ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Especificación y Diseño en UML del Control Teleoperado de un Robot e Implementación en una PDA usando Java Micro Edition

AUTOR: Violeta Martín Lorente
DIRECTOR: Pedro Sánchez Palma
CODIRECTOR: Juan A.Pastor Franco
Enero / 2005

Autor	Violeta Martín Lorente
E-mail del Autor	violeta.martin@gmail.com
Director(es)	Pedro Sánchez Palma
E-mail del Director	Pedro.Sanchez@upct.es
Codirector(es)	Juan A. Pastor Franco
Título del PFC	Especificación y Diseño en UML del Control Teleoperado de un Robot e Implementación en una PDA Usando Java MicroEdition
Descriptores	

Resúmen

El proyecto desarrollado aborda la construcción de un sistema para el control teleoperado de un robot. Para la realización de este sistema se han llevado a cabo una serie de estudios referentes a la infraestructura física utilizada y a la plataforma de programación empleada para la implementación y ejecución de la aplicación. El sistema de control ha sido especificado y diseñado según la notación UML. Respecto a la infraestructura física, destacar el uso de un dispositivo tipo PDA cuyas características, ventajas e inconvenientes han sido expuestos. Como plataforma de programación se ha empleado el estándar Java2 MicroEdition, habiéndose realizado un análisis sobre las diferencias de éste respecto a las otras plataformas Java2 y el por qué de su conveniencia a la hora de programar sobre dispositivos de recursos limitados.

Titulación	Ingeniería Tecnica de Telecomunicación, Especialidad Telemática
Intensificación	
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Enero- 2005

Índice

Capítulo 1 Introducción

Capítulo 2 Dominio de la Aplicación

Capítulo 3 Infraestructura Utilizada

Capítulo 5 Requisitos y Diseño de la Aplicación

Capítulo 6 Implementación de la Aplicación

Capítulo 1
Introducción

1.1 Agradecimientos

Mis agradecimientos al Departamento de Tecnologías de la Información y las Comunicaciones por la beca de colaboración que me ha sido concedida para la realización de este proyecto, y en especial al Doctor Pedro Sánchez Palma por la dirección del mismo.

1.2 Introducción

La realización de este proyecto fin de carrera surge dentro del marco de trabajo del sistema EFTCoR (Environmental Friendly and cost-effective Technology for Coating Renoval), tecnología respetuosa con el Medio Ambiente y de bajo coste para la limpieza de superficies de barcos de gran calado. El sector implicado en el sistema es el de servicios de reparación y mantenimiento naval, y el marco de trabajo es el Programa Marco de la Unión Europea. El espacio temporal asignado para el desarrollo del sistema EFTCoR va desde 1 de Octubre del 2002 a 1 de Octubre del 2005.

El líder del Proyecto es IZAR, habiendo participantes de siete países desempeñando distintas tareas para la construcción del producto, como el estudio de materiales de limpieza, etc. El Grupo de Investigación DSIE de la UPCT es uno de los participantes tomando el rol del diseño y construcción de la unidad de control del robot de limpieza.

El propósito del proyecto EFTCoR es solventar un problema que está siendo crítico en la industria de construcción de barcos europea: desarrollo y suministro de equipos de chorreado, desarrollo y suministro de equipos de pintura, eliminación de residuos, construcción y mantenimiento de barcos sostenible y competitiva, concerniente al proceso de preparación de la superficie del casco antes del proceso de pintura.

Además de la primera preparación de la superficie del casco cuando el buque es construído -antes de pintarlo-, el principal mantenimiento consiste en arrancar periódicamente (cada 4-5 años) las adherencias marinas que cubren el casco y luego repintar el casco. Esto es realizado para preservar la integridad del casco, garantizando unas condiciones de navegación seguras, y el mantenimiento de una superficie del casco lisa, minimizando el consumo de combustible, reduciendo los costes de operación y evitando el exceso de contaminación atmosférica. Otras operaciones de mantenimiento son programadas o hasta retrasadas para que sean hechas mientras se limpia y repinta el casco. La tecnología actual para la limpieza de cascos, chorreado con granalla, es muy contaminante, y está siendo progresivamente prohibida en la mayoría de los países sensibles con el medioambiente (principalmente el norte de Europa), manteniéndose únicamente en los países del sur (Grecia, Portugal, España) con una clara tendencia a ser reducida hasta ser definitivamente prohibida.

En este momento, la metodología ha sido parcialmente sustituida por chorreado de agua a alta presión. Esos sistemas evitan la limpieza pre-agua requerida para la desalinización del casco usada con chorreado con granalla; sin embargo, como informan los suministradores de pintura y propietarios de barcos, no muestran tan buen rendimiento como el sistema de chorreado de granalla.

El objetivo principal del proyecto EFTCoR es el desarrollo de una tecnología fiable y de bajo coste para la limpieza de cascos de buques capaz de obtener una superficie preparada de alta calidad con una drástica reducción de los residuos y una nula emisión al medioambiente. Para alcanzar este objetivo, será necesario desarrollar:

- Nuevas tecnologías de limpieza y materiales.
- Nuevos cabezales de limpieza.
- Sistemas robotizados necesarios para cubrir una superficie que garantice una limpieza eficiente.
- Un sistema original de recogida y reciclaje de residuos.
- Sistemas de visión para verificación on-line.
- Nuevas pinturas.

El sistema EFTCoR es una plataforma tele-operada altamente ambiciosa para operaciones no contaminantes de mantenimiento de cascos de buques en diferentes áreas de trabajo: desde largos diques secos libres de obstáculos, hasta áreas llenas de dificultades de acceso. Ofrece una solución integral para la preparación de cascos de barcos conforme con requisitos de eficiencia, seguridad en el trabajo y cuidado con el medio.

El sistema de chorreo definido es una nueva tecnología industrial para la supresión de capas del casco en el mantenimiento de barcos. De este modo se ataca un problema que esta convirtiéndose en crítico para la industria europea naval. Los usuarios de este sistema verán derivarse beneficios económicos gracias a la reducción de costes de consumo de material, eliminación de residuos y mano de obra. Con el sistema EFTCoR, el consumo del material de capas eliminadas puede ser reducido hasta un 90 por ciento.

La mayoría de las operaciones de la reparación del astillero no se automatizan hoy día. Hay un número de tareas que forman parte de las operaciones de mantenimiento tales como el chorreado del casco que son muy difícil de automatizar. Hay soluciones parciales como las turbinas de chorreado y unidades de chorreado del agua a ultra-presión para las paredes verticales, pero no cumplen los requisitos buscados. Este problema se afronta mediante un desarrollo por incrementos del sistema EFTCoR.

La siguiente figura nos adelanta una visión general del funcionamiento final del sistema. Como se muestra en la figura 1 hay varios subsistemas que interactúan entre sí.

- La unidad de control para los dispositivos robóticos
- El sistema de monitorizado
- El sistema de visión
- El sistema de reciclado

El **Sistema de Visión** implementa un mecanismo de visión artificial mediante el cual reconoce las partes dañadas del casco y de este modo decide cuáles deben ser tratadas. Versiones posteriores del sistema de visión controlarán la operación de chorreado mediante el cálculo de la trayectoria del robot; además, comprobarán los resultados de la calidad del chorreado. El Sistema de Visión también proveerá una guía de planificación en línea para el trabajo localizado y valoración de la superficie tratada, implementando un control de calidad. El sistema estará comunicado con el subsistema de control del robot y el sistema de monitorización.

El **Sistema de Monitorización**. Este sistema es externo al sistema de tele-operación. El sistema de monitorización considera un listado de información para cada tarea de

limpieza. El sistema se comunicará con el sistema tele-operado mediante tecnología *wireless*.

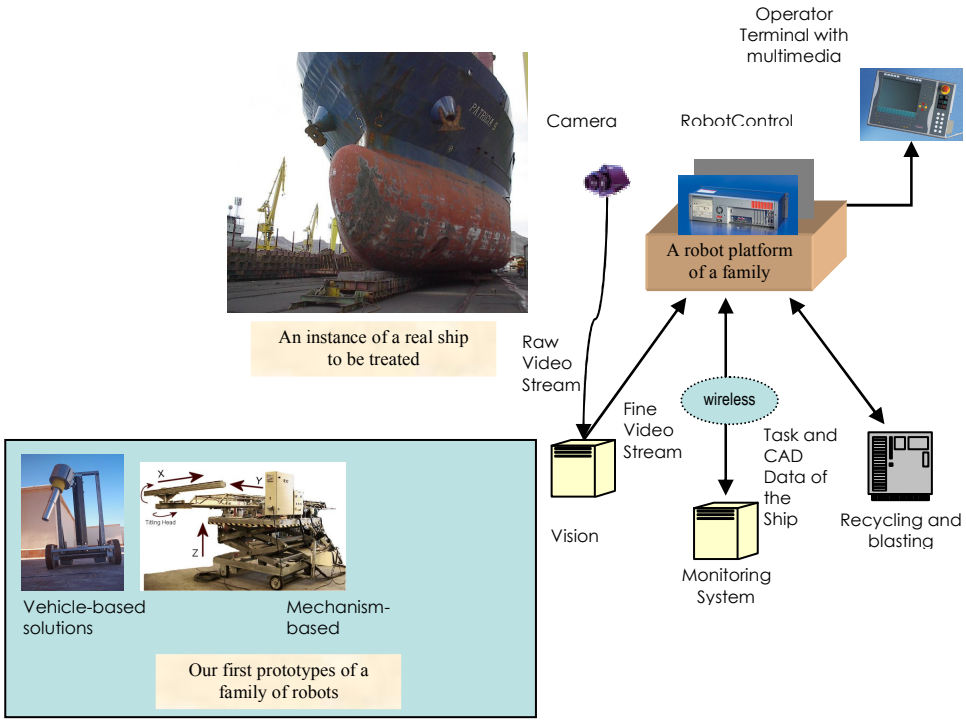


Figura 1. Aspecto final del sistema EFTCoR

El propósito del **Sistema de Posicionamiento** es mover la cabeza de limpiado acercándola o alejándola de la superficie del casco. El sistema de posicionamiento debe comprender dos sistemas, primario y secundario, de posicionamiento: el sistema primario de posicionamiento puede ser una larga grúa para alcanzar todas las áreas del casco, y el sistema secundario de posicionamiento debe ser un robot para cubrir áreas de alrededor de 3 m^2 (más o menos el área que un operador humano puede limpiar desde una posición dada).

El **Sistema de limpieza** está compuesto por tres partes: la cabeza de chorreo (unidad de inyección), la unidad de aspirado y la cabeza de limpieza. La cabeza de limpieza consiste en un cabezal de inclinación ajustable que guía la manguera de chorreo y permite que el ángulo de incidencia, presión de aire y alimentación de arena sean ajustados para controlar la operación de chorreo. El conjunto es adherido mediante una cubierta especialmente diseñada y colocada alrededor de las cabezas de chorreo para sellar las unidades a la superficie que está siendo limpiada, previniendo emisiones de polvo y residuos. Un contacto flexible entre el sellado y el casco es alcanzado por el uso combinado de fuentes de aire regulables. El manejo de deshechos y el sistema de reciclado elimina los residuos producidos por el chorreado del casco, permitiendo una reutilización del material arenoso y el correcto empaquetado de otros residuos.

El desarrollo global del proyecto ha sido dividido en diferentes paquetes de trabajo que han sido distribuidos entre las distintas entidades participantes. Los paquetes “Unidad de Control Remota” y “Sistema de Visión” han sido asignados a la Universidad

Politécnica de Cartagena, y es este el contexto en el que se centra el proyecto aquí expuesto.

La tarea asignada al paquete de trabajo "Unidad de Control Remota" es desarrollar un control de los dispositivos robóticos, posicionando sistemas y herramientas usadas en las tareas de mantenimiento. El desarrollo tiene dos dimensiones: la mecánica del sistema robótico y el hardware y software para controlar las operaciones.

Debido al tamaño de la cabeza de limpiado y la necesidad del operador de ser capaz de ver la superficie, una o más cámaras de vídeo deben de ser incorporadas para la captura de vídeo en tiempo real de los puntos a chorrear. Estas imágenes pueden ser mostradas en un terminal PDA que el operador usa para interactuar con el sistema tele-operado. De este modo, no hay una automatización de la tarea de limpiado. El operador puede usar el terminal PDA para operar el sistema de limpieza. Un enlace de comunicación conecta el sistema tele-operado con un sistema remoto de monitorización para el intercambio de información acerca de las tareas de limpieza.

1.3 Objetivos del Proyecto Fin de Carrera

El caso de estudio de este proyecto gira entorno a dos ejes. En primer lugar se trata de aportar al sistema EFTCoR un valor añadido mediante la inserción en el sistema de una PDA y el desarrollo de una aplicación que debe de ejecutarse sobre la misma. Este dispositivo debe representar tanto un enlace para comunicación entre subsistemas como una herramienta de monitorización.

Lo que se pretende es diseñar el prototipo de una aplicación que sirva de soporte al operador humano que realice las tareas de colocación de una parte del robot, y limpieza. También se pretende ampliar las posibilidades de comunicación del operario con el sistema de control. Dicha aplicación se ha dividido en diferentes objetivos parciales:

- **Vídeo en tiempo real**. El sistema de visión dispone de una o más cámaras que captan las imágenes del casco del barco. Mostrando estas imágenes sobre la pantalla del terminal PDA, el operador puede ver en todo momento cómo evolucionan las tareas de chorreado y limpieza. El vídeo pues, puede ser entendido por el operario como una guía adicional para realizar sus tareas de limpieza o preparación del casco.

- **Recepción de los valores de los sensores**. Unos sensores colocados en las extremidades del robot miden la distancia de éstas respecto a la superficie del casco. Debido a la distancia que puede llegar a haber entre el operador y la superficie del casco, es de gran utilidad que el operario pueda recibir en la PDA el valor de los sensores y de este modo saber con exactitud los movimientos que debe de hacer con el robot. El valor de los sensores puede ser representado de manera gráfica de modo que la interfaz sea más agradable y fácil para el operario. De este modo también podemos activar mecanismos visuales que alerten al operario cuando la cabeza del robot se encuentre a muy poca distancia de la superficie. La activación de la alerta se basa en un parámetro programable que define cuál es la distancia a partir de la cual el operario debe prestar más atención, o realizar los desplazamientos del cabezal de chorreado de manera más precisa. Siendo este parámetro programable permitimos que la activación de la señal pueda ser modificada en función del entorno o las necesidades específicas de cada tarea.

- **Envío de comandos al Sistema de Control**. A través de la PDA el operario puede comunicarse con el sistema de control para indicar las acciones que deben de llevarse a cabo o enviar notificaciones sobre las tareas que está realizando. Las acciones que provoquen dichos comandos son transparentes a la aplicación implementada sobre la PDA.

- **Comunicación inalámbrica del terminal**. Por diferentes razones como son facilidad de uso, movilidad y eficiencia del dispositivo se pretende que tanto el envío de comandos como la recepción de los valores de los sensores pueda realizarse de modo inalámbrico. Para ello, se pretende hacer uso de algún protocolo *wireless* que soporte la PDA. Se estudiarán las diferencias entre los distintos protocolos *wíreless* soportados por el dispositivo, escogiendo el más adecuado, el hardware necesario en el lado del servidor para realizar la comunicación inalámbrica, y los conceptos básicos de la conexión de dispositivos mediante esta tecnología.

En los capítulos posteriores se expone tanto la infraestructura hardware adquirida (justificando la adquisición de cada uno de los componentes), así como el diseño de la aplicación software. La plataforma utilizada en el desarrollo del proyecto, dispositivo tipo PDA, tiene diversas peculiaridades que hacen que el trabajo con la misma difiera del trabajo sobre plataformas habituales tipo ordenador personal. Antes de empezar a diseñar la aplicación software se ha de realizar un estudio completo de los dispositivos sobre los que se va a ejecutar dicha aplicación. Respecto al terminal PDA, han de observarse tanto las limitaciones del sistema operativo que posee como sus características físicas que determinan en gran medida aspectos tales como sus posibilidades de comunicación con otros dispositivos, el peso en cuanto a necesidades de memoria que puede tener la aplicación software y otros factores que influyen en el funcionamiento del sistema. También han de determinarse las necesidades que pueda tener un determinado ordenador personal para que pueda ser usado como servidor en la realización de pruebas de la aplicación. Principalmente se trata de adquirir el hardware necesario para posibilitar la comunicación inalámbrica entre la PDA y el ordenador personal.

Una vez analizados los dispositivos físicos que intervienen en el subsistema, así como el software que funciona sobre ellos, se pasa a un análisis de la aplicación software. Para ellos se ha empleado la notación de los diagramas de casos de uso de UML, así como otros diagramas basados en el mismo estándar. Los casos de uso son usados para definir los requisitos externos funcionales del sistema desde la perspectiva del usuario. Con el uso de esta notación se dejan bien definidos los requisitos de la aplicación así como todas las situaciones que pueden darse durante el funcionamiento de la misma.

Por razones como la portabilidad se ha elegido Java como lenguaje para desarrollar el software de la aplicación. De aquí se deriva el segundo objetivo del proyecto. Se trata del estudio de las plataformas Java para dispositivos empotrados y móviles: *PersonalJava* y J2ME. La plataforma general de Java está actualmente dividida o compuesta por diferentes estándares. Las tres principales plataformas Java son: *Java2StandarEdition*, J*ava2EnterpriseEdition* y J*ava2MicroEdition*. Cada una de ellas esta enfocada a unos determinados usos y son adecuadas para diferentes familias de dispositivos. Cada edición alberga las necesidades Java de una serie particular de aplicaciones, y amplía o restringe las posibilidades del lenguaje.

Java 2 Standard Edition (J2SE) es el entorno básico Java. Esta implementación provee el núcleo de clases y APIs que permite el desarrollo y ejecución de aplicaciones

estándar de clientes y servidores, incluyendo aplicaciones que corren en navegadores web.

Java 2 Enterprise Edition (J2EE) es un gran grupo de APIs Java y otras tecnologías no pertenecientes a Java. Es usado generalmente para crear aplicaciones potencialmente distribuidas. La tecnología J2EE puede servir como el pegamento y aguante para hacer trabajar juntas a grandes aplicaciones heterogéneas y multinivel.

Sun Corporation introdujo la plataforma *Java 2 Micro Edition* en Junio de 1999 en la convención anual JavaOne. J2ME está diseñado para albergar las necesidades Java de la comunidad de dispositivos empotrados y electrónicos personales. Inicialmente, J2ME fue construido para dispositivos con capacidades limitadas de memoria, conectividad (normalmente *wireless*), y capacidades de interfaz gráfica de usuario. Hoy por hoy, como se verá, la tecnología J2ME se ha expandido para cubrir un rango de dispositivos que van desde *pagers* hasta, pero no necesariamente incluidos, los ordenadores personales. Más adelante se expondrán en detalle las principales diferencias de J2ME con las plataformas J2SE y J2EE, casos de uso y las motivaciones que están llevando a J2ME a tener cada día un mayor protagonismo en el desarrollo de software para dispositivos de bolsillo con capacidades móviles y *wireless*.

Considerando lo anteriormente expuesto, lo que se pretende con la realización de este proyecto fin de carrera es el desarrollo de una aplicación prototipo insertada dentro del marco del sistema EFTCoR haciendo uso y realizando un estudio de:

- Una plataforma tipo PDA, realizando un estudio de las particularidades y posibilidades que esto conlleva.
- Desarrollo del SW mediante el lenguaje de programación Java, haciendo uso de la plataforma *PersonalJava* y *J2ME*. Estudio de la plataforma Java de forma genérica como conjunto y centrándose de manera más específica en el estándar Micro Edition. Análisis y descripción de la aplicación mediante metodologías basadas en UML.
- Dotar de funcionalidad inalámbrica al subsistema formado y constituido por la PDA y sus relaciones con los otros subsistemas. Estudio de la tecnología inalámbrica y de los principales protocolos y configuraciones que soporta.

Capítulo 2
Dominio de Aplicación

2.1 Introducción

La superficie del casco de un barco es preparada antes de ser pintada por primera vez. Después de esto, el mantenimiento consiste en una eliminación (cada 4-5 años) periódica de la suciedad, seguido de eliminación de capas y repintado. Esto se hace para preservar la integridad del casco, garantizar las seguras condiciones de navegación, y mantener lisa y uniforme la superficie del casco y así minimizar el consumo de combustible, reducir el coste de las operaciones y prevenir una contaminación ambiental excesiva. La tecnología existente de limpieza del casco, granulado por chorreo, es altamente contaminante, ambientalmente insostenible, y esta siendo progresivamente prohibida en la mayoría de países comprometidos con el medio ambiente. El chorreo granulado esta siendo desde hace tiempo sustituido por chorreo de agua a ultra presión, aunque el resultado es menos satisfactorio. Como el agua a presión sin abrasivo no produce el perfil de superficie deseado, la tecnología está restringida a trabajo de mantenimiento. El desarrollo más reciente en equipos de chorreo abrasivo se trata de unidades de control remoto o de tipo robot para superficies verticales. La principal desventaja es que la operación es restringida a extensas áreas lisas por razones de eficiencia del chorreado, longitud de la operación y economía.

El sistema EFTCoR es una plataforma tele-operada altamente ambiciosa para operaciones no contaminantes de mantenimiento de cascos de buques en diferentes áreas de trabajo: desde largos diques secos libres de obstáculos, hasta áreas llenas de dificultades de acceso. Esto es debido a que el operador no necesita atender las tareas de limpieza permanentemente, y las estrictas condiciones de seguridad están conformes con ello. Con el fin de reducir las posibilidades de error, se ha decidido construir un sistema completo partiendo de una base. La idea viene a considerar un sistema más autónomo en cada incremento. El desarrollo de la *Unidad de Control* del dispositivo robótico ha sido asignado a la Universidad Politécnica de Cartagena. La tarea de dicha unidad es desarrollar el control de los dispositivos robóticos, sistemas de posicionamiento y herramientas usadas en las tareas de mantenimiento, así como la generación de comandos y el uso de comandos y señales generadas por otros subsistemas que han sido asignados a otros compañeros en el proyecto EFTCoR. El desarrollo de esta unidad de trabajo todavía no ha sido realizado en su totalidad, aunque se encuentra muy avanzado. El sistema es capaz de alcanzar una preparación de la superficie de alta calidad junto con una radical reducción de los residuos y nula emisión de contaminantes. En el desarrollo de este capítulo se exponen los requisitos de alto nivel para la solución del sistema, y se muestran las fases en que el sistema está siendo desarrollado. Además, se da una detallada descripción del primer prototipo que ha sido construido. Se ha usado la metodología *COMET (Concurrent Object Modeling and Architectual Design con UML)* para la especificación y diseño distribuido de sistemas en tiempo real [1]. Esta metodología sigue la notación *UML (Unified Modeling Language*). En los apartados posteriores se hace especial mención a los subsistemas 'Unidad de Control' y 'Sistema de Visión', ya que son estos, los dos sistemas con los que debe interactuar la PDA. Finalmente se presentan una serie de conclusiones y detalles del trabajo a realizar en el futuro.

Una vez descrito el contexto o dominio de aplicación se analizarán las características que debe de tener la PDA para una correcta comunicación con el resto de sistemas, y la consecución de la funcionalidad que de ella se demanda.

2.2 Requisitos del Sistema

En esta sección se describen los requisitos de alto nivel para el sistema EFTCoR. Estos requisitos son establecidos sucesivamente mediante una descripción textual de calidad y atributos funcionales, junto con un diagrama UML de casos de uso con una detallada descripción del sistema visto por el usuario final. La metodología COMET considera el diagrama de casos de uso como un contrato inicial entre los desarrolladores y los compradores del producto. Los requisitos de alto nivel para un sistema robótico de confianza para limpieza/chorreado aplicado al mantenimiento de barcos son resumidos brevemente a continuación:

- El sistema debería de ser adaptable a diferentes ambientes de trabajo
- El sistema debería ser adaptable para chorreado dirigido a grandes extensiones del casco o localizado.
- El sistema debería ser adaptable a diferentes operaciones de mantenimiento del casco: chorreado, limpieza con agua fría y pintado.
- Los residuos deben ser reducidos y reciclados.
- El tiempo de las operaciones debería ser similar a los tiempos alcanzados con los habituales métodos manuales.
- El sistema debería ser fácil de operar.
- Sería deseable automatizar las operaciones tanto como sea posible.
- El sistema debe coordinar acciones de diferentes robots y herramientas

La interdependencia de los subsistemas constituyentes debe ser convenientemente estructurada para alcanzar los siguientes requisitos no funcionales o atributos de calidad estáticos:

- Movilidad. El sistema debe permitir cambios tales como la extensión de capacidades para trabajar con distintos controladores para cada eje del mismo robot o diferentes robots.
- Portabilidad. El sistema debe poder implementarse sobre diferentes plataformas. Esto puede asegurarse introduciendo una capa portable que encapsula todas las consideraciones específicas de cada plataforma.
- Reusabilidad. Diferentes partes del sistema necesitan ser reusadas para futuras aplicaciones. Consecuentemente, componentes genéricos deben ser desarrollados independientemente para trabajar juntos.
- Interoperabilidad. El sistema debe trabajar con sistemas tele-operados. La comunicación entre subsistemas debe por lo tanto garantizar la totalidad y consistencia de sus interfaces especificas.

Lo que hace al sistema EFTCoR estar un paso por delante de los sistemas automáticos de chorreo disponibles actualmente, es el amplio rango de operaciones de chorreo que provee, junto con un relativo bajo coste. Así pues, podemos enumerar los siguientes requisitos adicionales:

- Intensivo chorreado de grandes áreas de cascos. La herramienta de chorreado adjunta al robot debería de ser escalable en el sentido de albergar diferente número de mangas para proporcionar una tasa de chorreado de 100 m^2 por hora.
- Trabajo localizado. Para un manejo eficiente de esta operación, el dispositivo robótico debe incorporar un dispositivo primario para un rápido y fácil desplazamiento en el buque. También, debe ser capaz de la detección y procesado automático de porciones dañadas del casco.
- Chorreado de superficies no lisas. El cabezal de limpieza por chorreo adjuntado al dispositivo primario debería de ser adaptable para secciones de superficie de cascos con complejas formas.

2.3 Desarrollo de una Familia de Robots de Limpieza: un Acercamiento Evolutivo

Como se observó anteriormente, un acercamiento incremental a la solución nos proporciona la oportunidad de minimizar los riesgos en cuanto a retrasos, cambios en los requisitos y problemas de aceptación por el usuario final. Además, haciendo uso de entregas intermedias se puede conseguir una realimentación para nuevas versiones del sistema. Al mismo tiempo el usuario puede validar el sistema mientras la complejidad es incrementada. La decisión está basada en ambos, la experiencia previa del grupo de investigación DSIE (Universidad Politécnica de Cartagena) en distintos proyectos, y lo observado durante el tiempo que lleva el proyecto en marcha (tres años). Los requisitos del sistema EFTCoR están bien definidos, y se puede hacer un desarrollo del sistema de forma incremental, sin perder el alcance general del sistema final. De esta forma, el éxito del proyecto está prácticamente garantizado. Basándose en estas premisas, una serie de fases han sido establecidas para el desarrollo del sistema. Estas fases comprenden dos dimensiones ortogonales del desarrollo: la mecánica del sistema robótico y el hardware y software para controlar las operaciones. Cada fase cuenta con un porcentaje de cada dimensión. De este modo, la primera y segunda fase tratan el problema mecánico, mientras que la tercera fase está principalmente enfocada en las estrategias de control para la computación distribuida. Las características de los prototipos para cada fase son las siguientes:

Primera Fase: El sistema robótico tendrá en cuenta las consideraciones básicas de seguridad respecto al movimiento y limpieza. Esta fase considera dos sistemas diferentes: en primer lugar, un sistema de base mecánica destinado a la limpieza de áreas de gran extensión y fácilmente accesibles del casco; el segundo es un robot móvil tele-operado para limpiar áreas de difícil acceso. En términos mecánicos, el sistema tele-operado en esta fase es capaz de moverse (por control remoto) y de adherirse a la superficie, pero no contempla reciclaje de emisiones, evitar obstáculos o control de velocidad.

Segunda fase: Esta fase completa los aspectos mecánicos no considerados en la primera fase. De este modo, el sistema será diseñado para asegurar la reducción de los residuos con una nula emisión de contaminantes al medio. Además, el prototipo será capaz de ajustar sus trayectorias para alcanzar los objetivos, incluso ante la presencia de obstáculos. El operador solo será requerido para supervisar las operaciones de limpieza y los movimientos del sistema de control del robot por las diferentes áreas a ser tratadas. La coordinación entre robots se deja para la siguiente fase. El resultado de esta fase es un sistema tele-operado (móvil o no) con autonomía para decidir los puntos a ser tratados.

Tercera fase: Esta última fase alberga la cuestión del control de una familia de robots de limpieza tele-operados. El uso de un considerable número de robots (similares o no) cooperando puede reducir considerablemente el tiempo total de operación y los costes. Son necesarios una serie de algoritmos de coordinación y una arquitectura bien definida. En esta fase se alcanzan los requisitos globales del sistema EFTCoR.

2.4 Diagrama de Casos de Uso del Primer Prototipo

Los requisitos funcionales de un sistema definen lo que el sistema hará por el usuario. Cuando se definen los requisitos externos, el sistema debería ser visto como una caja negra, de forma que sólo las características exteriores del sistema son consideradas. Los casos de uso UML son usados para definir los requisitos externos funcionales del sistema. Esta sección describe el *UML Use Case Diagram* (diagrama de casos de uso UML) obtenido de los requisitos establecidos en nuestro sistema. Se ha considerado un sistema de posicionamiento primario (por ejemplo una plataforma de elevación) y un sistema de posicionamiento secundario (mesa XY) que adjunta la cabeza de limpieza. En este primer prototipo, el Sistema de Visión se usa para dar al operador una vista panorámica de la superficie a ser tratada. Así pues, no hay automatización en la tarea de limpiado. El operador usará el terminal tipo PDA para operar el sistema de limpieza, Un enlace de comunicación conectará el sistema tele-operado con el Sistema de Monitorizado Remoto para intercambiar información acerca de las tareas de limpieza. La descripción de los casos de uso se presenta en forma de plantilla que incluye: nombre, el principal actor (usuario) que interactúa con los casos de uso (operador, Sistema de Monitorizado y Sistema de Visión), un sumario de la funcionalidad, las precondiciones que deben de ser ciertas para empezar un caso de uso, una secuencia normal de ejecución del caso de uso y finalmente, una serie de alternativas que pueden surgir cuando el caso de uso es ejecutado.

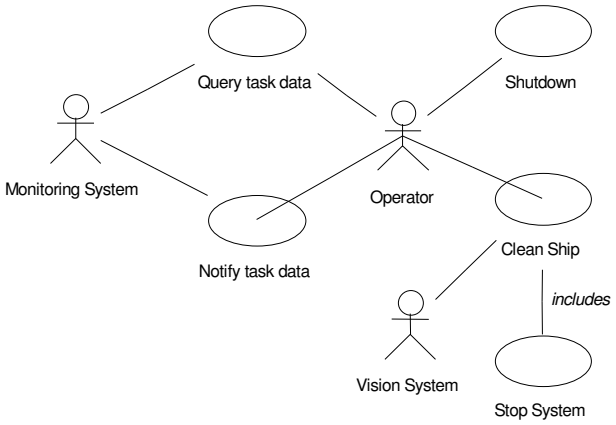


Figura 1. Diagrama de Casos de Uso

La tabla 1 contiene una completa descripción del diagrama de casos de uso de la figura 1:

Nombre del caso de uso: Clean Ship **Actor**: Operator **Sumario:** El operador posiciona la cabeza de limpieza con un joystick, usando la imagen del vídeo mostrado sobre el terminal. **Precondición**: El sistema debe de haber sido iniciado, estar en el estado correcto y encontrarse mostrando vídeo sobre el terminal multimedia.	**Secuencia normal:** 1. El operador mueve el joystick hasta que la cabeza del robot alcanza la superficie a ser limpieza 2. Una vez que la cabeza de limpieza ha sido posicionada, el operador pulsa el botón de comienzo de chorreado 3. El operador puede seguir los movimientos de la cabeza de limpieza sin desconectar la operación de chorreo **Alternativas:** 1. El final es alcanzado. El chorreo está parado y una alarma acústica es generada 2. El sistema de chorreo está parado y una alarma acústica es generada como señal de que el tiempo límite sin movimiento (pero chorreando) ha sido alcanzado 3. El operador pulsa el botón de parada de emergencia. El caso de uso "stop system" es ejecutado 4. El sistema para el chorreado y una alarma acústica es generada como señal de que el tiempo límite en inmovilidad y/o chorreado ha sido alcanzado
Nombre del caso de uso: Stop System **Actor:** Operator **Sumario:** El sistema permanece parado (standby mode). **Precondición:** El sistema no está parado.	**Secuencia normal:** 1. El operador pulsa el botón de stop 2. El movimiento se para 3. Los interruptores del extremo se fijan a encendido 4. El sistema de chorreo está parado **Comentarios**: Esta es la única opción de parada. Esto contempla emergencias u otras paradas, por ejemplo con el fin de mover el sistema primario de posicionamiento.
Nombre del caso de uso: Query Task Data **Actores:** Operator, Monitoring System **Sumario**: El operador envía una petición al sistema externo de monitorización y recibe información acerca de la tarea a realizar. **Precondición:** El sistema está parado.	**Secuencia normal:** 1. El operador pulsa el botón de solicitud 2. El sistema muestra un menú de solicitudes desde el cual el operador puede elegir la opción deseada 3. El sistema muestra los datos al operador **Alternativas:** 1. No hay un enlace de comunicación con el sistema de monitorización. Este error necesita ser mostrado al operador
Nombre del caso de uso: Notify Task Data **Actores:** Operator, Monitoring System **Sumario**: El operador pulsa el botón de notificación y envía los datos al sistema remoto de monitorización.	**Secuencia normal:** 1. El operador pulsa el botón de notificación 2. El sistema muestra un menú de notificación desde el cual el operador puede elegir la opción deseada 3. El sistema envía los datos al sistema de monitorización.

Precondición: El sistema está parado.	**Alternativas:** 1. No hay un enlace de comunicación con el sistema de monitorización. Este error necesita ser mostrado al operador
Nombre del caso de uso: Shutdown **Actor:** Operator **Sumario:** El operador finaliza la tarea. **Precondición:** El sistema está parado.	**Secuencia normal:** 1. El operador pulsa el botón de shutdown 2. Los registros historiales de la tarea finalizada son enviados al sistema de monitorización 3. El sistema suspende las comunicaciones 4. El sistema muestra un mensaje para recordar al operario las operaciones finales a llevar a cabo **Comentarios:** Este caso de uso está para ser ejecutado al cierre de la jornada de trabajo.

Tabla 1. Descripción del Diagrama de Casos de Uso del primer prototipo.

2.5 Análisis y Diseño Preliminar del Primer Prototipo

En la fase de análisis del modelado, se han desarrollado una serie de modelos estáticos y dinámicos. El modelo estático define las relaciones estructurales entre las entidades del dominio del problema. Las clases y sus relaciones son representadas en diagramas de clases. El modelo dinámico es desarrollado partiendo de los diagramas de casos de uso mediante el estudio de la interacción entre objetos los cuales serán representados usando diagramas de colaboración UML. Los siguientes pasos están basados en la metodología COMET.

1. Desarrollar un modelo estático del dominio del problema con
    - Un modelo estático del problema, que comprenda las clases que modelan entidades físicas y los diagramas de clases que representan sus relaciones
    - Un modelo del contexto del sistema el cual muestre como las clases externas se comunican con el sistema
    - Un modelo estático de clases de entidad del dominio del problema (clases que representan los datos persistentes del sistema)
2. Estructurar las clases y los objetos identificados en el escenario anterior en subsistemas, aplicando objetos bien conocidos y técnicas de estructuración de clase. Este paso determina la estructura de los subsistemas a un nivel muy alto de abstracción.
3. Desarrollar un modelo dinámico para cada caso de uso mediante la identificación de:
    - Los objetos que participan en cada caso de uso
    - El diagrama de interacción objeto-objeto (colaboración y secuencia)
    - Los diagramas de transición de estados que describen el comportamiento estado-dependencia de objetos complejos.

En las siguientes secciones, los diagramas mencionados anteriormente serán presentados.

2.5.1 Modelo Estático del Dominio del Problema

En un modelado estático del dominio del problema, el énfasis inicial en COMET se centra en modelar clases físicas y clases de entidad. Clases físicas incluyen dispositivos físicos, usuarios, sistemas externos y temporizadores. Con este modelo, podemos identificar una serie de requisitos para el sistema EFTCoR:

- **PDA Tele-operada**. A través del terminal, el operador puede ver mediante vídeo las áreas afectadas del casco, además de recibir gráficamente información de la posición de la cabeza del robot. Esta última se obtiene mediante unos sensores situados en los vértices de la mesa que sostiene el cabezal de limpieza, cuyos valores son recibidos en el terminal móvil vía *wireless*. Un conjunto de comandos enviados desde el terminal permiten al subsistema de tele-operación mover al robot cuando el modo de operación es seleccionado. En síntesis, a través del terminal, el operador humano puede recibir información que le será de ayuda para la realización de sus tareas; y además puede enviar comandos para notificar o suministrar información al sistema de control.
- El **Sistema de Visión** ofrece al operador la imagen del vídeo en tiempo real de la superficie sobre la cual está posicionada la cabeza de limpiado. Versiones posteriores del sistema de visión controlarán la operación de chorreado mediante el cálculo de la trayectoria del robot; además, comprobarán los resultados de la calidad del chorreado. El Sistema de Visión también proveerá una guía de planificación en línea para el trabajo localizado y valoración de la superficie tratada, implementando un control de calidad. El sistema estará comunicado con el subsistema de control del robot y el sistema de monitorización.
- El **Sistema de Monitorización**. Este sistema es externo al sistema de tele-operación. El sistema de monitorización considera un listado de información para cada tarea de limpieza. El sistema se comunicará con el sistema tele-operado mediante tecnología *wireless*. El operador preguntará y notificará los datos que producen las tareas de mantenimiento.
- **Sistema de Posicionamiento**. Su propósito es mover la cabeza de limpiado acercándola o alejándola de la superficie del casco. El sistema de posicionamiento debe comprender dos sistemas, primario y secundario, de posicionamiento: el sistema primario de posicionamiento puede ser una larga grúa para alcanzar todas las áreas del casco, y el sistema secundario de posicionamiento debe ser un robot para cubrir áreas de alrededor de 3 m^2 (más o menos el área que un operador humano puede limpiar desde una posición dada). Las principales razones que nos hacen considerar diferentes, primario y secundario, sistemas de posicionamiento son:
    - La dificultad de encontrar un sistema de posicionamiento que cumpla alcance, carga, precisión y requisitos de control al mismo tiempo.
    - La necesidad de diferentes sistemas de posicionamiento dependiendo del tamaño y la forma del casco y la parte del casco a limpiar.
    - La existencia de un sistema de posicionamiento primario para chorreado de superficies que puede ser reutilizado para chorreado por puntos si un sistema de posicionamiento secundario controlable es adosado al mismo.

Dependiendo de la naturaleza de la tarea a llevar a cabo y de las características de las áreas a ser tratadas, podría ser requerida la configuración en principio de solo uno de estos sistemas. En general, el sistema de posicionamiento primario no tiene que ser muy preciso, pero debe ser fácil de operar y razonablemente seguro y rápido. El sistema secundario de posicionamiento debe posicionar la cabeza de limpieza sobre el área a ser limpiada con razonable velocidad y precisión. El sistema secundario de posicionamiento es el primer candidato para automoción, así que es esencial que pueda ser operado como un robot. La figura 2 muestra una representación gráfica del sistema de posicionado del EFTCoR.

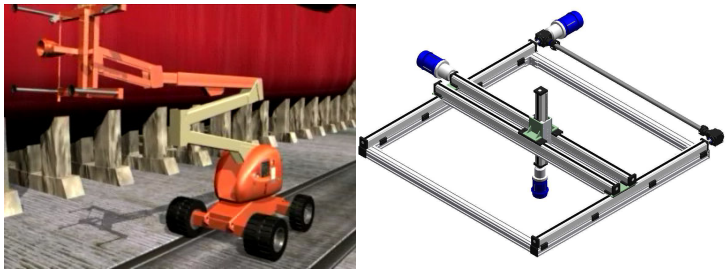


Figura 2. Sistemas de posicionamiento primario y secundario

- **Sistema de limpieza.** Este sistema está compuesto por tres partes primarias: la cabeza de chorreo (unidad de inyección), la unidad de aspirado y la cabeza de limpieza. La cabeza de limpiado consiste en un cabezal de inclinación ajustable que guía la manguera de chorreo y permite que el ángulo de incidencia, presión de aire y alimentación de arena sean ajustados para controlar la operación de chorreo. El conjunto es adherido mediante una cubierta especialmente diseñada y colocada alrededor de las cabezas de chorreo para sellar las unidades a la superficie que está siendo limpiada, previniendo emisiones de polvo y residuos. Un contacto flexible entre el sellado y el casco es alcanzado por el uso combinado de fuentes de aire y fuentes regulables. El manejo de deshechos y el sistema de reciclado elimina los residuos producidos por el chorreado del casco, permitiendo una reutilización del material arenoso y el correcto empaquetado de otros residuos.

En resumen, el modo de operación normal del sistema EFTCoR incluye un operario encargado de la monitorización y operatividad del robot en base a la información recibida del sistema de tele-operación. Este sistema recibe comandos del operador y lleva a cabo las acciones requeridas para ejecutarlos. Con este propósito, se comunica con la unidad remota de control, que controla físicamente al robot. La unidad de control del robot recibe una realimentación de los sensores del robot y la operación de chorreo. Usa estos datos para acceder al estado global del sistema y enviar la información al sistema de tele-operación. El sistema de tele-operación usa esta información para representar gráficamente el estado del robot y asegurarse de que esta trabajando correctamente.

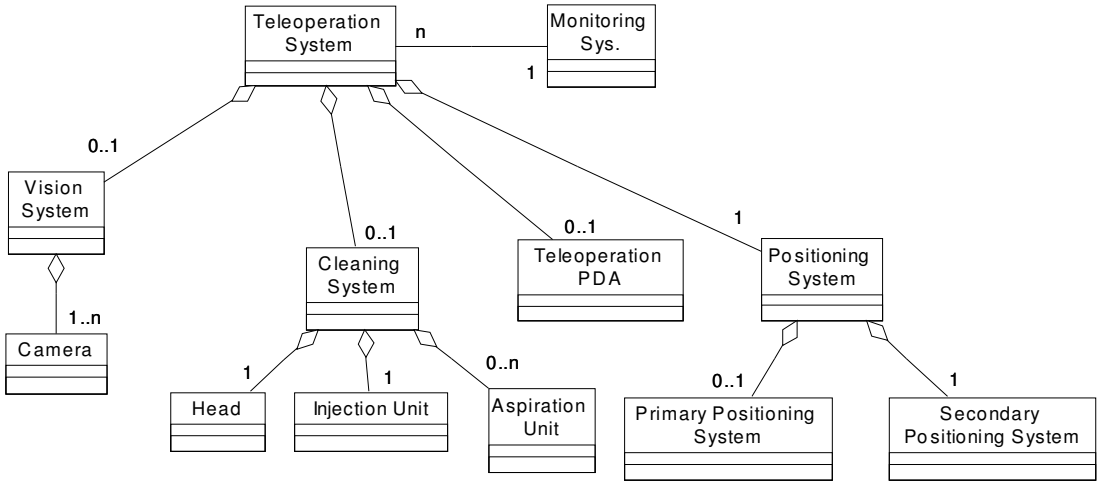


Figura 3. Modelo estático del dominio

La figura 3 muestra las clases físicas y las relaciones entre ellas, siguiendo la notación UML y los estereotipos provistos por la metodología COMET. Esto suministra una vista general del sistema tele-operado en términos de las entidades físicas que lo componen. La cardinalidad denota el número de componentes asociados a cada subsistema. Cada una de las entidades modeladas por las clases de la figura han sido explicadas anteriormente.

2.5.2 Modelo Contextual del Sistema EFTCoR

Es muy importante entender la interfaz hardware/software entre el sistema y el exterior. Esta interfaz es representada mediante la metodología COMET usando diagramas de clases UML, tal y como se muestra en la figura 4. Este modelo puede ser determinado por modelos estáticos o clases externas que se conectan al sistema. El diagrama muestra los límites del sistema y el tipo de elementos con los que el sistema está enlazado (esto es, dispositivos de entrada y salida).

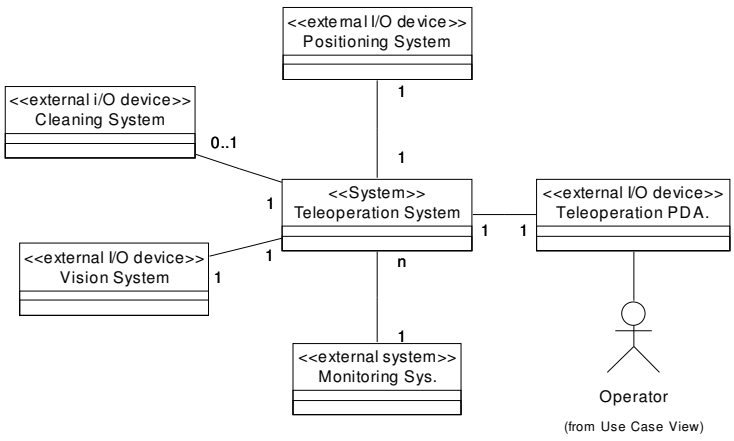


Figura 4. Modelo contextual del sistema tele-operado

El tipo (ejemplo: <<*external I/O device*>>) es denotado usando el mecanismo de estereotipado UML.

2.5.3 Diagramas de Interacción

El modelado dinámico se sostiene fuertemente sobre mensajes y eventos entre objetos. Después de que los objetos hayan sido identificados usando el criterio de estructuración de objetos de la metodología COMET, el modo en que los objetos cooperan entre sí puede ser representado sobre diagramas de interacción (y más específicamente con diagramas de colaboración UML). En esta fase de análisis, un diagrama de interacción será creado para cada caso de uso. Las clases han sido estereotipadas usando la taxonomía de COMET. El significado de los estereotipos puede ser deducido a partir de sus nombres y la forma en que son usados.

El diagrama de interacción incluye todos los mensajes entre los objetos para satisfacer el caso de uso referido. Se considera un objeto de la interfaz para cada dispositivo externo. Este objeto encapsula las particularidades de la interacción entre el dispositivo externo y el sistema. El objeto de la entidad se encuentra para almacenar datos persistentes sobre el proceso.

2.5.4 Diseño Preliminar del Sistema EFTCoR

El software para sistemas robóticos presenta un número de características especiales relativas a esta aplicación particular. Los principales requisitos son:

- capacidad para alcanzar complejos objetivos de alto nivel
- capacidad para interactuar con complejos, y con frecuencia dinámicos ambientes de trabajo, y
- capacidad para asegurar el comportamiento dinámico del sistema, incluso ante cambios imprevistos en el entorno de trabajo. Además, muchos sistemas robóticos pertenecen a la categoría de sistemas críticos. Las principales necesidades para sistemas críticos incluyen seguridad software, respuesta en tiempo y tolerancia a fallos, lo cual es esencial para eludir errores en las operaciones.

Hay numerosas referencias en literatura tecnológica sobre sistemas robóticos para el desarrollo de arquitecturas para alcanzar los requisitos anteriormente expuestos. Tales arquitecturas han sido diseñadas generalmente para cumplir con requisitos para una tarea muy específica (inspección, manipulación, etc) y proveen un número fijo de modos de operación. También hay algunas arquitecturas generales diseñadas para extensas aplicaciones. Estas proveen control del robot y se centran principalmente en tareas de planificación mas que en proveer un entorno de tele-operación reconfigurable. Actualmente se trabaja en desarrollar un marco de trabajo de componentes con interfaces bien definidas que pueden ser combinadas para llegar a la arquitectura más conveniente para cualquier aplicación dada. Bastantes autores han usado aproximaciones mecánico-electrónicas para integrar componentes de sistemas robóticos donde hay también interacción con el entorno. Estos sistemas contemplan arquitecturas abiertas con integración de componentes hardware y software. De cualquier modo, están desprovistos de notaciones estándar y metodologías para el desarrollo de software.

En la metodología COMET, el diseño primero identifica los subsistemas en los que el sistema global puede ser estructurado. Para alcanzar esto, la metodología define una

serie de pasos: *Consolidate Collaboration Diagram y Preliminary Architectral Diagram*. Esta sección describe el diagrama envuelto en el primer paso, incluyendo el consolidado diagrama de colaboración y la primera versión del diagrama estructural preliminar.

Un sistema esta compuesto de subsistemas los cuales contiene objetos que son funcionalmente dependientes unos de otros. Un subsistema puede ser relativamente independiente de los otros. En el análisis del dominio del problema y su descomposición en subsistemas, el énfasis es en la separación. Cada subsistema puede descomponerse en pequeños subsistemas. Una vez que la interfaz entre subsistemas ha sido determinada, el diseño de cada subsistema puede ser independiente. Si tomamos el diagrama de colaboración consolidado y aplicamos el criterio de estructuración definido por COMET, podemos definir una partición que permita definir un diagrama de diseño estructural preliminar que incluya comunicaciones entre los subsistemas. Las responsabilidades, las entradas y salidas de los subsistemas (definidos por el diagrama arquitectural preliminar) son mostradas en la Tabla 2.

Subsistema	**Responsabilidades**	**Entradas abstractas**	**Salidas abstractas**	**Observaciones**
Cleaning	Coordina el funcionamiento apropiado de los subsistemas incluidos según los criterios dados del alto nivel (dependiente del uso).	La suma de las entradas de los subsistemas incluidos.	La suma de las salidas de los subsistemas incluidos.	
Secondary Positioning Controller	Controla el sistema de colocación secundario. Traduce comandos del movimiento de alto nivel a órdenes para los dispositivos físicos y controla su ejecución.	Comandos del movimiento, incluyendo parada. Datos de los sensores (posición, esfuerzo de torsión, interruptores de límite, alarmas, etc)	El movimiento del nivel bajo ordena al sistema de colocación secundario (activación del motor y activación del interruptor).	Absolutamente independiente del subsistema de colocación primario y del subsistema de limpieza.
Cleaning Head Controller	Controla la cabeza de la limpieza, incluyendo los dispositivos asociados (los dispositivos del accesorio, los controles neumáticos e hidráulicos…).	Comandos de la activación. Datos de los sensores (presión del aire o del aceite, alarmar…)	Órdenes a la cabeza de limpieza.	Absolutamente independiente de colocar subsistemas.
Monitoring Management	Comunicaciones con el Remote Monitoring System, incluyendo conexiones wireless.	Datos del Remote Monitoring System (respuestas). Estado de la conexión.	Datos para el Remote Monitoring System (preguntas).	

PDA Terminal	Interacción con el operador. Proporciona el acceso a los comandos y muestra la información de estado, incluyendo imágenes del subsistema visión.	Sistema de información de estado. Responde a las preguntas (del Remote Monitoring System) Imágenes del sistema de visión.	Comandos del operador para el resto de los subsistemas.	
Vision Connection	Toma y procesa (si es requerido) las corrientes de vídeo del sistema de vídeo externo.	Vídeo en tiempo real de la superficie limpiándose	Ordena el movimiento de las cámaras.	

Tabla 2. Subsistemas identificados y sus responsabilidades

2.6 Unidad de Control Remota

El objetivo del paquete de trabajo es el desarrollo de la unidad de control para el sistema robótico.

Como se ha comentado en apartados anteriores, la PDA debe de interactuar con este subsistema mediante el intercambio de información y envío de comandos. Básicamente se pretenden dos cosas respecto al intercambio de información con el terminal PDA:

- La PDA debe de recibir con un intervalo de tiempo determinado la actualización de los valores de los sensores ópticos situados en la mesa del robot. De este modo el operador humano sabrá con exactitud a qué distancia se encuentra el cabezal de limpieza de la superficie del caso y por lo tanto que movimientos debe de realizar (avance/retroceso). El valor de los sensores puede ser representado gráficamente sobre la pantalla de la PDA, para hacer de ésta una interfaz más amigable.
- El sistema debe de ser capaz de recibir una serie de comandos enviados desde el terminal móvil (PDA), y de este modo realizar las acciones oportunas. Esta comunicación será preferiblemente inalámbrica.

2.6.1 Requisitos de Comunicación

Como se comentó anteriormente, el proyecto de EFTCoR considera:

- No un solo sistema robótico, sino una familia de ellos.
- No un solo sistema, sino un grupo de sistemas de cooperación

Otras características que se deben abarcar.

- Su capacidad de ser extendido con nueva funcionalidad (nuevos sistemas de colocación, nuevos procesos, herramientas nuevas, etc).
- Su capacidad inter-operar con las unidades de control de otros dispositivos robóticos.

- Su capacidad inter-operar con los sistemas externos considerados en el proyecto (sistema de la visión, sistema de supervisión y reciclaje del sistema).

La figura 5 muestra los requisitos de comunicación entre los distintos subsistemas que se incluyen en el proyecto EFTCoR.

Hay cuatro importantes subsistemas que deben de intercambiar comandos e información.

- La unidad de control para los dispositivos robóticos
- El sistema de monitorización
- El sistema de visión
- El sistema de reciclado

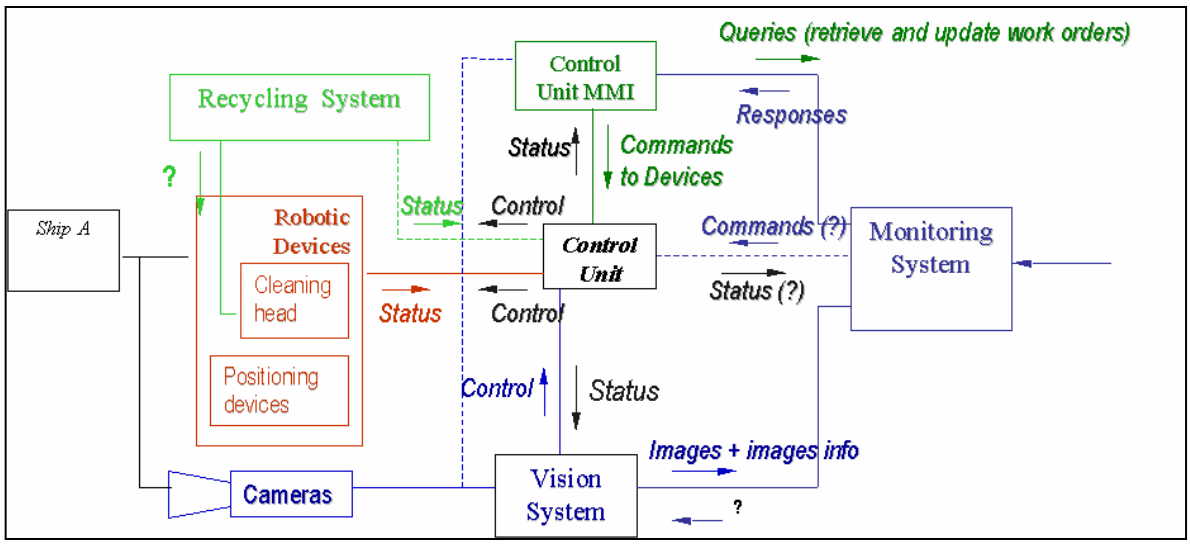


Figura 5. Requisitos de comunicación entre subsistemas

La información que intercambia la Unidad de Control es básicamente de tres tipos: información de estado, información de control y comandos. Todos los subsistemas están conectados a la Unidad de Control.

A través de la PDA, el operador podrá recibir y enviar comandos a otros subsistemas. Esto es posible a la interconexión existente entre los subsistemas constituyentes.

El sistema de visión puede disponer de una o varias cámaras. La captura de vídeo se mostrará en tiempo real sobre la pantalla del dispositivo tipo PDA, facilitando las tareas a desempeñar por el operario que la transporta.

2.6.2 Diseño y Desarrollo del Software de la Unidad de Control

La unidad de control es un sistema intensivo en software pero abarca ambos, hardware y software. Estas tareas deben ser llamadas más correctamente 'Diseño y desarrollo de la unidad de control'.

Una unidad de control se ha diseñado, se ha construido y se ha programado parcialmente para controlar algunos de los dispositivos robóticos. Esta unidad de control logra los requisitos descritos arriba y se compone principalmente de dispositivos comerciales de control y de los puentes de comunicaciones de uso estándar. Puede ser extendido o configurado con nueva funcionalidad y puede inter-operar con otras unidades de control que actuando como maestra de ellas.

Actualmente, la misma unidad de control se está utilizando para todos los sistemas robóticos desarrollados. Se planea construir una versión de pequeño tamaño como prototipo de los vehículos con el fin de que su desarrollo y pruebas sobre los mismos no interfieran con el desarrollo y la prueba del resto de los dispositivos

Figura 6. Elementos de la Unidad de Control

Como se observa en la figura, los elementos actuales no disponen de una pantalla donde puedan mostrar vídeo u otras informaciones útiles para la realización de tareas por parte del operador humano. Es aquí donde la PDA puede añadir una funcionalidad extra sobre los dispositivos utilizados hasta ahora en el desarrollo de la Unidad de Control. Por otra parte estos elementos, cuya funcionalidad es más limitada, están mejor adaptados al entorno industrial de trabajo. Por su constitución física son en principio más adecuados para ser usados en un entorno cuyas condiciones pueden llegar a ser muy agresivas (polvo y residuos que pueden caer sobre el dispositivo, caídas durante el desarrollo de las tareas, etc). Su manejo es también más tosco, con un número limitado de botones y mecanismo (tipo seta) para parar el sistema de un modo rápido en caso de emergencia. Sus funciones se limitan al control de los movimientos en las tareas de limpieza, y arranque y parada de dicho sistema. La PDA permite, no únicamente la entrada de vídeo en tiempo real, sino también la representación sobre su pantalla de otras informaciones y el envío de comandos por medio de los botones físicos de los que dispone, o mediante aplicación software. Por estas razones, el terminal PDA puede emplearse en un principio como un prototipo, que puede evolucionar o no a utilizable en las tareas de trabajo diarias. Su principal desventaja es su tamaño y constitución, en principio poco adecuados para entornos industriales agresivos.

