: GLOBAL
  - Valoarea ‘1’ a bitului 4 din MCUCR dezactivează toate rezistentele pull-up

MCUCR – MCU Control Register

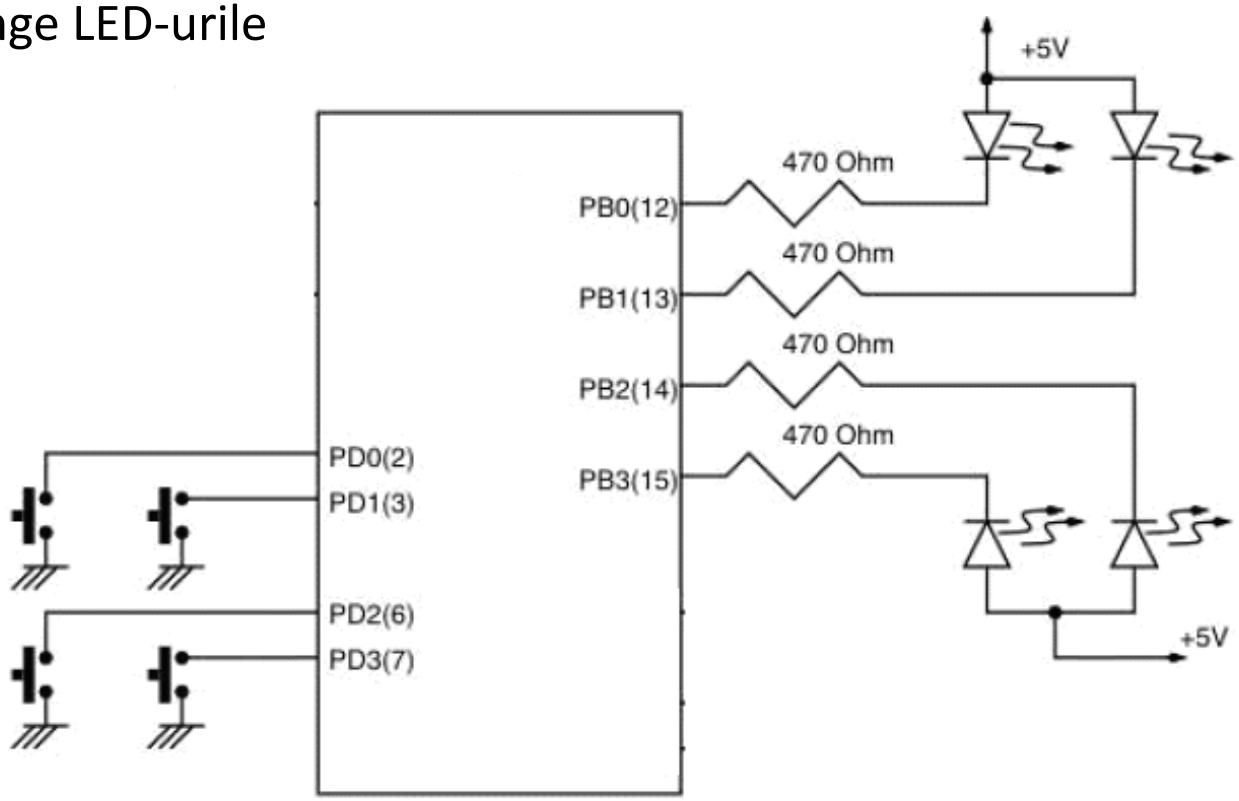
Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	JTD	–	–	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R/W	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

```
in r17, MCUCR
ori r17, 0b00010000
out MCUCR, r17
```

```
sbi MCUCR, 4
```

Intrare / Ieșire – Butoane si LED-uri

- Rezistentele pull-up asigură nivelul ‘1’ pe pin când butonul este in repaus
- Când butonul este apăsat, nivelul pinului este ‘0’ prin legare la GND
- Un nivel ‘0’ pe pinii de ieșire (B) cauzează diferența de potențial pe LED-uri, provocând aprinderea lor
- Nivelul ‘1’ pe pinii B stinge LED-urile



