Abordarea unei teme de examinare
din cadrul primului capitol – modul I de examinare –
„Sisteme de calcul în timp real bazate pe micro-controllere”

IPOTEZĂ: „Un anume beneficiar (din industria auto), face apel la „spiritul ingineresc” al studentului, impunând următoarea problemă de inginerie electrică: „Să se propună (în baza studiului **principiului de funcționare**) o soluție „inginerească” pentru un circuit electronic / ansamblu de circuite electronice, care va îndeplini rolul de **semnalizare luminoasă intermitentă** la **acționarea unui întreruptor fizic fără revenire mecanică.** Aplicația are ca și scop (indicat de către beneficiar): **implementarea unui sistem de semnalizare cu comandă electronică / digitală / programabilă și acționare prin releu, pentru un autoturism**, practic, găsirea unei **alternative digitale / programabile** pentru varianta **analogică** deja existentă (ex. bi-metalică), deoarece este necesară implementarea soluției „software” într-un computer de bord central. **Sursa luminoasă principală** de semnalizare este o **lampă cu filament / incandescență** acționată de un **releu electro-mecanic,** iar sursa luminoasă **secundară** este un **L.E.D. indicator de stare** situat în bordul autoturismului**.** Beneficiarul, specifică faptul că, are nevoie (pentru început) doar de **o singură parte din sistem** (ex. partea dreaptă).”

OBSERVAȚIE: În scopul abordării acestei cerințe, se vor avea în vedere următoarele aspecte:

-**conceperea, implementarea** și **manipularea** unui **aparat matematic**, prin care să se modeleze logica de comandă și control al procesului; **(se vor acorda 4 PUNCTE);**

-**monitorizarea stării întreruptorului de comandă fără revenire mecanică** (numeric + evoluție în timp); **(se vor acorda 0,5 PUNCTE);**

-**generarea semnalelor digitale** pentru **două mijloce de semnalizare luminoase** (diodă electro-luminiscentă – eng. L.E.D. - light emitting diode, și o lampă cu incandescență acționată prin intermediul unui releu electro-mecanic); **(se vor acorda 2 PUNCTE);**

-**utilizarea noțiunilor studiate** pe parcursul semestrului pentru **argumentarea solidă și pertinentă** a **fenomenelor fizice** implicate în acest proces; **(se vor acorda 2 PUNCTE);**

-**evidențierea domeniului de destinație și aplicabilitate** al S.C.T.R. (sistemului de calcul în timp real) implicat în acest proces. **(se vor acorda 0,5 PUNCTE);**

MENȚIUNI: 1. Se acordă **1 (un) punct din oficiu** în mod **necondiționat** pentru acest mod de examinare, din capitolul I - „Sisteme de calcul în timp real bazate pe micro-controllere”;

2. În vederea rezolvării cerinței propuse, studentul **poate** utiliza **documentația proprie** (adică noțiunile din caietul PERSONAL) și **TOATE resursele paginii Web:**
http://epe.utcluj.ro/index.php/sisteme-de-calcul-in-timp-real/

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

3. Se propune ca și **timp efectiv de lucru**: **o oră** plus **patru zeci de minute** (1 [h] + 40 [min.]).

METODĂ DE ABORDARE A CERINȚEI PROPUSE (pași de rezolvare):

În scopul rezolvării acestei probleme de inginerie electrică, se vor **identifica** (de către student) **toate cerințele efective** impuse de către „beneficiar”, anume:

I. IDENTIFICAREA CERINȚELOR ȘI RESURSELOR STABILITE DE BENEFICIAR:

1. Este **necesară TESTAREA** principiului de funcționare al unui sistem de semnalizare printr-o **simulare în timp real** cu **aparatura fizică atașată la un sistem de calcul în timp real.**

2. Elementele **fizice „hard”** propuse de către beneficiar sunt:
-întreruptorul clasic, mecanic, fără revenire;
-releul pentru interfațarea părții de forță;
-indicatorul L.E.D. din bord.

3. **Semnalele care intervin** în „logica de comandă” sunt:
-semnalul digital **furnizat** de către **întreruptorul mecanic fără revenire** (**INTRARE**);
-semnalul digital **intermitent necesar** pentru **comanda releului** (**IEȘIRE RELEU**);
-semnalul digital **intermitent necesar** pentru comanda **indicatorului L.E.D.** (**IEȘIRE L.E.D.**);

4. **Logica de comandă:**

a. La acționarea butonului / comutatorului **fără revenire mecanică** (adică **se auto-menține contactul închis în mod natural**, deci NU ESTE NEVOIE DE SCHEMĂ / LOGICĂ DE CONTROL PENTRU AUTO-REȚINERE SAU MONTAJ DE TIP „PUNERE LA SURSĂ” (eng. Pull-Up)), **se citește starea digitală a intrării la care este cuplat comutatorul / butonul**; Starea intrării digitale este preluată în caclculator (mediul Matlab – Simulink), după care, **se afișează „0” sau „1”** într-un un bloc de afișaj, apoi **se urmărește evoluția în timp a stării intrării digitale la care a fost cuplat comutatorul** prin intermediul unui osciloscop virtual;

b. În funcție de **starea comutatorului**, **se acționează sau nu** o **secvență de temporizare** sau **generator de tact / impuls** sau o **secvență repetitivă alternantă** între stările **„logic 0”** (eng. LOW) și **„logic 1”** (eng. HIGH);

c. **Semnalul rezultant** al **secvenței de temporizare**, va fi **direcționat înspre ieșirea digitală corespunzătoare „lămpii martor indicator de bord” (LED verde)**, apoi, **se va direcționa semnalul și** înspre **ieșirea digitală corepsunzătoare releului de control al lămpii incandescente de semnalizare**;

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

II. SPECIFICAȚIILE ȘI RESURSELE DISPONIBILE STUDENTULUI:

Fiind vorba despre o problemă de **interfațare a interacțiunii factorului uman** cu **algoritmul de control al procesul fizic**, este evident că, sistemul de calcul în timp real ales pentru a îndeplini această cerință va fi un **micro-controller** (conform tabel Lab. 6)**;**

1. Resurse **fizice** (hard):
-Platforma Arduino UNO / Intel Galileo (micro-controller);
-Releu **separat galvanic** (cu optocuplor intern și circuit de supresare / LOGICĂ INVERSATĂ);
-Lampă **incandescentă** U_N = 12 [V];
-Diodă electro-luminiscentă U_N = 3,3 ~ 5 [V];
-Întreruptor / comutator unipolar cu o singură traptă de comutație (eng. S.P.S.T.SW);
-Sursă de alimentare U_{IN} = 220 [V]; $U_{Ieș}$ = 12 [V];

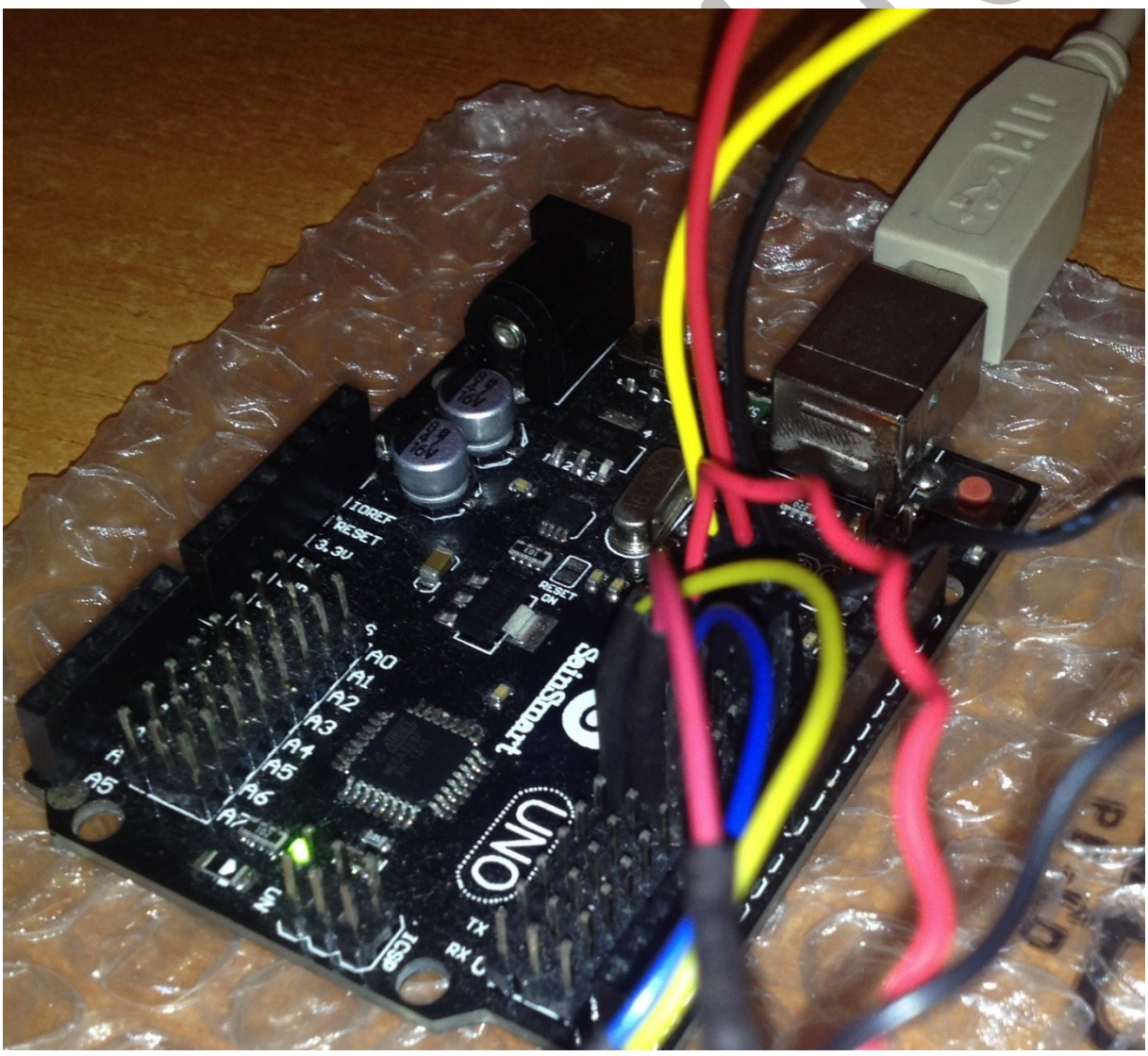


fig. 1 - Platforma Arduino
UNO / SainSmart (micro-controller)

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

fig. 2 - Releu separat galvanic

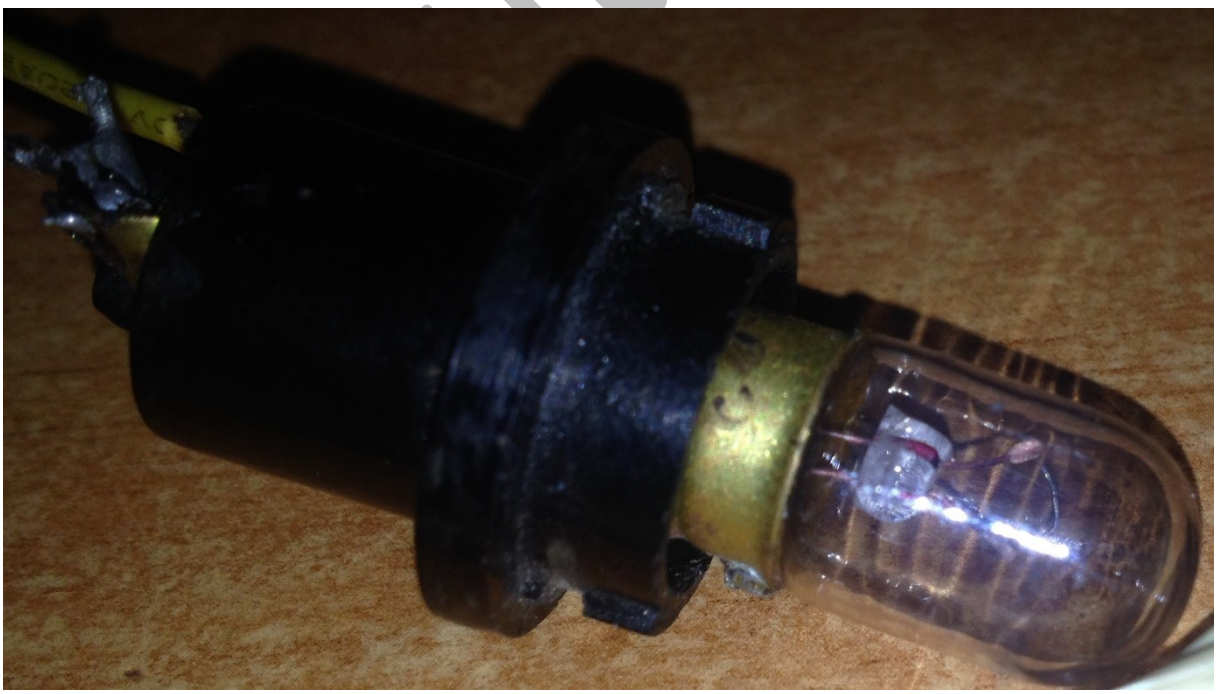


fig. 3 - Lampă incandescentă U_N = 12 [V]

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

fig. 4 - Diodă electro-luminiscentă U_N = 3,3 ~ 5 [V];

fig. 5 - Întreruptor / comutator
unipolar cu o singură traptă de comutație

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

fig. 6 - Sursă de alimentare U_{IN} = 220 [V]; $U_{Ieș}$ = 12 [V]

2. Resurse **logice** (soft):
-Mediul de **programare al platformei** Arduino – Arduino IDE;
-Rutina de **comunicare serial** dintre **platforma Arduino** și **mediul Matlab – Simulink**;
-Mediul de **simulare și testare** Matlab – Simulink;

