Sisteme cu microprocesoare
– Manipularea intrărilor şi ieşirilor digitale –

I. SCOPUL LUCRĂRII:

Lucrarea de laborator are ca scop:
- prezentarea platformei de dezvoltare Arduino Nano [1];
- prezentarea mediului de dezvoltare Arduino IDE [2];
- prezentarea noţiunilor specifice limbajului de programare „Wiring" [3];
- implementare aplicaţiilor specifice procesării semnalelor de natură digitală [4] [5];

II. INTRODUCERE:

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

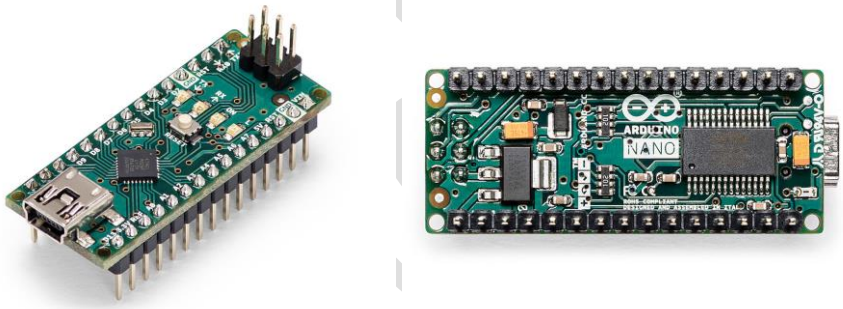

Fig. 1 – Platforma de dezvoltare Arduino Nano [1]

Dimensiunea redusă şi forma proeminentă a terminalelor de acces, conferă platformei de dezvoltare Arduino Nano posibilitatea instalării acesteia la nivelul unei plăcuţe pentru testare fără lipituri (eng. solderless breadboard) (Fig. 2).

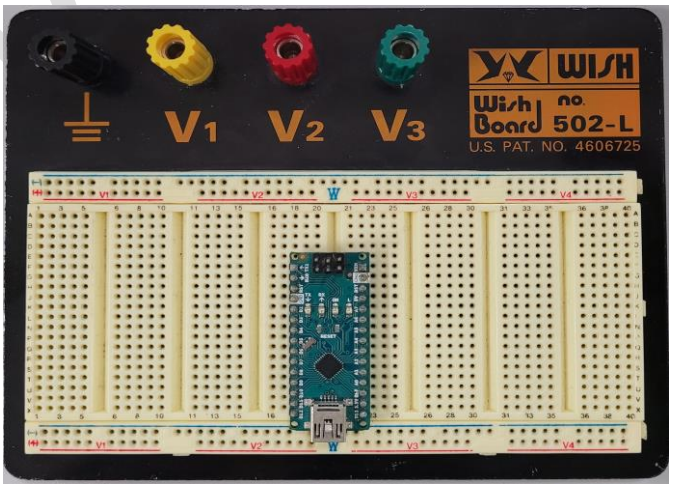

Fig. 2 – Instalarea platformei Arduino NANO la nivelul unei plăcuţe pentru testare

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

Harta de distribuţie a funcţiilor specifice terminalelor la nivelul platformei Arduino Nano este inscripţionată la nivelul cablajului imprimat. Suplimentar faţă de funcţiile standard inscripţionate pe cablaj, manualul de instrucţiuni, prezintă un set alternativ de funcţii pentru terminalele platformei de dezvoltare (Fig. 3).

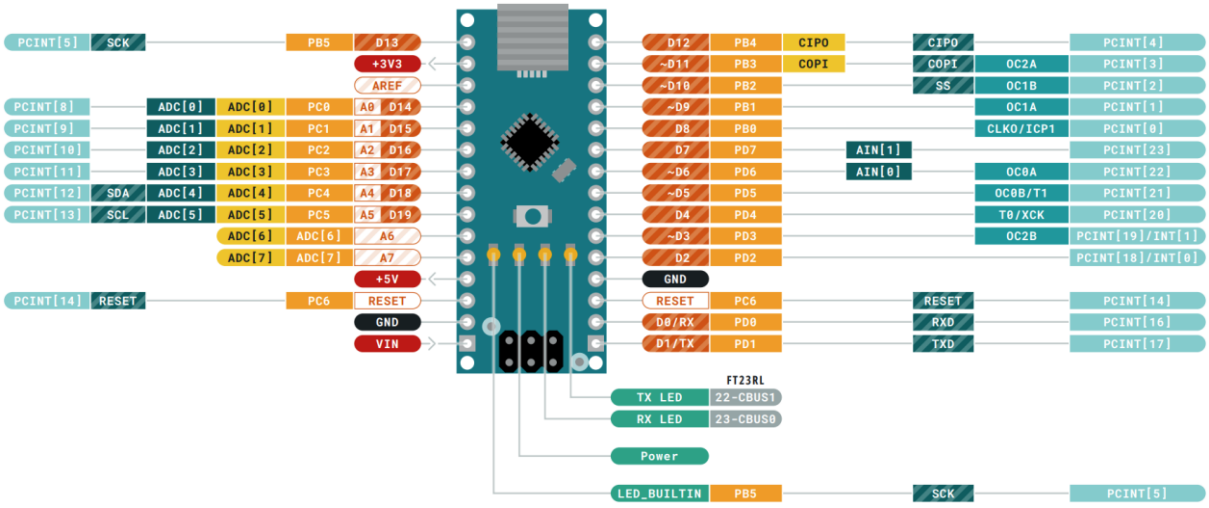

Fig. 3 – Harta funcţiilor specifice terminalelor platformei Arduino Nano [1]

Terminalele expuse la periferia platformei pot gestiona atât semnale digitale cât şi analogice. Notarea terminalelor pe cablajul imprimat al platformei se va realiza în funcţie de tipul semnalelor pe care acestea le gestionează. Spre exemplu, pentru terminalele care deservesc funcţia de preluare a semnalului analogic, indicativul acestora va fi „A_n” (terminal analogic, iar „n” reprezintă numărul de ordine al terminalului). Terminalele care deservesc funcţia de preluare şi furnizare a semnalelor digitale sunt notate „D_n”.

Alimentarea cu tensiune a platformei de dezvoltare Arduino Nano se realizează prin intermediul conectorului mini-USB sau prin intermediul terminalului „V_{in}”.

Programarea microcontrolerului ATMega 328 din componenţa platformei de dezvoltare Arduino Nano se realizează prin intermediul dispozitivului programator USB – Serial FTDI FT23RL [1].

Există şi alte variante constructive ale platformei de dezvoltare Arduino, precum UNO, Mega, Leonardo, Duemilanove etc. Există şi platforme asemănătoare şi compatibile cu mediul Arduino IDE, precum SainSmart UNO, Seeduino şi 4Duino.

Mediul de dezvoltare Arduino IDE [2] (eng. Integrated Development Environment), reprezintă pachetul de programe şi instrumente necesare în vederea parcurgerii procesului de programare a microcontrolerului ATMega 328 din componenţa platformelor de dezvoltare Arduino. În cadrul mediului Arduino IDE sunt incluse în mod implicit o serie de biblioteci şi fişiere sursă care deservesc funcţiile de bază ale microcontrolerelor din familia AVR. Tot în cadrul mediului de dezvoltare Arduino IDE există şi un compilator, care, pe baza fişierelor de definiţie, transpune instrucţiunile de program scrise în limbaj C standard sau C++ în instrucţiuni de procesor în format executabil de tip „.hex”. În cadrul prezentei lucrări, se va utiliza mediul Arduino IDE versiunea 1.8.19 pentru sistemul de operare Microsoft Windows (Fig. 4).

