Aplicații cu Microcontroller

Şcoala de vară 2015

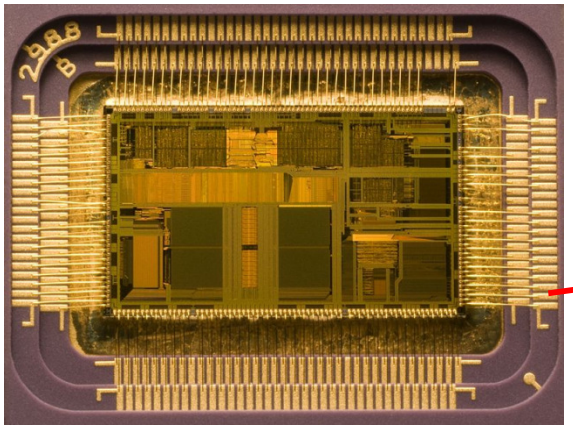
Radu Dănescu

Ce este un microprocesor

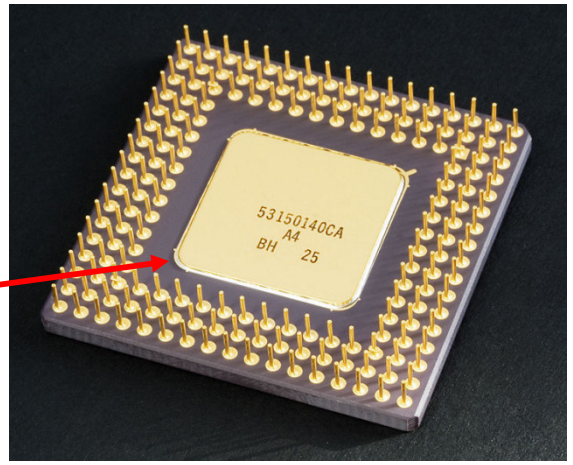
Un **microprocesor** incorporeaza toate sau majoritatea functiilor unei **unitati centrale de procesare** intr-un singur circuit integrat.

O unitate centrala de procesare **(Central Processing Unit, CPU)** este o masina logica ce poate executa programe de calculator.

Functia fundamentala a oricarui CPU, indiferent de forma fizica pe care o are, este sa execute o **secventa de instructiuni (programul)**, stocate intr-o memorie. Executia instructiunilor se face de obicei in patru pasi: citire instructiune (**fetch**), decodificare (**decode**), executie (**execute**) si scriere rezultate (**write back**).

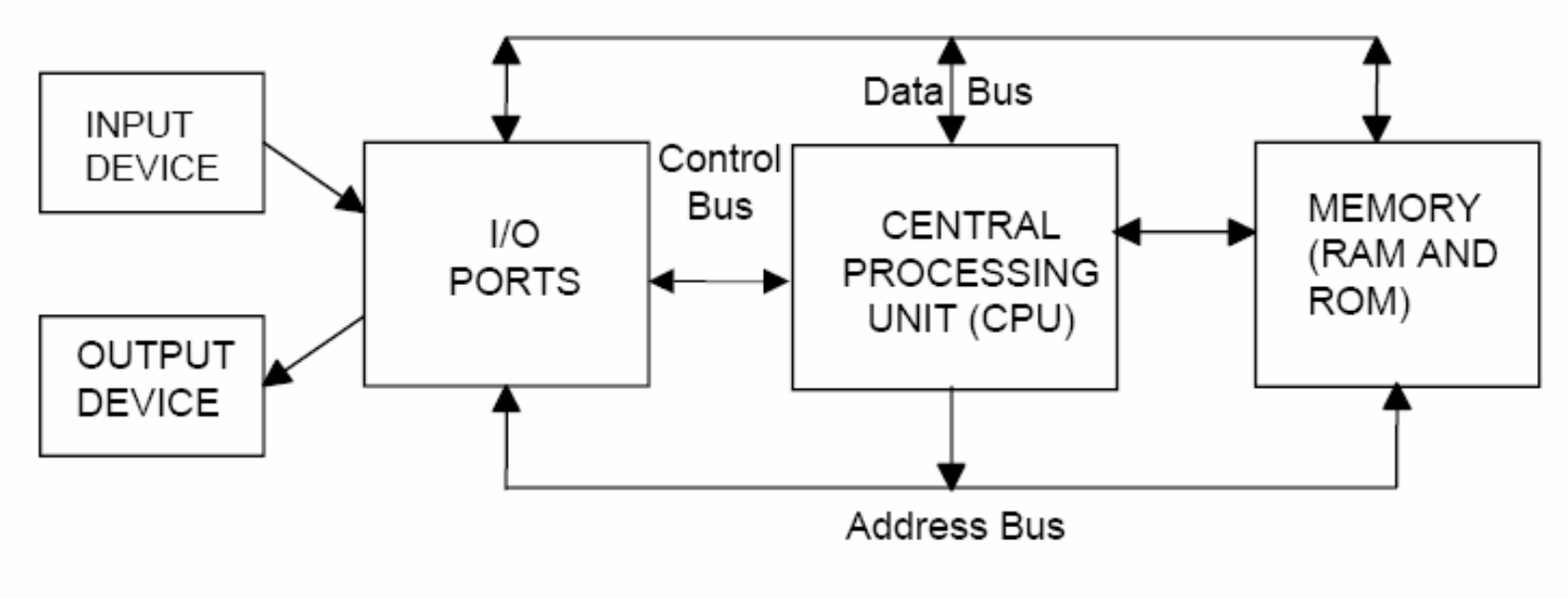


Intel 80486DX2 , interior



Intel 80486DX2 – vedere exterioara

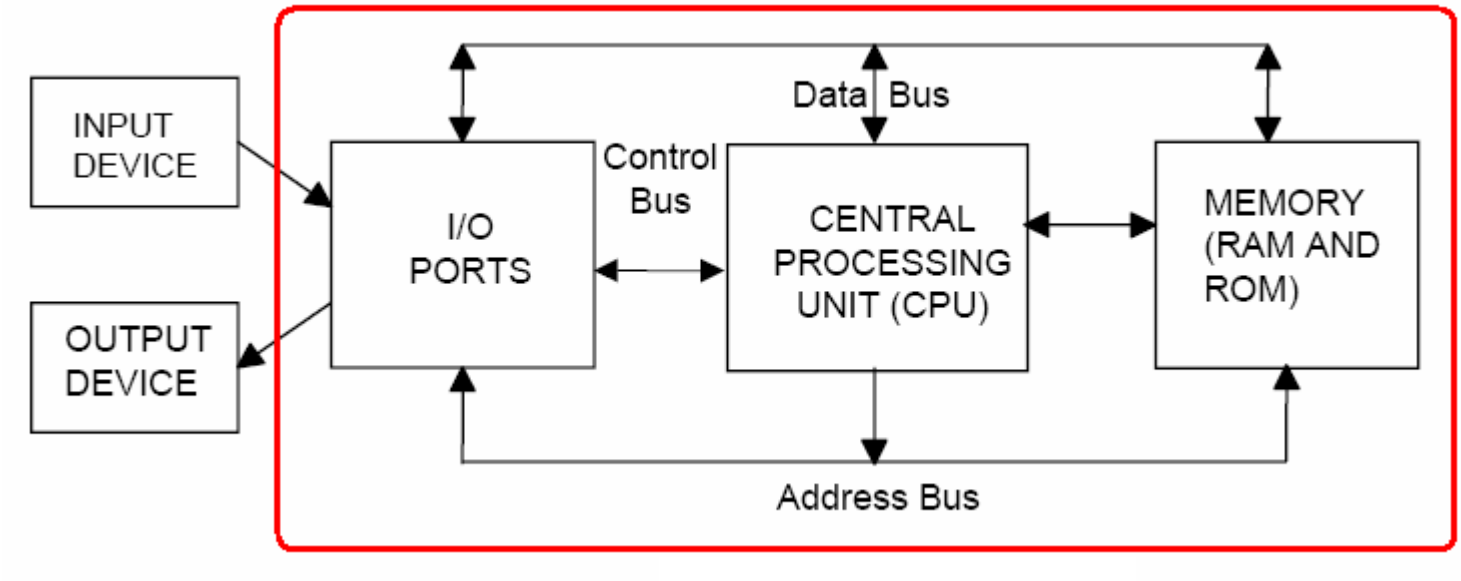
Sisteme cu microprocesor



Dispozitive esentiale: CPU, Memorie, I/O

Dispozitive aditionale: Controller intreruperi, DMA, coprocesor, etc

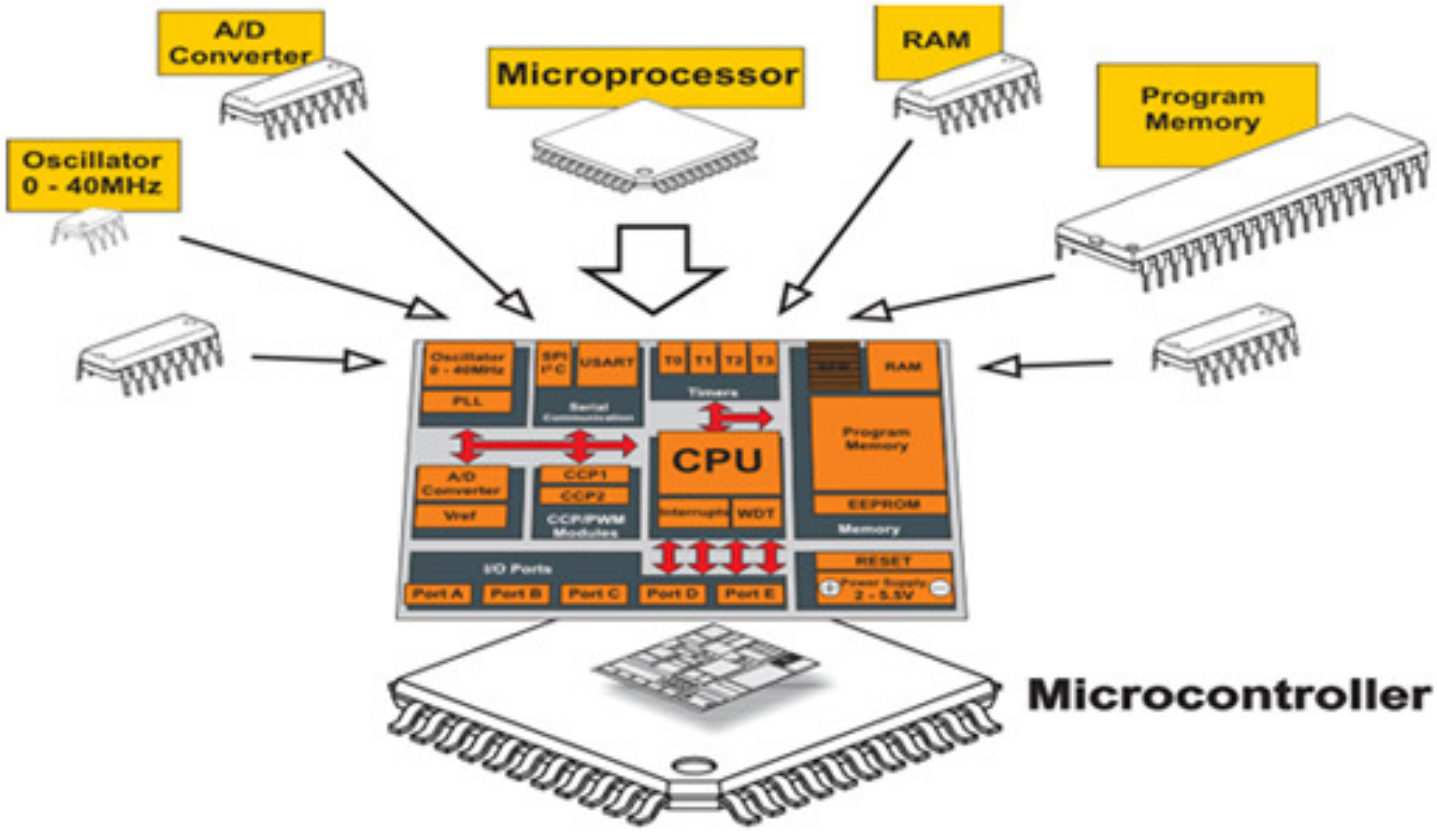
Microcontroller (MCU)



