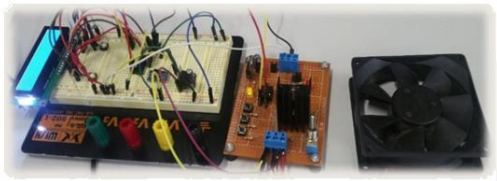
UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
FACULTATEA DE INGINERIE ELECTRICĂ

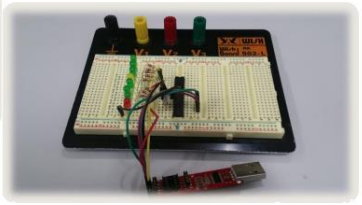
SISTEME CU MICROPROCESOARE

Ședința de laborator nr. 2

Manipularea intrărilor și ieșirilor digitale

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 platformei de dezvoltare Arduino Nano [1]
- ✓ Prezentarea mediului de dezvoltare Arduino IDE [2]
- ✓ Prezentarea noțiunilor specifice limbajului de programare „Wiring” [3]
- ✓ Implementare aplicațiilor specifice procesării semnalelor digitale [4] [5]

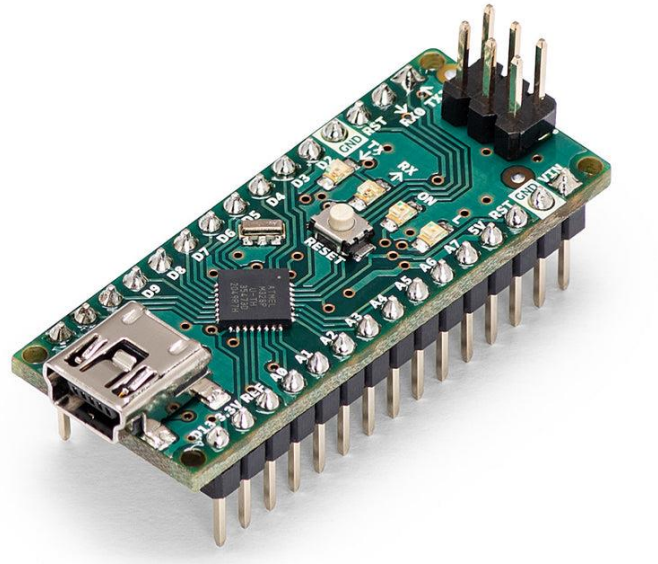
2. Introducere

- Platforma de dezvoltare **Arduino Nano** [1] reprezintă o formă de integrare cu ajutorul tehnologiei ON-Board a microcontrolerului **ATMega 328P** [6] la nivelul unei construcții de tip mono-placă (eng. single - board)

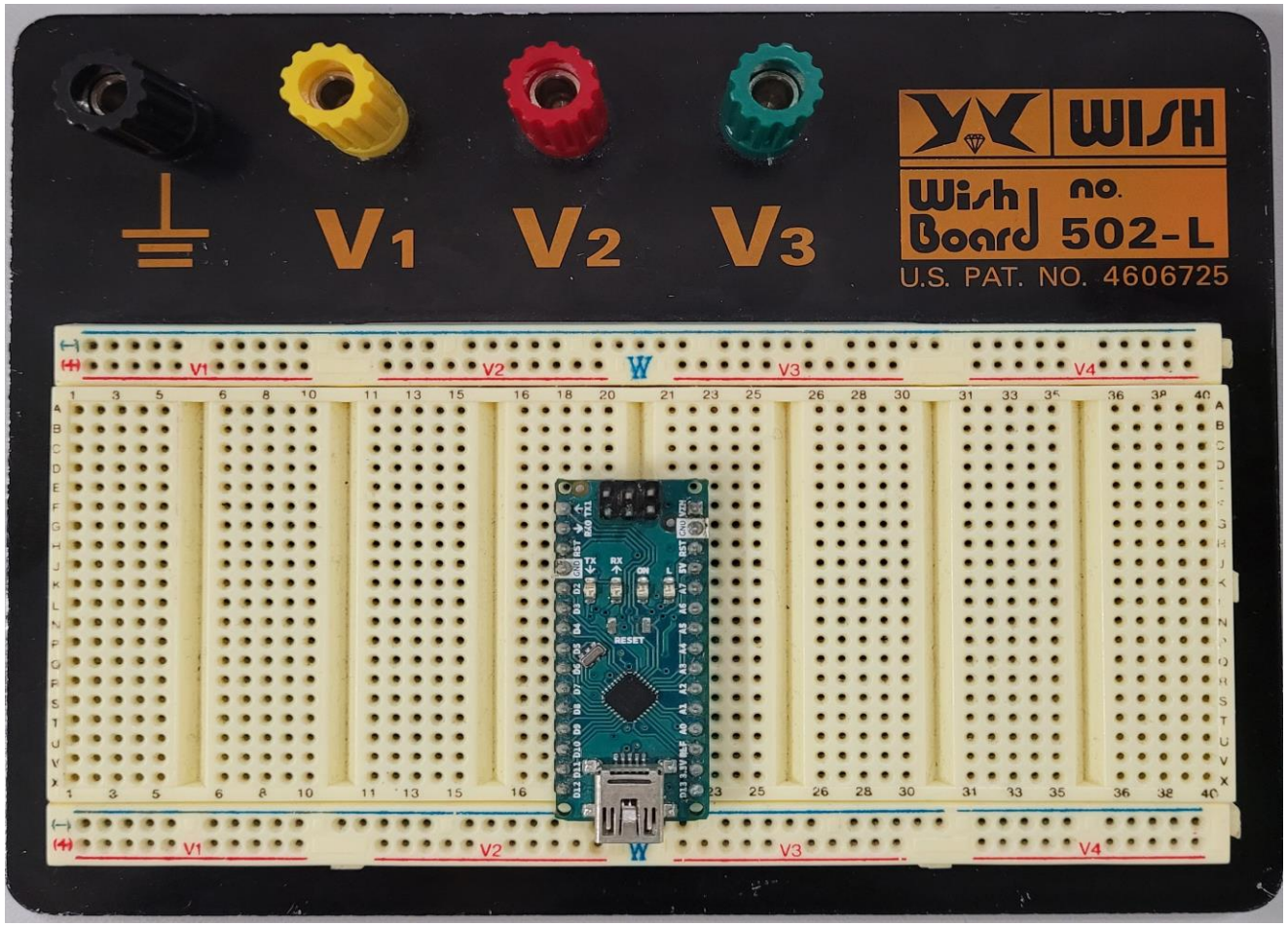
- **Dimensiunea redusă** și **forma proeminentă a terminalelor de acces**, conferă platformei de dezvoltare Arduino Nano posibilitatea instalării acesteia la nivelul unei **plăcuțe pentru testare fără lipituri** (eng. solderless breadboard)

- **Harta de distribuție** a funcțiilor specifice terminalelor la nivelul platformei Arduino Nano este inscripționată la nivelul cablajului imprimat. Suplimentar față de funcțiile standard inscripționate pe cablaj, manualul de instrucțiuni, prezintă un set alternativ de funcții pentru terminalele platformei de dezvoltare

2. Introducere



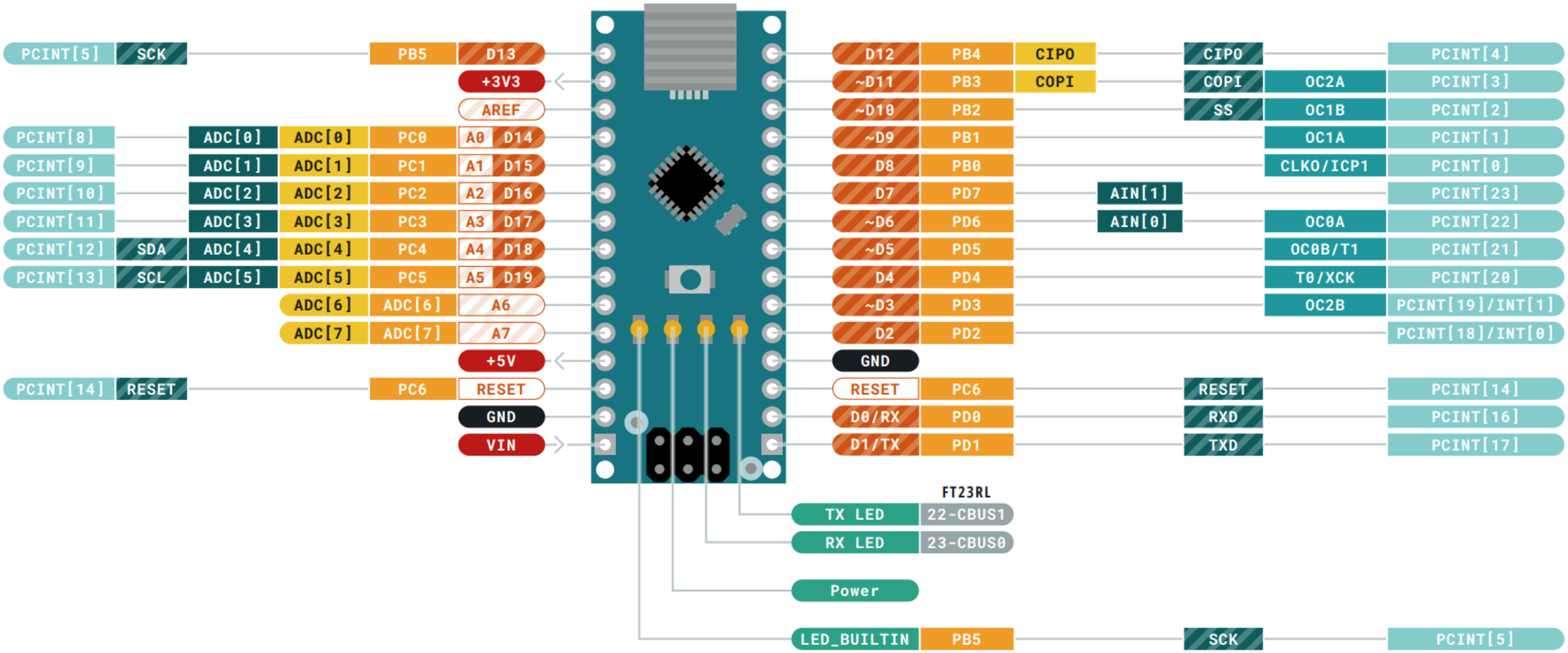
Platforma de dezvoltare
Arduino Nano



Instalarea platformei de dezvoltare Arduino Nano
la nivelul unei plăcuțe pentru testare

