Gheorghe SEBESTYEN **Anca HANGAN**

INFORMATICĂ INDUSTRIALĂ

Lucrări practice, exerciții și probleme

UTPRESS

CLUJ-NAPOCA, 2019

ISBN 978-606-737-414-8

Gheorghe SEBESTYEN **Anca HANGAN**

INFORMATICĂ INDUSTRIALĂ

Lucrări practice, exerciții și probleme

Editura UTPRESS
Cluj-Napoca, 2019
ISBN 978-606-737-414-8

Editura U.T.PRESS
Str. Observatorului nr. 34
C.P. 42, O.P. 2, 400775 Cluj-Napoca
Tel.:0264-401.999
e-mail: utpress@biblio.utcluj.ro
http://biblioteca.utcluj.ro/editura

Director: Ing. Călin D. Câmpean

Recenzia: Conf.dr.ing. Lucia Văcariu

ISBN 978-606-737-414-8

Prefață

Informatica industrială este un domeniu aplicativ, care studiază modalitățile de utilizare a tehnicii de calcul și a tehnologiilor informatice în scopul monitorizării și automatizării unor procese fizico-chimice, cu preponderență din domeniul industrial. Importanţa acestui domeniu s-a amplificat mai ales în ultimul deceniu, odată cu apariția unor concepte și tendințe noi cum ar fi Internetul Obiectelor (eng. Internet of Things), noua revoluție industrială, (cunoscută sub denumirea de Industry 4.0), robotizarea proceselor industriale și utilizarea pe scară largă a rețelelor senzoriale. În același timp importanţa domeniului este amplificată de tendința mai generală de globalizare a economiei mondiale, care presupune distribuirea proceselor economice și industriale pe mai multe continente.

Implementarea unor aplicații de monitorizare și control al proceselor prin tehnologii informatice și de comunicație implică metode specifice de modelare, proiectare, realizare și testare a sistemelor de calcul și a aplicațiilor software destinate acestui scop; aceste metode trebuie să ţină cont de o serie de cerințe și restricții specifice domeniului industrial, cum ar fi: fiabilitate ridicată, toleranță la defecte, comportament de timp-real, consum energetic redus, dimensiuni și forme adaptate procesului controlat și nu în ultimul rând costuri minime.

Lucrarea de față conține o serie de lucrări practice din domeniul Informaticii industriale, prin care autorii încearcă să familiarizeze cititorul cu aceste tehnici noi de realizare a sistemelor de calcul, dedicate pentru scopuri de monitorizare și control. Printr-o serie de exemple concrete sunt demonstrate posibilitățile de utilizare ale unor componente și tehnologii digitale în scopul automatizării proceselor fizico-chimice. De asemenea, se demonstrează modalitatea de utilizare a unor tehnici de comunicație în scopul realizării de sisteme de control distribuit.

În cadrul acestor lucrări se prezintă o serie de echipamente tipice (ex. microcontroloare PIC, PLC-uri, rețele senzoriale, etc.), utilizate în urmărirea și controlul proceselor, precum și platformele de proiectare și dezvoltare dedicate acestor echipamente (plăci de dezvoltare, platforme multifuncționale, medii de programare, depanatoare, etc.).

Această lucrare este destinată cu precădere pentru studenții specializărilor de calculatoare și tehnologia informației, dar poate fi utilizată cu succes și de cei care au o pregătire informatică de bază și doresc să realizeze proiecte din domeniul automatizărilor industriale.

Cuprins

1 Modelarea proceselor fizice și controlul automat al acestora

1.1 Obiectivul lucrării

Lucrarea urmărește identificarea și înțelegerea conceptelor de bază din domeniul Informaticii industriale precum și a problematicii domeniului. În acest scop se vor analiza câteva procese fizice reale din perspectiva posibilităților de automatizare a acestora. Se vor identifica mărimile fizice care descriu procesul studiat și relațiile care definesc comportamentul acestora.

1.2 Considerații teoretice

Un concept central în studiul informaticii industriale este cel de "**proces**". Se consideră un proces orice transformare suferită de un sistem, care este vizibilă prin modificări ale unor mărimi fizice. De exemplu deplasarea mecanică a unor obiecte, transformările fizico-chimice ale unor fluide, modificările de formă ale unor componente, variațiile unor mărimi fizice precum temperatură, nivel, concentrație, etc. sunt exemple de procese.

Din punct de vedere al sistemelor de automatizare ne interesează în speță procesele tehnice și în particular procesele industriale. Prin atributul "industrial" se face referire la acele procese care au scopul de a produce în mod controlat diverse produse sau servicii. De exemplu producerea de energie electrică, de ciment sau de substanțe chimice diverse intră în categoria de procese industriale. Dar și sistemele în care rezultă un anumit serviciu, cum ar fi controlul parametrilor unei clădiri, distribuția energiei electrice sau sistemele de securitate intră în categoria de **procese "industriale**". Într-o accepțiune mai largă face obiectul studiului nostru orice proces în care sunt implicate mărimi fizico-chimice măsurabile. Astfel procesele economice, politice, sociale, psihologice sau de altă natură în care nu avem de a face cu mărimi fizico-chimice nu sunt obiectul acestui studiu, deși tehnici de reglaj automat sunt aplicabile și în aceste domenii.

Parametrii unui proces sunt acele mărimi ce caracterizează evoluția și starea unui proces. În funcție de rolul acestora distingem parametrii de intrare, de ieșire sau de stare ai procesului. Informația cu privire la parametrii de proces se transmite prin intermediul unor **semnale.** Un semnal este o mărime fizică ce este capabilă să transmită o informație. De exemplu nivele de tensiune sau de curent, impulsuri electrice sau

optice, unde radio, etc. sunt diverse tipuri de semnale. În funcție de natura acestora distingem semnale analogice sau continue și semnale digitale sau discrete. Caracterul continuu sau discret se poate referi atât la evoluția în timp a semnalelor (semnale continue sau eșantionate) cât și la modul de variație a acestora (semnale continue sau digitizate).

Indiferent de natura fizică a proceselor analizate (chimice, mecanice, electrice, etc.), **modelarea** acestora se poate realiza printr-un set de ecuații matematice ce derivă din diverse legi fizice, chimice sau de altă natură, ce guvernează procesele respective. În majoritatea cazurilor reale, procesele sunt foarte complexe, implicând un număr mare de mărimi fizice care interacționează în diverse moduri.

În mod inerent prin modelarea procesului se omit o serie de detalii considerate mai puțin importante pentru descrierea și mai ales pentru evoluția procesului. Supozițiile și restricțiile simplificatoare sunt necesare pentru reducerea complexității sistemului și mai ales pentru a putea aplica o anumită tehnică de control automat. De exemplu în mod frecvent se omit condițiile de mediu care influențează un anumit proces (temperatură, umiditate, vibrații, etc.), datorită dificultății de cuantizare a acestora. Influenta cumulativă a factorilor de mediu se modelează prin conceptul de "**zgomot**". Zgomotul sau **perturbația** este un semnal a cărei evoluție este necunoscută sau imposibil de descris prin formule matematice exacte.

Un **sistem de control** are rolul de a influența un anumit proces astfel încât să se mențină unul sau mai mulți parametrii de proces la valori prescrise sau să se imprime sistemului o anumită evoluție prestabilită. De exemplu menținerea temperaturii unui cuptor la o valoare stabilită sau deplasarea unui obiect cu ajutorul unui braț robotic pe un anumit traseu prestabilit reprezintă posibile obiective ale unui sistem de control. Dacă controlul se realizează prin tehnologii digitale, în particular prin folosirea unui microsistem de calcul, atunci vorbim despre un **sistem digital de control**.

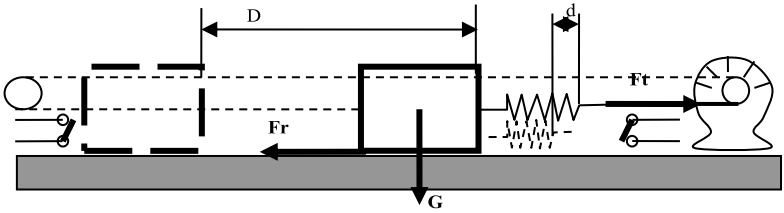
1.2.1 Modelarea proceselor

Așa cum s-a arătat în paragraful precedent, prin modelarea unui proces se urmărește exprimarea prin ecuații matematice exacte a comportamentului unui sistem. Modelarea este necesară în cazul în care dorim să influențăm într-un anumit fel procesul ce urmează a fi controlat. Astfel vom ști care parametru de proces trebuie modificat pentru a obține efectul dorit. Așa cum se va vedea în continuare pe exemplele următoare, procesul de modelare este unul relativ complex și adesea imprecis datorită elementelor care se omit.

Modelarea unui proces se poate face pe cale analitică sau experimentală (empirică). În primul caz se consideră cunoscute legile fizico-chimice care guvernează procesul respectiv. În al doilea caz procesul este mult prea complex pentru a fi descris prin ecuații matematice exacte; în acest caz se presupune că sistemul are un comportament de sistem liniar de ordinul 1 sau doi (ordinul fiind dat de numărul de constante de timp dominante) și prin măsurători experimentale se determină coeficienții (constantele) ecuației diferențiale ce descrie comportamentul sistemului. Această ultimă tehnică poartă numele de "identificarea procesului". În literatura de specialitate sunt date diverse tehnici de identificare experimentală a proceselor.

1.2.2 Exemple de procese fizice și modelarea analitică a acestora

Exemplul 1 Deplasarea pe orizontală a unui obiect.



Mărimi fizice identificate în cadrul procesului:

Ft – forța de tracțiune
Fr – forța rezistentă (de frecare)
Fi – forța de inerție
Fe – forța de elongație a arcului
D – deplasarea obiectului
d – elongația arcului
P – puterea motorului
U – tensiunea electrică
I – curentul electric
N – turația motorului
L – lucrul mecanic produs de motor
R – raza axului motorului
a – accelerația
v_0 – viteza inițială

Cazul I – fără element elastic (fără arc)

$P = U*I*\eta$ - puterea transmisă de motor
$L = P*t$ - lucrul mecanic produs de motor
$L = Ft*D$ - lucrul mecanic consumat prin frecare
$Ft = Fr + m*a$ - ecuația forțelor
$Fr = \mu*m*g$ - forța de frecare
$D = (a*t^2)/2$ - deplasarea
$v = dD/dt$ - viteza
$a = dv/dt$ - accelerația

$Ft = L/D = (P*t)/D = (U*I*\eta*t)/D$
$Ft = Fr+m*a = \mu*m*g + m*2*D/t^2$

$(U*I*\eta*t)/D = \mu*m*g + m*2*D/t^2$

$D = f(U, t)$

Factori perturbatori:
U – variabil
μ – variabil
m – variabil
limita de turație n

Controlul deplasării în buclă deschisă:

Ţinând cont de formula determinata mai sus, D= f(U, t), se aplică o tensiune U pe motorul electric pe o anumita perioada de timp t și se „speră" că obiectul se va deplasa în punctul dorit. In realitate precizia de estimare a deplasări este foarte grosieră, depinzând de mulţi factori care nu au fost prinși în formula de mai sus (ex. variaţia coeficientului de frecare).

Acest lucru se observă de exemplu la un robot cu două motoare de actionare de curent continuu: deși cele două motoare primesc aceeași tensiune pe aceeași perioadă de timp, robotul nu se deplasează în linie dreaptă, iar distanţa parcursă nu este repetabilă în timp.

Controlul în buclă deschisă se poate face cu un motor pas-cu-pas, la care numărul de paşi efectuaţi de rotorul motorului este în acord cu numărul de impulsuri date către înfăşurările motorului.

Varianta1: cu motor pas cu pas:

```
Deplasare(Dir, Nr_pasi) // returnează true dacă deplasarea s-a efectuat
{       Outport(Adr_dir_motor, Dir);
        For(int i=0; i<Nr_pasi; i++)
        {Outport(Adr_motor, HIGH)
```

```
        Delay(100);
        Outport(Adr_motor, LOW)
        Delay(100);
        }
}
```

Controlul deplasării în buclă închisă:

In buclă închisă comanda este generată în funcţie de reacţia sistemului (eng. feed-back), detectată printr-un senzor de deplasare.
Sunt necesare urmatoarele elemente:

- senzor de deplasare sau de rotație a axului (turometru), senzor de deplasare absolută sau relativă
- semnal digital pentru pornirea/optirea motorului (Start_motor),
- punte pentru acționarea motorului, comandă tip PWM pentru controlul puterii transmise
- 2 semnale pentru comanda punții (motor de C.C.) – DirDr, DirSt sau
- 2 senzori (ex. optici pentru limite de cursă și pentru stabilire referința zero): LimDr, LimSt

Inițializare: aducerea obiectului într-o stare inițială
Algoritm:

```
timeout= time_max;
while (LimDr = 1 AND Timeout != 0)
{       DirDr= 1;
        DirSt=0;
        Timeout - -;
}
```

Deplasare într-o anumită direcție cu un număr de pași
Varianta 2: cu motor de curent continuu şi senzor de deplasare

```
Deplasare(Dir, Nr_pasi)
{       Outport(Adr_dir_motor, Dir);
        Outport(Adr_start_motor, HIGH);         // pornire motor
        For(int i=0; i<Nr_pasi; i++)
        {  senzor_pas=Inport(Adr_senzor_pas);
           While(senzor_pas!=HIGH)
                senzor_pas=Inport(Adr_senzor_pas);
           While(senzor_pas!=LOW)
                senzor_pas=Inport(Adr_senzor_pas);
        }
        Outport(Adr_start_motor, LOW);          // opreste motor
}
```

Cazul II – cu element elastic

- mai multe faze :
  - rotire motor fără deplasare obiect
  - faza de oscilație a resortului
  - faza de deplasare staționară

Ecuații integro-diferențiale complexe pentru fiecare fază:

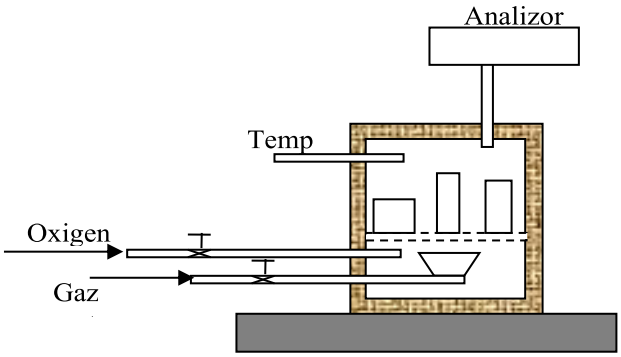
- lucrul mecanic este o integrală a forței și a deplasării
- forța de tracțiune este variabilă,
- accelerația nu este constantă (oscilează)
- forța de frecare este variabilă

Concluzie: dificil de descris analitic si descrierea nu este exactă datorită unor factori perturbatori care nu sunt luaţi în calcul.

Exemplul 2. Reglarea temperaturii unui cuptor cu gaz

Parametri:

- debit de gaz
- randament de ardere
- pierderi calorimetrice
- temperatură



mCH4 + nO2 = iCO2 +jCO +kH2O - ecuația de ardere a gazului metan

m = M_{CH4} /masa_molară(CH4)

n= M_{O2} /masa_molară(O2)

i, j, k, idem

$M_{CH4} = Q_{CH4} * t$

$M_{O2} = Q_{O2} * t$

$Q = M_{CH4} * \lambda_{CH4} * \eta$

$Q = (m1+m2+m3+Mcuptor)*c*(t_1 - t_0) + Qpierderi$

Factori perturbatori:

- temperatura, umiditatea externă
- presiunea gazului metan și a oxigenului
- neliniaritatea robinetelor de gaz
- timpul de propagare a căldurii/temperaturii în incintă

Ce se urmărește:

- menținerea cuptorului la o anumită temperatură
- menținerea unei anumite concentrații de CO_2 (factor de călire)

Controlul în buclă închisă:
Semnale:

- temperatura – semnal analogic
- concentrația de CO_2 – semnal analogic
- semnale de acționare a robinetelor de gaz și de oxigen – semnale analogice

1.3 Mersul lucrării

1. Se vor analiza conceptele prezentate în paragraful anterior și se vor da mai multe exemple practice de sisteme de control. Exemple posibile:
    a. Sistem de reglaj temperatură, nivel
    b. Sistem de monitorizare a unei clădiri
    c. Mașină automată de spălat
    d. Robot industrial
2. Pentru exemplele date se vor identifica: procesul controlat, sistemul de control, parametri de intrare/ieșire și de stare, semnale implicate, funcția de control, etc.
3. Se vor scrie ecuațiile care descriu diferitele procese controlate.

2 Microcontroloare și sisteme dedicate

2.1 Obiectivele lucrării

- prezentarea principalelor elemente arhitecturale ale unui microcontrolor, exemplificate prin circuitul PIC16F877
- prezentarea metodologiei și a instrumentelor de proiectare, execuție și testare a aplicațiilor dezvoltate cu ajutorul microcontroloarelor
- evaluarea posibilităților de implementare a unor aplicații dedicate și încapsulate folosind diverse variante de microcontroloare

2.2 Considerații teoretice

2.2.1 Microcontroloare

Microcontroloarele sunt circuite integrate care încorporează majoritatea componentelor necesare pentru realizarea unui micro-sistem de calcul dedicat:

- unitate centrală de prelucrare (UCP)
- memorie de program (MP) – păstrează aplicația ce rulează pe microcontrolor
- memorie de date (MD) – păstrează variabilele aplicației, registrele procesorului, memoria stivă și setul de registre cu funcții speciale (SRF – Special Register File)
- memorie nevolatilă (EEPROM) – păstrează parametrii de configurare sau alte date care trebuie păstrate și după decuplarea tensiunii de alimentare
- sistem de întreruperi – tratează atât întreruperile generate de componentele interne ale circuitului (ex: canal serial, contoare, etc.) cât și cele generate din exterior
- contoare și circuite de temporizare – componente folosite pentru: măsurarea timpului (ceasul sistemului), pentru generarea de semnale de frecvență prestabilite (ex: pentru transmisia serială de date), pentru numărarea de impulsuri, pentru generarea de întreruperi periodice, etc.
- interfețe de intrare/ieșire – asigură legătura dintre microcontrolor și mediul în care lucrează (procesul controlat, interfața utilizator sau un alt echipament de calcul); interfețele tipice care se regăsesc într-un microcontrolor sunt:

- interfețe paralele – asigură achiziția și generarea de semnale digitale
- interfețe seriale – folosite pentru comunicația cu alte componente "inteligente" inclusiv cu un calculator ierarhic superior (ex: un calculator personal)
- convertoare analog-numerice și numeric-analogice – folosite pentru achiziția și generarea de semnale analogice

- interfață de rețea – pentru comunicația în rețea (de obicei o rețea industrială, ca de exemplu CAN)

Microcontroloarele, după cum sugerează și numele, sunt destinate pentru dezvoltarea de aplicații de monitorizare și control. Ele oferă o soluție optimă pentru aplicații de complexitate redusă, în care parametrii de cost, dimensiune, consum și fiabilitate sunt dominanți în detrimentul celor de performanță și versatilitate.

2.2.2 Microcontroloare din familia Microchip-PIC

Firma Microchip (www.microchip.com) realizează o gama foarte diversă de microcontroloare (denumite PIC), de la variantele foarte ieftine, de dimensiuni mici (6 pini) și arhitectură pe 8 biți la variante mai performante, de dimensiuni mai mari (pana la 80 de pini) și având o arhitectură pe 16 biți. Variantele constructive diferă prin: performanţele UCP, dimensiunea instrucțiunilor (12, 14 sau 16 biți), dimensiunea și tipul memoriei de program și de date (de la 384octeti la 128Kocteti pentru memoria de program), tipul și numărul de interfețe incluse în structura internă a microcontrolorului (contoare, porturi paralele, canale seriale, interfețe de rețea, convertoare analog/numerice și numeric analogice, etc.). Microcontroloarele PIC au o arhitectură RISC de tip Harvard cu două magistrale (una pentru instrucțiuni și una pentru date) și memoria partajată în memorie de program și memorie de date; setul de instrucțiuni este simplu și se asigură compatibilitatea în sus (eng. "upward") de la variantele simple (instrucțiuni pe 12 biți) către cele mai performante (cu instrucțiuni pe 16 biți).

Alegerea variantei optime pentru o anumită aplicație se face pe baza următoarelor criterii:

- numărul de semnale de intrare și ieșire necesare
- tipul și numărul perifericelor necesare (RS232, USB, CAN, etc.)
- dimensiunea memoriei (de program, RAM și EEPROM)
- viteza procesorului
- dimensiunea fizică a circuitului

Principalele subclase de circuite PIC sunt: PIC10, PIC12, PIC14, PIC16, PIC18, PIC24. Caracteristicile acestor clase sunt sintetizate în anexa 1. Primele 3 sunt variante cu preț și dimensiuni mici, dar cu performante modeste; variantele PIC16 au performanta și cost mediu iar PIC18 și PIC24 sunt de performanta ridicata. Din punct de vedere arhitectural sunt procesoare pe 8 biți, cu un format al setului de instrucțiuni ce variază în funcție de clasă, de la 12 biți/instrucțiune, la 14 și respectiv 16 biți/instrucțiune.

2.2.3 Microcontrolorul PIC16F877

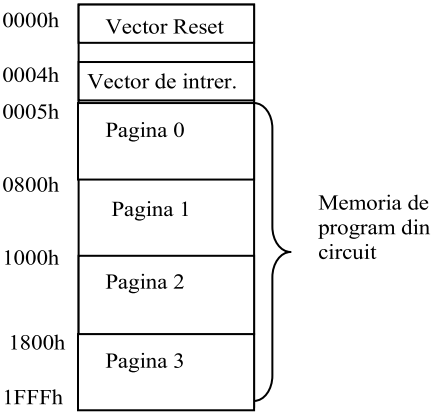
Pentru exemplificarea arhitecturii unui microcontrolor s-a ales unul din cele mai populare variante: PIC16F877. Principalele caracteristici ale acestui circuit sunt:

- arhitectură RISC performantă cu doar 35 de instrucțiuni; instrucțiunile au un format fix de 14 biți; adresare directă, indirectă și relativă
- frecvența de lucru 20MHz
- memorie de program de tip Flash cu capacitate maximă de 8K*14 biți
- memoria de date de tip RAM de maxim 368*8 biți
- memorie de date nevolatilă (EEPROM) de maxim 256*8 biți
- memorie stivă de 8 poziții
- 14 surse de întrerupere
- programabil pe o linie serială (pe 2 pini)
- contoare: Timer0 (8 biți), Timer1 (16 biți), Timer2 (8 biți)
- PWM cu rezoluție pe 10 biți
- convertor analog-numeric multicanal pe 10 biți
- interfețe seriale:
    - interfață sincronă - Synchronous Serial Port (SSP) cu SPI(Master mode) și I2C(Master/Slave)
    - interfață asincronă - Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) cu detecție de adresă pe 9-biți
- interfețe paralele - Parallel Slave Port (PSP) pe 8-biți

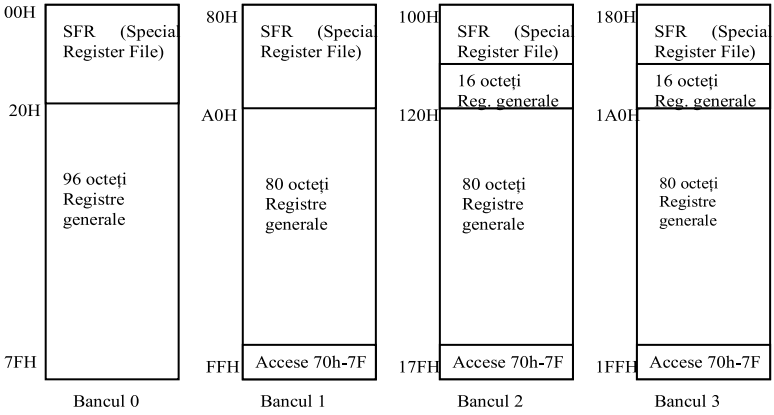
Manualul de utilizare al acestui circuit oferă o descriere detaliată a acestui circuit. În continuare se dau doar câteva date selective necesare pentru o mai bună înțelegere a funcționării acestui circuit.

2.2.3.1 Organizarea memoriei

Harta memoriei de program:



Harta memoriei de date:



Memoria de date este organizată pe 4 bancuri. În fiecare din cele 4 bancuri există zonă rezervată pentru registrele speciale ale microcontrolorului și o zonă utilizabilă pentru variabilele programului (registrele generale). Selectarea bancului curent se face cu ajutorul a 2 biți din registrul de stare (RP0 și RP1). Prin intermediul registrelor speciale programatorul poate să controleze funcționarea circuitului și poate să

acceseze (să citească și să scrie) diversele componente interne, în speță interfețe. De exemplu registrele PORTA (adr. 05H), PORTB (adr. 06h), PORTC (adr. 07h) și PORTD (adr. 08H) permit citirea și scrierea celor 4 porturi paralele ale circuitului, iar registrele TMR0, TMR1L, TMR1H și TMR2 controlează contoarele 0, 1 și 2. În mod similar fiecare resursă a circuitului are registre speciale asociate, unele pentru citirea/scrierea datelor propriu-zise, altele pentru programarea și configurarea acestora.

2.2.3.2 Setul de instrucțiuni

Microcontroloarele din familia PIC16 folosesc un format de instrucțiune de 14 biți. Instrucțiunile sunt de 3 tipuri:

- instrucțiuni pe octet
- instrucțiuni pe bit
- operații cu literal și de control

Pentru **instrucțiunile pe octet** parametrul "f" reprezintă indicatorul de registrul (file register) care se folosește în instrucțiune, iar parametrul "d" indicatorul de destinație. Dacă d=0 rezultatul se pune în registrul "W", iar dacă d=1 rezultatul se pune în registrul indicat de "f".

Pentru **instrucțiuni pe bit** "b" reprezintă un indicator al bitului care va fi afectat de instrucțiune, iar "f" este adresa registrului unde se află bitul. La **instrucțiunile cu literal sau de control** "k" reprezintă o constantă sau un literal reprezentat pe 8 sau 11 biți.

O instrucțiune se execută în unul sau cel mult 2 cicluri; un ciclu de instrucțiune durează 4 cicluri de ceas, ceea ce înseamnă că la o frecvență a ceasului de 4MHz o instrucțiune uzuală se execută în 1μs. Orice instrucțiune care operează cu un registru efectuează o operație de tip "citește-modifică-scrie". Astfel se efectuează o citire chiar și în cazul în care instrucțiunea precizează o operație de scriere. Tabelul de mai jos prezintă succint setul de instrucțiuni pentru familia PIC16.