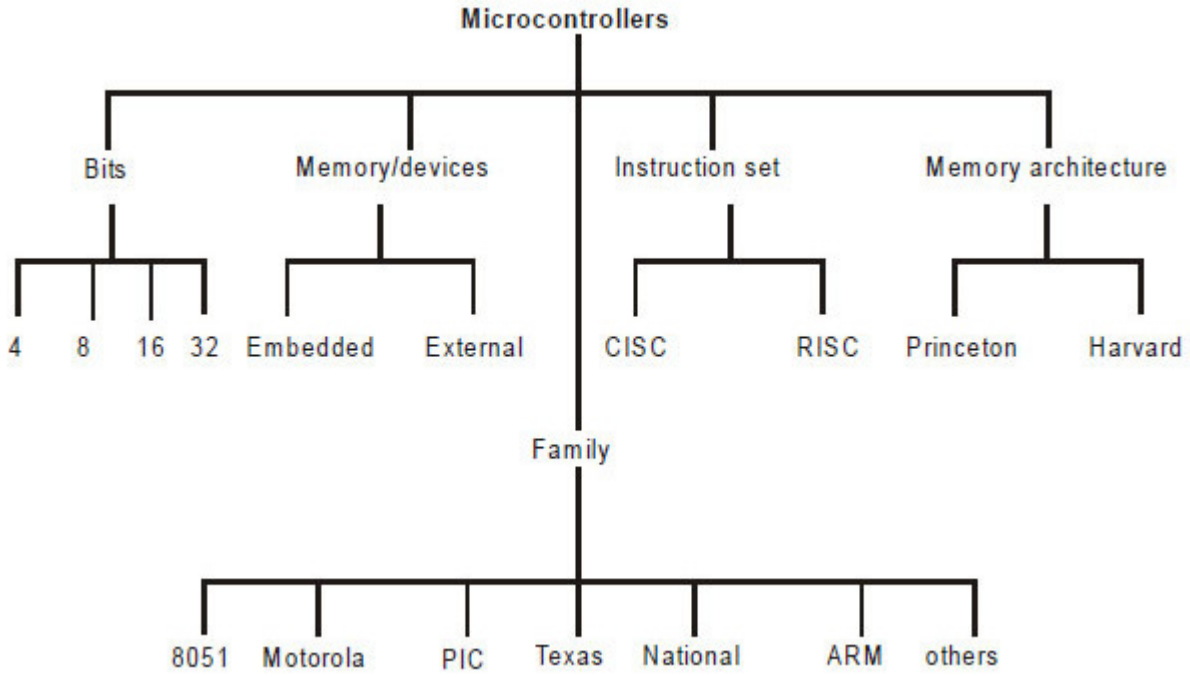
Multiple componente ale unui sistem cu microprocesor sunt incluse in acelasi circuit integrat - Microcontroler

- Memorie RAM si ROM (Flash), pentru program si date
- Unele dispozitive periferice (Timer, Numarator, Controllere pentru comunicatii seriale / paralele, etc)

Microcontroller (MCU)

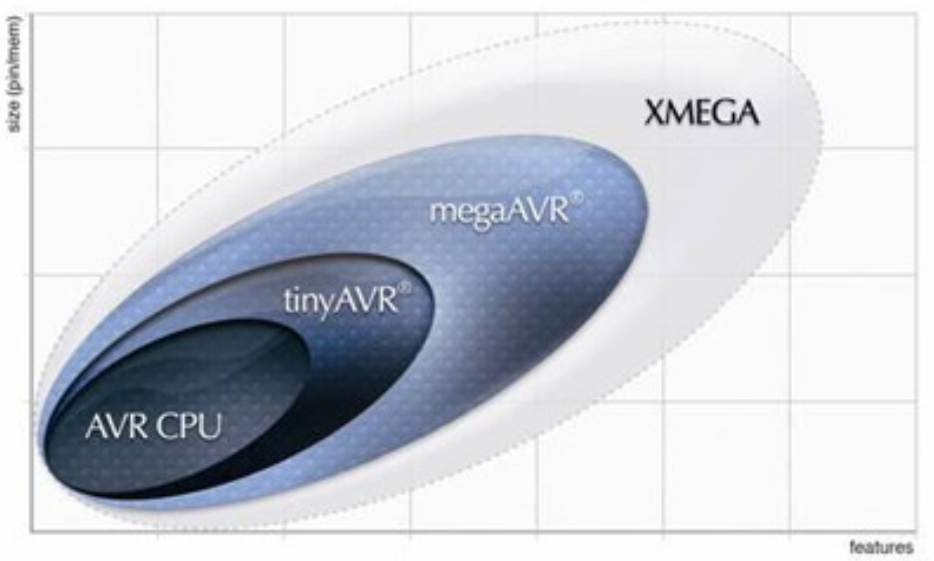


Clasificarea microcontrollerelor



Familia de microcontrollere Atmel AVR 8 biti

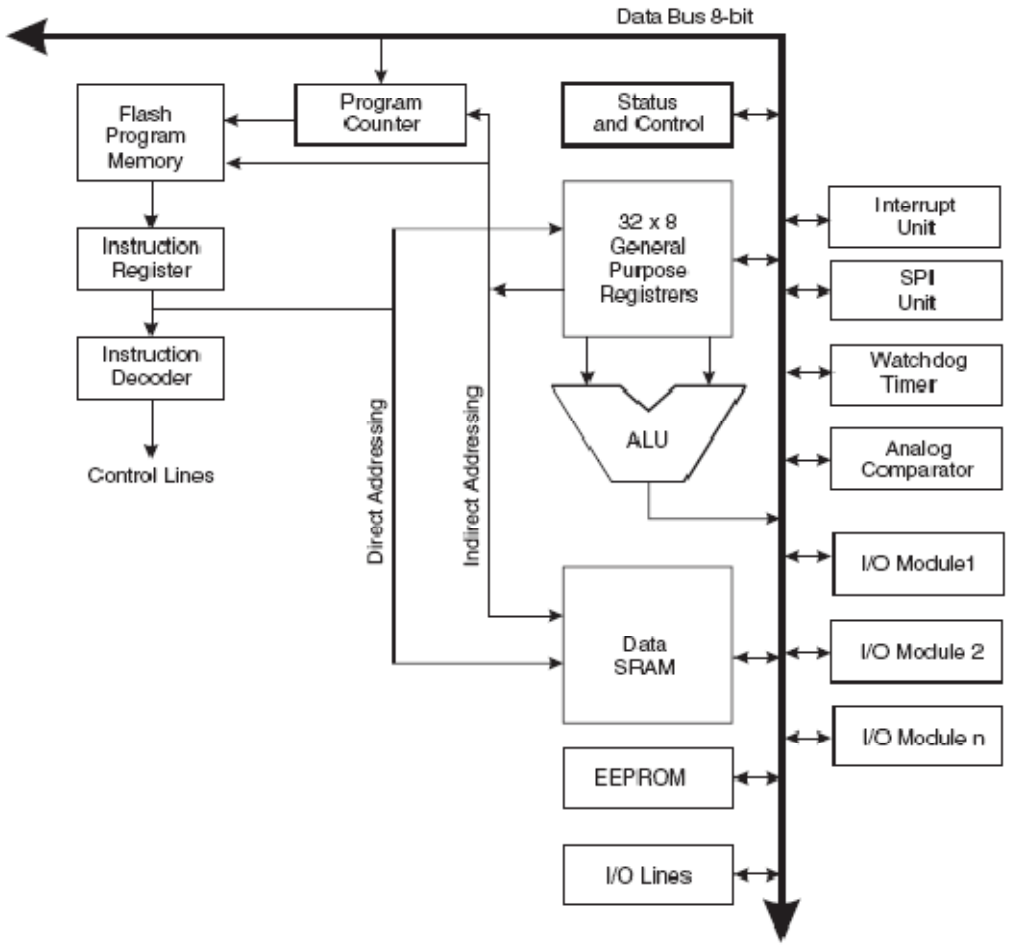
- Arhitectura RISC
- Executie 1 instructiune / ciclu
- 32 registri de uz general
- Arhitectura Harvard
- Tensiune de alimentare 1.8 - 5.5V
- Frecventa controlata software
- Mare densitate a codului
- Gama larga de dispozitive
- Numar de pini variat
- Compatibilitatea integrala a codului
- Familii compatibile intre pini si capabilitati
- Un singur set de unelte de dezvoltare



tinyAVR
1–8 kB memorie program
megaAVR
4–256 kB memorie program
Set extins de instructiuni (inmultire)
XMEGA
16–384 kB memorie program
Extra: DMA, suport pentru criptografie
AVR specific pentru aplicatii
megaAVR cu interfete particulare:
LCD, USB, CAN etc.

Arhitectura generală a unui microcontroller AVR

- Masina RISC (Load-store cu doua adrese)
- Arhitectura Harvard modificata – exista instructiuni speciale care pot citi datele din memoria program
- Pipeline pe doua nivele: citire instructiune (Fetch) şi execuție

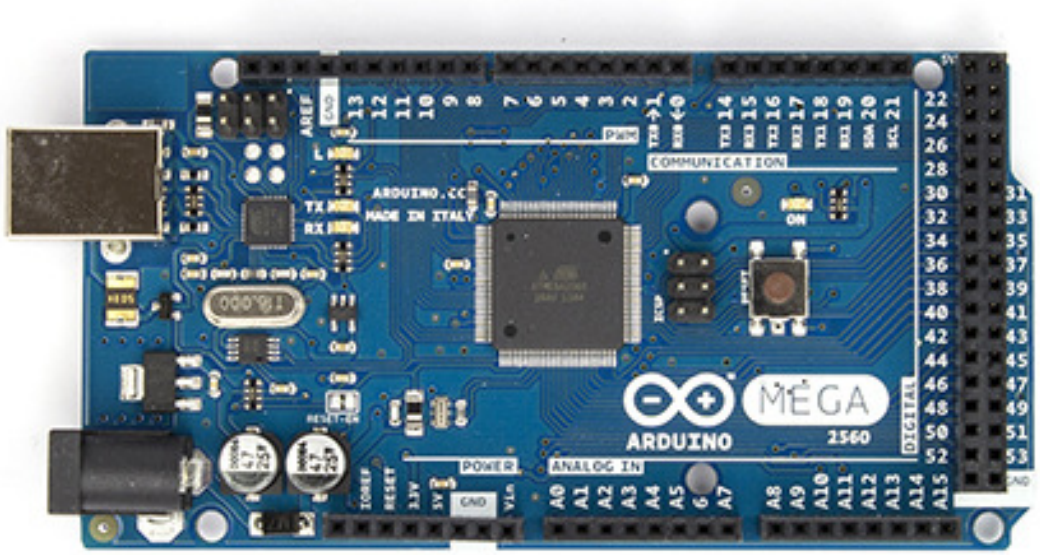


Arduino

- Placi cu microcontroller (in special AVR), si unelte de dezvoltare software open source
- Ascunde detaliile specifice diferitelor microcontrollere, folosind o abordare unificata
- Este disponibila o larga multime de placi, shield-uri si accesorii
- O cantitate impresionanta de documentatie gratuita sau contra cost
- O cantitate impresionanta de exemple pentru orice problema