Intrare / Ieșire – Butoane si LED-uri

- **Scrierea programului**

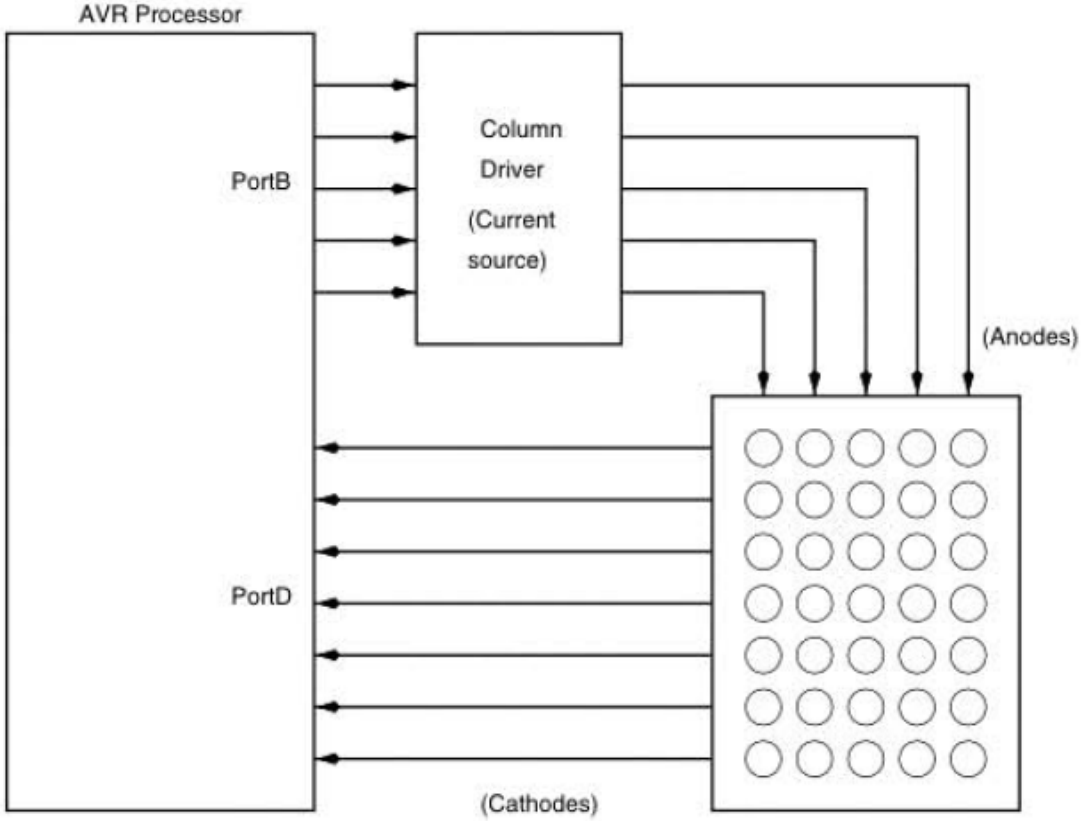
```
        ldi r16, 0x00
        out DDRD, r16     Direcția portului D - intrare
        ldi r16, 0xFF
        out PORTD, r16    '1' in PORTD – rezistente pull-up activate
        ldi r16, 0xFF
        out DDRB, r16     Direcția portului B - ieșire
loop:
        in r16, PIND      Citire port D
        out PORTB, r16    Scriere port B
        rjmp loop
```

- **Atenție!!!**

```
        in r16, PIND      Citește starea pinilor exteriori,
                          modificată de activitate exterioară
        in r16, PORTD     Citește starea registrului PORTD,
                          setat din interiorul micro-controllerului prin program
```