3. **Documentație**: În vederea abordării acestui subiect, se vor avea în vedere materialele următoare, disponibile pe pagina materiei „Sisteme de calcul în timp real” anume:

- Schema electrică, laborator nr. 1 - „Interacțiunea cu ieșirile digitale în timp real” - (http://epe.utcluj.ro/SCTR/Scheme/SCTR_Lab1_DORTI_SCH.pdf);
- Exemplul de model PDF, laborator nr. 1 - „Interacțiunea cu ieșirile digitale în timp real” - (http://epe.utcluj.ro/SCTR/Model/SCTR_Lab1_DORTI_SL.pdf);
- Exemplul de model Simulink, laborator nr. 1 - „Interacțiunea cu ieșirile digitale în timp real” – (http://epe.utcluj.ro/SCTR/Model/SCTR_Lab1_DORTI_SL.slx);
- Schema electrică, laborator nr. 2 – „Interacțiunea cu intrările digitale în timp real” – (http://epe.utcluj.ro/SCTR/Scheme/SCTR_Lab2_DIRTI_SCH.pdf);

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- Principiul de funcționare al intrărilor digitale, model PDF, laborator nr. 2 „Interacțiunea cu intrările digitale în timp real” – (http://epe.utcluj.ro/SCTR/Model/SCTR_Lab2_DIWP.pdf);
- Principiul de funcționare al intrărilor digitale, model Simulink, laborator nr. 2 „Interacțiunea cu intrările digitale în timp real” – (http://epe.utcluj.ro/SCTR/Model/SCTR_Lab2_DIWP.slx);
- Exemplul de model PDF, laborator nr. 2 - „Interacțiunea cu intrările digitale în timp real” - (http://epe.utcluj.ro/SCTR/Model/SCTR_Lab2_DIRTI_SL.pdf);
- Exemplul de model Simulink, laborator nr. 2 - „Interacțiunea cu intrările digitale în timp real” - (http://epe.utcluj.ro/SCTR/Model/SCTR_Lab2_DIRTI_SL.slx);

III. MOD DE LUCRU:

A. SCHEMA ELECTRICĂ A MONTAJULUI:

Se va realiza montajul după următoarea schemă electrică:

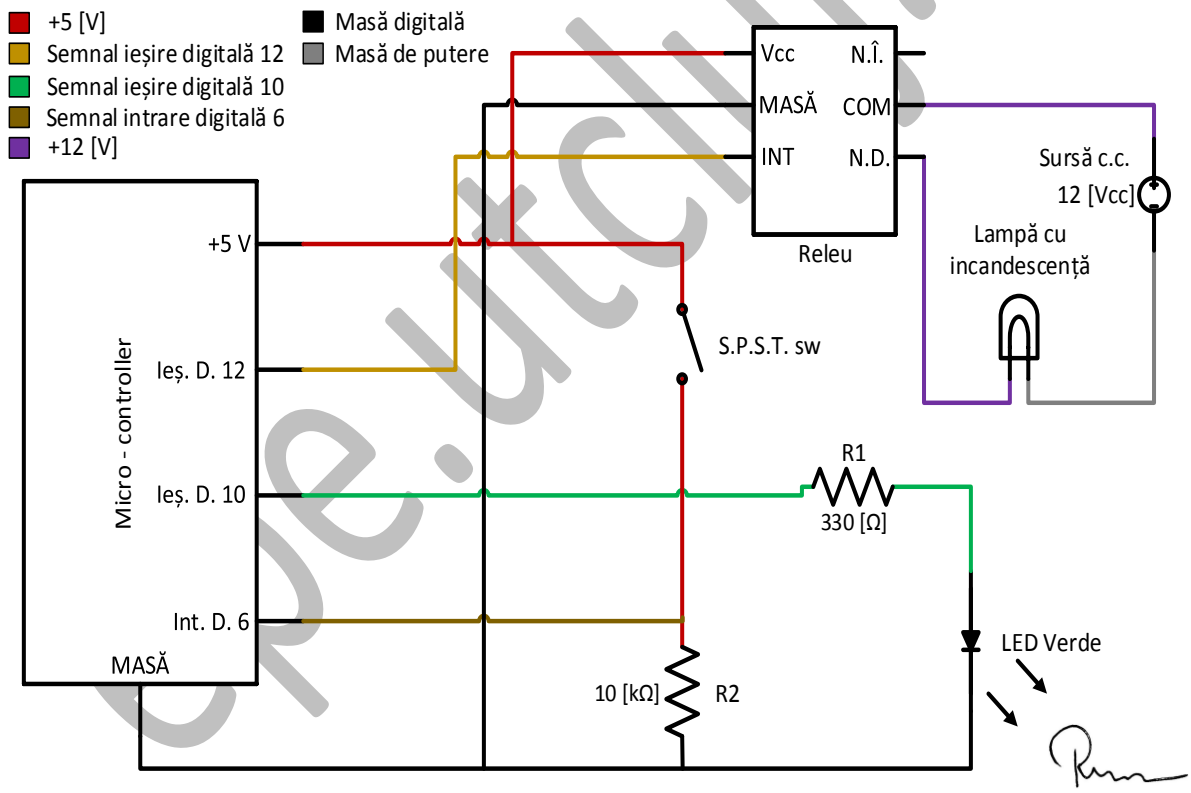


fig. 7 – Schema electrică a montajului propus
(Această schemă poate fi descărcată în format PDF, la rezoluție mai bună la adresa: http://epe.utcluj.ro/SCTR/Scheme/SCH_capI_microcontroller_SIGNED.pdf)

IMPORTANT: Într-un sistem de semnalizare, este necesară **acționarea „generatorului de tact” sau a releului de temporizare** prin **intermediul unui comutator** (maneta de semnalizare). Având în vedere faptul că, în cazul de față, modelul simulează **doar o**

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

singură direcție de semnalizare, se va alege un montaj cu **„punere la masă” (eng. pull – down)** pentru comutatorul unipolar, cu o singura treaptă de acționare. Astfel că:
-pentru SW1 = OPRIT – se va genera starea „logic 0”;
-pentru SW1 = PORNIT – se va genera starea „logic 1”;

B. SETĂRI INIȚIALE:

i. ÎNCĂRCAREA RUTINEI DE COMUNICARE CU MEDIUL MATLAB – SIMULINK:

1. Se va identifica **modelul platformei** cu microcontroller, după care **portul de comunicare serial** (ex. COMx unde „x” este numărul portului) din „Device Manager” (dacă platforma a fost instalată și atașată la calculator);

IMPORTANT: Pentru a accesa utilitarul „Device Manager”, se poate utiliza consola de comandă rapidă „Run”. Pentru a apela consola de comandă rapidă se apasă simultan combinația de taste „Windows + R” (⊞ + R). În consola, se va scrie comanda „devmgmt.msc” (fără ghilimele), însoțită de tasta „Enter”. În urma acestor acțiuni, se va deschide utilitarul „Device Manager”!

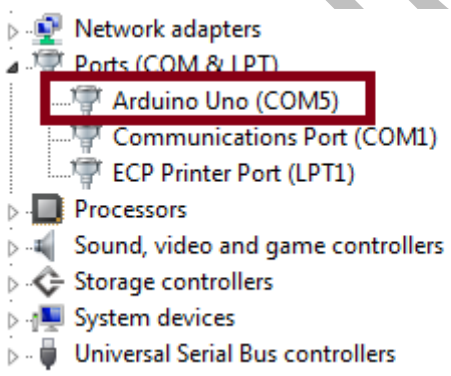


fig. 8 – Portul serial al platformei Arduino indicat în utilitarul „Device Manager”

2. Se va încărca programul **specific rutinei de comunicare** (a platformei Arduino cu mediul Matlab – Simulink) cu ajutorul **mediului de programare dedicat** pentru platforma Arduino, anume **Arduino IDE**;

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

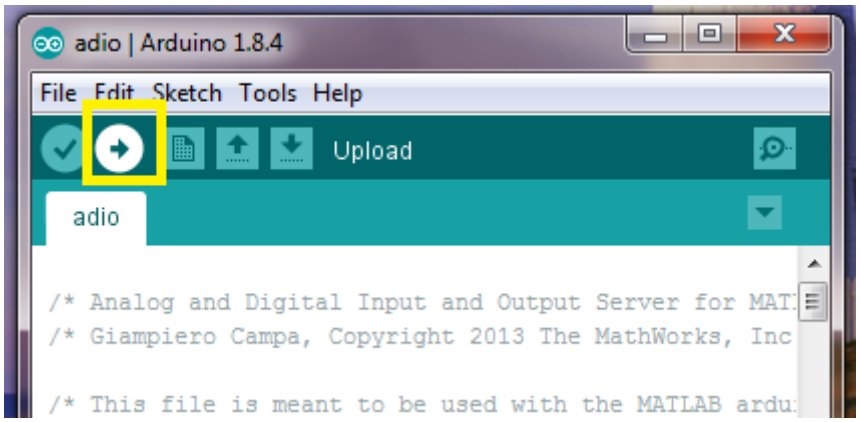


fig. 9 – Încărcarea rutinei de comunicare
din mediul de programare Arduino IDE

ii. SETĂRILE INIȚIALE DE SIMULARE ÎN MEDIUL MATLAB – SIMULINK:

1. Setarea **directorului de lucru** pentru mediul Matlab – Simulink este primul pas necesar, după lansarea în execuție a mediului Matlab. În vederea realizării acestui lucru, **se va creea un director nou** în spțiul de lucru (eng. Desktop) cu numele dorit (ex. SCTR_LAB), iar în mediul Matlab, se vor folosi **controalele grafice de navigare**, **structura arboriscentă** și **bara de specificare a căii**, pentru a alege calea (eng. path) directorului de lucru;

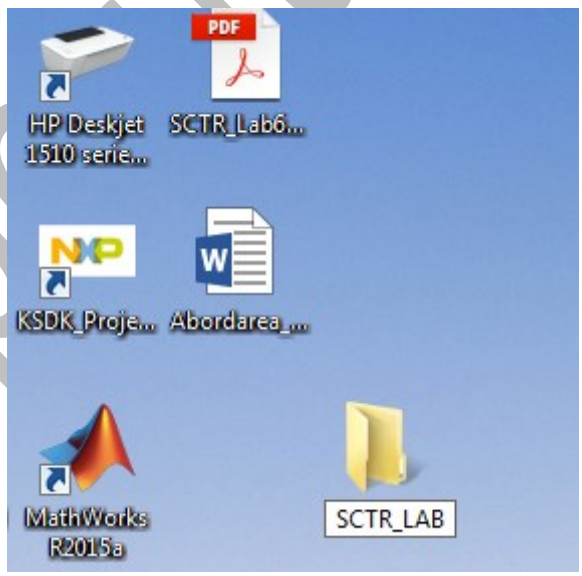


fig. 10 – Crearea unui director
(eng. folder) pe spațiul de lucru

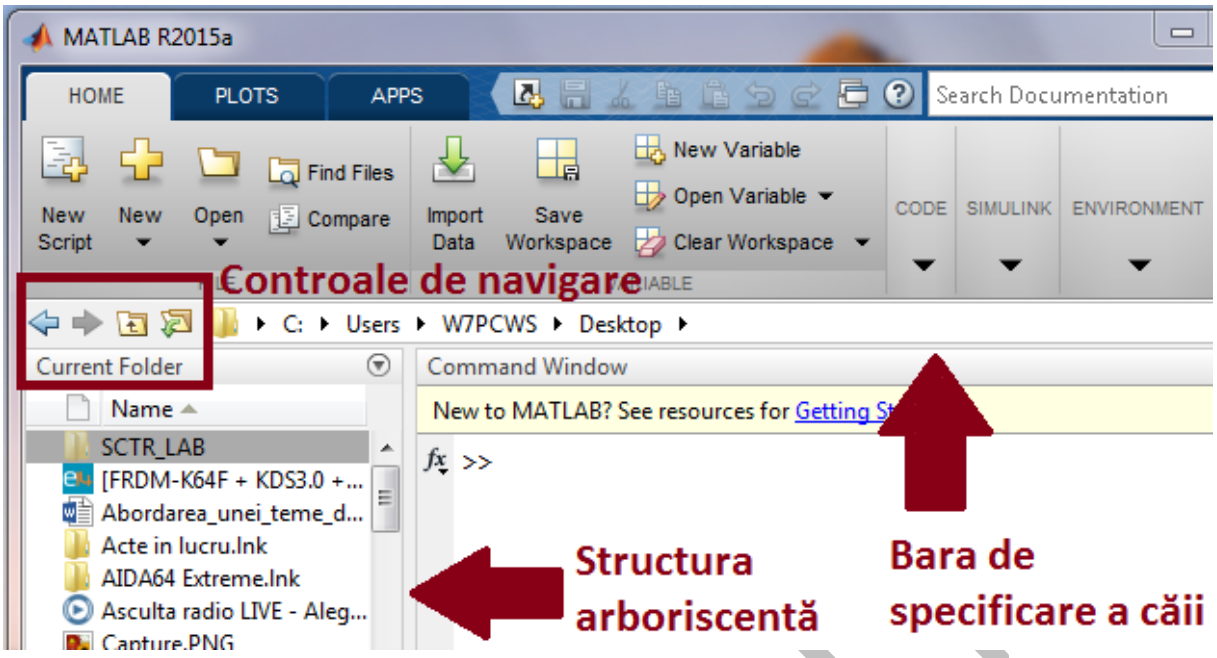


fig. 11 – Declararea căii de lucru pentru directorul creeat <literă_partiție_principală>:\Users\<nume_utilizator_pc>\Desktop\<nume_director>