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

2. Introducere



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

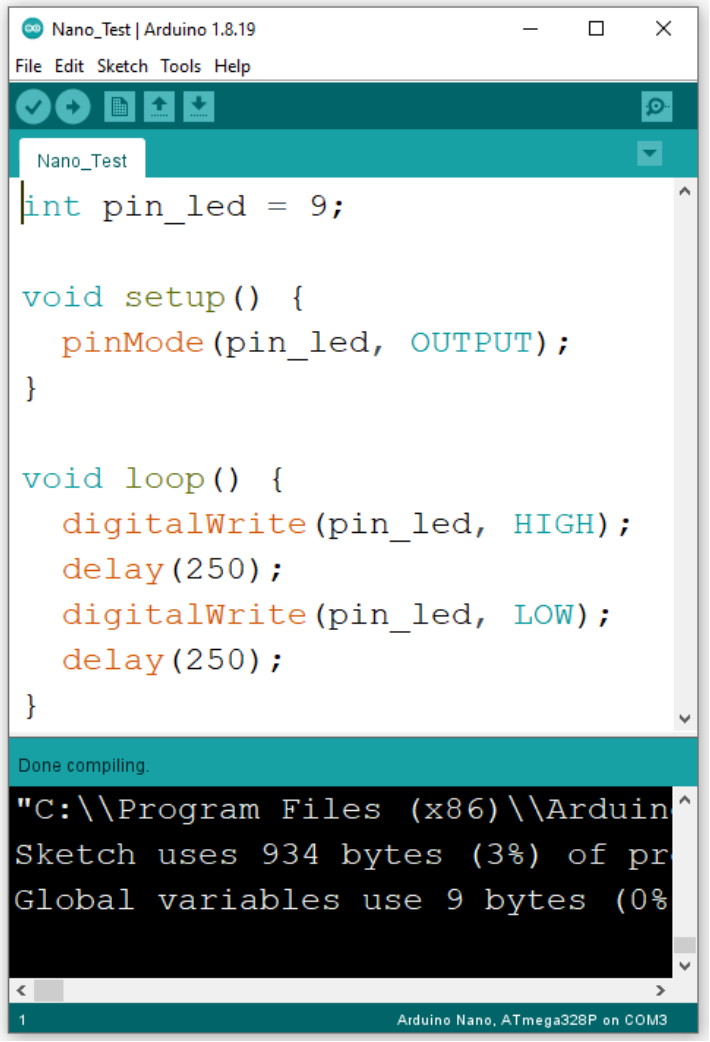
2. Introducere

- **Mediul de dezvoltare Arduino IDE** [2] (eng. Integrated Development Environment), reprezintă pachetul de programe și instrumente necesare în vederea parcurgerii procesului de programare a microcontrolerului ATMega 328 din componența platformelor de dezvoltare Arduino.

- În cadrul mediului Arduino IDE sunt incluse în mod implicit o serie de **biblioteci și fișiere sursă** care deservesc funcțiile de bază ale microcontrolerelor din familia AVR.

- Tot în cadrul mediului de dezvoltare Arduino IDE există și un **compilator**, care, pe baza fișierelor de definiție, **transpune instrucțiunile de program** scrise în limbaj C standard sau C++ în **instrucțiuni de procesor** în format numeric executabil de tip „.hex”.

2. Introducere



Mediul de dezvoltare
Arduino IDE

https://www.arduino.cc/en/software

3. Aspecte teoretice

- **Limbajul de programare** implementat în cadrul mediului Arduino IDE este similar limbajului „C” standard sau „C++” prezentând variațiuni, precum:
  - ✓ **structurile de program** precum „setup()” și „loop()”;
  - ✓ **declarațiile hardware** precum „pinMode()” și „digitalWrite”;

- Setul de instrucțiuni și declarații hardware suplimentare atașate limbajului de programarea „C / C++”, poartă denumirea **„Wiring”** sau în unele documentații de specialitate **„Wiring C”.**

- „Wiring”, reprezintă un mod de a **„intermedia”** procesul de programare între **operatorul uman care elaborează codul program** și **mașina de lucru finală** pe care urmează să fie implementat codul program. Acest concept (în cadrul literaturii de specialitate) poartă denumirea **API** (eng. Application Programming Interface).

3. Aspecte teoretice

Compilare = găsirea unor „comenzi echivalente” în limbajul de la nivel inferior;

ÎN SINTAXA WIRING

```
void setup() {
      pinMode(6, OUTPUT);
}
void loop() {
      digitalWrite(6, HIGH);
}
```

ÎN LIMBAJUL HIBRID ASAMBLARE - SINTAXĂ

```
void setup() {
      DDRD = B11111111;
}
void loop() {
      PORTD = B00100000;
}
```

Mediu grafic - nivel înalt (ex. Simulink)

Sintaxă – nivel mediu – inteligibil (ex. C++) ex. pinMode(), digitalWrite()

Limbaj de asamblare – tranziție – mnemonici DDRD = B00100000, PORTD = B00100000

Cod mașină (binar) – limbaj la nivel fizic 10010101 00000101 00000001 .....

4. Implementarea aplicațiilor

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

1. Semnalizare intermitentă continuă cu două diode electro-luminiscente
2. Semnalizare intermitentă variabilă cu două diode electro-luminiscente
3. Semnalizare intermitentă continuă cu opt electro-luminiscente
4. Preluarea stării digitale de la un întreruptor
5. Redirecționarea semnalului digital de la o intrare înspre o ieșire digitală
6. Auto-menținerea stării la acționarea unui buton cu apăsare și revenire

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];
- ✓ 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 două variabile de tip număr întreg „led_1” și „led_2” având ca și valori „9” și „10”, numărul de ordine al terminalelor „D9” și „D10” de pe platforma Arduino

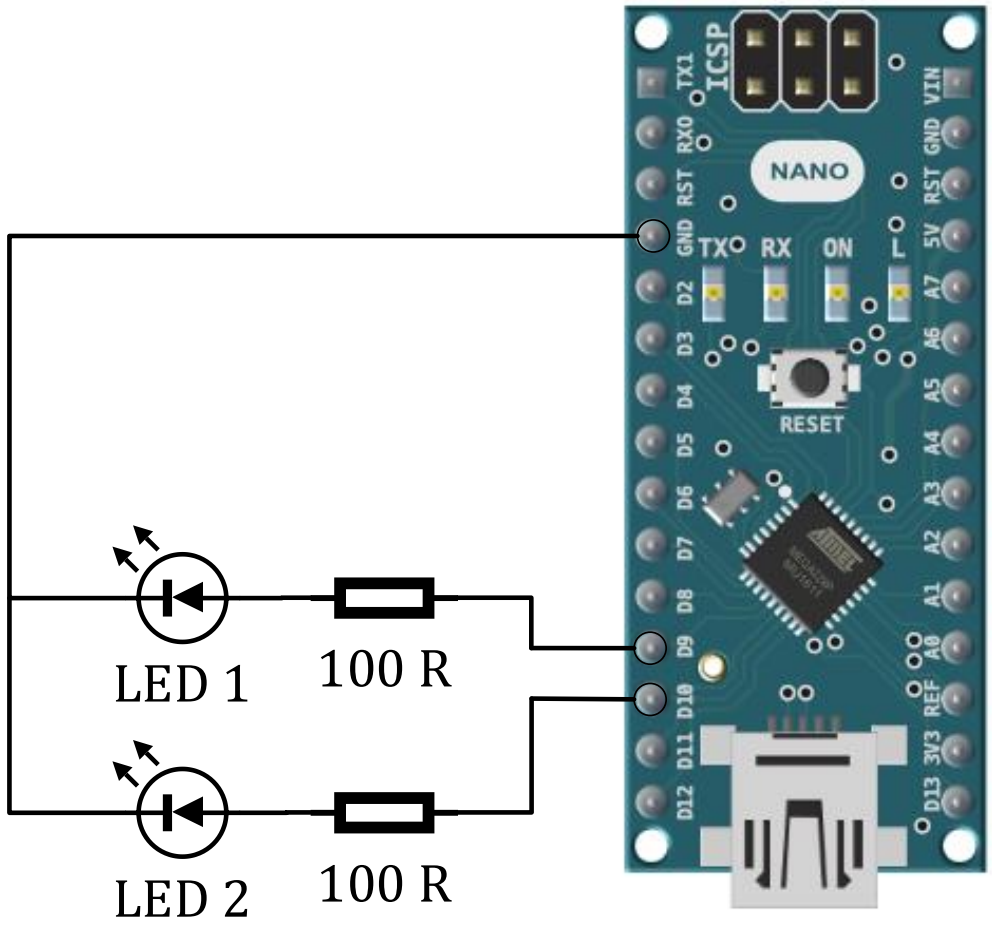
✓ configurarea terminalelor în modul de lucru „ieșire digitală” (eng. OUTPUT) în secțiunea „void setup()” cu ajutorul instrucțiunii „pinMode()”