Site web: **www.arduino.cc**
Distribuitori in Romania: **www.robofun.ro**

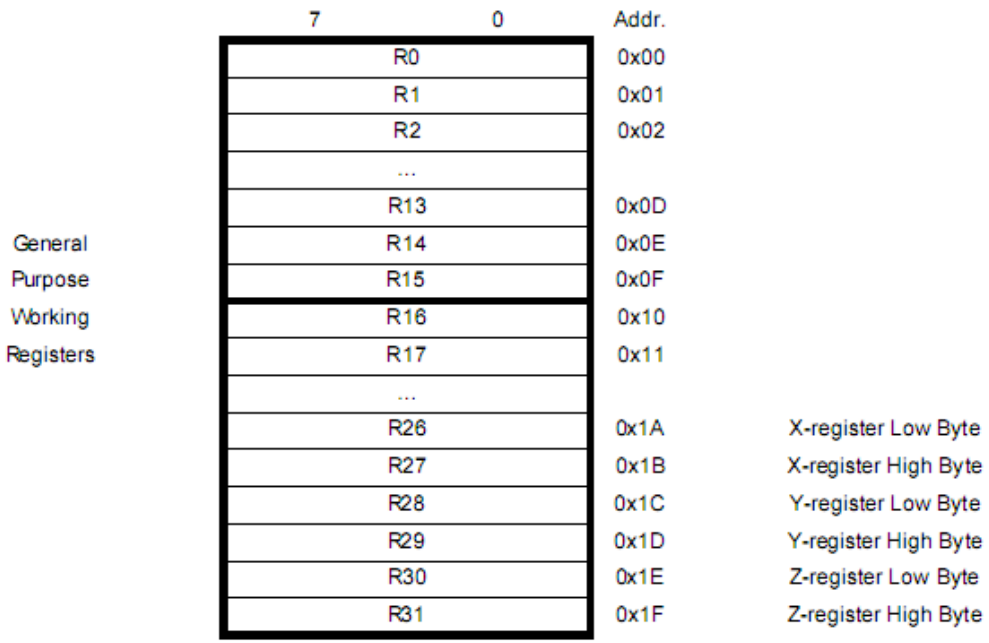
Arduino Mega 2560

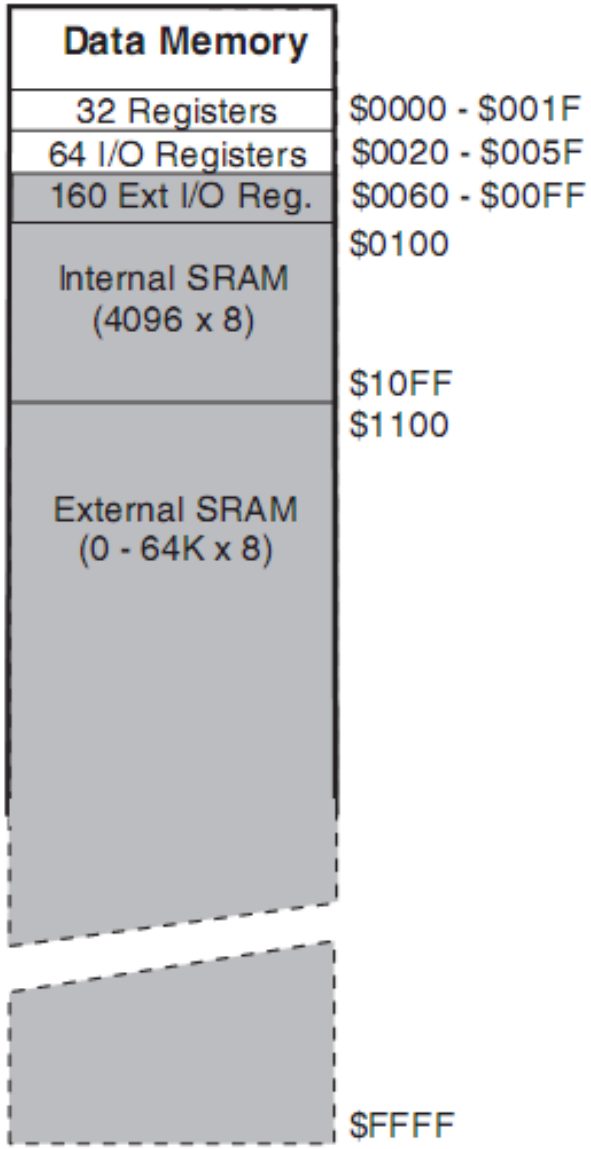


- Bazata pe microcontrollerul ATMega2560, pe 8 biti
- 54 pini de I/O digitali
- 16 pini de intrare pentru semnale analogice
- 4 porturi de comunicare seriala UART
- Frecventa procesorului: 16 MHz
- Alimentare si programare prin cablu USB

Harta memoriei AVR

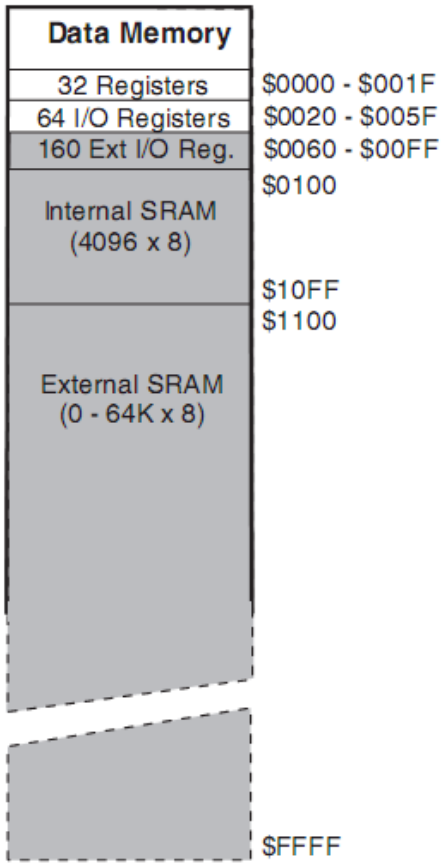
- Primele 32 de adrese – blocul de registri
- 64 de adrese – registri I/O accesabili prin instructiuni speciale
- 160 adrese – spatiu I/O extins, accesabil prin instructiuni standard de acces la memorie
- Următoarele adrese: RAM intern sau extern

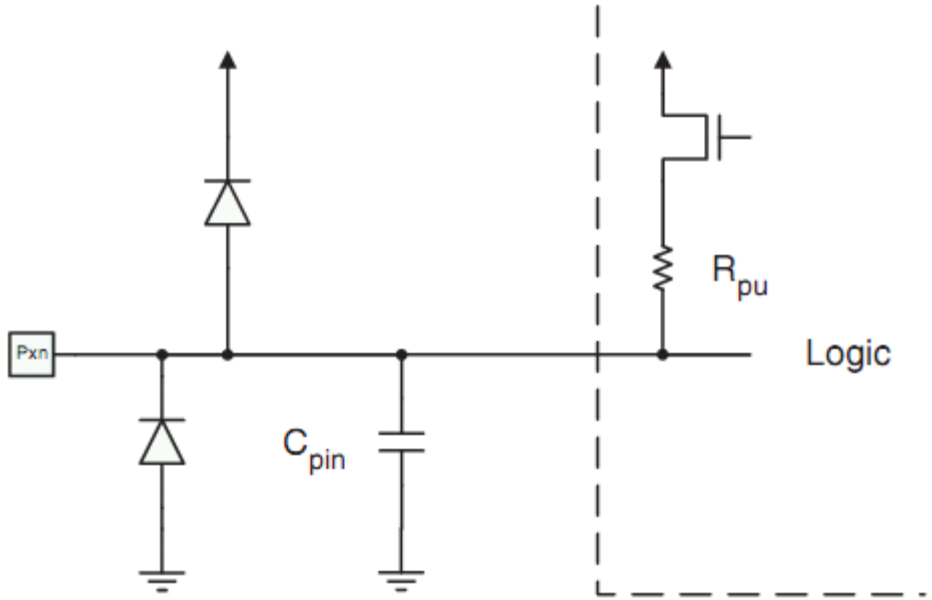




Intrare / Iesire la AVR

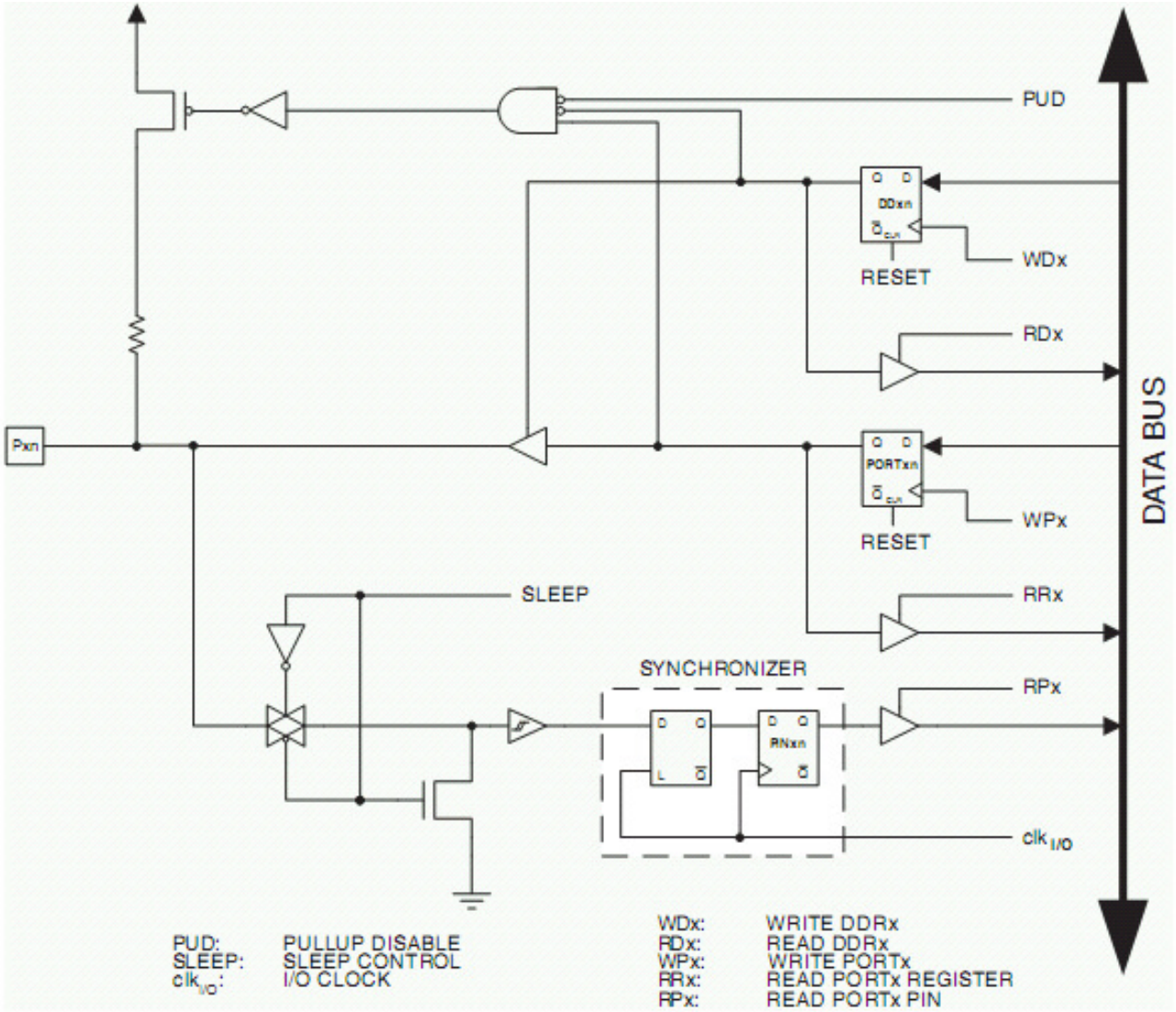
- Porturile de intrare/iesire: PORTA ... PORTG
- PORTA... PORTE pot fi accesate prin instructiuni speciale **in**, **out**
- PORTF, PORTG accesibile doar prin **ld**, **st** – spatiul de adrese I/O extins
- Fiecare bit din fiecare port poate fi configurat ca intrare sau ca iesire, prin scrierea registrului de directie DDRx
- Scrierea portului se face prin registrul PORTx
- Citirea starii pinilor se face prin PINx
- Diode de protectie, impotriva electricitatii statice
- Rezistenta “pull up”, care poate fi activata/ dezactivata prin logica