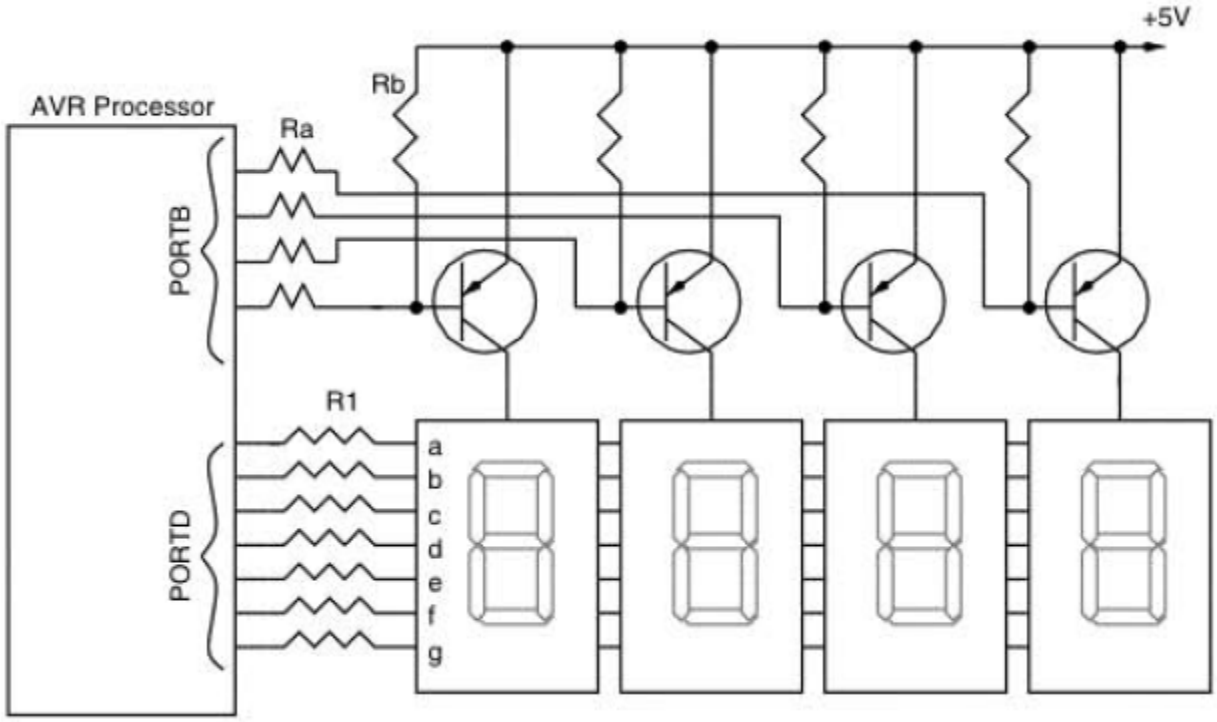
Intrare / Ieșire – Matrice de LED-uri

- Ambele porturi (D si B) sunt ieșire
- Pentru ca un LED sa se aprindă, anodul trebuie sa fie in ‘1’ si catodul in ‘0’
- Se poate controla o linie sau o coloana simultan
- Pentru utilizarea întregii matrici – **baleiere**



Intrare / Ieșire – Bloc 4x7 segmente

- Fiecare cifra este alcătuita din 7 led-uri, cu anod comun
- Nivelul ‘1’ pe anod activează cifra – una singură activă la un moment dat
- Valori selective de ‘0’ pe fiecare catod realizează modelul cifrei
- Baleiere pentru utilizarea întregului dispozitiv



Rezistențe limitatoare de curent pentru LED-uri

- Fiecare tip de LED are o cădere tipică de tensiune directă V_f
- Diferența de tensiune dintre voltajul de ieșire (digital) V_{cc} și V_f reprezintă căderea de tensiune pe rezistența de limitare
- Intensitatea curentului prin pinul de ieșire este:

$$I = \frac{V_R}{R} = \frac{V_{cc} - V_f}{R}$$

- Cu cât este mai mare intensitatea, cu atât LED-ul este mai luminos
- I trebuie limitată sub 20 mA, pentru a proteja microcontrolerul (și LED-ul).

Typical LED Characteristics			
Semiconductor Material	Wavelength	Colour	V_F @ 20mA
GaAs	850-940nm	Infra-Red	1.2v
GaAsP	630-660nm	Red	1.8v
GaAsP	605-620nm	Amber	2.0v
GaAsP:N	585-595nm	Yellow	2.2v
AlGaP	550-570nm	Green	3.5v
SiC	430-505nm	Blue	3.6v
GaInN	450nm	White	4.0v

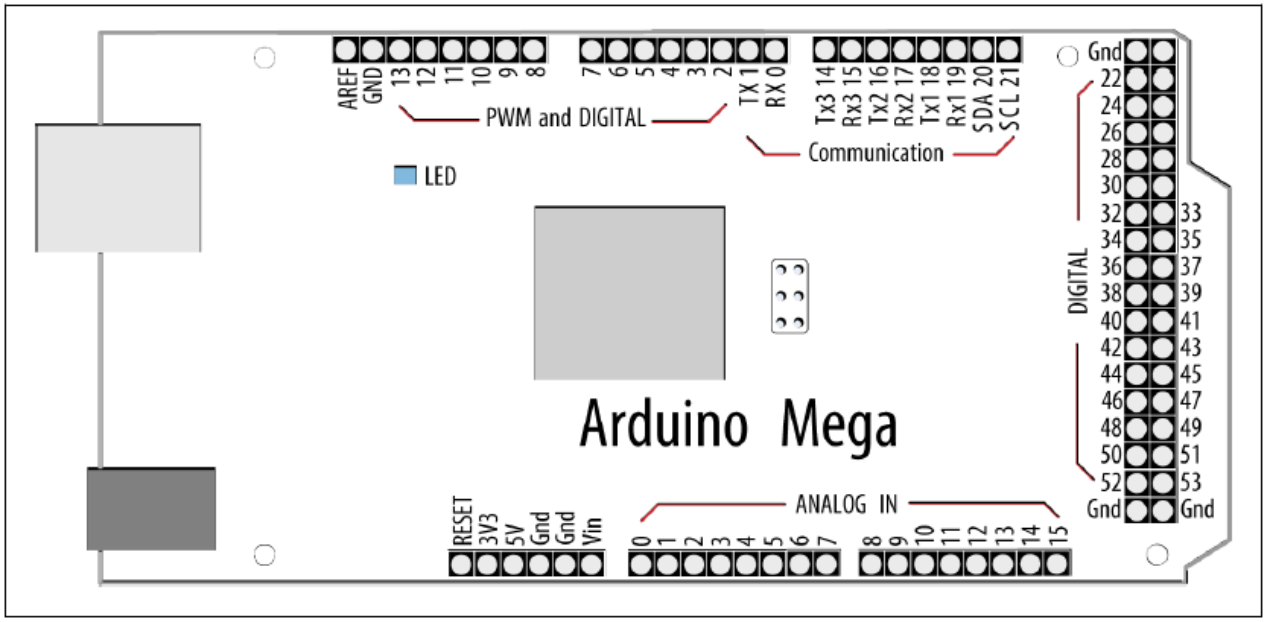
http://www.electronics-tutorials.ws/diode/diode_8.html