✓ comutarea alternativă a stării logice a ieșirilor digitale (eng. HIGH / LOW) la o durată de o secundă (1000 [ms]) în cazul celor două terminale „D9” și „D10” prin intermediul funcției „digitalWrite()”

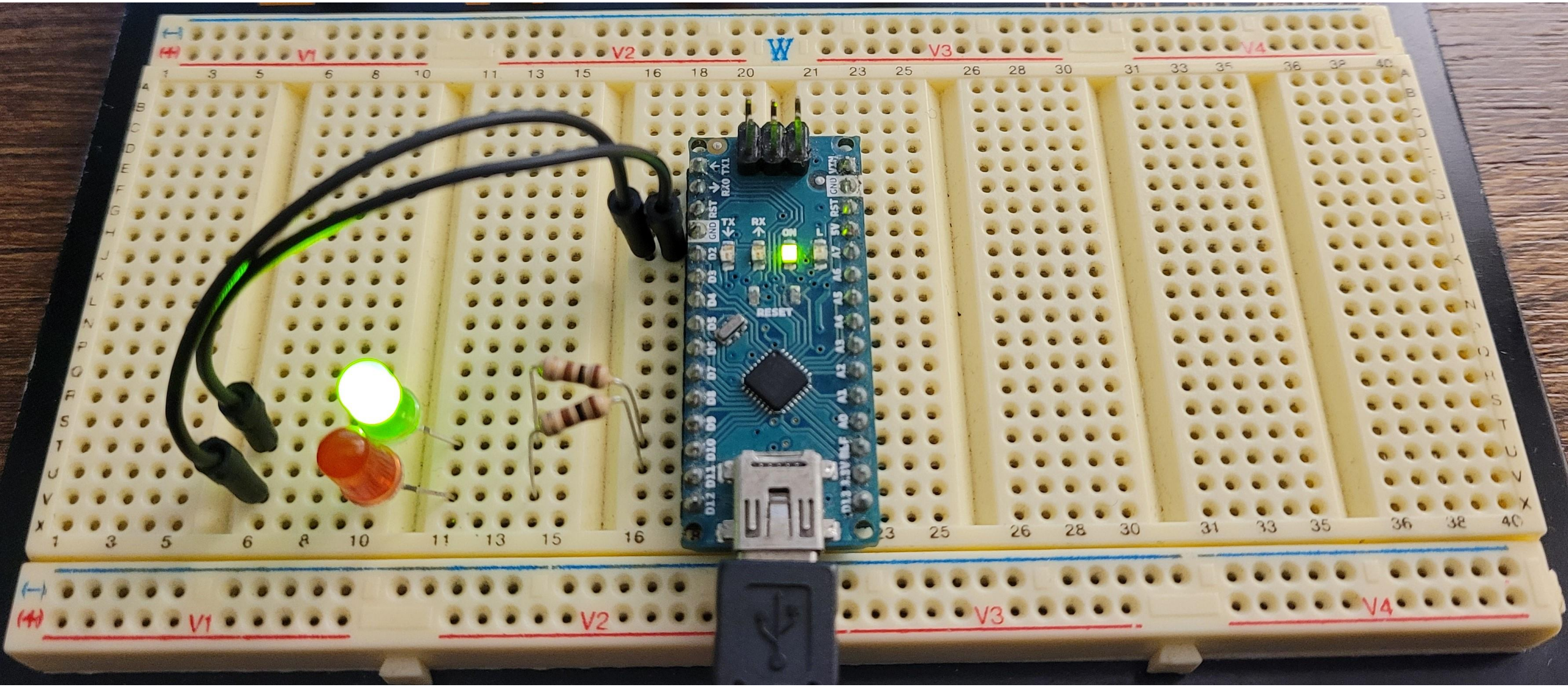
✓ efectul rezultant este semnalizarea intermitentă continuă (în undă plină) a celor două diode electro-luminiscente (LED) atașate la ieșirile digitale „D9” și „D10”

4. Implementarea aplicațiilor – Aplicația nr. 1

```
const int led_1 = 9;
const int led_2 = 10;

void setup() {
  pinMode(led_1, OUTPUT);
  pinMode(led_2, OUTPUT);
}

void loop() {
  digitalWrite(led_1, HIGH);
  digitalWrite(led_2, LOW);
  delay(1000);
  digitalWrite(led_1, LOW);
  digitalWrite(led_2, HIGH);
  delay(1000);
}
```



4. Implementarea aplicațiilor – Aplicația nr. 1



4. Implementarea aplicațiilor – Aplicația nr. 2

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

✓ declararea a două variabile de tip număr întreg „led_1” și „led_2” având ca și valori „9” și „10”, numărul de ordine al terminalelor „D9” și „D10” de pe platforma Arduino;

✓ declararea a două variabile de tip număr întreg pentru inițializarea valorii numerice a duratei de conducție sau a factorului de umplere atât pentru prima diodă cât și pentru a doua „dc_1” și „dc_2”

✓ declararea pasului de incrementare sau decrementare „i”

✓ configurarea terminalelor în modul de lucru „ieșire digitală” (eng. OUTPUT) în secțiunea „void setup()” cu ajutorul instrucțiunii „pinMode()”

✓ incrementarea valorii duratei de conducție „dc_1” specifică primei diode cu pasul „i”

✓ implementarea unei condiții pentru decrementarea pasului „i”

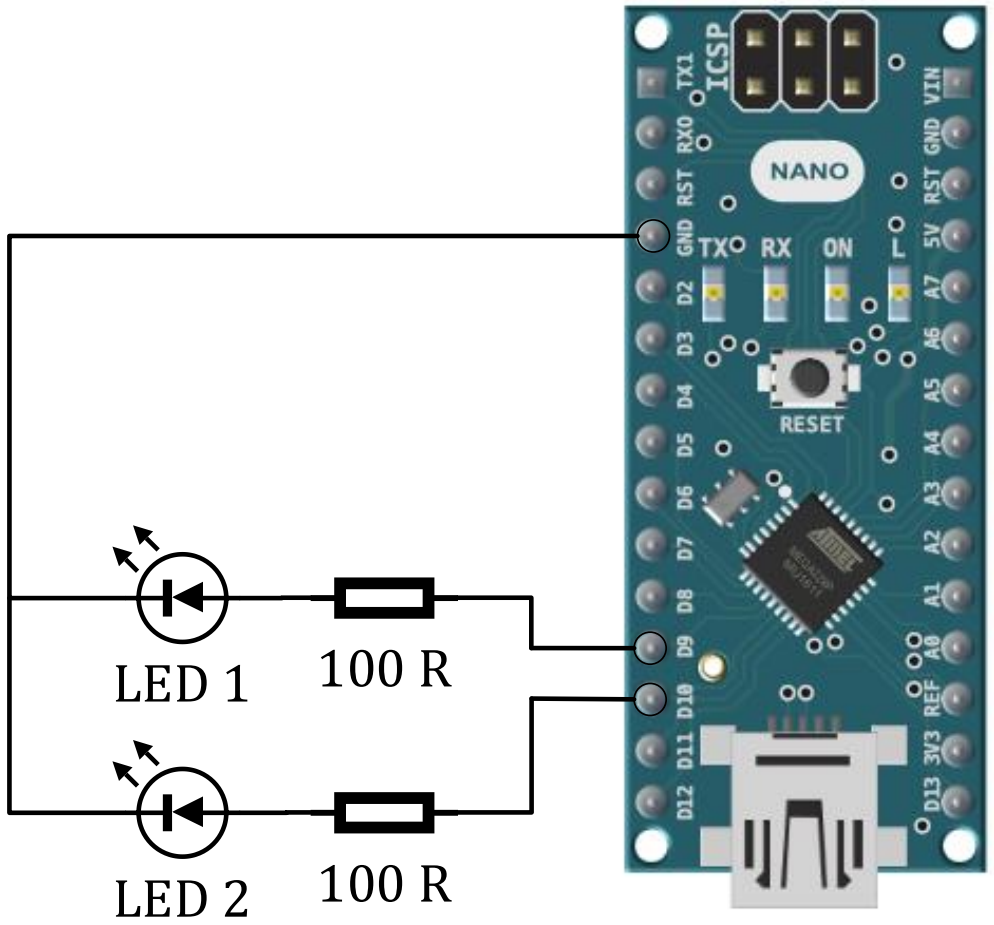
✓ determinarea valorii duratei de conducție „dc_2” pentru a doua diodă

✓ generarea a două trenuri de impulsuri cu lățime variabilă și frecvență constantă prin intermediul instrucțiunii „analogWrite()”

