UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

# UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
# FACULTATEA DE INGINERIE ELECTRICĂ

Facultatea de Inginerie Electrică

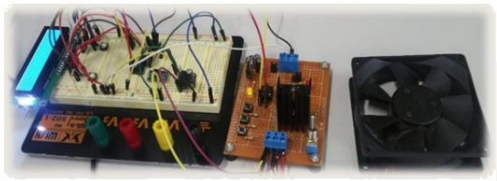
## SISTEME CU MICROPROCESOARE

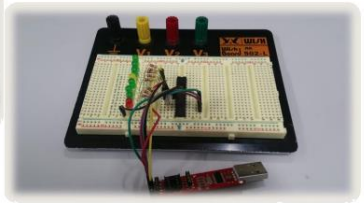
# Ședința de laborator nr. 5

## Protocoale de comunicație utilizate în sisteme cu microprocesoare

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 unor protocoale de comunicație în sisteme microprogramabile [1] [2]
- ✓ Prezentarea unor dispozitive compatibile cu protocoalele de comunicație [3] [4] [5] [6]
- ✓ Utilizarea microcontrolerelor pentru interfațarea protocoalelor de comunicație [5] [6] [7]

# 2. Introducere

- Un **protocol de comunicație**, reprezintă un **mod de asociere** și **transferare de date** între cel puțin **două** sisteme de calcul, pe baza **circuitelor periferice** aflate în dotarea sistemului de calcul.

- **Cadrul de date** vehiculat între cele două echipamente reprezintă orice **combinație alfa-numerică** exprimată prin intermediul protocolului.

- O **grupare de cadre de date**, reprezintă un **pachet de date**. În unele situații, cadrele de date reprezintă **un singur bit.**

# 2. Introducere

➢ În domeniul sistemelor cu microprocesoare, există două categorii principale de protocoale de comunicație, anume:

✓ Protocoale de comunicație cu transmitere de date în **mod paralel**

✓ Protocoale de comunicație cu transmitere de date în **mod serial**

# 2. Introducere

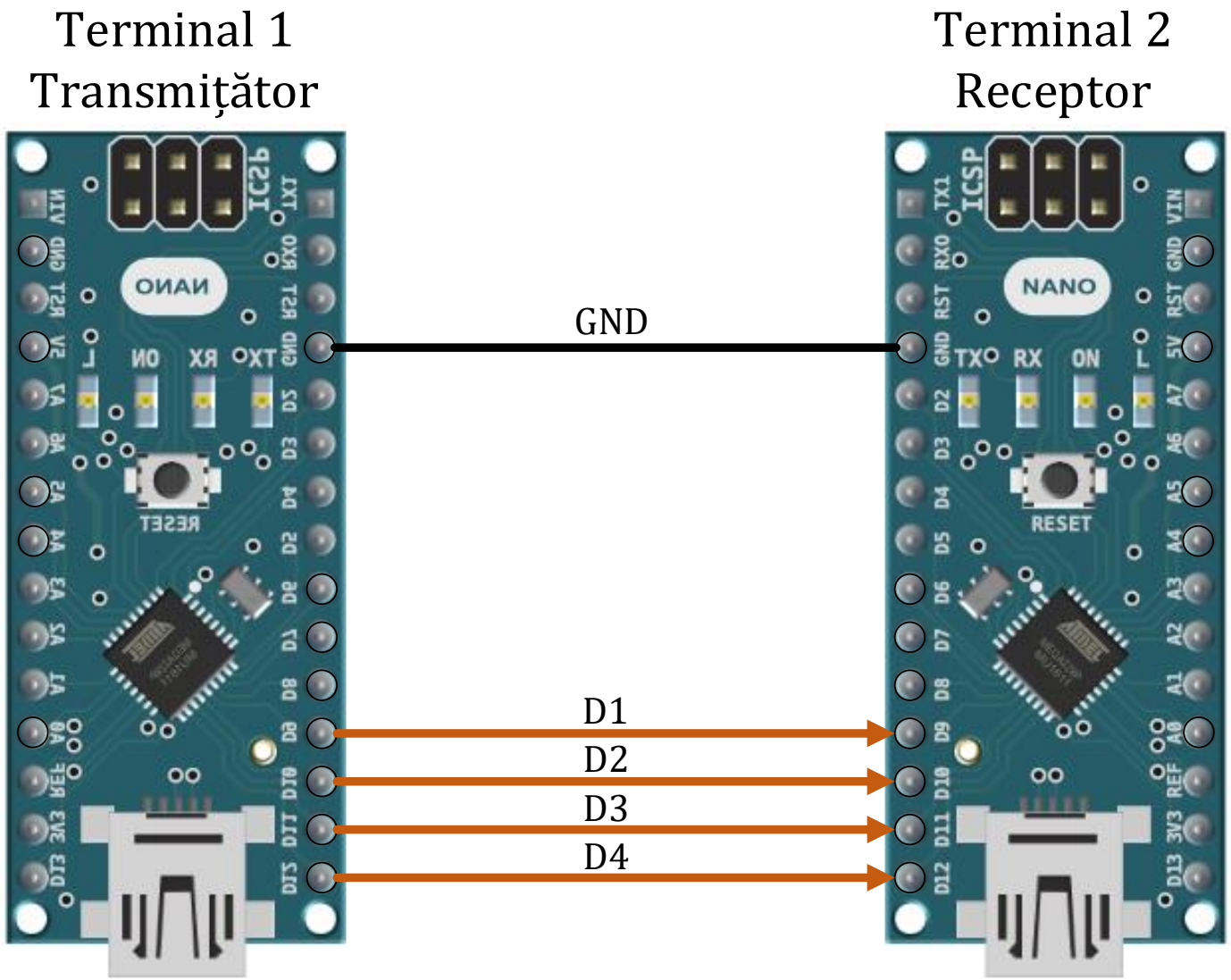
- Protocoalele de comunicație cu transmitere de date în **mod paralel**, pot vehicula cadrele de date pe **mai multe canale de comunicație în mod simultan**.

- Mai precis prin intermediul a patru intrări / ieșiri digitale, se pot vehicula un număr de $2^4 = 16$ combinații valide.

- Pentru a realiza asocierea între două sisteme de calcul de tip microcontroler, **Terminalul 1** va fi configurat prin intermediul codului program, ca și **transmițător**, iar **Terminalul 2** ca și **receptor**.

- Canalele de date digitale vor fi configurate ca și **ieșiri** în cazul **transmițătorului** iar în cazul receptorului ca și **intrări**. Protocolul de date în acest sens, poate fi considerat **unidirecțional**.

# 2. Introducere




Protocolul de comunicație cu transmitere de date în mod paralel

# 2. Introducere

- Protocoalele de comunicație cu transmitere de date în **mod serial**, pot vehicula cadrele de date **pe două canale de comunicație** în **mod succesiv**, **serializat**.

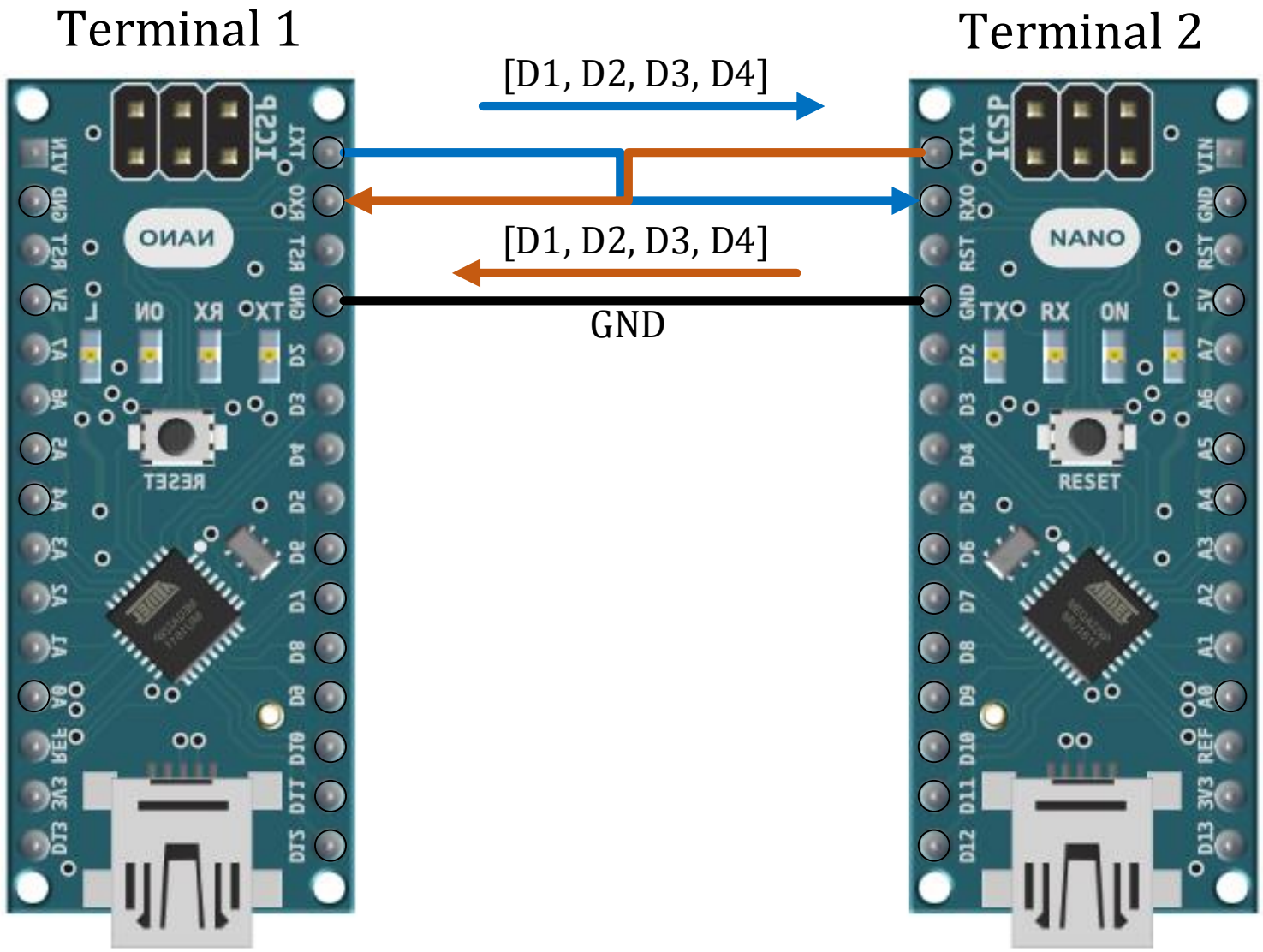
- Terminalele de date, pot avea rolul atât de **transmițător** cât și de **receptor**, astfel, protocoalele de comunicație serial, pot fi **bidirecționale**.

- În protocoalele serial există **două** canale pentru vehicularea informației anume:
  - ✓ „TxD” (eng. Transmit „x” = Any Data – canal pentru transmisia oricărui tip de date)
  - ✓ „RxD (eng. Receive „x” = Any Data – canal pentru recepționarea oricărui tip de date);

- Canalul pentru transmisia oricărui tip de date „TxD” este reprezentat printr-o **ieșire digitală**, iar datele sunt transmise sub formă de **impulsuri** (bit cu bit).

- Canalul pentru recepționarea oricărui tip de date „RxD” este reprezentat printr-o **intrare digitală**, iar datele sunt recepționate în mod serializat (bit cu bit).

# 2. Introducere




Protocolul de comunicație cu transmitere de date în mod serial

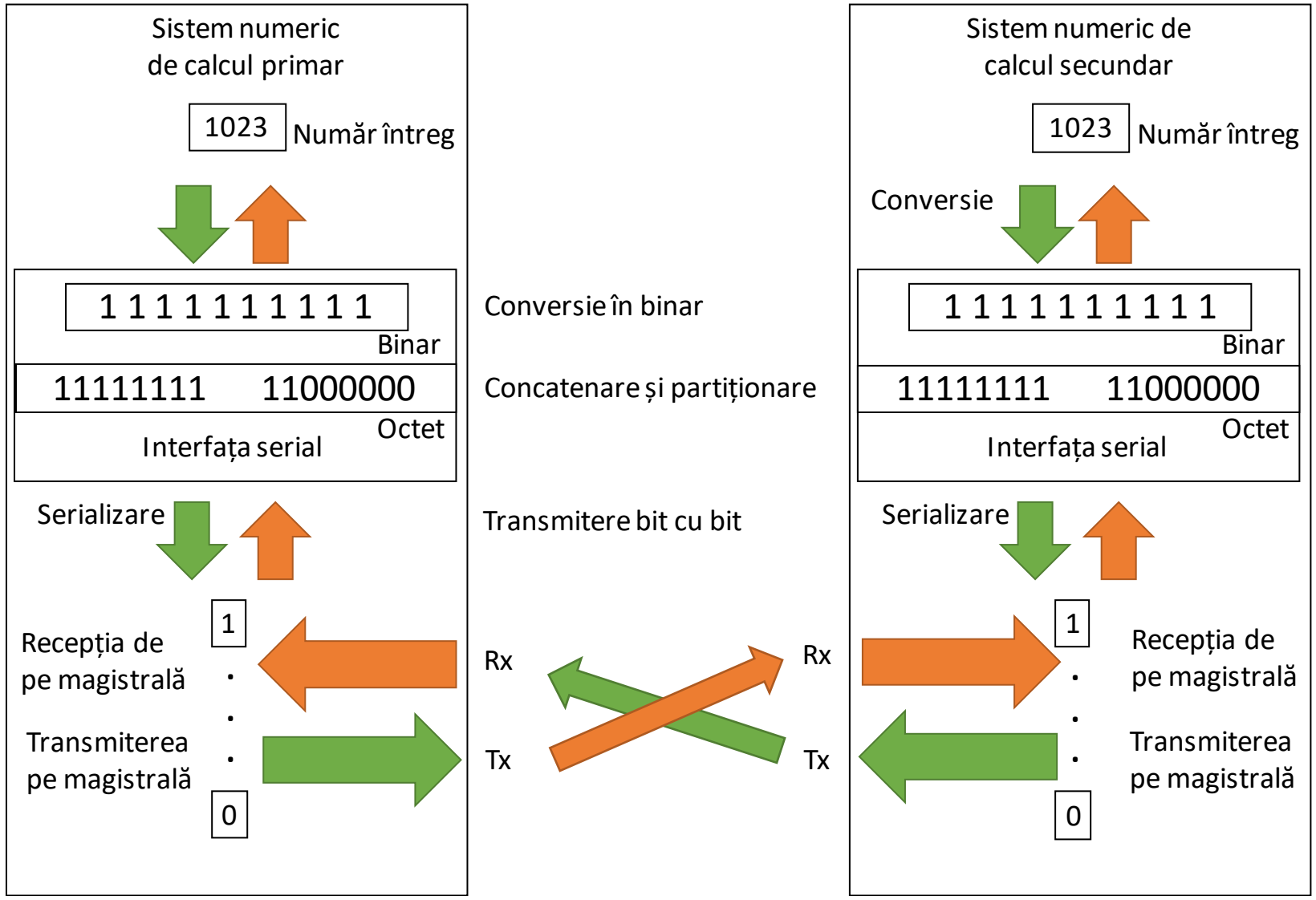
# 2. Introducere

➢ Comunicația de tip serial standard asincron (eng. Universal Asynchronous Receiver – Transmitter – UART), reprezintă **modelul de bază** al tuturor protocoalelor de comunicație de tip serial și poate fi reprezentat la următoarele etape:

✓ Etapa de transmitere și presupune:
- Conversia în binar a tipului de date care urmează a fi transmis
- Partiționarea informației binare rezultante în grupuri de opt biți (adică un byte)
- Transmiterea informației în mod serializat (succesiv) a fiecărui bit rezultant

✓ Etapa de recepționare și presupune:
- recepționarea în mod serializat (succesiv) a fiecărui bit component al informației
- concatenarea informației în grupuri a câte opt biți (adică un byte)
- mascarea biților care nu fac parte din cuvântul de date și conversia datelor în zecimal

# 2. Introducere




Modelul pentru vehicularea datelor în procesul de comunicație serial

# 3. Aspecte teoretice

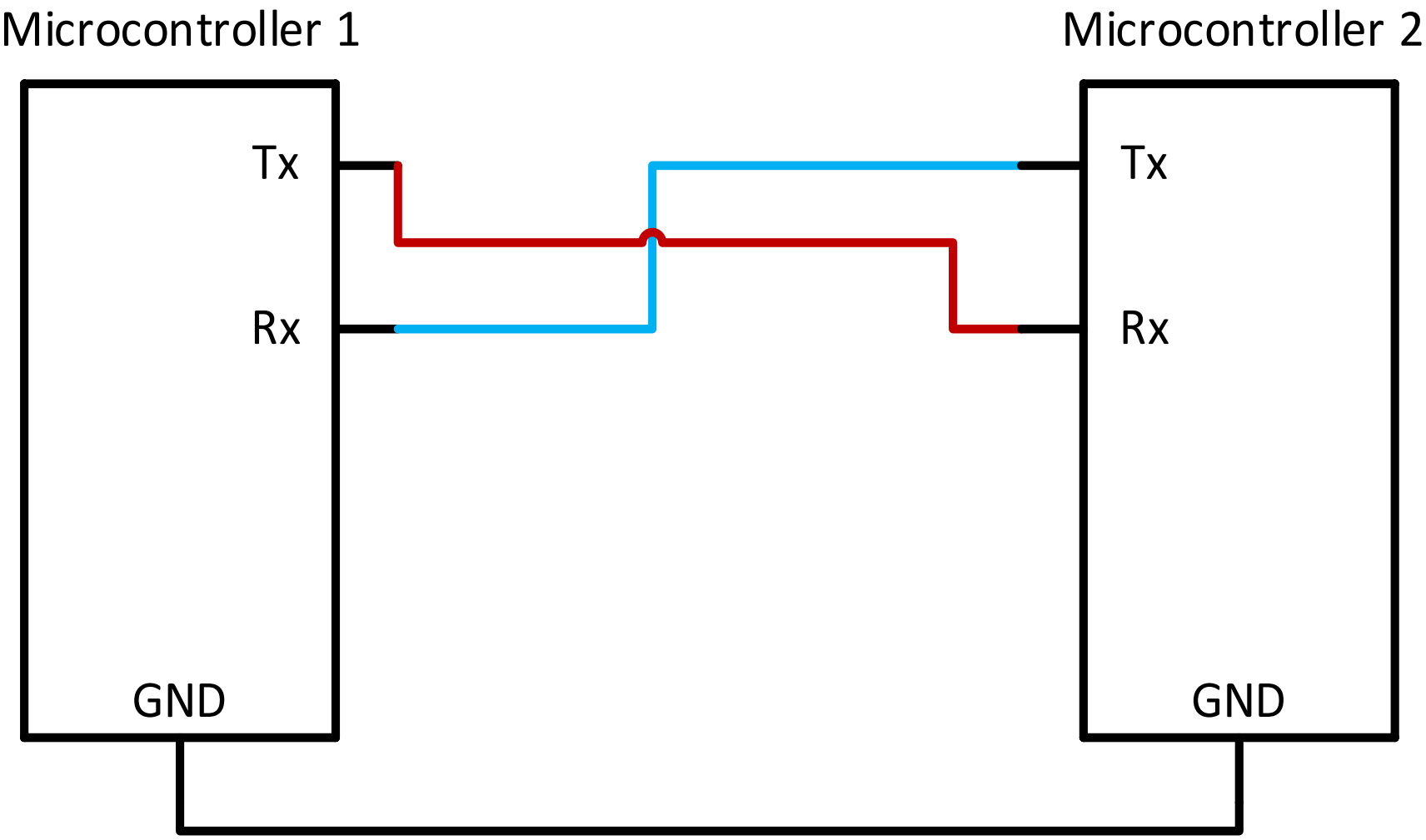
- În domeniul sistemelor microprogramabile, cele mai des utilizate protocoale de tip serial sunt:

