Modu-kommutazioaren bidezko sistemak ikuskatu eta kontrolatzeko arkitektura baten diseinua

*Nagore Iriondo**[1], *Elisabet Estévez*[2], *Marga Marcos*[1]

[1] Sistemen Ingeniaritza eta Automatika Saila, Ingeniaritza Goi Eskola Teknikoa (UPV/EHU)

[2] Elektronika Ingeniaritza eta Automatika Saila, Goi Eskola Politeknikoa (Jaengo Unibertsitatea)

* nagore.iriondo@ehu.es

Jasoa: 2014-05-14 Onartua: 2014-06-24

Laburpena: Azken urteetako teknologia-aurrerakuntzak, Sistema Hibridoen arloan elkarlanean bildu ditu Kontrol Ingeniaritzaren eta Software Ingeniaritzaren arloetako adituak. Sistema hauen kontrolak dakarren zailtasuna ezaguna da. Sarritan kontrolagailu bakar batekin ezin izaten dira lan-baldintza guztietako eskakizunak bete. Sistema hauek kontrolatzeko eta industrian inplementatzeko bideragarritasunaren ikuspegitik egindako lan honetan, sistema ikuskatu eta kontrolatuko duen arkitektura bat proposatuko da. Oinarrian, modu bakoitzeko kontrolagailu propioak eta euren arteko kommutazioa bideratuko duen logika baten diseinuan datza. Kontrol Ingeniaritzaren ikuspegitik haratago, industria-aplikazio gehienetan prozesu mailako kontrol-teknologiarik erabiliena PLC-a (Programmable Logical Controller) denez, IEC 61131-3 programaziorako estandarrarekin bat datorren inplementazioa proposatuko da.

Hitz gakoak: modu-kommutazioaren bidezko sistemak, ikuskapena, denboran oinarrituriko egikaritzea, IEC 61131-3 estandarra.

Abstract: In the last decades, the technological advances have led experts in the fields of control engineering and software engineering to work jointly in the area of Hybrid Systems. It is well known the difficulty for control them, how usually they do not perform as required if a unique controller is designed for all modes. Related with how to control these systems and the viability of their industrial implementation, this work presents a supervision and control architecture for switched mode systems. It is based on the use of different control algorithms designed specifically for different modes of the system operation and a decision logic that orchestrates the switching among them. Although the architecture can be extended with other improvement mechanisms, the focus here is on its implementation in PLC (Programmable Logical Controller), the

most common control technology used in industrial applications at process level, and being compliant to standards, particularly the IEC 61131-3.

Keywords: switched systems, supervision, time-driven scheduling, IEC 61131-3 standard.

1. **SARRERA**

Prozesuen industrian, sistema gehienak azpiprozesu jarraituz osatuta daude, eta hasi, parametrizatu, birkonfiguratu eta gelditu behar izaten dira. Eragiketa horiei bide emateko kontrol diskretuaren bidez hartzen da erabakia, logika batean oinarrituta. Sistema horietan, beraz, denbora jarraituko eta denbora diskretuko dinamiken arteko elkarrekintza gertatzen da. Horregatik esaten zaie Sistema Hibridoak.

Horien artean, ohikoak dira modu-kommutazioaren bidezko sistemak, non kontrolagailu-multzo bat diseinatzen den eta kontrolagailua aldatzeko erabakia nolabaiteko logikaren bidez hartzen den. Mota horretako irtenbideak industriako hainbat aplikazio arlotan ikus ditzakegu, hala nola energiaren kudeaketan, prozesu termikoetan eta kimikoetan, potentzia-estazioetan, petrolioaren industrian, garraioan, eta abar. Petrolio gordinaren ontzi-aldaketan esaterako, itsasontzien lokailua kokatzeko kontrol erak daude. Giro-baldintzek sortutako deriba indarrek, lokailuak tinko mantentzeko egin behar den indar-konbinazioa baldintzatzen dute. Hori kontutan izan behar da propultsatzaileak kudeatzerakoan. Eguzki-energiaren alorrean ere, hainbat operazio-egoera daude. Energia-iturri nagusia ezin daiteke manipulatu eta egunean zehar ere aldatu egiten denez, sistemak hainbat dinamika erakusten ditu. Energia-iturriak bakarka edo konbinatuta erabil ahal izateak, erabilera birkonfiguratu beharra dakar.

