



Laboratorio Sistemas de Eventos Discretos

Por Santiago Aljure Osorio

Introducción

En esta práctica se pretende apropiar los conceptos de control supervisorio de Sistemas de Eventos Discretos modelados por autómatas, y mostrar que dichos modelos pueden ser usados para generar el programa de un controlador real. Para ello se utiliza el software Factory I/O y una herramienta que permitirá interactuar con los elementos del software, que llamamos SDK, y cuyo proyecto se ejecuta utilizando el software Visual Studio.

Objetivo de la práctica

1. Apropiar conceptos de control supervisorio de sistemas de eventos discretos mediante la validación de modelos de autómatas generados con el software Suprémica en una planta virtual modelada en el software Factory I/O.
2. Integrar los conceptos teóricos del curso con una aplicación tecnológica concreta, pasando de un controlador supervisorio abstracto descrito por un autómata a un controlador encargado de una planta.

Actividades

1. Validación del funcionamiento de la herramienta con 2 máquinas y 1 buffer de 1 posición

Abra `machines2AndBuffer.factoryio` en Factory I/O (ver Imagen 1) y abra el proyecto de `Machines2AndBuffer` en Visual Studio. En Visual Studio abra el proyecto de 2 máquinas y 1 buffer. En Factory I/O configure los tiempos de falla de las máquinas 1 y 2 (ver Imagen 3), puede poner 1.10 en ambas máquinas. Finalmente haga clic sobre el botón de play en Visual Studio (ver Imagen 2).

Durante el funcionamiento, puede presionar el botón de forzar fallo (ver Imagen 4) para que la máquina falle. El operario se dará cuenta que la máquina se dañó cuando salga la siguiente pieza. Adicionalmente, si quiere detener la simulación, puede hacer clic sobre el botón de stop en Visual Studio (ver Imagen 5) y de Factory I/O (ver Imagen 6).

Advertencia: Si usted hace una selección de barrido en la pantalla de consola, bloquea el programa (ver Imagen 7). Para desbloquear, haga clic en la ventana y luego presione la flecha hacia abajo en su teclado.

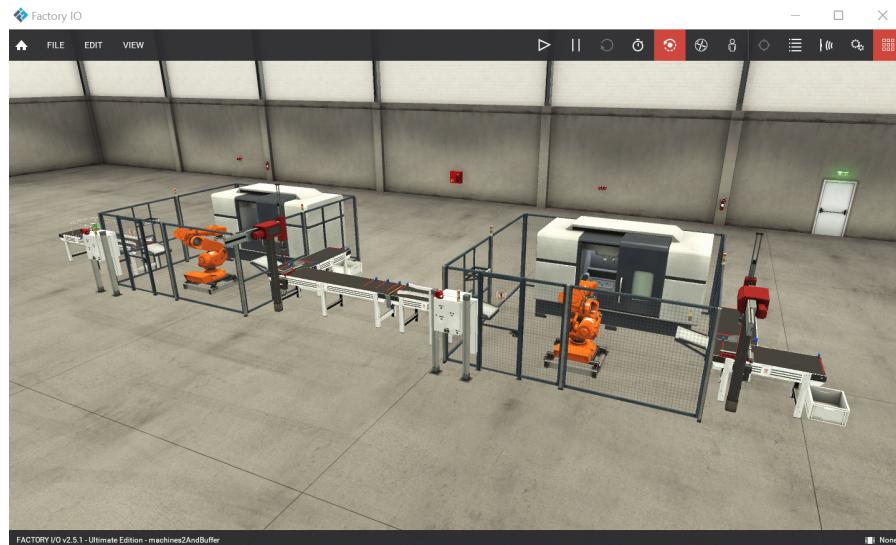


Imagen 1: machines2AndBuffer.factoryio en Factory I/O.

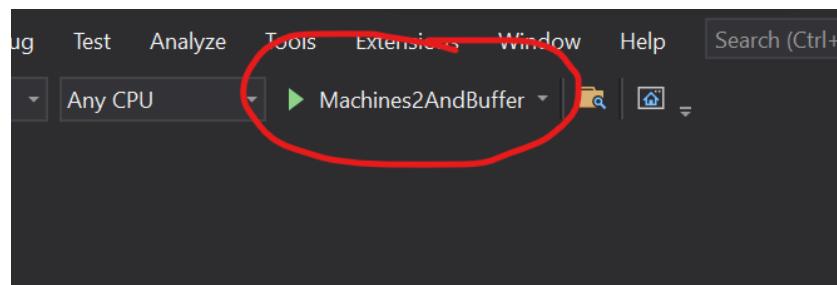


Imagen 2: Botón de play en Visual Studio para comenzar la simulación.



Imagen 3: Control del tiempo de falla de máquina 1. Funciona igual para las otras máquinas.



Imagen 4: Botón para forzar fallo en máquina.

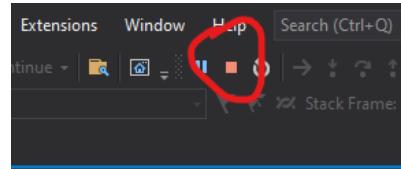


Imagen 5: Botón de stop en Visual Studio para detener la simulación.

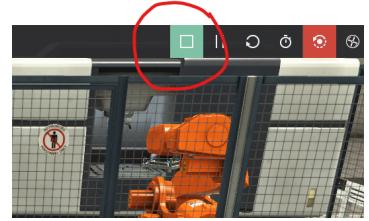


Imagen 6: Botón de stop en Factory I/O.

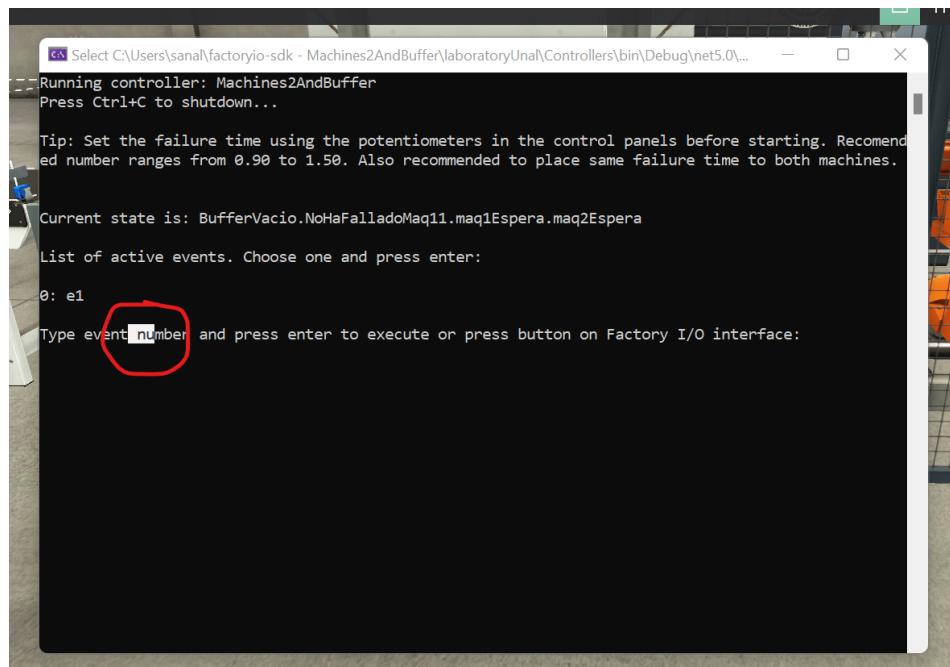


Imagen 7: Ventana de consola con selección de barrido, bloquea el programa.

2. Síntesis de supervisor de 2 máquinas y 1 buffer de 1 posición

Este punto utiliza los mismos archivos de Factory I/O (ver Imagen 1) y de Visual Studio que en el punto 1. Utilizando Suprémica, sintetice un controlador para 2 máquinas y 1 buffer de una posición usando los siguientes eventos.

Eventos:

- e1: empezar máquina 1 (controlable)
- e2: empezar máquina 2 (controlable)
- t1: terminó máquina 1 (no controlable)
- t2: terminó máquina 2 (no controlable)
- f1: falla máquina 1 (no controlable)
- f2: falla máquina 2 (no controlable)
- r1: reparar máquina 1 (controlable)
- r2: reparar máquina 2 (controlable)

Las máquinas 1 y 2 pueden estar en espera, trabajando o en falla.

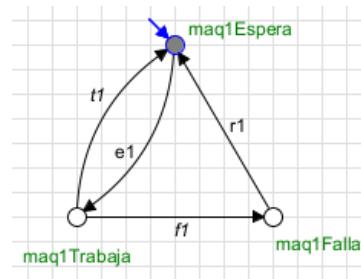


Imagen 8: Autómata máquina 1

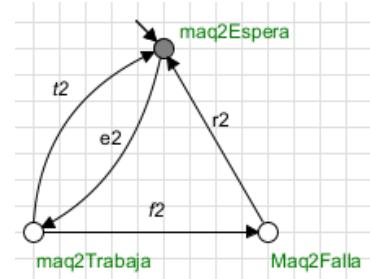


Imagen 9: Autómata máquina 2

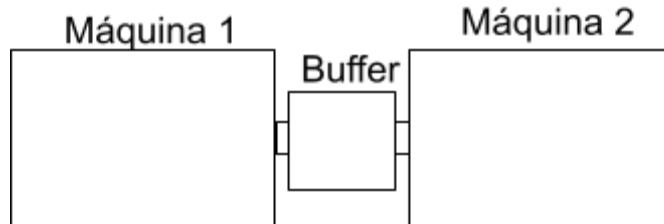


Imagen 10: Bosquejo de 2 máquinas y 1 buffer con espacio para 1 pieza

Especificaciones: Hay un buffer que puede estar lleno o vacío (es decir que tiene espacio para 1 pieza). Si las dos máquinas están dañadas, se debe reparar primero la máquina 1.

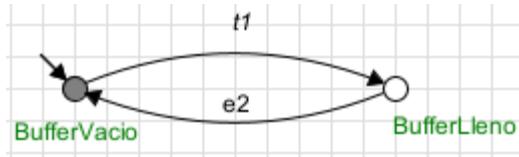


Imagen 11: Especificación buffer

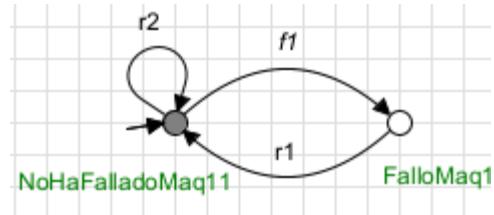


Imagen 12: Especificación falla máquina 2

Actividades específicas:

- Modele las plantas y las especificaciones en Suprémica
- Sintetice el supervisor en Suprémica
- Exporte y traduzca el supervisor a código c# (ver Anexo al final de esta guía)
- Experimento: Ejecute el SDK desde visual studio y verifique en consola la ejecución de la siguiente cadena de eventos: e1-t1-e2-t2

Preguntas:

- ¿Es posible tener la cadena de eventos e1-t1-e1-e2-t1-t2? Explique su respuesta.

3. Síntesis de controlador de 2 máquinas y 1 buffer con espacio para 3 piezas

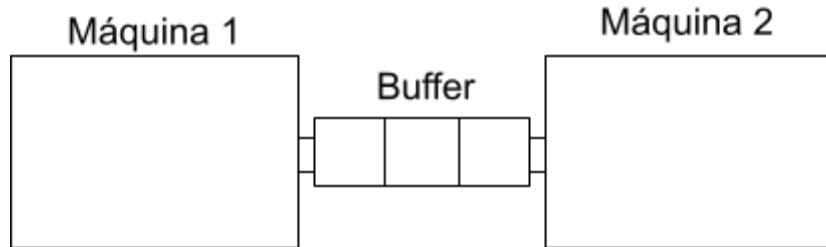


Imagen 13: Bosquejo de 2 máquinas y 1 buffer con espacio para 3 piezas

Éste utiliza el mismo archivo de Factory I/O machines2AndBuffer.factoryio (ver Imagen 1) pero debe abrir el proyecto de Machines2AndBuffer3 en Visual Studio.

Actividad de diseño: Ahora debe escribir la especificación y sintetizar el supervisor en Suprémica para que el buffer tenga espacio para 3 piezas. Adicionalmente la máquina 2 debe trabajar únicamente si en el buffer hay 2 o más piezas.

Actividades específicas:

- **Modifique el autómata del buffer**
- **Sintetice el nuevo supervisor en Suprémica**
- **Exporte y traduzca el nuevo supervisor a código c# desde Suprémica (ver Anexo al final de esta guía)**
- **Experimento 1: Ejecute la cadena que pone sólo una pieza en el *buffer* y verifique que la especificación no permite arrancar la máquina 2.**
- **Experimento 2: Ejecute la cadena de eventos que llena el *buffer* al máximo**

Preguntas:

- ¿Es posible tener la cadena de eventos e1-t1-e1-e2-t1-t2? Explique su respuesta.

4. Síntesis de controlador 3 Máquinas y 2 buffers de 1 posición

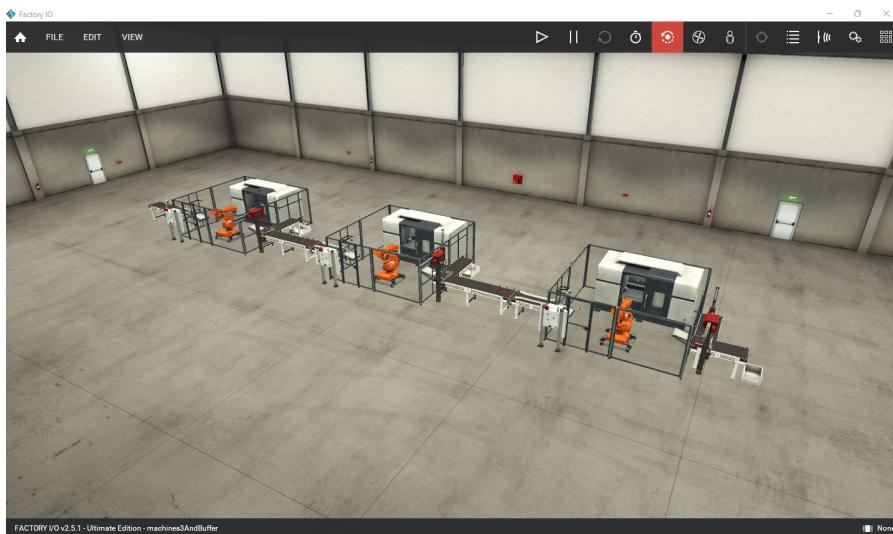


Imagen 14: *machines3AndBuffer.factoryio* en Factory I/O.

Abra *machines3AndBuffer.factoryio* en Factory I/O y abra el proyecto de *Machines3AndBuffer* en Visual Studio. Sintetice un controlador para 3 máquinas y 2 buffers usando los siguientes eventos.

Eventos:

- e1: empezar máquina 1
- e2: empezar máquina 2
- e3: empezar máquina 3
- t1: terminó máquina 1
- t2: terminó máquina 2
- t3: terminó máquina 3
- f1: falla máquina 1
- f2: falla máquina 2
- f3: falla máquina 3
- r1: reparar máquina 1
- r2: reparar máquina 2
- r3: reparar máquina 3

Los estados contemplan que las máquinas 1, 2 y 3 pueden estar en espera, trabajando o dañadas

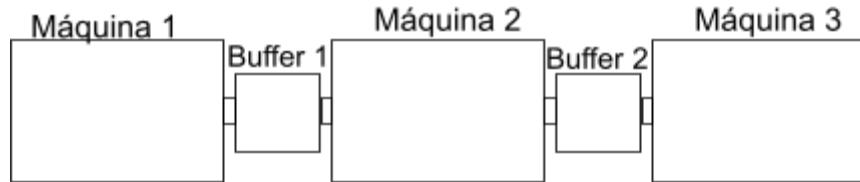


Imagen 15: Bosquejo de 3 máquinas y 2 buffers con espacio para 1 pieza

Especificaciones

Hay un buffer entre máquina 1 y máquina 2, y uno entre máquina 2 y máquina 3. Ambos buffers tienen espacio para almacenar 1 sola pieza.

Actividades específicas:

- Modele la nueva máquina como planta y la especificación del buffer faltante
- Sintetice el supervisor en Suprémica
- Exporte y traduzca el supervisor a código c# desde Suprémica (ver Anexo al final de esta guía)
- Experimento 1: ejecute la cadena de eventos que llena el buffer al máximo

Preguntas:

- ¿Por qué no modelar el buffer como planta?

5. Síntesis de controlador 3 Máquinas y 2 buffers de 3 posiciones

Éste utiliza el mismo `machines3AndBuffer.factoryio` (ver Imagen 3) pero debe abrir el proyecto de `Machines3AndBuffer3` en Visual Studio. Ahora debe modificar el controlador en Suprémica para que el buffer entre máquinas 1 y 2 tenga espacio para almacenar 2 piezas y el buffer entre máquina 2 y 3 tenga espacio para 3 piezas.

Actividades específicas:

- Modele el autómata del buffer como especificación
- Sintetice el supervisor en Suprémica
- Exporte y traduzca el supervisor a código c# desde Suprémica (ver Anexo al final de esta guía)

Preguntas:

- Ejecute una cadena de eventos que llene ambos buffers. Escriba la cadena que utilizó.

6. Síntesis de controlador de Sistema de manufactura

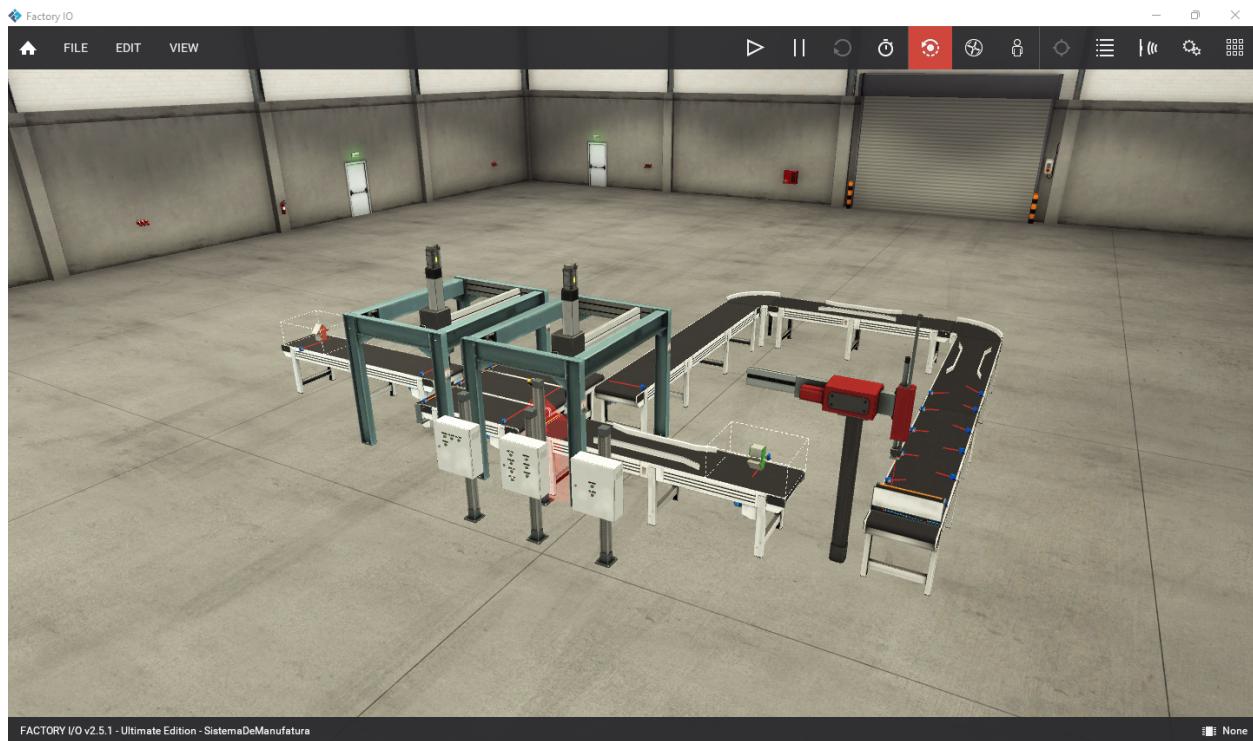


Imagen 16: SistemaDeManufactura.factoryio en Factory I/O.

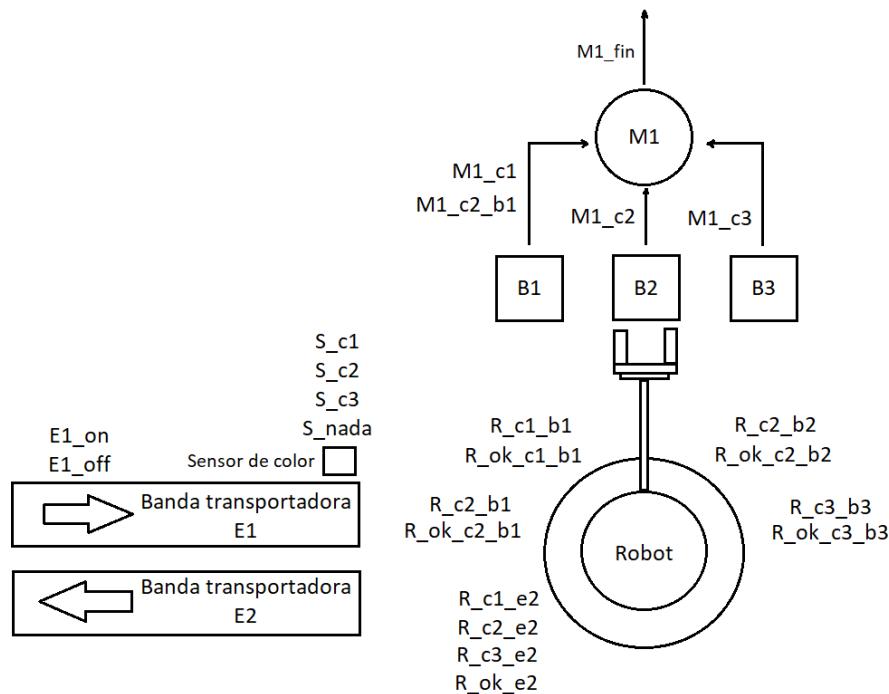


Imagen 17: Bosquejo de Sistema de manufactura con el listado de eventos en cada planta

Abra *SistemaDeManufactura.factoryio* en Factory I/O y abra el proyecto de *SistemaDeManufactura* en Visual Studio. En Suprémica abra el archivo *PracticaSistemaDeManufacturaLab.wmod* y sintetice un controlador usando los siguientes eventos (al abrir el archivo.wmod, debería ya tener todos estos eventos creados).

Eventos:

- E1_on: Encender banda transportadora E1
- E1_off: Apagar banda transportadora E1
- M1_fin: Terminó el ensamblaje
- M1_c3: Terminar ensamblaje con C3
- M1_c2: Empezar ensamblaje con C2 desde buffer B2
- M1_c1: Terminar ensamblaje con C1
- M1_c2_b1: Empezar ensamblaje con C2 desde buffer B1
- R_c1_b1: Robot lleve C1 a B1
- R_c1_e2: Robot lleve C1 a E2
- R_c2_b1: Robot lleve C2 a B1
- R_c2_e2: Robot lleve C2 a E2
- R_c3_b3: Robot lleve C3 a B3
- R_c3_e2: Robot lleve C3 a E2
- R_ok_c1_b1: C1 llegó a B1
- R_ok_c2_b1: C2 llegó a B1
- R_ok_c2_b2: C2 llegó a B2
- R_ok_c3_b3: C3 llegó a B3
- R_ok_e2: Pieza llegó a E2
- S_c1: Sensor de color vio C1
- S_c2: Sensor de color vio C2
- S_c3: Sensor de color vio C3
- S_nada: Sensor de color no ve pieza alguna

Los estados contemplan las siguientes especificaciones:

- a) La banda transportadora E1 puede estar encendida o apagada. La banda transportadora E2 no tiene eventos asociados.
- b) El sensor de color puede detectar pieza C1 (verde), C2 (metal), C3 (azul) o no ver pieza.
- c) El robot puede llevar una pieza C1 (verde) a B1 o a E2. El robot puede llevar una pieza C2 (metal) a B1, a B2 o a E2. El robot puede llevar una pieza C3 (azul) a B3 o a E2.
- d) M1 debe comenzar su ensamble con una pieza C2, y terminar con una pieza C1 o C3. C2 y C1 forman un producto A, mientras que C2 y C3 forman un producto B.
- e) Si la banda transportadora E1 está en movimiento, el robot no puede funcionar.

Actividades específicas

- **Sintetice el supervisor en Suprémica que satisface las especificaciones a) hasta e). Use el archivo de Suprémica llamado *PracticaSistemaDeManufacturaLab.wmod*.**
- **Exporte y traduzca el supervisor a código c# desde Suprémica (ver Anexo al final de esta guía)**

Actividades de diseño

1: Diseñe en Suprémica las especificación f) y sintetice un nuevo supervisor, incluyendo dicha especificación.

f) Si el buffer correspondiente al color está vacío, debe llenarse el buffer y no enviar la pieza a E2.

Solo si el buffer correspondiente al color está lleno, puede enviarse la pieza a E2.

Para esto, modifique las especificaciones de las Imágenes 19, 20 y 21.

2. Diseñe en Supremica un supervisor que implemente f) y además la especificación g) .

g) La diferencia entre productos A y B ensamblados no debe ser mayor a 1 (ver Imagen 18).

Para esto, use el autómata de la Imagen 22.

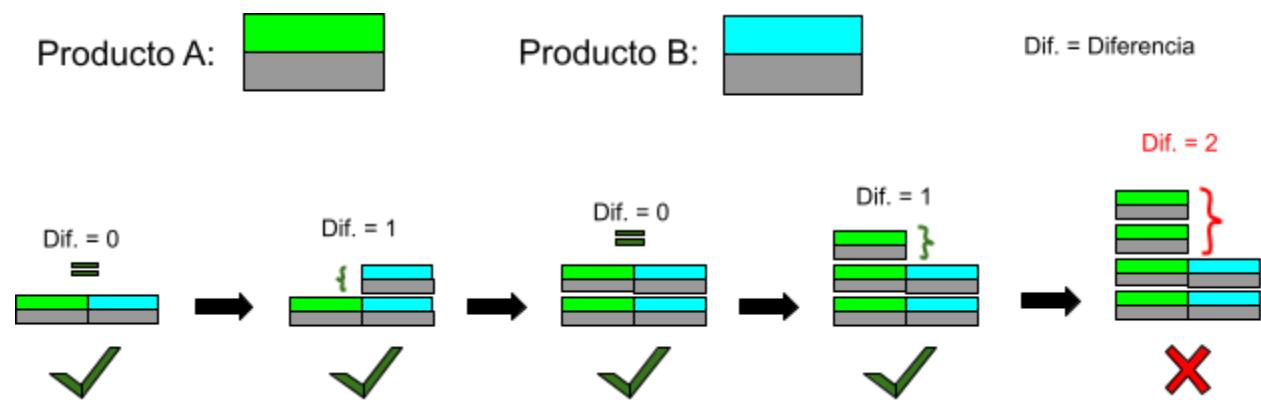


Imagen 18: Especificación g). La diferencia entre productos A y B no debe ser mayor a 1.

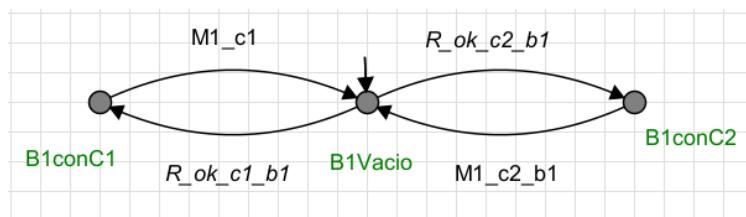


Imagen 19: Especificación Buffer 1

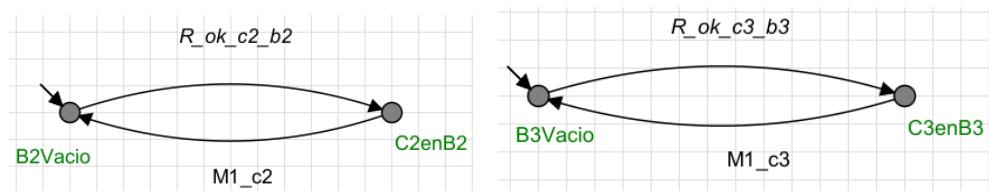


Imagen 20: Especificación Buffer 2

Imagen 21: Especificación Buffer 3

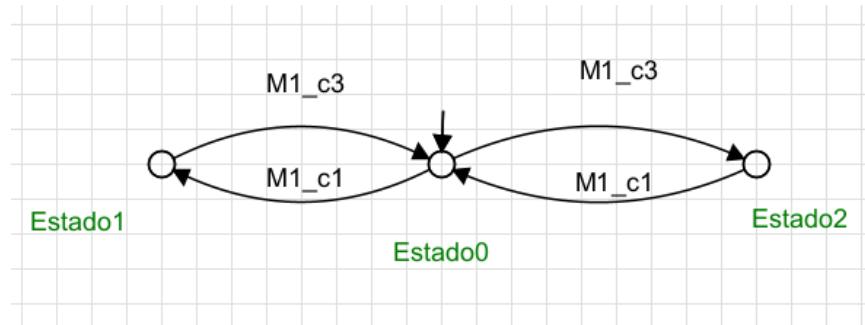


Imagen 22: Autómata para especificación g).

Preguntas:

- Explique, por favor, la modificación que hizo en los autómatas de la especificación f) para cumplir su objetivo.
- Explique, por favor, cómo funciona el autómata de la especificación g) para cumplir su objetivo.

GENERACIÓN DE SUPERVISORES PARA FACTORY I/O

Diseñe el controlador en Suprémica y expórtelo como XML.

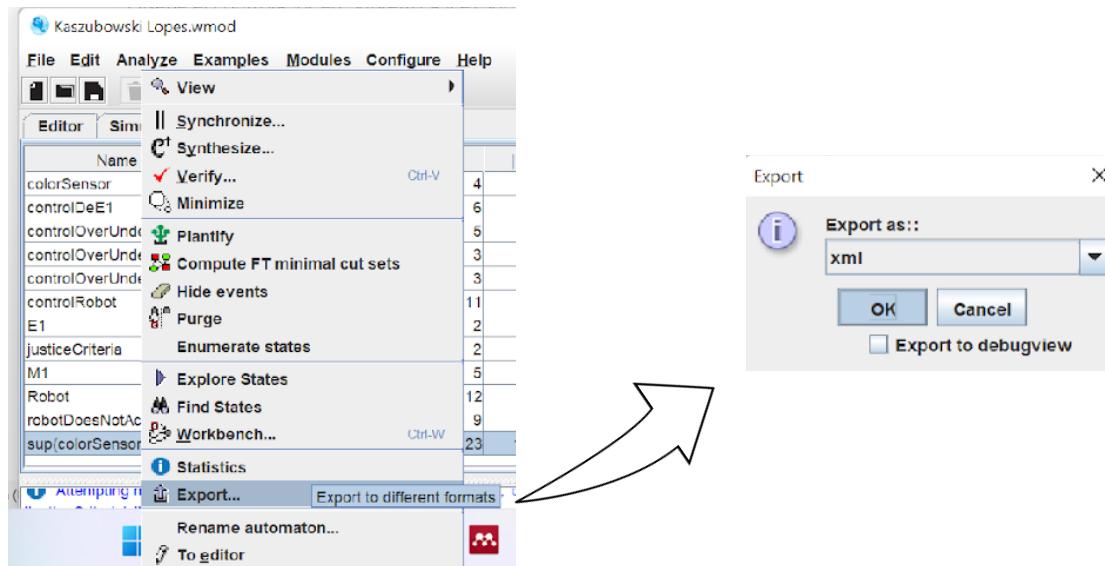


Imagen 23: Exportar controlador de Suprémica en formato .xml.

Lleve el archivo .xml a la carpeta de Supervisor.

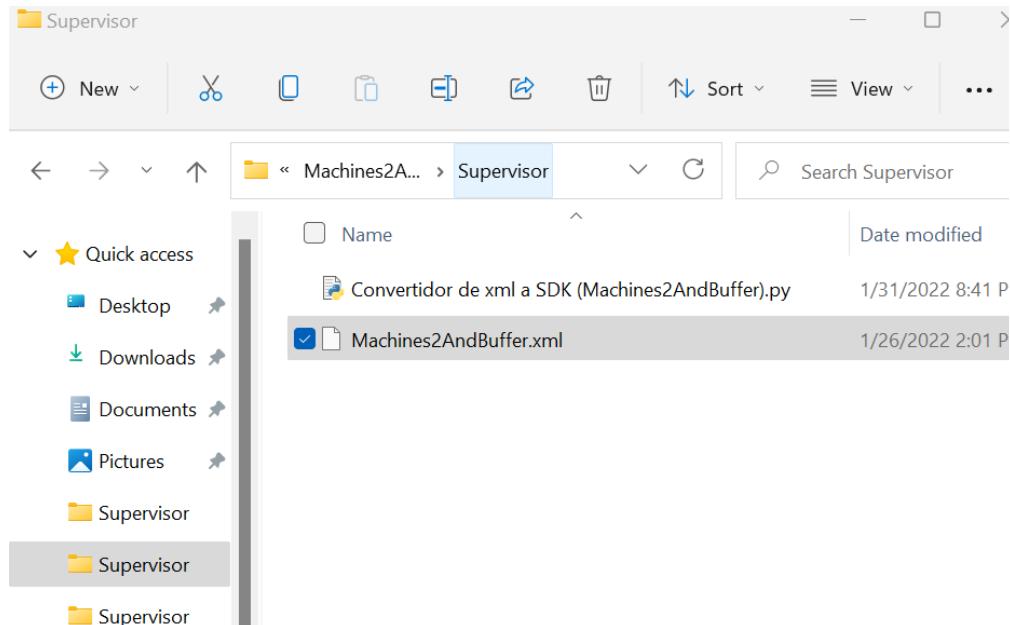


Imagen 24: .xml en carpeta de Supervisor

Ejecute el programa Convertidor de XML a SDK.py y haga clic sobre Open.

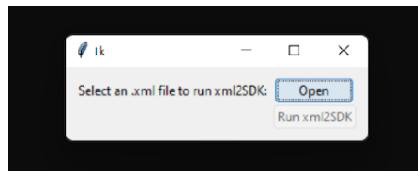


Imagen 25: Clic en Open en programa Convertidor de XML a SDK.py.

Navegue hasta el archivo .xml, selecciónelo y haga clic sobre Open.

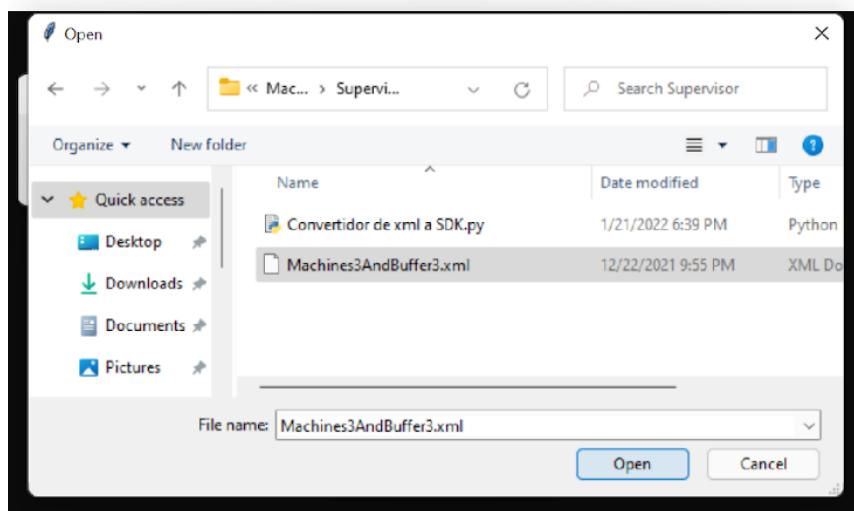


Imagen 26: Seleccionar y abrir .xml desde programa Convertidor de XML a SDK.py.

Ahora haga clic sobre Runxml2SDK. Si en consola lee un mensaje que diga *File named (nombre del xml)supervisor.cs created successfully* la conversión se realizó bien.

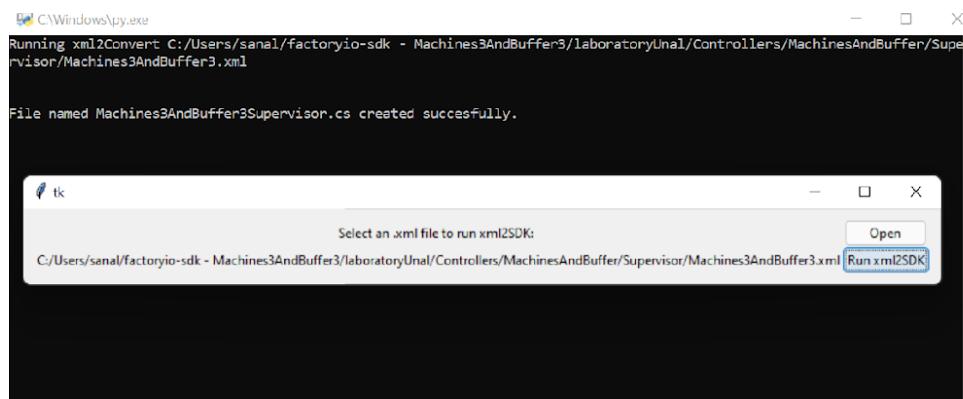


Imagen 27: Hacer clic en Run xml2SDK en el programa Convertidor de XML a SDK.py.

Ahora verá el controlador en formato .cs en la carpeta Supervisor.

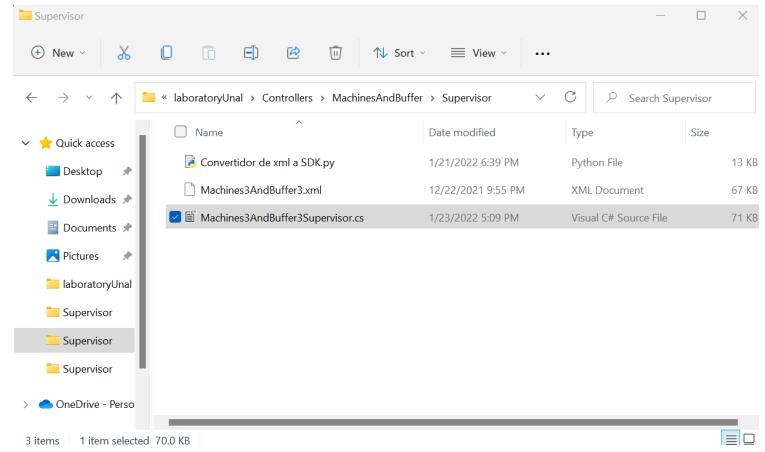


Imagen 28: Controlador en formato .cs creado satisfactoriamente.