Comanda Sistemelor Industriale Integrate

Titular curs:

Conf. dr. ing. Valeriu BOSTAN

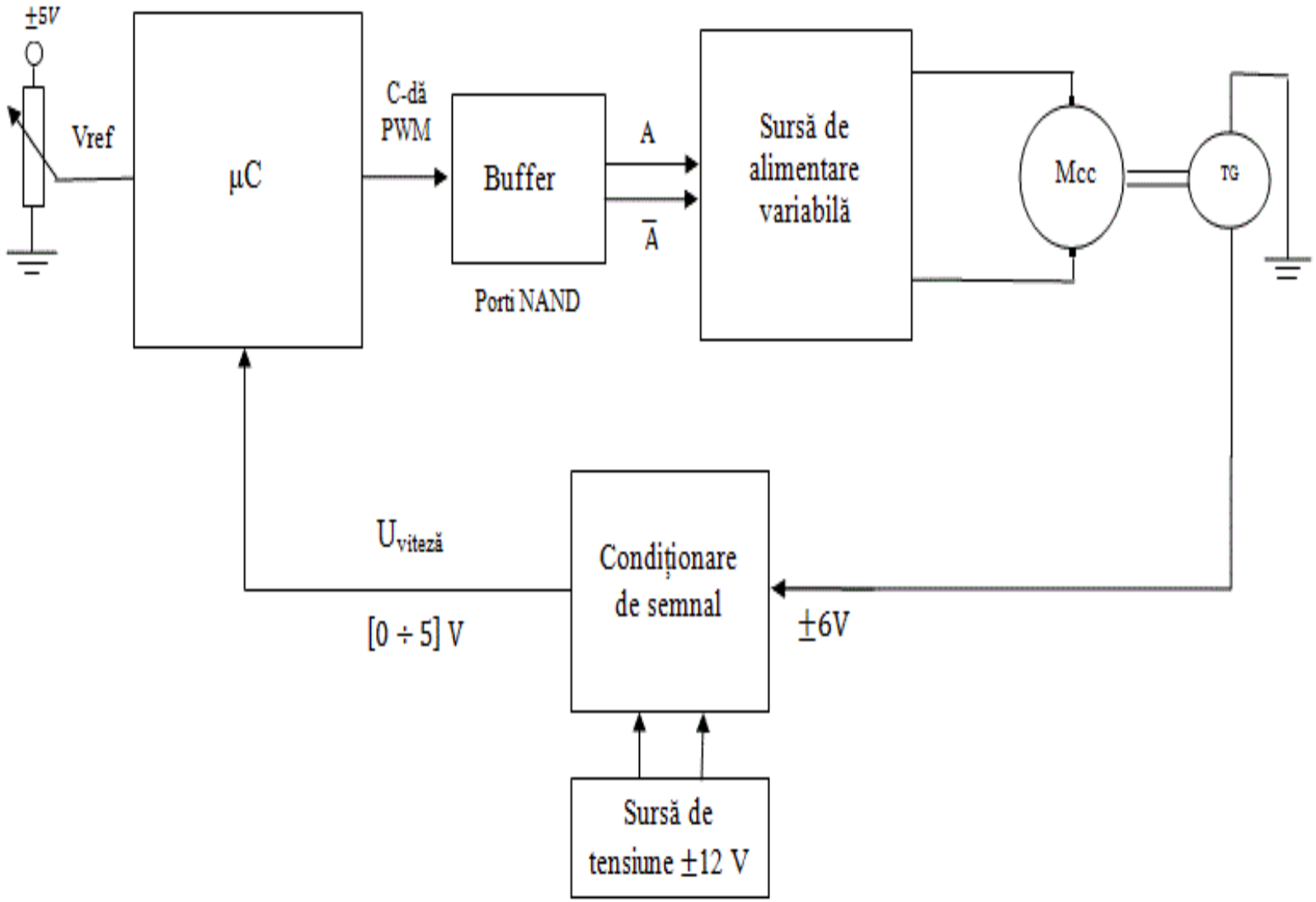
Sisteme integrate - “Embedded systems”

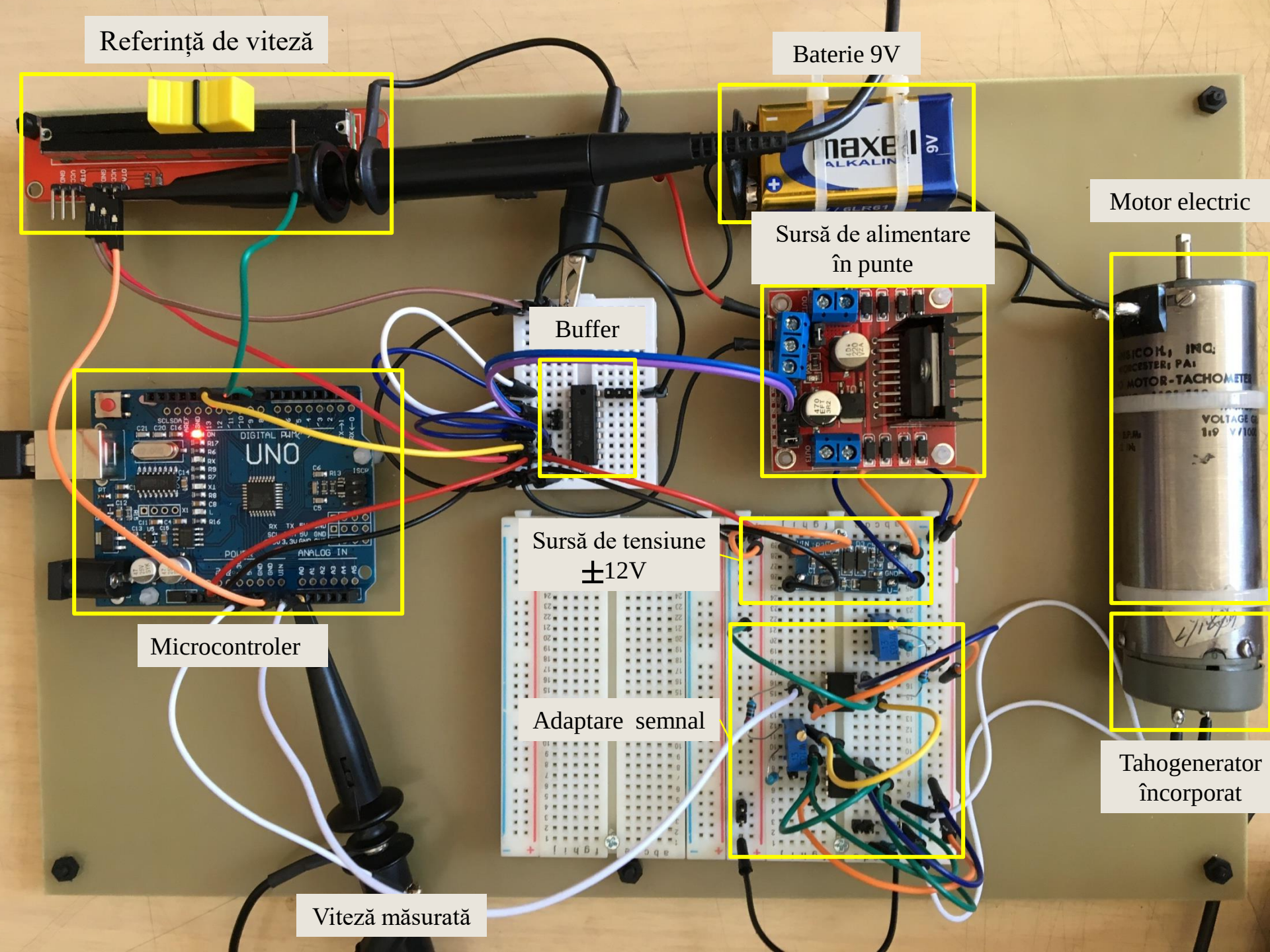
- **Sistem integrat – definiție:**
  un ansamblu de elemente proiectat să îndeplinească un set de funcții bine definit ce are cel puțin un microprocesor (microcontroller, PLC,…);

- **Direcții de studiu în domeniul sistemelor integrate:**
  - arhitectură;
  - configurare;
  - **comandă;**
  - proiectare;

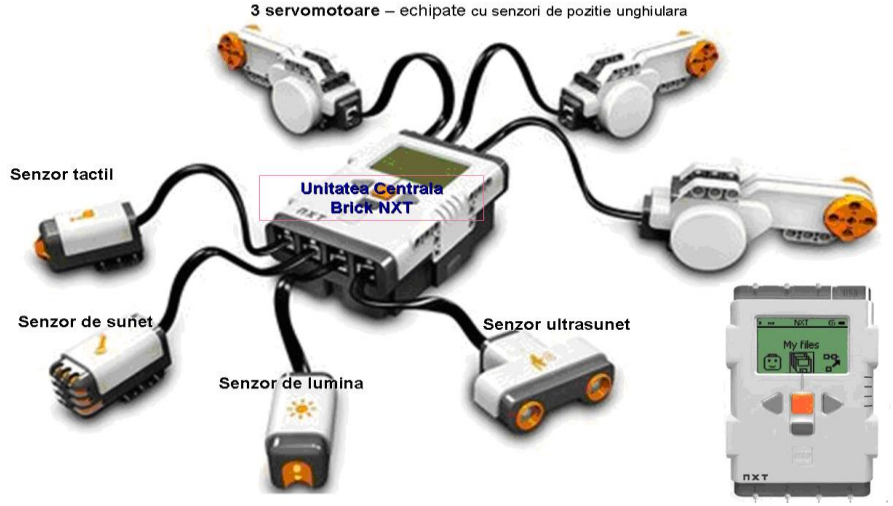
- **Exemple:**
  - linii de producție automatizate/robotizate;
  - **comanda numerică a motoarelor electrice;**
  - echipamente medicale;
  - automobile;
  - aplicații în domeniul militar;
  - telecomunicații;
  - …

Exemplul 1: Sistem de comandă pentru un servomotor de c.c.




