3 Convertoare analog-numerice cu numărare, urmărire şi aproximări succesive cu logică realizată cu ajutorul microcontrolelor AVR

3.1 Scopul lucrarii

Lucrarea prezintă convertoare anlog-numerice cu numărare, urmărire şi aproximări succesive.

3.2 Consideratii teoretice

Circuitele de conversie a datelor folosite împreună cu un sistem de tip calculator numeric constituie surse (în cazul convertoarelor analog-numerice) sau destinaţii (în cazul convertoarelor numeric-analogice) pentru datele numerice cu care lucrează calculatorul. În aplicaţii, convertoarele pot fi considerate ca elemente periferice ale sistemului de intrare-ieşire al calculatorului, sau pot fi asimilate unor adrese de memorie. În cele ce urmează, convertorul va fi comandat de către calculatorul numeric prin instrucţiuni de intrare-ieşire. În figura 3.1 este prezentată o schemă de conversie prin program.

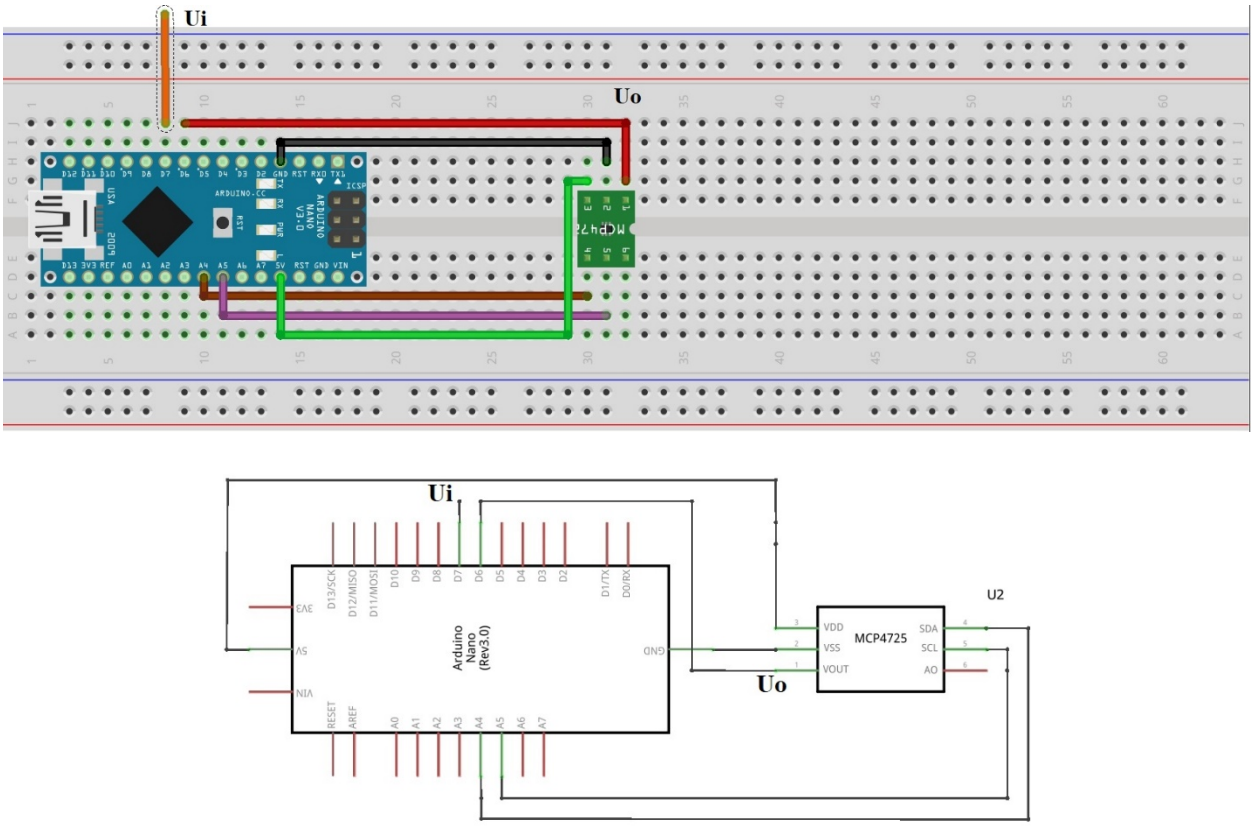

Fig. 3.1. Convertor analog-numeric cu reacţie cu logică de conversie prin program

Comanda convertorului analog-numeric se face prin intermendiul mediului de programare Arduino CC si protocolul I2C.

