Laborator 2
Proiectarea Sistemului

Obiective

- Prezentarea generală a conceptului de proiectare şi a arhitecturii de sistem.
- Descompunerea unui sistem în blocuri funcționale şi arii disciplinare.
- Identificarea blocurilor funcționale şi a ariilor disciplinare în cazul proiectului propus.

Cuprins

De ce este necesară proiectarea unui sistem?

Dezvoltarea unui produs, în industria automotive şi nu numai, începe cu definirea unui set de **cerințe (requirements) de sistem**, rezultate în urma analizei cerințelor clienților şi a tuturor celor implicați, dar şi a standardelor şi restricțiilor legale.

Setul de cerințe exprimă doar **ce** trebuie produsul să facă şi atributele acestuia, nu şi **cum** trebuie el implementat.

Soluția tehnică care corespunde cerințelor definite rezultă din **proiectarea sistemului**. În stabilirea soluției tehnice se ține cont de următorii factori:

- financiari (bugetul alocat, marja de profit, etc);
- tehnologiile tehnico-ştiințifice disponibile la acel moment (cunoştințe, tehnologii noi state-of-the-art, inovații, etc);
- resursele umane şi materiale disponibile (angajați, echipamente de lucru, etc).

Astfel, **proiectarea sistemului** este procesul prin care se definesc arhitectura sistemului, elementele componente şi interacțiunile dintre ele, precum şi fluxul de date din sistem.

Proiectarea sistemelor ca şi concept a apărut înainte de al Doilea Război mondial, când inginerii care încercau să rezolve probleme complexe au simțit nevoia de a standardiza metodele aplicate.

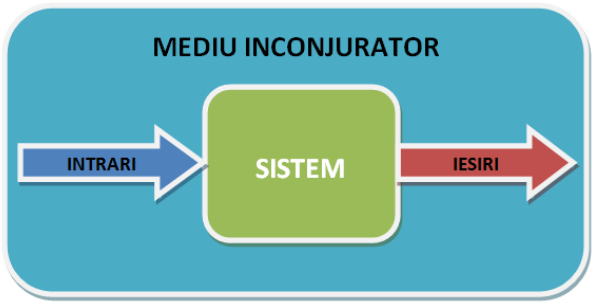
Fără o astfel de proiectare prealabilă şi fără a crea o trasabilitate între cerințe şi componentele sistemului, există riscul ca sistemul dezvoltat să nu îndeplinească total cerințele. Acest fapt este cu atât mai important în cadrul sistemelor embedded de o complexitate ridicată cum sunt cele din industria Automotive.

Ce reprezintă un sistem embedded?

În general, prin noțiunea de sistem se înțelege un set de obiecte/entități care împreună cu interacțiunile dintre ele formează un întreg.

Sistemul este delimitat temporar şi spațial. El poate fi descris prin structura lui, prin scopurile pe care le îndeplineşte şi prin funcționalitățile sale.

Un sistem este influențat de mediul înconjurător, prin intrările sale şi influențează la rândul său mediul înconjurător, prin ieşirile sale.



Există practic, un număr nelimitat de posibilități de clasificare a sistemelor. De exemplu, putem vorbi despre sisteme sociale, sisteme financiare, sisteme organizaționale, sisteme politice, sisteme tehnice, etc.

Un ***sistem embedded*** (embedded system) este un sistem tehnic ce conține un procesor ca element central de decizie şi control. De regulă, el este folosit pentru a controla un proces fizic (electric, mecanic, hidraulic, etc).