Exemplu:

- 1 Roșu LED: $V_f = 1.8$ V $V_{CC} = 5$ V
- Pentru I = 10 mA
  R = (5 V – 1.8 V) / 0.01A = 320 Ohm
- Pentru I = 20 mA
  R = (5 V – 1.8 V) / 0.02A = 160 Ohm

Intrare / Ieșire la sistemele Arduino

- Pini de intrare/ieșire digitali, conectați la porturile microcontrollerului
- Mediul de dezvoltare se ocupa de problema corespondentei
- Logica de programare este orientată pe numarul pin-ului
- O parte din pini au functii speciale (comunicație serială UART sau I2C, generator de undă PWM, sau semnale analogice)
- Pinii care au funcția RX0 și TX0 trebuie evitați – rezervați pentru comunicarea seriala prin USB, care include programarea placii
- De obicei există un LED pe placă, conectat la pin-ul 13



Intrare / Ieșire la sistemele Arduino

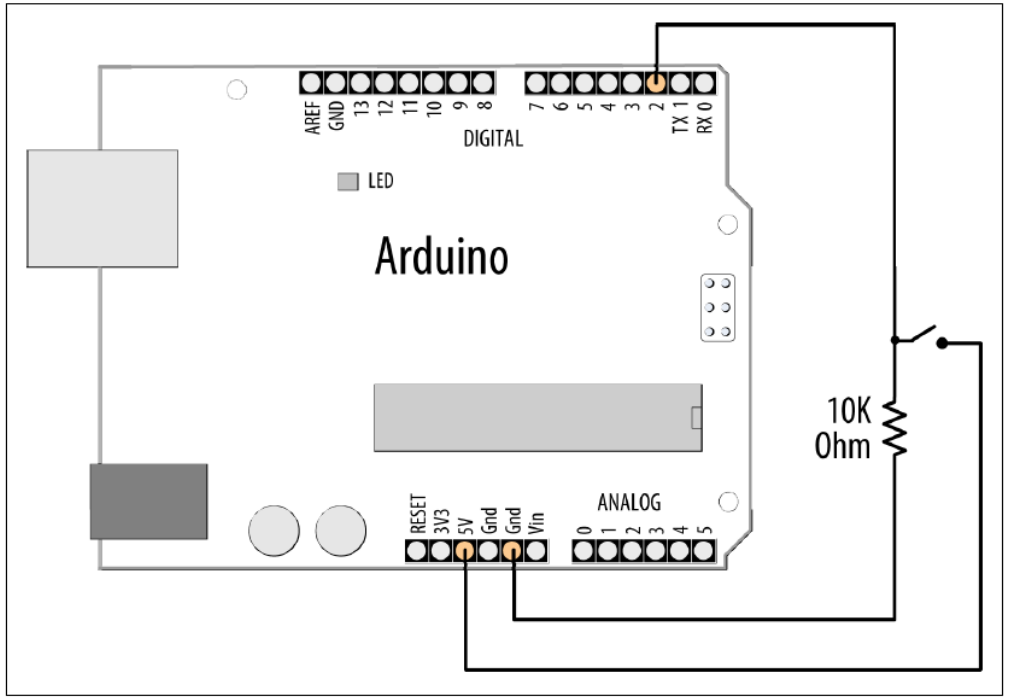
- Corespondență pinilor cu porturile microcontrollerului ATMega2560
- http://arduino.cc/en/Hacking/PinMapping2560
- Selecție:

43	PD0 (SCL/INT0)	Digital pin 21 (SCL)
44	PD1 (SDA/INT1)	Digital pin 20 (SDA)
45	PD2 (RXD1/INT2)	Digital pin 19 (RX1)
46	PD3 (TXD1/INT3)	Digital pin 18 (TX1)
47	PD4 (ICP1)	
48	PD5 (XCK1)	
49	PD6 (T1)	
50	PD7 (T0)	Digital pin 38

