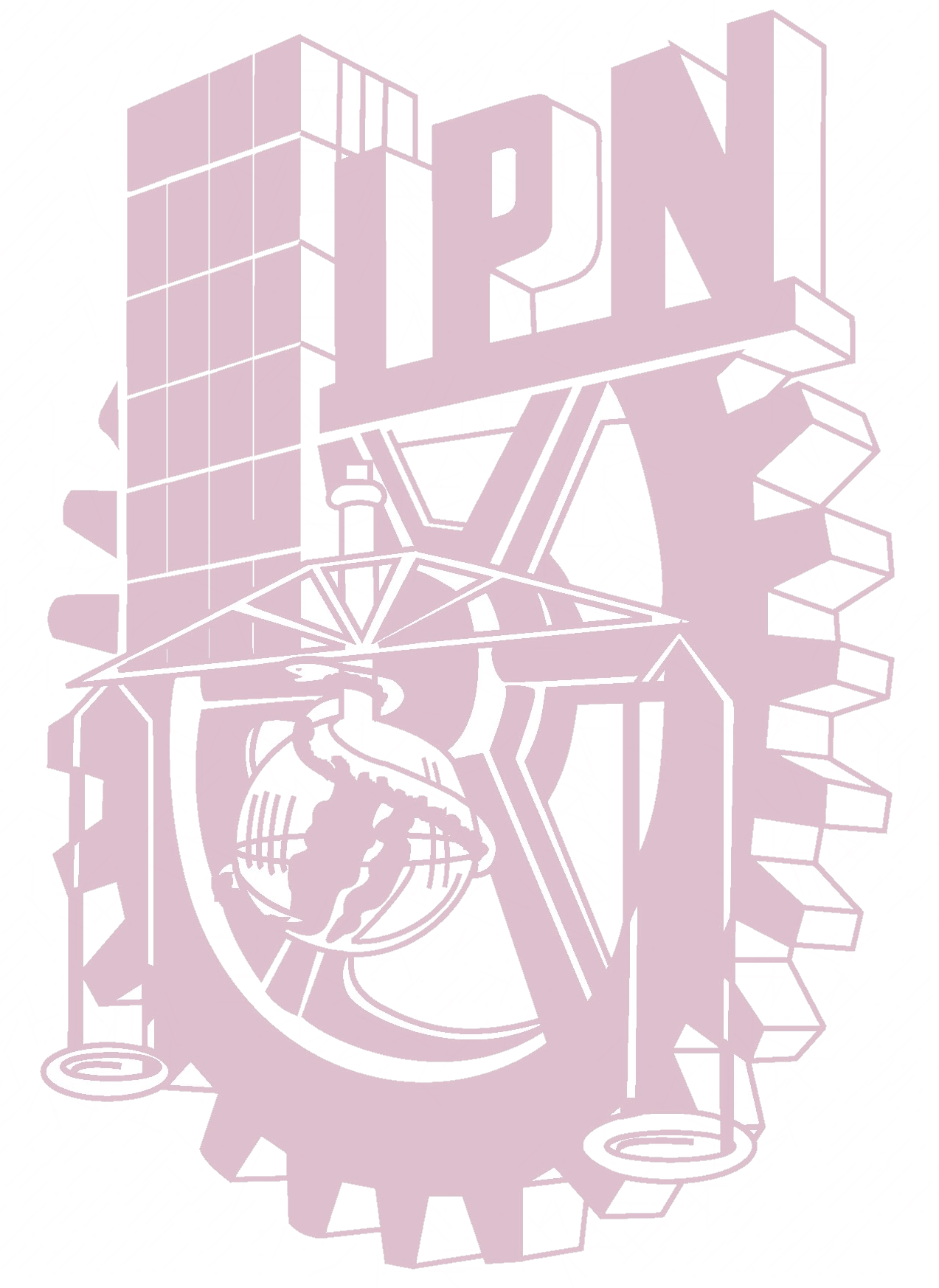
**Instituto Politécnico Nacional**

**“**Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas”

****Sistemas Operativos en Tiempo Real.

Lamberto Maza Casas.

Alternativas de Sistemas Operativos en Tiempo Real:

“Windows Embedded”

Alumnos: Fernández Delgado Carlos

Olvera Bahena David

Grupo: 3MV9

# Alternativas de Sistemas Operativos en Tiempo Real:

# “Windows Embedded”

## Introducción

Un sistema operativo es el núcleo del programa que administra servicios, recursos y funciones de un ordenador, así mismo es la conexión entre las entradas y salidas de hardware de las que dispone, esto le permite al usuario interactuar con todo el hardware.

Dentro de los sistemas operativos podemos distinguir los sistemas operativos de tiempo compartido y los sistemas operativos en tiempo real, estos primeros dependen de la velocidad del procesador, trabajan en una plataforma multitareas que asigna a cada proceso un tiempo, dando la apariencia de que el realiza varias tareas a la vez sin embargo la duración de cada proceso afecta a los siguientes, estos sistemas pueden ser multiusuarios.  
A diferencia de los sistemas de tiempo compartido que tienen un comportamiento estocástico, los sistemas operativos tienen un comportamiento determinista, donde en función de los estados presentes y pasados se tiene una respuesta para tener un acercamiento del estado futuro, estos sistemas interactúan con el mundo real, es de ahí la importancia de su respuesta a tiempo, sin embargo este tiempo de respuesta dependiendo de las necesidades puede darse en nano segundos, minutos, horas o inclusive días.

Los sistemas en tiempo real deben cumplir con las siguientes características:

1. Determinismo.
2. Sensibilidad.
3. Control de usuario.
4. Fiabilidad.
5. Tolerancia a fallos.

Actualmente la complejidad de los sistemas requiere que su tiempo de respuesta sea preciso para asegurar su correcto funcionamiento, tal es el caso de los aviones que requieren que el sistema esté actuando conforme al tiempo y no al procesamiento de los datos.

