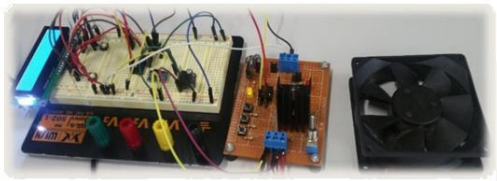
UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
FACULTATEA DE INGINERIE ELECTRICĂ

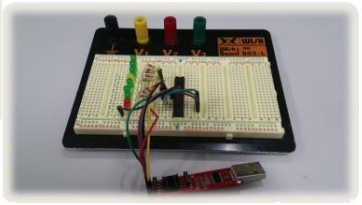
SISTEME CU MICROPROCESOARE

Ședința de laborator nr. 3

Manipularea intrărilor analogice

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 modului de funcționare al convertorului analog – digital [1] [2]
- ✓ Prezentarea modului de funcționare al convertorului digital – analog [2]
- ✓ Prezentarea modului de funcționare al unității pentru generare a impulsurilor [1] [2]
- ✓ Implementare aplicațiilor specifice procesării semnalelor analogice [1] [2] [3]

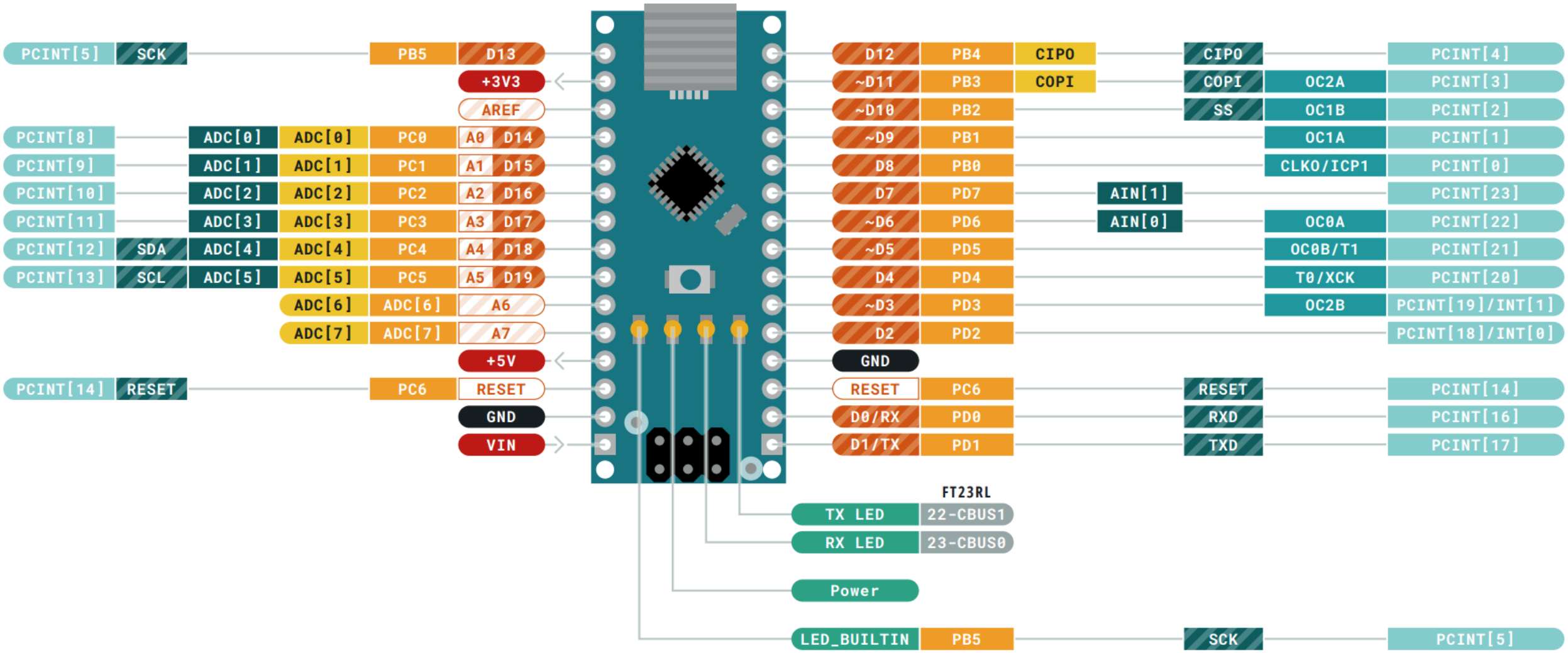
2. Introducere

- Microcontrolerul ATMega 328P din cadrul platformei de dezvoltare Arduino Nano, conferă posibilitatea achiziționării și prelucrării atât a semnalelor digitale cât și a **semnalelor analogice**. Terminalele marcate cu indicativul „Ax” (unde „x” reprezintă numărul de ordine), reprezintă un set de **intrări analogice**.

- Etajul din cadrul arhitecturii care permite achiziționarea semnalelor analogice, poartă denumirea de **convertor analog – digital** (eng. Analog to Digital Converter – ADC).

- Prin intermediul acestui etaj de circuit, microcontrolerul ATMega 328P poate măsura **semnale de tensiune variabile în timp** cu amplitudinea cuprinsă între 0 și 5 [V].

2. Introducere



https://store.arduino.cc/products/arduino-nano

3. Aspecte teoretice

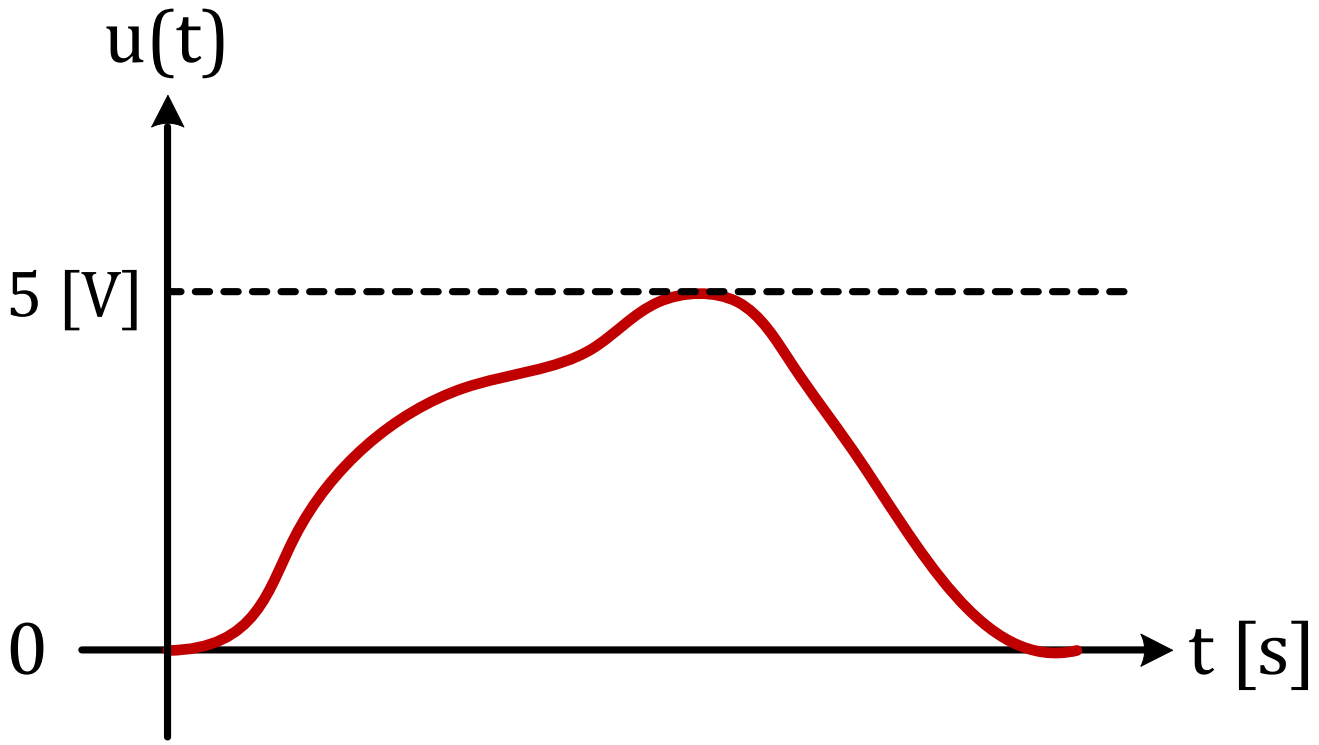
- Într-un circuit electronic, semnalele analogice sunt reprezentate ca și o **tensiune cu amplitudine variabilă în timp.**

- Semnalele de tensiune variabile în timp provin în mare parte de la **tradutoare sau senzori**.

- Există două clase de traductoare:
  - ✓ **traductoare active** (care nu necesită alimentare deoarece produc tensiune)
  - ✓ **traductoare pasive** (care necesită alimentare deoarece se bazează pe variația unui parametru de circuit precum rezistența, capacitatea sau inductivitatea)

- Există de asemenea două categorii de circuite sau etaje intermediare:
  - ✓ **circuite formatoare de semnal** pentru traductoarele active (ex. amplificatoare)
  - ✓ **circuite adaptoare de impedanță** pentru traductoarele pasive (ex. filtre pasive)

3. Aspecte teoretice



Reprezentarea semnalelor analogice în raport cu timpul

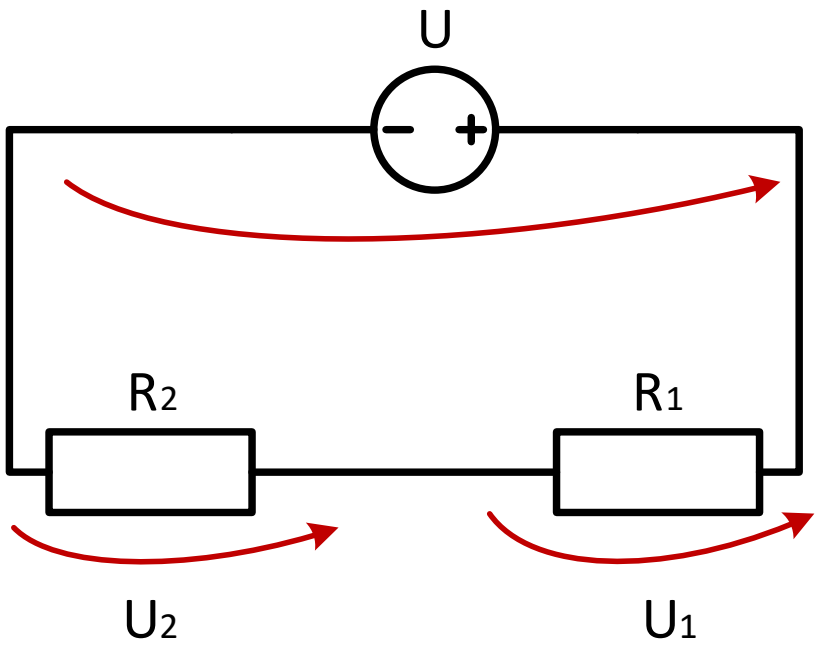
3. Aspecte teoretice

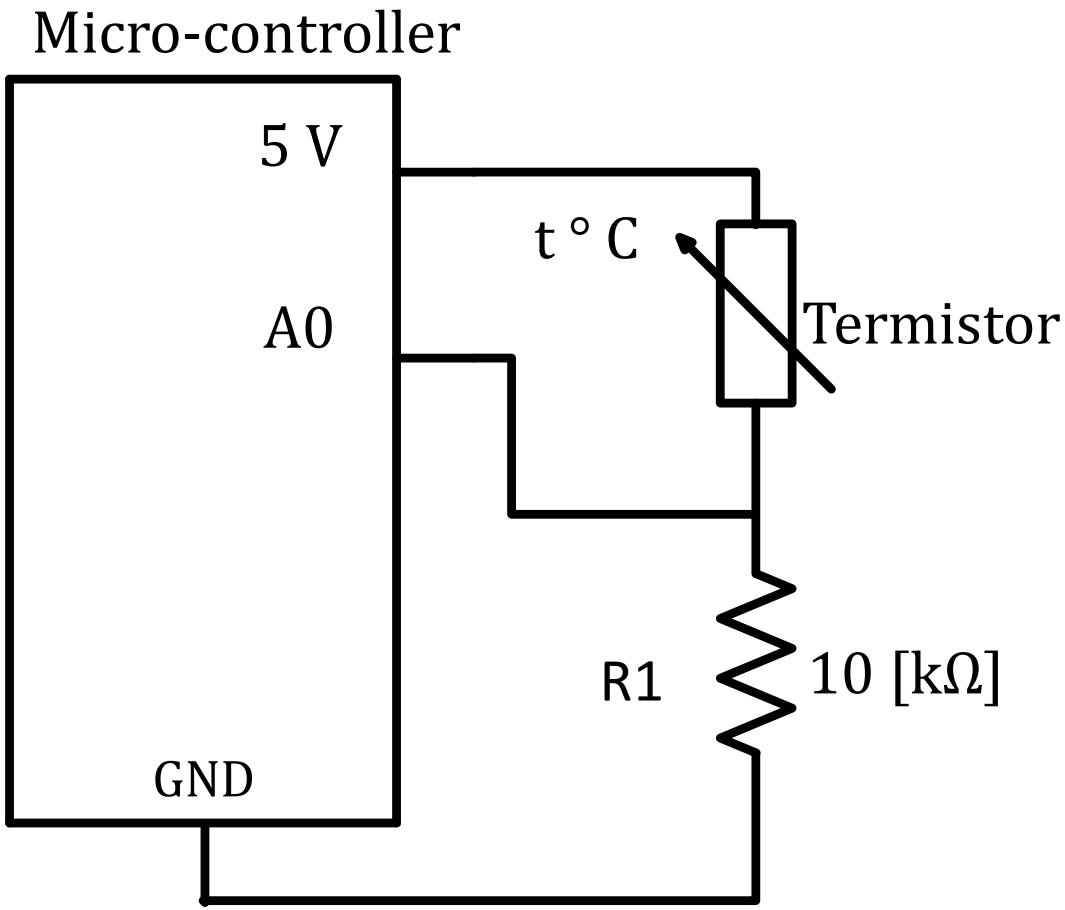
- **Traductoarele pasive** pot fi atașate la microcontroler prin intermediul **divizorului de tensiune** în componența căruia una dintre rezistențe este **variabilă** (ex. montaj în semi-punte sau potențiometric). Rezistența variabilă reprezintă însuși **traductorul** (ex. termistor)