2. În vederea **realizării aparatului matematic**, mai precis al unui **model matematic**, și în vederea **simulării în timp real** a logicii / algoritmului de comandă și control, **vom lansa în execuție mediul Simulink**, care este componentă grafică principală a mediului Matlab. Acest lucru se poate realiza prin **apăsarea butonului grafic numit „Simulink Library"** din **bara de instrumente a mediului Matlab;**

fig. 12 – Lansarea în execuție
a mediului Simulink – bara de instrumente

3. Pentru a începe un model „Simulink" matematic nou, în **fereastra nou deschisă „Simulink Library Browser"** se va alege **butonul „New Model"**;

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real"
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

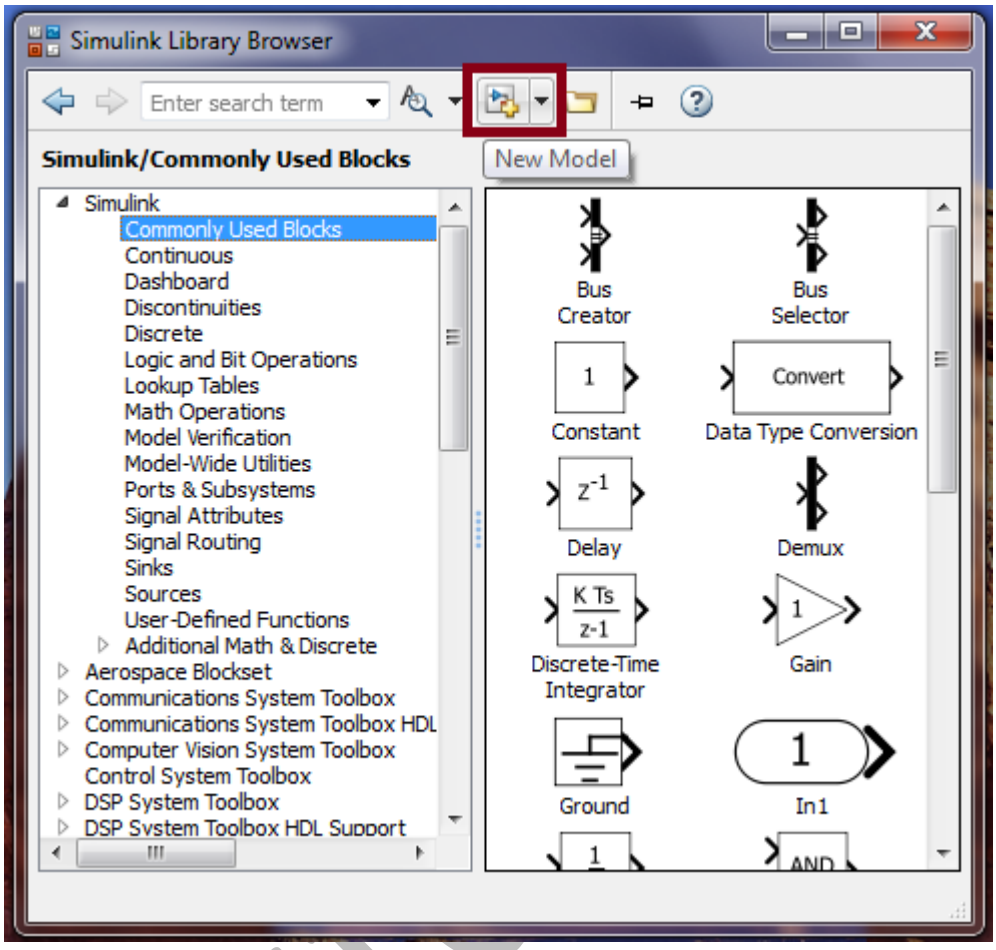


fig. 13 – Crearea unui model nou – Simulink Library Browser – Opțiunea „New Model”

4. **Setarea parametrilor de simulare** se realizează din fereastra nou deschisă **„untitled – Simulink”**, din meniul **„Simulation”** opțiunea **„Model Configuration Parameters”**;

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

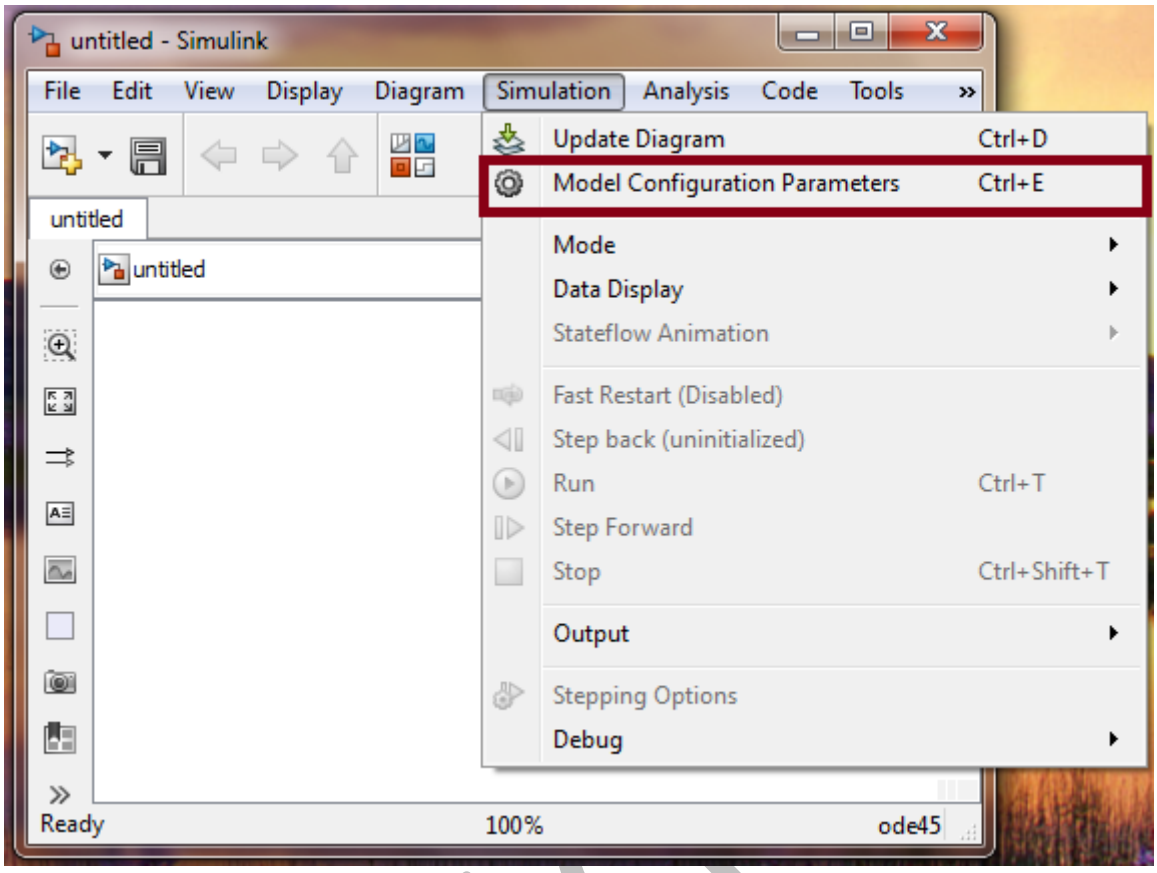


fig. 14 – Fereastra „untitled – Simulink" – meniul „Simulation" – opțiunea „Model Configuration Parameters"

5. În urma acestei acțiuni, se va deschide o fereastră cu titlul „**Configuration Parameters**". În **partea dreaptă** a acestei fereste, se pot alege opțiunile privitoare la **timpul total de simulare**, **timpul de eșantionare** și **metoda de rezolvare a ecuațiilor diferențiale** (dacă modelul impune o abordare de tip „calcul diferențial"). A se remarca faptul că, **opțiunile sunt grupate în categorii** (ex. categoria: Simulation Time). În această fereastră se vor efectua următoarele setări:

-În categoria „**Simulation Time**", la opțiunea „**Start time**" se va alege (sau rămâne neschimbată) **valoarea „0.0"**, iar la opțiunea **„Stop time"** se va alege **valoarea „inf"** („infinit" – simularea rulează până când utilizatorul alege să o oprească, sau prin utilizarea unui bloc de „STOP" se impune o anumită condiție de oprire a simulării);

-În categoria „**Solver options**" pentru opțiunea „**Type**" se va alege „**Fixed-step**", la opțiunea „**Solver**" se va alege valoarea „**discrete (no continuous states)**", iar la opțiunea „**Fixed-step size (fundamental sample time)**" (adică timpul fundamental de eșantionare – [calcul în unitate de timp]) se va alege valoarea „**1e-3**" (adică 10^{-3} [s] ~ aprox. 1000 [Hz] = 1 [kHz]);

După alegerea acestor opțiuni, se va apăsa butonul „**Apply**" apoi „**Ok**";

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real"
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

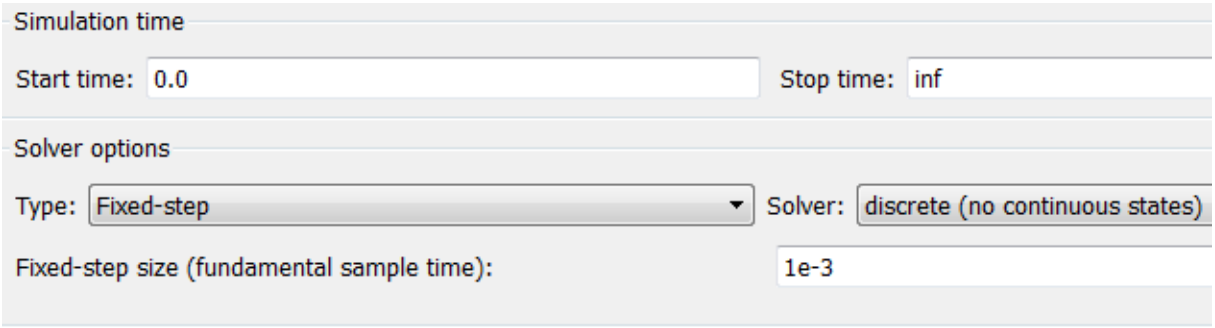


fig. 15 – Stabilirea parametrilor de simulare

IMPORTANT: În modelul matematic care se va concepe, **în acord cu setările de simulare impuse**, (**timp de eșantionare 10^{-3} [s]** din care rezultă **frecvența de eșantionare de 1 [kHz]**) se vor putea vehicula semnale în gama de frecvență [0 ... 50] [Hz], fără ca sistemul de calcul și achiziție să deformeze (sau să trunchieze / aproximeze greșit) semnalul!
(**VEZI OBLIGATORIU „DOCUMENTE AUXILIARE" PE PAGINA WEB: „Teorema lui Nyquist - Shannon – Demonstrație" – „cum se alege timpul de eșantionare optim" – „conceptul de timp și frecvență de eșantionare" (eng. sample time / sample frequency)).**

CUM ALEGEM CORECT TIMPUL DE EȘANTIONARE (eng. sample time):

- Fie, $f_{y(t)}$ = 50 [Hz] frecvența maximă vehiculată în model (frecvența uzuală alernativă);
- Perioada minimă vehiculată în model, deci va fi $T_{y(t)} = 1 / f_{y(t)} = 1 / 50$ [Hz] = 0.02 [s];
- Frecvența Nyquist, $f_{Nyquist} = 2 \times f_{y(t)} = 2 \times 50$ [Hz] = 100 [Hz];
- Perioada Nyquist, $T_{Nyquist} = T_{y(t)} / 2 = 0.01$ [s]; SAU $T_{Nyquist} = 1 / f_{Nyquist} = 1 / 100$ [Hz];

Având aceste date, conform teoremei lui Nyquist – Shannon **timpul de eșantionare**, va trebui să fie **mult mai mic** decât **perioada Nyquist**! Astfel, pentru o **platformă Arduino**, un timp de eșantionare de $\mathbf{T_{sample} = 10^{-3} = 0.001}$ **[s]** este **de 10 ori mai mic decât perioada Nyquist** la o **frecvență a fundamentalei de 50 [Hz]. Optim** pentru aplicația de semnalizare în care **nu este necesar** să atingem o frecventă de **50 [Hz],** dar **semnalele de comunicație serială, se încadrează în domeniul de frrecvență [0 – 50] [Hz], dar nu ating 50 [Hz].**

C. SETĂRI NECESARE ÎN REALIZAREA MODELULUI DE SIMULARE:

În urma realizării setărilor de simulare, se va salva modelul Simulink pentru a crea fișierul de lucru dorit. Acest pas, se va realiza accesând meniul **„File"** și opțiunea **„Save As"**. După alegerea aceste opțiuni o fereastră standard formelor de dialog „Windows_Shell" se va deschide pentru a îndruma utilizatorul înspre calea de lucru, înspre a completa numele fișierului și înspre a alege formatul sau extensia fișierului de lucru. Se va alege **opțiunea implicită „.slx" pentru format / extensie** și **se va specifica un nume fișierului de lucru.** (ex. SCTR_semnalizare.slx).
IMPORTANT: **NU UTILIZAȚI SPAȚII SAU CARACTERE SPECIALE ÎN CONȚINUTUL NUMELUI FIȘIERULUI. SPAȚIUL VA FI ÎNLOCUIT CU _ (bară jos).** A se observa faptul că, deja calea de lucru în care se salvează fișierul, este cea specificată inițial în mediul Matlab! **A NU SE MODIFICA!**

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real"
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