Modu-trantsizioak gertatzean, gehienetan, egin beharreko kontrolagailu-aldaketak *on*/*off* izaten dira (balbulak ireki/itxi, zirkuituak zabaldu/itxi, makinak abiatu/gelditu erakoak), eta horri dagokion kontrola logikoa izaten da. Aldi berean, badira kontrolpean eduki behar izaten diren aldagaiak ere, baina baldintza aldakorren eraginez prozesuaren dinamika aldatzen bada, aldagaion kontrol-modua ere aldatu behar izaten da, eskakizunak betetzeko edo egonkortasuna bermatzeko.

Horrelako sistemak kontrolatzeko erabiltzen diren arkitektura gehienak, bertako kontrolagailuez gain, ikuskatze-estrategia bat daukate. Estrategia horren arabera hartuko da prozesuaren bilakaera ikuskatu eta, antzemandako baldintzen arabera, kontrolagailuak aldatzeko erabakia.

Halere, gehienak aplikazio jakin batzuetarako bereziki diseinatuta daude, eta ez da ematen ez arkitektura beraren formulazio orokorrik ezta beste prozesuetan ezarri ahal izateko argibiderik ere.

Kontrol-aplikazioen garapenean, diseinuaren eta inplementazioaren artean oraindik ere hutsuneak daude, eta horren harira, alderdi ezberdinak jorratu izan ditugu gure lanean. Alde batetik, kontrol-piramidearen behe mailako arazoei begira, modu, dinamika eta begizta-aniztasunak sortzen dituen fenomeno gehigarrien inguruko arazoez jabetu eta konponbidea eskaintzeko asmoa dugu. Esate baterako, ez da hemen ezer esango kontrolagailuak aldatzean seinale ezberdinek sufri dezaketen jauziaz, ez eta alderdi horiek zaindu eta hobetzeko estrategiez, dagoeneko landuak baditugu ere [1]. Beste alde batetik, arkitektura hori industrian erabiliena den PLC kontrolagailuan inplementatzeko erabakia hartu da, mikroelektronikaren aurrerakuntzak eragindako PLC-en espezializazioak baliabideak eskaintzen dituelako kontrol-estrategia aurreratuak edo konplexuak inplementatzeko. Arlo horretan, software-saltzaileak benetan saiatzen ari dira estandarren arabera moldatzen, bereziki IEC 61131-3 PLC-en programazio-estandarrarekin [2,3]. Estandar honek baliabidek eskaintzen ditu hainbat teknologia-arazori konponbidea emateko, eta fabrikatzailearekiko menpetasunik gabeko elementu bezala programatzeko aukera izateko.

Artikulu honetan, azalduko da zein metodologia erabiltzen den modukonmutazioaren bidez sistemen ikuskatzerako eta kontrol-arkitekturaren diseinurako; gero, IEC 61131-3 estandarrak ematen dituen elementuekin inplementatzeko argibideak emango dira.

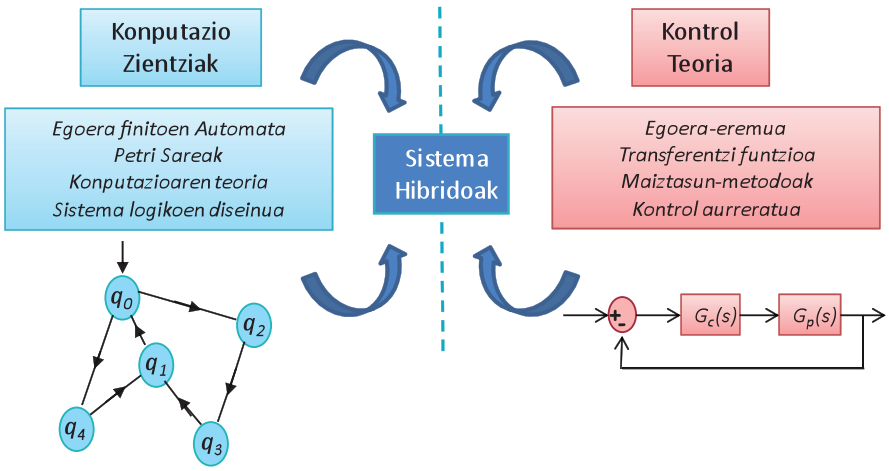
2. MODU-KOMMUTAZIOAREN BIDEZKO SISTEMEN DESKRIBAPENA

Kontrol-sistema diseinatzerakoan, argi dago dinamika mota bien bilakaera kontutan izango duten tresnak beharko direla.