- În cazul divizorului există următoarele relații de calcul pentru tensiune:

$$U_1 = R_1 \cdot \frac{U}{R_1 + R_2} \qquad U_2 = R_2 \cdot \frac{U}{R_1 + R_2}$$

3. Aspecte teoretice

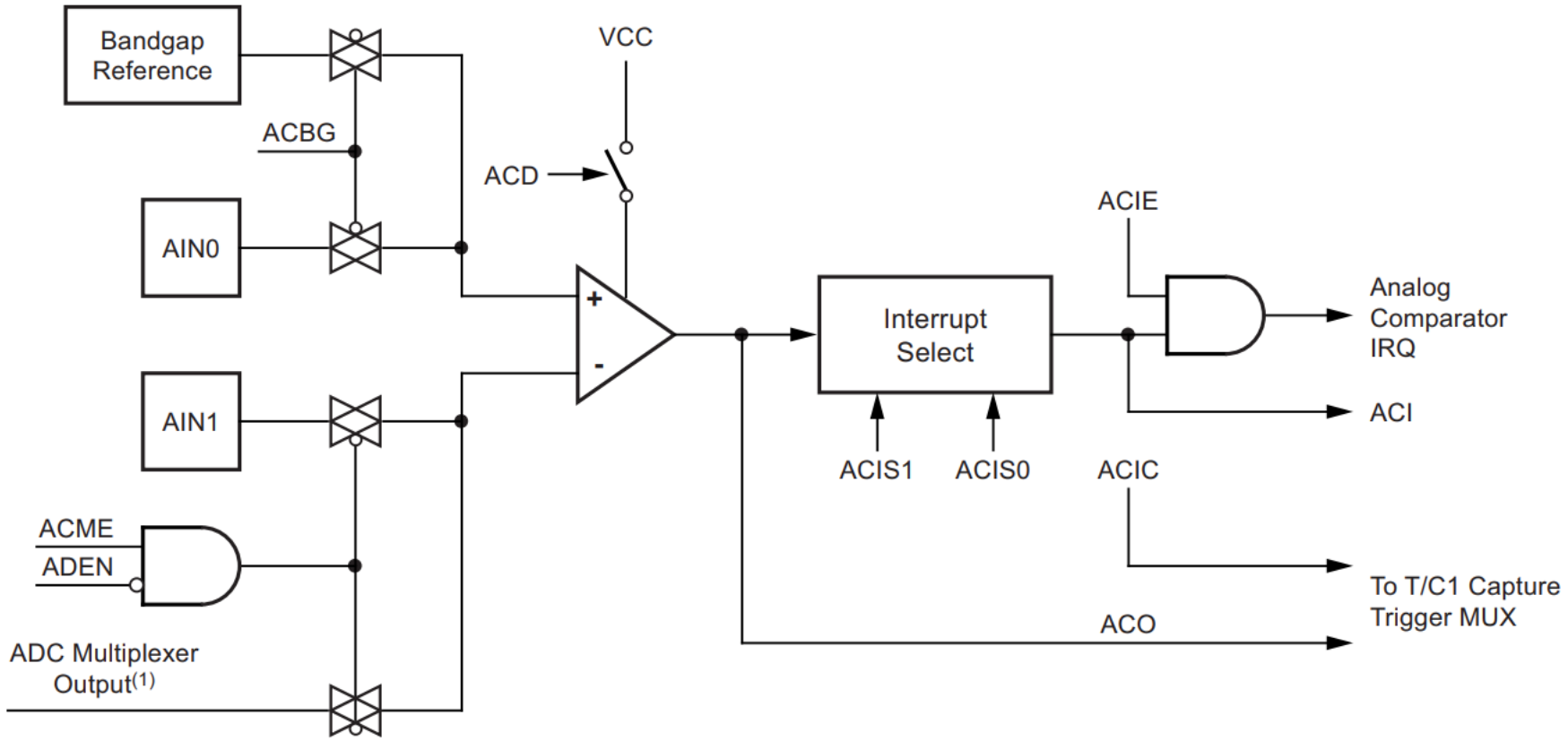




Divizorul de tensiune și traductoarele pasive atașate la microcontroler

3. Aspecte teoretice

➢ Cel mai simplu etaj din componența arhitecturii microcontrolerului ATMega 328P, care poate prelucra semnalul analogic este **comparatorul electronic**



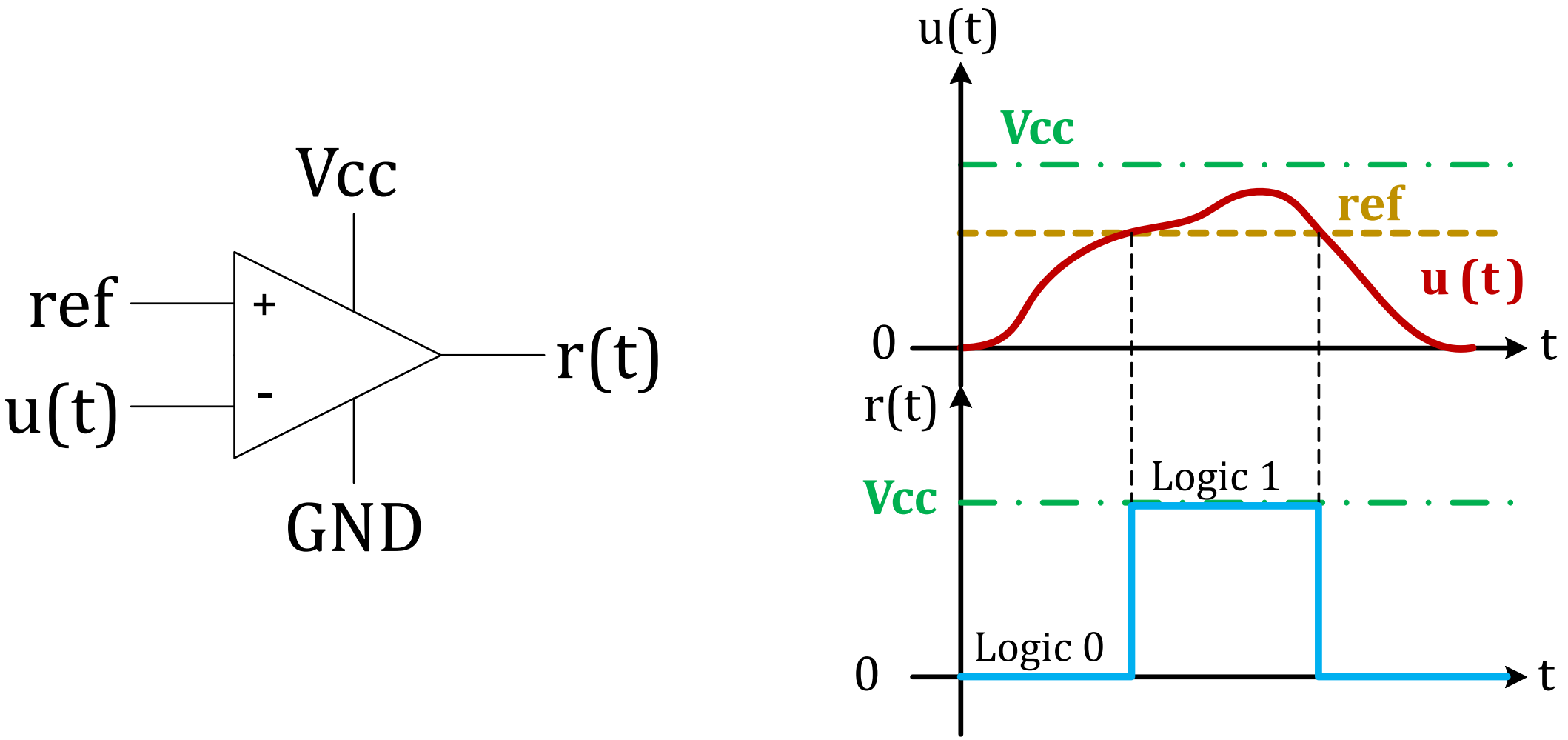
Comparatorul din cadrul arhitecturii microcontrolerului ATMega 328P

3. Aspecte teoretice

➢ Comparatorul electronic are rolul de a „sesiza” **diferența de potențial** dintre **tensiunea de referință** și **semnalul analogic măsurat** „u(t)”. Tensiunea de referință „ref” poate fi „0”, diferită de „0” sau egală cu tensiunea de alimentare „Vcc”.

➢ Când **tensiunea măsurată** devine **egală** cu **tensiunea de referință**, comparatorul produce un semnal **digital** la ieșire cu **nivelul 1 logic** (adică tensiunea de alimentare se va regăsi la ieșire)

3. Aspecte teoretice



Principiul de funcționare al comparatorului electronic

3. Aspecte teoretice

➢ Convertorul analog – digital (eng. Analog to Digital Converter – ADC) reprezintă o **serie progresivă de comparatoare**.

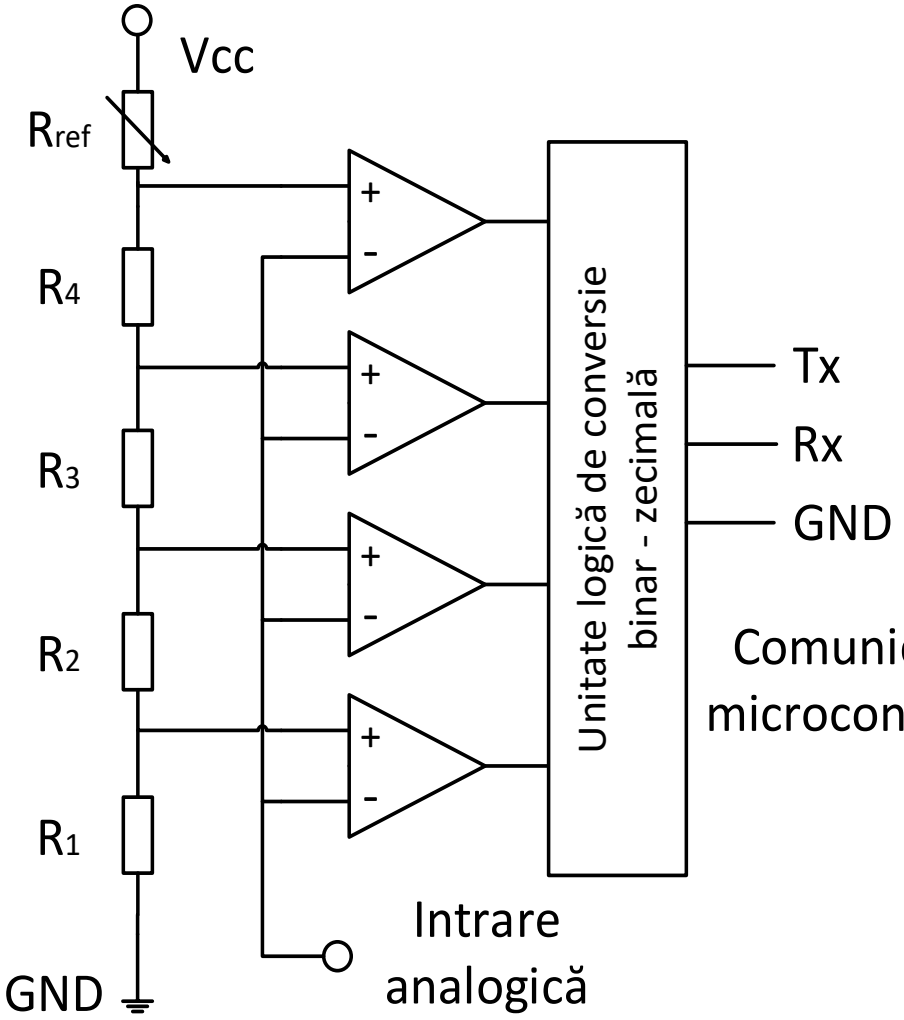
➢ Numărul de comparatoare redă **rezoluția convertorului** (ex. 4 comparatoare = 2^4 combinații posibile).

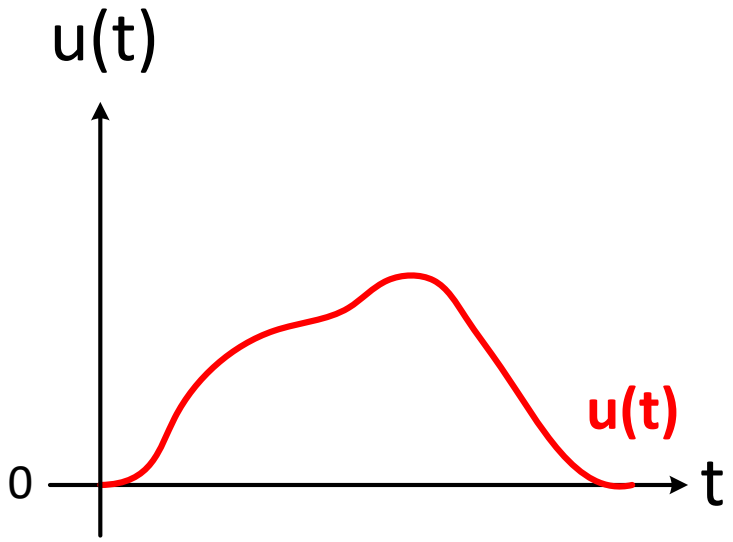
➢ În componența convertorului analog – digital există un **divizor rezistiv** cu un **număr finit** de rezistențe **egal** cu **numărul de comparatoare**.

