Modelarea Comportamentului Dinamic al Sistemelor Embedded

Sisteme continue. Sisteme Discrete.

Modele deterministe

- Ușor de definit
- Nu pot acoperii elemente non-deterministe: pachete pierdute, întârzieri necontrolabile, defectarea componentelor, zgomot, etc.

- Ce tipuri de modele cunoașteți?

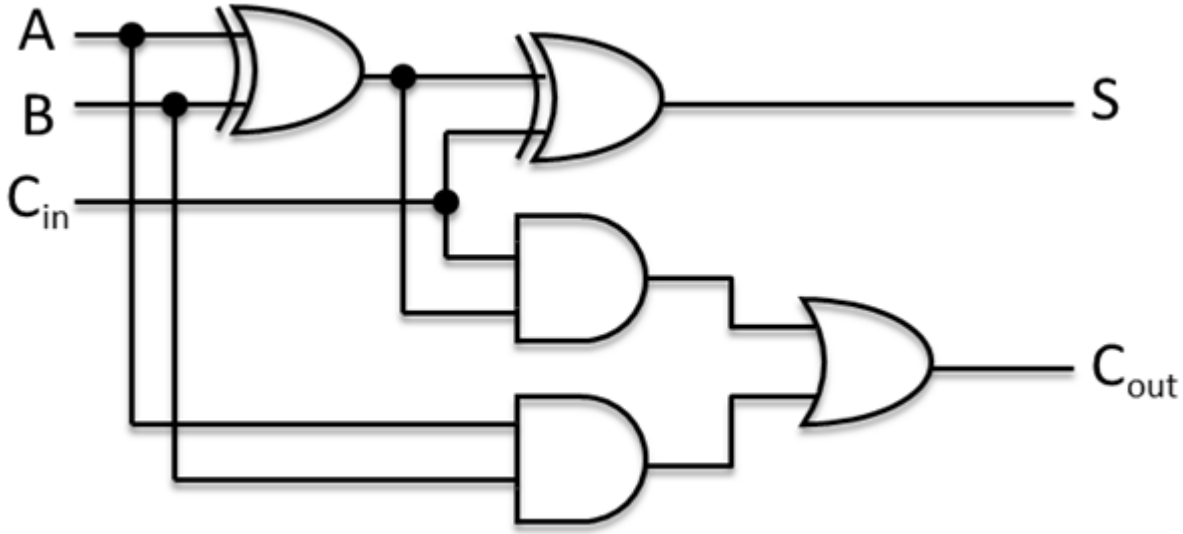
Software-ul este un model

- Programele cu un singur thread sunt modele deterministe
- Software-ul se bazează pe un alt model determinist: Arhitectura setului de instrucțiuni

```
/*****************************************************************************/
/**
* \brief    FlexRay module configuration
* \author   Jaime Orozco
* \param    void
* \return   void
*/
void vfnFlexRay_Init(void)
{
    /* Enable the FlexRay CC and force it into FR_POCSTATE_CONFIG */
    return_value = Fr_init(&Fr_HW_cfg_00, &Fr_low_level_cfg_set_00);
    if(return_value == FR_NOT_SUCCESS)
        Failed(1);   /* Call debug function in case of any error */

    /* Initialization of the FlexRay CC with protocol configuration parameter */
    Fr_set_configuration(&Fr_HW_cfg_00, &Fr_low_level_cfg_set_00);

    /* Initialization of all message buffers, receive shadow buffers
       and FIFO storages */
    Fr_buffers_init(&Fr_buffer_cfg_00[0], &Fr_buffer_cfg_set_00[0]);

    /* Set callback functions */
    /* No FlexRay interrupt is enabled -> no callback function */

    /* Leave FR_POCSTATE_CONFIG state */
    return_value = Fr_leave_configuration_mode();
    if(return_value == FR_NOT_SUCCESS)
        Failed(2);   /* Call debug function in case of any error */

    /* Retrieve the wakeup state */
    wakeup_status = Fr_get_wakeup_state();

    /* Check whether a wakeup pattern has been received */
    if(wakeup_status == FR_WAKEUPSTATE_UNDEFINED)
    {   /* No wakeup pattern has been received */
        /* Initiate wakeup procedure */
        return_value = Fr_send_wakeup();
        if(return_value == FR_NOT_SUCCESS)
            Failed(3);   /* Call debug function in case of any error */
    }


    protocol_state = Fr_get_POC_state();    /* Load current POC state */

    /* Wait till the FR CC is not in the FR_POCSTATE_READY */
    while(Fr_get_POC_state() != FR_POCSTATE_READY)
    {
        protocol_state = Fr_get_POC_state();    /* Load current POC state */
```

Logica digitală

- Logica digitală este și ea un model



Probleme în modelarea sistemelor embedded

- Combinații de modele deterministe sunt non-deterministe
- Sistemele embedded funcționează în medii non-deterministe
- Sunt de folos modelele deterministe?

Modele deterministe în designul sistemelor embeded

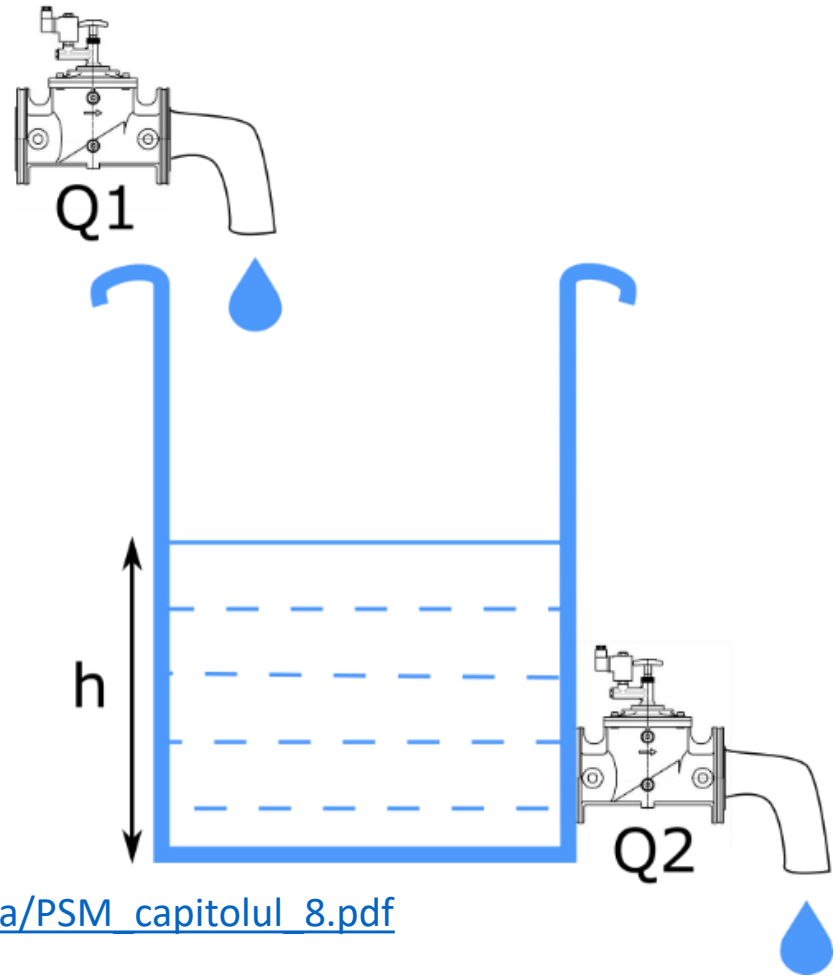
- Nu trebuie să ne concentrăm pe realizarea unor modele deterministe care să descrie comportamenul sistemului embedded
- Trebuie să ne concentrăm pe construirea unor sisteme embedded a căror comportament să-l urmărească pe cel al modelului

Realizarea modelului pentru sisteme dinamice continue

- Identificarea **parametrilor** caracteristici ai sistemului
- Identificarea **legilor care guvernează** sistemul
- **Variante simplificate** ale sistemului
- Bucle de **reglare/control**

Exemplu: Menținerea nivelului într-un rezervor

- Un rezervor de lichide este folosit la alimentarea unui consumator
- Se monitorizează bilanțul masic al sistemului pentru menținerea nivelului lichidului din rezervor prin acționarea unui robinet cu electrovalvă



Valer DOLGA, Proiectarea Sistemelor Mecatronice – curs, http://mec.upt.ro/dolga/PSM_capitolul_8.pdf

Exemplu: Menținerea nivelului într-un rezervor

Identificarea parametrilor:

- Q1 – debitul de intrare (m^3/s)
- Q2 – debitul de ieșire (m^3/s)
- h – nivelul lichidului din rezervor (m)
- m – masa de lichid (kg)
- A – aria secțiunii transversale prin rezervor (m^2)
- V – volumul lichidului (m^3)

Valer DOLGA, Proiectarea Sistemelor Mecatronice – curs, http://mec.upt.ro/dolga/PSM_capitolul_8.pdf

Exemplu: Menținerea nivelului într-un rezervor

Simplificarea modelului:

- Densitatea ρ a fluidului este constantă
- Lichidul este incompresibil
- Rezervorul este poziționat vertical
- secțiunea transversală prin rezervor este circulară și constantă

Valer DOLGA, Proiectarea Sistemelor Mecatronice – curs, http://mec.upt.ro/dolga/PSM_capitolul_8.pdf

Exemplu: Menținerea nivelului într-un rezervor

- Ecuația de bilanț masic al unui sistem – echilibrul masic

$$\frac{dm(t)}{dt} = \sum_i Q_{mi}$$

- Particularizare pentru echilibrul masic al lichidului din rezervor

$$\frac{dm(t)}{dt} = \rho Q_1(t) - \rho Q_2(t)$$

Valer DOLGA, Proiectarea Sistemelor Mecatronice – curs, http://mec.upt.ro/dolga/PSM_capitolul_8.pdf

Exemplu: Menținerea nivelului într-un rezervor

- Relația dintre masa de lichid și parametrii volumetrici ai rezervorului

$$m(t) = \rho V(t) = \rho A h(t)$$

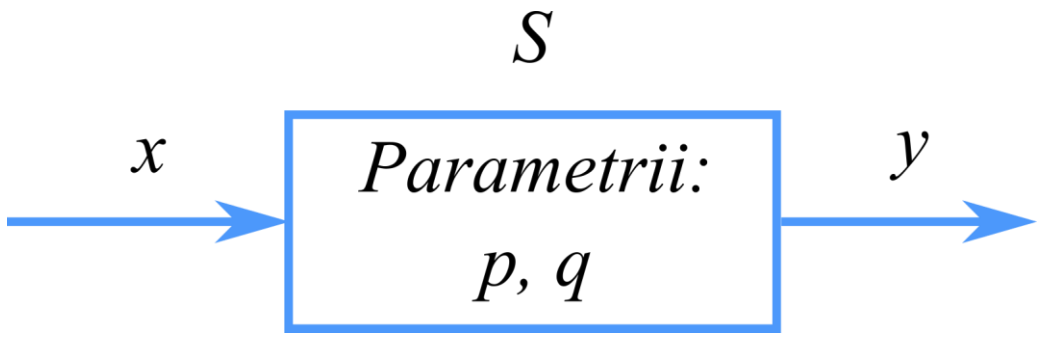
- Modelul matematic devine

$$\frac{dh(t)}{dt} = \frac{1}{A}(Q_1(t) - Q_2(t))$$

Valer DOLGA, Proiectarea Sistemelor Mecatronice – curs, http://mec.upt.ro/dolga/PSM_capitolul_8.pdf

Modele actor

- Modelul matematic al unui sistem poate să fie reprezentat ca un bloc cu intrări x și ieșiri y funcții de forma $x: \mathbb{R} \to \mathbb{R}$, $y: \mathbb{R} \to \mathbb{R}$ și funcția de transfer: $S: X \to Y, X = Y = \mathbb{R}^{\mathbb{R}}$
- Funcția de transfer a sistemului poate să depindă de parametrii. De ex. $S(p, q)$ depinde de cei doi parametrii p și q
- Un astfel de bloc poartă denumirea de **actor**



Modele actor

- **Modelul actor** este un model matematic pentru **calcul concurent** care tratează actorii ca primitive universale de calcul concurent.

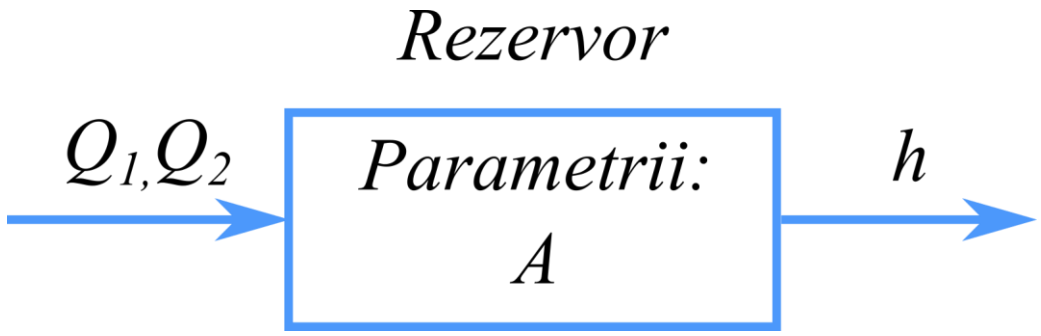
- Pe baza mesajelor primite, un actor poate să realizeze o serie de acțiuni.

- Un actor poate avea intrări și ieșiri multiple

Exemplu: Modelul actor al sistemului cu rezervor

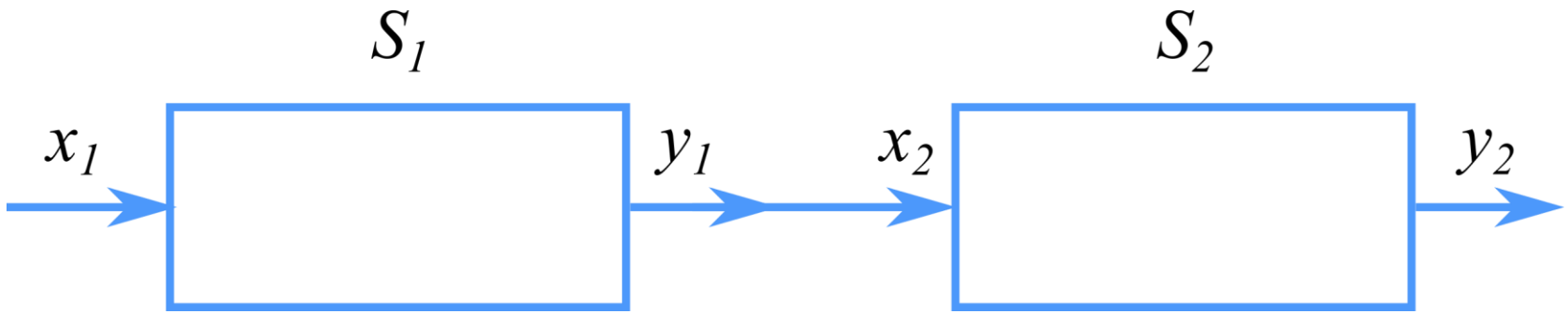
Modelul rezervorului:

- Intrări: Q_1, Q_2
- Ieșire: h
- Parametru: A



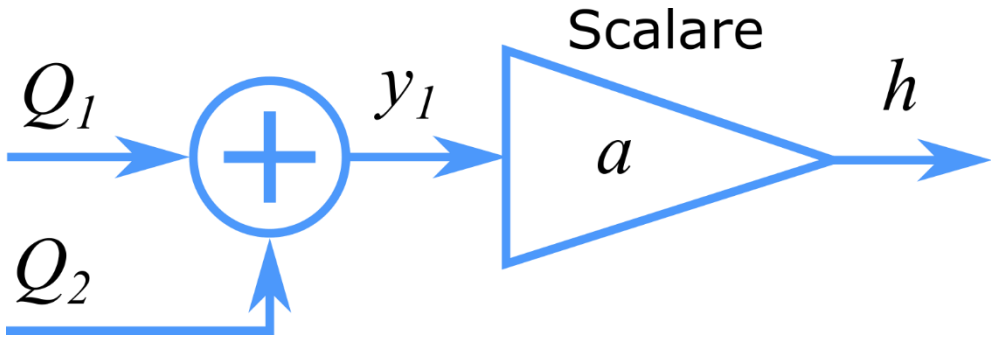
Compunerea actorilor

- Modelele actor pot fii compuse
- **Cascadare** – o formă particulară de compoziție



Compunerea actorilor - exemplu

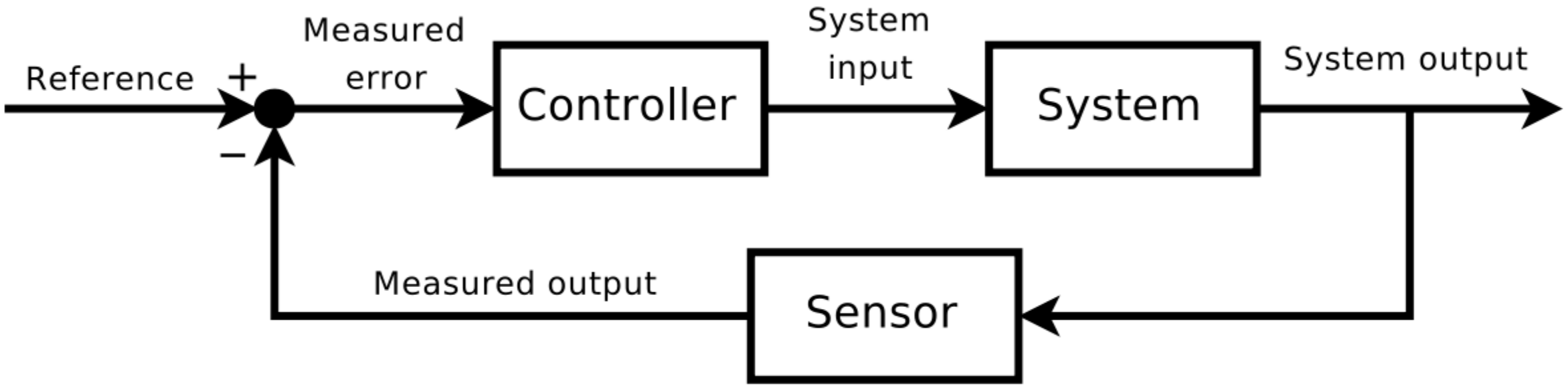
- $y_1(t) = Q_1(t) + Q_2(t)$
- $h(t) = a y_1(t),\ a = \frac{1}{A}$



