Proiectare cu Microprocesoare

Curs 3

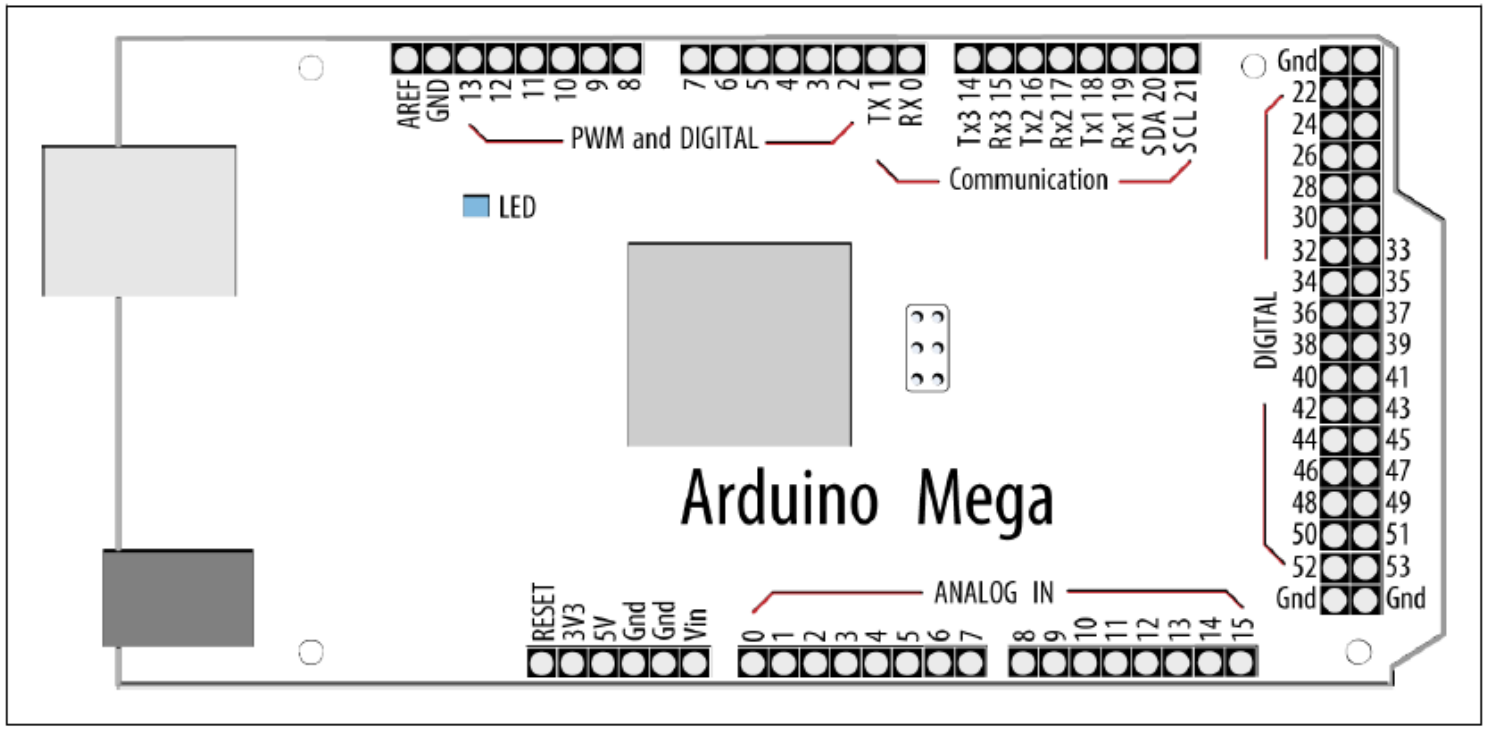
I/O si intreruperi cu Arduino

An 3 CTI
Semestrul I

Lector: Răzvan Itu

Intrare / Iesire la sistemele Arduino

- Pini de intrare/iesire digitali, conectati la porturile microcontrollerului
- Mediul de dezvoltare se ocupa de problema corespondentei
- Logica de programare este orientata pe numarul pin-ului
- O parte din pini au functii speciale (comunicatie seriala UART sau I2C, generator de unda PWM, sau semnale analogice)
- Pinii care au functia RX si TX trebuie evitati! Sunt rezervati pentru comunicarea seriala prin USB, care include programarea placii
- De obicei exista un LED pe placa, conectat la pin-ul 13