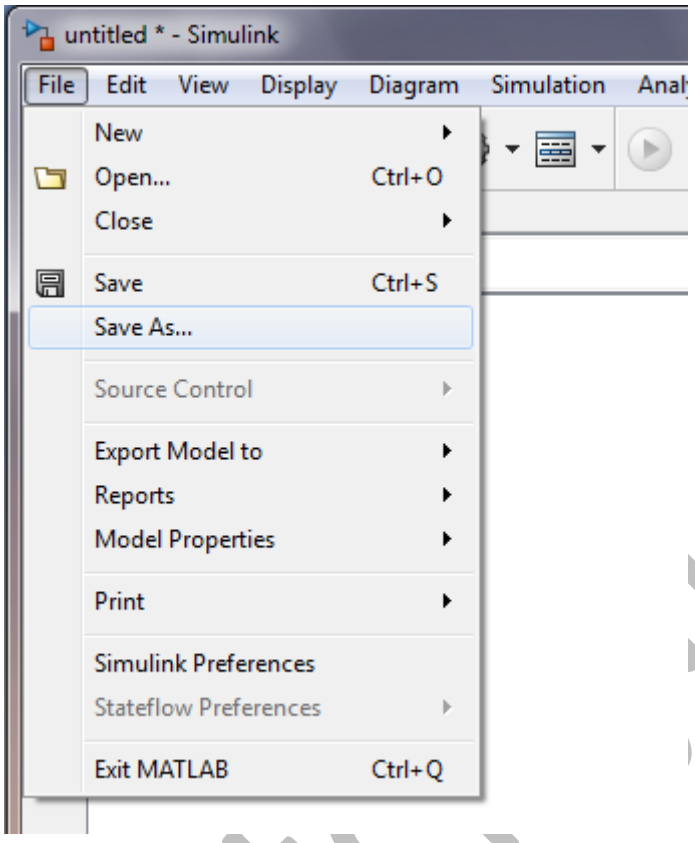


fig. 16 – Salvarea modelului Simulink

Dacă fișierul a fost creat, se va trece la pasul următor, al **implementării modelului.** Pentru a implementa modelul matematic, se vor utiliza instrumentele disponibile în bibleotecile și paletele de instrumente (en. libraries and toolboxes) conținute în îndrumătorul de categorii / bibleoteci (eng. Library Browser). Acesta poate fi accesat / apelat din bara de instrumente a spațiului de lucru Simulink (ca și în figura „fig. 10”):

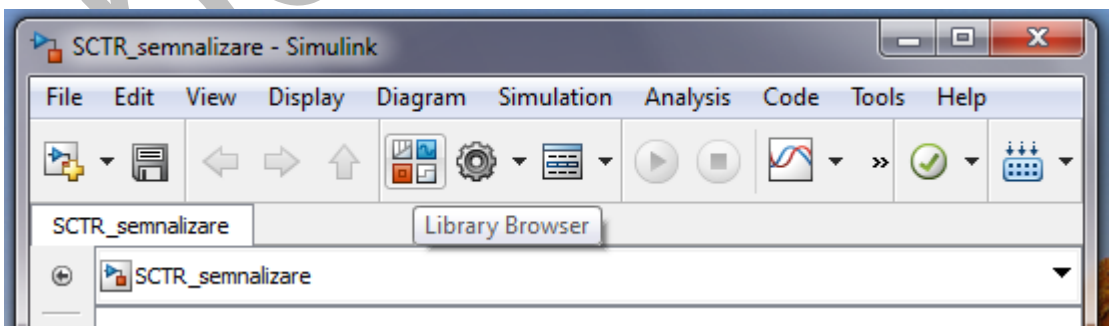


fig. 17 – Bara de instrumente a mediului Simulink – îndrumătorul de categorii / bibleoteci (eng. Library Browser)

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Odată deschis, îndrumătorul de categorii (eng. Library Browser) va pune la dispoziția utilizatorului paletele de instrumente necesare realizării modelului de simulare în timp real a sistemului de semnalizare. Elementele necesare conceperii modelului se regăsesc în următoarele categorii (fig. 11 verde) și subcategorii (fig. 11 portocaliu):

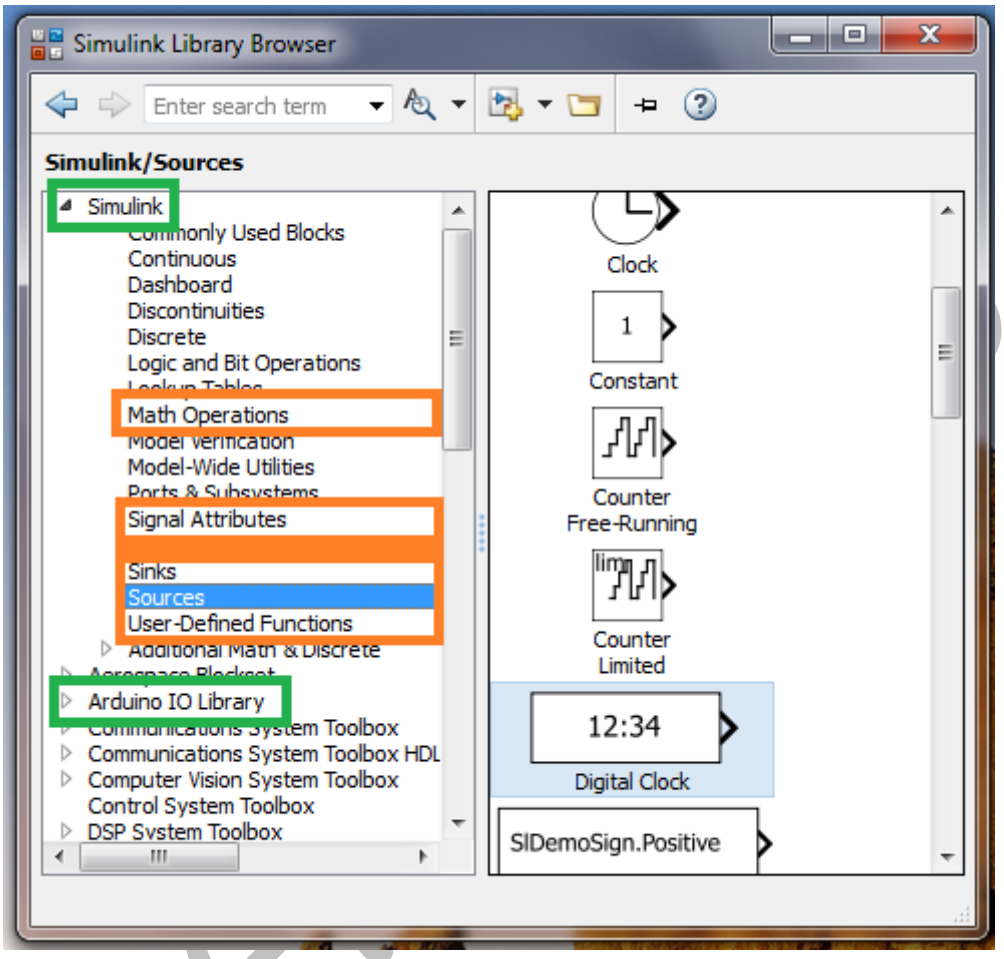


fig. 18 - Îndrumătorul de categorii / bibleoteci (eng. Library Browser) – categoria „Simulink” – subcategoriile: „Math Operations”, „Signal Attributes”, „Sinks”, „Sources”, „User-Defined Functions” și categotia „Arduino IO Library”

anume: categoria „Simulink” din care fac parte următoarele subcategorii: „Math Operations”, „Signal Attributes”, „Sinks”, „Sources”, „User-Defined Functions” și categotia „Arduino IO Library”. Elementele necesare modelului sunt: „Arduino IO Setup”, „Real-Time Pacer”, „Arduino Digital Read”, „Display”, „Scope”, „Digital Clock”, „Pulse Generator”, „Product”, „Fcn”, „Arduino Digital Write”. Fiecare bloc, îndeplinește un anumit rol funcțional în modelul matematic, deci, va fi necesară o clasificare / împărțire / grupare după rolul lor funcțional în model dar și o descriere a parametrilor. Se va evalua logica / algoritmul de lucru al modelului și se va prezenta de asemenea, sub formă tabelară, denumirea fiecărui bloc, și categoria / subcategoria sa de apartenență / proveniență, dar și numărul (cantitatea) necesar de blocuri de respectivul tip.

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

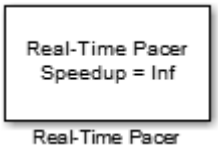
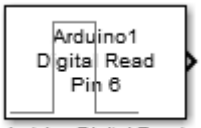
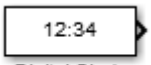
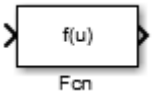
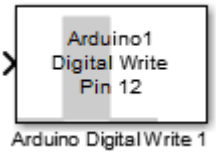
Simbol	Denumire consacrată	Rol funcțional	Categorie	Subcategorie
	Arduino IO Setup	Configurare a simulării sau a plăcii Arduino	Arduino IO Library	-
	Real-Time Pacer	Configurare a simulării sau a plăcii Arduino	Arduino IO Library	-
	Arduino Digital Read	Intrare / preluare de semnal de la periferice	Arduino IO Library	-
	Display	Afișare în calculator	Simulink	Sinks
	Scope	Afișare în calculator	Simulink	Sinks
	Digital Clock	Generare de semnal logic (eng. soft)	Simulink	Sources
	Pulse Generator	Generare de semnal logic (eng. soft)	Simulink	Sources
	Product	Procesare / prelucrare de semnal	Simulink	Math Operations
	Fcn	Procesare / prelucrare de semnal	Simulink	User-Defined Functions
	Arduino Digital Write	Generare de semnal fizic / înspre periferice (eng. hard)	Arduino IO Library	-

fig. 19 – Denumirea, rolul funcțional,
și locația, blocurilor componente modelului

Conform logicii de comandă descrise mai sus (capitolul „I" al acestui document subpunctul 4 – „Logica de comandă), se recomandă următorul model matematic Simulink, în care s-au introdus și blocurile de interacțiune în timp real cu platforma

Arduino, prin intermediul căreia se comandă elementele fizice ale sistemului:

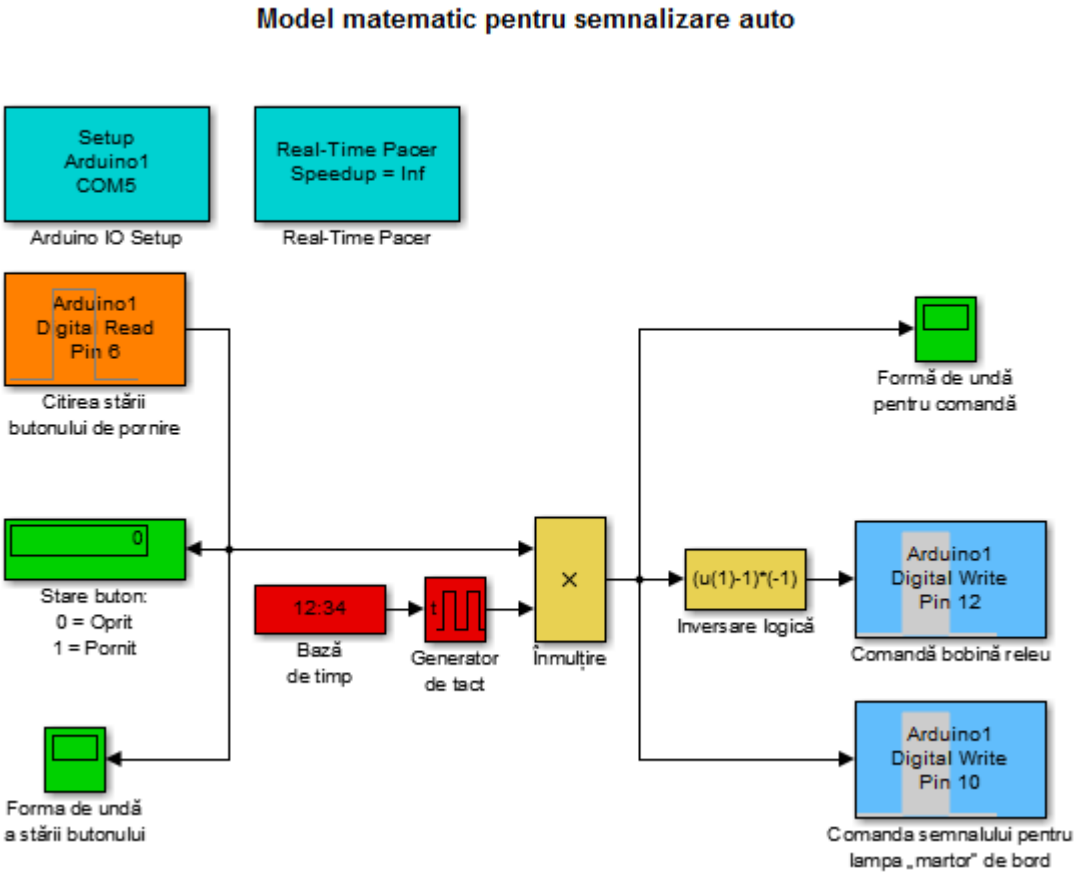


fig. 20 – Diagrama modelului matematic Simulink a sistemului de semnalizare

Descrierea parametrilor blocurilor componente din model:

1. Arduino IO Setup: - Este blocul de configurare al platformei Arduino, prin intermediul căruia se stabilesc parametrii de comunicare precum portul serial al plăcii (ex. în Windows „COMx", în Linux / Unix „/dev/ttySxyz"), dar și denumirea sau variabila atribuită platformei Arduino. (ex. în cazul utilizării a două sau mai multe platforme simultan, se poate alege care dintre ele să fie atribuită variabilei „ArduinoX"). În cazul modelului actual alegem portul „COMx" (unde „x" este numărul portului indicat în Device manager pentru placa Arduino (pentru noi COM5), și variabila „Arduino1"):