Instrucțiune	Descriere	Instrucțiune	Descriere
Instrucțiuni pe octet		**Instrucțiuni pe bit**	
ADDWF f,d	Adună W și f	BCF f, b	Sterge bit din f
ANDWF f,d	ȘI logic W și f	BSF f, b	Setează bit în f
CLRF f	Sterge f	BTFSC f, b	Test bit și salt dacă este 0
CLRW	Sterge W	BTFSS f, b	Test bit și salt dacă este 1
COMF f,d	Complementează f	**Instrucțiuni cu literale sau de control**	
DECF f,d	Decrementează f	ADDLW k	Adună literal cu

			W
DECFSZ f,d	Decrementează f și salt dacă este 0	ANDLW k	ȘI literal cu W
INCF f,d	Incrementează f	CALL k	Apel rutină
INCFSZ f,d	Incrementează f și salt dacă este 0	CLRWDT	Sterge watchdog timer
IORWF f,d	SAU inclusiv W și f	GOTO k	Salt la adresă
MOVF f,d	Transfer f	IORLW k	SAU inclusiv cu literal și W
MOVWF f	Transfer W în f	MOVLW k	Transferă literal în W
NOP	Nicio operație	RETFIE	Revenire din întrerupere
RLF f,d	Rotație stânga prin Carry	RETLW k	Revenire cu literal în W
RRF f,d	Rotație dreapta prin Carry	RETURN	Revenire din rutină
SUBWF f,d	Scadere W din f	SLEEP	Trecere în mod Standby
SWAPF f,d	Schimbă 4 biți (nibbles) în f	SUBLW k	Scade W din literal
XORWF f,d	SAU exclusiv W și f	XORLW k	SAU exclusiv literal cu W

2.2.3.3 Utilizarea interfețelor (porturilor) paralele

Porturile paralele PORTA, PORTB, PORTC, PORTD și PORTE pot fi configurate la nivel de bit ca și intrări sau ieșiri. Fiecărui port îi este atașat un registru de configurare (TRISA pentru PORTA, TRISB pentru PORTB și așa mai departe) care determină direcția de intrare sau de ieșire a fiecărui bit al portului. Dacă în registrul TRISA un anumit bit este 0 atunci bitul corespunzător din PORTA este ieșire, iar dacă este setat pe 1 atunci bitul corespunzător din PORTA este intrare. Anumiți pini ai circuitului corespunzători porturilor paralele au funcții multiple (ex: intrare/ieșire, întrerupere, intrare analogică, etc.). Prin configurare se alege funcția dorită a pinului. Din această cauză unele funcții/interfețe sunt reciproc exclusive.

Secvență de program pentru inițializarea portului PORTA:

```
BCF STATUS, RP0 ;
BCF STATUS, RP1      ; Selectare Banc0 unde se află PORTA
CLRF PORTA           ; Inițializare (ștergere) PORTA
```

```
BSF STATUS, RP0          ; Selectare Banc 1, unde se află TRISA
MOVLW 0x06     ; Configurează toți pinii ca intrări digitale
MOVWF ADCON1          ; scriere registru de control conv. AD*
MOVLW 0xCF            ; Valoare pt. inițializarea direcției datelor
MOVWF TRISA            ; Setează RA<3:0> ca și intrări
                       ; RA<5:4> ca și ieșiri
                  ; TRISA<7:6> sunt întotdeauna citite ca "0".
```

Notă: unii pini ai portului A sunt multiplexați cu intrările în convertorul analog-digital

Portul D poate să funcționeze în regim de "port slave" în sensul că poate fi citit sau scris din exteriorul circuitului (în regim "slave") de către un alt circuit; în acest scop se folosesc semnalele RD\, WR\ și CS\ pentru citirea, scrierea și respectiv selectarea portului pentru citire/scriere.

2.2.3.4 Descrierea sistemului demonstrativ PICDEM 4 DEMONSTRATION BOARD

Pentru familiarizarea utilizatorilor cu modul de proiectare și programare a aplicațiilor bazate de aceste circuite, s-au dezvoltat diverse sisteme demonstrative sau sisteme de dezvoltare. Aceste sisteme sunt un ansamblu de componente hardware și software care permit utilizatorului rularea unor programe demonstrative predefinite și programarea de noi aplicații prin extinderea sau modificarea celor date ca exemplu.

Un astfel de sistem demonstrativ oferit de firma Microchip este "PICDEM 4 demonstration board", care are următoarele componente și caracteristici funcționale:

1. socluri de 8, 14 și 18 pini pentru diverse tipuri de microcontroloare PIC (la un moment dat un singur circuit PIC poate fi utilizat)
2. regulator de tensiune de 5V pentru alimentarea circuitului de la un adaptor sau baterie de 9V (curent generat 100mA)
3. transceiver și conector pentru transmisie serială asincronă RS23
4. conector pentru "depanare în circuit" (In-Circuit Debugger - ICD) .
5. 3 butoane pentru interacțiune cu utilizatorul
6. 4 potențiometre de 5Kohmi pentru testarea intrărilor analogice
7. afișaj LCD cu 2*16 caractere

8. 8 LED-uri conectate la porturile PORTA și PORTB
9. zonă pentru dezvoltări hardware (zonă prototip)
10. cuarț de 32.768 kHz pentru operarea timerului de timp-real
11. circuit de expandare PIC16LF72 I/O
12. circuit imprimat pentru driver de motor și transceiver LIN (Local Interconnect Network)

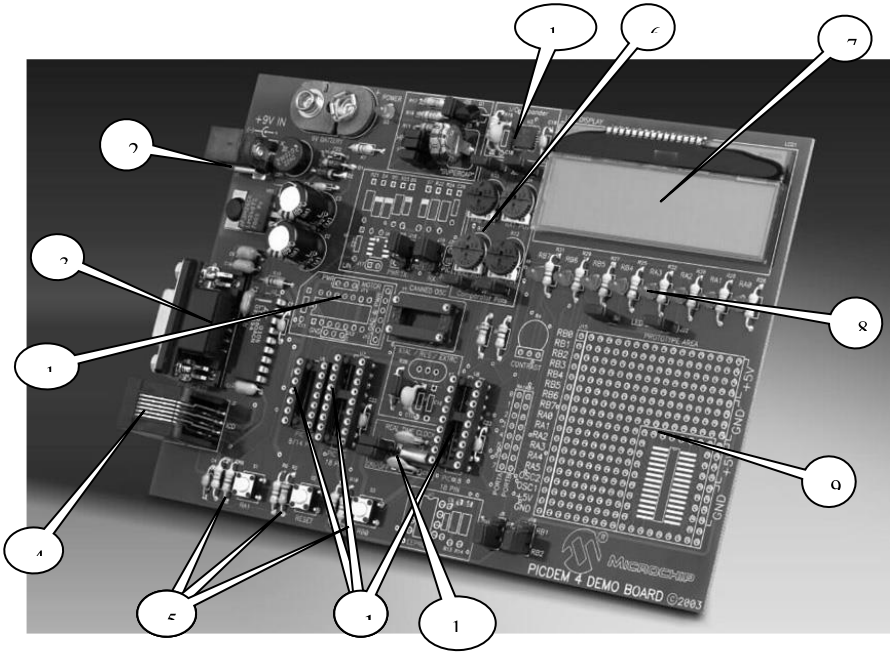


Figura 2. Vedere asupra plăcii demonstrative PICDEM 4

Detalii suplimentare privind placa PICDEM 4 pot fi găsite în Manualul de utilizare oferit de producător.

Descrierea Programului demonstrativ

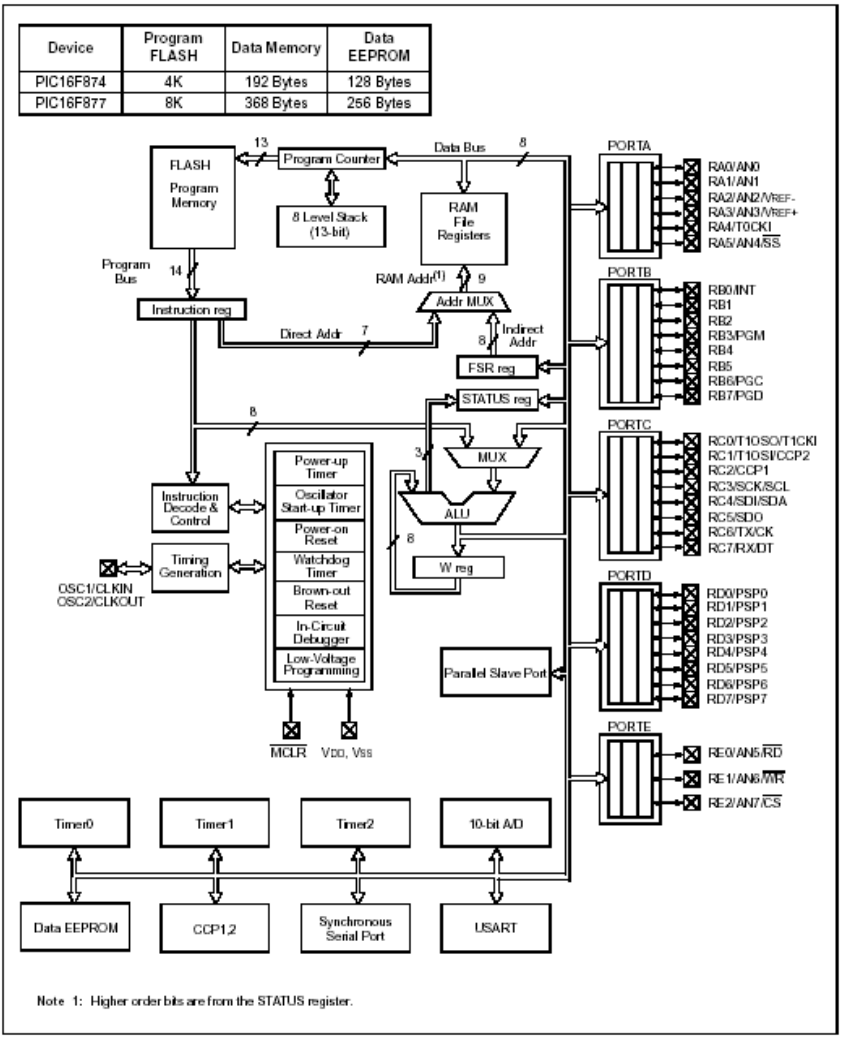
Programul demonstrativ implementează 2 funcții: voltmetru digital și ceas. Pentru prima funcție se folosește un potențiometru de pe placa PICDEM 4, convertorul analog numeric al microcontrolorului și afișajul LCD. Tensiunea reglata prin potențiometru în intervalul 0-5V este afișată periodic pe LCD. Pentru funcția de ceas se folosește un cuarț de 32KHz, contorul Timer1 al microcontrolorului și afișajul LCD. Cu ajutorul butoanelor SW1 și SW3 se poate seta timpul curent. La pornire ceasul indică 00:00:00. Butonul SW3 permite comutarea de pe modul voltmetru pe cel de ceas și invers.

2.3 Mersul lucrării

1. Se va analiza prin comparație arhitectură mai multor tipuri de microcontroloare PIC; se vor identifica componentele interne și facilitățile funcționale oferite
2. Se va face o comparație între un microcontrolor și un procesor uzual
3. Se va pune în funcțiune placa PICDEM 4 și se vor analiza diversele facilități ale plăcii
4. Se va analiza programul demonstrativ al plăcii (codul sursă); se vor identifica secvențele de program aferente componentelor/interfețelor de pe placă
5. Se vor face mici modificări în program în vederea schimbării funcționalității plăcii.

Anexa 1. Schema internă a circuitului PIC16F877

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Anexa 2. Secvențe din programul demonstrativ al plăcii PICDEM 4:

```
include <P16F628.INC>          ;fisier de descriere a configurației microcontrolorului
    ..................................................
    #define Switch1             PORTA,4 ; declararea poziției butoanelor în porturi
    #define Switch2             PORTA,5
    #define Switch3             PORTB,0
    #define TRISSwitch1     TRISA,4
    #define TRISSwitch2     TRISA,5
    #define TRISSwitch3     TRISB,0
    ..................................................
    WaitForSwitch1     macro              ; macro pentru a astepta apăsarea tastei SW1
          btfsc   Switch1                          ;asteaptă apăsarea tastei
          goto    $ -2
                        endm
    ..................................................
    WaitReleaseOfSwitch1     macro       ; macro de așteptare a eliberării tastei
           bcf          STATUS,RP1              ; alegere banc 0
           bcf          STATUS,RP0
          btfss   Switch1                          ;așteaptă eliberarea tastei
          goto    $ -1
              endm
    ..................................................
    WaitForMenuKeys macro              ;macro pentru asteptarea unei taste de meniu
           bcf          STATUS,RP1
           bcf          STATUS,RP0
      btfss   Switch3
      goto     $+4
      btfsc   Switch1
      goto     $-3
      endm
    ..........................................................

    ResetVector     CODE    0x00                ; salt la adresa de început a programului
           goto         start

    Main_Start        CODE

    start
    ;Initializare pini de intrare/iesire
           banksel     PORTA
           clrf    PORTA
           clrf    PORTB
           banksel     TRISA
           movlw   B'00111111'
           movwf   TRISA
           movlw   B'11110111'
           movwf   TRISB
           call    LongDelay           ;temporizare pentru extenderul de I/E
           call         IOExpLCDInit                ;Initializare LCD
           call    IOExpLCDHideCursor
    ;Ascunde cursor
           call         IOExpLCDLine1          ;Pune cursorul pe linia 1
```

```
        ;Display "Microchip"
          mIOExpLCDSendMessage    MsgMicrochip
        call    IOExpLCDLine2          ;Pune cursorul pe linia a 2-a
          mIOExpLCDSendMessage    MsgPICDEM4  ;Display " PICDEM 4"
          call    LongDelay


.....................................................................
LongDelay                                   ;funcţie de întârziere de aprox. 1s
        banksel Temp1
          movlw     0xFF
          movwf     Temp1
          movwf     Temp2
          movlw     0x05
          movwf     Temp3
LongDelay_1
          decfsz    Temp1,F
          goto      LongDelay_1
          decfsz    Temp2,F
          goto      LongDelay_1
          decfsz    Temp3,F
          goto      LongDelay_1
          return
```

Anexa 3 Comparație între familiile de microcontoloare PIC

TIP	pret	Arhit.	mem. progr Ko	eeprom	RAM	I/O	Canale ADC	Com-parat
PIC10	0,4-0,5$	8 biti	0,375- 0,75	0	16-24	4	0-2/ 8 biti	0-1
PIC12	0,6-0,7	8	0,75-1,75	0-256	25-128	6	0-4/ 10 biti	0-1
PIC14	5,5	8	7 (OTP)	0	192	20	0/8	2
PIC16	1-3$	8	1-14 otp/ flash	0-256	25-368	6-52	0-12	0-2
PIC18	2,5-7 $	8	flash	0-1024	512-3930	16-70	4-16/ 10-12 biti	0-2

TIP	Con-toare	HW-RTC	Inter-fete.	Vit. MHz	Nr pini	Vmin	Vmax	port paralel	Variante
PIC10	1/8 biti	nu		4	6	2V	5,5V	nu	
PIC12	1/8 biti	nu		4-20	8	2V	5,5V	nu	12
PIC14	1/8 1/16		SMB	20	28	2,7V	6V		1
PIC16	1-2/8 1/16 1/wd		USART I2C SPI MI2C	20-40	18-64	2V	5,5V	nu/ PSP	94 !!!
PIC18	1-2/8 1-3/16 1/wd		2USART I2C SPI MI2C	25-64	18-100	2V	5,5V	nu/ PSP	151 !!!!

3 Tehnici de programare a aplicațiilor pe sisteme cu microcontroloare

3.1 Obiectivul lucrării

Lucrarea urmărește studierea tehnicilor de dezvoltare și depanare a programelor scrise pentru sisteme de calcul dedicate; studierea mediului de dezvoltare MPLAB IDE (Integrated Development Environment) ca mijloc de proiectare, implementare și testare a aplicațiilor pentru microcontroloare din familia PIC.

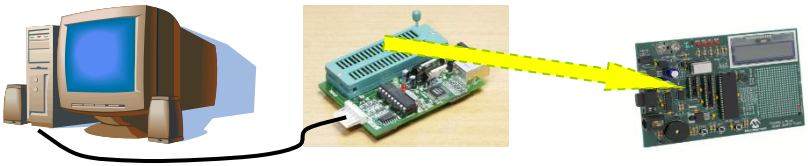
3.2 Considerații teoretice

3.2.1 Tehnici de programare și depanare a aplicațiilor dezvoltate pe sisteme cu microcontroloare

Ca metodologie de lucru dezvoltarea aplicațiilor pe platforme încapsulate, bazate pe microcontroloare diferă semnificativ de metodologia de dezvoltare a aplicațiilor pentru calculatoare personale. În general sistemele bazate pe microcontroloare nu dispun de resursele necesare pentru dezvoltarea de aplicații (memorie, tastatură, afișaj, sistem de operare, sistem de gestiune a fișierelor, etc.). De aceea dezvoltarea aplicațiilor pentru aceste sisteme se realizează pe un sistem de calcul uzual (ex.: PC), urmând ca programul executabil generat în urma compilării să fie transferat către sistemul țintă (target system) sau înscris direct în memoria microcontrolorului.

Anumite firme (inclusiv cele care furnizează circuitele) oferă diferite instrumente de proiectare, programare și testare a sistemelor încapsulate bazate pe microcontroloare. În funcție de serviciile oferite și de complexitatea acestora putem identifica următoarele categorii de instrumente de dezvoltare:

- programator de microcontroloare – ansamblu hardware-software relativ simplu care oferă funcții minime necesare pentru programarea (înscrierea) aplicațiilor în memoria internă a microcontrolorului; nu oferă facilități de dezvoltare a programelor sau de depanare a acestora



- starter kit – ansamblu minimal necesar pentru programarea și testarea programelor; se oferă o schemă generală simplă, utilă în verificarea și testarea unei familii de microcontroloare și instrumente de programare care includ un asamblor și un link-editor

- mediu integrat de dezvoltare IDE – o aplicație complexă ce permite parcurgerea principalelor etape de proiectare a unei aplicații, de la editarea programului, compilare, link-editare, transferul către un sistem țintă și execuția programului; mediul poate să includă un simulator de microcontroler, caz în care execuția aplicației se poate face pe calculatorul personal; în cazul microcontroloarelor PIC firma Microchip oferă în acest scop produsul MPLAB IDE
- ICD – in-circuit-debugger – circuit ce permite depanarea și testarea programelor în schema finală a aplicației (în sistemul țintă)

- ICE – in-circuit-emulator – un dispozitiv ce permite ca PC-ul să ia locul microcontrolorului din schema țintă și emularea funcționării acestuia; este cel mai performant instrument de testare a sistemelor încapsulate; programatorul poate să urmărească în detaliu funcționarea sistemului în diferite regimuri de lucru (pas-cu-pas, automat, etc.) și pot fi controlate componentele microcontrolorului (registre generale și speciale, interfețe, etc.).

3.2.2 Mediul de dezvoltare MPLAB IDE

MPLAB IDE este un mediu integrat de dezvoltare (IDE – Integrated Development Environment) oferit de firma Microchip pentru familiile de microcontroloare PIC12, PIC 16, PIC18, etc. Include:

- manager de proiecte
- editor de program sursă
- asamblor
- editor de legături
- depanator (debugger)
- motor de execuție

Aplicația permite și integrarea altor componente (ex.: compilator C, depanator, programator, ICD, ICE) realizate fie de firma Microchip fie de alte firme (ex. CCS).

Figura de mai jos indică schema de administrare a unui proiect prin MPLAB:

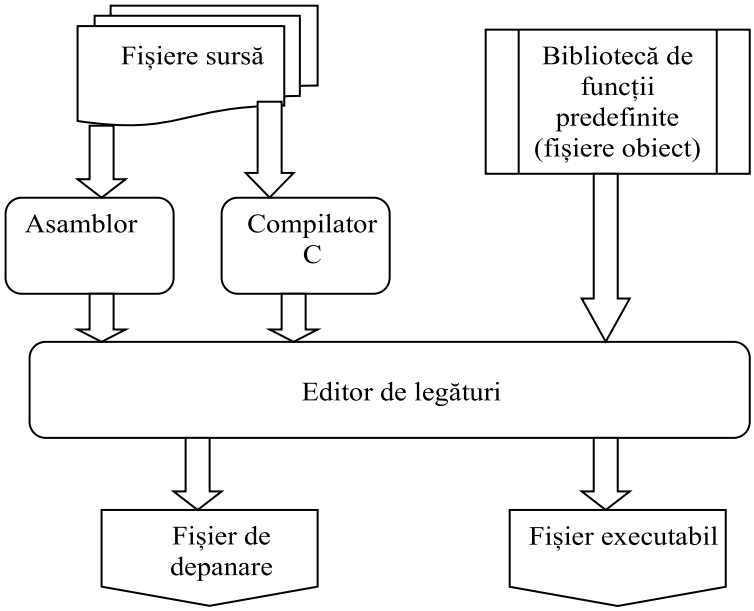


Figura 1. Administrarea unui proiect în MPLAB

Pașii necesari pentru dezvoltarea unei aplicații:

a. Lansarea mediului MPLAB IDE

- *Start>Programs>Microchip>MPLAB IDE vx.xx>MPLAB IDE*

b. Selectarea dispozitivului

- trebuie să preceadă crearea unui nou proiect
- din meniu se alege: *Configure->Select device*
- se alege tipul circuitului folosit (ex.: *PIC16F877*); mediul va indica tipurile de dispozitive suportate pentru programarea circuitului (ex.: MPLAB ICD 2), limbajele acceptate și instrumentele de depanare utilizabile

c. Crearea unui proiect - cu ajutorul "project wizard"

- din meniu se alege: *Project>Project Wizard*
- în pașii următori se aleg: tipul de dispozitiv și setul de instrumente de compilare și link-editare; se observă că pe lângă instrumentele Microchip pot fi folosite instrumente oferite de alți producători (ex.: compilator C al firmei HI-TECH)
- se alege un nume pentru noul proiect (ex.: Test_x)
- se adaugă fișiere sursă; pentru început se poate adăuga un fisier "model" (template) care conține elementele inițiale de configurare pentru fiecare circuit în parte (ex.: C:\Program Files\Microchip\MPASM Suite\Template\Object\16F877tmpo.asm); se apasă litera de la începutul fișierului până apare "C" care va copia fișierul în proiect
- se adaugă un fisier necesar pentru link-editare (ex.: C:\Program Files\Microchip\MPASM Suite\LKR\18F8722.lkr sau .... \18F8722i.lkr dacă se folosește ICD 2)
- proiectul creat se poate vizualiza cu: *View>Project*.; fișierele pot fi selectate pentru editare (dublu-click pe fisier sau click-dreapta și Edit)
- se construiește proiectul (asamblare, link-editare și conversie în format HEX) cu: *Project>Build All* sau prin apăsarea icoanei corespunzătoare la "Build All"
- pentru scriere de program se deschide fișierul model (dublu-click pe 16F877tmpo.asm) și se adaugă instrucțiuni în "Main" după comentariul: *remaining code goes here* ; se va adăuga următorul cod:

```
        clrf w_temp
        movwf PORTC ; clear PORTC
        movwf TRISC          ; configure PORTC as all
outputs
Init
        clrf COUNT           ; initialize counter
IncCount
        incf COUNT,F
```

```
        movf COUNT,W          ; increase count and
        movwf PORTC  ; display on PORTC
        call Delay            ; go to Delay subroutine
        goto IncCount         ; infinite loop
Delay
        movlw 0x40
        movwf DVAR2           ; set outer delay loop
DelayOuter
        movlw 0xFF
        movwf DVAR            ; set inner delay loop
DelayInner
        decfsz DVAR,F
        goto DelayInner
        decfsz DVAR2,F
        goto DelayOuter
        return
```

- în zona de declarare a variabilelor se va adăuga:

```
COUNT  RES 1       ; Counter
DVAR   RES 1  ; inner loop counter
DVAR2  RES 1       ; outer loop counter
```

- se reconstruiește proiectul și se corectează eventualele erori

d. Testarea codului – cu ajutorul simulatorului sau a unui sistem "țintă", cum ar fi de exemplu IDC 2 (in-circuit debugger) sau ICE (in-circuit emulator);

- alegerea instrumentului de testare se face prin : *Debugger>Select Tool>MPLAB ICD 2 sau Debugger>Select Tool>MPLAM SIM* (pentru simulator);
- în primul caz conectarea la ICD 2 se face prin: *Debugger>Connect*; în caz de eroare se parcurge *Debugger>MPLAB ICD2 setup wizard*
- rularea programului în regim continuu sau pas-cu-pas se face cu comenzile din Debugger: RUN (F9), Step into(F7), Step Over(F8), Step Out și Reset; pentru rularea cu puncte de întrerupere se folosesc facilitățile "Breakpoint".

3.3 Mersul lucrării

1. Se va instala mediul de dezvoltare MPLAB de pe CD-ul furnizat de firma Microchip sau de pe pagina web a firmei (www.microchip.com).
2. Se vor urma pașii de dezvoltare indicați în Manualul de utilizare a aplicației MPLAB (anexa: MPLAB.pdf).
3. Se va testa funcționarea aplicației create cu ajutorul simulatorului incorporat în mediul de depanare; se vor testa diversele facilități de depanare:
    a. execuția pas-cu-pas și în regim automat (step-in, step-out, Run, stop, Reset, etc.)
    b. vizualizarea variabilelor (watch)
4. Se va testa funcționarea aplicației create cu ajutorul circuitului MPLAM ICD 2;
    a. se conectează circuitul ICD 2 la calculator
    b. se conectează placa țintă (ex.: PICDEM 4) la ICD 2
    c. se încarcă programul în microcontrolerul de pe placa țintă
    d. se execută aplicația în regim pas-cu-pas și continuu

4 Aplicație RFID (RFID - Radio Frequency Identification) dezvoltată cu ajutorul unui microcontroler

4.1 Obiectivul lucrării

Lucrarea își propune studierea unui exemplu de implementare a unei aplicații RFID pe platformă de tip microcontroler. Se analizează un alt mediu de programare pentru microcontroloare care include un compilator "C" dedicat pentru familia de microcontroloare PIC.

4.2 Considerații teoretice

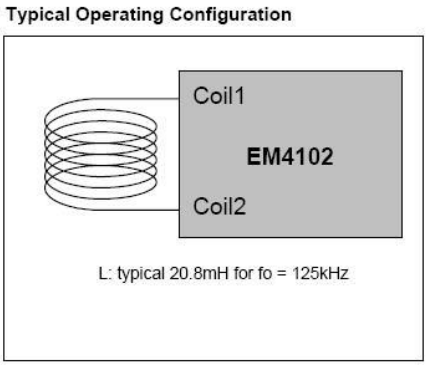
În diverse aplicații se solicită identificarea automată a persoanelor sau produselor pe baza unor coduri atașate. Cele mai utilizate sisteme de codificare și recunoaștere automată sunt cele pe bază de coduri de bare și respectiv cele de tip RFID. În primul caz codurile sunt marcaje pe suport de hârtie ușor de identificat pe cale optică. Marcajele se pot tipări pe o linie sau pe două direcții. Codurile sunt formate dintr-un număr limitat de cifre și o dată aplicate nu pot fi modificate. Pentru a fi identificat codul trebuie să fie adus în zona de vizibilitate a cititorului optic.

Mult mai versatile sunt etichetele (tag-urile sau transponderele) de tip RFID care pot fi identificate și uneori chiar modificate prin unde radio. Din punct de vedere principial un tag RFID este un microcircuit dotat cu o antenă radio încorporat într-un suport de plastic, hârtie, etc. Cititorul RFID emite o undă radio care alimentează tagul RFID și în același timp solicită codul înscris în tag. În preajma unui cititor, tagul emite o undă radio care conține codul tag-ului respectiv. De obicei codul conține o secvență de lungime predefinită de biți (40 , 64, 96 biți de date). În cazul tag-urilor mai evoluate sunt permise operații de ștergere și rescriere repetate, iar capacitatea de stocare a datelor este semnificativ mai mare.

Astfel de circuite de identificare se pot utiliza pentru evidența automată a mișcării produselor sau pentru identificarea persoanelor autorizate să acceseze un anumit perimetru. Există implementări de sisteme de autentificare a persoanelor bazate pe tag-uri RFID ce permit stocarea datelor relevante ale persoanei (date demografice sau informații medicale).

În principiu pentru a construi un cititor RFID sunt necesare următoarele elemente:

- o antenă construită sub forma unei bobine
- un emițător-receptor radio specializat pe standardul RFIF
- un circuit "inteligent" pentru interpretarea mesajelor și transmiterea acestora pe cale serială la un calculator; de obicei se folosește un microcontroler



Tag-urile RFID sau Transpoderele conțin un circuit integrat și o bobină pe post de antenă. Ele pot fi active sau pasive. Cele active conțin propria sursă de energie folosită în păstrarea informației în memorie și comunicare. Cele pasive se bazează pe energia electrică a frecvenței purtătoare a semnalului radio generat de cititor. Acestea conțin o memorie de tip EEPROM pentru a păstra informația în timpul în care transponderul (tag-ul) nu se află în aria unei antene.

În funcție de complexitate tag-urile au drepturi de citire, scriere, protecție cu parolă, criptare, protecție la suprapunerea mai multor tag-uri pe aceeași antenă, rescriere multiplă, etc. Cele incluse în kit-ul ce se studiază în acest laborator sunt: EM 4102 – citire, EM4150 – citire, scriere, parolă pe 32 biți. Date suplimentare despre aceste dispozitive pot fi obținute din fișele de catalog ale circuitelor (www.emmicroelectronic.com/webfiles/Product/RFID/DS/EM4102_DS.pdf,www.emmicroelectronic.com/webfiles/Product/RFID/DS/EM4150_DS.pdf)

Antena este un element cheie al sistemului RFID. Dimensiunea, forma și calitatea antenei influențează distanța comunicării. Odată cu dimensiunea crește și raza de acțiune a acesteia. Este un echilibru între limita maximă de construcție a antenei și limitele de siguranță ale parametrilor tensiunilor care vor alimenta circuitul integrat al tag-urilor.

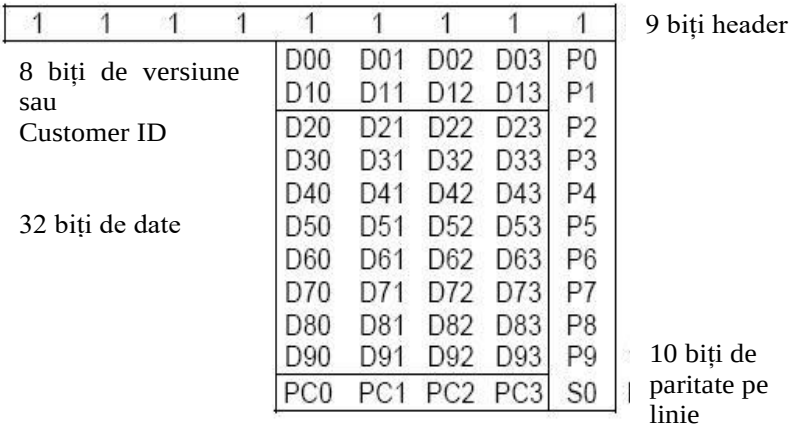
Microcontrolerul gestionează comunicația cu transpoderele și cu calculatorul gazdă. Acesta generează o frecvență purtătoare către antenă care prin fluctuații de câmp detectează prezența transponderelor la o distanță, ce poate să varieze de la câțiva centimetri la câțiva zeci de centimetri (aprox. 5-10cm pentru kit-ul utilizat în laborator). Comunicația cu calculatorul gazdă se face pe un canal serial asincron de tip RS232 sau RS485. Folosirea standardului RS485 permite conectarea mai multor dispozitive de tip RFID sau cu alte funcționalități pe același tronson de cablu; printr-un sistem de adresare calculatorul gazdă, care joacă rolul de master poate citi succesiv dispozitivele slave conectate în circuit. Deoarece un calculator obișnuit de tip PC nu are o interfață RS485, ci doar una de tip RS232, trebuie să se utilizeze un circuit de conversie RS485-RS232. În kit-ul studiat acest convertor este realizat cu un microcontroler PIC în care s-a înscris un program de conversie.

În schemele anexate se pot identifica componentele cititorului RFID și ale convertorului RS232-485. Date despre cele două standarde de comunicație seriale RS232 și RS485 pot fi obținute de pe Internet (www.lammertbies.nl/comm/info/RS-232_specs.html, www.lammertbies.nl/comm/info/RS-485.html)

Un cititor RFID funcționează pe mai multe game de frecvență: 50-500 kHz, 13.6 MHz și 0.9-2.5 GHz. Cele de frecvență mică sunt mai folosite datorită costului redus raportat la o performanță satisfăcătoare.

Transponderele EM 4102 (Read-only)

Acestea conțin 64 biți de informație structurați astfel : 9 biți de header, 40 biți de date(D), 10 biți de paritate pe rând (P), 4 biți de paritate pe coloană (C) și un bit de 0 de stop (S0).



Cei 40 biți de date sunt structurați după cum arată tabelul de mai jos: 8 biți alocați pentru câmpul Customer ID și 4 secvențe de 8 biți pentru Tag Id, care împreună (32 de biți) pot codifica peste 4 miliarde de id-uri unice.

Customer ID	Tag Id	Tag Id	Tag Id	Tag Id

Atunci când tag-ul EM 4102 este poziționat în zona cititorului RFID, care emite un semnal radio de 125kHz, va transmite în mod continuu secvența de 40 de biți. Secvența va fi preluată și interpretată de cititor.

Pentru evaluarea unui sistem de tip RFID se folosește un sistem de dezvoltare RFID oferit de firma CCS Inc. (Custom Computer Services, http://www.ccsinfo.com/product_info.php?products_id=RFIDkit). Firma este specializată în instrumente și platforme de dezvoltare pentru microcontroloare, în speță circuite din familia PIC.

Sistemul de dezvoltare este compus din următoarele componente:

1. placă "Cititor RFID"
2. placă de conversie RS232-RS485
3. modul de programare și depanare ICD U40 (In circuit debugger, USB)
4. alimentator de rețea
5. trei tipuri de transpondere (tag-uri RFID)
6. cabluri de legătură: RS232, RS485, USB și de depanare (DEP)

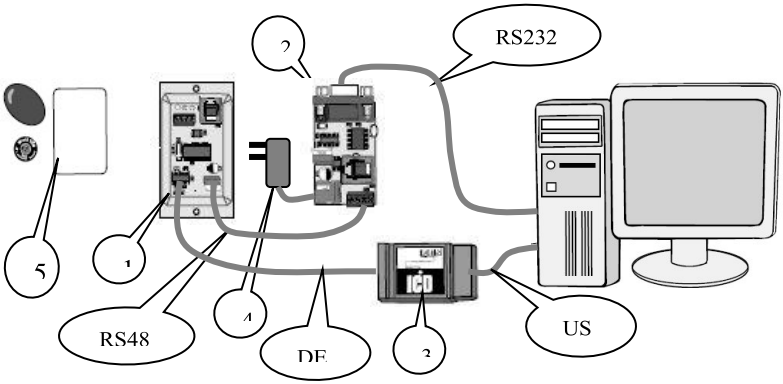


Figura 1. Sistemul de dezvoltare RFID

Pentru dezvoltare aplicației se folosește mediul PCW IDE și compilatorul PICC. Mediul se va instala de pe CD-ul atașat sistemului de dezvoltare. De asemenea se va instala driver-ul pentru modulul ICD-U40, în momentul în care circuitul este cuplat la o intrare USB a calculatorului gazdă. Pentru detalii privind modul de instalare se va consulta manualul de utilizare al sistemului de dezvoltare.

Se realizează schema din figura 1, iar alimentatorul se cuplează la rețea. Sistemul este pregătit pentru realizarea, încărcarea și execuția primului program scris în limbajul PICC (variantă de limbaj C pentru circuitele PIC). Programul aprinde și stinge un LED pe placa țintă, echivalent cu un program "Hello world" pentru un PC.

După lansarea mediului PCW IDE se parcurg următorii pași:

a. Editarea programului

- se închid eventualele fișiere deschise prin: *File>Close All* (File= prima icoană din meniu)
- se creează un nou fisier sursă cu: *File>New* și se atribuie numele de *EX3.C*
- se copiază programul din Anexa1 denumit EX3.C în noul fisier creat; programul cuprinde:
  - directiva "include <16F876A.h>" pentru a include declarațiile specifice pentru circuitul cu care se lucrează
  - linia de configurare a "fuzibilelor"; detalii despre configurarea fuzibilelor se obțin prin *View>Valid Fuses*
  - un set de 3 declarații care fac legătura dintre pinii microcontrolorului și cele 3 LED-uri de pe placă
  - secvența de program care aprinde și stinge la infinit unul dintre LED-uri; pentru temporizare se folosește rutina

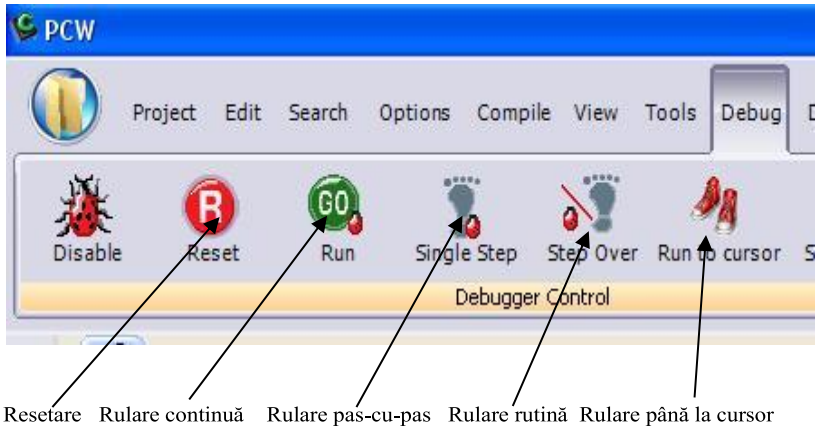
"delay_ms()" iar pentru setarea și resetarea ieșirii pentru LED se folosesc funcțiile: *output_low*() și respectiv *output_high()*

- se compilează programul prin: *Compile>Compile*; compilarea se va face cu opțiunea 14 Bit
- se salvează fișierul prin: File>Save All

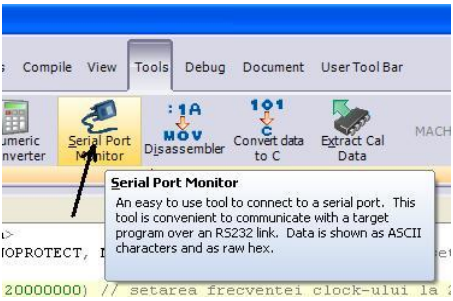
b. Crearea unui proiect
- se construiește un nou proiect prin: *Project>Create*
- se alege fișierul sursă fișierul creat anterior
- se setează tipul de dispozitiv pentru care se generează proiectul: *PIC16F876A*
- se apasă butonul *Apply*

c. Încărcarea și lansarea programului
- programul se descarcă prin: *Tools>ICD* sau prin *Compile>Program Chip>ICD*
- pe placă se observă aprinderea și stingerea periodică a LED-ului verde

d. Rularea programului în regim de depanare (debugger)
- programul se descarcă și se rulează prin: *Debug* sau *Compile>Debug>PCW*
- rularea programului se poate face în regim continuu sau pas-cu-pas, conform schemei de mai jos; prin apăsarea butonului RUN pe placă se observă aprinderea și stingerea LED-ului.



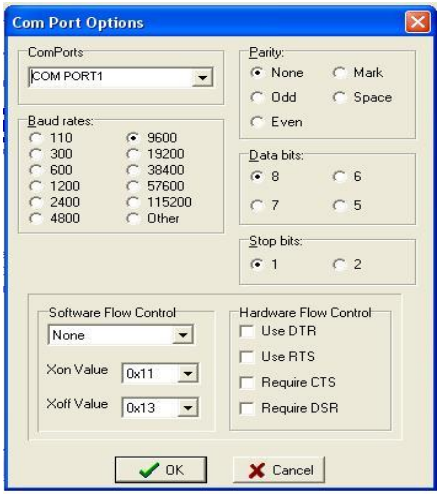
TOCMAI AȚI REALIZAT PRIMUL PROGRAM ÎN PICC COMPILER!



Următorul program testat (ex5.c) realizează comunicația dintre placa țintă și PC prin interfața serială asincronă RS232. Se închide proiectul anterior și se creează unul nou în mod similar cu cazul precedent. În fișierul sursă se va copia exemplul 5 din Anexă.

Pentru monitorizarea canalului serial se pornește un utilitar (similar cu HyperTerminal din Windows) prin secvența: *Tools>Serial Port Monitor*. Înainte de a continua, va trebui să conectam de aceasta dată și cablul serial la interfața RS232 de la PC pentru a putea facilita comunicarea cu sistemul nostru RFID.

Atenție, CABLUL SERIAL SE VA CONECTA ÎNAINTE DE ALIMENTAREA DE LA REȚEA A PLĂCII



Se configurează apoi portul serial pe care se va efectua comunicația, conform schemei alăturate. Programul va aștepta citirea unui caracter de la monitorul serial (un caracter ASCII sau un cod HEX). Pentru o mai bună siguranță, se recomandă trimiterea lor pe căsuța de editare HEX după modelul: Hex: 47 0d 0a ceea ce înseamnă în codificare ASCII: G, CR, LF. După cum reiese și din codul sursă, la apăsarea tastei G se va aprinde LED-ul verde, R LED-ul roșu, și O ambele LED-uri.

După aceste exerciții premergătoare se poate trece la testarea programului care implementează funcția de citire RFID.

4.3 Mersul lucrării

1. Se vor studia protocoalele seriale RS232 și RS485 pe baza documentației de la link-urile indicate în paragraful precedent; identificați diferențele dintre cele 2 standarde.
2. Rulați exemplele de program descrise mai sus, iar apoi modificați programele prin adăugarea de noi funcționalități (ex.: aprinderea mai multor LED-uri, într-o anumită secvență).
3. Folosind exemplele de mai sus, scrieți un program care citește un sir de n caractere, și în funcție de textul transmis (și identificat de program) va aprinde diferite combinații de LED-uri. Programul va rula în ciclu infinit.
4. Se va scrie un program care citește pachetul de date trimis de un transponder de tip EM 4102 și îl va transmite la PC pentru afișarea pe ecran; se va folosi programul Ex7 din anexă, care are următoarele elemente:

   - directivele de includere a unor module specifice pentru tag-uri, convertorul RS232-485 și unele utilitare

   #include <em4095.c> //controls the reader IC
   #include <em4102.c> // allows reading 4102 transponders
   #include <rs485.c> // protocol de comunicații
   #include <utilities.c>

   - în programul principal (main) apar următoarele funcții :

   rf_init(); //inițializare cititor RFID
   rf_powerUp(); //alimentarea antenei
   rs485_init(); //inițializarea conversiei RS232-485

   - directive de declarare a variabilelor:
     - int8 – întreg pe 8 biți
     - int32 – întreg pe 32 biți
   - funcții utile:
     - int 32 make32(int a, int b, int c, int d) // împachetare date pe 32 biți (conform structurii pachetului de date trimis de EM //4102)
     - sprintf(sir_de_caractere,"mesaj"); // copiază parametrul 2 în șirul de caractere dat de parametrul 1
     - boolean read_4102(msg) - > returnează true dacă pachetul de 40 biți din tag a fost salvat în variabila msg.
     - char rs485getc() – așteaptă citirea unui caracter de la tastatură pe consola de SIOW (serial port monitor)
     - void rs485send(msg) // trimite pe consolă mesajul indicat de "msg"
     - itoa(customerCode,10, msg) //conversie int to ascii

5. Testați codul ex8.c în cadrul unui nou proiect. Remarcați similitudinea cu programul realizat la problema 3.3; de această dată codul nu se va mai citi de la tastatură ci va fi trimis de tag-ul RFID apropiat de antenă.

Anexa 1 Programe

Programul Ex3

```
#include <16F876A.h>
#fuses HS, NOWDT, NOPROTECT, NOLVP, NOBROWNOUT, PUT // biți de setare

#USE delay(clock = 20000000) // setarea frecvenței clock-ului la 20Mhz

#DEFINE GREEN_LED  PIN_C3 // redenumirea pinului 4 din PORTC
#DEFINE YELLOW_LED PIN_C4 // redenumirea pinului 5 din PORTC
#DEFINE RED_LED    PIN_C5 // redenumirea pinului 6 din PORTC

void main()
{
 while (true)
   {
     output_low(GREEN_LED); // în logica negativă, 0 -> LED aprins
     delay_ms(1000); // ține LED-ul aprins 1000 ms = 1 sec
     output_high(GREEN_LED); // 1 -> LED stins
     delay_ms(1000);
   }
}
```

Programul Ex5

```
#include "rfid.h"
#define RS485_ID          0x11
#define RS485_USE_EXT_INT  FALSE
#define ADAPTER_RS485_ID   0X7F
#include <rs485.c>

 int msg[32] ;
typedef enum {OFF, GREEN, RED } Ledcolor;

 void twoColorLed (Ledcolor color) {
                switch (color) {
                  case OFF: {
                            output_low(PIN_A3);
                            output_low(PIN_A5);
                            break;
                           }
                  case GREEN:{
                            output_high(PIN_A3);
                            output_low(PIN_A5);
                            break;
                           }
```

```
                case RED:{
                        output_low(PIN_A3);
                         output_high(PIN_A5);
                         break;
                        }

                   }
               }
// trimite un sir de caractere consolei SIOW de comunicare serială
void RS485send(char * s){
                int8 size;
                for(size = 0 ; s[size]!= '\0'; ++size)
                     rs485_wait_for_bus(FALSE);
                while (!rs485_send_message(ADAPTER_RS485_ID, size, s)) {
                  delay_ms(RS485_ID);
                }
               }
               /* citește un caracter din monitorul comunicării seriale
                  SIOW. Din căsuța de editare HEX se va trimite codul hex
                            al cifrelor G , R sau O urmate de caracterele 0D 0A.
                   Ex: 47 0d 0a, pt. caracterul G
               */
               char RS485getc(){
                 rs485_get_message(msg, TRUE);
                 return msg[2];
               }

 void main(void) {
                 output_low(GREEN_LED); //SHOW POWER IS ON
                 rs485_init();   // inițializare comunicare serială
                              // + conversie rs485 -> rs232

                 while (true) {
                  sprintf(msg, "(O)ff , (G)reen, (R)ed \n\r");
                  RS485send(msg);
                  switch(toupper(RS485getc())) {
                    case 'O': twoColorLed(OFF); break;
                    case 'G': twoColorLed(GREEN); break;
                    case 'R': twoColorLed(RED); break;
                  }
                 }
               }
```

Programul Ex7

```
#include "rfid.h"
```

```
#include <em4095.c> //controls the reader IC
#include <em4102.c> // allows reading 4102 transponders
#include <rs485.c>
#include <utilities.c>
int8 msg[5];
void main(){
                    int8 customerCode;
                    int32 tagNum;

                    rf_init();          //initialize the RF reader
                    rf_powerUp();       //power up the antenna
                    rs485_init();      // initialize rs485 communication
                    output_low(GREEN_LED);// show the board is powered and ready
                    for (;;){
                      if (read_4102(msg)) {
                        customerCode = msg[0];
                        tagNum = make32(msg[1], msg[2],msg[3],msg[4]);

                        sprintf(msg,"customer code: %u\n\r ", customerCode);
                        RS485send(msg);

                        sprintf(msg,"Tag number: %lu\n\r ", tagNum);
                        RS485send(msg);
                      }
                    }//END FOR
                  }//END MAIN
```

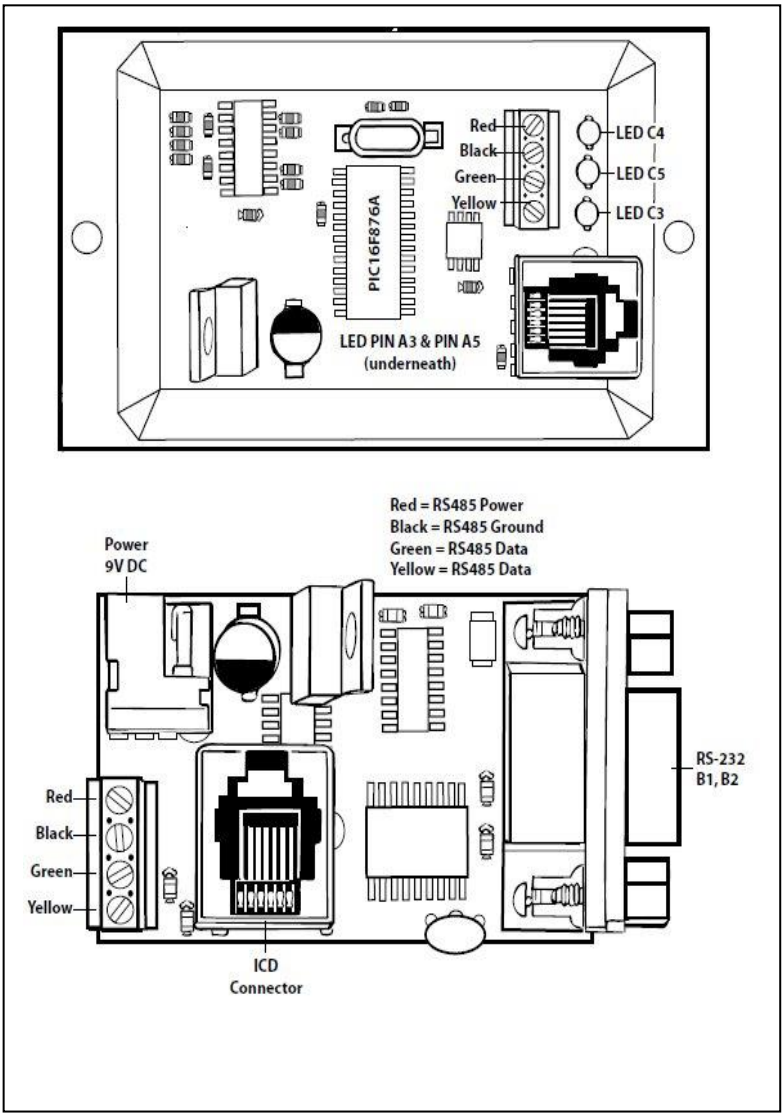
Programul Ex8

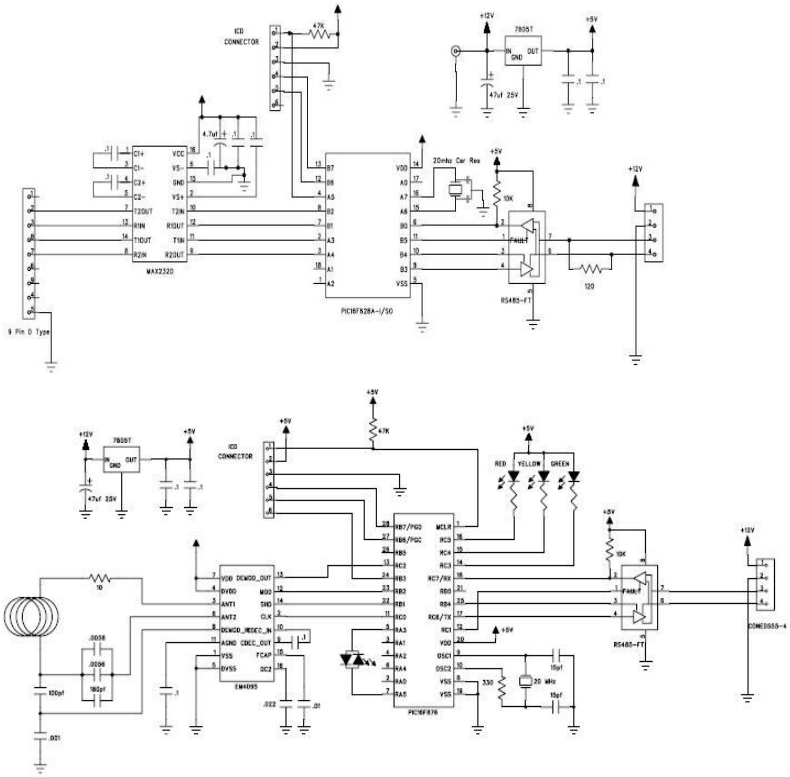
```
#include "rfid.h"
#include <em4095.c>
#include <em4102.c>
#include <rs485.c>
#include <stdlib.h>

int8 msg[32];
#include "utilities.c"

void main(){
  int8 customerCode, code;
  int32 tagNum, tag_ID;

  rf_init();
  rf_powerUp();
  rs485_init();
  output_low(YELLOW_LED);
```

```
  /*
  sprintf(msg, "Enter the customer code: ");
  RS485send(msg);
  code = RS485getInt();
    output_low(GREEN_LED);
    delay_ms(1000);
    output_high(GREEN_LED);

  sprintf(msg, "\n\n\r");
  RS485send(msg);

  sprintf(msg, "Enter the tag ID: ");
  Rs485send(msg);
  tag_ID = Rs485getI32();
    output_low(GREEN_LED);
    delay_ms(1000);
    output_high(GREEN_LED);

  sprintf(msg,"\n\n\rScanning...");
  Rs485send(msg);*/
  code = 176 ;
  tag_ID = 531324;
  for(;;){
     if (read_4102(msg)){

        customerCode =msg[0];
        tagNum = make32(msg[1], msg[2], msg[3], msg[4]);
       //itoa(customerCode,10, msg);//send code
        //RS485send(msg);
      //sprintf(msg, "\n\n\r");
        //RS485send(msg);
       //itoa(tagNum,10, msg);
       // RS485send(msg); // send tag id
         if ( customerCode == code && tagNum == tag_ID ){
           output_high(RED_LED);
           output_low(GREEN_LED);
         }else{
           output_high(GREEN_LED);
           output_low(RED_LED);
         }
        delay_ms(2000);
        output_high(GREEN_LED);
        output_high(RED_LED);
      }
    }
}
```

Anexa 2 Scheme electrice și desene de amplasare

Schemele de amplasare a componentelor pe cele două plăci




ICD CONNECTOR
4.7K
+12V
7805T
IN
GND
OUT
+5V
47uf 25V
.1
4.7uf
MAX232D
C1+
C1-
C2+
C2-
VCC
V-
GND
V+
T2OUT
R2IN
R1OUT
T1IN
T2IN
R1IN
R2OUT
9 Pin D Type
PIC16F628A-I/SO
20mhz Car Res
10K
RS485-FT
120
RED
YELLOW
GREEN
10
.0056
.0096
100pf
.001
EM4095
.022
.01
PIC16F876
330
20 MHz
15pf
DB9MDSS-4

5 Controloare logice programabile (PLC – Programmable Logic Controller)

5.1 Obiectivul lucrării

Lucrarea își propune studierea modului de implementare a unor aplicații de monitorizare și control cu ajutorul controloarelor logice programabile. Se studiază configurația hardware a unui astfel de dispozitiv și modul de conectare a unor elemente externe de tip senzori și elemente de execuție. Se prezintă modul de programare a unei aplicații utilizând un limbaj de programare specific.

5.2 Considerații teoretice

Controloarele logice programabile, denumite în jargonul curent PLC-uri, sunt dispozitive robuste și compacte special concepute pentru monitorizarea și controlul unor parametri de proces. Cu ajutorul lor pot fi implementate diferite scheme de automatizare de complexitate medie. Inițial ele au fost concepute pentru implementarea unor funcții de control binar: funcții logice combinaționale și automate programabile (control secvențial). Ulterior s-au adăugat funcții suplimentare de reglaj continuu și adaptiv, funcții de comunicație și funcții de vizualizare și stocare a datelor culese. Astăzi, prin caracterul lor robust, fiabil și autonom, reprezintă cele mai utilizate componente inteligente de automatizare.

Din punct de vedere constructiv, un dispozitiv PLC se compune din:

- microsistem de calcul, de obicei implementat cu ajutorul unui microcontroler
- set de interfețe digitale și analogice, care conțin circuite de adaptare pentru semnale industriale
- modul de alimentare electrică
- opțional: interfețe de comunicație serială (ex.: RS232, RS485) și în rețea (ex.: CAN, Profibus)
- carcasă de protecție

În cadrul lucrării de față, ca și exemplu de dispozitiv PLC, se va studia dispozitivul **FX3U** și componentele sale auxiliare, produse de firma MITSUBISHI. În figura 1 se poate observa imaginea acestui dispozitiv, pe care s-au marcat zonele de conectare/cuplare a unor elemente externe sau de extensie.

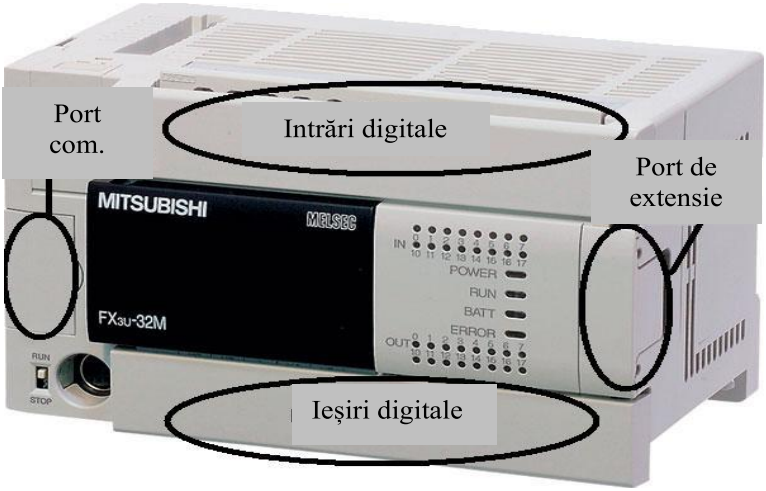


Figura 1. Exemplu de PLC – Mitsubishi FX3U

5.2.1 Interfețele digitale de intrare

Aceste interfețe permit conectarea unor semnale digitale generate de senzori, comutatoare sau întrerupătoare. Din punct de vedere al modului de conectare intrările digitale pot fi de două tipuri:

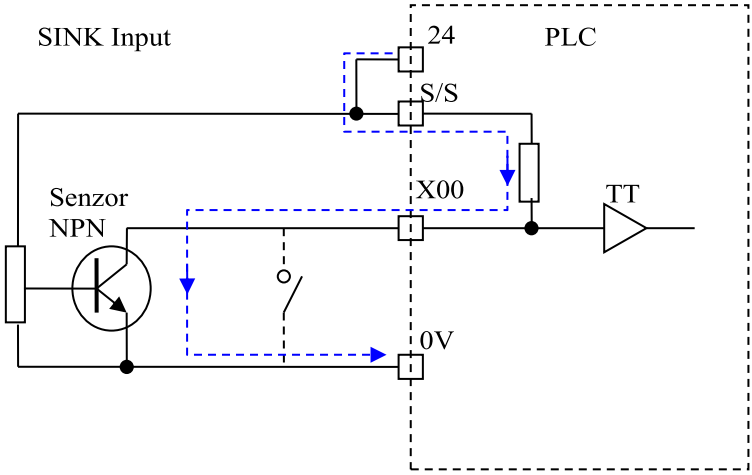


Figura 2. Schema de conectare a unui senzor NPN la o intrare de tip drenă

- drenă de curent (eng. Sink input), pentru senzori de tip NPN
- sursă de curent (eng. Source input), pentru senzori de tip PNP

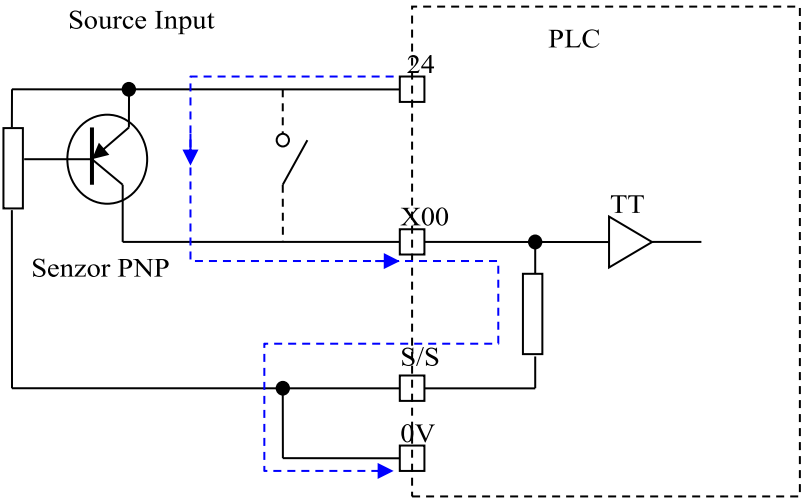


Figura 3. Schema de conectare a unui senzor PNP la o intrare de tip sursă

În cazul unei intrări de tip drenă, în starea "cuplat/închis" a senzorului, curentul circulă dinspre intrare către senzor. Figura 2 prezintă schema de principiu a unui senzor de tip NPN conectat la intrarea de tip Sink a unui PLC.

În cazul unei intrări de tip sursă, în starea "cuplat/închis" senzorul generează un curent ce intră în dispozitivul PLC. Figura 3 prezintă acest caz.

La PLC-ul FX3U intrările pot fi configurate în bloc ca sursă sau ca drenă. Toți senzorii trebuie să fie de același tip. Un simplu comutator poate fi atât senzor de tip sursă cât și senzor de tip drenă, deoarece curentul poate circula în ambele sensuri. La fel și unii senzori pot fi configurați să lucreze în unul din cele două moduri.

La un senzor activ, pe lângă cele două semnale care corespund terminalelor unui comutator, mai există un semnal suplimentar folosit pentru alimentarea senzorului. La cuplarea unui senzor la un PLC se va studia manualul de utilizare al senzorului și mai ales schema indicată pentru cuplarea acestuia la o intrare de tip sursă sau drenă.

În figura 4 s-a reprezentat schema de conectare a intrărilor dispozitivului PLC în cele două cazuri. Se observă că modul de cuplare a intrării S/S la 24V și respectiv la 0V determină configurația de drenă sau sursă a intrărilor. De asemenea se observă că în cazul unei intrări drenă un

comutator simplu se leagă între 0V și intrare, iar în cazul unei intrări de tip sursă între 24V și intrare.

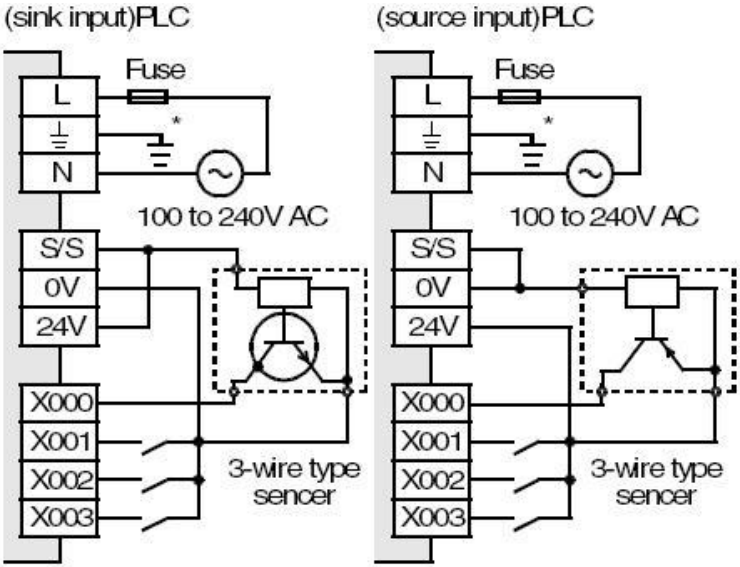


Figura 4. Schema de conectare a senzorilor la PLC

ATENȚIE: CUPLAREA GRESITĂ A SENZORILOR SAU A COMUTATOARELOR POATE DUCE LA ARDEREA CIRCUITELOR DE INTRARE

O intrare se consideră 1 logic dacă tensiunea de intrare este mai mare de 16V și 0 logic dacă este mai mică de 8V. Pe frontul urcător al semnalului de intrare comutarea se face la 16V iar pe frontul coborâtor la 8V.

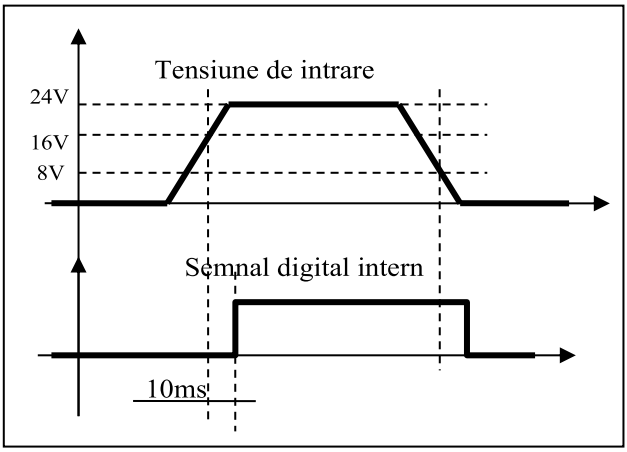


Figura 5. Diagrama de comutare a unei intrări

Din punct de vedere al programării intrările digitale sunt notate cu **X000-Xnnn.**

5.2.2 Interfețele digitale de ieșire

Cu ajutorul interfețelor de ieșire PLC-ul comandă diferite elemente de execuție digitale (bipoziționale). Ieșirile sunt de obicei izolate galvanic de partea de logică pentru a permite cuplarea unor tensiuni sau curenți de valoare mai mare și pentru a proteja partea de logică de

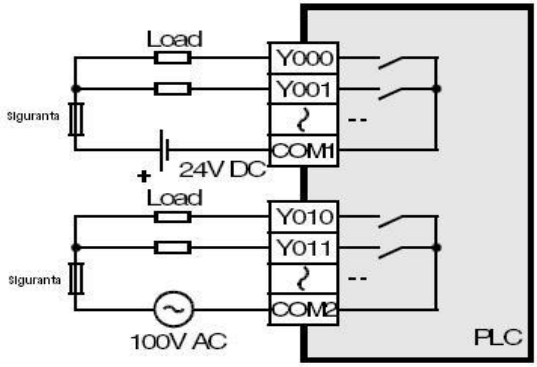


Figura 6. Modul de conectare a ieșirilor

eventuale tensiuni accidentale prea mari. În funcție de construcția PLC-ului ieșirile pot fi de tip contact de releu, tranzistor de putere sau tiristor. În cazul de față ieșirile sunt de tip contact de releu. Figura 6 prezintă modul de conectare a unui element de execuție sau sarcină (eng. Load) la o ieșire a PLC-ului.

Alimentarea elementului de execuție sau a sarcinii se va face de la o sursă externă de alimentare. Ieșirile se notează cu Y000-Ynnn. O ieșire are două contacte, din care unul se consideră că fiind semnalul de ieșire, iar al doilea contact este punctul comun (de masă sau de alimentare – COM).

ATENȚIE. A NU SE UTILIZA IESIRILE AUTOMATULUI ÎN CIRCUITE UNDE SE VEHICULEAZĂ CURENȚI MAI MARI DE 1-2A.

În momentul în care o ieșire este activă, pe panoul frontal al PLC-ului, LED-ul corespunzător acelei ieșiri se va aprinde. În mod similar un alt sir de LED-uri indică starea semnalelor de intrare.

5.2.3 Programarea PLC-ului

Standardul IEC 61131 stabilește 5 limbaje de programare utilizabile pentru programarea dispozitivelor de tip PLC:

- LD – Lader Diagram – limbaj grafic de tip "schemă cu relee"
- FBD – Function Block Diagram – limbaj grafic de tip "flux de date" (cu blocuri funcționale interconectate)
- ST – Structured Text – limbaj de nivel înalt asemănător cu C sau Pascal
- IL – Instruction List – limbaj de nivel scăzut de tip limbaj de asamblare
- SFC – Sequential Function Chart – limbaj care permite exprimarea secvențelor de pași pentru un automat de stare

Pentru dezvoltarea de aplicații se pot folosi diferite medii de programare, precum ISAGRAPH sau DX-Developer, care permit: editarea, compilarea, descărcarea pe un PLC țintă și execuția programului (în regim normal și în regim pas-cu-pas).

Pentru PLC-urile din familia Mitsubishi FX3U se va folosi mediul de programare Dx-Developer care pune la dispoziție 2 limbaje de programare: LD și SFC (bazat pe comenzi). În cadrul lucrării de față se folosește varianta LD.

Descrierea logicii programului se face similar cu modul de desenare a schemelor de automatizare cu relee. În schemă apar contacte de relee " --| |---", conexiuni "------" și ieșiri "—( )---". Suplimentar în schemele mai complexe pot să se utilizeze funcții predefinite. Principial, schema astfel construită se evaluează în timpul execuției programului ca și cum curentul circulă de la stânga la dreapta în mod paralel prin fiecare linie a schemei: acolo unde contactele pe o linie sunt închise, ieșirea este activă (contactul este închis). Practic schema se evaluează în trepte după cum se vede în figura 7.

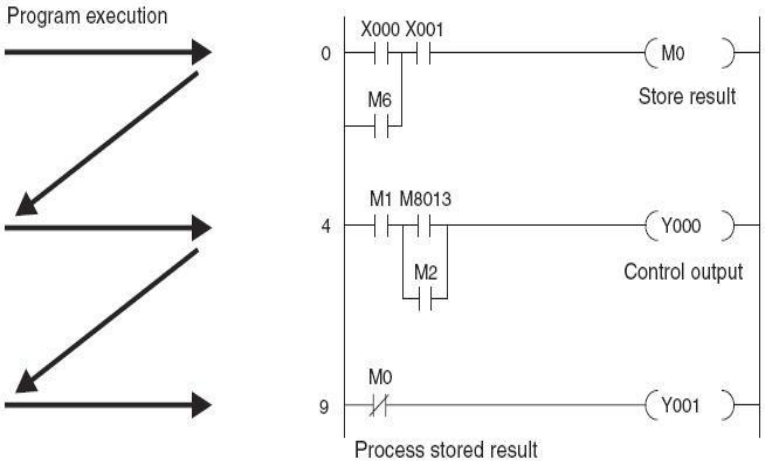


Figura 7 Modul de execuție a programului

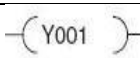
Execuția programului se face de sus în jos și de la stânga la dreapta. În cadrul liniilor de program se utilizează diferite tipuri de "dispozitive" incluse în structura PLC-ului, după cum urmează:

- **X – Intrări fizice** (ex.: X000-X007 la PLC-ul utilizat) – intrări digitale de tip contact
- **Y – Ieșiri fizice** (ex.: Y000-Y007 la PLC-ul utilizat) – ieșiri digitale de tip contact
- **M – Relee auxiliare** (ex.: M000-...) – dispozitive bistabile de tip releu folosite pentru memorarea temporara a unor stări
- **S – Relee de stare** – folosite în limbajul STL pentru a indica un pas (o stare) din diagrama de stare a automatului
- **T – timer** – dispozitive de măsurare a timpului tip "timer"; sunt de 3 feluri: cu increment la 100ms, 10ms sau 1ms; timerele sunt pe 16 biți; pot fi cu reținere (păstrează valoarea până se resetează) sau

fără reținere; ieșirea timerului este un contact care se închide în momentul în care se atinge valoarea prefixată a timerului; valoarea prefixată se specifică printr-o constantă (precedată de litera “K”, valoarea maximă 32767) sau un registru de date (precedat de literă ”D”)

- **C** – contoare – sunt similare cu timerele, doar că incrementarea nu este controlată de timp ci de impulsuri de semnal
- **D** – registre de date – sunt registre de 16 biți care se folosesc pentru păstrarea unor parametrii sau date de proces; se pot adresa la nivel de bit; de exemplu adresarea bitului 3 din registrul 0 se face astfel: D0.3
- **K, H, E** – constante numerice în zecimal, hexazecimal sau în formă exponențială

Simbolurile grafice cele mai des folosite sunt indicate în tabelul de mai jos:

	Contact normal deschis; X001 este numele semnalului care controlează releul
	Contact normal închis; M0 este numele semnalului care controlează releul
--| ↑ |--	Contact cu detecție de front urcător
--| ↓ |--	Contact cu detecție de front coborâtor
	Controlul bobinei unui releu; Y001 este controlul ieșirii
	Apel comandă

Folosind aceste simboluri de bază se pot forma instrucțiuni mai complexe gen:

ȘI logic

SAU logic

Cele două linii implementează următoarele funcții logice:

Y000 = X000 **ȘI** X001\ ȘI X002 și

Y000 = X000 **SAU** X001\ **SAU** X002

În continuare, pentru scrierea unui program destinat dispozitivului FX3U se folosește mediul de programare și depanare DX-Developer oferit de firma Mitsubishi. Mediul se instalează de pe CD-ul furnizat la achiziționarea dispozitivului PLC. Primul pas este crearea unui proiect nou prin secvența de comenzi *File -> New Project*. Se alege tipul dispozitivului pentru care se generează programul (PLC Type – FX3U(C)), tipul de limbaj folosit (în cazul nostru LD), numele proiectului și directorul unde se va salva proiectul (la sfârșit se apasă OK).

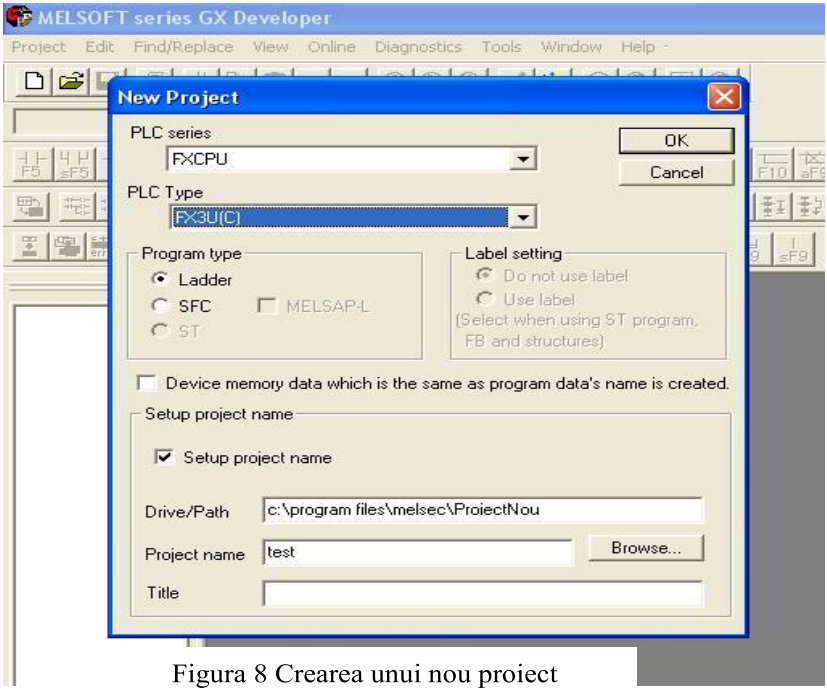


Figura 8 Crearea unui nou proiect

Editarea programului se poate face cu ajutorul simbolurilor grafice prezente în "toolbar"-ul mediului (Vezi figura de mai jos). Aceste simboluri reprezintă diferite elemente ale limbajului de programare. Simbolurile pot fi selectate și cu ajutorul tastelor funcționale F5-F10.

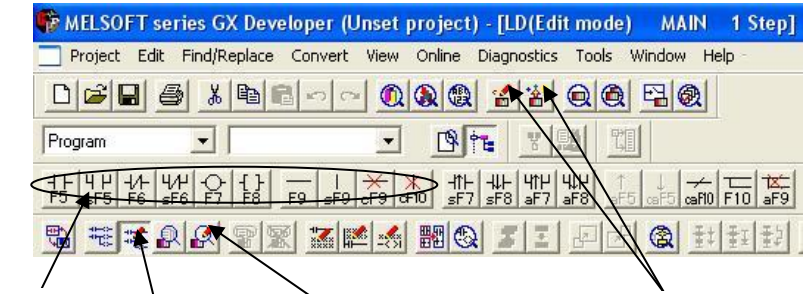


Figura 9 Meniul, icoanele și tastele funcționale ale mediului

Tasta F10 se va folosi pentru a crea ramificații. De asemenea se observă că după inserarea unui simbol pe o linie nou introdusă, la apăsarea Click dreapta vor apărea următoarele opțiuni:

- **Insert line** – va insera o linie nouă (numită rung) deasupra liniei curente
- **Delete line** – va șterge linia curentă
- **Insert row** – va insera o poziție nouă în fața simbolului curent
- **Delete row** – se va șterge simbolul curent

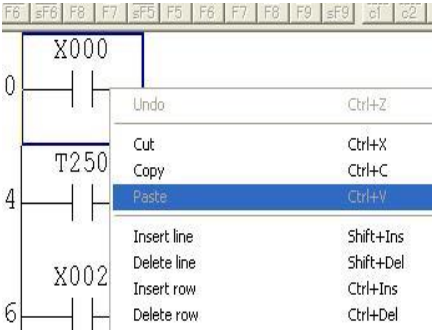
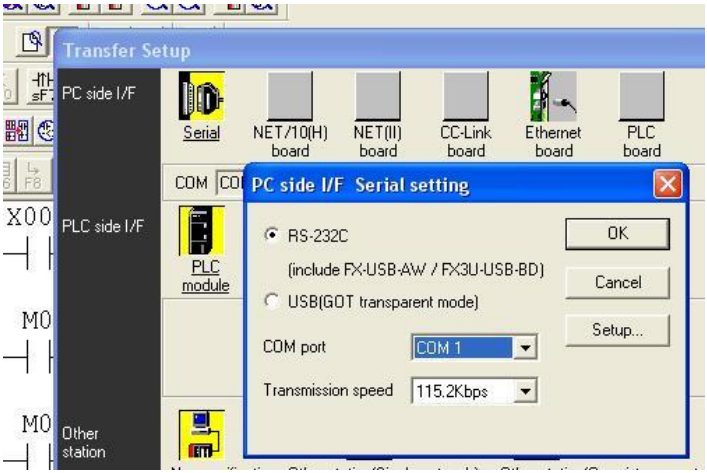


Figura 10. Opțiuni pentru linii

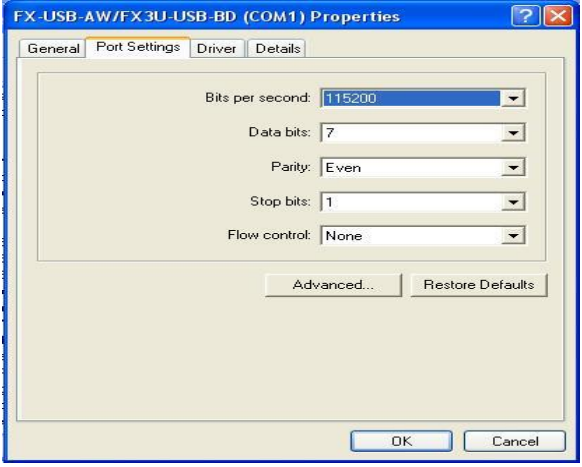
Pentru încărcarea și execuția programului trebuie să atașăm dispozitivului de bază PLC un modul de extensie pentru comunicație serială. Există două variante: un modul RS232 sau un modul de conversie USB-RS232.

a).Modul USB b)Modul RS232.

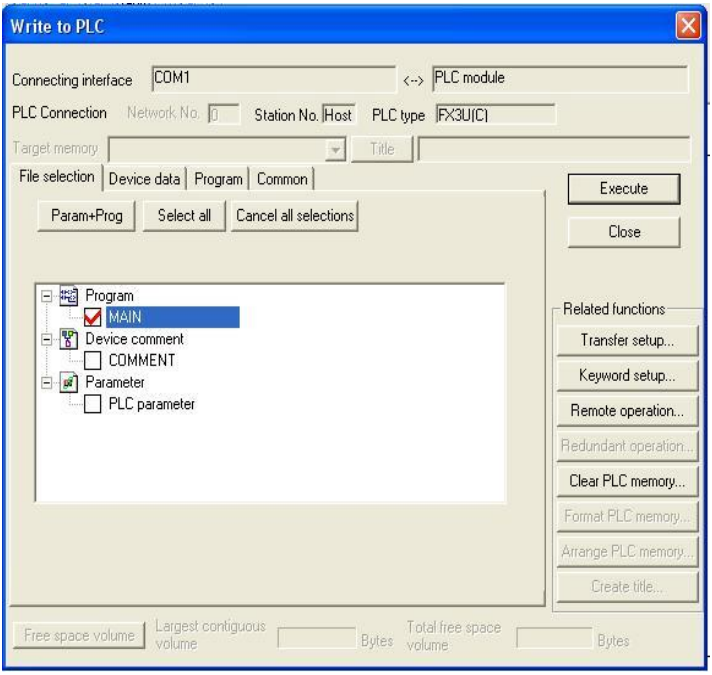
Pentru modulul a) se vor instala driverele existente în directorul curent cu numele KIT USB Bd. Setările portului serial se fac în modul următor. Se va intra în meniul *Online-> Transfer setup* și se va seta portul serial să fie același cu cel instalat pe stația de lucru (COM1, COM2, etc.).



De asemenea se va seta portul serial al stației prin *Control Panel->System-> Device Manager -> Ports (Com and LPT)* cu următoarele setări:

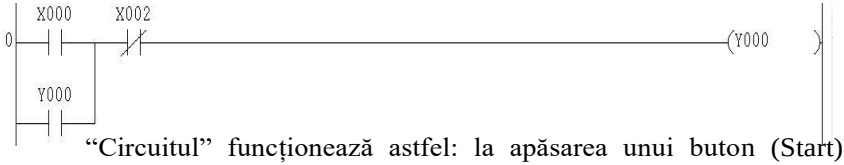


Când programul (LD) este complet se va converti într-un format transferabil prin secvența: *Convert -> Convert;* implicit se verifică și corectitudinea programului editat. După conversie programul este înscris în PLC prin selectarea icoanei: *Write PLC*. Se va bifa din fereastra apărută numai programul după care se va apăsa OK.



Acum sunteți pregătiți să începeți editarea primului program. În continuare urmează câteva exemple simple de programe:

1). Primul exemplu de program realizează un contact cu automenținere. (Menținerea în starea ON prin simpla apăsare și apoi eliberare a unui buton, sau activarea pentru un scurt moment a unui senzor). Aceasta secvență va fi folosită des mai departe.



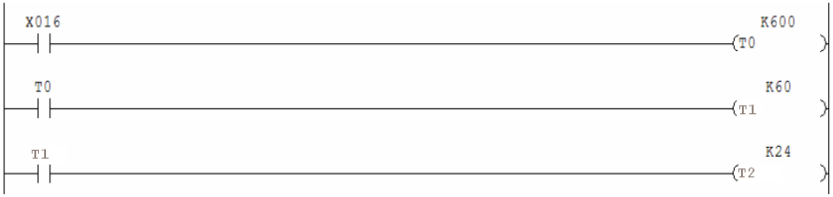
"Circuitul" funcționează astfel: la apăsarea unui buton (Start) conectat la intrarea X000 a PLC-ului ieșirea Y000 va fi activată (contactul Y000 închis). Se presupune că butonul (Stop) conectat la intrarea X002 nu este apăsat, ceea ce face ca prima linie a circuitului să fie închisă. În continuare, chiar și la eliberarea butonului Start ieșirea Y000 rămâne validă datorită ramurii paralele (funcție logică SAU) în care apare contactul Y000 închis; la apăsarea butonului Stop circuitul se întrerupe și ieșirea Y000 devine invalidă (contact deschis).

2). Următoarea aplicație va utiliza un timer pentru a obține un efect de temporizare. Timerele sunt de mai multe tipuri , diferind prin frecvența semnalului de ceas și prin modul de lucru (cu și fără reținere). Tabelul de mai jos indică adresele la care se află diferitele tipuri de timere.

Baza de timp	Adresă timer în PLC-ul FX3u
100ms	0-199
10ms	200-245
1ms (cu reținere)	246-249
100ms (cu reținere)	250-255
1 ms	256-511

Conform tabelului, timerul folosit în schema de mai sus este unul cu o perioadă de 100ms între 2 incrementări succesive. Atributul "cu reținere" denotă faptul că la inactivarea lui X000 timerul va păstra ultima valoarea până la care s-a incrementat, cât timp X000 era activ. Acest tip de timere are nevoie și de comandă de **Reset** care se execută în cazul de față prin activarea intrării X002. Conform schemei ieșirea Y000 se va activa după 4s (40*100ms) de la apăsarea butonului X000.

Printr-un calcul simplu se observă că durata maximă care poate fi obținută folosind un timer este de 32.767 x 0.1sec / 60 = 54.36 minute. Dacă în schimb e nevoie de mai mult decât atât, ținând cont că aplicațiile industriale pot avea nevoie de evenimente de verificare de exemplu odată la 12 sau 24 de ore, atunci se vor cascada mai multe timere obținând astfel durate mai lungi. În schema de mai jos s-au cascadat 3 timere: primul va număra minute, al doilea ore, iar al treilea zile.



Senzori de proximitate

Pentru controlul intrărilor PLC-ului se pot folosi diferite tipuri de senzori digitali de proximitate. În funcție de principiul fizic folosit distingem mai multe tipuri de senzori de proximitate:

- capacitivi – determină prezența unui obiect în perimetrul senzorului prin variația câmpului static creat în jurul senzorului; se detectează diferite tipuri de obiecte, indiferent de material (plastic, metal, sticlă, etc.); distanța de detecție este dependentă de dimensiunea obiectului, fiind în intervalul 0-2cm.
- inductivi – determină prezența unui obiect metalic în imediata apropiere a senzorului prin variația câmpului magnetic creat de senzor; distanța de detecție este mult mai mică (0-0,3 cm) și se detectează numai obiecte fero-magnetice
- optici – detectează trecerea unui obiect prin fața senzorului optic prin obturarea sursei de lumină (ex.: LED)
- mecanici – detectează un obiect care prin atingere cu senzorul închide un contact (microîntrerupătoare, limitatoare, etc.)

În lucrarea de față se folosesc senzori capacitivi (figura 11,a), inductivi (figura 11,b) și optici (figura 11,c) produși de firma Fotek

Figura 11. Senzori de proximitate.

Elemente de execuție.

Ca și elemente simple de execuție se pot folosi: electromagneți, electrovalve, servomotoare, motoare electrice de curent continuu sau de tip pas-cu-pas, dispozitive de avertizare sonoră (buzzer) și auditivă (vizuală) (LED-uri sau becuri), etc. Aceste elemente necesită o sursă de alimentare și o anumită schemă de acționare.

Pentru motoarele de curent continuu acționarea într-o singură direcție de rotație se poate face direct printr-o ieșire a PLC-ului. Ieșirea va juca rol de comutator într-o schemă în care mai apare motorul electric și sursa de alimentare. Cu scop de protecție, în paralel cu motorul se va conecta o diodă în sens invers cu direcția de curgere a curentului. Pentru acționarea motorului în două direcții de rotație este nevoie de un circuit de acționare în punte (denumit și circuit de tip H) și de 2 ieșiri ale PLC-ului (pornit/oprit și direcție). Electromagneții se pot folosi în mod similar cu acționarea într-o direcție.

Pentru acționarea buzzerelor sau a LED-urilor se poate folosi sursa internă de 24V a PLC-ului. Limitarea curentului se va face prin rezistențe calibrate la curentul nominal prin aceste dispozitive (ex.: 10mA în cazul LED-urilor).

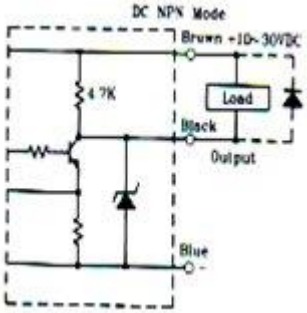
5.3 Mersul lucrării

1 Se instalează mediul de programare DX-Developer și se testează diferitele facilități ale mediului
2 Se editează mai multe programe folosind diverse tipuri de simboluri grafice, cu diferite funcționalități. Se va verifica corectitudinea programului scris prin funcția de conversie.
3 Să se conecteze senzori de diferite tipuri la intrările PLC-ului și să se controleze ieșirile pe baza unei logici prestabilite. Să se temporizeze activarea/dezactivarea ieșirilor cu ajutorul timerelor.

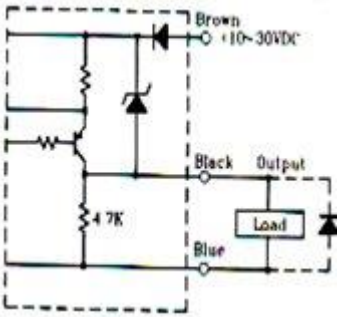
4 Să se realizeze un program care controlează o bandă transportoare pe care se deplasează piese. În momentul în care un senzor detectează existența unei piese pe bandă, PLC-ul va activa comanda unui braț de împingere (ex.: cu electromagnet) care va muta piesa pe o altă bandă transportoare; mișcarea va fi temporizată cu 2 secunde. Procesul începe la apăsarea unui buton simplu.

5 Să se realizeze un program care simulează pornirea a 8 benzi transportoarea legate la ieșirile Y000 – Y007. Pentru o economie de energie, benzile fiind foarte lungi iar motoarele consumând multă energie electrică, se cere pornirea celor 8 benzi transportoare la un interval de 5 secunde între fiecare 2 consecutive. După ce toate benzile au pornit, se așteaptă un interval de timp de 10 secunde (în realitate ar fi un regim de 8 ore de lucru) înainte ca acestea să fie oprite pornind de la ultima cu aceeași pauză de 5 secunde între ele.

Anexă:
Schema de conectare a senzorilor de proximitate din seria Fotek CP



Senzor de tip NPN



Senzor de tip PNP

6 Programarea dispozitivelor de tip PLC

6.1 Obiectivul lucrării

În cadrul acestei lucrări se vor aprofunda instrucțiunile de programare a dispozitivelor de tip PLC. Se vor prezenta mai multe categorii de instrucțiuni ale limbajului LD, cum ar fi instrucțiuni de setare/resetare variabile, transfer de date, generare de impulsuri, operații aritmetice și logice, transmisie și recepție serială, conversii de date, instrucțiuni de control al programului, etc.

6.2 Considerații teoretice

Limbajul LD oferă un set bogat de instrucțiuni, unele specifice pentru manipularea semnalelor (ex.: setare/resetare, conversie analog/digitală, etc.), altele cu caracter mai general, asemănătoare celor existente în alte limbaje de programare (instrucțiuni condiționate, de salt, etc.). O instrucțiune, sau mai corect o funcție se activează în cazul în care circuitul care o precedă s-a închis. Grafic funcția apare între paranteze drepte (un dreptunghi în standardul limbajului LD) și conține numele unei funcții/instrucțiuni și parametrii aferenți.

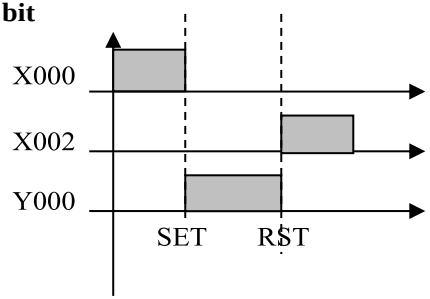
------[IFUNCTIE par1, par2, ...]----

În continuare se prezintă succint cele mai uzuale instrucțiuni, din fiecare categorie în parte.

6.2.1 Instrucțiuni comune

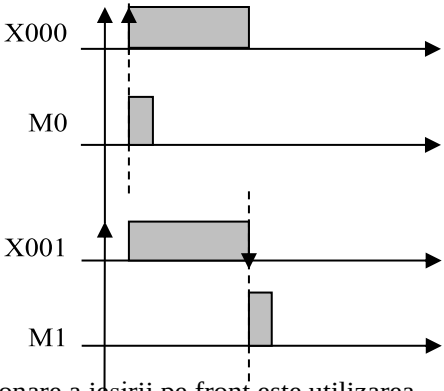
- **Setare SET și resetare RST bit**

```
    X000
|----| |-----[ SET Y000 ]-|
|    X002                 |
|----| |-----[ RST Y000]- |
```



- **Generare de impuls pe frontal urcător (PLS) și pe frontul coborâtor (PLF) al condiției** (se generează un impuls scurt care durează cât ciclul de parcurgere a diagramei)

```
    X000
|-----| |------[PLS M0]--|

    X001
|-----| |------[PLF M1]--|
```



- o metodă echivalentă de acționare a ieșirii pe front este utilizarea "releului" cu acționare pe front:

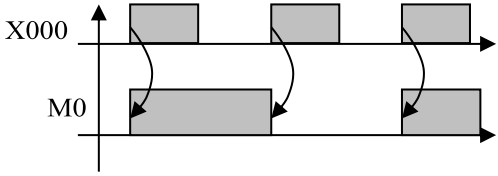
-----|↑|------ acționare pe front urcător

sau

-----|↓|------ acționare pe front coborâtor

- **Comutarea bitului între 0 și 1 la fiecare impuls al condiției (ALT)**

```
    X000
|-----| |------[ALTP  M0]---|
```



6.2.2 Instrucțiuni aplicative

- **Instrucțiuni de transfer**:

Transfer	16 biți	32 biți	flotant
1 la 1	MOV	DMOV	DEMOV

N la N	BMOV	-	-
1 la N	FMOV	DFMOV	-

X000
|----| |------[MOV D1 D2]--------|

- mută D1 în D2

|----| |------[BMOV D1 D7 K3]-|

- mută 3 cuvinte începând de la D1 la destinația D7,D8, D9

|----| |-----[FMOV D1 D7 K3]--|

- funcție "fill" de umplere; mută data din D1 în 3 câmpuri începând de la D7 (D7, d8, D9)

- **instrucțiuni de comparare**

Comparare	16 biți	32 biți	flotant
1 la 1	CMP	DCMP	DECMP
1 la interval	ZCP	DZCP	DEZCP

|----| | -------[CMP D0 K10 M0]-----|
- instrucțiunea setează M0 dacă D0>K10
Setează M1 daca D0=K10
Setează M2 dacă D0<K10

ZCP – comparare cu "zonă"

|----| | -------[CMP K10 K20 D0 M0]-----|
- instrucțiunea setează M0 dacă D0>K10
Setează M1 dacă K10<=D0<=K20
Setează M2 dacă D0<K10

- **comparații "inline"**

Comparație	16 biți	32 biți
Mai mare	>	D>
Egal	=	D=
Mai mic	<	D<
Mai mare sau egal	>=	D>=
Nu este egal	<>	D<>
Mai mic sau egal	<=	D<=

|--[= T10 K6]----------[Y000]-|
- Y000 este cuplat dacă condiția T10=K6 este îndeplinită

6.2.3 Instrucțiuni aritmetice

Operația	16 biți	32 biți	flotant
Adunare	ADD	DADD	DEADD
Scădere	SUB	DSUB	DESUB
Multiplicare	MUL	DMUL	DEMUL
Împărțire	DIV	DDIV	DEDIV
Rădăcina pătrată	SQR	DSQR	DESQR

X000
|----| |-----------------------[ADD K20 D0 D5] - când cond. X000 este îndeplinită atunci
D10=D0+20

6.2.4 Instrucțiuni de transmitere/recepție: TO/FROM

- sunt instrucțiuni care operează cu SFM-uri (Special Function Modules) – module cu funcții speciale atașabile la unitatea de bază PLC (ex.: modul de conversie analogic, modul de transmisie serială, etc.)
- în cadrul unui modul funcțional există o memorie tampon (BFM Buffer Function Memory) în care sunt locații cu funcții predefinite; de exemplu la modulul analogic FX2N-4DA BFM #0 păstrează "modul de ieșire" iar BFM #1 păstrează valoarea digitală pentru canalul 1.
- Manualul modulului special descrie funcția fiecărei locații din memoria tampon atașată

Instrucțiunea TO trimite o dată de la CPU la SFM. Formatul:

-[TO SFM BFM SRC NUMAR]-

Exemplu:

|----| |-------------------------[TO K2 K0 H1122 K1]-|

- scrie o dată (constanta hexa 1122) la BFM #0 în a 3-a unitate SFM întâlnită (K2); de exemplu dacă este o unitate FX2N-4DA atunci se setează canalul 1 și 2 la ieșire 0-20mA și canalul 3 și 4 la ieșire 4-20mA

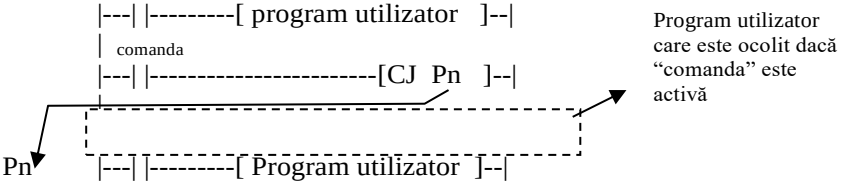
Instrucțiunea FROM mută o dată din SFM (dintr-o locație BFM) în memoria PLC-ului. Formatul este același cu instrucțiunea TO.

-[FROM SFM BFM DEST NUMAR]-

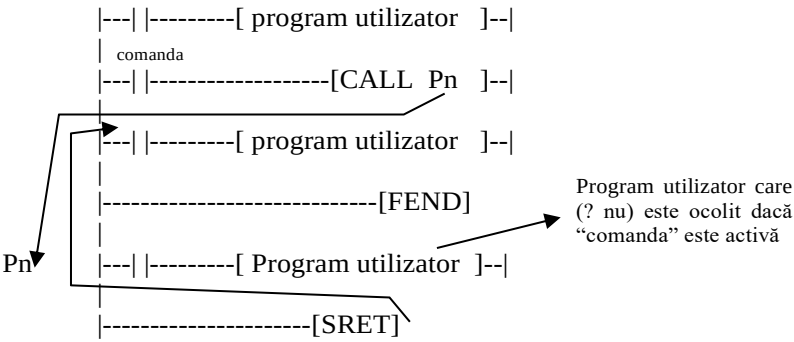
6.2.5 Instrucțiuni de control al programului

În această categorie intră instrucțiunile de salt condiționat (**CJ**), apel de subrutină (**CALL**), revenire din subrutină (**SRET**) și sfârșitul programului principal (**FEND**).

Instrucțiunea CJ – salt condiționat



Instrucțiunile CALL și SRET – salt la rutină și revenirea din rutină



6.2.6 Instrucțiuni ale ceasului de timp-real

PLC-ul are un ceas de timp-real incorporat care se poate utiliza pentru a măsura timpul scurs sau pentru a temporiza anumite acțiuni în funcție de ora curentă.

Informațiile de timp se păstrează în registrele D8013-D8019 după cum urmează:

D8013 – secunde
D8014 – minute
D8015 – ora (format 24 ore)
D8016 – ziua
D8017 – luna
D8018 – anul (2 cifre)
D8019 – ziua din săptămână (0 - duminică, 6 - sâmbătă)

Instrucțiunea TRD (time read) citește cele 7 registre ai ceasului într-o zonă de memorie indicate

Instrucțiunea TWR (time write) scrie cele 7 registre ai ceasului dintr-o zonă de memorie.

Instrucțiunea TCMP (time compare) compară ora, minutul și secunda cu conținutul a 3 registre consecutive.

6.3 Mersul lucrării

1. Se vor scrie programe care încorporează instrucțiunile prezentate în lucrare.
2. Să se scrie exemple de program care activează anumite elemente de acționare în funcție de timp (ex.: achiziție periodică de date).
3. Să se scrie un program cu mai multe regimuri de lucru controlate cu ajutorul timpului și al instrucțiunilor de salt.
4. Să se analizeze asemănările și deosebirile dintre limbajul LD și un limbaj de programare obișnuit (ex.: C); să se analizeze în mod special aspectele de execuție în regim concurent.

7 Achiziția și transmiterea datelor de proces cu ajutorul dispozitivelor de tip PLC

7.1 Obiectivul lucrării

Lucrarea prezintă tehnicile de configurare și programare a dispozitivelor de tip PLC în vederea achiziționării și transmiterii la distanță a datelor. Se folosesc funcții avansate ale limbajului LD pentru citirea interfeței analogice, pentru afișarea rezultatelor și pentru transmiterea datelor citite pe un canal serial.

7.2 Descriere tehnică

Cu ajutorul unui dispozitiv de tip PLC pot fi achiziționate și generate semnale digitale și analogice. Prin intermediul acestor semnale se realizează schimbul de informații dintre procesul controlat și dispozitivul de control (PLC-ul). Programul înscris în PLC prelucrează semnalele de intrare (achiziționate) și pe baza lor generează semnale de ieșire în conformitate cu obiectivul urmărit. În cazul în care PLC-ul este parte a unui sistem de control mai complex informațiile culese trebuie transmise sistemului ierarhic superior (ex.: calculator de proces) iar acesta poate transmite PLC-ului anumiți parametri de configurare (ex.: valoare prescrisă, regim de lucru, etc.).

În cazul PLC-ului FX3U studiat în laboratoarele anterioare unitatea de bază gestionează 8 semnale digitale de intrare și 8 semnale digitale de ieșire; conectarea acestora se face la cele două șiruri de cleme ale PLC-ului. Unitatea de bază se poate extinde cu module suplimentare care se atașează de o parte sau alta a PLC-ului. Aceste module permit:

- extinderea numărului de semnale digitale gestionate
- achiziția și generarea de semnale analogice
- comunicația cu alte echipamente de calcul pe un canal serial
- afișarea variabilelor interne și a stării PLC-ului pe un display de tip LCD

În cadrul prezentului laborator vor fi utilizate următoarele module suplimentare:

- FX_{ON}-3A – modul analogic pentru 2 semnale analogice de intrare și un semnal analogic de ieșire

- FX_{3U}-7DM – modul de afișare LCD
- FX_{3U}-USB- BD – modul de comunicație serială USB/RS232

Programarea și accesul la aceste module suplimentare se realizează prin instrucțiuni de tip TO/FROM (pentru modulul analogic) și respectiv instrucțiuni de tip RS (pentru modulul de comunicație).

7.2.1 Modulul FX_{ON}-3A

Acest modul conține 2 canale analogice de intrare și un canal analogic de ieșire. Semnalele pot fi de mai multe tipuri:

- de tensiune, în intervalul 0-10V sau 0-5V (ajustabil)
- de curent, în intervalul 4-20 mA

Conversia semnalelor de intrare și de ieșire se face pe 8 biți, plaja de variație digitală fiind de 0-255. Reglajul de amplificare și de offset se face astfel ca plaja utilă de variație (ex.: 0-10V) să corespundă pentru valori digitale în intervalul 0-250.

Impedanța de intrare pentru intrarea de tensiune este de 200kΩ și de 250Ω pentru intrarea de curent. Sarcina admisă la ieșirea de tensiune este de 1-1000KΩ și 0-500Ω la ieșirea de curent.

Programarea, citirea și scrierea canalelor analogice se face prin intermediul memoriei tampon BFM (Buffer Function Memory), după cum urmează:

BFM nr.	**b15-b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
0	rezervat	Valoarea curentă citită de la canalul de intrare selectat prin BFM17.0							
16	rezervat	Valoarea curentă scrisă în canalul de ieșire							
17	rezervat						D/A start	A/D start	Canal A/D
1-5, 18-31	rezervat								

Semnificația biților din BFM 17:

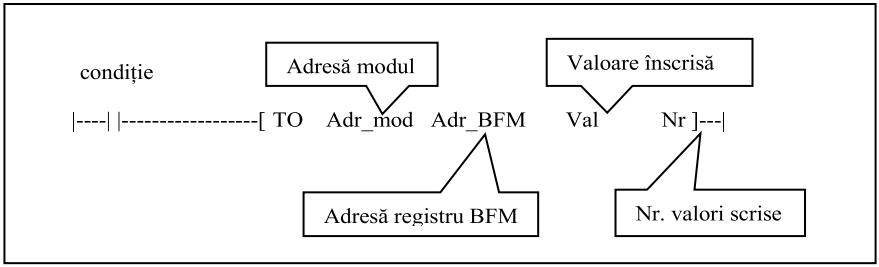
B0 – selectează canalul analogic de intrare (0 sau 1)

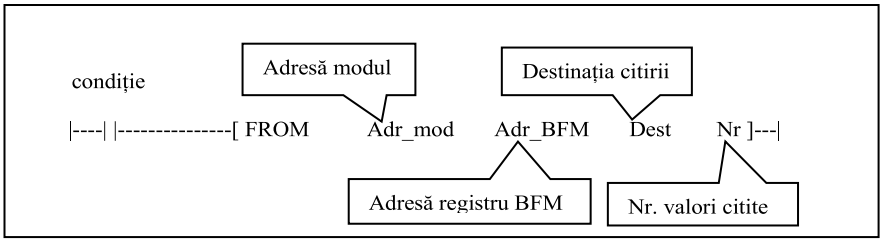
B1 – start conversie canal A/D pe frontul urcător (tranziția 0=>1)

B2 - start conversie canal D/A pe frontul coborâtor (tranziția 1=>0)

Programarea modulului:

Formatul instrucțiunilor TO (scriere modul) și FROM (citire modul)





Exemplul 1: Citirea celor 2 canale analogice de intrare:

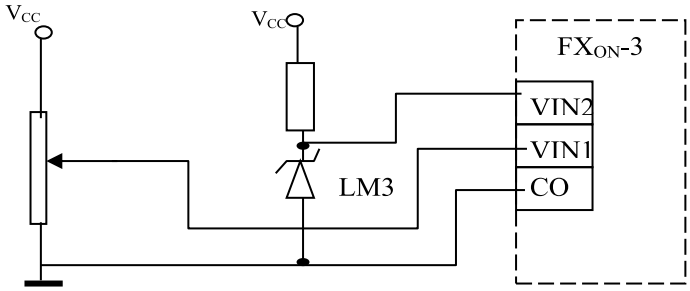
```
          M0
|------| |-------[ TO  K0  K17  H00  K1]--|
|          |-----[ TO  K0  K17  H02  K1]--| - start conversie A/D canal 0
|          |-----[From K0  K0  D00  K1]--|   - citire în D00 a valorii convertite
|     M1
|------| |-------[ TO  K0  K17  H01  K1]--|
           |-----[ TO  K0  K17  H03  K1]--|     - start conversie A/D canal 1
           |-----[From K0  K0  D01  K1]--|      - citire în D01 a valorii convertite
```

Exemplul 2: Scrierea unei valori în canalul analogic de ieșire

```
          M0
|------| |-------[ TO   K0  K16  D02  K1]--|    - scrierea valorii din D02 în
canalul D/A
|          |-----[ TO   K0  K17  H04  K1]--|
|          |-----[From K0  K17  H00  K1]--|      - start conversie D/A
```

Pentru verificarea funcționării canalului analogic de intrare se vor folosi:

- un divizor de potențial realizat cu un potențiometru
- un senzor de temperatură integrat (ex.: LM335)



Pe canalul analogic de ieșire se va înscrie valoarea citită de la unul din canalele de intrare. Cu ajutorul potențiometrelor de pe interfață se vor face ajustările necesare (amplificare și offset) astfel încât valoarea tensiunii de intrare să fie egala cu valoarea tensiunii de ieșire.

7.2.2 Modulul de comunicație seriala FX_{3U}-USB- BD

Acest modul poate fi folosit atât pentru programarea PLC-ului cât și pentru transmisia de date între calculatorul personal și PLC. Pentru a putea fi utilizat modulul trebuie configurat cu parametrii de comunicație. Configurarea se poate face din program (prin instrucțiuni ce scriu registrele interfeței seriale) sau prin mediul GX Developer, prin setarea parametrilor atașați proiectului. În al doilea caz se selectează (dublu click) din proiect *"Parameters-> PLC parameters-> PLC System(2)-> ch1*; se bifează "*Operate comm. Settings*" și apoi se configurează parametrii canalului serial.

La încărcarea proiectului în PLC (meniul "*online->Write PLC*") se va bifa pe lângă "*Main*" și căsuța "*parameters*".

Pentru programarea canalului serial se folosesc "dispozitivele" de 1 bit din tabelul de mai jos:

Dispozitiv	Nume	Operații permise
M8000	PLC pornit	read
M8063	Eroare de comunicație pe canalul ch1	read
M8122	Cerere de trimitere; când este pus pe ON PLC-ul începe transmisia	read/write
M8123	Sfârșit recepție; indică terminarea operației	read/write

	de recepție	
M8161	Pt. ON mod de lucru pe 8 biți, pentru OFF mod de lucru pe 16 biți	write

Programarea canalului serial se va face cu ajutorul instrucțiunilor "RS". La un moment dat o singură instrucțiune RS trebuie să fie activă. Formatul instrucțiunii:

```
          condiție
|-----| |--------[RS  Sursă   m  Dest.  n   ]--|
```

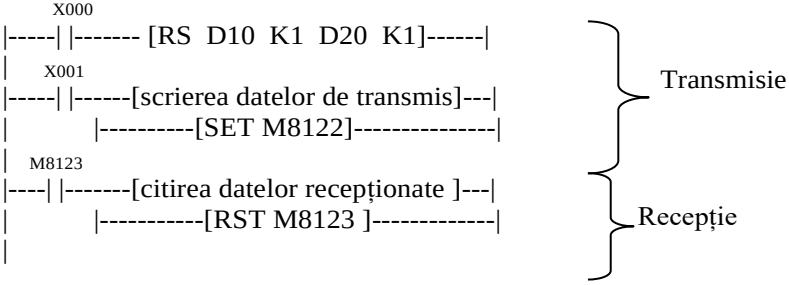
Unde: Sursă – adresa primului registru folosit ca sursă de transmisie (ex.: D001)

m – numărul de octeți transmiși începând de la adresa sursei (ex.: pt. m=3, D001, D002, D003)

Dest. – adresa primului registru destinație (ex.: D005)

n – numărul de octeți recepționați (ex.: pt. n=2 D005, D006)

Exemplu de secvență de transmisie și recepție a datelor:



Exemplu concret de transmisie și recepție a unui octet:

```
  M800
|-----| |-----------------(M8161 )-----|
|           |------[RS D1 K2 D4 K1]-|
|   X004                            |
|-----| |-------------------[PLS M0]---|
    M000
|-----| |---------------------[MOV H55 D1]---|
|             |---------------[MOV H56 D2]---|
```

```
|            |---------------[SET M8122]------|
|  M8123                                      |
|-----| |--------------------[MOV D4 D6]---|
|            |---------------[RST M8123]----|
|--------------------------------[END}--------|
```

7.2.3 Modulul de vizualizare a stării PLC-ului- FX_{3U}-7DM

Acest modul permite vizualizarea dispozitivelor incluse în PLC. Lucrează pe bază de meniu controlabil prin cele 4 butoane de pe panou :

- OK – buton de selecție a meniului
- "+" și "-" – butoane de navigare în meniu
- ESC – buton de ieșire din meniu

Pentru aplicațiile dezvoltate în cadrul laboratorului se va folosi funcția de monitorizare a dispozitivelor D, M, T, X și Y.

De exemplu pentru vizualizarea dispozitivelor D se alege: *"OK"->Monitor/Test->OK->D(16)->OK->* + sau – pentru navigare între dispozitivele D000- Dn.

7.3 Mersul lucrării

1. Se studiază modul de programare a modulelor de extensie prezentate în lucrare.
2. Se atașează la unitatea de bază FX3U modulele de extensie pentru interfața analogică, pentru afișaj și pentru canalul serial.
3. Se va scrie un program care efectuează următoarele operații:
    a. citește o valoare de tensiune de la un canal analogic,
    b. citește starea intrărilor X000-X003,
    c. transmite datele citite la calculator
    d. recepționează 2 octeți de la calculator
    e. primul octet se înscrie în canalul analogic de ieșire
    f. al 2-lea octet determină starea ieșirilor Y000-Y007.
4. Se va scrie o aplicație care efectuează un reglaj bipozițional de temperatură. Pe o intrare analogică se va conecta senzorul de temperatură iar pe o ieșire digitală se va cupla și decupla un element de încălzire (ex.: o rezistență de putere). Valoarea de temperatură citită se va transmite la PC.

8 Transmisia datelor într-un sistem de control prin rețea industrială

8.1 Obiectivul lucrării

Lucrarea își propune să prezinte metodologia de transmitere a datelor pe baza unui protocol de comunicație destinat aplicațiilor de control. În mod concret se prezintă o rețea de senzori inteligenți conectați într-o rețea de tip ModBus.

8.2 Considerații teoretice

8.2.1 Rețele industriale de comunicație

În aplicațiile de control informațiile sunt transmise în diferite moduri:

- prin conexiuni (leaturi) dedicate între sistemul de control (calculator) și dispozitivele de automatizare (senzori, elemente de execuție) plasate în procesul controlat
- prin rețea industrială de comunicație ce leagă toate elementele unui sistem de automatizare.

Rețelele industriale sunt medii de comunicație adaptate pentru cerințele specifice ale aplicațiilor de control. Dintre cele mai importante cerințe satisfăcute de aceste rețele se pot aminti:

- funcționare deterministă și predictibilă
- timp de transmisie a datelor relativ scurt și mai ales garantat
- fiabilitate sporită și toleranță la defecte
- imunitate la zgomote
- protocol optimizat pentru mesaje scurte și periodice.

Comunicația în rețea prezintă o serie de avantaje în comparație cu transmisia prin legături dedicate: costuri de cablare mai mici, o singură interfață pe dispozitiv, există mecanisme de detecție și de corecție a erorilor de transmisie, datele transmise pot fi mai complexe și transmisia se poate face la distanțe mai mari. Dar echipamentele conectate în rețea trebuie să dispună de o "inteligență" minimă pentru a putea implementa protocolul de comunicație. De obicei la dispozitivele simple de automatizare (senzori, elemente de acționare) se atașează un microcontroler care prin programul incorporat asigură administrarea dispozitivului și implementează funcțiile specificate de protocol.

8.2.2 Protocolul ModBus implementat pe RS485

Ca și exemplu de rețea industrială în cadrul prezentului laborator se va utiliza protocolul ModBus (http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf). Acest protocol se bazează pe schimbul de mesaje între dispozitivele conectate în rețea. Protocolul specifică formatul mesajelor de interogare și de răspuns. Din acest punct de vedere poate fi catalogat ca și un protocol de nivel 2 ("Legătura de date"). Datorita simplității sale, protocolul a fost adoptat de mai multe firme producătoare de echipamente de automatizare.

Ca suport fizic ("Nivel fizic") se poate folosi un canal serial de comunicație, cum ar fi de exemplu RS485. Standardul RS485 permite transmiterea de caractere în regim asincron pe o pereche de fire torsadate. Informația binară este codificată prin diferența de potențial pozitivă sau negativă dintre cele două fire de transmisie. Pe același tronson pot fi conectate în paralel până la 16 unități de transmisie/recepție. Protocolul impune ca la un moment dat o singură unitate să transmită restul unităților fiind în regim de ascultare (cu circuitul de transmisie în "înaltă impedanță").

Într-o rețea de tip ModBus există o unitate master care inițiază o comunicație și mai multe unități slave (max. 247) care primesc date și răspund la interogări. Unitatea master stabilește ordinea de comunicare în rețea și frecvența de transmisie a datelor. Formatul general al unui mesaj ModBus este dat în figura de mai jos:

Adresa dispozitiv	Codul funcției	Bloc de date	CRC

Sunt definite mai multe tipuri de funcții (mesaje), după cum urmează:

Cod funcție	Descriere
01	Citire stare relee (coil status)
02	Citire stare de intrare
03	Citire registre de memorare (holding registers)
04	Citire registre de intrare
05	Forțarea unui releu (single coil)

06	Setarea unui singur registru
07	Citirea stării de excepție
15	Forțarea mai multor relee (multiple coil)
16	Setarea mai multor registre
17	Raportează ID slave

În continuare vor fi detaliate funcțiile de Citire registru de memorare (cod 03) și Setarea unui singur registru (cod 06), care se vor utiliza în cadrul aplicației practice. Structura acestor mesaje precum și a răspunsurilor este dată mai jos:

Citire registru de memorare (Cod 03):

Adr. disp. slave	Cod funcție (03)	Adr. reg High	Adr. reg Low	Nr. reg. High	Nr. reg. Low	CRC

Răspunsul la funcția 03

Adr. disp. slave	Cod funcție 03	Nr. octeți return	Adr. start Low	Reg. 1 High	Reg. 1. Low		CRC

Setarea unui registru (Cod 06):

Adr. disp. slave	Cod funcție (06)	Adr. reg High	Adr. reg Low	Nr. reg. High	Nr. reg. Low	CRC

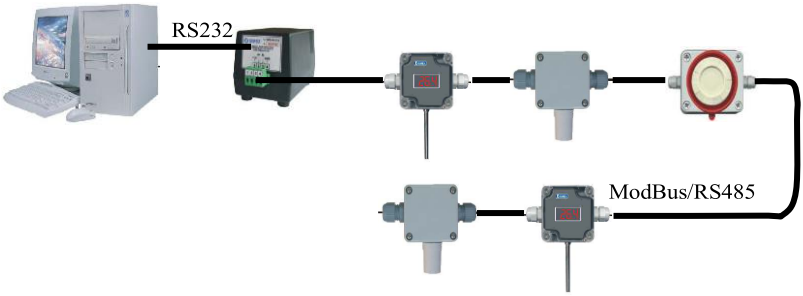
Răspunsul la funcția 06

Adr. disp. slave	Cod funcție 06	Adr. reg High	Adr. reg Low	Reg. Low	Reg. High	CRC

8.2.3 Prezentarea sistemului experimental

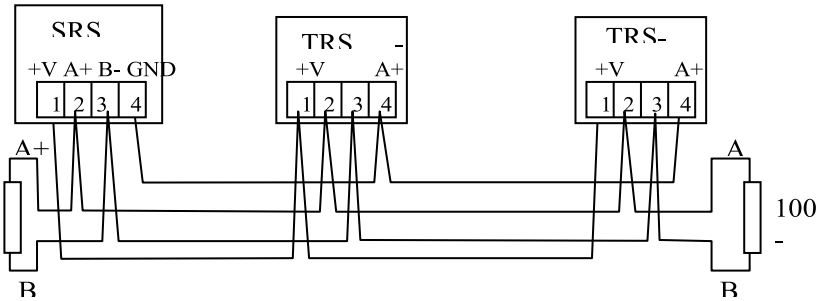
În cadrul lucrării se va utiliza sistemul de achiziție a informațiilor de temperatură și umiditate TRS produs de firma SIMEX Measurement and

Control. O descriere mai detaliată a acestui sistem se găsește în anexa “TRS-prezentare.pdf. Figura de mai jos prezintă principalele componente ale acestui sistem:



- calculator personal pe care rulează aplicația de culegere și vizualizare a datelor Piggy System
- modul (master) de conversie RS232<=>RS485 și stocare temporară a datelor; modulul asigură achiziția de date și în lipsa calculatorului personal; în prezența calculatorului modulul joacă doar rol de convertor RS232<=>RS485 – SRS-2/4-z16-B1a
- modul (slave) pentru măsurarea temperaturii – TRS-01a
- modul (slave) combinat pentru măsurarea temperaturii și a umidității – TRS-04a
- modul (slave) pentru măsurarea și afișarea temperaturii – TRS-11a
- modul (slave) pentru semnalizarea sonoră și vizuală a unei avarii – TRS-B1a
- modul (slave) pentru afișare
- modul pentru alimentare suplimentară

Conectarea dispozitivelor într-o rețea ModBus/RS485 se va realiza conform schemei de mai jos:



Pentru punerea în funcțiune a fiecărui modul în parte și a sistemului în ansamblu se va studia documentația existentă pe CD-ul de instalare a sistemului.

În tabelul de mai jos se prezintă un exemplu de organizare a registrelor interne ale modulului de măsurare și afișare a temperaturii TRS-11a; pentru citirea și scrierea celorlalte module se va consulta manualul de utilizare al acestora.

Nr. reg.	Scriere permisă	Interval	Descriere registru
01h	NU	-40 - 850	Temperatura curentă (exprimată în 0,1°C)
02h	NU	0, 20h, 40h, 80h	Starea măsurării curente (coduri de eroare)
03h	NU	1	Poziția punctului zecimal 1=0,0
04h	DA	0-5	Rata de filtrare a semnalului (0 – fără filtrare, 1 – filtrare slabă, 5- filtrare puternică)
05h	-		Registru de calibrare; A NU SE SCHIMBA
20h	DA	0-ffh	Adresă dispozitiv (din fabricație vine setat cu 0feh)
21h	NU	006ch	
0fff0h 0fff1h	NU		Număr Serie unic
0fff2h	NU	006ch	Cod de identificare dispozitiv (Device ID)
0fff3h	NU		Versiune firmware
0fff4h	NU		Număr construcție

De exemplu pentru citirea valorii curente a temperaturii se transmite următorul mesaj de interogare:

Adresa	Funcția	Nr. registru H,L		Număr registre		CRC	
01	03	00	01	00	01	D5	CA

Răspunsul modulului va fi:

Adresa	Funcția	Nr. octeți	Data H,L		CRC	
01	03	02	00	ff	F8	04

Valoarea temperaturii citite este 25,5°C (0ffh=255*0,1°C).

În caz de eroare răspunsul va fi de forma:

Adresa	Funcția	Eroare	CRC L,H	
01	83	40	40	C0

Eroarea 40h – limita de jos a domeniului de măsura este depășită.

Pentru fiecare mesaj transmis trebuie să se calculeze CRC-ul (cod ciclic redondant). La recepție CRC-ul se recalculează și se compară cu valoarea transmisă. În acest mod se verifică corectitudinea datelor transmise. Pentru calcularea "manuală" a codului CRC se poate folosi următorul site: http://www.lammertbies.nl/comm/info/crc-calculation.html.

8.3 Mersul lucrării

1. Se va studia documentația sistemului TRS.
2. Se vor conecta modulele pe un tronson ModBus conform descrierii din documentație.
3. Se va instala aplicația Piggy Soft Buzzer.
4. Se vor inițializa pe rând modulele slave prin acționarea butonului din interior; în acest mod se stabilește adresa fiecărui dispozitiv; programul Piggy Soft va identifica fiecare dispozitiv în parte
5. Cu ajutorul meniului de configurare se vor configura senzorii și Buzzerul; parola pentru configurare este: "Piggy System". Se stabilește frecvența de achiziție a datelor, limitele normale de măsură și condițiile de generare a alarmelor.
6. Se vor asocia mai mulți senzori în câte un grup logic.
7. Se vizualizează graficul evoluției mărimilor măsurate.
8. Se vor seta condiții de alarmă și se va acționa buzzerul în cazul apariției unei alarme.

9. Pentru controlul direct al rețelei (fără aplicația Piggy Soft) se va folosi un utilitar de comunicare serială (ex.: cel folosit la mediul PIC-C) și se vor transmite mesaje ModBus către fiecare modul în parte și se va analiza răspunsul primit.
10. Se va scrie o aplicație care interoghează dispozitivele conectate în sistem și afișează valorile citite. Atenție, pentru fiecare mesaj trebuie să se calculeze codul CRC. Modul de calcul a CRC-ului pentru protocolul ModBus este indicat în specificația standardului de protocol (a se vedea pe Internet).

9 Rețele senzoriale fără fir

9.1 Obiectivul lucrării

Lucrarea își propune să prezinte principiile de comunicație în rețele senzoriale fără fir și exemplificarea acestora prin două implementări practice.

9.2 Considerații teoretice

9.2.1 Rețele senzoriale fără fir

Rețelele senzoriale fără fir au cunoscut în ultima perioadă o dezvoltare semnificativă. Au apărut atât tehnici și protocoale noi de comunicație cât și implementări fizice de noduri de rețea bazate în principal pe microcontroloare și procesoare de semnale mixte (eng. MSP – Mixed Signal Processors).

În cazul acestor tipuri de rețele cele mai importante obiective de proiectare sunt:

- configurarea automată a rețelei – stabilirea unor rute optime de la nodurile rețelei la un punct de concentrare a datelor
- asigurarea conectivității între noduri în prezența unor defecte – stabilirea de noi trasee de comunicație în cazul în care anumite noduri devin inactive sau se defectează
- asigurarea unui consum de energie minim la nivelul nodurilor de rețea – pentru a asigura o durată de viață cât mai lungă pentru bateriile cu care se alimentează nodurile; se caută un optim între perioada de transmisie a datelor (transmisia fiind consumatoare de energie) și consumul nodurilor
- structurarea și fuziunea datelor culese de către senzori (eng. sensor fusion) – construirea unei imagini coerente asupra stării mediului sau procesului controlat prin suprapunerea și agregarea datelor transmise de fiecare nod
- stabilirea unui optim între puterea de transmisie a unui nod și distanța de acoperire – reglarea puterii de transmisie în funcție de distanțele dintre noduri și având în vedere reducerea consumului.

Un nod de rețea se compune dintr-un circuit inteligent (microcontroler, procesor de semnal sau procesor de semnale mixte), un sau mai mulți senzori și un circuit specializat de transmisie și recepție radio. În circuitul inteligent se înscrie un program (eng. firmware) care implementează protocolul de comunicație în rețea și procedura de achiziție, filtrare și stocare temporară a

datelor. Pentru noduri mai complexe funcțiile de comunicație și cele legate de achiziția datelor sunt implementate separat în 2 circuite de tip microprocesor.

Senzorii de la nivelul unui nod generează fie semnale digitale, fie semnale analogice. Aceste semnale sunt preluate de interfața paralelă și respectiv de interfața analogică a microcontrolorului. Uneori sunt necesare circuite de adaptare pentru amplificarea, filtrarea și eșantionarea semnalelor generate de senzori.

Transmisia datelor se face cu ajutorul undelor radio. Se folosesc de obicei anumite benzi frecvență destinate pentru uz industrial, științific și medical (benzi ISM – industrial, scientific and medical), pentru care nu trebuie să se folosească o licență de transmisie. Frecvențele de transmisie variază de la 6,78MHz până la 254GHz. Pentru comunicația în rețea se folosesc cu precădere frecvențele: 2450MHz (Bluetooth), 5800MHz (HIPERLAN) 2450 MHz și 5800 MHz (standardul IEEE 802.11). Datorită lipsei de restricții privind utilizarea acestor frecvențe proiectanții trebuie să ia în calcul posibilitatea apariției unor interferențe cu alte echipamente care comunică pe aceeași frecvență.

În cadrul lucrării se folosesc două tipuri de circuite de transmisie/recepție (transceiver) radio: CC2500 și CC2480, produse de firma Texas Instruments. Circuitul CC2500 este un transceiver de cost redus, care comunică pe frecvența de 2,4GHz și care a fost special proiectat pentru a lucra cu un consum cât mai mic de energie. Circuitul oferă suport pentru administrarea pachetelor, pentru memorarea temporară a acestora, pentru detectarea unui canal liber și pentru măsurarea calității transmisiei. Circuitul CC2480 este un transceiver similar ca și cost și consum, dar folosește ca și frecvență de comunicație 2,4GHz. Circuitul asigură funcționalitatea impusă de protocolul ZigBee (protocol pentru rețele fără fir de tip “mash”).

9.2.2 Sisteme de dezvoltare pentru rețele senzoriale

9.2.2.1 Sistemul eZ430-RF2500

Acest sistem (oferit de firma Texas Instruments) conține elementele minime necesare pentru realizarea unei rețele senzoriale care lucrează pe frecvența radio de 2500MHz. Are în componență o unitate conectabilă la un calculator PC prin interfața USB și un nod de rețea; unitatea conectată la calculator achiziționează mesajele transmise prin unde radio de la nodurile rețelei; se compune dintr-o parte de interfață USB și o unitate de comunicație radio.

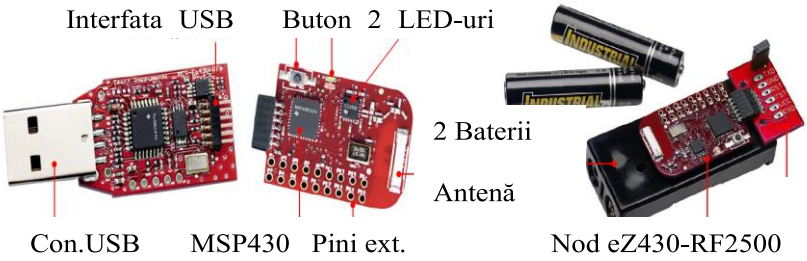


Figura 1. Elementele sistemului de dezvoltare *eZ430-RF2500*

Ambele unități au în componență un procesor de semnale mixte **MSP430-F2274** și un circuit de comunicație radio **CC2500**. Procesorul conține printre altele: 2 timere de 16 biți, un canal serial de comunicație, un convertor analog-numeric de 10 biți, 2 amplificatoare operaționale și interfață paralelă cu 32 de semnale digitale.

Principalele caracteristici tehnice ale sistemului de dezvoltare sunt:

- facilități de programare și depanare prin interfața USB
- 21 de pini pe care se pot conecta diverse dezvoltări
- procesor cu consum de curent extrem de mic care lucrează la 16MHz
- 2 LED-uri de semnalizare vizuală (galben și roșu)
- buton de întrerupere
- distanța de transmisie de 140m la o viteză de transmisie de 10kbps și de 90m la 250kbps

Pentru dezvoltarea de aplicații pe platforme cu MSP430 se pot folosi mediile de programare *TI Code Composer Essentials Evaluation v2.04* sau *IAR Workbench KickStart Version 4.09A*. Ambele instrumente se găsesc pe CD-ul de instalare al sistemului de dezvoltare. Aceste medii oferă facilități de editare, asamblare sau compilare (din limbajul C), descărcare program pe dispozitivul țintă și depanare.

9.2.2.2 Sistemul eZ430-RF2480

Este un sistem de dezvoltare pentru o rețea de tip 2.4GHz ZigBee, care are la bază procesorul MSP430 și transceiverul CCs480. Ca și în cazul precedent sistemul are în componență o unitate conectabilă la calculator prin interfață USB și un nod de rețea.

Caracteristici tehnice ale sistemului de dezvoltare sunt:

- procesor de consum extra-scăzut MSP430
- 5 pini de extensie de intrare/ieșire

- 2 LED-uri pentru semnalizare vizuală
- un buton de întrerupere

Figura 2 Sistemul de dezvoltare *eZ430-RF2480*

Pentru dezvoltare de aplicații se folosește mediul *IAR Embedded Workbench for MSP430.*

Software-ul pentru un nod de rețea are o structură stratificată, organizată pe 3 nivele de abstractizare:

- **Nivelul aplicație** – pachet ce conține o serie de exemple de aplicații cu acces la modulul RF și HAL
- **Remote Procedure Call (RPC)** – implementează un protocol simplu RPC pentru transmisia și recepția pe o interfață de tip SPI
- **Hardware Adaptation Layer (HAL)** – conține componente care asigură accesul la resursele fizice ale sistemului: senzor, LED-uri, buton, timere, interfața serială asincronă (UART), etc.

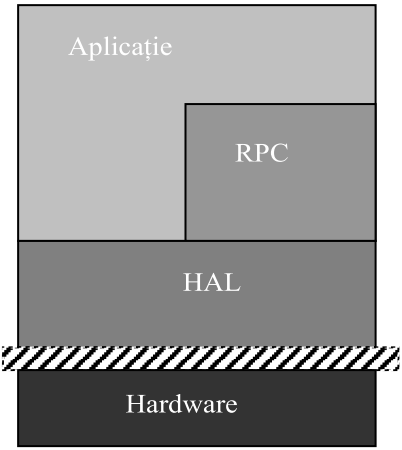


Figura 3 Structura software-ului

Se observă că exemplele de la nivelul Aplicație au acces la resursele fizice fie direct prin nivelul HAL fie prin funcții RPC. Detalii privind exemplele de programare pot fi găsite în documentația sistemului de dezvoltare.

9.2.3 Funcționarea rețelei

Cele două unități ale unui sistem de dezvoltare sunt preprogramate cu câte un firmware ce asigură comunicația în rețea. Firmware-ul pentru unitatea conectată la PC implementează funcțiile unui punct de acces (eng. AP-Access Point); acest nod funcționează în permanență și administrează rețeaua. De asemenea recepționează mesaje de la celelalte noduri ale rețelei (unul singur în cazul de față) și le transmite mai departe pe interfața USB la aplicația de monitorizare.

În a doua unitate firmware-ul implementează funcția unui dispozitiv terminal (eng. ED – end device). Acest nod transmite periodic, o dată pe secundă, datele culese de la senzori: temperatura și tensiunea bateriei.

9.3 Mersul lucrării

1. Instalarea și testarea modulului ***eZ430-RF2500***
   a. Instalarea unui mediu de programare și depanare pentru MSP430; pe CD-ul atașat sistemului se găsesc cele două medii descrise anterior; de pe CD se lansează instalarea unuia dintre medii
   b. Instalarea componentelor hardware: se introduce una dintre unități în interfața USB a calculatorului, și se alimentează cu 2 baterii tip AAA nodul autonom de rețea (se fixează jumper-ul pentru alimentarea modulului); softul aferent pentru interfață se instalează automat; opțional când se solicită software-ul pentru MSP430 Application UART se permite Windows-ului să instaleze automat software-ul (acest lucru este posibil dacă în prealabil s-a instalat IAR Kickstart).
   c. Se lansează aplicația *eZ430-RF Temperature Demo PC* (SensorMonitorGUI.exe) disponibilă pe CD; aplicația vizualizează grafic datele primite de la nodurile rețelei; pentru informații suplimentare privind aplicația se selectează *Help* din meniu.
2. Instalarea și testarea modulului ***eZ430-RF2480***
3. Să se scrie un program pentru cele două sisteme de dezvoltare care citește continuu temperatura, tensiunea bateriei și intensitatea semnalului radio, în mod continuu.

Referințe

1. eZ430-RF2500 Development Tool User guide (disponibil pe CD)
2. A wireless sensor monitor using the eZ430-RF2500 (disponibil pe CD)
3. eZ430-RF2480 user's guide

10 Studierea facilităților senzoriale, de calcul și de comunicație ale unei placi de tip Arduino

10.1 Obiectivul lucrării

Lucrarea își propune să prezinte facilitățile de calcul, senzoriale și de comunicație ale unei placi de dezvoltare de tip Arduino.

10.2 Considerații teoretice

Platforma Arduino este un microsistem de calcul dedicat, cu arhitectură deschisă (nu este proprietatea unei firme) care s-a dezvoltat cu scopul de a permite unor persoane mai puțin avizate în domeniul calculatoarelor (în speță al arhitecturilor hardware) să programeze aplicații simple de monitorizare și control.

O placă de tip Arduino conține de obicei un microcontroler (din familia ARM) și o serie de interfețe pentru achiziția și generarea de semnale analogice și digitale. În plus conține 1-2 canale seriale de comunicație și eventual o interfață de rețea.

Firma Intel (c) a dezvoltat o placă de tip Arduino denumita Intel Galileo care are ca și element central un circuit de tip SoC (system-on-chip) Intel Quark x1000. Acest circuit este de fapt un întreg calculator de tip PC integrat pe un singur circuit; folosește un procesor compatibil cu familia ISAx86 care lucrează la o frecventa de 400MHz. În figura 1 se observă componentele fizice ale plăcii Arduino Intel Galileo.

Figura 2 prezintă schema de conectare a componentelor plăcii Arduino Galileo. Se observă că intrările și ieșirile digitale și analogice nu sunt conectate direct la microprocesor (cum ar fi cazul la o placă Arduino obișnuita, bazata pe microcontroler) ci sunt controlate de niște circuite specializate de tip GPIO. Dialogul dintre microprocesor și circuitele GPIO se realizează pe un canal serial. Din aceasta cauză frecventa de comutare a semnalelor de ieșire este mult mai mică decât în cazul unor placi Arduino obișnuite. Frecventa maxima de comutare a unui singur semnal digital este de 230 Hz. În cazul în care programul încearcă să comute mai multe semnale, frecventa de comutare scade invers proporțional cu numărul de ieșiri comutate.

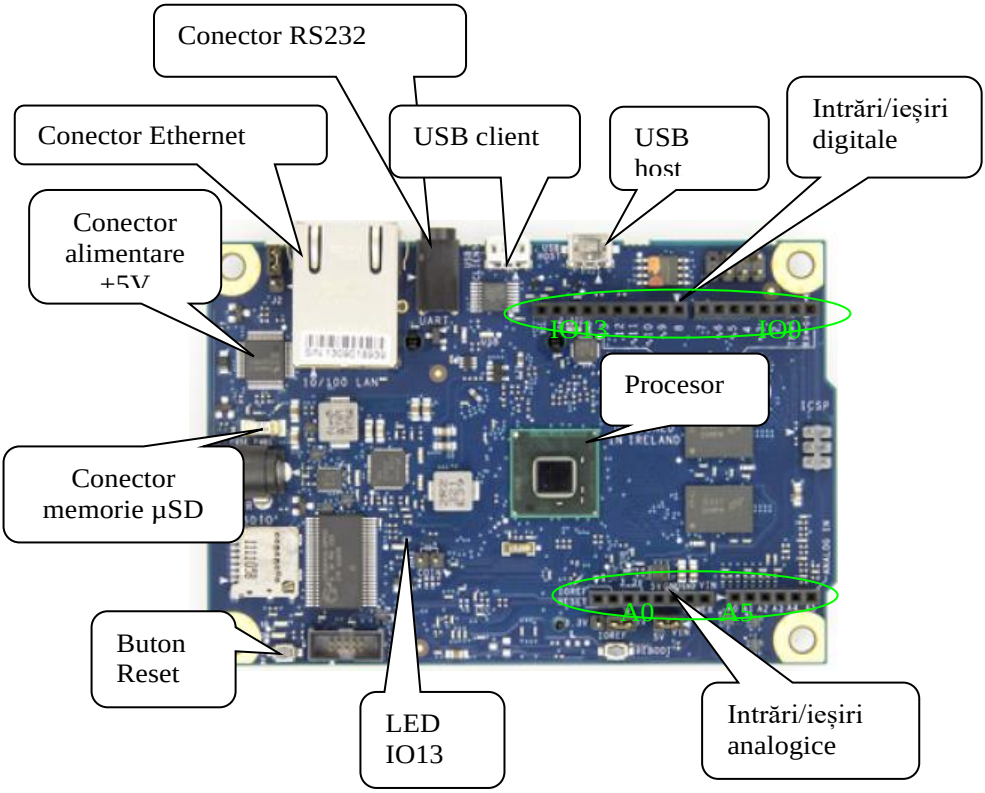


Figura 1. Placa Arduino Intel Galileo

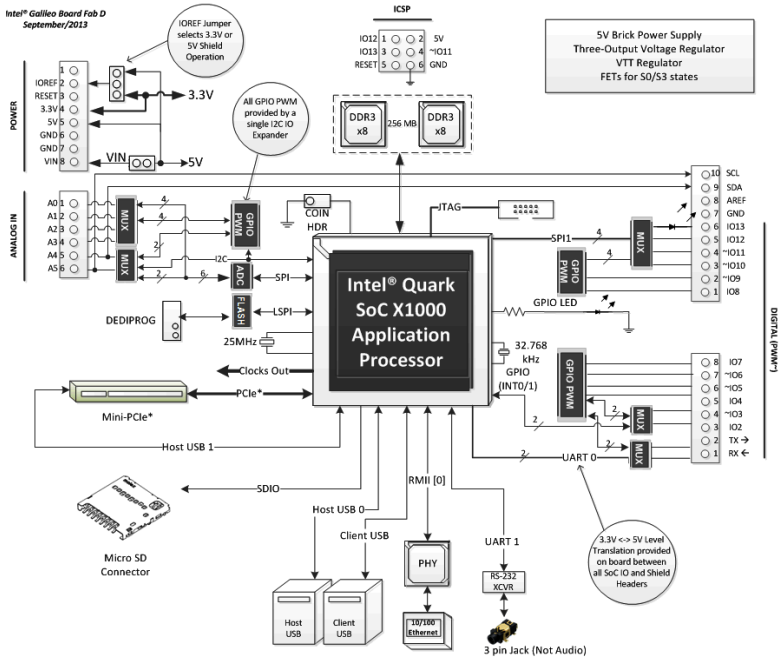


Figura 2 Schema de principiu a plăcii Arduino Galileo

Pe o placa Arduino Galileo s-a instalat o varianta minimala de sistem de operare Linux. O varianta extinsa de Linux se poate instala folosind o memorie externa de tip SD. Sistemul va răspunde la comenzi uzuale Linux (ls, dir, cd) pe un canal serial de tip RS232. Pentru operare de pe un terminal de tip PC se va conecta interfață seriala a plăcii (conectorul de tip microfon de pe placă) cu interfață seriala RS232 a calculatorului PC printr-un cablu special realizat în acest scop. Pe PC se va lansa o aplicație de tip hiperterminal (sau monitor de canal serial) și se va apăsa tasta Enter. În acest mod sistemul va identifica terminalul de pe care se face operarea.

O alta modalitate, mult mai frecventa de operare și de execuție a aplicațiilor pe placa Arduino, este prin intermediul mediului de programare (IDE) Arduino. Acest mediu care rulează pe un PC permite editarea, compilarea și încărcarea în vederea execuției a aplicațiilor scrise pentru placa Arduino. Descărcarea programelor se face prin intermediul unui cablu USB conectat intre conectorul USB al PC-ului și intrarea “USB

client" a plăcii. Pe conexiunea USB se deschide un canal serial virtual care asigura transferul programului ce urmează să se execute.

Important!!! Înainte de cuplarea cablului USB placa trebuie alimentata, folosind adaptorul din dotare. Nu este permisa cuplarea altor tensiuni la placă (inclusiv prin USB) pe durata cat placa nu este alimentată.

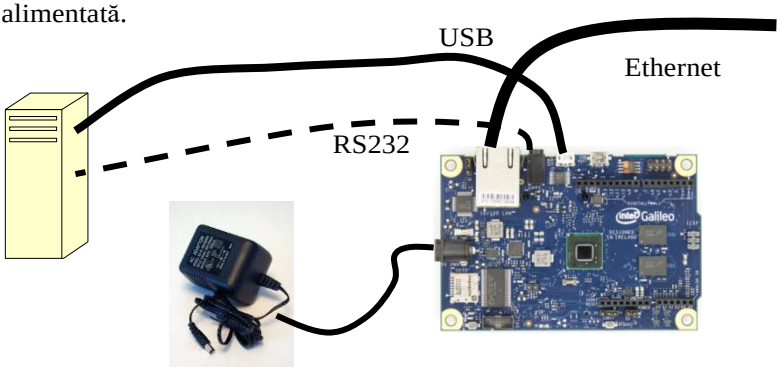


Figura 3 Conectarea plăcii Arduino

Limbajul de programare este unul asemănător cu limbajul "C". În structura unui program există două parți:

- o parte care se execută o singura dată – funcția "setup()
- o parte care se execută în mod repetitiv – funcția "loop()

În funcția setup() se vor include inițializările necesare pentru execuția programului, iar în funcția loop(), corpul aplicației, care înseamnă o secvența de operații executate într-o bucla infinita. Acest mod de execuție se bazează pe observația ca un program de monitorizare sau de control are o parte de inițializare și configurare și o alta parte care efectuează repetitiv operații de citire semnale, procesare și generare de comenzi.

Mediul de programare Arduino conține o biblioteca bogata de funcții (proceduri) prin intermediul cărora programatorul poate să acceseze resursele fizice ale plăcii: semnale digitale de intrare/ieșire, semnale analogice de intrare și de ieșire, interfețe seriale, sau alte interfețe care pot fi atașate plăcii Arduino.

Mediul de programare pune la dispoziția utilizatorilor un număr mare de exemple de programare (din meniu: File->Examples). Aceste exemple au menirea de a exemplifica modul de accesare a diferitelor resurse ale plăcii și modul de utilizare a funcțiilor de biblioteca. Exemplele variază de la unele simple care demonstrează cum se controlează un semnal digital de ieșire (exemplul Blink care aprinde și stinge un LED) sau cum se citește o intrare digitala (Buton) și pana la exemple mai complexe în care se utilizează o interfață de rețea Ethernet

pentru a implementa un client sau server de web (exemplele WEB server și WEB client).

O aplicație scrisa în acest mediu poarta numele de schița (eng. sketch). Prin intermediul mediului de programare utilizatorul poate să compileze programul scris (Sketch->Compile) și apoi poate să-l lanseze în execuție (butonul ->).

Mai jos este un exemplu simplu de program care aprinde și stinge un LED.

```
// semnalul de intrare/ieșire digitala IO13 pe cele mai multe placi Arduino are conectat un LED.
int led = 13;

void setup() {
  // inițializarea semnalului digital IO13 ca și ieșire
  pinMode(led, OUTPUT);
}

// bucla care se executa la infinit:
void loop() {
  digitalWrite(led, HIGH);   // aprinde LED-ul (HIGH – tensiune ridicata)
  delay(1000);               // așteaptă o secunda
  digitalWrite(led, LOW);    // stinge LED-ul
  delay(1000);               // așteaptă o secunda
}
```

10.3 Mersul lucrării

1. Se vor testa exemplele din mediul de programare pentru controlul semnalelor digitale și analogice; vor fi efectuate operații pentru:
   - aprinderea și stingerea unui LED,
   - citirea poziției unui buton,
   - citirea și scrierea unor semnale analogice
   - transmiterea unei informații pe canalul serial (virtual)

Pentru fiecare exemplu se va realiza o schema electrica în conformitate cu necesitățile exemplului.

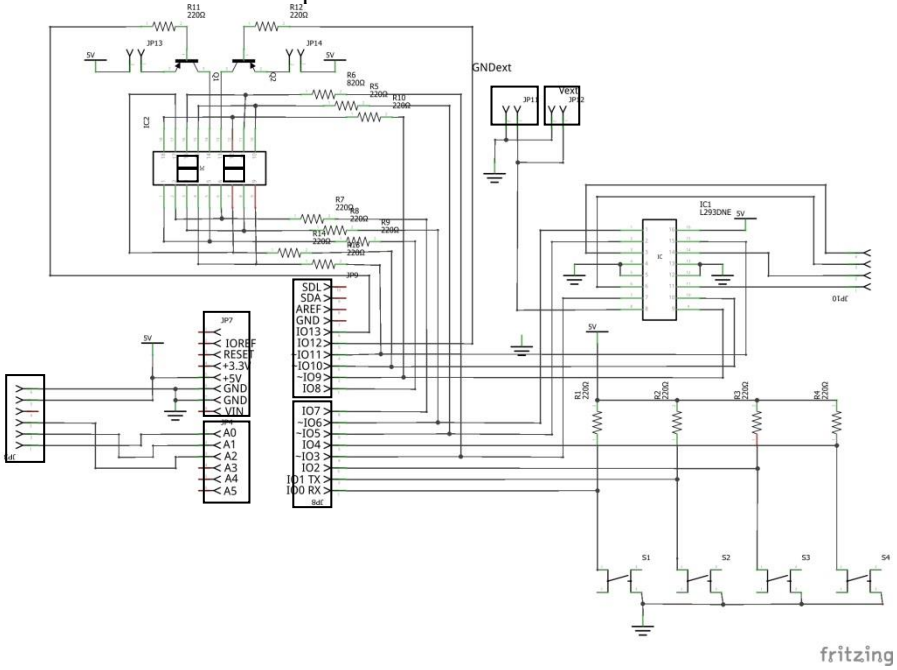
2. Se vor scrie și testa programe pentru placa de extensie (shield) dotata cu:
   - două afișoare 7 segmente
   - 4 butoane

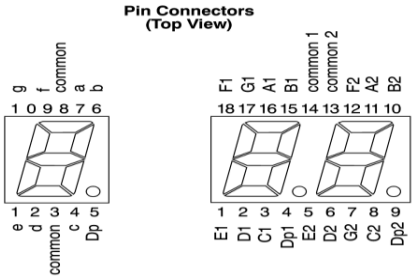
- circuit de adaptare (driver) pentru comanda în punte a unor motoare

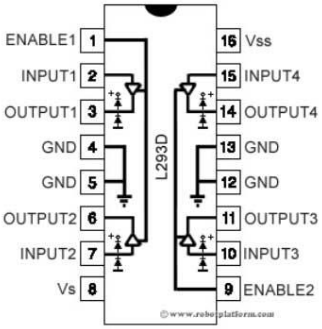
- conectori de extensie pentru semnale analogice

Schema electrica a plăcii de extensie se găsește în anexa 1.

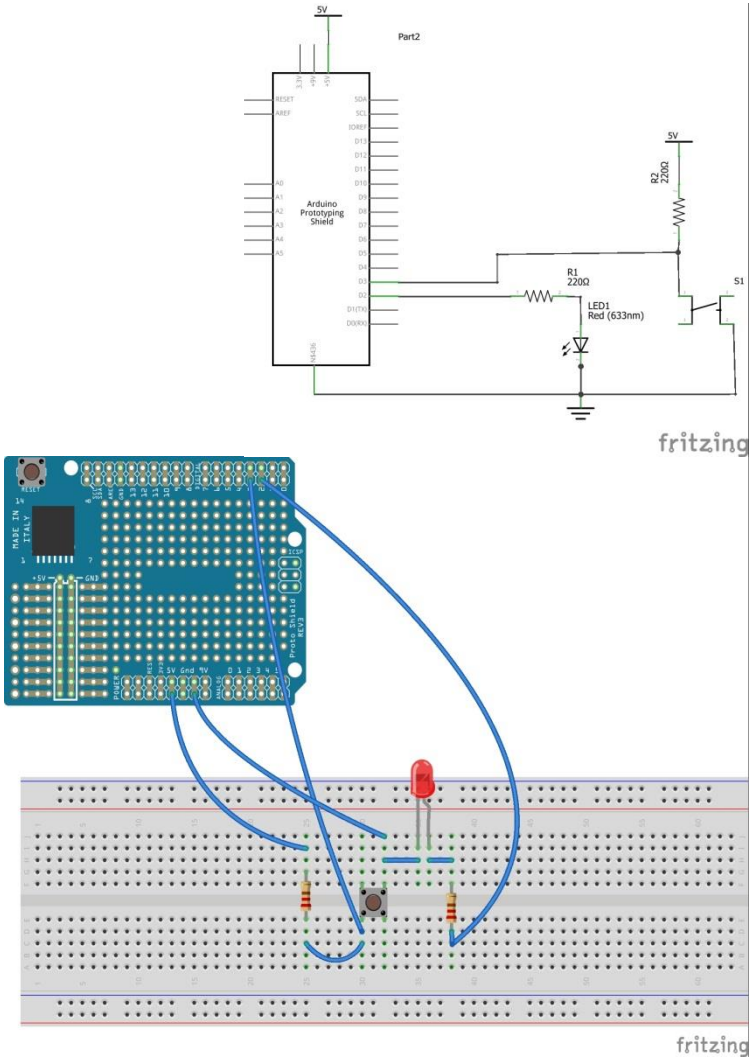
Anexa 1. Schema plăcii de extensie







Anexa 2 Schema pentru LED și Buton



11 Internet of Things - Accesul prin Internet la obiecte sau dispozitive simple

11.1 Obiectivul lucrării

Lucrarea își propune să studieze posibilitățile de conectare a unor dispozitive simple la rețeaua Internet. Vor fi monitorizate și controlate de la distanță anumite dispozitive simple (ex. LED, senzor de temperatură, motor electric, etc.) prin intermediul unui browser de web.

11.2 Considerații teoretice

IoT sau "Internetul obiectelor" este o tendință nouă în domeniul comunicației pe Internet care își propune să asigure mecanismele necesare pentru conectarea unui număr foarte mare de dispozitive simple pe Internet și controlul acestora de la distanță. În viziunea IoT toate dispozitivele care ne înconjoară ar putea fi conectate pe Internet cu scopul de a fi monitorizate sau controlate din orice punct geografic care asigura o conectivitate la Internet. Dispozitive industriale, echipamente medicale, dispozitive de monitorizare a mediului sau a sănătății pacienților pot fi conectate la Internet în măsura în care dispun de o interfață de rețea și au capabilitatea de a implementa stiva de protocoale tipica pentru internet, adică TCP/IP.

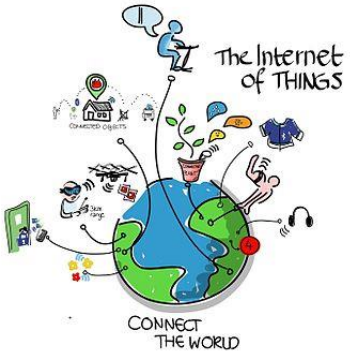


Figura 1. IoT- Internet of Things (wikipedia)

Realizarea unui astfel de deziderat implica soluționarea anumitor probleme specifice, cum ar fi: adresarea unui număr foarte mare de dispozitive (de ordinul zecilor de miliarde), optimizarea protocoalelor pentru transmisia de date de lungime redusa (biți, octeți), asigurarea securității și fiabilității datelor transmise, înregistrarea automata a datelor culese de la senzori, detectarea și tratarea evenimentelor, transmisia în timp-real a datelor, etc. Multe din aceste cerințe se regăsesc și în cazul rețelelor industriale de comunicații.

Deocamdată s-au făcut putini pași în aceasta direcție, ceea ce ne obliga să utilizam protocoalele existente pe Internet: în primul rând TCP/IP, HTTP sau XMPP (folosit pentru "almost real-time messaging).

Principalele aplicații ale unei astfel de conectivități ar fi: controlul de la distanță al proceselor industriale, monitorizarea mediului, automatizarea clădirilor, monitorizarea pacienților, controlul traficului rutier și feroviar, urmărirea și asigurarea trasabilității produselor, etc.. O astfel de abordare ar permite utilizarea multipla a unor senzori sau dispozitive. De exemplu un senzor de nivel care măsoară debitul unui râu ar fi utilizat de agenția de monitorizare a râurilor, de o companie hidroenergetica sau de Agenția de monitorizare a catastrofelor. Un alt exemplu ar fi o camera web care monitorizează traficul într-o anumita zona și ar fi accesibil atât instituțiilor responsabile cu controlul traficului cat și publicului larg.

Din punct de vedere al dispozitivelor conectabile pe Internet principala provocare este de a implementa setul de protocoale specifice Internetului pe un microsistem de calcul cu resurse limitate. Principalele neajunsuri ar fi: dimensiunea foarte mică a memoriei de date, lipsa unui sistem de operare (care să înglobeze driverele de protocol), lipsa unui mod de lucru multi-threaded (necesar pentru tratarea în paralel a mesajelor transmise și recepționate.

Există o serie de exemple de implementare a unei interfețe Ethernet și chiar Internet pe sisteme bazate pe microcontroloare [1]. În general aceste soluții exploatează în mod ingenios memoria de date existentă și simulează o execuție concurenta de tip multithreading. În cazul implementării pe un microcontroler de tip PIC16/18Fxx memoria de date disponibila este comparabila sau chiar mai mică decât dimensiunea unui mesaj minim Ethernet, ceea ce înseamnă ca formarea mesajelor transmise și interpretarea celor primite se face "on the fly" adică în timpul transmisiei propriu-zise.

Placa Arduino Intel Galileo este un caz fericit în care există elementele constitutive necesare pentru implementarea ideii de IoT și anume: pe placă există o interfață Ethernet (controlor Ethernet), pe placa rulează un sistem de operare (Linux minimal) și procesorul (un SoC –

System on chip) dispune de memorie suficienta pentru gestionarea mesajelor și are performante de calcul suficiente (ex. frecventa ceasului de 400MHz). În plus placa dispune de interfețe pentru achiziția și generarea de semnale analogice și digitale, prin care pot fi conectate diverse dispozitive de măsura și de execuție (senzori, relee, motoare electrice, etc.).

Printre exemplele oferite de mediul de programare Arduino există câteva care demonstrează modul în care placa poate fi conectata la Internet. Accesul la aceste exemple se face din meniul aplicației: File->examples->Ethernet. Primele exemple recomandate sunt "Web server" și Web client" în care aplicațiile implementează rolurile de server și client web. Aceste exemple arata secvența de comenzi necesare pentru realizarea unei conexiuni HTTP și transmiterea respectiv recepția unor pagini web. Prin aceste pagini web pot fi vizualizate anumite semnale achiziționate de placa. Se observă ca funcțiile de comunicație în rețea sunt soluționate prin biblioteca "Ethernet" accesibila prin fișierul header Ethernet.h. Detalii privind aceste funcții pot fi accesate din mediul de programare Help->Reference->Librarie->Ethernet (ex: file:///C:/arduino-1.5.3/reference/Ethernet.html).

Pentru vizualizarea datelor recepționate sau transmise de către placa Arduino se folosește un canal serial virtual implementat pe conexiunea USB dintre PC și placa Arduino. După lansarea programului se va deschide din meniu un monitor de canal serial (hyperterminal): Tools->Serial Monitor. În cadrul programului clasa "Serial" implementează funcțiile de recepție și transmisie.

11.3 Mersul lucrării

1. Se vor testa exemplele simple (Web server și Web client) date în mediul de programare. Pentru accesarea plăcii se va folosi o aplicație browser de web. În setările browserului se va scoate temporar accesul la Internet prin "proxy" (daca există) sau tratarea adresei IP setate ca o excepție de la accesul prin proxy (ex. exception 10.129.025/32).
   Atenție!!! adresa IP se va seta corespunzător configurației de rețea în care se conectează placa; se recomanda utilizarea unui cablu de rețea de la un calculator existent și copierea adresei de IP a calculatorului înlocuit.
2. Se vor conecta diferite dispozitive (ex. potențiometru, comutator, LED) la intrările și ieșirile analogice și digitale și se va testa posibilitatea de a citi valoarea intrărilor și de a seta valoarea semnalelor de ieșire.

3. Se va implementa aplicația din anexa 1 și se va verifica transferul bidirecțional al datelor. Se vor face modificări în program pentru a schimba regimul de lucru al aplicației.

Bibliografie

1. Jerome Bentham, TCP/IP Lean: Web Servers for Embedded Systems, Second Edition,
http://read.pudn.com/downloads30/sourcecode/embed/95033/TCPIP%20Lean.pdf

Anexa 1 Aplicație de citire a intrărilor analogice și control al unei ieșiri (LED).