71	PA7 (AD7)	Digital pin 29
72	PA6 (AD6)	Digital pin 28
73	PA5 (AD5)	Digital pin 27
74	PA4 (AD4)	Digital pin 26
75	PA3 (AD3)	Digital pin 25
76	PA2 (AD2)	Digital pin 24
77	PA1 (AD1)	Digital pin 23
78	PA0 (AD0)	Digital pin 22

Intrare / Ieșire la sistemele Arduino

- Sursa elementară de semnal: un buton conectat la un pin de intrare digital
- Se folosește o rezistență “pull down”, pentru ca atunci când butonul nu este apăsat, semnalul de intrare să fie nivel logic ‘0’
- Pentru ieșire, se folosește led-ul de pe placă



Intrare / Ieșire la sistemele Arduino

- Exemplu:

```
const int ledPin = 13;
const int inputPin = 2;

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

void loop(){
  int val = digitalRead(inputPin);
  if (val == HIGH)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

```
// Constante pentru numarul pinilor implicati
// se pot folosi direct numerele, dar solutia aceasta e mai flexibila

// configurarea directiei pinilor
// se declara pin-ul legat la LED ca iesire
// si cel legat la buton ca intrare

// citire stare buton
// daca este apasat, se scrie '1' pe pinul led-ului

// altfel se scrie '0'
```

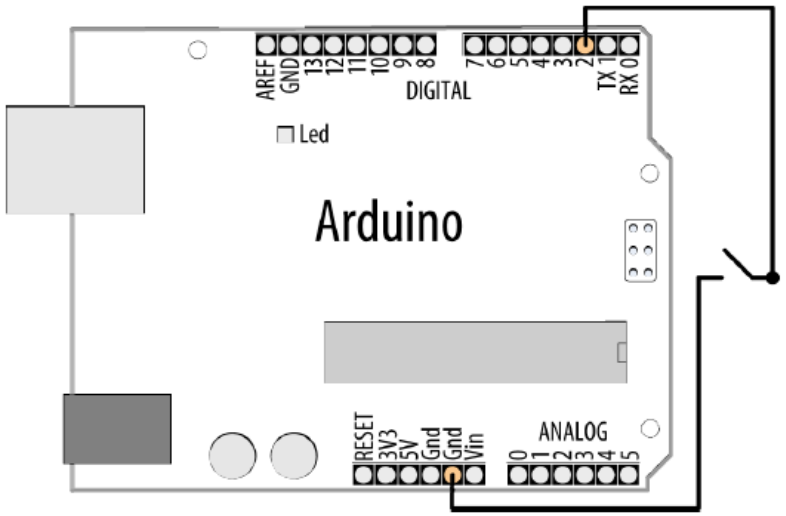
// Evident, se poate si transfera direct starea butonului catre LED:

```
void loop()
{
    digitalWrite(ledPin, digitalRead(inputPin));

}
```

Intrare / Ieșire la sistemele Arduino

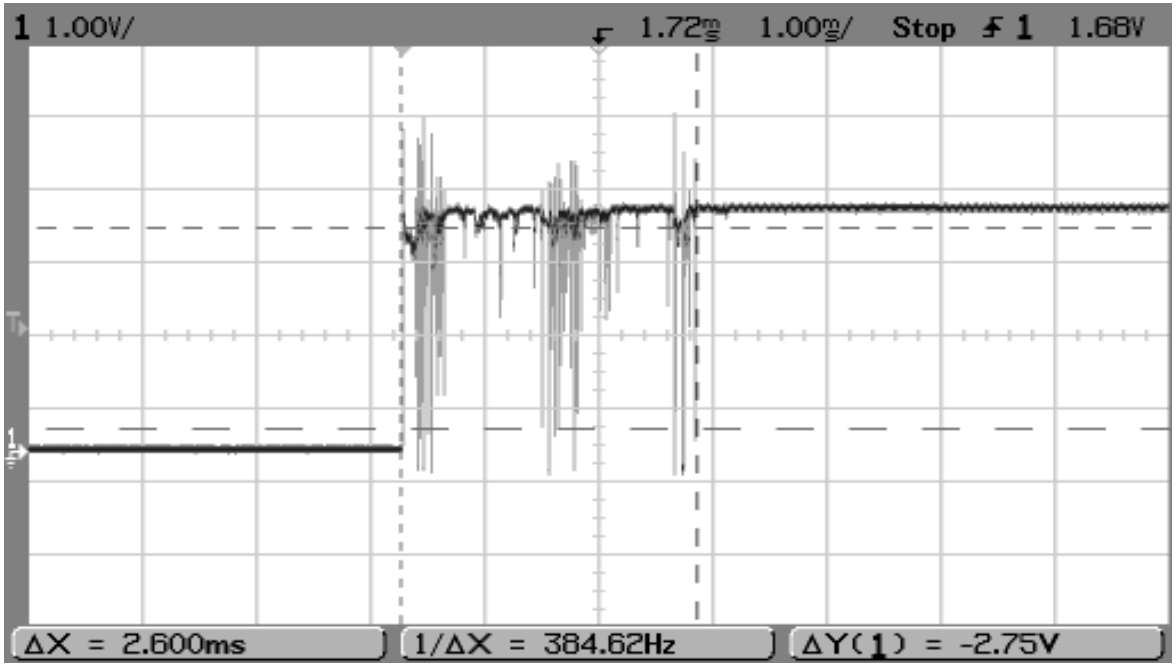
- Folosirea unui switch fară rezistențe externe
- Se folosesc rezistențele ‘Pull Up’ atașate fiecărui pin