Intrare / Iesire

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



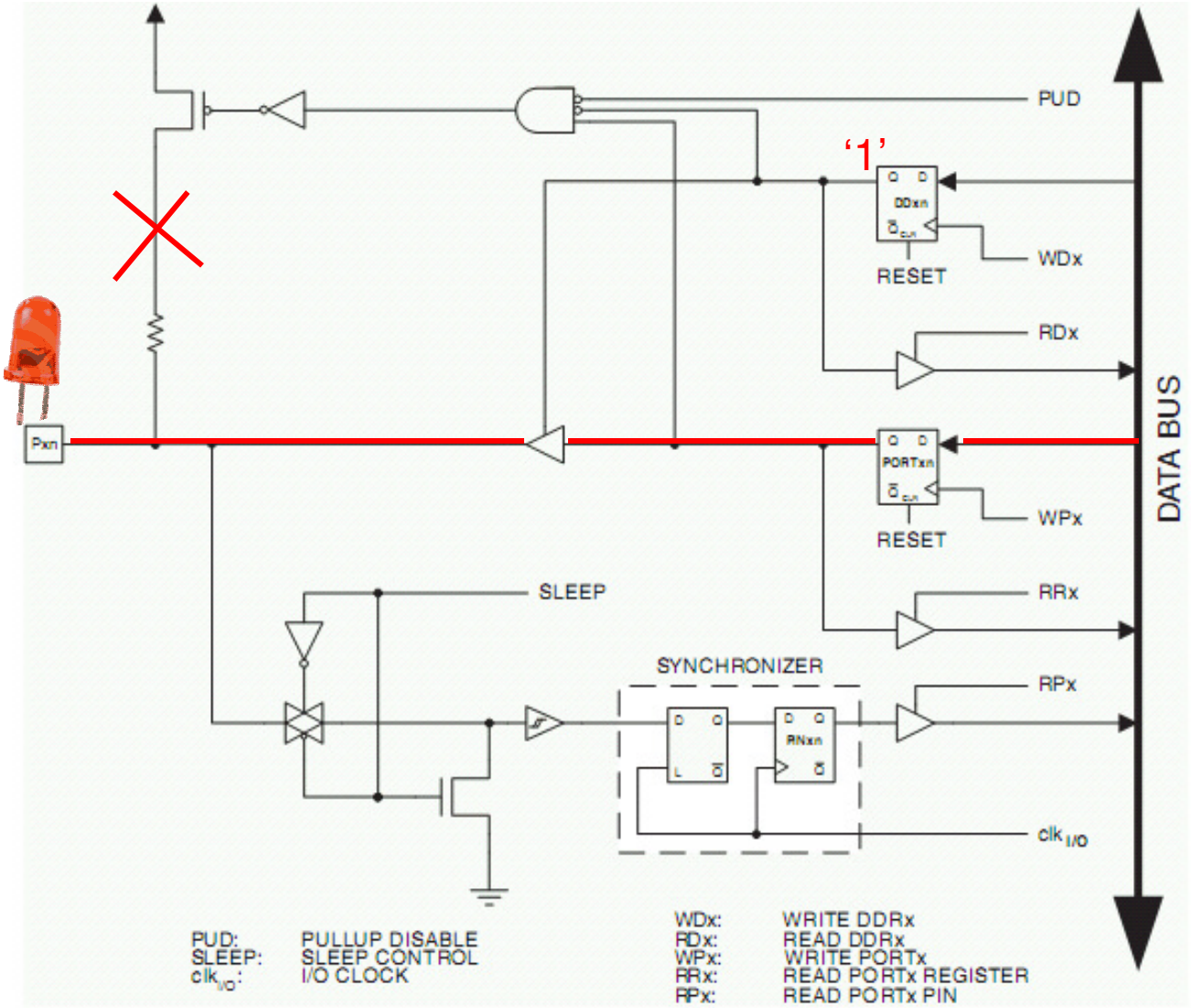
Control Directie

Datele ce vor fi trimise la iesire

Datele citite de pe intrare

Intrare / Iesire

- Configuratia pentru iesire

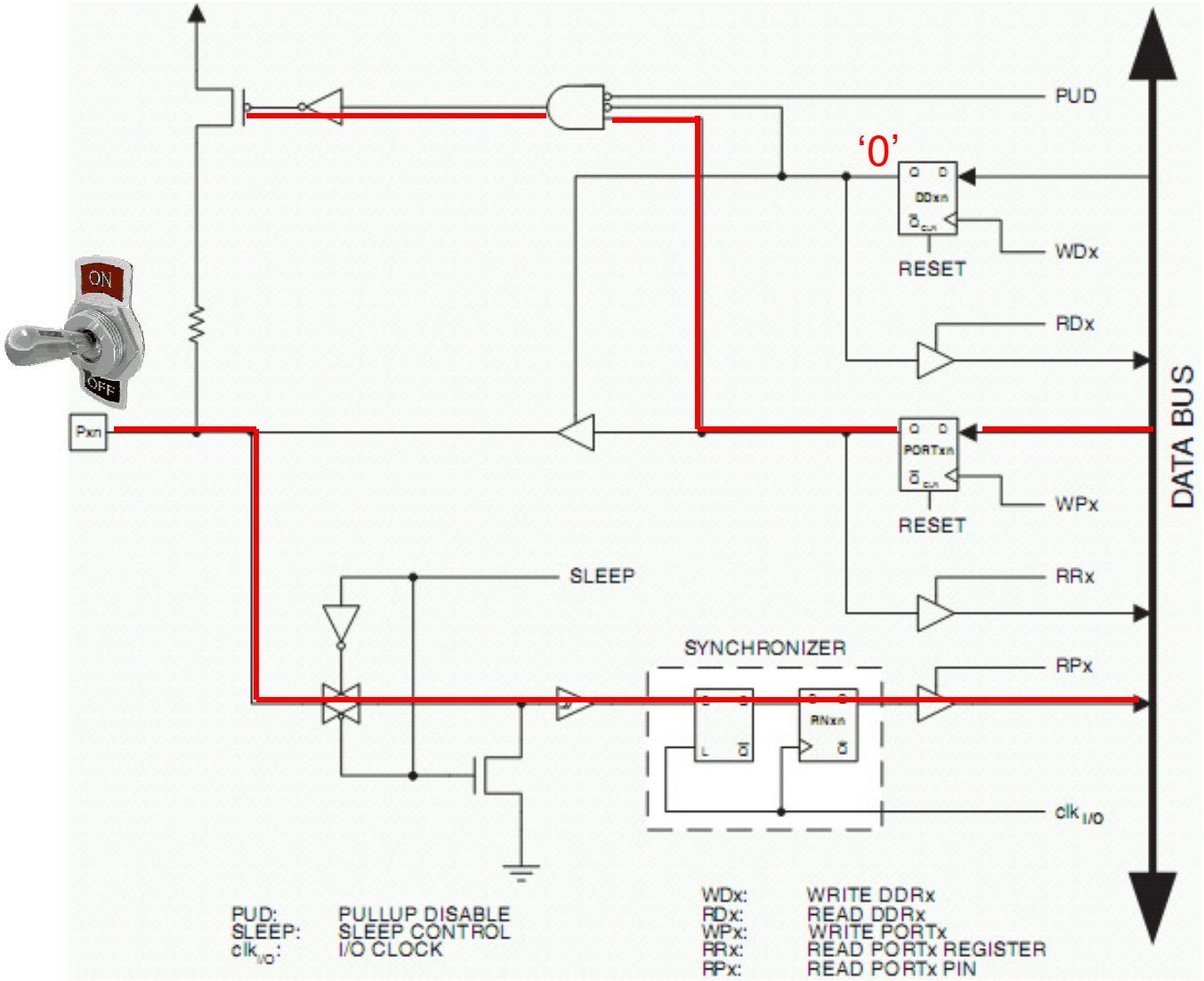


Directie = 1

Datele scrise in PORTx sunt trimise la iesire

Intrare / Iesire

- Configuratia pentru intrare

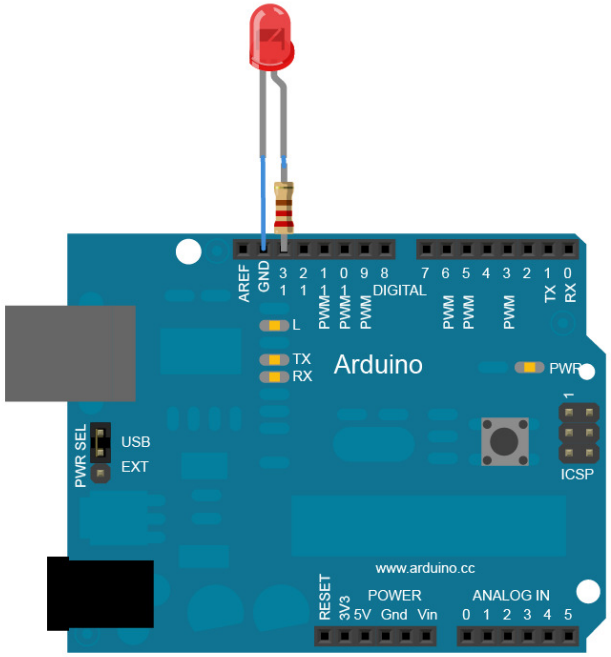


Directie = 0

'1' scris in PORTx activeaza rezistenta pull up

Datele devin disponibile ca PINx

Intrare / Iesire cu Arduino



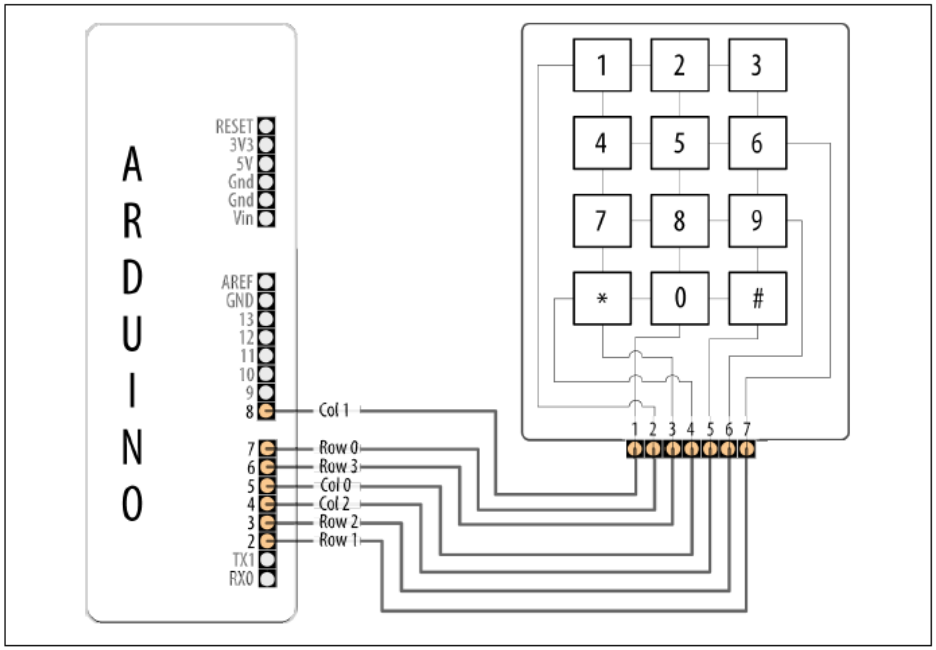
- Aprindere intermitenta a unui LED, conectat la un pin digital de iesire (digital output)

Intrare / Iesire cu Arduino

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Intrare / Iesire cu 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 cu 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
  }
}
```

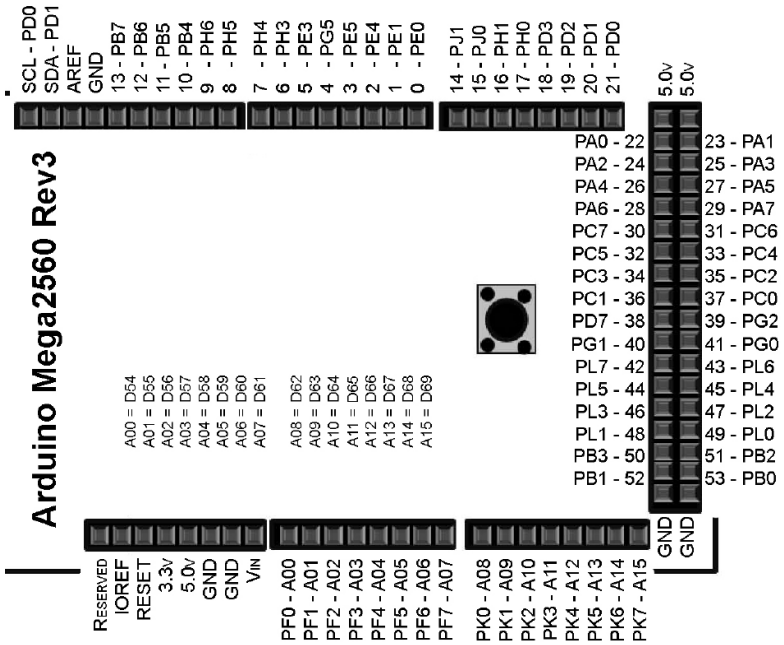
Intrare / Iesire cu Arduino

- **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 cu 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 cu 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 cu Arduino

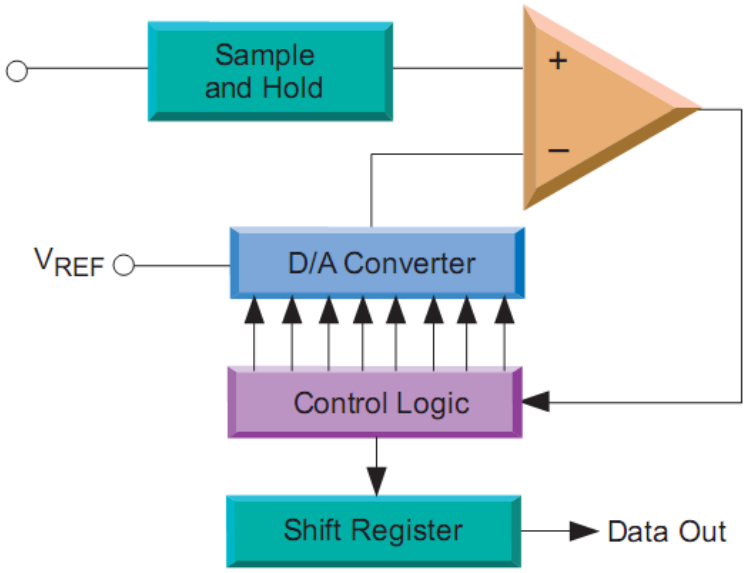
- **Exemplu:** se leaga 8 led-uri la pinii 22…29 ai Arduino Mega (conectati la PortA). Se doreste aprinderea alternativa a led-urilor pare si impare, cu o intarziere de 1 secunda intre comutatii
- **Cod sursa, abordarea folosind portul A al ATMega2560**:

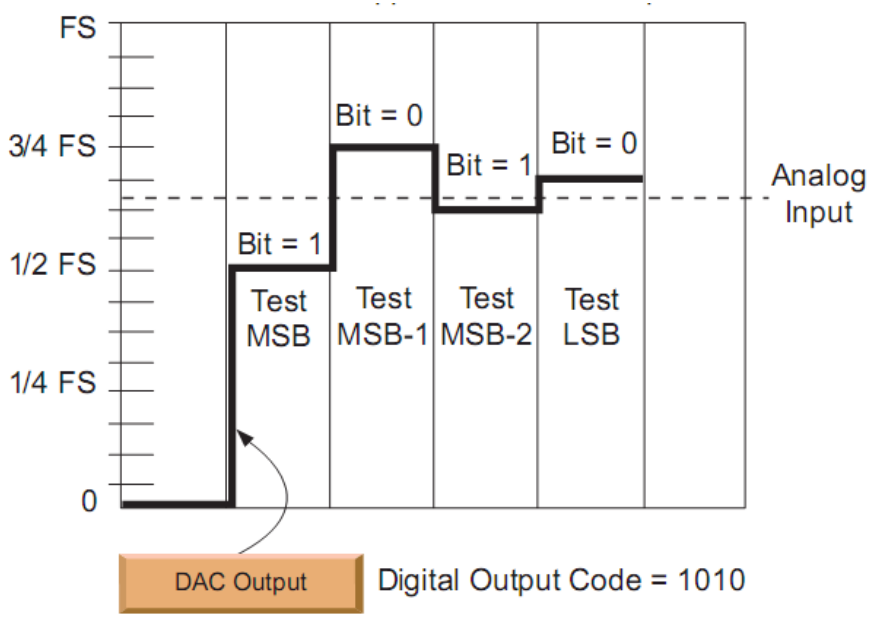
```
void setup()
{
    DDRA = B11111111;                    // toti pinii atasati portului A sunt configurati ca iesire
}

