Universidad de Oviedo

Memoria del Trabajo Fin de Máster realizado por

Carlos Ortea Fariñas

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

Gestión de usuarios y coches del proyecto Xplore basado en RFID

DICIEMBRE 2015

INDICE:

Tabla de contenido

Índice de ilustraciones

1. Introducción

1.1 Datos Generales

Título	Implementación de Tecnología RFID en el Proyecto Xplore-Uniovi
Autor	Carlos Ortea Fariñas
Tutor	Felipe Mateos Martín
Área	Ingeniería de Sistemas y Automática
Fecha de Presentación	22/12/2015

Tabla 1.1 Datos Generales del Proyecto

1.2 Antecedentes

El presente proyecto ha sido desarrollado por el alumno *Carlos Ortea Fariñas*, para la finalización del Máster en Ingeniería de Automatización e Informática Industrial, bajo la supervisión de Felipe Mateos Martín, en el Área de Ingeniería de Sistemas y Automática (ISA), perteneciente al Departamento de Ingeniería Eléctrica de la Universidad de Oviedo.

El Proyecto nace de la propuesta del profesor Felipe Mateos Martín para participar en el concurso *Xplore-2015*, impulsado por la empresa Phoenix Contact para desarrollar ideas relacionadas con el campo de la automatización y disciplinas relacionadas. Se trata de un certamen internacional en el que han tomado parte hasta 130 universidades de todo el mundo.

El proyecto Xplore-Uniovi consiste en el diseño, construcción y puesta en marcha de una plataforma educativa y de investigación, de carácter multidisciplinar y donde el elemento central lo constituye una pista del popular juego de rally-slot (Scalextric) al cual se le dota de alta tecnología en diferentes ámbitos: automatización, comunicaciones, visión por computador, sistemas drone, etc.

En lo referente a la parte de automatización (control y supervisión), se ha dotado al sistema, a partir del desarrollo de trabajos previos, de una serie funciones para la gestión de pruebas y carreras, manejando la velocidad de los coches de muy diferentes modos. Como funcionalidades avanzadas, entre otras, se han propuesto algunas prestaciones para la identificación de usuarios del sistema y de los coches que se utilizan y/o circulan por la pista mediante el uso de diferentes tecnologías. Este último será el foco principal en el que se desarrolla este trabajo fin de máster.

1.3 Alcance y objetivos

El objetivo del proyecto estará enfocado en el diseño e implantación de un sistema RFID para identificación, registro y explotación de una base de datos integrada en el proyecto Xplore-Uniovi. Para ello se ha pretendido cubrir los siguientes objetivos que identifican el alcance de este trabajo:

- Estudio de necesidades en cuanto a identificación, registro y explotación de datos relativos a los usuarios y los coches del sistema Xplore-Uniovi.
- Diseño de la arquitectura del sistema que permita la integración de las prestaciones definidas, pasando por las tecnologías, los componentes hardware y software necesarios: sistema de identificación, base de datos, herramientas de programación, etc…
- Diseño y desarrollo de las ampliaciones del programa de control para el autómata programable que soporta el sistema de control.
- Ampliación de la aplicación Scada de supervisión y explotación (interface HMI).
- Integración general de todo el conjunto, depuración y puesta en marcha.

1.4 Contenido

Este trabajo se organiza en dos documentos en formato electrónico **Memoria** y **Presupuesto**, y dos carpetas comprimidas) con los ficheros correspondientes al programa de control en PC Worx y la aplicación Scada desarrollada en Visu+:

- RFID PCWorx final (.zip)
- RFID Visu+ final (.zip)

Este documento **Memoria** incluye los siguientes capítulos:

1. **Introducción**. En este capítulo se comentan los antecedentes, alcance y objetivos principales del trabajo, así como el contenido de la información en formato electrónico suministrada.
2. **Descripción general del proyecto Xplore-Uniovi**. Este capítulo, aunque extenso, se considera importante para conocer de forma resumida las características del proyecto completo Xplore-Uniovi que es la base de este trabajo. Se ha de prestar especial atención a la arquitectura del sistema de control y supervisión.
3. **Sistemas de identificación**. Se hace una valoración de los sistemas de identificación de usuarios y coches existentes adaptables al sistema, los criterios de selección utilizados y la descripción de la solución adoptada.
4. **Base de datos**. Se define el sistema de base de datos empleado y los aspectos detallados del diseño, desarrollo e implementación de esta parte fundamental de la aplicación.
5. **Programa de control**. Este capítulo incluye los módulos de programación desarrollados mediante la herramienta PCWorx.

6. **Aplicación SCADA.** En este apartado se detallan especialmente los aspectos de ampliación de la interface de usuario desarrollada con la herramienta Visu**+.**
7. **Planificación**. Muestra las tareas y el tiempo de invertido para el desarrollo del trabajo fin de máster.
8. **Discusión y conclusiones**. Se hace una valoración de los objetivos conseguidos, dificultades encontradas y algunos aspectos de mejora.
9. **Bibliografía**. Se indican los textos de referencias técnicas principales y enlaces de internet de más importancia.

2 Descripción general del proyecto Xplore-Uniovi

Introducción

Este Proyecto Fin de Carrera que se presenta, forma parte de un proyecto más general, presentado al concurso internacional ***Xplore-2015*** promovido por la empresa Phoenix Contact. Este trabajo conjunto ha sido realizado desde Junio-2014 a Marzo-2015 con la participación de un grupo de profesores y estudiantes de diferentes titulaciones de la Escuela Politécnica de Ingeniería de Gijón.

IDENTIFICACIÓN DEL PROYECTO	
Competición	XPLORE-2015 (*http://www.xplore.org/2015/en/index.htm*)
Título del proyecto	An educational platform based on a racing cars game for training and research in innovative technologies
Centro	UNIVERSITY OF OVIEDO *Departamento de Ingeniería Eléctrica, Electrónica, de Computadores y de Sistemas* *Escuela Politécnica de Ingeniería de Gijón* *Campus de Gijón, s/n 33204-Gijón (Asturias-SPAIN)* *Tfno: +34 985182084; Fax: +34 985182068*
 *Campus Universitario de Gijón*	
Profesores	Felipe Mateos (*fmateos@uniovi.es*), Rafael C. González, Reyes Poo, José A. Sirgo, Jorge Alonso, J.A. Cancelas
Estudiantes	David Parte, José Manuel Piedra, Iván Granda, Javier Rodríguez, Juan Miguel Rodríguez, David Arias, César González, Sergio García, Olaya Álvarez, Pablo González, Fernando Aragón, Christian Virrueta, Elizabeth Menéndez, Alberto Suárez, Carlos García, Sonia Madero, Rocío Berros, Pablo Vidal, Carlos Ortea, Sara Mateos, Andoni Ayerdi

Ilustración 1 - Datos del proyecto Xplore-Uniovi

Es preceptivo dar las gracias de forma efusiva, especialmente a la empresa **Phoenix Contact** (promotora del concurso y colaboradora permanente con el Dpto DIEECS y la Universidad a través de los convenios establecidos), así como al **Real Grupo de Cultura Covadonga** quien han donado un par de bicicletas de spinning que, como se va comentar, forman parte uno de los mecanismos de generación de la referencia de velocidad para los coches en las pistas.

Agradecimiento especial	*Enrique Montes y Alvaro Sampedro* (Phoenix Contact Spain) *Enrique Tamargo (*Real Grupo de Cultura Covadonga – Gijón) PHOENIX CONTACT INSPIRING INNOVATIONS REAL GRUPO de CULTURA COVADONGA

Ilustración 2 - Agradecimiento XLORE-Uniovi

Con el fin de disponer de una idea global del contexto y el alcance del proyecto Xplore-Uniovi en el que se enmarca este proyecto final de carrera, ***en este capítulo*** *se tratan de resumir todos los aspectos del proyecto general sin el detalle de la parte correspondiente que será desarrollada particularmente a partir del tercer capítulo de este documento.*

2.2 Idea General

El proyecto Xplore-Uniovi, surgió de la idea de asociar diferentes elementos que permitieran construir y poner en marcha una gran y compleja infraestructura para el aprendizaje y la experimentación de diferentes tecnologías.

El proyecto se basa un popular juego de carreras de coches (tipo scalextric) que fue finalmente diseñado y montado en el laboratorio. Consta de una pista de 20 metros de longitud, con sendos circuitos para dos coches y que se extiende en una superficie aproximada de unos 30 m^2.

El diseño representa aproximadamente el espacio y los edificios del Campus Universitario de Gijón. Además de los diferentes modos de juego implementados con el control de potencia correspondiente, existe un sistema de visión por computador para determinar en todo momento la posición de los coches y eventualmente un sistema drone que podrá realizar acciones de seguimiento y rescate de vehículos.

Se han integrado algunas tecnologías innovadoras para crear finalmente una plataforma multidisciplinar para desarrollo de actividades docentes y de investigación, incluyendo: instrumentación básica, interface electrónica de señales, control avanzado de PLCs basados en el estándar IEC 61131-3, interfaces de usuario avanzadas, visión por computador, tecnología drone, coches inteligentes, robótica, tecnología RFID, comunicaciones, etc.

Para el concurso XPLORE New Automation Award 2015, se han desarrollado subsistemas para la mayoría de las actividades reseñadas. En la siguiente imagen se representa esta idea con el conjunto de áreas involucradas; en verde aparecen las que formaron parte de la implementación desarrollada para el concurso y actualmente disponibles en gran medida.

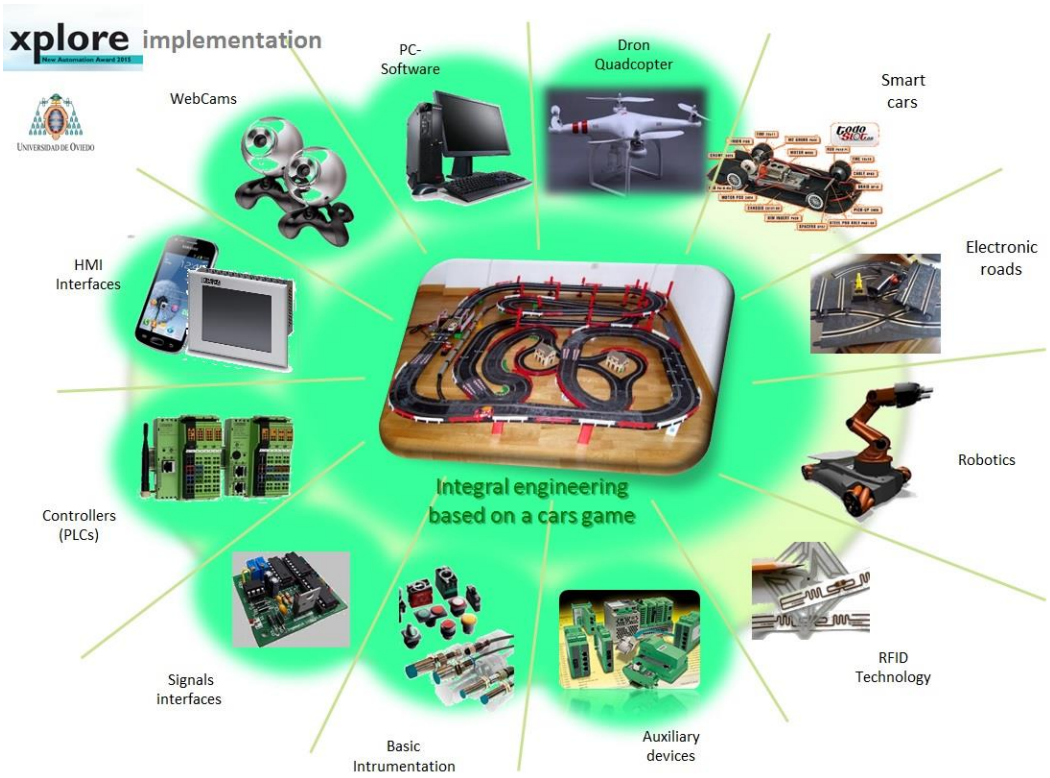


Ilustración 3 - Áreas implementadas para el concurso XPLORE-Uniovi

2.3 Infraestructura Base

La infraestructura del proyecto Xplore-Uniovi se compone principalmente de tres partes:

- *Circuito* de rally-slot para los coches
- *Mesas*, que sirven de soporte a toda la estructura del circuito.
- *Elementos decorativos y señalización*, para hacer más agradable a la vista la instalación y facilitar su funcionamiento.

La forma inicial del ***circuito de rally-slot*** ha venido determinada por la intención de darle una forma similar a la vista en planta del Campus de Gijón.

Ilustración 4 - Vista del Campus de Gijón

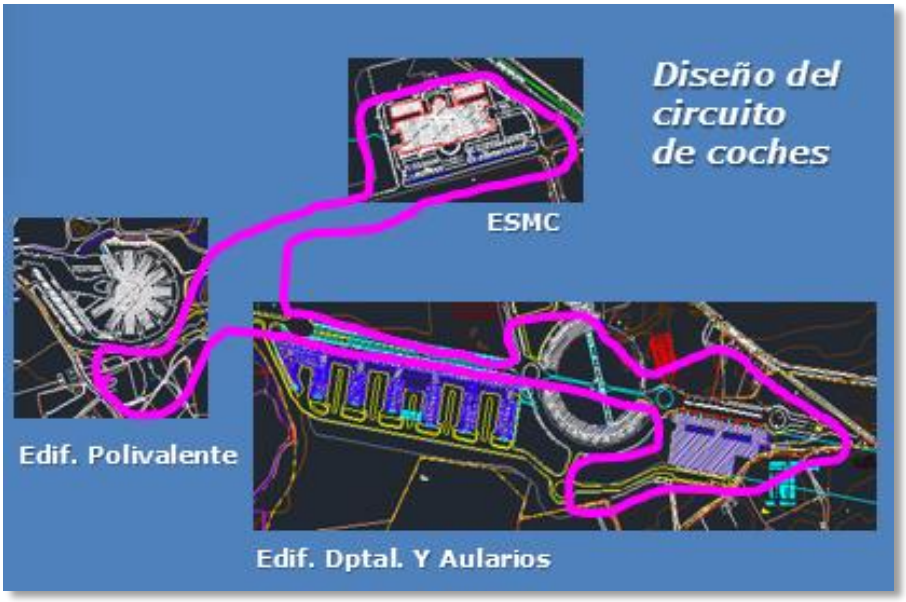


Ilustración 5 - Diseño del circuito según la estructura del Campus

Esto obligó a adaptar los tramos de pista disponibles al trazado de los edificios existentes, para lo que se utilizó un programa de optimización, como el *Tracker2000*. Tras ensamblar todos los tramos se marcó todo el recorrido sobre las mesas, para facilitar montajes posteriores, y se etiquetaron todos sus elementos. Con ello se creó un inventario de piezas que se resume la siguiente tabla.

INVENTORY:							
Type	Code SCX	Visual	Number	Type	Code SCX	Visual	Number
Rectas:	C160		16	Curvas:	C151		38
Rectas Cortas:	C159		9	Curvas cortas:	C153		4
Rectas muy cortas:	C157		1	Curvas muy cerradas:	C152		3
Recta < :	C174a		3	Curvas = :	C179		2
Recta = :	C176		1	Curva cambio sentido:	C000		1
Recta > :	C174b		3				
Cruce x :	C182		1				
Total Rectas:			**34**	**Total curvas:**			**48**
TOTAL PIECES:				**82**			

Tabla 2- Tabla resumen de inventario de tramos Scalextric

Todos los tramos están normalizados por la compañía Scalextric, por lo que en todas ellas se adjunta su código característico. La longitud total del circuito asciende a 41,39m, 20,83 en la pista A y 20,56 en el B.

Para crear la infraestructura de soporte del circuito se procuró aprovechar las ***mesas*** ya existentes en el laboratorio para ahorrar costes, y sobre todo evitar su retirada. Estas mesas tuvieron que ser adaptadas debido a que llevaban acopladas dos barras verticales que sobresalían con el fin de sujetar dos baldas horizontales. Aunque estas mesas proporcionaban una base sólida, ante la posibilidad de tener que realizar el transporte de la infraestructura completa a Alemania (fase final), se decidió adquirir unas mesas de jardín, mucho más ligeras y con patas desmontables, de 1500x845 mm que se colocaron sobre las anteriores.

Ilustración 6 - Mesas base y mesas de jardín soportando el circuito

La ventaja, aparte del transporte, fue que el escaso espesor de la tapa permitió un sencillo mecanizado; además al ser casi huecas por dentro, es posible ocultar la mayoría del cableado en su interior. Como inconveniente, debido a su poco peso se han tenido que unir con bridas de plástico para que el conjunto tuviera la rigidez suficiente de modo que permitiera mantener su posición ante posibles empujes, no afectando con ello al sistema de visión que es el más sensible en este sentido.

Tras el montaje de la instalación, se procedió a decorar la pista con diversos elementos para simular la apariencia del Campus de Gijón, además de colocar dispositivos de señalización que permiten facilitar su uso. Se añadió césped artificial a todas las partes visibles de las mesas, ocultando las marcas realizadas para facilitar el montaje y el cableado. También se situaron archivadores de plástico emulando los edificios (Polivalente, Marina Civil, etc.) dando un aspecto más realista con bajo coste.

Ilustración 7 - Vista final del circuito Xplore

Por último, se añadieron balizas de diferentes colores, una señalización acústica y cinco tiras de LEDs repartidas por el circuito para indicar con diferentes colores diversos estados de la carrera que está teniendo lugar en cada momento.

2.4 Arquitectura General del Sistema de Control

En la siguiente imagen se muestra un esquema simplificado de la arquitectura del sistema de control y supervisión del proyecto Xplore-Uniovi. Como ya se ha comentado, el componente central lo constituye una pista de rally-slot de la marca Scalextric, con dos circuitos cerrados realizados con tramos de pista de diferentes características. El circuito se sustenta sobre un

conjunto de 12 mesas. Los elementos del sistema de control: sensores, actuadores, módulos de autómata programable y auxiliares, elementos de supervisión y explotación, etc., están sujetos a estas mesas.

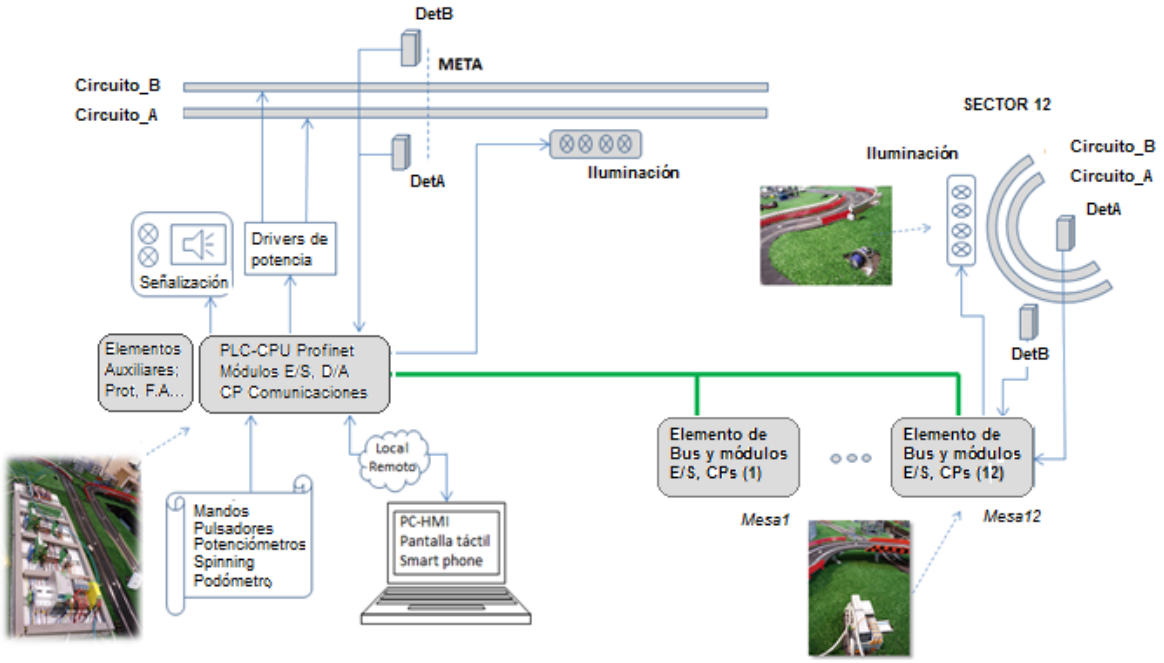


Ilustración 8 - Arquitectura simplificada del sistema de automatización y control

El controlador utilizado es el autómata programable AXC 1050, de Phoenix Contact. Se trata de un PLC de alto rendimiento con Ethernet integrado y conexiones de bus local con el sistema de E/S Axioline; hasta 63 módulos Axioline se pueden conectar a este controlador. Asimismo, AXC 1050 permite la conexión de módulos remotos, a través de su interfaz Ethernet, siguiendo el protocolo PROFINET. El PLC AXC 1050 se configura y programa según la norma IEC 61131-3 utilizando el software de automatización PCWorx.

Existe en el sistema implementado una localización especial centralizada, al lado del punto de meta para las carreras, donde está instalado el PLC junto con diversos módulos de entradas y salidas, así como las tarjetas de comunicaciones, y otros elementos auxiliares (protecciones, fuentes de alimentación, módulos de seguridad, repartidores, circuitos de potencia, etc). En la actualidad, conectados al bus local Axioline F del PLC están los módulos siguientes:

- 1 módulo de 16 entradas digitales (AXL DI 16/1-ME). Recibe las señales de los pulsos de la bicicleta estática, de las fotocélulas de detección del punto de inicio/fin de prueba, etc.
- 1 módulo de 16 salidas digitales (AXL DO 16/1-ME). Salidas para control de las lámparas de diversos colores de las balizas de señalización, bocina, 2 salidas para alimentación de las pistas (modo PWM), salidas para relés de cambio de

sentido de giro, selección de pista, etc.

- 1 módulo de 4 entradas analógicas (AXL AI4 U 1H). Entradas de mandos (2 mandos inalámbricos y 2 señales de mando provenientes de smartphones: valores proporcionales a la inclinación de los mismos).
- 1 módulo de 4 salidas analógicas (AXL AO4 1H). Salidas para alimentación de las 2 pistas (modo DCDC).
- 2 módulos de comunicación serie (AXL F RS UNI 1H ME). Para comunicación con el sistema de visión y comunicación con el dron.

Con el fin de dotar al sistema de una arquitectura de periferia descentralizada, se ha construido la estructura de bus PROFINET que se muestra en la imagen siguiente. Incluye 8 acopladores de bus Axioline para PROFINET (AXL BK PN). Cada uno de estos acopladores dispone de un módulo de 8 entradas y 8 salidas digitales (AXL F DI8/1 DO8/1), que permiten distribuir las E/S a través de las diferentes mesas. Actualmente, a estos módulos están conectadas la mayoría de las fotocélulas de detección de paso de los coches por los sectores (excepto las de meta), y los tramos de iluminación LED situados en diversos puntos de la pista.

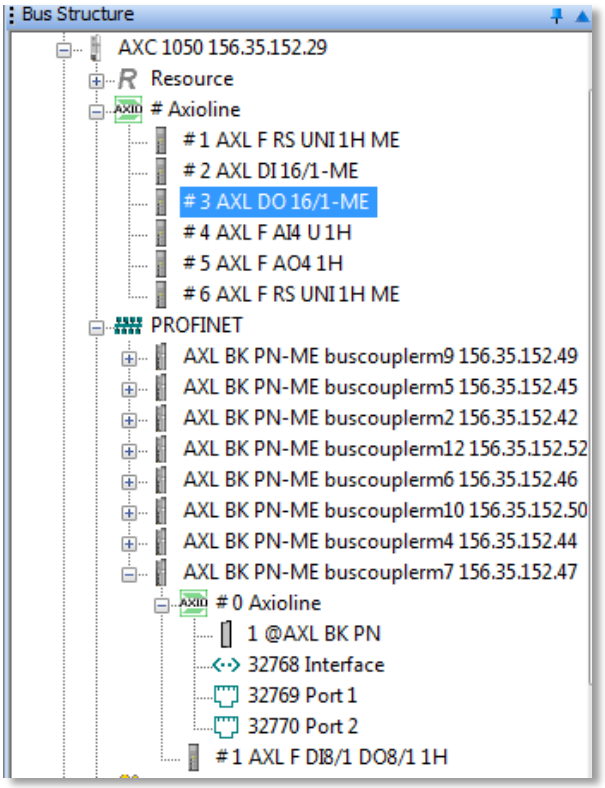


Ilustración 9 - Estructura del sistema de control en PCWorx

2.5 Diseño e Implementación de la Parte Eléctrica

En el diseño y montaje de la instalación eléctrica se ha tratado de ofrecer claridad, robustez y flexibilidad, de forma semejante a como se desarrolla a nivel profesional para una instalación industrial. El objetivo también ha sido intentar distanciarse, en este punto del concepto de "juguete" orientando al estudiante hacia una solución de integración de componentes habituales en la tecnología de automatización y control de procesos mediante el uso de autómatas programables.

La mayor concentración de componentes eléctricos se encuentra en el bastidor principal donde se ubican varios carriles normalizados, según la siguiente distribución que se aprecia en la figura.

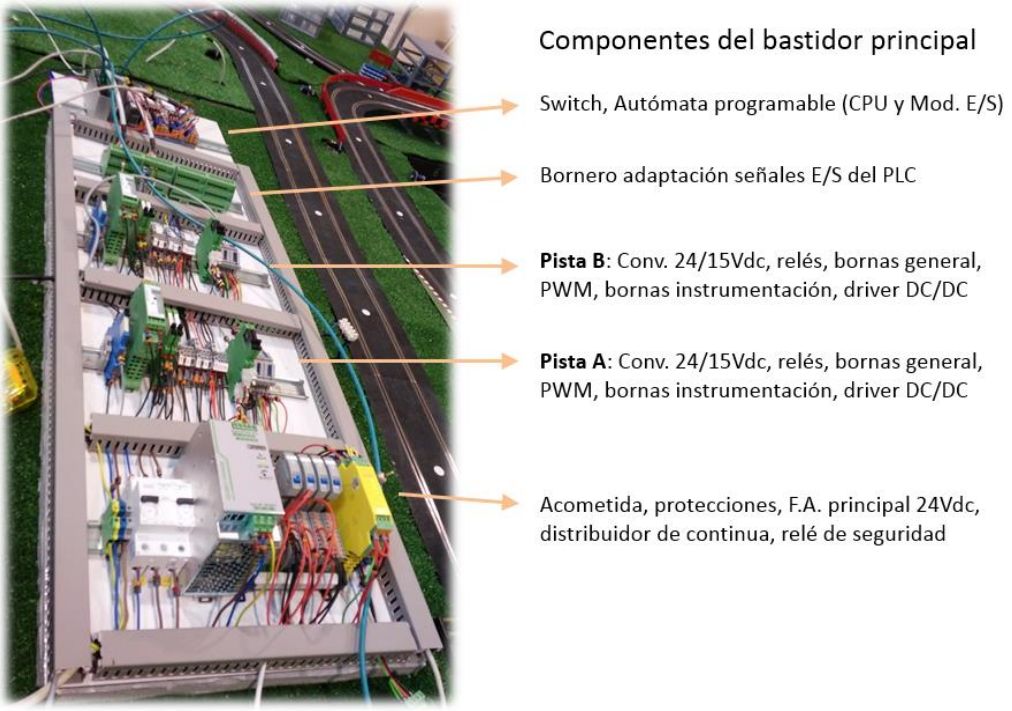


Ilustración 10 - Distribución de componentes en el bastidor principal

Como se ha comentado en cuanto a la arquitectura del sistema de control, se ha optado por un sistema de periferia descentralizada mediante un bus de campo basado en el estándar Profinet. Se han dispuesto hasta 8 módulos acopladores, y con cada uno, un módulo de 8E/8S digitales para la adquisición de señales de las fotocélulas de paso de coche por los sectores, y para la generación de colores RGB en las tiras de iluminación LEDs colocadas en cinco de las curvas del circuito.

Ilustración 11 - Acoplador Profinet

Ilustración 12 - Fotocélula y tira de LEDs

Ilustración 13 - Balizas en meta

Todos los aspectos a nivel de detalle, referentes a la instalación eléctrica, pueden encontrarse en un conjunto de 32 planos eléctricos desarrollados por el equipo Xplore-Uniovi.

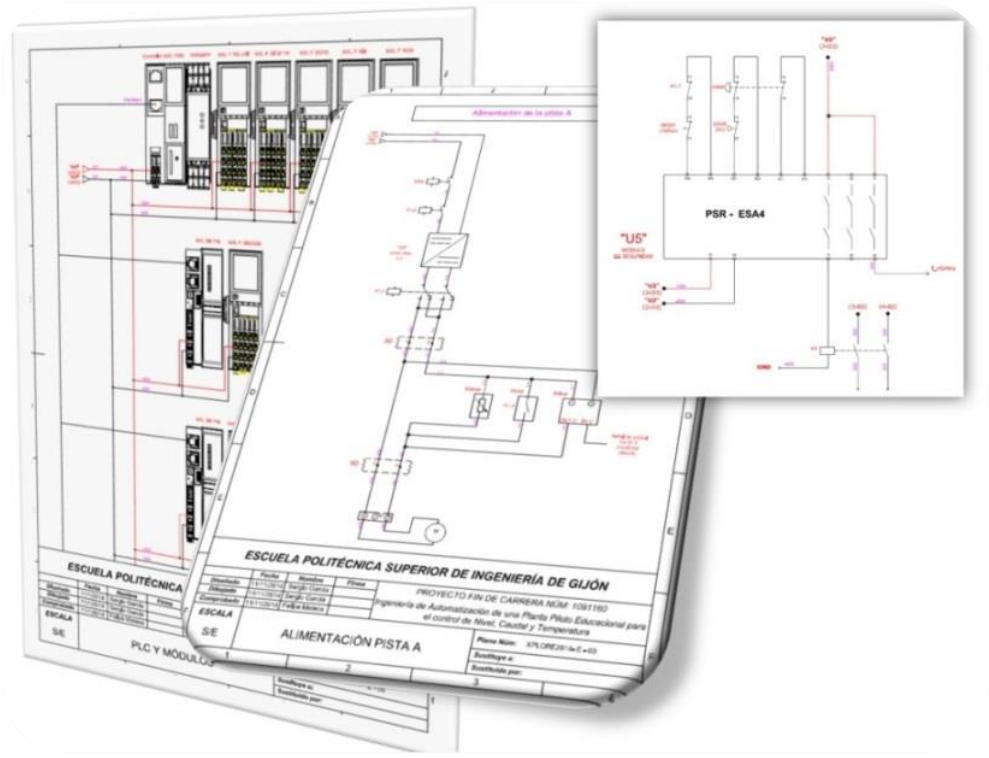


Ilustración 14 - Ejemplo de planos eléctricos del sistema Xplore-Uniovi

2.6 Especificación Funcional y Configuraciones

Dadas las características del sistema y el conjunto de elementos involucrados y sus prestaciones, se han considerado dos maneras de manejar los circuitos de potencia a los motores de los coches, una guiada por el usuario, ***Modo Manual***, y la otra en ***Modo Automático*** a través del controlador.

En ***Modo Manual***, el usuario puede, por un lado, especificar el valor de la señal de *referencia* de potencia al motor según estas posibles formas:

1. *Mando directo* (MD): Es la forma de uso convencional, a través del mando (joystick), aplicando de forma directa la salida de potencia a los circuitos.
2. *Mando indirecto* (MI): La señal del mando es capturada por una entrada, típicamente analógica, del PLC. Para ello se usan mandos inalámbricos, cuyo receptor ofrece esta posibilidad.
3. *Pulsadores* (MP): Según la frecuencia de pulsación sobre el botón se considera que la potencia aplicada al circuito debe ser proporcional a la misma. Uno o dos pulsadores por circuito. También pueden utilizarse distintos botones para incrementar y decrementar una cierta cantidad de potencia aplicada, o utilizar el tiempo con la señal activa, etc… No se han conectado por el momento.
4. *Smartphone* (MT): Se utiliza un Smartphone con una aplicación software, de modo que conectado con bluetooth a un circuito Arduino, teniendo en cuenta la inclinación del terminal móvil, se genera una tensión proporcional en una salida del Arduino que es, as su vez, una entrada al controlador.
5. *HMI* (MH): Usando un campo de entrada, una barra deslizadora, etc… en una aplicación para supervisión y explotación del sistema (interface HMI), se puede ajustar la potencia deseada que hay que aplicar al circuito.
6. *Máquina de Spinning* (MS): Según la velocidad aplicada con el pedaleo de la bicicleta tipo spinning, y registrada a través de la frecuencia de pulsos de un sensor inductivo en el volante de inercia, se ajustaría proporcionalmente la potencia de salida al coche correspondiente.
7. *Pulsos del podómetro* (MW): Se utiliza un generador de pulsos asociado a un dispositivo acoplado al cuerpo para registrar los pasos cuando se camina o se corre, y éste a su vez a una entrada digital del PLC. Según la frecuencia de pulsos, proporcional a la velocidad durante el ejercicio, el vehículo puede acelerar o frenar la marcha. Puede haber uno para cada circuito. No conectado actualmente.

En ***Modo Manual***, la salida de *potencia* a los circuitos de los coches se puede generar eléctricamente de estas 4 formas:

1. *A través del mando* (MANDO): De forma directa a como se realiza en el juego convencional. La señal de potencia no se genera por medio del PLC.
2. *Gestión del ancho de pulso* (PWM): Se usa un relé de estado sólido para que con una salida digital del PLC sea posible gestionar el ancho de pulso y con ello de forma proporcional la señal eléctrica entre 0..14Vdc a los circuitos de los coches.
3. *Mediante driver DC/DC* (DC/DC): Se usa un dispositivo que se maneja con una señal analógica de salida del PLC, por ejemplo entre 0..10Vdc, para aplicar una tensión variable entre 0..14Vdc al circuito del coche que corresponda.
4. *Directo a través del spinning* (SPINNING): Mediante un sistema generador de energía instalado en la bicicleta y un circuito de alimentación adecuado se puede hacer llegar la potencia necesaria al motor del coche. Sin implementar actualmente.

Y analizando las posibilidades disponibles para el uso en Modo Manual de los circuitos, éstas se traducen en la siguiente tabla:

Refe-rencia / Potencia	MD (1)	MI (2)	MP (3)	MT (4)	MH (5)	MS (6)	MW (7)
MANDO (1)	X						
PWM (2)		X	X	X	X	X	X
DC/DC (3)		X	X	X	X	X	X
SPINNING (4)						X	

Tabla 3 - Formas de especificar la referencia y generación de potencia a los circuitos en Modo Manual

El manejo en ***Modo Automático*** de un circuito puede estar determinado por las siguientes formas de trabajo:

1. *Auto PCte*: Se refiere a que el sistema aplicaría al circuito especificado una única potencia constante para todo el recorrido. Podrá variarse a medida que se desarrolla la prueba o carrera que se ejecute, bien a través de señales externas o más habitualmente mediante el interface HMI.
2. *Auto Sector*: Puesto que la pista está dividida en sectores, marcados por las fotocélulas que sirven de identificación de zonas y detección de paso de coches, en este modo el PLC aplicaría una potencia constante diferente en cada uno de dichos sectores; se trabaja sin realimentación pues no se conoce la velocidad en cada punto del circuito.

3. *Auto Visión*: Este es el Modo Automático por excelencia puesto que al conocer la posición del coche a través de los datos proporcionados por el sistema de visión, es posible realizar una regulación precisa de la velocidad y con ello optimizar los recorridos.

En el ***Modo Automático***, sólo se podrá generar la señal de salida de potencia a través del circuito mediante control por PWM (1) o del driver hardware de conversión DC/DC (2), dando por tanto lugar a la siguiente tabla.

Función Auto / Potencia	Auto_PCte (1)	Auto Sector (2)	Auto_Visión (3)
PWM (1)	X	X	X
DRIVER (2)	X	X	X

Tabla 4 - Formas de funcionamiento de los circuitos en Modo Automático

Para cada uno de los circuitos de coches, de forma independiente, debe ser posible establecer el modo de funcionamiento deseado, bien sea el ***Modo Manual*** o el ***Modo Automático***, según las clasificaciones especificadas en las tablas anteriores.

Para el diseño y la implementación de todo este sistema complejo de funcionamiento se ha requerido la realización de los programas de control correspondientes y de las interfaces de usuario (HMI) avanzadas que permiten establecer todas la configuraciones, así como hacer el seguimiento y supervisión de todo el sistema. De estos elementos del proyecto se hará mención en los siguientes subapartados de este capítulo general sobre el proyecto Xplore-Uniovi.

2.7 Programa Control PLC

El sistema de automatización y control de los modos de juego en el sistema se lleva a cabo por medio de un autómata programable industrial de nueva generación AXC 1050 de la marca Phoenix Contact. Estos equipos ejecutan programas de control basados en el estándar internacional **IEC 61131-3**, el cual se ha considerado muy recomendable por la amplia implantación que cada vez más está teniendo entre los diferentes fabricantes y colectivos en general, así como por proporcionar un conocimiento sostenible de la programación y no dependiente del proveedor.

Los equipos, por tanto, soportan los cinco lenguajes de programación de la norma: **LD** (Ladder Diagram, o diagrama de contactos), **FBD** (Function Block Diagram, o diagrama de bloques de funciones), **IL** (Instruction List, o lista de instrucciones), **SFC** (Sequential

Function Chart, o diagrama de funciones secuenciales), **ST** (Structured Text, o texto estructurado).

La herramienta de programación aplicada se denomina **PCWorX v6.30.767**, suministrada por Phoenix Contact. Ver en la siguiente imagen la versión que se utiliza.

Ilustración 15 - Versión de PCWorX utilizada en el proyecto

También han sido necesarias un conjunto de LIBRERÍAS PCWorX:

> ***AXL_ComSerial_v1_00*** que permite establecer comunicación serie con los sistemas PC que implementan el sistema de visión y el control del dron.
>
> ***DBFL_SQL_V1_17***, para manejar la base de datos con la información de carreras que se desarrollan en el juego.

El diseño e implementación del programa de aplicación PLC se ha estructurado a partir de una *Unidad de Organización de Programa* (POU) que se ha denominado **MAIN** desarrollada en SFC por ser de fácil comprensión. Para este grafcet, se han creado 5 ramas alternativas dependiendo del modo de juego o conjunto de acciones a realizar: *Prueba, Carrera, Contrarreloj, Aprendizaje automático y Rankings.*

El resto de los unidades de organización de programa (POU) se han realizado en Texto Estrucuturado (ST) por ser el que más flexible, compatible y que fácilmente maneja tipos de datos e instrucciones complejas.

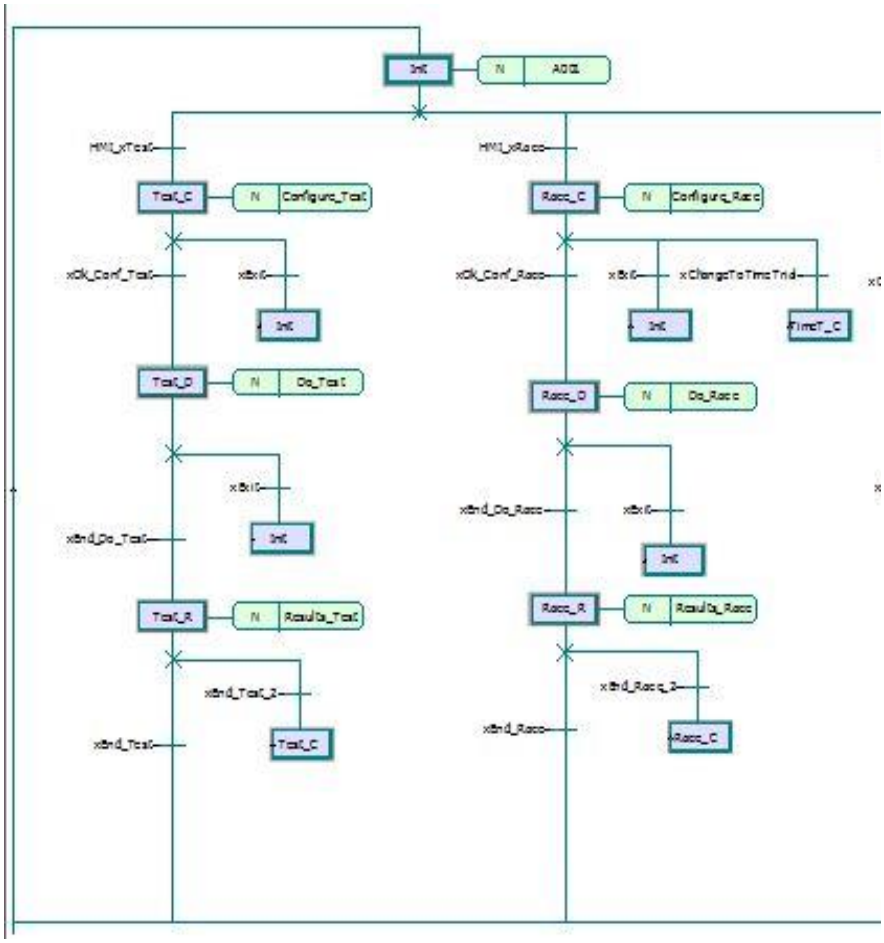


Ilustración 16 - Detalle de programación en SFC en el programa MAIN (2 ramas)

Cada una de las ramas del programa MAIN presenta varias etapas con sus correspondientes acciones, dividiéndose básicamente en tres partes. La primera es la de configuración, la segunda la de realización y la última la de resultados.

- **Configuración**: etapa donde se inicializan las variables oportunas y se da valor a las que se necesite, en función de lo indicado desde el HMI.
- **Realización**: etapa donde se juega. Aquí se encuentra el grueso del programa, donde se detectan los sensores (fotocélulas), se incrementa el número de sector y vuelta, se calculan las potencias a aplicar, se controlan las salidas físicas, se cuentan los tiempos, se controla la iluminación y se determina cuando acaba la prueba, entre otras cosas.
- **Resultados**: etapa con la prueba terminada. Se inicializan las variables que lo requieran y se llevan a cabo las tareas con la base de datos donde se registran ciertos valores.

Otros PROGRAMAS que se han desarrollado y que se ejecutan a través de una única tarea cada ciclo de scan del PLC, son los siguientes:

VISU: Programa que lleva a cabo las cuestiones necesarias referentes al sistema SCADA creado con el programa Visu+. Así mismo, obtiene con el formato adecuado algunos tiempos, tales como los de vuelta, utilizados también en la aplicación de WebVisit.

EVENT_WARNING: Programa donde se llevan a cabo las tareas relacionadas con la gestión de eventos, tales como avisos y alarmas.

READ_FOT: Programa donde se leen las variables de entrada asociadas a las fotocélulas, para activar otras variables usadas en otros puntos del programa.

READ_PHYS_INPUTS: Programa de lectura del resto de las variables de entrada del sistema, como la referencia de potencia desde distintos elementos.

WRITE_PHYS_OUTPUT: Programa de escritura de las variables de salida del sistema, como las potencias aplicadas y las luces.

P_SERIALCOMM: Programa para controlar la comunicación entre el PLC y los demás sistemas (visión y drone).

También se han realizado un conjunto de BLOQUES FUNCIONALES que han permitido una cierta reutilización de código en el proyecto, encapsulando ciertas funcionalidades que requerían o bien varios parámetros de entradas/salidas y/o almacenamiento de memoria entre llamada y llamada; estás son:

FB_SEPARATE_TIME: Bloque funcional que dada una variable de tipo TIME, devuelve los valores de los minutos, segundos y milisegundos en variables separadas, así como una variable tipo STRING que contiene el valor de la de entrada en el siguiente formato: 0:00.000.

FB_AutoMode: Bloque funcional que dependiendo del modo de referencia y del porcentaje de potencia dado, calcula la potencia real a aplicar, en función de si la salida es continua o mediante pulsos (PWM). Para el modo automático.

FB_ManualMode: Bloque funcional que dependiendo del modo de referencia y del porcentaje de potencia dado, calcula la potencia real a aplicar, en función de si la salida es continua o mediante pulsos (PWM). Para el modo manual.

FB_iPWM: Bloque funcional que dada una potencia a aplicar, calcula el tiempo que debe mantenerse el pulso activo, cuando la salida es mediante PWM.

FB_PulseTM_Power: Bloque funcional que a partir de los pulsos obtenidos de la bici de spinning calcula el valor de potencia que le corresponde (en porcentaje).

FB_InitVectorPoints: Bloque funcional con los valores de la potencia a aplicar en cada punto del circuito cuando se controla en modo automático por visión.

FB_CarToStartLine: Bloque funcional para obtener la potencia a aplicar adecuada en función del coche, circuito y sentido, cuando se hace uso de la funcionalidad de que el coche vuelva a la línea de salida.

Finalmente, como parte de la programación se han programado dos FUNCIONES para escalado de variables analógicas:

FU_iScale / ***FU_rScale:*** Funciones que, dada una escala, dan el valor de salida correspondiente a una entrada. En el primer caso para un valor entero y en el segundo para uno real.

2.8 Gestión de la Base de Datos del Proyecto

Una base de datos es un conjunto de datos organizados para un uso determinado. Existen multitud de sistemas de gestión de bases de datos (programas que permiten gestionarlos), optando para el presente proyecto por MySQL.

El PLC AXC 1050 de Phoenix Contact permite la conexión con un servidor de bases de datos MySQL o SQL Server a través de la dirección IP de la misma utilizando una conexión Ethernet, siendo necesaria la creación previa de la base de datos y su configuración.

Las acciones que es posible realizar desde el PLC sobre la base de datos son: creación de nuevas tablas, escritura en tablas ya existentes y lectura de estas. En este caso se hará uso solamente de las dos últimas funciones, ya que la creación de las tablas será necesaria una única vez, por lo que hace con otros programas más apropiados para ello.

El programa utilizado para la creación de la base de datos y de las tablas correspondientes es el MySQL Workbench, la herramienta oficial de MySQL para el diseño de bases de datos.

La utilización de la base de datos en el presente proyecto se puede reducir a dos usos. Por un lado el almacenamiento de información relativa a las pruebas y carreras llevadas a cabo, a los jugadores y a los coches disponibles, y su posterior lectura para la creación de rankings. Y por otro lado el almacenamiento de determinados valores de las potencias aplicadas en diversos puntos del circuito con el fin de aplicarlos en el modo automático.

Para ello son necesarias las siguientes tablas: **PLAYERS** (información de los jugadores), **CAR** (información de los coches), **DATA_LAPS** (información de la mejor vuelta de cada evento), **AUTOLEARN** (información de los valores de potencia obtenidos en el modo Aprendizaje Automático), y **BESTLAP** (información de los valores de las muestras de potencia que conforman las mejores vueltas).

2.9 Interfaces de Usuario y Modos de Control

2.9.1 Gestión de potencia con Joysticks y bici spinning

Para gestionar los valores de referencia de potencia que se desea aplicar a los circuitos de alimentación de las pistas en modo manual, y con ello la velocidad de los vehículos, se utilizan diferentes dispositivos y tecnologías convencionales, todas ellas implementadas de una u otra forma en el proyecto:

- Mandos convencionales (joysticks) cableados directamente al circuito sin pasar por el controlador-PLC. Una mayor presión con la mano en el mando hace aumentar la velocidad.
- Mandos inalámbricos, que por medio de radiofrecuencia permiten que el receptor genere una señal analógica en tensión de entrada al PLC, que finalmente genera la potencia proporcional correspondiente a los motores de continua de los coches.
- Un par de bicicletas de spinning, a la que se han acoplado a cierta distancia del volante de inercia sendos detectores inductivos que permite medir el paso de un conjunto de piezas metálicas pegadas al volante de inercia de modo que a cada vuelta se genera una señal digital con una frecuencia proporcional a la velocidad del pedaleo. Esta señal se lleva a una entrada digital del controlador que consigue posteriormente aplicar al circuito del coche correspondiente una tensión continua equivalente y con ello modificar su velocidad.

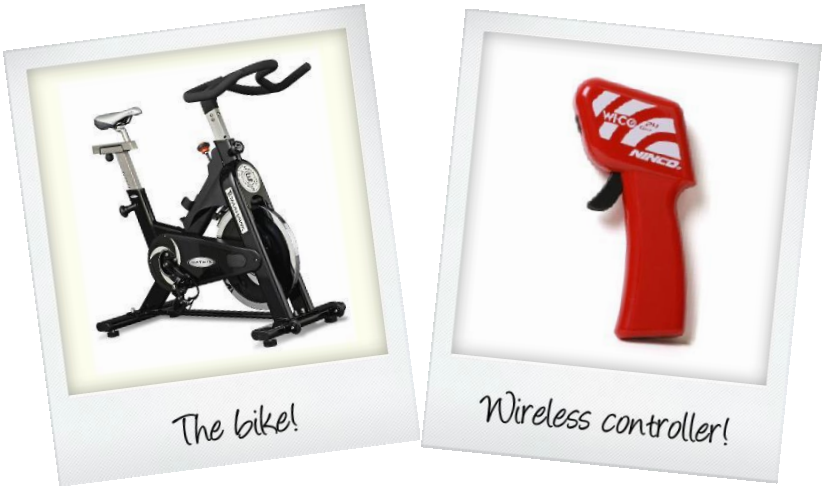


Ilustración 17 - Modelo de bici spinning y mando inalámbrico utilizados

2.9.2 Uso de Smartphone a modo de acelerador

El mundo de los teléfonos inteligentes es sin duda uno de los campos de mayor desarrollo en los últimos años, cada nueva generación de teléfonos dispone de mayor potencia de cálculo, incorpora aún más y mejor instrumentación de todo tipo, y especialmente un mayor abanico de opciones en lo que a conectividad con otros dispositivos del día a día se refiere. Así desarrolladores de todo el mundo centran actualmente sus esfuerzos en controlar múltiples dispositivos a través del teléfono móvil.

La idea de permitir indicar la referencia de potencia al coche de una de las pistas de circuito de rally-slot en el proyecto Xplore-Uniovi surgió a partir de un trabajo personal de José Manuel Piedra, intentando controlar un coche teledirigido con un Smartphone usando un Arduino como interface. En la aplicación desarrollada que se integra en Xplore, el Smartphone envía datos de su inclinación (a modo de acelerador) a la placa Arduino mediante bluetooth, el cual transformará las órdenes en una señal de tensión analógica, que puede ser leída por el PLC.

En cuanto a hardware, además del PLC, el proyecto se ha llevado a cabo empleando los siguientes componentes:

- Smartphone con sistema operativo Android (cualquier versión), conectividad Bluetooth y acelerómetro. Es necesario disponer de sistema Android debido a que se ha desarrollado una aplicación para éste sistema, aunque sería posible igualmente desarrollarla para otros como Windows Phone o iOS.
- Arduino Mega ADK R3, aunque un Arduino Uno podría cumplir las mismas funciones.
- Placa breakout JY-MCU HC-06 Bluetooth Wireless Serial Port Module.

En lo que a software se refiere, se ha empleado el IDE de Arduino (versión 1.05, Windows) para su programación y para transferir el código al microcontrolador. También se ha utilizado la versión web de App Inventor 2 del MIT, un entorno a través del cual se pueden construir aplicaciones para Android de forma sencilla, combinando bloques de funciones. La aplicación desarrollada se ha testeado en varias versiones de Android: Jelly Bean, KitKat y Lollipop.

2.10 Aplicación para Android

Antes de iniciar la aplicación es necesario activar la conectividad bluetooth en el teléfono y emparejarlo con el módulo HC-06, simplemente identificándolo por su nombre e introduciendo un pin previamente configurado.

Se ha construido una aplicación simple, cuya interfaz está únicamente compuesta por 4 botones: "Connect" es un botón mediante el cual se despliega un menú tipo lista en el cual se puede seleccionar el dispositivo con el cual se desea conectar vía bluetooth, una vez elegido se procederá a realizar la conexión. El botón "Disconnect" corta la comunicación bluetooth. Los botones "Control Track A" y "Control Track B" sirven para elegir qué pista se desea controlar a través de la aplicación.

Una vez conectado con el módulo bluetooth y seleccionada la pista que se desea controlar simplemente basta colocar el teléfono en posición vertical e inclinarlo gradualmente en cualquier dirección cuando se desee acelerar el vehículo de rally-slot. Un pequeño indicador presente en la aplicación permite observar el porcentaje de potencia que se está aplicando en cada momento a la pista.

Ilustración 18 - Interfaz principal de la aplicación desarrollada.

Como se ha dicho, la aplicación se ha construido usando App Inventor 2 del MIT. La aplicación toma la señal proporcionada por el acelerómetro del teléfono y la mapea al intervalo de números enteros [0,100]. Cada vez que el valor de esta variable cambia, se envía en formato texto a través de bluetooth, precedido por la letra "A" o "B" en función de la pista que se esté controlando.

2.11 Arduino

Para permitir que la placa Arduino y el Smartphone compartan información se recurre a un módulo Bluetooth JY-MCU HC-06, el cual recoge la señal Bluetooth y la convierte en una señal tipo serie que puede ser leída por Arduino. Este módulo dispone de 4 pines: tierra, alimentación a 5V, recepción de datos (RX) y transmisión de datos (TX). El módulo breakout debe ser también configurado a través de comunicación serial, enviándole unos comandos específicos para determinar la velocidad de transmisión de datos (baudrate), el nombre identificador para la red bluetooth y el código que permite su emparejamiento. Además el módulo dispone de un pequeño led que parpadea cuando el módulo está activo pero sin conexión y se queda fijado cuando algún dispositivo se le conecta. Se utilizan los pines 13 y 14 de la placa Arduino para establecer la comunicación serial con el módulo.

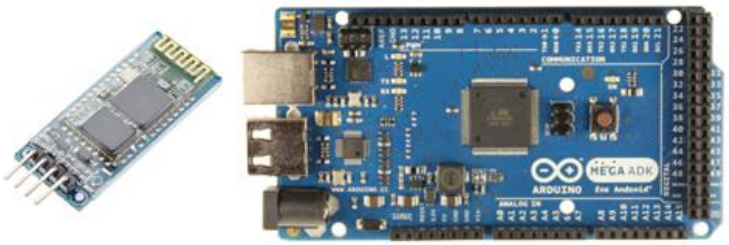


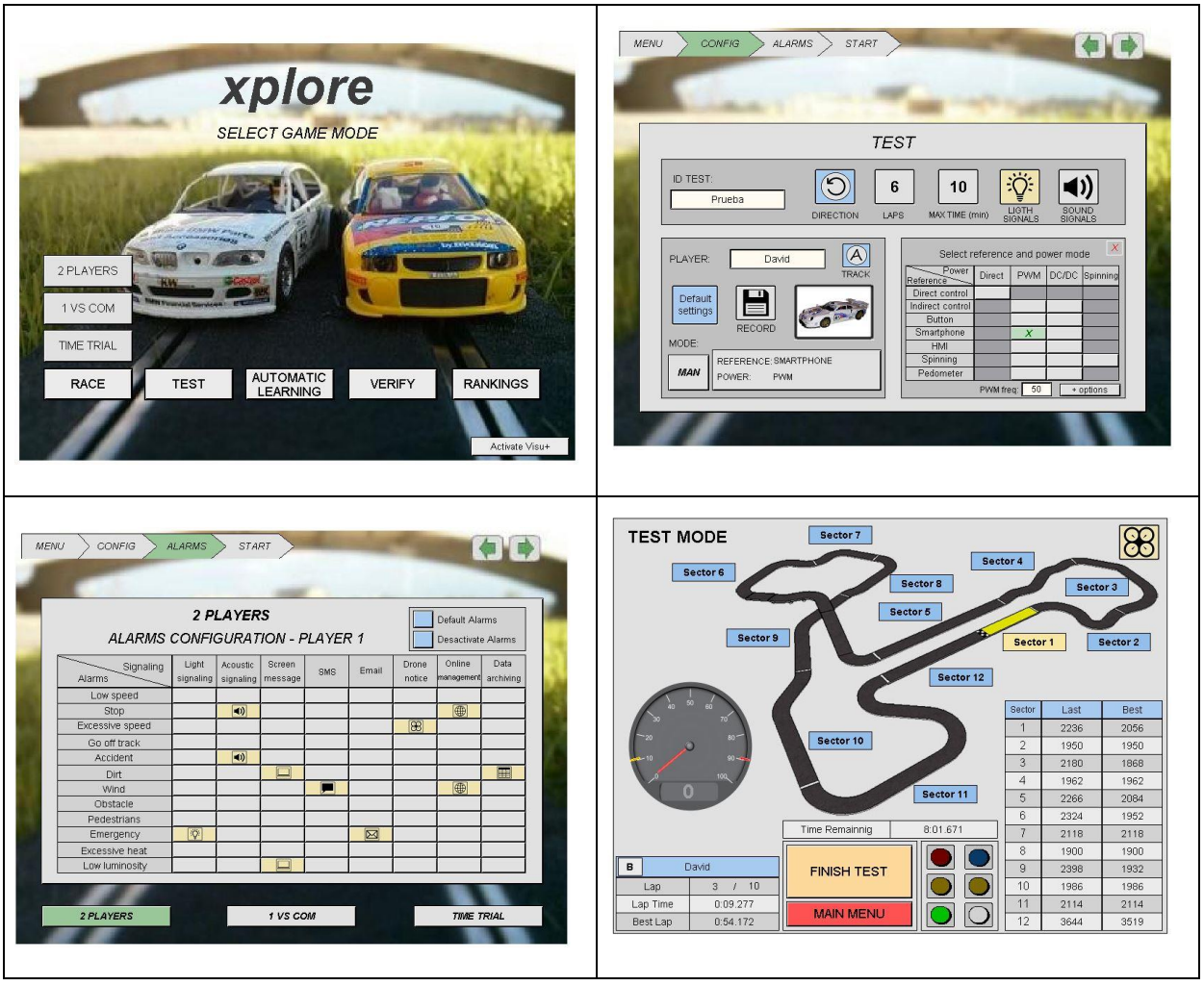
Ilustración 19 - Módulo bluetooth HC-06 (izq) y Arduino Mega ADK R3 (dcha).

El microcontrolador se ha programado para verificar constantemente el buffer de entrada, y si existen datos, leerlos y comprobar si se trata del carácter A o B, en cuyo caso se lee el número entero que sigue al carácter, se mapea al intervalo [0,255] y se escribe el dato como tensión analógica en el pin 13 o 4 respectivamente. Esto no generará realmente una señal analógica en dichos pines, sino una señal PWM de 62500Hz cuyo valor eficaz coincidirá con el que tendría la señal analógica. Para tener realmente una señal de tensión continua que pueda leer el PLC sin problemas se ha construido un filtro RC con una resistencia de 1KΩ y un condensador de 100μF.

2.12 Aplicación WebVisit como servidor web

WebVisit es un editor de páginas web, de la empresa Phoenix Contact, basado en Java, y se usa expresamente para la creación de interfaces hombre-máquina (HMI) de forma gráfica, asociando las variables del programa creado en PCWorx. La página terminada se transfiere al PLC, funcionando posteriormente el propio PLC como servidor de la aplicación. A dicha aplicación se puede acceder desde cualquier dispositivo: mediante un navegador de Internet en el PC, mediante paneles conectados al propio PLC, así como mediante una aplicación para móvil (o tablet).

La aplicación desarrollada con WebVisit se organiza en varias ventanas: *Principal, Configuración, Alarmas, Visualización, Resultados, Rankings y Verify*. En las siguientes imágenes se muestran algunas de ellas:



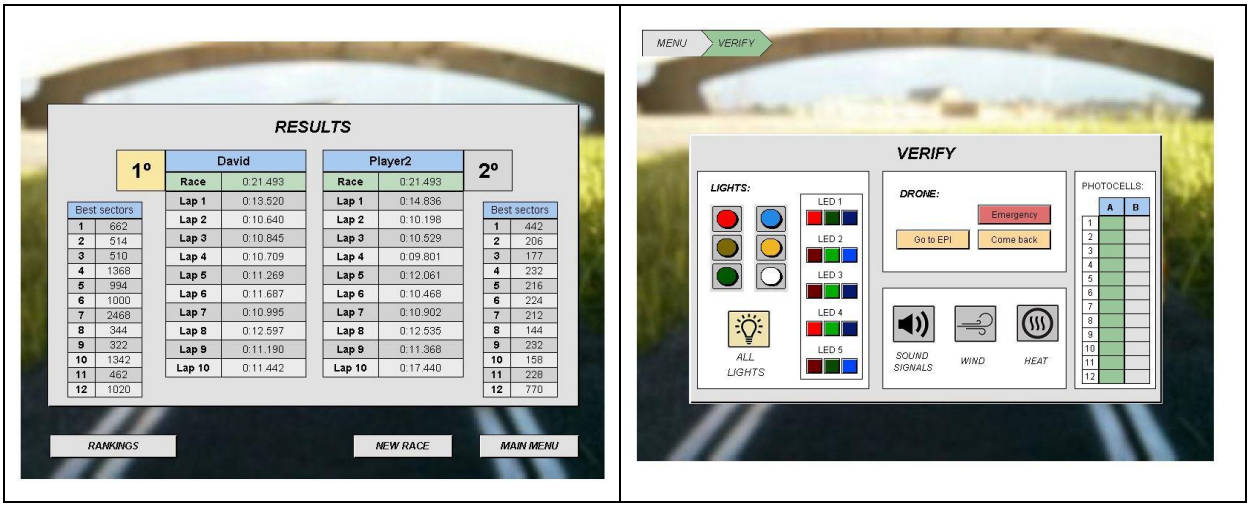
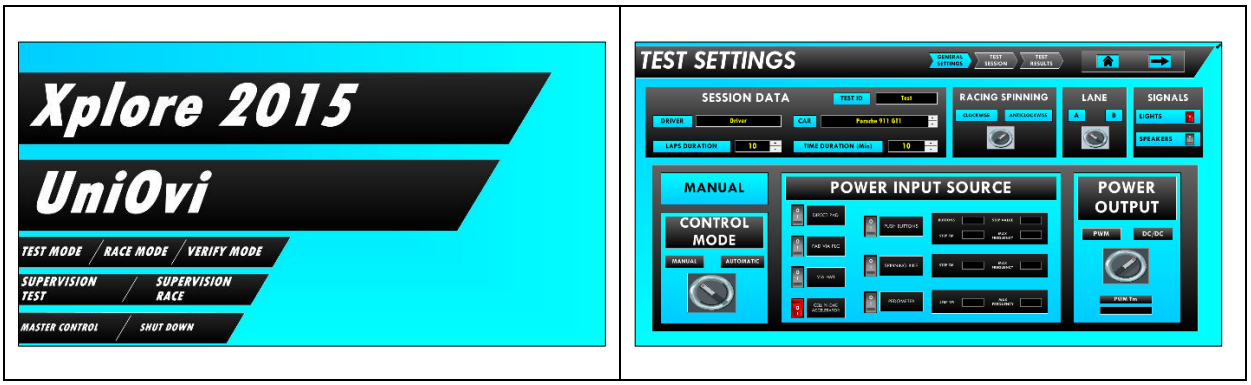


Ilustración 20 - Algunas pantallas de la aplicación WebVisit para Xplore-Uniovi

2.13 Aplicación SCADA basada en Visu+

Un conjunto de funcionalidades más avanzadas para la adquisición, supervisión y control del proceso (SCADA), también desde el punto de vista del usuario, se ha desarrollado con la herramienta software denominada **Visu+**, versión 2.31.

En base a este software se han desarrollado un conjunto de pantallas que permiten la selección del modo de funcionamiento, la configuración detallada, el arranque y parada del test o carrera, la monitorización avanzada y la presentación de resultados, así como verificación de señales y opciones de manejo del drone. A continuación se muestran algunas de estas pantallas.



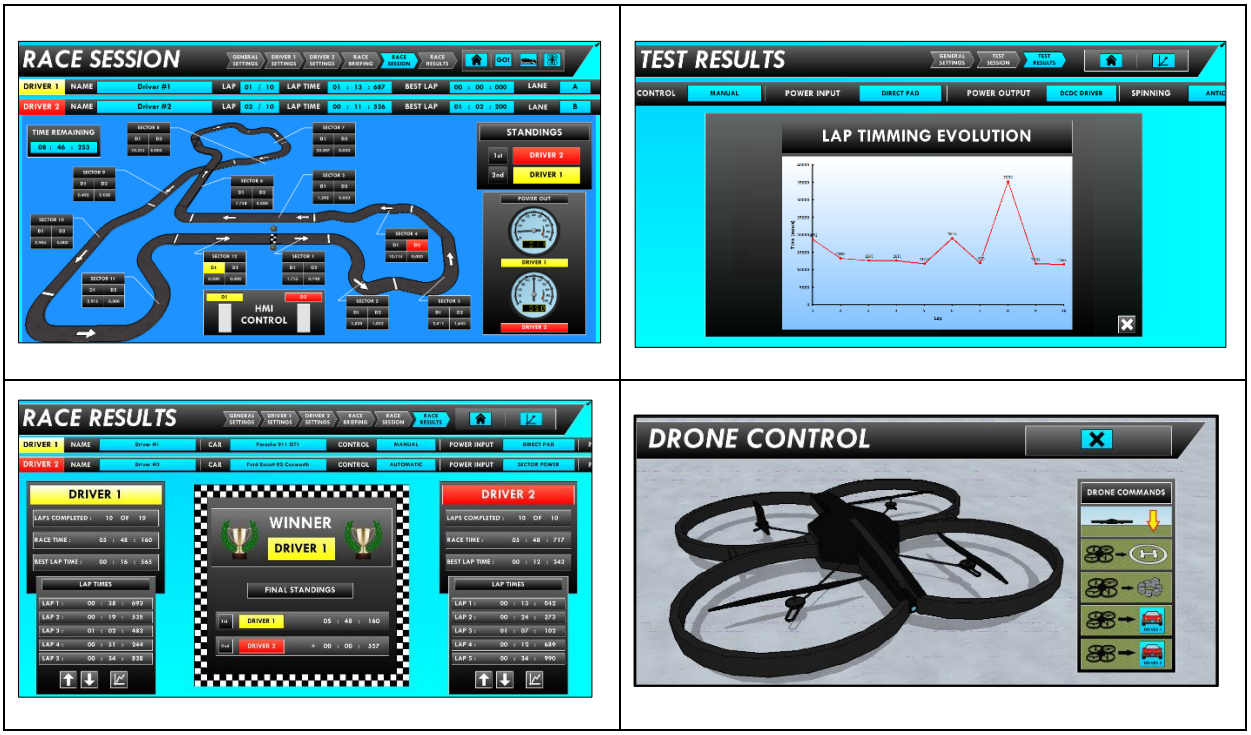


Ilustración 21 - Algunas pantallas de la aplicación Visu+ para Xplore-Uniovi

Como interfaz de comunicación entre el PLC y el ordenador donde se ejecuta la aplicación SCADA desarrollada con Visu+ para entorno Windows 7, se ha optado por el uso del **estándar OPC**, que resulta más sencillo, flexible y abierto que los drivers de comunicaciones propietarios.

La comunicación OPC se realiza en base una arquitectura de tipo cliente-servidor. En el caso de la infraestructura desarrollada el rol de servidor lo ocupa el autómata, que actúa como fuente de datos; mientras que la aplicación SCADA instalada en el PC actúa como cliente.; así, la aplicación es capaz de leer o escribir el valor o estado de cualquier variable que ofrezca el servidor. El paquete software suministrado por el fabricante incluye la versión 2.21 del servidor OPC.

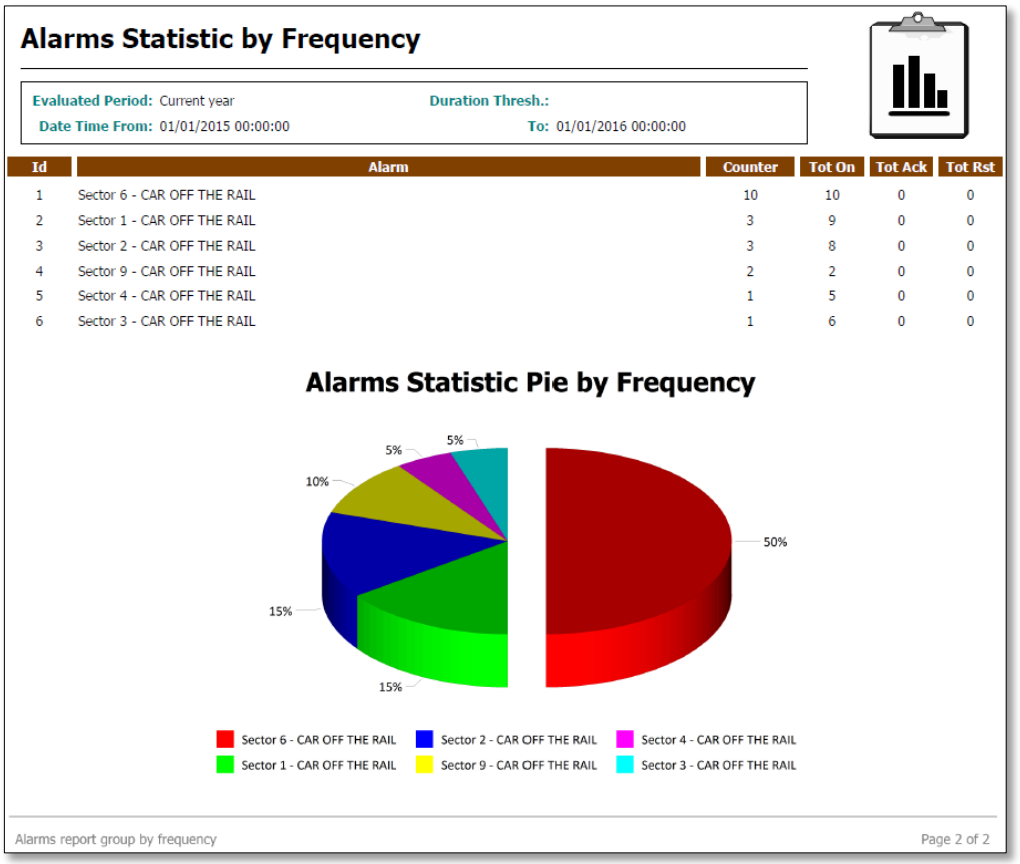
Mediante el configurador OPC instalado en el PC es posible identificar dispositivo que actúa como servidor a través de su dirección IP. Una vez establecida la interfaz de comunicación, *Visu+* es capaz de "importar" el conjunto de variables propias del proyecto residente en el autómata, para realizar las tareas de supervisión y control ya mencionadas.

El problema que subyace viene dado por el hecho de que *Visu+* no es capaz de trabajar con algunos de los diferentes tipos de datos propios de las variables utilizadas en el programa de control presente en el autómata. Para solucionar este problema se ha optado por generar nuevas unidades de programa en el propio proyecto de control con el fin de transformar las

variables cuyo tipo de dato es incompatible en otras, más simples, cuya "naturaleza" sea compatible con *Visu+*.

Las funcionalidades de las aplicaciones SCADA van más allá de la generación de pantallas de explotación de la infraestructura. Así, una de las tareas básicas de una aplicación SCADA es la generación, y posterior análisis, de **alarmas** en función del estado de determinadas variables de proceso cuya importancia para el funcionamiento de la instalación resulte crítica.

Haciendo uso de estas prestaciones avanzadas se han podido recoger los datos de salida de pista de los coches en cada sector del circuito, permitiendo un análisis estadístico y la presentación de resultados sobre los "puntos negros" del circuito. En la imagen se muestra un ejemplo de esta funcionalidad avanzada que se ha implementado para el proyecto con generación del **informe** correspondiente.



Alarms Statistic by Frequency

Evaluated Period: Current year **Duration Thresh.:**
Date Time From: 01/01/2015 00:00:00 **To:** 01/01/2016 00:00:00

Id	Alarm	Counter	Tot On	Tot Ack	Tot Rst
1	Sector 6 - CAR OFF THE RAIL	10	10	0	0
2	Sector 1 - CAR OFF THE RAIL	3	9	0	0
3	Sector 2 - CAR OFF THE RAIL	3	8	0	0
4	Sector 9 - CAR OFF THE RAIL	2	2	0	0
5	Sector 4 - CAR OFF THE RAIL	1	5	0	0
6	Sector 3 - CAR OFF THE RAIL	1	6	0	0

Alarms report group by frequency Page 2 of 2

Ilustración 22 - Informe de alarmas suministrado por la aplicación Visu+

2.14 Sistemas: Visión y Drone

2.14.1 Resumen del sistema de visión

Ante el deseo de establecer un óptimo control de la potencia aplicada a los vehículos que corren sobre la pista nació la necesidad de conocer sus parámetros cinemáticos y su localización de modo que, empleando esta información a modo de señal de realimentación, el sistema en cadena cerrada decida cuando el vehículo debe decrementar su velocidad por encontrarse en una zona sinuosa o bien adquirir una mayor velocidad en zonas menos complicadas.

Pese a que esta información (posición, velocidad...) puede ser ofrecida puntualmente por otros sensores más simples, se decidió implantar un sistema de visión artificial formado por un anillo de cámaras que rodea el espacio ocupado por el circuito, ya que es capaz de ofrecer una cobertura total del espacio de trabajo, una elevada tasa de actualización de la información y además podría permitir también tareas de identificación de los vehículos, es decir, vincular la información medida a un vehículo en concreto. Asimismo, un sistema de visión se podría utilizar también para reconocer la posición del drone-quadricoptero, ayudando así a su navegación. Todo ésto, unido al interés y la complejidad que suponen implantar un sistema de visión acentúa el carácter educacional de la plataforma y contribuye por tanto a enriquecer el proyecto.

2.14.2 Resumen del sistema drone

Basado en el resto del proyecto, se implementa una sección relacionada con robótica móvil, más concretamente un drone con el objetivo de futuras labores de inspección y/o rescate de vehículos.

La sección cuenta con un Parrot™ AR DRONE 2.0 controlado vía wifi, este dispositivo está diseñado y pensado para el ocio por lo que está pensado para su uso desde Smartphone o Tablet, no obstante, los propios fabricantes han desarrollado un driver para poder acceder a el desde el ordenador y , por extensión, poder controlarlo.

El driver viene preparado para funcionar en ROS (Robotics Operating System), una plataforma ideada para el desarrollo e implementación de aplicaciones de robótica cuyo principal lenguaje de programación es C++. El punto fuerte de ROS es la capacidad de interactuar mediante repositorios con una gran comunidad de desarrolladores.

2.15 Consideración final sobre el proyecto

En resumen, podemos decir que los objetivos del proyecto general Xplore-Uniovi han sido básicamente cubiertos: la plataforma educacional fue construida y está utilizando a pleno rendimiento. Ha sido un gran reto, porque un extenso campo de tecnologías están involucradas, y también un número notable de estudiantes de diferentes titulaciones han trabajado en ella. Los estudiantes no han podido dedicarse a tiempo completo en el proyecto, y han tenido que compartir trabajo y recursos.

Este proyecto es de claro carácter educacional e investigador, y por ello no es un producto terminado. Por eso, es siempre posible incluir más opciones en el proyecto de control del PLC, con otras perspectivas y metodologías, y mejorando su funcionalidad actual.

El pasado 6-Marzo-2015, se presentó el proyecto Xplore-Uniovi a la fase final del Concurso Internacional Xplore-2015 que se celebró en la central de la compañía Phoenix Contact de la localidad de Bad Pyrmont (Alemania); participaban otros 32 proyectos en diferentes categorías de 17 países. Este trabajo obtuvo el segundo premio.

3 Sistema de identificación

Una vez vistos los aspectos generales que cubren el proyecto completo Xplore-Uniovi, a continuación, en los siguientes capítulos se detallan las partes más concretas que se han desarrollado en este trabajo fin de máster comenzando por el sistema de identificación.

3.1 Especificación de requisitos

El uso de un sistema de identificación en el proyecto está orientado hacia la seguridad y control de accesos a la instalación, de forma que un usuario solo la podrá utilizar si dispone de algún tipo de dispositivo vinculado a una tecnología inalámbrica y portable de identificación. Como funcionalidades asociadas a este sistema se consideran las siguientes:

- Control de accesos
- Gestión de Usuarios
- Gestión de Coches
- Registro de Usuarios
- Registro de Alarmas

3.2 Tecnologías de identificación investigadas

NFC es una plataforma abierta pensada desde el inicio para teléfonos y dispositivos móviles, opera en la banda de 13.56 Mhz, esta tecnología posee una tasa de transferencia elevada por lo que está orientada hacia tasas elevadas de datos y posee un alcance limitado como máximo 20 cm.

HOTKNOT es una tecnología de NFC pero de bajo coste orientada hacia dispositivos móviles con pantalla táctil, opera en la misma banda que NFC, 13.56Mhz, requiere que ambas pantallas estén prácticamente en contacto debido a que utiliza la capacidad generada por las pantallas táctiles de los dispositivos. Esta tecnología podría ser implementada si se utilizaran pantallas táctiles de explotación en el sistema, aunque actualmente no se haya implantado fuertemente a nivel industrial.

RFID (siglas de ***R**adio **F**requency **ID**entification*, en español identificación por radiofrecuencia) Es un sistema de almacenamiento y recuperación de datos remoto que usa dispositivos denominados etiquetas, tarjetas, transpondedores o tags RFID. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio.

Fuente Wikipedia

3.2.1 Comparación entre tecnologías

	Precio	Implantación	Uso Industrial	Compatibilidad
NFC	Elevado	Media	Bajo	Media
HOTKNOT	Bajo	Baja	Nulo	Nula
RFID	Bajo	Elevada	Elevada	Elevada

Tabla 3.2.1 Comparación de Tecnologías

3.2.2 Elección de tecnología

En la tabla anterior se puede observar que, de las diferentes tecnologías investigadas, la tecnología de identificación por radiofrecuencia (RFID) es la mejor en prestaciones y coste en comparación con las otras dos.

Con la implementación de la tecnología RFID en el sistema, se pueden realizar las funciones descritas en el apartado 3.1 Por lo que esta tecnología será la elegida para implantar en el sistema.

3.3 Arquitectura de un sistema RFID

Un sistema RFID consta de los siguientes tres componentes:

- Etiqueta RFID o TAG: compuesta por una antena, un transductor radio y un material encapsulado o chip. El propósito de la antena es permitirle al chip, el cual contiene la información, transmitir la información de identificación de la etiqueta. Existen varios tipos de etiquetas. El chip posee una memoria interna con una capacidad que depende del modelo y varía de una decena a millares de bytes. Existen varios tipos de memoria:
    - Solo lectura: el código de identificación que contiene es único y es personalizado durante la fabricación de la etiqueta.
    - De lectura y escritura: la información de identificación puede ser modificada por el lector.
    - Anticolisión. Se trata de etiquetas especiales que permiten que un lector identifique varias al mismo tiempo (habitualmente las etiquetas deben entrar una a una en la zona de cobertura del lector).

- Lector de RFID: compuesto por una antena, un transceptor y un decodificador. El lector envía periódicamente señales para ver si hay alguna etiqueta en sus

inmediaciones. Cuando capta una señal de una etiqueta (la cual contiene la información de identificación de esta), extrae la información y se la pasa al subsistema de procesamiento de datos.

- Subsistema de procesamiento de datos: proporciona los medios de proceso y almacenamiento de datos.

3.4 Tipos de etiquetas RFID

Etiquetas/TAGs Pasivas: Las etiquetas pasivas no poseen alimentación eléctrica. La señal que les llega de los lectores induce una corriente eléctrica pequeña y suficiente para operar el circuito integrado de la etiqueta, de forma que puede generar y transmitir una respuesta.

Etiquetas/TAGs Activas: Las etiquetas activas poseen su propia fuente autónoma de energía, que utilizan para dar corriente a sus circuitos integrados y propagar su señal al lector. Estas son mucho más fiables que las pasivas debido a su capacidad de establecer sesiones con el lector. Gracias a su fuente de energía son capaces de transmitir señales más potentes que las de las pasivas, lo que implica una alta eficiencia en entornos difíciles para la radiofrecuencia. También son efectivas a distancias mayores pudiendo generar respuestas claras a partir de recepciones débiles. Suelen ser mayores y más caras, y su vida útil es en general mucho más corta.

Etiquetas/TAGs Semipasivas: Las etiquetas semipasivas son un hibrido entre las etiquetas activas y las pasivas, se parecen a las activas en que poseen una fuente de alimentación propia (batería), aunque en este caso se utiliza principalmente para alimentar un microchip y no para transmitir una señal. La energía contenida en la radiofrecuencia se refleja hacia el lector como en una etiqueta pasiva. La batería puede permitir al circuito integrado de la etiqueta estar constantemente alimentado y eliminar la necesidad de diseñar una antena para recoger potencia de una señal entrante. Las etiquetas RFID semipasivas responden más rápidamente, por lo que son más fuertes en el ratio de lectura que las pasivas. Este tipo de etiqueta tiene una fiabilidad comparable a la de las activas, a la vez que pueden mantener el rango operativo de una pasiva. También suelen durar más tiempo que las activas.

Fuente Wikipedia

Aunque las etiquetas activas y semipasivas ofrecen unas prestaciones superiores a las etiquetas pasivas se emplearán éstas últimas complementando el sistema de autentificación con una base de datos que se describirá más adelante.

Siendo el uso de TAGs pasivos más económico y cumpliendo con las mismas expectativas que con la implementación de etiquetas activas o semipasivas. Además se utilizarán dos tipos

de encapsulado, uno de tipo tarjeta para los usuarios y otro de tipo botón que ira colocado sobre los coches, debido a que de entre los disponibles en el mercado, se ha considerado que pueden llegar a ser los más cómodos de usar para el sistema.

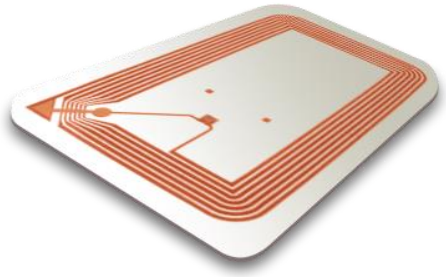


Ilustración 23 - Etiqueta RFID

Tipo Tarjeta

- Tarjeta plástica PVC laminada tamaño ISO estándar: 85,7 x 54 x 0,82 mm y 6gr.de peso aproximadamente
- Frecuencia: 125 kHz.
- Chip de lectura
- Número de serie único de 10 bytes
- Distancia de operabilidad: Hasta 10cm
- Duración de los datos: 10 años

Ilustración 24 - Tag Tarjeta

Tipo Botón

- Disco de Plástico de diámetro 22x4 mm
- Frecuencia: 125 kHz.
- Chip de solo lectura

Ilustración 25 - Tag Botón

3.5 Lector RFID

El lector RFID seleccionado para el desarrollo del proyecto ha sido un lector de Promag, modelo GP-20. Se ha elegido este modelo debido a que, ante el desconocimiento y la gran cantidad de lectores y marcas en el mercado, se ha consultado a la empresa Phoenix Contact, empresa que ha suministrado el PLC del proyecto, para intentar descartar problemas de incompatibilidad y conseguir alguna orientación hacia una correcta selección de un lector compatible y que cumpla las características necesarias para el desarrollo del proyecto.

La conexión del lector con el PLC se ha realizado mediante puerto serie RS-232 debido a que de entre las posibles conexiones que ofrece el lector esta era la más sencilla además de que en el proyecto ya se había utilizado este tipo de conexión para que el PLC interactuará con el sistema de visión y el drone.

Ilustración 26 - Lector RFID Seleccionado

3.6 DataSheet Promag GP-20

Proximity Reader GP-20 (5-13.5 Volts Version)

Power Requirements	5-13.5 Volts regulated DC@65mA tupical with 12V Supply a linear regulator is needed.
Output Interface	Wiegand,Magstripe 9.6K Baund Serial ASCII (RS-232)
Maximum Read Range	20cm@13.5VDC and 13cm@5V in ideal conditions.
Frecuency	125KHz standard
Dimensions	7.8x4.3x1.5cm
Temperature Range	-10 to 60 Deg C

Output Assignment

Red Power +VDC
Black Ground
White Magstripe clock & wiegandl, with internal 4K7 pull up
Green RS-232 data, Magstripe data & Wiegand 0, with internal 4K7 pull up
OrangeCard Present output with internal 4k7 pull up.
Yellow Program Input
Blue No Connection
Brown No Connection

Output Format

The output format can be cusomer programmed in Serial ASCII RS-232

Serial ASCII (RS-232)

Red Power +VDC
Black Ground
Green TX Data
Yellow No Connection
White No Connection
Orange No Connection

Data Structure

Serial ASCII (RS-232) Baud 9600, No Parity ,8data bits, 1 stop bit

STX (02 HEX)	DATA(10 HEX CHARACTERS)	CR	LF	ETX(03HEX)

3.7 Comunicación RS-232

La comunicación serial es un protocolo muy común para la comunicación entre dispositivos digitales, estandarizado e incluido en prácticamente todos los equipos informáticos. Destaca por su simplicidad y por ser también ampliamente utilizado en dispositivos electrónicos de instrumentación y sistemas empotrados. No debe confundirse con el Bus Serial de Comunicación (USB).

Es una comunicación digital, por tanto trabaja con bits, ceros y unos representados respectivamente por -12V y +12V. El puerto serial envía y recibe los bits que constituyen la información uno detrás de otro, bit a bit, o "en serie" por un único cable. Una comunicación en paralelo, en la que hay varios hilos por los que se transmiten varios bits a la vez, por ejemplo con 8 hilos podemos transmitir un byte en el tiempo en que la comunicación serial transmitiría un bit. La comunicación serial es por tanto una comunicación lenta, pero sencilla pues requiere pocos hilos de comunicación, es más simple de implementar y puede alcanzar distancias de comunicación mucho mayores.

Con sólo dos hilos, uno de tierra y otro de envío/recepción de datos ya podemos establecer una comunicación, sin embargo de éste modo la comunicación podrá únicamente ser unidireccional. Si añadimos un tercer hilo ya podemos hablar de una comunicación full dúplex (bidireccional y en la cual se pueden enviar y recibir datos al mismo tiempo). Uno de los hilos serviría de referencia de los niveles de voltaje, o tierra, y los otros dos hilos estarían destinados a la comunicación propiamente dicha, uno para cada dirección de la comunicación, y por ellos atravesarán señales cuadradas que representan la información binaria.

La Institución de Normalización Americana (EIA) ha escrito la norma RS-232-C que regula el protocolo de transmisión de datos, el cableado, las señales eléctricas y los conectores a utilizar por cualquier arquitectura RS-232.

La comunicación serial es asíncrona, el receptor necesita saber donde comienza y donde termina cada bit en la señal recibida para efectuar el muestreo de la misma en el centro de cada bit. Para esto es necesario que ambos dispositivos estén configurados con la misma velocidad de transmisión en baudios (bits por segundo), un valor muy típico es 9600 baudios. Además, aunque la información serie se transmite por definición bit a bit, se estructura en forma de palabras o bytes, se pueden enviar los bits en conjuntos de 5, 7 u 8 unidades.

Debido a esto surge la necesidad de la sincronización del mensaje: Es necesario conocer el inicio y fin de la palabra o byte por parte del receptor para interpretar correctamente un mensaje. En la transmisión asíncrona por cada carácter se envía al menos 1 bit de inicio y 1 o 2 bits de parada así como opcionalmente 1 bit de paridad que se calcula a partir del resto de bits del byte y se emplea a modo de comprobación.

Cuando se desea enviar un mensaje, se acumula en un buffer tipo FIFO (First-in, First-out, el primer dato introducido es el primero en salir) hasta que un driver decida que es posible su transmisión física y proceda a su escritura sobre los pines de salida. Lo mismo ocurre al recibir datos, todos los datos recibidos se acumulan en un buffer de entrada a la espera de ser procesados. Los buffers son zonas de memoria reservadas y controladas por un driver, por lo que hay que asegurar que no se desborden. Para lograrlo y conseguir que dos dispositivos se comuniquen correctamente será necesario configurar:

- Handshake activado o desactivado.
- Velocidad de transmisión de datos.
- Bit de paridad activado o desactivado.
- Bit de stop: un bit o dos bits.
- Número de bits que forman la palabra a enviar.

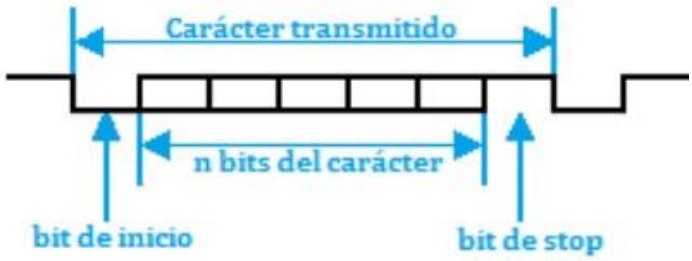


Ilustración 27 - Trama Comunicación RS-232

Estableciendo la misma configuración en ambos dispositivos, todo esto debería ser "transparente" de cara al usuario, el cual solo tendría que preocuparse de enviar o recibir bytes.

Más adelante se explicará detalladamente el programa diseñado para la comunicación por puerto serie así como la librería usada para ello.

3.8 Empleo del Sistema de Autentificación

Una vez seleccionado la tecnología a utilizar, el equipo y la comunicación se procede a la descripción del sistema de autentificación.

El sistema estará esperando a que se pase a través del lector un TAG de tipo tarjeta cuyo identificador este almacenado en una base de datos. En caso de que intente utilizar el sistema con un TAG no valido el sistema avisará de un intento de acceso por parte de un usuario no autorizado quedando dicha incidencia registrada en una base de datos. Se ha intentado introducir en el proyecto un sistema de autentificación registro y control de los coches, pero debido a problemas de lectura con el lector y los TAGs de tipo botón que irían ubicados sobre el coche para controlar sus tiempos, distancia recorrida, y propietario entre otras características no se ha podido implementar esta característica en el proyecto, aunque no se descarta como alternativa de mejora para un proyecto posterior.

4 Bases de Datos

4.1 Descripción General

Para la gestión de usuarios mediante tecnología RFID se ha desarrollado una pequeña base de datos que se describe a continuación.

Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para un posterior uso. Una base de datos consta de un servidor donde se vuelca información y de clientes que se conectaran al servidor de la base de datos para interactuar con la información alojada en él.

El método de adquisición de datos se produce mediante un sistema de cliente-servidor, en este caso particular el servidor esta alojado en el PC del sistema (local), aunque en otros tipos de bases de datos los servidores son de tipo remoto, y el acceso a este tipo de servidores se realiza mediante el uso internet.

De las diferentes bases de datos disponibles se ha optado por la base de datos de la compañía Oracle programada en SQL, MySQL. Esta elección se debe principalmente a la compatibilidad del PLC con MySQL gracias a una biblioteca específica en PC Worx.

SQL: es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como hacer cambios en ellas.

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Organiza la información mediante tablas, que son relacionadas al definir elementos comunes que hacen posible la combinación de datos de varias tablas en una consulta. Además permite el acceso concurrente a la información, de forma ordenada y segura, y es accesible a múltiples usuarios de forma simultánea.

El programa utilizado para la creación de la base de datos y de las tablas correspondientes es el MySQL Workbench, la herramienta oficial de MySQL para el diseño de bases de datos.

Además es necesaria la instalación del servidor (MySQL Server) donde se almacenan las bases de datos. Ambos pueden descargarse gratuitamente a través de su página web.

El PLC AXC 1050 de Phoenix Contact permite la conexión con un servidor de bases de datos MySQL o SQL Server a través de la dirección IP de la misma utilizando una conexión Ethernet, siendo necesaria la creación previa de la base de datos y su configuración.

4.2 MySQL Server Instalación

Debido a las bibliotecas usadas en el PC Worx recomiendan una versión del SQL Server 5 o inferior ya que versiones superiores pueden dar problemas de compatibilidad por lo que la versión que se ha instalado en el ordenador ha sido la 5.0 a continuación se describe la instalación del MySQL Server.

En el siguiente enlace se puede descargar el servidor http://dev.mysql.com/downloads/ y se selecciona "MySQL Community Server":

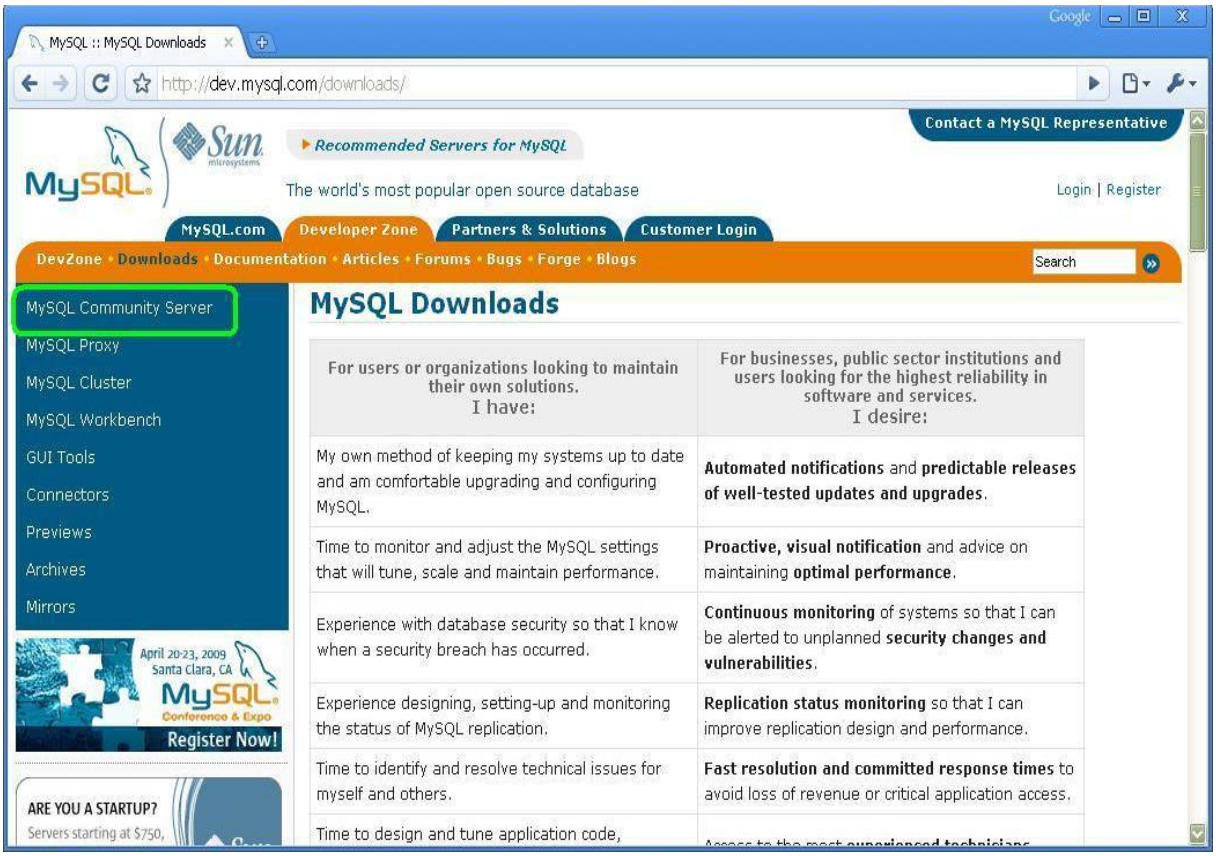


Ilustración 28- Instalación MySQL Server

Se localiza la versión de nuestro sistema operativo, en el caso particular del proyecto como se realiza bajo el sistema operativo de Microsoft Windows seleccionamos esa opción y descargamos el servidor.

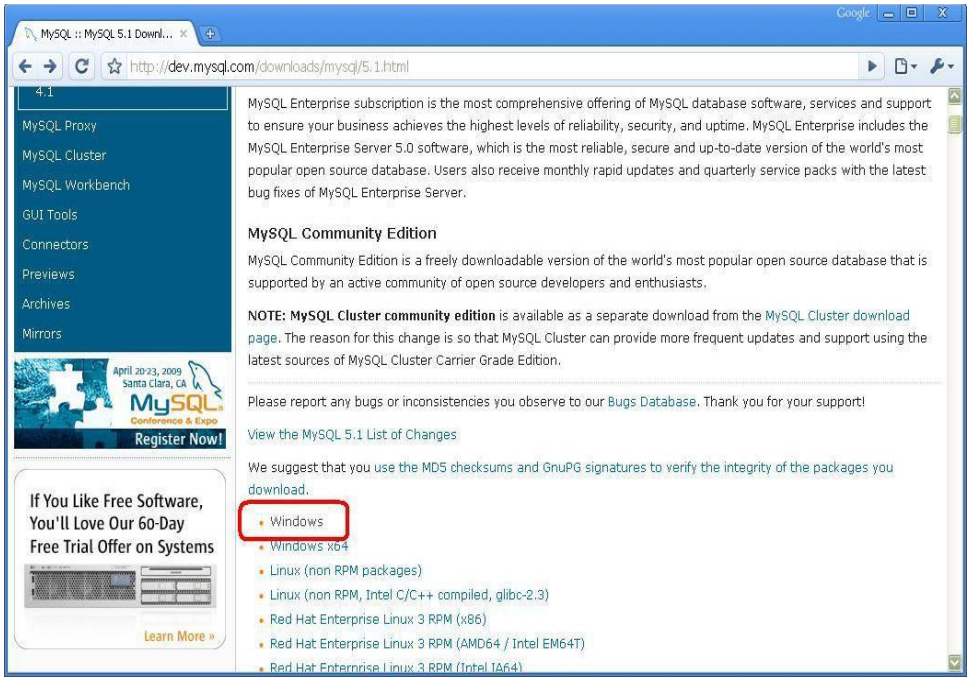


Ilustración 29 - Ilustración 15 - Instalación MySQL Server

Una vez descargado el servidor ejecutamos el instalador y se le va dando a next, cuando aparezca el tipo de instalación se selecciona complete, se le va dando a next hasta instalar el servidor, posteriormente aparecerá la opción de configuración del servidor, se seleciona el check, se le da a finish, a continuación de abrirá la opción para configurar el servidor SQL.

Ilustración 30 - - Instalación MySQL Server

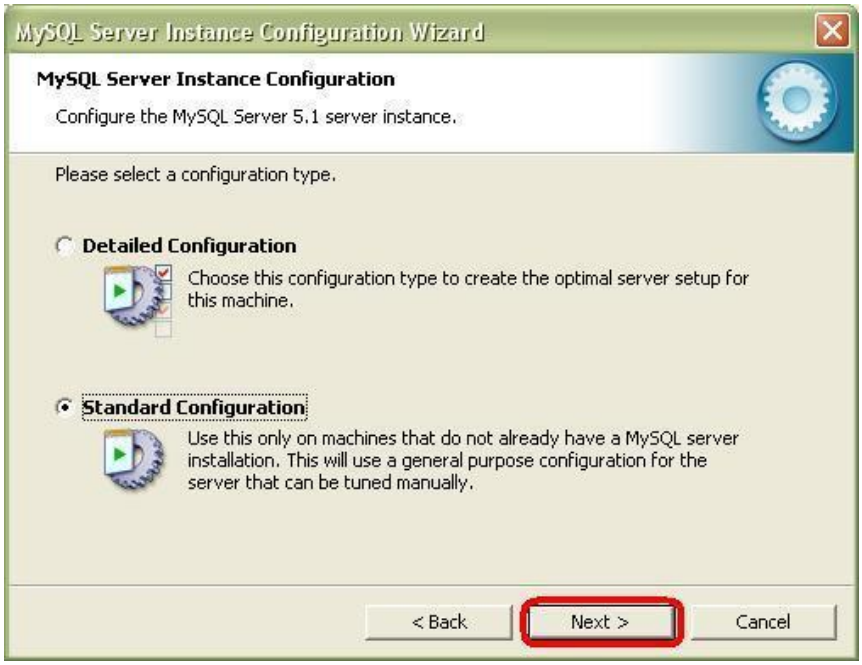


Ilustración 31 - Instalación MySQL Server

Se seleccionan las opciones "Install As Windows Service" e "Include Bin Directory in Windows Path". No siempre se desea que el motor arranque automáticamente cuando inicie el sistema operativo, para eso se deselecciona la opción "Launch the MySQL Server Automatically", si se hace, siempre se deberá iniciar el servidor manualmente. No hay problema si se deja seleccionada la opción. Presione "Next" para continuar.

Ilustración 32 - Instalación MySQL Server

Ilustración 33 - Instalación MySQL Server

Ilustración 34 - Instalación MySQL Server

Se introduce la contraseña para el usuario root (el administrador del motor). Si desea administrar el motor de forma remota, seleccione la opción "Enable root access from remote machines". En este caso particular no se ha diseñado para acceder a esta base de datos de manera remota. Si se desea una cuenta anónima (sin usuario ni clave) seleccione la opción "Create an Anonymous Account".

Al darle a next aparecerá otra pantalla en la que al ejecutarse cofigurará el servidor con todas las opciones elegidas anteriormente. Al finalizar el proceso de configuración la pantalla deberá indicar que los pasos se ejecutaron correctamente. Al darle a "Finish" se finaliza el proceso.

Ilustración 35- Instalación MySQL Server

Una vez configurado el servidor se puede probar el servidor ejecutándolo desde el menú de inicio del sistema operativo e introduciendo la contraseña que se configuró anteriormente.

4.3 Versión de MySQL Workbench

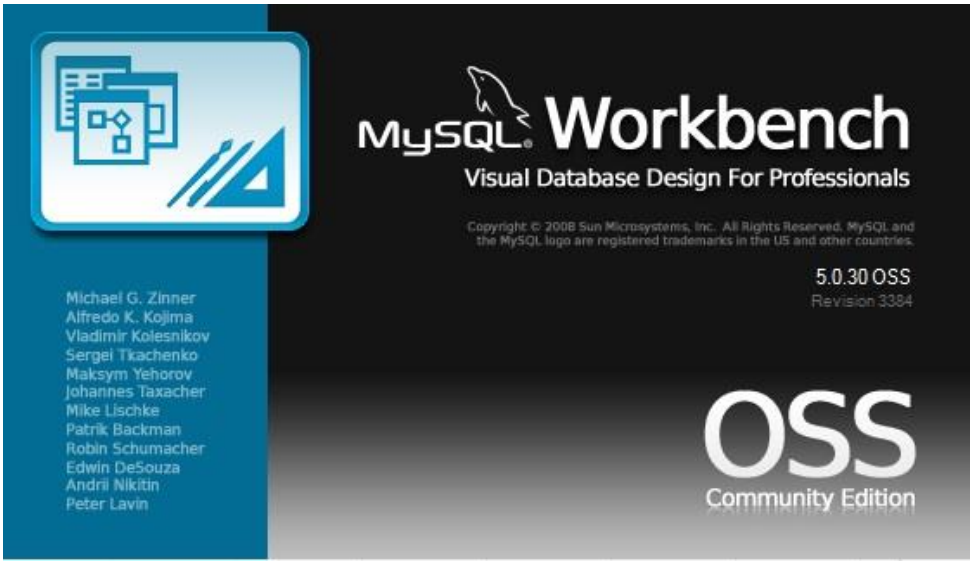


Ilustración 36 - Versión del programa MySQL Workbench utilizada

4.4 Tablas de la base de datos

La utilización de la base de datos se puede reducir a dos usos. Por un lado el almacenamiento y actualización de información relativa a los jugadores y a los coches disponibles. Y por otro lado el almacenamiento de determinados eventos que puedan ocurrir en el manejo del sistema como un registro de actividad y una registro de alarmas (Warnings). Para ello se han creado cuatro tablas: Players, Cars, Register y Warnings.

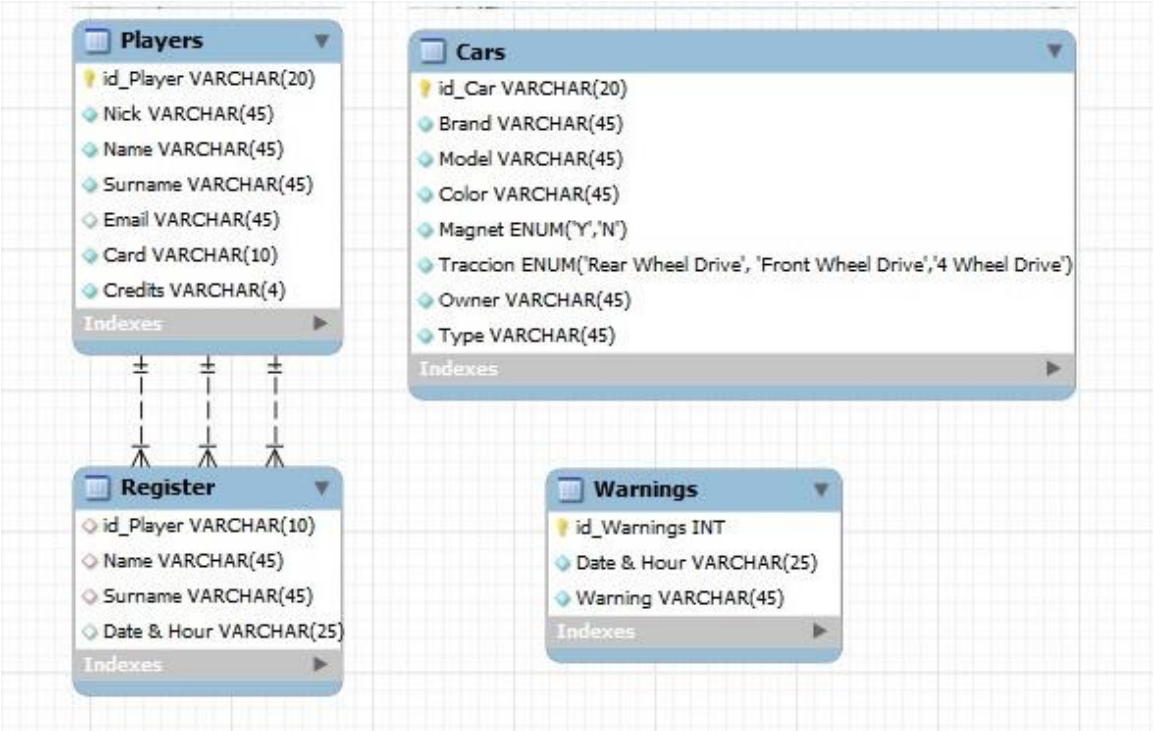


Ilustración 37 - Tablas de la base de datos

La mayoría del tipo de datos de las columnas son Varchar esto quiere decir que son de tipo STRING (cadena de carácter), y la longitud de la cadena se indica entre paréntesis en este caso particular se ha dejado la longitud que asignaba MySQL Workbench a los VARCHAR por defecto considerándose la extensión del STRING más que suficiente para asignar el valor preciso. No obstante pese a que en algunas columnas puede considerarse excesivo, es necesario mencionar que muchas columnas están diseñadas para autocompletarse solas, como en el caso de date o id_Warnings.

El tipo de dato ENUM sirve para obligar al usuario a elegir entre las opciones que muestran entre paréntesis, se ha considerado necesario incluirlo debido a que en las opciones como Magnet (Imán) o Traction (Tracción) solo tendrán una opciones precargadas.

4.4.1 Tabla Players

En esta tabla se almacenará toda la información relativa a los jugadores que utilizaran el sistema las columnas de la tabla son: un identificador (id_Player) único por jugador que será el id propio del chip integrado en el TAG RFID tarjeta, un nick/alias con el que el jugador interactuara con el sistema, nombre del usuario, apellidos, número de tarjeta que corresponderá con el número serigrafiado que tiene cada una de las tarjetas en la parte delantera inferior derecha, dirección de correo electrónico, y créditos. La dirección de correo electrónico para enviar cualquier dato referido al sistema al propietario de la dirección, aunque en este proyecto no tiene más que un dato informativo. Los créditos quedan implementados para realizar posteriormente un sistema de conteo y en caso de quedar sin créditos el sistema no permitiría jugar sin una recarga previa.

4.4.2 Tabla Cars

En esta tabla se almacenaran los datos relativos a todos los coches disponibles del circuito, las columnas de la tabla son: un identificador id_Car, único para cada coche, que será el id propio del chip integrado en el TAG RFID de tipo botón que estará colocado en el coche, la marca del coche, el modelo, tipo de tracción del coche, si es delantera, trasera o total, alimentación del motor del coche, propietario del coche, si posee imán o no, el fabricante del coche y el tipo de modelo del coche.

4.4.3 Tabla Register

Esta tabla esta relacionada con la tabla de jugadores Players, esta tabla lleva todo el registro de todos los eventos de "inicio de sesión" o Login que se haya realizado en el sistema. Las columnas de la tabla son: id_Player, Date en el que quedaría grabados la fecha y la hora del login.

4.4.4 Tabla Warnings

En esta tabla se almacenan todas las alarmas que se hayan registrado en sistema, las columnas de la tabla son un identificador de alarma que no esta vinculado a ningún tag como en las tablas anteriores, es un identificador que se irá autoincrementando a medida que se produzca una alarma esa es la razón por la variable es de tipo INT (entero), otra columna Date que almacenará la fecha y la hora en que se ha producido la alarma y otra columna en el que se almacenará el tipo de alarma (Warning), los warnings que se produzcan en el sistema, estos warnings estarán precargados y se describirán más adelante cuando se hable del código desarrollado en PC Worx.

Una vez diseñadas las tablas de la base de datos se procede a transformar el diseño en un script de SQL y exportarlo SQL Server para ello se utilizará otra herramienta que se describe a continuación.

4.5 MySQL Query Browser

El MySQL Query Browser es una herramienta del MySQL Workbench que permite entre otros usos visualizar el script generado por el MySQL Workbench de las tablas, compilarlo para comprobar que no haya ningún tipo de error en la sintaxis de SQL y lanzarlo al MySQL Server para poder operar con las tablas diseñadas.

Antes de lanzar las tablas creadas es necesario configurar la conexión con el servidor donde la herramienta alojarán las tablas de la base de datos.

En este caso particular a la conexión se la ha llamado xplore, el "server host" es local, el propio ordenador por lo que se ha configurado como localhost, el puerto es 3306 al ser localhost, el usuario se ha cogido el nombre por defecto que traía el propio MySQL, el password se ha elegido la palabra xplore, es importante mencionar que hace distinción entre mayúsculas y minúsculas por lo que en caso de no introducir correctamente el password dará un error.

Default Schema es el esquema de las tablas que mostrará por defecto. Una vez preconfigurada la conexión y si todo esta correcto al darle al OK se accedera al programa, una vez dentro solo hay que buscar el script creado con MySQL Workbench abrirlo y compilarlo y si el programa no encuentra ningún error en el script al ejecutar el script lo lanza al servidor SQL.

Ilustración 38 - Configuración para cargar la base de datos

Al darle a botón 'OK' se abre el programa, buscamos el script creado con el MySQL Workbench y lo abrimos, una vez abierto compilamos el script dándole al botón continue, si todo esta correcto no da ningún error por lo que se puede proceder a cargar el script en la base datos, con lo que las tablas quedan creadas.

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';

CREATE SCHEMA IF NOT EXISTS `xplore2015_WB` ;
USE `xplore2015_WB`;

-- -----------------------------------------------------
-- Table `xplore2015_WB`.`Cars`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `xplore2015_WB`.`Cars` ;

CREATE  TABLE IF NOT EXISTS `xplore2015_WB`.`Cars` (
  `id_Car` VARCHAR(20) NOT NULL ,
  `Brand` VARCHAR(45) NOT NULL ,
  `Model` VARCHAR(45) NOT NULL ,
  `Color` VARCHAR(45) NOT NULL ,
  `Magnet` ENUM('Y','N') NOT NULL ,
  `Traccion` ENUM('Rear Wheel Drive', 'Front Wheel Drive','4 Wheel Drive') NOT NULL ,
  `Owner` VARCHAR(45) NOT NULL ,
  `Type` VARCHAR(45) NOT NULL ,
  PRIMARY KEY (`id_Car`) )
ENGINE = InnoDB
COMMENT = 'Tabla a la que se le cargará de antemano los coches del sistema';


-- -----------------------------------------------------
-- Table `xplore2015_WB`.`Players`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `xplore2015_WB`.`Players` ;

CREATE  TABLE IF NOT EXISTS `xplore2015_WB`.`Players` (
  `id_Player` VARCHAR(20) NOT NULL COMMENT 'id_Player será el DNI del jugador, puesto que será
  `Nick` VARCHAR(45) NOT NULL ,
  `Name` VARCHAR(45) NOT NULL ,
  `Surname` VARCHAR(45) NOT NULL ,
  `Email` VARCHAR(45) NULL ,
  `Card` VARCHAR(10) NOT NULL ,
  `Credits` VARCHAR(4) NOT NULL ,
  PRIMARY KEY (`id_Player`) )
ENGINE = InnoDB
COMMENT = 'Tabla de cada usuario del sistema';

```

Ilustración 39 - Script parte 1

```
-- -----------------------------------------------------
-- Table `xplore2015_WB`.`Register`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `xplore2015_WB`.`Register` ;

CREATE  TABLE IF NOT EXISTS `xplore2015_WB`.`Register` (
  `id_Player` VARCHAR(10) NOT NULL ,
  `Name` VARCHAR(45) NOT NULL ,
  `Surname` VARCHAR(45) NOT NULL ,
  `Date & Hour` VARCHAR(25) NULL ,
  CONSTRAINT `id_Player`
    FOREIGN KEY (`id_Player` )
    REFERENCES `xplore2015_WB`.`Players` (`id_Player` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Name`
    FOREIGN KEY (`Name` )
    REFERENCES `xplore2015_WB`.`Players` (`Name` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Surname`
    FOREIGN KEY (`Surname` )
    REFERENCES `xplore2015_WB`.`Players` (`Surname` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE INDEX `id_Player` ON `xplore2015_WB`.`Register` (`id_Player` ASC) ;

CREATE INDEX `Name` ON `xplore2015_WB`.`Register` (`Name` ASC) ;

CREATE INDEX `Surname` ON `xplore2015_WB`.`Register` (`Surname` ASC) ;

```

Ilustración 40- Script parte 2

```
78 -- -----------------------------------------------------
79 -- Table `xplore2015_WB`.`Warnings`
80 -- -----------------------------------------------------
81 DROP TABLE IF EXISTS `xplore2015_WB`.`Warnings` ;
82
83 CREATE  TABLE IF NOT EXISTS `xplore2015_WB`.`Warnings` (
84   `id_Warnings` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
85   `Date & Hour` VARCHAR(25) NOT NULL ,
86   `Warning` VARCHAR(45) NOT NULL ,
87   PRIMARY KEY (`id_Warnings`) )
88 ENGINE = InnoDB;
89
90
91
92 SET SQL_MODE=@OLD_SQL_MODE;
93 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
94 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Ilustración 41 - Script parte 3

Las ilustraciones 12, 13 y 14 muestran el script de las tablas de la base de datos en SQL en el programa MySQL Query Browser cargado compilado y preparado para ser ejecutado en el servidor SQL. Una vez ejecutado el script al abrir el SQL Server se puede comprobar mediante comandos de SQL como efectivamente las tablas están cargadas en el servidor.

```
mysql>
mysql> describe players;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| id_Player | varchar(20) | NO   | PRI |         |       |
| Nick      | varchar(45) | NO   |     |         |       |
| Name      | varchar(45) | NO   |     |         |       |
| Surname   | varchar(45) | NO   |     |         |       |
| Email     | varchar(45) | YES  |     | NULL    |       |
| Card      | varchar(10) | NO   |     |         |       |
| Credits   | varchar(4)  | NO   |     |         |       |
+-----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql> describe cars;
+----------+-----------------------------------------------------------------+------+-----+---------+-------+
| Field    | Type                                                            | Null | Key | Default | Extra |
+----------+-----------------------------------------------------------------+------+-----+---------+-------+
| id_Car   | varchar(20)                                                     | NO   | PRI |         |       |
| Brand    | varchar(45)                                                     | NO   |     |         |       |
| Model    | varchar(45)                                                     | NO   |     |         |       |
| Color    | varchar(45)                                                     | NO   |     |         |       |
| Magnet   | enum('Y','N')                                                   | NO   |     |         |       |
| Traccion | enum('Rear Wheel Drive','Front Wheel Drive','4 Wheel Drive')    | NO   |     |         |       |
| Owner    | varchar(45)                                                     | NO   |     |         |       |
| Type     | varchar(45)                                                     | NO   |     |         |       |
+----------+-----------------------------------------------------------------+------+-----+---------+-------+
8 rows in set (0.00 sec)

mysql> describe register;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| id_Player   | varchar(10) | NO   | MUL |         |       |
| Name        | varchar(45) | NO   | MUL |         |       |
| Surname     | varchar(45) | NO   | MUL |         |       |
| Date & Hour | varchar(25) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> describe warnings;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| id_Warnings | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| Date & Hour | varchar(25)      | NO   |     |         |                |
| Warning     | varchar(45)      | NO   |     |         |                |
+-------------+------------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql>
```

Ilustración 42- Servidor SQL con las tablas cargadas

5 Programa de Control

La programación desarrollada en este proyecto toma como base el código de control realizado previamente. Este programa incluía la programación de diferentes modos de juego y por tanto algunas de las POUs (bloques funcionales, funciones y programas), tipos de datos y variables utilizados y/o ampliados en el presente trabajo. Estos módulos de ampliación para la integración de funcionalidades basadas en RFID serán explicados más adelante en este capítulo.

Se comienza con una breve descripción del estándar de programación IEC 61131-3 y las herramientas software de desarrollo y de depuración de la aplicación, suministradas por el fabricante del PLC AXC 1050 de Phoenix Contact.

5.1 IEC 61131-3

La norma IEC 61131-3 es la normativa internacional para la programación de los autómatas programables. Define 5 lenguajes distintos: LD (*Ladder Diagram*, o diagrama de contactos), FBD (*Function Block Diagram*, o diagrama de bloques de funciones), IL (*Instruction List*, o lista de instrucciones), SFC *(Sequential Function Chart*, o diagrama de funciones secuenciales), ST (Structured Text, o texto estructurado).

Un proyecto según la norma IEC 61131-3 consta de, al menos, una configuración. En cada configuración, encontramos uno o más recursos. Un recurso contiene todas las herramientas necesarias para ejecutar un programa determinado de usuario. Puede haber una o varias tareas por cada recurso, formadas por uno o varios programas, que a su vez pueden contener funciones y bloques funcionales. Una tarea puede ser cíclica (periódica), estar activada por un evento o de ejecución libre (ciclo scan).

5.2 PC Worx

PC Worx es el software de programación de PLCs de Phoenix Contact. Con él se configura la comunicación del PLC, es decir la estructura del bus del proyecto, en este caso utilizando la tecnología PROFINET, y se realiza el programa de control que luego se transfiere al autómata programable. Para la realización del programa es posible utilizar cualquiera de los 5 lenguajes de la norma IEC 61131-3 en las diferentes POUs que lo constituyen.

Las principales áreas de trabajo del PC Worx son las siguientes:

5.2.1 Área de trabajo de programación IEC

Área donde se realiza el programa. En la parte central aparece el código, a la izquierda se muestra el árbol del programa y en la inferior información diversa relativa al mismo.

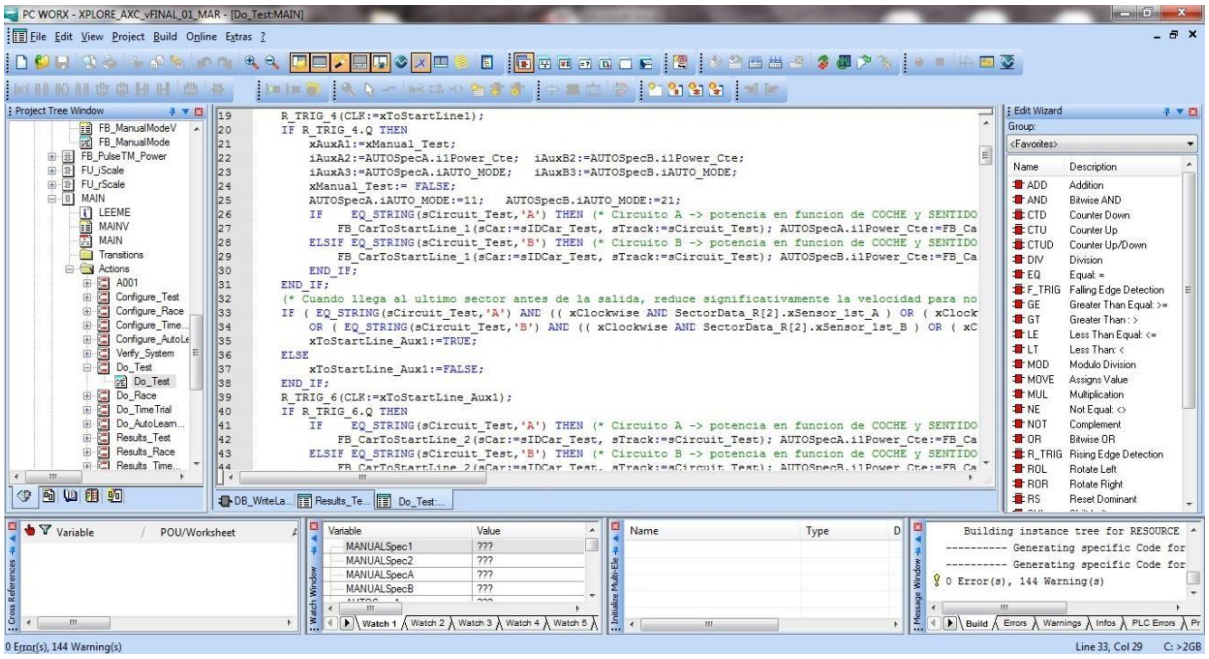


Ilustración 43 - Área de trabajo PC Worx

5.2.2 Área de trabajo de configuración del bus

Es la zona donde se configura el bus de comunicación, y donde se definen todos los módulos utilizados en el sistema.

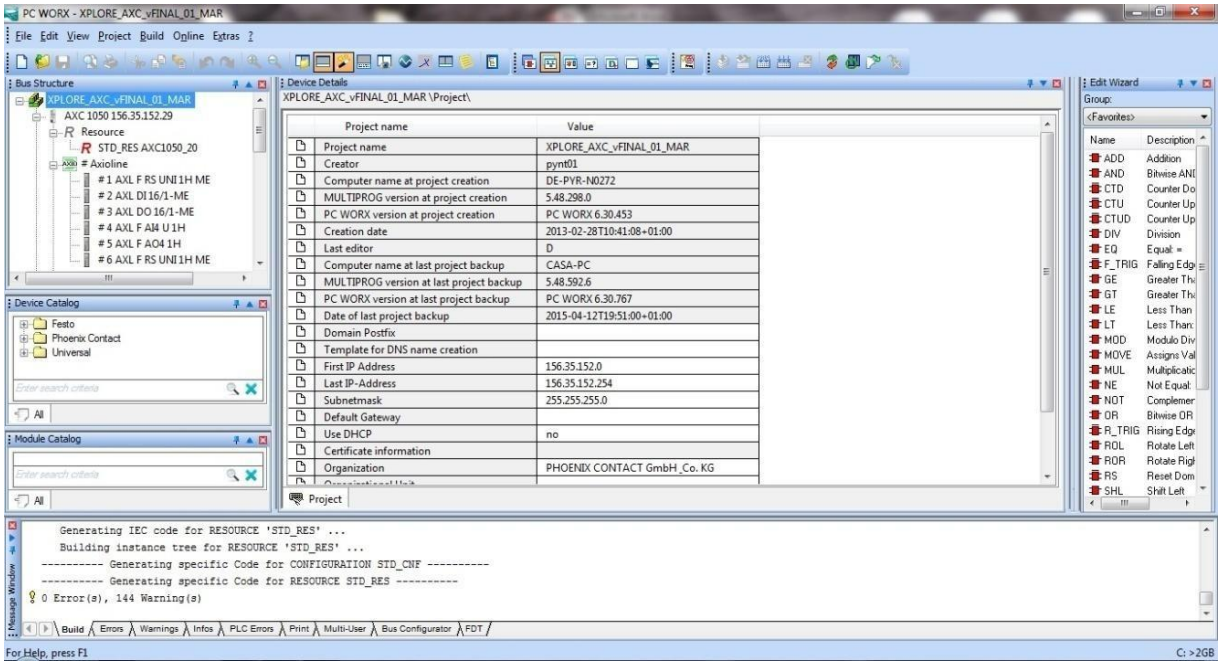


Ilustración 44 - Área de trabajo de configuración del bus

5.2.3 Área de trabajo de asignación de datos de proceso

En esta área se asignan las entradas y salidas físicas conectadas al controlador con las variables del programa.

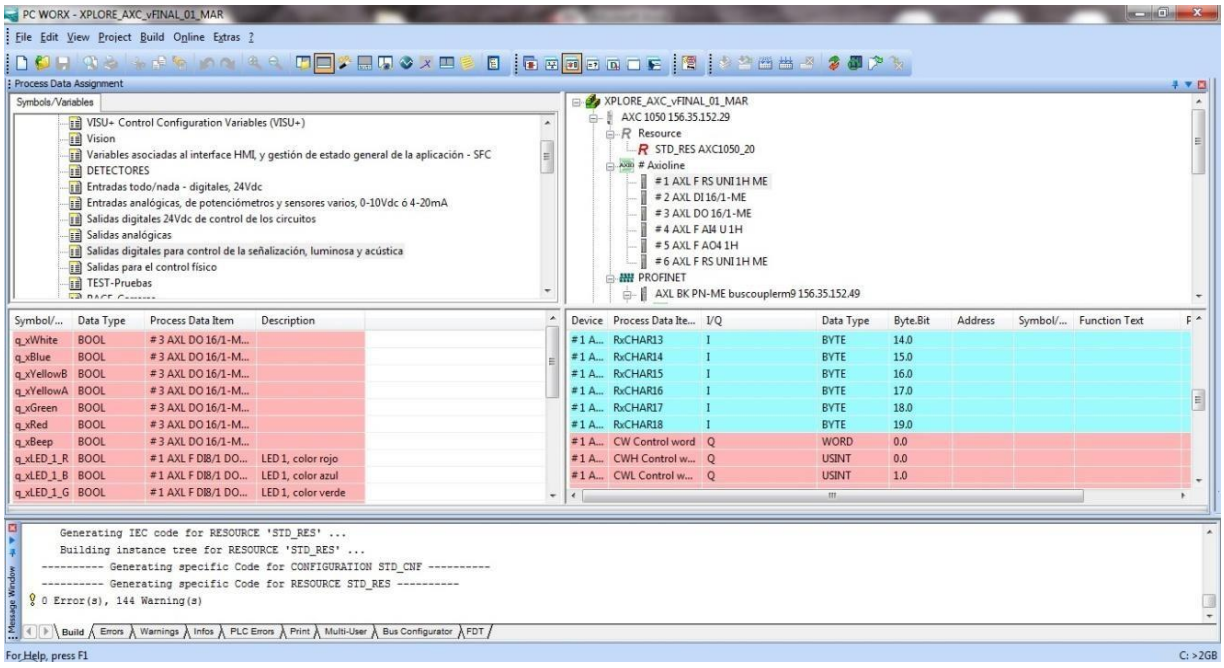


Ilustración 45 - Área de trabajo de asignación de datos de proceso

5.3 PC Worx SRT

PC Worx SRT es un simulador de PLC de la empresa Phoenix Contact. Instalado en un PC permite trabajar sobre él como si de un PLC se tratara, aunque sin las prestaciones en tiempo real. La ventaja que ofrece es la posibilidad de trabajar con el programa (simularlo, depurarlo...) sin ser necesario el uso del PLC y la instalación real, haciendo mucho más sencillo el trabajo. De esta forma el uso del PLC real se puede limitar a las comprobaciones finales de la puesta en marcha.

5.4 Tipos de Datos desarrollados

Para un correcto desarrollo del sistema RFID se ha tenido que desarrollar otro tipo de dato más y añadirlo al resto de tipos ya realizados.

5.4.1 RFID_Vector STRING

RFID_VectorString es un tipo de dato diseñado para cargar los valores procedentes de la base de datos, todo dato extraído será de tipo STRING (Cadena de carácter), por lo que se ha creado un tipo de datos específico para las variables asociadas a la extracción de datos.

Debido a que no es posible crear una variable dinámica que se vaya ajustando al tamaño deseado como solución se propone la creación un tipo de dato *RFID_VectorString* de tamaño determinado para llenar los vectores, en caso de que con un vector de tamaño 30 se quede corto habría que cambiar el tamaño por otro de mayor tamaño.

5.5 Programas de control desarrollados

A continuación se explican los programas que han tenido que ser modificados para adecuarlos a un correcto funcionamiento del sistema RFID.

5.5.1 Programa Visu +

Esta POU con estructura programa desarrollada con PCWorx permite llevar a cabo las conversiones necesarias en las variables para la comunicación con la aplicación SCADA que se ha desarrollado con la herramienta Visu +.

Una parte de este programa ya había sido creado por el proyectante que ha diseñado el SCADA general del proyecto Xplore, por lo que se ha aprovechado la estructura y se han añadido nuevas líneas de código para adaptarlo a las necesidades de la aplicación de supervisión y explotación que implementa este proyecto.

El problema principal que se trata de resolver, es que para poder visualizar las tablas de la base de datos es necesario descomponer el vector de strings en cadenas independientes ya que el estándar de comunicación OPC utilizado no admite vectores de estas características.

La variable *visu_icont* corresponderá a un botón asignado a visu que servirá para avanzar de grupo de visualización. Con el siguiente código visualizaremos las tablas de la base de datos de 5 en 5 con un sencillo cálculo posición x 5-4.

Si se quieren ver otras 5 filas se le da al botón por lo que contador aumentara y se cumplirá la condición para entrar en el código, se precarga en la variable *visu_pos* el valor del contador *5 y se irá restando en función de la posición que quiera mostrar.

```
(*******************************************************  TABLA COCHES  ********************************************************************************)

if xHMI_SHOW_CARS then
    xCars_Show:=true;
end_if;

if xCars_Show AND visu_icont_C=1 then
    sid_car1:=sid_car[1];  sBrand1:=sBrand[1];  sModel1:=sModel[1];  sColor1:=sColor[1];  sMotor1:=sMotor[1];  sTraccion1:=sTraccion[1];  sMagnet1:=sMagnet[1];
    sid_car2:=sid_car[2];  sBrand2:=sBrand[2];  sModel2:=sModel[2];  sColor2:=sColor[2];  sMotor2:=sMotor[2];  sTraccion2:=sTraccion[2];  sMagnet2:=sMagnet[2];
    sid_car3:=sid_car[3];  sBrand3:=sBrand[3];  sModel3:=sModel[3];  sColor3:=sColor[3];  sMotor3:=sMotor[3];  sTraccion3:=sTraccion[3];  sMagnet3:=sMagnet[3];
    sid_car4:=sid_car[4];  sBrand4:=sBrand[4];  sModel4:=sModel[4];  sColor4:=sColor[4];  sMotor4:=sMotor[4];  sTraccion4:=sTraccion[4];  sMagnet4:=sMagnet[4];
    sid_car5:=sid_car[5];  sBrand5:=sBrand[5];  sModel5:=sModel[5];  sColor5:=sColor[5];  sMotor5:=sMotor[5];  sTraccion5:=sTraccion[5];  sMagnet5:=sMagnet[5];
end_if;
```

Ilustración 46 - Separación de vector string en vectores independientes

En este caso, la tabla coches, la variable *visu_icont_X* se convierte en *visu_icont_C*, siendo para la tabla players *visu_icont_P*, *visu_icont _W* para warnings y en register *visu_icont _R*.

```
if xShow_More_Cars then
    visu_icont_C:=visu_icont_C+1;
    visu_pos:=visu_icont_C*5;

    visu_pos1:=visu_pos-4;
    visu_pos2:=visu_pos-3;
    visu_pos3:=visu_pos-2;
    visu_pos4:=visu_pos-1;
    visu_pos5:=visu_pos;

sid_car1:=sid_car[visu_pos1];       sBrand1:=sBrand[visu_pos1];    sModel1:=sModel[visu_pos1];  sColor1:=sColor[visu_pos1];               sMotor1:=sMotor[visu_pos1];
sTraccion1:=sTraccion[visu_pos1];   sMagnet1:=sMagnet[visu_pos1];  sOwner1:=sOwner[visu_pos1];  sManufacturer1:=sManufacturer[visu_pos1];  sType1:=sType[visu_pos1];

sid_car2:=sid_car[visu_pos2];       sBrand2:=sBrand[visu_pos2];    sModel2:=sModel[visu_pos2];  sColor2:=sColor[visu_pos2];               sMotor2:=sMotor[visu_pos2];
sTraccion2:=sTraccion[visu_pos2];   sMagnet2:=sMagnet[visu_pos2];  sOwner2:=sOwner[visu_pos2];  sManufacturer2:=sManufacturer[visu_pos2];  sType2:=sType[visu_pos2];

sid_car3:=sid_car[visu_pos3];       sBrand3:=sBrand[visu_pos3];    sModel3:=sModel[visu_pos3];  sColor3:=sColor[visu_pos3];               sMotor3:=sMotor[visu_pos3];
sTraccion3:=sTraccion[visu_pos3];   sMagnet3:=sMagnet[visu_pos3];  sOwner3:=sOwner[visu_pos3];  sManufacturer3:=sManufacturer[visu_pos3];  sType3:=sType[visu_pos3];

sid_car4:=sid_car[visu_pos4];       sBrand4:=sBrand[visu_pos4];    sModel4:=sModel[visu_pos4];  sColor4:=sColor[visu_pos4];               sMotor4:=sMotor[visu_pos4];
sTraccion4:=sTraccion[visu_pos4];   sMagnet4:=sMagnet[visu_pos4];  sOwner4:=sOwner[visu_pos4];  sManufacturer4:=sManufacturer[visu_pos4];  sType4:=sType[visu_pos4];

sid_car5:=sid_car[visu_pos5];       sBrand5:=sBrand[visu_pos5];    sModel5:=sModel[visu_pos5];  sColor5:=sColor[visu_pos5];               sMotor5:=sMotor[visu_pos5];
sTraccion5:=sTraccion[visu_pos5];   sMagnet5:=sMagnet[visu_pos5];  sOwner5:=sOwner[visu_pos5];  sManufacturer5:=sManufacturer[visu_pos5];  sType5:=sType[visu_pos5];

xShow_More_Cars:=false;
end_if;
```

Ilustración 47 - Código Botón + del SCADA

```
if xShow_Less_Cars then
    visu_icont_C:=visu_icont_C-1;
    visu_pos:=visu_icont_C*5;

    visu_pos1:=visu_pos-4;
    visu_pos2:=visu_pos-3;
    visu_pos3:=visu_pos-2;
    visu_pos4:=visu_pos-1;
    visu_pos5:=visu_pos;

    sid_car1:=sid_car[visu_pos1];       sBrand1:=sBrand[visu_pos1];    sModel1:=sModel[visu_pos1];  sColor1:=sColor[visu_pos1];               sMotor1:=sMotor[visu_pos1];
    sTraccion1:=sTraccion[visu_pos1];   sMagnet1:=sMagnet[visu_pos1];  sOwner1:=sOwner[visu_pos1];  sManufacturer1:=sManufacturer[visu_pos1];  sType1:=sType[visu_pos1];

    sid_car2:=sid_car[visu_pos2];       sBrand2:=sBrand[visu_pos2];    sModel2:=sModel[visu_pos2];  sColor2:=sColor[visu_pos2];               sMotor2:=sMotor[visu_pos2];
    sTraccion2:=sTraccion[visu_pos2];   sMagnet2:=sMagnet[visu_pos2];  sOwner2:=sOwner[visu_pos2];  sManufacturer2:=sManufacturer[visu_pos2];  sType2:=sType[visu_pos2];

    sid_car3:=sid_car[visu_pos3];       sBrand3:=sBrand[visu_pos3];    sModel3:=sModel[visu_pos3];  sColor3:=sColor[visu_pos3];               sMotor3:=sMotor[visu_pos3];
    sTraccion3:=sTraccion[visu_pos3];   sMagnet3:=sMagnet[visu_pos3];  sOwner3:=sOwner[visu_pos3];  sManufacturer3:=sManufacturer[visu_pos3];  sType3:=sType[visu_pos3];

    sid_car4:=sid_car[visu_pos4];       sBrand4:=sBrand[visu_pos4];    sModel4:=sModel[visu_pos4];  sColor4:=sColor[visu_pos4];               sMotor4:=sMotor[visu_pos4];
    sTraccion4:=sTraccion[visu_pos4];   sMagnet4:=sMagnet[visu_pos4];  sOwner4:=sOwner[visu_pos4];  sManufacturer4:=sManufacturer[visu_pos4];  sType4:=sType[visu_pos4];

    sid_car5:=sid_car[visu_pos5];       sBrand5:=sBrand[visu_pos5];    sModel5:=sModel[visu_pos5];  sColor5:=sColor[visu_pos5];               sMotor5:=sMotor[visu_pos5];
    sTraccion5:=sTraccion[visu_pos5];   sMagnet5:=sMagnet[visu_pos5];  sOwner5:=sOwner[visu_pos5];  sManufacturer5:=sManufacturer[visu_pos5];  sType5:=sType[visu_pos5];

    xShow_Less_Cars:=false;
end_if;
```

Ilustración 48 - Código Botón – del SCADA

Activación del botón *Admin Tables* del SCADA, con este código se nombra administrador a los diferentes TAGs, en este caso particular se ha nombrado un administrador y se tienen precargados otros dos, si se quiere nombrar otro administrador se pueden quitar los comentarios o por el contrario se puede nombrar otro TAG que no este previamente cargado, copiando la estructura ya desarrollada.

```
(************************************************************** ADMIN BOTTOM ***************************************************************)
(*Con este codigo hago que el boton de administracion de la base de datos en visu solo aparezca cuando se lea un id unico de un tag concreto,
 en el caso de querer hacer a una tarjeta admin habria que poner una disyuncion 'o' e ir poniendo los id que se quieran asignar a administracion*)

xAdmin_Table:=EQ_STRING(sTAG,'50676849674950'); (*TAG=0F02CD1C12*)

(*xAdmin_Table:=EQ_STRING(sTAG,'50676849675756');
  xAdmin_Table:=EQ_STRING(sTAG,'50676849655553');*)
(*********************************************************  FIN ADMIN BOTTON  ***************************************************************)
```

Ilustración 49 - Botón de administración SCADA

Código del botón *check* del SCADA para comprobar que no se va a introducir ningún registro repetido en la base de datos.

```
(********************************************************** Check Update Database *****
if xTag_Registrado =false and xEnd_Checking then
    (*el tag no esta registrado *)
    xHMI_Visu_Tag_Registrado:=true; (*variable para activar boton update database*)

end_if;

if xHMI_Visu_Check then
    xHMI_Show_OK_Button:=true;
end_if;

if xHMI_OK_Button then (*Variable para activar boton ok en el hmi*)
    xTag_Registrado:=false;
    xHMI_Visu_Tag_Registrado:=false;
    xEnd_Checking:=false;
(*  xHMI_Visu_Check:=true;*)
end_if;

(****************************************************** End Check Update Database *****
```

Ilustración 50 – Botón de Check

Código para el descuento de créditos, en cuanto se detecta un logueo se vuelca el valor de los créditos del TAG logueado, en la variable *iCredits*. Cuando la variable *xStart_TON* cambia a true, empieza a descontar créditos almacenados en la variable *iCredits*, y se vuelcan a la base de datos.

Por otro lado también se comprueba si hay suficientes créditos para poder operar con el sistema.

```
(************************************************ Programa descuento de credit
if xlogin then
    iCredits:=STRING_TO_INT (sVisu_Credits);
end_if;

s_Credits:=INT_TO_STRING(iCredits,'%03d');

if xStart_TON then<
    TON_1(IN:=NOT TON_1.Q,PT:=T_Credit_TIMER);
    if TON_1.Q then
    iCredits:=iCredits-1;
    sVisu_Credits:=INT_TO_STRING(iCredits,'%03d');
    xUpdate_Credits:=true;
    end_if;
end_if;
(* Colocamos un temporizador que cada tiempo PT de un pulso y se ponga solo a 0
 pondemos una variable de control a 1 *)

(* En caso de no haber suficientes creditos para jugar el sistema avisara por hm
    una vez restablecidos los creditos, la variable se pondra a false sola permi

if iCredits < 0 then
    xInsufficient_Credits:=true;
else
    xInsufficient_Credits:=false;
end_if;

(************************************************************** Fin de progra
```

Ilustración 51 - Programa descuenta créditos

5.5.2 Programa Main

El programa principal (MAIN) está programado en lenguaje SFC, con una etapa inicial y varias ramas, una por cada modo de juego: prueba, carrera, contrarreloj, aprendizaje automático, verificación y rankings.

La implementación de la tecnología RFID y de las llamadas a los bloques funcionales en el programa principal consiste principalmente en la adición de dos etapas colocadas en las primeras etapas del SFC, pequeños fragmentos de código alojados en las siguientes etapas del SFC para el reseteo de variables que se han ido activando en las etapas anteriores.

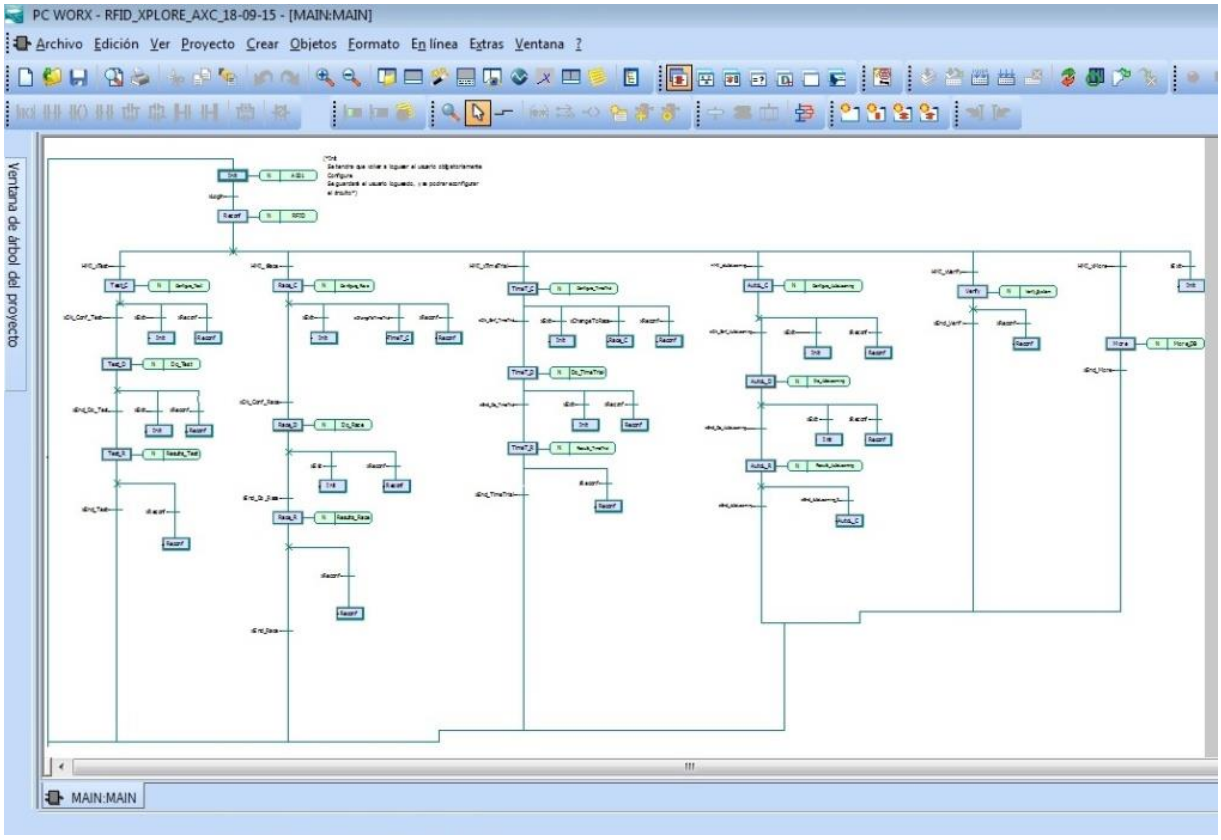


Ilustración 52 - Main

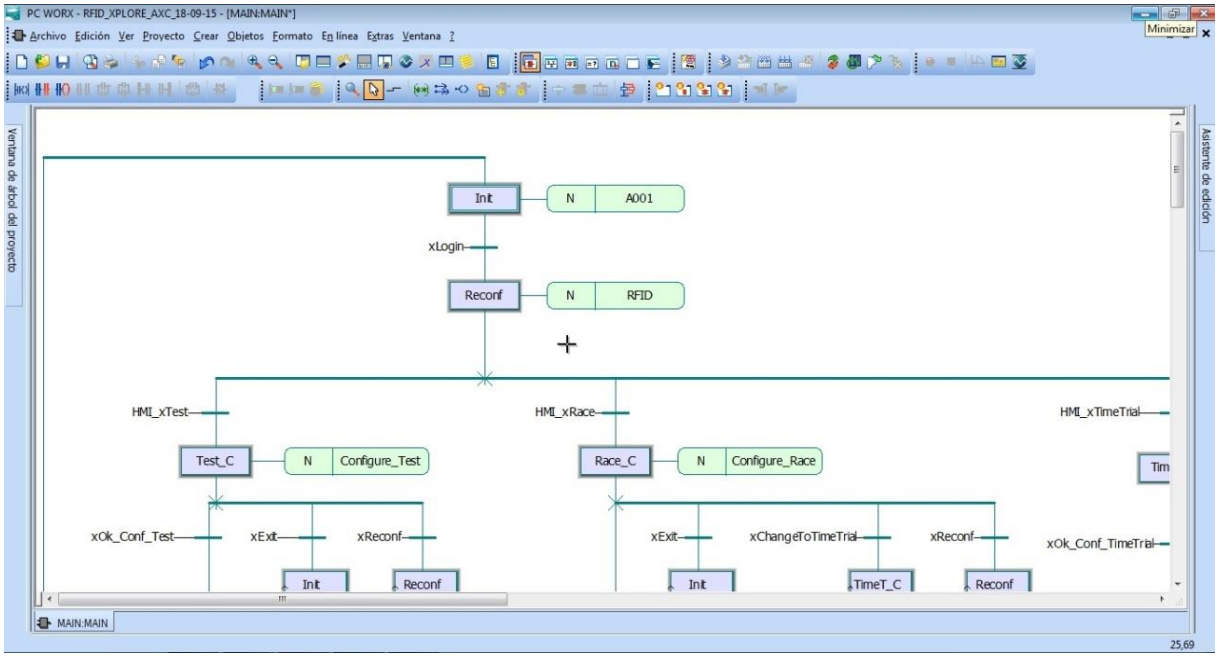


Ilustración 53 – Main

5.5.3 P_SerialComm

Este programa ha sido desarrollado para la comunicación entre el PLC y el resto de sistemas periféricos que interactúan con el PLC, el sistema de cámaras de visión, el drone, el lector de RFID a través del puerto serie. La parte del programa para la comunicación del PLC con el sistema de cámaras de visión y con el drone ya había sido desarrollado por otros miembros del equipo del proyecto Xplore por lo que este epígrafe se centrará en la parte del programa desarrollado para la comunicación puerto serie del lector RFID. Se ha mantenido la misma estructura y lenguaje de programación que el que ya habían diseñado otros miembros del equipo al desarrollar el programa SerialComm para las comunicaciones con las cámaras y el drone.

El programa hace uso de una biblioteca específica de Phoenix llamada AXL ComSerial necesaria para establecer la comunicación la cual se detallará más adelante.

Variables de Entrada (VAR_INPUT)

Variable	Nombre	Descripción
xActivate	BOOL	El bloque es activado.
xReset	BOOL	Los errores son borrados y el bloque es reiniciado.
strCR	STRING	Comunicación de referencia, resultados de la comunicación de interbus.
xAutoReset	BOOL	Los errores son detectados automaticamente e indicados por la variable xError
udtPara	COM_UDT_RS232_Para_V1	Estructuras con variables para pametrizar el modulo
xDTR	BOOL	La señal DTR es controlado. Solo si el modo DTR es activado.
xSend	BOOL	Parametro de disparo. El dato es enviado en flanco positivo.
uiSendLength	UINT	Numero de bytes para ser enviado.
xRecieve	BOOL	Parametro de disparo,el dato es leido en flaco positivo.

uiRcvLenght	UINT	Numero de bytes para ser leido, si el valor es 0, todos los datos disponibles son leidos.
tTimeout	TIME	Valor de tiempo de muestreo cuando se envia y recibe
winputAddress	WORD	Datos de proceso

Tabla 5 - Variables Entrada P_SerialComm

Variables de Salida (VAR_OUTPUT)

Nombre	Tipo	Descripcion
xReady	BOOL	El bloque esta preparado para ejecutar, cuando ejecuta el parametro esta en False
xDone	BOOL	True: EL servicio se ha ejecutado satisfactoriamente Cuando envía el dato se ha enviado correctamente Cuando Recibe: el nuevo dato ha sido leído y es disponible en arrRcvData
uiRcvBufferLength	UINT	Numero de bytes recibidos
xRcvBufferNotEmpty	BOOL	El buffer de recepción de datos no esta vacio, los datos pueden ser leídos
udtAddData	COM_UDT_RS232_ Data_V1	Estructuras con variables para parametrizar el modulo
xError	BOOL	Indica si ha habido algun error se pone a false
wDiagCode	WORD	Codigo de diagnostico
wAddDiagCode	WORD	Codigo de diagnostico adicional
wOutputAddress	WORD	Datos de proceso

Tabla 6 - Variables de Salida P_SerialComm

Variables de entrada/salida (VAR_IN_OUT)

Nombre	Tipo	Descripción
arrSendData	COM_ARR [1..xxx] OF BYTE	Array con los datos para ser enviados, este bloque de funciones esta disponible en diferentes tamaños 128,256,512,1024.
arrRcvData	COM_ARR [1..xxx] OF BYTE	Array con los datos recibidos, este bloque de funciones esta disponible en diferentes tamaños 128,256,512,1024.

Tabla 7 – Variables Entrada/Salida P_SerialComm

```
121
122
123 (************************************************** RFID ************************************
124 if xCard then  (* lee datos de TAG *)
125     AXL_RSUNI_PD_V1_00_3(xActivate:=xCard, xEndToEnd:= false, xAck:=xAck, xAutoAck:=xAutoAck,
126                          xDTR:=false, xReadStatusCounter:= xReadStatusCounter, xSend:=false,
127                          uiSendLength:=uint#0, xReceive:=xRcvBufferNotEmpty and not xDataReady,
128                          uiRcvLength:=uint#14, tTimeout:= t#0ms,
129                          arrInputAddress:= arrInputAddress, arrOutputAddress:= arrOutputAddress,
130                          arrRcvData:=arrRecvData, arrSendData:=arrSendData);
131
132     xActive:=AXL_RSUNI_PD_V1_00_3.xActive;
133     xBusy:=AXL_RSUNI_PD_V1_00_3.xBusy;
134     xError:=AXL_RSUNI_PD_V1_00_3.xError;
135     wDiagCode:=AXL_RSUNI_PD_V1_00_3.wDiagcode;
136     wAddDiagCode:=AXL_RSUNI_PD_V1_00_3.wAddDiagCode;
137     strDiag:=AXL_RSUNI_PD_V1_00_3.strDiag;
138     udtStatusCounter:=AXL_RSUNI_PD_V1_00_3.udtStatusCounter;
139     udtStatusSerialInterface:=AXL_RSUNI_PD_V1_00_3.udtStatusSerialInterface;
140     xStatusFailure:=AXL_RSUNI_PD_V1_00_3.xStatusFailure;
141     xRcvBufferNotEmpty:=AXL_RSUNI_PD_V1_00_3.xRcvBufferNotEmpty;
142     xReadCounterDone:=AXL_RSUNI_PD_V1_00_3.xReadCounterDone;
143     xSendDone:=AXL_RSUNI_PD_V1_00_3.xSendDone;
144
145     xDataReady:=AXL_RSUNI_PD_V1_00_3.xRcvNDR;
146     uiNumBytes:=AXL_RSUNI_PD_V1_00_3.uiRcvDataLength;
147
148     arrInputAddress:=AXL_RSUNI_PD_V1_00_3.arrInputAddress;
149     arrOutputAddress:=AXL_RSUNI_PD_V1_00_3.arrOutputAddress;
150     arrRecvData:=AXL_RSUNI_PD_V1_00_3.arrRcvData;
151     arrSendData:=AXL_RSUNI_PD_V1_00_3.arrSendData;
152 end if;
```

Ilustración 54 - Programa desarrollado para la conexión puerto serie

El programa tiene un funcionamiento muy sencillo, puesto que al estar vinculado a una tarea cíclica se va a estar ejecutando constantemente cada cierto tiempo. El código queda en espera hasta que un usuario pase una tarjeta por el lector, automáticamente la variable *xCard* se pone a true y el lector vuelca el contenido de la información de la tarjeta.

5.6 Configuración de *Tareas*

Hay una sola tarea, que incluye todos los programas del proyecto y que se repite en cada ciclo de scan.

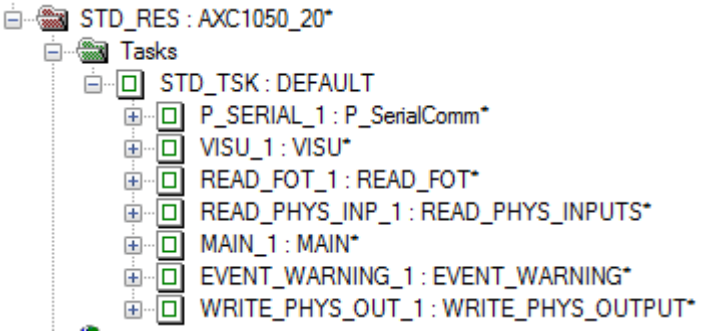


Ilustración 55 - Tareas Fragmento del árbol del proyecto

Librería PC Worx para SQL

Para implementar la funcionalidad de trabajar con bases de datos desde el propio PLC se requiere el uso de una biblioteca específica en el programa de PC Worx, DBFL_SQL_V1_17, de la que se utilizan varios bloques funcionales que serán explicados a continuación.

5.6.1 DBFL_MySQL_ACCESS

Bloque de conexión a la base de datos. Permite conectar con la base de datos y enviar un comando SQL.

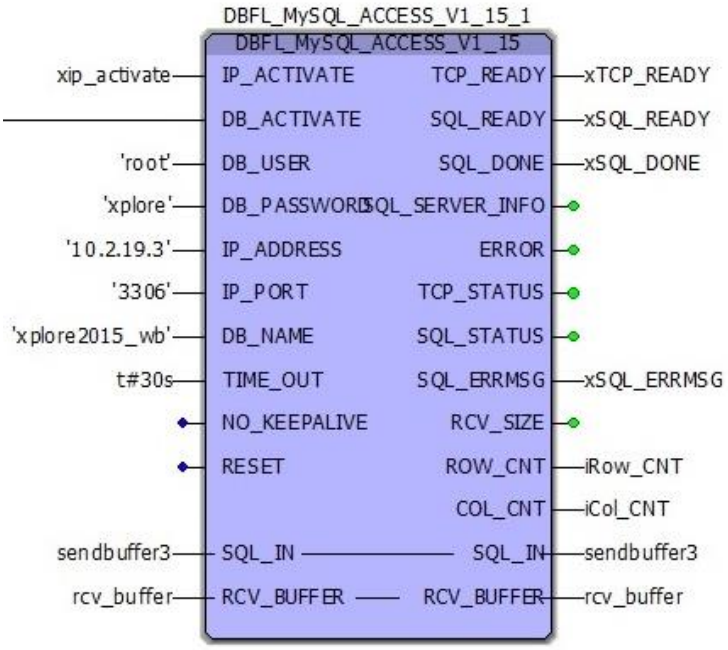


Ilustración 56- Bloque Funcional MySQL ACCESS

Variables de Entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
IP_ACTIVATE	BOOL	Establecer conexión con la base de datos
DB_ACTIVATE	BOOL	Envío del comando SQL
DB_USER	STRING	Usuario para la conexión
DB_PASSWORD	STRING	Contraseña para la conexión
IP_ADDRESS	STRING	Dirección IP del servidor
IP_PORT	STRING	Puerto de conexión con el servidor
DB_NAME	STRING	Nombre de la base de datos
TIME_OUT	TIME	Tiempo de reconexión por error
RESET	BOOL	Resetear la conexión

Tabla 8 - Variables Entrada MySQL Access

Variables de Salida (VAR_OUTPUT)

NOMBRE	TIPO	DESCRIPCIÓN
TCP_READY	BOOL	(TRUE) Conexión establecida con la base de datos
SQL_READY	BOOL	(TRUE) Preparado para el envío del comando SQL
SQL_DONE	BOOL	(TRUE) Comando SQL ejecutado
SQL_SERVER_INFO	DBFL_UDT_MySQL_ Server_Info	Información sobre el estado del servidor MySQL
ERROR	BOOL	(TRUE) Se ha producido un error
TCP_STATUS	DINT	Código de error TCP
SQL_STATUS	DINT	Código de error SQL
SQL_ERRMSG	DBFL_UDT_STRIN G_255	Información sobre el SQL
RCV_SIZE	DINT	Tamaño de los datos recibidos
ROW_CNT	DINT	Número de filas utilizadas
COL_CNT	DINT	Número de columnas utilizadas

Tabla 9 - Variables Salida MySQL Access

Variables de Entrada/Salida (VAR_IN_OUT)

NOMBRE	TIPO	DESCRIPCIÓN
SQL_IN	DBFL_ARR_BYTE_0_ 1439	Comando SQL en array de bytes
RCV_BUFFER	DBFL_ARR_BYTE_0_ 1439	Contiene los datos recibidos por el protocolo MySQL

Tabla 10 - Variables Entrada/Salida MySQL Access

5.6.2 DBFL_MySQL_DECODE

Bloque funcional que evalúa una tabla recibida de la base de datos, convirtiendo los arrays de bytes a otros tipos de datos. Se usa, por tanto, como continuación del bloque de acceso comentado antes.

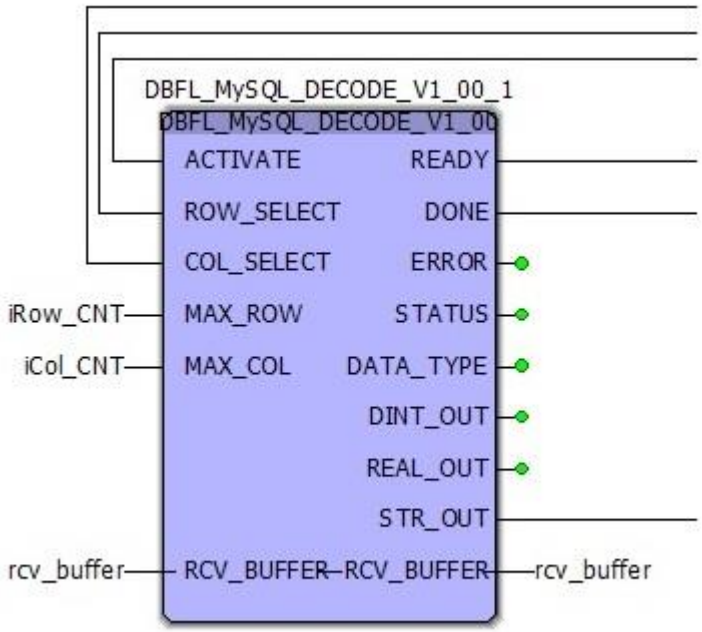


Ilustración 57 - Bloque funcional MySQL Decode

Variables de Entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
ACTIVATE	BOOL	Flanco positivo: decodifica la información obtenida en la entrada RCV_BUFFER
ROW_SELECT	DINT	Fila a evaluar
COL_SELECT	DINT	Columna a evaluar
MAX_ROW	DINT	Número máximo de filas en la tabla
MAX_COL	DINT	Número máximo de columnas en la tabla

Tabla 11 - Variables Entrada MySQL Decode

Variables de Salida (VAR_OUTPUT)

NOMBRE	TIPO	DESCRIPCIÓN
READY	BOOL	TRUE: Preparado para el envío del comando
DONE	BOOL	TRUE: Comando ejecutado
ERROR	BOOL	TRUE: Se ha producido un error
STATUS	INT	Código de error
DATA_TYPE	STRING	Tipo de dato de la celda seleccionada
DINT_OUT	DINT	Valor de la celda seleccionada (si es tipo DINT)
REAL_OUT	REAL	Valor de la celda seleccionada (si es tipo REAL)
STR_OUT	STRING	Valor de la celda seleccionada (si es tipo STRING)

Tabla 12 - Variables Salida MySQL Decode

Variables de Entrada/Salida (VAR_IN_OUT):

NOMBRE	TIPO	DESCRIPCIÓN
RCV_BUFFER	DBFL_ARR_BYTE_0_1439	Datos recibidos (del bloque MySQL_ACCESS)

Tabla 13 - Variables Entrada/Salida MySQL Decode

5.6.3 DBFL_CODE

Bloque de codificación de comandos SQL, que permite codificar la información para enviarla a la base de datos.

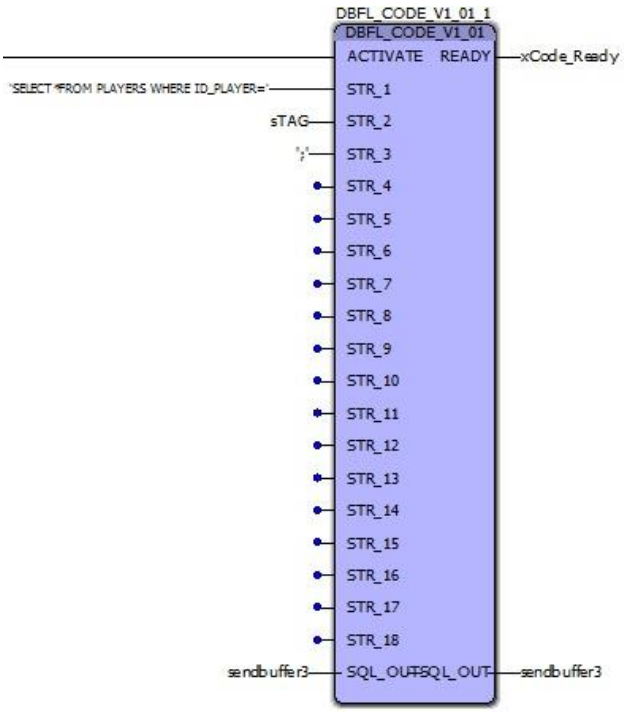


Ilustración 58 - Bloque Funcional DBFL Code

Variables de Entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
ACTIVATE	BOOL	Flanco positivo: la información de STR_1 – 18
STR_1 – 18	STRING	Entrada de variables STRING para formar el comando

Tabla 14 - Variables Entrada MySQL Code

Variables de Salida (VAR_OUT)

NOMBRE	TIPO	DESCRIPCIÓN
Ready	BOOL	(TRUE) Datos aceptados

Tabla 15 - Variables Salida MySQL Code

Variables de entrada/salida (VAR_IN_OUT)

NOMBRE	TIPO	DESCRIPCIÓN
SQL_OUT	DBFL_ARR_BYTE_0_ 1439	Comando SQL en array de bytes

Tabla 16 - Variables Entrada/Salida MySQL Code

5.7 Librería COMSERIAL

5.7.1 IL_RS232

Variables de Entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xActivate	BOOL	El Bloque esta Activado
xReset	BOOL	Errores borrados bloque reiniciado
strCR	STRING	Comunicación de referencia resultados de la comunicación por interbus
xAutoReset	BOOL	Errores detectados automáticamente cuando se detecte un error se reinicia el bloque
udtPara	COM_UDT_RS232_Para_V1	Estructura con variables para parametrizar el bloque
xDTR	BOOL	La señal DTR es controlada
xSend	BOOL	Se envia el dato con un flanco positivo

uiSendLenght	UINT	Numero de bits para ser mandados
xReceive	BOOL	El dato es leido en flanco positivo
uiRcvLength	UNIT	Numero de bytes para ser leidos
tTimeout	Time	Monitoriza el tiempo de envio y recepción
wInputAddress	Word	Datos de proceso

Tabla 17 - Variables Entrada Librería Comserial

Variables de Salida (VAR_OUTPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xReady	BOOL	El bloque esta preparado para ejecutar, cuando ejecuta el parametro esta en False
xDone	BOOL	True: EL servicio se ha ejecutado satisfactoriamente Cuando Envia el dato se ha enviado correctamente Cuando Recibe: el nuevo dato ha sido leido y es disponible en arrRcvData
uiRcvBufferLength	UINT	Numero de bytes recibidos
xRcvBufferNotEmpty	BOOL	El buffer de recepcion de datos no esta vacio, los datos pueden ser leidos
udtAddData	COM_UDT_RS232_Data_V1	Estructura con
xError	BOOL	Indica si ha habido algun error se pone a false
wDiagCode	WORD	Codigo de diagnostico
wAddDiagCode	WORD	Codigo de diagnostico adicional
wOutputAddress	WORD	Datos de proceso

Tabla 18 - Variables Salida Librería Comserial

5.8 Bloques funcionales desarrollados

Las operaciones a realizar con la base de datos consisten en interactuar en tablas, y no es necesaria la creación de ninguna nueva. Concretamente son necesarias tareas necesarias son las siguientes:

- Leer los registros de las tablas Players, Cars, Register y Warnings.
- Escribir en las tablas Players, Cars, Register y Warnings.
- Actualizar registros de las tablas Players, Cars, Register y Warnings.
- Borrar registros de las tablas Players, Cars, Register y Warnings

Para realizar estas funciones se ha desarrollado un SCADA en el programa VISU+ que se describirá más adelante tanto el desarrollo de la aplicación SCADA como el manual de usuario. Para separar la programación de la parte RFID de la parte ya desarrollada se ha desarrollado una carpeta llamada RFID donde estarán alojados todos los bloques funcionales desarrollados para la parte RFID del proyecto.

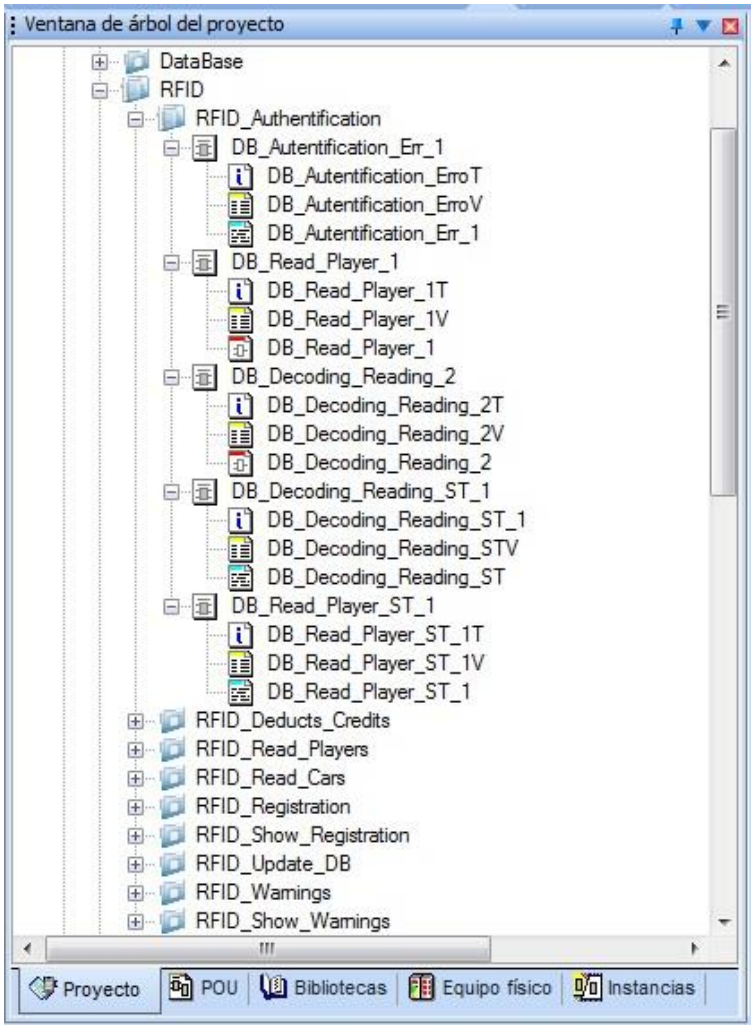


Ilustración 59 - Bloques funcionales de la base de datos en el árbol de proyecto

5.8.1 ASCII_CONV

Bloque funcional desarrollado para realizar una conversión ASCII, esta conversión es necesaria debido a que el código interno de los TAGs Tarjetas está cargado en código ASCII, este bloque funcional está relacionado con el programa *P_SerialComm*, cada vez que se pasa un TAG por el lector este, vuelca la información contenida en un vector de buffer de lectura en código decimal en la biblioteca ComSerial, por lo que es necesario si se quiere saber la relación del código interno de la tarjeta con el código serigrafiado en la propio TAG.

La conversión es un gran case que asigna el numero decimal del código interno del TAG al código ASCII, debido a que en los TAGs solo aparecen serigrafiados números del 0 al 9 y letras mayúsculas solo se ha realizado la conversión para esos valores siendo el 48 el carácter 0 y el 57 el carácter 9 y del 65 el carácter A y el 90 el carácter Z.

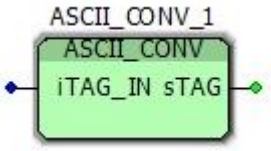


Ilustración 60 - Bloque Funcional ASCII CONV

```
(*Funcion de conversion a codigo ASCII*)
CASE iTAG_IN OF

        48: sTAG:='0';
        49: sTAG:='1';
        50: sTAG:='2';
        51: sTAG:='3';
        52: sTAG:='4';
        53: sTAG:='5';
        54: sTAG:='6';
        55: sTAG:='7';
        56: sTAG:='8';
        57: sTAG:='9';

        65: sTAG:='A';
        66: sTAG:='B';
        67: sTAG:='C';
        68: sTAG:='D';
        69: sTAG:='E';
        70: sTAG:='F';
        71: sTAG:='G';
        72: sTAG:='H';
        73: sTAG:='I';
        74: sTAG:='J';
        75: sTAG:='K';
        76: sTAG:='L';
        77: sTAG:='M';
        78: sTAG:='N';
        79: sTAG:='O';
        80: sTAG:='P';
        81: sTAG:='Q';
        82: sTAG:='R';
        83: sTAG:='S';
        84: sTAG:='T';
        85: sTAG:='U';
        86: sTAG:='V';
        87: sTAG:='W';
        88: sTAG:='X';
        89: sTAG:='Y';
        90: sTAG:='Z';
END_CASE;

```

Ilustración 61 - ASCII CONV ST

Cada uno de los grupos es independiente, y tiene un bloque funcional principal que es el llamado en el punto del programa que corresponda. Éste a su vez, para realizar su cometido, utiliza los demás bloques de su grupo y los propios de la biblioteca DBFL_SQL_V1_17.

Debido a que el funcionamiento de muchos bloques funcionales es similar se procede a describir uno y a indicar cuales son los bloques que funcionan de la misma forma.

5.8.2 Show Players

Este Bloque funcional es llamado en la etapa del programa principal MAIN llamada Reconf este bloque funcional realiza la función de mostrar los jugadores registrados en la base de datos para ello se inicia al cambiar el estado a 'true' de la variable *xHMI_SHOW_PLAYER* esta variable está vinculada a un interruptor en el SCADA el cual se describirá en un capitulo posterior.

Los bloques de color azul son bloques propios de la biblioteca *DBFL_SQL_V1_17* y funcionan a modo de "caja negra" por lo que la única interactuación que se puede realizar con ellos es vincularlos a las variables oportunas para realizar una conexión optima a la base de datos y una posterior decodificación de esos datos.

Los bloques de color verde son programas creados para la realización de la lectura de la base de datos los cuales se describen a continuación.

El bloque rojo corresponde a la función propia del PC Worx y del estándar IEC 61131 R_TRIG cuya función es poner la salida del bloque a uno durante un ciclo de scan cuando detecte un flanco positivo a la entrada.

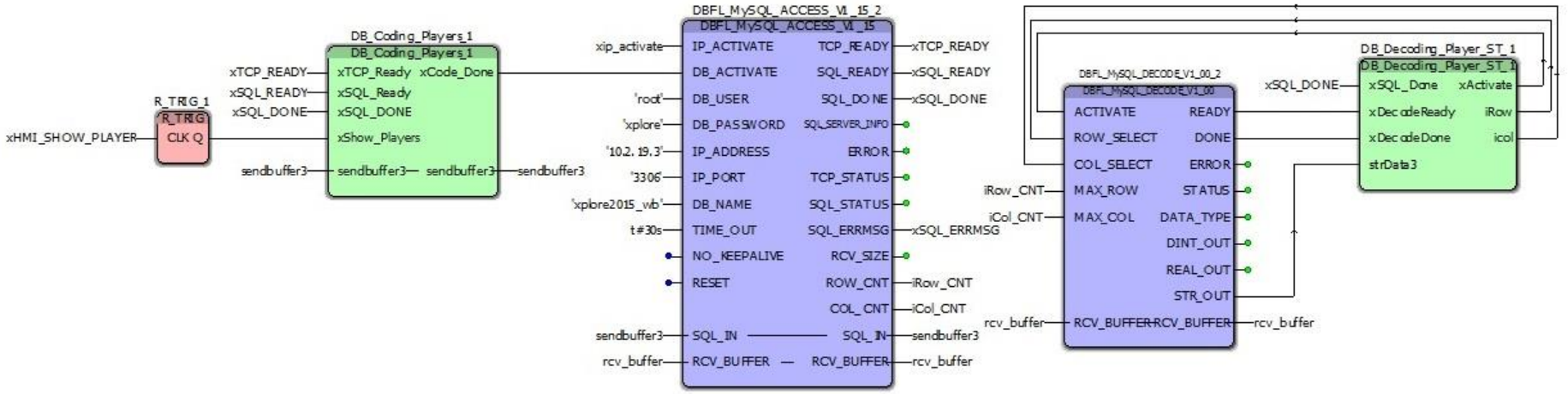


Ilustración 62 - Bloque de funciones Show Players

5.8.3 DB_Coding_Players_1

Bloque funcional escrito en FBD, que crea la instrucción a enviar a la base de datos

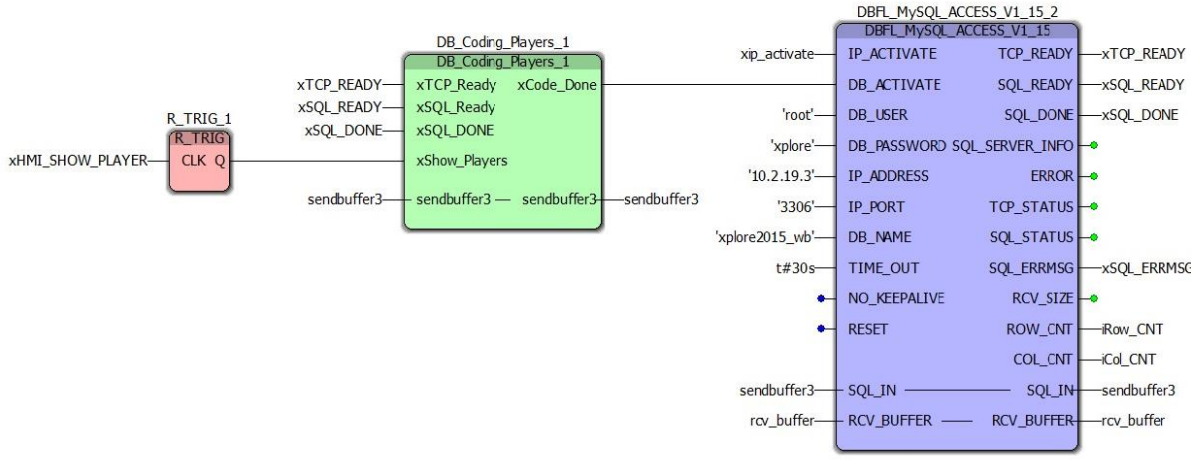


Ilustración 63 - Bloque de funciones DB_Coding_Players_1

Variables de Entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xTCP_READY	BOOL	Indica que la conexión ha sido establecida con la base de datos (Salida del bloque de conexión)
xSQL_Ready	BOOL	Indica que la base de datos está preparada para recibir comando SQL (Salida del bloque de conexión)
xSQL_Done	BOOL	Indica que el comando SQL ha sido ejecutado (Salida del bloque de conexión)
xShow_Players	BOOL	Entrada para activar el bloque por flanco de variable

Tabla 19 - Variables Entrada Coding_Players_1

Variables de Salida (VAR_OUTPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xCode_Done	BOOL	Variable que activa el envio del comando de SQL

Tabla 20 - Variables Salida Coding_Players_1

Variables de entrada/salida (VAR_IN_OUT)

NOMBRE	TIPO	DESCRIPCIÓN
xSend_Buffer3	DBFL_ARR_BYTE_0_1439	Comando SQL en array de bytes

Tabla 21 - Variables Entrada/Salida Coding_Players 1

Dentro de este bloque funcional se utilizan varios bloques, en uno de ellos se realiza la programación precisa en texto estructurado que se describirá más adelante y otros dos bloques propios de la librería *DBFL_SQL_V1_17* para codificar la información y enviarla a la base de datos.

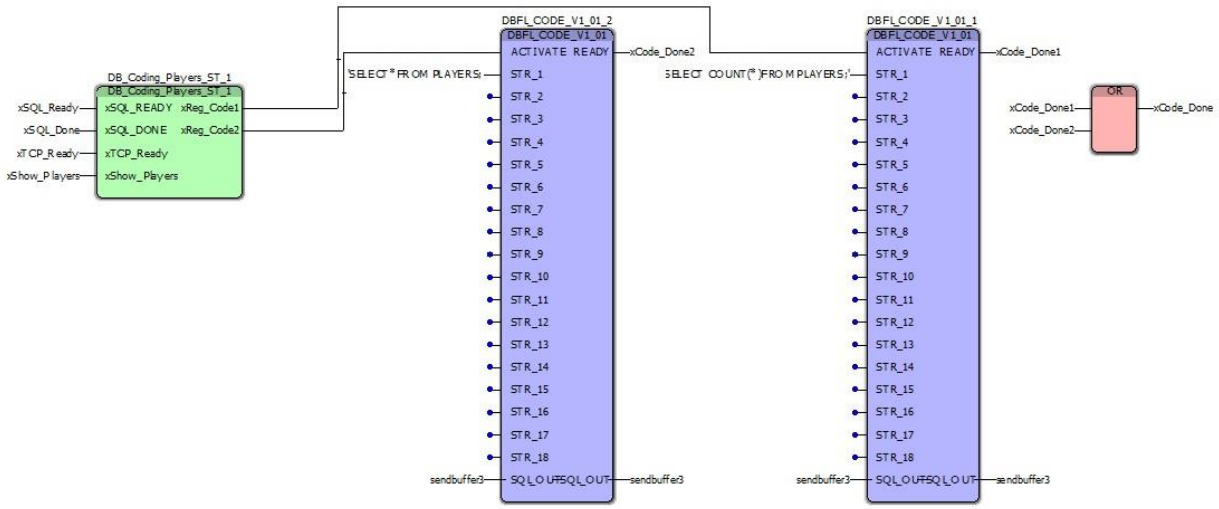


Ilustración 64 - DB_Coding_Player

Este bloque manda dos instrucciones a la base de datos sincronizadas, debido a que el número de filas en la base de datos serán siempre desconocidas y pueden variar en función de un aumento o reducción de los registros de la base de datos. Por lo que lo primero que se ha de hacer es consultar el número de filas que tiene la tabla para una posterior decodificación de los registros.

Los comandos a enviar a la base de datos son los siguientes:

SELECT COUNT(*) FROM PLAYERS, para contar el número de filas de la tabla Players.

SELECT*FROM PLAYERS, para mostrar la información de las tablas.

Estos comandos en función de la tabla que se quiere consultar, si se quiere cambiar la tabla a consultar basta con cambiar PLAYERS por la tabla a consultar

5.8.4 DB_Coding_Players_ST

Ilustración 65 - Bloque Funcional DB_Coding_Players

Bloque funcional escrito en ST, que activa los bloques de decodificación en el orden preciso para mostrar la información alojada en las tablas de la base de datos.

Variables de Entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xTCP_READY	BOOL	Indica que la conexión ha sido establecida con la base de datos (Salida del bloque de conexión)
xSQL_READY	BOOL	Indica que la base de datos está preparada para recibir comando SQL (Salida del bloque de

xSQL_DONE	BOOL	Indica que el comando SQL ha sido ejecutado (Salida del bloque de conexión)
xShow_Players	BOOL	Entrada para activar el bloque por flanco de variable

Tabla 22 - Variables Entrada Coding_Players_ST

Variables de Salida (VAR_OUTPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xReg_Code1	BOOL	Ejecuta la activación del primer bloque, para la codificación de la instrucción de la base de datos
xReg_Code2	BOOL	Ejecuta la activación del segundo bloque, para la codificación de la instrucción de la base de datos.

Tabla 23 - Variables Salida Coding_Players_ST

5.8.5 DB_DecodePlayers_ST

Bloque funcional escrito en ST, que guarda la información leída de la base de datos en las variables adecuadas del programa, que son las asociadas al HMI.

Después de enviar el comando, la base de datos devuelve una información que hay que convertir a un formato legible por el PCWorx, en este caso, de tipo string. Esto lo hace el bloque DBFL_MySQL_DECODE_V1_00, y paralelamente el DB_DecodePlayers_ST guarda esa información en variables del programa. Por eso con cada nuevo dato de la tabla deberán activarse ambos, lo que requiere esa realimentación con la que se avisan uno a otro sobre cuándo empiezan y terminan.

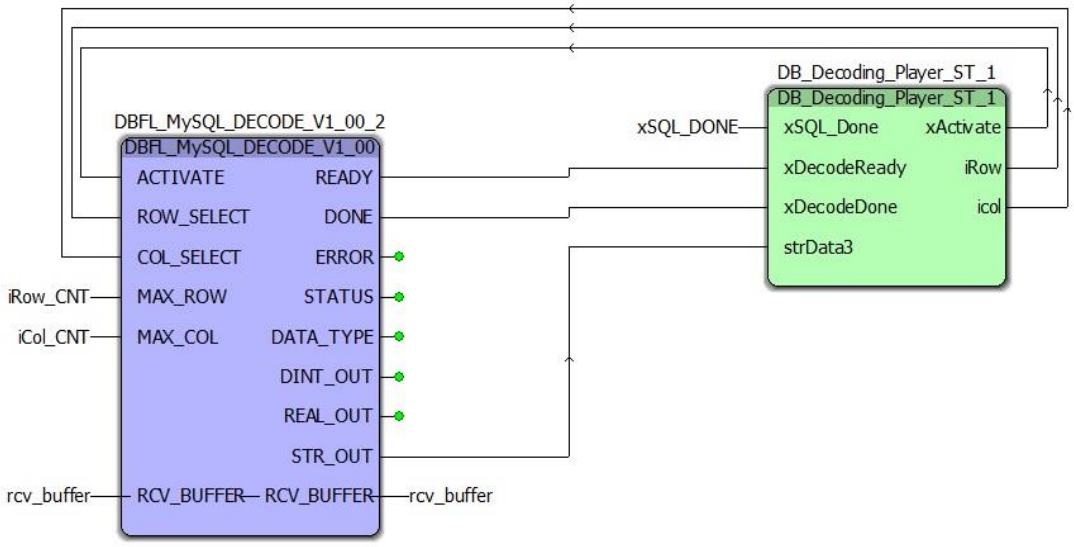


Ilustración 66 - Bloques asociados a la decodificación de la base de datos

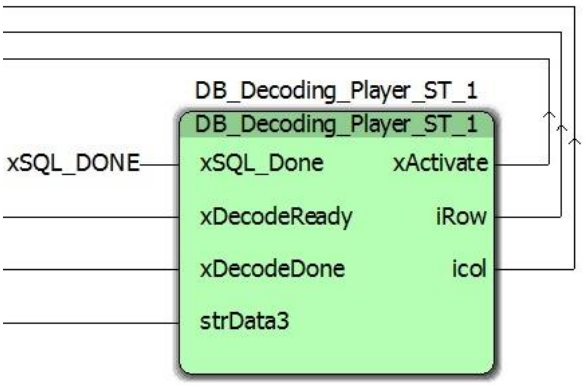


Ilustración 67 - Bloque Funcional Decoding_Player_S

Variables de Entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xDecodeReady	BOOL	Indica que el bloque DBFL_MySQL_DECODE_V1_00 está listo para comenzar (Salida del bloque decode)
xDecodeDone	BOOL	Indica que el bloque DBFL_MySQL_DECODE_V1_00 ha terminado (Salida del bloque decode)
strData3	BOOL	Información de la base de datos en formato string (Salida del bloque decode)
xSQL_Done	BOOL	Indica que el comando SQL ha sido ejecutado (Salida del bloque de conexión)

Tabla 24 - Variables Entrada Decoding_Player_ST

Variables de Salida (VAR_OUTPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xActivate	BOOL	Variable que activa el bloque DBFL_MySQL_DECODE_V1_00_1
iRow	BOOL	Variable que indica la fila a leer en ese momento (Entrada al bloque DBFL_MySQL_DECODE_V1_00_1)
iCol	BOOL	Variables que indica la columna a leer en ese momento (Entrada al bloque DBFL_MySQL_DECODE_V1_00_1)

Tabla 25 - Variables Entrada Decoding_Player_ST

Todos los bloques funcionales que muestran la información almacenada en las tablas de las bases de datos funcionan de la misma manera.

5.8.6 RFID_Authentification

Programa desarrollado en FBD cuya función principal es la comprobación del identificador interno del TAG del tipo tarjeta con los usuarios ya registrados en la base de datos.

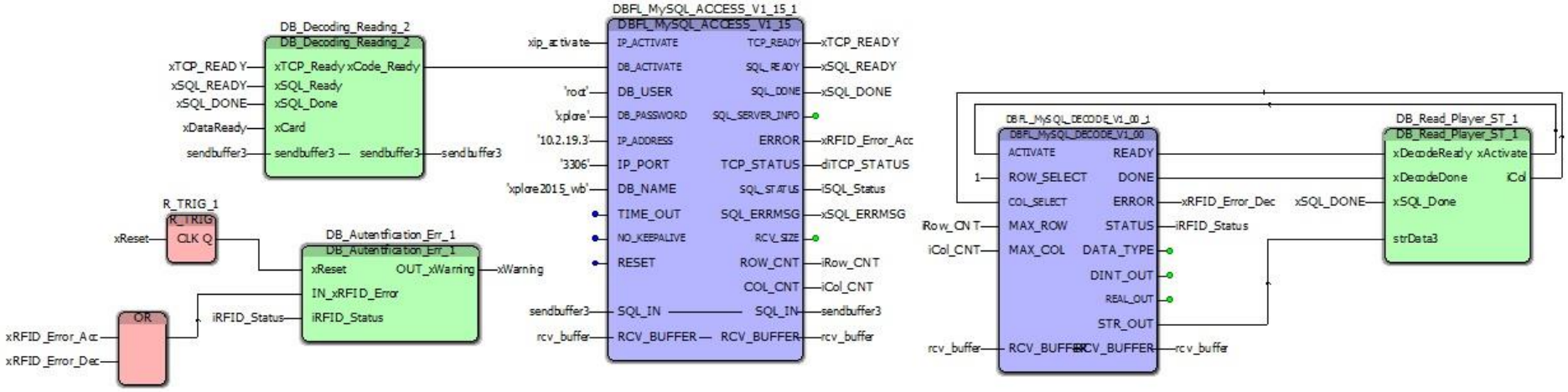


Ilustración 68 - Bloque de funciones principal del POU RFID Authentifi

5.8.7 DB_Authentification_Error

Este programa consta de una parte que indica si ha habido un error al pasar una tarjeta por el lector, en caso de que se intente autentificar un TAG no registrado en la base de datos y el lector sea capaz de leerlo si activará un warning bloqueando la base de datos e indicando el tipo de error a través de la variable iRFID_Status que identifica el error en función del número que muestra la variable

iRFID_Status	Error
0	No hay error
1	Error en las columnas
2	Error en las filas
3	Error en el tipo de dato introducido
4	Se intenta acceder a la posición 0
5	Dato no valido

Ilustración 69 - Tabla de Errores

Cuando el warning se produzca en función de número que tome la variable iRFID_Status grabará en la base de datos el tipo de error producido, pudiendo verse al mostrar la tabla warnings de la basew de datos y quedando el sistema bloquedo.

Para desbloquear el sistema es necesario poner a uno la variable xReset, y pasar un TAG valido en ese instante el sistema de desbloquea.

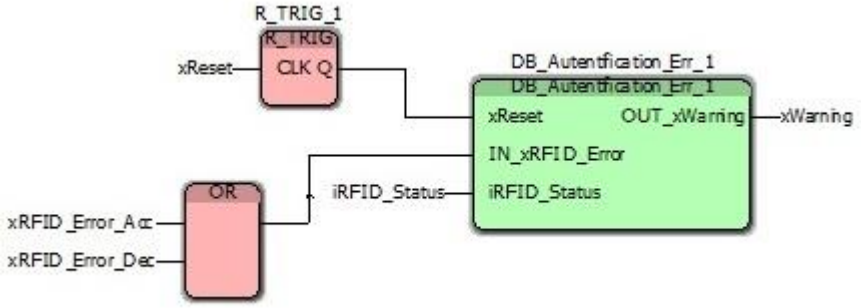


Ilustración 70 - Parte del error y rearme

Variables de Entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xReset	BOOL	Variable que resetea el sistema tras un warning
IN_xRFID_Error	BOOL	Variable que indica la presencia de un error
iRFID_Status	INT	Variables que indica el tipo de error cometido

Tabla 26 - Variables Entrada Authentication_Err

Variables de Salida (VAR_OUTPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xWarning	BOOL	Variable que activa los procesos asociados a los warnings

Tabla 27 - Variables Salida Authentication_Err

5.8.8 DB_Decoding_Reading_2

FBD que activa la decodificación del TAG para posteriormente comparar el identificador interno de la tarjeta con la columna id_Player de la tabla players.

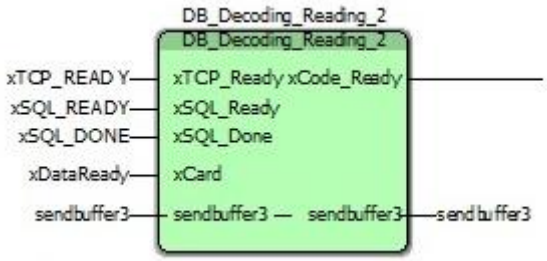


Ilustración 71 - Bloque funcional DB_Decoding_Reading_2

Variables de Entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xTCP_READY	BOOL	Indica que la conexión ha sido establecida con la base de datos (Salida del bloque de conexión)
xSQL_READY	BOOL	Indica que la base de datos está preparada para recibir comando SQL (Salida del bloque de
xSQL_DONE	BOOL	Indica que el comando SQL ha sido ejecutado (Salida del bloque de conexión)
xCard	BOOL	Entrada para activar la decodificación del TAG

Tabla 28 - Variables Entrada Decoding_Reading

Variables de Salida (VAR_OUTPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xCode_Ready	BOOL	Variable que activa el envio del comando de SQL

Tabla 29 - Variables Salida Decoding_Reading

Variables de entrada/salida (VAR_IN_OUT)

NOMBRE	TIPO	DESCRIPCIÓN
xSend_Buffer3	DBFL_ARR_BYTE_0_1439	Comando SQL en array de bytes

Tabla 30 - Variables Entrada/Salida Decoding_Reading

5.8.9 DB_Decoding_Reading_ST

Dentro de este bloque funcional se utilizan varios bloques, en uno de ellos se realiza la programación precisa en texto estructurado que se describirá más adelante y el otro bloque propios de la librería *DBFL_SQL_V1_17* para codificar la información y enviarla a la base de datos.

El bloque realiza una búsqueda en la base de datos, buscando un TAG que coincida con el número almacenado en la variable STAG

El comando que se manda a la base de datos es:

SELECT*FROM PLAYER WHERE ID_PLAYER=sTAG;

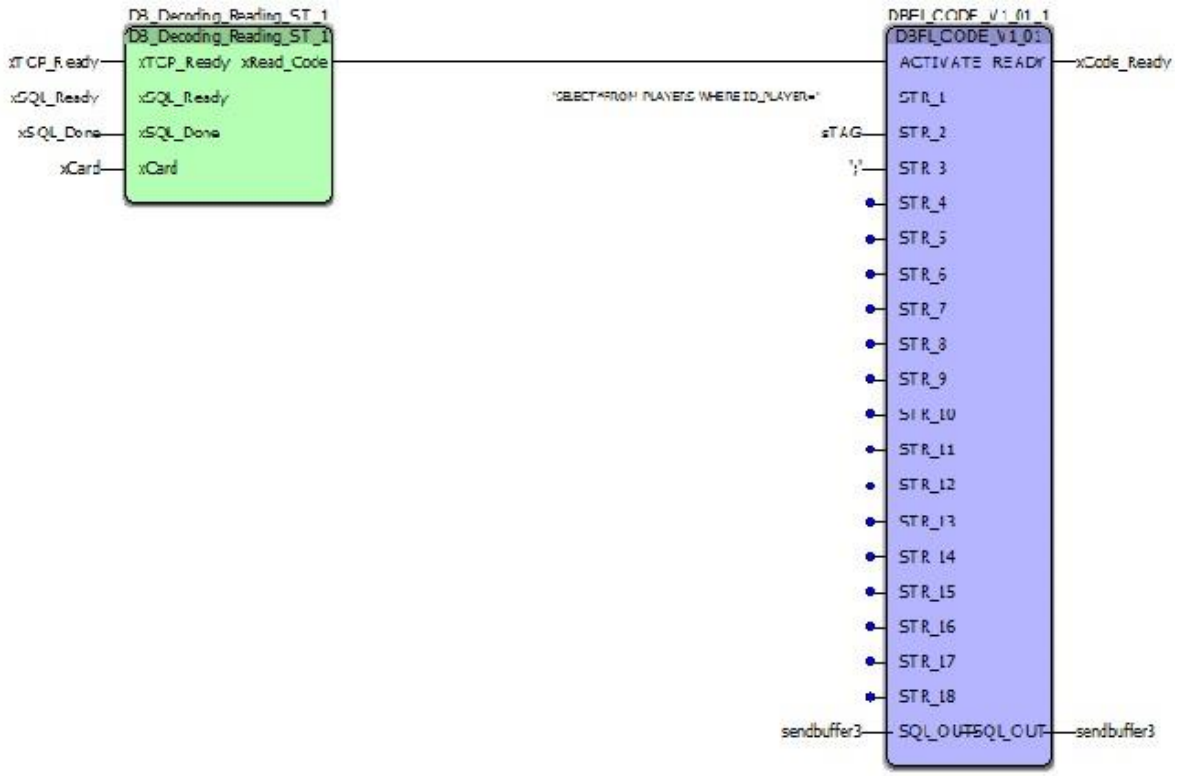


Ilustración 72 - Bloque decoding_Reading

5.8.10 DB_Read_Player_ST_2

Parte del programa que decodifica la información recibida de la base de datos, en este caso en particular solo se tiene que recorrer una fila , concretamente la fila que contenga el id_Player que coincida con el id interno de la tarjeta por lo que la fila será fija para la decodificación y variará solo la columna.

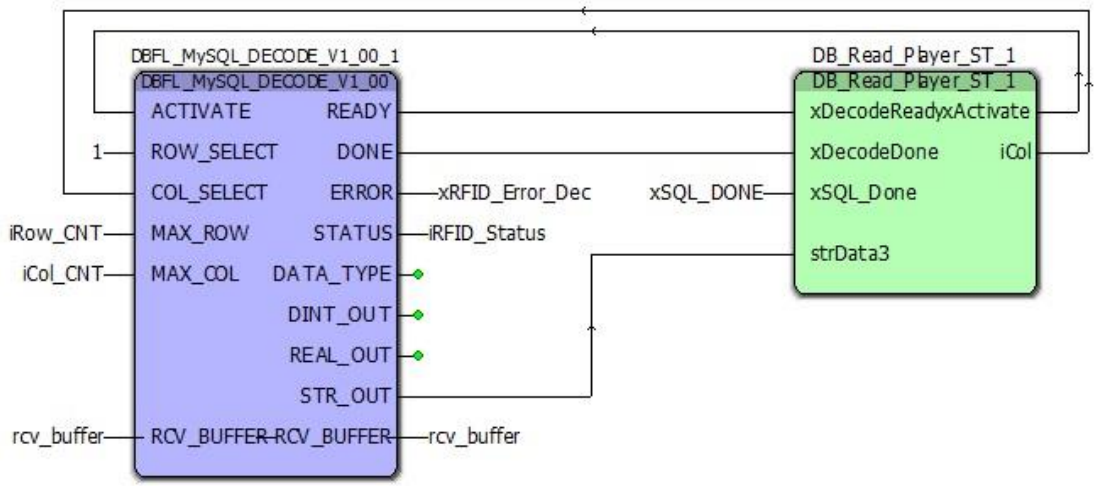


Ilustración 73 - Decodificación de la base de dato

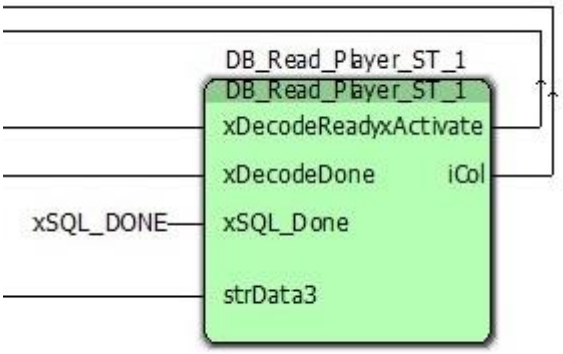


Ilustración 74 - DB_Read_Player_ST_1

Variables de Entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xDecodeReady	BOOL	Indica que el bloque DBFL_MySQL_DECODE_V1_00 está listo para comenzar (Salida del bloque decode)
xDecodeDone	BOOL	Indica que el bloque DBFL_MySQL_DECODE_V1_00 ha terminado (Salida del bloque decode)

strData3	BOOL	Información de la base de datos en formato string (Salida del bloque decode)
xSQL_Done	BOOL	Indica que el comando SQL ha sido ejecutado (Salida del bloque de conexión)

Tabla 31 - Variables Entrada Read_Player_ST

Variables de Salida (VAR_OUTPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xActivate	BOOL	Variable que activa el bloque DBFL_MySQL_DECODE_V1_00_1
iCol	BOOL	Variables que indica la columna a leer en ese momento (Entrada al bloque DBFL_MySQL_DECODE_V1_00_1)

Tabla 32 - Variables Salida Read_Player_ST

5.8.11 DB_Register_Players

Programa desarrollado en FBD cuya función es guardar en la base de datos la información del TAG y la fecha y hora que se ha producido el registro, este programa solo escribe en la base de datos por lo que no tendrá la parte de decodificación de datos.

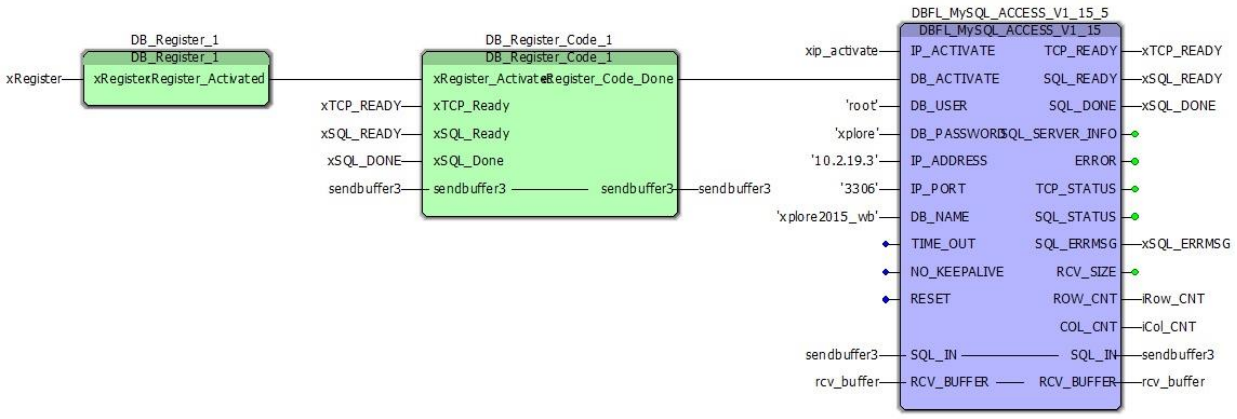


Ilustración 75- Programa para codificar información

5.8.12 DB_Register_1

Programa que ejecuta mediante flanco ascendente (R_TRIG) la orden de escritura es necesario ejecutar la orden de escritura de esta manera debido a que en caso contrario se estaría escribiendo constantemente en la base de datos, lo que provocaría un bucle infinito de escritura.

Ilustración 76 - DB_Register_1

Variables de entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xRegister	BOOL	Activa la orden de escritura

Tabla 33 - Variables Entrada Register_1

Variables de Salida (VAR_OUTPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xRegister_Activated	BOOL	Ejecuta la orden de escritura y se pone a 0

Tabla 34 - Variables Salida Register_1

5.8.13 DB_Register_Code_1

Programa que ejecuta la orden de escritura en la base de datos, en este bloque se almacena la información necesaria para almacenar los datos del TAG en la base de datos.

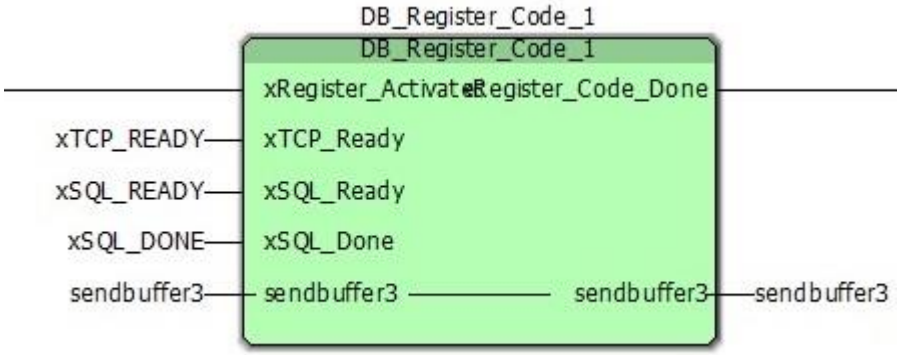


Ilustración 77 – Bloque que codifica la información

Variables de entrada (VAR_INPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xTCP_READY	BOOL	Indica que la conexión ha sido establecida con
xSQL_READY	BOOL	Indica que la base de datos está preparada
xSQL_DONE	BOOL	Indica que el comando SQL ha sido ejecutado
xRegister_Activate	BOOL	Entrada para activar el bloque

Tabla 35 - Variables Entrada Register Code

Variables de Salida (VAR_OUTPUT)

NOMBRE	TIPO	DESCRIPCIÓN
xRegister_Code_Done	BOOL	Variable que activa el envio del comando de SQL

Tabla 36 - Variables Salida Register Code

Variables de entrada/salida (VAR_IN_OUT)

NOMBRE	TIPO	DESCRIPCIÓN
xSend_Buffer3	DBFL_ARR_BYTE_0_1439	Comando SQL en array de bytes

Tabla 37 - Variables Entrada/Salida Register Code

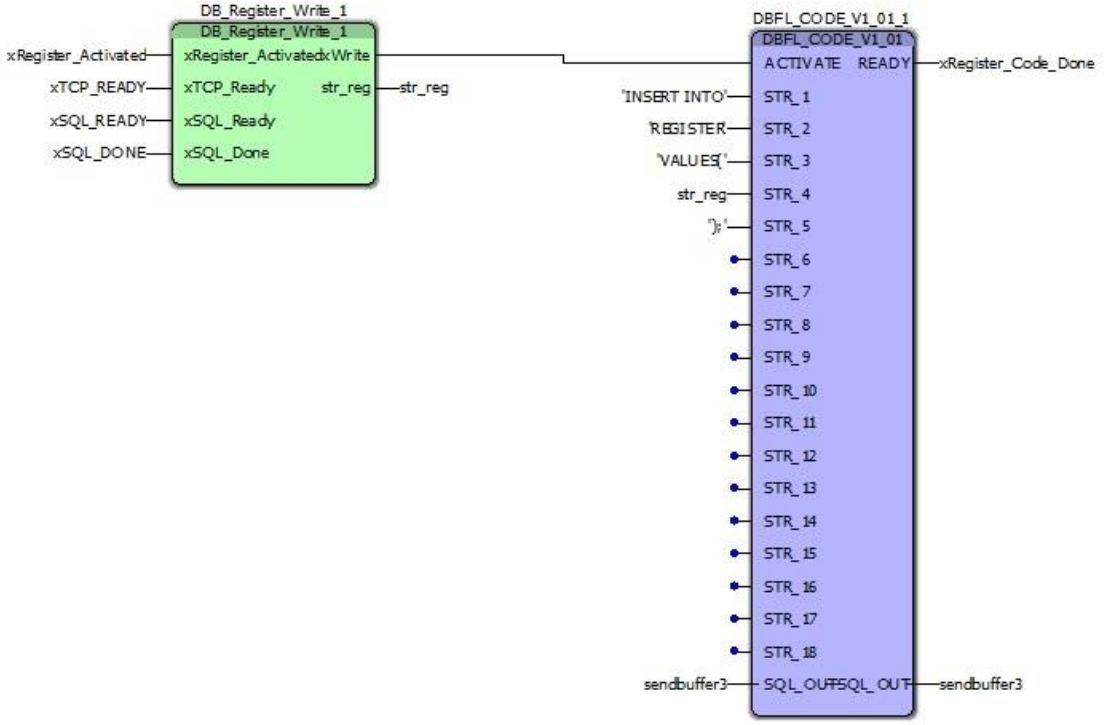


Ilustración 78 - DB_Register_Cod

Register_Activated es una variable global que se activa al autentificarse, una vez que alguien se ha autentificado se procede a la captura de los datos del usuario. Si la base de datos está preparada y conectada se procede a escribir toda esta información en la tabla registro.

El comando que se manda a la base de datos es:

INSERT INTO REGISTER VALUES (str_reg);

Siendo str_reg una variable de tipo String en la que se concatenan las variables de tipo String *id_Player*, *name*, *surname*, y fecha y hora, para guárdalo en la base de datos.

5.8.14 DB_Warnings

Programa desarrollado en FBD para codificar las alarmas (warnings) en la base de datos el sistema de codificación es prácticamente igual que el descrito en el anterior apartado, con la excepción que en este programa debido a que codifica los warnings no es necesario lanzar la variable que ejecuta el programa mediante un R_TRIG, al detectar un warning la base de datos se detiene y es necesario un rearme para volver a tener la base de datos y el programa principal operativo nuevamente.

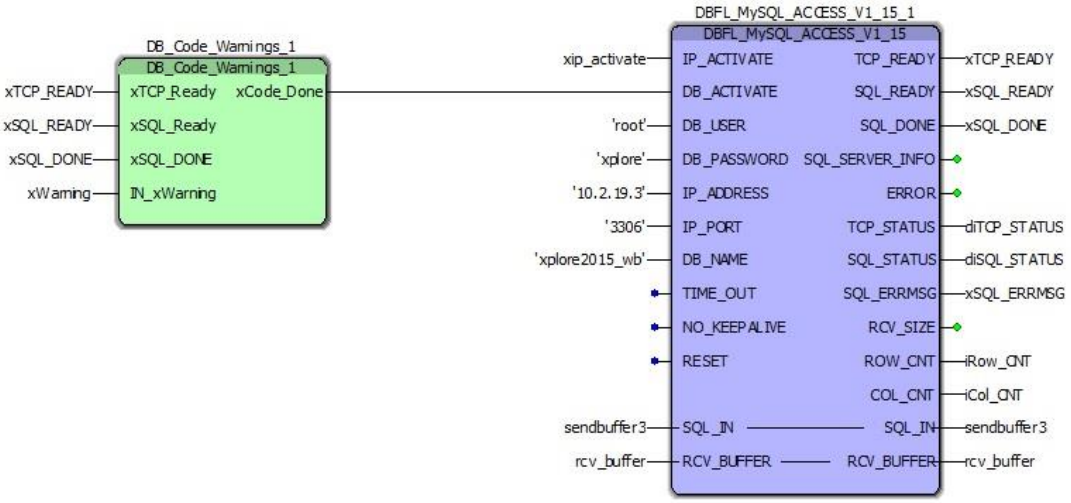


Ilustración 79 - Bloque de Code_Warning

Si se detecta un error se carga en un string el identificador del warning, la fecha y el tipo de error, en este caso se debe de hacer especial incapie en que se debe distinguir entre dos tipos de error, error de decodificación y error de acceso, en el primer caso no hay ningún problema en mandar el error directamente, la base de datos, pero en el segundo caso el acceso a la base queda bloqueada hasta que no haya un rearme el sistema, por lo se almacena el error en la base de datos una vez rearmada la base de datos.

Dentro de este bloque además de codificar la instrucción para enviar a la base de datos se encienden las luces rojas de los leds, y el flash azul de la pista para indicar que se ha producido un warning, al rearmar el sistema las variables que encienden las luces se pondrán a false.

5.8.15 DB_Deduct_Credits

Programa desarrollado en FDB para actualizar los créditos del sistema, necesarios para poder interactuar con él, este bloque consta de dos partes, una primera parte que descuenta créditos en función de un tiempo T establecido desde el HMI y la otra parte que cambia el valor de los créditos por otro establecido en la base de datos.

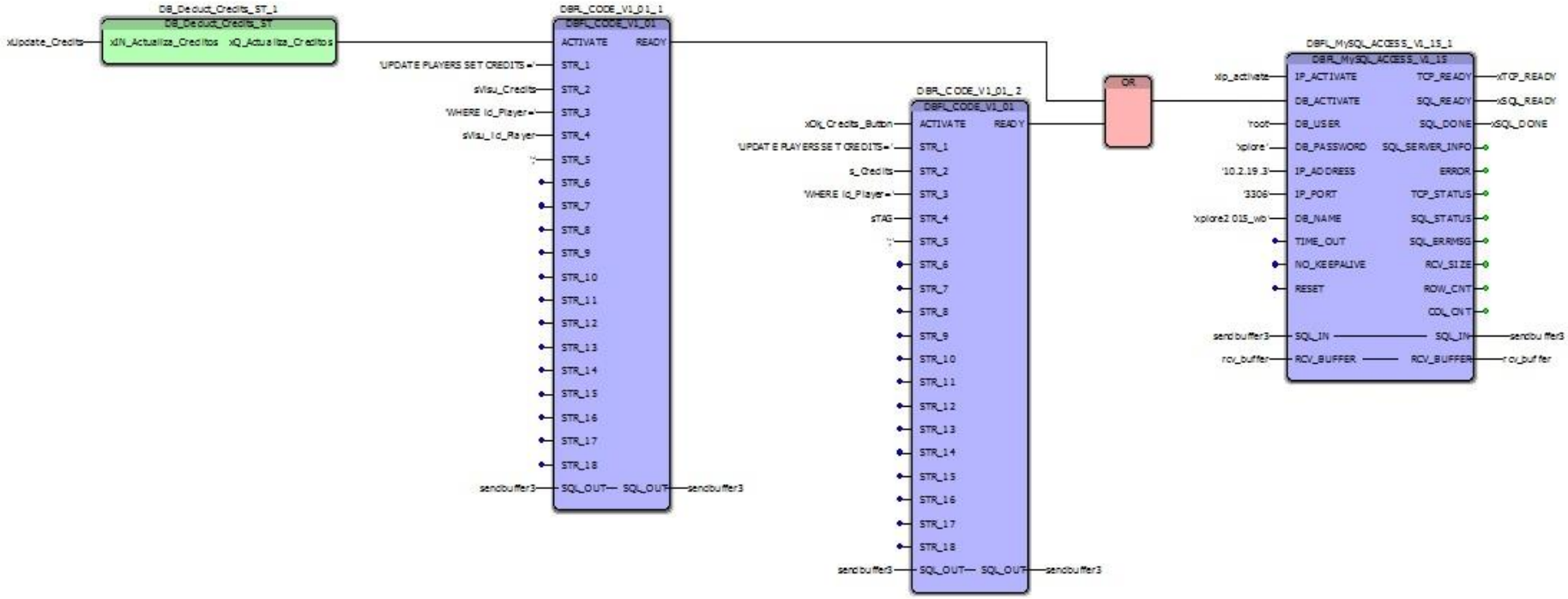


Ilustración 80 - Bloque que actualiza lo créditos

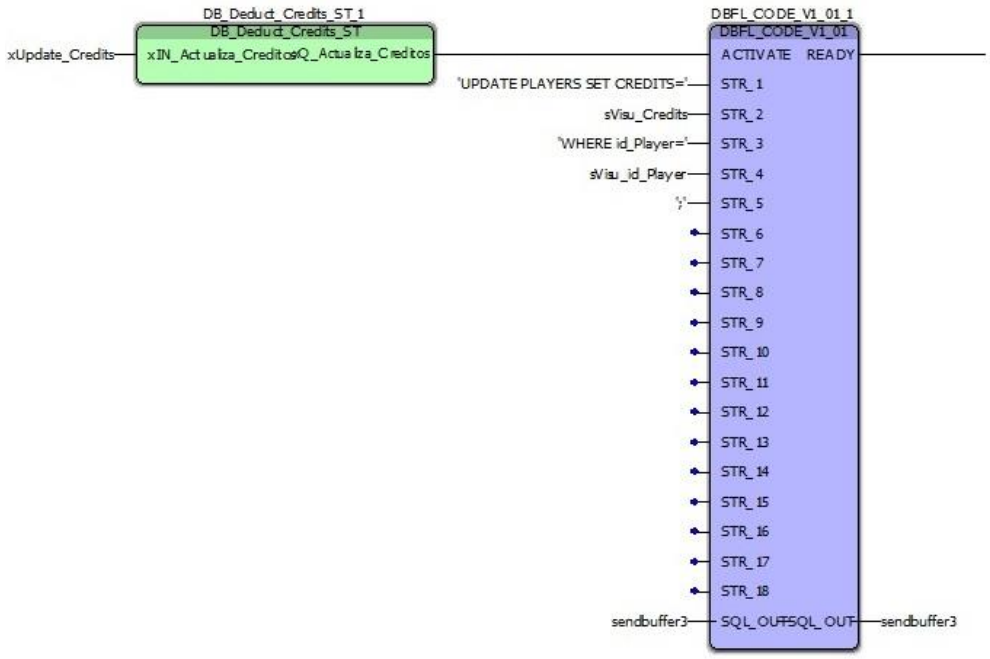


Ilustración 81 - Descuenta de Créditos

La instrucción a enviar a la base de datos es la siguiente:

UPDATE PLAYERS SET CREDITS=sVisu_Credits WHERE id_Player= sVisu_id_Player;

Siendo sVisu_Credits una variable asociada al HMI, una vez dada la orden se actualiza la base de datos. La actualización se créditos se produce desde el HMI, mediante una detección de flanco para evitar bucles infinitos.

La parte que descuenta créditos se actualiza cuando la variable *xOk_Credits_Button* esta variable se activa desde el HMI y activará el bloque.

La instrucción a mandar a la base de datos es la siguiente:

UPDATE PLAYERS SET CREDITS s_Credits WHERE id_Player=sTAG;

Siendo sTAG el identificador del TAG leido por el lector RFID.

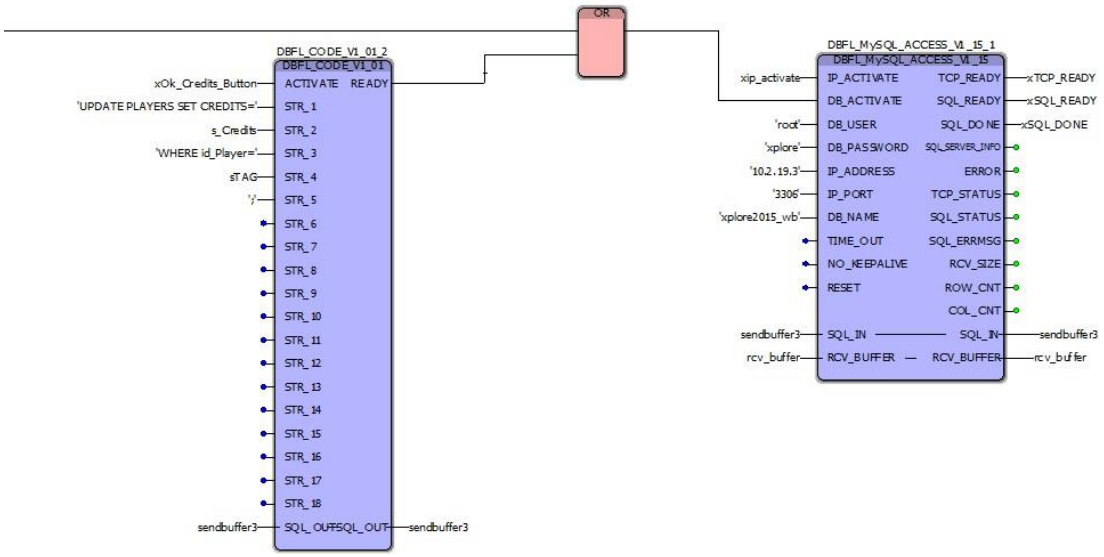


Ilustración 82 - Parte del programa que actualiza créditos

5.8.16 DB_Insert_

Estos programas desarrollados en lenguaje FDB han sido realizados para insertar nuevos registros en la base de datos los programas para insertar registros en Players y Cars son prácticamente idénticos por lo que solo se describirá uno de ellos.

Este bloque se activa desde el SCADA a través de la variable *xHMI_Update_DB* que está asociada a un botón.

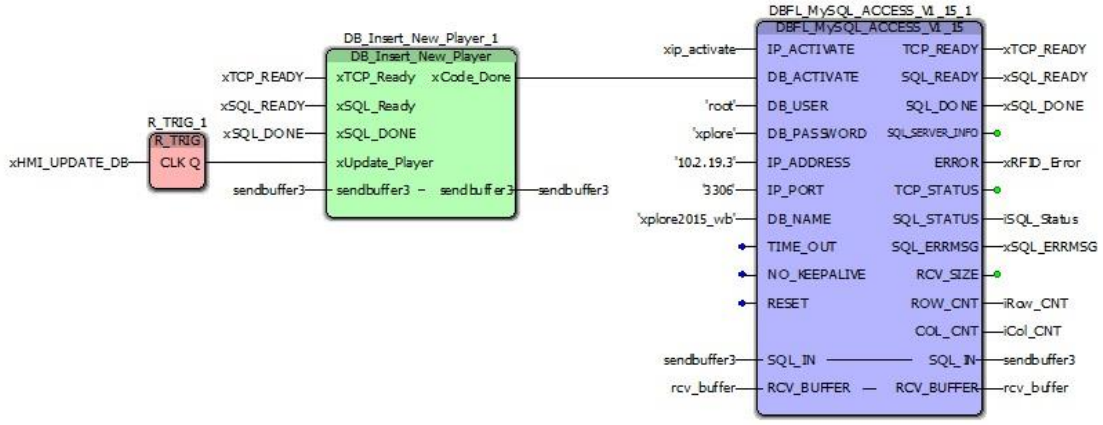


Ilustración 83 – Insert Cars

En la parte interna del bloque DB_Insert_New_Player_ST, tiene una estructura similar a todos los bloques de codificación.

La variable *str_Update* es una variable de tipo string que concatena todos los parámetros que se cargan a través del SCADA. Posteriormente cuando se active el bloque a través de la variable *xWrite* y con la instrucción de SQL precargada en el bloque CODE.