```
const int ledPin = 13;                  // Aceleasi constante, aceiasi pini
const int inputPin = 2;

void setup() {
  pinMode(ledPin, OUTPUT);              // configurarea directiei pinilor
  pinMode(inputPin, INPUT);             // se declara pin-ul legat la LED ca iesire
  digitalWrite(inputPin,HIGH);  //      // activare rezistente pull up prin scriera unei valori ‘HIGH’
}                                       // pe pin-ul de intrare!
void loop(){
  int val = digitalRead(inputPin);      // acelasi cod ca inainte
  if (val == HIGH)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```



Intrare / Ieșire la sistemele Arduino

- **Citirea unor date de intrare instabile**
  - Un contact mecanic poate oscila intre pozitia “închis” și “deschis” de mai multe ori pana la stabilizare.
  - Un microcontroller poate fi suficient de rapid pentru a sesiza unele dintre aceste oscilații, percepându-le ca apăsari multiple pe buton.
  - Unele dispozitive, precum Pmod BTN, au circuite speciale pentru eliminarea acestor oscilații.
  - Dacă aceste circuite nu există, problema trebuie rezolvată prin software.



Intrare / Ieșire la sistemele Arduino

- **Citirea unor date de intrare instabile**
  - Principiul filtrării oscilațiilor prin software: se verifică starea intrării de mai multe ori, până când aceasta nu se mai modifică.
  - Efectul: ignorarea perioadei de instabilitate, validând intrarea doar atunci când aceasta e stabilă.

```
const int inputPin = 2;
const int ledPin = 13;
const int debounceDelay = 10;    // Intervalul de timp (ms) in care semnalul trebuie sa fie stabil

boolean debounce(int pin)    // Functia returneaza starea intrarii, dupa stabilizare
{
  boolean state;             // Stare curenta, stare anterioara
  boolean previousState;

  previousState = digitalRead(pin);    // Prima stare
  for(int counter=0; counter < debounceDelay; counter++)   // Se parcurge intervalul de timp
  {
      delay(1);                        // Se asteapta 1 ms
      state = digitalRead(pin);        // Citire stare prezenta
      if( state != previousState)      // Daca starile difera, o luam de la inceput
      {
         counter = 0; // Numaratorul primeste din nou valoarea zero
         previousState = state;   // Starea curenta devine starea initiala pentru noul ciclu
      }
  }
  //  // Daca s-a ajuns aici, inseamna ca semnalul a fost stabil tot intervalul de timp
  return state;    // Se returneaza starea curenta, stabila
}
```

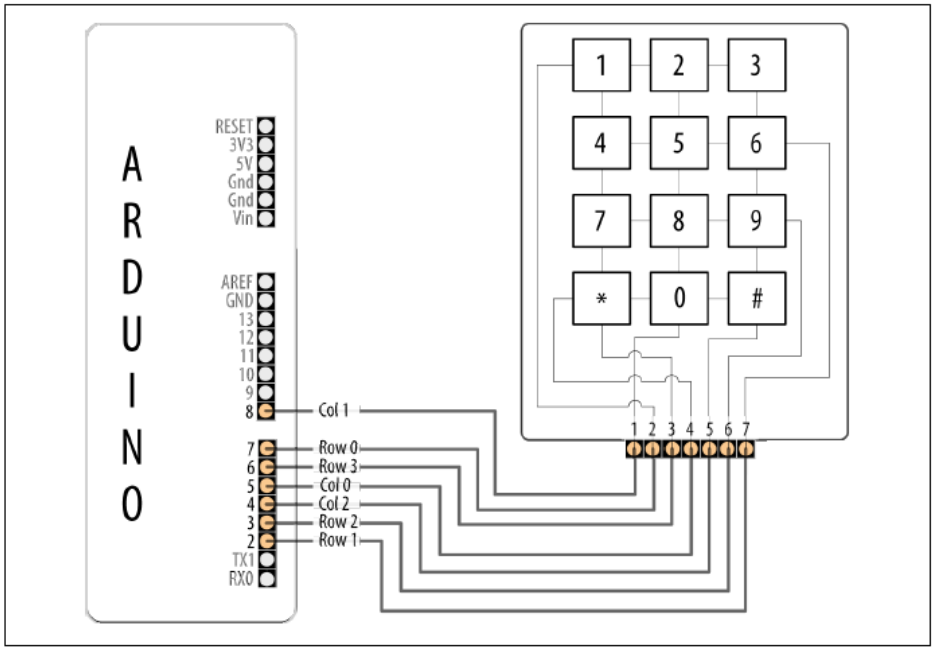
Intrare / Ieșire la sistemele Arduino

- **Citirea unor date de intrare instabile**

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

void loop()
{
  if (debounce(inputPin))          // Se foloseste functia debounce() in loc de digitalRead()
  {
    digitalWrite(ledPin, HIGH);
  }
}
```

Intrare / Ieșire la sistemele Arduino

- **I/O pe mai mulți pini. Utilizarea unei tastaturi**
  - Apăsarea unei taste face contact intre coloană si rând
  - Starea rândurilor este implicit ‘1’, prin folosirea unor rezistente “pull-up”
  - Dacă coloana pe care se află o tastă este ‘0’, si tasta este apasată, randul tastei devine ‘0’. Daca coloana pe care se afla o tasta este ‘1’, nu se intamplă nimic la apăsarea tastei.
  - Principiu: activarea pe rând a coloanelor (punerea lor pe rând la ‘0’), și citirea stării rândurilor
  - Coloanele trebuie legate la pini configurați ca ieșire, rândurile la pini configurați ca intrare



Arduino pin	Keypad connector	Keypad row/column
2	7	Row 1
3	6	Row 2
4	5	Column 2
5	4	Column 0
6	3	Row 3
7	2	Row 0
8	1	Column 1

Intrare / Ieșire la sistemele Arduino

- **I/O pe mai mulți pini. Utilizarea unei tastaturi**

```
const int numRows = 4;       // number of rows in the keypad
const int numCols = 3;       // number of columns
const int debounceTime = 20; // number of milliseconds for switch to be stable

// Se definesc pinii atasati randurilor si coloanelor, aranjati in ordinea logica
const int rowPins[numRows] = { 7, 2, 3, 6 }; // Rows 0 through 3
const int colPins[numCols] = { 5, 8, 4 };    // Columns 0 through 2

// LUT pentru identificarea tastei de la intersectia unui rand cu o coloana
const char keymap[numRows][numCols] = {
  { '1', '2', '3'  } ,
  { '4', '5', '6'  } ,
  { '7', '8', '9'  } ,
  { '*', '0', '#'  }
};
    void setup()   // Initializarea sistemului
    {
      Serial.begin(9600);  // Initializarea interfetei seriale via USB, folosita pentru comunicarea cu calculatorul
      for (int row = 0; row < numRows; row++)
      {
        pinMode(rowPins[row],INPUT);          // Pinii randurilor sunt intrare
        digitalWrite(rowPins[row],HIGH);      // Se activeaza rezistentele pull-up
      }
      for (int column = 0; column < numCols; column++)
      {
        pinMode(colPins[column],OUTPUT);      // Pinii coloanelor sunt iesire

        digitalWrite(colPins[column],HIGH);   // Initial toate sunt '1', inactive
      }
    }
```

Intrare / Ieșire la sistemele Arduino

- **I/O pe mai mulți pini. Utilizarea unei tastaturi**

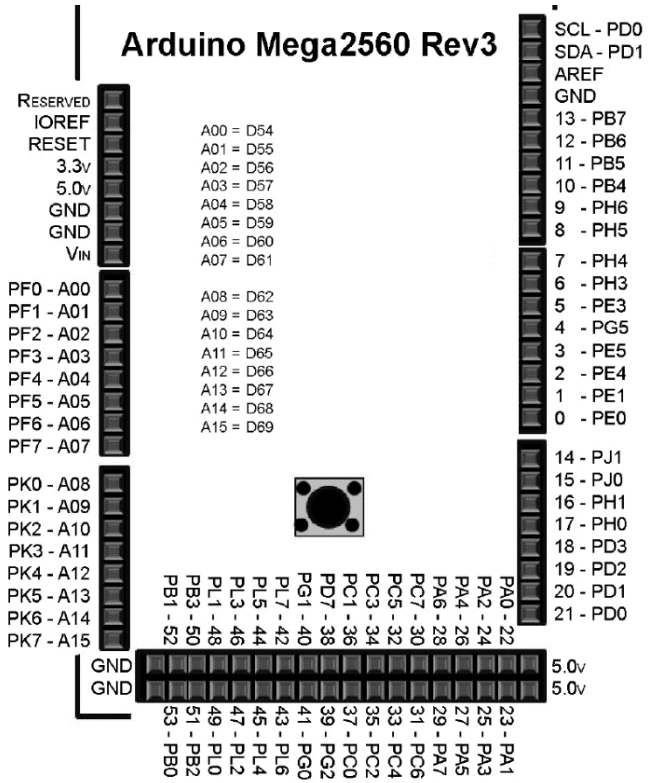
```
void loop()
{
  char key = getKey();      // Se apeleaza functia de citire a unei taste (mai jos)
  if( key != 0) {           // Daca functia returneaza '0', nicio tasta nu este apasata
                            // daca rezultatul e diferit de zero, este apasata o tasta, si functia returneaza codul acesteia
    Serial.print("Got key ")  // Folosirea interfetei seriale pentru a afisa in consola mesajul tasta apasata
    Serial.println(key);      // si codul acestei taste
  }
}

