UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

**TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES**

# DESARROLLO DE UNA PLATAFORMA SOFTWARE DE BAJO COSTE IMPLEMENTADA EN MATLAB PARA LA AUTOMATIZACIÓN DE PLANTAS INDUSTRIALES VIRTUALES


AUTOR: Daniel Marco Gil

TUTOR: José Vicente Salcedo Romero de Ávila


**Curso Académico: 2018-19**

# AGRADECIMIENTOS

En primer lugar, dar las gracias a mis padres por todo el esfuerzo que han hecho y por todo el apoyo que me han dado durante todo estos años.

También agradecer a mis amigos y compañeros por hacer que este largo camino haya sido más llevadero.

Y por último agradecer a mi tutor, José Vicente Salcedo, por la disposición a resolver cualquier duda y por la ayuda prestada durante todo el desarrollo del trabajo.

# RESUMEN

En este proyecto se pretende diseñar una plataforma de control de bajo coste en Matlab que permita reemplazar el uso de PLCs comerciales, para la automatización de plantas industriales virtuales. Detrás de este TFG subyace la idea reducir costes en una primera fase de diseño de un proyecto de automatización evitando la necesidad de comprar un PLC y la planta industrial o una maqueta de esta.

El automatismo en lugar de ser programado en el PLC se programaría mediante el entorno Stateflow de Matlab, el cual permite trabajar con máquinas de estado finitas, a las cuales se puede llegar a partir de automatismos realizados en lenguaje Grafcet.

Una vez el automatismo está implementado en Matlab el siguiente hito de este TFG es el de poder disponer de una plataforma de simulación en 3D de plantas industriales o, equivalentemente, una plataforma de plantas industriales virtuales. En este proyecto se va a emplear el software comercial Factory I/O, el cual permite diseñar gran variedad de plantas virtuales y ser renderizadas en 3D en tiempo real. Por otro lado, este software permite establecer conexión, por ejemplo, vía OPC con un controlador lógico (PLC), el cual le transmite las órdenes asociadas al automatismo desarrollado.

La comunicación entre Stateflow de Matlab y Factory I/O se va a realizar mediante la herramienta OPC Toolbox de Matlab y el servidor OPC KERServerEX, usando una estructura de cliente/servidor OPC.

Como resultado de este proyecto, con un único PC en el que se instalen Matlab, KEPServerEX y Factory I/O se podrá realizar la implementación de un automatismo, y verificar su correcto funcionamiento sobre una reproducción virtual de la planta industrial. Esto permite evitar la inversión asociada a la compra de un PLC y de una maqueta industrial en una primera fase del proyecto de automatización. En caso de que los resultados de esta primera fase fuesen positivos se podría acometer la adquisición tanto del PLC como de la maqueta.

**Palabras Clave:** Stateflow, OPC, bajo coste, automatismo, planta virtual.

# RESUM

En aquest projecte es pretén dissenyar una plataforma de control de baix cost en Matlab que permeta reemplaçar l'ús de PLCs comercials, per a l'automatització de plantes industrials virtuals. Darrere d'aquest TFG subjau la idea reduir costos en una primera fase de disseny d'un projecte d'automatització evitant la necessitat de comprar un PLC i la planta industrial o una maqueta d'aquesta.

L'automatisme en lloc de ser programat en el PLC es programaria mitjançant l'entorn Stateflow de Matlab, el qual permet treballar amb màquines d'estat finites, a les quals es pot arribar a partir d'automatismes realitzats en llenguatge Grafcet.

Una vegada l'automatisme està implementat en Matlab la següent fita d'aquest TFG és el de poder disposar d'una plataforma de simulació en 3D de plantes industrials o, equivalentment, una plataforma de plantes industrials virtuals. En aquest projecte s'emprarà el programari comercial Factory I/O, el qual permet dissenyar gran varietat de plantes virtuals i ser renderizadas en 3D en temps real. D'altra banda, aquest programari permet establir connexió, per exemple, via OPC amb un controlador lògic (PLC), el qual li transmet les ordres associades a l'automatisme desenvolupat.

La comunicació entre Stateflow de Matlab i Factory I/O es realitzarà mitjançant l'eina OPC Toolbox de Matlab i el servidor OPC KERServerEX, usant una estructura de client/servidor OPC.

Com a resultat d'aquest projecte, amb un únic PC en el qual s'instal·len Matlab, KEPServerEX i Factory I/O es podrà realitzar la implementació d'un automatisme, i verificar el seu correcte funcionament sobre una reproducció virtual de la planta industrial. Això permet evitar la inversió associada a la compra d'un PLC i d'una maqueta industrial en una primera fase del projecte d'automatització. En cas que els resultats d'aquesta primera fase foren positius es podria escometre l'adquisició tant del PLC com de la maqueta.

**Paraules clau:** Stateflow, OPC, baix cost, automatisme, planta virtual.

# ABSTRACT

This project aims to design a low-cost control platform in Matlab to replace the use of commercial PLCs for the automation of virtual industrial plants. Behind this TFG underlies the idea of reducing costs in a first phase of design of an automation project avoiding the need to buy a PLC and the industrial plant or a model of it.

The automation instead of being programmed in the PLC would be programmed through Matlab's Stateflow environment, which allows working with finite state machines, which can be reached from automatisms made in Grafcet language.

Once the automation is implemented in Matlab, the next milestone of this TFG is to be able to have a 3D simulation platform for industrial plants or, equivalently, a virtual industrial plants platform. In this project the commercial software Factory I/O will be used, which allows to design a great variety of virtual plants and to be rendered in 3D in real time. On the other hand, this software makes it possible to establish a connection, for example, via OPC with a logic controller (PLC), which transmits the commands associated with the developed automation.

The communication between Matlab Stateflow and Factory I/O is going to be done through Matlab's OPC Toolbox and the KERServerEX OPC server, using an OPC client/server structure.

As a result of this project, with a single PC in which Matlab, KEPServerEX and Factory I/O are installed, it will be possible to implement an automatism and verify its correct operation on a virtual reproduction of the industrial plant. This makes it possible to avoid the investment associated with the purchase of a PLC and an industrial model in the first phase of the automation project. If the results of this first phase were positive, the acquisition of both the PLC and the model could be undertaken.

**Keywords:** Stateflow, OPC, low cost, automation, virtual plant.

# ÍNDICE

**DOCUMENTOS CONTENIDOS EN EL TFG**

- Memoria
- Pliego de condiciones
- Presupuesto
- Anexos

## Índice Memoria

# Índice Pliego de condiciones

# Índice Presupuesto



# Índice Anexos

# Índice Figuras

# Índice Tablas

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

**TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES**

# DESARROLLO DE UNA PLATAFORMA SOFTWARE DE BAJO COSTE IMPLEMENTADA EN MATLAB PARA LA AUTOMATIZACIÓN DE PLANTAS INDUSTRIALES VIRTUALES

# MEMORIA

**Curso Académico: 2018-19**

# CAPÍTULO 1. INTRODUCCIÓN

## 1.1. OBJETIVO DEL DOCUMENTO

El principal objetivo que persigue el presente proyecto es el diseño de una plataforma software de bajo coste que permita realizar la implementación de automatismos sin la necesidad de adquirir un PLC y una maqueta del proceso.

Con este proyecto se busca además de la creación de la plataforma, remplazar el uso de PLCs comerciales en la automatización de plantas industriales virtuales. Con ello se reducirían los costes en una primera fase de diseño de un proyecto de automatización.

Además, se creará una interfaz de usuario a través de la cual se podrá seguir y controlar el proceso con los distintos modos de funcionamiento.

## 1.2. ESTRUCTURA DEL DOCUMENTO

Este documento se ha estructurado siguiendo el mismo recorrido que se ha seguido en el desarrollo del proyecto.

En primer lugar, se inició con la recogida de información de los programas informáticos que se iban a utilizar y el aprendizaje de estos programas para posteriormente realizar la plataforma.

Una vez adquirido los conocimientos sobre la utilización de estos programas informáticos, se pasó a la creación de la plataforma a través de la cual se implementarán los automatismos para controlar los procesos industriales que se deseen.

Con la plataforma ya creada, se seleccionó un proceso industrial virtual de Factory I/O y se realizó la implementación del automatismo en la plataforma para comprobar el correcto funcionamiento de esta.

Por último, una vez realizada la implementación del automatismo, se realizó una interfaz de usuario en App Designer a través de la cual permite controlar y seguir el proceso que se está llevando a cabo.

### 1.3. MOTIVACIÓN

Este proyecto se motiva en la creación de una herramienta que permita a las empresas reducir los costes en una primera fase de diseño de un proyecto de automatización.

Con esta motivación se intenta crear una plataforma software que permita al usuario realizar esta primera fase del proyecto sin la necesidad de adquirir un PLC y una maqueta industrial.

Una vez vistos los resultados de esta fase, en caso de que sean positivos se puede pasar a la compra tanto del PLC como de la maqueta.

# CAPÍTULO 2. HERRAMIENTAS UTILIZADAS

En este capítulo, se hablará de los distintos programas utilizados durante el desarrollo del trabajo, estos programas informáticos son: Matlab, Kepserver y Factory I/O.

## 2.1. MATLAB.

Matlab es un software matemático que utiliza su propio lenguaje de programación (lenguaje M) y permite al usuario representar funciones y datos, manipular matrices, desarrollar algoritmos, crear interfaces de usuarios (GUI) y comunicarse con otros programas escritos en un lenguaje distinto como C++, Java o Python.

Aunque la función principal de Matlab es la computación numérica, este incluye dos herramientas adicionales como Simulink (plataforma de simulación multidominio) y App Designer (editor de interfaces de usuario). También se puede aumentar las capacidades de Matlab con las toolboxes (cajas de herramientas), y las de Simulink con los blocksets (paquetes de bloques).

Para el desarrollo de este trabajo se han utilizado las herramientas adicionales de las que dispone Matlab como Simulink para la implementación del automatismo a través de la herramienta Stateflow y App Designer para el desarrollo de la interfaz gráfica.

También utilizaremos la herramienta OPC Toolbox de Matlab para la comunicación entre los distintos programas utilizados.

### 2.1.1. Simulink.

Simulink permite analizar, modelar y simular sistemas dinámicos multidominios a través de un entorno de programación gráfica en el cual el modelo se construye a través de los diferentes bloques de sus bibliotecas. Este puede ser ejecutado desde Matlab o ser él quien maneje Matlab, además puede simular modelos en tiempo continuo y discreto. Su utilización es muy común para el control automático y el procesamiento de señales digitales.

Para acceder a Simulink se dispone de tres métodos:

Escribir en el Command Windows de Matlab >>Simulink.

Seleccionando Simulink de la pestaña HOME de Matlab (ver figura 1).

En la pestaña HOME, seleccionar **New** → **Simulink Model** (ver figura 1).

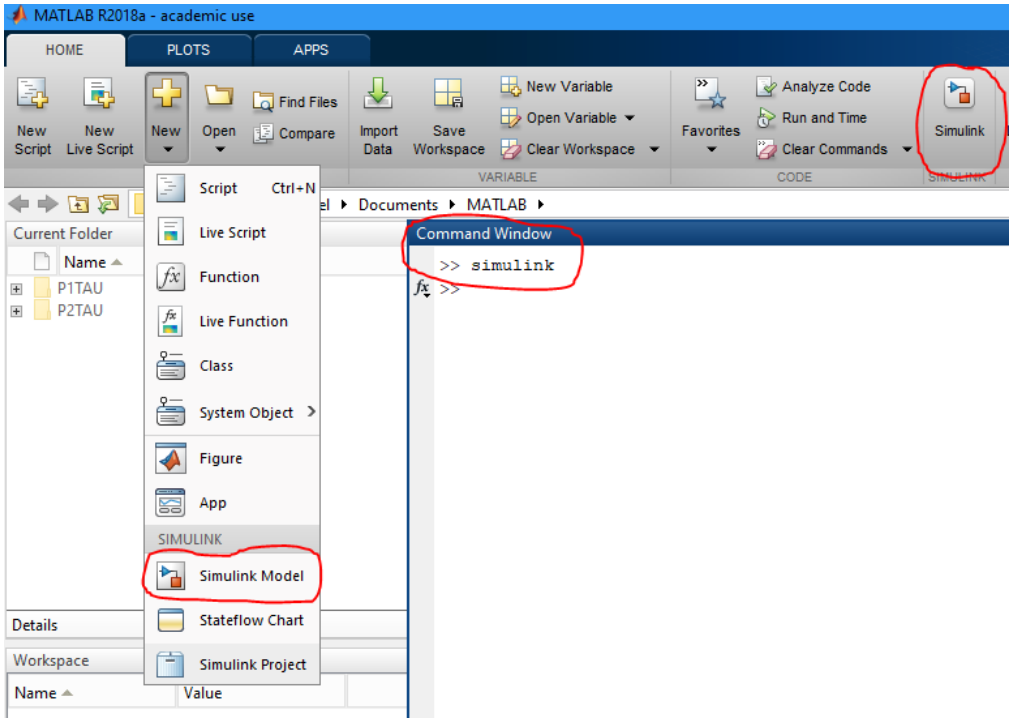



*Figura 1. Formas de acceso a Simulink.*

Antes de acceder a Simulink por uno de estos tres métodos, se abre la pantalla de la figura 2**,** en la que se puede abrir un nuevo modelo seleccionando **Blank Model** o abrir un modelo ya creado pulsando en la carpeta **Open** y seleccionando el modelo.

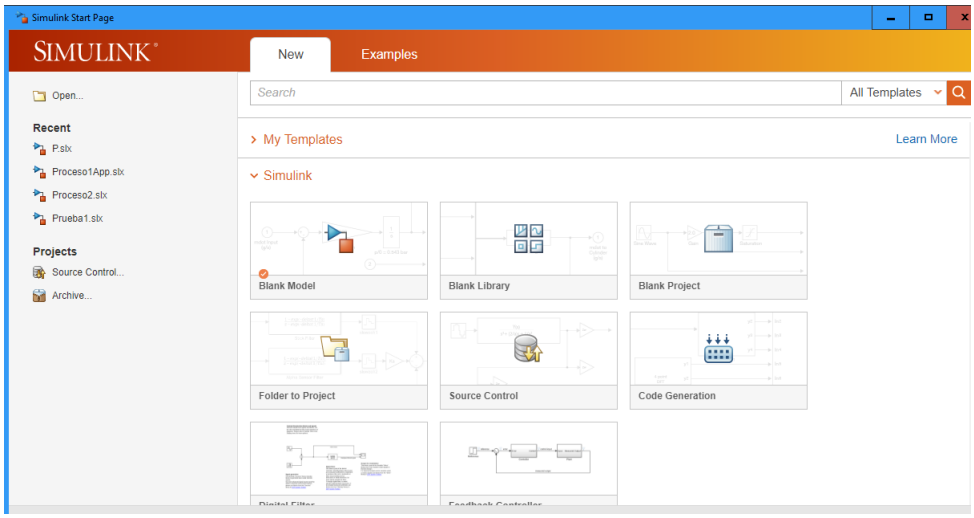



*Figura 2. Pantalla acceso a Simulink.*

Una vez accedido a Simulink se abrirá el entorno de trabajo mostrado en la figura 3, dentro de este entorno para crear el modelo es necesario abrir la biblioteca pulsando en el icono, donde  se encuentran todos los bloques que dispone Simulink.

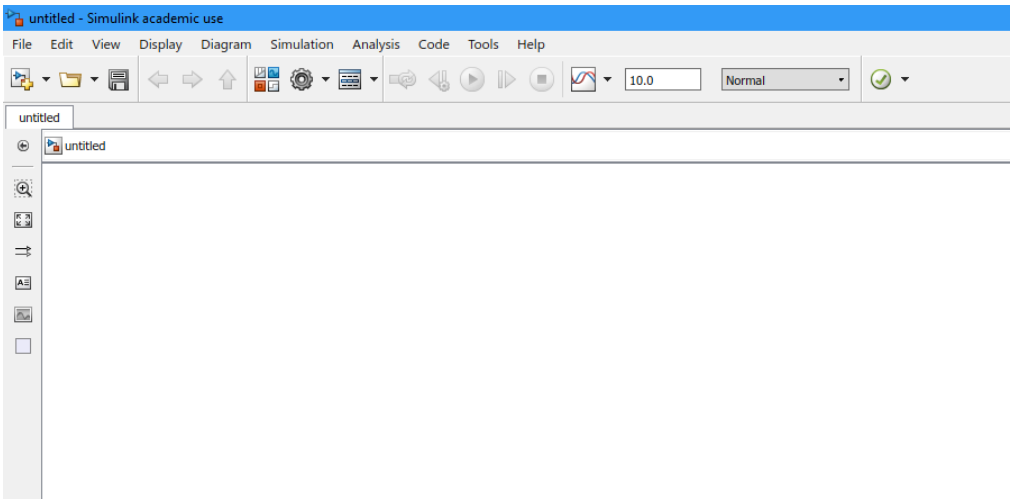



*Figura 3. Entorno de trabajo de Simulink.*

En la figura 4 se puede observar los distintos bloques de los que se disponen para crear un modelo, también es posible acceder a ellos clicando con el botón izquierdo sobre el entorno de trabajo y escribiendo el nombre del bloque que se desee.

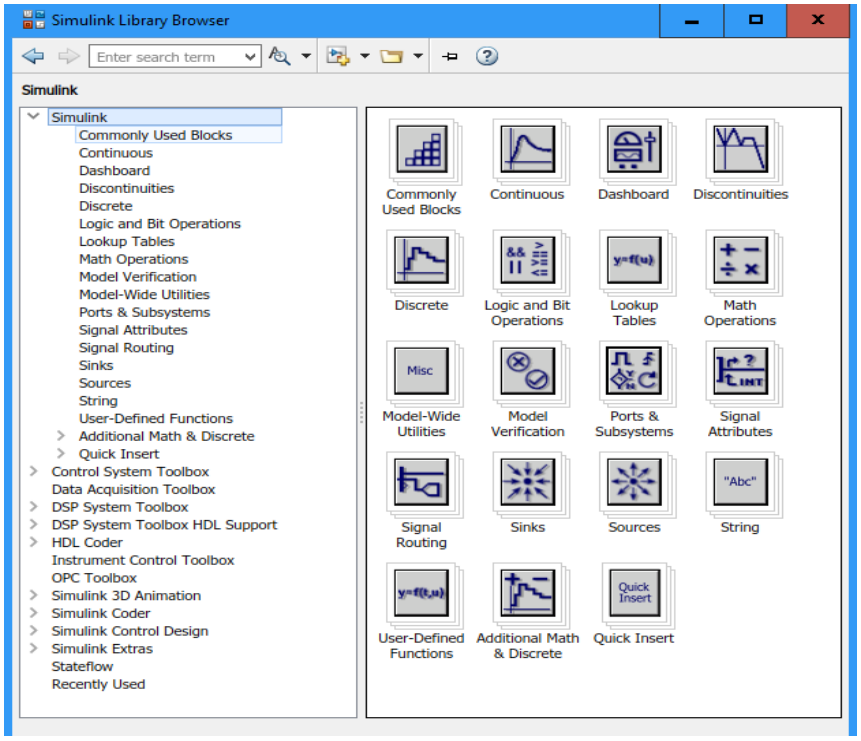



*Figura 4. Librería de Simulink.*

Dentro de estas bibliotecas se puede encontrar el bloque Stateflow, que será la base para realizar nuestro trabajo.

### 2.1.2. Stateflow.

Es una herramienta de diseño gráfico utilizada para modelar y simular sistemas a través de diagramas de flujo y máquinas de estado dentro de un modelo creado en Simulink. Los sistemas que se modelan en Sateflow se utilizan sobre todo como lógica de control.

Las máquinas de estado que utiliza Stateflow son una variante de las clásicas máquinas de estado. Esta variante fue descrita por Harel y se denomina statechart.

Estas máquinas de estado añaden las siguientes características fundamentales:

- Jerarquía del diagrama: Se pueden crear máquinas de estado tradicionales dentro de un estado de la de nivel superior, es decir, la máquina que se encuentra en el nivel inferior solo desarrollará su comportamiento cuando la máquina de nivel superior tenga activo el estado correspondiente.
- Estados paralelos: Esta permite que varios estados se ejecuten a la vez, que junto a la jerarquía del diagrama permite que dos máquinas de estado de nivel inferior estén activas al mismo tiempo.
- La historia dentro de un gráfico de estado.

Para abrir el entorno de trabajo de Stateflow es necesario abrir el bloque Chart de la biblioteca de Simulink. Este bloque también se puede conectar con otros bloques de Simulink, ya sean de tiempo continuo o discreto, para formar modelos de sistemas híbridos.

Una vez abierto el entorno de trabajo (ver figura 5), en la parte izquierda se observan los distintos elementos de Stateflow.

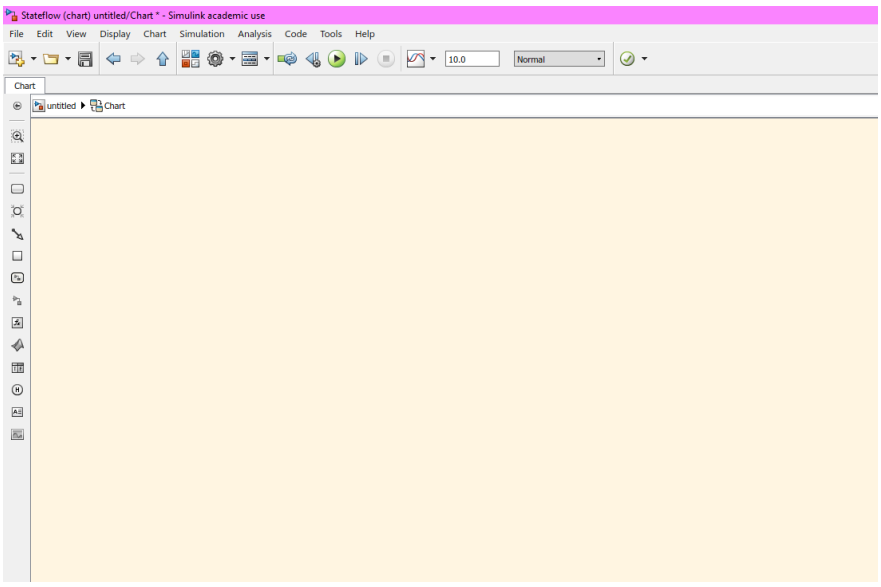



*Figura 5. Entorno de trabajo de Stateflow.*

Estos elemento son:

- **Estados (States).**

  Los estados pueden representar estados propiamente dichos o también pueden englobar una máquina de estados completa, la cual solo se ejecuta cuando el estado que la contiene se encuentre activo. De forma gráfica se representan con forma rectangular con los bordes redondeados como se aprecia en la figura 6.

  Por tanto, dentro de un estado pueden existir otros y estos pueden agruparse entre ellos de dos formas posibles:

  1. **Alternativamente (OR)**: Se ejecuta cada vez un estado, es decir, de forma exclusiva y se indica de forma gráfica con línea continua.
  2. **En paralelo (AND)**: Se ejecutan todos los estados a la vez y se indica en el gráfico con línea discontinua.

  En la siguiente figura se pueden observar los dos tipos de estados gráficamente.

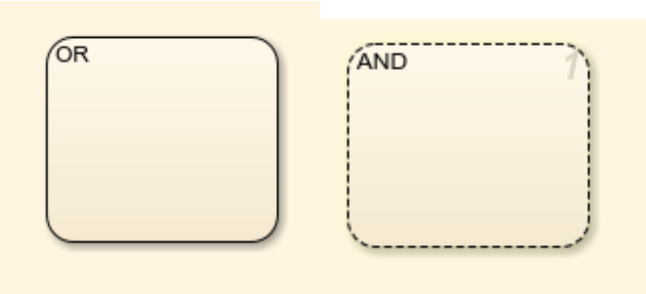



*Figura 6. Tipos de estados de Stateflow.*

Los estados pueden contener acciones que se llevaran a cabo en distintos instantes de su activación. Estos instantes son:

1. **Entry** (entrada): La acción se ejecuta al entrar en este estado.
2. **During** (durante): La acción se ejecuta al entrar en este estado y continúa ejecutándose mientras el estado siga activo.
3. **Exit** (salida): La acción se inicia cuando se desactiva el estado.
4. **On** (cuando): La acción se inicia cuando estando activo el estado, se produce cierto evento.

En la siguiente figura se muestra cómo sería la sintaxis dentro del estado de los diferentes instantes.

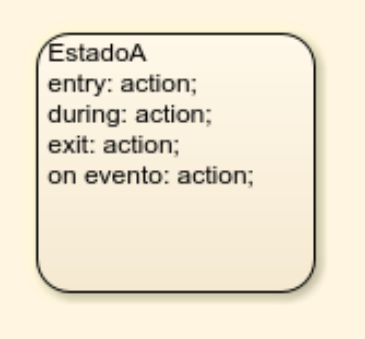



*Figura 7. Sintaxis de los estados.*

- **Transiciones (Transitions).**

  Las transiciones se representan con un segmento con punta de flecha. Esta flecha indica cual será el siguiente estado que se activará una vez lo esté el estado de origen, es decir, cuando un estado se encuentre activo y del que se origina una transición a otro, dejará de estarlo si se cumplen las condiciones que contiene la transición y pasará a activarse el segundo.

  Por lo general las transiciones llevan alguna condición asociada a cumplirse para que se produzca dicha transición, pero si una transición no tiene ninguna condición asociada se produce cada vez que se evalúa el diagrama (al ocurrir cualquier evento).

  Para representar las transiciones se utiliza la siguiente nomenclatura:

  - **Condition (condición)**: Indica la condición que debe cumplirse para activar la transición una vez se haya producido el evento correspondiente y se ejecute la condition_action.
  - **Event (evento)**: Expresa que evento provocará que la condición asociada a la transición sea evaluada. De forma explícita, en ausencia de evento será válido cualquier evento del diagrama.
  - **Condition_action (acción condicionada)**: Cuando la condición anterior es cierta esta se ejecuta.
  - **Transition_action (acción de transición)**: La acción se ejecuta antes de que el estado siguiente este excitado.

  Dentro de las transiciones se encuentra una transición especial llamada transición por defecto (default transition). Esta transición es aquella que apunta al estado inicial del sistema, es decir cada vez que una máquina de estados se active el estado señalado por la transición por defecto se activará.

Por tanto, la transición por defecto no tiene origen, ya que no procede de ningún estado anterior. Esta tiene sentido aplicarla para los estados agrupados de forma alternativa, y no cuando están de forma paralela, ya que en este caso no existe ambigüedad.

En la figura 8 se puede ver la utilidad que tiene la transición por defecto que provoca que al activarse el estado A siempre se activará el estado 1 y hasta que no se cumpla la condición 1 no pasará al estado 2.

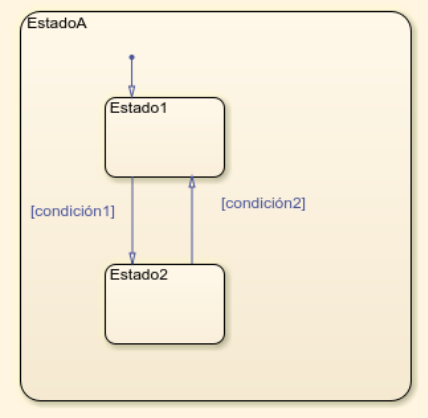



*Figura 8. Ejemplo de transiciones de Stateflow.*

- **Memoria de estado (History Junction).**

La memoria de estado provoca que cuando se reactiva una máquina de estados en lugar de activarse el estado señalado por la transición por defecto se active el último estado activo.
Se representa con el símbolo de un círculo con una H inscrita dentro de él.
En el siguiente ejemplo (figura 9) se observa como para el estado A la primera vez que esté activo se activará el estado 1, pero las siguientes veces se activará el último estado que estuvo activo. Aunque hay una transición por defecto esta sólo actuará para la primera ejecución de la máquina de estados.

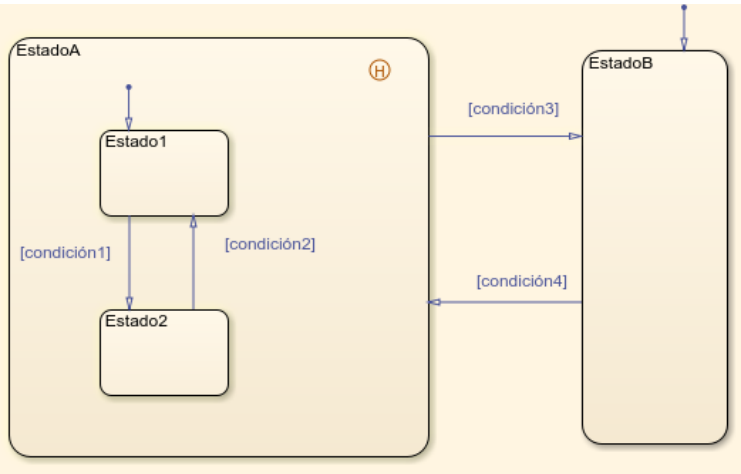



*Figura 9. Ejemplo memoria de estado.*

- **Uniones (Junctions).**

  Las uniones representan posibles caminos de transición desde una única transición y se símbolo es un pequeño círculo. Estas permiten representar las siguientes situaciones:

  - Estructuras 'if-then-else': Esta estructura consiste en una vez se haya producido un evento X llegamos a la unión en la cual, si se produce la condición Y pasaremos al estado B, si se produce la condición Z pasaremos al estado C y si no se ha producido ninguna de estas condiciones pasaremos al estado D. En la siguiente figura se observa de forma gráfica.

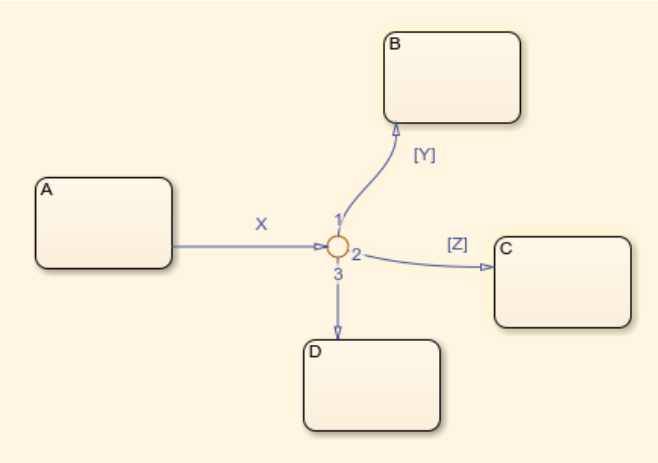



*Figura 10. Ejemplo de estructura if-the-else.*

  - Bucle 'for': En la siguiente figura se pued ver de forma gráfica como a ver un bucle de 5 iteraciones en la unión para que se produzca la transición entre dos estados.

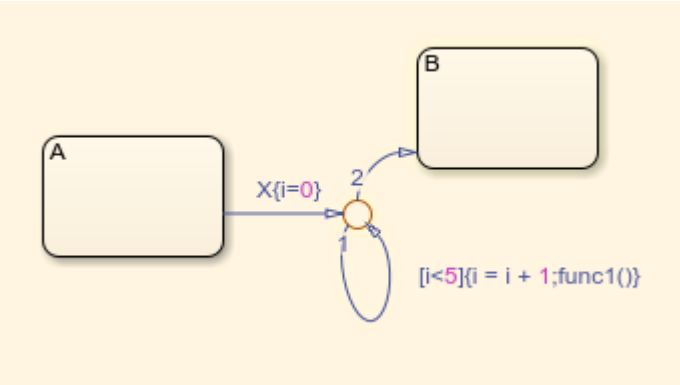



*Figura 11. Ejemplo de bucle for.*

  - Transiciones tanto de un único origen a múltiples destinos, como de múltiples orígenes a un único destino.

- **Eventos (Events).**

  Los eventos pueden ser locales o entradas o salidas de Simulink que se comunican con Stateflow a través de un puerto de entrada o salida en el bloque Chart, de forma que los eventos entran o salen, formando un vector de eventos, por el mismo puerto.
  Cada evento tiene asociado un índice que define su posición en el vector. La forma de producir los eventos en Simulink para que entren a través del bloque Chart es realizando cambios bruscos de alguna señal. Cundo los eventos se declaran en Stateflow, se puede elegir entre:

  - Flanco de subida
  - Flanco de bajada
  - Flanco indiferente
  - Llamada de función

  Cada vez que se produce un evento provoca que los estados y las transiciones del diagrama se evalúen.

- **Datos (Data).**

  Los datos pueden ser definidos como:

  - Dato de entrada
  - Dato de salida
  - Dato local
  - Constante
  - Parámetro
  - Memoria del almacén de datos

  Los datos de entrada y salida se comunican con Simulink a través de puertos en el bloque Chart.

  Para ver y definir los distintos datos y eventos en el espacio de trabajo es necesario activar la vista de símbolos (‘symbols’), donde se pueden definir datos y eventos ya creados o crear nuevos.

También se puede realizar desde la herramienta 'Model Explorer' (ver figura 12), en la parte central de la ventana se encontrarían todos los datos y eventos que han sido creados y en la parte derecha donde se definen según sus características estos datos y eventos.

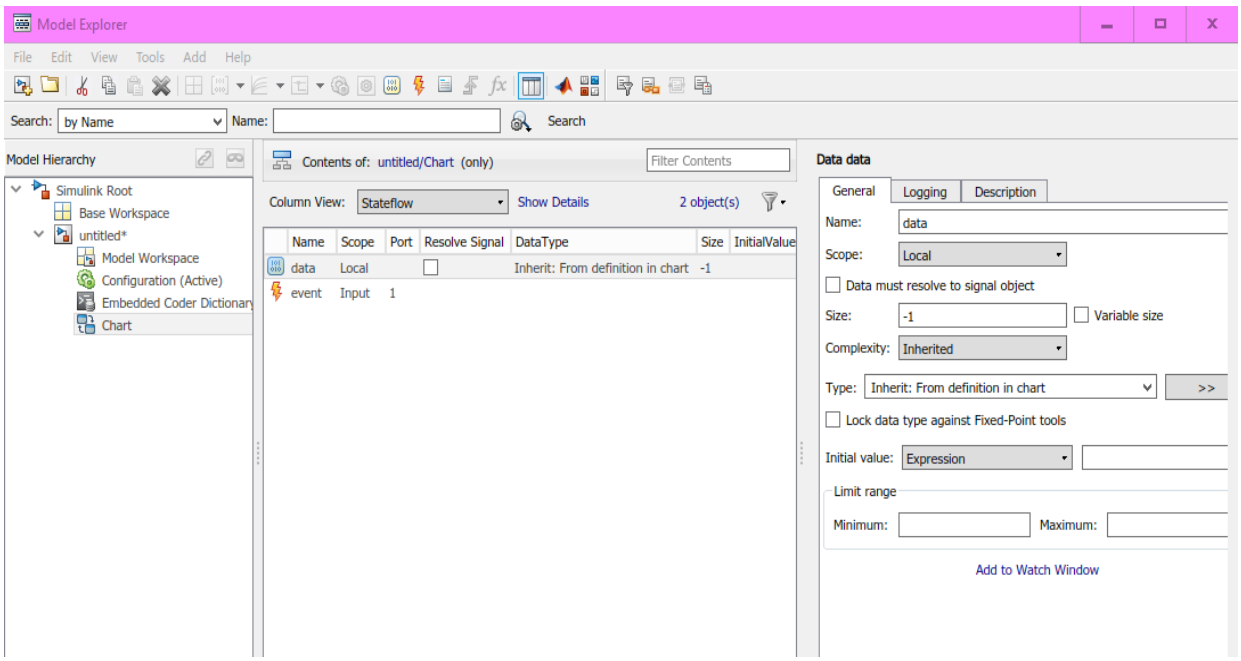



*Figura 12. Herramienta Model Explorer.*

- **Acciones (Actions).**

  Las acciones se pueden realizar en las transiciones, así como en los estados en las distintas opciones que estos contemplan, ya descritas anteriormente.

  Entre las operaciones que están permitidas a la hora de ejecutar las acciones están las operaciones lógicas, incrementos y decrementos, desplazamientos, punteros, etc.

  Incluso es posible declarar funciones en C en ficheros aparte de códigos fuente para ser llamadas desde las acciones del diagrama.

  También es posible ejecutar acciones basadas en condiciones con implicaciones temporales con los eventos. Estas opciones temporales son: before, after, at y every.

Para el desarrollo de este apartado la información ha sido obtenida de las referencias bibliográficas [1], [2], [3] y [4].

### 2.1.3. App Designer.

Es un entorno gráfico de desarrollo de interfaces que busca reducir el tiempo necesario para desarrollar interfaces y facilitar el diseño de estas.

En este entorno, las vistas de código y diseño están muy vinculadas entre sí, de manera que si se realizan cambios en una afectan inmediatamente a la otra.

Para acceder a App Designer es posible de dos maneras:

1. Escribir appdesigner en el Command Windows de Matlab.
2. Desde la pestaña HOME de Matlab, seleccionar **New→App** (al igual que se hacía para abrir Simulink, ver figura 1).

Una vez accedido a App Designer se abre el entorno de trabajo, dentro de este entorno es posible distinguir dos pestañas: Design View (figura 13) y Code View (figura 14).

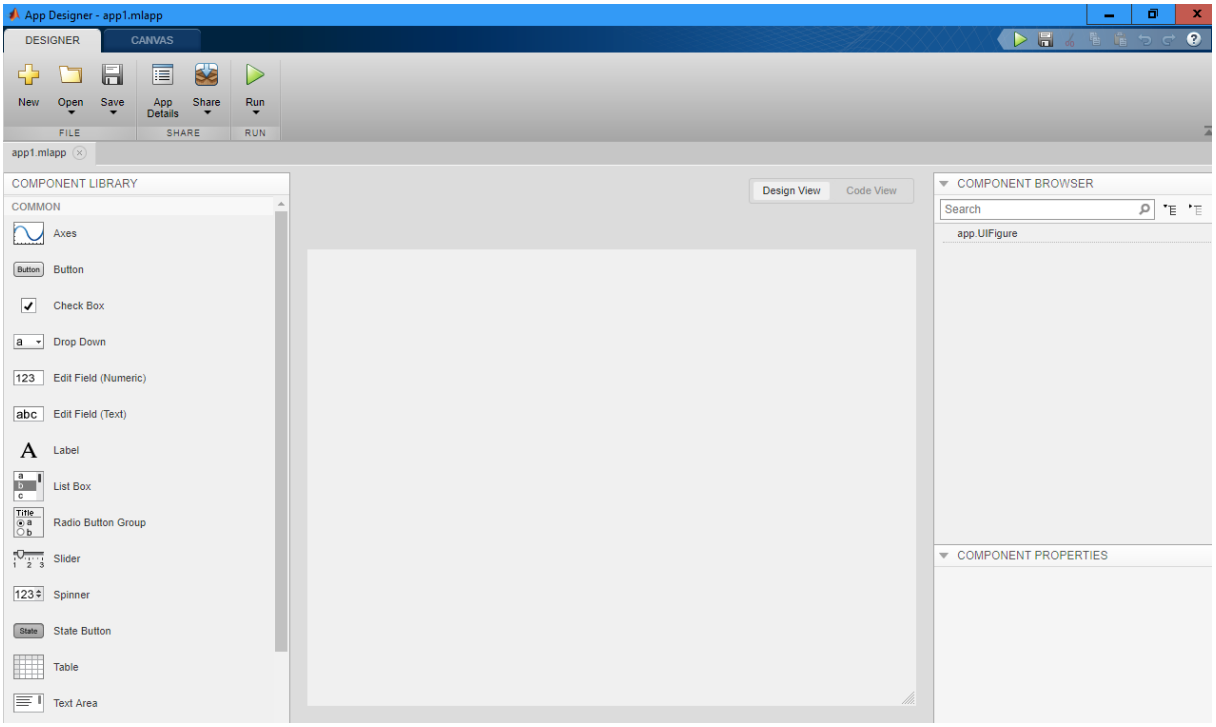



*Figura 13. Pestaña Design View de App Designer.*

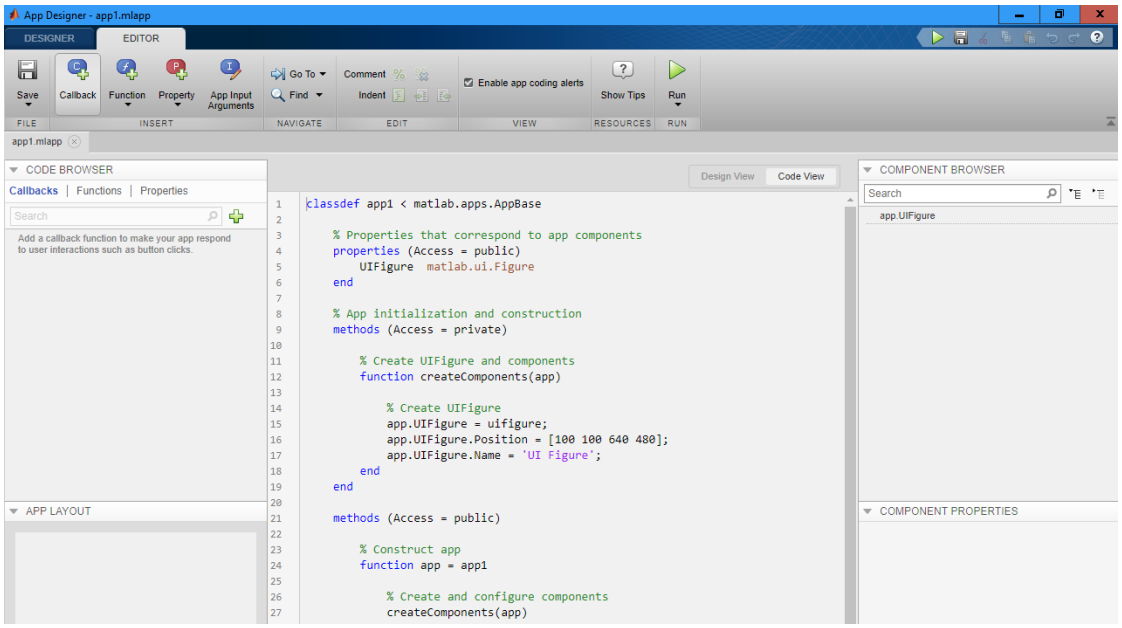



*Figura 14. Pestaña Code View de App Designer.*

En función de que pestaña se tenga activada es posible encontrar unos paneles u otros:

**A. Si está activada la pestaña Design View.**

Dentro de este entorno se encuentran los siguientes paneles.

**1. Design Editor.** Es el área sobre la que los elementos de control son distribuidos y supone una representación de lo que se mostrará por pantalla una vez de ejecute la interfaz.

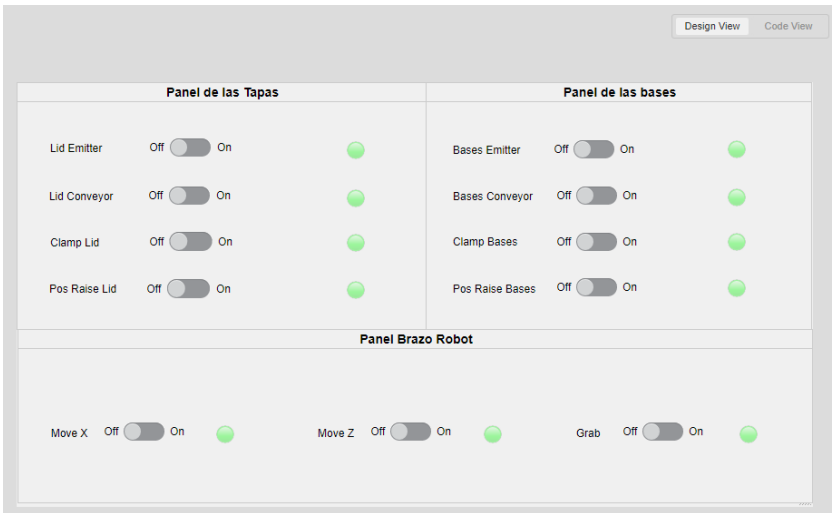



*Figura 15. Design Editor.*

**2. Component Library.** En este panel se encuentran los diferente componentes de los que se dispone para el diseño de la interfaz. Estos están clasificados en common, containers, figure tools e instrumentation.

**3. Component Properties.** En esta ventana se pueden observar las propiedades de los componentes utilizados en el Design Editor. Muestra las características que se pueden modificar del componente seleccionado.

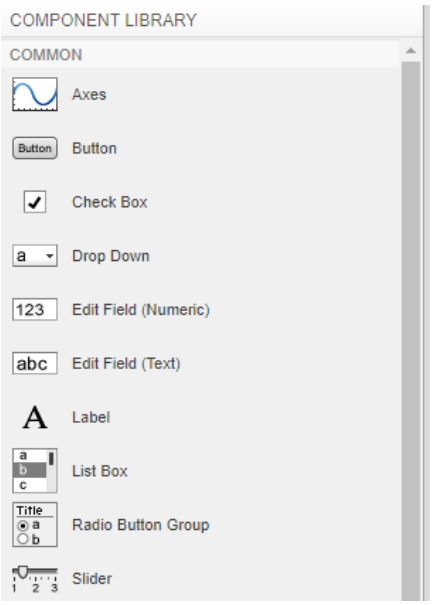



*Figura 16. Component library.*

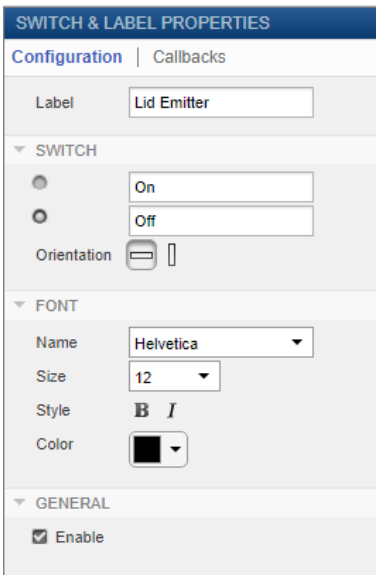



*Figura 17. Component Properties.*

**4. Canvas.** Dentro de esta pestaña se encuentran diferentes herramientas que se pueden utilizar en la Design View.

| Pestaña Canvas | |
|---|---|
|  | **Save.** Guarda la aplicación con la que se está trabajando. |
|  | **Align.** Esta herramienta alinea los componentes que se hayan seleccionado. |
|  | **Same Size**. Iguala al mismo tamaño los componentes seleccionados. |
|  | **Grouping.** Agrupa aquellos componentes que se han seleccionado. |

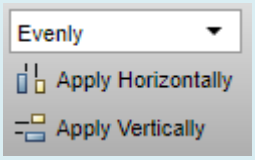
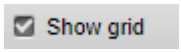
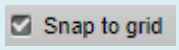
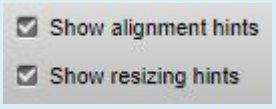
| | |
|---|---|
|  | **Space.** Distribuye los componentes seleccionados. La distribución es posible de realizar de modo uniforme (evently) o introduciendo la distancia que se desee. |
|  | **Show grid.** Activa la cuadrícula. |
|  | **Snap to grid.** La posición de los componentes se ajusta a las líneas de la cuadrícula. |
|  | **Interval.** Cambia las dimensiones de la cuadrícula. |
|  | **Show alignment hints y Show resizing hints.** Ambas muestran líneas de ayuda, la primera para alinear los componentes y la segunda para ajustar las dimensiones de los componentes. |
|  | **Run.** Ejecuta la aplicación. |

*Tabla 1. Herramientas de la pestaña Canvas.*

**B. Si está activada la pestaña Code View.**

Para esta pestaña se observan los distintos paneles que se describen a continuación.

**1. Code Editor.** En este panel es donde se encuentra todo el código de la aplicación, tanto el código generado automáticamente, que no es posible editar y tiene un fondo de color gris, como el código que generan las funciones, callbacks y propiedades. Al realizar cualquier cambio en el diseño, así como si se agregan nuevas funciones, callbacks o propiedades, estos cambios se verán reflejados de forma inmediata en el código.

La estructura que tiene el código que se va generando tiene la siguiente forma:

- En la primera parte del código están definidas las propiedades de la aplicación.

Esta sección contiene las características que corresponden a los componentes de la aplicación que se están utilizando.

- En la segunda parte se encuentran las funciones.

Es en esta sección donde se añaden las callbacks o funciones creadas en la aplicación.

- En la tercera parte se inicializa y crea la aplicación y sus componentes.

Es en esta sección del código donde aparecen las modificaciones si se han utilizado las propiedades de las UIfigure o el inspector para cambiar las características de los componentes.

```
classdef app1 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure     matlab.ui.Figure
        Button       matlab.ui.control.Button
        LampLabel    matlab.ui.control.Label
        Lamp         matlab.ui.control.Lamp
        SwitchLabel  matlab.ui.control.Label
        Switch       matlab.ui.control.Switch
        KnobLabel    matlab.ui.control.Label
        Knob         matlab.ui.control.DiscreteKnob
    end

    methods (Access = private)

        % Button pushed function: Button
        function ButtonPushed(app, event)

        end
    end

    % App initialization and construction
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'UI Figure';

```

*Figura 18. Ventana Code Editor.*

**2. Code Browser.** Se pueden ver las distintas funciones de utilidad, callbacks y propiedades creadas en la aplicación.

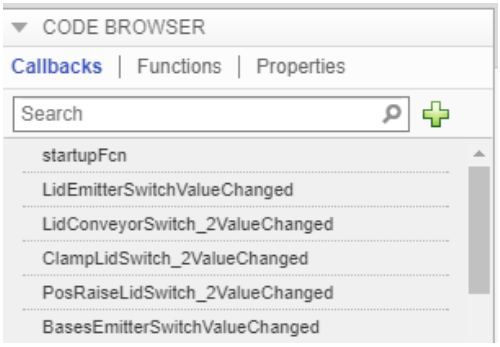



*Figura 19. Ventana Code Browser.*

**3. App Layout.** Proporciona una vista de la distribución de los elementos que conforman la aplicación.

**4. Component Properties.** Esta venta en similar a la que aparece en la Design View, pero contiene propiedades adicionales de los elementos seleccionados.

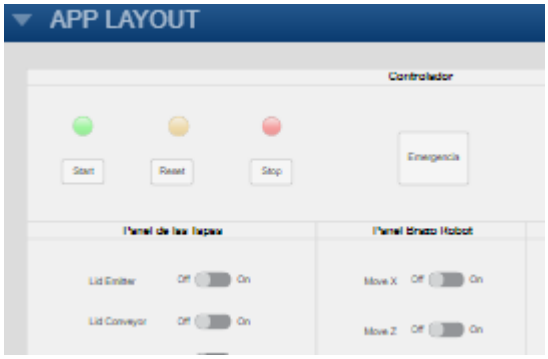



*Figura 20. Ventana App Layout.*

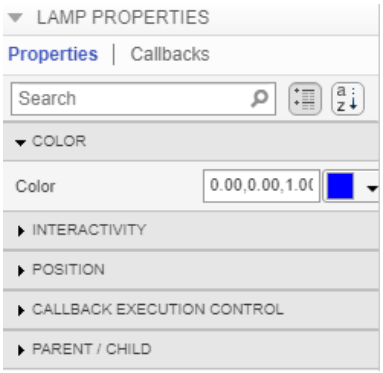
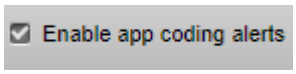



*Figura 21. Ventana Componen Properties*

**5. Editor.** Al igual que la Design View tenía sus propias herramientas, en la Code View ocurre lo mismo.

| Pestaña Editor | |
|---|---|
|  | **Callback.** Permite crear una nueva callback. |
|  | **Function.** Permite crear una nueva función. |
|  | **Property.** Permite crear una nueva propiedad. |
|  | **App Input Arguments.** Permite a la aplicación recibir argumentos de entrada. |
|  | **Go to.** Encuentra la función o la línea indicada. |
|  | **Find.** Busca en el código las palabras indicadas. |
|  | **Comment.** Permite crear o eliminar comentarios. |
|   | **Enable app coding alerts.** Avisa de la aparición de errores y advertencias en el código. |

| | |
|---|---|
|  | **Show Tips**. Muestra consejos en la Code View. |

*Tabla 2. Herramientas de la pestaña Editor.*

**C. Paneles comunes a ambas pestañas.**

Para finalizar con los paneles, existen dos panel comunes a ambas pestañas.

**1. Component browser.** En este panel se puede observar una lista con los elementos que se han creado en la aplicación.

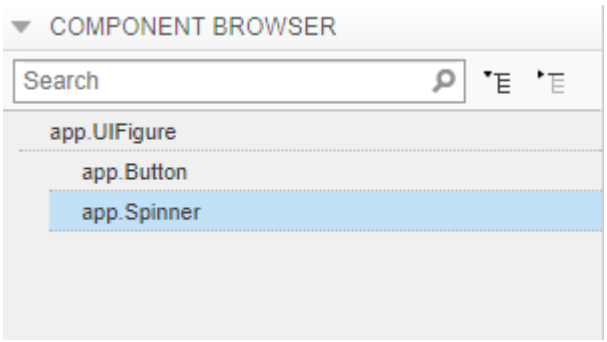



*Figura 22. Ventana Component browser.*

**2. Designer.** Es una pestaña con herramientas que se puede utilizar tanto para la Design View como para la Code View.

| Pestaña Designer | |
|---|---|
|  | **New.** Permite crea una nueva aplicación. |
|  | **Open.** Abre una aplicación existente. |
|  | **Save.** Permite guardar la aplicación que se está trabajando. |
|  | **App Details.** Permite la descripción de detalles de la aplicación. |
|  | **Share.** Crea una aplicación independiente de escritorio. |

| | |
|---|---|
|  | **Run.** Permite ejecutar la aplicación. |

*Tabla 3. Herramientas de la pestaña Designer.*

Una vez conocido el entorno de App Designer se va a comentar como se crean las callbacks y las funciones, así como la utilidad de las propiedades.

**A. Callbacks.**

Las callbacks son funciones que se ejecutan como respuesta a las interacciones predefinidas del usuario de la interfaz con un componente, como por ejemplo pulsar un botón o activar un switch.

Para crear la callback la forma más habitual es ir al panel Component Browser y hacer clic con el botón derecho sobre el nombre del componente para el que se quiera crear la función. Después se selecciona Callbacks y Add callback.

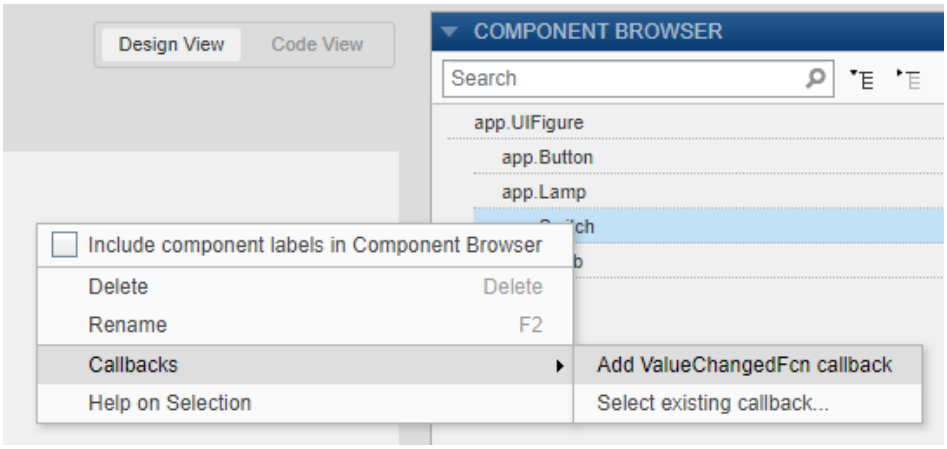



*Figura 23. Creación de callbacks.*

Cuando se ha creado la callback desde la Design View, App Designer abre la Code View y coloca el cursor en la nueva función.

**B. Funciones.**

App Designer dispone de tres tipos de funciones:

**- Función Start-Up**. La función de inicio se ejecuta cuando el usuario abre la aplicación por primera vez, antes de que se hayan creado los objetos que componen la interfaz. Es común utilizarla cuando se quieren inicializar propiedades de los componentes.

**- Funciones de utilidad privada.** Estas funciones realizan tareas que pueden ser reutilizadas dentro de la aplicación, pero fuera de ella no. Se crean cuando es necesario utilizar los mismos comandos en múltiples callbacks, así se consigue evitar que el código esté lleno de sentencias repetidas.

**- Funciones de utilidad pública.** Estas funciones son las realizan tareas que pueden ser usadas tanto dentro como fuera de la aplicación. Son útiles cuando es necesario compartir alguna función con el espacio de trabajo de Matlab, con otra aplicación o con algún dispositivo conectado al ordenador.

Las funciones de utilidad pública o privada se crean desde la Code View y es posible crearlas de dos manera diferentes:

1. Desde el panel Code Browser pulsando en  y posteriormente seleccionando la utilidad si es pública o privada.

2. Desde la pestaña Editor se hace clic sobre la opción function y posteriormente se selecciona la utilidad.

Una vez creada la función, App Designer agrega en el código el marco para la función.

```
methods (Access = private)

        function results = func(app)

        end
```

Este marco es igual para ambas funciones a excepción del atributo Acces.

Para la primera vez que se crea una función privada, en el código se genera un bloque methods privado en el cual se van incluyendo las funciones que se vayan creando posteriormente. De la misma manera ocurre para la función pública, se crea un bloque methods público cuando se agrega por primera vez una función pública.

Cuando ya se tiene creado el bloque methods es recomendable sustituir el identificador de la función (func) por un nombre que se identifique fácilmente.

Para acabar si no es necesario devolver un valor se elimina results.

Las llamadas de estas funciones se realizan de maneras diferentes. Las funciones de utilidad privada, para realizar su llamada, basta con escribir el identificador de la función junto con los argumentos de entrada en la callback correspondiente. Sin embargo, para las funciones de utilidad pública, se necesita que dichas funciones existan en el espacio de trabajo de la aplicación que las llama.

**C. Propiedades.**

Las propiedades se utilizan para compartir datos a través de callbacks. A las propiedades que han sido creadas se les añade el prefijo app por lo que es posible acceder a ellas dentro de la aplicación utilizando la notación de puntos.

Existen dos tipos de propiedades:

- Para compartir datos entre funciones dentro de la aplicación.

Estas propiedades son privadas por lo tanto los valores que estas tienen no están disponibles en el espacio de trabajo de Matlab.

- Para compartir datos dentro y fuera de la aplicación.

Estas propiedades son públicas y se encuentran disponibles en el espacio de trabajo de Matlab al mismo tiempo que la aplicación se está ejecutando.

Además, App Designer crea propiedades de componentes que son las que se encargan de controlar el panel y el comportamiento de los objetos. El usuario no puede crear estas propiedades, pero sí que es capaz de modificar los valores de estas.

Para este apartado la información ha sido sacada de la referencia bibliográfica [7].

## 2.2 KEPSERVEREX 5

Este software está basado en un servidor diseñado para comunicaciones precisas, configuraciones rápidas e interoperabilidad entre aplicaciones de clientes, dispositivos industriales y sistemas. El servidor proporciona una amplia gama de complementos, controladores de dispositivos y componentes que se adaptan a la mayoría de las necesidades de comunicación. La interfaz de usuario proporciona un acceso consistente desde aplicaciones basadas en estándares y aplicaciones no basadas en estándares con interfaces nativas. Esta información ha sido sacada de la referencia bibliográfica [6].

Es necesario crear un nuevo servidor para poder comunicarnos con el resto de los programas. Para crear este servidor primero es necesario abrir el entorno de trabajo (figura 24) y crear un canal ('Channel') haciendo clic sobre el símbolo  .

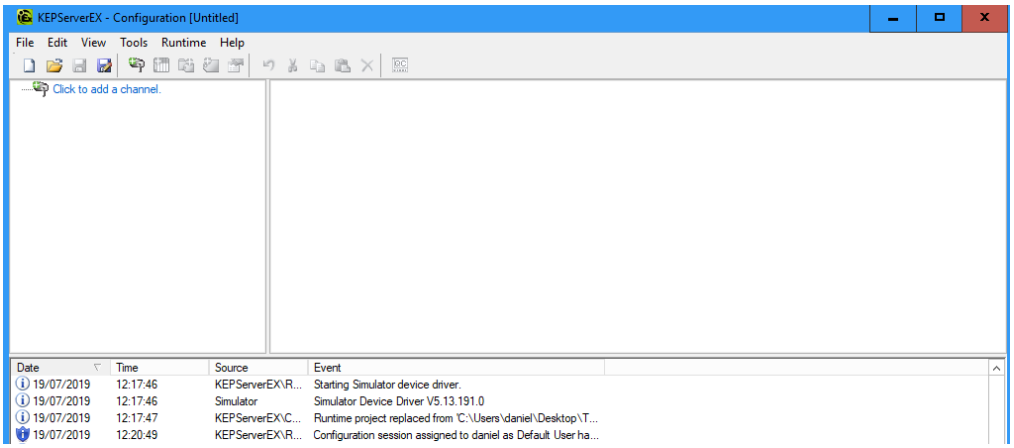



*Figura 24. Entorno de trabajo KEPServerEX.*

Cuando se está creando el canal es necesario elegir el nombre del canal y en las configuraciones hay que elegir el driver Simulator. A través de este canal se comunican con el servidor.

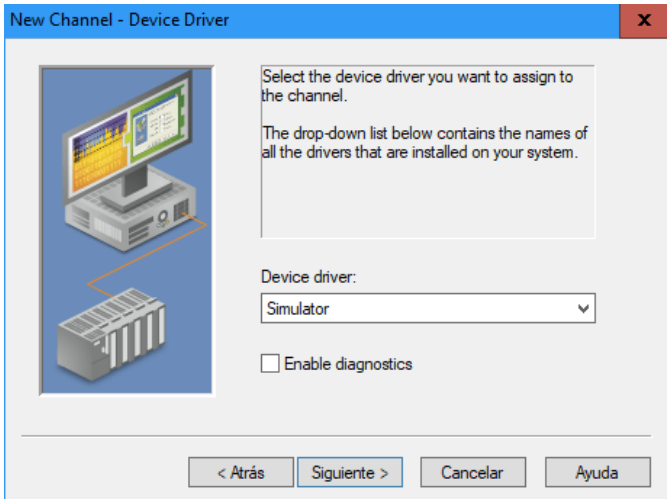



*Figura 25. Creación de canales nuevos.*

Una vez creado el canal, se pasa a crear el dispositivo (‘Device’) que se comunicará mediante el canal que se ha definido previamente.

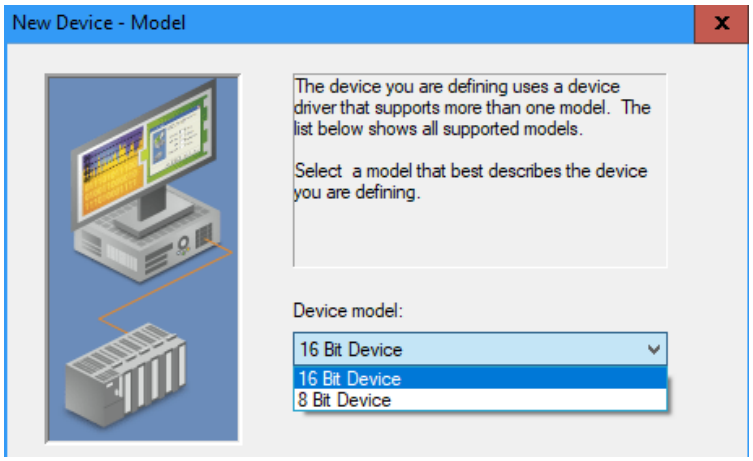



*Figura 26. Creación de nuevo Device.*

Y por último se seleccionan las posiciones de memoria del dispositivo en las que se leerán y se escribirán los datos. Para crear estos Tags es necesario completar los datos de identificación y de propiedades.

Dentro de estos datos se encuentran:

**Name:** Este parámetro se usa para identificar los datos disponibles de la etiqueta.

**Address:** Se utiliza para introducir la dirección del controlador donde se desea que se almacenen los datos.

**Description:** Sirve para hacer comentarios sobre la etiqueta que se está creando.

**Data type:** Este parámetro se utiliza para especificar el formato de los datos de la etiqueta tal como se encuentran en el dispositivo. Estos tipos de datos son:

Default, Boolean, Char, Byte, Short, Word, long, DWord, Float, Double, String, BCD, LBCD y Date.

**Client Access:** Se utiliza para especificar si la etiqueta será de solo lectura o de lectura/escritura.

En la siguiente figura se puede observar la ventana con los parámetros comentados anteriormente.

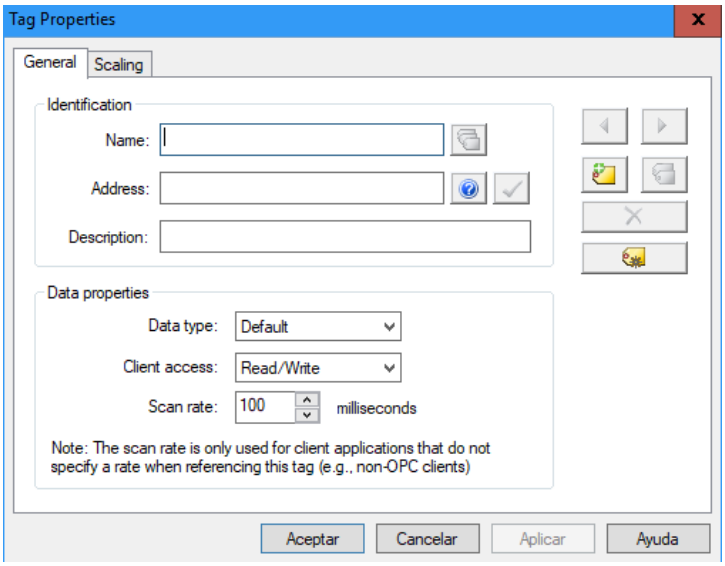



*Figura 27. Ventana Tag Properties.*

Una vez creado el servidor es necesario conectarlo para comunicarse con él. Esto se hace seleccionando de la barra de herramientas Runtime → Connect.

Para comprobar el correcto funcionamiento del programa se puede hacer a través del cliente que lleva incorporado KEPServerEX. Este cliente se abre haciendo clic sobre el símbolo  .

Una vez abierto el cliente y seleccionando el dispositivo (‘Device’) que se había creado se observan todos los tags y es posible escribir en ellos y leer los datos que contienen.

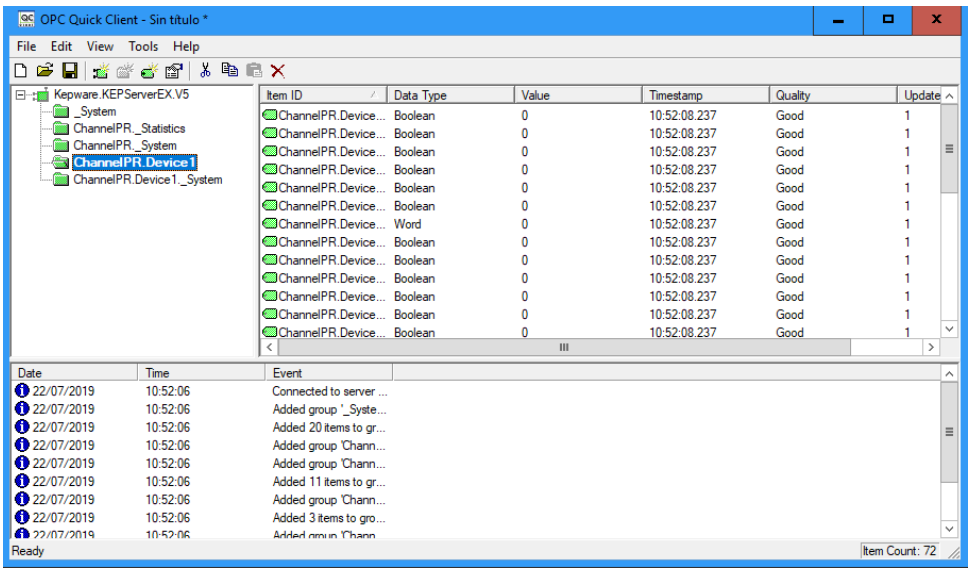



*Figura 28. Ventana de comprobación de etiquetas.*

## 2.3 FACTORY I/O

Es un simulador en 3D de fábricas que permite construir un proceso industrial virtual utilizando una selección de piezas comunes. Factory I/O también tiene escenas inspiradas en aplicaciones industriales.

Las figuras utilizadas y la información para este apartado han sido sacadas de la referencia bibliográfica [9].

### 2.3.1 Interfaz de usuario.

Una vez accedido a Factory I/O se va a comentar el entorno de trabajo del que dispone.

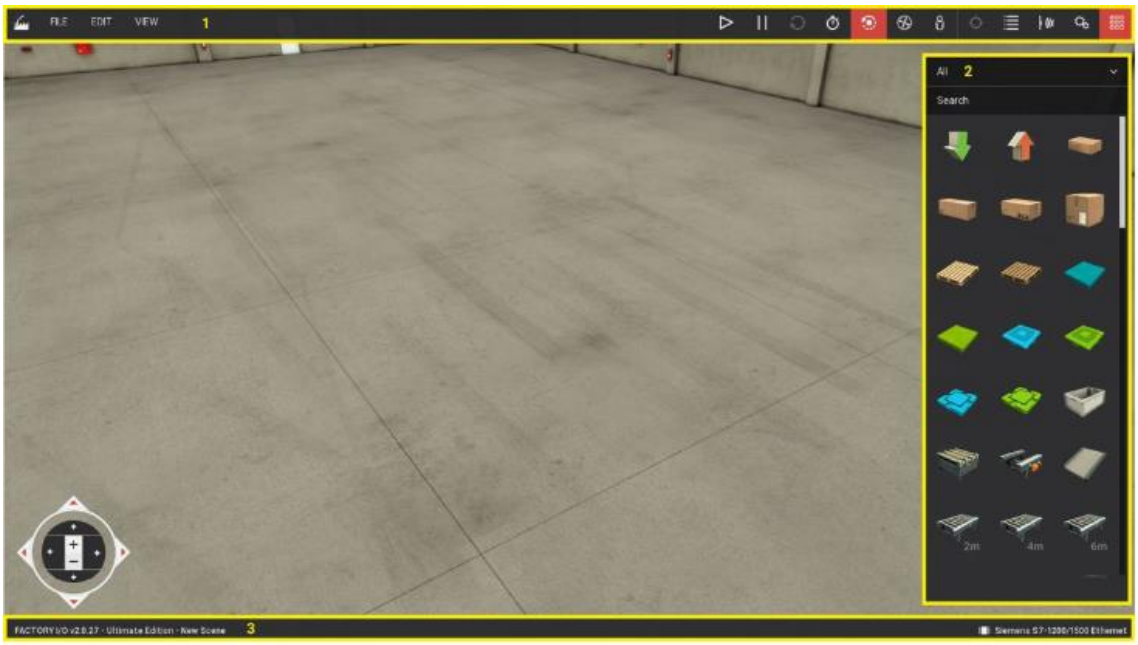



*Figura 29. Entorno Factory I/O.*

En la parte superior se encuentra la barra de herramientas, en esta barra se encuentran los siguientes botones:

| Barra de Herramientas | |
|---|---|
|  | **Ejecutar.** Inicia la simulación. |
|  | **Pausa.** Para la simulación. |
|  | **Restablecer.** Restablece la simulación. |
|  | **Escala de tiempo.** Permite seleccionar la escala de tiempo de la simulación. |

| | |
|---|---|
|  | **Ventana Cámaras.** Muestra la ventana Cámaras. |
|  | **Etiqueta del sensor.** Muestra/ oculta las etiquetas de los sensores. |
|  | **Etiquetas del actuador.** Muestra/oculta las etiquetas de los actuadores. |
|  | **Paleta.** Muestra /oculta la ventana paleta. |

*Tabla 4. Barra de herramientas Factory I/O.*

En la parte derecha se puede observar la venta llamada Paleta que se muestra desde el botón de la barra de herramientas comentado anteriormente. Esta ventana contiene todas las piezas de las que dispone Factory I/O. Para crear una escena basta con arrastrar la pieza de la paleta al espacio 3D. Estas piezas se pueden buscar seleccionando una categoría de la lista o buscar por nombre.


*Figura 30. Ventana Paleta.*

Y por último en la parte inferior se encuentra la barra de estado. Esta muestra la información sobre el estado actual: indica la versión que se está utilizando, la edición, el nombre de la escena y el controlador seleccionado.

### 2.3.2 Cámaras.

Son un elemento clave en Factory I/O ya que muchas tareas requieren de saber cómo manipular correctamente las cámaras. En Factory I/o existen tres tipos de cámaras:

**- Orbit:** Esta cámara facilita las acciones de edición, ya que funciona girando alrededor de un punto establecido al hacer doble clic con el botón izquierdo y se indica mediante un punto blanco.

**- Fly:** Esta cámara permite moverse libremente por el espacio 3D.

**- Primera persona:** Esta cámara simula a una persona de 1,80 m de altura.

Es posible cambiar entre cámaras desde la barra de herramientas.

Factory I/O también dispone de una ventana de cámaras que permite guardar la cámara actual, esto hace posible cambiar fácilmente la perspectiva y poder volver a un lugar guardado. Las cámaras se pueden eliminar en cualquier momento.

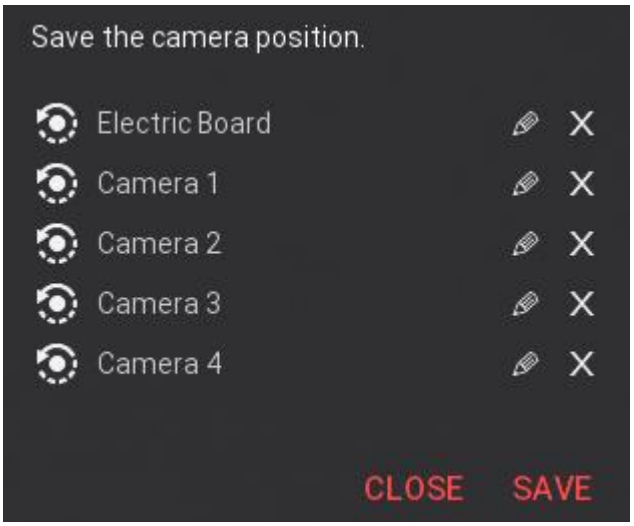



*Figura 31. Ventana para observar las cámaras disponibles.*

Para terminar con las cámaras Factory I/O dispone de la cámara Gizmo que puede realizar gran parte de las acciones que se han descrito. La cámara Gizmo se puede mostrar/ocultar desde el menú Ver.

*Figura 32. Cámara Gizmo.*

### 2.3.3 Etiquetas.

Las etiquetas sirven para vincular valores de actuadores y sensores al controlador. Cada actuador o sensor puede tener una o más etiquetas. Estas etiquetas también pueden utilizarse para controlar los actuadores manualmente.

Las etiquetas están formadas por un nombre y un valor. Estas pueden ser de tres tipos de datos en función del tipo de actuadores y sensores:

**- Booleano:** Para valores de On/Off.

**- Flotante:** Para valores analógicos.

**-Entero:** Para datos específicos.

Estas etiquetas se pueden mostrar haciendo clic en los iconos de la barra de herramientas que se han comentado anteriormente.

Es posible forzar las etiquetas haciendo clic con el botón izquierdo sobre el botón de la etiqueta, el control deslizante o el campo de entrada. Forzar las etiquetas de los actuadores anulará los valores leídos del controlador y permite controlar la escena manualmente. Al forzar las etiquetas de los sensores hará que se envíe al controlador, aunque el sensor no esté detectando un elemento.

Para liberar los valores forzados se debe hacer clic izquierdo sobre Release.

También existen las etiquetas de simulación. Estas son etiquetas incorporadas en cada escena que se pueden utilizar para que el controlador pueda obtener datos de la simulación (sensores) o para controlar la simulación (actuadores).

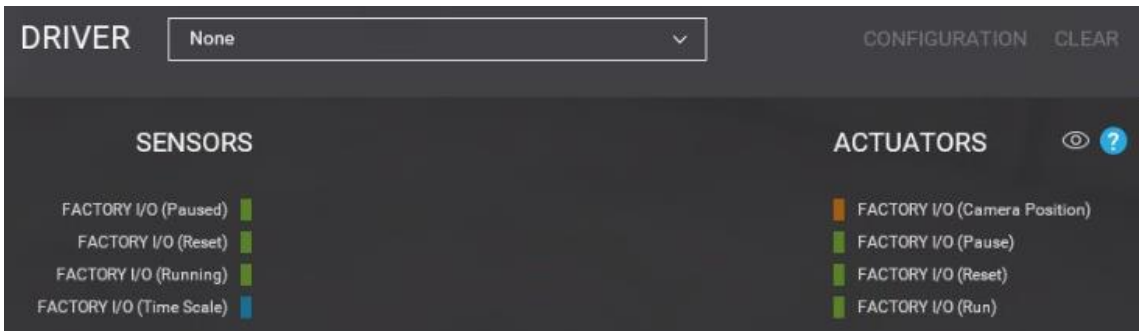



*Figura 33. Ventana para observar las etiquetas de simulación.*

### 2.3.4 Controladores de Entrada/Salida.

Los controladores de E/S son funciones que incorpora Factory I/O y permiten comunicarse con un controlador externo. Factory I/O dispone de una gran variedad de controladores de E/S, cada uno de estos se utiliza con una tecnología específica.

Para mostrar el menú de controladores basta con hacer clic en  de la barra de estado.

Una vez abierto el menú de controladores es necesario elegir el controlador deseado y configurarlo. Después de su configuración se asignan las etiquetas de los actuadores y de los sensores a los puntos de E/S del controlador.

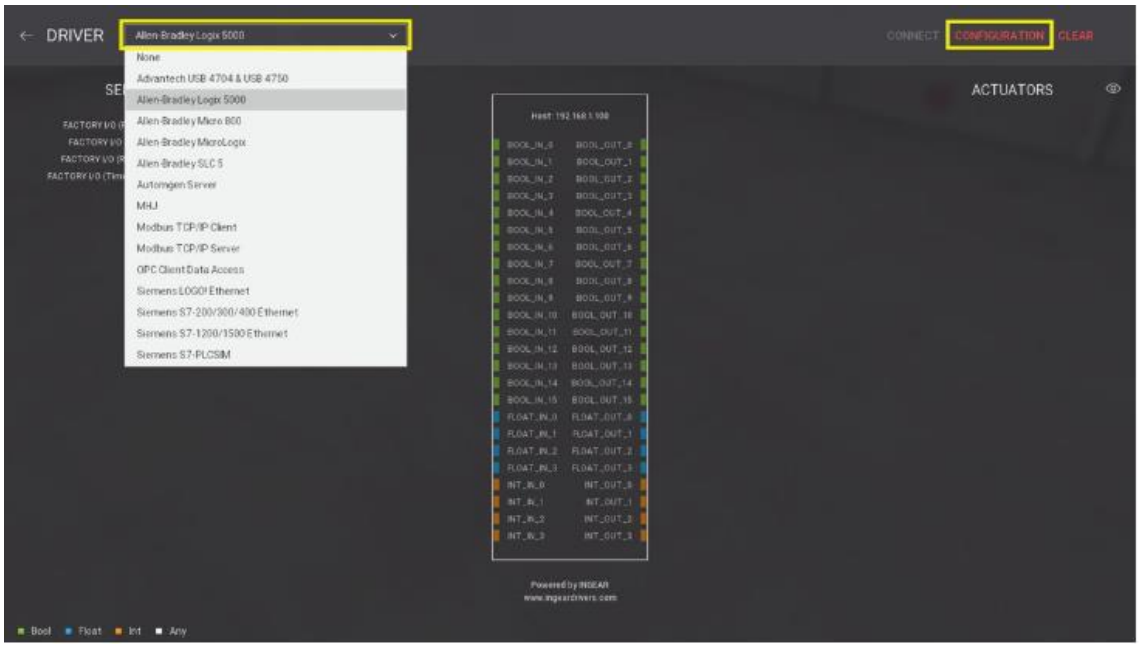



*Figura 34. Menú de controladores.*

# CAPÍTULO 3. COMUNICACIÓN ENTRE PROGRAMAS

La comunicación entre Stateflow de Matlab y Factory I/O se va a realizar mediante la herramienta OPC Toolbox de Matlab y el servidor OPC KERServerEX, usando una estructura de cliente/servidor OPC.

### 3.1. DEFINICIÓN DE OPC.

"Es una tecnología de comunicación con una arquitectura de cliente y servidor. Una aplicación actúa de servidor proporcionando datos y otra actúa como cliente leyéndolos o manipulándolo." Definición sacada de [10].

OPC permite el intercambio de información con múltiples aplicaciones y dispositivos sin restricciones o límites. Los servidores OPC pueden estar en continua comunicación con los PLCs u otras aplicaciones en tiempo real. Estos son la fuente de datos y cualquier dispositivo o aplicación basada en OPC puede acceder al servidor para leer/escribir cualquier variable de las que ofrece el servidor.

Por ello, OPC ha conseguido mejorar la cooperación entre los proveedores y usuarios dotando a los consumidores de más poder para elegir entre diferentes aplicaciones industriales.

Existen cuatro tipos de servidores OPC:

**- DA:** Se basa en las especificaciones de Acceso de Datos (Data Access). Su principal cometido es transmitir datos en tiempo real.

**- HDA:** Sus especificaciones se basan en el Acceso de Datos Historizados. Este servidor da los datos históricos al cliente OPC.

**- A&E:** Se basa en los que especifican Alarmas y Eventos. Este transmite alarmas y eventos al cliente.

**- UA:** Se basa en un sistema de Arquitectura Unificada. Este servidor es capaz de trabajar con cualquier dato, sea del tipo que sea.

## 3.2 OPC TOOLBOX DE MATLAB.

La caja de herramientas OPC de Matlab permite acceder a datos OPC directamente desde Matlab y Simulink y es capaz de leer, escribir y registrar datos OPC desde dispositivos.

Para instalar esta caja de herramientas en Matlab es necesario utilizar la función `opcregister`. Con esta función también es posible quitar o reparar la instalación de los componentes principales.

Una vez instalada la caja de herramientas, cuando se utiliza en Simulink se usa un bloque de Configuración para especificar los clientes de acceso a datos OPC para usar en el modelo y establecer el comportamiento en tiempo real. El bloque de Configuración se puede usar para definir el comportamiento de la caja de herramientas si la simulación se ejecuta más lenta que el reloj del sistema.

Además del bloque de configuración, también están los bloques OPC Read y OPC Write que recuperan y transmiten datos tanto de forma síncrona como de forma asíncrona hacia el servidor y desde el servidor OPC DA. Estos bloques tienen un administrador de clientes que permite especificar y administrar el servidor OPC DA, seleccionar elementos y definir tiempos de muestra de bloques.

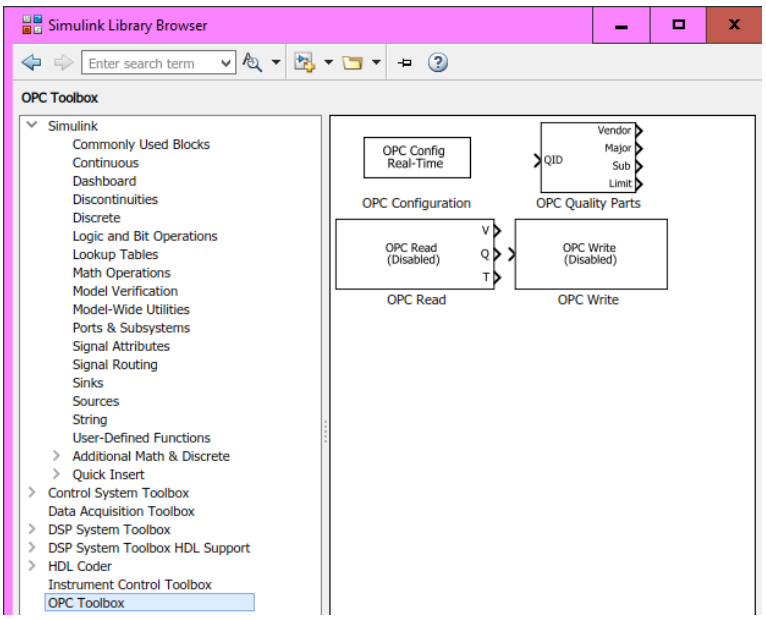



*Figura 35. Bloques OPC Toolbox.*

Para este capítulo la información ha sido sacada de [5] y [11].

# CAPÍTULO 4. PLATAFORMA SOFTWARE

En este capítulo se va a explicar desde la creación de la plataforma software utilizando los programas que han sido comentados anteriormente hasta su utilización en una aplicación industrial disponible en Factory I/O, pasando por la configuración de los distintos programas informáticos.

## 4.1 CREACIÓN DE LA PLATAFORMA.

Para crear la plataforma es necesario acceder a Simulink como se ha comentado en capítulos anteriores. Una vez dentro se crea el modelo de la plataforma, para crear este modelo se añade de la biblioteca de Simulink el bloque Chart de Stateflow, en este bloque será donde se implemente el grafcet asociado a la aplicación industrial que se desee automatizar. También se añaden los bloques de la caja de herramientas OPC Toolbox: OPC Configuration, OPC Read y OPC Write. Estos bloques son los que introducirán el valor de los datos de los sensores y actuadores al bloque Chart y escribirán el cambio de estos valores a través de su comunicación con el servidor OPC de KEPServerEX. Para el envío de lectura se añade un Demux de la biblioteca de Simulink conectado a la salida del bloque OPC Read y del cual salen tantas salidas como datos necesite leer el bloque Chart y, además, para la escritura se añade un Mux al cual llegan tantas entradas como datos se quieran escribir y se conecta a la entrada del bloque OPC Write.

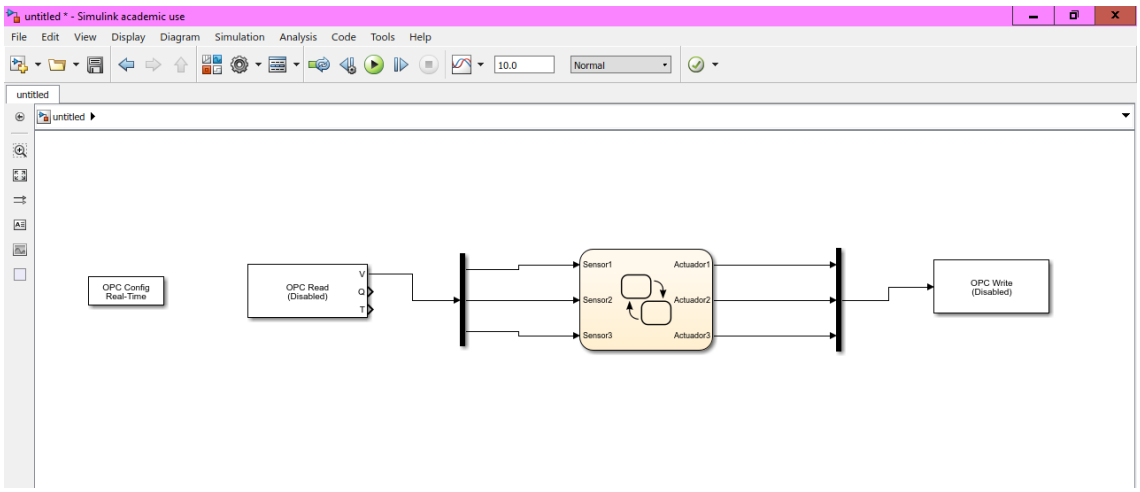



*Figura 36. Plataforma software.*

## 4.2 IMPLEMENTACIÓN DEL GRAFCET EN LA PLATAFORMA.

Para comprobar que esta plataforma software es capaz de controlar aplicaciones industriales virtuales, se ha escogido una escena de Factory I/O la cual se va a automatizar en dicha plataforma.

De entre todas las escenas que dispone Factoy I/O se ha escogido el Ensamblador. Esta escena consiste en ensamblar dos partes, una base y una tapa, utilizando una selección de dos ejes y un lugar.

Para conseguir la automatización de esta escena se implementará el grafcet correspondiente a dicha escena en Stateflow, es decir dentro del bloque Chart de la plataforma.

En el grafcet que se ha implementado (ver Anexo I) se pueden observar dos estados principales que corresponden a los modos de funcionamiento y al modo de emergencia. Para pasar del estado Funcionamiento al estado Emergencia sólo basta con pulsar la seta de emergencia del cuadro de mandos, mientras que para el caso contrario hay que desactivar dicha seta de emergencia y poner en manual el modo de funcionamiento, no basta con desactivar sólo la seta de emergencia.

Cuando se activa el estado Emergencia el funcionamiento del proceso industrial se detiene por completo. Estos estados se ejecutan de manera alternativa, es decir no se pueden ejecutar los dos al mismo tiempo.

Dentro del estado Funcionamiento se observan tres estados: Inicio, Manual y Automático. El estado Manual se corresponde al modo de funcionamiento del proceso de manera manual, al igual que el estado Automático corresponde al modo de funcionamiento automático. El estado Inicio es el primero que se ejecuta cuando se encuentra activo el estado Funcionamiento y cambia a uno de los otros dos estados según qué modo de funcionamiento se quiera emplear.

Los dos modos de funcionamiento son:

- **Modo Manual:** Este modo se corresponde al estado Manual, como su nombre indica. Dentro de este se encuentran tantos estados como actuadores tiene la escena de Factoy I/O, estos se ejecutan de forma paralela. Además, hay un estado para el botón de reset que permite poner el contador a cero. Este modo de funcionamiento se controlará a través de un panel de control.

- **Modo Automático:** Dentro de este modo se encuentran cuatro estados que se ejecutan de forma paralela. El primero de ellos, CintaTapas, corresponde al grafcet asociado al funcionamiento de la cinta por la que circulan las tapas y del funcionamiento de los ejes que permiten transportar la tapa hasta ponerla encima de la base. El segundo, CintaBases, corresponde al funcionamiento de la cinta de las bases. Y los dos últimos corresponden a los estados de Paro y Reset. Estos se activan cuando se pulsan los botones de paro o reset. El botón de paro se utiliza para terminar de producir, cuando se pulsa este botón el proceso industrial termina lo que estaba ejecutando y no continua. El botón de reset permite poner a cero el contador.

## 4.3 CONFIGURACIÓN DE LOS PROGRAMAS.

Para la comunicación entre programas informáticos es necesario crear un servidor en KEPServerEX para que la plataforma se comunique con este servidor y pueda leer/escribir los datos de los distintos sensores y actuadores.

Para crear dicho servidor primero hay que crear un canal por el cual se comunicarán. En la creación del canal es necesario elegir un nombre para este y configurar el driver como Simulator.

Una vez creado el canal se crea el dispositivo ('Device'). Y por último se crean las etiquetas para cada uno de los sensores y actuadores que tiene la escena. Con la creación de estas etiquetas se están seleccionando la posición de memoria del dispositivo donde se leerán y se escribirán los datos. La mayoría de estas etiquetas son de tipo booleano, exceptuando el correspondiente al contador que es un word.

Esta comunicación en Matlab se realiza a través de la caja de herramientas OPC Toolbox, la cual será necesario configurar. La configuración se realiza de la siguiente forma:

En primer lugar, en el bloque OPC Configuration se configura el OPC que se va a utilizar. Para ello se abre dicho bloque y se configura el OPC cliente añadiendo el servidor que se va a utilizar, en nuestro caso KEPServerEX.

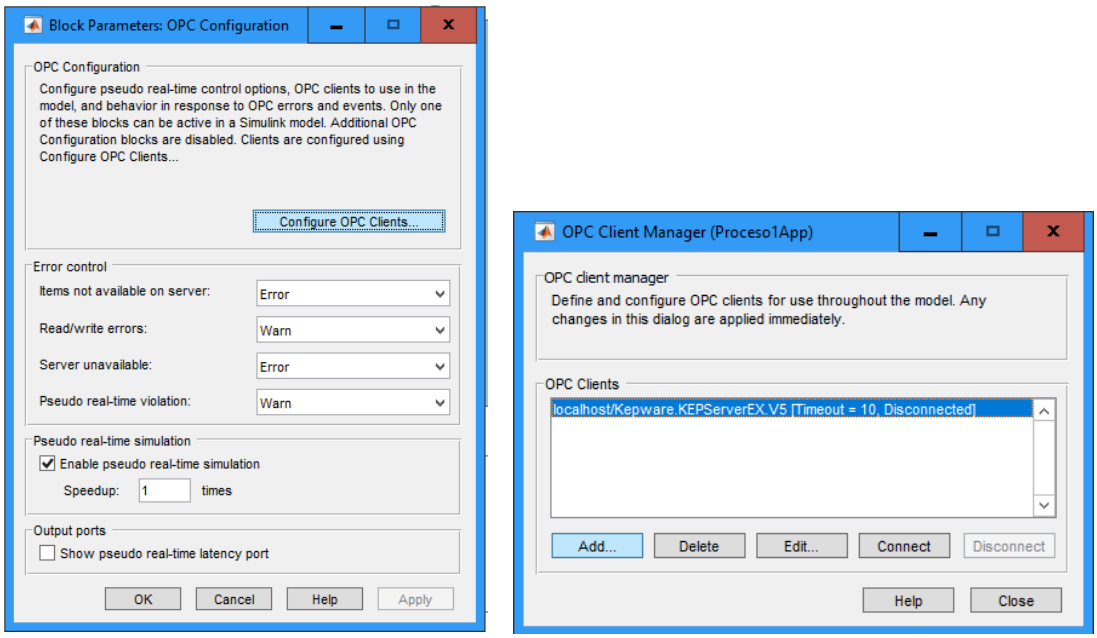



*Figura 37. Configuración del OPC.*

Una vez configurado el OPC cliente se agregan al bloque OPC Read las etiquetas que requieran ser leídos por el bloque Chart. Para ello se abre el bloque y en Add Items se seleccionan las etiquetas que se requieran del canal que se ha creado.

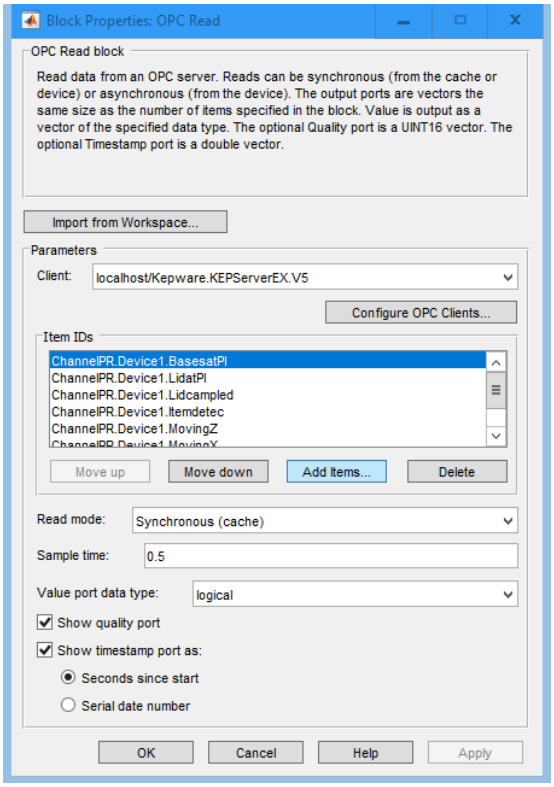



*Figura 38. Propiedades bloque OPC Read.*

Para el bloque OPC Write se procede del mismo modo, pero en este caso se añaden las etiquetas que requieren cambiar su valor.

Para conectar las entradas que han sido agregadas al bloque OPC Read con Stateflow, bloque Chart, se realiza a través de un Demux, disponible en la biblioteca de Simulink, al cual se conecta la salida del OPC Read y de él salen tantas salidas como etiquetas han sido agregadas. Algunos de los sensores actúan mediante flancos de subida o bajada, por lo que para conseguir simular estos flancos en Simulink se utiliza un Switch en la entrada al bloque Chart de cada uno de los sensores que actúen con flancos. Cada etiqueta se conecta al puerto de entrada a Stateflow que le corresponda.

También se puede observar que existen entradas que no provienen del bloque OPC, estás entradas se corresponden a los actuadores y sensores que serán controlados desde el panel de control que se creará en App Designer, como los actuadores cuando se encuentre en modo manual o los botones de start, paro, etc.

Las salidas de Stateflow se conectan con el bloque OPC Write a través de un Mux, al cual le llegan todas las salidas y la salida de este se conecta con el OPC Write. Es importante que todas las salidas que llegan al OPC Write estén añadidas en dicho bloque.

Una vez creado el servidor OPC y configurada su comunicación con Matlab, se pasa a configurar la escena de Factory I/O. Para su configuración en primer lugar es necesario abrir el menú de controladores como se ha visto anteriormente y seleccionar el driver OPC Client DA/UA. Una vez seleccionado el driver se pasa a configurarlo, para ello se pulsa en el botón configuración y se abre la siguiente vista.

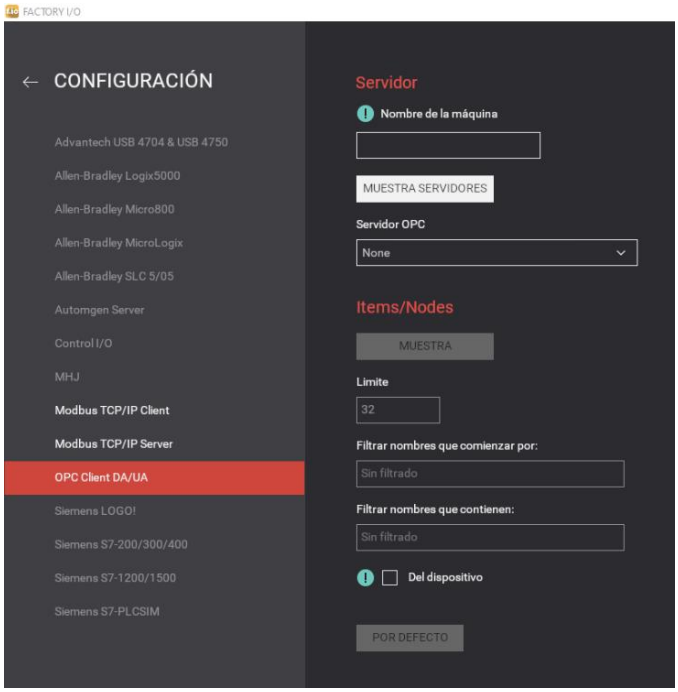



*Figura 39. Configuración escena Factory I/O.*

Del apartado Servidor se selecciona el servidor OPC que se va a utilizar, en este caso KEPServerEX y del apartado Items se pone el límite en 64 para que se muestren todas las etiquetas que han sido creadas y para encontrar las etiquetas se filtran mediante el nombre del canal en el que han sido creadas.

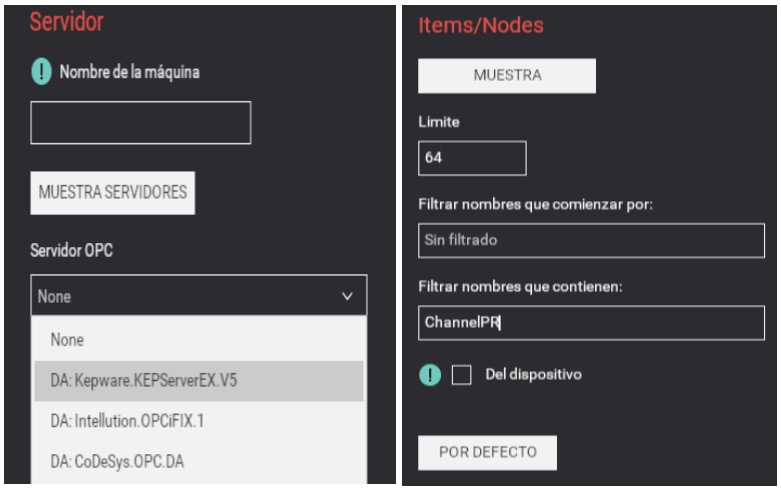



*Figura 40. Selección del servidor OPC y búsqueda del canal.*

Después de esta configuración en el menú de controladores aparecerán todas las etiquetas y estas se asociarán a los sensores o actuadores correspondientes de la escena. Para asociarlas solo hace falta pulsar en el nombre de un sensor o actuador y arrastrarlo hasta su etiqueta correspondiente. En la siguiente imagen se puede ver como quedaría.


*Figura 41. Asociación de etiquetas.*

Una vez han sido asociadas todas las etiquetas ya se habría completado la configuración y se vuelve a la escena y se comprueba que el servidor está conectado. Para ello se observa la barra de estados y debe aparecer el nombre del servidor con un tic en verde.

## 4.4 INTERFAZ GRÁFICA DE USUARIO.

Después de comprobar que todas las configuraciones están correctamente y que los programas se comunican entre ellos, se procede a crear una interfaz gráfica de usuario en App Designer.

Esta interfaz gráfica de usuario nos permitirá controlar el modo de funcionamiento manual, así como los cambios de funcionamiento o los botones de start, paro, reset y emergencia. Además, dicha interfaz dispone de una representación gráfica de la escena en la cual se pueden observar los sensores y los actuadores que están activos en cada momento de la producción.

En este apartado se explicarán las distintas funciones, propiedades y componentes que se han utilizado para la interfaz programada. Para la programación de la interfaz se ha consultado la referencia bibliográfica [8].

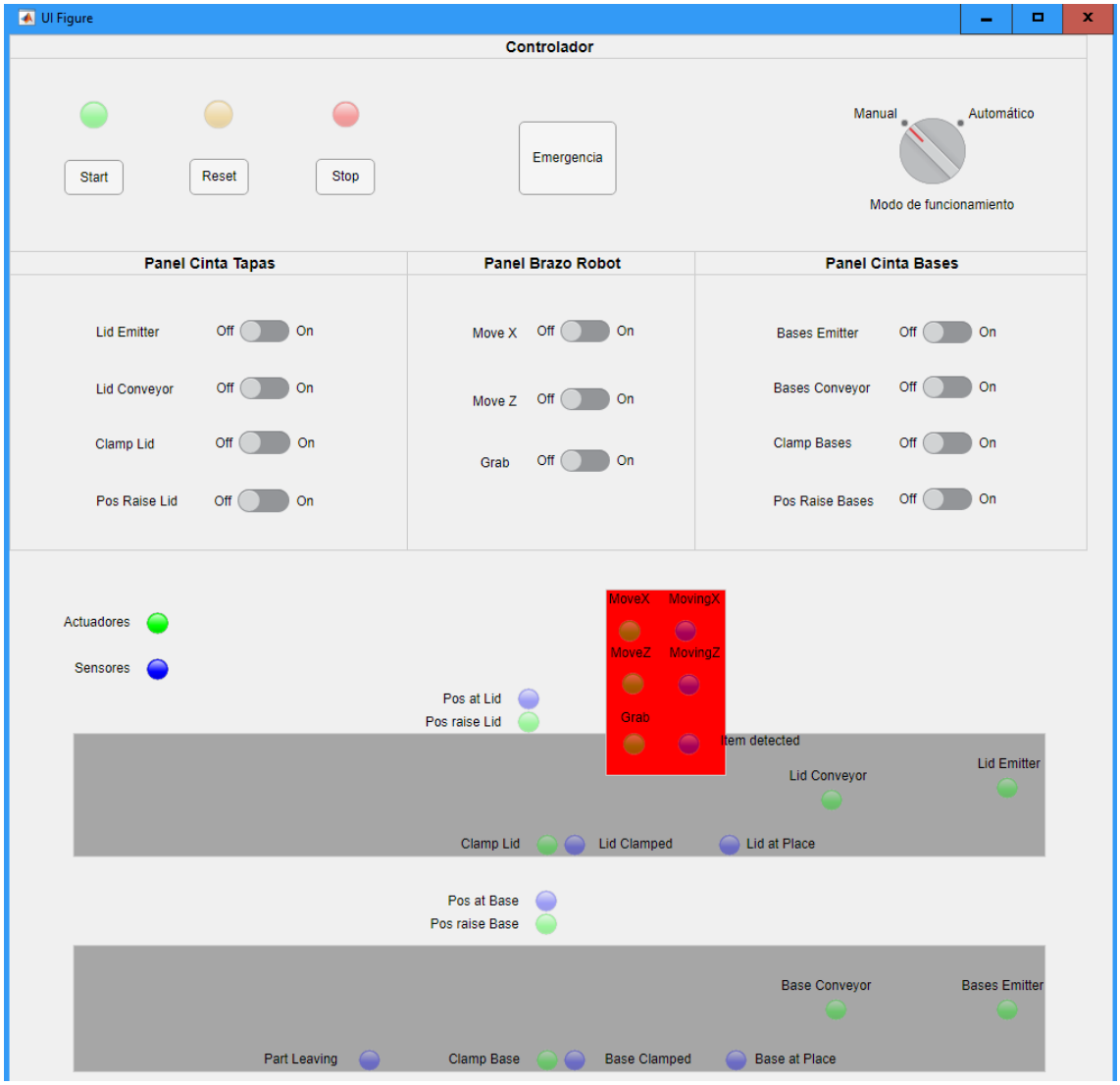



*Figura 42. Interfaz de usuario.*

### 4.4.1 Función Start-Up.

La función de inicio (Start-Up) se ejecuta la primera vez que se abre la aplicación. Esta función contiene una de las partes más importantes del programa ya que contiene la configuración que permite la comunicación entre la interfaz y el servidor OPC.

Para conseguir dicha comunicación se procede de la siguiente manera:

```
da = opcda('localhost','Kepware.KEPServerEX.V5');
connect(da);
grp = addgroup(da);
app.itm0 = additem(grp,'ChannelPR.Device1.Lidsemitter');
app.itm1 = additem(grp,'ChannelPR.Device1.Basesemitter');
```

Donde la función opcda es la que configura la comunicación y connect la que la inicia. Dentro de la función opcda, `'Kepware.KEPServerEX.V5'` es el nombre del servidor OPC que se está utilizando.

La función addgroup añade un grupo de variables que se corresponden a itm0, itm1, etc, que serán donde se almacenen los datos de cada una de las etiquetas.

### 4.4.2 Función de Utilidad.

En el código implementado se encuentra la función de utilidad **getModelValues (app).**

Esta función se encarga de leer los valores de los sensores y actuadores cada cierto periodo de tiempo, y en función de estos valores se encienden las bombillas correspondientes en la interfaz gráfica.

Para que esta función se repita cada periodo de tiempo es necesario incluir un objeto timer en las properties.

```
properties (Access = private)
        timerObj %timer
```

Además, en la función **startupFcn (app)** se declara y se inicializa la función **getModelValues (app)** con el siguiente código:

```
app.timerObj=timer('TimerFcn',@(~,~)getModelValues(app),'Period',0.1,
'ExecutionMode','fixedSpacing','BusyMode','drop');
start(app.timerObj);
```

Donde el nombre de la función se pone tras `@(~,~)` y `'Period'` tiene el valor en segundos.

Por lo tanto, esta función se ejecutará cada 0.1 segundos.

### 4.4.3 Propiedades.

Las propiedades declaradas en la interfaz son las siguientes:

**timerObj**

Como se ha comentado en el punto anterior esta propiedad permite incluir un objeto de tiempo para que la función **getModelValues** se puede ejecutar cada cierto periodo.

**itm0, itm1…**

Estas propiedades se utilizan para almacenar los datos que se extraen del servidor OPC para cada una de las etiquetas. Por lo tanto, hay tantas propiedades como etiquetas. De la propiedad itm0 hasta la itm10 se utilizan para las etiquetas de los actuadores y desde la itm11 hasta la itm20 para los sensores.

### 4.4.4 Componentes de la Interfaz.

En este apartado se describirán los objetivos de los distintos componentes que se han utilizado en la creación de la interfaz.

La interfaz gráfica se divide en cuatro paneles que se explican a continuación.

**Controlador**

El primer panel que se va a comentar tiene el nombre de controlador, en él contiene los componentes principales de la interfaz. En primer lugar, se encuentra la ruleta de modo de funcionamiento, la cual nos permite seleccionar el modo de trabajo. A continuación, se observan cuatro botones: El primero de ellos y más importante es el botón de emergencia, el cual si se pulsa detiene por completo el proceso. Y los otros tres botones restantes corresponden a las acciones de Start, Stop, y Reset, cada uno de estos botones tiene asociado una lámpara de distintos colores que se enciende cuando estas acciones están activas. Estos colores son verde para el start, rojo para el stop y amarillo para el reset.

**Cinta Tapas**

Este panel contiene los actuadores correspondientes a la cinta por la que circulan las tapas y se utilizan para el modo de funcionamiento manual, ya que para el modo automático no es necesario. Para representar estos actuadores se han utilizado unos switches, uno para cada actuador. Cuando el switch está en On quiere decir que el actuador está activo y para Off dicho actuador está desactivado.

**Cinta Bases**

En este panel al igual que el anterior contiene los actuadores correspondientes, pero en este caso para la cinta por la que circulan las bases. Al igual que el panel anterior, este también contiene un switch para cada actuador.

**Brazo Robot**

Al igual que los dos paneles anteriores contiene los actuadores en este caso para la selección de dos ejes y el lugar.

Para todos estos paneles se han utilizado las funciones callbacks para cada uno de los componentes. El código de estas funciones se puede observar en el Anexo II**.**

Por último, la interfaz tiene una representación gráfica de la escena, la cual contiene una lámpara que está asociada a cada uno de los actuadores y sensores que tiene dicha escena. Estas lámpara se encienden y se apagan a medida que se van activando y desactivando sus sensores o actuadores correspondientes. Para los sensores se ha utilizado el color azul y para los actuadores el color verde. Por lo tanto, con esta representación gráfica se puede observar cómo está funcionando el proceso.

### 4.5 FUNCIONAMIENTO DE LA PLATAFORMA.

Una vez se ha explicado como es el funcionamiento de los componentes de la interfaz gráfica, se va a explicar cuál es el procedimiento para utilizar la interfaz.

En primer lugar, hay que comprobar que tanto el servidor OPC de KEPServerEX como el servidor de Factory I/O están conectados.

Una vez hecha esta comprobación se ejecutará la interfaz por medio del Run, al igual que la plataforma en Matlab y la escena en Factory I/O. Tras la ejecución de estos programas todo pasa a controlarse a través de la interfaz gráfica.

En las siguientes imágenes se puede observar un instante del proceso en la escena de Factory I/O como en la interfaz gráfica. En estas imágenes se puede comprobar como en la interfaz gráfica se van encendiendo las lámparas de los sensores y actuadores a medida que estos están activos y con ello se puede seguir el proceso que se está realizando.


*Figura 43. Instante de la escena de Factory I/O.*

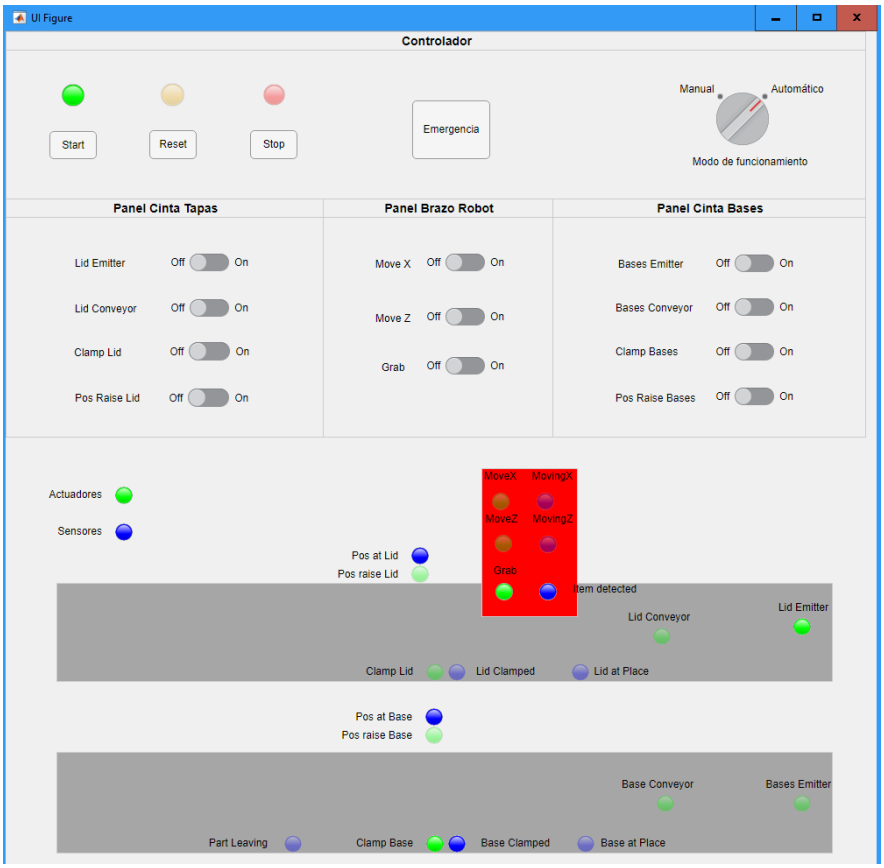



*Figura 44. Interfaz de usuario para un instante.*

Además, también es posible seguir el proceso observando los estados que están activos en el bloque Chart de la plataforma.

# CAPÍTULO 5. CONCLUSIONES

La realización de este proyecto culmina en la creación de una plataforma software realizada en Matlab, la cual a través de un único PC e instalando Matlab, KEPServerEX y Factory I/O permite realizar la implementación de un automatismo, y verificar su correcto funcionamiento sobre una reproducción virtual de la planta industrial.

Mediante esta plataforma se consigue evitar la inversión que supondría la compra de un PLC y de una maqueta industrial en una primera fase de un proyecto de automatización, lo que permitiría a la empresa reducir gastos en el caso de que el automatismo no fuera el deseado.

En caso de que los resultados de esta primera fase fuesen positivos se podría acometer la adquisición tanto del PLC como de la planta industrial.

Con todo ello se puede decir que el proyecto deriva en un producto completo, ya que permite la implementación de cualquier automatismo y comprobar su funcionamiento simplemente con un PC en el cual se puedan instalar los programas informáticos necesarios y sin la necesidad de hacer una gran inversión en una primera fase de proyecto.

# CAPÍTULO 6. BIBLIOGRAFÍA


[1] Introducción al Stateflow en MATLAB-Simulink. Ingeniería Mecatrónica. [Consulta: 10 de junio 2019]. Disponible en: http://ute-mecatronica.blogspot.com/2014/02/introduccion-al-stateflow-en-matlab.html

[2] Toolbox Stateflow. Monografias.com [Consulta: 12 de junio 2019]. Disponible en: https://www.monografias.com/trabajos22/toolbox-stateflow/toolbox-stateflow.shtml

[3] The MathWorks (2003). *Stateflow and Stateflow Coder.* [Consulta: 12 de junio 2019]. Disponible en: http://fractale.gecif.net/si/logiciels/matlab/stateflow.pdf

[4] Stateflow. MathWorks. [Consulta: 3 de julio 2019]. Disponible en: https://es.mathworks.com/help/stateflow/index.html

[5] OPC Toolbox. MathWorks. [Consulta: 16 de julio 2019]. Disponible en: https://es.mathworks.com/help/opc/

[6] Raúl Simarro Fernández. Transparencias: Seminario: Automatización Integrada. [Consulta: 2 de julio 2019]. Disponible en: https://poliformat.upv.es/access/content/group/GRA_13237_2018/Automatizaci%C3%B3n/Se Seminar%20Automatizaci%C3%B3n%20de%20Procesos.pdf

[7] App Designer. MathWorks. [Consulta: 18 de julio 2019]. Disponible en: https://es.mathworks.com/help/matlab/app-designer.html?s_tid=CRUX_lftnav

[8] Setparam. MathWorks. [Consulta: 19 de julio 2019]. Disponible en: https://es.mathworks.com/help/xpc/api/setparam.html

[9] Manual. Factory I/O. [Consulta: 20 de junio 2019]. Disponible en: https://factoryio.com/docs/manual/index.html

[10] Logitek Team (2019). *Qué es OPC y qué es un OPC Server*. [Consulta: 21 de junio 2019]. Disponible en: https://www.kepserverexopc.com/que-es-opc-y-que-es-un-opc-server/

[11] Vester Industrial Training Center (2018). *El servidor OPC al detalle*. [Consulta: 24 de junio 2019]. Disponible en: http://www.cursosingenieriaindustrial.com/servidor-opc-detalle/

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

**TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES**

# DESARROLLO DE UNA PLATAFORMA SOFTWARE DE BAJO COSTE IMPLEMENTADA EN MATLAB PARA LA AUTOMATIZACIÓN DE PLANTAS INDUSTRIALES VIRTUALES

# PLIEGO DE CONDICIONES

**Curso Académico: 2018-19**

# CAPÍTULO 1. OBJETO DEL PLIEGO DE CONDICIONES

El presente documento hace referencia a las exigencias de índole técnica y legal que han de regir la ejecución del proyecto. Así pues, se muestran los parámetros que condicionarán el diseño.

El principal objetivo que persigue el presente proyecto es el diseño de una plataforma software de bajo coste que permita realizar la implementación de un automatismo. Con este proyecto se busca además de la creación de la plataforma, remplazar el uso de PLCs comerciales en la automatización de plantas industriales virtuales. Con ello se reducirían los costes en una primera fase de diseño de un proyecto de automatización.

Para el correcto funcionamiento del proyecto deberán ser tenidas en cuenta las condiciones técnicas que se exponen en los siguientes apartados. Es por esto por lo que la omisión de cualquiera de las especificaciones que se detallan a continuación podría presentar dificultades en el proceso de diseño.

# CAPÍTULO 2. CONDICIONES PARA EL DESARROLLO DEL PROYECTO

## 2.1 INTRODUCCIÓN.

En el siguiente apartado se presenta de forma detallada las condiciones que se deben cumplir en las estaciones de trabajo y su entorno para poder realizar así de una manera eficiente y correcta el desarrollo del proyecto.

A continuación, se describen los aspectos que se van a analizar en este apartado:

- **Equipos de trabajo.**

  Para realizar un diseño de una planta industrial virtual por ordenador en 3D es necesario el uso de equipos que contengan un microprocesador con una alta velocidad de reloj y también una potente tarjeta gráfica que permita la ejecución fluida de los programas de diseño por ordenador, estas caracterizadas en su mayoría por consumir una gran cantidad de recursos del sistema principalmente en memoria RAM y tarjeta gráfica.

- **Programas utilizados.**

  Los programas que se utilizarán para llevar a cabo el desarrollo de la plataforma serán Matlab, KEPServerEX y Factory I/O. En Matlab se desarrollará la plataforma en la cual se implementarán los automatismos de la planta industrial a automatizar y, además se desarrollará la interfaz gráfica que permitirá controlar dicha planta. Factory I/O será el software que permita diseñar la planta industrial que se quiere automatizar y por último KEPServerEX será el servidor OPC que permitirá la comunicación entre los distintos programas que se han utilizado.

- **Puesto de trabajo.**

  La realización de este proyecto requiere la necesidad de estar sentado frente a la pantalla del ordenador a lo largo de muchas horas, por ello, el puesto de trabajo en el cual se realice deberá estar ergonómicamente adaptado para evitar dolores o molestias musculares que perjudiquen a la realización de la tarea. Debido a la importancia que tiene este aspecto, se le dedicará un apartado exclusivo, **3. Condiciones del puesto de trabajo.**

## 2.2 CONDICIONES DE LOS EQUIPOS DE TRABAJO.

La descripción de los equipos de trabajo empleados para la realización del proyecto servirá para presentar las especificaciones técnicas del ordenador utilizado.

Ordenador del puesto de trabajo.

A través de este ordenador portátil se realiza todo el proceso de desarrollo de la plataforma como todas las actividades relacionas como la creación de la planta industrial virtual o la creación de un servidor OPC para la comunicación.

- Procesador Intel Core i7 – 4500U de 1.8 GHz.
- Memoria RAM 4 GB.
- Unidad de almacenamiento, disco duro, de 750 GB.
- Tarjeta gráfica NVIDIA GeForce 820M.
- Sistema operativo Windows 10 Home.
- Ratón óptico inalámbrico.

# CAPÍTULO 3. CONDICIONES DEL PUESTO DE TRABAJO.

## 3.1 INTRODUCCIÓN.

Durante la realización del presente proyecto, el proyectista se ve sometido a una serie de condiciones de trabajo que pueden influir tanto en el rendimiento como en la propia salud de el mismo. Por ello, es importante poner los medios necesarios para la prevención de riesgos laborales. En este caso, debido a la labor que se realiza, estos riesgos pueden provocar estrés, fatiga, malestar, etc.

La normativa que se ajusta al tipo de trabajo que se va a realizar está recogida en el Real Decreto 488/1997, del 14 de Abril, sobre disposiciones mínimas de seguridad y salud relativas al trabajo con equipos que incluyen pantallas de visualización. Para comprobar que se debe seguir dicha normativa, en el artículo 2 de este Real Decreto se observa la siguiente definición: "Puesto de trabajo: el constituido por un equipo con pantalla de visualización provisto, en su caso, de un teclado o dispositivo de adquisición de datos, de un programa para la interconexión persona/máquina, de accesorios ofimáticos y de un asiento y mesa o superficie de trabajo, así como el entorno laboral inmediato".

A continuación, se exponen las distintas variables para prever el tipo de riesgo a los que el trabajador puede enfrentarse.

- Tiempo de trabajo con la pantalla de visualización.
- Tiempo de atención requerida ante la pantalla, que puede ser continua o discontinua.
- Exigencia y grado de complejidad de la tarea realizada ante la pantalla.
- Necesidad de obtener una información de una manera muy rápida.

Por otra parte, los riesgos que pueden surgir en este tipo de trabajos son:

- Riesgo por contacto eléctrico.
- Riesgo de ergonomía:
    - Fatiga física.
    - Fatiga visual (pesadez, picores, …).
    -Fatiga menta (ansiedad, insomnio, …).
- Higiene Industrial:
    - Iluminación.
    - Ruido.
    - Temperatura y humedad.

## 3.2 CONDICIONES NECESARIAS EN EL PUESTO DE TRABAJO

Los entornos de trabajo en los cuales se lleva a cabo la actividad deben atenerse a lo establecido en el Real Decreto 486/1997 del 14 de Abril, por el que se establecen las disposiciones mínimas de seguridad y salud en los lugares de trabajo, además de los establecidos en el Real Decreto 488/1997 para puestos de trabajo con pantallas de visualización. Estas condiciones se pueden agrupar en los apartados que se detallan a continuación:

### 3.2.1 Medidas de emergencia. Vías y salidas de evacuación.

La empresa debe haber adoptado las medidas de emergencias necesarias en las que se prevean las vías y salidas de evacuación en caso de que se declare una emergencia. Estas medidas deben ser conocidas por todos los trabajadores de dicha empresa.

Existen numerosas normativas en este aspecto, la mayoría de estas normativas están enfocadas a la fase constructiva de la instalación. Para este pliego de condiciones se presupone que estas medidas fueron debidamente validadas en la construcción de la instalación.

### 3.2.2 Condiciones de protección contra incendios.

Las instalaciones de protección contra incendios deberán de estar proyectadas, implantadas y mantenidas por empresas debidamente autorizadas por el organismo competente.

Los dispositivos de detección, alarma y extinción deberán estar de acuerdo con la normativa de aplicación, así como las señales deberán ser visibles en todo momento.

### 3.2.3 Instalación eléctrica.

La instalación eléctrica debe estar proyectada, controlada y puesta en funcionamiento por una empresa debidamente autorizada por el Ministerio de Industria y Energía o la Conserjería de Industria o similar de la Comunidad Autónoma donde esté ubicado el lugar de trabajo.

La instalación eléctrica deberá prever que, debido al uso que va a hacerse de la energía eléctrica, no se puedan causar incendios y/o explosiones, los trabajadores estén protegidos frente a contactos directos o indirectos. Para ello se atiende a lo establecido en los Reglamentos de Baja y Alta tensión en vigor sobre tensiones de seguridad en los sistemas de protección, conductores, etc.

Desde la perspectiva de la seguridad eléctrica, lo establecido para los puestos de trabajo equipados con pantallas de visualización es lo siguiente:

- Cumplir los requerimientos de la directiva sobre emisiones electromagnéticas, que exige reducir a niveles insignificantes toda radiación electromagnética producida, desde el punto de vista de la protección, seguridad y salud de los trabajadores.
- Mantener separados los cables eléctricos de los telefónicos.
- Garantizar el adecuado mantenimiento de los cables y las conexiones.
- Facilitar el mantenimiento y acceso a los cables sin interrupción de las actividades del trabajo.

- Emplear longitudes de cable suficientemente grandes como para permitir futuros cambios. Disponer de tal modo los cables que, al tiempo su mantenimiento sea el correcto, fuera de superficies sometidas a condiciones adversas o donde puedan ser pisadas.

### 3.2.4 Condiciones termo-higiénicas.

Para cada trabajador la situación de bienestar o confort térmico es distinta, si bien depende de factores como:

- Temperatura, humedad y velocidad del aire.
- Temperatura de paredes, suelos y objetos.
- La vestimenta.
- La actividad a desarrollar.

El Real Decreto 488/1997 sobre puestos de trabajo con pantallas de visualización establece ciertos valores de estas magnitudes para dicho puesto de trabajo:

- La temperatura operativa de confort debe mantenerse dentro de los siguientes rangos:
  Para verano de 23° a 26°C.
  Para invierno de 20° a 24°C.
- La humedad relativa del aire debe mantenerse entre el 45 y 65%, con ello se puede prevenir la sequedad de ojos y mucosa.
- Según el Real Decreto 486/1997 que establece las disposiciones mínimas de seguridad y salud en los lugares de trabajo, la velocidad del aire no debe exceder los 0.25 m/s para trabajos sedentarios en ambientes no calurosos.

### 3.2.5 Condiciones de iluminación.

La iluminación puede ser natural o artificial, siendo la más recomendable la natural. No obstante, la intensidad de la iluminación natural varía con la hora del día y con las estaciones del año, por lo que generalmente debe complementarse con iluminación artificial, que puede ser localizada o general.

Habitualmente se trata de iluminación general, complementada con localizada. Para ambos tipos de iluminación se debe tener en cuenta que no produzcan deslumbramientos, ni un excesivo contraste entre zonas iluminadas y de sombra, tal y como se establece en la normativa específica de estos puestos de trabajo. Para controlar dicho deslumbramiento en el Real Decreto 488/1997 establece no sobrepasar el límite de los 500 $Cd/m^2$ para las vistas bajo un ángulo inferior a 45° sobre el plano horizontal, siendo recomendable no sobrepasar los 200 $Cd/m^2$.

Los puestos de trabajo con pantallas de visualización deben tener una iluminación general y en caso de utilizar una fuente de iluminación individual complementaria, esta no puede ser usada cerca de la pantalla si produce cierto deslumbramiento directo o reflexiones. Además, se recomienda que el puesto de trabajo se oriente adecuadamente respecto a las ventanas, para evitar los reflejos que originarían las pantallas y el deslumbramiento que sufriría el usuario.

Estas medidas se pueden complementar con el uso de cortinas o persianas que amortigüen la luz, o mediante mamparas en los locales que dispongan de ventanas en más de una pared.

Para evitar deslumbramientos por reflejos, la superficie del mobiliario y de los elementos de trabajo deberán ser de aspecto mate.

Los niveles de iluminación deberán ser suficientes para las tareas que se realicen en dicho puesto de trabajo, pero no deben alcanzar valores que reduzcan el contraste de la pantalla por debajo de lo tolerable.

### 3.2.6 Ergonomía.

Los riesgos que se han presentado con anterioridad precisan de medidas que se recogen en el Real Decreto 488/1997 sobre pantallas de visualización. El diseño del puesto de trabajo está directamente relacionado con los problemas posturales. Si se tienen en cuenta que el trabajo con pantallas de visualización se caracteriza por adoptar posturas estáticas prolongadas, se puede deducir que los efectos de estas posturas se agravan cuando el puesto de trabajo no está diseñado correctamente mediante medidas adecuadas.

Para resolver este problema se ha fabricado un mobiliario que se adecua a los usuarios dentro de unos estándares establecidos. Además, el mobiliario y las superficies de trabajo deberán carecer de esquinas y de aristas agudas, un color preferiblemente neutro y las superficies susceptibles de entrar en contacto con el usuario no deben ser buenas conductoras de calor, con el fin de evitar la transmisión con la piel del usuario.

Para el diseño del puesto de trabajo se tienen en cuanta los siguientes aspectos:

- **Asientos.**
  Los asientos del puesto de trabajo deberán de poder ajustar la altura de dicho asiento al rango necesario del usuario. El respaldo debe de disponer de una suave prominencia para dar apoyo a la zona lumbar. La inclinación del asiento debe ser ajustable, al igual que la profundidad del asiento para la comodidad del usuario. Todos los mecanismos de ajuste deben ser fácilmente manejables desde la posición sentada. Se recomienda la utilización de sillas dotadas de 5 apoyos sobre el suelo y, además, debería incluir ruedas, especialmente para superficies de trabajo muy amplias. La resistencia de las ruedas al iniciar un movimiento debe evitar desplazamientos involuntarios en suelos lisos y con actividades de tecleo intensivo.

- **Mesa y espacio de trabajo.**
  La superficie de la mesa deberá de ser de dimensiones suficientes, permitir una colocación flexible de la pantalla, del teclado, de los documentos y del material de accesorio y además ser poco reflectante.
  El espacio de trabajo deberá ser suficiente como para permitir una posición cómoda a los trabajadores.
  El soporte de los documentos deberá ser estable y regulable y estará colocado de tal modo que se reduzcan al mínimo los movimientos incómodos de la cabeza y los ojos.

- **Postura del usuario.**
  La postura que se debe de tomar es: los muslos horizontales, las piernas verticales, los brazos verticales y los antebrazos horizontales formando un ángulo recto desde el codo, las manos relajadas sin extensión ni desviación lateral, guardando al máximo la recta del antebrazo para evitar el síndrome del túnel carpiano, por eso es recomendable utilizar reposa manos, donde descansan las muñecas. La columna vertebral recta, las plantas de los pies formando un ángulo recto con respecto las piernas. La línea de visión paralela al plano horizontal sin torsión del tronco. Ángulo de la línea de visión menor de 60° bajo horizontal.

- **Colocación de la pantalla.**
  La distancia de la pantalla a los ojos del usuario no debe ser menor de 40 cm, y la distancia óptima para el punto de vista del confort visual, debe ser entre 45 y 75 cm.
  La pantalla deberá colocarse de manera que su área útil pueda ser visible para ángulos comprendidos entre la línea de visión horizontal y la trazada a 60° bajo la horizontal. En el plano horizontal la pantalla deberá estar colocada dentro del ángulo de 120° del campo de visión del usuario, si bien es aconsejable situarla dentro de un ángulo de 70°.

- **Pantalla.**
  Los caracteres de la pantalla deberán estar bien definidos y configurados de forma clara, y tener una dimensión suficiente.
  Las imágenes de la pantalla deberán ser estables, es decir sin fenómenos de destello, centelleos u otras formas de inestabilidad.
  Los usuarios de terminales con pantallas deben poder ajustar fácilmente la luminosidad y el contraste entre los caracteres y el fondo de pantalla.
  La pantalla debe ser orientable e inclinable a voluntad del usuario y, además, no deberá tener reflejos ni reverberaciones que ocasionen molestia al usuario.

- **Teclado.**
  El teclado deberá ser independiente de la pantalla y con inclinación, para permitir que el usuario pueda adoptar una postura cómoda que no provoque cansancio de brazos y manos. Además, deberá haber espacio suficiente delante del teclado para que el usuario pueda apoyar los brazos y las manos. La superficies del teclado deberá ser mate para evitar reflejos.
  La disposición del teclado y las características de las teclas deben tender a facilitar su utilización, al igual que los símbolos de las teclas deben resaltar y ser legibles desde la posición normal de trabajo.

### 3.2.7 Ruido.

El Real Decreto 286/2006 del 10 de Marzo, sobre la protección de la salud y la seguridad de los trabajadores contra los riegos relacionados con la exposición al ruido, establece que el empresario deberá evaluar la exposición del ruido a los trabajadores, con el objetivo de determinar si estos superan los límites establecidos en esta norma, para poder aplicar las medidas que procedan en el origen, en el medio o en el receptor así como los reconocimientos médicos específicos para las personas expuestas al ruido con la periodicidad y características recogidas en el Real Decreto.

Para entornos en los cuales los niveles sonoros sean altos o medios deberán tomarse las medidas necesarias según el Real Decreto 286/2006 para reducir estos niveles a lo más bajo posible. Para ello se deben utilizar equipos con una mínima emisión sonora y optimizar la acústica del lugar. La Directiva establece que, para tareas difíciles y complejas el nivel sonoro continuo equivalente no de sobrepasar de 55 db(A).

Es necesario reducir estos niveles de ruido ya que, pueden molestar y perturbar la atención de los operarios de equipos informáticos, incluso pueden llegar a ser insoportables para determinadas actividades. Sin olvidar que los usuarios de ordenadores en ocasiones necesitan concentración para realizar su trabajo, por lo que es más susceptible de poder ser molestado por ruidos o medio ambiente incómodo.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

**TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES**

# DESARROLLO DE UNA PLATAFORMA SOFTWARE DE BAJO COSTE IMPLEMENTADA EN MATLAB PARA LA AUTOMATIZACIÓN DE PLANTAS INDUSTRIALES VIRTUALES

## Presupuesto

**Curso Académico: 2018-19**

# PRESUPUESTO

## 1. INTRODUCCIÓN.

En este apartado del documento se recogen los costes que se estiman para el desarrollo del proyecto. Entre estos gastos se encuentran las licencias de los programas informáticos como los gastos del equipo y la mano de obra.

## 2. CÁLCULO DEL PRESUPUESTO.

### 2.1 Mano de obra.

En primer lugar, se tomará como referencia el salario medio de un ingeniero industrial junior que corresponde a unos 21.000€ anuales. Si se tiene en cuenta una cifra de 223 días laborales al año y una jornada completa de 8 horas se puede establecer un coste de unos 12€ por hora trabajada, además el proyecto tiene una duración de 300 horas, por lo que el coste de la mano de obra será:

$$12\,\frac{€}{hora} * 300\ horas = 3600€$$

Las 300 horas empleadas han sido empleadas en:

- 100 horas de aprendizaje y desarrollo de la plataforma.
- 80 horas de aprendizaje y creación de la interfaz gráfica.
- 90 horas de aprendizaje y configuración escena Factory I/O.
- 30 horas de aprendizaje y configuración servidor OPC.

El coste en cuanto a mano de obra del proyecto asciende a TRES MIL SEISCIENTOS EUROS.

### 2.2 Materiales.

En este apartado se detallan los costes de los materiales y programas informáticos utilizados para la realización del proyecto, como se comentó en capítulos anteriores, para llevar a cabo el proyecto se ha utilizado un ordenador portátil y los distintos programas comentados anteriormente. A continuación, se exponen los precios del ordenador como de las licencias de los programas:

- Ordenador portátil: 700€
    - Procesador Intel Core i7– 4500U de 1.8 GHz.
    - Memoria RAM 4 GB.

  - Unidad de almacenamiento, disco duro, de 750 GB.
  - Tarjeta gráfica NVIDIA GeForce 820M.
- Software Matlab (licencia 1 año) 800€
- Software Factory I/O (licencia 1 año) 144€
- Software KEPServerEX (licencia 1 año) 452€

El precio de estos materiales no debe contabilizarse como tal en el presupuesto del proyecto, ya que no se han adquirido exclusivamente para realizar este proyecto ni se han desechado tras la finalización de este, por lo que deberá fijarse un periodo de amortización tanto del equipo como de los programas informáticos y contabilizar el coste que han supuesto durante el periodo que se ha desarrollado el proyecto.

Para el ordenador se ha fijado un periodo de amortización de cuatro años, que es lo más común en equipos informáticos. Teniendo en cuenta esto, se calculará el coste horario del equipo informático y se multiplicará este coste por las horas de utilización del equipo.

El coste horario del equipo será:

$$\frac{700€}{4\ años * \left(223\frac{días}{año} * 8\frac{horas}{día}\right)} = 0.1\frac{€}{hora}$$

Si multiplicamos el coste horario del equipo por las 300 horas que han sido utilizadas para la realización del proyecto, el coste será:

$$0.1\frac{€}{hora} * 300\ horas = 30€$$

El coste del ordenador teniendo en cuenta la amortización y las horas de trabajo, asciende a TREINTA EUROS.

Por otra parte, queda por contabilizar los costes debido a la adquisición de los programas informáticos utilizados para el desarrollo de la plataforma, que como han sido comentado anteriormente son Matlab, Factory I/O y KEPServerEX.

En primer lugar, se lleva a cabo la amortización de Matlab para ello se saca el coste horario del software:

$$\frac{800€}{1\ año * \left(223\frac{días}{año} * 8\frac{horas}{día}\right)} = 0.45\frac{€}{hora}$$

Una vez se ha calculado el coste horario del software se tiene en cuenta el número de horas con las que se ha trabajado, el número de horas total del proyecto son 300 horas, de las cuales se estima que con Matlab se ha trabajado un 60% de las horas, es decir 180 horas.

$$0.45\frac{€}{hora} * 180\ horas = 81\ €$$

En segundo lugar, se realiza la amortización para el software Factory I/O, para el cual se saca el coste horario:

$$\frac{144€}{1\ año * \left(223\frac{días}{año} * 8\frac{horas}{día}\right)} = 0.08\frac{€}{hora}$$

Para el caso de Factory I/O se estima que el número de horas empleadas en dicho software es del 30%, es decir 90 horas y por lo tanto el gasto sería:

$$0.08\frac{€}{hora} * 90\ horas = 7.20€$$

Y, por último, se realiza la amortización de KEPServeEX:

$$\frac{452€}{1\ año * (223\frac{días}{año} * 8\frac{horas}{día})} = 0.25\frac{€}{hora}$$

En este caso la estimación de horas empleadas es el 10% de las horas totales, por lo que el gasto de este software será:

$$0.25\frac{€}{hora} * 30\ horas = 7.50\ €$$

La suma de los tres software dará un coste de:

$$81€ + 7.20€ + 7.50€ = 95.70€$$

Por lo tanto, el gasto en cuanto a programas informáticos del proyecto, ascienden a NOVENTA Y CINCO EUROS CON SETENTA CÉNTIMOS.

**3. RESUMEN DEL PRESUPUESTO.**

En los apartados anteriores se ha visto todos los gastos que este proyecto lleva asociado, por lo que en este apartado se va a realizar un resumen de todas las partes y se dará el valor final del presupuesto de este proyecto:

| Descripción | Importe (€) |
|---|---|
| 1. Coste mano de obra | 3600€ |
| 2. Coste del equipo (ordenador portátil) | 30€ |
| 3. Coste programas informáticos | 95.70€ |
| Presupuesto de Ejecución Material | 3725.70€ |
| Gastos Generales 13% | 484.34€ |
| Beneficio Industrial 6% | 223.54€ |
| Presupuesto Total | 4433.58€ |
| IVA 21% | 931.05€ |
| Presupuesto de Ejecución por Contrata | 5364.63€ |

El presente presupuesto asciende a la cantidad de:

CINCO MIL TRESCIENTOS SESENTA Y CUATRO CON SESENTA Y TRES CÉNTIMOS.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

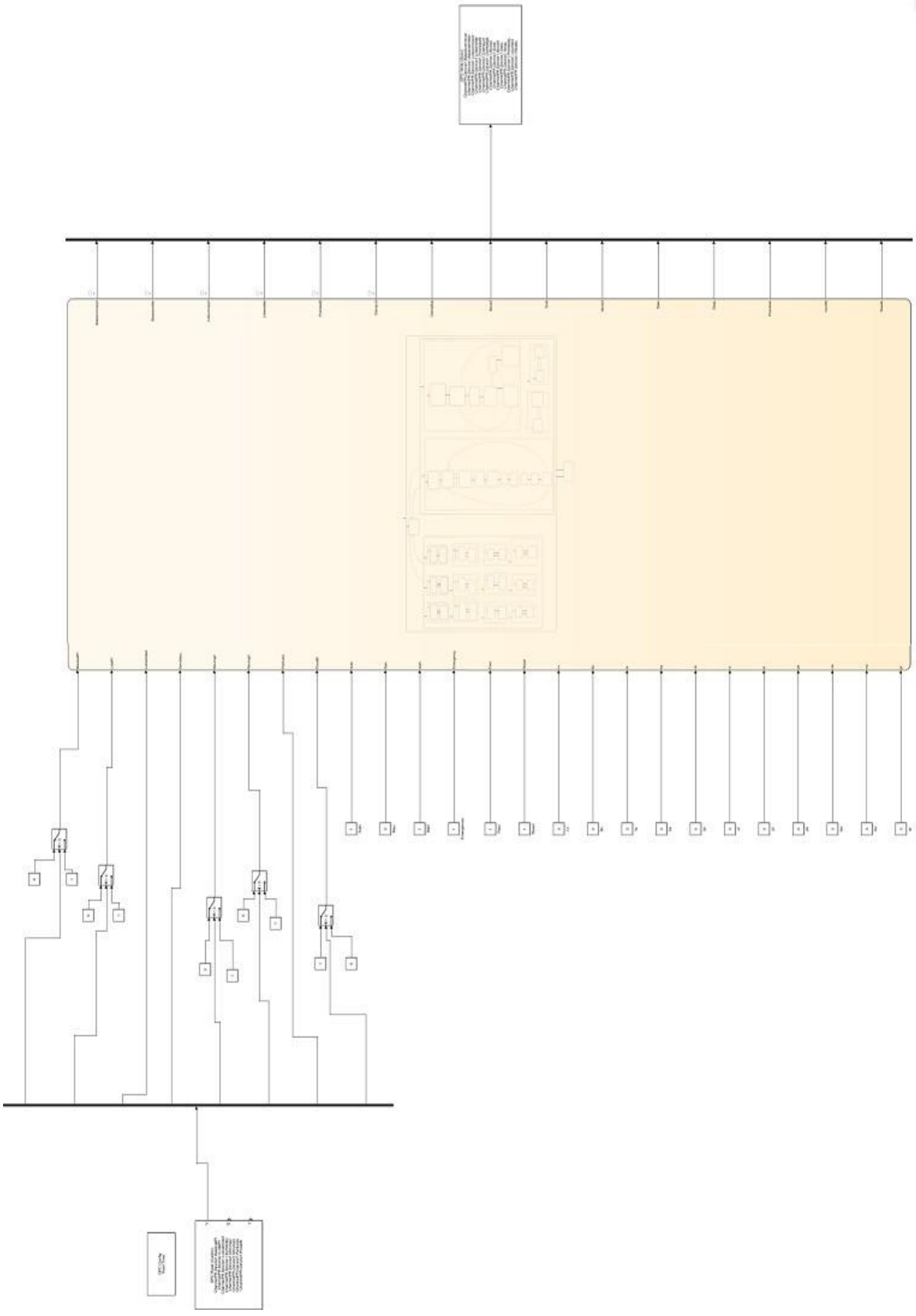
**TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES**

# DESARROLLO DE UNA PLATAFORMA SOFTWARE DE BAJO COSTE IMPLEMENTADA EN MATLAB PARA LA AUTOMATIZACIÓN DE PLANTAS INDUSTRIALES VIRTUALES

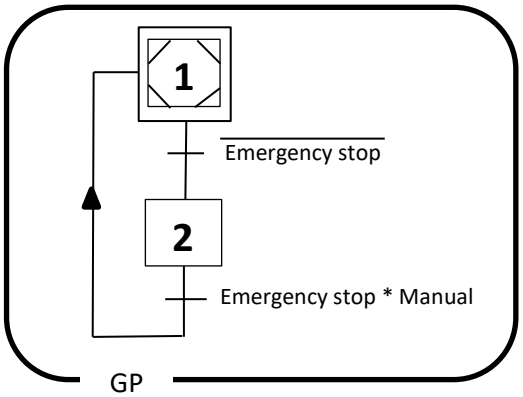
# ANEXOS

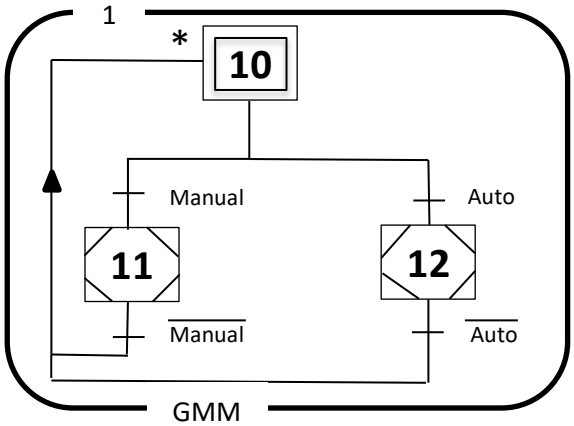
**Curso Académico: 2018-19**

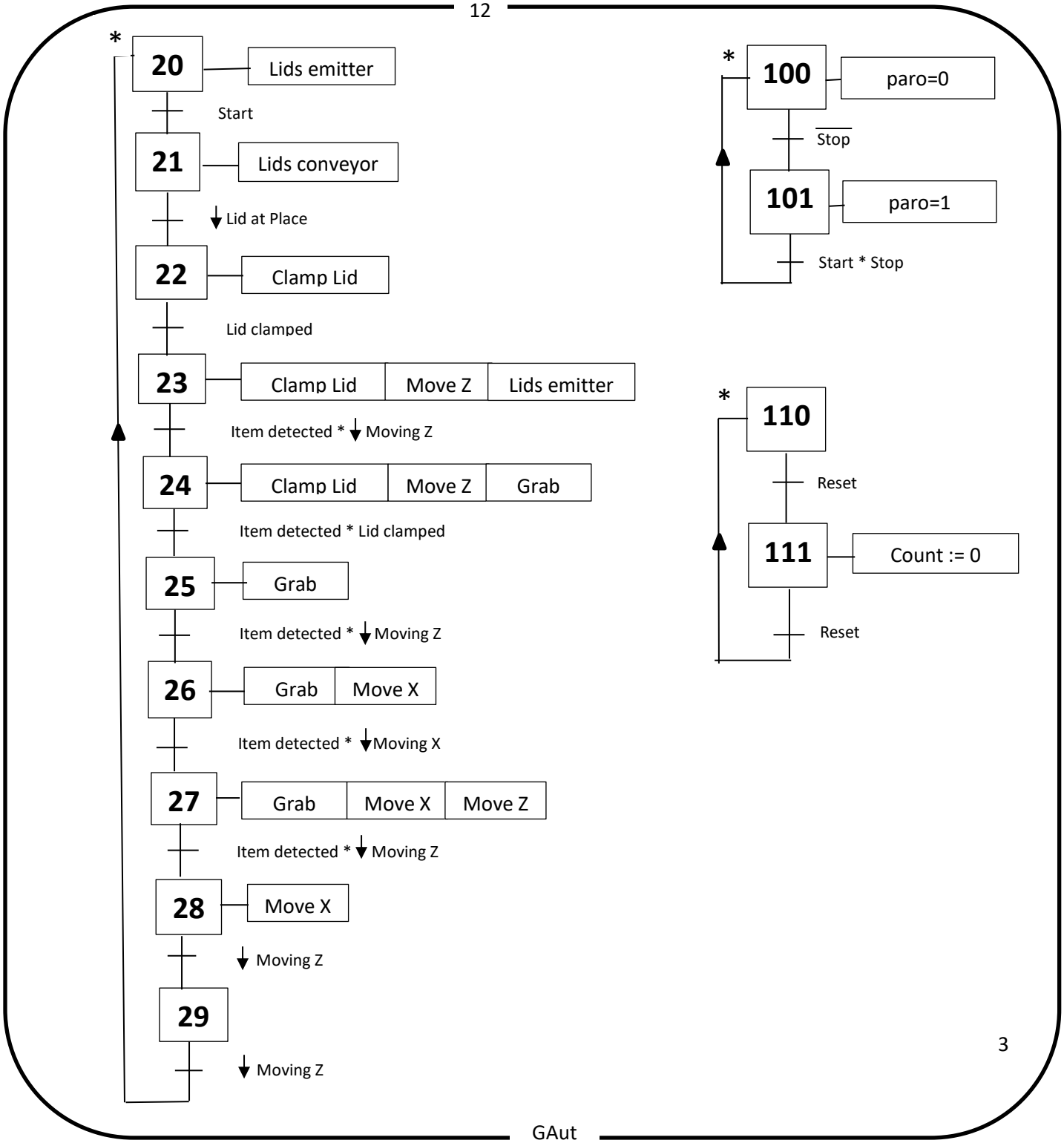
**ANEXO I. PLATAFORMA SOFTWARE**

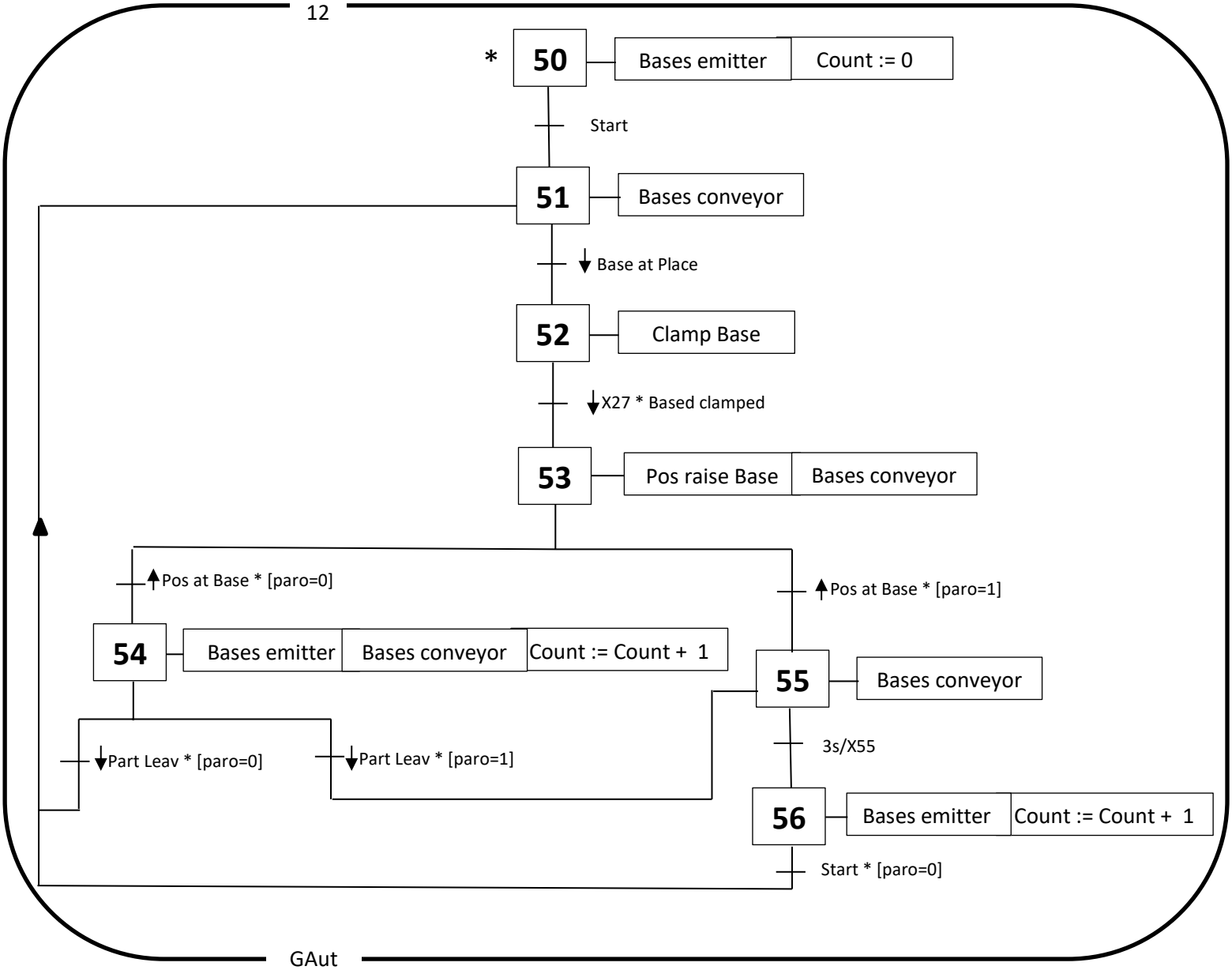


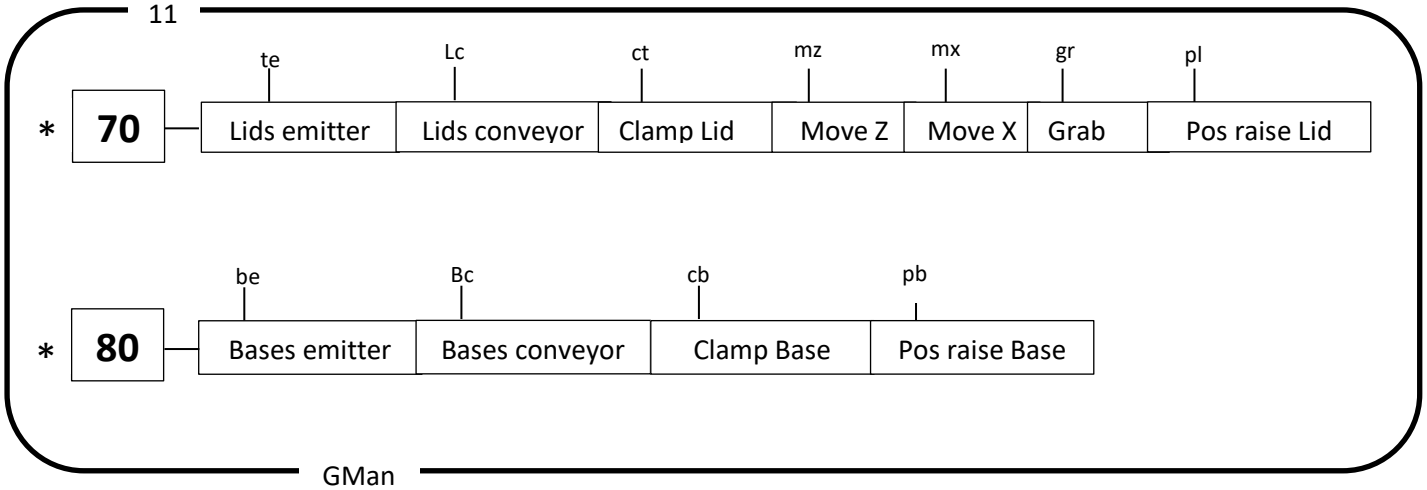
**ANEXO II. GRAFCET IMPLEMENTADO EN LA PLATAFORMA**



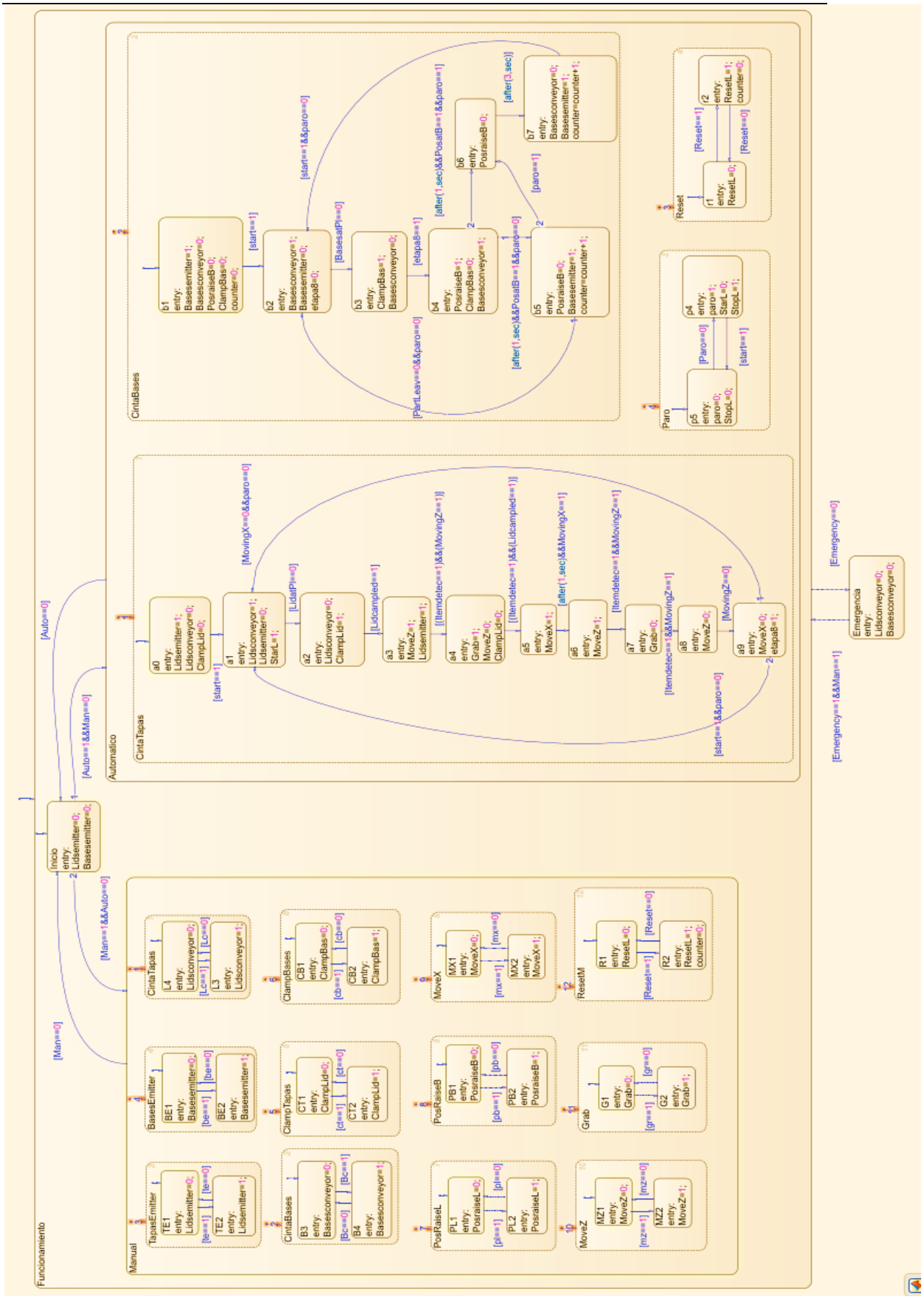






El grafcet parcial GAut se encuentra dividido en dos hojas al no caber en una sola.


Funcionamiento
inicio
entry:
Lidsemitter=0;
Basesemitter=0;
[Auto==0]
[Auto==1&&Man==1]
[Man==1&&Auto==0]
[Man==0]
Automatico
CintaTapas
a0
entry:
Lidsemitter=1;
Lidsconveyor=0;
ClampLid=0;
a1
entry:
Lidsconveyor=1;
Lidsemitter=0;
StarL=1;
a2
entry:
Lidsconveyor=0;
ClampLid=1;
a3
entry:
MoveZ=1;
Lidsemitter=1;
a4
entry:
Grab=1;
MoveZ=0;
ClampLid=0;
a5
entry:
MoveX=1;
a6
entry:
MoveZ=1;
a7
entry:
Grab=0;
a8
entry:
MoveZ=0;
a9
entry:
MoveX=0;
etapa8=1;
[start==1]
[LidatPli==0]
[Lidcampled==1]
[!Itemdetec==1]&&(MovingZ==1)]
[!Itemdetec==1]&&(Lidcampled==1)]
[after(1,sec)&&MovingX==1]
[!Itemdetec==1&&MovingZ==1]
[!Itemdetec==1&&!MovingZ==1]
[MovingZ==0]
[MovingX==0&&paro==0]
[start==1&&paro==0]
CintaBases
b1
entry:
Basesemitter=1;
Basesconveyor=0;
PosraiseB=0;
ClampBas=0;
counter=0;
b2
entry:
Basesconveyor=1;
Basesemitter=0;
etapa8=0;
b3
entry:
ClampBas=1;
Basesconveyor=0;
b4
entry:
PosraiseB=1;
ClampBas=1;
Basesconveyor=1;
b5
entry:
PosraiseB=0;
Basesemitter=1;
counter=counter+1;
b6
entry:
PosraiseB=0;
b7
entry:
Basesconveyor=0;
Basesemitter=1;
counter=counter+1;
[BaseatPli==0]
[etapaB==1]
[after(1,sec)&&PosatB==1&&paro==1]
[after(3,sec)]
[PartLeav==0&&paro==0]
[after(1,sec)&&PosatB==1&&paro==0]
[paro==1]
Paro
p5
entry:
paro=0;
StopL=0;
p4
entry:
paro=1;
StarL=0;
StopL=1;
[Paro==0]
[start==1]
Reset
r1
entry:
ResetL=0;
r2
entry:
ResetL=1;
counter=0;
[Reset==1]
[Reset==0]
Emergencia
entry:
Lidsconveyor=0;
Basesconveyor=0;
[Emergency==0]
[Emergency==1&&Man==1]
Manual
TapasEmitter
TE1
entry:
Lidsemitter=0;
TE2
entry:
Lidsemitter=1;
[te==1]
[te==0]
CintaTapas
L4
entry:
Lidsconveyor=0;
L3
entry:
Lidsconveyor=1;
[Lc==1]
[Lc==0]
BasesEmitter
BE1
entry:
Basesemitter=0;
BE2
entry:
Basesemitter=1;
[be==1]
[be==0]
CintaBases
B3
entry:
Basesconveyor=0;
B4
entry:
Basesconveyor=1;
[Bc==0]
[Bc==1]
ClampBases
CB1
entry:
ClampBas=0;
CB2
entry:
ClampBas=1;
[cb==1]
[cb==0]
ClampTapas
CT1
entry:
ClampLid=0;
CT2
entry:
ClampLid=1;
[ct==1]
[ct==0]
PosRaiseL
PL1
entry:
PosraiseL=0;
PL2
entry:
PosraiseL=1;
[pl==1]
[pl==0]
PosRaiseB
PB1
entry:
PosraiseB=0;
PB2
entry:
PosraiseB=1;
[pb==1]
[pb==0]
MoveZ
MZ1
entry:
MoveZ=0;
MZ2
entry:
MoveZ=1;
[mz==1]
[mz==0]
MoveX
MX1
entry:
MoveX=0;
MX2
entry:
MoveX=1;
[mx==1]
[mx==0]
Grab
G1
entry:
Grab=0;
G2
entry:
Grab=1;
[gr==1]
[gr==0]
ReseM
R1
entry:
ResetL=0;
R2
entry:
ResetL=1;
counter=0;
[Reset==1]
[Reset==0]

**ANEXO III. CÓDIGO INTERFAZ DE USUARIO**

```
classdef appProcesoBueno < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        PanelCintaTapas                 matlab.ui.container.Panel
        LidEmitterSwitchLabel           matlab.ui.control.Label
        LidEmitterSwitch                matlab.ui.control.Switch
        PosRaiseLidSwitch_2Label        matlab.ui.control.Label
        PosRaiseLidSwitch_2             matlab.ui.control.Switch
        ClampLidSwitch_2Label           matlab.ui.control.Label
        ClampLidSwitch_2                matlab.ui.control.Switch
        LidConveyorSwitch_2Label        matlab.ui.control.Label
        LidConveyorSwitch_2             matlab.ui.control.Switch
        PanelCintaBases                 matlab.ui.container.Panel
        BasesEmitterSwitchLabel         matlab.ui.control.Label
        BasesEmitterSwitch              matlab.ui.control.Switch
        PosRaiseBasesSwitch_2Label      matlab.ui.control.Label
        PosRaiseBasesSwitch_2           matlab.ui.control.Switch
        ClampBasesSwitch_2Label         matlab.ui.control.Label
        ClampBasesSwitch_2              matlab.ui.control.Switch
        BasesConveyorSwitch_2Label      matlab.ui.control.Label
        BasesConveyorSwitch_2           matlab.ui.control.Switch
        PanelBrazoRobot                 matlab.ui.container.Panel
        MoveXSwitchLabel                matlab.ui.control.Label
        MoveXSwitch                     matlab.ui.control.Switch
        MoveZSwitchLabel                matlab.ui.control.Label
        MoveZSwitch                     matlab.ui.control.Switch
        GrabSwitchLabel                 matlab.ui.control.Label
        GrabSwitch                      matlab.ui.control.Switch
        ControladorPanel                matlab.ui.container.Panel
        MododefuncionamientoKnobLabel   matlab.ui.control.Label
        MododefuncionamientoKnob        matlab.ui.control.DiscreteKnob
        StartButton                     matlab.ui.control.Button
        StopButton                      matlab.ui.control.Button
        Label_3                         matlab.ui.control.Label
        Lamp_13                         matlab.ui.control.Lamp
        Lamp_14                         matlab.ui.control.Lamp
        Lamp_15                         matlab.ui.control.Lamp
        EmergenciaButton                matlab.ui.control.StateButton
        ResetButton                     matlab.ui.control.Button
        Panel                           matlab.ui.container.Panel
        LidEmitterLampLabel             matlab.ui.control.Label
        LidEmitterLamp                  matlab.ui.control.Lamp
        LidConveyorLampLabel            matlab.ui.control.Label
        LidConveyorLamp                 matlab.ui.control.Lamp
        ClampLidLampLabel               matlab.ui.control.Label
        ClampLidLamp                    matlab.ui.control.Lamp
        LidClampedLampLabel             matlab.ui.control.Label
        LidClampedLamp                  matlab.ui.control.Lamp
        LidatPlaceLampLabel             matlab.ui.control.Label
        LidatPlaceLamp                  matlab.ui.control.Lamp
```

```
        Panel_2                         matlab.ui.container.Panel
        BasesEmitterLampLabel           matlab.ui.control.Label
        BasesEmitterLamp                matlab.ui.control.Lamp
        BaseConveyorLampLabel           matlab.ui.control.Label
        BaseConveyorLamp                matlab.ui.control.Lamp
        ClampBaseLampLabel              matlab.ui.control.Label
        ClampBaseLamp                   matlab.ui.control.Lamp
        BaseClampedLampLabel            matlab.ui.control.Label
        BaseClampedLamp                 matlab.ui.control.Lamp
        BaseatPlaceLampLabel            matlab.ui.control.Label
        BaseatPlaceLamp                 matlab.ui.control.Lamp
        PartLeavingLampLabel            matlab.ui.control.Label
        PartLeavingLamp                 matlab.ui.control.Lamp
        Panel_3                         matlab.ui.container.Panel
        MoveXLampLabel                  matlab.ui.control.Label
        MoveXLamp                       matlab.ui.control.Lamp
        MoveZLampLabel                  matlab.ui.control.Label
        MoveZLamp                       matlab.ui.control.Lamp
        MovingXLampLabel                matlab.ui.control.Label
        MovingXLamp                     matlab.ui.control.Lamp
        MovingZLampLabel                matlab.ui.control.Label
        MovingZLamp                     matlab.ui.control.Lamp
        GrabLampLabel                   matlab.ui.control.Label
        GrabLamp                        matlab.ui.control.Lamp
        PosraiseLidLampLabel            matlab.ui.control.Label
        PosraiseLidLamp                 matlab.ui.control.Lamp
        PosraiseBaseLampLabel           matlab.ui.control.Label
        PosraiseBaseLamp                matlab.ui.control.Lamp
        PosatBaseLampLabel              matlab.ui.control.Label
        PosatBaseLamp                   matlab.ui.control.Lamp
        PosatLidLampLabel               matlab.ui.control.Label
        PosatLidLamp                    matlab.ui.control.Lamp
        ItemdetectedLampLabel           matlab.ui.control.Label
        ItemdetectedLamp                matlab.ui.control.Lamp
        ActuadoresLampLabel             matlab.ui.control.Label
        ActuadoresLamp                  matlab.ui.control.Lamp
        SensoresLampLabel               matlab.ui.control.Label
        SensoresLamp                    matlab.ui.control.Lamp
    end
    properties (Access = private)
        timerObj %timer
        itm0
        itm1
        itm2
        itm3
        itm4
        itm5
        itm6
        itm7
        itm8
        itm9
        itm10
        itm11
```

```
        itm12
        itm13
        itm14
        itm15
        itm16
        itm17
        itm18
        itm19
        itm20


    end
    methods (Access = private)


        function getModelValues(app)

            Lidsemitter = read(app.itm0) %Lectura de Salidas(Actuadores)
            Basesemitter = read(app.itm1)
            Lidsconveyor = read(app.itm2)
            Basesconveyor = read(app.itm3)
            PosraiseB = read(app.itm4)
            PosraiseL = read(app.itm5)
            ClampLid = read(app.itm6)
            ClampBas = read(app.itm7)
            MoveX = read(app.itm8)
            MoveZ = read(app.itm9)
            Grab = read(app.itm10)

            LidatPl = read(app.itm11) %Lectura de Entradas(Sensores)
            BaseatPl = read(app.itm12)
            Lidclamped = read(app.itm13)
            Baseclamped = read(app.itm14)
            PosatL = read(app.itm15)
            PosatB = read(app.itm16)
            MovingX = read(app.itm17)
            MovingZ = read(app.itm18)
            Itemdetec = read(app.itm19)
            PartLeav = read(app.itm20)


            app.LidEmitterLamp.Enable=(Lidsemitter.Value)  %Encendido de
bombillas de Actuadores
            app.BasesEmitterLamp.Enable=(Basesemitter.Value)
            app.LidConveyorLamp.Enable=(Lidsconveyor.Value)
            app.BaseConveyorLamp.Enable=(Basesconveyor.Value)
            app.PosraiseBaseLamp.Enable=(PosraiseB.Value)
            app.PosraiseLidLamp.Enable=(PosraiseL.Value)
            app.ClampLidLamp.Enable=(ClampLid.Value)
            app.ClampBaseLamp.Enable=(ClampBas.Value)
            app.MoveXLamp.Enable=(MoveX.Value)
            app.MoveZLamp.Enable=(MoveZ.Value)
            app.GrabLamp.Enable=(Grab.Value)
```

```
            app.LidatPlaceLamp.Enable=(LidatPl.Value) %Encendido de
bombillas de Sensores
            app.BaseatPlaceLamp.Enable=(BaseatPl.Value)
            app.LidClampedLamp.Enable=(Lidclamped.Value)
            app.BaseClampedLamp.Enable=(Baseclamped.Value)
            app.PosatLidLamp.Enable=(PosatL.Value)
            app.PosatBaseLamp.Enable=(PosatB.Value)
            app.MovingXLamp.Enable=(MovingX.Value)
            app.MovingZLamp.Enable=(MovingZ.Value)
            app.ItemdetectedLamp.Enable=(Itemdetec.Value)
            app.PartLeavingLamp.Enable=(PartLeav.Value)

        end



    end
    methods (Access = private)
        % Code that executes after component creation
        function startupFcn(app)
             da = opcda('localhost','Kepware.KEPServerEX.V5');
             connect(da);
             grp = addgroup(da);
             app.itm0 = additem(grp,'ChannelPR.Device1.Lidsemitter');
%Actuadores
             app.itm1 = additem(grp,'ChannelPR.Device1.Basesemitter');
             app.itm2 = additem(grp,'ChannelPR.Device1.Lidsconveyor');
             app.itm3 = additem(grp,'ChannelPR.Device1.Basesconveyor');
             app.itm4 = additem(grp,'ChannelPR.Device1.PosraiseB');
             app.itm5 = additem(grp,'ChannelPR.Device1.PosraiseL');
             app.itm6 = additem(grp,'ChannelPR.Device1.ClampLid');
             app.itm7 = additem(grp,'ChannelPR.Device1.ClampBas');
             app.itm8 = additem(grp,'ChannelPR.Device1.MoveX');
             app.itm9 = additem(grp,'ChannelPR.Device1.MoveZ');
             app.itm10 = additem(grp,'ChannelPR.Device1.Grab');

             app.itm11 = additem(grp,'ChannelPR.Device1.LidatPl');
%Sensores
             app.itm12 = additem(grp,'ChannelPR.Device1.BasesatPl');
             app.itm13 = additem(grp,'ChannelPR.Device1.Lidcampled');
             app.itm14 = additem(grp,'ChannelPR.Device1.Baseclamped');
             app.itm15 = additem(grp,'ChannelPR.Device1.PosatL');
             app.itm16 = additem(grp,'ChannelPR.Device1.PosatB');
             app.itm17 = additem(grp,'ChannelPR.Device1.MovingX');
             app.itm18 = additem(grp,'ChannelPR.Device1.MovingZ');
             app.itm19 = additem(grp,'ChannelPR.Device1.Itemdetec');
             app.itm20 = additem(grp,'ChannelPR.Device1.PartLeav');

             app.timerObj = timer('TimerFcn', @(~,~)getModelValues(app),
'Period', 0.1, 'ExecutionMode', 'fixedSpacing', 'BusyMode', 'drop');
             start(app.timerObj);
```

```
            set_param('Proceso1App/Start','value','0');  %Inicialización
de valores
            set_param('Proceso1App/Paro','value','1');
            set_param('Proceso1App/Man','value','1');
            set_param('Proceso1App/Auto','value','0');
            set_param('Proceso1App/Lc','value','0');
            set_param('Proceso1App/Bc','value','0');
            set_param('Proceso1App/te','value','0');
            set_param('Proceso1App/be','value','0');
            set_param('Proceso1App/cb','value','0');
            set_param('Proceso1App/ct','value','0');
            set_param('Proceso1App/pl','value','0');
            set_param('Proceso1App/pb','value','0');
            set_param('Proceso1App/mx','value','0');
            set_param('Proceso1App/mz','value','0');
            set_param('Proceso1App/gr','value','0');
            set_param('Proceso1App/Emergencia','value','1');
            set_param('Proceso1App/Reset','value','0');




        end
        % Value changed function: LidEmitterSwitch
        function LidEmitterSwitchValueChanged(app, event)

            switch  app.LidEmitterSwitch.Value
                case 'On'
                set_param('Proceso1App/te','value','1');

                case 'Off'
                set_param('Proceso1App/te','value','0');

            end



        end
        % Value changed function: LidConveyorSwitch_2
        function LidConveyorSwitch_2ValueChanged(app, event)
             switch  app.LidConveyorSwitch_2.Value
                case 'On'
                set_param('Proceso1App/Lc','value','1')

                case 'Off'
                set_param('Proceso1App/Lc','value','0')

            end

        end
        % Value changed function: ClampLidSwitch_2
        function ClampLidSwitch_2ValueChanged(app, event)
            switch  app.ClampLidSwitch_2.Value
```

```
            case 'On'
            set_param('Proceso1App/ct','value','1')

            case 'Off'
            set_param('Proceso1App/ct','value','0')

        end

    end
    % Value changed function: PosRaiseLidSwitch_2
    function PosRaiseLidSwitch_2ValueChanged(app, event)
        switch  app.PosRaiseLidSwitch_2.Value
            case 'On'
            set_param('Proceso1App/pl','value','1')

            case 'Off'
            set_param('Proceso1App/pl','value','0')

        end

    end
    % Value changed function: BasesEmitterSwitch
    function BasesEmitterSwitchValueChanged(app, event)
        switch  app.BasesEmitterSwitch.Value
            case 'On'
            set_param('Proceso1App/be','value','1')

            case 'Off'
            set_param('Proceso1App/be','value','0')

        end

    end
    % Value changed function: BasesConveyorSwitch_2
    function BasesConveyorSwitch_2ValueChanged(app, event)
        switch  app.BasesConveyorSwitch_2.Value
            case 'On'
            set_param('Proceso1App/Bc','value','1')

            case 'Off'
            set_param('Proceso1App/Bc','value','0')

        end

    end
    % Value changed function: ClampBasesSwitch_2
    function ClampBasesSwitch_2ValueChanged(app, event)
        switch  app.ClampBasesSwitch_2.Value
            case 'On'
            set_param('Proceso1App/cb','value','1')

            case 'Off'
            set_param('Proceso1App/cb','value','0')
```

```
        end

    end
    % Value changed function: PosRaiseBasesSwitch_2
    function PosRaiseBasesSwitch_2ValueChanged(app, event)
        switch  app.PosRaiseBasesSwitch_2.Value
            case 'On'
            set_param('Proceso1App/pb','value','1')

            case 'Off'
            set_param('Proceso1App/pb','value','0')

        end

    end
    % Value changed function: MoveXSwitch
    function MoveXSwitchValueChanged(app, event)
        switch  app.MoveXSwitch.Value
            case 'On'
            set_param('Proceso1App/mx','value','1')

            case 'Off'
            set_param('Proceso1App/mx','value','0')

        end

    end
    % Value changed function: MoveZSwitch
    function MoveZSwitchValueChanged(app, event)
        switch  app.MoveZSwitch.Value
            case 'On'
            set_param('Proceso1App/mz','value','1')

            case 'Off'
            set_param('Proceso1App/mz','value','0')

        end

    end
    % Value changed function: GrabSwitch
    function GrabSwitchValueChanged(app, event)
        switch  app.GrabSwitch.Value
            case 'On'
            set_param('Proceso1App/gr','value','1')

            case 'Off'
            set_param('Proceso1App/gr','value','0')

        end

    end
    % Button pushed function: StartButton
```

```
        function StartButtonPushed(app, event)
            if app.StartButton.Interruptible=='on'
                set_param('Proceso1App/Start','value','1')
                app.Lamp_14.Enable='on';
                app.Lamp_15.Enable='off';
                app.Lamp_13.Enable='off';
                set_param('Proceso1App/Paro','value','1')
                set_param('Proceso1App/Reset','value','0')

            end
        end
        % Value changed function: MododefuncionamientoKnob
        function MododefuncionamientoKnobValueChanged(app, event)
            value = app.MododefuncionamientoKnob.Value;
            switch value
                case 'Manual'
                    set_param('Proceso1App/Man','value','1')
                    set_param('Proceso1App/Auto','value','0')
                case 'Automático'
                    set_param('Proceso1App/Man','value','0')
                    set_param('Proceso1App/Auto','value','1')
            end
        end
        % Button pushed function: StopButton
        function StopButtonPushed(app, event)
            if app.StopButton.Interruptible=='on'
                set_param('Proceso1App/Paro','value','0')
                app.Lamp_15.Enable='on';
                app.Lamp_14.Enable='off';
                set_param('Proceso1App/Start','value','0')
            end
        end
        % Value changed function: EmergenciaButton
        function EmergenciaButtonValueChanged(app, event)
            value = app.EmergenciaButton.Value;
            if value==1
                set_param('Proceso1App/Emergencia','value','0')
            else value==0
                set_param('Proceso1App/Emergencia','value','1')
            end

        end
        % Button pushed function: ResetButton
        function ResetButtonPushed(app, event)
            if app.ResetButton.Interruptible=='on'
                set_param('Proceso1App/Reset','value','1');
                app.Lamp_13.Enable='on'
            end

        end
    end
    % App initialization and construction
    methods (Access = private)
```

```
        % Create UIFigure and components
        function createComponents(app)
            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [100 100 1005 999];
            app.UIFigure.Name = 'UI Figure';
            % Create PanelCintaTapas
            app.PanelCintaTapas = uipanel(app.UIFigure);
            app.PanelCintaTapas.TitlePosition = 'centertop';
            app.PanelCintaTapas.Title = 'Panel Cinta Tapas';
            app.PanelCintaTapas.FontWeight = 'bold';
            app.PanelCintaTapas.FontSize = 14;
            app.PanelCintaTapas.Position = [1 536 363 273];
            % Create LidEmitterSwitchLabel
            app.LidEmitterSwitchLabel = uilabel(app.PanelCintaTapas);
            app.LidEmitterSwitchLabel.HorizontalAlignment = 'center';
            app.LidEmitterSwitchLabel.Position = [76 188 63 22];
            app.LidEmitterSwitchLabel.Text = 'Lid Emitter';
            % Create LidEmitterSwitch
            app.LidEmitterSwitch = uiswitch(app.PanelCintaTapas,
'slider');
            app.LidEmitterSwitch.ValueChangedFcn = createCallbackFcn(app,
@LidEmitterSwitchValueChanged, true);
            app.LidEmitterSwitch.Position = [210 190 47 20];
            % Create PosRaiseLidSwitch_2Label
            app.PosRaiseLidSwitch_2Label = uilabel(app.PanelCintaTapas);
            app.PosRaiseLidSwitch_2Label.HorizontalAlignment = 'center';
            app.PosRaiseLidSwitch_2Label.Position = [76 34 80 22];
            app.PosRaiseLidSwitch_2Label.Text = 'Pos Raise Lid';
            % Create PosRaiseLidSwitch_2
            app.PosRaiseLidSwitch_2 = uiswitch(app.PanelCintaTapas,
'slider');
            app.PosRaiseLidSwitch_2.ValueChangedFcn =
createCallbackFcn(app, @PosRaiseLidSwitch_2ValueChanged, true);
            app.PosRaiseLidSwitch_2.Position = [208 35 49 21];
            % Create ClampLidSwitch_2Label
            app.ClampLidSwitch_2Label = uilabel(app.PanelCintaTapas);
            app.ClampLidSwitch_2Label.HorizontalAlignment = 'center';
            app.ClampLidSwitch_2Label.Position = [75 86 60 22];
            app.ClampLidSwitch_2Label.Text = 'Clamp Lid';
            % Create ClampLidSwitch_2
            app.ClampLidSwitch_2 = uiswitch(app.PanelCintaTapas,
'slider');
            app.ClampLidSwitch_2.ValueChangedFcn = createCallbackFcn(app,
@ClampLidSwitch_2ValueChanged, true);
            app.ClampLidSwitch_2.Position = [209 88 48 21];
            % Create LidConveyorSwitch_2Label
            app.LidConveyorSwitch_2Label = uilabel(app.PanelCintaTapas);
            app.LidConveyorSwitch_2Label.HorizontalAlignment = 'center';
            app.LidConveyorSwitch_2Label.Position = [76 136 76 22];
            app.LidConveyorSwitch_2Label.Text = 'Lid Conveyor';
            % Create LidConveyorSwitch_2
```

```
            app.LidConveyorSwitch_2 = uiswitch(app.PanelCintaTapas,
'slider');
            app.LidConveyorSwitch_2.ValueChangedFcn =
createCallbackFcn(app, @LidConveyorSwitch_2ValueChanged, true);
            app.LidConveyorSwitch_2.Position = [210 138 47 20];
            % Create PanelCintaBases
            app.PanelCintaBases = uipanel(app.UIFigure);
            app.PanelCintaBases.TitlePosition = 'centertop';
            app.PanelCintaBases.Title = 'Panel Cinta Bases';
            app.PanelCintaBases.FontWeight = 'bold';
            app.PanelCintaBases.FontSize = 14;
            app.PanelCintaBases.Position = [625 536 358 273];
            % Create BasesEmitterSwitchLabel
            app.BasesEmitterSwitchLabel = uilabel(app.PanelCintaBases);
            app.BasesEmitterSwitchLabel.HorizontalAlignment = 'center';
            app.BasesEmitterSwitchLabel.Position = [72 187 80 22];
            app.BasesEmitterSwitchLabel.Text = 'Bases Emitter';
            % Create BasesEmitterSwitch
            app.BasesEmitterSwitch = uiswitch(app.PanelCintaBases,
'slider');
            app.BasesEmitterSwitch.ValueChangedFcn =
createCallbackFcn(app, @BasesEmitterSwitchValueChanged, true);
            app.BasesEmitterSwitch.Position = [208 189 45 20];
            % Create PosRaiseBasesSwitch_2Label
            app.PosRaiseBasesSwitch_2Label =
uilabel(app.PanelCintaBases);
            app.PosRaiseBasesSwitch_2Label.HorizontalAlignment =
'center';
            app.PosRaiseBasesSwitch_2Label.Position = [69 34 97 22];
            app.PosRaiseBasesSwitch_2Label.Text = 'Pos Raise Bases';
            % Create PosRaiseBasesSwitch_2
            app.PosRaiseBasesSwitch_2 = uiswitch(app.PanelCintaBases,
'slider');
            app.PosRaiseBasesSwitch_2.ValueChangedFcn =
createCallbackFcn(app, @PosRaiseBasesSwitch_2ValueChanged, true);
            app.PosRaiseBasesSwitch_2.Position = [208 37 45 20];
            % Create ClampBasesSwitch_2Label
            app.ClampBasesSwitch_2Label = uilabel(app.PanelCintaBases);
            app.ClampBasesSwitch_2Label.HorizontalAlignment = 'center';
            app.ClampBasesSwitch_2Label.Position = [69 87 77 22];
            app.ClampBasesSwitch_2Label.Text = 'Clamp Bases';
            % Create ClampBasesSwitch_2
            app.ClampBasesSwitch_2 = uiswitch(app.PanelCintaBases,
'slider');
            app.ClampBasesSwitch_2.ValueChangedFcn =
createCallbackFcn(app, @ClampBasesSwitch_2ValueChanged, true);
            app.ClampBasesSwitch_2.Position = [208 88 45 20];
            % Create BasesConveyorSwitch_2Label
            app.BasesConveyorSwitch_2Label =
uilabel(app.PanelCintaBases);
            app.BasesConveyorSwitch_2Label.HorizontalAlignment =
'center';
            app.BasesConveyorSwitch_2Label.Position = [69 137 94 22];
```

```
            app.BasesConveyorSwitch_2Label.Text = 'Bases Conveyor';
            % Create BasesConveyorSwitch_2
            app.BasesConveyorSwitch_2 = uiswitch(app.PanelCintaBases,
'slider');
            app.BasesConveyorSwitch_2.ValueChangedFcn =
createCallbackFcn(app, @BasesConveyorSwitch_2ValueChanged, true);
            app.BasesConveyorSwitch_2.Position = [208 139 45 20];
            % Create PanelBrazoRobot
            app.PanelBrazoRobot = uipanel(app.UIFigure);
            app.PanelBrazoRobot.TitlePosition = 'centertop';
            app.PanelBrazoRobot.Title = 'Panel Brazo Robot';
            app.PanelBrazoRobot.FontWeight = 'bold';
            app.PanelBrazoRobot.FontSize = 14;
            app.PanelBrazoRobot.Position = [363 536 263 273];
            % Create MoveXSwitchLabel
            app.MoveXSwitchLabel = uilabel(app.PanelBrazoRobot);
            app.MoveXSwitchLabel.HorizontalAlignment = 'center';
            app.MoveXSwitchLabel.Position = [57 187 46 22];
            app.MoveXSwitchLabel.Text = 'Move X';
            % Create MoveXSwitch
            app.MoveXSwitch = uiswitch(app.PanelBrazoRobot, 'slider');
            app.MoveXSwitch.ValueChangedFcn = createCallbackFcn(app,
@MoveXSwitchValueChanged, true);
            app.MoveXSwitch.Position = [140 190 45 20];
            % Create MoveZSwitchLabel
            app.MoveZSwitchLabel = uilabel(app.PanelBrazoRobot);
            app.MoveZSwitchLabel.HorizontalAlignment = 'center';
            app.MoveZSwitchLabel.Position = [57 125 46 22];
            app.MoveZSwitchLabel.Text = 'Move Z';
            % Create MoveZSwitch
            app.MoveZSwitch = uiswitch(app.PanelBrazoRobot, 'slider');
            app.MoveZSwitch.ValueChangedFcn = createCallbackFcn(app,
@MoveZSwitchValueChanged, true);
            app.MoveZSwitch.Position = [140 128 45 20];
            % Create GrabSwitchLabel
            app.GrabSwitchLabel = uilabel(app.PanelBrazoRobot);
            app.GrabSwitchLabel.HorizontalAlignment = 'center';
            app.GrabSwitchLabel.Position = [64 69 32 22];
            app.GrabSwitchLabel.Text = 'Grab';
            % Create GrabSwitch
            app.GrabSwitch = uiswitch(app.PanelBrazoRobot, 'slider');
            app.GrabSwitch.ValueChangedFcn = createCallbackFcn(app,
@GrabSwitchValueChanged, true);
            app.GrabSwitch.Position = [140 72 45 20];
            % Create ControladorPanel
            app.ControladorPanel = uipanel(app.UIFigure);
            app.ControladorPanel.TitlePosition = 'centertop';
            app.ControladorPanel.Title = 'Controlador';
            app.ControladorPanel.FontWeight = 'bold';
            app.ControladorPanel.FontSize = 14;
            app.ControladorPanel.Position = [1 808 982 192];
            % Create MododefuncionamientoKnobLabel
```

```
            app.MododefuncionamientoKnobLabel =
uilabel(app.ControladorPanel);
            app.MododefuncionamientoKnobLabel.HorizontalAlignment =
'center';
            app.MododefuncionamientoKnobLabel.Position = [781 32 137 22];
            app.MododefuncionamientoKnobLabel.Text = 'Modo de
funcionamiento';
            % Create MododefuncionamientoKnob
            app.MododefuncionamientoKnob = uiknob(app.ControladorPanel,
'discrete');
            app.MododefuncionamientoKnob.Items = {'Manual',
'Automático'};
            app.MododefuncionamientoKnob.ValueChangedFcn =
createCallbackFcn(app, @MododefuncionamientoKnobValueChanged, true);
            app.MododefuncionamientoKnob.Position = [811 62 60 60];
            app.MododefuncionamientoKnob.Value = 'Manual';
            % Create StartButton
            app.StartButton = uibutton(app.ControladorPanel, 'push');
            app.StartButton.ButtonPushedFcn = createCallbackFcn(app,
@StartButtonPushed, true);
            app.StartButton.Position = [50 46 54 32];
            app.StartButton.Text = 'Start';
            % Create StopButton
            app.StopButton = uibutton(app.ControladorPanel, 'push');
            app.StopButton.ButtonPushedFcn = createCallbackFcn(app,
@StopButtonPushed, true);
            app.StopButton.Position = [279 46 54 33];
            app.StopButton.Text = 'Stop';
            % Create Label_3
            app.Label_3 = uilabel(app.ControladorPanel);
            app.Label_3.HorizontalAlignment = 'right';
            app.Label_3.Enable = 'off';
            app.Label_3.Position = [255 109 25 22];
            app.Label_3.Text = '';
            % Create Lamp_13
            app.Lamp_13 = uilamp(app.ControladorPanel);
            app.Lamp_13.Enable = 'off';
            app.Lamp_13.Position = [177 106 27 27];
            app.Lamp_13.Color = [0.9294 0.6902 0.1294];
            % Create Lamp_14
            app.Lamp_14 = uilamp(app.ControladorPanel);
            app.Lamp_14.Enable = 'off';
            app.Lamp_14.Position = [64 106 26 26];
            % Create Lamp_15
            app.Lamp_15 = uilamp(app.ControladorPanel);
            app.Lamp_15.Enable = 'off';
            app.Lamp_15.Position = [294 107 25 25];
            app.Lamp_15.Color = [1 0 0];
            % Create EmergenciaButton
            app.EmergenciaButton = uibutton(app.ControladorPanel,
'state');
            app.EmergenciaButton.ValueChangedFcn = createCallbackFcn(app,
@EmergenciaButtonValueChanged, true);
```

```
            app.EmergenciaButton.Text = 'Emergencia';
            app.EmergenciaButton.Position = [464 46 89 67];
            % Create ResetButton
            app.ResetButton = uibutton(app.ControladorPanel, 'push');
            app.ResetButton.ButtonPushedFcn = createCallbackFcn(app,
@ResetButtonPushed, true);
            app.ResetButton.Position = [164 46 54 33];
            app.ResetButton.Text = 'Reset';
            % Create Panel
            app.Panel = uipanel(app.UIFigure);
            app.Panel.TitlePosition = 'centertop';
            app.Panel.BackgroundColor = [0.651 0.651 0.651];
            app.Panel.Position = [59 257 886 113];
            % Create LidEmitterLampLabel
            app.LidEmitterLampLabel = uilabel(app.Panel);
            app.LidEmitterLampLabel.HorizontalAlignment = 'right';
            app.LidEmitterLampLabel.Position = [818 74 63 22];
            app.LidEmitterLampLabel.Text = 'Lid Emitter';
            % Create LidEmitterLamp
            app.LidEmitterLamp = uilamp(app.Panel);
            app.LidEmitterLamp.Enable = 'off';
            app.LidEmitterLamp.Position = [841 53 20 20];
            % Create LidConveyorLampLabel
            app.LidConveyorLampLabel = uilabel(app.Panel);
            app.LidConveyorLampLabel.HorizontalAlignment = 'right';
            app.LidConveyorLampLabel.Position = [647 63 76 22];
            app.LidConveyorLampLabel.Text = 'Lid Conveyor';
            % Create LidConveyorLamp
            app.LidConveyorLamp = uilamp(app.Panel);
            app.LidConveyorLamp.Enable = 'off';
            app.LidConveyorLamp.Position = [681 42 20 20];
            % Create ClampLidLampLabel
            app.ClampLidLampLabel = uilabel(app.Panel);
            app.ClampLidLampLabel.HorizontalAlignment = 'right';
            app.ClampLidLampLabel.Position = [347 1 60 22];
            app.ClampLidLampLabel.Text = 'Clamp Lid';
            % Create ClampLidLamp
            app.ClampLidLamp = uilamp(app.Panel);
            app.ClampLidLamp.Enable = 'off';
            app.ClampLidLamp.Position = [422 1 20 20];
            % Create LidClampedLampLabel
            app.LidClampedLampLabel = uilabel(app.Panel);
            app.LidClampedLampLabel.HorizontalAlignment = 'right';
            app.LidClampedLampLabel.Position = [473 1 73 22];
            app.LidClampedLampLabel.Text = 'Lid Clamped';
            % Create LidClampedLamp
            app.LidClampedLamp = uilamp(app.Panel);
            app.LidClampedLamp.Enable = 'off';
            app.LidClampedLamp.Position = [447 1 20 20];
            app.LidClampedLamp.Color = [0 0 1];
            % Create LidatPlaceLampLabel
            app.LidatPlaceLampLabel = uilabel(app.Panel);
            app.LidatPlaceLampLabel.HorizontalAlignment = 'right';
```

```
app.LidatPlaceLampLabel.Position = [608 1 68 22];
app.LidatPlaceLampLabel.Text = 'Lid at Place';
% Create LidatPlaceLamp
app.LidatPlaceLamp = uilamp(app.Panel);
app.LidatPlaceLamp.Enable = 'off';
app.LidatPlaceLamp.Position = [588 1 20 20];
app.LidatPlaceLamp.Color = [0 0 1];
% Create Panel_2
app.Panel_2 = uipanel(app.UIFigure);
app.Panel_2.TitlePosition = 'centertop';
app.Panel_2.BackgroundColor = [0.651 0.651 0.651];
app.Panel_2.Position = [59 61 886 116];
% Create BasesEmitterLampLabel
app.BasesEmitterLampLabel = uilabel(app.Panel_2);
app.BasesEmitterLampLabel.HorizontalAlignment = 'right';
app.BasesEmitterLampLabel.Position = [804 68 80 22];
app.BasesEmitterLampLabel.Text = 'Bases Emitter';
% Create BasesEmitterLamp
app.BasesEmitterLamp = uilamp(app.Panel_2);
app.BasesEmitterLamp.Enable = 'off';
app.BasesEmitterLamp.Position = [841 47 20 20];
% Create BaseConveyorLampLabel
app.BaseConveyorLampLabel = uilabel(app.Panel_2);
app.BaseConveyorLampLabel.HorizontalAlignment = 'right';
app.BaseConveyorLampLabel.Position = [639 68 88 22];
app.BaseConveyorLampLabel.Text = 'Base Conveyor';
% Create BaseConveyorLamp
app.BaseConveyorLamp = uilamp(app.Panel_2);
app.BaseConveyorLamp.Enable = 'off';
app.BaseConveyorLamp.Position = [685 47 20 20];
% Create ClampBaseLampLabel
app.ClampBaseLampLabel = uilabel(app.Panel_2);
app.ClampBaseLampLabel.HorizontalAlignment = 'right';
app.ClampBaseLampLabel.Position = [336 1 71 22];
app.ClampBaseLampLabel.Text = 'Clamp Base';
% Create ClampBaseLamp
app.ClampBaseLamp = uilamp(app.Panel_2);
app.ClampBaseLamp.Enable = 'off';
app.ClampBaseLamp.Position = [422 1 20 20];
% Create BaseClampedLampLabel
app.BaseClampedLampLabel = uilabel(app.Panel_2);
app.BaseClampedLampLabel.HorizontalAlignment = 'right';
app.BaseClampedLampLabel.Position = [479 1 84 22];
app.BaseClampedLampLabel.Text = 'Base Clamped';
% Create BaseClampedLamp
app.BaseClampedLamp = uilamp(app.Panel_2);
app.BaseClampedLamp.Enable = 'off';
app.BaseClampedLamp.Position = [447 1 20 20];
app.BaseClampedLamp.Color = [0 0 1];
% Create BaseatPlaceLampLabel
app.BaseatPlaceLampLabel = uilabel(app.Panel_2);
app.BaseatPlaceLampLabel.HorizontalAlignment = 'right';
app.BaseatPlaceLampLabel.Position = [615 1 80 22];
```

```
app.BaseatPlaceLampLabel.Text = 'Base at Place';
% Create BaseatPlaceLamp
app.BaseatPlaceLamp = uilamp(app.Panel_2);
app.BaseatPlaceLamp.Enable = 'off';
app.BaseatPlaceLamp.Position = [594 1 20 20];
app.BaseatPlaceLamp.Color = [0 0 1];
% Create PartLeavingLampLabel
app.PartLeavingLampLabel = uilabel(app.Panel_2);
app.PartLeavingLampLabel.HorizontalAlignment = 'right';
app.PartLeavingLampLabel.Position = [168 1 73 22];
app.PartLeavingLampLabel.Text = 'Part Leaving';
% Create PartLeavingLamp
app.PartLeavingLamp = uilamp(app.Panel_2);
app.PartLeavingLamp.Enable = 'off';
app.PartLeavingLamp.Position = [260 1 20 20];
app.PartLeavingLamp.Color = [0 0 1];
% Create Panel_3
app.Panel_3 = uipanel(app.UIFigure);
app.Panel_3.BackgroundColor = [1 0 0];
app.Panel_3.Position = [544 331 110 170];
% Create MoveXLampLabel
app.MoveXLampLabel = uilabel(app.Panel_3);
app.MoveXLampLabel.HorizontalAlignment = 'right';
app.MoveXLampLabel.Position = [-3 150 43 22];
app.MoveXLampLabel.Text = 'MoveX';
% Create MoveXLamp
app.MoveXLamp = uilamp(app.Panel_3);
app.MoveXLamp.Enable = 'off';
app.MoveXLamp.Position = [12 122 20 20];
% Create MoveZLampLabel
app.MoveZLampLabel = uilabel(app.Panel_3);
app.MoveZLampLabel.HorizontalAlignment = 'right';
app.MoveZLampLabel.Position = [-1 101 42 22];
app.MoveZLampLabel.Text = 'MoveZ';
% Create MoveZLamp
app.MoveZLamp = uilamp(app.Panel_3);
app.MoveZLamp.Enable = 'off';
app.MoveZLamp.Position = [15 74 20 20];
% Create MovingXLampLabel
app.MovingXLampLabel = uilabel(app.Panel_3);
app.MovingXLampLabel.HorizontalAlignment = 'right';
app.MovingXLampLabel.Position = [52 150 52 22];
app.MovingXLampLabel.Text = 'MovingX';
% Create MovingXLamp
app.MovingXLamp = uilamp(app.Panel_3);
app.MovingXLamp.Enable = 'off';
app.MovingXLamp.Position = [63 122 20 20];
app.MovingXLamp.Color = [0 0 1];
% Create MovingZLampLabel
app.MovingZLampLabel = uilabel(app.Panel_3);
app.MovingZLampLabel.HorizontalAlignment = 'right';
app.MovingZLampLabel.Position = [52 101 52 22];
app.MovingZLampLabel.Text = 'MovingZ';
```

```
% Create MovingZLamp
app.MovingZLamp = uilamp(app.Panel_3);
app.MovingZLamp.Enable = 'off';
app.MovingZLamp.Position = [66 73 20 20];
app.MovingZLamp.Color = [0 0 1];
% Create GrabLampLabel
app.GrabLampLabel = uilabel(app.Panel_3);
app.GrabLampLabel.HorizontalAlignment = 'right';
app.GrabLampLabel.Position = [8 42 32 22];
app.GrabLampLabel.Text = 'Grab';
% Create GrabLamp
app.GrabLamp = uilamp(app.Panel_3);
app.GrabLamp.Enable = 'off';
app.GrabLamp.Position = [16 19 20 20];
% Create PosraiseLidLampLabel
app.PosraiseLidLampLabel = uilabel(app.UIFigure);
app.PosraiseLidLampLabel.HorizontalAlignment = 'right';
app.PosraiseLidLampLabel.Position = [374 369 75 22];
app.PosraiseLidLampLabel.Text = 'Pos raise Lid';
% Create PosraiseLidLamp
app.PosraiseLidLamp = uilamp(app.UIFigure);
app.PosraiseLidLamp.Enable = 'off';
app.PosraiseLidLamp.Position = [464 370 20 20];
% Create PosraiseBaseLampLabel
app.PosraiseBaseLampLabel = uilabel(app.UIFigure);
app.PosraiseBaseLampLabel.HorizontalAlignment = 'right';
app.PosraiseBaseLampLabel.Position = [379 185 86 22];
app.PosraiseBaseLampLabel.Text = 'Pos raise Base';
% Create PosraiseBaseLamp
app.PosraiseBaseLamp = uilamp(app.UIFigure);
app.PosraiseBaseLamp.Enable = 'off';
app.PosraiseBaseLamp.Position = [480 186 20 20];
% Create PosatBaseLampLabel
app.PosatBaseLampLabel = uilabel(app.UIFigure);
app.PosatBaseLampLabel.HorizontalAlignment = 'right';
app.PosatBaseLampLabel.Position = [395 206 70 22];
app.PosatBaseLampLabel.Text = 'Pos at Base';
% Create PosatBaseLamp
app.PosatBaseLamp = uilamp(app.UIFigure);
app.PosatBaseLamp.Enable = 'off';
app.PosatBaseLamp.Position = [480 207 20 20];
app.PosatBaseLamp.Color = [0 0 1];
% Create PosatLidLampLabel
app.PosatLidLampLabel = uilabel(app.UIFigure);
app.PosatLidLampLabel.HorizontalAlignment = 'right';
app.PosatLidLampLabel.Position = [390 390 59 22];
app.PosatLidLampLabel.Text = 'Pos at Lid';
% Create PosatLidLamp
app.PosatLidLamp = uilamp(app.UIFigure);
app.PosatLidLamp.Enable = 'off';
app.PosatLidLamp.Position = [464 391 20 20];
app.PosatLidLamp.Color = [0 0 1];
% Create ItemdetectedLampLabel
```

```
            app.ItemdetectedLampLabel = uilabel(app.UIFigure);
            app.ItemdetectedLampLabel.HorizontalAlignment = 'right';
            app.ItemdetectedLampLabel.Position = [643 352 78 22];
            app.ItemdetectedLampLabel.Text = 'Item detected';
            % Create ItemdetectedLamp
            app.ItemdetectedLamp = uilamp(app.UIFigure);
            app.ItemdetectedLamp.Enable = 'off';
            app.ItemdetectedLamp.Position = [610 350 20 20];
            app.ItemdetectedLamp.Color = [0 0 1];
            % Create ActuadoresLampLabel
            app.ActuadoresLampLabel = uilabel(app.UIFigure);
            app.ActuadoresLampLabel.HorizontalAlignment = 'right';
            app.ActuadoresLampLabel.Position = [45 460 66 22];
            app.ActuadoresLampLabel.Text = 'Actuadores';
            % Create ActuadoresLamp
            app.ActuadoresLamp = uilamp(app.UIFigure);
            app.ActuadoresLamp.Position = [126 460 20 20];
            % Create SensoresLampLabel
            app.SensoresLampLabel = uilabel(app.UIFigure);
            app.SensoresLampLabel.HorizontalAlignment = 'right';
            app.SensoresLampLabel.Position = [55 418 56 22];
            app.SensoresLampLabel.Text = 'Sensores';
            % Create SensoresLamp
            app.SensoresLamp = uilamp(app.UIFigure);
            app.SensoresLamp.Position = [126 418 20 20];
            app.SensoresLamp.Color = [0 0 1];
        end
    end
    methods (Access = public)
        % Construct app
        function app = appProcesoBueno
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            % Execute the startup function
            runStartupFcn(app, @startupFcn)
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```