  - ✓ UART – (eng. Universal Asynchronous Receiver – Transmitter) – Serial asincron
  - ✓ I2C – (eng. Inter – Integrated Circuit) – Serial sincron adresabil
  - ✓ SPI – (eng. Serial Peripheral Interface) – Serial sincron pentru dispozitive periferice
  - ✓ 1 - Wire – One Wire – Serial asincron

# 3. Aspecte teoretice

- **Protocolul Serial standard asincron UART:**
  - ✓ reprezintă un protocol **fizic** (eng. **hardware**) de comunicație
  - ✓ se realizează pe **două fire**: „Tx” – eng. Transmit Data și „Rx” – eng. Receive Data
  - ✓ face parte din clasa protocoalelor „**punct la punct**” (eng. Point To Point – PTP), adică, protocolul nu funcționează decât între **cel puțin și cel mult două terminale**
  - ✓ poate fi optimizat pentru a vehicula datele pe **distanțe medii** între două terminale, însă **volumul de date este redus**
  - ✓ poate funcționa în regim **bidirecțional**

# 3. Aspecte teoretice



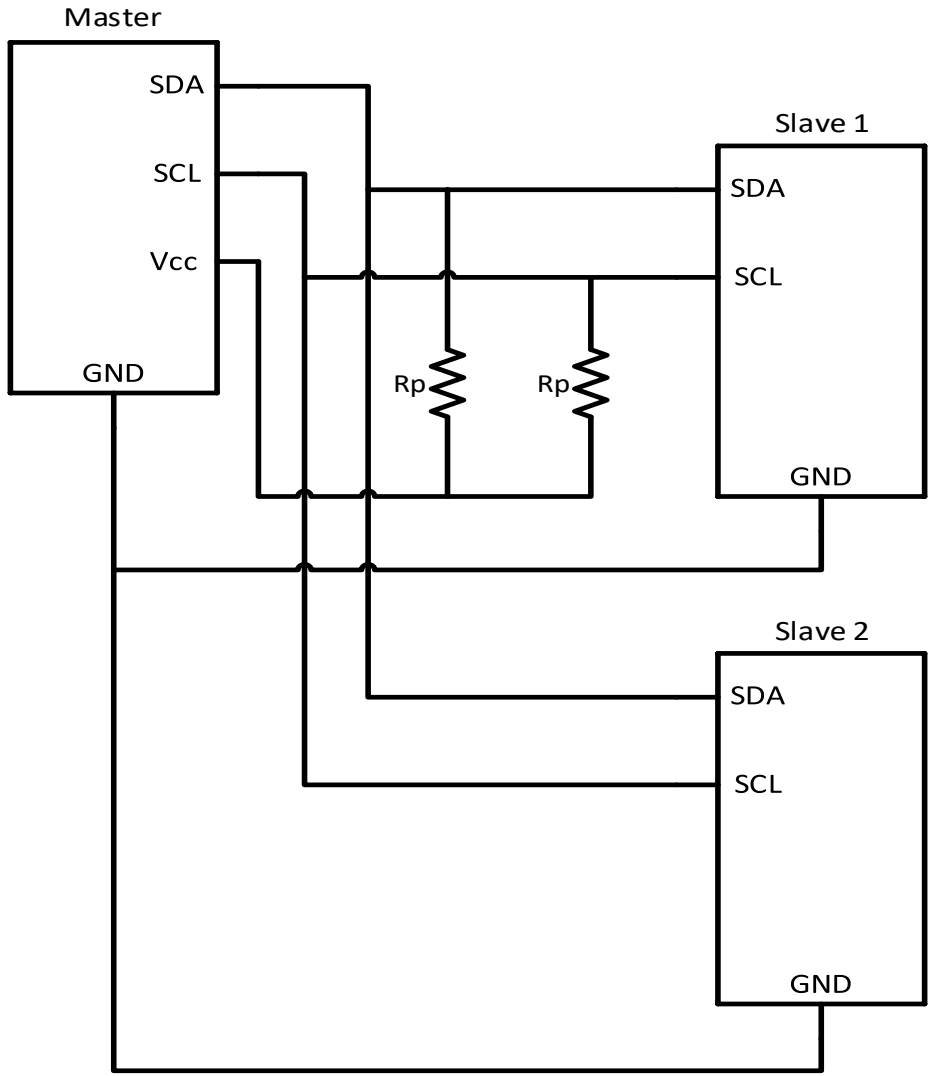

Conexiunea între terminale în cadrul protocolului serial asincron UART

# 3. Aspecte teoretice

❖ **Protocolul Serial sincron adresabil $I^2C$:**

- ✓ reprezintă un protocol **fizic** (eng. **hardware**) de comunicație;
- ✓ se realizează pe **două fire**: „SCL” – eng. **S**erial **CL**ock și „SDA” – eng. **S**erial **DA**ta
- ✓ face parte din clasa protocoalelor **adresabile** de tip eng. „**Master – Slave**”, și poate deservi **mai mult decât două terminale**. Echipamentele sunt cuplate **în paralel** pe o **magistrală comună**, având **adrese distincte**
- ✓ poate vehicula datele pe distanțe relativ reduse, în volum relativ mediu, la viteze cuprinse în intervalul 100 [kb / s] până la 3,5 [Mb / s]
- ✓ în vederea generării unei diferențe de potențial la nivelul canalelor de comunicație sunt necesare două rezistențe pentru conectare la potențialul sursei (eng. pull – up);

# 3. Aspecte teoretice




Conexiunea între terminale în cadrul protocolului serial sincron $I^2C$

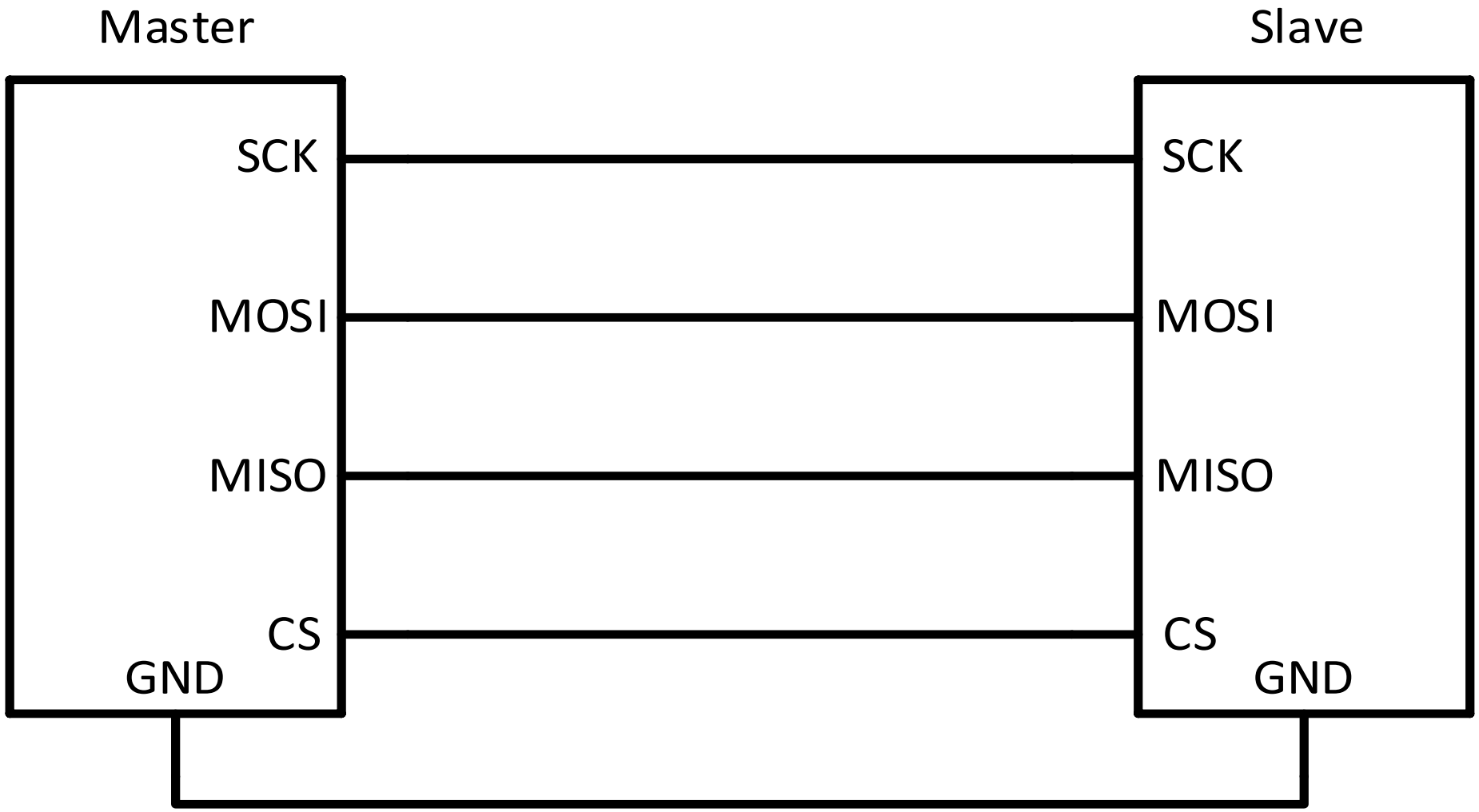
# 3. Aspecte teoretice

## ❖ Protocolul Serial standard sincron SPI:

- ✓ reprezintă un protocol **fizic** (eng. **hardware**) de comunicație;
- ✓ se realizează pe **patru** canale de transmiterea datelor:
  - ➢ „SCLK” – eng. **S**erial **CL**oc**K**
  - ➢ „MOSI” – eng. **M**aster **O**ut **S**lave **I**n
  - ➢ „MISO” – eng. **M**aster **I**n **S**lave **O**ut
  - ➢ „SS” / „CS” – eng. **S**lave **S**elect / eng. **C**hip **S**elect

- ✓ face parte din clasa **protocoalelor adresabile hibride**, adică poate funcționa atât în modul de conectare **punct la punct** cât și în modul eng. „**Master – Slave**”, și poate deservi **mai mult decât două terminale**
- ✓ echipamentele sunt cuplate în **paralel** pe o **magistrală comună**
- ✓ adresarea în cazul protocolului SPI poate fi **opțională**
- ✓ este utilizat pentru a vehicula datele pe **distanțe relativ reduse**, în volum **relativ mare**, la viteze de maxim 60 [Mb / s];

# 3. Aspecte teoretice



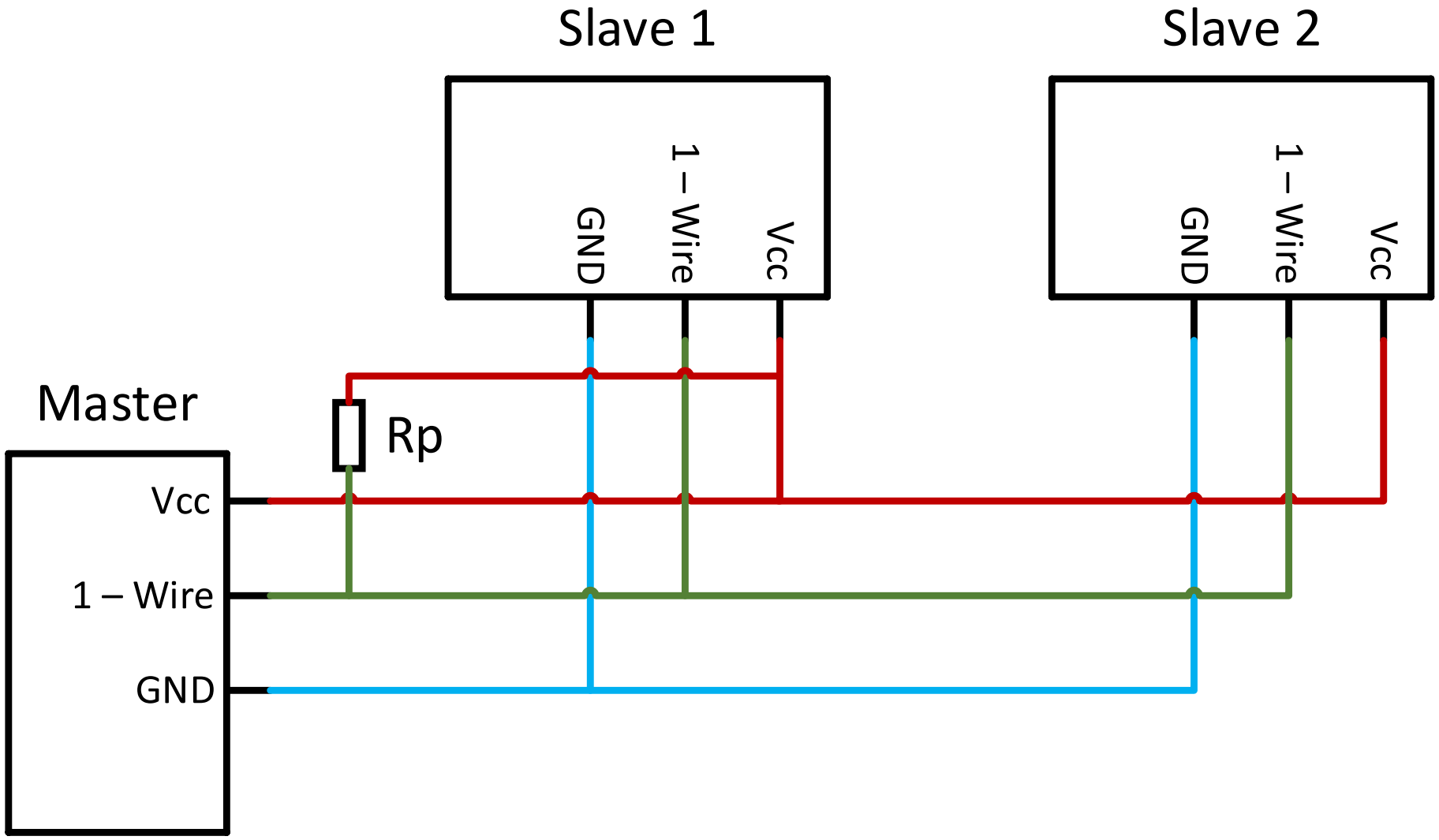

Conexiunea între terminale în cadrul protocolului serial sincron SPI

# 3. Aspecte teoretice

❖ **Protocolul Serial asincron 1-Wire (One Wire):**

- ✓ reprezintă un protocol **fizic** (eng. **hardware**) de comunicație;
- ✓ se realizează prin intermediul **unui singur canal** de transmiterea datelor
- ✓ funcționează în modul **SMMS** (eng. „Single Master and Multiple Slaves”);
- ✓ funcționează în modul (eng. **half - duplex**), adică, **un singur dispozitiv** este capabil să **inițieze procesul de comunicație** și să **transfere date** în mod **bidirecțional în același timp.** Echipamentele sunt cuplate în **paralel** pe o **magistrală comună**
- ✓ prezintă viteză de transfer redusă (aprox. 15,4 [kb / s])
- ✓ poate transfera un volum redus de date