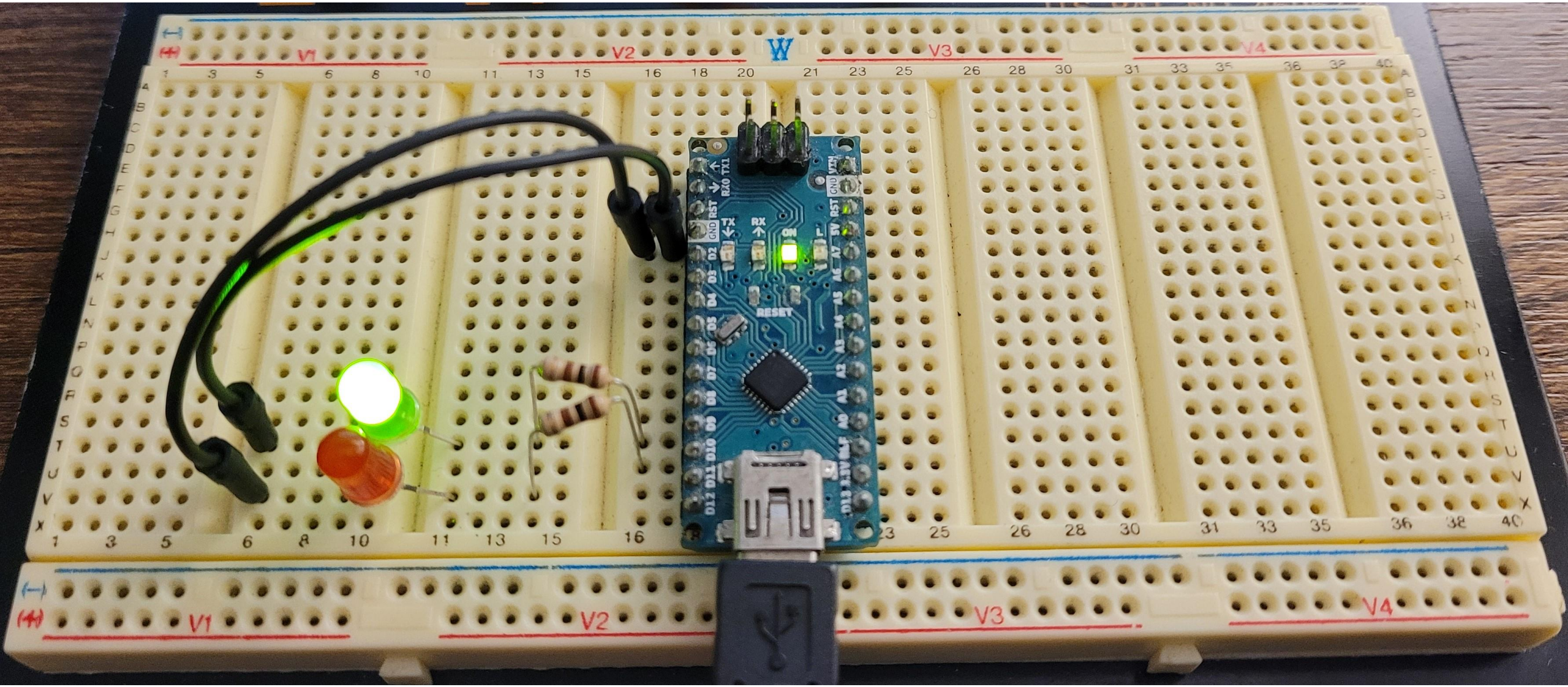
✓ efectul rezultant este semnalizarea intermitentă variabilă (în undă modulată) a celor două diode electro-luminiscente (LED) atașate la ieșirile digitale „D9” și „D10”

4. Implementarea aplicațiilor – Aplicația nr. 2

```
const int led_1 = 9;
const int led_2 = 10;
int dc_1 = 0;
int dc_2 = 0;
int i = 5;
void setup() {
  pinMode(led_1, OUTPUT);
  pinMode(led_2, OUTPUT);
}
void loop() {
  dc_1 = dc_1 + i;
  if (dc_1 <= 0 || dc_1 >= 255) {
    i = - i;
  }
  dc_2 = 255 - dc_1;
  analogWrite(led_1, dc_1);
  analogWrite(led_2, dc_2);
  delay(30);
}
```



4. Implementarea aplicațiilor – Aplicația nr. 2



4. Implementarea aplicațiilor – Aplicația nr. 3

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

✓ declararea unui șir finit de opt valori cu numerele de ordine ale terminalelor

✓ declararea unei constante globale prin intermediul căreia se poate regla frecvența de parcurgere a șirului finit

✓ stabilirea modului de lucru „ieșire digitală” pentru fiecare terminal în parte

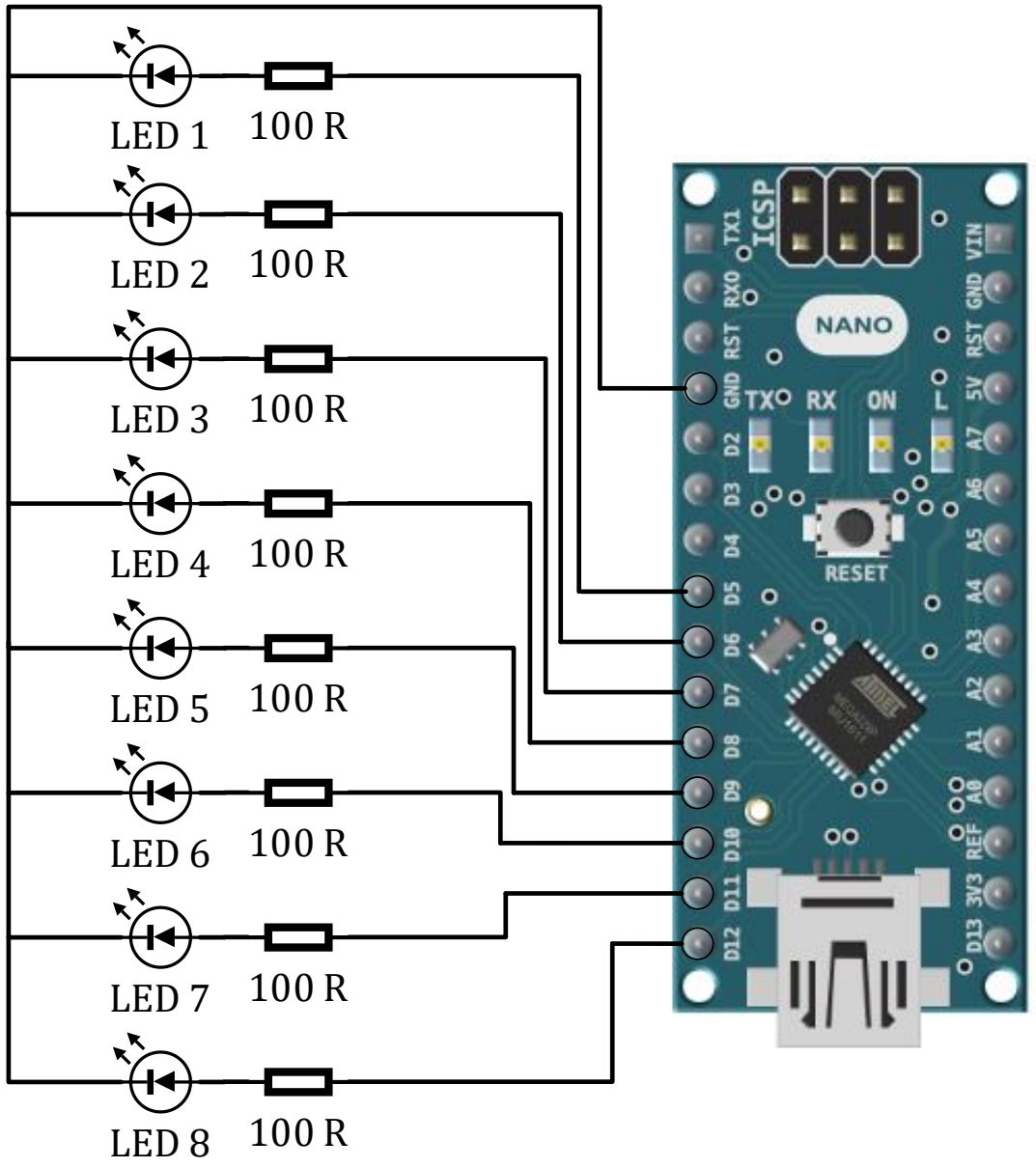
✓ resetarea contorului „j” la fiecare ciclu nou

✓ parcurgerea în sens crescător a șirului prin semnalizare intermitentă

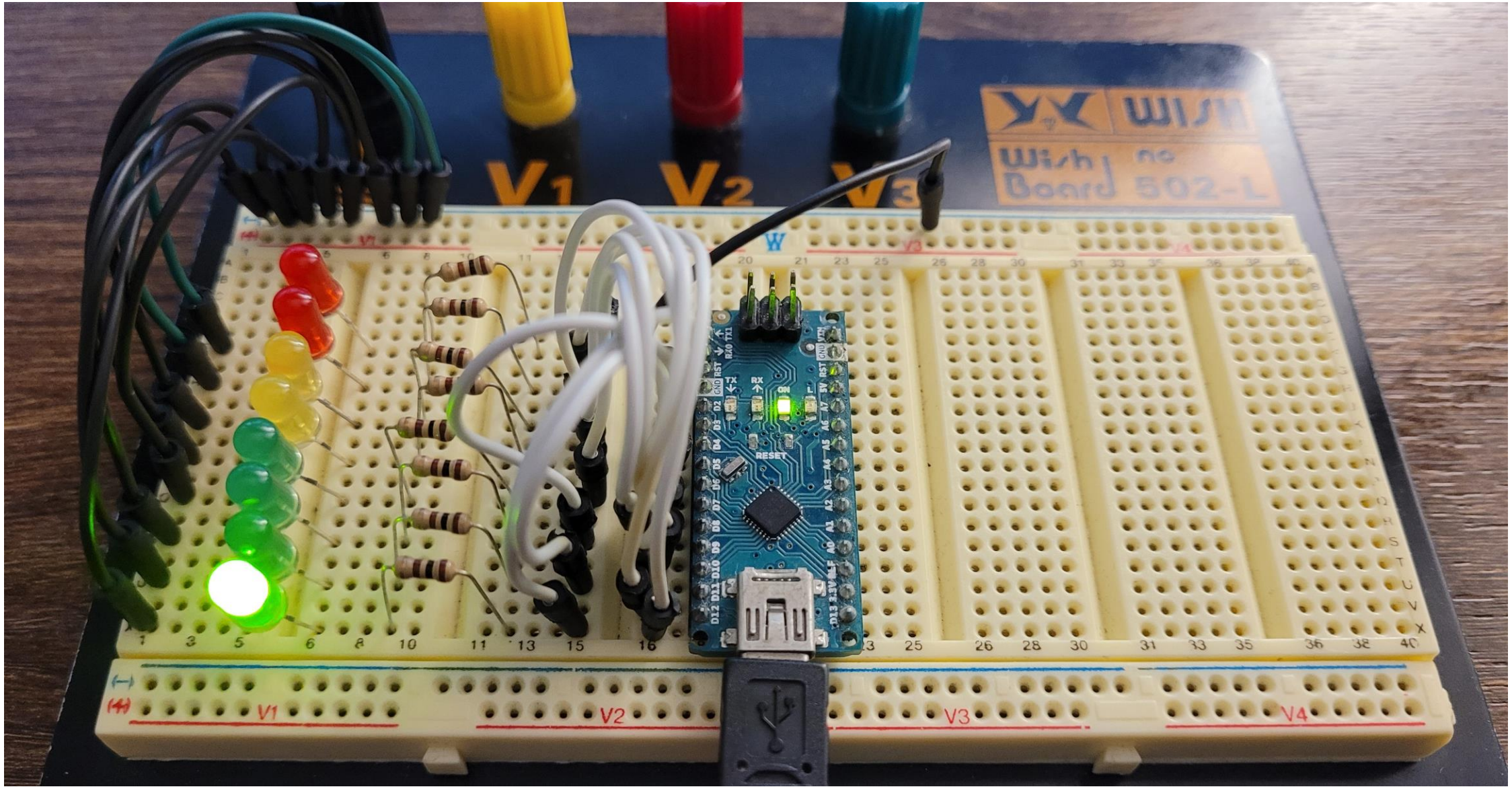
✓ parcurgerea în sens descrescător a șirului prin semnalizare intermitentă

4. Implementarea aplicațiilor – Aplicația nr. 3

```
int pin_led[] = {5, 6, 7, 8, 9, 10, 11, 12};
int intarziere = 50;

void setup() {
  for(int i = 0; i <= 7; i++) {
    pinMode(pin_led[i], OUTPUT);
  }
}
void loop() {
    int j = 0;
    for(j = 0; j <= 7; j++){
      digitalWrite(pin_led[j], HIGH);
      delay(intarziere);
      digitalWrite(pin_led[j], LOW);
      delay(intarziere);
    }
    for(j = 7; j >= 0; j--){
      digitalWrite(pin_led[j], HIGH);
      delay(intarziere);
      digitalWrite(pin_led[j], LOW);
      delay(intarziere);
    }
}
```



4. Implementarea aplicațiilor – Aplicația nr. 3



4. Implementarea aplicațiilor – Aplicația nr. 4

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

✓ declararea unei constante globale, pentru stabilirea numărului de ordine al terminalului

✓ stabilirea modului de lucru „intrare digitală” pentru terminalul ales

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

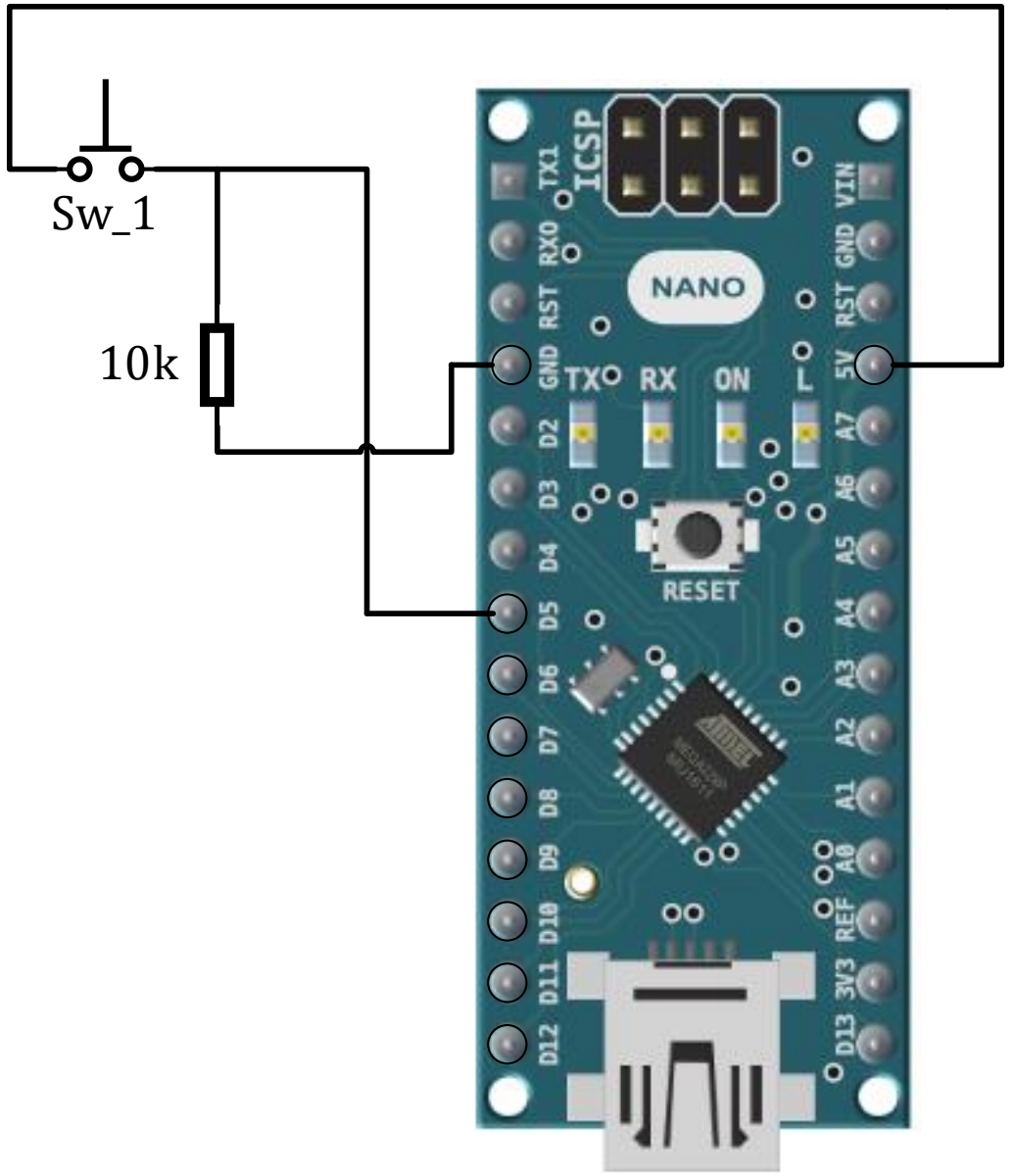
✓ preluarea stării logice a terminalului ales prin intermediul instrucțiunii „digitalRead ()”

✓ afișarea în consolă a mesajului text static „Stare contact: ” la un interval de 100 [ms]

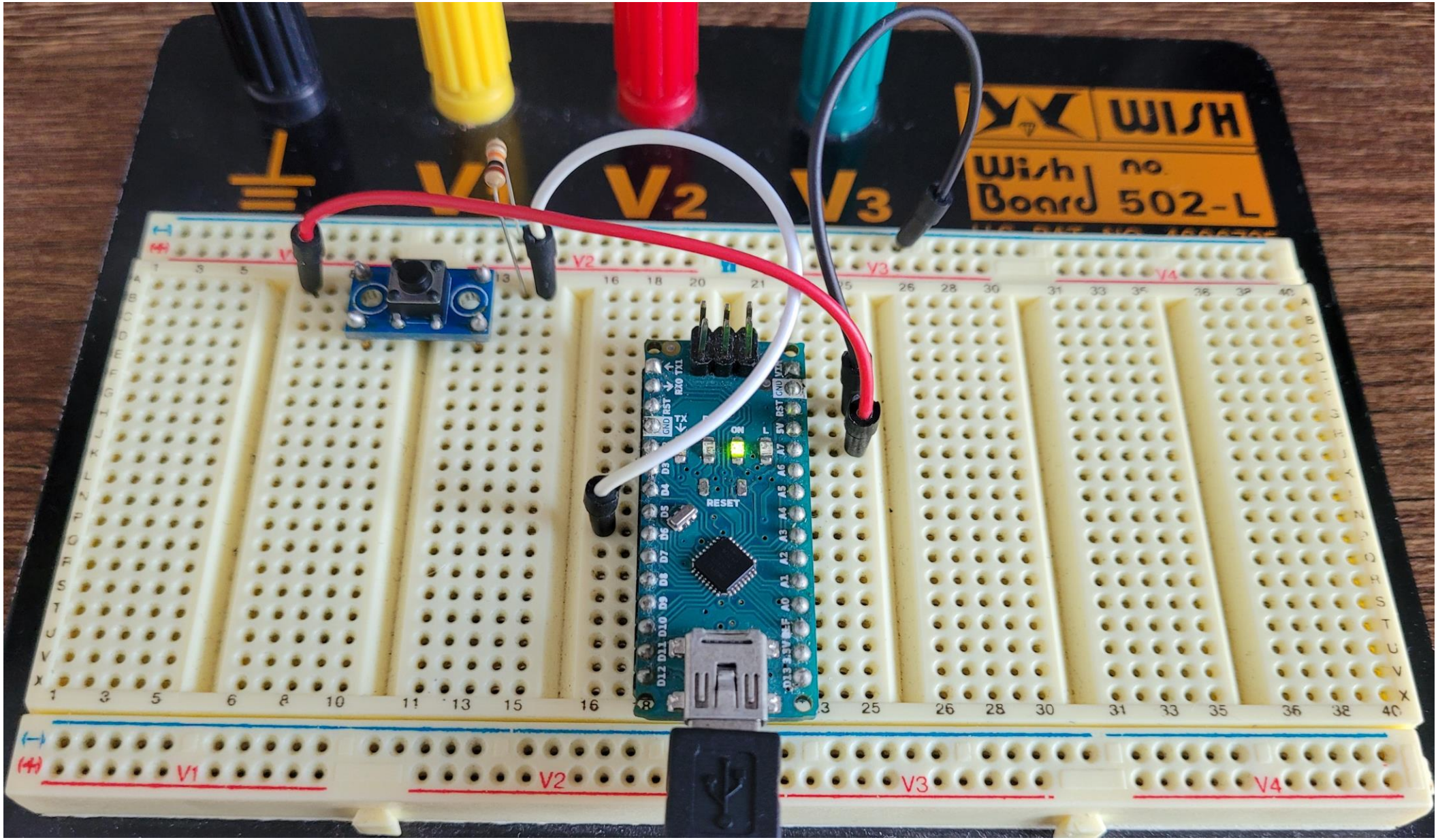
✓ afișarea stării logice a terminalului ales odată la 100 [ms]

4. Implementarea aplicațiilor – Aplicația nr. 4

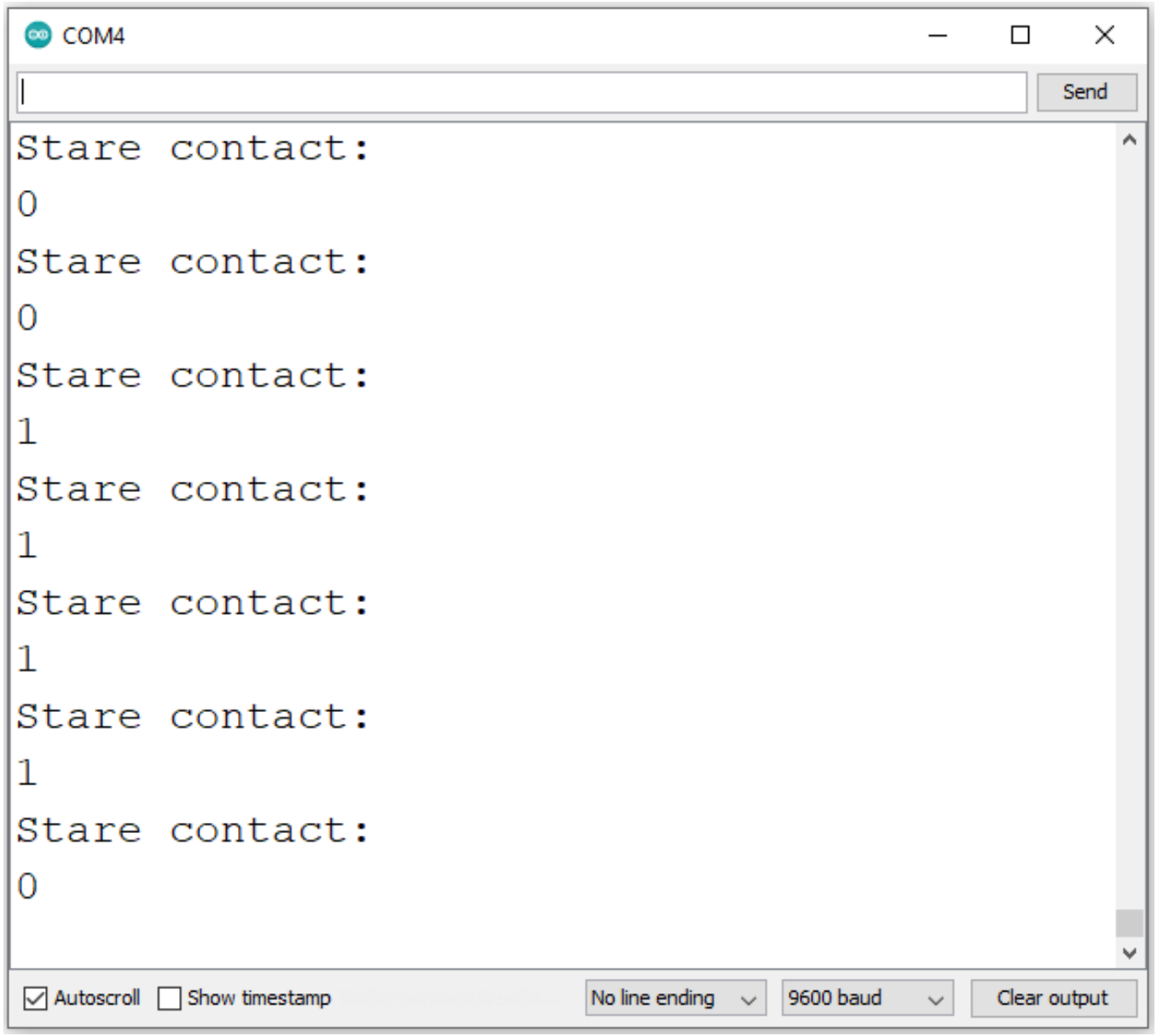
```
const int sw_pin = 5;

void setup() {
  pinMode(sw_pin, INPUT);
  Serial.begin(9600);
}

void loop() {
  int sw_state = digitalRead(sw_pin);
  Serial.println("Stare contact: ");
  Serial.println(sw_state);
  delay(100);
}
```



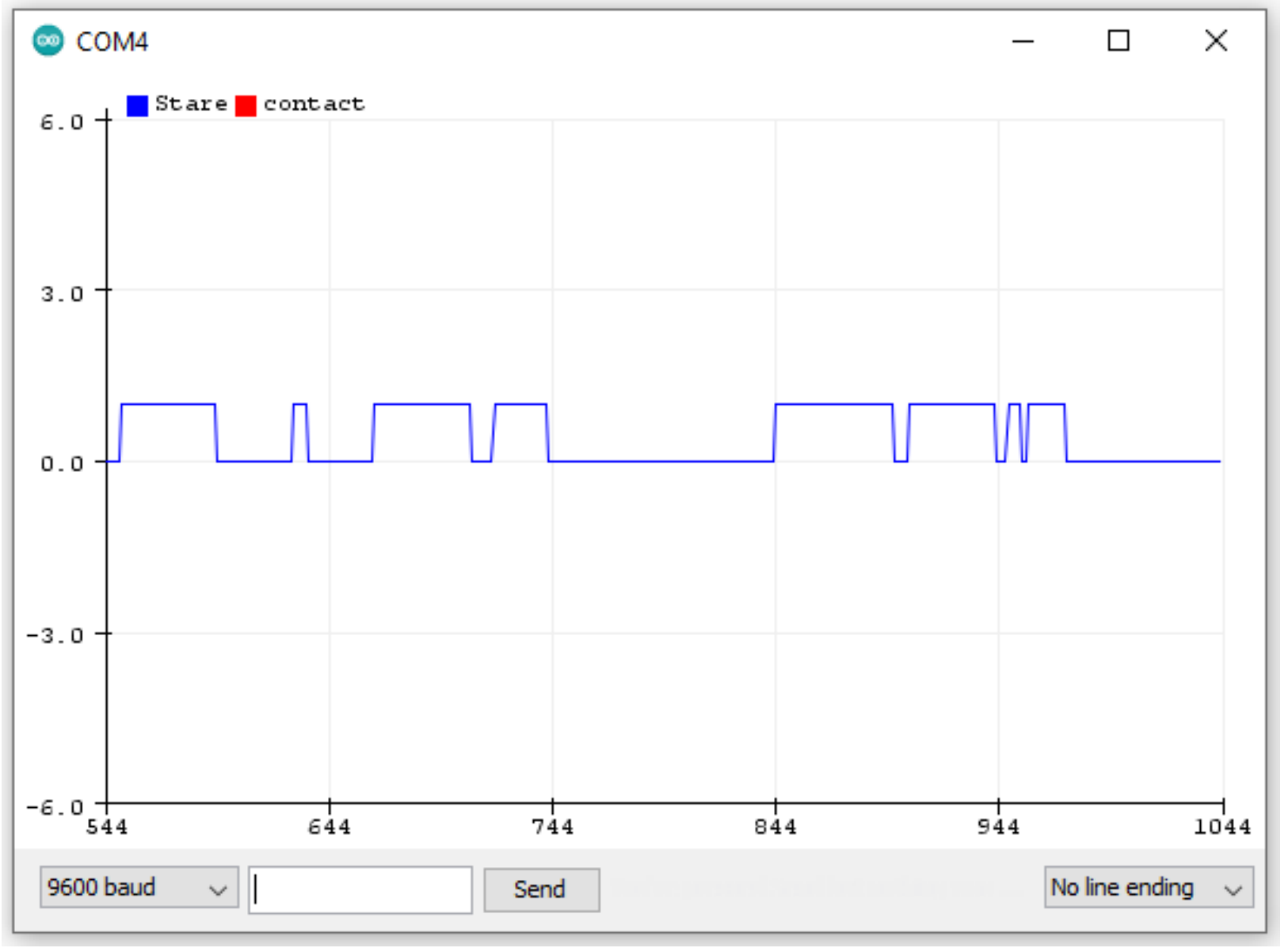
4. Implementarea aplicațiilor – Aplicația nr. 4



4. Implementarea aplicațiilor – Aplicația nr. 4



4. Implementarea aplicațiilor – Aplicația nr. 4



4. Implementarea aplicațiilor – Aplicația nr. 5

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

✓ 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)

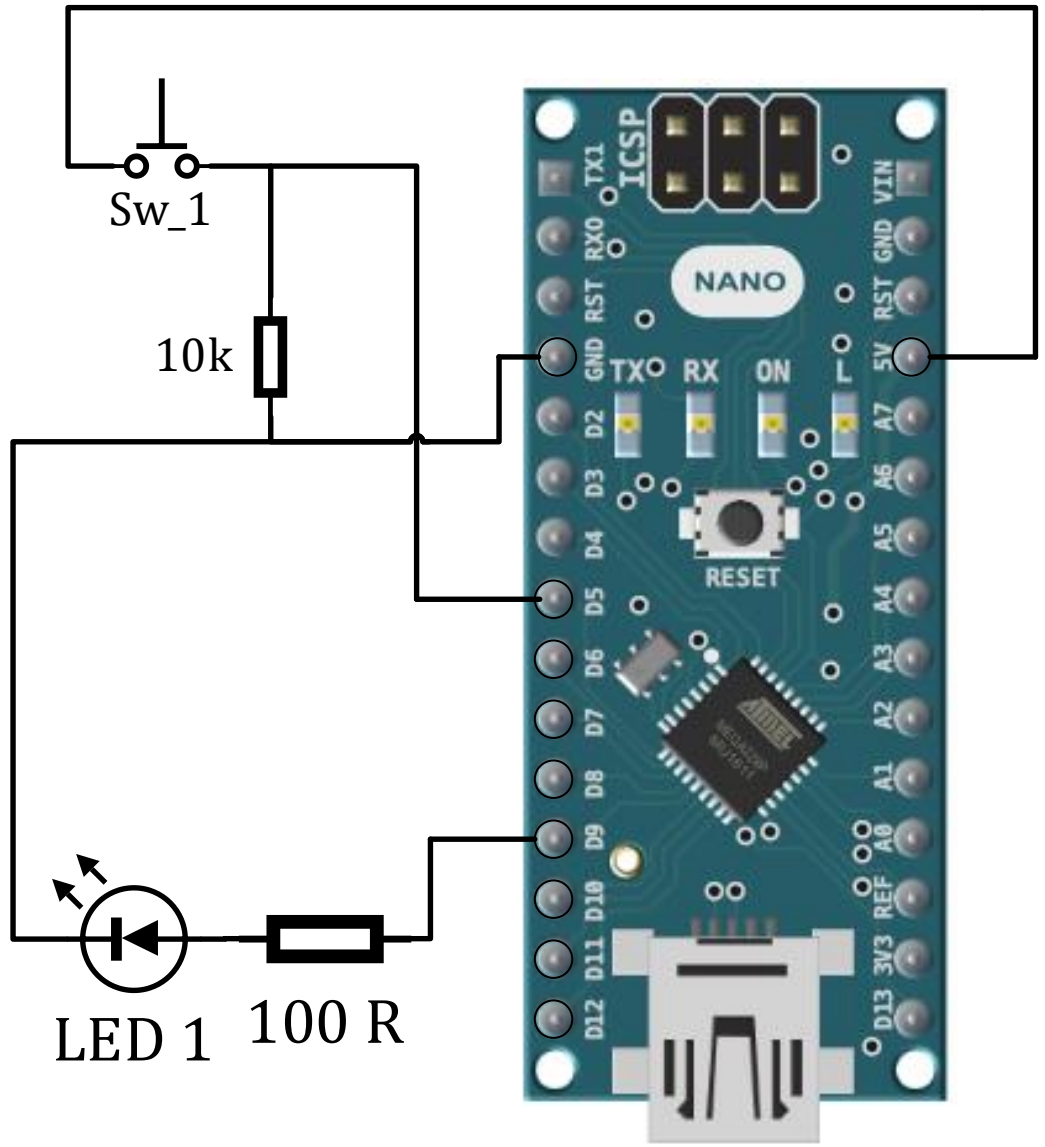
✓ preluarea stărilor logice ale terminalelor alese pentru atașarea butoanelor cu apăsare și revenire prin intermediul instrucțiunii „digitalRead ()”

✓ redirecționarea stării intrării digitale înspre terminalul de ieșire digitală (mai precis, crearea unei funcții de legătură între o intrare și o ieșire digitală prin intermediul arhitecturii procesorului)

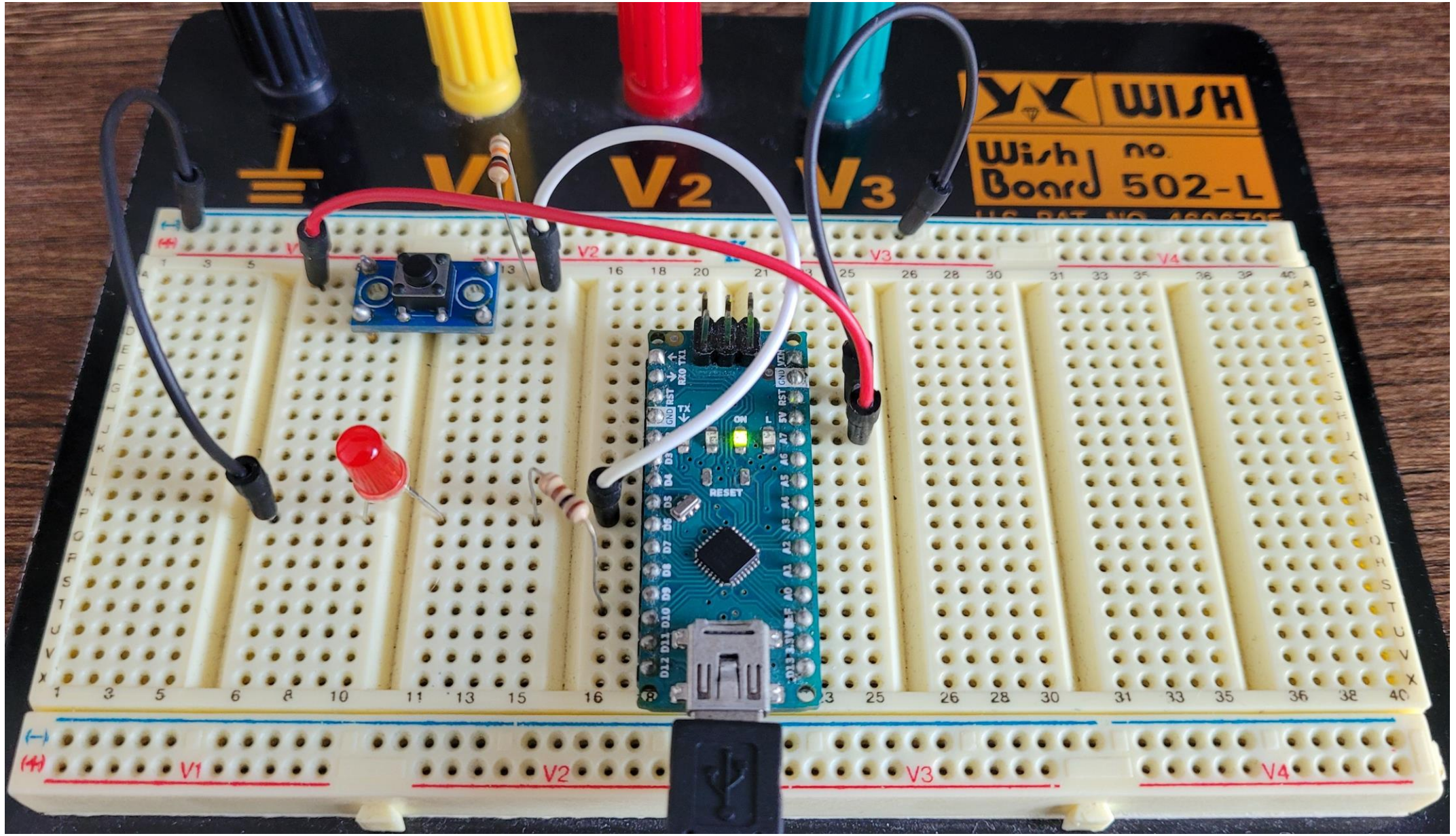
✓ crearea unei strategii de auto-menținere prin intermediul unei structuri de tip „while()”, și a unei instrucțiuni de suspendare de tip „break”

4. Implementarea aplicațiilor – Aplicația nr. 5

```
const int pin_sw = 5;
const int pin_led = 9;
int sw_state = 0;

void setup() {
  pinMode(pin_sw, INPUT);
  pinMode(pin_led, OUTPUT);
}
void loop() {
  sw_state = digitalRead(pin_sw);
  digitalWrite(pin_led, sw_state);
}
```



4. Implementarea aplicațiilor – Aplicația nr. 5



4. Implementarea aplicațiilor – Aplicația nr. 6

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

✓ declararea a două constante globale, în vederea stabilirii numerelor de ordine ale terminalelor corespondente atât butonului cu apăsare și revenire cât și diodei (LED)

✓ stabilirea modului de lucru „intrare digitală” pentru terminalul butonului

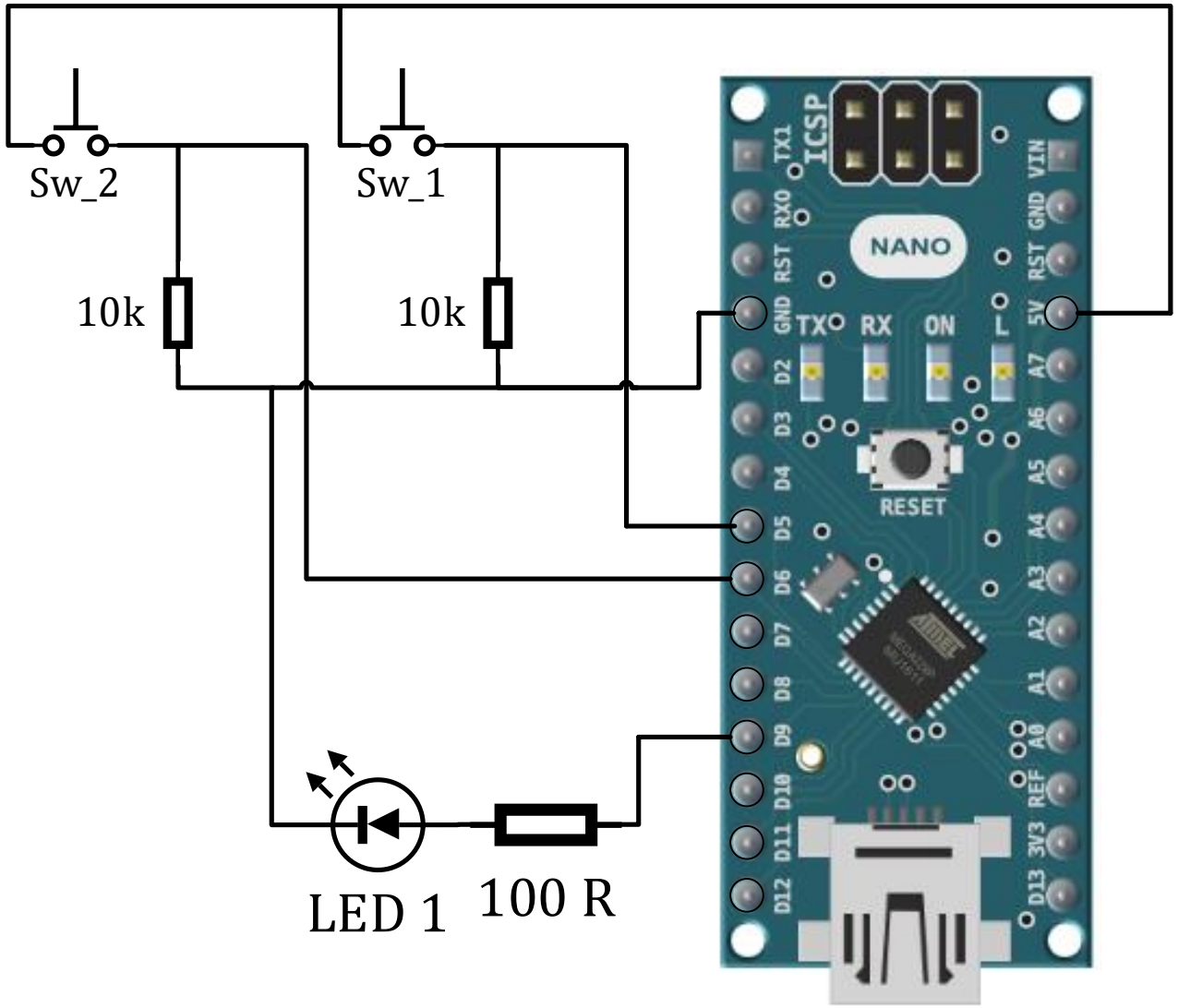
✓ stabilirea modului de lucru „ieșire digitală” pentru terminalul diodei (LED)

✓ preluarea stării logice a terminalului ales prin intermediul instrucțiunii „digitalRead ()”

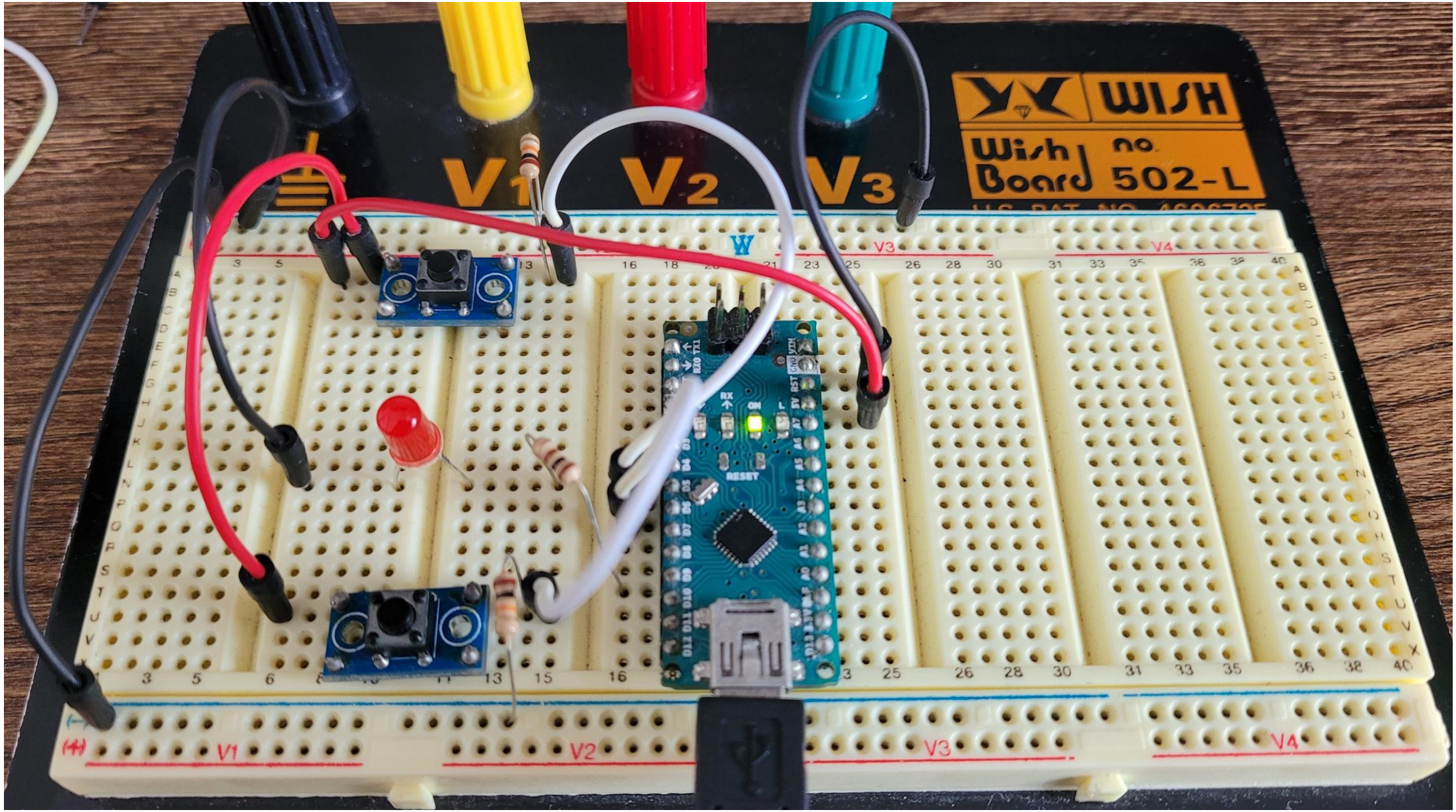
✓ redirecționarea stării intrării digitale înspre terminalul de ieșire digitală (mai precis, crearea unei funcții de legătură între o intrare și o ieșire digitală prin intermediul arhitecturii procesorului)

4. Implementarea aplicațiilor – Aplicația nr. 6

```
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;

void setup() {
  pinMode(pin_sw_1, INPUT);
  pinMode(pin_sw_2, INPUT);
  pinMode(pin_led, OUTPUT);
}
void loop() {
  sw_1_state = digitalRead(pin_sw_1);
  while(sw_1_state == 1) {
    sw_2_state = digitalRead(pin_sw_2);
    digitalWrite(pin_led, HIGH);
    if (sw_2_state == 1) {
      break;
    }
  }
  digitalWrite(pin_led, LOW);
}
```



4. Implementarea aplicațiilor – Aplicația nr. 6



5. Concluzii

- ✓ Simplificarea procesului de programare a microcontrolerului ATMega 328P din componența platformei de dezvoltare Arduino NANO, poate fi facilitată prin intermediul limbajului de interfațare (API) Wiring

- ✓ Acesta permite creare unor asocieri (în carul codului program) între ieșirile și intrările fizice ale microcontrolerului

6. Bibliografie

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

2. Arduino official website © 2024 – „Arduino IDE”
https://www.arduino.cc/en/software

3. Wikipedia – „Wiring (software)”
https://en.wikipedia.org/wiki/Wiring_(software)

4. Conf. Dr. Ing. Ioana - Cornelia Gros, Asist. Dr. Ing. Lucian - Nicolae Pintilie, Prof. Dr. Ing. Teodor Crișan Pană – „SISTEME EMBEDDED ÎN INGINERIE ELECTRICĂ - GHID DE APLICAȚII”, Editura UTPRESS, Cluj-Napoca, 2020, ISBN 978-606-737-431-5, recenzia: Conf. Dr. Ing. Ioan Incze Iov, Șl. Dr. Ing. Călin Cenan;

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/

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

7. Edis Techlab © 2024 Alle Rechte vorbehalten – „Der Arduino Nano - Pin-out”
https://edistechlab.com/arduino-nano-pinout/