# 3 Resultados.

¡Pero se mueve!
Galileo Galilei.

Al construirse el sistema se comprobó que sí se pudo manipular componentes físicos (*hardware*) a través de Internet, mediante una página web donde se muestra la respuesta de la videocámara (la obtención de la imagen), y donde también se despliegan los controles de posición de la misma (para manipular el puerto paralelo). En este caso, estos periféricos son controlados por ***servlets***, los cuales no tienen restricciones para tener acceso a todos los recursos computacionales del equipo donde se ejecutan -a diferencia de los *applet*s, que sí tienen restricciones para acceder a los recursos de la máquina donde se ejecutan.

Los ***servlets***, en particular la clase **ControlDireccionPuerto01.java**, invocan a las interfases ***Java Media Framework*** y ***Java Communications***, que implantan, a su vez, la interfaz *Java Native Interface*, la cual permite la comunicación entre el lenguaje java y librerías dinámicas (nativas del sistema operativo) escritas en otros lenguajes de computación, que controlan directamente los periféricos.

La combinación de ***servlets***, y el uso (directo o indirecto) de la interfaz *Java Native Interface* (*JNI*) y librerías dinámicas, permite, por un lado, invocar a métodos nativos (que están en el lenguaje binario del sistema operativo) y obtener distintos valores (enteros, flotantes, cadenas, etcétera) como respuestas, las cuales son más descriptivas que evaluar el resultado de invocar a un proceso del sistema operativo; *JNI*, también permite obtener respuestas de forma síncrona de los métodos de las librerías dinámicas, lo que brinda mayor control y certeza que esperar respuestas asíncronas de procesos independientes que se ejecutan periódicamente. Por otro lado, los ***servlets*** permiten construir aplicaciones web complejas y escalables, de forma sencilla, usando patrones tal como el ***MVC*** (***Model View Controller***, Modelo Vista Control) o alguno otro derivado de éste.

También al construir y migrar el sistema se encontraron ventajas y desventajas de la ***API Java Communications*** para *Windows* de 32 bits, en su uso para la instrumentación y supervisión a distancia.

Cuando se comenzó a evaluar y a usar esta *API*, la documentación (muy escueta) indicaba que estaba diseñada para el control de los puertos serie y paralelo, en un lenguaje orientado a objetos (Java), por lo que las ventajas parecían obvias, ya que se evitaba el problema de implantar rutinas de bajo nivel para el control de los puertos. Durante el desarrollo del sistema, se observó que sí se podía usar para abrir el puerto paralelo, y enviar datos; posteriormente, al integrarlo en un ambiente web (en ***Apache-Tomcat***, usando ***Windows Millenium*** como sistema operativo), funcionó correctamente.

Más adelante, en algunas pruebas, se modificó el alambrado para tratar de leer un dato por el puerto paralelo, pero la ***API Java Communications*** arrojó una excepción (*Unsupported operation. Output only mode*) al tratar de cambiar el modo *SPP* a *ECP* o a *EPP*, indicando que sólo permitía datos de salida. En una referencia bibliográfica (***Java I/O™***, de Elliotte Rusty Harold [33]), se encontró que la ***API*** sólo permite el modo ***SPP***, es decir, sólo puede enviar datos por el puerto paralelo, a pesar que en la documentación de la ***API*** ofrece abrir el puerto paralelo en distintos modos.

Se probaron varias combinaciones de las opciones de configuración del puerto paralelo en *Windows XP* con la interfaz ***Java Communications***, pero esta no opera correctamente en *Windows XP*, puesto que no envía datos, a menos que se inhabilite el puerto, se reinicie el equipo, y se habilite el puerto; incluso, a pesar de que esté trabajando correctamente, si se reinicia la computadora, la interfaz ***Java Communications*** vuelve a presentar problemas, por lo que se tiene que repetir los pasos de inhabilitar el puerto, reiniciar el equipo y habilitar el puerto.

También se intentó registrar la librería dinámica en el registro de *Windows XP*, pero el sistema operativo envía mensajes de advertencia al tratar de hacerlo, no importando cualquier combinación de opciones del comando

*regsvr32*, ya que *Windows XP* no reconoce a *win32com.dll* como una librería dinámica válida.