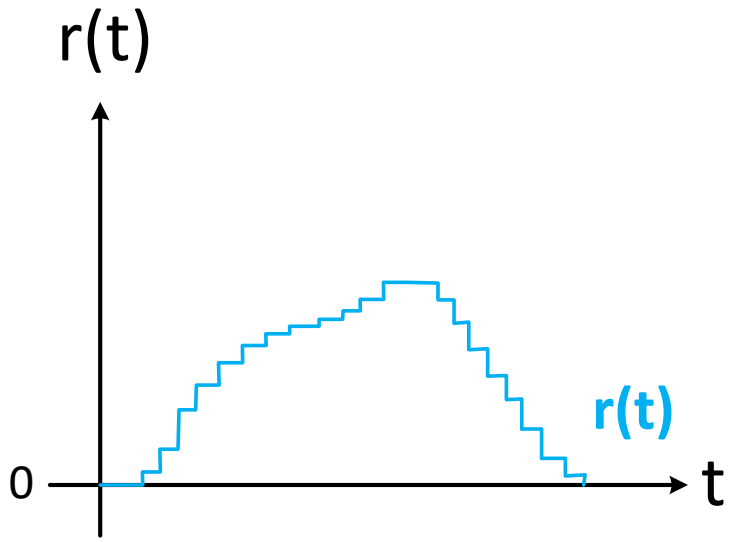
➢ Fiecare comparator verifică **nivelul de tensiune** corespunzător căderii la bornele rezistenței corespunzătoare

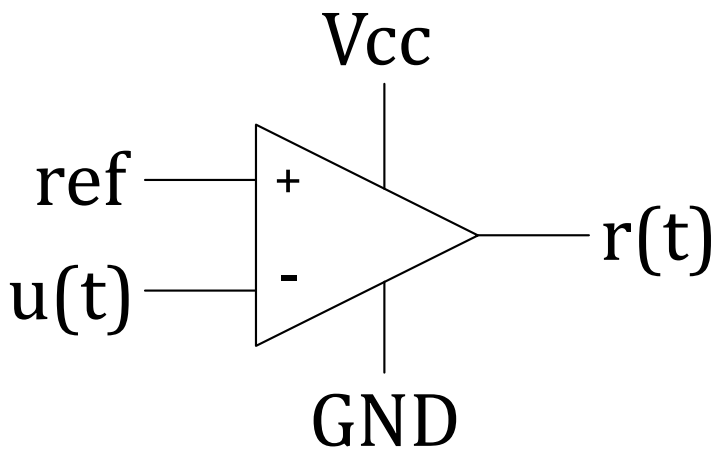
➢ **Semnalul rezultant** „r(t)” înregistrat în memoria microcontrolerului de către convertorul analog – digital, va fi de **natură digitală**, anume un **semnal discretizat** (construit prin variația în trepte a amplitudinii)

3. Aspecte teoretice









Principiul de funcționare al convertorului analog – digital

3. Aspecte teoretice

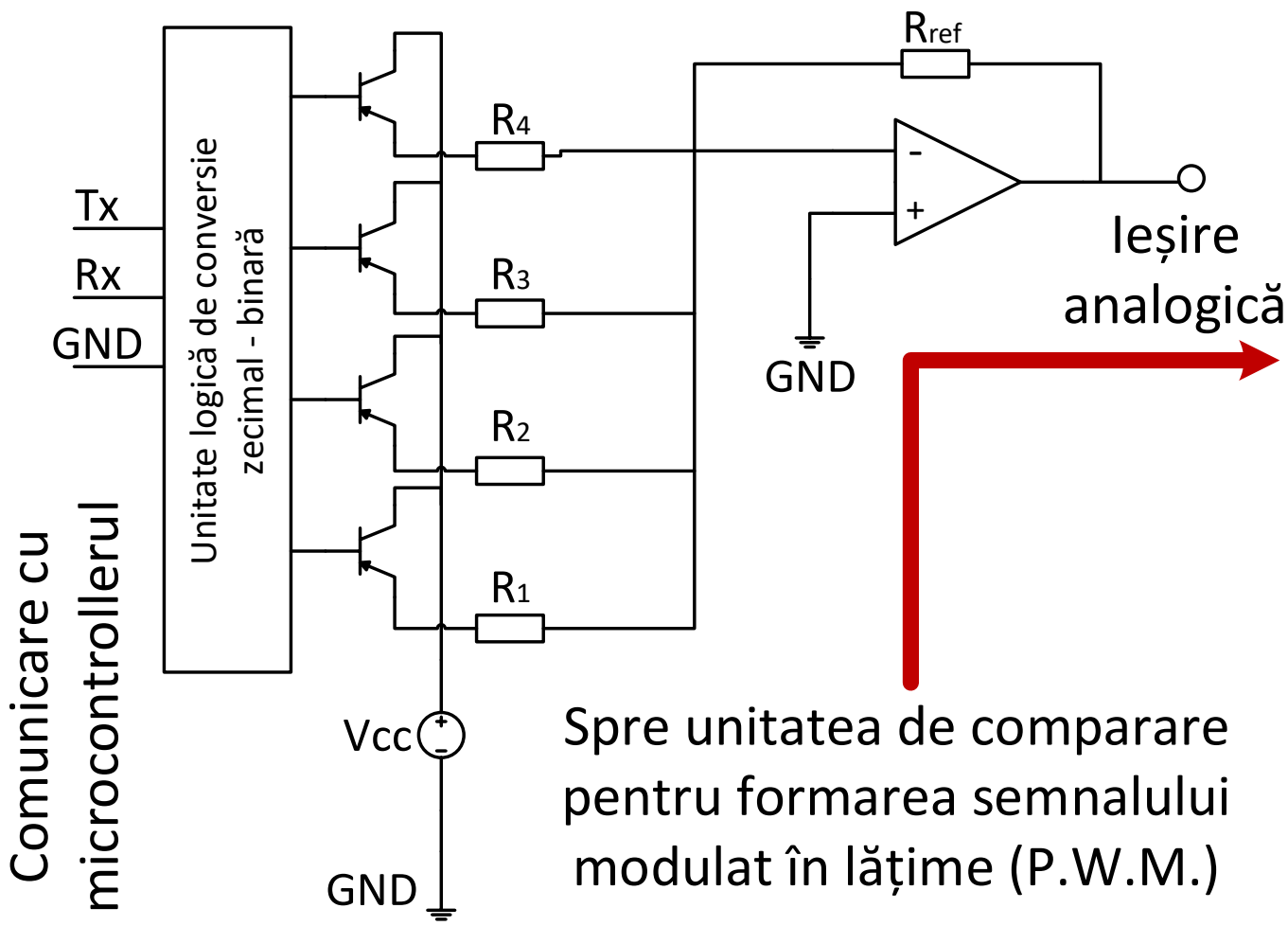
➢ Circuitul care realizează **funcția inversă** conversiei analog – digitale (numerice), este convertorul digital – analog (eng. Digital to Analog Converter – DAC)

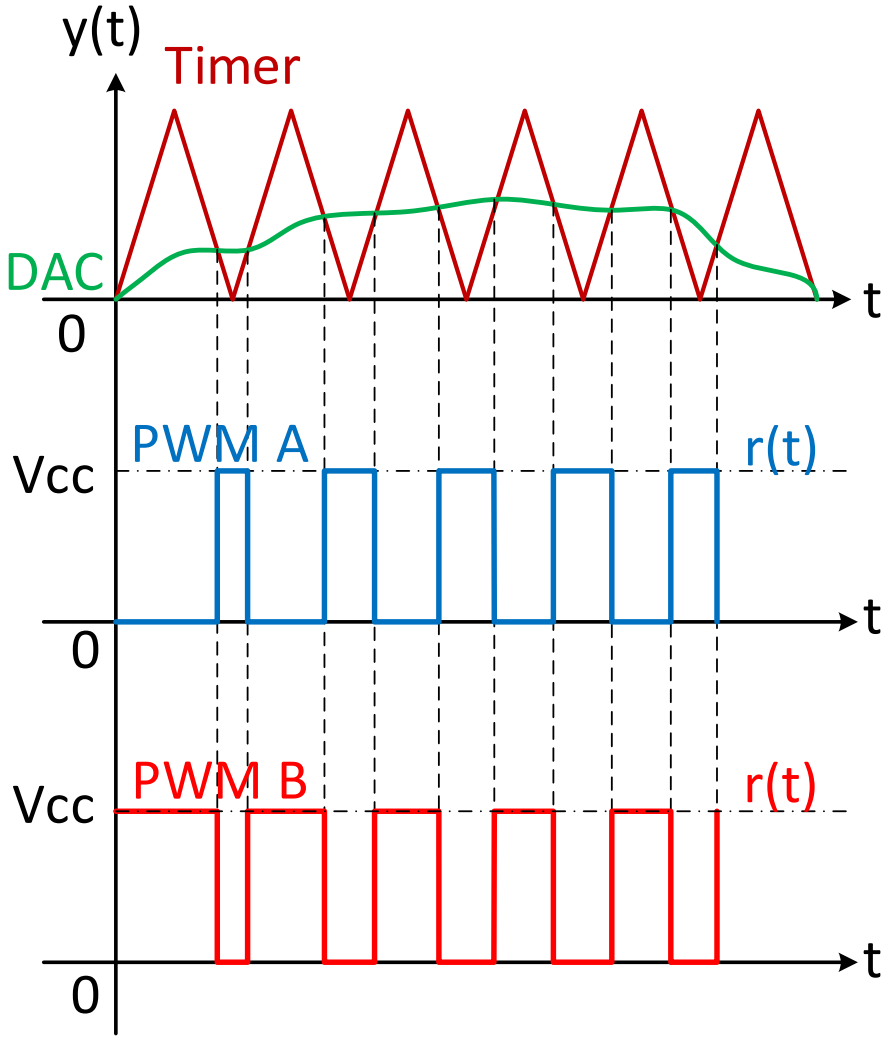
➢ Rolul acestui circuit este de a **furniza un semnal de tensiune cu amplitudine variabilă în funcție de comanda numerică dată**

➢ În componența convertorului digital – analog există **un divizor de tensiune** și o **serie de comutatoare electronice** care cuplează mai multe valori de tensiune la intrarea unui **amplificator sumator**

➢ În cadrul unor arhitecturi de microcontroler sau procesor digital de semnal, **nivelul de tensiune rezultant** „DAC” al convertorului digital – analog este utilizat în vederea **generării semnalului dreptunghiular modulat în lățime** „r(t)” (eng. Pulse Width Modulation – PWM).

3. Aspecte teoretice





Principiul de funcționare al convertorului digital – analog
și al unității pentru generarea impulsurilor

3. Aspecte teoretice

➢ În vederea implementării aplicațiilor specifice procesării semnalelor analogice cu ajutorul microcontrolerului ATMega 328P, se vor avea în vedere două instrucțiuni:

✓ Instrucțiunea **condițională „if ()”** (ro. dacă) și **„else ()”** (ro. altfel), spre exemplu:

```
if (variabila_1 > variabila_2) {
   instrucțiune_1 (argument_1, ARGUMENT_2);
}
else {
   instrucțiune_2 (argument_1, ARGUMENT_2);
}
```

3. Aspecte teoretice

➢ În vederea implementării aplicațiilor specifice procesării semnalelor analogice cu ajutorul microcontrolerului ATMega 328P, se vor avea în vedere două instrucțiuni:

✓ Instrucțiunea de **constrângere** și **scalare „map ()”** (ro. restrângerea unui domeniu de variație la un anumit interval prin regula de trei simplă). Spre exemplu:

```
variabila_2 = map (variabila_1, valoare_inferioară_interval_1,
valoare_superioară_interval_1, valoare_inferioară_interval_2,
valoare_superioară_interval_2);
```

❖ 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**]:

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

4. Implementarea aplicațiilor

- În vederea studierii modului de implementare al aplicațiilor specifice procesării semnalelor digitale se propune următoarea serie de aplicații:

1. Determinarea rezoluției convertorului analog – digital
2. Determinarea preciziei convertorului analog – digital și a valorii de tensiune
3. Determinarea temperaturii cu ajutorul traductorului LM-35
4. Implementarea unui comparator numeric cu prag digital reglabil
5. Implementarea unei coloane luminoase indicatoare de nivel
6. Generarea unui semnal dreptunghiular modulat în lățime

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
- ✓ rezistențe cu valoarea de 100 [Ω]
- ✓ senzor de temperatura LM-35
- ✓ potențiometru cu valoarea maximă a rezistenței 10 [kΩ]
- ✓ 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 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”