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)
}
```

Citirea semnalelor analogice

Semnal analogic: un nivel de tensiune
Convertor analog / digital: transformă nivelul de tensiune într-o valoare numerică

- Se fac comparatii succesive cu diviziuni ale unei tensiuni de referintă
- Se foloseşte un comparator analogic pentru a decide pentru fiecare bit dacă acesta este unu (tensiunea e mai mare) sau zero (tensiunea e mai mică)
- Conversia nu este instantanee: are nevoie de cel puțin atâtea perioade de ceas câți biți are rezultatul





Citirea semnalelor analogice

Conversia semnalului analogic in semnal digital

- Transformarea unui semnal analogic (tensiune) de intrare intr-o valoare digitala pe **10 biti (0 … 1023)**
- Intrarea poate fi “single end” – tensiunea de intrare se calculeaza intre pinul de intrare si GND, sau diferentiala – diferenta de tensiune intre doi pini de intrare
- Pentru intrarea diferentiala, se poate utiliza amplificare (Gain)

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

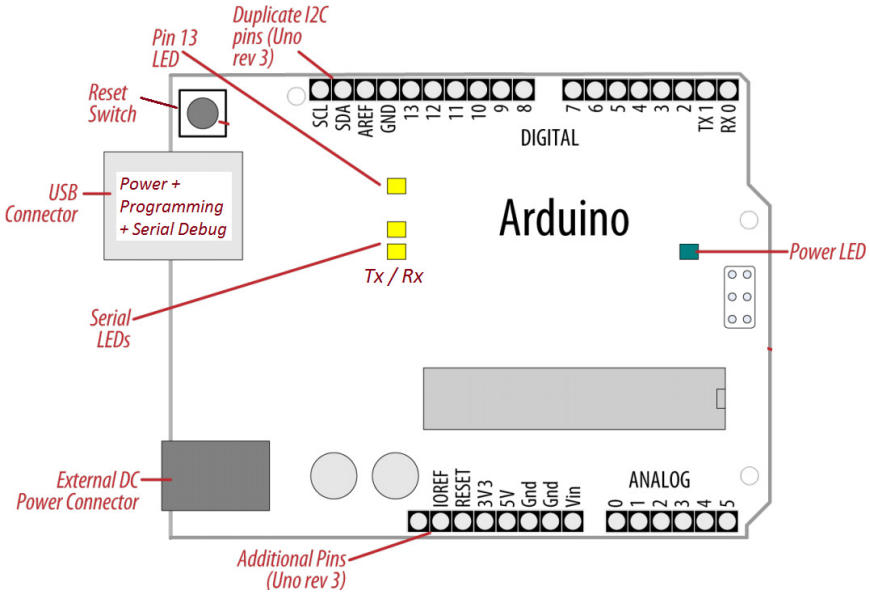
ADC: 0…1023

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$

ADC: -512…511

- V_{IN} – tensiunea ce trebuie măsurată.
- V_{REF} – tensiunea de referință. Poate fi internă, generată de microcontroller, sau externă. Se selectează prin program.

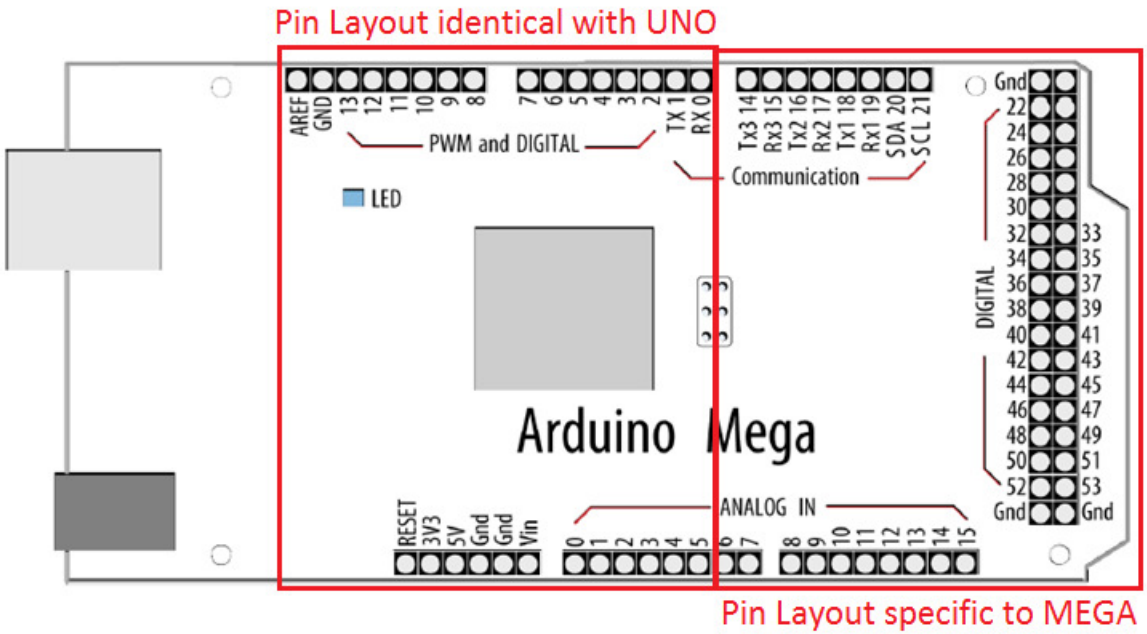
Citirea semnalelor analogice cu Arduino



Arduino UNO: A0 .. A5

Arduino MEGA: A0 .. A15

- Pinii analogici sunt intrari pentru convertorul ADC al microcontrollerului.
- ADC are rezolutia de 10 biti, returnand valori intre 0 si 1023



Alti pini

AREF (intrare) – tensiune de referinta externa pentru ADC

IOREF (iesire) – tensiune de referinta pentru shield-uri

Citirea semnalelor analogice cu Arduino

- **Functia principala a pinilor analogici**: **citirea semnalelor analogice**
- Pinii analogici au si functia de pin digital de uz general, ca si pinii digitali. Exemplu:

```
pinMode(A0, OUTPUT);
digitalWrite(A0, HIGH);
```

- Pinii analogici au de asemenea rezistente pull up resistors, care functioneaza in acelasi fel ca rezistentele pinilor digitali. Ele sunt activate scriind HIGH pe pinul configurat ca intrare.

```
digitalWrite(A0, HIGH); // activare pullup la A0 configurat ca input.
```

Activarea unei rezistente pull up va influenta valorile citite cu analogRead() !!!

Functii:

analogRead(pin) – citeste o valoare de pe un pin analogic

analogReference(type) - configureaza tensiunea de referinta care va fi folosita pentru intrarea analogica (i.e. valoarea maxima a tensiunii de intrare masurabila pe pin-ul analogic)

Citirea semnalelor analogice cu Arduino

analogReference(type) – configureaza tensiunea de referinta care va fi folosita pentru intrarea analogica.

type - care referinta este folosita:

- DEFAULT: tensiunea referinta implicita, de 5 V (pentru UNO & MEGA)
- INTERNAL: tensiune interna de referinta, 1.1 V la UNO (*nu exista la Arduino Mega*)
- INTERNAL1V1: tensiune interna de referinta 1.1V (doar *Arduino Mega*)
- INTERNAL2V56: tensiune interna 2.56V (doar *Arduino Mega*)
- EXTERNAL: tensiune de referinta externa, aplicata la pinul AREF (**0 … 5V**).

Dupa schimbarea tensiunii de referinta, prima citire cu analogRead() poate fi eronata !!!

Nu folositi o tensiune de referinta externa negativa (<0V) sau mai mare de 5V pe pinul AREF! Daca folositi o tensiune externa de referinta, configurati referinta ca externa apeland analogReference() inainte de a apela functia analogRead(). In caz contrar, veti pune in contact tensiunea de referinta interna, generata in mod activ, cu tensiunea externa, putand cauza scurtcircuit si distrugerea microcontrollerului. !!!

Citirea semnalelor analogice cu Arduino

int *digital_value* **analogRead**(*pin*) – citeste o valoare de pe pinul analogic specificat

- O valoare analogica intre 0 .. RANGE va produce un numar *digital_value* intre 0 si 1023.
- Rezolutia de masurare este deci: RANGE volti / 1024 unitati.
- Pentru referinta DEFAULT (5V) rezolutia devine:

  resolutionADC = .0049 volti (4.9 mV) / unitate.
- Pentru a converti valoarea citita *digital_value* la tensiunea analogica:

  Voltage = ***resolutionADC*** * ***digital_value***
- Dureaza aproximativ 100 microsecunde (0.0001 s) pentru a citi o intrare analogica, astfel incat rata maxima de citire este 10000 valori pe secunda.

Daca pinul analogic nu este conectat la nimic, valoarea returnata de analogRead() va fluctua in functie de mai multi factori (e.g. ce tensiuni sunt pe ceilalti pini analogici, apropierea mainii de placa…) !!!

Citirea semnalelor analogice cu Arduino

Senzor de temperatura folosind LM50 http://www.ti.com/lit/ds/symlink/lm50.pdf

Caracteristici:

- Iesire liniara +10.0 mV/°C = 0.01V/°C
- Domeniu de temperaturi –40°C … +125°C
- Deplasament constant +500 mV pentru citirea temperaturilor negative

Circuitul LM50 este inclus in senzorul de temperatura Brick:
http://www.robofun.ro/senzori/vreme/senzor-temperatura-brick

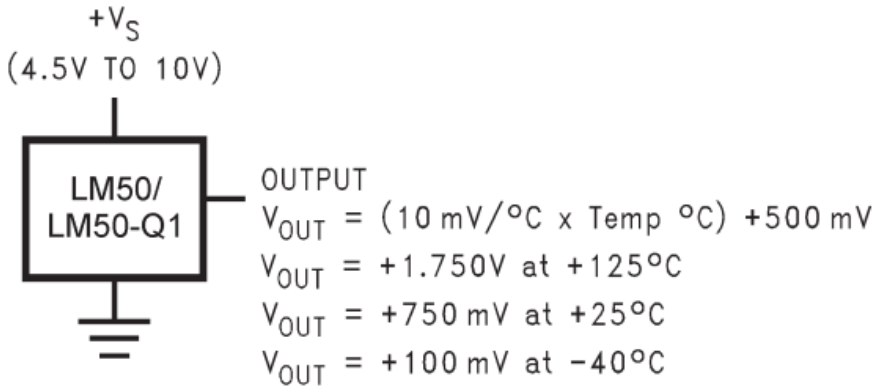
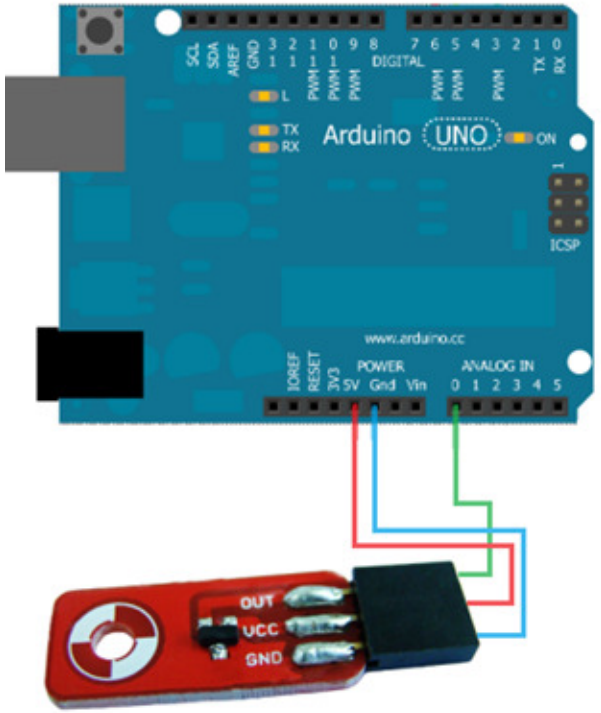


Figure 2. Full-Range Centigrade Temperature Sensor Application (−40°C to +125°C)



Citirea semnalelor analogice cu Arduino

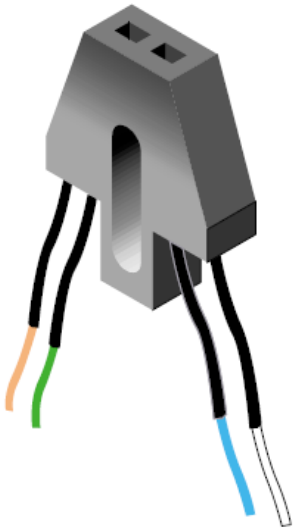
Examplu – Citire temperatura de la senzor, face media a 10 citiri consecutive, si trimite catre PC