## Windows Embedded

Microsoft desarrolló versiones simplificadas de algunos de sus sistemas para uso específico en sistemas embebidos y computadoras antiguas, dichas plataformas también desarrolladas por Microsoft. Se desarrollaron 4 versiones:

* Windows XP Embedded
* Windows Embedded Standard 2008
* Windows Embedded Standard
* Windows Embedded 7

Para la version de Windows Embedded 7, se tiene una versión aún más ligera, Windows Embedded Compact 7 (WEC 7), un sistema operativo en tiempo real capaz de manejar 32,000 procesos concurrentes, las aplicaciones destinadas para este SO se desarrollan en el Entorno de Desarrollo Integrado (IDE) Visual Studio 2008 con el plug-in “Platform Builder” instalado, siendo una de las principales ventajas la flexibilidad de lenguajes soportadas para el desarrollo de aplicaciones, tales como:

* Visual C++
* Visual C#
* Visual Basic

Otra de las ventajas que presenta WEC 7 es el uso de herramientas remotas, lo que permite depurar una imagen del SO Compact 7 de manera remota.  
Los pasos para generar los dispositivos con WEC 7 son:

* Desarrollar un gestor de arranque (bootloader) del Compact 7, los drivers del dispositivo y paquete de soporte de tarjeta (BSP) para el hardware a usar.
* Crear y personalizar un proyecto de diseño de SO para el hardware.
* Generar una imagen personalizada del Compact 7 OS.
* Deployar la imagen del Compact 7 OS en el hardware para hacer pruebas y depurar.
* Generar un SDK del WEC 7 desde el diseño del SO para soportar el desarrollo de aplicaciones.
* El desarrollo de la aplicación puede realizarse de manera concurrente.
* Después de crear la imagen del WEC 7 y sus aplicaciones, se deploya la solución final para su distribución.

## Plataforma de desarrollo



Figura 1. eBox 3300 /3310

La plataforma eBox 3300 /3310 Figura 1 desarrollada por Microsoft es compatible con los SO Windows Embedded, cuenta con las siguientes características:

|  |  |
| --- | --- |
| CPU | MSTI-PDX-600 |
| RAM | 512MB DDR2 |
| VGA | XGI Z9S with 32MB DDR2 Resolution up to 1,280x1,024 High Colors |
| Periféricos | USB V2.0 (host) x 3  Audio (Mic In, Line Out)  Type I/II CF Slot and Micro-SD  Serial ports x 2 (available for eBox3300/3310-JSK)  Mini PCI socket x1 (available for eBox3300/3310-M and 3300/3310-JSK)  2nd RJ-45 LAN (only available on eBox-3300/3310-L2)  PS/2 Keyboard and Mouse  Mic In, Line Out |

## Desarrollo del SO y aplicaciones

Para el desarrollo del SO y aplicaciones es necesario tener instalado:

* Visual Studio 2008
* Windows Embedded Compact 7 CTP
* ICOP\_eBox3310A\_70CTP BSP
* PruebaWE\_SOTR\_SDK.msi
* CoreCon\_v100\_Compact7.msi
* AutoLaunch\_v100\_x86\_Compact7.msi

## Ejemplo de desarrollo de un SO

## Iniciar proyecto

Para iniciar un nuevo proyecto se utilizan las plantillas y se siguen las instrucciones del asistente (wizard) que provee Visual Studio 2008, para ello se accede en la ventana de inicio de Visual Studio 2008 a “**Archivo | Nuevo | Proyecto …**” como se muestra en la Figura 2.

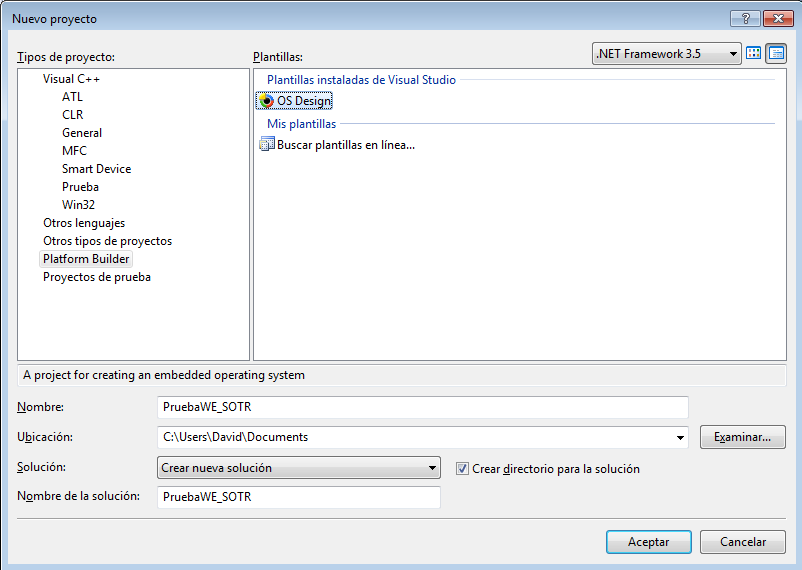


Figura 2. Ventana de nuevo proyecto Visual Studio 2008.

* Seleccionamos como tipo de proyecto la opción Platform Builder.
* Del panel derecho, se selecciona OS Design.
* Se nombra el proyecto, para el ejemplo se nombra PruebaWE\_SOTR.
* Se debe seleccionar la opción de crear directorio para la solución.
* Click en Aceptar y se desplegará un asistente para el OS Design como en la Figura 3.

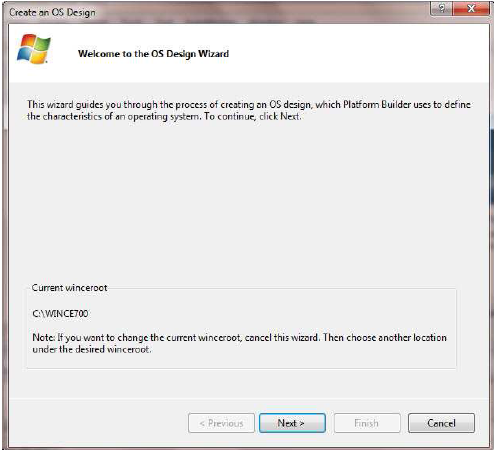


Figura 3. Asistente para crear OS Design.