Cuando se cambió el sistema operativo a *Windows XP Profesional 2002* (con los *Service Pack* 2 ó 3), se encontró otra desventaja adicional; como el circuito electrónico no es un dispositivo apegado a los estándares *Plug and Play* -que requiere que el periférico envíe un identificador del dispositivo en una cadena en modo *nibble*-, el sistema operativo no lo reconoce, y la ***API Java Communications*** no opera correctamente. Se tiene que inhabilitar el puerto paralelo, apagar la computadora, conectar el circuito (alimentado correctamente), encender la computadora, habilitar el puerto paralelo, y finalmente, levantar ***Tomcat***. Otra forma de hacer que funcione la interfaz ***Java Communications*** en *Windows XP*, es tener seleccionada la opción "Usar cualquier interrupción asignada al puerto, en las propiedades de Puerto de impresora *ECP*", del Administrador de dispositivos; al cambiar la configuración esta interfaz puede enviar datos correctamente, y si ya se tenía dicha configuración, se tiene que inhabilitar y habilitar el puerto paralelo, sin necesidad de reiniciar la computadora.

Sin embargo, la librería dinámica *wincom32.dll* si presentó muchos conflictos al cambiar el sistema operativo, a pesar de que Windows ofrece compatibilidad con aplicaciones hechas para versiones anteriores. Por una parte, dicha librería dinámica no está completa y no se puede registrar en *Windows XP* porque depende de otras librerías dinámicas -las cuales son desconocidas porque no se mencionan en ninguna parte-; *Windows XP* sí puede registrar librerías dinámicas hechas para otras versiones, tales como *NT* y *Windows 2000*. Por otra parte, *Windows XP* con *Service Pack 2* ó *Service Pack 3* cambia la forma en que se tiene acceso a los periféricos, algo que no menciona explícitamente; la compañía *Microsoft Corporation* recomienda la instalación del *Service Pack 2* ó del *Service Pack 3*, presentándolas como mejoras a su sistema operativo, las cuales corrigen varios defectos y aumentan la seguridad del mismo, pero no mencionan de antemano que efectúan cambios radicales respecto a la comunicación con los dispositivos periféricos.

Por otra parte, este sistema de cómputo demuestra que sí se puede operar un periférico complejo, tal como la videocámara, por medio de Internet; la videocámara requiere de una configuración compleja para activala, y de varios pasos para obtener imágenes de ella, tan sólo como aplicación de consola; en ambiente de web se resolvieron los problemas de sincronización y de concurrencia. En este caso, al solicitar una imagen (por medio del ***servlet*** **ControlDireccionPuerto01.java**), la aplicación, por medio de las clases **VideoCamara.java**, de su clase padre, **CuadroVideo05.java** y la interfaz ***Java Media Framework***, sí se activa (se prende) la videocámara de estado sólido al ser llamada por primera vez, comienza a obtener video, se consigue una imagen del flujo de video y se detiene la toma de video de la videocámara. En llamadas posteriores, las clases **VideoCamara.java** y **CuadroVideo05.java** detectan que ya se había encendido la videocámara, por lo que no es necesario encenderla nuevamente, por lo que reinician la toma de video hasta obtener una nueva imagen, deteniendo la toma de video hasta que llegue otra solicitud.

En el caso de la interfaz ***Java Media Framework***, ésta usa a su vez la interfaz ***Java Native Interface*** para comunicarse con las librerías dinámicas que controlan los componentes físicos de audio y video (*hardware*). Esta colaboración entre el lenguaje java (independiente de la plataforma) y las librerías dinámicas (con código del sistema operativo en que se crearon) es más eficiente y obtiene más datos del resultado del llamado al método o función nativos, que si se invocara a un proceso del sistema operativo (el cual regresa menos datos), o si se tuviera que esperar el resultado de un proceso totalmente independiente (asíncrono), ya que en este último caso no se garantiza que se reciba respuesta después de hacerle una petición.

Respecto a la parte mecánica del movimiento de la videocámara, se encontró que el movimiento horizontal mediante un motor de pasos es relativamente sencillo; el movimiento vertical por medio de otro motor de pasos sí es factible, pero se requiere de la colaboración de otras ramas de la ingeniería para hacer una más eficiente. Se encontró que el peso de la videocámara y de un motor de pasos es mucho para que un segundo motor de pasos los mueva completamente, pero se hizo un mecanismo controlado por el segundo motor de pasos, el cual efectúa un movimiento de inclinación (vertical) del primer motor de pasos (y de la videocámara a la cual está unida); el diseño

mecánico para efectuar ambos movimientos de forma eficiente está fuera del alcance de esta tesis, ya que corresponde a la ingeniería mecánica.

Se pudo construir un módulo de administración relativamente sencillo, el cual tiene interacción con la base de datos *MySQL* para insertar, modificar, borrar o consultar información, mediante *J/Connector*, el cual implanta las interfases de ***JDBC***. También se pudo implantar la autenticación de usuarios, al construir un reino de seguridad con base en ***JDBC*** y papeles o funciones de usuarios, para permitir o negar el acceso a los recursos de la aplicación web, ya sean dinámicos (como los ***servlets*** y ***JSPs***), o estáticos (archivos ***HTML*** o imágenes). Se construyó un filtro de peticiones, para permitir o desviar solicitudes a un recurso web específico.

La construcción del módulo de administración comprobó que se pude tener interacción fácil y eficientemente con la base de datos relacional *MySQL*, la cual permite guardar datos numéricos, cadenas y fechas, así como información binaria, la cual puede corresponder a imágenes (como en el caso de este sistema) o que podrían ser datos de mediciones de otros sistemas. Esta base de datos puede ser utilizada por otro sistema de cómputo, para compartir información. Se pudo apreciar que las bases de datos relacionales son muy populares y fáciles de usar, pero se requiere de mucho trabajo hacer el código que ejecute las sentencias *SQL*.

Las interfases para el usuario (la respuesta ***HTML*** de los ***JSPs***), tanto del módulo de administración como del control de posición de la videocámara, son más sencillos y menos poderosos que los que se podrían tener mediante la construcción de una aplicación cliente hecha en otro lenguaje (que genere ejecutables), pero son eficaces, por lo que la aplicación web ahorra la construcción de la aplicación cliente (usando en cambio el navegador), mientras que se concentran los esfuerzos en la construcción del servidor; también permite que se tenga acceso a esta aplicación desde diversos sistemas operativos, siempre y cuando tengan acceso a Internet. También se ahorra la instalación y mantenimiento de aplicaciones cliente, de los sistemas Cliente-Servidor tradicionales, ya que al cambiar alguna regla, se tenía que reemplazar el cliente en todos los equipos donde estuviera instalado.

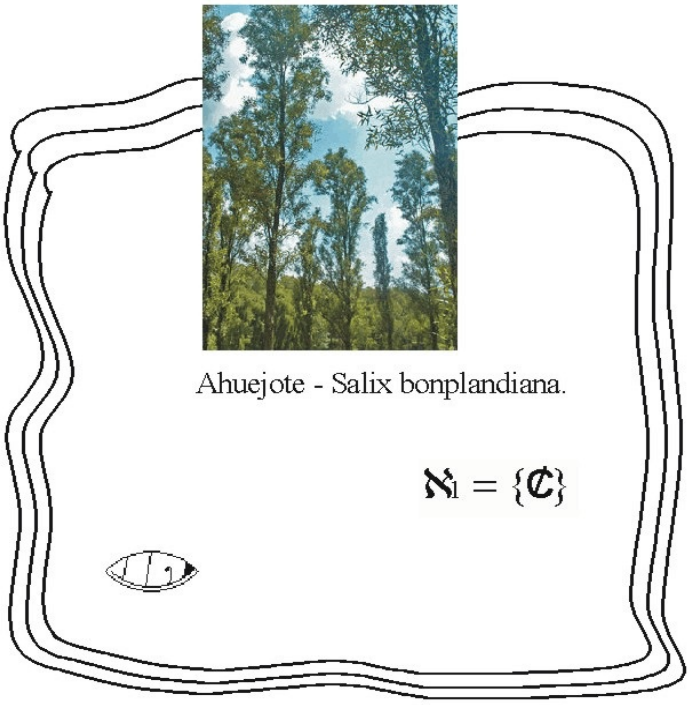


**Figura 25**: *La construcción de un sueño.*