```
float resolutionADC  = .0049 ;       // rezolutia implicita (pentru referinta 5V) = 0.049 [V] / unitate
float resolutionSensor  = .01 ;      // rezolutie senzor = 0.01V/°C

void setup()
{ Serial.begin(9600);
}
void loop(){
  Serial.print("Temp [C]: ");
  float temp = readTempInCelsius(10, 0);  // citeste temperatura de 10 ori, face media
  Serial.println(temp);                   // afisare
  delay(200);
}
float readTempInCelsius(int count, int pin) {
// citeste temperatura de count ori de pe pinul analogic pin
  float sumTemp = 0;
  for (int i =0; i < count; i++) {
      int reading = analogRead(pin);
      float voltage = reading * resolutionADC;
      float tempCelsius = (voltage - 0.5) / resolutionSensor ;  // scade deplasament, converteste in grade C
      sumTemp = sumTemp + tempCelsius;                          // suma temperaturilor
      }
  return sumTemp / (float)count;                                // media returnata
}
```

Perceptia Mediului

Fairchild QRB 1134 - Senzor IR pentru detectia reflectivitatii suprafetelor

- Pereche formata din dioda infrarosu si fototranzistor in acelasi spectru
- Tranzistorul raspunde prin variatia curentului de colector in functie de lumina incidenta
- Raspunsul tranzistorului depinde de reflectivitatea suprafetei si de distanta la aceasta



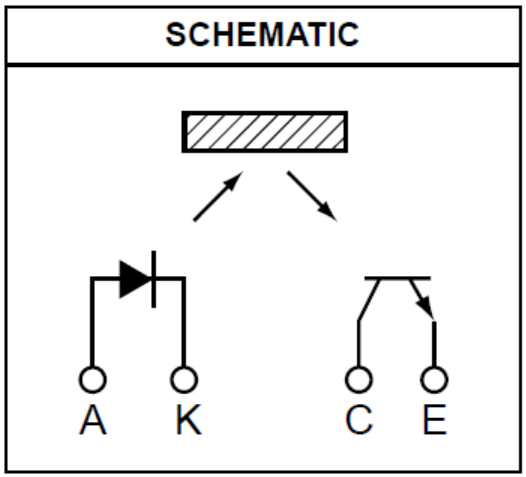
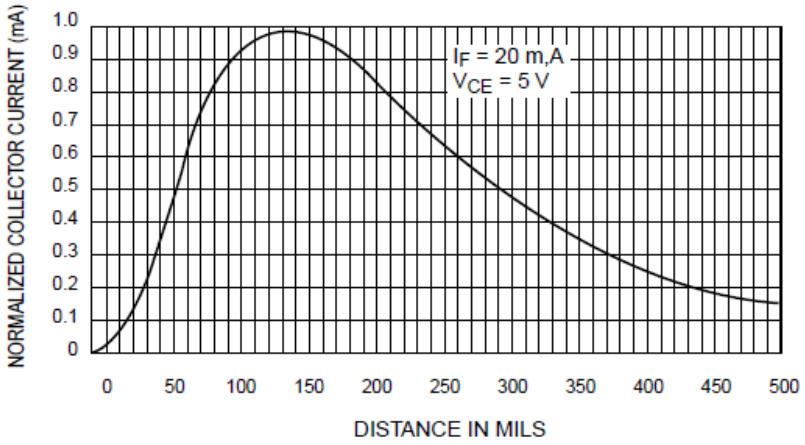


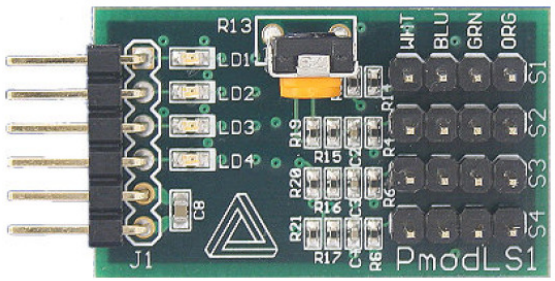
Fig. 5 Normalized Collector Current vs. Distance

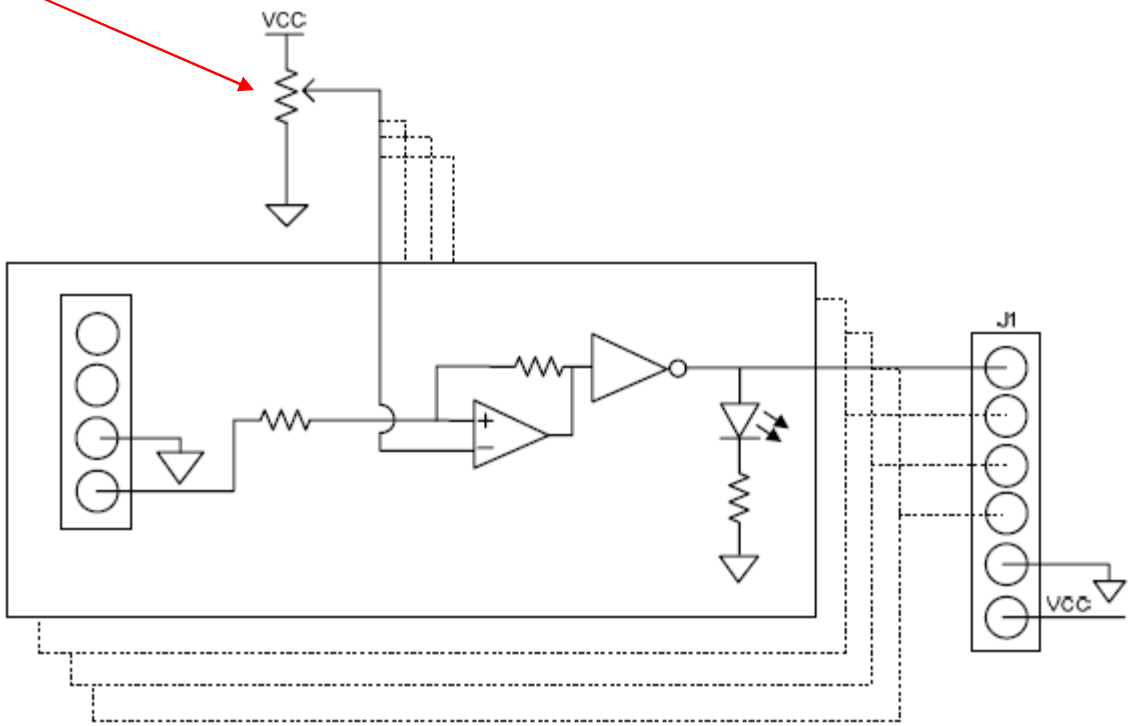


Perceptia Mediului

PMod LS1

- Se utilizeaza in tandem cu senzorul de reflectivitate IR
- Se pot atasa pana la 4 senzori
- Raspunsul analogic al senzorului se compara cu o referinta, si se generează un semnal digital pentru microcontroller
- Referinta se poate regla, ajustandu-se astfel sensibilitatea

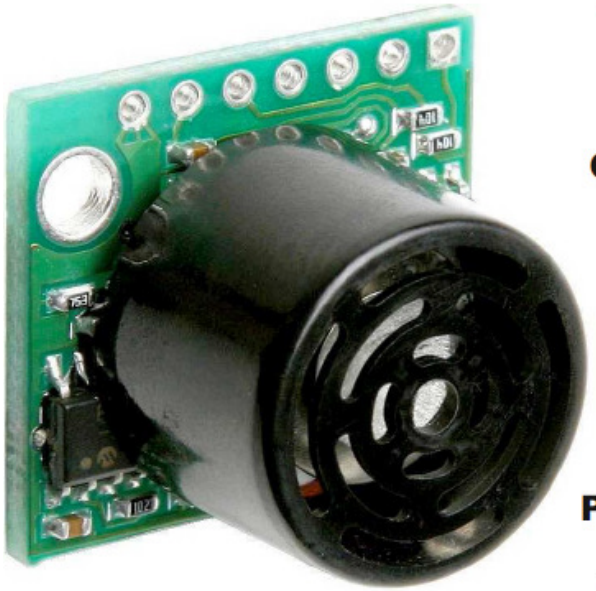


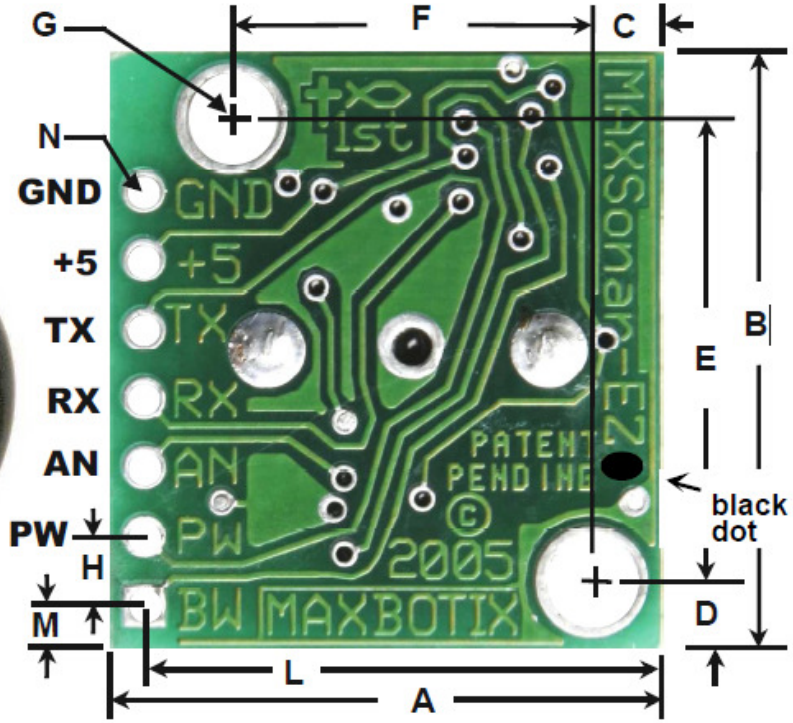


Perceptia Mediului

LV-MaxSonar-EZ0 Sonar pentru detectia obstacolelor

- Comunicare seriala (UART), Baud 9600, 8 bit date, 1 bit stop, fara paritate
- Iesire analogica, **Vcc/512** Volti per inch (1 inch = 2.54 cm)
- Iesire PWM, **0.147 ms / inch**
- Frecventa ultrasunetelor: 42 KHz
- Distanta: 0-6.45 m, depinde foarte mult de dimensiunea obstacolului
