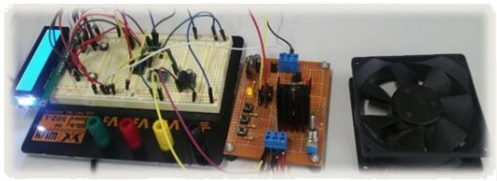
UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
FACULTATEA DE INGINERIE ELECTRICĂ

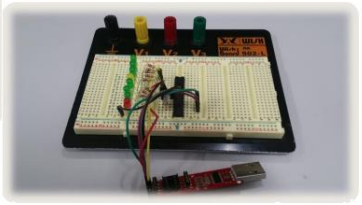
SISTEME CU MICROPROCESOARE

Ședința de laborator nr. 4

Aplicații generale de interfațare

SISTEME EMBEDDED ÎN INGINERIE ELECTRICĂ
GHID DE APLICAȚII





epe.utcluj.ro

https://epe.utcluj.ro/index.php/sisteme-cu-microprocesoare/
Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

Cuprins

1. Scopul lucrării

Lucrarea de laborator are ca scop:

- ✓ Prezentarea conceptului „aplicație de interfațare” [1] [2]
- ✓ Prezentarea metodelor și procedeelor pentru optimizarea unei aplicații [1] [2]
- ✓ Prezentarea modului de funcționare al unui senzor ultrasonic [3]
- ✓ Prezentarea modului de funcționare al unui afișaj LCD pe interfață I2C [4] [5]
- ✓ Implementarea unor aplicații de interfațare [1] [2] [3] [4] [5]

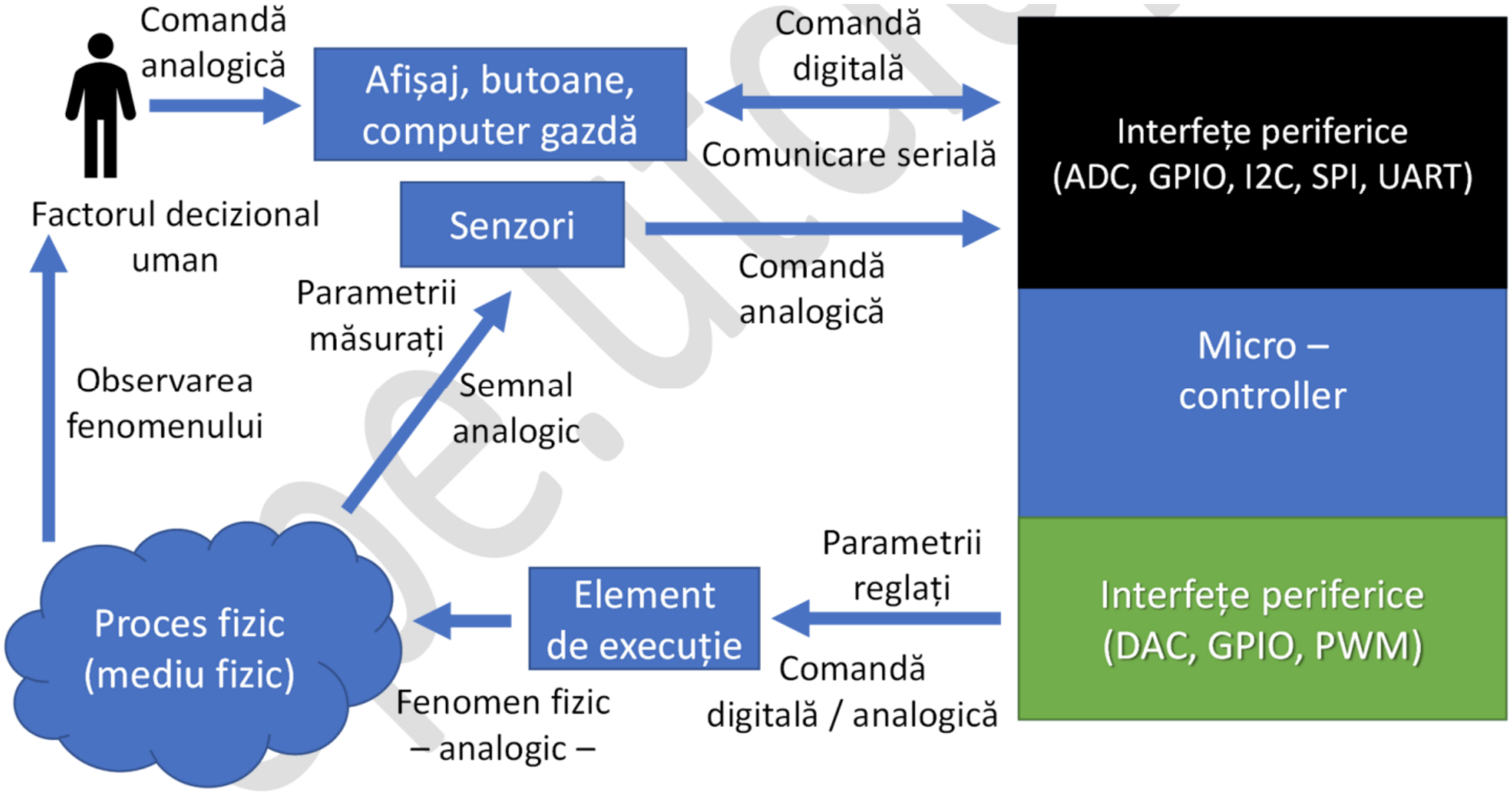
2. Introducere

- Conceptul de **aplicație** reprezintă orice formă de **implementare a unui circuit electronic** atât pe baza **dispozitivelor periferice specializate** ale microcontrolerelor cât și pe baza **altor sisteme de calcul specializate** în acest sens (ex. comunicare cu calculatorul gazdă) [1] [2]

- **Perifericele de intrare și ieșire** ale microcontrolerului pot fi atașate la **elemente de circuit** precum: afișaje LCD, butoane, senzori sau elemente de execuție precum motoare, rezistențe sau semnalizatoare acustice

- **Rolul dispozitivului** rezultat în urma implementării, este de a **intermedia** operația de reglare a procesul tehnologic final realizată de **factorul uman**. Funcția îndeplinită în acest sens se numește **interfațare**

Funcția de interfațare



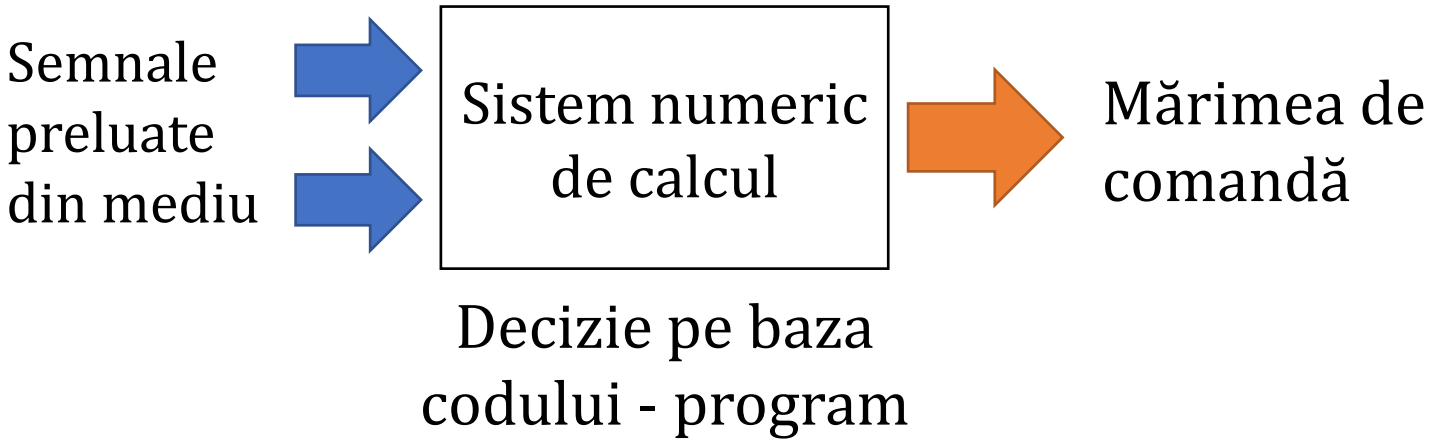
3. Aspecte teoretice

- **Există trei tipuri de aplicații** [1] [2]:

  - ✓ Aplicații independente (cu algoritm de condiționare proprie)
  - ✓ Aplicații dependente de elementele de interfațare (LCD și butoane sau calculator)
  - ✓ Aplicații hibride (comandate de utilizator dar cu algoritm de condiționare propriu)

3. Aspecte teoretice

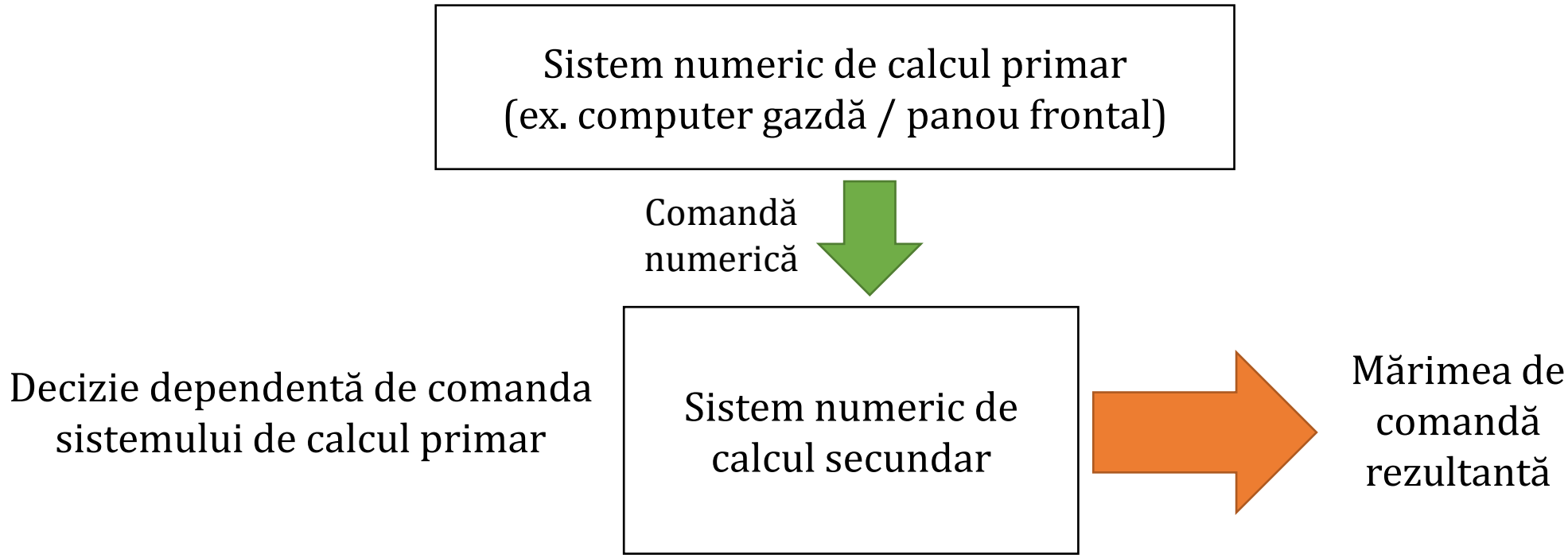
- **Aplicații independente, cu decizie bazată direct pe datele colectate de la senzori:**
  - ✓ În această categorie se încadrează aplicații precum: detectoare, sesizoare, elemente de semnalizare sau orice alte dispozitive cu logică de comandă și control prestabilite și impuse de către producător, asupra căreia utilizatorul nu poate interveni cu modificări.



3. Aspecte teoretice

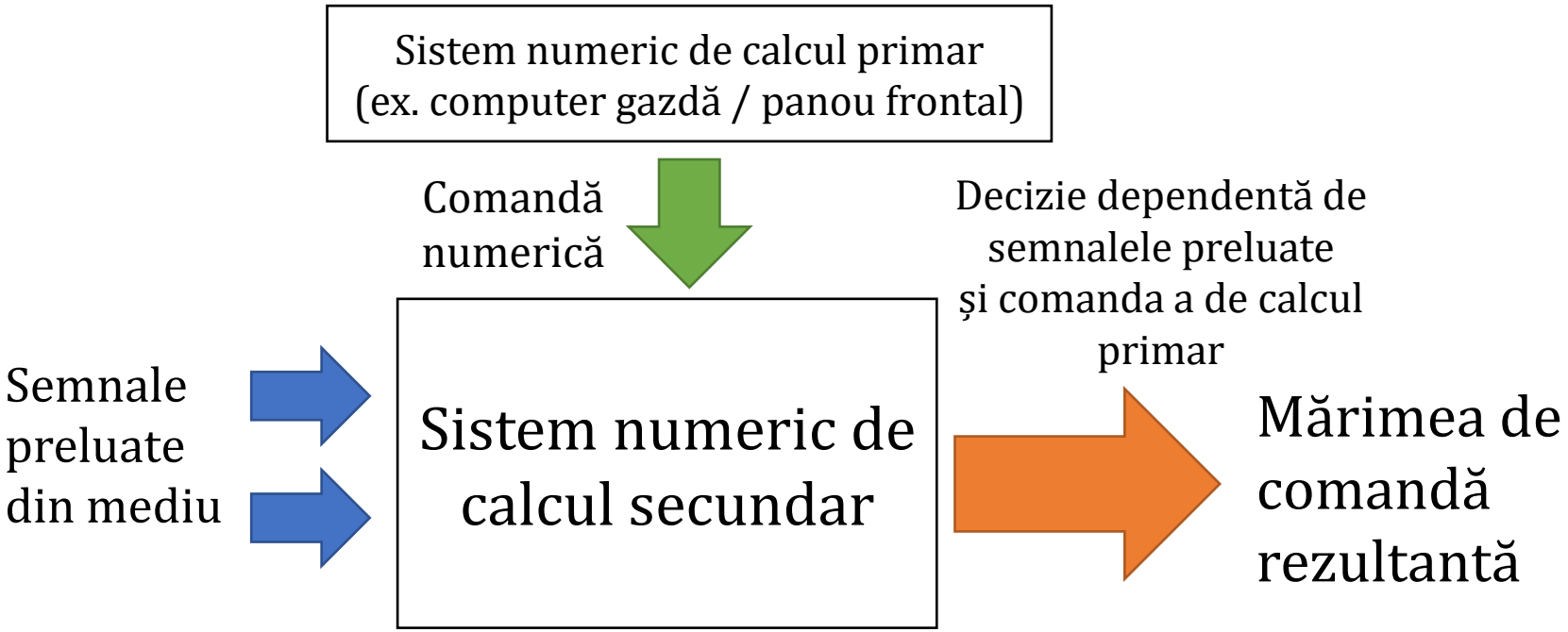
❖ Aplicații dependente de elementele de interfațare:

- ✓ Câteva exemple de aplicații din această categorie sunt: dispozitivele periferice ale computerului, panouri frontale cu afișaj și butoane, echipamente de interacțiune umană cu procesul fizic (Human Interface Device – HID), precum termostatul centralei termice.



3. Aspecte teoretice

- **Aplicații hibride (comandate de utilizator, cu decizie asupra algoritmului propriu):**
  - ✓ Câteva exemple din această categorie pot fi dispozitivele complexe care pot fi configurate și asistate de la distanță pentru a deservi un proces fizic: termostate digitale programabile atât de la distanță cât și prin intermediul panoului frontal propriu, ventilatoarele industriale comandate de la distanță dar care au și posibilitatea ajustării în funcție de datele preluate de la senzori, imprimante multifuncționale etc.

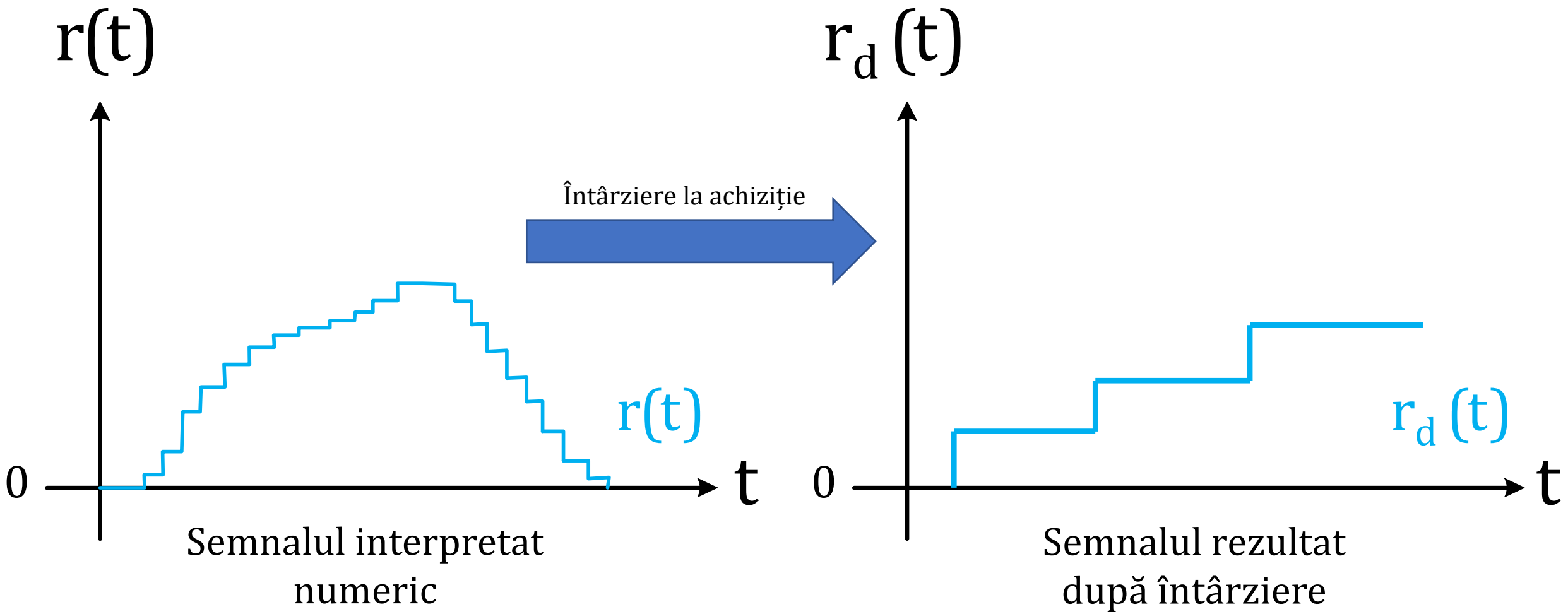


4. Implementarea aplicațiilor

- **Există diverse metode și procedee utilizate în vederea îmbunătățirii modului de funcționare al aplicației concepute, precum:**

  - ✓ Introducerea unui timp de întârziere prin intermediul funcției „delay()”
  - ✓ Introducerea unor constrângeri condiționale pentru a limita un interval de variație
  - ✓ Introducerea unui algoritm de mediere în timp a semnalului

Introducerea timpului de întârziere



Semnalul interpretat numeric

Semnalul rezultat după întârziere

Constrângerea condițională simplă

- Pentru o variabilă oarecare în intervalul [constantă_1 constantă_2]:
- Exemplu de constrângere simplă inferioară:

```
if (variabilă <= constantă_1) {
    variabilă = constantă_1;
}
```

- Exemplu de constrângere simplă superioară:

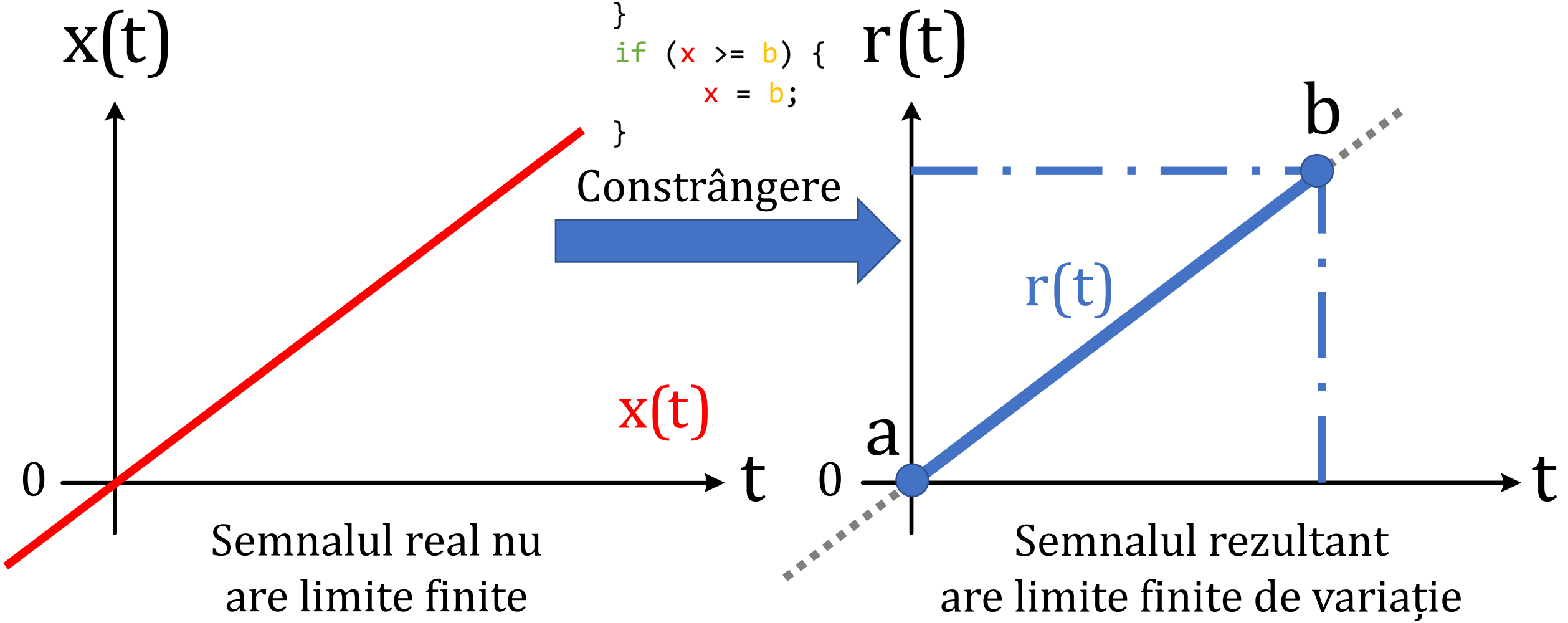
```
if (variabilă >= constantă_2) {
    variabilă = constantă_2;
}
```

- O metodă alternativă ar fi: constrain(variabilă, constantă_1, constantă_2);

Constrângerea condițională simplă (mărginire)

```
if (x <= a) {
    x = a;
}
if (x >= b) {
    x = b;
}
```

Constrângere



Constrângere prin interpolare liniară

❖ Exemplu: Fie variabila „p” cu valori în intervalul [0 1023]. Să se scrie funcția necesară pentru care, variabila „q” variază direct - proporțional cu variabila „p” în intervalul [0 255]:

- Forma generală:

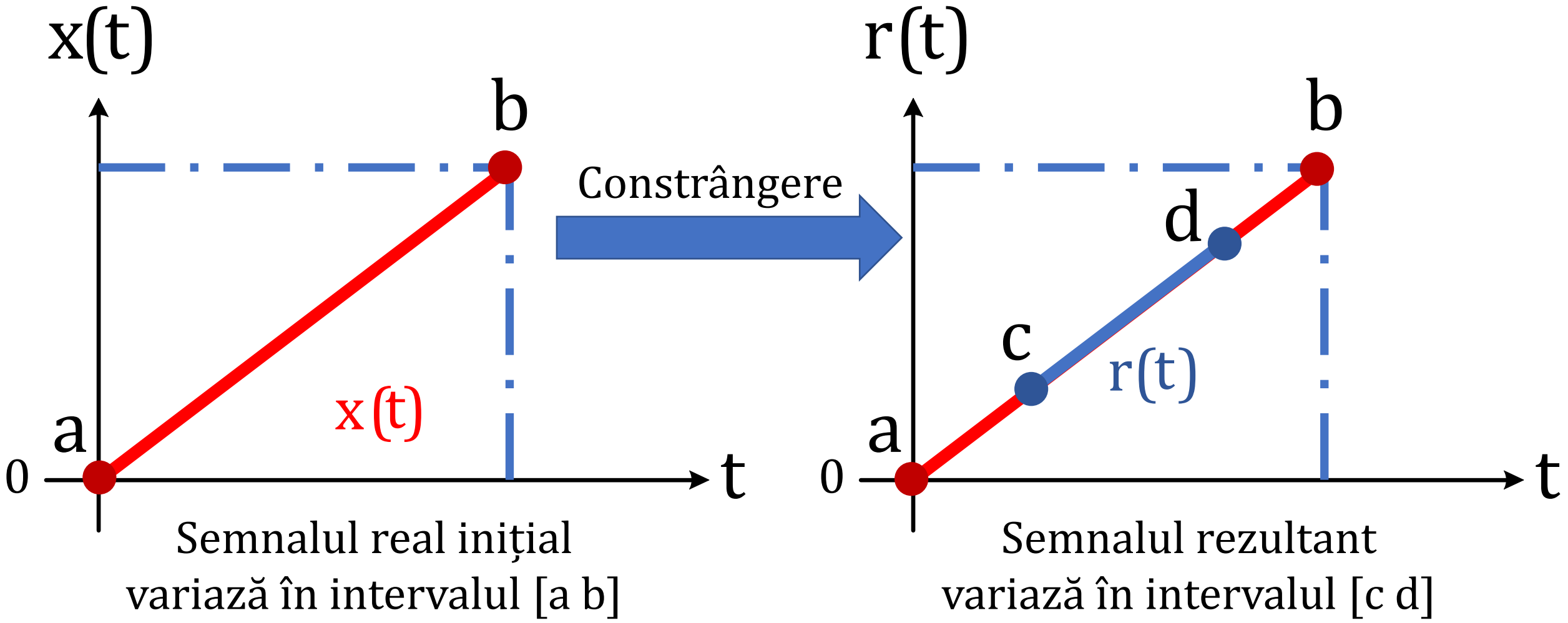
variabila_2 = map (variabila_1, valoare_inferioară_1, valoare_superioară_1, valoare_inferioară_2, valoare_superioară_2);

- Rezultatul:

```
q = map (p, 0, 1023, 0, 255);
```

Constrângere prin interpolare liniară (scalare)

```
r = map (x, a, b, c, d)
```



Semnalul real inițial
variază în intervalul [a b]

Semnalul rezultant
variază în intervalul [c d]

Algoritmul de mediere

❖ **Algoritmul de mediere se implementează cu ajutorul unei bucle iterative „for”:**

```
int suma = 0;
for (int i = 1; i <= N; i++) {
    x = analogRead (pin_analogic);
    suma += x;
}
X = suma / N;
```

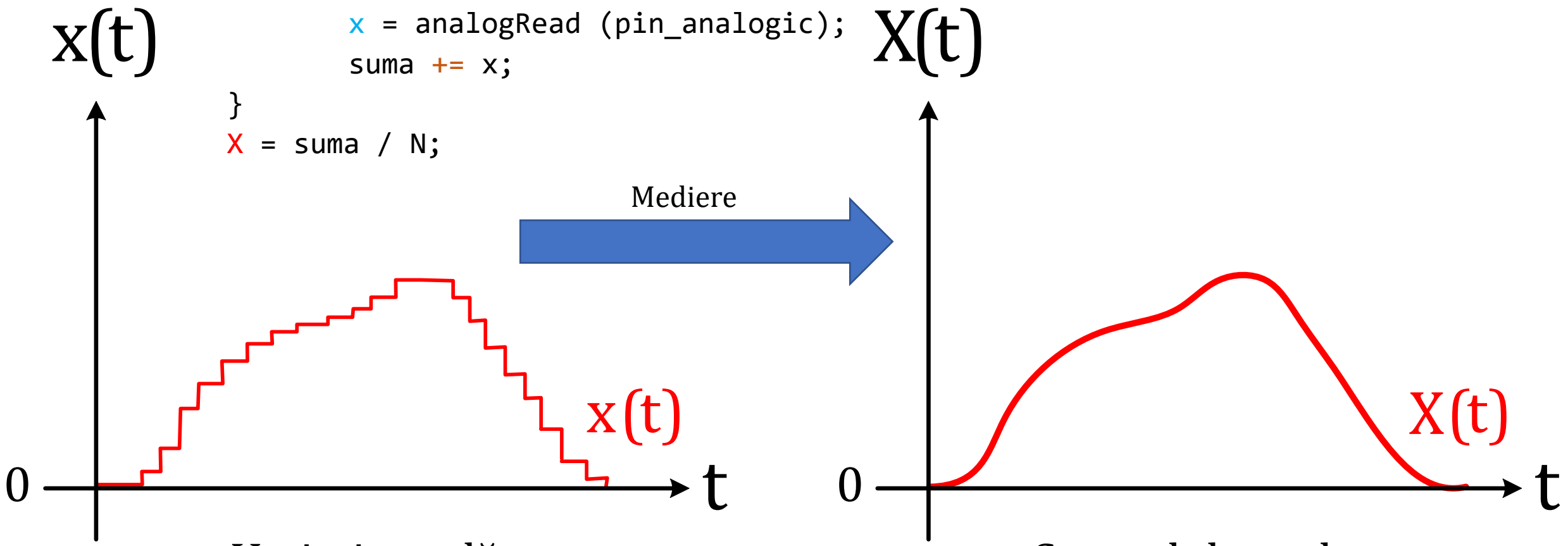
OBSERVAȚII:

1. Algoritmul se implementează în bucla infinită „void loop()”;

2. Variabila „suma” trebuie inițializată cu valoarea zero la începutul buclei infinite „void loop()”, pentru a realiza re-inițializarea algoritmului pentru fiecare ciclu.

Algoritmul de mediere

```
int suma = 0;
for (int i = 1; i <= N; i++) {
        x = analogRead (pin_analogic);
        suma += x;
}
X = suma / N;
```



Variația reală a semnalului analogic

Semnalul rezultant în urma medierii

4. Implementarea aplicațiilor

- **Se propune implementarea a șase aplicații:**

1. Controlul digital al factorului de umplere
2. Preluarea temperaturii de la un traductor și filtrarea zgomotelor
3. Măsurarea distanței cu ajutorul traductorului ultrasonic HC – SR04
4. Utilizarea unui afișaj de tip LCD
5. Implementarea unui termometru digital
6. Implementarea unei rigle digitale

4. Implementarea aplicațiilor

➢ **Echipamentele și componentele electronice necesare în vederea implementării aplicațiilor propuse sunt**:

- ✓ placă pentru testare rapidă a circuitelor electronice (Wisher WBU-502L);
- ✓ platformă de dezvoltare Arduino NANO cu microcontroler ATMega 328;
- ✓ diode electro-luminiscente;
- ✓ butoane cu apăsare și revenire;
- ✓ rezistențe cu valoarea de 100 [Ω];
- ✓ rezistențe cu valoarea de 10 [kΩ];
- ✓ senzor de temperatura LM-35;
- ✓ traductor de distanță pe bază de efect ultrasonic HC-SR04;
- ✓ modulul pentru afișare LCD QAPASS cu adaptor I2C la Paralel M.H.;
- ✓ fire pentru conexiune rapidă compatibile cu placa de testare;
- ✓ calculator gazdă având mediul Arduino IDE instalat;
- ✓ cablu adaptor USB A la mini USB;

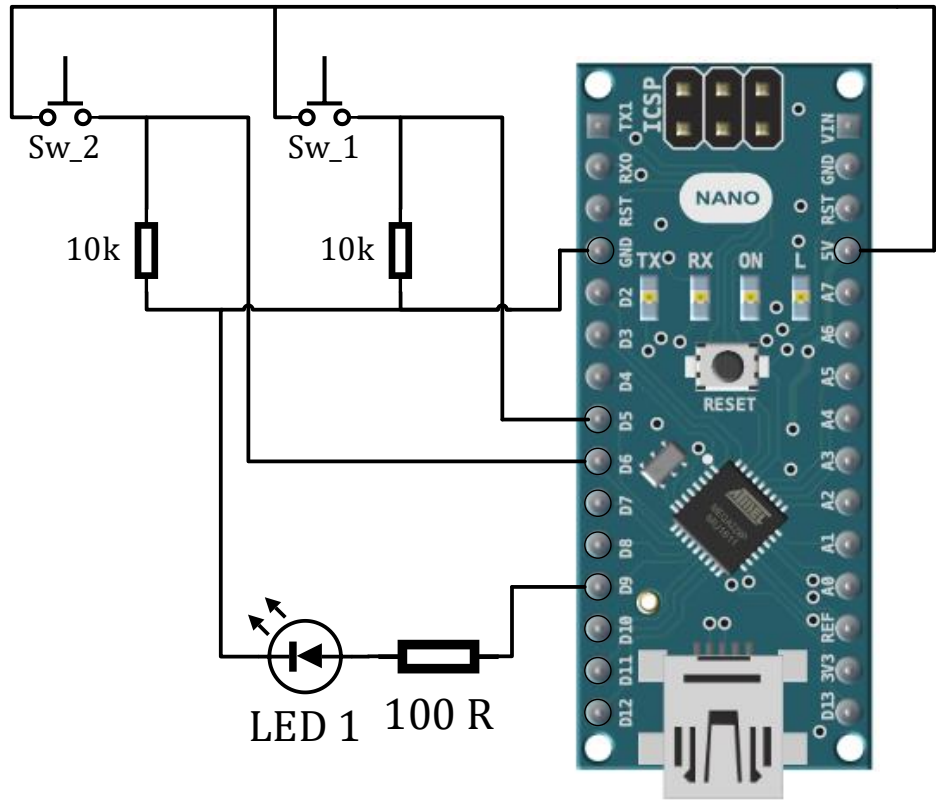
4. Implementarea aplicațiilor – Aplicația nr. 1

➢ Implementarea aplicației nr. 1 presupune:

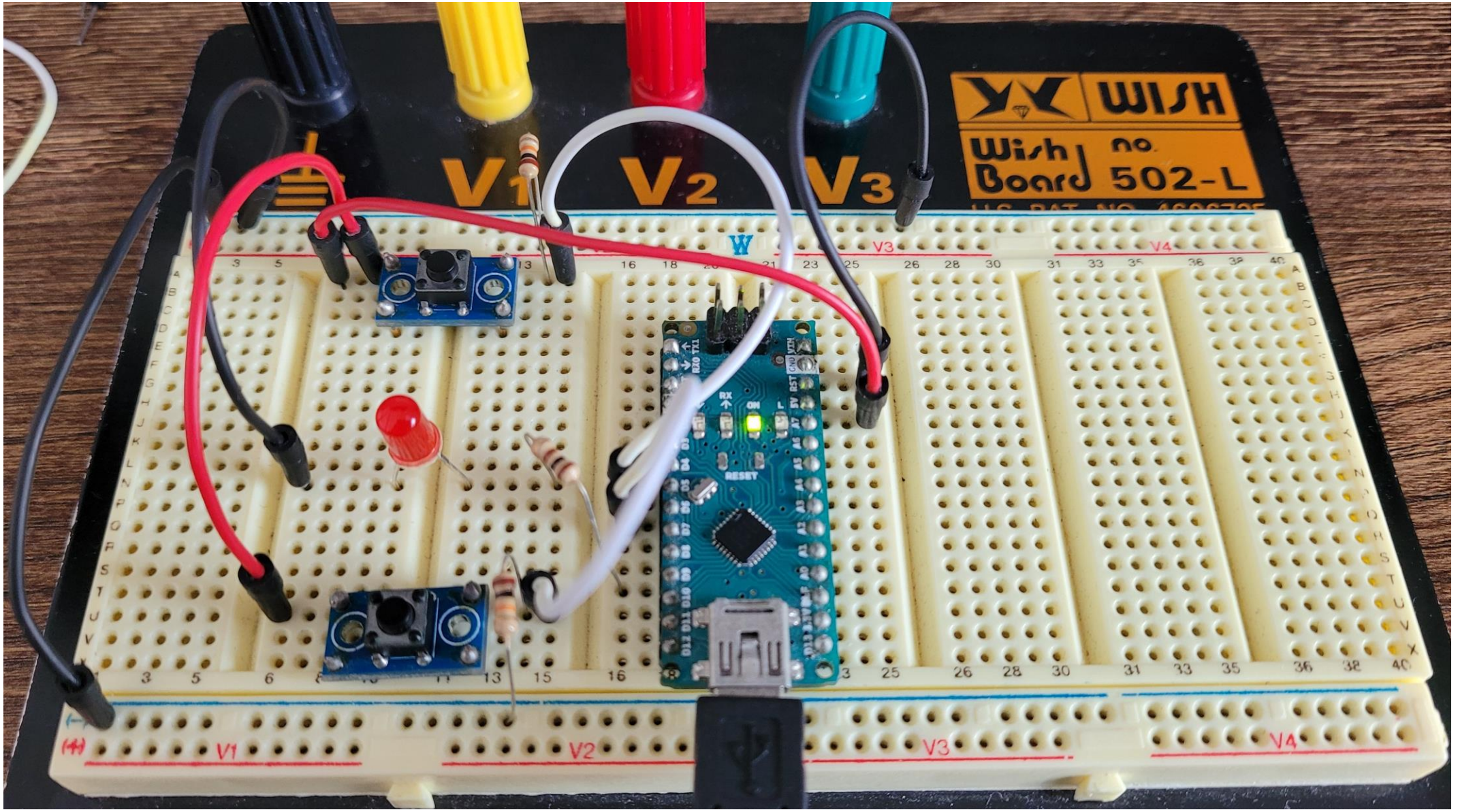
- ✓ declararea a trei constante globale, în vederea stabilirii numerelor de ordine ale terminalelor corespondente atât butoanelor cu apăsare și revenire cât și diodei (LED);
- ✓ stabilirea modului de lucru „intrare digitală” pentru terminalele butoanelor;
- ✓ stabilirea modului de lucru „ieșire digitală” pentru terminalul diodei (LED);
- ✓ preluarea stărilor logice ale terminalelor alese pentru atașarea butoanelor cu apăsare și revenire prin intermediul instrucțiunii „digitalRead ()”;
- ✓ incrementarea sau decrementarea factorului de umplere „dc” la menținerea apăsată sau apăsarea repetată a butoanelor;
- ✓ impunerea unor constrângeri simple;
- ✓ afișarea în consola Serial a valorii pentru reglarea factorului de umplere (Fig. 15 / 16);
- ✓ generarea unui semnal modulat în lățime prin intermediul funcției „analogWrite()”;