Referință de viteză
Baterie 9V
Motor electric
Sursă de alimentare în punte
Buffer
Sursă de tensiune ±12V
Microcontroler
Adaptare semnal
Tahogenerator încorporat
Viteză măsurată

Exemplul 2: Sistemul “MINDSTORM NXT”



- Software:

  - NXC - (*Not eXactly C*);

  - MATLAB/SIMULINK - toolbox MINDSTORMS NXT (*Remote & Embedded Control*);

  LabVIEW - toolkit-ul MINDSTORMS NXT;

- Interfață:

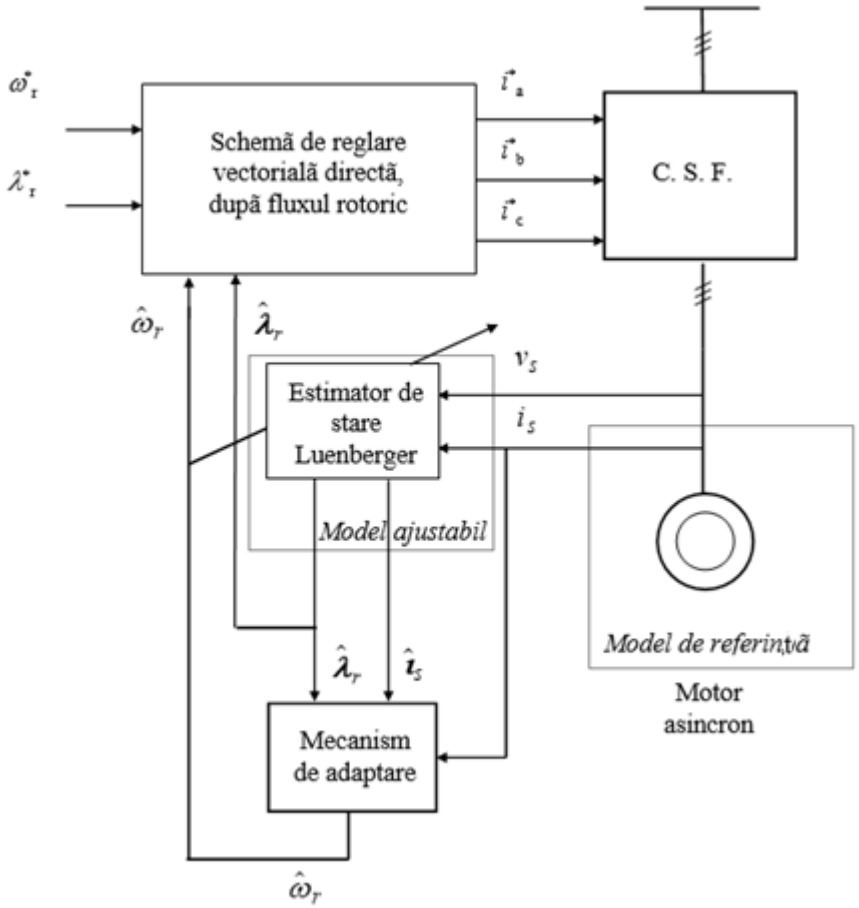
  - USB/Bluetooth/I2C (Inter-Integrated Circuit);

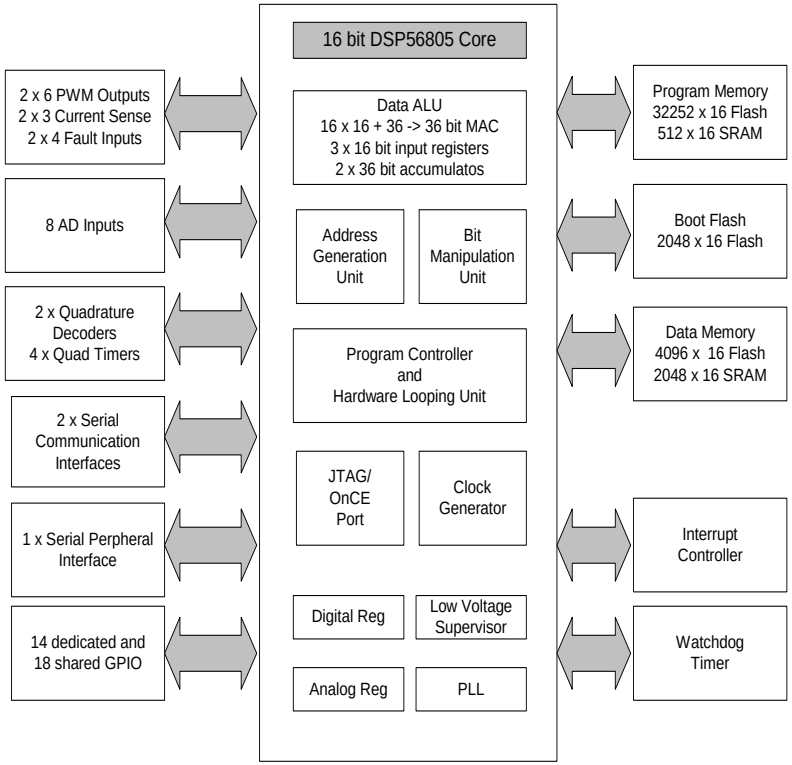
  - Butoane/Display;

- Senzori: de ultrasunete/de contact/optic/acustic;

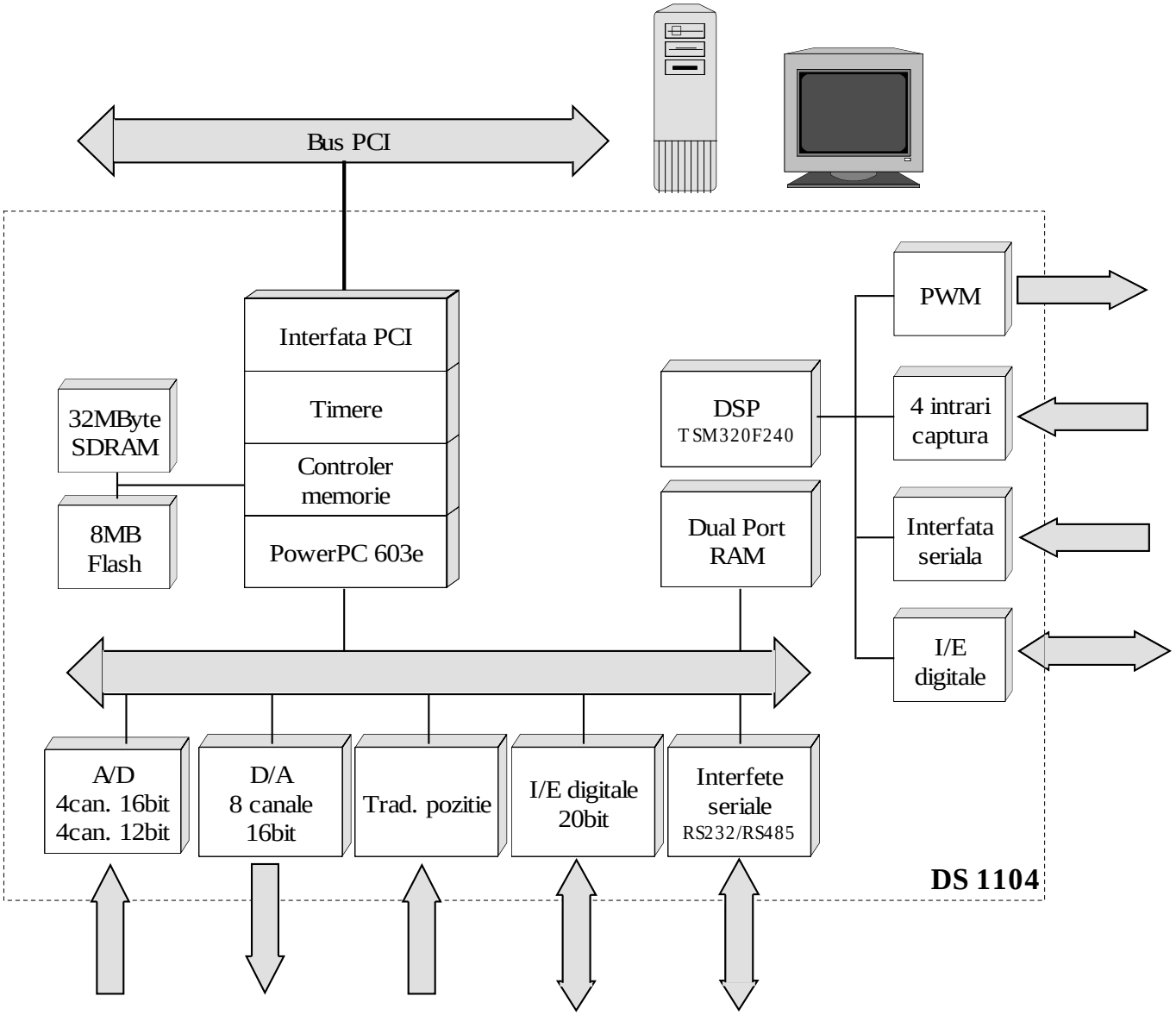
- Actuatoare: - 3 servomotoare de cc, cu reductoare și encodere;

Exemplul 3: Sistem de comandă pentru un motor asincron cu microprocesor pe 16 biți Motorola 56F805





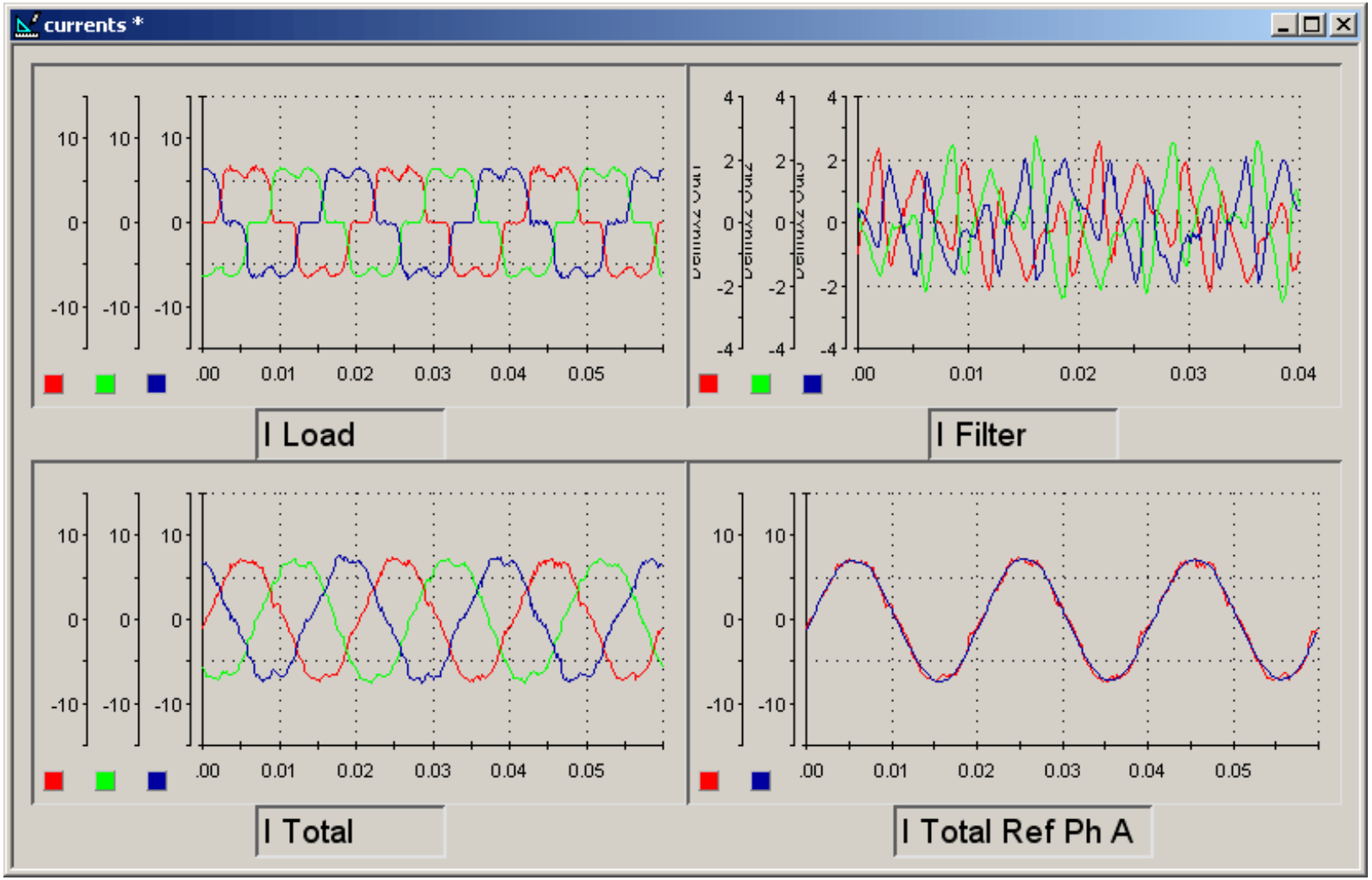
Exemplul 4: Sistemul “dSPACE” – arhitectura internă