La instrucción final que se envía a la base de datos quedaría:

INSERT INTO PLAYERS VALUES (str_Update)

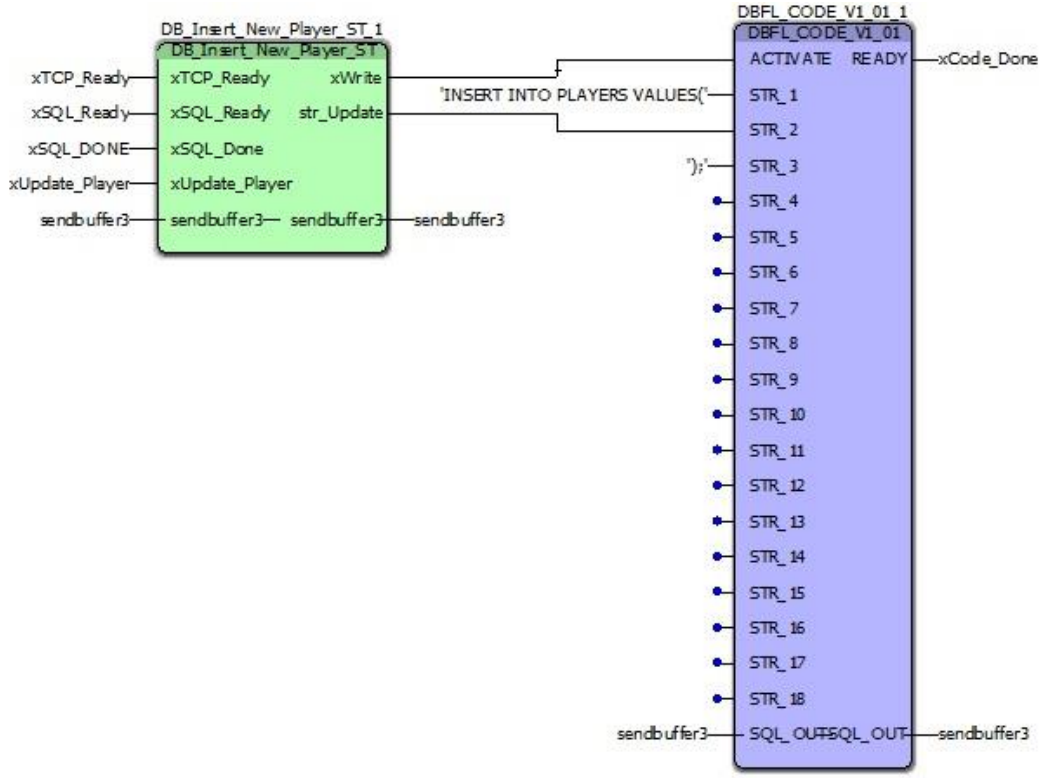


Ilustración 84 – Code New Player

```
(*Si quiero actualizar la tabla de coches con un nuevo valor *)
if xUpdate_Player then
sNew_id :=sTAG;
sNew_Nick:=sHMI_New_Nick;
sNew_Name:=sHMI_New_Name;
sNew_Surname:=sHMI_New_Surname;
sNew_Email:=sHMI_New_Email;
sNew_Card:=sASCII;
sHMI_New_Credits:=INT_TO_STRING(iHMI_New_Credits,'%03d');
sNew_Credits:=sHMI_New_Credits;

str_Update:= CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(
CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(CONCAT(
CONCAT(CONCAT('"',sTAG),'"'),','),'"'),sNew_Nick),'"'),','),'"'),sNew_Name),'"'),','),'"'),
sNew_Surname),'"'),','),'"'),sNew_Email),'"'),','),'"'),sNew_Card),'"'),','),'"'),sNew_Credits),'"');
end_if;

if xTCP_Ready and xSQL_Ready and xUpdate_Player then
xWrite:=true;
xUpdate_Player:=false;
xflag:=true;
end_if;

if xSQL_Done then
xWrite:=false;
xUpdate_Player_Table:=false;
end_if;
```

Ilustración 85 - DB_Insert_New_Player

Es necesario mencionar que en la parte de Cars se ha tenido que partir la concatenación en dos partes debido al número de columnas que tiene la tabla Cars.

5.8.17 DB_Delete_

Estos programas desarrollados en lenguaje FDB han sido realizados para eliminar registros en la base de datos, los programas para eliminar registros en Players y Cars son prácticamente idénticos por lo que solo se describirá uno de ellos.

Este bloque se activa desde el SCADA a través de la variable *xUpdate_Cars_Table* que está asociada a un botón.

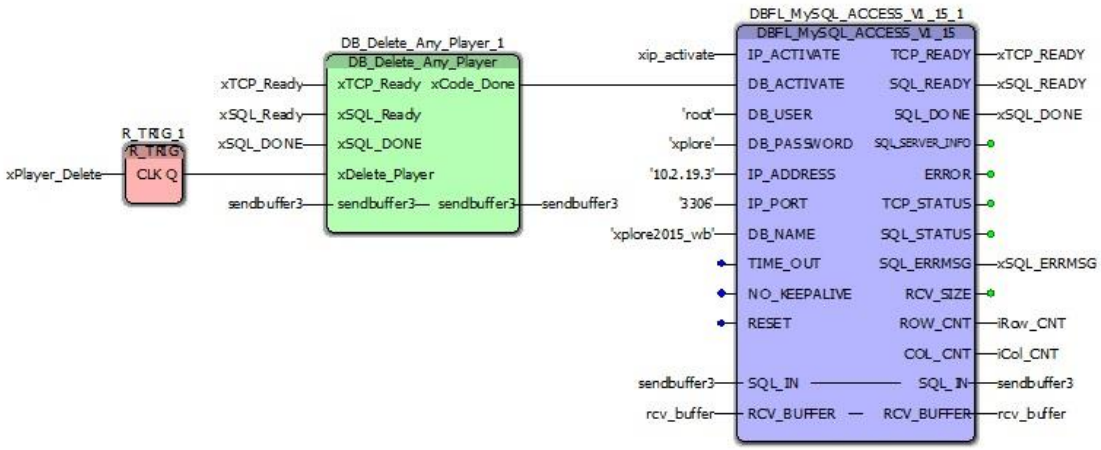


Ilustración 86 - Delete_Player

Para eliminar un registro se busca en la base de datos la fila que coincida con el identificador que se quiera eliminar, que cargara desde el SCADA, posteriormente se guarda en la variable *str_player* y se añade a la instrucción de SQL precargada en el bloque CODE.

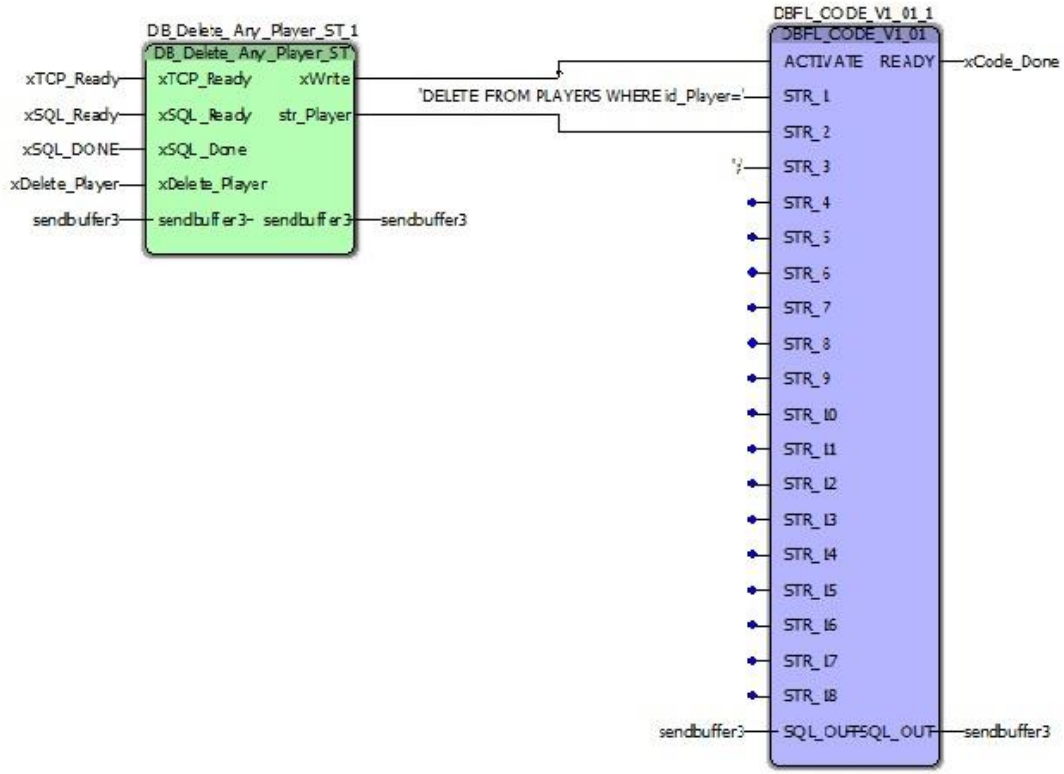


Ilustración 87 - Delete_Any_Player

```
if xDelete_Player then
sid_Player:=sTAG;
(*sid_Player sera asignada a una variable del HMI*)
str_Player:=CONCAT(CONCAT('"',sid_Player),'"');
xDelete:=true;
end_if;

if xTCP_Ready and xSQL_Ready and xDelete then
xWrite:=true;
xDelete:=false;
end_if;

if xSQL_Done then
xWrite:=false;
xPlayer_Delete:=false;
xDelete_Player:=false;
xflag:=true;
end_if;
```

Ilustración 88 – DB_Delete_Any_Player

5.8.18 DB_Check

Programa desarrollado en lenguaje FDB realizado para comprobar las filas de las tablas de la base de datos, con este código se intenta comprobar que no haya una fila ya existente y se pueda bloquear la base de datos.

Este bloque se activa desde el SCADA a través de la variable *xHMI_Visu_Check* que está asociada a un botón.

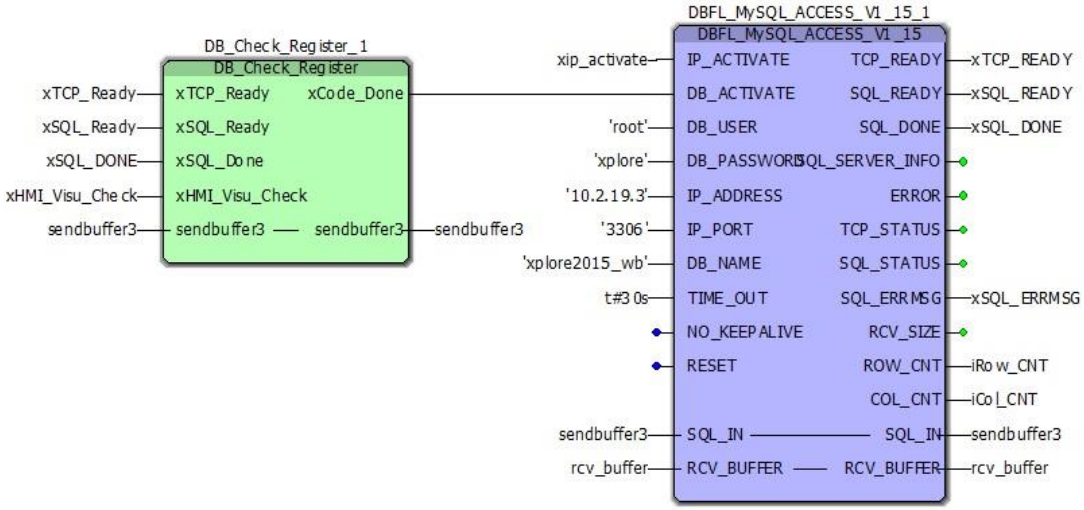


Ilustración 89 - Parte de acceso a la base de datos del programa

El procedimiento de acceso a la base de datos es similar al utilizado para mostrar las tablas de Cars, Players, Register y Warnings. Primero se cuentan las filas de la tabla y luego se recorre hasta el número máximo de filas contado previamente.

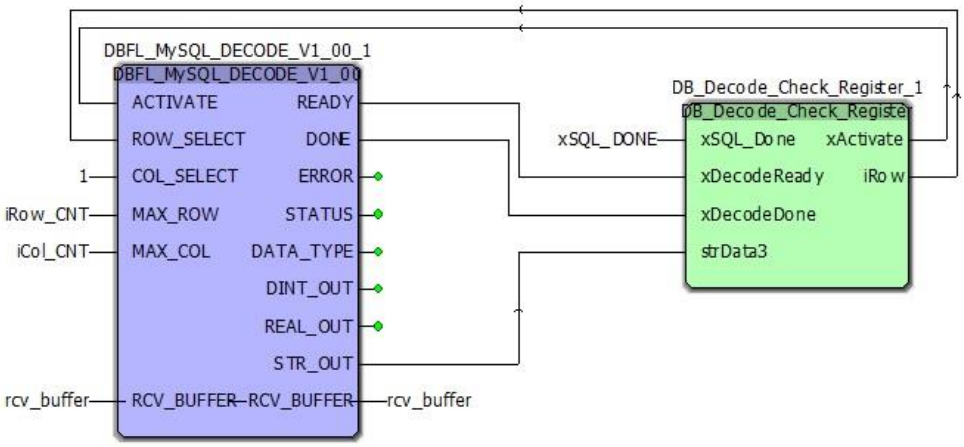


Ilustración 90 - Parte de Decodificación de la información de la base de datos

En esta parte se decodifica la información, debido a que solo se busca coincidencia con algún identificador ya registrado en la base de datos, solo recorrerá una columna concretamente la columna donde están los identificadores.

6. Aplicación SCADA

6.1. Introducción

Los programas necesarios, y en su caso el hardware adicional, se denominan en general sistemas SCADA (Supervisory Control And Data Acquisition). Mediante dichas aplicaciones es posible el acceso a datos remotos de un proceso y, utilizando las herramientas de comunicación necesarias en cada caso, el control del mismo. Atendiendo a dicha definición se observa que no se tratan de sistemas de control, sino de utilidades software de monitorización o supervisión, que toman el rol de interfase entre los niveles de control (PLC) y los de gestión, a un nivel superior.

Como se ha mencionado, los sistemas SCADA se conciben como herramientas de supervisión y mando. Entre sus principales prestaciones es posible destacar:

- Monitorización y representación de datos en tiempo real.
- Supervisión de datos de proceso y herramienta de gestión para la toma de decisiones (mantenimiento predictivo).
- Control. Posibilidad de que el operador pueda cambiar consignas u otros datos claves para el proceso directamente desde la pantalla de explotación.
- Adquisición de datos, con el fin de recoger y procesar información acerca de la evolución del proceso.
- Visualización de alarmas y eventos. Reconocimiento de eventos excepcionales acaecidos en la planta para su inmediata puesta en conocimiento de los operarios con el fin de efectuar las acciones correctoras pertinentes.
- Ejecución de acciones o “recetas” para la ejecución de programas preestablecidos.
- Seguridad de datos. Con el fin de proteger las tareas de envío y recepción de datos de influencias no deseadas, intencionadas o no (fallos en la programación, intrusos u otras causas)
- Seguridad en los accesos. Restringiendo zonas de programa comprometidas a usuarios no autorizados, restringiendo todos los accesos y acciones llevadas a cabo por cualquier operador.
- Programación numérica. Permite realizar cálculos aritméticos de elevada resolución sobre la CPU del ordenador mediante el uso de lenguajes de alto nivel.

En un sistema SCADA existe algo mas que pantallas de explotación que informa acerca del estad de la infraestructura. Tras éstas se encuentran multitud de elementos de regulación y control, sistemas de comunicaciones y múltiples unidades de software cuyo objetivo es que el sistema funcione de forma eficiente y segura. Las ventajas más evidentes de los sistemas de control automatizado y supervisado (SCADA) pueden enumerarse según:

- Creación de aplicaciones funcionales sin la necesidad de ser un experto en la materia.
- Rápida localización de errores y fallos de funcionamiento gracias a la presencia de herramientas de diagnóstico. Como consecuencia se favorece una disminución en los periodos de paro en las instalaciones repercutiendo en una disminución de los costes de mantenimiento.
- El concepto de telemantenimiento permite realizar modificaciones de software en las estaciones remotas desde el centro de control. Además, la tecnología Web permite el acceso al sistema de control desde cualquier punto geográfico.
- Presencia en los programas de visualización de todo tipo de ayudas al usuario, desde la aparición de alarmas hasta la localización de la causa de las mismas.
- Mediante el uso de tecnologías celulares (GSM, GPRS), los sistemas de control pueden mantener informados a los operadores sobre cualquier incidencia mediante mensajes de correo electrónico o de voz.
- Los protocolos de seguridad permiten una gestión segura y eficiente de los datos, limitando el acceso a personal no autorizado.

6.2. Interfaz gráfica

Las interfaces gráficas permiten la elaboración de pantallas de usuario a través del uso de múltiples combinaciones de imágenes y/o textos otorgando al operador las funciones de control y supervisión de la planta. Gracias a las librerías de objetos propias de los paquetes de diseño de los sistemas es posible relacionar, de manera sencilla, variables del sistema a objetos previamente generados.

6.3. Estándar OPC

OPC (OLE for Process Control) es un estándar de comunicación en el campo de la automatización y supervisión de procesos industriales, basado en la tecnología OLE (Object Linking and Embedding) desarrollada por Microsoft. Gracias a esta tecnología es posible gestionar documentos compuestos por elementos heterogéneos, es decir,

documentos que, además de texto, contengan otros archivos de datos como imágenes, sonido o video.

Ilustración 91 - OPC

El problema de la utilización de protocolos de comunicación específicos para cada fabricante reside en la rigidez a la hora de establecer una arquitectura del sistema versátil, debido a que cada controlador sólo sirve para comunicar unos elementos determinados, en función del fabricante.

Cuando el estándar fue lanzado por primera vez en 1996, su objetivo era el de abstraer los protocolos específicos en una interfaz estandarizada que permitiera a los sistemas HMI/SCADA interactuar con intermediario que convirtiera peticiones de lectura/escritura OPC genéricas en peticiones de los controladores específicos, y viceversa.

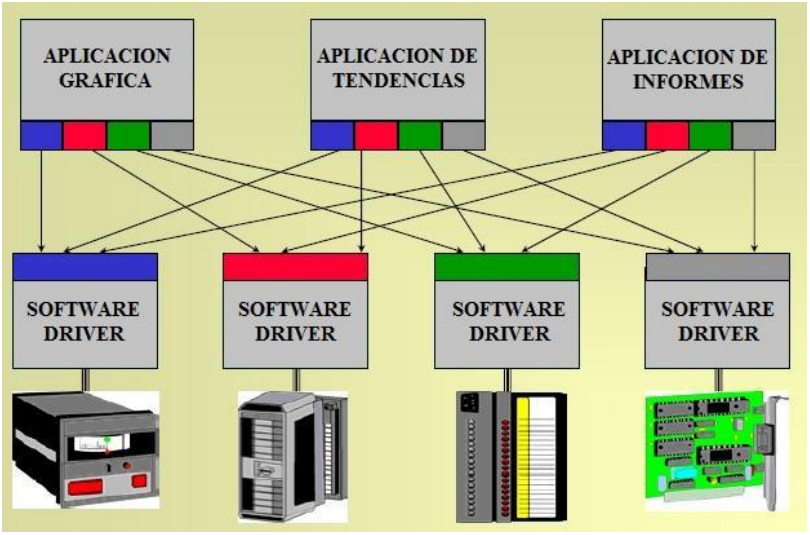


Ilustración 92 - Problema anterior OPC

Así, el estándar OPC representa una solución sencilla, flexible y abierta al problema de los drivers propietarios estableciendo una interface estándar, de forma que los datos se envían y reciben de una determinada manera, independientemente del elemento que

realice el intercambio. Utilizando la tecnología OPC, con cualquiera que sea la fuente de los datos, el formato de presentación y acceso a los mismos será fijo. De esta manera será posible el intercambio de datos con cualquier equipo que cumpla el estándar.

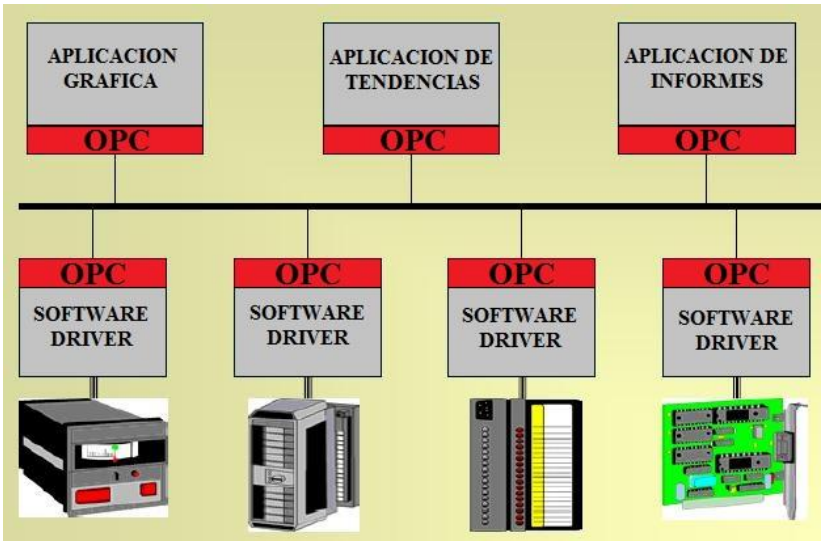


Ilustración 93 - Solución OPC

La comunicación OPC se realiza en base a una arquitectura de tipo servidor-cliente. En ella el servidor (OPC Server) realiza la recopilación de datos de los diversos elementos de campo del sistema automatizado y permite el acceso libre a ellos desde las aplicaciones que lo soliciten. Por otro lado, el cliente (OPC Client) es una aplicación que sólo utiliza los datos. Gracias a las características de estandarización OPC cualquier cliente puede comunicarse con cualquier servidor OPC sin importar el tipo de dispositivo que recoge los datos.

Según la norma del estándar se diferencian varios tipos de servidores:

- Servidor de Acceso a Datos (OPC DA): Proporciona información sobre el estado del hardware y de las conexiones, así como de los parámetros de control del sistema.
- Servidor de Acceso a Datos Históricos (OPC HDA): Facilita que otras aplicaciones accedan a datos de un dispositivo, a través de un interfaz Cliente OPC.
- Servidor de Alarmas, Condiciones y Eventos (OPC A&E): Provee al sistema de un interfaz utilizado exclusivamente para la notificación de alarmas, así como sus características asociadas y eventos de importancia.
- Intercambio de Datos (OPC DX): Permite a servidores OPC-DA intercambiar directamente datos sin la exigencia de un cliente OPC intermedio.

6.4. VISU +

Para llevar a cabo el desarrollo de la aplicación SCADA en cuestión se ha utilizado la herramienta software *Visu+* v2.31. Dicha programa forma parte del paquete software de automatización ofrecido por la empresa patrocinadora del concurso. En él se incluyen, entre otros, la herramienta utilizada para la implementación del programa de control del autómata (PCWorx) y el servidor OPC necesario para generar el interfaz de comunicación entre los dispositivos de campo y la propia aplicación SCADA.

Ilustración 94 - Versión VISU +

Mediante el uso de *Visu+* es posible la generación de interfaces graficas de usuario utilizadas en PCs industriales, basados en sistemas operativos Windows; o utilizadas directamente en paneles táctiles que presenten sistemas operativos Windows CE. Además, las prestaciones de *Visu+* van más allá de la simple implementación de pantallas de explotación. Como software de desarrollo de aplicaciones SCADA dicha herramienta incluye, entre otras, las funciones de:

- Supervisión y gestión de datos de proceso, así como almacenamiento de los mismos.
- Suministro en tiempo real de la evolución de las variables del sistema, además de generación de informes acerca de dicha evolución.
- Tratamiento y análisis de alarmas, así como suministro de informes históricos de las mismas.
- Gestión de usuarios, identificación de operadores y establecimiento de niveles de seguridad y acceso a la instalación.

- Motor lógico propio. Dedicado a la realización de tareas sencillas como la ejecución de cálculos utilizados en funciones lógicas o para la modificación de propiedades dinámicas de objetos utilizados en las diferentes pantallas de explotación.
- Gestión de centralita de notificación de alarmas para el aviso a usuarios vía teléfono, email, SMS o fax.

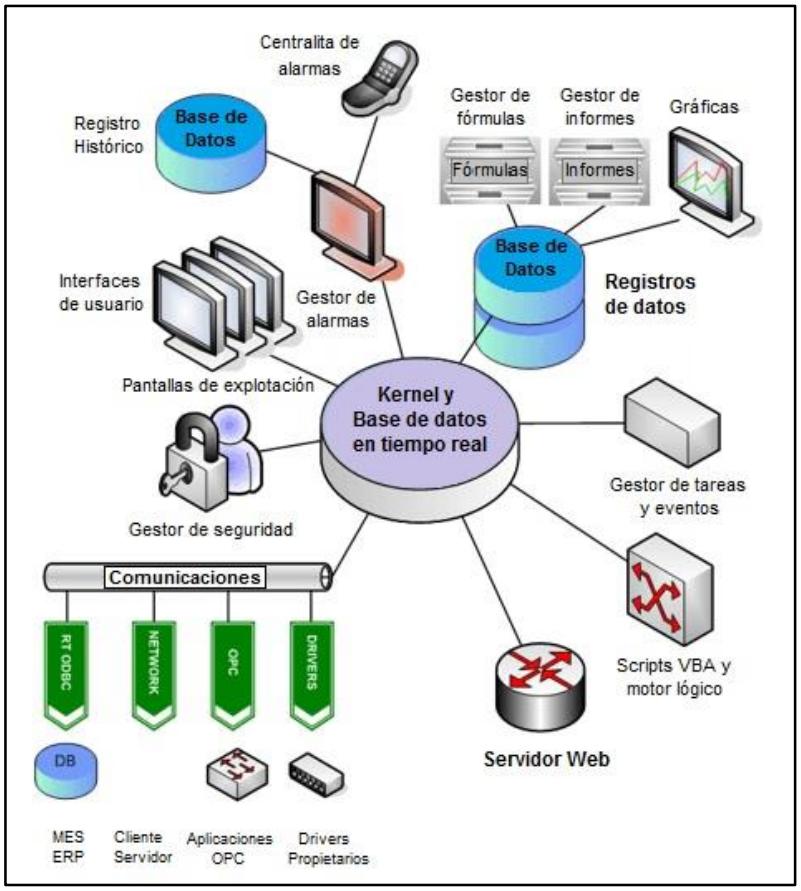


Ilustración 95 - Esquema VISU +

La herramienta presenta dos modos de funcionamiento: el modo de ejecución y el modo de diseño. En el primer caso el programa permite la ejecución de cualquier aplicación desarrollada con anterioridad. En el modo de diseño es posible la creación de nuevas aplicaciones o la edición de las ya existentes por medio de la incursión de nuevos recursos.

6.5.Aplicación SCADA desarrolloPlanteamiento, especificación de requisitos, análisis

Para una correcta visualización de los registros almacenados en la base de datos, ha sido necesario el diseño y desarrollo de una interfaz de usuario en la que además de visualizar las celdas se podrá también almacenar actualizar y borrar registros de la base de datos.

6.5.2. Diseño y desarrollo

El Interfaz de usuario desarrollado ha seguido el mismo diseño del SCADA general del proyecto, desarrollado por otro proyectante. Se han utilizado los mismos recursos, forma y tipo de interruptores, botones, colores, tipos de letra etc, con el fin de intentar asemejar el SCADA realizado con el ya desarrollado, a con el propósito de que se integre en el HMI desarrollado

6.5.3. Guía de usuario del sistema

Para el diseño del SCADA se ha elegido un esquema simple y sencillo, el cuál sigue el siguiente flujograma que se describe con más detalle a continuación.

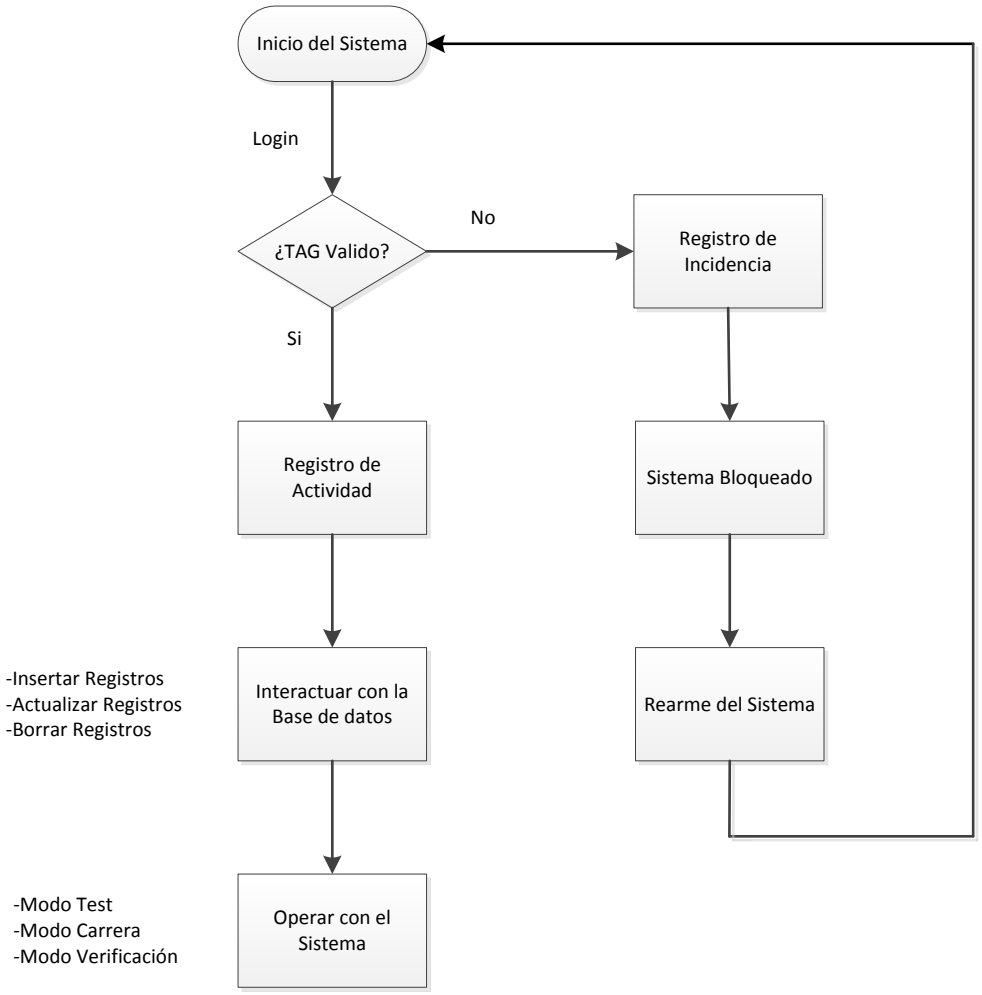


Ilustración 96 - Esquema SCADA

6.5.3.1. Inicio del Sistema

Al ejecutar la aplicación SCADA se encenderán las luces amarillas de la pista, iniciando una cuenta atrás, que terminará con el sonido de la bocina, esto confirmará que la base de datos se ha conectado con el PC Worx, en el SCADA en la parte superior derecha aparecerá un check cuando el servidor OPC esté conectado con el VISU+.

Ilustración 97 - Check

También aparecerá un mensaje solicitando pasar por el lector un TAG valido, considerando valido un TAG cuyo identificador interno este registrado en la base de datos, quedando este registro grabado en la base de datos. Si es válido se avanzará a la siguiente pantalla pero si el identificador interno del TAG no está almacenado en la base de datos se producirá un warning (alarma), grabando la incidencia en la base de datos, encendiendo señales luminosas en el sistema y emitiendo un aviso en el SCADA con ello aparecerá también un botón con forma de seta de emergencia para la realización de un rearme, con esto las señales luminosas se detienen para realizar el rearme completo es necesario pasar por el lector un TAG valido.

Ilustración 98 - TAG no válido

Al pasar un TAG cuyo identificador este almacenado en la base de datos se encenderán en el sistema los LEDs verdes de la pista, y la luz blanca del semáforo, confirmando la lectura además de grabar el registro en la base de datos.

Ilustración 99 - Sistema en espera de leer TAG

6.5.3.2. Interactuar con la base de datos

Una vez registrado un TAG valido el SCADA avanzará hasta la siguiente pantalla, en la cual se podrá interactuar en la base de datos, mostrando la información de las tablas de la base de datos. Players, Cars, Register y Warnings. Es necesario mencionar que si le muestra una tabla no se podrá visualizar otra hasta que se desactive la tabla actual.

Junto a la tabla aparecerá un botón con el símbolo +, presionando este botón si en la base de datos hay mas registros de los mostrados, aparecerán nuevos registros, y un botón con - que servirá para mostrar los registros anteriores, en caso de no haber más registros, aparecerá el botón menos pero no aparecerán mas registros.

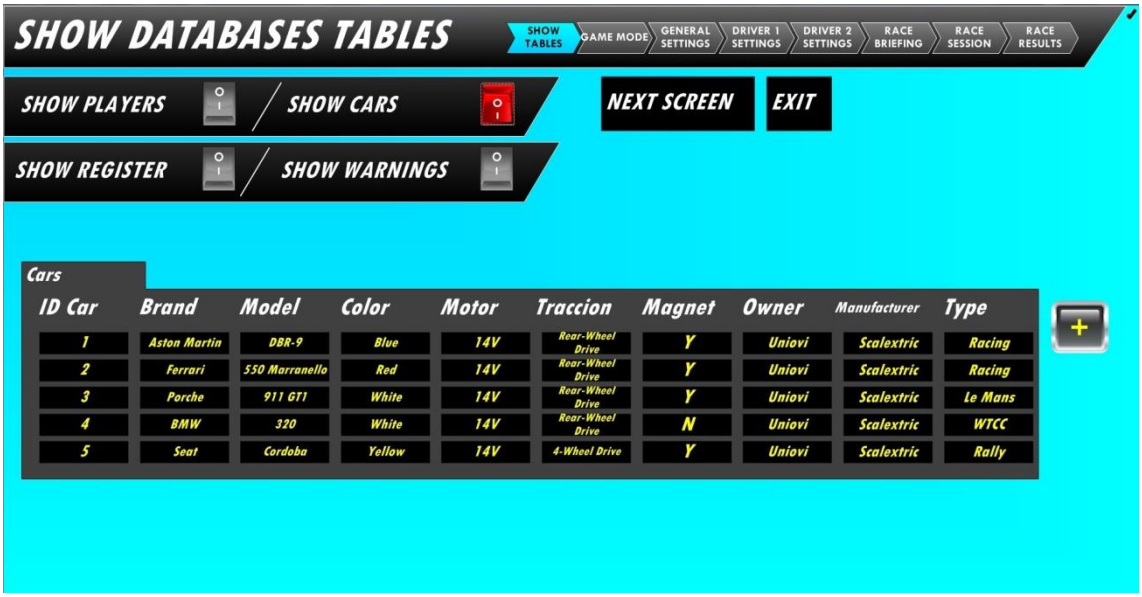


Ilustración 100 - Show Cars Activado

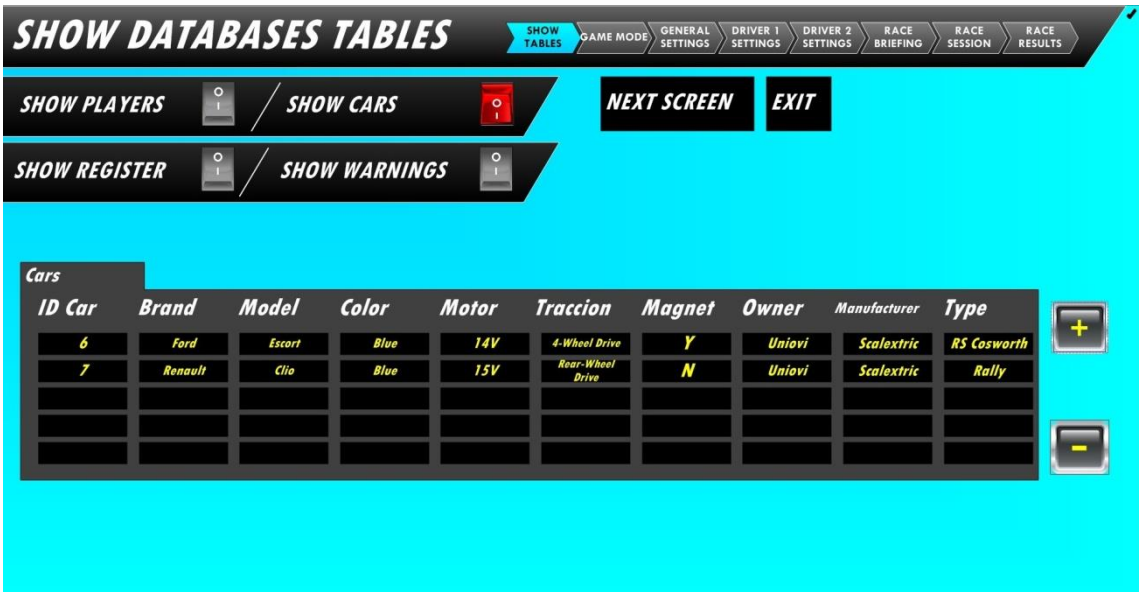


Ilustración 101 - Show Cars 2

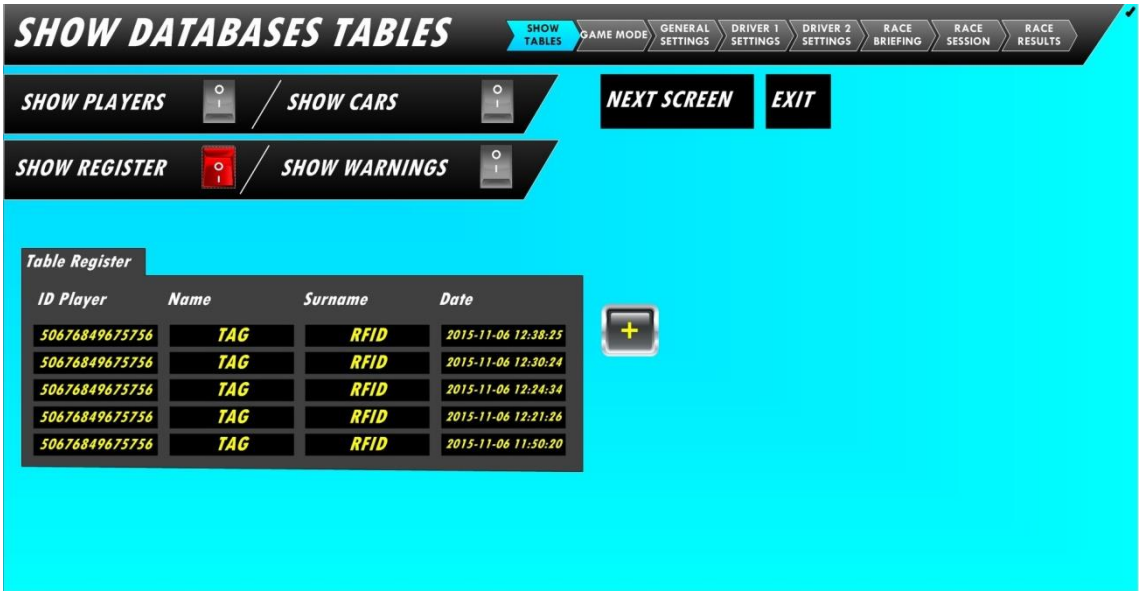


Ilustración 102 - Show Register

Si al leer un TAG es el que esta configurado como administrador, aparecerá en esta pantalla la opción de administrar tablas, presionando este nuevo botón, se podrá entrar a las opciones de administración de la base de datos, con lo que se podrá modificar valores de la base de datos modificar el número de créditos y el tiempo de descuento de los créditos.

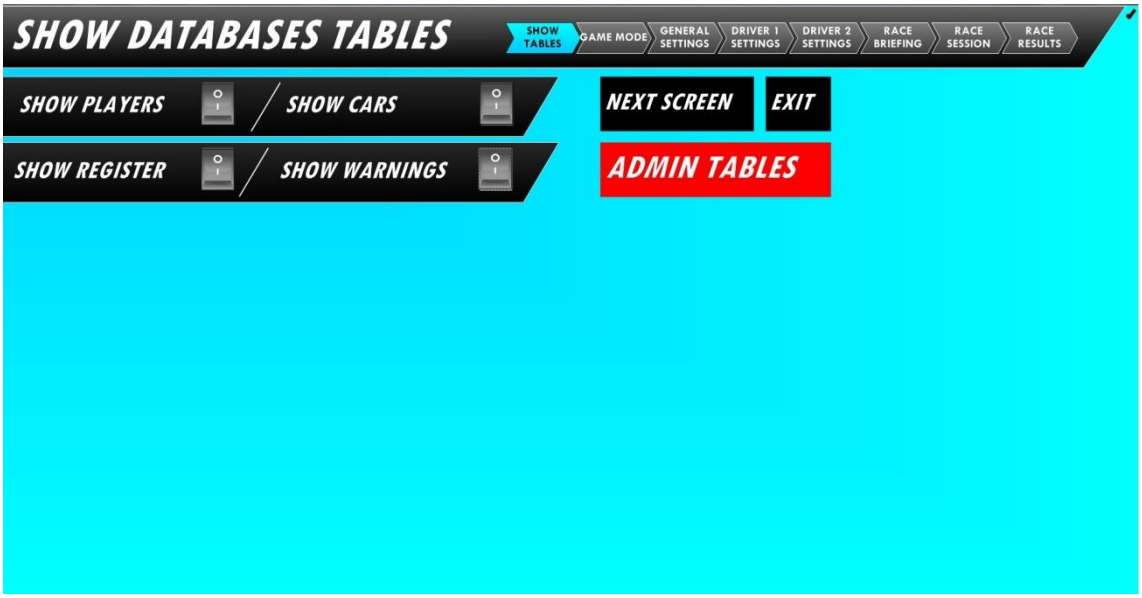


Ilustración 103 - Botón Admin Tables

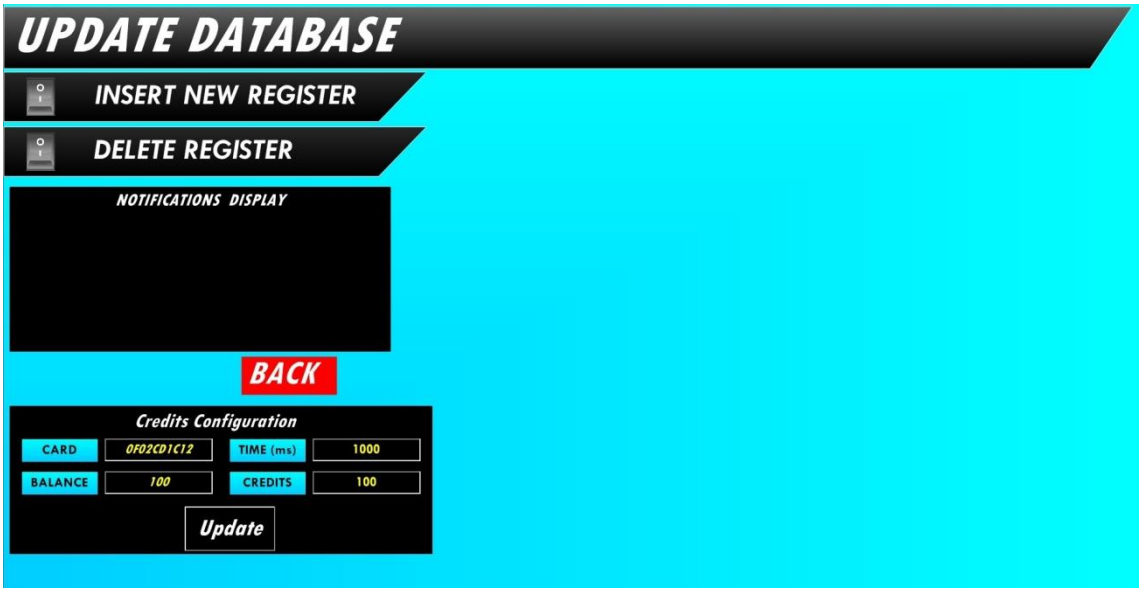


Ilustración 104- Update Database

Dentro de esta pantalla se tiene la opción de insertar un nuevo registro en la base de datos, borrar un registro de la base de datos, un display para mostrar las notificaciones, un botón back, para volver a la pantalla anterior, y debajo un submenú para configurar los créditos del sistema.

Si se quiere insertar un nuevo registro se le tiene que dar al botón de 'Insert New Register' una vez hecho esto se desplegarán dos paneles en los que se podrá modificar los valores de los registros para las tablas de *players* y *cars*.

Es importante destacar que en la casilla *id Player* y *Card*, no se podrán modificar y solo cambiaran a pasar un TAG por el lector, en la casilla *Credits* se solo admitirá valores numéricos naturales en el resto de casillas se podrá poner lo que el administrador quiera.

Una vez rellenadas todas las casillas se procederá a checkear si el TAG seleccionado para vincular la información esta disponible o si por el contrario ya esta registrado en la base de datos.

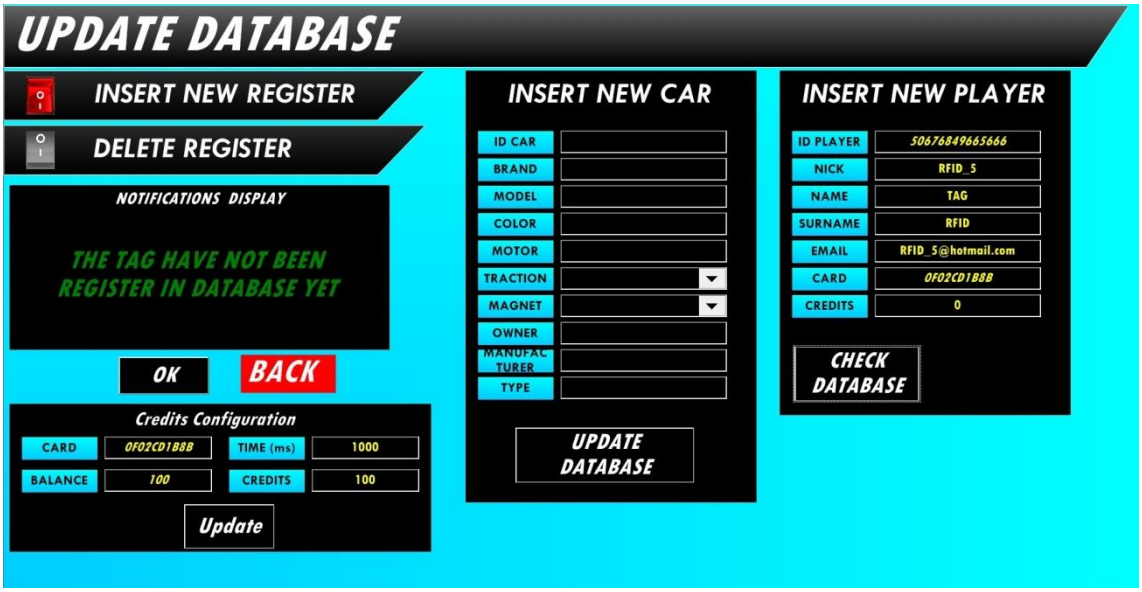


Ilustración 105 - Register Good

Si el TAG no está ya registrado en la base de datos aparecerá un mensaje en verde en el display confirmándolo, pero si por el contrario ya estuviera registrado aparecería un mensaje en rojo en el Display avisando que el TAG ya está registrado.

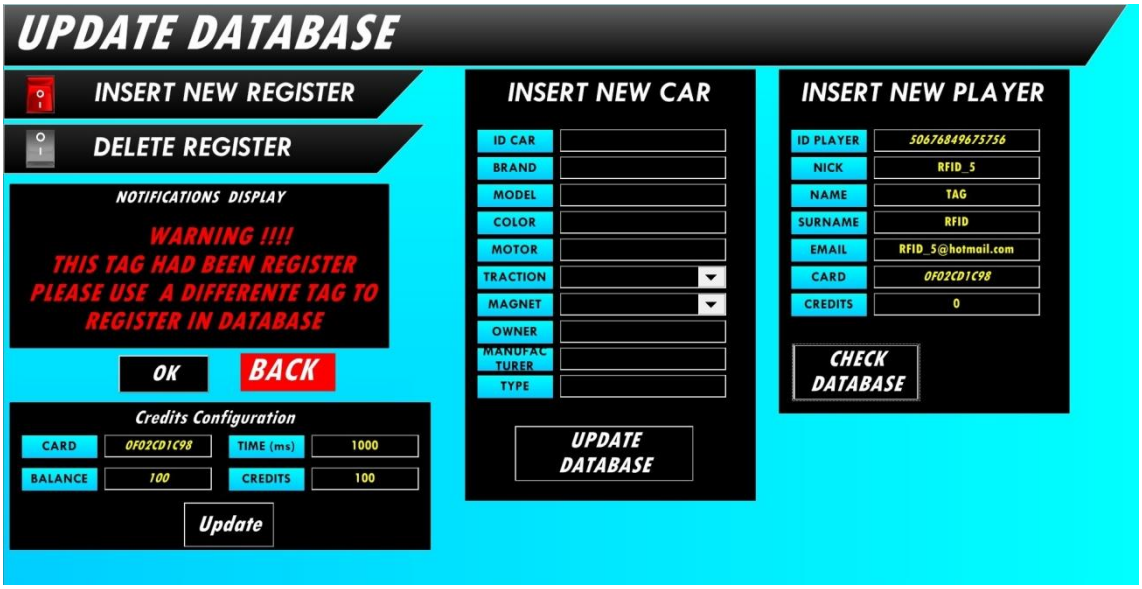


Ilustración 106 - Register Fail

Para poder proseguir es necesario presionar el botón OK para confirmar la recepción del aviso, si el TAG es válido aparecerá un botón al lado de 'Check Database' con la opción de actualizar la base de datos con el botón 'Update Database'.

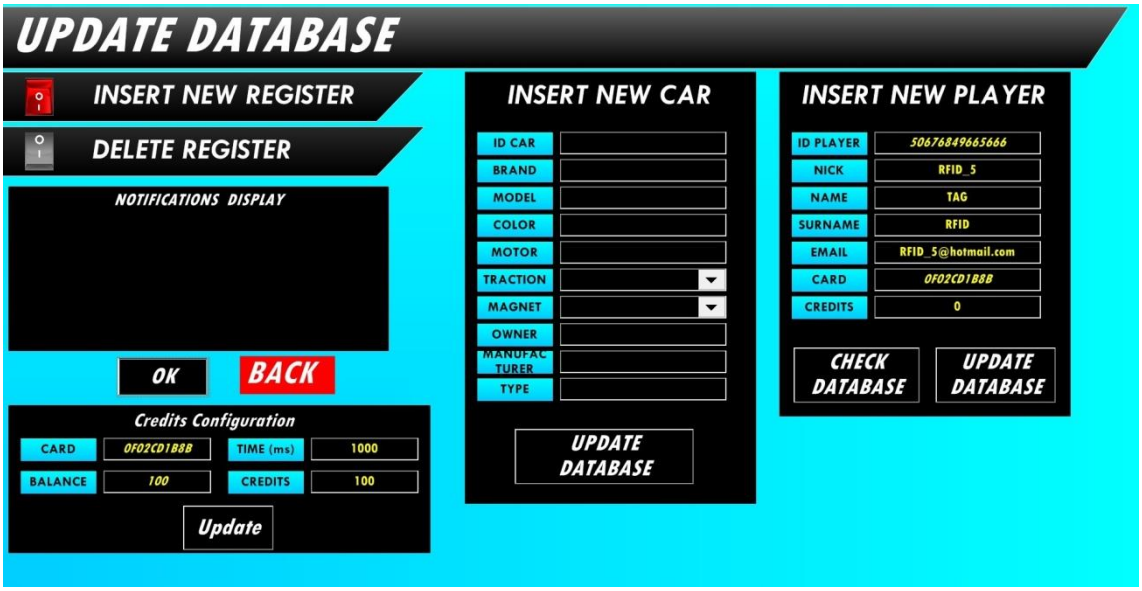


Ilustración 107 - Update Database

La parte para insertar un nuevo coche debido a que los TAGs de tipo botón el nuevo id car habrá que introducirlo a mano, teniendo cuidado de no repetir un TAG ya introducido anteriormente. Las casillas de *Traction* y *Magnet* tienen unos valores precargados debido a que un coche solo dispone de tracción trasera (Rear Wheel Drive), tracción delantera (Front Wheel Drive) o tracción total (4- Wheel drive), y con el imán o el coche dispone de imán (Y) o no dispone (N).

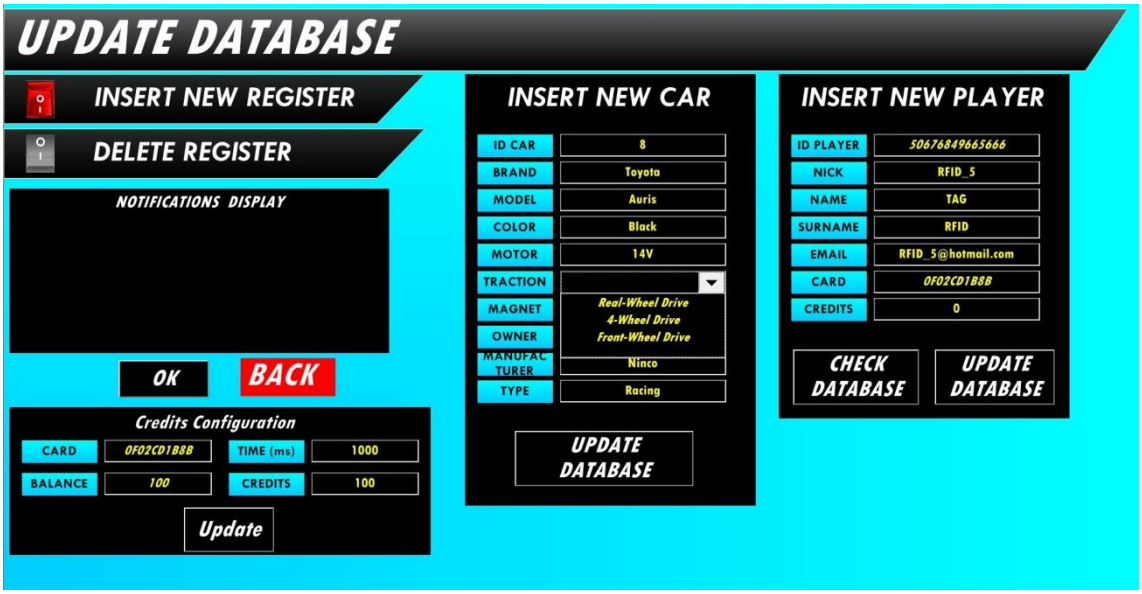


Ilustración 108 - New Car

Una vez insertado un nuevo registro se puede volver a la pantalla anterior y comprobar si efectivamente se ha grabado en la base de datos el nuevo registro.

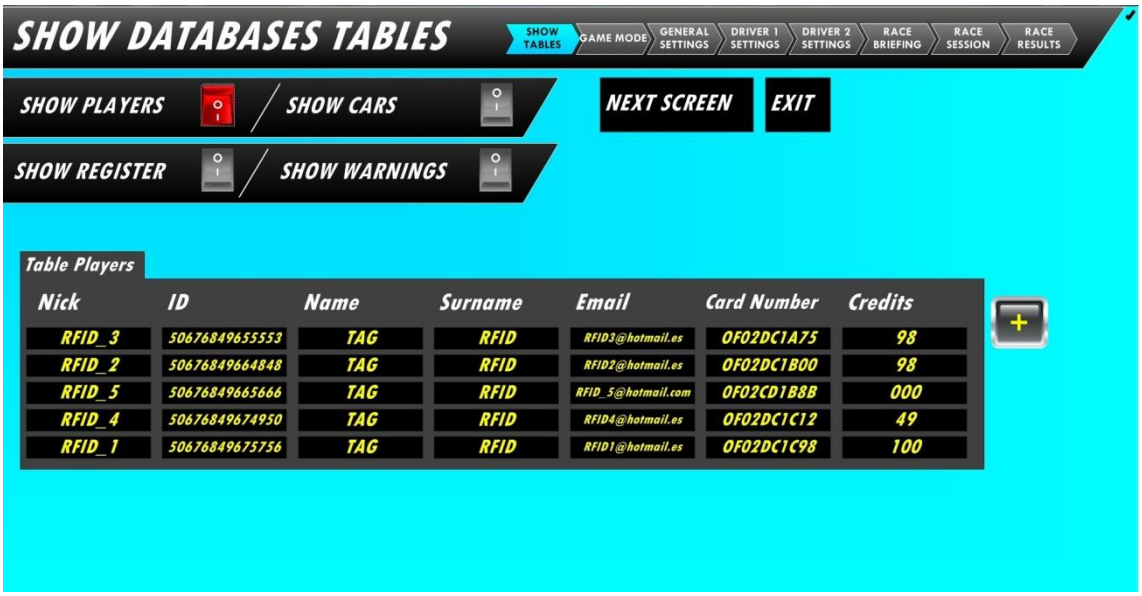


Ilustración 109 - Inserción nuevo registro

6.5.4. Borrar Registro

Si se quiere borrar un registro ya existente hay que seleccionar la opción de delete register, al presionar el botón se desplegaran dos paneles con las opciones para borrar un registro de la base de datos.

Para borrar un jugador es necesario pasar el TAG que se quiere borrar por el lector, se cargarán los datos de la tarjeta en las casillas 'ID PLAYER' y 'CARD', una vez hecho esto para borrar el registro hay que darle al botón 'Delete player', una vez hecho esto aparecerá un aviso en el Display confirmando la acción.

Si se quiere borrar un coche hay que introducir a mano el ID CAR del coche que se quiere borrar, una vez hecho esto al darle a botón 'Delete Car' y se eliminará el registro de la base de datos y se notifica en el Display.

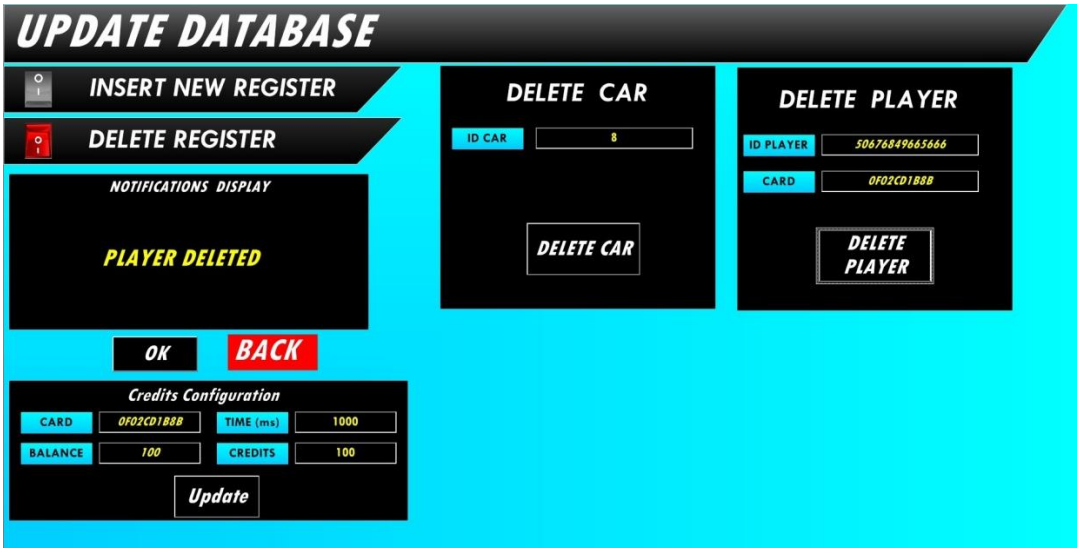


Ilustración 110 - Delete Player

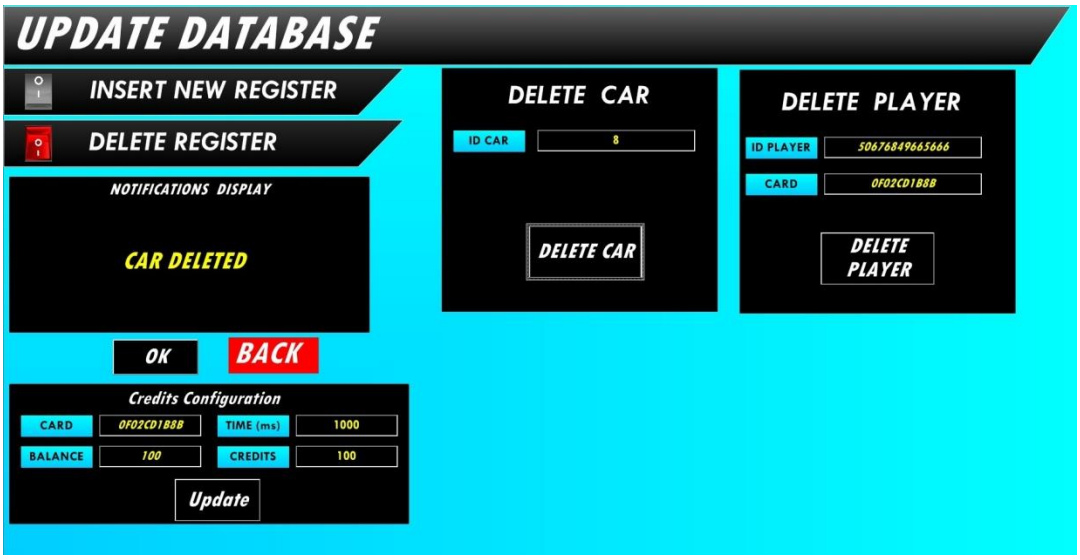


Ilustración 111 - Car Deleted

6.5.5. Credits Configuration

En este panel se pueden modificar los créditos de cada TAG, en la casilla *Card* se puede ver sobre el que se va a actuar, *Time* el tiempo de descuento de créditos en milisegundos, *Balance* que muestra los créditos que tiene actualmente el TAG, y *Credits* donde se puede añadir nuevos créditos. Como máximo el sistema acepta un máximo de 100 créditos y un máximo de 10 segundos por crédito (10000 ms).

Hay que mencionar que si no se dispone de algún crédito antes de empezar un modo de juego, el sistema no permitirá avanzar hasta que se carguen créditos.

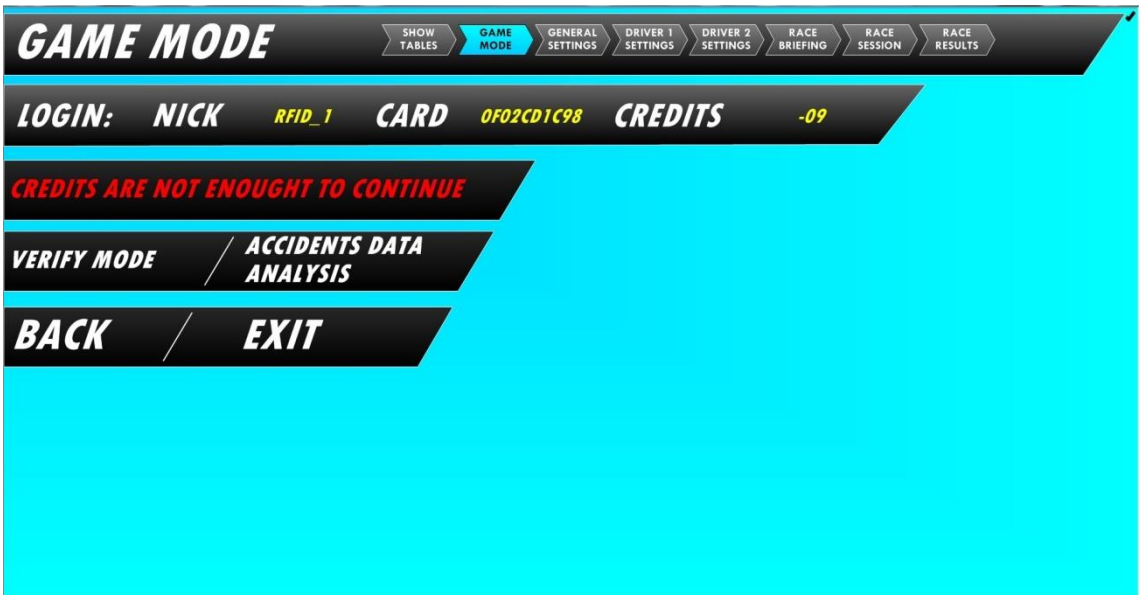


Ilustración 112 - Créditos Insuficientes

Es importante mencionar que si tienen créditos suficientes para interactuar con el sistema pero, al cabo de tiempo se agotan la carrera o el test no se detendrán sino que seguirá descontando por lo que cuando se salga al menú de elección de modo de juego se mostrará cuantos créditos de mas se han consumido, y siendo menor que cero no permitirá realizar un test o una carrera hasta que se realice una recarga de créditos o se pase otro TAG con créditos suficientes para realizar un test o una carrera.

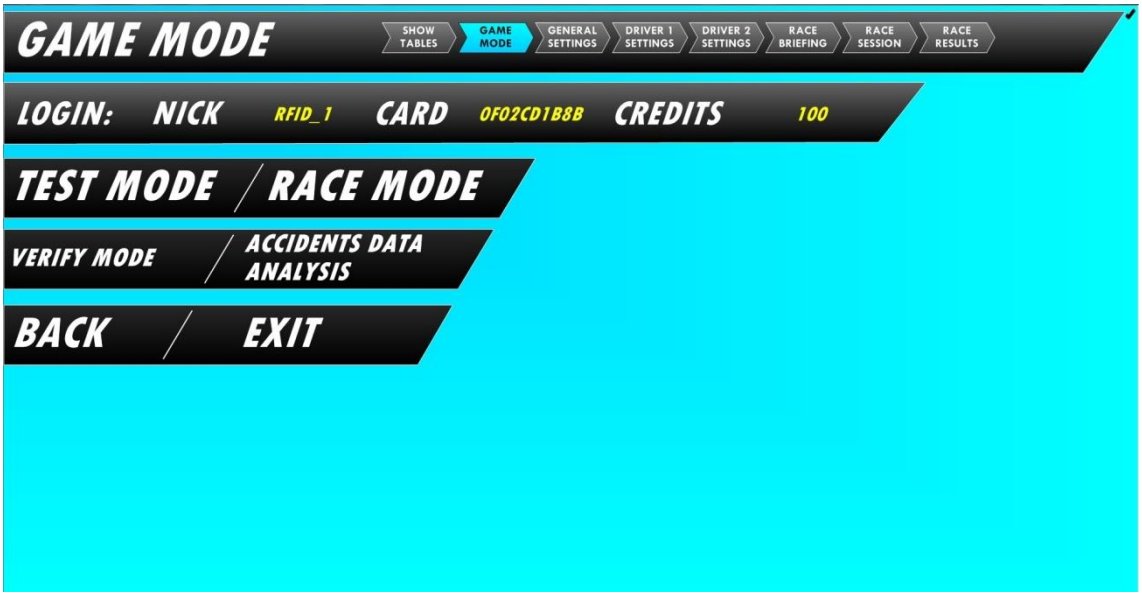


Ilustración 113 - Créditos suficientes

Posteriormente al avanzar de esta pantalla se entrará ya en la parte desarrollada del SCADA principal del Proyecto realizada por otro alumno. Realizando pequeños retoques en algunos botones para volver a diferentes pantallas propias de este proyecto.

7. Planificación

7.1. Recursos humanos

Para la realización del proyecto Xplore-Uniovi en su conjunto han colaborado hasta 17 personas. Para los trabajos descritos en el presente documento se dispone de 1 persona para las etapas del proyecto que se describirán a continuación.

7.2. Etapas del Proyecto

La planificación es una parte fundamental en la realización de cualquier proyecto, pero es a la vez complicada de establecer con detalle. Especialmente en el presente proyecto, la participación de un total de 17 alumnos hace muy difícil su seguimiento sin que surjan imprevistos que afecten a todas las áreas. Esto implica que no siempre se puedan realizar las tareas de principio a fin de una vez, y se intercalen un poco unas y otras. De cualquier forma, la planificación seguiría la siguiente estructura

1. Labores previas: definición de objetivos, elección del software, búsqueda de información, estudio, planificación, etc.
2. Estado del arte RFID: Estudio de diferentes tecnologías de transmisión de datos inalámbricas, arquitectura, componentes…
3. Diseño de la base de datos: Búsqueda de diferentes bases de datos y diseño de las tablas necesarias.
4. Creación de la base de datos: Creación de la base de datos y de las tablas requeridas, además de inicialización de determinados valores en caso de ser necesarios.
5. Diseño del Programa de control: Estudio del código previamente realizado y diseño del software necesario para adecuarlo a las necesidades RFID.
6. Desarrollo del Programa de Control: Programación del control y comprobación de este en un simulador del PLC en PC.
7. Diseño y Desarrollo de la aplicación HMI y del programa de control, realización de bocetos de las pantallas que conformarán la interfaz así como la programación en VISU+ y comprobación de un correcto funcionamiento.
8. Implementación en el sistema real: Con la infraestructura ya terminada, implementación de todo el trabajo anterior en el PLC real y comprobación de su correcto funcionamiento en la práctica. Corrección de fallos y rediseño de las funcionalidades que necesiten mejoras.
9. Realización de la documentación del proyecto.

7.3. Diagrama GANTT

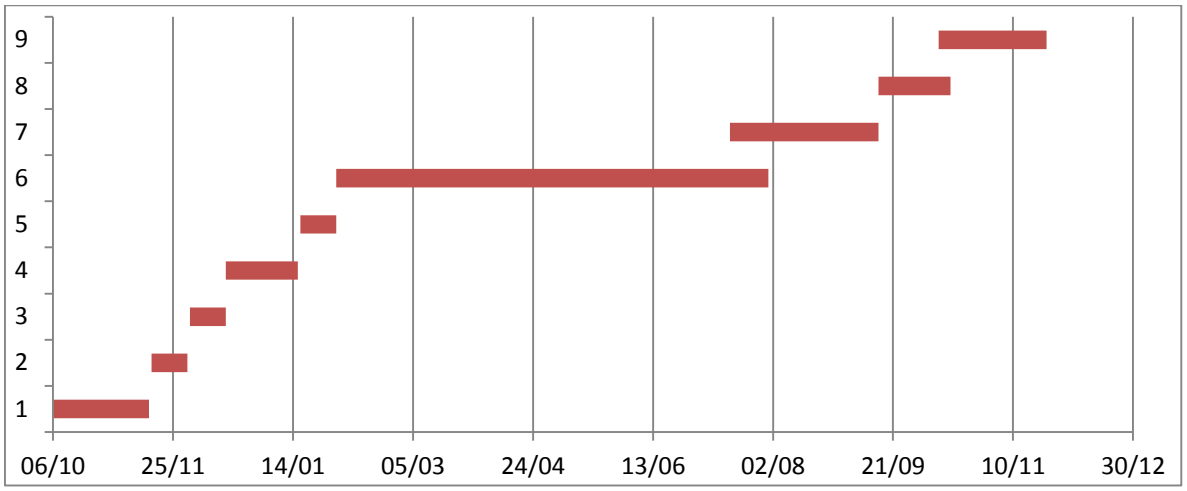


Ilustración 114 - Diagrama GANTT

8. Discusión y conclusiones

8.1. Objetivos conseguidos

El objetivo principal propuesto ha sido la implementación de tecnología RFID en el actual proyecto Xplore, para ello se ha conseguido desarrollar satisfactoriamente una base de datos para almacenar los registros característicos de usuarios y coches. Se ha desarrollado un programa en PC Worx para el control de acceso a la base de datos y una ampliación de la aplicación SCADA basada en Visu+ para supervisión y explotación del sistema.

Por otro lado, la integración de todas las funcionalidades con el resto de tecnologías del proyecto, has sido llevadas a cabo satisfactoriamente.

Como aspecto de interés a comentar del proyecto hay que mencionar la realización las comprobaciones del correcto funcionamiento de cada nueva funcionalidad a medida que se iba implementando, mediante el uso del software PC Worx SRT, un programa que simula un PLC en el PC y que permite la ejecución de los programas de control sin necesidad de usar uno físico. De igual forma, con el programa de control y el HMI terminados se han realizado comprobaciones del funcionamiento general de todo el programa, controlandolo desde la interfaz, que permite la ejecución de todas las opciones programadas, comprobando que responde de forma correcta cada una de las implementaciones realizadas.

Con la infraestructura ya disponible se han llevado a cabo pruebas durante la puesta en marcha. Haciendo uso ya del sistema físico, y tras algunas modificaciones que se vieron necesarias al probarlo en la práctica, se pudo finalmente comprobar su correcto funcionamiento en todas las situaciones posibles del sistema.

Finalmente el sistema completo fue controlado durante días por varios miembros del proyecto Xplore, para mostrar su funcionamiento a otras personas o de cara a probar otras funcionalidades. El diseño del HMI ha demostrado ser suficientemente sencillo de utilizar (a pesar de la cantidad de información que maneja) y bastante intuitivo.

Así y todo han sido necesarias pruebas adicionales de integración con otros proyectos.

8.2.Dificultades encontradas

Al igual que en cualquier proyecto, ha habido diversas dificultades a las que hacer frente durante la realización del mismo. La más significativa ha sido la comunicación entre los diferentes programas:

Base de Datos - PC Worx - Visu +

En especial la comunicación OPC del Visu+ con el PLC y los tiempos de respuesta del SCADA con la base de datos.

Por otro lado la adaptación a la forma de programación ya existente realizada para el programa de control, y para el SCADA desarrollada por otros miembros del equipo Xplore, implica además una dificultad adicional.

8.3.Posibles ampliaciones

Tratándose de un proyecto educacional y pensado desde un primer momento como un proyecto abierto a nuevas mejoras, las posibles ampliaciones pueden ser numerosas. Algunas de las que se podrían destacar, centrándose solamente en lo descrito en este proyecto y sin la utilización de otras tecnologías podrían llegar a ser:

- Utilización de la información almacenada en la base de datos para la creación de estadísticas de los coches y jugadores (tiempo jugado, distancia total recorrida, número de veces vencedor…).
- Cambiar el lector RFID por otro lector de lectura y escritura para desarrollar almacenar la información en los TAGs prescindiendo de la base de datos.
- Desarrollar el proyecto bajo otras tecnologías sin contacto similares al RFID.

9. Bibliografía

- PC Worx Quick Start. Phoenix Contact. (https://www.phoenixcontact.com/online/portal/us?uri=pxc-oc-itemdetail:pid=2985725&library=usen&pcck=P-19-05-01&tab=1)

- User Manual PC Worx SRT. Phoenix Contact. (https://www.phoenixcontact.com/online/portal/us?uri=pxc-oc-itemdetail:pid=2701680&library=usen&tab=1)

- MySQL Reference Manual. MySQL AB.

- (http://dev.mysql.com/doc/)

- User Manual DBFL FB. Phoenix Contact.

- Manual de la biblioteca para el trabajo en MySQL con PC Worx

- Manuales AXC 1050. Phoenix Contact. (https://www.phoenixcontact.com/online/portal/es?uri=pxc-oc-itemdetail:pid=2700988&library=eses&tab=5)

- Proyecto Fin de Carrera *David Parte Rodríguez* - Control PLC y supervisión webvisit para desarrollo de carreras de Rally-Slot del proyecto Xplore-Uniovi.

- Proyecto Fin de Carrera *Juan Miguel Rodríguez Corral* – Diseño e implementación del sistema de supervisión, control y adquisición de datos basado en VISU+ para el proyecto XPLORE-Uniovi