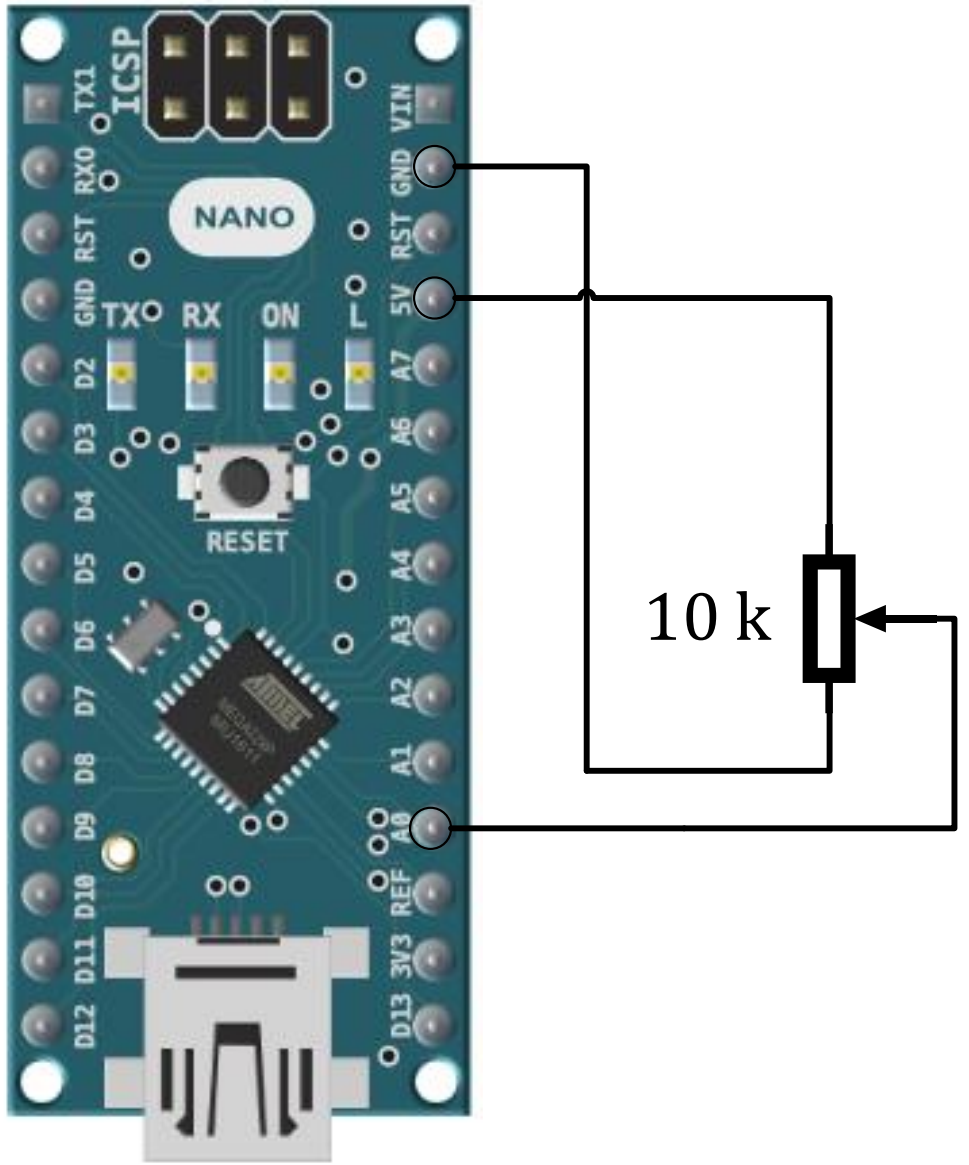
✓ inițializarea comunicației Serial la viteza de transfer 9600 [b/s]

✓ preluarea valorii zecimale rezultante în urma procesului de conversie analog – digital

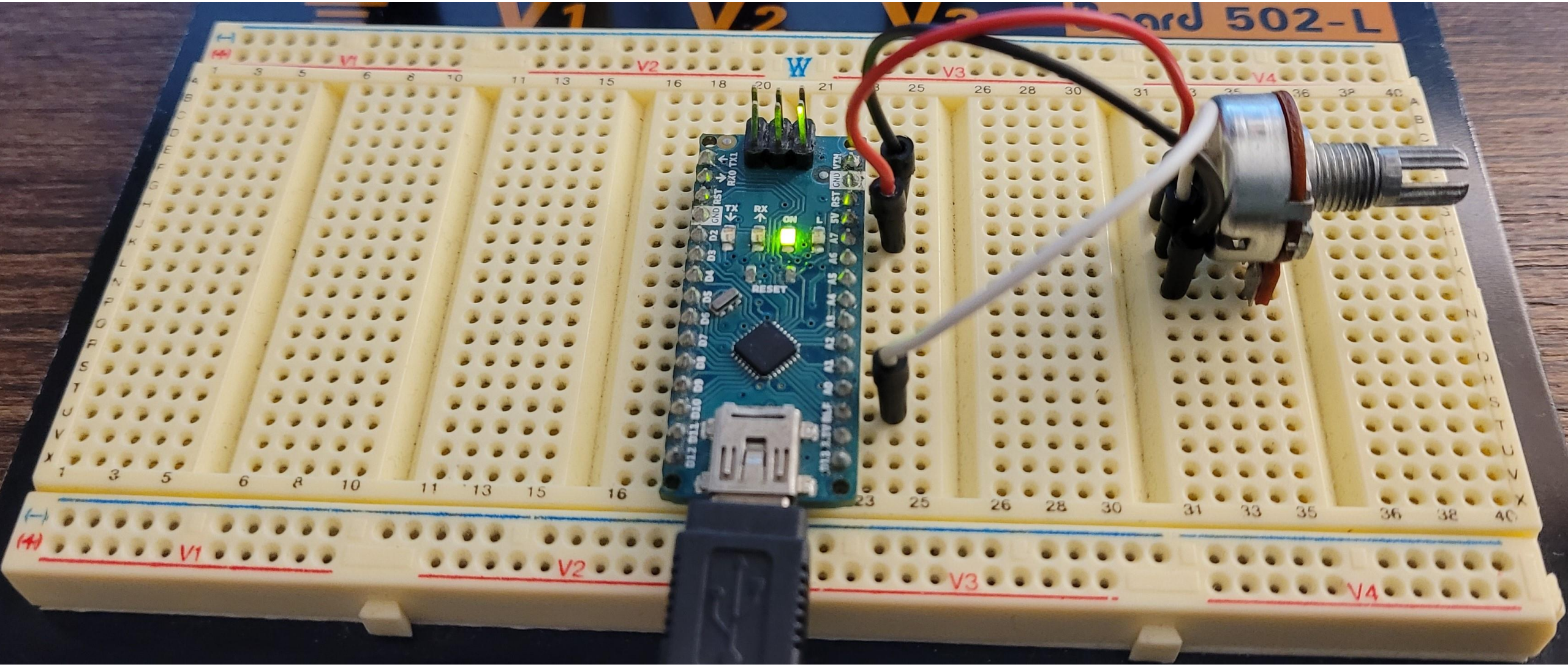
✓ afișarea rezultatului de conversie în consola serial însoțit de mesajul „ADC: ”

4. Implementarea aplicațiilor – Aplicația nr. 1

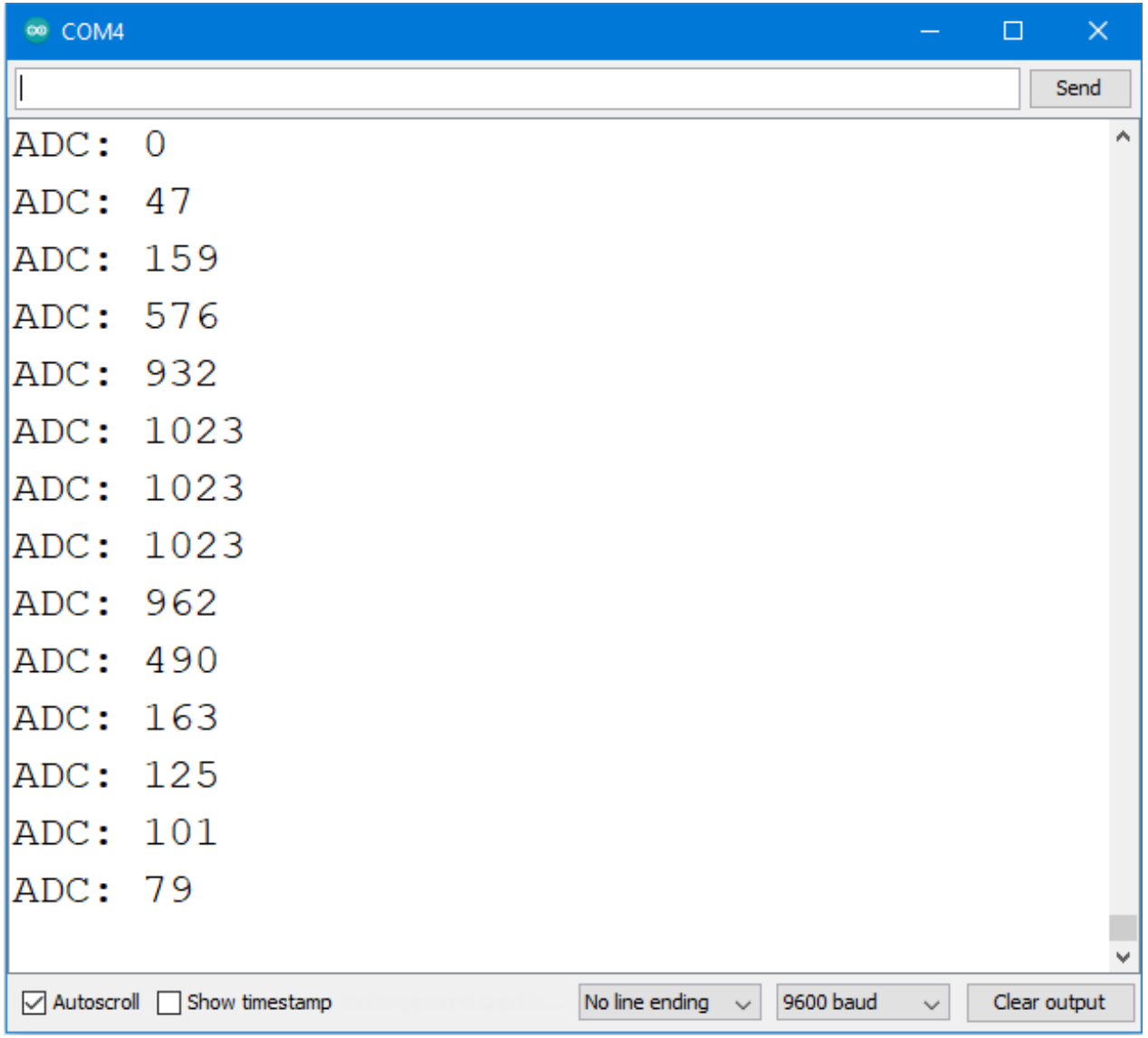
```
const int analog_pin = 0;
int ADC_val = 0;

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

void loop() {
  ADC_val = analogRead(analog_pin);
  Serial.print("ADC: ");
  Serial.print(ADC_val);
  Serial.println("");
  delay(250);
}
```



4. Implementarea aplicațiilor – Aplicația nr. 1



4. Implementarea aplicațiilor – Aplicația nr. 1



Consola Serial – Rezultatul conversiei analog - digital

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 fracționar „U” cu valoarea „0.00”

✓ inițializarea comunicației Serial la viteza de transfer 9600 [b/s]

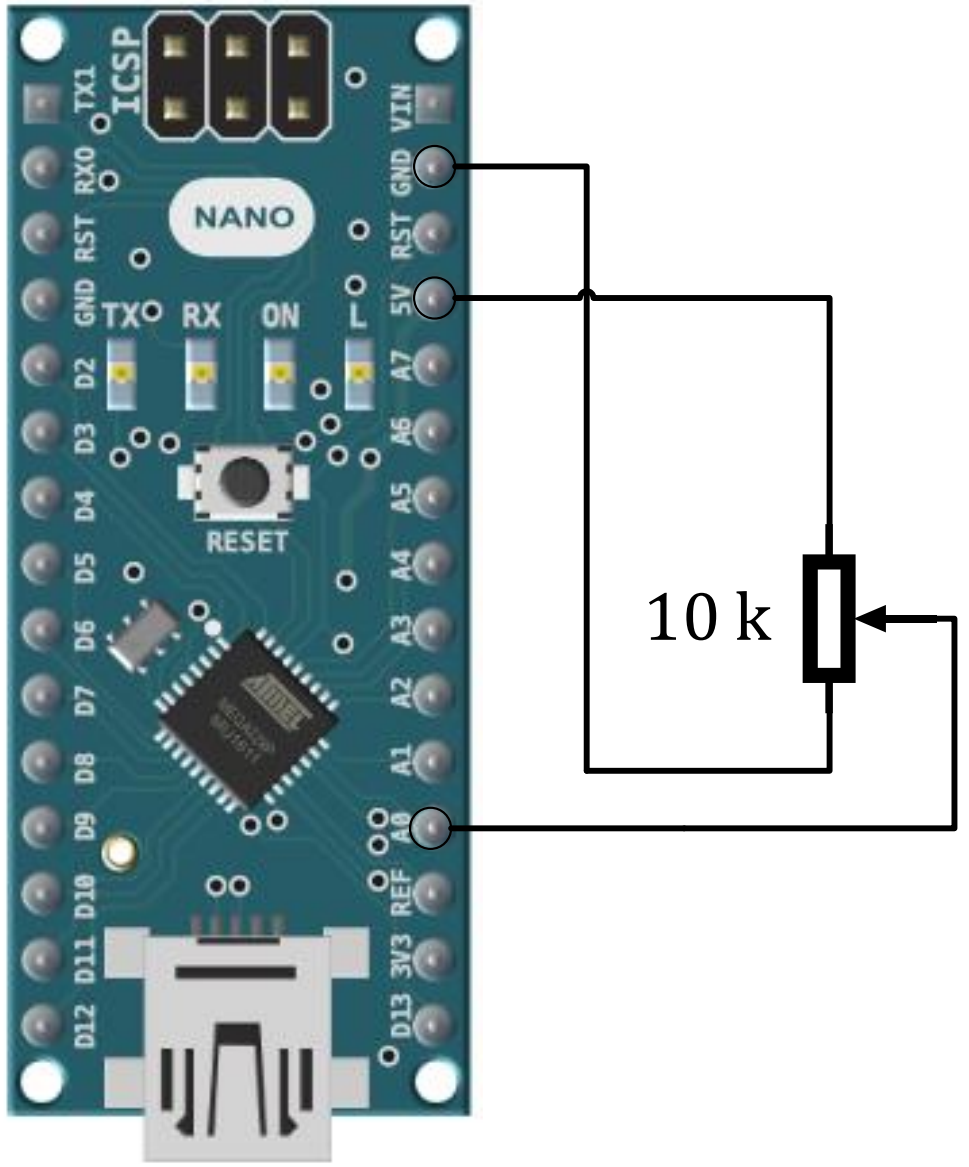
✓ preluarea valorii rezultante în urma procesului de conversie analog – digital