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

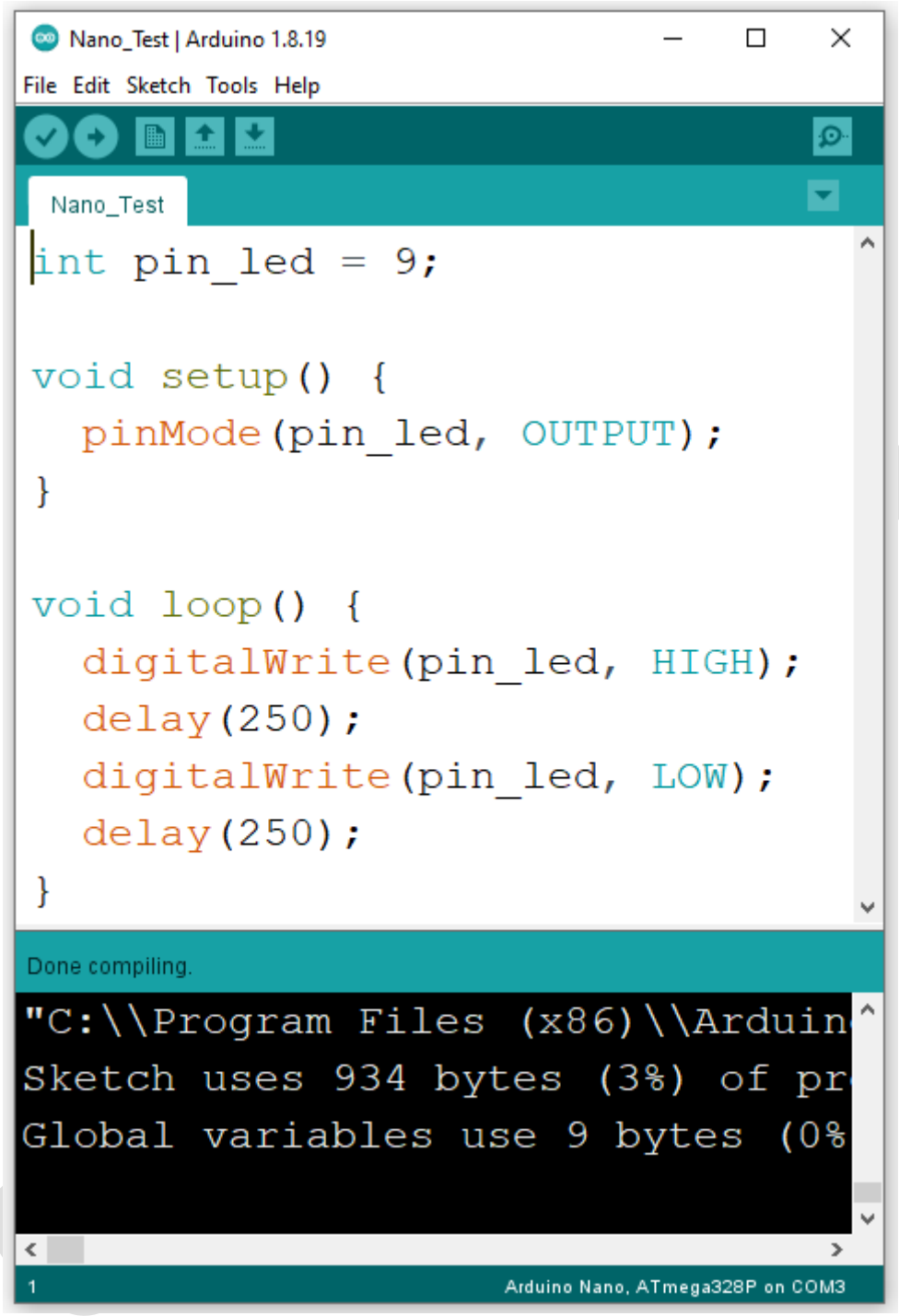

Fig. 4 – Mediul de dezvoltare Arduino IDE versiunea 1.8.19

În cadrul mediului Arduino IDE versiunea 1.8.19 se disting cinci secţiuni:
- secţiunea meniurilor principale, a paletei de instrumente şi comenzi rapide;
- editorul de text pentru redactarea codului program;
- bara de stare pentru afişarea etapei actuale;
- consola pentru descriere a operaţiei actuale;
- bara de afişare a rândului actual şi a platformei actuale împreună cu setările aferente;

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

Secţiunea meniurilor principal, a paletei de instrumente şi comenzi rapide, permite efectuarea diverselor operaţii de parametrizare a procesului de programare şi a platformei de dezvoltare (ex. alegerea altei platforme sau iniţierea procesului de compilare şi încărcare a codului program în memoria microcontrolerului) (Fig. 5).

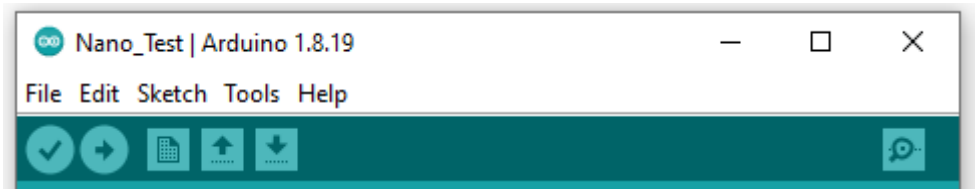

Fig. 5 – Secţiunea meniurilor principale, a paletei de instrumente şi comenzi rapide

Editorul de text permite redactarea codului program după regulile de formatare ale limbajului Wiring sau C++ (Fig. 6).

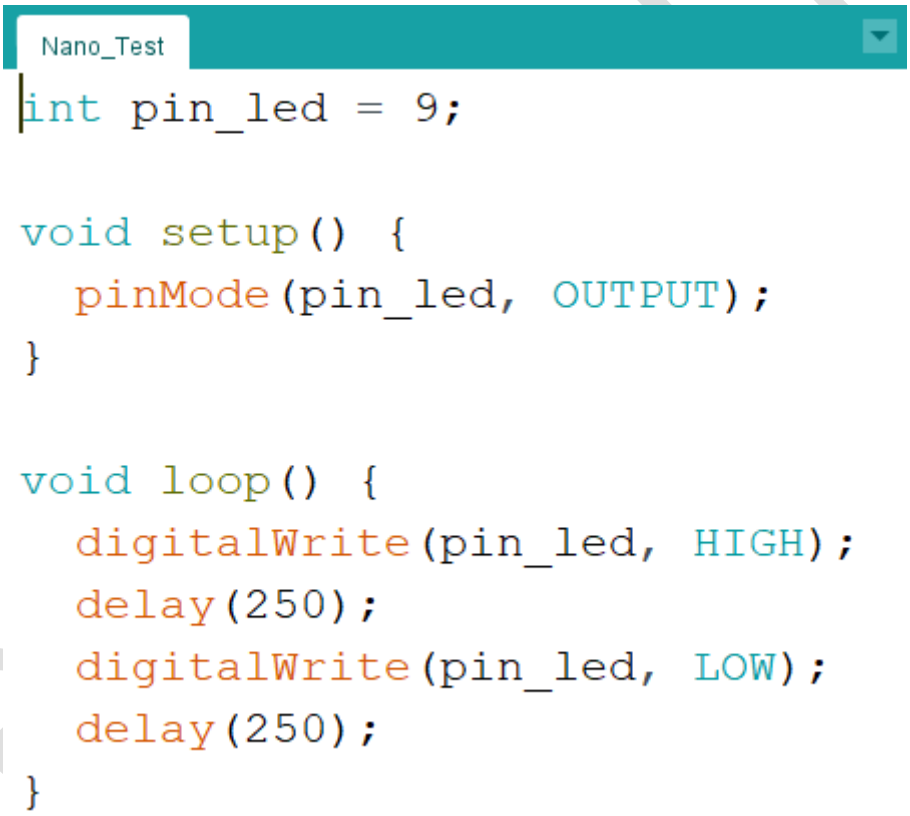

Fig. 6 – Editorul de text

Bara de stare afişează etapa actuală a procesul de programare şi indicatorul de progres atunci când etapa considerată prezintă un grad ridicat de complexitate (Fig. 7).

Fig. 7 – Bara de stare

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

Consola afișează descrierea în mod detaliat a operațiilor efectuate în vederea parcurgerii procesului de programare al microcontrolerului (Fig.8). Gradul de detaliere a descrierii operațiilor efectuate poate fi stabilit prin alegerea opțiunilor „Show verbose output during compilation and upload" din meniul „File" → „Preferences".

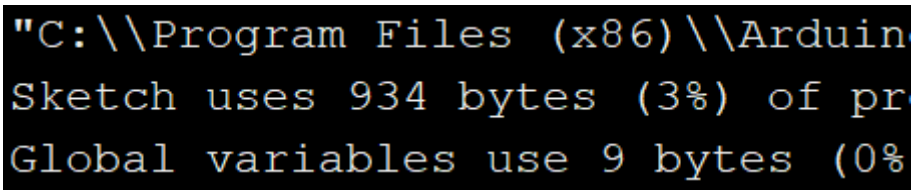

Fig. 8 – Consola pentru descriere a operației actuale

Bara pentru afișare a rândului actual și a platformei actuale împreună cu setările aferente permite utilizatorului să efectueze operații de verificare și diagnosticare a potențialelor probleme care au loc în cadrul procesului de programare (Fig. 9). Tot în cadrul acestei secțiuni, există posibilitatea de a identifica portul pentru comunicația „Serial" (ex. COM3) sau tipul dispozitivului programator utilizat.

Fig. 9 – Bara pentru afișare a rândului actual și a platformei actuale

În cadrul mediului Arduino IDE proiectele poartă denumirea (eng.) „sketch" (schiță de lucru). Extensia fișierului de proiect generat în Arduino IDE poate să fie „.ino" sau „.pde". Fișierele care conțin codul mașină rezultant pot avea fie extensia „.hex" (fișier cu conținut numeri reprezentat în cod hexazecimal) sau fișier „.bin" (fișier cu conținut numeric reprezentat în cod binar) (Fig. 10).

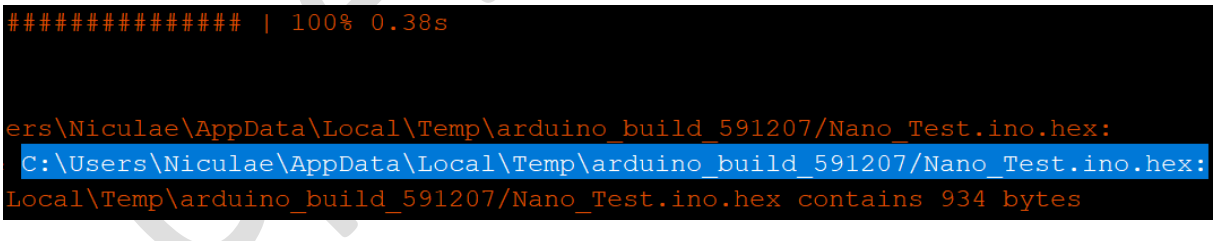

Fig. 10 – Fișierul care conține codul mașină rezultant în format numeric hexazecimal

Fișierul rezultant „.hex" în format numeric hexazecimal, va fi transferat în memoria microcontrolerului prin interfața USB – Serial FTDI cu ajutorul programului utilitar „avrdude". Acesta rulează în fundal în consola de comandă invocată la rularea mediului Arduino IDE. Având în vedere atât comportamentul aplicației „avrdude", care rulează în fundal ca și un serviciu similar serviciilor din componența sistemului de operare UNIX / Linux, cât și modul de declarare al căilor de acces la fișiere (ex. „\avr\etc\avrdude.conf"), se poate afirma faptului că, aplicația „avrdude" și alte componente ale mediului Arduino IDE rulează în mod virtualizat pe platforma sistemului de operare Microsoft Windows.

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

Instrumentele necesare pentru a permite rularea aplicațiilor din sistemul de operare UNIX / Linux pe platforma sistemului de operare Windows, poartă denumirea CygWin.

III. ASPECTE TEORETICE:

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

Setul de declarații hardware atașat limbajului de programarea „C / C++", poartă denumirea „Wiring" sau în unele documentații de specialitate „Wiring C". Limbajul sau setul de instrucțiuni „Wiring", reprezintă un mod de a „intermedia" procesul de programare între persoana 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). Practic, fără pachetul de instrucțiuni Wiring, ar fi necesară deci programarea microcontrolerului în limbaj hibrid de tip (eng.) „mixed assembly – syntax", în cadrul căruia, registrele de memorie sunt manipulate în mod direct (a se revedea documentația din cadrul primului laborator, capitolul specific procedeului de implementare al aplicațiilor) [3].

Prin urmare, instrucțiunile „pinMode()" și „digitalWrite()", specifice setului API Wiring, au ca și operație corespondentă în limbajul de asamblare, manipularea în mod direct a registrelor „DDRD" (eng. Data Direction Register of port D) și „PORTD" (registrul specific grupului de terminale PORTD amplasate în periferia carcasei microcontrolerului). Spre exemplu, următoarea relație de echivalență poate fi exprimată astfel [6]:

```
pinMode(4, OUTPUT); ----> DDRD = B00001000;
digitalWrite(4, HIGH); ----> PORTD = B00001000;
```

Pentru elaborarea codului program, în cadrul mediului Arduino IDE, pot fi utilizate atât expresii și declarații hardware cât și elemente de sintaxă a limbajului „C / C++", dar și denumiri de registre dedicate precum „PORTD" și „DDRD".

Spre exemplu, pentru semnalizare intermitentă cu o diodă electroluminiscentă, se va elabora codul program în formatul următor:

```
int pin_led = 4;

void setup() {
  pinMode(pin_led, OUTPUT);
}
void loop() {
  digitalWrite(pin_led, HIGH);
  delay(250);
  digitalWrite(pin_led, LOW);
  delay(250);
}
```


Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

În cadrul codului program prezentat mai sus, se disting cinci funcţii şi structuri de funcţii specifice limbajului Wiring:
- void setup ();
- void loop ();
- pinMode ();
- digitalWrite ();
- delay ();

Secţiunea de cod conţinută în cadrul corpului de funcţie „void setup ()", reprezintă secvenţa de iniţializare, care se va executa o singură dată în următoarele condiţii:
- la alimentarea cu tensiune a microcontrolerului;
- la apăsarea butonului RESET de pe cablajul imprimat al platformei de dezvoltare;
- la deschiderea comunicaţiei Serial;

Secţiunea de cod conţinută în cadrul corpului de funcţie „void loop ()", reprezintă secvenţa principală de cod, care, se va executa în mod repetat la frecvenţa exprimată prin intermediul funcţiei de întârziere „delay ()". Instrucţiunea „void loop()" este echivalentă cu expresia „C / C++" „while(1)" sau „while(TRUE)". Execuţia unei astfel de secvenţe se repetă la infinit, iar dacă există elemente de calcul iterativ, rezultatul acestora se va actualiza odată cu fiecare etapă de execuţie.

Instrucţiunea „pinMode ()" stabileşte modul de lucru al unui terminal electric (eng. pin) din componenţa microcontrolerului (ex. intrare, eng. INPUT sau ieşire eng. OUTPUT). Instrucţiunea respectivă este valabilă doar în situaţia în care terminalul electric este capabil să gestioneze semnale digitale. Numărul de ordine al terminalului se stabileşte prin intermediul primului argument al funcţiei, iar modul de lucru se stabileşte prin intermediul celui de-al doilea argument:

```
pinMode(<numărul_de_ordine_al_terminalului>, <modul_de_lucru>);
```

Instrucţiunea „digitalWrite ()" stabileşte starea logică a unui terminalului electric din componenţa microcontrolerului (adică prezenţa sau absenţa tensiunii la nivelul terminalului electric). Instrucţiunea respectivă este valabilă doar în situaţia în care terminalul electric este capabil să gestioneze semnale digitale. . Numărul de ordine al terminalului se stabileşte prin intermediul primului argument al funcţiei, iar starea logică se stabileşte prin intermediul celui de-al doilea argument:

```
digitalWrite(<numărul_de_ordine_al_terminalului>, <starea_logică>);
```

Instrucţiunea „digitalRead ()", are un singur argument, numărul de ordine al terminalului şi poate returna starea digitală preluată în mod instantaneu de la terminal. Rezultatul instrucţiunii va fi stocat într-o variabilă tot timpul!

```
int <variabilă> = digitalRead(<numărul_de_ordine_al_terminalului>);
```

Instrucţiunea „delay ()" introduce un timp de întârziere la executarea unei secvenţe din cadrul codului program. Parametrul funcţiei, timpul de întârziere este redat în milisecunde. Instrucţiunea are un singur parametru:

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

```
delay(<timp_de_întârziere_ms>);
```

Practic, prin intermediul instrucţiunilor „pinMode”, „digitalWrite” şi „delay”, este posibilă, manipularea în mod direct a circuitelor logice care stau la baza implementării arhitecturii interne pentru intrările şi ieşirile digitale ale microcontrolerului (Fig. 11).

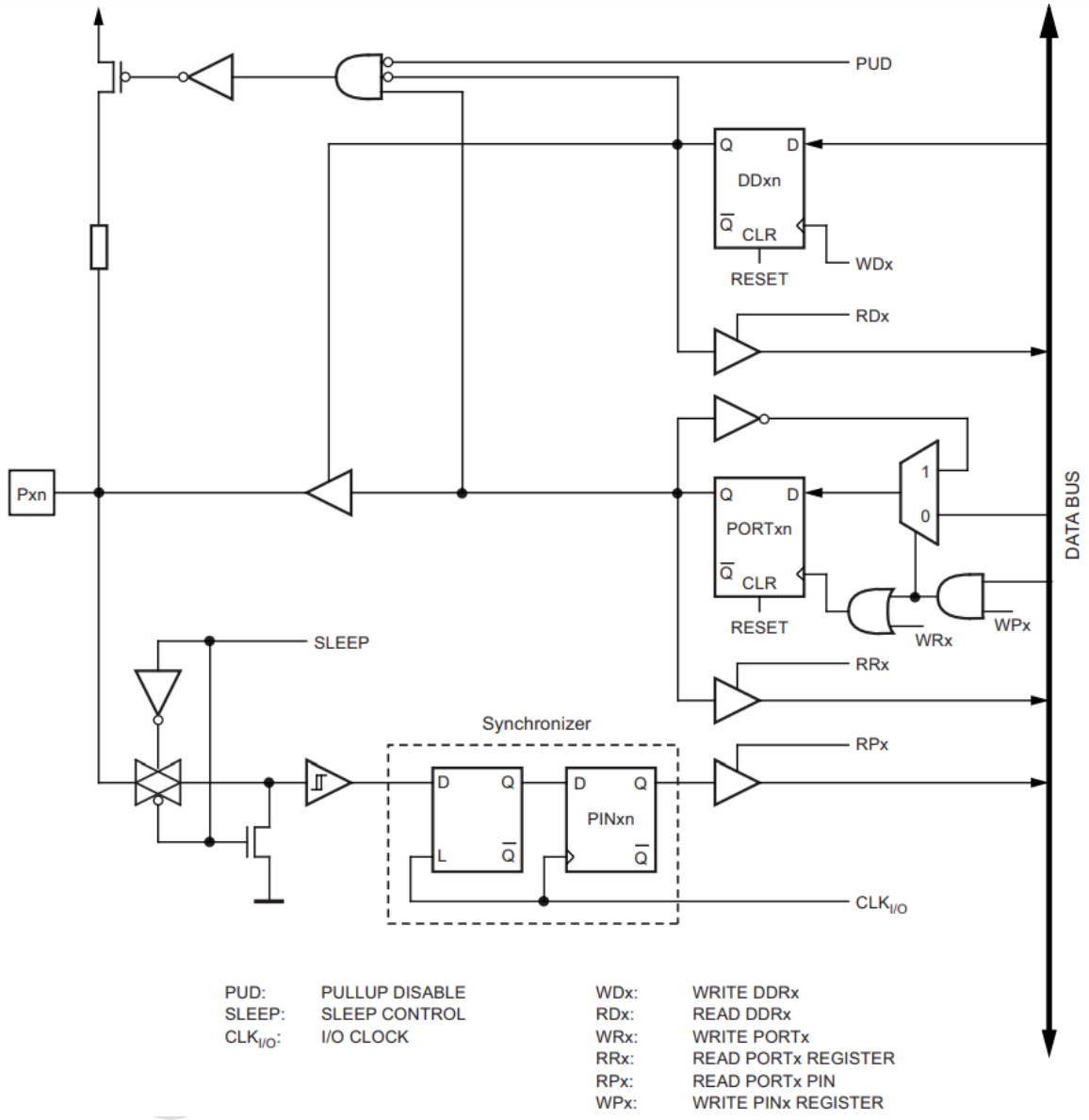

Fig. 11 – Arhitectura internă a microcontrolerului ATMega 328 – circuitele logice care stau la baza grupurilor de intrări şi ieşiri digitale [6]

Pe lângă instrucţiunile amintite anterior, tot în domeniul procesării semnalelor logice sau digitale, mai există şi instrucţiunea „analogWrite ()”. Chiar dacă denumirea ar sugera faptul că prin intermediul funcţiei s-ar genera un semnal de natură analogică, efectul acesteia este contrar denumirii, deoarece prin intermediul funcţiei date, se va genera un semnal de natură digitală sub forma unui tren de impulsuri cu amplitudine constantă, frecvenţă constantă dar lăţime sau durată a impulsurilor variabilă (factor de

umplere variabil sau durată de conducţie variabilă „d_c”). Efectul produs de un astfel de semnal digital, este similar cu efectul produs la alimentarea unui consumator cu un nivel variabil de tensiune (Fig. 12). În literatura de specialitate modulaţia în lăţime a impulsului poartă denumirea (eng. Pulse Width Modulation – PWM).

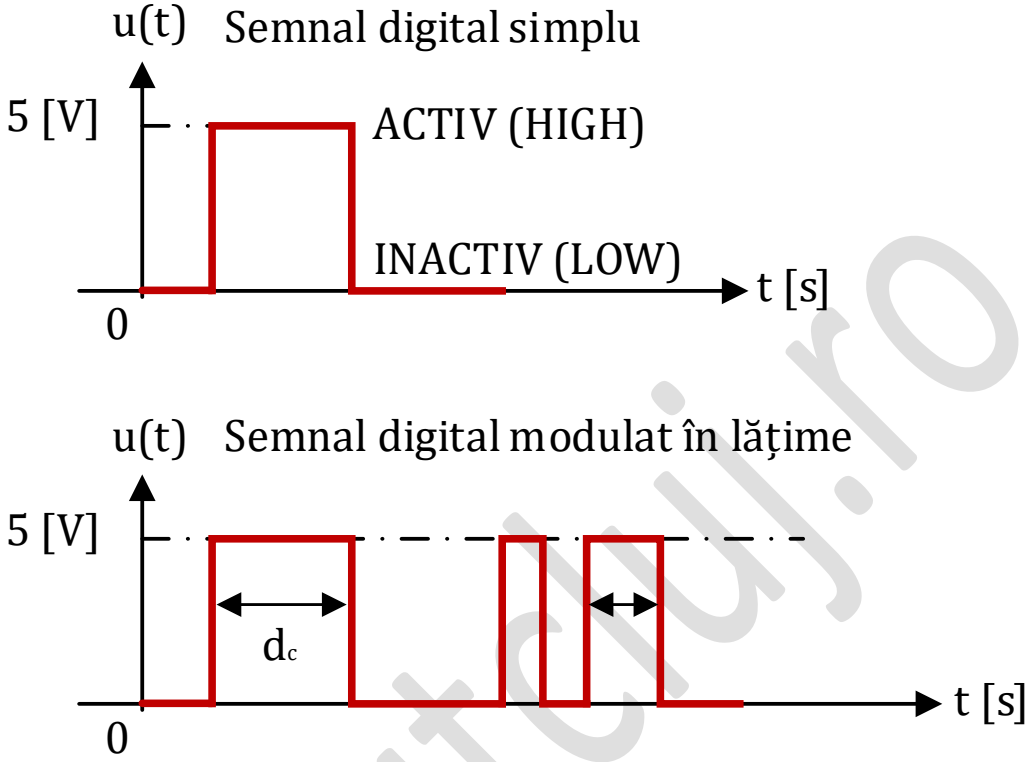

Fig. 12 – Semnale de natură digitală

Pentru diagnoză şi depanare, se va utiliza de asemenea comunicaţia USB – Serial stabilită între platforma de dezvoltare şi calculatorul gazdă. Conexiunea Serial va fi gestionată prin intermediul următoarele instrucţiuni:
- Serial.begin(<viteză_de_transfer_bps>) – pentru iniţializarea comunicaţiei Serial;
- Serial.println(<”text_mesaj_”>) – pentru afişarea mesajelor text în consola Serial;
În cadrul mediului Arduino IDE, consola Serial are denumirea (eng.) „Serial Monitor” şi poate fi regăsită în meniul (eng.) „Tools” (instrumente). Pe lângă consola Serial, mai există şi instrumentul pentru afişarea sub formă de grafic a valorilor vehiculate pe magistrala Serial, anume (eng.) „Serial Plotter” (afişaj grafic în funcţie de timp).

În vederea implementării unor instrucţiuni condiţionale şi iterative specifice sintaxei „C / C++” se vor utiliza următoarele instrucţiuni:
- if (<condiţie>) {};
- else (<condiţie>) {};
- for (<condiţie>; <instrucţiune_efectuată_la_fiecare_iteraţie>) {};

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

IV. IMPLEMENTAREA APLICAȚIILOR:

Pentru a studia facilitățile microcontrolerului ATMega 328 în contextul procesării semnalelor de natură digitală, se vor utiliza următoarele componente:
- 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;

În continuare, se propune deci, implementare celor șase aplicații:
- semnalizare intermitentă continuă cu două diode electro-luminiscente (eng. LED);
- semnalizare intermitentă variabilă cu două diode electro-luminiscente;
- semnalizare intermitentă continuă cu opt electro-luminiscente;
- preluarea stării digitale de la un întreruptor;
- redirecționarea semnalului digital de la o intrare digitală înspre o ieșire digitală;
- auto-menținerea stării digitale la acționarea unui buton cu apăsare și revenire;

APLICAȚIA 1:

Se va implementa circuitul conform următoarei scheme (Fig. 13):

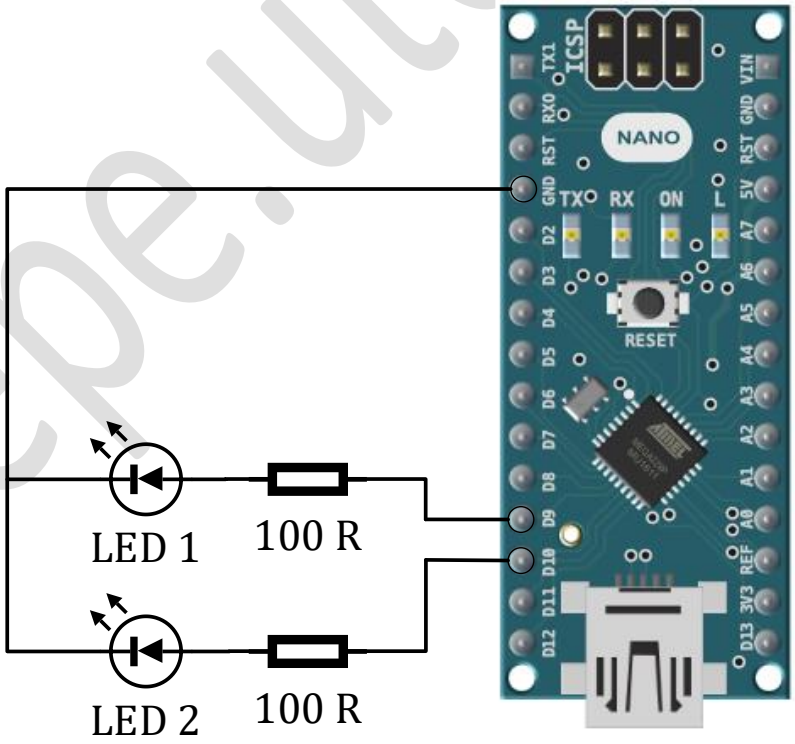

Fig. 13 – Schema electronică pentru implementarea circuitului specific aplicației 1 [7]

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

Se va implementa următorul cod program:

```
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);
}
```

În urma implementării circuitului și a codului program, se va transfera codul program în memoria microcontrolerului ATMega 328 din cadrul platformei de dezvoltare Arduino NANO. Pentru a transfera codul program se vor parcurge următoarele etape:

A. Din meniul „Tools” se va alege sub-categoria „Board” --> „Arduino AVR Boards” --> „Arduino Nano” (Fig. 14).

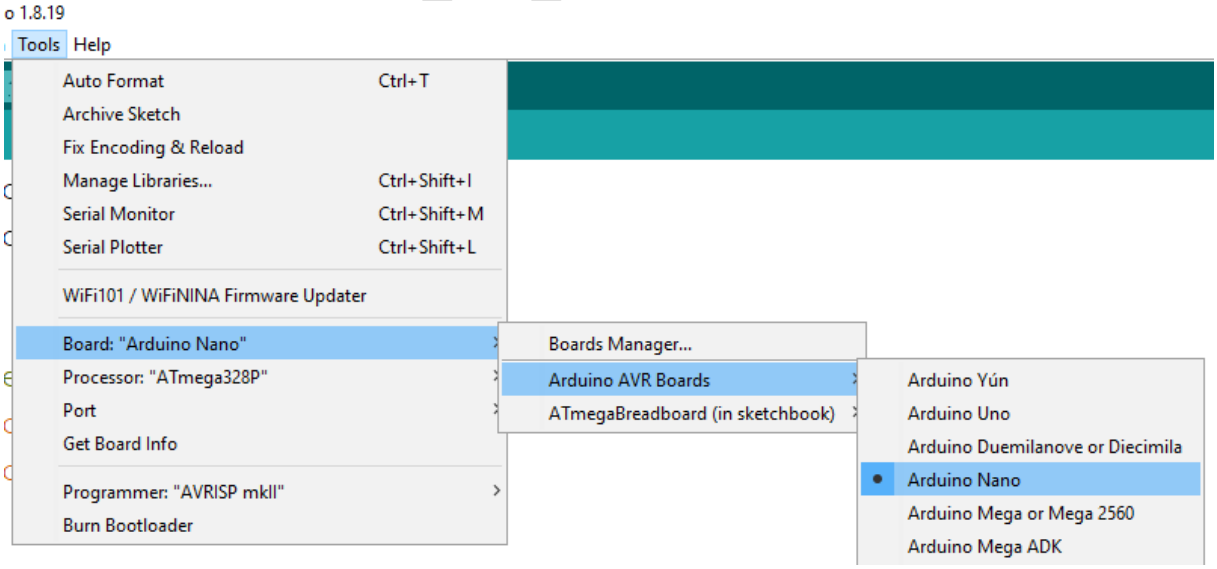

Fig. 14 – Alegerea platformei de dezvoltare care urmează să fie programată

B. Din meniul „Tools” se va alege sub-categoria „Processor” --> „ATMega 328P” (Fig. 15).

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

Fig. 15 – Alegerea variantei constructive a procesorului platformei de dezvoltare

C. Din meniul „Tools” se va alege sub-categoria „Port” --> „COMx” (Fig. 16).

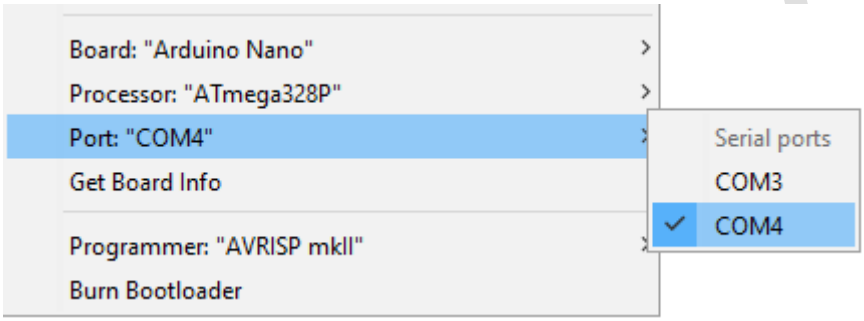

Fig. 16 – Alegerea portului pentru comunicația Serial

D. Din meniul „Sketch” se va alege opțiunea „Upload” (Fig. 17).

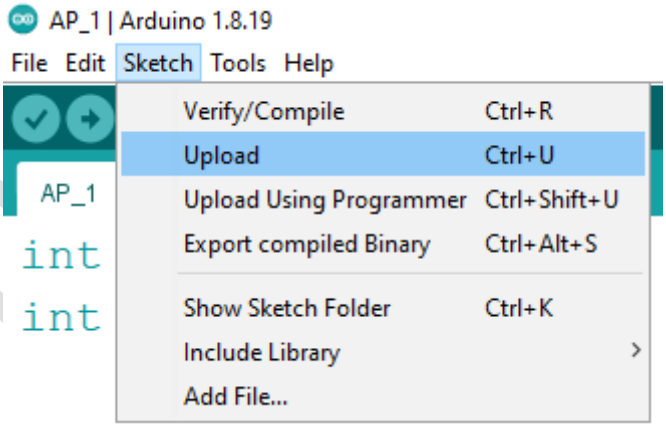

Fig. 17 – Încărcarea codului program în memoria microcontrolerului

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

În urma încărcării aplicației se va verifica funcționalitatea montajului (Fig. 18).

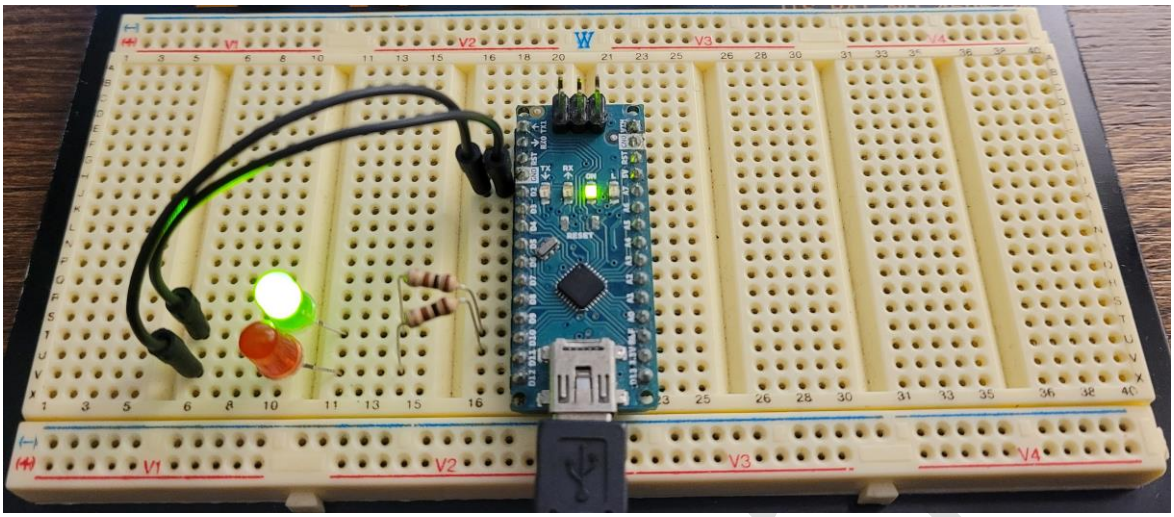

Fig. 18 – Montaj experimental pentru aplicația 1 și 2

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" (Fig. 18).

APLICAȚIA 2:

Pe baza aceluiași circuit de la aplicația nr. 1 (Fig. 13 și 18), se va implementa semnalizarea intermitentă cu variația intensității luminoase a celor două diode electro-luminiscente atașate la terminalele „D9" și „D10".

Se va implementa următorul cod program:

```
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);
}
```


Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

```
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);
}
```

OBSERVAȚIE: Instrucțiunea „analogWrite ()” necesită două argumente, anume, numărul de ordine al terminalului și factorul de umplere sau durata de conducție. Factorul de umplere este exprimat în format numeric zecimal printr-o valoare cuprinsă în intervalul [0 – 255]. Intervalul de valori corespunde unei mărimi exprimate pe 8 biți în format numeric binar.

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” (Fig. 18).

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

APLICAȚIA 3:

Se va implementa circuitul conform următoarei scheme (Fig. 19):

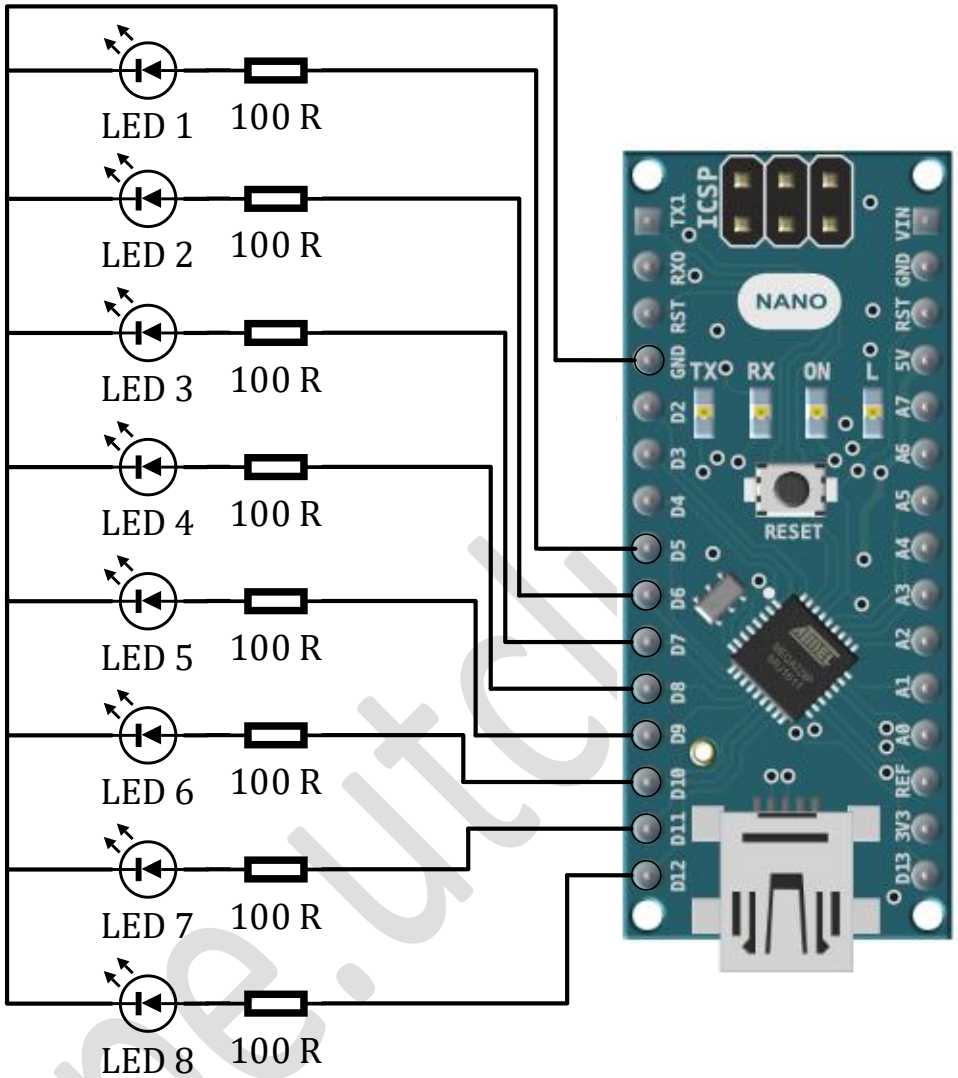

Fig. 19 – Schema electronică pentru implementarea circuitului specific aplicației 3 [7]

Se va implementa următorul cod program:

```
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);
  }
}
```


Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

```
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);
    }
}
```

Montajul experimental se va realiza conform (Fig. 20):

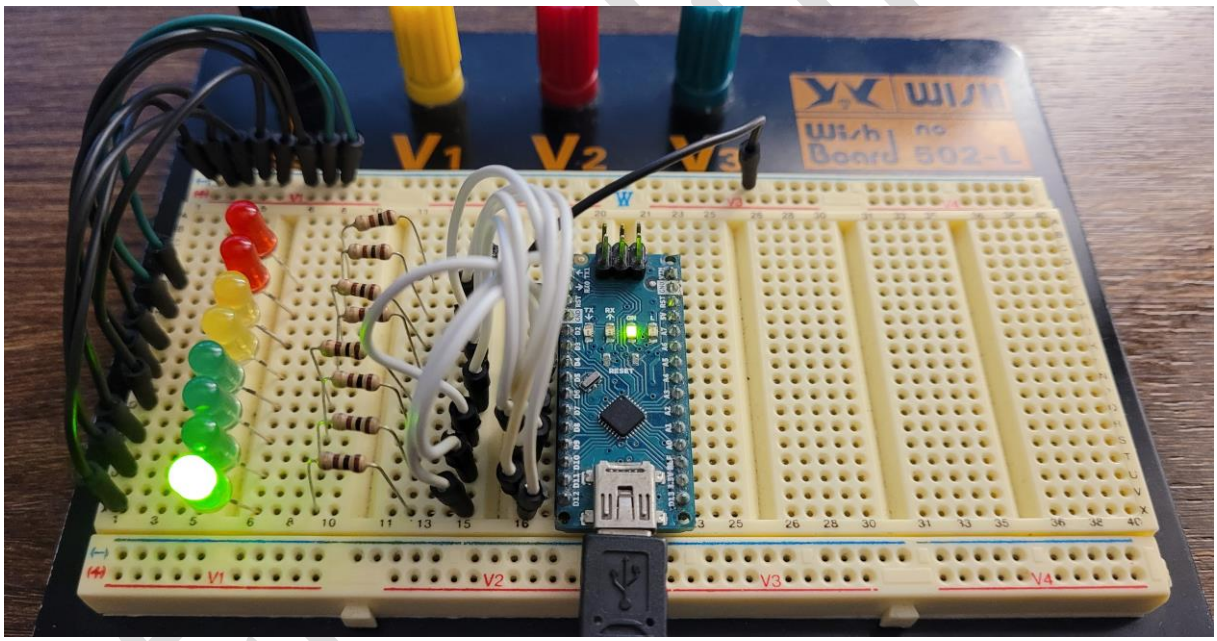

Fig. 20 – Montaj experimental pentru aplicația 3

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;

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

APLICAȚIA 4:

Se va implementa circuitul conform următoarei scheme (Fig. 21):

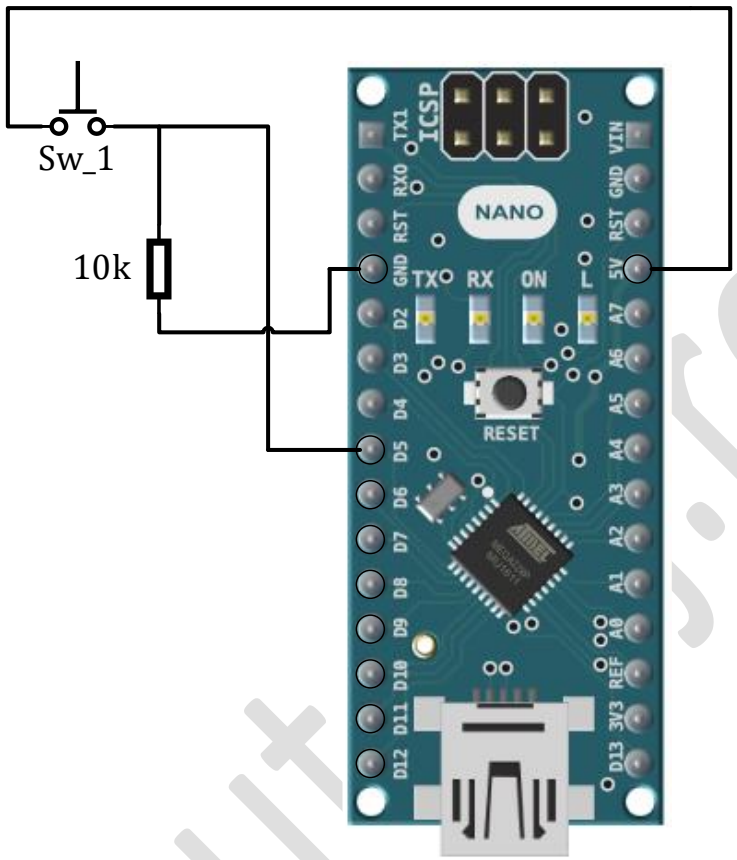

Fig. 21 – Schema electronică pentru implementarea circuitului specific aplicației 4 [7]

Se va implementa următorul cod program:

```
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);
}
```


Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

Pentru a verifica funcționalitatea aplicației nr. 4, se va deschide consola Serial din meniul „Tools” opțiunea „Serial Monitor” (Fig. 22).

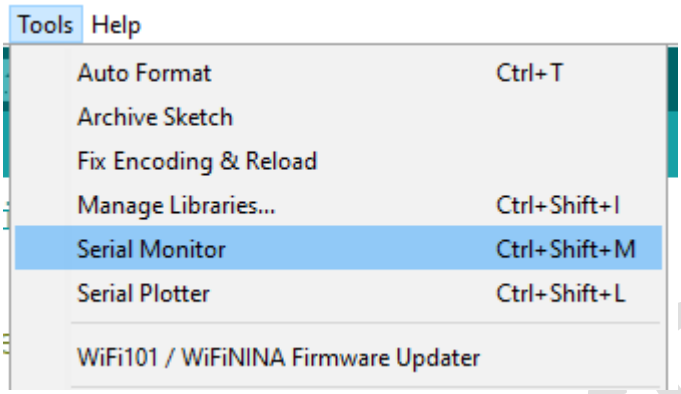

Fig. 22 – Meniul „Tools” opțiunea „Serial Monitor”

La apăsarea butonului din cadrul montajului, în consola Serial se va actualiza valoarea variabilei (din „0” în „1”) care stochează starea logică a terminalului la care este atașat butonul (Fig. 23).

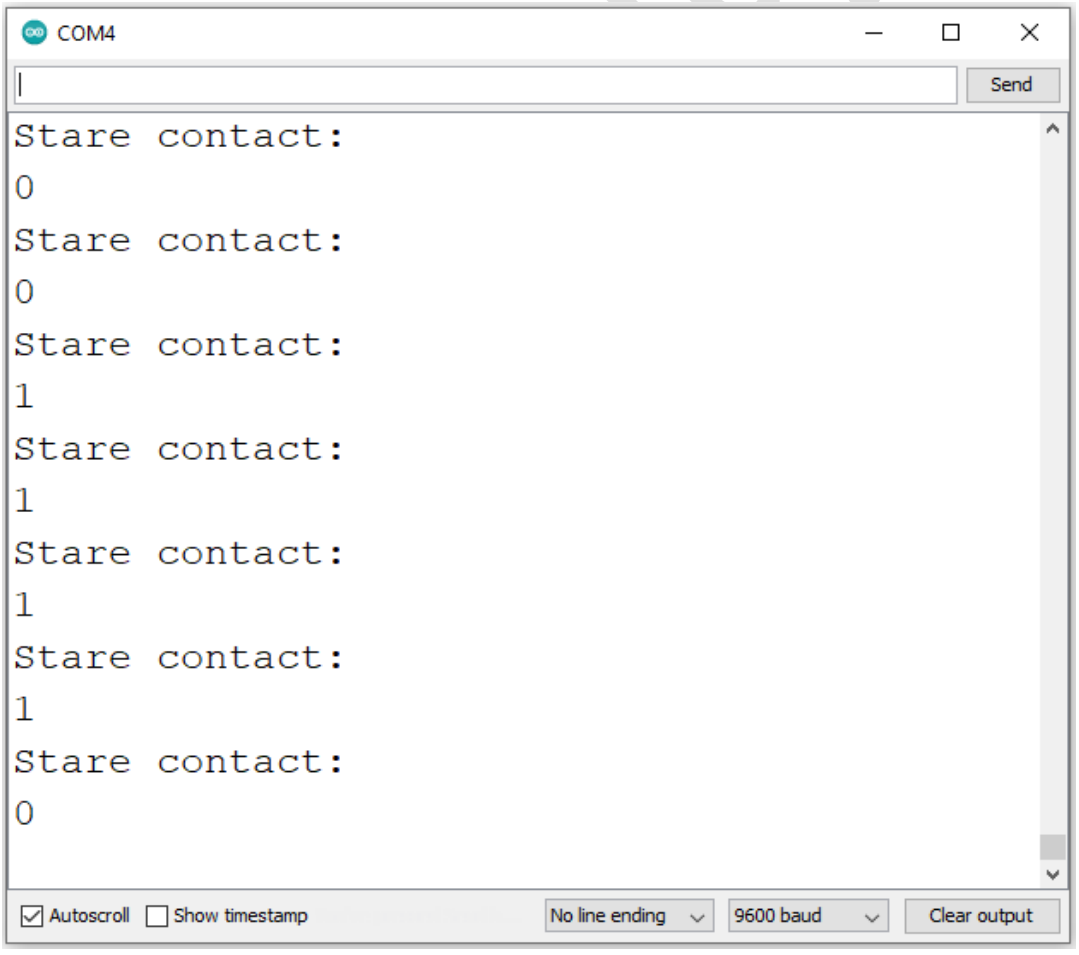

Fig. 23 – Consola Serial afișând mesajul implementat în cadrul aplicației 4

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

Pentru aceeaşi aplicaţie, din meniul „Tools” se va deschide şi afişajul grafic „Serial Plotter” pentru a vizualiza evoluţia stării digitale a terminalului „D5” (Fig. 24).

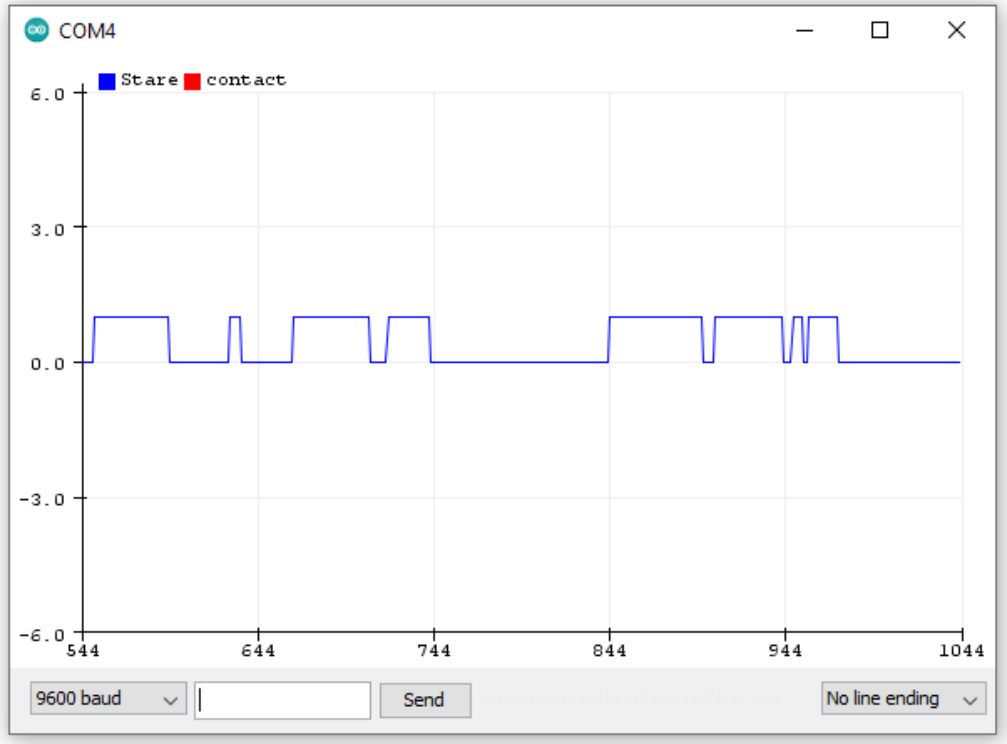

Fig. 24 – Fereastra de afişare grafică a valorilor vehiculate pe magistrala Serial

Montajul experimental se va realiza conform (Fig. 25):

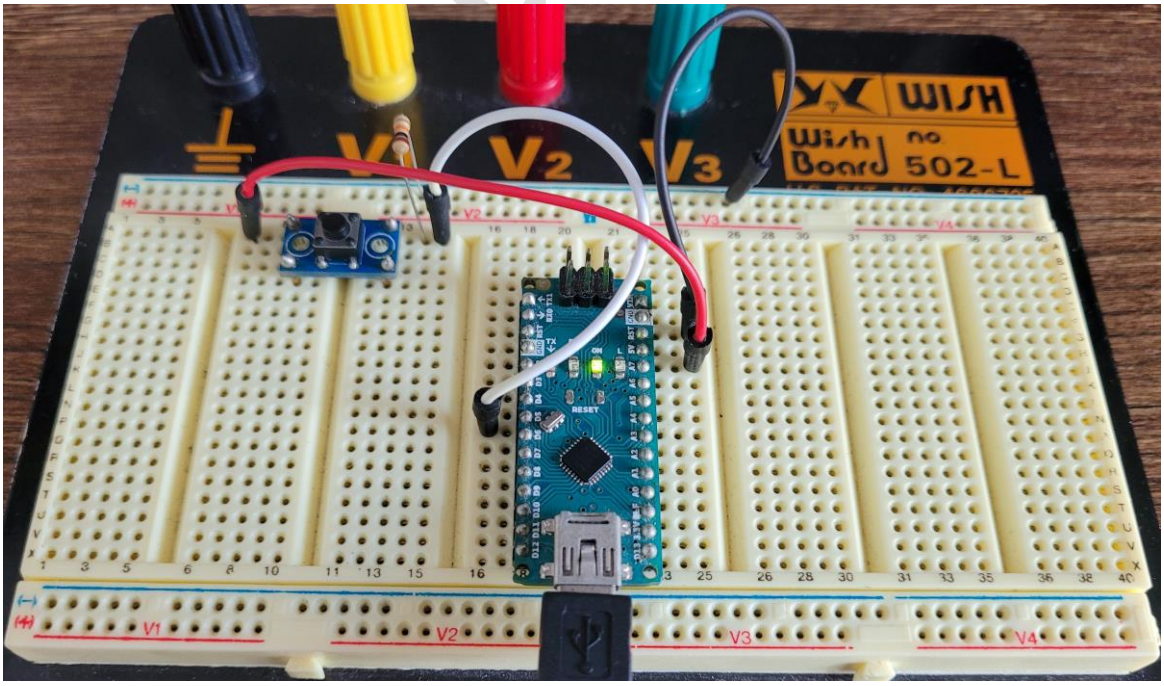

Fig. 25 – Montaj experimental pentru aplicaţia 4

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

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];

APLICAȚIA 5:

Se va implementa circuitul conform următoarei scheme (Fig. 26):

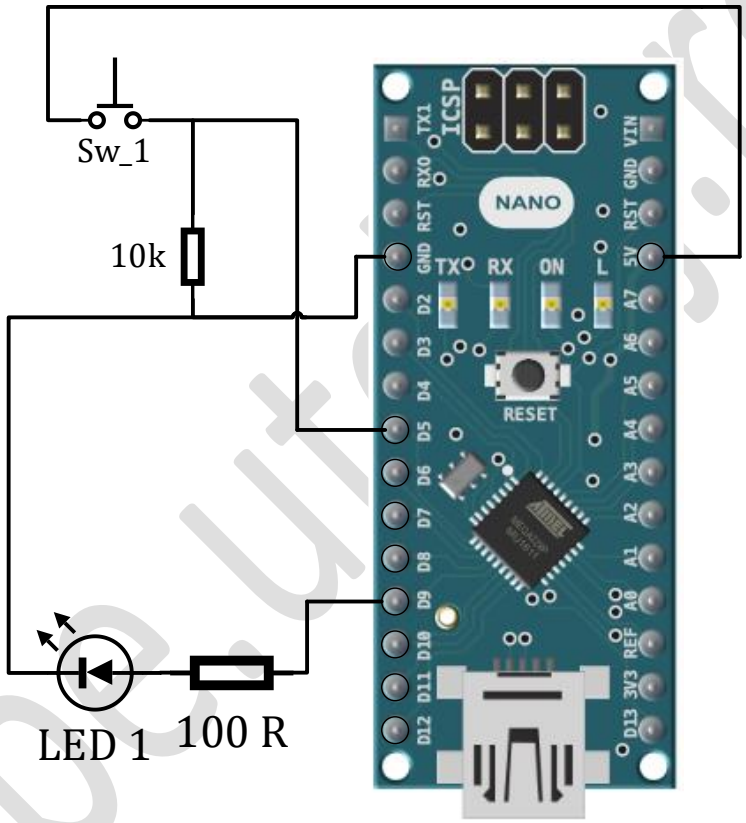

Fig. 26 – Schema electronică pentru implementarea circuitului specific aplicației 5 [7]

Se va implementa următorul cod program:

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

void setup() {
  pinMode(pin_sw, INPUT);
  pinMode(pin_led, OUTPUT);
}
```


Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

```
void loop() {
  sw_state = digitalRead(pin_sw);
  digitalWrite(pin_led, sw_state);
}
```

Montajul experimental se va realiza conform (Fig. 27):

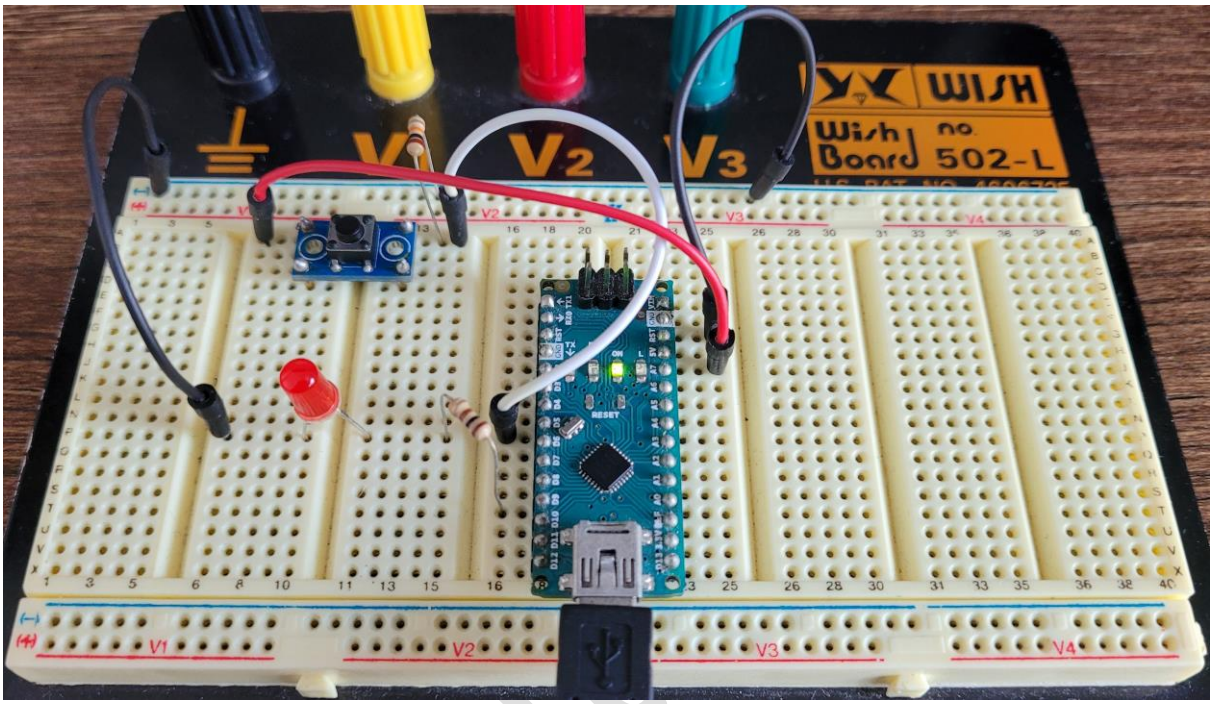

Fig. 27 – Montaj experimental pentru aplicația 5

Implementarea aplicației nr. 5 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);

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

APLICAȚIA 6:

Se va implementa circuitul conform următoarei scheme (Fig. 28):

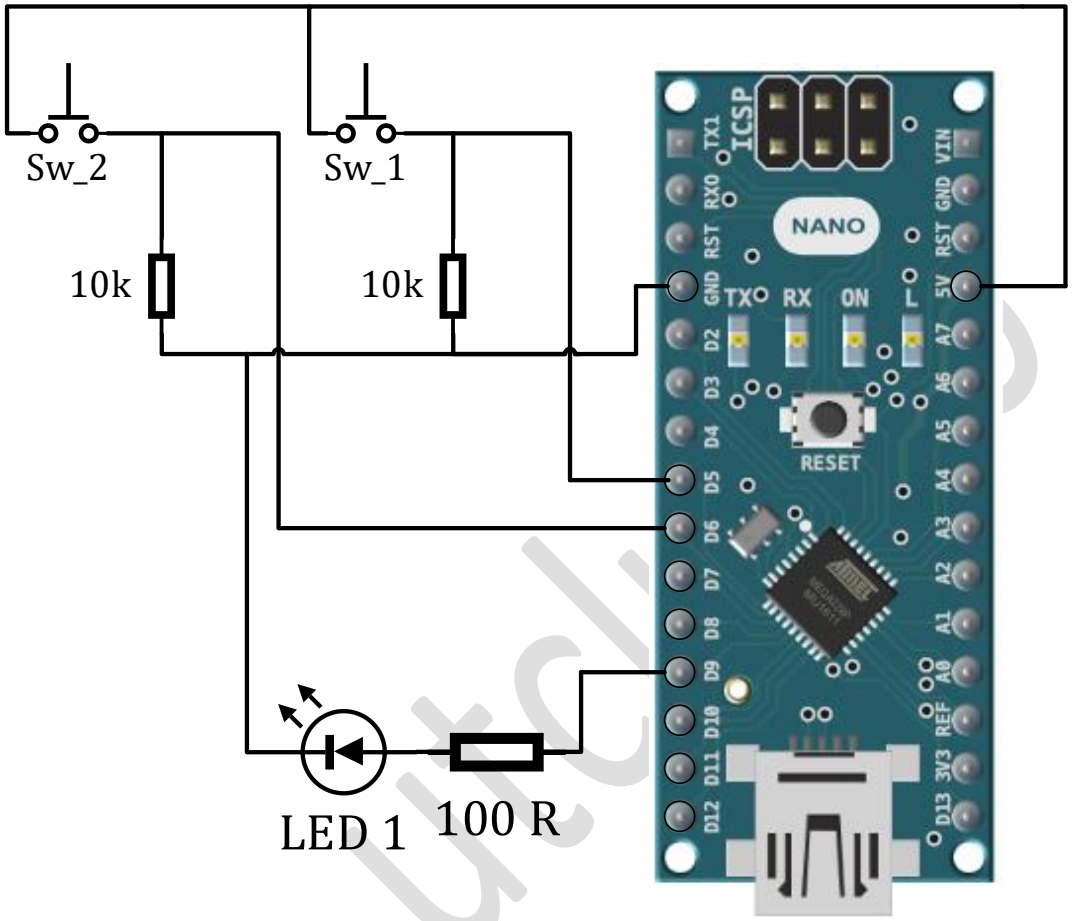

Fig. 28 – Schema electronică pentru implementarea circuitului specific aplicației 6 [7]

Se va implementa următorul cod program:

```
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);
}
```


Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

```
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);
}
```

Montajul experimental se va realiza conform (Fig. 29):

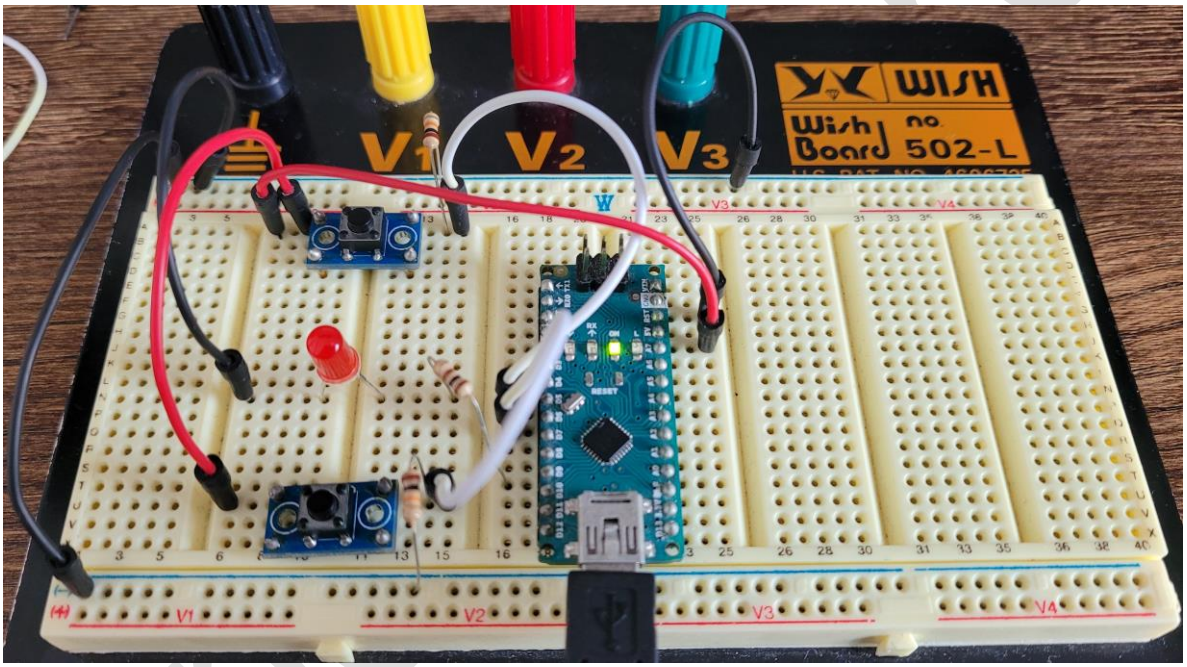

Fig. 29 – Montaj experimental pentru aplicația 6

Implementarea aplicației nr. 6 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";

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro

V. CONCLUZIE:

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.

VI. 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/

Conf. Dr. Ing. Ioana – Cornelia Gros – Ioana.Gros@emd.utcluj.ro
Asist. Dr. Ing. Lucian – Nicolae Pintilie – Lucian.Pintilie@emd.utcluj.ro