4. Implementarea aplicațiilor – Aplicația nr. 1

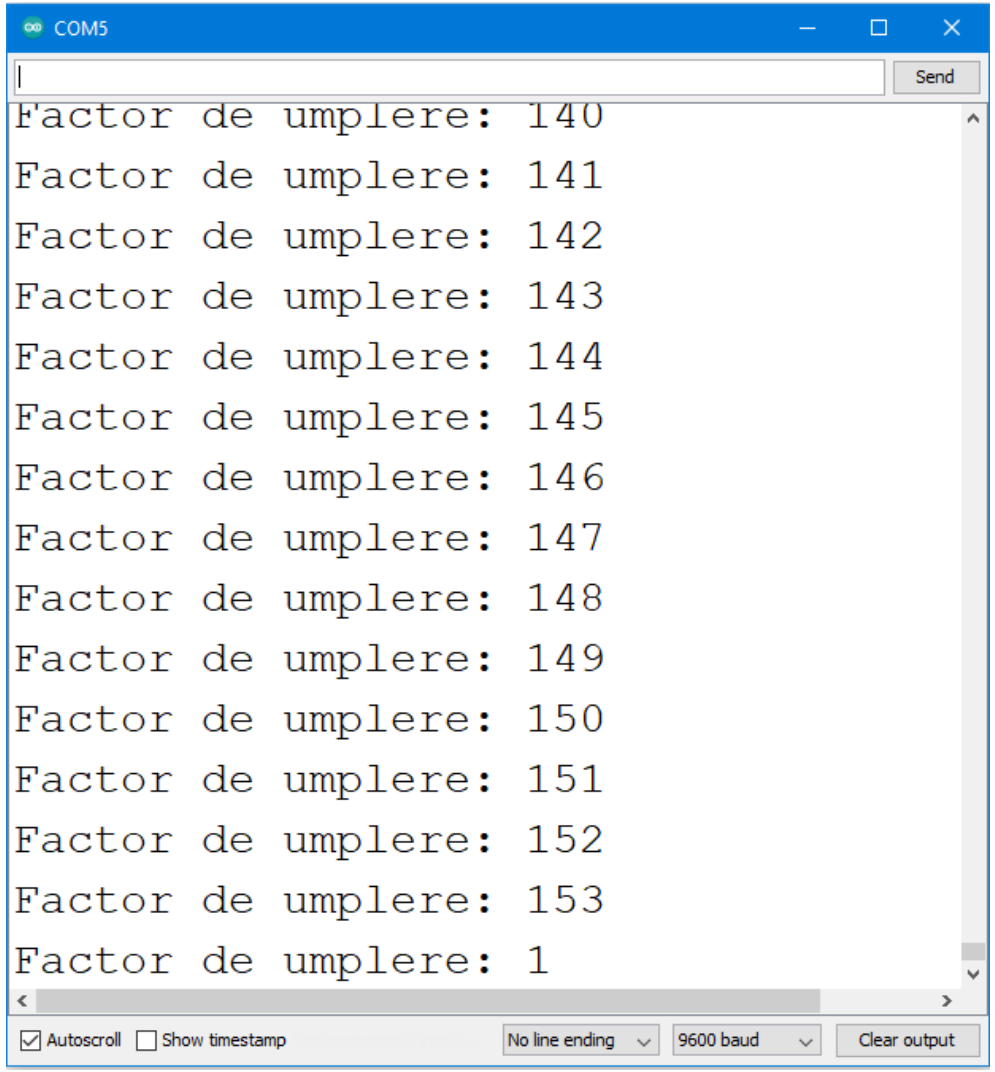
```
const int pin_sw_1 = 5;
const int pin_sw_2 = 6;
const int pin_led = 9;

int sw_1_state = 0;
int sw_2_state = 0;
int dc = 0;

void setup() {
  pinMode(pin_sw_1, INPUT);
  pinMode(pin_sw_2, INPUT);
  pinMode(pin_led, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  sw_1_state = digitalRead(pin_sw_1);
  sw_2_state = digitalRead(pin_sw_2);

  if(sw_1_state == 1) {
    dc++;
    if (dc >= 255) {
      dc = 255;
    }
  }
  if(sw_2_state == 1) {
    dc--;
    if (dc <= 0) {
      dc = 0;
    }
  }

  Serial.print("Factor de umplere: ");
  Serial.print(dc);
  Serial.println("");
  analogWrite(pin_led, dc);
  delay(10);
}
```

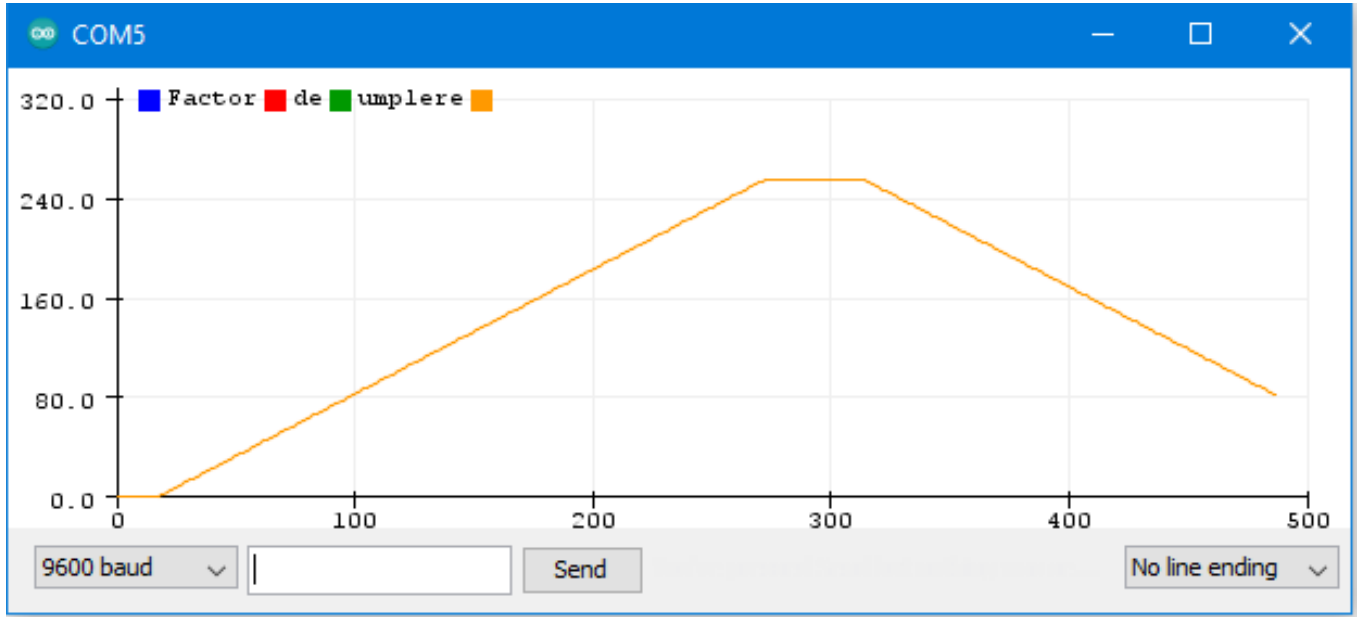


4. Implementarea aplicațiilor – Aplicația nr. 1



4. Implementarea aplicațiilor – Aplicația nr. 1





Consola Serial – Afișarea valorii zecimale a factorului de umplere

4. Implementarea aplicațiilor – Aplicația nr. 2

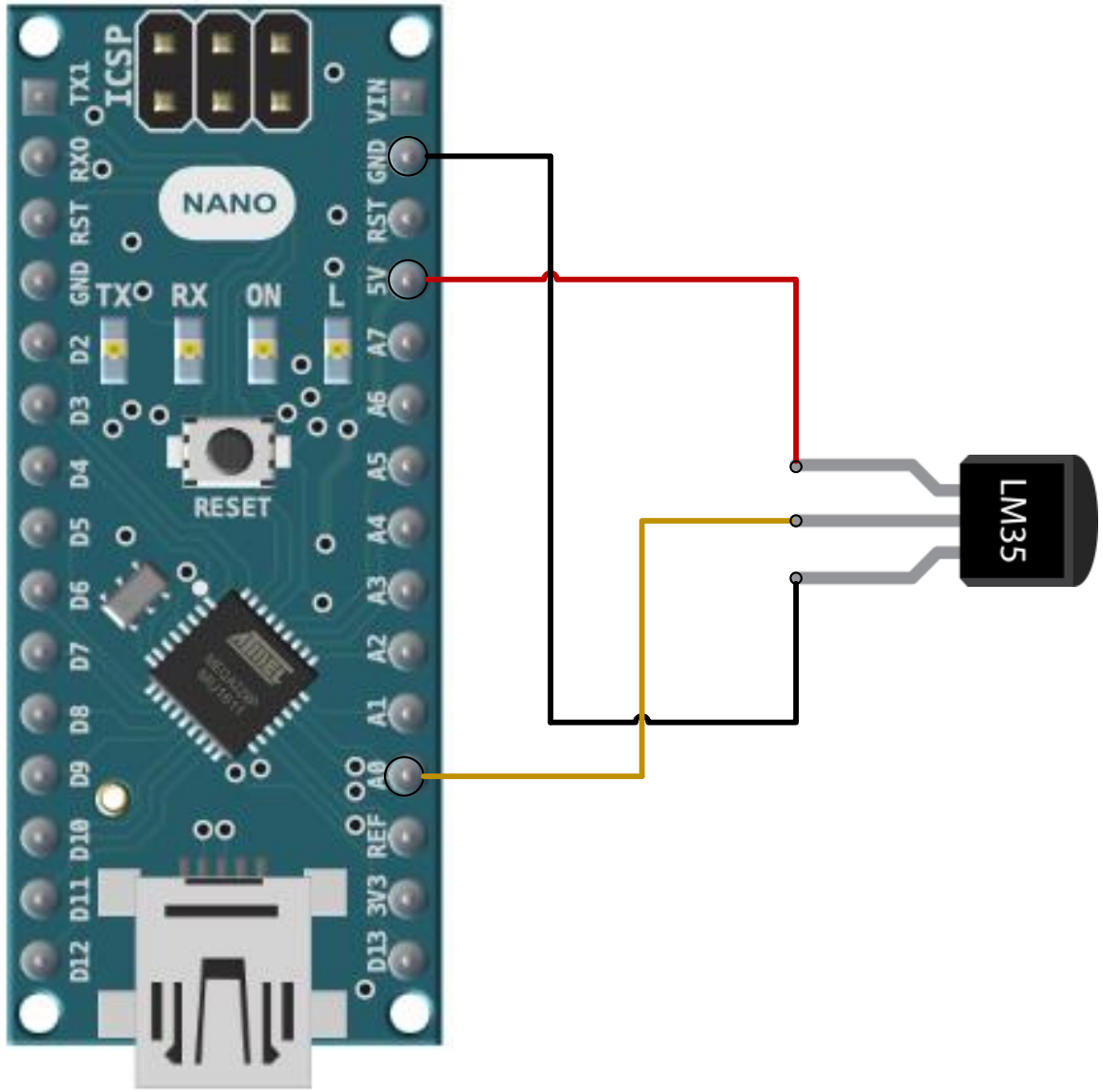
➢ **Implementarea aplicației nr. 2 presupune:**

✓ declararea unei constante de tip număr întreg „analog_pin” având ca și valoare „0”;
✓ inițializarea unei variabile de tip număr întreg „ADC_val” cu valoarea „0”;
✓ inițializarea unei variabile de tip număr întreg „N” cu valoarea „0”;
✓ inițializarea unei variabile de tip fracționar „U” cu valoarea „0.00”;
✓ inițializarea unei variabile de tip fracționar „temp” cu valoarea „0.00”;
✓ inițializarea unei variabile de tip fracționar „media” cu valoarea „0.00”;
✓ inițializarea comunicației Serial la viteza de transfer 9600 [b/s];
✓ inițializarea unei variabile locale cu denumirea „suma”;
✓ preluarea valorii zecimale rezultante în urma procesului de conversie analog – digital;
✓ determinarea tensiunii de măsură pe baza preciziei convertorului analog – digital;
✓ determinarea temperaturii pe baza tensiunii de măsură și a constantei de calibrare;
✓ însumarea a 500 de valori prin intermediul structurii iterative de tip „for ()”;
✓ determinarea mediei aritmetice prin intermediul raportului dintre suma tuturor valorilor înregistrate în variabila „suma” și numărul de iterații „N”;
✓ afișarea valorii medii a temperaturii în consola grafică;

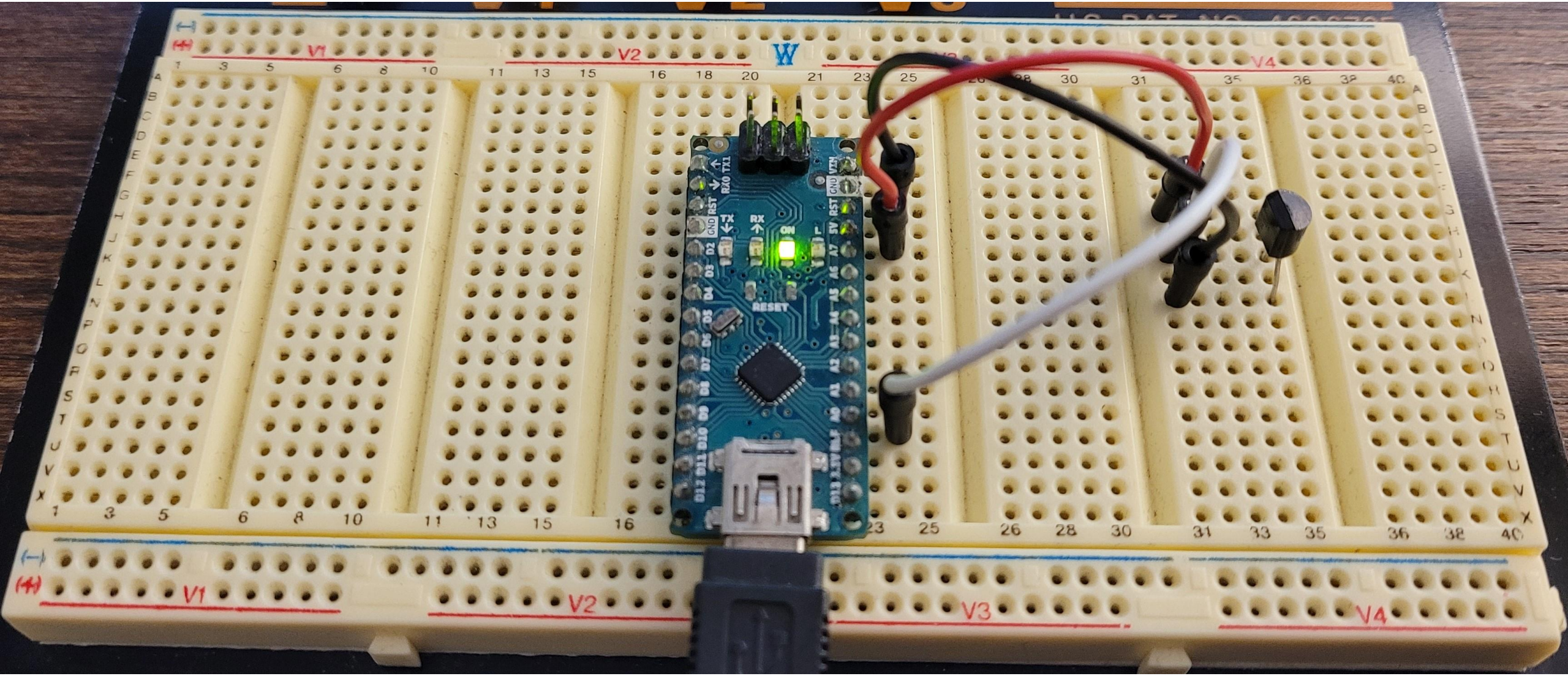
4. Implementarea aplicațiilor – Aplicația nr. 2

```
const int analog_pin = 0;
int ADC_val = 0;
int N = 500;
float U = 0.00;
float temp = 0.00;
float media = 0.00;

void setup() {
  Serial.begin(9600);
}

void loop() {
  float suma = 0.00;
  for (int i = 1; i <= N; i++) {
  ADC_val = analogRead(analog_pin);
  U = (4.80 / 1023.00) * ADC_val;
  temp = 100.00 * U;
  suma += temp;
  }
  media = suma / N;
  Serial.print("Temperatura: ");
  Serial.print(media);
  Serial.print(" [*C]");
  Serial.println("");
  delay(250);
}
```

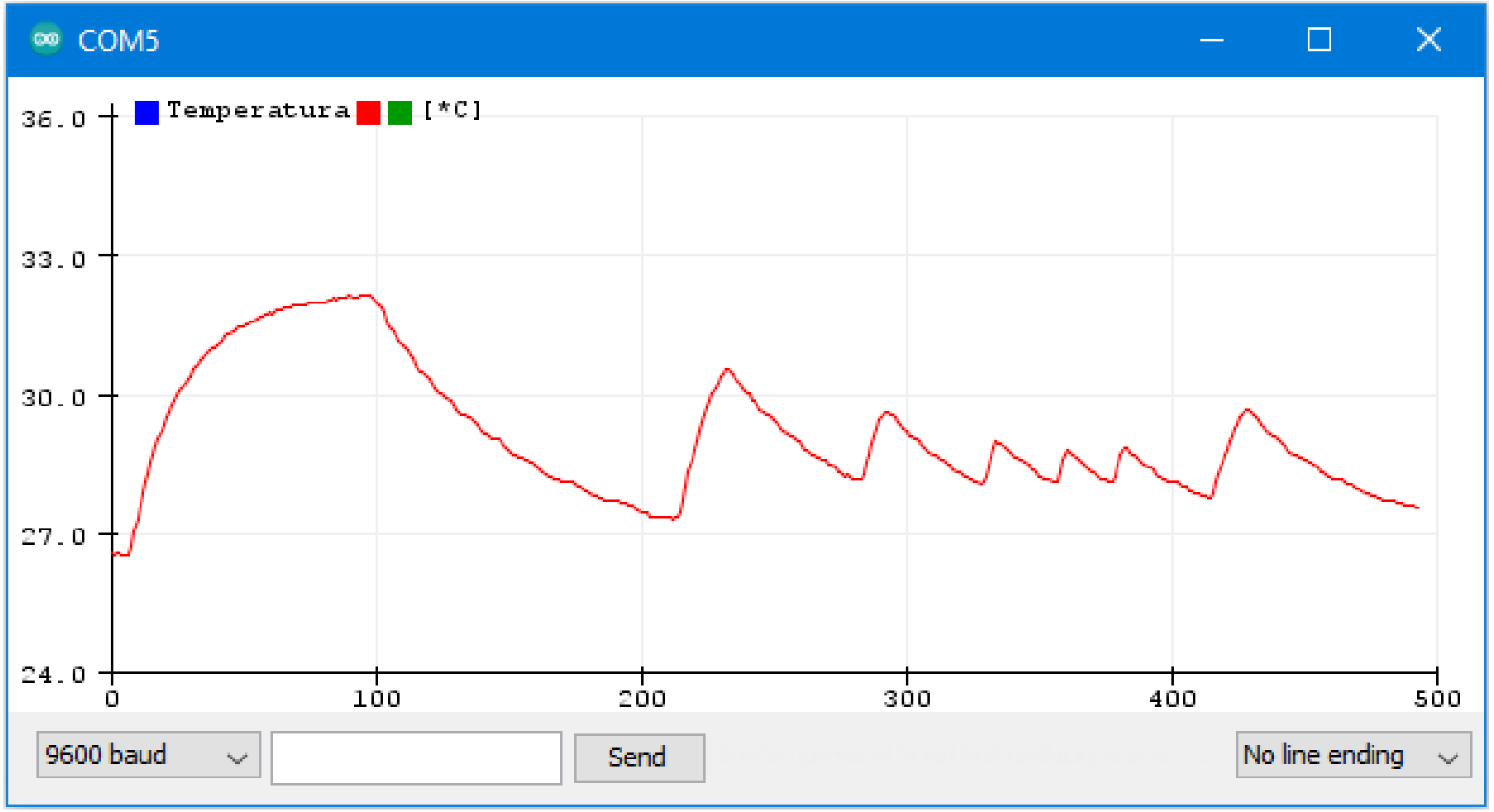
4. Implementarea aplicațiilor – Aplicația nr. 2



4. Implementarea aplicațiilor – Aplicația nr. 2



4. Implementarea aplicațiilor – Aplicația nr. 2



Consola grafică – Afișarea valorilor medii ale temperaturii

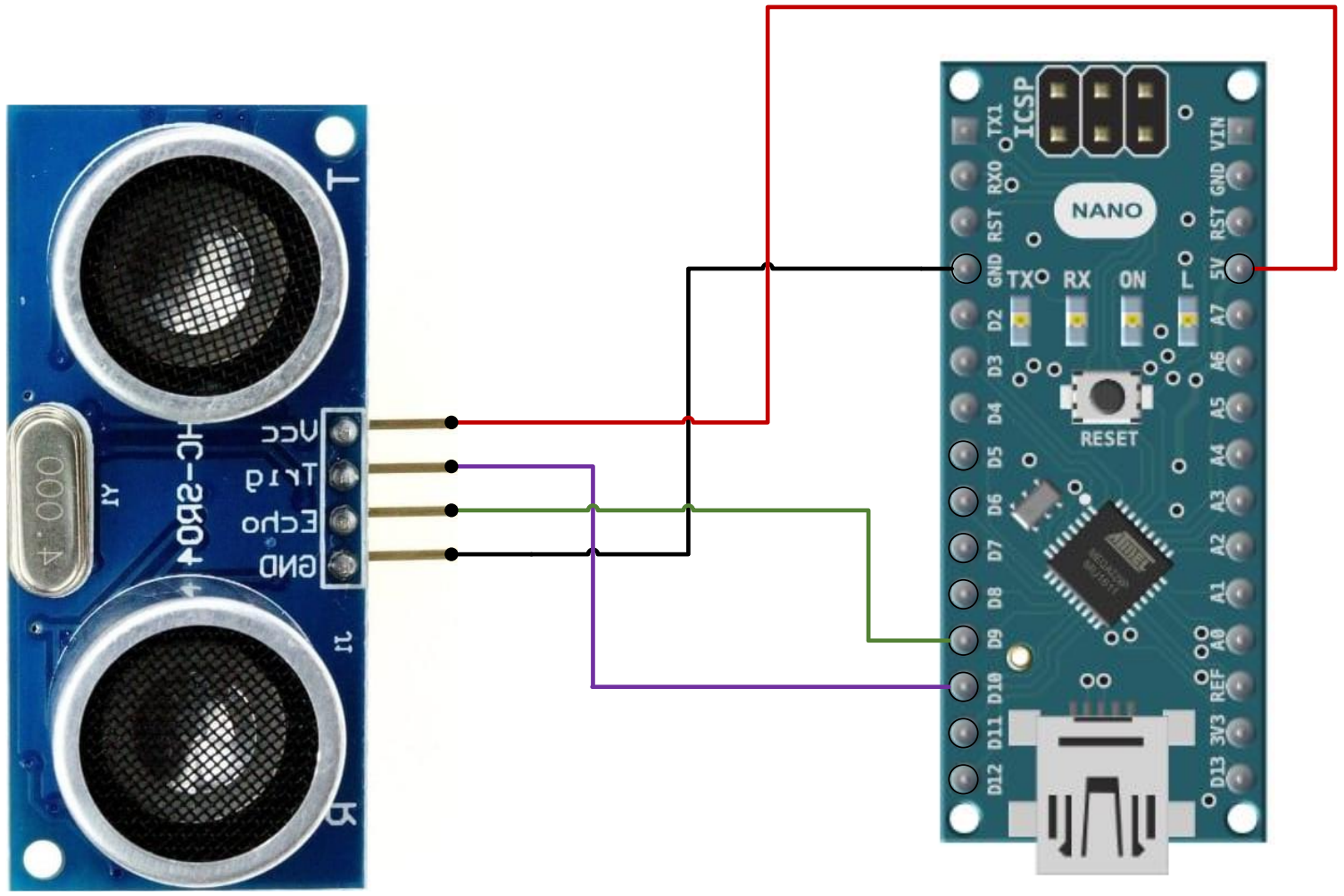
4. Implementarea aplicațiilor – Aplicația nr. 3

➢ **Implementarea aplicației nr. 3 presupune:**

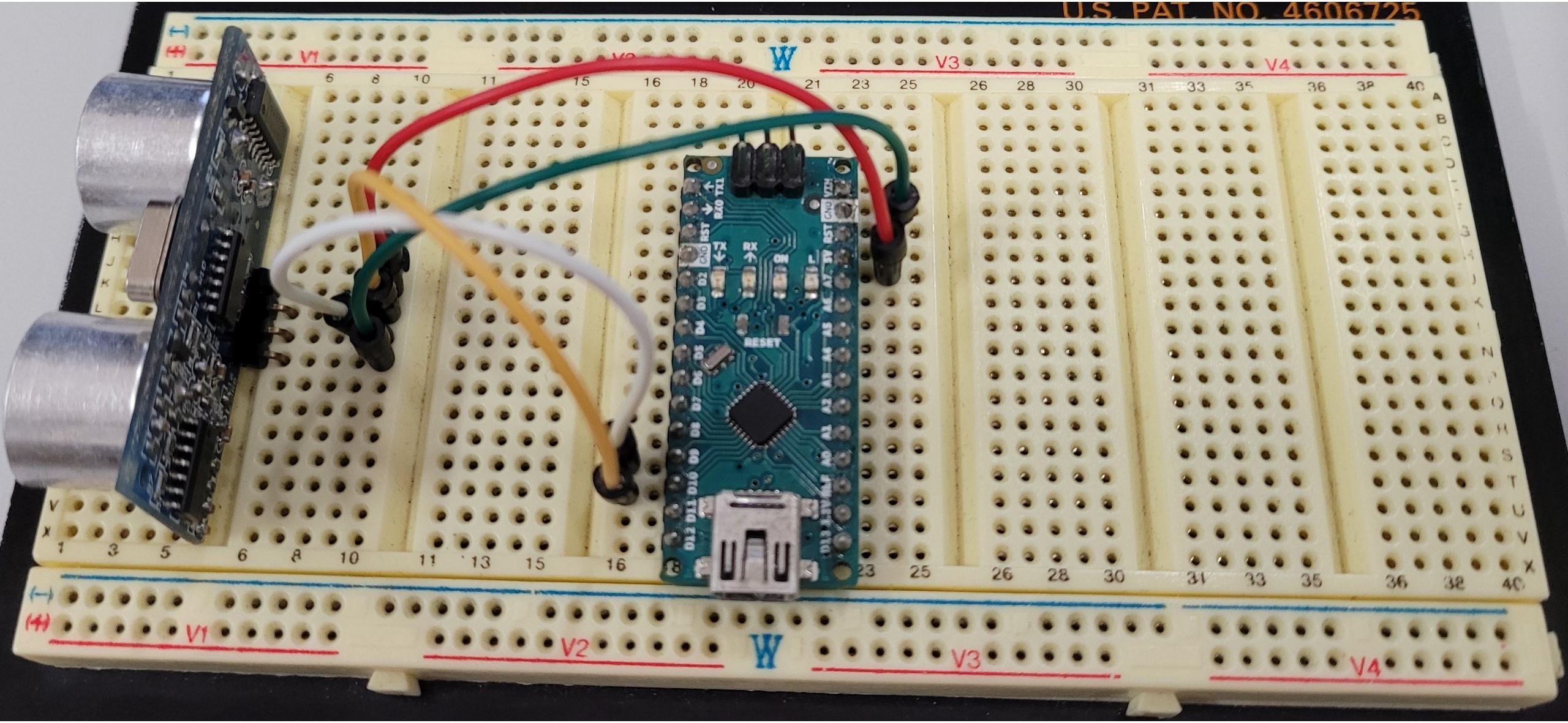
- ✓ declararea unei constante de tip număr întreg „echo” având ca și valoare „9”;
- ✓ declararea unei constante de tip număr întreg „trigger” având ca și valoare „10”;
- ✓ inițializarea unei variabile de tip număr întreg „t” cu valoarea „0”;
- ✓ inițializarea unei variabile de tip număr întreg „d” cu valoarea „0”;
- ✓ stabilirea modului de lucru „ieșire digitală” pentru terminalul „9”;
- ✓ stabilirea modului de lucru „intrare digitală” pentru terminalul „10”;
- ✓ inițializarea comunicației Serial;
- ✓ generarea unei secvențe de semnalizare acustică la o frecvență de 40 [kHz];
- ✓ determinarea timpului de propagare a undei sonore cu ajutorul funcției „pulseIn ()”;
- ✓ determinarea distanței pe baza formulei de calcul a vitezei de propagare a undei sonore;
- ✓ afișarea în consola Serial a rezultatului măsurării;

4. Implementarea aplicațiilor – Aplicația nr. 3

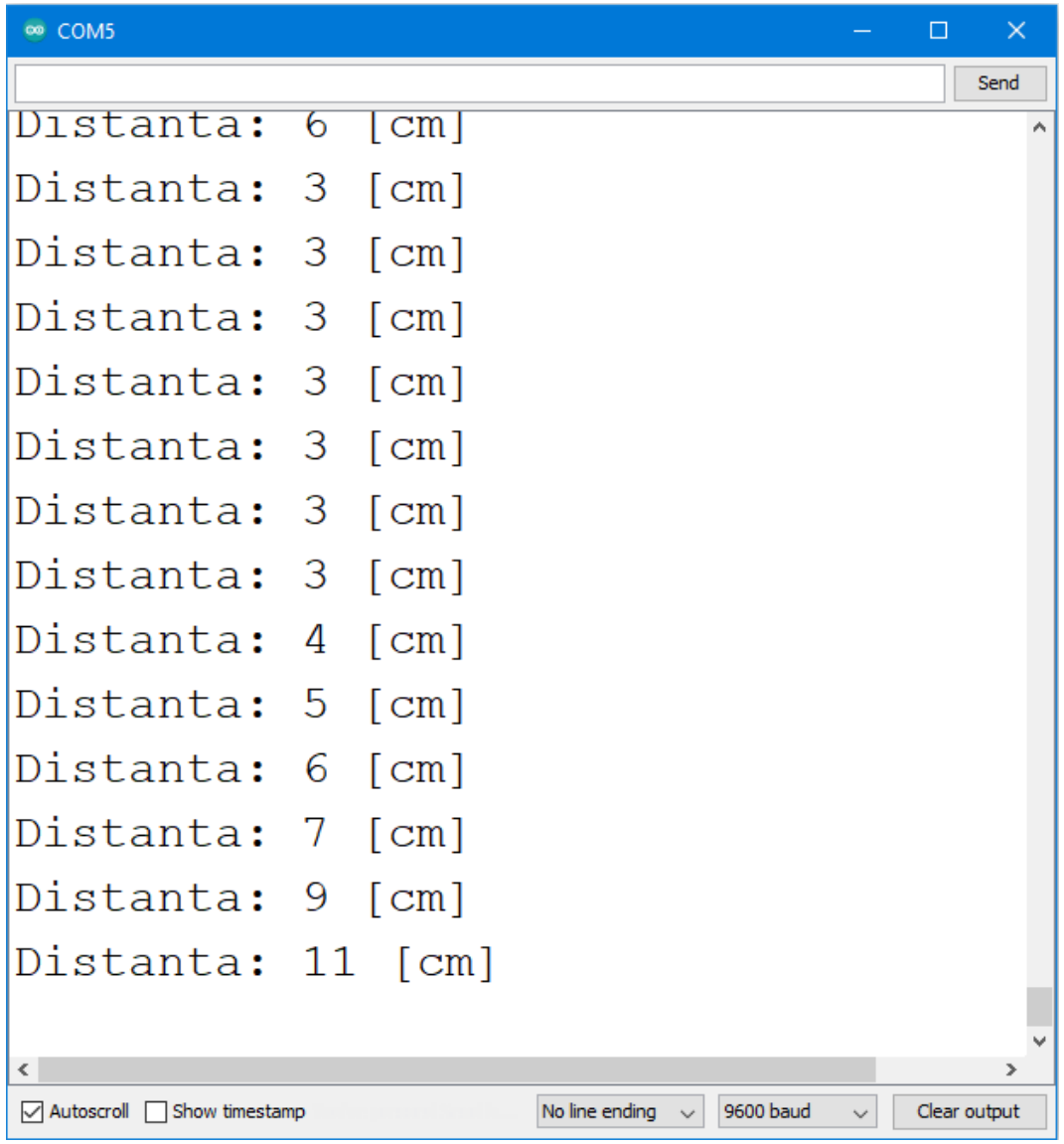
```
const int echo = 9;
const int trigger = 10;
long t = 0.0;
long d = 0.0;

void setup() {
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  t = pulseIn(echo, HIGH);
  d = t * 0.034 / 2.0;
  Serial.print("Distanta: ");
  Serial.print(d);
  Serial.print(" [cm]");
  Serial.println("");
  delay(100);
}
```



4. Implementarea aplicațiilor – Aplicația nr. 3



4. Implementarea aplicațiilor – Aplicația nr. 3



Consola Serial – Afișarea rezultatului de măsurare a distanței

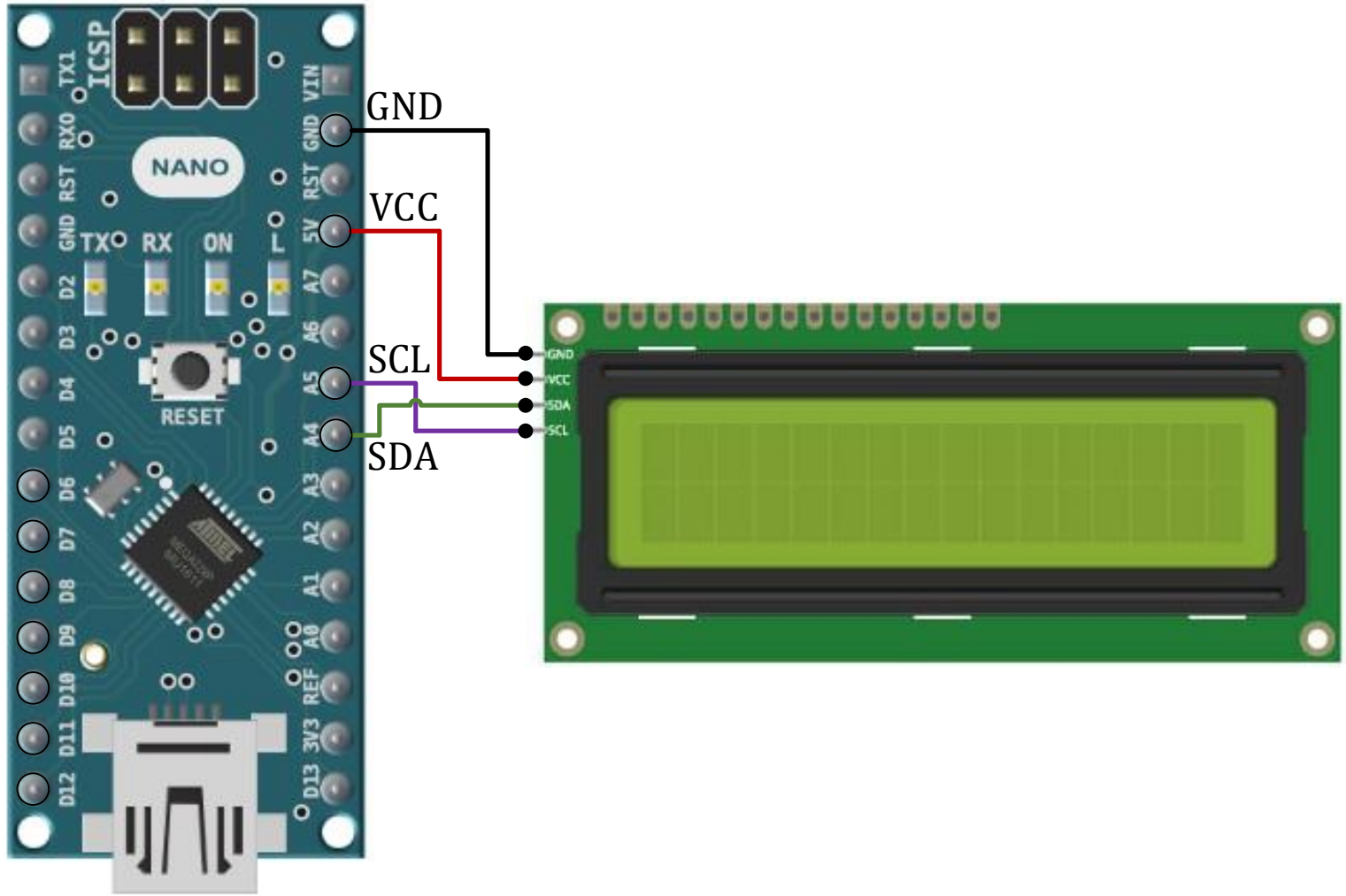
4. Implementarea aplicațiilor – Aplicația nr. 4

➢ **Implementarea aplicației nr. 4 presupune:**

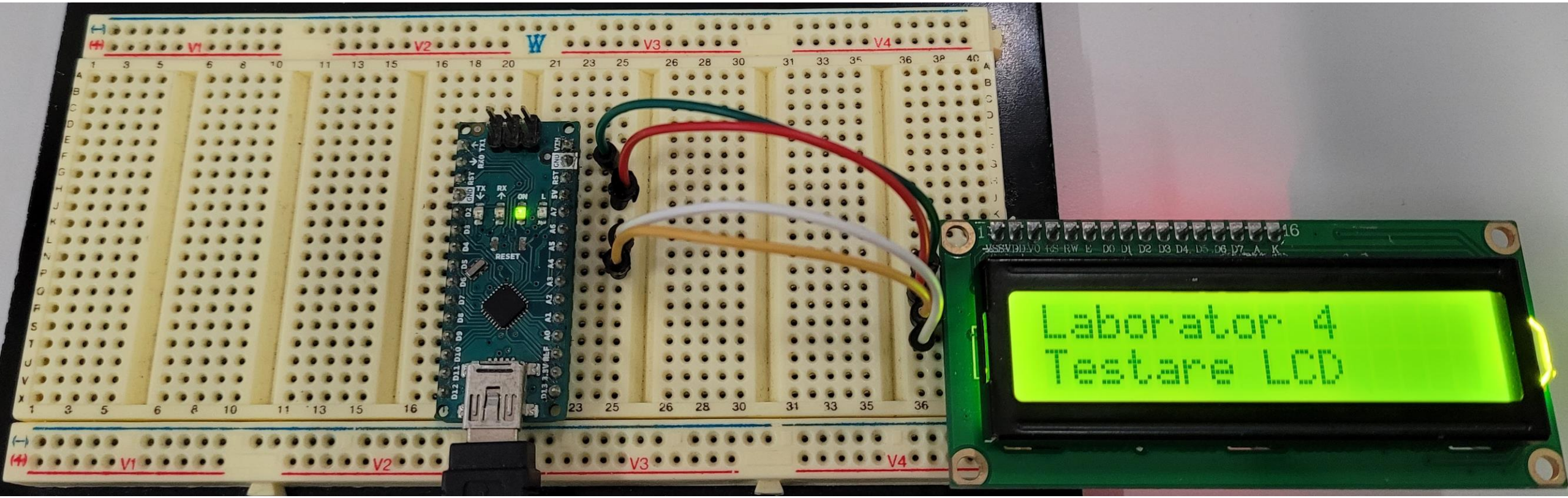
- ✓ inițializarea bibliotecii de funcții „Wire.h” pentru comunicația I2C;
- ✓ inițializarea bibliotecii de funcții „LiquidCrystal_I2C.h”;
- ✓ inițializarea parametrilor aferenți obiectului „lcd” (ex. adresa I2C și terminale);
- ✓ inițializarea rezoluției afișajului prin intermediul funcției „lcd.begin ()”;
- ✓ inițializarea luminiței de fundal a ecranului printr-o secvență de licărire;
- ✓ menținerea permanent aprinsă a luminiței prin intermediul funcției „lcd.backlight ()”;
- ✓ plasarea cursorului de scriere cu ajutorul funcție „lcd.setCursor ()”;
- ✓ afișarea unui mesaj pe ecran cu ajutorul funcției „lcd.print ()”;

4. Implementarea aplicațiilor – Aplicația nr. 4

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

void setup() {
  lcd.begin(16,2);
  for(int i = 0; i< 3; i++)
  {
    lcd.backlight();
    delay(250);
    lcd.noBacklight();
    delay(250);
  }
  lcd.backlight();
}

void loop() {
  lcd.setCursor(0,0);
  lcd.print("Laborator 4");
  lcd.setCursor(0,1);
  lcd.print("Testare LCD");
}
```



4. Implementarea aplicațiilor – Aplicația nr. 4



4. Implementarea aplicațiilor – Aplicația nr. 5

➢ Implementarea aplicației nr. 5 presupune:

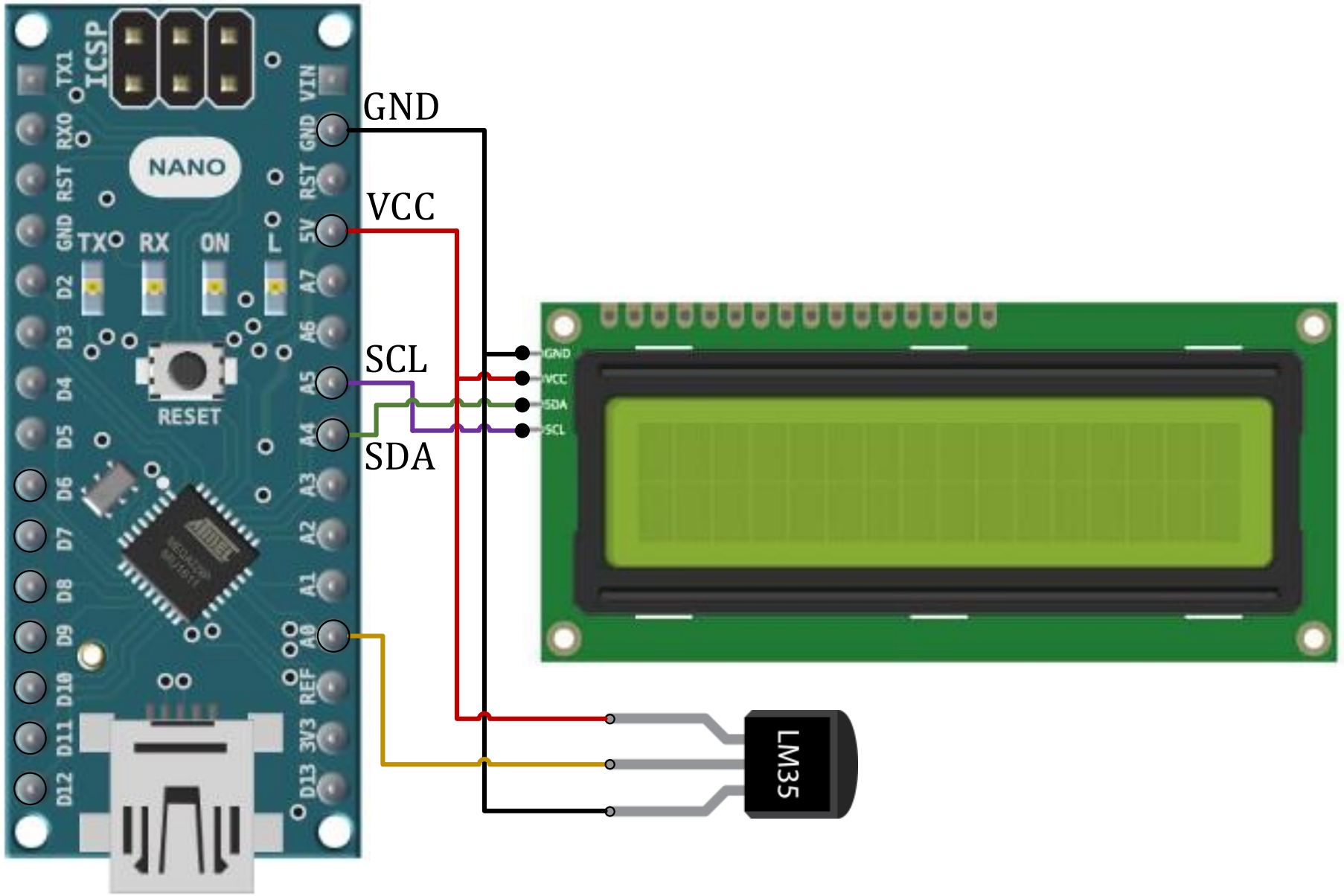
- ✓ inițializarea bibliotecii de funcții „Wire.h” pentru comunicația I2C;
- ✓ inițializarea bibliotecii de funcții „LiquidCrystal_I2C.h”;
- ✓ inițializarea parametrilor aferenți obiectului „lcd” (ex. adresa I2C și terminale);
- ✓ inițializarea unui terminal analogic „A0”;
- ✓ inițializarea variabilelor pentru determinarea conținutului registrului convertorului analog – digital, a tensiunii de măsură și a temperaturii medii;
- ✓ inițializarea rezoluției afișajului prin intermediul funcției „lcd.begin ()”;
- ✓ inițializarea luminiței de fundal a ecranului printr-o secvență de licărire;
- ✓ afișarea mesajului inițial „Temperatura [*C]: ” pe primul rând la începutul ecranului;
- ✓ menținerea permanent aprinsă a luminiței prin intermediul funcției „lcd.backlight ()”;
- ✓ determinarea temperaturii medii pe baza datelor furnizate de către traductorul LM-35 conectat la o intrare analogică a convertorului analog – digital (similar aplicației 2);
- ✓ plasarea cursorului de scriere cu ajutorul funcție „lcd.setCursor ()”;
- ✓ afișarea unui mesaj și a valorii măsurate pe ecran cu ajutorul funcției „lcd.print ()”;
- ✓ actualizarea valorilor numerice prin intermediul algoritmului „Leading Zeros Clearing”;

4. Implementarea aplicațiilor – Aplicația nr. 5

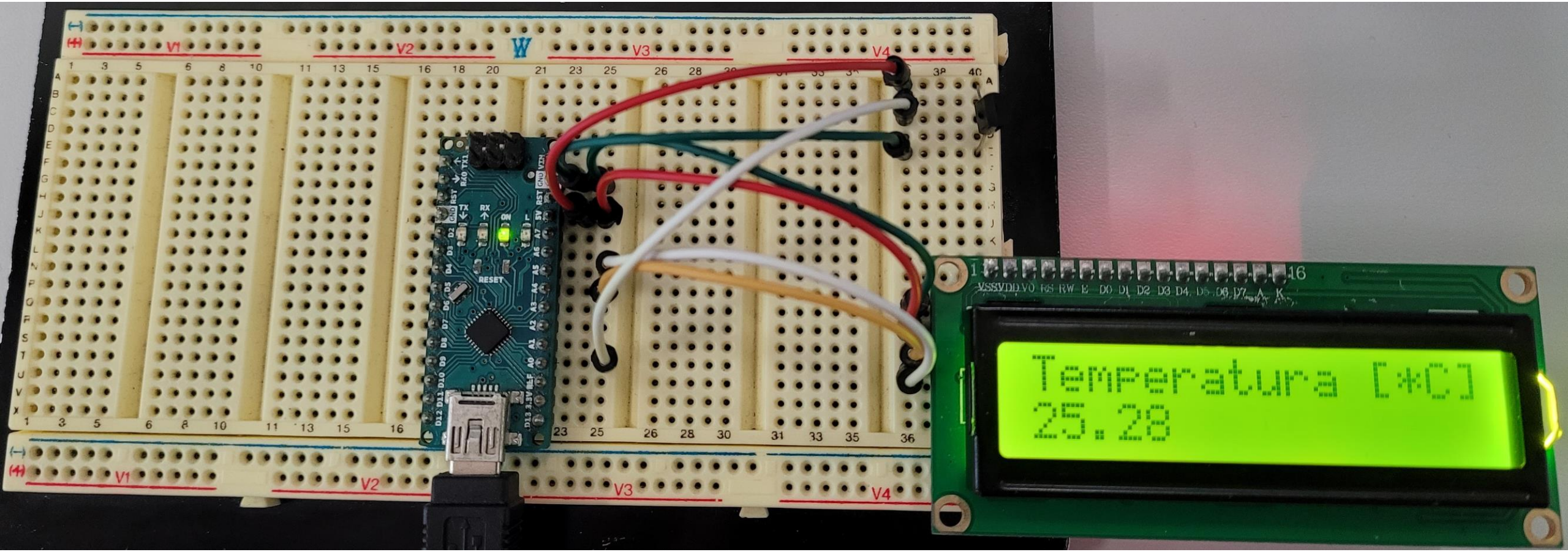
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

const int analog_pin = 0;

int ADC_val = 0;
int N = 500;
float U = 0.00;
float temp = 0.00;
float media = 0.00;

void setup() {
  lcd.begin(16,2);

  for(int i = 0; i< 3; i++) {
    lcd.backlight();
    delay(250);
    lcd.noBacklight();
    delay(250);
  }

  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Temperatura [*C]: ");
}
```

```
void loop() {

  float suma = 0.00;

  for (int i = 1; i <= N; i++) {
     ADC_val = analogRead(analog_pin);
     U = (4.78 / 1023.00) * ADC_val;
     temp = 100.00 * U;
     suma += temp;
  }
  media = suma / N;
  lcd.setCursor(0, 1);
    if(media < 1000) {
    lcd.setCursor(3, 1);
    lcd.print(" ");
    lcd.setCursor(0, 1);
    lcd.print(media);
  }
    if(media < 100) {
    lcd.setCursor(2, 1);
    lcd.print(" ");
    lcd.setCursor(0, 1);
    lcd.print(media);
  }
    if(media < 10) {
    lcd.setCursor(1, 1);
    lcd.print(" ");
    lcd.setCursor(0, 1);
    lcd.print(media);
  }
  delay(500);
}
```

4. Implementarea aplicațiilor – Aplicația nr. 5



4. Implementarea aplicațiilor – Aplicația nr. 5



4. Implementarea aplicațiilor – Aplicația nr. 6

➢ **Implementarea aplicației nr. 6 presupune:**

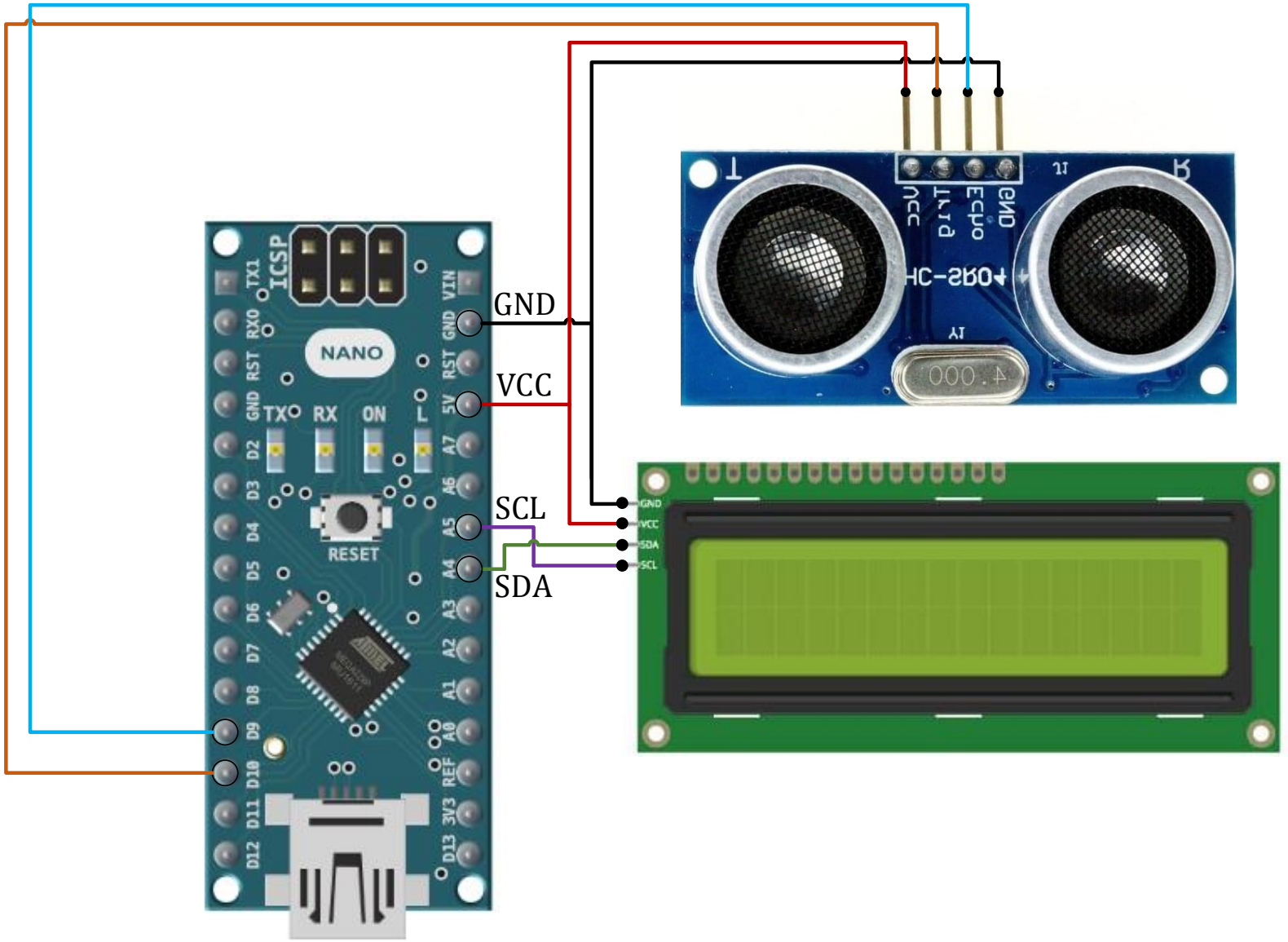
✓ inițializarea bibliotecii de funcții „Wire.h” pentru comunicația I2C;
✓ inițializarea bibliotecii de funcții „LiquidCrystal_I2C.h”;
✓ inițializarea parametrilor aferenți obiectului „lcd” (ex. adresa I2C și terminale);
✓ inițializarea unui terminalului „D10” ca și ieșire digitală;
✓ inițializarea unui terminalului „D9” ca și intrare digitală;
✓ inițializarea variabilelor pentru determinarea duratei de propagare și a distanței;
✓ inițializarea rezoluției afișajului prin intermediul funcției „lcd.begin ()”;
✓ inițializarea luminiței de fundal a ecranului printr-o secvență de licărire;
✓ afișarea mesajului inițial „Distanta [cm]: ” pe primul rând la începutul ecranului;
✓ menținerea permanent aprinsă a luminiței prin intermediul funcției „lcd.backlight ()”;
✓ determinarea distanței pe baza traductorului HC – SR04 (similar aplicației 3);
✓ plasarea cursorului de scriere cu ajutorul funcție „lcd.setCursor ()”;
✓ afișarea unui mesaj și a valorii măsurate pe ecran cu ajutorul funcției „lcd.print ()”;
✓ actualizarea valorilor numerice prin intermediul algoritmului „Leading Zeros Clearing”;

4. Implementarea aplicațiilor – Aplicația nr. 6

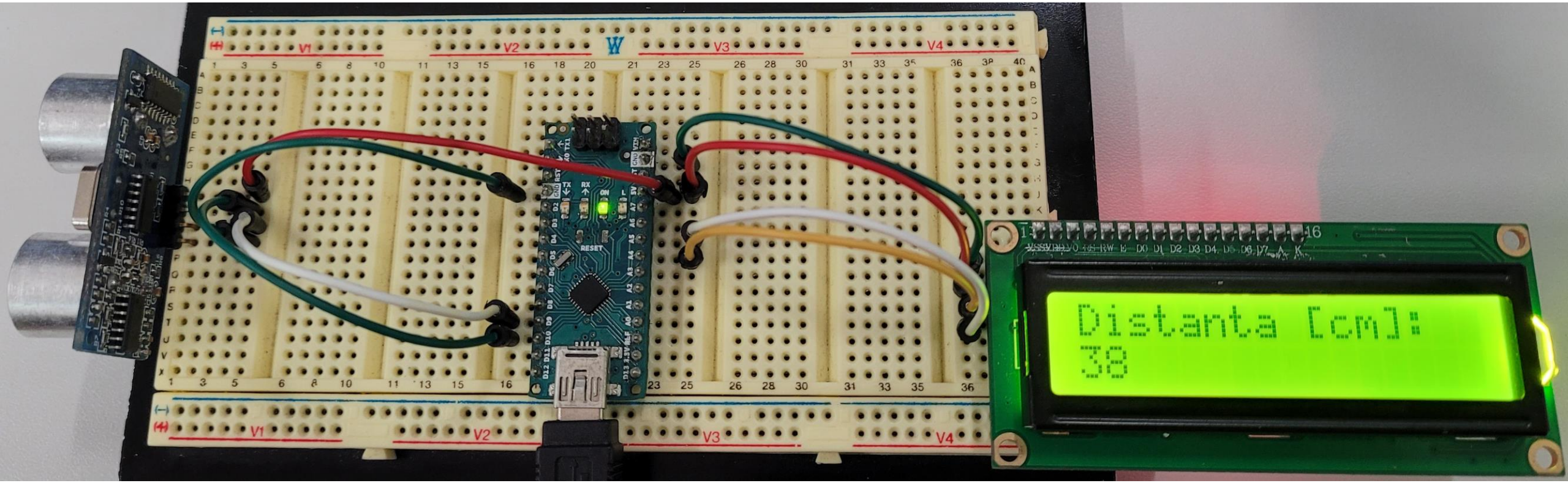
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

const int echo = 9;
const int trigger = 10;
long t = 0.0;
long d = 0.0;

void setup() {
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  lcd.begin(16,2);
  for(int i = 0; i< 3; i++)
  {
    lcd.backlight();
    delay(250);
    lcd.noBacklight();
    delay(250);
  }
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Distanta [cm]: ");
}
```

```
void loop() {
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  t = pulseIn(echo, HIGH);
  d = t * 0.034 / 2.0;
  lcd.setCursor(0, 1);
    if(d < 1000) {
    lcd.setCursor(3, 1);
    lcd.print(" ");
    lcd.setCursor(0, 1);
    lcd.print(d);
  }
    if(d < 100) {
    lcd.setCursor(2, 1);
    lcd.print(" ");
    lcd.setCursor(0, 1);
    lcd.print(d);
  }
    if(d < 10) {
    lcd.setCursor(1, 1);
    lcd.print(" ");
    lcd.setCursor(0, 1);
    lcd.print(d);
  }
  delay(500);
}
```

4. Implementarea aplicațiilor – Aplicația nr. 6



4. Implementarea aplicațiilor – Aplicația nr. 5



5. Concluzii

- ✓ Aplicațiile implementate pe baza microcontrolerelor pot funcționa în mod independent de calculator dacă există mijloace periferice de intrare sau ieșire atașate la terminalele microcontrolerului care pot înlocui rolul calculatorului gazdă.

- ✓ Funcționarea în mod independent de calculatorul gazdă al aplicației finale presupune aplicarea unor algoritmi de optimizare atât al modului de funcționare cât și al modului de execuție specific codului program.

6. Bibliografie

1. Ioana - Cornelia Gros, Lucian - Nicolae Pintilie, Teodor Crișan Pană – „SISTEME EMBEDDED ÎN INGINERIE ELECTRICĂ - GHID DE APLICAȚII”, Editura UTPRESS, Cluj-Napoca, 2020, ISBN 978-606-737-431-5

2. Electronics and Power electronics (EPE) Brings power and electronics together © 2017 – „Documentație pentru laboratorul de Sisteme cu Microprocesoare” https://epe.utcluj.ro/index.php/sisteme-cu-microprocesoare/

3. Handson Technology – „HC-SR04 Ultrasonic Sensor Module User Guide” - Ultrasonic Sensor V2.0, SKU: MDU-1014 - www.handsontec.com https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf

4. Shenzhen Eone Electronics CO. LTD. – „Specification for LCD module 1602A-1 V1.2” https://www.openhacks.com/uploadsproductos/eone-1602a1.pdf

5. Handson Technology – „I2C Serial Interface 1602 LCD Module” - SKU: DSP-1182 https://www.handsontec.com/dataspecs/module/I2C_1602_LCD.pdf