(1)

Tensiunea de intrare U_i se compară cu tensiunea U_0 şi comparatorul basculează în momentul când cele două tensiuni sunt egale $U_0 = U_i$. Starea comparatorului este citită in mediul de programare Arduino CC, prin citirea registrului *ACSR*, mai precis bitul 6 din acest registru.

Algoritmul de conversie poate fi implementat în mai multe variante, de unde şi tipurile de convertoare analog-numerice cu reacţie:

- convertor analog-numeric cu numărare;
- convertor analog-numeric cu urmărire;
- convertor analog-numeric cu aproximaţii succesive.

3.2.1 Convertor analog-numeric cu algoritm de numărare

În figura 2 poate fi urmărită evoluţia tensiunii de ieşire a convertorului numeric-analogic în funcţie de tensiunea de intrare.

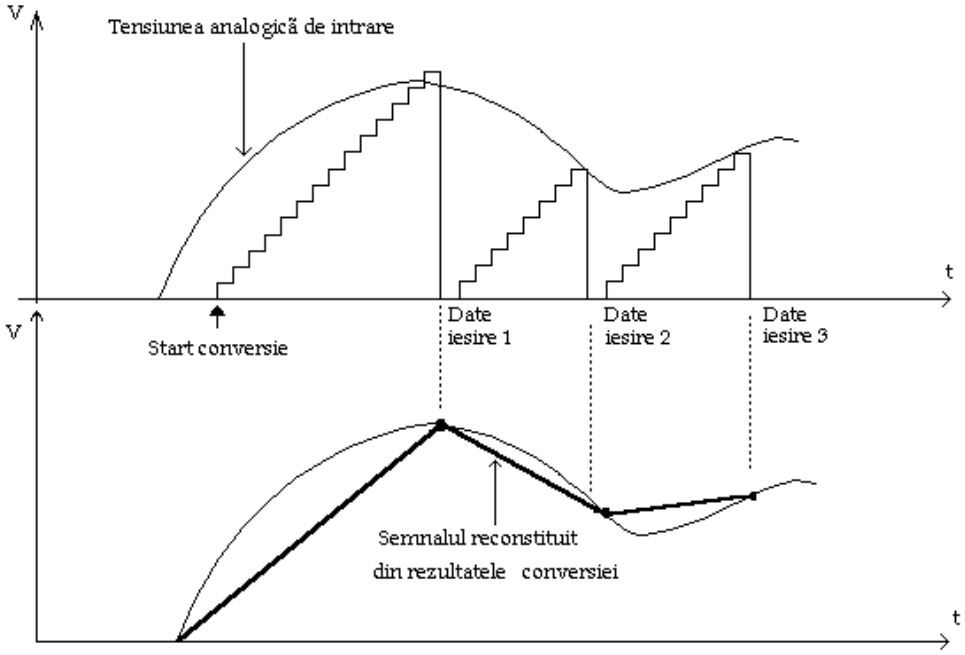

Fig. 2 Conversia unui semnal cu variaţie continuă în timp cu convertor analog-numeric de tip numărător

Iniţial conţinutul numărătorului este zero. Comparatorul "testează" U_0 şi U_i. Dacă în urma testului $U_0 < U_i$, conţinutul numărătorului se incrementează. Odată cu creşterea conţinutului numărătorului creşte şi tensiunea de la ieşirea convertorului numeric-analogic. Incrementarea numărătorului are loc până când $U_0 \geq U_i$. În acest moment se citeşte valoarea numărătorului care este tocmai rezultatul conversiei analog-numerice. Programul conţine o variabilă de tip întreg pe care o notăm cu *val*. La parcurgerea buclei de test a comparatorului, valoarea anterioară a variabilei *val* va fi înlocuită cu $val \leftarrow val+1$.