* Click en siguiente, con ello se desplegará la pantalla de selección de paquetes de soporte de tarjeta (**Board Support Packages**) para seleccionar la plataforma a usar.
* En la pantalla de selección de BSP se muestran los BSPs disponibles, para el ejemplo usaremos la plataforma eBox 3300/3310 por ello seleccionamos el BSP **ICOP\_eBox3310A\_70CTP: x86** tal como se muestra en la Figura 4.

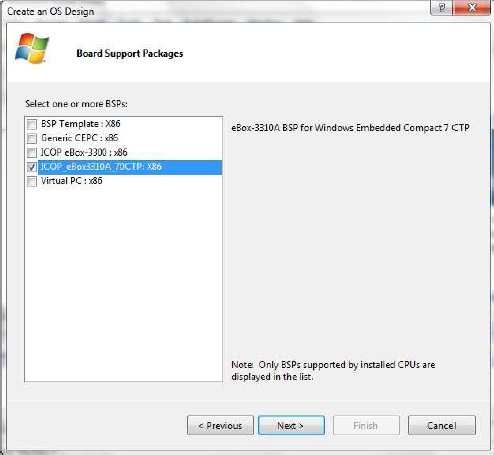


Figura 4. Selección de BSP.

* Damos click en siguiente para mostrar las platillas de diseño de la Figura 5.

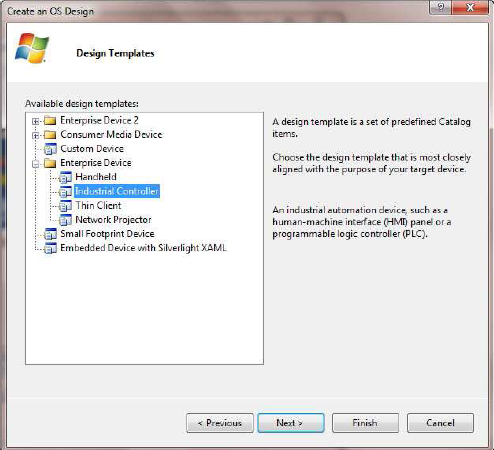


Figura 5. Plantillas de diseño.

* En la pantalla de plantillas de diseño se muestran diferentes opciones cuya selección depende del propósito del dispositivo, para el ejemplo se selecciona la opción de C**ontrolador Industrial** que está dentro de los dispositivos empresariales y damos click en siguiente**.**
* Se desplegará la pantalla de **“Applications & Media”** para seleccionar las aplicaciones que incluirá el SO y se deberán incluir Console Window,Internet Explorer 7.0,Network User Interface,Waveform Audio yWindows Internet Services como se muestra en la Figura 7.

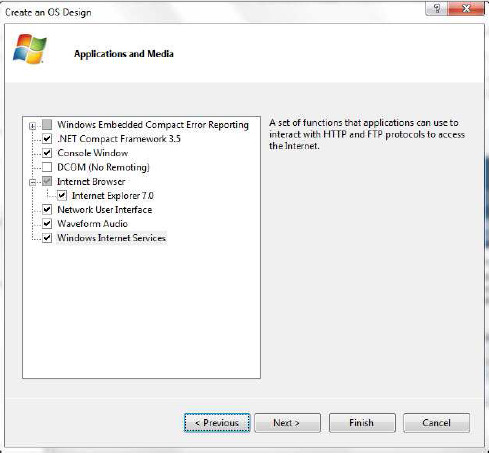


Figura 6.Pantalla de Applicactions and Media.

* Click en siguiente para mostrar la pantalla de “**Networking & Communication”.**
* Seleccionamos la opción de red de área local por cable (wired Local Area Network) como en la Figura 7 y damos click en siguiente.

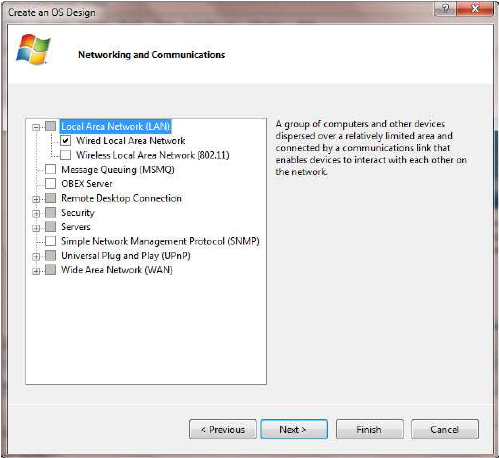


Figura 7. Pantalla de Redes y comunicación.

* La pantalla siguiente muestra que se ha completado el asistente del proyecto, damos click en finalizar.

Al finalizar el asistente se mostrará una pantalla de advertencias de seguridad ya que uno más de los componentes suponen un riesgo para la seguridad, se da click en de acuerdo (acknowledge) con lo que se desplegará la pantalla principal del proyecto como en la Figura 8.