✓ determinarea tensiunii de măsură pe baza preciziei convertorului analog – digital;

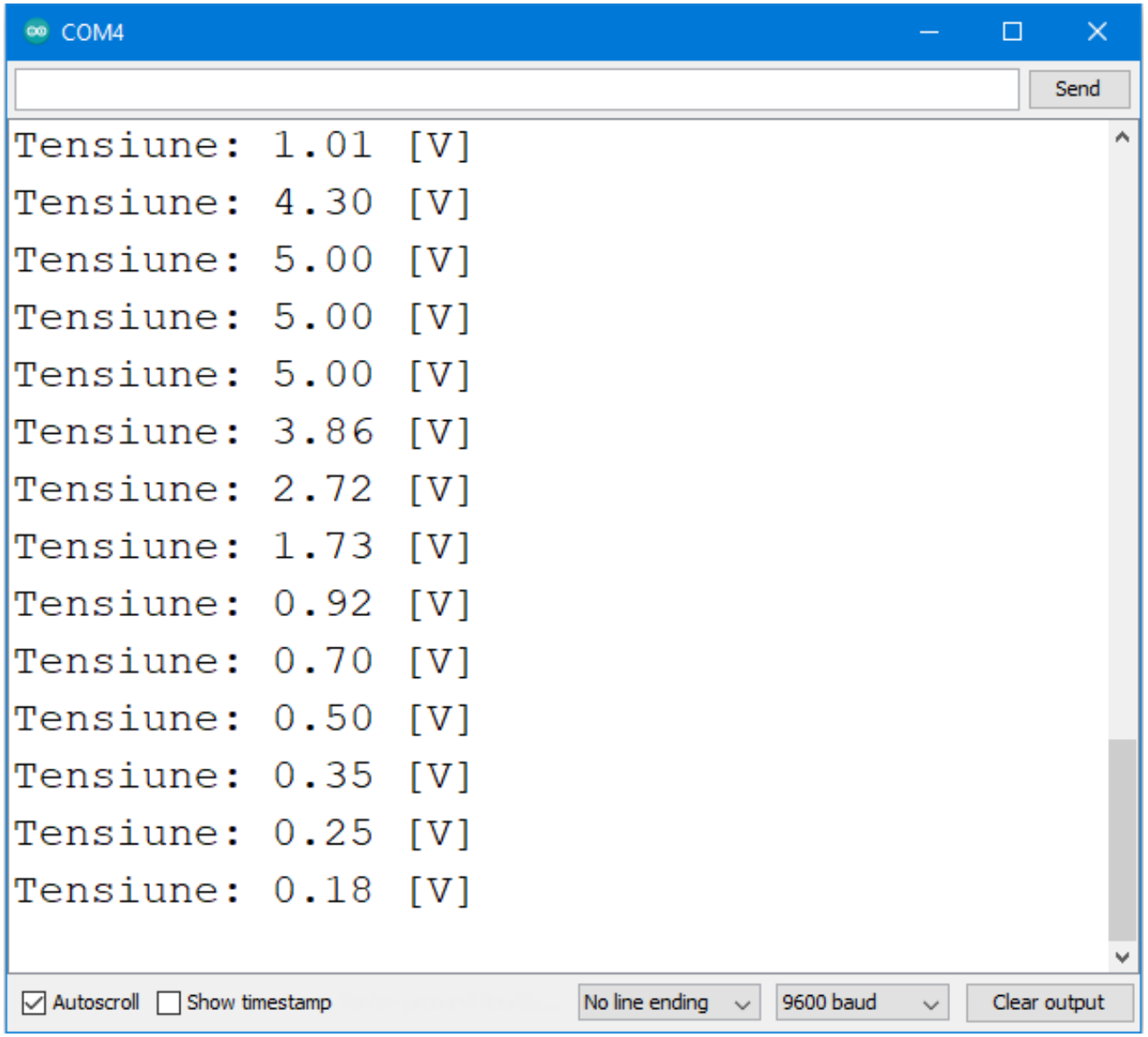
✓ afișarea nivelului de tensiune în consola Serial însoțit de mesajul „Tensiune: __ [V] ”

4. Implementarea aplicațiilor – Aplicația nr. 2

```
const int analog_pin = 0;
int ADC_val = 0;
float U = 0.00;

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

void loop() {
  ADC_val = analogRead(analog_pin);
  U = (5.00 / 1023.00) * ADC_val;
  Serial.print("Tensiune: ");
  Serial.print(U);
  Serial.print(" [V]");
  Serial.println("");
  delay(250);
}
```



4. Implementarea aplicațiilor – Aplicația nr. 2



Consola Serial – Afișarea nivelului de tensiune măsurat de convertor

4. Implementarea aplicațiilor – Aplicația nr. 3

➢ **Implementarea aplicației nr. 3 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 fracționar „U” cu valoarea „0.00”

✓ inițializarea unei variabile de tip fracționar „temp” cu valoarea „0.00”

✓ inițializarea comunicației Serial la viteza de transfer 9600 [b/s]

✓ preluarea valorii rezultante în urma procesului de conversie analog – digital

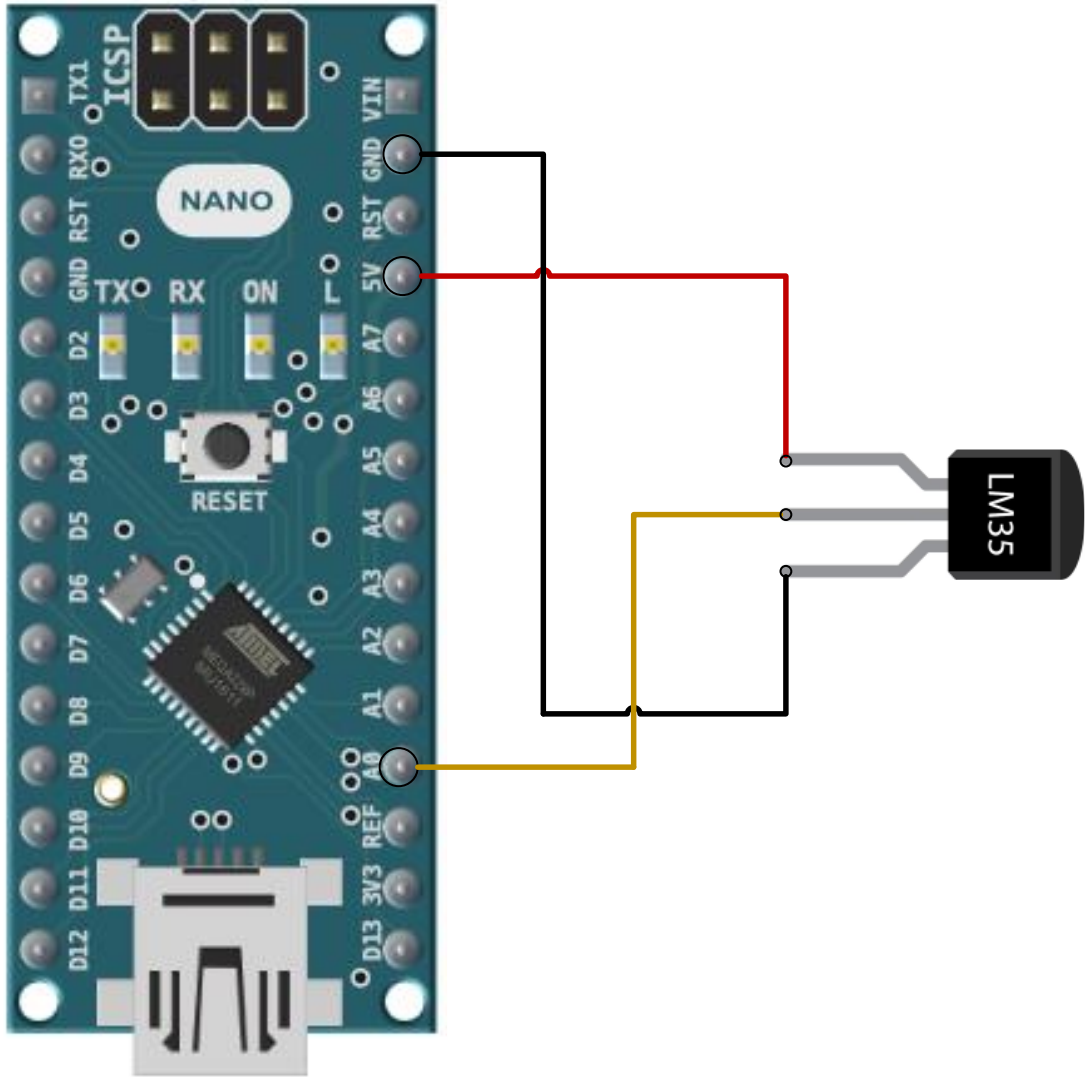
✓ determinarea tensiunii pe baza preciziei convertorului analog – digital

✓ determinarea temperaturii pe baza tensiunii și a constantei de calibrare

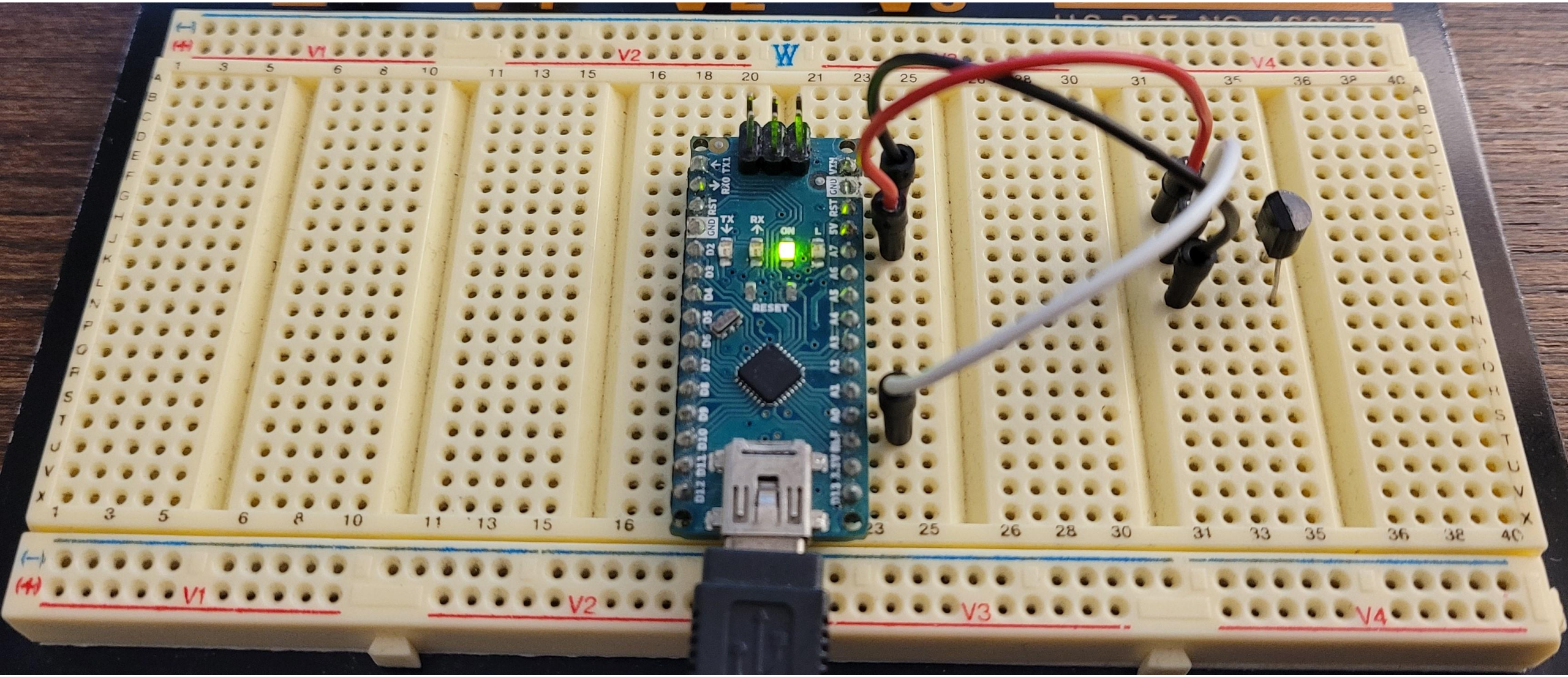
✓ afișarea valorii temperaturii în consola Serial însoțită de mesajul „Temperatura: _ [*C] ”