Exemplul 4: Sistem industrial pentru compensarea armonicilor de curent controlat cu “dSPACE”

Exemplul 4: Sistemul “dSPACE” – “ControlDesk”

“ControlDesk” este o interfaţă grafică între sistemul de comandă în timp real şi PC-ul gazdă. Permite modificarea parametrilor în timpul funcţionării si vizualizarea mărimilor din sistem:



Particularități ale sistemelor industriale de comandă

Particularități ale sistemelor industriale de comandă

1. Sistemul de comandă operează *"în timp real";*

- se utilizează sistemul de întreruperi al microprocesorului;
- în modelarea unor astfel de sisteme se pot utiliza **blocurile cu trigger**;

2. Procesul pe care-l controlăm poate fi *nelinear* sau cu *parametrii variabili;*

- în modelarea acestor sisteme se pot folosi **s-Functions** scrise **în limbaj Matlab** sau **în C**;

3. Sistemele de comandă folosesc microprocesoare *"în virgulă fixă";*

- microprocesoarele sau microcontrolerele industriale sunt în general pe 8, 16 sau 32 biți. Din acest motiv calculele sunt realizate cu precizie redusă;
- algoritmii de comandă folosiți în modelare vor utiliza numere întregi;

4. Programarea este specifică fiecărui echipament de comandă în parte;

- programarea microcontrolerelor din sistemele integrate se face în limbaj de asamblare specific sau în **C** dar ținand cont de particularitățile hardware;
- în anumite situații se poate realiza o conectare directă între Matlab-Simulink și echipamentul de comandă cu ajutorul toolbox-ului **Real Time Workshop**;

Principalele obiective ale disciplinei

- Prezentarea noțiunilor generale despre sisteme integrate (**Embedded Systems**): arhitectură, funcționare, programare;
- Analiza particularităților sistemelor de comandă industriale: funcționare în timp real, sisteme nelineare sau cu parametrii variabili, microprocesoare în **virgulă mobilă** (calcule cu numere reale) respectiv **în virgulă fixă** (calcule cu numere întregi);
- Descrierea unei metode specifice de modelare a sistemelor de comandă industriale în timp real utilizând programul Matlab/Simulink:
  - blocuri de comandă cu **trigger;**
  - modelarea sistemelor nelineare cu blocuri tip **s_Functions**;
  - calcule cu **numere întregi**;
- Conectarea programului Matlab-Simulink cu echipamente hardware externe pe baza toolbox-ului **Real-Time Workshop (RTW);**
- Metode de proiectare și modelare a sistemelor integrate folosind medii de dezvoltare specifice (**Proteus, TinkerCad, Power Systems**).

Comanda Sistemelor Industriale Integrate
(extras din Fișa Disciplinei)

Date despre disciplină:

- Titular curs (2h/săpt): Conf. Dr. Ing. **Valeriu BOSTAN**
- Titular aplicații (1h/săpt): Conf. Dr. Ing. **Valeriu BOSTAN**
  Conf. Dr. Ing. **Ana-Maria DUMITRESCU**

Conținut curs – partea I:

- Structura și funcționarea sistemelor de comandă integrate;
- Particularități ale sistemelor de comandă industriale;
- Etape ale proiectării părții de comandă a sistemelor industriale integrate;
- Sisteme nelineare si cu parametrii variabili. Modelarea sistemelor cu s_Functions scrise in limbaj Matlab sau in C;
- Modelarea și testarea sistemelor integrate de comandă în ansamblu:
  - utilizarea mediilor de proiectare TinkerCad și Simulink Power Systems;
- Analiza regulatoarelor PID utilizate în sistemele industriale de comandă;

Comanda Sistemelor Industriale Integrate
(extras din Fișa Disciplinei)

- Utilizarea Real Time Workshop în comanda sistemelor industriale
  - descriere Real Time Workshop (RTW) – "Simulink Coder"
  - conectare directă Matlab/Simulink – sistem de comanda integrat. Exemple.
- Particularitati privind implementarea sistemului de comandă
  - sisteme de comandă modelate cu blocuri cu trigger;
  - comanda de tip PWM numeric;

Conținut curs – partea II:

- Sisteme de monitorizare și control SCADA (Supervisory Control And Data Acquisition). Descriere generală, funcționare, avantaje.
- Componentele de bază ale sistemelor SCADA: senzori și elemente de executie, unități de telemetrie – "remote telemetry units" (RTU), unitățile centrale SCADA ("master units");
- Funcții specifice sistemelor SCADA;
- Comunicații industriale. Protocolul Modbus, Profibus/Fieldbus;

Comanda Sistemelor Industriale Integrate
(extras din Fișa Disciplinei)

Conținut laborator:

- Metode de modelare a sistemelor continue și discrete. Reprezentarea în spațiul stărilor și cu funcții de transfer;
- Modelarea sistemelor cu ajutorul blocurilor de tip s_Functions scrise în limbaj Matlab și in C;
- Proiectarea și modelarea sistemului de comandă pe baza modelului de referință;
- Sistem de comandă numeric modelat cu ajutorul blocurilor cu trigger;
- Particularități ale sistemelor de comandă cu procesoare în virgulă fixă;
- Real-Time Workshop (RTW): prelucrare, compilare și trimiterea codului executabil direct pe sistemul hardware;
- Sustinerea temei de casă – proiectarea și modelarea unui sistem integrat.

Comanda Sistemelor Industriale Integrate

(extras din Fișa Disciplinei)

Punctaj curs:

1) Colocviu final: 20 puncte

2) Verificare pe parcurs: 20 puncte

Punctaj laborator:

3) Media notelor de la referatele de laborator: 30 puncte

4) Notarea activității în timpul ședințelor de laborator: 10 puncte

Temă de casă: - 20 puncte

- proiectarea și modelarea unui sistem integrat
- se va folosi mediul de lucru TinkerCad

Anexă: extras din conținutul cursului și al laboratorului disciplinei:

Comanda Sistemelor Industriale Integrate

Sisteme de comandă în timp real

Pentru un sistem de comandă ce operează ***"în timp real"***, succesiunea principalelor etape este următoarea:

A – Achiziția datelor la momentul curent. Se citesc valorile de la CAN (mărimile măsurate și eventual referința)

CALC. – Calcule În această etapă se fac calculele conform algoritmului de reglare proiectat

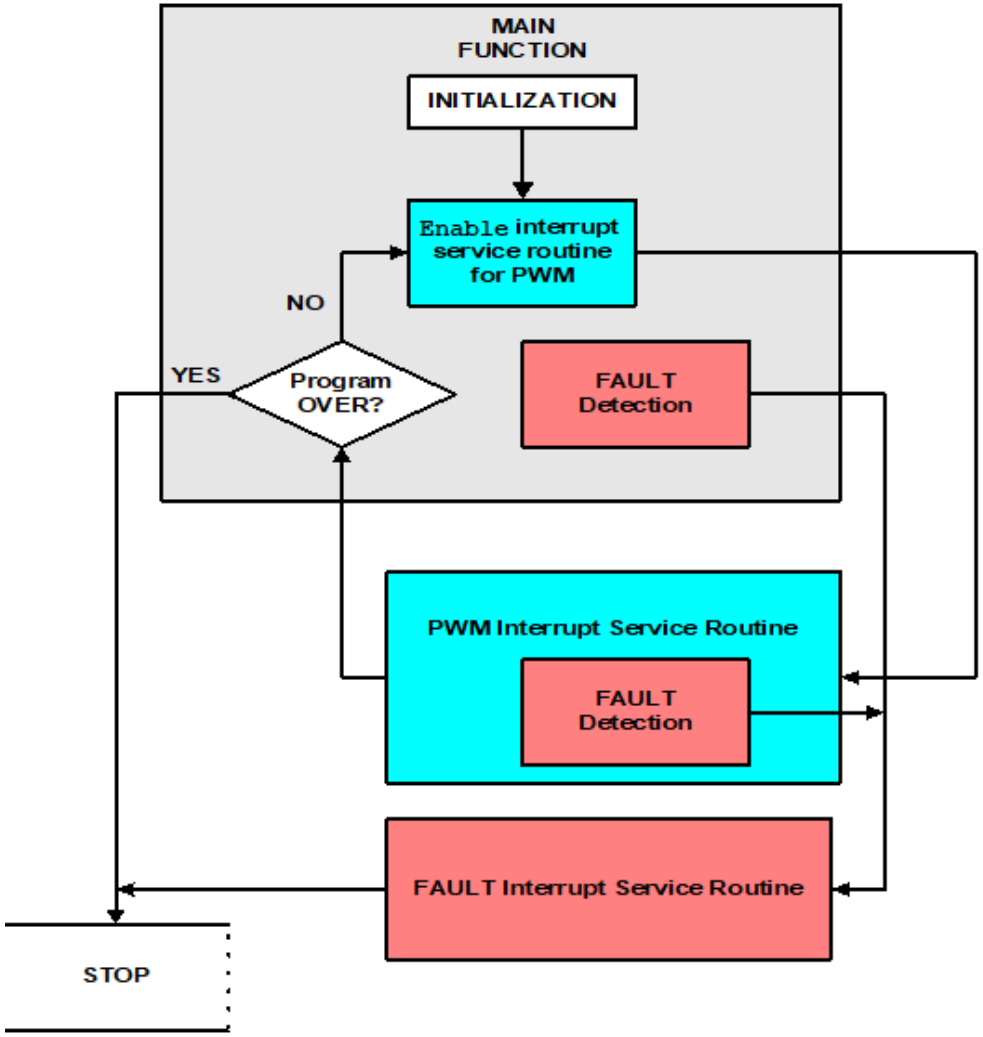
C-DA – Comandă În final, prin intermediul CNA, se furnizează către sistemul fizic comanda calculată anterior

