4 Conclusiones.

Un número irracional para entender el universo,
y uno imaginario para inventar nuevas soluciones.

4.1 Conclusiones sobre el sistema de cómputo.

Después de construir el sistema y ponerlo en línea, se encontró que sí se puede controlar de forma remota un dispositivo, pero con varias condiciones. En el caso de la videocámara, la respuesta del dispositivo (la toma de una imagen), se efectúa de forma indirecta; en este caso, la computadora (servidor) se comunica con la videocámara a través de un cable *USB*, mientra que la aplicación web en ***Tomcat*** llama a las clases de la librería ***JMF***, las cuales invocan a métodos nativos (ejecutables del sistema operativo). El control de posición de la videocámara se logra con datos de salida únicamente.

La idea principal de este sistema es demostrar que es posible hacer uso de un dispositivo periférico a través de Internet, y con base en las técnicas aplicadas y en las experiencias obtenidas se quiere mostrar cómo se podrían utilizar otros periféricos, tales como instrumentos de medición.

En primer lugar, el lenguaje de programación Java demostró que es un lenguaje de programación bastante poderoso y flexible, con el cual se pude hacer distintos tipos de sistemas; las características del mismo, tales como la orientación a objetos y la herencia sencilla, permitieron resolver problemas lógicos tales como el modelado de características y comportamiento de los periféricos (el puerto paralelo y la videocámara de estado sólido) a través del uso de atributos y métodos.

El desarrollo del módulo de administración mostró que se puede hacer una aplicación compleja sobre web, usando ***servlets***, ***JSPs*** y *JavaBeans* con el modelo ***MVC*** (Modelo, Vista, Control), junto con la interfaz ***JDBC*** *J/Connector*, la cual es de tipo 4 (que no requiere de instalación o código nativo); también al implantar un reino de seguridad con base en ***JDBC*** en ***Apache-Tomcat***, para permitir o denegar el acceso a las páginas web mediante nombres de usuarios, contraseñas y papeles o funciones de usuarios, así como el filtrado de peticiones, muestra la versatilidad del contenedor de ***servlets Apache-Tomcat*** y del lenguaje java.

Por otra parte, los ***servlets*** y ***JSPs*** tienen acceso ilimitado a los recursos de la computadora sobre la cual se ejecutan, por lo cual pueden llamar a procesos o comunicarse con los periféricos de ese equipo, probándose de forma efectiva en ***Apache-Tomcat*** (el cual es gratuito); en servidores de aplicaciones comerciales cambiaría la configuración, pero dichos servidores son más eficientes y tienen mayor capacidad.

El lenguaje java también puede hacer uso de otros componentes de programación hechos en otros lenguajes de programación; a través de su interfaz *Java Native Interface* pude invocar a funciones de librerías dinámicas. No todos los sistemas de cómputo pueden construirse con un sólo lenguaje de programación, por lo que es importante identificar los alcances y limitaciones de cada lenguaje, sobre todo en este tipo de sistemas, donde se manipulan periféricos. Los lenguajes '*C*', '*C++*', o ensamblador, son más adecuados para controlar periféricos, pero llevaría mucho tiempo construir un sistema de cómputo con sólo lenguaje ensamblador (y el mantenimiento del mismo también sería costoso), mientras que los lenguajes '*C*' y '*C++*' presentan muchas desventajas para hacer páginas de Internet, ya que los *CGI* de dichos lenguajes crean procesos que demandan muchos recursos computacionales. El lenguaje java, al pretender ser multi-plataforma, no puede tener acceso directo a los periféricos de una computadora, pero en cambio, puede construir elementos de programación que tienen interacción con base de datos o que construyen páginas web rápida y eficientemente.

Respecto a otros lenguajes de programación, tiene más ventajas económicas y técnicas que otros lenguajes, a excepción del lenguaje *Ruby*. Algunos lenguajes de programación tal como el *Delphi*, puede crear aplicaciones que sí tengan acceso a los periféricos, e incluso crear aplicaciones web -mediante *CGI*, con las ventajas y desventajas

mencionadas en el capítulo de *Análisis*-, pero requiere el pago de licencia, y si se quisiera crear una aplicación similar en otro sistema operativo (puesto que existen instrumentos con controladores para otros sistemas operativos), se tendría que repetir todo el ciclo de desarrollo, ya que *Delphi* genera aplicaciones sólo para la plataforma *Windows*.

Algunos lenguajes de guiones crean páginas web rápidas y eficientes, e incluso se acoplan con distintas bases de datos, pero para la interacción con periféricos, tienen la misma desventaja que el lenguaje java, por lo que se tendrían que elaborar módulos en lenguaje '*C*' o '*C++*' para manipular los puertos locales y la videocámara; la ventaja del lenguaje java en este aspecto, es que cuenta con la interfaz *Java Native Interface*, la cual permite llamar a librerías dinámicas.

Respecto al lenguaje *Ruby*, parecería quedar en desventaja. El lenguaje Ruby parece ser un lenguaje de programación ideal, pero ha tenido menos respaldo y difusión que el lenguaje java; también genera código que necesita ser interpretado, y carece de la capacidad para tener interacción directa con los periféricos de una computadora, por lo que también necesitaría que se construyeran módulos en los lenguajes '*C*' o '*C++*' para controlar los puertos locales y la videocámara. *Ruby* cuenta con librerías para el uso de dispositivos multimedia, pero la interfaz ***Java Media Framework*** es más poderosa, debido a que fue creada por el acuerdo entre varias empresas de sistemas y de fabricantes de periféricos multimedia, lo que permite que dicha interfaz pueda manipular distintos dispositivos multimedia mediante métodos genéricos.

4.2 Acerca de los formatos de imágenes.

Las partes más complejas del sistema se hicieron en *Windows Millenium*, y al cambiar el sistema a *Windows XP*, el módulo de video presentó muy pocos problemas. Se pudo activar, obtener video, una imagen fija del video en formato ***JPEG***, y apagar la videocámara, a través de Internet; el formato ***JPEG*** es un formato muy ampliamente difundido, y adecuado para presentar imágenes en páginas de la red Internet, por lo que la gran mayoría de los navegadores o exploradores de Internet pueden interpretarla y desplegarla correctamente. Por otra parte, el formato ***JPEG*** tiene compresión con pérdida, por lo que no se recomienda para mediciones científicas, siendo más adecuado el formato ***PNG***, cuyos archivos son de mayor tamaño que los de ***JPEG***, requieren de más tiempo de transmisión en Internet, y no todos los navegadores o exploradores pueden desplegarlo correctamente.

El formato ***JPEG*** es adecuado para un sistema de video-vigilancia o supervisión remota; para un sistema el cual requiera capturar imágenes como medición científica, se requeriría cambiar la videocámara por otra de mayor resolución y cuyos controladores manejen el formato ***PNG***; en este caso, se podría usar la misma técnica de tratamiento de video, especificando que el formato de los *códecs* sea ***PNG***.

4.3 Conclusiones sobre las bases de datos.

Cuando se comenzó con el análisis y desarrollo del sistema, y se revisó el requerimiento de guardar la información, se consideró adecuado utilizar una base de datos relacional para guardar información de los usuarios del sistema, de las imágenes del sistema, e incluso para que otras aplicaciones pudieran guardar información de otros tipos de mediciones. El administrador de base de datos *MySQL* sí ha permitido que la aplicación guarde y recupere información, aunque el desarrollo de los objetos de acceso a datos (***DAOs***) y de otras clases que tienen interacción con la base de datos consumieron mucho tiempo. Sin embargo, el modelo Entidad-Relación diseñado, encausó la construcción de la aplicación, junto con el uso del modelo ***MVC*** (Modelo-Vista-Control).

De haber usado una base de datos orientada a objetos, se hubiera ahorrado mucho tiempo, evitando la construcción de los objetos ***DAOs*** y otras clases del lenguaje java relacionadas con base de datos, ya que ese tipo

de datos permite guardar las instancias de objetos java directamente; en esta caso se hubiera tenido que hacer más énfasis en el diseño de un diagrama de clases para guiar correctamente la construcción de la aplicación. Las bases orientadas a objetos ofrecen muchas ventajas, pero no han tenido mucha difusión, más que nada porque las bases de datos relaciones siguen siendo muy populares, tanto para aplicaciones propietarias de los fabricantes de bases de datos, de aplicaciones orientadas a objetos, y de lenguajes estructurados o con base en guiones.

Es por la última razón por la cual se debe seguir conociendo las bases de datos relacionales; las bases de datos orientadas a objetos ofrecen nuevas perspectivas, pero sólo los lenguajes orientados a objetos las pueden usar; también hay que considerar que sólo algunos fabricantes de administradores de bases de datos ofrecen administradores de bases de datos orientados a objetos libres de licencia y/o código abierto.

4.4 Conclusiones sobre el uso de puertos locales y periféricos con el lenguaje de programación java.

Respecto al uso de los puertos locales por parte de una aplicación de red Internet, aparecen ventajas y desventajas, influyendo el cambio de sistema operativo en el cual se ejecuta la aplicación (de *Windows Millenium* a *Windows XP*). El módulo de control del puerto paralelo comenzó a presentar problemas conforme se aplicaron "parches" de seguridad a *Windows XP*, ya que cambian la arquitectura con que la cual el sistema operativo permite el acceso a los periféricos. Los sistemas operativos y los periféricos de una computadora están íntimamente relacionados, ya que el sistema operativo coordina el envío y recepción de información por los canales de datos que unen los periféricos con el microprocesador, por lo que un cambio de versión de sistema operativo sí afecta el comportamiento de una aplicación que haga uso de periféricos -a pesar de que *Windows XP* garantizaba compatibilidad con aplicaciones hecha en versiones anteriores de *Windows*.

Para otro tipo de dispositivos, ¿cómo se podrían controlar y obtener una respuesta de los mismos, en Internet? A la anterior respuesta se ofrecen las siguientes soluciones:

- Para dispositivos que tengan controladores y protocolos de comunicación entre el periférico y la computadora bien definidos, se puede utilizar *JNI* (*Java Native Interface*) para implantar métodos que llamen a las funciones de código nativo, o utilizar la *API Jacob* (http://danadler.com/jacob/), la cual tiene base en *JNI*; finalmente, los ***servlets*** de su contenedor o servidor de aplicaciones, pueden llamara al código que utilice *JNI*.
- Utilizar la comunicación vía puerto serial y la ***API Java Communications***.
- Probar la ***API Java Communications*** en otros sistemas operativos.
- Reemplazar la librería dinámica ***win32.dll***, por otra que tenga el mismo nombre, pero tenga mayor funcionalidad, implantando las rutinas de bajo nivel (en lenguaje '*C*' o ensamblador).
- Desarrollar una ***API*** de comunicaciones (tanto como las clases como las rutinas de bajo nivel).
- Reemplazar la ***API Java Communications*** por la ***API RXTX*** (http://www.rxtx.org/).

En sistemas operativos anteriores, tal como ***MS-DOS***, las aplicaciones que deseaban usar los puertos paralelo o serie, solicitaban una referencia de memoria reservada a dichos puertos, y luego escribían datos en la memoria asignada al puerto correspondiente (envío), o leían datos de dichas direcciones de memoria (recepción).

Al aparecer *Windows 95*, *98* y *Millenium*, con su arquitectura de 32 bits, cambió la forma de tener acceso a dichos puertos. La arquitectura de 32 bits de *Windows* cambió la arquitectura con la que permitía la comunicación con los periféricos de una computadora, ya que en un ambiente multi-tareas se deseaba evitar conflictos entre aplicaciones por el uso de los periféricos, por lo que debían solicitarle permiso al sistema operativo para tener acceso

a los puertos locales, y registrar que están haciendo uso de los mismos (*ownership*). Microsoft permitía el desarrollo de controladores de periféricos por medio del *Device Development Kit* (*DDK*).

Más adelante, en Windows XP, y en específico, con los *Service Pack 2* y *Service Pack 3*, Microsoft cambia la arquitectura con la permite hacer el uso de los periféricos de la computadora; dichos *Service Pack* fueron introducidos como paquetes de mejoras a su sistema operativo, pero no mencionaban explícitamente que se cambiaba la arquitectura del uso de periféricos, reemplazando las herramientas del *Device Development Kit* por el *Windows Driver Kit (WDK)*, ocasionando problemas de compatibilidad para controladores de diversos periféricos, hechos con el *DDK*, ya que dicha compañía considera que dichas librerías dinámicas ocasionan problemas de seguridad.

Microsoft seguía permitiendo el acceso a los periféricos (puertos locales y otros) para crear nuevos controladores, mediante el pago de una licencia para el *WDK*, o adquiriendo alguna de sus herramientas de programación. Por otra parte, Microsoft, en los últimos sistemas operativos que ha liberado, ha reducido las capacidades del puerto paralelo, cambiándolo de un puerto de comunicaciones a uno exclusivamente de impresión.

También ha influido que los puertos serie, y en especial el paralelo, se reemplacen por puertos *USB*; los puertos serie y paralelo quedaron definidos por estándares físicos y eléctricos, pero cada sistema operativo define cómo envía y recibe datos de ellos. Por otra parte, el puerto *USB* (*Universal Serial Bus*) aparece como un acuerdo entre fabricantes de periféricos, de sistemas operativos y de fabricantes de computadoras, con el fin de presentar un puerto de comunicaciones mejorado. Este acuerdo permite que puedan usar distintos tipos de periféricos en distintos tipos de computadoras, que pueden tener distintos sistemas operativos.

Al crear el sistema de la presente tesis se demuestra que sí es posible controlar y hacer uso de dispositivos a través de páginas de la red Internet, pero debido a cambios de los sistemas operativos más recientes surge una disyuntiva.

Por un lado, para la instrumentación remota, se podría hacer uso de un servidor dedicado a un instrumento en particular; dicho servidor no necesitaría ser la computadora más reciente, ni tener el sistema operativo más actual; se podría re-usar un equipo de cómputo, y seguir utilizando el puerto paralelo o se podría usar el puerto serie; si bien la interfaz ***Java Communications*** presenta varios defectos y *Sun Microsystems* ha dejado de darle mantenimiento y respaldo para la plataforma *Windows*, no significa que no se pueda hacer, sino que se podría reemplazar por otra interfaz de programación -*Sun Microsystems* ha señalado que ***Java Communications*** se reemplaza por el proyecto ***RXTX***, el cual es un proyecto de *GNU*-, o se podría crear otra interfaz, creando una librería dinámica para las rutinas de bajo nivel, y hacer uso de la interfaz ***Java Native Interface*** (*JNI*), para que aplicaciones del lenguaje de programación java puedan hacer uso de dicha librería dinámica. Incluso, se podría utilizar otro sistema operativo, tal como *Linux*, que requiera menos recursos computacionales, con el fin de ahorrar dinero en compra de equipo de cómputo.

Por otra parte, para poder difundir una aplicación de videocámara con movimiento (en cualquier tipo de licencia), sería muy recomendable reemplazar el uso del puerto paralelo por uno *USB*; para poder hacer uso de instrumentos a través de puertos locales en las versiones más recientes de *Windows*, se tendría que desarrollar nuevos controladores de dispositivos, mediante el pago de una licencia a *Microsoft*, en especial si se crea un nuevo instrumento de medición, ya sea para uso local o remoto.

4.5 Trabajo a futuro.

En la aplicación web creada, se controla una videocámara, y las clases desarrolladas para ese fin se pueden aplicar en otros proyectos.

Para el caso concreto del control de la videocámara y de su posición, este sistema podría funcionar en un servidor dedicado a este proceso, sobre todo para evitar el problema de tener que inhabilitar/habilitar el puerto paralelo y reiniciar el equipo; también se podría instalar en *Windows 95*, *98* o *Millenium*, donde la ***API Java Communications*** tiene menos conflicto.

Respecto a la parte mecánica, se encontró que el movimiento horizontal es fácil de conseguir, pero para el movimiento vertical se encuentran problemas con el peso combinado de la videocámara y el motor de pasos del movimiento horizontal; para la presente tesis se contempló cómo resolver problemas lógicos de programación, principalmente. Para llevar a cabo el movimiento vertical de forma efectiva, se requiere de un nuevo diseño mecánico (que está fuera del alcance esta tesis), pero se puede esbozar uno, en el cual la videocámara y el motor de pasos del movimiento horizontal se encontrarían en una esfera -este motor movería la cámara, no la esfera-, con dientes de engrane en la parte posterior; la esfera descansaría en una semiesfera con un engrane (movido por el motor de pasos del movimiento vertical) que movería la esfera, pero el diseño de la parte mecánica ya es otro problema distinto que pertenece al área de mecatrónica.

Las videocámaras de estado sólido se han vuelto más populares, además de que manejan otros formatos de video de mejor definición, a la vez de que su tamaño se ha reducido, por lo que se pueden utilizar motores de paso más pequeños que consuman menos electricidad; para el módulo de video, se pueden cambiar los formatos de video y fotografía (siempre y cuando tengan soporte en la ***API Java Media Framework***).

Debido al los cambios con el que *Windows XP*, y otros sistemas operativos más recientes de Microsoft, administran el acceso a los periféricos, a la vez que *Sun Microsystems* descontinúa su interfaz de comunicación por puertos locales, se podría desarrollar nuevas librerías dinámicas para el uso de los puertos locales para las nuevas plataformas.

Por otra parte, el uso del puerto paralelo ha disminuido, siendo reemplazado por los puertos *USB* y *Firewire*, por lo que se podría desarrollar librerías de comunicaciones para las mismas, creando las rutinas de bajo nivel en lenguaje ensamblador o en '*C*', cuyas funciones serían invocadas desde el lenguaje java mediante *JNI*, y a su vez, estas clases se pueden llamar desde ***servlets*** o ***JSPs*** en ***Apache-Tomcat*** (o desde un servidor de aplicaciones).

Recientemente, Microsoft permitió la distribución gratuita del *WDK*[51] (desde finales de diciembre del 2009), como incentivo para los programadores que deseen desarrollar controladores y aplicaciones de periféricos para los sistemas operativos que ha liberado recientemente (en específico, *Windows 7*). Esto brinda más oportunidades para el desarrollo de aplicaciones que requieran tener acceso a puertos locales -sin importar si es a través de web o de una aplicación que se ejecute localmente, ya que están proporcionando las únicas librerías que permiten la comunicación entre periféricos y aplicaciones dentro de los sistemas operativos más recientes de Microsoft.

El uso remoto de dispositivos de medición y/o control permitirá seguir haciendo uso de instrumentos o dispositivos cuando existan problemas de causa mayor que impidan o limiten el tránsito de personas (tal como la contingencia sanitaria por la epidemia de la influenza *AH1N1*), por lo que se deberá desarrollar nuevos controladores para los sistemas operativos más recientes, así como crear librerías compartidas para invocar a dichos controladores (tanto por aplicaciones locales, como aplicaciones de Internet).

Por último, faltaría probar la comunicación de los puertos serie y paralelo en otros sistemas operativos (tal como *Linux* o *Solaris)*, donde se podrían controlar otro tipo de dispositivos no comerciales; también se podría probar las capacidades de los puertos *USB* y *firewire* en los sistemas operativos antes mencionados, para ampliar la variedad de plataformas en las cuales se podrían usar dispositivos (comerciales o no), ya sea por medio de aplicaciones locales, o usando *JNI* y ***servlets*** para crear aplicaciones de Internet.

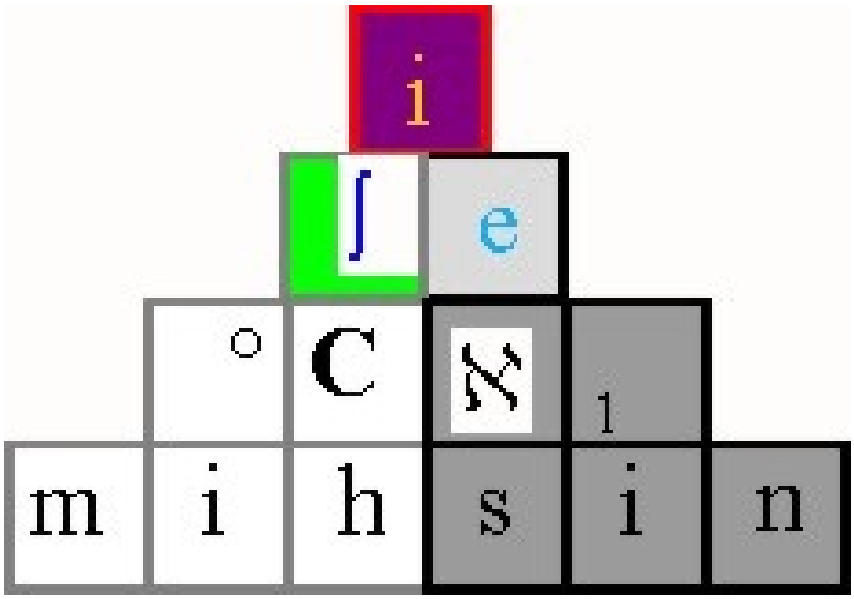

Figura 26: *Los conocimientos reunidos.*