4. Implementarea aplicațiilor – Aplicația nr. 3

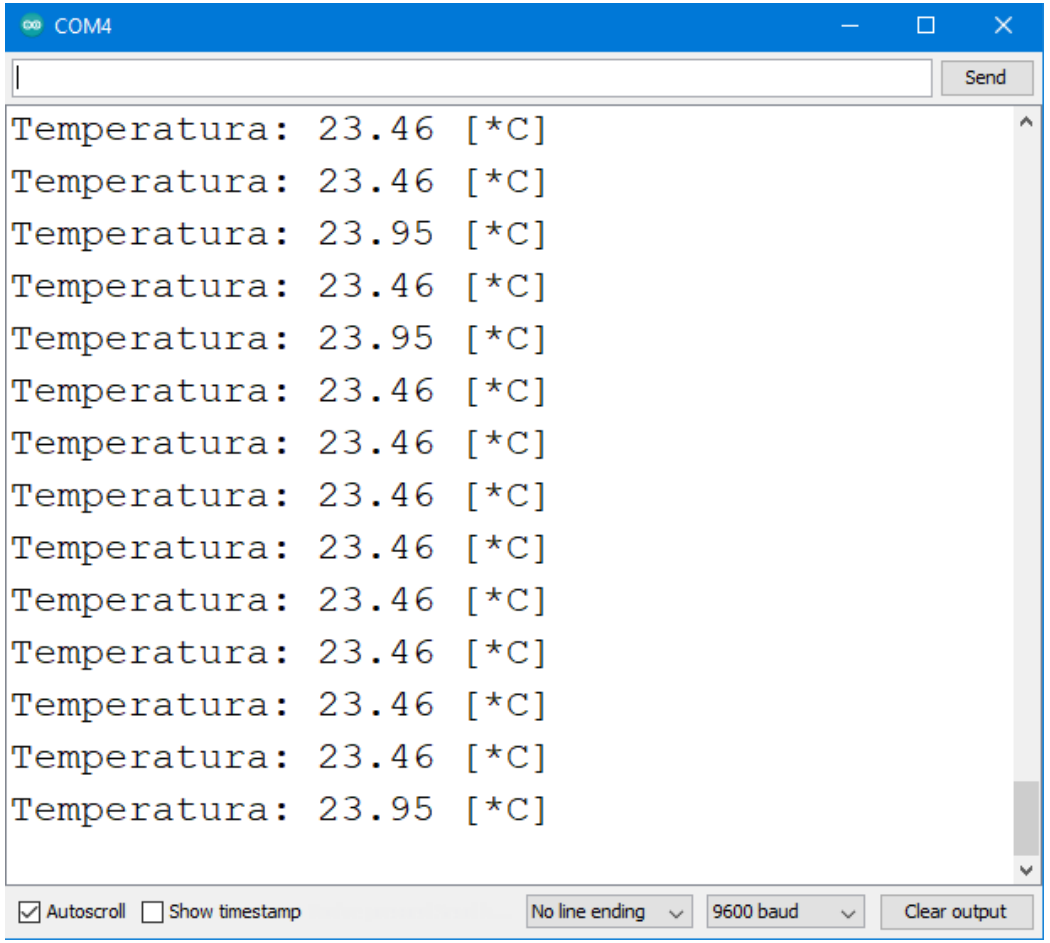
```
const int analog_pin = 0;
int ADC_val = 0;
float U = 0.00;
float temp = 0.00;

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

void loop() {
  ADC_val = analogRead(analog_pin);
  U = (5.00 / 1023.00) * ADC_val;
  temp = 100.00 * U;
  Serial.print("Temperatura: ");
  Serial.print(temp);
  Serial.print(" [*C]");
  Serial.println("");
  delay(250);
}
```

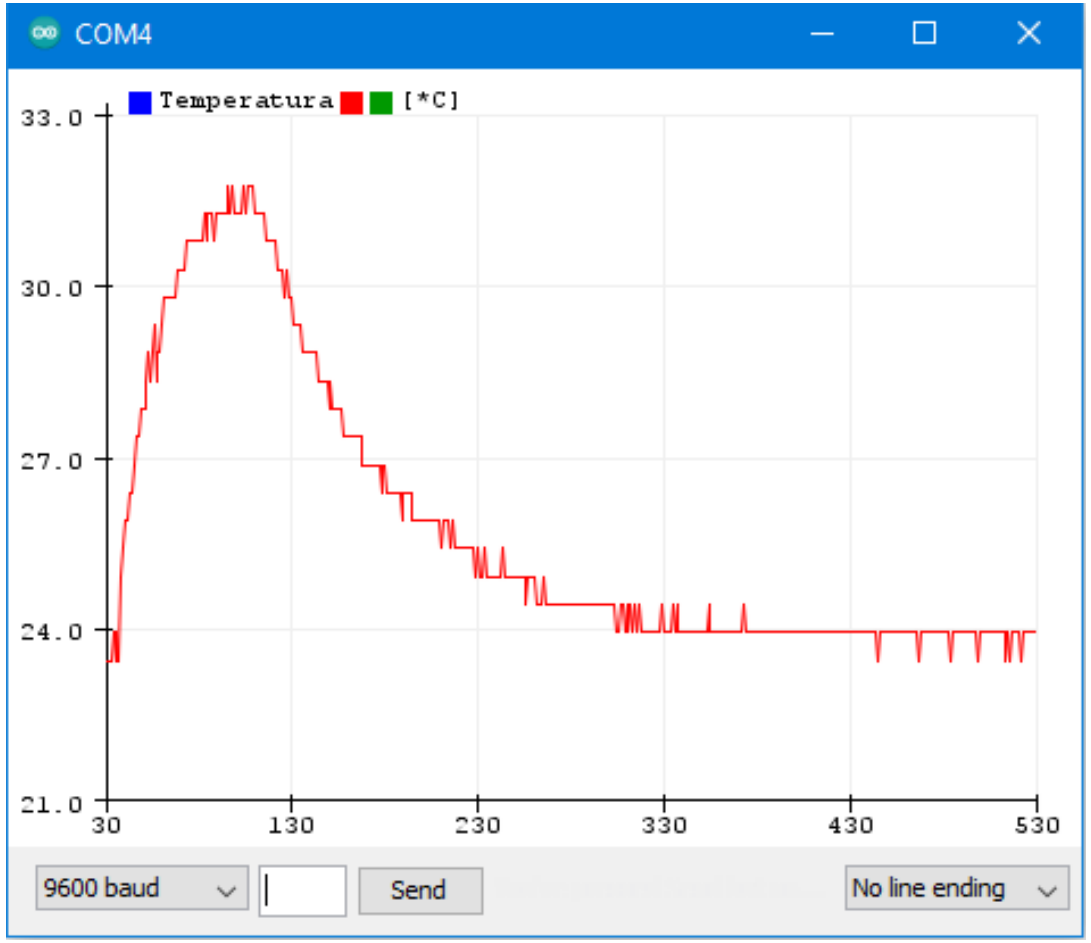


4. Implementarea aplicațiilor – Aplicația nr. 3



4. Implementarea aplicațiilor – Aplicația nr. 3





Consola Serial și afișajul grafic – Variația temperaturii în raport cu timpul

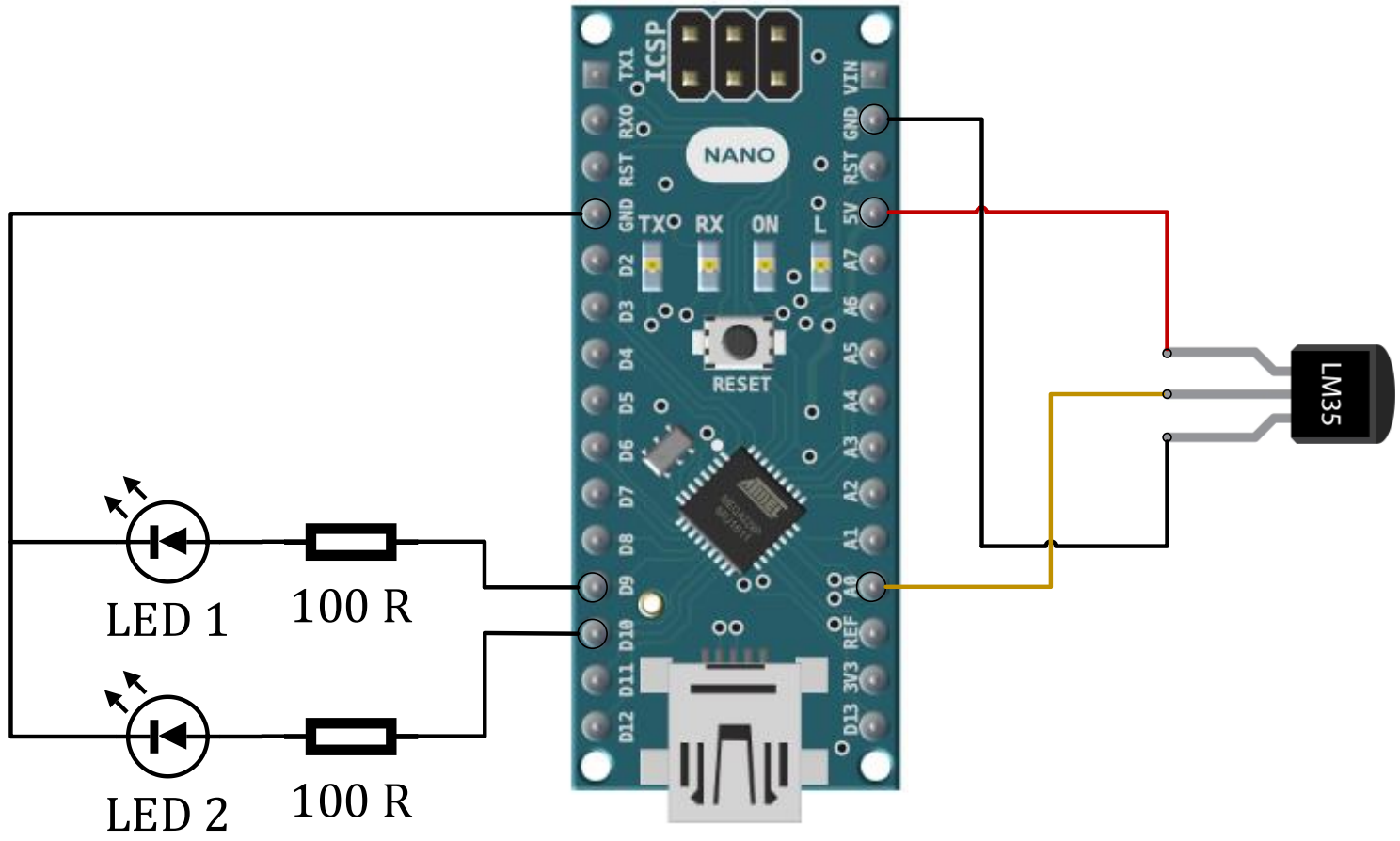
4. Implementarea aplicațiilor – Aplicația nr. 4

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

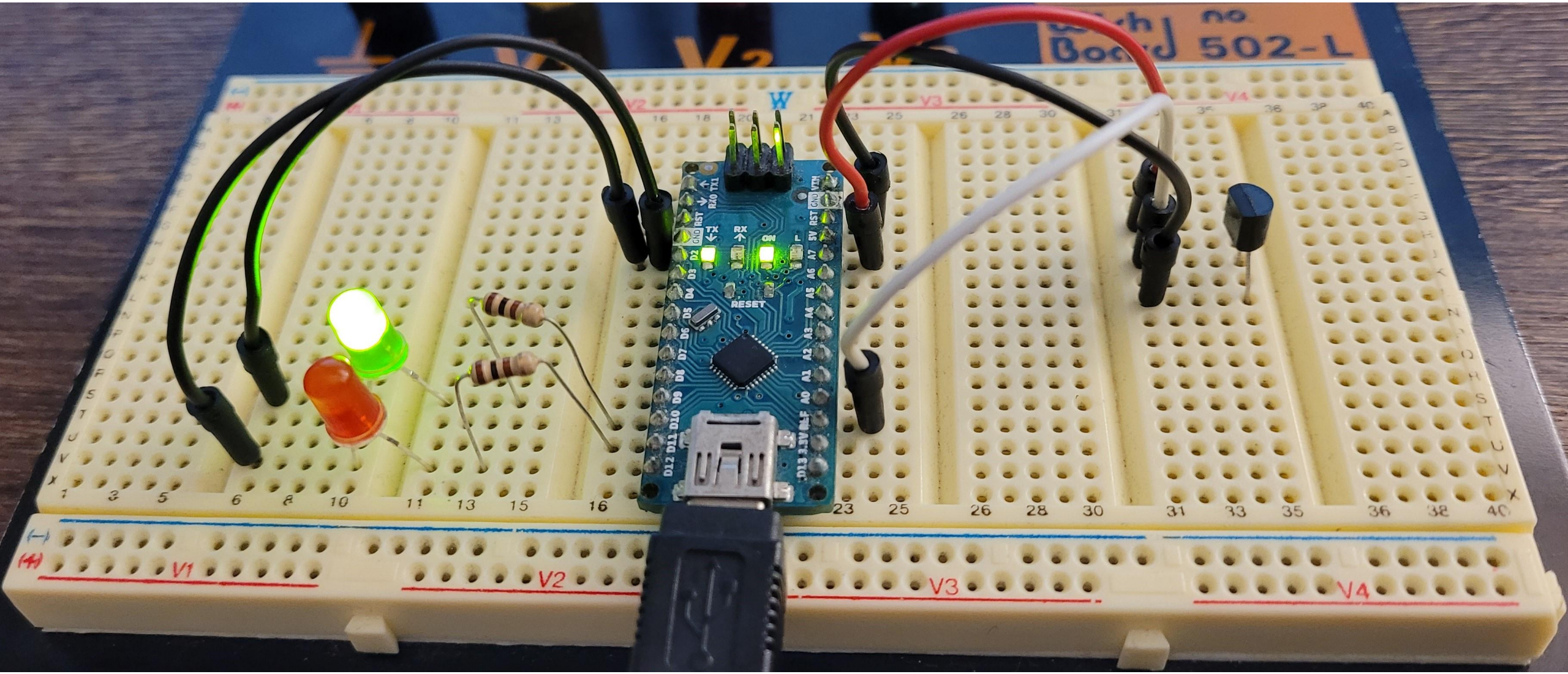
- ✓ declararea unei constante de tip număr întreg „analog_pin” având ca și valoare „0”
- ✓ declararea unei constante de tip număr întreg „led_1” având ca și valoare „9”
- ✓ declararea unei constante de tip număr întreg „led_2” având ca și valoare „100”
- ✓ inițializarea unei variabile de tip număr întreg „ADC_val” 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 număr întreg „prag” cu valoarea „28”
- ✓ stabilirea modului de lucru „ieșire” pentru terminalele digitale „D9” și „D10”
- ✓ inițializarea comunicației Serial la viteza de transfer 9600 [b/s]
- ✓ preluarea valorii 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
- ✓ compararea nivelului temperaturii actuale cu valoarea de prag impusă inițial
- ✓ afișarea în consolă atât a valorii actuale cât și a pragului de temperatură