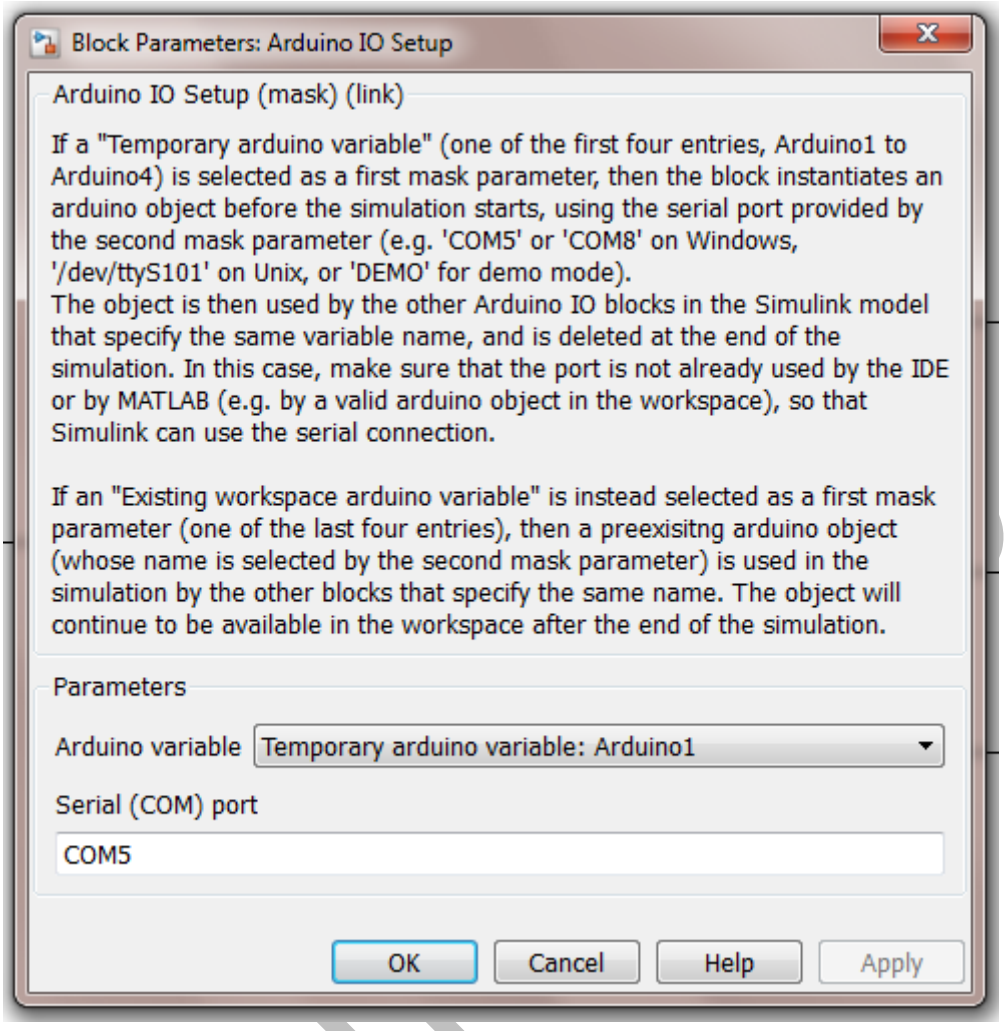


fig. 21 – Setările blocului „Arduino IO Setup”

2. Real-Time Pacer: - Este blocul prin intermediul căruia, se poate adjusta raportul dintre timpul real și timpul virtual de simulare. Pentru un raport 1:1 (o secundă de simulare corespunde unei secunde în timp real) se va alege valoarea „inf” (infinit):

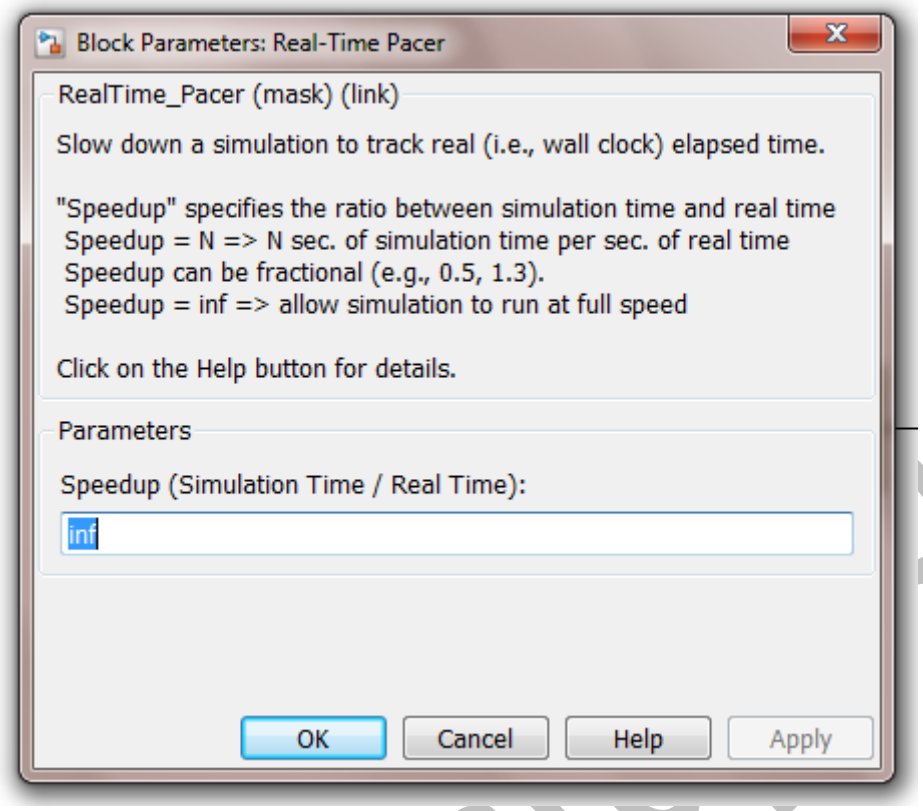


fig. 22 - Setările blocului „Real-Time Pacer”

3. Arduino Digital Read: - Este un bloc prin intermediul căruia „informația” numerică de la platforma Arduino, mai precis dinspre intrările digitale, este preluată în calculator, în mediul Matlab Simulink. Practic prin intermediul acestui bloc, se pot configura, parametrii specifici achiziției de date dinspre intrările digitale disponibile pe platforma Arduino. Se va alege numărul pinului de intrare (pentru noi „6”) și timpul de achiziție / eșantionare 0.01. **Acest pin corespunde comutatorului fizic din montaj!**

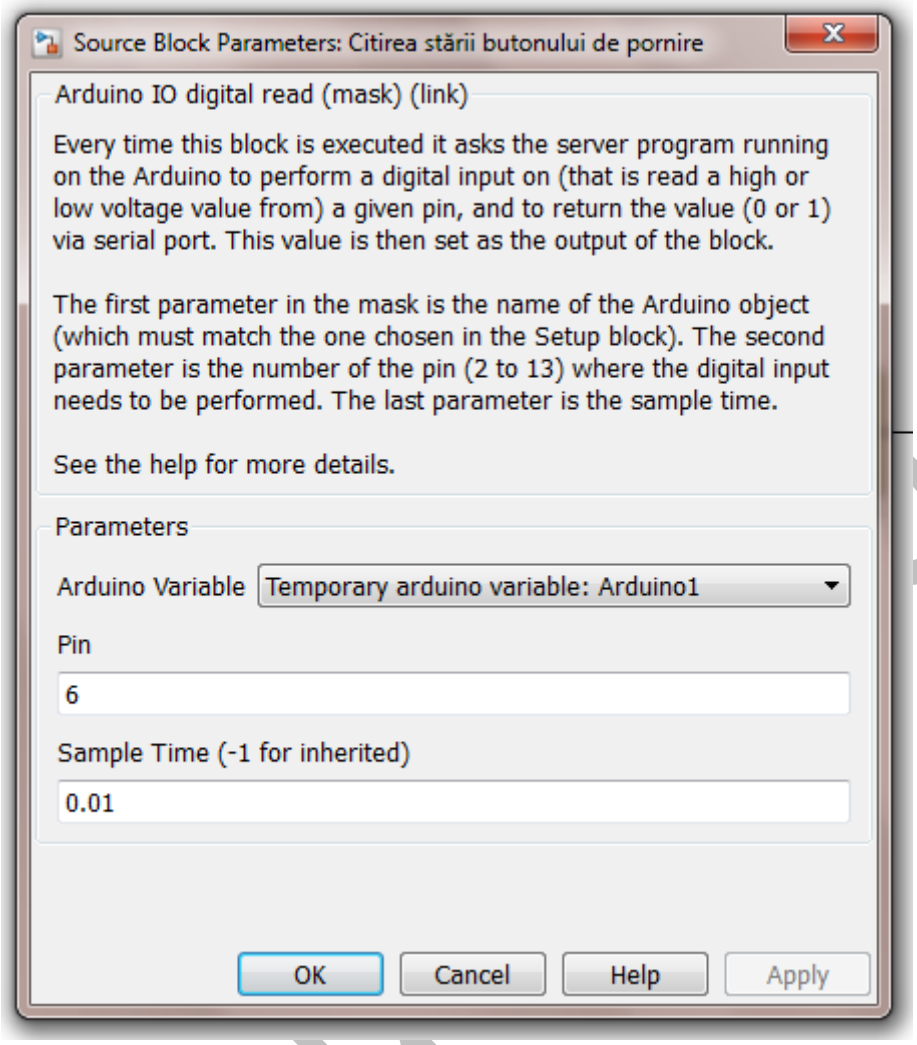


fig. 23 - Setările blocului „Arduino Digital Read”

4. Display: - Prin intermediul acestui bloc, se pot vizualiza valorile numerice ale semnalelor vehiculate în modelul matematic. Prin intermediul căsuței glisante (eng. ComboBox) „Format” se pot alege tipurile de date sub forma cărora se afișează pe ecran valorile. Prin intermediul opțiunii „Decimation” se poate specifica „numărul de afisări” în unitatea de timp (de obicei o secundă) sau rata de actualizre a afișajului. Se păstrează valorile: Format: „Short” și Decimation: „1”; Prin intermediul acestui afișaj, se va putea monitoriza starea butonului / comutatorului de pornire.

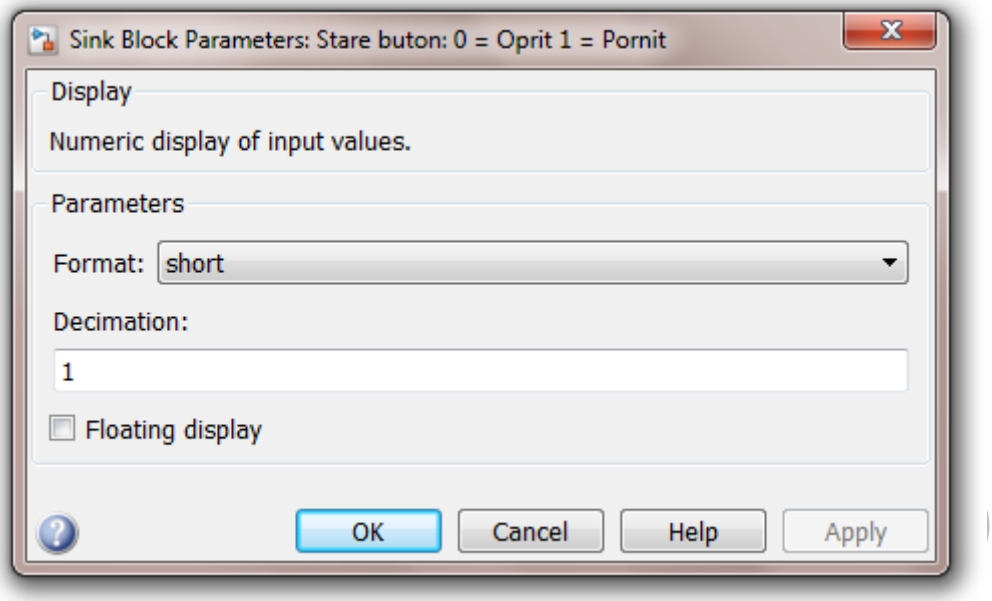


fig. 24 - Setările blocului „Display”

5. Scope: - Acest bloc, este un osciloscop virtual, prin intermediul căruia se pot urmări formele de undă sau evoluțiile în timp ale unor semnale. Pentru a configura parametrii acestui bloc, se va executa manevra de „dublu click” pe acest bloc. Se va deschide fereastra osciloscopului virtual. În această fereastră, pe porțiunea cadranului negru, se va executa manevra de „click dreapta”, care va rezulta un meniu, din care se va alege opțiunea: „Axes properties”.

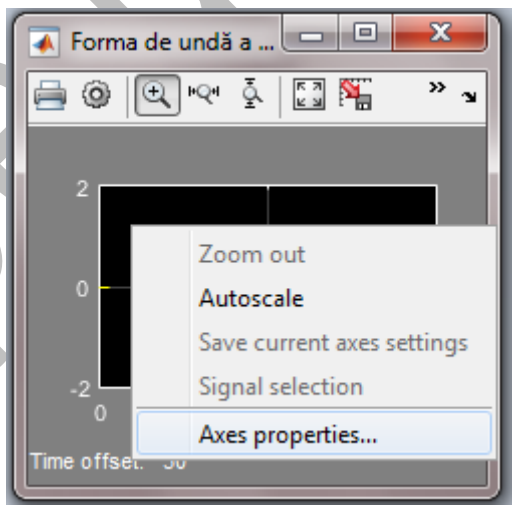


fig. 25 - Setările blocului „Scope”

Având în vedere faptul că, semnalele vehiculate în acest model, nu depășesc valori mai mari de „1” și „0” (fiind semnal digitale / booleene), se vor alege limitele astfel: „Y-min = -2” și „Y-max = 2”, restul opțiunilor rămân neschimbate! S-au ales limitele + / - 2 deoarece, se dorește ca imaginea afișajului de osciloscop, să fie simetric centrată față

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

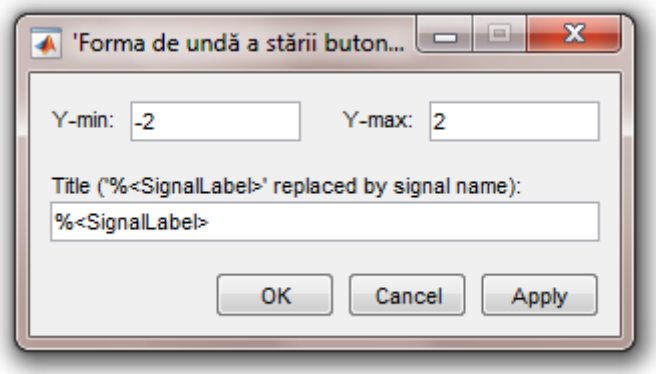


fig. 26 - Setările blocului „Scope” – stabilirea limitelor axei „Y”

de zero! Pentru setările de achiziție, se va deschide meniul de configurare denumit „Parameters” având ca și simbol „roata dințată” în bara de instrumente a ferestrei. În fereastra care se va deschide, pentru categoria „General” se vor păstra setările implicite, la fel și pentru categoria „Style”, doar în categoria „History” se va dezactiva opțiunea „Limit data points to last”. Această opțiune când este activă, limitează numărul de puncte achiziționate de osciloscop la numărul specificat în căsuța de text (implicit 5000). În cazul de față, nu se dorește acest lucru, deoarece, este necesară urmărirea evoluției pe parcursul timpului de simulare.

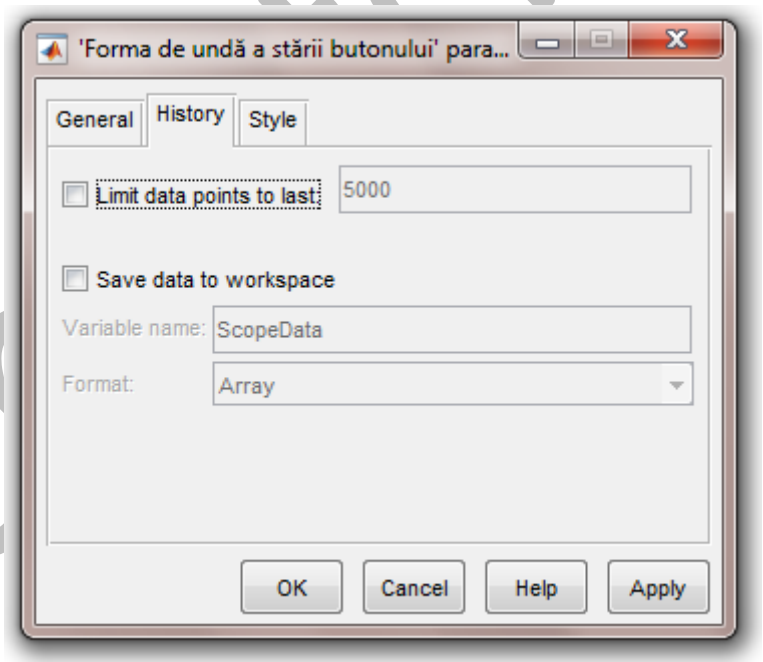


fig. 27 - Setările blocului „Scope” –
dezactivarea opțiunii „Limit data points to last”

6. Digital Clock: - Constituie baza de timp pentru generatorul de semnal / de tact. Prin intermediul acestuia, se va putea stabili timpul de eșantionare al generatorului de tact.

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Se va alege același timp de eșantionare (eng. Sample Time) ca și în cazul intrărilor / ieșirilor digitale, adică 0.01 [s].

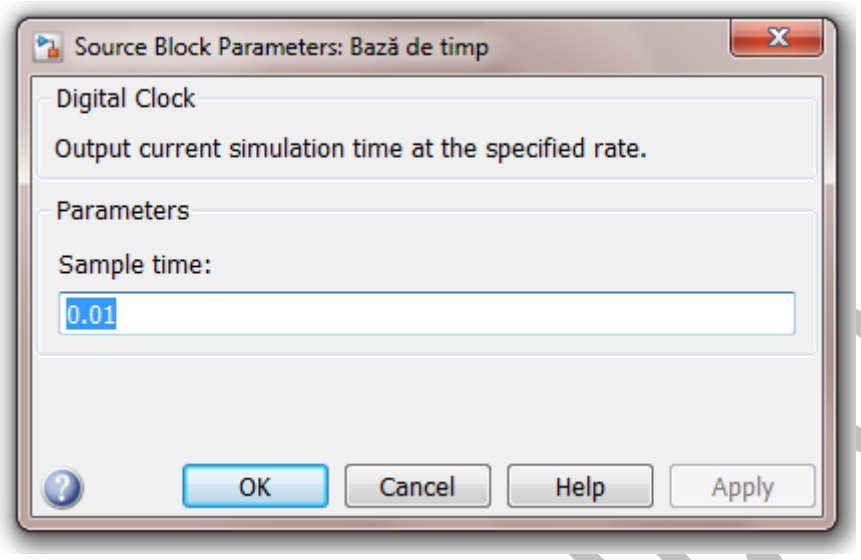


fig. 28 - Setările blocului „Digital Clock”

7. Pulse Generator: - Este blocul care generează trenul de impulsuri sau temporizarea sistemului de semnalizare. În caseta de dialog a acestui bloc, se regăsesc parametrii precum modul de generare al impulsului (pe bază de perioadă, sau pe bază de eșantion), sursa de timp (bază de timp internă sau externă), amplitudine, perioadă, lățimea impulsului și defazaj. În cazul modelului de față, se va face următoarea parametrizare:

- **Pulse type**: Time based;
- **Time:** Use external signal;
- **Amplitude:** 1;
- **Period:** 0.5 [s];
- **Pulse width:** 0.5 [%];
- **Phase delay:** 0 [s];

IMPORTANT: În urma alegerii opțiunii „Use external signal”, blocul generator de impuls, va avea și o intrare de semnal! Aceasta este intrarea pentru baza de timp externă, furnizată de către blocul „Digital Clock” (tact de ceas digital sau bază de timp digitală). Cu ajutorul acestui bloc se va realiza sincronizarearea generatorului!

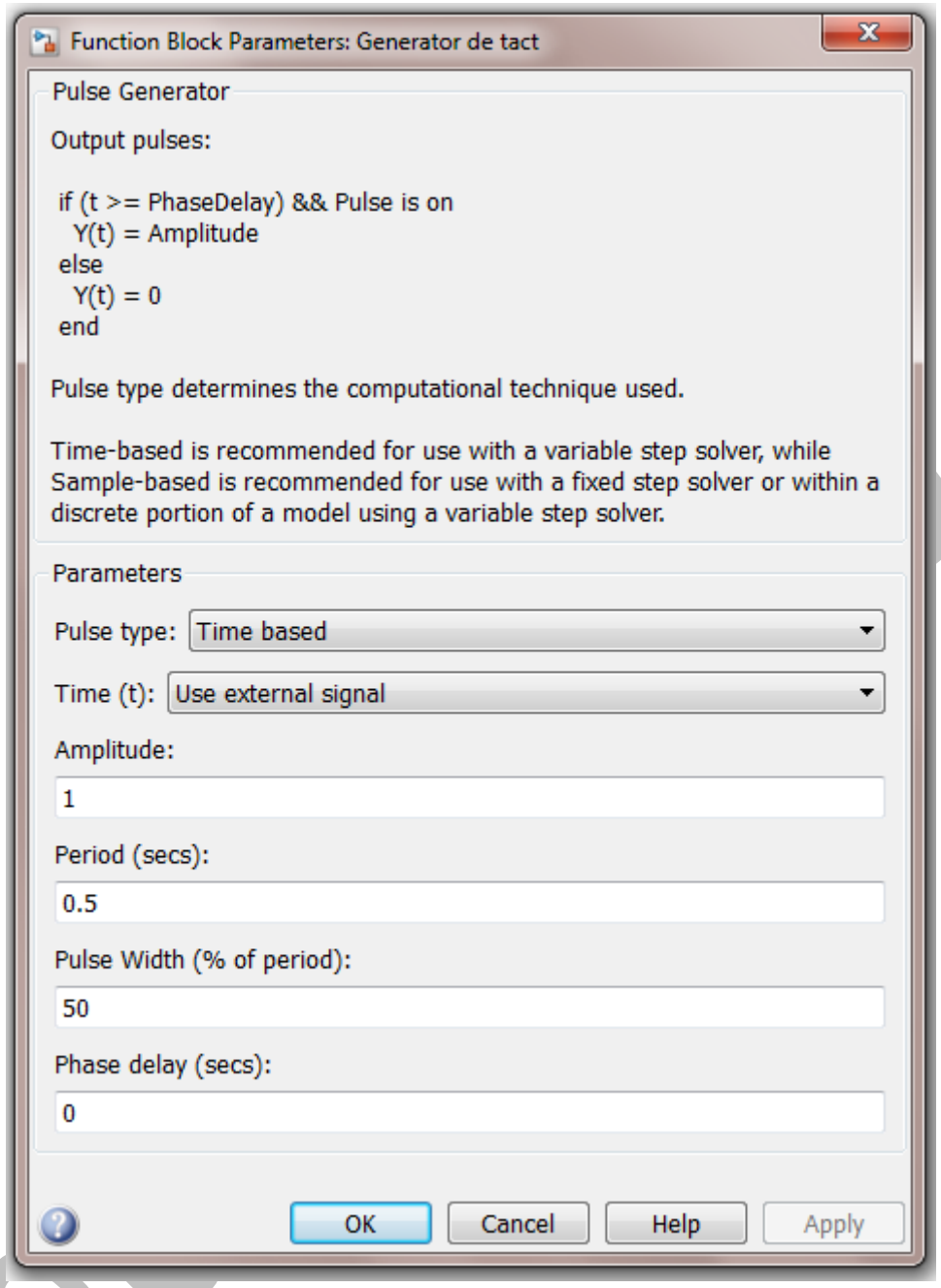


fig. 29 - Setările blocului „Pulse Generator”

8. Product: - Acest bloc realizează produsul scalar dintre două semnale (unidimensionale – vector sau șir). În acest model, se vehiculează doar valorile numerice „1” și „0”. Pentru a acționa sistemul de semnalizare, (a-l porni de la buton), se va realiza produsul dintre valorile măsurate la timpul de eșantionare „T_{sample}” ale stării butonului (rezultă un șir / vector de valori cvasi-constante în timp [0 0 0 0 0]- oprit sau [1 1 1 1 1]- pornit) și valorile discrete generate în timp de către generatorul de impuls ([1 0 1 0 1 0 .... 1 0]). Rezultă deci două cazuri:

- Pentru <stare_buton> = 0 → [0 0 0 0 0 0] x [0 1 0 1 0 1] = [0 0 0 0 0 0];
- Pentru <stare_buton> = 1 → [1 1 1 1 1 1] x [0 1 0 1 0 1] = [0 1 0 1 0 1];

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

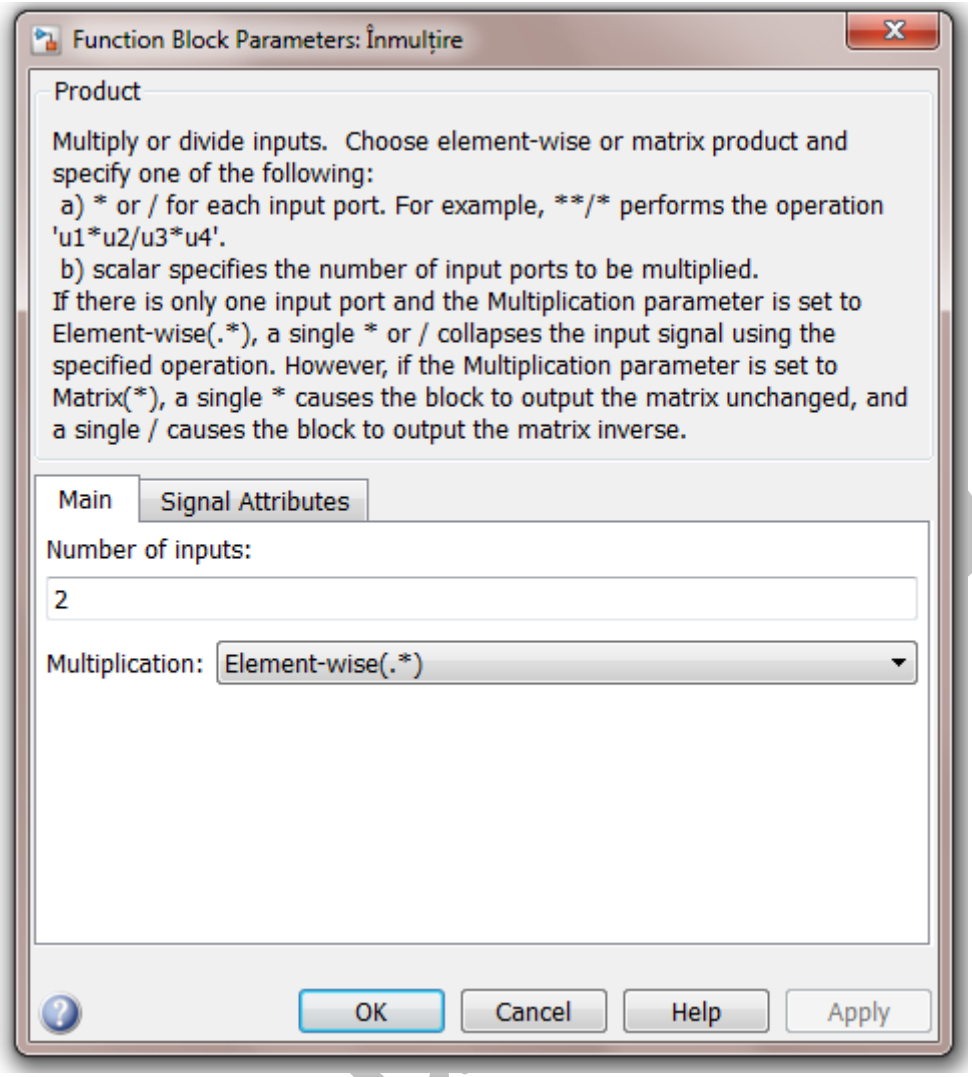


fig. 30 - Setările blocului „Product”

9. Fcn (Function - User-Defined Function): - Este un bloc pentru definirea analitică a expresiei matematice a unei funcții. Cu ajutorul acestei funcții, în cazul de față, s-a realizat inversarea logică a semnalului, deoarece, releul care acționează lampa cu incandescență lucrează pe bază de logică inversată (adică pentru „logic 0” = pornit, iar pentru „logic 1” = oprit). Astfel, respectivul bloc, implementează funcția:
f(x) = (x – 1) x (-1), rezultă deci, două cazuri:

- Pentru „x” = 0 → f(x) = (0 – 1) x (-1) = -1 x (-1) = 1, (iar x = 0);
- Pentru „x” = 1 → f(x) = (1 – 1) x (-1) = 0 x (-1) = 0, (iar x = 1);

În cadrul acestui bloc, variabila „x” este înlocuită cu „u(i)” unde „i” este indicele de ordine al elementului din matricea unidimensională (vector) sau șir.