 // functia principala: returneaza codul tastei, sau 0 daca nicio tasta nu e apasata.
char getKey()
{
  char key = 0;                                   // codul implicit zero, nicio tasta apasata

  for(int column = 0; column < numCols; column++)  // baleierea coloanelor
  {
    digitalWrite(colPins[column],LOW);            // se activeaza coloana curenta
    for(int row = 0; row < numRows; row++)        // se verifica randurile unul cate unul

    {
      if(digitalRead(rowPins[row]) == LOW)        // daca randul e '0', avem tasta apasata pe acel rand
      {
        delay(debounceTime);                      // intarziere pentru filtrare intrare
        while(digitalRead(rowPins[row]) == LOW)
            :                                     // asteptare eliberare tasta
        key = keymap[row][column];                // se cunoaste coloana si randul tastei apasate
                                                  // se foloseste LUT pentru determinarea codului ASCII al tastei
      }
    }
    digitalWrite(colPins[column],HIGH);           // dezactivare coloana
  }
  return key;  // returns the key pressed or 0 if none
}              // returneaza codul tastei, sau 0
```

Intrare / Ieșire la sistemele Arduino

- **I/O folosind porturile microcontrollerului**

- **Dezavantaje**
  - Abordare dependentă de hardware, nu este portabilă intre plăci diferite
  - Trebuie cunoscută relația dintre pin și portul/bitul corespunzator
  - Unele porturi sunt rezervate, si modificarea stării lor nu este recomandabilă

- **Avantaje**
  - Viteza ridicată. Scrierea și citirea unui port sunt de aproximativ 10 ori mai rapide decat digitalWrite() si digitalRead()
  - Posibilitatea de a citi mai mulți pini simultan, sau de a scrie mai mulți pini simultan (digitalRead și digitalWrite lucreaza doar la nivel de pin)



Intrare / Ieșire la sistemele Arduino

- **Exemplu:** se conectează 8 led-uri la pinii 22…29 ai Arduino Mega (la PortA). Se dorește aprinderea alternativă a led-urilor pare și impare, cu o intarziere de 1 secundă între comutații
- **Cod sursa, abordarea clasică Arduino**:

```
const  int PortAPins[8]={22, 23, 24, 25, 26, 27, 28, 29}

void setup()
{
    for (int b=0; b<8; b++)
        pinMode(PortAPins[b], OUTPUT);          // toti pinii sunt configurati ca iesire
}
void loop()
{
    for (int b=0; b<8; b+=2)                    // b=0, 2, 4, 6
    {
        digitalWrite(PortAPins[b], HIGH);       // scriem ‘1’ pe pinii pari
        digitalWrite(PortAPins[b+1], LOW);      // scriem ‘0’ pe pinii impari
    }
    delay(1000);                                // asteptare de 1 secunda (1000 ms)
    for (int b=0; b<8; b+=2)
    {
        digitalWrite(PortAPins[b], LOW);        // scriem ‘0’ pe pinii pari
        digitalWrite(PortAPins[b+1], HIGH);     // scriem ‘1’ pe pinii pari
    }
    delay(1000);                                // asteptare de 1 secunda (1000 ms)
}
```

Intrare / Ieșire la sistemele Arduino

- **Exemplu:** se conectează 8 led-uri la pinii 22…29 ai Arduino Mega (la PortA). Se dorește aprinderea alternativă a led-urilor pare și impare, cu o intarziere de 1 secundă între comutații
- **Cod sursa, abordarea folosind portul A al ATMega2560** :

```
void setup()
{
    DDRA = 0B11111111;                  // toti pinii atasati portului A sunt configurati ca iesire
}

void loop()
{
     PORTA = 0B01010101;                // 1 pe pinii pari, 0 pe pinii impari
     delay(1000);                       // asteptare de 1 secunda (1000 ms)
     PORTA = 0B10101010;                // 0 pe pinii pari, 1 pe pinii impari
     delay(1000);                       // asteptare de 1 secunda (1000 ms)
}
```

Referințe

1. Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V datasheet
2. Atmel Atmega64 datasheet