# 3. Aspecte teoretice

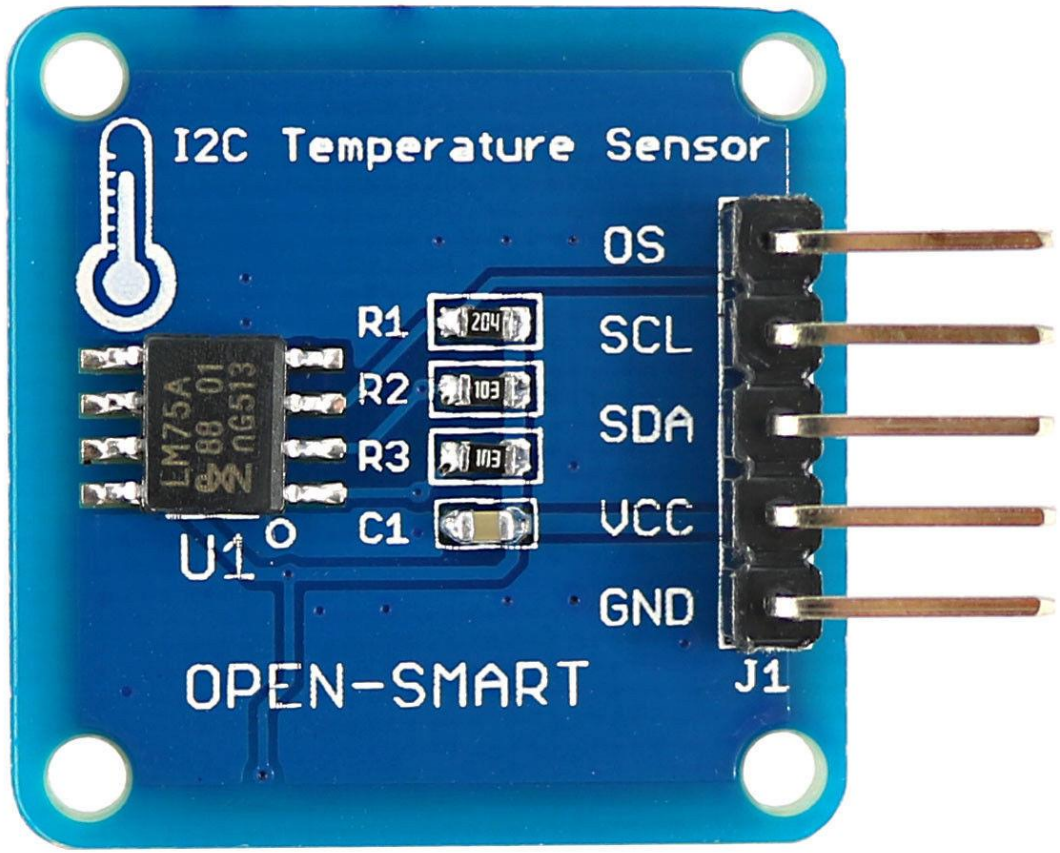



Conexiunea între terminale în cadrul protocolului serial asincron 1 - Wire

# 3. Aspecte teoretice

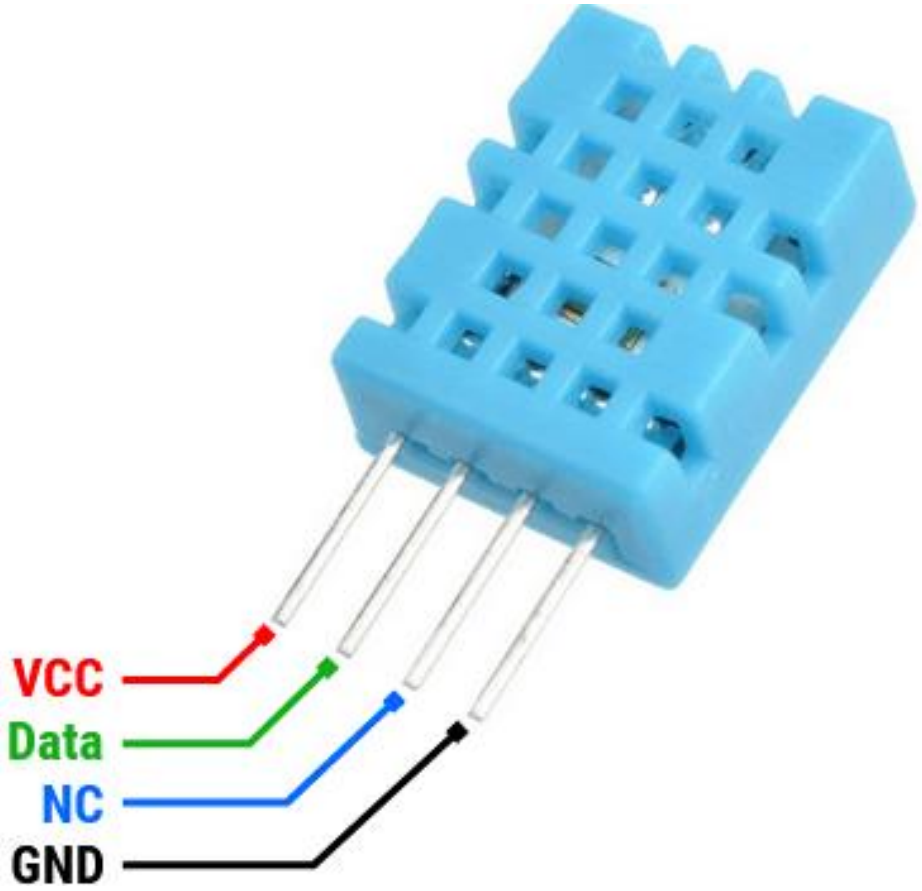
- **Există situații în care, se utilizează un microcontroler pentru comunicație:**
  - ✓ atunci când sistemul de calcul principal nu dispune de convertor analog - digital
  - ✓ atunci când este necesară transferarea datelor la distanțe mari în format digital
  - ✓ atunci când este necesară trecerea de la un protocol de comunicație la altul
  - ✓ atunci când este necesară implementarea unui sistem distribuit în sub-ansamble
  - ✓ atunci când este necesar un consum redus de putere
- În acest sens, pe baza microcontrolerului ATTiny 85 sau 45 în format DIP-8, pot fi implementate diverse soluții de comunicație cu sistemul de calcul principal
- În mod similar, microcontrolerele care există în construcția traductoarelor de tip MEMS (eng. Micro Electro Mechanical Systems), realizează funcția de comunicație.

# Exemplu de aplicație – senzor de temperatură pe interfața $I^2C$



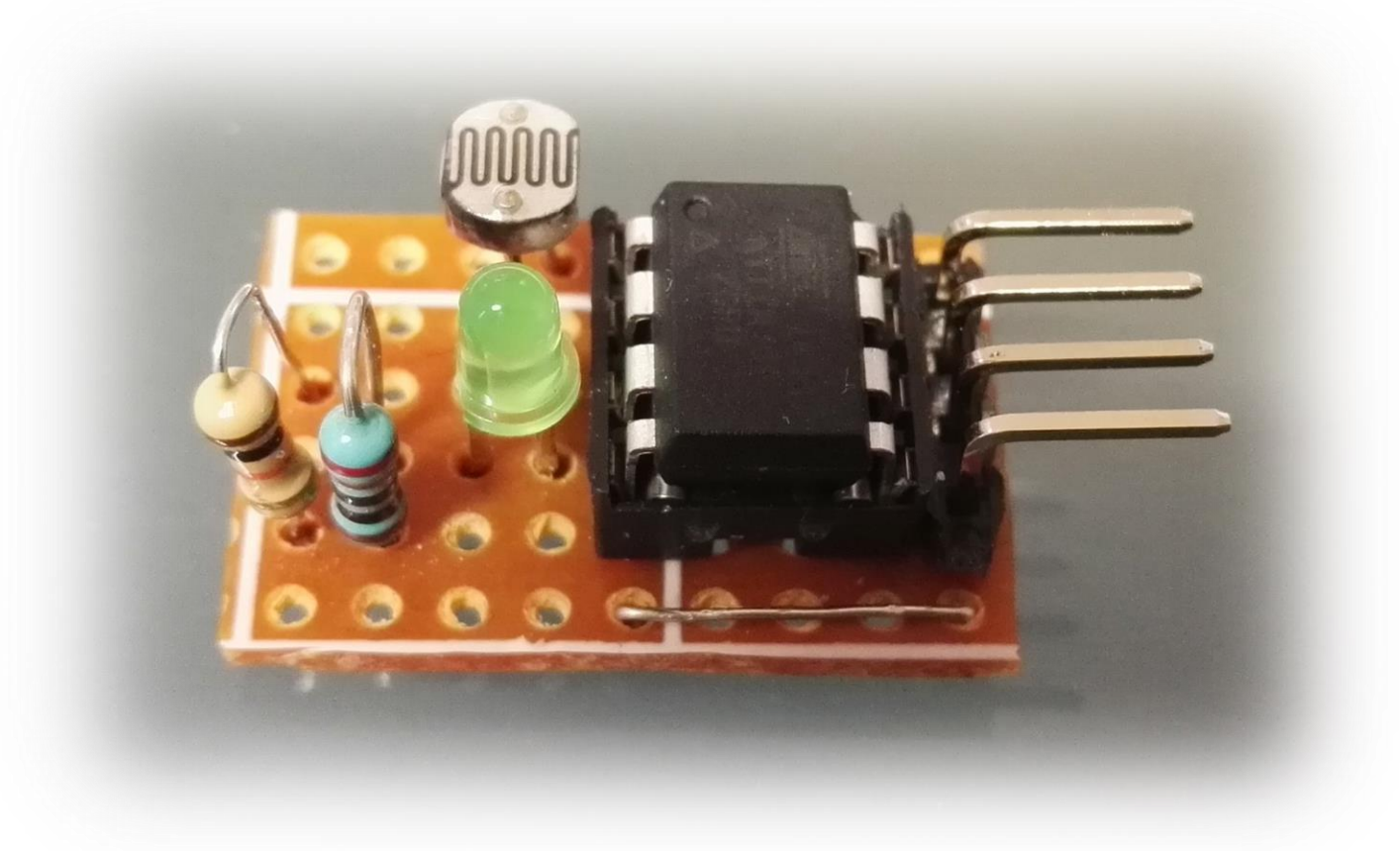

https://www.ebay.com/itm/202103666134

# Traductorul pentru umiditate și temperatură DHT-11



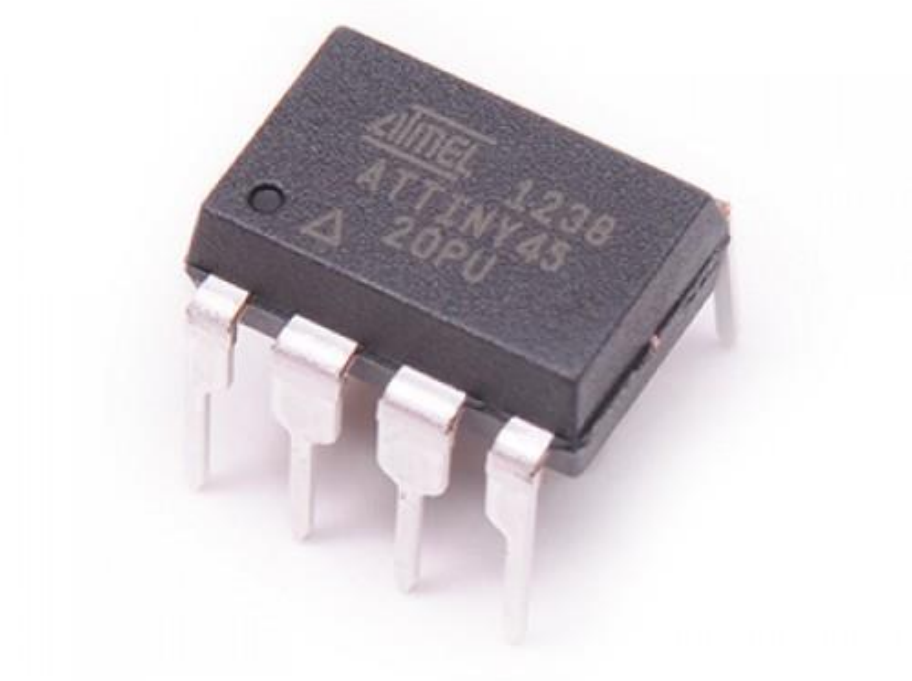

https://www.circuitgeeks.com/arduino-dht11-and-dht22-sensor-tutorial/

# Exemplu de aplicație – senzor de lumină pe interfața $I^2C$



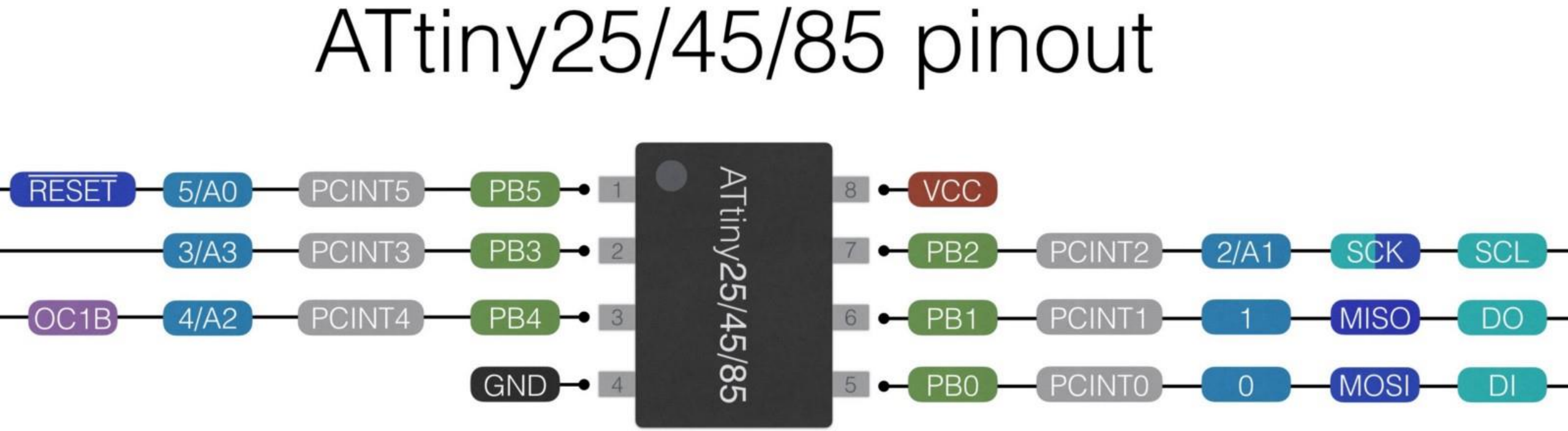
https://quadmeup.com/attiny85-light-sensor-i2c-slave-device/

# Micro – controllerul ATTiny 45



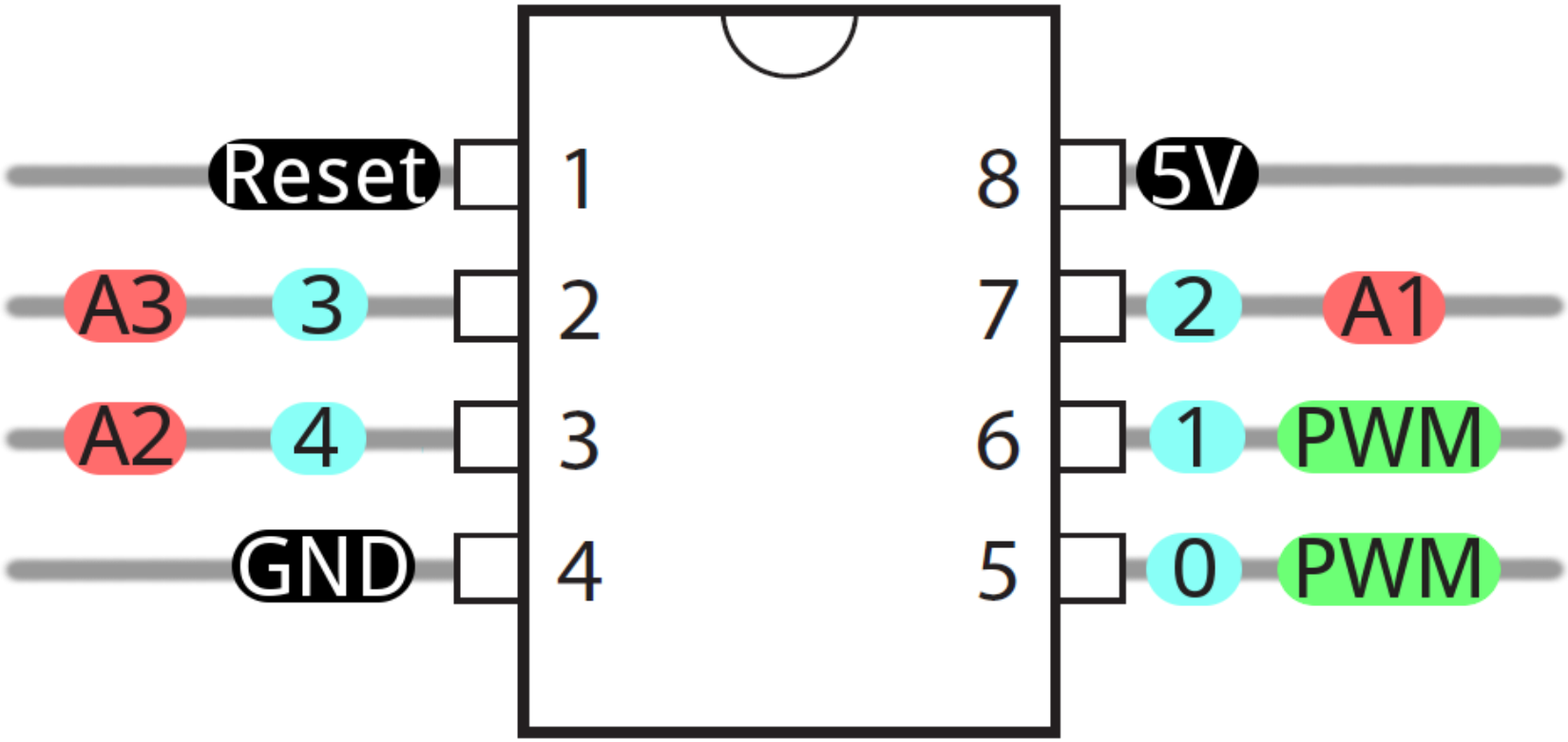

https://www.indiamart.com/proddetail/eeprom-ic-19487422955.html

# Terminale de acces



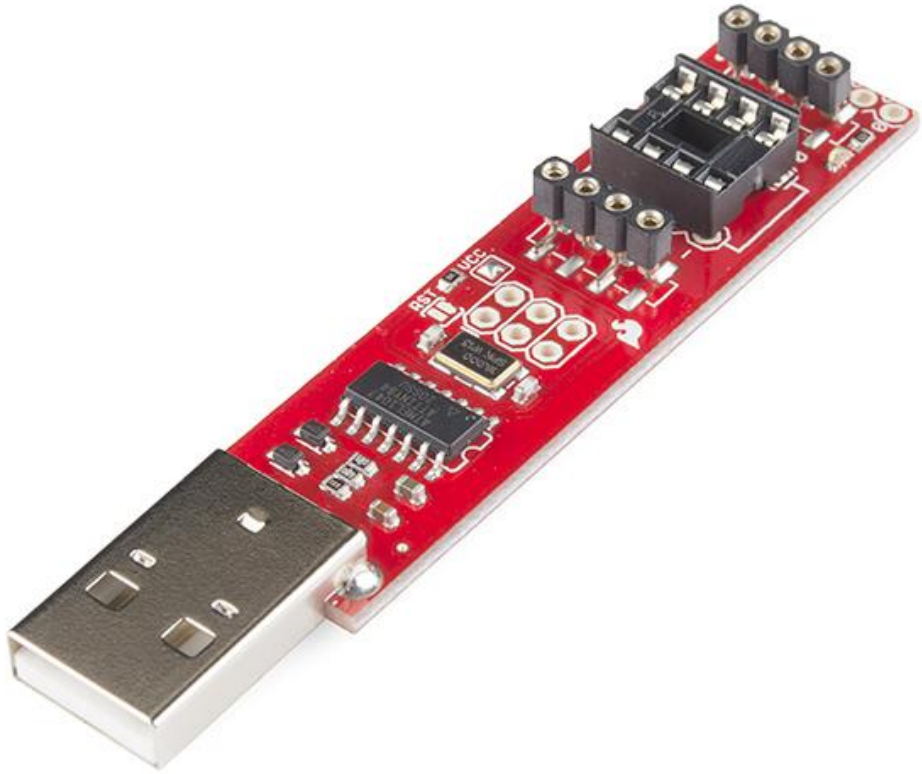

https://medium.com/jungletronics/attiny85-easy-flashing-through-arduino-b5f896c48189

# Terminale de acces



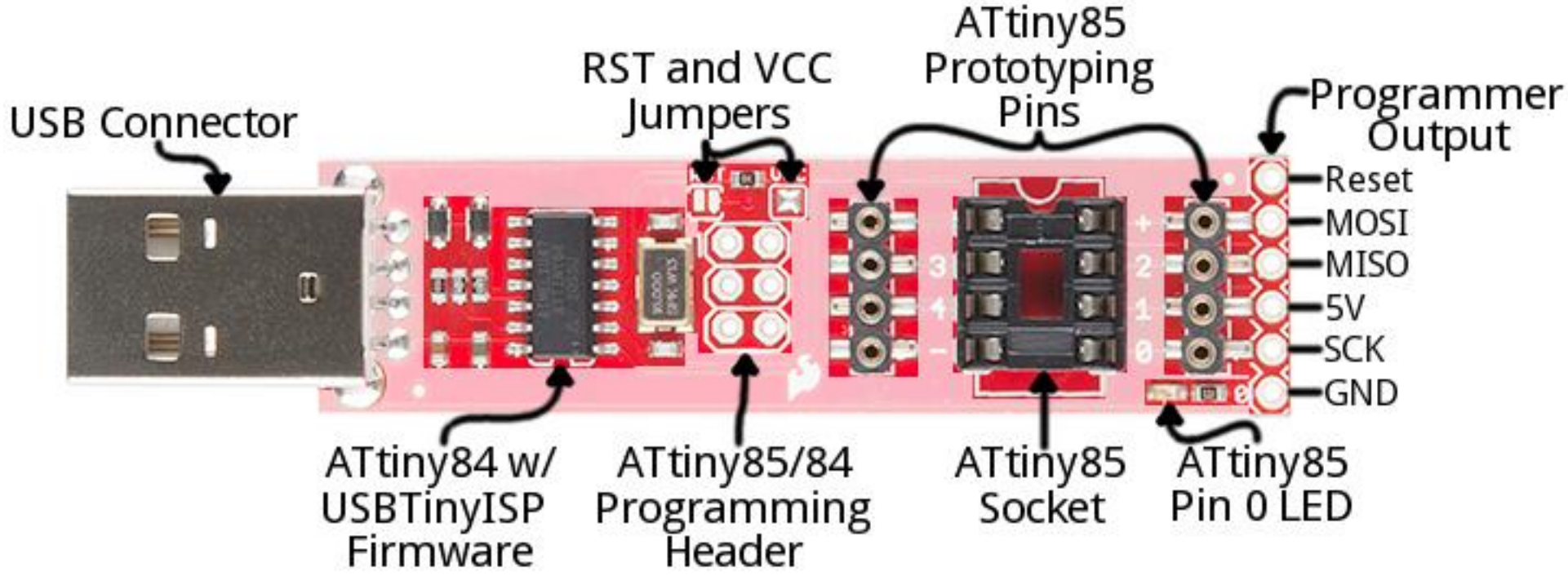

https://learn.sparkfun.com/tutorials/tiny-avr-programmer-hookup-guide/attiny85-use-hints

# Programatorul SPI SparkFun USBTinyISP - TinyAVR



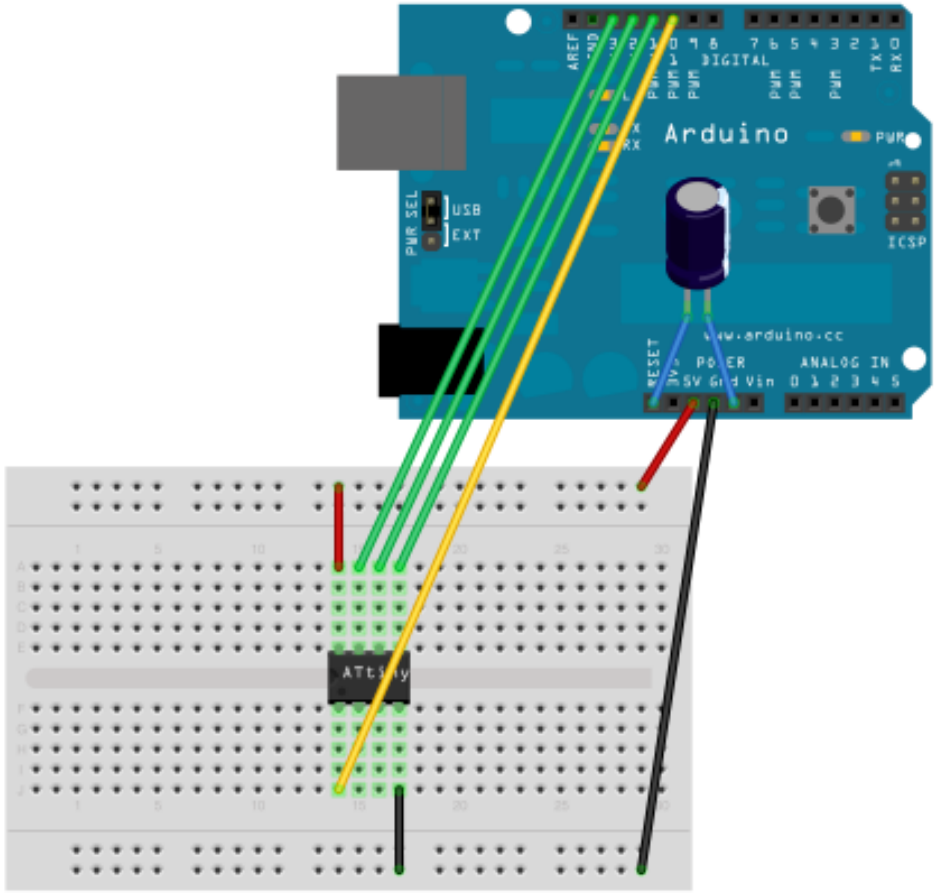

https://www.sparkfun.com/products/11801

# Programatorul SPI SparkFun USBTinyISP - TinyAVR



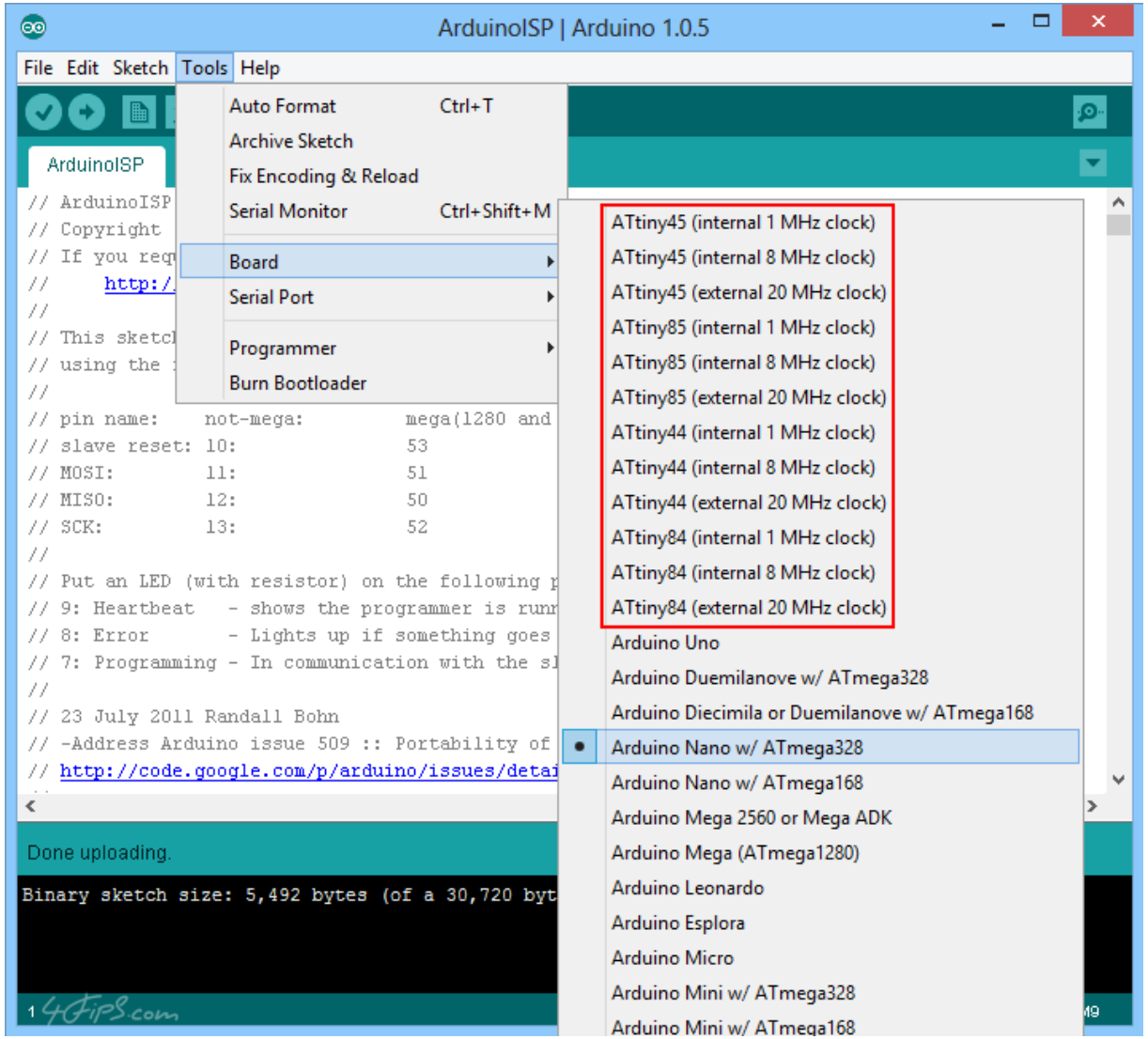

http://fablearn.stanford.edu/fellows/comment/reply/394

# Programarea micro – controllerului ATTiny prin intermediul platformei Arduino și a interfeței SPI




http://highlowtech.org

# Programarea micro – controllerului ATTiny utilizând Arduino IDE




http://forums.4fips.com/viewtopic.php?f=3&t=1080

# 4. Implementarea aplicațiilor

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

1. Recepționarea datelor prin interfața serial de la microcontroler
2. Transmiterea datelor prin intermediul interfeței serial la microcontroler
3. Recepționarea datelor prin intermediul modulului Bluetooth HC-05
4. Transmiterea datelor prin intermediul modulului Bluetooth HC-05
5. Preluarea datelor de la traductorul de umiditate și temperatură DHT-11
6. Implementarea unei aplicații de monitorizare multipunct a parametrilor ambientali

# 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;
- ✓ traductor pentru umiditate și temperatură DHT-11
- ✓ rezistențe cu valoarea de 100 [Ω];
- ✓ rezistențe cu valoarea de 10 [kΩ];
- ✓ senzor de temperatura LM-35;
- ✓ 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

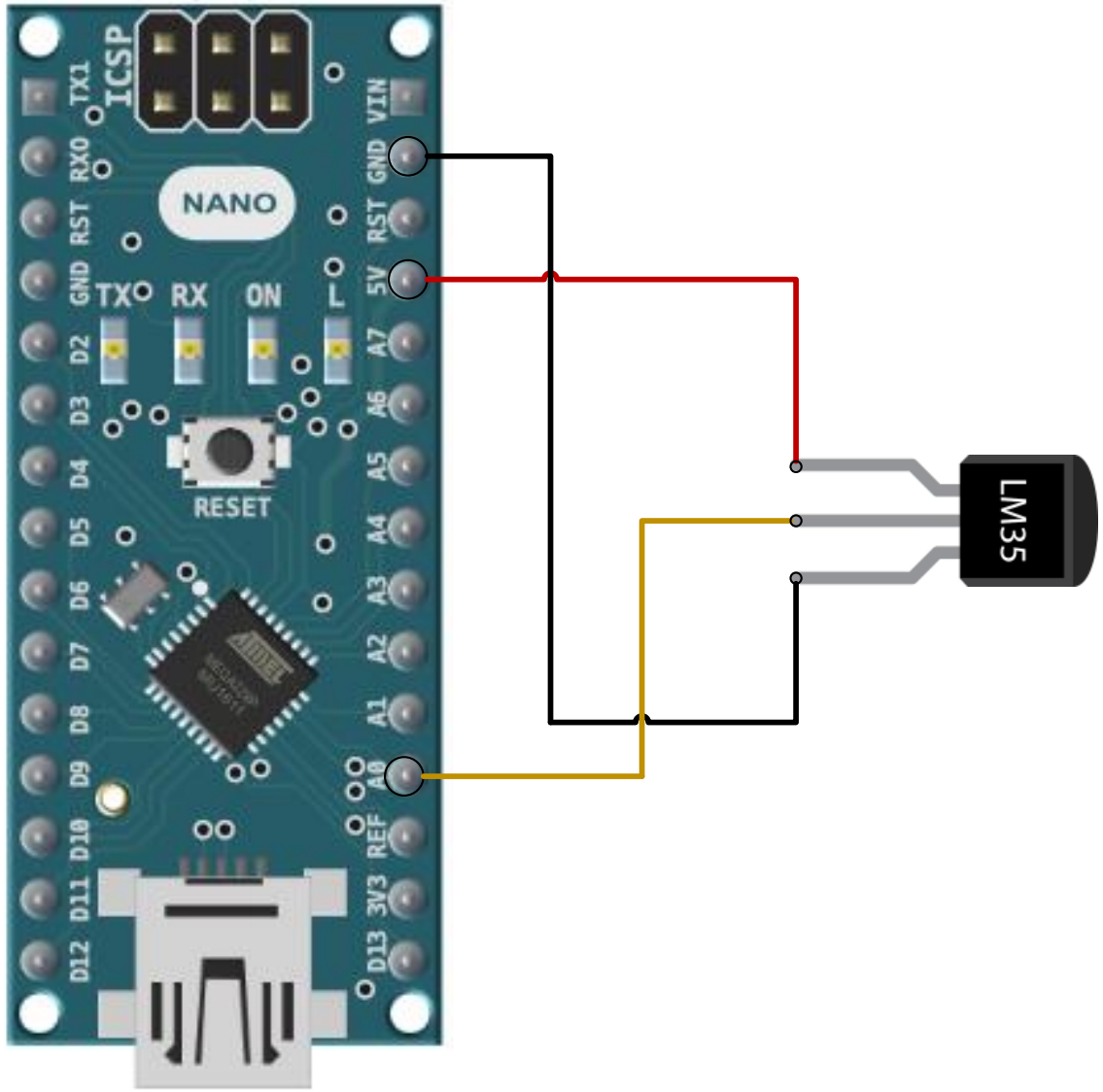
➢ **Implementarea aplicației nr. 1 presupune:**

✓ declararea 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 serial;

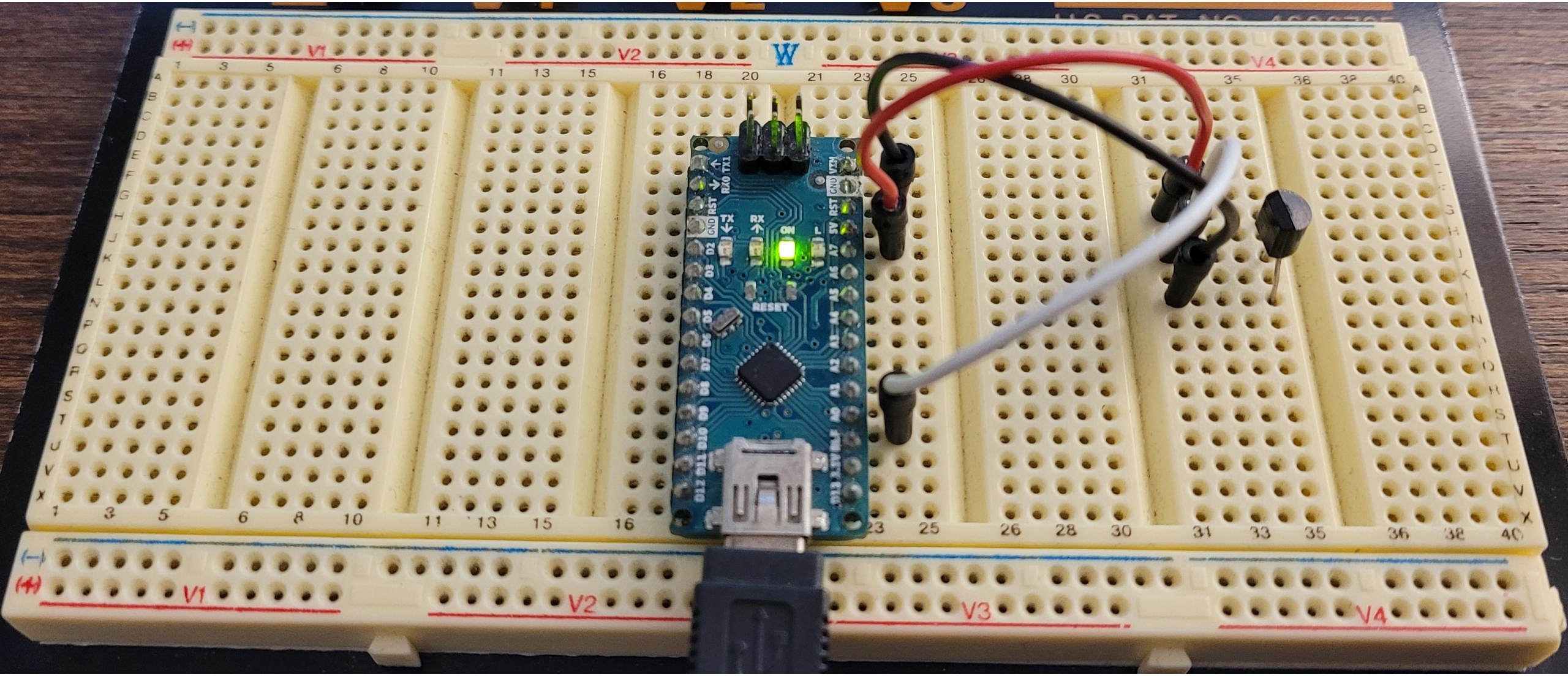
# 4. Implementarea aplicațiilor – Aplicația nr. 1

```
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.print("\n");
  delay(250);
}
```

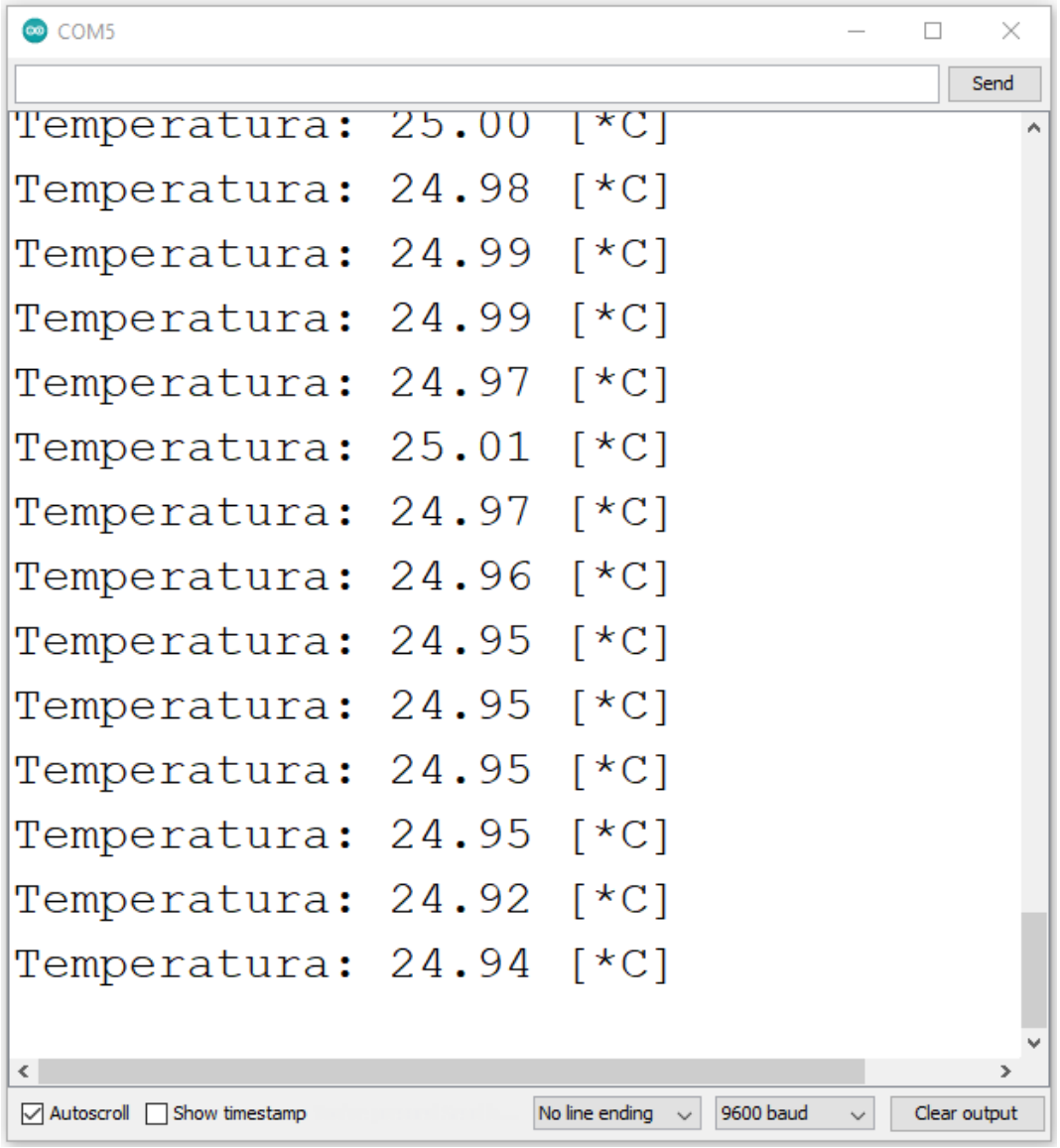
# 4. Implementarea aplicațiilor – Aplicația nr. 1



# 4. Implementarea aplicațiilor – Aplicația nr. 1



# 4. Implementarea aplicațiilor – Aplicația nr. 1




Consola serial – Afișarea valorilor medii ale temperaturii

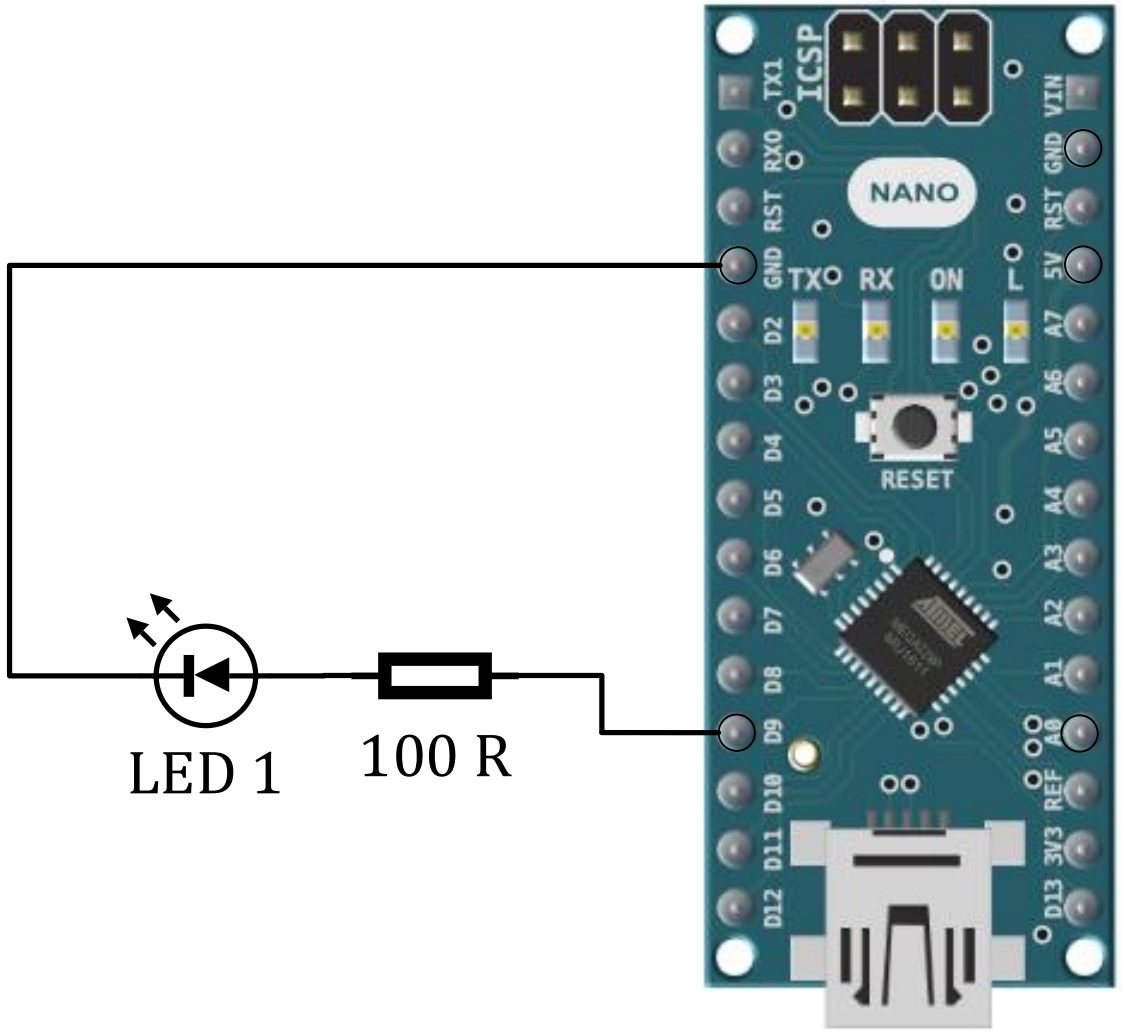
# 4. Implementarea aplicațiilor – Aplicația nr. 2

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

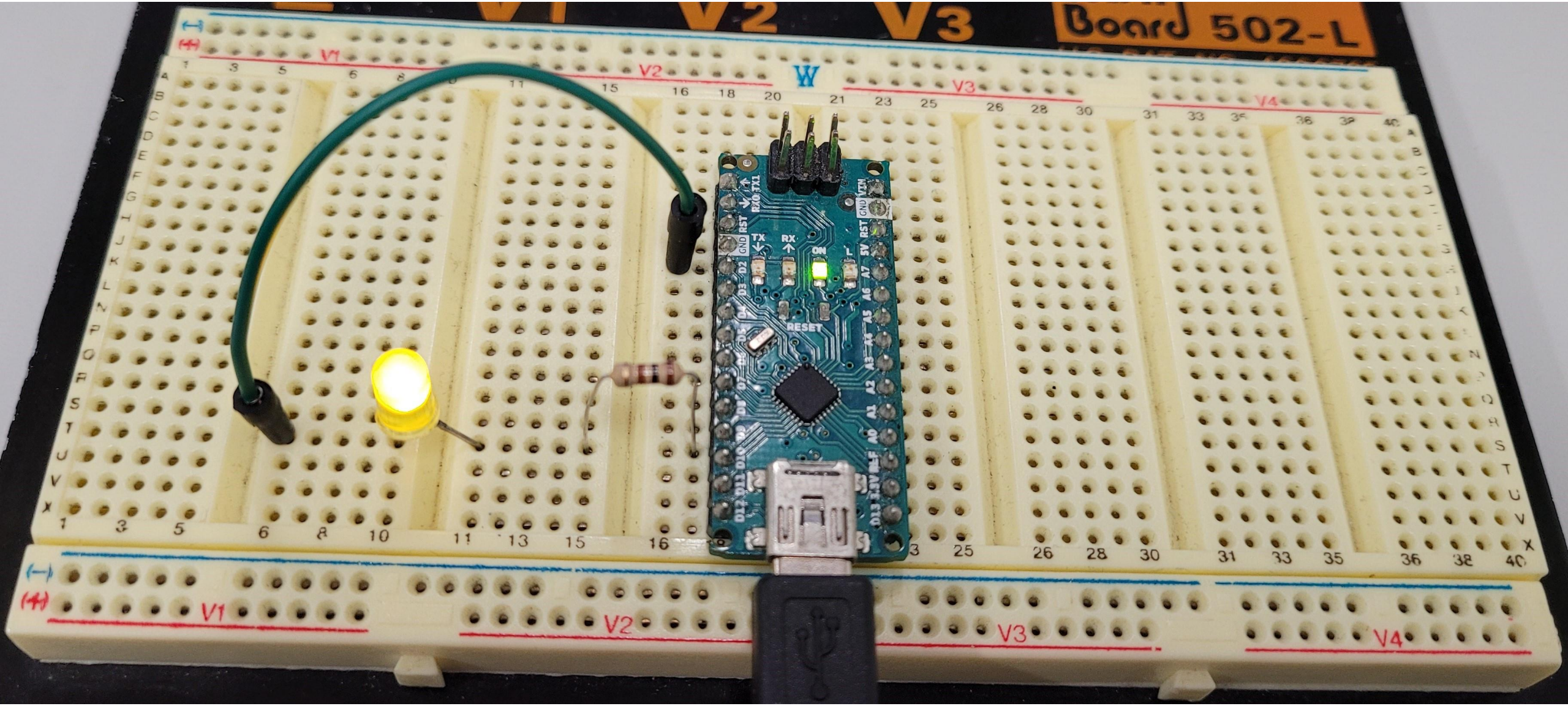
✓ declararea unei constante de tip număr întregi „pin_led” cu valoarea „9”;
✓ inițializarea unei variabile de tip număr întreg „dc” cu valoarea „0”;
✓ inițializarea terminalului digital „9” în modul de lucru „ieșire digitală”;
✓ inițializarea comunicației Serial la viteza de transfer 9600 [b/s]
✓ afișarea mesajului: "Introduceti factorul de umplere 0 - 255: ";
✓ verificarea disponibilității datelor de pe magistrala serial cu funcția „Serial.available()”;
✓ preluarea factorului de umplere de la magistrala serial cu funcția „Serial.parseInt()”;
✓ afișarea valorii exprimată pe 8 biți a duratei de conducție în consola serial (Fig. 16);
✓ ajustarea factorului de umplere pentru un tren de impulsuri furnizat pe terminalul „9”;

# 4. Implementarea aplicațiilor – Aplicația nr. 2

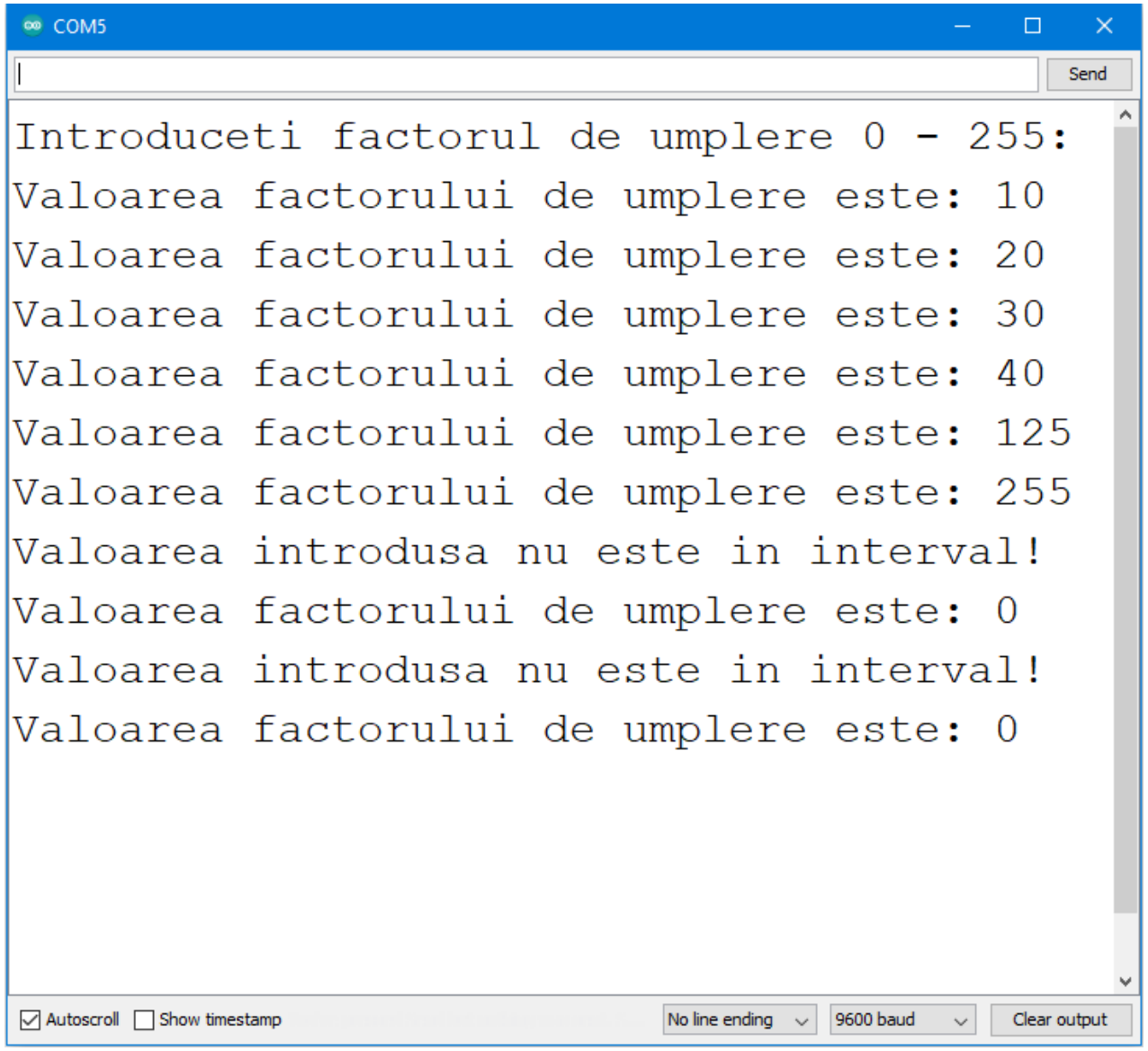
```
const int pin_led = 9;
int dc = 0;
void setup() {
  pinMode(pin_led, OUTPUT);
  Serial.begin(9600);
  Serial.println("Introduceti factorul de umplere 0 - 255: ");
}
void loop() {
  if(Serial.available()){
    dc = Serial.parseInt();
    if(dc < 0){
      Serial.print("Valoarea introdusa nu este in interval!");
      Serial.print("\n");
      dc = 0;
    }
    if(dc > 255){
      Serial.print("Valoarea introdusa nu este in interval!");
      Serial.print("\n");
      dc = 0;
    }
    Serial.print("Valoarea factorului de umplere este: ");
    Serial.print(dc);
    Serial.print("\n");
  }
  analogWrite(pin_led, dc);
}
```



# 4. Implementarea aplicațiilor – Aplicația nr. 2



# 4. Implementarea aplicațiilor – Aplicația nr. 2




Consola Serial – Introducerea factorului de umplere sau a duratei de conducție în consolă

# 4. Implementarea aplicațiilor – Aplicația nr. 3

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

✓ determinarea temperaturii medii achiziționată de la traductorul LM-35;

✓ inițializarea comunicației serial atât locală cât și la distanță;

✓ afișarea locală a mesajului pentru monitorizare în consola serial;

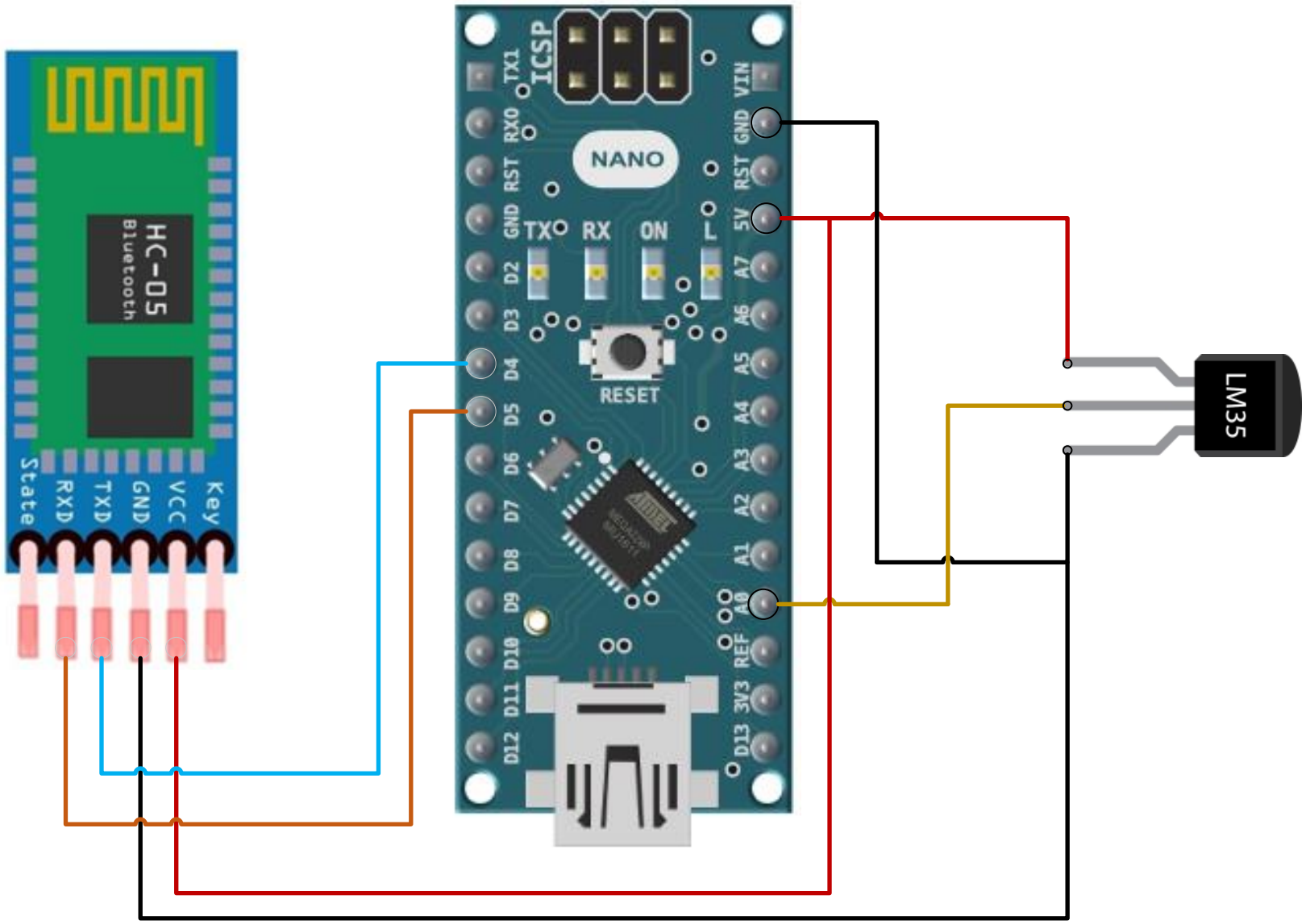
✓ afișarea în aplicația Bluetooth Serial Terminal a conținutului consolei serial.

# 4. Implementarea aplicațiilor – Aplicația nr. 3

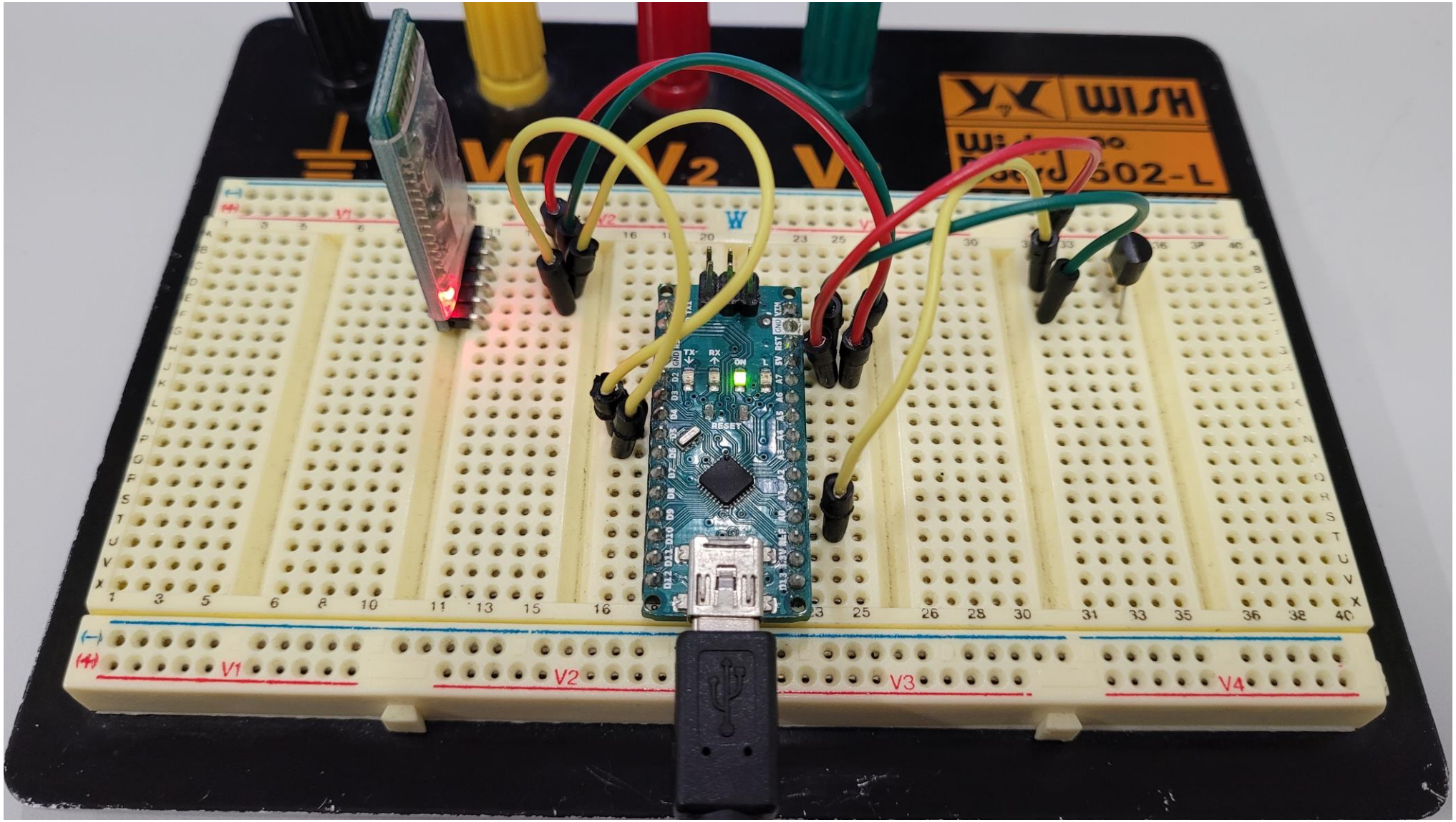
```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(4, 5); //4 - RX, 5 - TX

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);
  BTSerial.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.print("\n");

  BTSerial.print("Temperatura: ");
  BTSerial.print(media);
  BTSerial.print(" [*C]");
  BTSerial.print("\n");
  delay(250);
```

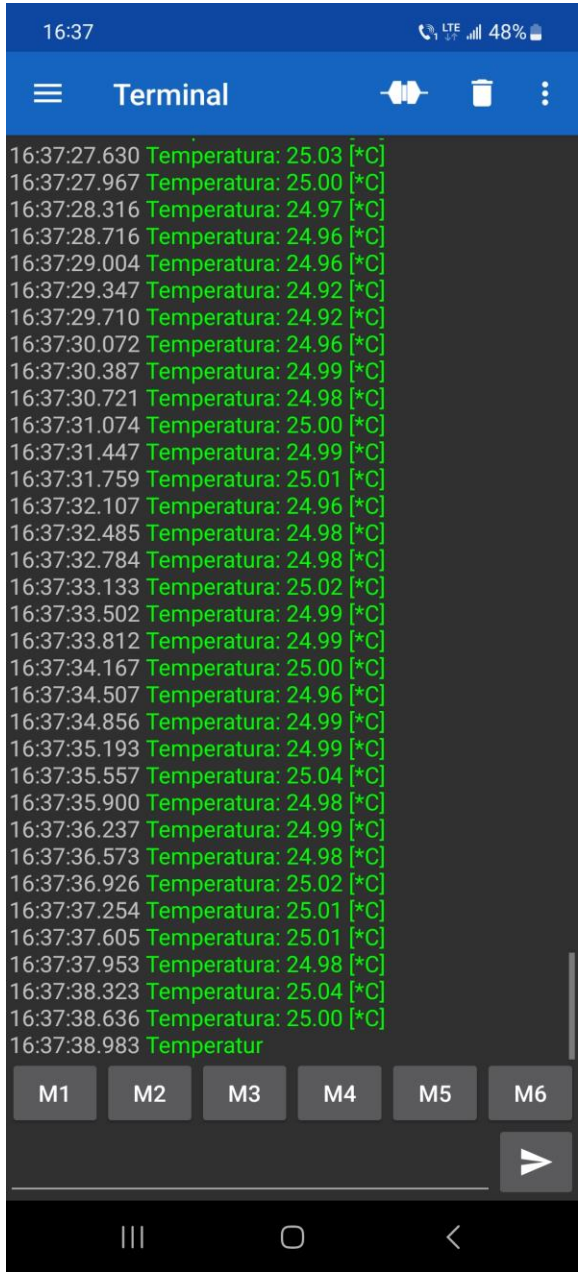
# 4. Implementarea aplicațiilor – Aplicația nr. 3



# 4. Implementarea aplicațiilor – Aplicația nr. 3



# 4. Implementarea aplicațiilor – Aplicația nr. 3



# 4. Implementarea aplicațiilor – Aplicația nr. 4

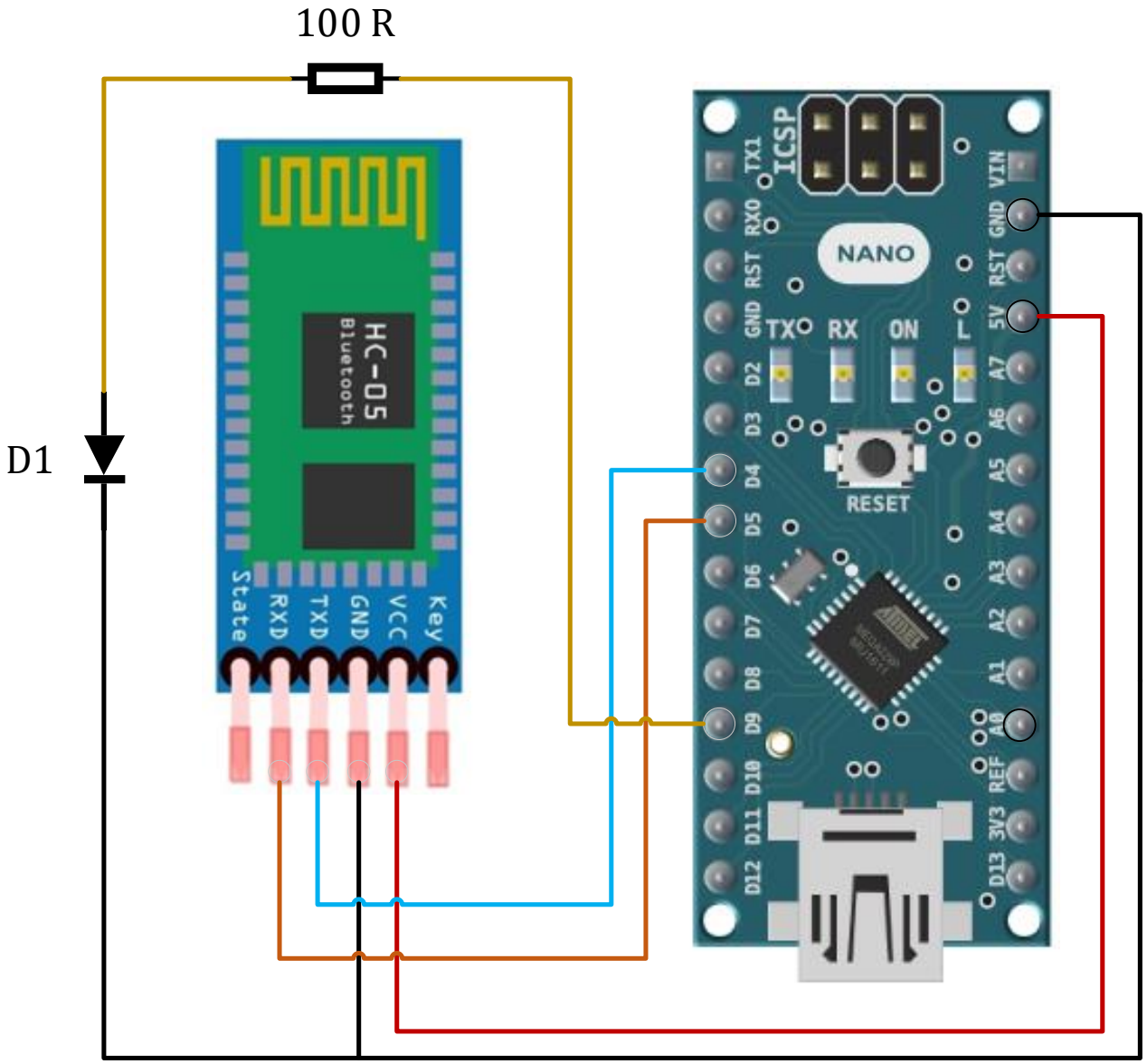
## ➢ Implementarea aplicației nr. 4 presupune:

- ✓ inițializarea comunicației Serial la viteza de transfer 9600 [b/s]
- ✓ inițializarea comunicației serial atât locală cât și la distanță;
- ✓ verificarea disponibilității datelor de pe magistrala serial cu funcția „Serial.available()”;
- ✓ verificarea disponibilității datelor de pe magistrala cu funcția „BTSerial.available()”;
- ✓ preluarea factorului de umplere de la magistrala serial cu funcția „Serial.parseInt()”;
- ✓ preluarea factorului de umplere de la magistrala serial cu funcția „BTSerial.parseInt()”;
- ✓ afișarea valorii exprimată pe 8 biți a duratei de conducție în consola serial;
- ✓ afișarea valorii pe 8 biți a duratei de conducție în aplicația Bluetooth Serial Terminal;
- ✓ ajustarea factorului de umplere pentru un tren de impulsuri furnizat pe terminalul „9”;
- ✓ preluarea atât local cât și la distanță a datelor numerice din consolă;
- ✓ afișarea locală a mesajului pentru monitorizare în consola serial;
- ✓ afișarea în aplicația Bluetooth Serial Terminal a conținutului consolei serial.

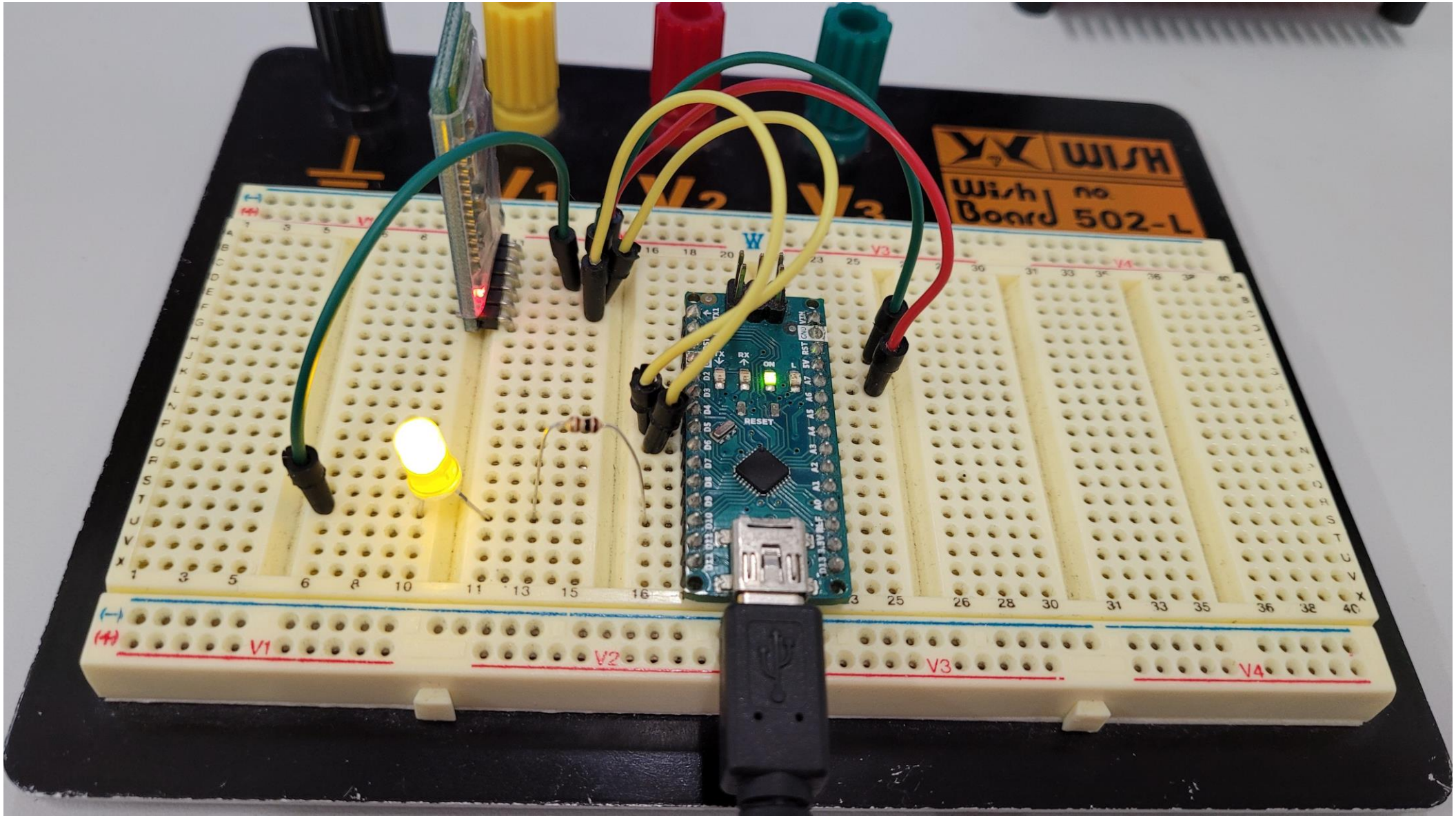
# 4. Implementarea aplicațiilor – Aplicația nr. 4

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(4, 5); //4 - RX, 5 - TX

const int pin_led = 9;
int dc = 0;

void setup() {
  Serial.begin(9600);
  BTSerial.begin(9600);
  Serial.println("Introduceti factorul de umplere: ");
  BTSerial.println("Introduceti factorul de umplere: ");
}

void loop() {
  if(Serial.available()){
    dc = Serial.parseInt();
    if(dc < 0){
      Serial.print("Valoarea introdusa nu este in interval!");
      Serial.print("\n");
      dc = 0;
    }
    if(dc > 255){
      Serial.print("Valoarea introdusa nu este in interval!");
      Serial.print("\n");
      dc = 0;
    }
    Serial.print("Valoare factor de umplere: ");
    Serial.print(dc);
    Serial.print("\n");
  }

  if(BTSerial.available()){
    dc = BTSerial.parseInt();
    if(dc < 0){
      BTSerial.print("Valoarea introdusa nu este in interval!");
      BTSerial.print("\n");
      dc = 0;
    }

    if(dc > 255){
      BTSerial.print("Valoarea introdusa nu este in interval!");
      BTSerial.print("\n");
      dc = 0;
    }

    BTSerial.print("Valoare factor de umplere: ");
    BTSerial.print(dc);
    BTSerial.print("\n");
  }
  analogWrite(pin_led, dc);
}
```

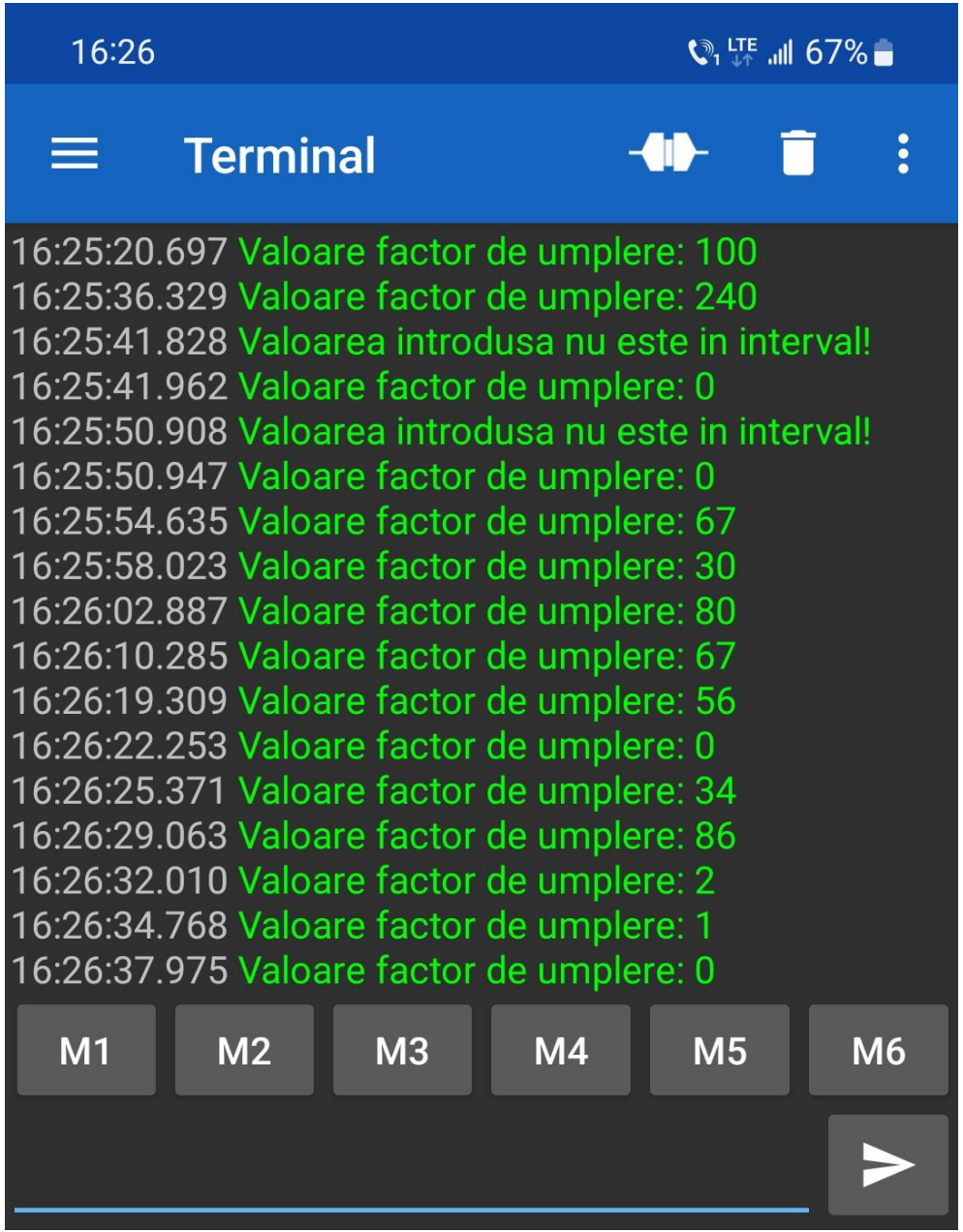
# 4. Implementarea aplicațiilor – Aplicația nr. 4



# 4. Implementarea aplicațiilor – Aplicația nr. 4



# 4. Implementarea aplicațiilor – Aplicația nr. 4



# 4. Implementarea aplicațiilor – Aplicația nr. 5

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

✓ inițializarea bibliotecii „dht.h” specifică traductorului de umiditate și temperatură;

✓ inițializarea obiectului „DHT” pentru interogarea punctuală a parametrilor descriptivi;

✓ interogarea traductorului cu ajutorul bibliotecii de funcții;

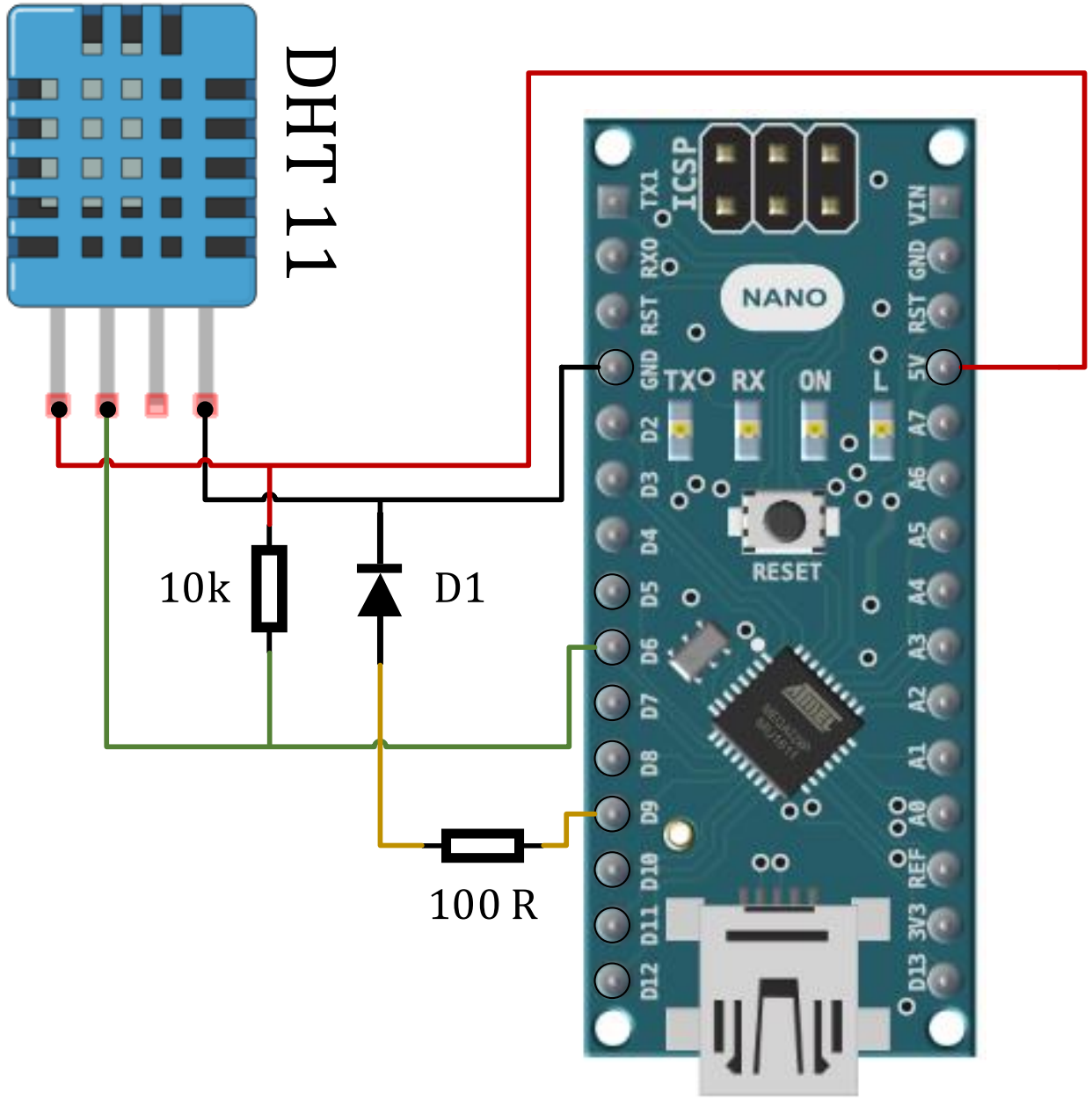
✓ preluarea de pe magistrala „1 – Wire” a informațiilor digitale de la traductor;

✓ afișarea în consola serial a valorilor de temperatură și umiditate;

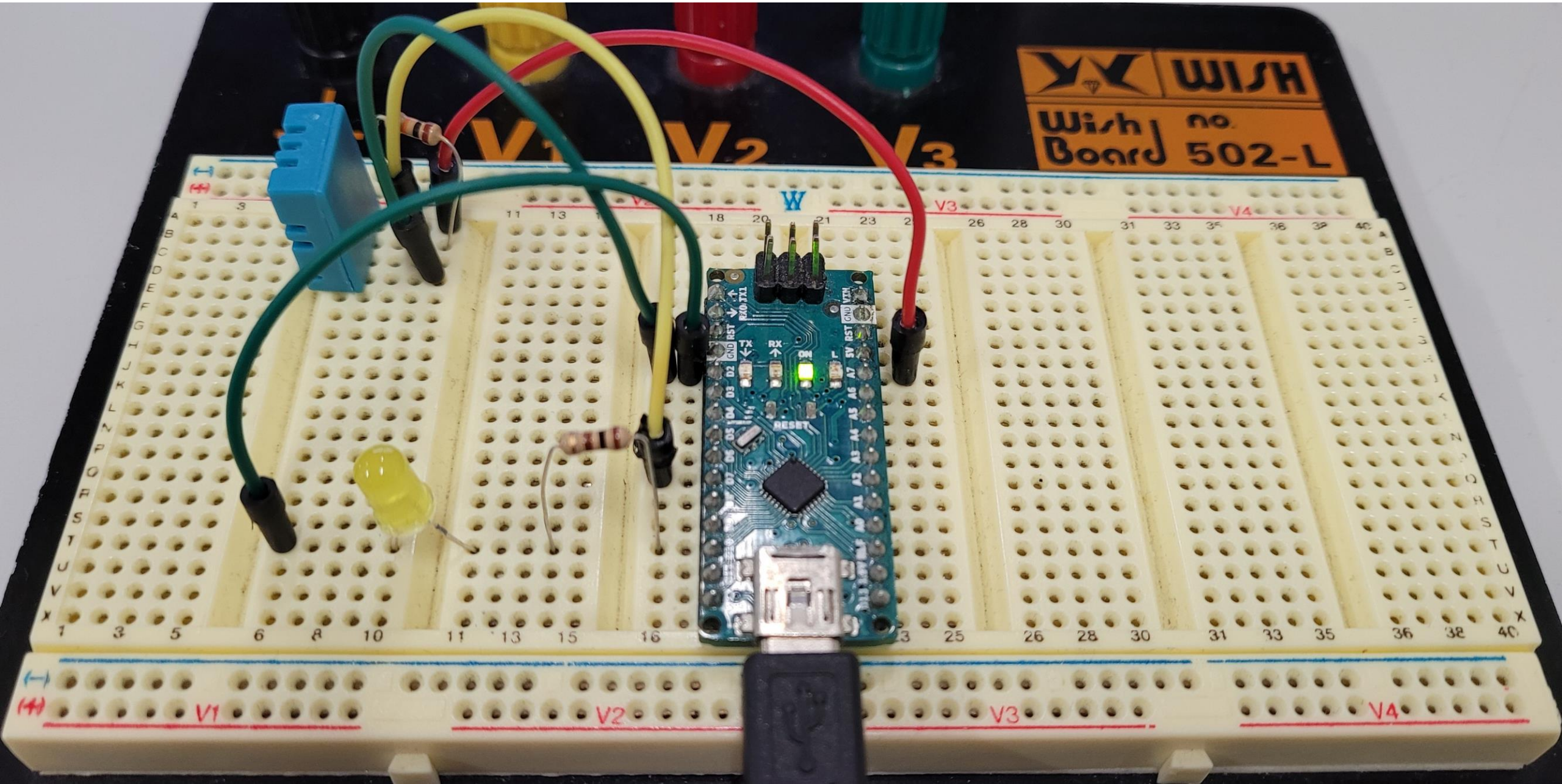
✓ semnalizarea prin intermediul unui LED de stare la fiecare achiziție;

# 4. Implementarea aplicațiilor – Aplicația nr. 5

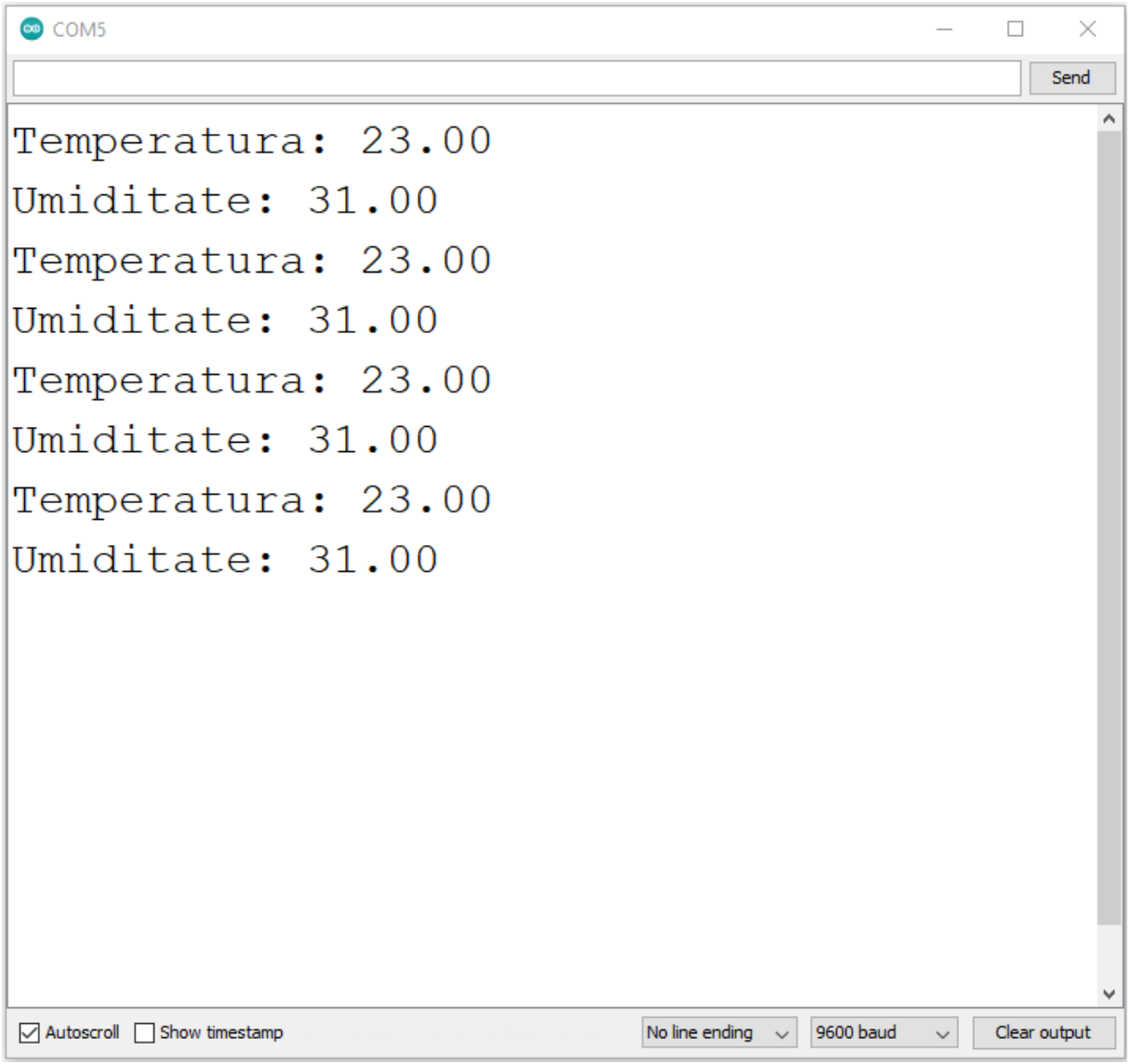
```
#include <dht.h>
#define DHT11_PIN 6

const int status_led = 9;
dht DHT;

void setup() {
  pinMode(status_led, OUTPUT);
  Serial.begin(9600);
  digitalWrite(status_led, LOW);
}

void loop() {
  int chk = DHT.read11(DHT11_PIN);
  digitalWrite(status_led, LOW);
  delay(5000);
  Serial.print("Temperatura: ");
  Serial.println(DHT.temperature);
  Serial.print("Umiditate: ");
  Serial.println(DHT.humidity);
  digitalWrite(status_led, HIGH);
  delay(100);
}
```



# 4. Implementarea aplicațiilor – Aplicația nr. 5



# 4. Implementarea aplicațiilor – Aplicația nr. 5




Consola Serial – Afișarea parametrilor ambientali achiziționați de la traductorul DHT-11

# 4. Implementarea aplicațiilor – Aplicația nr. 6

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

✓ interogarea periodică a traductorului digital de umiditate și temperatură DHT – 11;

✓ afișarea locală (în consola serial a calculatorului gazdă) a valorilor preluate;

✓ afișarea la distanță a valorilor înregistrate de la traductor (Bluetooth Serial Terminal);

✓ expunerea valorilor preluate de la traductor pe afișajul LCD;

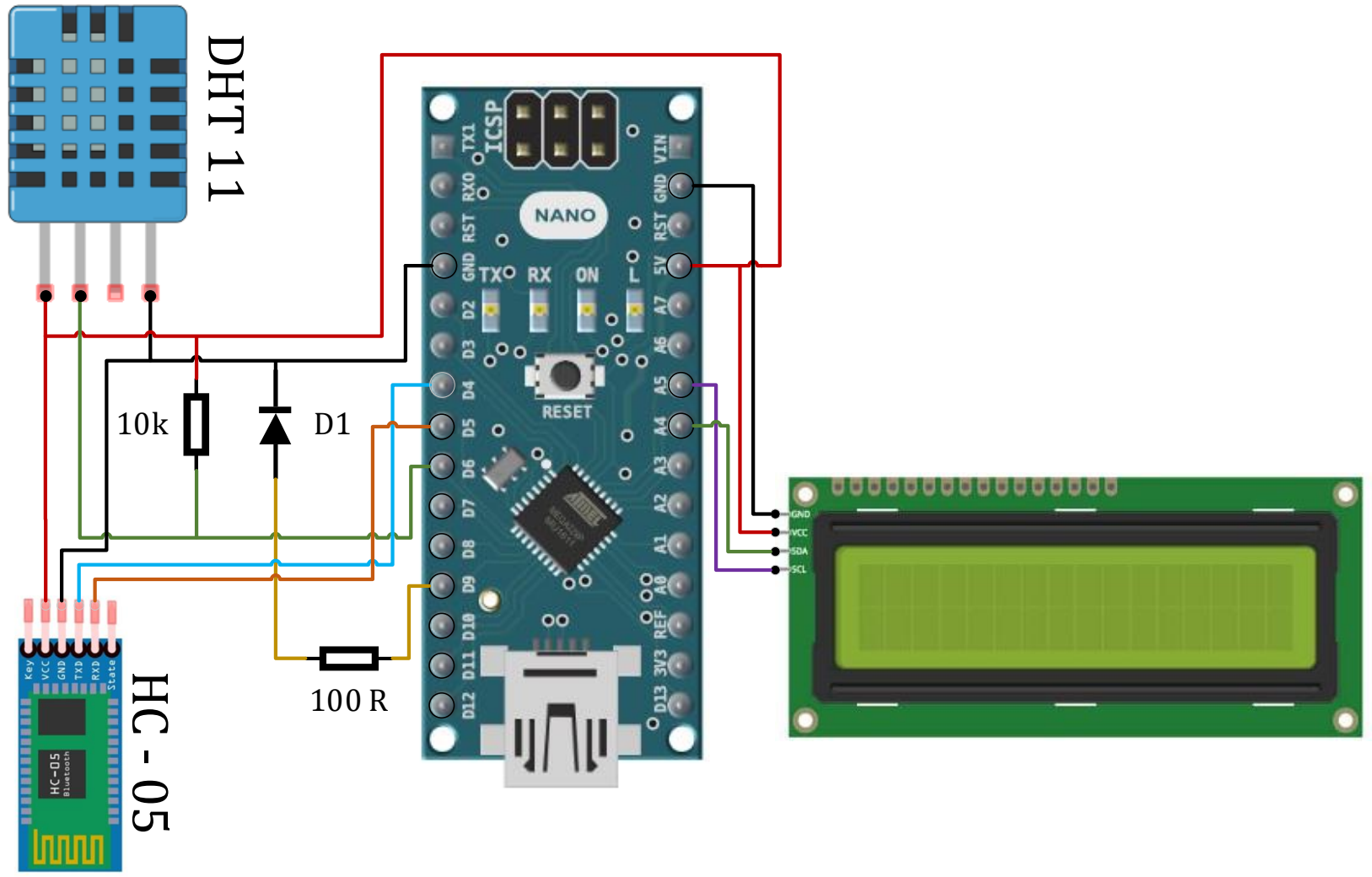
# 4. Implementarea aplicațiilor – Aplicația nr. 6 – P1

```
#include <Wire.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>
#include <dht.h>
#define DHT11_PIN 6
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
SoftwareSerial BTSerial(4, 5); //4 - RX, 5 – TX
const int status_led = 9;
dht DHT;

void setup() {
  pinMode(status_led, OUTPUT);
  Serial.begin(9600);
  BTSerial.begin(9600);
  digitalWrite(status_led, LOW);
  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("Temp. [*C] ");
  lcd.setCursor(0,1);
  lcd.print("Umid. [%] ");
}
```

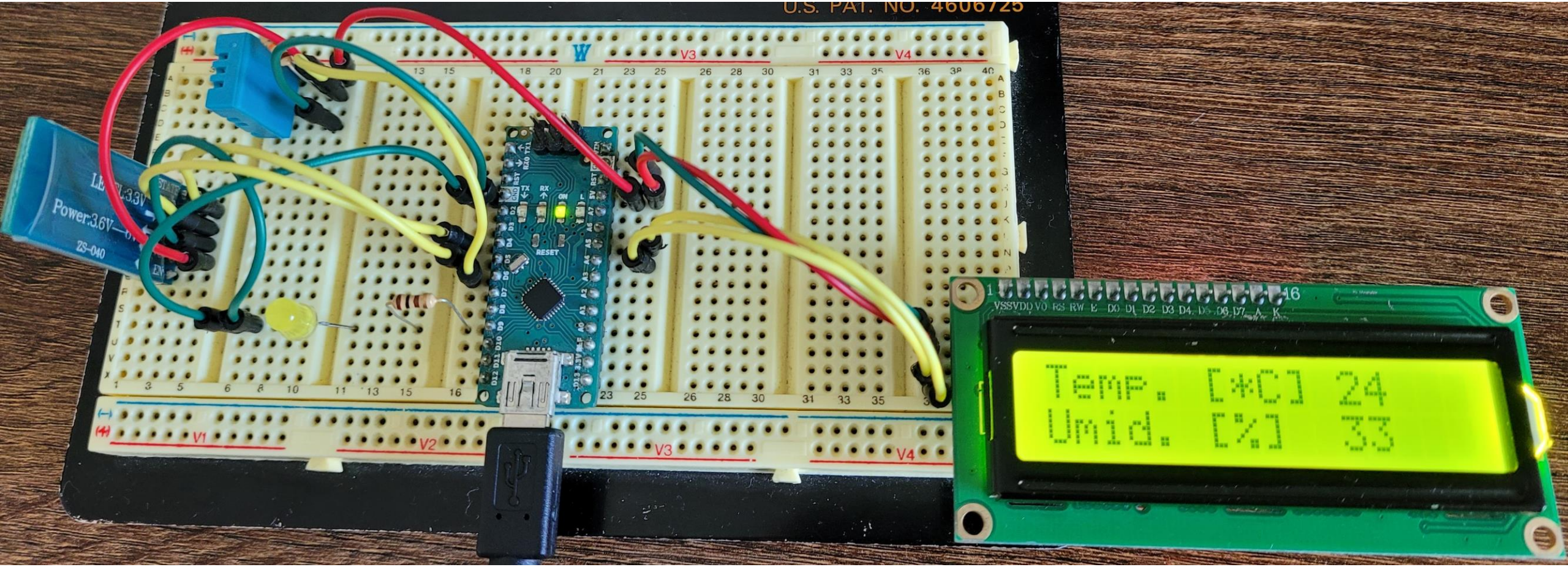
```
void loop() {
  int chk = DHT.read11(DHT11_PIN);
  digitalWrite(status_led, LOW);
  delay(2000);
  Serial.print("Temperatura [*C]: ");
  Serial.println(DHT.temperature, 0);
  Serial.print("Umiditate [%]: ");
  Serial.println(DHT.humidity, 0);
  BTSerial.print("Temperatura [*C]: ");
  BTSerial.println(DHT.temperature, 0);
  BTSerial.print("Umiditate [%]: ");
  BTSerial.println(DHT.humidity, 0);
  digitalWrite(status_led, HIGH);

  lcd.setCursor(11,0);
  if(DHT.temperature < 1000) {
    lcd.setCursor(14,0);
    lcd.print(" ");
    lcd.setCursor(11,0);
    lcd.print(DHT.temperature, 0);
  }
  if(DHT.temperature < 100) {
    lcd.setCursor(13,0);
    lcd.print(" ");
    lcd.setCursor(11,0);
    lcd.print(DHT.temperature, 0);
  }
  if(DHT.temperature < 10) {
    lcd.setCursor(12,0);
    lcd.print(" ");
    lcd.setCursor(11,0);
    lcd.print(DHT.temperature, 0);
  }
```

# 4. Implementarea aplicațiilor – Aplicația nr. 6 – P2

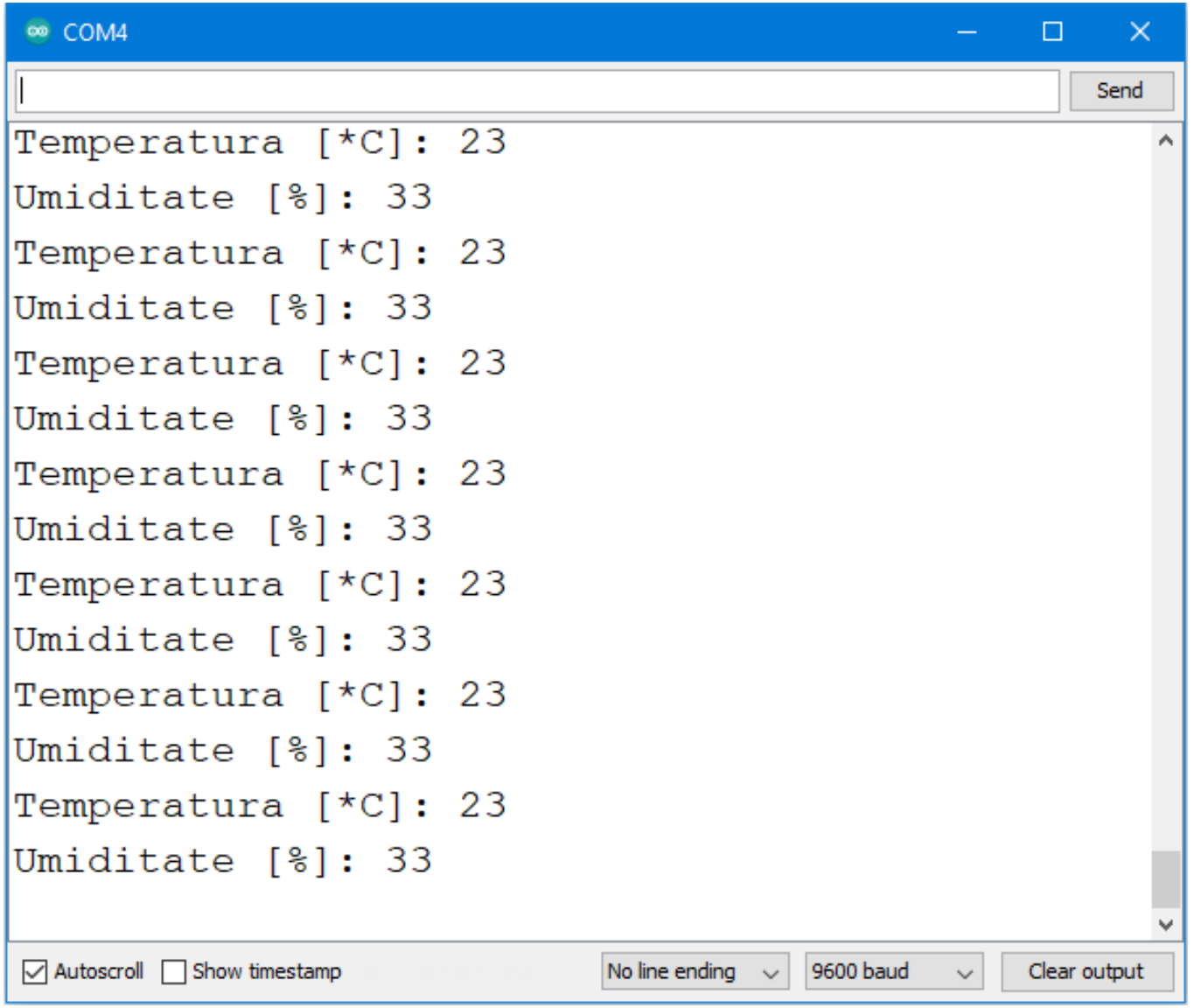
```
 lcd.setCursor(11,1);

  if(DHT.humidity < 1000) {
    lcd.setCursor(14,1);
    lcd.print(" ");
    lcd.setCursor(11,1);
    lcd.print(DHT.humidity, 0);
  }
  if(DHT.humidity < 100) {
    lcd.setCursor(13,1);
    lcd.print(" ");
    lcd.setCursor(11,1);
    lcd.print(DHT.humidity, 0);
  }
  if(DHT.humidity < 10) {
    lcd.setCursor(12,1);
    lcd.print(" ");
    lcd.setCursor(11,1);
    lcd.print(DHT.humidity, 0);
  }
  delay(100);
}
```



# 4. Implementarea aplicațiilor – Aplicația nr. 6

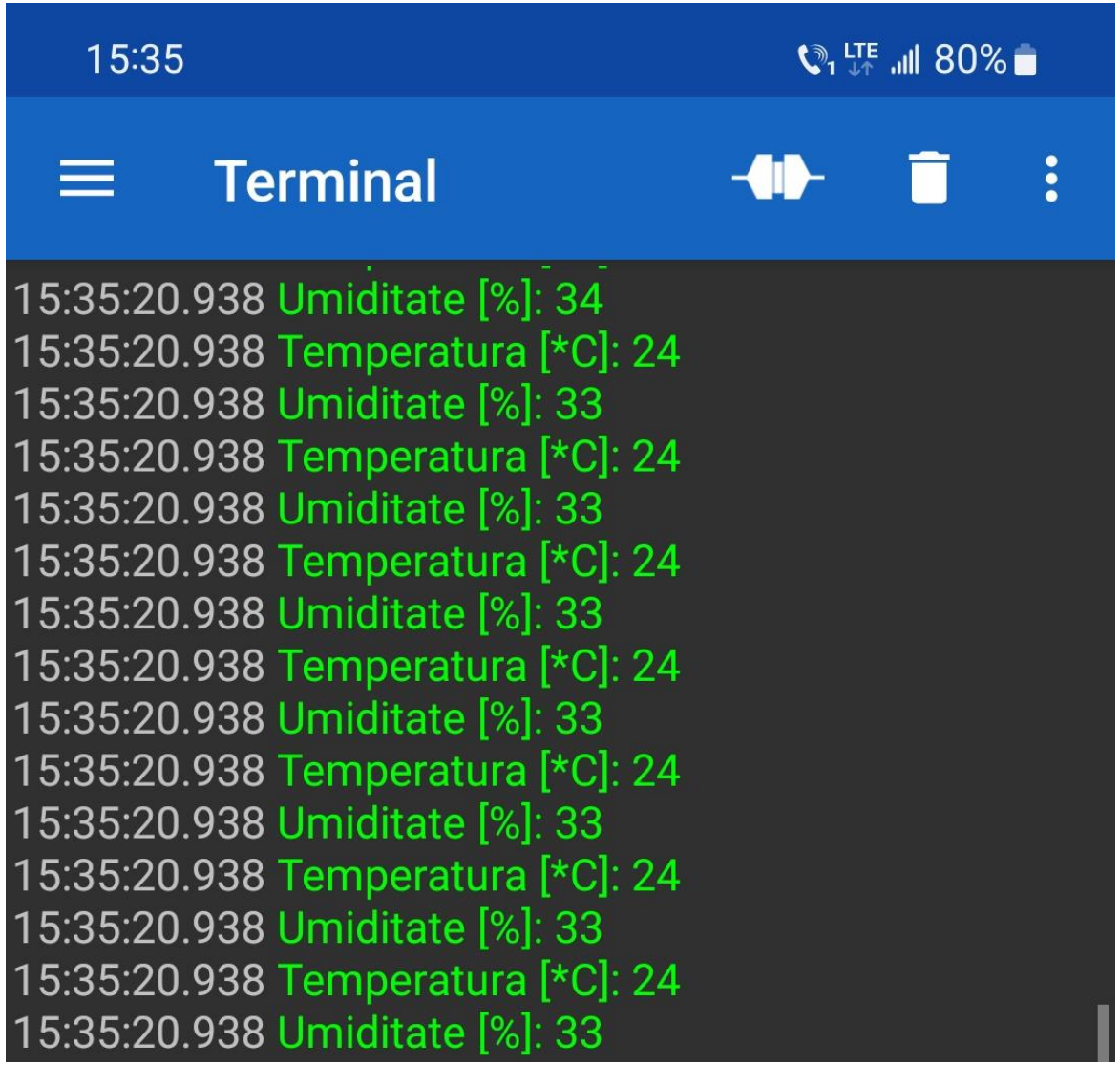


# 4. Implementarea aplicațiilor – Aplicația nr. 6




Consola Serial – Afișarea parametrilor ambientali achiziționați de la traductorul DHT-11

# 4. Implementarea aplicațiilor – Aplicația nr. 6




Bluetooth Serial Terminal – Afișarea parametrilor ambientali achiziționați de la traductorul DHT-11

# 5. Concluzii

❖ Prin intermediul protocoalelor de comunicație implementate atât la nivelul codului program (eng. software protocol) cât și la nivelul fizic (eng. hardware protocol) al unui sistem de calcul, pot fi realizate următoarele funcții:

✓ diagnoză și depanare (eng. debugging and repairing);

✓ monitorizarea la distanță a unor parametrii (eng. remote monitoring);

✓ accesul multipunct la mijloacele fizice ale aplicației (eng. multi point operation);

✓ traversarea sau conversia de la un protocol la altul (ex. adaptorul Bluetooth – Serial);

✓ transportarea datelor pe distanțe mari în format digital codificat;

# 6. Bibliografie

**1.** Technical University of Wien – „Communication Protocols for Embedded Systems” - https://ti.tuwien.ac.at/

**2.** Universitatea Politehnica Timișoara – „Interfațarea circuitelor - Tipuri de interfețe, Protocoale seriale de comunicare” - https://www.aut.upt.ro

**3.** Universitatea Tehnică din Cluj – Napoca: 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.** Washington University – „MEMS: Micro-Electro-Mechanical Systems – What are MEMS?” - https://courses.cs.washington.edu/

**5.** ITead Studio – „HC-05 -Bluetooth to Serial Module” - https://components101.com/

**6.** Mouser Electronics and OSEPP Electronics – „DHT11 Humidity & Temperature Sensor” https://www.mouser.com/

**7.** Electronics-lab.com – „BUILD YOUR OWN I2C SENSOR” – ATTiny 85 MCU https://www.electronics-lab.com/build-i2c-sensor/