Sistema horiek aztertzeko eta kontrol-sistema seguruak diseinatu eta garatzeko erremintak eskaintzea da Sistema Hibridoen jakintza arloak bilatzen duena [4,5], horrela, kontrol ingeniaritza eta konputazio zientzien antzera kontrol-sistemen alderdi ezberdinak lantzen zituzten diziplina ezberdinetako adituak elkarlanean aritzeko esparrua eskaintzen da (1. irudia). Sistema Hibridoen teoriak, eredu jarraituak eta eredu diskretuak elkarrekin biltzen dituzten modelizazio-formalismoak eta kontrol-teknikak lantzen ditu.

Sarrerak/Irteerak dituen automata hibrido sinplifikatu baten adierazpidea erabiliko dugu prozesuaren dinamika formulatzeko; adierazpideak eredu mota biak semantika matematiko uniforme batez formalismo bakarrean txertatzen ditu [6]:

$$H = (Q, X, V, Y, Init, f, I, E, G, R) \quad (1),$$



1. irudia. Sistema hibridoen ikuspegiak.

non terminoen esanahia hauxe den: $Q= \{q_1, \ldots, q_m\}$ moduak, X egoera jarraituko aldagaiak; V sarrerak eta Y irteerak, $Init \subseteq Q \times X$ hasierako balizko egoerak, $f: Q \times X \times V \rightarrow R^n$ egoeraren bektore-eremua, $I: Q \rightarrow 2^{X \times V}$ inbarianteak, $E \subseteq Q \times Q$ arku edo trantsizioak, $G: E \rightarrow 2^{X \times V}$ zaintza-baldintzak inbarianteak ebaluatu eta trantsizioak bideratzeko eta $R: E \times X \times V$ trantsizio-unean, egoera jarraituarentzako hasierako baldintzak.

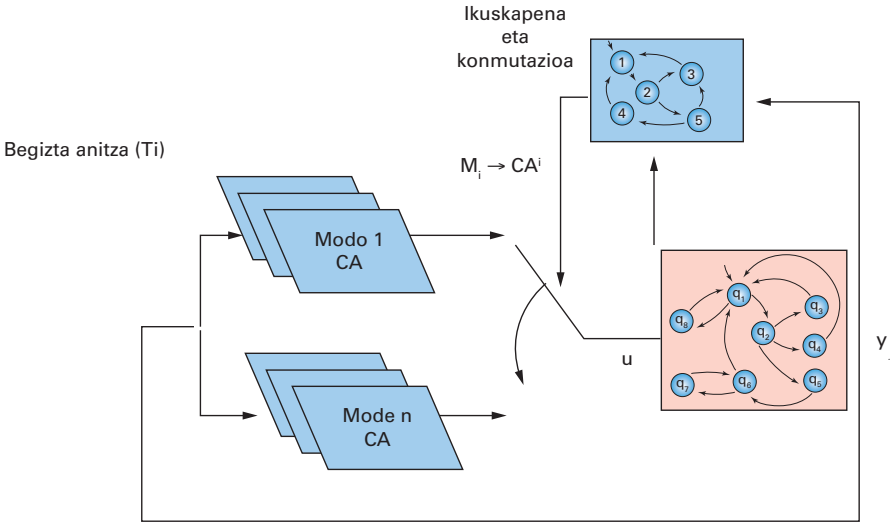
Modu-aldaketak, behe mailako kontrol-begiztak ixten dituzten kontrolagailuak aldatu beharra ekarriko du zenbait kasutan.

3. KONTROL ETA IKUSKAPEN-ARKITEKTURA

Horrelako sistemen kontrol arazoei aurre egiteko, kontrolagailu anitzez osatutako arkitektura bat erabiliko dugu, kontrolagailuen kommutazioaren bidezko estrategia ezagun batean oinarrituta [7]; arkitektura honek ere kontrol-begizten aniztasuna eskaintzen du. Elementu ezberdinek osatzen dute (2. Irudia):

— **Kontrol-begizta anitzek**. Begizta bakoitzak aldagai bat kontrolatuko du, horretarako sarrera-aldagai bat ($U \subseteq V$) manipulatuz. Edozein dela modua, begizta horiek kontrolpean egon beharko dute.
— **Modu-kontrolerako algoritmoak**. Moduei dagozkien begizta-kontrolagailuek osatuta daude.

(CA – Control Algorithm).



2. irudia. Kontrol-arkitekturaren modu anitza eta begizta anitza.

— **Ikuskatzaileak**. Kontrol-arkitekturaren elementu nagusia da, eta uneoro, modu-trantsizioak detektatu eta kontrolagailu berriak hautatzea du funtzioa.