4. Implementarea aplicațiilor – Aplicația nr. 4

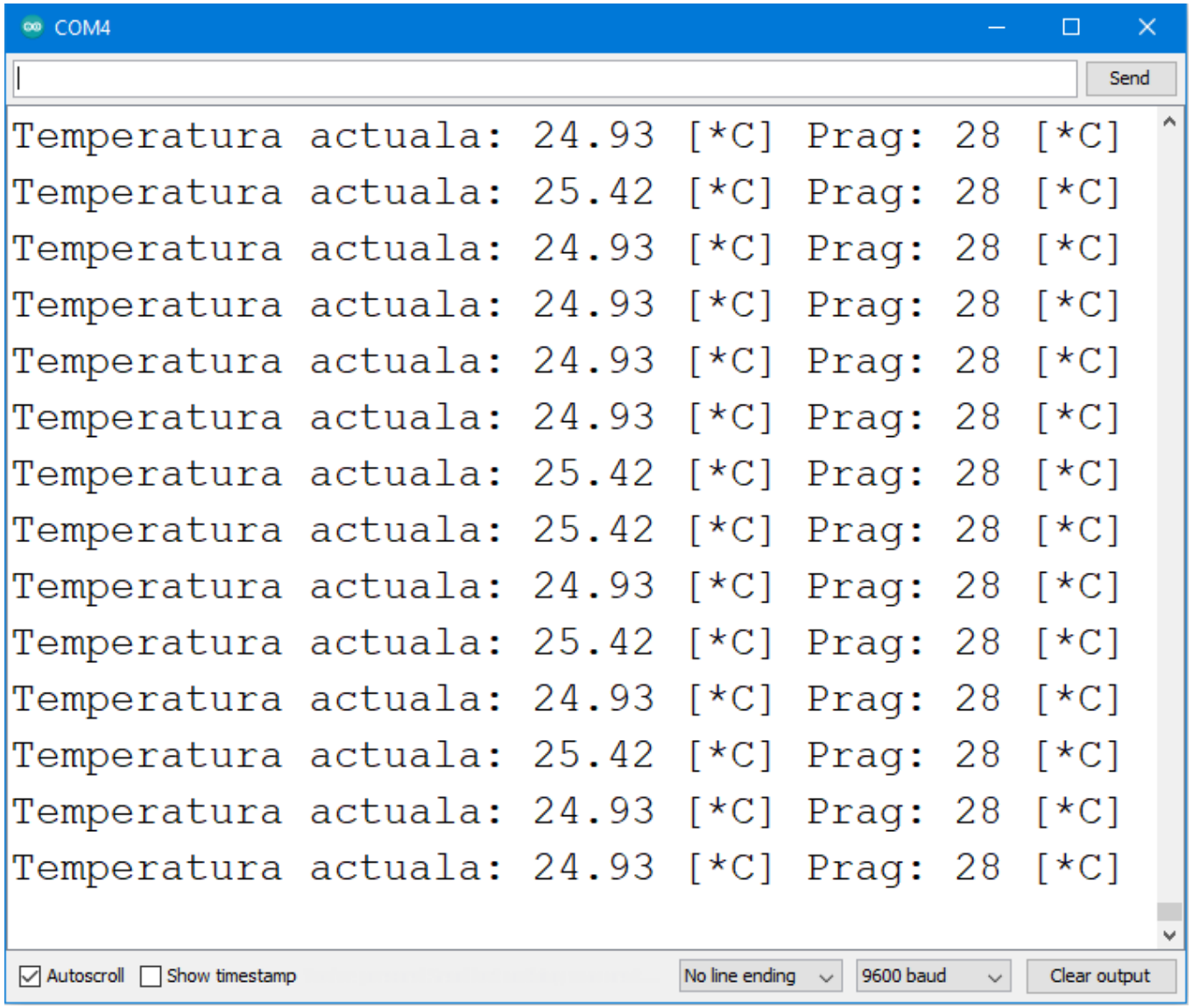
```
const int analog_pin = 0;
const int led_1 = 9;
const int led_2 = 10;
int ADC_val = 0;
float U = 0.00;
float temp = 0.00;
int prag = 28;

void setup() {
  pinMode(led_1, OUTPUT);
  pinMode(led_2, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  ADC_val = analogRead(analog_pin);
  U = (5.00 / 1023.00) * ADC_val;
  temp = 100.00 * U;
  if (temp < prag) {
    digitalWrite(led_1, HIGH);
    digitalWrite(led_2, LOW);
  }
  else {
    digitalWrite(led_1, LOW);
    digitalWrite(led_2, HIGH);
  }
  Serial.print("Temperatura actuala: ");
  Serial.print(temp);
  Serial.print(" [*C]");
  Serial.print(" Prag: ");
  Serial.print(prag);
  Serial.print(" [*C]");
  Serial.println("");
  delay(250);
}
```



4. Implementarea aplicațiilor – Aplicația nr. 4



4. Implementarea aplicațiilor – Aplicația nr. 4



Consola Serial – Variația temperaturii în raport cu valoarea de prag

4. Implementarea aplicațiilor – Aplicația nr. 4

Afișajul grafic Serial – Variația temperaturii în raport cu valoarea de prag impusă inițial

4. Implementarea aplicațiilor – Aplicația nr. 5

➢ **Implementarea aplicației nr. 5 presupune:**

✓ declararea unei constante de tip număr întreg „analog_pin” având ca și valoare „0”

✓ declararea a unui șir finit de opt numere întregi „pin_led []”

✓ inițializarea unei variabile de tip număr întreg „ADC_val” cu valoarea „0”

✓ inițializarea unei variabile de tip număr întreg „nivel” cu valoarea „0”

✓ inițializarea terminalelor digitale din șir în modul de lucru „ieșire digitală”

✓ preluarea valorilor de la convertorul analog – digital

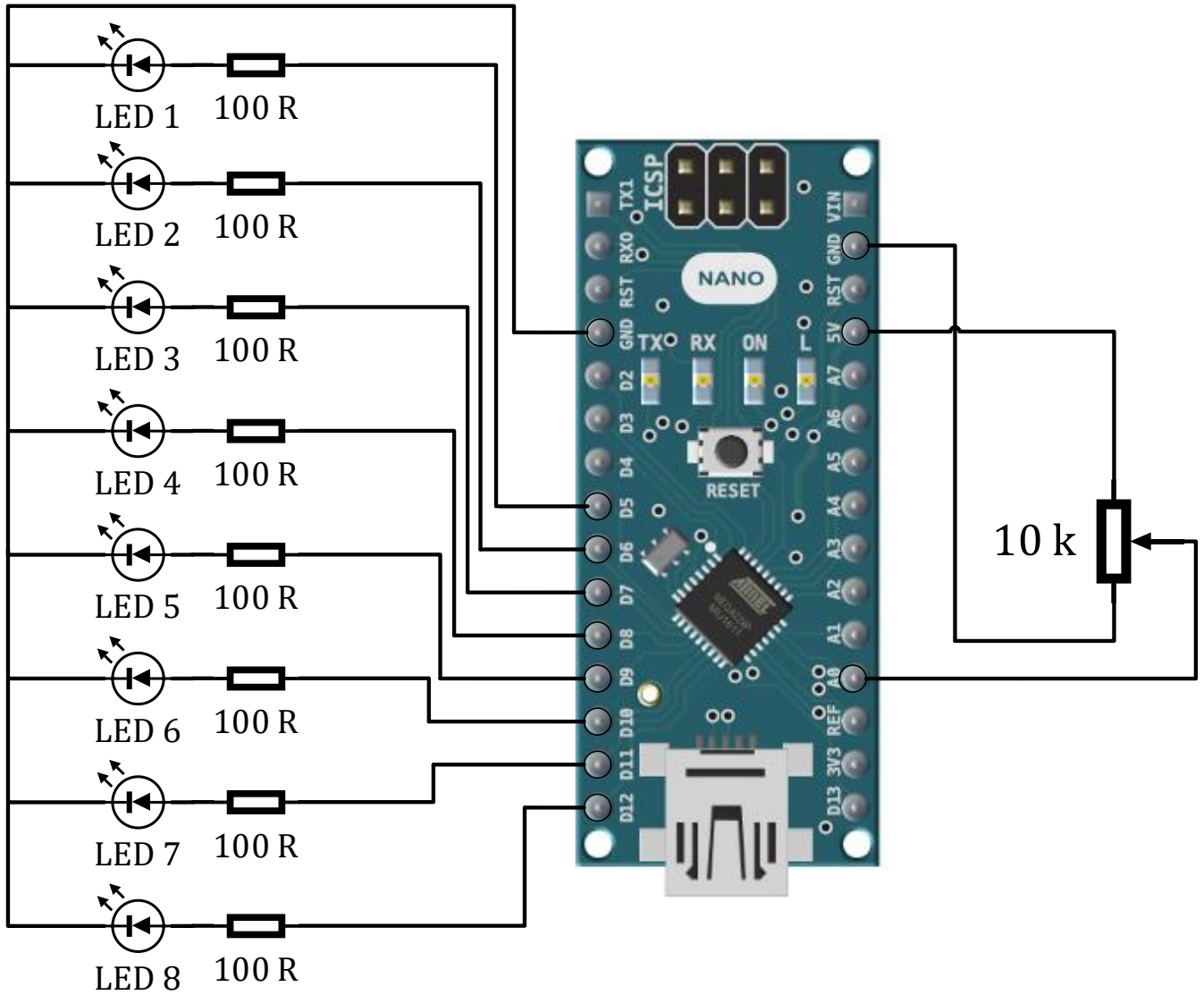
✓ scalarea domeniului de variație al convertorului la numărul de terminale din șir

✓ parcurgerea șirului în sens crescător și activarea stării digitale la fiecare terminal

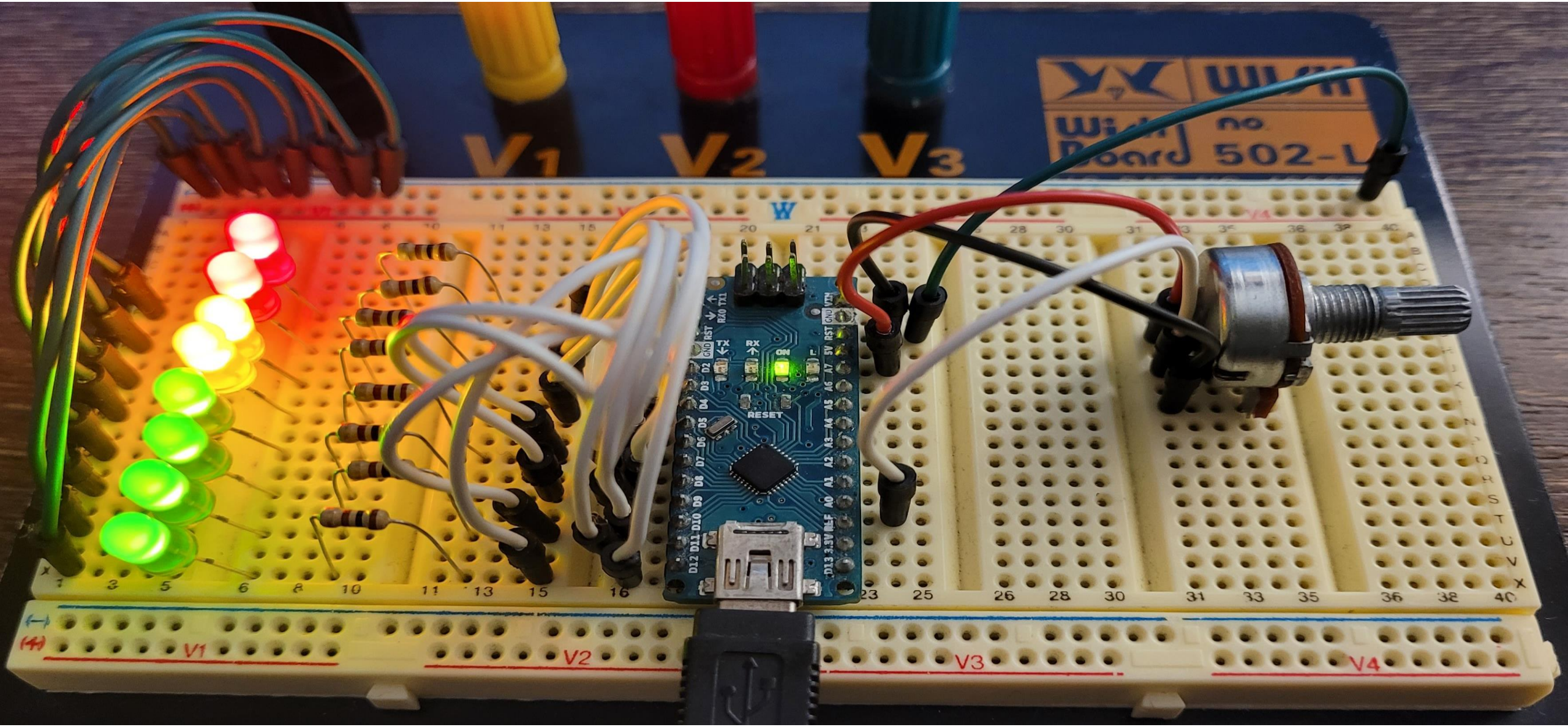
✓ parcurgerea șirului în sens descrescător și dezactivarea stării digitale la fiecare terminal

4. Implementarea aplicațiilor – Aplicația nr. 5

```
const int analog_pin = 0;
int pin_led[] = {5, 6, 7, 8, 9, 10, 11, 12};

int ADC_val = 0;
int nivel = 0;

void setup() {
  for (int i = 0; i <= 7; i++) {
    pinMode(pin_led[i], OUTPUT);
  }
}

void loop() {
  ADC_val = analogRead(analog_pin);
  nivel = map(ADC_val, 0, 1023, 0, 8);
  for (int i = 0; i <= 7; i++) {
    if (i < nivel) {
      digitalWrite(pin_led[i], HIGH);
    }
    else {
      digitalWrite(pin_led[i], LOW);
    }
  }
}
```



