UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO

TRABAJO FIN DE GRADO

COMUNICACIÓN DE UN PLC SIEMENS CON UNA BASE DE DATOS Y APLICACIÓN ANDROID

ALUMNO:

MARCO ANTONIO YONFÁ URUCHIMA

ESPECIALIDAD:

ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DIRECTOR ACADÉMICO:

D. RICARDO PIZÁ FERNANDEZ

ÍNDICE

1. MEMORIA

1.1. Justificación del proyecto

En un mundo industrial donde la presencia de sistemas automatizados está cada vez más presente, y el mantenimiento de dichos sistemas cada vez se vuelve más complejo, es importante que el operario disponga de herramientas para su correcto mantenimiento. Por todo esto, se decidió iniciar una idea que facilitara al operario el mantenimiento de los sistemas automatizados.

Como núcleo de estos sistemas se encuentra el PLC (Programmable Logic Controller), que es el encargado de dirigir las operaciones en un sistema automático, si alguna entrada o salida deja de funcionar pueden empezar los problemas de funcionamiento de todo el sistema. Por ello surgió uno de los conceptos fundamentales del proyecto, que se debe al registro de fallos en una base de datos donde se va recopilando los incidentes ocasionados en un PLC con alguna de sus entradas o salidas.

A partir de esta idea, una vez registrado el fallo en la base de datos, dicha base se compararía con otra base de datos donde se encontrarían registradas las posibles soluciones, éstas soluciones junto al posible fallo, se enviarían al operario a través de una aplicación Android, para así ayudar al operario a un ágil y correcto mantenimiento.

1.2. Objeto del proyecto

La empresa Ogénica Ingeniería SL pide la creación de un sistema de comunicación entre un PLC Siemens S7-1200 y un operario, para que el operario tenga un diagnóstico previo de un posible fallo provocado por el PLC.

Para establecer dicha comunicación, se creará un servidor TCP/IP mediante lenguaje PHP debido a su versatilidad en este tipo de comunicaciones. Este servidor hará de nexo entre las comunicaciones del PLC con la base de datos y la aplicación Android.

Para la comunicación del PLC y la base de datos se hará uso del software oficial de Siemens, TIA Portal, donde se programará con lenguaje KOP (esquema de contactos), el bloque de comunicación TCP/IP con el que la tarjeta PROFINET se comunicará con el servidor PHP.

Por otro lado, en la base de datos se utilizará el programa MySQL para configurar las tablas necesarias que se encargaran de registrar los fallos, que además proporcionan una solución y la comunicación con el servidor.

Con respecto a la aplicación Android, se hará uso del programa Android Studio, con Java como fuente de programación se creará una aplicación capaz de recibir datos mediante conexión TCP/IP del servidor PHP, con el que el operario será capaz de visualizar el posible error del PLC.

1.3. Estudios de necesidades

1.3.1. Condiciones de encargo

El cliente ha encargado la creación de un sistema de comunicación entre un PLC Siemens y una base de datos donde pueda registrar y almacenar los posibles errores que vayan surgiendo de las entradas y salidas del PLC. Además, mostrar dichos errores a una aplicación Android para que el operario sea capaz de recibir un aviso de inmediato, definiendo varios factores a tener en cuenta:

- Se trabajará únicamente con el PLC S7-1200 de la marca Siemens.
- Ya que es un PLC de la casa Siemens, se utilizará el programa oficial TIA Portal como software de programación, que al igual que el PLC, lo ofrece el contratista.
- MySQL será el servidor de la base de datos con el que se deberá trabajar propiedad de la empresa.
- Será necesario trabajar con software de código abierto en el resto de apartados.
- El sistema de codificación del error, ha de ser lo suficientemente claro para que el operario sea capaz de identificar el fallo.
- Ha de ser un sistema preciso y eficaz.
- Cuando se registre el fallo, deberá registrarse la fecha y hora del momento exacto.

1.3.2. Estudios de legislación

Se tendrá en cuenta la legislación detallada a continuación:

- Reglamento Electrotécnico de Baja Tensión (REBT), así como la Guía Técnica asociada a éste.
- RD 1580/2006, de 22 de diciembre, por el que se regula la compatibilidad electromagnética de los equipos eléctricos y electrónicos.

- RD 7/1988, de 8 de enero, relativo a las exigencias de seguridad del material eléctrico (y posteriores modificaciones por RD 154/95).
- RD y Normas UNE relativas al montaje, utilización y mantenimiento de los autómatas.
- EN 62061:2005: Seguridad de las máquinas. Seguridad funcional de sistemas de mando eléctricos, electrónicos y programables relativos a la seguridad.
- EN ISO 16484:2003: Automatización de edificios y sistemas de control.
- Norma IEC-1131 sobre la estandarización de los lenguajes de programación y sobre los diferentes tipos de autómatas programables y sus periféricos.
- Manual y Guía de Usuario del elemento de control.
- Manual de funcionamiento del software utilizado para la programación.

1.3.3. Estudios de limitación

En el proyecto se dispone únicamente del PLC Siemens S7-1200 como pieza clave, con el que se realizarán las distintas pruebas, junto a unos simples interruptores. Este proyecto está enfocado para ponerse en práctica en una instalación industrial, ya que con ello se podría ampliar la base de datos.

Al no disponer de acceso a un sistema industrial, la base de datos quedará simplemente vacía, llenando con ella los errores que nosotros provoquemos. Un error simplemente puede ser, que una entrada deje de funcionar. Con esto se simulará una hipotética situación del PLC funcionando en un sistema automático.

Por otra parte, el proyecto no es del todo utilizable de manera industrial, ya que muchas de las partes de éste se podrían llegar a desarrollar mucho más.

1.4. Estudios de alternativas

1.4.1. El autómata programable

Fig. 1 PLC Siemens S7-1200.

Como se especifica en las condiciones de encargo, se trabajará única y exclusivamente con el PLC Siemens S7-1200, ya que es el que cede el contratista, además del programa oficial TIA Portal, con el que se realizará la programación de dicho autómata.

El autómata programable presenta las siguientes características:

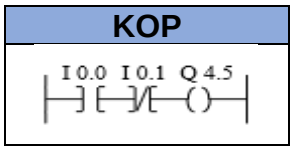
- Ofrece la flexibilidad y capacidad de controlar una gran variedad de dispositivos para las distintas tareas de automatización.
- Diseño compacto.
- Fuente de alimentación integrada.
- La CPU incluye gran cantidad de lógicas, como, la booleana, instrucciones de contaje y temporización, funciones matemáticas complejas etc.
- Protección de código “know-how”.
- Puerto PROFINET, con el que el PLC puede comunicarse a través de una red RS485 o RS232.

1.4.2. Lenguaje de programación en TIA Portal: KOP-AWL-FUP

Fig. 2 Logotipo de la compañía Siemenes.

TIA Portal dispone de diferentes tipos de lenguaje de programación:

- **KOP**: Es un esquema de contactos, escalera o ladder. Lenguaje de Step 7 gráfico. Este tipo de lenguaje presenta unas ventajas como:
  - Gran facilidad para su comprensión.
  - Lenguaje visual.
  - Probablemente, el lenguaje más extendido de las tres opciones.
  - Fácil detección de errores.
  - Gran versatilidad.



- **AWL**: Es un lenguaje de programación textual orientado a la máquina ya que las instrucciones equivalen en gran medida a los pasos con los que la CPU ejecuta el programa, algo parecido al lenguaje ensamblador. Presenta algunos inconvenientes como:
  - Lenguaje complejo.
  - Poco extendido.
  - Necesidad de muchas líneas de programación para realizar diferentes acciones, al ser un lenguaje a bajo nivel.

  Por otra parte, dispone de ciertas ventajas:
  - Lenguaje bastante preciso.
  - Programar a tan bajo nivel permite realizar ciertas funciones específicas.

AWL

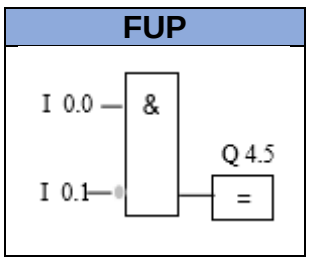
```
A   I 0.0
AN I 0.1
=   Q 4.5
```

- **FUP**: Es un lenguaje de Step 7 gráfico que utiliza los cuadros del álgebra booleana para representar la lógica que el PLC debe seguir. Presenta ventajas como:
    - Permite representar funciones complejas mediante cuadros lógicos.
    - Lenguaje gráfico.
    - Fácil de entender.
    - Bastante intuitivo.

  Pero, por otro lado, presenta unas desventajas como:

    - Cuadros de programación abultados para la realización de funciones complejas.
    - Poco extendido.

FUP



Para esta parte del proyecto se ha decidido utilizar el **lenguaje de contactos (KOP)**, debido a su alta popularidad en el sector industrial, además de que es de fácil comprensión y versatilidad.

1.4.3. Servidor TCP/IP: PHP-Java

La idea inicial del proyecto, era la comunicación directa entre el PLC y la base de datos MySQL, sin haber ningún tipo de intermediario. Pero a medida que se iba profundizando en la programación en TIA Portal, se veía que no era lo suficientemente potente para administrar datos hacia dicha base de datos.

Por ello, se estudió la posibilidad de la creación de un intermediario que sea capaz de administrar el paquete de datos enviados por el PLC, que sería la codificación del error. Con todo esto, se decidió programar un servidor TCP/IP, pero se barajaron diferentes lenguajes para la programación de dicho servidor.

Fig. 3 Logotipos de los lenguajes de programación PHP y Java.

- **PHP**: Acrónimo de Hypertext Proprocessor, PHP está enfocado principalmente a la programación de scripts del lado del servidor, por lo que se puede hacer cualquier cosa que pueda hacer otro programa CGI, como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Con todo esto, se pueden encontrar diferentes ventajas como:
    - Curva de aprendizaje muy baja.
    - Lenguaje multiplataforma.
    - No se requiere de ningún tipo de licencia, es totalmente libre y gratuito.
    - Permite la programación orientada a objetos.
    - Gran soporte de la comunidad con guías, libros y solución de dudas.

- Entornos de desarrollo rápidos y de fácil acceso.
- Lenguaje orientado a aplicaciones web.
- Fácil acceso a bases de datos.
- Ampliamente extendido en la comunidad.

Por otro lado, presenta unas desventajas como:

- Poca portabilidad de código fuente, a otros servidores.
- Necesidad de instalar un servidor web.
- Dependencia de HTML, para hacer trabajos medianamente funcionales.

- **Java**: Es un lenguaje de programación de propósito general que apareció en 1995, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo. Al igual que PHP, java presenta ventajas como:
  - Lenguaje multiplataforma.
  - Curva de aprendizaje media.
  - Dispone de una librería JDK bastante completa.
  - Lenguaje orientado a objetos.
  - Gran comunidad.
  - Gran portabilidad.

  Por otro lado, tiene unas ciertas desventajas como:

  - Sintaxis algo compleja.
  - Poco optimizado a la hora de utilizar los recursos del servidor.
  - Algunas herramientas tienen un costo adicional.

Debido a que es un proyecto de investigación, se elige el **lenguaje PHP** para la programación del servidor, principalmente por su alta versatilidad para el trabajo con bases de datos, pero también por su amplio apoyo por parte de la comunidad, con multitud de ejemplos resueltos que son de gran utilidad.

1.4.4. Plataforma de la aplicación: Android-iOS

Para la parte de la aplicación móvil, al ser un proyecto de aprendizaje e investigación, se barajó programar entre las dos plataformas de sistemas operativos para móvil más populares en el mercado, Android e iOS, donde se estudiaron sus ventajas y desventajas.

Fig. 4 Logotipos de las compañías Apple con su sistema operativo iOS, y Android (Google).

- **Android**: Es un sistema operativo para móviles basado en Linux. La estructura del sistema operativo Android, se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en el tiempo de ejecución hasta la versión 5.0, luego cambió al entorno Android Runtime (ART).

  Destacan las siguientes ventajas:

  - Android es open source, es decir, cualquier persona puede realizar una aplicación para Android gratuitamente.
  - Sistema operativo multihilo.
  - Gran cuota de mercado, alcanzando en 2011 un 50,9 % de cuota, casi el doble que su competidor iOS.
  - Fácil acceso a las aplicaciones de la comunidad.
  - Funcionalidad en multitud de dispositivos.

  Pero también presenta unas desventajas como:

  - Necesidad de recursos altos para mover ciertas aplicaciones.
  - En ciertos terminales, Android es ineficiente debido al punto anterior.
  - Fragmentación en el mercado. Debido a esto, no puedes garantizar el buen funcionamiento de la aplicación para todos los terminales con sistema operativo Android.

- **iOS**: Es un sistema operativo móvil de la multinacional Apple Inc lanzado a mediados del 2009. Originalmente desarrollado para el móvil iPhone (iPhone OS). El sistema operativo deriva de macOS, que a su vez está basado en Darwin BSD, que es un sistema operativo Tipo Unix. Dispone de cuatro capas de abstracción: la del núcleo del sistema, la de "Servicios Principales", la de "Medios" y la de "Cocoa Touch". El sistema operativo de la multinacional de Cupertino, dispone de las siguientes ventajas:
  - Sistema operativo muy eficiente en cuanto a uso de los recursos.
  - Mayor filtro y exclusividad dentro de su mercado de aplicaciones. Esto permite una mayor calidad de las aplicaciones que se encuentran en ella.
  - Interfaz intuitiva.
  - Sincronización en todos los dispositivos Apple.
  - Gran soporte con actualizaciones constantes de su sistema operativo, por parte de Apple.
  - Fragmentación muy baja en el mercado, ya que el sistema operativo sólo funciona en los teléfonos móviles iPhone.
  - Con el anterior punto, puedes asegurar la fluidez necesaria de una aplicación en la mayoría de dispositivos.

  Por otra parte, también presenta unas ciertas desventajas:

  - El sistema operativo no es de código abierto.
  - A diferencia de su competidor, la comunidad de usuarios no es tan grande.
  - Una cuota de mercado bastante inferior a la de Android.
  - Demasiados filtros para poder acceder a su tienda de aplicaciones.

En este caso, se ha decidido optar por la opción de **Android** como plataforma de programación de la aplicación. Debido a su código abierto y a su libertad de programación.

Con esta elección, se utilizará **Android Studio**, ya que es la única plataforma de programación oficial que existe para Android y que es totalmente gratuita.

Fig. 5 Logotipo de Android.

1.5. Desarrollo de la solución adoptada

Una vez desarrollado el estudio de alternativas, se pasará a detallar la solución adoptada y el desarrollo del propio proyecto paso a paso.

Como se ha detallado anteriormente, el proyecto dispondrá de los siguientes elementos principales, algunos elementos fijos y otros que se han tenido que elegir en el estudio de alternativas:

- Autómata programable Siemens S7-1200.
- Programación del automatismo en TIA Portal con lenguaje KOP.
- Base de datos MySQL.
- Servidor TCP/IP programado en PHP.
- Aplicación Android, programada en Android Studio (Java).

1.5.1. Diseño del automatismo en TIA Portal

El diseño del automatismo en TIA Portal debe seguir los siguientes factores:

- La activación de envío de datos al servidor deberá estar ligada a cualquier entrada o salida
- Se ha de configurar un bloque de comunicación con el protocolo TCP/IP capaz de comunicarse con el servidor PHP.
- Existirá un código numerado que identificará cada entrada y salida, que estará justo al empezar cualquier código de programación.
- Las entradas se accionarán mediante interruptores.
- Se ha de elegir correctamente el tipo de datos del paquete que se tenga que enviar.

1.5.2. Organigrama del automatismo

El automatismo seguirá las siguientes pautas de trabajo:

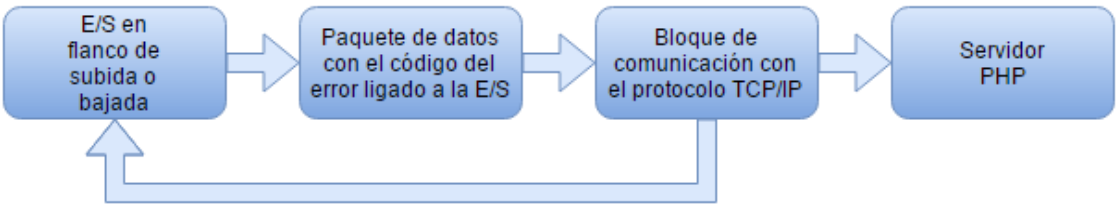


Fig. 6 Organigrama del automatismo.

A continuación, se detallará cada una de las partes del automatismo:

1. Entrada/salida en flanco de subida o bajada; una vez se active o desactive ya sea una entrada o salida, el automatismo enviará un paquete de datos con el código que se le haya adjudicado a esa entrada o salida.
2. El paquete de datos, con el que se haya configurado la entrada o salida, estará codificado por bloques MOVE de TIA Portal, con el que se podrá modificar como se desee el código.
3. El bloque de comunicación se encarga de recibir ese paquete de datos. Establece una comunicación por protocolo TCP/IP con el servidor PHP, y envía el código.
4. Una vez el servidor recibe el paquete de datos, éste se encarga de redirigirlos a las siguientes aplicaciones, como MySQL o la aplicación Android.

1.5.3. Programación del automatismo

1.5.3.1. Creación de un nuevo proyecto en TIA Portal

En la ventana inicial se debe pulsar en la opción “Crear proyecto” del panel central, lo cual permitirá en la parte derecha de la ventana principal introducir las propiedades básicas del proyecto (nombre, ruta, autor y comentarios) que se deseen introducir.

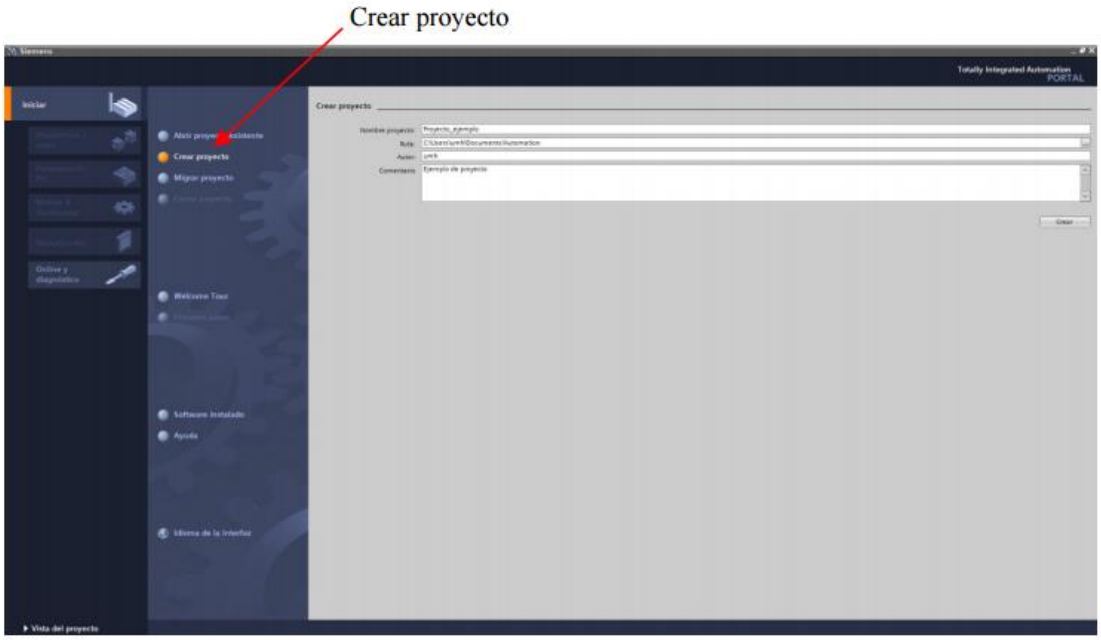


Fig. 7 Menú de creación de un nuevo proyecto en TIA Portal.

Una vez introducidos los datos del nuevo proyecto se deberá pulsar el botón “Crear” y después de unos segundos se habrá creado el nuevo proyecto, pasando así a la ventana siguiente, en la que se deberá seguir especificando otros parámetros del proyecto.

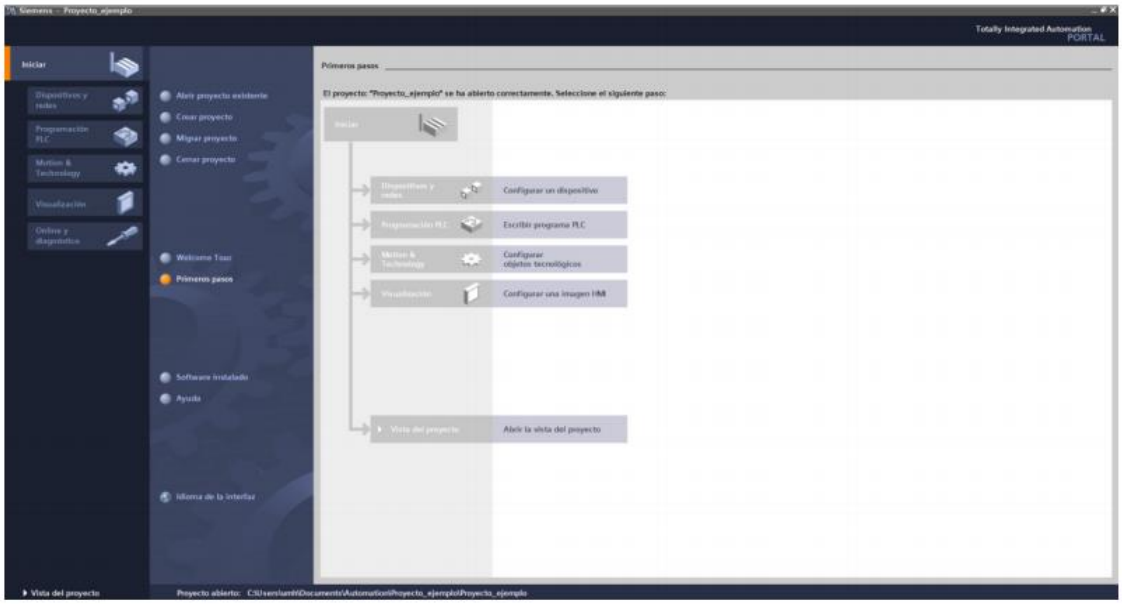


Fig. 8 Menú de creación de nuevo proyecto.

Para tener una mejor perspectiva del proyecto se ha de pasar a la “Vista del proyecto”. Para ello se pulsará en el botón “Vista del proyecto” que aparece en la esquina inferior izquierda.

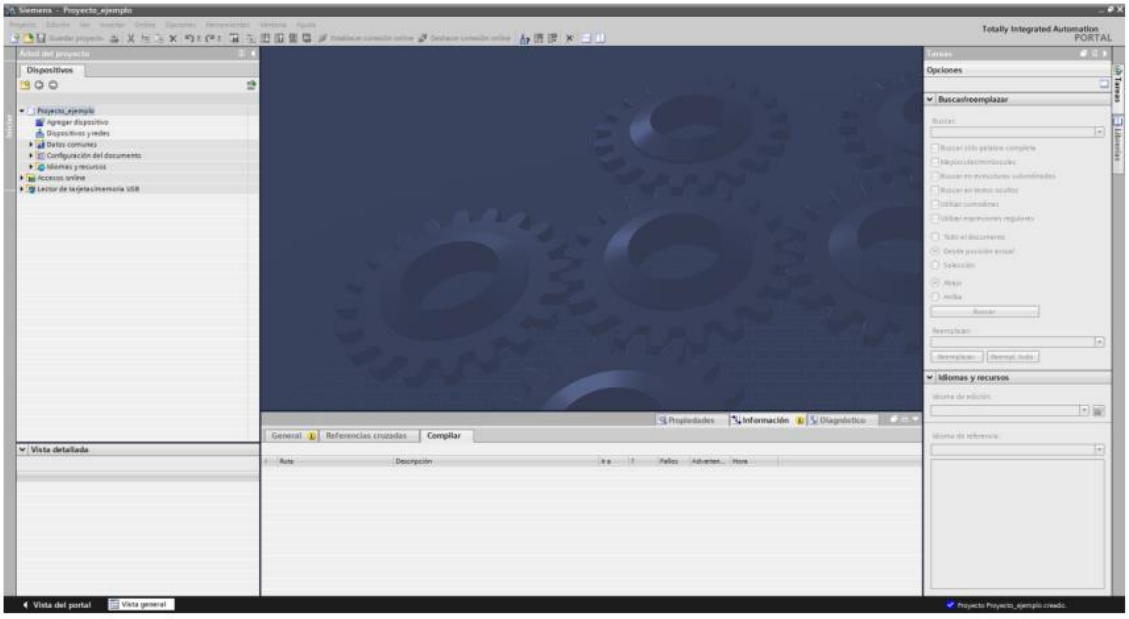


Fig. 9 Ventana del proyecto una vez creado.