Intrare / Iesire la sistemele Arduino

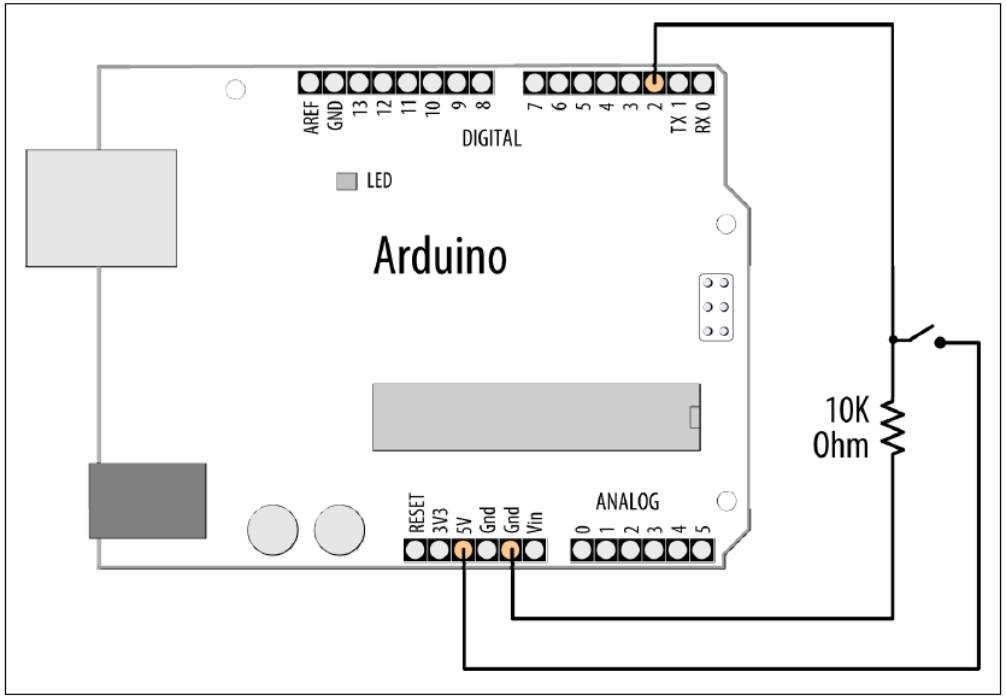
- Corespondenta pinilor cu porturile microcontrollerului ATMega2560
- http://arduino.cc/en/Hacking/PinMapping2560
- Selectie:

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 / Iesire la sistemele Arduino

- Sursa elementara de semnal: un buton conectat la un pin de intrare digital
- Se foloseste o rezistenta “pull down”, pentru ca atunci cand butonul nu este apasat, semnalul de intrare sa fie nivel logic ‘0’
- Pentru iesire, se foloseste led-ul de pe placa



Intrare / Iesire la sistemele Arduino

- Cod 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:

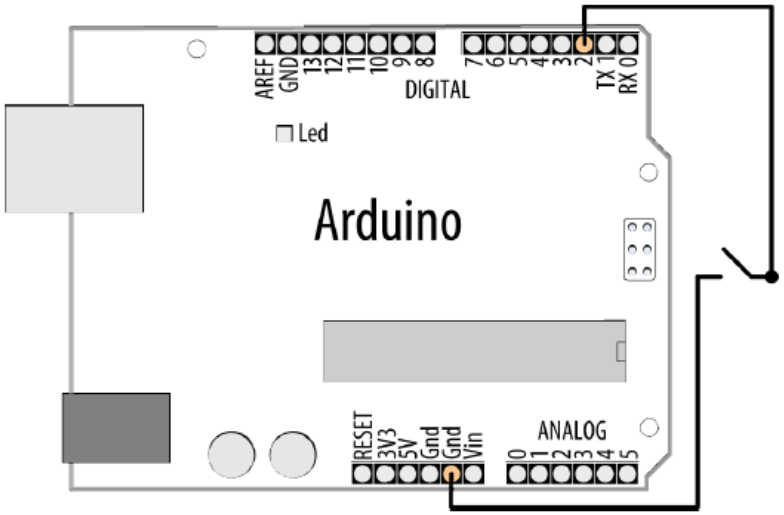
```
void loop()
{
    digitalWrite(ledPin, digitalRead(inputPin));

}
```

Intrare / Iesire la sistemele Arduino

- Folosirea unui switch fara rezistente externe
- Se folosesc rezistentele ‘Pull Up’ atasate fiecarui pin