4. Implementarea aplicațiilor – Aplicația nr. 5



4. Implementarea aplicațiilor – Aplicația nr. 6

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

✓ declararea unei constante de tip număr întreg „analog_pin” având ca și valoare „0”

✓ declararea unei constante de tip număr întregi „led_1”

✓ inițializarea unei variabile de tip număr întreg „ADC_val” cu valoarea „0”

✓ inițializarea unei variabile de tip număr întreg „dc_1” cu valoarea „0”

✓ inițializarea unei variabile de tip fracționar „dc_1_p” cu valoarea „0.00”

✓ inițializarea terminalului digital „9” în modul de lucru „ieșire digitală”

✓ inițializarea comunicației Serial la viteza de transfer 9600 [b/s]

✓ preluarea valorilor de la convertorul analog – digital

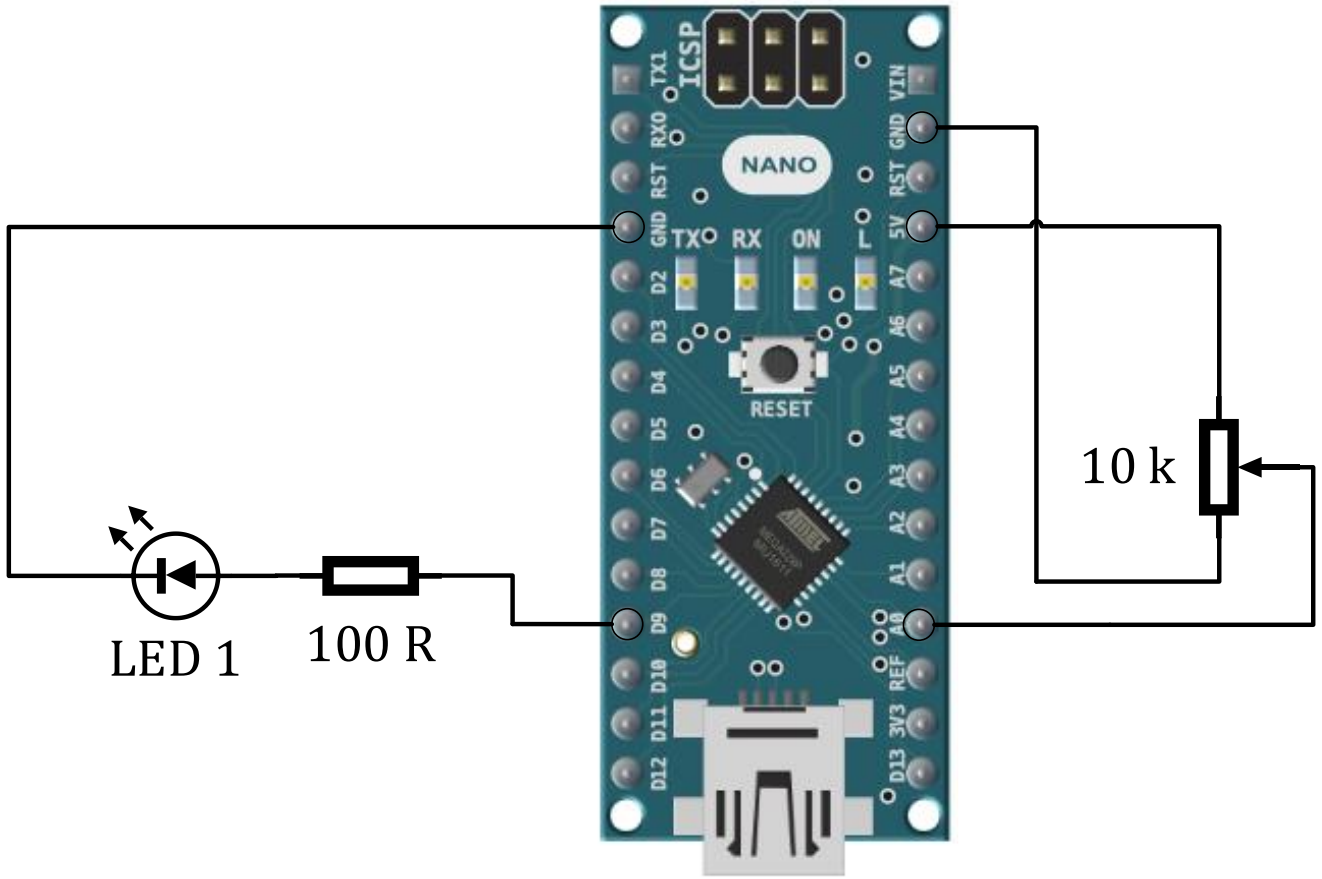
✓ scalarea domeniului convertorului la valoarea maximă reprezentată pe 8 biți

✓ afișarea atât a valorii exprimată pe 8 biți a duratei de conducție cât și procentuală

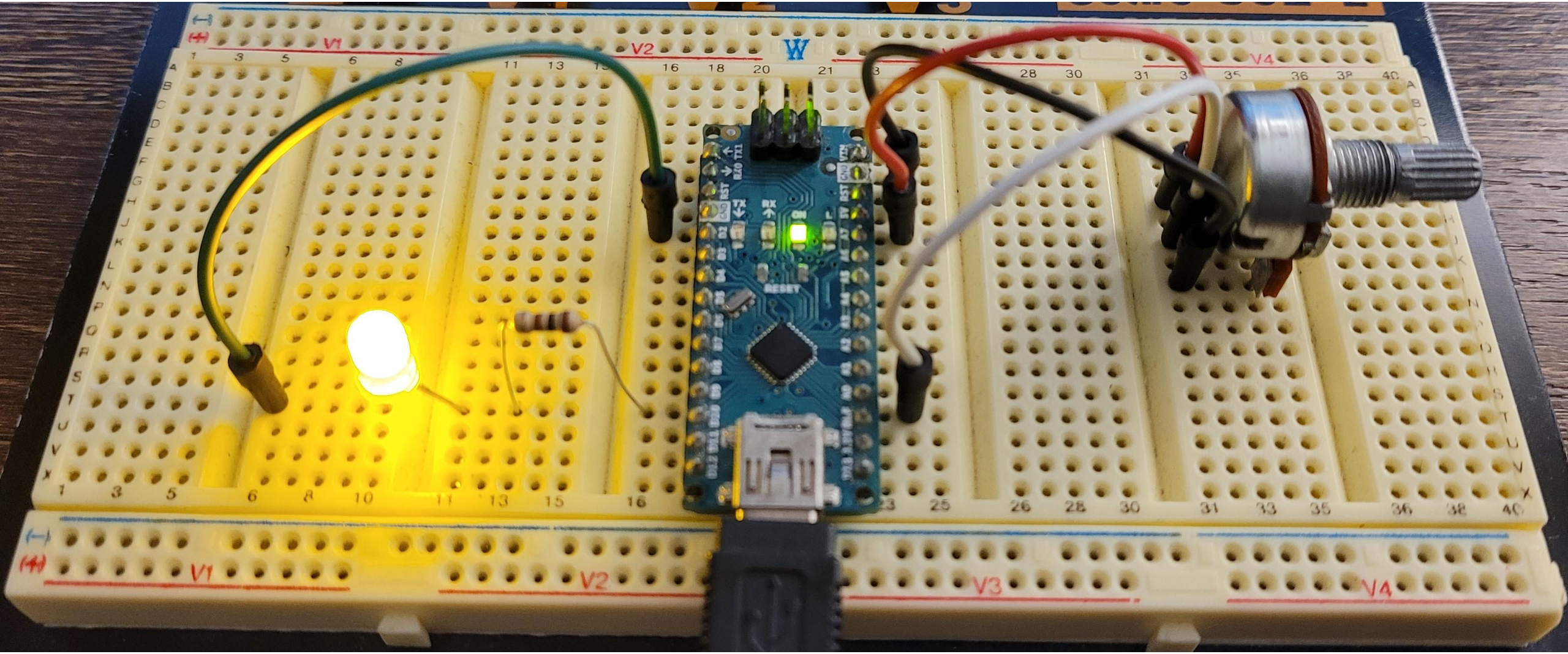
✓ ajustarea factorului de umplere pentru un tren de impulsuri furnizat pe terminalul „9”

4. Implementarea aplicațiilor – Aplicația nr. 6

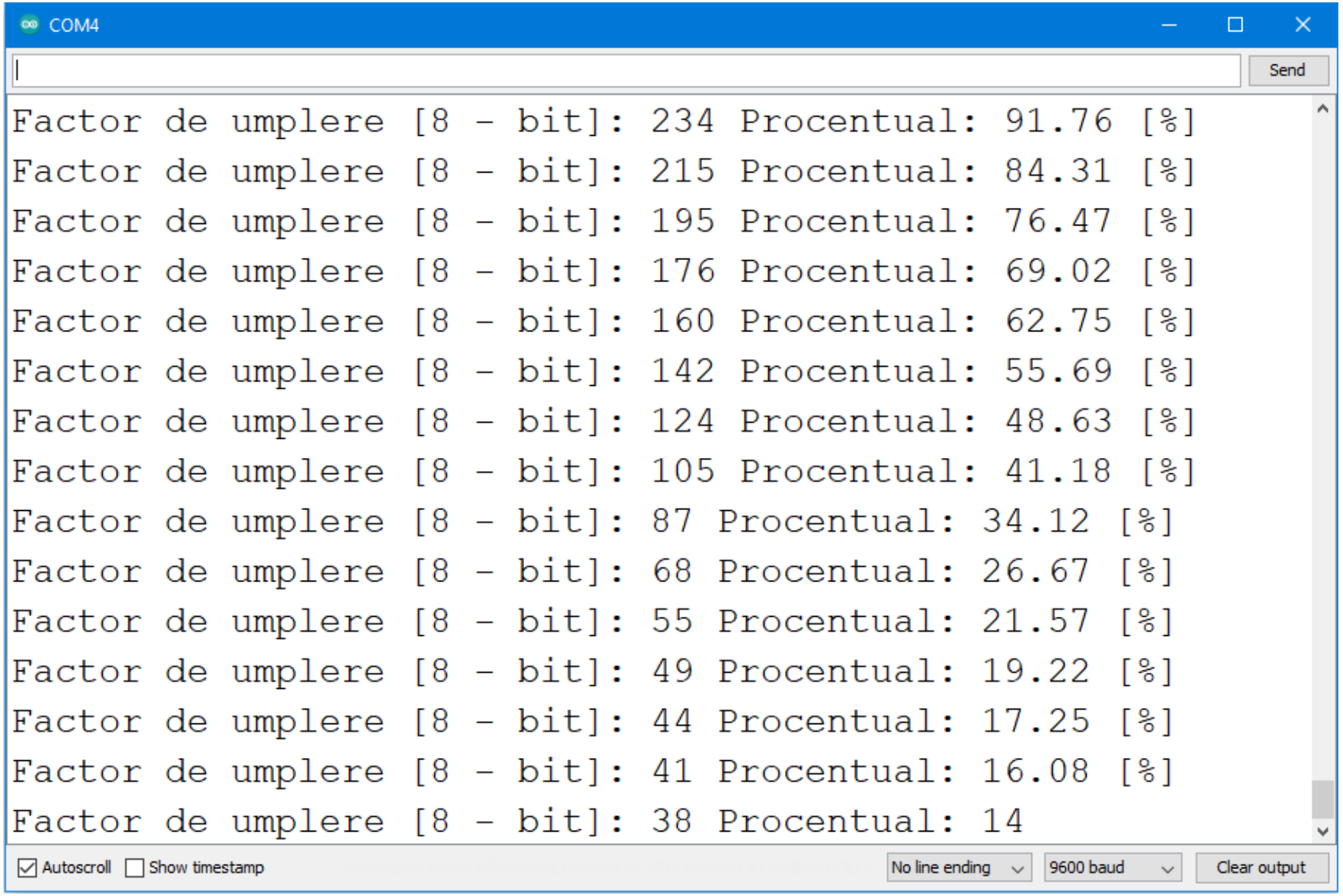
```
const int analog_pin = 0;
const int led_1 = 9;

int ADC_val = 0;
int dc_1 = 0;
float dc_1_p = 0.00;

void setup() {
  pinMode(led_1, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  ADC_val = analogRead(analog_pin);
  dc_1 = map(ADC_val, 0, 1023, 0, 255);
  dc_1_p = ((1.00 / 255.00) * dc_1) * 100;
  Serial.print("Factor de umplere [8 - bit]: ");
  Serial.print(dc_1);
  Serial.print(" Procentual: ");
  Serial.print(dc_1_p);
  Serial.print(" [%]");
  Serial.println("");
  analogWrite(led_1, dc_1);
}
```



4. Implementarea aplicațiilor – Aplicația nr. 5



4. Implementarea aplicațiilor – Aplicația nr. 6



Consola Serial – Afișarea factorului de umplere sau a duratei de conducție

5. Concluzie

- ✓ Procesarea semnalelor atât de natură analogică cât și de natură digitale, poate fi realizată prin intermediul oricărei strategii de implementare a codului program la nivelul microcontrolerului ATMega 328P.

6. Bibliografie

1. Atmel Corporation © 2015, Rev.: 7810D – AVR – 01 / 15 – „ATmega328P datasheet - 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash”;

2. Atmel Corporation © 2015, Rev.: Atmel-11057C-ATARM-SAM3X-SAM3A – „SAM3X / SAM3A Series – Atmel SMART ARM-based MCU datasheet”

3. 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

4. Arduino Store © 2021 Arduino SRL - Partita IVA 09755110963 – „Arduino Nano” https://store.arduino.cc/products/arduino-nano

5. 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/