2.6.3 Test de Verificación y Validación

La unidad de control se ha utilizado para controlar la tabla de XYZ y los prototipos de vehículos, pero no se ha probado sistemáticamente aun. Se ha dotados a las extremidades del robot de unos sensores que informan de la distancia de éstas respecto a la superficie del barco. De este modo es posible disponer en todo momento del valor de las coordenadas X, Y, y Z; y activar las órdenes oportunas para modificar la posición de las extremidades del robot según convenga.

Comparación de objetivos alcanzados y de objetivos indicados:

- El concepto de la unidad de control ha sido respecto modificado a la oferta, pero no el alcance de sus funciones y responsabilidades
- La oferta indica que la unidad de control era incluida por razones de seguridad, ofreciendo una manera alternativa del control en el caso de ausencia del sistema de monitorización.
- Después de repasar las funciones del sistema de monitorización la unidad de control es la única responsable del control en línea de los dispositivos.

El marco conceptual de la comunicación y la infraestructura de la comunicación se definen y se eligen respectivamente, pero los mensajes concretos entre subsistemas y sus formatos se deben definir, así como algunos aspectos detallados de los protocolos de comunicación.

2.7 Sistema de Visión

Es el otro sistema que tiene una relación directa, respecto a funcionalidad, con el terminal PDA. Este sistema implementa un mecanismo de visión artificial mediante el cual reconoce las partes dañadas del casco y de este modo decide cuáles deben ser tratadas. No se pretende que la PDA implemente ninguno de estos mecanismos. El objetivo buscado es que se muestre el vídeo en tiempo real sobre la PDA antes de ser manipulado. Se utilizan las cámaras de vídeo empleadas por el Sistema de Visión para mostrar las imágenes del casco del barco al operador humano, mientras realiza sus tareas. Esto supone una guía adicional para la realización de las mismas, ya que en ocasiones la visualización directa de las partes que están siendo tratadas puede ser complicada para el operador, ya sea por la distancia, como por las condiciones atmosféricas derivadas de las operaciones de limpieza.

2.7.1 Papel del Sistema de Visión

- Visualización

La operación de chorreo es visualizada por el operador en un monitor TFT dentro del panel de control. Una o más cámaras serán las encargadas de captar las imágenes.

- Guía para los dispositivos de limpieza

Combina información de imágenes junto con información de proximidad. Esta última viene dada por los datos recibidos de unos sensores ultrasónicos que ayudarán al operador al colocar la herramienta de chorreo.

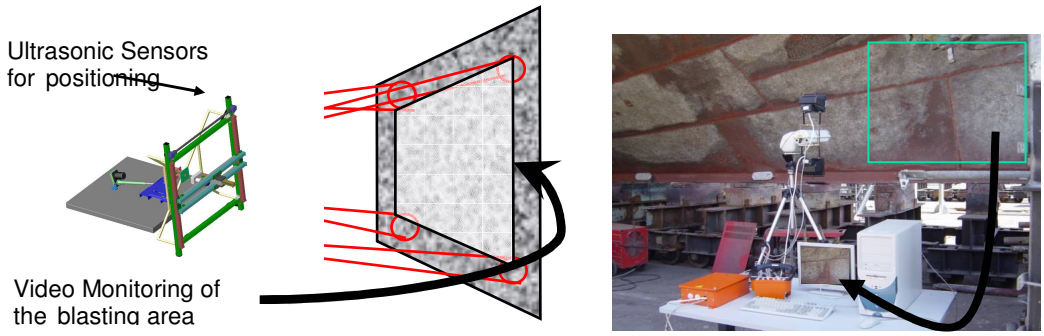


Figura 7. Uso de los sensores ultrasónicos y captura de imágenes

Cuatro sensores son colocados en cada uno de los vértices de la mesa; mediante ellos se puede conocer en todo momento y con exactitud la distancia de la misma respecto de la superficie del casco. El terminal PDA debe recibir estos valores y presentarlos gráficamente de manera que se le de soporte al operador humano que esté realizando las tareas.

- Operación de chorreo medio automatizada

Gracias al uso de una interfaz amigable, el operador puede seleccionar un área a chorrear de un modo automatizado. Es el sistema de visión el encargado de determinar que áreas están dañadas, y así liberar de esta tarea al operador humano.

Como se puede observar en la secuencia de imágenes, en primer lugar el área a tratar es determinada por el sistema de visión artificial. Esto se consigue por un lado gracias al sistema de posicionamiento guiado por los sensores. Este sistema coloca en cabezal de limpieza exactamente en el área que se desea. Por otro lado el Sistema de Visión, con sus mecanismos de visión artificial, detecta cuáles son las áreas afectadas e indica donde debe de colocarse el sistema de posicionamiento:

- Chorreo automático
- Clasificación de los daños del casco
- Operación automática
    - Guía para el movimiento del dispositivo secundario (traslación X-Y)
    - El sistema determina las áreas a ser chorreadas

2.7.2 Módulos del Sistema

A través de la siguiente figura se observa cuál es la situación que ocupa la PDA dentro del Sistema de Visión. La PDA es alimentada con información a través del PROFIBUS. De esta forma, es capaz de acceder al valor actual de los sensores, que indican la situación del robot respecto a la superficie de barco, como al flujo de vídeo que es captado por la cámara. La PDA debe poder recibir el vídeo mediante una conexión directa con la cámara. También podría recibir el flujo de vídeo a través del BUS de comunicación, después de ser tratado por un servidor.

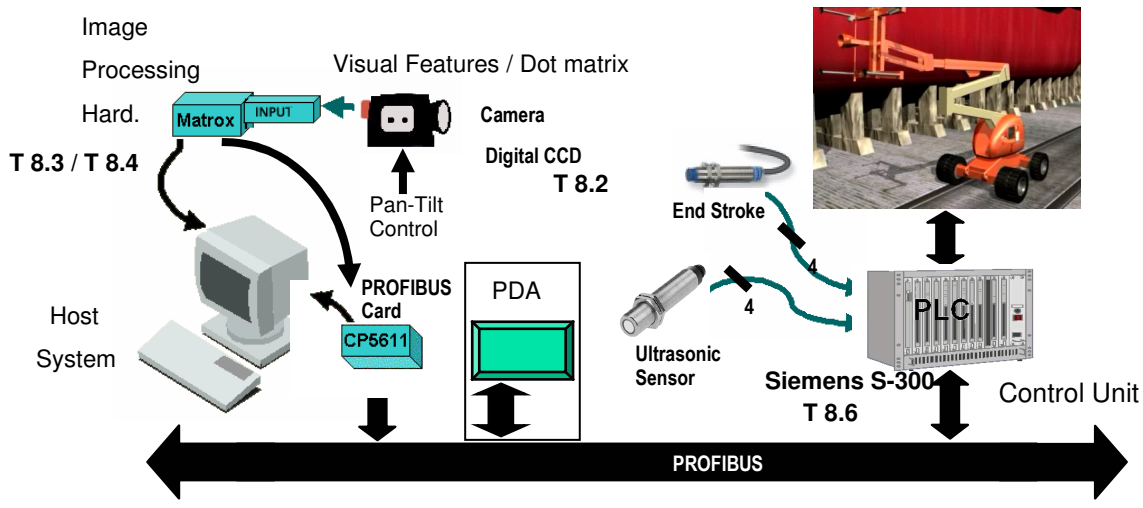


Figura 8. Interacción de una PDA con el sistema de visión

A continuación se muestran las principales características de los componentes de este subsistema

- Módulo De la Adquisición
  - Color CCD 1.2 Mpix, 15 fps
  - FIreWire connection. Optic fiber link (up to 500m).
  - Halogen / Discharge 5000ºK lamps
  - Pressured enclosure IP 66
  - Pan-tilt mount IP 66

- Hardware de procesado de imagen
  - Procesado de imagen en tiempo real
  - 7 MOPs
  - Independencia
  - Entradas analógicas/digitales de vídeo
  - Salidas digitales
  - Tarjeta Profibus

- Módulo para el control del chorreado
  - BERO 3RG6 Sensores ultrasónicos (300mm<d<3m) para maniobrar durante el posicionamiento (usados en vez de sensores láser).
  - Sensores inductivos para detectar el contacto con la superficie
  - CP 5611 Tarjeta Profibus

2.8 Función de la PDA dentro del sistema EFTCoR

Una vez analizados los requisitos del sistema en el que la PDA se verá integrada, obtenemos como consecuencia los requisitos que debe de cumplir el dispositivo por sí mismo.

El sistema de mando debe ser ágil, ligero, permitir comunicación inalámbrica y que muestre vídeo. Veamos las ventajas, frente a estas demandas, de incluir en el sistema global un dispositivo de tipo PDA.

- Por su tamaño y peso, hace que sea posible una utilización del mismo ágil, pudiendo desplazarse con suma facilidad dentro del entorno de trabajo.
- Este tipo de dispositivos vienen provistos de tecnologías *wireless*, típicamente LAN inalámbrica (802.11b) y *Bluetooth*. De este modo queda garantizada su comunicación inalámbrica con los sistemas con los que interactúa, sin necesidad de dispositivos complementarios.
- Las marcas comerciales suelen proporcionar software específico multimedia con el fin de recibir vídeo en la pantalla de la PDA con una calidad óptima.

La PDA adquirida para la realización de este proyecto cumple con todos estos requisitos. Sus características y funcionamiento serán descritas mas adelante en el Capítulo 3, donde se da una descripción detallada de todos sus componentes.

El dispositivo adquirido debe satisfacer una serie de necesidades concretas. Debe de ser capaz de mostrar vídeo en tiempo real; para ello debe de disponer del correspondiente hardware y software necesario para ello. En el caso del dispositivo utilizado en este proyecto, se ha adquirido un módulo expansor multimedia que permite entrada y salida de vídeo en tiempo real. Este módulo viene equipado con las aplicaciones software necesarias. Otra de las funciones que queremos dar a la PDA es la de enviar comandos al sistema de control. Estos comandos pueden ser activados a través de una aplicación software, pero principalmente interesa que el envío pueda llevarse a cabo mediante los botones (físicos) propios del terminal. Además de vídeo, sobre la pantalla de la PDA se mostrará información sobre el valor de los sensores ópticos. Tanto los datos entrantes como salientes del terminal deben de ser transmitidos de forma inalámbrica, por lo que la PDA debe de ofrecer la posibilidad de comunicación *wireless.*

La PDA adquirida para la realización de este proyecto cumple con todos estos requisitos. Sus características y funcionamiento serán descritas más adelante en el Capítulo 3.

Capítulo 3
Infraestructura Utilizada

3.1 Introducción

A lo largo de este capítulo se hace una exposición de la infraestructura, tanto a nivel hardware como software, utilizada para el desarrollo e implementación de la aplicación realizada. También se presentará una introducción a la tecnología *wireless.*

Los elementos utilizados han sido:

- Un dispositivo tipo PDA, en concreto **iPAQ Pocket PC 5500** de la casa HP. Se describirán con detalle las características físicas del dispositivo: sistema operativo, capacidades de procesador y memoria, puertos de comunicación disponibles, modos de comunicación con otros dispositivos, protocolos *wireless* soportados, etc. Además se describen las principales funcionalidades del software que tiene instalado, así como sus modos de uso. Para la implementación de la aplicación escrita en Java ha sido necesario instalar una máquina virtual específica para este tipo de dispositivos. El proceso de instalación y uso será descrito a lo largo del capítulo, así como las peculiaridades de esta plataforma de Java.
- Dispositivo de expansión multimedia **FlyJacket i3800** de LifeView para captura de vídeo en tiempo real sobre la PDA. Junto con el dispositivo de expansión se proporciona un software específico para explotar las capacidades multimedia del modulo: *Suite PowerMedia*. Se trata de un conjunto potente compuesto por cuatro aplicaciones de la *IA Style* diseñadas para aprovechar al máximo las capacidades multimedia del *FlyJacket i3800*. Las aplicaciones suministradas son: *IA Album*, *IA Presenter*, *IA Screen Mirror* y *IA Capture*. La herramienta que se usa en el proyecto es *IA Capture*, que es la aplicación que permite mostrar vídeo en tiempo real sobre la pantalla de la PDA.
- Ordenador personal para la simulación del comportamiento de la Unidad de Control con la que se comunica el subsistema de la PDA. Para cumplir con esta funcionalidad el PC debe de estar dotado de: una plataforma Java adecuada donde puedan ejecutarse los programas servidores, las herramientas adecuadas para el desarrollo de aplicaciones Java que después serán instaladas en la PDA, y una tarjeta *wireless* que permita comunicarse con el *Pocket PC* de manera inalámbrica. Este ordenador también es usado para la instalación y configuración del software de la PDA, por lo que debe disponer de los puertos adecuados para poder realizar la comunicación con tal dispositivo. La comunicación entre PC y el *iPAQ Pocket PC* puede realizarse a través de la base de escritorio universal del *iPAQ Pocket PC* (USB o puerto serie), una conexión inalámbrica por infrarrojos o *Bluetooth*. La instalación de aplicaciones sobre el *Pocket PC* se realiza a través de un PC de sobremesa. Para cumplir con tal objetivo solo es necesario instalar la aplicación *ActiveSync* sobre el PC, y conectar ambos dispositivos correctamente tal y como se detallará en este capítulo.

3.2 HP iPAQ Pocket PC Serie H5500

3.2.1 Introducción

La implementación del software desarrollado en este proyecto ha sido enfocada a su ejecución sobre el dispositivo *HP iPAQ Pocket PC Serie H5500*. Se trata de una PDA de altas prestaciones dotada del sistema operativo *Windows CE*. Sus capacidades en cuanto a memoria y procesador, y sus características de conectividad, hacen de esta un dispositivo óptimo para su inserción en sistemas de cierta relevancia. Dispone de conexión inalámbrica con otros dispositivos mediante los principales protocolos *wireless*, así como entrada/salida de vídeo mediante un módulo de expansión. Así pues, se trata de un producto a través del cual se pueden cumplir los objetivos planteados para este proyecto.

Figura 1. HP iPAQ Pocket PC Serie H5500

A continuación se muestran los aspectos relativos a la PDA de más relevancia para el proyecto.

3.2.2 Descripción del Dispositivo

Especificaciones del Sistema

La tabla que se muestra a continuación contiene las principales características del dispositivo iPAQ Pocket PC, capacidad de procesamiento y memoria, puertos de comunicación, características audiovisuales, etc.

Característica Del sistema	Descripción
Procesador	Intel Xscale a 400MHz
RAM	SDRAM de 128MB
ROM	ROM de 48MB
Ranura SDIO	Memoria SD y soporte de tarjeta SDIO
Pantalla	TFT en color transflectiva de 3'8 pulgadas con 65536 colores, 240x320 pixeles, punto de 0.24 mm.
Sensor de luz de fondo	Ajuste de brillo con varios niveles
Audio	Micrófono, altavoz, toma de auriculares estéreo de 3,5 mm, MP3 estéreo a través de toma de audio.
Infrarrojos	IrDA, transferencia de datos de hasta 115,2Kb por segundo
Bluetooth	Dispositivo de clase II; transmisión de hasta 4 dBm, rango normal de 10 metros.
Lector huellas digitales	Tecnología de gradiente térmico
Comunicaciones	Puerto de comunicaciones, conector de módulo de expansión
Indicadores	3 modos de notificaciones de alarma: Indicador intermitente de color verde, tono, mensaje emergente En carga: indicador intermitente de color ámbar Cargado: indicador fijo de color ámbar Bluetooth activo: indicador intermitente de color azul
Batería	Batería extraíble/recargable de polímetro de iones de litio de 1.250 mAh con batería interna de duración10 minutos para conservar los datos durante la sustitución de la batería principal.

Tabla 1. Especificaciones del sistema

Especificaciones Físicas

En cuanto a las características físicas y dimensiones de la PDA, la siguiente tabla muestra los valores correspondientes.

Longitud	138,0 mm
Anchura	77,0 mm (84,0 en la parte superior)
Profundidad	15,9 mm
Peso	206,8 g

Tabla 2. Especificaciones físicas

Productos Complementarios

Uno de los principales objetivos que se persigue al introducir una PDA en el sistema es que esta ofrezca una interfaz abierta. El modo de interacción de la PDA con el sistema global, u otras características, deben de poder ser ampliadas o modificadas. También es fundamental que se posibilite la adición de nuevas funciones. HP pone a disposición una serie de accesorios para el modelo *iPAQ Pocket PC serie 5500*. Estas son las distintas familias de accesorios:

- Módulos inalámbricos
- Módulos de expansión
- Dispositivos de entrada
- Fundas
- Tarjetas de memoria y de módem
- Baterías
- Cargadores y adaptadores

Es conveniente tener estas posibilidades en cuenta de cara a posibles modificaciones o ampliaciones en la funcionalidad de la aplicación desarrollada. Durante el desarrollo de este proyecto se ha hecho uso de un módulo de expansión para la entrada de vídeo que será detallado más adelante.

3.2.3 Puesta en Marcha del Dispositivo

Para comenzar a usa el dispositivo hay que arrancarlo y seguir una secuencia de acciones. Los pasos a seguir son los que se describen a continuación:

1. Instalación de la batería. Alineación de pantalla mediante el punteo de la misma.
2. Configuración: Reiniciar el dispositivo mediante el puntero, pulsando suavemente el botón de reinicio situado en la parte inferior de la unidad.
3. Insertar el CD proporcionado al adquirir el dispositivo, en el ordenador personal, seguir los pasos que se indican.
4. Instalación del Microsoft Outlook: este paso es necesario si se desea sincronizar correo electrónico, contactos, datos del calendario o tareas del ordenador personal con el Pocket PC.
5. Instalación ActiveSync 3.7. Insertar CD y seguir instrucciones. Este es el software que permite la comunicación entre el PC y el PDA. Más tarde se detallaran sus características y modos de funcionamiento.

6. Conexión del iPAQ Pocket PC al ordenador personal: junto al iPAQ se proporciona una base de escritorio que permite la conexión de la PDA con un ordenador personal mediante el conector USB o el conector serie en el puerto apropiado del ordenador personal. El iPAQ Pocket PC solo debe ser insertado en la base una vez que se haya instalado el software conveniente en el ordenador personal (ActiveSync 3.7). Conectar el adaptador a la base para cargar la unidad. La unidad tarda aproximadamente 4 horas en cargarse completamente.
7. Establecer una asociación software entre el Pocket PC y el ordenador personal. Se permiten dos tipos de asociación, cada una de ellas enfocada a un uso determinado de la PDA.

3.2.4 Software de Conexión PC ⇔ iPAQ Pocket PC: Microsoft Activesync versión 3.7

Esta aplicación es suministrada en el CD para *hp iPAQ PocketPC* y su instalación es sencilla y guiada.

Facilidades que Aporta

- Sincronizar la información entre el *iPAQ Pocket PC* y el ordenador personal o el servidor y de este modo, disponer de la información más actualizada en ambos dispositivos.
- Cambiar la configuración y planificación de la sincronización.
- Copiar archivos entre el dispositivo y el ordenador personal.
- Instalar aplicaciones en el *iPAQ Pocket PC.*
- Realizar copias de seguridad y restaurar la información en el dispositivo.

ConexiónPC ⇔ iPAQ Pocket PC

Hay varias posibilidades: base de escritorio universal (USB o puerto serie), una conexión inalámbrica por infrarrojos o *Bluetooth*. Al conectar *el iPAQ Pocket PC Serie H5500* al ordenador, *ActiveSync* da dos posibilidades de asociación:

- Asociación estándar: conveniente cuando se requiere que los archivos y aplicaciones en ambos dispositivos, ordenador personal y HP *iPAQ Pocket PC* Serie H5500 estén sincronizados.
- Asociación como invitado: opción a seleccionar si tenemos únicamente como propósito el intercambio de archivos y la instalación de aplicaciones. Cuando la opción con sincronización es seleccionada cada cierto tiempo programable la conexión comprueba la sincronización, así como cada vez que conectemos el dispositivo al ordenador personal. En el desarrollo de este proyecto se ha trabajado en modo invitado, de este modo nos ahorramos el tiempo de sincronización.

Al usar el *iPAQ Pocket PC* como una herramienta sobre la cual correr aplicaciones desarrolladas por nosotros mismos, se manejan básicamente las siguientes dos funciones de *ActiveSync:*

Copia de Archivos

Es posible copiar archivos desde el ordenador personal y en éste mediante la opción *Explorer* de *ActiveSync* y mediante el *Explorador de Windows*.

Para copiar archivos:

1. Insertar el *iPAQ Pocket PC* en la base de escritorio universal.

2. En el menú *Inicio* del ordenador, hacer clic en programas *Microsoft ActiveSync.*

3. Hacer clic en *Explorar.*

4. Hacer doble clic en el icono *Mi Pocket PC.*

5. En el ordenador personal, hacer clic con el botón derecho del ratón en el menú *Inicio* y, a continuación, seleccionar *Explorar.*

6. Localizar el archivo que desea mover.

7. Arrastrar los archivos del *iPAQ Pocket PC* y soltarlos en el ordenador, y viceversa, según proceda. *ActiveSync* convierte los archivos para que puedan utilizarse con las aplicaciones de *Pocket Office*, en caso necesario. Una ventana de advertencia es mostrada cuando un archivo a transferir va a ser convertido.

Instalación de Aplicaciones

Para instalar aplicaciones en el *iPAQ Pocket PC* desde el ordenador mediante *ActiveSync:*

1. Conectar el *iPAQ Pocket PC* al ordenador mediante la base de escritorio universal.
2. Seguir las instrucciones que se proporcionan en la aplicación y en el asistente para la instalación.
3. Observar la pantalla del *iPAQ Pocket PC* y comprobar si es necesario efectuar otros pasos para finalizar la instalación de la aplicación.

Aspectos a Tener en Cuenta

Durante el desarrollo del proyecto han surgido ciertos aspectos referentes al uso o las características del PDA que se deben destacar:

- Debido a que la mayoría de aplicaciones y datos que el usuario instala en el *iPAQ Pocket PC* se almacenan en la memoria RAM, será preciso volver a instalarlos si la batería se descarga por completo o si la batería se extrae de la unidad un período de tiempo prolongado. No es necesario volver a instalar las aplicaciones y los datos instalados en la carpeta *Almacenamiento de archivos de iPAQ,* puesto que se guardan en la memoria ROM.
- Utilizar tarjetas *Secure Digital* para realizar copias de seguridad de los archivos importantes almacenados en el *iPAQ Pocket PC.* Otra opción es usar *iPAQ Backup* o *Microsoft ActiveSync Backup* para realizar copias de seguridad y restauraciones de la información.
- Prolongación de la duración de la *batería*: Seleccionando un intervalo de modo de espera inferior, aumentamos la duración de la batería. Si se utiliza el *iPAQ* durante largos períodos de tiempo, conviene seleccionar un intervalo de modo de espera inferior. El uso de la luz de fondo reduce notablemente la vida de la

batería. Se puede ver en todo momento el nivel de carga de la batería, tanto principal como de repuesto, mediante los menús de configuración del PDA.

3.2.5 JeodeRuntime

Pocket PC viene equipado con variedad de aplicaciones, la mayor parte de ellas para uso ofimático. En el CD de instalación para el *Pocket PC* se incluyen algunas aplicaciones adicionales.

El software proporcionado en el CD de instalación para ejecución de aplicaciones Java ha sido diseñado para ser instalado sólo en los modelos *Pocket PC h5500.* Mediante *JeodeRuntime de Insignia*, es posible el uso de Java desde el *hp iPAQ Pocket PC*. *JeodeRuntime* es un entorno de ejecución acelerado compatible con *PersonalJava*. El producto incluye**:**

- Una máquina virtual autorizada por Sun, que permite disfrutar del contenido compatible con *PersonalJava* que se encuentra en la *World Wide Web*, como juegos, seguimiento de noticias personalizadas, información de cotizaciones en bolsa y resultados deportivos en tiempo real, titulares de noticias, calculadoras financieras, puzzles y otros contenidos dinámicos; así como la ejecución de todo tipo de aplicaciones Java escritas bajo el estándar PersonalJava.
- La tecnología *Dynamic Adaptive Compiler* (DAC) de Insignia (en proceso de patente) permite el rendimiento acelerado de aplicaciones Java en dispositivos de capacidad de memoria limitada y es de 6 a 10 veces más rápida que las JVM interpretadas.
- Soporte para tecnología *PersonalJava* completo y funcional.
- Soporte de conexión para *Pocket Internet Explorer.*
- Instalación sencilla a través de *Microsoft ActiveSync.*

Instalación en el PC

Para instalar Jeode*Runtime* en el iPAQ, sólo se tiene que hacer clic en el botón *Instalar* que aparece en la pantalla de guía de instalación y, a continuación, ejecutar el archivo autoextraíble en el PC. Basta con seguir los indicadores para después instalar Jeode*Runtime* en el iPAQ mediante *ActiveSync*.

Cómo Ejecutar Jeode*Runtime* en un Equipo iPAQ PocketPC de Compaq.

Jeode*Runtime* es una implementación certificada de la especificación *PersonalJava 1.2* de Sun.

JeodeRuntime se puede utilizar:

- Como conexión del *Pocket Internet Explorer del iPAQ*, para ejecutar subprogramas de Java desde una página web.
- Como VM (*Virtual Machine*) de Java independiente, para ejecutar aplicaciones de Java en *iPAQ*.

Instalación Sobre el *iPAQ PocketPC*

Como paso previo, la oportuna versión del *Microsoft ActiveSync* debe de haber sido instalada en el *Pocket PC*. Jeode*Runtime* se proporciona como archivo ejecutable. Después de copiar Jeode*Runtime* en el equipo, sólo se tiene que hacer clic en el archivo ejecutable para instalarlo en el *iPAQ*.

Los archivos de Jeode*Runtime* se instalan de forma predeterminada en *Windows* y los subdirectorios de *Windows* del *iPAQ*.

Notas:

- Es necesario disponer de una conexión *ActiveSync* válida entre el *PC Windows* y el *iPAQ.*
- Para equipos con Windows 2000, se debe disponer de privilegios de administrador.

Desinstalación

Para desinstalar Jeode*Runtime* en el iPAQ, seleccionar *Inicio > Configuración*, después seleccionar la pestaña *Sistema,* seleccionar *Quitar programas* y, finalmente, eliminar *Insignia Solutions JeodeRuntime.*

Nota: si la desinstalación no se produce correctamente, reiniciar el iPAQ y volver a intentarlo.

Demostraciones Suministradas

Para demostrar la funcionalidad de Jeode*Runtime*, la distribución incluye un subprograma y una sencilla aplicación de Java. Antes de probar nuestras propias aplicaciones, es conveniente hacer uso de estas demostraciones para tener la certeza de que el funcionamiento del producto es correcto.

- *Quasar*: un subprograma de Java que se invoca abriendo un archivo HTML.
- *PrimTest:* una aplicación de Java que utiliza la biblioteca de gráficos AWT para demostrar algunas prestaciones gráficas de Java.

Para ejecutar las aplicaciones o el subprograma de demostración, seleccionar *Inicio-> Programas -> Jeode -> Ejemplos,* y después seleccionar la aplicación de demostración o el subprograma que ejecutar. Por ejemplo, para ejecutar el subprograma de demostración *Quasar*, seleccionar *Inicio -> Programas- > Jeode- > Ejemplos -> Quasar.*

Ejecución de Subprogramas de Java

La conexión de Jeode para Pocket Internet Explorer se incluye al instalar Jeode*Runtime*. La próxima vez que el explorador abra una página web que contenga un subprograma de Java la conexión lo ejecutará. Todo esto sucede automáticamente, sin necesidad de que el usuario haga nada.

Ejecución de Aplicaciones de Java

Para ejecutar una aplicación de Java mediante Jeode*Runtime*, se debe copiar las clases, bibliotecas y propiedades de la aplicación en el iPAQ.

Jeode*Runtime* se puede ejecutar de dos maneras:

- Puntear el icono de EVM (*Inicio > Programas > Jeode*) y escribir las opciones en el indicador *Introduzca argumentos:* tal como se indica:
  `[opciones-línea-comando] nombre-clase [argumentos-clase]`
  donde:
  - [*opciones-línea-comandos]* son opciones de línea de comandos que pasar a la EVM (por ejemplo, -*cp*).
  - *nombre-clase* es el nombre de la clase para la aplicación de Java (por ejemplo, *PrimTest*).
  - **[***argumentos-clase]* son los argumentos opcionales para que se ejecute la aplicación de Java.

  Por ejemplo:
  -*cp \Windows\lib\jeodedemos.jar PrimTest*

- Ejecutar Jeode*Runtime* como acceso directo. Crear un acceso directo (un archivo *.lnk*) de *evm.exe* situado en el menú *Inicio* y después puntearlo. Eso permite almacenar argumentos de EVM (nombre de clase y opciones de línea de comandos) en las *Propiedades* del acceso directo, con lo cual no hace falta volver a escribirlos cada vez que se ejecute Jeode*Runtime*.

Creación de Accesos Directos

Para crear un acceso directo, utilizar el *Explorador de Windows* del equipo para ir a la ventana *Dispositivos móviles -> Mi Pocket PC -> Windows*, seleccionar *evm.exe*, seleccionar el menú *Editar -> Copiar* y seleccionar *Pegar como acceso directo.*

Esto crea un archivo llamado *Acceso directo a evm.exe.lnk* en el iPAQ. Copiar este archivo en un directorio del PC y abrirlo con editor de texto como el Bloc de notas. El contenido del archivo será similar al siguiente:

18#"\Windows\evm.exe"

Ahora se puede editar este archivo para agregar parámetros de línea de comandos. Así, para ejecutar *PrimTest*, que se suministra como ejemplo, el archivo se puede editar de esta manera:

18#"\Windows\evm.exe" -cp \Windows\lib\jeodedemos.jar PrimTest

Se puede agregar otras opciones de línea de comandos que se necesiten al acceso directo, hasta un máximo de 255 caracteres.

Ahora se puede copiar el acceso directo otra vez al iPAQ y después puntearlo para ejecutar la aplicación, *PrimTest* en el caso del ejemplo.

Se pueden usar los accesos directos de Jeode*Runtime* (en *\Windows\Menú Inicio \Programas\Jeode*) como ejemplo.

Opciones de Línea de Comandos

Esta sección describe algunas opciones de línea de comandos habituales de Jeode*Runtime* que puede pasar a la EVM de Jeode al ejecutarla mediante la línea de comandos o al configurar un archivo *.lnk.*

- -?
- -classpath o -cp

- -D
- -v o -verbose
- -version
- -Xnowinceconsole

-?, -h o -help
Muestra ayuda sobre todas las opciones de línea de comandos de Jeode*Runtime*.
La mejor manera de verla es con la opción *-Djeode.evm.console.local.paging=TRUE.*

-classpath **<***nombres-ruta***>** o *-cp* **<***nombres-ruta***>**
Especifica la ruta o rutas que se usan para cargar clases de aplicaciones. Los *nombres-ruta* se separan con caracteres de punto y coma.
Por ejemplo, para incluir clases que están en *jeodedemos.jar*, se puede usar el comando: *-cp \Windows\lib\jeodedemos.jar*

*-D***<***NombrePropiedad***>=<***valor***>**
Proporciona el valor para una propiedad de Jeode*Runtime* o estándar de sistema Java. Por ejemplo, para mantener abierta la consola de visualización, use el comando: -Djeode.evm.console.local.keep=TRUE

-v, -verbose
Hace que los mensajes se muestren en la consola de Jeode*Runtime* cuando:

- Se carga correctamente un archivo de clase o biblioteca dinámica.
- Se realiza un ciclo de recogida de basura.

-version
Muestra información sobre la versión de Jeode*Runtime* en la consola EVM. Si tecleamos esta opción en el modelo usado para este proyecto el resultado es:

Jeode EVM Versión 1.9.3

Supported Java PLataforms: PersonalJava1.2

-Xnowinceconsole
Use la opción *-Xnowinceconsole* para deshabilitar la consola EVM si no desea verla, por ejemplo al ejecutar aplicaciones de gráficos.

Problemas o Dudas que Pueden Surgir

¿Por qué no funciona un subprograma ? Dos son los motivos habituales por los que un subprograma de Java quizá no funcione:

- Subprogramas en archivos CAB.
  Actualmente, Jeode*Runtime* no admite la carga de subprogramas de archivos CAB de Microsoft, ya que los archivos CAB no forman parte de la tecnología estándar de Java. Así que si, el sitio web asume (según el tipo de explorador) el comportamiento de la VM que se está usando, puede provocar problemas en la carga del subprograma.
- Subprogramas en archivos JAR o ZIP.
  Jeode*Runtime* admite el uso de subprogramas contenidos en archivos JAR o ZIP.

Sin embargo, debido a que *Pocket Internet Explorer* es compatible con la especificación *HTML 3.2*, éste no admite el uso del atributo *ARCHIVE* de la etiqueta

APPLET en la página HTML que llama al subprograma; por lo tanto, en esas circunstancias *Pocket Internet Explorer* no admitirá la carga de subprogramas de archivos JAR o ZIP.

Si se quiere reescribir el HTML de la página web que llama al subprograma, se pueden usar los archivos JAR y ZIP especificando el contenedor como nombre de parámetro así como atributo ARCHIVE (mediante *<param name=archive>*). Por ejemplo: *<param name=archive value=myclasses.jar>*

Un ejemplo de esta técnica se muestra en *\Windows\lib\Quasar.html*, el archivo HTML que llama al subprograma de demostración Quasar.

¿Por qué aparece el mensaje "*Failed to run applet"* ? Se puede ver un mensaje del tipo:

```
Failed to run applet. Please Reload
```

Esto significa que la conexión ha detectado una condición que impide la ejecución del subprograma. Hay varias posibilidades: un problema habitual es la falta de memoria en el iPAQ, ya que algunos subprogramas están escritos asumiendo que habrá una cantidad ilimitada de memoria.

La conexión Jeode no tiene control sobre los requisitos de memoria del subprograma. Si no queda suficiente memoria para que la conexión siga ejecutándose, emitirá un mensaje para el usuario y se cerrará.

¿Funciona Jeode*Runtime* con *JavaScript*? *JavaScript* es un lenguaje de secuencias de objetos para la creación y personalización de aplicaciones. Mientras que Java sólo lo usan programadores para crear objetos y subprogramas nuevos, *JavaScript* está diseñado para que lo usen autores de páginas HTML para secuenciar dinámicamente el comportamiento de objetos que se ejecutan en el cliente o en el servidor. *JavaScript* y Java son lenguajes de programación completamente distintos; por lo tanto, Jeode no se utiliza para ejecutar *JavaScript.*

Además, la conexión Jeode actualmente no admite las comunicaciones entre *JavaScript* y subprogramas de Java. Eso significa que los subprogramas que requieren esta interacción no funcionarán con Jeode*Runtime*.

¿Se admite Swing? Lamentablemente, Swing 1.1.1 no es compatible con PersonalJava 1.2; por lo tanto, Jeode*Runtime* no admitirá Swing 1.1.1. El problema se debe a un test incorrecto en Swing 1.1.1, tal como se describe en el *bugID 4309057* del sitio web *Java Developer Connection* de Sun.

¿Se pueden eliminar algunos archivos de JeodeRuntime para ahorrar memoria?

El conjunto mínimo de archivos depende de la aplicación.

Si un subprograma o aplicación no requiere compatibilidad con entorno nacional internacionalizado, se puede eliminar el archivo *i18n_a.jar*. Además, si la aplicación no usa gráficos (AWT), se pueden eliminar los archivos *evmawt.dll* y *awt.jar.* Los dos archivos *.jar* están almacenados en el subdirectorio *lib* del directorio donde haya instalado Jeode*Runtime*, por ejemplo, *\Windows\lib*. El archivo *evmawt.dll* está almacenado en el directorio de nivel superior al de Jeode*Runtime*, por ejemplo, *\Windows.*

Si después de eliminar los archivos anteriores se intenta ejecutar un subprograma o aplicación que requiera alguna de estas funcionalidades, estos no funcionarán correctamente y pueden generar el mensaje de error correspondiente. Para sustituir estos archivos una vez se hayan eliminado, se debe reinstalar Jeode*Runtime*.

¿Cómo evitar que la consola se cierre cuando EVM finaliza? Utilizar la opción de línea de comandos *-Djeode.evm.console.local.keep=TRUE*. Por ejemplo, para ejecutar la aplicación de demostración PrimTest y evitar que la consola se cierre, se puede usar una invocación de línea de comandos similar a esta:

```
-Djeode.evm.console.local.keep=TRUE    -cp    \Windows\lib\jeodedemos.jar
PrimTest
```

También puede usar la opción de línea de comandos:

Djeode.evm.console.local.paging=TRUE

Es una manera parecida para habilitar la paginación de consola. Esto resulta útil si la aplicación envía más de una pantalla de información a la consola, por ejemplo al mostrar la ayuda de Jeode*Runtime* con la opción *-?*

¿Por qué el programa no encuentra un archivo de datos? Algunas aplicaciones de Java buscan archivos (datos, preferencias, etc.) en un directorio de trabajo actual asumido, en lugar de especificar un nombre de ruta absoluto. En los iPAQ, el directorio de trabajo actual siempre se asume que es el directorio raíz (\), así que esto puede causar problemas en algunas aplicaciones de Java.

¿Por qué aparece el mensaje '*NoClassDefFoundError*'? Por lo general, este tipo de error significa que Jeode*Runtime* no puede encontrar una clase que está buscando, así que por ejemplo puede haber olvidado especificar una ruta de clase (mediante la opción de línea de comandos *-classpath o -cp*).

Sin embargo, si obtiene un mensaje del tipo:
Exception in thread "main", java.lang.NoClassDefFoundError: LocalizedError at ConvertEncoding.main (bytecode 146)

quizás sea necesario instalar la compatibilidad de internacionalización adicional (el archivo *i18n_b.jar*).

Notas Sobre Internacionalización

Hay dos archivos JAR que proporcionan compatibilidad para la internacionalización.

- *i18n_a.jar*
  Este archivo se instala en su dispositivo iPAQ de forma predeterminada (pero se puede eliminar para ahorrar espacio si fuera necesario). Este archivo contiene las partes más usadas de la compatibilidad para internacionalización, entre las que cabe citar:
  - Formato de fecha y de hora internacional.
  - Zona horaria.
  - Símbolos de moneda internacionales.
  - Información sobre ordenación de caracteres.
- *i18n_b.jar*
  Este archivo no se instala en su dispositivo iPAQ, pero se almacena en el directorio Jeode*Runtime* del PC, por ejemplo en: *C:\Archivos de programa \Insignia Solutions Plc\JeodeForIPAQInstaller\i18n_b.jar*. Este archivo contiene partes de la compatibilidad para internacionalización menos utilizadas, que consisten en la codificación de caracteres y de tablas de conversión.

Como *i18n_b.jar* no se suele utilizar, no lo hemos instalado en el iPAQ, para minimizar los requisitos de memoria de Jeode*Runtime*; así, podrá asignar más memoria de iPAQ para ejecutar aplicaciones y subprogramas.

Si se necesita la compatibilidad adicional que proporciona *i18n_b.jar*, sólo hay que copiar el archivo del PC al subdirectorio *lib* del producto Jeode instalado (por ejemplo, *\Windows\lib)* del iPAQ. Esta compatibilidad estará disponible a partir de entonces la próxima vez que ejecute Jeode*Runtime*.

Normalmente, sólo se necesitará *i18n_b.jar* para ejecutar aplicaciones que precisen convertir archivos de un juego de caracteres a otro (por ejemplo, para convertir Shift-JIS a Unicode).

Comentarios Adicionales

- *MyDocuments* solo permite un nivel de directorios, así que si, por ejemplo, queremos usar un Applet que use un directorio “Imagenes”, dicho *applet* no debe de guardarse en *MyDocuments* sino por ejemplo en el *directorio Archivos de Programa*

3.3 FlyJacket i3800

3.3.1 Introducción

Uno de los servicios que ofrece la PDA es la capacidad de mostrar y almacenar vídeo. Para este fin se ha adquirido el dispositivo de expansión multimedia **FlyJacket i3800.** El FlyJacket i3800 permite conectarse a un proyector, a un monitor VGA o a un televisor para presentaciones directamente desde un PC de bolsillo iPAQ y seguir teniendo acceso a la ranura *CompacFlash*. También pueden obtenerse instantáneas desde una cámara de vídeo o una cámara estática digital mediante la función de entrada de vídeo. La pila interna proporciona al *FlyJacket i3800* electricidad y se puede recargar directamente desde el adaptador de corriente de iPAQ.

Figura 2. FlyJacket i3800

3.3.2 Características del Dispositivo

Funciones

- Salida VGA: resolución de hasta 1024*768
- Salida de vídeo: estándares NTCS y PAL
- Entrada de vídeo: dispositivos compuesto y S-Vídeo
- Captura de imagen parada
- Vista preliminar en vídeo
- *Ranura CompacFlash* incorporada (compatible con Tipo I/II)
- Mando a distancia/Puntero infrarrojo de tamaño bolígrafo
- Pila interna

Contenido del Paquete

- Dispositivo de expansión *FlyJacket*
- Mando a distancia/puntero de tamaño bolígrafo
- Un cable de adaptador eléctrico tipo Y
- Guía de usuario
- CD-ROM. Utilidad *FlyJacket* y paquete de software de la aplicación
- Software de la *Suite IA Style PowerMedia* y Guía de usuario

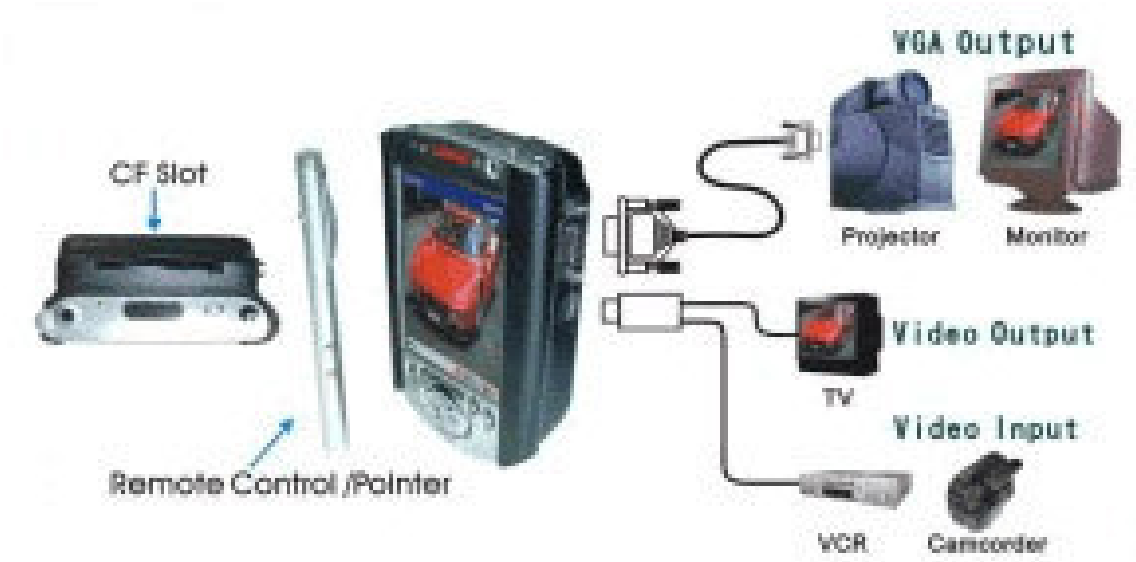


*Figura 3.*Conexionado del FlyJacket i3800

3.3.3 Puesta en Marcha del Dispositivo

Instalación del Controlador y del Hardware

Instalación del software del controlador:

1. Establecer una conexión entre el iPAQ y el PC.
2. Introducir el CD de *FlyJacket* en la unidad de CD-ROM del ordenador.
3. Aparecerá el programa de instalación de *FlyJacket*. Hacer clic en *FlyJacket Utility* para iniciar la instalación del programa de utilidad y del controlador.
4. La primera ventana que aparecerá indicará los archivos que se instalarán. Hacer clic en *siguiente* para continuar.
5. Leer y aceptar el contrato de licencia.
6. Aparecerá un cuadro sugiriendo la instalación en un directorio predeterminado. Seleccionar el directorio deseado.
7. Seguir todas las instrucciones de la pantalla para finalizar la instalación.

Conexión por primera vez:

1. Deslizar el PC de bolsillo iPAQ en el *FlyJacket i3800* hasta que se ajuste y se oiga el tono de “conectado”.
2. En caso de mensaje de error volver a repetir los pasos 2-7.

Instalación del Software

El paquete de software del *FlyJacket i3800* está compuesto por los siguientes elementos:

- Software de *FlyJacket Utility*
- Software de la *Suite IA Style PowerMedia*

La *FlyJacket Utility* está compuesta por las aplicaciones *LifeView Shadow* y *LifeView PowerShow*. Ambas se instalarán de forma automática durante el proceso de instalación del software del controlador.

La aplicación *Life View Shadow* del dispositivo de expansión multimedia *FlyJacket i3800* es un programa de configuración de salida de vídeo. La utilidad del mismo es establecer la configuración de la pantalla de salida de un dispositivo de pantalla de salida VGA/Vídeo, como por ejemplo un proyector, un monitor o un televisor. La aplicación *Life View Shadow* detectará de forma automática si el FlyJacket i3800 está conectado o no. Si el *FlyJacket i3800* no está conectado, la aplicación se cerrará, y aparecerá la configuración de la pantalla predeterminada. *Life View Shadow* no funcionará cuando la *Suite IA Style PowerMedia* esté abierta.

El *FlyJacket i3800* posee na pila Li-on que puede recargarse con el mismo adaptador que el PC de bolsllo iPAQ. Para ver el estado de la vida de la batería de FlyJacket i3800 se utiliza la aplicación *LifeView PowerShow. LifeView PowerShow* muestra un medidor de barras horizontal. Cuanto más llena esté la barra, mayor será la vida de la pila. Cuando la barra esté cerca o al 10%, se deberá recargar la pila.

3.3.4 Herramientas Multimedia Disponibles

Instalación de la Suite IA Style PowerMedia

La *Suite PowerMedia* es un conjunto potente compuesto por cuatro aplicaciones de la IA Style diseñadas para aprovechar al máximo las capacidades multimedia del *FlyJacket i3800:*

- *IA Album*: administrador de álbum digital de fotos y utilidades de captura de pantalla.
- *IA Presenter:* conversión y presentación de diapositivas PowerPoint para salida VGA/TV.
- *IA Screen Mirror:* proyección de una exacta emulación de la pantalla del Pocket PC's para salida VGA/TV.
- *IA Capture*: captura de instantáneas y video a través de una cámara adjunta al dispositivo.

Para las necesidades del proyecto únicamente es necesario instalar la herramienta *IA Capture*. Una vez instalada correctamente debe de poder observarse en la sección *Programas* del Pocket PC el siguiente icono:

Figura 4. Icono de IA Capture: Aplicación de captura de vídeo

La recepción de vídeo en el PDA se llevará a cabo a través de esta aplicación. Pasamos pues, a una descripción más detallada de la herramienta.

IA Capture es una aplicación de captura de vídeo e imagen diseñada para una completa utilización del hardware de entrada de vídeo para el PDA. La ventana digital en la porción superior de la pantalla permite ver la entrada del flujo de vídeo activo proveniente de una cámara adjunta o de un canal de entrada de vídeo. Es posible capturar una única instantánea de la entrada de vídeo, o un vídeo clip completo. Los vídeo clips capturados pueden ser reproducidos inmediatamente usando IA Capture.

Las prestaciones de IA Capture van más allá de la captura vídeo e imágenes. Es posible elegir el tipo de archivo/calidad, controlar la reproducción, ver la información de los archivos, seleccionar el método de administración de memoria durante la grabación de vídeo, orientar el display de salida en sentido vertical/horizontal, o a pantalla completa, entre otras opciones...

Para comprobar la correcta recepción de la entrada de vídeo con *IA Capture,* seleccionar el botón *Options* y a continuación seleccionar las características en los dos campos de la pestaña *General* que se muestra a continuación en la figura 5. A continuación, la figura 6 muestra la pantalla Capture Mode de IA Capture.

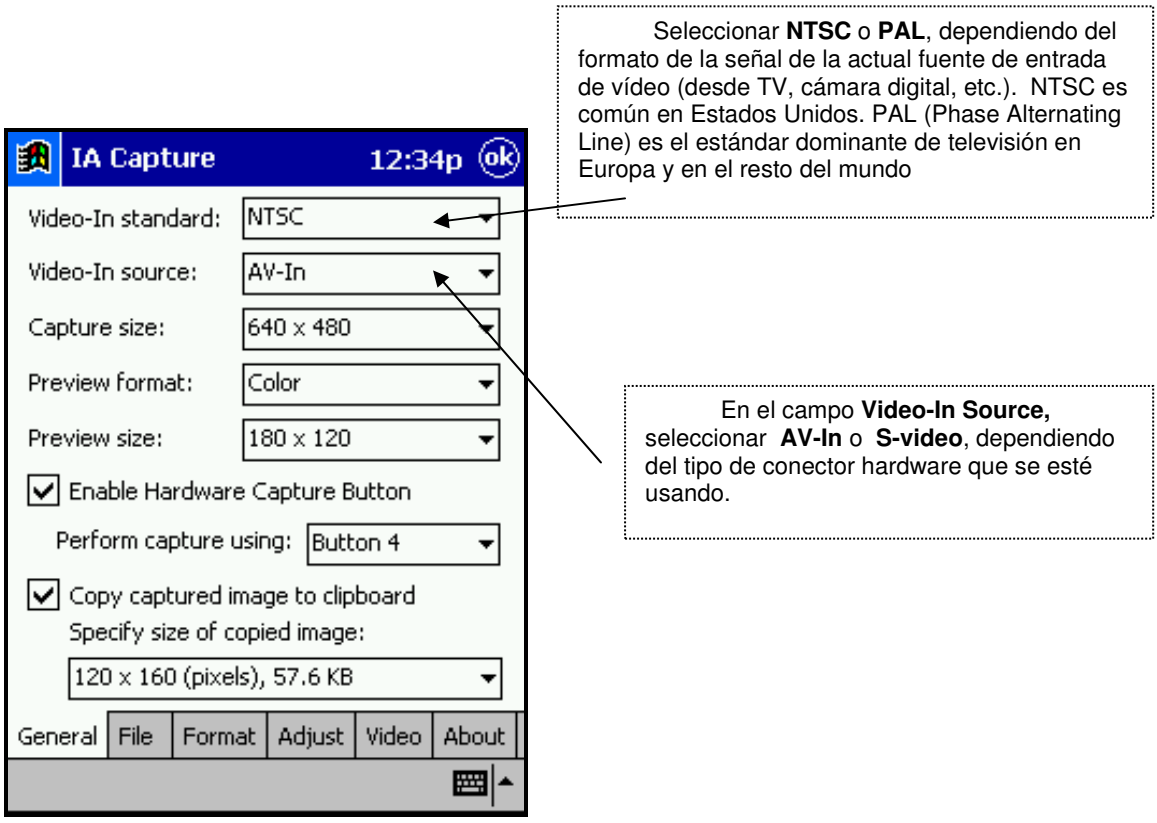


Figura 5. Panel de configuración de captura de vídeo

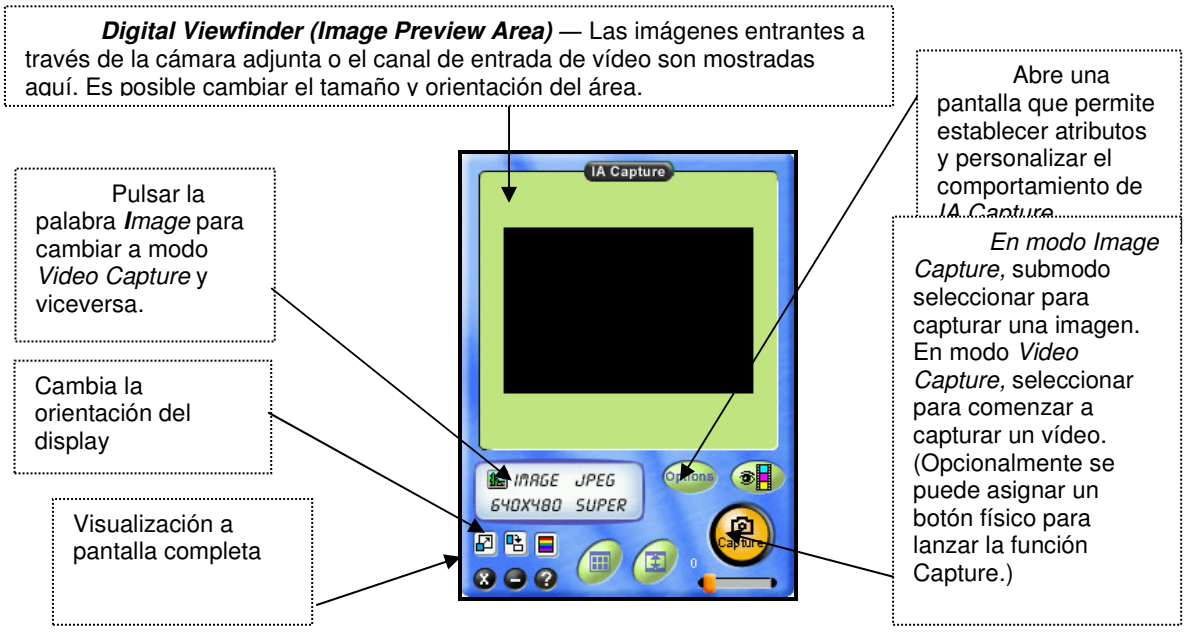


Figura 6. Ventana de visualización de IA Capture

Después de haber especificado la señal de la fuente de entrada de vídeo en el campo *Video-In Standard* de la pestaña *General*, y seleccionado el tipo de conector hardware que se está usando en el campo *Video-In Source***,** se mostrará el flujo de vídeo entrante en la *Digital Viewfinder* (*Image Preview area).*

Cuando el programa se encuentra en modo *Video Capture*, el icono indicador de modo mostrará la palabra "Video", tal y como se muestra en la figura inferior.

Figura 7. Icono para la muestra de vídeo

3.4 Tecnología Wireless

3.4.1 Introducción a la Tecnología Wireless

El dispositivo adquirido, iPAQ Pocket PC h5500, dispone de funcionalidad *wireless* a través de dos protocolos: *Bluetooth* y WLAN 802.11b. En primer lugar vamos a estudiar los fundamentos de la tecnología inalámbrica. Veremos aspectos tales como los elementos característicos de esta tecnología, modos de funcionamiento y utilidades, entre otros. Una vez establecidos los conceptos básicos de la comunicación *wireless* pasaremos a describir en detalle su configuración, capacidad y modo de empleo sobre la PDA. Respecto a los dos protocolos de comunicación inalámbrica soportados por la PDA, haremos una descripción completa de todas sus funciones para tener un visión completa de todas las posibilidades de las que disponemos. Más adelante, cuando se implemente la aplicación Java sobre la PDA, únicamente se hará uso de algunas de estas posibilidades.

¿Qué es una red inalámbrica?

Una red inalámbrica es, básicamente, una red que permite interconectar equipos sin el uso de cables. La principal ventaja es poder compartir recursos (como en una red normal), pero desechando el uso de cableado. Este hecho conlleva una serie de ventajas. Ejemplo de ventajas en una universidad:

1. El ahorro de dinero que supone la instalación de todo el cableado.
2. La posibilidad de que profesores y alumnos accedan a la red con sus propios portátiles (o *wap, palm*... en el futuro) una vez dentro del centro, con total movilidad dentro de la zona de alcance de la red inalámbrica.

Si se dispone de una red estándar, es posible combinar nodos inalámbricos con la red de cable. La figura 8 muestra una red inalámbrica instalada en un edificio a la cual se puede acceder desde el exterior.

Ejemplo de red inalámbrica.

1 NODO - OMNIDIRECCIONAL - SIN CABLE

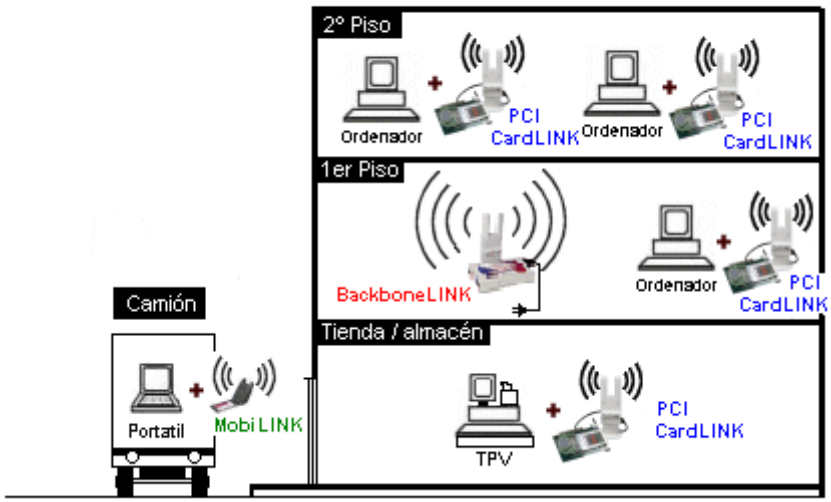


Figura 8. Red inalámbrica

Es importante familiarizarse con algunos términos propios de la tecnología wi*reless*:

Punto de acceso (Access Point): Desempeña básicamente la función de un HUB de red normal: es un dispositivo que 'gestiona' los paquetes lanzados por otras estaciones inalámbricas, haciéndolas llegar a su destino. Además el punto de acceso, da conectividad a una red cableada, por lo que la red inalámbrica puede acceder a otros equipos que estuvieran en una red cableada.