Toate aceste etape se repetă la fiecare pas de eșantionare ***h***:

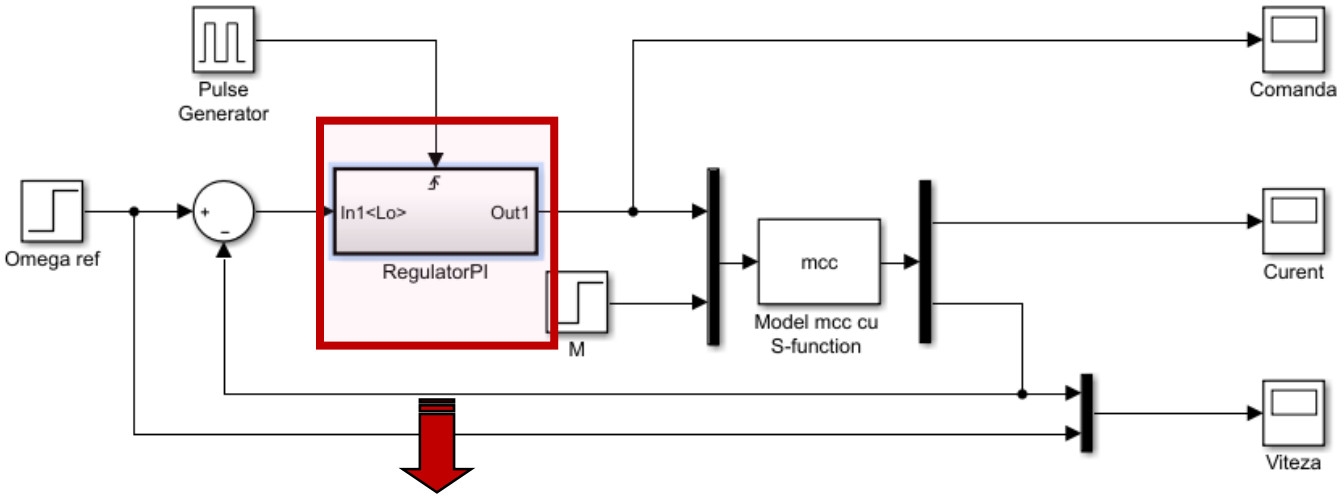
Momentele de timp ***h*** sunt generate cu ajutorul sistemului de întreruperi al microcontrolerului utilizat. Calculele se fac în rutina de tratare a întreruperii (**RTI - Real Time Interrupt**).

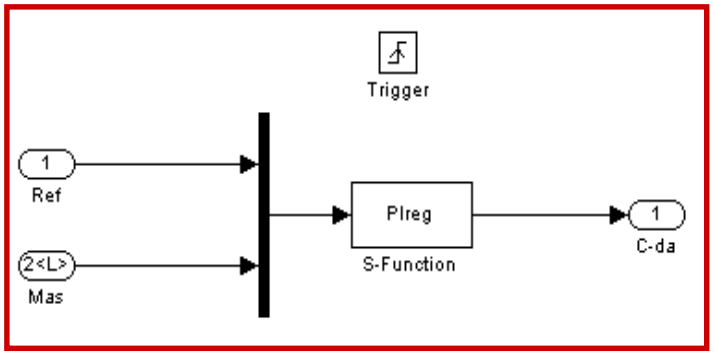
Sisteme de comandă în timp real

Organizarea programului folosind întreruperile:



Sistem de comandă modelat cu blocuri cu trigger





Varianta 1: Regulator PI implementat în Matlab

```
function cda  = fcn(err)

persistent I_ant;
    if isempty(I_ant)
        I_ant = 0;
    end;

h=100e-6;
kp=0.02;
ki=41.67*h;
sat=1.1*12;

% Calcul comanda
P=kp*err;
I=I_ant+ki*err;
cda = P+I;
```

```
% Saturatie regulator

if (cda>=sat)
    cda=sat;
    I=I_ant;
end
if (cda<=-sat)
    cda=-sat;
    I=I_ant;
End

I_ant=I;
```

Varianta 2: Regulator PI implementat cu *s_function* în C

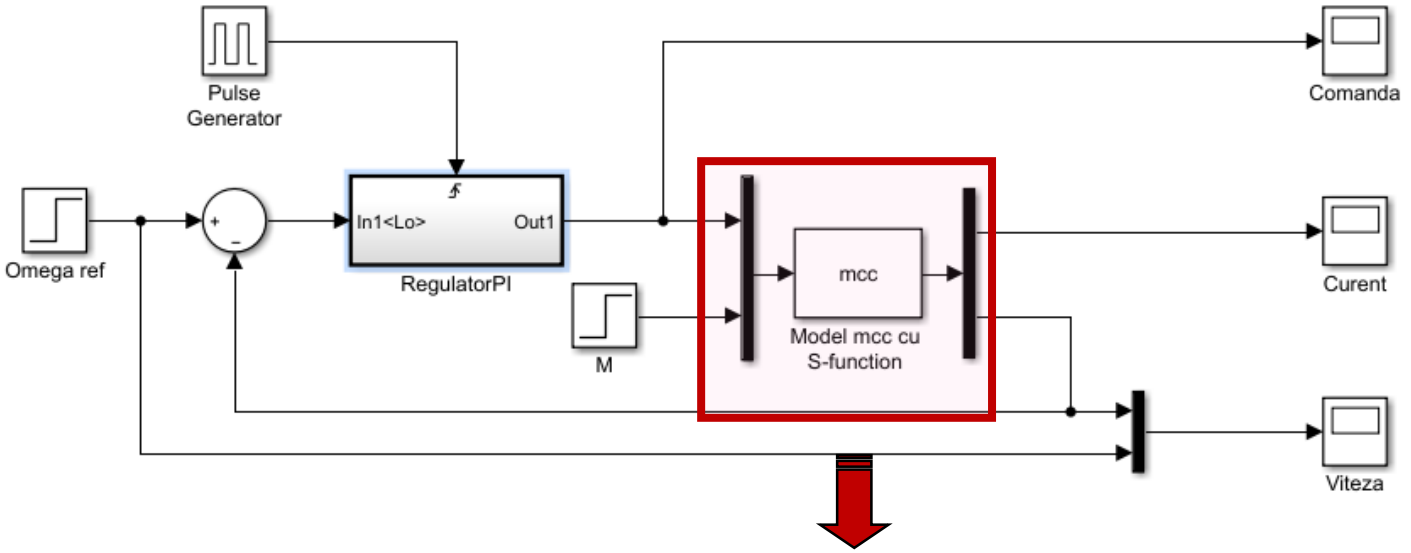
```
// Function: mdlInitializeSampleTimes =========================================
static void mdlInitializeSampleTimes(SimStruct *S)
{
   ssSetSampleTime(S, 0, -1);    // se pune -1 pentru ca lucreaza in trigger sistem!!!
   ssSetOffsetTime(S, 0, 0.0);
   ssSetModelReferenceSampleTimeDefaultInheritance(S);
}

#define MDL_INITIALIZE_CONDITIONS
// Function: mdlInitializeConditions ========================================
static void mdlInitializeConditions(SimStruct *S)
{
  Iold=0;
}

// Function: mdlOutputs =======================================================
//  y = Cx + Du
static void mdlOutputs(SimStruct *S, int_T tid)
{
   real_T            *y    = ssGetOutputPortRealSignal(S,0);
   InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

   err=U(0)-U(1);
   P=kp*err;
   I=ki*err+Iold;
   cmd=(P+I)/1000; // impartim la 1000 pentru a reveni la rezultatul corect, trunchiat la nr intregi

   // saturare comanda si partea integrala la valoarea anterioara
   if (cmd>sat) { cmd = sat; I=Iold; }
   if (cmd<-sat) { cmd = -sat; I=Iold; }
   y[0]=cmd;
   Iold=I;
 }
```

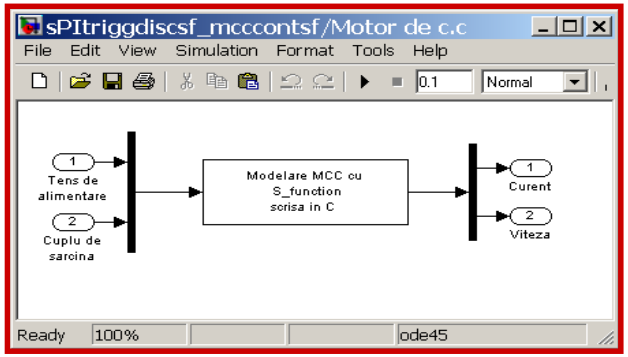
Varianta 3: Sisteme de comandă cu procesoare în virgulă fixă

```
function cda = fcn(ref,mas)

persistent Iold;
    if isempty(Iold)
        Iold=0;
    end;
u=12;

% coeficient de scalare
ks=1000;

%coeficienti regulator PI - scalati!
kp=int16(0.02*ks);
ki=int16(0.004167*ks);

%valoarea de saturatie
sat=int16(1.1*u);

%citire marimi intrare
referinta=int16(ref);
masura   =int16(mas);
```

```
% calcul regulator
eroare=int16(referinta-masura);

P=int16(kp*eroare);
I=int16(int16(ki*eroare)+int16(Iold));
comanda= int16((P+I)/ks);

%saturarea comenzii
if (comanda>sat) comanda = sat;
                 I=int16(Iold);
end;
if (comanda<-sat) comanda = -sat;
                  I=int16(Iold);
end;

% furnizare comanda la iesire
cda=double(comanda);

Iold=double(I);
```

Parametrii variabili - motor modelat cu blocuri tip *s_functions*





Exemplu: Modelare MCC cu *s_function* în C

```
// Function: mdlInitializeSampleTimes =========================
static void mdlInitializeSampleTimes(SimStruct *S)
{
   ssSetSampleTime(S, 0,  INHERITED_SAMPLE_TIME);
   ssSetOffsetTime(S, 0, 0.0);
   ssSetModelReferenceSampleTimeDefaultInheritance(S);
}

#define MDL_INITIALIZE_CONDITIONS
// Function: mdlInitializeConditions ==========================
static void mdlInitializeConditions(SimStruct *S)
{
   real_T   *x0 = ssGetContStates(S);

   x0[0]=0.0;
   x0[1]=0.0;
   R=10.8; // valoarea rezistentei la 20 grade C
}


// Function: mdlOutputs =======================================
static void mdlOutputs(SimStruct *S, int_T tid)
{
   real_T         *y   = ssGetOutputPortRealSignal(S,0);
   real_T         *x   = ssGetContStates(S);
   InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

   /* y=Cx+Du=x */
   y[0]=x[0];
   y[1]=x[1];
   y[2]=R; //doar pentru vizualizarea valorii lui R in exteriorul functiei
}
```

```
// Function: mdlOutputs =======================================
static void mdlOutputs(SimStruct *S, int_T tid)
{
   real_T         *y   = ssGetOutputPortRealSignal(S,0);
   real_T         *x   = ssGetContStates(S);
   InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

   /* y=Cx+Du=x */
   y[0]=x[0];
   y[1]=x[1];
   y[2]=R; //doar pentru vizualizarea valorii lui R in exteriorul functiei
}

#define MDL_DERIVATIVES
// Function: mdlDerivatives ===================================
//     xdot = Ax + Bu
 static void mdlDerivatives(SimStruct *S)
{
   real_T         *dx  = ssGetdX(S);
   real_T         *x   = ssGetContStates(S);
   InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

   dx[0] = (-R/L)*x[0]+(-K/L)*x[1]+(1/L)*U(0);
   dx[1] = (K/J)*x[0]+(-F/J)*x[1]+(-1/J)*U(1);
   R=R0*(1+alpha*U(2));  //variatia rezistentei cu variatia temperaturii
}

/* Function: mdlTerminate =====================================
*  Aceasta subrutina este obligatorie desi nu contine nimic.
*/
static void mdlTerminate(SimStruct *S)
{
}

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Etape ale proiectării părții de comandă a sistemelor integrate

Etapa 1:

Alegerea modelului matematic adecvat aplicației și modelarea lui

Etapa 2:

Proiectarea părții de comandă (a regulatorelor)

Etapa 3:

Modelarea părții de comandă

Etapa 4:

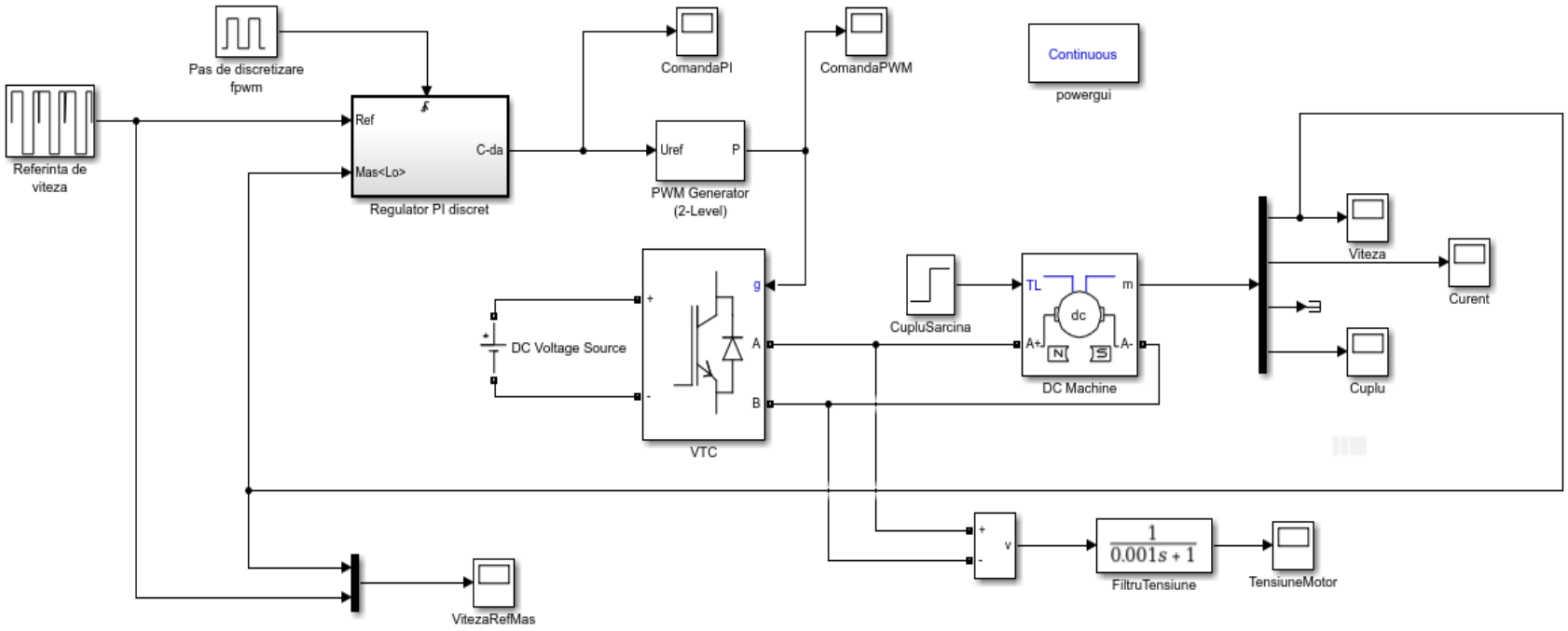
Modelarea sistemului integrat de comandă în ansamblu

Etapa 5:

Realizarea și testatea experimentală a sistemului

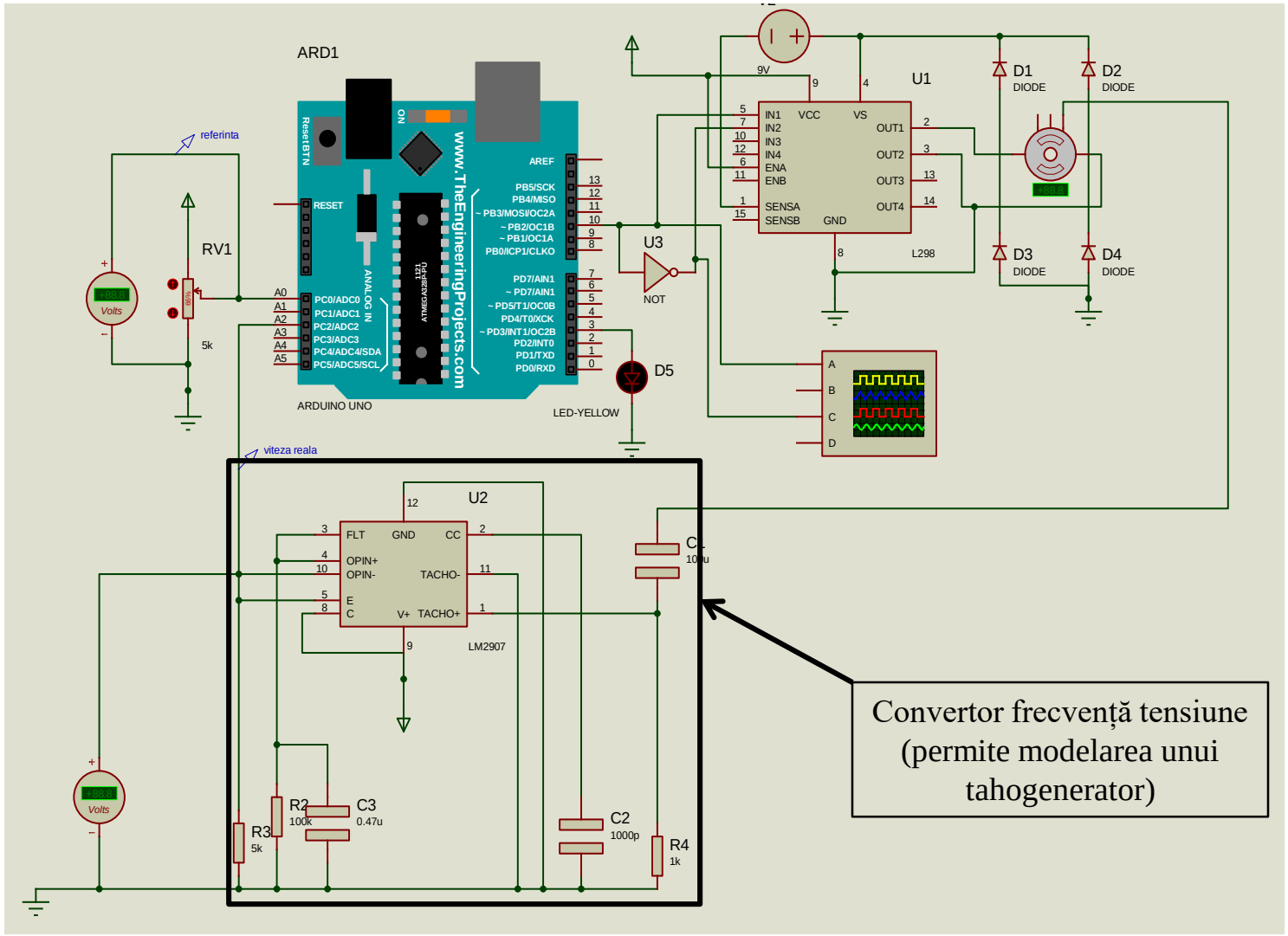
Etapa 4: Modelarea sistemului integrat de comandă în ansamblu

Varianta 1: Simscape/PowerSystems Toolbox



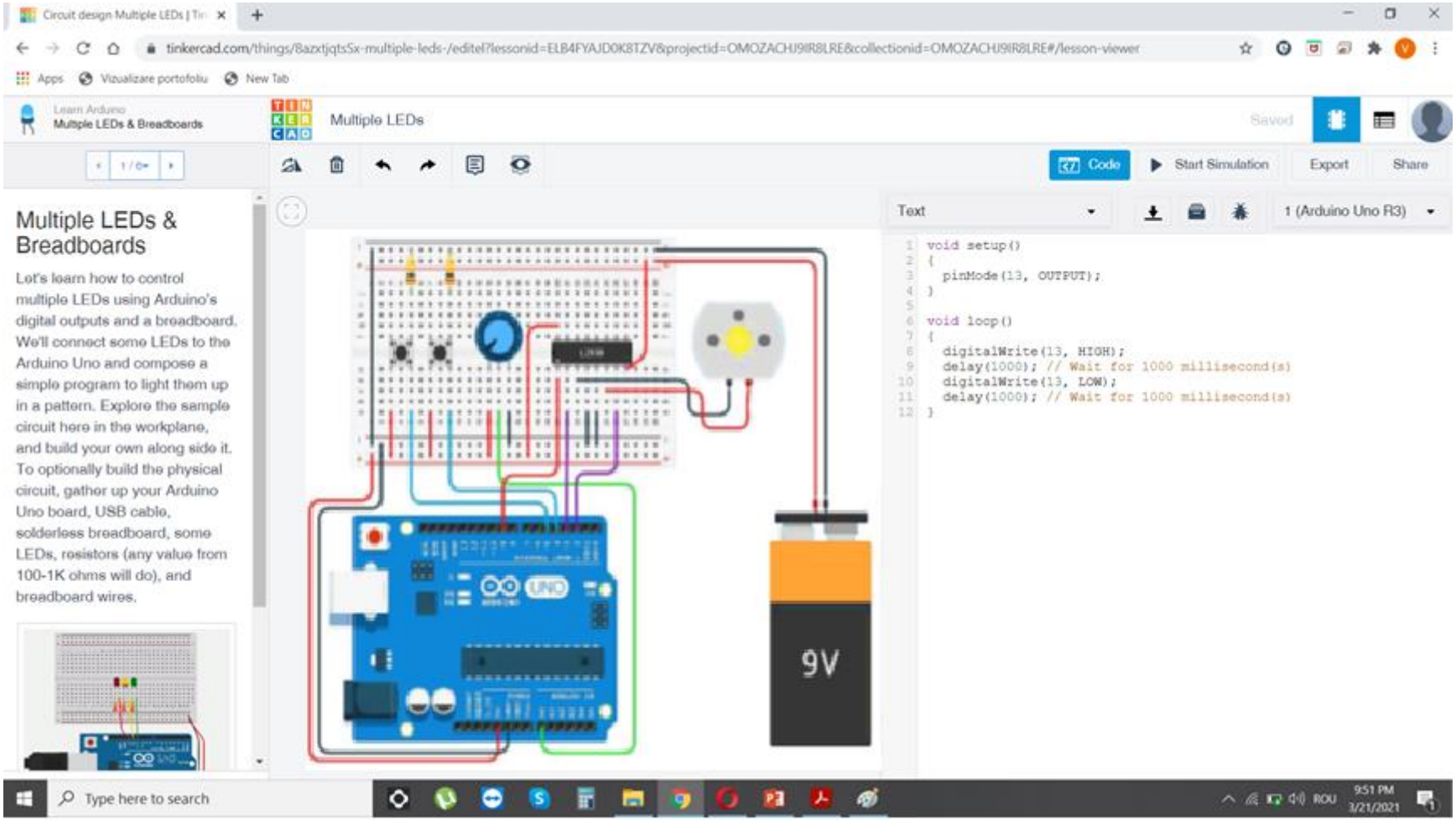
Etapa 4: Modelarea sistemului integrat de comandă în ansamblu

Varianta 2: Mediul de dezvoltare **Proteus**



Etapa 4: Modelarea sistemului integrat de comandă în ansamblu

Varianta 3: mediul de dezvoltare **TinkerCAD**



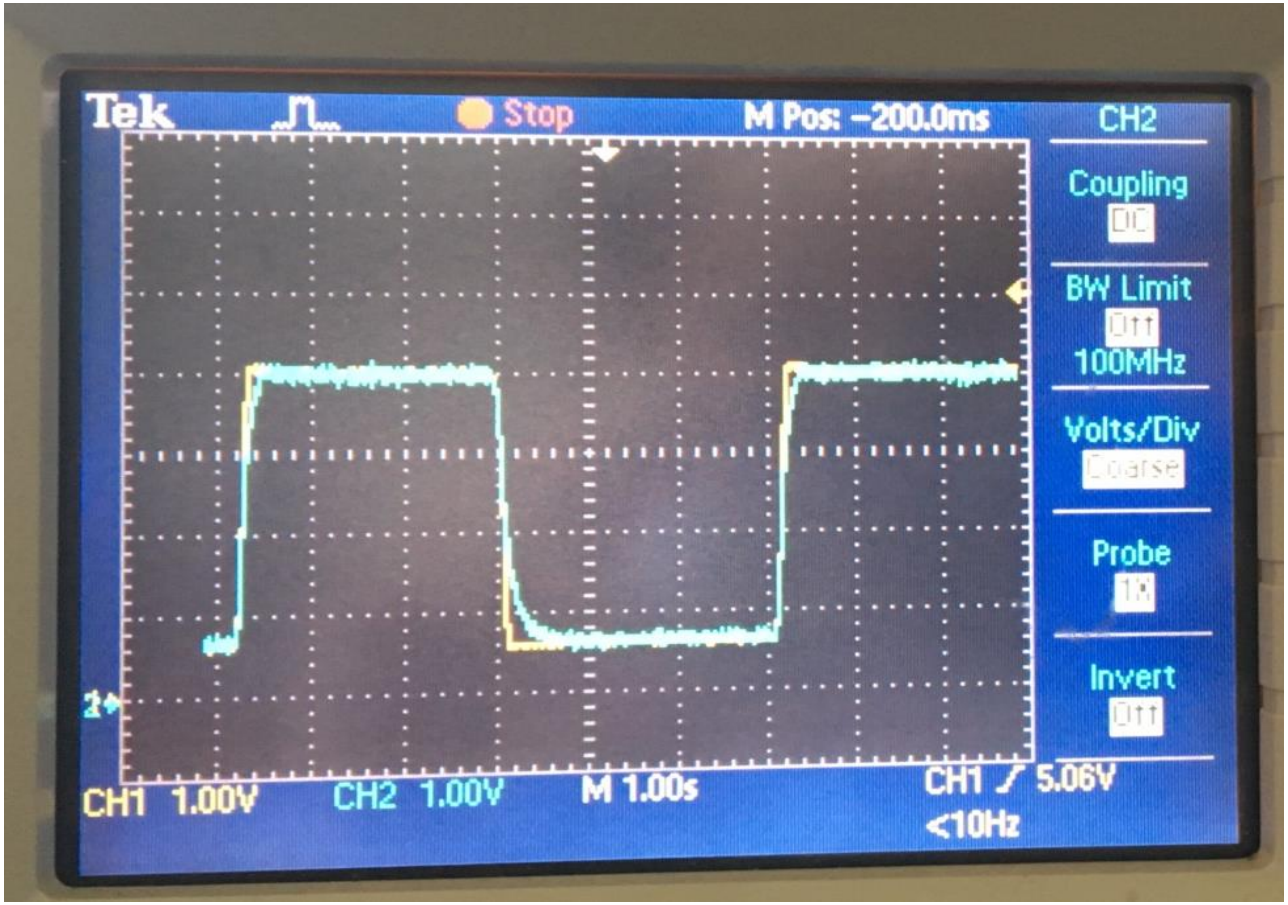
Etapa 5: Realizarea și testarea experimentală a sistemului

Regulator PI implementat pe un procesor Atmel (numere întregi pe 16 biți)

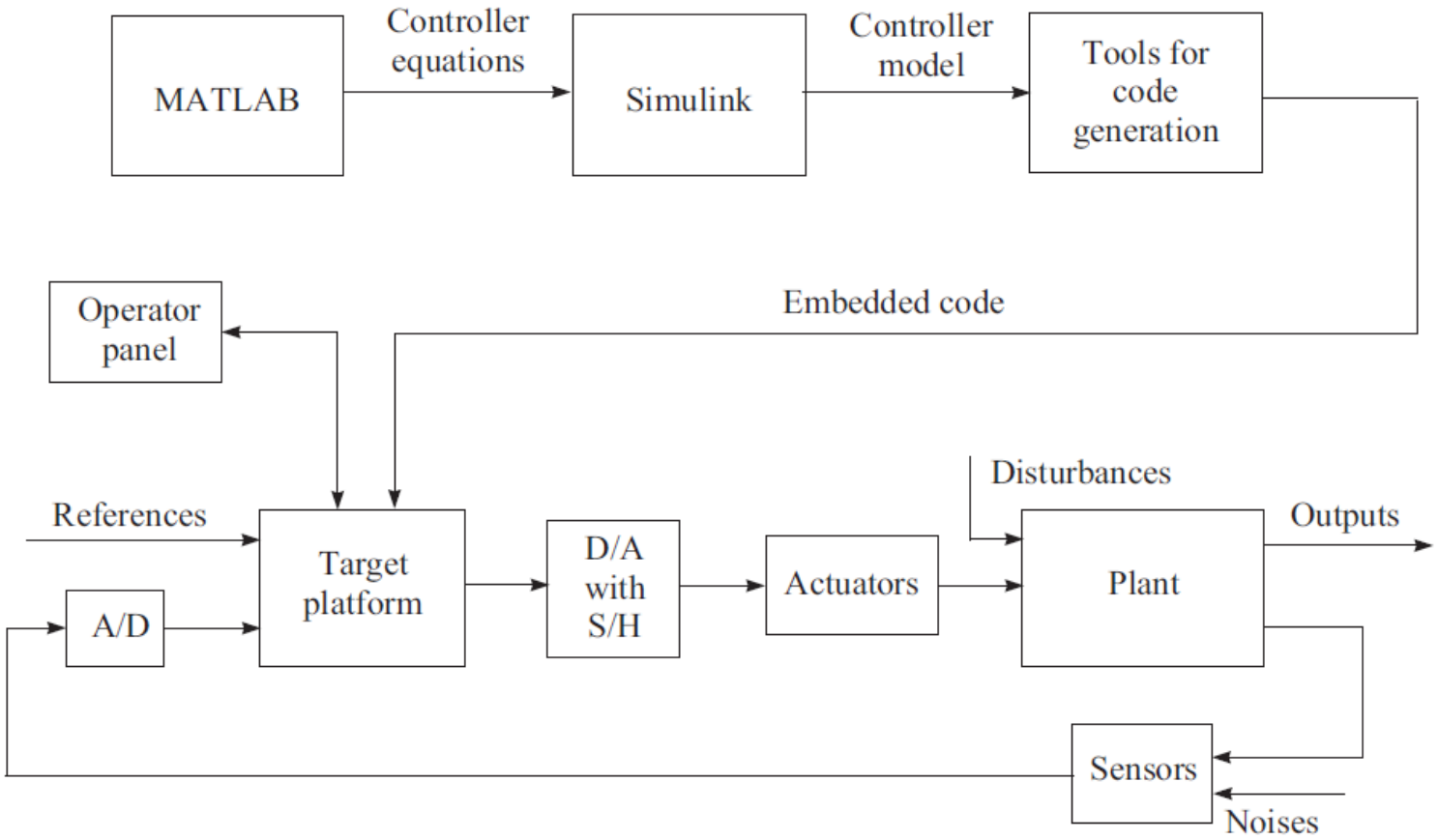
```
ISR(TIMER1_COMPA_vect)
{
 // achiziție date de intrare
  viteza_referinta = analogRead(potPin);
  viteza_reala = analogRead(potTG);

// calcule – conform algoritmului de comandă
  eroare = viteza_referinta - viteza_reala;

  parte_prop = eroare*Kp;
  parte_integrala = parte_integrala_ant + eroare*Ki;

  output = parte_prop + parte_integrala;
  parte_integrala_ant = parte_integrala;

// saturație regulator
if(output > 5110){
        output = 5110;
        parte_integrala_ant = parte_integrala -
    Ki*eroare;
    }
    if(output<10){
      output = 10;
      parte_integrala_ant = parte_integrala - Ki*eroare;
    }
// comandă (pentru stabilirea factorului de umplere)
    OCR1B = output ;
 }
```

Etapa 5: Realizarea și testarea experimentală a sistemului

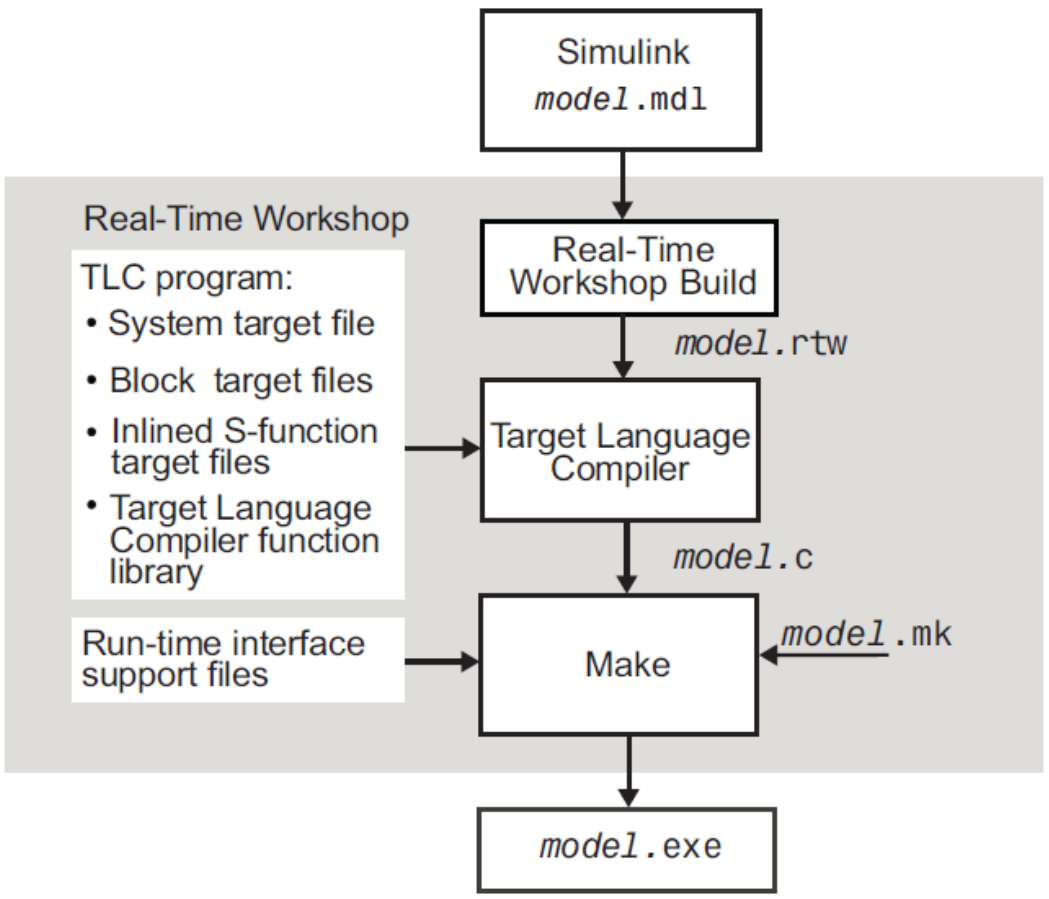
Funcționarea sistemului de comandă considerând reversări succesive:



Utilizarea Real-Time Workshop în comanda sistemelor integrate



Utilizarea Real-Time Workshop în comanda sistemelor integrate



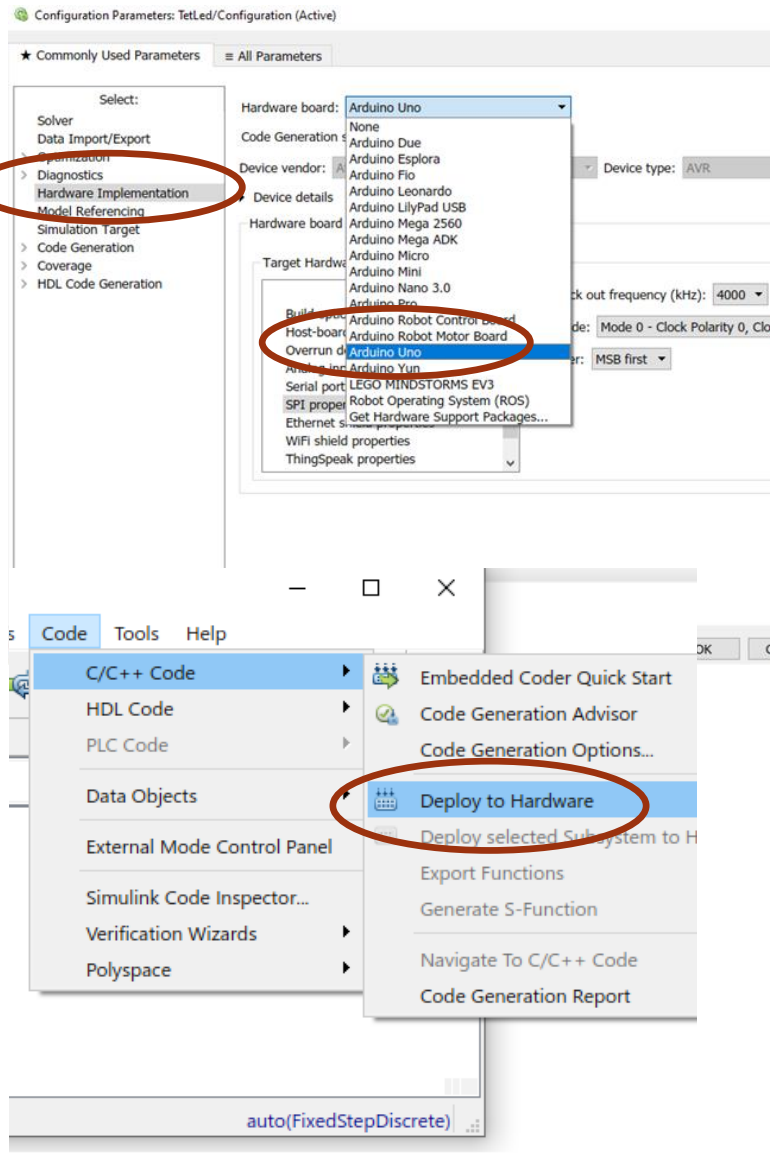
Real-Time Workshop (RTW) converteste fisierul de tip diagrama bloc Simulink ***model.mdl*** in ***model.rtw*** (o forma intermediara a diagramei bloc Simulink).

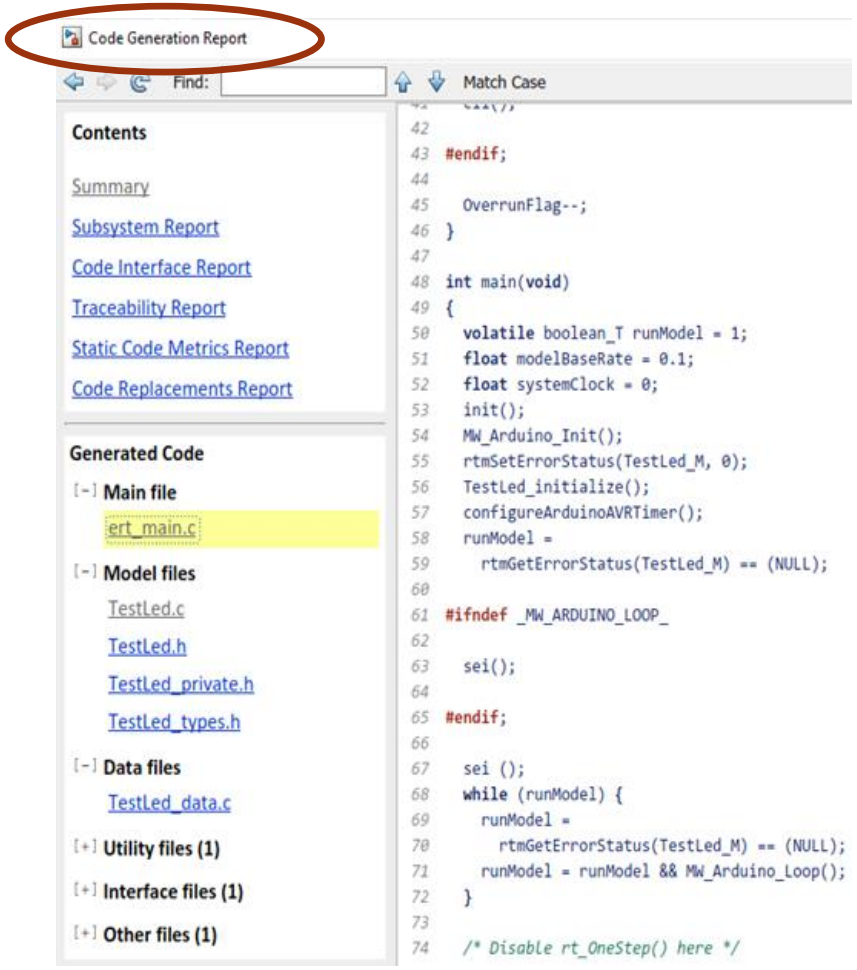
Target Language Compiler (TLC) converteste fisierul ***model.rtw*** in cod C ***model.c.***

Make construieste fisierul executabil ***model.exe***.

Fisierul ***model.mk*** este un target makefile (specific aplicatiei hardware).

Utilizarea Real-Time Workshop în comanda sistemelor integrate





Partea a II-a:

Sisteme integrate pe scară largă

Sisteme SCADA

Sisteme de monitorizare și control

(**S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition)

Control Room Building

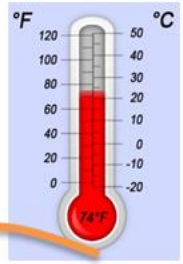
HMI (Human Machine Interface)

Plant

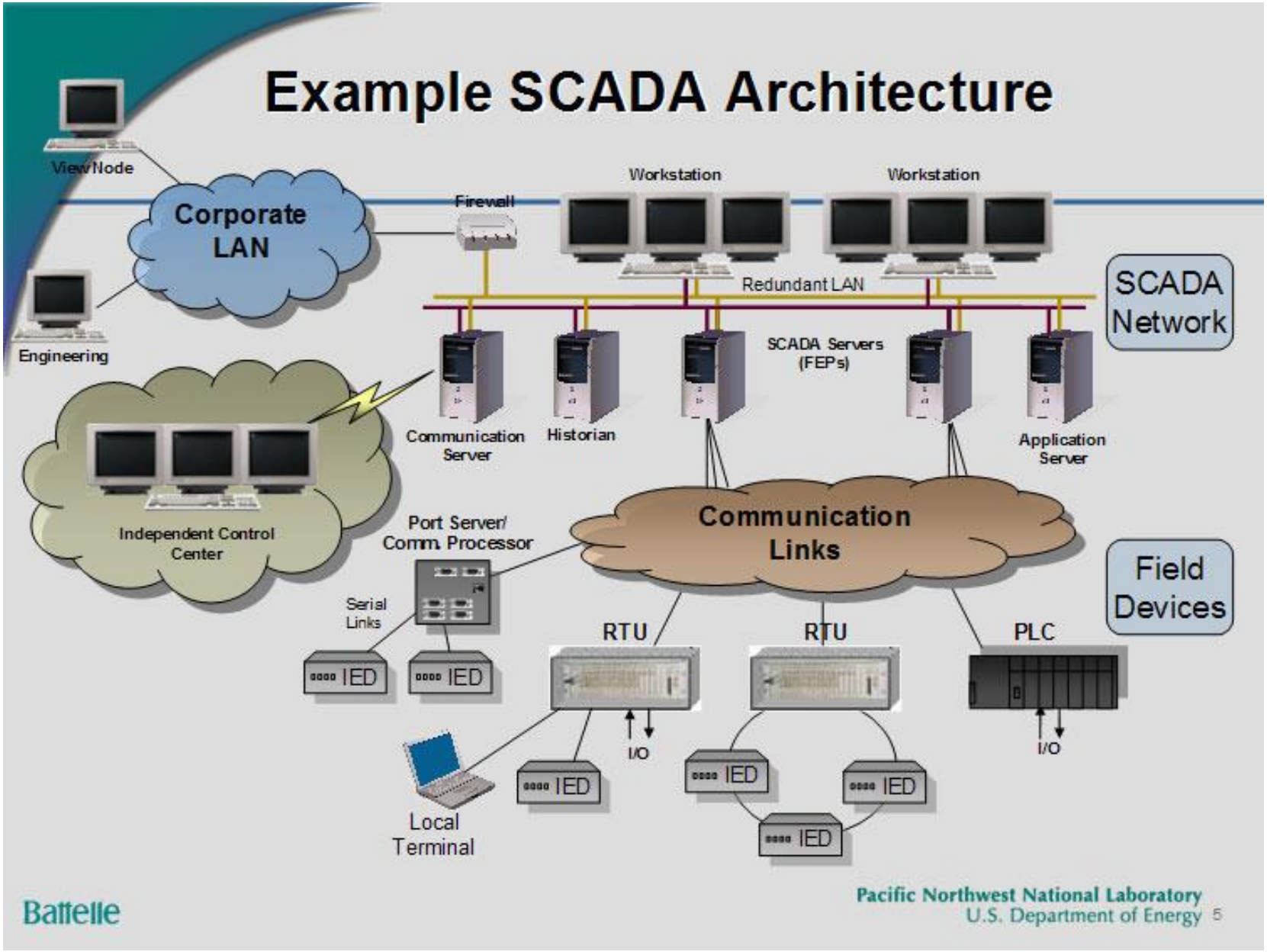
PLC
(Programmable
Logic Controllers)

Industrial Equipment

RTU
(Remote
Transmission
Unit)

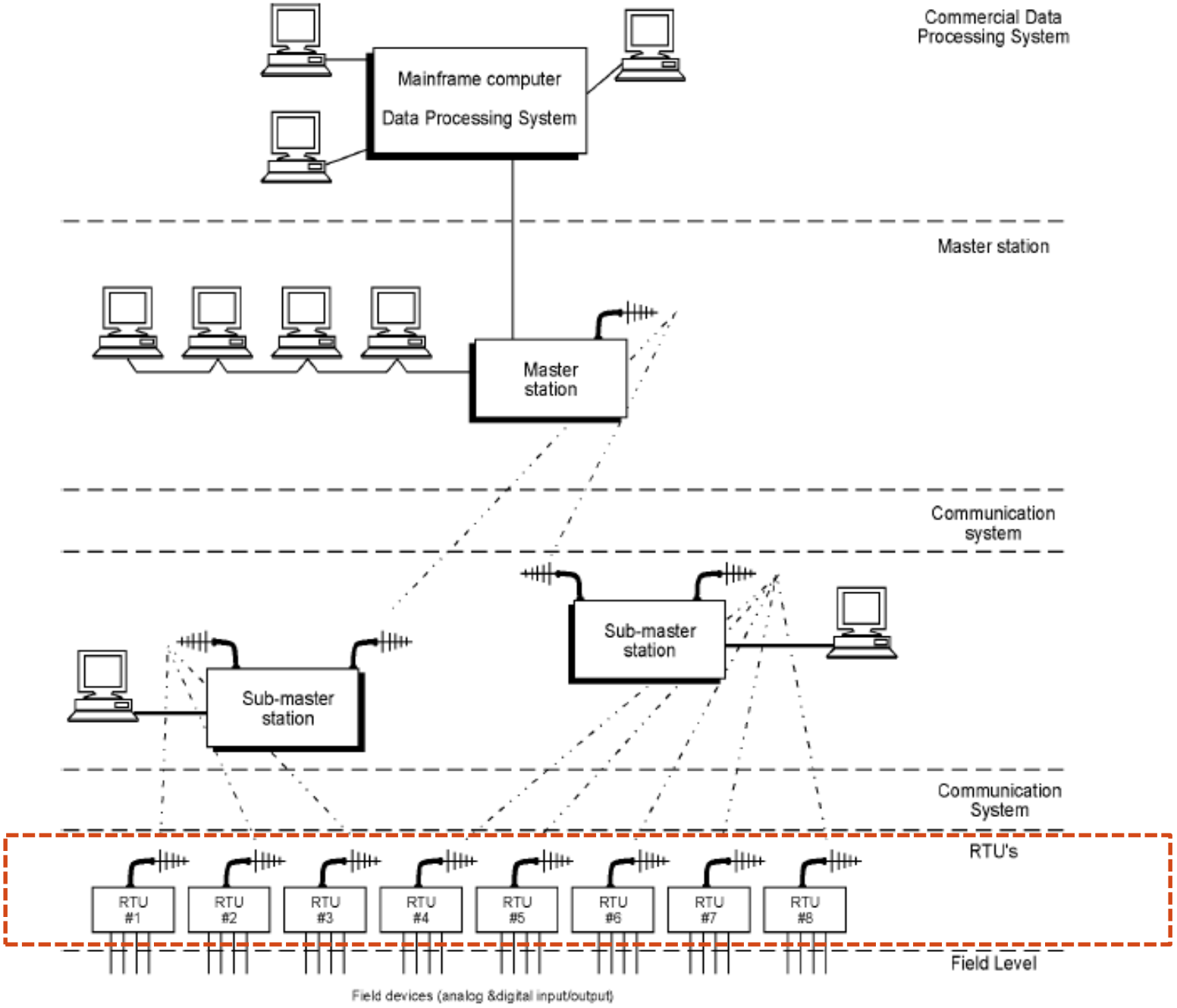


Temperature Sensor

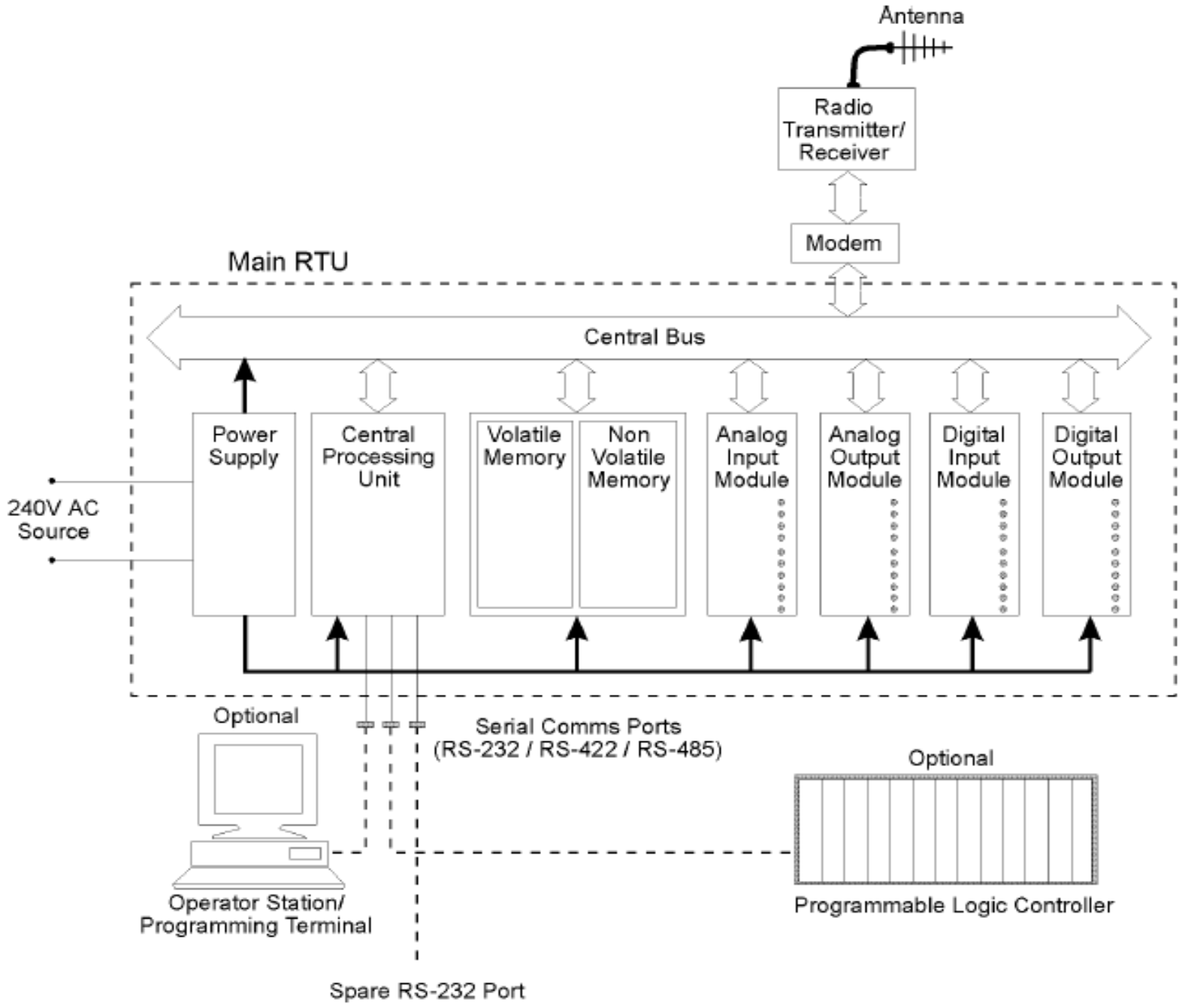


RTU = Remote Telemetry Unit
= Remote Transmission Unit

IED = Intelligent Electronic Device


Commercial Data Processing System
Mainframe computer
Data Processing System
Master station
Master station
Communication system
Sub-master station
Sub-master station
Communication System
RTU's
RTU #1
RTU #2
RTU #3
RTU #4
RTU #5
RTU #6
RTU #7
RTU #8
Field Level
Field devices (analog &digital input/output)


NI cRIO-9201


Antenna
Radio Transmitter/ Receiver
Modem
Main RTU
Central Bus
Power Supply
Central Processing Unit
Volatile Memory
Non Volatile Memory
Analog Input Module
Analog Output Module
Digital Input Module
Digital Output Module
240V AC Source
Optional
Serial Comms Ports (RS-232 / RS-422 / RS-485)
Optional
Operator Station/ Programming Terminal
Programmable Logic Controller
Spare RS-232 Port