1.5.3.2. Agregación del autómata al proyecto

Una vez creado el nuevo proyecto, se puede observar que no existe ningún dispositivo asociado al mismo (en el panel de la izquierda, en el "Árbol del proyecto"). Para agregar cualquier tipo de dispositivo al proyecto, el autómata debe estar conectado a la tarjeta PROFINET mediante el cable RJ-45 al PC. Una vez alimentado el autómata y conectado al PC, se hará doble click sobre la opción "Agregar dispositivo" del panel izquierdo.

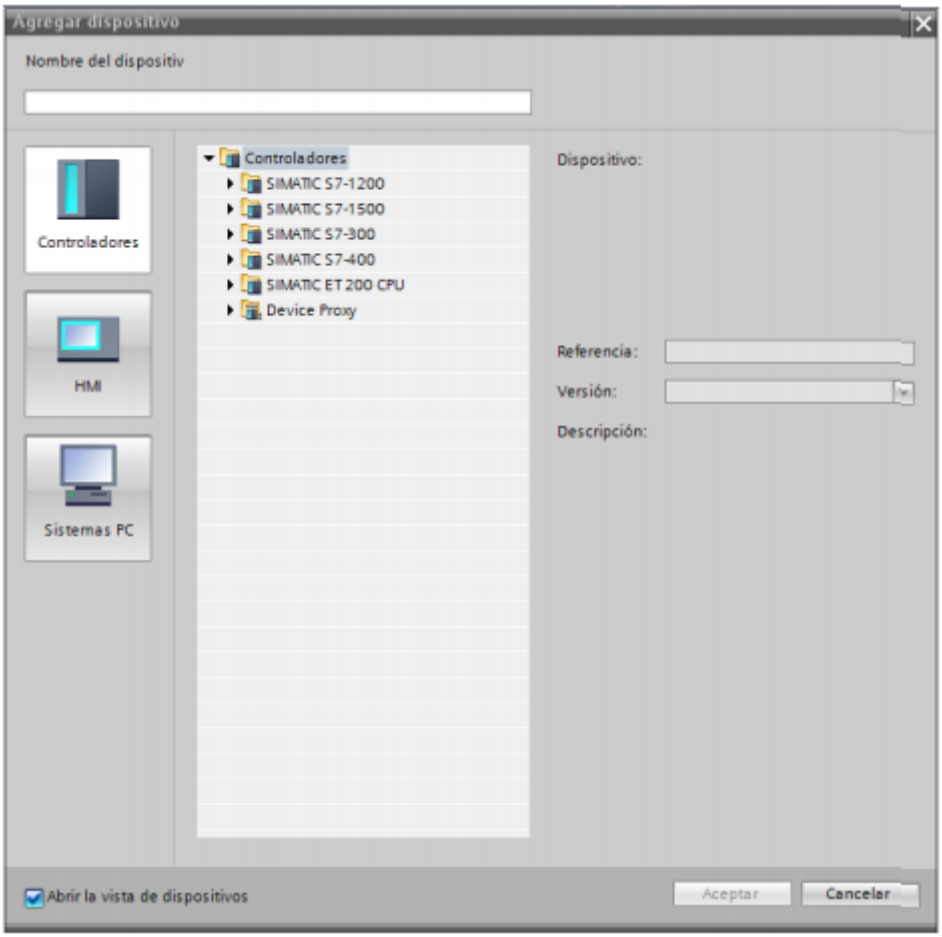


Fig. 10 Ventana para la agregación de del PLC que se disponga.

Deberá aparecer la imagen anterior, en la que se puede observar una lista de opciones. En esta ventana se deberá seleccionar el icono de "Controladores" de

la columna de la izquierda. Esta opción desplegará en la parte central un listado con todos los posibles modelos de autómatas que pueden conectarse.

Existen varios métodos para agregar el autómata, dejando que TIA Portal detecte automáticamente el modelo del autómata, o introduciéndolo manualmente.

A continuación, se detallará como funciona cada uno de estos métodos:

- **Agregación manual del dispositivo**: Cuando se conoce la versión de CPU que incorpora el autómata, puede agregarse al proyecto de forma manual. En primer lugar, se debe elegir el modelo del autómata en la parte central de la ventana de la figura anterior. Por ejemplo, SIMATIC S7-1200 y aparecerá un listado con todas las posibles CPUs para ese modelo.

  El modelo en concreto de CPU puede leerse en el lateral derecho del propio PLC.

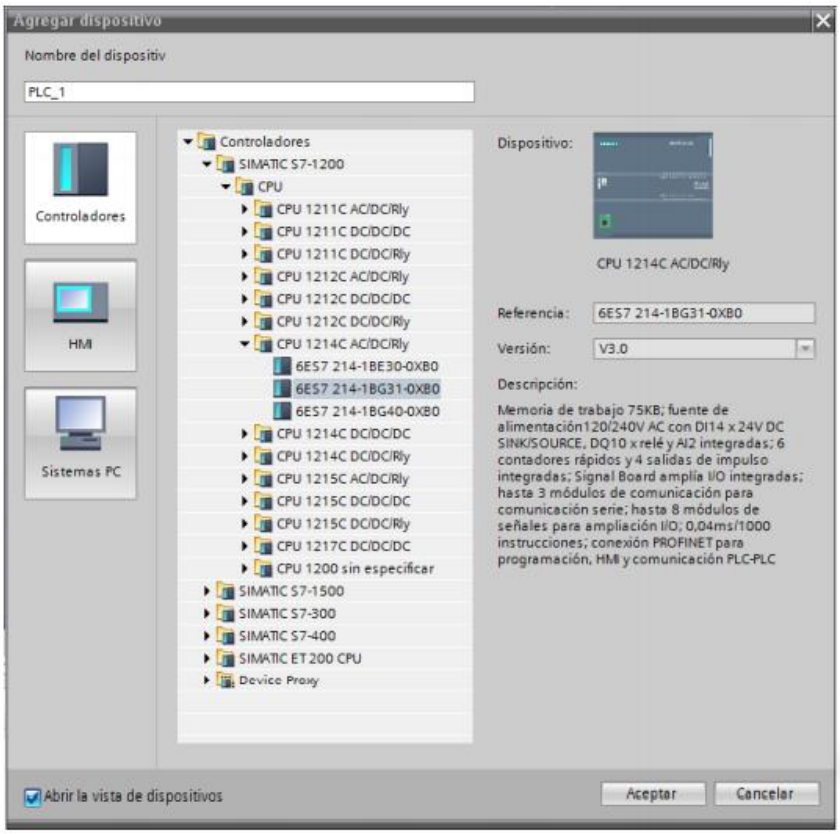


Fig. 11 Ventana donde se puede observar donde se encuentra el PLC del proyecto.

En la parte superior de la ventana anterior se puede escribir un nombre para el dispositivo, que por defecto es PLC_1. Se pulsa el botón “Aceptar” y después de un tiempo se volverá a la “Vista del proyecto” con un autómata ya agregado, (debe aparecer el autómata “PLC_1” en el panel de la izquierda).

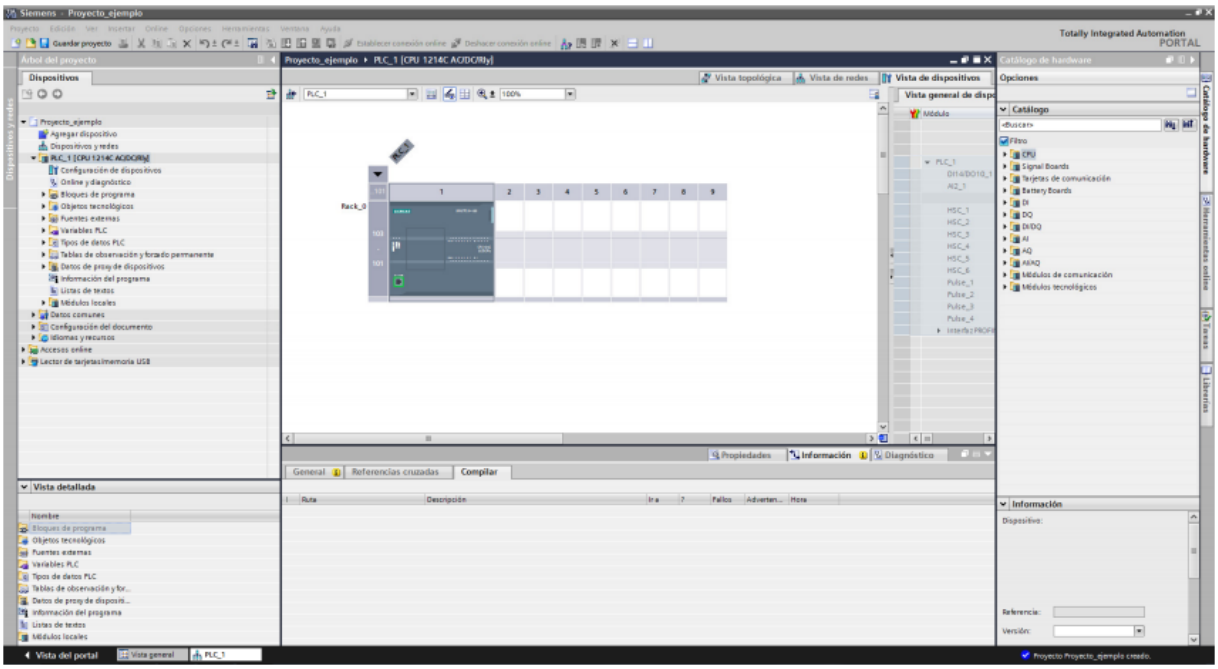


Fig. 12 Entorno de programación de TIA Portal.

- **Detección automática del dispositivo**: En este caso se debe elegir el modelo del autómata “CPU 1200 sin especificar” en la ventana “Agregar dispositivos”. En esta ventana se deberá elegir cualquier versión del firmware (no es necesario conocerla, ya que el programa la detectará, aunque es conveniente elegir la versión más baja de las que aparezca en la lista).

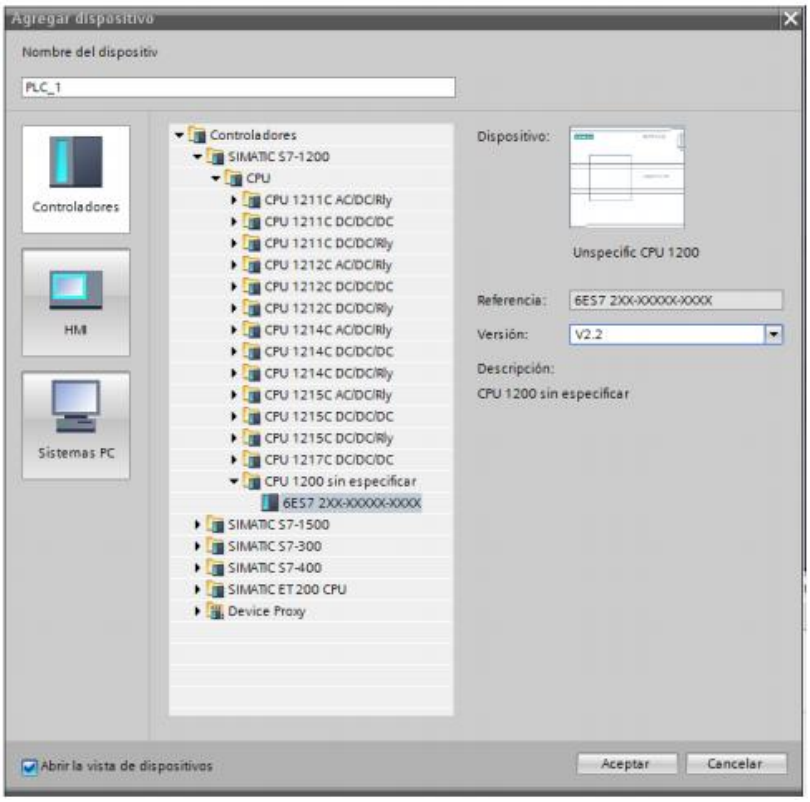


Fig. 13 Ventana de "Detección automática" del PLC.

Se pulsará el botón aceptar y aparecerá la venta de “Vista del dispositivo”.

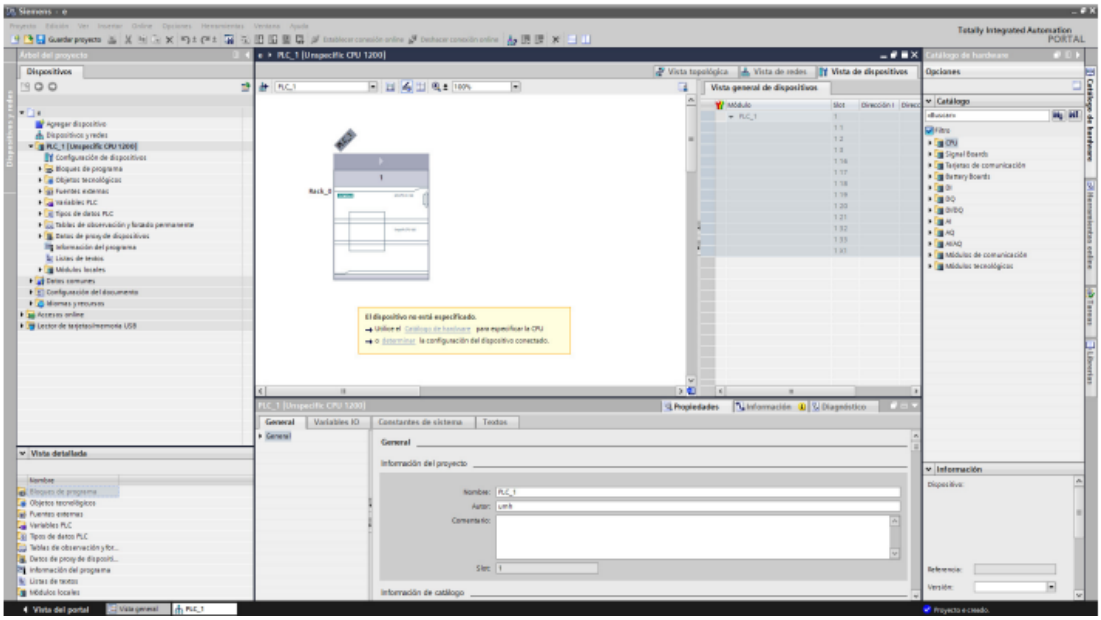


Fig. 14 Entorno de trabajo de TIA Portal.

Una vez agregado el dispositivo con cualquiera de los dos anteriores procedimientos, se deberá seleccionar la opción “Determinar la configuración del dispositivo conectado” (en el cuadro amarillo de la parte central). A continuación, aparecerá la ventana de “Detección de hardware”.

En esta ventana, se deberá configurar las opciones del interfaz, es decir, indicar al programa cómo está conectado el autómata al PC. Por ejemplo, si el PLC está conectado a través de una tarjeta de red, se deberá introducir la configuración siguiente:

- Tipo de interfaz: PG/PC: PN/IE.
- Interfaz PG/PC: NIC de Fasr Ethernet de la familia Realtek RTL…

Se demorará unos segundos para detectar el PLC, se pulsará el botón “Detección”. Una vez hecho esto, el programa volverá a la ventana “Vista del dispositivo” con el autómata detectado.

1.5.3.3. Creación del código de error

El código de error es una de las partes fundamentales del proyecto para que el operario pueda identificar rápidamente el fallo ocurrido. Por ello, deberá seguir las siguientes características:

- Seguir una codificación numérica.
- El código deberá tener 20 dígitos.
- Los primeros 4 dígitos están reservados para la identificación de la Línea en la que se encuentre instalado el PLC.
- Los 4 siguientes para la identificación de la Zona.
- Los 4 siguientes están reservados para la identificación de la Máquina.
- Los siguientes serán para identificar al dispositivo.
- Por último, los 4 dígitos finales están reservados para la identificación del Fallo.
- El cuarto byte de cada vector está reservado para el separador ":".

Código
001:002:003:004:0005

Por lo tanto, se deberá declarar cada una de estas variables en TIA Portal. Se deberá crear un tipo de dato llamado "**Struct**" que se llamará, por ejemplo, "receta" en DB_1, este tipo de dato será el paquete de datos que contendrá el código. El paquete contendrá 5 vectores char, con una longitud de 5 bytes cada uno. Como se ha dicho anteriormente, cada vector corresponderá a un código específico:

Línea	**Zona**	**Máquina**	**Dispositivo**	**Error/Fallo**

Fig. 15 Estructura del tipo de dato en TIA Portal.

Hay que tener en cuenta el tipo de datos con el que se envía el paquete, ya que de ello dependerá cómo se programe el servidor.

Para crear el código, en TIA Portal se utilizarán unos bloques específicos, llamados "**MOVE**". Con estos bloques, en el momento que se active o desactive un bit del PLC, se activarán los bloques que modificarán el mensaje a enviar. Ese mismo bit, será el que active a su vez el bloque de comunicación, que se detallará en los siguientes puntos.

Se deberán poner en paralelo tantos los bloques MOVE como los dígitos que se requieran para el código.

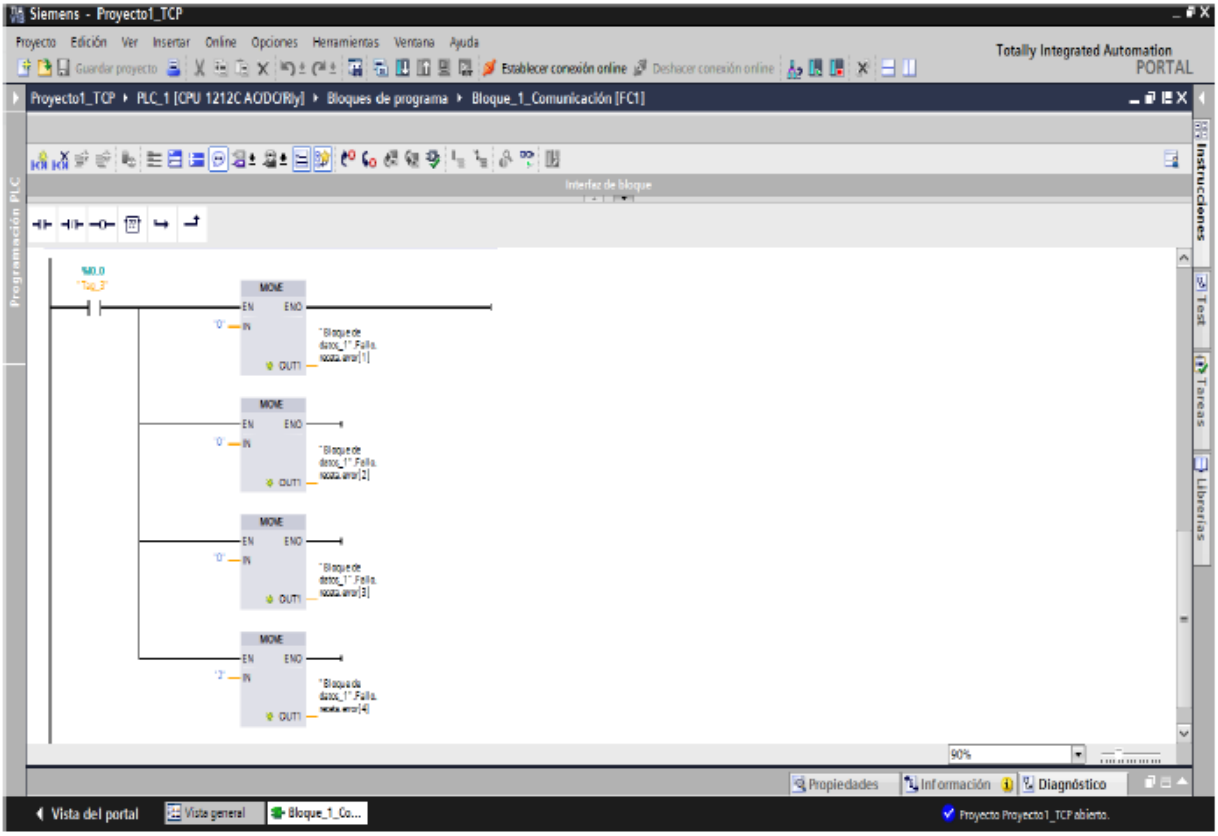


Fig. 16 Bloques MOVE que son los encargados de pasar el dato que se quiera, al bloque de comunicaciones.

Como se puede observar en la imagen, el bloque dispone de dos entradas y una salida:

- La primera entrada, es el bit de inicialización del bloque, para activarlo o desactivarlo.
- La segunda entrada, es el dato que se requiere enviar, por ejemplo, el primer bloque será, el primer dígito del error y el último el separador.
- Su salida, modificará el array de nuestra “receta”, por ejemplo, los 4 bloques MOVE de la imagen anterior, modificará el primer array del bloque de datos llamado “receta”.

1.5.3.4. Programación del bloque de comunicación

En este apartado, se detallará el bloque encargado de realizar la comunicación por protocolo TCP/IP y su configuración.

En primer lugar, se debe localizar el bloque en las instrucciones que aporta TIA Portal, en este caso, se localiza en la carpeta “Open user communication”:

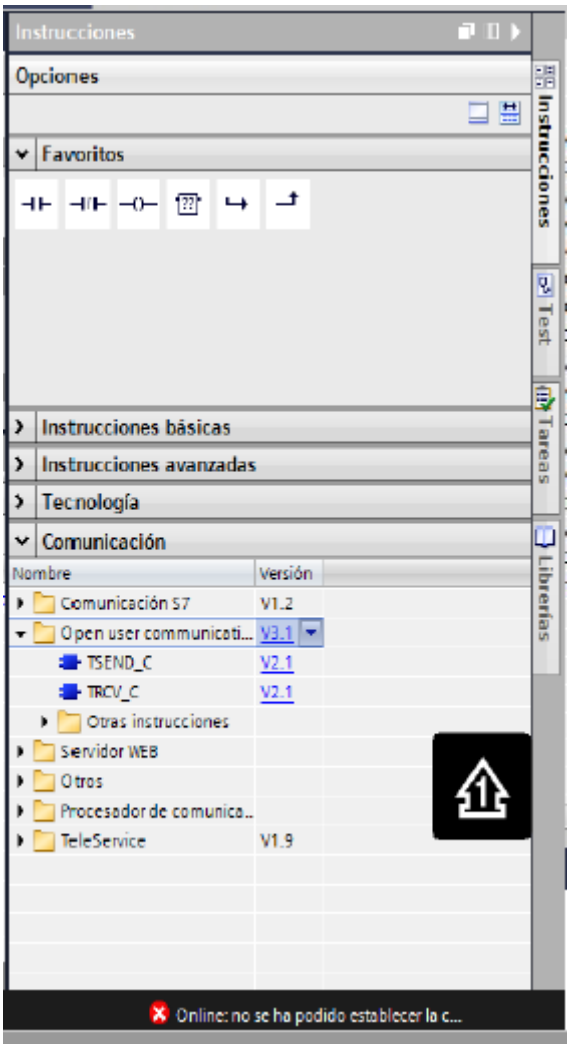


Fig. 17 Ubicación del bloque de comunicaciones que se utiliza en el proyecto.

El bloque que se adapta a las especificaciones se llama **TSEND_C**.

En el ANEXO III, se podrá observar cada una de las especificaciones de este bloque.

En este apartado se especificará la configuración de los parámetros paso a paso.

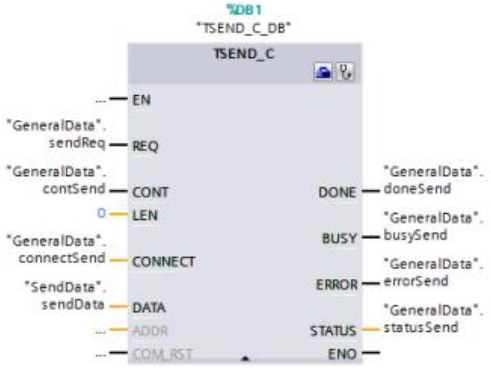


Fig. 18 bloque TSEND_C predeterminado.

Si se da un click derecho encima del bloque con el ratón, se podrá acceder a las propiedades del bloque, con lo que aparecerá la siguiente ventana:

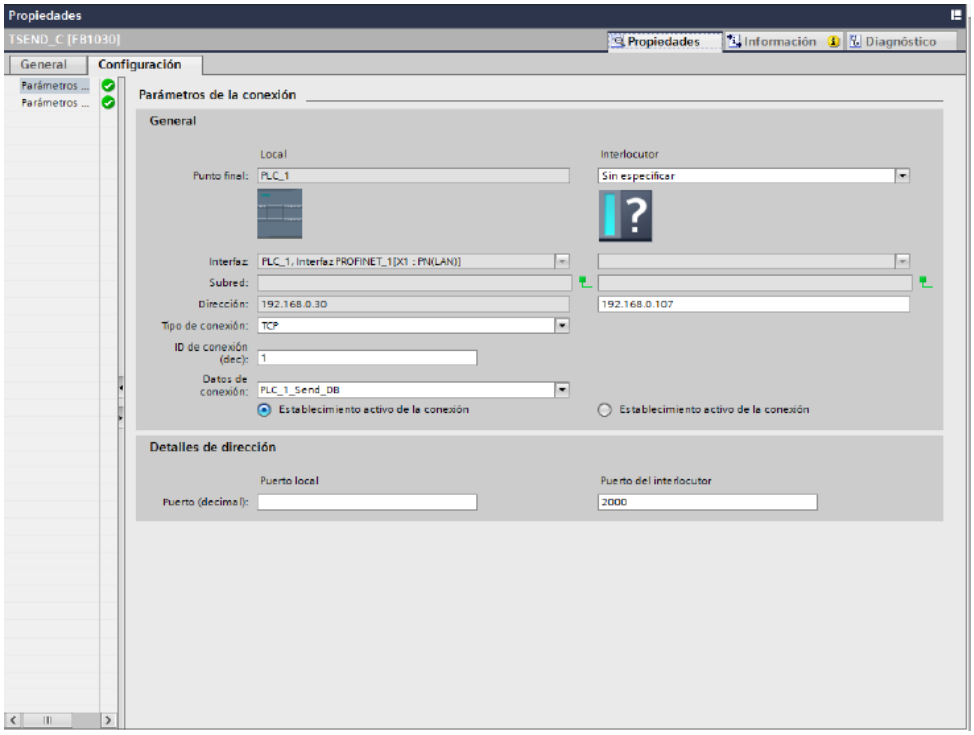
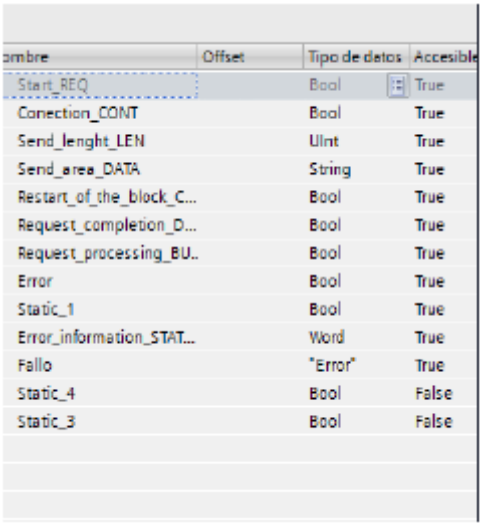


Fig. 19 Panel de configuración del bloque TSEND_C.

Habrá ya conexiones predeterminadas, como la IP del PLC o el tipo de conexión. Por otro parte, el apartado de datos de conexión es el fundamental que se ha de cumplimentar correctamente para que el bloque pueda funcionar.

Otro de los parámetros fundamentales es, **la IP del interlocutor**, que en este caso sería la dirección IP del servidor PHP, con la que deberá establecer conexión.

En el árbol de carpetas del proyecto, se encontrará el bloque de datos DB_1, donde se van guardando las variables que se vayan creando. El PLC_1_SEND_DB deberá contener los siguientes datos:



ombre	Offset	Tipo de datos	Accesible
Start_REQ		Bool	True
Conection_CONT		Bool	True
Send_lenght_LEN		UInt	True
Send_area_DATA		String	True
Restart_of_the_block_C...		Bool	True
Request_completion_D...		Bool	True
Request_processing_BU..		Bool	True
Error		Bool	True
Static_1		Bool	True
Error_information_STAT...		Word	True
Fallo		"Error"	True
Static_4		Bool	False
Static_3		Bool	False

Fig. 20 Lista de datos para la configuración del bloque TSEND_C-

Con estos datos se deberá atribuir a cada una de las entradas y salidas al bloque de comunicación para su funcionamiento. Las instrucciones se ejecutan de forma asíncrona y se pueden destacar las siguientes funciones tal y como se registra en el diccionario de bloques de TIA Portal:

- La conexión se configura y establece con CONT=1. Si la conexión se establece correctamente, el parámetro DONE se pone a "1" durante un ciclo. Si la CPU pasa al estado operativo STOP, se interrumpe una conexión existente y se elimina la conexión creada. Para volver a configurar y establecer la conexión, es preciso volver a ejecutar

"TSEND_C". El número de conexiones posibles se indica en los datos técnicos de la CPU.

- El área de transmisión se especifica en el parámetro DATA. Este contiene la dirección y la longitud de los datos que deben enviarse. No se debe utilizar en el parámetro DATA áreas de datos con el tipo de datos BOOL o Array of BOOL. Si utiliza únicamente valores simbólicos en el parámetro DATA, el parámetro LEN debe tener el valor "0".
- La petición de transmisión se ejecuta cuando se detecta un flanco ascendente en el parámetro REQ. En el parámetro LEN se especifica el número máximo de bytes que deben enviarse con una petición de transmisión. Al enviar datos (flanco ascendente en el parámetro REQ), el parámetro CONT deberá tener el valor "1" para establecer o mantener una conexión. Los datos por enviar no se pueden editar hasta que no se haya ejecutado por completo la petición de transmisión. Si la petición de transmisión se ejecuta correctamente, el parámetro DONE se pone a "1". No obstante, el estado lógico "1" en el parámetro DONE no confirma que el interlocutor haya leído ya los datos enviados.
- La conexión se deshace cuando el parámetro CONT se pone al valor "0" aunque no haya finalizado aún una transferencia de datos en curso. Esto no es aplicable si se utiliza una conexión ya configurada para "TSEND_C".
- Ajustando el parámetro COM_RST a "1", se puede resetear el establecimiento de la conexión o una transferencia de datos en curso en cualquier momento. Con ello, la conexión existente se deshace y se establece una nueva. Si se están transfiriendo datos al reiniciarse la instrucción, podrían perderse datos.
- Para volver a habilitar "TSEND_C" tras la ejecución (DONE = 1), debe llamar la instrucción una vez con REQ = 0.

Una vez configurado todo el bloque, como se ha especificado en los puntos anteriores, en el parámetro **DATA**, se encontrará el bloque de datos del código, que proceden de los bloques **MOVE**.

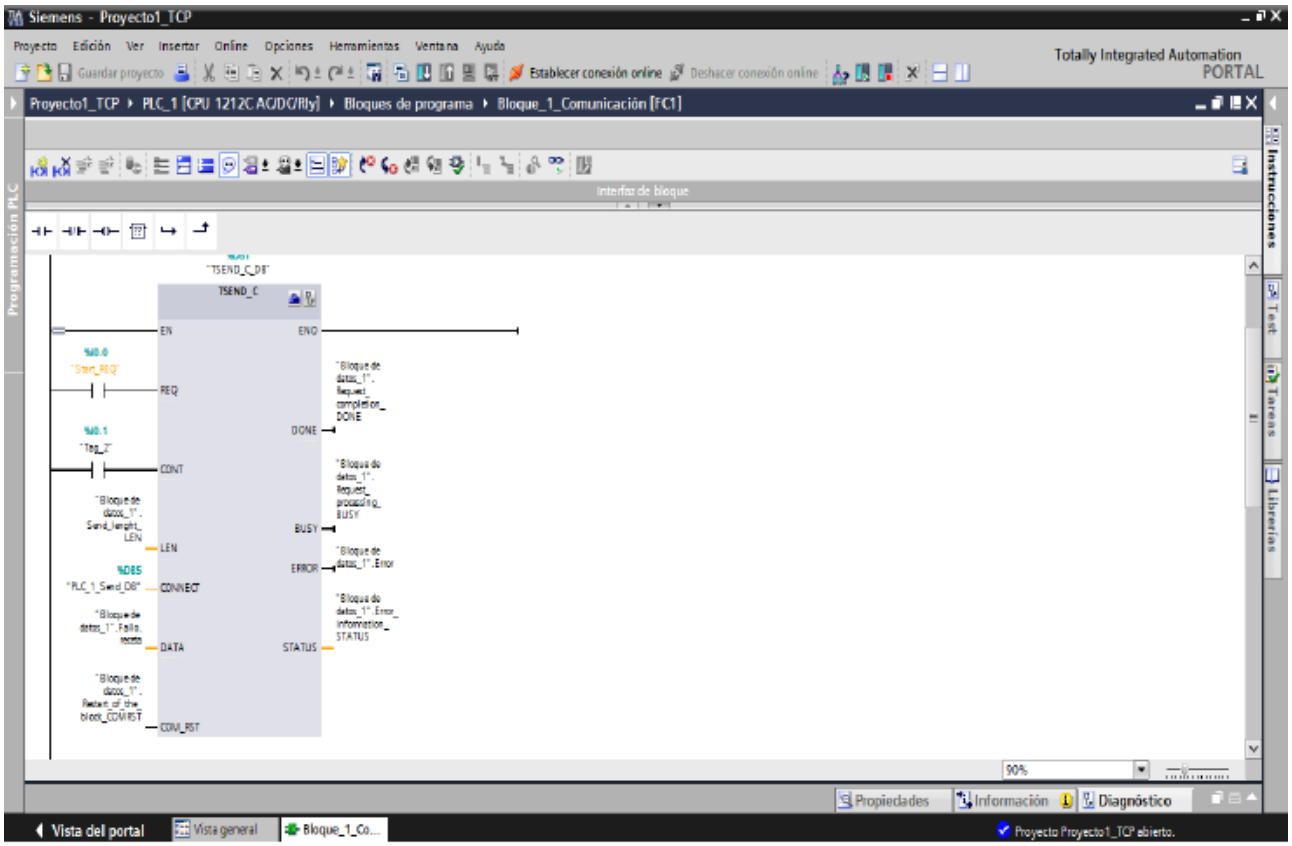


Fig. 21 Bloque de comunicaciones TSEND_C configurado.

1.5.3.5. Ejecución del programa

En este apartado se detallarán los pasos que se deben seguir para la ejecución del programa:

1. Se volcará el programa a la memoria del PLC, este volcado debe realizarse con autómata en modo STOP.
2. Se inicializará el programa con TIA Portal y el PLC conectado con el cable RJ-45.
3. Para poner en funcionamiento las diferentes entradas del PLC se hará uso del periférico.
4. Se activarán las entradas que se le hayan vinculado al bloque de comunicación. En este caso I0.0 estará siempre activado y I0.1 será la que establezca la conexión.
5. Por otro lado, I0.4 estará vinculado a la simulación de fallo de entrada.

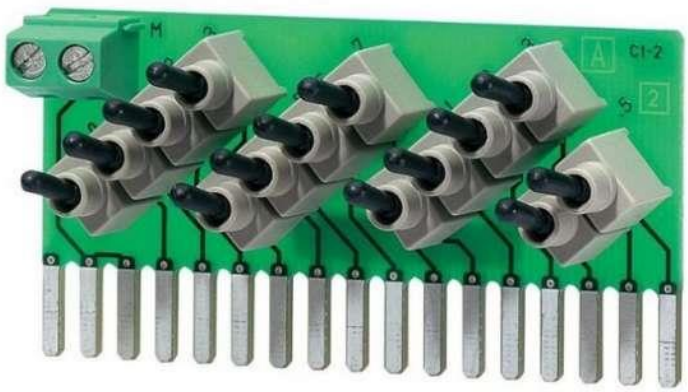


Fig. 22 Interruptores para la entrada del PLC.

1.5.4. Diseño del servidor programado en PHP

El diseño del servidor, que es el centro neuronal del proyecto, deberá seguir ciertas pautas ya que debe establecer una comunicación con los distintos servidores:

- El protocolo del servidor que debe utilizar será TCP/IP.
- Deberá programarse para que sea capaz de recibir correctamente el paquete de datos proporcionado por el bloque TSEND_C.
- Deberá establecer una conexión con el servidor MySQL.
- El paquete de datos también deberá enviarse a la aplicación Android.

1.5.5. Organigrama del servidor

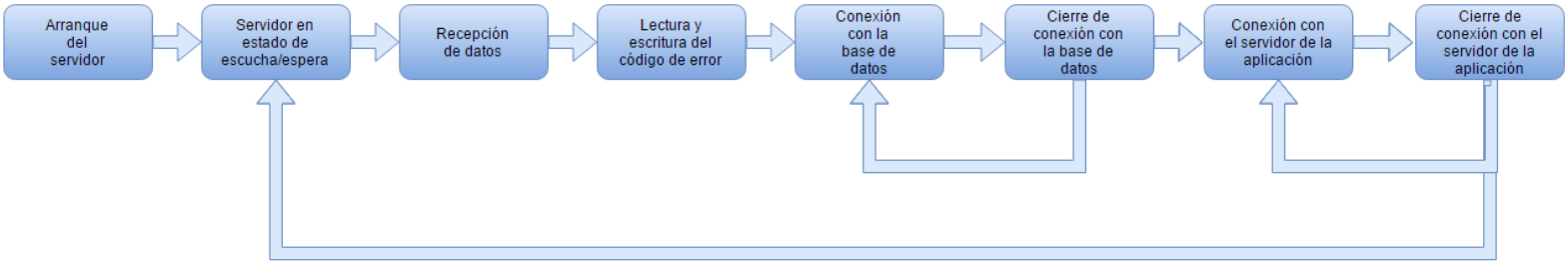


Fig. 23 Organigrama del servidor.

En el organigrama se puede observar cómo funcionará a grandes rasgos el servidor y el tratamiento del paquete de datos enviado por el PLC, hasta el establecimiento de conexión con la base de datos MySQL y la aplicación Android.

1.5.6. Programación del servidor en lenguaje PHP

1.5.6.1. Introducción a la programación en PHP

Hypertext Preprocessor (PHP) es un lenguaje interpretado de alto nivel vinculado a páginas HTML y ejecutado en el servidor.

Antes de empezar a programar en PHP es necesario conocer las nociones básicas del lenguaje de programación, ya sea en ejemplos con operaciones sencillas, para así poder relacionarse con el entorno de programación.

- **Tipos de variables**: Los nombres de variables comienzan con el signo “$” y son sensibles a mayúsculas y minúsculas (no así las palabras claves del lenguaje). En PHP no es necesario definir el tipo de dato que almacena antes de utilizarla, las mismas se crean en el momento de emplearlas. Las variables se declaran cuando se le asigna un valor, por ejemplo:
  - $dia = 24: Se declara una variable de tipo integer.
  - $sueldo = 758.43: Se declara una variable de tipo double.
  - $nombre = "juan": Se declara una variable de tipo string.
  - $exite = true: Se declara una variable booleana.
- **Tipos de expresiones y operadores**: En PHP una expresión es cualquier cosa que pueda contener un valor. Las expresiones más simples son las variables y las constantes y otras más complicadas serán las funciones, puesto que cada función devuelve un valor al ser invocada, es decir, contiene un valor, por lo tanto, es una expresión.

  Todas las expresiones en PHP son exactamente igual que en C. Los operadores abreviados, los incrementos, etc. son exactamente iguales. Incluso existen otros operadores adicionales como el operador "." que concatena valores de variables, o el operador "===" denominado operador

de identidad que devolverá verdadero si las expresiones a ambos lados del operador contienen el mismo valor y a la vez son del mismo tipo.

Operadores
,
or
xor
and
print echo
= += -* *= /= .= %= &= |= ^= ~= <<= >>=
?:
||
&&
|
^
&
== != ===
< <= > >=
<< >>
+ - .
* / %
! ~ ++ -- (int) (double) (string) (array) (object) @
[
new

- **Estructuras de control**: Además de la sintaxis normal (parecida al Perl o al C), PHP ofrece una sintaxis alternativa para alguna de sus estructuras de control: if, while, for, y switch.

  En cada caso, la forma básica de la sintaxis alternativa es cambiar abrir-llave por dos puntos (:) y cerrar-llave por endif, endwhile, endfor, or endswitch, respectivamente. En la siguiente tabla se resumirá cada una de las opciones que ofrece PHP para las estructuras de control.

Estructuras	Alternativas
if, if else, if elseif	if: endif;
while	while: endwhile;
for	for; endfor;
do... while	-
foreach (array as $value) foreach (array as $key=>$value)	-
switch	switch: endswitch;
continue	-
break	-
require ()(Necesitan estar dentro de tags PHP)	-
Include ()(Necesitan estar dentro de tags PHP)	-

Una vez repasados los conceptos básicos, se deberá pasar a la programación del servidor. Aunque sólo se hayan repasados los conceptos básicos, el servidor hará uso de funciones centradas en otro tipo de acciones, sus características se podrán encontrar en el ANEXO I.

1.5.6.2. Creación del código del servidor

A continuación, se detallará la estructura de ejecución que se puede ver gráficamente en el organigrama, con la que se ha realizado el programa, que dicho código se puede comprobar en el ANEXO I.

1. Creación del socket (conector entre servidores) eligiendo los parámetros y protocolos de conexión, en este caso TCP.
2. Proporcionar información al socket sobre la dirección IP, con la que se requiera establecer información.
3. El socket mantendrá una conexión abierta para la “escucha” de peticiones.
4. Creación de un buffer como contenedor de datos, para así poder almacenar el paquete de datos que pueda recibir el servidor.
5. Lectura del paquete de datos recibido.
6. Descomponer el código recibido en diferentes variables, recordando que cada parte del código está dividida por el separador “:”.
7. Establecer una conexión con el servidor MySQL.
8. En el servidor se encontrará las funciones en SQL con las que se trabajará en la base de datos.
9. Cada parte del código se almacenará en diferentes registros: Línea, Zona, Máquina, Dispositivo, Fallo.
10. Cada registro se almacenará con la hora y fecha del servidor, justo en el momento de activación del error.
11. Se cerrará la conexión con la base de datos MySQL.
12. Se creará otro socket, pero esta vez, un socket cliente, que será el que envíe datos a la aplicación Android.
13. Como en el punto número 2, se le debe proporcionar los datos correspondientes como IP o protocolos de conexión.
14. Cerraremos la conexión del socket cliente.

1.5.6.3. Ejecución del programa

Para poder ejecutar cualquier programa en PHP, es necesaria la instalación de un servidor Apache y una framework “php.exe” capaz de ejecutar los programas en lenguaje PHP, para ello se hará uso del software gratuito **Wamp Service**, con el que se podrá ejecutar el código del servidor.

Una vez instalado dicho programa, para poder ejecutar el código se deberá ejecutar sobre la consola de comando de Windows. Se deberán seguir los siguientes pasos:

1. Abrir la consola de Windows.
2. Abrir el archivo “php.exe” desde la consola, que está en la raíz donde se haya instalado Wamp Service.
3. Con “php.exe” se deberá abrir el archivo donde esté el código de programación del servidor.
4. Cuando se haya abierto el programa, el servidor estará abierto escuchando peticiones.

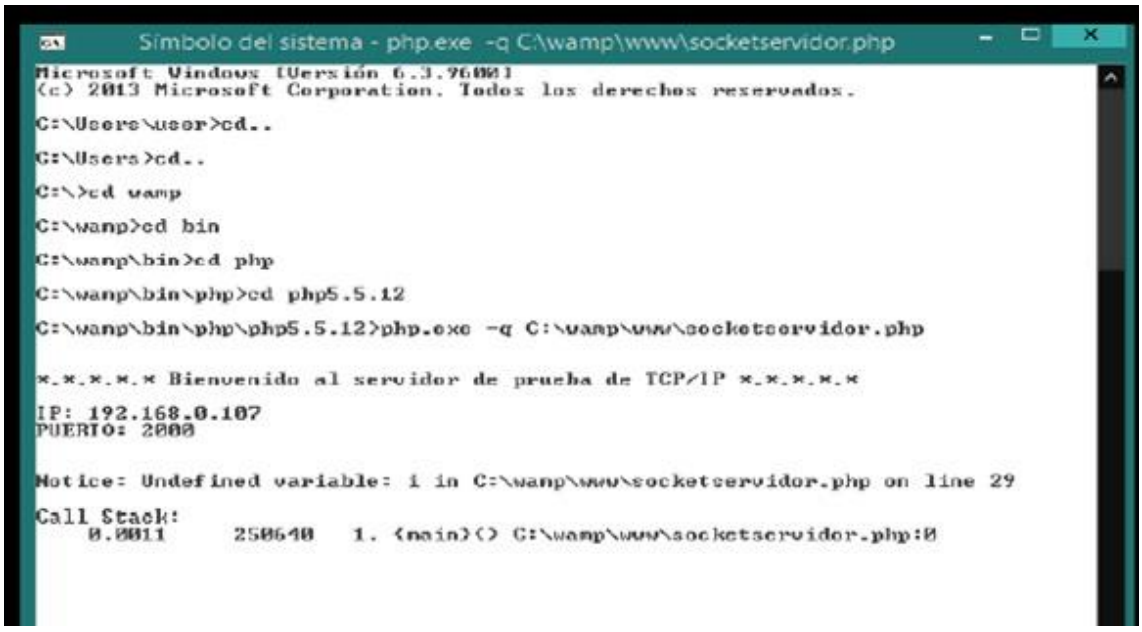


Fig. 24 Pantalla de funcionamiento del servidor PHP.

1.5.7. Diseño de la base de datos con MySQL

El correcto diseño de la base de datos, es otro de los elementos principales del proyecto. En la red local, se encontrará un servidor MySQL que se encargará de almacenar todos los posibles errores registrados. La base de datos deberá seguir las siguientes características:

- La tabla principal deberá contener los siguientes campos:
  - Línea.
  - Zona.
  - Máquina.
  - Dispositivo.
  - Fallo/Error.
  - Fecha_hora.
- Cada campo deberá estar correctamente programado para la recepción correcta del tipo de datos enviados por el servidor PHP.
- El campo “fecha_hora” será un registro automático.

Una vez repasados los conceptos por los que se regirá la base de datos, en primer lugar, se repasarán los conceptos previos sobre el lenguaje SQL, y la programación en MySQl, para así establecer unos conceptos básicos previos.

1.5.7.1. Introducción al lenguaje SQL

SQL proviene de las siglas de su nombre en inglés “Structured Query Language”, que en español significa “lenguaje de consulta estructurada”.

Es un lenguaje específico del dominio, con el que podemos gestionar bases de datos, permitiendo realizar diversos tipos de operaciones entre ellas. Una de sus características, es el uso de operaciones con el fin de efectuar consultas para

así recuperar información que se encuentra alojada en la base de datos, así como hacer cambios en ellas.

1.5.7.2. Concepto y características de una base de datos

Una base de datos es un conjunto de datos organizados e interrelacionados que se organizan y relacionan entre sí de manera sistemática, es decir, siguiendo unas determinadas reglas.

Para comprender mejor el concepto de base de datos, es mejor conocer los objetos de los cuales está formada. Normalmente presentan 6 tipos de objetos:

- **Tablas**: Son los objetos principales en una base de datos. Es la forma "física" donde se almacenan todos los datos. Cada una de las filas de las que está compuesta la tabla recibe el nombre de "registro", mientras que las columnas reciben el nombre de "campo".

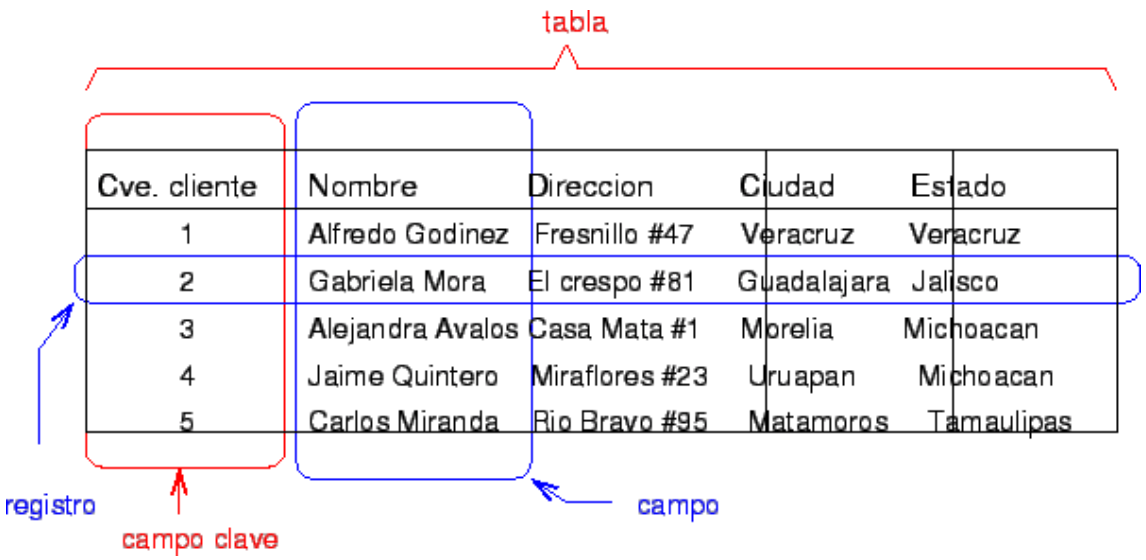


Fig. 25 Esquema ilustrativo para la identificación de registros y campos en una tabla.

- **Vistas**: Las vistas son aquellas tablas que son el resultado de una consulta SQL. Sobre estas tablas, se pueden realizar operaciones sobre ellas mismas.
- **Funciones**: Las funciones, son aquellas operaciones que el sistema gestor de base de datos realiza sobre las mismas, son las operaciones necesarias para poder modificar la base de datos a voluntad.
- **Índices**: El índice es aquella tabla que permite el acceso a los elementos con mayor dinámica a los registros.
- **Procesos almacenados**: Se trata de un programa que se almacena en la base de datos y que se ejecuta directamente en el sistema gestor de base de datos.
- **Triggers**: Es un proceso que se ejecuta únicamente cuando se cumple una condición preestablecida. Los triggers o disparadores pueden crear, editar o borrar tablas en una base de datos.

En cuanto a la organización de las tablas de las bases de datos existen varios tipos de modelos, entre ellos cabe destacar:

- **Modelo tabla**: Como ya se ha explicado anteriormente, se trata de una serie formada por una tabla bidimensional, compuesta por “registros” (filas) y por “campos” (columnas), en la que se van recogiendo los datos.
- **Modelo jerárquico**: Se basa en registros organizados en forma de árbol jerárquico inverso.
- **Modelo de redes**: Está basado en registros, un modo jerárquico normal, del que un registro puede tener otro registro.
- **Modelo racional**: El más utilizado, un modelo basado en el sistema de tablas, pero cada tabla presenta una relación entre cada una de ellas. Por lo tanto, cada elemento de una base de datos relacional es capaz de relacionarse sin necesidad de duplicar la información.

1.5.7.3. Gestor de base de datos: MySQL

Un sistema gestor de bases de datos es un software informático que permite interactuar con toda la base de datos. Esta herramienta informática a través de su interfaz permite acceder a cada uno de los datos que integran dicha base de datos.

Como sistema gestor de base de datos, entra a la palestra MySQL. Es un sistema basado en el modela relacional, mencionado anteriormente que además dispone de las funciones de multihilo (distribuye las tareas entre los procesadores disponibles aumentando y optimizando el rendimiento) y multiusuario.

MySQL dispone de las siguientes características, de las cuales lo hacen destacar por encima de otros gestores de base de datos, para poder trabajar con él:

- Unas de las características más importante es que el programa es open source (código abierto), lo que significa que es totalmente gratuito de utilizar.
- Tiene un uso bastante extendido.
- Un fácil aprendizaje y bastante intuitivo.
- Buena integración con el lenguaje PHP.

Hay varias maneras de poder ejecutar comandos de MySQL, obviamente hay que tener el programa instalado y en marcha para poder ejecutar los comandos. En la raíz donde se ha instalado el programa se encontrará el ejecutable “mysql”, es el monitor de MySQL, este es un programa de comandos por línea.

Otra forma de acceder en Windows 10 para poder iniciar la consola, se debe hacer click derecho sobre el botón de inicio y click a “Símbolo de sistema”. Una vez ejecutada la consola se debe acceder a la carpeta donde esté instalado MySQL, mediante comandos, para ello se teclea lo siguiente:

- *c:\mysql\bin* (y enter).

Una vez tecleado esto, se habrá entrado en la consola de MySQL, si el inicio de sesión, no dispone de ninguna contraseña se deberá teclear lo siguiente:

- *mysql -u root* (y enter).

Si se diera el caso que la sesión disponga de contraseña, se teclearía la contraseña de la siguiente forma:

- *mysql -u root -p* (contraseña de la sesión).

Una vez iniciado la consola, se obtendrá lo siguiente:

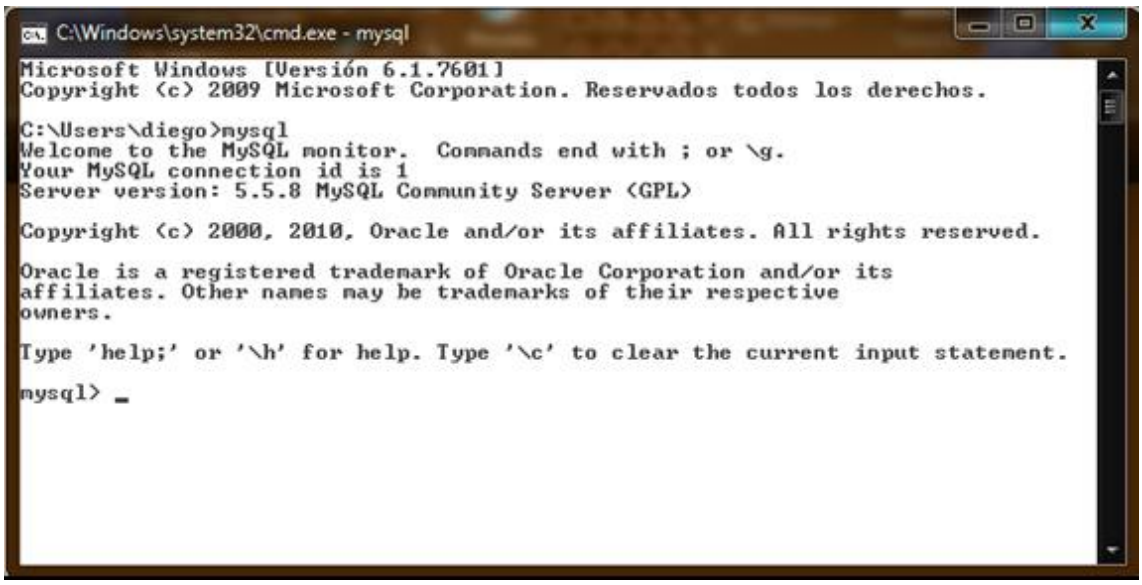


Fig. 26 Pantalla de inicio del servidor MySQL.

1.5.7.4. Funciones básicas en MySQL

Una vez iniciada la consola se repasará algunas de las funciones más básicas en MySQL, explicando su función para así poder empezar a trabajar con MySQL.

- **Show databases**: Esta función muestra las bases de datos que tengamos instaladas en el terminal.
- **Show tables**: Al igual que la función anterior, ésta mostrará las tablas que estén creadas en el terminal mysql, mostrando una lista con todas las tablas creadas.
- **Use (nombre de la base de datos)**: Si se escribe la función “use” seguida del nombre de la base de datos, se seleccionará la base de datos, por lo tanto, todas las operaciones que se realicen después afectarán únicamente a dicha base de datos.
- **Create database (nombre de la base de datos)**: Con esta función, como es de suponer, permitirá crear una base de datos con el nombre que se desee.
- **Create table**: La función para crear una tabla nueva en una base de datos es algo más compleja que las anteriores funciones. A continuación, se mostrará un ejemplo sencillo de la creación de una tabla:

```
create table usuarios (nombre varchar(30), clave varchar(10));
```

En este caso, se ha creado una tabla con el nombre “usuarios”. Entre paréntesis se encuentra los campos (columnas) de las que se compone la tabla, cada campo va separado de una coma y se les nombra de la forma en la que se requiera.

Por otra parte, cuando se defina cada campo, se debe definir qué tipo de dato almacenará esa columna. El campo “nombre” contendrá una cadena de hasta 30 caracteres de longitud (varchar(30)). Lo mismo para el campo “clave”, que contendrá una cadena de hasta 10 caracteres.

- **Describe (nombre de la tabla)**: La función “describe” nos mostrará la estructura de una tabla o base de datos, donde muestra cada campo de la que está compuesta la tabla, lo que ocupa en bytes y otros datos.
- **Drop (nombre de la tabla)**: Con esta función, se elimina la tabla que se desee.
- **Alter table (nombre de la tabla)**: Para modificar una tabla ya creada, ya sea añadiendo un nuevo campo o modificando el tipo de datos que van a conservar dichos elementos, se podrá realizar mediante la función “alter table”. Si, por ejemplo, se requiere añadir un campo se realizaría de la siguiente forma:

```
alter table usuarios add apellidos varchar(50);
```

Si se desea modificar el tipo de datos, sería de la siguiente forma:

```
alter table usuarios modify colum apellidos varchar (60);
```

- **Insert into (nombre de la tabla)**: Con esta sentencia, se podrá insertar registros o modificar los registros que ya existentes. Ejemplo:

```
insert into usuarios (nombre, clave) values (‘Juan’,’Juan1234’);
```

Como se observa en el ejemplo, en primer lugar, se detalla el campo que se requiere modificar o rellenar, a continuación, entre comas flotantes se escribe el registro que se quiera insertar. Si hay algún otro registro previo éste lo modificará.

- **Select**: Este comando permite la selección de registros que se especifiquen en la sentencia. Ya sea los registros de un campo entero, a un registro en específico que se declare.

1.5.7.5. Tipos de datos

Al crear una tabla es importante elegir la estructura adecuada, esto es, definir los campos y sus tipos más precisos, según campo. Por ejemplo, si se crea un campo numérico tipo “integer” con el atributo “unsigned” almacenará solamente valores enteros positivos y no podrá almacenar tipos numéricos con parte decimal tipo “float”.

Los tipos de datos se estructuran en cinco tipos y cada tipo en diferentes subtipos:

- **Texto**: Para poder almacenar texto se utiliza cadenas de caracteres. Estas cadenas se colocan entre comillas simples. Dentro de esta familia se tienen los siguientes tipos:
    - **Varchar(x)**: Cadena de caracteres de longitud variable en la cual se determina el máximo de caracteres que tendrá dicha cadena con el argumento “x” que va entre paréntesis. Su rango alcanza entre 1 a 255 caracteres.
    - **Char(x)**: A diferencia de “varchar”, esta cadena define una cadena de longitud fija, donde su rango también es de 1 a 255 caracteres. Si la cadena ingresada es menos a la longitud definida, por ejemplo, si es de 10 y almacenamos un dato de 6 bytes, se almacenará espacios en blancos a la derecha. Por lo tanto, ocupará siempre los bytes que se le asignen con la variable “x”.
    - **Blob o text**: Bloques de datos de 60.000 caracteres de longitud aproximadamente.
- **Numéricos**: Al igual que los datos tipo texto, existe diferentes subtipos para poder almacenar datos numéricos en los registros:
    - **Integer(x)** o **int(x)**: Este tipo de datos sirve para almacenar tipos de datos numéricos y enteros, donde su rango es de -2000000000 a 2000000000 aproximadamente. Si llevara el atributo “unsigned” el rango sería de 0 a 4000000000, es decir, sólo números enteros positivos. El tipo “integer” o “int” tiene diferentes subtipos que se diferencian en el rango de cada uno:

        - **Mediumint(x)**: Disponen de un rango de -8000000 a 8000000 aproximadamente. Sin signo va de 0 a 16000000 aproximadamente.
        - **Smallint(x)**: Con un rango de -30000 a 30000 aproximadamente, sin signo, de 0 a 60000 aproximadamente.
        - **Tinyint(x)**: Tiene un valor entero pequeño, cuyo rango es de -128 a 127. El tipo sin signo va de 0 a 255.
        - **Bool**: sinónimos de tinyint(1). Un valor cero se considera falso, los valores distintos de cero, verdadero.
        - **Bigint(x)**: Es un tipo de entero que dispone del mayor rango, que va entre 9000000000000000000 a 9000000000000000000 aproximadamente. Sin signo es de 0 a 10000000000000000000.
    - **Float (x,y)**: Este tipo de dato engloba los números con coma flotante. Su rango es de -3.4e+38 a -1.1e-38.
    - **Decimal (x,y)**: A diferencia que “float”, “decimal” permite ajustar el número de dígitos con la variable “x” y el número de decimales con la variable “y”.
- **Fechas y horas**: Con este tipo de datos, se podrán registrar como ya se supone, datos de fechas y hora con un formato específico:
    - Date: Este tipo de dato, representa una fecha con formato “YYYY-MM-DD” (Año-Mes-Día). El rango va de “1000-01-02” a “9999-12-31”.

1.5.8. Programación de la base de datos

Una vez introducido los conceptos básicos del lenguaje de programación de base de datos SQL, y el gestor MySQL, se pasará a los procedimientos seguidos para su creación:

1. Acceder a la base de datos mediante la consola.
2. Mediante lenguaje SQL programar la tabla.
3. Programar correctamente el tipo de dato que ha de contener cada campo.

1.5.8.1. Accediendo al servidor MySQL

Durante el proyecto se trabajó con un servidor remoto MySQL, instalado en un servidor con sistema operativo Linux. Por lo tanto, el ordenador donde se estaba desarrollando el proyecto, debía acceder mediante conexión IP al servidor.

Para ello, el primer paso es la instalación del programa MySQL en el propio ordenador, para así disponer de las herramientas necesarias para acceder a otros servidores MySQL remotamente.

El acceso al servidor se debía de realizar mediante comandos de la consola de Windows. Una vez ejecutada la consola, se debe acceder al siguiente archivo mediante comando, para así poder acceder al servidor:

- *C:\Archivos de Programa\MySQL\MySQL Server 5.6\bin\mysql -h 192.168.1.100 -u root*

Aplicando los comandos anteriores, se podrá acceder al servidor para así empezar a programar, apareciendo la siguiente ventana:

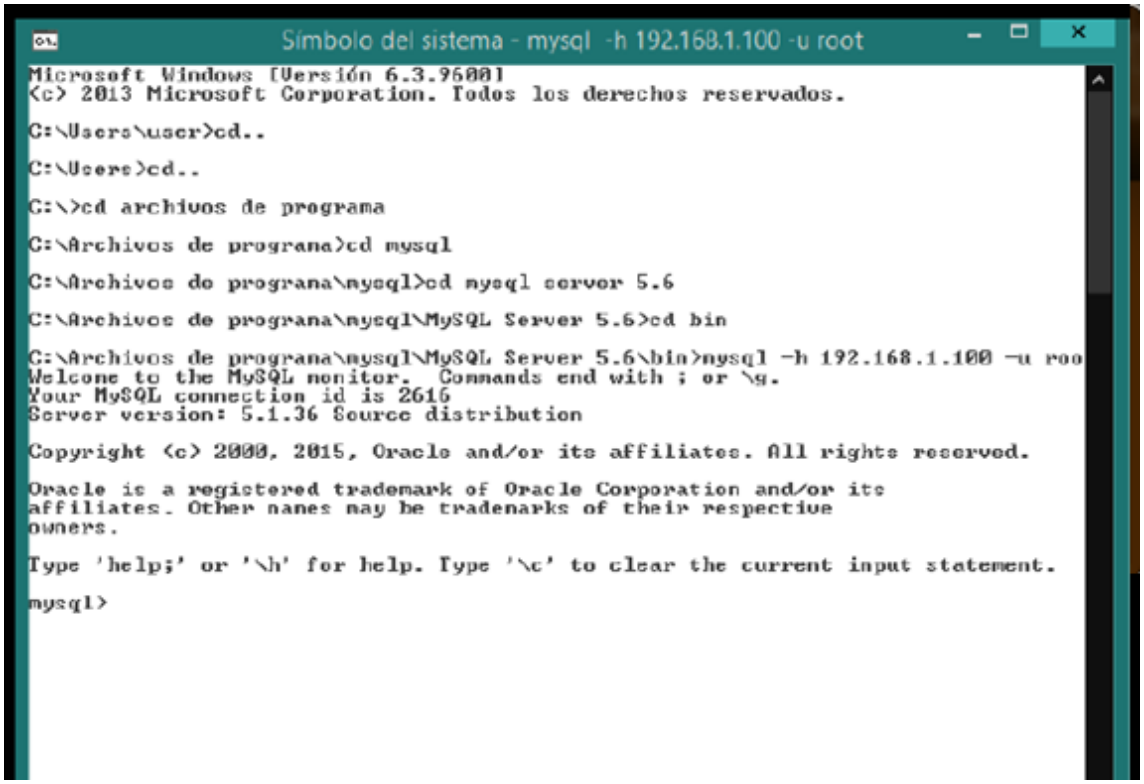


Fig. 27 Pantalla de inicio del servidor MySQL del proyecto.

1.5.8.2. Creación de la base de datos y tabla

Una vez se haya accedido al servidor, se creará la base de datos con los siguientes comandos en SQL:

- show databases;
- use test;
- create table registro (ID_linea int, ID_zona int, ID_maquina int, ID_dispositivo int, ID_fallo int, fecha_hora TIMESTAMP);

Como se observa en el código, el tipo de dato de cada campo a excepción de “fecha_hora” son de tipo “int”. Por lo cual, como se ha explicado previamente, se podrá almacenar datos numéricos enteros de gran tamaño.

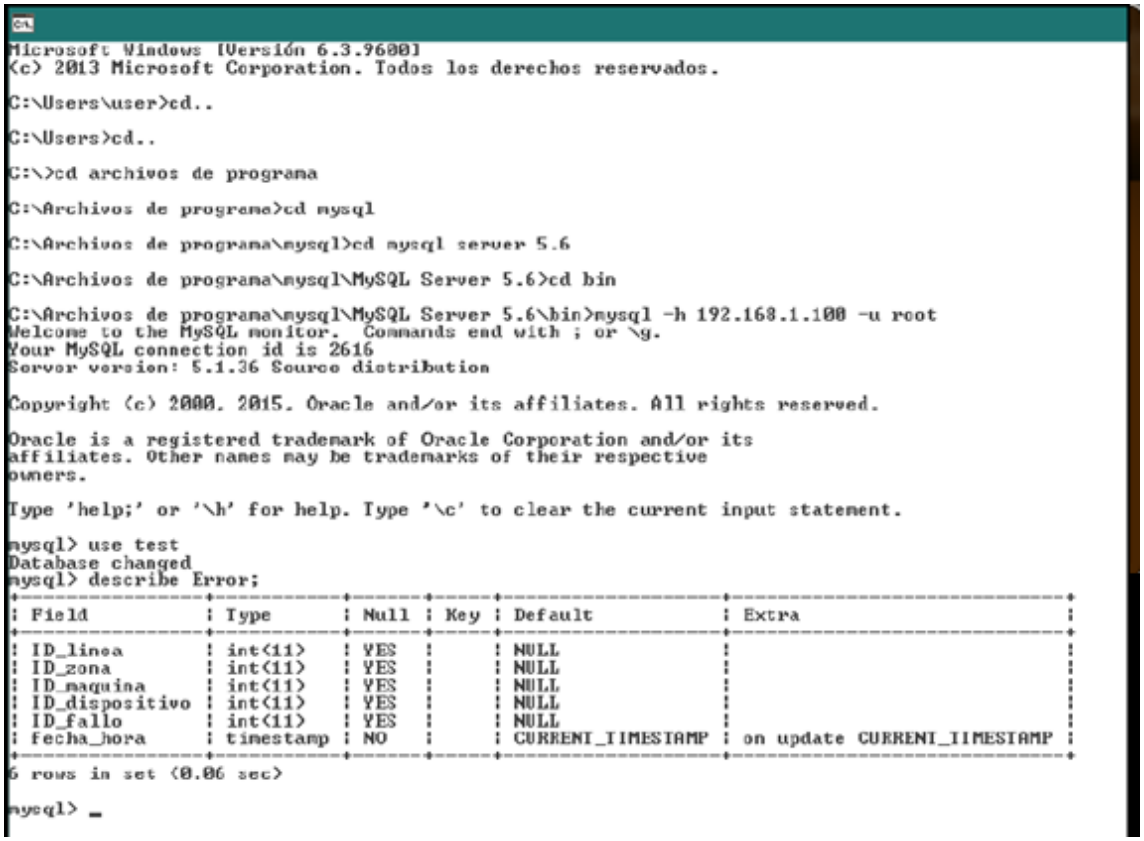


Fig. 28 Pantalla después de haber creado los campos necesarios para el proyecto.

En la imagen anterior, se puede observar la base de datos ya creada, después de haber introducido los comandos anteriores.

Con esta base de datos, ya se podrán registrar el código de error proveniente del PLC, que como se recuerda, estaba dividido para así poder almacenar con mayor claridad dicho código.

1.5.9. Diseño de la aplicación Android

En este apartado se detallará los aspectos básicos que deberá seguir la aplicación Android para su diseño:

- Será una aplicación Android sencilla, que muestre el código de error enviado por el PLC.
- Deberá establecer una conexión con el servidor PHP.
- Su función será simplemente mostrar el mensaje.
- Deberá estar abierta a cualquier petición siempre y cuando la app esté abierta.

Antes de describir el proceso que se ha seguido para el desarrollo de la aplicación, se realizará una breve introducción a la programación en Android, para así establecer un contexto dentro del mundo de la programación en Java.

1.5.9.1. Introducción a la programación en Android

El lenguaje de programación en Android es el Java, aunque no utilice los estándares establecidos de Java en sí, ya que solo utiliza la sintaxis y la semántica de Java, pero no incorpora la mayoría de bibliotecas de clases de Java y APIs que vienen incluidas a Java SE o ME, sin embargo, hay ciertas herramientas en el mercado para realizar una conversión.

Actualmente, existe un entorno de programación oficial y totalmente gratuito llamado Android Studio, ha sido diseñado específicamente para la programación en Android. Está basado en el software IntelliJ IDEA de JetBreains.

1.5.9.2. Primeros pasos para la creación de la aplicación

Los pasos para la creación del código de la aplicación, son algo similares a las vistas con el servidor PHP, que son las siguientes:

1. La aplicación, será un servidor TCP/IP que sea capaz de escuchar las peticiones del servidor PHP que se encuentre en la misma red.
2. El servidor se mantendrá siempre activado.
3. Cuando se reciba el dato lo mostrará en pantalla.
4. Volverá a su estado inicial.

1.5.10. Organigrama de la aplicación

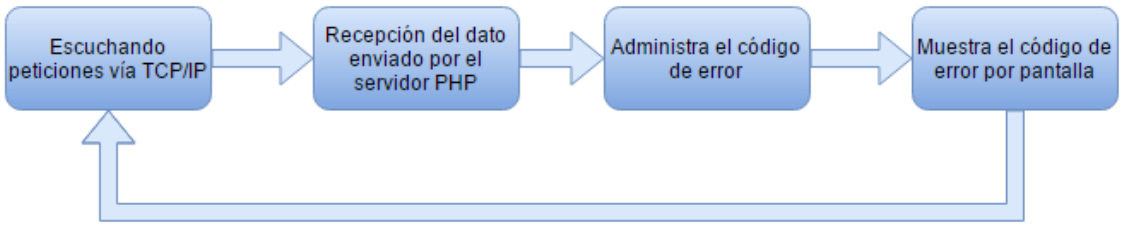


Fig. 29 Organigrama de la aplicación Android.

Como se observa en la imagen, el servidor Android seguirá dichos pasos para su ejecución, volviendo al estado de espera, una vez haya recibido el código de error y lo haya mostrado por pantalla para que así el operario lo identifique rápidamente.

1.5.11. Programación de la aplicación

A continuación, se detallará paso por paso los procedimientos a la hora de programación. Cabe destacar que para programar una aplicación Android, se debe programar varias partes y que su código se puede observar en el ANEXO II:

- La parte del código fuente que seguirá la aplicación será un fichero .java.
- La parte gráfica de la aplicación será un fichero .xml.
- La parte de permisos que deberá tener accesos la aplicación.

Para el código fuente (fichero.java), es decir, la lógica que seguirá la aplicación, se ha programado de la siguiente forma:

1. Se detallan las bibliotecas y variables a utilizar.
2. Se crea una función que será para crear el socket servidor, utilizando todas las funciones necesarias de funciones de conexión.
3. Las peticiones entrantes son escuchadas.
4. Las funciones de extracción de datos como IP del propio teléfono móvil, IP del PLC, para la identificación del dispositivo.
5. Se creará diferentes condicionantes donde si hay algún tipo de error en la aplicación, lo notifique.
6. Los datos que lee el servidor, los guarda en varias strings concatenadas para así almacenarlo en una variable que será la que se muestre por pantalla.
7. El servidor se mantiene activo.

En el apartado gráfico de la aplicación (fichero .xml):

1. Se configuran los diferentes layout de la aplicación.
2. El layout superior estará dedicado al logo de la empresa.
3. El layout inferior estará reservado para el código de error se muestre en esa sección.
4. Las dimensiones del layout se han configurado manualmente.

Respecto a los permisos, en esta sección, se programa los servicios de los que la aplicación deberá tener acceso:

1. A parte de los permisos predeterminados, se ha configurado para que la aplicación pueda tener permiso para acceder a la IP del móvil, para así el servidor pueda crear el socket.

1.5.11.1. Ejecución del programa

Para la ejecución del programa, se ha de instalar la aplicación en el teléfono móvil, para probar la aplicación, se debe activar en el móvil las aplicaciones de desarrollador para activar la depuración por USB. Esta opción permitirá al teléfono ejecutar la aplicación que compilemos en Android Studio.

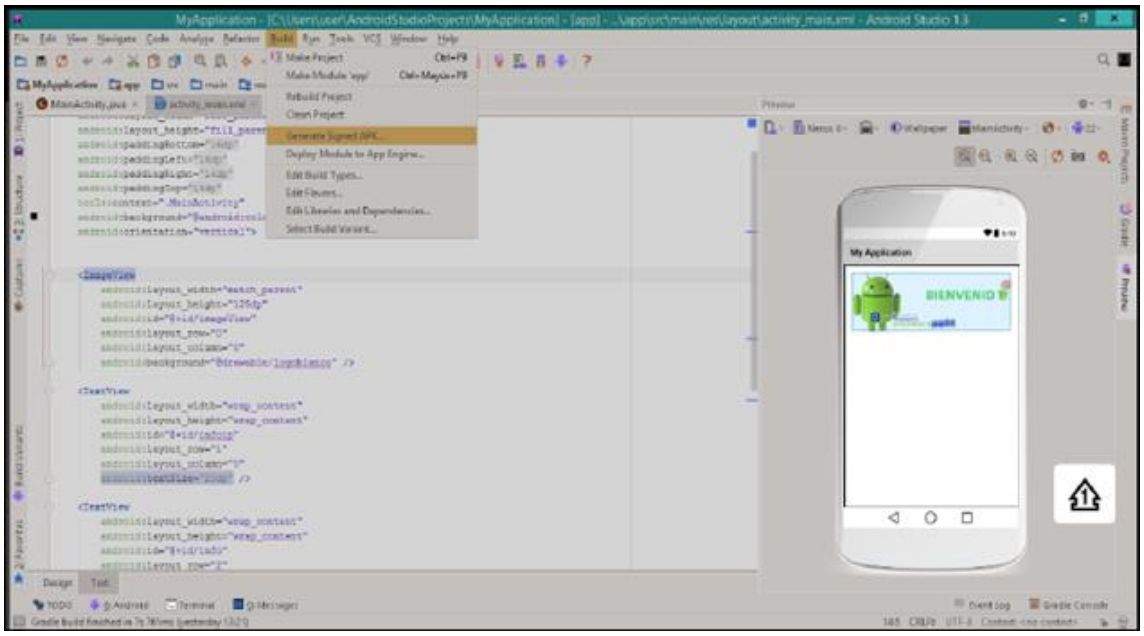


Fig. 30 Parte gráfica de la aplicación.

Una vez instalada la aplicación en el teléfono móvil, simplemente se tiene que iniciar como cualquier aplicación, y se iniciará el servidor Android de la aplicación.

1.6. Verificación del diseño

Finalizada la implementación, se deberá realizar una serie de pruebas para poder así verificar el correcto funcionamiento del diseño que se ha implementado como, por ejemplo:

- El correcto funcionamiento del PLC.
- El correcto encendido del servidor PHP.
- La buena recepción de datos del servidor.
- El registro del dato se haya registrado correctamente en la base de datos.
- La buena recepción del dato por parte de la aplicación.

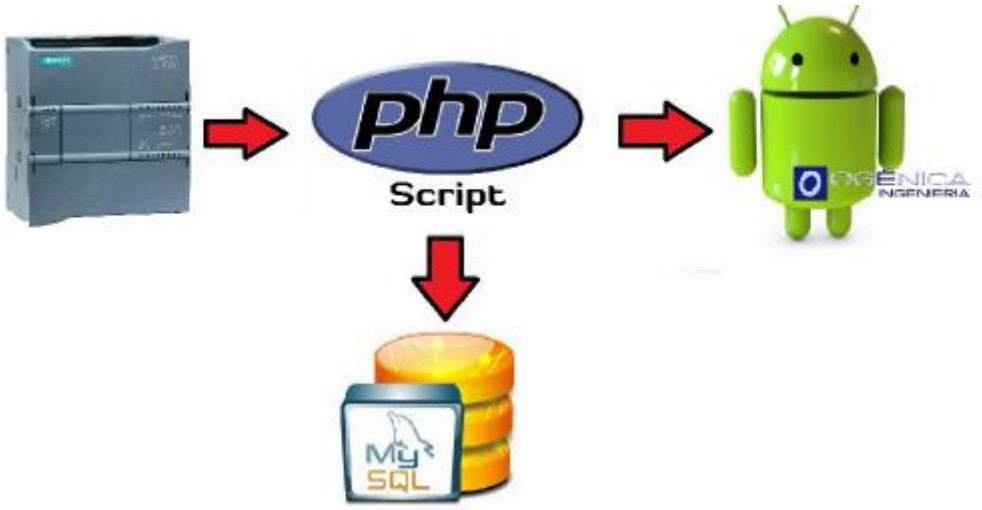


Fig. 31 Esquema ilustrativo del proyecto.

Una vez comprobado el resultado correcto de todos los componentes del proyecto, dicho proyecto recaerá sobre la empresa contratante, que deberá comprobar también el correcto funcionamiento.

1.7. Conclusiones

1.7.1. Objetivos cumplidos

Tras comprobar el correcto funcionamiento de todo el sistema de comunicación, se puede extraer conclusiones sobre el proyecto:

- Se ha conseguido emitir un código numérico a través de un bloque de comunicación del PLC.
- El servidor PHP es capaz de recibir el dato, y acondicionarlo.
- El servidor ha podido de registrar el código correctamente en cada campo creado.
- También el servidor ha establecido un puente de comunicación con la aplicación Android.
- La aplicación Android ha creado correctamente el servidor de escucha.
- La aplicación ha sido capaz de mostrar correctamente el código en pantalla.

Se puede concluir, que se han cumplido todos los objetivos principales del proyecto propuesto por la empresa.

1.7.2. Consideraciones a tener en cuenta

- La aplicación funciona solo en red local, ya que toda la comunicación se ha configurado para ello, debido a las limitaciones de hardware.
- La aplicación Android es básica, pudiendo ampliarse con muchas opciones más.
- No se ha podido comprobar el sistema, en un ambiente real de trabajo, por lo que la base de datos no hay errores reales registrados.
- Es un proyecto de investigación de unos meses de trabajo, por lo que se puede ampliar en todas sus ramas.

1.7.3. Problemas surgidos

- Uno de los primeros problemas surgidos, fue encontrar el tipo de dato correcto para enviar por parte del PLC, ya que cuando se enviaban otros tipos de datos como tipo integer, el servidor no lo recibía correctamente.
- Al principio del proyecto, el objetivo era establecer un registro directo del PLC con la base de datos, pero se llegó a la conclusión de que no se podía hacer, por ello, se creó un intermediario que es el servidor PHP.
- Otro de los problemas que surgieron, fue a la hora de configurar en TIA Portal la IP del PLC. La tarjeta de red del propio servidor necesita tener el mismo rango de IP que el PLC, si no es así, el programa no detecta correctamente el PLC.
- Para que el servidor PHP, pueda emitir por la tarjeta de red Wi-Fi a la base de datos externa y aplicación (pero en la misma red), la tarjeta Ethernet como la tarjeta Wi-Fi deben estar conectadas a través de un bridge de conexiones, porque surgieron diferentes problemas de comunicación. Al crearse el bridge de conexiones, Windows asigna sólo una IP a las dos tarjetas, por lo que se debe establecer dicha IP al servidor PHP.
- Por parte de la aplicación, los principales problemas fueron el encontrar el tipo de dato correcto para la aplicación Android, ya que los primeros datos enviados, se imprimían en pantalla con caracteres extraños.

2. PLIEGO DE CONDICIONES

2.1. Definición y alcance del pliego de condiciones

2.1.1. Objeto del pliego

El presente pliego de condiciones tiene por objeto detallar las distintas condiciones técnicas que se debe seguir para la correcta ejecución del proyecto que se detalla en este documento.

Este proyecto presenta el objetivo de diseñar un sistema de comunicación entre un PLC Siemens S7-1200 y una base de datos MySQL y una aplicación Android, para el registro de un código de error enviado por el PLC. Como nexo de comunicación, se encuentra el servidor PHP que hace de intermediario entre las diversas partes del proyecto. Como función de la aplicación Android, es la muestra del código de error a un operario para que actúe con rapidez sobre dicho fallo.

2.1.2. Descripción general del montaje

La ejecución del proyecto a grandes rasgos seguirá la siguiente lista:

1. Adquisición de los materiales.
2. Montaje del PLC con el módulo de interruptores en las entradas.
3. Conectar vía PROFINET el PLC al ordenador con el cable RJ-45.
4. Ejecutar el programa diseñado en TIA Portal, poniendo el PLC en modo “run”.
5. Arrancar el servidor PHP en el ordenador.
6. Iniciar aplicación Android.
7. Comprobación del software de TIA Portal, activando las diferentes entradas que activen el bloque de comunicación.
8. Comprobación de el correcto funcionamiento del PLC y sus módulos.
9. Comprobar que el servidor PHP esté en modo de escucha esperando la entrada de datos por parte del PLC.
10. Comprobar que el código se muestra correctamente por pantalla en la aplicación Android.

El encargado de la puesta en marcha del proyecto debe tener conocimiento de todos los elementos del sistema y de sus posibles errores ya sean de programación, de conexión, o de montaje.

2.2. Condiciones y normas de carácter general

El posible fallo de cualquiera de los componentes del equipo durante la realización del proyecto, se debería notificar directamente al director del proyecto, para la sustitución de dicho componente por uno de características similares siempre y cuando apruebe la sustitución. Si se produce algún accidente o fallo, la responsabilidad recae sobre el individuo que lo haya producido sin la debida autorización.

Es obligatorio la correcta comprobación de cada uno de los elementos del proyecto para comprobar si reúnen las especificaciones técnicas reflejadas en el datasheet proporcionada por el fabricante.

El proyecto deberá cumplimentar las normativas vigentes impuestas para este tipo de proyectos de los diferentes ámbitos en los que se trabaja.

2.3. Condiciones de los materiales

En este proyecto, tanto el software como el hardware proporcionado por la empresa contratista, son materiales estandarizados que no presentan ningún tipo de problemas para su adquisición.

El hardware (PLC) que se utiliza en este proyecto no podrá ser sustituido por otro de diferente marca por mucho que cumpla con las mismas características, debido a la incompatibilidad con el programa utilizado. Por otra parte, si se da el caso de la imposibilidad utilización del PLC dado por la empresa y es necesario el cambio por otro de diferente marca, se deberá adoptar las medidas necesarias para poder adaptar el código a dicho producto. El autor del proyecto no se hace responsabiliza del mal funcionamiento de la aplicación, ya que la solución adoptada está programada para los elementos aquí expuestos. La marca Siemens es una marca reconocida y bien valorada en el sector industrial no debería haber ningún impedimento a la hora de posibles suministros por parte de la marca.

2.3.1. Ordenador personal

Para un correcto funcionamiento del software utilizado en el proyecto, TIA Portal avisa que el ordenador debe de cumplir unos requisitos mínimos de hardware, para su correcta programación.

Este elemento lo proporciona la empresa contratante y es un parte importante del proyecto, ya que con el ordenador se debe programar todos los códigos que utiliza el proyecto.

Dada a la movilidad que requiere el proyecto, la empresa proporciona un ordenador portátil, suficiente en componentes, para cumplir con los requisitos mínimos que proporciona TIA Portal, que son los siguientes:

- **Requisitos de hardware**:
  - Procesador: CoreTM i5-3320M 3.3 GHz, similar o superior.
  - Memoria principal: 8 GB de memoria (recomendado) o más.
  - Disco duro: 300 GB SSD.
  - Gráficos: Mín 1920x1080
  - Pantalla: 15,6” de display.
  - Conexiones: Tarjeta de red con conexión Ethernet.
- **Requisitos de software**:
  - MS Windows 7 Home Premium SP1 (sólo para STEP 7 Basic).
  - MS Windows 7 Professional SP1.
  - MS Windows 7 Enterprise SP1.
  - MS Windows 7 Ultimate SP1.
  - Microsoft Windows 8.1 (sólo STEP 7 Basic).
  - Microsoft Windows 8.1 Pro.
  - Microsoft Windows 8,1 Enterprise.
  - Microsoft Server 2012 R2 Standard Edition.
  - MS Windows Server 2008 R2 Standard Edition SP1 (sólo para STEP 7 Professional).
- **Derechos de administrador**:
  - Se necesitan derechos de administrador para poder instalar en el PC el STEP 7 (TIA Portal) V13.

2.3.2. Autómata programable

El autómata programable es de la marca Siemens modelo S7-1200 CPU 1214C, que tiene las siguientes características generales:

- **General**:
  - Dimensiones A x A x P (mm): 110 x 100 x 75.
  - Peso: 475 g.
  - Disipación de potencia: 14 W.
  - Intensidad disponible (SM y bus CM): 1600 mA máx (5 V DC).
  - Intensidad disponible (24 V DC): 400 mA máx. (alimentación de sensores).
  - Consumo de corriente de las entradas digitales (24 V DC): 4 mA/entrada utilizada.
- **Características de la CPU**:
  - Memoria de usuario: 50 KB de memoria de trabajo / 2 MB de memoria de carga / 2 KB de memoria remanente.
  - E/S digitales integradas: 14 entradas/10 salidas.
  - E/S analógicas integradas: 2 entradas.
  - Tamaño de la memoria imagen de proceso: 1024 bytes de entradas (I)/1024 bytes de salidas (Q)
  - Área de marcas (M): 8192 bytes.
  - Ampliación con módulos de señales: 8 SMs máx.

Esta son una de sus características principales, además cumplirá con las leyes de seguridad eléctrica certificadas, incluyendo las protecciones:

- Cortocircuitos.
- Sobretensiones.
- Perturbaciones presentes en el laboratorio.
- Perturbaciones de red.
- Tensiones de red fuera de rango.

Además, incluye las señalizaciones necesarias para su correcta utilización y precauciones de tensión de alimentación y la que proporciona el PLC, ya que se debe alimentar correctamente y se debe comprobar si suministra la tensión marcada por el fabricante de 5 V.

2.3.3. Módulo de entrada

Como módulo de entrada, el proyecto utiliza un simulador de 8 entradas, un pequeño circuito impreso, con 8 interruptores que sustituye a las posibles entradas con dos posiciones lógicas, “1” ó “0”.

Este pequeño circuito, se conecta a las entradas del PLC.

2.3.4. Teléfono móvil

Durante el proyecto se ha utilizado distintos teléfonos móviles para las pruebas del programa. La aplicación será compatible con cualquier móvil que cumpla las siguientes características:

- Sistema operativo: Android 5.0 o superior.
- Depuración USB activada, en Opciones de desarrollo.

2.3.5. Condiciones de entrega

La compra de los materiales corresponde al contratista en cuestión que vaya a implementar el proyecto.

Antes de empezar a trabajar con los componentes, es recomendable comprobar el correcto funcionamiento de dichos componentes para asegurar que no producirán fallo debido a su mal funcionamiento. Los posibles ensayos deberán llevarse a cabo debido a la legislación vigente:

- UNE 20-512-74/2 “Fiabilidad de equipos y componentes electrónicos”.
- UNE 20-504-84 “Métodos de medida de las características antiparásita de filtros pasivos y otros dispositivos de perturbaciones radioeléctricos”.
- UNE 20-504-84 “Métodos de medida de las características antiparásita de filtros pasivos y otros dispositivos de perturbaciones radioeléctricos”.
- UNE 20-501-85/2 “Ensayos fundamentales, climáticos y de robustez”.

2.4. Condiciones de la ejecución

En este apartado se concretará el plan de ejecución del proyecto indicando los pasos que se deben seguir para su funcionamiento completo.

2.4.1. Programación

2.4.1.1. PLC y TIA Portal

La programación del PLC debe realizarse por un técnico de programación de PLC. Para empezar, se encargará de instalar el software de programación, y configurarlo adecuadamente, así como cualquier tipo de software adicional que sea necesario para el correcto funcionamiento del programa. El procedimiento a seguir es el siguiente:

- **Alimentación y comunicación del PLC y PC**: Antes de empezar a trabajar, se ha de comprobar que tanto el ordenador como el autómata programable, estén correctamente alimentados, el PLC dispone de una fuente de alimentación interna, por lo que no hace falta de una fuente externa. Por lo que respecta al PC, también deberá estar conectado a corriente con su respectivo cable.

  La conexión PLC/PC se establece gracias al cable RJ-45, que un extremo se conectaría al PLC, que se encuentra en la parte inferior, y el otro extremo al conector del PC.

- **Instalación del software**: Se deberá instalar el software TIA Portal con STEP 7 en su versión más actual posible, así como todo el software adicional que se requiera como drivers de Microsoft que no se dispongan en el PC a utilizar.
- **Diseño del automatismo**: Para la programación del automatismo, hay que matizar que consta de dos partes, la del código de error y la del bloque de comunicación TCP/IP.
    - **Preparación de datos**: Antes de empezar a programar los diferentes bloques, se ha configurar el tipo de dato del código de error, que será un contenedor de datos llamado “struct” que contendrá 5 vectores de 5 bytes. Por otra parte, el bloque de comunicaciones, también debe de tener sus datos específicos de funcionamiento, que los detalla en el manual de TIA Portal, donde el dato de entrada, será el enviado por los bloques MOVE.

- **Código de error**: Para el código se deben utilizar 20 bloques MOVE 5 series de 4 bloques en paralelo, donde a cada entrada del bloque se le proporcionará un dígito del código. El código seguía la siguiente serie 000:000:000:0000, que cada parte es el código de identificación. Este conjunto de bloques, se colocarán a la entrada que se requiera monitorizar para posibles fallos.
- **Bloque de comunicación**: Para el bloque de comunicación, se utilizará el bloque TSEND_C, que se configurará según los datos que se hayan creado previamente además del dato que modifican los bloques MOVE.

2.4.1.2. Servidor PHP

A lo que corresponde al servidor PHP su código fuente se puede encontrar en ANEXO I, los pasos para su programación deben ser los siguientes:

- **Creación del socket servidor**: El primer paso, es establecer mediante las funciones correspondientes, la creación del socket estableciendo los parámetros y protocolos de conexión TCP. Este socket se deberá vincular a una IP y un puerto de enlace, que este caso será el del PC que se cree dicho socket. Si está todo correctamente configurado, el socket abrirá un puerto de escucha para poder leer los datos recibidos y se mantendrá en estado de espera hasta la entrada de datos.
- **Entrada del paquete de datos**: Se debe programar que el socket acepte la entrada de dato, que a su vez el dato se guardará en un buffer llamado “input”. Se deberá imprimir por pantalla el paquete de datos recibido, y dicho paquete se guardará en diferentes variables: ID_linea, ID_zona, ID_maquina, ID_dispositivo, ID_fallo. Se creará un filtro ya que, si el paquete de datos es nulo, no se creará conexión con la base de datos.
- **Conexión con el servidor MySQL**: En este caso, se debe crear un socket de tipo cliente, es decir, que proporcione datos al servidor MySQL, para ellos se establece el código necesario para su conexión.
- **Consulta al servidor MySQL**: Una vez conectado con el servidor, se realizará el registro de las diferentes variables, en los campos correspondientes que se hayan creado previamente en la base de datos. Una vez realizada la consulta, se cerrará la conexión con el servidor de la base de datos.
- **Conexión con el servidor Android**: Se creará un cliente para que establezca la conexión con el servidor Android, que a dicho servidor se le proporcionará los datos de fecha y hora que tengan el servidor PHP en el momento de la conexión, para que así se muestre en la pantalla del móvil la hora y la fecha del envío. Finalmente se cerrará la conexión.

2.4.1.3. Base de datos MySQL

El servidor MySQL propiedad de la empresa, se encuentra en una red local. La programación de dicha base de datos se seguirán los siguientes procedimientos:

- Establecer la conexión a la base de datos mediante la consola de comandos de Windows, aplicando los diferentes comandos de conexión.
- Una vez establecida la conexión, se debe acceder a la base de datos que se desee.
- Para crear la tabla donde se registrará los datos correspondientes, se debe crear los campos ID_linea, ID_zona, ID_maquina, ID_dispositivo, ID_fallo, todos de tipo integer y, por último, fecha_hora de tipo timestamp.
- Una vez creado ya se tendrá la tabla con la que se registrarán todos los errores.

2.4.1.4. Aplicación Android

En lo que corresponde a la aplicación Android, se deben seguir los siguientes pasos, pero hay que recordar que esta aplicación está programada en Android Studio. La aplicación constará de tres partes de programación, la primera es el código fuente de las funciones de conexión del servidor, la segunda la parte visual de la aplicación. Por último, la parte de permisos que deberá tener la aplicación sobre el teléfono móvil y que el código fuente de la aplicación está disponible en el ANEXO II.

- **Fichero java**: En primer lugar, se han de colocar las librerías que se van a utilizar dependiendo de los tipos de funciones, en este caso, funciones para la creación de sockets. Seguidamente, los pasos a seguir para la creación del socket difieren del servidor PHP, se debe crear la diferentes variables y clases que definirán al socket con la IP del teléfono. Por otra parte, el socket se mantendrá en estado de escucha hasta la entrada de algún tipo de dato. Como en el servidor PHP, se creará un buffer donde almacenará dicho paquete, y que se concatenará para mostrarlo por pantalla, así como la IP del servidor, fecho y hora. Finalmente se cerrará la conexión, pero volviendo al estado de escucha.
- **Fichero gráfico**: Android Studio nos permite importar imágenes que se desee y configurar el layout de la página principal. En la parte superior quedará reservado para el logotipo de la empresa contratante y la parte central para mostrar el código de error.
- **Fichero de permisos**: En este fichero hay que modificar una pequeña parte del código para que la aplicación pueda tener acceso a la IP.

2.4.2. Implementación

Al ser un proyecto de investigación para la empresa, ella determinará si el sistema está preparado para su funcionamiento en la zona de trabajo, así como la instalación del software y de su ejecución.

2.4.3. Personal

El personal que se contrate para la manipulación o instalación del sistema de comunicación, deberá ser conocedor de cada aspecto del sistema debido a su complejidad. El personal deberá estar familiarizado con los diferentes aspectos que se tocan en el proyecto, como programación de PLC Siemens, base de datos MySQL, aplicaciones Android o programación en PHP.

2.4.4. Pruebas de servicio

- Se debe comprobar que el PC y PLC funcionan correctamente, así como la correcta conexión entre el programa TIA Portal y PLC, para que el sistema de comunicación esté listo para un funcionamiento en un proceso real y no simulado.
- Se debe comprobar la correcta conexión del servidor PHP, y que se encuentre en modo escucha cuando se active.
- Se debe comprobar que el registro en la base de datos se ha realizado correctamente.
- Se debe ejecutar la aplicación Android, si arranca correctamente y si muestra correctamente el código de error por pantalla.
- Se debe comprobar que la fecha y hora es correcta del servidor PHP.

2.5. Condiciones de mantenimiento

En este apartado se valorarán los aspectos relativos para el correcto mantenimiento de los elementos que componen el proyecto.

2.5.1. Mantenimiento de los programas

Por parte del servidor, se necesita comprobar periódicamente las IP están correctamente establecidas en el servidor PHP, java y en el programa de TIA Portal.

Bajo ningún concepto se pueden realizar modificaciones de los programas sin supervisión, ya que el mínimo cambio supondría un error total del sistema. En caso de algún error irreparable se deberá realizar un backup del sistema total.

2.5.2. Soporte de copia de seguridad

El programa se podrá entregar vía USB, para su rápida instalación y portabilidad. No es necesario un mantenimiento del USB, pero sí su utilización con precaución y correcta utilización, extrayendo correctamente el USB del PC, que podrían producir pérdidas irreparables.

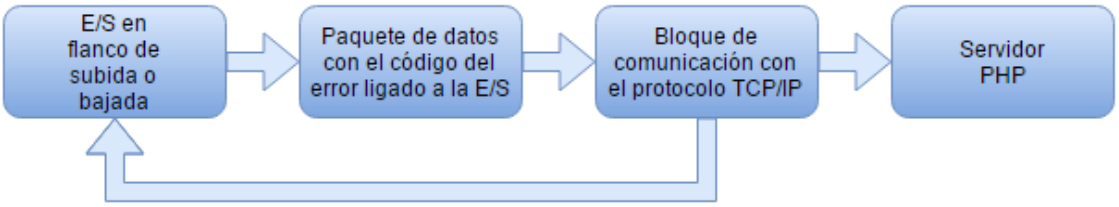
Se guardará una copia en el servidor de la empresa.

3. PLANOS

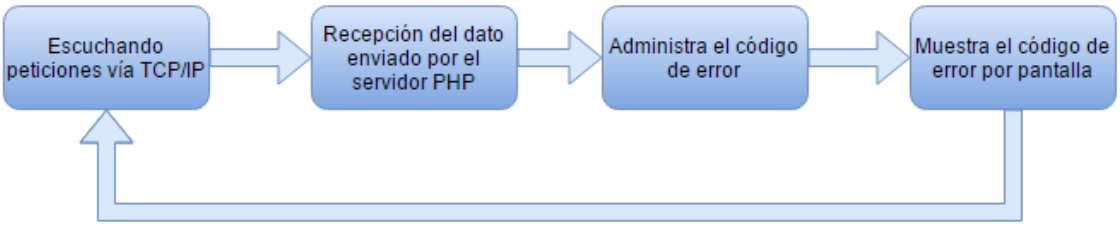
En este punto, se recopilarán los diferentes flujogramas de cada parte del sistema que han ido apareciendo a lo largo del proyecto para su fácil búsqueda.

Como ya se ha explicado en los anteriores puntos los flujogramas corresponderán al camino que recorre el dato del código de error enviado desde que el PLC lo envía hasta cuando lo muestra por pantalla del teléfono móvil.

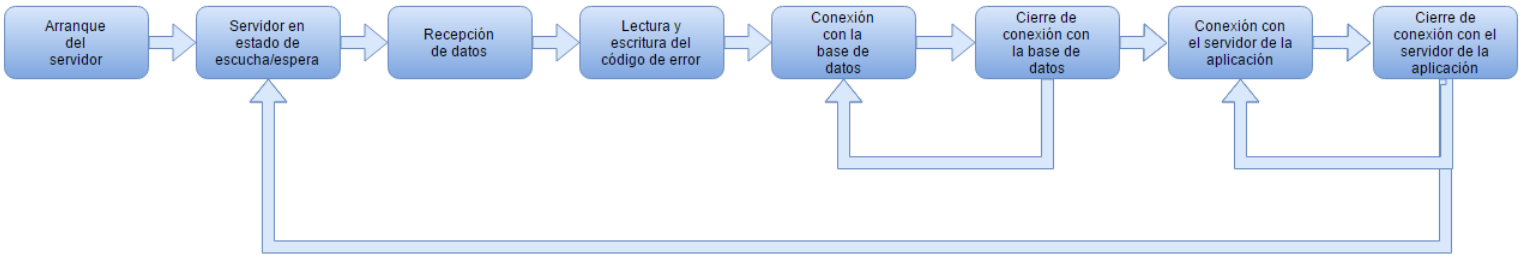
3.1. Flujograma del automatismo en TIA Portal



3.2. Flujograma de la aplicación Android



3.3. Flujograma del servidor PHP



4. PRESUPUESTO

4.1. Introducción

En el presente apartado se especificará detalladamente el presupuesto del proyecto, con el objeto de concretar una estimación de los diferentes gastos económicos que supone el sistema que se ha realizado.

Se realizará un presupuesto de todos los gastos y componentes utilizados, ya sean de hardware, software o recursos humanos.

La empresa Ogénica Ingeniería SL ha proporcionado todos los recursos necesarios para la producción del sistema de comunicación. Los precios de los elementos que se reflejen en este documento estarán basados en la fecha de la redacción de dicho documento.

4.2. Recursos de hardware

Los recursos de hardware que han sido utilizados para la producción correcta del sistema de comunicación son:

- 1 PC.
- 1 PLC Siemens S7-1200.
- 1 servidor NAS.
- 1 periférico de emulación de entradas para el PLC.
- 1 cable RJ-45.
- 1 ratón.

Concepto	**Cantidad (Ud.)**	**Precio/Unidad (€/Ud.)**	**Precio total (€)**
Ordenador portátil Lenovo G70-80	1	424,50	424,50
Ratón inalámbrico Logitech M185	1	12,95	12,95
Servidor NAS de 8 TB Synology DS214play	1	340,68	340,68
Cable de red Gigabit Ethernet Lan CAT.7 (RJ-45)	1	8,85	8,85
PLC Siemens Simatic S7 1200 CPU 1214C	1	298,86	298,86
Simatic S7-1200, modulo simulador SIM1274 8 Entradas	1	80,09	80,09
Total			**1165,93**

4.3. Recursos de software

Por lo que concierne al software, en el sistema se ha trabajado con todos los programas open source posibles, como la base de datos MySQL instalado en el servidor, Android Studio o los editores de texto Notepad ++.

Por lo tanto, el único recurso que ha tenido que utilizar la empresa en este apartado ha sido comprar una licencia para el programa Siemens SIMATIC STEP 7 Basic V14 (TIA Portal).

Concepto	Cantidad (Ud.)	Precio/Unidad (€/Ud.)	Precio total (€)
Licencia Siemens SIMATIC STEP 7 Basic V14	1	1	293
Total			**293**

4.4. Recursos humanos

En este apartado de recursos humanos del proyecto derivan de las horas de trabajo del proyectista, con un salario pactado por los sindicatos. En este proyecto debe realizarse por un ingeniero técnico para su montaje, que tienen un sueldo aproximado de 10,71 €/h.

El proyecto se ha desarrollado en el total de un tiempo de dos meses, con 44 días efectivos en una jornada laboral de 8 h.

Tarea	Horas (h)	Coste/Hora (€/h)	Coste total (€)
Análisis	20	10,71	214,20
Diseño	200	10,71	2142
Implementación	100	10,71	1071
Comprobación	32	10,71	342,72
Total			**3769,92**

4.5. Coste total del proyecto

Tras los estudios anteriores, de los tres aspectos básicos del proyecto, se puede englobar todos los costes en una tabla total.

Concepto	Coste (€)
Hardware y material	1165,93
Software	293
Mano de obra	3769,92
Total	**5228,85**

5. AGRADECIMIENTOS

Para finalizar, con este documento quería agradecer a la empresa Ogénica Ingeniería SL, por su confianza en un alumno sin experiencia para la total realización de este proyecto y ofrecerle todos los recursos necesarios para su diseño e implementación.

Este proyecto ha ayudado para afianzar conocimientos en los diversos campos en los que se ha trabajado, diseño de servidores, implementación de automatismos, base de datos, etc.

He querido realizar este proyecto realizado en prácticas de empresa en formato TFG, ya que me parece interesante los diferentes campos en los que se trabaja, principalmente el desarrollo de automatismos siendo una parte de enseñanza del grado.

ANEXOS

ANEXO I: Código de programación del servidor PHP

```
<?php
set_time_limit(0); // Dejamos en cero para que la conexion acepte la
conexiones a ese y esta nunca
se cierre.
ob_implicit_flush(); // Para poder ver por pantalla las ejecuciones.
$ip='192.168.0.107'; // La IP de nuestro servidor.
$puerto=2000; // El puerto de enlace al que queremos escuchar a nuestro
cliente.
$puerto_cliente=2001; // Puerto al que enviaremos datos a nuestro cliente.
$ip_cliente='192.168.0.101'; // IP de nuestro cliente.
//EMPEZAMOS CREANDO UN SOCKET SERVIDOR.
if(($sock=socket_create(AF_INET,SOCK_STREAM,SOL_TCP))===false){
echo "socket_create()fallo: razon:" .socket_strerror(socket_last_error())."\n";
} // Creación del socket, eligiendo los parámetros de conexión, y protocolos a
utilizar, en este caso
TCP.
if(socket_bind($sock, $ip, $puerto)===false){
echo "socket_bind() fallo: razon" .socket_strerror(socket_last_error($sock))."\n";
} // Vincula nuestro socket creado a una IP y un puerto de enlace.
if (socket_listen($sock)===false){
echo "socket_listen() fallo: razon:"
.socket_strerror(socket_last_error($sock))."\n";
} //El socket abre un puerto de escucha para poder leer los datos recibidos.
echo "\n\n*.*.*.*.* Bienvenido al servidor de prueba de TCP/IP *.*.*.*.*\n\n";
echo "IP: $ip \nPUERTO: $puerto\n\n";
//EL SERVIDOR SE MANTIENE A LA ESPERA DE LA RECEPCIÓN DE
DATOS.
//CUANDO LO RECIBE EJECUTA EL SIGUIENTE CÓDIGO.
while(1){
$cliente[++$i]=socket_accept($sock); // El socket acepta cada vez que le entra
un dato.
$input=trim((socket_read($cliente[$i],1024))); // Guardamos en nuestra variable
$input
(buffer) lo que lea el socket.
echo "\n\nDato recibido: \n\n$input"; // Imprimimos por pantalla el dato recibido.
list($ID_linea,$ID_zona,$ID_maquina,$ID_dispositivo,$ID_fallo)=explode(':',$inp
ut); //
Guardamos cada uno de los vectores que le enviemos con el PLC en variables
diferentes.
// EN LA SIGUIENTE CONDICIÓN APLICAREMOS UN FILTRO PARA
COMPROBAR SI $INPUT ES NULA O NO Y ASÍ NO CREAR REGISTROS
VACÍOS, SI
DEVUELVE FALSE SE REALIZARÁ EL REGISTRO EN MYSQL.
if(empty($input)){
echo "\n\n La variable es NULA.\n\n"; // Imprimimos por pantalla que la variable
es nula y
```

```
cerramos la conexión.
echo "Conexion global cerrada.";
socket_close($cliente[$sock]);
} elseif(is_numeric($ID_fallo)) { //En este condicionante aplicamos un filtro para
comprobar si el primer valor de nuestro vector es numérico.
// REALIZAMOS LA CONSULTA A NUESTRO SERVIDOR MYSQL.
$conexion=mysqli_connect("192.168.1.100","root","","test") or die ("Problemas
de
conexión"); // Abrimos una conexión con nuestro servidor MySQL.
$hoy = date("Y-m-d H:i:s"); // Obtenemos la fecha y hora del servidor dándole el
formato
adecuado para que coincida con el tipo TIMESTAMP de MySQL y lo
guardamos en la variable
'hoy'.
mysqli_query($conexion,"insert into Error
(ID_linea,ID_zona,ID_maquina,ID_dispositivo,ID_fallo, fecha_hora) values
('$ID_linea','$ID_zona','$ID_maquina','$ID_dispositivo','$ID_fallo','$hoy')")
or die ("Problemas en el select\n".mysqli_error($conexion)); // Hacemos la
consulta que
queramos con leguaje SQL, si la conexión falla, el programa dejará de
ejecutarse.
mysqli_close($conexion); //Cerramos la conexión con nuestro servidor MySQL.
echo "\n\nRegistro realizado con fecha: '$hoy'";
//CREAMOS UN SOCKET CLIENTE PARA QUE ENVIE DATOS A NUESTRO
SERVIDOR ANDROID.
$socket=socket_create(AF_INET,SOCK_STREAM,SOL_TCP);// Creamos de
nuevo otro
socket, pero esta vez un socket cliente para enviar datos
if($socket===false){
echo "Socket_create() falló razón:".socket_strerror(socket_last_error());
} else{
echo "\n\nSocket cliente creado. \n\n";
} echo "\n\nIntentando conectar a
'$ip_cliente' de nuestro cliente, en el puerto
'$puerto_cliente'...\n\n";
$result=socket_connect($socket,$ip_cliente,$puerto_cliente); //Conectamos
con nuestro
socket.
if($result===false){
echo "socket_connect()
fallo.\nRazon:".socket_strerror(socket_last_error($socket));
} else {
$fecha="Fecha y hora del registro: ".$hoy;
$dato_cliente=$input." ".$fecha; // Concatenamos el dato de error con la fecha
actual para mostrarlo en la aplicación.
echo "Conectado.\n\n";
echo "Enviando codigo de error: $input \n\n $hoy\n\n";
socket_write($socket,$dato_cliente, 1024); // Escribimos en nuestro socket el
dato de error
```

```
junto con la fecha del registro.
echo "Ok\n\n";
echo "Cerrando socket cliente\n\n";
socket_close($socket);//Cerramos conexión con nuestro socket cliente.
echo "Cerrado.\n\n";
}}
}s
ocket_close($sock);
echo "Conexion global cerrada.";
?>
```

ANEXO II: Código de programación de la aplicación Android

Parte de la lógica, archivo .java

```
package com.example.user.myapplication;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
import java.util.Enumeration;
import android.os.Bundle;
import android.app.Activity;
import android.widget.TextView;
import com.example.androidserversocket.R;
public class MainActivity extends Activity {
TextView info, infoip, msg;
String message = "";
ServerSocket serverSocket;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
info = (TextView) findViewById(R.id.info);
infoip = (TextView) findViewById(R.id.infoip);
msg = (TextView) findViewById(R.id.msg);
infoip.setText(getIpAddress());
Thread socketServerThread = new Thread(new SocketServerThread());
```

```
socketServerThread.start();
}
@Override
protected void onDestroy() {
super.onDestroy();
if (serverSocket != null) {
try {
serverSocket.close();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
private class SocketServerThread extends Thread {
static final int SocketServerPORT = 2001;
int count = 0;
@Override
public void run() {
Socket socket = null;
DataInputStream dataInputStream = null;
DataOutputStream dataOutputStream = null;
try {
serverSocket = new ServerSocket(SocketServerPORT);
MainActivity.this.runOnUiThread(new Runnable() {
@Override
public void run() {
info.setText("Puerto: "
+ serverSocket.getLocalPort());
}
```

```
});
while (true) {
socket = serverSocket.accept();
dataInputStream = new DataInputStream(
socket.getInputStream());
dataOutputStream = new DataOutputStream(
socket.getOutputStream());
String messageFromClient;
messageFromClient = dataInputStream.readLine();
count++;
message += "Error "+count + "\n" + "IP del cliente: "+socket.getInetAddress()
+"\n\n"
+ "Código de error: " + messageFromClient + "\n\n";
MainActivity.this.runOnUiThread(new Runnable() {
@Override
public void run() {
msg.setText(message);
}
});
}
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
final String errMsg = e.toString();
MainActivity.this.runOnUiThread(new Runnable() {
@Override
public void run() {
msg.setText(errMsg);
}
});
```

```
} finally {
if (socket != null) {
try {
socket.close();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
if (dataInputStream != null) {
try {
dataInputStream.close();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
if (dataOutputStream != null) {
try {
dataOutputStream.close();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
}
}
private String getIpAddress() {
String ip = "";
```

```
try {
Enumeration<NetworkInterface> enumNetworkInterfaces = NetworkInterface
.getNetworkInterfaces();
while (enumNetworkInterfaces.hasMoreElements()) {
NetworkInterface networkInterface = enumNetworkInterfaces
.nextElement();
Enumeration<InetAddress> enumInetAddress = networkInterface
.getInetAddresses();
while (enumInetAddress.hasMoreElements()) {
InetAddress inetAddress = enumInetAddress.nextElement();
if (inetAddress.isSiteLocalAddress()) {
ip += "IP local: "
+ inetAddress.getHostAddress() + "\n";
}
}
}
} catch (SocketException e) {
// TODO Auto-generated catch block
e.printStackTrace();
ip += "¡Error! " + e.toString() + "\n";
}
return ip;
}
}
```