Tarjetas WIFI: Las más conocidas son las que vienen en formato PCMCIA, para portátiles, aunque también las hay en formato PCI (PC normal), en *CompactFlash, Smart Card* y similares. Son equivalentes a una tarjeta de red normal, sólo que sin cables. Disponen de un Rx/Tx interno. Su configuración a nivel de IP es exactamente igual que una *Ethernet.* Cada tarjeta dispone de una dirección física única en el mundo. Las diferencias más importantes entre una tarjeta WIFI y una tarjeta *Ethernet,* (a parte de que las primeras no llevan cable) son: el cifrado de datos, el ESSID, el Canal, y el ajuste de velocidad.

En cuanto a la distancia máxima entre tarjetas, lo más importante es una línea visual clara entre los dos dispositivos a comunicar. Evidentemente la distancia depende de la antena. En comunicaciones *Ad-hoc* (punto a punto entre dos dispositivos con capacidades *wireless*) se alcanzan distancias de hasta 30 metros dentro de un edificio. Cuantos menos obstáculos haya entre los dos dispositivos a mayor distancia podrá realizarse la comunicación.

Antenas: Antenas Direccionales y las antenas *omnidireccionales*. Dentro del grupo de antenas direccionales, tenemos las de *Rejilla o Grid*, las *Yagi*, las parabólicas, las "*Pringles*" y las de *panel*. Las omnidireccionales suelen ser una simple varilla vertical. Cuanta más alta sea la ganancia de la antena, mayores distancias podremos cubrir, y con mejor calidad podremos captar señales que pudieran llegarnos muy débilmente. Algunas distancias conseguidas con antenas son:

- *Grid de 24dB:* 70,5Km - *Grid de 19dB:* 54Km
- *Omnidireccional de 8dB:* 25 Km y al otro extremo una de 19dB grid. A 10Km el enlace a 11Mbps, y a esa misma distancia conectar entre 2 Omnis a 2Mbps.

El Pigtail: El *Pigtail*, no es más que un pequeño cable, que sirve de adaptación entre la tarjeta WIFI y la antena o el cable que vaya hacia la antena. Este Pigtail tiene 2 conectores: el propietario de cada tarjeta en un extremo, y por el otro un conector N estándar en la mayoría de los casos. El *pigtail* depende del fabricante de la tarjeta, por lo que no es una cosa estándar, aunque el más conocido es el compatible con las tarjetas AVAYA y ORINOCO. El uso de este cable es imprescindible para conectar una antena a la tarjeta, salvo en algunos modelos de antenas diseñadas expresamente para usar en interiores, que ya vienen con ese conector de serie. Del cable depende que la señal llegue correctamente desde la tarjeta a la antena, y viceversa.

Los conectores: Básicamente se van a usar los conectores N para las antenas (salvo marcas poco extendidas), tanto en macho como hembra. De ellos depende la calidad de un buen enlace. Una mala soldadura, un conector de baja calidad, puede introducir una cantidad importante de pérdidas que hagan imposible establecer un enlace. Los conectores también tienen pérdidas, no por el conector en sí, sino por el enlace entre el cable y el conector: el estaño, mala sujeción, mala calidad de ambos, etc.

Modos de Funcionamiento

Tanto las tarjetas como los *Access Point* tienen diversas formas de trabajar, las más conocidas son *AD-HOC* e *Infrastructure:*

- **AD-HOC:** Una red "*Ad Hoc"* consiste en un grupo de ordenadores que se comunican cada uno directamente con los otros a través de las señales de radio sin usar un punto de acceso. Las configuraciones "Ad Hoc" son comunicaciones de tipo punto-a-punto. Los ordenadores de la red inalámbrica que quieren comunicarse entre ellos necesitan configurar el mismo canal y ESSID en modo "*Ad Hoc*". El ESSID es un identificador de red inalámbrica. Es algo así como el nombre de la red, pero a nivel WIFI.
- **Infrastructure:** Esta es la forma de trabajar de los puntos de acceso. Si queremos conectar nuestra tarjeta a uno de ellos, debemos configurar nuestra tarjeta en este modo de trabajo. Sólo decir que esta forma de funcionamiento es bastante más eficaz que AD HOC, en las que los paquetes "se lanzan al aire, con la esperanza de que lleguen al destino.", mientras que I*nfrastructure* gestiona y se encarga de llevar cada paquete a su sitio. Se nota además el incremento de velocidad con respecto a AD HOC. Otros conceptos a tener en cuenta son:

WEP: Se puede habilitar o deshabilitar WEP y especificar una clave de encriptación. *Wired Equivalent Privacy (WEP)* proporciona transmisión de datos "segura".

PS Mode: Se puede habilitar la función de ahorro de energía (*Power Saving)* para ahorrar batería en los portátiles cuando no se esté usando la red.

Channel: Cuando un grupo de ordenadores se conectan a través de radio como una red inalámbrica independiente (*Ad Hoc),* todas las estaciones deben usar el mismo canal de radio. Aunque si te conectas a una red a través de un punto de acceso (modo infraestructura), entonces la tarjeta de red se configura automáticamente para usar el mismo canal que usa el punto de acceso más cercano.

Tx Rate: es la velocidad del enlace. Por defecto se ajusta automáticamente en función de la calidad de la señal, aunque se puede forzar a mano. Es recomendable dejarla automática.

3.4.2 Bluetooth En iPAQ Pocket PC

El *iPAQ Pocket PC Serie 5500* está equipado con tecnología *Bluetooth* integrada, que permita realizar conexiones a corta distancia y ofrece una comunicación inalámbrica rápida, fiable y segura.

Con *Bluetooth* activado, se puede enviar información o efectuar las tareas siguientes de forma inalámbrica entre dos dispositivos *Bluetooth*, dentro de un límite aproximadamente 10 metros:

- Intercambiar contactos, elementos de calendario y tareas
- Enviar o intercambiar tarjetas de presentación
- Transferir archivos
- Sincronizar datos con un ordenador a través de una conexión ActiveSync
- Asociarse con un teléfono móvil compatible *Bluetooth* y utilizarlo como módem inalámbrico
- Conectarse a otros dispositivos *Bluetooth* (Puerto COM virtual)
- Conectarse a redes de áre*a local inalámbricas (WLAN) Bluetooth*
- *Imprimir en un impresora Bluetooth*
- Crear una red de área personal (PAN) para conversar en línea, ejecutar juegos, etc
- Utilizar auriculares *Bluetooth*

3.4.2.1 Funciones básicas con Bluetooth

Antes de utilizar *Bluetooth* para establecer conexiones inalámbricas a través del *iPAQ Pocket PC*, conviene familiarizarse con varios aspectos:

- La terminología utilizada
- Servicios compatibles
- La configuración de *Bluetooth*
- *Bluetooth Manager*

Terminología

- Autenticación: Verificación de una clave de acceso de paso numérica para completar una conexión o actividad
- Autorización: Aprobación de una conexión o actividad antes de completarla
- Enlace (dispositivos emparejados): Permite crear una conexión fiable entre su dispositivo y otro dispositivo. Una vez que se ha creado un enlace, los dos dispositivos se emparejan. Un dispositivo emparejado, no precisa autenticación ni autorización
- Dirección de dispositivo: Dirección electrónica exclusiva de un dispositivo *Bluetooth*
- Descubrimientos de dispositivos: Localización y reconocimiento de otros dispositivo *Bluetooth*
- Nombre de dispositivo: Nombre que proporciona un dispositivo *Bluetooth* cuando lo detecta otro dispositivo
- Cifrado: Método de protección de datos
- Clave de vínculo: Código que se introduce para autenticar las conexiones que solicitan otros dispositivos
- Sistema PIM (*Personal Information Manager)* Conjunto de aplicaciones utilizadas para administrar tareas profesionales diarias
- Perfiles: Conjunto de dispositivos *Bluetooth*
- Descubrimiento de servicios: Detección de aplicaciones en común con otros dispositivos.

Servicios Compatibles

Las funciones compatibles con *Bluetooth* se denominan servicios. Sólo puede comunicarse con dispositivos *Bluetooth* que sean compatibles con al menos uno de los servicios siguientes

- BPP (perfil de impresora básica)
- DUN (perfil de redes telefónicas)
- FAX
- FTP (perfil de transferencia de archivos)
- GAP (perfil de acceso genérico)
- HCRP (perfil de sustitución de cable hardware)
- LAP (perfil de acceso a LAN)
- OBEX(perfil de intercambio de objetos)
- OPP (Perfil de introducción de objetos)
- PAN (perfil de red de área personal)
- SPP (perfil de puerto serie)
- *ActiveSync* (utiliza SPP para conectarse a *ActiveSync* en el ordenador)
- AGP (perfil de puerta de enlace de audio)

3.4.2.2 Cómo Trabajar con la Configuración Bluetooth

Desde las fichas de configuración de *Bluetooth*, se pueden realizar las siguientes acciones:

- Activar y desactivar *Bluetooth*
- Especificar o cambiar el nombre *Bluetooth* del *iPAQ Pocket PC*
- Establecer las preferencias de conexión
- Activar los servicios *Bluetooth*
- Especificar valores de configuraciones de la seguridad
- Definir valores para compartir y conectarse
- Seleccionar un perfil de usuario
- Visualizar información de software y puertos

Para abrir la configuración de *Bluetooth:*

1. En pantalla *Hoy*, puntear en el icono de *Bluetooth*
2. En el menú emergente, puntee en la opción *Configuración de Bluetooth*

Para activar *Bluetooth* desde la configuración de *Bluetooth:*

1. En la pantalla *Hoy*, puntear en el icono *Bluetooth*
2. En el menú emergente, puntear en la opción *Configuración de Bluetooth*
3. Puntear en el botón de Encender para activar Bluetooth

Existen dos accesos directos para activar *Bluetooth*:

- En el menú emergente, puntear *Active Bluetooth*
- En la pantalla Hoy , puntear *Inicio>Bluetooth Manager* para activar *Bluetooth* automáticamente

Cuando *Bluetooth* está activo, el color del icono de *Bluetooth* pasa a ser azul, y el indicador de la izquierda del *iPAQ Pocket PC* parpadea en color azul. Para ahorrar batería, se recomienda activar *Bluetooth* solamente cuando se vaya a utilizar.

Puntear en el botón Apagar para desactivar *Bluetooth*. Cuando se desactiva *Bluetooth*, el icono pasa a ser inactivo y aparece una X de color rojo, tras la cual no es posible realizar conexiones de entrada ni salida

3.4.2.3 Establecimiento de Propiedades de Accesibilidad

Se pueden introducir o cambiar propiedades de accesibilidad para definir cómo va a interactuar el *iPAQ Pocket PC* con otros dispositivos *Bluetooth.*

Introducción de un Nombre de Dispositivo

El nombre del dispositivo es lo que ven otros dispositivos cuando lo localizan.

Para especificar un nombre de dispositivo:

1. En la pantalla hoy, puntear en el icono de *Bluetooth>Configuración de Bluetooth> ficha Accesibilidad*
2. Resaltar el nombre en el campo *Nombre* y especificar uno nuevo
3. Puntear en OK para guardar los cambios

Cómo Permitir que se Establezcan Conexiones

Se puede especificar si se pueden conectar todos los dispositivos o únicamente los emparejados con el *iPAQ Pocket PC.*

Cómo Permitir que se Conecten Todos los Dispositivos

1. En la pantalla *Hoy*, puntear en el icono de *Bluetooth> Configuración de Bluetooth> ficha Accesibilidad*
2. Seleccionar *Permitir que se conecten otros dispositivos* y, a continuación, *Todos los dispositivos*
3. Puntear en OK

Cómo Permitir que se Conecten los Dispositivos Emparejados

Los dispositivos emparejados comparten e intercambian una clave de vínculo generada internamente antes de la conexión. La clave de vínculo se obtiene a partir de una dirección exclusiva de dispositivo *Bluetooth*, un número aleatorio y una contraseña definida por el usuario. Con esta opción, únicamente los dispositivos que conoce pueden conectarse al *iPAQ Pocket PC.*

Para permitir que los dispositivos emparejados se conecten:

1. En la pantalla *Hoy*, puntear en el icono de *Bluetooth> Configuración de Bluetooth> ficha Accesibilidad*
2. Seleccionar Permitir que se conecten otros dispositivos y, a continuación, Solo dispositivos emparejados
3. Puntear OK

Cómo Permitir a Otros Dispositivos Localizar el iPAQ

Se puede permitir a otros dispositivos buscar y localizar el *iPAQ Pocket PC*

Para permitir a otro localizar el dispositivo:

1. En la pantalla *Hoy*, puntear en el icono *de Bluetooth> Configuración de Bluetooth> ficha Accesibilidad*
2. 2. Seleccionar Otros dispositivos pueden descubrirme
3. 3.Puntear en OK para guardar los cambios

En el caso de que otro dispositivo remoto tenga la dirección de nuestro dispositivo, es posible que dicho dispositivo remoto localice nuestro *iPAQ Pocket PC* y se conecte a él aunque no se haya seleccionado la opción que permita descubrir el dispositivo.

Activación de los Servicios Bluetooth

Se puede definir que determinados servicios se activen:

- Automáticamente al activar *Bluetooth*
- Cuando la conexión lo permita
- Cuando se especifique una clave de paso o una clave de vínculo correctamente

Se pueden utilizar cualquiera de las opciones de seguridad al transferir archivos, al crear una conexión de puerto serie, al intercambiar información de tarjetas de presentación, al configurar redes telefónicas y al unirse a una red personal.

Activación Automática de Servicios

Es posible permitir que los dispositivos se conecten con autorización para que *Bluetooth* esté listo para la conexión siempre que esté activado

Para activar servicios automáticamente

1. En la pantalla *Hoy*, puntear en el icono de *Bluetooth> Configuración de Bluetooth*
2. Puntear en la ficha correspondiente al servicio que se quiere activar: transferencia de archivos, puerto serie, servidor de red personal o puerta de enlace audio
3. Seleccionar Activar servicio
4. Asegúrarse de que las casillas de verificación Se necesita autorización o Se necesita autenticación (contraseña) no estén marcadas
5. Puntear OK

Solicitud de Autorización Para Acceder a Servicios

Si se opta por solicitar autorización para acceder a servicios, deben autorizarse todas las conexiones. Una vez hecho esto, el *iPAQ pocket PC* siempre preguntará si se debe permitir la conexión.

Para solicitar autorización para acceder a servicios:

1. En la pantalla Hoy, puntear en el icono de *Bluetooth> Configuración de Bluetooth*
2. Puntear en la ficha correspondiente al servicio que se quiere activar: transferencia de archivos, puerto serie, servidor de red personal o puerta de enlace audio
3. Seleccionar Se necesita autorización
4. Puntear OK

Conexiones Seguras Mediante una Clave de Paso o un Enlace

Para establecer una conexión segura con otro dispositivo, puede utilizar la función de clave de paso o un enlace establecidos. También se puede agregar la función de cifrado de datos a este tipo de seguridad.

Una clave de paso es un código que se introduce para autenticar las conexiones que solicitan otros dispositivos. Esta clave debe ser conocida y utilizada por ambas partes, o no se permitirá la conexión.

Para solicitar una clave de paso o un enlace:

1. En la pantalla Hoy, puntear en el icono de *Bluetooth> Configuración de Bluetooth*
2. Puntear en la ficha correspondiente al servicio que se quiere activar: transferencia de archivos, puerto serie, servidor de red personal o puerta de enlace audio.
3. Seleccionar Se necesita autenticación (contraseña)
4. Seleccionar Se necesita encriptación se desea solicitar que se cifren los datos intercambiados entre los dispositivos
5. Puntear OK

Establecimiento de una Carpeta Compartida

Para seleccionar una carpeta compartida a la que acceden otros dispositivos cuando se conectan al *iPAQ Pocket PC*:

1. En la pantalla *Hoy*, puntear en el icono de *Bluetooth> Configuración de Bluetooth*
2. Puntear en la ficha de Transferencia de archivos
3. Puntear en el icono de carpeta y localizar la carpeta de archivos que se desee

Perfiles

Utilizar los perfiles para activar rápidamente la configuración personal seleccionada en varios entornos.

Para crear un perfil:

1. En la pantalla *Hoy*, puntear en el icono de *Bluetooth> Configuración de Bluetooth> ficha General> icono de perfil*
2. Puntee en el botón Nuevo
3. Especifique un nombre descriptivo
4. Seleccione un perfil existente para utilizarlo como plantilla
5. Puntear OK

Cuando haya creado un nuevo perfil, deberá activarlo. Para activar un perfil después de crearlo:

1. Crear un perfil
2. En la pantalla Hoy, puntear en el icono de Bluetooth> Configuración de Bluetooth
3. En la ficha General, seleccionar el perfil en la lista de la flecha hacia abajo de Perfil actual
4. Puntear OK

Para guardar los valores de configuración de *Bluetooth* del *iPAQ Pocket PC* en cualquier perfil:

1. En la pantalla *Hoy*, puntear en el icono de *Bluetooth> Configuración de Bluetooth*
2. En la ficha General, puntear en el icono de perfil

3. Seleccionar un perfil en la lista Agregar/ Eliminar perfiles o crear un perfil nuevo
4. Puntear OK
5. Configurar el iPAQ Pocket PC especificando los valores de conexión, compartición y seguridad que desee en las fichas General Accesibilidad Transferencia de archivos, Intercambio de información, Puerto serie, Acceso telefónico a redes o Gateway Audio, respectivamente de la configuración de Bluetooth
6. Puntear en OK para cerrar la configuración de Bluetooth. Los cambios se guardan automáticamente

Bluetooth debe estar activo para que se guarden los cambios.Para cambiar el nombre a un perfil:

1. En la pantalla *Hoy*, puntear en el icono de *Bluetooth> Configuración de Bluetooth*
2. En la ficha General, puntear en el icono de perfil
3. Seleccionar un perfil en la lista Agregar/ Eliminar perfiles
4. Puntear en el botón Cambiar
5. Especificar un nombre descriptivo nuevo
6. Puntear en Intro
7. Puntear en OK

Para eliminar un perfil:

1. En la pantalla *Hoy*, puntear en el icono de *Bluetooth> Configuración de Bluetooth*
2. En la ficha General, puntera en el icono de perfil
3. Selecciona un perfil en la lista Agregar/ Eliminar perfiles
4. Puntear en el botón eliminar
5. Puntear en Sí para confirmar que se desea eliminar el perfil
6. Puntear OK

Cómo Trabajar con Bluetooth Manager

Utilizar *Bluetooth Manager* para:

- Establecer conexiones
- Mostrar accesos directos
- Intercambiar tarjetas de presentación
- Controlar los elementos que aparecen en pantalla

Para abrir *Bluetooth Manager:*

- En la pantalla *Hoy*, Puntear en *Inicio> Bluetooth Manager. L*a primera pantalla que aparece es *Mis accesos directos.* Cuando se abre *Bluetooth Manager*, se activa automáticamente *Bluetooth*

Localización y Selección de un Dispositivo

Para localizar un dispositivo y conectarlo a *Bluetooth Manager* son necesarias varias tareas. Si se le solicita que localice un dispositivo, el explorador de *Bluetooth* ayudará a buscar la función deseada que admite otros dispositivos *Bluetooth.*

Cómo Emparejar Dispositivos

Se pueden emparejar dispositivos de modo que deban intercambiar una clave generada por ordenador antes de cada conexión. La clave de seguridad también se denomina 'clave de paso'. Esta clave se crea a partir de una única dirección de dispositivo *Bluetooth*, de un número aleatorio y de una contraseña definida por el usuario

Una vez emparejados dos dispositivos, estos mantienen una relación de confianza que puede comprobarse mediante una clave de vínculo. No es necesario que el usuario introduzca más datos. Por lo tanto, las conexiones y las actividades pueden llevarse a cabo entre los dispositivos emparejados sin que el usuario tenga que autorizarlas constantemente

Para emparejar dispositivos:

1. En la pantalla *Hoy*, puntear en *Inicio> Bluetooth Manager*
2. Puntear Herramientas> Dispositivos emparejados
3. Puntear Agregar
4. Puntear en el icono de búsqueda
5. Puntear en un dispositivo
6. Especificar una contraseña en el campo Contraseña
7. Puntear en OK
8. Especificar la misma clave de paso en el otro dispositivo

Cómo Desemparejar Dispositivos

Es posible eliminar la relación de emparejamiento entre dispositivos

1. En la pantalla *Hoy*, puntear *Inicio >> Bluetooth Manager*
2. Puntear en Herramientas, y , a continuación, en Dispositivos emparejados
3. Puntear en un nombre de dispositivo
4. Puntear en Quitar
5. Puntear en Sí para desemparejar los dispositivos

Conexión a Otros Dispositivos

Utilizar *ActiveSync* y conexiones serie o telefónicas para comunicarse con otros dispositivos Bluetooth. Establezca una asociación con un teléfono móvil y configure los servicios Bluetooth que ofrece.

Especificación del Puerto de Comunicaciones

Puede especificar los puertos COM virtuales utilizados para crear una conexión de puerto serie. Es posible que tenga que especificar estos puertos COM para actividades como la impresión

Cuando otros dispositivos inician una conexión serie, se utiliza el puerto COM de entrada. Cuando se inicia una conexión serie con otro dispositivos, utiliza el puerto COM de salida. Para especificar el puerto de comunicaciones:

1. En la pantalla Hoy, puntear el icono de *Bluetooth> Configuración de Bluetooth*
2. Puntear en la ficha Puerto serie
3. Anotar los nombres de los puertos COM de entrada y de salida
4. Puntear OK

Establecimiento de una Conexión de ActiveSync

Puede configurar una asociación entre *ActiveSync* y un ordenador compatible con *Bluetooth*

Para establecer una conexión de ActiveSync

1. En la pantalla *Hoy*, puntear en I*nicio*-> Bluetooth Manager
2. Puntear en Nuevo, y a continuación, en Conectar
3. Puntear en ActiveSync vía Bluetooth-> Siguiente
4. Seguir las instrucciones del asistente para la conexión

Establecimiento de una Conexión Serie

Utilizar la conexión inalámbrica de puerto serie *Bluetooth* de igual modo que utilizaría una conexión de cable física. Se debe de configurar la aplicación que utilizaría la conexión con el puerto de serie correcto:

1. En la pnatllan Hoy, puntear en Inicio> Bluetooth Manager
2. Puntear en Nuevo, y a continuación, en Conectar
3. Puntear en Explorar dispositivo Bluetooth-> Siguiente
4. Seguir las instrucciones del asistente para la conexión

Redes Telefónicas

Al utilizar una conexión telefónica (DUN), el dispositivo remoto que ofrece el servicio de conexión telefónica y el ordenador remoto al que se conecta deben de disponer de acceso telefónico.

Los dispositivos que pueden ofrecer conexiones telefónicas incluyen los dispositivos *Bluetooth* siguientes:

- Teléfonos móviles
- Ordenadores de escritorio
- Módems

Utilización de Redes Telefónicas

Para conectarse a un dispositivo que ofrece acceso telefónico por módem:

1. En la pantalla *Hoy*, puntear en Inicio> *Bluetooth manager*
2. Puntear en Nuevo> Conectar
3. Puntear en Conectar a Internet> Siguiente
4. Seguir las instrucciones de asistente para la conexión
5. Puntear en Nueva conexión.
6. Puntear en OK
7. Especificar el nombre en el campo de nombre de conexión
8. Puntear en OK para empezar a marcar

Para conectarse a Internet y utilizar *Pocket Internet Explorer*, primero debe conectarse a un teléfono que disponga de la función *Bluetooth* desde *Bluetooth Manager*. Para establecer esta conexión como conexión telefónica predeterminada para *Pocket Internet Explorer:*

1. En la pantalla *Hoy*, puntear en *Inicio-> Configuración-> ficha Conexiones*
2. Puntear en el icono *conexiones -> ficha Avanzada*
3. Puntear en *Seleccionar redes*
4. Habilite la configuración de *Bluetooth* en la lista desplegable

Unión a una Red Personal

Conecte dos o más dispositivos *Bluetooth* para compartir archivos, colaborar o jugar con juegos para varios jugadores. Para establecer una conexión de red personal:

1. En la pantalla *Hoy*, puntear en *Inicio-> Bluetooth Manager*
2. Puntear en Nuevo-> Conectar
3. Puntear en Unirme a una red personal-> Siguiente
4. Seguir las instrucciones del asistente para la conexión

Cómo Trabajar con Archivos

Puede intercambiar información con un dispositivo conectado y utilizar el explorador de archivos *Bluetooth* para realizar lo siguiente

- Explorar los directorios
- Visualizar archivos y carpetas
- Crear carpetas nuevas
- Enviar y recibir archivos desde un dispositivo remoto
- Eliminar y cambiar el nombre de archivos en un dispositivo remoto

Creación de una Conexión de Transferencia de Archivos

1. En la pantalla *Hoy*, puntear *Inicio-> Bluetooth Manager*
2. Puntear en Nuevo-> Conectar
3. Puntear en Examinar archivos remotos-> Siguiente
4. Seguir las instrucciones del asistente para la conexión

Envío de Archivos

1. Puntear y mantener el puntero en un icono de acceso directo de transferencia de archivos y, a continuación, puntear en *Conectar*
2. Puntear en Archivo-> Enviar un archivo...
3. Localizar el archivo que se desea enviar
4. Puntear en el archivo para enviarlo
5. Puntear OK

Creación de una Carpeta en un Dispositivo Remoto

1. Puntear y mantener el puntero en un icono de acceso directo de transferencia de archivos y, a continuación puntear en *Conectar*
2. Desplácese hasta donde desea ubicar la nueva carpeta
3. 3. Puntee en Archivo-> Crear una carpeta
4. 4. Especifique un nombre de carpeta mientras la opción carpeta nueva esté seleccionada y, a continuación, puntear en Intro. Puntear en OK.

Recepción de un Archivo de un Dispositivo Remoto

1. Puntear y mantener el puntero en un icono de acceso directo de transferencia de archivos y, a continuación, puntear en *Conectar*
2. Desplazarse hasta la ubicación del archivo en el dispositivo remoto
3. Puntear el archivo
4. Puntear en Archivo-> Obtener
5. Puntear en OK

Eliminación de un Archivo de un Dispositivo Remoto

1. Puntear y mantener el puntero en un icono de acceso directo de transferencia de archivos y, a continuación, puntear en *Conectar*
2. Desplazarse hasta la ubicación del archivo en el dispositivo remoto
3. Puntear el archivo
4. Puntear en Archivo> Eliminar
5. Puntear en Sí para verificar que desea eliminar el archivo seleccionado
6. Puntear en OK

Utilización del Intercambio de Tarjetas de Presentación

Mediante el intercambio de tarjetas de presentación, puede realizar lo siguiente:

- Configurar su propia tarjeta de presentación
- Enviar una tarjeta de presentación a uno o varios dispositivos
- Solicitar una tarjeta de presentación a uno o varios dispositivos
- Intercambiar tarjetas de vista con uno o varios dispositivos

Debe establecer un nombre de contacto determinado para enviar o intercambiar información de tarjetas de presentación.

En primer lugar, debe especificar la tarjeta de presentación predeterminada en la ficha Intercambio de Información de la configuración de *Bluetooth.* Este nombre de contacto será el predeterminado en las transferencias de tarjetas de presentación

Configuración de la Información de la Tarjeta de Presentación

Dos dispositivos *Bluetooth* pueden intercambiar información de tarjetas de presentación de forma electrónica. Puede especificar que su información personal se envíe a otro dispositivo siempre que se solicite. Esta información proviene de la lista de contactos de *PocketPC Outlook.* La opción predeterminada es “Ninguna”.

Para configurar la información de la tarjeta de presentación:

1. Crear un contacto en la aplicación Contactos que incluya su nombre, cargo y otra información relevante
2. En la pantalla Hoy, puntear en el icono de Configuración de Bluetooth
3. Puntear en la ficha Intercambio de información
4. Puntear en el icono Mi tarjeta de presentación (vCard)
5. Seleccione el contacto en la lista
6. Puntear OK

Envío de Tarjetas de Presentación

1. En la pantalla *Hoy*, puntear *Inicio> Bluetooth Manager*
2. Puntear en el icono de tarjeta de presentación
3. Puntear en el icono de enviar
4. Puntear OK

Solicitud de Tarjetas de Presentación

1. En la pantalla *Hoy*, puntear en *Inicio> Bluetooth Manager*
2. Puntear en el icono de tarjeta de presentación
3. Puntear en el icono de solicitar

4. Puntear en el dispositivo desde el que se desea solicitar una tarjeta de presentación

Intercambio de Tarjetas de Presentación

Se puede intercambiar la información de las tarjetas de presentación con otro dispositivo. Si se encuentra disponible, la información del dispositivo se enviará directamente a la lista *Contactos en Pocket Outlook.*

Para intercambiar tarjetas de presentación:

1. En la pantalla *Hoy*, puntear en *Inicio> Bluetooth Manager*
2. Puntear en el icono de tarjeta de presentación
3. Puntear en el icono de intercambiar
4. Puntear en el dispositivo con el que desea intercambiar la tarjeta de presentación
5. Puntear en OK

Cómo Abrir una Conexión

1. En la pantalla *Hoy*, puntear en *Inicio> Bluetooth Manager*
2. Puntear y mantener el puntero en el nombre de lista o icono y , a continuación, puntear en Conectar
3. Puntear en OK

Visualización del Estado de la Conexión

Puede visualizar los elementos siguientes:

- El nombre de la conexión
- El nombre del dispositivo
- El estado de la conexión
- La longitud de la conexión
- La fuerza de la señal

Para visualizar el estado de la conexión:

1. En la pantalla *Hoy*, puntear en *Inicio> Bluetooth Manager*
2. Puntear y mantener el puntero en un nombre de lista o icono de conexión activo y, a continuación, puntear en Estado.
3. Puntear en OK

Cierre de una Conexión

1. En la pantalla *Hoy*, puntear en *Inicio> Bluetooth Manager*
2. Puntear y mantener el puntero en el nombre de lista o icono de conexión
3. En el menú, puntee en Propiedades
4. Puntear en OK

Visualización de la Información de la Conexión

1. En la pantalla *Hoy*, puntear en *Inicio> Bluetooth Manager*
2. Puntear y mantener el puntero en el nombre de lista o icono de conexión
3. En el menú, puntear en Propiedades
4. Puntear en OK

Cómo Trabajar con Conexiones

Se puede crear accesos directos para abrir y visualizar información de estado de todas las conexiones

Creación de un Acceso Directo

Al crear un acceso directo para uno o varios servicios, no se establece ninguna conexión. Sólo se coloca un acceso directo para dicho servicio en la ficha Mis accesos directos de *Administrador de Bluetooth.*

Para crear un acceso directo:

1. En la pantalla *Hoy* puntear en *Inicio> Bluetooth Manager*
2. Puntear en Nuevo y, a continuación, en Conectar.
3. Seleccione un tipo de servicio y puntee en Siguiente
4. Seguir las instrucciones del asistente para la conexión

Eliminación de un Acceso Directo

1. En la pantalla *Hoy*, puntear en *Inicio> Bluetooth Manager*
2. Puntear y mantener el puntero en el nombre de lista o icono de conexión que desea eliminar
3. En el menú, puntear en Eliminar
4. Puntear en Sí para confirmar que debe eliminarse el acceso directo seleccionado
5. Puntear OK

Visualización de Accesos Directos

Puede ver los accesos directos en forma de iconos o en formato de lista

1. En la plantilla *Hoy*, puntear *Inicio> Bluetooth Manager*
2. Puntear en Ver
3. Puntear en Lista o en Iconos. Puntear en OK

3.4.3 Utilización de la LAN Inalámbrica en el iPAQ Pocket PC 5500

3.4.3.1 Funciones Básicas

El iPAQ Pocket PC puede conectarse a una LAN inalámbrica (WLAN) 802.11b, o puede conectarse directamente a otros dispositivos compatibles con la WLAN. Con la WLAN, se pueden realizar las operaciones siguientes:

- Acceder a Internet
- Enviar y recibir mensajes de correo electrónico
- Acceder a información de red corporativa
- Utilizar redes privadas virtuales (VPN) para un acceso remoto seguro
- Utilizar puntos de acceso para la conexión inalámbrica

3.4.3.2 Introducción LAN inalámbrica

Activación y Desactivación de la WLAN

Para utilizar la WLAN en el iPAQ Pocket PC, se debe de activar y configurar en el dispositivo.

Para activar y desactivar la WLAN:

1. En la pantalla *Hoy*, puntear en *Inicio->iPAQ Wireless*
2. Puntear en el icono de Red Local

Si la WLAN está activada, el color del icono de *Red Local* cambiará del naranja al verde, y el indicador de la LAN inalámbrica parpadeará en verde para indicar que la WLAN está activa y conectada, o en ámbar para indicar que la WLAN está activa pero no está conectada.

Si la WLAN está desactivada, el color del icono de *Red Local* cambiará de verde a naranja.

Conexión Automática a una Red

Si hay una o varias redes, el icono de *indicador de red* aparecerá en la barra de exploración. Puntear en la red a la que se desee conectar y especificar si la red de conecta a Internet o a la red de trabajo.

Si se solicita una clave de red (WEP), introducirla y puntear en *Conectar.*

Introducción Manual de una Nueva Configuración de Red

Una red inalámbrica puede agregarse cuando se detecta la red (el icono de *indicador de red* se muestra en la barra de exploración) o manualmente introduciendo información de configuración. Para agregar manualmente una red inalámbrica:

1. Comprobar que la WLAN está activa.
2. Puntear en el icono Conexiones ->Configuración-> ficha Avanzado-> botón Tarjeta de red-> ficha Inalámbrico-> Agregar nueva configuración
3. Puntear en la ficha General e introducir un nombre red (SSID). SSID o nombre de red es el acrónimo del identificador de conjunto de servicios. Este número de identificación utiliza un máximo de 32 caracteres, y distingue entre mayúsculas y minúsculas. Si se ha detectado una red en el paso 2, el SSID se introducirá manualmente y no podrá cambiarse.
4. En el cuadro Conecta con:, seleccionar el tipo de conexión de la red (Internet o Trabajo).
5. Si se desea una conexión ad-hoc, puntear en la casilla de verificación Ésta es una conexión de equipo a equipo (ad-hoc)
6. Si se requiere información de autenticación, en la pantalla Configurar red inalámbrica, puntear en la ficha Autenticación
7. Seleccionar el tipo de autenticación de red que se desea utilizar:
    a. Para utilizar la función de cifrado de datos, puntear en la casilla de verificación *Cifrado de datos (WEP habilitado)..*
    b. Para utilizar la autenticación de clave compartida, puntear en la casilla de verificación *Autenticación de red (modo compartido).* Escribir la clave de red en el cuadro *Clave de red:.*
    c. Si la red proporciona automáticamente una clave de red, puntear en la casilla de verificación *La clave se proporciona automáticamente.*

d. Para mayor seguridad, puntear en la casilla de verificación *Habilitar acceso de red a través de IEEE*. Sólo se debe seleccionar esta opción si el entorno de red la admite.

Búsqueda de Redes a las que Acceder

Las redes ya configuradas son redes preferidas y aparecen enumeradas en el cuadro *Redes inalámbricas*. Se puede optar por conectarse solo a las redes preferidas, o puede hacer que el *iPAQ Pocket PC* busque y conecte con alguna red que esté disponible (preferida o no).

1. Puntear en el icono *Conexiones-> Configuración->* ficha *Avanzado->* botón *Tarjeta de red->* ficha *Inalámbrica.*
2. En el cuadro Acceso a las redes, puntear el tipo de red a la que desea conectarse (Todas las disponibles, Sólo los puntos de acceso o Sólo de equipo a equipo).
3. Para conectarse solamente a las redes ya configuradas, desactive la casilla de versificación Conectar automáticamente a redes no preferidas.

Si se selecciona la casilla de verificación *Conectar automáticamente a redes no preferidas*, el *iPAQ Pocket PC* detectará las redes nuevas y ofrecerá la posibilidad de configurarlas.

3.4.3.3 Gestión de la Configuración de una Red Inalámbrica

Visualización o Edición de una Red Inalámbrica

Para ver o editar una red inalámbrica existente o disponible:

1. Comprobar que la WLAN esté activa.
2. Puntear en el icono Conexiones-> Configuración-> ficha Avanzado-> botón Tarjeta de red-> ficha Inalámbrico.
3. En el cuadro Redes inalámbricas, puntear en el nombre de red que desee.
4. Modifique los valores de configuración existentes que deseas y, a continuación, puntear en OK para guardas los cambios.

Eliminación de una Red Inalámbrica

Para eliminar una red inalámbrica existente o disponible:

1. Comprobar que la WLAN esté activa.
2. Puntear en el icono Conexiones-> Configuración-> ficha Avanzado-> botón Tarjeta de red-> ficha Inalámbrico.
3. En el cuadro Redes Inalámbricas, puntear y mantener el puntero en la red que desee eliminar.
4. Puntear en Quitar configuración.

Control del Estado y la Fuerza de la Señal

Para ver la fuerza de la señal entre el *iPAQ Pocket PC* y el punto de acceso de la conexión WLAN:

1. Puntear en el icono *Conexiones* en la barra de exploración.
2. Cuando se visualice el cuadro Conectividad, podrá ver el tipo de red a la que está conectado el iPAQ Pocket PC (por ejemplo trabajo o Internet) y un icono indica la fuerza de la señal.

3. Para realizar cambios en la configuración de la conexión, puntear en Configuración
4. Para salir del cuadro Conectividad, puntear en el botón Ocultar.

3.4.3.4 Cómo Trabajar con la Configuración de la Red

Búsqueda de una Dirección IP

Para buscar la dirección IP que la red inalámbrica utiliza:

1. Comprobar que la WLAN esté activa.
2. Conectarsese a la red adecuada.
3. Puntear en el icono Conexiones-> Configuración-> ficha Avanzado> botón Tarjeta de red-> ficha Adaptadores de red.
4. Puntear en la ficha Dirección IP. La dirección IP aparece en el cuadro Dirección IP.

Cambio de la Configuración de TCP/IP

Para cambiar la configuración de TCP/IP:

1. Comprobar que la WLAN está activada.
2. Puntear en el icono Conexiones-> Configuración-> ficha Avanzado-> botón Tarjeta de Red -> ficha Adaptadores de red.
3. En el cuadro desplegable Mi tarjeta de red conecta a..., puntear en Internet o en Trabajo.
4. En el cuadro Puntee en un adaptador para modificar la configuración, puntear en Adaptador de WALN para iPAQ.
5. Puntear en la ficha Dirección IP.
6. Puntear en Usar dirección IP específica e introducir la información solicitada.
7. Puntear en OK para guardar la configuración.

Cambio de la Configuración DNS y WINS

Los servidores que requieren una dirección IP asignada también pueden necesitar un método para asignar nombres de ordenador a direcciones IP. *El iPAQ Pocket PC* admite dos opciones de resolución de nombre:

- DNS
- WINS

Para cambiar la configuración del servidor:

1. Comprobar que la WLAN está conectada
2. Puntear en el icono Conexiones> Configuración> ficha Avanzado> botón Tarjeta de red> ficha Adaptadores de red
3. En el cuadro desplegable Mi Tarjeta de red conecta a.., puntear en Internet o en Trabajo
4. En el cuadro Puntee en un adaptador para modificar la configuración , puntee en Adaptador de WLAN para iPAQ
5. Puntear en la ficha Servidores de nombres e introduzca la información solicitada
6. Puntear en OK para guardar la configuración

Configuración de las Conexiones de Servidor VPN

Una conexión VPN permite conectarse de manera segura a servidores, como por ejemplo una red corporativa, mediante Internet. Para configurar una conexión del servidor VPN:

1. Comprobar que la WLAN esté activa
2. En la pantalla Hoy, puntear en Inicio-> Configuración -> ficha Conexiones- > Conexiones -> ficha Tareas
3. En Red de trabajo, puntear en Agregar nueva conexión de servidor VPN
4. Seguir las instrucciones del asistente Realizar conexión nueva

Para obtener ayuda en línea sobre cualquier pantalla del *asistente Realizar conexión nueva* o al modificar la configuración, puntear en *?.*

Cambio de las Conexiones de Servidor VPN

1. Comprobar que la WLAN esté activa.
2. En la pantalla Hoy, puntear en Inicio->Configuración-> ficha Conexiones -> Conexiones -> ficha Tareas
3. En Red de trabajo, puntear en Administrar conexiones existentes-> ficha VPN
4. Puntear en la conexión VPN que se desee cambiar y, a continuación, puntear en Edición
5. En el cuadro Nombre, introducir un nombre para la conexión.
6. En el cuadro Nombre del host o dirección IP, introducir el nombre del servidor VPN o la dirección IP.
7. Junto al tipo de VPN, puntear en el tipo e autenticación que se va a utilizar con el dispositivo (IPsec/L2TP o PPTP). Puntear en Siguiente
8. Si se ha seleccionado IPsec/L2TP en la pantalla anterior, puntear en el tipo de autenticación. Si se selecciona Una clave previamente compartida, introducir la clave y, a continuación, puntear en Siguiente. Si se ha seleccionado PPTP en la pantalla anterior, se puede omitir este paso
9. Introducir el nombre de usuario, la contraseña y el nombre del dominio.
10. Para cambiar la configuración avanzada, puntear en el botón Avanzada. Solo se deberá cambiar la configuración avanzada si:
11. Se debe de introducir la configuración TCP/IP debido a que el servidor al que nos conectamos no utiliza direcciones IP asignadas dinámicamente.
12. Se debe cambiar la configuración de DNS o WINS del servidor.
13. Puntear en el botón Finalizar.

Inicio de las Conexiones de Servidor VPN

Para iniciar una conexión mediante un servidor VPN, asegurarse de que la WLAN está activa y, a continuación, seleccionar la red VPN. El *iPAQ Pocket PC* iniciará la conexión automáticamente.

Establecimiento de la Configuración del Servidor PROXY

Si se está conectado a un ISP o red privada durante la sincronización, el *iPAQ Pocket PC* deberá descargar del ordenador personal los valores de configuración de *proxy* adecuados. Si dichos valores no se encuentran en el ordenador personal, o si es necesario cambiarlos, se deberán configurar manualmente. Para establecer los valores de configuración del servidor *proxy:*

1. Ponerse en contacto con el ISP o administrador de red correspondiente para obtener el nombre del servidor *proxy*, el tipo de servidor, el puerto, el tipo de protocolo *Sock* utilizado, y el nombre de usuario y contraseña.
2. Comprobar que la WLAN esté activa
3. En la pantalla Hoy, puntear en Inicio-> Configuración-> ficha Conexiones -> Conexiones-> ficha Tareas
4. En Red de trabajo, puntear en Editar mi servidor proxy-> ficha Configuración de proxy.
5. Puntear en las casillas de verificación Esta red conecta a Internet y Esta red utiliza un servidor proxy para conectar a Internet
6. En el cuadro Servidor proxy, introduzca el nombre del servidor proxy.
7. Si se necesita cambiar el número de puerto o la configuración del tipo de servidor proxy, puntear en el botón Avanzado y cambiar los valores de configuración que se desee. Puntear en OK.

3.5 Ordenador de Sobremesa

3.5.1 Introducción

Como es lógico, antes de insertar la aplicación desarrollada, y los componentes utilizados en la misma dentro del sistema real, todos los aspectos funcionales del sistema deben ser chequeados y simulados. Además, como se ha visto en los apartados anteriores, el uso de un ordenador de sobremesa se hace indispensable tanto a la hora de la puesta en marcha de los componentes hardware empleados, como en la fase de desarrollo del software. Las características principales del ordenador que se ha empleado en el desarrollo del proyecto son las siguientes:

- Buena capacidad de memoria y procesador.
    - 512 Mb RAM
    - Procesador a 2,6 GHz
- Alta capacidad de conectividad de red y adhesión de otros dispositivos, permitiendo:
    - Soporte para conexión con el *iPAQ Pocket PC*, a través de puerto USB, infrarrojos o puerto serie.
    - Soporte para instalación de tarjeta WiFi a través de puerto USB.
- Sistema operativo Windows XP

Las principales tareas que han sido adjudicadas al ordenador han sido las siguientes:

1. Soporte de instalación para ActiveSync.
2. Plataforma de programación.
3. Comunicación *wireless.*
4. Simulación de los sensores ópticos.

3.5.2 Soporte de Instalación Para ActiveSync

La instalación de *ActiveSync* en el ordenador personal debe de realizarse previamente a la conexión del dispositivo PDA al mismo. Una vez instalado el software en el ordenador de escritorio, cada vez que se conecte el *iPAQ Pocket PC* al mismo, mediante infrarrojos, puerto USB o puerto serie, el programa *ActiveSync* detectará dicha conexión. Es entonces cuando se pregunta al usuario que tipo de conexión quiere establecer entre el iPAQ y el PC. Para la transferencia de archivos bidireccional y la instalación de aplicaciones sobre el *Pocket PC*, la configuración adecuada es configuración como invitado. Usando esta configuración de conexionado podemos disponer de todos los servicios que necesitamos para el proyecto – configuración del iPAQ, instalación de programas y traspaso de archivos entre el PC y el iPAQ- ahorrándonos el tiempo de sincronización que emplea el otro tipo de conexión.

ActiveSync proporciona una interfaz de usuario gráfica a través de la cual se pueden hacer las tareas necesarias de una manera intuitiva. Así por ejemplo, para copiar archivos de un directorio a otro dentro de la jerarquía del *iPAQ Pocket PC* se hace del mismo modo que se haría usando el explorador de *Windows.*

Un icono presente en ambos lados, PC y *Pocket PC* indica en todo momento la activación o desconexión de la comunicación entre ambos.

Resumiendo, a través del software de *ActiveSync* se permite la instalación de aplicaciones sobre el *iPAQ Pocket PC* a través del ordenador personal y la transferencia de archivos bidireccional.

3.5.3 Plataforma de Programación

El ordenador personal utilizado también actúa como plataforma de programación de las aplicaciones que luego correrán sobre el *Pocket PC*. Para ello se instalarán en el ordenador las implementaciones adecuadas de la plataforma Java. Mediante el ordenador, se escribirán las aplicaciones y se compilarán. Además, puede llegarse a emular el comportamiento de dichas aplicaciones antes de su transferencia al *Pocket PC.* En el capítulo dedicado al desarrollo del trabajo se explicará con detalle cuales son los estándares de Java que deben de ser instalados en el ordenador, así como el uso de todas las herramientas utilizadas para la implementación de la aplicación Java.

El lenguaje de programación seleccionado para el desarrollo de la aplicación ha sido Java. Como se expuso anteriormente la máquina virtual Java instalada en la PDA esta basada en una implementación de PersonalJava 1.2. Esta especificación de la plataforma Java es compatible con el JDK 1.2. Los programas pueden ser escritos con cualquier aplicación construida con ese propósito, desde aplicaciones muy sencillas como RealJ o incluso un editor de texto, pasando por otras más completas como lo es Kawa. Para la compilación de las aplicaciones se ha usado el compilador de JDK 1.2. También pueden usarse compiladores de versiones superiores con este fin, teniendo en estos

casos especial cuidado de no utilizar APIs que no están definidas en la especificación de PersonalJava 1.2. Mediante el uso de la herramienta JavaCheck 3.0 puede comprobarse que efectivamente los programas escritos se ciñen a la especificación de PersonalJava 1.2. Puede instalarse también el PersonalJava Emulation Environment 3.1 con el fin de simular las aplicaciones antes de ser ejecutadas sobre el Pocket PC.

3.5.4 Comunicación Wireless

Una de las funcionalidades que debe de aportar la aplicación que corra sobre la PDA es la de comunicarse de modo inalámbrico con otros subsistemas del proyecto EFTCoR. Esta comunicación puede hacerse mediante WLAN 802.11b o usando *Bluetooth*. El ordenador servirá para explorar la funcionalidad *wireless* mediante el lenguaje de programación que soportará el *iPAQ Pocket* PC. La instalación de una tarjeta wíreless en el PC tan solo requiere conectar la misma a alguno de los puertos USB del ordenador de escritorio. Dos aspectos son tratados haciendo uso de esta tecnología:

- Recepción de comandos enviados desde el *Pocket PC*
- Envío de valores al *Pocket PC*

Para ambos casos se han construidos pequeñas aplicaciones gráficas que monitoricen el intercambio de información mediante *wireless.*

3.5.5 Simulación de Sensores Ópticos

Se ha construido una aplicación en el lado del PC que simule de manera aleatoria el cambio de valores de los sensores ópticos, y su posterior envío al *iPAQ Pocket PC*. El *iPAQ Pocket PC* representará gráficamente la llegada de esos datos en tiempo real. En la práctica, la mesa que soporte el cabezal de chorreado debe disponer de una serie de sensores que capten la distancia exacta existente entre la superficie del buque y los mismos. Esta información debe de ser leida por la PDA y de ese modo informar al operario humano de los valores en cada momento. Para simular este comportamiento, se implementa una aplicación en el ordenador personal. Esta aplicación se comunica mediante sockets con la PDA. Cada cierto tiempo la PDA lee tales valores, y estos le son transferidos inmediatamente desde el PC.

Capítulo 4
Plataforma de Programación

4.1 Introducción

La plataforma de desarrollo software escogida para la implementación de la aplicación ha sido Java.

Java es un lenguaje de programación de alto nivel con el que se pueden implementar todo tipo de aplicaciones. Una de las ventajas de Java sobre otros lenguajes de programación es su independencia de la plataforma, tanto en código fuente como en binario. Esto quiere decir que el código producido por el compilador Java puede transportarse a cualquier plataforma (ENTEL, Sparc, Motorota, etc) que tenga instalada una máquina virtual Java y ejecutarse. Pensando en Internet esta característica es crucial ya que esta red conecta ordenadores muy distintos. En cambio, C++, por ejemplo, es independiente de la plataforma sólo en código fuente, lo cual significa que cada plataforma diferente debe proporcionar el compilador adecuado para obtener el código máquina que tiene que ejecutarse.

Según lo expuesto, Java incluye dos elementos: un *compilador* y un *intérprete.* El compilador produce un código de bytes que se almacena en un fichero para ser ejecutado por el intérprete Java denominado *máquina virtual* de Java.

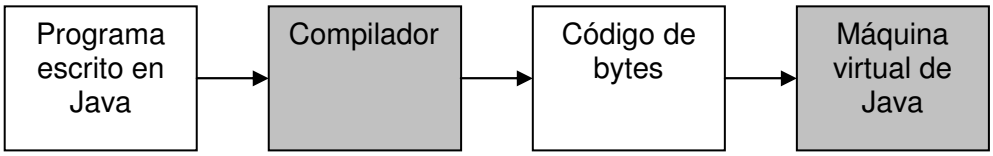


Figura 1. Fases del código Java

Los códigos de bytes de Java son un conjunto de instrucciones correspondientes a un lenguaje máquina que no es específico de ningún procesador, sino de la máquina virtual de Java. Hoy día casi todas las compañías de sistemas operativos y de navegadores han implementado máquinas virtuales según las especificaciones publicadas por *Sun Microsystems,* propietario de Java, para que sean compatibles con el lenguaje Java. Para las aplicaciones de Internet (denominadas *applets*) la máquina virtual está incluida en el navegador y para las aplicaciones Java convencionales, puede venir con el sistema operativo, con el paquete Java, o bien puede obtenerla a través de Internet.

La interpretación, si bien es cierto que proporciona independencia de la máquina, conlleva también un problema grave, y es la pérdida de velocidad en la ejecución del programa. Por esta razón, la solución fue diseñar un compilador que produjera un lenguaje que pudiera ser interpretado a velocidades, si no iguales, sí cercanas a la de los programas nativos (programas en código máquina propio de cada ordenador), logro conseguido mediante la máquina virtual de Java.

Con todo, las aplicaciones todavía adolecen de una falta de rendimiento apreciable. Éste es uno de los problemas que siempre se ha achacado a Java. Afortunadamente, la

diferencia de rendimiento con respecto a aplicaciones equivalentes escritas en código máquina nativo ha ido disminuyendo hasta márgenes muy reducidos gracias a la utilización de compiladores *JIT (Just In Time* – compilación al instante).

Un compilador JIT interacciona con la máquina virtual para convertir el código de bytes en código máquina nativo. Como consecuencia, se mejora la velocidad durante la ejecución. Sun sigue trabajando sobre este objetivo y prueba de ello son los resultados que se están obteniendo con el nuevo y potente motor de ejecución *HotSpot (HotSpot Perfomance Engine)* que ha diseñado, o por los microprocesadores específicos para la interpretación hardware de código de bytes.

4.1.1 Historia de Java

El lenguaje de programación Java fue desarrollado por *Sun Microsystem* en 1991. Nace como parte de un proyecto de investigación para desarrollar software para comunicación entre aparatos electrónicos de consumo como vídeos, televisores, equipos de música, etc. Durante la fase de investigación surgió un problema que dificultaba enormemente el proyecto iniciado: cada aparato tenía un microprocesador diferente y muy poco espacio de memoria; esto provocó un cambio en el rumbo de la investigación que desembocó en la idea de escribir un nuevo lenguaje de programación independiente del dispositivo que fue bautizado inicialmente como *Oak.*

La explosión de Internet en 1994, gracias al navegador gráfico *Mosaic* para la *Word Wide Web (WWW),* no pasó desapercibida para el grupo investigador de Sun. Se dieron cuenta de que los logros alcanzados en su proyecto de investigación eran perfectamente aplicables a Internet. Comparativamente, Internet era como un gran conjunto de aparatos electrónicos de consumo, cada uno con un procesador diferente. Y es cierto, básicamente, Internet es una gran red mundial que conecta múltiples ordenadores con diferentes sistemas operativos y diferentes arquitecturas de microprocesadores, pero todos tienen en común un navegador que utilizan para comunicarse entre sí. Esta idea hizo que el grupo investigador abandonara el proyecto de desarrollar un lenguaje que permitiera la comunicación entre aparatos electrónicos de consumo y dirigiera sus investigaciones hacia el desarrollo de un lenguaje que permitiera crear aplicaciones que se ejecutaran en cualquier ordenador de Internet con el único soporte de un navegador.

A partir de aquí se empezó a hablar de Java y de sus aplicaciones, conocidas como *applets.* Un a*pplet* es un programa escrito en Java que se ejecuta en el contexto de una página Web en cualquier ordenador, independientemente de su sistema operativo y de la arquitectura de su procesador. Para ejecutar un *applet* sólo se necesita un navegador que soporte la máquina virtual de Java como, por ejemplo, *Microsoft Internet Explorer* o N*etscape*. Utilizando un navegador de éstos, se puede descargar la página Web que contiene el *applet* y ejecutarlo. Precisamente en este campo, es donde Java como lenguaje de programación no tiene competidores. No obstante, con Java se puede programar cualquier cosa, razón por la que también puede ser considerado como un lenguaje de propósito general; pero desde este punto de vista, hoy por hoy, java tiene muchos competidores que le sobrepasan en claridad; por ejemplo Ada o C++.

Una de las ventajas más significativas de Java es su independencia de la plataforma. En el caso en que sea necesario desarrollar aplicaciones que tengan que ejecutarse en sistemas diferentes esta característica es fundamental.

Otra característica importante de Java es que es un lenguaje de programación orientado a objetos (POO). Además de ser transportable y orientado a objetos, Java es un lenguaje fácil de aprender. Tiene un tamaño pequeño que favorece el desarrollo y reduce las posibilidades de cometer errores; a la vez es potente y flexible.

Java está fundamentado en C++. Esto quiere decir que mucha de la sintaxis y diseño orientado a objetos se tomó de este lenguaje. De todas formas existen notables diferencias entre ambos lenguajes; en Java no existen punteros ni aritmética de punteros, las cadenas de caracteres son objetos y la administración de memoria es automática, lo que elimina la problemática que presenta C++ con las lagunas de memoria al olvidar liberar bloques de la misma que fueron asignados dinámicamente.

4.1.2 Características Beneficiosas de Java

Java tiene mucho que ofrecer. Más allá de una plataforma independiente, tiene un número de características que lo hacen un atractivo lenguaje de programación. Estas características hacen de Java un buen candidato para el desarrollo de las aplicaciones software, independientemente del tamaño o forma de la plataforma receptora.

Seguridad

La seguridad fue un asunto tratado desde el principio por los diseñadores y desarrolladores de Java. Incluso en el desarrollo de un lenguaje de programación inicialmente enfocado a dispositivos electrónicos de consumidor, los desarrolladores sabían que estos dispositivos iban a trabajar en red. El trabajo en red supone una vulnerabilidad potencial ante ataques de otros sistemas dentro de la red de trabajo. Como Java ha llegado a ser el lenguaje de programación *Web*, y a causa de que la seguridad fue edificada en Java desde un principio, está bien adaptado para tomar los resultados de seguridad asociados a Internet.

Confiable

La confianza que aporta Java procede de no tener que manejar algunos de los recursos de los niveles bajos tales como memoria y punteros. Los desarrolladores están disponibles para trabajar con la información como un objeto, o una serie de objetos, de un modo más semejante a como se afrontarían los problemas en el mundo real. Los recursos de niveles bajos como los punteros tienden a ser con frecuencia olvidados, esto puede crear problemas en el acceso a memoria de las aplicaciones software. Java establece la carga del manejo de referencias a memoria en sistema recolector de basura donde estas referencias a memoria son mantenidas de manera más metódica, tal y como se haría en C++ donde la carga es soportada por el programador.

Orientado a Objeto

Java es considerado un verdadero lenguaje orientado a objeto. ¿Por qué es importante ser un lenguaje orientado a objeto? Para empezar, las aplicaciones basadas en objetos están consideradas más fáciles de construir y mantener. Esto es debido a que las estructuras de programación orientada a objeto, anima activamente a los programadores a organizar las aplicaciones en piezas manejables y de fácil entendimiento. Grandes y complicadas aplicaciones son entonces edificadas uniendo muchas pequeñas piezas. Estas piezas, o más propiamente dicho objetos, contienen comportamiento y estado. Manteniendo el comportamiento y el estado juntos en un objeto ayuda a asegurar que solo el código responsable del estado puede cambiarlo.

Finalmente, objetos, como representaciones de cosas en el mundo real, permiten que más gente tome parte y entienda el propio modo de trabajo de la aplicación. Una persona de negocios no debe de porque saber lo que un método o procedimiento hace, pero si puede hacerse una idea bastante buena de lo que hace el objeto *Guardar_Cuentas.*

Libre

Como es lógico, a los programadores y los *managers* de estos les gusta que el software sea de libre distribución. Los entornos Java básicos, *runtime* y de desarrollo, son libres. Toda una serie de referencias a la implementación y herramientas de desarrollo pueden encontrarse en la página *web* de Sun.

Esto no quiere decir que todos los entornos de desarrollo y ejecución de Java son libres. Sun mantiene el control sobre Java. La empresa ofrece las implementaciones básicas de referencia, herramientas de desarrollo, clases API, y entornos *runtime* gratis. De cualquier modo, si se requieren características no cubiertas por los entornos básicos, hay que ir a buscarlas fuera de los acuerdos de licencias libres. Además, entornos de desarrollo integrado (IDE), sistemas servidores, herramientas de test, y otras herramientas y productos que un equipo de desarrollo pueden necesitar para ser más productivos y producir productos de más calidad, no son normalmente gratis. Lo importante, sin embargo, es que aprender el lenguaje y comenzar a desarrollar aplicaciones no cuesta nada.

Simple

Java es simple. Esta afirmación es cierta si se está familiarizado con la mayoría de los otros lenguajes de programación orientados a objetos. La sintaxis de Java es fácil de aprender. La sintaxis de Java esta cercana a la da C++ e incluye muchas de las características ventajosas de otros lenguajes de programación. El comentario de muchos programadores expertos es que Java es C++ bien hecho. Por ejemplo, Java es tipado estáticamente. Esto permite al compilador atrapar muchos errores de código. También, la memoria y el recolector de basura están manejadas por la máquina virtual. Estas características por supuesto suponen un precio, que incluye el coste de correr la aplicación dentro de la máquina virtual.

Lo que no es simple en lo referente a Java, son los matices del desarrollo orientado a objetos, la multitud de clases pre-edificadas de Java que están disponibles con incluso los más básicos entornos de Java, entender los aspectos y características de Java que lo hacen funcionar bien o pobremente, y otros muchos aspectos de Java.

Otras Características Útiles

Hay otras muchas características que hace de Java un lenguaje ideal de programación. Algunas de estas razones pueden o no ser aplicables al desarrollo de una aplicación en concreto. *Multi-threading*, manejo de excepciones/errores, mapeo dinámico, son solo algunas de las razones adicionales por las que Java es un buen entorno de programación.

4.1.3 La Trilogía de la Edición Java 2

Como todos los entornos de desarrollo y lenguajes de programación, Java ha evolucionado. Han sido añadidas muchas características y capacidades a Java desde su inicio. Ha sido mejorado también en términos de eficiencia y seguridad. De este modo, el término '2' de Java 2 se refiere a esta actual versión mejorada. *Sun Microsystems*, el creador y organizador de Java, ha agrupado Java2 en tres ediciones:

- *Standard Edition* (J2SE)
- *Enterprise Edition* (J2EE)

- *Micro Edition* (J2ME)

Cada edición alberga las necesidades Java de una serie particular de aplicaciones. J2ME es la tercera y última de las tres ediciones de la versión Java 2.

Uno de los puntos principales de las plataformas Java es el hecho de proporcionar fuera del núcleo de la especificación un rico *set* de herramientas, clases, y APIs que proveen comúnmente, componentes de aplicación funcionales. En teoría, estas clases y APIs son el marco de trabajo básico que permite a las aplicaciones ser desarrolladas más rápido. Hay más de 5000 clases en el estándar de *Java Development Kit (JDK*). Conectividad a bases de datos, varios mecanismos de entrada/salida, manejo de excepciones y errores, y clases de interfaz de usuario son solo algunas de las funcionalidades básicas que viene con los entornos de desarrollo de Java.

Como el entorno de desarrollo de Java se ha expandido y crecido para cubrir las necesidades de numerosas aplicaciones, el número de paquetes y clases disponibles también han crecido. De manera notable, Java se ha expandido para incluir redes de trabajo, interoperabilidad, y distribución de componentes y procesos.

Debido a que las necesidades de la comunidad Java se han ensanchado, y el número de APIs ha crecido, Sun estableció tres ediciones de la plataforma Java para albergar de una manera más adecuada las necesidades de cada comunidad general de desarrolladores de aplicaciones. Esta división en tres ediciones tuvo lugar cuando Sun lanzó Java 2. Las ediciones realmente no proveen funcionalidad adicional a Java. En cambio, han redistribuido los paquetes de la tecnología Java en agrupaciones lógicas basadas en usos de desarrollo típicos.

Otra razón para la división en ediciones llevada a cabo, es que Sun ha usado este mecanismo para generar beneficios. El entorno de desarrollo JDK (ahora SDK) y *Java Runtime Environment (JRE)* han sido siempre productos libres. Las otras ediciones de Java, situadas bajo diferentes acuerdos de licencia, permiten a Sun recuperar algunos de los beneficios perdidos en la base, no costes de producto.

Sea cual sea el motivo real para la separación, los programadores deben tener conocimiento de las tres ediciones de la plataforma Java y como pueden ser aplicadas para los distintos problemas de desarrollo. Fundamentalmente, las tres ediciones son todavía bastante similares. La sintaxis del lenguaje y la base arquitectónica de cada edición son generalmente la misma. De cualquier modo, como sus nombres indican, cada edición oferta ahora únicas características aplicables para el tamaño de los dispositivos para los cuales la edición fue construida.

4.2 J2EE

Java 2 Enterprise Edition (J2EE) es un gran grupo de *APIs* Java y otras tecnologías no pertenecientes a Java. Es generalmente usado para crear aplicaciones potencialmente distribuidas. La tecnología J2EE puede servir como el pegamento y soporte para hacer trabajar juntas a grandes aplicaciones heterogéneas y multi-nivel. J2EE es descrito usualmente como un *middleware* o tecnología del lado del servidor, pero esta denominación es un poco limitada. De hecho, J2EE incluye tecnologías que pueden ser usadas en cualquier capa del sistema de información. Se puede tomar como ejemplo JDBC. Esta tecnología es usada para el acceso de datos desde un cliente Java *applet*, un *serverlet* de Java, o un *Enterprise JavaBean*. Resaltar que J2EE incluye algunas tecnologías que no son controladas por Sun y no están necesariamente conectadas directamente a Java, tales como XML y *CORBA.*

La Plataforma Para Soluciones Empresariales

La plataforma *Java 2 Enterprise Edition (J2EE)* define el estándar para desarrollar aplicaciones a nivel de empresa. La plataforma J2EE simplifica las aplicaciones empresariales basándolas en estándares, componentes modulares, ofreciendo un completo *set* de servicios para estos componentes, y manejando muchos detalles del comportamiento de manera automática, sin una programación complicada.

La plataforma J2EE toma las ventajas de muchas características de la plataforma Java 2, Standard Edition (J2SE), tales como la portabilidad "*Write Once, Run Anywhere"*, JDBC API para acceso a bases de datos, CORBA tecnología para interacción con recursos existentes de la empresa, y un modelo de seguridad que proteja los datos incluso dentro de aplicaciones de Internet. Partiendo de esta base, *Java 2 Platform Enterprise Edition* añade soporte completo para componentes *Enterprise JavaBeans*, *Java Servlets API*, *JavaServer Pages* y tecnología XML. El estándar J2EE incluye completas especificaciones y tests de conformidad para asegurar la portabilidad de las aplicaciones a través de la amplia gama de los sistemas existentes en la empresa capaces de soportar la plataforma de J2EE. Además, la especificación de J2EE asegura interoperabilidad de los servicios Web a través del soporte para *WS-I Basic Profile.*

Haciendo el M*iddleWare* más Fácil

Las empresas de hoy día aumentan sus ventajas competitivas mediante un rápido desarrollo y despliegue de aplicaciones que provean un servicio de negocio único. Si estos desarrollos son aplicaciones internas para la productividad de los empleados, o aplicaciones de Internet para consumidores específicos o vendedores de servicios, un desarrollo rápido es la clave para el éxito.

Portabilidad y escalabilidad son también importantes para la viabilidad a largo plazo. Las aplicaciones empresariales deben escalarse desde pequeños prototipos de trabajo y casos de chequeo hasta alcanzar los casos en los que cientos e incluso miles de usuarios acceden al servicio simultáneamente.

En cualquier caso, este tipo de aplicaciones son duras para la arquitectura. Requieren reunir una variedad de habilidades y recursos, datos y código heredados. En los entornos heterogéneos de hoy día, las aplicaciones empresariales tienen que integrar servicios de una variedad de distribuidores con una serie de diferentes modelos de aplicación y otros estándares. La experiencia de esta industria muestra que integrando estos recursos se puede ahorrar el 50% del tiempo de desarrollo de una aplicación.

Como un único estándar que puede situarse sobre un amplio rango de sistemas empresariales existentes – sistemas de manejo de bases de datos, monitores de transiciones, servicios de directorios y nombrado, y más – la plataforma J2EE rompe las barreras establecidas entre los sistemas existentes de las empresas. El estándar unificado J2EE cubre y abraza recursos requeridos por las aplicaciones con un unificado modelo de aplicación. Esto da paso a la próxima generación de componentes, herramientas, sistemas, y aplicaciones solventando los requisitos estratégicos de las empresas.

La especificación J2EE también soporta emergentes tecnologías *Web Services* a través de la inclusión del *WS-I Basic Profile*. La conformidad respecto a *WS-I Basic Profile* significa que los desarrolladores pueden construir aplicaciones en la plataforma J2EE como servicios Web que puedan interoperar con servicios Web procedentes de entornos distintos al de J2EE.

Con simplicidad, portabilidad, escalabilidad, e integración, la plataforma J2EE se convierte en la más adecuada para las soluciones de empresa.

Un Estándar con Ímpetu de Empresa

Mientras que Sun Microsystems inventó el lenguaje de programación Java y lo inició para su uso en servicios para empresas, el estándar J2EE representa una colaboración entre líderes del campo de empresas de software. Entre los participantes se encuentran distribuidores de Sistemas Operativos y manejadores de bases de datos, vendedores de *middleware* y herramientas, aplicaciones de mercado y desarrolladores de componentes. Trabajando con estos participantes, Sun ha definido una plataforma robusta y flexible que puede ser implementada en una amplia variedad de sistemas existentes en las empresas actualmente disponibles, y que soportan cierto rango de aplicaciones.

El éxito de la plataforma J2EE en la empresa continua creciendo hasta el punto en que dos tercios de las empresas de desarrollo software usan la plataforma J2EE para desarrollar y desplegar sus aplicaciones. Con la adición de servicios Web interoperables y otras nuevas características en la versión 1.4, la plataforma J2EE continuará por mucho tiempo siendo el estándar industrial para soluciones de empresas.

Modelo de Aplicación de Empresa

La *Enterprise Java BluePrints* para la plataforma J2EE describe el modelo de aplicación J2EE y las mejores prácticas para usar dicha plataforma. Construyendo sobre la plataforma J2SE, el modelo de aplicación J2EE provee un simplificado acercamiento al desarrollo altamente escalable y gran disponibilidad en aplicaciones basadas en internet o intranet.

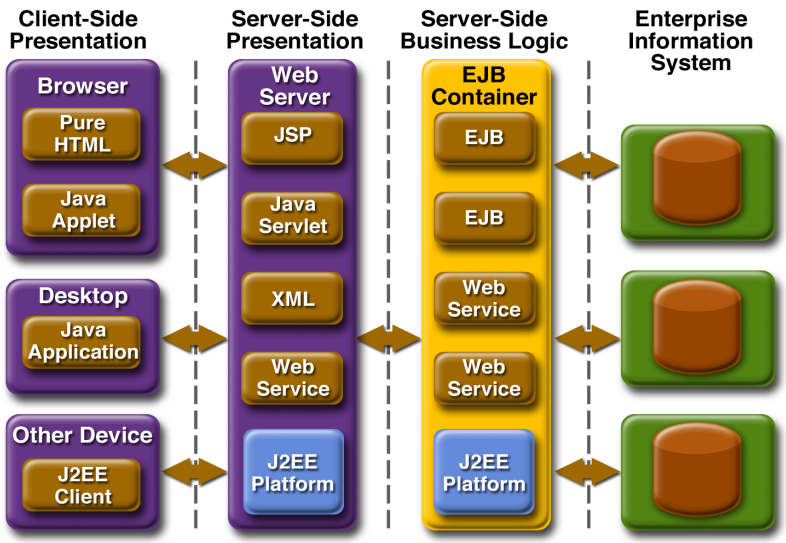


Figura 2. Arquitectura de J2 Enterprise Edition

Gracias al modelo de aplicación de J2EE, quizás lo más interesante acerca de las utilidades de J2EE es lo que estas no hacen. Esto es, varias complejidades inherentes en las aplicaciones de empresas -- gerencia de la transacción, gerencia del ciclo vital, reunión de recursos – son construidas dentro de la plataforma y automáticamente provistas a los componentes soportados. Desarrolladores de componentes y aplicaciones son libres de centrarse en específicos tales como lógica de negocios o interfaces de usuario.

Otra ventaja de la plataforma J2EE es que el modelo de aplicación encapsula capas de funcionalidad y especifica tipos de componentes. La lógica de negocio es encapsulada en componentes *Enterprise JavaBeans (EJB).* La interacción con el cliente puede ser presentada como simples páginas web HTML, páginas web más potentes con applets, Java Servlets, tecnología JavaServer Pages, o aplicaciones Java independientes. Los componentes se comunican de manera transparente usando varios estándares: HTML, XML, HTTP, SSL, RMI, IIOP, y otros.

Los componentes reusables J2EE son elecciones competitivas para empresas desarrolladoras y organizaciones IT. La plataforma J2EE permite ensamblar aplicaciones procedentes de una combinación de estándares, componentes comercialmente disponibles y componentes propios de usuario. Por lo general para componentes de aplicaciones de negocios para soluciones verticales del mercado, una serie de funcionalidades J2EE estandarizadas son disponibles. Esto significa que un sitio de comercio electrónico puede ser construido usando una combinación de componentes EJB para el comportamiento de carros de compra, modificando componentes EJB para especializar servicios de cliente, y capas modificadas completamente usando tecnología *JavaServer Pages* que proporcionan un único aspecto y sentido al sitio Web.

Este acercamiento representa un tiempo de desarrollo más rápido, mejor calidad, mantenimiento y portabilidad, e interoperatividad de servicios Web a través de un rango de plataformas empresariales. Los beneficios de fondo son un incremento de la productividad del programador, mejor uso estratégico de los recursos computacionales, y una mejor recompensa de las inversiones en investigación por parte de la empresa.

4.3 J2SE

Java 2 Standard Edition (J2SE) es el entorno básico Java. Esta implementación provee el núcleo de clases y APIs que permite al desarrollo y ejecución de aplicaciones estándar de clientes y servidores, incluyendo aplicaciones que corren en navegadores web.

Hay dos principales productos en la familia de la plataforma J2SE: *Java 2 Runtime Environment, Standard Edition (JRE)* y *Java 2 Software Development Kit, Standard Edition (SDK)*. JRE provee las APIs Java, máquina virtual Java, y otros componentes necesarios para correr *applets* y aplicaciones escritas en el lenguaje de programación Java. Es también la base para las tecnologías en la plataforma *Java 2 Enterprise Edition (J2EE)* para el desarrollo y despliegue de software para empresas. JRE no contienen herramientas o utilidades tales como compiladores o depuradores para el desarrollo de *applets* o aplicaciones.

Java 2 SDK es el conjunto superior a JRE, contiene todos los componentes de JRE, además de herramientas tales como compiladores y depuradores necesarios para el desarrollo de *applets* y aplicaciones. Este diagrama conceptual ilustra todas las tecnologías componentes de la plataforma J2SE y como se ajustan juntas.

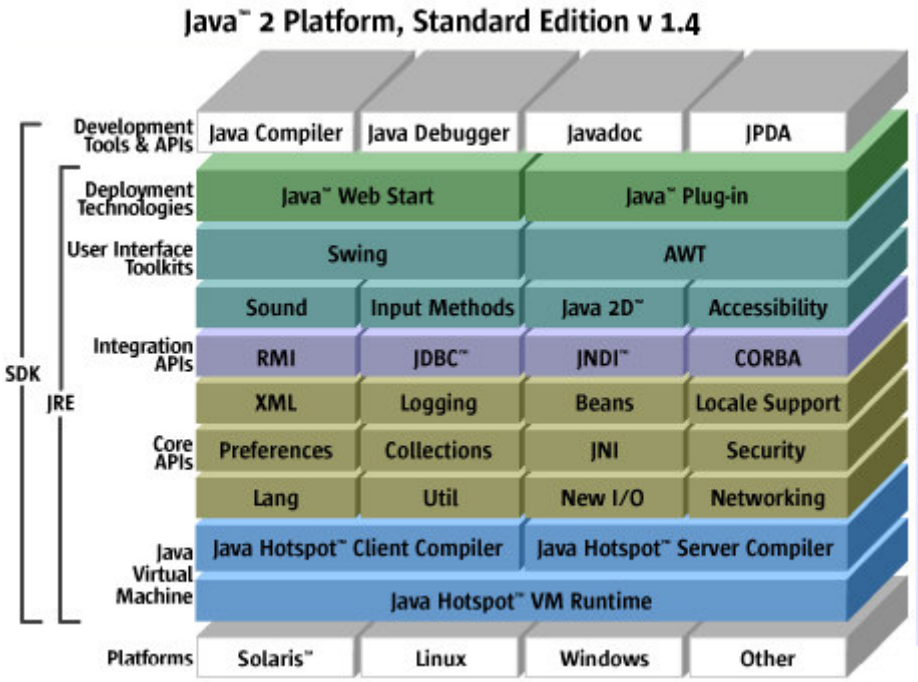


Figura 3. Pila de estándares de la plataforma Java2 Estándar Edition

J2SE API

La API (*application programming interface*) de J2SE define la manera establecida mediante la cual un applet o aplicación puede hacer peticiones o hacer uso de la funcionalidad disponible en las bibliotecas de clases de J2SE. Las librerías de clases J2SE son también parte de la plataforma J2SE.

La API J2SE consiste en una serie de tecnologías que pueden organizarse en dos grupos: *Core Java* y *Desktop Java.*

- *Core Java* provee la funcionalidad esencial para escribir protentes aplicaciones paa empresas en áreas claves como el acceso a bases de datos, seguridad, *remote method invocation (RMI*), y comunicaciones, por nombrar algunas.
- *Desktop Java* provee un rango completo de características que ayudan a construir aplicaciones de escritorio. *Desktop Java* consiste en el despliegue de productos tales como *Java Plug-in*, *JavaBeans, graphical user interface (GUI)* APIs tales como *Java Foundation Classes (JFC)* y *Swing*, y multimedia APIs tales como Java3D.

Java Virtual Machine

La *Java Virtual Machine* es responsable de la independencia del hardware – y sistema operativo- respecto a la plataforma J2SE, del pequeño tamaño del código compilado (*bytecodes*), y de la plataforma de seguridad.

Herramientas de la Plataforma Java

La plataforma J2SE trabaja con un array de herramientas, incluyendo *Integrated Development Environments (IDEs)*, funcionamiento y herramientas de pruebas, y herramientas de supervisión del funcionamiento.

4.4 PersonalJava

4.4.1 Introducción

La interfaz de programación de uso de *PersonalJava* es una colección de paquetes, de clases y de métodos definidos por una especificación de alto nivel.

La tecnología *PersonalJava* es la tecnología de Java diseñada específicamente para construir dispositivos de usuario web-conectables para el hogar, la oficina, y uso móvil. Se compone de una *Java Virtual Machine* y de un sistema de bibliotecas de clases de Java cargadas específicamente para dispositivos tales como *set-top boxes,* teléfonos con servicios Web y PDAs. Además, el uso de memoria del entorno de ejecución de *PersonalJava* se ha optimizado para funcionar en ambientes con recursos limitados proporcionando una fidelidad web cercana a la ofrecida para dispositivos de escritorio. Implantando el entorno de ejecución de *PersonalJava* en un dispositivo, el fabricante del dispositivo da al consumidor la opción de hacer correr sobre el mismo una amplia gama de *applet y* de aplicaciones escritos en el lenguaje de programación Java. De igual modo, escribiendo *applets* y aplicaciones para la especificación de *PersonalJava*, el programador del software se puede asegurar la compatibilidad al funcionar el software en una amplia gama de dispositivos de consumidor.

Actualmente Sun está haciendo la fuente del entorno de *PersonalJava*, el código fuente que implementa la especificación de la API *PersonalJava* en ejecución, disponible para la comunidad de desarrolladores como parte del *Sun's Community Source Licensing Program*. Este lanzamiento está motivado por el deseo de propulsar más lejos la innovación de la plataforma de Java y para proveer a los programadores de un acceso más fácil a las fuentes de la plataforma de Java mientras que se mantenga la compatibilidad. Con el código fuente, los programadores del software pueden:

- Arreglar plataformas o embotellamientos del funcionamiento que están impidiendo el desarrollo o el despliegue de una aplicación.
- Implementar nuevas APIs que necesiten acceso a plataformas internas.

Los productos de la familia *PersonalJava* comenzarán en breve el proceso de *Sun 'End of Life' (EOL*). Durante el período de transición de EOL, estos productos serán apoyados según acuerdos existentes de ayuda al cliente. Después del período de transición de EOL, Sun no dará soporte durante más tiempo a estos productos.

Durante el período de transición de EOL, los programadores deben comenzar a emigrar a los productos dentro de la familia *Java 2 MicroEdition (J2ME).*

Éstos incluyen:

- *Connected Device Configuración (CDC)* es un marco estándar para la construcción y desarrollo de aplicaciones móviles de uso para una variedad de dispositivos móviles personales. La CDC se diseña para productos con recursos típicamente de 2 MB de RAM y 2.5 de la ROM para el ambiente de uso de Java. La CDC también se basa en compatibilidad con las APIs J2SE estándar. La familia de la CDC incluye *Foundation Profile*, *Personal Basis Profile* and *Personal Profile.*

- *Connected Limited Device Configuration (CLDC*) es el ambiente de uso general más pequeño de Java. Se diseña para productos con recursos muy limitados, típicamente de 128 KB a 512 KB de RAM. La familia de CLDC incluye el Mobile Information device Profile (MIDP) que agrega capacidades del GUI a los dispositivos basados en CLDC.

Puesto que *PersonalJava* se encuentra en vías de extinción en cuanto a soporte por parte de la empresa que lo ha desarrollado, se presupone que el sowftware desarrollado con esta tecnología deberá migrar a *Java2MicroEdition* en un periodo de tiempo corto-medio. Por lo tanto más adelante se expondrá con detalle las características de esta tecnología y las distintas posibilidades y configuraciones que ofrece.

Especificación PersonalJava™ 1.0

El entorno de aplicación *PersonalJava*, basado en un principio en las APIs JDK 1.1 APIs, con algunos paquetes de JDK 1.2, requiere completo soporte de ambos, *The Java Language Specification* y *The Java Virtual Machine Specification*. La más reciente y quizás la última versión de la especificación de PersonalJava es la versión 1.2.

Para minimizar demandas de memoria, la especificación de PersonalJava define dos tipos de paquetes: paquetes obligatorios que todas las implementaciones deben soportar, y paquetes opcionales. El entorno de aplicación de PersonalJava no soporta las APIs de interfaz de usuario Swing y en cambio provee APIs modificadas AWT. También hay algunas tecnologías específicas PersonalJava technology-specific relativas a AWT y procesamiento Timer.

El entorno de aplicación completo de PersonalJava incluye un kit de desarrollo software development kit y librerías de clases optimizadas, una implementación de referencia y un entorno de emulación, y herramientas tales como un *applet viewer* y una utilidad llamada J*avaCheck* que ayuda a validar una aplicación frente a la especificación de *PersonalJava. L*a tabla siguiente lista los paquetes soportados por la especificación de PersonalJava.

Paquetes obligatorios (disponibles en todas las implementaciones de PersonalJava)	`java.applet, java.awt, java.awt.datatransfer, java.awt.event, java.awt.image, java.awt.peer, java.beans, java.io, java.lang, java.lang.reflect, java.net, java.security, java.security.cert, java.text, java.text.resources, java.util, java.util.zip`
Paquetes opcionales (no necesariamente disponible en una implementación de PersonalJava dada)	`java.math, java.rmi, java.rmi.dgc, java.rmi.registry, java.rmi.server, java.security.acl (unsupported), java.security.interfaces, java.security.spec, java.sql, java.util.jar` El acceso a una clase o paquete no soportado provoca un error *NoClassDefFoundError* o en lanzamiento de una excepción *UnsupportedOperationException*.

Tabla 1. PersonalJava Specification 1.2a Core Java Packages

4.4.2 Entorno de Emulación para PersonalJava

El entorno de emulación para *PersonalJava* ayuda a verifica que los applet y las aplicaciones desarrollados con el kit de desarrollo de Java (JDK) funcionen en la puesta en práctica en un entorno de ejecución *PersonalJava*. El software del entorno de emulación de *PersonalJava* permite:

- Probar los applet que funcionarán en los browsers que apoyan el ambiente de ejecución de *PersonalJava*
- Probar las aplicaciones que funcionarán en el ambiente de ejecución de *PersonalJava*

El entorno de emulación *PersonalJava* no incluye un compilador u otras herramientas de desarrollo. Deben de usarse las herramientas proporcionadas por el *JDK (Java*

Development Kit) para escribir y compilar los applets y aplicaciones. La siguiente tabla explica que versión del JDK usar cuando se escribe código para una versión específica de la *API PersonalJava.*

Especificación para la cual se escribe	Usar esta versión del JDK	Con clases compatibles PJCC
1.1, 1.1.x	1.1.6+	1.2
1.2	1.2	1.2

Tabla 2. Relación entre versiones PersonalJava y JDKs

Hay múltiples versiones del entorno de emulación de *PersonalJava* dependiendo de sus necesidades. Utilizar una de las versiones de *min+* si se desea tener un ambiente que incluya el sistema mínimo de bibliotecas definido por la versión relevante de la especificación de la *API PersonalJava* a menos que incluya la ayuda del archivo *I/O*.

Versión de la especificación	Ref. Imple. Versión	Plataforma de desarrollo	Look-and-Feel	Build Configuration
1.1, 1.1.x	3.0.2	Solaris/SPARC	Touchable	min+
1.1, 1.1.x	3.0.2	Solaris/SPARC	Motif	max
1.1, 1.1.x	3.0.2	Windows/x86	Touchable	min+
1.1, 1.1.x	3.0.2	Windows/x86	Win32	max
1.2	3.1	Solaris/SPARC	Touchable	min++
1.2	3.1	Solaris/SPARC	Motif	max
1.2	3.1	Windows/x86	Touchable	min++
1.2	3.1	Windows/x86	Win32	max

Tabla 3. Versiones de entorno de emulación PersonalJava

Utilizar una de las versiones de *min++* si se desea tener un ambiente que incluya el sistema mínimo de bibliotecas definido por la versión relevante de la especificación de la *API PersonalJava*, a menos que incluya la ayuda del archivo *I/O*, y se compile con la opción de construcción *LOCALE=ALL*.

Utilizar una de las versiones *máx* si se desea tener un ambiente que incluya el sistema máximo de bibliotecas definidas por la versión relevante de la especificación de la *API PersonalJava*.

Una versión mínima del entorno de emulación de *PersonalJava* es útil si se está escribiendo un applet o aplicación usando el denominador común más bajo de la especificación de la *API PersonalJava*. Esto asegura que el applet o aplicación pueda funcionar en cualquier dispositivo que tenga un entorno de ejecución *PersonalJava*. Una versión máxima del entorno de emulación de *PersonalJava* es útil si se tiene un dispositivo específico para la carga del applet o la aplicación y es seguro que ciertas partes opcionales de la especificación han sido implementadas. Observar que todas las versiones de la especificación son actualizaciones de menor importancia que clarifican y quitan errores en versiones anteriores. Así, la especificación de la versión 1.1.3 hace anticuada la versión 1.1.2. Semejantemente, la puesta en práctica de la referencia de la versión 3.0.2 hace anticuado la versión 3.0.1.

Compatibilidad de Versiones

En general, cualquier applet o aplicación que corre sobre versiones superiores a 3.0.2 de *PersonalJava* deberían de correr correctamente en la versión 3.1 del entorno de emulación P*ersonalJava emulation environment.*

Por supuesto, applets que dependen de las *APIs PersonalJava 1.1* no funcionaran en otro sistema *PersonalJava* que solo soporte *APIs PersonalJava 1.0.*

Contenido Software del Entorno de Emulación PersonalJava Runtime

- Clases del núcleo (*classes.zip*)

No se debe de descomprimir este fichero. Debe de permanecer comprimido para que el compilador y el intérprete accedan a las clases correctamente. Este fichero contiene todos los ficheros compilados *.class* de la plataforma.

- *PersonalJava Interpreter (pjava)*

Ejecuta bytecode Java. En otras palabras, corre el programa escrito en el lenguaje de programación Java.

- PersonalJava AppletViewer (pappletviewer)

Usado para chequear y correr applets

NOTA: El entorno de emulación PersonalJava no contienen un Web browser. Para obtener el HotJava(tm) Browser puede accederse a la página http:///java.sun.com/products/hotjava/

Estructura de Directorios del Software P*ersonalJava Emulation Environment*

La figura muestra la capa de directorios en el entorno de simulación de *PersonalJava*:

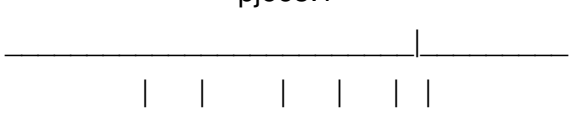


Figura 4. Directorios contenidos en PJEE3.1

El programa de instalación de PJEE continene el software necesario para correr PJEE en Microsoft Wdows NT o en la plataforma Solaris. PJEE no contiene un navegador web.

Instalando PJEE

PJEE puede ser instalado en Microsoft Windows NT o en la plataforma Solaris. Estos son los respectivos ejecutables para ambas plataformas que se encuentran en la página de descarga de PJEE.

- Microsoft Windows NT-> pjee3_1-win32.exe: aplicación auto-extraible para instalar PJEE
- Solaris 2.6 o superior-> pjee3-solaris.sh: *shell script* auto-extraible para instalar PJEE.

La siguiente tabla describe los contenidos software del directorio de PJEE después de la instalación.

Componente	Descripción	
COPYRIGHT	Aviso de Cpyright para PJEE	
LICENSE	Acuerdo de licencia para PJEE	
bin/pjava	Versión optimizada	El *PersonalJava applicaction*
bin/pjava_g	Versión de depuración	*launcher* carga y ejecuta las aplicaciones Java
bin/pjavaw	Versión optimizada	
bin/pjavaw_g	*Debug* Version	
bin/pappletviewer	Version optimizada	
bin/pappletviewer_g	*Debug* versión	
bin/*.dll	(Solo para *Microsoft windows NT).* DLLs *(Dynamic link libraries*) para la máquina virtual y métodos nativos de la librería de clases de PersonalJava	
bin/sparc	(Solo para la plataforma Solaris). Ejecutables binarios para la invocación de herramientas llamadas por scripts de consola en bin	
doc/*	Guia de usuario	
lib/appletviewer.properties	Mensajes de estado y seguridad para *sun.applet.AppletViewer*	
lib/awt.properties	Clave y nombre modificador usados por *java.awt.event.KeyEvent*	
lib/content-types.properties	Fichero de descripcion de tipos MIME usados por *sun.net.www*	
lib/font.properties	Fichero de fuentes dependiente de la plataforma	

lib/jvm.Prof..txt	Cabecera de texto para informes generados por java
lib/touchable.paletees	Base de datos que contiene los valores RGB para nombrar la paleta de colores
lib/classes.zip	Versión optimizada
lib/classes_g.zip	*Debug* versión
lib/sparc	(Plataforma Solaris solamente). Librerías compartidas por la máquina virtual y métodos nativos de la librería de clases de *PersonalJava*
lib/java.security	Fichero que contiene propiedades relativas a la seguridad
lib/java.policy	Fichero que especifica que *security police* se usa para las aplicaciones basadas en tecnología Java

Tabla 4. Contenidos después de la instalación de PJEE

Ejecutando *PersonalJava Emulation Environment Tools* en *Microsoft Windows*

Después de instalar el entorno de emulación de PersonalJava, puede iniciarse una herramienta tecleando el nombre en una ventana DOS con un nombre de fichero como argumento. Ninguna de las principales herramientas del entorno de emulación PersonalJava son programas Windows con interfaces GUI – todas son ejecutadas desde la línea de comandos de DOS. (Por ejemplo, si se hace doble-click sobre el icono de "pjava", esto abrirá brevemente una ventana de DOS que será cerrada inmediatamente, porque no es el método apropiado para ejecutarlo).

Es posible especificar la ruta a una herramienta escribiendo la ruta delante del nombre de la herramienta cada vez que vayamos a hacer uso de ella, o añadiendo la ruta al fichero de arranque (*autoexec.bat).* Por ejemplo, si el software del entorno de emulación *PersonalJava* es instalado en C:\pjee3.1, para ejecutar el intérprete sobre el fichero myfile.class, ir a DOS y ejecutar:

escribir: C:\pjee3.1\bin\pjava myfile

-o-

añadir C:\pjee3.1\bin al fichero de rutas preestablecidas

escribir pjava myfile

Seguridad JDK 1.2

En PersonalJava 3.1, la seguridad ha sido realzada para implementar la seguridad de JDK 1.2. Un fino *Access Control* es requerido en PersonalJava3.1.

Firma de código es opcional para Applets mientras que para aplicaciones no está soportado en PersonalJava 3.1. Aplicaciones como el correo electrónico deben de implementar el mecanismo de AppletClassLoader para verificar un fichero .jar mientras se está descargando.

En PersonalJava 3.1, "-*bootclasspath*" ha sido añadido como una opción de línea de comando para indicar clases “de confianza”. Todas las clases en *bootclasspath* son consideradas de confianza mientras que todas las clases contenidas en el *classpath* son consideradas de “no confianza”.

En PersonalJava 3.1, la seguridad JDK1.2 asigna "dominios de protección" para cada clase usando protocolos manejadores de URL. Las clases cargadas desde el CLASSPATH o *-classpath* son asignadas a dominios de protección usando el fichero manejador del protocolo. La estructura mínima por defecto no incluye un fichero de sistema opcional y manejadores de protocolo, el uso de *CLASSPATH/-classpath* podría lanzar excepciones. En cambio, clases en *-bootclasspath* son las clases de confianza y no necesitan dominios de protección. Ejemplo:

${JAVA_HOME}/lib/classes.zip sun.applet.Appletiewer http://<URL>

En PersonalJava3.1, el classpath por defecto ha cambiado a "" desde "." si no se especifica ningún classpath. Además, el classpath no puede ser accedido sin el soporte de sistema del fichero del sistema y el fichero de URL.

PersonalJava Compatibility Classes (PJCC)

El PersonalJava Application Environment (PJAE) incluye una pequeña lista de nuevas clases PJCC que no derivan de la API JDK. Las PersonalJava Compatibility Classes (PJCC) contienen una serie de ficheros Java con implementaciones de estas clases PJAE específicas que permiten al programador usar herramientas de programación basadas en la API JDK para compilar y ejecutar programas basados en tecnología Java que usan estas clases PJAE específicas. De este modo, las clases compatibles de PersonalJava permiten a los applets y aplicaciones desarrolladas para la API PersonalJava correr sobre en un entorno basado en JDK.

Las clases PJCC son:

- Paquete *com.sun.awt*
    - *com.sun.awt.ActionInputPreferred*
    - *con.sun.awt.KeyboardInputPreferred*
    - *con.sun.awt.NoInputPreferred*
    - *com.sun.awt.PositionINputPreferred*
- Paquete *com.sun.lang*
    - *com.sun.lang.UnsuportedOperationException*
- Paquete *com.sun.util*
    - *com.sun.util.PTimer*
    - *com.sun.util.PTimerSpec*
    - *com.sun.util.PTimerScheduleFailedException*
    - *com.sun.util.PTimerWentOffEvent*
    - *Com.sun.util.PTimerWentOffListener*

Contenido Software:

PJCC contiene dos series de ficheros:

- Pj.jar es un archivo jar que contiene PJCC clases
- Javadoc documentación para las clases PJCC

Notas adicionales

- Los programadores deberían usar JavaCheck para determinar que un programa Java usa solo clases y métodos que están disponibles en PJAE
- Los interfaces provistos en el paquete com.sun.awt están solo propuestos de manera indirecta. Actualmente son ignorados por la implementación PJCC.

Establecer la variable de entorno CLASSPATH:

La variable de entorno CLASSPATH define una lista de directorios que la máquina virtual de Java usa como ruta de búsqueda para encontrar ficheros .class. Eso permite al compilador de Java compilar código fuente Java para ejecutar ficheros .class que dependen de clases programadas con el lenguaje Java que no son parte de la librería de clases por defecto. Si un programa Java intenta acceder a una clase cuyo fichero.class no está en ninguno de los directorios en la variable de entorno CLASSPATH, la máquina Virtual de Java genera un error *NoClassDefFoundError.*

El fichero pj.jar debe de estar localizado en un directorio de la variable de entorno CLASSPATH. En los siguientes apartados se exponen los pasos para distintas plataformas específicas para modificar la variable de entorno CLASSPATH para incluir el directorio que contiene el fichero pj.jar.

4.4.3 JavaCheck

JavaCheck es una herramienta de desarrollo para comprobar que una aplicación o applet es compatible con un entorno particular de Java y puede, de este modo, correr sobre todos los dispositivos que implementan dicho entorno Java. Analiza ficheros de clases para encontrar dependencias no incluidas en una especificación particular de API Java. Esto ayuda a los programadores a escribir aplicaciones Java y applets que puedan correr de forma segura sobre distintas implementaciones de la plataforma Java.

JavaCheck lee ficheros .spc (*platform specification files*) de una concreta especificación de la API Java que ha sido codificada. Actualmente, los ficheros de especificación de plataforma están disponibles únicamente para la API PersonalJava.

Versión de la especificación PersonalJava API	**La cual corresponde a esta implementación del entorno de emulación de Sun**	**Usar esta versión de JavaCheck**
1.0	1.0	2.0.1
1.1x	1.1,3.0.x	3.0
1.2	3.1	No disponible actualmente

Tabla 5. Relación entre versiones de PersonalJava y JavaCheck

Cada nueva versión de la especificación son pequeñas actualizaciones que clarifican y eliminan errores de las versiones previas.

JavaCheck usa un fichero de especificación de plataforma como la definición de la plataforma Java. Un fichero de especificación de plataforma es descrito o definido usando SGML y tiene una extensión de archivo .spc. J*avaCheck* lee un fichero de especificación de plataforma para representar la plataforma en memoria de modo que se pueda determinar si un programa Java es conforme a la plataforma. Los ficheros de especificación de plataforma son creados por *Sun Microsystem* para ser usados con

especificaciones oficiales Java. J*avaCheck* incluye un fichero de especificación para la plataforma *PersonalJava* de modo que se puede determinar que código es conforme con la plataforma. Otros ficheros .spc para otras plataformas estarán disponibles en futuras ediciones de *JavaCheck.*

JavaCheck 3.0 viene bajo dos configuraciones: una esta basada en línea de comandos, la otra tiene una interfaz gráfica de usuario (GUI). Ambas configuraciones proveen la misma utilidad.

JavaCheck 3.0 informa de las dependencias para campos y métodos. Esto significa que se verán menos mensajes asociados a las clases, solamente el uso específico de campos o de métodos modificados generará estos errores. JavaCheck 3.0 permite a los usuarios especificar rutas, esto simplifica el proceso de especificación de las clases que deben ser chequeadas. JavaCheck 3.0 también permite filtrar clases específicas a cargar (Ejemplo: solo cargar clases en el paquete xx.yy). Finalmente, JavaCheck 3.0 permite especificar al usuario las acciones concretas a tomar cuando se encuentran clases duplicadas.

La especificación del formato ha cambiado por lo que en la versión JavaCheck 3.0 respecto a la anterior, por lo que JavaCheck 2.0.1 no es capaz de leer el fichero de especificación de PersonalJava 1.1 (pJava_1.1.0.spc). Igualmente, JavaCheck 3.0 no leerá correctamente el fichero de especificación de PersonalJava 1.0.1 (pJava1.0.spc).

4.5 J2ME

4.5.1 Introducción

Qué es J2ME y Donde se Aplica

Java 2 Micro Edition es un entorno runtime y de desarrollo diseñado para poner software Java en dispositivos empotrados y consumibles electrónicos. Como ocurre en muchas ocasiones, cuando se habla de software, una única especificación no suele cubrir todas las necesidades. Es de esperar que una única tecnología Java que corra sobre todo tipo de dispositivo, desde un mainframe a un teléfono móvil, sea inconcebible.

Hoy día, Java se ha convertido en uno de los lenguajes de programación más populares. Esto es debido, y no en pequeña medida, a la habilidad de Java para correr sobre todo tipo de plataformas. J2ME trata de que el lenguaje de programación Java esté disponible para incluso un rango mayor de dispositivos y más diversas plataformas. En particular, J2ME lleva Java al mundo de los dispositivos personales de información y comunicación. Normalmente, estos dispositivos son más pequeños y con menos potencia que los computadores tradicionales. Así pues, la tecnología J2ME supone un esfuerzo por condensar y reducir el estándar Java.

El desarrollo de J2ME fue iniciado por *Sun Microsystems*, pero ahora es soportado por algunos de los más importantes fabricantes de dispositivos electrónicos del mundo. En particular, muchos de los vendedores de tecnologías móviles y Wireless están explorando o participando activamente en la tecnología J2ME. Aquellos que dan soporte a J2ME lo hacen bajo una comunidad de desarrollo de Sun, para estandarizar y guiar las direcciones futuras de todos los aspectos de Java. Este proceso es llamado *Java Community Process (JCP).*

Mientras que Java corre sobre cualquier dispositivo, desde mainframes a ordenadores portátiles, no ha sido hasta hace relativamente poco tiempo que Sun ha vuelto a centrar su interés en acercar Java a los dispositivos pequeños. Decimos volver a

centrar porque Java fue inicialmente desarrollada para asistir a los programadores respecto a dispositivos de control digitales (TV,VCR,*video disc players*, etc).

El renacimiento de Java, a través de J2ME, como un lenguaje de programación y plataforma software para pequeños dispositivos es significante en cuanto que el número de estos dispositivos superará con mucho, el número de sistemas informáticos tradicionales en un futuro próximo. Como una nueva tecnología, J2ME está todavía en desarrollo y la base de soporte para J2ME está aún creciendo. De cualquier modo, J2ME y otras tecnologías basadas en Java, ofrecen a la comunidad de desarrollo software un soporte de cara al futuro, que abarca todo tipo de plataformas.

J2ME es una tecnología que ha encontrado cabida dentro de muchos dispositivos empotrados y electrónicos, algunos de los cuales son de uso diario. J2ME es una tecnología cuyo uso se puede prever sea más frecuente que el de otros estándar Java u otros lenguajes de programación software. Esto es debido a que las aplicaciones software J2ME están destinadas a dispositivos muy personales, sobre los cuales las personas están desarrollando una gran dependencia. Un gran número de teléfonos móviles ya contienen J2ME.

Las aplicaciones software J2ME controlan o proveen algún tipo de servicio en los teléfonos móviles, *pagers*, asistentes personales digitales(PDAs), televisiones, VCRs, sistemas electrónicos de entretenimiento, etc. J2ME está ayudando a extender sistemas corporativos de empresa (tanto datos como aplicaciones) a dispositivos Wireless y móviles.

Qué es un 'Small Device'

Con el término *small devices* o pequeños dispositivos se hace referencia genéricamente a la plataforma sobre la que corre J2ME. Es importante hacer distinciones entre los dispositivos que conforman este grupo. Esto es debido a que frecuentemente son los dispositivos los que definen la variedad de la estructura J2ME. Vamos a definir algunos términos que aparecen con frecuencia cuando se habla de *small devices*. Una '*smart card'* es un dispositivo de plástico del tamaño de una tarjeta de crédito con un circuito integrado construido en su interior. Un *set-top box* es un dispositivo electrónico que produce salida para la televisión convencional, y que también se conecta a canales de comunicación para permitir al usuario, o más apropiadamente, al visor de televisión, interactuar de algún modo. Algunos de los dispositivos incluidos en el grupo calificado como *small devices* son los siguientes:

- *pagers*
- teléfonos móviles
- asistentes personales digitales (PDAs)
- sistemas de puntos de venta
- comunicadores de bolsillo
- sistemas de navegación para coches
- sets de internet para televisión

Generalmente, J2ME incluye dispositivos con capacidades mínimas de memoria, ancho de banda de comunicaciones, potencia, y capacidades de interfaz de usuario. De este modo, es considerado adecuado para suplir necesidades de programación sobre dispositivos con capacidades mayores que una *smart card* pero menores que un ordenador personal. Un término frecuentemente usado para este tipo de dispositivos es el

de *information appliance.* Los *information appliances,* proveen menos potencia que un ordenador personal, y están considerados para tener una función especial. En muchas ocasiones, estos dispositivos son de una naturaleza más personal, es decir, son propiedad de una sola persona, que es la que los opera. Además, al contrario que los ordenadores portátiles, estos dispositivos son casi siempre llevados por sus dueños, en un bolsillo o bolsa de mano.

Como una tecnología separada, Java para smart cards (*Java Card Tecnology*) dispone de su propia especificación. Para dispositivos con más memoria, potencia, y capacidades, se dispone de la especificación *Java 2 Standar Edition (J2SE),* definida para ser usada sobre PCs.

En ocasiones, los dispositivos empotrados, *'embedded devices'*, son excluidos de J2ME. Para cubrir este rango de dispositivos Sun dispone de una tecnología llamada E*mbeddedJava*.

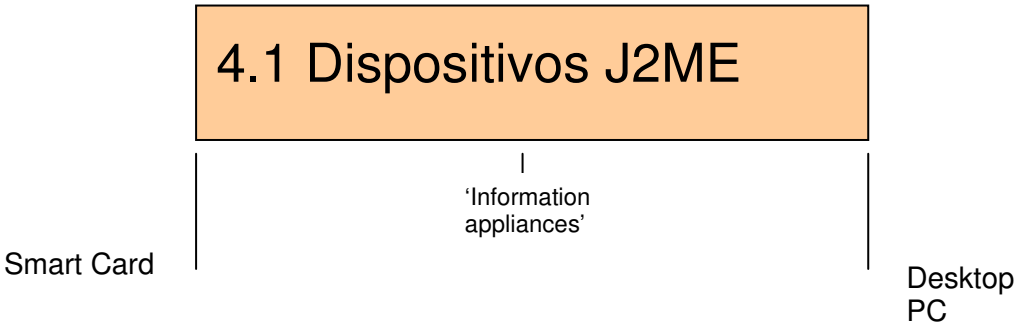


Figura 5. Rango de dispositivos soportados por J2ME

Los dispositivos que son específicamente soportados por J2ME van desde Smart Cards a ordenadores personales de escritorio o portátiles. CLDC Y CDC son especificaciones que define J2ME en ambos extremos del espectro de dispositivos. Ambas especificaciones serán ampliamente descritas más adelante.

4.5.1 El Papel de J2ME en Aplicaciones Wireless Y Móviles

La plataforma J2ME es frecuentemente referida como Java para dispositivos móviles y Wireless. De hecho, esta tecnología es usada en muchos dispositivos móviles y Wireless, pero no es usada únicamente en estos entornos. Hay que aclarar que J2ME cubre un área más allá de los dispositivos Wireless y móviles.

¿Es J2ME Móvil?

Dispositivos móviles son definidos como aquellos que son lo suficientemente pequeños como para que sus usuarios puedan llevarlos consigo en todo momento, y además puedan ser usados mientras se transportan. Estos dispositivos proveen a los usuarios de una parte de la capacidad de computación e información disponible en sistemas de información fijos que pueden estar ubicados en sus hogares o lugares de trabajo. En general, la mayoría de dispositivos móviles son capaces de sincronizarse con sistemas fijos, para actualizaciones de software y datos.

Como ejemplo, una oficina debe disponer de una aplicación de gestión de usuarios que ponga a disposición del personal de ventas la información de los clientes. Un dispositivo móvil permitiría a un comercial descargar una cantidad limitada de datos de clientes para un limitado número de clientes, y de este modo hace uso de ella mientras

está en la carretera. Las actualizaciones de cualquier dato en el dispositivo móvil requerirían la conexión de este con el sistema fijo de administración de clientes situado en la oficina.

Con este tipo de definición, el término móvil esta sujeto a cambio, de hecho hace algún tiempo se podría haber denominado a un ordenador portátil de 4 kilos como un dispositivo móvil. Ciertamente, muchos, pero no todos los dispositivos que abarca J2ME pueden ser considerados como móviles. PDAs, teléfonos móviles, y tantos otros, pueden ser considerados como plataformas móviles cuando están provistos de software y datos. De cualquier modo, los *set-top boxes*, por ejemplo, no deben de ser considerados dispositivos móviles. Aunque estos dispositivos corren programas escritos en J2ME, no son móviles.

Además, otras tecnologías Java, J2SE y *Java Card Technology* incluidas, son usadas sobre sistemas móviles. Así que mientras J2ME es una importante tecnología Java para plataformas móviles, no es la única tecnología de Java para estas plataformas. Y Java no es la única solución.

El término móvil simplemente define la capacidad o el estado de un dispositivo. Así que cabe preguntarse, ¿es J2ME móvil ?. Debido a que los dispositivos móviles son usualmente más pequeños y de capacidades limitadas, J2ME es una solución desarrollada viable para estos dispositivos. Además, J2ME puede y frecuentemente lo hace, jugar un importante papel en los dispositivos móviles, pero el término móvil no clasifica todas las aplicaciones J2ME.

¿Es J2ME Wireless?

Un dispositivo Wireless es simplemente un dispositivo con capacidad para comunicaciones sin cables. Muchos dispositivos J2ME son Wireless. Los teléfonos móviles*, pagers*, comunicadores de bolsillo son solo algunos de los dispositivos con comunicaciones Wireless que pueden usar tecnología J2ME. La lista de estos dispositivos está constantemente creciendo. Muchos de los ordenadores portátiles y PDAs de hoy día poseen adaptadores para comunicaciones Wireless para permitir a estos dispositivos trabajar sin depender de un medio físico tal como el cable.

Se pretende que los dispositivos Wireless se comporten como si estuviesen conectados directamente a la red de trabajo mediante cable. Desde la perspectiva del usuario, debería parecer que cualquier dato o aplicación es local al dispositivo o que está directamente conectado al dispositivo que provee los datos o aplicaciones. Por ejemplo, un comercial podría usar el teléfono móvil para buscar información sobre un cliente que está en un sistema de gestión de clientes en la oficina. Para el comercial, debería parecer que los datos y/o aplicaciones obtenidas y mostradas son locales al móvil, cuando en realidad, los datos han sido transmitidos vía Wireless al dispositivo personal.

De cualquier modo, hay bastantes dispositivos que soportan J2ME que no son Wireless. Otra vez, los *set-top boxes* y televisores son normalmente cableados. De hecho, una gran porción de J2ME está establecida para sistemas de los cuales se esperan que tengan una segura, rica y de alta fidelidad conectividad a redes de trabajo, lo cual usualmente implica tener conexiones vía cable. La tecnología Wireless soporta muchos dispositivos Wireless, pero no es la tecnología Java para sistemas y dispositivos Wireless. De hecho, otras tecnologías tales como *Wireless Access Protocol* y *Wireless Marckup Language* están destinados a proveer capacidades Wireless a los dispositivos sin tener que proveer necesariamente aplicaciones móviles.

Wireless define un tipo de comunicación usado por el dispositivo. J2ME puede y frecuentemente es una parte importante de una solución Wireless. Pero mientras que Java y J2ME pueden ser usadas en dispositivos y aplicaciones Wireless, no todas las aplicaciones J2ME son Wireless.

Wireless vs MóvilL

Los términos Wireless y móvil son usados frecuentemente de manera indistinta. Erróneamente, estos adjetivos son directamente aplicados a aplicaciones J2ME. Debido a que los dispositivos J2ME son normalmente pequeños y pueden almacenar un cierto número de datos y aplicaciones y son fáciles de transportar, J2ME puede ser usado con frecuencia en sistemas móviles. Igualmente, debido a que J2ME es usado con frecuencia en aplicaciones que comunican información a través de redes Wireless, J2ME pueden usarse frecuentemente en sistemas Wireless. De nuevo, esto no significa que todas las aplicaciones J2ME son Wireless y móviles.

Un dispositivo de tipo PDA almacenando una pequeña cantidad de datos de clientes y una aplicación para ver y actualizar los datos de clientes, es móvil, pero no necesariamente Wireless. Si el usuario requiere conectar el dispositivo a una red de trabajo o a otro dispositivo con el propósito de bajar o actualizar datos, permanece siendo móvil, pero también cableado.

De manera alternativa, un teléfono móvil equipado con un pequeño navegador que permite al dispositivo mostrar pequeñas o especiales páginas web. Este dispositivo es considerado Wireless, pero no móvil. Con el fin de ser móvil, el dispositivo debe ofrecer algún valor, en forma de datos o funciones de aplicación, cuando no se conecta a otros sistemas.

Mucha de la confusión entre Wireless y móvil, existe porque muchas de las aplicaciones J2ME están diseñadas para trabajar en ambos sistemas. Por ejemplo, una aplicación de información de clientes construida sobre una PDA es también equipada con una función de actualización que transmite los datos al dispositivo mediante comunicación Wireless y almacenados en el mismo para su posterior uso. En este caso, la PDA es un dispositivo Wireless y móvil

4.5.3 J2ME como Plataforma

Sun introdujo *Java 2 Micro Edition* en Junio de 1999 en la convención anual JavaOne. J2ME está diseñado para albergar las necesidades Java de la comunidad de dispositivos empotrados y electrónicos personales. Inicialmente, J2Me fue construido para dispositivos con capacidades limitadas de memoria, conectividad (normalmente Wireless), y capacidades de interfaz gráfica de usuario. Hoy por hoy, como se verá, la tecnología J2ME se ha expandido para cubrir un rango de dispositivos que van desde *pagers* hasta, pero no necesariamente incluidos, los ordenadores personales.

Por qué es Necesario J2ME

Desde que Java fue inicialmente dirigida para dispositivos electrónicos de consumidor, una pregunta natural sería, ¿Por qué otra edición? ¿Por qué no simplemente usar el estándar Java para los dispositivos pequeños? Además de las necesidades de separar la multitud de APIs en tres ediciones distintas para una mejor organización y, posiblemente, la necesidad de Sun de obtener beneficios, hay otra razón, más convincente, para introducir otra edición de Java: los dispositivos para los que J2ME está adaptado, tienen necesidades especiales.

Estos dispositivos tienen requisitos de software distintos a los grandes entornos de aplicación software. En general, el software debe de tener poco peso en la memoria. En algunos casos, una máquina virtual es implementada en unos cientos de kilobytes.

Además, las aplicaciones software destinadas para este tipo de dispositivos empotrados usualmente tienen mecanismos únicos de despliegue. Por ejemplo, los

dispositivos tipo PDA frecuentemente tienen lo que es conocido como un dispositivo 'cradle' que es adjuntado al ordenador de escritorio para descargar aplicaciones y datos.

Finalmente, estos dispositivos tienen interfaces de usuario, redes de trabajo, y otras necesidades que no pueden ser albergadas por una Java API que abarque toda la memoria. El paquete Swing de Java para el desarrollo de interfaces de usuario puede ser aumentado para el uso de componentes en la construcción de interfaces de usuario orientadas a ser usadas sobre pequeñas pantallas, tales como las de los teléfonos móviles. Pero, ¿puede este paquete ser adecuado en la memoria de un teléfono móvil?.

J2ME pone de manifiesto el hecho de que un entorno de Java de un único tamaño no puede realmente ser adecuado para todos los dispositivos. Los mismos principios de independencia de la plataforma, sintaxis del lenguaje, y seguridad son adheridos a todas las ediciones de Java, incluyendo J2ME. De cualquier modo, la separación de ediciones contempla las necesidades específicas inherentes en el rango de los pequeños dispositivos que la edición J2ME cubre.

Sun contempla la escalabilidad desde J2ME a otras ediciones de Java (digamos J2SE o J2EE) como una importante característica de J2ME. Para una aplicación desarrollada en el entorno de los pequeños dispositivos su transición a un entorno de Java más potente puede ser posible dependiendo de la arquitectura de la aplicación.

Java como Plataforma de Desarrollo

Antes de entrar más profundamente en los beneficios de J2ME deberíamos dar un paso atrás y preguntar incluso una cuestión más fundamental, específicamente, ¿Por qué el lenguaje de programación Java es importante para el desarrollo de software para dispositivos pequeños y dispositivos empotrados? Vamos a hacer una revisión de algunas de las razones por las que Java es un lenguaje tan popular y porque es un sólido contenedor de software para pequeños dispositivos.

¿Es Java Adecuado para 'Small Devices'?

Como ha sido ya mencionado, Java fue primeramente diseñado y construido como una plataforma común para soportar el desarrollo de software para una serie de dispositivos electrónicos de consumidor. Parece adecuado y natural volver a la raíz para ser usado en una colección heterogénea de pequeños dispositivos.

El término clave aquí es heterogéneo. Si se está usando un particular dispositivo de información, se puede encontrar que algún otro lenguaje o entorno de desarrollo cubre las necesidades al igual, o incluso mejor que lo hace Java. Por ejemplo, muchos de los fabricantes de PDAs proporcionan kits de desarrollo de programación que producen aplicaciones para sus sistemas. Estos kits frecuentemente toman las ventajas de las características de la específica plataforma y típicamente lo llevan a cabo mejor sobre ese dispositivo, que otros programas desarrollados con lenguajes más genéricos tales como Java. Como ha sido expuesto, Java tiene una rica serie de APIs y clases que vienen con las bases de los entornos. Un fabricante de dispositivos no tiene que ofrecer tan rico entorno. Debido a que muchos de estos entornos de programación y lenguajes producen código ejecutable como C, características no usadas son eliminadas en tiempo de compilación. En contraste, el entorno runtime de Java debe de estar preparado para interpretar cualquiera y todas de las instrucciones soportadas.

Así que cabe preguntarse, ¿Donde va a usarse la aplicación a desarrollar? ¿Debe ser capaz de poderse ejecutar sobre una variedad de pequeños dispositivos de información tales como PDAs, teléfonos móviles o incluso set-top boxes? Si esto es lo que se pretende, lo que se necesita es portabilidad. Lo que se quiere es tener que escribir la aplicación el menor número de veces posible. Programas escritos en entornos

de desarrollo propiedad de fabricantes, u otros lenguajes de programación no serán portables a un tal rango de dispositivos.

¿Hay otras razones para usar Java en pequeños dispositivos? En algunos casos, Java es la única forma de desplegar una aplicación en un dispositivo. Por ejemplo, algunos fabricantes de teléfonos móviles soportan solo Java.

Cuando se considera que lenguaje de programación usar para aplicaciones que corren sobre pequeños dispositivos, debería considerarse también que lenguajes de programación se están usando en los dispositivos de más peso así como en el sistema. Si se está utilizando una aplicación Java corriendo en el servidor y en el sistema de escritorio, usar Java en el *small device* elimina el uso de otro lenguaje en el desarrollo del entorno. Además esto puede suponer algunas oportunidades de reutilización de código.

J2ME Productos y Alternativas

Mientras que Sun ha conducido muchos de los desarrollos de Java, esto no significa un bloqueo sobre la tecnología. Esto es quizás más evidente en el mundo de los pequeños consumibles electrónicos y dispositivos empotrados donde otras organizaciones han hecho productos disponibles antes de que J2ME fuese lanzado

J2ME no es una implementación, más bien una especificación (o siendo más precisos, una serie de especificaciones). Sun tiene implementaciones de algunas de las especificaciones, pero también otras organizaciones. La razón de ser de llevar a cabo otra s implementaciones de J2ME es que Sun no ha sido siempre el mejor implementador de las especificaciones que han fomentado y conducido. Por ejemplo, es generalmente considerado a través de la comunidad Java que IBM provee un compilador más rápido y superior de Java llamado Jikes. Así mismo, otras organizaciones están construyendo máquinas virtuales Java y APIs que satisfacen las especificaciones de J2ME pero como un menor coste de memoria y mejor funcionamiento que las implementaciones de Sun.

Hay otros competidores Java ara J2ME que corren en los mismos ambientes generales, pero no cumplen con las especificaciones de J2ME. Sin tener en cuenta los esfuerzos iniciales de Sun con Java sobre consumibles electrónicos, J2ME es una tecnología muy joven. También direcciona un problema de muy difícil solución. A saber, este problema es como escribir software en un único lenguaje de programación para dispositivos tan diversos como un pagers o un set-top box de televisión. Otras organizaciones han decidió centrarse en proveer Java para una serie menos diversa de dispositivos. Se deposita entonces más responsabilidad sobre el programador para asegurar que las aplicaciones son realmente independientes de la plataforma.

Por supuesto, otra alternativa a J2ME es no usar Java en definitiva, y en su lugar usar otros lenguajes de programación tales como C/C++ o usar un dispositivo fabricado que venga con un kit de desarrollo software. En general, Java provee el grado de plataforma independiente, lo cual es importante paral os productores de software enfocado a un amplio rango de dispositivos. La independencia de la plataforma sin embargo, no tiene por que ser importante para los desarrolladores de un solo dispositivo, situación en la cual el uso de otro lenguaje no solo es válido sino que incluso puede ser preferible.

4.5.4 La arquitectura de J2ME

4.5.4.1 Introducción

La arquitectura J2ME comprende una variedad de configuraciones, perfiles y paquetes opcionales entre los que los programadores y desarrolladores pueden elegir y combinar para construir un completo entorno Runtime Java que agrupe lo más

cercanamente posible los requisitos de una gama particular de dispositivos y de un mercado. Cada combinación esta optimizada para la memoria, capacidad de proceso, y capacidades I/0 de una categoría de dispositivos relacionados. El resultado es una plataforma Java común que toma todas las ventajas de cada tipo de dispositivo para entregar una experiencia rica del usuario.

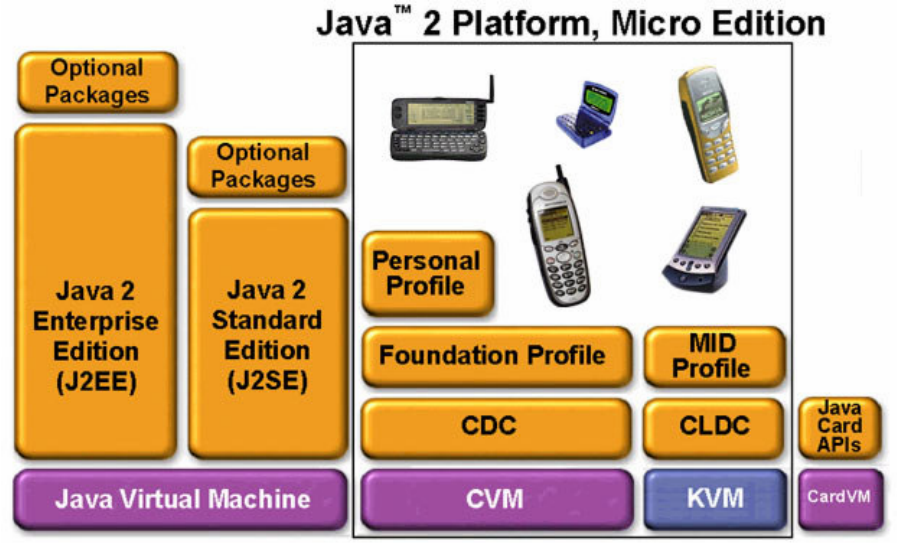


Figura 6. Plataformas Java

Objetivos de la Arquitectura J2ME

J2ME tiene una serie de metas muy distintas comparado con J2SE o J2EE, resultando en una arquitectura mucho más diferente. A continuación, un sumario de los metas claves en un recorrido por la arquitectura J2ME:

Proveer soporte para una variedad de dispositivos con diferentes capacidades. Estos dispositivos frecuentemente varían en las áreas de interfaces de usuario, almacenamiento de datos, conectividad de red y ancho de banda, presupuestos de memoria, consumo de potencia, seguridad, y requisitos de despliegue. Proveer una arquitectura que pueda ser optimizada para pequeños espacios. Centrada en dispositivos que pueden ser altamente personalizados, normalmente usados por una sola persona.

Proveer conectividad a red a lo largo de un rango variable de capacidades de red de trabajo y servicios. La conectividad de red es frecuentemente vital para los dispositivos en el espacio de J2ME y el rango de sus capacidades, desde bajo ancho de banda, Wireless, e intermitentes conexiones de alta fidelidad, conexiones de alto ancho de banda.

Proveer optimizados medios para entregar aplicaciones y datos sobre conexiones de red. Normalmente la red de trabajo es el método preferido para entrega de aplicaciones J2ME a los dispositivos. Las aplicaciones deben de tener la habilidad de ser instaladas en el dispositivo o descargadas directamente en la memoria y desechadas después de la ejecución.

Maximizar las capacidades de la plataforma del lenguaje Java, al mismo tiempo que se toman las ventajas las capacidades y obligaciones de cada dispositivo de cada dispositivo.

Maximizar la flexibilidad y proporcionar un medio para soportar los rápidos cambios del mercado y adaptarse a las existentes y a las imprevistas aplicaciones.

Proveer un medio para que los programadores puedan escribir y desplegar aplicaciones para dispositivos que soportan J2ME independientemente del OEM (Original Equipment Manufacturer).

Proveer un medio para escalar las aplicaciones a través de dispositivos con diferentes capacidades, características, y habilidades de proceso.

A continuación se da una explicación más detallada de algunos de estos objetivos.

Soporte para Múltiples Dispositivos

El soporte para múltiples dispositivos es una meta que ha tenido una gran influencia sobre la arquitectura J2ME. En el espacio J2ME de dispositivos desde un teléfono móvil, que debe de tener una memoria tan pequeña como 160kB y están alimentado por una batería, hasta la parte más alta del rango donde encontramos los Tv set-top boxes que son casi tan potentes como un ordenador personal y se sitúan en la pared. Por el contrario que las arquitecturas J2SE y J2EE, que están diseñadas para servidores y ordenadores de escritorio, la arquitectura de J2ME debe ser lo suficientemente flexible para acomodarse a las obligaciones y únicas características de los *small devices* sin tener que imponer innecesariamente restricciones sobre dispositivos más potentes o aplicaciones de Internet.

Un objetivo clave que viene con el soporte para múltiple tipos de dispositivos es permitir la portabilidad entre los dispositivos. Con este fin, J2Me identifica el núcleo de las características de Java que necesitan estar disponibles en todas las plataformas. Estas características incluyen clases como *java.lang, java.util y java.io.* De cualquier modo, es importante hacer notar que en ocasiones, J2ME solo soporta un subconjunto de clases y métodos de estos paquetes como el núcleo de la funcionalidad de Java para todas las plataformas J2ME.

Soporte para Dispositivos de Funcionalidad Específica

La flexibilidad es otra meta que tiene una significante influencia en la arquitectura de J2ME. Los dispositivos electrónicos de consumidor y las aplicaciones de Internet normalmente ofrecen específicos servicios, más que servir para una máquina de propósito general. Con el fin de llevar acabo estos papeles más específicos, los dispositivos tienden por si mismos a ser más personalizados. Un teléfono móvil será usado en principio por una única persona. Las PDAs tienden a almacenar información específica de la persona que las posee.

La modularidad de la arquitectura de J2ME es un buen ejemplo de como acomoda esa flexibilidad. A diferencia de J2SE que provee un rico set de características para un único grupo, J2ME provee un medio para particionar estas capacidades en unidades independientes de funcionalidad. Vamos a examinar el uso de RMI (Remote Method Invocation) como ejemplo. Un teléfono móvil puede ser significantemente más limitado en términos de memoria, potencia de procesado, y en tantos otros aspectos que un dispositivo de tipo PDA tal como un iPAQ. Debido a las limitaciones del teléfono móvil, no sería práctico o factible el uso de RMI como parte de una aplicación. Además, para reducir el peso en memoria de las librerías de J2ME, no es deseable requerir que las características de RMI estén presentes en la instalación de J2ME del teléfono móvil. Otros dispositivos más potentes si serían capaces de manejar las demandas de RMI, por lo que la aplicación debería elegir el uso de estas características. De este modo hay unas necesidades de acomodar de algún modo esta situación. Por razones como esta, J2ME ha particionado la funcionalidad de J2ME en varios grupos que permiten a los diferentes dispositivos requerir y soportar diferentes características de Java para ser aplicadas a

cada dispositivo. Esto ayuda a optimizar el uso de Java para dispositivos específicos, sin restringir las capacidades

Mantenimiento de una Arquitectura Común

Una arquitectura común a través de diferentes dispositivos provee una base que hace a las aplicaciones ser más fácilmente portables, sino directamente implementables, entre los dispositivos. Soportando múltiples arquitecturas de dispositivos específicos haría la portabilidad de aplicaciones, incluso entre los dispositivos más similares, difícil y costosa de soportar.

Una única y flexible arquitectura es más efectiva respecto al coste de mantenimiento que múltiples arquitecturas de propósito específico para uno o varios dispositivos. Una arquitectura común toma ventaja de las similitudes entre dispositivos y permite una reutilización de las aplicaciones. En el caso en el que diferentes dispositivos requieren distintas capacidades para se soportados, la arquitectura común puede ser extendida o configurada para proporcionar esa funcionalidad específica.

Con el propósito de ser mantenible y portable, J2ME necesita ser extensible. Ya que nuevos dispositivos entran en el mercado constantemente, J2ME necesita ser capaz de adaptarse rápidamente y dar soporte a esos dispositivos.

Uno de los problemas claves que la arquitectura de J2ME tienta es solventar como soportar un amplio rango de dispositivos con diferentes limitaciones, capacidades, características, y usos sin introducir limitaciones en ningún dispositivo específico. Una solución podría ser crear una extensa arquitectura que incluya todo, cualquier aplicación que pueda necesitarse para cualquier dispositivo dado. De cualquier modo, una arquitectura de este tipo sería demasiado extensa en términos de memoria demandada para los pequeños dispositivos que J2ME pretende soportar, tales como un teléfono móvil.

Otra solución podría ser identificar un denominador común de funcionamiento que está presente en todos los dispositivos en el espacio de J2ME. El problema con este alcance es que los dispositivos más potentes son limitados por los más pequeños. Además, las características únicas de los dispositivos no pueden ser soportadas adecuadamente. J2ME define un alcance común que direcciona como soportar muchos dispositivos sin limitar sus características establecidas.

Configuraciones y Perfiles

J2ME introduce dos conceptos arquitectónicos: configuraciones y perfiles. Las configuraciones establecen el set de APIs de bajo nivel que definen las características de ejecución de un entorno particular de J2ME. Específicamente, las configuraciones son las responsables de definir:

- Core Java Clases
- Características del lenguaje d programación Java
- Características de la máquina virtual

Los perfiles definen las APIs para dispositivos y usos específicos tales como interfaces de usuario, mecanismos de almacenamiento de datos y otras características específicas de los dispositivos tales como el uso de puertos IR (Infra-red) para intercambio de información entre PDAs o acceder a características telefónicas de un teléfono móvil.

Las configuraciones y los perfiles proveen un separación en el entorno de la arquitectura J2ME entre la necesidad de portabilidad y la necesidad de soportar un

amplio rango de dispositivos y capacidades. Las configuraciones sirven para incrementar la portabilidad a través de diferentes dispositivos mientras lso perfiles suministran las características de un dispositivo específico o un grupo de dispositivos similares. Por ejemplo, las configuraciones incluyen las características del núcleo de Java tales como String, System, Thread, y Object, así como flujos de entrada salida y conectividad de red. Un perfil suministra características de un dispositivo tales como interfaz gráfica de usuario, manejador de eventos, y almacenamiento de datos. Los perfiles también proveen el modo de empaquetar sets específicos de funcionalidad como son capacidades multimedia o características de vídeo juego.

Una característica importante de las configuraciones es que comparte una relación anidada. Esto significa que una configuración puede ser mayor o menor, pero todas ellas se incluyen dentro de la más extensa de las configuraciones de J2ME. Esta relación debe ser siempre relaciones de superset – subset. Este concepto es ilustrado en la siguiente figura.

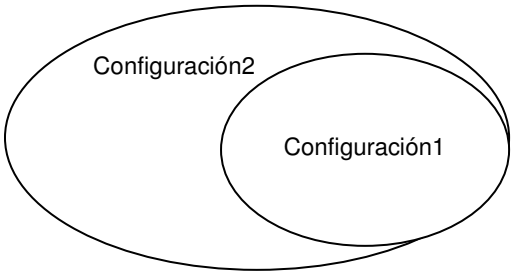


Figura 7. Configuraciones anidadas

La relación anidada de las configuraciones permite una gran portabilidad. La portabilidad es maximizada cuando el movimiento es desde el entorno más limitado al más extenso y rico en características. Por ejemplo, si una aplicación fue desarrollada para un teléfono móvil usando la Configuración 1, sería deseable hacer que la aplicación estuviera disponible para dispositivos que corran la Configuración 2 también.

Los fabricantes de dispositivos deben adherir la especificación de configuración cuando implementan o portan máquinas virtuales Java en sus plataformas para ser conformes a J2ME. Esta conformidad permite por portabilidad cruzar diferentes fabricantes de dispositivos así como entre diferentes tipos de dispositivos. Esto es un ejemplo de las capacidades WORA (Write Once, Run Anywhere) de la arquitectura de J2ME. Por ejemplo, una aplicación J2ME para un teléfono móvil puede ser desplegada sobre cualquier teléfono móvil conforme a J2ME con pequeñas o ningunas modificaciones. Intentar esto con las librerías C o incluso APIs de Java significaría portar la aplicación a cada casa fabricante de teléfonos móviles, ya que cada teléfono móvil tiene un sistema operativo distinto. En pocas palabras, las capacidades WORA de J2ME empiezan con el lenguaje de programación Java y son realizadas a través de la arquitectura J2ME.

Visión Global de J2ME

Un entorno completo de J2ME esta compuesto de una configuración y uno o más perfiles. Como estos dos conceptos arquitectónicos pueden ser mezclados para cubrir una necesidad específica, la arquitectura de J2ME se hace más amoldable para poder soportar diversas necesidades dentro del espacio J2ME.

J2ME despliega diferentes versiones de la máquina virtual Java basadas en las necesidades de una situación particular. Las especificaciones de configuración definen

las características de las máquinas virtuales Java. En la mayoría de casos, esto envuelve la eliminación de características de la máquina Virtual de Java con el fin de acomodarse a las necesidades de la configuración. La eliminación de características generalmente se hace para reducir el tamaño de la máquina virtual.

La máquina virtual es el componente que se posiciona en el nivel lógico superior al sistema operativo. La configuración y las APIs de los perfiles acceden a las APIS del sistema operativo a través de la máquina virtual. Esta pila de componentes puede verse gráficamente en la siguiente figura.

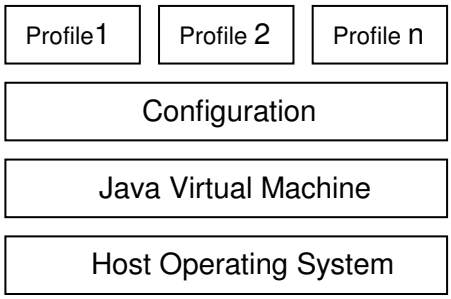


Figura 8. Bloques que edifican J2ME

Paquetes Opcionales

La plataforma J2ME puede ser extendida añadiendo una variedad de paquetes opcionales a la pila de tecnologías que incluyen CLDC o CDC y sus perfiles asociados. Creados para albergar requisitos de aplicación muy específicos, los paquetes opcionales ofrecen APIs estándar para usar ambas, existentes y emergentes tecnologías tales como conectividad de bases de datos, mensajes Wireless, multimedia, *Bluetooth* y servicios web. Debido a que los paquetes opcionales son modulares, los programadores pueden evitar asumir los gastos indirectos de la funcionalidad innecesaria incluyendo solamente los paquetes que una aplicación necesita realmente.

4.5.4.2 Configuraciones

Las configuraciones comprenden una maquina virtual y un mínimo numero de librerías de clases. Ellas proveen la funcionalidad base para un rango particular de dispositivos que comparten similares características, tales como conectividad de red y memoria. Actualmente, hay dos configuraciones J2ME:

- CONNECTED LIMITED DEVICE CONFIGURATION (CLDC)
- CONNECTED DEVICE CONFIGURATION (CDC)

CLDC acapara las necesidades de los dispositivos con estrictas limitaciones de memoria, potencia de procesamiento, consumo de potencia, y conectividad de red. CDC acapara las necesidades de los dispositivos más potentes, dentro de J2ME.

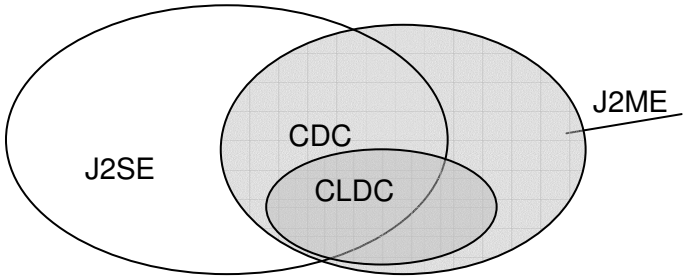


Figura 9. Jerarquía de configuraciones

Debido a las jerarquizadas relaciones entras las configuraciones, la portabilidad puede ser enormemente realzada cuando se traslada de un entorno más limitado a otro con características más ricas. Es importante de todas formas, recalcar que J2ME define algunas APIs no presentes en J2SE.

Las configuraciones definen un contrato entre un perfil y la máquina virtual de Java. Ambas configuraciones, CDC y CLDC, tiene sus propias implementaciones de maquinas virtuales. CDC usa la C-Virtual Machine (CVM) y CLDC usa la que es denominada como Kilobyte Virtual Machine (KVM).

Connected Limited Device Configuration (CLDC)

Connected Limited Device Configuration (CLDC) define el sistema base de interfaces de programación de aplicaciones y una máquina virtual para dispositivos con recursos limitados como los teléfonos móviles. Unida a un perfil tal como Mobile Information Device Profile (MIDP), proporciona una plataforma sólida de Java para las aplicaciones que corren sobre dispositivos con recursos limitados de memoria, energía de proceso, y capacidades gráficas.

Dos máquinas virtuales están disponibles para soportar CLDC. La K Virtual Machine es la máquina virtual más pequeña para CLDC, y soporta dispositivos de usuario con una memoria disponible para la pila de tecnologías Java de entre 128K a 512K. La implementación CLDC HotSpot agrega compilación adaptable para un aumento dramático en funcionamiento, con un aumento mínimo en las necesidades de memoria.

Una máquina virtual de Java es el origen para la tecnología Java, permitiendo a las aplicaciones escritas en el lenguaje de programación Java ser portadas a otros entornos hardware diferentes y otros sistemas operativos. La máquina virtual media entre la aplicación y el sistema operativo, convirtiendo el bytecode de la aplicación en código de nivel máquina apropiado para el hardware y el sistema operativo que están siendo usados. Además de gobernar la ejecución del bytecode de las aplicaciones, la máquina virtual lleva a cabo tareas tales como el control del sistema de memoria, aporta seguridad ante código malicioso, y maneja múltiples hilos de ejecución del programa.

La K Virtual Machine

Para resolver la necesidad del mercado de una implementación de Java con un uso de la memoria muy pequeño, KVM fue diseñada para superar tres desafíos técnicos dominantes: reducir el tamaño de la máquina virtual y de las librerías de clases, reducir la memoria utilizada por la máquina virtual durante la ejecución, y permitir a los

componentes de la máquina virtual ser configurados para satisfacer a dispositivos particulares.

El equipo de diseño de KVM empleó una serie de estrategias en la superación de estos desafíos técnicos. Una de estas estrategias era permitir repartir las capacidades de la máquina virtual. El equipo también reconstruyó el intérprete de bytecode y el recolector de basura para reducir el uso dinámico de la memoria, e implemento cuidadosamente la máquina virtual y las librerías para reducir al mínimo su tamaño.

Los resultados del cuidado diseño e implementación son visibles:

- ***Reducido tamaño de la VM*** - la K máquina virtual tiene actualmente un tamaño de solo 50-80 K de código objeto en la configuración estándar, dependiendo del dispositivo en el que se instale y de las opciones de compilación.
- ***Utilización de memoria reducida*** - Además del pequeño tamaño del código objeto de la máquina virtual, requiere solo unas pocas decenas de kilobytes de memoria dinámica para funcionar de manera efectiva. Debido al reducido tamaño de la VM y la utilización de memoria, incluso con una memoria total disponible de solo 128K la K Virtual Machine permite la ejecución de aplicaciones basadas en tecnología Java sobre el dispositivo.
- ***Funcionamiento*** - la K *Virtual Machine* puede ejecutarse correctamente sobre procesadores de 16 bits con una frecuencia de trabajo tan baja como 25 MHz, y pueden ser escalados a procesadores más potentes de 32.
- ***Portabilidad*** - la K *Virtual Machine* tiene una arquitectura altamente portable que reduce las dependencias del sistema al mínimo. Incluso *multi-threads* y recolector de basura han sido implementados en un sistema completamente independiente, permitiendo una rápida adaptación a cualquier plataforma receptora.

CLDC HotSpot Implementation

Esta implementación logra un fuerte balance entre un buen funcionamiento y las limitaciones impuestas por los dispositivos. La arquitectura de la máquina virtual CLDC HotSpot Implementation incluye las siguientes características:

- Compilador dinámico y adaptable.
- Intérprete optimizado
- Soporte para threads de poco peso, con gran portabilidad
- Capa de objetos compacta
- Fuentes unificadas de gerencia
- Exacta generación del colector de basura
- Rápida sincronización
- No restricciones respecto al número de clases cargadas
- ROMizer, el cual almacena clases del sistema en un formato compacto que permite una ejecución más rápida
- Soporte para CLDC 1.0 o CLDC 1.1
- Soporte para tecnologías hardware de aceleración
- Capacidad de direccionamiento de 32 bits para soportar un amplio rango de dispositivos
- Soporte de 16bits Thum
- Puede construirse como un programa principal o como una subrutina en un evento.

Especificaciones

- JSR-000139 Connected Limited Device Configuration 1.1

- JSR-000030 J2ME Connected Limited Device Configuration 1.0

CLDC Paquetes Opcionales

- Mobile Media API: La API Mobile Media (JSR 135) especifica una pequeña API multimedia para los dispositivos que disponen de Java, desde simples teléfonos móviles hasta más sofisticados dispositivos multimedia. La API permite un simple acceso y control de audio y vídeo en tiempo real. Es extensible y escalable para soportar características multimedia mas sofisticadas.

- Wireless Messaging API

- Bluetooth API

Perfiles Asociados a CLDC

- MIDP: Mobile Information Device Profile

  Este es el primer perfil oficial realizado por Sun y alberga teléfonos móviles y pagers. Este perfil ha sido también implementado para correr sobre el sistema operativo Palm (Palm OS), haciéndolo disponible para dispositivos que funcionen con este sistema operativo. Los dispositivos que implementan este perfil tienden a estar muy personalizados. Estos dispositivos suelen tener recursos myu limitados, como una pantalla de interfaz de usuario muy reducida, capacidades de entrada de datos muy limitadas, y escasas capacidades de almacenamiento de datos normalmente implementados sobre arrays de bytes.

Connected Device Configuration (CDC)

Marco de trabajo base para la construcción y desarrollo de aplicaciones móviles. Está diseñado para escenarios de productos con recursos limitados, típicamente 2MB de RAM y 2,5Mb de ROM para el entorno de aplicación de Java. CDC es compatible con la API J2SE. La familia CDC incluye los perfiles:

- *Foundation Profile*
- *Personal Basis Profile y*
- *Personal Profile.*

Ventejas

Las empresas pueden:

- Añadir movilidad sin sacrificar seguridad
- Integración con aplicaciones de empresa a bajo nivel
- Usar tecnología Java punto a punto, desde clientes móviles a servidor
- Confiar en el robusto modelo de seguridad Java
- Soportar un amplio rango de dispositivos desde una plataforma integrada

Los usuarios pueden:

- Consistencia y previsión de la experiencia
- Disfrutar de una gran seguridad para la información sensible

Los vendedores de dispositivos pueden:

- Confiar en una plataforma de aplicaciones móviles segura
- Ofertar software que enganche con las aplicaciones de la empresa

Los programadores pueden:

- Dibujar un rico set de APIs a todos los niveles
- Disfrutar la seguridad y productividad del lenguaje de programación Java
- Cargar múltiples dispositivos desde una única plataforma
- Aplicar sus conocimientos de J2SE
- Reutilizar código desarrollado para aplicaciones basadas en J2SE
- Trabajar con herramientas compatibles con el estándar J2SE

Los proveedores de servicios pueden:

- Suministrar ofertas escalables a las empresas
- Desplegar aplicaciones móviles que son seguras

Dispositivos Adecuados

La configuración CDC fue diseñada para traer las ventajas de la plataforma Java a un amplio rango de dispositivos de consumidor conectados a red y empotrados, incluyendo comunicadores inteligentes, PDAs de gama alta, y set-top boxes.

Los dispositivos que soportan CDC típicamente incluyen un microprocesador/controlador de 32 bits y tienen entre 2MB de RAM y 2,5 MB de ROM disponible para en entorno de aplicación Java.

Perfiles

Un perfil específico es combinado con la configuración CDC para proveer un completo entorno Java de aplicación para la clase de dispositivos a los que se dirige. CDC soporta tres perfiles.

Foundation Profile	Personal Basis Profile	Personal Profile
▪ Librerías de clases J2SE-basico ▪ No soporte GUI ▪ CLDC 1.0 librerías compatibles	▪ Soporte de componentes de poco peso ▪ Soporte Xlet ▪ Foundation Profile APIs	▪ Soporte AWT completo ▪ Soporte de Applet ▪ Ruta de migración para la tecnología PersonalJava ▪ Personal Basis Profile APIs, incluyenddo Foundation Profile APIs

Figura 10. Perfiles asociados a la configuración CDC

Paquetes Opcionales

Un paquete opcional es una serie de APIs de una tecnología específica qye extiende la funcionalidad de el entorno de aplicaciones Java. CDC soporta una serie de paquetes opcionales que permiten a los diseñadores de productos hacer un balance de las necesidades de la funcionalidad de un diseño, frente a las limitaciones de recursos.

- El RMI Optional Package provee un subconjunto de J2SE RMI . Expone protocolos de aplicación distribuidos a través de Interfaces Java, clases,

invocaciones de métodos, y aísla al programador de los detalles de comunicación de la red.
- El JDBC Optional Package provee un subconjunto de JDBC 3.0 API que da a las aplicaciones basadas en Java acceso a bases de datos

Especificaciones

- CDC 1.0.1 (JSR 36): define las bases de los dispositivos J2ME que tienen un micoprocesador de 32 bits y amplia memoria.
- CDC 1.1 en progreso (JSR 218) define una revisión de la especificación de J2ME CDC. Esta JSR provee actualizaciones (basadas en J2SE, v1.4) del núcleo existente, Java APIs no gráficas para dispositivos electrónicos pequeños.

4.5.4.3 Perfiles

Para proporcionar un ambiente runtime completo para una categoría específica de dispositivos, una configuración se debe combinar con un perfil. Un perfil es una serie de APIs de alto nivel que define más concretamente el modelo del ciclo vital de la aplicación, la interfaz de usuario, y el acceso a características específicas del dispositivo.

Un perfil soporta una categoría más estrecha de dispositivos en el marco de una configuración elegida. Un ejemplo extensamente adoptado es combinar

CLDC con Mobile Information Device Profile (MIDP) para proporcionar un ambiente completo para aplicaciones Java para los teléfonos móviles y otros dispositivos de capacidades similares.

- **FP: Foundation Profile:**

Foundation Profile es una serie de APIs de Java que dan soporte a dispositivos con recurosos limitados sin un sistema estándar GUI. Combinado con *Connected Device Configuration(CDC),* Foundation Profile ofrece un entorno de aplicación J2ME completo para productos de consumo y dispositivos empotrados.

Foundation Profile es el más básico de la familia de perfiles CDC. Tiene las siguientes características:

- Basado en APIs J2SE privadas.
- Adaptado para ambientes con recursos limitados.
- No ofrece soporte GUI

Ejemplos de escenarios de uso para Foundation Profile:

- Impresoras de red
- Routers
- Residential Gateways
- Enterprise-Class Server Applications

Descripción de la API

Combinado con Connected Device Configuration (CDC), Foundation Profile ofrece un entorno de aplicación J2ME completo. CDC y Foundation Profile toman APIs de J2SE

y adaptan sus implementaciones para cubrir las necesidades de los dispositivos sobre los que se quieren aplicar estos estándares.

La mayoría de las APIs fundamentales son idénticas entre CDC/Foundation Profile y J2SE 1.3.1. Las principales diferencias entre CDC/Foundation Profile APIs y J2SE 1.3.1 son listadas a continuación:

Tomadas de J2SE 1.3.1	java.io java.lang java.lang.ref java.lang.reflect java.net java.security java.text java.util java.util.jar
Tomadas de J2SE 1.3.1	java.io java.lang java.lang.ref java.lang.reflect java.net java.security java.text java.util java.util.jar
Disponibles de manera separada en forma de paquetes opcionales	java.rm java.sql
No heredadas de J2SE 1.3.1	javax.accessibility javax.naming.* javax.rmi.* javax.sound.* javax.swing.* javax.transaction.* java.org.omg.*
Nuevas en CDC/FP	javax.microedition.io

Las diferencias restantes están en los paquetes de java.awt y java.beans. La compatibilidad de AWT y la ayuda de bean son el foco principal de Personal Basis Profile y Personal Profile

- **PBP: Personal Basis Profile**

J2ME Personal Basis Profile es una serie de APIs de Java que dan soporte a dispositivos con recursos limitados, con un marco de trabajo que dispone de

componentes estándar GUI. Combinado con Connected Device Configuration (CDC), J2ME Personal Basis Profile ofrece un entorno de aplicación J2ME completo para productos de consumo y dispositivos empotrados.

J2ME Personal Basis Profile tiene las siguientes principales características:

- Un marco de trabajo GUI para construcciones gráficas, con un conjunto de herramientas ligero, componentes limitados.
- Suporte para el modelo de programación de aplicaciones xlet
- Todas las APIs para soporte de aplicaciones incluidas en Foundation Profile

J2ME Personal Basis Profile es conveniente para escenarios de diseño de productos que requieren un interfaz gráfico de usuario (GUI) sin compatibilidad completa con AWT. Ejemplos de escenarios de aplicación para J2ME Personal Basis Profile incluyen:

- Televisión interactiva
- Automoción
- Dispositivos de consumo de propósito específico (Ej. camcordes)

Descripción de la API

Combinado con CDC, J2ME Personal Basis Profile ofrece un entorno de aplicación Java para dispositivos de recursos limitados que necesitan un conjunto de herramientas ligero con componentes gráficos. CDC y J2ME Personal Basis Profile toman APIs de J2SE y adaptan sus implementaciones a las necesidades de los dispositivos con recursos limitados.

J2ME Personal Basis Profile incluye todas las APIs de Foundation Profile. Las principales diferencias entre J2ME Personal Basis Profile APIs y J2SE 1.3.1 son listadas a continuación:

- El paquete `java.awt` incluido en J2ME Personal Basis Profile no incluye componentes GUI pesados como `java.awt.Button` o `java.awt.Panel`. Una única instancia de `java.awt.Frame` es permitida como contenedor de componentes ligeros.
- `javax.microedition.xlet` y `javax.microedition.xlet.ixc` proveen soporte para los modelos de programación de aplicación de xlet.

Debido a que J2ME Personal Basis Profile da soporte completo para componentes ligeros, es fácil construir y extender cajas de herramientas con las herramientas del estándar J2SE. Estos juegos de herramientas pueden ser estándar como MHP.

- **PP Personal Profile**

J2ME Personal Profile es una serie de APIs Java que dan soporte a dispositivos de recursos limitados con un juego de herramientas GUI basado en AWT. Combinado con Connected Device Configuration (CDC), J2ME Personal Profile ofrece un entorno de aplicación J2ME completo para productos de consumo y dispositivos empotrados.

J2ME Personal Profile tiene las siguientes principales características:

- Completa compatibilidad AWT
- Soporte para aplicaciones modelo applet
- Una trayectoria para la migración de tecnologías como PersonalJava
- Todas las APIs incluidas en J2ME Personal Basis Profile

J2ME Personal Profile es conveniente para escenarios de diseño que requieran completa compatibilidad AWT y soporte applet.

Ejemplos de escenarios de aplicación para J2ME Personal Profile incluyen:

- High-end PDAs
- Embedded Web browsers

Descripción de la API

En combinación con CDC, J2ME Personal Profile provee un entorno de aplicación Java para dispositivos de recursos limitados que requieren compatibilidad completa AWT y soporte applet. CDC y J2ME Personal Profile toman APIs de J2SE y adaptan sus implementaciones para satisfacer las necesidades de los dispositivos con recursos limitados.

La implementación AWT en J2ME Personal Profile está basada en una combinación de `java.awt` de JDK 1.1 y algunos paquetes 2D de J2SE 1.3.1. Algunos paquetes y clases son subconjuntos de sus correspondientes en J2SE.

- **The Mobile Information Device Profile (MIDP):**

Mobile Information Device Profile (MIDP) es elemento clave de Java 2 Platform, Mobile Edition (J2ME). En combinación con Connected Limited Device Configuration (CLDC), MIDP proporciona un entorno runtime estándar de Java para los dispositivos de información móviles más populares de hoy día, tales como teléfonos y PDAs.

La especificación MIDP fue definida a través de Java Community Process (JCP) por un grupo de expertos de más de 50 compañías, incluyendo fabricantes de dispositivos, portadores Wireless, y vendedores de software para móviles. Define una plataforma para un despliegue optimizado, dinámico y seguro, de aplicaciones gráficas, y aplicaciones de red de trabajo.

CLDC y MIDP proporcionan la funcionalidad base requerida para aplicaciones móviles, en forma de un entorno runtime Java estandarizado y una rica lista de APIs Java. Usando MIDP, los programadores pueden escribir aplicaciones una vez, y desplegarlas rápidamente a una amplia variedad de dispositivos de información móviles. MIDP se ha adoptado extensamente como la plataforma para aplicaciones móviles. Esta desplegada globalmente sobre millones de teléfonos y PDAs, y está soportada por entornos integrados de desarrollo (IDEs). Compañías de todo el mundo ya han aprovechado las ventajas de MIDP para escribir un amplio rango de aplicaciones móviles para usuarios o empresas.

Especificaciones

- **MIDP 2.0** (JSR 118) es una versión revisada de a especificación MIDP 1.0. Las nuevas características incluyen una interfaz de usuario realzado, multimedia, con funcionalidad para juegos, más extensiones de conectividad, aprovisionamiento vía aérea (OTA), y seguridad punto a punto. MIDP 2.0 es compatible con MIDP

1.0, y continua dando soporte para dispositivos móviles de información tales como teléfonos móviles y PDAs.

- **MIDP 1.0** (JSR 37) es la especificación original la cuál proporciona la funcionalidad del uso básica requerida para aplicaciones móviles, incluyendo interfaz de usuario y seguridad básica de red.

Beneficios

- Ricas capacidades de interfaz de usuario
- Conectividad extensa.
- Funcionalidad de multimedia y juegos.
- Aprovisionamiento sobre el aire
- Seguridad punto a punto

4.5.5 Write One, Run Anywhere

WORA (Write one, run anywhere) no es automático cuando se implementan aplicaciones Java en dispositivos. En gran parte, es por esto por lo que J2ME existe. La arquitectura J2ME lleva al lenguaje de programación Java a realzar la portabilidad entre un amplio rango de dispositivos con distintas capacidades y necesidades.

Es importante destacar que el lenguaje Java en si no se ve modificado, solo las características de la máquina virtual como describen las configuraciones CDC y CLDC. Las ediciones compatibles entran en juego normalmente cuando diferentes capacidades de dispositivo deben repartirse. Capacidades de dispositivo diferentes son manejadas a través de diferentes perfiles en J2ME. Como resultado, dispositivos que soportan perfiles diferentes pueden incurrir en problemas de compatibilidad.

Variedad de Necesidades de Dispositivo

Las necesidades de los dispositivos varias a través de las categorías de dispositivos, por lo que no es real soportar todas las características en todos los dispositivos. Este es en especial el caso de los dispositivos pequeños que tiene limitaciones en memoria, capacidad de procesado, consumo de potencia, conectividad de red, y almacenamiento de datos. Fabricantes y desarrolladores de teléfonos móviles, por ejemplo, son forzados frecuentemente a tomar difíciles decisiones acerca de que es esencial para desarrollar aplicaciones en estos dispositivos. Soportar todas las características de J2ME en todos los dispositivos requeriría mucha memoria, potencia de procesado que eliminaría automáticamente muchos dispositivos enmarcados en J2ME.

Por ejemplo, algunso dispositivos, como PDAs, tiene interfaces de pantalla táctil mientras otros dispositivos, como los pagers no los tienen. Los pagers, por otro lado, normalmente soportan un teclado alfanumérico completo mientras que los teléfonos móviles tienen sencillos teclados para ser usados por una sola mano. Internet TV set-top boxes soportan un alto ancho de banda y conexiones de red de alta fidelidad, mientras lso dispositivos Wireless tienen mayores limitaciones de ancho de banda. En grado en que WORA puede ser alcanzado es en gran parte debido a las capacidades de especificación de perfiles a elegir para soportar una aplicación concreta.

Estas diferencias en capacidades pueden prevenir una aplicación migrando paulatinamente sobre diferentes tipos de dispositivos donde los mismos perfiles no están soportados. Por lo tanto, cierto diseño previo es requerido si una aplicación debe de correr en dos perfiles diferentes.

La Arquitectura J2ME y WORA

La arquitectura J2ME no rompe con WORA negligentemente. Diseñada para balancear ediciones compatibles entre dispositivos y las necesidades especiales para cada tipo de dispositivo. Por esta razón, es importante entender la arquitectura de J2ME cuando se crean aplicaciones para dispositivos J2ME. Entendiendo como los perfiles y las configuraciones se relacionan se aumentan las opciones de crear aplicaciones quesean compatible para un rango de muchos y distintos dispositivos.

4.5.6 Entorno de Ejecución

Hay dos formas básicas de correr aplicaciones J2ME sobre dispositivos. Una forma es correrlas de manera transitoria sobre la red. En este modo de operación la aplicación es cargada en memoria tras ser descargada desde la red. Una vez que la aplicación se ha ejecutado, es descartada. Ejecutar aplicaciones de este modo requiere conectividad a la red. Las aplicaciones pueden ser también instaladas en el dispositivo. En este caso, la aplicación está disponible para ser ejecutada haya o no conexión a red.

Independientemente del método usado para correr las aplicaciones J2ME, un controlador específico de cada dispositivo es envuelto en la parte de implementación J2ME que corre en el dispositivo actual. La parte del entorno de J2ME responsable de controlar las aplicaciones en el dispositivo se denomina JAM (Java Application Manager). La implementación de JAM es hecha por el fabricante del dispositivo. El JAM por si mismo maneja actividades como la descarga, instalación, inspección, lanzamiento y desinstalación de aplicaciones Java en el dispositivo.

Muchos dispositivos del espacio J2ME serán suministrados con el entorno Java ya preparado en el dispositivo. Es bueno que los usuarios de los dispositivos, y en último lugar la aplicación, no tengan que participar en la carga del entorno de ejecución Java (JRE). De todos modos, esto también significa que solo hay un JRE disponible en el dispositivo. Partiendo de que diferentes fabricantes lanzarán inevitablemente versiones de sus productos en diferentes fechas con diferentes JREs, el código necesita ser compatible sobre cierto número de distintos entornos de ejecución J2ME dependiendo de cómo el dispositivo maneja esa situación.

4.5.7 Sumario

La tecnología J2ME representa un esfuerzo por condensar y reducir el estándar Java. La plataforma de *Java 2 Micro Edition (J2ME)* provee un robusto y flexible entorno para aplicaciones que corren sobre dispositivos de usuario, tales como teléfonos móviles, PDAs, y TV set-top boxes, también un gran rango de dispositivos empotrados. Al igual que los entornos para la empresa (J2EE), escritorio (J2SE) y tarjetas inteligentes (Java Card), J2ME incluye una Maquina Virtual de Java y una serie de estándar Java APIs definidos a través del Java Community Process, por un grupo de expertos entre cuyos miembros se encuentran fabricantes de dispositivos, vendedores de software, y abastecedores de servicios.

J2ME entrega la energía y las ventajas de la tecnología de Java al consumidor y a los dispositivos empotrados. Esto incluye interfaces de usuario, un robusto de modelo de seguridad, una amplia gama de los protocolos de red incorporados, y extensiones de ayuda para aplicaciones de red de trabajo y aplicaciones fuera de línea que se pueden descargar dinámicamente. Las aplicaciones basadas en especificaciones de J2ME se escriben una vez para una amplia gama de dispositivos, explotando las capacidades nativas de cada dispositivo.

La plataforma de J2ME se despliega en millones de dispositivos, y usado por compañías en todo el mundo. Resumiendo, se trata de la plataforma elegida hoy día para dispositivos empotrados y de usuario.

La arquitectura J2ME define configuraciones para abarcar las necesidades horizontales del espacio J2ME. Las *J2ME Virtual Machines* son firmemente acopladas a las configuraciones que las definen. Los perfiles rellenan los agujeros dejados por las configuraciones e implementan capacidades específicas de una familia de dispositivos. Los perfiles tienen a abarcar los aspectos verticales del espacio J2ME, tales como las específicas capacidades y limitaciones de los dispositivos, o encapsulan una serie de APIs que albergan mercados específicos, o necesidades tecnológicas. Por compatibilidad entre dispositivos, los fabricantes deben implementar completamente las especificaciones de los perfiles. Esto permite a las aplicaciones conformes a un perfil, correr sobre cualquier dispositivo que implemente el perfil. Finalmente, es importante entender que un dispositivo puede soportar uno o más perfiles sobre una configuración.

Capítulo 5
Requisitos y Diseño de la Aplicación

5.1 Introducción

En los capítulos anteriores se han expuesto entre otros:

- El contexto en el que se encuentra la aplicación que debemos de desarrollar. Requisitos funcionales del sistema EFTCoR, relaciones entre los subsistemas constituyentes, y que es lo que se demanda del subsistema formado por la PDA y más concretamente de la aplicación que sobre ella debe implantarse.
- Objetivos que se deben alcanzar con la implementación de la aplicación destinada a ejecutarse sobre el terminal tipo PDA.
- Infraestructura utilizada. Componentes hardware necesarios para constituir el subsistema tele-operado a manejar por el operador humano. Cada uno de los componentes hardware se ha descrito con detalle, resaltando las partes más influyentes sobre la aplicación software. Como se verá más adelante, en ocasiones los dispositivos utilizados condicionan el código a la hora de escribir la aplicación; por ello resulta importante conocer las características, posibilidades y limitaciones, del dispositivo sobre el cual se ejecutará una determinada aplicación. Como es evidente, también se ha expuesto el software propio de cada uno de los dispositivos, así como el uso de otras aplicaciones adicionales que vayan a ser utilizadas.
- Elección de la plataforma software y lenguaje de programación con el que implementar la aplicación que va a ser desarrollada. Se ha desarrollado una amplia visión de la plataforma Java mostrando las diferencias entre sus distintos estándares, y justificando la elección de uno de ellos en función de la plataforma sobre la que vaya a ejecutarse el software que va a ser diseñado.

Una vez estudiados todos estos aspectos, se está en disposición de especificar y diseñar la aplicación. Es importante una buena descripción del sistema que se va a implementar antes de pasar a la escritura del código. El diseño debe de ser robusto y cubrir todas las características del sistema que se va a desarrollar. Un buen diseño de aplicación asegura un ahorro en tiempo y una mejor resolución en la fase de implementación del código.

Basándose en la notación UML, se definen dos formas de capturar los límites del domino.

Casos de Uso: capturan los requisitos funcionales y las razones por las que el sistema está siendo construido. Identifican el valor que distintas características proveen al sistema.

Contratos: articulan los servicios (operaciones) provistas por el sistema (aplicación software) para alcanzar los valores deseados.

Ambos son escritos normalmente en términos de negocios, más que tecnológicos.

Para captar los límites del dominio y los requisitos del sistema se ha hecho uso de la notación *Use Case*. Los requisitos funcionales definen lo que el sistema hará para el usuario. Utilizando esta notación, los requisitos funcionales se definen en términos de actores y casos de uso. Un *actor* participa en un caso de uso. Un *caso de uso* define una secuencia de interacciones entre uno o más actores y el sistema.

Antes de comenzar la especificación de los requisitos propios de la aplicación, vamos a introducir las características de la notación que se aplicará. Se han identificado la terminología usada y los distintos tipos de descripción que se hacen del sistema, diagramas gráficos y descripción textual.

5.2 Los Casos de Uso de UML

5.2.1 Introducción a los Casos de Uso

Veamos la terminología usada para la especificación de casos de uso:

- *Goal: E*l valor de negocio para los usuarios del sistema que normalmente inician la interacción con el mismo.
- *System:* Es la aplicación, con todas las asociaciones hardware que serán usadas por los usuarios.
- *Actor: E*s una entidad externa que interactúa con el sistema.
- *Use Case:* Es una descripción de una interacción que alcanza un uso objetivo para un actor.
- *Use Case bundle:* Es una colección de casos de uso que son altamente correlativos con alguna actividad o elemento organizativo de negocio. Representan una forma de organizar los casos de uso en colecciones que ayudarán a un mejor entendimiento de la funcionalidad del sistema que se esté desarrollando.

Con la notación de casos de uso, el sistema es visto como una caja negra. Sólo se consideran las características externas, entradas y salidas del sistema.

SISTEMA: Históricamente, los sistemas han sido entendidos utilizando tres modelos básicos: cajas negras, cajas blancas, y cajas transparentes. Cada modelo sucesivo, provee un detalle incremental en la implementación y la descripción interna del sistema. El modelo de cajas negras muestra los valores y funcionalidad que provee el sistema al exterior. No indica el modo en que la información es procesada dentro del sistema, únicamente muestra las entradas y salidas del sistema. El modelo de cajas blancas presenta el sistema en términos de las funciones específicas que el sistema ofrece. El modelo de cajas transparentes presenta el interior del sistema y como este compone la funcionalidad del mismo.

En la construcción de casos de uso, se supone una vista del sistema desde el punto de vista de las cajas negras. Pero la experiencia ha puesto en evidencia la necesidad en ocasiones de adoptar el punto de vista de las cajas blancas para determinar la secuencia de interacciones entre un actor y el sistema.

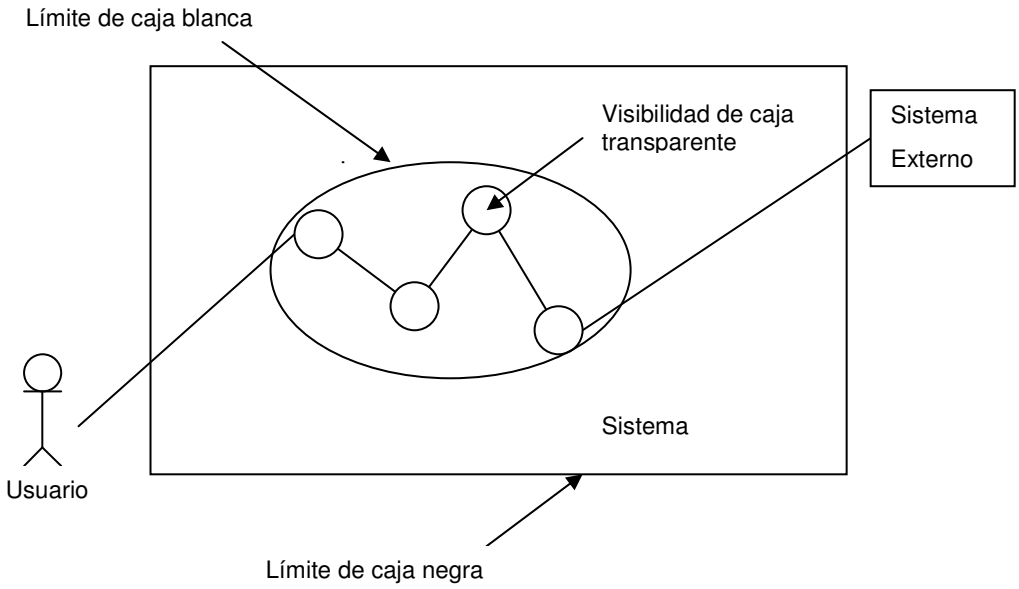


Figura 1. Visibilidad del sistema conforme a los modelos de caja negra, caja blanca y caja transparente

Actores: El modelo de casos de uso divide el mundo en dos partes: el sistema y los usuarios (las entidades externas que lo usan). Los actores son un mecanismo para categorizar los usuarios del sistema, quienes comparten una serie de interacciones comunes para alcanzar un objetivo o una serie de objetivos. Los usuarios suelen ser entidades físicas. Un actor puede ser un usuario, un sistema externo o un dispositivo. Un actor puede hacer un servicio de peticiones al sistema, ser requerido para proveer un servicio, e interactuar con el sistema. De este modo, un actor es una representación de cualquier entidad que pueda iniciar una acción en una parte del sistema o recibir una petición desde el sistema. En un sentido real, el conjunto completo de peticiones/respuestas de todos los actores establecen los límites del dominio del sistema. Un sistema nunca puede responder a aspectos de un dominio para el cual no ha sido diseñado.

Caso de Uso: Un *escenario* es una pequeña historia que acota alguna secuencia esperada de peticiones y respuestas entre usuarios y el sistema. Escribir un escenario es tan sencillo como predecir qué es lo que va a ocurrir a continuación. La mayoría de los escenarios son simples; sólo hay una secuencia lógica de operaciones desde el estado inicial. Otros escenarios son más complicados, con múltiples casos de excepción (cosas que pueden ir mal) o diferentes rutas de interacción (opciones).

La Figura 2 ilustra un diagrama de Caso de Uso simple para un sistema de registro de cursos.

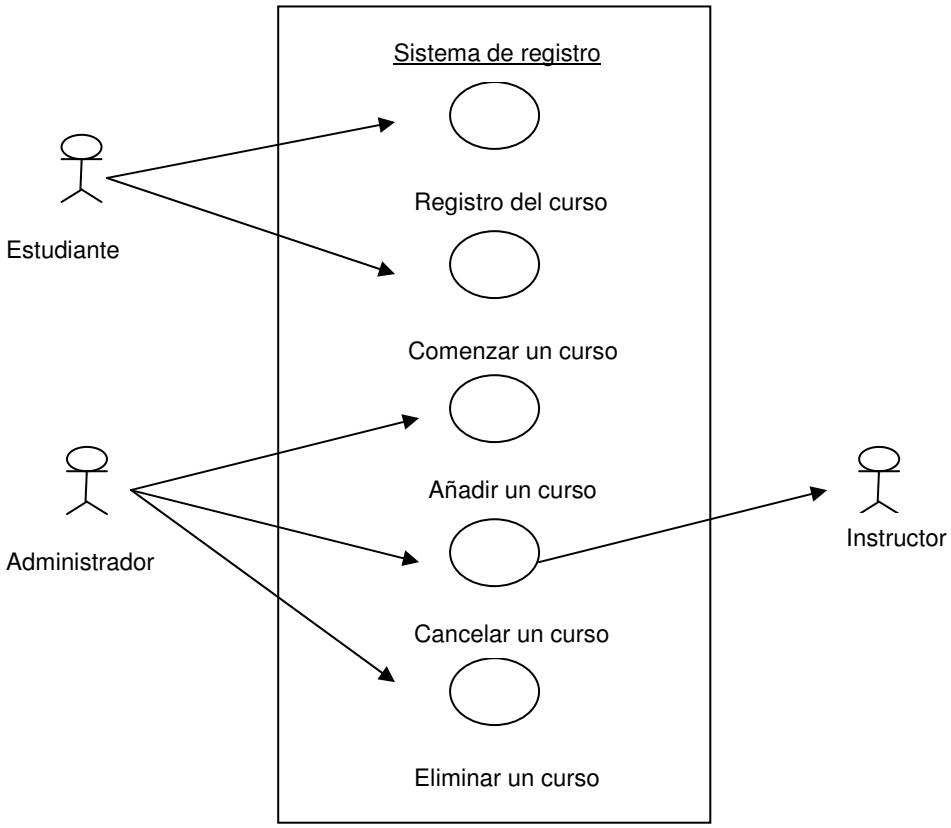


Figura 2. Diagrama de Caso de Uso simple

Un usuario específico es una instancia de un actor. Por ejemplo, Jorge es un usuario. Jorge es una instancia del actor 'usuario'.

Un caso de uso en la forma generalizada de una familia de escenarios. De este modo un escenario es una instancia específica de un caso de uso.

Un caso de uso describe el sistema en términos de secuencias de interacciones entre varios actores del sistema (ejemplo, un caso de uso específico captura todos los escenarios que empiezan con la misma petición para alcanzar el mismo objetivo de usuario). También, es un camino formal para describir todas las interacciones entre los actores y el sistema para alcanzar los objetivos funcionales. El actor que inicia el caso de uso es denominado actor iniciador. En la mayoría de las interacciones, el dialogo resulta de la interacción del sistema con otros actores; estos actores son denominados actores participantes. La interacción asume que el sistema es una caja negra y usa elementos del dominio como actores que interactúan con el sistema en modo solicitud/respuesta. Un caso de uso identifica las precondiciones que deben existir para ser válido, las post-condiciones que definen el estado del sistema una vez que el caso de uso ha finalizado, detalles de la acción que son realizados (no dependientes de la tecnología), excepciones que deben lanzarse, y limitaciones del sistema ante peticiones específicas de un actor.

Al capturar los aspectos funcionales del sistema, es importante mantener la descripción en un nivel de abstracción consistente. Para que los casos de uso sean desarrollados con éxito, es necesario conocer la dimensión de la descripción funcional que se pretende capturar. Entonces se puede determinar el nivel de detalle en la información que debería ser capturada.

En una primera dimensión, podemos distinguir entre descripciones funcionales de un sistema a alto y bajo nivel. Un alto nivel de descripción provee una descripción general

y breve de la esencia de los valores provistos del asunto. No se preocupa de cómo estos valores son alcanzados. Una descripción de bajo nivel, provee detalles del asunto tratado mostrando el orden exacto de actividades, tareas, o alternativas.

En una segunda dimensión, podemos distinguir entre funciones primarias y secundarias del sistema. Las funciones primarias son las funciones esenciales que va a desempeñar el sistema, las funciones ofrecidas a los usuarios que constituyen las razones por las que el sistema existe. Los procesos secundarios tratan los casos raros y excepcionales. Son estas las funciones necesarias para entregar un sistema robusto.

Finalmente en una tercera dimensión, podemos distinguir entre funciones esenciales y concretas de un sistema. Los casos de uso esenciales son soluciones que son independientes de la implementación (hardware o software), mientras que los casos de uso concretos son dependientes del diseño. La distinción entre esencial y concreto es la distinción entre los modelos de caja negra y caja transparente.

Relaciones a través de casos de uso. Un caso de uso típico traza una secuencia de interacciones entre el iniciador de un servicio de peticiones al sistema y el sistema. Con frecuencia ocurre que algunas secuencias de interacciones son comunes entre múltiples casos de uso. En esta situación, se pueden extraer estas secuencias comunes como un caso de uso independiente. Posteriormente, se incluye este caso de uso formando parte de los casos de uso desde los cuales fue extraído. Es cómo una subrutina común que es usada por muchas rutinas. Cuando esta situación existe, se dice que cada uno de los 'múltiples' casos de uso incluye al caso de uso común. Las relaciones incluidas permiten localizar en un caso de uso una secuencia común de actividades a lo largo de distintos casos de uso. Esto tiene la ventaja de que cuando un cambio tiene lugar en esa secuencia común, solo es necesario cambiarla en un sitio.

Además, puede darse la situación en que distintos casos de uso sean idénticos con la excepción de una o dos subsecuencias específicas de interacciones. En este caso podemos extraer un núcleo común (caso de uso base) y tratar los casos de uso que difieren como extensiones del caso de uso base. Serían como subclases en las cuales solo está permitido añadir código sobre la rutina padre. De este modo existe una relación de extensión entre el caso de uso extendido y el núcleo.

Finalmente, en el desarrollo de casos de uso a alto nivel, es frecuente el caso en que se abarcan diferentes detalles y casos de uso extendidos. La relación que existe entre el caso de uso a alto nivel y el detallado y el caso de uso extendido es una relación de generalización / especialización.

Sujeto de descripción. Como un estado previo, un caso de uso puede ser clasificado en tres dimensiones diferentes:

1. Primario – secundario
2. Esencial – concreto
3. Alto nivel – bajo nivel

Una descripción usando casos de uso de un sistema enfatiza uno de cada dos pares de factores. Distintas agrupaciones son usadas comúnmente para alcanzar el nivel de detalle capturado en casos de uso:

- Primario, esencial, alto nivel
- Primario y secundario, esencial, bajo nivel
- Primario, concreto, bajo nivel
- Primario y secundario, concreto, bajo nivel

Generalmente una completa y detallada descripción funcional de un sistema produce toda una variedad de casos de uso empezando desde arriba y trabajando hasta los niveles más bajos.

Información capturada. Una de las razones para emplear el alcance de los casos de uso para delimitar el sistema es la facilidad con la que la información puede ser definida. Los casos de uso pueden capturar los siguientes tipos de información:

- Actores. Un caso de uso identificará todos los actores que participan en él.

- Relaciones con otros casos de uso. Una descripción de caso de uso identificará las relaciones (generalización/especialización, inclusión, y relaciones de extensión) que los casos de uso tienen entre ellos

- Precondiciones. Un caso de uso debe requerir condiciones específicas a mantener con el fin de ser invocado con éxito. Todas estas condiciones deben ser identificadas. En algunos casos, los sistemas son concebidos para exhibir modalidades de comportamiento. De ellos se espera que operen en diferentes modos mostrando distintos comportamientos para cada modo diferente. Una precondición identificará el modo requerido y cualquier otra condición que debe de cumplirse para que el caso de uso sea válido.

- Detalles. Un caso de uso describe los detalles acerca de como un sistema provee algunos servicios. Los detalles de un caso de uso identifican los detalles de las secuencias de interacciones. Los detalles son capturados como interacciones paso a paso a lo largo de los objetos del dominio. Cada paso provee suficiente detalle para identificar que entidades están envueltas, que hace cada entidad, y el resultado de dicho paso. Esto puede llevarse a cabo de manera textual o mediante diagramas de secuencia.

- Post-condiciones. La ejecución de un caso de uso pretende alcanzar algún estado o cómputo deseado. Las post- condiciones identifican exactamente que resultados se esperan de la ejecución de ese caso de uso. Esto incluye cualquier efecto colateral, tales como objetos creados o destruidos. Se recomiendan las siguientes especificaciones:
    - Instancias creadas o destruidas
    - Relaciones (asociación y agregación) formadas o rotas
    - Valores cambiados en las variables
    - Estados cambiados (incluyendo el estado final)

- Excepciones. Cada acción llevada a cabo en un caso de uso es susceptible de error. Los datos deseados no son localizados, el cómputo ha sido abortado, o la conectividad se ha perdido, son algunos de los errores que pueden ser capturados mediante excepciones. Es necesario identificar todos los posibles errores que puedan ocurrir en el caso de uso. Por lo tanto, para cada excepción debemos saber en que circunstancias pueden ocurrir y que medidas se deben tomar.

- Limitaciones. También es necesario identificar las limitaciones que pueden requerirse a un caso de uso. Son normalmente las condiciones invariantes, condiciones que deben ser siempre ciertas. Las condiciones invariantes deben de cumplirse al principio (precondición) del servicio (operación) y al final (post-condición). La

violación de estas limitaciones pueden producir errores, y tales errores deben ser identificados como excepciones.

- Variantes/Alternativas. Es también necesario identificar las variaciones que pueden ocurrir en un caso de uso. Se trata normalmente de las variaciones que no son cubiertas por casos de uso independientes. Normalmente estas variaciones son fáciles de manejar o son consideradas una parte de otro caso de uso.

Por supuesto, la información capturada para un caso de uso específico variará en función de una serie de factores. En particular variará si se trata de una especificación a alto o bajo nivel, y esencial o concreto. La tabla 1 contiene la información que es desarrollada típicamente para tipos comunes de casos de uso que son desarrollados.

Por ejemplo, un desarrollo de alto nivel, primario y esencial para un caso de uso requiere esta identificación:

- La información esencial del sistema
- Las precondiciones que deben de cumplirse para que el caso de uso pueda ser completado
- Las post-condiciones que se esperan, y
- Cualquier limitación o variación que pueda existir

5.2.2 Documentando Casos de Uso

Un aspecto importante de los casos de uso es documentarlos de forma que se facilite la comprensión del sistema. Ya que una imagen suele plasmar la información contenida en muchas palabras, un camino muy útil para documentar casos de uso es usar la notación UML (Unified Modeling Language).

5.2.2.1 Diagrama de Casos de Uso

Un diagrama de casos de uso muestra cómo los casos de uso se relacionan entre ellos y con los actores. Un ejemplo de caso de uso es ilustrado en la figura 3. Un actor es representado por una figura humana, incluso para situaciones de sistemas externos. Los casos de uso individuales son dibujados como óvalos etiquetados con el nombre del caso de uso. Los actores se conectan mediante líneas a los casos de uso que inicializan. También mediante líneas se conectan los casos de uso con los actores participantes, desde los cuales el sistema realiza peticiones. Si la interacción es unidireccional, la línea termina con una flecha. La dirección va desde el solicitante hasta el proveedor del servicio. En el ejemplo, el director del banco hace una petición al sistema, el cuál hace una petición a la base de datos de cuentas.

En algunas situaciones, la interacción es bidireccional y el enlace es ilustrado como una línea con una flecha en ambos lados. En este caso, el actor puede generar una petición al sistema, o éste puede realizar una petición al actor para que lleve a cabo alguna acción. Este podría ser el caso en el que el manager quiere monitorizar algunas cuentas específicas para acciones concretas y el sistema genera alarmas solicitando atención al manager cuando ocurran eventos específicos. El sistema debe de realizar peticiones periódicas sobre la base de datos para identificar cuando ocurren estos eventos.

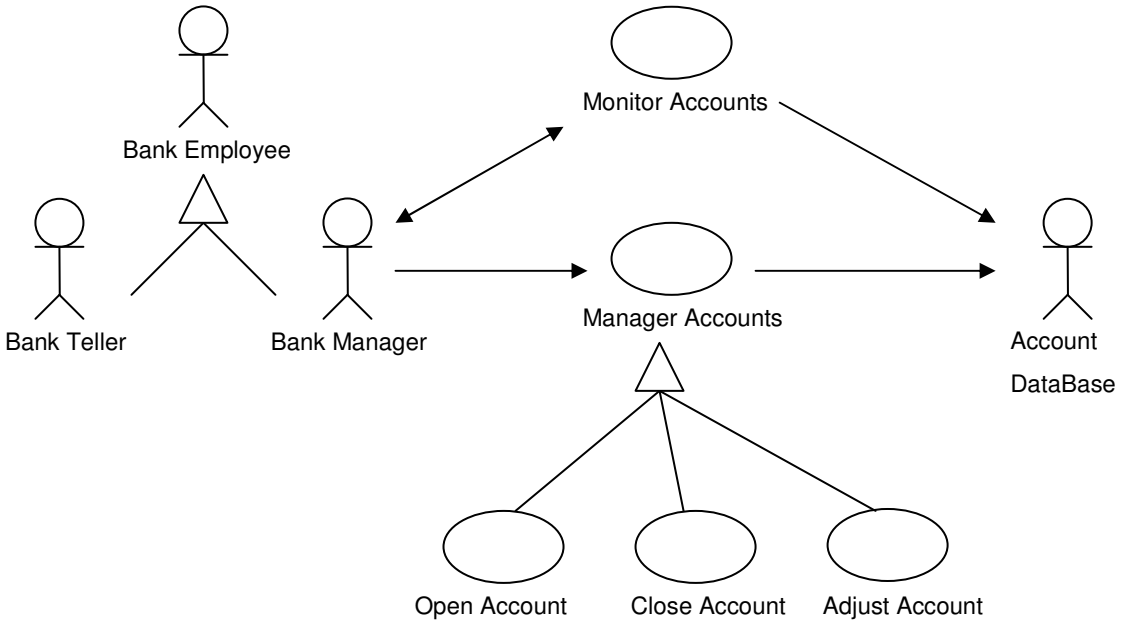


Figura 3. Un ejemplo de diagrama de casos de uso ilustrando relaciones de generalización / especificación entre actores, e interacciones bidireccionales entre casos de uso y actores.

Las generalizaciones/especializaciones en casos de uso y actores que fueron anteriormente identificados también son documentados en los diagramas de casos de uso. La relación es ilustrada con un triángulo al final de la línea. La línea es originada en la especialización y apunta con el triángulo a la generalización. En el ejemplo, el actor *bank employeed* es una generalización de los actores especializados *bank teller* y *bank manager*. La idea es que algunos casos de uso (ver el balance de una cuenta, cambiar la dirección de un cliente, etc) pueden ser iniciados por ambos, un *bank manager* o un *bank teller*. UML dispone de un mecanismo para ilustrar casos de uso. Por ejemplo, la figura 4 muestra una notación para relaciones de extensión e inclusión.

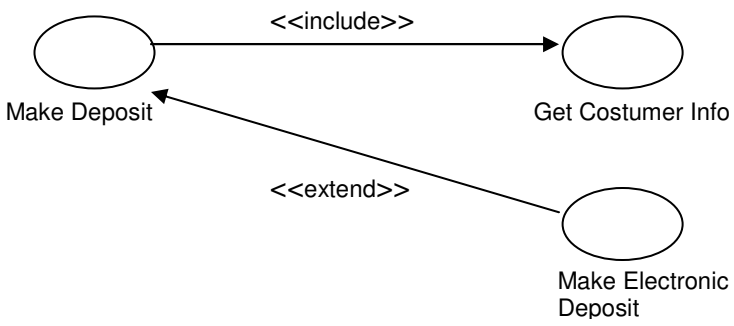


Figura 4. Representación con UML de relaciones de extensión e inclusión entre casos de uso

Una flecha es dibujada entre el caso de uso sostenido y el caso de uso que lo soporta, con la flecha etiquetada por el tipo de asociación que está siendo representada. La dirección de la flecha indica la dirección de la relación. Para casos de uso que extienden a otros, la flecha nace del caso de uso extendido. En relaciones de inclusión, la flecha apunta hacia el caso de uso incluido. En la figura 4, el caso de uso *Make Deposit* incluye el caso de uso *Get Costumer Info*, mientras que el caso de uso *Make Electronic Deposit* extiende al caso de uso *Make deposit*, añadiéndole funcionalidad.

5.2.2.2 Descripción Textual

Además de la representación gráfica de casos de uso, es de práctica habitual usar descripciones textuales de cada caso de uso individual. Una descripción textual contiene la información identificada en la sección anterior. Una plantilla típica es la que se muestra a continuación:

Nombre del caso de uso: debe de ser único.

Descripción: una o dos frases que describan el caso de uso

Sumario: identifica los casos de uso incluidos en él.

Extensiones: identifica los casos de uso que de él se extienden

Actores: identifica los actores que participan en el caso de uso.

Precondiciones: identifica las condiciones que deben de ser ciertas para que el caso de uso pueda ser invocado.

Secuencia: identifica los detalles del caso de uso.

Alternativas: narrativa de las alternativas a la secuencia principal.

Post-condiciones: condiciones ciertas tras terminar la ejecución del caso de uso.

Excepciones: identifica cualquier excepción que deba lanzarse durante la ejecución del caso de uso

Limitaciones: Identifica cualquier limitación que deba de aplicarse

Comentarios: Suministra información adicional que pueda ser importante para el caso de uso.

El valor clave de la descripción textual es que se captura más información que en la representación gráfica. Como resultado, lo más habitual es usar una combinación de diagramas de caso de uso para proporcionar una vista global del sistema, diagramas de secuencia para capturar las interacciones, y una descripción textual para capturar las precondiciones, post-condiciones, excepciones, condiciones invariantes, y variaciones.

5.3 Requisitos de una Aplicación de Control para el Sistema EFTCoR

5.3.1 Introducción

Una vez presentada la notación que se ha empleado para plasmar los requisitos del sistema, se prosigue con la especificación de la aplicación a desarrollar.

El operador usará el terminal tipo PDA para operar el sistema robótico. En su fase final, la plataforma de tele-operación, deberá ser capaz de controlar y coordinar el robot de limpieza para optimizar los estándares de calidad con el fin de minimizar recursos y tiempo de operación. El terminal de tele-operación muestra el estado del robot al

operador, el cual puede de manera remota llevar a cabo las tareas de chorreo. Así mismo se proveen una serie de comandos y se permite el movimiento del robot cuando el modo de operación es seleccionado.

Estas son las funciones a desempeñar por la PDA, y por tanto las tareas que debe de realizar la aplicación que se ejecute sobre la misma:

- Proporcionar un interfaz amigable y de uso sencillo a través del cual el operador humano pueda realizar las tareas que siguen.
- Comunicación con el sistema de monitorización para intercambio de notificaciones entre ambos subsistemas, y consultas del operador sobre tareas a realizar. Se contempla la posibilidad de que cada una de las notificaciones enviadas se almacene de manera local en un fichero a modo de historial.
- Control de tareas de limpieza para las cuales se requiere:
    - Monitorización de información de estado, más concretamente de los sensores que capturan la distancia entre el casco del buque y la mesa del robot.
    - Control de posicionamiento de la mesa para realizar tareas de chorreo. Para ello se envían una serie de comandos al sistema de control que indican avance o retroceso para cada una de las posiciones. Debe de permitirse que el sistema de posicionamiento pare.
    - Captura de vídeo en tiempo real procedente de la cámara, para un mejor control de las tareas de posicionamiento y limpieza.
- Se articulará un mecanismo de desconexión del sistema.

La figura 6 representa las entradas y salidas del sistema.

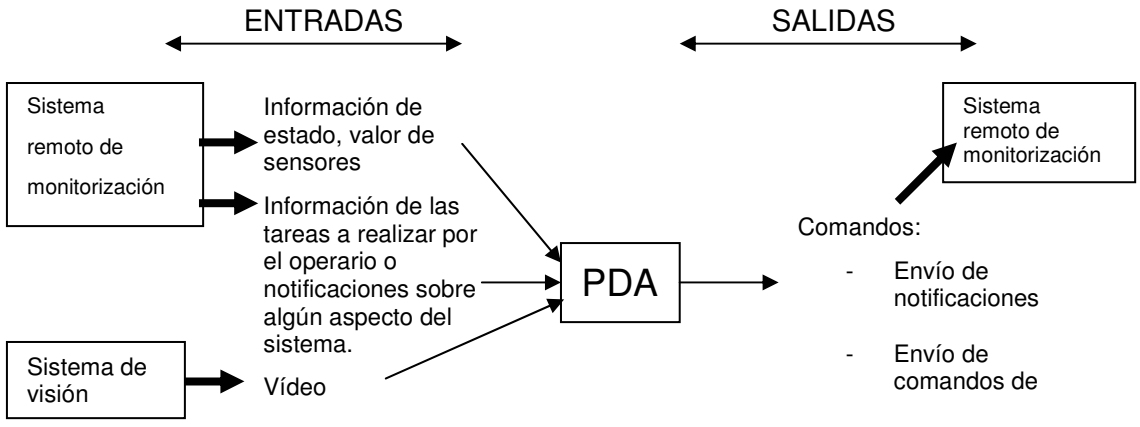


Figura 5. Entradas y salida del subsistema formado por el dispositivo tipo PDA

Tres son los tipos de información que le llegan a la PDA desde el exterior. Desde el sistema de monitorización remoto, la aplicación que se ejecuta sobre la PDA recibe información sobre el estado de los sensores de la mesa robótica, además de información

sobre las tareas a realizar por el operador. El sistema de visión suministra vídeo en tiempo real.

Por su parte, el subsistema formado por la PDA envía al sistema de monitorización dos tipos de comandos: notificaciones sobre las tareas que se están realizando (tarea finalizada, incidencia/error,…), y órdenes de direccionamiento para mover la plataforma robótica ('x' avanza, 'y' retrocede,…).

5.3.2 Diagrama de Casos de Uso del Sistema Desarrollado

Recordemos que los casos de uso de UML:

- Describen el comportamiento del sistema desde el punto de vista del usuario
- Establecen los límites del sistema y definen las relaciones entre éste y el entorno.
- Los casos de uso son descripciones de la funcionalidad, independientemente de la implementación.
- Están basados en un lenguaje natural, lo que facilita la comprensión por parte de los usuarios.

En la primera fase de diseño de nuestro sistema, adoptamos el punto de vista del modelo de caja negra. Anteriormente se han identificado las entradas y salidas que tendrá el sistema. Con esta información es posible comenzar a aplicar la notación de captura de requisitos de casos de uso. En primer lugar se define el diagrama de casos de uso. Este diagrama representa la funcionalidad del sistema. Se deben identificar los actores y casos de usos que constituyen el sistema, y las relaciones existentes entre ambos. Recordemos que un actor puede ser un usuario, un sistema externo o un dispositivo. Un actor puede hacer ambos, realizar peticiones al sistema u ofrecer un servicio al mismo. De este modo, un actor es una representación de cualquier entidad que pueda iniciar una acción en una parte del sistema o recibir una petición desde éste. Un caso de uso es un camino formal para describir todas las interacciones entre los actores y el sistema para alcanzar los objetivos funcionales de la aplicación.

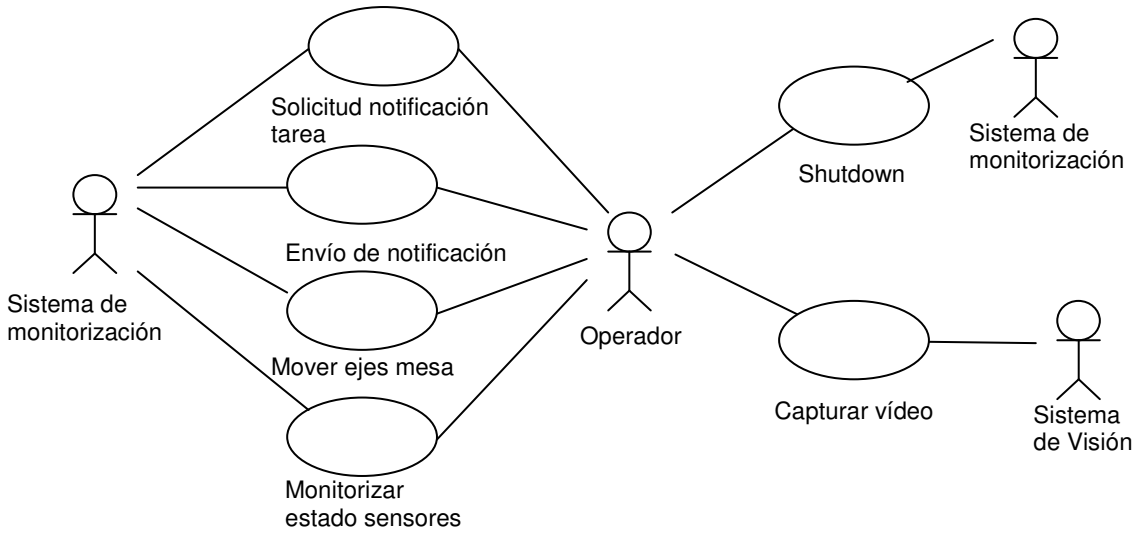


Figura 6. Diagrama de casos de uso del sistema a implementar.

Observando este diagrama se deducen las siguientes acciones. El actor Operador intercambia de manera directa información con el actor Sistema de monitorización de varias maneras. El actor Operador puede realizar peticiones al sistema de monitorización para que éste le informe de las tareas a realizar. El Operador envía notificaciones al sistema de monitorización informándole de las incidencias ocurridas. De igual modo el Operador inicia los casos de uso Mover ejes mesa y Monitorizar estado sensores. Mediante el caso de uso Mover ejes mesa, el sistema realiza la tarea de enviar las órdenes de movimiento a la plataforma robótica. Mediante el caso de uso Monitorizar estado sensores el operador recibe la información del valor de los sensores de la mesa robótica. Esta función junto con la de captura de vídeo ayuda al operador a enviar los comandos de posicionamiento más adecuados Continuando con el actor Operador, este se encarga de iniciar el caso de uso Shutdown, que produce la desconexión del sistema. El caso de uso Capturar vídeo es iniciado por el Operador y consiste en la captura de vídeo suministrado por el Sistema de Visión.

5.3.3 Descripción Textual

- Caso de Uso: Shutdown

  Actor: El operador, sistema de monitorización

  Sumario: El operador termina de operar el sistema y lo desconecta

  Precondición: el sistema está parado

  Secuencia:
  1. El operador pulsa el botón de shutdown
  2. Se envía notificación al sistema de monitorización informando de que las comunicaciones serán abortadas
  3. El sistema aborta las comunicaciones con el resto de subsistemas.

  Comentario: Este caso de uso debe ser ejecutado al final de la jornada de trabajo.

- Caso de Uso: Envío de notificación

  Actores: Operador, Sistema de monitorización

  Sumario: el operador pulsa el botón de envío de notificaciones seleccionando una de las que el menú ofrece, y envía datos al sistema remoto de monitorización

  Precondición: el sistema de limpieza está parado, y debe de estar seleccionado el submenú 'Notificar' en el menú principal.

  Secuencia:

  1. El operador pulsa el botón de Notificaciones
  2. El sistema muestra un menú donde el operador puede elegir la opción deseada
  3. El sistema envía los datos correspondientes a esa opción

  Alternativas: No hay enlace de comunicación con el sistema de monitorización. El error debe de ser mostrado al operador.

- Caso de Uso: Solicitud notificación tarea

  Actores: Operador, Sistema de monitorización

Sumario: El operador selecciona, desde el menú principal, recibir información desde el sistema externo de monitorización acerca de la tarea que debe desempeñar. La información se muestra en la PDA.

Precondición: el sistema de limpieza está parado, y debe de estar seleccionado el submenú 'Notificar' en el menú principal.

Secuencia:

1. El operador selecciona la opción de solicitud de información
2. El sistema de monitorización envía la información
3. El texto recibido se muestra en la pantalla de la PDA

Alternativas: No hay enlace de comunicación con el sistema de monitorización. El error debe de ser mostrado al operador.

- Caso de Uso: Monitorizar estado sensores

  Actor: Operador, sistema de monitorización

  Sumario: el valor de los sensores es mostrado por pantalla.

  Precondición: el sistema debe haberse iniciado previamente, y debe de estar seleccionado el submenú 'Posicionar' en el menú principal.

  Secuencia:

  1. El operador selecciona la opción de posicionamiento
  2. La aplicación lee el valor de los sensores situados en la mesa de limpieza del robot.
  3. Se muestran gráficamente los valores al operario a través de la pantalla de la PDA.

- Caso de Uso: Mover ejes mesa

  Actor: Operador, sistema de monitorización

  Sumario: el operador envía mediante comandos los movimientos que debe de realizar la plataforma robótica.

  Precondición: el sistema debe haberse iniciado previamente, y debe de estar seleccionado el submenú 'Posicionar' en el menú principal.

  Secuencia:

  1. El operador selecciona la opción de posicionamiento.
  2. El operador envía comandos, indicando los ejes a mover y la dirección.

- Caso de Uso : Capturar vídeo

  Actor: Operador, Sistema de visión

  Sumario: se visualiza sobre la pantalla de la PDA el vídeo captado por la cámara del sistema de visión

  Precondición: el sistema debe haberse iniciado previamente.

Comentario: debe de poderse recibir información de los sensores y enviar comandos de posicionamiento al mismo tiempo que se visualiza el vídeo.

5.4 Diseño de una Aplicación de Control para el Sistema EFTCoR

5.4.1 Introducción

Para el diseño de la aplicación se ha empleado la notación UML=Unified Modeling Language.

Es un lenguaje gráfico de propósito general, orientado a objetos y que permite visualizar, especificar, construir y documentar sistemas de información. Esta notación se basa en la construcción de modelos.

- Cada modelo describe completamente el sistema, pero desde un único punto de vista.
- Modelos Casos de Uso: desde el punto de vista del usuario
- Modelo estático: elementos del sistema y sus relaciones
- Modelo dinámico: comportamiento del sistema en el tiempo

Cada modelo se representa gráficamente mediante uno o más diagramas. UML ofrece dos clases de diagramas:

- Diagramas estructurales:
  - Diagramas de Clases
  - Diagramas de Objetos
  - Diagramas de Componentes
  - Diagramas de Distribución

- Diagramas de comportamiento:
  - Diagramas de Casos de Uso
  - Diagramas de Secuencia
  - Diagramas de Colaboración
  - Diagramas de Estados
  - Diagramas de Actividad

5.4.2 Diagrama de Clases y Paquetes

5.4.2.1 Diagrama de Clases

Esencial tanto en fase de análisis como de diseño. Permite representar las clases del sistema así como las relaciones estructurales y de herencia existentes entre ellas.

La definición de cada clase incluye información sobre sus atributos y operaciones.

El modelo de casos de uso aporta la información necesaria para establecer y definir las clases y objetos del sistema

En UML cada clase se representa con un rectángulo dividido en tres compartimientos en los que se especifica:

- Nombre de la clase
- Atributos de la clase
- Métodos de la clase

A continuación se muestra el diagrama de clases de la aplicación que será desarrollada.

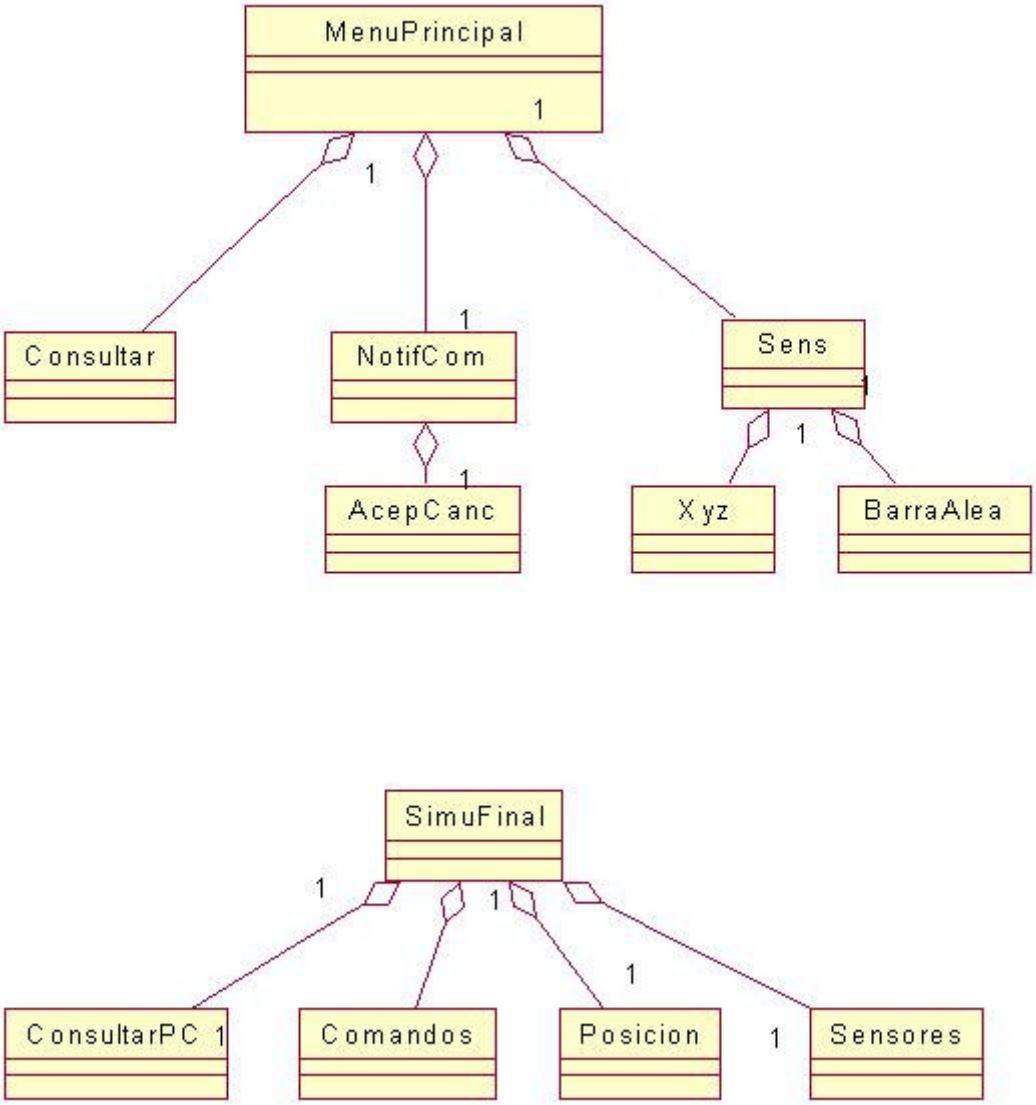
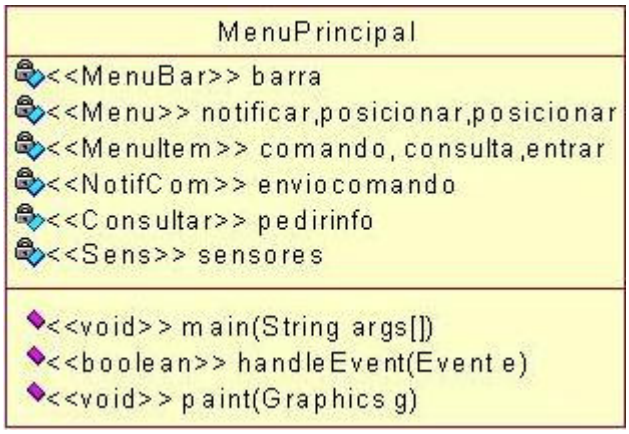
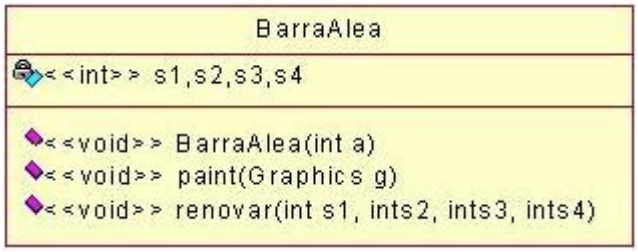
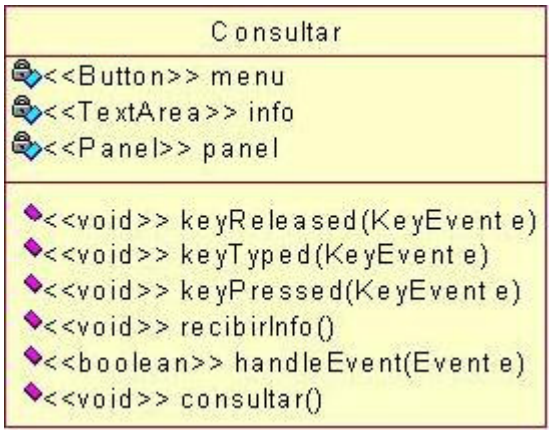


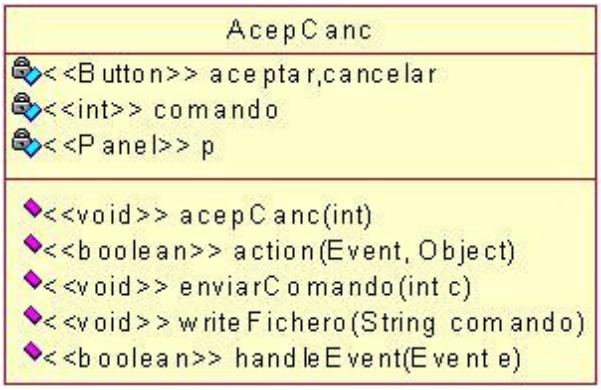
Figura 7. Diagrama de clases de la aplicación a desarrollar

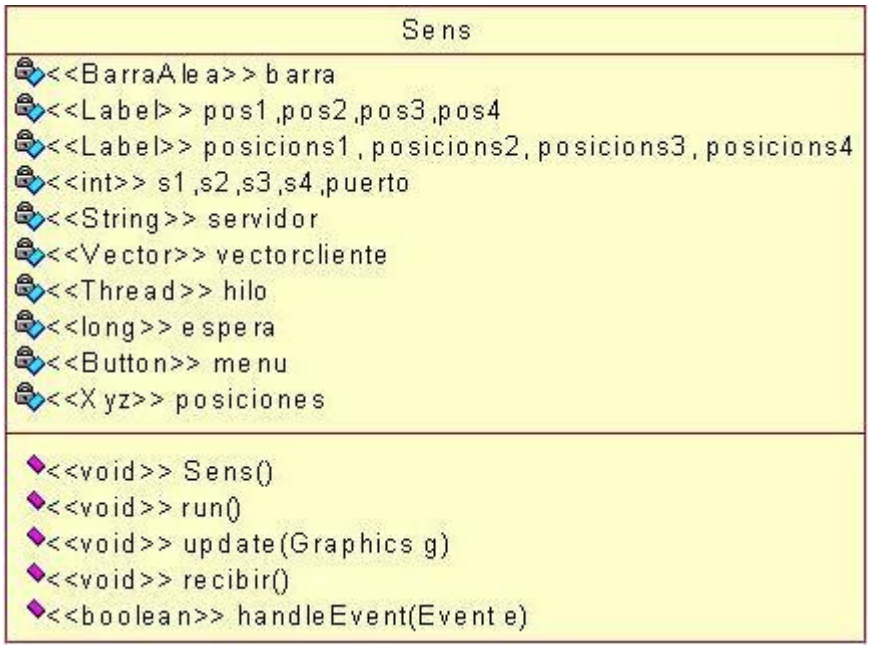

MenuPrincipal
<<MenuBar>> barra
<<Menu>> notificar,posicionar,posicionar
<<MenuItem>> comando, consulta,entrar
<<NotifCom>> enviocomando
<<Consultar>> pedirinfo
<<Sens>> sensores
<<void>> main(String args[])
<<boolean>> handleEvent(Event e)
<<void>> paint(Graphics g)


BarraAlea
<<int>> s1,s2,s3,s4
<<void>> BarraAlea(int a)
<<void>> paint(Graphics g)
<<void>> renovar(int s1, ints2, ints3, ints4)


Consultar
<<Button>> menu
<<TextArea>> info
<<Panel>> panel
<<void>> keyReleased(KeyEvent e)
<<void>> keyTyped(KeyEvent e)
<<void>> keyPressed(KeyEvent e)
<<void>> recibirInfo()
<<boolean>> handleEvent(Event e)
<<void>> consultar()


NotifCom
<<Button>> not1,not2,not3,not4,menu
<<Panel>> p1,p2,p3,p4,p5
<<float>> CENTER_ALIGNMENT
<<void>> NotifCom()
<<boolean>> handleEvent(Event e)





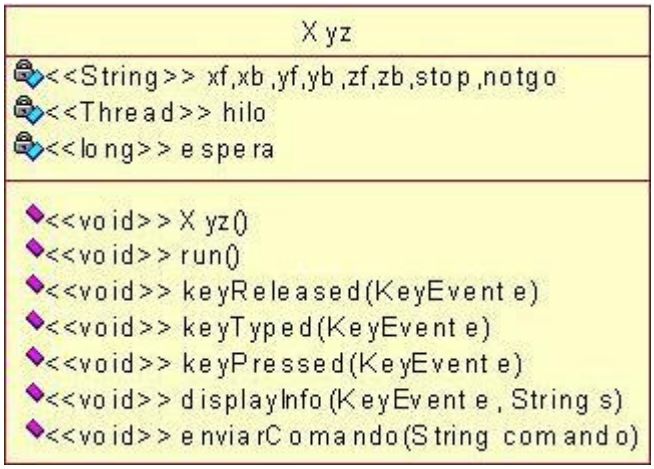
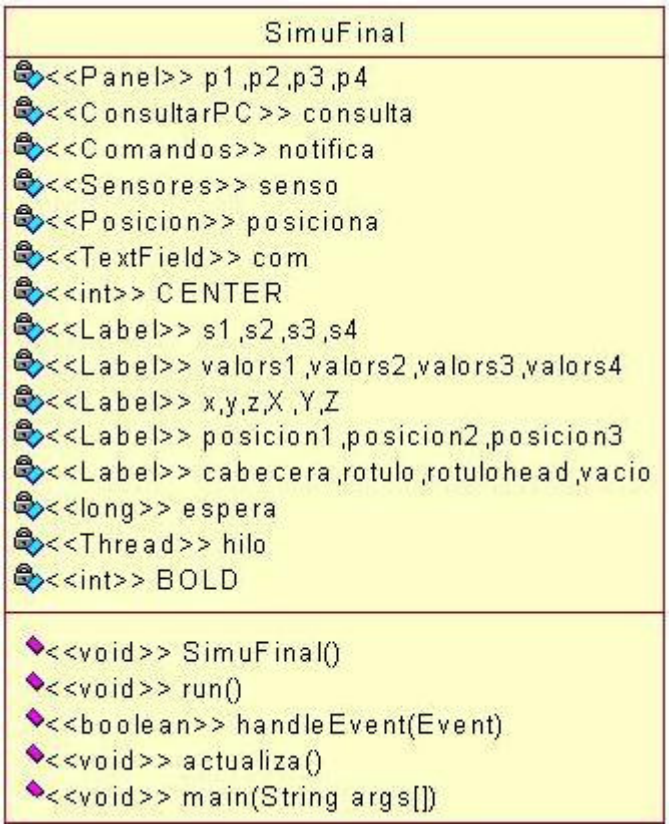
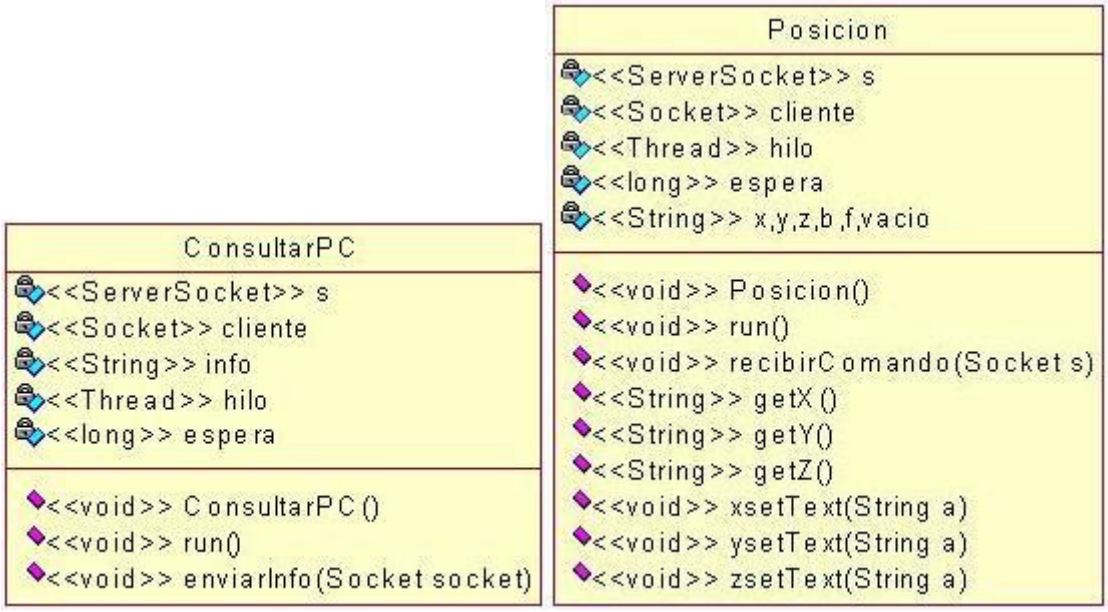


Figura 8. Detalle de las clases que se ejecutarán sobre la PDA


SimuFinal
<<Panel>> p1,p2,p3,p4
<<ConsultarPC>> consulta
<<Comandos>> notifica
<<Sensores>> senso
<<Posicion>> posiciona
<<TextField>> com
<<int>> CENTER
<<Label>> s1,s2,s3,s4
<<Label>> valors1,valors2,valors3,valors4
<<Label>> x,y,z,X,Y,Z
<<Label>> posicion1,posicion2,posicion3
<<Label>> cabecera,rotulo,rotulohead,vacio
<<long>> espera
<<Thread>> hilo
<<int>> BOLD
<<void>> SimuFinal()
<<void>> run()
<<boolean>> handleEvent(Event)
<<void>> actualiza()
<<void>> main(String args[])


Posicion
<<ServerSocket>> s
<<Socket>> cliente
<<Thread>> hilo
<<long>> espera
<<String>> x,y,z,b,f,vacio
<<void>> Posicion()
<<void>> run()
<<void>> recibirComando(Socket s)
<<String>> getX()
<<String>> getY()
<<String>> getZ()
<<void>> xsetText(String a)
<<void>> ysetText(String a)
<<void>> zsetText(String a)
ConsultarPC
<<ServerSocket>> s
<<Socket>> cliente
<<String>> info
<<Thread>> hilo
<<long>> espera
<<void>> ConsultarPC()
<<void>> run()
<<void>> enviarInfo(Socket socket)

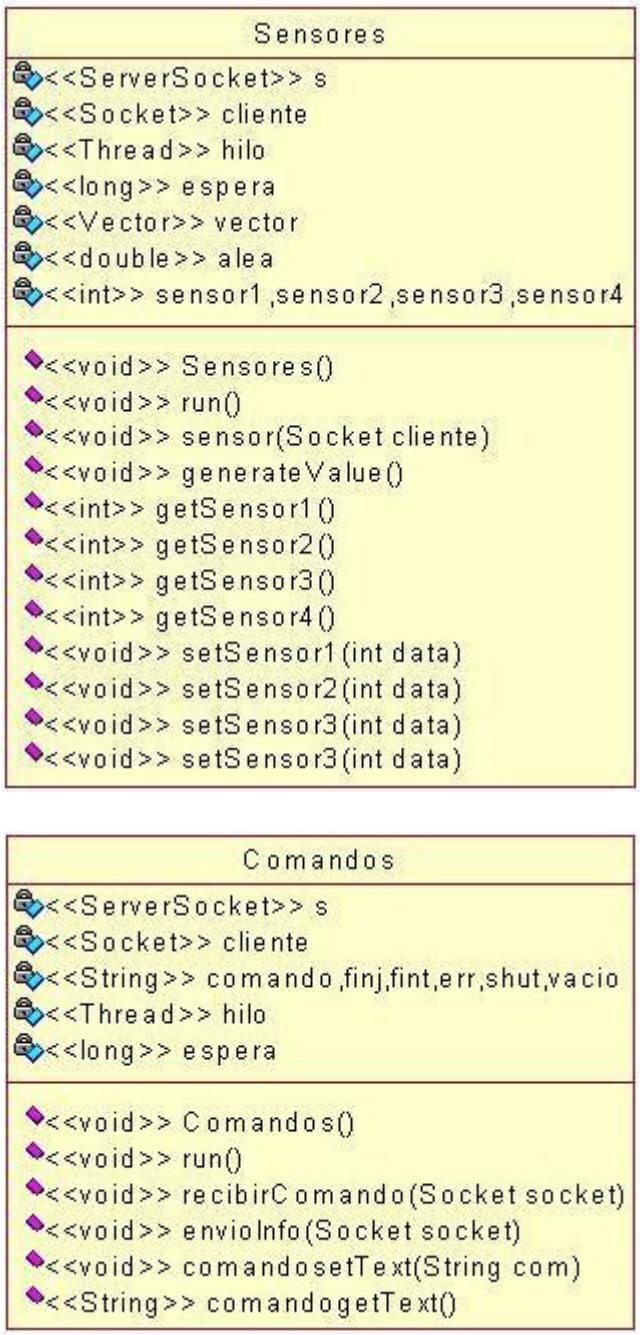


Figura 9. Detalle de las clases que se ejecutarán en el PC

5.4.2.2 Paquetes UML

Los paquetes ofrecen un mecanismo general para organizar los distintos submodelos (subsistemas) en los que se divide un determinado modelo (sistema).

Todas las clases no son necesariamente visibles desde el exterior del paquete, es decir, un paquete encapsula a la vez que agrupa.

La aplicación a desarrollar puede agruparse en dos paquetes: Servidor y PDA.

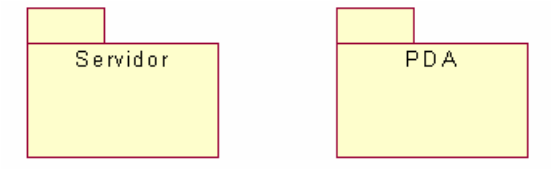


Figura 10. Paquetes del sistema a desarrollar

El paquete Servidor contiene las clases: Simufinal.class, ConsultarPC.class, Comandos.class, Sensores.class, Posición.class. De éstas, la clase SimuFinal no tiene porque ser visible desde el paquete PDA. Esta clase implementa un interfaz gráfico para mostrar la información generada por el sistema, pero no intercambia información con las clases de otros paquetes.

El paquete PDA contiene las clases MenuPrincipal.class, Consultar.class, NotifCom.class, AcepCanc.class, Sens.class, Xyz.class y BarraAlea.class. Las clases Consultar.class, AcepCanc.class, Sens.class, Xyz.class son las encargadas del intercambio de información con las clases del paquete Servidor, por lo que deben de ser visibles desde el exterior. El resto de clases implementan interfaces gráficos y funcionalidad interna del subsistema, por lo que no tienen por que ser visibles desde el exterior.

5.4.3 Diagramas de Secuencia

Los detalles de un caso de uso pueden ser documentados usando diagramas de secuencia. Un diagrama de secuencia muestra el orden en el cual los mensajes son intercambiados entre los actores y el sistema. El diagrama de secuencia representa los participantes mediante cajas rectangulares. Partiendo de la caja rectangular, se extiende una línea discontinua vertical. Los mensajes intercambiados entre los participantes son ilustrados como flechas orientadas y son etiquetados por el mensaje que esta siendo comunicado. La secuencia de mensajes es leída de arriba abajo. De este modo, el tiempo transcurre desde arriba hacia abajo. Cuando la línea discontinua es sustituida por un rectángulo, significa que el objeto esta activo y usando recursos durante ese periodo de tiempo.

La siguiente figura ilustra un ejemplo de diagrama de secuencia de un caso de uso. El ejemplo muestra como un director de banco solicita al sistema crear una nueva cuenta. El sistema pregunta por información acerca del cliente, la cual es proporcionada por el director del banco. El sistema entonces pide al director identificar el tipo de cuenta, y el banco suministra esta información. El sistema solicita la información del balance inicial, y el manager la proporciona. Una vez que toda la información ha sido adquirida, el sistema realiza una petición a la base de datos de cuentas para crear una nueva cuenta. El manager es informado por el sistema cuando la cuenta ha sido creada con éxito.

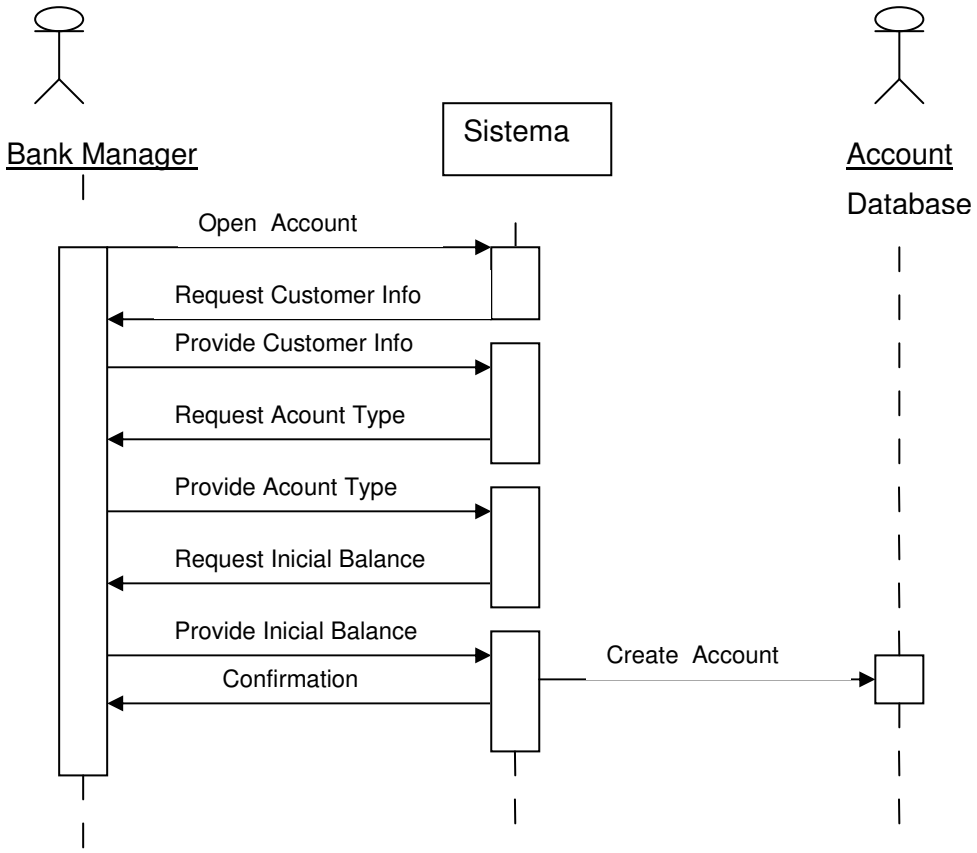


Figura 11. Diagrama de secuencia que ilustra la interacción detallada entre el sistema y los actores del caso de uso Open Account

Diagramas de Secuencia de la Aplicación a Desarrollar

Las siguientes ilustraciones muestran los diagramas de secuencia de todos y cada uno de los casos de uso definidos anteriormente para la aplicación que se va a implementar.

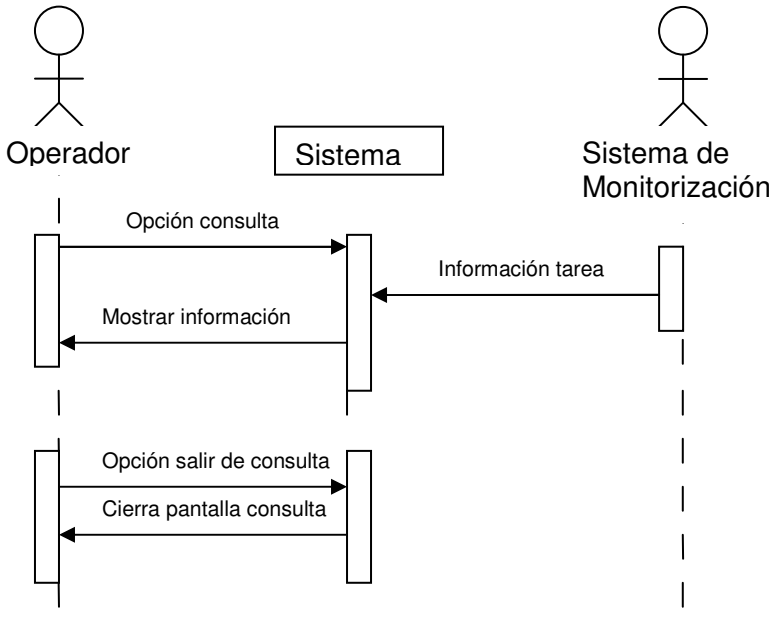


Figura 12. Caso de Uso Solicitud notificación tarea

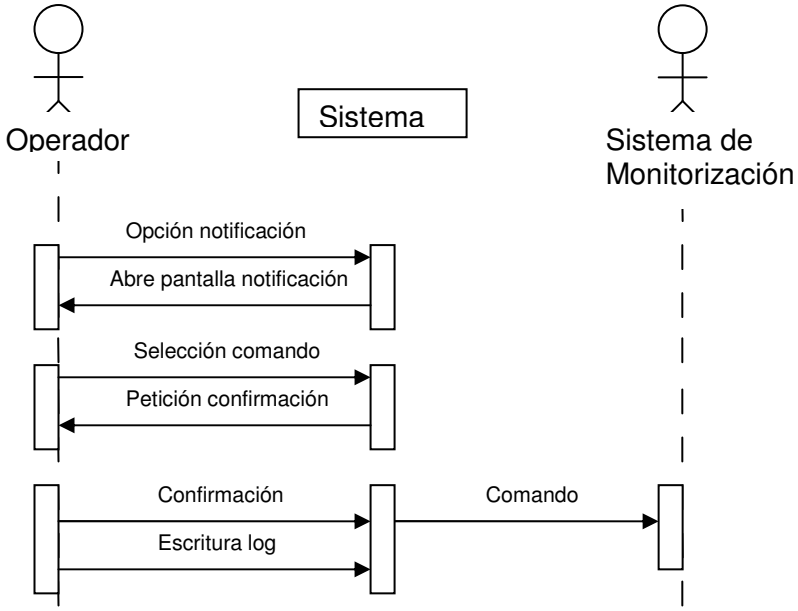


Figura 13. Caso de Uso Envío notificación

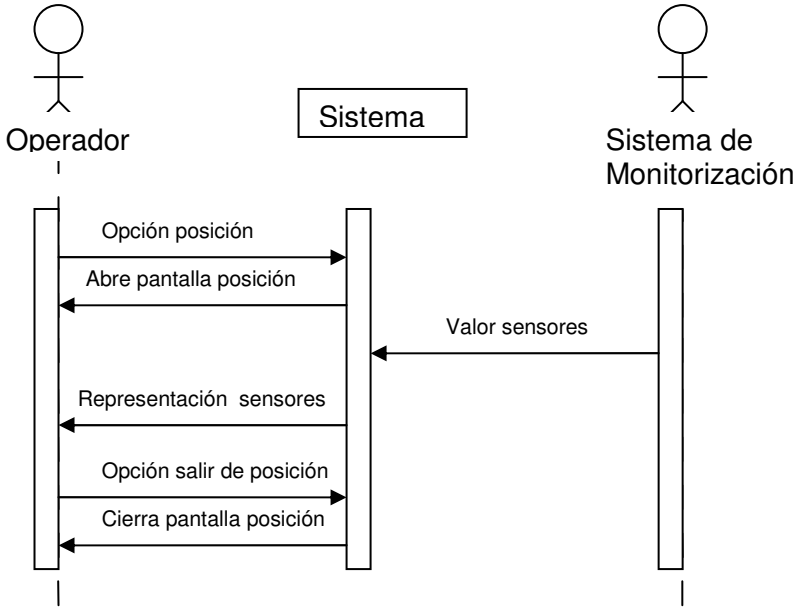


Figura 14. Caso de Uso Monitorizar estado sensores

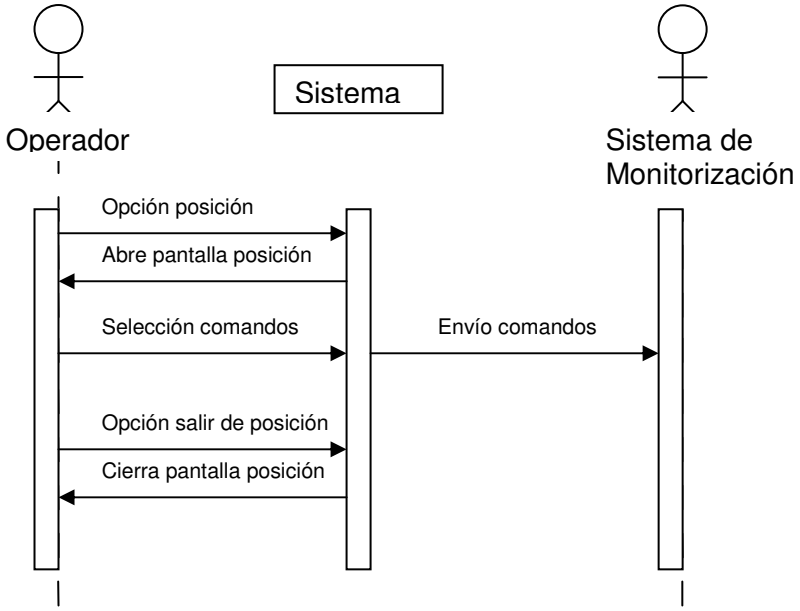


Figura 15. Caso de Uso Mover ejes mesa

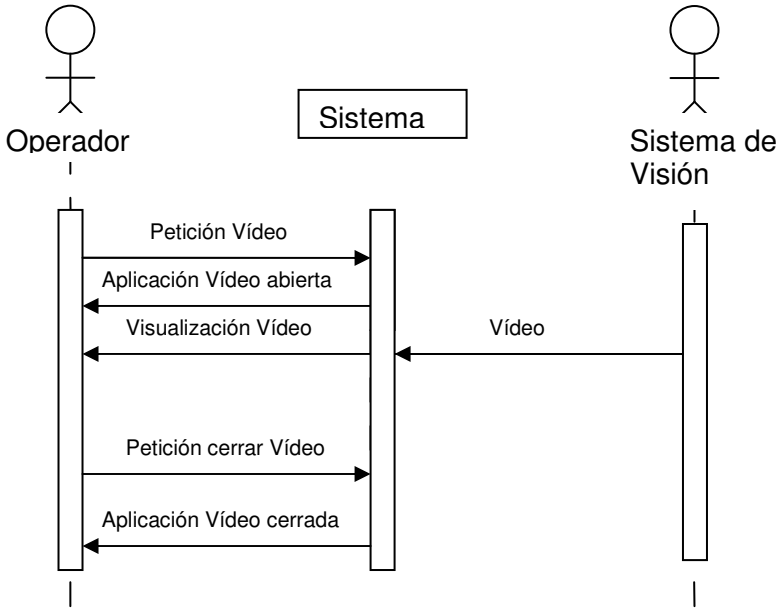


Figura 16. Caso de Uso Captura de vídeo

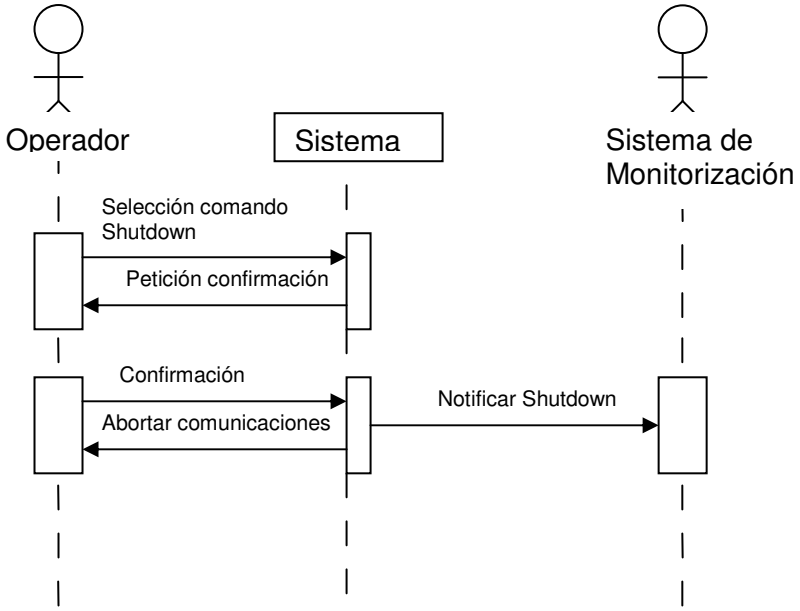


Figura 17. Caso de Uso Shutdown

5.4.4 Diagrama de Componentes

Un componente es un grupo de clases que trabajan estrechamente. Los componentes se suelen corresponder con código fuente, binario o ejecutable. Un diagrama de componentes permite modelar la estructura del software y la dependencia entre componentes. Una relación de dependencia indica que un componente utiliza otro.

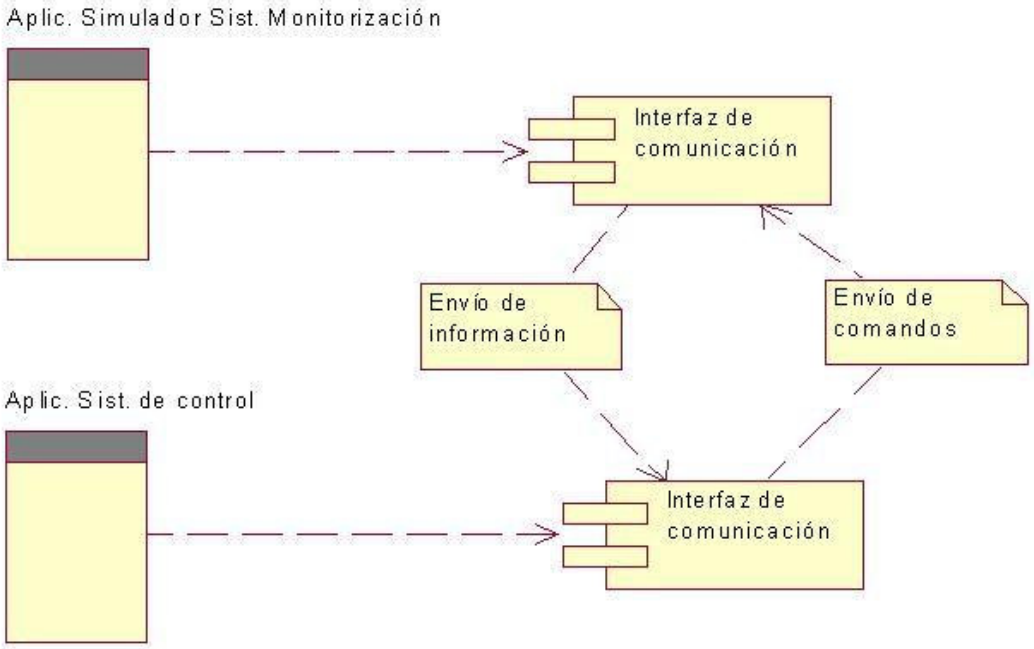


Figura 18. Diagrama de componentes de la aplicación a desarrollar

5.4.5 Diagrama de Distribución

Modela como se distribuyen en tiempo de ejecución los elementos de procesamiento y los componentes software, así como los procesos y objetos asociados.

En el diagrama de distribución se modelan los nodos de procesamiento y la comunicación entre ellos. Cada nodo puede contener varias instancias de uno o más componentes.

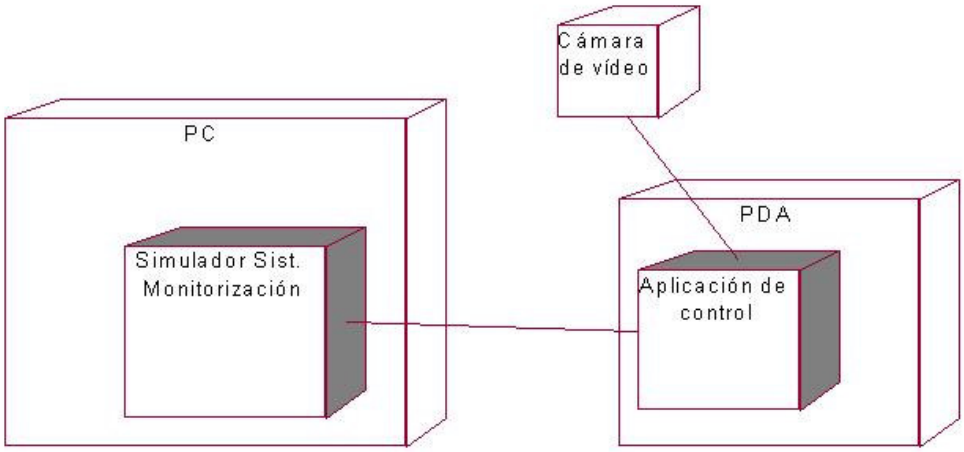


Figura 19. Diagrama de distribución de la aplicación a desarrollar

Capítulo 6
Implementación de la Aplicación

6.1 Introducción

El objetivo de este capítulo es exponer la implementación de la aplicación desarrollada, así como las fases que se han llevado a cabo para alcanzar la funcionalidad que se demanda de dicha aplicación. A continuación se resumen las principales funciones que se esperan de la infraestructura hardware/software que se ha implementado:

- Recepción y representación gráfica, a través de la pantalla de la PDA, de los valores de los sensores situados en la mesa robótica. De este modo también podemos activar mecanismos visuales que alerten al operario cuando las extremidades del robot se encuentren a muy poca distancia de la superficie. Se implementará un mecanismo de alerta basado en un parámetro programable. Dicho parámetro define cual es la distancia a partir de la cual el operario debe de prestar más atención, o realizar los desplazamientos del cabezal de chorreado de manera más precisa. Siendo este parámetro programable permitimos que la activación de la señal pueda ser modificada en función del entorno o las necesidades específicas de cada tarea.

- A través de la PDA el operario puede comunicarse con el sistema de control para indicar las acciones de posicionamiento que deben de llevarse a cabo o enviar notificaciones sobre las tareas que se están realizando. Las posibles notificaciones a enviar serán seleccionadas desde un menú mostrado en la pantalla del dispositivo. De igual forma el operador también puede solicitar información sobre las tareas que debe realizar; la información sobre las mismas será mostrada en la pantalla de la PDA. Las acciones que provoquen los comandos enviados son transparentes a la aplicación implementada sobre la PDA. Entre la información intercambiada también se encuentra la relativa a las tareas de movilización del robot. Estas se realizan mediante el envío de comandos de posición al sistema de control. Los botones físicos del Pocket PC deben ser aprovechados al máximo, evitando dentro de lo posible el uso de botones software para el envío de comandos de posicionamiento.

- La plataforma tipo PDA captura vídeo en tiempo real. El sistema de visión dispone de una o más cámaras que captan las imágenes del casco del barco. Mostrando estas imágenes sobre la pantalla del terminal, el operador puede ver en todo momento cómo evolucionan las tareas de chorreado y limpieza. Con este propósito se ha adquirido un módulo de expansión multimedia *FlyJacket i3800* de *LifeView*, el cual está dotado de un software específico para captura de vídeo. En este primer prototipo del subsistema constituido por la PDA se ha considerado la captura de vídeo desde una única cámara.

- Se ha estudiado el funcionamiento de los sockets TCP/IP implementados con el lenguaje Java al ejecutarse sobre una red inalámbrica.

Una vez definida la funcionalidad que el sistema debe alcanzar, se exponen las fases y los pasos llevados a cabo para la implementación de la aplicación.

En primer lugar, se prepara la infraestructura física necesaria para poder afrontar el desarrollo y prueba de la aplicación software. Para escribir el código es necesario haber instalado previamente cierto software que permita la compilación y prueba del mismo. Se dispone de un PC donde escribir el código y de la propia PDA donde deberá ejecutarse finalmente, y en la cual se han hecho las pruebas de cada módulo software desarrollado. Para completar la infraestructura hardware se instala el módulo de expansión multimedia que conectado a la PDA permitirá la captura de vídeo.

La segunda fase consiste en la implementación del código. Esta implementación se desarrollará en distintos módulos que se irán validando de manera independiente sobre la plataforma física antes instalada.

6.2 Instalación y Configuración de la Plataforma Física

Al igual que se expuso ampliamente en el Capítulo 3, dedicado a la infraestructura utilizada para desarrollar la aplicación, tres han sido los elementos físicos que la constituyen: *iPAQ Pocket PC Serie H5500*, módulo de expansión multimedia *FlyJacket i3800* de *LifeView*, y un ordenador de escritorio. En esta ocasión el texto se centra exclusivamente en las funciones y configuraciones que han sido necesarias para el desarrollo de la aplicación final.

6.2.1 Configuración de un Ordenador Personal

Para dotar de capacidad wíreless al ordenador personal, es suficiente con instalar vía USB una tarjeta inalámbrica de modo que pueda configurarse una comunicación ad-hoc ya sea WLAN o mediante Bluetooth.

Para la programación del código Java se ha instalado en el ordenador la versión del JDK jdk1.2.2. La especificación de *PersonalJava* y su equivalente *J2ME Personal Profile* están basados en la versión jdk1.2, por lo que se puede compilar el código con esta versión en el ordenador, y después ejecutarlo sobre el entorno de ejecución Java del *iPAQ Pocket PC*. También es posible utilizar para la compilación una versión más actual, por ejemplo el j2sdk1.4.2_01. Si el código escrito es conforme a la especificación de *PersonalJava*, su ejecución será correcta sobre la PDA, independientemente de que se haya compilado con un jdk superior. No obstante es posible la instalación de un entorno de emulación *PersonalJava* en el ordenador, como ya se expuso anteriormente, para probar las aplicaciones antes de instalarlas en el iPAQ Pocket PC. En cualquier caso, por el tamaño de los módulos de software desarrollados es eficiente probar las aplicaciones directamente sobre la PDA.

Como ya se ha dicho anteriormente, el ordenador también servirá de soporte para la configuración e instalación tanto del *iPAQ Pocket PC* como del módulo multimedia *FlyJacket i3800* de *LifeView*.

6.2.2 Puesta en Marcha de la PDA

6.2.2.1 Arranque del Dispositivo

La PDA escogida, *HP iPAQ Pocket PC Serie H5500*, ha sido previamente descrita con detalle en el Capítulo 3. A lo largo de este capítulo se describen aquellas funciones o capacidades de la plataforma móvil que han sido necesarias para la implantación de la aplicación software. En primer lugar, la puesta en marcha del Pocket PC e instalación del software *ActiveSync*. Una vez comprobado el funcionamiento de la plataforma se pasa a la instalación de la máquina virtual de Java y se ejecutan algunas aplicaciones Java para observar el funcionamiento y opciones disponibles.

Para encender el *iPAQ* por primera vez se debe de realizar un reinicio normal o “reinicio por software”. El *iPAQ Pocket PC* se entrega con un puntero que se utiliza para

puntear o escribir en pantalla. Utilizando dicho puntero se debe de presionar ligeramente el botón de reinicio situado en la parte inferior del iPAQ. Cuando se enciende el dispositivo por primera vez aparece una guía para la alineación de la pantalla. Una vez alineada la pantalla el sistema solicita la selección de una zona horaria. Una vez hecho esto, se deben de establecer la fecha y la hora en la unidad. A continuación se muestra la pantalla de inicio y el dispositivo queda disponible para su utilización.

Una vez iniciado el iPAQ si se vuelve a efectuar un reinicio normal, se detendrán las aplicaciones que se estén ejecutando, pero no se borran los programas ni los datos guardados.

Por motivos de seguridad se ha optado por configurar un código de acceso a la unidad. La función de seguridad del *iPAQ Pocket PC* permite que se solicite un PIN, una contraseña y/o una huella digital. Si se especifica de manera repetitiva o incorrecta un código PIN, una contraseña o una huella digital no válidos, todos los datos y programas almacenados en la memoria RAM se eliminarán, y el *iPAQ Pocket PC* volverá a su configuración original.

Existen varias maneras de establecer las contraseñas de inicio de sesión. Se puede seleccionar:

- Sin contraseña (opción predeterminada)
- PIN simple de cuatro dígitos
- Contraseña alfanumérica fuerte
- PIN o huella digital
- PIN y huella digital
- Contraseña o huella digital
- Contraseña y huella digital
- Sólo huella digital

Para la fase de construcción del sistema se ha considerado suficiente la configuración de un código PIN sencillo de cuatro dígitos para inicio de sesiones. Estos son los pasos seguidos:

1. En la pantalla *Hoy*, puntear en *Inicio* -> *Configuración* -> *Contraseña*
2. Selección de la opción "*PIN simple de 4 dígitos*" en la lista desplegable.
3. Selección en la lista desplegable del tiempo en el cual el iPAQ puede ser utilizado sin necesidad de volver a especificar el código PIN: 2 horas.
4. Selección mediante el puntero de cuatro dígitos en el campo *PIN.*
5. Puntear *OK* o *Intro.* Salvar cambios.

Si se utiliza ActiveSync es necesario introducir también la contraseña en el ordenador personal para utilizar el servicio.

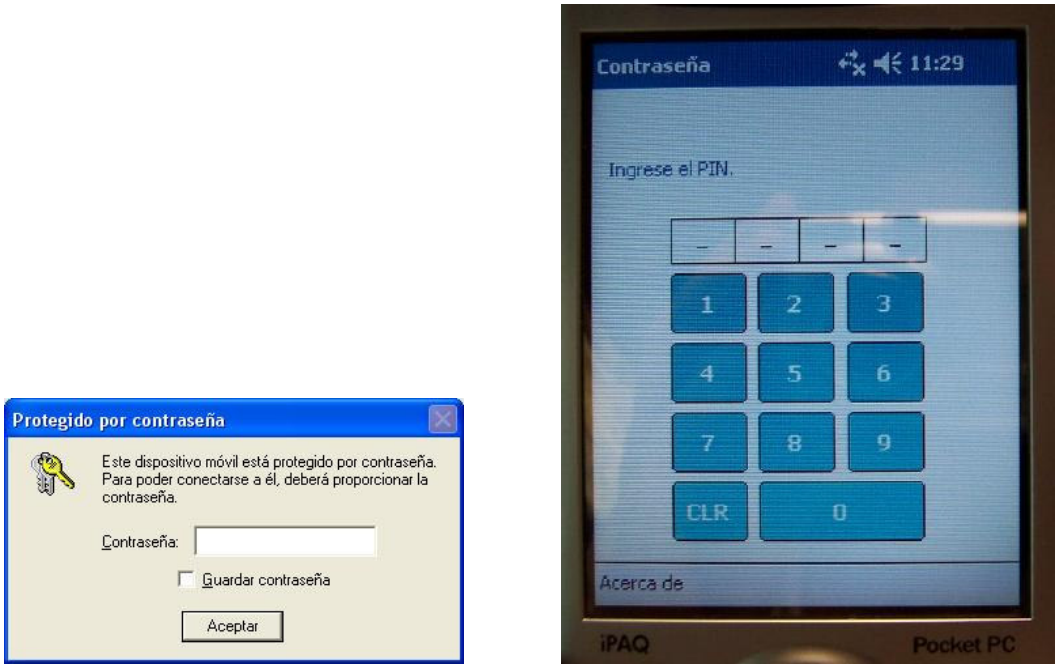


Figura 1. Petición de contraseña en el PC y el Pocket PC respectivamente.

En este momento el dispositivo se encuentra arrancado y con los parámetros básicos configurados. Se pasa pues a la instalación de *ActiveSync.*

6.2.2.2 Instalación ActiveSync

La aplicación ActiveSync se proporciona junto con la PDA, estando contenida en el CD de documentación del *iPAQ Pocket PC*. Su instalación es guiada de manera gráfica al abrir el CD. Se ha efectuado una instalación por defecto. Es importante no conectar el iPAQ al PC antes de que *ActiveSync* esté correctamente instalado en el ordenador. El tipo de conexión utilizada en este proyecto ha sido la conexión mediante la base de escritorio universal, que es suministrada junto con la PDA y se conecta al ordenador mediante puerto USB sin necesidad de configuración adicional.

El tipo de asociación seleccionada en todas las conexiones que se han establecido ha sido "*Asociación como invitado*". Seleccionando este tipo de asociación es posible el intercambio de archivos entre PC y la PDA, y la instalación de aplicaciones en el *iPAQ Pocket PC*. Precisamente estas son las dos funciones de *ActiveSync* que han sido necesarias utilizar en el desarrollo de la aplicación. No ha sido necesaria la sincronización de archivos o servicios entre el PC y el *iPAQ Pocket PC*.

El intercambio de archivos se realiza tal y cómo se explicó en el Capítulo 3.

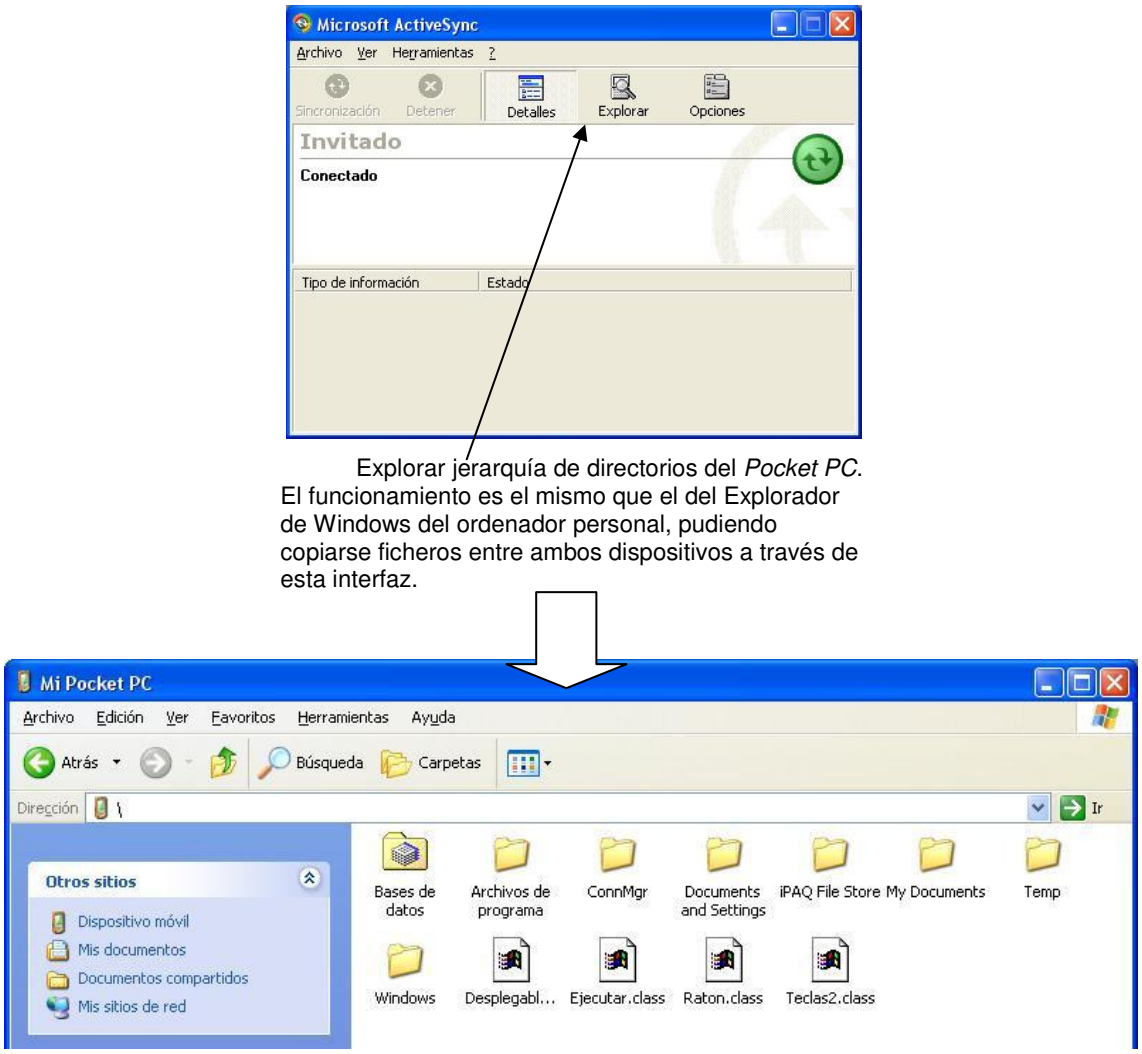


Figura 2. Pantalla inicio de *ActiveSync* con el estado de conexión activado y explorador de archivos del *Pocket PC*

A la hora de programar el código de la aplicación, se han ido probando los distintos módulos sobre la PDA. Una vez validados, respecto a su funcionalidad, se copian en la carpeta de *Almacenamiento de archivos de iPAQ*. Las aplicaciones y los archivos almacenados en la carpeta de *Almacenamiento de archivos de iPAQ* se conservan en la memoria ROM, y se guardan si se realiza un reinicio completo del iPAQ Pocket PC o si la batería se descarga completamente.

Antes de comenzar a guardar archivos en la carpeta de *Almacenamiento de archivos de iPAQ*, se ha determinando la cantidad de memoria disponible en la carpeta: *Inicio* -> *Configuración* -> ficha *Sistema* -> ficha *Memoria* -> *Tarjeta de Almacenamiento* -> *Almacenamiento de archivos de iPAQ*. Para el modelo usado, *iPAQ Pocket PC Serie 5500*, la memoria ROM asignada a esta carpeta es de 17'4 M, tamaño más que suficiente para guardar cuantos archivos de código Java se quiera.

De manera periódica se han ido realizando copias de seguridad. A la hora de restaurar la información hay que tener en cuenta que se sustituye la información actual del *iPAQ pocket PC* por la información guardada en la copia de seguridad. Las copias de seguridad pueden hacerse mediante *iPAQ Backup*, pudiendo escoger la carpeta de almacenamiento entre tarjetas externas a la PDA o carpetas internas. También se pueden realizar copias de seguridad mediante *Microsoft ActiveSync*.

6.2.2.3 Capacidades Wireless del iPAQ Pocket PC

Dos son las opciones de las que dispone el iPAQ Pocket PC para comunicarse de manera inalámbrica con otros dispositivos: WLAN y Bluetooth. Mediante el protocolo WLAN 802.11b, la PDA puede comunicarse con otras dispositivos Wireless hasta una distancia de 100 metros; usando el protocolo Bluetooth esta distancia es de 10 metros.

Para ambos protocolos, una vez que los dos dispositivos estén configurados (en este caso iPAQ y PC), la detección de otras redes o dispositivos inalámbricos es automática, siempre que estén dentro de su rango de alcance. Cuando la red o dispositivo remoto es detectado, seleccionando éste, se pueden configurar las distintas opciones disponibles y expuestas en el Capítulo 3.

El iPAQ Pocket PC serie hp5500 dispone de capacidad Bluetooth integrada. Para hacer uso de WLAN se puede adquirir un módulo de expansión Plus de PC Card.

6.2.2.4 Instalación de un Entorno de Ejecución Java en el iPAQ Pocket PC

Una vez copiado el ejecutable de *JeodeRuntime de Insignia* en el ordenador de escritorio, se establece una asociación entre el *iPAQ Pocket PC* y el ordenador. Es suficiente con lanzar el ejecutable en el PC para que se instale en la PDA. Se ha seleccionado la ruta por defecto para la instalación: *Inicio-> Programas-> Jeode*. Dentro de la carpeta Jeode, que se ha creado de manera automática, se encuentran tres archivos. El correspondiente al icono de *Jeode* es el que da acceso a la línea de comandos para ejecutar los archivos *.class*. Se observa que se ha creado una carpeta llamada Examples. Esta carpeta contiene dos ejemplos que sirven para comprobar que el funcionamiento y la instalación de la máquina virtual con correctos.

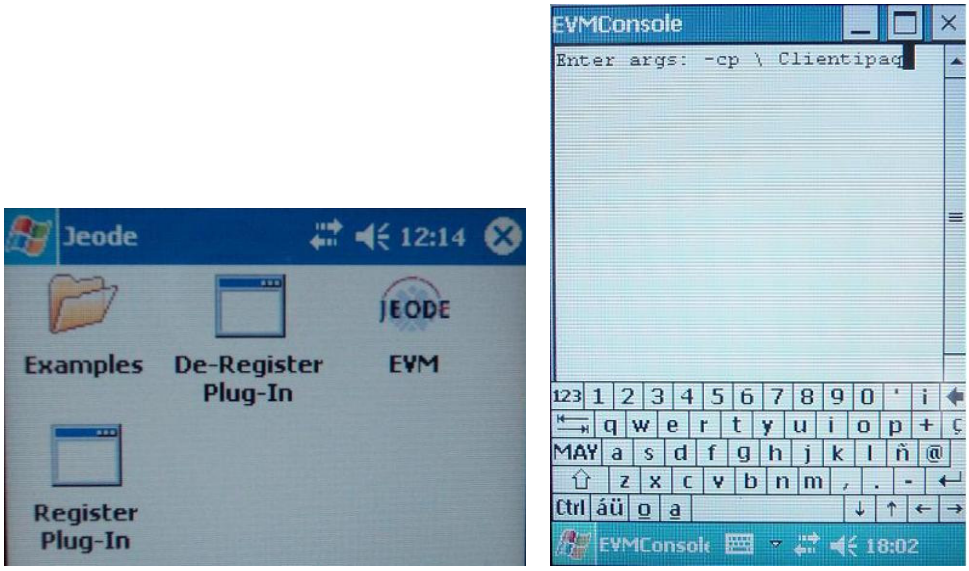


Figura 3. a) Contenido de la carpeta *Jeode.* b) Consola de comandos del entorno de ejecución de Java

La PDA no dispone de un ratón ni de un teclado convencional. Se ha estimado oportuno hacer algunas pruebas previas para ver como se trabaja con el puntero, y al mismo tiempo comenzar a familiarizarse con *PersonalJava*.

Para comprobar el funcionamiento del puntero sobre la pantalla del *iPAQ Pocket PC* al usar aplicaciones con entornos gráficos, se ha programado un sencillo entorno gráfico con cuatro botones. Un área de texto en la parte inferior de la pantalla debe de mostrar el valor del botón que ha sido marcado con el puntero.

Tras la ejecución del programa se ha observado que el puntero actúa de manera similar a lo que lo haría un ratón en un PC, generando eventos en los componentes que presiona.

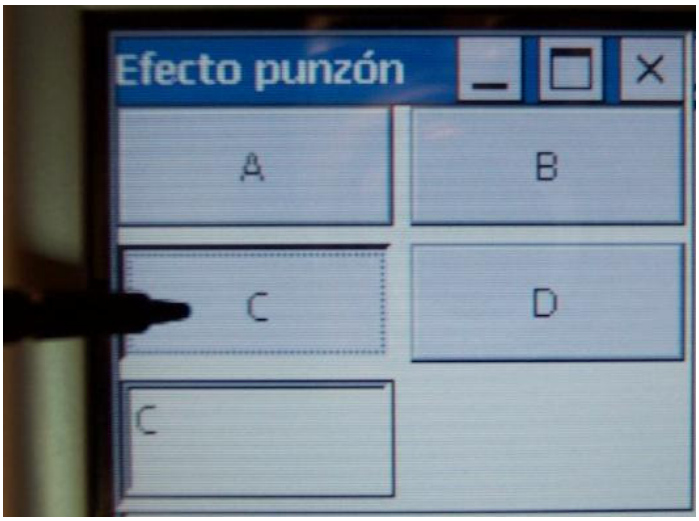


Figura 4. Ejecución del programa Raton.java en el Pocket PC

Cuando un botón es presionado por el punzón, debe de lanzar un evento. La captura de este evento provoca la aparición, en la caja de texto, de la letra que identifica a cada botón. El código comentado de esta aplicación se muestra a continuación:

```
import java.awt.*;
 class Raton extends Frame{
  private Button a,b,c,d; // se definen cuatro botones
  private TextField indicador; //muestra cual de los botones se ha pulsado
          public Raton() {
      super("Efecto punzón");
      a= new Button("A"); // se crean los botones
      b= new Button("B");
      c= new Button("C");
      d= new Button("D");
      indicador= new TextField("Dice que botón pulsastes");//texto por defecto
      setLayout(new GridLayout(3,2,5,5));
      //añadimos los objetos al Frame
      add(a); add(b); add(c); add(d); add(indicador);
      setSize(150,150);
      setVisible(true);
      }
//se capturan los eventos
public boolean action(Event e, Object o){
          if (e.target instanceof Button){//evento generado por botón
              if (e.target==a){//identifica que botón se pulsó
                  indicador.setText("A");
                  return true;
              }
```

```
            if (e.target==b){
                    indicador.setText("B");
                    return true;
                }
            if (e.target==c){
                    indicador.setText("C");
                    return true;
                }
            if (e.target==d){
                    indicador.setText("D");
                    return true;
                }
                return true;
        }
        return true;
    }
        public static void main(String args[]){
        Raton prueba = new Raton();
        }
}
```

El siguiente programa muestra alguna de las capacidades gráficas de *PersonalJava*. Se ha querido estudiar la posibilidad de construir menús flotantes sobre la PDA, ya que podría ser interesante agregar alguno a la aplicación final.

Figura 5. Ejecución de un entorno gráfico con menú flotante sobre el iPAQ Pocket PC

El código del programa es el siguiente:

```
import java.awt.*;
class Desplegable extends Frame {
 public Desplegable(){
     super("Menú desplegable");
     MenuBar barra= new MenuBar();
     //Crear menús
     Menu viewMenu= new Menu("Ver");
     Menu confirmar= new Menu("Confirmar");
     Menu notificar= new Menu("Notificar");
```

```
        //construir submenús
        confirmar.add(new MenuItem("Aceptar"));
        confirmar.add(new MenuItem("Cancelar"));
        notificar.add(new MenuItem("Good news"));
        notificar.add(new MenuItem("Bad news"));
        //construir menú de 'Ver'
        viewMenu.add(confirmar);
        viewMenu.add(new MenuItem("-"));
        viewMenu.add(notificar);
        viewMenu.add(new MenuItem("-"));
        barra.add(viewMenu);
        setMenuBar(barra);
        resize(150,150);
        show();
    }
    public static void main(String args[]){
        Desplegable prueba = new Desplegable();
        }
}
```

Se ha establecido una comparación entre la memoria asignada a la máquina virtual de Java en el PC y en el Pocket PC, así como entre las memorias disponibles durante la ejecución de un sencillo programa escrito en Java.

Este es el código del programa ejecutado en ambas plataformas para la obtención de los datos.

```
import java.io.*;
import java.util.*;
import java.lang.*;

public class MemoriaVM{

    public static void main( String args[] ) {

    Runtime rt = Runtime.getRuntime();
    System.out.println("Total memory allocated to VM: " + rt.totalMemory());
          System.out.println("Memory currently available: " + rt.freeMemory());

        }
   }
```

Los resultados obtenidos se muestran en la siguiente tabla.

	PC	Pocket PC
Memoria para VM	2031616	131072
Memoria disponible en ejecución	1530512	12300

Tabla 1. Comparativa de la memoria para JVM en un PC y en el iPAQ Pocket PC

6.2.3 Puesta en Marcha del FlyJacket i3800

El siguiente paso en el desarrollo del sistema es la instalación del módulo de expansión multimedia *FlyJacket i3800 de LifeView*.

En primer lugar se instalan los drivers del módulo multimedia. Los drivers proporcionados en el CD junto con el módulo multimedia (*FlyJacket_driver1.7*) no fueron compatibles con el modelo de *iPAQ Pockect PC serie 5500*. La casa fabricante tampoco puso a disposición en su página web la actualización de los drivers, por lo que fue necesario ponerse en contacto con el servicio técnico de dicha casa para que finalmente los proporcionasen. El driver instalado finalmente fue: *FlyJacket(Ver 1.82 Only for iPAQ 5450)*

Al igual que en la instalación de *JeodeRuntime*, estableciendo una conexión entre el PC y el *iPAQ Pocket PC* a través de *ActiveSync*, la instalación se realiza de manera automática en la PDA al lanzar los ejecutables en el PC.

En dispositivos tales como las PDAs, son características las limitaciones en recursos tales como la memoria. Teniendo en cuenta este factor se ha decidido por el momento instalar únicamente el módulo *IA capture* de la *IA Style PowerMedia Suite*. Con este módulo instalado, es posible la captura de vídeo en tiempo real sobre la pantalla del *iPAQ Pocket PC*.

Es posible seleccionar el estándar de vídeo entrante, el tamaño de la captura, carpeta de almacenamiento predeterminada y visualización a pantalla completa, entre otras opciones ya expuestas en el Capítulo 3.

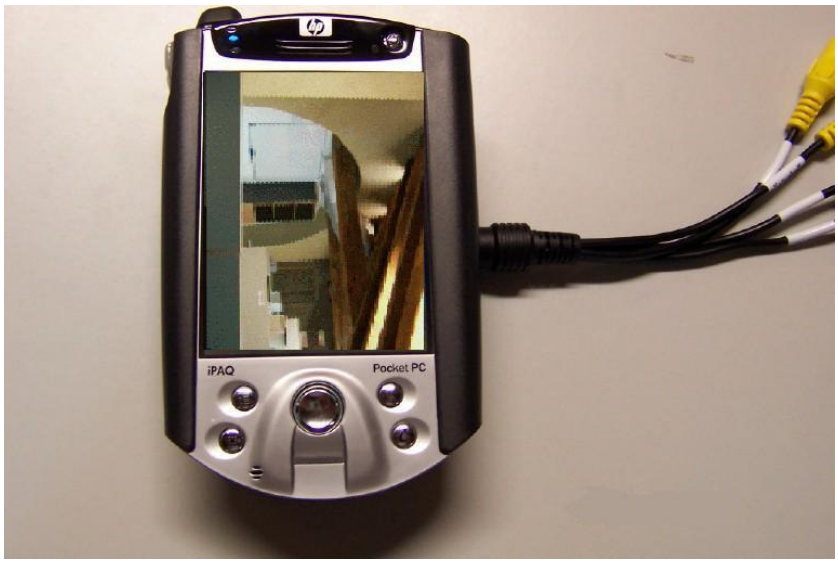


Figura 6. iPAQ Pocket PC capturando vídeo en tiempo real a pantalla completa

6.3 Implementación de la Funcionalidad de los Módulos Software

6.3.1. Introducción

Se ha buscado modularidad a la hora de solucionar el problema planteado por la aplicación. Se han dividido los objetivos de la aplicación en áreas funcionales y se han desarrollado de manera independiente. Al tratarse de una plataforma no común sobre la que programar, es lógico pensar que pueda surgir más de un problema inesperado. Tratando por separado las distintas funciones de la aplicación, estos problemas son fácilmente acotados, y no tienen porque interferir en otras partes de la implementación.

Antes de entrar en la implementación de la funcionalidad que demanda el sistema, se han resuelto dos aspectos que aún no estando ligados de manera directa a la funcionalidad a alcanzar, han de ser empleados en la aplicación. En primer lugar, hay que ver como se resuelve la comunicación *wireless* a través de Java. El segundo aspecto trata de ver como utilizar los botones físicos de la PDA para controlar parte de la aplicación software.

6.3.1.1 Comunicación Wireless Usando el Lenguaje Java

Para observar el funcionamiento de los *Sockets* escritos en Java sobre una comunicación inalámbrica, se ha implementado una pequeña aplicación. Se trata de una clase servidor y otra clase cliente que se comunican vía *Sockets TCP*. La comunicación se establece entre dos dispositivos dotados de tarjetas inalámbricas conformes al estándar WLAN 802.11b. En una de estas máquinas se ejecuta el servidor, que envía mensajes al cliente cada vez que se pulsa un botón perteneciente al entorno gráfico del programa. En la otra máquina se ejecuta el cliente, que recibe los mensajes enviados por el servidor y los muestra en una caja de texto.

La figura muestra la ejecución del programa y la comunicación que se establece entre ambos dispositivos.

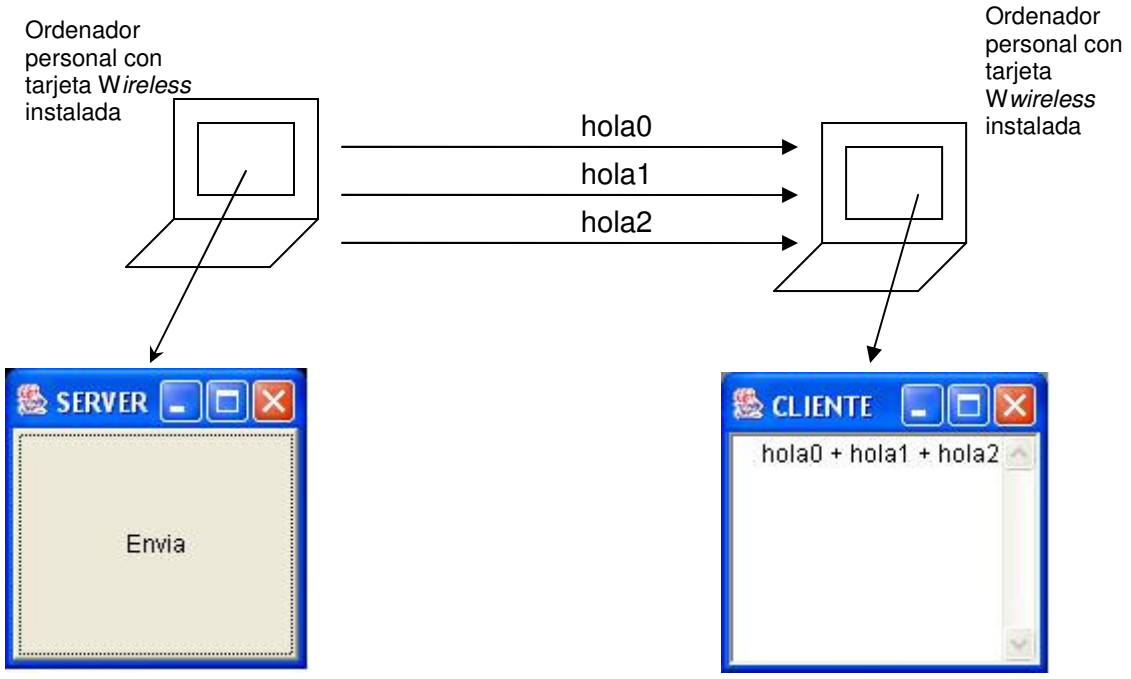


Figura 7. Establecimiento de un socket escrito en Java vía wíreles

El código de ambas clases, servidor y cliente, es el siguiente:

- Clase servidor

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
//Acepta peticiones de clientes y cada vez que el botón es pulsado envía
//un mensaje por el canal creado
class Works extends Frame{
private Button boton;
ServerSocket s = null; // definimos el socket
Socket cliente = null;
private static int count;
public Works(){
super("SERVER");
count=0;
boton = new Button("Envía");
add(boton);//añadimos el botón al frame
setSize(150,150);
setVisible(true);
try {
          s = new ServerSocket( 3003 );//puerto de escucha=3003
     } catch( IOException e ) {System.out.println( e );}
while (true) { //esperando la conexión de clientes
       try {
              cliente = s.accept();//se acepta petición del cliente
          } catch( IOException e ) {
               System.out.println( e );}
       }
}
public boolean action(Event e, Object o){
     if (e.target instanceof Button){
     if (e.target==boton){ //se captura evento de botón
     try{
        manejaPeticion(cliente);//envío el mensaje al cliente
        }catch( IOException io ) {
           System.out.println( io );}
        return true;
     }
     return true;
  }
```

```
        return true;
      }
      //método que maneja la petición del cliente
    public static void manejaPeticion (Socket s) throws IOException  {
          PrintWriter salida = new PrintWriter (new
          OutputStreamWriter(s.getOutputStream()),true);
          String sms= "hola"+ String.valueOf(count);
          salida.println(sms); //enviamos el mensaje al cliente
          count++; //incrementamos en 1 cada vez que enviamos un mensaje
          s.close(); //cerramos el socket
       }
    public static void main( String args[] ) {
          Works prueba = new Works();
            }
}
```

- Clase cliente

```
      import java.awt.*;
      import java.awt.event.*;
      import java.net.*;
      import java.io.*;
      import java.util.*;
// socket cliente. Crea un canal de comunicación con el servidor y a través de
//él recibe mensajes y los muestra en el área de texto.
      public class Workc extends Frame
      {
       private TextArea campo;// aquí se muestran los mensajes
       public Workc ()
       {
          super("CLIENTE");
          campo = new TextArea(80,80);
          add(campo);
          setSize(150,150);
          setVisible(true);
       }
          public void recibirNota() //maneja la rx de mensajes
          {
             try{
             String servidor = "80.32.201.78";//IP del servidor
             int puerto = 3003 ; // puerto del socket
             //creamos el socket
             Socket socket= new Socket(servidor , puerto);
             DataInputStream sIn;
            sIn = new DataInputStream (socket.getInputStream());
```

```
                String  texto = sIn.readLine ();//leemos del canal
                campo.setText(texto);
                sIn.close();//cerramos flujo
                socket.close(); // cerramos el socket
                }
                catch (UnknownHostException e )
                { e.printStackTrace();
            System.out.println(" debes estar conectado para que esto funcione
bien");
                }catch (IOException e)
                {   e.printStackTrace();
                }


            }
        public static void main( String args[] ) {
            Workc comunica = new Workc();
            while(true)//cliente escuchando en todo momento
            comunica.recibirNota();}}
```

Se ha observado que el comportamiento de la aplicación cliente/servidor es el mismo si se ejecuta sobre dos máquinas pertenecientes a una misma red cableada, que si la comunicación establecida es inalámbrica. Para los Sockets implementados en Java, el hecho de que la comunicación entre cliente y servidor sea cableada o inalámbrica es transparente. Al solicitar una conexión, el cliente debe de especificar la dirección IP de la máquina servidora, de igual modo que se haría si la comunicación fuese a través de una red cableada.

6.3.1.2 Captura del Código los Botones del iPAQ Pocket PC

Como ya ha sido expuesto, se requiere el uso de los botones para enviar comandos o iniciar algún tipo de acción durante la ejecución del software. El iPAQ Pocket PC dispone de cuatro botones que por defecto están configurados como acceso directo a algunas aplicaciones básicas de las que dispone el dispositivo, tales como agenda de contactos, o bandeja de correo. Un quinto botón circular se sitúa entre estos cuatro. El botón circular puede presionarse en cuatro direcciones (arriba, abajo, derecha e izquierda), y puede ser pulsado en el centro, su comportamiento puede compararse al de un *joystick*. No es posible cambiar su configuración.

Para poder hacer uso de estos botones en la aplicación software, es necesario poder capturar el código de cada uno de estos botones físicos.

La figura 8 muestra la botonera del iPAQ Pocket PC y las funciones que los botones tienen asignados por defecto.

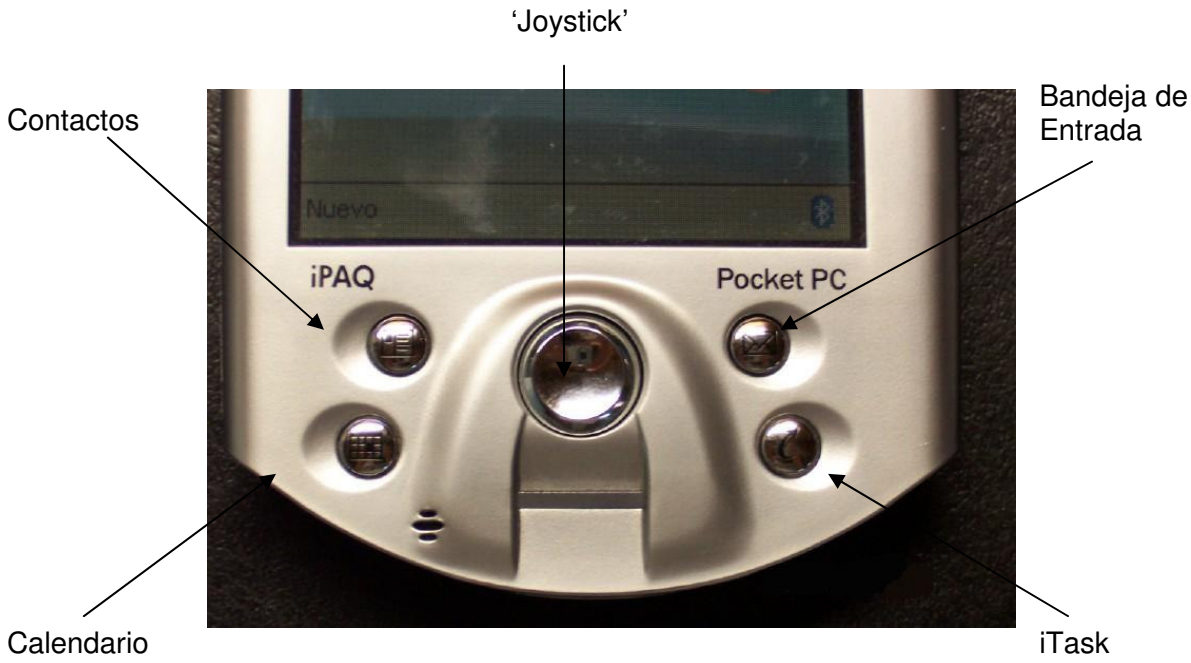


Figura 8. Botones del iPAQ con sus configuraciones por defecto

Para los cuatro botones laterales las posibilidades de asignación son las siguientes:

- <Aceptar/Cerrar>
- <Desplazar a la derecha>
- <Desplazara la izquierda>
- <Desplazar hacia abajo>
- <Desplazar hacia arriba>
- <Hoy>
- <Menú contextual>
- <Menú inicio>
- <Ninguno>
- <Panel de entrada>
- ActiveSync
- Bandeja de entrada
- Bluetooth Manager
- Calculadora
- Contactos
- Explorador de archivos
- iPAQ Image Viewer
- iPAQ Wíreless
- iTask
- Jawbreaker
- LifeView PowerShow
- LifeView Shadow
- IA Capture (FJ)
- Resto de aplicaciones instaladas en el iPAQ

La especificación de PersonalJava dispone de una serie de interfaces y clases para la captura de códigos de teclas. De la especificación se obtiene:

```
java.awt.event
Interface KeyListener

All Superinterfaces: EventListener

public interface KeyListener extends  EventListener
```

KeyListener es un interfaz escuchador para recibir eventos de teclado (keystrokes). La clase interesada en procesar un evento de teclado debe implementar este interfaz (y todos los métodos que este contiene) o extender la clase abstracta `KeyAdapter` (sobre escribiendo únicamente los métodos de interés). El objeto escuchador creado desde esta clase es entonces registrado con un componente usando el método `addKeyListener`. Un evento de teclado es generado cuando una tecla es presionada, soltada, o tecleada (presionada y soltada). El método relevante en el objeto escuchador es entonces invocado, y recibe el KeyEvent como argumento.

Métodos de la interfaz KeyListener:

void **keyPressed**(KeyEvent e), invocado cuando una tecla ha sido presionada.

void **keyReleased** (KeyEvent e), invocado cuando una tecla ha sido soltada

void **keyTyped** (KeyEvent e), invocado cuando una tecla ha sido tecleada. Este evento ocurre cuando una tecla es presionada y a continuación soltada.

A continuación la especificación de la clase KeyEvent:

```
java.awt.event
Class KeyEvent

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--Java.awt.AWTEvent
              |
              +-- java.awt.event.ComponentEvent
                    |
                    +-- java.awt.event.InputEvent
                          |
                          +--java.awt.event.KeyEvent
Principales métodos
char getKeyChar(), devuelve un carácter asociado con la tecla en este evento.

int getKeyCode(), devuelve el entero que representa al código de tecla asociado
con la tecla en este evento.
```

Un evento de esta clase indica que un *evento de teclado* ha ocurrido en un componente. Este evento de bajo nivel es generado por un objeto componente (tal como un campo de texto) cuando una tecla es presionada, soltada, o tecleada. Este evento se pasa a cada objeto *KeyListener* o *KeyAdapter* que ha sido registrado para recibir tales eventos usando el método *addKeyListene*r. (Los objetos *KeyAdapter* implementan la interfaz *KeyListener.).* Cada tipo de objeto escuchador obtiene este `KeyEvent` cuando un evento ocurre.

Los eventos *"Key typed"* son de un nivel más alto y generalmente no dependen de la plataforma o la capa de teclado. Son generados cuando un carácter es introducido, y es el camino más adecuado para capturar la entrada de caracteres. En el caso más simple, un evento de teclado es producido por una simple tecla presionada, por ejemplo 'a'. A menudo, los caracteres son producidos por series de teclas presionadas, por ejemplo, *'shift' + 'a'*), y el mapeo desde los eventos *key pressed* a los eventos *key typed* deben ser de muchos a uno o de muchos a muchos. Los eventos *key releases* no son usualmente necesarios para generar un evento *key typed*, pero hay algunos casos donde un evento *key typed* no es generado hasta que una tecla es soltada (ejemplo, secuencias de ASCII vía el método Alt-Numpad en Windows). Eventos que no son del tipo *key typed* son generados por teclas que no generan caracteres (ejemplo, teclas de acción, teclas modificadoras, etc.). El método *getKeyChar* siempre devuelve un carácter válido Unicode o CHAR_UNDEFINED. Para eventos *key pressed* y *key released,* el método *getKeyCode*

devuelve el *keyCode* del evento. Para eventos key typed, el método *getKeyCode* siempre devuelve VK_UNDEFINED.

Los eventos *"Key pressed" y "key released"* son de bajo nivel y dependientes de la plataforma y la capa de teclado. Son generados cada vez que una tecla es presionada o soltada, y son el único modo de captar teclas que no generan entrada de caracteres (ejemplo, teclas de acción, teclas modificadoras, etc.). La tecla que está siendo presionada o soltada es indicada por el método *getKeyCode*, que devuelve el *key code virtual*. Los v*irtual key codes* son usados para informar sobre que tecla del teclado ha sido presionada.

Por ejemplo, presionado la tecla Shift se producirá un evento KEY_PRESSED con un keycode o código de tecla VK_SHIFT, mientras presionando la tecla 'a' el resultado será un código de tecla VK_A. Después de que la tecla 'a' es soltada, un evento KEY_RELEASED será lanzado con VK_A. De forma separada, es generado un evento KEY_TYPED con un valor keyChar de 'A'.

Notas:

- Combinaciones de teclas que no producen caracteres, tales como teclas de acción como F1 y la tecla HELP, no generan eventos KEY_TYPED.
- No todos los teclados o sistemas son capaces de generar todos los *virtual key codes*.
- *Virtual key codes* no identifican una tecla física, dependen de la plataforma y el teclado.

Manteniendo la configuración por defecto de los botones laterales, resulta complicado capturar su código asociado, ya que se abren las distintas aplicaciones asociadas y se pierde la visibilidad de la ventana Java. En el caso del botón redondo central, no es posible cambiar su configuración, aunque tampoco es necesario.

Para poder capturar el código de las 4 teclas laterales se ha cambiado su configuración por defecto. A cada una de ellas se le ha asignado la función <<ninguna>>. También es posible hacer que generen un código reconocible si se les asignan direcciones (<<izquierda>>, <<derecha>>, etc) pero en este caso, generarían los mismos códigos que el botón redondo central y se dispondrían de menos opciones para configurar un mayor número de comandos y acciones asociadas a los botones de la PDA. Cada código diferente que pueda ser captado, dará lugar a un comando o acción diferente; por ello, cuantos más códigos diferentes se puedan captar, más acciones podrán iniciarse mediante los botones físicos de la PDA.

Haciendo uso de las clases disponibles en la especificación Java, se ha escrito una aplicación mediante la cual se han obtenido los siguientes códigos:

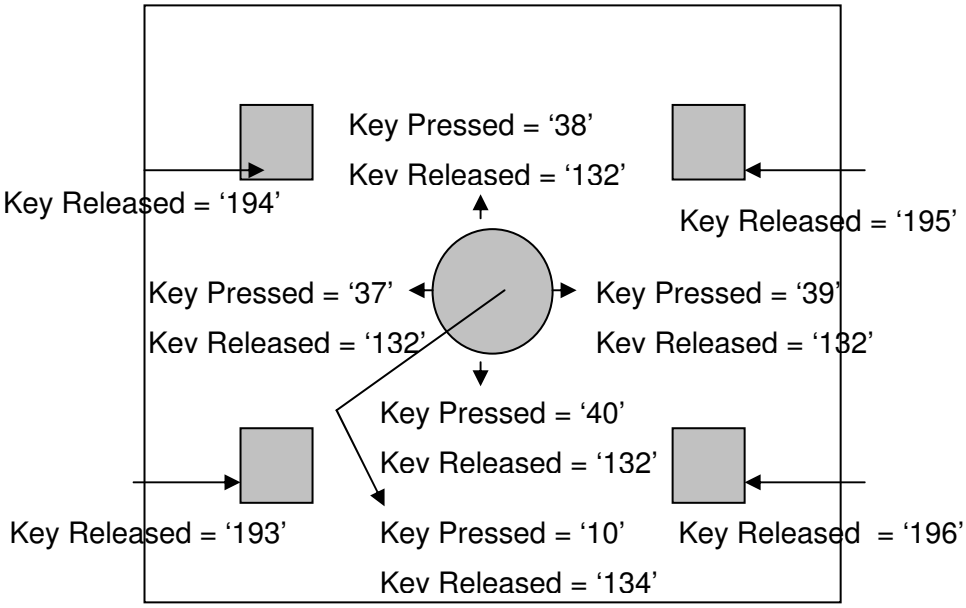


Figura 9. Códigos capturados de los botones de la PDA

El código de la aplicación escrita para capturar el código de los botones es el que sigue:

```
import java.awt.*;
import java.awt.event.*;
class Teclas extends Frame implements KeyListener{
//captura el código de las teclas que se pulsan y lo muestra por pantalla
//junto con el tipo de evento que se ha generado.
 private TextField indicador;
 public Teclas() {
     super("Código Teclas");
     indicador= new TextField("Dice que tecla pulsaste");
     indicador.setEditable(false);
     add(indicador);
     setSize(150,150);
     setVisible(true);
 indicador.addKeyListener(this);
     }
//se deben de implementar todos los métodos de la interfaz KeyListener
 public void keyReleased(KeyEvent e) {
     displayInfo(e, "KEY RELEASED: ");
      }
 public void keyTyped(KeyEvent e) {
 displayInfo(e, "KEY TYPED: ");
      }
```

```
public void keyPressed(KeyEvent e) {
    displayInfo(e, "KEY PRESSED: ");
     }
protected void displayInfo(KeyEvent e, String s){
     int id = e.getID();
     if (id == KeyEvent.KEY_PRESSED) { // presionamos tecla
           int c = e.getKeyCode();
           String sindicador= "key pressed = '" + c + "'";
           indicador.setText(sindicador); // se muestra el código
       } else if (id == KeyEvent.KEY_RELEASED) {//soltamos tecla
           int c = e.getKeyCode();
           String sindicador= "key released = '" + c + "'";
           indicador.setText(sindicador); // se muestra el código
          }else {
           int keyCode = e.getKeyCode();
           String sindicador= "key code = '" + c + "'";
           indicador.setText(sindicador);// se muestra el código
          }
}
public static void main(String args[]){
    Teclas prueba = new Teclas();
    }
}
```

El botón central produce eventos *keyPressed* y *keyReleased*, mientras que los botones laterales únicamente producen eventos *keyReleased*. Estos códigos capturados con el programa serán usados por la aplicación final para diferenciar las distintas acciones a realizar, más concretamente que comandos deben ser enviados (tanto para notificaciones como para comandos de movimiento).

Una vez resueltos estos dos aspectos, comunicación inalámbrica con Java y captura de código de los botones, se ha pasado a implementar distintos módulos software que definen las partes de funcionalidad que debe de agrupar la aplicación final.

6.3.2 Envío de Comandos

Se deben de enviar una serie de comandos desde la PDA, haciendo uso de los botones físicos del iPAQ, al servidor. El envío de comandos se hará efectivo al pulsar los distintos botones físicos de la PDA. Se ha programado una aplicación que simula el comportamiento del envío de comandos, de notificación y posicionamiento, al servidor.

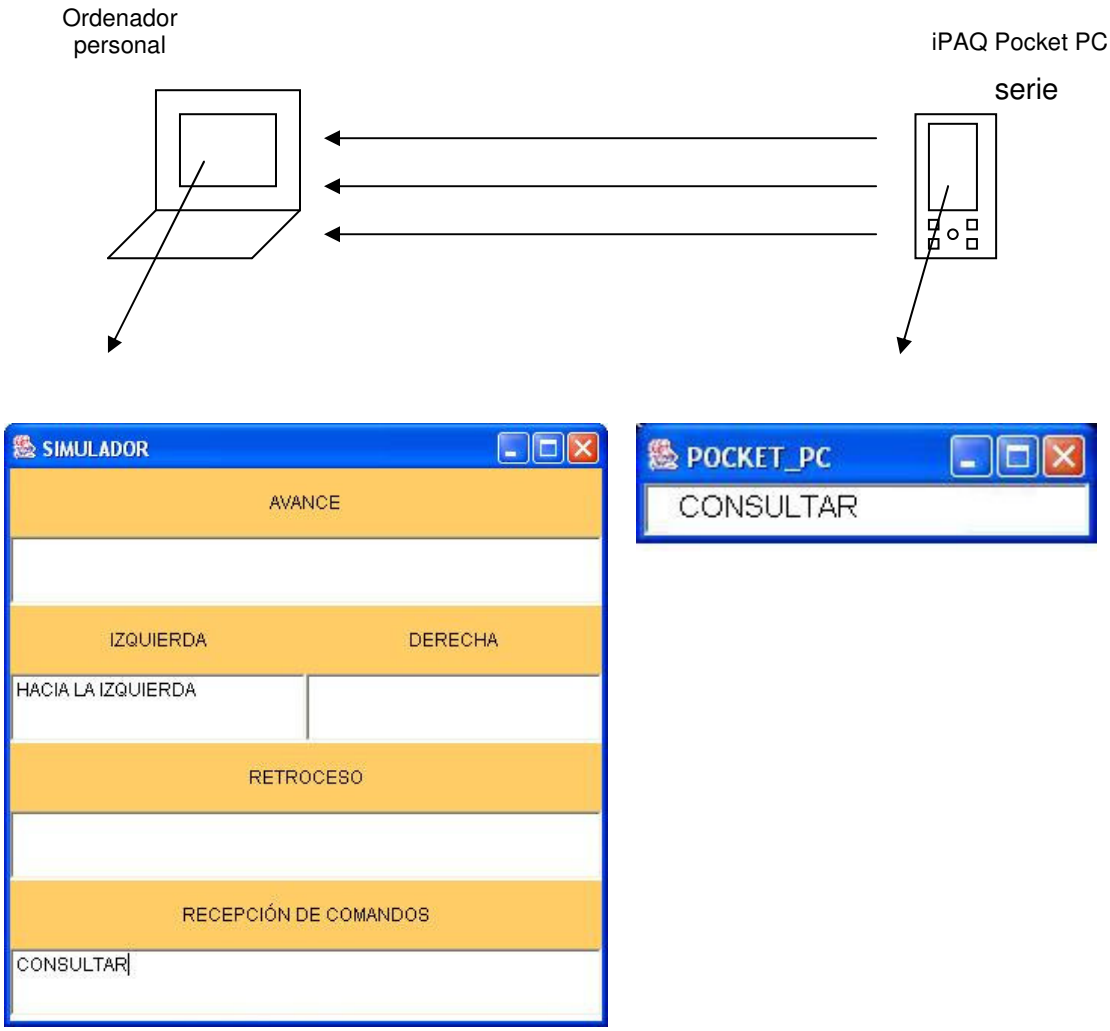


Figura 10. Ejecución de la aplicación de envío de comandos

- Lado servidor: aplicación que reciba comandos y los muestre por pantalla. Al recibir un comando de direccionamiento, su mensaje asociado se muestra en la casilla correspondiente. Cuando la tecla se suelta en el iPAQ Pocket PC, la caja de texto en el lado del servidor se queda en blanco, evidenciando que el avance en esa dirección debe de detenerse.

Cuando se reciben comandos de texto (consultar, configurar…) la palabra permanece en la caja de texto hasta que se recibe el siguiente comando de texto. El código de la clase servidor es el que sigue:

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
import java.lang.*;

public class SimuTeclaPC extends Frame{

   public static final int CENTER =1;
    private Panel p1,p2,p3,p4;
    private Label izquierda,derecha,avance,retroceso,comandos;
    private static TextField izq,der,forward,backward,rxcomandos;
```

```
    public SimuTeclaPC(){
    super("SIMULADOR");

    izquierda=new Label("IZQUIERDA",CENTER);izquierda.setBackground(new
Color(255,204,102));
    derecha=new Label("DERECHA",CENTER);derecha.setBackground(new
Color(255,204,102));
    avance=new Label("AVANCE",CENTER);avance.setBackground(new
Color(255,204,102));
    retroceso= new Label("RETROCESO",CENTER);retroceso.setBackground(new
Color(255,204,102));
    comandos= new Label("RECEPCIÓN DE COMANDOS",
CENTER);comandos.setBackground(new Color(255,204,102));
    setLayout(new GridLayout(4,1));

    p1=new Panel();
    p2=new Panel();
    p3=new Panel();
    p4=new Panel();
    p1.setLayout(new GridLayout(2,1));
    p2.setLayout(new GridLayout(2,2));
    p3.setLayout(new GridLayout(2,1));
    p4.setLayout(new GridLayout(2,1));


    izq=new TextField("",CENTER);izq.setEditable(true);
    der=new TextField("",CENTER);der.setEditable(true);
    forward=new TextField("",CENTER);forward.setEditable(true);
    backward=new TextField("",CENTER);backward.setEditable(true);
    rxcomandos=new TextField("",CENTER);rxcomandos.setEditable(true);

    p1.add(avance);p1.add(forward);
    p2.add(izquierda);p2.add(derecha);p2.add(izq);p2.add(der);
    p3.add(retroceso);p3.add(backward);
    p4.add(comandos);p4.add(rxcomandos);
    add(p1);
    add(p2);
    add(p3);
    add(p4);

    setSize(400,400);
    setVisible(true);
}

    //los comandos se reciben a través de un socket
      public static void recibirComando (Socket s) throws IOException
          {
               int comp=0;
               DataInputStream sIn;
               sIn = new DataInputStream (s.getInputStream() );
               String  comando = sIn.readLine ();
               Integer com=new Integer(comando);
               int comint=com.intValue();
               comp=comint;//los  n° que se reciben tienen asociados un comando
                           //en l lado del servido
               switch(comint){
                   case 1: izq.setText("HACIA LA IZQUIERDA");comp=1; break;
                   case 2: der.setText("HACIA LA DERECHA");comp=2; break;
                   case 3: forward.setText("AVANCE");comp=3; break;
                   case 4: backward.setText("RETROCESO");comp=4; break;
                   case 5: rxcomandos.setText("STOP"); break;

                   case 6: rxcomandos.setText("CONFIGURAR");break;
                   case 7: rxcomandos.setText("CONSULTAR"); break;
                   case 8: rxcomandos.setText("NOTIFICAR"); break;
                   case 9: rxcomandos.setText("SHUTDOWN"); break;
```

```
                case 10:izq.setText("");der.setText("");forward.setText("");
                        backward.setText(""); break;

            }
            sIn.close();//cerramos flujo
            s.close(); // cerramos el socket
  }

 public static void main( String args[] ) {

          SimuTeclaPC prueba = new SimuTeclaPC();
           ServerSocket s = null;
        Socket cliente = null;

        try {
            s = new ServerSocket( 3003 );
        } catch( IOException e ) {
            System.out.println( e );
            }
         while (true) {
           try {
                cliente = s.accept();
               recibirComando(cliente);
           } catch( IOException e ) {
               System.out.println( e );
           }
        }

    }

}
```

- Lado pda: envío de comandos al pulsar las teclas del Pocket PC. En la parte del *iPAQ Pocket PC* se ha implementado un programa que capture el código de las teclas pulsadas y en función de éste, envíe una serie de comandos al servidor. Se ha asignado comandos de direccionamiento a la tecla central, y comandos textuales o notificaciones a las laterales. El código comentado se muestra a continuación.

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
import java.lang.*;

// Socket cliente. Crea un canal de comunicación con el servidor y a través de
//él le envía comandos (generados por las pulsaciones de teclas) que serán
//mostrados en el ordenador servidor.

public class Simuteclas extends Frame implements KeyListener {

    //declaración de variables globales
    private TextField campo;

   //constructor de la clase
    public Simuteclas ()
    {
        super("POCKET_PC");
        //inicializacion de variables globales
        campo = new TextField("INFORMACION DE LOS COMANDOS", 60);
        campo.addKeyListener(this);
        add(campo);
```

```
        //tamaño y visibilidad del Frame
        setSize(230,60);
        setVisible(true);

    }

    //método para capturar eventos de teclas "Released"
    public void keyReleased(KeyEvent e) {
        displayInfo(e, "KEY RELEASED: ");
     }

    //método para capturar eventos de teclas "Type"
    //aunque no lo usemos tenemos que implementarlo ya que
    //ya que la interfaz así lo requiere
    public void keyTyped(KeyEvent e) {
        displayInfo(e, "KEY TYPED: ");
         }

    //método para capturar eventos de teclas "Pressed"
    public void keyPressed(KeyEvent e) {
        displayInfo(e, "KEY PRESSED: ");
         }

    //envía los comandos a través del socket, en función de si el evento
    //ha sido Pressed o Releassed
    protected void displayInfo(KeyEvent e, String s){
         int id = e.getID();
         //presionamos tecla
         //sabemos de antemano que teclas generan cada uno de los códigos
         if (id == KeyEvent.KEY_PRESSED) {
               int c = e.getKeyCode();
               switch (c){
                  case 37: enviarComando("1");campo.setText("ENVIANDO 'MOVE LEFT
ON' COMANDO");break;
                  case 38: enviarComando("3");campo.setText("ENVIANDO COMANDO
'MOVE FORWARD ON'");break;
                  case 39: enviarComando("2");campo.setText("ENVIANDO 'MOVE RIGHT
ON' COMANDO");break;
                  case 40: enviarComando("4");campo.setText("ENVIANDO COMANDO
'MOVE BACKWARD ON'");break;
                  case 10: enviarComando("5");campo.setText("ENVIADO COMANDO
'STOP'");break;
                  default: campo.setText("TECLA SIN COMANDO ASOCIADO");break;
                 }

          } else if (id == KeyEvent.KEY_RELEASED) {//soltamos tecla
              int c = e.getKeyCode();
              switch (c){
                  case 193: enviarComando("6");campo.setText("COMANDO
'CONFIGURACIÓN' ENVIADO");break;
                  case 194: enviarComando("7");campo.setText(" COMANDO 'CONSULTA'
ENVIADO");break;
                  case 195: enviarComando("8");campo.setText(" COMANDO 'NOTIFICAR'
ENVIADO");break;
                  case 196: enviarComando("9");campo.setText("COMANDO 'SHUTDOWN'
ENVIADO ");break;
                  case 132: enviarComando("10");campo.setText(" ");break;
                  default: campo.setText("TECLA SIN VALOR ASOCIADO ");break;
                 }

           }

    }

       //envío de los comandos a través del Socket, serán mostrados por el
simulador
       public void enviarComando(String comando){
```

```
            String servidor = "192.168.0.15";// dirección IP del servidor
            int puerto = 3003 ; // puerto del socket

       try{
              Socket socket= new Socket(servidor , puerto);// creamos el socket
              //creamos un flujo de salida
             PrintWriter salida = new PrintWriter (new
OutputStreamWriter(socket.getOutputStream()),true);
              salida.println(comando); //enviamos el mensaje al servidor
              socket.close(); // cerramos el socket

        } catch (UnknownHostException e )   { e.printStackTrace();
          System.out.println(" Debes estar conectado para que esto funcione
bien");}
          catch (IOException e){  e.printStackTrace();    }

       }

       public static void main( String args[] ) {
              Simuteclas comunica = new Simuteclas();
              }
}
```

6.3.3 Simulación de Sensores

Se trata de que el operador humano disponga sobre la pantalla del dispositivo móvil, de la información actualizada correspondiente a los valores que proporcionan los sensores ópticos. De este modo, puede saber a que distancia se encuentra, y hacer las oportunas modificaciones del posicionamiento.

- Lado servidor: se generan aleatoriamente valores de los sensores y se envían al cliente. Este programa genera aleatoriamente valores para los cuatro sensores. Un quinto sensor podría ser incluido en versiones posteriores como media de los cuatro anteriores. Para cada sensor, se pueden generar tres valores: una distancia de 1'5 metros, una distancia de 0'80 m de la superficie del casco del barco, y 0'40 m de la superficie. Para esta última distancia, el cliente deberá de activar algún mecanismo de aviso, ya que se considera distancia critica de [0-0,4] metros de la superficie. Para cada uno de los sensores los valores aleatorios deben de mantener una lógica de variación. Si en un momento dado un sensor tiene un valor de 0'40 m, para el instante siguiente, la distancia puede ser de 0'40 (el eje representado permanece en la misma posición) o descender a 0'80; no es posible que pase directamente a 1'50 metros. El parámetro que determina la activación de la alarma visual puede pasarse por la línea de comandos en la consola, ya que dependiendo de la tarea, la distancia de seguridad puede variar.

- Lado Cliente (PDA): se reciben esos valores y se muestran gráficamente en pantalla en tiempo real: diagramas de barras. La actualización de los valores se hace cada 0.25 segundos. Una barra es dibujada para cada uno de los sensores. La información recibida también se muestra numéricamente. Mientras los sensores detecten una distancia más lejana de un metro, no se mostraran las barras. Es a partir de 1 metro de distancia respecto a la superficie del buque, cuando los valores recibidos se representan gráficamente. Cuando los sensores detectan una distancia menor o igual a 0.4 metros de la superficie del barco, se pintará una línea

roja de seguridad, que alertará al operador de la proximidad de las extremidades del robot a la superficie.

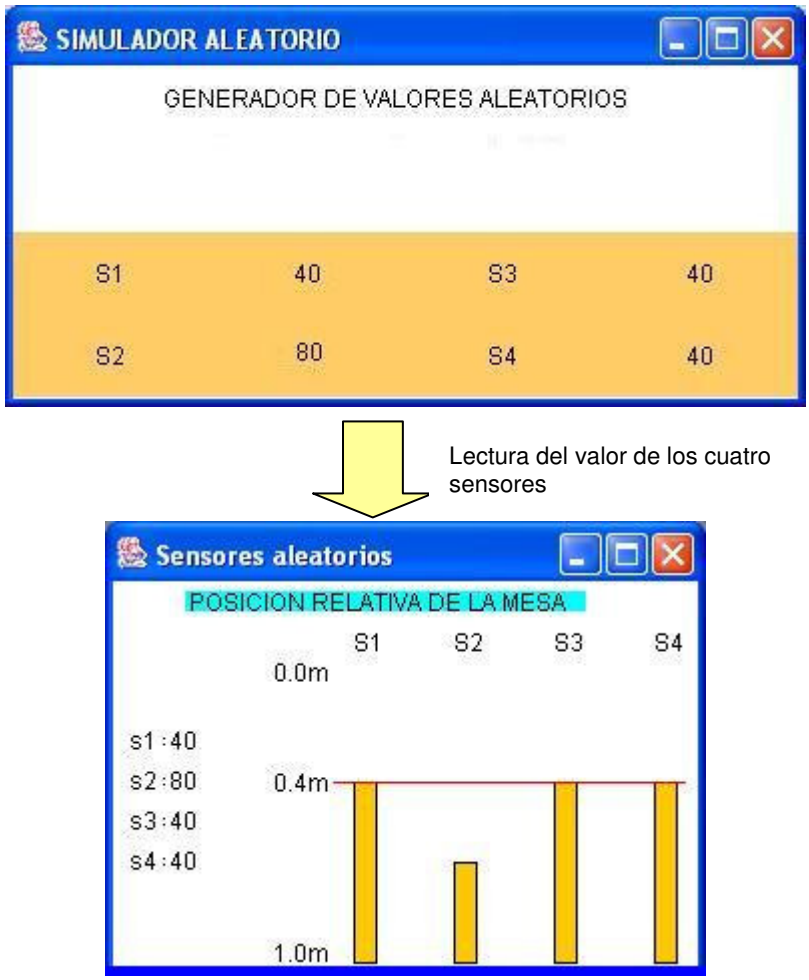


Figura 11. Captura de pantallas durante la simulación del programa de monitorización de la información de los sensores.

El código de ambas clases es el siguiente:

Clase servidor

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
import java.lang.*;

//  FUNCIÓN DE ESTA CLASE:
//  DE MANERA ALEATORIA GENERAMOS POSIBLES VALORES DE LOS SENSORES.
//  ESTOS VALORES LOS USARA LA CLASE QUE SE EJECUTA EN EL POCKET PC
//  PARA INCREMENTAR O DECREMENTAR LAS BARRAS CORESPONDIENTES A CADA SENSOR

public class ServerAlea extends Frame implements Runnable{


 // SE DEFINEN LAS VARIABLES GLOBALES
   public static final int CENTER =1;
   private Panel p1,p2;
```

```
private Label s1, s2, s3, s4, valors1, valors2, valors3, titulo,
explicación;
private Vector vector;
private double alea;
private static int sensor1,sensor2,sensor3,sensor4;
private Thread hilo;
private long espera;
private  ServerSocket s = null;
private  Socket cliente = null;


//CONSTRUCTOR DE LA CLASE
public ServerAlea(){

super("SIMULADOR ALEATORIO");
espera= 255;
titulo= new Label("GENERADOR DE VALORES ALEATORIOS");
explicacion= new Label("Envía las actualizaciones cada 0.25 seg");
valors1= new Label("    ",CENTER);valors1.setBackground(new
Color(255,204,102));
valors2= new Label("    ",CENTER);valors2.setBackground(new
Color(255,204,102));
valors3= new Label("    ",CENTER);valors3.setBackground(new
Color(255,204,102));
valors4= new Label("    ",CENTER);valors4.setBackground(new
Color(255,204,102));
vector= new Vector(4);
vector.add(0,new Integer(sensor1));
vector.add(1,new Integer(sensor2));
vector.add(2,new Integer(sensor3));
vector.add(3,new Integer(sensor4));

try{
      s = new ServerSocket( 3000);

       } catch( IOException e ) {
          System.out.println( e );
          }

//lanzamos el hilo
hilo= new Thread(this);
hilo.start();


//CREAMOS LAS ETIQUETAS DEL MARCO
s1=new Label("S1",CENTER);
s1.setBackground(new Color(255,204,102));
s2=new Label("S2",CENTER);
s2.setBackground(new Color(255,204,102));
s3=new Label("S3",CENTER);
s3.setBackground(new Color(255,204,102));
s4= new Label("S4",CENTER);
s4.setBackground(new Color(255,204,102));

setLayout(new GridLayout(2,1));

p1=new Panel();
p2=new Panel();
p2.setLayout(new GridLayout(2,4));

//AÑADIMOS LOS COMPONENTES LOS PANELES
p1.add(titulo); p1.add(explicacion);
p2.add(s1);p2.add(valors1);p2.add(s3);p2.add(valors3);
p2.add(s2);p2.add(valors2);p2.add(s4);p2.add(valors4);

//AÑADIMOS LOS PANELES AL FRAME
add(p1);
add(p2);
```

```
//TAMAÑO Y VISIBILIDAD DEL FRAME
setSize(400,400);
setVisible(true);
}

public void run(){
while(true){
generateValue();
try {
    sensores();
    } catch( IOException io ) {System.out.println( io ); }

//Aquí debemos de parar el thread 0.25 segundos y volver a activarlo después
try{
   hilo.sleep(espera,0);
   }catch(InterruptedException iie){ System.err.println(iie);
      }
   }
  }

public void generateValue(){

   alea= Math.random();
   if (alea<=0.4){
     if (getSensor1()<=80){
         setSensor1(40);
         vector.set(0,new Integer(sensor1));
         valors1.setText(String.valueOf(sensor1));}
       else { setSensor1(80);
              vector.set(0,new Integer(sensor1));
               valors1.setText(String.valueOf(sensor1));
              }
             }
      else if(alea>=0.8){
             setSensor1(80);
             vector.set(0,new Integer(sensor1));
             valors1.setText(String.valueOf(sensor1));
           }
               else {
                     if (getSensor1()==40){
                     setSensor1(80);
                     vector.set(0,new Integer(sensor1));
                     valors1.setText(String.valueOf(sensor1));
                     }else{
                     setSensor1(150);
                         vector.set(0,new Integer(sensor1));
                          valors1.setText(String.valueOf(sensor1));
                         }
                   }

   alea= Math.random();
   if (alea<=0.4){

             if (getSensor2()<=80){
             setSensor2(40);
             vector.set(0,new Integer(sensor2));
             valors2.setText(String.valueOf(sensor2));}
             else { setSensor2(80);
             vector.set(0,new Integer(sensor2));
             valors2.setText(String.valueOf(sensor2));
                 }
               }
      else if(alea>=0.8){
             setSensor2(80);
             vector.set(0,new Integer(sensor2));
             valors2.setText(String.valueOf(sensor2));
           }
```

```
                else {
                       if (getSensor2()==40){
                       setSensor2(80);
                       vector.set(0,new Integer(sensor2));
                       valors2.setText(String.valueOf(sensor2));
                           }else{
                      setSensor2(150);
                       vector.set(0,new Integer(sensor2));
                       valors2.setText(String.valueOf(sensor2));
                              }
                    }
    alea= Math.random();
    if (alea<=0.4){
                if (getSensor3()<=80){
                setSensor3(40);
                vector.set(0,new Integer(sensor3));
                valors3.setText(String.valueOf(sensor3));}
                else { setSensor3(80);
                vector.set(0,new Integer(sensor3));
                valors3.setText(String.valueOf(sensor3));
                       }
                 }
       else if(alea>=0.8){
               setSensor3(80);
               vector.set(0,new Integer(sensor3));
               valors3.setText(String.valueOf(sensor3));
             }
                else {
                       if (getSensor3()==40){
                       setSensor3(80);
                       vector.set(0,new Integer(sensor3));
                      valors3.setText(String.valueOf(sensor3));
                           }else{
                       setSensor3(150);
                      vector.set(0,new Integer(sensor3));
                      valors3.setText(String.valueOf(sensor3));
                              }
                    }
    alea= Math.random();
    if (alea<=0.4){
                if (getSensor4()<=80){
                setSensor4(40);
                vector.set(0,new Integer(sensor4));
                valors4.setText(String.valueOf(sensor4));}
                else { setSensor4(80);
                vector.set(0,new Integer(sensor4));
                valors4.setText(String.valueOf(sensor4));
                       }
                 }
       else if(alea>=0.8){
               setSensor4(80);
               vector.set(0,new Integer(sensor4));
               valors4.setText(String.valueOf(sensor4));
             }
                else {
                       if (getSensor4()==40){
                       setSensor4(80);
                       vector.set(0,new Integer(sensor4));
                       valors4.setText(String.valueOf(sensor4));
                           }else{
                       setSensor4(150);
                    vector.set(0,new Integer(sensor4));
                      valors4.setText(String.valueOf(sensor4));
                              }
                    }
    }

//MÉTODO QUE NOS PERMITE ENVIAR LOS 4 VALORES DE LOS SENSORES
```

```
   //ENCAPSULADOS EN UN ÚNICO OBJETO VECTOR
   public void sensores() throws IOException
        {

         try{
                cliente=s.accept();

         }catch(IOException e){System.out.println(e);}

         try{

                ObjectOutputStream out = new
ObjectOutputStream(cliente.getOutputStream());
                out.writeObject(vector);
                out.flush();
                out.close();
                }catch(Exception e){System.out.println(e.getMessage());  }
         }


   //MÉTODOS PARA OBTENER LOS VALORES DE LOS SENSORES
   public int getSensor1(){
        return sensor1;
        }

   public  int getSensor2(){
       return sensor2;
     }

   public int getSensor3(){
      return sensor3;
     }

   public int getSensor4(){
      return sensor4;
     }

   // METODOS PARA ACTUALIZAR EL VALOR DE LOS SENSORES
   public void setSensor1(int data){
       sensor1= data;
     }

   public  void setSensor2(int data){
       sensor2=data;
     }

   public void setSensor3(int data){
      sensor3=data;
       }

   public void setSensor4(int data){
      sensor4=data;
       }

   public static void main(String args[]){

          ServerAlea prueba = new ServerAlea();

       }


     }
```

Clases en el cliente:

```
import java.awt.*;

// ESTA CLASE SE ENCARGA DE PINTAR LOS GRÁFICOS QUE SE MUESTRAN EN EL Ipaq.
```

```
// MUESTRA UNA BARRA PARA CADA SENSOR Y ADEMÁS UNA LÍNEA QUE INDICA SI HEMOS
// SUPERADO LA DISTANCIA DE "SEGURIDAD".

class Barraipaq extends Panel {

//cuando creamos el objeto "barratest" le pasamos las alturas (valor de los
//sensores)como parámetros, y pintamos según estos. Cada vez que queramos
//repintar, actualizamos primero los valores de los sensores con el método
//renovar.

private int s1,s2,s3,s4;

//CONSTRUCTOR POR DEFECTO DE LA CLASE
public Barraipaq(){
   s1=0;s2=0;s3=0;s4=0;
}
//CONSTRUCTOR DE LA CLASE
public Barraipaq(int a, int b, int c, int d){

   s1=a;s2=b;s3=c;s4=d;
 }


// MÉTODO QUE PINTA LOS GRÉFICOS
public void paint( Graphics g) {

     // drawRect(int x,int y,int width, int height);
     g.drawString("0.0m",10,50);
     g.drawString("0.4m",10,105);
     g.drawString("1.0m",10,190);

    g.drawString("S1",50,35);
    g.setColor(Color.orange);
    g.fillRect(50,190-s1,11,s1);
    g.setColor(Color.black );
    g.drawRect(50,190-s1,11,s1);

    g.drawString("S2",90,35);
    g.setColor(Color.orange);
    g.fillRect(90,190-s2,11,s2);
    g.setColor(Color.black );
    g.drawRect(90,190-s2,11,s2);

    g.drawString("S3",130,35);
    g.setColor(Color.orange);
    g.fillRect(130,190-s3,11,s3);
    g.setColor(Color.black );
    g.drawRect(130,190-s3,11,s3);

    g.drawString("S4",170,35);
    g.setColor(Color.orange);
    g.fillRect(170,190-s4,11,s4); //cuando sumamos 'a' unidades a 'height'-->
restamos
    g.setColor(Color.black );//"a" unidades a la Y, ya q sino , lo pinta
                                 //hacia abajo
    g.drawRect(170,190-s4,11,s4);

    //comprobamos si debemos de pintar la línea de seguridad o no
     if(s1>=60 || s2>=60 || s3>=60 ||s4>=60){
           g.setColor(Color.red);
          //drawLine(int x1,inty1,int x2, int y2)
           g.drawLine(50,100,225,100);
          }
        }

public void renovar(int a,int b, int c, int d){
             s1=a;s2=b;s3=c;s4=d;
```

```
            }

  }
```

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
import java.lang.*;

// ESTA CLASE CORRE SOBRE EL POCKET PC Y DEBE DE REPRESENTAR GRÁFICAMENTE
// EL VALOR DE LOS SENSORES. LA ACTUALIZACIÓN DE LOS VALORES SERÁ CADA 0.25
SEGUNDOS
class Clientipaq extends Frame {

//DECLARACION DE VARIABLES GLOBALES
 private Barraipaq barra; // objeto que pinta los gráficos
 private Label mensaje;//mensaje de cabecera
//indica a que coordenada corresponde el valor
 private Label pos1,pos2,pos3,pos4;
//muestran los valores de las coordenadas

 private Label posicions1,posicions2,posicions3,posicions4;
 private int s1,s2,s3,s4;//valores de los sensores
 private String servidor;
 private int puerto;
 private Vector vectorcliente;

 //constructor de la clase
 public Clientipaq(){
       super("Sensores ópticos");

       servidor = "80.32.201.78";// dirección IP del servidor
       puerto = 3000 ; // puerto del socket
        vectorcliente=new Vector(4);

       barra= new Barraipaq();

       pos1= new Label("s1 = ");
       pos1.setLocation(30,240);
       pos1.setSize(20,20);

       pos2= new Label("s2 = ");
       pos2.setLocation(30,260);
       pos2.setSize(20,20);

       pos3= new Label("s3 = ");
       pos3.setLocation(120,240);
       pos3.setSize(20,20);

       pos4= new Label("s4 = ");
       pos4.setLocation(120,260);
       pos4.setSize(20,20);

       //muestran los valores actuales de los sensores
       posicions1=new Label(String.valueOf(s1));
       posicions1.setLocation(50,240);
       posicions1.setSize(30,20);

       posicions2=new Label(String.valueOf(s2));
       posicions2.setLocation(50,260);
       posicions2.setSize(30,20);

       posicions3=new Label(String.valueOf(s3));
       posicions3.setLocation(140,240);
       posicions3.setSize(30,20);
```

```
        posicions4=new Label(String.valueOf(s4));
        posicions4.setLocation(140,260);
        posicions4.setSize(30,20);

        mensaje= new Label (" POSICION DE LA MESA");
        mensaje.setLocation(20,25);
        mensaje.setSize(150,10);
        mensaje.setBackground(Color.cyan);

        //añadimos  los componentes al Frame
        add(mensaje);
        add(pos1);add(pos2);add(pos3);add(pos4);
        add(posicions1); add(posicions2); add(posicions3);add(posicions4);
        add(barra);

        //tamaño y visibilidad del Frame
        setSize(200,300);
        setVisible(true);

        }

//LEEMOS LOS VALORES DE LOS SENSORES Y ACTUALIZA LOS GRAFICOS
public void recibir(){

        try{
            // creamos el socket
            Socket cliente= new Socket(servidor,puerto);

            try{

                ObjectInputStream in= new
                ObjectInputStream(cliente.getInputStream());
                vectorcliente=(Vector)in.readObject();
                in.close();

                }catch(Exception e){System.out.println(e.getMessage());  }

            Integer sensor1=(Integer)vectorcliente.elementAt(0);
            s1=sensor1.intValue();

            Integer sensor2=(Integer)vectorcliente.elementAt(1);
            s2=sensor2.intValue();

            Integer sensor3=(Integer)vectorcliente.elementAt(2);
            s3=sensor3.intValue();

            Integer sensor4=(Integer)vectorcliente.elementAt(3);
            s4=sensor4.intValue();

            //mostramos tambien numericamente el valor de los sensores
             posicions1.setText(String.valueOf(s1));
             posicions2.setText(String.valueOf(s2));
             posicions3.setText(String.valueOf(s3));
             posicions4.setText(String.valueOf(s4));

             if(s1>100)s1=0;if(s1==40)s1=90;if(s1==80)s1=50;
             if(s2>100)s2=0;if(s2==40)s2=90;if(s2==80)s2=50;
             if(s3>100)s3=0;if(s3==40)s3=90;if(s3==80)s3=50;
             if(s4>100)s4=0;if(s4==40)s4=90;if(s4==80)s4=50;
            //actualizamos los graficos
            barra.renovar(s1,s2,s3,s4);
            barra.repaint();
            //cerramos el socket
            cliente.close();

         } catch (UnknownHostException e )
           { e.printStackTrace();
```

```
           System.out.println(" Debes estar conectado para que esto funcione
bien");
           }catch (IOException e){e.printStackTrace();    }

       }

public void update( Graphics g ) {
       paint( g );
    }

public static void main( String args[] ) {
      Clientipaq prueba = new Clientipaq();
       while(true){
              prueba.recibir();
           }
       }

 }
```

6.3.4 Captura de Vídeo

Debido a las limitaciones del sistema operativo instalado en el iPAQ Pocket PC, Windows CE, no es posible mostrar simultáneamente sobre la pantalla de la PDA una aplicación gráfica Java y la captura de vídeo (mediante IA Capture). Una vez iniciada la captura de vídeo, al poner la aplicación Java en primer plano, el entorno de IA Capture permanece en pantalla, pero su funcionamiento se detiene.

Se pretende poder acceder a la captura de vídeo mientras la aplicación Java se está ejecutando. De este modo, el operador puede disponer de todos los recursos de manera conjunta; es decir, podría por ejemplo ver la evolución de los sensores que muestra la aplicación Java y conmutar a modo captura de vídeo sin tener que cerrar o reiniciar la aplicación.

La aplicación Java sigue funcionando con normalidad aunque esté en segundo plano. Por lo tanto, los valores de los sensores se irán actualizando en la PDA aunque en ese momento se esté en modo captura de vídeo.

La solución planteada para compatibilizar la ejecución de Java y el programa de captura de vídeo IA Capture es la planteada a continuación. La figura muestra la asignación de los botones a realizar.

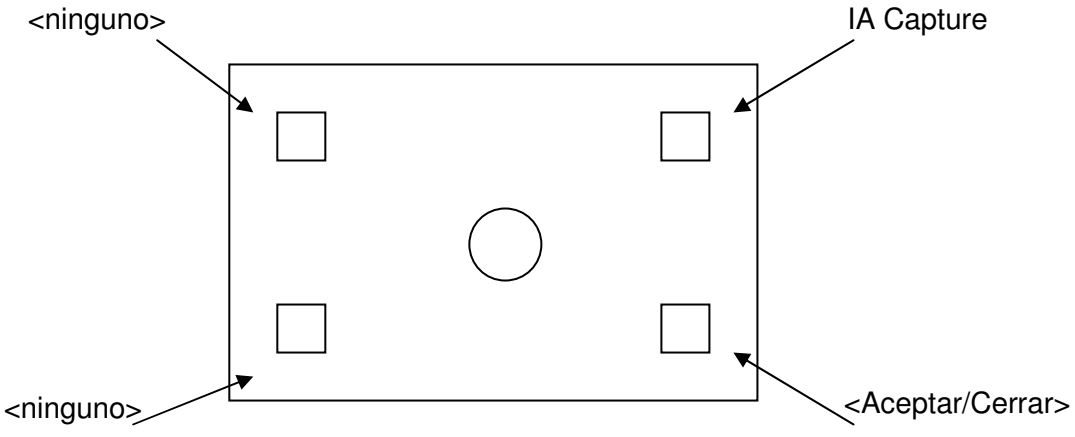


Figura 12. Asignación de los botones para captura de vídeo y ejecución de la aplicación Java simultáneamente.

A uno de los botones le asignamos la aplicación de captura de vídeo *'IA Capture'*. Este botón quedará inhabilitado para envío de comandos al sistema de monitorización. Cada vez que sea pulsado se abrirá *IA Capture*, permitiendo la visualización de vídeo procedente de la cámara conectada a la PDA. A otro de los botones se le asigna la función de <Aceptar/Cerrar>. Si la aplicación Java se encuentra en primer plano, por ejemplo el operador está visualizando las barras que representan el valor de los sensores, este botón puede ser usado para el envío de comandos. Si por el contrario, se esta en modo captura de vídeo, su pulsación supone el cierre de IA Capture y el regreso al entorno gráfico que se estaba visualizando anteriormente.

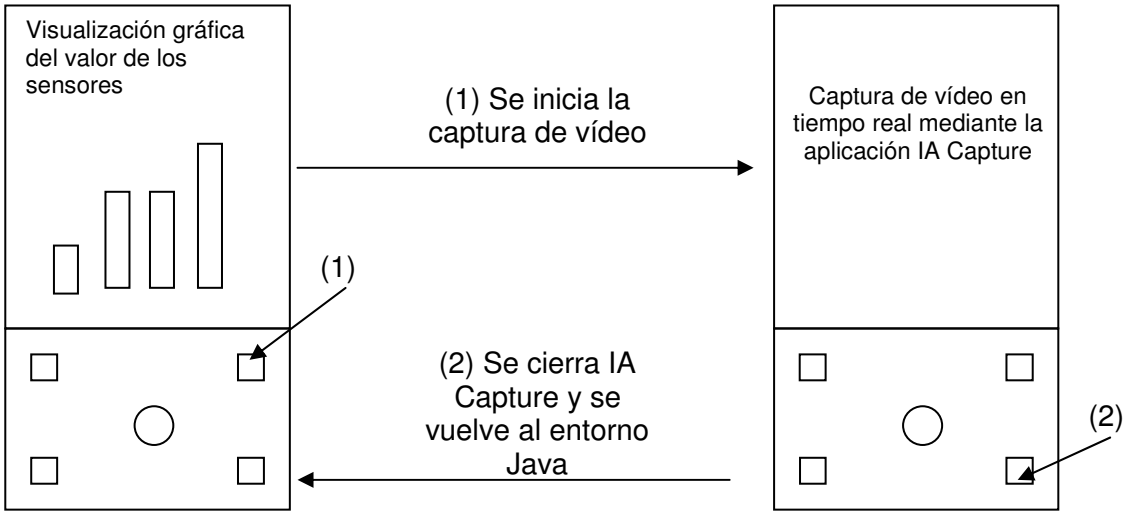


Figura 13. Simulación del funcionamiento de la configuración para mostrar vídeo

6.3.5 Implementación de un Historial de la Jornada

Una cuestión interesante a implementar sería poder almacenar en un fichero el historial de comandos que se han ejecutado a lo largo de una jornada. A continuación se muestra el código de un programa que almacena en un archivo de texto en la PDA, un determinado comando y la hora a la que se envió. El archivo es almacenado en el directorio donde se encuentre el programa Java.

```
import java.io.*;
import java.util.*;

//para escribir datos en un fichero:
// 1. Definimos un flujo hacia el fichero en el que deseamos escribir datos
//2.leemos o definimos los datos a escribir en el fichero
//3. Cerramos el flujo

public class LogiPAQ{

    public static void crearFichero(File fichero) throws IOException
    {
    Date horaT = new Date();
    DataOutputStream dos= null;// salida de datos hacia el fichero
    try
```

```
    {
        //crear un flujo hacia el fichero que permita escribir
        //datos de tipos primitivos y que utilice un buffer
        dos = new DataOutputStream (new BufferedOutputStream(new
FileOutputStream(fichero)));
        //Declarar los datos a escribir en el fichero (comando y hora en la que el
comando se envió
        String hora=horaT.toString();
        String comando= "Confirmar";
        String nombre= comando.concat(hora);
        dos.writeUTF(nombre);

    }

        finally
        {
            // cerrar el flujo
            if(dos!=null)dos.close();
        }
}

public static void main(String[] args){

    String nombreFichero= null; // nombre del fichero
    File fichero = null; // objeto que identifica el fichero
    try
    {
        nombreFichero= "LogiPAQ";
        fichero= new File(nombreFichero.concat(".txt"));
        crearFichero(fichero);
        }catch(IOException e){System.out.println("Error: " + e.getMessage()); }
}
}
```

6.4 Implementación del Prototipo de Aplicación de Control a Ejecutar sobre la PDA

6.4.1 Introducción

En este apartado se muestra la implementación del sistema desarrollado. En primer lugar se presentan los entornos gráficos correspondientes a los dos subsistemas, servidor y cliente A continuación se mostrarán las imágenes correspondientes a una ejecución de la aplicación, mostrando cada una de las funciones implementadas.

En el lado servidor se ejecuta un simulador, que emula el funcionamiento del sistema de monitorización. Además genera valores aleatorios imitando el comportamiento de los sensores ópticos colocados en la mesa robótica.

El entorno gráfico es el que se muestra en la figura 14. El interfaz se divide en tres partes, mostrando cada una de ellas diferentes tipos de informaciones que se envían o reciben en la comunicación con el sistema implantado en la PDA.

El simulador comienza a generar valores aleatorios para los sensores en el momento en el que el cliente se conecta a él. Una vez establecida la comunicación, los valores son periódicamente generados y enviados a la aplicación que se ejecuta en la PDA. Los valores generados se muestran en el interfaz del simulador.

Respecto al posicionamiento, el simulador muestra los comandos enviados por el operador a través de la PDA. Para cada una de las coordenadas (x,y,z), el interfaz mostrará los mensajes de “Forward” o “Backward” que se reciben. Mientras se reciba la orden, el texto se mostrará debajo de la coordenada correspondiente.

Figura 14. Entorno gráfico del simulador ejecutado sobre un PC

Las notificaciones enviadas desde la PDA son mostradas en una caja de texto. Una vez recibida una notificación, ésta permanece visible en el interfaz hasta que se reciba otro mensaje, siendo sustituida en la caja de texto por éste último.

El menú principal de la aplicación que se va a ejecutar sobre la PDA es el que se muestra en la siguiente figura.

Figura 15. Interfaz de la aplicación a ejecutar sobre la PDA

Desde el menú principal es posible acceder a tres funciones básicas. Mediante un menú desplegable se accede a los submenús de Notificar (funciones de envío de comandos y recepción de información de tareas) y Posicionar (funciones de visualización de los valores de los sensores y envío de comandos de posicionamiento). Mediante los botones de la PDA situados debajo de las etiquetas respectivas, se habilita la apertura o cierre de la aplicación que muestra el vídeo en tiempo real.

6.4.2 Ejecución de la Aplicación

En los siguientes apartados se irán mostrando las distintas pantallas que aparecen en una ejecución de la aplicación.

6.4.2.1 Consulta de Tarea

Una vez arrancadas aplicación cliente y servidor, se comienza consultando las tareas a realizar. Desde el menú principal, se selecciona el submenú Notificar, y dentro de éste la opción Consultar. En este momento se establece una comunicación vía socket con el servidor, y la información recibida se muestra en pantalla. Para regresar al menú principal, pulsar el botón Menú.

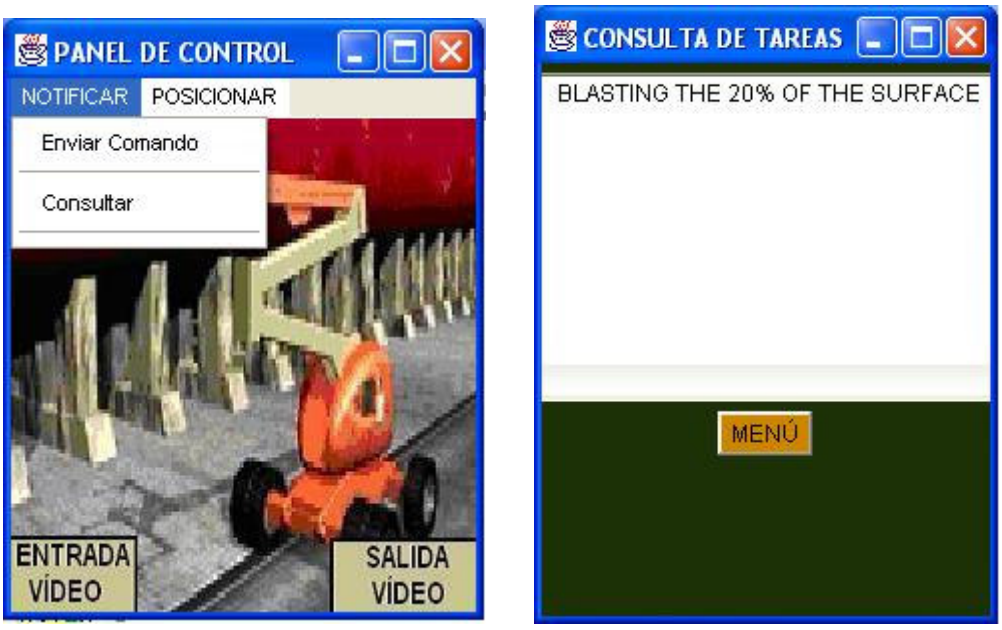


Figura 16. Consulta de tareas desde la PDA

6.4.2.2 Envío de Comandos

Para el envío de comandos de texto se accede desde el menú principal a través del submenú Enviar Comando. Es entonces cuando se muestra una pantalla con los distintos comandos disponibles. Cada vez que se seleccione un comando se requiere una confirmación de envío del mismo. En caso de cancelar, se regresa a la pantalla de Notificaciones, y ninguna información es enviada al servidor. En caso de aceptar el envío se establece un socket con el servidor y se envía la información correspondiente. El mensaje enviado se muestra en el servidor. Para cada comando enviado, se escribe un registro en un archivo log en la PDA. Los campos del archivo son: tipo comando, fecha de envío.

Como comando especial se encuentra el de Shutdown. El envío de este comando significa el cierre de la aplicación ejecutada sobre la PDA.

Para volver al menú principal, se debe pulsar el botón Menú.

Como ya se comentó anteriormente, el mensaje o comando recibido permanece visible en la pantalla del simulador hasta que sea recibido otro que lo sustituya.

Figura 17. Envío de comandos desde la PDA

Figura 18. Recepción de comandos textuales procedentes de la PDA

6.4.2.3 Recepción del Valor de los Sensores y Envío de Comandos de Posicionamiento

La función de posicionamiento tiene dos vertientes. Para acceder a ambas se debe seleccionar desde el menú principal el submenú Posicionar. Una vez seleccionado, se entra a la pantalla de visualización del valor de los sensores, a la vez que el sistema queda a la escucha de los eventos de teclas que provocan el envío de comandos de posicionamiento.

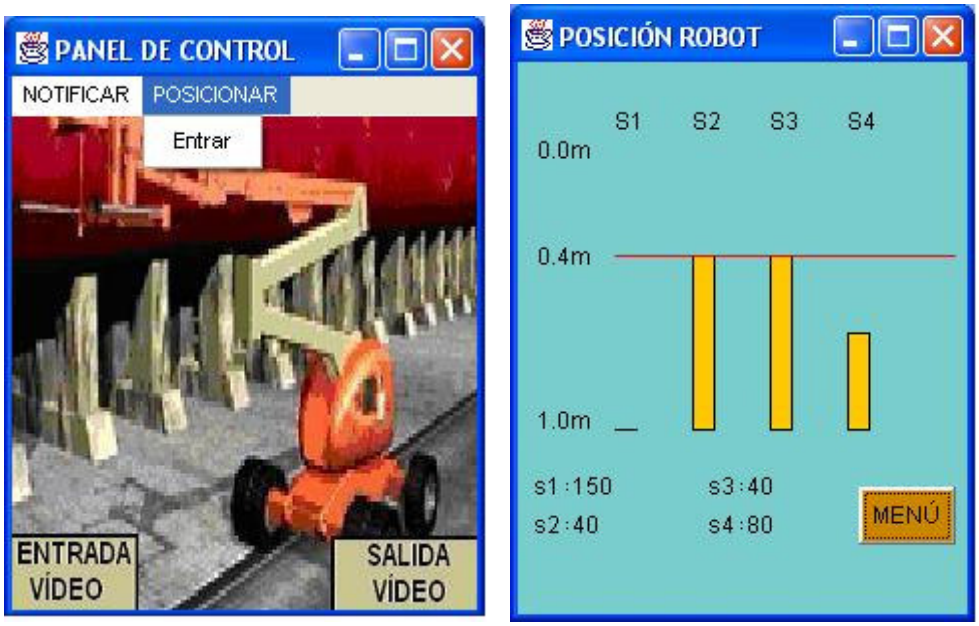


Figura 19. Lectura y representación de los valores de los sensores

Tanto desde el menú principal, como estando dentro de los submenús, es posible la visualización del vídeo en tiempo real. Una vez cerrada la aplicación de vídeo, se regresa a la pantalla desde la cual se la llamó. Estando dentro del menú Posicionar, si la aplicación de vídeo es abierta, se continúa la lectura del valor de los sensores; así como el envío de los comandos de posicionamiento que sean seleccionados.

Los sensores toman de manera aleatoria los valores de 40, 80 o 150. Estos valores representan la distancia en centímetros de la mesa robótica respecto de la superficie del barco. Cuando alguno de los sensores detecta una distancia igual o menor a 40 centímetros, aparece sobre la pantalla una línea roja alertando de la proximidad a la superficie. El valor de cada sensor es representado de ambos modos, gráficamente y de textualmente.

Para volver al menú principal, pulsar el botón Menú.

Los comandos son mostrados mientras la tecla de la PDA correspondiente a su envío permanezca presionada. Una vez la tecla deja de ser pulsada, la etiqueta en el simulador correspondiente a esa coordenada queda vacía.

Figura 20. Visualización de los valores de sensores generados y de los comandos de posicionamiento recibidos

Los comandos recibidos para cada coordenada pueden ser:

X : Forward / Backward

Y: Forward / Backward

Z: Forward / Backward

6.5 Problemas de Implementación

6.5.1 Vídeo

El vídeo es capturado mediante la aplicación *IA Capture* de la utilidad *IA Style PowerMedia Suite*. Esta aplicación es suministrada junto con el módulo de expansión multimedia *FlyJacket i3800 de LifeView*. Cuando la aplicación es abierta tras la pulsación del botón asociado a esta función, el vídeo es mostrado en tiempo real, y a pantalla completa, por lo que la aplicación Java deja de ser visible. Mientras este programa se está ejecutando, es posible la lectura del valor de los sensores y el envío de los comandos de posicionamiento que sean seleccionados. No es posible mostrar en primer plano el interfaz Java mientras la entrada de vídeo sigue activa. En el momento en que Java pasa a primer plano, la entrada de vídeo se detiene.

6.5.2 Refresco de la Pantalla

Se han observado algunos problemas a la hora de refrescar la imagen en la pantalla de la PDA cuando se trata de representar gráficos en movimiento. Para representar el valor de los sensores gráficamente es necesario parar y reactivar continuamente el Thread de la clase encargada de dicha función. Además en la clase contenedora del gráfico se incluye el método:

```
public void update( Graphics g ) {
      paint( g );
            }
```

También es conveniente, cuando se arranca la aplicación, hacer uso de la opción:

-Dsun.java2d.noddraw=true

La ejecución sería como sigue:

-cp \ -Dsun.java2d.noddraw=true MenuPrincipal

Tomando estas medidas la representación del movimiento de los gráficos es correcta.

6.5.3 Botones de la PDA

Para el envío de comandos de posicionamiento se emplean las dos teclas inferiores de los laterales y el botón redondo central. Al presionar el botón central en cada una de sus direcciones, son capturados los eventos de KeyPressed y KeyReleased. De este modo, se puede implementar el envío del comando durante todo el tiempo que sea pulsado, y deteniéndose el movimiento cuando se capture el evento KeyReleased, o soltar tecla. Por el contrario, las teclas laterales únicamente producen los eventos de KeyReleased, por lo que el comando solo se enviará cuando la tecla sea soltada. Esto obliga a que uno de los ejes se mueva por impulsos, los otros dos pueden accionarse de manera continuada mientras se mantenga pulsada la tecla. La opción propuesta adjudica al botón central las coordenadas ‘x’ e ‘y ’. La coordenada ‘z’ es controlada mediante los botones laterales inferiores. El modo de interpretar estos comandos queda en manos de la aplicación servidor. Así por ejemplo el envío de un comando ‘z avanza’ puede ser interpretado por el sistema de monitorización como un avance de 10 cm en profundidad o únicamente de 1 cm.

Capítulo 7
Conclusiones y Trabajos Futuros

7.1 Conclusiones del Proyecto

A lo largo de este proyecto se han estudiado y desarrollado una serie de cuestiones que han servido posteriormente para realizar el diseño e implementación de una aplicación para el control teleoperado de un robot.

La aplicación diseñada se ubica dentro del marco del sistema EFTCoR. Los subsistemas componentes del sistema EFTCoR han sido explicados, de modo que las relaciones existentes entre éstos y el sistema ejecutado en la PDA queden claramente identificados.

El uso de una PDA como plataforma de ejecución de la aplicación de control diseñada, invita a la reflexión sobre el uso de este tipo de dispositivos como soporte de aplicaciones de uso industrial. En este caso se ha utilizado para control de un sistema robótico. El desarrollo en los últimos años de este tipo de dispositivos digitales los han dotado de unas prestaciones cada vez mayores. El procesador alcanza las velocidades de los ordenadores de sobre mesa de hace tan solo unos pocos años. Su capacidad de memoria también ha aumentado de manera notable. A diferencia de los PCs que poseen gran catidad de memoria ROM (del orden de Gigas), en las PDAs la capacidad de almacenamiento permanente es bastante reducido. Por el contrario, el tamaño de la RAM es comparable al de un PC, siendo este tipo de memoria usado para almacenamiento de la mayoría de las aplicaciones y archivos dentro del dispositivo. Sus deficiencias de almacenamiento permanente (memoria ROM) se pueden suplir con el uso de tarjetas de almacenamiento extraíbles.

Un inconveniente importante de este tipo de dispositivos es el sistema operativo del que disponen. En el caso de la PDA adquirida para este proyecto, HP iPAQ Pocket PC 5500, el sistema operativo es Windows CE. Se trata de un sistema operativo tipo Windows, adaptado para esta familia de máquinas. Su desarrollo es conjunto entre Microsoft y HP, por lo que el soporte técnico es muy deficiente, al no depender enteramente de ninguna de las compañías. Como consecuencia de disponer de un sistema operativo tan específico, el software disponible para el mismo es bastante limitado. Un ejemplo de ello se encuentra a la hora de instalar un entorno de ejecución para Java. Sun no proporciona una máquina virtual para ser instalada sobre este sistema operativo. Dependemos pues de la maquina virtual suministrada por el fabricante de la PDA. En este caso es otra empresa independiente (Jeode), la que ha diseñado la máquina virtual para las series iPAQ Pocket PC de HP.

Como ventajas más destacables para este tipo de dispositivos se encuentran su portabilidad y capacidad de comunicación wireless. Además se dispone de módulos de expansión que extienden su capacidad de conexión o comunicación con otros dispositivos.

El lenguaje de programación utilizado para la implementación del código de la aplicación ha sido Java. Dentro de Java, se ha utilizado un estándar relativamente joven denominado J2ME, que es el estándar de programación para dispositivos móviles,

wíreless y de recursos limitados. Esta plataforma de programación permite aprovechar al máximo las capacidades de un amplio rango de dispositivos. La tendencia actual es un uso cada vez mayor de la plataforma Java2 Micro Edition, motivado en parte por el éxito cada vez mayor a nivel comercial de los dispositivos sobre los que se utiliza este estándar de Java.

7.2 Líneas Futuras de Actuación

Al ser la aplicación desarrollada un prototipo, a la hora de implantarla en el sistema real, se deben de hacer una serie de cambios. Tales modificaciones son comentadas a continuación. Así mismo, hay algunos aspectos del sistema que pueden ser modificados y mejorados. En este capítulo se trazan una serie de pautas de actuación para la ampliación o modificación de la aplicación desarrollada en este proyecto.

7.2.1 Sensores de Posición

Un simulador ejecutado sobre un PC genera de manera aleatoria los valores de los sensores en el prototipo desarrollado. Una vez implantada la aplicación en el entorno real deben de ser los valores reales de los sensores los que sean representados en la pantalla de la PDA. Estos valores serán enviados por el sistema de monitorización o leídos directamente del bus del sistema, concretamente de un bus tipo Profibus. En cualquiera de ambos casos el modo de representar los datos en la PDA usado en el prototipo es reutilizable, ya que únicamente cambia la fuente de obtención de los valores.

El rango de valores representado debe ser correspondiente a la realidad. Los valores representados en el prototipo de la aplicación de control: 40, 50 y 150 deben ser sustituidos por una escala de valores realmente representativa del entorno de trabajo, incluyendo más valores separados entre sí por la misma distancia. El comienzo del rango determinará el valor a partir del cual se pintarán las barras del gráfico. Por ejemplo, podría representarse una escala de valores desde 2 metros hasta 0,10 metros de la superficie del barco. Incluyendo todos los valores en dicho rango con un intervalo de 10 centímetros: [2, 1’90, 1’80…0’20, 0’10].

Respecto a la distancia a la cuál se activa la línea roja de alerta debe poder ser predefinida para cada situación en la que el sistema vaya a ser utilizado. Para ello se propone que este parámetro sea pasado mediante la línea de comandos cuando se ejecute la clase principal de la aplicación. La clase principal almacenará este parámetro en una variable. Cuando se cree el objeto de la clase que se encarga de representar el valor de los sensores (*Sens.class*), esta variable se le pasará como valor. De este modo, la línea de alerta aparecerá a una distancia o a otra en función del valor pasado como parámetro al ejecutar la clase principal. La modificación de este parámetro solo afecta al paquete de clases ejecutadas en la PDA. No es necesario hace ninguna modificación en el paquete de clases del servidor.

7.2.2 Captura de Vídeo

Una de las mayores ventajas de la utilización de un dispositivo tipo PDA para el control del sistema teleoperado, es la posibilidad de disponer de una pantalla sobre la que ejecutar interfaces gráficos de usuario y la entrada de vídeo procedente del sistema de visión.

En versiones futuras de la aplicación de control desarrollada, sería conveniente explotar en mayor profundidad la capacidad de captura de vídeo en tiempo real sobre la

PDA. Como objetivo futuro podría marcarse el poder mostrar de manera simultanea el vídeo en tiempo real y componentes gráficos Java.

Como ya se comentó anteriormente, cabe la posibilidad de que el Sistema de Visión disponga de más de una cámara. En tal caso, se debería de implementar el modo de conmutar de una cámara a otra, mediante el interfaz gráfico de usuario ejecutado en la PDA. Así mismo, queda por determinar si en la implementación real del sistema, el vídeo será suministrado a la PDA directamente desde la cámara, o si será procesado previamente por el Sistema de Visión.

Capítulo 8
Bibliografía

Libros

- Java 2 Micro Edition. Java in small things
  Autores: James p.White
  David A.Hemphill
  Editorial: Manning

- UML and C++
  A practical Guide To Object-Oriented Development
  Second Edition
  Autores: Richard C.Lee
  William M.Tepfenhart
  Editorial: Prentice-Hall, Inc

- Core Java 2 Volumen I – Fundamentals
  Autores: Cay S.Horstmann
  Gary Cornell
  Editorial: Sun Microsystems, Inc

Manuales

- Guía de usuario de *HP iPAQ Pockect PC*

- Guía de usuario de *Multimedia Expansion Pack for iPAQ FlyJacket i3800*

Recursos Web

- http://java.sun.com
- www.omg.org/uml
- http://www.pdagold.com
- http://www.pocketpccity.com
- http://pocketpcmag.com
- http://www.pdacore.com
- http://www.cegadgets.com
- http://discussion.brighthand.com
- http://www.wkmn.com/newsite/wireless.html

Informes Técnicos

- Development of a family of robots for ship hull blasting using an incremental approach.
  Autores: Pedro Sánchez, Bárbara Alvarez, Andrés Iborra, Carlos Fernández y Juan A. Pastor
  Proyecto europeo EFTCoR. Universidad Politécnica de Cartagena