Ieşirea din bucla de test este condiţionată de bascularea comparatorului, adică $U_0 = U_i$. În figura 3 poate fi urmărită organigrama şi în continuare secvenţa de program corespunzătoare pentru algoritmul de numărare.

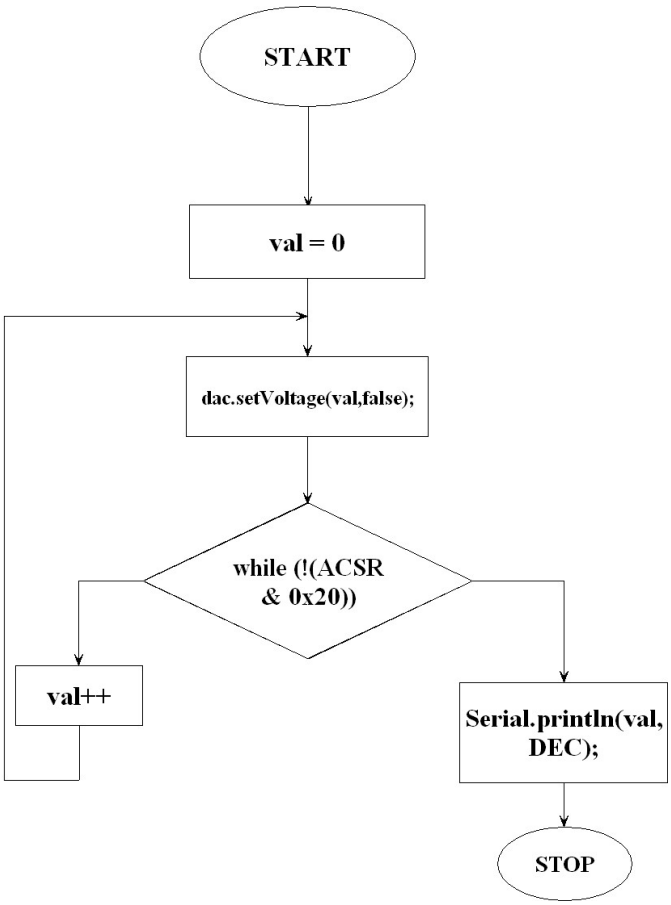

Fig. 3. Organigrama pentru algoritmul de tip numărător

```
#include<avr/io.h>
#include <Adafruit_MCP4725.h>
#include <Wire.h>
Adafruit_MCP4725 dac;
void setup() {
  Serial.begin(9600);
 dac.begin(0x60);
}
int val, oldval, i=0;
void loop() {

  char input[5];
  int charsRead;
  String rx_str = "       ";

  byte vall=0;
   DDRB = 0x20;
    ACSR = 0x00;
  if ((Serial.available() > 0)) {
    while(Serial.available() > 0) {
```

```
      rx_str = Serial.readString();
      }
      for (int i =0; i < 4; i++)
      {
      input[i] = rx_str[i];

      }
      input[5] = '\0';
    }
      if (input[0] = '\n')
       val = 0 ;
      if (input[0]= 'a')
      {
        delay(5000);
         unsigned char temp;
         unsigned char q;
         dac.setVoltage(val,false);
         Serial.println(ACSR & 0x20);
     while (!(ACSR & 0x20))
     {
      dac.setVoltage(val,false);
      val++;
      Serial.println(val,DEC);
      delay(100);
     }

        PORTB &= ~0x20;
        Serial.println(val,DEC);
        Serial.print("Tensiunea este:");
        Serial.println(val*1.14); //

}
}
```

Se observă că durata unei conversii depinde de viteza de execuţie a instrucţiunilor, respectiv a calculatorului şi de valoarea tensiunii de intrare U_i. Pentru tensiuni mari la intrare rezultă

$$T_C = N \cdot T_0$$

un timp de conversie mare. Deci timpul de conversie T_C este egal cu durata de parcurgere a buclei while T_0 şi numărul de parcurgeri al acesteia *val*.

3.2.2 Convertor analog-numeric cu algoritm de urmărire

În cazul semnalelor de intrare care nu au variaţii în salt sau trepte mari, este mai convenabil a se utiliza o numărare înainte-înapoi. În figura 4 poate fi urmărită evoluţia tensiunii la ieşirea convertorului numeric-analogic U_0 în funcţie de tensiunea de intrare U_i

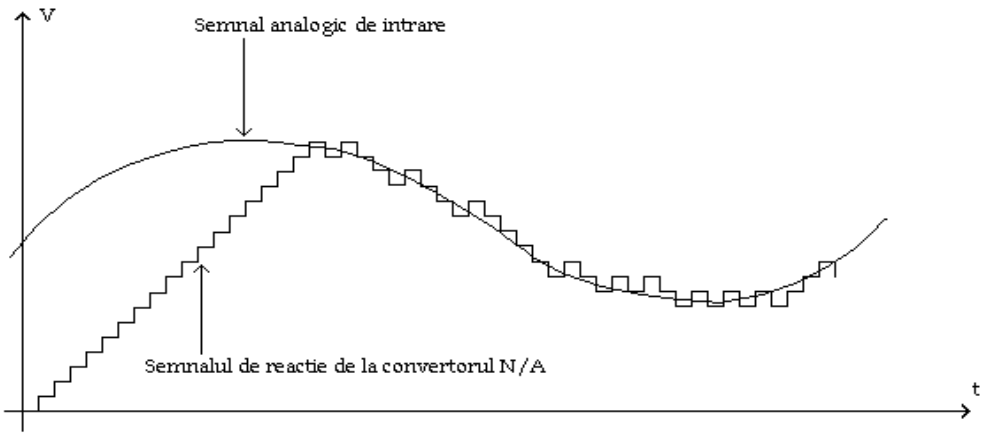

Fig. 4. Variaţia tensiunii de reacţie pentru convertorul analog-numeric cu algoritm de urmărire

Se observă că la început numărarea are loc doar într-un sens, până când $U_0 > U_i$. În acest moment are loc bascularea comparatorului şi in program se va lua decizia de numarare inapoi cu val ← val-1. Dar în scurt timp, datorită scăderii conţinutului numărătorului, scade $U_0 < U_i$. Comparatorul basculează în cealaltă stare şi decizia programului va fi de numărare înainte val ← val+1. În figura 5 poate fi urmărită organigrama programului urmată de program.

