Nombre alumno: José Antonio Esparza Isasa
Titulación: Ingeniería en Informática
Curso académico: 2007 - 2008

1.TÍTULO DEL PROYECTO

Plataforma de telecontrol sobre Linux embebido en arquitecturas AVR32

2.DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

Las nuevas tecnologías permiten hoy en día no solo automatizar procesos industriales sino dar un paso más allá y poder telecontrolarlos. Estos procesos industriales pueden ser de muy diversos tipos, desde cadenas de montaje, gestión de turbinas que producen electricidad, hasta grandes extensiones en las que hay que regular el proceso de riego.

El abaratamiento de los costes de producción de hardware electrónico hace que conforme pasa el tiempo se cuenten con dispositivos de grandes prestaciones a precios cada vez más asequibles, lo que en combinación con el creciente desarrollo de la informática hace que cada día podamos encontrar pequeños ordenadores en multitud de situaciones.
Estos pequeños equipos concebidos para llevar a cabo una tarea particular, a menudo compleja o repetitiva (o ambas cosas), son conocidos como sistemas embebidos. Muy habitualmente los sistemas embebidos realizan tareas de telecontrol.

MACRAUT ingenieros es una empresa dedicada a la automatización y telegestión de sistemas y procesos. Esta empresa diseña equipos que son de aplicación en numerosos campos (automatización de procesos industriales, generación de energía, agua y ambiente, sector naval, procesos de riego, etc.) y cuenta con vastas redes de procesos telecontrolados a lo largo del territorio nacional. Estos procesos son controlados por equipos modulares y jerárquicos. Una de las piezas clave en estos equipos son los sistemas LINX, encargados de las comunicaciones y a los que se conectan equipos de menor peso en la red.

Las aplicaciones a desarrollar en este proyecto se plantean como la modernización software de los equipos LINX utilizados por MACRAUT en el telecontrol de regadíos.

Este proyecto es la continuación del trabajo presentado durante el periodo de prácticas en empresa, realizado en el verano del 2008 en la empresa MACRAUT ingenieros. Durante este tiempo se desarrollaron librerías básicas que serán necesarias en los futuros módulos del proyecto.

3.OBJETIVOS DEL PROYECTO

El sistema LINX actual está controlado por un micro de la empresa BenQ y corre un RTOS (Sistema Operativo en Tiempo Real) propietario de la misma casa. La programación de las rutinas de control se llevan a cabo a través de la API suministrada por el fabricante del micro.

Con la expansión del movimiento del software libre a otros campos de la informática que van más allá de los PC's de sobremesa, podemos encontrar sistemas operativos como Linux en sistemas embebidos. Ejemplo de ello son las versiones de Linux que se pueden correr sobre los micros AVR32 de atmel.

En este proyecto se tratará de llevar a cabo el diseño de la arquitectura software y parte de su implementación para modernizar los sistemas LINX y utilizar micros AVR de 32 bits corriendo LINUX optimizado para las tareas que controle el sistema.

4.METODOLOGÍA

Metodología de desarrollo

Se utilizará un proceso de desarrollo en espiral. Esta metodología de trabajo se caracteriza especialmente por ser iterativa y seguir una estructura claramente definida en cada iteración, compuesta por:

1. Determinación de los objetivos

2. Análisis, Identificación y resolución de los riesgos

3. Construcción y test
    - La fase de construcción incluye la generación de la documentación correspondiente al trabajo realizado en esta iteración.
4. Planificación de la siguiente iteración
    - Se planificarán los objetivos para la siguiente iteración, programando los que hayan quedado pendientes en la actual para la siguiente.

Aunque originalmente esta metodología de desarrollo se diseño para realizar iteraciones de 6 meses a 2 años, como veremos a continuación, esto cambiará en este proyecto.

¿Por qué una metodología iterativa e incremental?
La razón principal para la utilización de una metodología de estas características es que se puedan añadir características al proyecto de manera incremental y que se puedan realizar entregas parciales previas a la finalización del software. Esto permitirá ir realizando validaciones y verificaciones de estas partes y llevar un control del producto que se está desarrollando.

Reuniones y entrega de resultados

Con el tutor del proyecto en la universidad

Después de cada iteración se mantendrá una reunión con el tutor del proyecto para presentar los siguientes puntos:

- Aprobación del acta de la reunión anterior
- Resultados de la iteración finalizada
- Problemas encontrados
- Porcentaje completado del proyecto
- Presentación de objetivos de la siguiente iteración

Con el tutor de la empresa

Tras finalizar cada iteración se enviará por correo electrónico la documentación generada y se actualizará el repositorio de código con las versiones actualizadas del software desarrollado.

La documentación a entregar comprende:

- Documentación de las pruebas realizadas
- Documentación sobre la generación de los binarios

5.PLANIFICACIÓN DE TAREAS

Iteraciones que componen el proyecto

El proyecto se ha planificado para ser ejecutado en iteraciones de 2 semanas de duración. En cada iteración la espiral que modela el desarrollo "realizará un ciclo" pasando por todas las fases que le corresponden.

El proceso de desarrollo elegido es un proceso ágil y ligero, que tiene como objetivo que la salida de una iteración sea un sistema ejecutable pero incompleto. Un subconjunto con calidad de producción del sistema final.

En las primeras iteraciones se abordarán las cuestiones más costosas y de mayor riesgo para el proyecto.

Un ejemplo de planificación para la iteración número 0 sería el siguiente:

Iteración 0	Introducción al proyecto: situación inicial
Tareas a realizar	• Redacción de la propuesta de proyecto • Elaboración del documento de visión • Primera versión del documento SRS

Conforme se vayan ejecutando y planificando las siguiente iteraciones este documento se irá completando.

Localización temporal de las iteraciones

Noviembre	Comienzo del proyecto
Semana 1	Iteración 1
Semana 2	Iteración 1
Semana 3	Iteración 2
Semana 4	Iteración 2

Diciembre	
Semana 1	Iteración 3
Semana 2	Iteración 3
Semana 3	No disponible: preparación exámenes 1º cuatrimestre
Semana 4	

Enero	
Semana 1	No disponible: preparación exámenes 1º cuatrimestre
Semana 2	
Semana 3	
Semana 4	

Febrero	
Semana 1	No disponible: preparación exámenes 1º cuatrimestre
Semana 2	
Semana 3	Iteración 4
Semana 4	Iteración 4

Marzo	
Semana 1	Iteración 5
Semana 2	Iteración 5
Semana 3	Iteración 6
Semana 4	Iteración 6

Abril	
Semana 1	Iteración 7
Semana 2	Iteración 7
Semana 3	Iteración 8
Semana 4	Iteración 8

Mayo	
Semana 1	Iteración 9
Semana 2	Iteración 9
Semana 3	
Semana 4	

Junio	Defensa del proyecto. Fecha preliminar 30 de Junio.
Semana 1	

Semana 2	No disponible: preparación exámenes 2º cuatrimestre
Semana 3	
Semana 4	

6.OBSERVACIONES ADICIONALES

Riesgos que afectan al proyecto

A continuación se estudian los principales riesgos que afectan al proyecto y que pueden causar el retraso de su defensa en la fecha prefijada (30 de junio).

Riesgo	Perdida de datos en el ordenador de desarrollo
Probabilidad	Media
Gravedad	Alta
Causas	Diversas
Mecanismos de prevención	• Actualización del equipo. • Firewall • Uso exclusivo para el proyecto
Mecanismos de contención	• Copias de seguridad semanales • Actualización del repositorio con frecuencia • Almacenar documentación e información importante en el servidor ftp de la empresa

Riesgo	Perdida de datos en el servidor de MACRAUT
Probabilidad	Baja
Gravedad	Alta
Causas	Diversas
Mecanismos de prevención	Fuera del ámbito de mi proyecto
Mecanismos de contención	• Copias en un disco duro local guardado en el laboratorio

Riesgo	Imposibilidad de acceso al servidor de MACRAUT
Probabilidad	Media
Gravedad	Media
Causas	Error de administración, problemas en la conexión a internet
Mecanismos de prevención	Fuera del ámbito de mi proyecto
Mecanismos de contención	• Copias en un disco duro local guardado en el laboratorio

Riesgo	Daño físico en la placa de desarrollo
Probabilidad	Media
Gravedad	Baja. Sólo la pérdida material
Causas	Diversas
Mecanismos de prevención	• Evitar el contacto con los circuitos integrados, cuidado con la electricidad estática. • Comprobación polaridad. • Evitar esfuerzos mecánicos.
Mecanismos de contención	Se dispone de una segunda placa en el laboratorio

Riesgo	Falta de tiempo por periodos de exámenes
Probabilidad	Alta
Gravedad	Media
Causas	Exámenes de asignaturas y otros trabajos
Mecanismos de prevención	• Organización y cumplimiento de horarios y objetivos de las iteraciones. Previsión.
Mecanismos de contención	• Reorganización de las tareas para la próxima iteración y empleo de mas horas en los días/semanas sucesivos

Material necesario

A continuación se presenta una lista con los materiales necesarios para el desarrollo del proyecto, tanto hardware como software y de otro tipo.

Hardware

- Placa de prototipos STK1000
- Placa de prototipos NGW100
- Ordenador portátil
- Swicth de 5 bocas de red como mínimo.
- Placa auxiliar con capacidad para manejar el bus I2C.
- Circuito adaptador USB a I2C
- 4 cables de red no cruzados con conectores RJ45
- 2 Tarjetas SD 2 Gb
- Programador JTAG
- Cable serie
- Fuente de Alimentación: 12 Voltios 600mA
- Modem GPRS

Software

- Ubuntu Linux Desktop edition 8.04 (Instalado en el equipo de desarrollo)
- Microsoft Windows XP
- AVR32 Studio: IDE basado en eclipse para el desarrollo en C/C++ orientado a plataformas atmel. Disponible bajo Linux y Windows.
- AVR32 toolchain: Conjunto de herramientas (compilador, linker, debugger) que permiten llevar a cabo la compilación cruzada del código fuente escrito en un PC ordinario, generando binarios para los micros AVR32. Disponible para Linux y Windows.
- Programa para lectura/escritura de datos en el puerto serie. Bajo el entorno Windows se utilizará el ya incluido por defecto Hyperterminal y bajo Linux minicom.
- Java Development Kit: Kit de desarrollo de java, necesario para poder utilizar AVR32 Studio
- Java Runtime Environment: Máquina virtual de java, necesario para poder utilizar AVR32 Studio
- Imagen de Ubuntu Linux 7.10 de Atmel: distribución linux proporcionada por Atmel en sus cursos de formación. Contiene todas las aplicaciones y un entorno listo para comenzar el desarrollo sobre micros AVR32.
- VMWare player: de libre distribución en la página oficial.
- Microsoft Word 2007
- Microsoft Power Point 2007
- Microsoft Project 2007
- Microsoft Visio 2007
- Dia 0.97: programa para la generación de gráficos.
- Cliente socket desarrollado por MACRAUT: Código fuente y binarios. Será necesario a la hora de realizar las pruebas del servidor socket a implementar en la placa.
- Doxygen: programa para la generación de documentación a partir del código fuente
- Doxy-widard: fronend para doxygen
- X-MAN: frontend basado en X11 para la presentación de las páginas del manual

Documentación

- man-pages: disponibles en los repositorios estándar de Ubuntu
- developer-man-pages: disponibles en los repositorios estándar de Ubuntu

Otros medios

- Conexión a Internet: es necesario que sea por cable para poder conectar el switch al que comunica el equipo de desarrollo y las placas de prototipos.
- Armario para guardar material
- Mesa de trabajo en el laboratorio

7. Recursos consultados

Libros de texto:

- Larman, Craig. "UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado". Pearson, Prentice Hall. 2ª Edición. 2002 ISBN 9788420534381

Páginas web:

- http://www.macraut.com/: Página de la empresa MACRAUT Ingenieros.