Universidad del País Vasco
Euskal Herriko Unibertsitatea

BILBOKO
INGENIARITZA ESKOLA

ESCUELA
DE INGENIERÍA DE BILBAO

TRABAJO DE FIN DE GRADO

Grado en Ingeniería en Tecnología Industrial

MONITORIZACIÓN DE SISTEMAS DE CONTROL INDUSTRIAL BASADA EN ECLIPSE SCADA

Alumno/a *Salinero Lobo, Adrián*
Fecha *Julio, 2018*
Director/a *Pérez González, Federico*
Curso académico *2017/2018*

DATOS BÁSICOS DEL TRABAJO DE FIN DE GRADO

- *Alumno/a:* Adrián Salinero Lobo.
- *Director/a:* Federico Pérez González.
- *Departamento*: Departamento de Ingeniería de Sistemas y Automática.
- *Título del Trabajo*: Monitorización de Sistemas de Control Industrial Basada en Eclipse SCADA.
- *Resumen*: El objetivo del trabajo es la puesta en marcha de una plataforma Eclipse SCADA para la monitorización de un sistema básico de automatización con acceso a las variables de proceso a través de un sistema de comunicación con dispositivos de control.
- *Palabras clave*: Eclipse, SCADA, Software-in-the-loop, Java, S7, PLC, TIA PORTAL, PLCSIM Advanced, monitorización, control

- *Izenburua:* Eclipse SCADAn oinarritutako Kontrol Industrialeko Sistemen Monitorizazioa.
- *Laburpena:* Lan honen helburua Eclipse SCADA plataforma martxan jartzea da, oinarrizko automatizazio sistema baten jarraipena egiteko, prozesuaren aldagaietara sartzeko kontrol dispositiboak dituen komunikazio sistema baten bidez.
- *Hitzgakoak:* Eclipse, SCADA, Software-in-the-loop, Java, S7, PLC, TIA PORTAL, PLCSIM Advanced, monitorizazio, kontrol

- *Title*: Monitoring of Industrial Control Systems Based on Eclipse SCADA.
- *Abstract*: The aim of the project is the start-up of an Eclipse SCADA platform for the monitoring of a basic automation system with access to the process variables through a communication system with control devices.
- *Keywords*: Eclipse, SCADA, Software-in-the-loop, Java, S7, PLC, TIA PORTAL, PLCSIM Advanced, monitoring, control

INDICE DE CONTENIDOS

LISTA DE ABREVIATURAS

SCADA ..... *Supervisory Control And Data Acquisition*
SIL ..... *Software-in-the-loop*
HMI ..... *Human-Machine Interface*
PLC ..... *Programmable Logic Controller*
RTU ..... *Remote Terminal Unit*
DNP ..... *Distributed Network Protoco*l
IEC ..... *International Electrotechnical Commission*
IDE ..... *Integrated Development Environment*
PC ..... *Personal computer*
JDK ..... *Java Development Kit*
ESAC ..... *Eclipse* SCADA *Admin Client*
VI ..... *Visual Interface*
FOSS ..... *Free and Open Source Software*
SO ..... Sistema Operativo
JRE ..... *Java Runtime Environment*
ESCM ..... *Eclipse SCADA Component Model*
ESIM ..... *Eclipse SCADA Infrastructure Model*
DB ..... *Data Block*

INDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

Este Trabajo de Fin de Grado se basa en la puesta en marcha de una plataforma Eclipse SCADA para la monitorización de un sistema básico de automatización. El presente informe está dividido en cuatro partes:

En la primera se describe el contexto en que se realiza el proyecto, explicando al lector conceptos como SCADA, *Software-in-the-loop* (SIL) e introduciendo la plataforma Eclipse sobre la que se llevará a cabo. También se detallan los objetivos y el alcance del proyecto, y se analizan los beneficios que este pueda brindar una vez finalizado.

En la segunda parte se realiza el análisis de alternativas, en el cual se presentan y comparan las alternativas de software SCADA disponibles, para posteriormente elegir la más adecuada en función de las necesidades del proyecto.

A lo largo de la tercera parte se procede a detallar la implementación de la solución que se ha adoptado para realizar el proyecto. Primero se detalla la puesta en marcha de la plataforma Eclipse SCADA y a continuación, se prueba y valida la solución escogida mediante una aplicación sencilla.

Para concluir, se muestra mediante un diagrama de Gantt el tiempo de dedicación para las diferentes tareas que han sido necesarias para llevar a cabo el proyecto, y también el presupuesto estimado del mismo.

2 CONTEXTO

2.1 SCADA

2.1.1 Introducción

Supervisory Control And Data Acquisition (SCADA) no es una tecnología concreta, sino un tipo de aplicación que permite:

- Controlar y supervisar procesos industriales localmente o de forma remota
- Obtener, monitorizar y procesar información en tiempo real
- Interactuar directamente con dispositivos tales como sensores, válvulas, bombas o motores a través de una *Human-Machine Interface* (HMI).
- Registrar sucesos en archivos de registro

Los sistemas SCADA son cruciales dentro de los entornos industriales ya que ayudan a mantener la eficiencia, procesar información para tomar decisiones más inteligentes e informar de problemas para reducir las interrupciones.

Una arquitectura básica SCADA comienza con un *Programmable Logic Controller* (PLC) o un *Remote Terminal Unit* (RTU). Los PLCs y RTUs son microcomputadores que se comunican con una serie de objetos tales como maquinaria de fábrica, HMIs y sensores, y envían información desde dichos objetos hacia los ordenadores mediante software SCADA. El software SCADA procesa, distribuye y expone la información, ayudando a los operarios y otros empleados en los análisis y toma de decisiones.

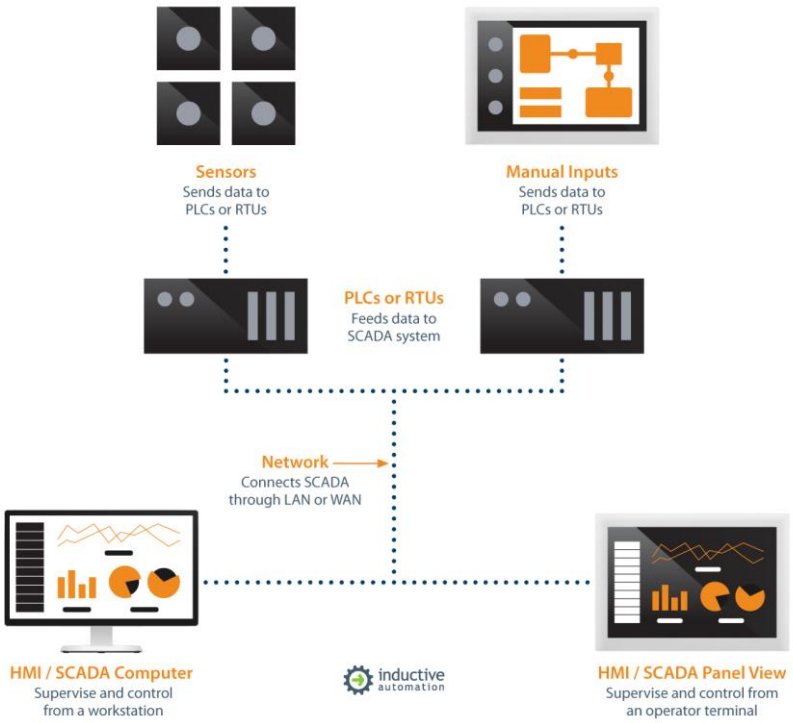


Figura 1. Diagrama básico de un sistema SCADA

Por ejemplo, el sistema SCADA rápidamente informa al operario de que un lote de producto contiene un elevado número de errores. El operario pausa la operación y observa la información del sistema SCADA para determinar la causa del fallo. El operario revisa la información y se da cuenta de que una de las máquinas no está funcionando correctamente. La capacidad del sistema SCADA de informar al operario acerca de un error le ayuda a arreglarlo y evitar mayores pérdidas de producto.

2.1.2 ¿Quién utiliza SCADA?

Los sistemas SCADA funcionan bien en diferentes tipos de empresa porque comprenden desde simple configuraciones hasta grandes y complejas instalaciones. Estos sistemas son la columna vertebral de muchas industrias modernas, ya que su uso efectivo puede resultar en ahorros significativos de tiempo y dinero, como muchos estudios han demostrado.

Hoy en día existen sistemas SCADA ejecutándose en segundo plano prácticamente en cualquier lugar del mundo como, por ejemplo, manteniendo los sistemas de refrigeración de un supermercado local, controlando la producción y seguridad de una fábrica, asegurando la calidad del agua en una planta de tratamiento de aguas e incluso controlando el uso de energía en las viviendas.

2.1.3 Evolución de los SCADA

Históricamente, los sistemas SCADA han utilizado protocolos de comunicación propietarios, desarrollados por los fabricantes de hardware para satisfacer las necesidades específicas de las industrias, pero incapaces de conectarse con dispositivos de otras marcas. Este tipo de protocolos tiene como desventaja que cuando se quiere ampliar el sistema el usuario está obligado a seguir utilizando equipo de ese mismo fabricante o a reemplazar parte de su sistema para poder utilizar el protocolo de otra compañía distinta. Un ejemplo es el protocolo desarrollado por Siemens para la comunicación de sus PLCs, denominado S7.

Debido a esta clara desventaja y al uso incremental de los sistemas SCADA, se reconoció la necesidad de protocolos abiertos. Esto se tradujo en esfuerzos por parte de numerosas empresas de distintos países por el desarrollo conjunto de este tipo de protocolos, como pueden ser, por ejemplo, Modbus, DNP (*Distributed Network Protoco*l) o IEC 60870 (*International Electrotechnical Commission* 60870).

La principal ventaja de los protocolos abiertos es que permiten integrar equipos de diversas marcas dentro del mismo sistema. A esta característica se le conoce como interoperabilidad, la cual proporciona gran versatilidad a los sistemas y elimina la dependencia de cualquier marca.

El reto de los SCADA de hoy en día es integrar componentes de diferentes fabricantes y conseguir que se comuniquen unos con otros. En este contexto, una plataforma como la que se desarrolla en este trabajo puede convertirse en un servicio interesante, ya que soporta una amplia variedad de protocolos de comunicación, tanto abiertos como propietarios.

2.2 Eclipse

Eclipse es un *Integrated Development Environment (*IDE) escrito en Java y destinado principalmente al desarrollo de aplicaciones en dicho lenguaje de programación, aunque también soporta otros lenguajes. Es un proyecto de código abierto que comenzó en el año 2001 y hoy en día es la IDE más utilizada para el desarrollo de aplicaciones Java.

Una de sus principales características es su extensa colección de *plug-ins*, los cuales añaden nuevas funcionalidades. La colección de *plug-ins* disponible es muy grande, ya que se van añadiendo nuevos a lo largo del tiempo. Estos se descargan desde *Eclipse Marketplace*, donde se pueden encontrar tanto gratuitos como de pago.

Eclipse SCADA es el sucesor de otro proyecto denominado *open*SCADA. La empresa desarrolladora de *open*SCADA decidió migrar su proyecto a Eclipse con el objetivo de aumentar su visibilidad y atraer a otras compañías y contribuyentes. Al tratarse de una empresa pequeña, es imposible lograr que su proyecto se convierta en un estándar de la industria, sin embargo, el formar parte de Eclipse puede ayudar a relacionarse con otros proyectos y juntos ampliar su nicho de mercado.

Figura 2. Logotipo de Eclipse

2.3 *Software-in-the-loop* (SIL)

Este término se utiliza para describir una metodología de prueba en la que se testea y verifica un software, firmware, algoritmo o sistema de control en un entorno de simulación virtual, sin utilizar hardware real.

Utilizando SIL los ingenieros pueden conectar el PC a un modelo de planta digital sustitutivo de otros sistemas o prototipos más costosos para testear y modificar de forma iterativa el software que se pretende desarrollar. SIL hace posible testear software de forma previa al comienzo de la fase de prototipo, acelerando de forma significativa el ciclo de desarrollo y permite la detección temprana de defectos y fallos, reduciendo significativamente los costes de una posterior fase de solución de problemas.

3 ALCANCE

El primer paso es realizar un análisis exhaustivo de las opciones disponibles para acometer el proyecto. Esto incluye la búsqueda de información en internet y la posterior comparación, que se detalla en el apartado *Análisis de alternativas*, para elegir el software SCADA que más se adapte a las necesidades del proyecto.

Una vez elegido el software más adecuado, el diseño de la plataforma Eclipse SCADA comienza con el estudio de los diferentes protocolos de comunicación con los que es compatible. Tras analizar todas las opciones, se ha optado por utilizar comunicación S7, el protocolo de comunicación propietario de Siemens, debido a que es la empresa líder en el campo de la automatización y a que ya se tienen conocimientos previos a la hora de manejar su software.

En la *Figura 3* se ha dibujado un esquema aproximado del sistema que se pretende desarrollar en este trabajo. Se dispone de un PLC encargado de controlar un proceso, que en este caso se trata de un pequeño manipulador neumático. Para programar dicho PLC se utiliza TIA PORTAL, el software desarrollado por Siemens para programar sus controladores. El otro elemento es el sistema SCADA, que comunica con el PLC mediante el protocolo S7 y se encarga de supervisar, monitorizar y controlar el proceso.

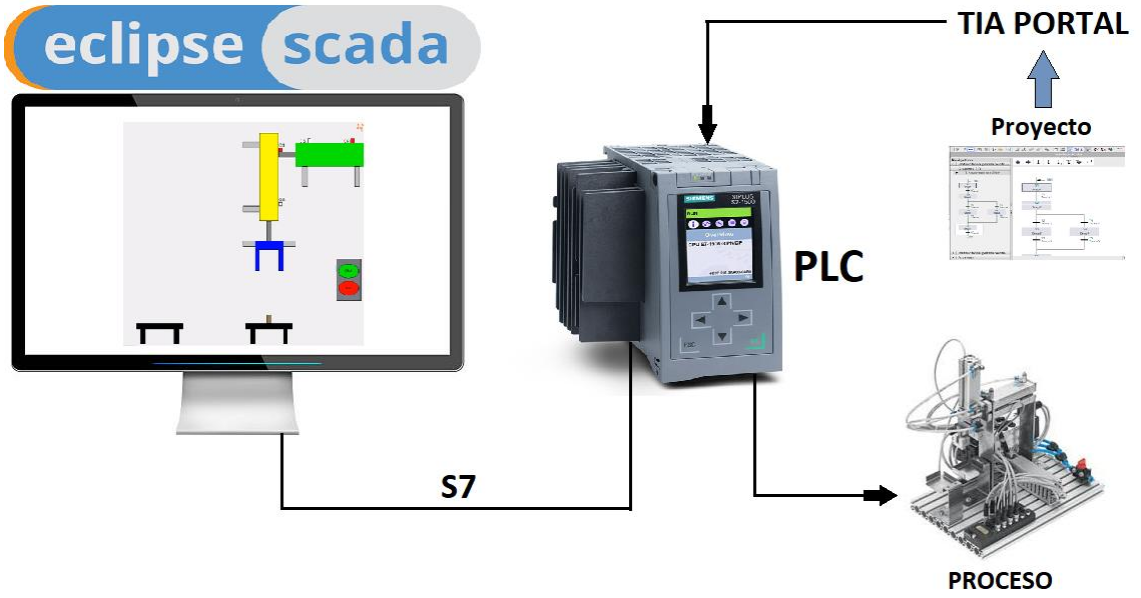


Figura 3. Esquema del sistema SCADA

En este caso no se dispone de un PLC real para llevar a cabo el proyecto, por lo que se ha tenido que sustituir dicho elemento por software de Siemens con capacidad para simular PLCs, en este caso se utilizará PLCSIM Advanced. En la *Figura 4* se puede observar un esquema aproximado del sistema definitivo que se ha desarrollado en este trabajo. De esta forma, se está utilizando la técnica de Software-in-the-loop (SIL) descrita anteriormente, ya que todo el testeo se lleva a cabo en un entorno simulado dentro de un mismo PC.

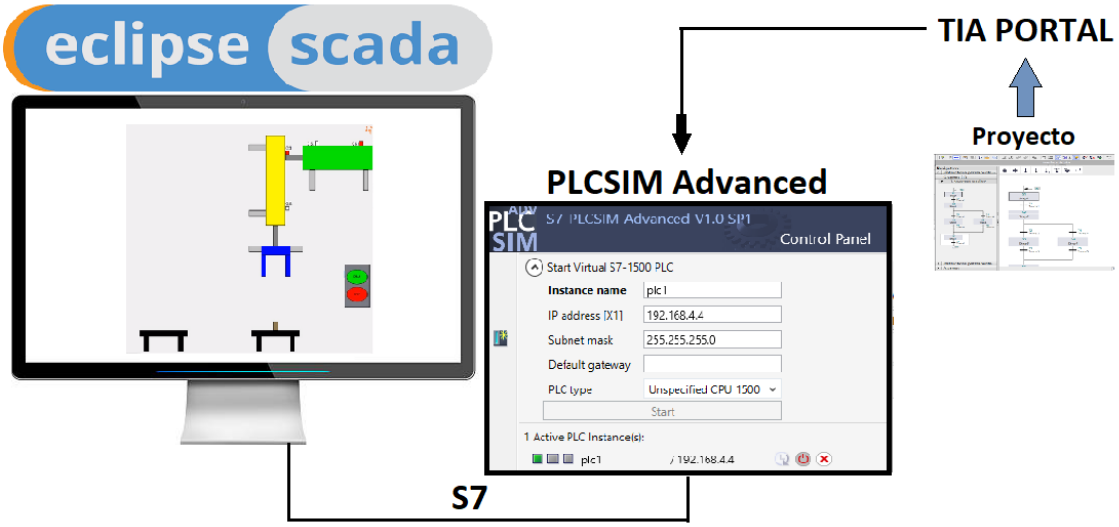


Figura 4. Esquema definitivo del sistema SCADA

Una vez terminado el diseño de la plataforma, se pasa a la etapa de desarrollo, la cual comienza con la instalación y configuración de *Java Development Kit* (JDK), requisito indispensable para poder desarrollar en Java dentro de Eclipse.

El siguiente paso consiste en instalar y configurar Eclipse, prestando especial atención a la compatibilidad entre la versión del JDK instalado anteriormente y la versión de Eclipse que se va a instalar, ya que cada versión de Eclipse requiere una versión del JDK diferente, lo cual podría ser fuente de incompatibilidades. Una vez instalada la versión correcta, se descargan dentro de Eclipse los *plug-ins* Eclipse SCADA, que contienen las componentes necesarias para crear los sistemas que se verán a continuación.

También es necesario descargar e instalar el *Eclipse* SCADA *Admin Client* (ESAC), un cliente externo a Eclipse desde el cual se monitorizan y modifican los datos del dispositivo al que se conecta, y el software de Siemens mencionado anteriormente.

Una vez llegados a este punto, al tener instalado y configurado todo el software necesario para desarrollar el proyecto se puede dar por concluida la puesta en marcha y comenzar con la fase de pruebas. En primer lugar, hay que configurar el *driver* y el *master server* para lograr que Eclipse SCADA se conecte al PLC simulado y pueda leer y modificar los datos allí almacenados. Por último, se desarrolla la *Visual Interface* (VI), un *dashboard* desde el que se controla y monitoriza el proceso en 2D.

El propósito principal del trabajo es la puesta en marcha de la plataforma Eclipse SCADA, por lo que queda fuera del alcance del proyecto el desarrollo de cualquier tipo de aplicación compleja.

4 BENEFICIOS DEL PROYECTO

4.1 Beneficios sociales

Primeramente, la apuesta por desarrollar aplicaciones basadas en proyectos de tipo *Free and Open Source Software* (FOSS) como la plataforma Eclipse SCADA que se ha utilizado para este proyecto tiene un doble objetivo: por un lado, reducir el coste del desarrollo, y por otro, acercar este desarrollo a la gente, haciéndoles partícipes del mismo y facilitando la educación de futuras generaciones. El software libre y de código abierto ofrece la posibilidad de acceder al código fuente de los programas, lo cual permite a las personas utilizar, estudiar, modificar, copiar y redistribuir el software.

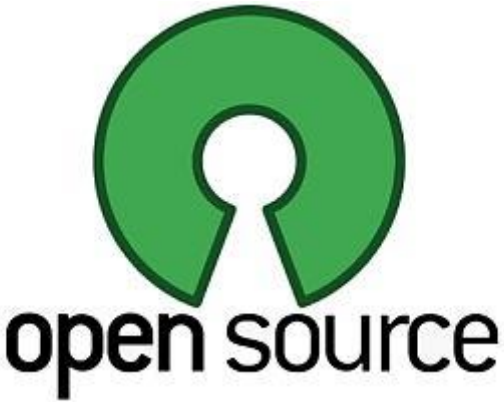


Figura 5. Logotipo de Open Source

En este proyecto se desarrolla un sistema SCADA utilizando, en medida de lo posible, software libre y de código abierto, conceptos que empujan de manera conjunta a la industria y la sociedad hacia un futuro de innovación abierta en el que las fronteras entre las empresas y la gente sean cada vez menos relevantes.

4.2 Beneficios económicos

La plataforma que se ha creado para este Trabajo de Fin de Grado puede considerarse de bajo coste, debido a que tanto el software Eclipse como los *plug-ins* que contienen las componentes Eclipse SCADA son completamente gratuitos. Un software SCADA de una compañía como Siemens o Rockwell Automation, tiene un coste mucho mayor que el de la plataforma diseñada, cuyo precio se detallará en el apartado de *Presupuesto*.

4.3 Beneficios técnicos

En cuanto al apartado técnico, el uso de software SCADA compatible con un número elevado de protocolos de comunicación, tanto abiertos como propietarios, supone la posibilidad integrar equipos de diversas marcas dentro de un mismo sistema, como se ha explicado anteriormente.

Muchos de los SCADA que se pueden encontrar en el mercado son soluciones *out-of-the-box*, están listos para ser utilizados después de su instalación sin necesidad de ningún tipo de configuración o modificación adicional, pero a cambio no permiten a los usuarios modificar su código fuente. Eclipse SCADA es todo lo contrario, se trata de una solución con un alto grado de personalización, lo cual permite a los usuarios añadir

nuevas características y modificar cualquier detalle, sin depender de la compañía desarrolladora.

Otro de los beneficios de la plataforma diseñada es que es fácil de utilizar por cualquier persona que tenga nociones básicas de programación en JavaScript. Además, existe de una pequeña comunidad de usuarios repartida entre el foro de Eclipse, GitHub y Google Groups, que puede ayudar a solucionar las dudas.

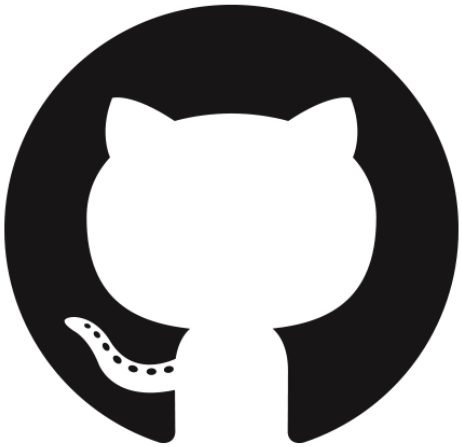


Figura 6. Logotipo de GitHub

5 ANALISIS DE ALTERNATIVAS

5.1 Introducción

En este apartado se van a detallar las alternativas de software SCADA disponibles para la realización del proyecto. Existen miles de alternativas en el mercado, por lo que solo se han tenido en cuenta las más utilizadas y conocidas globalmente, intentando incluir software de diferentes características para que exista variedad en los diferentes aspectos a la hora de hacer la comparativa.

5.2 SIMATIC WinCC

SIMATIC es el nombre del sistema de automatización desarrollado por la empresa alemana Siemens. Nació hace 60 años y permitió a Siemens establecerse como líder en el campo de la automatización. Dentro de SIMATIC existen aplicaciones con diferentes funciones, entre las cuales se encuentra WinCC, el software SCADA desarrollado por Siemens.

En cuanto a su capacidad de personalización, WincCC está preparado para trabajar *out-of-the-box*, es decir, requiere de una mínima configuración para empezar a funcionar, pero a cambio, no permite a los usuarios hacer modificaciones para adaptarlo a sus necesidades particulares.

Su precio varía en función del número de *tags* que es capaz de utilizar. Una licencia con capacidad para 128 *tags* tiene un precio aproximado de 3400€, mientras que para 256.000 *tags* el precio ascendería a 16400€. Estos precios no están listados en la página web de Siemens, hay que contactar con algún distribuidor y preguntar, es decir, solo quieren ofrecer esa información a la gente que está realmente interesada en contratar sus servicios.

En cuanto a la compatibilidad, solo es compatible con las últimas versiones de Microsoft Windows.

Figura 7. SIMATIC WinCC, el sistema SCADA de Siemens

Siemens ofrece numerosas opciones de soporte técnico: correo electrónico, soporte online o llamando a un número de teléfono el cual está disponible 24h./365 días. De esta forma ayudan a sus clientes a resolver los problemas e incidencias rápidamente. También existe un foro donde los usuarios pueden exponer sus dudas y problemas para recibir ayuda de otros usuarios y de los moderadores.

5.3 FactoryTalk View

FactoryTalk View es el sistema SCADA de Rockwell Automation, empresa americana que ofrece sistemas de automatización e información industrial fundada en el año 2001.

Figura 8. Logotipo de FactoryTalk View

Al igual que ocurre en el caso de WinCC, este sistema SCADA no es personalizable, en el sentido de que está diseñado tal y como Rockwell Automation desea y no ofrece a los usuarios flexibilidad en este aspecto.

En este caso también existe secretismo en cuanto al precio de las licencias. Según la información obtenida de varios vendedores particulares, hay que pagar aproximadamente 1800$ por la licencia del programa, además de otra licencia adicional en función del número de *displays* que se desean contratar:

Tipo de licencia	Precio
Licencia para 25 Displays	3200$
Licencia para 100 Displays	4700$
Licencia para 250 Displays	6600$
Licencia con Displays ilimitados	11000$

Tabla 1. Precios licencia FactoryTalk View

Solo es compatible con las últimas versiones de Microsoft Windows.

En cuanto al soporte técnico, al tratarse de la empresa más importante de automatización junto a Siemens y estar presente en multitud de mercados, dispone de las típicas opciones para ayudar a sus clientes: chat online, correo electrónico, teléfono de atención al cliente…

5.4 Ignition SCADA

Ignition es una plataforma de software para sistemas SCADA desarrollada por Inductive Automation y publicada en 2010. Esta empresa considera que los software SCADA disponibles en el mercado tienen una serie de problemas y ha diseñado su software Ignition buscando su solución y así poder venderse como la solución SCADA definitiva.

La principal característica de Ignition es que está basado en Tecnología Web. Todas las funcionalidades se configuran a través de un Cliente Web sin necesidad de instalación.

Tanto el acceso para Desarrollo como para *Runtime* se realizan a través de la Web empleando el innovador sistema Java Web Launch. La instalación y el mantenimiento de aplicaciones Cliente en cada puesto de trabajo han quedado en el pasado.

Figura 9. Logotipo de Ignition

Existen varios paquetes con diferentes características, el precio de los cuales se detalla y explica claramente en su página web, sin necesidad de hacer una búsqueda exhaustiva por Internet ni contactar con ningún distribuidor. Los precios listados en su página web pueden verse en la *Figura 10*.

Su punto fuerte en este aspecto es que una única licencia puede utilizarse en varios equipos simultáneamente, a diferencia de lo que ocurre con las licencias de WinCC o FactoryTalk View, las cuales solo se pueden utilizar en un PC al mismo tiempo, por lo que es necesario comprar varias licencias si se quiere poner en marcha una planta industrial de dimensiones considerables. Además, las licencias de Ignition no tienen límite en cuanto a clientes, *tags*… no es necesario pagar más para aumentar la capacidad del sistema, la diferencia entre los distintos paquetes está en sus funcionalidades.

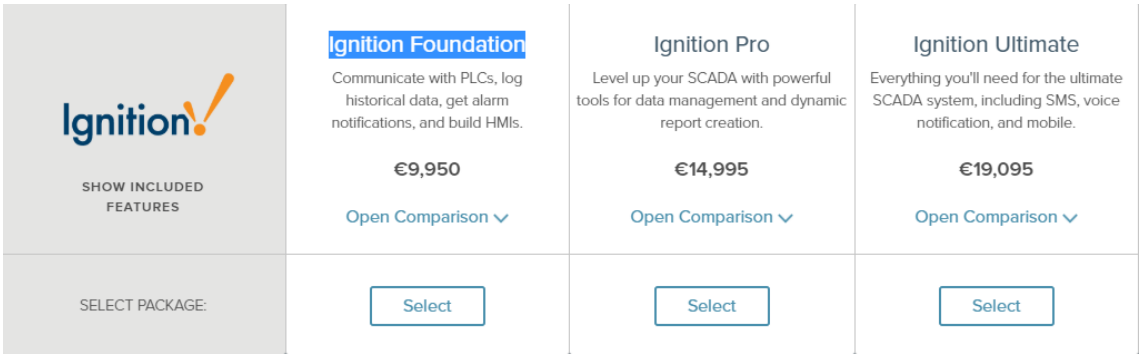


Figura 10. Precios de Igniton SCADA

En cuanto a la personalización, Ignition ofrece una gran flexibilidad, lo definen como una "pizarra en blanco", hace exactamente lo que el cliente necesita, como él quiera. Lógicamente esto requiere tener un gran conocimiento e invertir tiempo para poder desarrollar el sistema deseado desde cero, por eso Inductive Automation ofrece cursos de pago para enseñar a los clientes a manejar su software.

La arquitectura basada en Web hace posible que Ignition sea independiente de la plataforma. Puede correr en cualquier Sistema operativo: Windows, Linux, OS X, gracias a que está desarrollado 100% en Java.

5.5 Eclipse SCADA

Es un proyecto iniciado en 2013 con el objetivo de proporcionar un sistema SCADA de vanguardia y código abierto. Es la continuación del proyecto openSCADA, el cual se optó por migrar a la plataforma Eclipse para aumentar la visibilidad del proyecto y atraer a otras compañías y contribuyentes. Actualmente se utiliza 24/7 en varias instalaciones alrededor del mundo.

Figura 11. Eclipse SCADA es la continuación de OpenScada

El hecho de ser un software de código abierto le otorga gran flexibilidad. Eclipse SCADA proporciona librerías de desarrollo, herramientas de configuración masiva, aplicaciones back-end y front-end. Todas estas herramientas pueden ser combinadas de muchas formas diferentes, para diseñar una solución personalizada. La empresa desarrolladora, IBH SYSTEMS GmbH, también ofrece soluciones *out-of-the.box,* es decir, si algún cliente no quiere perder tiempo desarrollando solo tiene que ponerse en contacto con ellos para que le hagan el trabajo y conseguir una solución que se adapte a su sistema.

Es completamente gratuito y tiene la posibilidad de ser instalado en Windows, Linux y Mac. En cuanto al soporte, existe de una pequeña comunidad de usuarios repartida entre el foro de Eclipse, GitHub y Google Groups, que puede ayudar a solucionar las dudas.

5.6 IndigoSCADA

Se trata de un sistema SCADA de código abierto desarrollado en C y C++ y con soporte para Linux y Windows, completamente gratuito.

Una de sus principales características es que se trata de un sistema que ocupa muy poco espacio en el disco duro, ofreciendo las típicas funciones de los sistemas SCADA: alarmas, eventos, notificaciones, almacenamiento de datos, representación gráfica de los datos, programación de *scripts*…Además tiene un editor SQL para mantenimiento de datos online y a tiempo real.

Aunque este proyecto continúa en desarrollo a día de hoy, con actualización y mejoras frecuentes, no posee una gran comunidad a sus espaldas, lo cual hace difícil recibir ayuda de otros usuarios.

5.7 Criterio de selección

A continuación, se presentan los criterios que se han tenido en cuenta a la hora de elegir la alternativa más adecuada para el proyecto.

SOFTWARE LIBRE / CÓDIGO ABIERTO

Se ha valorado de forma positiva el software libre y de código abierto por las ventajas que suponen. Hay que aclarar que son conceptos parecidos pero diferentes: el *open source* es un paradigma de distribución y desarrollo, mientras que el libre se trata de un movimiento social. Un programa es software “libre” si los usuarios pueden ejercer las cuatro libertades esenciales: libertad para usar el programa con cualquier propósito; libertad para acceder al código fuente, estudiarlo y modificarlo; libertad para redistribuir gratuitamente copias del software; y libertad para distribuir gratuitamente versiones modificadas del software.

El movimiento del software libre cree que el mundo debería librarse del software de pago. El movimiento *open source*, por el contrario, es práctico y mucho menos político, busca el acceso al código fuente para poder modificarlo y compartirlo con la comunidad, no aboga por un tipo de licencia, sino que promueve un acercamiento más práctico.

Si el software engloba ambos conceptos, entonces se dice que es *Free and Open-Source Software* (FOSS). Este es el caso de Eclipse SCADA.

Entre las ventajas que ofrecen el software libre y el código abierto, se pueden destacar el precio económico o nulo, la libertad de utilizar, modificar y redistribuir el software, y el impulso a la colaboración entre la comunidad de usuarios.

PERSONALIZACIÓN

Muchas plataformas SCADA están montadas de la forma que ha decidido la empresa desarrolladora y no ofrecen a los usuarios la posibilidad de hacer modificaciones. Sabiendo esto, es lógico que no todos los softwares SCADA se adapten a las necesidades de cada cliente, muchas veces ocurre que no son capaces de hacer lo que el usuario desea. Si se necesita una nueva funcionalidad hay que pedírsela al desarrollador y esperar a que la implementen, si es que lo hacen. También puede ocurrir lo contrario, que un software incluya demasiadas funcionalidades sin la posibilidad de desactivarlas provocando un gasto innecesario de recursos.

En este contexto, un sistema SCADA con un alto grado de personalización gana muchos enteros respecto a otros más rígidos en este aspecto. Lógicamente, todos los SCADA *open source,* al estar disponible su código fuente para su libre modificación, se les supone una alta maleabilidad.

COMPATIBILIDAD con SO

Se ha valorado positivamente la compatibilidad con los sistemas operativos más utilizados hoy en día.

SOPORTE

Se trata de un aspecto clave a la hora de desarrollar un sistema SCADA, especialmente si se trata de uno con muchas opciones de personalización.

Se valora la existencia de foros de discusión, tutoriales, documentación, videos, etc. para guiar a los usuarios y solucionar los problemas que van apareciendo.

5.8 Selección de la solución

Para la elección del software más adecuado se ha utilizado una matriz de Pugh, en la que se han tenido en cuenta los criterios de selección explicados anteriormente. En esta matriz se utiliza Ignition SCADA como base, es decir, su puntuación va a ser la misma en todos los campos, y en función de esto se compara cada una de las siguientes alternativas con la base. La valoración será +1 si es mejor que el criterio base o -1 si es peor, 0 para el caso de ser similares.

		ALTERNATIVAS				
		Ignition	**WinCC**	**FactoryTalk**	**Eclipse SCADA**	**IndigoSCADA**
CRITERIO	**SW Libre/Código abierto**	=	0	0	1	1
	Personalización	=	-1	-1	0	0
	Compatibilidad con SO	=	-1	-1	0	0
	Soporte	=	0	0	0	-1
	TOTAL	=	-2	-2	1	0

Tabla 2. Matriz de Pugh de las alternativas

Puede observarse que, al sumar las puntuaciones, la alternativa que más puntos ha obtenido es Eclipse SCADA, por lo que será la solución adoptada para la realización del proyecto.

6 METODOLOGÍA

6.1 Introducción

El objetivo principal del proyecto es desarrollar un sistema SCADA utilizando la plataforma de código abierto Eclipse.

Tal y como se ha mencionado en el apartado de *Alcance*, en primer lugar, se instalará Eclipse junto con todos los complementos necesarios para crear proyectos de Eclipse SCADA, esta fase es la denominada *Puesta en marcha*.

A continuación, comienza la *Fase de pruebas*, en la que primeramente se configuran el *master server* y el *driver* para poder realizar una conexión entre el sistema SCADA y el PLC y después se desarrolla la *Visual Interface* (VI).

El proyecto se llevará a cabo en un PC con sistema operativo Windows 10 de 64 *bits*, aunque, como se ha visto en el apartado de *Análisis de Alternativas,* Eclipse SCADA es compatible con una amplia variedad de sistemas operativos.

6.2 Puesta en marcha

6.2.1 Instalación y configuración de JDK

Eclipse es una aplicación basada en Java, por lo que requiere como mínimo de un *Java Runtime Environment* (JRE) para funcionar. En este caso, como además se va a utilizar Eclipse para desarrollar en Java, no basta con instalar un JRE, sino que es necesario un *Java Development Kit* (JDK), el cual contiene el JRE anteriormente mencionado y componentes extra para desarrollar en Java.

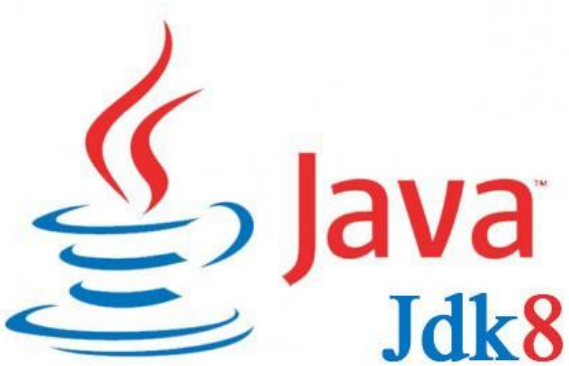


Figura 12. Logotipo de JDK8

A pesar de que en la página web de Eclipse SCADA se recomienda la versión JDK7, se ha utilizado la versión JDK8, ya que al utilizar JDK7 aparecían unos errores a la hora de crear la *Visual Interface* (VI) que hacían imposible avanzar en el proyecto. Su descarga, la cual es completamente gratuita, se realiza a través del siguiente link: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Después de la instalación es recomendable dirigirse al Panel de Control y realizar un par de ajustes en el JDK recién instalado, para evitar posibles errores relacionados con Java más adelante. En concreto, hay que añadir la carpeta *bin* del JDK a la lista de variables de entorno y crear una nueva variable de entorno denominada "JAVA_HOME", la cual debería apuntar al directorio de instalación del JDK.

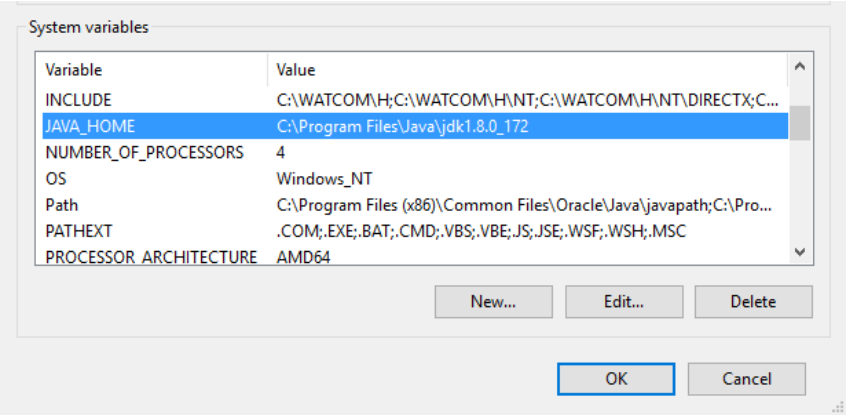


Figura 13. Configuración de JDK en el Panel de Control

6.2.2 Instalación y configuración de Eclipse

Para descargar Eclipse hay que dirigirse a la sección de descargas de su página web (https://www.eclipse.org/downloads/). Se puede descargar en dos formas distintas: en instalador o en archivo .zip. Se ha optado por esta última opción debido a su comodidad y sencillez, ya que únicamente hay que extraer el archivo descargado para tener Eclipse completamente funcional.

En cuanto a la versión de Eclipse, se ha optado por Eclipse Mars ya que es la que se recomienda en la web de Eclipse SCADA. Como se ha mencionado en el apartado de *Alcance*, hay que prestar atención a la compatibilidad entre la versión del JDK instalado en el paso anterior y la versión de Eclipse.

La primera vez que se ejecuta Eclipse el programa pide seleccionar una carpeta que haga de *workspace*, es decir, una carpeta para almacenar los proyectos. Es recomendable marcar la opción de utilizar la misma carpeta por defecto para que no aparezca este mismo cuadro de dialogo cada vez que se arranca Eclipse.

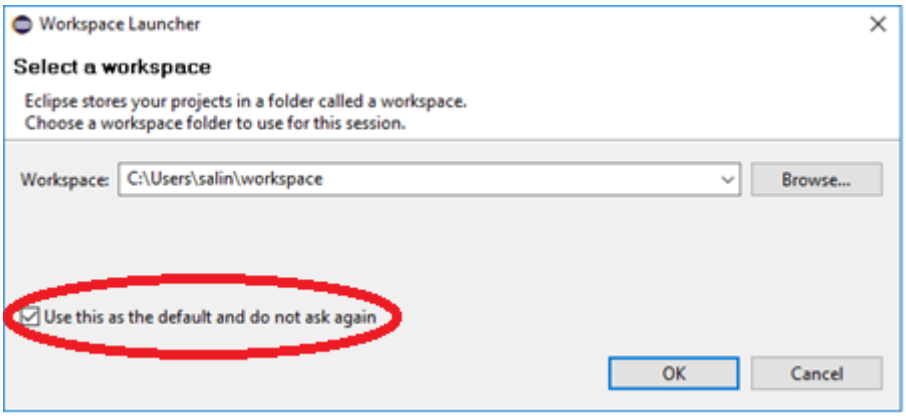


Figura 14. Selección del *workspace*

Una vez dentro de Eclipse, se accede a la configuración del programa clicando en "Preferences" para modificar ciertos parámetros a fin de reducir la posibilidad de errores y mejorar el funcionamiento general del problema.

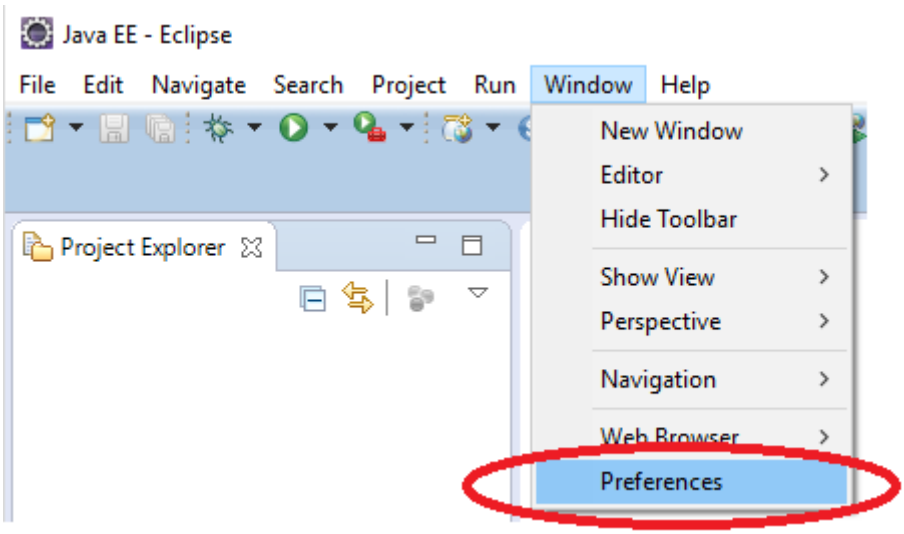


Figura 15. Accediendo a la configuración de Eclipse

Por ejemplo, es interesante marcar la opción "*Refresh using native hooks or polling*" para que el *workspace* de Eclipse se actualice y mantenga sincronizado automáticamente cuando se realizan cambios en los archivos del *workspace* de forma externa a Eclipse.

6.2.3 Instalación de complementos

El siguiente paso consiste en descargar e instalar los *plug-ins*, que contienen las componentes Eclipse SCADA necesarias para llevar a cabo el proyecto. Para ello, simplemente hay que dirigirse al apartado "*Install New Sofware*" en la barra de herramientas de Eclipse.

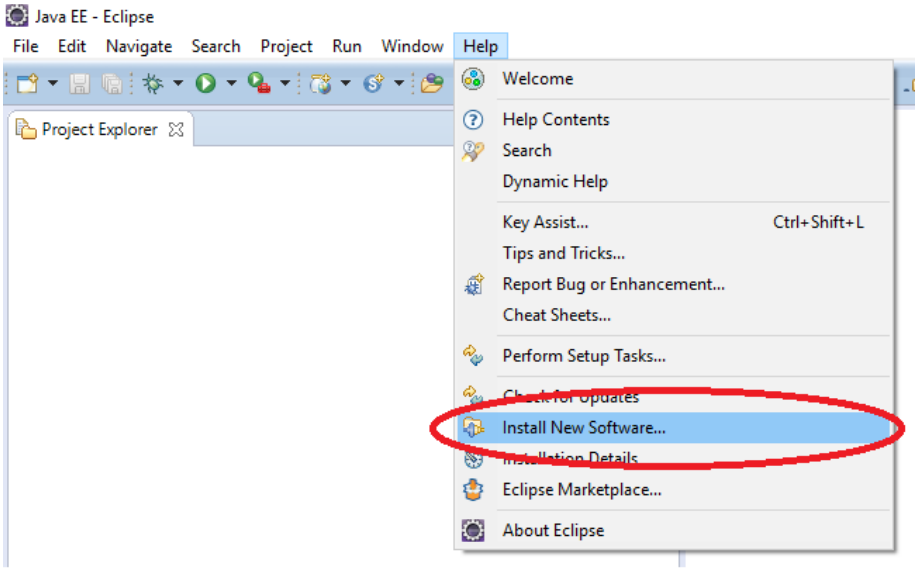


Figura 16. Descarga e instalación de *plug-ins*

Existen dos formas de instalar los *plug-ins*: indicando la dirección URL en la que se encuentran alojados o descargándolos en forma de archivo ZIP. En este caso se utilizará la primera forma ya que es mucho más cómoda, no es necesario descargar archivos que luego se quedan "perdidos" por el PC. Únicamente se introduce la URL, se seleccionan las componentes que se desean instalar de entre todas las que se alojan en dicha URL y Eclipse se encarga de descargar e instalarlas.

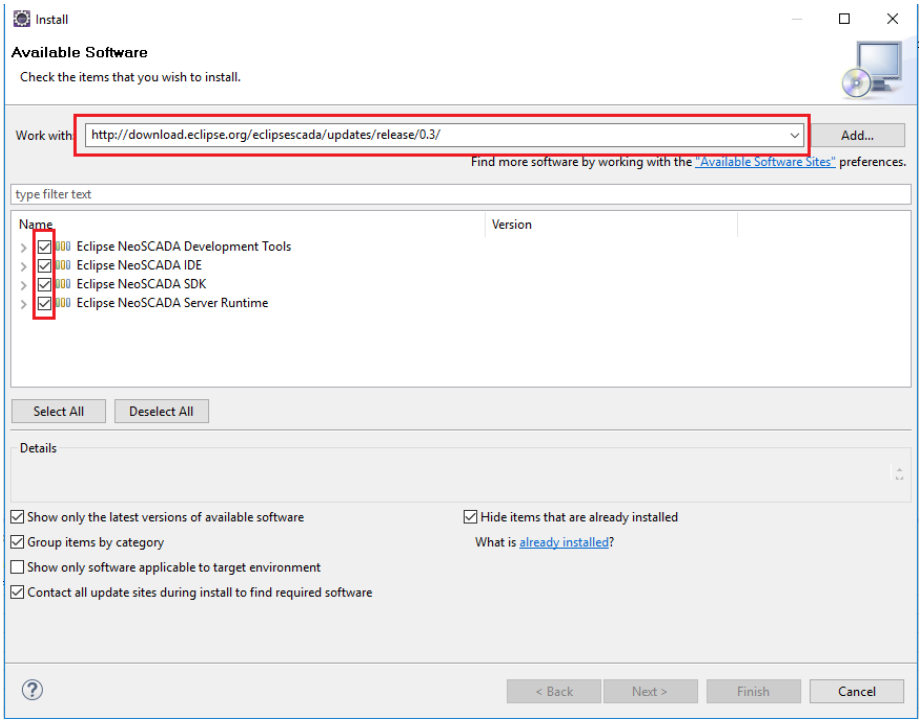


Figura 17. Selección de *plug-ins* a instalar

También es necesario descargar desde la página web de Eclipse SCADA el cliente *Eclipse* SCADA *Admin Client*, desde el cual se pueden monitorizar y modificar los datos de los dispositivos a los que se realiza la conexión.

6.2.4 Instalación de software Siemens

En las primeras pruebas se instaló TIA PORTAL V13 SP1 y la versión de PLCSIM que viene incluida en dicho software, pero no se logró establecer comunicación entre Eclipse SCADA y el PLC simulado.

Después de una búsqueda de información en internet se descubrió la existencia de PLCSIM Advanced, otro software de Siemens para simular PLCs pero más avanzado que PLCSIM, ya que soporta comunicación por TCP/IP. Se procede, por tanto, a instalar PLCSIM Advanced V1.0, aunque antes es necesario instalar TIA PORTAL V14 SP1, una versión más moderna que la instalada anteriormente, ya que PLCSim Advanced no soporta la V13, requiere versiones modernas de TIA PORTAL.

Ahora ya se tienen todos los programas necesarios instalados y configurados, y se puede dar por concluida la puesta en marcha.

6.3 Fase de pruebas

En general, un sistema Eclipse SCADA tiene 3 componentes principales:

- *Driver:* En la terminología de Eclipse SCADA el *driver* es el adaptador de protocolo, que se encarga de convertir del protocolo que se desea utilizar a NGP, el protocolo interno de Eclipse SCADA. Existen *drivers* para la comunicación con sistemas Modbus, S7, JDBC, SNMP, IEC 60870-5-104 y OPC. Se puede decir que el *driver* es el elemento que permite que el sistema se comunique, obtenga información y envíe órdenes a otros sistemas.
- *Master server*: Se encarga del almacenamiento y la gestión de datos y proporciona funcionalidades tales como Almacenamiento de Históricos o Alarmas y Eventos.
- Clientes: Acceden a los datos almacenados en el *master server* y se los muestran al usuario. En este caso, se utilizarán dos tipos de clientes: *Eclipse* SCADA *Admin Client* (ESAC) y la *Visual Interface* (VI).

6.3.1 *Master server* y *driver*

En primer lugar, se crea un nuevo proyecto dentro de Eclipse, que será del tipo "*Eclipse NeoSCADA Configuration Project*", al que se le dará el nombre de "*S7*".

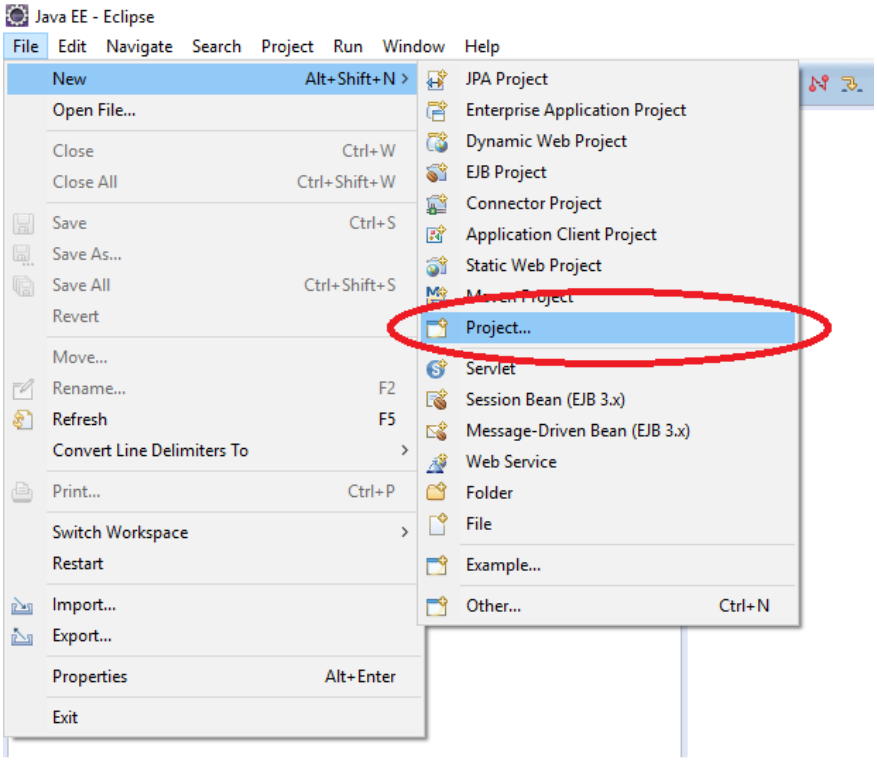


Figura 18. Creación de un proyecto dentro de Eclipse

Al hacer esto se crea dentro del *workspace* una carpeta con el nombre del proyecto elegido (S7), la cual se puede visualizar desde el explorador de Eclipse. Se puede comprobar como dentro de ella ya existen otras carpetas y elementos creados automáticamente, los cuales habrá que modificar para adaptarlos a este proyecto.

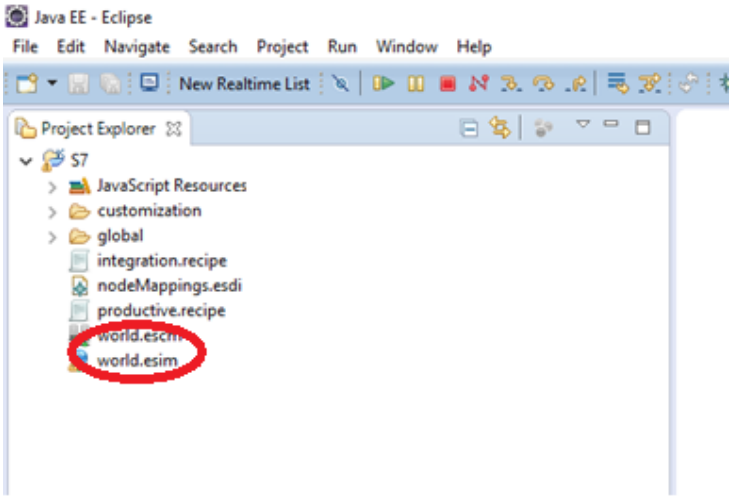


Figura 19. Carpeta que aparece al crear un nuevo proyecto de Eclipse SCADA

Para realizar la configuración de los *drivers* y el *master server* hay que modificar los archivos denominados "*world.escm"* y "*world.esim"*, los cuales significan *Eclipse SCADA Component Model* (ESCM) y *Eclipse SCADA Infrastructure Model* (ESIM) respectivamente.

Se abre el *Infrastructure Model (world.esim)* y se borran los dos *nodes* que vienen creados por defecto. Se crea un "*External Node"*, en cuyas propiedades se tiene que introducir la dirección IP del PLC. Esta dirección IP tiene que coincidir con la que más adelante se introducirá en PLCSIM Advanced y TIA PORTAL. En este caso se ha elegido la dirección 192.168.4.4. A continuación, dentro del "*External Node*" se añade el PLC, que será del tipo "*S7 Communication Processor*" y en cuyas propiedades hay que elegir el puerto 102, el que se usa para las comunicaciones de Siemens.

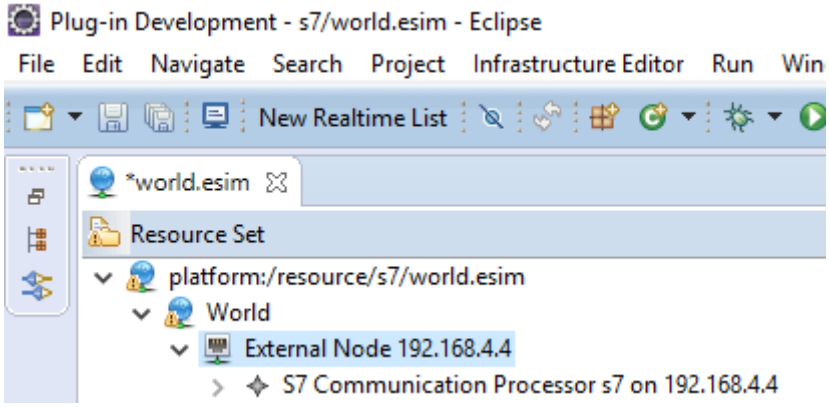


Figura 20. Creacion del *External Node*, que representa al PLC

El siguiente paso consiste en crear un "*Memory Type Model*" dentro del proyecto actual de Eclipse, un archivo en el que se van a definir los datos que Eclipse SCADA va a leer del PLC. En el caso de comunicación por S7, Eclipse SCADA solo es capaz de leer información almacenada en *Data Block*s (DB), los cuales habrá que crear al realizar el proyecto de TIA PORTAL. Dentro de este "*Memory Type Model"* se definen el tipo de datos que se quiere leer (booleanos, enteros, reales, etc.) y su dirección correspondiente dentro del DB.

Hasta ahora se ha estado definiendo como se realizan las consultas al PLC, ahora se va a elegir el *driver.* Para ello se regresa al archivo "*world.esim"*, se crea en su interior un "*System Node*", dentro del cual se añade un "*Equinox Driver*", que se debe configurar para adecuarlo a la comunicación por S7 tal y como se muestra en la *Figura 21* y según la información obtenida en la página web de Eclipse SCADA (https://wiki.eclipse.org/EclipseSCADA/Documentation/StandardPorts).

Devices	S7 Communication Processor s7 on 192.168.4.4
Driver Type Id	org.eclipse.scada.da.server.dave
Instance Number	10

Figura 21. Configuración del *Equinox Driver*

Por último, se abre el archivo "*world.escm*" y se borra el nivel denominado "*REGION1*", ya que hace referencia a los nodos que se han borrado antes. Se puede dar por concluida la configuración del *master server* y el *driver*.

Se hace clic derecho en el archivo "*productive.recipe*" y se selecciona "*Run Configurator*", lo cual creara dentro de la carpeta "*output*" otra carpeta denominada "*s7*", dentro de la cual se tiene el archivo "*s7.profile.xml*". Haciendo clic derecho y seleccionando "*Run Equinox Application Profile*" se pone en marcha el *master server*, al cual hay que conectarse más adelante usando el cliente ESAC.

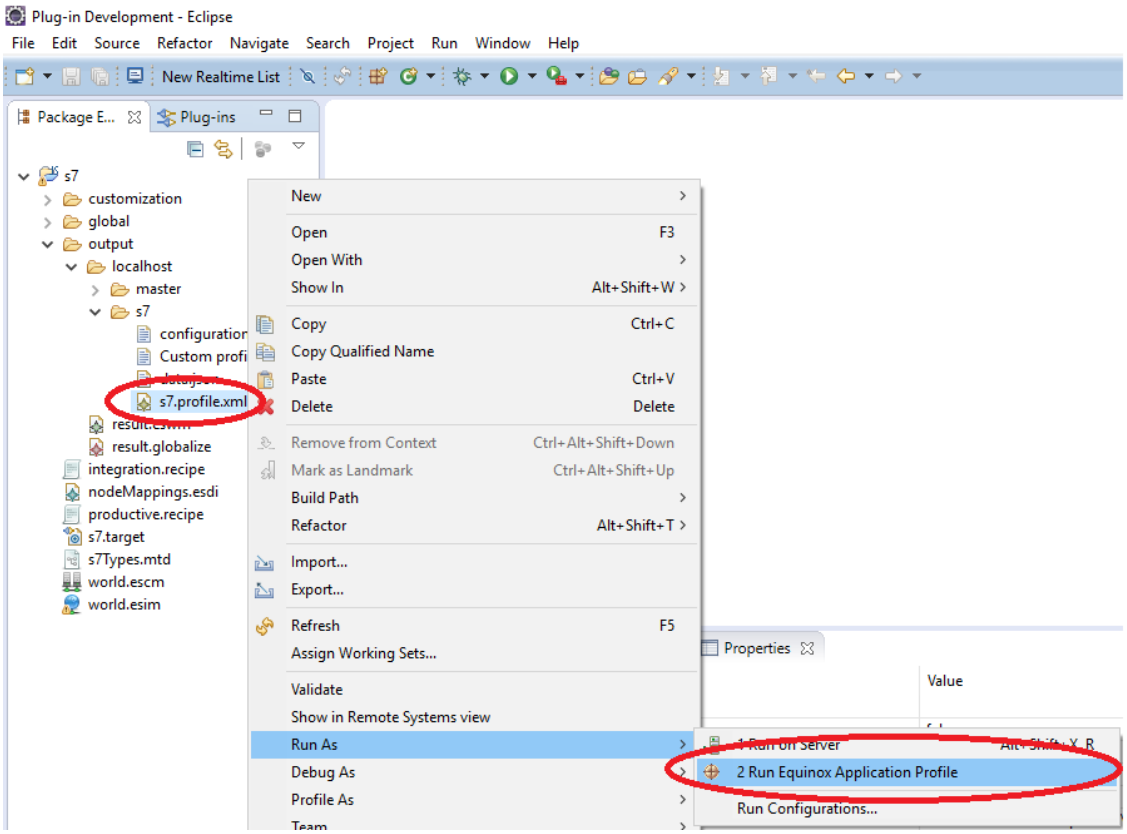


Figura 22. Arranque del *master server*

El siguiente paso será poner en marcha el PLC simulado. Se inicia PLCSIM Advanced, aparecerá un icono en la barra de tareas desde el cual se realiza todo el manejo del programa. Se escoge la opción de "*PLCSIM Virtual Eth. Adapter*" y se configuran los

parámetros (nombre, dirección IP y máscara de subred) para simular un PLC, que será obligatoriamente de la serie S7-1500, la única que soporta esta versión de PLCSIM Advanced.

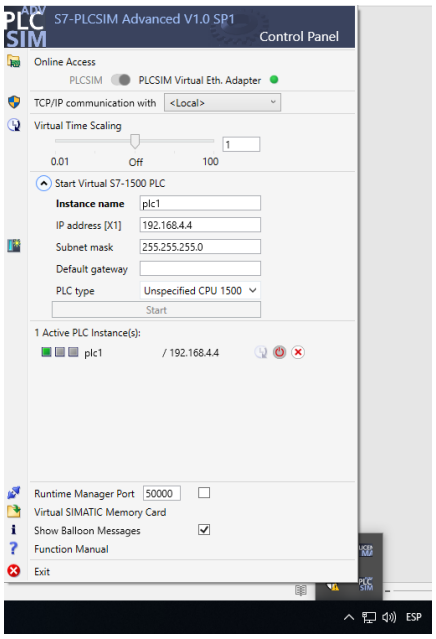


Figura 23. Puesta en marcha de PLCSIM Advanced

Por último, se inicia TIA PORTAL, se crea un nuevo proyecto eligiendo como controlador uno de la serie S7-1500 y modificando su dirección IP para que coincida con la elegida anteriormente. Se carga el proyecto dentro del PLC simulado y entonces ya debería existir conexión entre el sistema SCADA y el PLC.

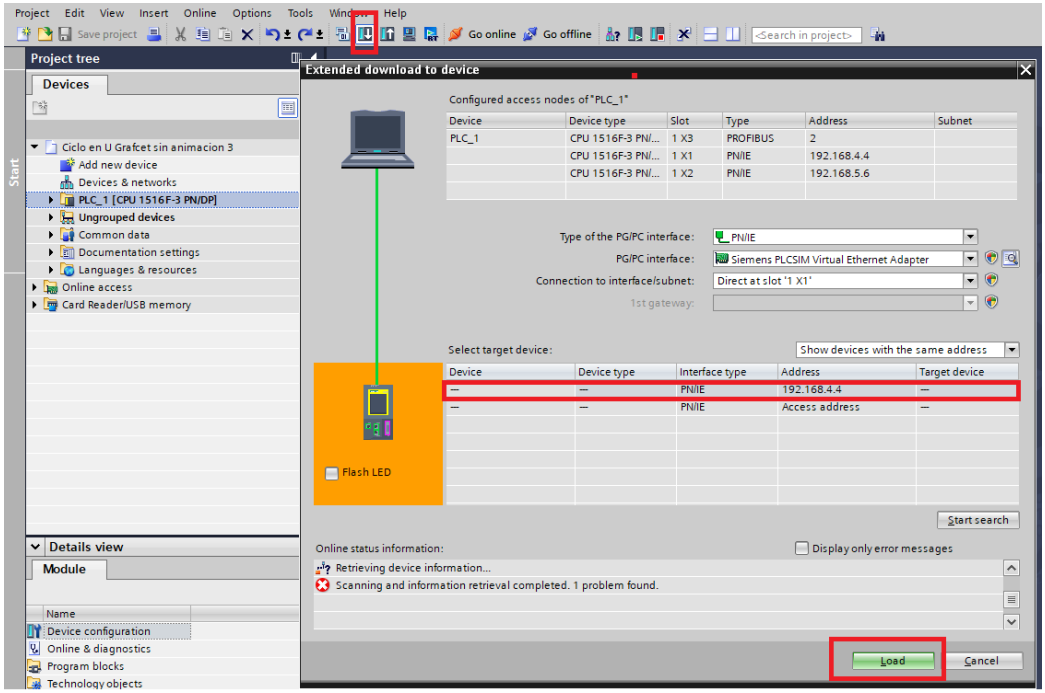


Figura 24. Carga del proyecto de TIA PORTAL en el PLC simulado

Se abre el cliente ESAC y se añade una nueva conexión utilizando el puerto 2110, el elegido anteriormente al configurar el *driver*. Después de añadirla, se hace doble clic en ella, se introduce como usuario “admin” y como contraseña “admin12” (valores que vienen así por defecto) y se habrá realizado con éxito la conexión al *master server*. Desde aquí se podrá leer la información almacenada en los DBs del PLC.

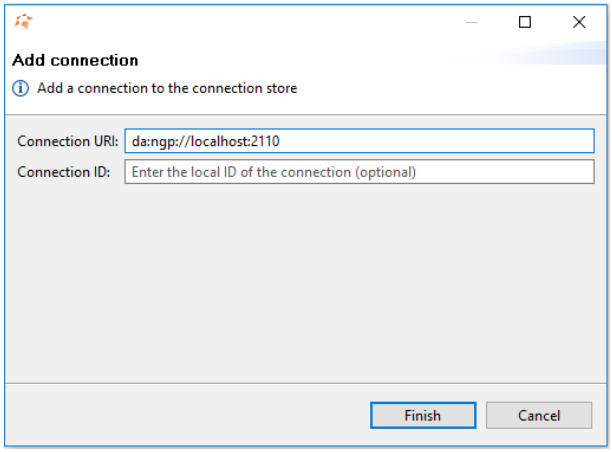


Figura 25. Añadir nueva conexión en ESAC

En la parte de la izquierda aparecen todas las variables configuradas anteriormente en el “*Memory Type Model*”. Simplemente hay que arrastrarlas a la parte de la derecha para poder recibir información sobre ellas y modificar su valor. En la *Figura 26* se observa un pequeño ejemplo en el cual se está accediendo al DB de un PLC para leer datos de tipo entero, los cuales pueden ser modificados bien desde TIA PORTAL o desde el cliente ESAC.

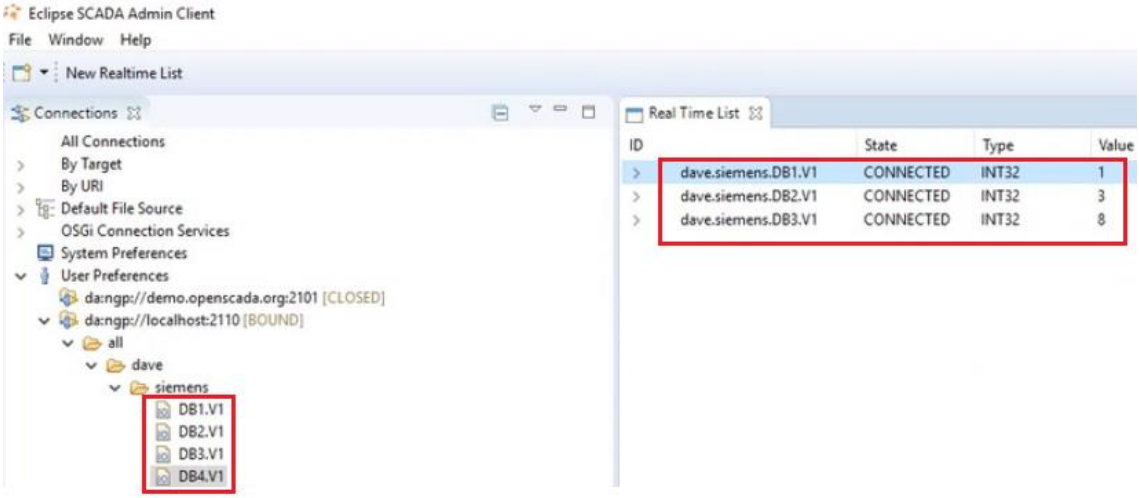


Figura 26. Monitorización y control de los datos del PLC desde ESAC

6.3.2 Visual Interface *(VI)*

Para desarrollar la VI, el primer paso es crear otro nuevo proyecto dentro de Eclipse, igual que se ha hecho anteriormente, pero esta vez será del tipo “Plug-in Project”.

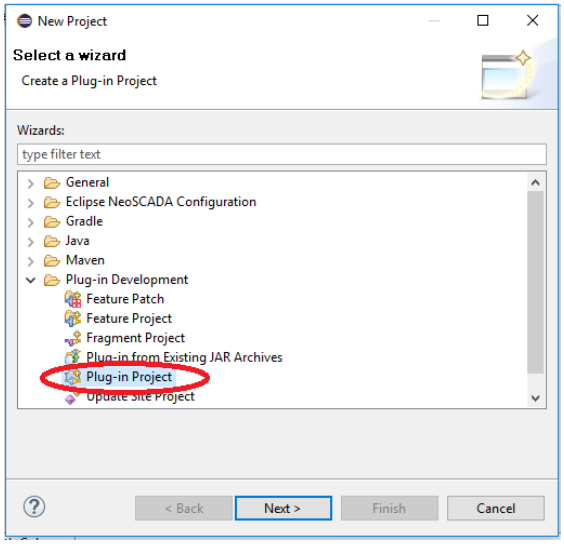


Figura 27. Elección del tipo de proyecto: *Plug-in Project*

Se elige un nombre para el proyecto, que en este caso será “manipulador”, se marca “Sí” en la opción de crear una aplicación de cliente pesado y se selecciona “*Eclipse NeoSCADA Client Application*” en la lista de *templates*, concluyendo así su creación. Esto creará 4 carpetas nuevas dentro del *workspace*, como se ve en la *Figura 28*.

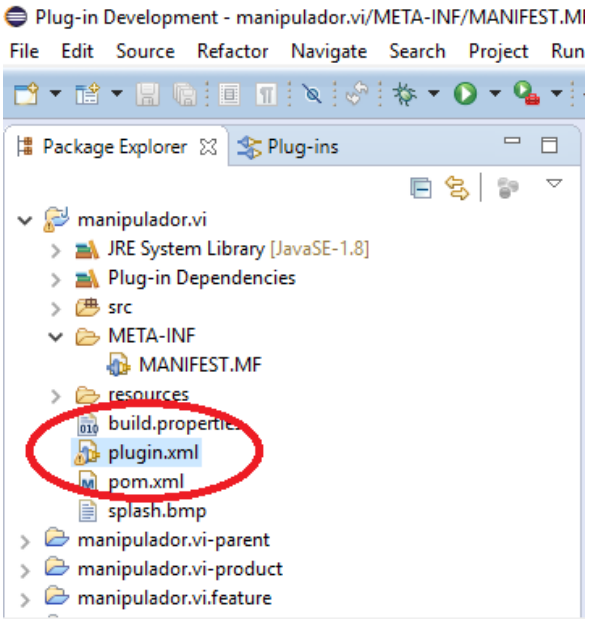


Figura 28. Carpetas que aparecen al crear un pryecto de VI

Hay que modificar el archivo “*plugin.xml*”, en concreto la línea 150, en la que hay que sustituir el número que allí aparece por 2110, el puerto que se ha elegido anteriormente al configurar el *master server*.

Figura 29. Modificación del archivo *plugin.xml*

Este archivo "*plugin.xml*" es el que hay que ejecutar haciendo clic con el botón derecho y seleccionando "*Run As Eclipse Application*" para arrancar la VI. Al igual que ocurre en ESAC, el usuario y la contraseña por defecto son "admin" y "admin12" respectivamente.

6.3.2.1 Símbolos

Eclipse SCADA organiza los elementos de visualización básicos (línea, rectángulo, triángulo, polígono…) dentro de símbolos para formar elementos más complejos. Los símbolos también pueden incluir imágenes bitmap (png, bmp, …) y referencias a otros símbolos.

El proceso para crear un nuevo símbolo es muy parecido al seguido para crear un proyecto, simplemente hay que seleccionar "Visual Interface Model". Los símbolos son archivos con formato "*vi*".

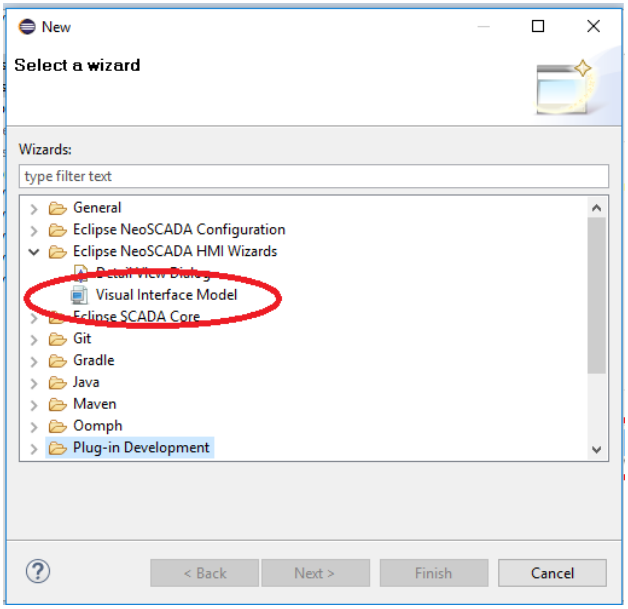


Figura 30. Creación de un símbolo

Los elementos básicos que se crean dentro de los símbolos tienen una serie de propiedades que hay que configurar porque de ellas depende como se van a ver al ejecutar la VI, como pueden ser el tamaño, la posición o el color.

6.3.2.2 Contenido dinámico

Eclipse SCADA no se limita a mostrar los símbolos de forma estática, también permite añadir comportamientos dinámicos a partes específicas de la aplicación. Para ello se incluirán archivos JavaScript junto a los símbolos.

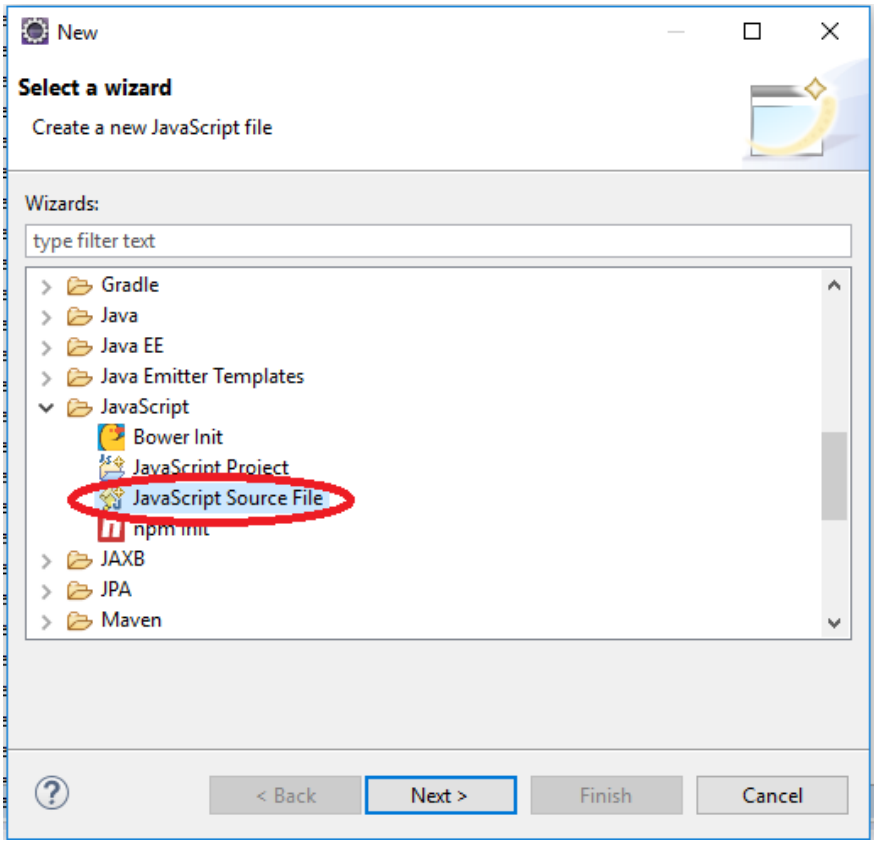


Figura 31. Creación de un archivo JavaScript

Estos *scripts* no solo son capaces de modificar las propiedades de los elementos de visualización, sino que también pueden acceder a los DBs para leer e incluso modificar la información allí almacenada.

6.4 Validación de la solución

Teniendo en cuenta todo lo explicado anteriormente, se ha desarrollado una plataforma Eclipse SCADA para monitorizar y controlar un pequeño manipulador neumático. La fase de configuración del *master server* y el *driver* es análoga a lo mencionado unas páginas atrás, aunque en este caso hay que crear unas cuantas variables dentro de un DB para que puedan ser leídas por Eclipse SCADA, ya que no tiene la capacidad de leer *tags*.

Data_block_2

	Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Static								
2	1	Bool	0.0	false					
3	2	Bool	0.1	false					
4	3	Bool	0.2	false					
5	4	Bool	0.3	false					
6	5	Bool	0.4	false					
7	6	Bool	0.5	false					
8	7	Bool	0.6	false					
9	8	Bool	0.7	false					
10	C1S1	Bool	1.0	false					
11	C1S2	Bool	1.1	false					
12	C2S1	Bool	1.2	false					
13	C2S2	Bool	1.3	false					
14	C3M1	Bool	1.4	false					
15	PM	Bool	1.5	false					
16	PP	Bool	1.6	false					
17	Pieza	Bool	1.7	false					
18	C1M1	Bool	2.0	false					
19	C1M2	Bool	2.1	false					
20	C2M1	Bool	2.2	false					
21	C2M2	Bool	2.3	false					

Figura 32. Variables creadas dentro del DB

En cuanto a la VI, se ha utilizado un elevado número de símbolos para representar con claridad todos los elementos del proceso. También se han configurado *scripts* para intentar que el manipulador se comporte de forma más realista posible.

Hay que destacar el pulsador de marcha-paro, el cual permite detener y reanudar el proceso desde la misma VI.

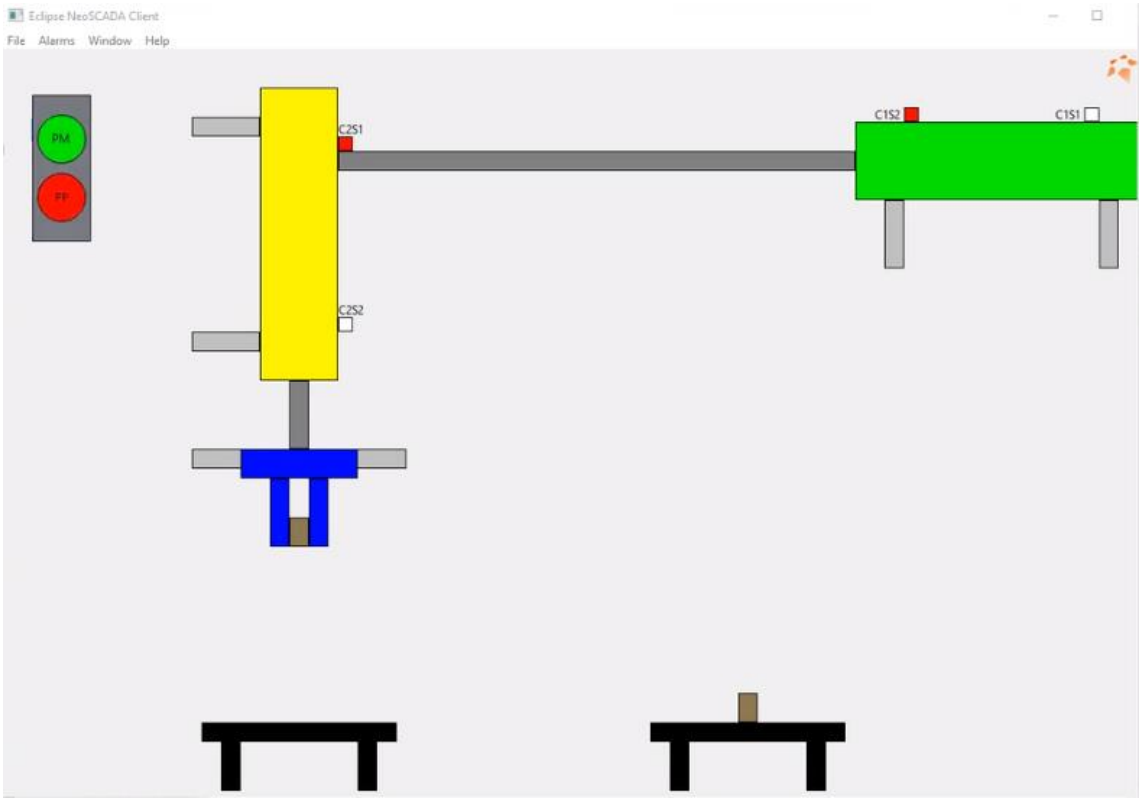


Figura 33. VI para visualizar y controlar el manipulador

7 DESCRIPCION DE TAREAS. GANNT

El desarrollo de este Trabajo de Fin de Grado se puede dividir en cinco fases:

1. ANALISIS DE ALTERNATIVAS

En esta fase se estudian las distintas opciones que se disponen en cuanto a software SCADA para poder escoger la solución que más se adecúa a las necesidades del proyecto. Esta fase tiene una duración de 14 días laborables.

2. DISEÑO SISTEMA

En un primer lugar se realiza una investigación en la página web de Eclipse SCADA con el objetivo de elegir de entre todos los protocolos de comunicación que soporta el más adecuado para este proyecto.

Una vez decidido que se realizará un proyecto de comunicación con S7 se realiza el diseño del sistema. En la *Figura 4* se puede ver un esquema del mismo. Hay que tener en cuenta que el diseño inicial puede sufrir modificaciones ya que la fase de diseño se realiza de forma paralela al desarrollo, y en el caso de encontrar algún problema durante el mismo es posible que sea necesario realizar cambios en el diseño. Por ejemplo, en este caso en el diseño inicial se utilizó PLCSIM como software para simular el PLC, pero luego resultó imposible establecer comunicación entre el PLC simulado y Eclipse SCADA, por lo que se sustituyó PLCSIM por PLCSIM Advanced.

Esta fase tiene una duración de 28 días laborables.

3. DESARROLLO SISTEMA

El primer paso es instalar y configurar el programa Eclipse y todos los complementos necesarios (JDK, *plug-ins* de Eclipse Scada, ESAC y software de Siemens) en el PC. Una vez esté todo listo se comienza la fase de pruebas en la que primero se configuran el *driver* y el *master server* y después se desarrolla la *Visual Interface* (VI). Se va probando hasta dar con un sistema válido. Esta fase tiene una duración de 80 días laborables.

4. DOCUMENTACIÓN

Las tareas de esta etapa se inician ya en la primera fase, pero sólo finalizan una vez terminadas las pruebas. Esta fase tiene una duración de 108 días laborables.

5. REDACCION DE LA MEMORIA

El Trabajo de Fin de Grado concluye con la redacción de la memoria del proyecto, una vez ha finalizado la fase práctica en la que se ha puesto en marcha la plataforma objeto de diseño. Esta fase tiene una duración de 30 días laborables.

El proyecto comenzó el día 5 de febrero de 2018 y acabó el día 20 de julio de 2018.

El Trabajo de Fin de Grado lleva 120 días para su desarrollo que suponen en total 180 horas de trabajo suponiendo una carga de 1,5 horas por día.

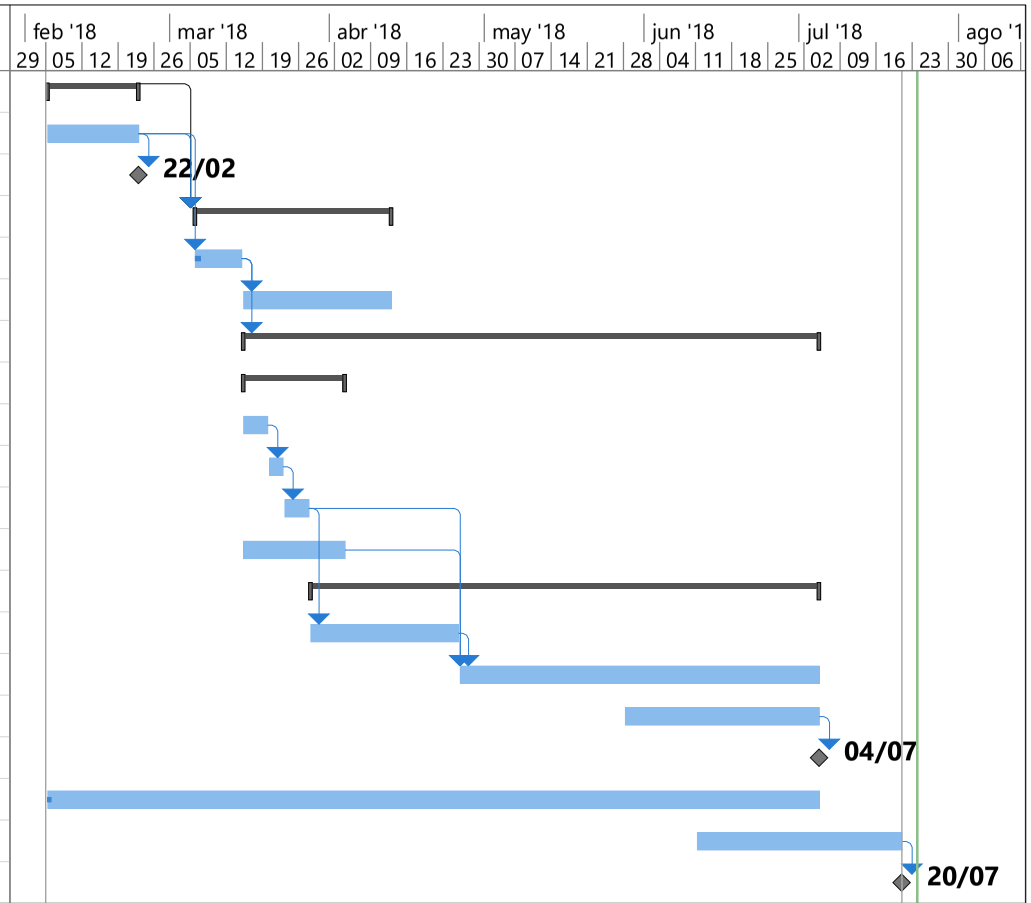
Los hitos del proyecto se reflejan en la siguiente tabla:

HITOS DEL PROYECTO
H.1.1 Solución escogida
H.3.1 Sistema funcionando
H.5.1 Fin del trabajo de fin de grado

Tabla 3. Hitos del proyecto

Estos hitos, así como todas las fases del proyecto, se representan en el diagrama de Gantt que se adjunta en la siguiente página

Id	EDT	Nombre de tarea	Duración	Comienzo
1	**P.T.1**	**Análisis de alternativas**	**14 días**	**lun 05/02/18**
2	T.1.1	Alternativas de software SCADA	14 días	lun 05/02/18
3	H.1.1	Solución escogida	0 días	jue 22/02/18
4	**P.T.2**	**Diseño sistema**	**28 días**	**mar 06/03/18**
5	T.2.1	Elección protocolo comunicación	7 días	mar 06/03/18
6	T.2.2	Validación diseño	21 días	jue 15/03/18
7	**P.T.3**	**Desarrollo sistema**	**80 días**	**jue 15/03/18**
8	**T.3.1**	**Puesta en marcha**	**14 días**	**jue 15/03/18**
9	T.3.1.1	Instalación de JDK	3 días	jue 15/03/18
10	T.3.1.2	Instalación de Eclipse	3 días	mar 20/03/18
11	T.3.1.3	Instalación de complementos	3 días	vie 23/03/18
12	T.3.1.4	Instalación software Siemens	14 días	jue 15/03/18
13	**T.3.2**	**Fase de pruebas**	**71 días**	**mié 28/03/18**
14	T.3.2.1	*Master server y driver*	21 días	mié 28/03/18
15	T.3.2.2	*Visual Interface*	50 días	jue 26/04/18
16	T.3.3	Validación sistema	28 días	lun 28/05/18
17	H.3.1	Sistema funcionando	0 días	mié 04/07/18
18	**P.T.4**	**Documentación**	**108 días**	**lun 05/02/18**
19	**P.T.5**	**Redacción de la memoria**	**30 días**	**lun 11/06/18**
20	H.5.1	Fin del Trabajo de Fin de Grado	0 días	vie 20/07/18



Proyecto: TFG
Fecha: mar 24/07/18

Tarea	Resumen inactivo	Tareas externas
División	Tarea manual	Hito externo
Hito	solo duración	Fecha límite
Resumen	Informe de resumen manual	Progreso
Resumen del proyecto	Resumen manual	Progreso manual
Tarea inactiva	solo el comienzo	
Hito inactivo	solo fin	

8 PRESUPUESTO

A la hora de desarrollar el presupuesto del Trabajo de Fin de Grado, éste se ha dividido en tres partidas: horas internas, amortizaciones y gastos. Por último, se incluye un resumen de las tres partidas con el coste total del proyecto.

A continuación, se muestran las tablas correspondientes a las distintas partidas:

HORAS INTERNAS			
Concepto	Coste unitario (€/h)	Nº unidades (h)	Coste
Ingeniero técnico	30	180	5.400,00 €
		TOTAL	5.400,00 €

Tabla 4. Horas internas

AMORTIZACIONES				
Concepto	Precio adquisición	Vida útil (años)	Utilización (h)	Coste
Ordenador	900,00 €	5	150	20,77 €
			TOTAL	20,77 €

Tabla 5. Amortizaciones

GASTOS			
Concepto	Coste unitario	Nº unidades	Coste
Licencia TIA PORTAL V14	1.417,50 €	1	1.417,50 €
Licencia PLCSIM Advanced V1.0	2.000,00 €	1	2.000,00 €
Licencia Windows 10	259,00 €	1	259,00 €
		TOTAL	3.676,50 €

Tabla 6: Gastos

En la partida de amortizaciones se ha supuesto un uso del ordenador de 1300 horas al año. Cabe mencionar que la licencia de TIA PORTAL, PLCSIM Advanced y Windows 10 se incluyen como gasto ya que no tienen caducidad.

El coste total del proyecto es de 9370,19 €. Este coste se desglosa en horas internas, amortizaciones y gastos, y aparece representado en la *Tabla 7*.

RESUMEN	
Horas internas	5.400,00 €
Amortizaciones	20,77 €
Gastos	3.676,50 €
SUBTOTAL	9.097,27 €
Costes indirectos (3%)	272,92 €
TOTAL	9.370,19 €

Tabla 7: Coste total

A la hora de calcular el coste final se han sumado un 3% de costes indirectos no imputables al proyecto, como por ejemplo la electricidad.

En el siguiente gráfico se puede observar el peso de cada partida en el presupuesto:

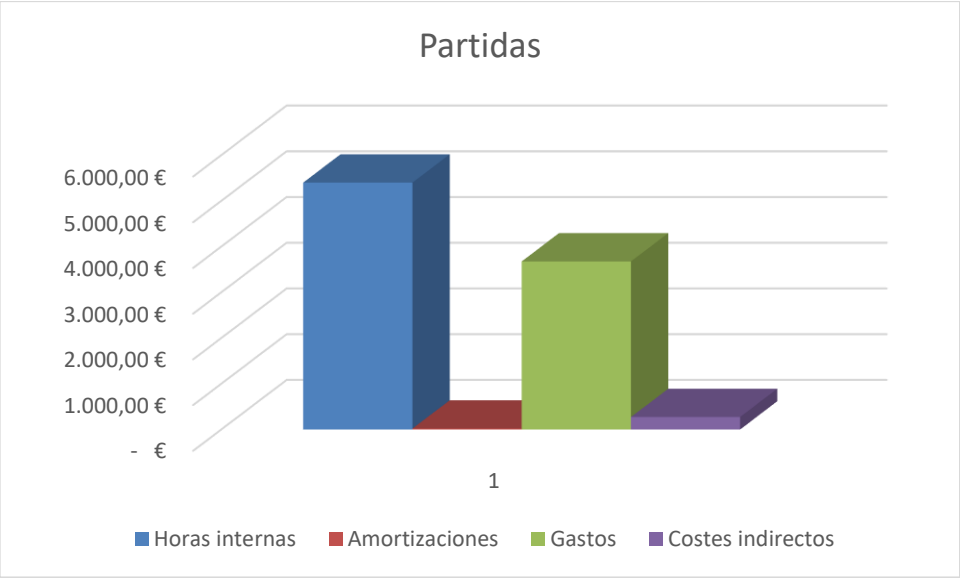


Figura 34. Partidas del presupuesto

La partida con más peso sobre el total del presupuesto es la correspondiente a las horas internas, ya que este trabajo ha requerido una cantidad considerable de horas de investigación, aprendizaje, diseño y desarrollo.

En segundo lugar, se encuentra la partida correspondiente a los gastos, en la que se incluye la compra de las licencias. El coste correspondiente a este apartado sería mucho mayor si se hubiera utilizado software SCADA de pago, como se ha visto en el apartado de *Análisis de alternativas*.

En tercer y cuarto lugar se encuentran los costes indirectos y las amortizaciones, con una baja contribución al importe final del proyecto.

9 CONCLUSIONES

A lo largo de este documento ha quedado reflejada la utilidad de una plataforma SCADA basada en Eclipse como alternativa al resto de sistemas SCADA que se encuentran en el mercado, ya que constituye una plataforma de bajo coste.

La plataforma se basa en el uso de software libre, lo cual abre las puertas a que la gente utilice, modifique, adapte, distribuya y, en general, mejore las posibilidades del programa. Además, no se requieren grandes conocimientos técnicos para ponerla en marcha, únicamente es necesario saber programar en JavaScript para realizar los *scripts* y tener suficiente nivel de inglés escrito para seguir los tutoriales de la web, preguntar las dudas que vayan surgiendo en los foros, etc.

Otra ventaja de esta plataforma es su compatibilidad con distintos protocolos de comunicación, tanto abiertos como propietarios, lo cual otorga la posibilidad de combinar equipos de distintos fabricantes dentro de un mismo sistema.

Una plataforma SCADA basada en Eclipse puede ser de gran utilidad en el campo de la docencia, donde puede servir para enseñar a los alumnos de forma económica el funcionamiento y la utilidad de los sistemas SCADA.

10 FUENTES DE INFORMACIÓN

[1] *openSCADA – Google Groups* [en línea]. Google Inc. Disponible en:<https://groups.google.com/forum/#!forum/openscada> [Última consulta: 23 de abril de 2018]

[2] *Eclipse Community Forums* [en línea]. Eclipse Foundation, Inc. Disponible en: <https://www.eclipse.org/forums/index.php/i/> [Última consulta: 5 de mayo de 2018]

[3] Jens Reimann, Jürgen Rose. *Eclipse SCADA* [en línea]. 2015. Disponible en: <http://book.openscada.org/>

[4] Ben Schneider, Georg Neugschwandtner. *Visualising a Fluid Level Control Loop: Eclipse SCADA and 4DIAC* [en línea]. 2015. Disponible en: <http://download.fortiss.org/public/IA/FLCDemo/flc_demo_doc_20150420.pdf>

[5] *Eclipse SCADA* [en línea]. IBH SYSTEMS GmbH and others. 2018. Disponible en: <https://www.eclipse.org/eclipsescada/>

[6] *EclipseSCADA – Eclipsepedia* [en línea]. Eclipse Foundation, Inc. Disponible en: <https://wiki.eclipse.org/EclipseSCADA>

[7] *What is SCADA? Supervisory Control and Data Acquisition* [en línea]. Inductive Automation. 2016. Disponible en: <https://inductiveautomation.com/what-is-scada>

[8] *Jürgen Rose – YouTube* [en línea]. YouTube, LLC. Disponible en: <https://www.youtube.com/user/CptMauli>

[9] *GitHub - eclipse/neoscada: Eclipse NeoSCADA™* [en línea]. GitHub, Inc. 2018. Disponible en: <*https://github.com/eclipse/neoscada*>

[10] Matt Mccallum. *Ignition Versus: FactoryTalk View – Licensing and Training Costs – Perfect Abstractions Blog* [en línea]. 2017. Disponible en: <https://blog.perfectabstractions.com/2017/03/13/ignition-versus-factorytalk-view-licensing-and-training-costs/>

[11] *Software-in-the-loop* [en línea]. OPAL-RT TECHNOLOGIES, Inc. 2018. Disponible en: <https://www.opal-rt.com/software-in-the-loop/>

[12] *Software-in-the-loop testing applications* [en línea]. add2. 2018. Disponible en: <http://www.add2.co.uk/applications/sil/>

[13] *Enscada's IndigoSCADA Section* [en línea]. Enscada. 2014. Disponible en: <http://www.enscada.com/a7khg9/IndigoSCADA.html>

[14] Gordon Clarke, Deon Reynders. *Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems.* Elsevier, 2004. ISBN: 0080480241.

ANEXOS

Anexo 1: *Time Trigger*

Uno de los problemas que se encontraron durante la *Fase de pruebas* fue la incapacidad de Eclipse SCADA para retrasar *scripts*, es decir, hacer que una función de JavaScript se ejecute después de un cierto periodo de tiempo. Esto es algo imprescindible para simular el comportamiento del manipulador en la VI, ya que entre unos movimientos y otros tiene que transcurrir un tiempo, no puede suceder todo a la vez, lógicamente.

Se intentó vencer esta problemática de varias formas. En primer lugar, se probó a utilizar la función *setTimeout*, una función de JavaScript que ejecuta la función que se le indique después de transcurrir un determinado periodo de tiempo. Esto en teoría debería servir para lograr el objetivo, pero por algún motivo la función nunca se llegaba a ejecutar.

También se probó a utilizar *java.lang.Thread.sleep*, que sirve para suspender la ejecución del hilo principal durante un determinado número de milisegundos. En este caso el problema era que el cliente se congelaba durante la ejecución de esta función.

Se contactó con uno de los desarrolladores de Eclipse SCADA en busca de respuestas y soluciones, explicándole lo que se pretendía lograr. Éste tomó nota de la problemática y la anotó en GitHub con el objetivo de desarrollar una solución en un corto periodo de tiempo.

En un par de semanas este desarrollador añadió una nueva funcionalidad: el *Time Trigger*. Este elemento sirve para ejecutar una función JavaScript con un periodo de tiempo a partir del momento en el que se inicia la VI. Por ejemplo, si se crea un *Time Trigger* con un periodo de 13 segundos y se le asigna la función “prueba.js”, el *Time Trigger* provocará que esa función se ejecute cada 13 segundos desde el momento en el que se carga la VI. No es exactamente lo que se buscaba, pero sirve para avanzar un paso más en el proyecto.

Lo que se ha hecho para aprovechar esta nueva funcionalidad es asignar a cada función JavaScript un *Time Trigger* con un periodo de 1000ms, lo que provocaría que cada segundo se ejecutaran todas las funciones. Después hay que limitar de algún modo qué funciones interesa que se ejecuten en cada instante, porque de lo contrario se ejecutarían todas a la vez. Para ello se han creado unas variables llamadas “Posición 1”, “Posición 2”, etc., que tendrán el valor de *true* cuando la garra se encuentre en dicha posición. Se limita la ejecución las funciones mediante condicionales para que solo se ejecuten cuando la variable de la posición anterior está en *true*.

Por ejemplo: supóngase que la garra está en la posición 1, es decir, la variable “Posición 1” está en *true* y el resto de variables “Posicion X” están en *false*. Transcurre 1s y las únicas funciones que se ejecutan son las que mueven los distintos elementos a la posición 2, el resto de *scripts* no se ejecutan porque están limitadas por sentencias *IFs.*

Ahora la garra se ha movido en la posición 2, por lo que la única variable en *true* es “Posición 2”. Al transcurrir 1s se ejecutan solo los *scripts* que mueven el sistema a la posición 3… y así sucesivamente.

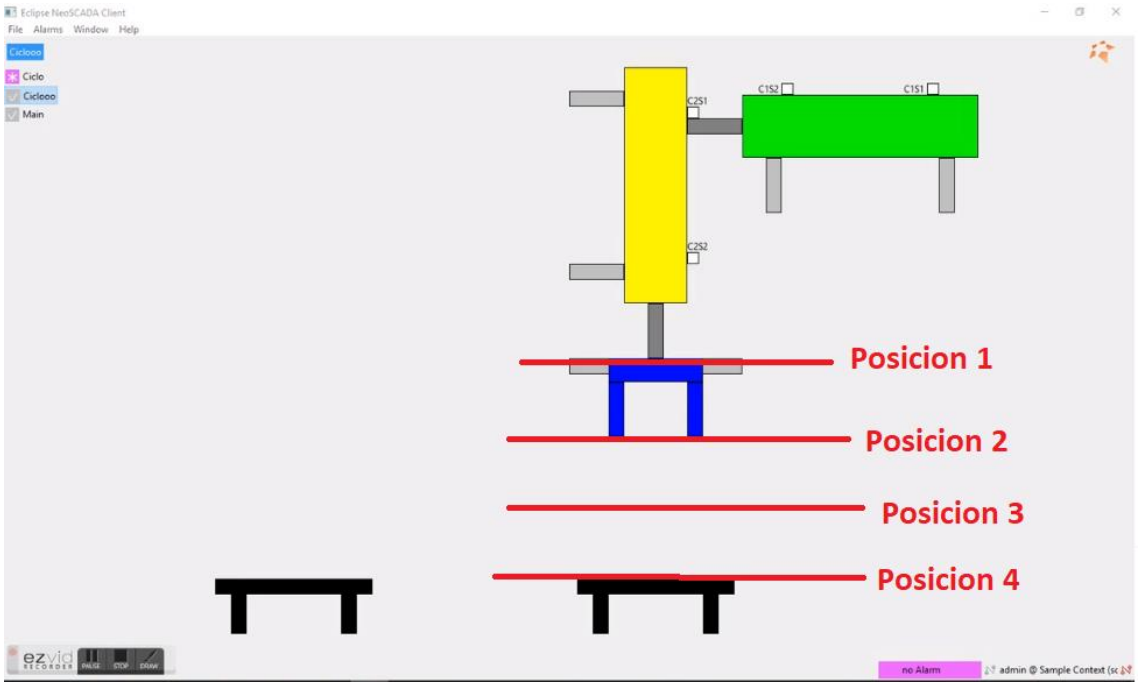


Figura 35. Posiciones para controlar los Time Triggers

Como se ha mencionado antes, los *Time Trigger* no son la solución perfecta, hay muchos fallos y limitaciones en su utilización que saltan a la vista, pero han permitido avanzar un poco más en el diseño del sistema SCADA y verificar varias de las características que por las que se ha optado por este software, como son el hecho de tener una buena comunidad de usuarios y la posibilidad de añadir nuevas funcionalidades rápidamente.

Cabe mencionar que para poder utilizar esta nueva funcionalidad ha sido necesario utilizar otra versión de Eclipse más moderna, Eclipse Oxygen, ya que las *nightly builds* de Eclipse SCADA que se suben a *GitHub* solo funcionan con esa versión.