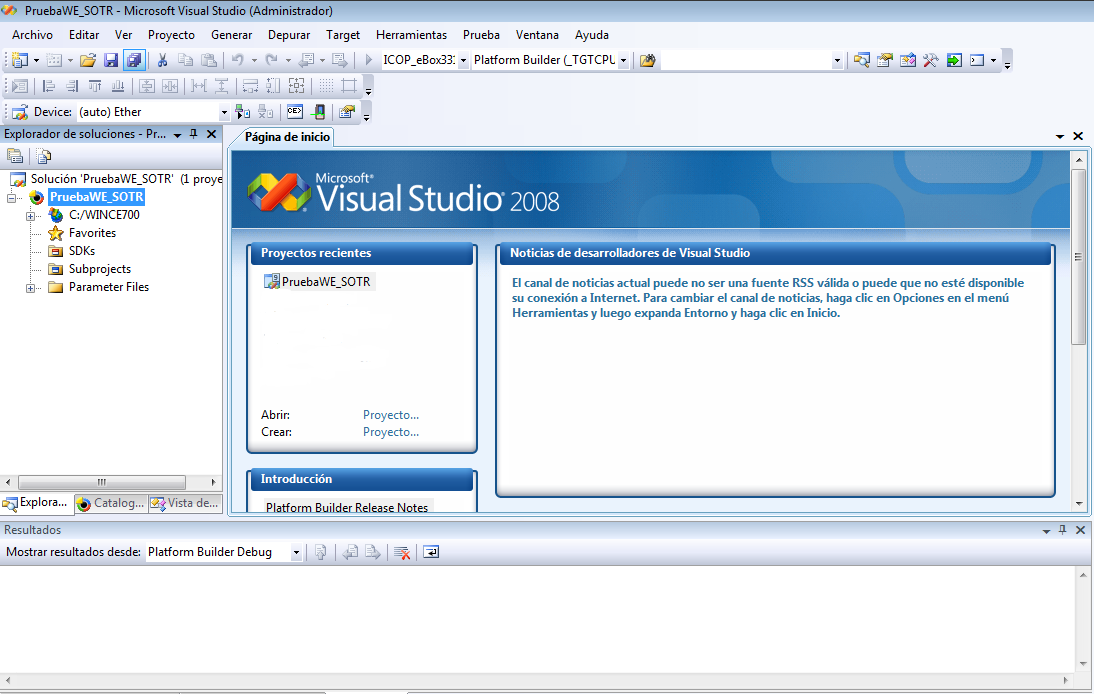


Figura 8. Pantalla de proyecto OS Design

## Configurar proyecto

Para configurar el proyecto anterior se accede a la vista de ítems por catálogo mediante el menú **“File | View | Other Windows | Catalog Items View”**  y mostrará la vista de la Figura 9, dicha vista contiene las aplicaciones, librerías, drivers, utilidades y componentes instalados en la estación de trabajo disponibles para agregar al proyecto.

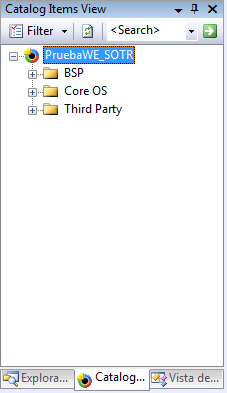


Figura 9. Vista de items por catálogo.

Se expande la carpeta “Third Party” y después la carpeta BSP y se verifica que estén seleccionados los siguientes ítems mostrados en la Figura 10.

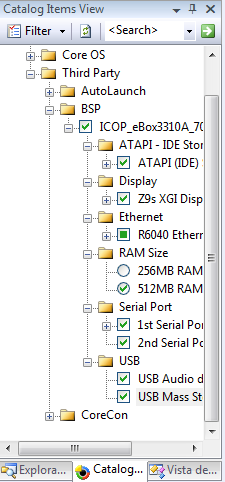


Figura 10. Componentes BSP

Se deben agregar los siguientes components expendiendo el nodo **\Core OS\Windows Embedded Compact**  para soportar mouse y teclado.

* \Device Drivers\Input Devices\Keyboard/Mouse\**8042 PS/2 Keyboard/Mouse**
* \Device Drivers\USB\USB Host\USB Class Drivers\**USB Human Input Device (HID) Class**

Seleccionar el component USB HID Keyboard and Mouse.

Del nodo \**Core OS\Windows Embedded Compact** se incluye el componente:

* \Applications-End User\**CAB File Installer/Uninstaller**

Del nodo \**Core OS\Windows Embedded Compact\Applications and Services**

**Development**  se selecciona:

* **.NET Compact Framework 3.5**
* **OS Dependencies for .NET Compact Framework 3.5**

Seleccionamos del nodo **\Core OS\ Windows Embedded Compact\Communication Services and Networking\Servers** los componentes**:**

* **FTP server**

Del nodo **\Third Party\CoreCon** seleccionamos:

* **CoreCon\_v100\_Compact7**

Expandimos el nodo **\Third Party\AutoLaunch** para agregar:

* **AutoLaunch\_v100\_x86\_Compact 7**

En la parte superior de la vista de ítems ingresamos “**Remote display**” y damos click en la fecha verde, cuando se muestren los componentes se seleccionan los elementos que se muestran en la Figura 11.

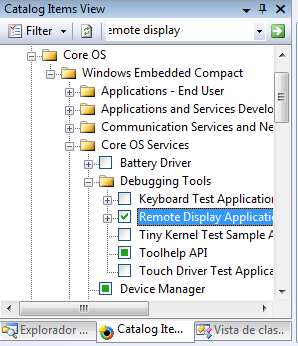


Figura 11. Componentes “Remote display”.

## Configuración del constructor

Hay 2 formas de construir la imagen del SO, una es crear una versión del SO en modo ejecutable para cargarlo al dispositivo y la otra es depurarlo, siendo esta última un 50% más grande en tamaño, para fines del ejemplo se configura como ejecutable (reléase).

Para ello desde el menú accedemos a **“Generar| Configuration Manager…”** y se desplegará la ventana del administrador de configuración como se muestra en la Figura 12, después seleccionamos la opción “**ICOP\_eBox3310A\_70CTP x86 Release”.**