Parte gráfica de la aplicación .xml

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity"
android:background="@android:color/white"
android:orientation="vertical">
<ImageView
android:layout_width="match_parent"
android:layout_height="125dp"
android:id="@+id/imageView"
android:layout_row="0"
android:layout_column="0"
android:background="@drawable/logoblanco" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/infoip"
android:layout_row="1"
android:layout_column="0"
android:textSize="@dimen/tm2" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/info"
android:layout_row="2"
android:layout_column="0"
android:textSize="@dimen/tm2" />
<ScrollView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@android:color/white"
android:layout_row="4"
android:layout_column="0">
<TextView
android:id="@+id/msg"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textStyle="normal|bold"
android:textColor="@android:color/holo_red_dark" />
</ScrollView>
</GridLayout>
```

ANEXO III: Datasheet bloques de comunicación de TIA Portal

FAQ • 02/2017

Open User Communication with TSEND_C and TRCV_C

SIMATIC S7-1200 CPU

https://support.industry.siemens.com/cs/ww/en/view/67196808

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com.

Table of contents

1 Introduction

You can use the open user communication by means of the TSEND_C and TRCV_C instructions for data exchange over the integrated PROFINET interface of the S7-1200 CPU.

In STEP 7 (TIA Portal) you will find the TSEND_C and TRCV_C instructions in the "Instructions" task card in the "Communication > Open User Communication" palette.

Table 1-1

Instruction	Description
TSEND_C	The TSEND_C instruction is executed asynchronously and has the functions below: • Configure and establish communication connection • Send data through the existing communication connection • Disconnect communication connection
TRCV_C	The TRCV_C instruction is executed asynchronously and has the functions below: • Configure and establish communication connection • Receive data through the existing communication connection • Disconnect communication connection

The protocols below are supported for this:

- ISO-on-TCP
- TCP
- UDP

Description of the sample program

The sample program was created in STEP 7 (TIA Portal V14). The project consists of one S7-1200 CPU and one S7-1500 CPU including hardware configuration and user program.

Marker byte 10 (MB10) is configured as clock marker byte in both CPUs.

The S7 program contains the call of the "TSEND_C" and "TRCV_C" instructions and the parameterization of the ISO-on-TCP connection for data exchange between the S7-1200 CPU and S7-1500 CPU. The connection parameters for establishing the ISO-on-TCP connection are saved in the "GeneralData" data block.

The user program consists of the components below.

Table 1-2

Block	Symbolic name	Description
OB100	Startup	Startup OB
OB1	Main	The functions FC100 "FC_SEND" and FC200 "FC_RECV" are called in OB1.
FC100	FC_SEND	The FC100 "FC_SEND" function calls the TSEND_C instruction internally to send data through an ISO-on-TCP connection.
FC200	FC_RECV	The FC200 "FC_RECV" function calls the TRCV_C instruction internally to receive data through an ISO-on-TCP connection.
DB3	SendData	The sent data is stored in the DB3 data block.
DB5	RecvData	The received data is stored in the DB5 data block.

2 Sample Program

2.1 OB100

The OB100 is a startup OB and is run when the S7-1200 CPU is restarted (warm start). The following variables are set to value “1”, to establish two ISO-on-TCP connections and enable the job to receive data.

- “GeneralData”.contSend
- “GeneralData”.contRecv
- “GeneralData”.enable

2.2 OB1

OB1 is called cyclically. The functions FC100 "FC_SEND" and FC200 "FC_RECV" are called in OB1.

Figure 2-1

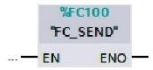


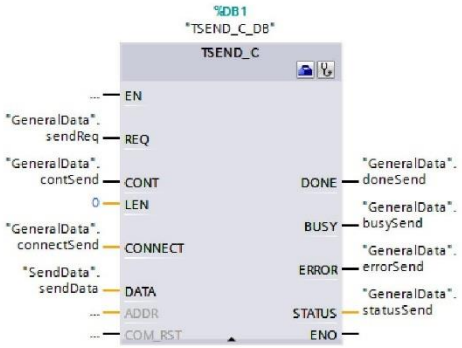
Figure 2-2

2.3 FC100 "FC_SEND"

The FC100 "FC_SEND" function calls the TSEND_C instruction internally to establish a communication connection over ISO-on-TCP, TCP or UDP and send data through the communication connection.

The following figure shows the call of the TSEND_C instruction.

Figure 2-3



2.3.1 Inputs and Outputs of the TSEND_C Instruction

Inputs

The table below gives an overview of

- the inputs of the TSEND_C instruction
- the variables assigned to the inputs

Table 2-1

Input	Data type	Variable	Description
REQ	BOOL	"GeneralData".sendReq	Starts the send job at a rising edge.
CONT	BOOL	"GeneralData".contSend	Controls the communication connection: 0: Disconnects the communication connection. 1: Establishes and maintains the communication connection. If CONT=1, the TSEND_C instruction configures and establishes a communication connection. Once the connection has been configured and established, it is maintained and monitored automatically by the S7-1200 CPU.

Input	Data type	Variable	Description
LEN	UINT	0	Maximum number of bytes that can be sent with the job. **Note** If you use a send area with optimized access at the DATA parameter, the value "0" must be used at the LEN parameter.
CONNECT	VARIANT	"GeneralData".connectSend	Pointer to the structure of the connection description: • For ISO-on-TCP, use the TCON_IP_RFC system data type. For a description, refer to chapter 2.5 • For TCP, use the TCON_IP_V4 system data type. For a description, refer to chapter 2.6
DATA	VARIANT	"SendData".sendData	Pointer to the send area that contains the address and length of the data to be sent.

Outputs

The table below gives an overview of

- the outputs of the TSEND_C instruction
- the variables assigned to the outputs

Table 2-2

Output	Data type	Variable	Description
DONE	BOOL	"GeneralData".doneSend	Status parameter with the values below: 0: Job not yet started or still being executed. 1: Job executed error-free.
BUSY	BOOL	"GeneralData".busySend	Status parameter with the values below: 0: Job not yet started or already terminated. 1: Job has not yet terminated. A new job cannot be started.
ERROR	BOOL	"GeneralData".errorSend	Status parameter with the values below: 0: No error 1: Error occurred.
STATUS	WORD	"GeneralData".statusSend	Status of the instruction

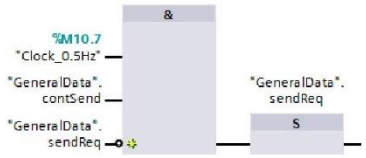
2.3.2 Start Send Job

The send job is started by the clock marker M10.7 and controlled with the "GeneralData".sendReq and "GeneralData".contSend variables.

When data is sent, the CONT input of the TSEND_C instruction must be set to the value "1" in order to establish the communication connection. The CONT input is set to the value "1" by means of the "GeneralData".contSend variable. This means that when "GeneralData".contSend is set to the value "1", the send job can be started.

When the send job is running, "GeneralData".reqSend is set to the value "1", which means that no new send job can be started as long as this send job is running.

Figure 2-4



2.3.3 Establish and Maintain the Communication Connection

The "GeneralData".contSend variable is set permanently to the value "1" when the S7-1200 CPU is restarted (warm restart).

The CONT input of the TSEND_C instruction is set permanently to the value "1" by means of the "GeneralData".contSend variable in order to establish and maintain the communication connection.

If the S7-1200 CPU goes into STOP mode, the existing communication connection is aborted and the configured communication connection is removed. You must once again execute the TSEND_C instruction to reconfigure and re-establish the communication connection.

2.3.4 LEN

A send area with optimized access is used at the DATA parameter in this sample program. For this reason the value “0” is used at the LEN parameter.

2.3.5 Send Area

In this sample program, the send area “SendData”sendData is defined with optimized access.

The send area is 100 bytes long and the data to be sent is contained in data block DB3.

2.3.6 Reset "GeneralData".sendReq

If there is no send job running, the BUSY output of the TSEND_C instruction has the value "0" and "GeneralData".sendReq is reset to the value "0". This means that a new send job can only be triggered once the previous job has been completed.

Figure 2-5

2.3.7 Save STATUS of the TSEND_C Instruction

If the TSEND_C instruction is executed successfully or with errors, the status of the TSEND_C instruction is saved in the "GeneralData".statusSendSave variable. The status informs you of the cause if the send job is not running.

Figure 2-6

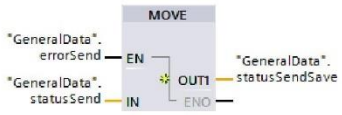
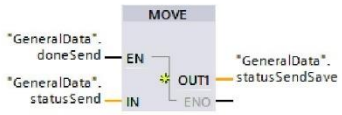


Figure 2-7

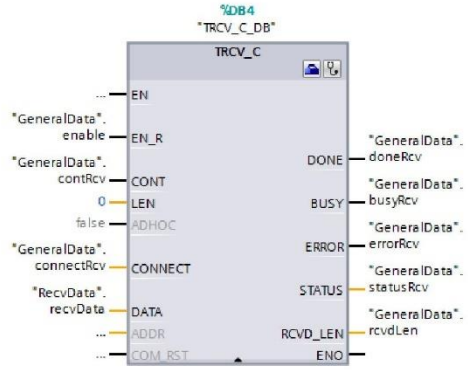


2.4 FC200 "FC_RECV"

The FC200 "FC_RECV" function calls the TRCV_C instruction to establish a communication connection over ISO-on-TCP, TCP or UDP and receive data through the communication connection.

The following figure shows the call of the TRCV_C instruction.

Figure 2-8



2.4.1 Inputs and Outputs of the TRCV_C Instruction

Inputs

The table below gives an overview of

- the inputs of the TRCV_C instruction
- the variables assigned to the inputs

Table 2-3

Input	Data type	Variable	Description
EN_R	BOOL	"GeneralData".enable	Enable receive with EN_R = 1
CONT	BOOL	"GeneralData".contRcv	Controls the communication connection: 0: Disconnects the communication connection. 1: Establishes and maintains the communication connection. If CONT=1, the TRCV_C instruction configures and establishes a communication connection. Once the connection has been configured and established, it is maintained and monitored automatically by the S7-1200 CPU.

Input	Data type	Variable	Description
LEN	UINT	0	Maximum number of bytes that can be sent with the job. **Note** If you use a receive area with optimized access at the DATA parameter, the value "0" must be used at the LEN parameter.
CONNECT	VARIANT	"GeneralData".connectRcv	Pointer to the structure of the connection description: • For ISO-on-TCP, use the TCON_IP_RFC system data type. For a description, refer to chapter 2.5 • For TCP, use the TCON_IP_V4 system data type. For a description, refer to chapter 2.6
DATA	VARIANT	"RecvData".recvData	Pointer to the receive area that contains the address and length of the data to be received.

Outputs

The table below gives an overview of

- the outputs of the TRCV_C instruction
- the variables assigned to the outputs

Table 2-4

Output	Data type	Variable	Description
DONE	BOOL	"GeneralData".doneRcv	Status parameter with the values below: 0: Job not yet started or still being executed. 1: Job executed error-free.
BUSY	BOOL	"GeneralData".busyRcv	Status parameter with the values below: 0: Job not yet started or already terminated. 1: Job has not yet terminated. A new job cannot be started.
ERROR	BOOL	"GeneralData".errorRcv	Status parameter with the values below: 0: No error 1: Error occurred.
STATUS	WORD	"GeneralData".statusRcv	Status of the instruction
RCVD_LEN	UINT	"GeneralData".rcvdLen	Volume of data actually received in bytes.

2.4.2 Enable Receiving of Data

The "GeneralData".enable variable is set permanently to the value "1" when the S7-1200 CPU is restarted (warm restart). The EN_R input of the TRCV_C instruction is set permanently to the value "1" by means of the "GeneralData".enable variable in order to enable receiving of data.

2.4.3 Establish and Maintain the Communication Connection

The "GeneralData".contRcv variable is set permanently to the value "1" when the S7-1200 CPU is restarted (warm restart). The CONT input of the TRCV_C instruction is set permanently to the value "1" by means of the "GeneralData".contRcv variable in order to establish and maintain the communication connection.

If the S7-1200 CPU goes into STOP mode, the existing communication connection is aborted and the configured communication connection is removed. You must once again execute the TRCV_C instruction to reconfigure and re-establish the communication connection.

2.4.4 LEN

A receive area with optimized access is used at the DATA parameter in this sample program. For this reason the value “0” is used at the LEN parameter.

2.4.5 Receive Area

In this sample program, the receive area “RecvData”.recvData is defined with optimized access.

The receive area is 100 bytes long and the data received is contained in data block DB5.

2.4.6 Save STATUS of the TRCV_C Instruction

If the TRCV_C instruction is executed successfully or with errors, the status of the TRCV_C instruction is saved in the "GeneralData".statusRcvSave variable. The status informs you of the cause if the data is not received successfully.

Figure 2-9

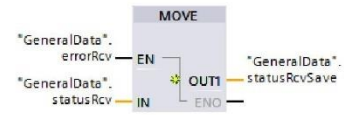
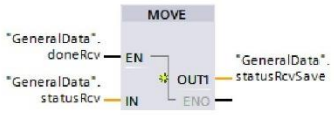


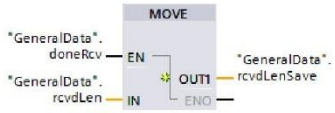
Figure 2-10



2.4.7 Save Length of the Data Received

If the TRCV_C instruction is executed successfully, the length of the data received is saved in the "GeneralData".rcvdLenSave variable.

Figure 2-11



2.5 Connection Parameters with Structure according to TCON_IP_RFC

A connection description DB with a structure according to TCON_IP_RFC is used for CPUs of S7-1200 V4.0 and higher and S7-1500 to assign parameters to ISO-on-TCP communication connections. The fixed data structure of the TCON_IP_RFC contains all parameters that are required to establish the connection.

The following table shows the structure of the connection description according to TCON_IP_RFC.

Table 2-5

Byte	Parameter	Data Type	Value	Description
0 and 1	InterfaceId	HW_ANY	64	Hardware identifier of the integrated PROFINET interface of the CPU
2 and 3	ID	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TSEND_C and TRCV_C instruction under ID.
4	ConnectionType	BYTE	16#0C	Connection type: • 12: ISO-on-TCP (16#0C hex = 12 dez)
5	ActiveEstablished	BOOL	False	Identifier for the type of connection establishment • False: passive connection establishment • True: active connection establishment

Byte	Parameter	Data Type	Value	Description
6 to 9	RemoteAddress	ARRAY [1..4] of BYTE	-	IP address of the partner endpoint, e. g. for 192.168.0.3 • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 3
10 to 43	RemoteTSelector	TSelector	-	TSelector of the remote connection partner: • TSelLength = value range 0 to 32 as UINT • TSel[1-32] = value range each 0 to 255 in bytes
44 to 77	LocalTSelector	TSelector	-	TSelector of the local connection partner: • TSelLength = value range 0 to 32 as UINT • TSel[1-32] = value range each 0 to 255 in bytes

NOTE You have to use different connection numbers for the TSEND_C and TRCV_C. Otherwise two connections with the same connection number will be establish.

The same TSelector is used for the remote and local connection partner in this sample program.

2.6 Connection Parameters with Structure according to TCON_IP_V4

A connection description DB with a structure according to TCON_IP_V4 is used for CPUs of S7-1200 V4.0 and higher and S7-1500 to assign parameters to TCP or UDP communication connections. The fixed data structure of the TCON_IP_V4 contains all parameters that are required to establish the connection.

The following table shows the structure of the connection description according to TCON_IP_V4.

Table 2-6

Byte	Parameter	Data Type	Value	Description
0 and 1	InterfaceId	HW_ANY	64	Hardware identifier of the integrated PROFINET interface of the CPU
2 and 3	ID	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TSEND_C and TRCV_C instruction under ID.

Byte	Parameter	Data Type	Value	Description
4	ConnectionType	BYTE	16#0B	Connection type: • 11: TCP (16#0B hex = 11 dez) • 19: UDP (16#13 hex = 19 dez)
5	ActiveEstablished	BOOL	False	Identifier for the type of connection establishment • False: passive connection establishment • True: active connection establishment
6 to 9	RemoteAddress	ARRAY [1..4] of BYTE	-	IP address of the partner endpoint, e. g. for 192.168.0.3 • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 3
10 and 11	RemotePort	UINT	2000	Port address of the remote connection partner (value range: 1 to 49151)
12 and 13	LocalPort	UINT	2000	• Port address of the local connection partner (value range: 1 to 49151)

NOTE You have to use different connection numbers for the TSEND_C and TRCV_C. Otherwise two connections with the same connection number will be establish.

The same port address is used for the remote and local connection partner in this sample program.

ANEXO IV: Datasheet PLC Siemens S7-1200 (Resumido)

Debido a que esto es un documento puramente académico, se ha recortado parte del datasheet del PLC.

SIEMENS

SIMATIC

S7
Controlador programable S7-1200

Manual de sistema

11/2009
A5E02486683-02

Notas jurídicas

Filosofía en la señalización de advertencias y peligros

Este manual contiene las informaciones necesarias para la seguridad personal así como para la prevención de daños materiales. Las informaciones para su seguridad personal están resaltadas con un triángulo de advertencia; las informaciones para evitar únicamente daños materiales no llevan dicho triángulo. De acuerdo al grado de peligro las consignas se representan, de mayor a menor peligro, como sigue.

⚠ DANGER
Significa que, si no se adoptan las medidas preventivas adecuadas **se producirá** la muerte, o bien lesiones corporales graves.

⚠ WARNING
Significa que, si no se adoptan las medidas preventivas adecuadas **puede producirse** la muerte o bien lesiones corporales graves.

⚠ CAUTION
con triángulo de advertencia significa que si no se adoptan las medidas preventivas adecuadas, pueden producirse lesiones corporales.

CAUTION
sin triángulo de advertencia significa que si no se adoptan las medidas preventivas adecuadas, pueden producirse daños materiales.

NOTICE
significa que puede producirse un resultado o estado no deseado si no se respeta la consigna de seguridad correspondiente.

Si se dan varios niveles de peligro se usa siempre la consigna de seguridad más estricta en cada caso. Si en una consigna de seguridad con triángulo de advertencia se alarma de posibles daños personales, la misma consigna puede contener también una advertencia sobre posibles daños materiales.

Personal cualificado

El producto/sistema tratado en esta documentación sólo deberá ser manejado o manipulado por **personal cualificado** para la tarea encomendada y observando lo indicado en la documentación correspondiente a la misma, particularmente las consignas de seguridad y advertencias en ella incluidas. Debido a su formación y experiencia, el personal cualificado está en condiciones de reconocer riesgos resultantes del manejo o manipulación de dichos productos/sistemas y de evitar posibles peligros.

Uso previsto o de los productos de Siemens

Considere lo siguiente:

⚠ WARNING
Los productos de Siemens sólo deberán usarse para los casos de aplicación previstos en el catálogo y la documentación técnica asociada. De usarse productos y componentes de terceros, éstos deberán haber sido recomendados u homologados por Siemens. El funcionamiento correcto y seguro de los productos exige que su transporte, almacenamiento, instalación, montaje, manejo y mantenimiento hayan sido realizados de forma correcta. Es preciso respetar las condiciones ambientales permitidas. También deberán seguirse las indicaciones y advertencias que figuran en la documentación asociada.

Marcas registradas

Todos los nombres marcados con ® son marcas registradas de Siemens AG. Los restantes nombres y designaciones contenidos en el presente documento pueden ser marcas registradas cuya utilización por terceros para sus propios fines puede violar los derechos de sus titulares.

Exención de responsabilidad

Hemos comprobado la concordancia del contenido de esta publicación con el hardware y el software descritos. Sin embargo, como es imposible excluir desviaciones, no podemos hacernos responsable de la plena concordancia. El contenido de esta publicación se revisa periódicamente; si es necesario, las posibles las correcciones se incluyen en la siguiente edición.

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
ALEMANIA

Ordernumber: 6ES7298-8FA30-8DH0
10/2009

Prólogo

Objeto del manual

La gama S7-1200 abarca distintos controladores lógicos programables (PLCs) que pueden utilizarse para numerosas tareas. Gracias a su diseño compacto, bajo costo y amplio juego de instrucciones, los PLCs S7-1200 son idóneos para controlar una gran variedad de aplicaciones. Los modelos S7-1200 y el software de programación basado en Windows ofrecen la flexibilidad necesaria para solucionar las tareas de automatización.

Este manual contiene información sobre cómo montar y programar los PLCs S7-1200 y está dirigido a ingenieros, programadores, técnicos de instalación y electricistas que dispongan de conocimientos básicos sobre los controladores lógicos programables.

Nociones básicas

Para comprender este manual se requieren conocimientos básicos en el campo de la automatización y de los controladores lógicos programables.

Objeto del manual

Este manual es válido para el software STEP 7 Basic V10.5 y la gama de productos S7-1200. En los datos técnicos (Página 319) encontrará una lista completa de los productos S7-1200 descritos en el manual.

Homologaciones, marcado CE, C-Tick y otras normas

Para más información, consulte los datos técnicos (Página 319).

Service & Support

Además de nuestra documentación, ponemos nuestros conocimientos técnicos a su disposición en Internet:

http://www.siemens.com/automation/support-request
(http://www.siemens.com/automation/support-request)

Contacte con el representante de Siemens más próximo si tiene consultas de carácter técnico, así como para obtener información sobre los cursillos de formación o para pedir productos S7. Puesto que los representantes de Siemens han sido debidamente aleccionados y tienen conocimientos detallados sobre las operaciones, los procesos y la industria, así como sobre los distintos productos de Siemens empleados, pueden solucionar cualquier problema de forma rápida y eficiente.

Sinopsis del producto 1

1.1 Introducción al PLC S7-1200

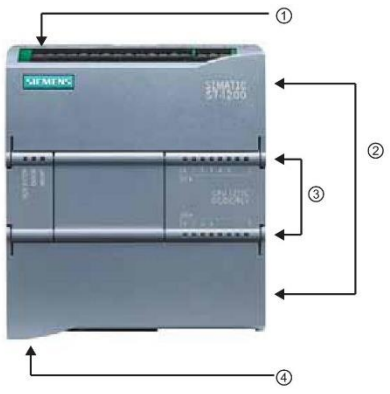
El controlador lógico programable (PLC) S7-1200 ofrece la flexibilidad y capacidad de controlar una gran variedad de dispositivos para las distintas tareas de automatización. Gracias a su diseño compacto, configuración flexible y amplio juego de instrucciones, el S7-1200 es idóneo para controlar una gran variedad de aplicaciones.

La CPU incorpora un microprocesador, una fuente de alimentación integrada, así como circuitos de entrada y salida en una carcasa compacta, conformando así un potente PLC. Una vez cargado el programa en la CPU, ésta contiene la lógica necesaria para vigilar y controlar los dispositivos de la aplicación. La CPU vigila las entradas y cambia el estado de las salidas según la lógica del programa de usuario, que puede incluir lógica booleana, instrucciones de contaje y temporización, funciones matemáticas complejas, así como comunicación con otros dispositivos inteligentes.

Numerosas funciones de seguridad protegen el acceso tanto a la CPU como al programa de control:

- Toda CPU ofrece protección por contraseña que permite configurar el acceso a sus funciones.
- Es posible utilizar la "protección de know-how" para ocultar el código de un bloque específico. Encontrará más detalles en el capítulo "Principios básicos de programación" (Página 99).

La CPU incorpora un puerto PROFINET para la comunicación en una red PROFINET. Los módulos de comunicación están disponibles para la comunicación en redes RS485 o RS232.



① Conector de corriente
② Conectores extraíbles para el cableado de usuario (detrás de las tapas)
② Ranura para Memory Card (debajo de la tapa superior)
③ LEDs de estado para las E/S integradas
④ Conector PROFINET (en el lado inferior de la CPU)

Los diferentes modelos de CPUs ofrecen una gran variedad de funciones y prestaciones que permiten crear soluciones efectivas destinadas a numerosas aplicaciones. Para más información sobre una CPU en particular, consulte los datos técnicos (Página 319).

Función	CPU 1211C	CPU 1212C	CPU 1214C
Dimensiones físicas (mm)	90 x 100 x 75		110 x 100 x 75
Memoria de usuario • Memoria de trabajo • Memoria de carga • Memoria remanente	 • 25 KB • 1 MB • 2 KB		 • 50 KB • 2 MB • 2 KB
E/S integradas locales • Digitales • Analógicas	 • 6 entradas/4 salidas • 2 entradas	 • 8 entradas/6 salidas • 2 entradas	 • 14 entradas/10 salidas • 2 entradas
Tamaño de la memoria imagen de proceso	1024 bytes para entradas (I) y 1024 bytes para salidas (Q)		
Área de marcas (M)	4096 bytes		8192 bytes
Ampliación con módulos de señales	Ninguna	2	8
Signal Board	1		
Módulos de comunicación	3 (ampliación en el lado izquierdo)		
Contadores rápidos • Fase simple • Fase en cuadratura	3 • 3 a 100 kHz • 3 a 80 kHZ	4 • 3 a 100 kHz 1 a 30 kHz • 3 a 80 kHz 1 a 20 kHz	6 • 3 a 100 kHz 3 a 30 kHz • 3 a 80 kHz 3 a 20 kHz
Salidas de impulsos	2		
Memory Card	SIMATIC Memory Card (opcional)		
Tiempo de respaldo del reloj de tiempo real	Típico: 10 días / Mínimo: 6 días a 40 °C		
PROFINET	1 puerto de comunicación Ethernet		
Velocidad de ejecución de funciones matemáticas con números reales	18 µs/instrucción		
Velocidad de ejecución booleana	0,1 µs/instrucción		

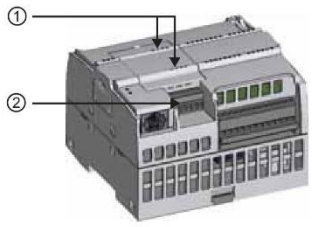
La gama S7-1200 ofrece una gran variedad de módulos de señales y Signal Boards que permiten ampliar las prestaciones de la CPU. También es posible instalar módulos de comunicación adicionales para soportar otros protocolos de comunicación. Para más información sobre un módulo en particular, consulte los datos técnicos (Página 319).

Módulo		Sólo entradas	Sólo salidas	Entradas y salidas
Módulo de señales (SM)	Digital	8 entradas DC	8 salidas DC 8 salidas de relé	8 entradas DC/8 salidas DC 8 entradas DC/8 salidas de relé
		16 entradas DC	16 salidas DC 16 salidas de relé	16 entradas DC/16 salidas DC 16 entradas DC/16 salidas de relé
	Analógico	4 entradas analógicas 8 entradas analógicas	2 salidas analógicas 4 salidas analógicas	4 entradas analógicas/2 salidas analógicas
Signal Board (SB)	Digital	-	-	2 entradas DC/2 salidas DC
	Analógico	-	1 salida analógica	-
Módulo de comunicación (CM) • RS485 • RS232				

1.2 Signal Boards

Una Signal Board (SB) permite agregar E/S a la CPU. Es posible agregar una SB con E/S digitales o analógicas. Una SB se conecta en el frente de la CPU.

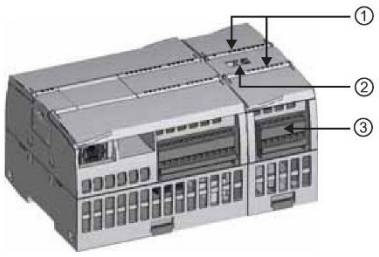
- SB con 4 E/S digitales (2 entradas DC y 2 salidas DC)
- SB con 1 entrada analógica



① LEDs de estado en la SB
② Conector extraíble para el cableado de usuario

1.3 Módulos de señales

Los módulos de señales se pueden utilizar para agregar funciones a la CPU. Los módulos de señales se conectan a la derecha de la CPU.



① LEDs de estado para las E/S del módulo de señales
② Conector de bus
③ Conector extraíble para el cableado de usuario

1.4 Módulos de comunicación

La gama S7-1200 provee módulos de comunicación (CMs) que ofrecen funciones adicionales para el sistema. Hay dos módulos de comunicación, a saber: RS232 y RS485.

- La CPU soporta como máximo 3 módulos de comunicación
- Todo CM se conecta en lado izquierdo de la CPU (o en lado izquierdo de otro CM)

① LEDs de estado del módulo de comunicación
② Conector de comunicación

1.5 STEP 7 Basic

El software STEP 7 Basic ofrece un entorno amigable que permite desarrollar, editar y observar la lógica del programa necesaria para controlar la aplicación, incluyendo herramientas para gestionar y configurar todos los dispositivos del proyecto, tales como PLCs y dispositivos HMI. STEP 7 Basic ofrece dos lenguajes de programación (KOP y FUP) que permiten desarrollar el programa de control de la aplicación de forma fácil y eficiente. Asimismo, incluye las herramientas para crear y configurar los dispositivos HMI en el proyecto.

Para poder encontrar la información necesaria, STEP 7 Basic ofrece un completo sistema de ayuda en pantalla.

Para instalar STEP 7 Basic, inserte el CD en la unidad de CDROM del equipo. El asistente de instalación arranca automáticamente y le guía por el proceso de instalación. Encontrará más información en el archivo Léame.

Nota

Para instalar el software STEP 7 Basic en un equipo con el sistema operativo Windows 2000, Windows XP o Windows Vista, es preciso iniciar la sesión con derechos de administrador.

Montaje 2

Los equipos S7-1200 son fáciles de montar. El S7-1200 puede montarse en un panel o en un raíl DIN, bien sea horizontal o verticalmente. El tamaño pequeño del S7-1200 permite ahorrar espacio.

⚠ ADVERTENCIA

Los PLCs S7-1200 SIMATIC son controladores abiertos. Por este motivo, el S7-1200 debe montarse en una carcasa, un armario eléctrico o una sala de control. Sólo el personal autorizado debe tener acceso a la carcasa, el armario eléctrico o la sala de control.

Si no se cumplen los requisitos de montaje, pueden producirse la muerte, lesiones corporales graves y/o daños materiales.

Vigile siempre los requisitos de montaje de los PLCs S7-1200.

Alejar los dispositivos S71200 de fuentes de calor, alta tensión e interferencias

Como regla general para la disposición de los dispositivos del sistema, los aparatos que generan altas tensiones e interferencias deben mantenerse siempre alejados de los equipos de baja tensión y de tipo lógico, tales como el S71200.

Al configurar la disposición del S7-1200 en el panel, se deben tener en cuenta los aparatos que generan calor y disponer los equipos electrónicos en las zonas más frías del armario eléctrico. Si se reduce la exposición a entornos de alta temperatura, aumentará la vida útil de cualquier dispositivo electrónico.

También se debe considerar la ruta del cableado de los dispositivos montados en el panel. Evite tender las líneas de señales de baja tensión y los cables de comunicación en un mismo canal junto con los cables AC y DC de alta energía y conmutación rápida.

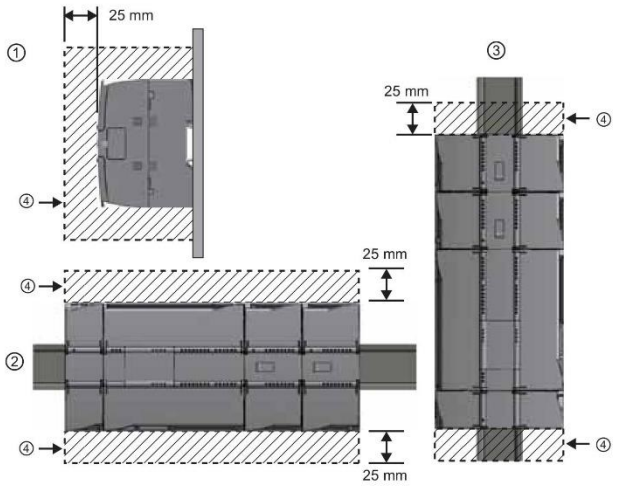
Prever espacio suficiente para la refrigeración y el cableado

La refrigeración de los dispositivos S71200 se realiza por convección natural. Para la refrigeración correcta es preciso dejar un espacio mínimo de 25 mm por encima y por debajo de los dispositivos. Asimismo, se deben prever como mínimo 25 mm de profundidad entre el frente de los módulos y el interior de la carcasa.

⚠ PRECAUCIÓN

En el montaje vertical, la temperatura ambiente máxima admisible se reduce en 10 grados centígrados. Un sistema S7-1200 montado verticalmente debe orientarse de manera que la CPU se encuentre en el extremo inferior del conjunto.

Al planificar la disposición del sistema S71200, prevea espacio suficiente para el cableado y la conexión de los cables de comunicación.



① Vista lateral
② Montaje horizontal
③ Montaje vertical
④ Espacio libre

Corriente necesaria

La CPU dispone de una fuente de alimentación interna que suministra energía eléctrica a la CPU, los módulos de señales, la Signal Board y los módulos de comunicación, así como otros equipos consumidores de 24 V DC.

En los datos técnicos (Página 319) encontrará más información sobre la corriente de 5 V DC que suministra la CPU y la corriente de 5 V DC que requieren los módulos de señales, la Signal Board y los módulos de comunicación. En "Calcular la corriente necesaria" (Página 361) encontrará más información sobre cómo determinar cuánta energía (o corriente) puede proveer la CPU para la configuración.

La CPU provee una alimentación de sensores de 24 V DC que puede suministrar 24 V DC a las entradas y bobinas de relé de los módulos de señales, así como a otros equipos consumidores. Si los requisitos de corriente de 24 V DC exceden la capacidad de la alimentación de sensores, es preciso añadir una fuente de alimentación externa de 24 V DC al sistema. En los datos técnicos (Página 319) se indica la corriente necesaria para la alimentación de sensores de 24 V DC de las distintas CPUs S7-1200.

Si se requiere una fuente de alimentación externa de 24 V DC, vigile que no se conecte en paralelo con la alimentación de sensores de la CPU. Para aumentar la protección contra interferencias, se recomienda conectar los cables neutros (M) de las distintas fuentes de alimentación.

⚠ ADVERTENCIA

Si se conecta una fuente de alimentación externa de 24 V DC en paralelo con la fuente de alimentación de sensores de 24 V DC, puede surgir un conflicto entre ambas fuentes, ya que cada una intenta establecer su propio nivel de tensión de salida.

Este conflicto puede reducir la vida útil u ocasionar la avería inmediata de una o ambas fuentes de alimentación y, en consecuencia, el funcionamiento imprevisible del sistema PLC. El funcionamiento imprevisible puede producir la muerte, lesiones corporales graves y/o daños materiales.

La fuente de alimentación DC de sensores y cualquier fuente de alimentación externa deben alimentar diferentes puntos.

Algunos puertos de entrada de alimentación de 24 V DC del sistema S7-1200 están interconectados, teniendo un circuito lógico común que conecta varios bornes M. Por ejemplo, los circuitos siguientes están interconectados si no tienen aislamiento galvánico según las hojas de datos técnicos: la fuente de alimentación de 24 V DC de la CPU, la entrada de alimentación de la bobina de relé de un SM, o bien la fuente de alimentación de una entrada analógica sin aislamiento galvánico. Todos los bornes M sin aislamiento galvánico deben conectarse al mismo potencial de referencia externo.

ADVERTENCIA

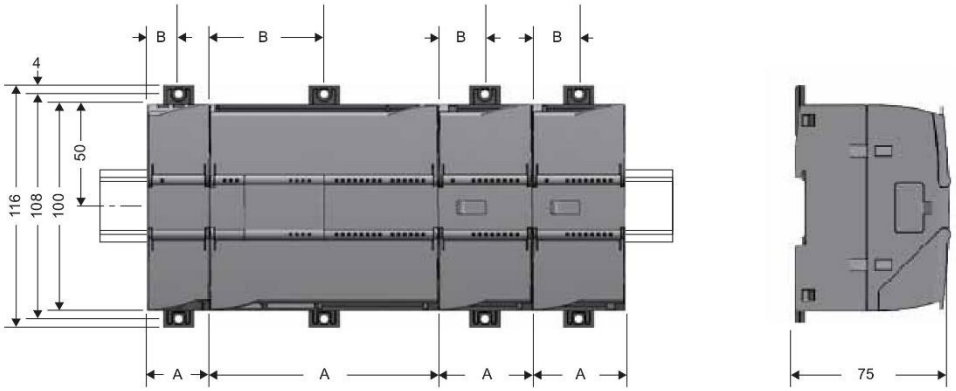
Si los bornes M sin aislamiento galvánico se conectan a diferentes potenciales de referencia, circularán corrientes indeseadas que podrían averiar o causar reacciones inesperadas en el PLC y los equipos conectados.

Si no se cumplen estas directrices, es posible que se produzcan averías o reacciones inesperadas que podrían causar la muerte, lesiones corporales graves y/o daños materiales.

Asegúrese que todos los bornes M sin aislamiento galvánico de un sistema S7-1200 están conectados al mismo potencial de referencia.

2.2 Procedimientos de montaje y desmontaje

Dimensiones de montaje (mm)



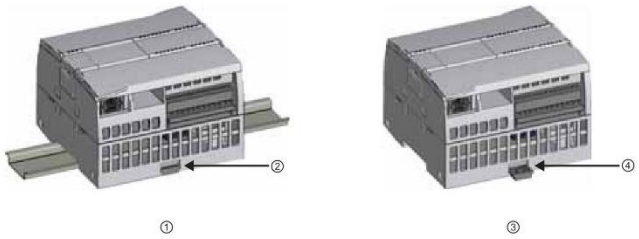
Dispositivos S7-1200		Ancho A	Ancho B
CPUs:	CPU 1211C y CPU 1212C	90 mm	45 mm
	CPU 1214C	110 mm	55 mm
Módulos de señales:	8 y 16 E/S, DC y relé (8I, 16I, 8Q, 16Q, 8I/8Q) Analógicos (4AI, 8AI, 4AI/4AQ, 2AQ, 4AQ)	45 mm	22,5 mm
	16I/16Q relé (16I/16Q)	70 mm	35 mm
Módulos de comunicación:	CM 1241 RS232 y CM 1241 RS485	30 mm	15 mm

Las CPUs, los SMs y CMs pueden montarse en un perfil DIN o en un panel. Utilice los clips del módulo previstos para el perfil DIN para fijar el dispositivo al perfil. Estos clips también pueden extenderse a otra posición para poder montar la unidad directamente en un panel. La dimensión interior del orificio para los clips de fijación en el dispositivo es 4,3 mm.

Es preciso prever una zona de disipación de 25 mm por encima y por debajo de la unidad para que el aire pueda circular libremente.

Montaje y desmontaje de dispositivos S7-1200

La CPU se puede montar fácilmente en un perfil estándar o en un panel. Los clips de fijación permiten fijar el dispositivo al perfil DIN. Estos clips también encajan en una posición extendida para proveer orificios de montaje que permiten montar el dispositivo directamente en un panel.



① Montaje en perfil DIN
② Clip de fijación al perfil enclavado
③ Montaje en panel
④ Clip de fijación en posición extendida para el montaje en panel

Antes de montar o desmontar cualquier dispositivo eléctrico, asegúrese que se ha desconectado la alimentación. Asegúrese también que está desconectada la alimentación eléctrica de todos los dispositivos conectados.

⚠ ADVERTENCIA

Si el S7-1200 o los dispositivos conectados se montan o desmontan estando conectada la alimentación, puede producirse un choque eléctrico o un funcionamiento inesperado de los dispositivos.

Si la alimentación del S7-1200 y de los dispositivos conectados no se desconecta por completo antes del montaje o desmontaje, podrían producirse la muerte, lesiones corporales graves y/o daños materiales debidos a choques eléctricos o al funcionamiento inesperado de los equipos.

Respete siempre las medidas de seguridad necesarias y asegúrese que la alimentación del S7-1200 está desconectada antes de montar o desmontar las CPUs S7-1200 o los equipos conectados.

2.2.1 Montaje y desmontaje de la CPU

Montaje

La CPU se puede montar en un panel o en un perfil DIN.

Nota

Conecte los módulos de comunicación necesarios a la CPU y monte el conjunto en forma de unidad. Los módulos de señales se montan por separado una vez montada la CPU.

Para montar la CPU en un panel, proceda del siguiente modo:

1. Posicione y taladre los orificios de montaje (M4 o estándar americano n.º 8) según las dimensiones de montaje indicadas en la tabla.
2. Extienda los clips de fijación del módulo. Asegúrese que los clips de fijación al perfil DIN en los lados superior e inferior de la CPU están en posición extendida.
3. Atornille el módulo al panel utilizando tornillos dispuestos en los clips.

Nota

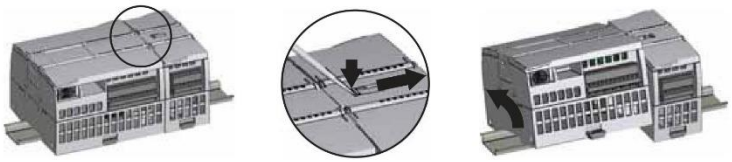
Si el sistema está sometido a vibraciones fuertes o si se monta verticalmente, el montaje en panel ofrece mayor protección al S7-1200.

Para montar la CPU en un perfil DIN, proceda del siguiente modo:

1. Monte el perfil DIN. Atornille el perfil al panel de montaje dejando un espacio de 75 mm entre tornillo y tornillo.
2. Enganche la CPU por el lado superior del perfil.
3. Extraiga el clip de fijación en el lado inferior de la CPU de manera que asome por encima del perfil.
4. Gire la CPU hacia abajo para posicionarla correctamente en el perfil.
5. Oprima los clips hasta que la CPU encaje en el perfil.

Desmontaje

Para preparar la CPU para el desmontaje, desconecte la alimentación eléctrica y los conectores de E/S y retire el cableado y demás cables de la CPU. Desmonte la CPU y los módulos de comunicación conectados en forma de conjunto. Todos los módulos de señales deben permanecer montados.



Si un módulo de señales está conectado a la CPU, retraiga el conector de bus:

1. Coloque un destornillador junto a la lengüeta en el lado superior del módulo de señales.
2. Oprima hacia abajo para desenclavar el conector de la CPU.
3. Desplace la lengüeta por completo hacia la derecha.

Desmonte la CPU:

1. Extraiga el clip de fijación para desenclavar la CPU del perfil DIN.
2. Gire la CPU hacia arriba, extráigala del perfil y retírela del sistema.

2.2.2 Montaje y desmontaje de un módulo de señales

Montaje

El SM se monta una vez montada la CPU.

Retire la tapa del conector en el lado derecho de la CPU.

- Inserte un destornillador en la ranura arriba de la tapa.
- Haga palanca suavemente en el lado superior de la tapa y retírela. Guarde la tapa para poder reutilizarla.

Coloque el SM junto a la CPU.

1. Enganche el SM por el lado superior del perfil DIN.
2. Extraiga el clip de fijación inferior para colocar el SM sobre el perfil.
3. Gire el SM hacia abajo hasta su posición junto a la CPU y oprima el clip de fijación inferior para enclavar el SM en el perfil.

Extienda el conector de bus.

1. Coloque un destornillador junto a la lengüeta en el lado superior del SM.
2. Desplace la lengüeta por completo hacia la izquierda para extender el conector de bus hacia la CPU.

Al extender el conector de bus se crean las conexiones mecánicas y eléctricas para el SM.

Siga el mismo procedimiento para montar un módulo de señales en otro módulo de señales.

Desmontaje

Cualquier SM se puede desmontar sin necesidad de desmontar la CPU u otros SMs. Para preparar el SM para el desmontaje, desconecte la alimentación eléctrica de la CPU y los conectores de E/S y retire el cableado del SM.

Retraiga el conector de bus.

1. Coloque un destornillador junto a la lengüeta en el lado superior del SM.
2. Oprima hacia abajo para desenclavar el conector de la CPU.
3. Desplace la lengüeta por completo hacia la derecha.

Si hay otro SM en el lado derecho, repita este procedimiento para ese SM.

Desmonte el SM:

1. Extraiga el clip de fijación inferior para desenclavar el SM del perfil DIN.
2. Gire el SM hacia arriba y extráigalo del perfil. Retire el SM del sistema.
3. En caso necesario, cubra el conector de bus de la CPU para impedir que se ensucie.

Siga el mismo procedimiento para desmontar un módulo de señales de otro módulo de señales.

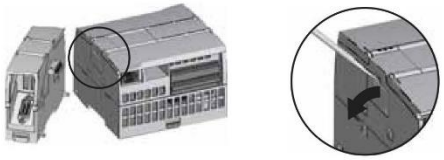
2.2.3 Montaje y desmontaje de un módulo de comunicación

Montaje

Acople el CM a la CPU antes de montar el conjunto en forma de unidad en el perfil DIN o panel.

Retire la tapa de bus en el lado izquierdo de la CPU:

1. Inserte un destornillador en la ranura arriba de la tapa de bus.
2. Haga palanca suavemente en el lado superior de la tapa.



Retire la tapa de bus. Guarde la tapa para poder reutilizarla.

Conecte las unidades:

1. Alinee el conector de bus y las clavijas del CM con los orificios de la CPU.
2. Empuje firmemente una unidad contra la otra hasta que encajen las clavijas.

Montar las unidades en un perfil DIN o panel.

1. Para el montaje en un raíl DIN, asegúrese de que el clip de fijación superior está en la posición enclavada (interior) y que el clip de fijación inferior está extendido, tanto en la CPU como en los CMs acoplados.
2. Monte la CPU y los CMs acoplados de la forma descrita en Montaje y desmontaje de la CPU (Página 29).
3. Una vez montados los dispositivos en el perfil DIN, enclave los clips de sujeción para sujetar los dispositivos al raíl.

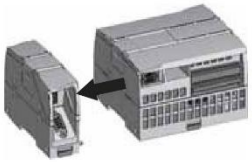
Para el montaje en un panel, asegúrese de que los clips de fijación al raíl DIN están en posición extendida.

Desmontaje

Desmonte la CPU y el CM en forma de unidad del raíl DIN o panel.

Prepare el CM para el desmontaje.

1. Desconecte la alimentación eléctrica de la CPU.
2. Desconecte los conectores de E/S y retire el cableado y demás cables de la CPU y los CMs.
3. Para el montaje en un raíl DIN, extienda los clips de sujeción inferiores de la CPU y los CMs.
4. Desmonte la CPU y los CMs del raíl DIN o panel.



Desmonte el CM.

1. Sujete la CPU y los CMs con las manos.
2. Sepárelos.

No utilice herramientas para separar los módulos, puesto que podrían deteriorarse.

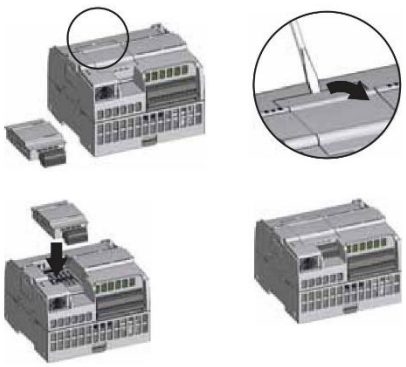
2.2.4 Montaje y desmontaje de una Signal Board

Montaje

Prepare la CPU para el montaje de la SB desconectando la alimentación de la CPU y retirando las tapas superior e inferior de los bloques de terminales de la CPU.

Para montar el SB, proceda del siguiente modo:

1. Inserte un destornillador en la ranura arriba de la CPU en el lado posterior de la tapa.
2. Haga palanca suavemente para levantar la tapa y retírela de la CPU.
3. Coloque la SB rectamente en su posición de montaje en el lado superior de la CPU.
4. Oprima firmemente la SB hasta que encaje en su posición.
5. Coloque nuevamente las tapas de los bloques de terminales.

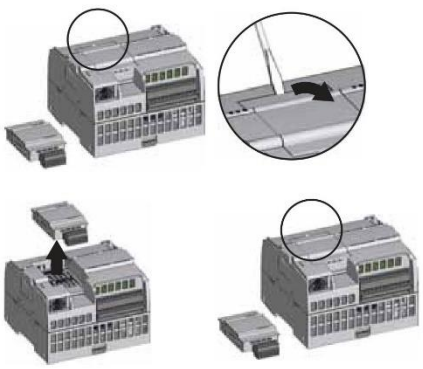


Desmontaje

Prepare la CPU para el desmontaje de la SB desconectando la alimentación de la CPU y retirando las tapas superior e inferior de los bloques de terminales de la CPU.

Para desmontar la SB, proceda del siguiente modo:

1. Inserte un destornillador en la ranura en el lado superior de la SB.
2. Haga palanca suavemente para desacoplar la SB de la CPU.
3. Retire la SB rectamente desde arriba de su posición de montaje en el lado superior de la CPU.
4. Coloque nuevamente la tapa de la SB.
5. Coloque nuevamente las tapas de los bloques de terminales.



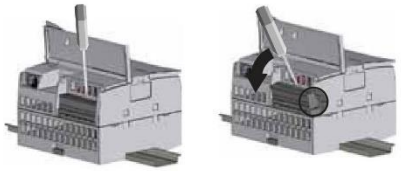
2.2.5 Extraer y reinsertar el conector del bloque de terminales del S7-1200

La CPU, la SB y los módulos SM incorporan conectores extraíbles que facilitan la conexión del cableado. Para preparar el sistema para la extracción del conector del bloque de terminales:

- Desconecte la alimentación eléctrica de la CPU.
- Abra la tapa por encima del conector.

Para desmontar el conector, proceda del siguiente modo:

1. Busque la ranura para insertar la punta del destornillador en el lado superior del conector.
2. Inserte un destornillador en la ranura.
3. Haga palanca suavemente en el lado superior del conector para extraerlo de la CPU. El conector se desenclava audiblemente.
4. Sujete el conector con las manos y extráigalo de la CPU.



Para montar el conector, proceda del siguiente modo:

1. Prepare los componentes para el montaje del bloque de terminales desconectando la alimentación de la CPU y abriendo la tapa del bloque de terminales.
2. Alinee el conector a los pines del dispositivo.
3. Alinee el lado de cableado del conector en el zócalo.
4. Con un movimiento giratorio, empuje firmemente el conector hacia abajo hasta que encaje.

Compruebe si el conector está bien alineado y encajado correctamente.

2.3 Directrices de cableado

La puesta a tierra y el cableado correctos de todos los equipos eléctricos es importante para garantizar el funcionamiento óptimo del sistema y aumentar la protección contra interferencias de la aplicación y del S7-1200. Encontrará los diagramas de cableado del S7-1200 en los datos técnicos (Página 319).

Requisitos

Antes de poner a tierra o cablear cualquier dispositivo eléctrico, asegúrese que la alimentación está desconectada. Asegúrese también que está desconectada la alimentación eléctrica de todos los equipos conectados.

Vigile que se respeten todos los reglamentos eléctricos vinculantes al cablear el S7-1200 y los equipos conectados. El equipo se debe montar y operar conforme a todas las normas nacionales y locales vigentes. Contacte con las autoridades locales para determinar qué reglamentos y normas rigen en su caso específico.

⚠ ADVERTENCIA

Si el S7-1200 o los equipos conectados se montan o cablean estando conectada la alimentación, puede producirse un choque eléctrico o un funcionamiento inesperado de los equipos. Si la alimentación del S7-1200 y de los equipos conectados no se desconecta por completo antes del montaje o desmontaje, pueden producirse la muerte, lesiones corporales graves y/o daños debidos a choques eléctricos o al funcionamiento inesperado de los equipos.

Respete siempre las medidas de seguridad necesarias y asegúrese que la alimentación eléctrica del S7-1200 está desconectada antes de montar o desmontar el S7-1200 o los equipos conectados.