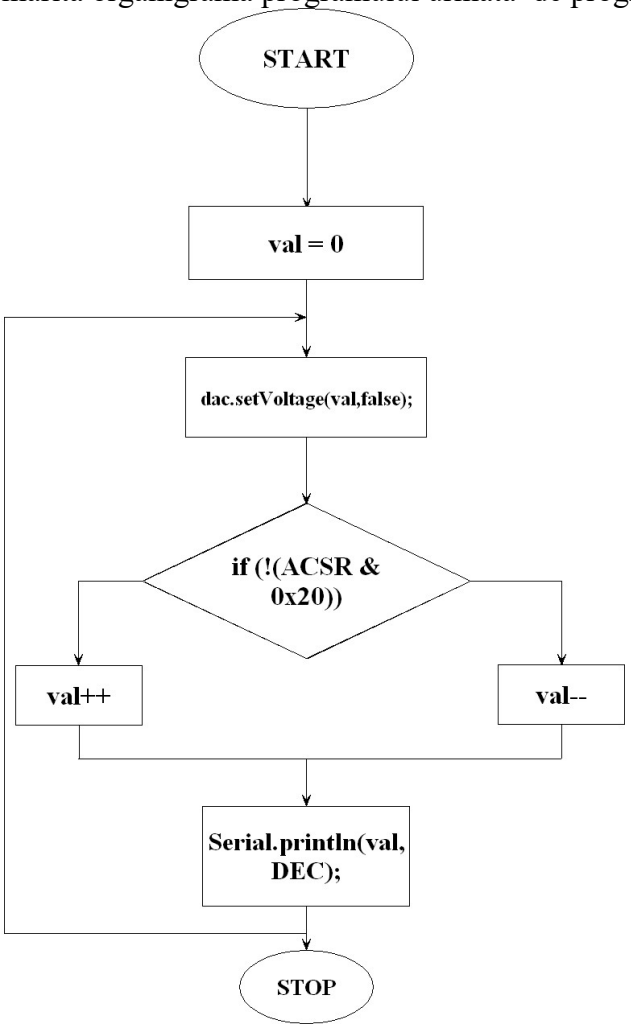

Fig. 5 Organigrama pentru algoritmul de urmărire al convertorului analog-numeric

```
#include<avr/io.h>
#include <Adafruit_MCP4725.h>
#include <Wire.h>
Adafruit_MCP4725 dac;
void setup() {
  Serial.begin(9600);
 dac.begin(0x60);
  DDRB = 0x20;
  ACSR = 0x00;
}
int val = 2048;
void loop() {
   if (!(ACSR & 0x20))
   {
    dac.setVoltage(val,false);
    val++;
    if (val > 4095) val = 4095;
    Serial.println(val*1.14);
    delay(100);
   }
     else
     {
    dac.setVoltage(val,false);
    val--;
    if (val < 0 ) val = 0;
    Serial.println(val*1.14);
    delay(100);
   }
}
```

Observaţie: După intrarea convertorului în regim de urmărire, timpul de conversie al convertorului T_C este tocmai durata unui ciclu T_0, deci $T_C = T_0$. Viteza de urmărire este limitată de viteza de lucru a calculatorului, precum şi de frecvenţa tensiunii de intrare. Pentru a putea urmări un semnal sinusoidal de frecvenţă f, condiţia limită este:

$$f \le \frac{1}{\pi \cdot T_0 \cdot 2^n} \tag{3}$$

unde T_0 este durata unui ciclu, iar n – numărul de biţi ai convertorului.

3.2.3 Convertor analog-numeric cu algoritm de aproximări succesive

Aceste tipuri de convertoare funcţionează corect numai dacă tensiunea aplicată la intrare este constantă sau memorată pe durata conversiei. Convertoarele cu aproximări succesive funcţionează corect numai asociate cu circuite de eşantionare-memorare, ceea ce a dus la integrarea celor două circuite în aceeaşi capsulă. Algoritmul de funcţionare este simplu: se împarte domeniul

$$T_C = n \cdot T_0$$

(4)

de lucru al intrării în două părţi egale şi se determină cu ajutorul comparatorului în care parte se găseşte mărimea de intrare. Intervalul obţinut se împarte din nou în două părţi egale şi procedeul continuă până la obţinerea rezoluţiei dorite. Se obţin astfel n înjumătăţiri, sau altfel spus, durata conversiei T_C este n durate T_0.

unde n este numărul de biţi ai convertorului şi T_0 durata unui ciclu de instrucţiuni.

În figura 6 este prezentat algoritmul de aproximare succesivă, iar în figura 7 evoluţia tensiunii la ieşirea convertorului numeric-analogic.