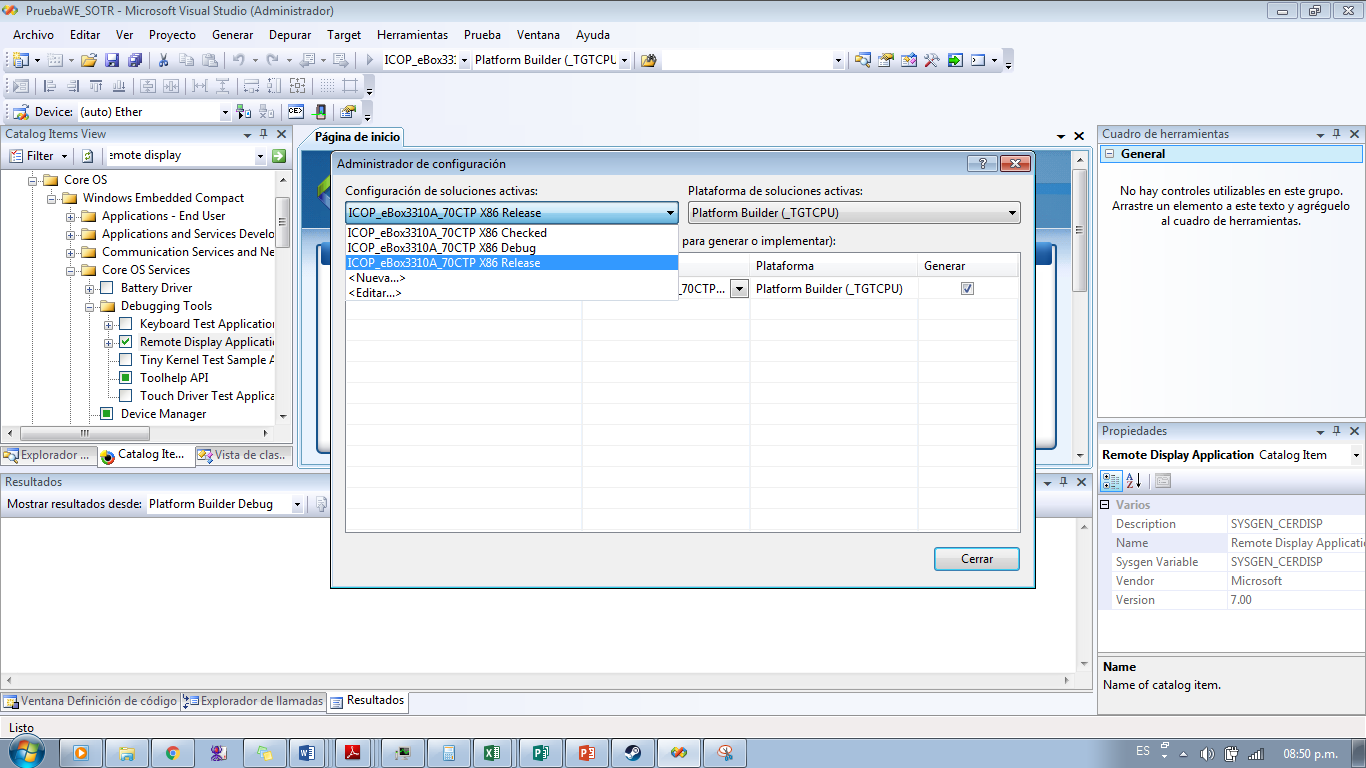


Figura 12. Administrador de configuración.

## Opciones del constructor.

Se accede a las opciones del constructor del proyecto desde el menú en “**Proyecto | Propiedades de PruebaWE\_SOTR…”**, siendo PruebaWE\_SOTR el nombre del proyecto, una vez se abra la ventana de propiedades, se expande las opciones del constructor y se modifican las siguientes opciones:

* **Enable eboot space in memory** - **No**
* **Enable KITL** - **No**

## Construir y generar la imagen del SO

Para iniciar el proceso de construcción de la imagen del SO basta con ingresar al menú y seguir la secuencia “**Generar | Advanced Build Commands | Sysgen (blddemo -q)”** y con ello dará inicio la creación de la imagen mostrando un mensaje similar al de la Figura 13; si todo se configuro de esta manera se generará la imagen del SO.