Poprietăți ale sistemelor

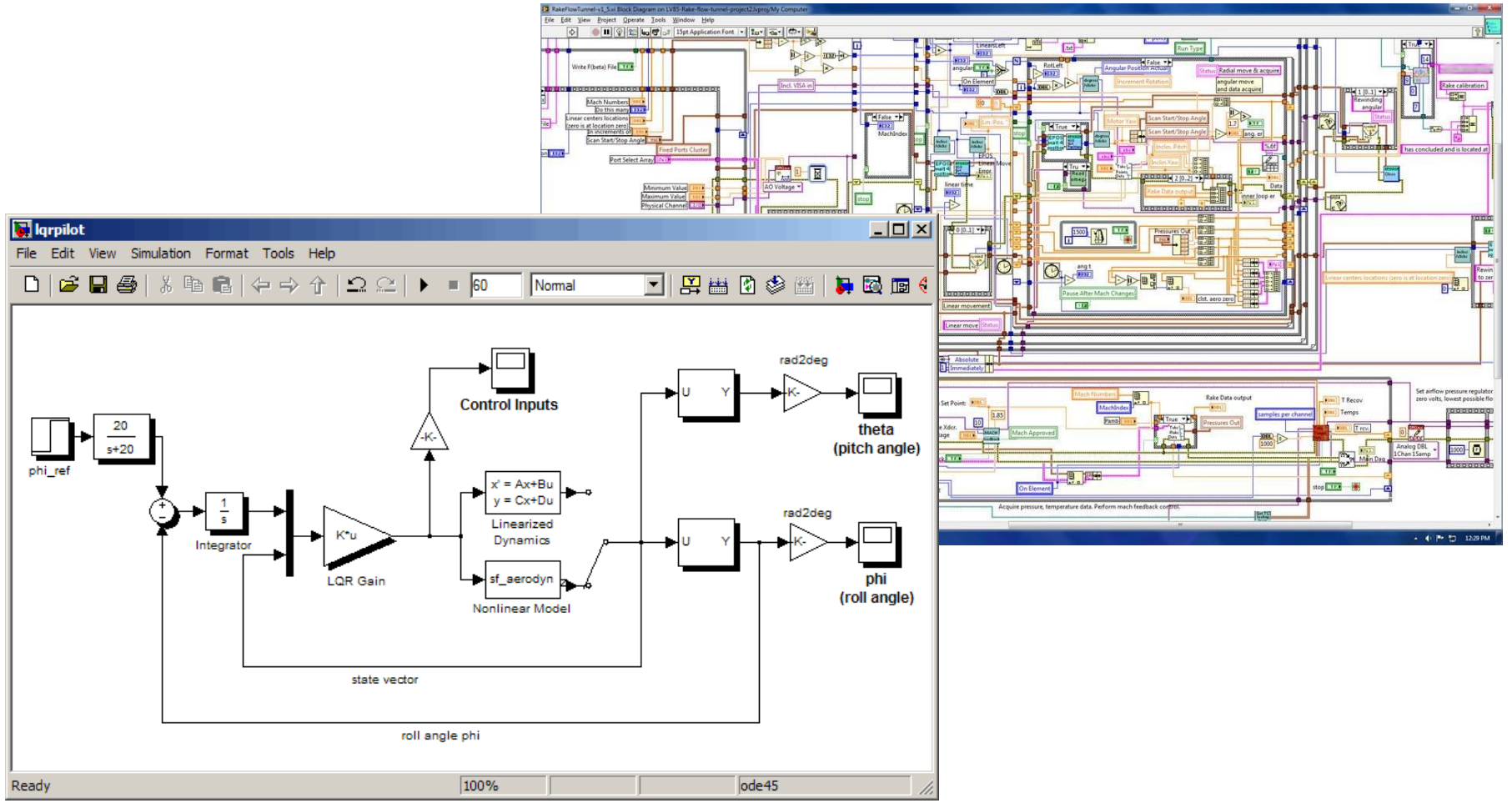
- **Cauzalitate** – ieșirile unui sistem depind doar de intrări curente sau trecute
- **Memoria** – ieșirile unui sistem depind de intrări precedente
- **Liniaritate** – un sistem este liniar dacă satisface proprietatea superpoziției: $S(ax + by) = aS(x) + bS(y)$
- **Invariabilitate în timp** – un sistem este invariabil în timp dacă pentru $S\big(x(t)\big) = y(t)$, decalarea temporală a intrării produce o decalare temporală asupra ieșirii $x(t + \tau) \rightarrow y(t + \tau)$

Bucla de control



Modelearea cu actori

- LabVIEW
- Simulink



Sisteme discrete

- Un **sistem discret** funcționează urmând o **secvență de pași discreți** sau are **semnale ce iau valori discrete**
- Exemple de sisteme discrete: sumator, numărător

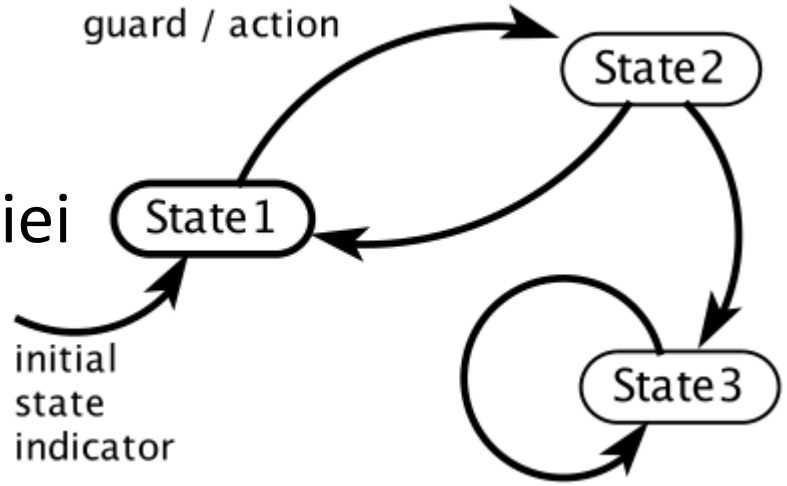
Stări

- **Starea**
  - Condiția sistemului la un anumit moment în timp
  - Codifică informații despre influențe trecute ce afectează reacția sistemului la intrarea curentă
- **Spațiul stărilor** – mulțimea tuturor stărilor în care poate să se afle sistemul
- Un număr finit de stări este preferat. În sisteme software, în general, trebuie să lucrăm cu un set infinit de stări

Tranziții de stare

- **Tranziția de stare** sau **reacția unui sistem** – schimbarea stării interne a sistemului și a felului în care reacționează sistemul la intrări
- Tranzițiile sunt simbolizate prin săgeți ce indică direcția tranziției
- De-a lungul săgeții se notează **condiția de tranziție** (guard) și **acțiunea**

- Condiția de tranziție este un predicat logic
- Acțiunea specifică ieșirile produse în urma tranziției

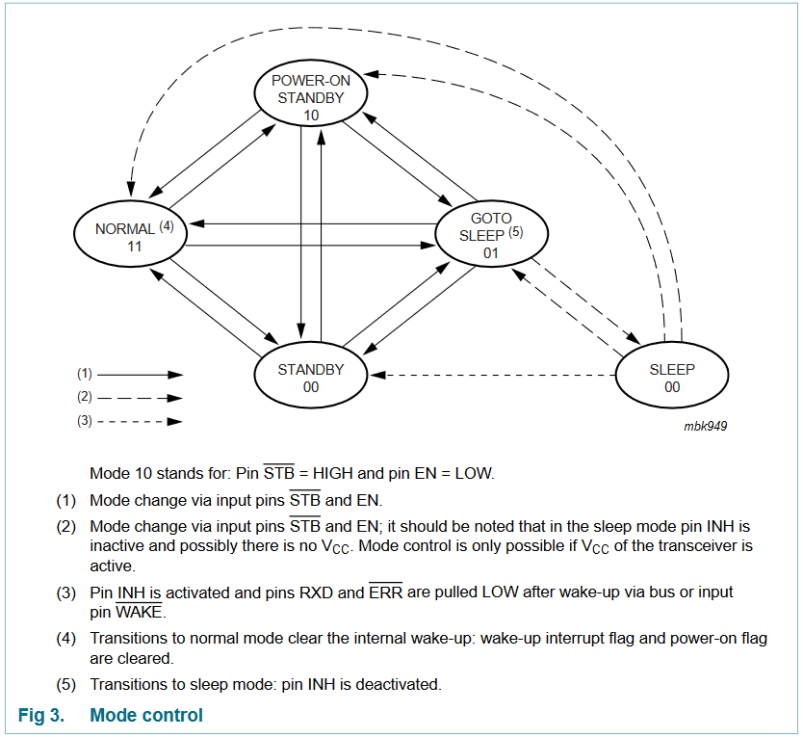


Tranziții implicite

- **Tranziții implicite** – sunt tranziții care nu schimbă starea apar atunci când nu sunt activate alte tranziții sau când condiția de tranziție nu există sau este adevărată
- Un alt tip de tranziții implicite sunt cele care în lipsa unei intrări, nu schimbă starea și nu produc ieșiri

Diagrame de stare, Mașini cu Stări Finite

- **Diagrama de stare** este un model al unui sistem discret care la fiecare tranziție gestionează într-un anumit fel generarea ieșirilor în funcție de intrări
- Ilustrează **stările sistemului** împreună cu **tranzițiile** între stări și **condițiile** ce declanșează tranziția
- **Mașina cu stări finite** – spațiul stărilor este finit
- Putem desena diagrame de stare pentru sisteme cu stări infinite?



Mode 10 stands for: Pin $\overline{STB}$ = HIGH and pin EN = LOW.

(1) Mode change via input pins $\overline{STB}$ and EN.

(2) Mode change via input pins $\overline{STB}$ and EN; it should be noted that in the sleep mode pin INH is inactive and possibly there is no V_{CC}. Mode control is only possible if V_{CC} of the transceiver is active.

(3) Pin INH is activated and pins RXD and $\overline{ERR}$ are pulled LOW after wake-up via bus or input pin $\overline{WAKE}$.

(4) Transitions to normal mode clear the internal wake-up: wake-up interrupt flag and power-on flag are cleared.

(5) Transitions to sleep mode: pin INH is deactivated.

Fig 3. Mode control

Diagama stări transceiver CAN TJA1055

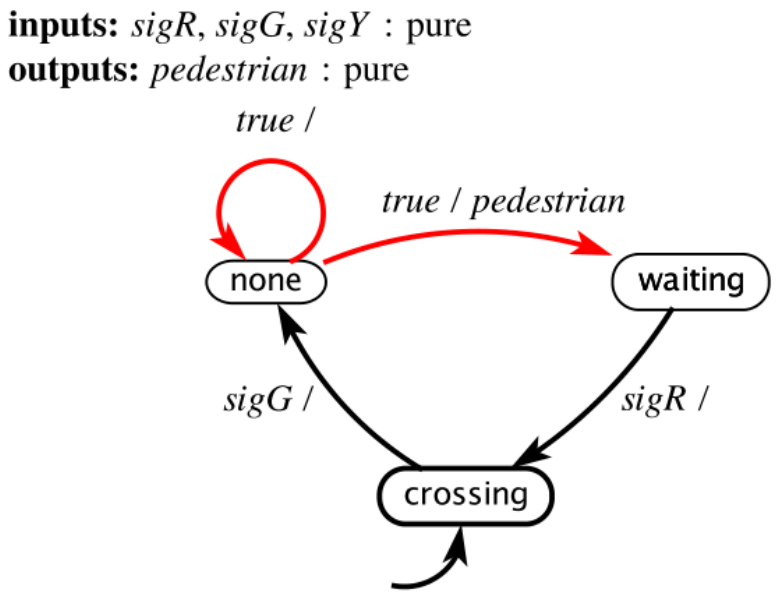
Notația matematică a unei diagrame de stare

- Reprezentarea grafică a unei diagrame de stare se poate transpune într-o notație matematică – **reprezentare formală**
- O astfel de reprezentare formală este un cvintuplu: *(Stări, Intrări, Ieșiri, Funcție_update, Stare_inițială)*
- O tranziție de stare poate fii descrisă prin: $update(s(n), x(n)) = \big(s(n+1), y(n)\big)$

Stări – set finit de stări
Intrări – set al valorilor de intrare
Ieşiri – set al valorilor de ieșire
Funcţia de update – mapează o stare curentă într-o stare următoare și valoarea intrării într-o valoare de ieșire, $update: Stări \times Intrări \rightarrow Stări \times Ieșiri$
Starea inițială – Starea în care se află sistemul la momentul inițial

Nondeterminism

- Dacă orice stare din diagrama de stări are două tranziții a căror condiții sunt la un moment dat adevărate concomitent avem de-a face cu un model nondeterminist
- Un alt exemplu de nondeterminist este dat de mașinile de stări care au mai mult de o stare inițială



Comportament și trace-uri

- **Comportamentul** unui model cu stări finite este o succesiune de pași
- Un **trace** este o înregistrare a intrărilor, ieșirilor și stărilor dintr-un comportament
- Un **arbore computațional** este o reprezentare grafică a tuturor trace-urilor posibile