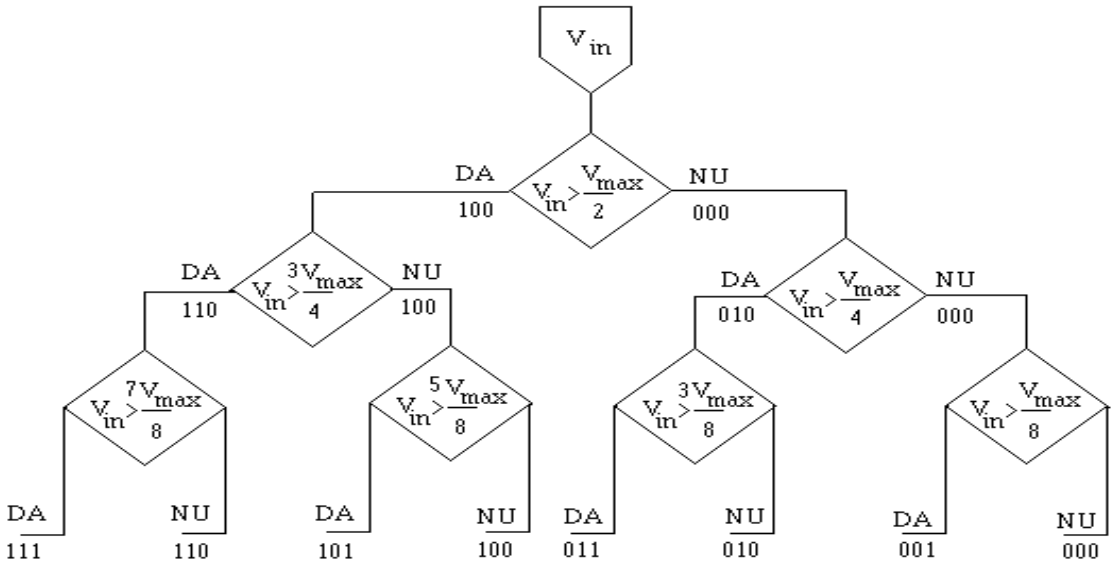

Fig. 6 Schema comparărilor posibile şi a codurilor ce pot rezulta la un convertor analog-numeric cu aproximări succesive de 3 biţi.

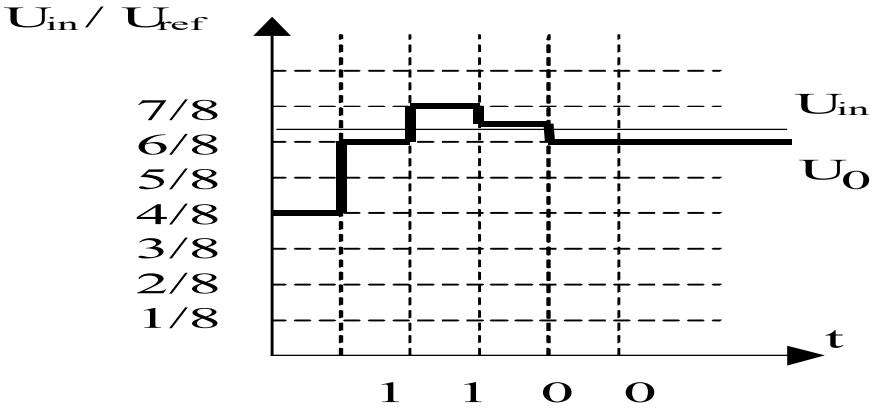

Fig. 7 Evoluţia tensiunii U_0 la ieşirea convertorului numeric-analogic

În figura 8 este ilustrată organigrama pentru conversia cu aproximaţii succesive şi în continuare o variantă de program pentru a implementa automat acest algoritm.

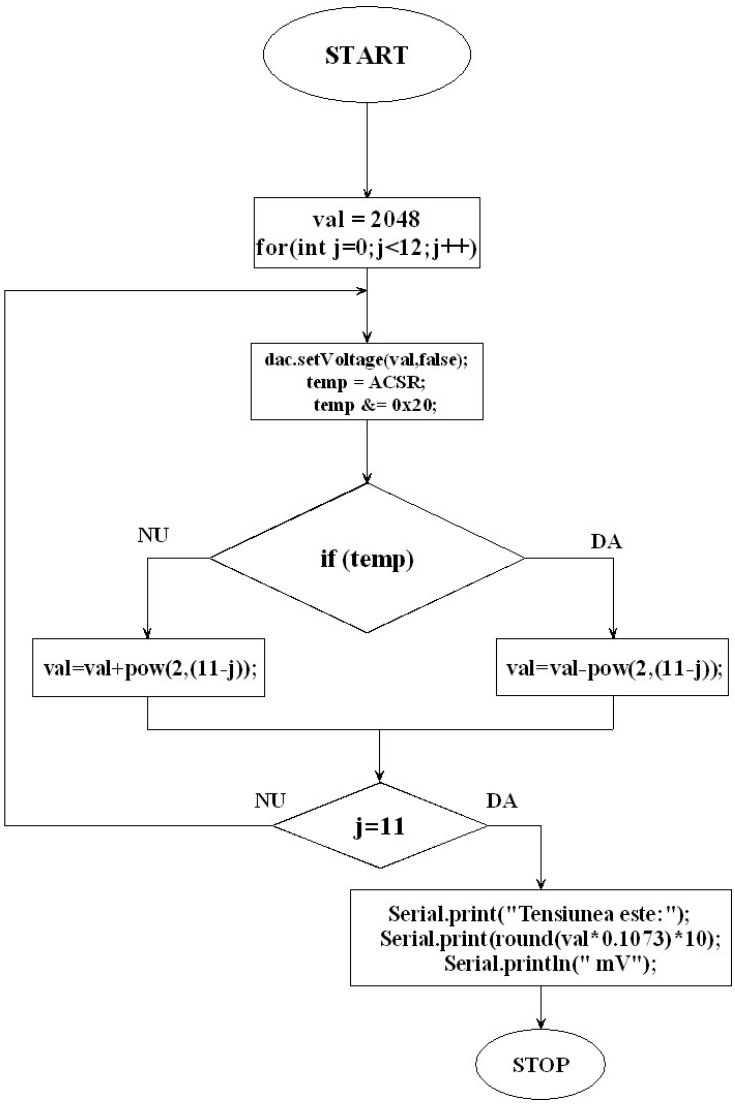

Fig. 8 Schema logică pentru conversia cu aproximaţii succesive

```
#include<avr/io.h>
#include <Adafruit_MCP4725.h>
#include <Wire.h>
Adafruit_MCP4725 dac;
void setup() {
  Serial.begin(9600);
 dac.begin(0x60);
}
int val, oldval;
void loop() {
  char input[5];
  int charsRead;
  String rx_str = "       ";
  byte vall=0;
  if ((Serial.available() > 0)) {
    while(Serial.available() > 0) {
```

```
        rx_str = Serial.readString();
        }
        for (int i =0; i < 4; i++)
        {
        input[i] = rx_str[i];
        }
        input[5] = '\0';
      }
         DDRB = 0x20;
        if (input[0]!= '\n' )
        {
      val = 2048;
      dac.setVoltage(val,false);
      rx_str= "";
      for(int j=0;j<12;j++)
    {
         dac.setVoltage(val,false);
       unsigned char temp;
      ACSR = 0x00;
        temp = ACSR;
        temp &= 0x20;
     if (val != oldval && val>=0 && val<4096)
          {
          oldval = val;
        if(temp & 0x20)
        {
           Serial.print(val,DEC);
          PORTB &= ~0x20;
          val=val-pow(2,(11-j));
        }
        else
    {
      Serial.print(val,DEC);
          PORTB |= 0x20;
          val=val+pow(2,(11-j));
    }
          }
         else {
          delay(5000);
            Serial.println("valoarea  depaseste  domeniul
0-4095 mV");
          }
    }
    delay(1000);
    }
        Serial.print("Tensiunea este:");
        Serial.print(round(val*0.1073)*10);
        Serial.println(" mV");
      }
```

3.3 **Desfăşurarea lucrării**

- Se va utiliza limbajul de programare Arduino CC pentru a comanada convertorul numeric-analogic şi se va citi starea comparatorului din registru *ACSR*. Se vor testa cele trei tipuri de convertoare analog-numerice.
- Comandarea convertorului se va face in sistem: binar, hexazecimal si zecimal.
- Se vor rula programele de mai sus şi se va urmari functioanarea convertoarelor.
- Se vor scrie programe care sa reprezinte grafic valorile numerice obtinute.