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(10,129,0,25);

// Initializarea bibliotecii Ethernet server
// cu adresa  IP și portul dorit
// (port 80 este “default” pentru HTTP):
EthernetServer server(80);
String readString;

void setup() {
  // Deschide canal serial:
  Serial.begin(9600);
   while (!Serial) {
    ; // asteapta să se deschida; doar pentru placa Leonardo
  }

 pinMode(13, OUTPUT);
  Ethernet.begin(mac, ip);  // porneste conexiunea Ethernet și serverul:
  server.begin();
  Serial.print("serverul este la ");
  Serial.println(Ethernet.localIP());
}

void loop() {
  EthernetClient client = server.available();  // asculta apelul clientilor
  if (client) {
    Serial.println("client nou");
    boolean currentLineIsBlank = true;      // o cerere http se sfarseste cu o
linie goala

    while (client.connected())
    {
      if (client.available())
      {
        char c = client.read();
         if (readString.length() < 100) {
```

```
        readString += c; //stocheaza caracterele în string
      //Serial.print(c);
    }
     Serial.write(c);
     // daca ai ajuns la sfarsitul liniei (receptionezi caracterul newline)
     // și linia este goala, cererea http s-a sfarsit,
     // deci poti să trimiti un raspuns
     if (c == '\n' && currentLineIsBlank)
     {
      // trimite un header standard HTTP
      client.println("HTTP/1.1 200 OK");
      client.println("Content-Type: text/html");
      client.println("Connection: close");
      client.println();
      client.println("<!DOCTYPE HTML>");
      client.println("<html>");
      // adauga un meta tag de refresh pentru ca
      //browserul să resolicite pagina la fiecare 5 secunde:
      //client.println("<meta http-equiv=\"refresh\" content=\"5\">");
      // trimite valoarea fiecarui canal analogic
      for (int analogChannel = 0; analogChannel < 6; analogChannel++)
      {
       int sensorReading = analogRead(analogChannel);
       client.print("intrarea analogica ");
       client.print(analogChannel);
       client.print(" este ");
       client.print(sensorReading);
       client.println("<br />");
      }
      client.println("<a href=\"/?lighton\"\">Aprinde LED</a>");
      client.println("<a href=\"/?lightoff\"\">Stinge LED</a><br />");
      client.println("</html>");
      break;
     }
     if (c == '\n')
     {
      // incepi o noua linie
      currentLineIsBlank = true;
     }
     else if (c != '\r')
     {
      // ai primit un caracter pe noua linie
      currentLineIsBlank = false;
```

```
        }
             // Serial.println(readString);
        ////////////////////// controlul LED-ului de pe arduino
         if(readString.indexOf("?lighton") >0)//verifica “Aprins
         {
           digitalWrite(13, HIGH);    // pune bitul 13 pe HIGH
           Serial.println("Led Aprins");
           //Serial.println(readString);
           readString="";
         }
         else{
         if(readString.indexOf("?lightoff") >0)//verifica pentru stins
         {
           digitalWrite(13, LOW);    // pune bitul 13 pe LOW
           Serial.println("Led Stins");
           //Serial.println(readString);
           readString="";
         }
        }

      }
    }
    // timp necesar pentru ca browserul să receptioneze data
    delay(1);
    // inchide conexiunea:
    client.stop();
    Serial.println("client deconectat");
    readString="";
  }
}
```

12 Controlul unor procese complexe printr-o platforma de tip Arduino

12.1 Obiectivul lucrării

Lucrarea își propune să prezinte modul de implementare a unor aplicații complexe controlate prin intermediul unei placi Arduino:

- instalație industriala
- casa inteligenta

În ambele cazuri se folosesc machete ale proceselor fizice reale.

12.2 Considerații teoretice

O aplicație de monitorizare și control este destinata pentru urmărirea și controlul automat al unui anumit proces, cum ar fi o instalație industriala, o linie de fabricație, o rețea de distribuție (energie electrica, gaze, etc.), un echipament complex (ex. robot, utilaj cu comanda numerică, etc.) sau o clădire. O astfel de aplicație integrează mai multe categorii de componente de automatizare cum ar fi:

a. **Senzori sau traductoare** – menite să măsoare starea/valoarea anumitor parametrii de proces cum ar fi:

- temperatura, presiune, umiditate
- poziție, orientare, nivel, proximitate, viteza, accelerație,
- curent, tensiune, putere, factor de putere
- compoziție chimica, PH, vâscozitate

În funcție de natura semnalelor generate senzorii pot fi de mai multe tipuri:

- senzori analogici – generează o tensiune sau un curent proporțional cu valoarea parametrului măsurat; plaja de variatie:0-5V, 010V, 4-20mA, etc.
- senzori digitali – generează semnale digitale simple (0 sau 1) sau complexe: modulate în frecventa de impulsuri, lățime de impuls (PWM), mesaj în rețea, etc.

b. **Elemente de execuție** – care au rolul de a acționa asupra procesului controlat cu scopul de a modifica anumiți parametrii de proces și implicit evoluția procesului; exemple de elemente de execuție (acționare):

- motoare electrice pas-cu-pas, de curent continuu (cc) și de curent alternativ (ca)
- electromagneți, motoare liniare, servi-motoare

- relee
- elemente de încălzire (ex. rezistente de încălzire)
- robinete, valve, pompe, etc.

c. **Elemente de control** – destinate pentru generarea controlului pe baza unei anumite legi prestabilite (legea de reglaj automat); astfel de elemente pot fi:

- microcontroloare, procesoare de semnal, platforme cu astfel de circuite (ex. PICDEM 4, EZ430)
- regulatoare, controloare logice programabile (PLC)
- platforme universale de monitorizare și control (ex. Arduino, Raspberry pi, Edison, WeMOS, etc.)

d. **Elemente de comunicație** – destinate pentru interconectarea componentelor de mai sus, cum ar fi:

- extensii de rețea: Bluetooth, WiFi, ZigBee,
- interfețe pentru rețele industriale: CAN, ProfiNet, EtherCat

Programul de aplicație, înscris de obicei într-o memorie nevolatila a elementului de control, implementează următoarele funcții de baza:

- culege date despre proces prin intermediul senzorilor
- procesează datele achiziționate (ex. prin proceduri de filtrare, agregare, calculul comenzii următoare, etc.)
- generează comenzi în scopul controlării procesului; comenzile sunt transmise către elementele de execuție
- comunica cu un operator uman printr-o interfață de tip "panou frontal" ce conține: butoane, afișaj cu leduri, cu 7 segmente, afișaj alfanumeric, etc.)
- opțional comunica cu un calculator ierarhic superior de obicei printr-o interfață seriala sau printr-o rețea (ex. wifi, bluetooth sau chiar Internet)

Primele trei funcții (achiziție, procesare și generare de comenzi) se execută într-o ordine secvențială cu o anumită periodicitate. Perioada de repetiție este dată de dinamica procesului controlat, adică de viteza de variație a parametrilor urmăriți. Conform teoremei lui Shannon din teoria semnalelor frecventa de repetiție sau de eșantionare trebuie să fie mai mare decât dublul frecventei maxime prezente în semnalul achiziționat. Pentru o calitate mai buna a reglajului se alege de obicei o frecventa de 5-10 ori mai mare decât frecventa maxima a semnalelor de intrare.

Funcțiile de reglaj (menționate anterior) vor fi executate în regim concurent cu celelalte funcții ramase (comunicarea cu operatorul uman și cu un alt calculator). Execuția în regim concurent poate ridica anumite probleme de implementare mai ales în cazul sistemelor uni-procesor și care nu dispun de un sistem de operare care să ofere mecanisme de tip multi-tasking sau multi-fir. Cum se va vedea în exemplele din lucrare, acest regim de lucru concurent trebuie să se implementeze în aplicația

propriu-zisa de control și sunt necesare masuri speciale care să nu blocheze procesorul pe o anumita funcție în detrimentul celorlalte funcții. De exemplu de durata cat operatorul uman sau un alt calculator comunica cu procesorul funcțiile de monitorizare și reglaj nu trebuie să fie întrerupte sau să se modifice periodicitatea acestora.
Pentru a asigura o execuție concurenta a funcțiilor implementate de aplicație se impune mixarea pașilor care alcătuiesc aceste funcții. De exemplu pașii unui motor pas cu pas se pot mixa cu pașii altui motor și cu pașii de deservire a unei cereri utilizator (venita fie pe canalul serial fie pe web). Figura 1 sugerează acest mod de funcționare.

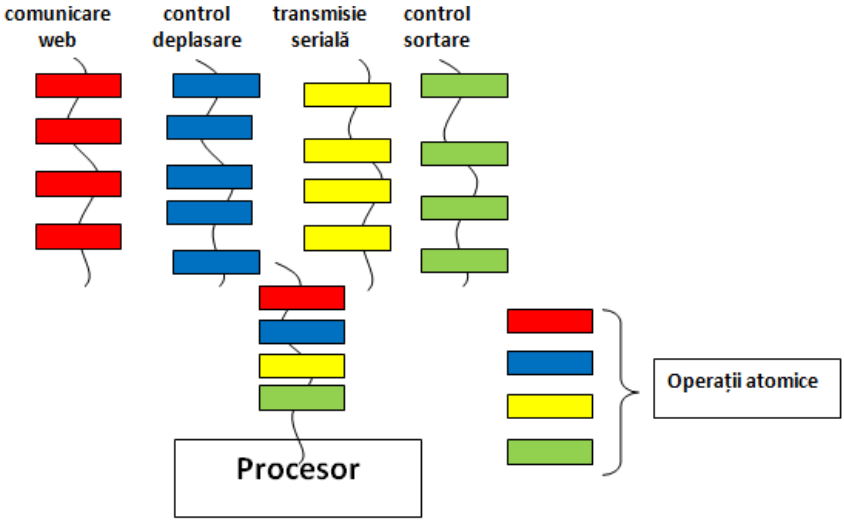


Figura 1 Execuție în regim concurent

În continuare vor fi descrise două aplicații de monitorizare și control, relevante pentru domeniul industrial și respectiv cel al clădirilor inteligente.

12.3 Sistem de monitorizare a unui proces industrial (Partea I)

În scopul demonstrării facilităților de achiziție, procesare, control și comunicare a unei platforme universale de tip Intel Arduino Galileo s-a realizat o macheta a unei linii de dozare. În figura 2 se pot observa principalele subsisteme ale procesului:

- Sistemul de alimentare cu cutii mici și mari

- Sistemul (masa) de deplasare a cutiilor
- Stația de dozare
- Sistemul de sortare (debarasare)
- Sistemul optic pentru detecția poziției cutiilor
- Sistemul de detecție a poziției inițiale a mesei
- Modulul de control format din placa Arduino Galileo și placa de extensie (shield)
- Sursa externa de alimentare

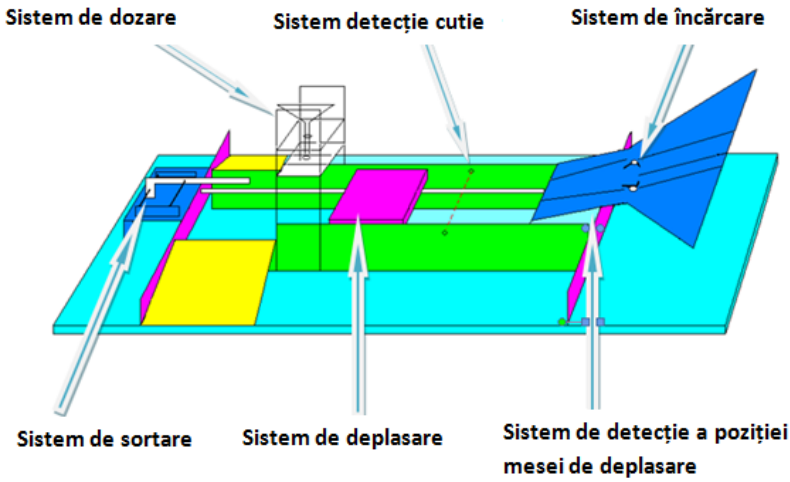


Figura 2. Macheta unei linii industriale de dozare

Cele 6 etape ale procesului de fabricație sunt:

- calibrarea poziției mesei de deplasare

Această etapă este necesară la începutul unui proces întrucât masa de deplasare a cutiilor trebuie să se afle sub sistemul de încărcare. Din diverse motive, cum ar fi o pană de curent aceasta ar putea să nu fie în starea potrivită. Astfel, această etapă verifică dacă masa de deplasare e în poziția corespunzătoare sau nu , în continuare acționându-se în consecință.

- etapa de încărcare

Aceasta etapă presupune punerea pe masa de deplasare a unui recipient (cutie în cadrul machetei de față).

- etapa de detecție

În cadrul acestei etape se detectează dacă pe masa de deplasare se află un recipient care urmează a fi încărcat cu bile. În cazul în care se remarcă absența acestuia masa se întoarce la etapa de încărcare, în caz contrar se trece la etapa următoare.

- etapa de deplasare

Presupune deplasarea mesei împreună cu cutia până la etapa de dozare.

- **etapa de dozare**

În această etapă se realizează umplerea cu un anumit număr de bile (specificat de utilizator) a cutiei de pe masa de deplasare.

- **etapa de sortare**

Această etapă presupune sortarea celor două tipuri de recipiente : cutii mici și mari. Astfel, cutiile mici sunt deplasate în partea dreaptă iar cele mari în cealaltă parte.

Figura următoare prezintă modul de operare al sistemului. Se observă că sistemul poate fi operat (control și monitorizare) atât local printr-un PC atașat pe un canal serial (implementat pe USB) cat și de pe Internet, printr-o interfață web.

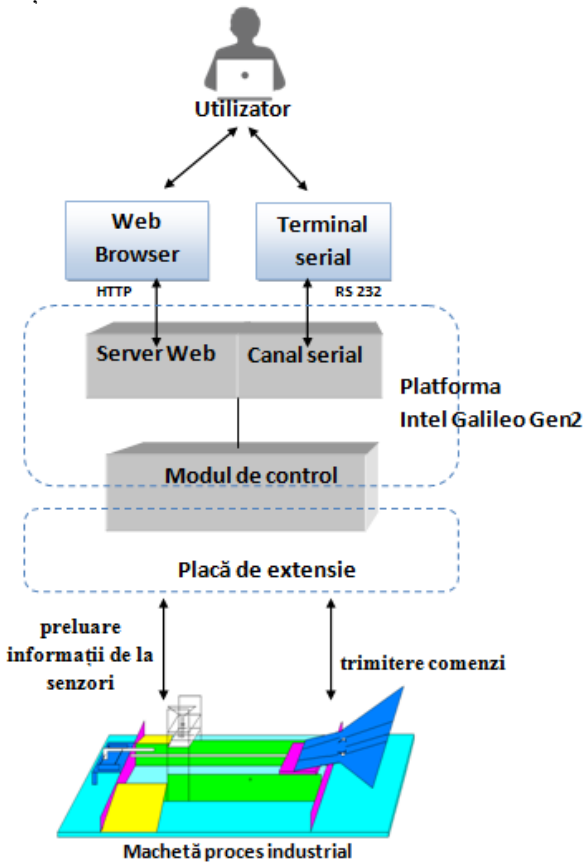


Figura 3. Modul de operare al sistemului: local și de la distanță.

Elementele de automatizare utilizate în cadrul sistemului au fost:

- Motorul de antrenare a mesei – motor pas-cu-pas unipolar
- Motorul de sortare – motor pas-cu-pas bipolar
- Cate o pereche de electromagneți (stânga-dreapta) pentru alimentarea cu cutii mari și mici
- Senzor pentru calibrarea poziției mesei – senzor mecanic bipozițional
- Sistem optic pentru detecția poziției cutiilor pe masa – LED +fototranzistor
- Electromagnet de dozare a bilelor – electromagnet bipozițional (normal destins)
- Interfața dintre placa Arduino și elementele de automatizare menționate mai sus s-a realizat pe o placă de extensie cuplată direct pe conectorii standard ai plăcii Arduino. Aceasta interfață conține următoarele circuite de adaptare:
- Driver pentru motor pas-cu-pas bipolar – L293D
- Driver pentru motor pas-cu-pas unipolar și controlul releelor – ULN2803A
- 4 relee pentru comanda eliberării cutiilor mari și mici
- 1 releu + un tranzistor pentru comanda dozatorului de bile
- Circuit invertor – pentru implementarea logicii de control
- Rezistente de adaptare pentru senzorul optic și mecanic

Figura de mai jos indica modul de amplasare a conectorilor și destinația acestora.

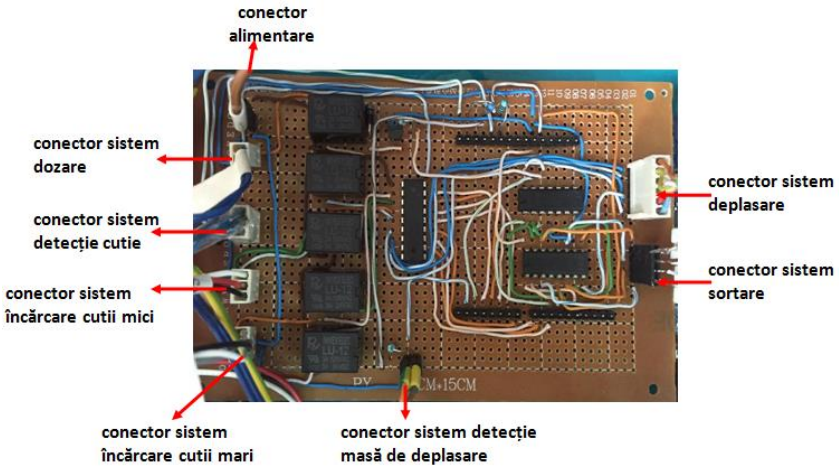


Figura 4. Placa de interfață

În tabelul de mai jos se prezintă modul de alocare a semnalelor plăcii Arduino.

Tabel 1. Alocarea semnalelor plăcii Arduino.

Nume semnal	Tip	Destinație
IO0	nedefinit	Liber
IO1	Intrare digitala	Senzor calibrare poziție masa
IO2	Ieșire digitala	Releu dozator
IO3	Ieșire digitala	Selecție electromagnet stânga-dreapta cutii mari
IO4	Ieșire digitala	Alimentare releu cutii mari
IO5	Ieșire digitala	Selecție electromagnet stânga-dreapta cutii mici
IO6	Ieșire digitala	Alimentare releu cutii mici
IO7	Ieșire digitala	Motor masa Faza 1
IO8	Ieșire digitala	Motor masa Faza 2
IO9	Ieșire digitala	Motor masa Faza 3
IO10	Ieșire digitala	Motor masa Faza 4
IO11	Ieșire digitala	Motor sortare Faza 1
IO12	Ieșire digitala	Motor sortare Faza 2
IO13	Ieșire digitala	Validare motor sortare
GND	masa	
+5V	alimentare	
A0	Intrare analogica	Senzor optic pentru detecția poziție cutiilor

12.3.1 Mersul lucrării (Partea I)

1. Se vor analiza componentele utilizate și schema electrică generală (anexa 1A).
2. Utilizându-se mediul de programare Arduino Galileo IDE se vor scrie proceduri pentru operarea principalelor componente ale sistemului:
    a. Selecție cutii
    b. Deplasare masa (inclusiv cu detecție poziție inițială)
    c. Sortare,
    d. Dozare bile
    e. Detecție cutie (sistem optic)
3. Dispecerizarea comenzilor primite de la utilizator și vizualizarea pașilor executați.
4. Se va scrie un program care lucrează în regim concurent.

5. Se va scrie un program ce permite controlul de la distanță prin Internet (aplicație web-server).

12.4 Monitorizarea și controlul unei case inteligente (Partea a II-a)

Prin definiție, o clădire inteligenta conține o serie de elemente de automatizare care să permită monitorizarea și controlul parametrilor de funcționare ai unei clădiri cu scopul de a optimiza funcționarea acesteia. Pentru a demonstra soluțiile principiale de monitorizare și control aplicabile la o clădire s-a realizat o macheta a unei case (vezi figura 5)

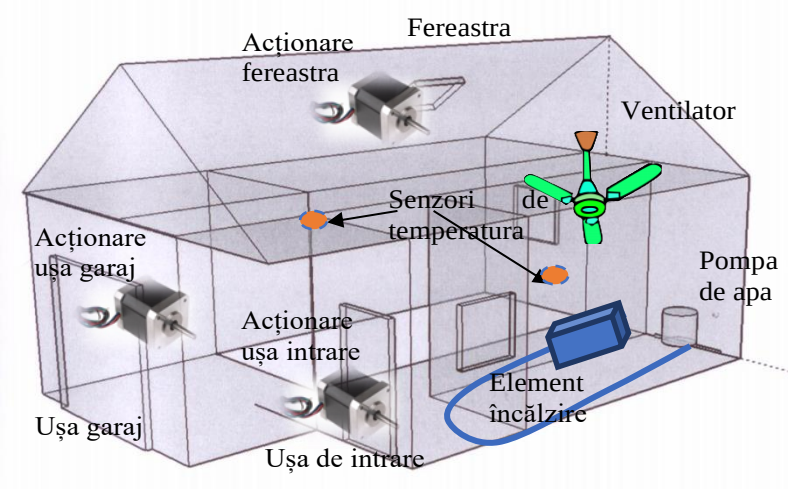


Figura 5. Macheta unei case

Principalele subsisteme ale casei inteligente sunt:

- Sistem de acționare a ușii de garaj
- Sistem de acționare a ușii de intrare
- Sistem de acționare a unei ferestre
- Sistem de încălzire (cu element de încălzire și pompa de apa)
- Sistem de ventilație/aer condiționat (ventilator)
- Senzori pentru măsurarea temperaturii din interior și exterior (opțional-umiditate)
- Sistem pentru telecomanda ușilor
- Senzor de ploaie
- Senzor pentru închiderea ferestrelor (în lucru)

- Modulul de control format din placa Arduino Galileo și placa de extensie (shield)
- Sursa externa de alimentare

Sistemul îndeplinește următoarele funcționalități:
- Deschiderea/închiderea ușii de la intrare pe baza unui cod de acces
- Deschiderea/Închiderea ușii de garaj
- Controlul temperaturii din interior prin sistemul de încălzire și de ventilație
- Deschiderea/închiderea ferestrei de la mansarda în funcție de umiditatea externa
- Verificarea stării ferestrelor

Elementele de automatizare utilizate în cadrul sistemului au fost:
- Motorul de antrenare a ușii de garaj – motor pas-cu-pas unipolar
- Motorul de antrenare a ușii principale – motor pas-cu-pas unipolar
- Motor de antrenare a ferestrei – motor pas-cu-pas unipolar
- Senzori infraroșu pentru detecția codului de acces –(în lucru)
- 2 senzori de temperatura (interior și exterior)
- 1 senzor de umiditate externa
- Senzori pentru ferestre (în lucru)
- Ventilator
- Pompa de recirculare a apei
- Rezistenta de încălzire

Interfața dintre placa Arduino și elementele de automatizare menționate mai sus s-a realizat pe o placă de extensie cuplată direct pe conectorii standard ai plăcii Arduino. Aceasta interfață conține următoarele circuite de adaptare:
- Driver pentru motor pas-cu-pas bipolar – L293D – folosit pentru motor unipolar
- Driver pentru motor pas-cu-pas unipolar și controlul releelor – ULN2803A – folosit în regim de multiplexare pentru 2 motoare pas-cu-pas unipolare
- 2 relee pentru selecția alimentarii celor 2 motoare controlate alternativ prin multiplexare
- 2 relee pentru alimentarea sistemului de încălzire
- 1 releu + un tranzistor pentru ventilație
- Circuit invertor – pentru implementarea logicii de control
- Rezistente de adaptare pentru senzorul optic și mecanic (ferestre)

Figura 6 indica imaginea plăcii de extensie. Se pot observa componentele amplasate pe placa și destinația conectorilor.

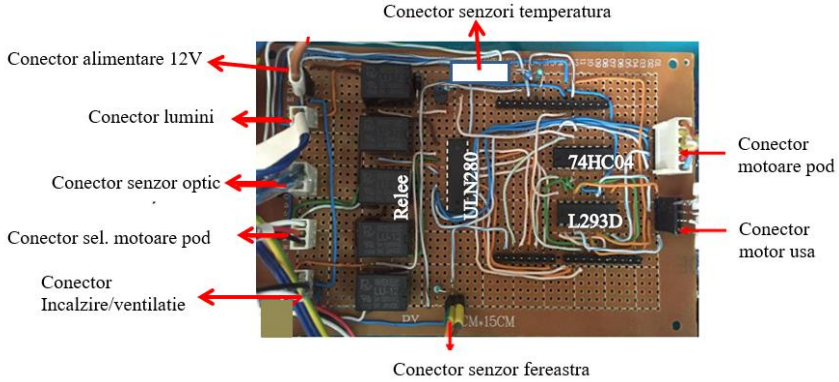


Figura 6. Placa de interfață (extensie)

În tabelul de mai jos se prezintă modul de alocare a semnalelor plăcii Arduino.

Tabel 2. Alocarea semnalelor plăcii Arduino.

Nume semnal	Tip	Destinație
IO0	nedefinit	Liber
IO1	Intrare digitala	Senzor fereastra
IO2	Ieșire digitala	Releu alimentare lumini
IO3	Ieșire digitala	Selecție sistem de încălzire și ventilație (0-incalzire, 1-ventilatie)
IO4	Ieșire digitala	Alimentare încălzire sau ventilație
IO5	Ieșire digitala	Selecție motor ușa garaj sau fereastra (1-garaj, 0-fereastra)
IO6	Ieșire digitala	Alimentare motoare
IO7	Ieșire digitala	Motoare Faza 1
IO8	Ieșire digitala	Motoare Faza 2
IO9	Ieșire digitala	Motoare Faza 3

IO10	Ieșire digitala	Motoare Faza 4
IO11	Ieșire digitala	Motor ușa Faza 1
IO12	Ieșire digitala	Motor ușa Faza 2
IO13	Ieșire digitala	Validare motor ușa
GND	masa	
+5V	alimentare	
A0	Intrare analogica	Sistem optic
A1	Intrare analogica	Temperatura interioara
A2	Intrare analogica	Temperatura exterioara

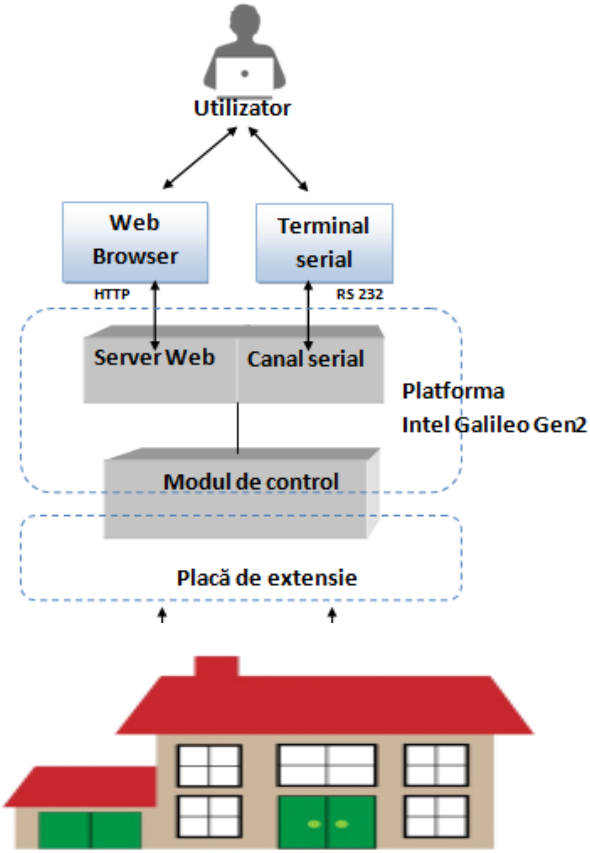


Figura 7. Modul de operare al sistemului: local și de la distanță

Figura 7 prezintă modul de operare al sistemului. Se observă că sistemul poate fi operat (control și monitorizare) atât local printr-un PC atașat pe un canal serial (implementat pe USB) cat și de pe Internet, printr-o interfață web.

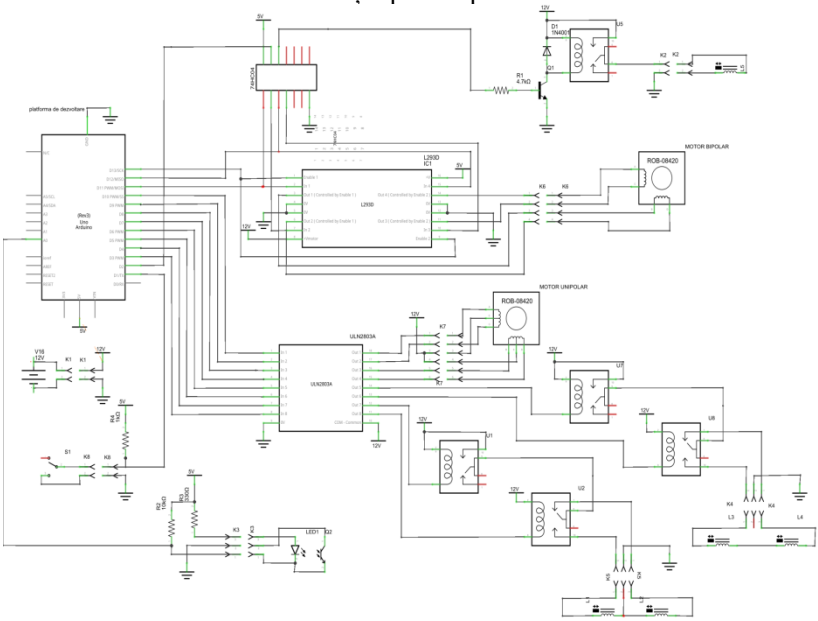
12.4.1 Mersul lucrării (Partea a II-a)

1. Se vor analiza componentele utilizate și schema electrica generala (anexa 1B)
2. Utilizându-se mediul de programare Arduino Galileo IDE se vor scrie proceduri pentru operarea principalelor componente ale sistemului:

a. Deschidere/închidere ușa principala
b. Deschidere/închidere ușa garaj
c. Deschidere/Închidere fereastra
d. Verificare cod de acces (în lucru)
e. Pornire/Oprire pompa și încălzire
f. Pornire/oprire ventilație
g. Reglare temperatura interioara
h. Detecție ferestre deschise
i. Închidere fereastra în caz de ploaie
j. Dispecerizarea comenzilor primite de la utilizator și vizualizarea pașilor executați

3. Se va scrie un program care lucrează în regim continuu și concurent
4. Se va scrie un program ce permite controlul clădirii de la distanță prin Internet (aplicație web-server)

Anexa 1A. Schema interfeței pentru procesul industrial



Anexa 1B. Schema interfeței pentru casa inteligenta