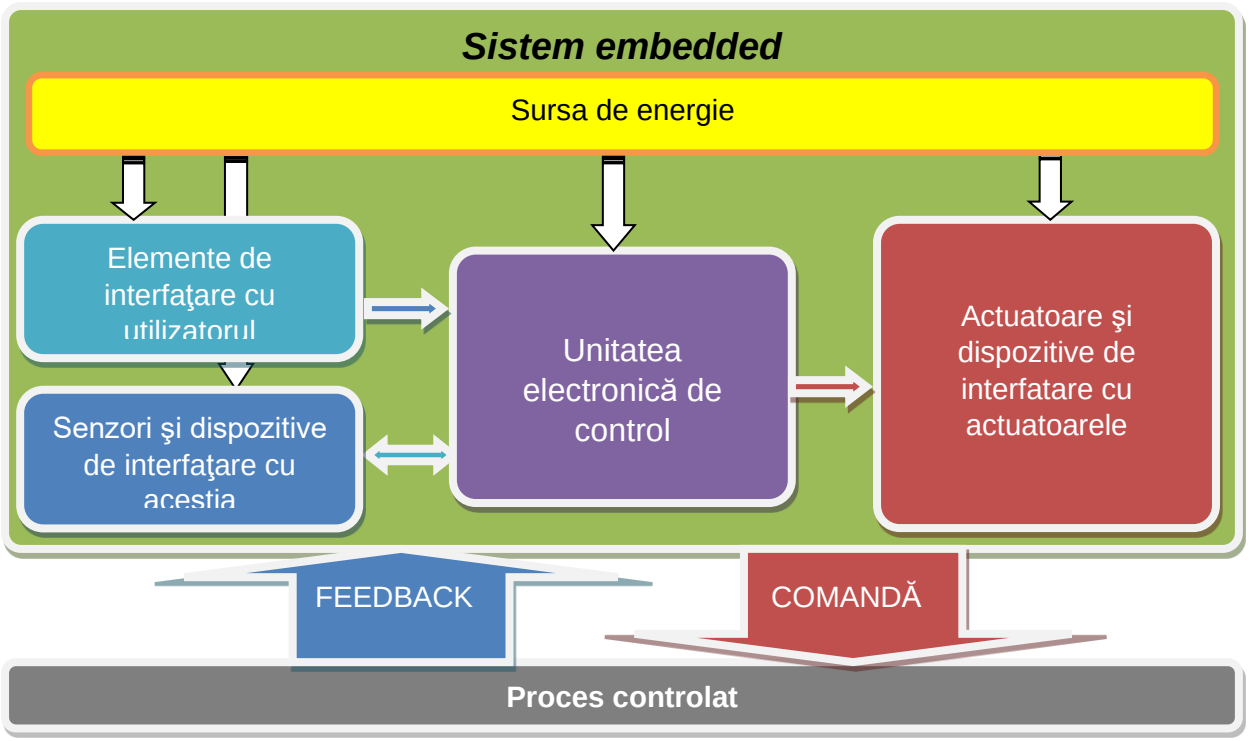
În continuare, prin noțiunea de sistem ne vom referi la sistemele embedded.

Descompunerea unui sistem în blocuri funcționale

Pentru a uşura descrierea soluției tehnice, sistemul este descompus în **componente** sau **blocuri funcționale**. Acestea, la rândul lor pot fi considerate sisteme şi pot fi descompuse în alte componente. În acest fel, pentru a se reduce complexitatea, sistemul complex se „sparge" într-o mulțime de sub-sisteme. Vom descompune până cand vom obține componente atomice (indivizibile), ce vor realiza implementarea fizică a sistemului.

Procesul invers, de re-compunere a componentelor pentru a forma sistemul inițial, pe baza arhitecturii sistemului, se numeşte **integrare**.

La sfârşitul procesului de integrare, sistemul trebuie să funcționeze ca un tot unitar, respectând cerințele specificate.



Componentele principale ale unui sistem embedded

Sursa de energie

Din legea de conservare a energiei aflăm că nici un sistem tehnic nu produce lucru mecanic sau energie (chimică, electrică, mecanică, termică, etc), dacă nu o consumă dintr-o ***sursă de energie***. Mai mult, energia consumată de sistem este întotdeauna mai mare decat energia utilă sau lucrul mecanic pe care sistemul o/îl generează.

În sistemele embedded, sursa de energie este o sursă de curent electric, ce poate fi: o baterie, un acumulator, o sursă de alimentare de la rețeaua de curent alternativ sau o baterie solară.

În alegerea sursei de energie se ține cont atât de: prețul, masa și dimensiunile geometrice, cât și de parametrii sistemului și ai sursei: tensiunea nominală de funcționare, consumul de curent, puterea electrică.

Unitate electronică de control

Unitatea electronică de control, ***ECU*** (Electronic Control Unit) este numit „creierul" sistemului și are abilitatea de a executa un ***program software*** (firmware), care este scris de regulă în ***limbajul C***.

Aceasta este formată dintr-un ***microcontroler*** (mai rar un microprocesor), împreună cu electronica necesară funcționării lui (circuit de clock, condensatori de decuplare pe pinii de alimentare, circuit de reset, etc). Totodată se urmărește simplificarea schemei electronice a sistemului prin implementarea unor strategii și algoritmi complecși (de interfațare cu utilizatorul sau de control) în software.

Microcontrolerul, din punct de vedere electronic, este conectat la restul componentelor sistemului prin intermediul:

- **porturilor** de intrare-ieşire (***D**igital **I**nput*, ***D**igital **O**utput*) sau analogice (***A**nalog **I**nput*),
- **perifericelor** incorporate (***P**ulse **W**idth **M**odulation*, ***U**niversal **S**ynchronous **A**syncronous **R**eceiver **T**ransmitter*, ***C**ontrol **A**rea **N**etwork*, ***L**ocal **I**nterconnection **N**etwork*, ***S**erial **P**eripheral **I**nterface*, ***I**nter-**I**ntegrated **C**ircuit or **I2C***, *Timer*, etc).

În alegerea unui microcontroler se ține cont de:

- resursele microcontrolerului: puterea de procesare, capacitatea memoriei, existența perifericelor necesare;
- costul şi disponibilitatea uneltelor de dezvoltare: programator, *debugger*, compilator pentru limbajul C, **IDE** (***I**ntegrated **D**evelopment **E**nvironment*);
- tensiunea de alimentare, puterea consumată, precum şi existenta unor moduri de funcționare de tip *sleep;*
- data până la care microcontrolerul este produs;
- prețul de achiziție al microcontrolerului.

Senzori şi dispozitive de interfaţare cu aceştia

Orice sistem trebuie să obțină și să interpreteze informații din mediul său înconjurător. Iar un ***senzor*** este un obiect ce poate detecta valoarea unor mărimi fizice din mediul înconjurător (de intrare) și apoi să reacționeze prin modificarea corespunzătoare a stării (ieșirii) sale.

În cazul sistemelor embedded, un senzor este un dispozitiv ce poate detecta mărimi fizice de intrare precum: tensiunea, curentul electric, temperatura, presiunea, radiația sau intensitatea luminoasă, forță, masă, etc; și apoi să reacționeze la ieșire prin modificarea unei ***tensiuni electrice*** (mai rar a unui ***curent electric***). Apoi, prin intermediul porturilor de intrare digitale sau analogice, ieșirea senzorului va fi citită de programul aflat în microcontroler.

Așadar, în proiectarea sistemului embedded trebuie avut grijă ca între senzor și microcontroler să existe ***dispozitivele de interfațare*** cu senzorii. Acestea au rolul de a converti, după o relație cunoscută, tensiunea sau curentul de ieșire al senzorului, într-o tensiune sau curent de intrare compatibil cu porturile microcontrolerului (de regulă [0..5V] sau [0..3.3V] și 10-30mA). Simultan au și rolul de a limita curentul care circulă prin portul microcontrolerului, protejându-l astfel de supra-tensiuni sau supra-curenți ce l-ar putea distruge.

În alegerea unui senzor, se ține cont în general, de următoarele caracteristici:

- tipul de mărime fizică masurată,
- plaja de valori ale mărimii de ieşire a senzorului,
- caracteristica (funcția de tansfer) intrare-ieşire: poate fi liniară, neliniară, tabelară, etc
- rezoluția senzorului (cea mai mică schimbare a mărimii fizice de intrare ce mai poate determina schimbarea mărimii de ieşire),
- costul, disponibilitatea şi complexitatea dispozitivelor de interfațare,
- necesitatea calibrării senzorului,
- prețul senzorului.

Elemente de interfațare cu utilizatorul

Am văzut că senzorii au rolul de a „citi” și a „traduce” parametrii mediului înconjurător astfel încât să fie „înțeleşi” de către ECU. În foarte multe cazuri însă, este necesar ca un sistem embedded să poată fi controlat sau să pună la dispoziție informații pentru un om. Cum oamenii nu pot sesiza variații de tensiuni în plaja [0..5V] sau curenți de ordinul zecilor de mA, este nevoie de utilizarea unor elemente de ***interfațare cu utilizatorul*** (*user interface*). Pentru a controla un sistem embedded oamenii folosesc, de regulă, butoane (cu reținere sau fără), comutatoare cu două sau mai multe poziții, potențiometre, *touch-screen-uri* și altele. Iar pentru a transmite informații către oameni, sistemele embedded folosesc dispozitive de afișare (LCD, LED cu 7 segmente), dispozitive luminoase (LED-uri), dispozitive sonore (difuzor, buzzer) sau cu vibrații.

Pentru alegerea interfeței cu utilizatorul se vor lua în considerare urmatoarele:

- distanța şi vizibilitatea dintre om şi sistem (îl pot vedea? îl pot auzi?);
- usurința de manipulare sau de citire a informațiilor;
- usurința de interfațare cu microcontrolerul;
- prețul elementelor de interfațare.

Actuatoare şi dispozitive de interfațare cu actuatoarele

Majoritatea sistemelor embedded transferă energie în mediul înconjurător prin intermediul lucrului mecanic. Cu alte cuvinte, se „miscă” ele însele sau „miscă” ceva din mediul înconjurator. Un ***actuator*** sau ***element de execuție*** este o componentă ce transformă o formă de energie în lucru mecanic (sau, cu alte cuvine, în miscare). Exemple uzuale de actuatoare folosite în sistemele embedded sunt motoarele electrice, releele, valvele, pompele. Ca şi în cazul senzorilor, pentru a putea controla aceste dispozitive este nevoie de putere electrică mult mai mare decât se poate transfera direct de la microcontroler. De aceea este nevoie de ***dispozitive de interfațare*** cu actuatoarele ce au rolul de a converti semnalul de putere scazută de la ECU în semnal de putere mare ce poate controla actuatorul.

Printre cele mai importante criterii în alegerea actuatoarelor se pot menționa:

- tipul de lucru mecanic (mişcare) necesar: liniară, rotație, ritmică, oprit-pornit;
- tipul de mărime fizică controlată: viteză, debit, curent electric;
- plaja de valori ale mărimii fizice controlate;
- caracteristica intrare-ieşire: liniară, neliniară;
- costul, disponibilitatea şi complexitatea dispozitivelor de interfațare.

Procesul controlat

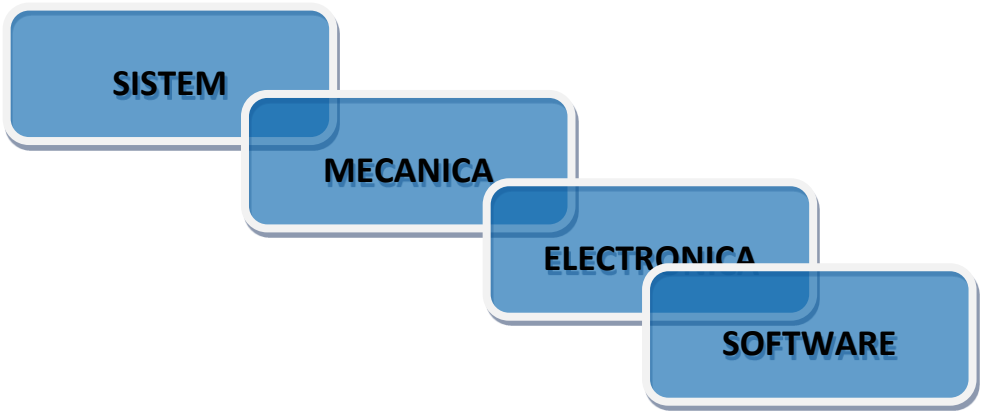
Sistemele embedded sunt construite cu un anumit scop. Produc sau transferă energie, efectuează o mişcare, se mişcă ele însele.

Procesul controlat reprezintă tocmai tipul de acțiune pe care sistemul embedded trebuie să o efectueze pentru a-și îndeplini scopul pentru care a fost construit. Ca și exemple simple de procese controlate putem menționa: deplasarea unui robot dintr-un punct în altul, controlul intensității curentului electric printr-un circuit electric, ridicarea unui ascensor cu viteză constantă, etc. Sistemul embedded acționează asupra procesului controlat prin ***comenzi*** și primește informații despre procesul controlat prin ***feedback***.

Descompunerea unui sistem în arii disciplinare

Complexitatea crescândă a sistemelor embedded a dus la o implementare a acestora, în care sunt implicate arii disciplinare (discipline) precum electronică, mecanică sau software (aceste sunt cele mai folosite în proiectarea sistemelor embedded). Întrebarea cheie este: cum poate fi descris și cum funcționează sistemul din punct de vedere Software (electronic, mecanic)?

Orice componentă a sistemului poate fi împărțită la rândul ei, în arii disciplinare, iar un exemplu poate fi cel prezentat mai jos, în care se poate vedea împărțirea unui senzor de măsurare a temperaturii.



În **disciplina mecanică**, se urmăresc în principal următoarele aspecte: dimensiunile geometrice, masa, comportamentul mecanic la temperatură, vibrații, coroziune, etc.

În **disciplina electronică**, se urmăresc în principal următoarele aspecte: tensiunea nominală de funcționare, consumul de curent și puterea disipată, caracteristicile electrice maxime, comportamentul în funcție de temperatură, caracteristica de funcționare intrare-ieșire, influența zgomotului electro-magnetic, etc.

În **disciplina software**, se urmăresc în principal următoarele aspecte: modul de citire a mărimilor de ieșire, intervalul de valori a mărimii de ieșire, plauzabilizarea valorilor mărimii de ieșire, frecvența de citire a mărimii de ieșire pentru senzori; modul de comandă, plaja de valori de comandă, frecvența de trimitere a comenzii pentru actuatoare; diagnoza defectelor pentru toate componentelor sistemului.

La nivelul fiecarei discipline se va respecta ciclul de dezvoltare (*V-cycle*) pentru implementarea soluției aferente disciplinei respective.

Ca și în cazul descompunerii în componente și în această situație există procesul invers de **integrare** a disciplinelor pentru a forma înapoi soluția sistemului. De această integrare este responsabilă disciplina numită **sistem**.

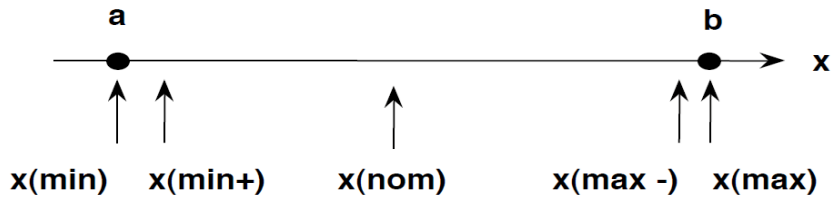
În cadrul proiectului nostru, datorită faptului că utilizăm kit-uri deja fabricate, vom aprofunda aspectele de implementare din punctul de vedere al disciplinei **software** și vom accentua aspectele de interfațare cu celelalte două discipline.

Metode de testare

Testarea presupune stabilirea unui set de inputuri și rezultate așteptate folosite pentru validarea unui produs conform cu specificațiile. Scopul validării este asigurarea că sistemul funționează optim și că poate fi livrat către utilizator, având cât mai puține probleme.

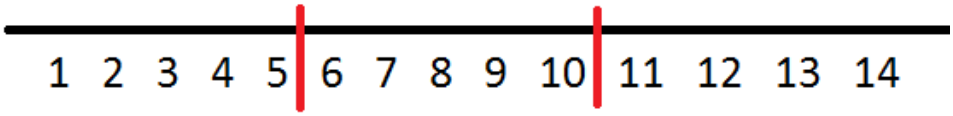
Pentru testarea unui sistem, fără a avea acces la structura software, se folosește testarea de tip **black box.** Metodele de testare incluse în această categorie sunt:

- **Boundary Values Analasys (BVA)** – ideea de bază la această metodă este selectarea variabilei de intrare cu:
  - valoare putin sub minim
  - valoare minimă
  - valoare puțin peste minim
  - valoare nominală (undeva la mijlocul intervalului)
  - valoare aproape de maxim
  - valoare maximă
  - valoare putin peste maxim



- **Equivalence partitioning (EP)** – setul de valori de test este divizat în partiții (intervale) în care comportamentul sistemului este considerat a fi același. Această metodă este potrivită pentru testarea intervalelor. Daca o valoare din interval trece testul, toate celelalte valori vor trece de asemenea. Pe de altă parte, dacă o valoare din interval pică testul, toate celelalte valori vor fi considerate picate.

Exemplu: Sa presupunem că un câmp în care se introduce o parolă, acceptă minim 6 caractere și maxim 10. Pe axa de mai jos este reprezentat numărul de caractere.



Conform axei, se pot forma 3 partiționări 0-5, 6-10, 11-14.

Scenariu de test	Descrierea scenariului	Rezultatul așteptat
1	Se adaugă între 0 și 5 caractere în câmpul parolei	Sistemul nu ar trebui să accepte
2	Se adaugă între 6 și 10 caractere în câmpul parolei	Sistemul ar trebui să accepte
3	Se adaugă între 11 și 14 caractere în câmpul parolei	Sistemul nu ar trebui să accepte

Lucrare practică

Identificarea blocurilor funcționale şi a ariilor disciplinare pentru proiectul curent. Detaliați rolul fiecărui bloc funcțional şi a fiecărei discipline identificate. Verificați trasabilitatea între componentele alese şi cerințele proiectului.