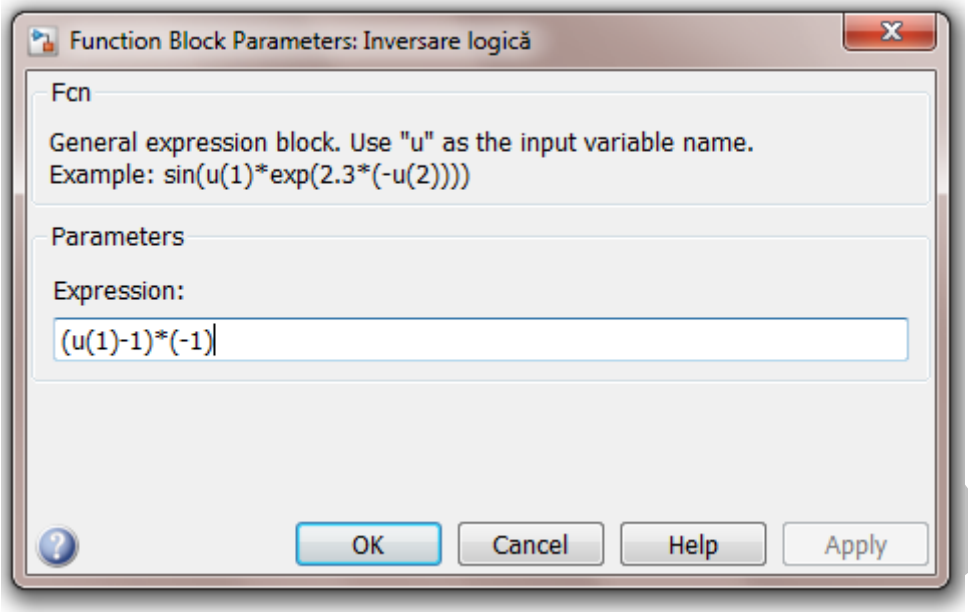


fig. 31 - Setările blocului „Fcn (Function - User-Defined Function)”

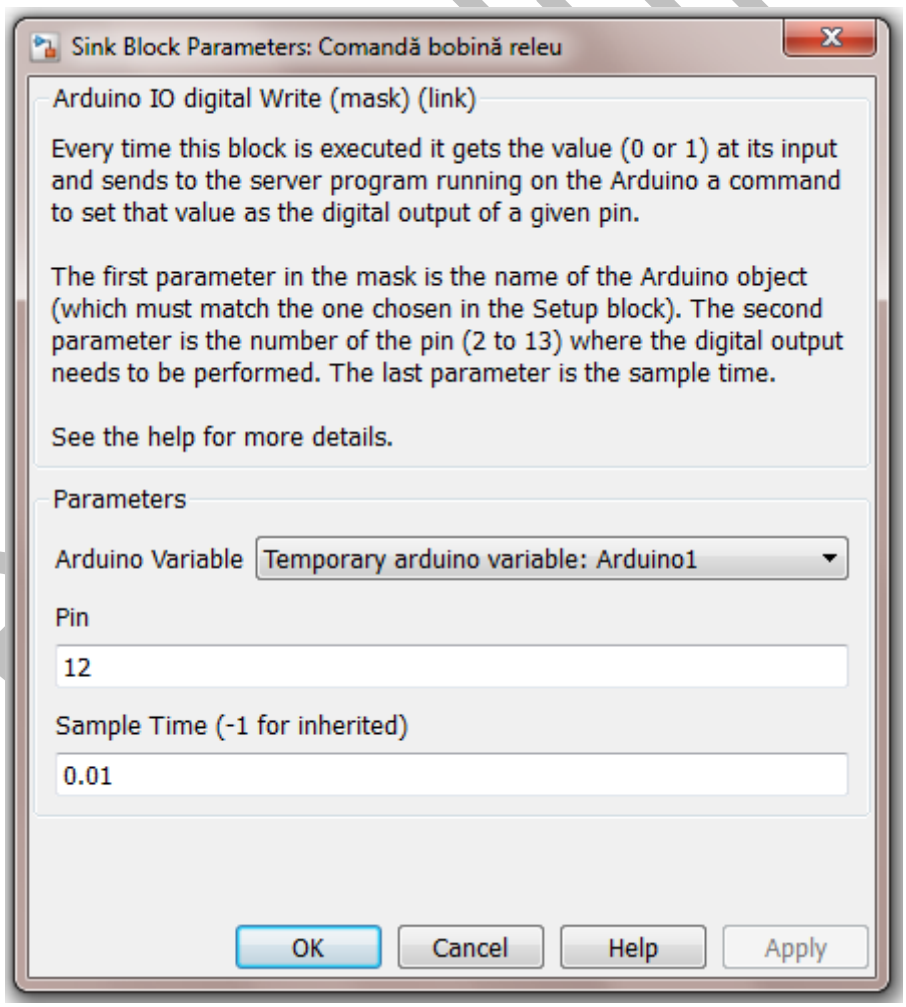


fig. 32 - Setările blocului „Arduino Digital Write”

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

10. Arduino Digital Write: - Este blocul prin intermediul căruia se vor furniza semnalele (numerice) de comandă înspre ieșirile digitale (care generează semnalele fizice). În modelul de față, se vor alege parametrii: Pin: „12” și Sample time: „0.01”; - **La acest pin se va lega fizic, releul.** Se multiplică același bloc încă odată, dar pentru cel rezultat, se va alege: Pin „10” și Sample time: „0.01”; ”; - **La acest pin se va lega fizic, un led.**

IV. CONCLUZII ȘI FINALIZARE:

În urma parcurgerii tuturor etapelor de mai sus, se va obține următorul montaj, având modelul matematic indicat mai sus, funcțional. Se va constata că lampa de semnalizare cu incandescență funcționează într-un regim **intermitent împreună cu lampa „martor” indicator de semnalizare (LED verde)**, și totul este **condiționat de starea comutatorului.**

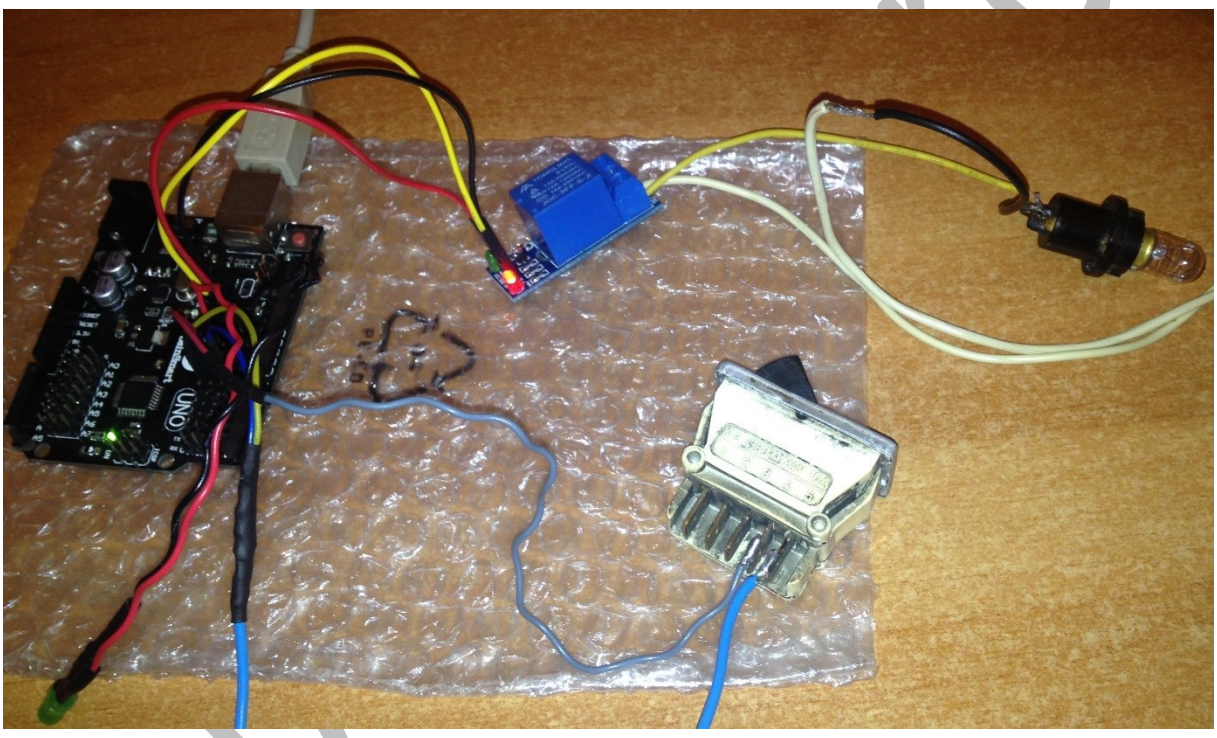
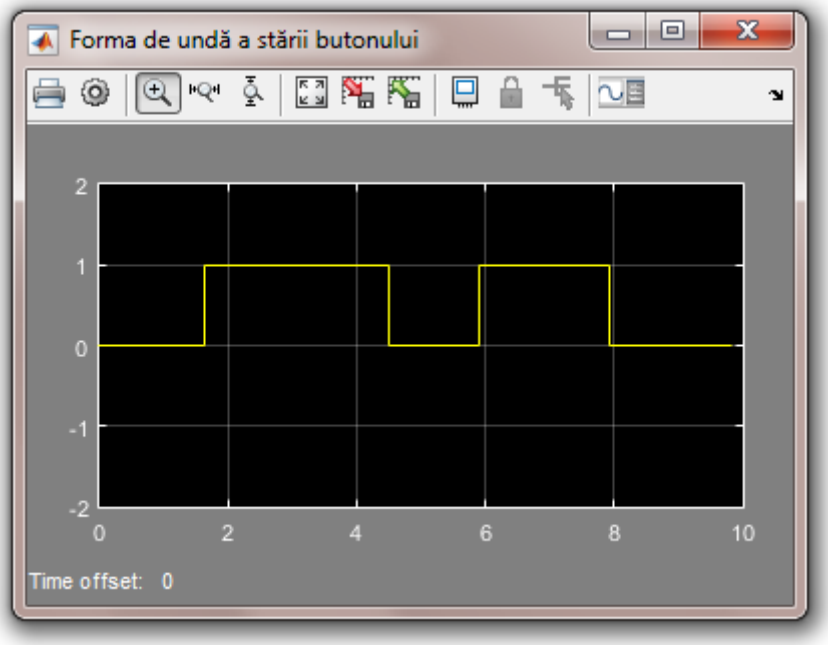


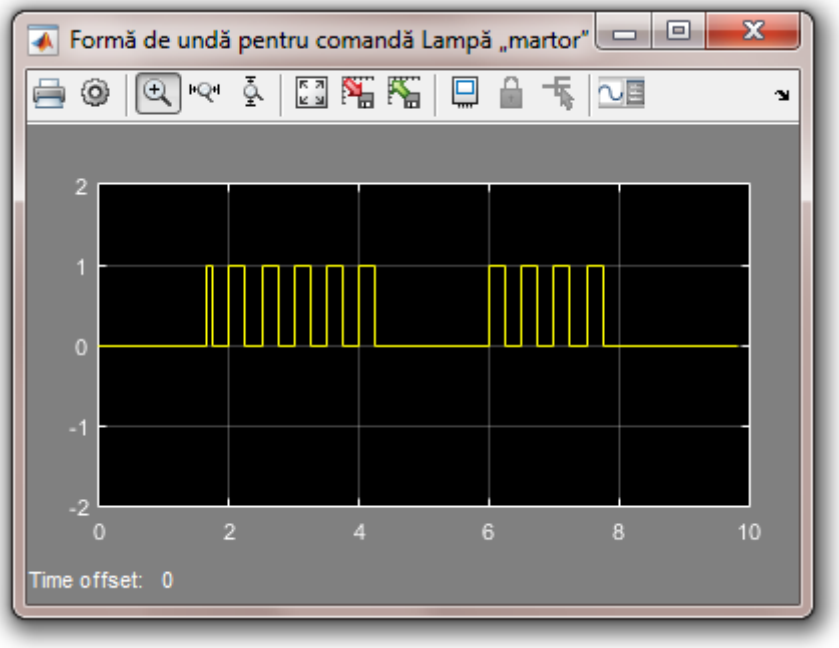
fig. 33 – Montajul experimental

Dacă se urmăresc „oscilografiile” generate de către osciloscoapele virtuale, se va observa funcționarea (evidentă) a sistemului conform celor cerute de către beneficiar. În **prima oscilografie**, este evidențiată **starea butonului de pornire / oprire a sistemului de semnalizare**, iar în **a doua oscilografie**, starea **ieșirilor digitale (LED și releu)**. Se observă că, **la acționarea butonului**, începe **secvența repetitivă a generatorului de impuls.** Trebuie precizat de asemenea, faptul că, **ieșirea digitală la care a fost cuplat releul**, funcționează în **contra-tact** (sau atinfază / logică inversată) **față de ieșirea digitală a „lămpii martor”,** deoarece, releul funcționează pe bază de **logică inversată** (adică, circuitul de comandă al bobinei are nevoie de un impuls electric pentru blocare, iar în absența impulsului de blocare, acesta conduce conduce în permanență).

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

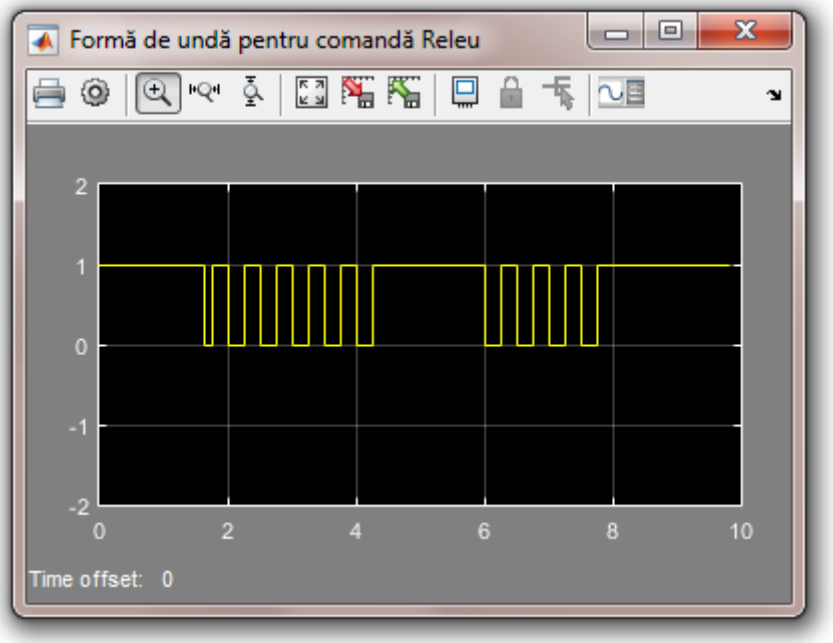


A.



B.

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real"
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro



C.

fig. 34 – Evoluția în timp a semnalelelor vehiculate în sistemul de semnalizare

Astfel, s-a obținut un sistem de semnalizare **controlat numeric / digital**, **reprogramabil,** având o logică de comandă **simplă** și **ușor de implementat la nivelul unui microcontroller** sau **calculator dedicat de bord.**

OBSERVAȚIE: Pentru ca modelul să fie încărcat **permanent** în memoria **micro-controllerului,** se recomandă utilizarea paletei de instrumente **„Simulink Support Package for Arduino”**, care are un compilator C standard / C++ prin intermediul căruia, modelul Simulink este transpus **sub formă de cod program**, iar apoi este încărcat în memoria flash a microcontrollerului. Pentru a face acest lucru, este necesară **înlocuirea blocurilor folosite pentru simulare** (din bibleoteca / categoria **Arduino IO**) cu cele din bibleoteca: „Simulink Support Package for Arduino”. După această etapă, din meniul **„Tools”**, se va alege opțiunea **„Run on Target Hardware”** iar apoi **„Prepare to Run”**. Se va deschide o fereastră, prin intermediul căreia, se va putea alege **tipul platformei Arduino**, **portul de comunicare** și alți parametrii. După alegerea setărilor corespunzătoare platformei utilizate, se vor confirma prin butonul **„Apply”** și **„Ok”**. În final, din meniul **„Code”**, categoria **„C / C++ Code”** se alege opțiunea **„Deploy to Hardware”**, sau mai simplu din bara de instrumente **iconița albastră dreptunghiulară indicată în figura 36**, o modalitate și mai simplă este să se apese simultan **combinația de taste „Ctrl + B”** (eng. „Build” command). Odată procedat astfel, se va începe procesul de compilare, și încărcare programului în memoria flash a platformei Arduino. După încheierea procedurii de programare, micro-controllerul va putea funcționa **independent de calculator,** doar cu programul stocat în memoria sa flash, ideal aplicației specificate de către beneficiar. **Codul programului** se poată găsi **în directorul de lucru stabilit la început**, în

subdirectorul <nume_model>_rtt/<nume_model>.c, unde variabila **<nume_model>** este **numele ales pentru modelul în care s-a făcut compilarea** (în cazul de față „Semnalizare”);

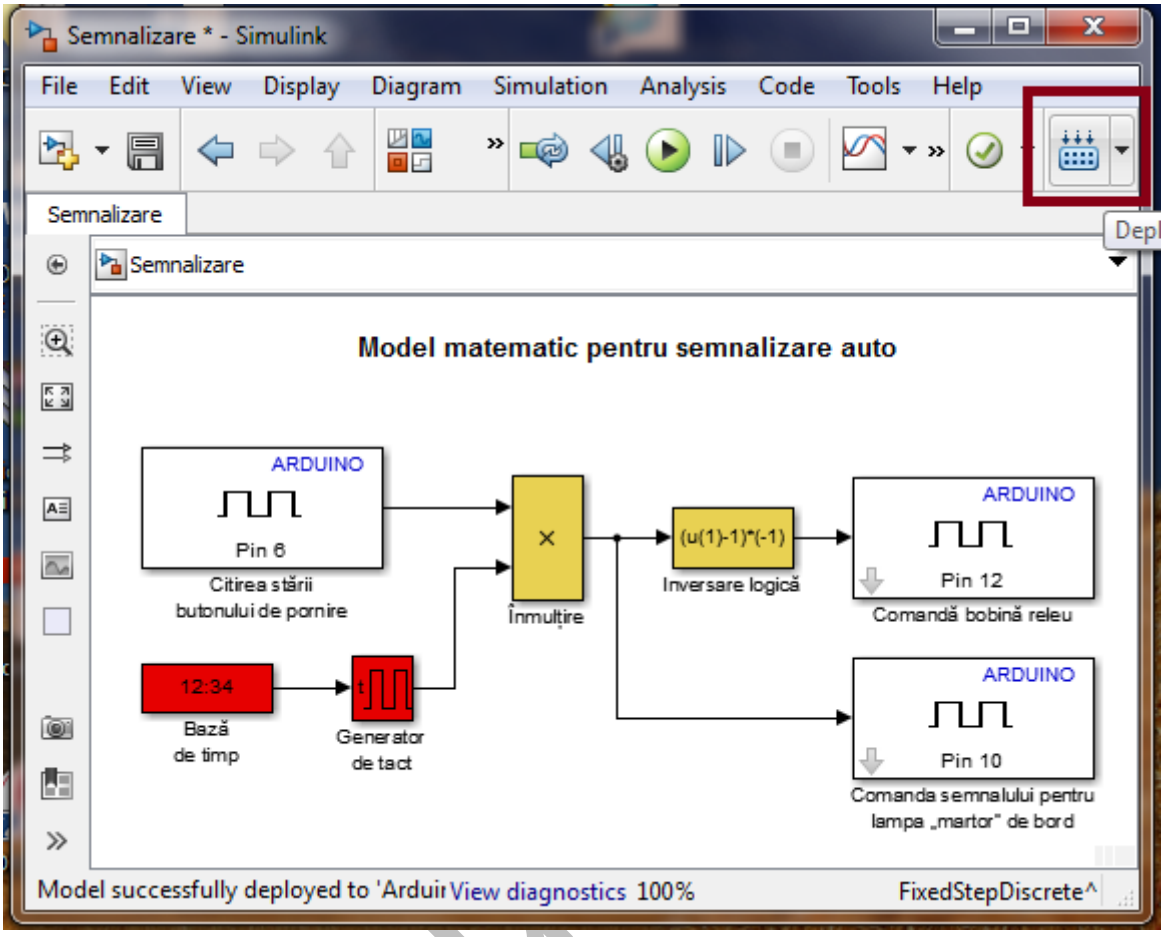


fig. 35 – Compilarea modelului matematic, în vederea încărcării în memoria flash a microcontrollerului

Codul programului generat:

```
/*
 * File: Semnalizare.c
 *
 * Code generated for Simulink model 'Semnalizare'.
 *
 * Model version                  : 1.12
 * Simulink Coder version         : 8.8 (R2015a) 09-Feb-2015
 * TLC version                    : 8.8 (Jan 20 2015)
 * C/C++ source code generated on : Fri Dec 29 01:21:07 2017
 *
 * Target selection: realtime.tlc
 * Embedded hardware selection: Atmel->AVR
 * Code generation objectives: Unspecified
 * Validation result: Not run
```

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

```
 */

#include "Semnalizare.h"
#include "Semnalizare_private.h"

/* Real-time model */
RT_MODEL_Semnalizare_T Semnalizare_M_;
RT_MODEL_Semnalizare_T *const Semnalizare_M = &Semnalizare_M_;

/* Model output function */
void Semnalizare_output(void)
{
  boolean_T rtb_Citireastriibutonuluidepo_0;
  real_T rtb_Generatordetact;
  real_T ratio;
  uint32_T numCycles;
  uint8_T rtb_Generatordetact_0;

  /* S-Function (arduinodigitalinput_sfcn): '<Root>/Citirea stării butonului de pornire' */
  rtb_Citireastriibutonuluidepo_0 = MW_digitalRead
    (Semnalizare_P.Citireastriibutonuluidepornire_);

  /* DigitalClock: '<Root>/Bază de timp' */
  rtb_Generatordetact = ((Semnalizare_M->Timing.clockTick0) * 0.01);

  /* DiscretePulseGenerator: '<Root>/Generator de tact' */
  if (rtb_Generatordetact >= Semnalizare_P.Generatordetact_PhaseDelay) {
    rtb_Generatordetact -= Semnalizare_P.Generatordetact_PhaseDelay;
    ratio = rtb_Generatordetact / Semnalizare_P.Generatordetact_Period;
    numCycles = (uint32_T)floor(ratio);
    if (fabs((real_T)(numCycles + 1UL) - ratio) < DBL_EPSILON * ratio) {
      numCycles++;
    }

    if (rtb_Generatordetact < Semnalizare_P.Generatordetact_Duty / 100.0 *
        Semnalizare_P.Generatordetact_Period +
        Semnalizare_P.Generatordetact_Period * (real_T)numCycles) {
      rtb_Generatordetact = Semnalizare_P.Generatordetact_Amp;
    } else {
      rtb_Generatordetact = 0.0;
    }
  } else {
    rtb_Generatordetact = 0.0;
  }
```

Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

```
 /* End of DiscretePulseGenerator: '<Root>/Generator de tact' */

 /* Product: '<Root>/Înmul?ire' incorporates:
  *  S-Function (arduinodigitalinput_sfcn): '<Root>/Citirea stării butonului de pornire'
  */
 rtb_Generatordetact *= (real_T)rtb_Citireastriibutonuluidepo_0;

 /* DataTypeConversion: '<S1>/Data Type Conversion' */
 if (rtb_Generatordetact < 256.0) {
   if (rtb_Generatordetact >= 0.0) {
     rtb_Generatordetact_0 = (uint8_T)rtb_Generatordetact;
   } else {
     rtb_Generatordetact_0 = 0U;
   }
 } else {
   rtb_Generatordetact_0 = MAX_uint8_T;
 }

 /* End of DataTypeConversion: '<S1>/Data Type Conversion' */

 /* S-Function (arduinodigitaloutput_sfcn): '<S1>/Digital Output' */
 MW_digitalWrite(Semnalizare_P.DigitalOutput_pinNumber, rtb_Generatordetact_0);

 /* Fcn: '<Root>/Inversare logică' */
 rtb_Generatordetact = -(rtb_Generatordetact - 1.0);

 /* DataTypeConversion: '<S2>/Data Type Conversion' */
 if (rtb_Generatordetact < 256.0) {
   if (rtb_Generatordetact >= 0.0) {
     rtb_Generatordetact_0 = (uint8_T)rtb_Generatordetact;
   } else {
     rtb_Generatordetact_0 = 0U;
   }
 } else {
   rtb_Generatordetact_0 = MAX_uint8_T;
 }

 /* End of DataTypeConversion: '<S2>/Data Type Conversion' */

 /* S-Function (arduinodigitaloutput_sfcn): '<S2>/Digital Output' */
 MW_digitalWrite(Semnalizare_P.DigitalOutput_pinNumber_d, rtb_Generatordetact_0);
}
```


Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

```
/* Model update function */
void Semnalizare_update(void)
{
  /* Update absolute time for base rate */
  /* The "clockTick0" counts the number of times the code of this task has
   * been executed. The resolution of this integer timer is 0.01, which is the step size
   * of the task. Size of "clockTick0" ensures timer will not overflow during the
   * application lifespan selected.
   */
  Semnalizare_M->Timing.clockTick0++;
}

/* Model initialize function */
void Semnalizare_initialize(void)
{
  /* Registration code */

  /* initialize real-time model */
  (void) memset((void *)Semnalizare_M, 0,
                sizeof(RT_MODEL_Semnalizare_T));

  /* Start for S-Function (arduinodigitalinput_sfcn): '<Root>/Citirea stării butonului de
pornire' */
  MW_pinModeInput(Semnalizare_P.Citireastriibutonuluidepornire_);

  /* Start for S-Function (arduinodigitaloutput_sfcn): '<S1>/Digital Output' */
  MW_pinModeOutput(Semnalizare_P.DigitalOutput_pinNumber);

  /* Start for S-Function (arduinodigitaloutput_sfcn): '<S2>/Digital Output' */
  MW_pinModeOutput(Semnalizare_P.DigitalOutput_pinNumber_d);
}

/* Model terminate function */
void Semnalizare_terminate(void)
{
  /* (no terminate code required) */
}

/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```


Realizat de: Ing. mast. Pintilie Lucian Nicolae
Pentru disciplina: „Sisteme de calcul în timp real”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro