XLIV Jornadas de Automática 2023

Modelo digital para definir y emular automatismos secuenciales

Illana, S.*, Sánchez, A.*, Estévez, E.*, Gómez, J.*, Gámez, J.*

* *Departamento Ingeniería Electrónica y Automática, Escuela Politécnica Superior de Jaén, Universidad de Jaén, Campus las Lagunillas s/n, 23071, Jaén, España.*

To cite this article: Illana, S., Sánchez, A., Estévez, E., Gómez, J., Gámez, J., 2023. Digital Model for Defining and Emulating Sequential Automatisms, XLIV Jornadas de Automática, 801-806. https://doi.org/10.17979/spudc.9788497498609.801

Resumen

La importancia de un diseño maduro y mantenible del software de automatización en los sistemas ciber-físicos de producción (CPPS) aumenta continuamente, ya que una proporción cada vez mayor de la funcionalidad del sistema se implementa mediante software. Como consecuencia, la complejidad del sistema se desplaza cada vez más hacia el lado del software. Características de calidad como la flexibilidad, la mantenibilidad y la extensibilidad, cuya importancia está aumentando en el contexto de la Industria 4.0, son cada vez más difíciles de lograr. Para conseguir dichos retos, se ha de partir de un diseño software sólido. En este sentido, este trabajo, propone una aproximación basada en modelos para diseñar automatismos secuenciales siguiendo GRAFCET. Un modelo digital del automatismo secuencial conteniendo el diseño software de automatización va a permitir testear el automatismo con la parte operativa real o emulada, de una manera totalmente independiente al PLC.

Palabras clave: Industria 4.0, Modelo Digital gráfico, Automatización industrial.

Digital Model for Defining and Emulating Sequential Automatisms

Abstract

The importance of a mature and maintainable design of automation software in cyber-physical production systems (CPPS) is continuously increasing, as a growing proportion of system functionality is implemented by software. As a consequence, system complexity is increasingly shifting to the software side. Quality characteristics such as flexibility, maintainability and extensibility, which are becoming increasingly important in the context of Industry 4.0, are becoming more and more difficult to achieve. In order to meet these challenges, a solid software design has to be the starting point. In this sense, this paper proposes a model-based approach to design sequential automation following the GRAFCET standard. A digital model of the sequential automatism containing the automation software will allow testing the design of the automatism with the real or emulated operative part, in a completely independent way from the PLC.

Keywords: Industry 4.0, Digital Graphic Model, Industrial Automation.

1. Introducción

Los sistemas ciber-físicos de producción (CPPS-Cyber-Physical Production Systems) representan un tipo especial de sistema mecatrónico, caracterizado por su complejidad hardware y software. Los CPPS desempeñan un papel fundamental en el paradigma de la Industria 4.0 (I4.0), ya que proporcionan autonomía y adaptabilidad a los sistemas de los que forman parte, contribuyendo a aumentar la eficiencia de los procesos de fabricación (Zanero 2017). Por este motivo, no es difícil encontrar un gran número de trabajos de investigación dedicados al diseño y desarrollo de CPPS (S. Karnouskos 2020), (Salafia 2020), (R. S. Peres 2018). Una parte importante de estos esfuerzos se centra en el problema de la integración entre las partes física y virtual de los CPPS, y algunos de estos trabajos proponen soluciones ad-hoc orientadas a la integración de un activo (o tipo de activo)

específico en un caso de estudio concreto (A. D. Neal 2021), (J. de las Morenas 2017).

Este trabajo se centra en el diseño y desarrollo de software de automatización en los CPPS, que está fuertemente influido por otras disciplinas, e.g. la eléctrica/electrónica y la mecánica, donde se sigue un proceso de desarrollo secuencial, en el que el desarrollo de software suele ser el último paso (S. Biffl 2017). Los CPPS suelen tener una vida útil de hasta varias décadas y, durante ese tiempo, el software de automatización debe poder mantenerse y adaptarse a requisitos cambiantes (B. Vogel-Heuser 2015). Sin embargo, la gestión sistemática del cambio apenas se aplica en la industria, y el software sigue reutilizándose principalmente mediante el método de copiar, pegar y modificar, lo que a la larga da lugar a un software heredado históricamente crecido y propenso a errores (J. Fischer 2018).

El software de automatización en CPPS tiene que hacer frente a condiciones límite que difieren mucho de la programación clásica en lenguajes de alto nivel en informática, lo cual dificulta la aplicación de enfoques bien establecidos en dicha disciplina. El desarrollo eficiente de software, incluidas las estrategias adecuadas de reutilización y modularización, es esencial para que las empresas implementen las tecnologías emergentes que surgen en el contexto de la I4.0 y, de este modo, sigan siendo competitivas en un mercado globalizado. Sin embargo, las arquitecturas de software modularizadas son difíciles de conseguir debido a la heterogeneidad del software y a la falta de mejores prácticas y directrices de aplicación general. En este sentido, este trabajo, propone una aproximación basada en modelos (Cretu, Model Driven Engineering of Information Systems: Principles Techniques and Practice 2021) para diseñar automatismos secuenciales siguiendo la norma IEC 61131-3 (Karl Heinz John, Michael Tiegelkamp, 2010).

La herramienta propuesta, se centra en ofrecer los mecanismos y pautas necesarias para que, a través de un modelo digital del automatismo secuencial, se permita testear el diseño del proyecto de automatización con su correspondiente parte operativa real o emulada, de una manera totalmente independiente al PLC. Únicamente será necesario tener conocimiento de las I/Os de la parte operativa. Además, la herramienta propone diferentes alternativas estándares para comunicar el modelo digital del proyecto de automatización con las I/Os (Modbus (MODBUS s.f.), OPC-UA (OPC-UA s.f.) y MQTT (MQTT s.f.).

El resto del artículo está organizado de la siguiente manera: el apartado 2 describe los módulos principales de la herramienta MODAS (MOdelo Digital para Automatismos Secuenciales). El apartado 3 se centra en la sincronización y comunicación del GRAFECT diseñado con la parte operativa. Para ello, MODAS emula el Ciclo-Scan de un autómata programable. El apartado 4 hace uso de la herramienta a través de un caso de estudio. Finalmente, el apartado 5 recoge las conclusiones.

2. Modelo Digital para Automatismos Secuenciales (MODAS)

La plataforma MODAS ofrece soporte para la definición y posterior testeo del diseño de un proyecto de automatización independientemente al PLC. Como se ha comentado anteriormente, este servicio es de gran interés en la comunidad científica pero también para la comunidad educativa ya que permite al alumnado diseñar y testear proyectos de automatización sin necesidad de especializarse en la codificación que ya es dependiente del autómata programable.

Este apartado describe la plataforma MODAS la cual está fundamentada en la MDE y software de código abierto. En concreto, su núcleo está programado en Java y tiene la siguiente estructura:

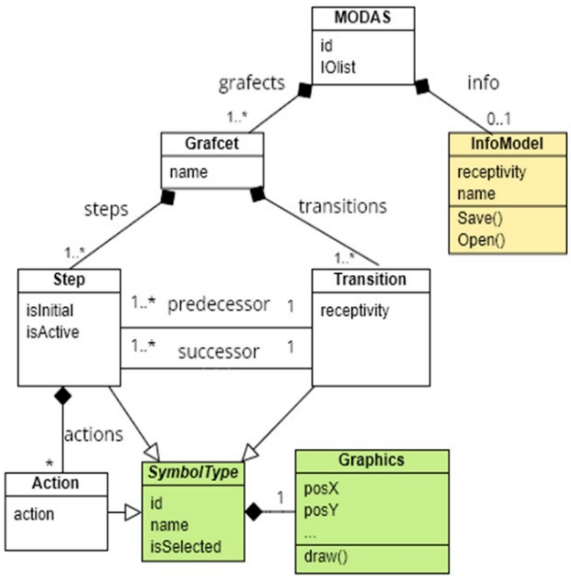

Figura 1Estructura General de MODAS

Como se aprecia, parte del software está centrado en procesar el léxico básico de un GRAFCET:

- Etapa (*Step*): representa cualquier situación estable de un sistema. A una etapa puede(n) ir asociada(s) acción(es). Una acción es una orden la cual puede implicar activación de Salidas del sistema o activación/actualización de recursos internos como por ejemplo temporizadores o contadores
- Transición permite definir la evolución o secuencia del automatismo. Tiene asociada una condición que cuando su valor es verdadero valida la evolución de una etapa a la siguiente. Por eso, una transición además tendrá etapa(s) predecesor(as) y sucesor(as). En función de su número se pueden definir secuencias simples o más complejas:
    - Lineal: cuando la transición presenta una evolución simple, es decir, tiene una etapa predecesora y otra sucesora.
    - Divergencia Simultaneidad: cuando una transición tiene una etapa predecesora y múltiples sucesoras.
    - Convergencia Simultaneidad: cuando una transición tiene múltiples etapas predecesores para una sola etapa sucesora.

- Divergencia Selectora: cuando una etapa aparece como predecesora de varias transiciones.
- Convergencia Selectora: cuando una etapa aparece como sucesora de múltiples transiciones.

La unión entre la automatización de un sistema y la planta en sí a controlar se encuentra en la identificación de las Entradas y Salidas del sistema de control. Esta tarea es previa a la definición del automatismo y por tanto necesaria para la plataforma MODAS. Por ello, cuando se defina un nuevo proyecto habrá que indicarle dicha información. Se ha seleccionado la tecnología XML (eXtensible Markup Language), en concreto un fichero de marcado para almacenar dicha información. La Figura 2 presenta el léxico y sintaxis que debe seguir para que sea procesable por MODAS. Como se puede apreciar, es en dicho fichero donde se fija el protocolo de comunicación entre el modelo digital del proyecto de automatización y la parte operativa.

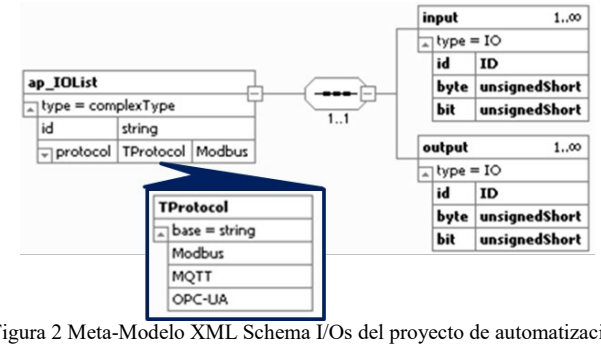

Figura 2 Meta-Modelo XML Schema I/Os del proyecto de automatización

Actualmente MODAS soporta: Modbus, MQTT y OPC-UA. Las entras y salidas se caracterizan por su identificador y su dirección de memoria (byte y bit). La interfaz de la herramienta MODAS permitirá al usuario seleccionar el fichero y cuyo path completo queda almacenado en el atributo IOList de la clase MODAS ilustrada en Figura 1.

La plataforma procesa la información de dicho fichero con objeto de guiar y dar soporte al usuario a la hora de definir las acciones vinculadas a una etapa y las receptividades de las transiciones. En cuanto a las acciones destacar que éstas pueden ser externas o internas. En el primer caso, el usuario podrá vincular únicamente la activación de salidas del proyecto de automatización. Las acciones internas se corresponden a la activación de recursos como por ejemplo temporizadores o contadores (carga, reseteo, actualización). De igual manera, a la hora de definir la condición (receptividad) de una transición, la herramienta permitirá utilizar una combinación lógica de entradas al sistema de control o consultas a recursos internos (por ejemplo, si ha pasado un tiempo o situación de un contador).

MODAS se basa en la librería Processing (Processing s.f.) para dar soporte a la parte gráfica y por tanto definición del GRAFCET. Se trata de un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, diseñado para el desarrollo del arte gráfico, para las animaciones y aplicaciones gráficas de todo tipo. Processing se utiliza como medio para la enseñanza y producción de proyectos multimedia e interactivos. Además, ofrece la posibilidad de generar un ejecutable para las diferentes plataformas Mac OS, Windows o Linux e incluso ofrece soporte para desarrollar aplicaciones móviles gracias a la SDK que ofrece para Android. El núcleo de MODAS implementa esta parte a través de las clases remarcadas en verde de la Figura 1.

Todo elemento de GRAFECT ha de estar localizado en el gráfico (*posX*, *posY*) y el método *Draw()* será dependiente del símbolo a dibujar (etapa, transición o acción). Algo común a todos ellos es que pueden estar o no seleccionados. En el caso de las etapas, se distinguen gráficamente la etapa inicial del resto (*isInitial*) y cuando en ejecución una etapa está activa o no (*isActive).* La Figura 3 ilustra cómo varía su representación gráfica en cada caso. Siguiendo la nomenclatura GRAFCET se distingue un símbolo para la tapa inicial y otro para el resto de etapas. Además, se ha barajado la posibilidad re-colorear las líneas de los símbolos para diferenciar cuándo se encuentra seleccionada (rojo) durante el diseño y cuándo se encuentra activa (verde) en simulación/ejecución. El método responsable de llevar todo ello a cabo es el método *Draw* de las etapas (véase Figura 1).

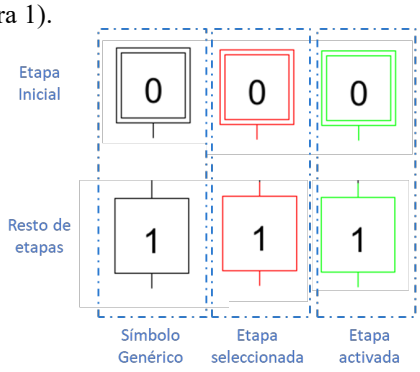

Figura 3 Simbología de las etapas en un GRAFCET

La Figura 4 ilustra dicho método donde se puede apreciar la configuración del color de la línea del gráfico en función si se trata una etapa que se encuentre seleccionada o activa; la definición del símbolo en función de si se trata de una etapa inicial o no y cómo dibujar las acciones vinculadas.
Además del soporte gráfico, MODAS guía y orienta al usuario a la hora de diseñar un GRAFCET. Tiene implementadas las reglas de sintaxis, e.g. no permite definir dos etapas seguidas...De igual manera, limita qué puede ser considerado acción y qué puede participar en la definición de una receptividad. Para ello, maneja un modelo de información que contiene todo lo necesario. Se ha determinado utilizar ficheros de Marcado (XML) como modelos de información. La Figura 6 presenta el Meta-Modelo en XMLschema que fija el léxico y sintaxis que siguen los modelos de información manejados internamente por la herramienta MODAS. Gracias a estos ficheros, MODAS además de definir nuevos proyectos permite abrir ya existentes.

```
public void Draw(PApplet root) {
    root.pushMatrix();
    root.translate(x: posx, y: posy);
    root.noFill();
    root.strokeWeight(weight: grosor);
    root.rectMode(mode: CENTER);
    if (isSelected)
        root.stroke(v1: 255, v2: 00, v3: 00);
    else if (isActive)
        root.stroke(v1: 00, v2: 255, v3: 00);
    else
        root.stroke(rgb: 0);
    root.rect(a: 0, lado2 + transicion, c: lado, d: lado);
    switch(isInitial) {
        case INICIAL:
            root.rect(a: 0, lado2 + transicion, lado - 10, lado - 10);
            break;
        case ETAPA:
            root.line(x1: 0, y1: 0, x2: 0, y2: transicion);
            break;
        case MACRO:
            break;
    }
    root.line(x1: 0, lado + transicion, x2: 0,
            transicion + lado + transicion);
    root.fill(rgb: 0);
    root.textSize(size: fontsize);
    root.textAlign(alignX: CENTER, alignY: CENTER);
    root.text(str: nombre, x: 0, lado2 + transicion);
    root.translate(x: lado2, y: transicion);
    boolean primeraAccion = true;
    for (Accion a : acciones) {
        root.translate(x: 0, y: lado2);
        a.Draw(root, inicial: primeraAccion);
        primeraAccion = false;
    }
    if (acciones.size() > 1)
        root.line(acciones.get(index: 0).getSeparacion() / 2,
                -(acciones.size() - 1)* lado2,
                acciones.get(index: 0).getSeparacion() / 2, y2: 0);
    root.popMatrix();
}
```

Fijar el color de línea

Símbolo

Invocar acciones asociadas a la etapa

Dibujar

Figura 4 Visualización de una Etapa en MODAS

```
public void CicloScan(){
    for (Entrada e : listaEntradas.getEntradas()) {
        e.Update();
    }
    for (Transicion t : transiciones) {
        t.evaluaTransicion(lista: listaEntradas.getEntradas());
    }

    for (Accion a : listaAcciones.getListaAcciones()) {
        a.Update();
    }
}
```

Figura 5 Ciclo Scan del Autómata Simulado

Finalmente, para testear el diseño de proyecto de automatización, es imprescindible emular el Ciclo-Scan de un Autómata y testearlo con la planta real. En este caso, se dispone del método *Simular* encargado de ello. Tal y como sucede en el Ciclo Scan primero se leen las entradas, posteriormente se ejecuta el diseño de usuario y finalmente se actualizan las salidas. La Figura 5 ilustra cómo se ha implementado en MODAS.

La ejecución del GRAFCET se centra a través de las transiciones. Solamente se comprobará la receptividad de las transiciones si su(s) etapa(s) predecesora(s) está(n) activa(s). En tal caso, si dicha condición se cumple, se franqueará dicha transición, lo cual implica activar su(s) etapa(s) sucesoras y desactivar la(s) predecesor(as).

Por último, conocido el nuevo estado del automatismo se han de actualizar las salidas correspondientes.

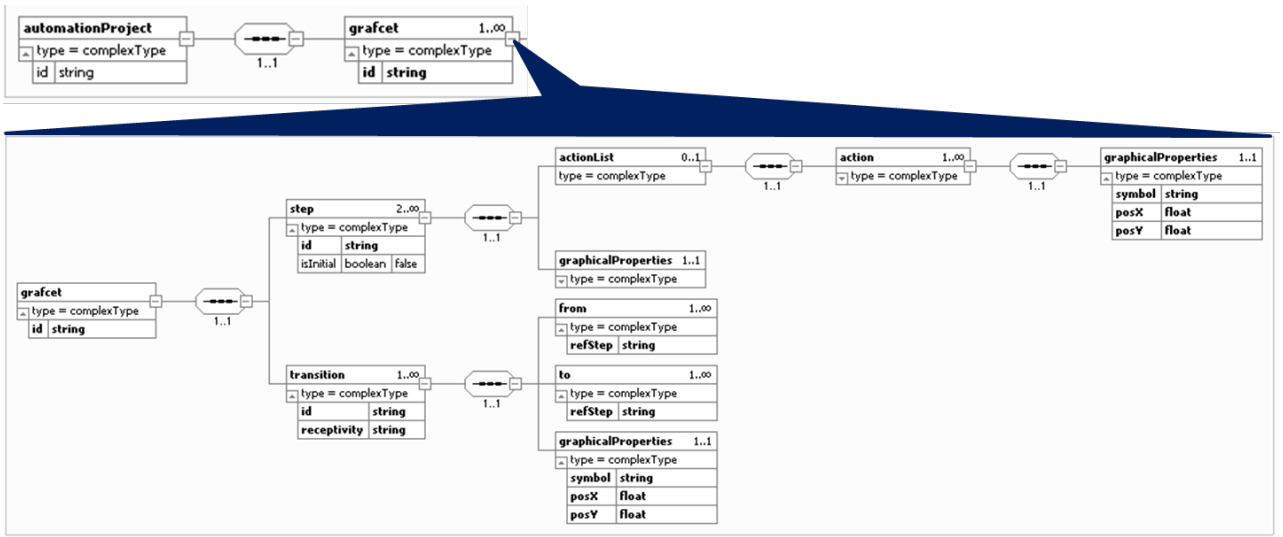

Figura 6 Meta-Modelo XML diseño proyecto automatización

3. Sincronización con la planta.

Este apartado se centra en conectar y sincronizar la parte operativa (Planta) con la que se pueda testear el diseño GRAFCET. Para ello, MODAS ofrece soporte para conectar y testear el GRAFCET a través de:

(1) Modelo digital de una planta. Trabajo previo de los autores (A. Sánchez 2022) ofrece una plataforma y soporte para ello. En este caso el protocolo de comunicación utilizado es MODBUS-TCP.

(2) Planta Real. Para ello, es necesario un módulo HW donde se encuentren cableadas las E/S. MODAS soporta dos posibilidades:

   a. Una placa desarrollada por el grupo de investigación GRAV basada en el microcontrolador 8051, dotada de 8 entradas 0-24 y 6 salidas a relé. Dicha placa está dotada de una interfaz RS485, a través de la cual se permite acceder a sus E/S por simulador por MODBUS-RTU. (Véase Figura 7).

b. PLC como pasarela. Es necesario habilitar la opción de control bajo Modbus. Por ejemplo, en caso de un PLC S7-1200 a través del módulo software MB_SERVER se realiza dicha comunicación teniendo acceso a todas las E/S del PLC. En este caso, el protocolo de comunicación es MODBUS-TCP.

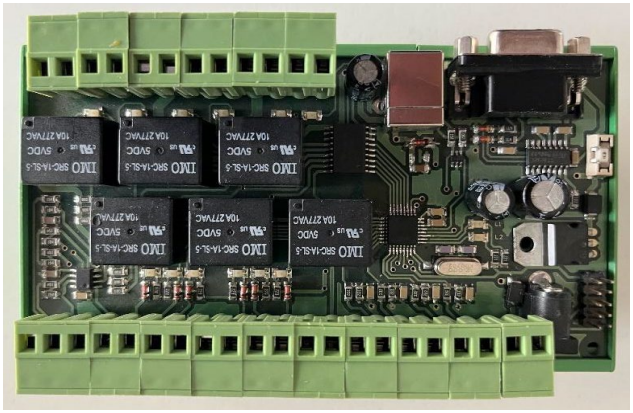

Figura 7 Tarjeta desarrollada por GRAV

4. Caso de estudio

En este apartado se hará uso de la herramienta para automatizar una maqueta multifuncional que se dispone en los laboratorios de la EPSJ de la Universidad de Jaén, muy utilizada en asignaturas de automatización de Grado y Máster (véase la Figura 8).

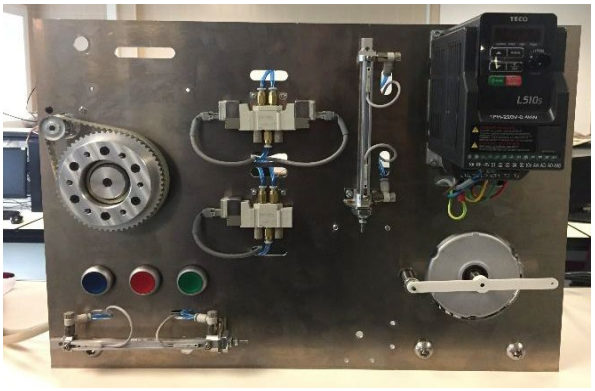

Figura 8 Maqueta Multifuncional

Dispone de 3 pulsadores (PR, PB, PG) y dos vástagos uno de doble efecto (A, colocado vertical) y otro de simple efecto (B). Dichos vástagos están dotados de dos finales de carrera y para manipularlos se dispone de una electroválvula biestable y monoestable respectivamente. La Figura 9 fichero AP_IOList con las E/S de dicha maqueta.

Pese a tratarse de una maqueta multifuncional, con infinidad de combinaciones en cuanto a funcionamiento de los vástagos, en este ejemplo la lógica funcional fijada es la siguiente. Cuando se pulse el pulsador azul (PB) se expande y contrae el vástago B una sola vez. Si se pulsa el pulsador rojo (PR) se emulará un prensado donde en primer lugar se fija la pieza con el vástago B y posteriormente se prensa con el vástago A. Cuando termina el prensado ambos vástagos han de volver a su posición de reposo.

AP_IOList

input (7)

	id	byte	bit	value
1	PR	0	0	false
2	PG	0	1	false
3	PB	0	2	false
4	ar	0	3	true
5	ae	0	4	false
6	br	0	5	true
7	be	0	6	false

output (3)

	id	byte	bit	value
1	Are	0	0	false
2	Aex	0	1	true
3	Bex	0	3	true

Figura 9 Entradas y Salidas del Automatismo

automationProyect

id grafet

grafcet

step (7)

	id	isInitial	graphicalProperties
1	0	true	graphicalProperties posX=0 posY=0
2	1	false	graphicalProperties posX=0 posY=181
3	2	false	graphicalProperties posX=0 posY=362
4	3	false	graphicalProperties posX=0 posY=543
5	4	false	graphicalProperties posX=0 posY=724
6	5	false	graphicalProperties posX=263 posY=198
7	6	false	graphicalProperties posX=263 posY=379

transition (8)

	id	receptivity	from	to	graphicalProperties
1	0	PR	from refStep=0	to refStep=1	graphicalProperties posX=0 posY=106
2	1	be	from refStep=1	to refStep=2	graphicalProperties posX=0 posY=287
3	2	ae	from refStep=2	to refStep=3	graphicalProperties posX=0 posY=468
4	3	ar	from refStep=3	to refStep=4	graphicalProperties posX=0 posY=649
5	4	br	from refStep=4	to refStep=0	graphicalProperties posX=0 posY=830
6	5	PB	from refStep=0	to refStep=5	graphicalProperties posX=263 posY=123
7	6	be	from refStep=5	to refStep=6	graphicalProperties posX=263 posY=304
8	7	br	from refStep=6	to refStep=0	graphicalProperties posX=263 posY=485

Figura 10: Modelo de Información del Automatismo

La Figura 10 ilustra el correspondiente modelo de información generado automáticamente por la herramienta. Es totalmente transparente al usuario pero fundamental a la hora de poder abrir modelos ya existentes en la herramienta o en un futuro generación de código automática.

Una vez definido el GRAFCET del automatismo, antes de la generación de código se ha de testear. Como se ha comentado anteriormente, MODAS ofrece hacerlo bien a través de una planta real o su correspondiente modelo digital. La Figura 11 ilustra el testeo del automatismo a través de la planta real haciendo uso de la placa HW previamente comentada.

5. Conclusiones

Este trabajo ha presentado la plataforma MODAS, plataforma de código abierto que da soporte a través de un modelo digital del automatismo secuencial, al diseño y testeo del proyecto de automatización con su correspondiente parte operativa real o virtual. Es importante remarcar que todo ello es totalmente independiente al PLC. Únicamente será necesario tener conocimiento de las I/Os de la parte operativa. Por tanto, MODAS permitirá diseñar arquitecturas software modularizadas y además ofrece pautas y soporte para diseño de aplicaciones en general.

En un trabajo futuro, se pretende añadir el servicio de generación automática de código de automatización en Texto Estructurado tanto en formato PLCopen XML que es acorde a IEC61131-3 como en STEP 7, abarcando así los estándares más extendidos en automatización.

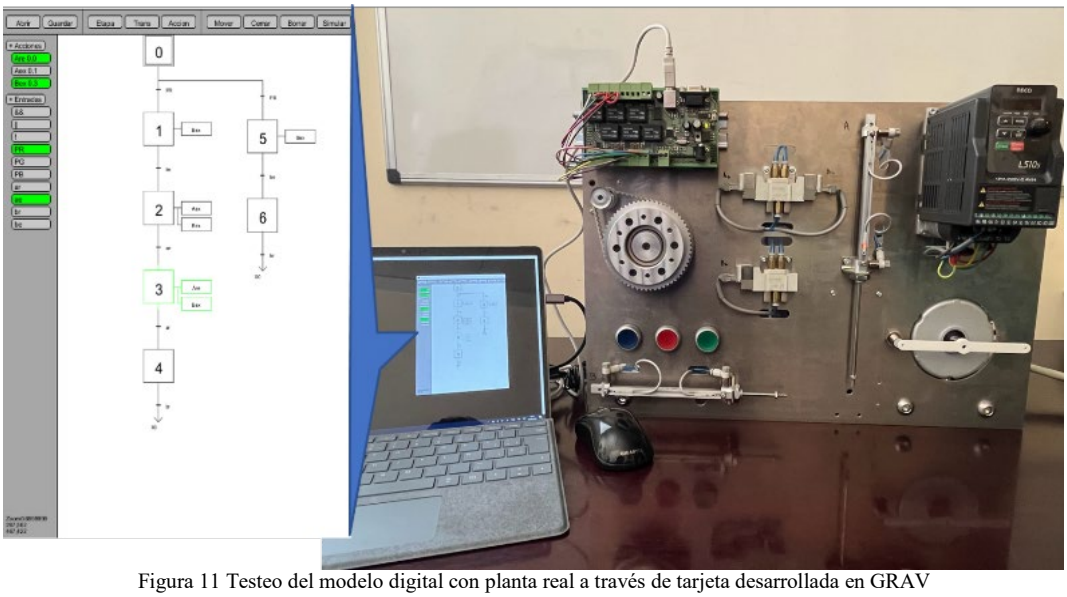

Figura 11 Testeo del modelo digital con planta real a través de tarjeta desarrollada en GRAV

Agradecimientos

Los autores quieren agradecer la subvención parcial de este trabajo a través de los proyectos PID2019-110291RB-I00 y FEDER Andalucía con el código FEDER A1123060E00010 y la referencia 1380776.

Referencias

A. D. Neal, R. G. Sharpe, K. van Lopik, J. Tribe, P. Goodall, T. W. Jackson, and A. A. West. "The potential of industry 4.0 Cyber Physical System to improve quality assurance: An automotive case study for wash monitoring of returnable transit items." *CIRP Journal of Manufacturing Science and Technology* 32 (2021): 467-475.

A. Sánchez, E. Estévez, I. Ruano, Juan Gómez Ortega and Javier Gámez García. "Plataforma Código abierto para Generar Gemelos Digitales." *XLIII Jornadas de Automática*, 2022: 941-948.

B. Vogel-Heuser, A. Fay, I. Schaefer, and M. Tichy. "Evolution of software in automated production systems: Challenges and research directions." *Journal of Systems and Software* 110 (2015): 54-84.

Cretu, Liviu Gabriel. "Model Driven Engineering of Information Systems: Principles Techniques and Practice." 2021.

J. de las Morenas, A. García-Higuera, and P. García-Ansola. "Shop Floor Control: A Physical Agents Approach for PLC-Controlled Systems." *IEEE Trans. Ind. Inf.* 13, no. 5 (2017): 2417-2427.

J. Fischer, S. Bougouffa, A. Schlie, I. Schaefer, and B. Vogel-Heuser. "A Qualitative Study of Variability Management of Control Software for Industrial Automation Systems." *IEEE ICSME*, 2018: 615-624.

Karl Heinz John, Michael Tiegelkamp. "IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids." 2010.

MODBUS. *Modicon Modbus Protocol Reference Guide.* n.d. https://www.modbus.org/docs/PI_MBUS_300.pdf (accessed 06 2023).

"MQTT." *The Standard for IoT Messaging.* n.d. https://mqtt.org/ (accessed 06 2023).

"OPC-UA." n.d. https://opcfoundation.org/about/opc-technologies/opc-ua/ (accessed 06 2023).

Processing. n.d. https://processing.org (accessed 06 2023).

R. S. Peres, A. D. Rocha, P. Leitão, and J. Barata. "IDARTS – Towards intelligent data analysis and real-time supervision for industry 4.0." *Computers in Industry* 101 (2018): 138-146.

S. Biffl, A. Lüder, and D. Gerhard, Eds. *Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects.* Cham: Springer,, 2017.

S. Karnouskos, P. Leitão, L. Ribeiro, and A. W. Colombo. "Industrial Agents as a Key Enabler for Realizing Industrial Cyber-Physical Systems." *IEEE Industrial Electronics Magazine* 14, no. 3 (2020): 18–32.

Salafia, S. Cavalieri and M. G. "Asset Administration Shell for PLC Representation Based on IEC 61131–3." *IEEE Access* 8 (2020): 142 606—142 621.

Zanero, S. "Cyber-Physical Systems." *Computer* 50, no. 4 (2017): 14–16.