- Utilizarea cea mai simpla: folosind **convertorul Analog/Digital**

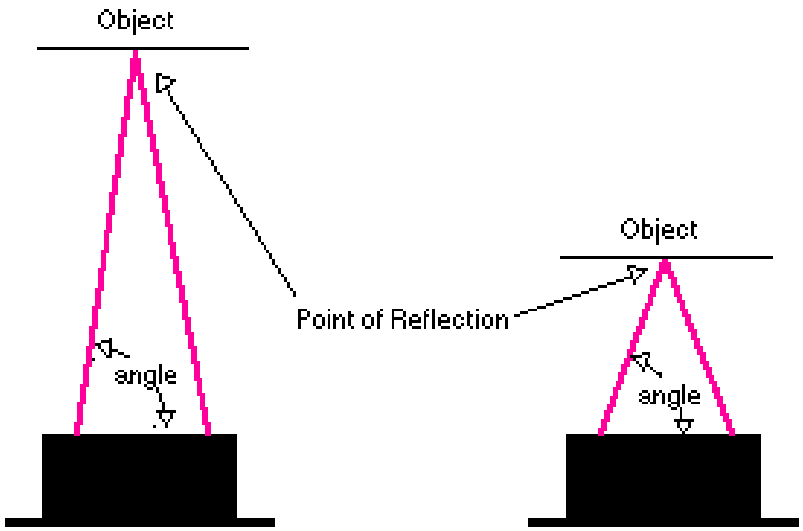


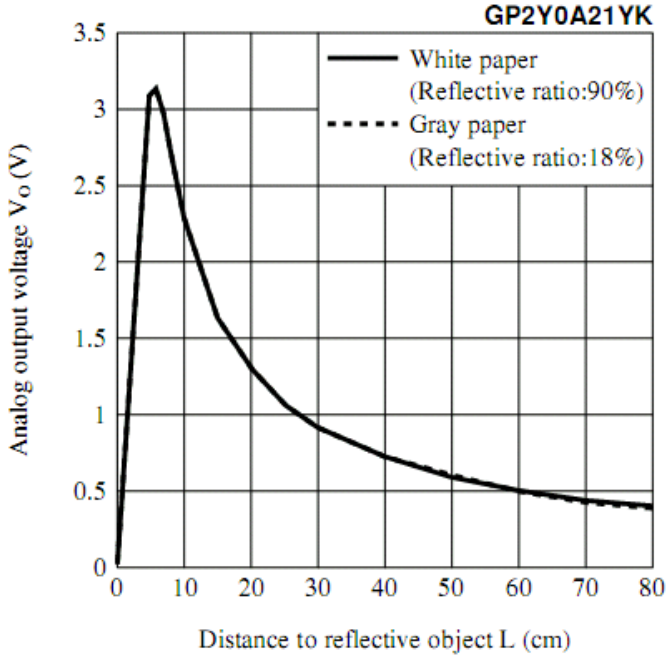


Perceptia Mediului

SHARP GP2XX, familie de senzori pentru distanța bazați pe reflexia IR

- Foloseste triangulatia pentru calculul distantei
- Se masoara unghiul sub care se intoarce raza emisa
- Iesire analogica, **neliniara**
- Cost redus (approx. 10 usd)
- Usor de montat, robust

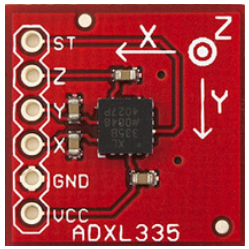




Perceptia Mediului

Accelerometru ADXL335

- Masoara acceleratia pe 3 axe, de la -3… 3 g
- Alimentare intre 1.8 … 3.6 V
- Iesire pentru 0 G: Vcc/2
- Sensibilitate tipica pentru Vcc=3.3V: 300 mv/G



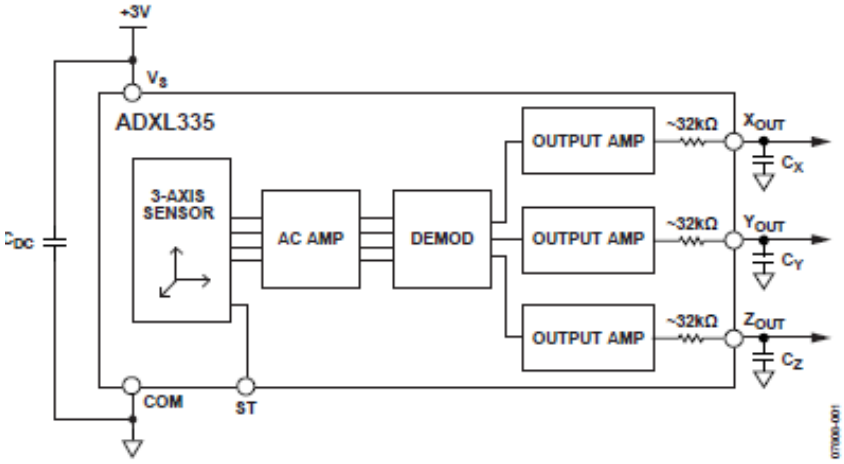


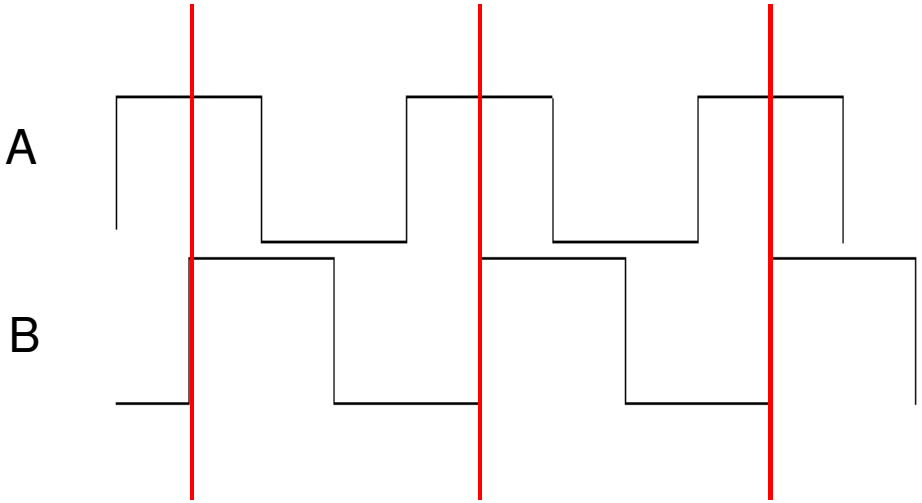
Figure 1.

Arduino 3.3 V	ADXL335 VCC
Arduino GND	ADXL335 GND
Arduino Analog0	ADXL335 X
Arduino Analog1	ADXL335 Y
Arduino Analog2	ADXL335 Z
Arduino 3.3	Arduino AREF

Motoare de curent continuu (DC)

Motor/cutie de viteze Digilent MT-Motor

- Motor clasic DC, viteza e data de tensiune, directia de polaritate
- Cutie de viteze raport 1:19 sau 1:53
- Contine senzori pentru viteza de rotatie si sens – “quadrature encoding”
  - Tren de pulsuri generate pentru fiecare rotatie
  - Viteza – proportionala cu viteza pulsurilor

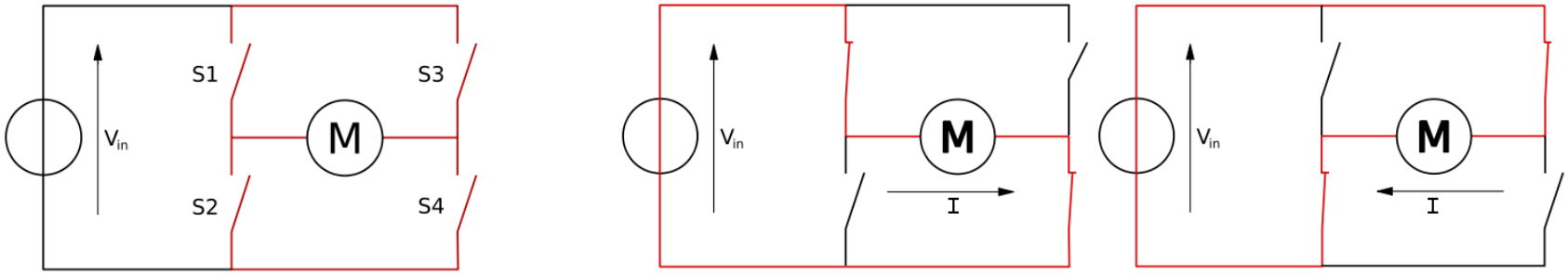


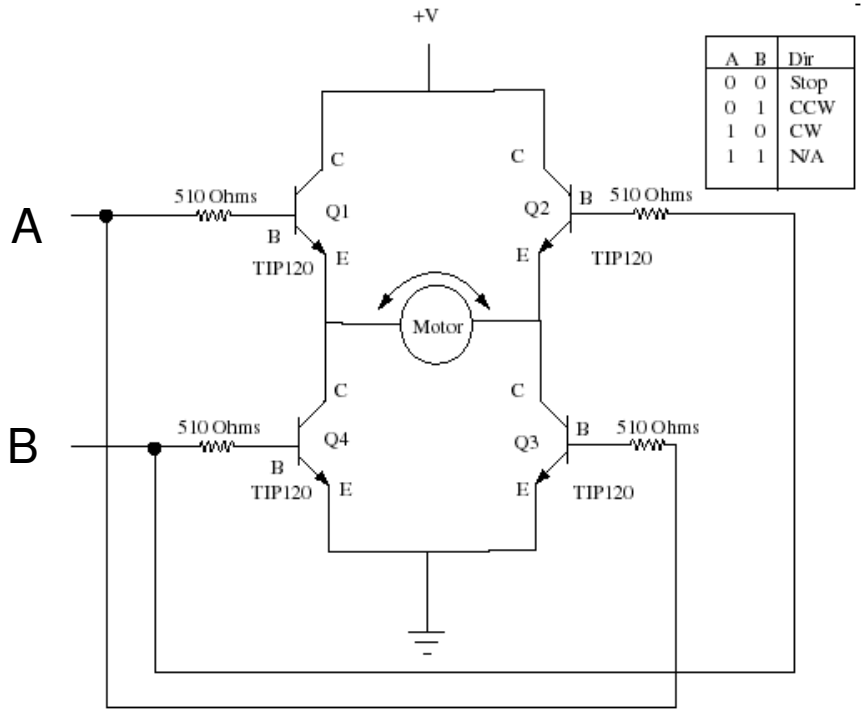
- Orientare: se monitorizeaza fronturile crescatoare sau descrescatoare ale unui semnal
- Starea celuilalt semnal in momentul tranzitiei da orientarea

Controlul motoarelor de curent continuu (DC)

Puntea H

- Controlul pornirii-opririi si a directiei unui motor





A	B	Dir
0	0	Stop
0	1	CCW
1	0	CW
1	1	N/A

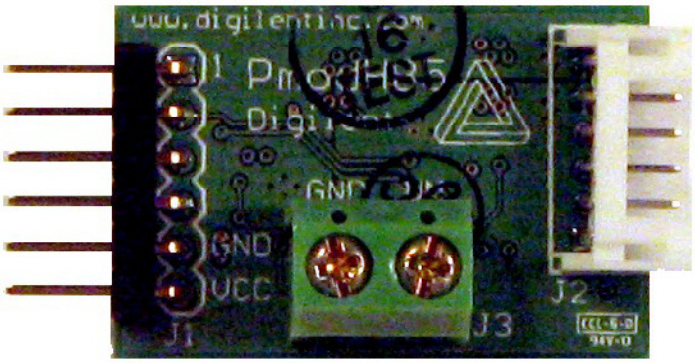
A = EN **and** DIR

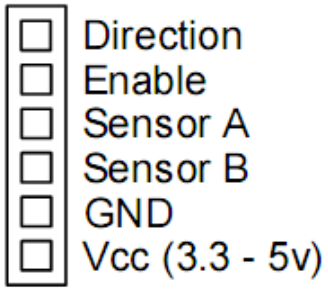
B = EN **and not**(DIR)

Controlul motoarelor de curent continuu (DC)

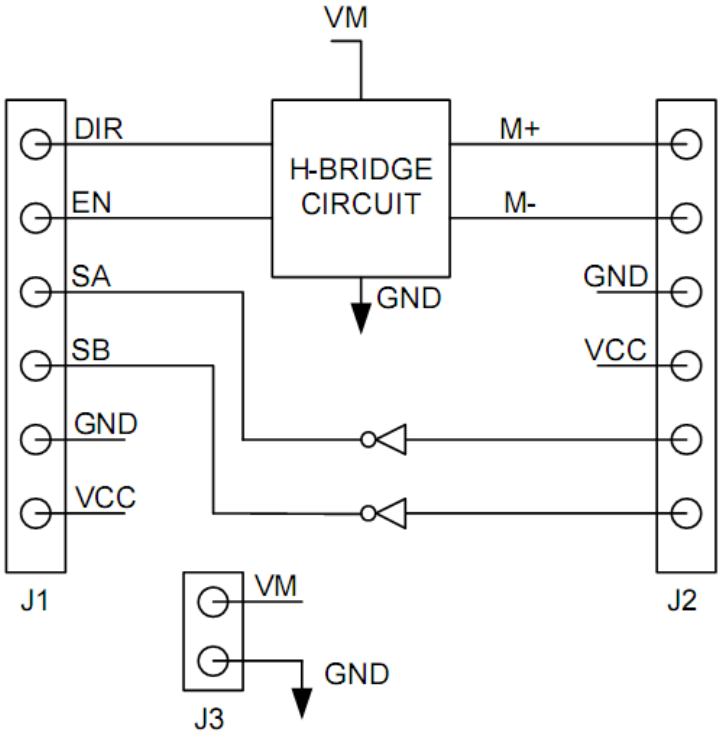
Puntea H

- Digilent PMOD HB5
- DIR – control directie
- EN – daca e ‘1’, motorul functioneaza – se poate atasa PWM pentru viteza variabila a motorului
- SA, SB – semnale de la motor, pentru a monitoriza starea acestuia





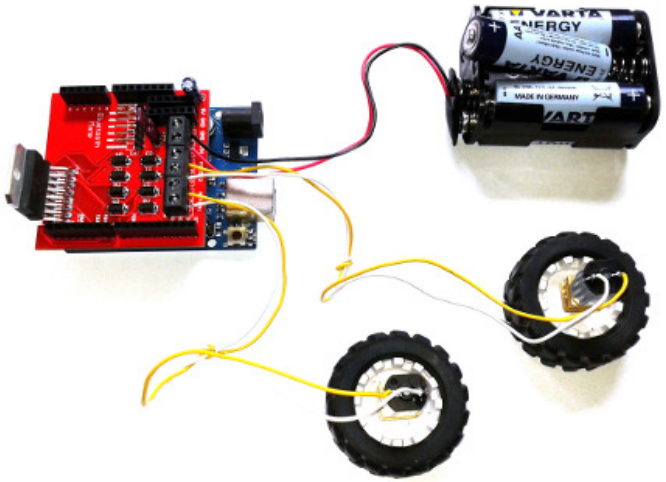
HB5 6-Pin Header, J1



Controlul motoarelor de curent continuu (DC)

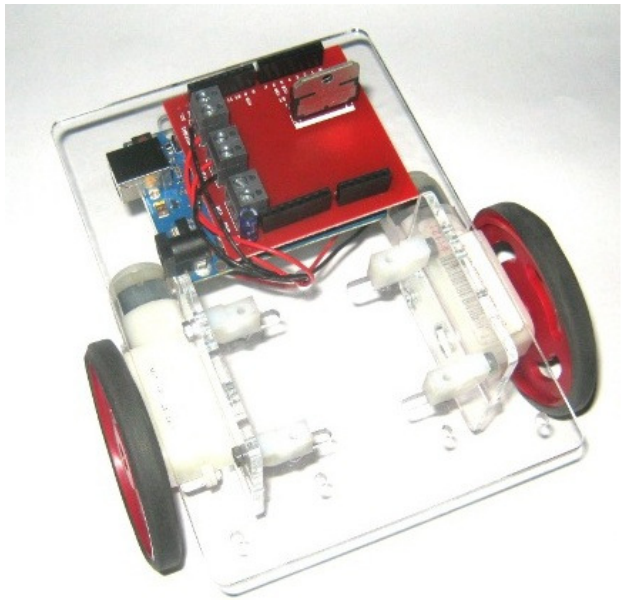
Driver-ul de motoare L298 (Shield) pentru Arduino

- http://www.robofun.ro/shields/shield-motoare-l298-v2
- Driver-ul se conecteaza la platforma Arduino folosind 4 pini digitali (3, 5, 6 si 9) prin infigere directa in pinii placii Arduino.
- Poate controla motoare care necesita cel mult 2 Amperi (2000 mA).



Platforme roboti cu motoare DC

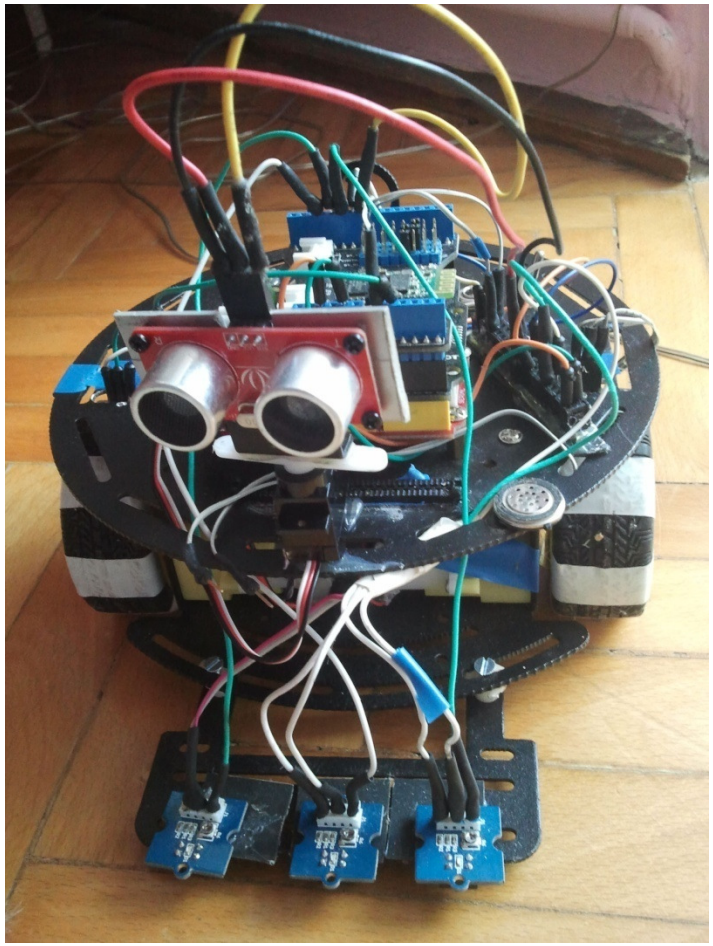
http://www.robofun.ro/kit-roboti/magician-robot-arduino-driver



http://www.robofun.ro/kit-roboti/kit-robot-senile-arduino-driver-sharp

Exemplu de aplicație complexă

Exemplu: proiectarea unui robot capabil sa se deplaseze autonom, evitand obstacolele, sau sub controlul unui operator uman, sau ghidat de o banda de culoare inchisa.



Microcontroller: AVR ATMega328, placa Arduino, programare in C/C++

Componente interne: porturi I/O, intreruperi, interfata de comunicare seriala, generator PWM

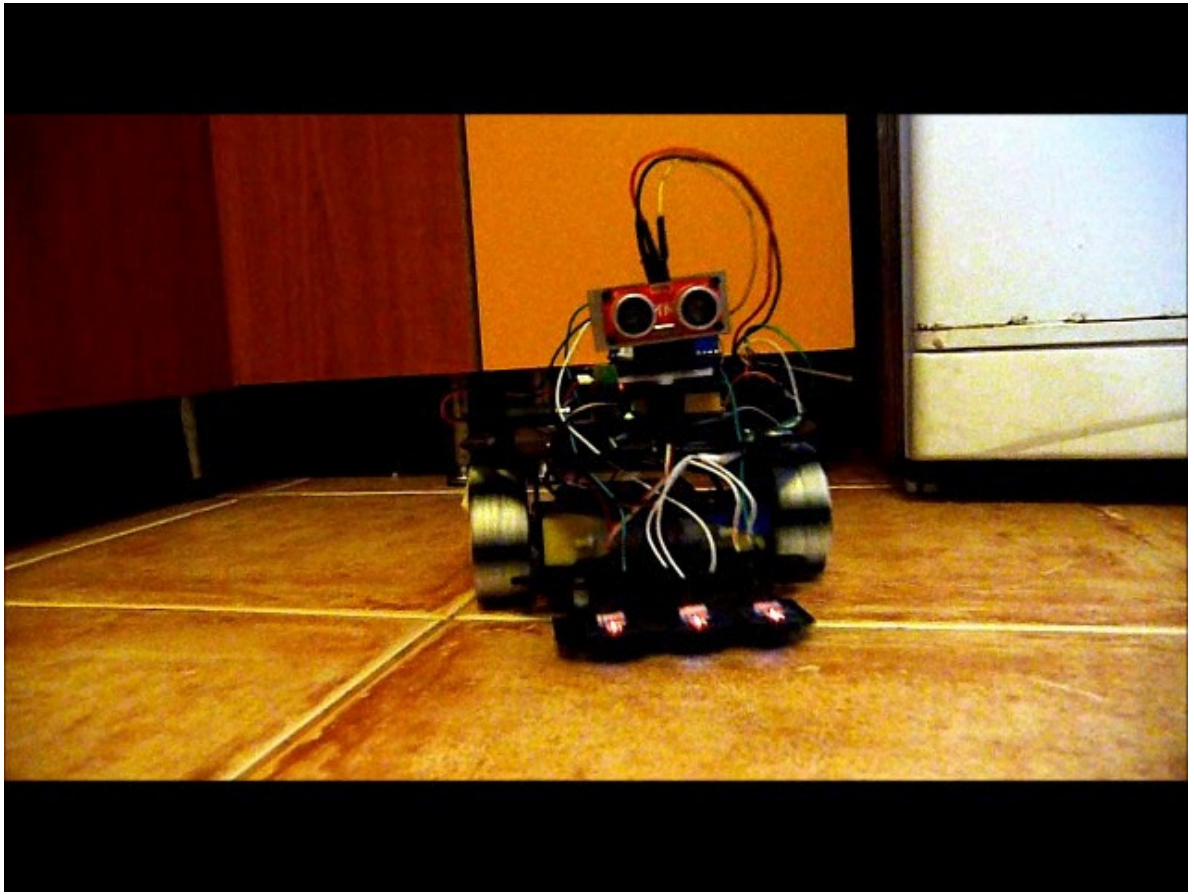
Componente externe: motoare DC, 1 motor servo, senzori de reflectivitate, punte H, senzor de distanta sonar, modul de comunicare Bluetooth.

Comunicare: seriala, tip UART, intre MCU si modulul Bluetooth, PWM intre MCU si servo, si intre MCU si puntea H, semnal analogic de la senzorii de reflectivitate, puls digital intre sonar si MCU.

Algoritmi: scanare mediu si detectia obstacolelor, urmarirea liniei, controlul rotilor pentru mersul in linie dreapta, etc.

Exemplu de aplicație complexă

Exemplu: proiectarea unui robot capabil sa se deplaseze autonom, evitand obstacolele, sau sub controlul unui operator uman, sau ghidat de o banda de culoare inchisa.



Exemplu de aplicație complexă

Exemplu: proiectarea unui robot capabil sa se deplaseze autonom, evitand obstacolele, sau sub controlul unui operator uman, sau ghidat de o banda de culoare inchisa.

Bibliografie

Slide-urile de curs ale disciplinei Proiectare cu Microprocesoare:

http://users.utcluj.ro/~rdanescu/teaching_pmp.html

Cărți:

S. Barrett, D. Pack, “Atmel AVR Microcontroller Primer: Programming and interfacing”, Morgan&Claypool, 2008.

M. Margolis, “Arduino cookbook”, 2nd edition, 2011

M. Mazidi, S. Naimi, “The AVR microcontroller and embedded system using assembly and c”, Prentice Hall, 2011.