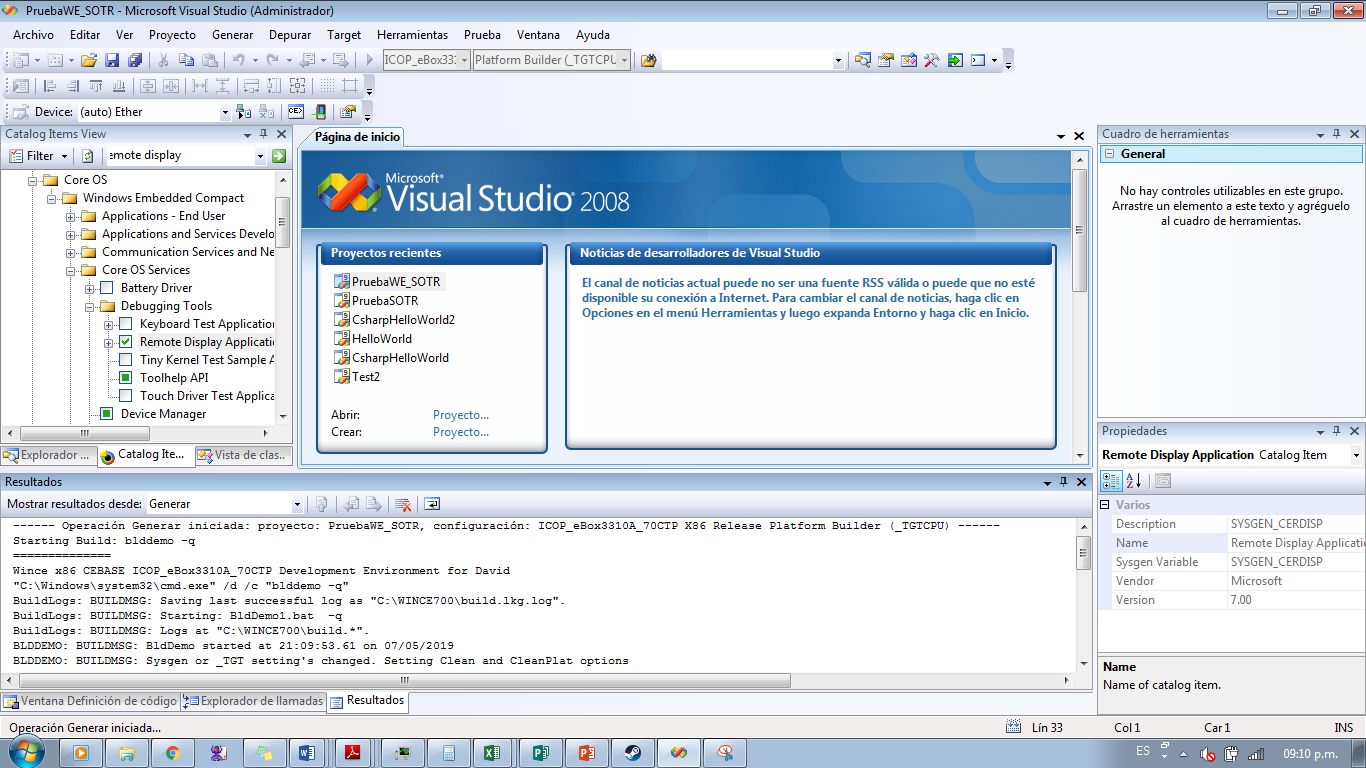


Figura 13.Mensaje de operación generar iniciada.

## Descarga de la imagen del SO a la plataforma (eBox)

Para descargar la imagen del SO a la eBox es necesario crear una conexión entre la estación de trabajo donde se creó el SO en Visual Studio 2008 y la plataforma, para ello se tienen las siguientes opciones:

* IP estática
* Conexión inalámbrica por router
* Ethernet
* Depuración por conectividad serial
* DCHP

Para el ejemplo se utilizó una conexión alámbrica por modem utilizando una IP estática, para ello fue necesario configurar la plataforma desde el símbolo del sistema (DOS), seleccionando la opción de cargar la imagen del SO desde la estación de trabajo con servicio DCHP.

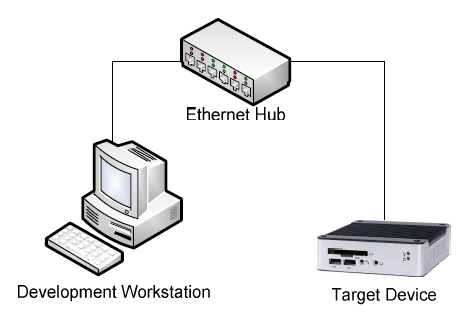


Figura 14. Conexión por servicio DCHP.

## Perfil de conexión

Se requiere crear un perfil de conexión en la computadora desde la que se cargará la imagen del SO para ello se accede desde el menú a “**Target | Connectivity Options**”, ello desplegará la ventana Target Device Connectivity Options similar a la mostrada en la Figura 15.

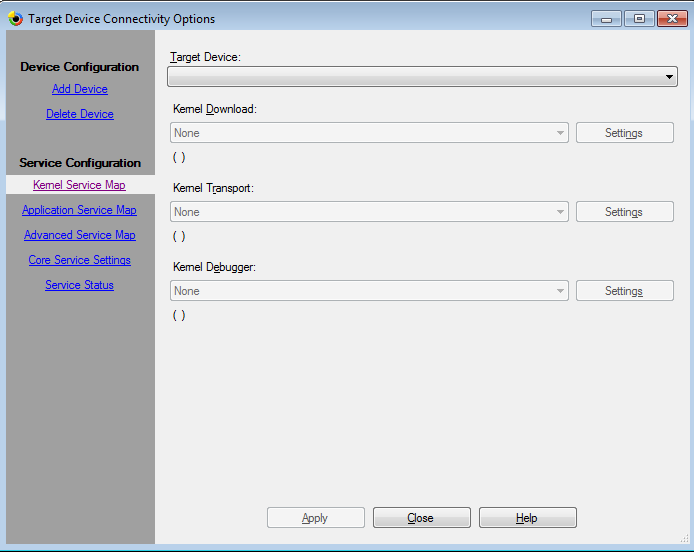


Figura 15. Target Device Connectivity Options.

* En la sección de configuración de dispositivo (Device Configuration) seleccionamos Agregar dispositivo (Add Device).
* Se agrega un nombre al dispositivo, no se selecciona SO y se da click en Add, véase Figura 16.

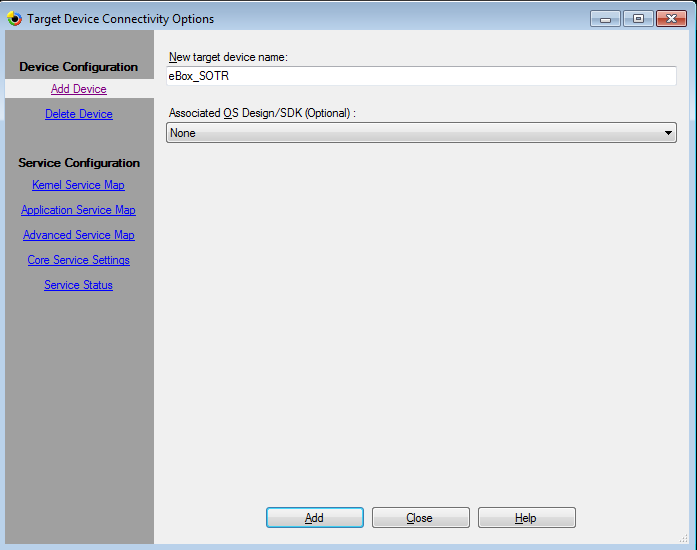


Figura 16. Agregar dispositivo.

Una vez disponible el dispositivo en el apartado de Kernel Service Map se realiza la siguiente configuración:

* **Ethernet** para la opción **Download**.
* **Ethernet** para la opción **Transport**.
* **KdStub** para la opción **Debugger**.
* Click en **Apply** para guardar la configuración.

## Asociar perfil de conexión con plataforma

Para asociar el perfil con la plataforma en que se cargara la imagen del SO, desde la pantalla de **Target Device Connectivity Options:**

* Se da click sobre las configuraciones (settings) de la descarga del kernel.
* Se enciende la plataforma (eBox) y selecciona la opción de carga de imagen por servicio DCHP.
* Una vez detectada por la estación de trabajo se mostrará una ventana similar a la Figura 17.

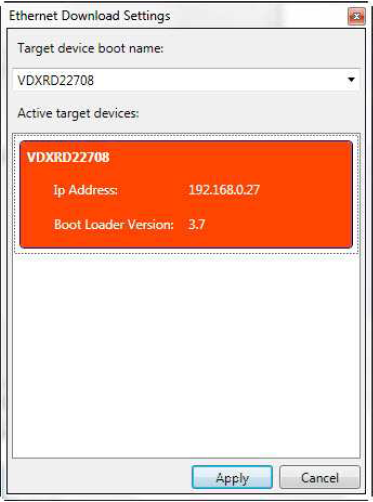


Figura 17. Configuración de descarga vía Ethernet

* Se selecciona el dispositivo en el que se cargará la imagen y se aplican los cambios.
* En la ventana de **Target Device Connectivity Options** se mostrará el dispositivo asociado al perfil de conexión, basta con aplicar los cambios y cerrar dicha ventana.

## Descargar imagen del SO al dispositivo

Con el perfil de conexión activo la descarga se realiza desde el menú mediante “**Target | Attatch Device**” y se desplegará la ventana de **Device Status** como en la Figura 18.



Figura 18. Device Status.

* Se reinicia la plataforma (eBox) y se selecciona la descarga de imagen por servicio DCHP para iniciar la descarga.
* En la pantalla conectada a la plataforma se mostrará una pantalla similar a la Figura 19



Figura 19. SO en plataforma.

## Ejemplo de aplicación

Para poder ejecutar aplicaciones en el SO precargado es necesario crear un SDK en el proyecto desde el que se descargó el SO, para ello desde el menú de Visual Studio 2008 “**Proyecto | Agregar nuevo SDK**”, con ello se desplegará la ventana de propiedades del SDK.

* Agregamos un nombre para el SDK en las opciones generales.
* Después en la sección install, agregamos el nombre del SDK con la extensión “**.msi**”.
* En Development Languages, habilitamos la opción “Managed development support”.
* Aplicamos los cambios.

Si se desea compilar basta acceder de “**Build | Build All SDKs**”.

Aplicación nativa

Como ejemplo para desarrollar una aplicación nativa creamos un nuevo proyecto desde “**Archivo | Nuevo | Proyecto …**”.

* De las plantillas de Visual C++ seleccionamos **Smart Device** como tipo de proyecto.
* Seleccionamos **Win32 Smart Device Project**, como en la Figura 20**.**

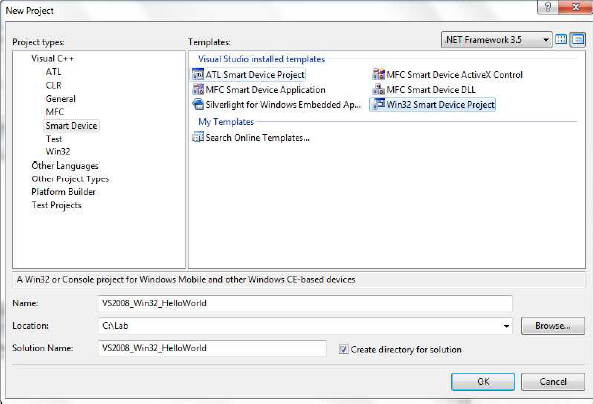


Figura 20. Proyecto Visual C++.

* Agregamos nombre al proyecto y damos click en aceptar para continuar con el asistente.

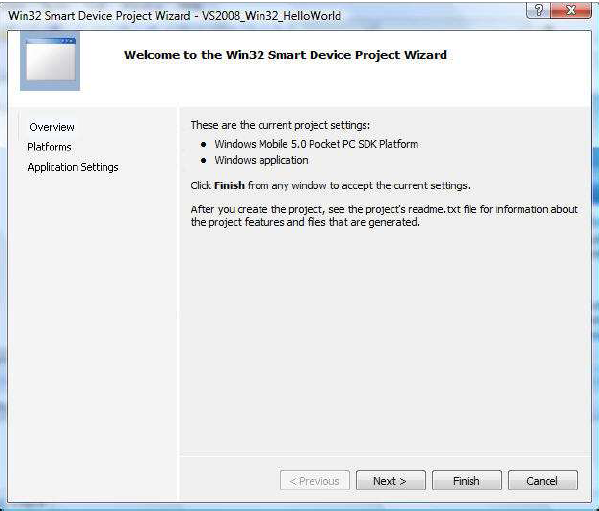


Figura 21. Asistente para Smart Device.

* En la ventana de plataformas retiramos la opción por default y agregamos el SDK que se creó anteriormente.

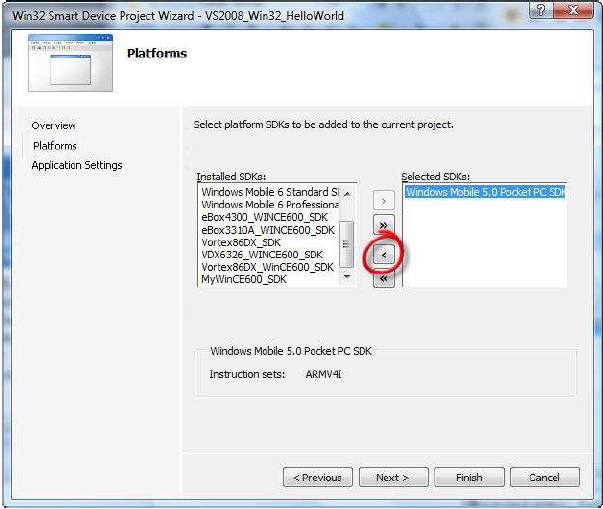


Figura 22. Ventana de plataformas del asistente de Smart Device.

* En la ventana de configuración del proyecto dejamos los valores por defecto y como tipo de aplicación “**Windows application**” y finalizamos.