3.1. **Ikuskatzailearen formulazioa**

Ikuskatzailea Mooreren makina bezala definitu da [8]. Modu bakoitzean, dagozkion kontrol-algoritmoen exekuzio-agindua izango da automata horren *irteera*, eta berak bermatuko du laginketa-tarte ezberdinen exekuzioa. Ondorengo adierazpidea erabil daiteke bere formulaziorako:

$$S = (M, M_0, T, \Sigma, \delta, Y, U, L, CA, I, \tau, \Lambda, O) \qquad (2).$$

Formulazio honetan, M moduak dira, dagozkien kontrol-algoritmoen bidez bereiziak, eta $M_0 \in M$ balizko hasiera-moduak. $T \subseteq M \times$ M trantsizioak dira. Σ sarrera-alfabetoa da, trantsizioak bideratzen dituzten gertakizunen multzoa. δ: $M \times \Sigma \rightarrow M$ trantsizio-funtzioa da. Y eta U, kontrolatutako eta manipulatutako aldagaien multzoak dira eta $L \subseteq Y \times U$, kontrol-begiztak. Prozesuaren irteerari dagokion aldagai kontrolatu batek eta prozesuaren sarrerari dagokion aldagai manipulatuak, begizta bat osatzen dute.

CA modu kontrol-algoritmoen multzoa da: $CA = \{CA^1, CA^2, ..., CA^n\} = \{CA^i, i = 1..n \text{ (}modua\text{)}\}$. Eta CA^i begizta-algoritmoz osatuta dago,

$CA^i = \{CA^i_1, CA^i_2, ..., CA^i_l\} = \{CA^i_j, i = 1..n$ (*modulua*), $j = 1..l$ (*begizta*)$\}$. Kontrol-algoritmoaren ezaugarriak, bere konfigurazio-parametroen eta denbora-parametroen balioen bidez ematen dira (*laginketa-tartea* (T^i_j), *konputazio-denbora maximoa* (C^i_j) eta *epea* (D^i_j)). *I*: $M \rightarrow CA$ inbariantea da (kontrol-algoritmoen multzoa).

τ, Λ eta *O* terminoek ikuskatzailearen irteera definituko dute: $\tau = \{\tau^1, \tau^2, ..., \tau^n\} = \{\tau^i|\tau^i_k; i = 1..n, k = 1..k^i\}$ denbora diskretuen sekuentziak dira, eta hauek kontrol-algoritmoen sekuentziak exekutatzeko uneak zehaztuko dute. Modu bakoitzak bere denbora-sekuentzia dauka (τ^i), k^i elementuz osatua. $\Lambda = \{\Delta^1 \subseteq P(CA^i)\}$ irteera-alfabetoa da, moduei dagozkien ekintzen multzoa. Ekintza batek, modu bati dagokion kontrol-algoritmoen sekuentzia definituko du. Eta *O*: $M \times \Sigma \rightarrow \Lambda$ irteera-funtzioa da, modu bakoitzean eta une diskretu bakoitzeko, exekutatu behar diren algoritmoen multzoa adieraziko duena.

3.2. **Ikuskatzailearen diseinua**

Kontrol-arkitekturaren diseinu-metodoak ondorengo informazioa behar du abiapuntu gisa:

— Prozesuaren formulazioa, moduak eta S/I (Sarrerak/Irteerak) interfazea argi definituz.
— Kontrol-begizten definizioa, *sarrera manipulatua-irteera kontrolatua* pareen bitartez.
— Modu bakoitzeko, begizta guztien kontrol-algoritmoak, konfigurazio eta denbora-parametro guztien balioen bidez.

Kontrol-arkitekturaren oinarrizko egitura da lehendabizi identifikatu eta antolatu behar dena, hau da, prozesu-modua (*q*), ikuskatze-modua (*m*), moduen indizea (*i*) eta begizten indizea (*j*). Diseinurako urratsak hauek dira:

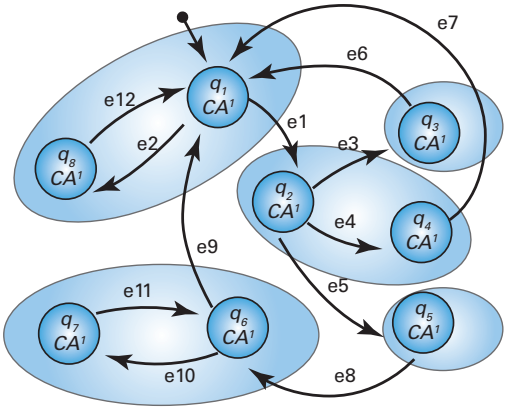
3.2.1. *Ikuskatzailearen moduak eta trantsizioak*

Ikuskatzailearen **moduak** identifikatuko ditugu, prozesuaren moduetatik eta dagozkien kontrol-algoritmoetatik abiatuta, modu kontrol-algoritmo (CA^i) berdina daukaten prozesu-moduak ($q \in Q$) elkartuz.

$$Q \times CA^i \rightarrow M \subseteq P(Q) \ / \ \forall m \in M \rightarrow CA^i = \text{cte}. \qquad (3).$$

Prozesuaren trantsizioetatik, kontrolagailuren bat aldatzea dakartenak izango dira ikuskatzailearen moduen arteko **trantsizioak** (ikus 3. irudia).

$$E \times M \rightarrow T \subseteq E \ / \ \forall e = (q, q') \in E \rightarrow CA^q \neq CA^{q'} \qquad (4).$$



3. irudia. Ikuskatzailearen moduen eta trantsizioen identifikazioa

3.2.2. *Kontrol-algoritmo anitzen exekuzioa*

Ikuskatze modu bakoitzean, begiztak bezainbeste kontrol-algoritmo exekutatu beharko dira, bakoitza bere denbora-parametroek eskatu bezala. Beraz, modu bakoitzeko, denbora diskretuen sekuentzia bat definituko da, algoritmoen exekuzioa planifikatzeko. Algoritmo hauek, bere laginketa tartearen (nT_j^i) eta epearen ($nT_j^i + D_j^i$) artean exekutatu behar dira. Denak periodikoak direnean, denboraren araberako planifikazio estatikorako teknikak erabil daitezke [9].

Horien artean, exekutibo ziklikoen diseinua bereziki egokia da prozesadore-baliabidea konpartitu behar duten algoritmoen planifikaziorako, ziklo nagusi (T_M - hyper-period) bakoitzean planifikatutakoa errepikatu egiten delako [10]. Ziklo nagusia, aktibitate guztien laginketa tarteen multiplo komun txikiena da. Nahikoa da ziklo nagusia planifikatu eta etengabe errepikatzea.

Ziklo nagusia, aldi berean, ziklo sekundarioetan zatitzen da, eta bakoitzean zenbait kontrol-algoritmo exekutatuko dira.

Aktibazio uneak ziklo sekundarioen hasieran egiten direla kontutan izanda, algoritmo bakoitzaren aktibazio jarraitu biren artean, gutxienez ziklo sekundario bat existitu behar da. Bestalde, hobe izango litzateke algoritmoak osorik ziklo sekundario batean sartzea. Hori bermatzeko, ziklo sekundarioa aukeratzerakoan, (5) ekuazioan laburbildutako baldintzak betetzea bilatzen da:

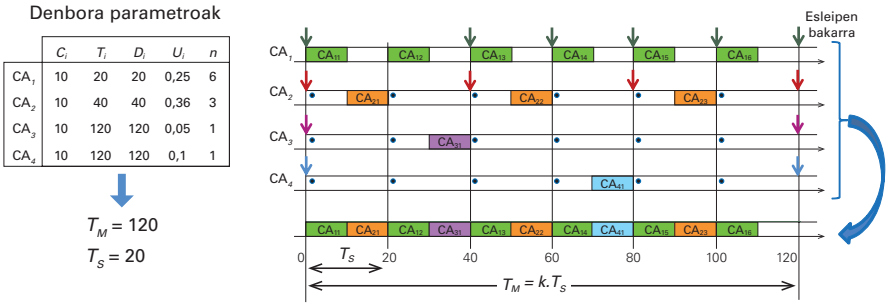
$$\begin{aligned} &T_S^i \le \min(D_j^i) \\ &T_S^i \le \max(C_j^i) \\ &T_M^i = k' \; T_S^i \\ &2T_S^i - zkh\{T_S^i, T_j^i\} \le D_j^i \end{aligned} \qquad (5).$$

Hemen, *zkh:* zatitzaile komun handiena da, i: modua, T_S^i: ziklo sekundarioaren periodoa, T_M^i: ziklo nagusiaren periodoa, j: begizta-kopurua eta T_j^i, C_j^i eta D_j^i begizta-algoritmoen denbora-parametroak. Baldintzok (5) betetzen dituen T_S^i-ren balio handiena hautatuko da ziklo sekundario bezala. Gero, exekutibo ziklikoak lortzeko, algoritmoak iteratiboki esleituko zaizkie ziklo sekundarioei [11]. Erabili den irizpidea hau da:

