2 Utilizarea Convertorului Numeric Analogic MCP4725 pentru aplicaţii: sursa de tensiune programabilă, generator de semnal de formă arbitrară

2.1 Scopul lucrarii

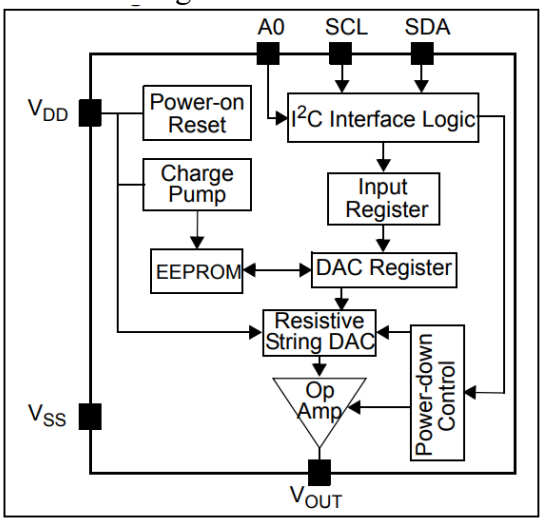
Lucrarea prezinta aplicatii a convertoarelor numeric-anlogice comandate cu ajutorul unui microcontroler.

2.2 Consideratii teoretice

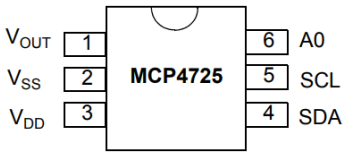
Funcţionarea circuitelor electronice necesită alimentare de la o sursă de curent continuu. Energia este obţinută prin redresarea, filtrarea, stabilizarea tensiunii alternative a reţelei de energie electrică.

Pentru o versatilitate cât mai mare, este de preferat ca sursa să fie reglabilă. În general laboratoarele folosesc surse reglabile sau programabile. Sursele programabile, a căror valoare de tensiune poate fi riguros reglată se pretează în laboratoarele de metrologie pentru verificarea aparatelor de măsurare.

Pentru aceasta lucrare de laborator se va folosi Convertorul Numeric Analogic MCP 4725. Diagrama bloc a acestuia este prezentata in figura 2.1:



Iar in figura 2.2 este prezentata diagrama pinilor de conexiune:



MCP 4725 este un convertor numeric analogic, pe 12 biti, bazat pe o retea de rezistente R-2R. Pentru comutarea bitilor se foloseste de interfata I2C. Poate functiona cu o tensiune de alimentare intre 3-5V, iar adresa de baza a acestuia este 0x62. Daca pinul A0 este conectat la tensiunea de alimentare atunci adresa accestuia devine 0x63.

Pentru a realiza sursa programabila de tensiune se va realiza montajul din figura 2.1. Astfel de la pinul A4 ne vom conecta la pinul 4 (SDA) al convertorului, A5 la pinul 5 (SCL), La pinul 3 (VDD) ne vom conecta de la tensiunea de 5V, iar cu pinul 2 (VSS) ne conecta la masa (GND).

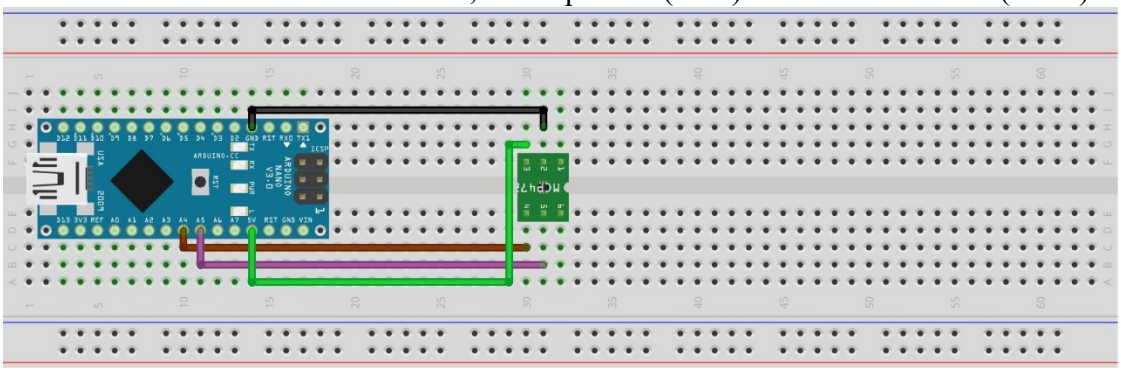


Figura 2.1 Schema de montaj pentru conectarea CNA MCP 4725

Iar in figura 2.2 se prezinta schema electrica de conexiune:

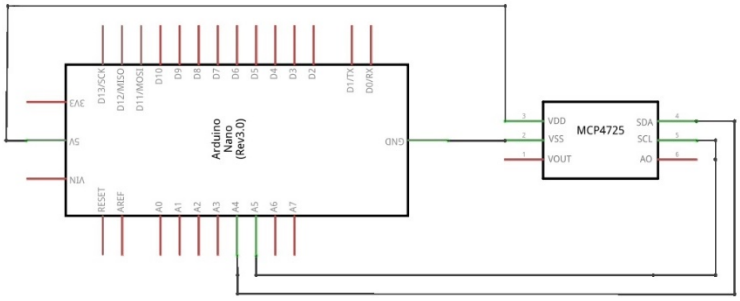


Figura 2.2. Schema electrica de conexiune a CNA MCP 4725

Tensiunea de iesire a convertorului va fi citita cu ajutorul unui voltmetru conectat intre pinul 1 (VOUT) si pinul 2 (VSS->GND). Comunicarea cu convertorul numeric analogic se realizeaza prin intermediul protocolului I2C. Astfel din mediul de programarea Arduino CC se vor realiza programe pentru a varia tensiunea de iesire a convertorului

2.3 Reglarea tensiunii de iesire prin metoda aproximarii succesive

Se va realiza un program prin care se va regla tensiunea de iesire, utilizand metoda aproximarii succesive. Un exemplu pentru acesta metoda este prezentat mai jos:

```
#include <Adafruit_MCP4725.h>
#include <Wire.h>

Adafruit_MCP4725 dac;
void setup() {
  Serial.begin(9600);
 dac.begin(0x60);
}

void loop() {

  char input[13];
  int charsRead;
  String rx_str = "       ";
```

```
  int val;
  if ((Serial.available() > 0)) {
   //charsRead=Serial.readBytes(input, 2);
    while(Serial.available() > 0) {
    rx_str = Serial.readString();
    }
    for (int i =0; i < 12; i++)
    {
    input[i] = rx_str[i];

    }
    input[12] = '\0';
    if (input[0]!= '\n' )
    {
    val = StrToBIN(input);
  }
  Serial.println(val);
  dac.setVoltage(val,false);
  }
   rx_str= "";
}

int StrToBIN(char str[])
{
  return (int) strtol(str, 0, 2);
}
```

In consola programului se va scrie valoarea numerica N in binar corespunzatoarea pozitilor bitilor CNA. In functie de valoarea indicata de voltmetru se va regla valoare numerica N, pana cand se ajunge la tensiunea dorita.

- Se va modifica programul astfel incat valoarea numerica sa poata fi introdusa in sistem:
    - Hexazecimal
    - Zecimal

Tabel 2.1. Corespondenţa între sistemele de numeratie binar-hexa-zecimal:

Binar	Hexa	Zecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14

1111	F	15

2.4 Sursa programabila de tensiune

Pornind de la calculul tensiuni date de cel mai putin semnificativ bit (V_{LSB}) se poate determina ce tensiune de iesire va fii pentru incrementarea valori numerice N, cu orice numar de biti, cunoscandu-se tensiunea de iesire. Se va implementa un program sub forma celui prezentat mai jos pentru a incrementa si decrementa tensiunea de iesire:

```
#include <Adafruit_MCP4725.h>
#include <Wire.h>
int StrToDec(char str[])
{
  return (int) strtol(str, NULL, 10);
}
Adafruit_MCP4725 dac;
 int val = 0;
void setup() {
  Serial.begin(9600);
 dac.begin(0x60);
  Serial.println("q/w incrementare 1000 mV");
  Serial.println("a/s incrementare 100 mV");
  Serial.println("z/x incrementare 10 mV");
}
void loop() {
  char input[10];
  int charsRead;
  String rx_str = "                    ";
  if ((Serial.available() > 0)) {
   //charsRead=Serial.readBytes(input, 2);
    while(Serial.available() > 0) {   //while there are characters in the
serial buffer, because Serial.available is >0
     rx_str = Serial.readString();
     }
     // get one character
     for ( int i=0; i<2; i++)
       input[i] = rx_str[i];
     input[1] = '\0';// Make it a string
          // Ends the serial communication once all data is receive
     if (input[0]!= '\n' )
     {
       switch (input[0]) {
     case 'a':    // your hand is on the sensor
       val = val + (100/1.143);
       break;
     case 's':    // your hand is close to the sensor
       val = val - (100/1.143);
       break;
     case 'q':    // your hand is a few inches from the sensor
       val = val + (1000/1.143);
       Serial.println(val);
       break;
     case 'w':    // your hand is nowhere near the sensor
       val = val - (1000/1.143);
       break;
     case 'z':    // your hand is nowhere near the sensor
```

```
      val = val + (10/1.143);
      break;
    case 'x':    // your hand is nowhere near the sensor
      val = val - (10/1.143);
      break;
     }
      if (val < 0)
          val = 0;
      if (val > 4095)
          val = 4095;
    }
     if ( val >=0 && val <= 4095) {
    Serial.print("Tensiunea este:");
    Serial.print(round(val*0.1143)*10);
    Serial.println(" mV");
    dac.setVoltage(val,false);
     }
  }
   rx_str= "";
}
```

- Se va modifica programul anterior astfel incat in consola seriala sa se poata introduca direct tensiune a dorita la iesire in mV, prin calcularea valorii numerice necesare pentru a obtine accea tensiune la iesire.

2.5 Generator de semnale

Avand in vedere posbilitatea modificarii tensiuni de iesire in functie de valoarea numerica trimisa spre convertor, se pot realiza programe pentru a genera diferite forme de unda.

2.5.1 Semnal dreptunghiular

Pentru a genera un semnal dreptunghiular practic este nevoie de a genera doua valorii de tensiune diferite, la un anumit interval de timp:

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>
Adafruit_MCP4725 dac;
void setup(void) {
  Serial.begin(9600);
  dac.begin(0x62);
  int val_min=0;
  int val_max=4095;
}
void loop(void) {
      dac.setVoltage(val_max, false);
      delay(1000);
      dac.setVoltage(val_min, false);
      delay(100);
}
```

- Se va modifca programul de mai sus, astfel incat sa varieze: amplitudinea, frecventa si factorul de umplere a semnalului dreptunghiular.

2.5.2 Semnal triunghiular.

Pentru a realiza semnalul triunghiular tensiunea de la iesire trebuie sa creasca de la o valoarea minima pana la o valoarea maxima treptat, si apoi sa scada iar la valoarea minima. Un exemplu pentru a genera un semnal triunghiular este prezentat mai jos:

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>
Adafruit_MCP4725 dac;
void setup(void) {
  Serial.begin(9600);
  dac.begin(0x62);
  int valmin=0;
  int valmax=4095;
}
void loop(void) {
    uint32_t counter;
    for (counter = valmin; counter < valmax; counter++)
    {
      dac.setVoltage(counter, false);
      delay(0);
    }
    for (counter = valmax; counter > valmin; counter--)
    {
      dac.setVoltage(counter, false);
       delay(0);
    }
}
```

- Se va modifca programul de mai sus, astfel incat sa varieze: amplitudinea, frecventa, timpul de urcare si coborare.
- De asemenea se va modifica programul pentru a genera un semnal de tip rampa.

2.5.3 Semnal sinusoidal

Programul urmator genereaza valorile numerice pentru a realiza o forma de unda sinusoidala:

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>

Adafruit_MCP4725 dac;

// Set this value to 9, 8, 7, 6 or 5 to adjust the resolution
#define DAC_RESOLUTION    (8)
const PROGMEM uint16_t DACLookup_FullSine_9Bit[512] =
{
  2048, 2073, 2098, 2123, 2148, 2174, 2199, 2224,
  2249, 2274, 2299, 2324, 2349, 2373, 2398, 2423,
  2448, 2472, 2497, 2521, 2546, 2570, 2594, 2618,
  2643, 2667, 2690, 2714, 2738, 2762, 2785, 2808,
  2832, 2855, 2878, 2901, 2924, 2946, 2969, 2991,
  3013, 3036, 3057, 3079, 3101, 3122, 3144, 3165,
  3186, 3207, 3227, 3248, 3268, 3288, 3308, 3328,
  3347, 3367, 3386, 3405, 3423, 3442, 3460, 3478,
  3496, 3514, 3531, 3548, 3565, 3582, 3599, 3615,
  3631, 3647, 3663, 3678, 3693, 3708, 3722, 3737,
  3751, 3765, 3778, 3792, 3805, 3817, 3830, 3842,
  3854, 3866, 3877, 3888, 3899, 3910, 3920, 3930,
  3940, 3950, 3959, 3968, 3976, 3985, 3993, 4000,
  4008, 4015, 4022, 4028, 4035, 4041, 4046, 4052,
```

```
  4057, 4061, 4066, 4070, 4074, 4077, 4081, 4084,
  4086, 4088, 4090, 4092, 4094, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4094, 4092, 4090, 4088,
  4086, 4084, 4081, 4077, 4074, 4070, 4066, 4061,
  4057, 4052, 4046, 4041, 4035, 4028, 4022, 4015,
  4008, 4000, 3993, 3985, 3976, 3968, 3959, 3950,
  3940, 3930, 3920, 3910, 3899, 3888, 3877, 3866,
  3854, 3842, 3830, 3817, 3805, 3792, 3778, 3765,
  3751, 3737, 3722, 3708, 3693, 3678, 3663, 3647,
  3631, 3615, 3599, 3582, 3565, 3548, 3531, 3514,
  3496, 3478, 3460, 3442, 3423, 3405, 3386, 3367,
  3347, 3328, 3308, 3288, 3268, 3248, 3227, 3207,
  3186, 3165, 3144, 3122, 3101, 3079, 3057, 3036,
  3013, 2991, 2969, 2946, 2924, 2901, 2878, 2855,
  2832, 2808, 2785, 2762, 2738, 2714, 2690, 2667,
  2643, 2618, 2594, 2570, 2546, 2521, 2497, 2472,
  2448, 2423, 2398, 2373, 2349, 2324, 2299, 2274,
  2249, 2224, 2199, 2174, 2148, 2123, 2098, 2073,
  2048, 2023, 1998, 1973, 1948, 1922, 1897, 1872,
  1847, 1822, 1797, 1772, 1747, 1723, 1698, 1673,
  1648, 1624, 1599, 1575, 1550, 1526, 1502, 1478,
  1453, 1429, 1406, 1382, 1358, 1334, 1311, 1288,
  1264, 1241, 1218, 1195, 1172, 1150, 1127, 1105,
  1083, 1060, 1039, 1017,  995,  974,  952,  931,
   910,  889,  869,  848,  828,  808,  788,  768,
   749,  729,  710,  691,  673,  654,  636,  618,
   600,  582,  565,  548,  531,  514,  497,  481,
   465,  449,  433,  418,  403,  388,  374,  359,
   345,  331,  318,  304,  291,  279,  266,  254,
   242,  230,  219,  208,  197,  186,  176,  166,
   156,  146,  137,  128,  120,  111,  103,   96,
    88,   81,   74,   68,   61,   55,   50,   44,
    39,   35,   30,   26,   22,   19,   15,   12,
    10,    8,    6,    4,    2,    1,    1,    0,
     0,    0,    1,    1,    2,    4,    6,    8,
    10,   12,   15,   19,   22,   26,   30,   35,
    39,   44,   50,   55,   61,   68,   74,   81,
    88,   96,  103,  111,  120,  128,  137,  146,
   156,  166,  176,  186,  197,  208,  219,  230,
   242,  254,  266,  279,  291,  304,  318,  331,
   345,  359,  374,  388,  403,  418,  433,  449,
   465,  481,  497,  514,  531,  548,  565,  582,
   600,  618,  636,  654,  673,  691,  710,  729,
   749,  768,  788,  808,  828,  848,  869,  889,
   910,  931,  952,  974,  995, 1017, 1039, 1060,
  1083, 1105, 1127, 1150, 1172, 1195, 1218, 1241,
  1264, 1288, 1311, 1334, 1358, 1382, 1406, 1429,
  1453, 1478, 1502, 1526, 1550, 1575, 1599, 1624,
  1648, 1673, 1698, 1723, 1747, 1772, 1797, 1822,
  1847, 1872, 1897, 1922, 1948, 1973, 1998, 2023
};

const PROGMEM uint16_t DACLookup_FullSine_8Bit[256] =
{
  2048, 2098, 2148, 2198, 2248, 2298, 2348, 2398,
  2447, 2496, 2545, 2594, 2642, 2690, 2737, 2784,
  2831, 2877, 2923, 2968, 3013, 3057, 3100, 3143,
  3185, 3226, 3267, 3307, 3346, 3385, 3423, 3459,
  3495, 3530, 3565, 3598, 3630, 3662, 3692, 3722,
```

```
  3750, 3777, 3804, 3829, 3853, 3876, 3898, 3919,
  3939, 3958, 3975, 3992, 4007, 4021, 4034, 4045,
  4056, 4065, 4073, 4080, 4085, 4089, 4093, 4094,
  4095, 4094, 4093, 4089, 4085, 4080, 4073, 4065,
  4056, 4045, 4034, 4021, 4007, 3992, 3975, 3958,
  3939, 3919, 3898, 3876, 3853, 3829, 3804, 3777,
  3750, 3722, 3692, 3662, 3630, 3598, 3565, 3530,
  3495, 3459, 3423, 3385, 3346, 3307, 3267, 3226,
  3185, 3143, 3100, 3057, 3013, 2968, 2923, 2877,
  2831, 2784, 2737, 2690, 2642, 2594, 2545, 2496,
  2447, 2398, 2348, 2298, 2248, 2198, 2148, 2098,
  2048, 1997, 1947, 1897, 1847, 1797, 1747, 1697,
  1648, 1599, 1550, 1501, 1453, 1405, 1358, 1311,
  1264, 1218, 1172, 1127, 1082, 1038,  995,  952,
   910,  869,  828,  788,  749,  710,  672,  636,
   600,  565,  530,  497,  465,  433,  403,  373,
   345,  318,  291,  266,  242,  219,  197,  176,
   156,  137,  120,  103,   88,   74,   61,   50,
    39,   30,   22,   15,   10,    6,    2,    1,
     0,    1,    2,    6,   10,   15,   22,   30,
    39,   50,   61,   74,   88,  103,  120,  137,
   156,  176,  197,  219,  242,  266,  291,  318,
   345,  373,  403,  433,  465,  497,  530,  565,
   600,  636,  672,  710,  749,  788,  828,  869,
   910,  952,  995, 1038, 1082, 1127, 1172, 1218,
  1264, 1311, 1358, 1405, 1453, 1501, 1550, 1599,
  1648, 1697, 1747, 1797, 1847, 1897, 1947, 1997
};

const PROGMEM uint16_t DACLookup_FullSine_7Bit[128] =
{
  2048, 2148, 2248, 2348, 2447, 2545, 2642, 2737,
  2831, 2923, 3013, 3100, 3185, 3267, 3346, 3423,
  3495, 3565, 3630, 3692, 3750, 3804, 3853, 3898,
  3939, 3975, 4007, 4034, 4056, 4073, 4085, 4093,
  4095, 4093, 4085, 4073, 4056, 4034, 4007, 3975,
  3939, 3898, 3853, 3804, 3750, 3692, 3630, 3565,
  3495, 3423, 3346, 3267, 3185, 3100, 3013, 2923,
  2831, 2737, 2642, 2545, 2447, 2348, 2248, 2148,
  2048, 1947, 1847, 1747, 1648, 1550, 1453, 1358,
  1264, 1172, 1082,  995,  910,  828,  749,  672,
   600,  530,  465,  403,  345,  291,  242,  197,
   156,  120,   88,   61,   39,   22,   10,    2,
     0,    2,   10,   22,   39,   61,   88,  120,
   156,  197,  242,  291,  345,  403,  465,  530,
   600,  672,  749,  828,  910,  995, 1082, 1172,
  1264, 1358, 1453, 1550, 1648, 1747, 1847, 1947
};

const PROGMEM uint16_t DACLookup_FullSine_6Bit[64] =
{
  2048, 2248, 2447, 2642, 2831, 3013, 3185, 3346,
  3495, 3630, 3750, 3853, 3939, 4007, 4056, 4085,
  4095, 4085, 4056, 4007, 3939, 3853, 3750, 3630,
  3495, 3346, 3185, 3013, 2831, 2642, 2447, 2248,
  2048, 1847, 1648, 1453, 1264, 1082,  910,  749,
   600,  465,  345,  242,  156,   88,   39,   10,
     0,   10,   39,   88,  156,  242,  345,  465,
   600,  749,  910, 1082, 1264, 1453, 1648, 1847
```

```
};

const PROGMEM uint16_t DACLookup_FullSine_5Bit[32] =
{
  2048, 2447, 2831, 3185, 3495, 3750, 3939, 4056,
  4095, 4056, 3939, 3750, 3495, 3185, 2831, 2447,
  2048, 1648, 1264,  910,  600,  345,  156,   39,
     0,   39,  156,  345,  600,  910, 1264, 1648
};
void setup(void) {
  Serial.begin(9600);
  dac.begin(0x62);
  Serial.println("Generating a sine wave");
}
void loop(void) {
    uint16_t i;
    // Push out the right lookup table, depending on the selected
resolution
    #if DAC_RESOLUTION == 5
      for (i = 0; i < 32; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine_5Bit[i])),
false);
      }
    #elif DAC_RESOLUTION == 6
      for (i = 0; i < 64; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine_6Bit[i])),
false);
      }
    #elif DAC_RESOLUTION == 7
      for (i = 0; i < 128; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine_7Bit[i])),
false);
      }
    #elif DAC_RESOLUTION == 9
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine_9Bit[i])),
false);
      }
    #else    // Use 8-bit data if nothing else is specified
      for (i = 0; i < 256; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine_8Bit[i])),
false);
      }
    #endif
}
```

- Se va modifica programul prezentat astfel incat sa se poata varia: Amplitudinea, rezolutia si frecventa semnalului sinusoidal.