Una vez abierto el proyecto:

* En el navegador de archivos buscamos el archivo que tenga el nombre de nuestro proyecto con extensión “.cpp”
* Reemplazamos el código contenido en la cláusula “case WM\_PAINT:” por el código de la :

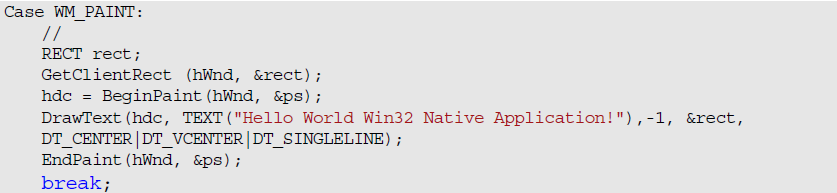


Figura 23. Código de ejemplo Visual C++.

* Se compila el proyecto desde “**Build | Build Solution**”.

Para descargar la aplicación al SO, es necesario establecer la comunicación mediante CoreCon, una vez cargado el SO a la plataforma, desde la misma plataforma:

* Click en **Start | Run** y abrimos la línea de comandos mediante **CMD**.
* Ejecutamos el comando **IPConfig** y se desplegará una ventana como en la Figura 24.

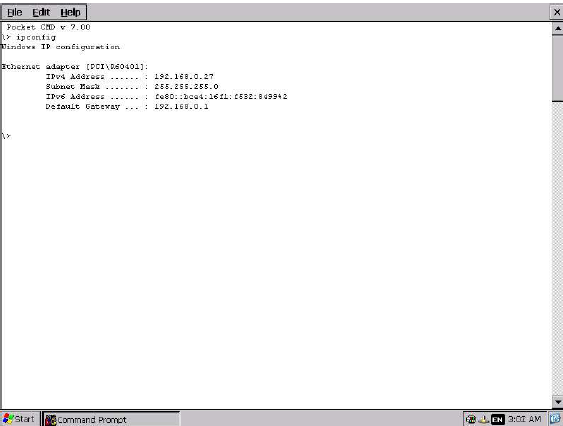


Figura 24. Detalles de IP

Desde Visual Studio 2008:

* Ingresamos a “**Herramientas | opciones…**”.
* Expandimos las opciones de “**Device Tools | Devices**” como en la Figura 25.

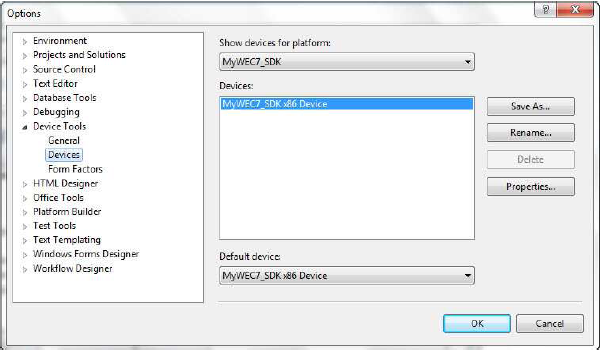


Figura 25. Opciones de dispositivos.

* Hacemos click en el botón de propiedades, teniendo seleccionado el SDK que se creo anteriormente.
* En la ventana de propiedades damos click en el botón de configuración de Transport.
* Seleccionamos la opción de usar dirección IP específica e ingresamos la IP de la plataforma.
* Damos click en aceptar hasta volver a la pantalla principal del proyecto.
* Ingresamos a “**Tools | Connect to Device**” desde el menú.
* Seleccionamos el SDK como en la Figura 26 .

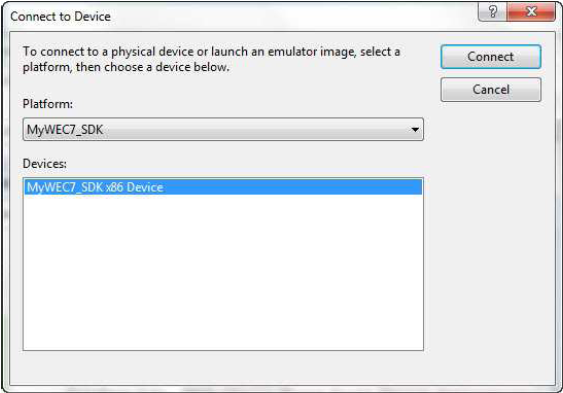


Figura 26. Conectar a dispositivo.

* Click en conectar y esperar el mensaje “**Connection succeeded**” y dar click en cerrar.
* Para iniciar la depuración, desde el menú se accede a “**Debug | Start Debugging**”

En la pantalla de la plataforma se deberá mostrar la aplicación como se muestra en Figura 27.

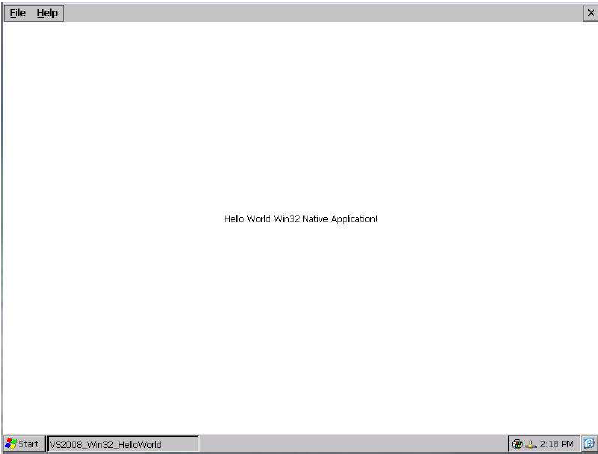


Figura 27. Ejemplo de aplicación nativa.

## Conclusiones

Windows Embedded cumple con varios de los requisitos de un sistema operativo en tiempo real, tiene como ventajas la flexibilidad de código, la facilidad de realizar tareas y monitorear de manera remota la plataforma. Sin embargo su aplicación a plataformas únicamente desarrolladas por Microsoft impide la compatibilidad con otros sistemas, por lo que dado el caso sería sustituir un sistema existente, además de que la licencia de este sistema es de paga salvo que sea para fines educativos donde se tienen descuentos.