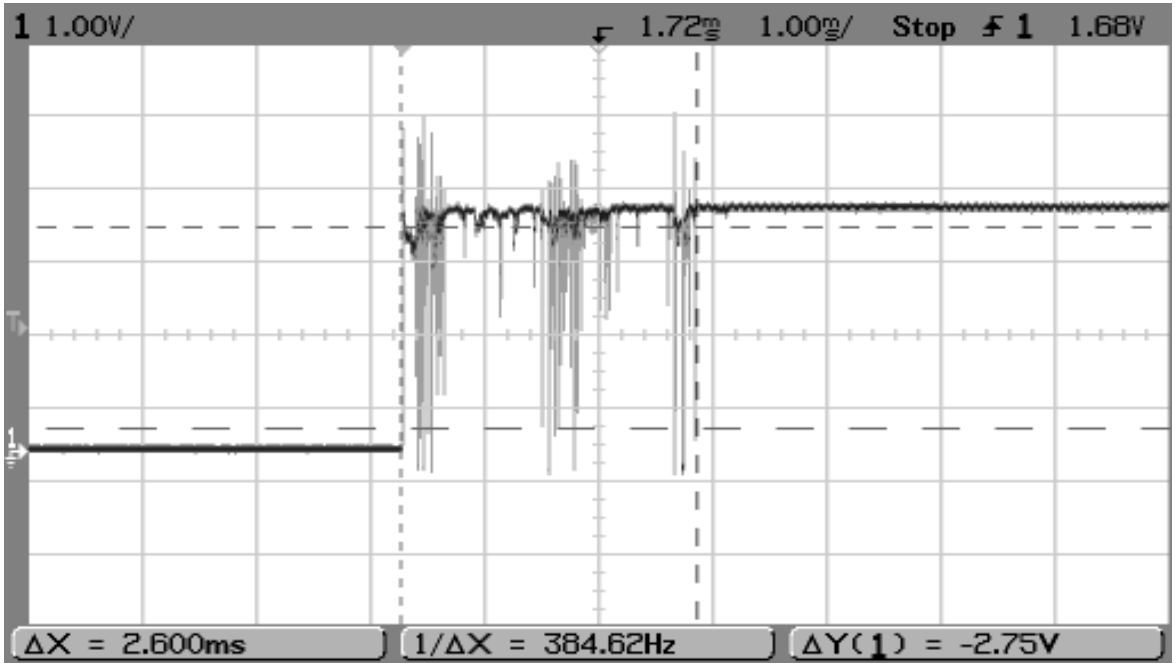
```
const int ledPin = 13;
const int inputPin = 2;

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

```
// Aceleasi constante, aceiasi pini

// configurarea directiei pinilor
// se declara pin-ul legat la LED ca iesire
// activare rezistente pull up prin scriera unei valori ‘HIGH’
// pe pin-ul de intrare!

// acelasi cod ca inainte
```



Intrare / Iesire la sistemele Arduino

- **Citirea unor date de intrare instabile**
  - Un contact mecanic poate oscila intre pozitia “inchis” si “deschis” de mai multe ori pana la stabilizare.
  - Un microcontroller poate fi suficient de rapid pentru a sesiza unele dintre aceste oscilatii, percepandu-le ca apasari multiple pe buton.
  - Unele dispozitive, precum Pmod BTN, au circuite speciale pentru eliminarea acestor oscilatii.
  - Daca aceste circuite nu exista, problema trebuie rezolvata prin software.



Intrare / Iesire la sistemele Arduino

- **Citirea unor date de intrare instabile**
  - Principiul filtrarii oscilatiilor prin software: se verifica starea intrarii de mai multe ori, pana cand aceasta nu se mai modifica.
  - Efectul: ignorarea perioadei de instabilitate, validand intrarea doar atunci cand aceasta e stabila.
  - Cod sursa exemplu:

```
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
}
```

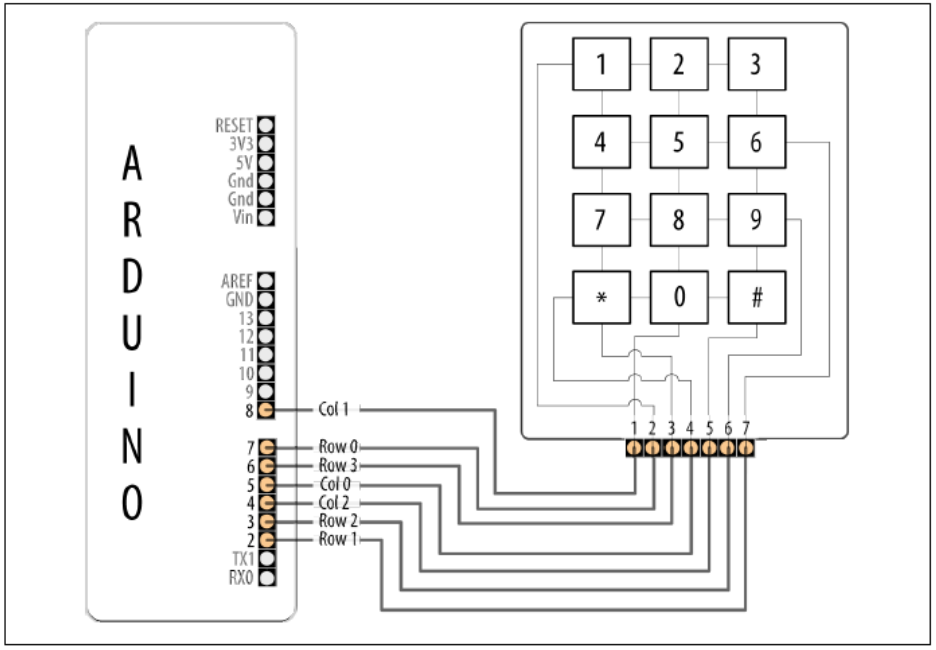
Intrare / Iesire la sistemele Arduino

- **Citirea unor date de intrare instabile**
  - Cod sursa exemplu (continuare):

```
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 / Iesire la sistemele Arduino

- **I/O pe mai multi pini. Utilizarea unei tastaturi**
  - Apasarea unei taste face contact intre coloana si rand
  - Starea randurilor este implicit ‘1’, prin folosirea unor rezistente ‘pull up’
  - Daca coloana pe care se afla o tasta este ‘0’, si tasta este apasata, randul tastei devine ‘0’. Daca coloana pe care se afla o tasta este ‘1’, nu se intampla nimic la apasarea tastei.
  - Principiu: activarea pe rand a coloanelor (punerea lor pe rand la ‘0’), si citirea starii randurilor
  - Coloanele trebuie legate la pini configurati ca iesire, randurile la pini configurati 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 / Iesire la sistemele Arduino

- **I/O pe mai multi pini. Utilizarea unei tastaturi**
  - Cod exemplu:

```
const int numRows = 4;          // Numarul de randuri
const int numCols = 3;          // Numarul de coloane
const int debounceTime = 20;    // Numarul de milisecunde de asteptare

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

// 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
  }
}
```

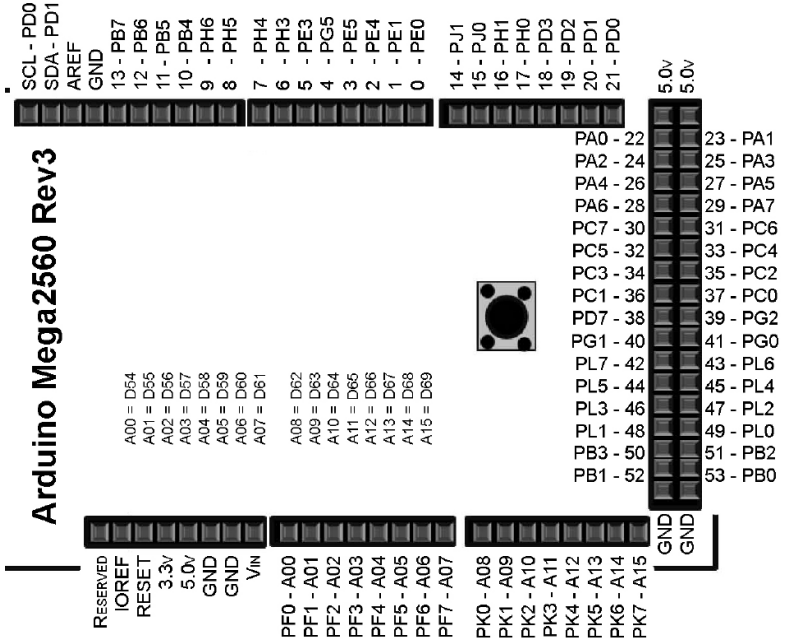
Intrare / Iesire la sistemele Arduino

- **I/O pe mai multi pini. Utilizarea unei tastaturi**
  - Cod exemplu (continuare):

```
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;   // returneaza codul tastei, sau 0
}
```

Intrare / Iesire la sistemele Arduino

- **I/O folosind porturile microcontrollerului**
- **Dezavantaje**
  - Abordare dependenta de hardware, nu este portabila intre placi diferite
  - Trebuie cunoscuta relatia dintre pin si portul/bitul corespunzator
  - Unele porturi sunt rezervate, si modificarea starii lor nu este recomandabila
- **Avantaje**
  - Viteza ridicata. Scrierea si citirea unui port sunt de aproximativ 10 ori mai rapide decat digitalWrite() si digitalRead()
  - Posibilitatea de a citi mai multi pini simultan, sau de a scrie mai multi pini simultan (digitalRead si digitalWrite lucreaza doar la nivel de pin)



Intrare / Iesire la sistemele Arduino

- **Exemplu:** se leaga 8 led-uri la pinii 22…29 ai Arduino Mega (conectati la PortA). Se doreste aprinderea alternativa a led-urilor pare si impare, cu o intarziere de 1 secunda intre comutatii
- **Cod sursa, abordarea clasica 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 / Iesire la sistemele Arduino

- **Exemplu:** se leaga 8 led-uri la pinii 22…29 ai Arduino Mega (conectati la PortA). Se doreste aprinderea alternativa a led-urilor pare si impare, cu o intarziere de 1 secunda intre comutatii
- **Cod sursa, abordarea folosind portul A al ATMega2560**:

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

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

Tratarea intreruperilor externe

- Detectarea unor evenimente pe pini, fara a verifica in permanenta starea acestora prin digitalRead
- In functie de placa Arduino folosita, numarul intreruperilor externe disponibile este variabil:

Board	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
Leonardo	3	2	0	1	7	

- Pentru utilizarea unei intreruperi, acestei intreruperi trebuie sa i se ataseze o procedura de tratare (Interrupt Service Routine - ISR). In acest scop, se foloseste functia **attachInterrupt()**, cu sintaxa:

  attachInterrupt(interrupt, ISR, mode)

  interrupt - numarul intreruperii externe (0, 1, 2, …)

  ISR – numele procedurii de tratare a intreruperii (o procedura din program)

  mode – modul de declansare:

  LOW – declansare pe nivel ‘0’

  CHANGE – declansare cand se schimba nivelul pinului

  RISING – declansare pe front crescator

  FALLING – declansare pe front descrescator

Tratarea intreruperilor externe

- Dezactivarea tratarii unei intreruperi se face prin apelul functiei **detachInterrupt()**, cu sintaxa:

  detachInterrupt(interrupt)
  interrupt - numarul intreruperii

- Daca se doreste dezactivarea temporara a tuturor intreruperilor, se apeleaza functia **noInterrupts()**, fara parametri. Pentru re-activarea intreruperilor, se apeleaza functia **interrupts()**.
- Intreruperile sunt active implicit! Dezactivarea lor trebuie facuta doar pentru perioade scurte de timp, in caz contrar alte functii arduino pot fi afectate.

Tratarea intreruperilor externe

- **Exemplu:** masurarea latimii pulsurilor unui semnal (de exemplu, semnalul receptionat de un senzor IR de la o telecomanda, unde latimea pulsului face diferenta dintre un ‘0’ si un ‘1’)

```
const int irReceiverPin = 2;          // Pinul 2, unde se gaseste intreruperea externa 0
const int numberOfEntries = 64;       // Numarul de tranzitii de analizat

volatile unsigned long microseconds; //Variabila care tine numarul de microsecunde trecute de la pornirea programului
volatile byte index = 0;              //Pozitia in sirul de tranzitii
volatile unsigned long results[numberOfEntries];  //Sirul cu duratele tranzitiilor analizate (rezultat)

void setup()
{
  pinMode(irReceiverPin, INPUT);          //Se declara pinul intreruperii ca intrare
  Serial.begin(9600);                     //Se activeaza comunicarea prin USB, pentru afisare date
  attachInterrupt(0, analyze, CHANGE);    //Se ataseaza ISR-ul analyze la intreruperea 0, declansata la schimbarea
  results[0]=0;                           semnalului
}

void loop()
{
  if(index >= numberOfEntries)            // Se verifica daca numarul de tranzitii maxim a fost analizat
  {
    Serial.println("Durations in Microseconds are:") ; // Daca da, se afiseaza toate intervalele masurate
    for( byte i=0; i < numberOfEntries; i++)
    {
      Serial.println(results[i]);
    }
    index = 0; // Dupa afisare, se re-initializeaza contorul de tranzitii la 0, si procesul se reia
  }
  delay(1000);
}
```

Tratarea intreruperilor externe

- **Exemplu:** masurarea latimii pulsurilor unui semnal (de exemplu, semnalul receptionat de un senzor IR de la o telecomanda, unde latimea pulsului face diferenta dintre un ‘0’ si un ‘1’)
- Continuare:

```
void analyze()            // Procedura de tratare a intreruperii
{
  if(index < numberOfEntries  )     // Daca nu s-a ajuns la capatul sirului
  {
    if(index > 0)                   // Dar nu este prima tranzitie detectata
    {
      results[index] = micros() - microseconds;   // Se masoara timpul trecut de la ultima tranzitie
    }
    index = index + 1;              // Se incrementeaza indexul
  }
  microseconds = micros();          // Se retine timpul curent, pentru a fi etalon pentru tranzitia urmatoare
}
```

- Functia **micros()** returneaza numarul de microsecunde de la pornirea programului.
- Pentru masurarea unor intervale de timp mai mari, cu precizie mai mica, se poate folosi functia **millis()**, care returneaza numarul de milisecunde de la pornirea programului.

Tratarea intreruperilor externe

Atentie:

- Toate variabilele care se pot modifica intr-o functie de tip ISR trebuie sa fie declarate ca “**volatile**”. Astfel, compilatorul va sti ca aceste variabile se pot modifica in orice moment, si nu va optimiza codul prin atasarea acestora la registri, ci le va stoca intotdeauna in RAM
- Doar o procedura ISR poate rula la un moment dat, celelalte intreruperi fiind dezactivate in acest timp
- Deoarece **delay()** si **millis()** folosesc la randul lor intreruperi, ele nu vor functiona in timp ce o procedura de tip ISR este in executie.
- Pentru intarzieri scurte intr-o procedura ISR, se poate folosi functia **delayMicroseconds()**, care nu foloseste intreruperi.
- Nu se recomanda scrierea datelor prin interfata seriala intr-o procedura ISR

Confuzia numarului de intrerupere

- Numarul specificat ca parametru nu este acelasi cu numarul intreruperii externe al microcontrollerului AVR:

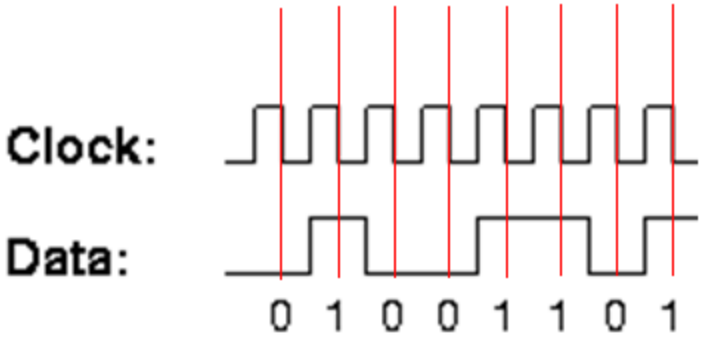
attachInterrupt	Name	Pin on chip	Pin on board
		(TQFP)	
0	INT4	6	D2
1	INT5	7	D3
2	INT0	43	D21
3	INT1	44	D20
4	INT2	45	D19
5	INT3	46	D18

- Soluţia: utilizarea funcţiei digitalPinToInterrupt(pin)
  - Exemplu:

  attachInterrupt(digitalPinToInterrupt(21) , isr, FALLING) va ataşa întreruperii Arduino **2**, corespunzătoare întreruperii externe **INT0** de la ATMega2560, accesibilă prin pinul digital 21, procedura **isr**, care se va apela când pe pin se va efectua o tranziţie din HIGH în LOW.
- Dacă un pin digital nu este în relaţie cu o întrerupere, funcţia digitalPinToInterrupt va returna valoarea -1 .

Exerciţii

- Afişaţi, prin interfaţa serială, numărul pinilor care sunt în relaţie cu întreruperi externe.
- Scrieţi un program capabil să preia date seriale, sincronizate pe un semnal de ceas, ca în figura de mai jos:



- Modificaţi programul pentru a utiliza un semnal suplimentar, care marchează începutul şi sfârşitul octetului: