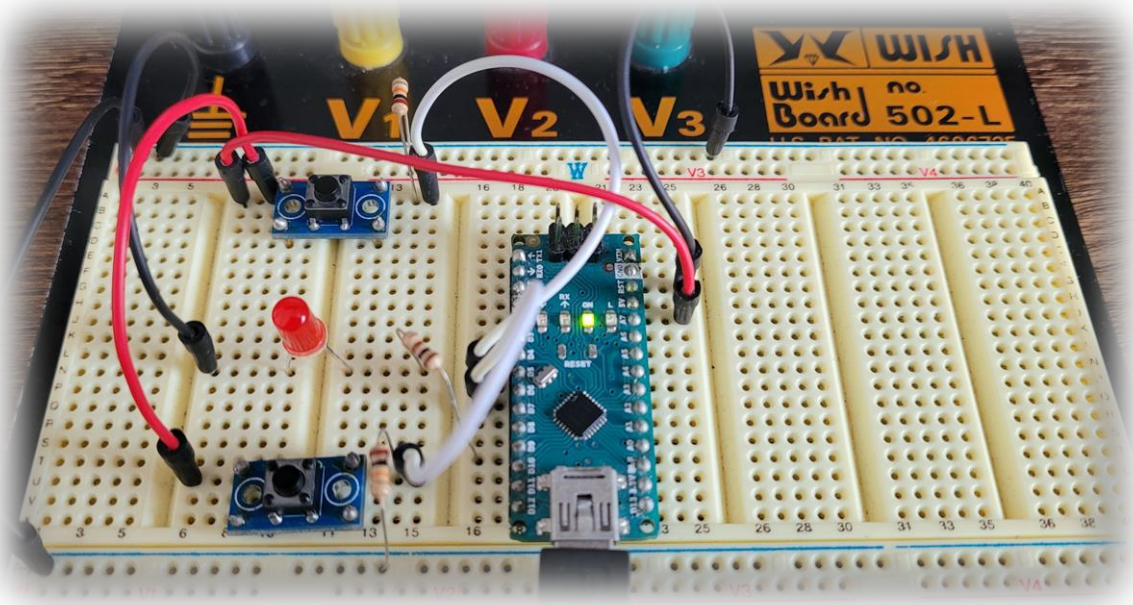
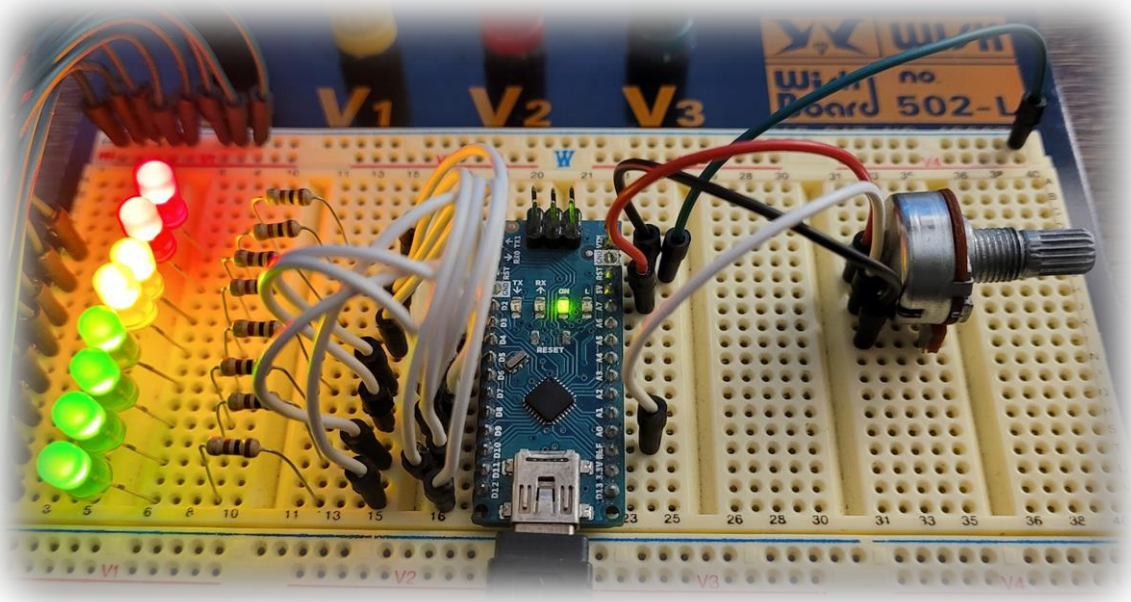
APLICAȚII CU SISTEME MICROPROGRAMABILE ÎN INGINERIE ELECTRICĂ

Ghid de aplicații

Lucian Nicolae PINTILIE, Ioana-Cornelia GROS, Adrian Mihai IUORAȘ, Vasile Mihai SUCIU, Teodor Crișan PANĂ





UTPRESS
Cluj-Napoca, 2024
ISBN 978-606-737-722-4

Lucian Nicolae PINTILIE, Ioana-Cornelia GROS, Adrian Mihai IUORAȘ,

Vasile Mihai SUCIU, Teodor Crișan PANĂ

APLICAȚII CU SISTEME MICROPROGRAMABILE ÎN INGINERIE ELECTRICĂ

Ghid de aplicații

U.T.PRESS
Cluj - Napoca, 2024
ISBN 978-606-737-722-4

Editura U.T.PRESS
Str. Observatorului nr. 34
400775 Cluj-Napoca
Tel.: 0264-401.999
e-mail: utpress@biblio.utcluj.ro
http://biblioteca.utcluj.ro/editura

Recenzia: Conf.dr.ing. Simina Emerich
Conf.dr.ing. Mircea Bojan

Pregătire format electronic on-line: Gabriela Groza

ISBN 978-606-737-722-4

PREFAȚĂ

Integrarea sistemelor microprogramabile în instalaţii complexe şi aplicaţii specifice de comandă şi control are o istorie solidă în domeniul ingineriei electrice, încă de la apariţia primelor microcontrolere şi microprocesoare programate în limbaj de asamblare. Ceea ce s-a schimbat în ultimii ani este reprezentat de resursele tehnologice în continuă expansiune, pe diverse niveluri de complexitate, de accesibilitatea acestora atât amatorilor cât şi celor care îşi dezvoltă o carieră în domeniu şi de dezvoltarea unor perspective noi în folosirea acestor resurse, legate de contextul apariţiei Internet of Things (IoT), Cloud - Computing şi Industry 4.0.

Într-o accepţiune mai largă, conceptul de sisteme microprogramabile (cu varianta de largă accepţiune din limba engleză – „embedded systems") cuprinde toate soluţiile arhitecturale de sisteme cu elemente de procesare numerică ce pot fi integrate fizic în sisteme care le pot folosi ca şi „creiere" pentru executarea unor sarcini impuse prin algoritmi de către utilizatori.

Materialul de faţă se adresează în primul rând studenţilor din cadrul Facultăţii de Inginerie Electrică, Universitatea Tehnică din Cluj-Napoca, cât şi celor de la Extensia Bistriţa, ca suport pentru aplicaţiile realizate în cadrul disciplinei Sisteme cu microprocesoare. Implementările conţin o abordare teoretică condensată, structurată pe şase teme de studiu, care urmează un fir logic al înţelegerii pas cu pas a conceptelor şi componentelor de bază, urmate de implementări hardware şi software detaliate pe o varianta populară de microcontroler – Arduino Nano. Totodată, autorii invită orice cititor interesat de înţelegerea ghidată şi implementarea pas cu pas a unor proiecte folosind această platformă de dezvoltare să folosească acest material, asigurând-i că vor câştiga o aprofundare corectă şi solidă a materialului prezentat, atât prin abordarea sistematică propusă, cât şi prin proiectele demonstrative propuse.

În cadrul prezentului ghid de aplicaţii tematica este structurată sub forma a şase capitole principale sau teme de studiu:

1. Noţiuni introductive;
2. Manipularea intrărilor şi ieşirilor digitale;
3. Manipularea intrărilor analogice;
4. Aplicaţii generale de interfaţare;
5. Protocoale de comunicaţie;
6. Mediul Matlab – Simulink;

În cadrul primei teme de studiu, au fost prezentate principalele aspecte legate de logica binară, baze de numeraţie, metode de reprezentare şi manipulare a datelor numerice precum şi diverse noţiuni introductive legate de arhitectura microcontrolerului ATMega 328P şi diverse mijloace fizice (eng. hardware) de programare. Tot în cadrul primei teme au fost propuse trei aplicaţii demonstrative pe baza instrucţiunilor în limbaj de asamblare specifice arhitecturii interne a microcontrolerului ATMega 328P:

1. Manipularea directă a regiştrilor „DDRD" şi „PORTD" - aplicaţie specifică studierii circuitelor logice combinaţionale (statice);
2. Manipularea indirectă a regiştrilor „DDRD" şi „PORTD" pe baza operatorilor binari - aplicaţie specifică studierii circuitelor logice secvenţiale (dinamice);

3. Manipularea indirectă a regiştrilor „DDRD” şi „PORTD” pe baza structurilor secvenţiale iterative (eng. „for ()”) - aplicaţie specifică instrucţiunilor iterative (numărător binar);

A doua temă de studiu a avut ca scop prezentarea generală a principalelor metode şi limbaje de programare specifice microcontrolerului ATMega 328P. Tot în cadrul acestui capitol a fost subliniată şi noţiunea de platformă de dezvoltare (platforma Arduino NANO). Începând cu această temă, a fost stabilit de asemenea ca şi limbaj de programare implicit limbajul C / C++ având în plus setul de instrucţiuni „Wiring” specific domeniului sistemelor microprogramabile (eng. embedded systems). În cadrul acestei teme a fost descrisă şi procedura de manipulare a intrărilor şi ieşirilor digitale pe baza funcţiilor „digitalRead ()”, „digitalWrite ()” şi „analogWrite()”. În acest sens, au fost propuse şase aplicaţii spre implementare:

1. Semnalizare intermitentă continuă cu două diode electroluminiscente (eng. LED);
2. Semnalizare intermitentă variabilă cu două diode electroluminiscente;
3. Semnalizare intermitentă continuă cu opt diode electroluminiscente;
4. Preluarea stării digitale de la un întreruptor;
5. Redirecţionarea semnalului digital de la o intrare digitală înspre o ieşire digitală;
6. Auto-menţinerea stării digitale la acţionarea unui buton cu apăsare şi revenire;

A treia temă de studiu a cuprins noţiunile introductive specifice procesării semnalelor analogice pe baza sistemelor microprogramabile. A fost prezentată noţiunea de convertor analog – digital (sau analog – numeric – eng. ADC – Analog to Digital Converter) şi principiul de funcţionare. A fost descris de asemenea modul de funcţionare al etajului comparator din cadrul arhitecturii microcontrolerului ATMega 328P. A fost prezentată de asemenea procedura de calibrare a unui senzor analogic. Totodată în cadrul acestei teme de studiu au fost prezentate principalele funcţii specifice procedurii de manipulare a convertorului analog – digital precum „analogRead ()”. Au fost propuse de asemenea şase teme de studiu:

1. Determinarea rezoluţiei convertorului analog – digital (cu afişare în monitorul serial);
2. Determinarea preciziei convertorului analog – digital şi a valorii de tensiune măsurată;
3. Determinarea temperaturii cu ajutorul traductorului LM-35;
4. Implementarea unui comparator numeric cu prag digital reglabil (funcţia „if”);
5. Implementarea unei coloane luminoase indicatoare de nivel (funcţia „map”);
6. Generarea unui semnal dreptunghiular modulat în lăţime (funcţia „map”);

A patra temă de studiu a rezumat toate noţiunile introductive studiate în capitolele anterioare într-o serie de şase aplicaţii generale de interfaţare. A fost prezentat conceptul de interfaţare. Au fost prezentate şi principalele tipuri de aplicaţii cu microcontrolere:

- aplicaţii independente;
- aplicaţii dependente de elementele de interfaţare;
- aplicaţii hibride;

Au fost prezentate de asemenea o serie de metode şi proceduri de optimizare:

- introducerea unui timp de întârziere;
- introducerea unor constrângeri;
- introducerea unui algoritm de mediere în timp a semnalului;
- algoritmul de ştergere automată a caracterelor remanent de pe ecran;

Primele trei aplicaţii au avut ca şi scop validarea procedurilor de optimizare. În final, pe baza noţiunilor generale însuşite în capitolele anterioare şi a algoritmilor de optimizare

s-a procedat la implementarea a două aplicaţii concrete precum „termometru digital” şi „riglă digitală”. Prin urmare, au fost propuse spre implementare următoarele aplicaţii:
1. Controlul digital al factorului de umplere pentru un semnal modulat în lăţime;
2. Preluarea temperaturii de la un traductor şi filtrarea zgomotelor;
3. Măsurarea distanţei cu ajutorul traductorului ultrasonic HC – SR04;
4. Utilizarea unui afişaj de tip LCD;
5. Implementarea unui termometru digital;
6. Implementarea unei rigle digitale;

A cincea temă a avut ca scop prezentarea principalelor protocoale de comunicaţie între sistemele microprogramabile. A fost evidenţiat conceptul de protocol „paralel” sau „serial”. Au fost prezentate conceptele de protocoale seriale asincrone şi sincrone. A fost prezentat conceptul de protocol „punct la punct” şi „master – slave”. A fost prezentat de asemenea modulul Bluetooth – Serial HC-05 şi traductorul digital de umiditate şi temperatură DHT-11. A fost prezentat de asemenea conceptul de aplicaţie multipunct şi variabile comune sau partajate (eng. shared variables). În acest sens, au fost propuse şase aplicaţii spre implementare:
1. Recepţionarea datelor prin interfaţa serial de la microcontroler;
2. Transmiterea datelor prin intermediul interfeţei serial la microcontroler;
3. Recepţionarea datelor prin intermediul modulului Bluetooth HC-05;
4. Transmiterea datelor prin intermediul modulului Bluetooth HC-05;
5. Preluarea datelor de la traductorul de umiditate şi temperatură DHT-11;
6. Implementarea unei aplicaţii de monitorizare multipunct a parametrilor ambientali;

A şasea temă a avut ca şi scop prezentarea modului de lucru a microcontrolerului în mediul grafic de simulare testare şi programare MathWorks Matlab – Simulink. Pe baza pachetului de instrumente Arduino IO a fost realizată legătura între platforma de dezvoltare Arduino şi mediul Matlab – Simulink. A fost prezentat conceptul de „Rapid Control Prototyping” şi „Computer Aided Engineering”. Toate aceste proceduri pot fi realizate în scopul testării şi validării automate a unui echipament aflat în faza de producţie. Mediul grafic de simulare şi testare Matlab – Simulink prezintă o largă varietate de instrumente şi facilităţi în ceea ce priveşte procesarea numerică a datelor recepţionate de la traductoare. În acest sens, au fost propuse trei aplicaţii în vederea implementării:
1. Semnalizare alternativă intermitentă cu două diode electroluminiscente (eng. LED);
2. Achiziţionarea unui semnal digital;
3. Măsurarea temperaturii şi declanşarea unei ieşiri digitale la depăşirea pragului impus;

În cele din urmă, autorii doresc să aducă mulţumiri şi deosebit aprecieri tuturor colegilor din colectivul de Electronică de Putere şi Acţionări Electrice care au ajutat la întocmirea prezentului ghid de aplicaţii.

Cluj – Napoca, septembrie 2024 Autorii

Cuprins

Tema de studiu nr. 1 – Noțiuni introductive

I. SCOPUL TEMEI

- familiarizarea cu noţiunile de bază introductive în domeniul programării şi al operaţiilor logice binare;
- familiarizarea cu noţiunile legate de tehnologiile de integrare a dispozitivelor de calcul specializate;
- însuşirea noţiunilor de sistem de calcul şi a conceptelor fundamentale legate de sistemele încorporate (eng. Embedded Systems);
- studiul şi analiza funcţională a arhitecturii microcontrolerului ATMega 328P, împreună cu metodele şi procedurile de programare specifice [1].

II. INTRODUCERE

În cadrul aplicaţiilor specifice domeniilor inginereşti, cu precădere cel al ingineriei electrice şi electronice, este necesară de cele mai multe ori proiectarea unor sisteme de comandă şi control care au rolul de a deservi anumite procese tehnologice. Prin acest fapt a fost facilitată de utilizarea, pe scară largă a sistemelor microprogramabile de tip **microcontroler** [2]. Microcontrolerul există în echipamente precum:
- aparatura electrocasnică şi dispozitive sau bunuri de larg consum (ex. imprimanta);
- sistemele de monitorizare şi automatizare (ex. termostatul);
- aparatura specializată în domeniul de măsurare (ex. termometrul medicinal);
- aparatura industrială (ex. maşina pentru înfiletat);

În plus faţă de microcontroler se regăsesc şi alte tipuri de sisteme microprogramabile precum:
- sistemele monobloc cu microprocesoare dedicate **SoC** (eng. System On a Chip);
- procesoarele digitale de semnal **DSP** (eng. Digital Signal Processor);
- ariile de porţi programabile **FPGA** (eng. Field Programmable Gate Array);

În literatura de specialitate toate aceste dispozitive microprogramabile pot fi regăsite sub denumirea de „microsisteme integrate sau încorporate” (eng. embedded systems). Integrarea sau încorporarea dispozitivului de procesare central într-o aplicaţie conduce la obţinerea unui echipament sau produs finit. În general, integrarea dispozitivului central de calcul se realizează pe baza unui cablaj imprimat care la rândul său interconectează o serie de **periferice** precum:
- componente electronice pentru semnalizarea unor situaţii (ex. indicatori de tip LED sau difuzor);
- componente pentru preluarea comenzilor externe (ex. butoane sau senzori);
- dispozitive pentru comunicare cu alte echipamente (ex. adaptoare Bluetooth – Serial);

În cadrul proceselor de fabricare a echipamentelor electronice se pot distinge două tehnologii de integrare a dispozitivului central de procesare:
- tehnologia (eng.) „On board” (integrarea pe un cablaj imprimat);
- tehnologia (eng.) „System On a Chip” (integrarea monobloc într-o singură capsulă);

Microcontrolerul reprezintă o soluţie de integrare de tip **SoC** a tuturor perifericelor necesare funcţionării (ex. memorie RAM, ROM, Flash, procesor etc.).

În practică există mai multe modele şi variante constructive, de la diverşi producători, ale microcontrolerelor precum:
- MicroChip PIC 10F200, [3];
- Infineon XMC1400, [4];
- STMicroelectronics STM32 F205, [5];
- Atmel, AVR, MicroChip ATMega 328P;

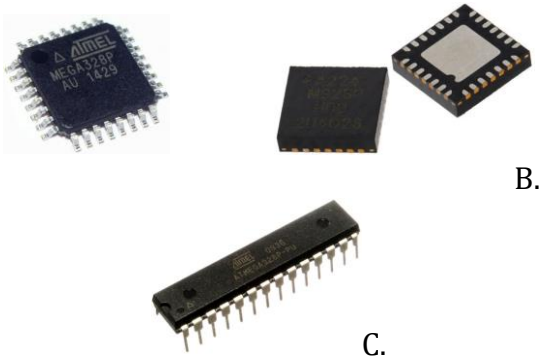
Platformele de dezvoltare construite în jurul unui dispozitiv central de procesare reprezintă o soluţie de integrare **On Board** a tuturor perifericelor opţionale dar **nu neapărat necesare sau vitale** funcţionării sistemului de calcul.

În practică, există mai multe modele şi variante constructive ale platformelor de dezvoltare, de la diverşi producători, precum:
- MicroChip Explorer 16/32 Development Board [6];
- Infineon AURIX TC275 ARDSBTOBO1 [7];
- STMicroelectronics NUCLEO - L010RB [8];
- Arduino NANO, MEGA, UNO, DUE, Leonardo etc [9].

III. INTEGRAREA MICROCONTROLERULUI ATMEGA 328P ÎNTR-O APLICAŢIE

În vederea evidenţierii conceptelor teoretice fundamentale specifice funcţionării sistemelor microprogramabile (cu concentrare pe microcontrolere), va fi analizată arhitectura microcontrolerului ATMega 328P.

Conform catalogului dispozitivului electronic ATMega 328P, acesta poate fi regăsit în trei variante constructive, anume, TQFP, MLF, PDIP-28 (Fig. 1.1).



A. B. C.

Fig. 1.1 – Microcontrolerul ATMega 328P (A. TQFP, B. MLF, C. PDIP-28)

Arhitectura internă a microcontrolerului ATMega 328P (Fig. 1.2), prezintă o serie de etaje constructive precum:
- unitate centrală de procesare cu frecvenţa maximă de lucru la 16 [MHz];
- memorie EEPROM având capacitatea de 1 [KB] şi 100000 de cicluri de inscripţionare;
- memorie SRAM având capacitatea de 2 [KB];
- memorie FLASH având capacitatea de 32 [KB] şi 10000 de cicluri de inscripţionare
- numărător sau temporizator având rezoluţia pe 8 şi 16 biţi;
- interfeţe de comunicare UART / USART, I^2C, TWI, SPI, debugWIRE;
- registre de intrare şi ieşire de uz general (eng. General Purpose Input or Output – GPIO);
- convertoare analog – digitale (eng. Analog to Digital Converter – ADC);

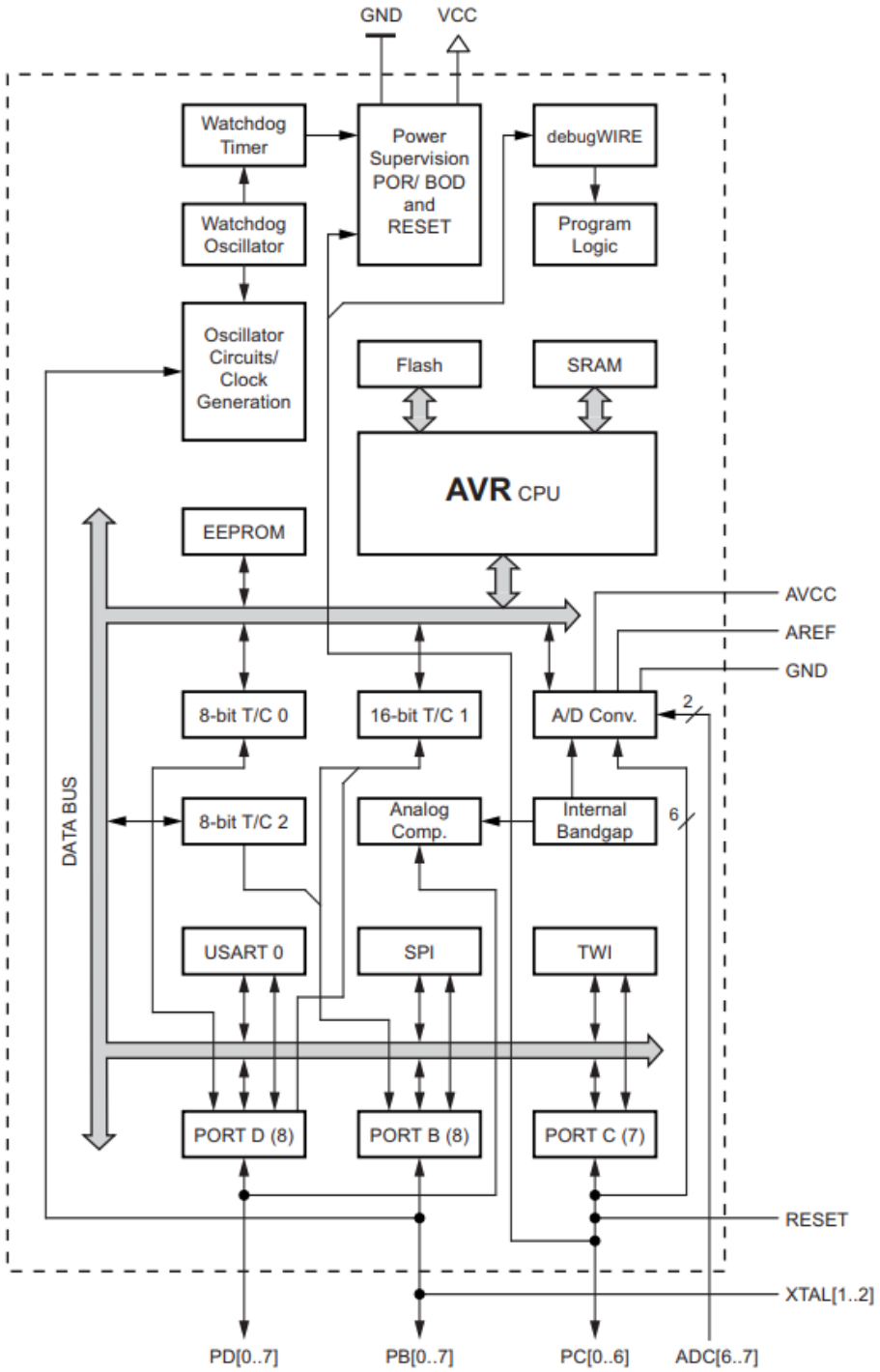


Fig. 1.2 – Arhitectura internă a microcontrolerului ATMega 328p (schema bloc)

Tot în cadrul datelor de catalog, se regăsește diagrama de distribuție a terminalelor (eng. pinout diagram) reprezentând funcțiile posibile ale acestora (Fig. 1.3) [10].

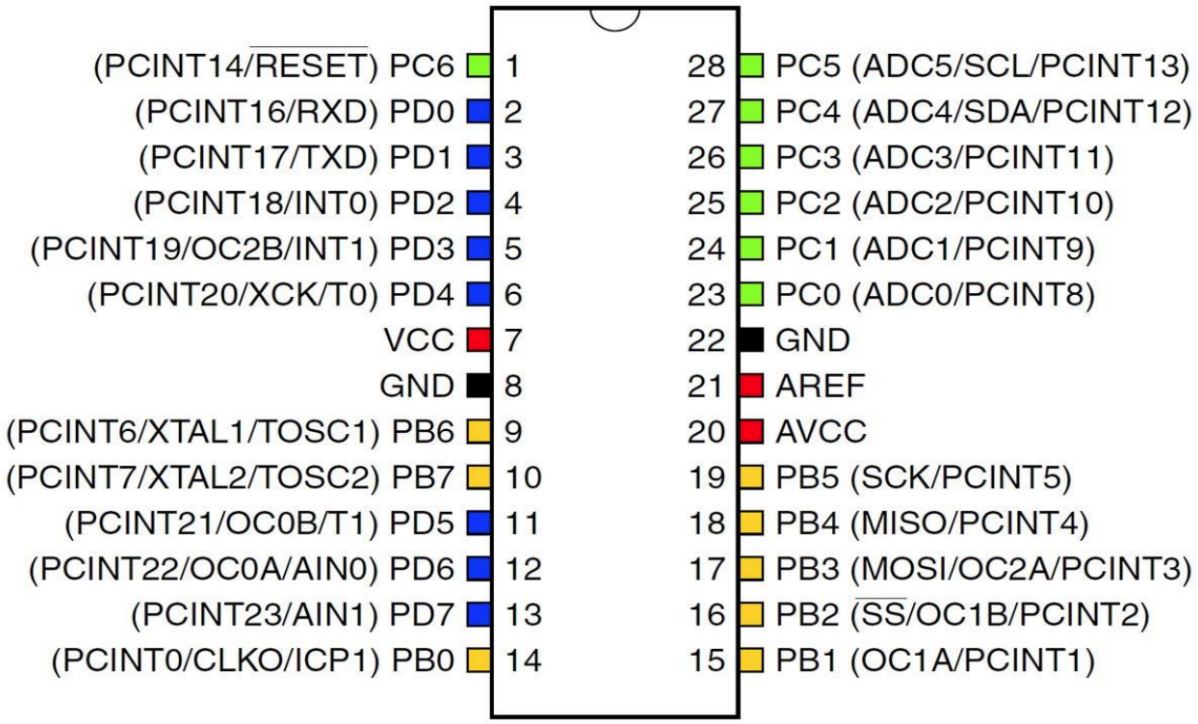


Fig. 1.3 – Diagrama de distribuţie a terminalelor
şi funcţiile alocate acestora la nivel de microcontroler

Pentru a integra microcontrolerul ATMega 328P într-o anumită aplicaţie, este necesară introducerea a cel puţin trei etaje de circuit fundamentale (Fig. 1.4) [11]:
- etajul de alimentare pe bază de stabilizator liniar de tensiune (marcaj verde);
- etajul oscilatorului cu cuarţ (marcaj portocaliu);
- etajul pentru tratarea rutinei de reiniţializare (eng. RESET), (marcaj albastru);

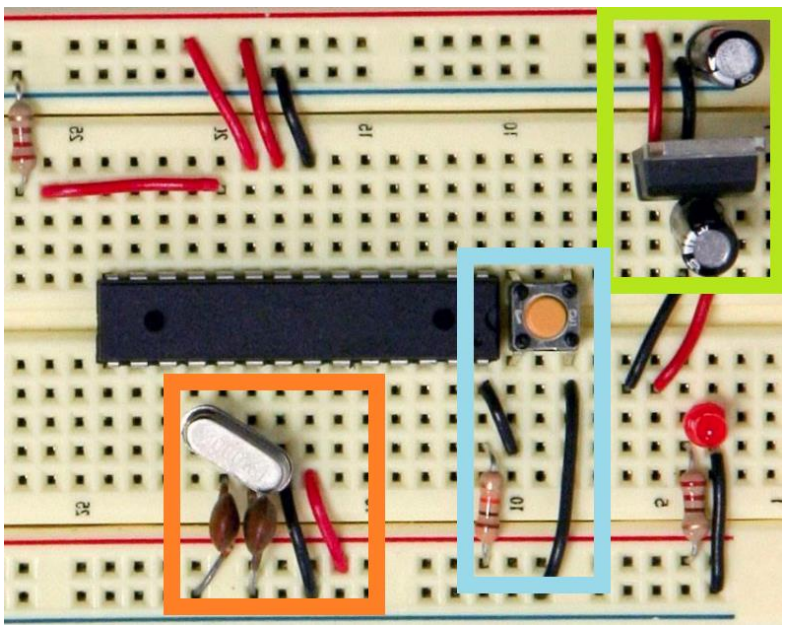


Fig. 1.4 – Integrarea microcontrolerului ATMega 328P într-o aplicaţie

În vederea transferării inițiale a informației numerice în memoria microcontrolerului și în vederea programării acestuia, dispozitivul programator (ex. SparkFun USB Tiny AVR Prog) se va atașa la terminalele „VCC”, „AREF”, „AVCC”, „GND”, „SCK”, „MOSI”, „MISO” și „RESET” ale microcontrolerului (Fig. 1.5) [12]. Procedura utilizată în această situație, reprezintă un protocol de comunicare direct (între calculatorul gazdă și microcontroler) prin interfață USB – SPI (eng. Serial to Peripheral Interface).

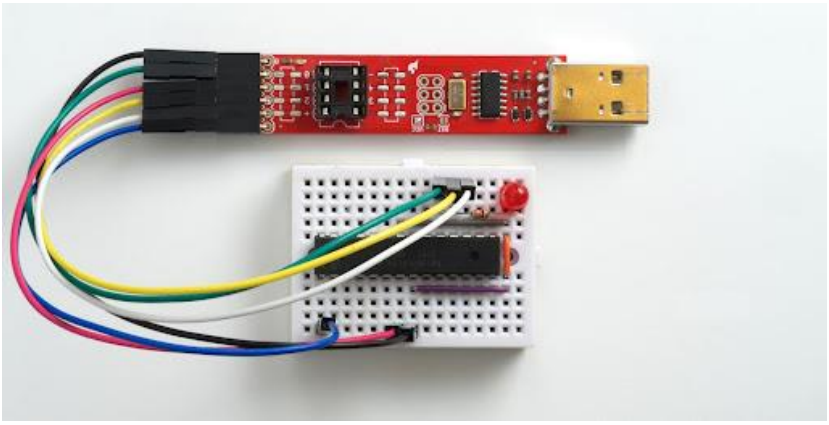


Fig. 1.5 – Atașarea dispozitivului programator USB – SPI la ATMega 328P

Există, de asemenea, posibilitatea de a programa microcontrolerul ATMega 328P și prin intermediul unui dispozitiv adaptor de tip USB – TTL – Serial (ex. FTDI FT232RL) (Fig. 1.6) [13]. Această procedură de programare și comunicare cu calculatorul gazdă, spre deosebire de metoda expusă anterior, necesită transferarea inițială în memoria FLASH a microcontrolerului a secvenței de cod pentru inițializarea a comunicației Serial dintre microcontroler și calculatorul gazdă. În literatura de specialitate această secvență de cod care inițializează comunicația Serial în vederea programării, poartă denumirea de „Bootloader”. Aceasta trebuie inscripționată inițial cu ajutorul unui dispozitiv programator de tip USB – SPI.

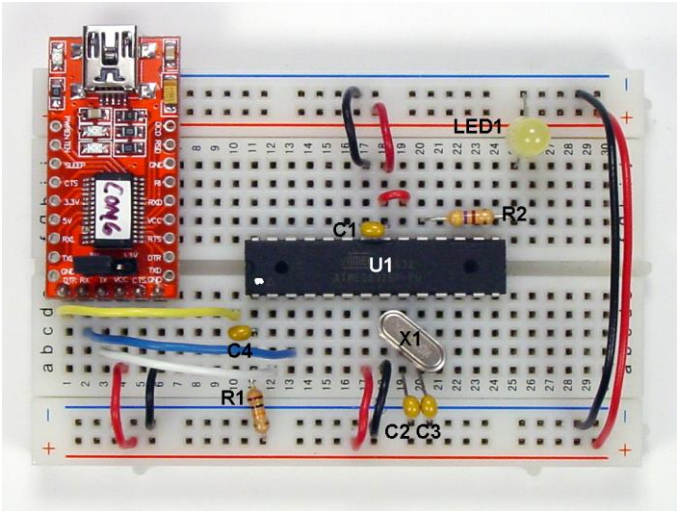


Fig. 1.6 – Atașarea dispozitivului programator USB – Serial la ATMega 328P

IV. METODE DE PROGRAMARE A MICROCONTROLERULUI

Există patru metode de programare a unui sistem de calcul, anume:
- programarea în cod maşină;
- programare în limbaj de asamblare;
- programare în sintaxă;
- generare automată de cod pe baza mediilor grafice de programare;

Trecerea de la un nivel superior al limbajului de programare precum mediul grafic, la un nivel inferior cum este codul maşină sau limbajul de asamblare, în literatura de specialitate poartă denumirea de „proces de compilare" (eng. compilation). Practic, procesul de compilare reprezintă „translatarea" codului program de la un nivel inteligibil al factorului uman înspre un nivel compatibil cu sistemul de calcul dat (Fig. 1.7), [14].

Corespondenţa dintre elementele de program regăsite în limbajele de programare se realizează prin intermediul funcţiilor şi a bibliotecilor de funcţii (eng. function libraries). Există biblioteci de funcţii care asigură trecerea de la mediul grafic la sintaxă, de la sintaxă la limbaj de asamblare, de la limbajul de asamblare la codul specific arhitecturii sistemului de calcul, anume instrucţiuni în cod binar sau hexazecimal.

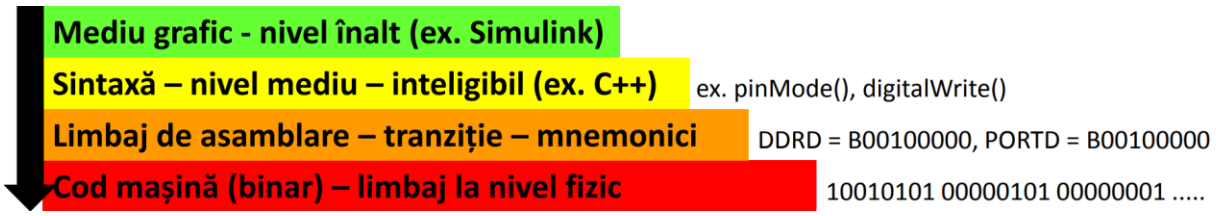


Fig. 1.7 – Metode şi limbaje de programare specifice lucrului cu microcontrolere
Ilustrarea procesului de compilare

Există de asemenea, posibilitatea de a manipula în mod direct conţinutul registrelor de memorie prin aşa zisa **metodă hibridă de programare în sintaxă şi limbaj de asamblare**, denumită în literatura de specialitate Mixed assembly – syntax (eng). Memoria microcontrolerului ATMega 328P este organizată tabelar în registre a câte opt unităţi fundamentale, anume opt biţi, sau un octet (byte).

Registrul reprezintă o zonă dedicată din memorie pentru a îndeplini o anumită funcţie în cadrul microcontrolerului. Un exemplu de astfel de registru, poate fi „SREG" (Fig. 8), registrul de stare.

SREG – AVR Status Register

The AVR status register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fig. 1.8 – Structura registrului de stare „SREG" al microcontrolerului ATMega 328P

Rolul registrului de stare SREG este de a consemna situaţiile prevăzute sau neprevăzute care au loc în timpul efectuării unor operaţii aritmetice sau logice. În cadrul registrului de stare, există opt poziţii binare, care descriu funcţionarea unităţii centrale de procesare (CPU) în operaţiile aritmetice sau logice pe care le executa şi semnalizează apariţia unor evenimente speciale. Fiecare poziţie din şirul de biţi componenţi ai registrului SREG are o anumită semnificaţie.

Există de asemenea, la nivelul memoriei microcontrolerului ATMega 328P, o zonă specială, rezervată terminalelor specifice intrărilor şi ieşirilor digitale. Prin intermediul intrărilor şi ieşirilor de uz general (GPIO) microcontrolerul poate furniza sau accepta din exterior un nivel de tensiune cuprins între zero şi valoarea nivelului tensiunii de alimentare. Terminalele care au funcţia de intrare şi ieşire digitală, sunt grupate câte opt la nivelul carcasei microcontrolerului. Grupările a câte opt terminale se numesc „PORTx" (unde „x" reprezintă litera alocată în ordine crescătoare a grupărilor). Pentru exemplificare, se va alege gruparea de terminale „PORTD" (corespunzătoare portului D al microcontroler-ului ATmega 328P).

Registrul care controlează modul de lucru al unei grupări de tip PORTx, este denumit „DDRD" (eng. Data Direction Register of PORT D) (Fig. 1.9).

Dacă valoarea tuturor poziţiilor de memorie din registrul „DDRB" va fi zero, atunci toate terminalele de pe carcasa microcontrolerului care aparţin grupului „PORTD" vor avea rolul de intrări digitale, şi vor prelua o diferenţă de potenţial. În cazul în care diferenţa de potenţial este diferită de zero, la unul dintre terminale, în zona de memorie, în registrul „PORTD" se va modifica valoarea de la zero (0) la valoarea unu (1).

În cazul contrar, când valoarea tuturor poziţiilor din registrul DDRD va fi unu logic, atunci, terminalele de la nivelul carcasei grupate în PORTD vor avea funcţia de ieşire digitală, iar conform conţinutului din memorie aflat în registrul de stare PORTD, potenţialul la terminalele date, se va modifica conform secvenţei memorate.

13.4.9 DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fig. 1.9 - Structura registrului de stabilire a modului de lucru „DDRD"

Registrul „PORTD" gestionează diferenţa de potenţial la nivelul terminalelor grupate în „PORTD" de la periferia carcasei microcontrolerului (Fig. 1.10).

13.4.8 PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fig. 1.10 - Structura registrului „PORTD" pentru stabilirea diferenţei de potenţial la nivelul terminalelor grupate în registrul „PORTD"

Manipularea în mod direct a informației binare aflată în conținutul registrelor din memoria microcontrolerului ATMega 328P presupune, cunoașterea temeinică a metodelor și procedeelor de programare în limbaj de asamblare și gestionarea eficientă a valorilor numerice prin transformarea lor dintr-o bază de numerație în alta.

Pentru a gestiona valorile numerice dintr-un șir binar în mod corect, se vor respecta trei reguli de bază:

1. Într-un șir de valori scrise în sistem de numerație binar indicele de ordonare în șir începe de la ZERO tot timpul!

Drept consecință a primei reguli, rezultă a doua regulă pentru gestionarea șirurilor de valori binare, anume:

2. Valoarea maximă a indicelui de ordonare într-un șir binar are expresia (2^{n-1})!

În cadrul expresiei (2^{n-1}), „2” reprezintă baza de numerație iar „n” reprezintă numărul de termeni sau de biți din cadrul șirului binar utilizat în vederea reprezentării unei valori numerice. Rezultă faptul că, pentru determinarea valorii în sistemul de numerație zecimal a unui și binar, se va utiliza metoda de exprimare sub forma unui polinom ponderat cu baza „2”.

Spre exemplu: Se dă șirul binar 1001, să se determine valoarea numerică zecimală exprimată prin intermediul șirului binar dat.

$$X_{(2)4} = b_{4-1} \cdot 2^{4-1} + b_{4-2} \cdot 2^{4-2} + b_{4-3} \cdot 2^{4-3} + b_{4-4} \cdot 2^{4-4}$$

$$X_{(2)4} = b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

$$X_{(2)4} = b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

$$X_{(2)4} = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$X_{(2)4} = 8 + 0 + 0 + 1 = 9$$

Unde „b_{n-i}” reprezintă elementul din șir (bitul) cu indicele de ordine „n-i”.

Totodată pe baza primelor două reguli, va fi posibilă întocmirea tabelului de adevăr, de corespondență între valorile binare, zecimale și hexazecimale. Spre exemplu, în vederea exprimării în binar, pe patru biți a unei valori zecimale, și în hexazecimal, tabelul (Tabel 1.1) de adevăr se va întocmi astfel:

N	Hex.	b3	b2	b1	b0
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1
8	8	1	0	0	0
9	9	1	0	0	1
10	A	1	0	1	0
11	B	1	0	1	1
12	C	1	1	0	0
13	D	1	1	0	1
14	E	1	1	1	0
15	F	1	1	1	1

Tabel 1.1 – Tabelul de adevăr pentru exprimarea valorilor de la 0 la 15 pe patru biţi

Ca şi consecinţă celei de-a doua regulă, rezultă a treia regulă de transformare numerică din orice bază de numeraţie în zecimal, anume:

3. Există o formulă general valabilă pentru transformare în sistem zecimal din orice bază de numeraţie având o reprezentare pe un număr „n" de biţi:

$$X_{(z)n} = \sum_{\substack{i=1 \\ i \le n}}^{n} b_{n-i} \cdot z^{n-i}$$

Unde, „z" reprezintă baza de numeraţie, „n" numărul de biţi iar „i" reprezintă indicele de ordine al elementului apartenent şirului de valori binare.

Spre exemplu: Fie valoarea numerică „0xFE" exprimată în hexazecimal. Să se determine, valoarea numerică în sistem zecimal corespunzătoare numărului dat iniţial.

$$X_{(16)2} = b_{2-1} \cdot 16^{2-1} + b_{2-2} \cdot 16^{2-2}$$

$$X_{(16)2} = b_1 \cdot 16^1 + b_0 \cdot 16^0$$

$$X_{(16)2} = 15 \cdot 16^1 + 14 \cdot 16^0 = 240 + 14 = 254$$

Efectuarea operaţiilor în binar presupune semnalarea unei serii de „evenimente" care pot avea loc în memoria sistemului de calcul precum:
- rezultat zero (presupune activarea bitului „Z" din registrul SREG);
- rezultat negativ (presupune activarea bitului „N" din registrul SREG);
- transport pe jumătate de octet (presupune activarea bitului „H" din registrul SREG);

- transport pe întreg octetul (presupune activarea bitului „C” din registrul SREG);

Spre exemplu, la însumarea valorii binare „0001” cu „0001”, are loc fenomenul de transport binar, de la poziţia bitului zero la poziţia bitului unu.

Fenomenul de transport binar are loc din cauza depăşirii domeniului de reprezentare al bazei de numeraţie în poziţia din şir considerată!

BINAR:

0001 +
0001

0010

Un alt exemplu ar fi însumarea în format zecimal a numărului „99999999” cu „1” (în cazul calculatorului cu număr finit de digiţi).

ZECIMAL:

99999999 +
1

E00000000 – unde caracterul „E” reprezintă eroare cauzată de depăşirea domeniului de reprezentare pe un ecran cu număr finit de digiţi

Un alt exemplu ar fi însumarea valorilor „0x0F” şi „0x01” în format hexazecimal:

HEXAZECIMAL:

0x0F +
0x01

0x10 → BINAR: 0001 0000

În limbaj de asamblare, pentru a manipula conţinutul registrelor de memorie corespunzătoare intrărilor şi ieşirilor digitale, se vor utiliza următoarele instrucţiuni:
- sensul curentului prin terminale digitale grupate în PORTD:

`DDRD = B00000000;` (terminalele îndeplinesc funcţia de intrări digitale);
`DDRD = B11111111;` (terminalele îndeplinesc funcţia de ieşiri digitale);

- starea logică sau diferenţa de potenţial a terminalelor grupate în PORTD:
`PORTD = B00000000;` (terminalele au starea logic „0” şi nu furnizează tensiune);
`PORTD = B11111111;` (terminalele au starea logic „1” şi furnizează tensiunea de alimentare);

V. EXEMPLE DE IMPLEMENTARE

Pentru a evidenţia modul de programare în limbaj hibrid de asamblare şi sintaxă a microcontrolerului ATMega 328P, se va realiza o aplicaţie cu un şir de 8 LED-uri pe care se vor afişa trenuri binare sau valori hexazecimale. Pentru exersarea noţiunilor teoretice prezentate se vor fi utiliza următoarele componente:
- placă pentru testare rapidă a circuitelor electronice (Wisher WBU-502L);
- microcontroler ATMega 328P;
- diode electroluminiscente (LED-uri);
- rezistenţe cu valoarea de 100 [Ω];
- fire pentru conexiune rapidă compatibile cu placa de testare;
- dispozitiv programator SparkFun Tiny AVR Programmer – PGM 11801;
- oscilator pe bază de cristal de cuarţ (având frecvenţa de 16 MHz);
- calculator gazdă având mediul Arduino IDE instalat;
- cablu adaptor sau prelungitor de la tip USB A la tip USB A;

Se va realiza montajul (Fig. 1.12), [16], conform schemei din figura 1.11, [15].

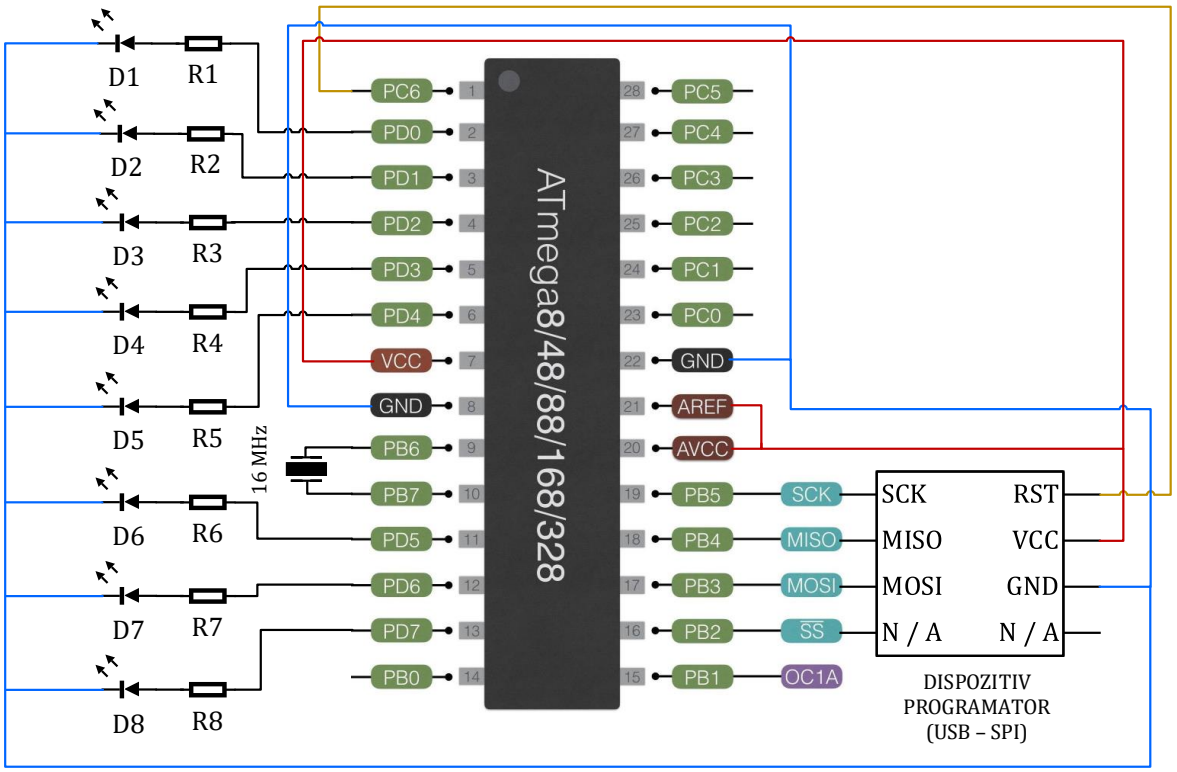


Fig. 1.11 – Schema electronică a montajului

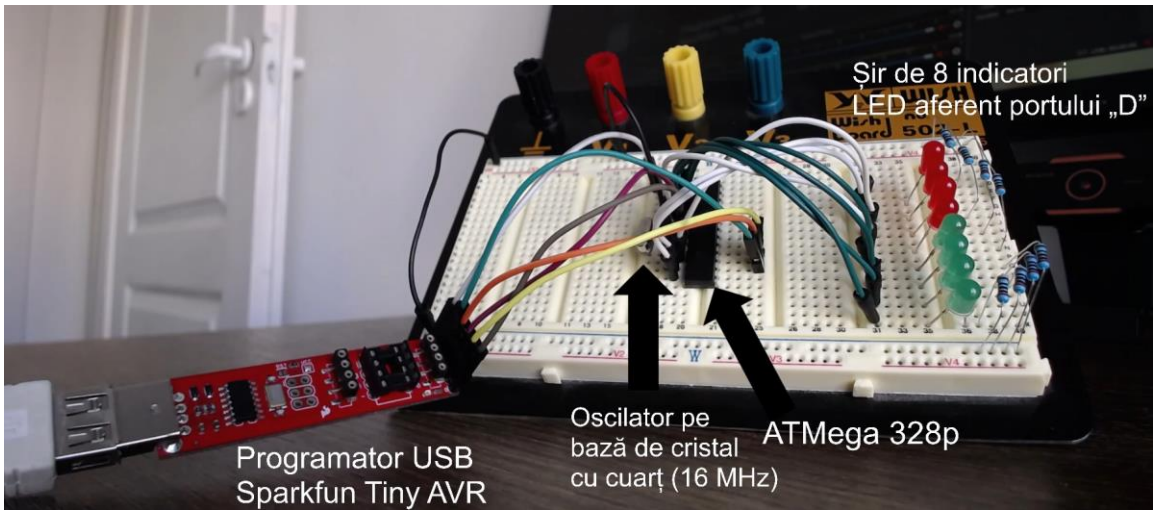


Fig. 1.12 – Montaj experimental

Pentru a transfera informaţia numerică în memoria microcontrolerului ATMega 328P, se va utiliza dispozitivul SparkFun Tiny AVR Programmer – PGM 11801. Dispozitivul programator reprezintă un adaptor USB la SPI (eng. Serial Peripheral Interface). Pentru a instala pachetul de instrucţiuni aferent dispozitivului programator specific sistemului de operare Microsoft Windows, se va accesa pagina de la adresa următoare: https://www.sparkfun.com/products/11801.

De asemenea, este necesar şi setul de instrucţiuni pentru compilator în vederea programării microcontrolerului ATMega 328P în mediul Arduino IDE, anume „ATMega Breadboard”. Acest pachet de instrucţiuni poate fi regăsit la următoarea adresa: https://github.com/technoblogy/atmegabreadboard. Arhiva se va decomprima în locaţia „C:\Users\<numele_utilizatorului>\Documents\Arduino\hardware”. Iniţial directorul cu denumirea „hardware” nu există în locaţia indicată. Acesta se va crea în mod manual de către utilizatorul calculatorului. De asemenea, locaţia indicată nu există numai după prima rulare a mediului Arduino IDE. Versiunea mediului Arduino IDE utilizată pentru acest laborator este 1.8.19.

După pregătirea tuturor componentelor necesare în vederea programării microcontrolerului ATMega 328P, se vor parcurge următoarele etape:

A. Se va crea un proiect nou în cadrul mediului Arduino IDE 1.8.19, din meniul „File”, cu opțiunea „New” (Fig. 1.13).

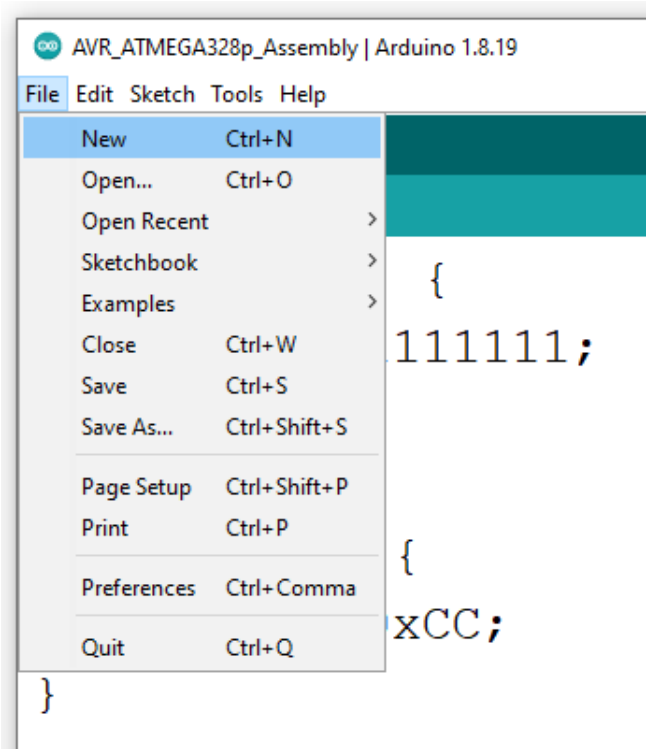


Fig. 1.13 – Crearea unui proiect nou în mediul Arduino IDE

B. Se va salva proiectul nou început într-o locație accesibilă, prin intermediul meniului „File” și a opțiunii „Save As...” (Fig. 1.14).

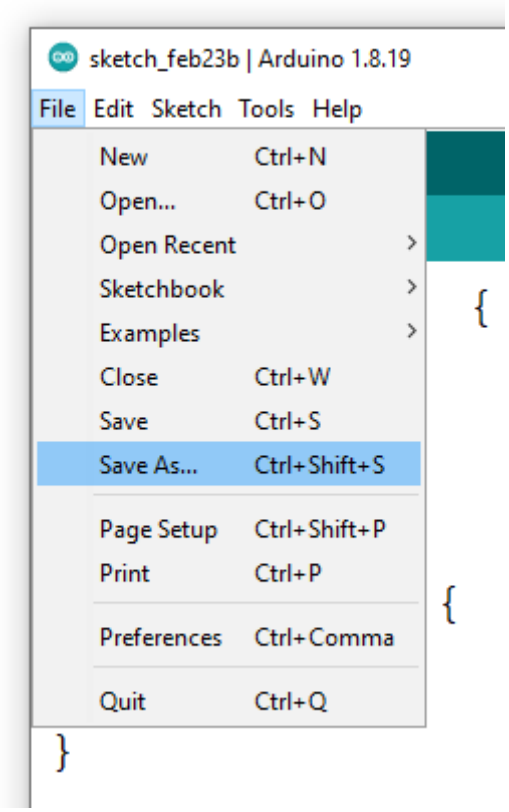


Fig. 1.14 – Salvarea proiectului nou creat

C. Se va alege tipul de microcontroler care urmează a fi programat cu ajutorul mediul Arduino IDE, din meniul „Tools”, opţiunea „Boards”, „ATmegaBreadboard (in sketchbook)”, „ATmega328” (Fig. 1.15).

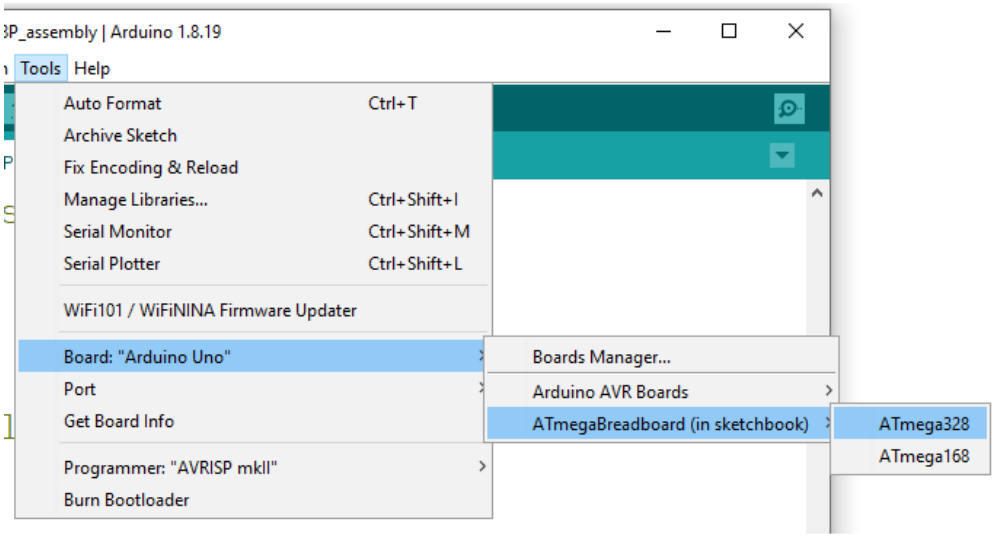


Fig. 1.15 – Alegerea tipului de microcontroler care urmează să fie programat

D. Se va alege varianta constructivă a microcontrolerului, tot din meniul „Tools”, opţiunea „Chip”, „ATmega328P” (Fig. 1.16).

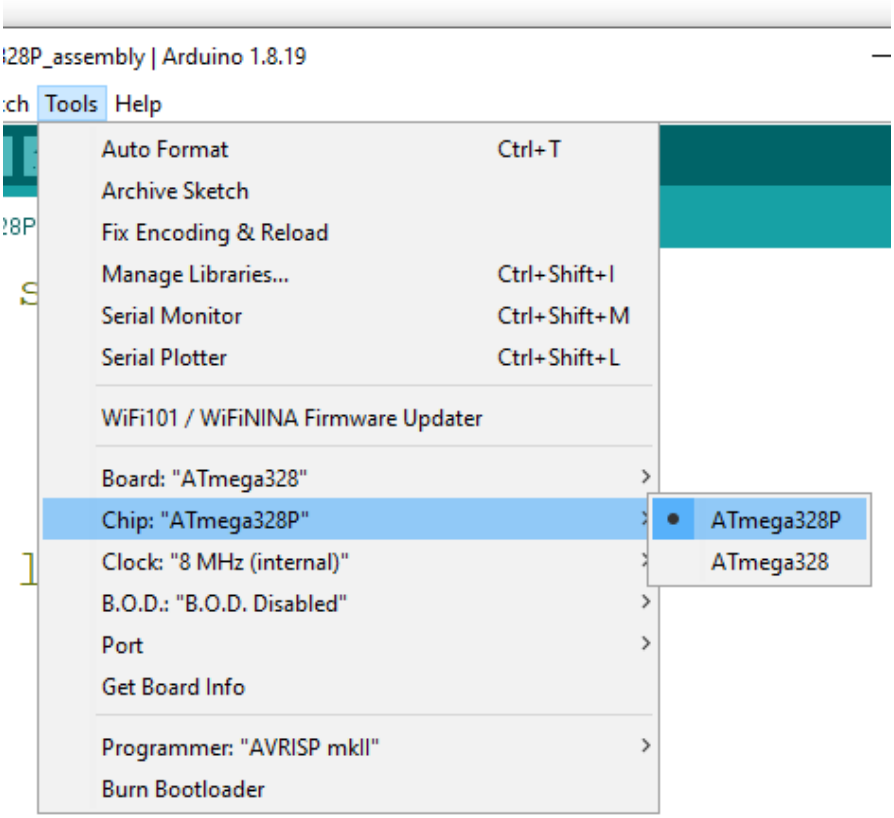


Fig. 1.16 – Alegerea variantei constructive a microcontrolerului care urmează să fie programat

E. Se va alege frecvenţa oscilatorului pe bază de cuarţ ataşat la terminalele „XTAL1” şi „XTAL2” ale microcontrolerului ATMega 328P, tot din meniul „Tools” opţiunea „Clock”, „16 MHz (external)” (Fig. 1.17).

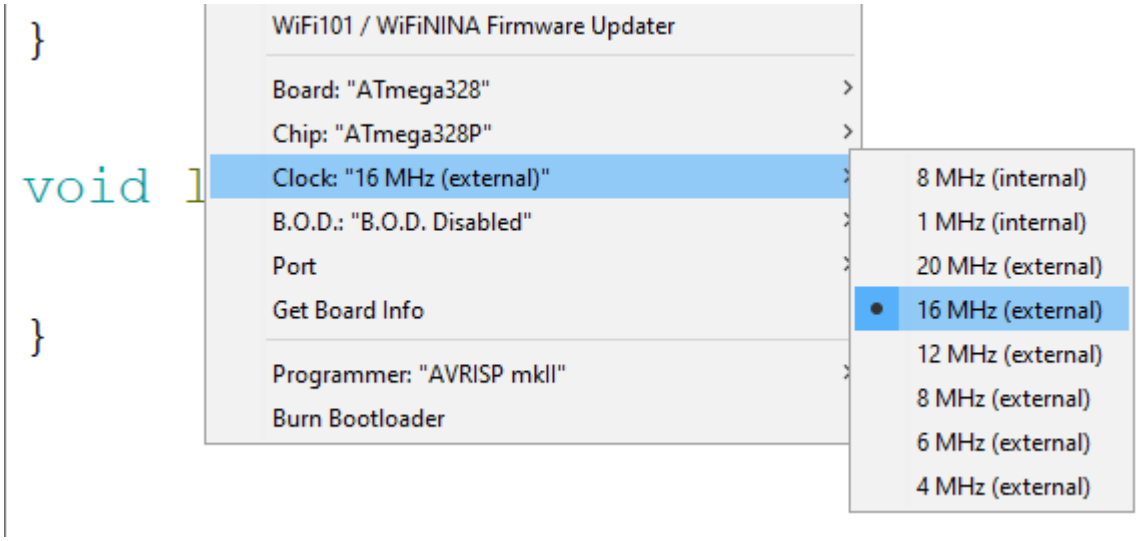


Fig. 1.17 – Alegerea frecvenţei oscilatorului pe bază de cuarţ ataşat la microcontroler

F. Se va alege de asemenea, nivelul critic al tensiunii de alimentare, tot din meniul „Tools”, opţiunea „B.O.D.”, „B.O.D. Disabled” (eng. Brown Out Detection = determinarea nivelului minim sau critic al tensiunii de alimentare) (Fig. 1.18).

Fig. 1.18 – Alegerea nivelului minim al tensiunii de alimentare

G. Se va alege dispozitivul programator tot din meniul „Tools”, opţiunea „Programmer”, „USBtinyISP” (eng. In – System Programmer – ISP) (Fig. 1.19).

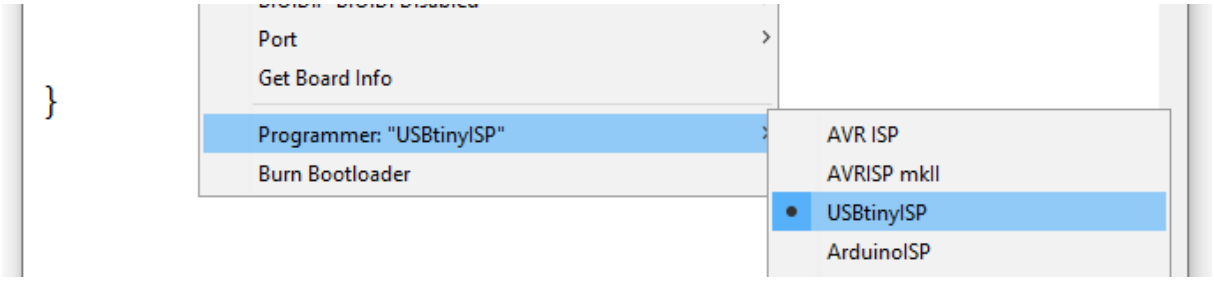


Fig. 1.19 – Alegerea dispozitivului programator

H. Pentru a încheia etapa de parametrizare inițială se va alege opțiunea „Burn Bootloader” (încărcare secvență de cod pentru inițializare), din meniul „Tools” (Fig. 1.20).

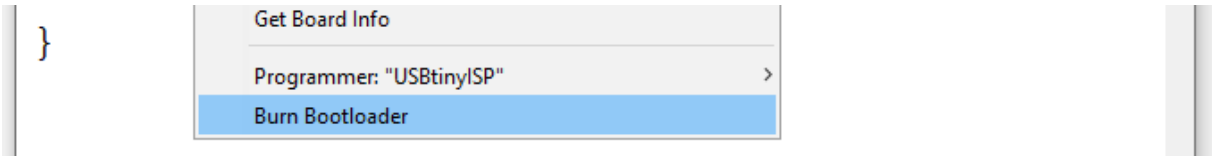


Fig. 1.20 – Încărcarea secvenței de cod pentru inițializarea codului program din memorie

În urma parcurgerii tuturor etapelor de configurare a microcontrolerului și a dispozitivului programator, se va trece la etapa de elaborare a codului program. Pentru a studia modul de manipulare directă a registrelor de memorie alocate intrărilor și ieșirilor digitale de uz general grupate în „PORTD”, se propune implementare a trei aplicații:
- aplicația specifică studierii circuitelor logice combinaționale (statice);
- aplicația specifică studierii circuitelor logice secvențiale (dinamice);
- aplicația specifică instrucțiunilor iterative (numărător binar);

OBSERVAȚII:

- structura „void setup()” reprezintă o secvență de inițializare care se va executa o singură dată la alimentarea cu tensiune sau la apăsarea butonului „RESET”;
- structura „void loop()” reprezintă o secvență repetitivă la infinit și este echivalentă cu expresia în sintaxă C standard „while(1)” sau „while(true)”;

APLICAȚIA 1:

Se va implementa următorul cod program:

```
void setup() {
  DDRD = B11111111;
}
void loop() {
  PORTD = B00001111;
}
```

În urma implementării codului program cu ajutorul editorului de text, din meniul „Sketch” se va alege opțiunea „Upload Using Programmer” (încărcarea programului în memoria microcontrolerului cu ajutorul dispozitivului programator) (Fig. 1.21).

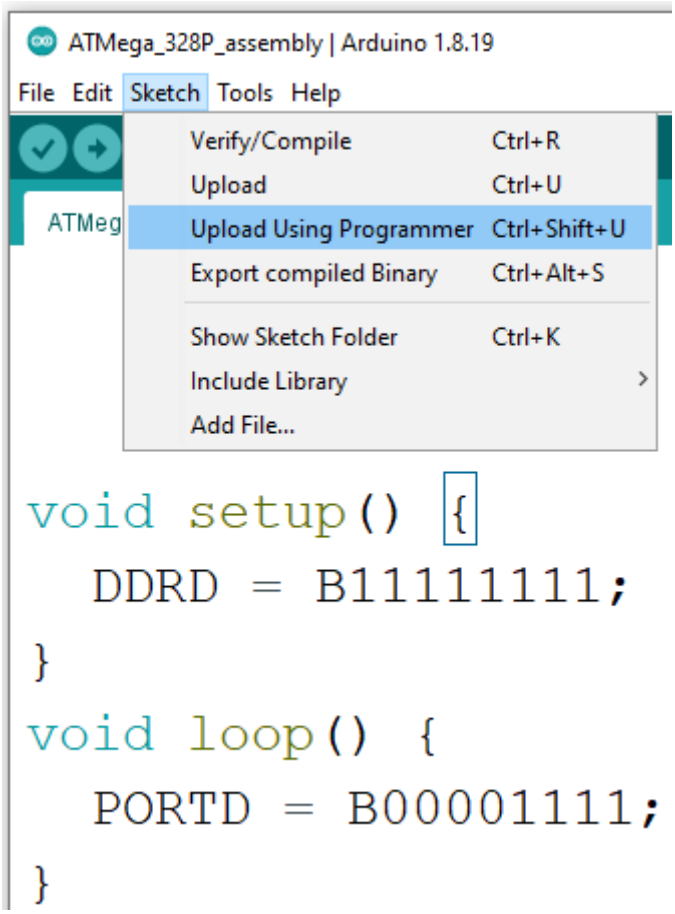


Fig. 1.21 – Încărcarea codului program în memoria microcontrolerului cu ajutorul dispozitivului programator

În urma încărcării codului program în memoria microcontrolerului se va observa evoluția stării logice a grupării de terminale „PORTD" prin intermediul șirului de diode LED indicatoare atașate la microcontroler. Starea logică a grupării PORTD nu se va modifica în timp până la introducerea unei noi combinații.

Se vor introduce diverse valori numerice atât în format binar (B********), cât și hexazecimal (0x**) în registrul „PORTD".

APLICAȚIA 2:

Se va implementa următorul cod program:

```
void setup() {
  DDRD = B11111111;
}
void loop() {
  PORTD = B10101010;
  delay(100);
  PORTD = ~B10101010;
  delay(100);
}
```

Spre deosebire de aplicaţia anterioară, în acest caz, avem declarat şi parametrul „timp de întârziere sau aşteptare” într-o anumită stare (eng. delay). Timpul exprimat cu ajutorul funcţiei „delay()” este de ordinul milisecundelor. Totodată, în cazul unui circuit logic secvenţial se va constata faptul că, există cel puţin două stări logice distincte care se manifestă la un anumit interval de timp. A doua stare logică a registrului „PORTD” a fost impusă prin aplicarea operatorului de inversare sau negare logică „~”.

APLICAŢIA 3:

Se va implementa următorul cod program:

```
void setup() {
  DDRD = B11111111;
}
void loop() {
  for(byte i = 0; i <= 0xFF; i++) {
    PORTD = i;
    delay(1000);
  }
}
```

În cadrul ultimei aplicaţii a fost implementat un cod program numărător, care parcurge de la 0 la 255 toate valorile hexazecimale care pot fi afişate pe 8 biţi (reprezentaţi prin cele 8 LED-uri), urmând ca la ieşirile digitale grupate în registrul „PORTD” să fie transferate toate combinaţiile în binar ale valorilor rezultante, la un interval de o secundă.

VI. CONCLUZII

Manipularea în mod direct a registrelor de memorie în cadrul unui microcontroler reprezintă o soluţie viabilă atât din punct de vedere al optimizării modului de execuţie al unei aplicaţii cât şi din punct de vedere al spaţiului ocupat de aceasta în memoria sistemului de calcul dedicat implementării. Această metodă permite de asemenea, deblocarea unor funcţii speciale la nivel de microcontroler (ex. modificarea frecvenţei semnalului modulat în lăţime – eng. PWM).

VII. BIBLIOGRAFIE

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

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

3. MicroChip Technology Inc. © 2004 – 2014 – „PIC 10F200 / 202 / 204 / 206 datasheet”;

4. Infineon Technologies AG, 81726 Munich, Germany © 2017 – „XMC1400 AA-Step, Microcontroller Series for Industrial Applications, the XMC1000 Family with ARM ® Cortex ® - M0 32-bit processor core" – Datasheet;

5. STMicroelectronics © 2020, DS6329 Rev. 18 – „STM32F205xx, STM32F207xx – datasheet", „ARM®-based 32-bit MCU, 150 DMIPs, up to 1 MB Flash / 128 + 4KB RAM, USB OTG HS / FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm, interfaces and camera;

6. MicroChip Technology Inc. © 2004 – 2014 – „Explorer 16/32 Development Board (datasheet) - A Perfect Platform for Discovering the Full Capabilities of PIC ® MCUs and dsPIC ® DSCs";

7. Infineon Technologies AG, 81726 Munich, Germany © 2017 – „AURIX™ TC275 lite Kit - Evaluation Board datasheet";

8. STMicroelectronics © 2020 – „STM32 Nucleo-64 boards (MB1136) – user manual";

9. Arduino ® – „Arduino UNO R3 - Product Reference Manual";

10. Protostack.com – „ATMEGA328P-PU Atmel 8 Bit 32K AVR Microcontroller";

11. Arduino ® forum – „Reducing power supply noise";

12. Technoblogy.com – „Using an ATmega328 without a crystal";

13. Jet Support Service – „Arduino Parts DIY Arduino Guide Microcontroller Tutorials";

14. epe.utcluj.ro – Prezentare laborator nr. II la disciplina Sisteme cu Microprocesoare – „Utilizarea platformei de dezvoltare Arduino";

15. github.com – „MiniCore - DIP-28 package ATmega8 / 48 / 88 / 168 / 328 pinout"

16. epe.utcluj.ro – Material video demonstrativ la disciplina Sisteme cu Microprocesoare - „Programarea microcontrolerului ATMega 328p în limbaj hibrid sintaxă - asamblare";

Tema de studiu nr. 2 - Manipularea intrărilor şi ieşirilor digitale

I. SCOPUL TEMEI

- 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. 2.1).

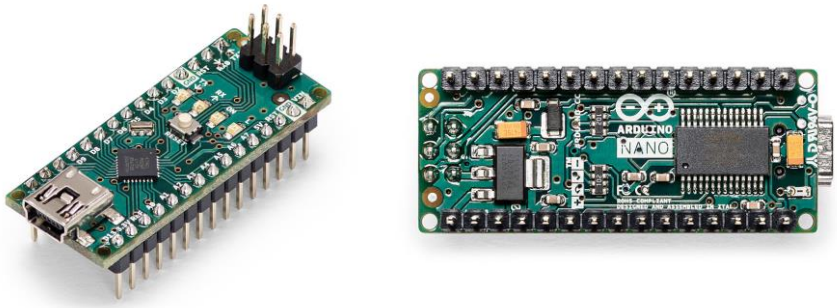


Fig. 2.1 – Platforma de dezvoltare Arduino Nano [1]

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. 2.2).

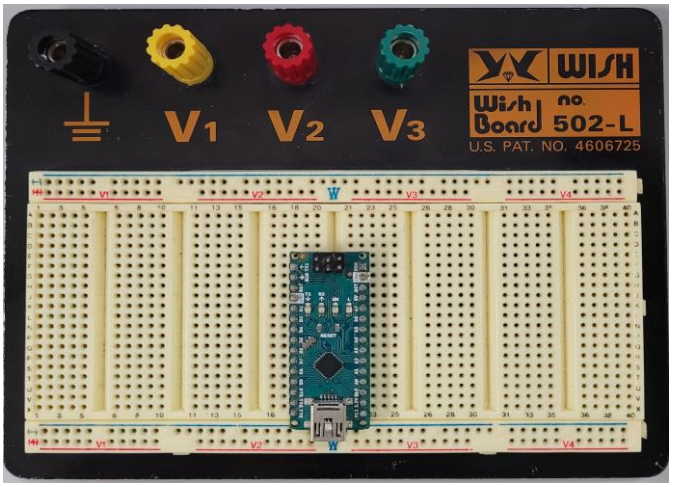


Fig. 2.2 – Instalarea platformei Arduino NANO la nivelul unei plăcuţe pentru testare

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. 2.3).

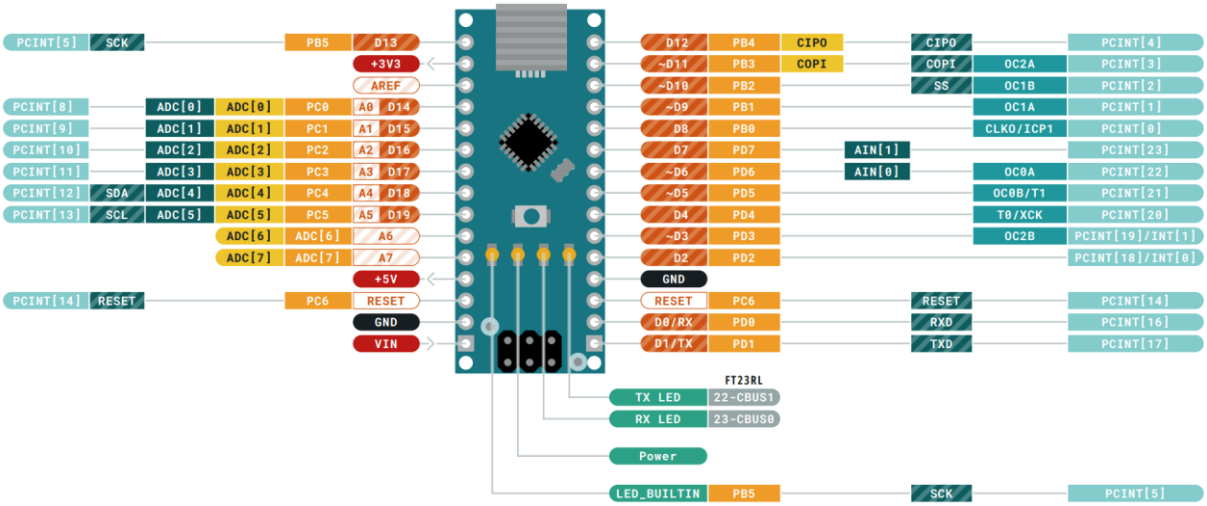


Fig. 2.3 – Harta funcţiilor specifice terminalelor platformei Arduino Nano [1]

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. 2.4).

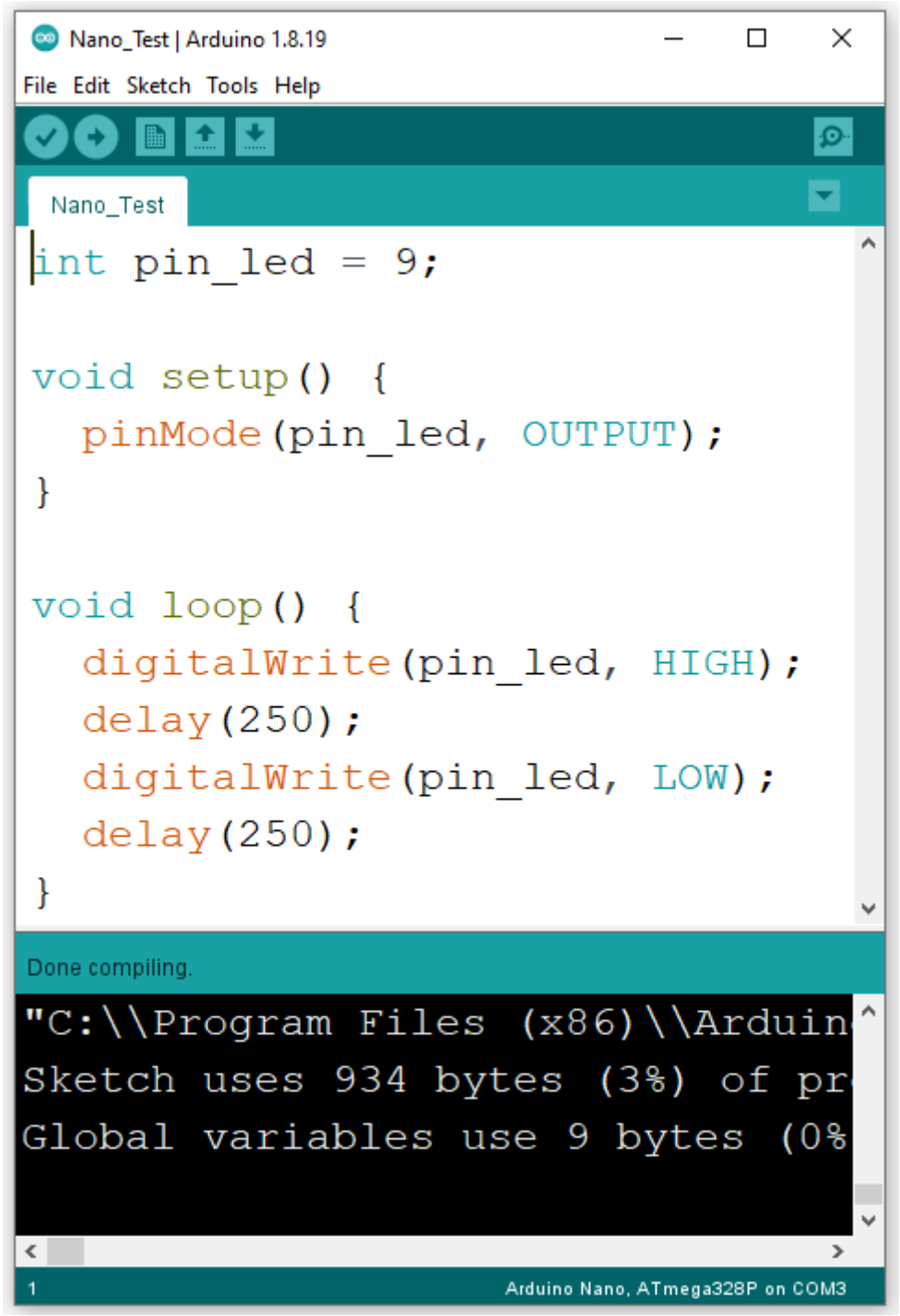


Fig. 2.4 – Mediul de dezvoltare Arduino IDE versiunea 1.8.19

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;

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. 2.5).

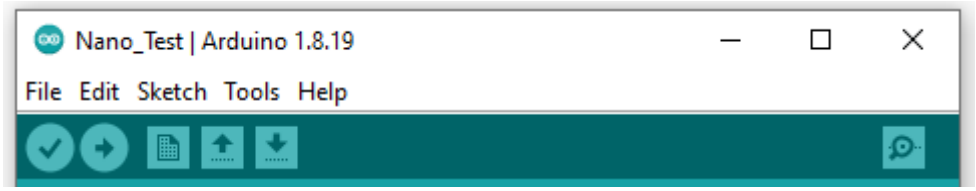


Fig. 2.5 – Secţiunea meniurilor principale, a paletei de instrumente şi comenzi rapide

Editorul de text permite redactarea codului program după regulile de formatare ale limbajului Wiring sau C++ (Fig. 2.6).

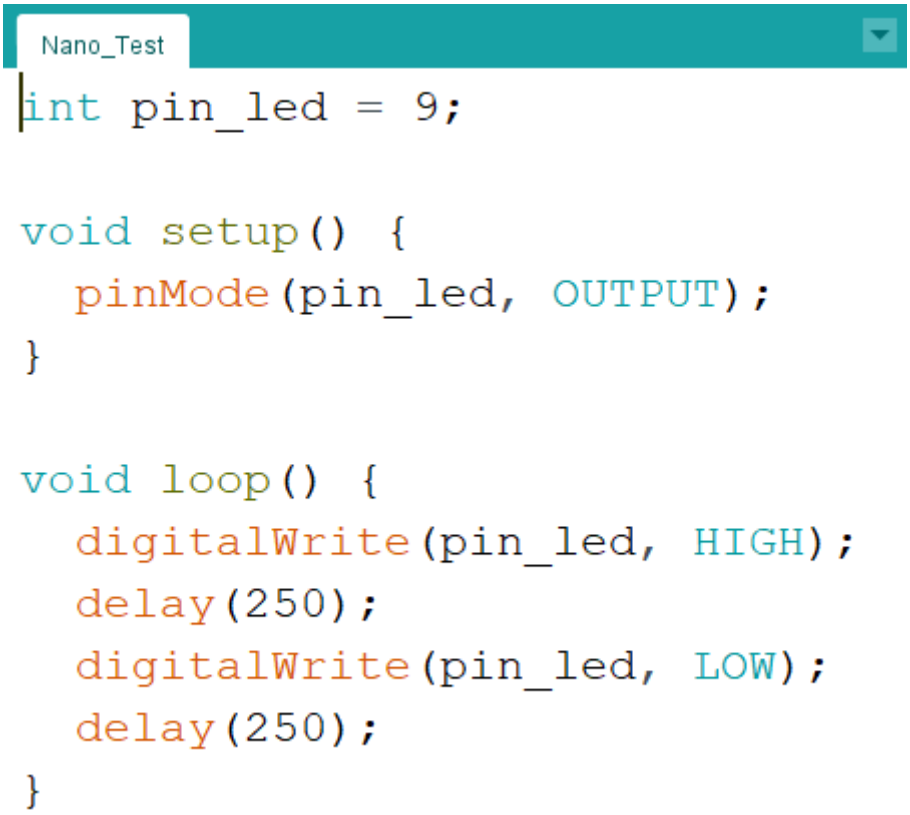


Fig. 2.6 – Editorul de text

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. 2.7).

Fig. 2.7 – Bara de stare

Consola afişează descrierea în mod detaliat a operaţiilor efectuate în vederea parcurgerii procesului de programare al microcontrolerului (Fig. 2.8). Gradul de detaliere a descrierii operaţiilor efectuate poate fi stabilit prin alegerea opţiunilor „Show verbose output during compilation and upload" din meniul „File" → „Preferences".

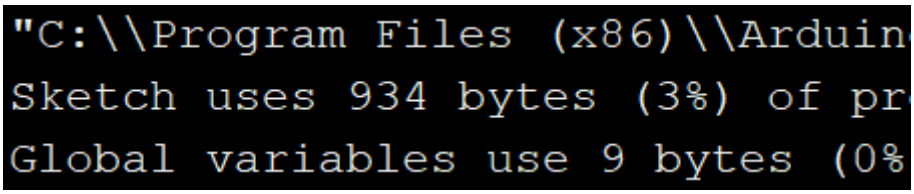


Fig. 2.8 – Consola pentru descrierea operaţiei actuale

Bara pentru afişare a rândului actual şi a platformei actuale împreună cu setările aferente permite utilizatorului să efectueze operaţii de verificare şi diagnosticare a potenţialelor probleme care au loc în cadrul procesului de programare (Fig. 2.9). Tot în cadrul acestei secţiuni, există posibilitatea de a identifica portul pentru comunicaţia „Serial" (ex. COM3) sau tipul dispozitivului programator utilizat.

Fig. 2.9 – Bara pentru afişare a rândului actual şi a platformei actuale

În cadrul mediului Arduino IDE proiectele poartă denumirea „sketch" (schiţă de lucru). Extensia fişierului de proiect generat în Arduino IDE poate să fie „.ino" sau „.pde". Fişierele care conţin codul maşină rezultant pot avea fie extensia „.hex" (fişier cu conţinut numeri reprezentat în cod hexazecimal) sau fişier „.bin" (fişier cu conţinut numeric reprezentat în cod binar) (Fig. 2.10).

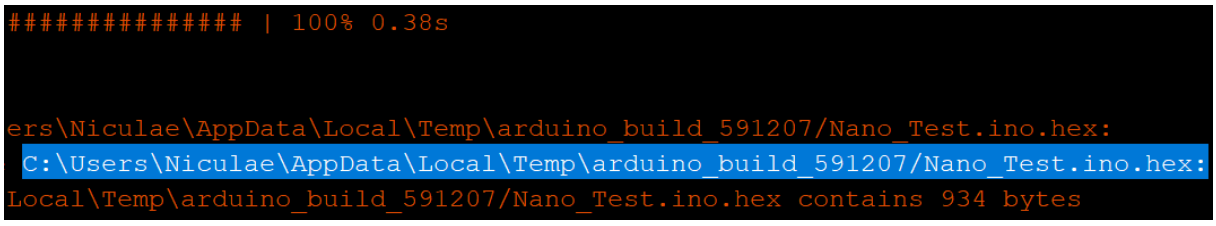


Fig. 2.10 – Fişierul care conţine codul maşină rezultant în format numeric hexazecimal

Fişierul rezultant „.hex" în format numeric hexazecimal va fi transferat în memoria microcontrolerului prin interfaţa 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 aplicaţiei „avrdude", care rulează în fundal ca şi un serviciu similar serviciilor din componenţa sistemului de operare UNIX / Linux, cât şi modul de declarare al căilor de acces la fişiere (ex. „\avr\etc\avrdude.conf"), se poate afirma faptului că aplicaţia „avrdude" şi alte componente ale mediului Arduino IDE rulează în mod virtualizat pe platforma sistemului de operare Microsoft Windows. Instrumentele necesare pentru a permite rularea aplicaţiilor 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 variaţiuni, precum:
- structurile de program precum „setup()" şi „loop()";
- declaraţiile hardware precum „pinMode()" şi „digitalWrite";

Setul de declaraţii hardware ataşat limbajului de programarea „C / C++" poartă denumirea „Wiring" sau în unele documentaţii de specialitate „Wiring C". Limbajul sau setul de instrucţiuni „Wiring" reprezintă un mod de a intermedia procesul de programare între persoana care elaborează codul program şi maşina de lucru finală pe care urmează să fie implementat codul program. Acest concept poartă denumirea API (eng. Application Programming Interface). Practic, fără pachetul de instrucţiuni Wiring, ar fi necesară programarea microcontrolerului în limbaj hibrid de tip „mixed assembly – syntax", în cadrul căruia registrele de memorie sunt manipulate în mod direct (a se revedea documentaţia din cadrul primului laborator, capitolul specific procedeului de implementare al aplicaţiilor) [3].

Prin urmare, instrucţiunile „pinMode()" şi „digitalWrite()", specifice setului API Wiring au ca şi operaţie corespondentă în limbajul de asamblare manipularea în mod direct a registrelor „DDRD" (eng. Data Direction Register of port D) şi „PORTD" (registrul specific grupului de terminale PORTD amplasate în periferia carcasei microcontrolerului). Spre exemplu, următoarea relaţie de echivalenţă 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 şi declaraţii hardware cât şi elemente de sintaxă a limbajului „C / C++", dar şi denumiri de registre dedicate precum „PORTD" şi „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);
}
```

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 INPUT sau ieşire 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:

```
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. 2.11).

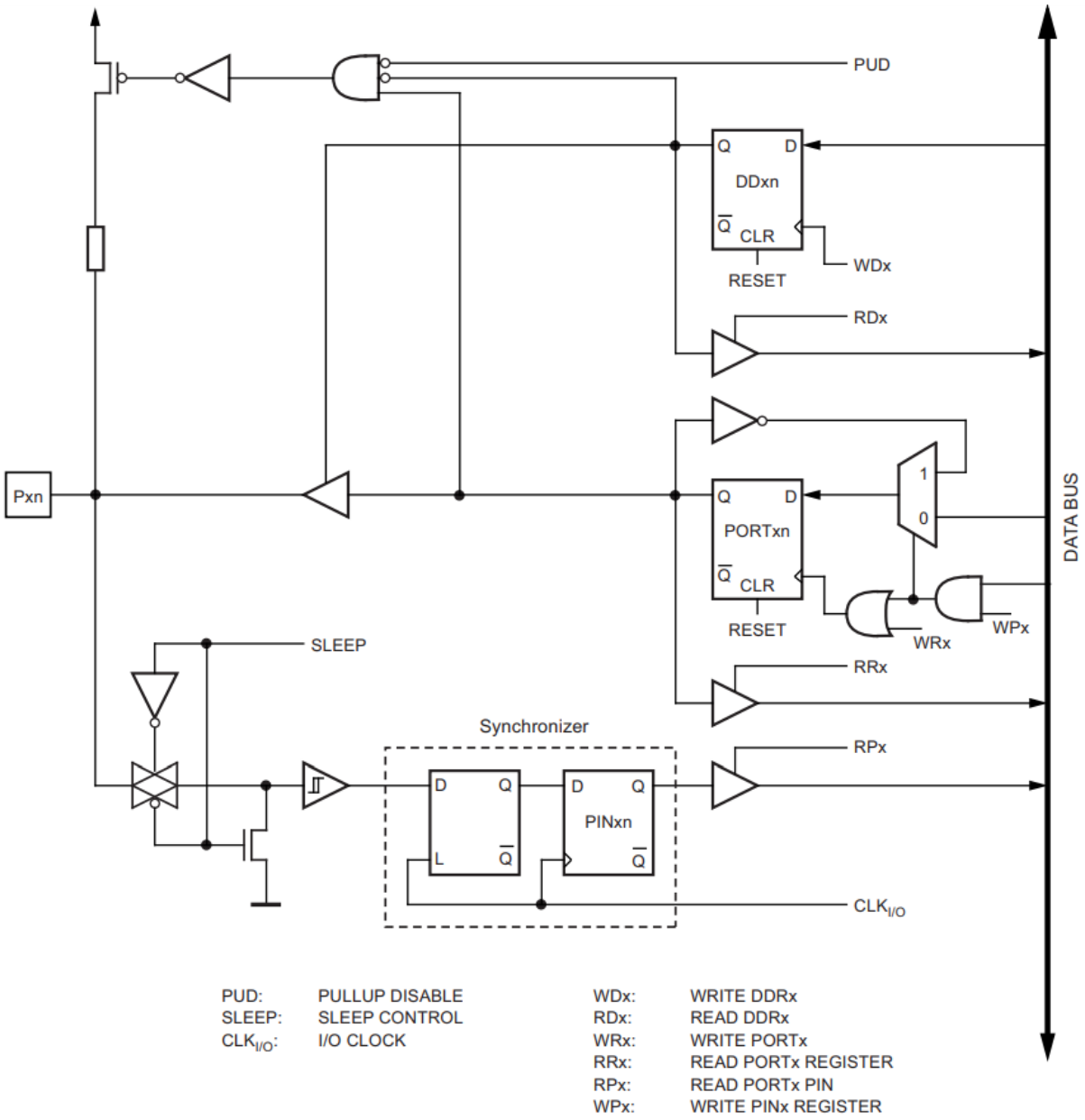


Fig. 2.11 – Arhitectura internă a microcontrolerului ATMega 328 – circuitele logice care stau la baza grupurilor de intrări şi ieşiri digitale [6]

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. 2.12). În literatura de specialitate modulaţia în lăţime a impulsului poartă denumirea (eng. Pulse Width Modulation – PWM).

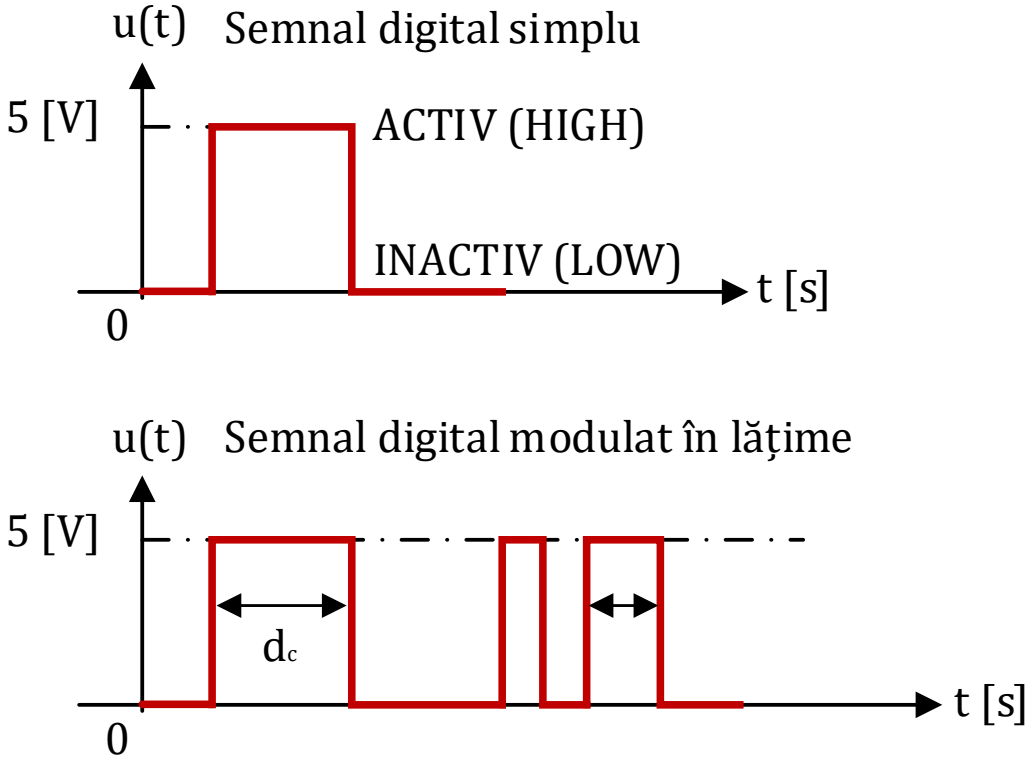


Fig. 2.12 – Semnale de natură digitală

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 „Tools". 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>) {};

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 propun spre implementare șase aplicații:
- Semnalizare intermitentă continuă cu două diode electroluminiscente (eng. LED);
- Semnalizare intermitentă variabilă cu două diode electroluminiscente;
- Semnalizare intermitentă continuă cu opt diode electroluminiscente;
- 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. 2.13):

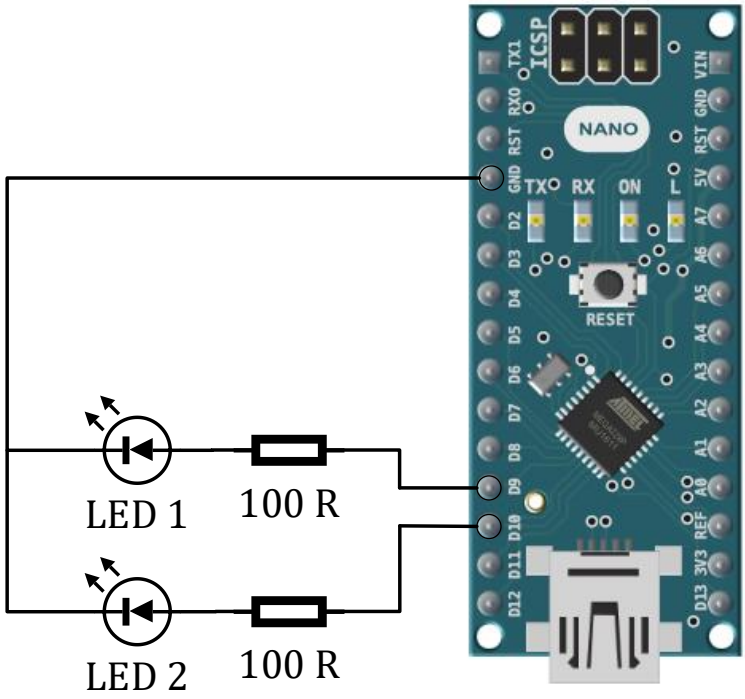


Fig. 2.13 – Schema electronică pentru implementarea circuitului specific aplicației 1 [7]

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. 2.14).

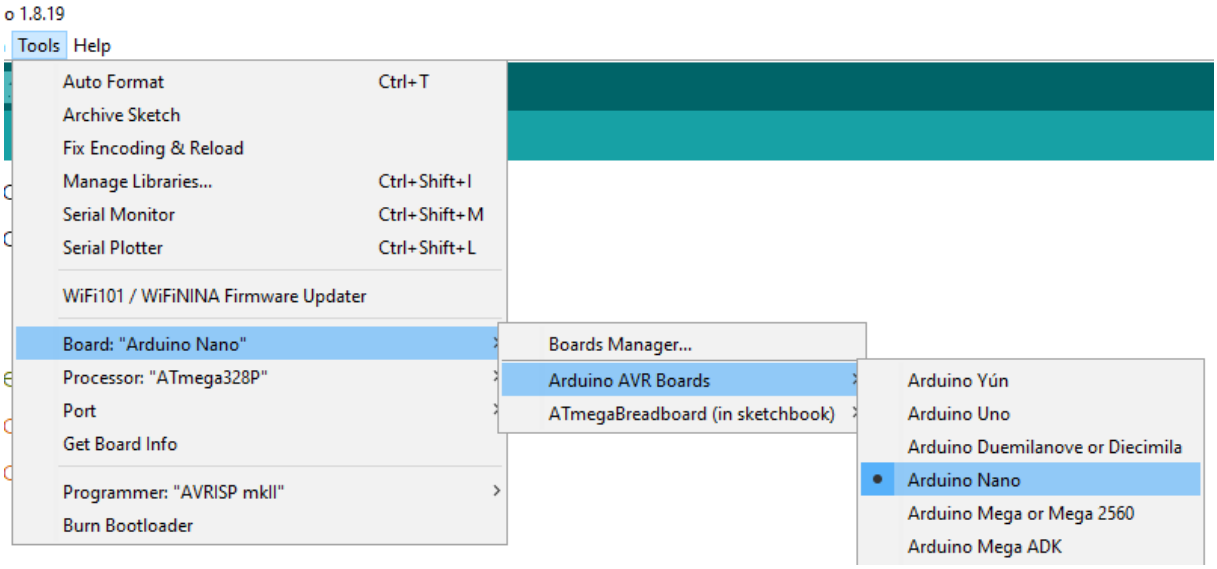


Fig. 2.14 – Alegerea platformei de dezvoltare care urmează să fie programată

B. Din meniul „Tools” se va alege sub-categoria „Processor” --> „ATMega 328P” (Fig. 2.15).

Fig. 2.15 – Alegerea variantei constructive a procesorului platformei de dezvoltare

C. Din meniul „Tools” se va alege sub-categoria „Port” --> „COMx” (Fig. 2.16).

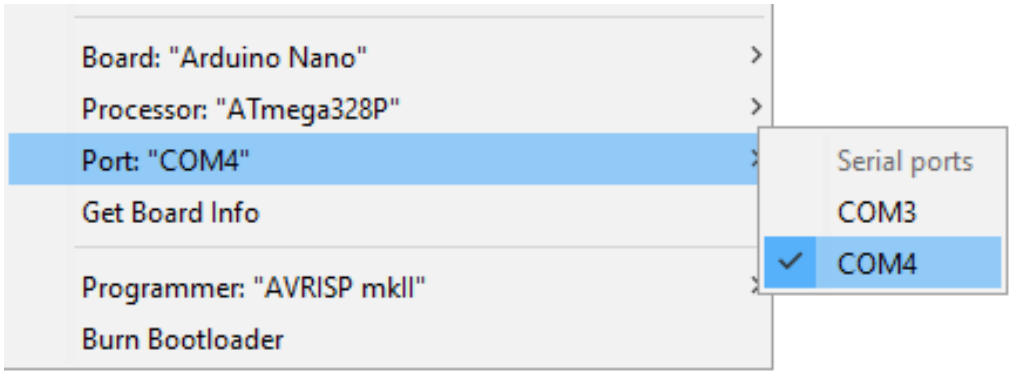


Fig. 2.16 – Alegerea portului pentru comunicaţia Serial

D. Din meniul „Sketch” se va alege opţiunea „Upload” (Fig. 2.17).

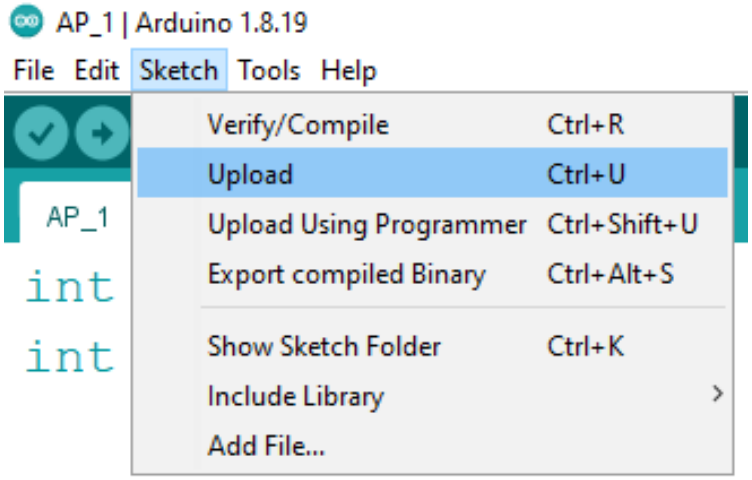


Fig. 2.17 – Încărcarea codului program în memoria microcontrolerului

În urma încărcării aplicației se va verifica funcționalitatea montajului (Fig. 2.18).

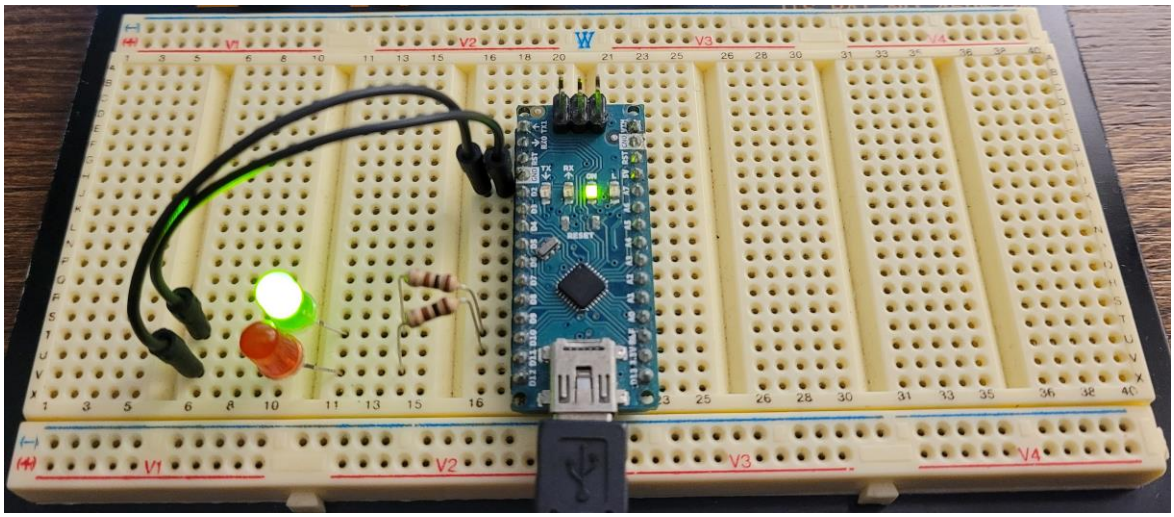


Fig. 2.18 – Montaj experimental pentru aplicația 1 și 2

Implementarea aplicației nr. 1 presupune:

- declararea a două variabile de tip număr întreg „led_1” și „led_2” având ca valori numerele „9” și „10”, adică 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. 2.18).

APLICAȚIA 2

Pe baza aceluiași circuit de la aplicația nr. 1 (Fig. 2.13 și 2.18), 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);
}
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: 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. 2.18).

APLICAȚIA 3

Se va implementa circuitul conform următoarei scheme (Fig. 2.19):

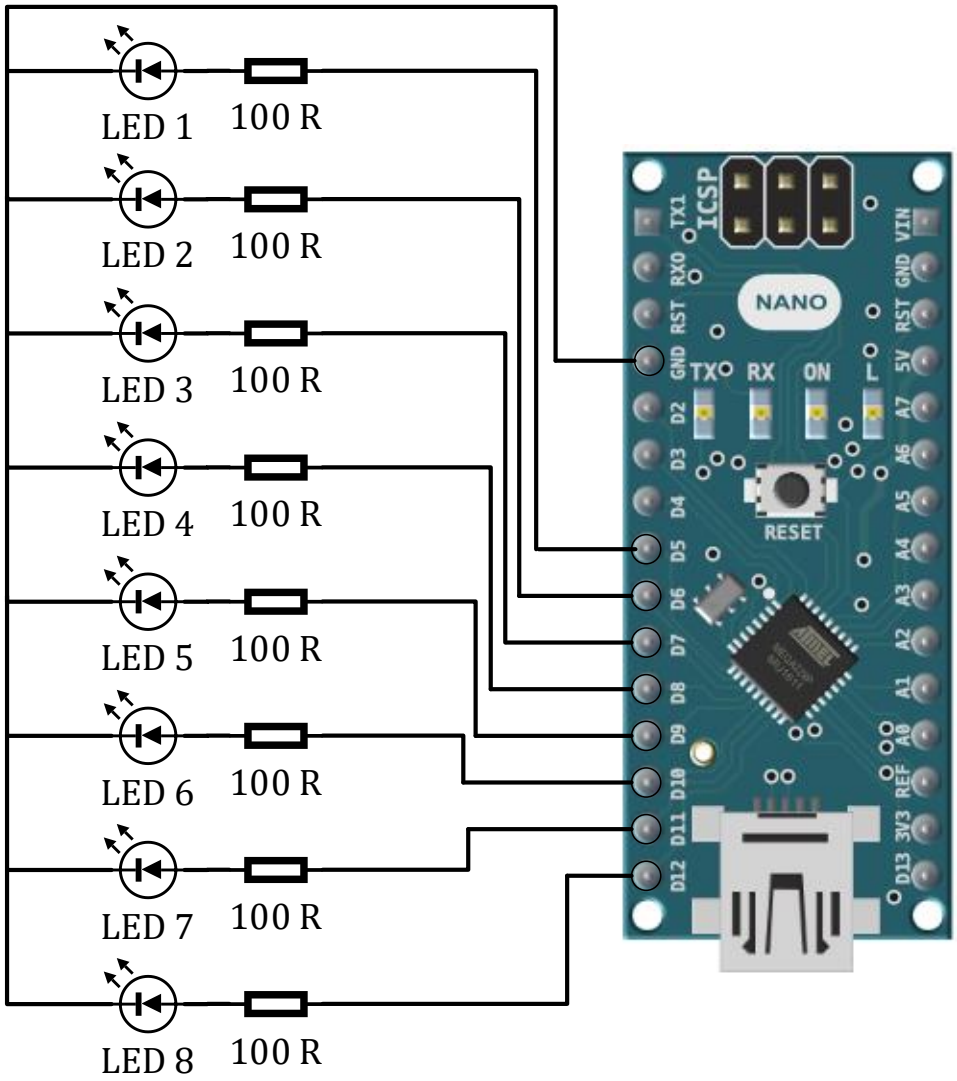


Fig. 2.19 – Schema electronică pentru implementarea circuitului specific aplicației 3 [7]

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);
  }
}
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. 2.20):

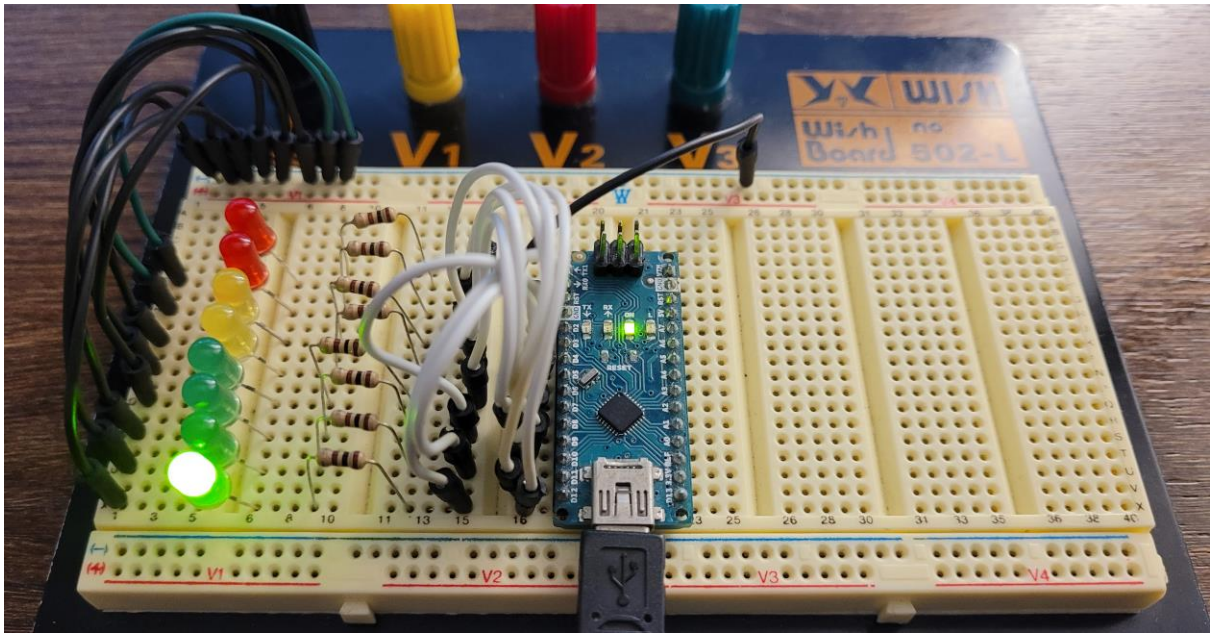


Fig. 2.20 – Montaj experimental pentru aplicația 3

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;

APLICAȚIA 4

Se va implementa circuitul conform următoarei scheme (Fig. 2.21):

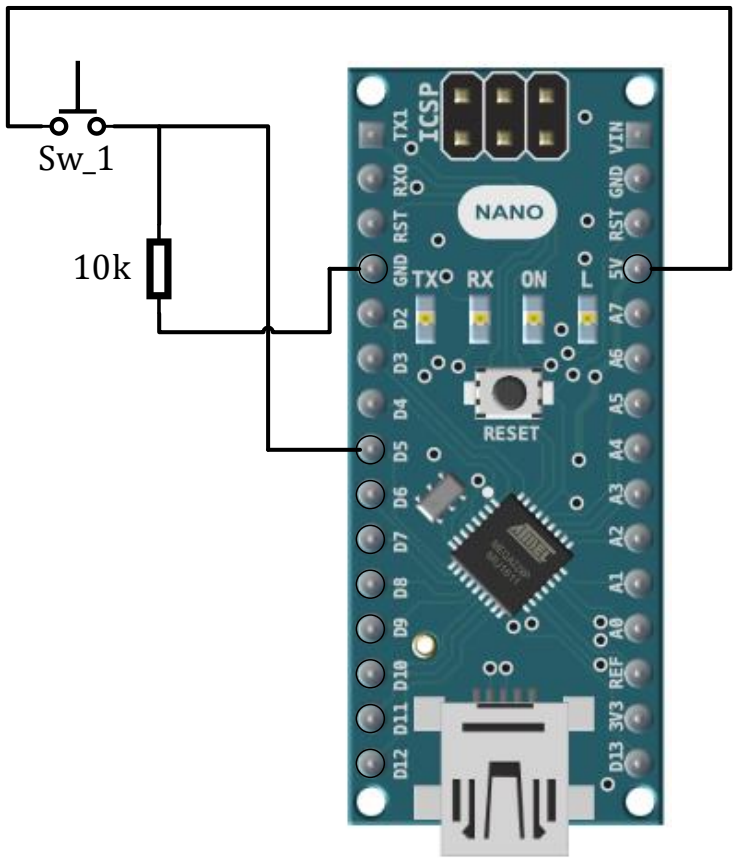


Fig. 2.21 – Schema electronică pentru implementarea circuitului specific aplicației 4 [7]

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);
}
```

Pentru a verifica funcţionalitatea aplicaţiei nr. 4, se va deschide consola Serial din meniul „Tools” opţiunea „Serial Monitor” (Fig. 2.22).

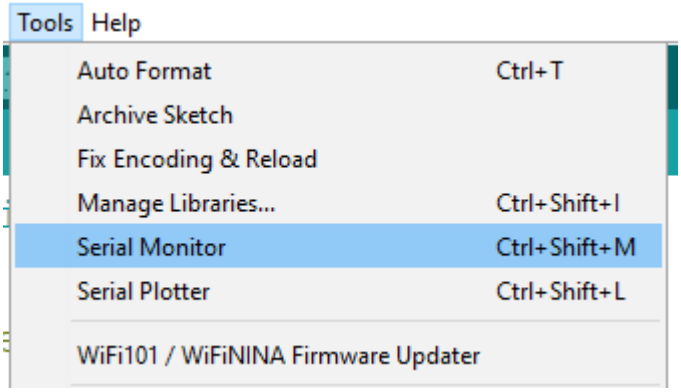


Fig. 2.22 – Meniul „Tools” opţiunea „Serial Monitor”

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 ataşat butonul (Fig. 2.23).

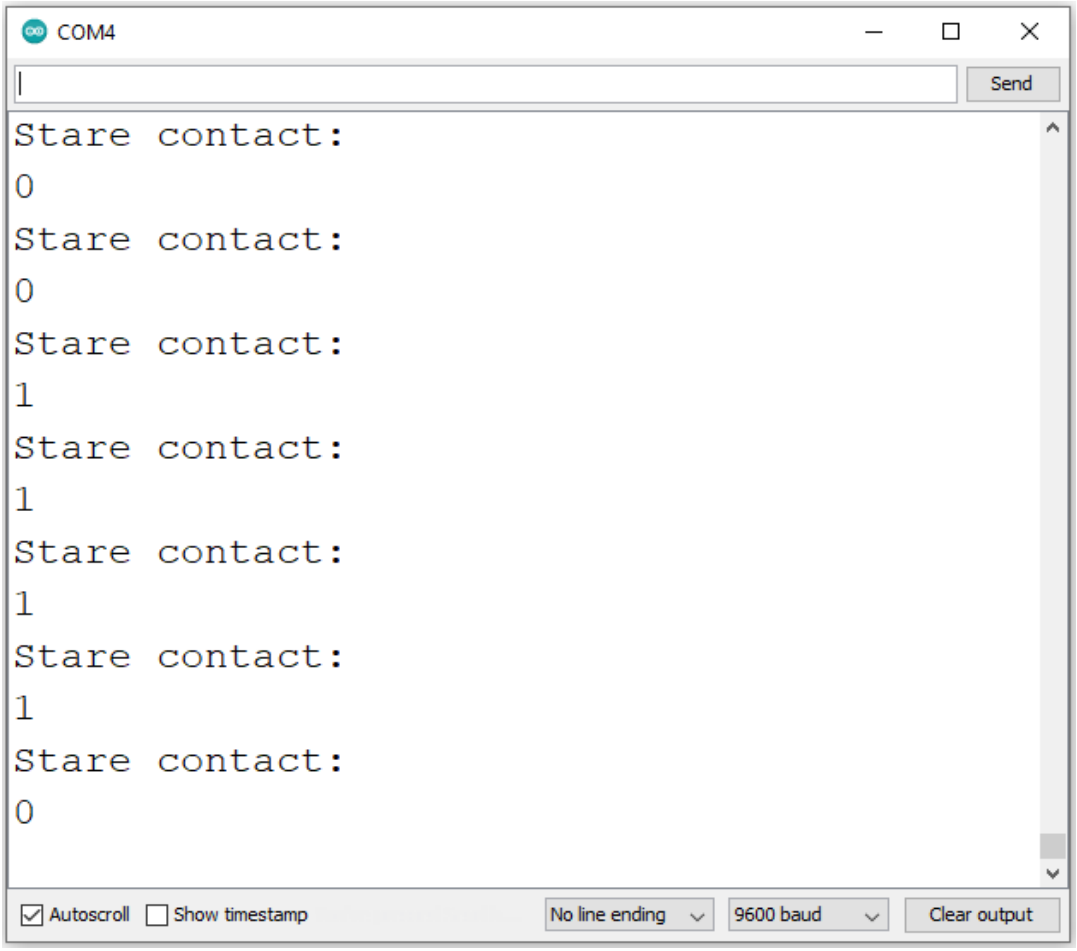


Fig. 2.23 – Consola Serial afișând mesajul implementat în cadrul aplicației 4

Pentru aceeași aplicație, din meniul „Tools” se va deschide și afișajul grafic „Serial Plotter” pentru a vizualiza evoluția stării digitale a terminalului „D5” (Fig. 2.24).

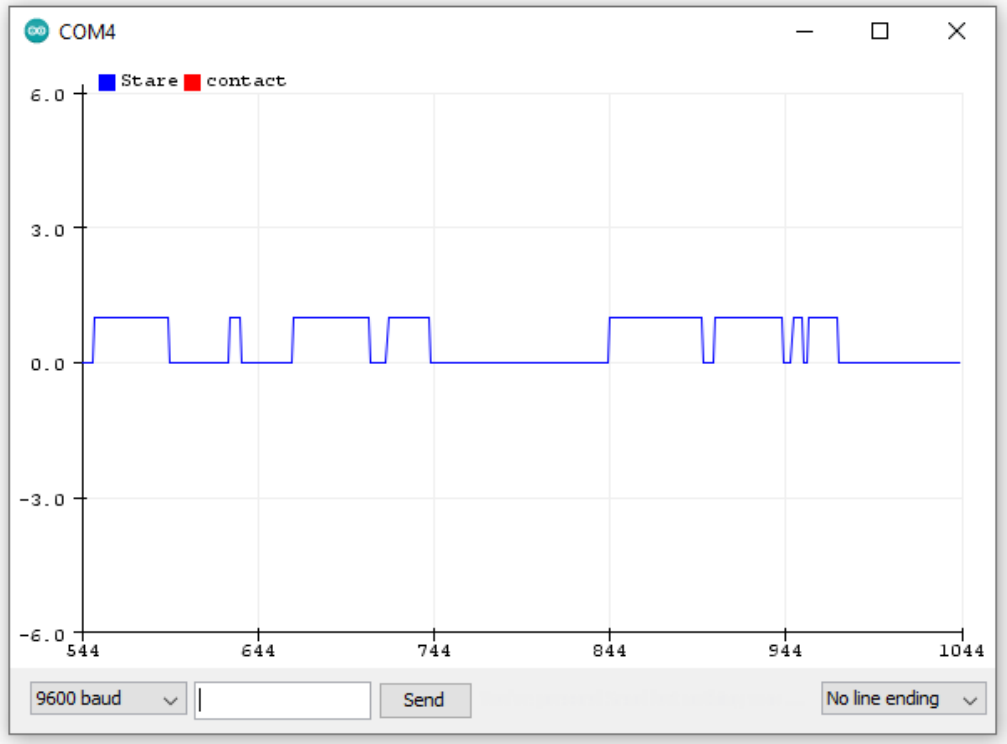


Fig. 2.24 – Fereastra de afișare grafică a valorilor vehiculate pe magistrala Serial

Montajul experimental se va realiza conform (Fig. 2.25):

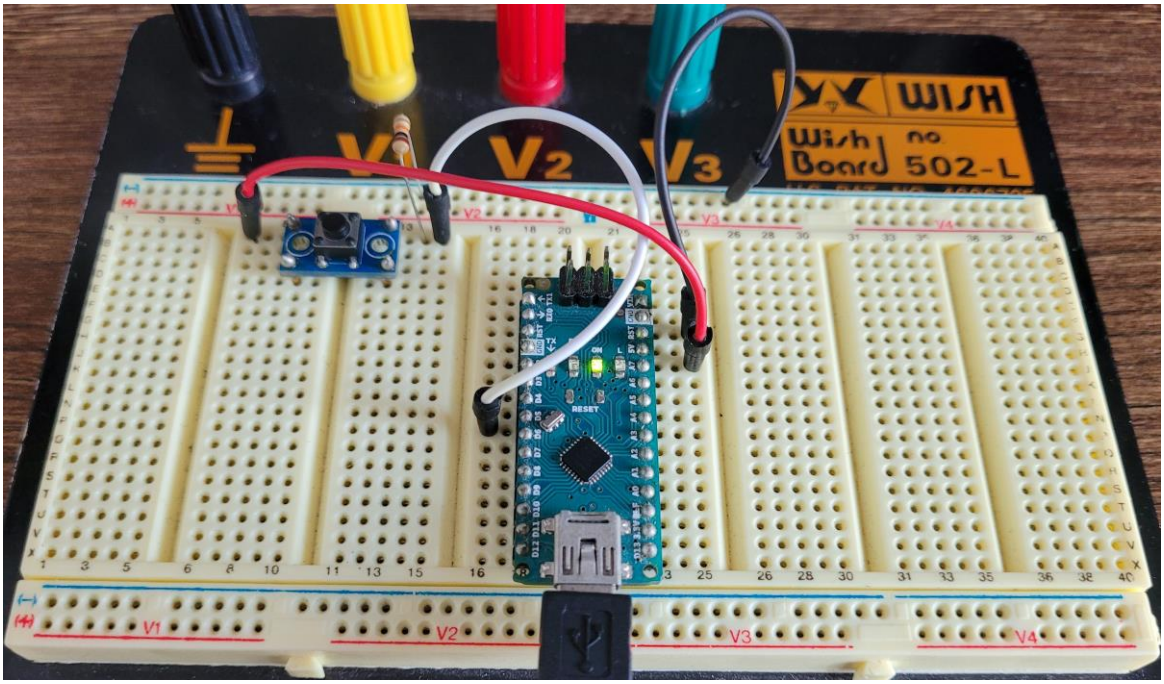


Fig. 2.25 – Montaj experimental pentru aplicația 4

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];

APLICAȚIA 5

Se va implementa circuitul conform următoarei scheme (Fig. 2.26):

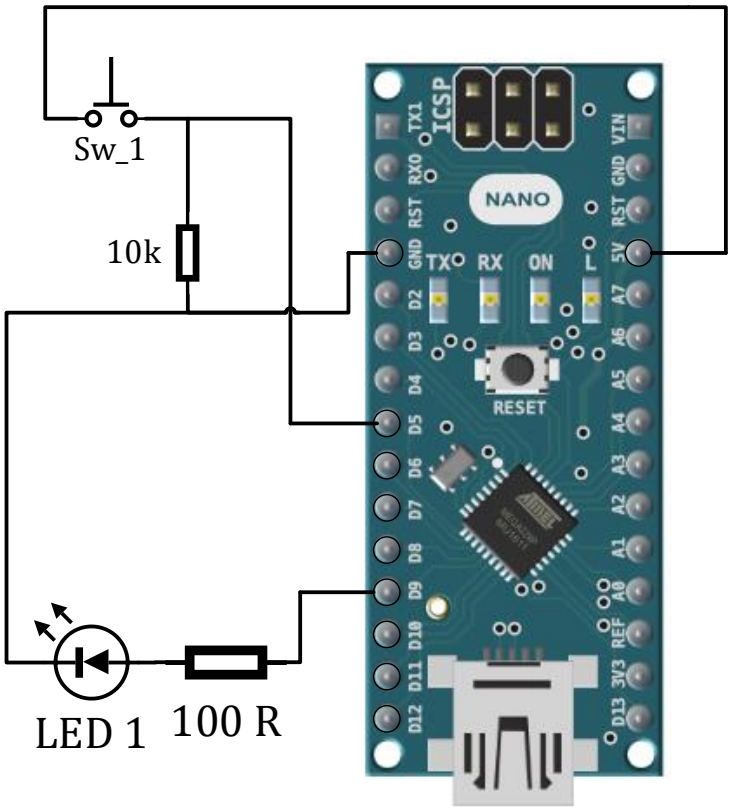


Fig. 2.26 – Schema electronică pentru implementarea circuitului specific aplicației 5 [7]

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);
}

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

Montajul experimental se va realiza conform (Fig. 2.27):

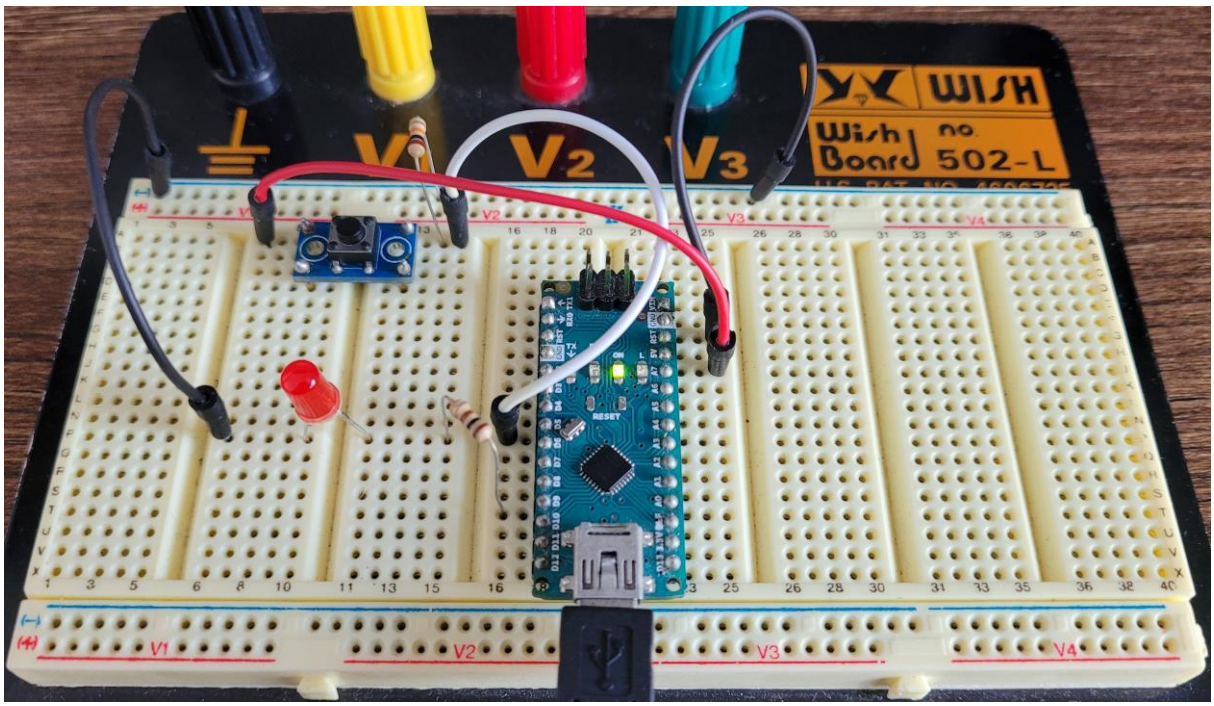


Fig. 2.27 – Montaj experimental pentru aplicația 5

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 ()”

- redirecţionarea stării intrării digitale înspre terminalul de ieşire digitală (mai precis, crearea unei funcţii de legătură între o intrare şi o ieşire digitală prin intermediul arhitecturii procesorului);

APLICAŢIA 6

Se va implementa circuitul conform următoarei scheme (Fig. 2.28):

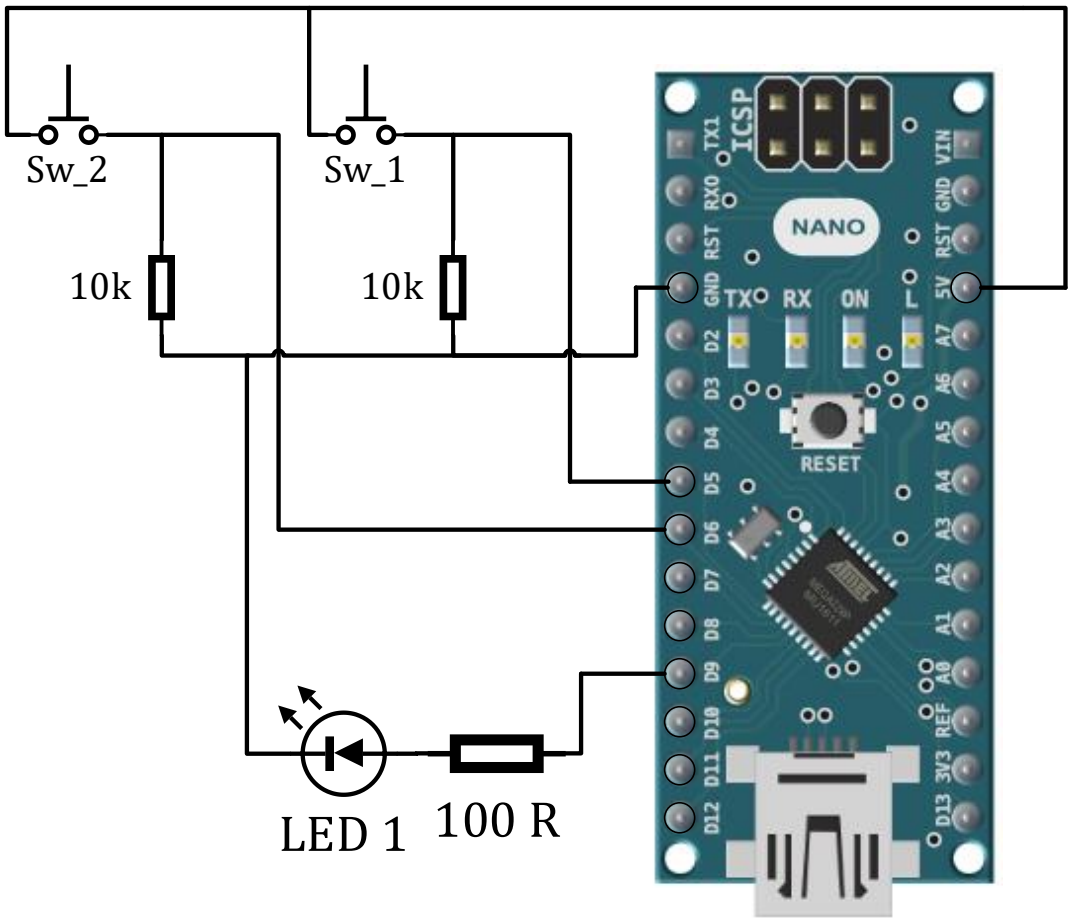


Fig. 2.28 – Schema electronică pentru implementarea circuitului specific aplicaţiei 6 [7]

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);
}

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. 2.29):

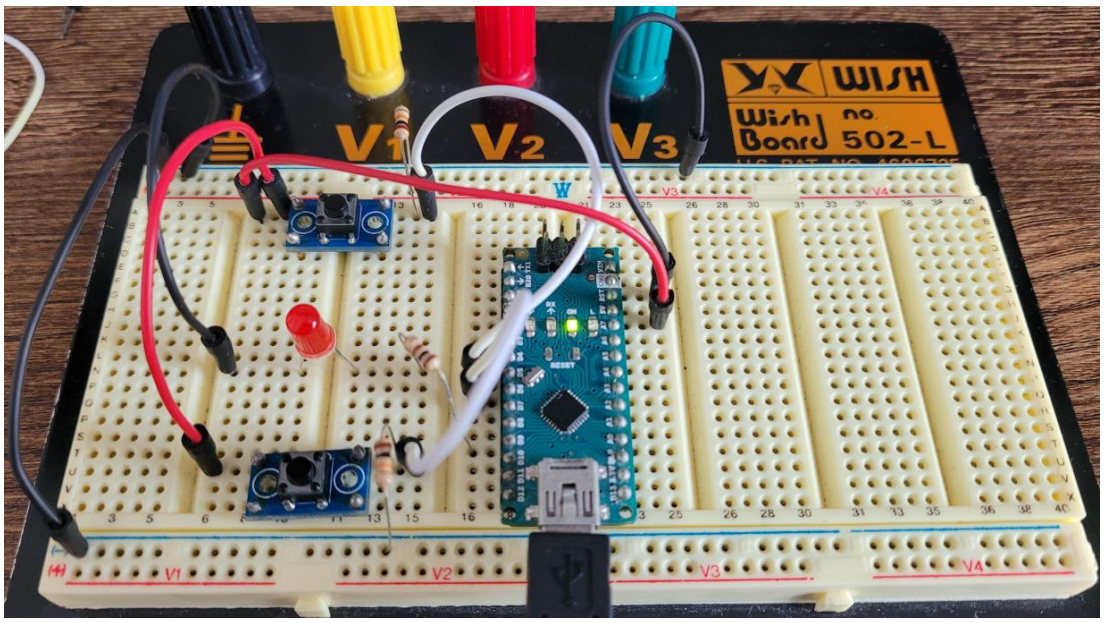


Fig. 2.29 – Montaj experimental pentru aplicația 6

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";

V. 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 crearea unor asocieri (în cadrul 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.

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/

Tema de studiu nr. 3 – Manipularea intrărilor analogice

I. SCOPUL TEMEI

- prezentarea modului de funcţionare a convertorului analog – digital sau numeric [1] [2]
- prezentarea modului de funcţionare a convertorului numeric sau digital – analog [2]
- prezentarea modului de funcţionare a unităţii pentru generare a impulsurilor [1] [2]
- implementarea aplicaţiilor specifice procesării semnalelor analogice [1] [2] [3]

II. INTRODUCERE

Microcontrolerul ATMega 328P [1] din cadrul platformei de dezvoltare Arduino Nano conferă posibilitatea achiziţionării şi prelucrării atât a semnalelor digitale cât şi a **semnalelor analogice**. Terminalele marcate pe cablajul imprimat cu indicativul „Ax” (unde „x” reprezintă numărul de ordine), reprezintă un set de **intrări analogice** (Fig. 3.1).

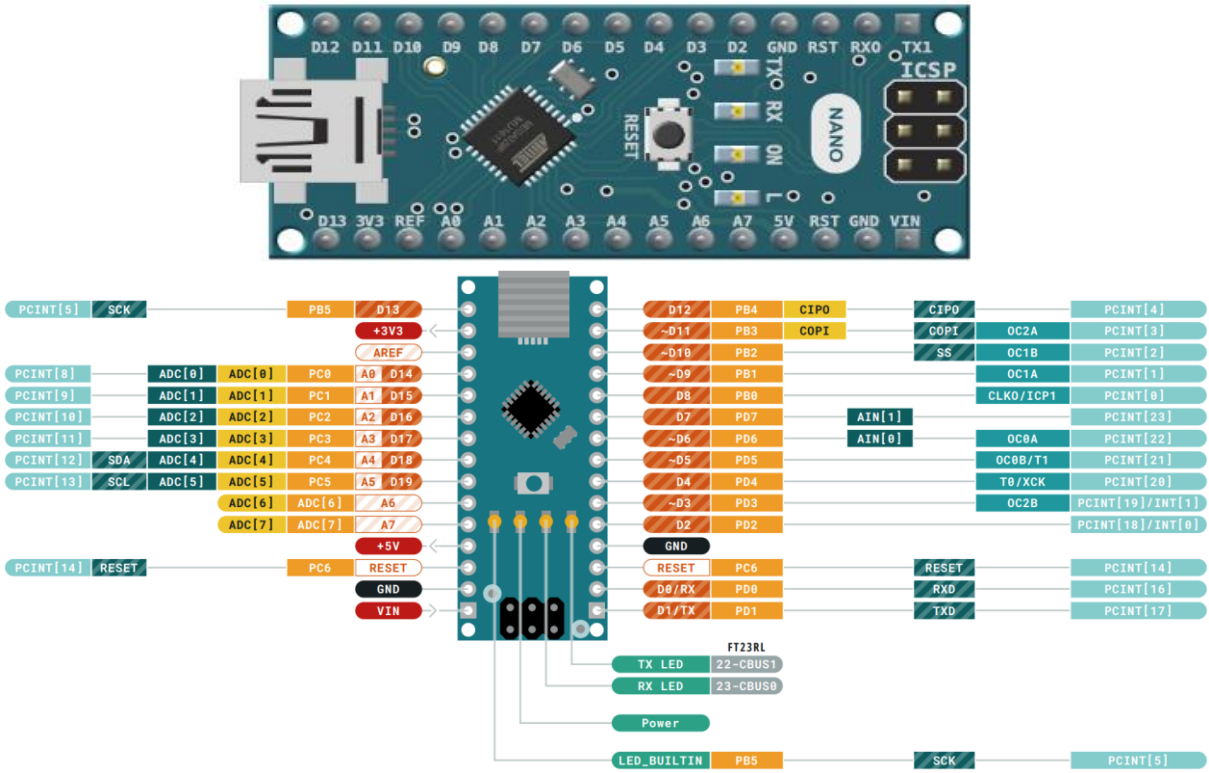


Fig. 3.1 – Harta funcţiilor specifice terminalelor platformei Arduino Nano [4]

Etajul din cadrul arhitecturii care permite achiziţionarea semnalelor analogice, poartă denumirea de **convertor analog – digital CAD** (eng. Analog to Digital Converter – ADC). Prin intermediul acestuia, microcontrolerul ATMega 328P poate măsura semnale de tensiune variabile în timp cu amplitudinea cuprinsă între 0 şi 5 [V].

III. ASPECTE TEORETICE

Într-un circuit electronic, semnalele analogice sunt reprezentate ca şi o tensiune cu amplitudine variabilă în timp (Fig. 3.2).

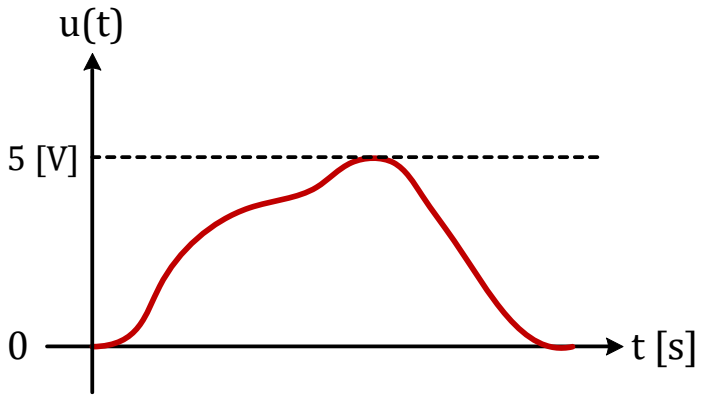


Fig. 3.2 – Semnal analogic [3]

Semnalele de tensiune variabile în timp provin în mare parte de la traductoare. Rolul unui traductor este de a transforma o mărime de natură neelectrică într-o mărime de natură electrică precum tensiune sau curent cu amplitudine variabilă. Există şi situaţia în care mărimea neelectrică măsurată de traductor este proporţională cu un anumit parametru de circuit precum: rezistenţa electrică, capacitatea, inductivitatea sau impedanţa echivalentă. Prin urmare, există deci, două clase de traductoare:
- **traductoare active** (care nu necesită alimentare deoarece produc tensiune sau curent);
- **traductoare pasive** (care necesită alimentare deoarece se bazează pe variaţia unui parametru de circuit precum rezistenţa, capacitatea sau inductivitatea);

În vederea adaptării semnalului de măsură la sistemul de achiziţie există de asemenea două categorii de circuite sau etaje intermediare:
- circuite formatoare de semnal pentru traductoarele active (ex. amplificator);
- circuite adaptoare de impedanţă pentru traductoarele pasive (ex. filtre pasive);

Cel mai simplu circuit de adaptare a impedanţei şi atenuare a semnalului este **divizorul de tensiune** (Fig. 3.3).

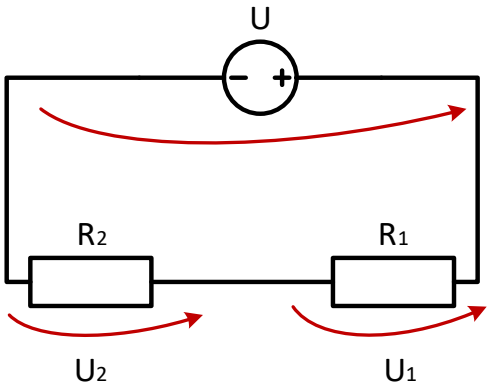


Fig. 3.3 – Divizorul de tensiune [3]

Există următoarele relaţii de calcul pentru tensiune (în cazul divizorului):

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

Traductoarele pasive pot fi ataşate la microcontroler prin intermediul divizorului de tensiune având o rezistenţă variabilă (ex. montaj în semi-punte sau potenţiometric). Rezistenţa variabilă reprezintă însuşi traductorul (ex. termistor) (Fig. 3.4).

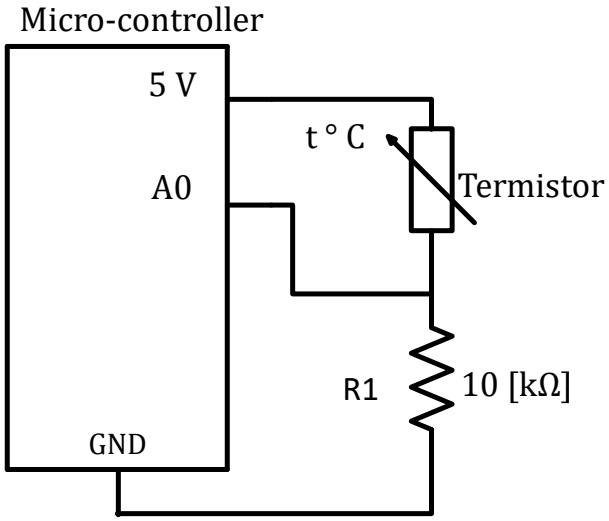


Fig. 3.4 – Ataşarea traductoarelor pasiv-rezistive la microcontroler [3]

Cel mai simplu etaj electronic din componenţa arhitecturii microcontrolerului ATMega 328P, care poate prelucra semnalul analogic este **comparatorul** [1] (Fig. 3.5).

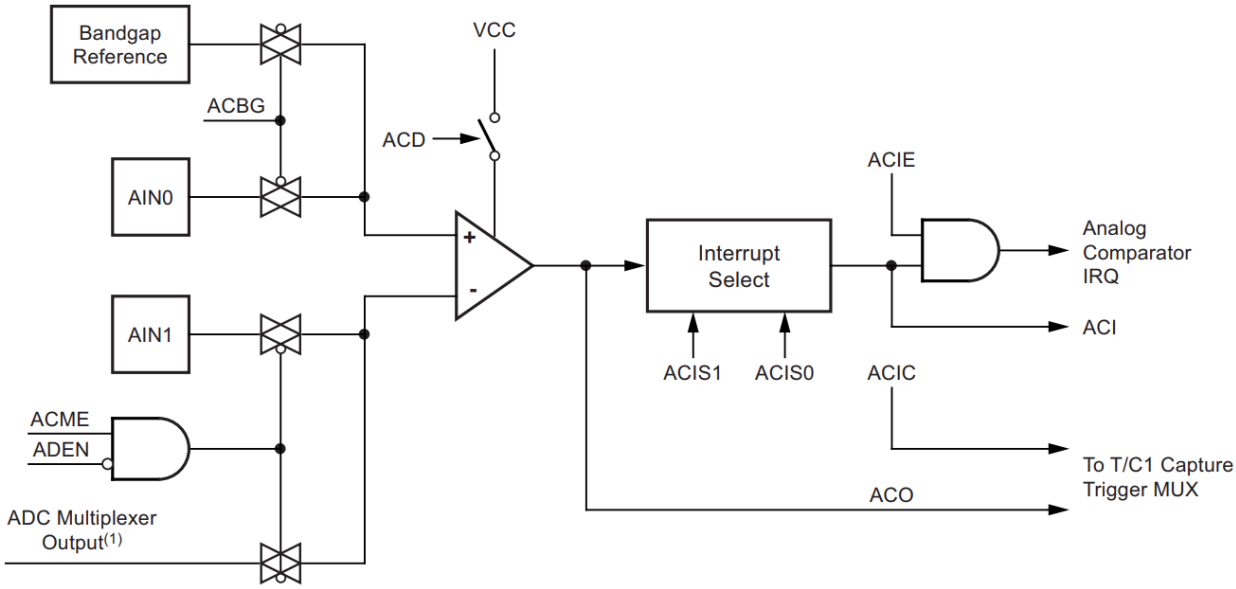


Fig. 3.5 – Comparatorul din cadrul arhitecturii microcontrolerului ATMega 328P [1]

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) (Fig. 3.6).

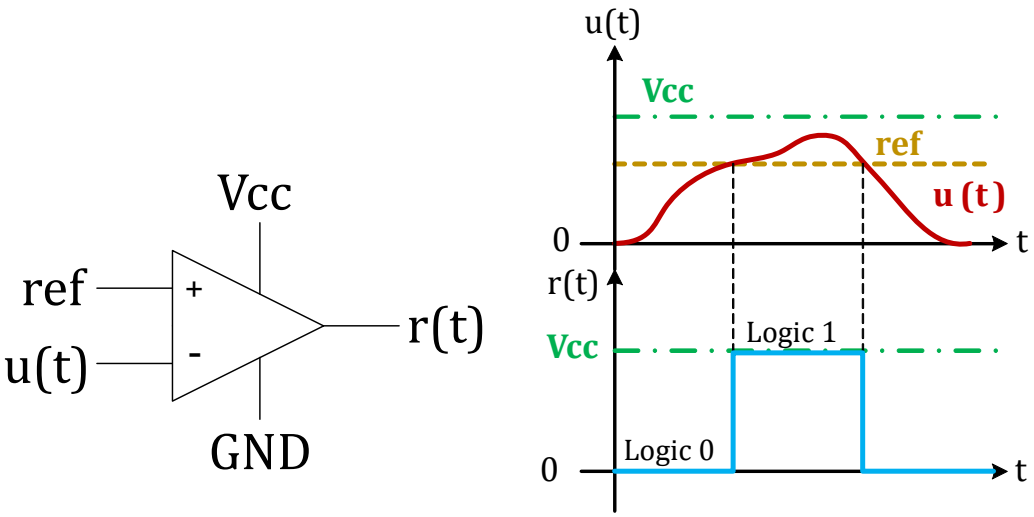


Fig. 3.6 – Principiul de funcționare al comparatorului electronic [3]

Convertorul analog – digital (eng. Analog to Digital Converter – ADC) reprezintă o serie progresivă de comparatoare. Pe baza numărului de comparatoare se poate determina 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 dacă există sau nu căderea de tensiune prestabilită la bornele rezistenței corespunzătoare (Fig. 3.7).

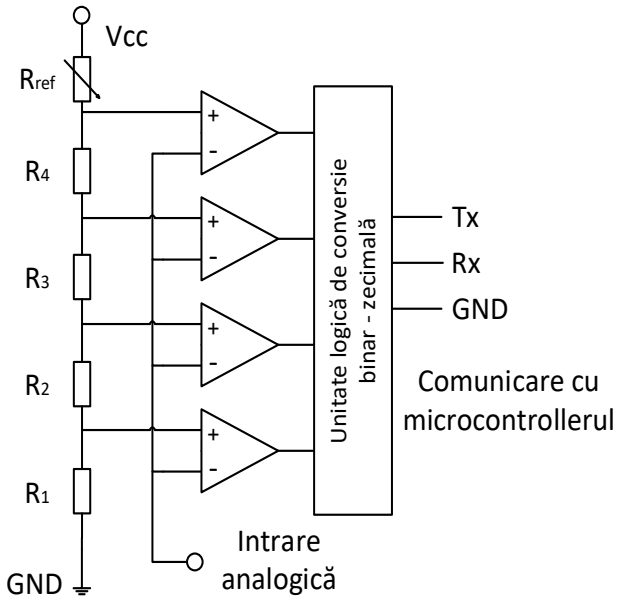


Fig. 3.7 – Schema electronică principială a convertorului analog digital [3]

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) (Fig. 3.8).

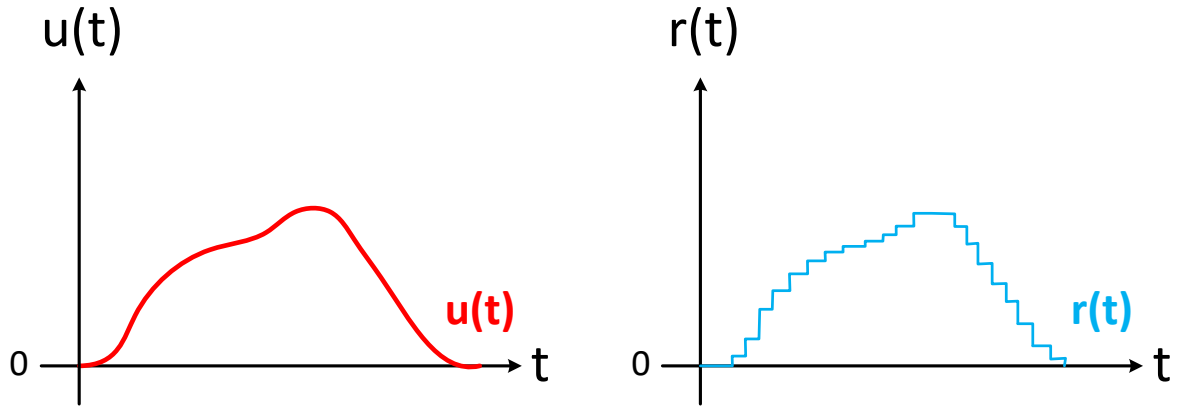


Fig. 3.8 – Semnalul analogic discretizat [3]

Fidelitatea semnalului înregistrat depinde de **rezoluția convertorului** analog – digital. Rezoluția depinde de **numărul de comparatoare** din componența convertorului analog – digital. **Precizia** convertorului analog – digital depinde de **rezoluție** și **tensiunea de alimentare**. Spre exemplu:

Fie, un convertor analog – digital având rezoluția pe 10 biți, alimentat cu tensiunea de referință 5 [V]. Să se calculeze precizia convertorului:

$$U_{ref} = 5\ [V],\ \ r = 10\ [bit] = (2^{10}) - 1 = 1023$$

$$x = \frac{U_{ref}}{r} = \frac{5}{1023} \approx 0{,}004\ [V] \rightarrow 4\ [mV\ /\ treaptă];$$

Unde, „U_{ref}” reprezintă tensiunea de referință, „r” rezoluția, iar „x” precizia convertorului analog – digital. **Precizia** reprezintă **constanta de calibrare în tensiune** a convertorului.

Fie „x”, precizia unui convertor analog – digital, „r” rezoluția egală cu 10 biți, iar tensiunea de referință 5 [V]. Să se determine **nivelul de tensiune măsurat cu ajutorul convertorului analog - digital**, pentru care **indicația numerică** în format zecimal din registrul convertorului ar fi ADC = 345:

$$U_{măs} = ADC \cdot x = ADC \cdot \frac{U_{ref}}{r}$$

$$ADC = 345,\ \ U_{măs} = ADC \cdot x = ADC \cdot \frac{U_{ref}}{r} = 345 \cdot \frac{5}{1023} \approx 1{,}38\ [V]$$

Unde, „ADC” reprezintă indicația numerică din registrul convertorului analog - digital, iar „$U_{măs}$” reprezintă tensiunea măsurată pe intrarea analogică.

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 (Fig. 3.9).

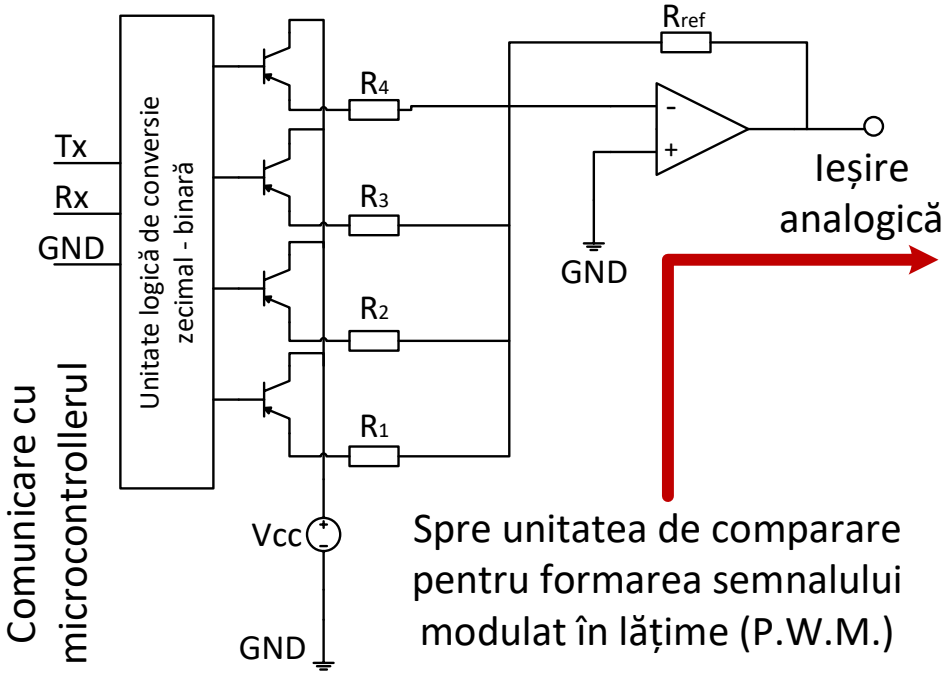


Fig. 3.9 – Schema bloc principială a convertorului digital – analog [2]

În cadrul unor arhitecturi de microcontroler sau procesor digital de semnal, nivelul de tensiune rezultant „DAC" [2] al convertorului digital – analog este utilizat în vederea **generării semnalului dreptunghiular modulat în lățime** „r(t)" (eng. Pulse Width Modulation – PWM). Convertorul digital – analog reprezintă în acest sens, o **sursă de tensiune cu amplitudine controlabilă în mod digital**. **Tensiunea generată** de către convertor, poate fi furnizată înspre un **etaj comparator** împreună cu un **semnal triunghiular** „Timer" cu **frecvență constantă** (Fig. 3.10 ~ 3.11). Semnalul triunghiular reprezintă „**unda purtătoare**", iar tensiunea cu amplitudine variabilă reprezintă „**unda modulatoare**". Semnalul rezultant, este de natură **digitală** având **frecvența** și **amplitudinea constantă** iar **lățimea variabilă** (Fig. 3.11).

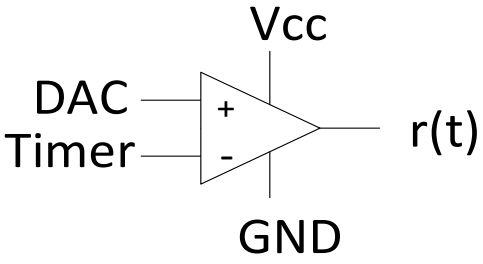


Fig. 3.10 – Etaj comparator intern pentru generarea semnalului modulat în lățime [2] [3]

Semnalul triunghiular va fi generat de un circuit numărător sau temporizator (eng. counter ~ timer).

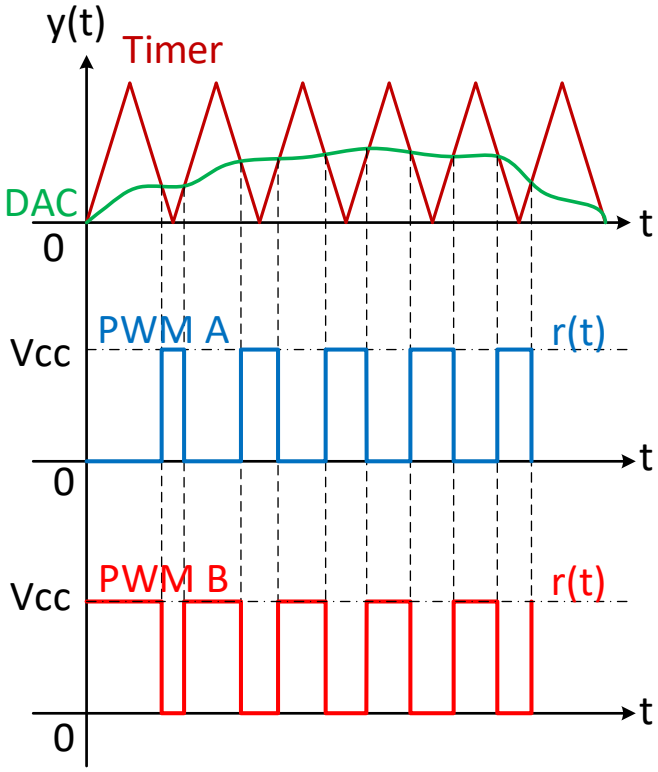


Fig. 3.11 – Generarea semnalului modulat în lățime [1] [2] [3]

Fidelitatea pentru reprezentarea factorului de umplere al semnalului modulat în lățime depinde de **rezoluția numărătorului** și **rezoluția convertorului digital – analog** (dacă semnalul este generat pe baza unei strategii de comparare). Spre exemplu:

Fie indicația numerică a registrului „PWM = 125". Pentru o unitate de generare a semnalului modulat în lățime având rezoluția „r" pe 8 biți, să se determina factorul de umplere „d":

$$r = 8\,[bit] = (2^8) - 1 = 255$$

$$x = \frac{1}{r} = \frac{1}{255} \approx 0{,}0039$$

$$d_{[\%]} = PWM \cdot \frac{1}{r} = 125 \cdot \frac{1}{255} \approx 0{,}49 \rightarrow 49\,[\%]$$

Unde „PWM" este indicația numerică a factorului de umplere (între [0 – 255]), iar „$d_{[\%]}$" factorul de umplere exprimat procentual cu fracție subunitară. Motivul pentru care factorul de umplere are valori subunitare constă în faptul că, amplitudinea undei purtătoare (triunghiulară) este maxim 1.

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);
}
```

- instrucțiunea de constrângere și scalare „map ()” (restrângerea unui domeniu de variație la un anumit interval de finit 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);
```

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

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

IV. IMPLEMENTAREA APLICAȚIILOR

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];
- 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;

În vederea studierii aspectelor şi conceptelor teoretice se propune implementarea următoarelor aplicaţii pe baza platformei Arduino Nano:
- Determinarea rezoluţiei convertorului analog – digital (cu afişare în monitorul serial);
- Determinarea preciziei convertorului analog – digital şi a valorii de tensiune măsurată;
- Determinarea temperaturii cu ajutorul traductorului LM-35;
- Implementarea unui comparator numeric cu prag digital reglabil (funcţia „if”);
- Implementarea unei coloane luminoase indicatoare de nivel (funcţia „map”);
- Generarea unui semnal dreptunghiular modulat în lăţime (funcţia „map”);

APLICAŢIA 1

Se va implementa circuitul conform următoarei scheme (Fig. 3.12):

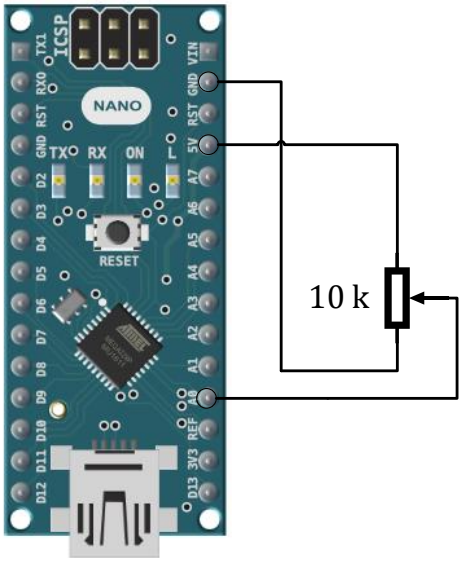


Fig. 3.12 – Schema electronică pentru implementarea aplicaţiei 1 şi 2 [3] [4]

Se va realiza următorul montaj experimental (Fig. 3.13):

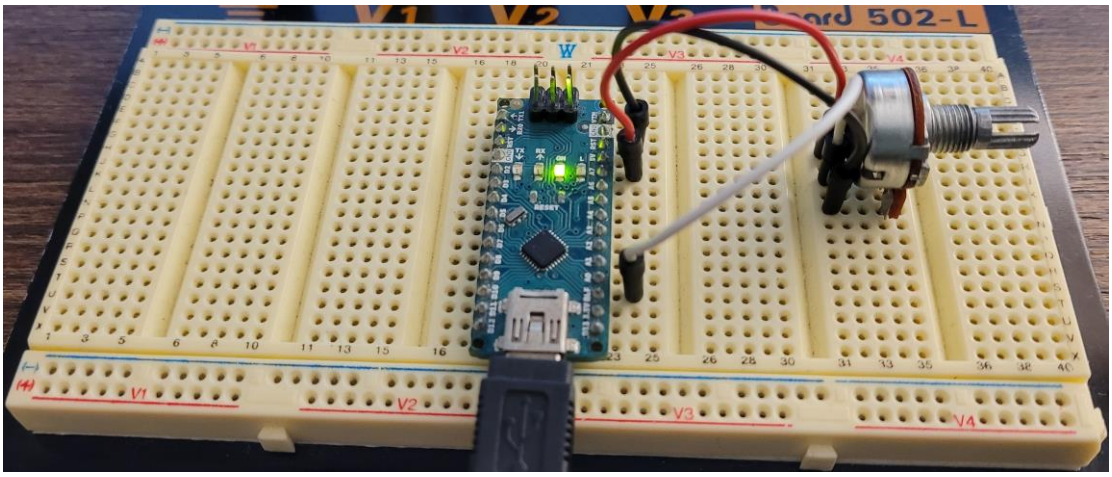


Fig. 3.13 – Montajul experimental specific aplicaţiei 1 şi 2

Se va implementa următorul cod program:

```
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);
}
```

În urma implementării codului program la nivelul microcontrolerului ATMega 328P, se va deschide consola Serial (meniul „Tools” --> „Serial Monitor”) (Fig. 3.14).

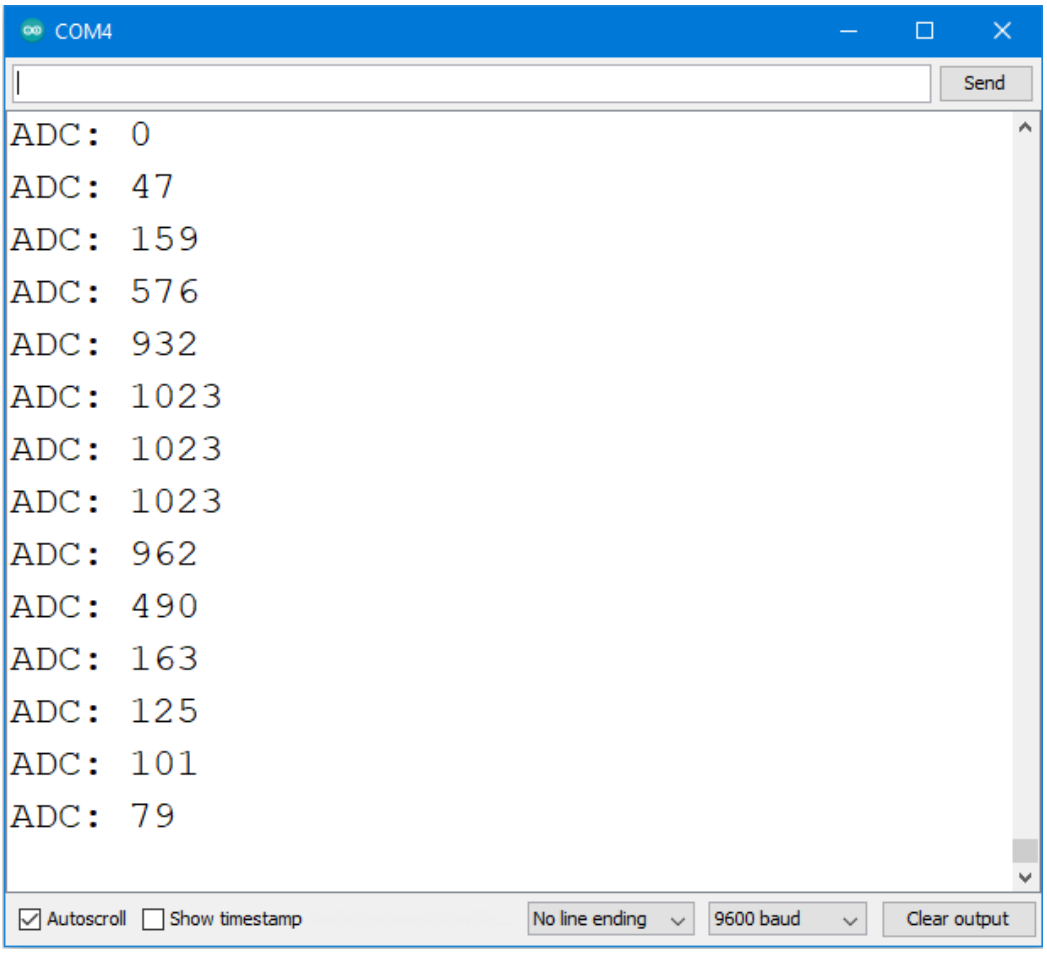


Fig. 3.14 – Consola Serial – Rezultatul conversiei analog – digital

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: ”;

Se constată faptul că intervalul de variație al rezultatului conversiei analog – digitale este cuprins între [0 1023] (Fig. 3.14). Valorile obținute corespund unei rezoluții de 10 biți, astfel valoarea maximă în format zecimal poate fi determinată pe baza relației:

$$r = 10\ [bit] = (2^{10}) - 1 = 1023$$

APLICAȚIA 2

Utilizând același montaj ca și în cazul aplicației anterioare (Fig. 3.12 și 3.13), se va determina (pe baza următoarelor relații) atât precizia convertorului analog – digital cât și valoarea nivelului de tensiune înregistrat în mod dinamic de către convertor.

$$x = \frac{U_{ref}}{r} \quad \rightarrow \quad U_{măs} = ADC \cdot \frac{U_{ref}}{r}$$

Se va implementa următorul cod program:

```
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);
}
```

În urma implementării codului program se va deschide consola Serial (Fig. 3.15). Precizia convertorului analog – digital este aproximativ 4 [mV / pas], iar tensiunea măsurată variază în intervalul [0 5] [V], în situaţia în care nivelul tensiunii de referinţă sau nivelul tensiunii de alimentare este egal cu 5 [V], (Fig. 3.15).

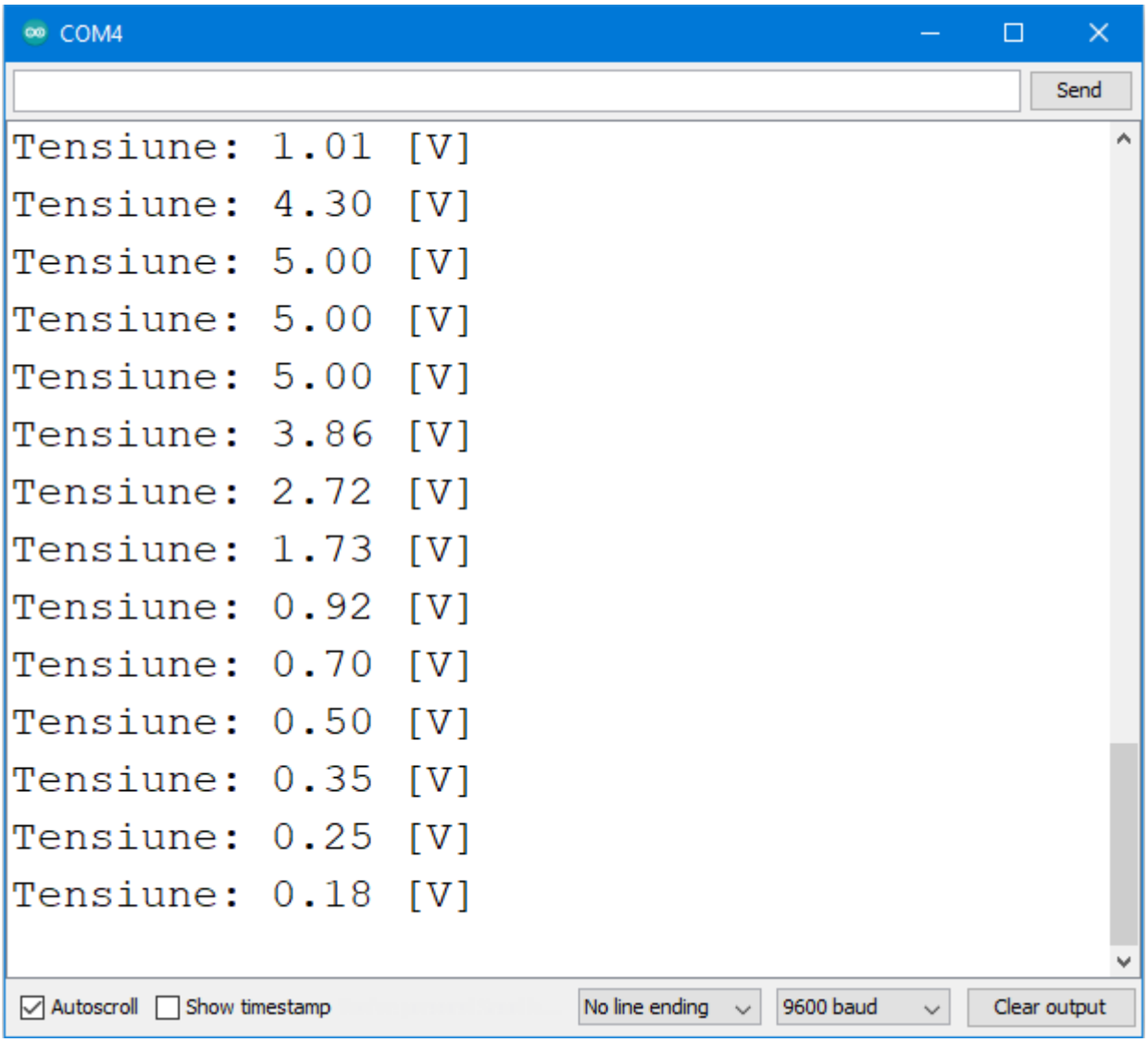


Fig. 3.15 – Consola Serial – Afişarea nivelului de tensiune măsurat de convertor

Implementarea aplicaţiei nr. 2 presupune:

- declararea unei constante de tip număr întreg „analog_pin” având ca şi valoare „0”;
- iniţializarea unei variabile de tip număr întreg „ADC_val” cu valoarea „0”;
- iniţializarea unei variabile de tip fracţionar „U” cu valoarea „0.00”;
- iniţializarea comunicaţiei Serial la viteza de transfer 9600 [b/s];
- preluarea valorii zecimale rezultante în urma procesului de conversie analog – digital;
- determinarea tensiunii de măsură pe baza preciziei convertorului analog – digital;
- afişarea nivelului de tensiune în consola Serial însoţit de mesajul „Tensiune: ____ [V] ”;

APLICAȚIA 3

Se va implementa circuitul conform următoarei scheme (Fig. 3.16):

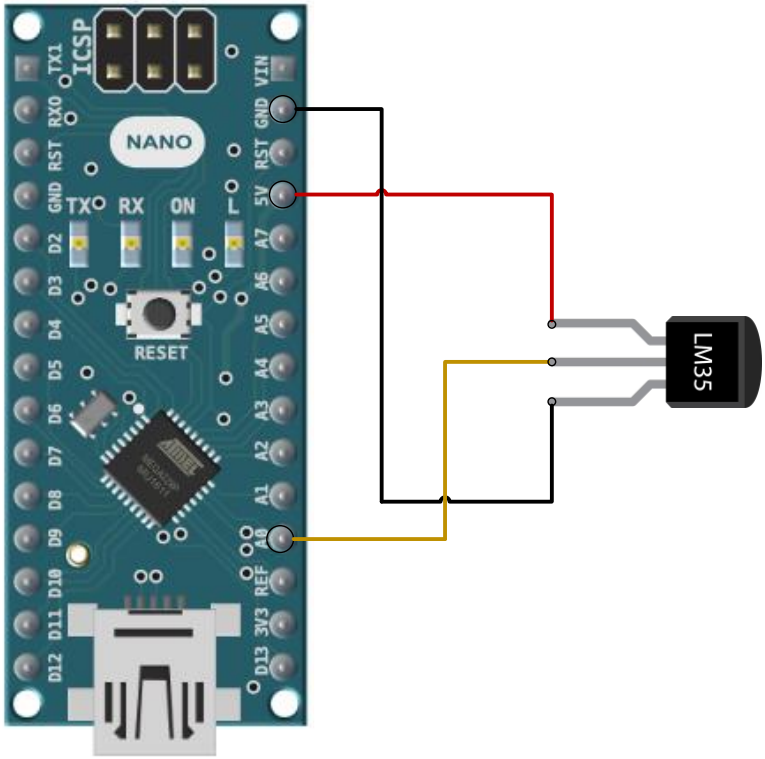


Fig. 3.16 – Schema electronică pentru implementarea circuitului specific aplicației 3 [3] [4]

Se va realiza următorul montaj experimental (Fig. 3.17):

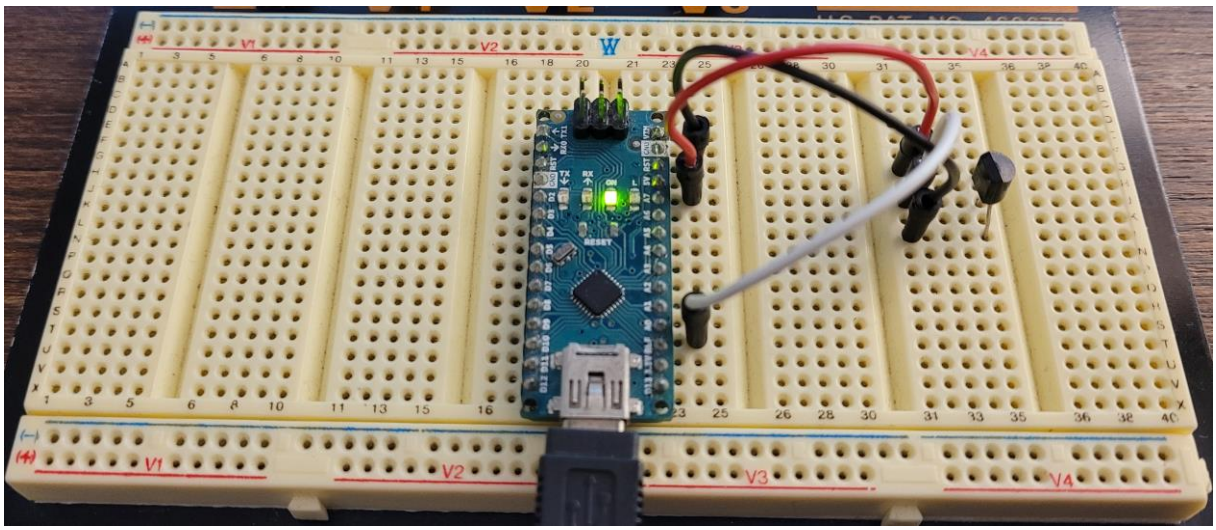


Fig. 3.17 – Montajul experimental specific aplicației 3

Pe baza datelor de catalog a fost identificată constanta de calibrare 10 [mV / *C].

Se va implementa următorul cod program:

```
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);
}
```

Pe baza instrumentului grafic de analiză (Serial Plotter), se va observa evoluția în timp a temperaturii înregistrate de traductorul LM-35 (Fig. 3.18):

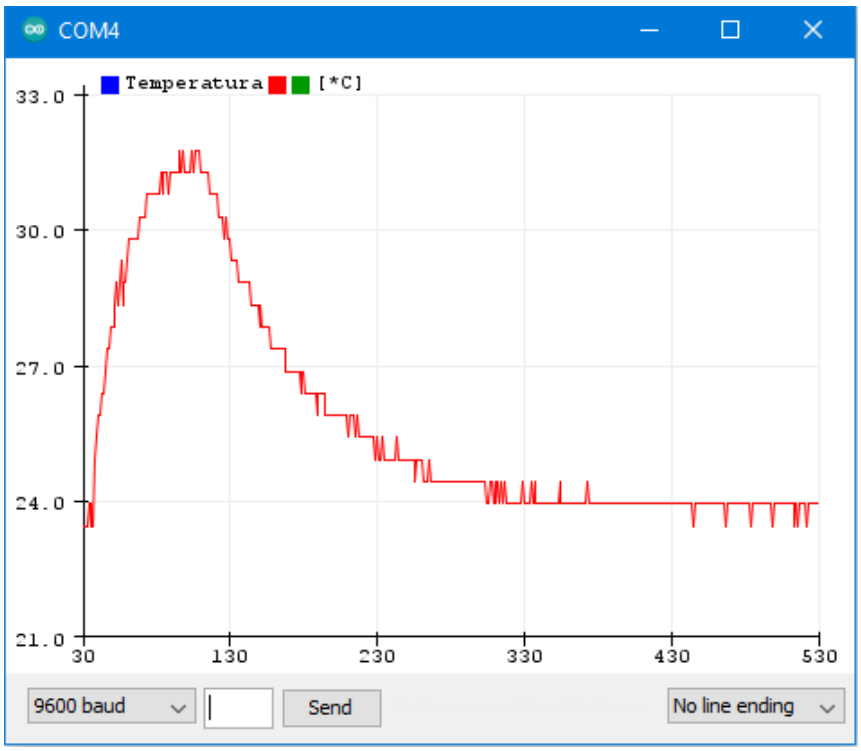


Fig. 3.18 – Variația temperaturii înregistrate de traductorul LM-35 în raport cu timpul

Se va observa de asemenea modul de transmitere în consola Serial al mesajului care indică temperatura ambientală (Fig. 3.19).

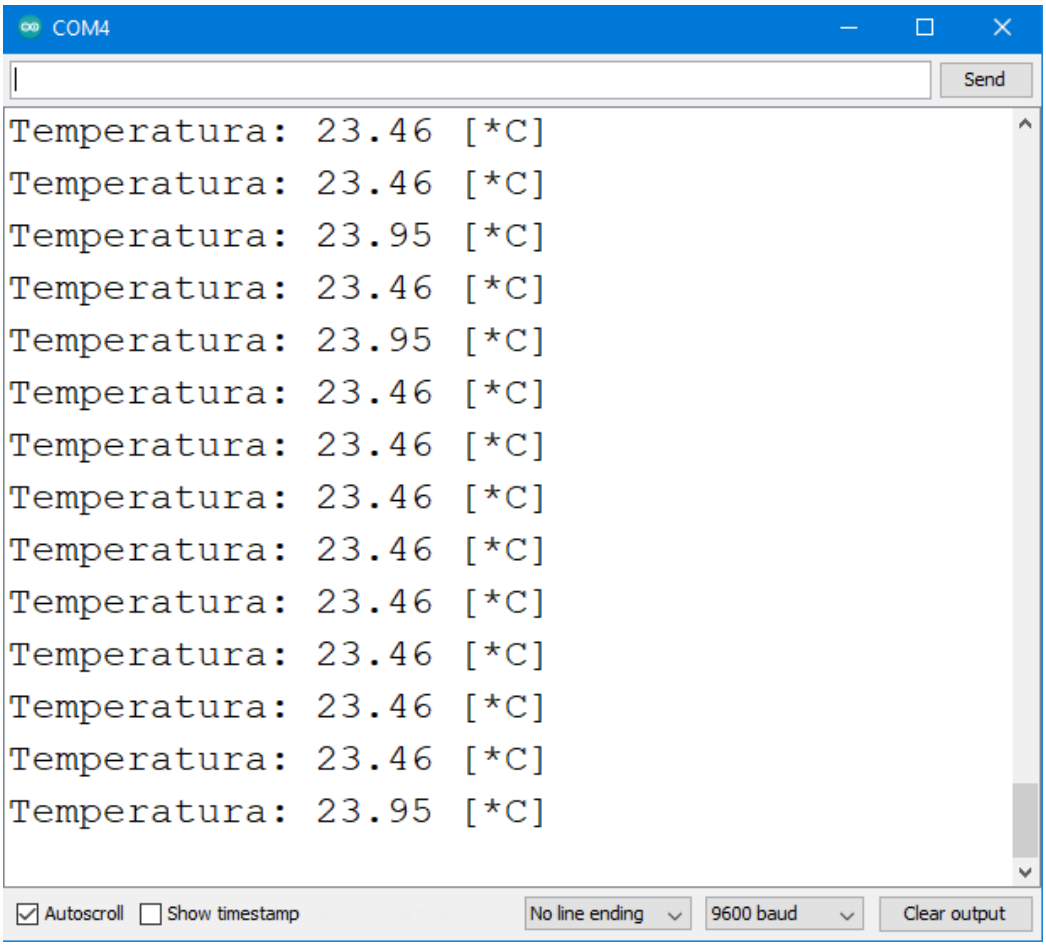


Fig. 3.19 – Consola Serial – Variaţia temperaturii în raport cu timpul

Se constată faptul că, faţă de valoarea reală a temperaturii, există o serie de abateri, care pot fi percepute ca şi erori de măsurare sau „zgomote” induse în semnalul de măsură. Din acest motiv, va fi necesar ca semnalul de măsură achiziţionat să fie filtrat cu ajutorul algoritmului de mediere dinamică (determinarea în mod dinamic a mediei aritmetice pe baza punctelor de măsură înregistrate).

Implementarea aplicaţiei nr. 3 presupune:

- declararea unei constante de tip număr întreg „analog_pin” având ca şi valoare „0”;
- iniţializarea unei variabile de tip număr întreg „ADC_val” cu valoarea „0”;
- iniţializarea unei variabile de tip fracţionar „U” cu valoarea „0.00”;
- iniţializarea unei variabile de tip fracţionar „temp” cu valoarea „0.00”;
- iniţializarea comunicaţiei Serial la viteza de transfer 9600 [b/s];
- 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 şi a constantei de calibrare;
- afişarea valorii temperaturii în consola Serial însoţită de mesajul „Temperatura: _ [*C] ”;

APLICAȚIA 4

Se va implementa circuitul conform următoarei scheme (Fig. 3.20):

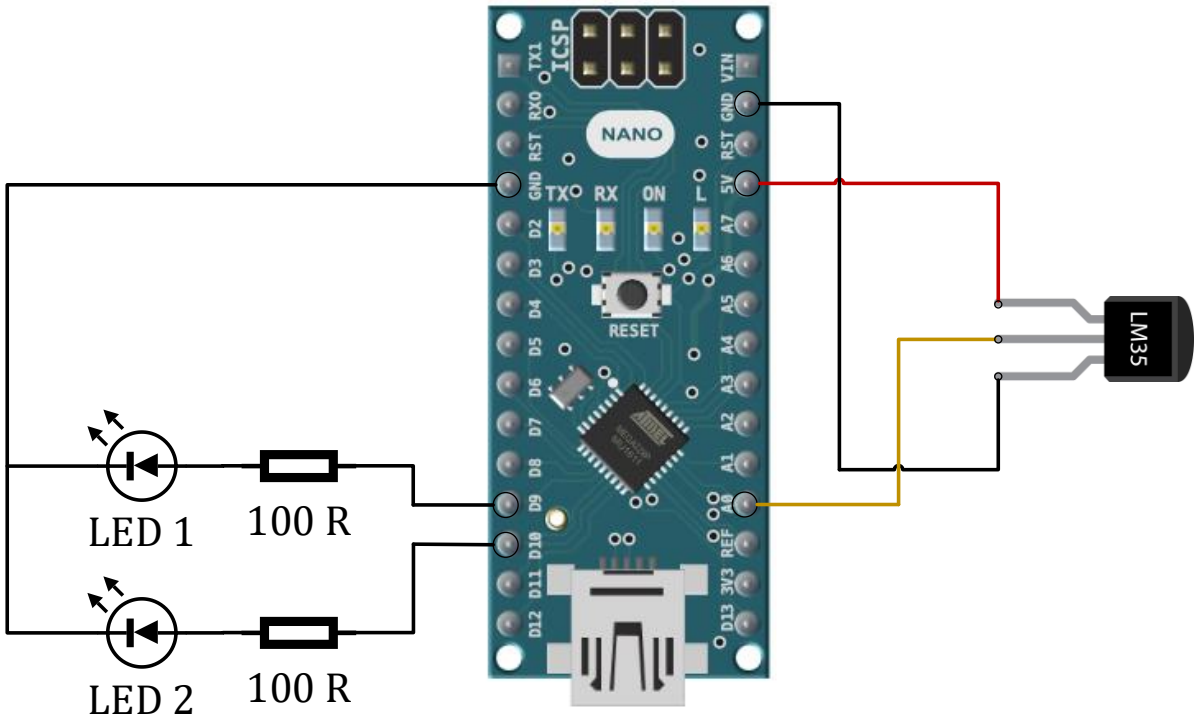


Fig. 3.20 – Schema electronică pentru implementarea circuitului specific aplicației 4 [3] [4]

Se va realiza următorul montaj experimental (Fig. 3.21):

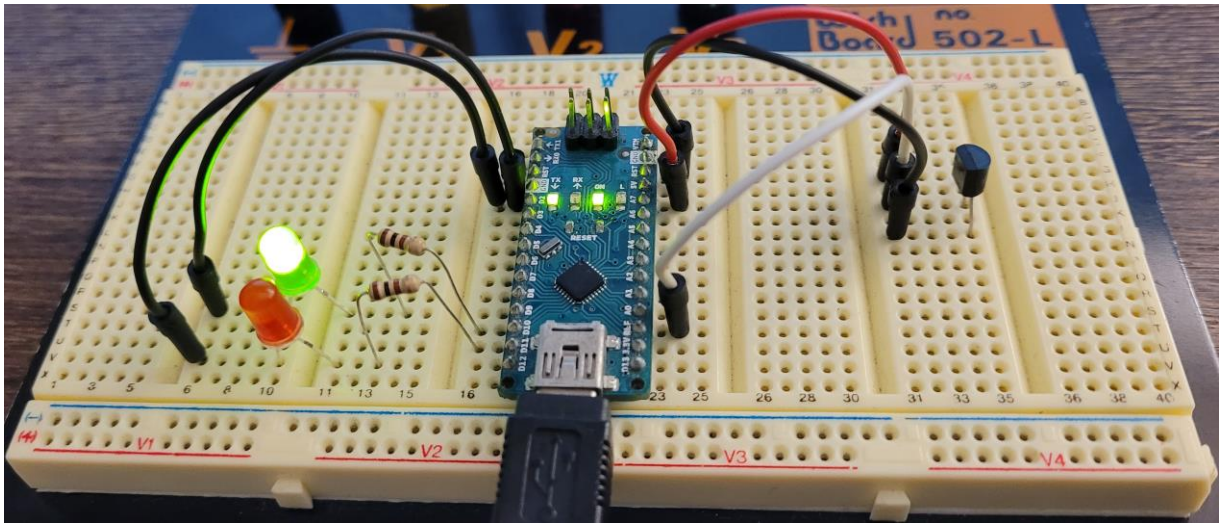


Fig. 3.21 – Montajul experimental specific aplicației 4

Se va implementa următorul cod program:

```
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);
}
```

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 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;
- compararea nivelului temperaturii actuale cu valoarea de prag impusă inițial;
- afișarea în consolă atât a valorii actuale cât și a pragului de temperatură;

Se va deschide consola Serial pentru vizualizarea valorilor (Fig. 3.22).

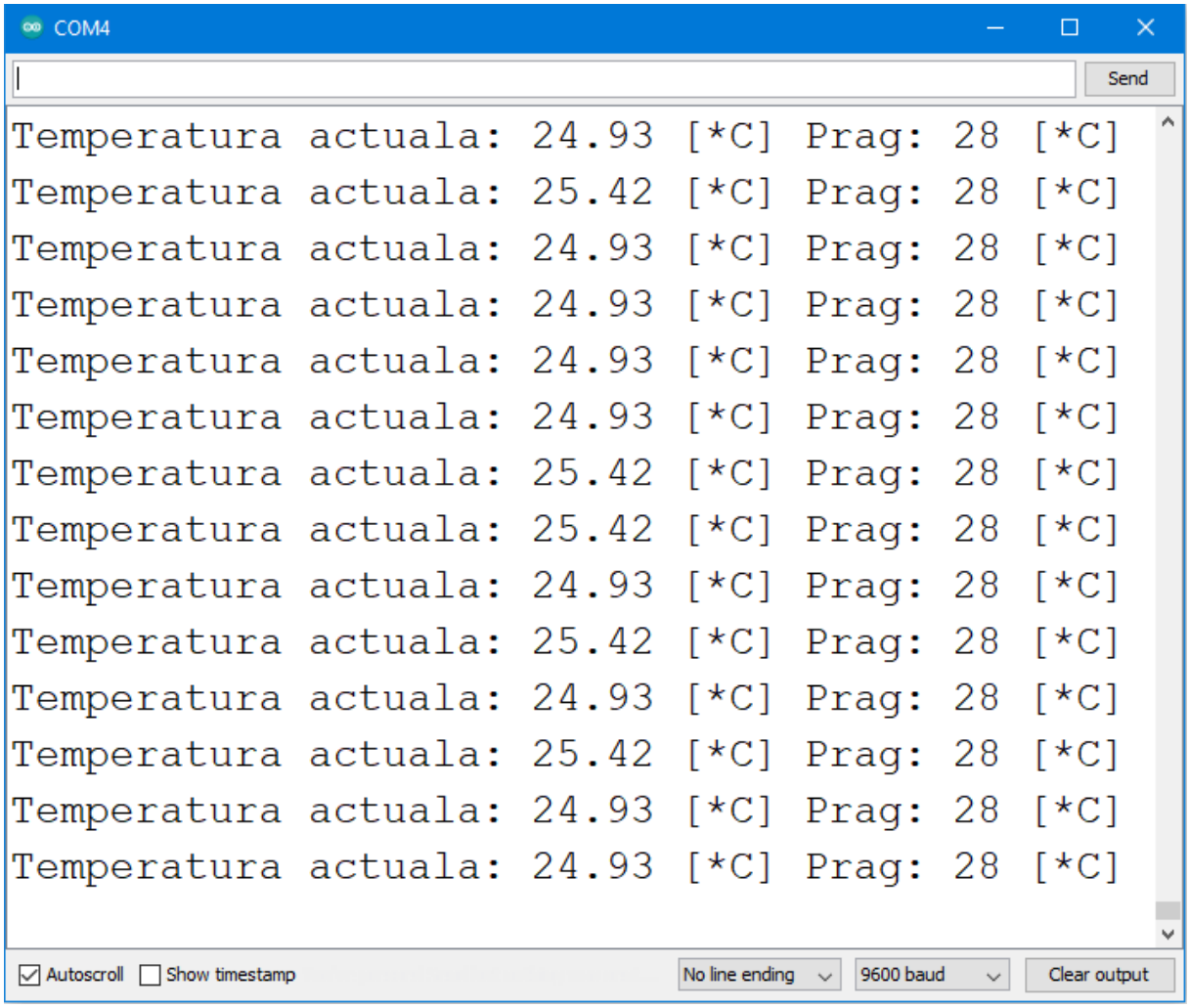


Fig. 3.22 – Consola Serial – Variația temperaturii în raport cu valoarea de prag

Pentru vizualizare se va utiliza de asemenea instrumentul pentru analiză grafică a variației temperaturii ambientale în raport cu o valoare de prag impusă (Fig. 3.23).

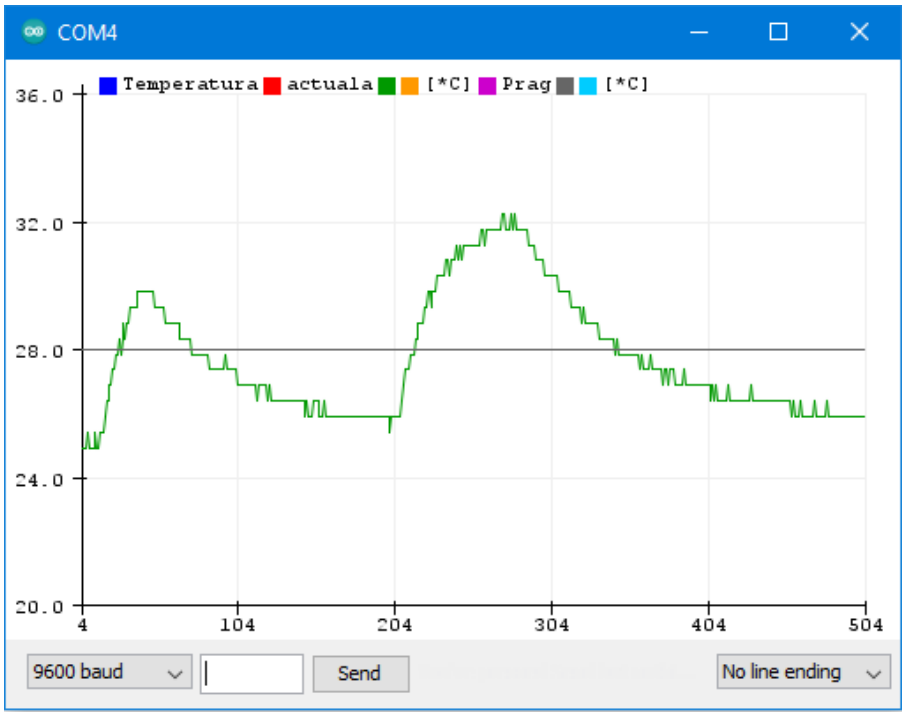


Fig. 3.23 – Variația temperaturii în raport cu valoarea de prag impusă inițial

APLICAȚIA 5

Se va implementa circuitul conform următoarei scheme (Fig. 3.24):

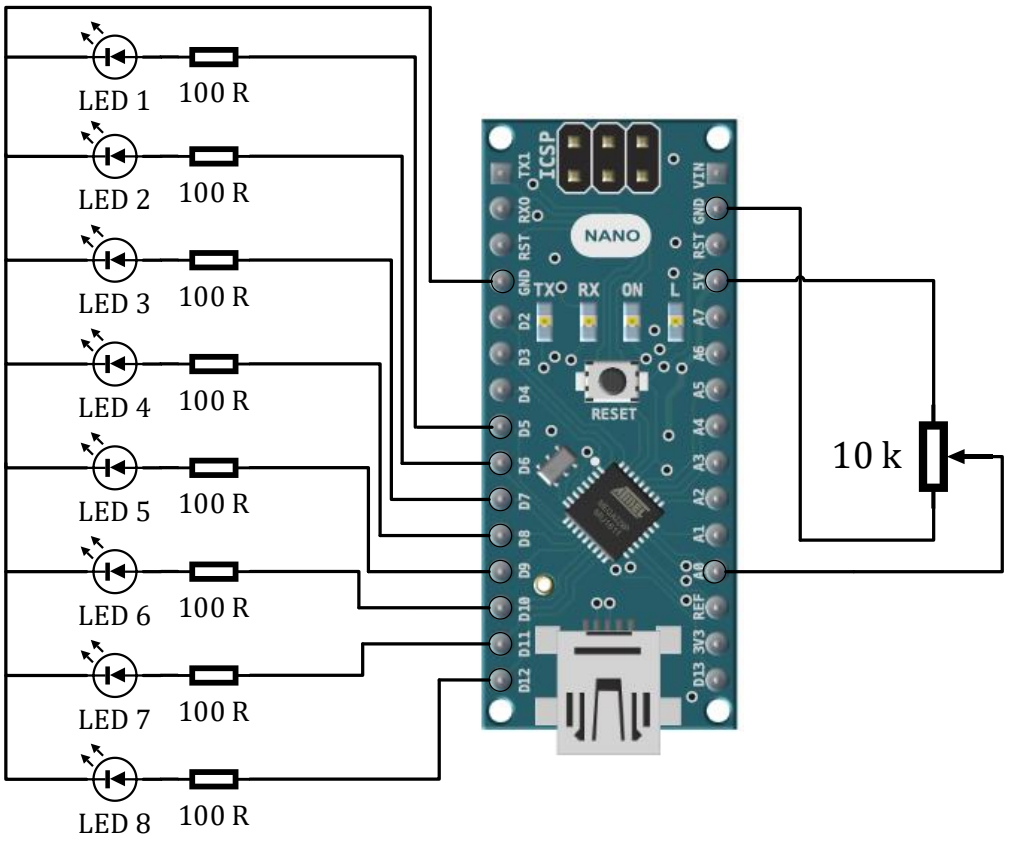


Fig. 3.24 – Schema electronică pentru implementarea circuitului specific aplicației 5 [3] [4]

Se va realiza următorul montaj experimental (Fig. 3.25):

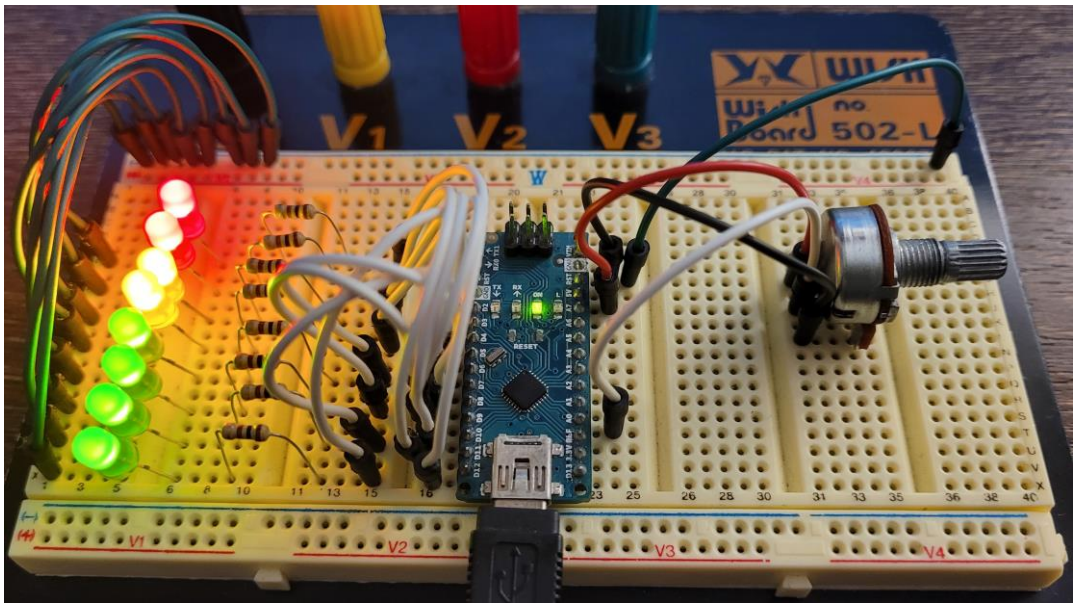


Fig. 3.25 – Montajul experimental specific aplicației 5

Se va implementa următorul cod program:

```
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);
    }
  }
}
```

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;

OBSERVAȚIE: Montajul și codul program realizat în cadrul aplicației nr. 5 poate fi adaptat la orice tip de senzor analogic, care furnizează o tensiune cu nivel variabil. De asemenea, numărul de elemente semnalizatoare poate fi modificat (pentru coloane indicatoare cu un număr mai mare de elemente).

APLICAȚIA 6

Se va implementa circuitul conform următoarei scheme (Fig. 3.26):

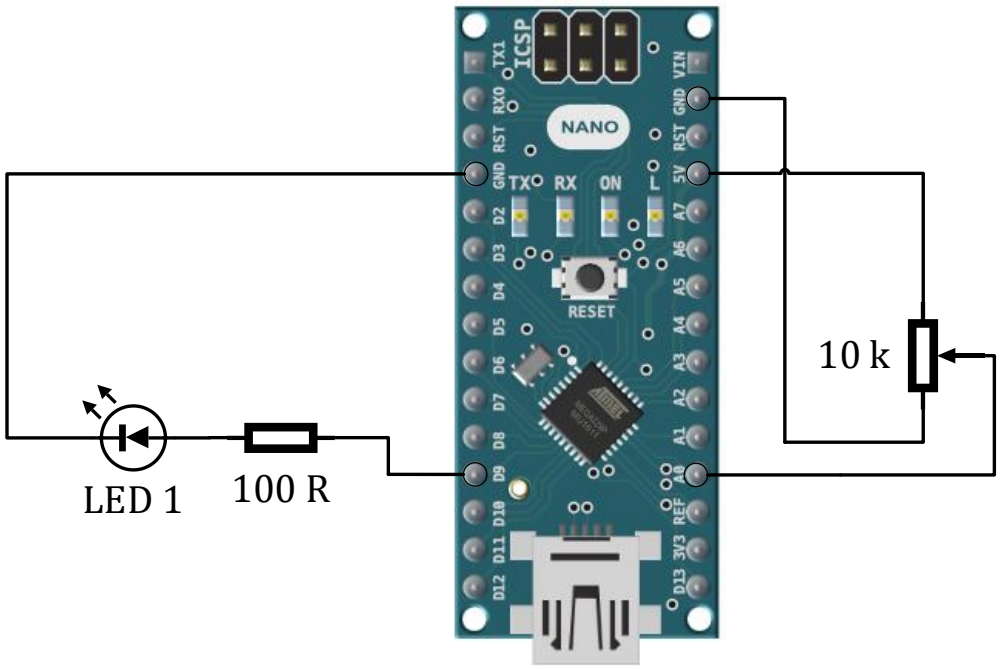


Fig. 3.26 – Schema electronică pentru implementarea circuitului specific aplicației 6 [3] [4]

Se va realiza următorul montaj experimental (Fig. 3.27):

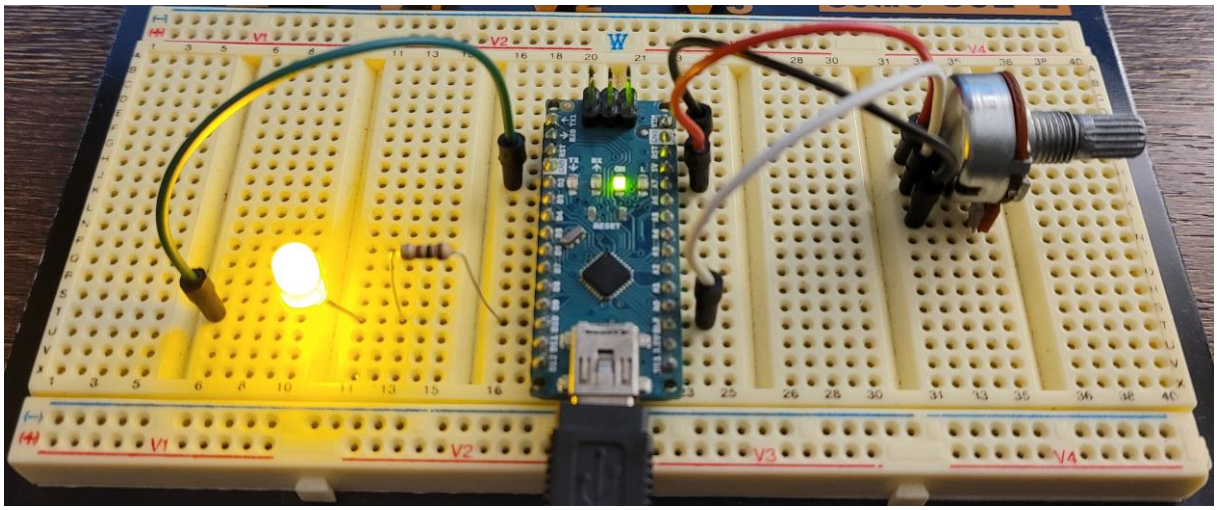


Fig. 3.27 – Montajul experimental specific aplicației 6

Se va implementa următorul cod program:

```
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);
}
```

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”;

Se va deschide consola Serial pentru a vizualiza datele furnizate de către microcontroler înspre calculator (Fig. 3.28).

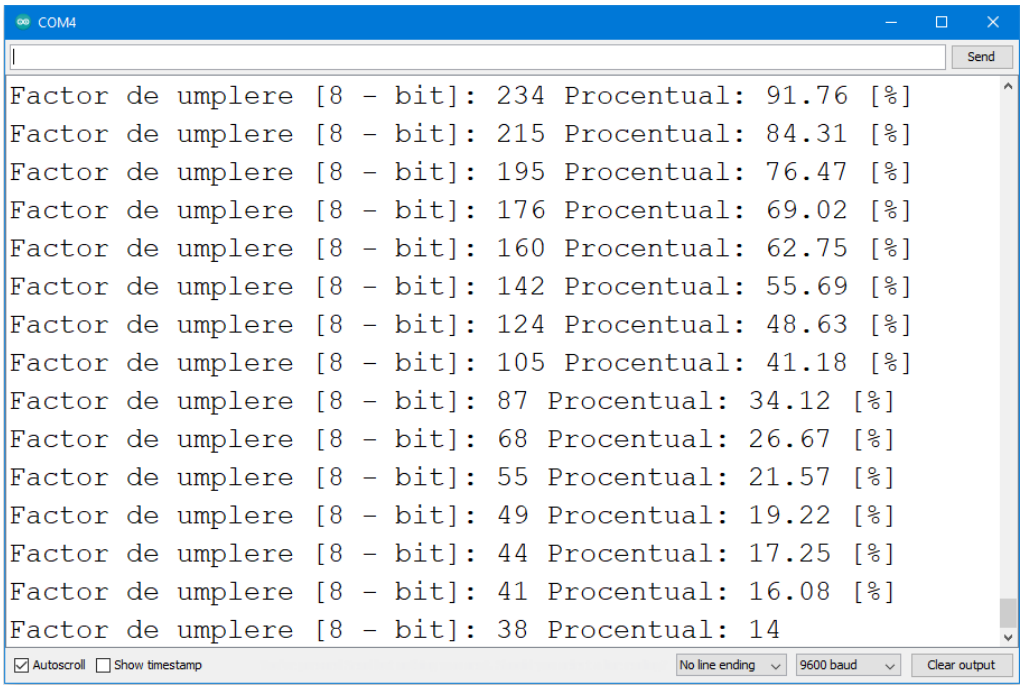


Fig. 3.28 – Afișarea factorului de umplere sau a duratei de conducție în consola Serial

V. CONCLUZII

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

VI. 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/

Tema de studiu nr. 4 – Aplicații generale de interfațare

I. SCOPUL TEMEI

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

II. INTRODUCERE

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

Conceptul de **aplicație de interfațare**, presupune implementarea unui etaj de circuit pe baza perifericelor specializate ale unui microcontroler (ex. intrări – ieșiri digitale, intrări analogice, interfețe de comunicare). Perifericele de intrare și ieșire ale microcontrolerului pot fi atașate la elemente de circuit precum: afișaje LCD, butoane, senzori sau elemente de execuție precum motoare, rezistențe sau semnalizatoare acustice. Rolul dispozitivului rezultat în urma implementării este de a **intermedia operația de reglare** a procesul tehnologic final realizată de factorul uman. Funcția îndeplinită în acest sens se numește „interfațare” (Fig. 4.1).

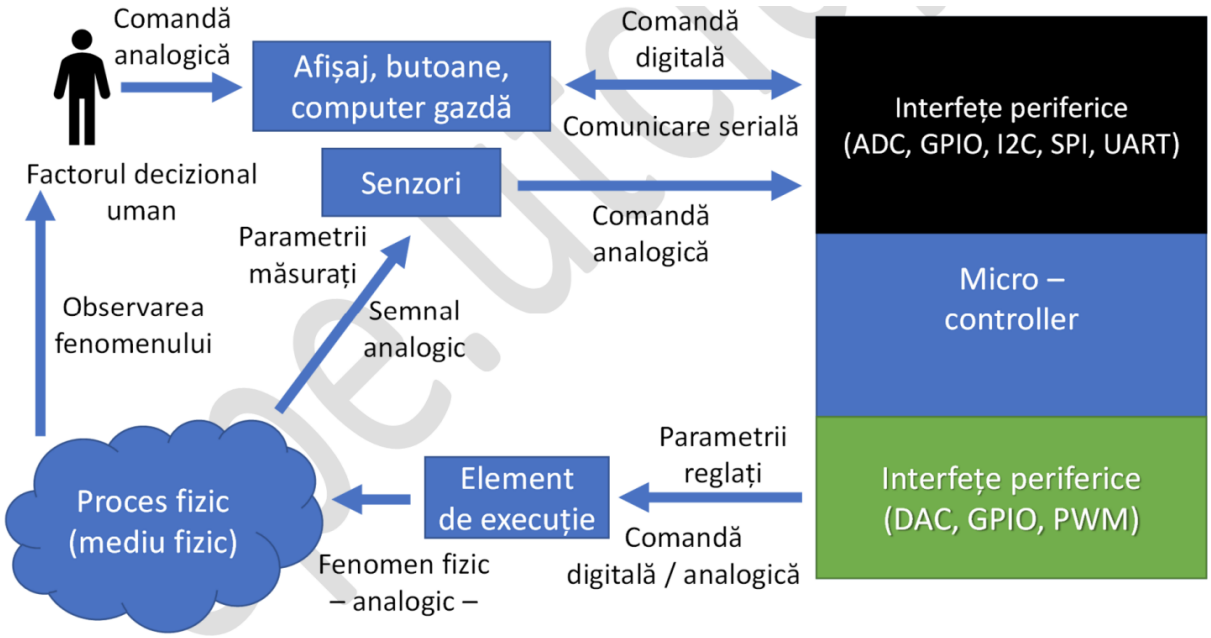


Fig. 4.1 – Funcția de interfațare [1] [2]

Există trei tipuri de aplicaţii [1] [2]:
- aplicaţii independente (cu algoritm de condiţionare proprie);
- aplicaţii dependente de elementele de interfaţare (LCD şi butoane sau calculator);
- aplicaţii hibride (comandate de utilizator dar cu algoritm de condiţionare propriu);

A. Aplicaţii independente

Acest mod de implementare se bazează pe programul propriu inscripţionat de către producătorul echipamentului în memoria microcontrolerului, iar utilizatorul nu poate modifica modul de funcţionare şi nici nu poate furniza o comandă de modificare a stării în care se află dispozitivul. Interacţiunea cu sistemul de calcul se va realiza prin intermediul senzorilor (sau mai precis al perifericelor de intrare), iar semnalul rezultant va fi furnizat către un element de execuţie prin intermediul perifericelor de ieşire.

În această categorie se încadrează aplicaţii precum: detectoare, sesizoare, elemente de semnalizare, elemente de execuţie cu algoritm condiţional propriu (Fig. 4.2).

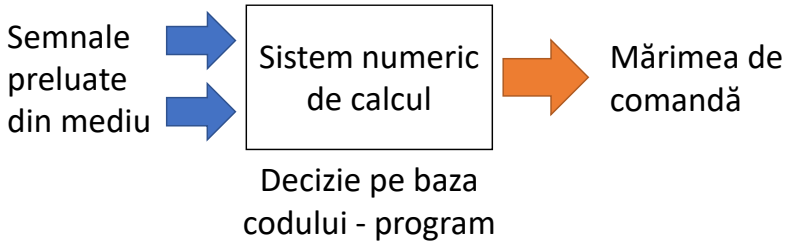


Fig. 4.2 – Aplicaţii independente [1] [2]

B. Aplicaţii dependente de elementele de interfaţare

Acest mod de implementare necesită o comandă furnizată de către operator pentru a schimba modul de funcţionare al aplicaţiei. Interacţiunea dintre operator şi dispozitiv sau procesul tehnologic final se va realiza prin intermediul elementelor de interfaţare ataşate la perifericele microcontrolerului (ex. LED ataşat la ieşirea digitală pentru semnalizare, buton ataşat la intrarea digitală pentru preluarea comenzilor, afişaj LCD pentru transmiterea mesajelor text de informare asupra stării de funcţionare).

În această categorie se încadrează aplicaţii precum: dispozitivele periferice ale computerului, panouri frontale cu afişaj şi butoane, echipamente de interacţiune umană cu procesul fizic (Human Interface Device – HID) (Fig. 3).

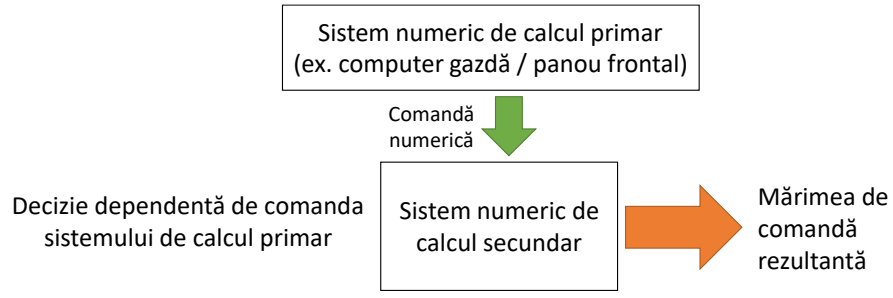


Fig. 4.3 – Aplicaţii dependente de elementele de interfaţare [1] [2]

C. Aplicaţii hibride

Acest mod de implementare are la baza funcţionării atât programul pre-implementat de către producător în memorie microcontrolerului cât şi un program pentru interacţiune cu procesul din exterior (prin intermediul unui alt sistem de calcul sau prin intermediul propriilor elemente de interfaţare cu afişaj LCD şi butoane). Modul de funcţionare al acestui tip de aplicaţii poate fi influenţat atât prin intermediul elementelor proprii de interfaţare ale dispozitivului cât şi prin intermediul unui alt sistem de calcul precum calculatorul personal sau telefonul mobil (Fig. 4.4).

Câteva exemple din această categorie pot fi dispozitivele complexe care pot fi configurate şi asistate de la distanţă pentru a deservi un proces fizic: termostate digitale programabile atât de la distanţă cât şi prin intermediul panoului frontal propriu, ventilatoarele industriale comandate de la distanţă dar care au şi posibilitatea ajustării în funcţie de datele preluate de la senzori, imprimante multifuncţionale etc.

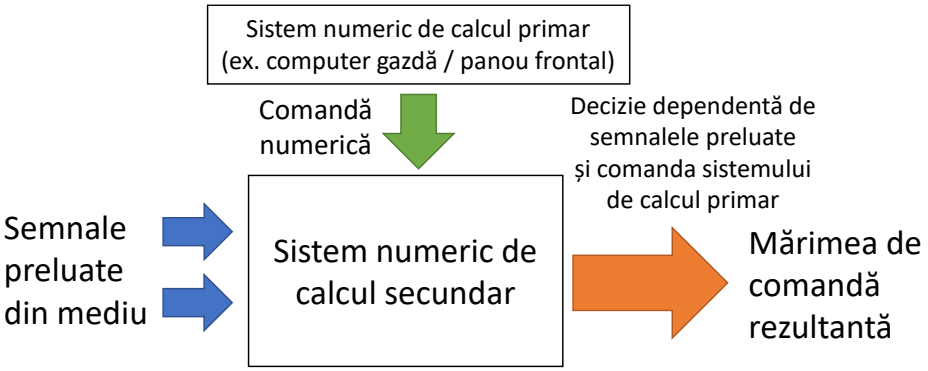


Fig. 4.4 - Aplicaţii hibride [1] [2]

III. ASPECTE TEORETICE

Există diverse metode şi procedee utilizate în vederea îmbunătăţirii modului de funcţionare al aplicaţiei concepute, precum [1] [2]:
- introducerea unui timp de întârziere (prin intermediul funcţiei „delay()”);
- introducerea constrângerilor simple (pentru limitarea unui interval de variaţie);
- introducerea unui algoritm de mediere în timp a semnalului;

A. Introducerea unui timp de întârziere

Prin intermediul funcţiei „delay()”, pot fi impuse **intervale de timp constante** la care să aibă loc o anumită operaţie impusă prin codul – program. Spre exemplu, în situaţia preluării temperaturii de la un senzor în cele mai multe cazuri **nu este necesară achiziţia de date la frecvenţă mare**, deoarece variaţia temperaturii în timp reprezintă un proces **relativ lent**. Având în vedere modul de funcţionare al convertorului analog – digital se constată că procesul de achiziţie al semnalului furnizat de traductor este oarecum susceptibil la **erorile cauzate de frecvenţa la care se realizează achiziţia de date**.
Acest impediment poate fi corectat introducând un **timp de întârziere,** (Fig. 4.5) astfel va avea loc şi stabilizarea termică a traductorului!

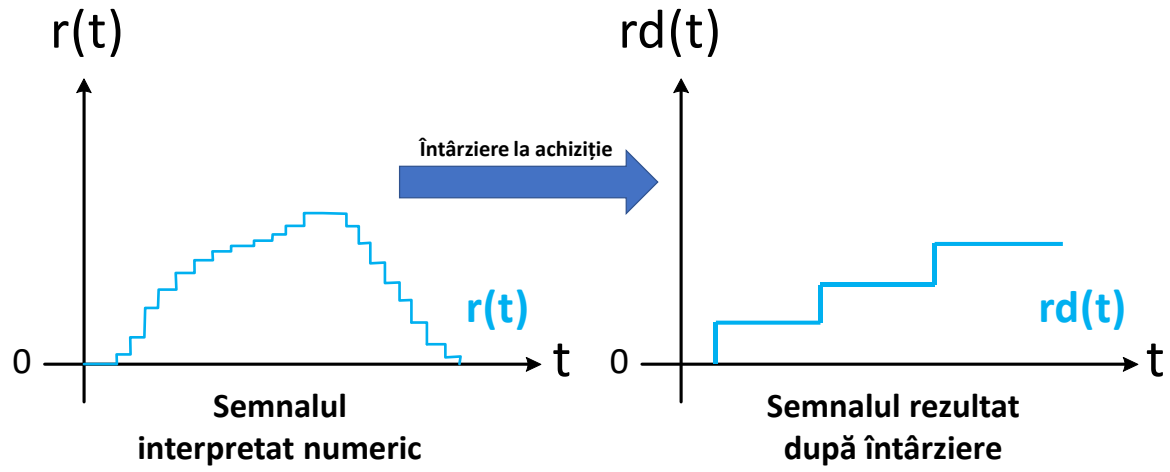


Fig. 4.5 - Aplicații hibride [1] [2]

B. Introducerea unor constrângeri

În unele cazuri este necesară **constrângerea** sau **restrângerea** intervalului de variație al unei mărimi. În vederea implementării acestui lucru, există două procedee:
- constrângerea unui semnal cu variație infinită la un interval de variație finit (mărginire);
- restrângerea sau lărgirea unui interval de variație finit (scalare);

Spre exemplu:
- constrângerea sau mărginirea se poate aplica în cazul în care se dorește limitarea variabilei care se incrementează sau decrementează în vederea reglării factorului de umplere pentru PWM cu două butoane (Fig. 4.6);
- restrângerea sau scalarea se poate aplica în cazul în care se dorește redefinirea intervalului de variație în vederea reglării factorului de umplere pentru PWM folosind rezultatul conversiei analog – digitale (Fig. 4.7);

Pentru o variabilă oarecare în intervalul [**constantă_1 constantă_2**] se va impune o constrângere simplă prin următorul procedeu:

```
if (variabilă <= constantă_1) {
      variabilă = constantă_1;
      }
if (variabilă >= constantă_2) {
      variabilă = constantă_2;
      }
```

Există și metoda alternativă în cadrul Arduino IDE:

```
constrain (variabilă, constantă_1, constantă_2);
```

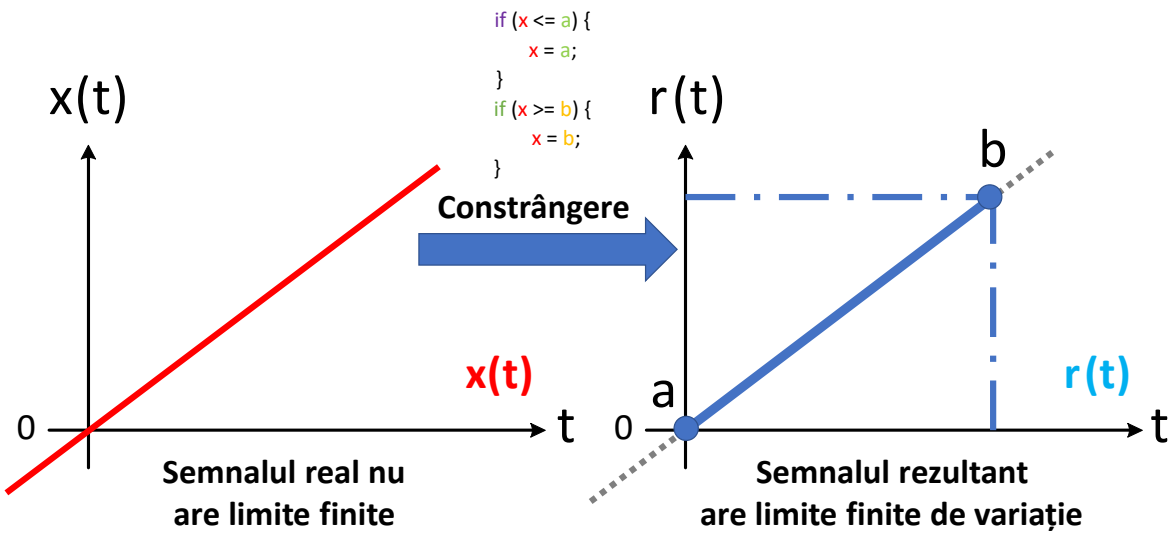


Fig. 4.6 – Constrângere sau mărginire simplă a unui interval infinit [1] [2]

Restrângerea intervalului de variație finit se va realiza prin intermediul funcției „map()" sub forma următoarei sintaxe:

```
var_2 = map(var_1, val_inf_1, val_sup_1, val_inf_2, val_sup_2);
```

Spre exemplu: fie variabila „p" cu valori în intervalul [0 1023]. Să se scrie funcția necesară pentru care, variabila „q" variază direct - proporțional cu variabila „p" în intervalul [0 255]:

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

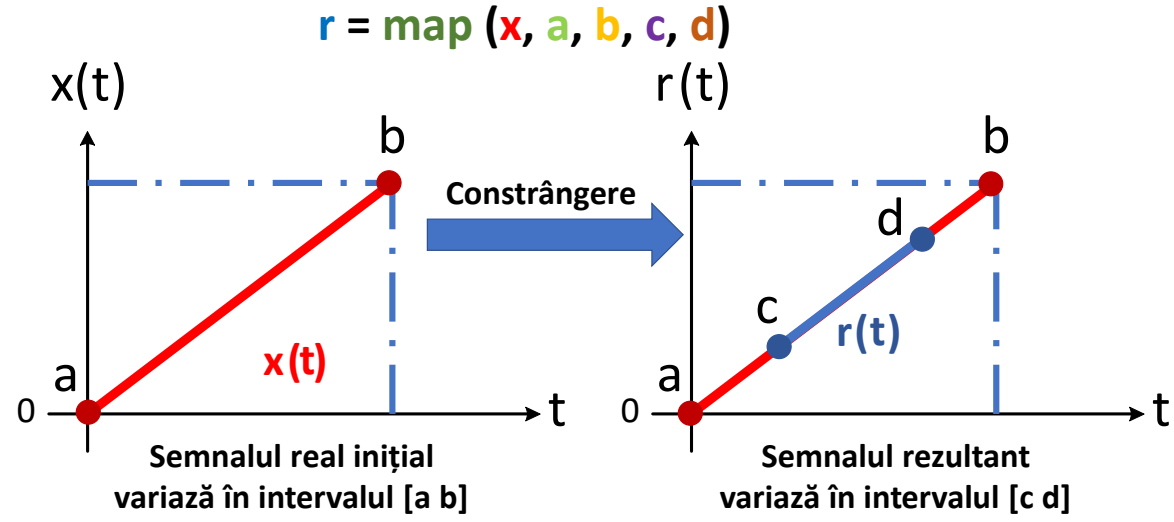


Fig. 4.7 - Restrângerea sau scalarea unui interval finit [1] [2]

C. Introducerea unui algoritm de mediere în timp a semnalului

Semnalul analogic preluat de la convertorul analog – digital este, în realitate, susceptibil zgomotelor şi erorilor de achiziţie. Astfel, unele valori achiziţionate nu reprezintă tocmai mărimea reală; se procedează deci, la excluderea datelor nesemnificative din şirul de măsurători (Fig. 4.8). Media aritmetică reprezintă una din metodele numerice de filtrare a semnalului şi poate fi exprimată sau implementată prin intermediul următoarei relaţii de calcul:

$$X(t) = \frac{1}{N} \cdot \sum_{i=1}^{N} x(t)$$

Unde „X(t)" reprezintă media aritmetică a valorii instantanee „x(t)", „N" este numărul de eşantioane sau măsurători achiziţionate, iar „i" reprezintă o variabilă – contor.

Algoritmul de mediere se implementează cu ajutorul unei structuri iterative „for":

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

OBSERVAŢII:

- algoritmul se implementează în bucla infinită „void loop()";
- variabila „suma" trebuie iniţializată cu valoarea zero la începutul structurii „void loop()", pentru a realiza reiniţializarea algoritmului pentru fiecare ciclu.

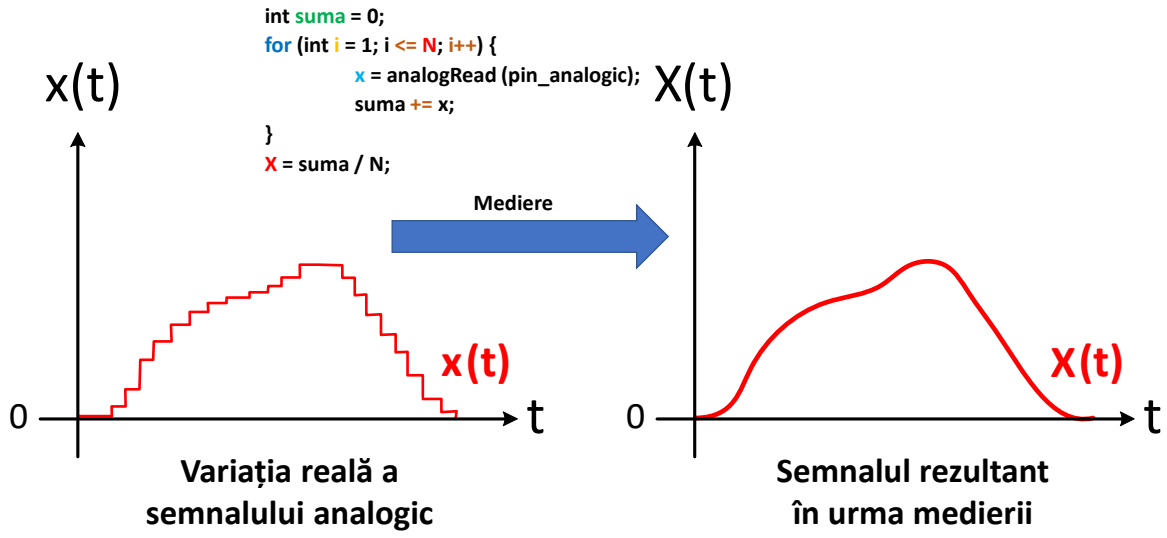


Fig. 4.8 – Media aritmetică a semnalului de măsură [1] [2]

IV. IMPLEMENTAREA APLICAȚIILOR

În vederea implementării aplicațiilor din cadrul acestei lucrări, vor fi utilizate două module specializate precum:
- traductorul de distanță HC – SR04 [3];
- modulul LCD QAPASS cu adaptor I²C M.H. [4] [5];

A. Principiul de funcționare al traductorului HC – SR04

Traductorul HC – SR04 funcționează pe baza efectului de reflexie și propagare a undei sonore. Modulul are în componență un dispozitiv transmițător („T") de tip difuzor de înaltă frecvență pe baza efectului piezoelectric și un receptor („R") de tip microfon de înaltă frecvență, funcționând pe baza efectului invers piezoelectric (Fig. 4.9) [3].

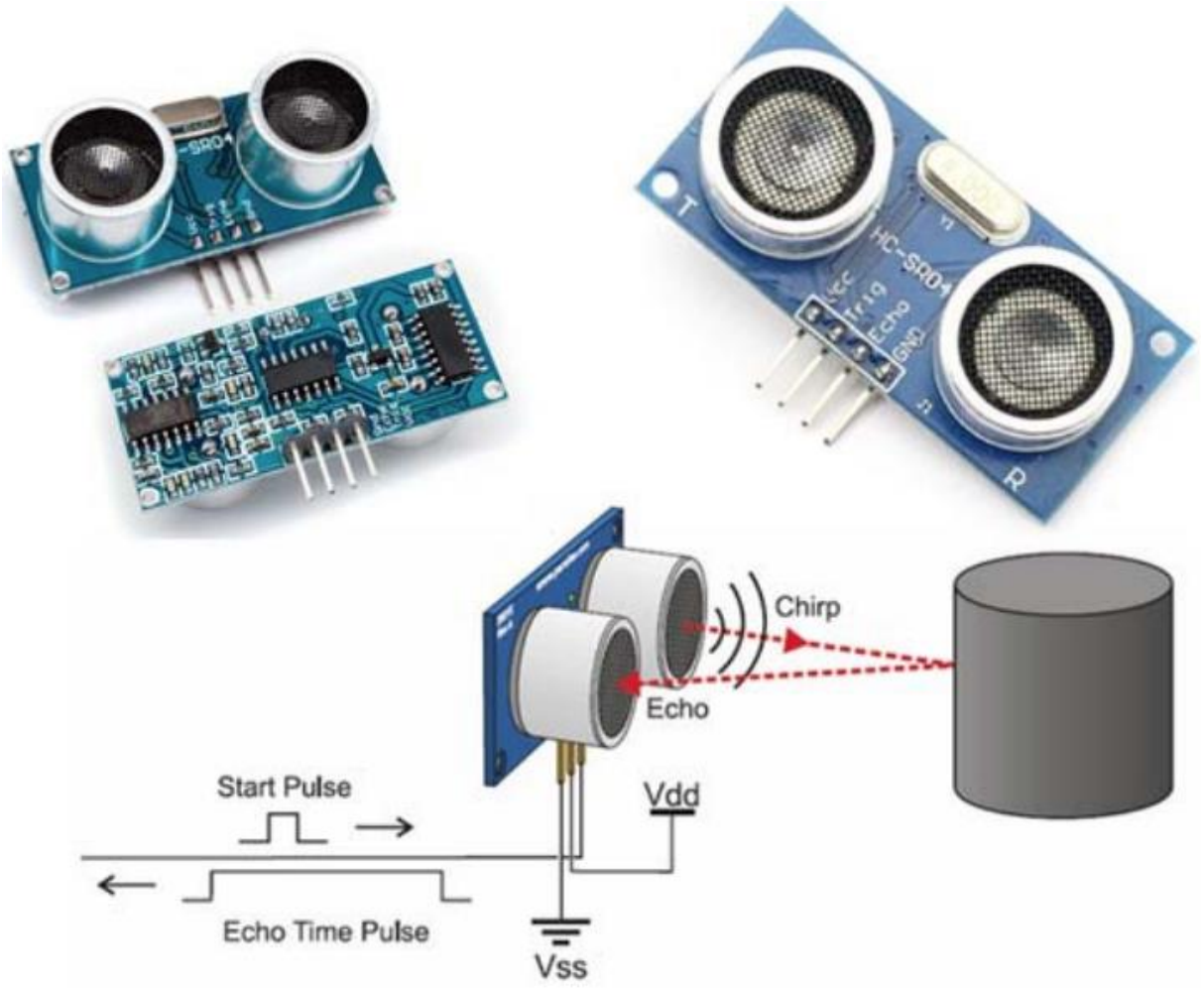


Fig. 4.9 – Traductorul HC-SR04 [3]

Frecvența undei sonore pe care o vehiculează modulul traductor este 40 [kHz]. Raza de acțiune este între 3 [cm] și 4 [m] la un unghi de incidență de 15°. Rezoluția aproximativă de măsură este de 1 [cm] / pas. Tensiunea de alimentare este 5 [V]. Semnalele furnizate la ieșirea modulului sunt de natură digitală (Fig. 4.10) [3].

Pe baza frecvenţei undei sonore şi a vitezei sale de deplasare se poate estima distanţa de la modul până la obstacolul întâmpinat în calea de propagare conform relaţiilor de calcul, cunoscând constanta v_{sunet} = 340 [m/s]:

$$v = \frac{d}{t} \rightarrow t = \frac{d}{v} \rightarrow d = \frac{0.034 \cdot t}{2} \ [cm]$$

Microcontrolerul măsoară distanţa pe baza timpul de răspuns echivalent timpului de propagare al undei sonore prin mediul de transmitere (Fig. 4.10).

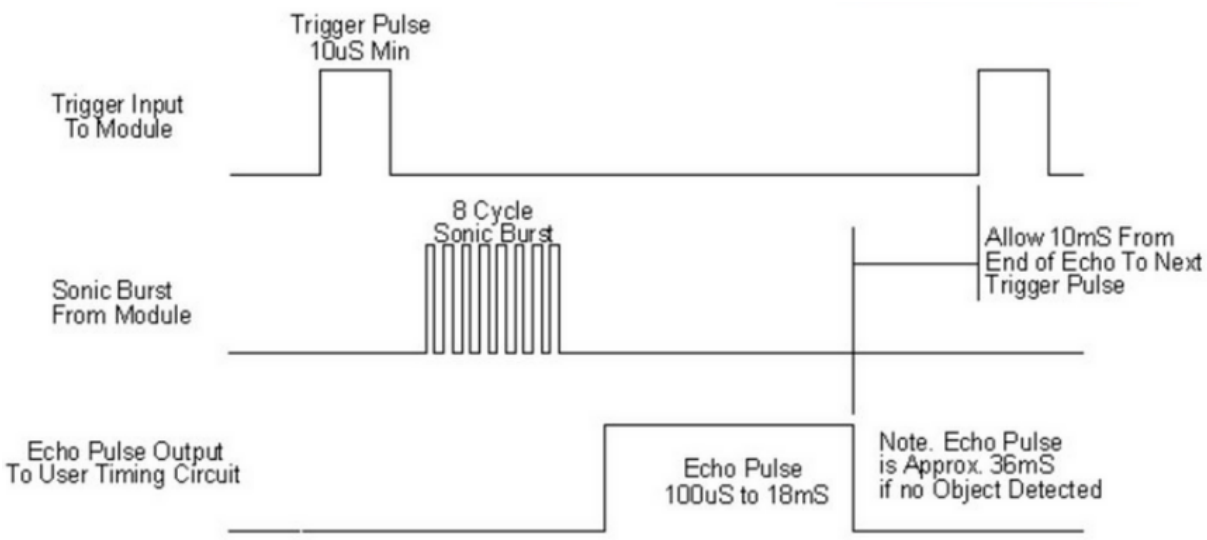


Fig. 4.10 – Semnalele vehiculate de modulul HC-SR04 [3]

B. Principiul de funcţionare a modulului cu afişaj LCD

Principiul de funcţionare al afişajului LCD (eng. Liquid Crystal Display) se bazează pe efectul de magnetizare al cernelii ferofluidice (cristale lichide). În construcţia afişajului se regăseşte o matrice de bobine având rolul de electromagneţi. Prin intermediul micro-electro-magneţilor cerneala ferofluidică este dirijată printr-un sistem de vase comunicante transparente care au forma caracterelor ce pot fi afişate (Fig. 4.11) [4].

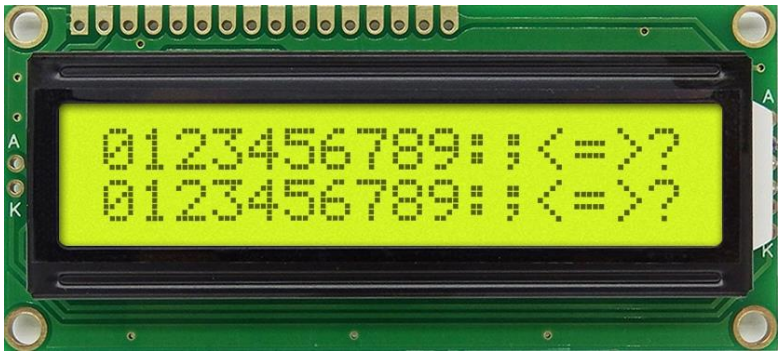


Fig. 4.11 – Afişaj LDC caracter [4]

Modulul LCD caracter utilizat în această lucrare dispune de 32 celule caracter dispuse pe două rânduri, rezultând deci o rezoluţie 16 x 2. Caracterul se construieşte pe baza unor matrice de puncte (celulele). Pachetul modular dispune de 16 terminale de racord (cu posibilitatea reglării luminii de fundal). Substratul constructiv de siliciu înglobează un microcontroler pentru gestionarea modulului. Acceptă comenzi pe 4 sau 8 biţi vehiculaţi în mod paralel la terminalele de date (DB0 – DB7). Pot fi transferate seturi suplimentare de caractere în memoria modulului LCD [4].

Pentru a reduce numărul de terminale conectate la platforma de dezvoltare Arduino Nano, împreună cu modulul LCD se va utiliza adaptorul pentru interfaţă de comunicare de tip Paralel la interfaţă de tip I²C (eng. LCD I²C back Pack) (Fig.4.12). În acest sens, pentru conectarea modulului LCD la platforma de dezvoltare Arduino Nano, se vor utiliza patru terminale, anume [5]:

- Vcc – pentru alimentare la 5 [V];
- GND – pentru conectare la potenţial zero;
- SDA (eng. Serial Data) – pentru conectare la terminalul A4;
- SCL (eng. Serial Clock) – pentru conectare la terminalul A5;

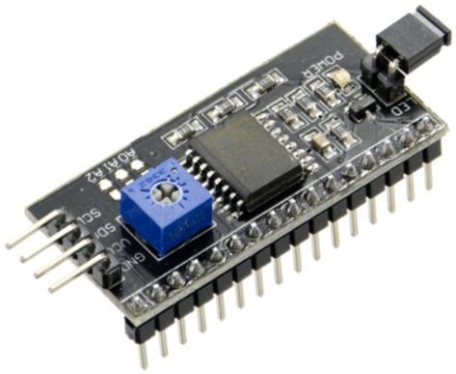


Fig. 4.12 – Adaptor I²C la paralel pentru afişaj LCD [5]

Pentru a utiliza afişajul LCD împreună cu adaptorul I²C la interfaţă Paralel, este necesară instalarea şi includerea bibliotecii de funcţii „Wire.h" pentru iniţializarea comunicaţiei pe interfaţa I²C şi a bibliotecii de funcţii „LiquidCrystal_I2C.h" [5].

Pentru a preveni efectul remanent al caracterelor de pe ecran este necesară implementarea unui algoritm pentru ştergere şi actualizare a zonei de afişare. În principal, acest fenomen poate fi observat la afişarea numerelor fracţionare. Odată cu afişarea unui sub-multiplu, celula din faţă se va popula cu un caracter „0". Procesul de eliminare a caracterelor remanente sau a cifrelor „0" din faţa fracţiei (ex. 005,0) poartă denumirea (în literatura de specialitate) de (eng.) Leading Zeros Clearing, spre exemplu:

```
if(media < 1000) {          // verificarea numărului de caractere
lcd.setCursor(3,1);         // plasarea cursorului de scriere la coordonate
lcd.print(" ");       // afişarea unui spaţiu gol la coordonate
lcd.setCursor(0,1);         // plasarea cursorului de scriere la coordonate
lcd.print(media);           // afişarea variabilei la coordonate
}
```

Pentru a evidenția conceptele prezentate anterior, se propune deci implementarea următoarelor aplicații cu ajutorul platformei Arduino Nano:
- Controlul digital al factorului de umplere pentru un semnal modulat în lățime;
- Preluarea temperaturii de la un traductor și filtrarea zgomotelor;
- Măsurarea distanței cu ajutorul traductorului ultrasonic HC – SR04;
- Utilizarea unui afișaj de tip LCD;
- Implementarea unui termometru digital;
- Implementarea unei rigle digitale;

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;
- butoane cu apăsare și revenire;
- rezistențe cu valoarea de 100 [Ω];
- rezistențe cu valoarea de 10 [kΩ];
- senzor de temperatura LM-35;
- traductor de distanță pe bază de efect ultrasonic HC-SR04;
- modulul pentru afișare LCD QAPASS cu adaptor I²C la Paralel M.H.;
- fire pentru conexiune rapidă compatibile cu placa de testare;
- calculator gazdă având mediul Arduino IDE instalat;
- cablu adaptor USB A la mini USB;

APLICAȚIA 1

Se va implementa circuitul conform următoarei scheme (Fig. 4.13):

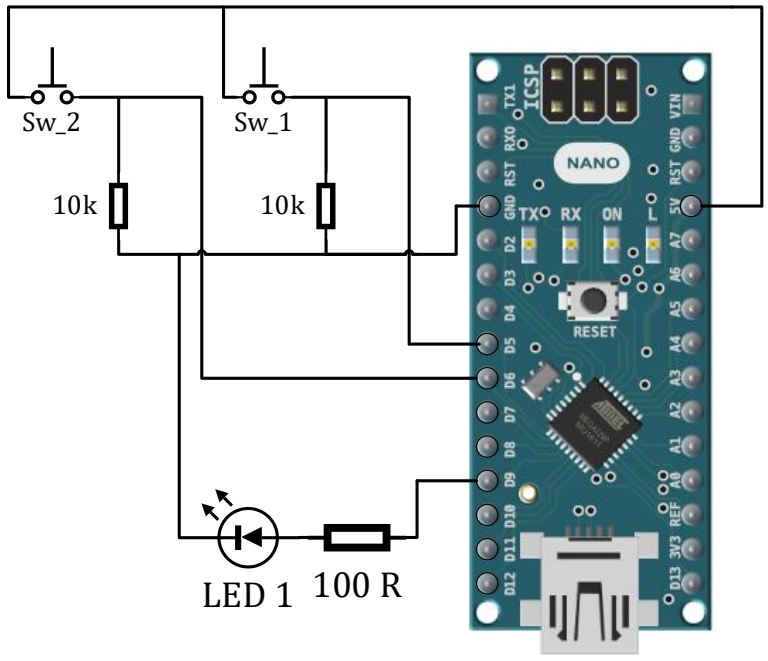


Fig. 4.13 – Schema electronică pentru implementarea circuitului specific aplicației 1

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;
int dc = 0;

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

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

  if(sw_1_state == 1) {
    dc++;
    if (dc >= 255) {
      dc = 255;
    }
  }

  if(sw_2_state == 1) {
    dc--;
    if (dc <= 0) {
      dc = 0;
    }
  }
  Serial.print("Factor de umplere: ");
  Serial.print(dc);
  Serial.println("");
  analogWrite(pin_led, dc);
  delay(10);
}
```

Montajul experimental se va realiza conform (Fig. 4.14):

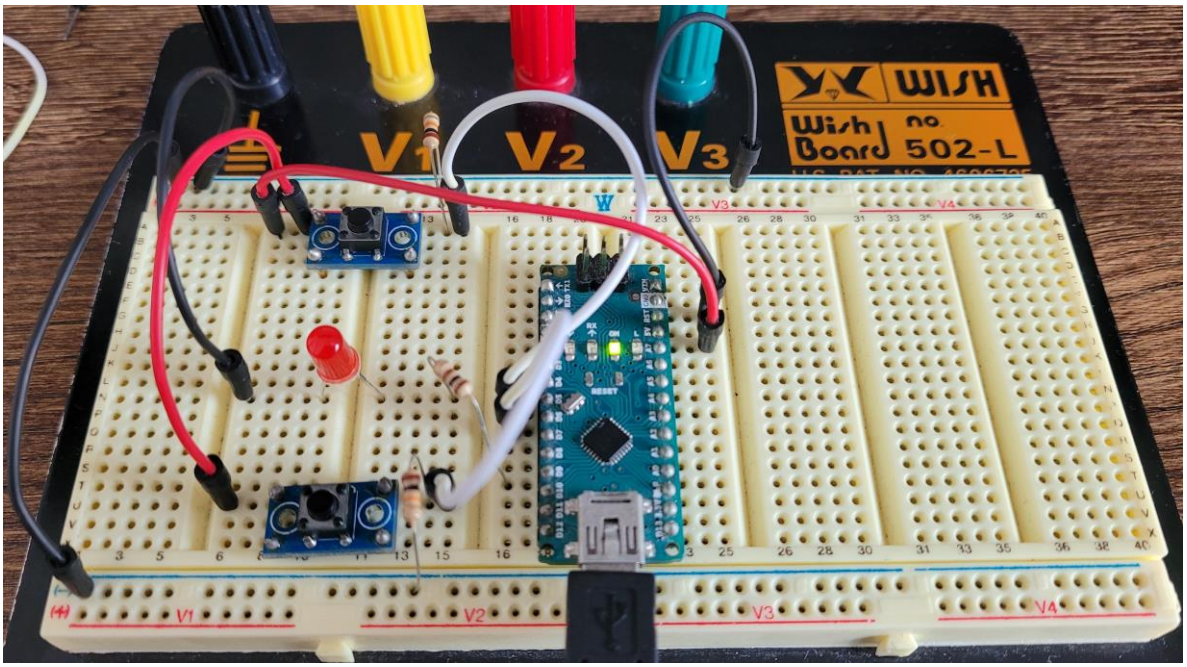


Fig. 4.14 – Montaj experimental pentru aplicaţia 1

Implementarea aplicaţiei nr. 1 presupune:

- declararea a trei constante globale, în vederea stabilirii numerelor de ordine ale terminalelor corespondente atât butoanelor cu apăsare şi revenire cât şi diodei (LED);
- stabilirea modului de lucru „intrare digitală" pentru terminalele butoanelor;
- stabilirea modului de lucru „ieşire digitală" pentru terminalul diodei (LED);
- preluarea stărilor logice ale terminalelor alese pentru ataşarea butoanelor cu apăsare şi revenire prin intermediul instrucţiunii „digitalRead ()";
- incrementarea sau decrementarea factorului de umplere „dc" la menţinerea apăsată sau apăsarea repetată a butoanelor;
- impunerea unor constrângeri simple;
- afişarea în consola Serial a valorii factorului de umplere (Fig. 4.15 / 4.16);
- generarea unui semnal modulat în lăţime prin intermediul funcţiei „analogWrite()";

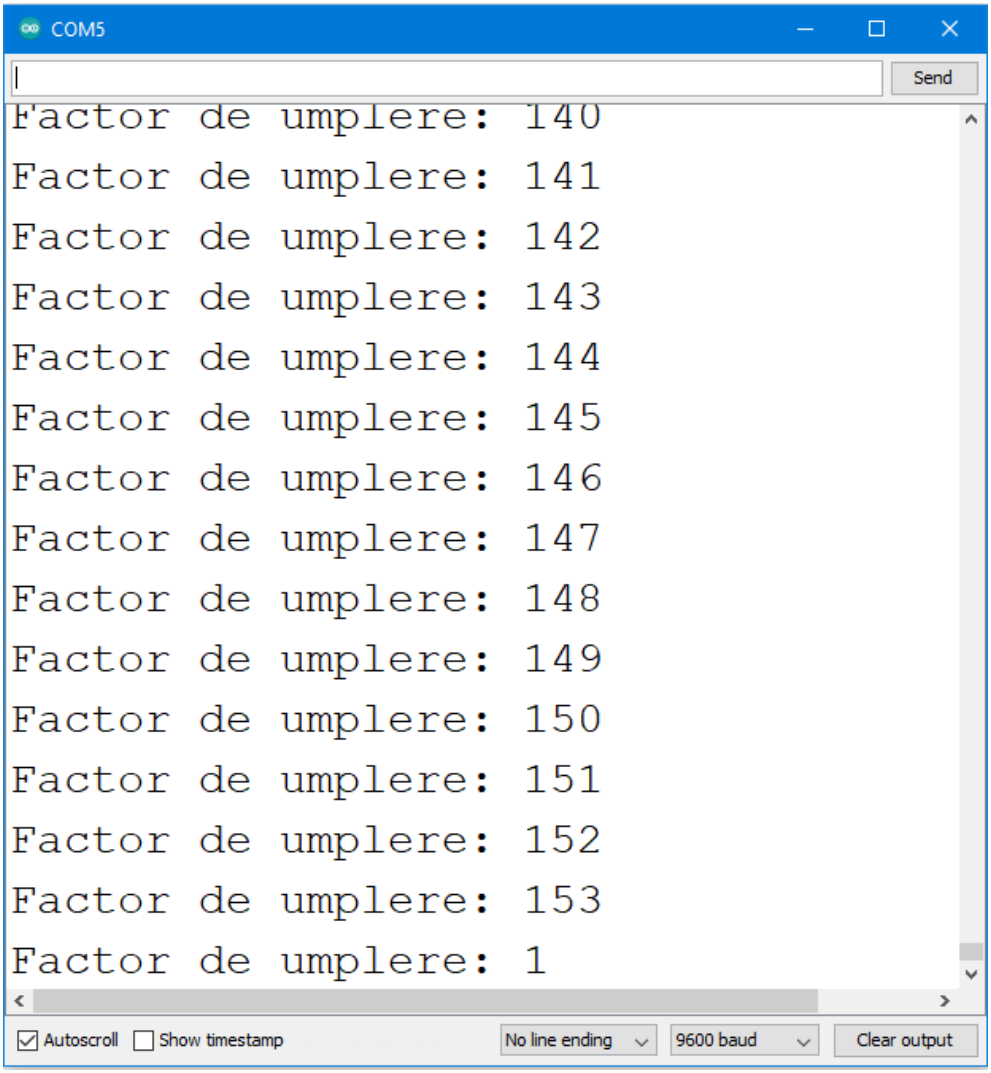


Fig. 4.15 – Afișarea valorii zecimale a factorului de umplere în consola Serial

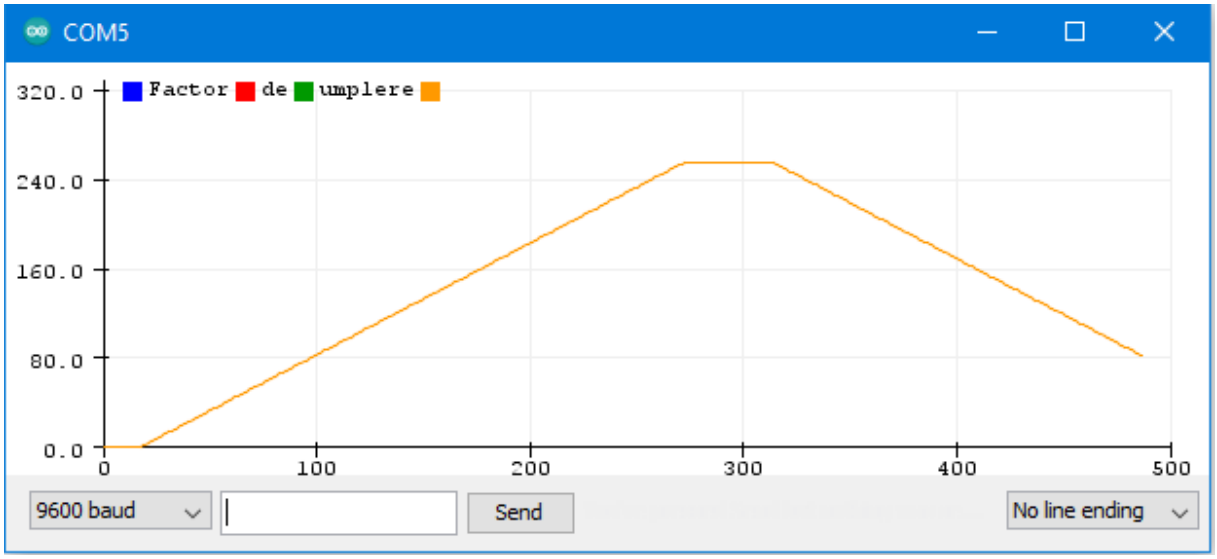


Fig. 4.16 – Afișarea valorii zecimale a factorului de umplere în consola grafică

APLICAȚIA 2

Se va implementa circuitul conform următoarei scheme (Fig. 4.17):

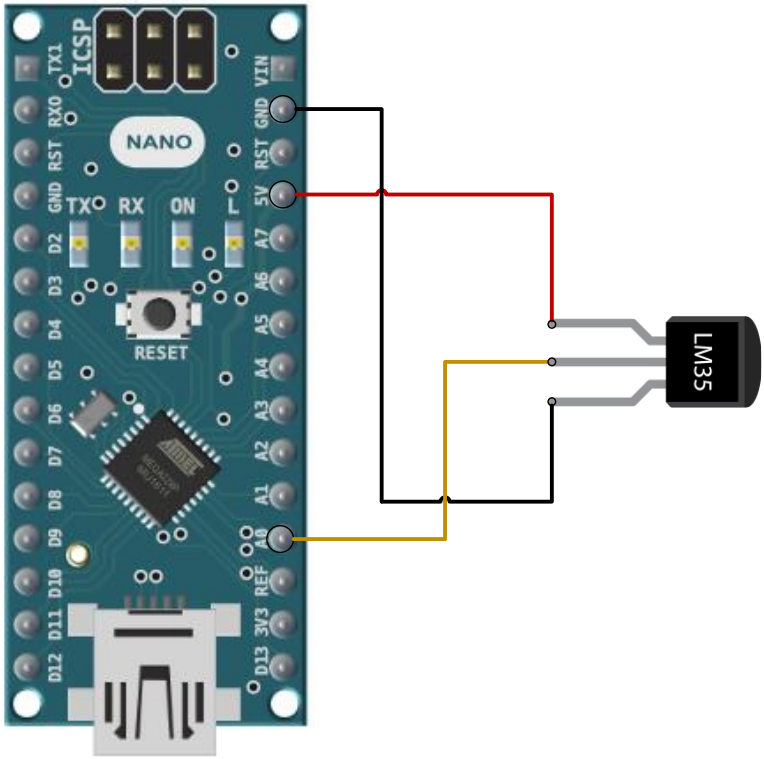


Fig. 4.17 – Schema electronică pentru implementarea circuitului specific aplicației 2

Se va realiza următorul montaj experimental (Fig. 4.18):

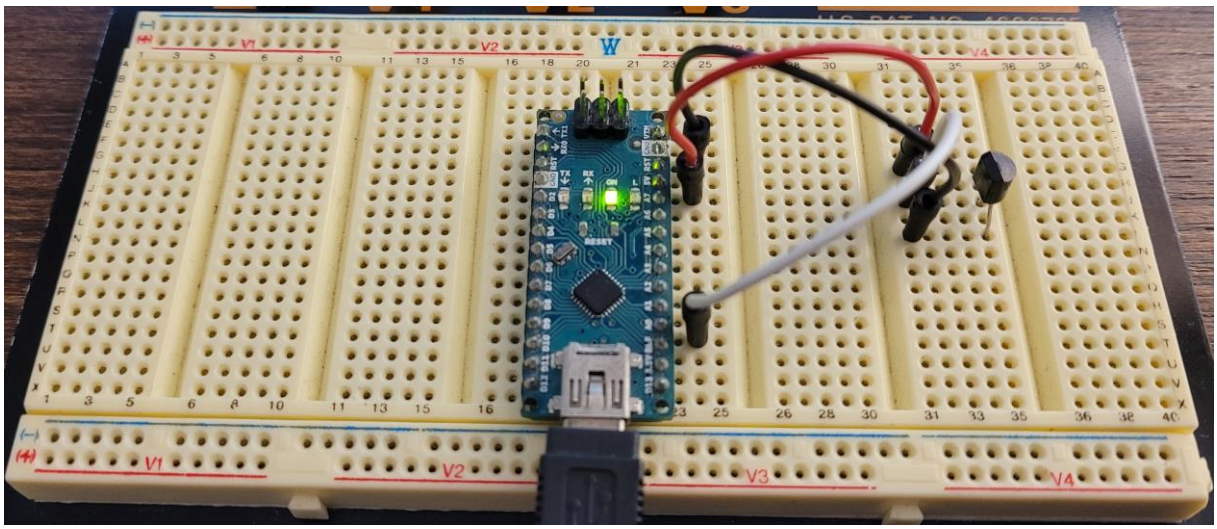


Fig. 4.18 – Montajul experimental specific aplicației 2

Se va implementa următorul cod program:

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

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

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

Implementarea aplicației nr. 2 presupune:

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

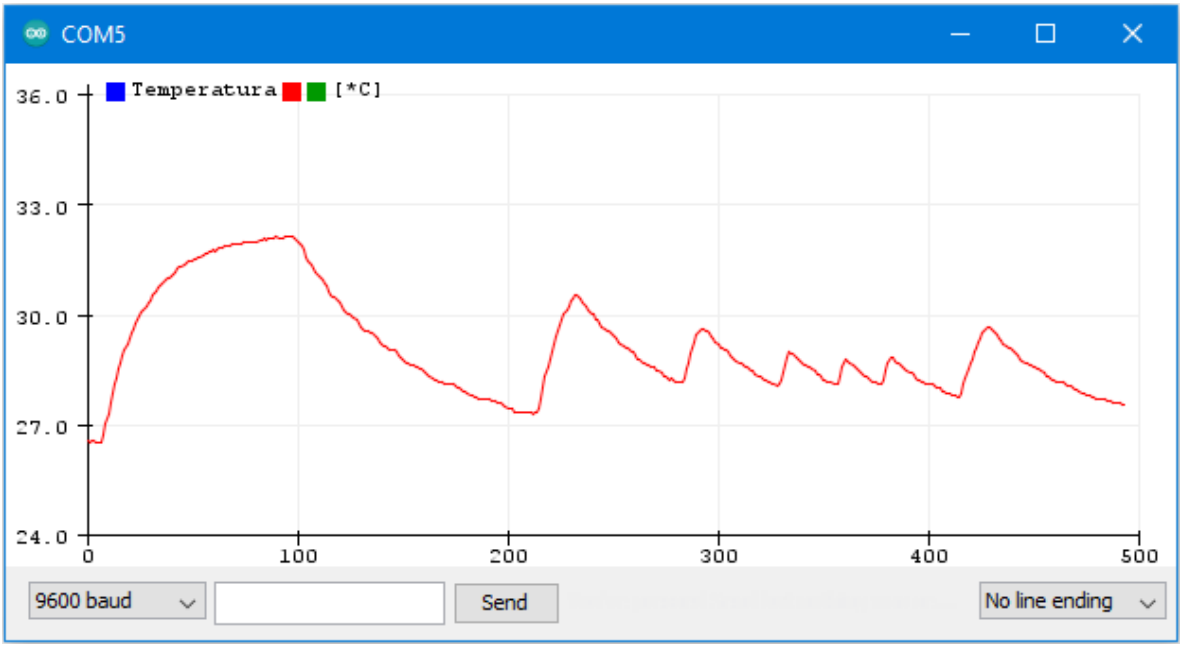


Fig. 4.19 – Afișarea valorilor medii ale temperaturii în consola grafică

APLICAȚIA 3

Se va implementa circuitul conform următoarei scheme (Fig. 4.20):

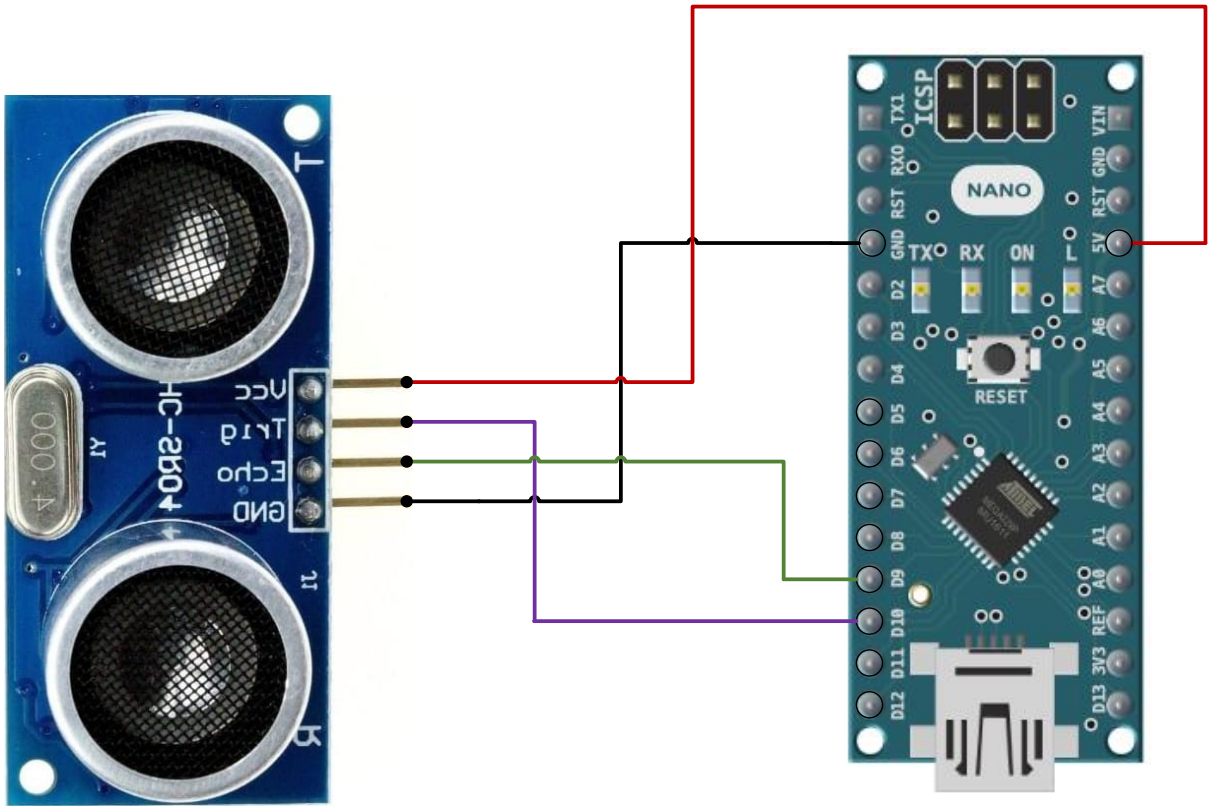


Fig. 4.20 – Schema electronică pentru implementarea circuitului specific aplicației 3

Se va realiza următorul montaj experimental (Fig. 4.21):

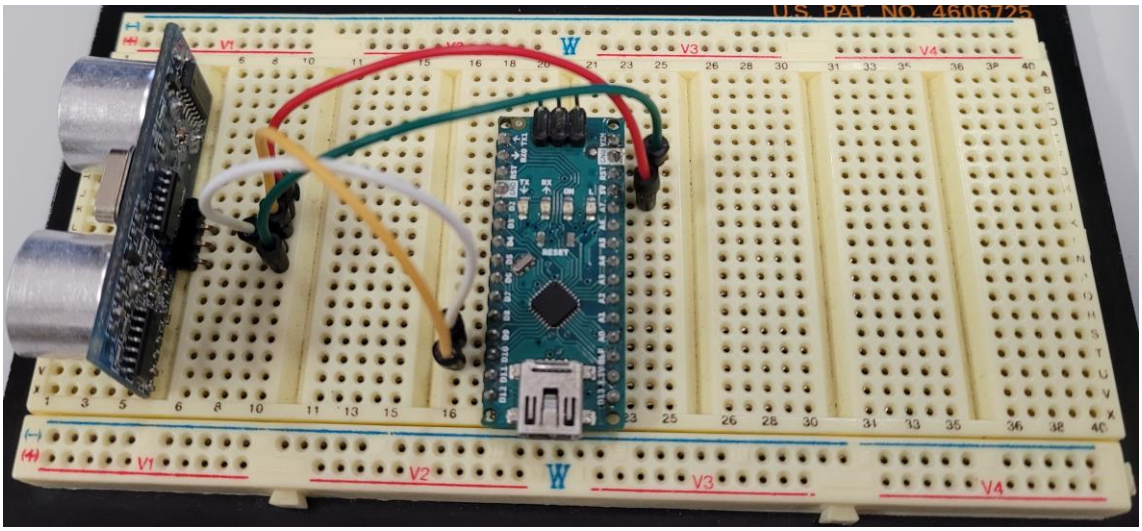


Fig. 4.21 - Montajul experimental specific aplicației 3

Se va implementa următorul cod program:

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

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

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

Implementarea aplicației nr. 3 presupune:

- declararea unei constante de tip număr întreg „echo” având ca și valoare „9”;
- declararea unei constante de tip număr întreg „trigger” având ca și valoare „10”;
- inițializarea unei variabile de tip număr întreg „t” cu valoarea „0”;
- inițializarea unei variabile de tip număr întreg „d” cu valoarea „0”;
- stabilirea modului de lucru „ieșire digitală” pentru terminalul „9”;
- stabilirea modului de lucru „intrare digitală” pentru terminalul „10”;
- inițializarea comunicației Serial;
- generarea unei secvențe de semnalizare acustică la o frecvență de 40 [kHz];
- determinarea timpului de propagare a undei sonore cu ajutorul funcției „pulseIn ()”;
- determinarea distanței pe baza formulei de calcul a vitezei de propagare a undei sonore;
- afișarea în consola Serial a rezultatului măsurării (Fig. 4.22);

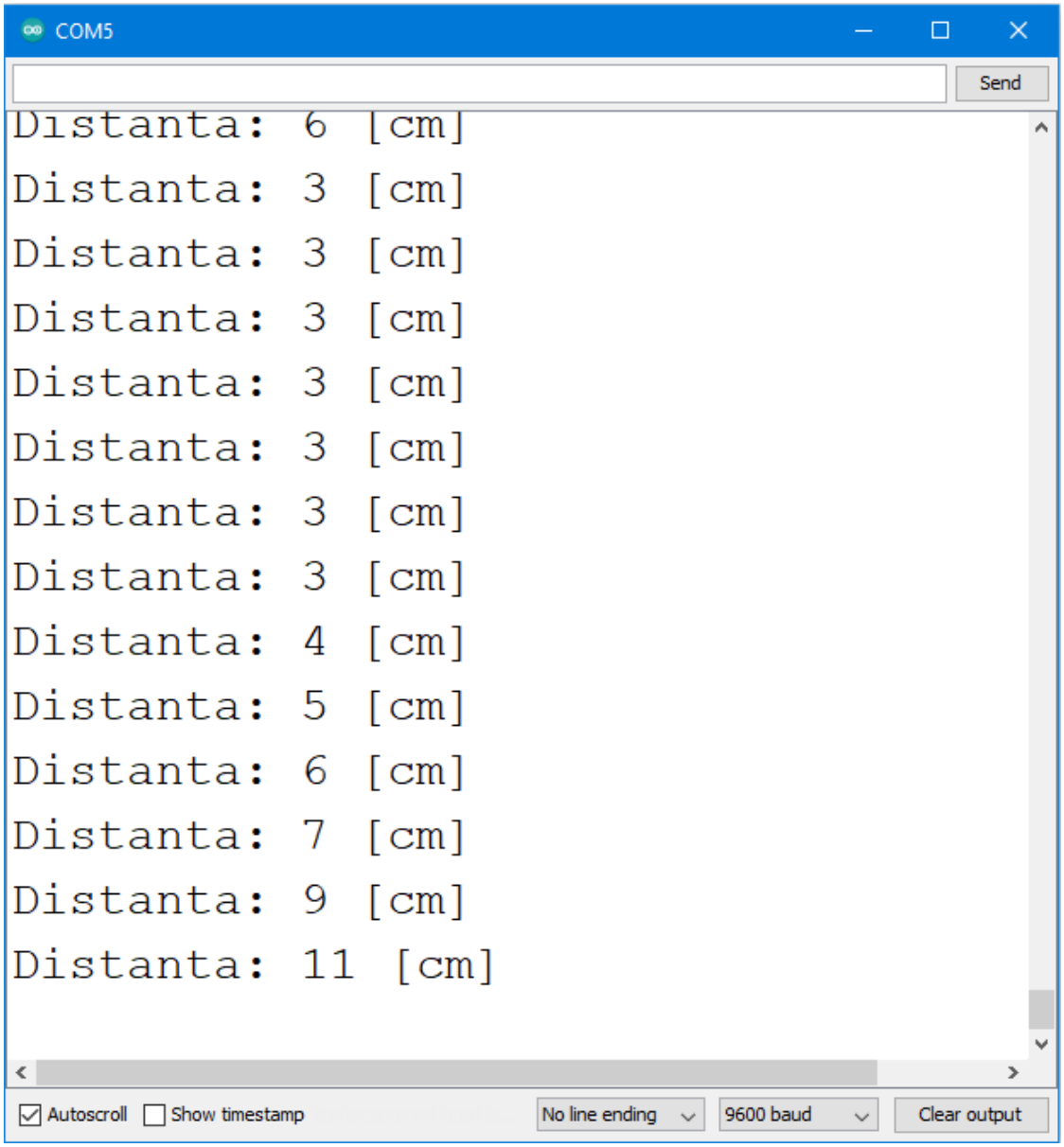


Fig. 4.22 – Afișarea în consola Serial a rezultatului de măsurare a distanței

APLICAȚIA 4

Se va implementa circuitul conform următoarei scheme (Fig. 4.23):

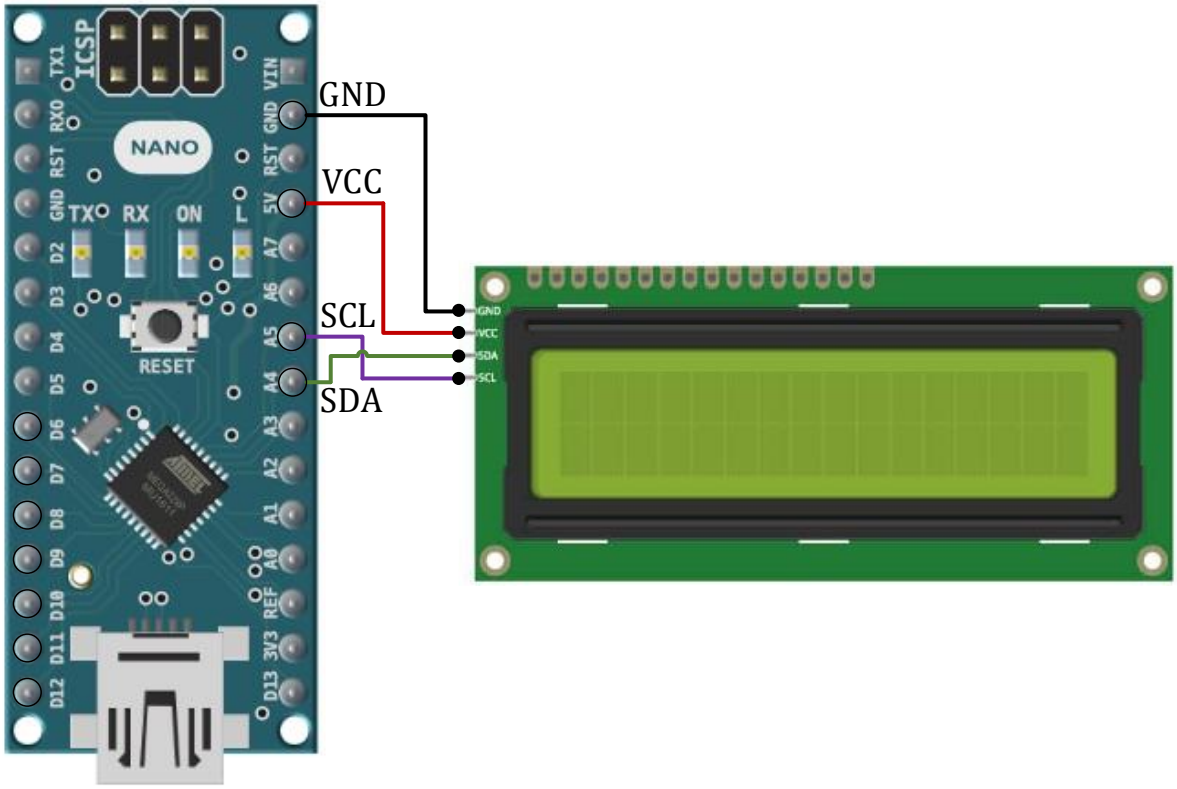


Fig. 4.23 - Schema electronică pentru implementarea circuitului specific aplicației 4

Se va realiza următorul montaj experimental (Fig. 4.24):

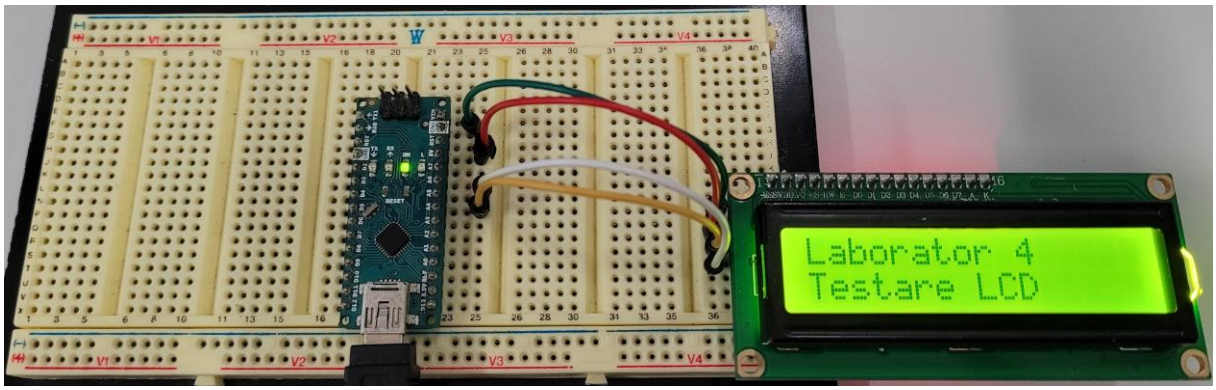


Fig. 4.24 - Montajul experimental specific aplicației 4

Se va implementa următorul cod program:

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

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

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

Implementarea aplicației nr. 4 presupune:

- inițializarea bibliotecii de funcții „Wire.h” pentru comunicația I^2C;
- inițializarea bibliotecii de funcții „LiquidCrystal_I2C.h”;
- inițializarea parametrilor aferenți obiectului „lcd” (ex. adresa I^2C și terminale);
- inițializarea rezoluției afișajului prin intermediul funcției „lcd.begin ()”;
- inițializarea luminiței de fundal a ecranului printr-o secvență de licărire;
- menținerea permanent aprinsă a luminiței prin intermediul funcției „lcd.backlight ()”;
- plasarea cursorului de scriere cu ajutorul funcție „lcd.setCursor ()”;
- afișarea unui mesaj pe ecran cu ajutorul funcției „lcd.print ()”;

OBSERVAȚII:

1. Adresa dispozitivului adaptor al ecranului atașat pe interfața I^2C (în cazul de față) este „0x27”. Aceasta se poate determina pe baza datelor de catalog sau prin intermediul codului program „i2c_scanner” regăsit în spațiul public al domeniului „arduino.cc”. Cele mai uzuale adrese ale dispozitivelor adaptoare sunt „0x27” și „0x3F”.

2. Funcția „lcd.SetCursor” are ca și argumente lcd.SetCursor(<celulă>, <rând>);

APLICAȚIA 5

Se va implementa circuitul conform următoarei scheme (Fig. 4.25):

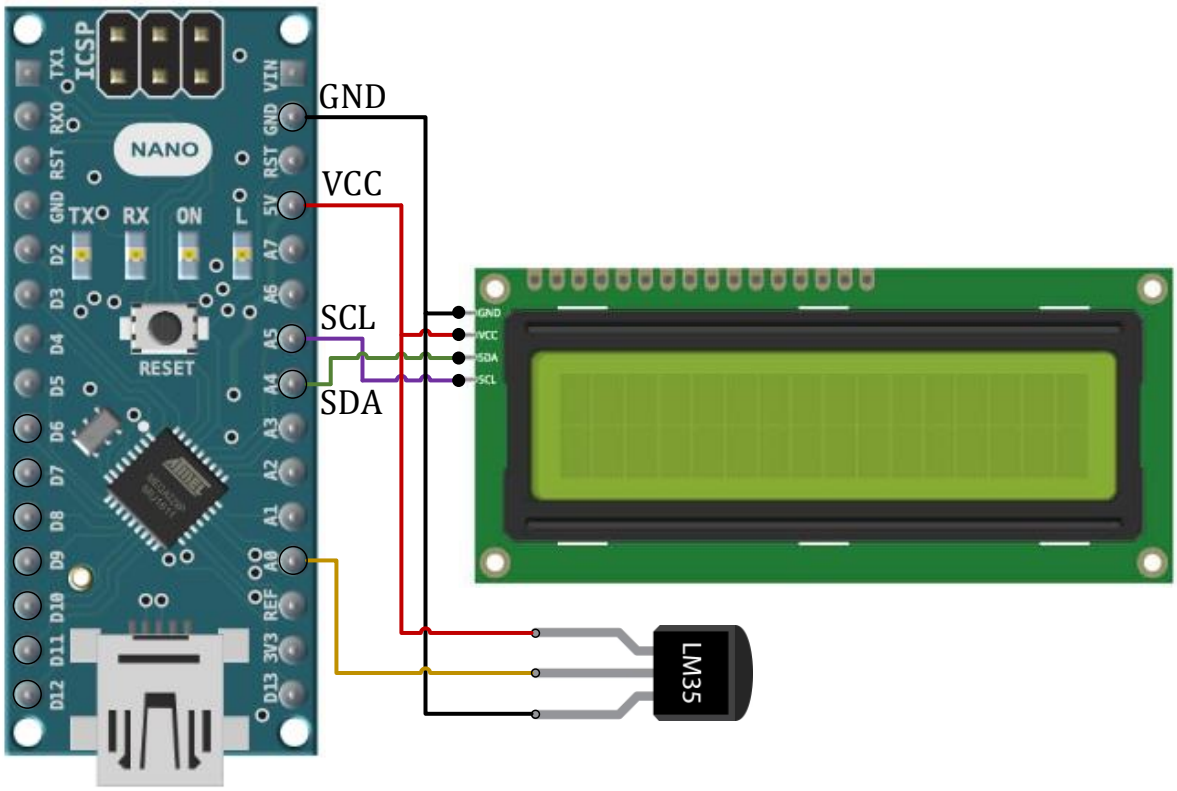


Fig. 4.25 - Schema electronică pentru implementarea circuitului specific aplicației 5

Se va realiza următorul montaj experimental (Fig. 4.26):

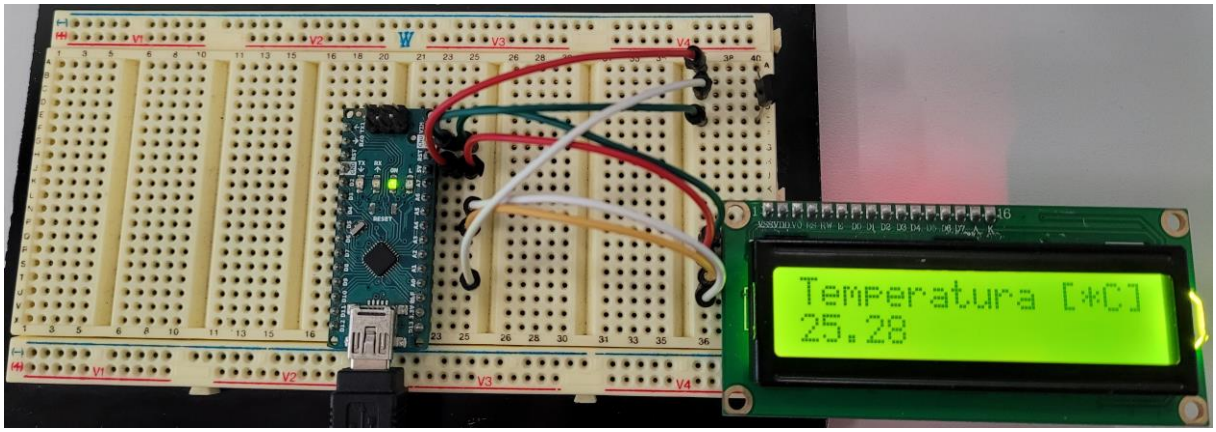


Fig. 4.26 - Montajul experimental specific aplicației 5

Se va implementa următorul cod program:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
const int analog_pin = 0;
int ADC_val = 0;
int N = 500;
float U = 0.00;
float temp = 0.00;
float media = 0.00;

void setup() {
  lcd.begin(16,2);
  for(int i = 0; i< 3; i++) {
    lcd.backlight();
    delay(250);
    lcd.noBacklight();
    delay(250);
  }
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Temperatura [*C]: ");
}

void loop() {
  float suma = 0.00;

  for (int i = 1; i <= N; i++) {
  ADC_val = analogRead(analog_pin);
  U = (4.78 / 1023.00) * ADC_val;
  temp = 100.00 * U;
  suma += temp;
  }

  media = suma / N;

  lcd.setCursor(0,1);

    if(media < 1000) {
    lcd.setCursor(3,1);
    lcd.print(" ");
    lcd.setCursor(0,1);
    lcd.print(media);
  }
```

```
if(media < 100) {
    lcd.setCursor(2,1);
    lcd.print(" ");
    lcd.setCursor(0,1);
    lcd.print(media);
  }
    if(media < 10) {
    lcd.setCursor(1,1);
    lcd.print(" ");
    lcd.setCursor(0,1);
    lcd.print(media);
  }
  delay(500);
}
```

Implementarea aplicaţiei nr. 5 presupune:

- iniţializarea bibliotecii de funcţii „Wire.h” pentru comunicaţia I^2C;
- iniţializarea bibliotecii de funcţii „LiquidCrystal_I2C.h”;
- iniţializarea parametrilor aferenţi obiectului „lcd” (ex. adresa I^2C şi terminale);
- iniţializarea unui terminal analogic „A0”;
- iniţializarea variabilelor pentru determinarea conţinutului registrului convertorului analog – digital, a tensiunii de măsură şi a temperaturii medii;
- iniţializarea rezoluţiei afişajului prin intermediul funcţiei „lcd.begin ()”;
- iniţializarea luminiţei de fundal a ecranului printr-o secvenţă de licărire;
- afişarea mesajului iniţial „Temperatura [*C]: ” pe primul rând la începutul ecranului;
- menţinerea permanent aprinsă a luminiţei prin intermediul funcţiei „lcd.backlight ()”;
- determinarea temperaturii medii pe baza datelor furnizate de către traductorul LM-35 conectat la o intrare analogică a convertorului analog – digital (similar aplicaţiei 2);
- plasarea cursorului de scriere cu ajutorul funcţie „lcd.setCursor ()”;
- afişarea unui mesaj şi a valorii măsurate pe ecran cu ajutorul funcţiei „lcd.print ()”;
- actualizarea valorilor numerice prin intermediul algoritmului „Leading Zeros Clearing”;

APLICAŢIA 6

Se va implementa circuitul conform următoarei scheme (Fig. 4.27):

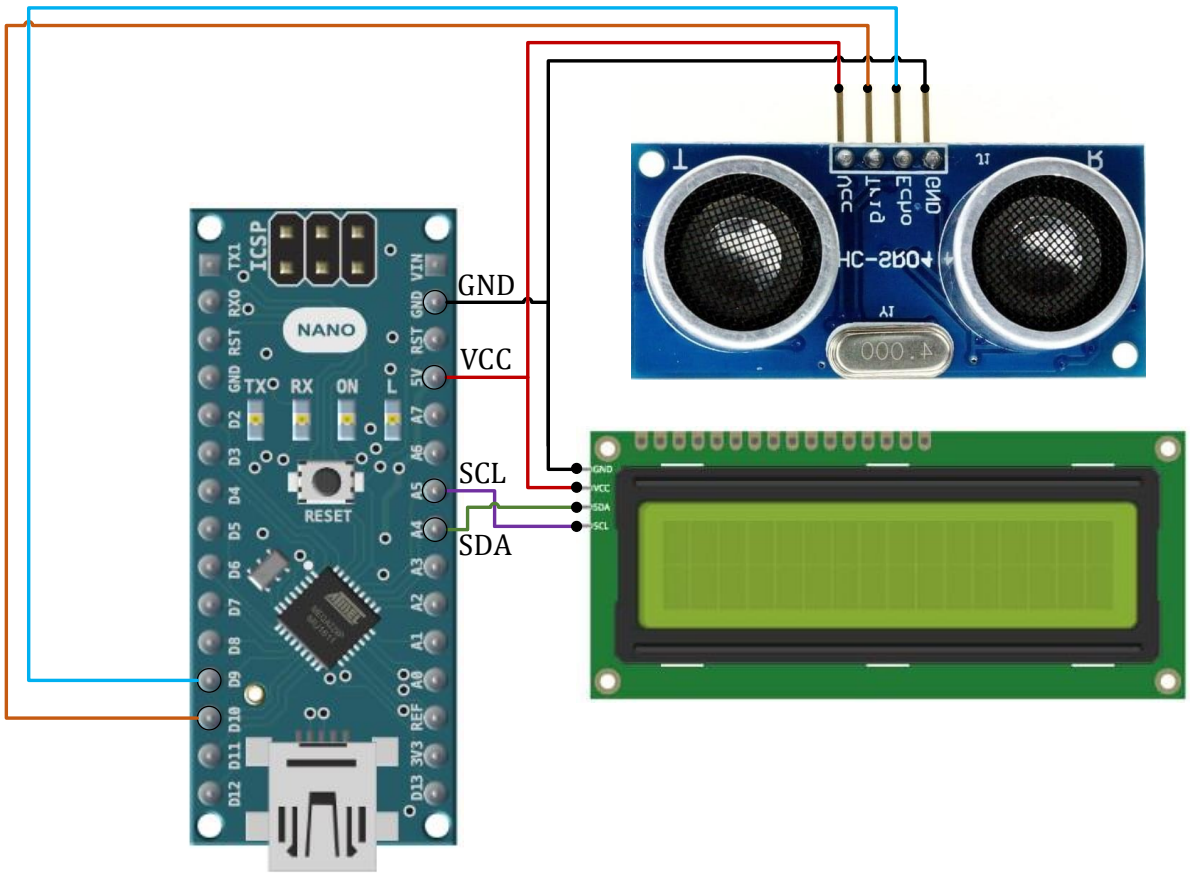


Fig. 4.27 – Schema electronică pentru implementarea circuitului specific aplicaţiei 6

Se va realiza următorul montaj experimental (Fig. 4.28):

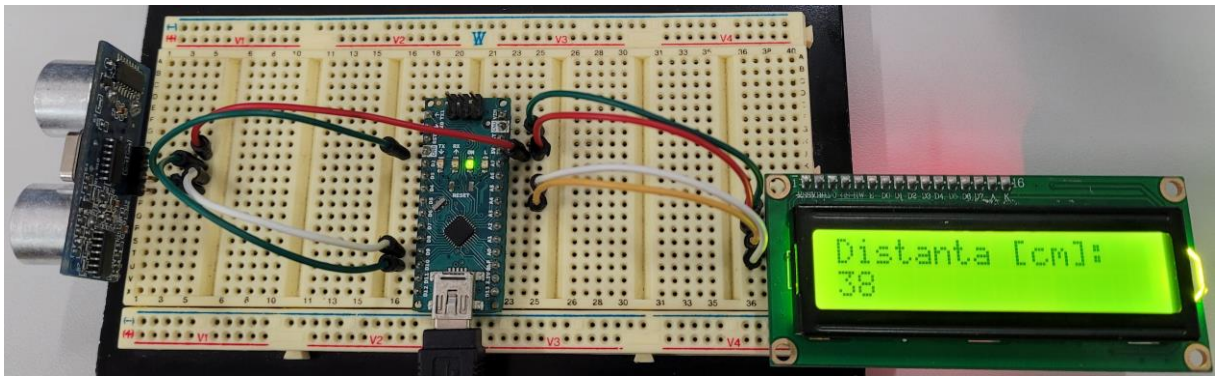


Fig. 4.28 – Montajul experimental specific aplicaţiei 6

Se va implementa următorul cod program:

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

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

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

void loop() {
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);

  t = pulseIn(echo, HIGH);
  d = t * 0.034 / 2.0;

  lcd.setCursor(0,1);

    if(d < 1000) {
    lcd.setCursor(3,1);
    lcd.print(" ");
    lcd.setCursor(0,1);
    lcd.print(d);
  }
```

```
if(d < 100) {
    lcd.setCursor(2,1);
    lcd.print(" ");
    lcd.setCursor(0,1);
    lcd.print(d);
  }
    if(d < 10) {
    lcd.setCursor(1,1);
    lcd.print(" ");
    lcd.setCursor(0,1);
    lcd.print(d);
  }
  delay(500);
}
```

Implementarea aplicaţiei nr. 6 presupune:

- iniţializarea bibliotecii de funcţii „Wire.h” pentru comunicaţia I^2C;
- iniţializarea bibliotecii de funcţii „LiquidCrystal_I2C.h”;
- iniţializarea parametrilor aferenţi obiectului „lcd” (ex. adresa I^2C şi terminale);
- iniţializarea unui terminalului „D10” ca şi ieşire digitală;
- iniţializarea unui terminalului „D9” ca şi intrare digitală;
- iniţializarea variabilelor pentru determinarea duratei de propagare şi a distanţei;
- iniţializarea rezoluţiei afişajului prin intermediul funcţiei „lcd.begin ()”;
- iniţializarea luminiţei de fundal a ecranului printr-o secvenţă de licărire;
- afişarea mesajului iniţial „Distanta [cm]: ” pe primul rând la începutul ecranului;
- menţinerea permanent aprinsă a luminiţei prin intermediul funcţiei „lcd.backlight ()”;
- determinarea distanţei pe baza traductorului HC – SR04 (similar aplicaţiei 3);
- plasarea cursorului de scriere cu ajutorul funcţie „lcd.setCursor ()”;
- afişarea unui mesaj şi a valorii măsurate pe ecran cu ajutorul funcţiei „lcd.print ()”;
- actualizarea valorilor numerice prin intermediul algoritmului „Leading Zeros Clearing”;

V. CONCLUZII

Aplicaţiile implementate pe baza microcontrolerelor pot funcţiona în mod independent de calculator dacă există mijloace periferice de intrare sau ieşire ataşate la terminalele microcontrolerului care pot înlocui rolul calculatorului gazdă.

Funcţionarea în mod independent de calculatorul gazdă al aplicaţiei finale presupune aplicarea unor algoritmi de optimizare atât al modului de funcţionare cât şi al modului de execuţie specific codului program.

VI. BIBLIOGRAFIE

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

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

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

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

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

Tema de studiu nr. 5 - Protocoale de comunicaţie

I. SCOPUL TEMEI

- prezentarea unor protocoale de comunicaţie în sisteme cu microprocesoare [1], [2]
- prezentarea unor dispozitive compatibile cu protocoalele de comunicaţie [3], [4], [5], [6]
- utilizarea microcontrolerelor pentru interfaţarea protocoalelor de comunicaţie [5], [6], [7]

II. INTRODUCERE [1], [2], [3]

Un **protocol de comunicaţie** reprezintă un **mod de asociere** şi **transferare de date** între **cel puţin două** sisteme de calcul, pe baza **circuitelor periferice** aflate în dotarea sistemului de calcul. **Cadrul de date** vehiculat între cele două echipamente reprezintă orice **combinaţie alfa-numerică** exprimată prin intermediul protocolului. O **grupare de cadre de date** reprezintă un **pachet de date**. În unele situaţii, cadrele de date reprezintă un singur bit.

În domeniul sistemelor cu microprocesoare există două categorii principale de protocoale de comunicaţie, anume:
- protocoale de comunicaţie cu transmitere de date în mod paralel;
- protocoale de comunicaţie cu transmitere de date în mod serial;

Protocoalele de comunicaţie cu transmitere de date în mod **paralel**, pot vehicula cadrele de date pe **mai multe canale de comunicaţie** în **mod simultan**. Mai precis prin intermediul a patru intrări / ieşiri digitale, se pot vehicula 2^4 combinaţii (Fig. 5.1).

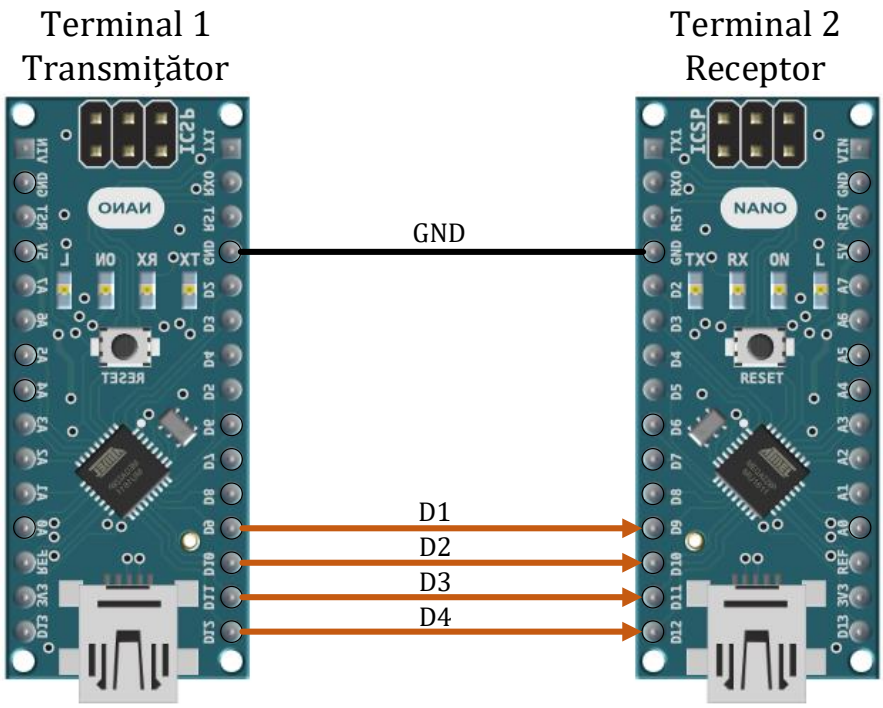


Fig. 5.1 – Protocolul de comunicaţie cu transmitere de date în mod paralel

Pentru a realiza asocierea între două sisteme de calcul de tip microcontroler, **Terminalul 1** va fi configurat prin intermediul codului program, ca şi **transmiţător**. Canalele digitale vor fi în acest sens **ieşiri digitale**. **Terminalul 2** va fi configurat prin intermediul codului program, ca şi **receptor**. Canalele digitale, în acest sens, vor fi configurate ca şi **intrări digitale**. Informaţia se va transmite în mod paralel sau simultan ca şi o **combinaţie de patru biţi**. Numărul maxim de combinaţii posibile în acest sens va fi 2^4 = 16 combinaţii. Protocolul de date în acest sens, poate fi considerat **unidirecţional**.

Protocoalele de comunicaţie cu transmitere de date în mod **serial** pot vehicula cadrele de date pe **două canale de comunicaţie** în **mod succesiv, serializat**. Terminalele de date pot avea rolul atât de transmiţător cât şi de receptor. Astfel, protocoalele de comunicaţie serial, pot fi **bidirecţionale**. În protocoalele seriale există **două canale** pentru vehicularea informaţiei (Fig. 5.2), anume:

- „TxD" (eng. Transmit „x" = Any Data – canal pentru transmisia oricărui tip de date);
- „RxD (eng. Receive „x" = Any Data – canal pentru recepţionarea oricărui tip de date);

Canalul pentru transmisia oricărui tip de date „TxD" este reprezentat printr-o ieşire digitală, iar datele sunt transmise sub formă de impulsuri (bit cu bit). Canalul pentru recepţionarea oricărui tip de date „RxD" este reprezentat printr-o intrare digitală, iar datele sunt recepţionate în mod serializat (bit cu bit).

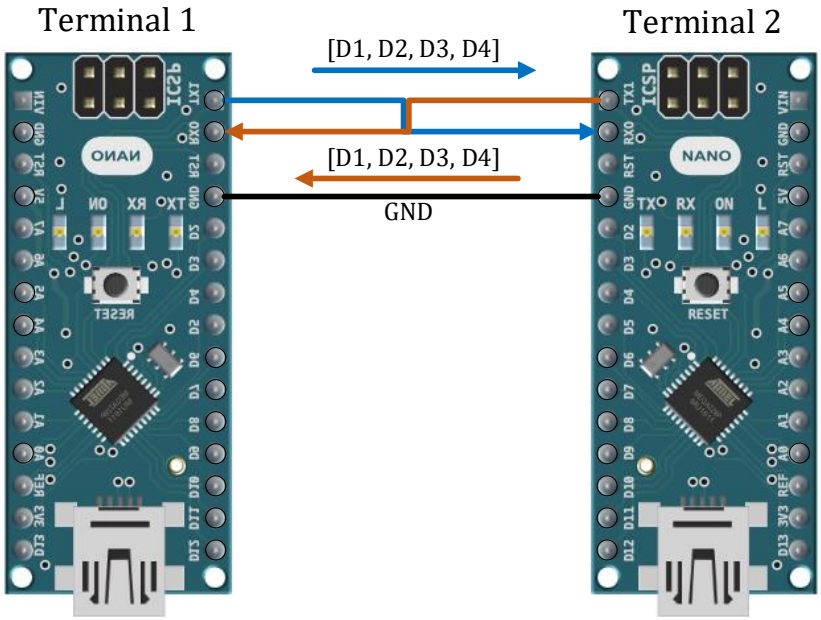


Fig. 5.2 – Protocolul de comunicaţie cu transmitere de date în mod serial

Protocoalele de tip serial nu pot vehicula în mod simultan datele pe magistrala de comunicaţie, dar în schimb conferă siguranţa suspendării procesului de comunicaţie în cazul întreruperii canalelor de comunicaţie. În cazul comunicaţiei de tip paralel, la întreruperea unui canal digital de comunicaţie (întreruperea unui fir de date), informaţia se va transfera în mod eronat la terminalul receptor.

Comunicația de tip serial standard asincron (eng. Universal Asynchronous Receiver – Transmitter – UART) reprezintă modelul de bază al tuturor protocoalelor de comunicație de tip serial (Fig. 5.3) și poate fi reprezentat la următoarele etape:

A. Etapa de transmitere implică:

- 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;

B. Etapa de recepționare implică:

- 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;

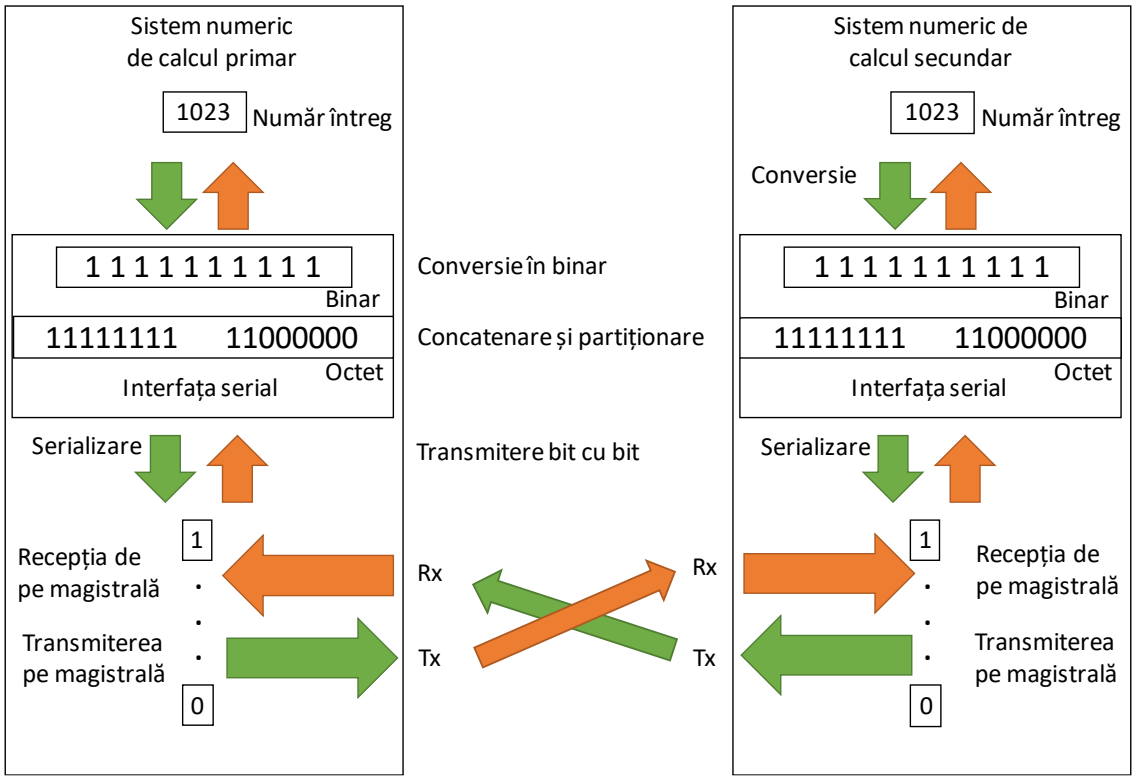


Fig. 5.3 – Modelul pentru vehicularea datelor în procesul de comunicație serial [3]

OBSERVAȚIE: În exemplul de mai sus (Fig. 5.3), este ilustrat faptul că, pe magistrala de date serial, pot fi vehiculate și alte tipuri de date având dimensiunea mai mare decât opt biți, dimensiunea maximă a unui pachet de date.

III. ASPECTE TEORETICE [1], [2], [3], [4]

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;**

A. Protocolul Serial standard asincron UART

- reprezintă un protocol **fizic** (eng. **hardware**) de comunicaţie;
- se realizează pe **două fire**: „Tx" – eng. Transmit Data şi „Rx" – eng. Receive Data (Fig. 5.4);
- face parte din clasa protocoalelor „**punct la punct**" (eng. Point To Point – PTP), adică, protocolul nu funcţionează decât între **cel puţin şi cel mult două terminale**;
- poate fi optimizat pentru a vehicula datele pe **distanţe medii** între două terminale, însă **volumul de date este redus**;
- poate funcţiona în regim **bidirecţional**;

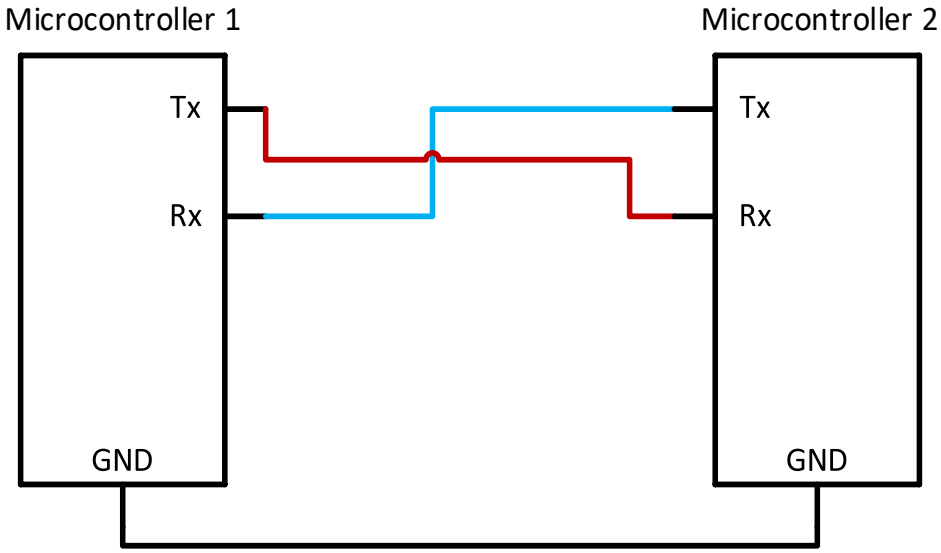


Fig. 5.4 – Conexiunea între terminale în cadrul protocolului serial standard asincron

Acest tip de protocol se utilizează pentru:

- transfer de date între două sisteme de calcul;
- diagnoză şi depanare asistată software (ex. Consola Serial + Text Mode);
- trecerea de la un mediu de comunicare la altul (ex. Bluetooth – Serial HC-05);

B. Protocolul Serial sincron adresabil I²C

- reprezintă un protocol fizic (eng. hardware) de comunicaţie;
- se realizează pe două fire: „SCL" – eng. **S**erial **CL**ock şi „SDA" – eng. **S**erial **DA**ta (Fig. 5.5);
- face parte din clasa protocoalelor **adresabile** de tip eng. „**Master – Slave**", şi poate deservi **mai mult decât două terminale**. Echipamentele sunt cuplate **în paralel** pe o **magistrală comună**, având **adrese distincte**;
- poate vehicula datele pe distanţe relativ reduse, în volum relativ mediu, la viteze cuprinse în intervalul 100 [kb / s] până la 3,5 [Mb / s];
- în vederea generării unei diferenţe de potenţial la nivelul canalelor de comunicaţie sunt necesare două rezistenţe pentru conectare la potenţialul sursei (eng. pull – up);

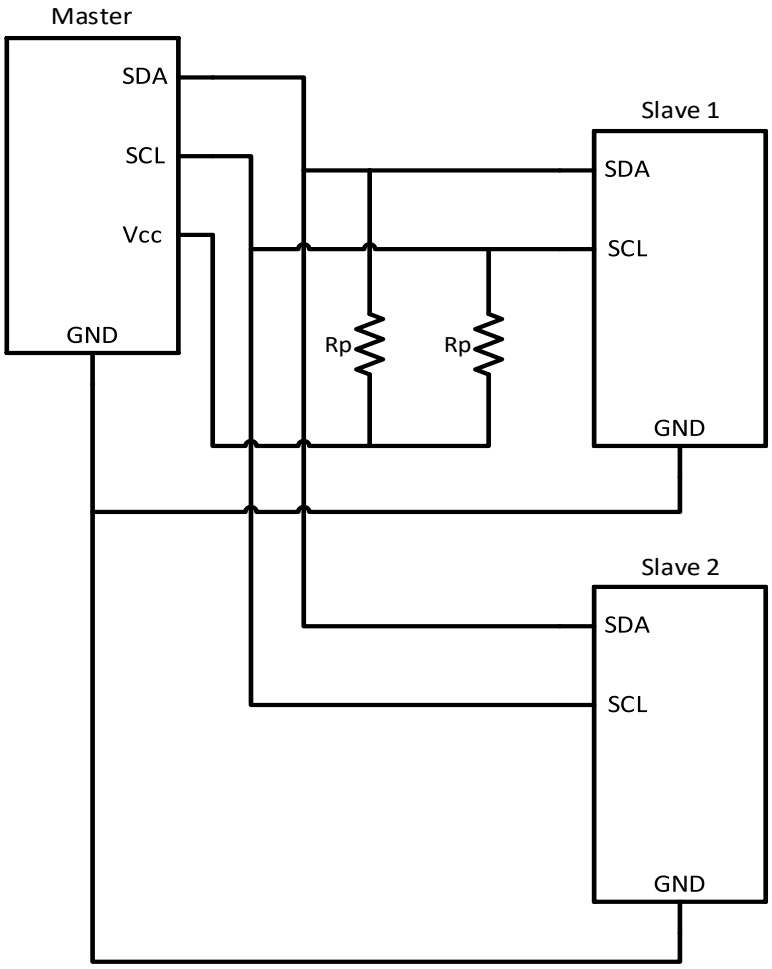


Fig. 5.5 - Conexiunea între terminale în cadrul protocolului serial sincron I²C

Acest tip de protocol se utilizează pentru:
- conectarea diverselor echipamente compacte la micro-controller (ex. LCD, senzori);
- inter-conectarea mai multor sisteme de calcul în vederea cooperării (Master / Slave);
- integrare de module şi sisteme (ex. echipamente sau instalaţii multi-modulare) (Fig. 5.6);

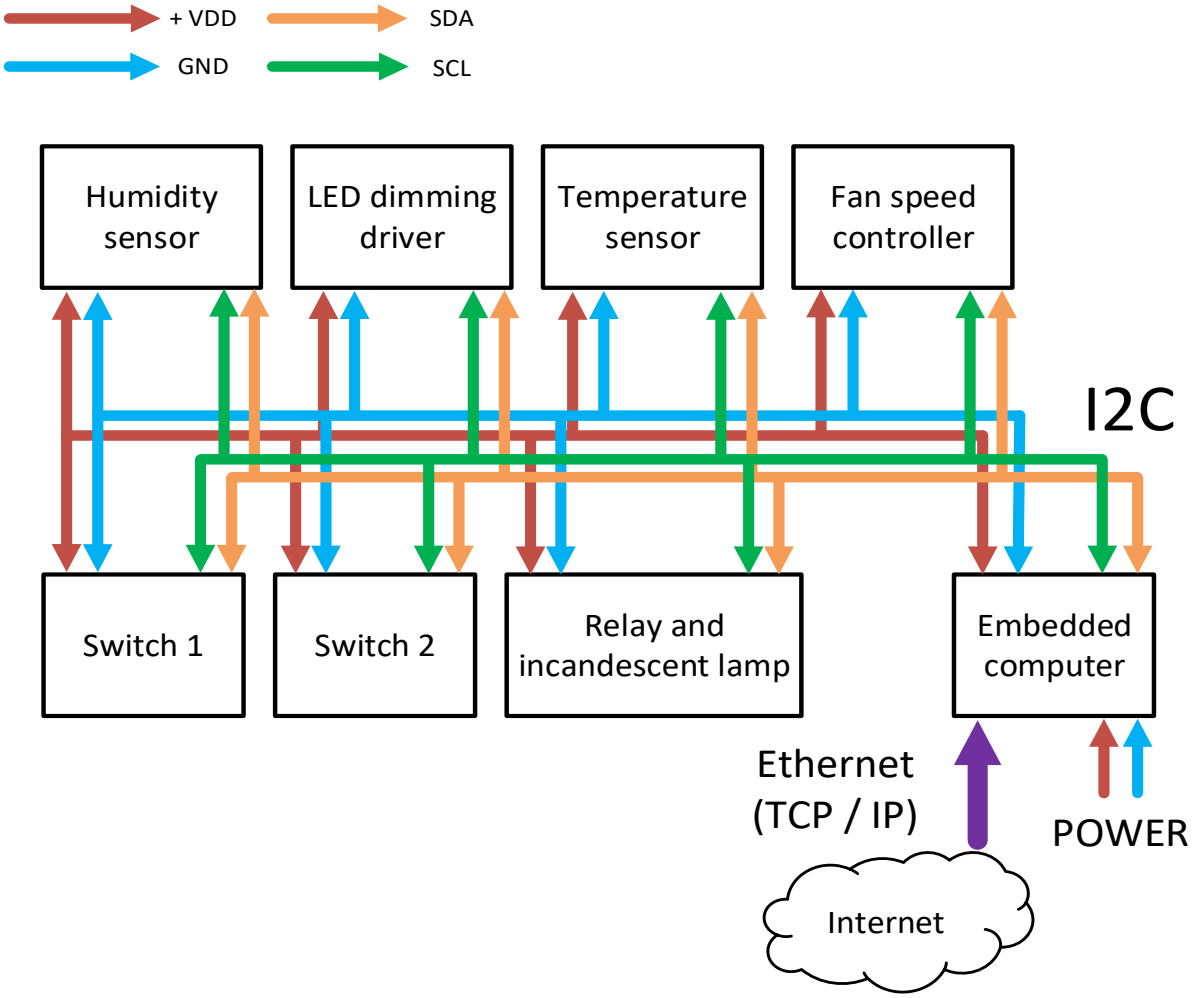


Fig. 5.6 – Automatizarea sistemelor pentru asigurarea confortului în domeniul rezidențial

C. Protocolul Serial standard sincron SPI

- reprezintă un protocol **fizic** (eng. hardware) de comunicație;
- se realizează pe **patru canale** de transmiterea datelor (Fig. 5.7):
  - „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" – eng. **S**lave **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];

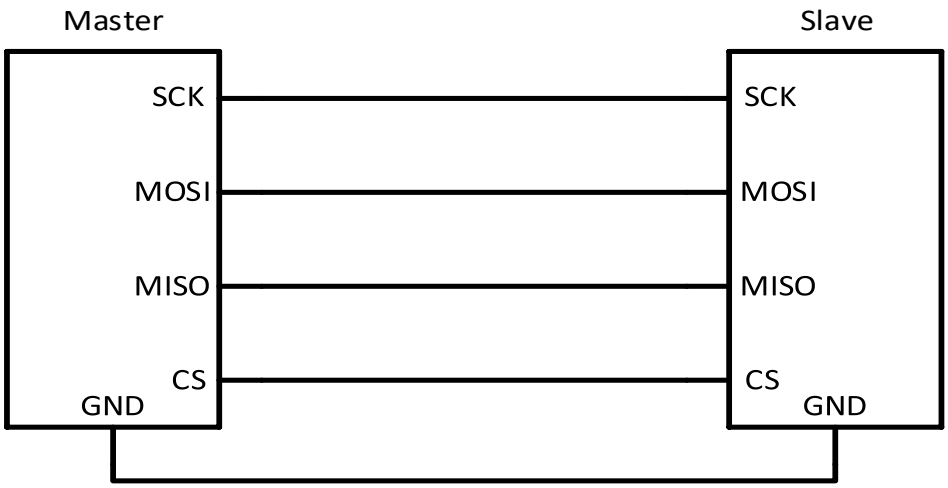


Fig. 5.7 - Conexiunea între terminale în cadrul protocolului serial sincron SPI

Acest tip de protocol se utilizează pentru:
- transfer de date între sisteme de calcul;
- programarea micro-controllerelor şi transferul de date direct în memorie;
- trecerea de la un mediu de comunicare la altul şi conectarea altor echipamente de expansiune la sistemul de calcul dat (ex. eng. Ethernet – SPI Shield, SD card etc.);

D. Protocolul Serial asincron 1-Wire (One Wire)

- reprezintă un protocol fizic (eng. hardware) de comunicaţie;
- se realizează prin intermediul unui singur canal de transmiterea datelor (Fig. 5.8);
- funcţionează în modul SMMS (eng. „Single Master and Multiple Slaves”);
- funcţionează în modul (eng. half - duplex), adică, un singur dispozitiv este capabil să iniţieze procesul de comunicaţie şi să transfere date în mod bidirecţional în acelaşi timp;
- prezintă viteză de transfer redusă (aprox. 15,4 [kb / s]);
- poate transfera un volum redus de date;

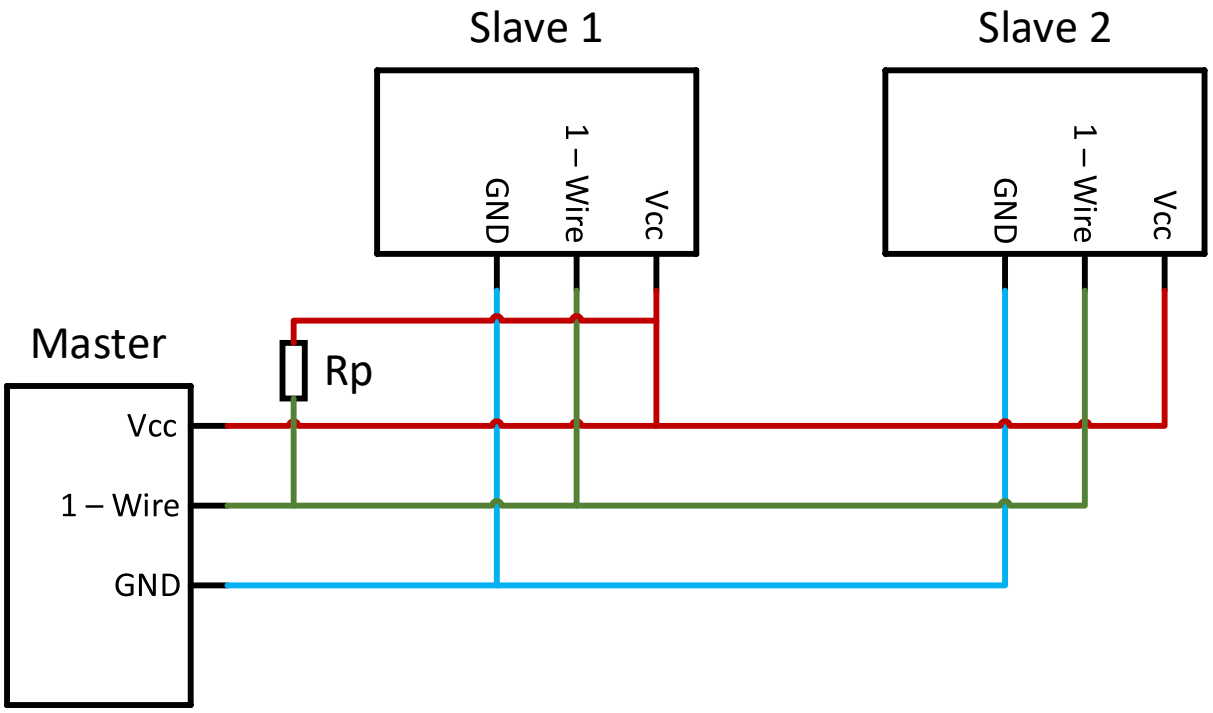


Fig. 5.8 - Conexiunea între terminale în cadrul protocolului serial asincron One Wire

Acest tip de protocol se utilizează pentru:
- transfer de date între sisteme de calcul;
- interfațarea traductoarelor cu sistemele care nu prezintă convertor analog - digital;
- monitorizarea parametrilor de funcţionare pentru sursele de alimentare în comutație, destinate alimentării echipamentelor electronice portabile;

IV. IMPLEMENTAREA APLICAȚIILOR [4], [5], [6], [7]

În vederea implementării aplicațiilor din cadrul acestei lucrări vor fi utilizate două module specializate precum:
- modulul adaptor Bluetooth – Serial HC-05;
- traductorul digital DHT-11;

A. Modulul adaptor Bluetooth – Serial HC-05 [5]

Modulul HC-05 este un adaptor Bluetooth - Serial (eng. Serial Port Protocol - SPP), conceput pentru configurarea transparentă a conexiunii Serial fără fir. Este compatibil cu standardul „Bluetooth V2.0" și „EDR" (eng. Enhanced Data Rate). Tehnica de modulație se realizează la viteza de 3 [Mb / s] prin intermediul unui dispozitiv transmițător și receptor (eng. transceiver) radio cu bandă de frecvență 2,4 [GHz]. Acest modul înglobează de asemenea circuitul integrat specializat pentru aplicații în domeniul radio „Bluetooth CSR Bluecore 04 – External" cu construcție monobloc și implementare în tehnologie CMOS și tehnică de modulare AFH (eng. Adaptive Frequency Hopping Feature) (Fig. 5.9).

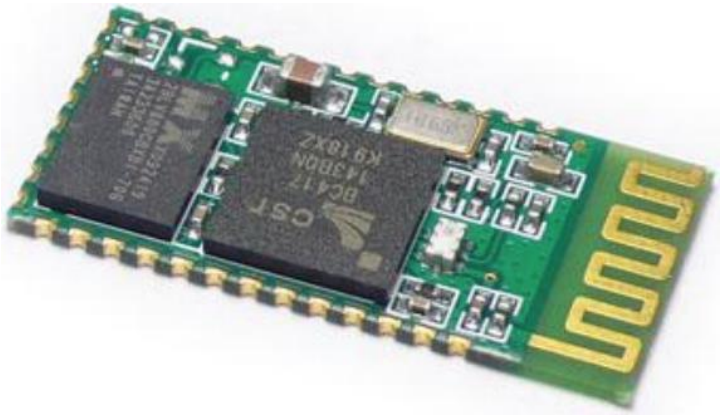


Fig. 5.9 – Modulul adaptor Bluetooth – Serial HC – 05 [5]

Modulul HC-05 prezintă următoarele dotări şi specificaţii:
- nivel de sensibilitate -80 [dBm];
- putere de transmitere +4 [dBm];
- consum redus de putere, de la 1,8 [V] până la 3,6 [V] pe ieşirile digitale;
- compatibil cu protocolul serial asincron UART;
- antenă încorporată;

B. Traductorul digital DHT-11 [6]

Senzorul de temperatură şi umiditate DHT11 furnizează mărimea rezultantă sub forma unui semnal digital. Acesta poate fi interpretat prin intermediul unei biblioteci de funcţii (DHT.h) furnizată de către producător. Semnalul digital este generat prin intermediul unui microcontroler integrat în capsula traductorului. Protocolul de comunicaţie implementat la nivelul traductorului este serial de tip One Wire (Fig. 5.10).

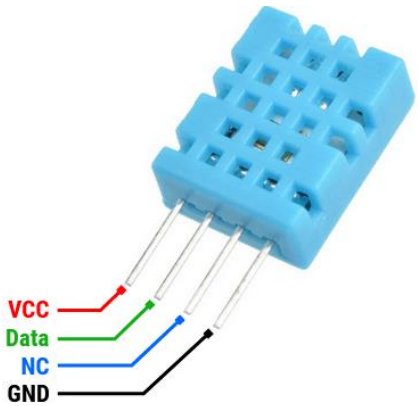


Fig. 5.10 – Traductorul de umiditate şi temperatură DHT - 11 [6]

Pentru a evidenția conceptele prezentate anterior, se propune deci implementarea următoarelor aplicații cu ajutorul platformei Arduino Nano:
- recepționarea datelor prin interfața serial de la microcontroler;
- transmiterea datelor prin intermediul interfeței serial la microcontroler;
- recepționarea datelor prin intermediul modulului Bluetooth HC-05;
- transmiterea datelor prin intermediul modulului Bluetooth HC-05;
- preluarea datelor de la traductorul de umiditate și temperatură DHT-11;
- implementarea unei aplicații de monitorizare multipunct a parametrilor ambientali

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;
- traductor pentru măsurarea temperaturii și a umidității ambientale 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;

APLICAȚIA 1

Se va implementa circuitul conform următoarei scheme (Fig. 5.11):

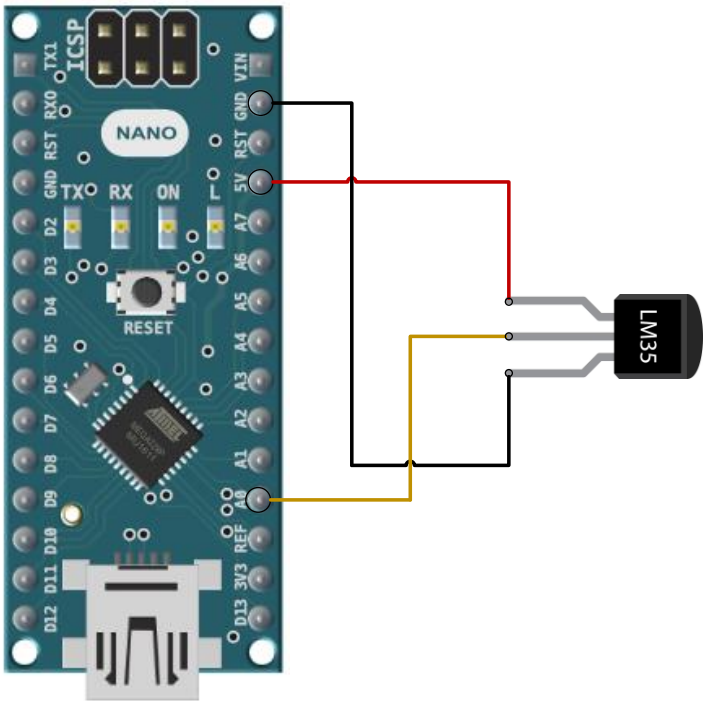


Fig. 5.11 – Schema electronică pentru implementarea circuitului specific aplicației 1

Se va realiza următorul montaj experimental (Fig. 5.12):

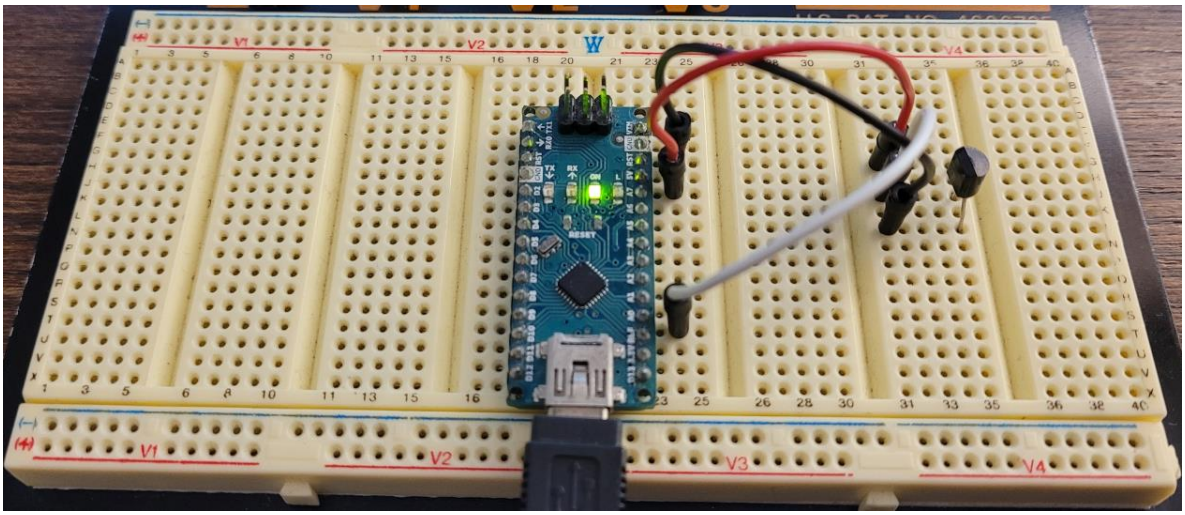


Fig. 5.12 – Montajul experimental specific aplicației 1

Se va implementa următorul cod program:

```
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);
}
```

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 (Fig. 5.13);

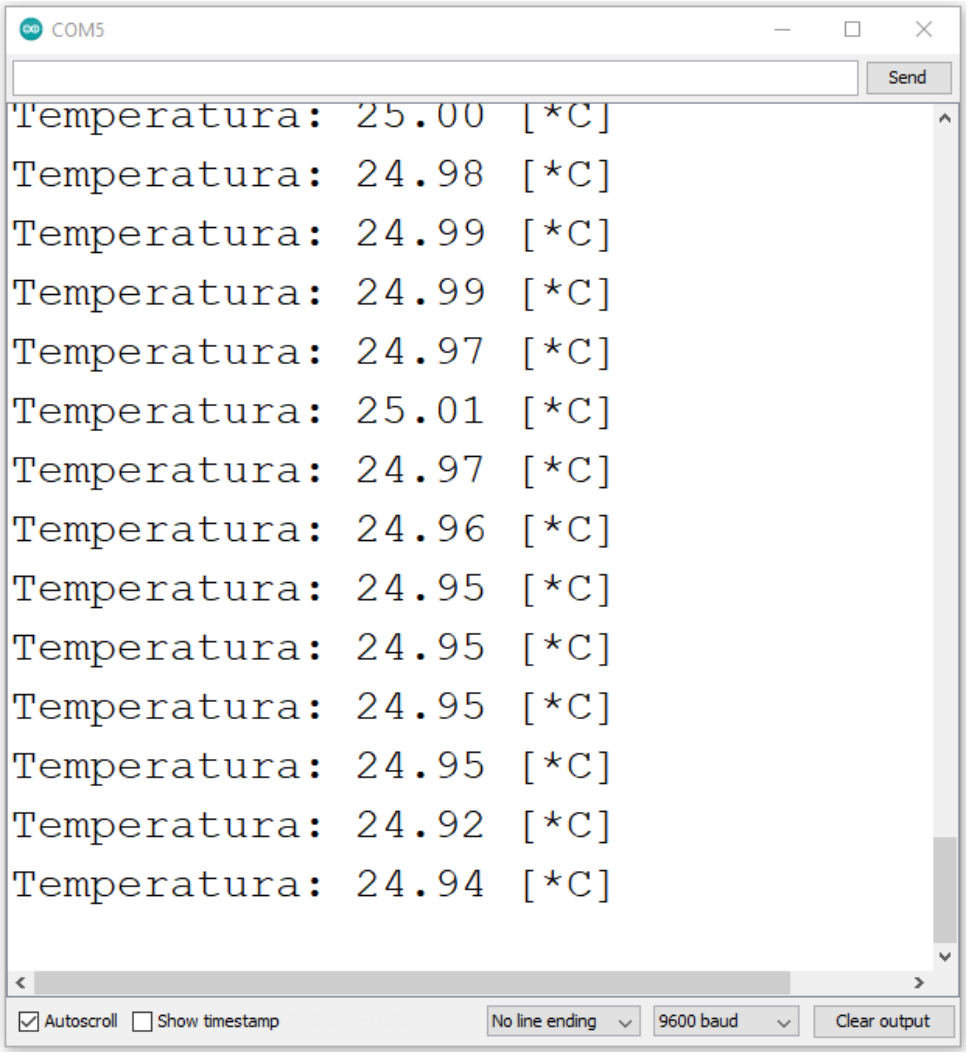


Fig. 5.13 – Afișarea mesajelor în consola serial

APLICAȚIA 2

Se va implementa circuitul conform următoarei scheme (Fig. 5.14):

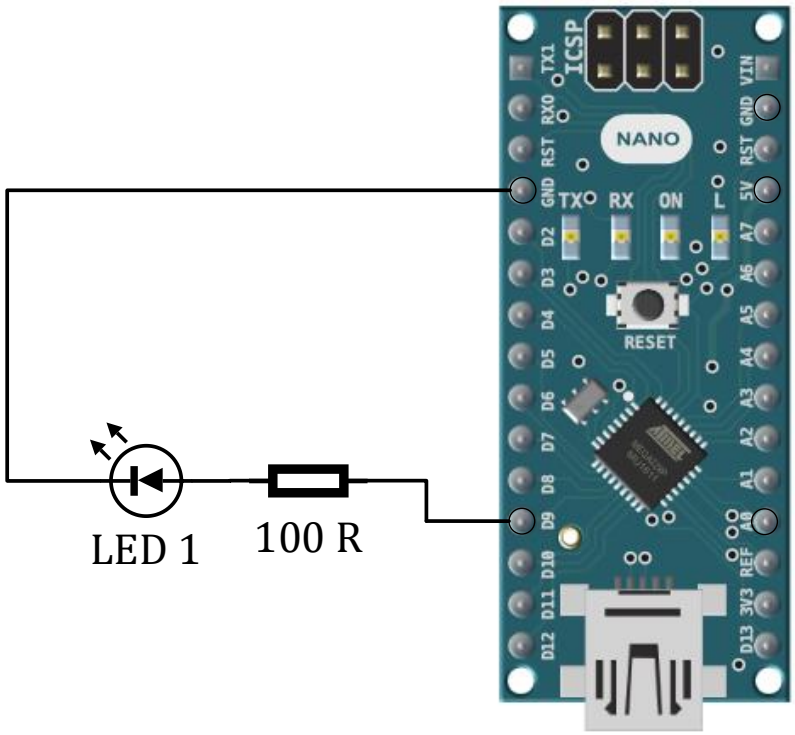


Fig. 5.14 – Schema electronică pentru implementarea circuitului specific aplicației 2

Se va realiza următorul montaj experimental (Fig. 5.15):

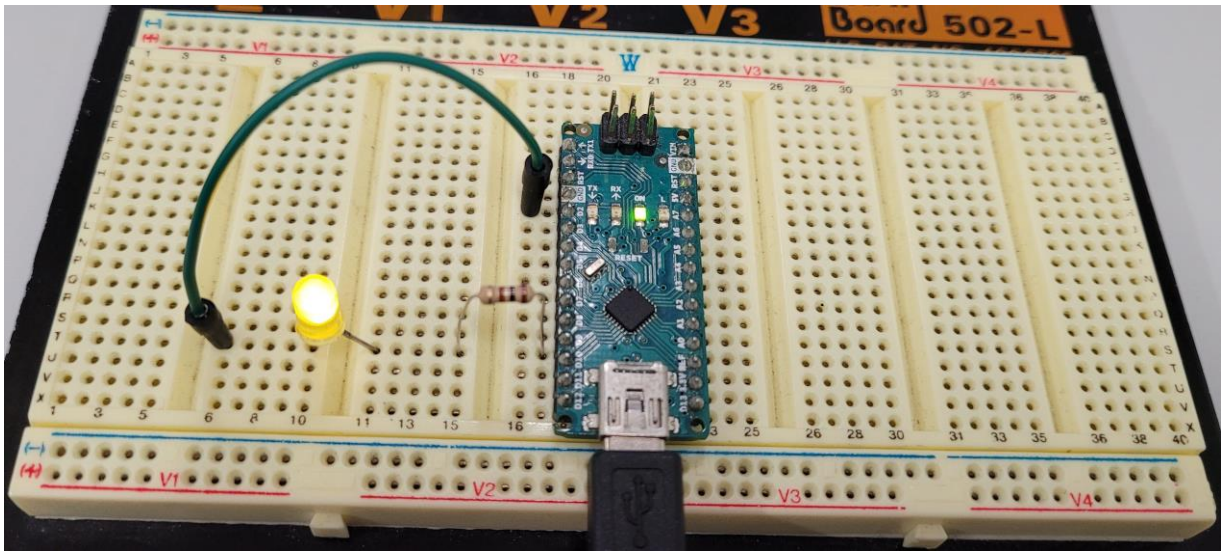


Fig. 5.15 – Montajul experimental specific aplicației 2

Se va implementa următorul cod program:

```
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 || 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);
}
```

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”;

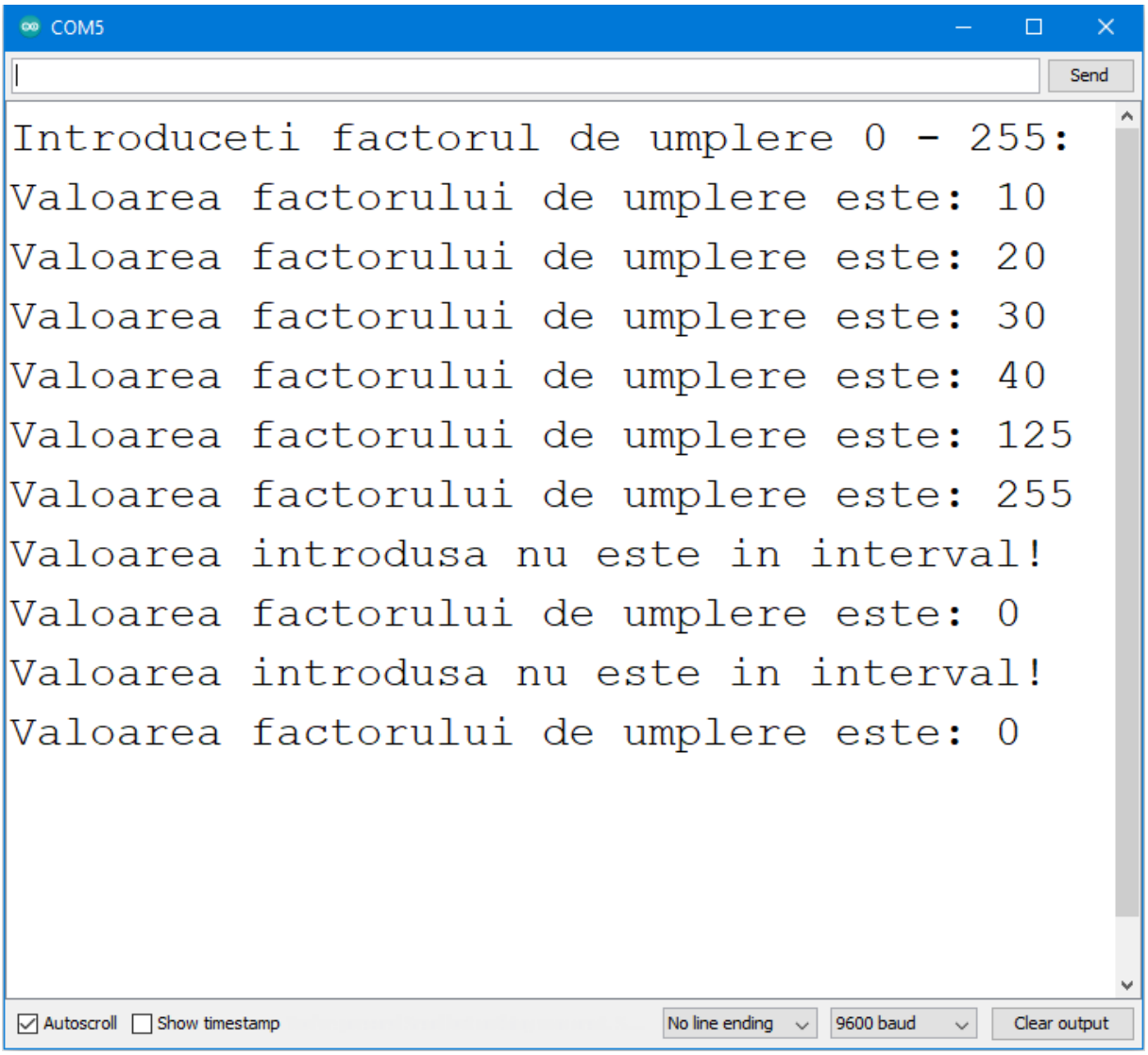


Fig. 5.16 – Afișarea factorului de umplere sau a duratei de conducție în consola Serial

APLICAȚIA 3

Se va implementa următorul cod program:

```
#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);
}
```

Se va implementa circuitul conform următoarei scheme (Fig. 5.17):

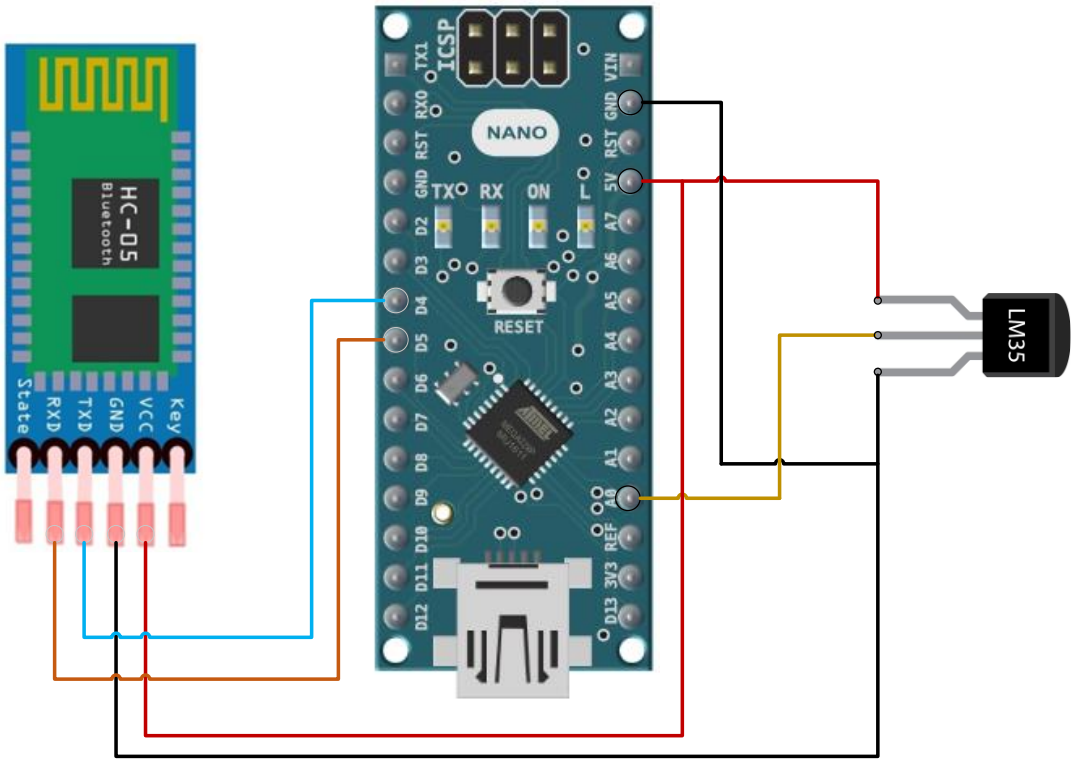


Fig. 5.17 - Schema electronică pentru implementarea circuitului specific aplicației 3

Se va realiza următorul montaj experimental (Fig. 5.18):

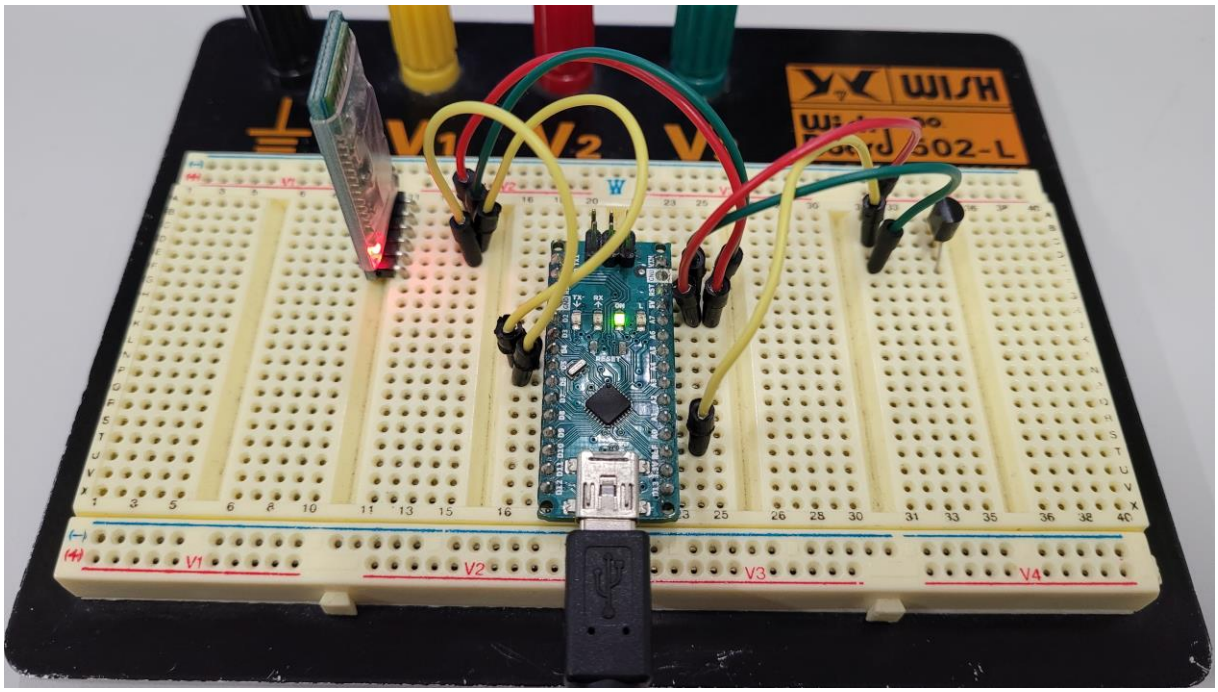


Fig. 5.18 - Montajul experimental specific aplicației 3

OBSERVAȚII:

1. Prin intermediul bibliotecii de funcții „SoftwareSerial.h” se va inițializa comunicația serial în mod virtual prin intermediul a două terminale digitale la alegere (în cazul de față „D4” – RxD iar „D5” – TxD).

2. Pentru a verifica funcționalitatea modulului Bluetooth – Serial HC-05, se va utiliza aplicația „Bluetooth Serial Terminal” pe telefonul mobil. Prin intermediul acestei aplicații va fi posibilă recepționarea mesajelor din consola serial pe telefonul mobil (Fig. 5.19).

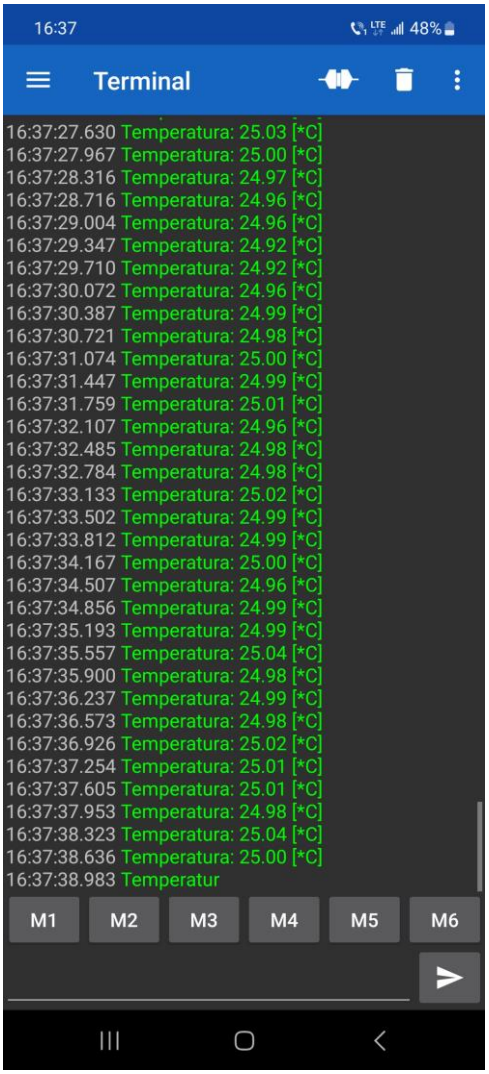


Fig. 5.19 – Aplicația Bluetooth Serial Terminal – vizualizarea conținutului consolei serial

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.

APLICAŢIA 4

Se va implementa circuitul conform următoarei scheme (Fig. 5.20):

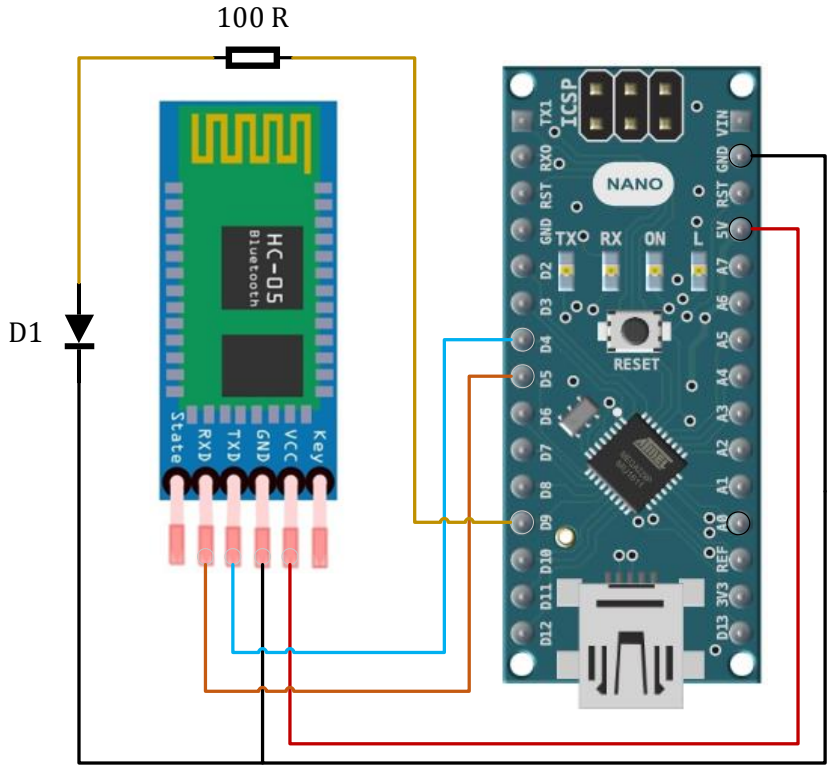


Fig. 5.20 - Schema electronică pentru implementarea circuitului specific aplicaţiei 4

Se va realiza următorul montaj experimental (Fig. 5.21):

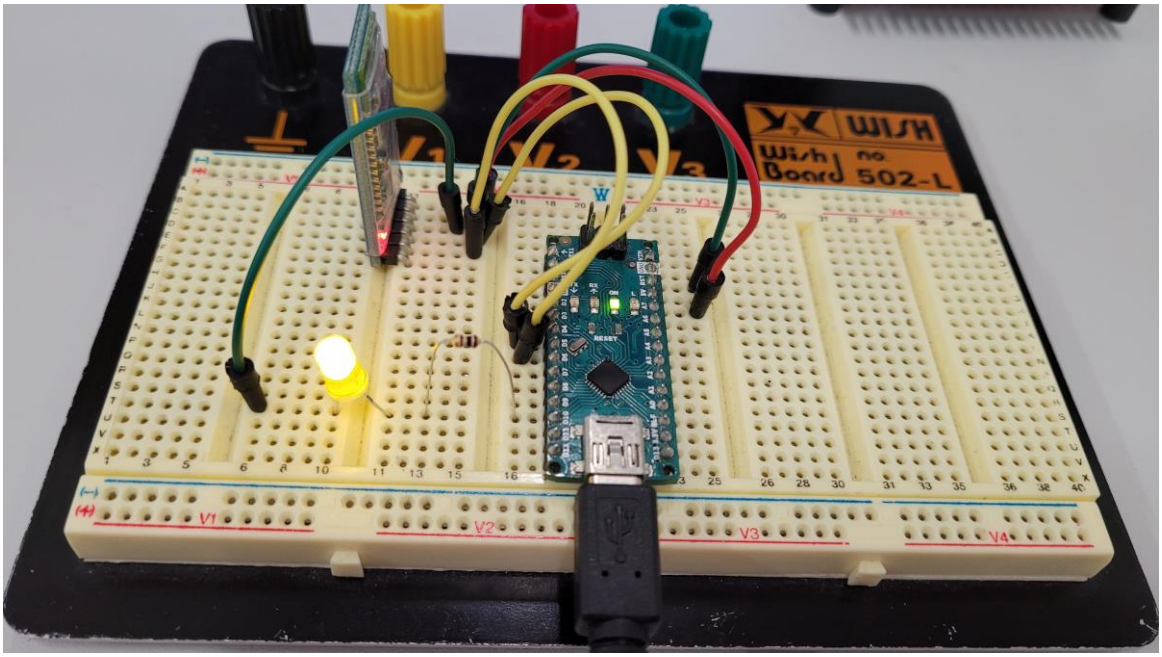


Fig. 5.21 - Montajul experimental specific aplicaţiei 4

Se va implementa următorul cod program:

```
#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 || 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 || 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);
}
```

Implementarea aplicaţiei nr. 4 presupune:

- iniţializarea comunicaţiei Serial la viteza de transfer 9600 [b/s]
- iniţializarea comunicaţiei serial atât locală cât şi la distanţă;
- verificarea disponibilităţii datelor de pe magistrala serial cu funcţia „Serial.available()”;
- verificarea disponibilităţii datelor de pe magistrala cu funcţia „BTSerial.available()”;
- preluarea factorului de umplere de la magistrala serial cu funcţia „Serial.parseInt()”;
- preluarea factorului de umplere de la magistrala serial cu funcţia „BTSerial.parseInt()”;
- afişarea valorii exprimată pe 8 biţi a duratei de conducţie în consola serial;
- afişarea valorii pe 8 biţi a duratei de conducţie în aplicaţia Bluetooth Serial Terminal;
- ajustarea factorului de umplere pentru un tren de impulsuri furnizat pe terminalul „9”;
- preluarea atât local cât şi la distanţă a datelor numerice din consolă (Fig. 5.22);
- afişarea locală a mesajului pentru monitorizare în consola serial;
- afişarea în aplicaţia Bluetooth Serial Terminal a conţinutului consolei serial.

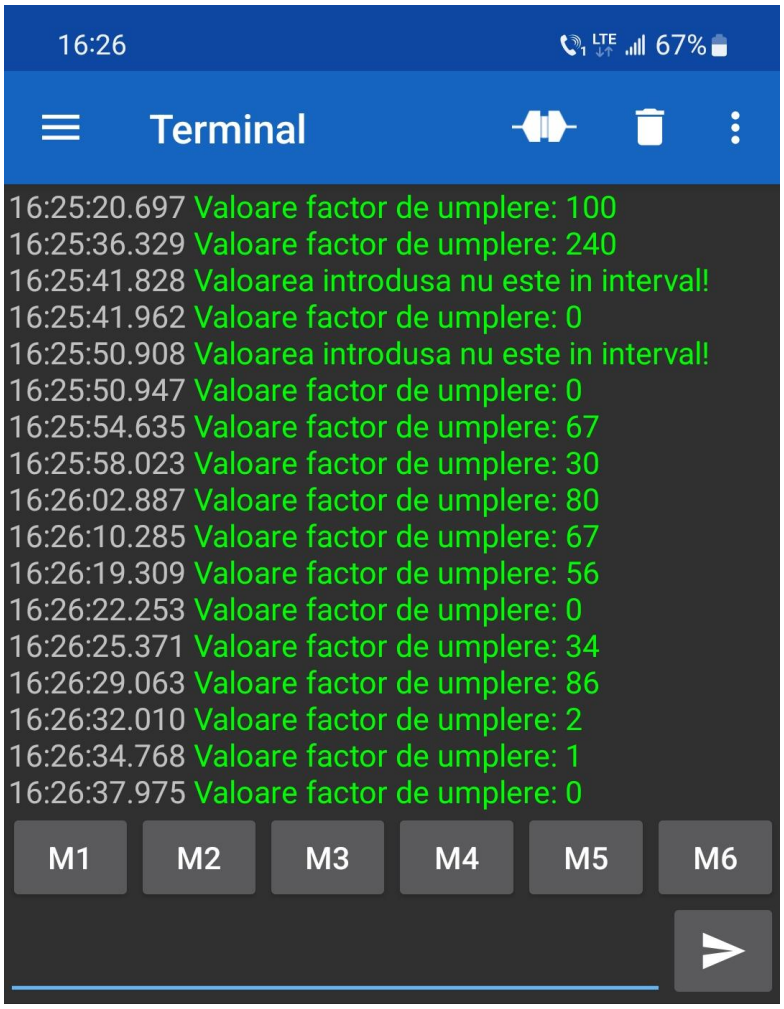


Fig. 5.22 – Aplicaţia Bluetooth Serial Terminal – preluarea valorii factorului de umplere şi afişarea în consolă a mesajelor

APLICAŢIA 5

Se va implementa circuitul conform următoarei scheme (Fig. 5.23):

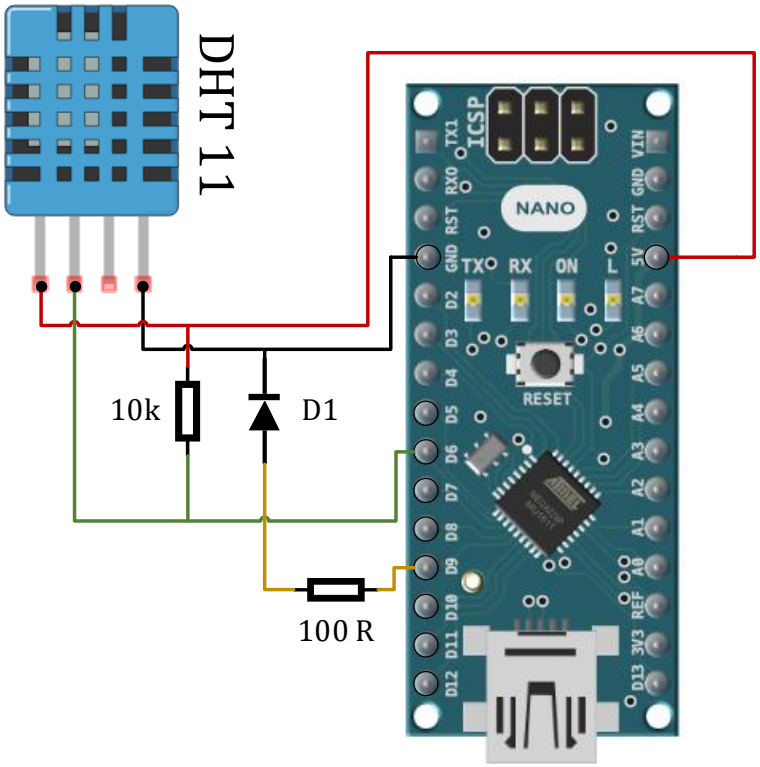


Fig. 5.23 - Schema electronică pentru implementarea circuitului specific aplicaţiei 5

Se va realiza următorul montaj experimental (Fig. 5.24):

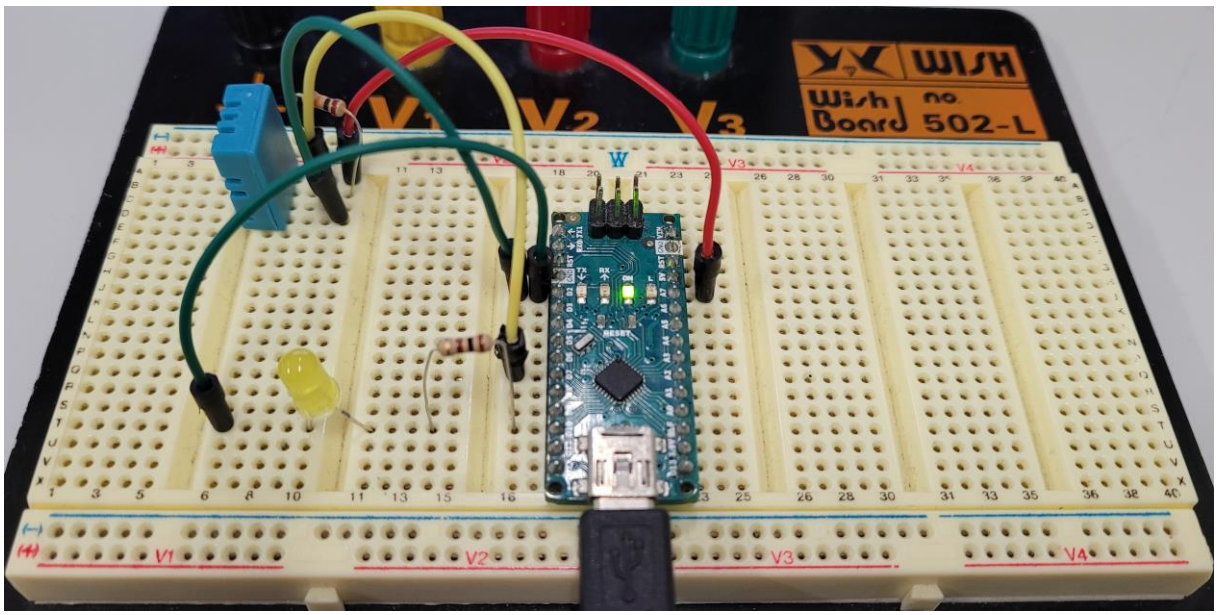


Fig. 5.24 - Montajul experimental specific aplicaţiei 5

Se va implementa următorul cod program:

```
#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);
}
```

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 (Fig. 5.25);
- semnalizarea prin intermediul unui LED de stare la fiecare achiziție;

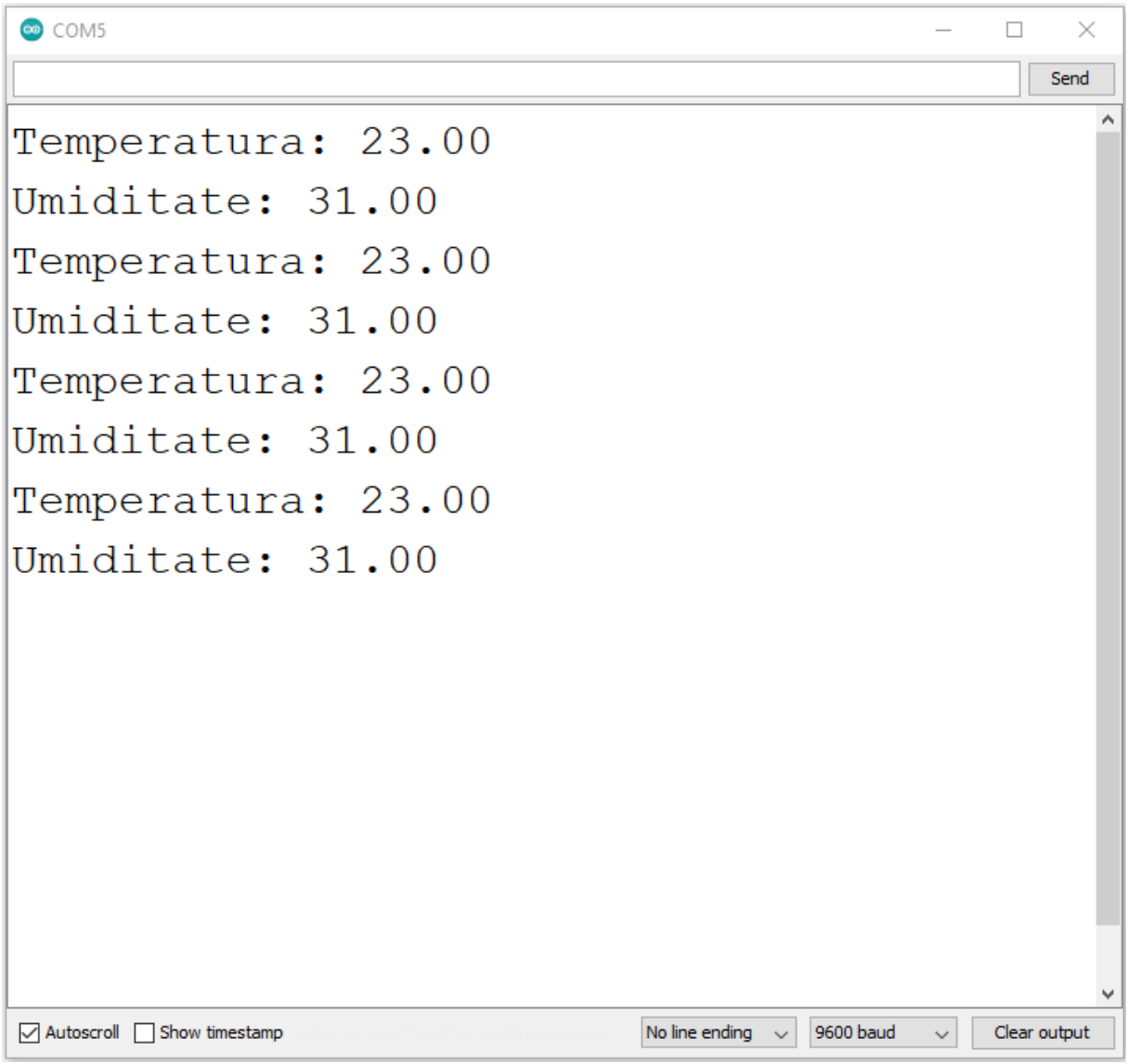


Fig. 5.25 – Consola serial – preluarea datelor de la traductorul digital de umiditate și temperatură DHT 11 și afișarea lor în consolă

APLICAȚIA 6

Se va implementa circuitul conform următoarei scheme (Fig. 5.26):

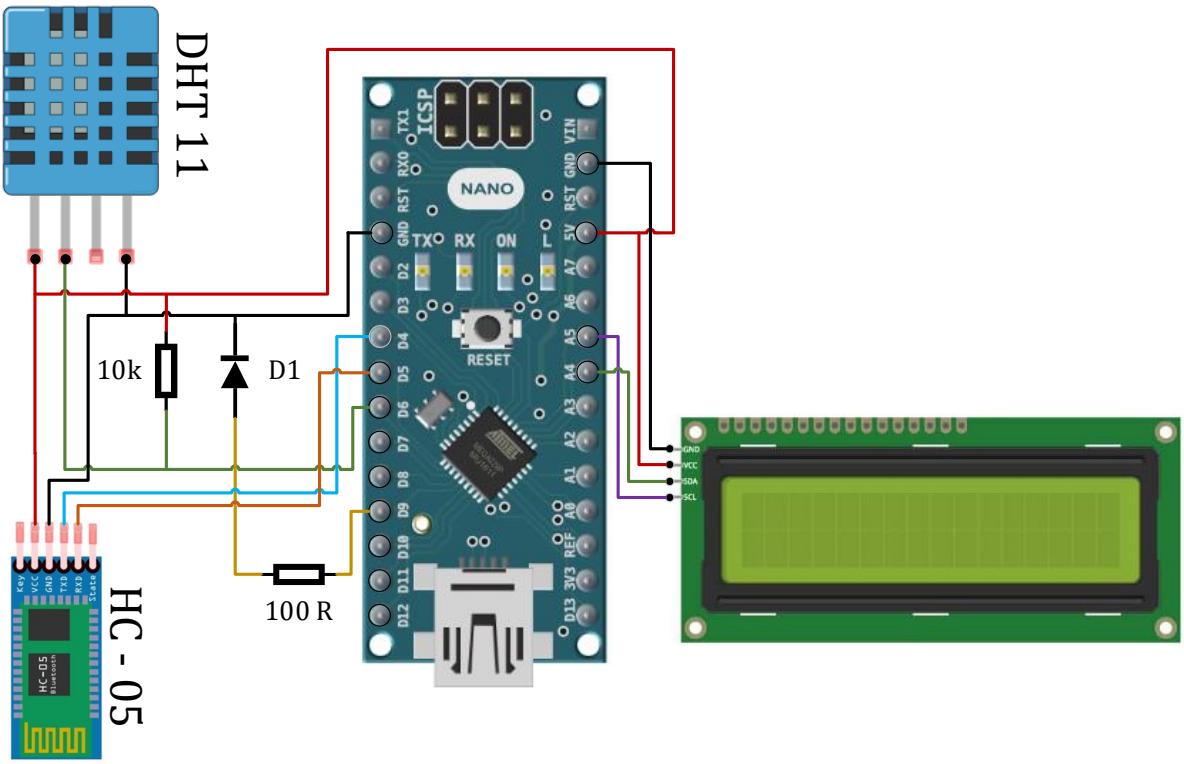


Fig. 5.26 - Schema electronică pentru implementarea circuitului specific aplicației 6

Se va realiza următorul montaj experimental (Fig. 5.27):

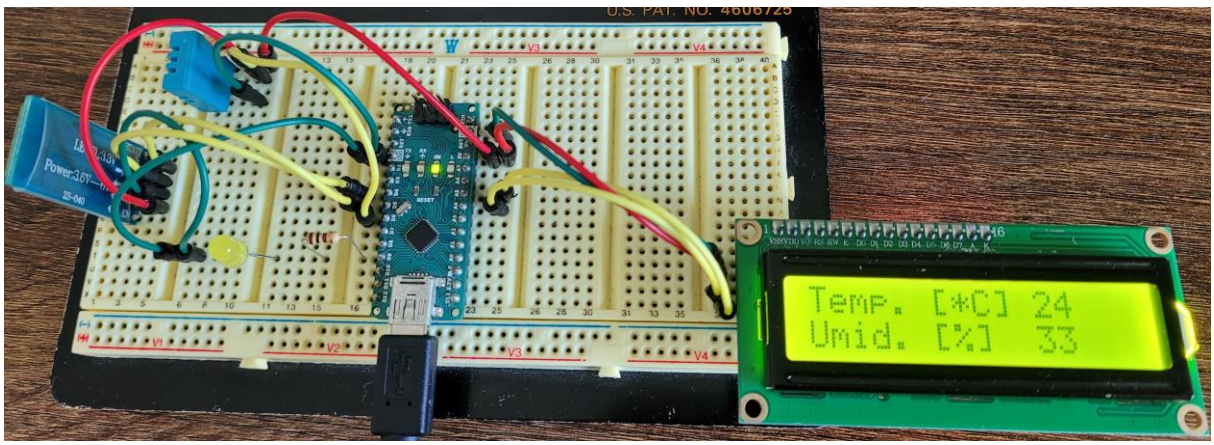


Fig. 5.27 - Montajul experimental specific aplicației 6

Se va implementa următorul cod program:

```
#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);
  }

  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);
}
```

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;

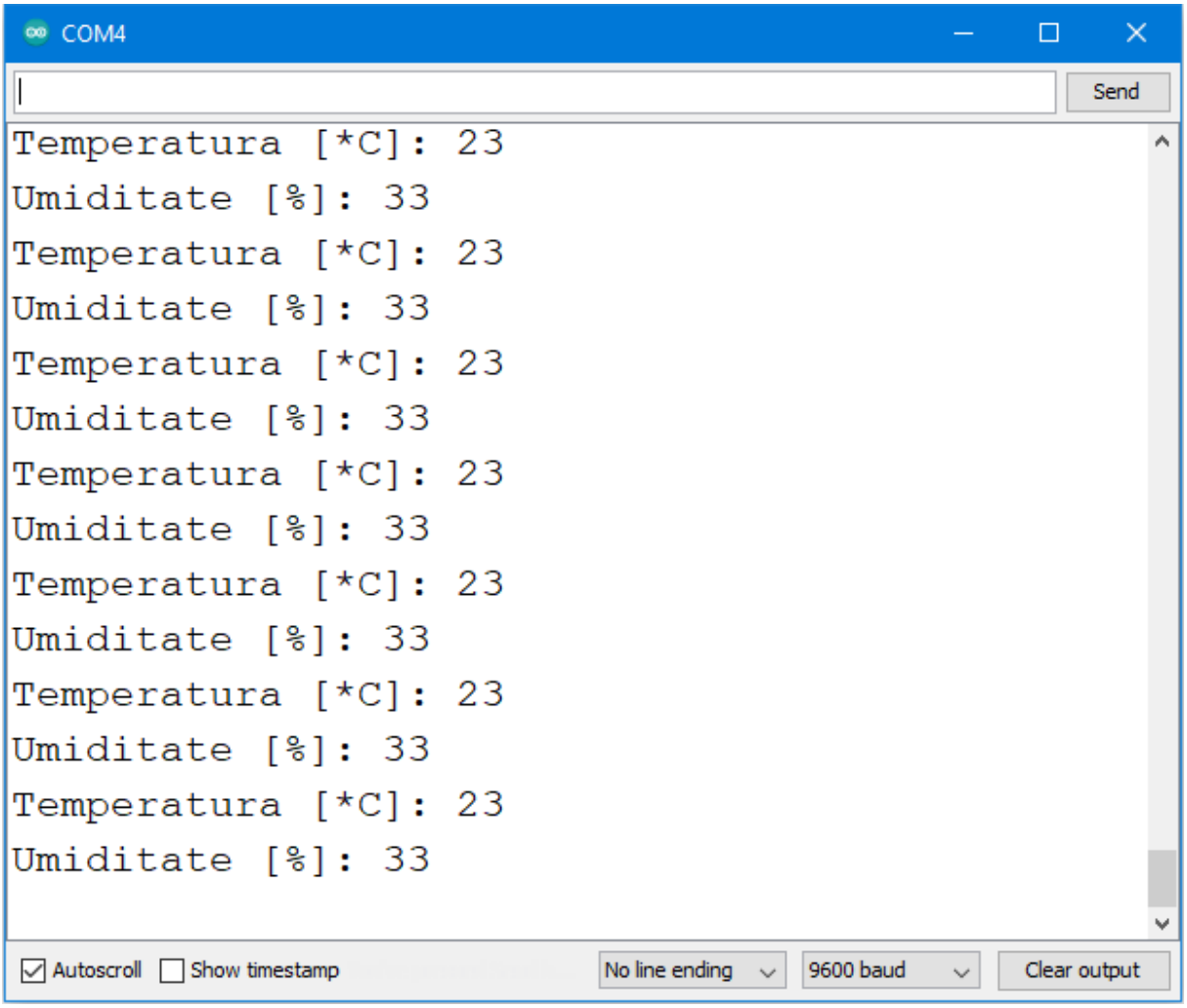


Fig. 5.28 – Consola serial – preluarea datelor de la traductorul digital de umiditate și temperatură DHT 11 și afișarea lor în consolă

OBSERVAȚIE: Această aplicație poate fi considerată o modalitate de tip „multipunct" de supraveghere și monitorizare a parametrilor ambientali, precum temperatura și umiditatea aerului din încăpere, deoarece, informația finală poate fi preluată atât de la fața locului (prin intermediul afișajului LCD – fizic, eng. hardware) cât și de la consola serial a calculatorului sau a telefonului mobil (logic sau virtual - eng. software). Funcționalitatea „multipunct" a aplicației poate deservi de asemenea procesul și metodele de depanare și diagnosticare a unui echipament complex (eng. debugging methods).

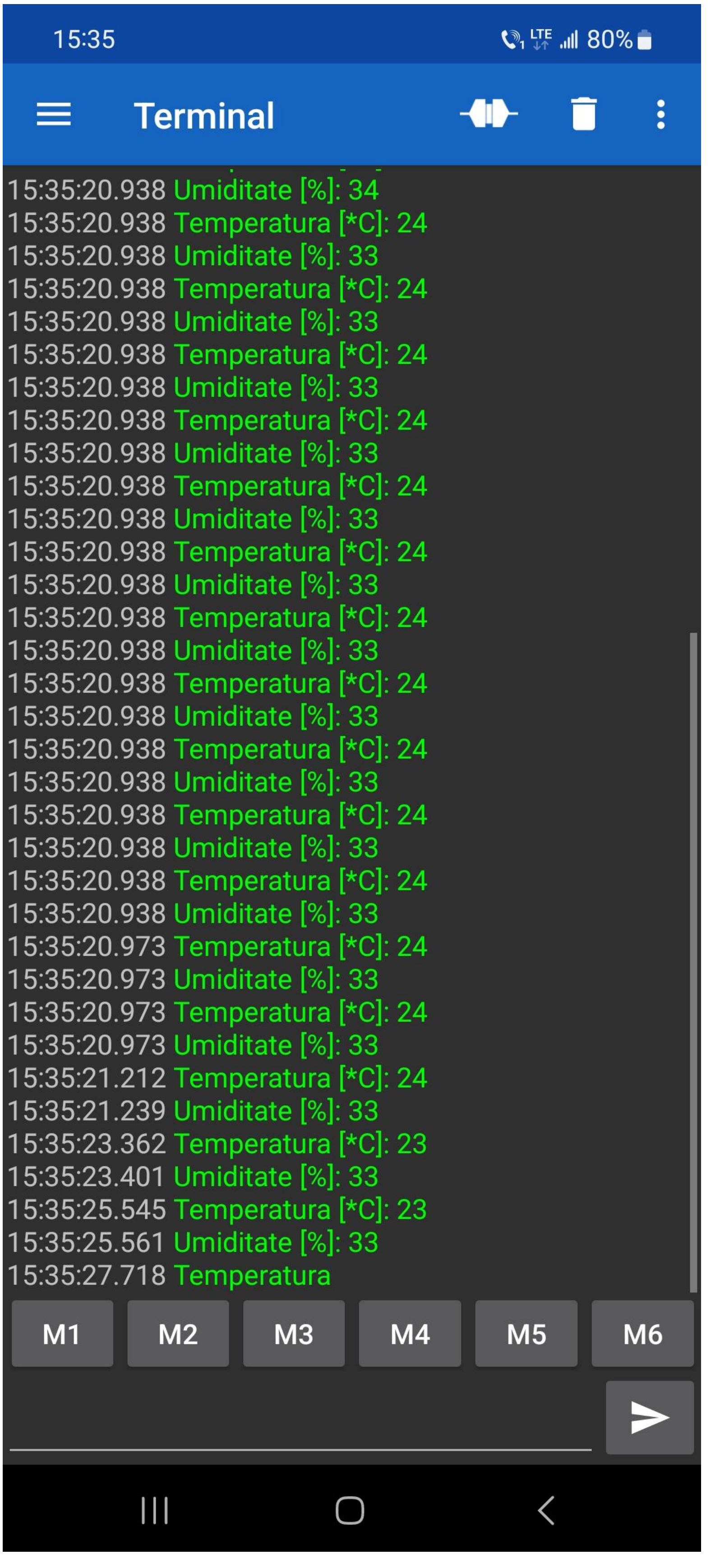


Fig. 5.29 - Aplicația Bluetooth Serial Terminal – afișarea valorilor de temperatură și umiditate preluate de la traductorul DHT-11

V. CONCLUZII

Prin intermediul protocoalelor de comunicaţie implementate atât la nivelul codului program (eng. software protocol) cât şi la nivelul fizic (eng. hardware protocol) al unui sistem de calcul, pot fi realizate următoarele funcţii:
- diagnoză şi depanare (eng. debugging and repairing);
- monitorizarea la distanţă a unor parametrii (eng. remote monitoring);
- accesul multipunct la mijloacele fizice ale aplicaţiei (eng. multi point operation);
- traversarea sau conversia de la un protocol la altul (ex. adaptorul Bluetooth – Serial);
- transportarea datelor pe distanţe mari în format digital codificat;

VI. BIBLIOGRAFIE

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

2. Universitatea Politehnica Timişoara – „Interfaţarea circuitelor - Tipuri de interfeţe, Protocoale seriale de comunicare" - https://www.aut.upt.ro

3. Universitatea Tehnică din Cluj – Napoca: Ioana - Cornelia Gros, Lucian - Nicolae Pintilie, Teodor Crişan Pană – „SISTEME EMBEDDED ÎN INGINERIE ELECTRICĂ - GHID DE APLICAŢII", 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/

Tema de studiu nr. 6 – Mediul Matlab – Simulink

I. SCOPUL TEMEI

- prezentarea recapitulativă a metodelor de programare a microcontrolerelor [1], [2]
- prezentarea facilităţilor introduse de mediul grafic Matlab – Simulink [1], [2], [3]
- prezentarea instrumentelor „ArduinoIO" prin intermediul aplicaţiilor propuse [3], [4]

II. INTRODUCERE [1], [2], [3]

În vederea soluționării diverselor probleme de inginerie, există o serie de instrumente numerice dedicate sau specializate, precum:
- pachete de programe pentru soluţionarea calculelor (ex. MathCAD, Excel, Matlab etc.);
- pachete de programe pentru modelare şi simulare (ex. DIALux, ePLAN, SolidWorks etc.);
- suite şi medii complexe de modelare, simulare şi cuplare cu aparatura fizică de laborator (ex. Matlab – Simulink, LabVIEW, Altair Embed, Plexim, Scilab etc.);

Toate aceste instrumente menţionate în paragraful anterior, reprezintă soluţii asistate de calculator în diverse faze ale elaborării unui produs sau serviciu:
- **CAD** (eng. Computer Aided Design – proiectarea asistată de calculator);
- **CAE** (eng. Computer Aided Engineering – soluţionarea problemelor de inginerie cu asistenţa calculatorului);
- **CAM** (eng. Computer Aided Manufacturing – producţia sau fabricarea produselor asistată de calculator);

Ţinând cont de procesele care au loc în vederea elaborării unui produs finit, se poate afirma faptul că orice produs de natură electronică cu un grad de complexitate ridicat parcurge cele trei faze în procesul de producţie asistat de calculator (Fig. 6.1):
- CAD – proiectarea schemei, dimensionarea componentelor, proiectarea cablajului;
- CAE – testarea şi simularea modelului proiectat şi elaborarea codului program, achiziţie;
- CAM – realizarea fizică a cablajului, a carcasei şi asamblarea sau lipirea componentelor;

În completarea fazele de proiectare pentru a constitui ciclul complet de fabricaţie, există şi etape intermediare precum:
- verificarea validităţii la nivel principial al schemei electronice (pe baza principiilor, legilor şi teoremelor lui Ohm, Kirchhoff, Maxwell, Thevenin, Norton, Joule-Lenz etc.)
- simulări virtuale, pe baza datelor de catalog ale componentelor şi pe baza măsurătorilor la nivel de laborator în diverse condiţii fizice şi climatologice (ex. testare în condiţii de vibraţie, impact, câmp electromagnetic intens, variaţii de umezeală şi temperatură etc.)
- testarea fizică, reală a produsului finit în condiţiile amintite
- reîntoarcerea produsului sau prototipului în fazele de testare şi proiectare iniţiale

Considerând aceste concepte, se poate afirma faptul că, prin intermediul mediului de testare simulare și programare Matlab – Simulink, pot fi rezolvate atât problemele de proiectare sau dimensionare, cât și problemele de inginerie precum programare, testare automată, generare automată de cod.

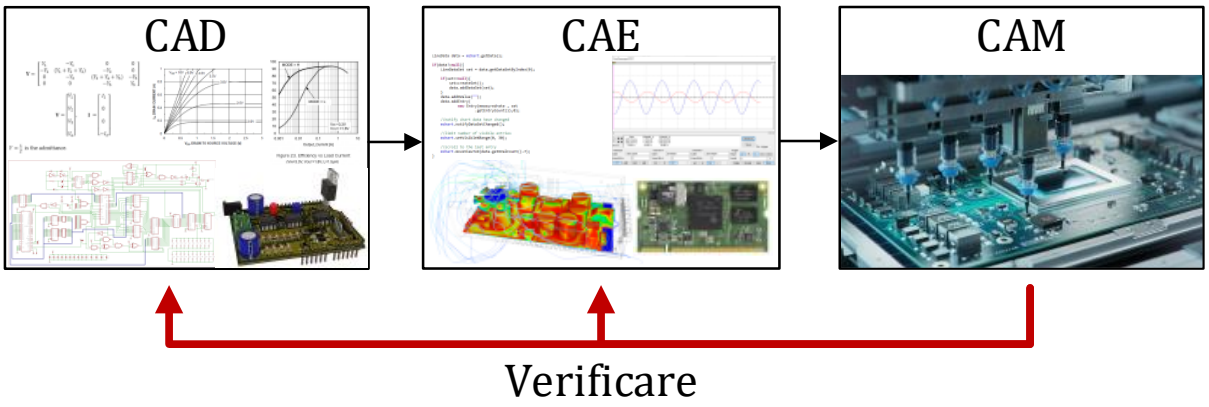


Fig. 6.1 – Parcurgerea etapelor de proiectare a unui produs de natură electronică

Cu ajutorul platformei de dezvoltare Arduino Nano şi a mediului de simulare Matlab – Simulink va fi posibilă realizarea a cel puţin două etape din faza de elaborare a unui produs, anume: CAD – proiectarea produsului şi CAE – modelarea şi simularea circuitului sau codului program care urmează a fi implementat.

Totodată, mediul Matlab – Simulink împreună cu pachetul Arduino IO [3], [4] permit realizarea simulării în timp real a modelului corespondent codului program implementat în Simulink, în strânsă legătură cu prototipul de circuit implementat fizic cu ajutorul perifericelor sistemului de calcul specializat (ex. intrări / ieşiri digitale, intrări analogice, interfeţe de comunicare etc.) (Fig. 6.2).

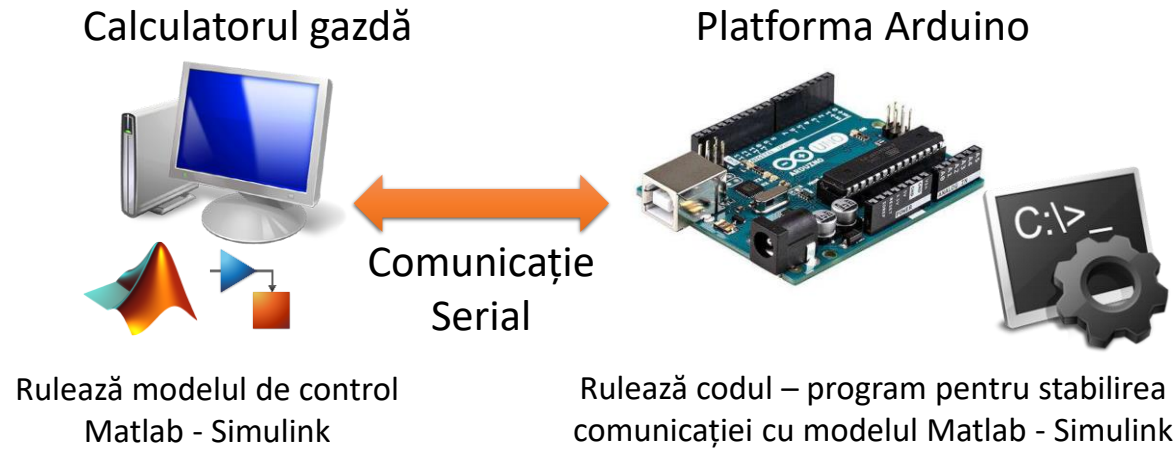


Fig. 6.2 – Schema bloc descriptivă a modului de lucru specific platformei de dezvoltare Arduino ataşată la calculatorul gazdă cu mediul Matlab – Simulink instalat [1], [2]

Procedura de simulare şi analiză a comportamentului în timp real a întregului sistem compus din modelul grafic specific şi circuitul fizic implementat, poartă denumirea în literatura de specialitate „Rapid Control Prototyping” (eng.)– RCP – (ro. dezvoltarea rapidă a algoritmului de control al produsului final). Procedura poate fi întâlnită în etapa de elaborare a strategiilor de comandă şi control pentru convertoare electronice de putere şi presupune: calibrarea senzorilor, acordarea regulatoarelor, impunerea de limite şi protecţii în program, optimizarea programului.

III. ASPECTE TEORETICE [1], [2]

Există patru moduri de elaborarea a codului program pentru un sistem de calcul specializat cu microcontroler precum (Fig. 6.3 sau Fig. 1.7 - Lab. I) [1], [2].
- cod mașină (limbajul specific comenzilor directe pentru procesor);
- limbaj de asamblare (cu instrucțiuni specifice arhitecturii interne a procesorului);
- sintaxă specifică limbajelor de programare (ex. C / C++, Python, Wiring etc.);
- mediu grafic de programare (ex. Matlab – Simulink, LabVIEW, Altair Embed);

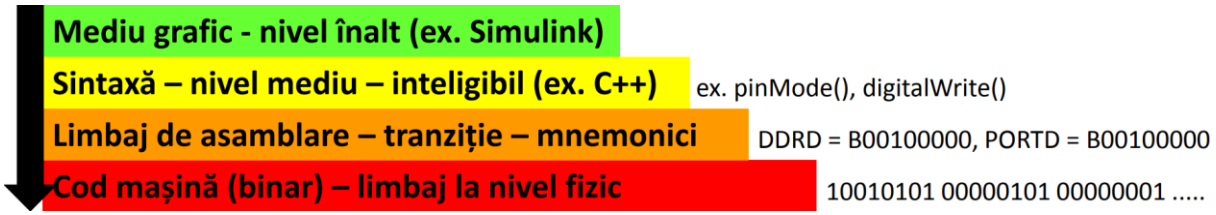


Fig. 6.3 – Metode și limbaje de programare specifice lucrului cu microcontrolere [1], [2]

Mediile grafice de programare sunt utilizate în mod preponderent în domeniul Ingineriei Electrice, atunci când este necesară identificarea unei soluții într-un timp scurt la o problemă complexă precum strategia de comandă și control a unui sistem electronic.

În situația în care este necesară elaborarea codului program destinat unei strategii de control pentru un convertor electronic de putere, se recomandă utilizarea unei abordări sistemice care are la bază structurarea codului program utilizând diagramele sinoptice și simbolurile specifice domeniului reglării automate. Matlab – Simulink, presupune o astfel de abordare, prin introducerea diverselor simboluri și diagrame constituind astfel așa-zisul „model Simulink" al strategiei de comandă și control (Fig. 6.4).

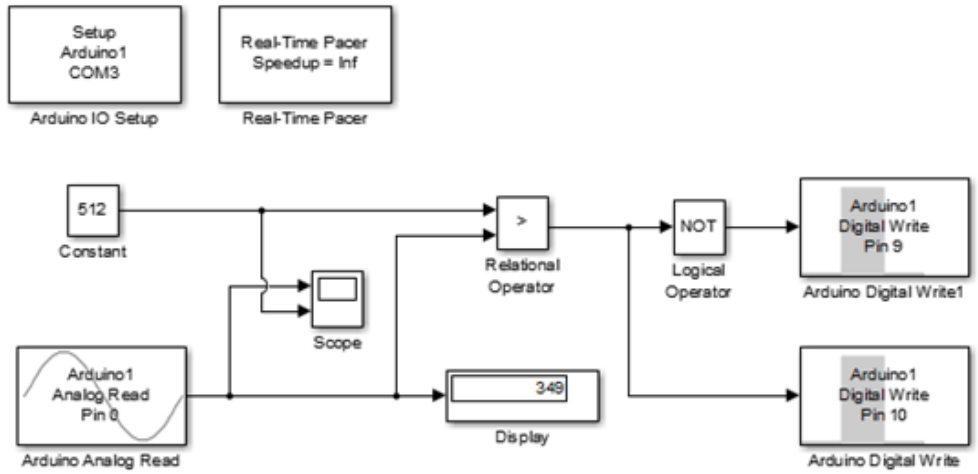


Fig. 6.4 – Reprezentarea modelului Simulink prin diagrame și simboluri [1], [2]

Pe baza diagramei sau modelului se va genera în mod automat codul program destinat procesorului de aplicație sau microcontrolerului. Generatorul automat de cod regăsit în mediul de simulare Matlab – Simulink, are denumirea „Embedded Coder" [4].

În cadrul mediului Matlab – Simulink există două metode de testare în timp real a strategiei de comandă şi control elaborată pe baza modelelor, anume:
- metoda de programare bazată pe pachetul de instrumente „Arduino IO” [3];
- metoda de programare bazată pe „Simulink support package for Arduino” [4];

Cele două metode pot conlucra în mod complementar, anume:
- pe baza pachetului „Arduino IO” se generează modelul de testare în timp real [3];
- pe baza pachetului „Simulink support package for Arduino” se va genera codul final [4];

Prin urmare, se poate afirma faptul că, prin combinarea modului de lucru al celor două pachete de instrumente, se poate parcurge un ciclu complet de dezvoltare a unui produs electronic pe bază de microcontroler (CAD, CAE, CAM) şi anume, furnizarea codului final şi încărcarea acestuia în memoria program (eng. Flash) a microcontrolerului.

În cadrul acestui document se va avea în vedere metoda de simulare şi validare în timp real a modului de funcţionare al codului program generat pe baza modelului Simulink, anume, se va utiliza pachetul de instrumente Arduino IO împreună cu platforma Arduino Nano, şi versiunea Matlab – Simulink R2014a [3]. Astfel, se vor realiza etapele:
- descărcarea şi instalarea mediului de testare, simulare şi programare Matlab – Simulink;
- descărcarea şi instalarea mediului de programare Arduino IDE;
- descărcarea şi dezarhivarea pachetului de instrumente Arduino IO [3];
- încărcarea programului „adio.pde” în memoria permanentă a microcontrolerului [3];
- instalarea paletei de instrumente „Arduino IO” în mediul Matlab – Simulink [3];
- implementarea modelului dorit şi validarea funcţionării acestuia;

Pentru a instala pachetul „Arduino IO”, este necesară obţinerea acestuia de la adresa: https://ctms.engin.umich.edu/CTMS/Content/Activities/ArduinoIO.zip. (Acelaşi fişier se va regăsi şi în arhiva cu documentaţia specifică şedinţei de laborator) [1], [3]. În urma descărcării pachetului de instrumente, se va de arhiva în locaţia „C:\ArduinoIO”. Se va lansa în execuţie mediul Matlab R2014a, şi se va fixa calea de navigaţie „C:\ArduinoIO”. În consola de comanda Matlab, se va introduce comanda `install_arduino` (Fig. 6.5).

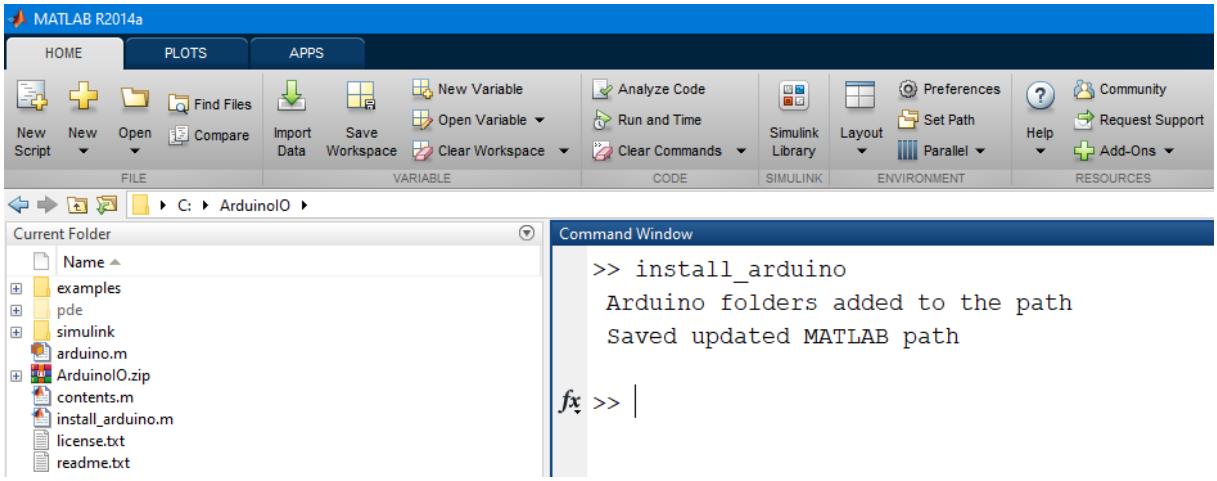


Fig. 6.5 – Procedura de instalare a pachetului de instrumente Arduino IO în Matlab

În cazul în care procedura de instalare decurge corect, în consolă se va obține mesajul „`Arduino folders added to the path -> Saved updated MATLAB path`”.

Pentru a pregăti canalul de comunicație dintre platforma de dezvoltare Arduino Nano și calculatorul gazdă prin intermediul interfeței Serial, va fi necesară încărcarea codului program „adio.pde” în memoria permanentă (Flash) a microcontrolerului (Fig. 6.6). Fișierul se regăsește în directorul în care a fost de arhivat pachetul Arduino IO, anume: „C:\ArduinoIO\pde\adio\adio.pde”. Extensia „.pde” reprezintă echivalentul extensiei „.ino” compatibilă cu mediul Arduino IDE.

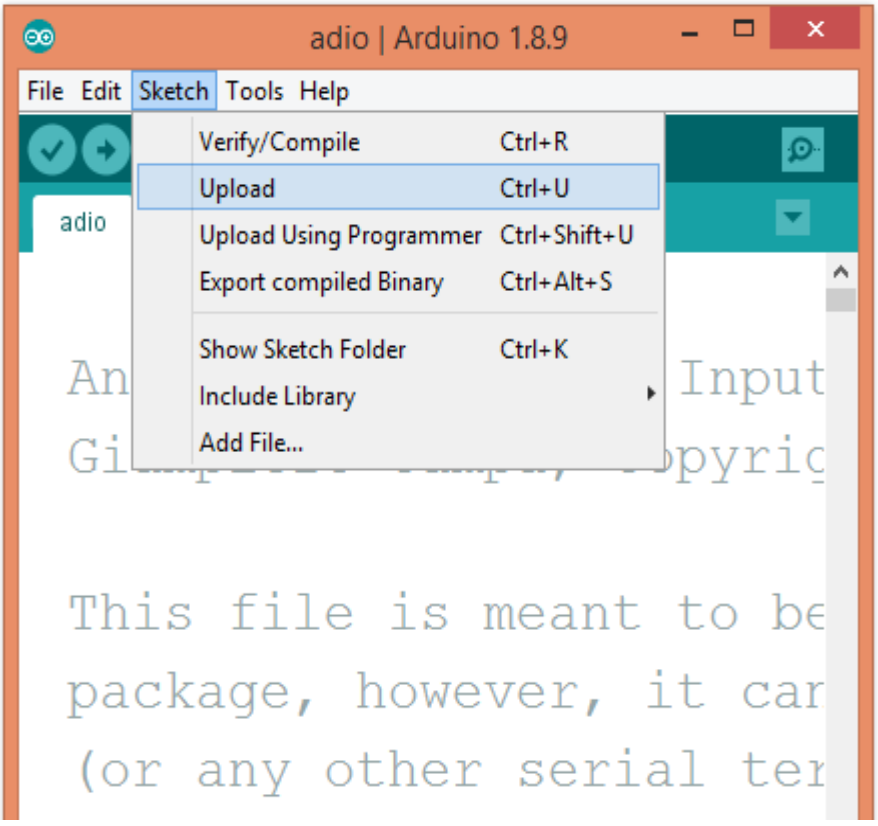


Fig. 6.6. – Încărcarea codului program pentru stabilirea comunicației Serial între platforma Arduino Nano și calculatorul gazdă [1]

Tot în cadrul acestei etape, se va determina portul de comunicație serial, fie din meniul „Tools” -> „Port” al mediului Arduino IDE, fie din utilitarul „Device Manager”, categoria „Ports (COM & LPT)”, dispozitivul cu denumirea „USB Serial Port (COMx)”. În cazul de față, portul de comunicație serial, pe care este atașată platforma de dezvoltare Arduino Nano este „COM3”.

Cunoscând datele de identificare ale portului serial, având încărcat în memorie programul de comunicare serial cu platforma de dezvoltare și având pachetul de instrumente instalat în mediul Matlab – Simulink, se vor implementa o serie de aplicații.

IV. IMPLEMENTAREA APLICAȚIILOR

Pentru a evidenția conceptele prezentate anterior, se propune implementarea următoarelor aplicații cu ajutorul platformei Arduino Nano și mediului Matlab - Simulink:
- semnalizare alternativă intermitentă cu două diode electroluminiscente (eng. LED);
- achiziționarea unui semnal digital;
- măsurarea temperaturii și declanșarea unei ieșiri digitale la depășirea pragului impus;

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 electroluminiscente;
- rezistențe cu valoarea de 100 [Ω];
- rezistențe cu valoarea de 10 [kΩ];
- traductor de temperatura LM-35;
- fire pentru conexiune rapidă compatibile cu placa de testare;
- calculator gazdă având mediul Arduino IDE și Matlab Simulink + Arduino IO instalate;
- cablu adaptor USB A la mini USB;

La fiecare model Simulink nou, se va stabili timpul total de simulare (eng. Stop time) „inf" (infinit) și timpul de eșantionare (eng. Fixed-step (fundamental sample time)) „1e-3" (1 * 10^{-3} [s]). Pentru a impune timpul de eșantionare este necesară alegerea opțiunii „Fixed-step" în categoria „Solver options → Type" și opțiunea „Solver → discrete (no continuous states)", (Fig. 6.7).

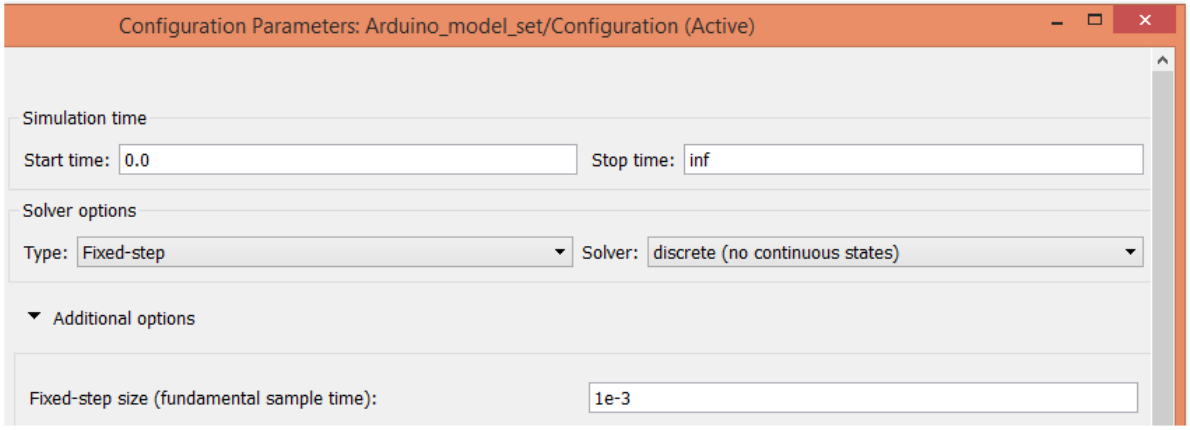


Fig. 6.7 – Stabilirea timpului total de simulare [1]

De asemenea, tot în cadrul modelelor Simulink noi create se vor introduce (de fiecare dată) următoarele blocurile elementare:
- Arduino IO Setup - pentru inițializarea conexiunii dintre model și platforma Arduino;
- Real – Time Pacer - pentru stabilirea parametrilor de temporizare ai simulării;

În cadrul blocului „Arduino IO Setup" se va impune portul de comunicație serial „COMx" (Fig. 6.8). Prin intermediul acestui bloc, va fi posibilă declararea unei noi platforme de dezvoltare Arduino (prin intermediul opțiunii „Arduino variable"), în paralel cu cea stabilită anterior, doar că, va fi necesară declararea unui nou port de comunicație serial.

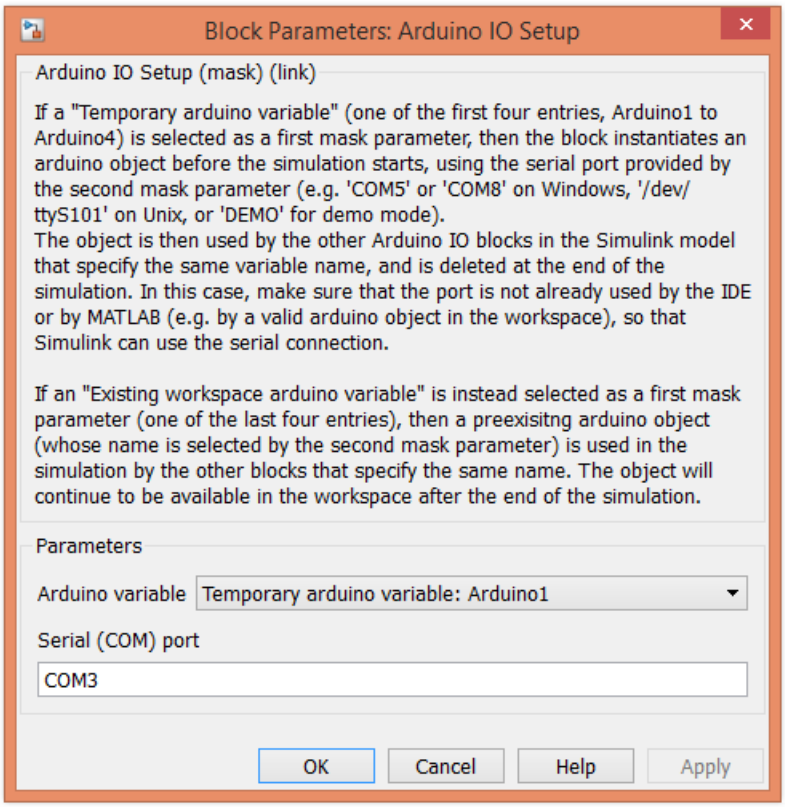


Fig. 6.8 – Stabilirea portului de comunicaţie serial „COMx” prin intermediul blocului „Arduino IO Setup” [1]

Tot în cadrul unui model nou, prin intermediul blocului elementar „Real - Time Pacer” se va stabili pasul de parcurgere în timp a modelului „inf” (infinit), (Fig. 6.9).

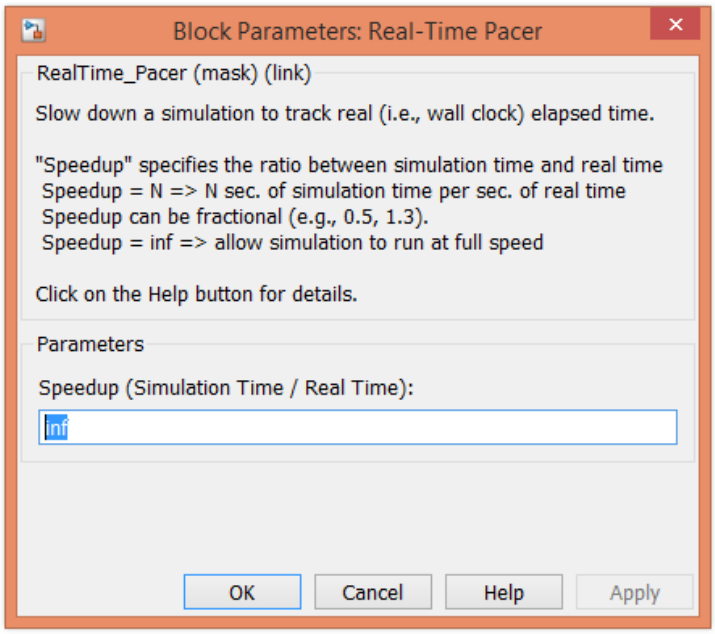


Fig. 6.9 – Stabilirea pasului de parcurgere prin intermediul blocului „Real – Time Pacer”

APLICAȚIA 1

Se va implementa circuitul conform următoarei scheme (Fig. 6.10):

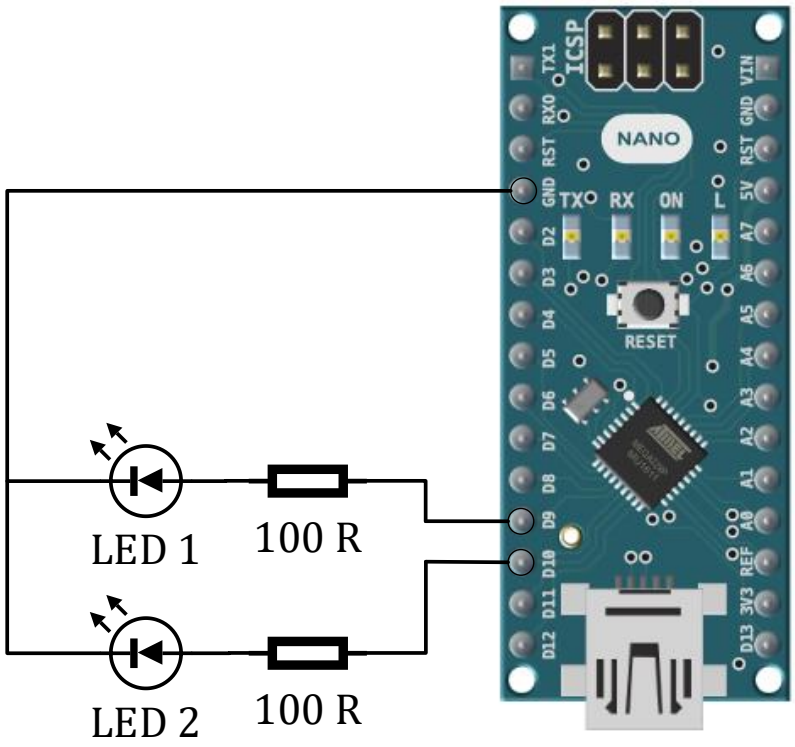


Fig. 6.10 – Schema electronică pentru implementarea circuitului specific aplicației 1

Montajul experimental se va realiza conform (Fig. 6.11):

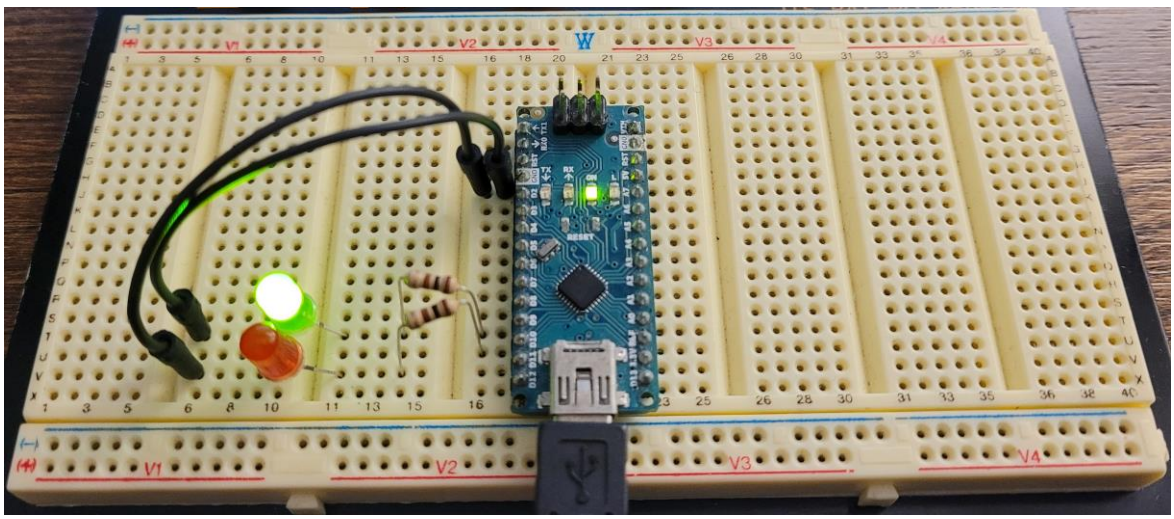


Fig. 6.11 – Montaj experimental pentru aplicația 1

Se va implementa următorul cod program pe baza modelului Simulink (Fig. 6.12):

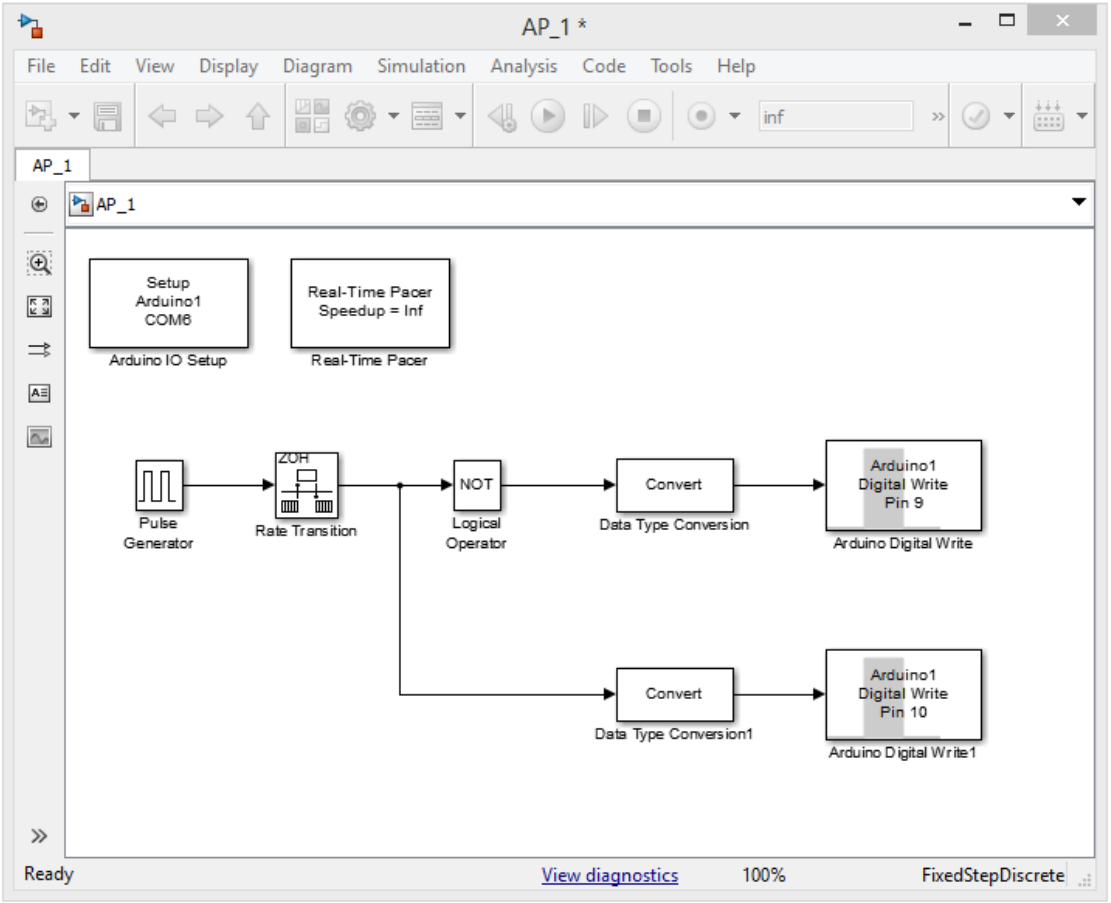


Fig. 6.12 – Codul program specific aplicației 1 implementat sub formă de model Simulink

În vederea implementării modelului Simulink sunt necesare blocurile (Fig. 6.13):

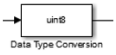
Simbol bloc	Denumire bloc	Categoria din care face parte
	Arduino IO Setup	Arduino IO Library
	Real – Time Pacer	Arduino IO Library
	Pulse Generator	Simulink → Sources
	Rate Transition	Simulink → Signal Attributes
	Logical Operator	Simulink → Logic and Bit Operations
	Arduino Digital Write	Arduino IO Library
	Data Type Conversion	Simulink → Signal Attributes

Fig. 6.13 – Lista blocurilor necesare în vederea implementării modelului Simulink

Implementarea aplicației nr. 1 presupune:

- comutarea alternativă a stării logice a ieșirilor digitale a celor două terminale „D9” și „D10” prin intermediul generatorului de semnal dreptunghiular la frecvența de 1 [Hz];
- blocul „Rate Transition” adaptează timpul de eșantionare al generatorului de semnal la timpul de eșantionare al blocurilor pentru ieșire digitală;
- blocul „Logical Operator” realizează funcția de inversare logică;
- blocul „Data type conversion” asigură trecerea de la un tip de date stabilit inițial la un alt tip de date necesar la ieșirea din model (ex. stare logică „ACTIV” (eng. HIGH) sau „INACTIV” (eng. LOW) transformat în număr întreg reprezentat pe opt biți „0” sau „1”);
- blocul „Arduino Digital Write” asigură modificarea stării logice a terminalului digital specificat în parametrii de configurare (ex. D9).

OBSERVAȚIE: Timpul de eșantionare specific intrărilor și ieșirilor fizice (digitale sau analogice) este 0,01 [s]!

APLICAȚIA 2

Se va implementa circuitul conform următoarei scheme (Fig. 6.14):

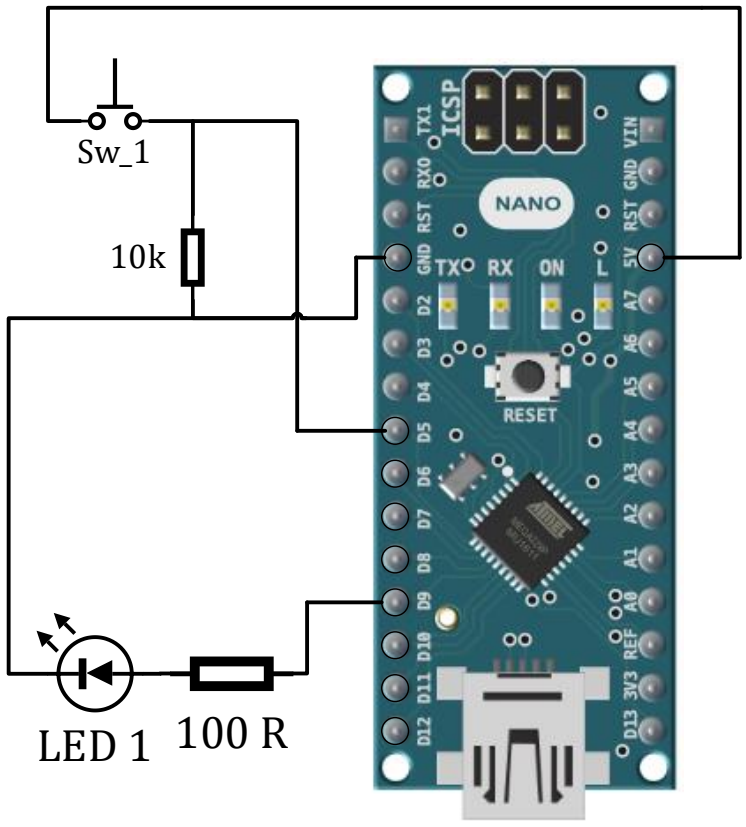


Fig. 6.14 – Schema electronică pentru implementarea circuitului specific aplicației 2

Montajul experimental se va realiza conform (Fig. 6.15):

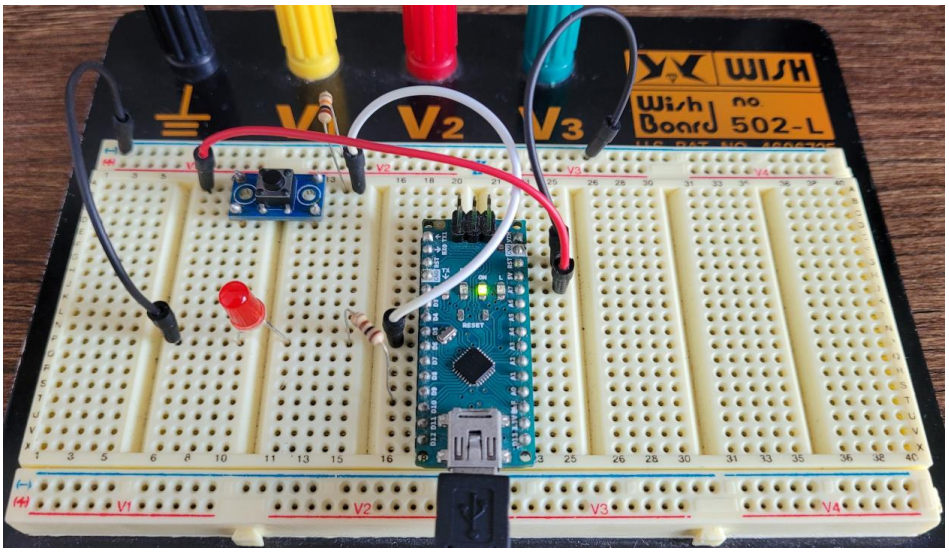


Fig. 6.15 – Montaj experimental pentru aplicația 2

Se va implementa următorul cod program pe baza modelului Simulink (Fig. 6.16):

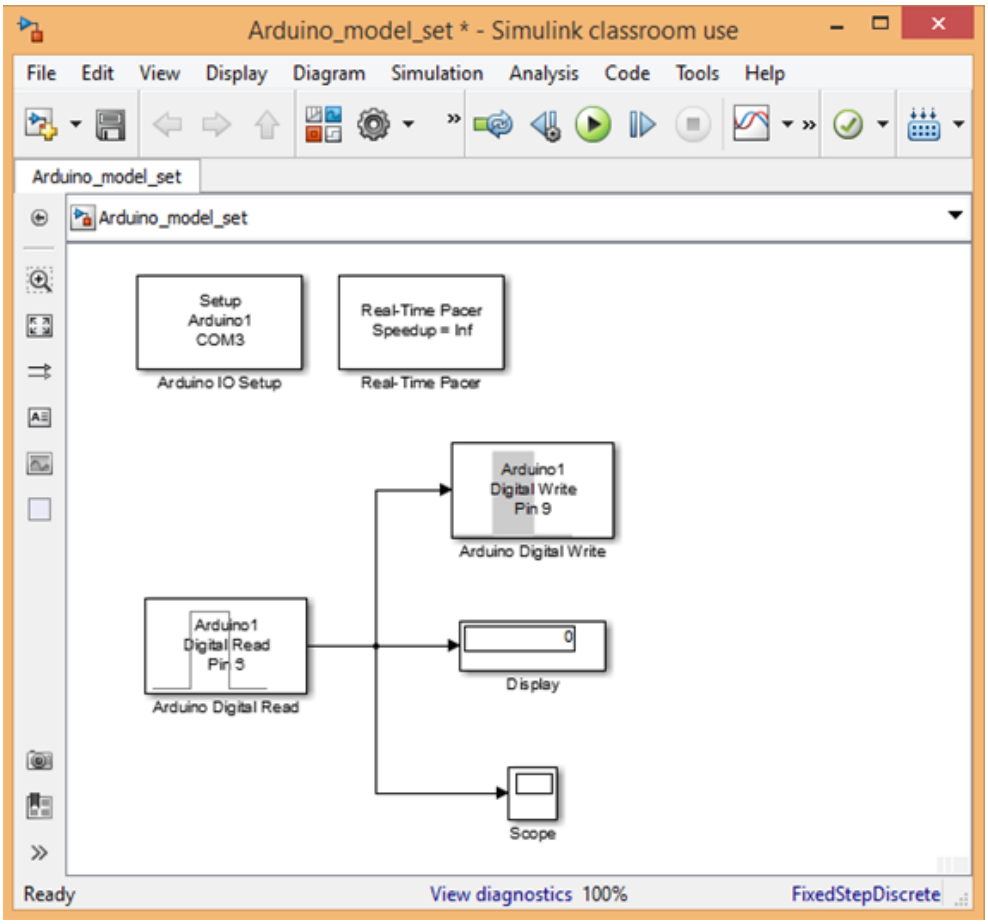


Fig. 6.16 - Codul program specific aplicației 2 implementat sub formă de model Simulink

În vederea implementării modelului Simulink sunt necesare blocurile (Fig. 6.17):

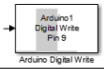
Simbol bloc	Denumire bloc	Categoria din care face parte
	Arduino IO Setup	Arduino IO Library
	Real – Time Pacer	Arduino IO Library
	Arduino Digital Read	Arduino IO Library
	Arduino Digital Write	Arduino IO Library
	Display	Simulink → Sinks
	Scope	Simulink → Sinks

Fig. 6.17 – Lista blocurilor necesare în vederea implementării modelului Simulink

Implementarea aplicației nr. 2 presupune:

- preluarea stării terminalului ales prin intermediul blocului „Arduino Digital Read”
- 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);
- semnalizarea prin intermediul unei diode electro-luminiscente atașată la o ieșire digitală la care se va modifica starea logică în funcție de starea intrării digitale asociată prin intermediul modelului Simulink;
- monitorizarea semnalelor vehiculate în model pe calea virtuală de semnal (eng. virtual signal path) cu ajutorul blocurilor „Display” și „Scope” (Fig. 6.18).

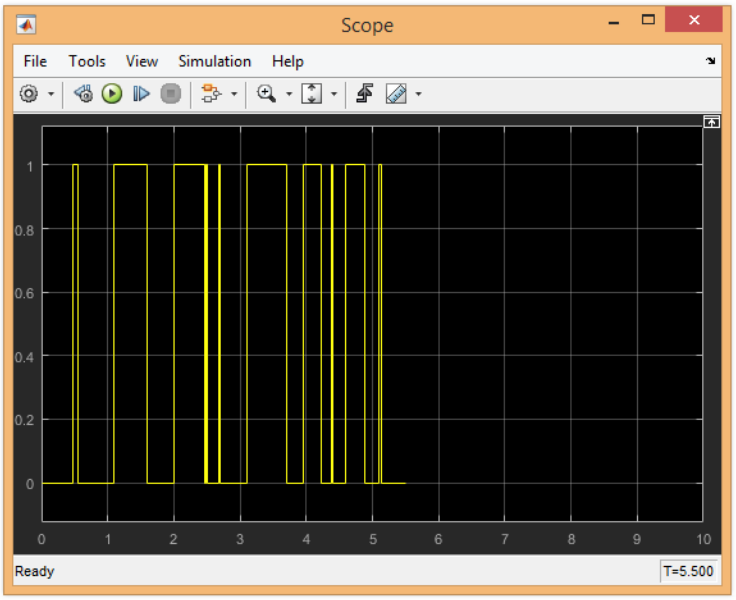


Fig. 6.18 – Monitorizarea semnalului digital vehiculat pe calea virtuală de semnal

OBSERVAȚIE: Cu ajutorul instrumentului „Display” se va realiza monitorizarea valorilor numerice vehiculate pe calea virtuală de semnal. Cu ajutorul blocului „Scope” se va realiza monitorizarea formei de undă a semnalului vehiculat pe calea virtuală de semnal.

APLICAȚIA 3

Se va implementa circuitul conform următoarei scheme (Fig. 6.19):

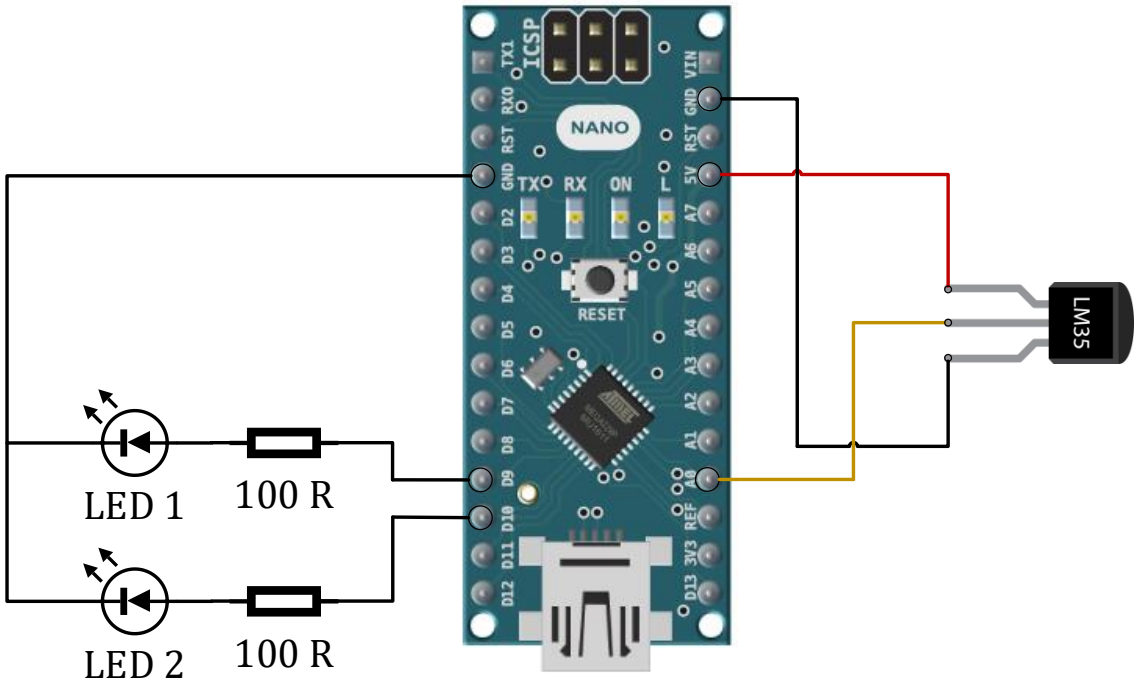


Fig. 6.19 – Schema electronică pentru implementarea circuitului specific aplicației 3

Se va realiza următorul montaj experimental (Fig. 6.20):

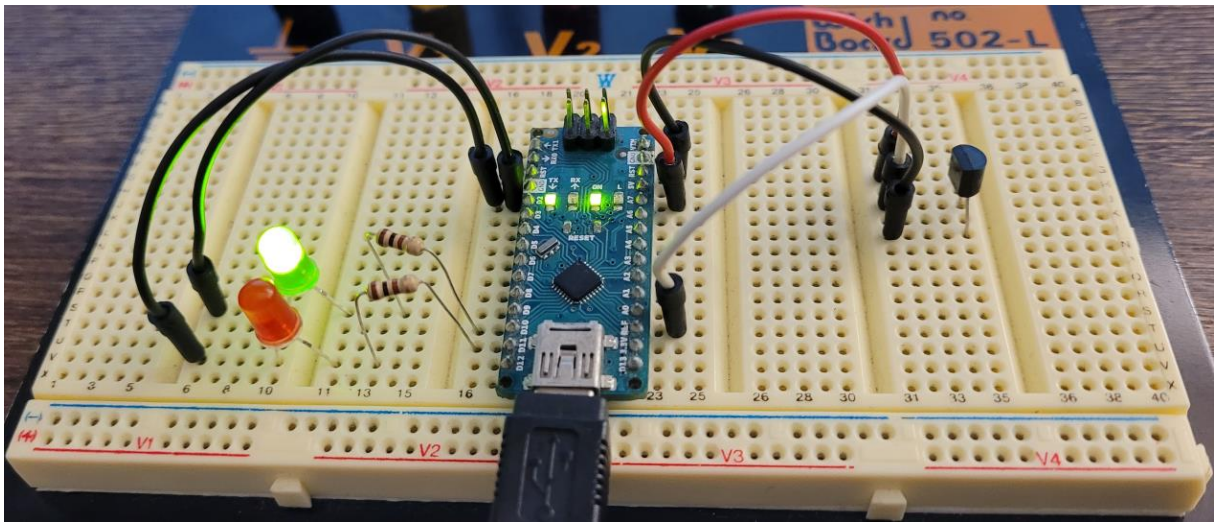


Fig. 6.20 – Montajul experimental specific aplicației 3

În vederea implementării modelului Simulink sunt necesare blocurile (Fig. 6.21):

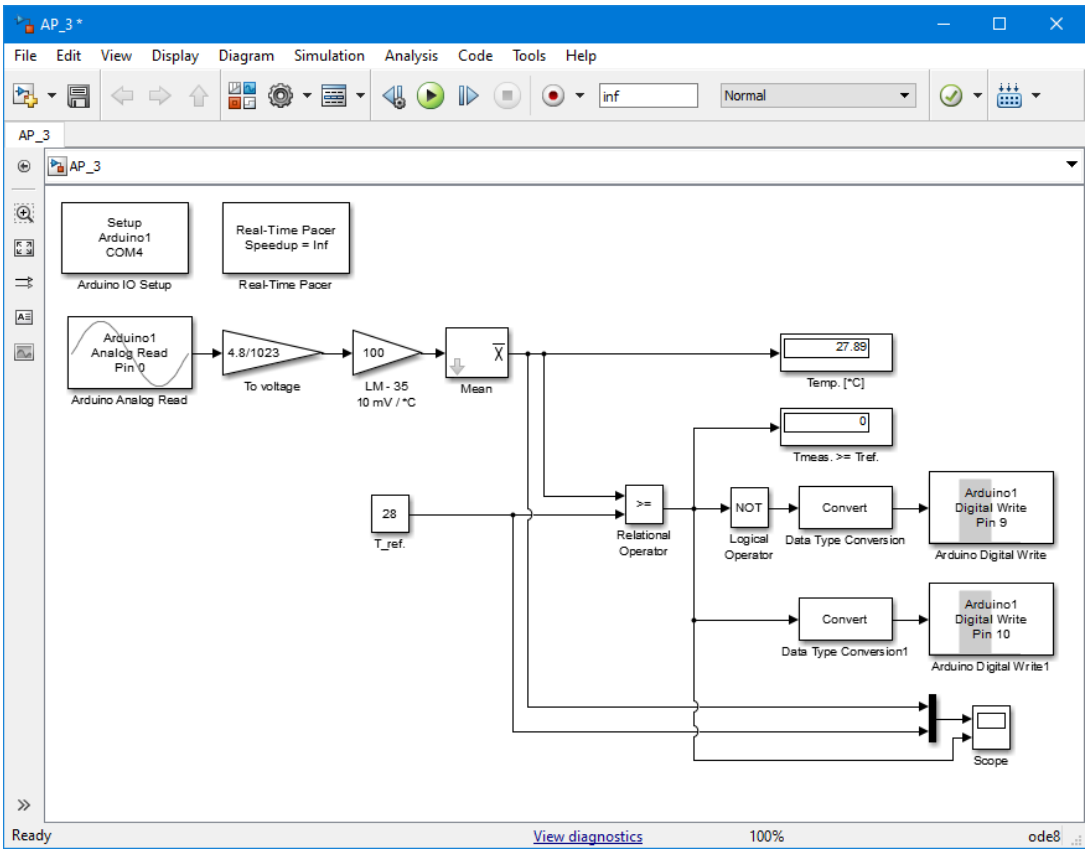


Fig. 6.21 - Codul program specific aplicației 3 implementat sub formă de model Simulink

În vederea implementării modelului Simulink sunt necesare blocurile (Fig. 6.22):

Simbol bloc	Denumire bloc	Categoria din care face parte
	Arduino IO Setup	Arduino IO Library
	Real – Time Pacer	Arduino IO Library
	Arduino Analog Read	Arduino IO Library
	Display	Simulink → Sinks
	Scope	Simulink → Sinks
	Mux	Simulink → Commonly Used Blocks
	Mean	Simscape → SimPowerSystems → Specialized Technology → Control and Measurements Library → Measurements

Fig. 6.22 – Lista blocurilor necesare în vederea implementării modelului Simulink

Implementarea aplicației nr. 3 presupune:

- 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;
- compararea nivelului temperaturii actuale cu valoarea de prag impusă inițial;
- afișarea pe grafic atât a valorii actuale cât și a pragului de temperatură (Fig. 6.23);

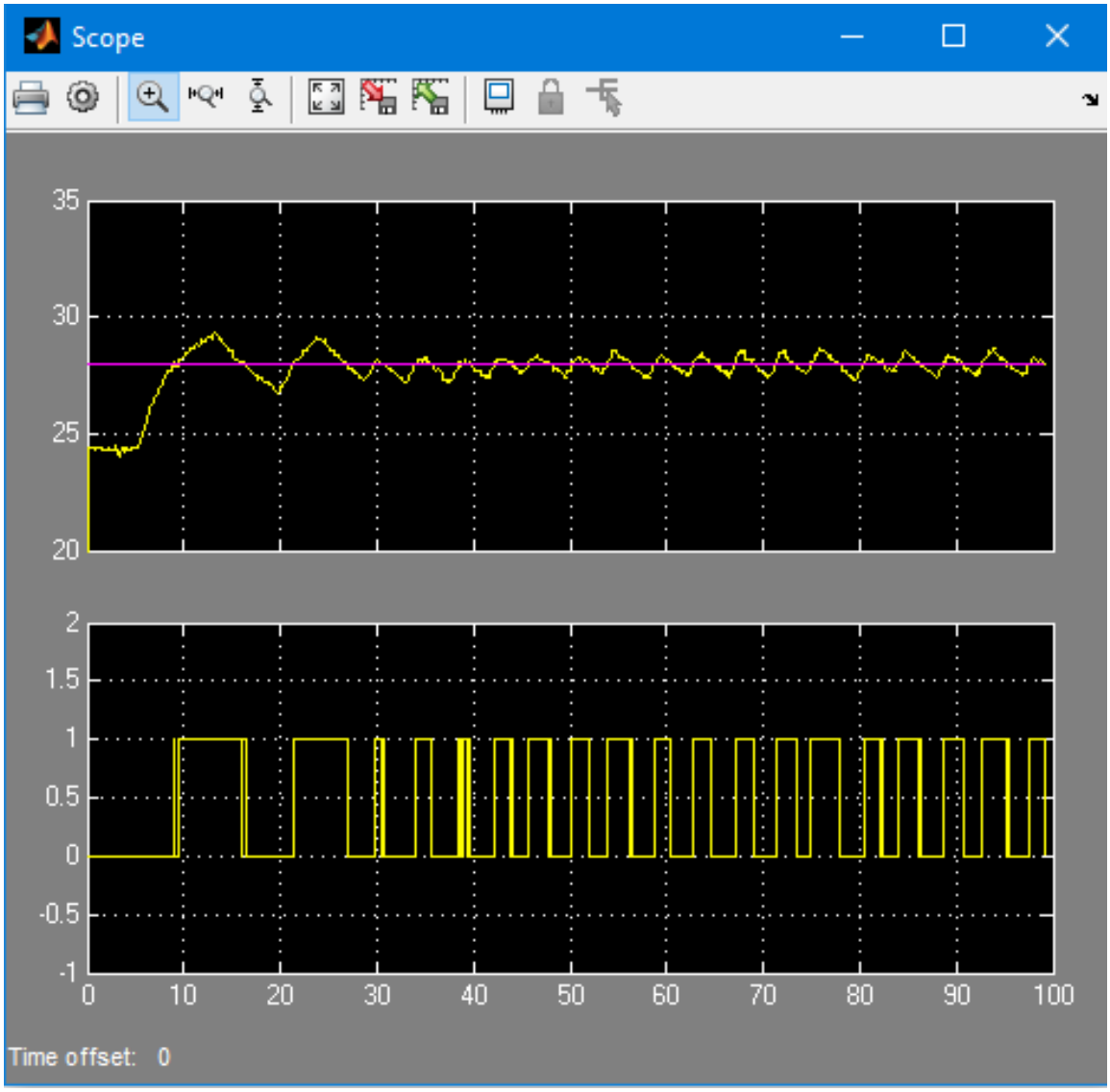


Fig. 6.23 – Monitorizarea semnalelor vehiculate pe calea virtuală de semnal

OBSERVAȚIE: În vederea determinării mediei aritmetice în mod dinamic (prin intermediul blocului „Mean"), este necesară alegerea unei metode de rezolvare a ecuațiilor diferențiale liniare cu coeficienți constanți care fac parte din modelul blocului de mediere. Pentru aceasta, este necesară alegerea opțiunii „ode8 (Dormand-Prince)" în categoria „Solver" (Fig. 6.24)

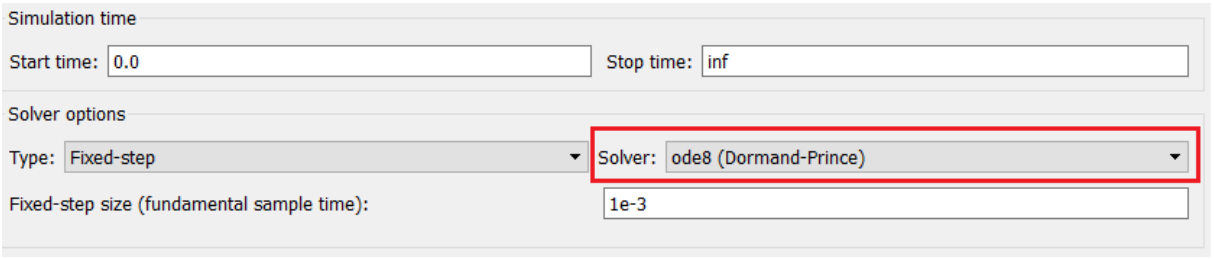


Fig. 6.24 – Modificarea parametrilor pentru metoda de rezolvare a ecuaţiilor diferenţiale

În vederea filtrării zgomotelor suprapuse peste semnalul de măsura preluat de la traductorul de temperatura LM - 35, se vor efectua următoarele parametrizări la nivelul blocului de mediere (Fig. 6.25):
- frecvenţa componentei fundamentale: 5 [Hz];
- stare iniţială: 0;
- timp de eşantionare: 0,01 [s];

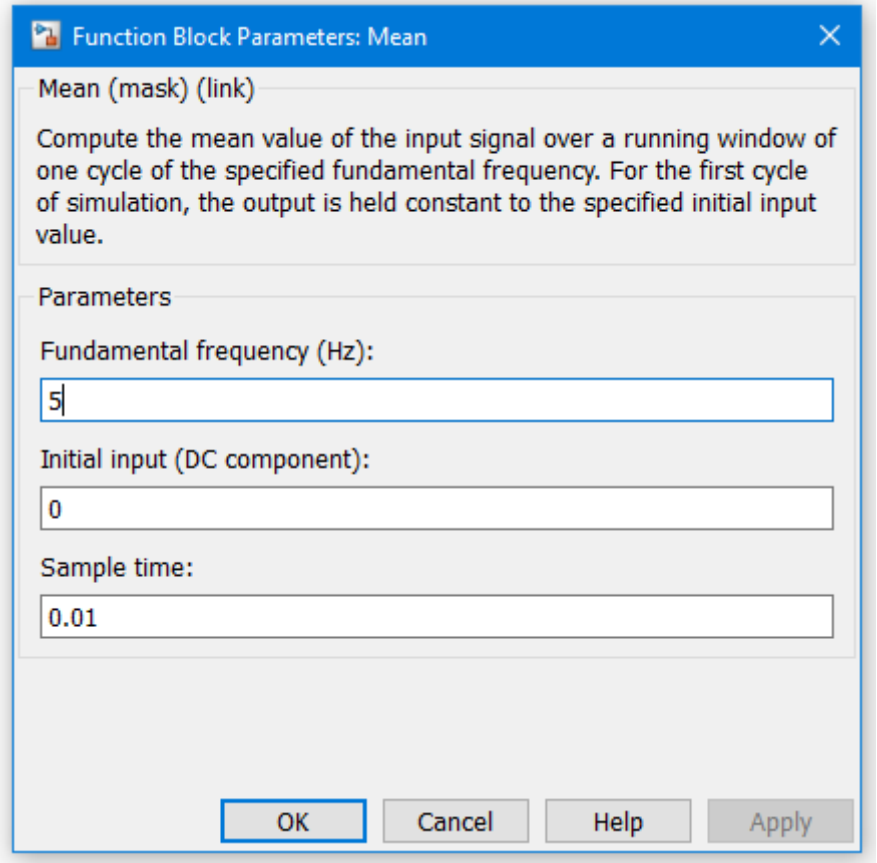


Fig. 6.25 – Parametrizarea blocului de calcul dinamic al mediei aritmetice a semnalului

V. CONCLUZII

- Utilizând mediul Matlab – Simulink în strânsă legătură cu platforma Arduino, pot fi dezvoltate aplicații complexe de achiziție și control datorită faptului că, în acest mediu de simulare, există posibilitatea unei abordări sistemice a programului realizat;

- Programarea platformei Arduino pe baza blocurilor funcționale reprezintă esența abordării sistemice;

- Simularea în timp real permite interacțiunea utilizatorului cu parametrii procesului direct din model, prin intermediul calculatorului gazdă;

- Procedeul de analiză și depanare în timp real a unei bucle de control poartă denumirea de Rapid Control Prototyping. Utilizarea mediul Matlab – Simulink și platforma Arduino, împreună cu pachetul „Arduino IO", reprezintă un procedeu similar.

VI. BIBLIOGRAFIE

1. Universitatea Tehnică din Cluj – Napoca – Electronics and Power Electronics – „SISTEME CU MICROPROCESOARE" – „Documentație pentru laboratorul de Sisteme cu Microprocesoare": https://epe.utcluj.ro/index.php/sisteme-cu-microprocesoare/

2. 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:

3. University of Michigan – Control Tutorials for Matlab & Simulink – „Simulink ArduinoIO Package": https://ctms.engin.umich.edu/

4. The MathWorks Inc. – „Legacy MATLAB and Simulink Support for Arduino": https://www.mathworks.com/