1. Algoritmoak epe txikienetik handienera ordenatu (edo periodoa $T_j^i = D_j^i$ bada).
2. Emandako ordenari jarraituz, algoritmo bakoitzaren $\frac{T_j^i}{D_j^i}$ aktibazioak balizko ziklo sekundarioei esleitzen zaizkie. Algoritmo baten aktibazioa, $(n-1)T_j^i$ aktibazioarekin batera edo geroago hasten diren zikloei, edo $(n-1)T_j^i + D_j^i$ epearekin batera edo lehenago amaitzen diren zikloei eslei dakizkieke $n = 1..\frac{T_M}{T_j^i}$ = algoritmoaren aktibazioak, n = zenbatgarren ziklo sekundarioa). Gainera, n-garren exekuziorako, denbora nahikoa geratu behar da ziklo nagusian:

$$C_j^i \leq T_S^i - \Sigma C_i \quad \forall C_i \in marko \tag{6}.$$

4. irudian ikus dezakegu lau algoritmoren exekuzioa planifikatzen duen balizko exekutibo zikliko baten adibidea. Ezkerreko taulak erakusten ditu algoritmo bakoitzaren denbora-parametroak, baita bakoitzaren CPU-erabilera ($U_i = C_i/T_i$) eta aktibazio-kopurua (n) ere. Heziek algoritmo bakoitzaren aktibazio uneak adierazten dituzte eta puntu beltzek zein ziklotan eslei daitekeen adierazten dute. Algoritmoak laginketa tarte txikienetik handienera ordenatuz, ziklo sekundario ezberdinetan algoritmoak zelan esleitu diren erakusten da. Erabilera osoaren faktoreak, $U = \Sigma\ C_i/T_i \leq 1$ baldintza bete behar du exekutibo ziklikoa kalkulatu ahal izateko.



4. irudia. Algoritmoen aktibazio-esleipenaren adibidea.

Hainbat esleipen aukera daudenez, algoritmoren baten aktibazioa posible ez denean, berriz atzera egin eta algoritmoak beste era batera esleitu behar da.

Exekutibo zikliko bakoitzak, moduari dagokion denbora diskretuen sekuentzia finkatuko du:

$$\tau = \{\tau^i, \forall i = 1..n, \tau_k^i \in \tau^i \mid \tau_k^i = (k-1)T_S^i, (k = 1..k^i)\} \quad (7).$$

3.2.3. *Ikuskatzailearen irteera*

Modu guztietako ziklo sekundario guztiei egindako esleipenak bilduta, *Irteera Alfabetoa* osatzen da. Eta *Irteera Funtzioak,* modu eta ziklo sekundario bakoitzari Irteera Alfabetoko elementu bat esleituko dio.

Irteera Alfabetoa, moduen irteeren multzoa da, hau da, ziklo sekundario bakoitzean aktibatu behar den begizta-algoritmoen sekuentzia ordenatu bat.

$$\Lambda = \{\Lambda^i, \forall i = 1..n, \Delta_k^i \in \Delta^i \mid k = 1..k^i)\} \quad (8).$$

Behin denbora diskretuak (7) eta irteera alfabetoko elementuak (8) kalkulatuta, *Irteera funtzioak* τ_k^i bakoitzari dagokion Λ_k^i elementua esleituko dio.

$$O: M^i \times (\tau_k^i = nT_S^i \mathrm{mod} k^i) \rightarrow \Lambda_k^i \mid (m = 1..\infty) \wedge k = 1..k^i) \quad (9),$$

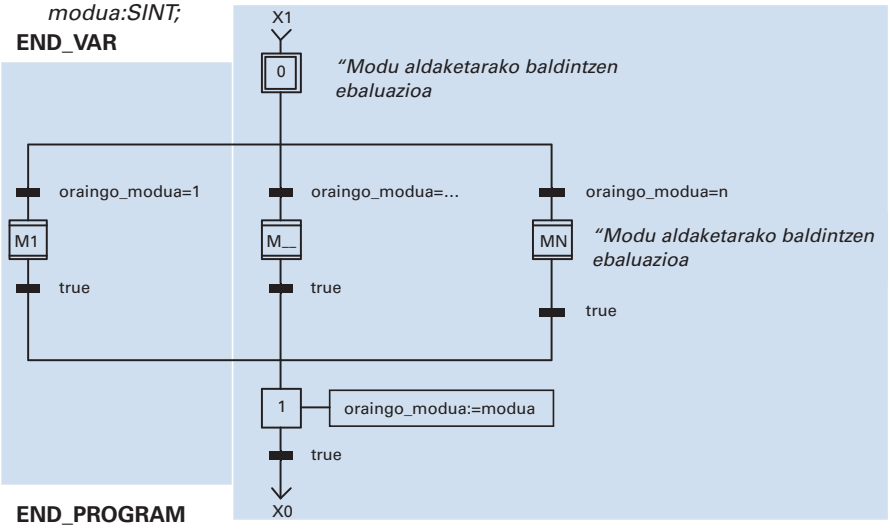
non m aktibazio unea eta k aktibazioaren indizea (ziklo nagusi barruan) diren.

4. IEC 61131-3 ESTANDARRAREKIN BAT DATORREN INPLEMENTAZIOA

Atal honetan arkitektura hau PLC-tan inplementatzeko modu bat proposatuko da. IEC 61131-3 estandarrak eskaintzen dituen elementu eta metodoak erabiliz, aplikazio bat gero berrerabil daitezkeen kode unitatetan egitura daiteke, eta programa baten zati ezberdinak ere laginketa tarte ezberdinekin exekuta daitezke [12].

Arkitektura hau PROGRAM motako $n + 1$ POU-ren (Program Organization Unit) bidez inplementa daiteke. Programa hauetako bat ikuskatzailearena da, modu-aldaketen detekzioa inplementatuko duena (ikus 5. Irudia).

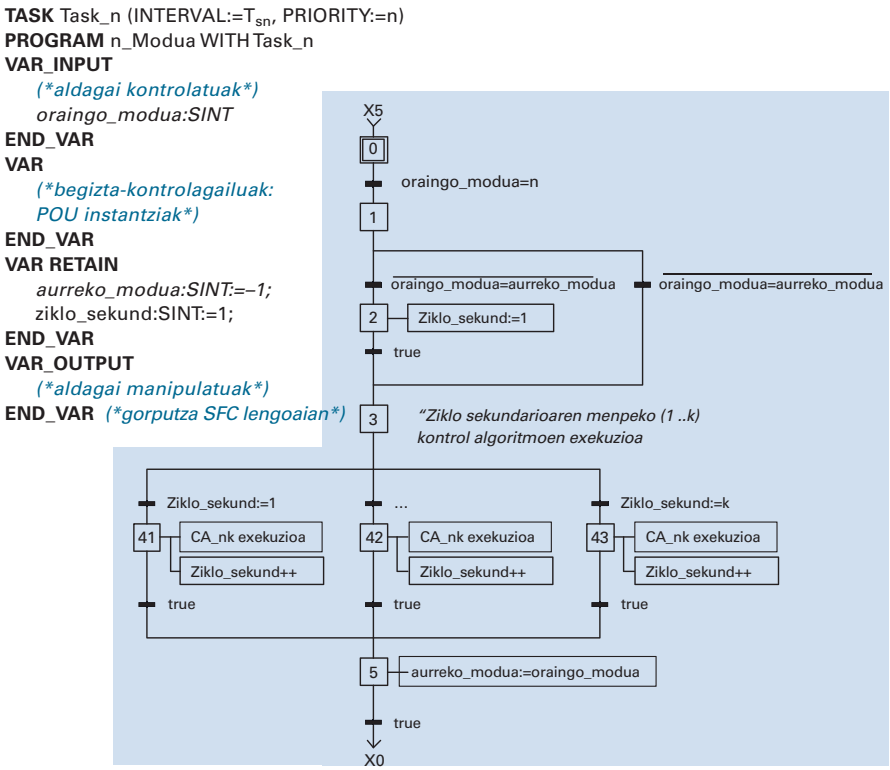
```
PROGRAM Ikuskatzailea
VAR_INPUT
   (*Prozesuan modu-aldaketak antzemateko informazioa*)
END_VAR
INOUT_VAR
   oraingo_modua:SINT;
END_VAR
VAR
   modua:SINT;
END_VAR
```




```
END_PROGRAM
```

5. irudia. Ikuskatzailearen txantiloia IEC 61131-3 SFC lengoaian.

Beste *n* programek, modu bakoitzeko batek, exekuzio periodikoa izan behar dute, eta horretarako bakoitzari **task** bati lotuko zaio; task bakoitzak periodo bezala ziklo sekundarioa izango du. Programa honek, exekuzioa zein ziklo sekundariotan ote dagoen kontrolatuko du eta bere exekutibo ziklikoak esleitu dizkion algoritmoak exekutatuko ditu (ikus 6. Irudia). Kontrol algoritmoak FUNCTION BLOCK motako POU-ren bidez inplementatuko dira.



6. irudia. Moduaren txantiloia IEC 61131-3 SFC lengoaian.

6. **ONDORIOAK**

Dinamika hibridoak dituzten prozesuen kontrolerako, modu eta begizta anitzez osatutako arkitekturak erabiltzeak abantaila nabariak dakartza. Era berean, ondorioztatu dugu egokia dela arkitektura bakun hau diseinatzeko denboraren araberako planifikazioa egitea, betiere begizta-algoritmoen periodoen artean alde handia ez dagoenean [13].

Diseinurako metodologia aurkeztu bada ere, diseinutik inplementaziorainoko hutsunean urratsak emateko aukera ere eskaini digu. IEC 61131-3 estandarrarekin bat datorren inplementazioa, moduak kodea unitate ezberdinetan egituratuz lortu da. Modu-detekzioa ziklikoki exekutatzen da, eta moduetan programatzen da begizta guztien exekuzio periodikoa planifikatzen duen plan estatikoa. Programa hauek *ST* (Structured Text) eta *SFC* (Sequential Function Chart) direlako estandarrak eskaintzen dituen lengoaia bidez inplementa daitezke.

Hurrengo urratsa, exekutatuko den azken kode hori automatikoki sortzea izan da. Kontrol-sistemen analisian eta diseinuan erabili diren ereduak gero inplementazio-kodea sortzeko fasean ere erabil daitezkeela frogatzeko, Software Ingeniaritzaren teknikak erabili dira, bereziki Ereduen Bidezko Ingeniaritza (MDE-Model Driven Engineering) delako paradigmak proposatzen duen metodologia. Meta ereduak bezalako kontzeptu berriak erabiliz, diseinatutako arkitektura honen exekuzio-kodea automatikoki sortzeko prozedura garatu eta erabilgarritasuna frogatu da.

7. **BIBLIOGRAFIA**

[1] IRIONDO N., ESTÉVEZ E., PRIEGO R., MARCOS M. 2013. «Arquitectura multi-controlador con transferencia sin salto para procesos con conmutación de modos». *Revista Iberoamericana de Automática e Informática Industrial RIAI*, ***10 (2)***, 204-215.

[2] INTERNATIONAL ELECTROTECHNICAL COMMISSION. 2003. *IEC International Standard IEC 61131-3 Programmable Controllers, Part 3: Programming Languages*.

[3] EELCO VAN DER WAL (2009). «PLCopen». *IEEE Industrial Electronics Magazine*, **3**.

[4] NETWORK OF EXCELLENCE HYCON. 2004. *Hybrid Control: Taming Heterogeneity and Complexity of Networked Embedded Systems*. European Project FP6-IST-511368, http://www.ist-hycon.org/

[5] LUNZE J. eta LAMNABHI-LAGARRIGUE F. (eds.). 2009. *Handbook of Hybrid Systems Control. Theory, Tools and Applications*. Cambridge University Press.

[6] LYGEROS J., JOHANSSON K. H., SLOBODAN N. S., ZHANG J. eta SASTRY S. 2003. «Dynamical properties of hybrid automata». *IEEE Transactions on Automatic Control*, **48 (1)**, 2-17.

[7] LIBERZON D. 2003. «*Switching in systems and control*». Birkhäuser, Boston. Ch. III, 75-76.

[8] HOPCROFT J. E., MOTWAMI R. eta ULLMAN J. D. 1979. *Introduction to automata theory, languages, and computation*. Reading, MA: Addison-Wesley.

[9] LIU J.W.S. 2000. *Real-time Systems*. Prentice Hall.

[10] ZAMORANO J., ALONSO A., DE LA PUENTE J. A. 1997. «Building safety-critical real-time systems with reusable cyclic executives». *Control Engineering Practice*, **5 (7)**, 999-1005.

[11] BURNS A., HAYES N., RICHARDSON M.F. 1995. «Generating feasible cyclic schedules». *Control Engineering Practice*, **3 (2)**, 151-162.

[12] KARL-HEINZ J., TIEGELKAMP M. 2001. «IEC 61131-3: Programming Industrial Automation Systems». *Springer. Ch. 2*, 47, Ch. 6, 233-236.

[13] IRIONDO N., ESTÉVEZ E. eta MARCOS M. 2012. «Automatic generation of the supervisor code for industrial control systems». *IEEE Transactions on Industrial Informatics*, **9 (4)**, 1868-1878.