Universitat Autònoma de Barcelona

escola d'enginyeria

PROYECTO Nº 2017. IMPLEMENTACIÓN DE MOVIMIENTOS SUAVES EN ROBOTS INDUSTRIALES MEDIANTE SPLINES CÚBICOS. APLICACIÓN A UN LABORATORIO REMOTO DE ROBÓTICA

Memoria del Proyecto Fin de Carrera
De Ingeniería en Informàtica
Realizado por
Marcos Mohedano Gómez
y dirigido por
Asier Ibeas Hernández
Bellaterra, Junio 2010

Universitat Autònoma de Barcelona

escola d'enginyeria

El abajo firmante, **Asier Ibeas Hernández**

Profesor de la "Escola Tècnica Superior d'Enginyeria" de la UAB,

CERTIFICA:

Que el trabajo al que corresponde esta memoria ha sido realizado bajo su dirección por **Marcos Mohedano Gómez**

Y para que conste firma la presente.

Firmado: **Asier Ibeas Hernández**

Bellaterra, Junio 2010

RESUMEN

Uno de los principales obstáculos con los que se pueden encontrar los estudiantes universitarios (muchos de los cuales compaginan trabajo y estudios) a la hora de realizar las clases prácticas de las diversas asignaturas de la carrera, son los horarios de estas. Las aulas destinadas a la realización de dichas clases tienen una capacidad limitada y las horas disponibles para su utilización deben repartirse entre varias asignaturas de la misma carrera, o incluso de carreras distintas.

Este problema puede solucionarse gracias a los laboratorios remotos, que ofrecen la posibilidad de acceder desde cualquier ordenador con conexión a internet a los elementos físicos o software instalados en las aulas destinadas a la realización de las clases prácticas, e interactuar con estos sin necesidad de estar presentes en dicho lugar. De esta manera los alumnos disponen de una mayor flexibilidad de horarios que les facilitará compaginar sus estudios con otras actividades.

Un alumno de la carrera Ingeniería Técnica en Informática de Sistemas de la EUIS (*Escola Universitària d' Informàtica de Sabadell*) de la UAB, Ramón González, creó como proyecto final de carrera en el curso académico 2008/2009 un laboratorio remoto para la realización de prácticas de la asignatura "Robótica y Automatización Industrial", perteneciente a la carrera de Ingeniería en Informática impartida en la ETSE (*Escola Tècnica Superior d'Enginyeria)*, UAB. Dicho sistema consistía en una interfaz web que permitía programar y ejecutar a distancia operaciones PPO (*Pick & Place Operations*) sobre un brazo robot situado en el laboratorio de automática de la ETSE. Con esto se pretendía no sólo mejorar la flexibilidad de horarios en las prácticas de la asignatura si no también dotar a estas de un atractivo mayor para el alumno y acercarlo un poco más a la realidad del mundo de la robótica, ya que hasta entonces las prácticas de la asignatura se realizaban en Matlab, un entorno de programación que se utilizaba para verificar las ecuaciones que gobiernan a los robots industriales.

El actual proyecto trata de crear un nuevo laboratorio remoto con mejores prestaciones que el anterior, añadiendo como mejora principal una funcionalidad que dote al robot con la capacidad de seguir trayectorias con movimientos suaves gracias a unas ecuaciones matemáticas denominadas *splines cúbicos.* Además, se añadirán dos cámaras de red de alta resolución y se

mejorará la interfaz web añadiendo la retransmisión en vivo del vídeo captado por las cámaras, para así poder observar con nitidez los movimientos realizados por el robot.

ÍNDICE

1. INTRODUCCIÓN. MOTIVACIÓN Y OBJETIVOS

Con este proyecto se intenta crear una herramienta útil para la docencia relacionada con las prácticas de la asignatura "Robótica y Automatización Industrial" perteneciente a la carrera de Ingeniería en Informática impartida en la ETSE, UAB.

Hasta ahora, en dicha asignatura cuyo temario consta de dos bloques bien diferenciados (Robótica, y Automatización Industrial), las prácticas pertenecientes al bloque de Robótica consistían en realizar simulaciones con Matlab (ver figura 1), y tenían un enfoque prácticamente matemático, bastante teórico.

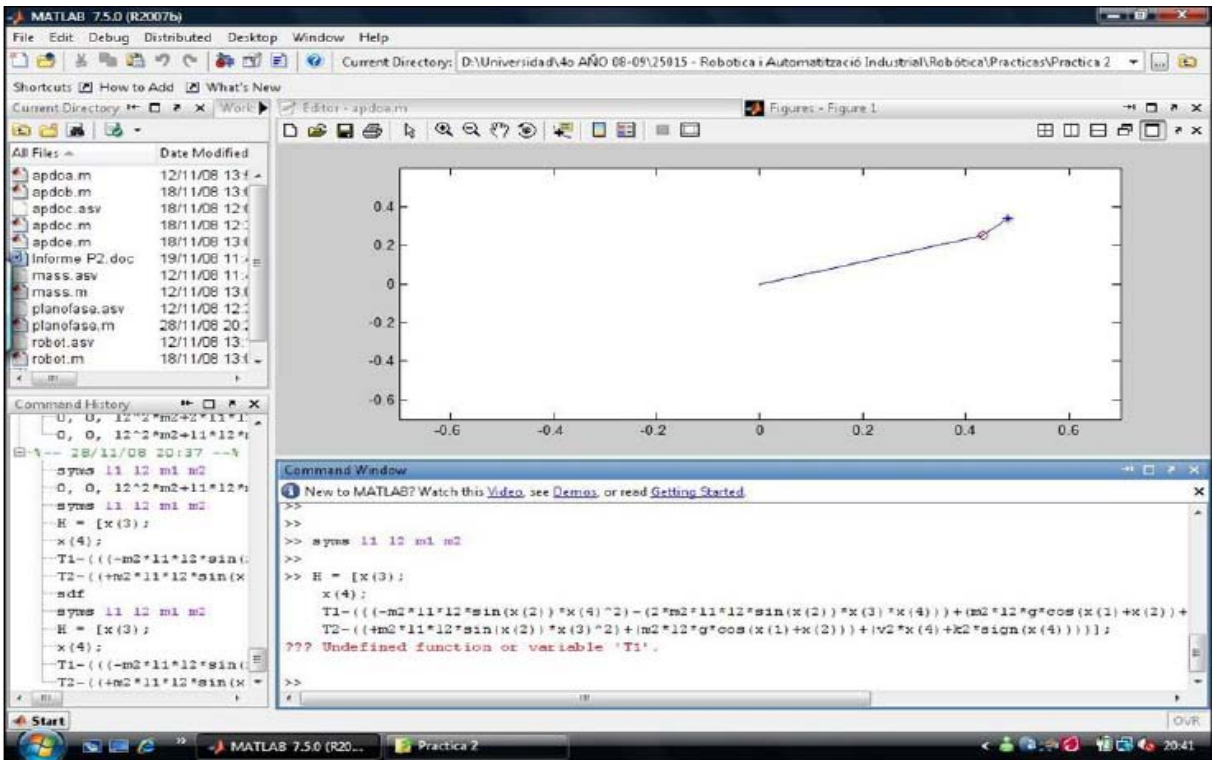


Figura 1 - Ejemplo de simulación con Matlab

Durante el curso 2008/2009 un alumno de la carrera Ingeniería Técnica en Informática de Sistemas de la Escuela Universitaria de Informática de Sabadell de la UAB, Ramón González, realizó un proyecto que consistía en montar un brazo robot de bajo coste e implementar una web desde la cual se pudiera acceder al laboratorio de Robótica remotamente para poder realizar operaciones PPO (*Pick and Place Operations*, es decir, operaciones en las que el

brazo robot coge objetos y los deposita en las coordenadas deseadas) y visualizar las acciones del robot mediante una webcam casera.

El proyecto actual es una mejora del nombrado en el anterior párrafo, cuyos objetivos principales son los siguientes:

- Crear una nueva interfaz web que permita acceder remotamente al laboratorio de Robótica y enviar programas (realizados en un lenguaje de etiquetas) para ser ejecutados por el robot.

- Tratar de mejorar la visualización remota del robot implementando *streaming* de vídeo en vivo en la aplicación web (en el proyecto anterior se visualizaba el robot a través de una webcam casera guardando capturas cada cierto tiempo). Para la captura de vídeo se han comprado dos webcams IP de videovigilancia de la marca AXIS. Como son cámaras de videovigilancia, la resolución de la imagen será mucho mayor (1,3 Megapíxeles) que la ofrecida por la webcam casera del proyecto inicial (VGA). Además, el streaming de vídeo en directo permitirá observar el movimiento del robot con total fluidez, frente a la visualización "a trozos" que ofrecían las capturas de imágenes en intervalos de tiempo de la webcam casera. El hecho de tener dos webcams en lugar de una, ofrecerá además una visión más "tridimensional" del robot en movimiento, al poder capturar el vídeo desde distintos ángulos.

- Implementar algoritmos para la generación de trayectorias suaves mediante interpolación con funciones *splines* cúbicos de manera que los alumnos puedan indicarle al robot distintas posiciones y que éste siga una trayectoria que pase por los puntos indicados con un movimiento de velocidad y aceleración contínuas.

Con todo ésto se pretende acercar la asignatura "Robótica y Automatización Industrial" al alumnado de manera que le resulte más atractiva y que las clases prácticas sean lo más parecido posible a lo que se va a encontrar en casos reales. También servirá para poder flexibilizar los horarios de prácticas ya que gracias al acceso vía web al laboratorio, cualquier alumno podrá realizarlas desde cualquier lugar que disponga de conexión a internet sin necesidad de estar presente físicamente en dicho laboratorio. Además, otras facultades o

centros docentes que no dispongan de los medios técnicos y/o económicos para comprar un brazo robot y montarlo en un laboratorio, podrían acceder al nuestro a través de la web.

2. ESTADO DEL ARTE. ANTECEDENTES

Según la enseñanza tradicional, la realización de las sesiones de prácticas de ciertas asignaturas de ingeniería implica habitualmente que el alumno debe asistir a determinados turnos de horarios fijos, en un laboratorio que posee unos equipos físicos limitados.

El conjunto de prácticas propuestas deben tener por objetivo el ilustrar los resultados obtenidos en las clases teóricas, así como familiarizar al alumno con el manejo de instrumentos y equipos empleados en situaciones reales.

Pero por motivos técnicos o económicos, no todos los centros docentes pueden poseer equipos físicos lo suficientemente similares a los que se emplearían en situaciones reales, así que hasta la aparición de los primeros laboratorios remotos, las prácticas debían basarse en simulaciones software.

Gracias a los laboratorios remotos, los alumnos de centros que no dispongan de dicho equipamiento o que no puedan asistir a las sesiones de prácticas en los horarios propuestos, tienen la oportunidad de realizar dichas prácticas desde cualquier lugar con conexión a internet a través de interfaz web o aplicaciones de conexión remota.

En el caso concreto de la robótica, actualmente existen varios laboratorios a los cuales se puede acceder vía internet y así poder manipular brazos robots, robots móviles etc. remotamente.

A continuación se muestran algunos ejemplos de laboratorios remotos relacionados con el mundo de la robótica:

- **RobUALab.Ejs**, de la Universidad de Alicante. Accediendo a este laboratorio sus alumnos pueden manipular un brazo robot con 5 grados de libertad.

  Se puede acceder al laboratorio en la dirección:

  http://robualab.eps.ua.es/

  Mientras que un vídeo demostrativo del mismo se encuentra en:

http://robualab.eps.ua.es/videos/robualab-high.mpg

- **The Tele-labs Proyect (LR)**. Sistema para la operación remota de un brazo robot industrial real, que puede manipular objetos básicos sobre un tablero. Utiliza un software cliente propio muy completo y está accesible en:

  http://telerobot.mech.uwa.edu.au/

- **Augmented Reality Interface for Teleoperation via Internet**. Sistema que permite la visualización, con realidad aumentada, y el control de un robot de 4 DGF a través de Internet:

  http://lsc.cemif.univ-evry.fr:8080/Projets/ARITI/index.html

Hay varios motivos por los cuales no se ha decidido utilizar para las prácticas de Robótica de la ETSE ninguno de estos laboratorios remotos ya existentes. Los principales són los siguientes:

- Al depender de un servicio externo a la UAB, los horarios y calendario de prácticas deberían amoldarse a las fechas en que se encuentre operativo el servicio a utilizar. Por ello se ha creído conveniente desarrollar un laboratorio remoto propio, con una web creada en un servidor de la ETSE y así no tener los inconvenientes de depender de factores externos.

- Siendo nosotros los creadores de nuestro propio servicio de laboratorio remoto, podremos añadir nuevas funcionalidades en el futuro o modificar las existentes para enriquecer el nivel de las prácticas y que estas se ajusten a los criterios de docencia de la facultad.

3. ESTUDIO DE VIABILIDAD

En este apartado se estudiarán las características técnicas, tiempo necesario para realizar el proyecto y coste económico para llegar a una conclusión sobre la viabilidad del proyecto.

3.1. RECURSOS

Los recursos necesarios para la realización del proyecto son los siguientes:

- Tiempo del alumno.
- Tiempo del director de proyecto.
- Robot *Lynx 6* con servocontrolador *ssc-32* montado en plataforma de madera.
- PC servidor con sistema operativo *Windows XP Professional* e IIS (*Internet Information Services*) instalado.
- Licencia *Microsoft Windows XP Professional Edition*.
- 2 cámaras de red *AXIS 207*.
- Conexión Internet ADSL.
- Software
    - Entorno de programación
        - *Microsoft Visual Web Developer Express Edition*.
    - Software para la captura y streaming de video
        - *Microsoft Windows Media Encoder*.
    - *Drivers* de las cámaras de red.

3.2. VIABILIDAD TÉCNICA

Seguidamente estudiaremos la compatibilidad entre los elementos físicos y *software* que componen el sistema, así como su disponibilidad, para determinar si el proyecto es viable técnicamente.

3.2.1. Robot

El robot ya se montó y se fabricó su entorno de trabajo en el proyecto antecesor al actual, realizado por el alumno de Ingeniería en Informática de Sistemas Ramón González. Por lo tanto ya se comprobó en su momento la viabilidad técnica del montaje de este y de la compatibilidad de sus componentes y circuito de control con el puerto serie del PC.

3.2.2. Lenguajes de programación

Se ha escogido realizar el diseño web y la programación de las funcionalidades que permiten la interacción entre el usuario, la interfaz web, el servidor y el robot como un único conjunto, gracias a la tecnología ASP.NET bajo el lenguaje de programación orientada a objetos C#. Hay múltiples razones por las cuales se ha elegido utilizar dicha tecnología , entre las cuales destacan, [1] :

- La tecnología ASP.NET es totalmente compatible con el servidor escogido (IIS de Microsoft) para la realización del proyecto.

- ASP.NET aplicada en un servidor IIS sobre un sistema operativo windows ofrece mayor velocidad de respuesta en el servidor que cualquier otra tecnología como pudiera ser PHP.

- ASP.NET introduce el concepto de "código oculto" (*code-behind*), que separa cláramente la parte de la programación (capa de negocio) del diseño de la interfaz web (capa de presentación) en dos ficheros distintos permitiendo de esta manera la programación por capas, lo que maximiza la facilidad de mantenimiento de la web.

- Se ha escogido el lenguaje C# porque ya se estaba familiarizado con él, y porque al ser orientado a objetos facilita la modularidad y el encapsulamiento. Además posee un

elevado número de clases y métodos que simplifican mucho la programación frente a otros lenguajes.

- ASP.NET proporciona una gran cantidad de clases, métodos y controles web que permiten al programador realizar con muy poco código tareas que en otros lenguajes/tecnologías como PHP ocuparían muchas más líneas de código y complicarían la programación de la web.

Nota: En el Capítulo 6 se explicará con mayor detalle en qué consisten ASP.NET y C#.

3.2.3. Servidor web

Como servidor se ha utilizado un PC con sistema operativo Windows XP Professional Edition, en el cual se ha instalado el componente IIS (*Internet Information Services*). Es dicho componente de Windows XP el que dota al PC de las características de servidor web.

Es un componente de Microsoft, por lo que es totalmente compatible con la tecnología ASP.NET, del mismo fabricante.

3.2.4. Cámaras de red

Se han adquirido dos cámaras IP AXIS 207, que conectadas a la red local donde se ubica el servidor web permiten su visualización accediendo a la dirección IP de cada cámara. Codifican vídeo en formatos Motion JPEG y MPEG-4 simultániamente, lo cual proporciona la mejor calidad de imagen y gestión de ancho de banda de su clase.

Tanto los controladores de las cámaras como los *códecs* necesarios para la visualización del vídeo codificado por estas son totalmente compatibles con el sistema operativo utilizado en el proyecto.

3.2.5. Software de captura de vídeo y servidor de *streaming*

Se ha utilizado la aplicación *Windows Media Encoder*, de Microsoft. Los motivos han sido los siguientes:

- Dicho software realiza a la vez las funciones de servidor de *streaming* de vídeo y de software codificador de video.

- Windows Media Encoder codifica el vídeo proviniente de la fuente elegida (en este caso las webcams AXIS) en formato WMA (*Windows Media Archive*), el formato de vídeo de Microsoft Windows. Por lo tanto no existirán problemas de incompatibilidad con el sistema operativo del servidor. Además es un formato de vídeo compatible con la mayoría de navegadores web más utilizados (Internet Explorer, Firefox, Opera,….).

- Además, Windows Media Encoder es el único servidor de *video streaming* compatible con los *drivers* de captura de vídeo de las cámaras AXIS.

Tras analizar las tecnologías de los recursos utilizados, la total compatibilidad entre estas, comprobar que el robot utilizado está preparado para seguir trayectorias de tipo *spline* cúbico y que estas son factibles para la programación del robot, se ha llegado a la conclusión de que el proyecto es técnicamente viable.

3.3. COSTES TEMPORALES

Las labores a realizar que componen el proyecto son las siguientes:

1. Buscar información sobre *video streaming* en vivo, y descargar el software necesario para su implementación (20 h).

2. Elegir lenguaje de programación para la creación de la web del laboratorio y de los algoritmos relacionados con el movimiento del robot (10 h).

3. Creación de la interfaz web con streaming de vídeo en directo proviniente de dos webcams IP Axis 207 en la cual se puedan crear/modificar usuarios, y preparar la web para que el servidor pueda enviar datos al robot a través del puerto serie (50 h).

4. Desarrollo de los algoritmos necesarios para el movimiento del robot siguiendo trayectorias suaves mediante *splines* cúbicos (80 h).

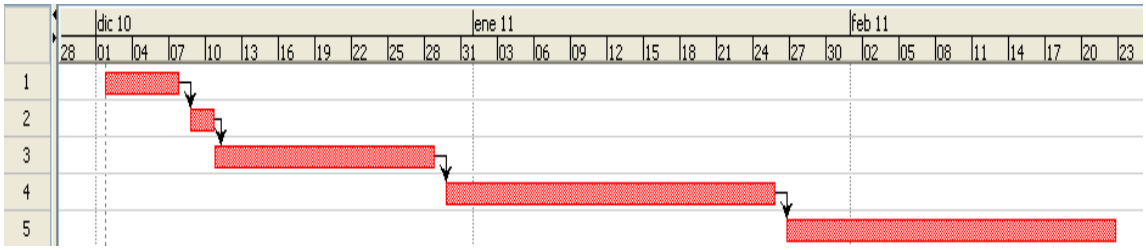
5. Elaboración de la memoria del proyecto (80 h).

Tiempo total: 240 h.

Los cuatro primeros puntos mencionados se pueden considerar como módulos independientes entre sí, entre los cuales no existirá un intercambio de información. Al final todos los módulos se integrarán en un todo, pero no es necesario conocer la implementación de uno para poder desarrollar otro. Por lo tanto no importa el orden temporal en el cual se realicen las 4 labores principales en las que se ha dividido el proyecto. La memoria se elaborará una vez terminados los puntos anteriores.

Se dedicarán una media de 20 horas semanales de trabajo.

Por lo tanto, la planificación temporal del trabajo quedará de la siguiente manera:



Dates			
Start	1/12/10 18:00	Finish	22/02/11 22:00

Duration	
Scheduled	60 days

Work	
Scheduled	240 horas

Nota: A pesar de que las cuatro primeras labores no dependen unas de otras y por lo tanto no importa el orden en que se realicen, se ejecutarán en el orden cronológico que se ha indicado en el diagrama de Gantt.

En el eje vertical del diagrama de Gantt se encuentran las labores mencionadas al inicio del apartado 3.3 numeradas del 1 al 5.

El plazo de entrega de la memoria del proyecto es entre el 17 y el 22 de Junio del 2010. Por lo tanto, según la planificación realizada confirmamos la total viabilidad temporal del proyecto.

3.4. COSTE ECONÓMICO

Analicemos el coste económico de cada recurso:

- **Tiempo del alumno**.

  Ya que el proyecto forma parte de los créditos obligatorios a realizar por el alumno, esto no supone ningún coste económico para el departamento.

- **Tiempo del director de proyecto.**

  El sueldo a cobrar por parte del director del proyecto por el tiempo dedicado ya se encuentra incluido en su salario como profesor de la UAB. Por lo tanto tampoco implica un gasto que descontar del presupuesto asignado.

- **Robot *Lynx 6* con servocontrolador *ssc-32* montado en plataforma de madera.**

  Este elemento pertenece al proyecto desarrollado por otro alumno en el curso 2008/2009. Por consiguiente, el coste económico de este recurso es nulo.

- **PC servidor con sistema operativo *Windows XP Professional* e IIS (*Internet Information Services*) instalado.**

  Este recurso supone un gasto de 0€, pues ya se disponía de dicho PC y su correspondiente licencia de Windows XP Professional Edition.

- **2 cámaras de red *AXIS 207*.**

  El coste total de las dos cámaras (IVA incluído) es de 512,60€.

- **Conexión Internet ADSL.**

La ETSE dispone de conexión a Internet, por lo tanto no hay que añadir un coste extra.

- **Software**

Todo el software utlizado para la realización del proyecto (software de desarrollo web, *drivers* de las cámaras, captura y codificación de vídeo) es de licencia gratuïta.

El presupuesto anual asignado al departamento de Telecomunicación e Ingeniería de Sistemas para el Laboratorio de Automática es de aproximadamente 1200€. Teniendo en cuenta que el único gasto que se efectuará es el de la compra de las dos cámaras de red (512,60€), se puede afirmar que el proyecto es económicamente viable.

4. FUNDAMENTOS TEÓRICOS

En este capítulo se explicarán una serie de fundamentos teóricos sobre los robots industriales y las ecuaciones para la generación de movimientos suaves (*splines* cúbicos), necesarios para una mejor comprensión del funcionamiento de este proyecto.

4.1. ROBOTS INDUSTRIALES

Se explicarán conceptos básicos sobre robots industriales, elementos que los componen y teoría sobre robótica.

4.1.1. Definición y clasificación

El estándar ISO (ISO 8373:1994, Robots industriales manipuladores – Vocabulario) define un **robot industrial** como un manipulador programable en tres o más ejes multipropósito, controlado automáticamente y reprogramable, [2].

Según el modo en que son controlados, los robots industriales se clasifican en, [3]:

- **Manipuladores**: poseen un sencillo sistema de control que permite gobernar el movimiento de sus elementos.

- **De repetición o aprendizaje:** se limitan a repetir una secuencia de movimientos previamente ejecutada por un operador humano, haciendo uso de un controlador manual o un dispositivo auxiliar.

- **Con control por computador:** están controlados por un computador (normalmente un microordenador) mediante un lenguaje específico compuesto por un repertorio de instrucciones que permiten la programación de los movimientos del robot. El robot utilizado en el proyecto se corresponde con esta clasificación.

- **Inteligentes**: similares a los anteriores pero capaces de interaccionar con el entorno mediante sensores y de tomar decisiones en tiempo real.

4.1.2. Componentes

Los principales elementos que componen un robot industrial son (ver figura 2):

- **Manipulador:** Formado por una secuencuencia de cuerpos rígidos (*enlaces*) que se conectan entre ellos mediante *articulaciones*, formando una *cadena cinemática*. En uno de los extremos de dicha cadena se encuentra el *efector final*, que en nuestro caso será una pinza para coger objetos. El movimiento del manipulador se efectúa gracias a una serie de motores que pueden ser de tipo hidráulico, pneumático o eléctrico. Nuestro manipulador contiene motores eléctricos denominados *servomotores*, colocados en sus articulaciones.

  Un *servomotor* (o servo) es un motor de corriente continua que tiene la capacidad de ser controlado en posición. Es capaz de ubicarse en cualquier posición dentro de un rango de operación (generalmente permite rotaciones de valor máximo 180º) y mantenerse estable en dicha posición, [6].

- **Controlador:** Controla los movimientos del manipulador enviando señales a los servomotores que se encuentran en sus articulaciones y guardando sus posiciones. Para ello recibe y envía señales a los dispositivos de entrada/salida.

- **Dispositivos de entrada/salida:** Envían información al controlador para que este la procese y envíe al manipulador las señales necesarias para su funcionamiento. También reciben datos del controlador con información sobre el estado del manipulador. En nuestro caso el único dispositivo de entrada/salida que utilizaremos será una computadora adicional (el servidor web).

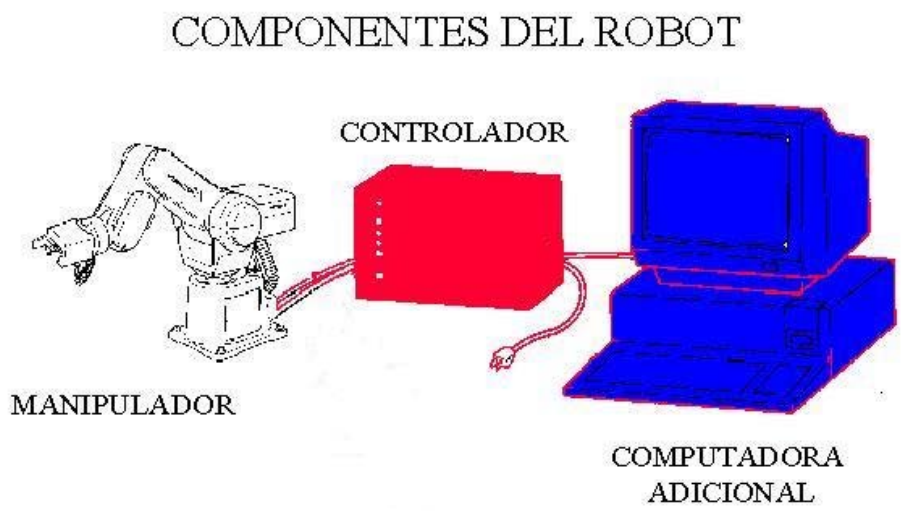


Figura 2 - Componentes de un robot industrial

4.2. CONCEPTOS IMPORTANTES EN ROBÓTICA

Para comprender el funcionamiento de nuestro sistema, es necesario tener claros los siguientes conceptos básicos sobre robótica:

4.2.1. Espacio de trabajo

El **espacio de trabajo diestro** es el conjunto de puntos del espacio en los que puede situarse el efector final del manipulador con todas las orientaciones [4].

El **espacio de trabajo alcanzable** es el conjunto de puntos del espacio en los que puede situarse el efector final del manipulador en por lo menos una orientación.

A partir de ahora, cuando se hable del espacio de trabajo, nos estaremos refiriendo al espacio de trabajo alcanzable.

4.2.2. Grados de libertad

Se definen como cada una de las coordenadas independientes necesarias para describir el estado mecánico de un sistema. Normalmente, cada par enlace-articulación posee un solo grado de libertad (*rotacional* en el caso de que un enlace gire alrededor de un eje fijo al enlace anterior, o *translacional* cuando se deslice sobre él en línea recta), pero no necesariamente.

4.2.3. Coordenadas en el spacio cartesiano/de articulaciones

El **espacio cartesiano** está definido por uno, dos o tres ejes perpendiculares entre sí. A estos ejes se les conoce como *sistema de referencia*. En el caso del manipulador empleado en este proyecto nos referiremos siempre a un espacio de tres ejes (x, y, z) ya que tiene la capacidad de moverse en tres dimensiones. Las coordenadas cartesianas definirán la posición del efector final del manipulador en dicho espacio, teniendo como orígen del sistema de referencia la base del robot.

El **espacio de articulaciones** se define por los ángulos que forman los enlaces en las articulaciones con sus enlaces contiguos. Por lo tanto, las coordenadas en el espacio de articulaciones de un manipulador serán una serie de variables (tantas como articulaciones) que expresan dichos ángulos.

4.2.4. Cinemática del manipulador

La cinemática de un manipulador define las propiedades geométricas y temporales del movimiento de este. Disntinguiremos dos situaciones:

- **Cinemática directa:** consiste en determinar la posición y orientación en coordenadas cartesianas del efector final del manipulador, dados los valores de las variables de articulación (ángulos).

- **Cinemática inversa:** es justo lo contrario. A partir de las coordenadas cartesianas del efector final, se calculan los ángulos de las articulaciones. Esto es lo más usado, y absolutamente necesario, dado que las tareas a realizar o trayectorias a recorrer por un manipulador se dan casi siempre en coordenadas cartesianas referidas a algún sistema fijo.

4.3. GENERACIÓN DE TRAYECTORIAS DE TIPO *SPLINE* CÚBICO

Una trayectoria es una sucesión de puntos en el espacio con restricciones temporales. Por lo tanto, cuando se desea que el efector final de un manipulador siga una trayectoria determinada, lo que se está pretendiendo es fijarle una trayectoria completa en el espacio. Eso se puede hacer describiendo la curva que se quiere que siga por medio de herramientas de

geometría diferencial [22]. Como esto constituye una tarea ardua, en lugar de describir toda la trayectoria se suele relajar esa condición obligando a que se pase sólo por ciertos puntos del espacio (ver figura 3).

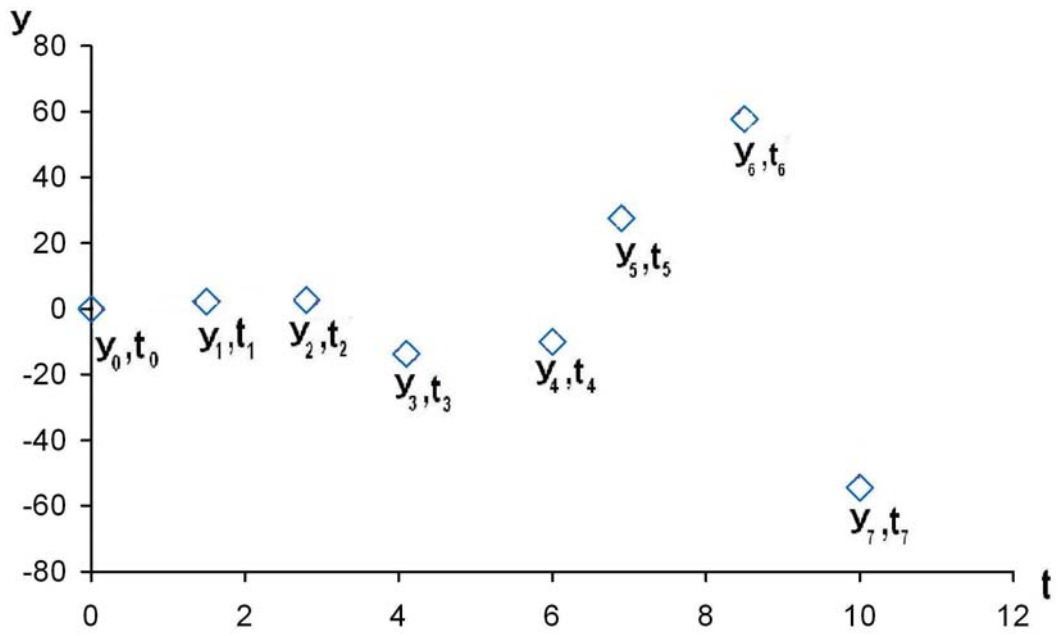


Figura 3 – Sucesión de posiciones del espacio en determinados instantes de tiempo

En la figura 3 observamos una serie de puntos representados por rombos de color azul, cada uno de los cuales indica la posición deseada (eje vertical de la gráfica) en un instante de tiempo concreto (eje horizontal).

Ahora el problema reside en cómo saber qué posiciones intermedias entre los puntos indicados debe recorrer el manipulador para completar la trayectoria. En esto consiste la **interpolación.** Una función de interpolación q(t) evaluada en un instante de tiempo t, devuelve el valor de la posición perteneciente a dicho instante de tiempo.

Uno de los métodos de interpolación más sencillos es la **interpolación lineal** a través de polinomios de grado uno (ecuaciones de rectas). Por ejemplo, si se tiene un punto inicial representado por una posición en un instante de tiempo (q_i,t_i), y un punto final (q_f,t_f), la ecuación de interpolación lineal será la siguiente:

$q(t)=a*t+b$

Como ya conocemos el punto inicial y el final:

$q(t_i)=q_i$; $q(t_f)=q_f$

Con estos datos podemos realizar un sistema de ecuaciones para averiguar las incógnitas "a" y "b", y entonces sólo quedará evaluar q(t) en el instante de tiempo t para saber la posición correspondiente.

El gráfico correspondiente a los puntos de la figura 3 unidos mediante interpolación lineal sería el siguiente:

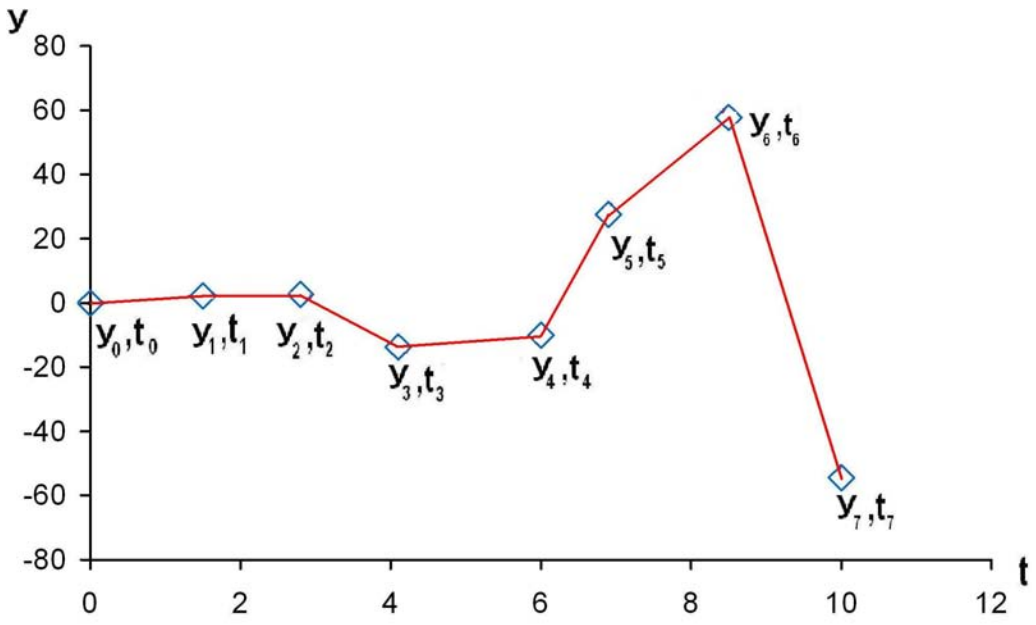


Figura 4- Interpolación lineal

Pero como podemos observar, la trayectoria que pasa por los puntos dados es abrupta (no es derivable, lo que implica que la velocidad tenga discontinuidades), y esto se traduciría en un movimiento del robot con cambios bruscos de velocidad.

Si queremos que la trayectoria sea suave (contínua y diferenciable de manera que la primera y segunda derivada, es decir, velocidad y aceleración, sean contínuas), la opción más usada son las funciones ***splines* cúbicas** (ver figura 5)**,** debido a que proporcionan un excelente ajuste a los puntos tabulados y su cálculo no es excesivamente complejo.**.** Estos polinomios de grado 3 tienen la siguiente forma:

$$S_i(t) = a_{0i} + a_{1i}(t - t_i) + a_{2i}(t - t_i)^2 + a_{3i}(t - t_i)^3$$

donde a_{0i}, a_{1i}, a_{2i}, a_{3i} son los coeficientes del *spline* que interpola entre los puntos (y_i,t_i) y (y_{i+1},t_{i+1}), obtenidos a partir del sistema de ecuaciones generado por el conjunto de funciones *spline* (posición en grados) y sus derivadas (velocidad) pertenecientes a todos los tramos entre los puntos a unir.

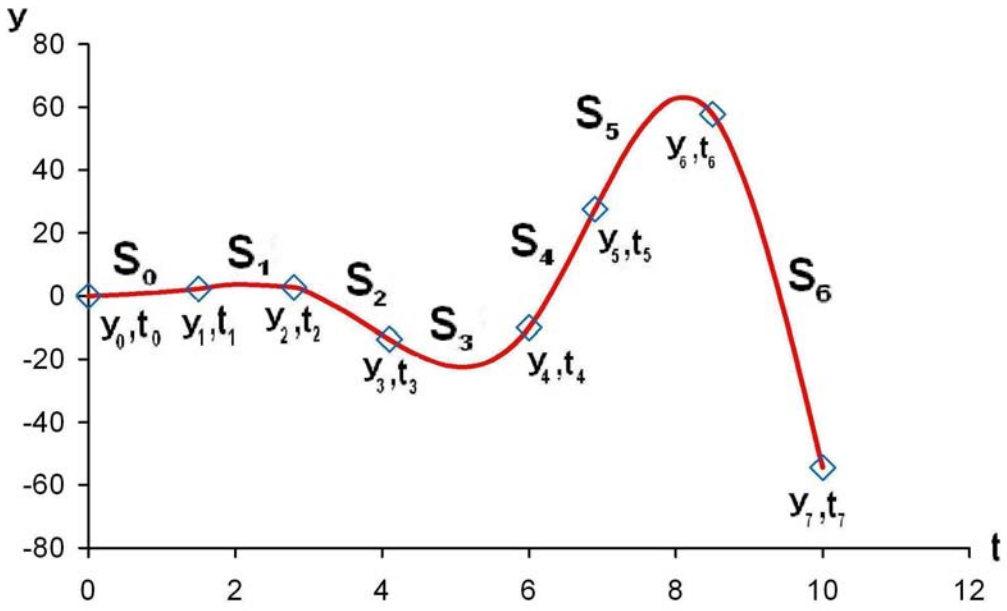


Figura 5 – Interpolación con *splines* cúbicos

Los polinomios S_{i-1} y S_i interpolan el mismo valor en el punto t_i, es decir, se cumple:

$$S_{i-1}(t_i) = y_i = S_i(t_i) \quad 1 \le i \le n - 1$$

Por lo que se garantiza que *S* es continuo en todo el intervalo. Además, se supone que *S*' (velocidad) y *S*" (acceleración) son continuas, condición que se emplea en la deducción de una expresión para la función del spline cúbico.

Aplicando las condiciones de continuidad del spline *S* y de las derivadas primera *S*' y segunda *S*", es posible encontrar la expresión analítica del spline, que será la siguiente:

$$S_i(t) = \frac{z_i}{6h_i}(t_{i+1} - t)^3 + \frac{z_{i+1}}{6h_i}(t - t_i)^3 + \left(\frac{y_{i+1}}{h_i} + \frac{z_{i+1}h_i}{6}\right)(t - t_i) + \left(\frac{y_i}{h_i} - \frac{z_i h_i}{6}\right)(t_{i+1} - t) \quad \textbf{(1)}$$

donde $h_i = t_{i+1} - t_i$ y z_i son incógnitas. Para determinar sus valores, se aplican las condiciones de continuidad que deben cumplir estas funciones y el resultado es el siguiente:

$$h_{i-1}z_{i-1} + 2(h_i + h_{i-1})z_i + h_i z_{i+1} = \frac{6}{h_{i-1}}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}) \quad \textbf{(2)}$$

Aplicando esta ecuación a i=1,2…n-1 se genera un sistema de n-1 equaciones lineales con n+1 incógnitas $z_0, z_1, \ldots, z_n$.

A los coeficientes z_0 y z_n se les suele dar el valor "0", y el sistema de ecuaciones expresado en forma matricial es el siguiente:

$$\begin{pmatrix} u_1 & h_1 & & & & \\ h_1 & u_2 & h_2 & & & \\ & h_2 & u_3 & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & & h_{n-2} & u_{n-1} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n-2} \\ z_{n-3} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{pmatrix}$$

Donde:

$$h_i = t_{i+1} - t_i$$

$$u_i = 2(h_i + h_{i-1}) - \frac{h_{i-1}^2}{u_{i-1}}$$

$$b_i = \frac{6}{h_i}(y_{i+1} - y_i)$$

$$v_i = b_i - b_{i-1} - \frac{h_{i-1}v_{i-1}}{u_{i-1}}$$

Resolviendo dicho sistema de ecuaciones se obtendrán los valores de $z_1, \ldots, z_{n-1}$. El algoritmo escrito en pseudocódigo para resolver el sistema es el siguiente:

```
input n,(t),(y)
```

```
for i=0,1,….,n-1 do
    hi := ti+1-ti
    bi := 6(yi+1-yi)/hi
end

u1 := 2(h0+ h1)
v1 := b1- b0
for i=2,3,….,n-1 do
    ui := 2(hi+ hi+1) – (hi-1 2 / ui-1)
    vi := bi - bi-1- hi-1(vi-1/ui-1)
end

zn :=0
z0 :=0
for i=n-1,…,1 do
    zi := (vi - hizi+1)/ui
end

output (z)
```

Dicho algoritmo recibe como entradas un vector con el conjunto de posiciones de los nodos y otro con los tiempos pertenecientes a cada posición. En nuestro sistema dichas posiciones representarán los grados (coordenadas en el espacio articular) de una articulación del robot en los tiempos pertenecientes al vector de tiempos. Como salida devuelve un vector con los valores de los coeficientes zi.

Con los valores de $z_0,z_1,....z_n$ ya resueltos, podremos evaluar el *spline* S_i en un instante de tiempo t (tiempos intermedios entre los dos nodos a unir) de forma eficiente empleando la siguiente función, para así obtener la posición correspondiente de la articulación del robot en ese instante:

$$S_i(t) = y_i + (t - t_i)[C_i + (t - t_i)[B_i + (t - t_i)A_i]] \quad \textbf{(3)}$$

Donde:

$$A_i = \frac{1}{6h_i}(z_{i+1} - z_i)$$

$$B_i = \frac{z_i}{2}$$

$$C_i = -\frac{h_i}{6}z_{i+1} - \frac{h_i}{3}z_i + \frac{1}{h_i}(y_{i+1} - y_i)$$

Mientras más tiempos intermedios utilizemos para evaluar en la ecuación de spline cúbico (ecuación 3), más suave será la trayectoria entre los nodos. Es decir, la función que describe dicha trayectoria tendrá más aspecto de continuidad en la velocidad y aceleración.

La ecuación de spline cúbico S_i interpola entre los nodos (y_i,t_i) y (y_{i+1},t_{i+1}). Por lo tanto si queremos generar una trayectoria que pase por todos los nodos, tenemos que aplicar la ecuación 3 a todos los tiempos intermedios que existan entre el tiempo del primer nodo y el tiempo del último nodo de esta.

Todo este procedimiento sirve, en lo que a nuestro sistema se refiere, para interpolar el valor en grados de una de las articulaciones del robot en un instante de tiempo determinado.

Como en realidad lo que nos interesa es generar una trayectoria para el efector final del manipulador (coordenadas en el espacio cartesiano), y el robot tiene 6 articulaciones, tendremos que obtener para cada posición cartesiana del efector final, el equivalente en grados de cada articulación que compone el manipulador, y repetir todo el proceso anterior para cada articulación del robot. El hecho de obtener el valor de los ángulos de las articulaciones (espacio articular) a partir de una posición en el plano cartesiano es, como ya se ha definido en el apartado 4.2.4, calcular la *cinemática inversa* del manipulador.

La interpolación mediante *splines* cúbicos permitirá, en lo que a este proyecto nos atañe, que el robot siga una trayectoria que pase por los puntos que el usuario le indique, en los tiempos deseados, sin cambios bruscos de velocidad durante el recorrido.

5. ANÁLISIS DEL SISTEMA

En este capítulo se describen paso a paso la arquitectura y funcionamiento del sistema a realizar, sin entrar en detalles sobre su implementación. Se explicará qué elementos forman parte del conjunto total del sistema y cómo interactúan entre ellos. En cuanto a la parte software, se distinguirán las distintas capas de programación (capa de presentación, capa de negocio y capa de datos), todo ello sin hacer referencia a código.

5.1. MÓDULOS

A continuación se hará una breve descripción de las características principales de cada módulo del sistema y su funcionamiento, sin concretar sobre su implementación o tecnologías utilizadas.

5.1.1. Robot

Se trata de un brazo robot con 6 grados de libertad, dotado de una pinza en su extremo final que le permite realizar operaciones PPO (*Pick and Place Operations*), es decir, coger objetos con la pinza y moverlos a otras coordenadas de su espacio de trabajo.

Está montado sobre una plataforma de madera donde se hayan inscritas una serie de semicircunferencias y radios que representan coordenadas en el plano horizontal.

Los usuarios del laboratorio remoto de robótica podrán enviar, a través de la interface web, las posiciones (nodos) por las que se desee que el robot efectúe una trayectoria suave. Estas posiciones son indicadas por el radio, semicircunferencia y plano dibujados en la tabla de madera, y por una serie de coordenadas de altura predefinidas (ver figura 6).

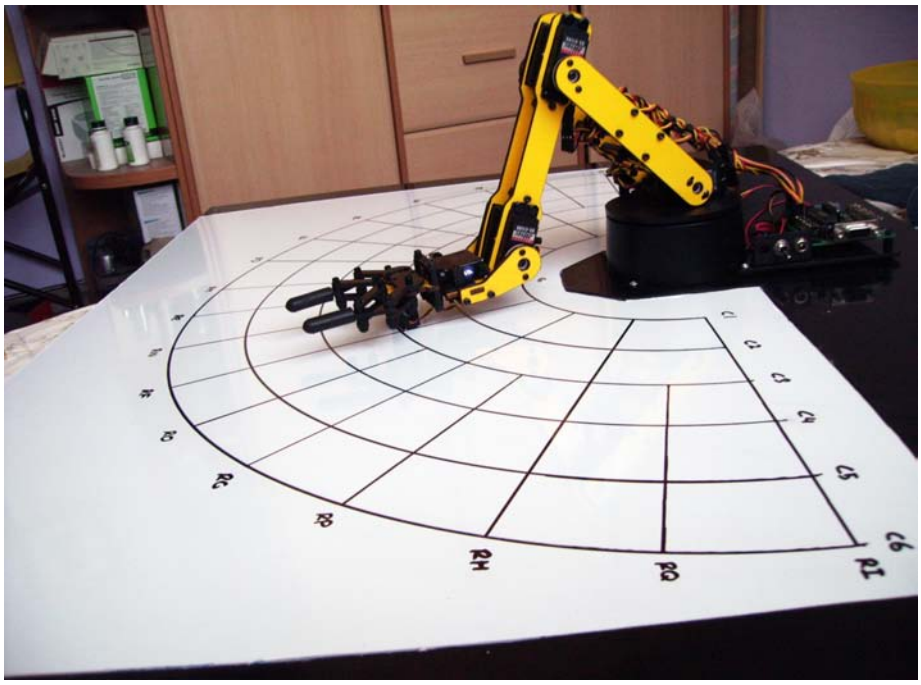


Figura 6- robot

5.1.2. Circuito de control SSC-32

Se haya acoplado a la base del brazo robot. Es un controlador de la empresa *Lynxmotion* que permite controlar hasta un máximo de 32 servos.

Sirve para enviar a los servomotores de las articulaciones las señales necesarias para ponerlo en movimiento. Recibe instrucciones a través del puerto serie de un PC (en este caso, del PC que hace de servidor web del sistema) que indican a qué posición ha de moverse cada servo y en cuanto tiempo realizar dicha operación.

Figura 7 – Servocontrolador SSC-32

5.1.3. Servidor web

Se trata de un PC en el cual se haya instalada una aplicación que le permite alojar páginas web. En su disco duro se hayan tanto las páginas que componen la interface gráfica del laboratorio remoto, como los programas necesarios para el funcionamiento del sistema global, interacción del usuario con el robot, bases de datos asociadas a la web, y software necesario para la codificación y transmisión en vivo del vídeo recogido por las dos cámaras de red.

5.1.4. Cámaras de red

Se dispone de dos cámaras IP de videovigilancia de la marca AXIS, modelo 207.

Una cámara IP (también conocida como cámara de red o *netcam*) es toda aquella que sea capaz de codificar y enviar vídeo directamente a la red (intranet o Internet) sin necesidad de un PC ni aparatos de codificación de vídeo externos. Cada webcam dispone de su propia dirección IP, y se puede acceder a la visión en directo del vídeo capturado accediendo a dicha IP.

En nuestro caso, las dos cámaras irán conectadas a la intranet de la UAB, y enfocarán desde distintos ángulos al robot para que los usuarios de la web del laboratorio remoto puedan visualizar en sus navegadores cómo el robot ejecuta las trayectorias que estos le han indicado.

5.1.5. Software de captura y codificación de vídeo

Como ya se ha comentado en el punto anterior, las cámaras de red codifican por sí mismas el vídeo capturado y lo emiten directamente a la red en un formato determinado. Teniendo en cuenta esto, en principio no debería existir la necesidad de tener en el servidor un software que capture el vídeo transmitido por las cámaras y lo codifique. Pero ciertas limitaciones de las cámaras utilizadas (no permiten la visualización simultánea por más de 10 usuarios mediante el método de acceso directo a su IP) han obligado a la implementación de otro sistema de visualización, ya que en las prácticas de la asignatura en la que se aplicará nuestro proyecto, estarán matriculados más de 20 alumnos, lo cual implica que es más que probable que más de 10 alumnos quieran acceder a la vez al laboratorio remoto con lo cual muchos de ellos no tendrían acceso al vídeo.

Dicho sistema consiste en lo siguiente (ver figura 8):

En el PC servidor se ha instalado un software que accede a la IP de las netcams y captura el vídeo emitido por estas, codificándolo en un nuevo formato y asignándolo a un puerto del servidor. Entonces, los usuarios que deseen visualizar el vídeo emitido por las cámaras, accederán a la IP del servidor y al puerto asignado, ahora sin restricciones de cantidad de usuarios.

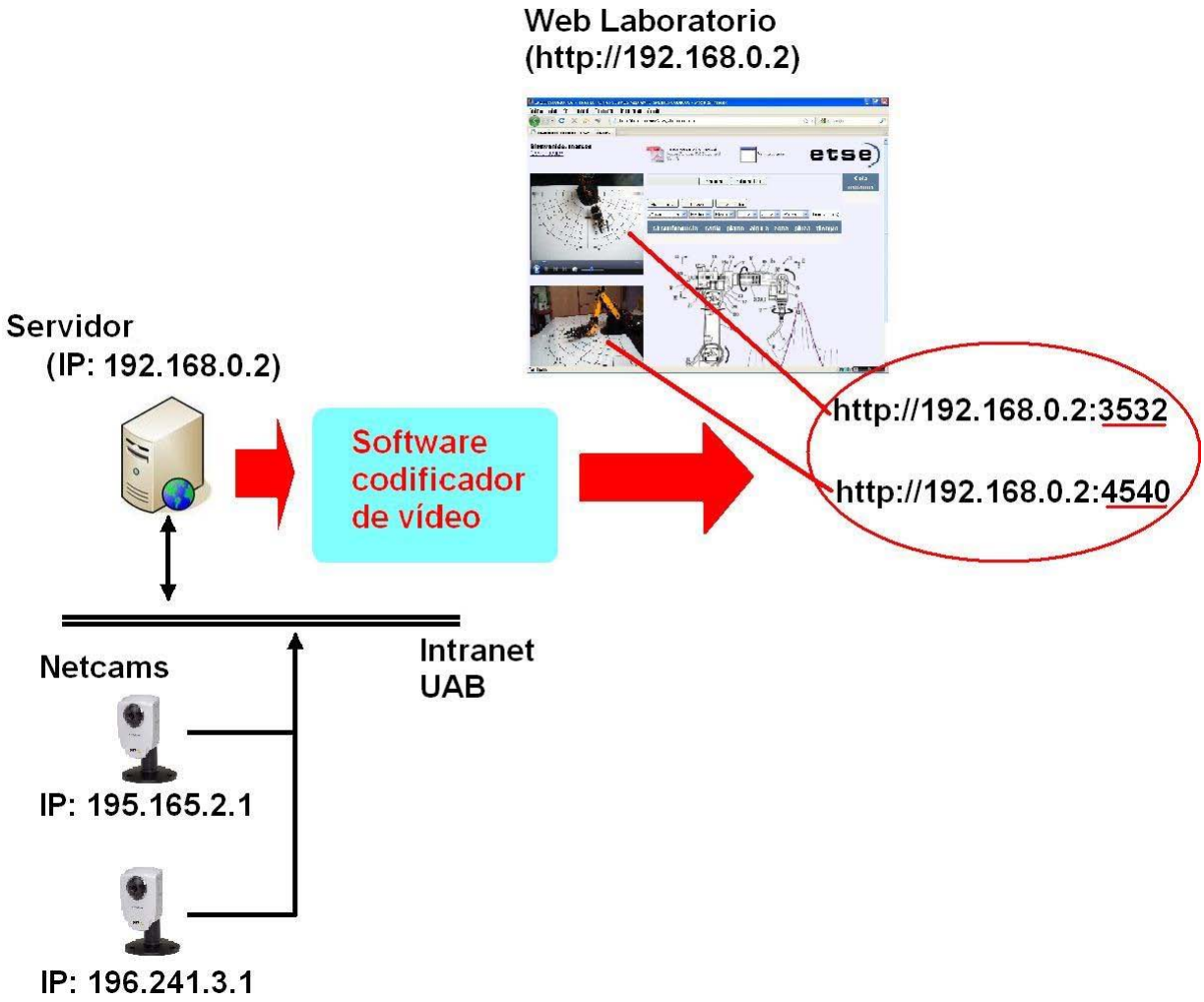


Figura 8 – Esquema de codificación de vídeo proviniente de las netcams

5.1.6. Páginas web

Este módulo lo componen un conjunto de páginas web que permiten al usuario interaccionar con el robot de manera remota. Concrétamente se han creado dos páginas. Una

para el registro y/o autentificación de usuarios, y otra para enviar órdenes al robot y su visualización en directo una vez ya registrados y logueados.

Cabe distinguir dos submódulos dentro de este mismo:

1- **Interfaz gráfica.**

2- **Código para el funcionamiento del robot y la interacción con el usuario.**

La tecnología utilizada para la creación de las dos páginas web permite separar perfectamente el código html que conforma la interfaz gráfica, del código (escrito en lenguaje C#) de la parte funcional (movimientos del robot, gestión de usuarios, etc.). Pero en este capítulo no se entrará en detalles sobre esta tecnología ni se comentará el código que implementa las páginas. Esto se hará en el capítulo 6.

5.1.6.1. Interfaz gráfica

Es lo que se muestra por pantalla al usuario que se conecta al laboratorio remoto de Robótica. Se ha intentado implementar de manera que la interacción del usuario con el sistema sea lo más intuitiva y agradable a la vista posible, de manera que no se necesite un complejo manual para hacer entender al alumno cómo funciona.

A continuación se presentan dos ilustraciones ue muestran el aspecto visual de dicha interfaz (ver figuras 9A y 9B):

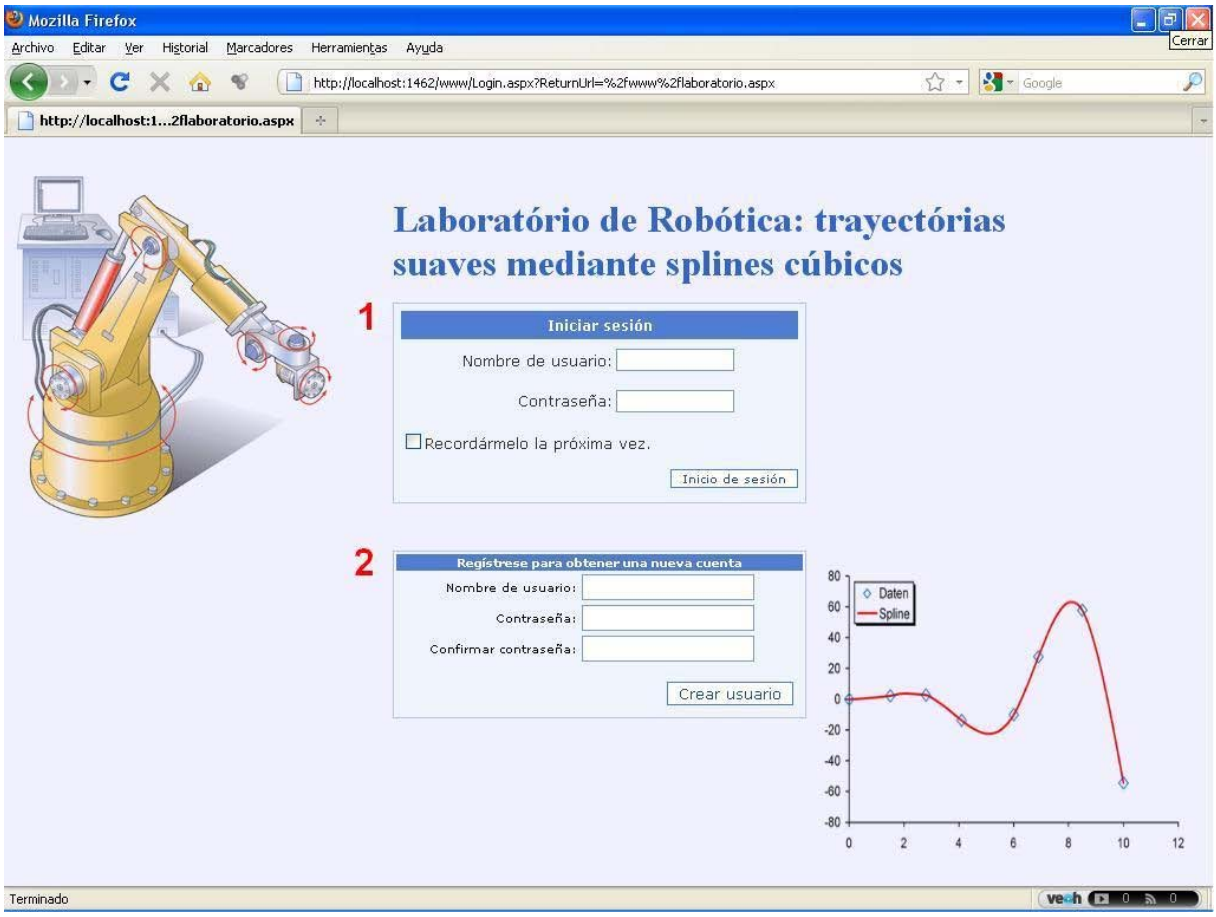
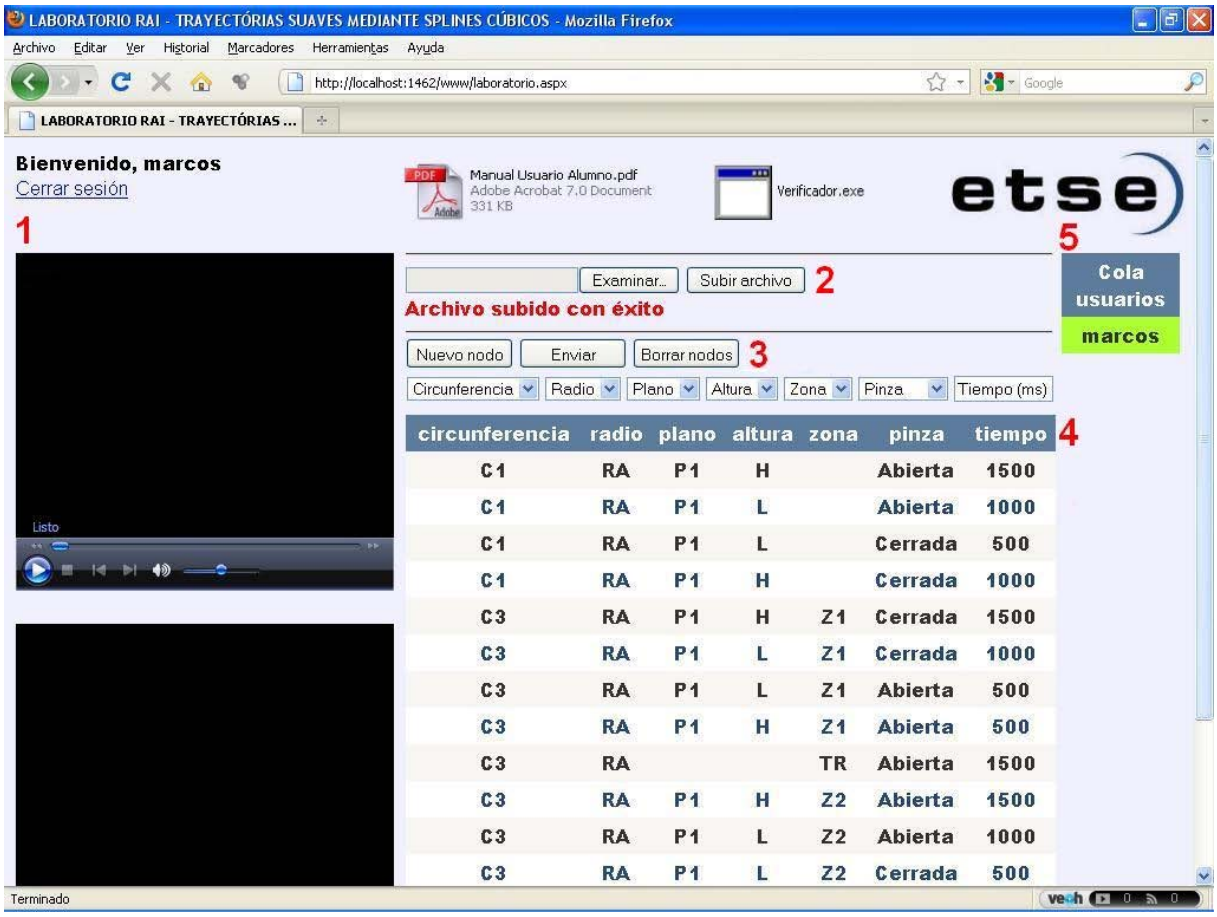


Figura 9A – Interfaz web de registro/login



circunferencia	radio	plano	altura	zona	pinza	tiempo
C1	RA	P1	H		Abierta	1500
C1	RA	P1	L		Abierta	1000
C1	RA	P1	L		Cerrada	500
C1	RA	P1	H		Cerrada	1000
C3	RA	P1	H	Z1	Cerrada	1500
C3	RA	P1	L	Z1	Cerrada	1000
C3	RA	P1	L	Z1	Abierta	500
C3	RA	P1	H	Z1	Abierta	500
C3	RA			TR	Abierta	1500
C3	RA	P1	H	Z2	Abierta	1500
C3	RA	P1	L	Z2	Abierta	1000
C3	RA	P1	L	Z2	Cerrada	500

Figura 9B – Interfaz web del usuario logueado

En las figuras 9A y 9B se han numerado los diferentes elementos gráficos que componen las páginas que forman parte de la interfaz. A continuación se explica la función de cada elemento:

- **Página de registro/login (figura 9A):**

    1. **Inicio de sesión:** Podemos observar dos cuadros de texto. Sirven para que el usuario que ya se encuentre registrado escriba su nombre de usuario y contraseña. Bajo los cuadros de texto se encuentra el botón de inicio de sesión.

    2. **Registro:** Hay una serie de cuadros de texto y un botón de confirmación para crear una nueva cuenta de usuario.

- **Página de usuario logueado (figura 9B):**

1. **Vista netcams:** Estas dos ventanas muestran en directo las imágenes de vídeo capturadas por las dos cámaras de red. Así el usuario logueado puede observar cómo el robot realiza las trayectorias que se le ordenan.

2. **Subir archivo:** Pulsando el botón "Examinar" se abre el navegador para la búsqueda del archivo de texto que el usuario puede enviar con los nodos de la trayectoria que desea que el robot ejecute. Cuando pulse el botón "Subir archivo" se enviará dicho archivo al servidor y el programa indicado por el usuario pasará a la cola de espera para ser ejecutado por el robot.

3. **Introducir nodos "on line":** En lugar de enviar un archivo de texto donde se indiquen los nodos de la trayectoria, el usuario logueado puede añadir uno a uno dichos nodos a través de las seis listas desplegables que permiten indicar la circunferencia, radio, plano, altura, zona y posición de la pinza que representan a cada uno de estos, e indicando en un cuadro de texto el tiempo en que desea que se llegue a cada nodo desde el nodo anterior. El botón "Nuevo nodo" sirve para enviar el nodo introducido por los desplegables a la lista de nodos que conformarán la trayectoria. El botón "Borrar nodos" permite borrar dicha lista, y el botón "Enviar" envía la lista a la cola de espera para ser ejecutada por el robot.

4. **Lista de nodos:** Cada fila de esta lista representa un nodo de la trayectoria introducida por el usuario logueado.

5. **Cola de usuarios:** Es la cola donde se almacenan los nombres de los usuarios cuyas trayectorias se encuentran a la espera de ser ejecutadas por el robot.

5.1.6.2. Códigode la capa de negocio.

Este submódulo será descrito con mayor detalle en el capítulo 6. Por ahora sólo es necesario saber que forma parte de la capa de negocio en la programación por capas del

sistema y que es la parte del proyecto más importante y más costosa en cuanto a tiempo de desarrollo, ya que es aquí donde residen todas las funciones y procedimientos que recogen la información introducida por los usuarios en la interfaz gráfica, obtienen o guardan datos en las bases de datos pertenecientes a la capa de datos y procesan toda esta información para que el robot ejecute los programas introducidos por cada usuario.

5.1.7. Bases de datos

Nuestro sistema dispone de dos bases de datos, que almacenan la siguiente información:

Una de ellas almacena los datos relacionados con las cuentas de usuario (contraseñas, nombre de usuario). Ya viene integrada en el entorno de programación utilizado para crear la web del laboratorio remoto.

La segunda base de datos, creada por el autor del proyecto, contiene tres tablas con distintos tipos de información:

- **Nombres de los usuarios** cuya trayectoria se encuentra a la espera de ser ejecutada por el robot.

- **Nodos de las trayectorias** de todos los usuarios.

- **Tabla de equivalencias** para transformar las coordenadas (circunferencia, radio, plano, …) de los nodos introducidos por los usuarios, en instrucciones escritas en el lenguaje aceptado por el circuito controlador del robot.

5.2. ARQUITECTURA FÍSICA

Los componentes físicos que forman parte de nuestro sistema se relacionan entre ellos de la siguiente manera (ver figura 10):

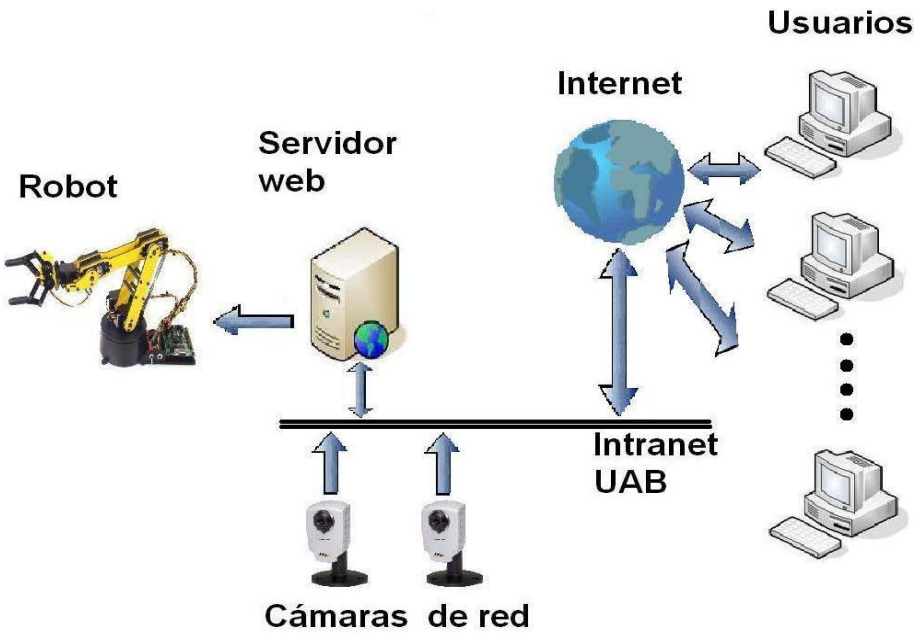


Figura 10– Arquitectura del sistema

El robot se comunica a través de su circuito de control, con el servidor web, desde el cual se le enviarán las órdenes necesarias para su movimiento.

El servidor, que aloja la web del laboratorio remoto, está conectado a la intranet de la UAB, donde también se hayan conectadas las dos webcams de red, a las direcciones IP de las cuales el servidor accederá para obtener las imágenes de vídeo registradas por estas.

La intranet recibe y envía datos a Internet. De esta manera los usuarios podrán acceder a la red de la UAB desde sus casas para interaccionar con la web del laboratorio remoto y enviar programas que serán ejecutados por el servidor para realizar en el robot trayectorias suaves.

5.3. FLUJO DE TRABAJO

Como podemos ver en el esquema de la figura 11, el funcionamiento del sistema fluye de la siguiente manera:

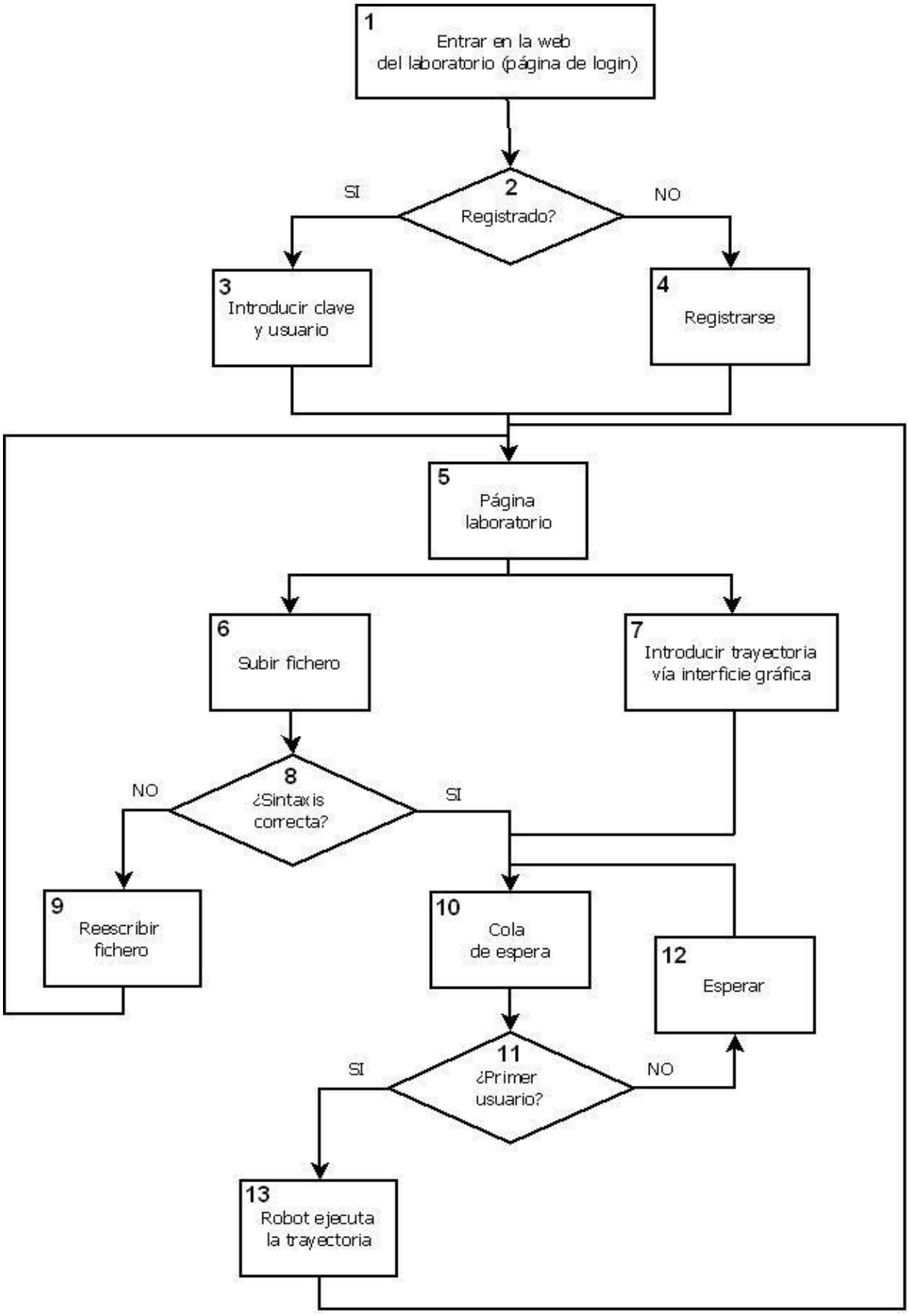


Figura 11 – Diagrama de flujo de trabajo

El usuario introduce en el navegador la dirección web del laboratorio para acceder a la página de login (1).

(2) Si ya tiene cuenta de usuario, introduce el nombre de usuario y la clave (3). Si no, crea una nueva cuenta con un nombre de usuario y contraseña (4).

A continuación accede a la página principal del laboratorio (5).

Una vez accedido al laboratorio, debe introducir las posiciones por las que desea que pase el actuador final del manipulador siguiendo una trayectoria suave. Esto se puede hacer de dos maneras:

a) Enviando al servidor un fichero de texto que contiene los nodos de la trayectoria especificados en un lenguaje de etiquetas (6).

b) Introduciendo las posiciones seleccionándolas de una serie de listas que aparecen en la interfaz web (7).

En el caso de haber escogido el método consistente en subir un archivo (paso 6), se comprueba si la sintaxis de este es correcta (8). Si no lo es, deberá reescribir el fichero (9) y volver a repetir el proceso (paso 6 o 7).

Una vez introducidos los nodos de la trayectoria, el usuario se añade a una cola de espera (10) en la cual se encuentran los programas con las trayectorias introducidas por cada usuario a la espera de ser ejecutadas por el manipulador.

El usuario se mantiene en espera (12) hasta que se encuentre el primero en la cola (11). Cuando así ocurra, el robot jecutará los movimientos necesarios para realizar la trayectoria correspondiente a los nodos introducidos por el usuario (13). Una vez ejecutada, se elimina de la cola el nombre del usuario y el robot vuelve a su estado de reposo, para seguidamente ejecutar la trayectoria del siguiente usuario de la cola (si existe). El usuario eliminado puede volver a iniciar el proceso a partir del paso 5 si así lo desea.

Nota: El Anexo 9 contiene un vídeo de ejemplo del funcionamiento del sistema.

5.4. PROGRAMACIÓN POR CAPAS

La separación de la programación del sistema en capas consiste en distinguir cláramente la parte relacionada con la interfaz gráfica (capa de presentación), los datos (capa de datos) y las parte de la programación que procesa los datos ingresados en la capa de presentación (capa de negocio).

Esta manera de programar ofrece múltiples ventajas, entre las cuales se encuentran las siguientes:

- Se asegura un trabajo de forma ordenada y separada.

- Cada capa está dividida según su funcionalidad. Cuando se quiere modificar el sistema basta con cambiar un objeto o conjunto de objetos de una capa. Esto se llama *modularidad.*

En ocasiones, además de las tres principales, pueden distinguirse más capas lógicas, pero en nuestro caso dividiremos el desarrollo en las tres nombradas. Añadir también, que las capas que componen nuestra solución se encuentran distribuidas en un solo ordenador, por lo tanto diremos que hemos utilizado una *arquitectura de tres capas y un nivel.*

En los siguientes subapartados se especifica qué módulos forman parte de cada una de las capas lógicas que componen nuestra solución.

5.4.1. Capa de presentación

Es lo que ve el usuario. También llamada *capa de usuario*, la capa de presentación contiene los objetos encargados de comunicar al usuario con el sistema mediante el intercambio de información, capturando y desplegando los datos necesarios para realizar alguna tarea. Esta capa se comunica únicamente con la capa de negocio. También es conocida como *interfaz gráfica* y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. En esta capa los datos se procesan de manera superficial por ejemplo, para determinar la validez de su formato o para darles algún orden específico.

El módulo que corresponde a esta capa en nuestro sistema sería la interfaz web del laboratorio remoto. Lo componen las clases y funciones que pertenezcan a los objetos visibles en la interfaz gráfica y el código html que la implementa.

5.4.2. Capa de datos

Aquí es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. Esta capa envía la información directamente a la capa de negocio para que sea procesada e ingresada en objetos según se necesite, esta acción se denomina *encapsulamiento.*

En nuestro proyecto, pertenece a esta capa el módulo formado por las dos bases de datos descritas en el apartado 5.1.7.

5.4.3. Capa de negocio

En esta capa se definen las reglas que se deben cumplir para una correcta ejecución del programa. Se comunica con las capas de presentación y datos. Recibe lo datos ingresados por los usuarios en la capa de presentación, los procesa, si es necesario obtiene datos de la capa de datos o los ingresa en esta, y devuelve resultados a la capa de presentación. También forman parte de la capa de negocio todas las aplicaciones software necesarias para el funcionamiento del sistema a desarrollar.

Forman parte de la capa de negocio en nuestro sistema, los siguientes elementos:

- Funciones y procedimientos que procesan los datos introducidos por los usuarios en la interfaz web del laboratorio remoto para enviar por orden al robot las instrucciones necesarias para la ejecución de las trayectorias y para mostrar los resultados por pantalla en la interfaz gráfica.

- Software de captura y codificación del vídeo emitido por las cámaras de red.

- *Drivers* de las cámaras de red.

En resumen, llegamos a la conclusión de que la programación por capas permite separar claramente los distintos elementos del sistema según su funcionalidad para así ofrecer una mayor modularidad y facilitar las tareas de modificación de código ya sea para corregir errores, añadir o quitar funcionalidades al sistema.

Como podremos ver en el capítulo 6, el entorno de programación utilizado en este proyecto facilita enormemente la separación del código en las tres capas descritas.

6. DISEÑO

En este capítulo se entrará en detalle sobre cada módulo del sistema explicando las tecnologías específicas utilizadas, así como la descripción de las entradas y salidas de cada una de las funciones, procedimientos y clases principales que forman parte de la implementación de la capa de negocio.

6.1. ROBOT

En el capítulo 5 se ha descrito superficialmente en qué consiste este módulo. A continuación se hará una descripción más precisa indicando detalles sobre su configuración física, funcionamiento (entradas y salidas) y las tecnologías específicas que podemos apreciar en el robot.

6.1.1. Descripción del módulo

Se trata del robot *Lynx 6*, de la empresa *Lynxmotion*. Es un robot de configuración antropomórfica (estructura similar a la de un brazo humano) de 6 grados de libertad, que permite realizar operaciones PPO. Para realizar los movimientos, posee una serie de servomotores situados en la base, en cada articulación y en la pinza (*gripper*) de su extremo final.

A continuación se muestra una imagen donde se pueden observar claramente los grados de libertad del robot, sistemas de referencia de sus articulaciones y el sentido del movimiento de cada una de ellas (ver figura 12).

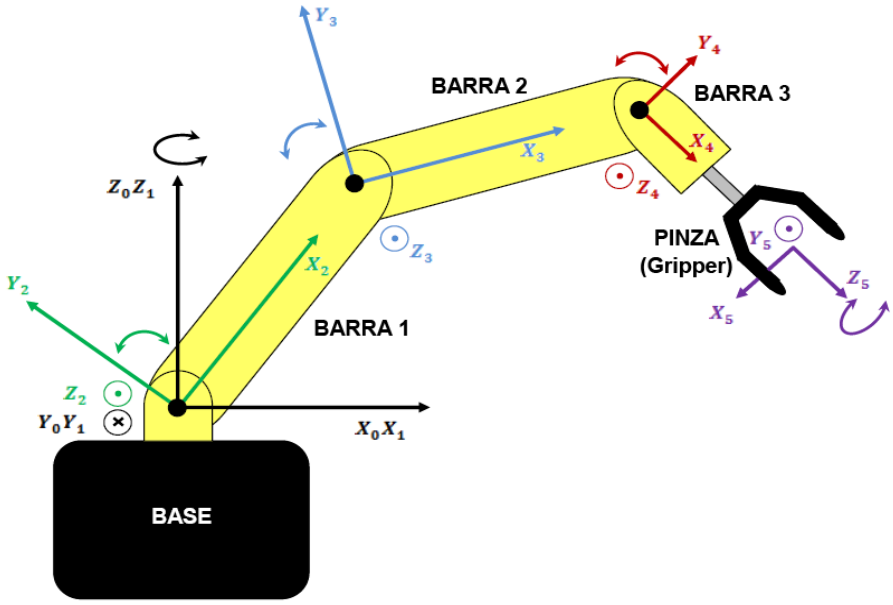


Figura 12 – Robot Lynx 6 y sistemas de referencia

Los servomotores reciben impulsos eléctricos desde un circuito de control (el circuito SSC-32), que será descrito en el apartado 6.2. De esta manera, el robot puede efectuar movimientos para alcanzar con el efector final (pinza) las coordenadas indicadas, coger objetos y colocarlos en la posición que se desee dentro del espacio de trabajo del manipulador.

El brazo robot se halla montado sobre una plataforma en la cual se han dibujado una serie de semicircunferencias y radios que definen parte de su espacio de trabajo (ver figura 13).

El espacio de trabajo total del manipulador viene dado por la unión de una serie de alturas predefinidas (ver figura 14), las semicircunferencias y radios dibujados en la tabla y los puntos intermedios entre estos puntos característicos.

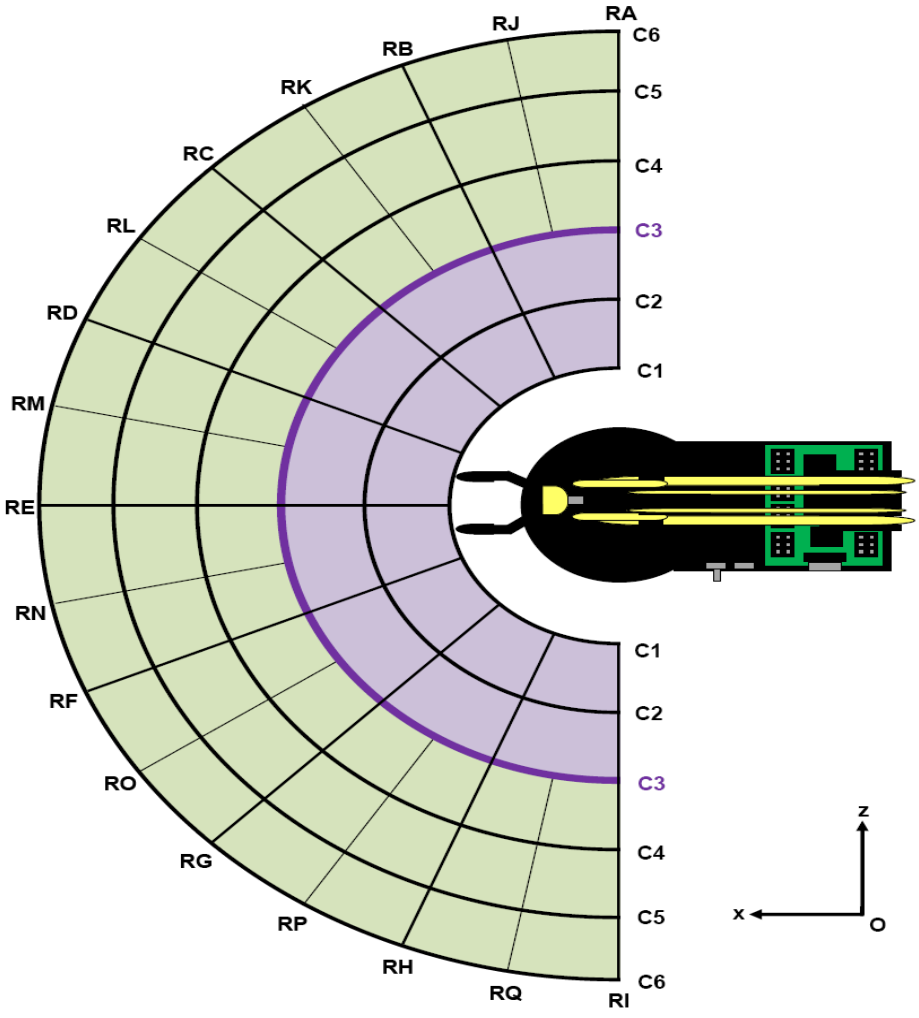


Figura 13 – Coordenadas en el tablero

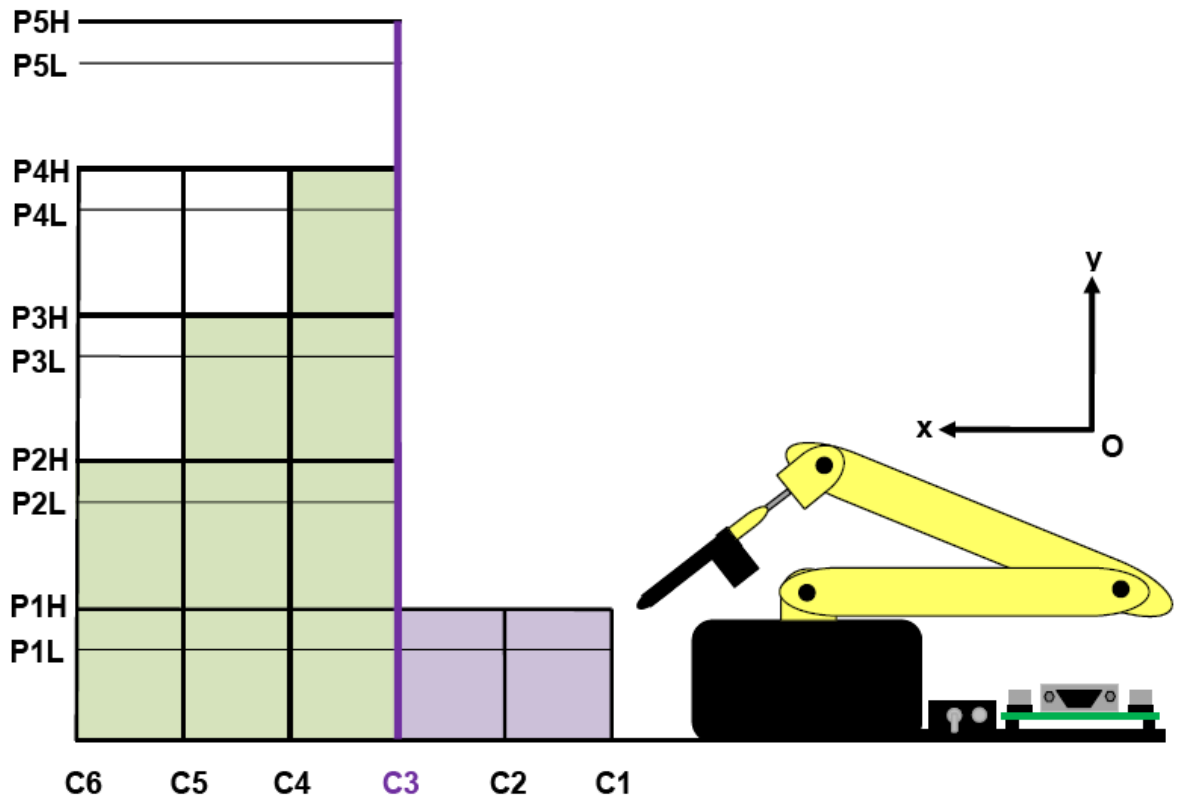


Figura 14 – Alturas predefinidas

En las ilustraciones de las figuras 13 y 14, se ha dividido el espacio de trabajo del manipulador en dos zonas, una de color violeta y la otra de color verde.

La "**Zona 1"** (color violeta) corresponde a las semicircunferencias C1, C2, C3, radios RA, RB, RC, RD, RE, RF, RG, RH, RI y las alturas P1L y P1H. En esta zona el robot adquiere la configuración que se muestra a continuación (ver figura 15). Dicha configuración le permite coger objetos que se encuentren muy cercanos a la base del manipulador.

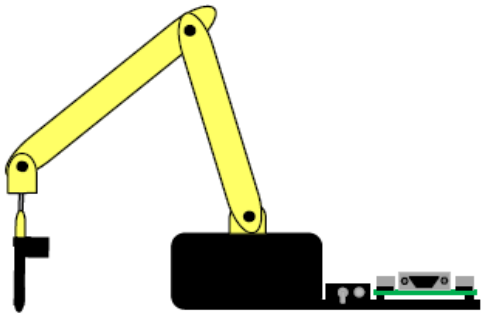


Figura 15 – Configuración en "Zona 1"

A la **"Zona 2"** (color verde) corresponden las semicircunferencias C3, C4, C5, C6, todos los radios existentes (RA, RB, RC,…, RQ) y todas las alturas. Pero no todas las alturas son accesibles desde cualquier semicircunferencia. La distribución es la siguiente:

- P1L y P1H: C3, C4, C5 y C6".
- "P2L y P2H: C3, C4, C5 y C6".
- "P3L y P3H: C3, C4 y C5".
- "P4L y P4H: C3 y C4".
- "P5L y P5H: C3".

En esta zona el robot adquiere una configuración diferente a la mostrada para la "Zona 1" (ver figura 16).

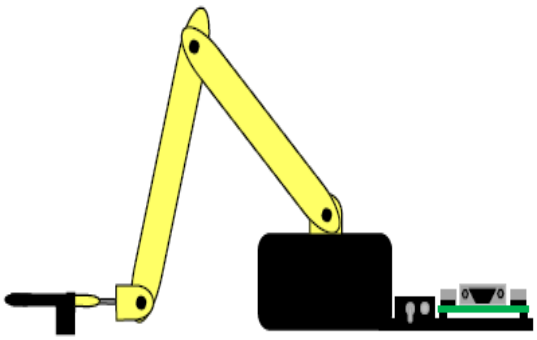


Figura 16 – Configuración en "Zona 2"

6.1.2. Entradas/Salidas

- **Entradas:** Pulsos eléctricos emitidos por el servocontrolador SSC-32 que se encuentra acoplado al manipulador.
- **Salidas:** Movimientos realizados por el robot.

6.1.3. Tecnologías utilizadas

Los componentes que forman el robot son básicamente:

- **Piezas de plástico** que conforman los enlaces, engranajes y la base del manipulador**.**
- **Tornillos de metal** para fijar algunas de las piezas.
- **Servomotores** que permiten el movimiento del robot.

Los elementos más importantes a comentar son los servomotores.

El robot *Lynx 6* contiene en sus articulaciones, pinza y base una serie de servomotores digitales de la marca HITEC.

Los servos HITEC se caracterizan por su calidad técnica y sus excelentes características mecánicas y electrónicas hacen que sean los más utilizados en el montaje de robots. Constan de los siguientes elementos:

- **Motor de corriente continua (DC):** Permite la movilidad al servo cuando se le aplica un potencial eléctrico. Todos los servos HITEC giran en el sentido de las agujas del reloj.
- **Engranajes reductores:** Reducen la velocidad del motor para acrecentar la capacidad de torque (o par motor).
- **Sensor de desplazamiento:** Generalmente un potenciómetro colocado en el eje de salida del servo que sirve para conocer la posición angular del motor.
- **Circuito de control:** Placa electrónica que implementa una estrategia de control de la posición por realimentación.

Disponen de tres cables (ver figura 17): dos cables de alimentación (positivo y negativo/masa) que suministran un voltaje 4.8-6V y un cable de control que indica la posición deseada al circuito de control mediante señales PWM (*Pulse Width Modulation*).

Fihgura 17 – cableado servo Hitec

Las señales PWM utilizadas para controlar los servos están formadas por pulsos positivos cuya duración es proporcional a la posición deseada del servo y que se repiten cada 20ms (50Hz).

Nuestro robot dispone de los siguientes modelos de servomotores:

Servomotor HITEC HS-475HB

Es un servo de dimensiones estándar en el que destacan su sistema de transmisión de karbonite de gran resistencia y un circuito de control de gran capacidad que ofrece una mayor potencia y una mejor resolución de centrado que transfiere toda la potencia al eje de salida con precisión y suavidad.

El manipulador empleado dispone de 5 servomotores de este modelo: uno en la base, 2 en la primera articulación (contando desde la base), otro en la segunda y el último en la tercera articulación (muñeca).

Estas son sus características técnicas (ver figuras 18A y 18B):

- **Sistema de Control:** Control por Anchura de Pulso. 1,5ms al centro.
- **Tensión de funcionamiento:** 4,8V a 6V.
- **Velocidad a 6V:** 0,18seg /60º sin carga**.**
- **Fuerza a 6V:** 5,5kg · cm.
- **Corriente en reposo:** 7,7mA.
- **Corriente en funcionamiento:** 180mA sin carga.
- **Corriente Máxima:** 1100mA.
- **Zona Neutra:** 5µseg.
- **Rango Trabajo:** 1100 a 1900µseg.
- **Dimensiones:** 38,8x19,8x36mm.
- **Peso:** 40g.
- **Rodamiento Principal:** Metálico.
- **Engranajes:** Karbonite.
- **Longitud del cable:** 300mm.

Figura 18A – Servo HITEC 475HB

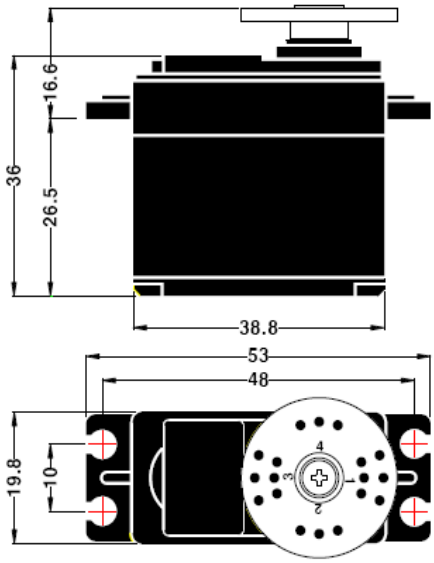


Figura 18B – Medidas servo HITEC 475HB

Servomotor HITEC HS-85BB

Este modelo (ver figura 19A) cuenta con un motor de alta potencia para mayor velocidad y par motor. Se encuentra disponible en dos versiones, fabricado en resina y en metal. Es un motor idóneo si se desean obtener altas prestaciones y movimientos precisos. La robustez del eje y de los rodamientos de bolas junto con su precisión de movimientos son las razones por las cuales este motor se encuentra en la articulación de la muñeca que otorga un movimiento de giro sobre el eje Z_5 (ver figura 12, apartado 6.1.1).

Sus características técnicas son las siguientes:

- **Sistema de Control:** Control por Anchura de Pulso. 1,5ms al centro.
- **Tensión de funcionamiento:** 4,8V a 6V.
- **Velocidad a 6V:** 0,14seg /60º sin carga**.**
- **Fuerza a 6V:** 3,5kg · cm.
- **Corriente en reposo:** 8mA.

- **Corriente en funcionamiento:** 280mA sin carga.
- **Zona Neutra:** 8µseg.
- **Rango Trabajo:** 1500 a 1900µseg.
- **Dimensiones:** 29x13x30mm (ver figura 19B).
- **Peso:** 19,2g.
- **Longitud del cable:** 160mm.

Figura 19A - Servomotor HITEC HS-85BB

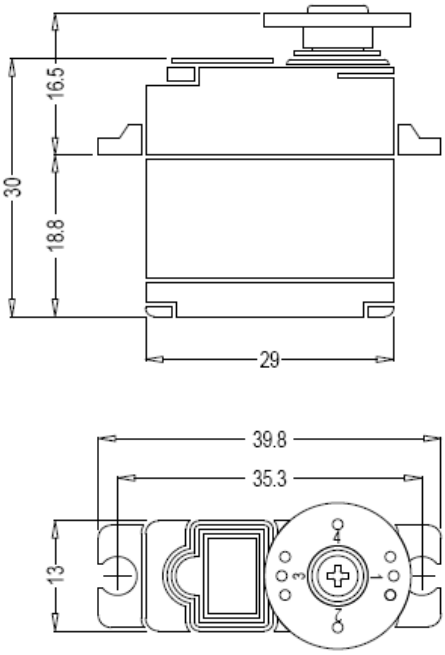


Figura 19B – Medidas servomotor HITEC HS-85BB

Servomotor HITEC HS-81

El HS-81 (ver figura 20A) es uno de los servos más populares de HITEC. Existen dos versiones (resina y metal), y ofrece un buen equilibrio entre velocidad y par motor. Se trata de un modelo diseñado para ser económico a la par que fiable.

Este servomotor se encuentra situado en la pinza (*gripper*) del robot.

Características técnicas (ver figura 20B):

- **Sistema de Control:** Control por Anchura de Pulso. 1,5ms al centro.
- **Tensión de funcionamiento:** 4,8V a 6V.
- **Velocidad a 6V:** 0,09seg /60º sin carga**.**
- **Fuerza a 6V:** 2,6kg· cm.
- **Corriente en reposo:** 9,1mA.
- **Corriente en funcionamiento:** 280mA sin carga.
- **Zona Neutra:** 8µseg.
- **Rango Trabajo:** 1500 a 1900µseg.
- **Dimensiones:** 29,8x12x29,6mm.
- **Peso:** 16,6g.
- **Longitud del cable:** 160mm.

Figura 20A – Servomotor HITEC HS-81

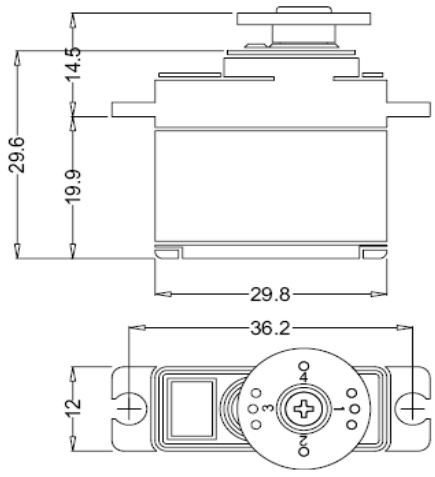


Figura 20B – Medidas servomotor HITEC HS-81

6.2. CIRCUITO SERVOCONTROLADOR SSC-32

En este módulo se hará una descripción detallada de qué es y para que sirve el circuito SSC-32 acoplado al brazo robot, especificando las tecnologías que lo implementan. También se comentarán las entradas y salidas del módulo y se especificarán los comandos básicos necesarios para su funcionamiento.

6.2.1. Descripción del módulo

Un SSC (*Serial Servo Controller*, Controlador Serie de Servos) es un dispositivo utilizado para controlar servos desde un PC a través del puerto serie. Los SSC aceptan comandos con un determinado formato desde el puerto serie del PC y los transforman en pulsos PWM que son enviados a los servos que se desea controlar.

El SSC-32 (Figura 21) es un controlador de la empresa *Lynxmotion* que permite controlar hasta un máximo de 32 servos. Dispone de un conjunto de funcionalidades entre las que destacan el control de servos por tiempo/velocidad/posición, movimiento síncrono de varios servos, consulta de posición de los servos y utilización de los pines de control de los servos como salidas digitales TTL. Además, dispone de 4 entradas (A, B, C y D) que pueden ser leídas de manera digital (bits) o de manera analógica (voltajes).

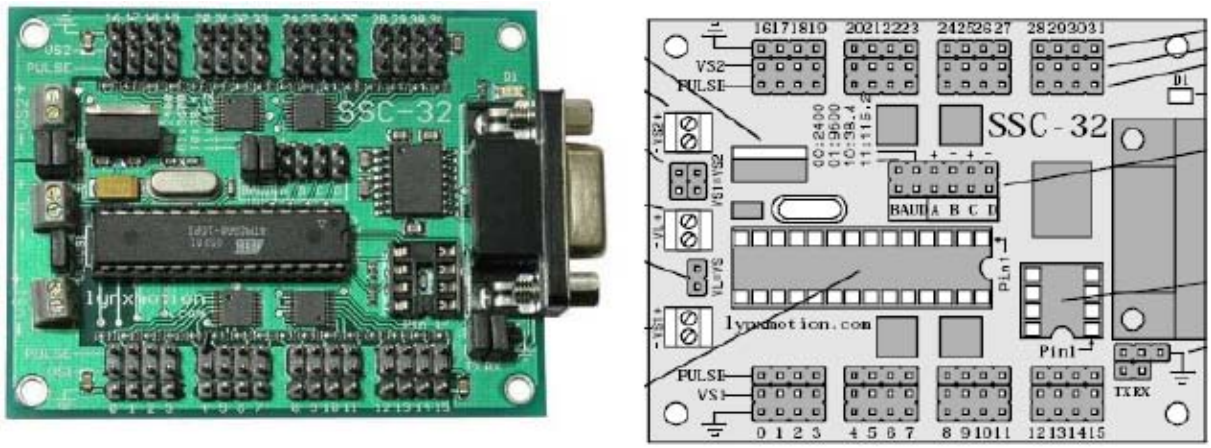


Figura 21 – Servocontrolador SSC-32

6.2.2. Entradas/Salidas

- **Entradas:** Comandos provinientes del puerto serie de un PC. En este proyecto, del PC servidor web del sistema.

- **Salidas:** Pulsos PWM destinados a uno o varios servos del manipulador.

6.2.3. Comandos básicos

El circuito controlador de servos SSC-32 posee una gran variedad de tipos de comando, entre los cuales destacan:

- Movimiento de servos

- Salida directa

- Salida de bytes

- Consulta de estado de movimiento

- Consulta de ancho de pulso

- Lectura de salidas digitales

- Lectura de entradas analógicas

Todos los comandos SSC-32 deben terminar en un carácter de retorno de carro (ASCII 13). Se pueden ejecutar varios comandos del mismo tipo de forma simultánea en un *Grupo de comandos*. Todos los comandos incluidos en un grupo de comandos se ejecutarán después de recibir el retorno de carro. Los comandos de tipos diferentes no pueden combinarse dentro del mismo grupo de comandos. Además, los argumentos numéricos para todos los comandos SSC-32 deben ser cadenas ASCII de los números decimales, por ejemplo, "1234". Algunos comandos admiten números negativos. El formato de ASCII no es sensible al uso de mayúsculas/minúsculas. Se ignorarán los espacios, tabuladores y saltos de líneas.

En lo que a este proyecto nos atañe, sólo se necesita conocer el formato de los **comandos de movimiento de servos.**

Dicho formato es el siguiente:

 # <ch> P <pw> S <spd> ... # <ch> P <pw> S <spd> T <time> <cr>

Donde:

<ch> Número de canal en formato decimal, 0-31.

<pw> Ancho de pulso en microsegundos, 500-2500.

<spd> Velocidad de movimiento en uS por segundos para cada canal (opcional).

<time> Tiempo en mS para el movimiento completo, que afecta a todos a los canales, 65535 máx (opcional).

<cr> Carácter de retorno de carro, ASCII 13 (necesario para iniciar la acción).

<esc> Cancela la acción actual, ASCII 27.

A continuación se muestra un ejemplo de comando de movimiento de servos aplicado al robot de este proyecto:

#0P2485#1P1580#2P1850#3P680#5P1415

Este comando haría que el circuito SSC-32 envíe los pulsos PWM necesarios para que el servo “0” se situe en la posición angular 2485,el servo “1” en la posición 1580, el “3” en la posición 680 y el “5” en la posición 1415.

Los servos del robot utilizado en nuestro sistema se han conectado al circuito SSC-32 de la siguiente manera:

- **Servo #0:** Base del manipulador.

- **Servo #1:** 1ª artiulación (contando desde la base).
- **Servo #2:** 2ª articulación.
- **Servo #3:** 3ª articulación.
- **Servo #4:** Pinza del extremo final del manipulador.
- **Servo #5:** Articulación de la muñeca del robot.

6.3. SERVIDOR WEB

En este apartado se hará primeramente una descripción del módulo. Posteriormente se describirán las entradas y salidas del módulo, se especificarán las tecnologías utilizadas y la implementación (instalación en el PC del software necesario para que funcione como un servidor web, configuración del servidor, etc). Finalmente, se hará una valoración sobre los problemas surgidos durante la implementación de dicho módulo y cómo se han solucionado.

6.3.1. Descripción del módulo

Se trata de un PC con sistema operativo *Windows XP Professional Edition*, al cual se le ha instalado un componente de *Windows* llamado **IIS (*Internet Information Services*).** Este componente proporciona al PC donde se halla este instalado, la capacidad para alojar y servir páginas web que podrán ser vistas por los usuarios de Internet o de la red donde se encuentre conectado dicho ordenador. Es decir, el componente *IIS* convierte un PC con sistema operativo *Windows XP* en un servidor web.

El robot se halla conectado al puerto serie del PC servidor, y este le enviará al robot los comandos de movimiento correspondientes a los programas que los usuarios del laboratorio remoto hayan enviado al servidor a través de la interfaz gráfica de las páginas web alojadas en el servidor.

6.3.2. Entradas/Salidas

- **Entradas:**
  - Vídeo procedente de las netcams conectadas a la red de la UAB.
  - Código html y C# que implementan la interfaz gráfica de las páginas web y el resto de funciones del sistema.
- **Salidas:**
  - Páginas web accesibles desde la red de la UAB.
  - Comandos para el movimiento de los servos del robot, a través del puerto serie.

6.3.3. Tecnologías utilizadas

Las tecnologías que forman parte de este módulo son básicamente las tres siguientes:

- **Sistema operativo *Windows XP Professional Edition***
- **PC**
- **IIS (*Internet Information Services*)**

A continuación se mostrarán las características principales de cada uno de estos tres elementos y el porqué se han elegido estos, centrándonos en el más importante, el IIS.

6.3.3.1. Sistema operativo *Windows XP Professional Edition*

Windows XP es el sistema operativo más usado en todo el mundo, perteneciente a la empresa *Microsoft.* Las ediciones más conocidas són la *Home*, orientada a usuarios de PC's domésticos y la *Professional,* dirigida al uso en empresas.

Windows XP Professional Edition se diferencia de la versión *Home*, entre otras cosas, en que posee componentes que le permiten funcionar como servidor web. Esta es la razón principal por la que se ha utilizado dicha versión del conocido sistema operativo en el PC de nuestro sistema.

Podría haberse utilizado cualquier otro sistema operativo no perteneciente a Microsoft ya que Windows no es el único que posee funciones de servidor. Otra opción podría haber sido instalar en una máquina con sistema operativo *Windows XP Home* Edition un servidor http como por ejemplo *Apache*. Pero hay un motivo por el cual se ha escogido esta vía:

Se ha optado por realizar la programación de las webs del sistema con la tecnología *ASP.NET*, y esta es sólo compatible con el componente ISS, disponible en *Windows XP Professional*. En realidad podrían haberse instalado otros sistemas operativos de *Microsoft* que también disponen del IIS (*Windows NT*, *Windows 2000* y *Windows Server 2003*) pero se ha creído conveniente usar la edición profesional por ser la más reciente y así evitar posibles incompatibilidades con las versiones del software de desarrollo web que se pretendía instalar y utilizar en el mismo PC que hace de servidor. Las razones por las que se ha escogido la tecnología ASP.NET para el desarrollo web del sistema serán explicadas en el apartado corespondiente al módulo "Páginas web" del capítulo actual.

6.3.3.2. PC

Se trata de un computador en el cual se hayan instalados el servidor web, el software de desarrollo web para la creación de las páginas web del laboratorio remoto y las aplicaciones de captura y codificación del vídeo proviniente de las cámaras de red.

Las características del PC utilizado son las siguientes:

- **CPU:** Pentium Duo 3 GHz.
- **RAM:** 2 GHz.
- **Sistma Operativo:** Windows XP Professional Edition SP3.

6.3.3.3. IIS (*Internet Information Services*)

Internet Information Services es un paquete de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del *Option Pack* para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS, [12].

Este servicio convierte a un ordenador en un servidor de Internet o Intranet, es decir, que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente (servidor web).

Los Servicios de IIS proporcionan las herramientas y funciones necesarias para administrar de forma sencilla un servidor Web seguro.

El servidor web se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas, como por ejemplo las ASP, ASP.NET o PHP.

Existen varias versiones de IIS. La versión para Windows XP Profesional y por lo tanto la utilizada en nuestro sistema , es la 5.1.

6.3.4. Implementación

Este apartado tratará únicamente sobre la instalación y configuración del IIS en el PC del sistema para su correcto funcionamiento como servidor web.

6.3.4.1. Instalación de IIS

Los pasos a seguir para la instalación de IIS en una máquina con sistema operativo Windows XP Professional son los siguientes, [13]:

1) Se introduce el CD de instalación del sistema operativo, se prepara el PC para que inicie desde el CD, y se procede al reinicio de la máquina.

2) Aparece un menú (ver figura 22). Seleccionamos la opción "Instalar componentes adicionales de Windows".

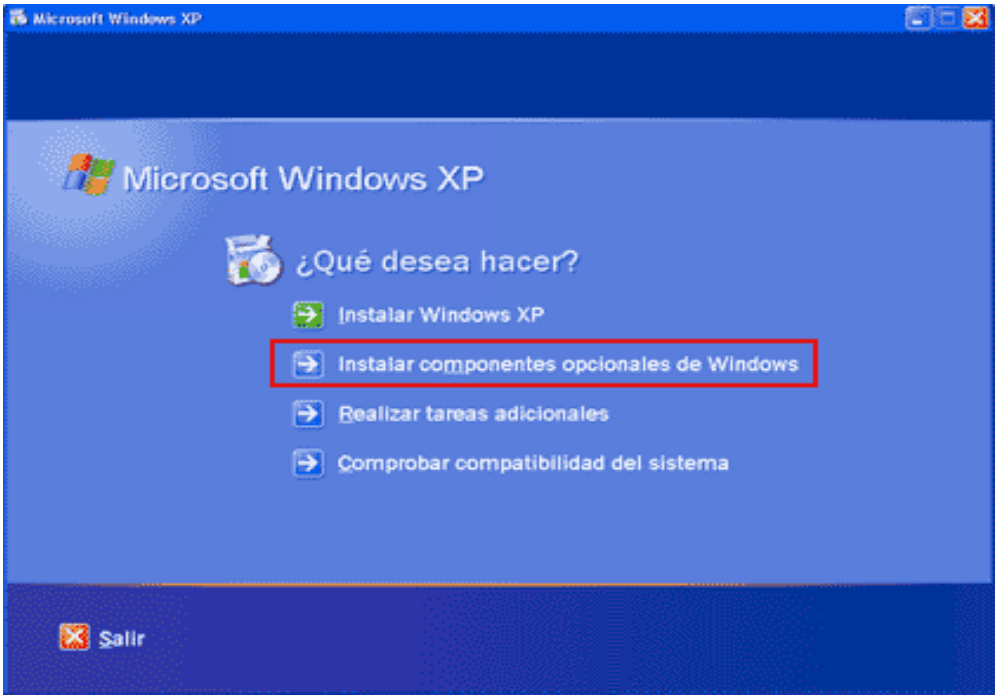


Figura 22 – Menú de instalación de *Windows XP Professional Edition*

3) En el menú "Inicio" de Windows, seleccionamos "Panel de control". A continuación "Agregar o quitar Programas" y finalmente seleccionamos la pestaña "Agregar o quitar componentes de Windows" (ver figura 23).

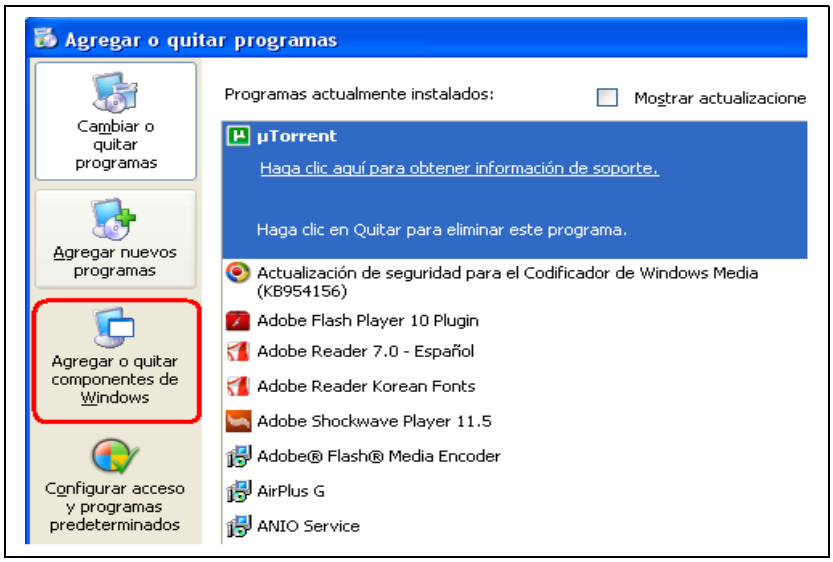


Figura 23 – Menú "Agregar o quitar programas"

4) Aparecerá la ventana del asistente para componentes de Windows. Marcar la casilla "Servicios de Internet Information Server (IIS) y seleccionar el botón "Siguiente".

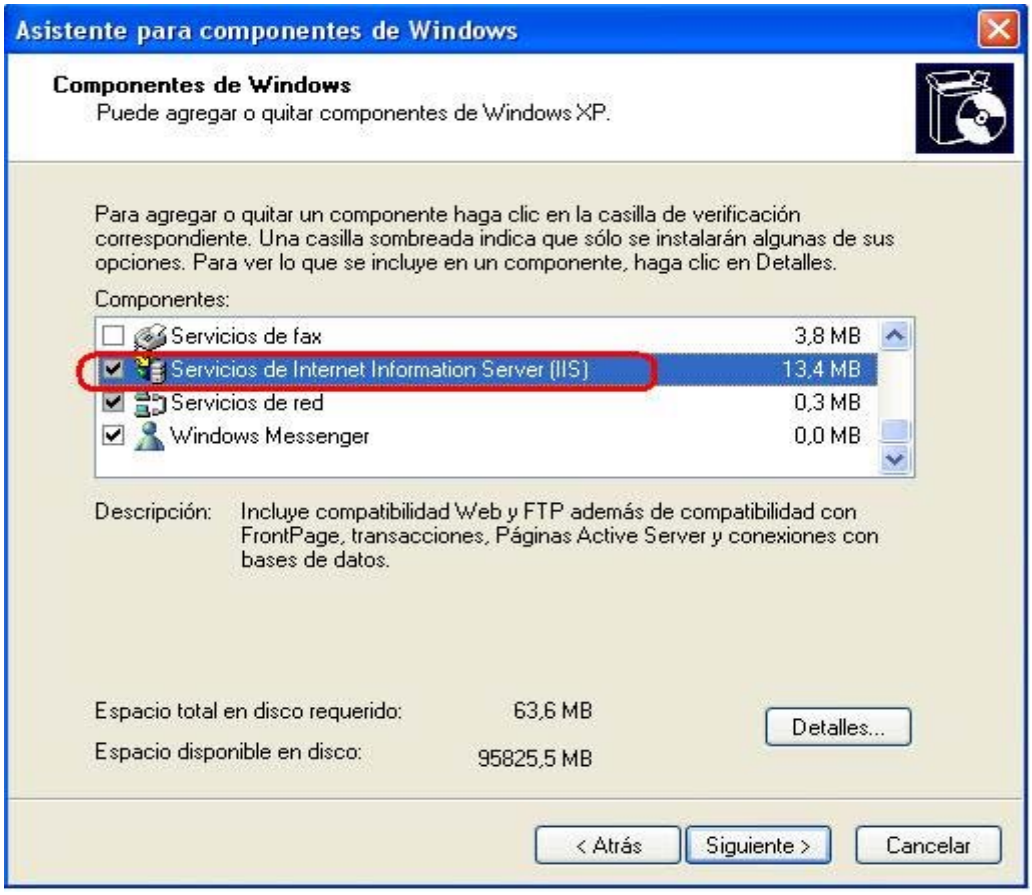


Figura 24 – Asistente para componentes de windows

Una vez realizados los 4 pasos, se instalará el componente IIS en el ordenador. Ya sólo quedará configurar el servidor correctamente para que esté listo para servir páginas web.

6.3.4.2. Configuración del servidor

Para que la máquina con IIS instalado pueda mostrar correctamente las páginas web y que estas funcionen con normalidad, hay que tener en cuenta tres puntos básicos:

- **Ubicación en el disco duro del sitio web predeterminado.**

- **Formato del documento por defecto.**

- **Configuración de permisos sobre carpetas y archivos.**

El **sitio web predeterminado** es la ruta donde se han de colocar las páginas web que desean ser mostradas desde el servidor. Dicha ruta es "C:\Inetpub\wwwroot". Es decir, que las páginas web que se coloquen dentro de la carpeta "\wwwroot" formarán parte del sitio web alojado en el PC servidor. Una de las páginas web de este proyecto se llama "laboratorio.aspx", entonces, una vez colocada en la carpeta "\wwwroot" del servidor, cuando este se encuentre totalmente operativo la forma de acceder remotamente a través de un navegador web sería escribiendo en su barra de direcciones la siguiente dirección: "http://<nombre_del_dominio>/laboratorio.aspx", donde en lugar del nombre del dominio puede escribirse la dirección IP del servidor. En el caso de este proyecto,no se ha dotado d enombre de dominio al sitio web, así que para acceder a este se hará a través de la IP del servidor.

Dentro de la carpeta del sitio web predeterminado podemos tener varios documentos web (en nuestro caso hay dos: "laboratorio.aspx" y "Default.aspx") y normalmente se desea que una de ellas sea la página a la cual se accederá por defecto al escribir en el navegador la dirección web sin especificar ningún archivo (en nuestro caso, escribiendo "http://<IP_del_servidor>"). A esta página se le conoce como **documento por defecto.** ISS tiene inicialmente asignados como documentos por defecto una lista de varios nombres, por ejemplo: "Default.aspx, "Index.aspx", "Default.asp", Index.asp", "Index.html" etc. en un orden concreto de preferencia por si hubiese más de una página que tuviese un nombre de la lista. Hay formas de cambiar ese orden y de añadir o eliminar nombres, pero en nuestro caso se ha decidido simplemente nombrar nuestra página principal como "Default.aspx".

En principio, si se trata de una página web sencilla sin subcarpetas que contengan otros archivos como por ejemplo bases de datos, no haría falta modificar nada en IIS para que el sitio web funcione correctamente. Pero nuestro sitio web alberga una carpeta en la cual hay bases de datos, y esta por defecto no permite el acceso a dichos archivos por parte de los usuarios de la web. Como en nuestro sistema es necesario poder acceder a las bases de datos tanto en el registro/login de usuarios como para mostrar por pantalla la cola de usuarios en espera y la cola de nodos de las trayectorias, debemos proceder a la **configuración de permisos sobre carpetas y archivos** y dar los permisos correspondientes a la carpeta que contiene dichas bases de datos. Esto se hace de la siguiente manera:

1. Seleccionamos en el menú de inicio de Windows "Inicio", "Panel de control", "Herramientras administrativas" y "Servicios de Internet Information Server".

2. Pulsar el botón derecho del ratón sobre el nodo "Sitio Web preseterminado" y a continuación seleccionar "Propiedades". En la ventana de propiedades nos vamos a la pestaña "Directorio particular" y marcamos la casilla "Escritura" (ver figura 25).

Figura 25 – Propiedades Sitio Web predeterminado

3. Pulsar el botón derecho del ratón sobre el nodo correspondiente a la carpeta a la que se le quieren dar los permisos de escritura, en este caso, "App_Data", y seleccionamos "Todas las tareas" y "Asistente para permisos". En el asistente para permisos marcamos la opción "Heredar la configiración de seguridad" y pulsamos "Siguiente" hasta que finalize el asistente (ver figura 26).

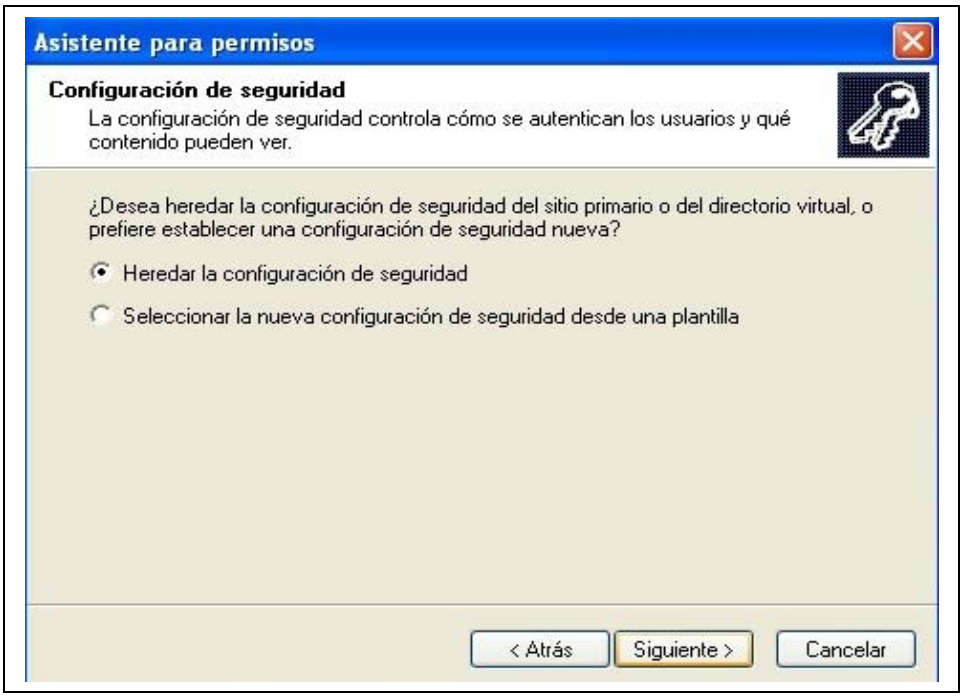


Figura 26 – Asistente para permisos

Una vez realizados estos pasos, el servidor web del sistema a desarrollar se encontrará completamente operativo.

6.3.5. Problemas surgidos y soluciones

El único problema acontecido durante el desarrollo de este módulo ha sido que al intentar acceder por primera vez a la web del laboratorio e intentar crear un nuevo usuario, aparecía un error que decía lo siguiente (ver figura 27):

Figura 27 – Error de servidor

Este problema se debía a que las dos bases de datos que nuestro sitio web utiliza, situadas en la carpeta "\App_Data", no tenían los permisos necesarios para su acceso tanto de lectura como de escritura. Este problema se solucionó siguiendo los pasos de configuración de permisos sobre carpetas y archivos descritos en el último párrafo de la sección anterior.

6.4. CÁMARAS DE RED

La web del laboratorio remoto creado en este proyecto permite la visualización en directo de los movimientos del robot gracias a dos cámaras de red que enfocan al manipulador y su espacio de trabajo desde dos ángulos distintos. Este apartado trata las características técnicas de estas cámaras, su funcionamiento, entradas/salidas del módulo y su implementación (instalación del software para el funcionamiento de las cámaras, situación física en el sistema, etc).

6.4.1. Descripción del módulo

Como ya se ha comentado en el capítulo 5, una cámara de red (también conocida como cámara IP, o *netcam* en inglés) es aquella capaz de codificar y enviar vídeo directaente a la red sin necesidad de un PC ni hardware externo de codificación.

Se dispone de dos netcams de videovigilancia **AXIS 207** (ver figura 28), que enfocan al robot y su espacio de trabajo desde distintos ángulos y envían el vídeo codificado simultáneamente en los formatos *Motion JPEG* y *MPEG-4* a la red de la UAB.

Figura 28 – Cámaras de red AXIS 207

Sus características técnicas más importantes son las siguientes:

Cámara

- **Resolución:** VGA.
- **Sensor de imagen:** CMOS de barrido progresivo de 1/4", VGA, RGB.
- **Objetivo:** 4,0 mm, F2.0, iris fijo, ángulo de visión horizontal: 55°.
- **Sensibilidad lumínica:** 1-10000 lux, F2.0.
- **Velocidad de obturación:** 1/10000s a 1/2s.

Vídeo

- **Compresión de vídeo:** MPEG-4, Motion JPEG.
- **Velocidad de imagen:** Hasta 30 imágenes por segundo.
- **Secuencias de vídeo:** MPEG-4 y Motion JPEG simultáneos. Frecuencia de imagen y ancho de banda controlables VBR/CBR MPEG-4.
- **Ajustes de imagen:** Compresión, rotación, color, brillo, contraste, balance de blancos. Superposición de texto.

Audio

- **Transmisión de audio:** Simplex.
- **Entrada/salida de audio:** Micrófono integrado.

Red

- **Seguridad:** Protección por contraseña.

- **Protocolos compatibles:** IP, HTTP, TCP, ICMP, QoS Layer 3 DiffServ, RTSP, RTP, UDP, IGMP, RTCP, SMTP, FTP, DHCP, UPnP, Bonjour, ARP, DNS, DynDNS, SOCKS, NTP.

General

- **Procesador y memoria:** ARTPEC-A, 32MB de RAM y 4MB de memoria flash.

- **Alimentación:** 4,9 – 5,1 V CC, 2,5 W máx.

- **Conectores:** RJ-45 Ethernet 10BASE-T/100BASE-TX, Auto-MDIX Bloque de terminales para 1 entrada, 1 salida de alarma y conexión de alimentación eléctrica alternativa.

- **Condiciones de funcionamiento:** 5° a 50° C. Humedad relativa: 20 a 80% (sin condensación).

- **Peso:** 180 gr.

Las cámaras de red AXIS 207 ofrecen una excelente calidad de imagen incluso en condiciones de poca luz, y una gran eficiencia del ancho de banda gracias a su implementación de MPEG-4, que incluye estimación del movimiento. MPEG-4 y Motion JPEG simultáneos permiten optimizar los sistemas tanto en calidad de imagen como en ancho de banda.

6.4.2. Entradas/Salidas

- **Entradas:** Imágenes desde dos ángulos distintos del robot y su entorno de trabajo.

- **Salidas:** Vídeo codificado simultáneamente en formatos *Motion JPEG* y *MPEG-4*.

6.4.3. Implementación

Se han comprado dos cámaras porque así el usuario del laboratorio remoto podrá observar los movimientos del robot desde dos ángulos distintos y así comprobar más fácilmente que la trayectoria de este pasa por las coordenadas exactas indicadas por el usuario.

Como las coordenadas son tridimensionales, se ha considerado que la mejor forma de colocar las cámaras de red para una óptima visualización del manipulador y su espacio de trabajo es aquella que muestre al robot desde las dos siguientes vistas:

- **Vista perpendicular al plano formado por los ejes x-z** del sistema de referencia orígen del manipulador (ver figura 13, apartado 6.1). De esta manera el usuario del sistema podrá visualizar con claridad el movimiento horizontal del robot y las coordenadas referentes a la semicircunferencia, radio y plano dibujadas en el tablero sobre el cual se sustenta.

- **Vista perpendicular al plano x-y** (ver figura 14, apartado 6.1)**.** El usuario visualizará el movimiento vertical del robot y comprovará que la trayectoria de su efector final pasa por las posiciones de altura que se le han indicado.

En cuanto a la puesta en funcionamiento de las netcams, hay que seguir los siguientes pasos:

1) **Conectar las cámaras a la red de la UAB.**

2) **Asignarles una dirección IP a cada una.**

3) **Configurar la contraseña de usuario de las netcams.**

El primer paso es sencillo. Simplemente hay que conectar el transformador de cada una de las cámaras a una toma de corriente, y utilizar dos cables de red RJ-45 para conectarlas a los puertos de un router perteneciente a la red de la UAB, donde también se encontrará conectado el servidor web.

Para el segundo paso, el fabricante proporciona una aplicación llamada *AXIS IP Utility*, que no necesita ser instalada en el PC servidor. Simplemente guardamos en su disco duro el archivo ejecutable de la aplicación, y lo ejecutamos. Se abrirá una ventana donde podremos visualizar una lista de todas las netcams conectadas a la red, y automáticamente serán asignadas las IP's a las cámaras que no tuvieran ya una (ver figura 29). La aplicación permite también la opción de configurar manualmente una IP para cada cámara.

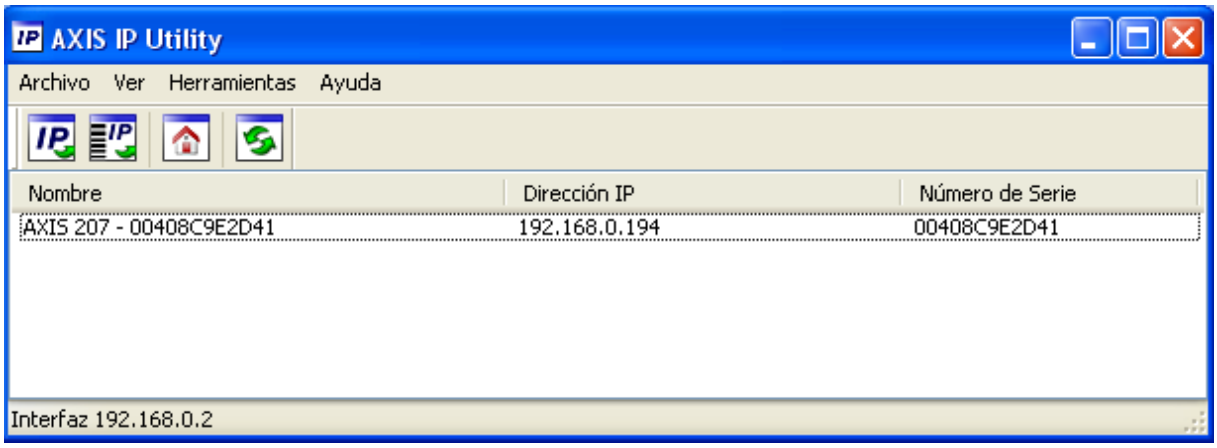


Figura 29 – *AXIS IP Utility*

Ahora debemos configurar la contraseña de acceso a las netcams. Esto es muy sencillo, simplemente al intentar acceder a una de las webcams por primera vez aparecerá el cuadro de diálogo "Configure root password" (ver figura 30). El nombre de usuario es "root", no se puede modificar. Escribimos la contraseña dos veces (en los dos cuadros de texto que aparecen en el cuadro de diálogo) y hacemos click en el botón "Ok".

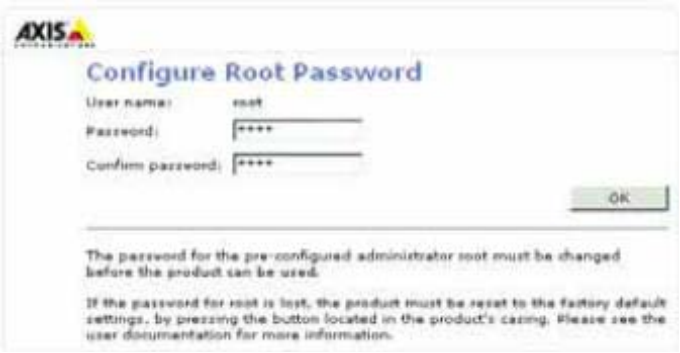


Figura 30 – Configuración de la contraseña

Una vez seguidos estos tres pasos, ya tenemos las cámaras de red listas para poder acceder a las imágenes en directo desde la red interna de la UAB. Desde fuera de la intranet no se podrá acceder directamente a las cámaras, será el servidor web del sistema el que acceda a estas y a través del software de codificación de vídeo que se presentará en el siguiente apartado, se cambiará el formato del vídeo obtenido de las netcams y se enviará a Internet a través de un puerto del servidor.

Además, accediendo directamente a la IP de las cámaras sólo se permite un máximo de 10 usuarios conectados por cada netcam. Esta restricción obliga a buscar un método que permita el acceso a más usuarios, ya que se prevee que los alumnos matriculados en la asignatura y que por consiguiente necesitarán utilizar simultáneamente el laboratorio remoto serán más de 50. Por lo tanto, procesar el vídeo a través del PC servidor y realizar el *streaming* a través de este utilizando un software específico es la mejor opción ya que la limitación de usuarios conectados la marcará la capacidad de ancho de banda del servidor web y no la de las cámaras de red.

6.5. SOFTWARE DE CAPTURA, CODIFICACIÓN Y *STREAMING* DE VÍDEO

A continuación se hablará del software que se ha utilizado para recodificar el vídeo proviniente de las cámaras de red y prepararlo para que sea accesible desde cualquier PC conectado a Internet.

6.5.1. Descripción del módulo

Como se ha comentado en capítulos anteriores, las cámaras de red codifican por sí mismas las imágenes obtenidas en algún formato de vídeo (en nuestro caso en *MPEG-4 y Motion JPEG*) sin necesidad de hardware externo ni conexión a un PC. Pero el vídeo codificado por las netcams que se han utilizado, sólo puede ser visualizado accediendo a la IP de cada cámara desde ordenadores conectados a la intranet de la UAB.

Se pretende que el laboratorio remoto de Robótica sea accesible desde cualquier ordenador con conexión a Internet, por lo tanto, necesitamos que el vídeo procedente de las netcams sea enviado a la red global a través del servidor web de nuestro sistema, incrustado en la interfaz gráfica de la web del laboratorio.

Como el vídeo será incrustado en la interfaz web y no todos los usuarios del laboratorio remoto usarán el mismo sistema operativo en sus PC's ni el mismo navegador web, se ha buscado un formato para las imágenes retransmitidas que sea soportado por la mayoría de navegadores web. El formato elegido ha sido el WMA (*Windows Media Archive*).

Una vez recodificado el vídeo al formato escogido, hay que redirigir el flujo de vídeo hacia un puerto del servidor web para que pueda ser transmitido en vivo hacia Internet. A esto se le denomina ***streaming* de vídeo en vivo**.

Cuando los usuarios del laboratorio visualicen en sus navegadores el vídeo, estarán accediendo a la IP del servidor y el puerto al que ha sido redirigido, en lugar de acceder a la IP de las netcams.

6.5.2. Entradas/Salidas

- **Entradas:** Flujos de vídeo procedentes de las netcams en formatos *MPEG-4* y *Motion JPEG*.

- **Salidas:** *Streaming* de los flujos de vídeo de entrada recodificados a formato *WMA* hacia dos puertos del servidor.

6.5.3. Tecnologías utilizadas

Para la **captura de vídeo** procedente de las netcams, el fabricante AXIS proporciona el controlador *AXIS Video Capture Driver*.

Para el ***streaming* de vídeo** en el servidor, se ha utilizado el software gratuíto de *Windows*, WME (*Windows Media Encoder*). Esta aplicación obtiene el vídeo, en este caso proviniente de una cámara de red AXIS 207 capturado por el controlador Axis Video Capture Driver, lo recodifica al formato de vídeo WMA y lo redirige a un puerto del PC servidor.

Existen otros programas gratuítos para la codificación y *streaming* de vídeo compatibles con el sistema operativo utilizado en nuestro servidor, por ejemplo el *VLC Media Player* o el

Flash Media Encoder, pero el único que ofrece una total compatibilidad con el controlador de vídeo de AXIS es el WME.

6.5.4. Implementación

Para poder llevar a cabo correctamente las tareas de captura de vídeo, recodificación y *streaming*, debemos seguir en orden los siguientes pasos:

1) **Instalación del controlador Axis Video Capture driver.**

2) **Instalación de Windows Media Encoder.**

3) **Configuración de WME para el streaming del vídeo procedente de las netcams AXIS 207.**

Los pasos 1) y 2) son sumamente sencillos. Solamente hay que ejecutar los respectivos archivos de instalación y seguir las indicaciones que nos muestran los cuadros de diálogo. La instalación no tiene dificultad alguna, así que no se entrará en detalles sobre ello.

Para realizar la tarea de streaming del vídeo procedente de alguna de las dos cámaras conectadas a la red y operativas, debemos hacer lo siguiente:

i) Ejecutamos la aplicación WME. Se abrirá la siguiente ventana (ver figura31) y a continuación escogemos la opción “Broadcast a live event”:

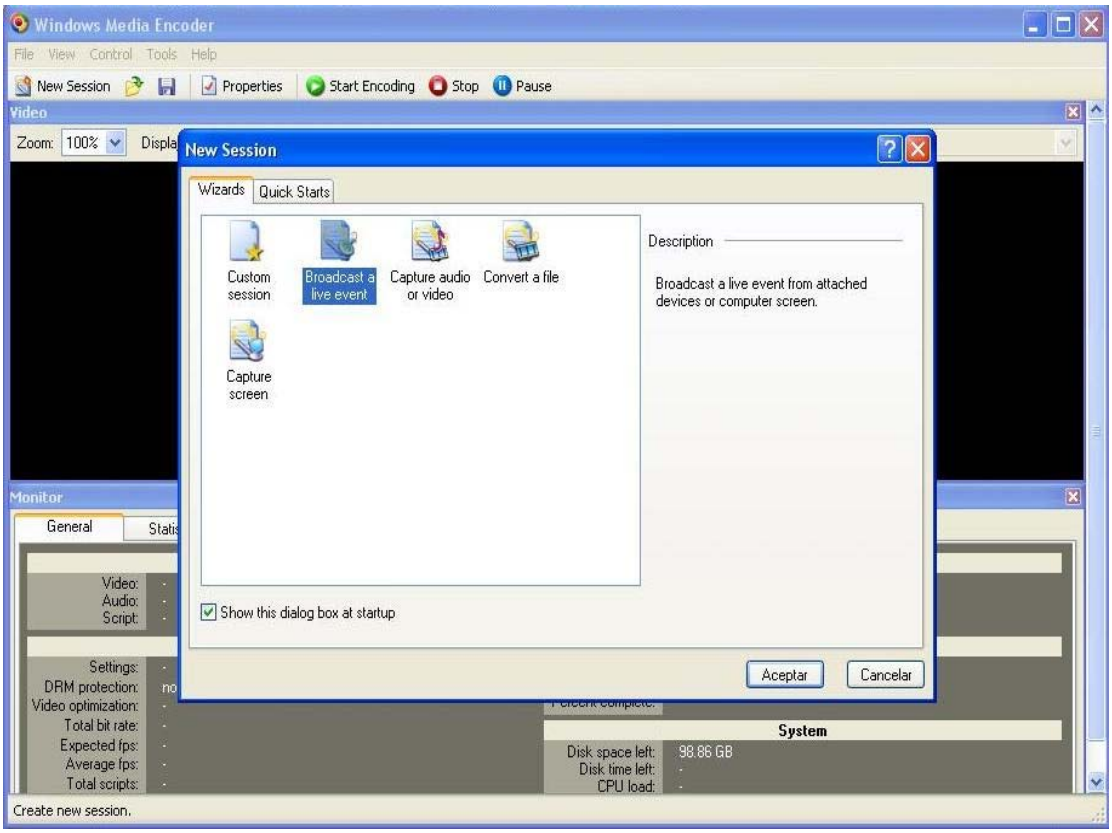


Figura 31 – Windows Media Encoder

ii) Se abrirá el asistente de nueva sesión (ver figura 32). Seleccionamos la casilla "Video:" y en la lista desplegable que hay a su derecha seleccionamos el controlador AXIS MJPEG Capture Driver (es el controlador que hemos instalado en el paso 1 nombrado al principio de este apartado). Como no necesitamos audio para este proyecto, no seleccionamos la casilla "Audio:".

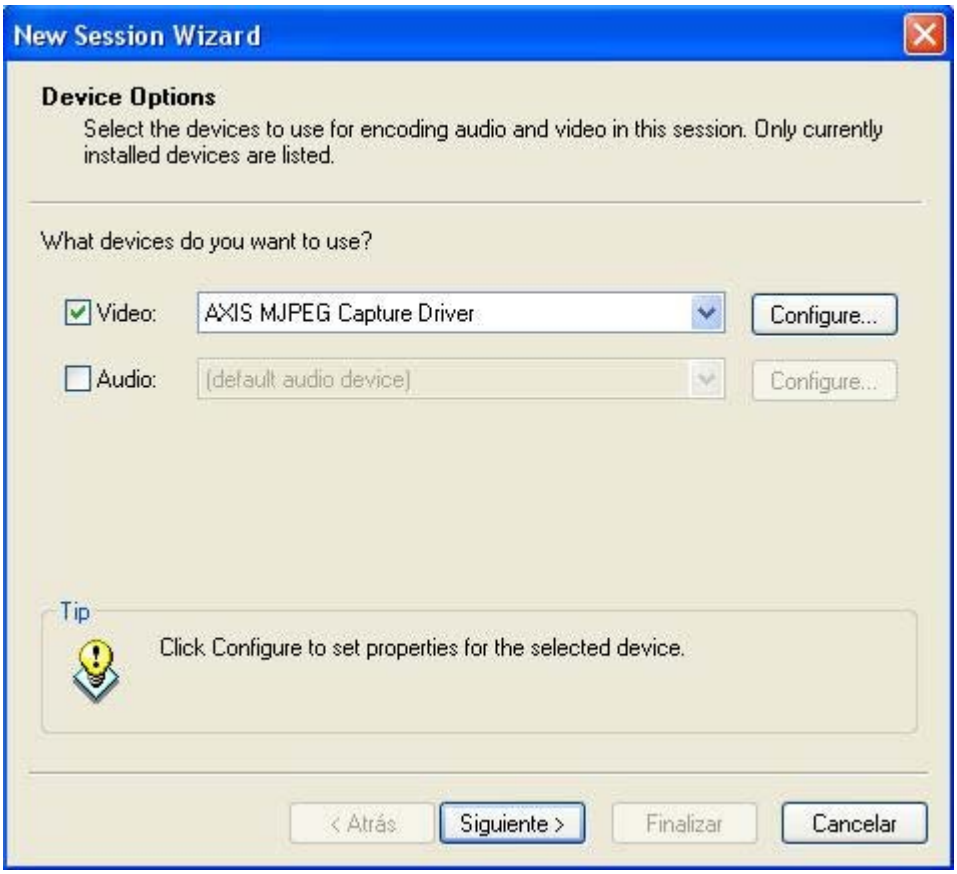


Figura 32 – Asistente de nueva sesión

iii) Hacemos click en el botón "Configure…" que hay a la derecha del desplegable perteneciente a la opción "Video:". Se abrirá la ventana de configuración del controlador de captura de vídeo seleccionado (ver figura 33). El cuadro de diálogo se divide en tres grupos de opciones: "Network", "Video" y "PTZ Control", de los cuales para nuestro proyecto sólo nos interesan los dos primeros.

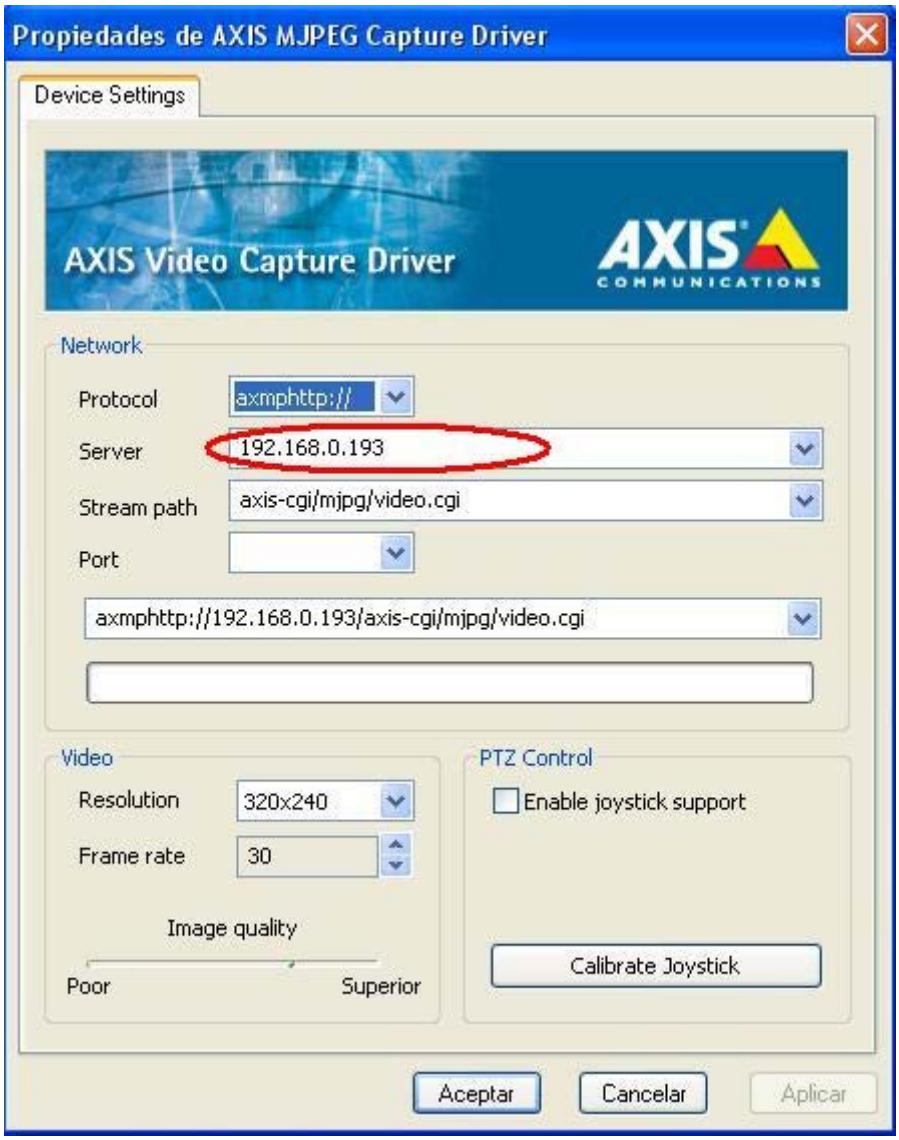


Figura 33 – Configuración del controlador de vídeo AXIS

En el grupo "Network", abrimos la lista desplegable a la derecha de la etiqueta "Server" y seleccionamos la IP de la cámara cuyo *streaming* se desea realizar, y en el grupo "Video" seleccionamos la resolución del vídeo a mostrar (desplegable "Resolution"). Una vez hecho esto, pulsamos "Aceptar" en la ventana de configuración del controlador de vídeo, y "Siguiente" en el asistente de nueva sesión.

iv) En el siguiente paso del asistente se nos pide escoger el método de emisión (ver figura 34). Seleccionamos la opción "Pull from the encoder" y pulsamos "Siguiente".

Figura 34 – Método de emisión

v) Seleccionamos un puerto libre para redirigir el flujo de vídeo capturado (ver figura 35) y pulsamos “Siguiente”.

vi) Aparece el menú de opciones de codificación (ver figura 36). En el desplegable perteneciente a “Video” elegimos la opción “Live broadcast video (CBR)” y pulsamos “Siguiente”.

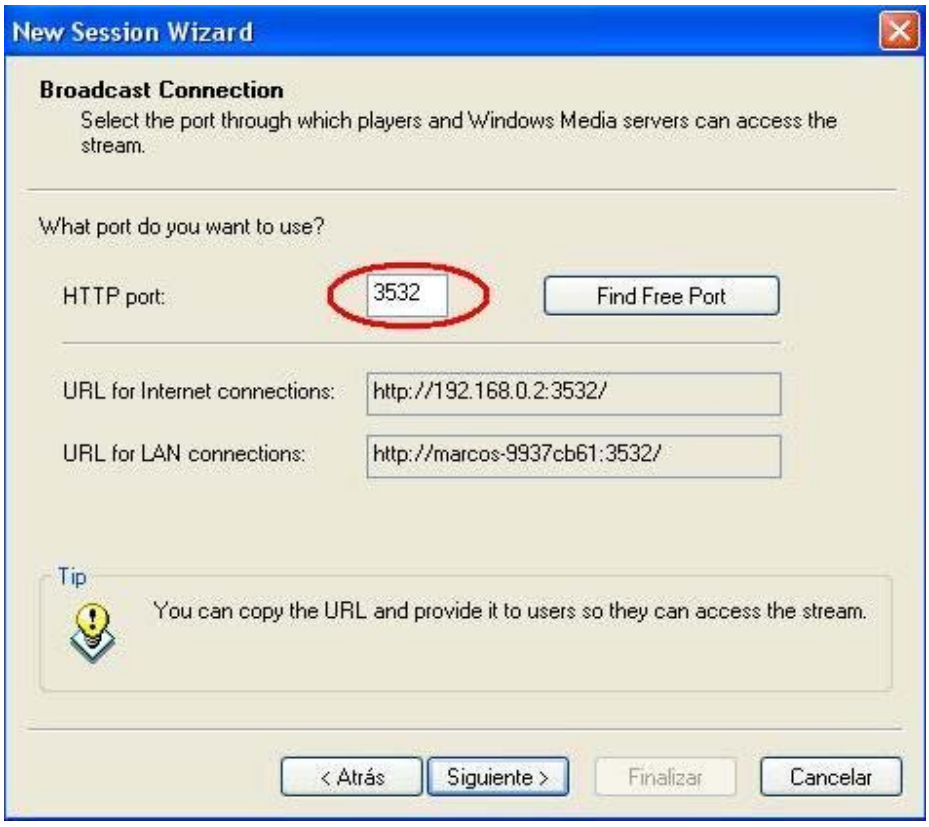


Figura 35 – Selección de puerto

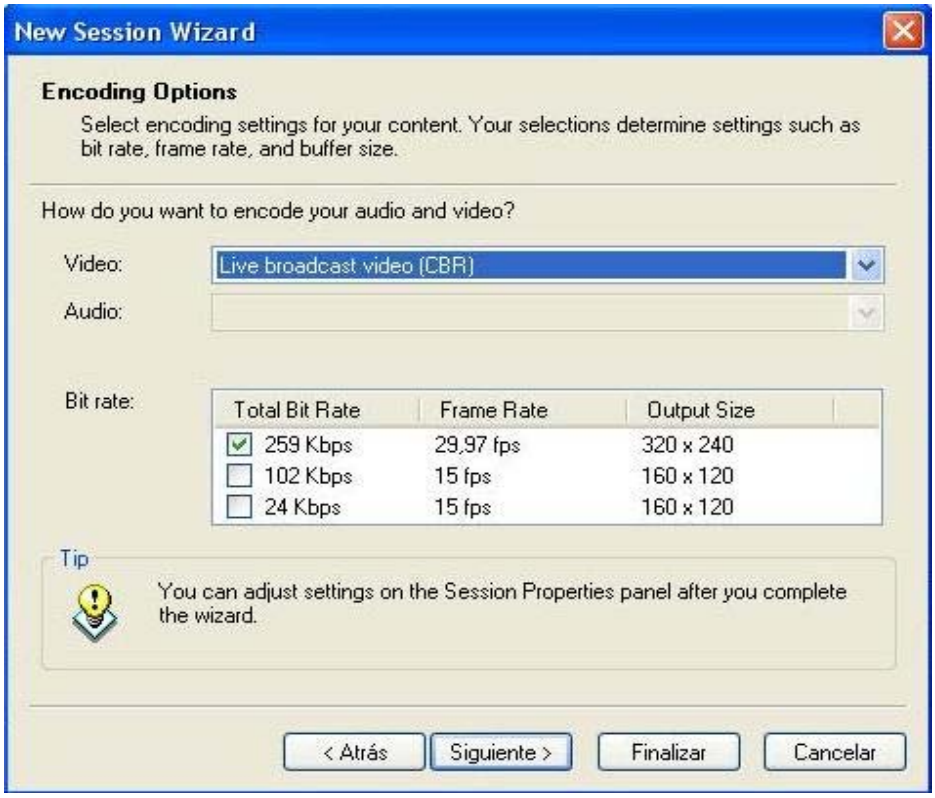


Figura 36 – Selección de opciones de codificación

vii) En el resto de pasos del asistente pulsamos "Siguiente" y finalmente pulsamos "Finalizar".

Si se han realizado correctamente todos los pasos, aparecerá en pantalla la ventana de WME mostrando la salida de vídeo (ver figura 37). En la pestaña "General" aparece, entre otras cosas, el puerto al cual se ha redirigido el flujo de la captura de vídeo.

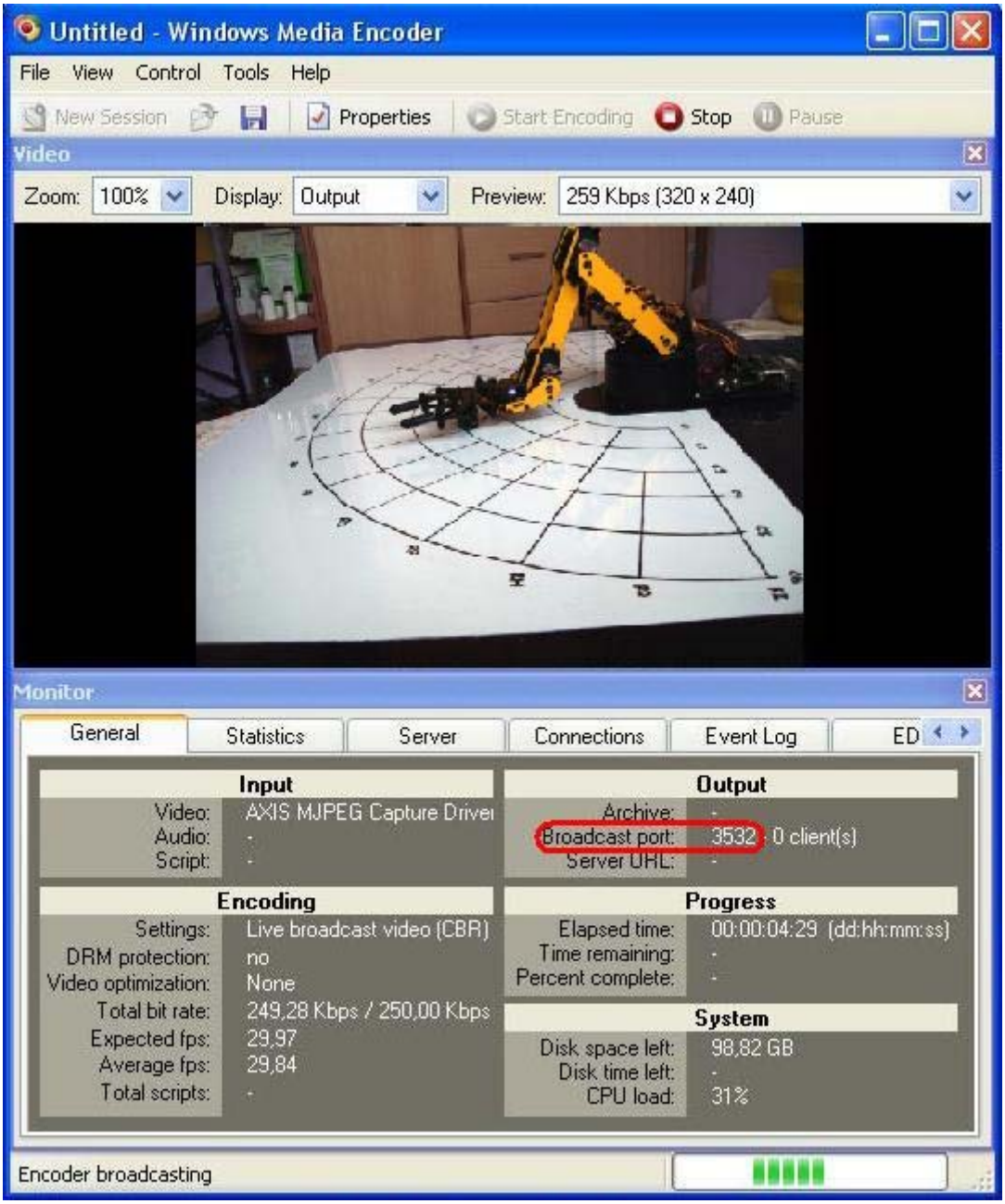


Figura 37 – *Streaming* de vídeo con WME

Cada ventana de WME sólo permite realizar el *streaming* de un flujo de vídeo. Por lo tanto, si queremos que los usuarios del laboratorio remoto visualicen las dos netcams, debemos realizar dos veces el proceso indicado en los pasos descritos, escogiendo un puerto distinto para cada cámara.

6.6. BASES DE DATOS

Como se dijo en el capítulo 5, el sistema a realizar cuenta con dos bases de datos, cuyas características y detalles de implementación serán descritos a continuación.

6.6.1. Descripción del módulo

Se dispone de dos bases de datos con funcionalidades distintas. Una de ellas viene integrada en el entorno de desarrollo web que se ha utilizado para la creación de las páginas web del sistema y posee una serie de tablas predefinidas por el fabricante. El software de desarrollo utilizado posee una serie de elementos que automatizan en gran parte las tareas de implementación de controles para el login y creación de nuevos usuarios, accediendo a dicha base de datos, denominada **"ASPNETDB"**.

Se ha creado una nueva base de datos a la cual se le ha puesto el nombre **"usuarios"**. En esta, se han creado las siguientes 3 tablas:

- **"equivalencias":** Contiene las coordenadas de todos los puntos característicos posibles del espacio de trabajo del manipulador en el lenguaje de etiquetas (semicircunferencia,radio,plano,altura y tiempo) referente al espacio cartesiano a introducir por los usuarios, y sus equivalentes en el lenguaje referente al espacio articular del servocontrolador SSC-32. Esta tabla no se modificará en ningún momento de la ejecución del sistema.

- **"nodos_trayectoria":** Guarda las coordenadas de los nodos de la trayectoria a seguir por el robot que van introduciendo los usuarios (espacio cartesiano).

- **"tanda":** Guarda los nombres de los usuarios cuyas trayectorias se encuentran a la espera de ser ejecutadas por el robot.

6.6.2. Entradas/Salidas

- **Entradas:** Inserción de datos procedentes de la interfaz de usuario o de la capa de negocio del sistema.

- **Salidas:** Datos devueltos hacia la interfaz de usuario o hacia la capa de negocio.

6.6.3. Tecnologías utilizadas

Las dos bases de datos están basadas en la tecnología **Microsoft SQL Server 2008**.

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL (*Transact-SQL*) y ANSI SQL. Constituye la alternativa de *Windows* a gestores de bases de datos como *Oracle* y *MySQL*, entre otros. El formato de los archivos de bases de datos en *SQL Server 2008* es ".mdf", [14].

Se ha decidido usar dicha tecnología en lugar de otras como las nombradas en el párrafo anterior porque viene integrada en el entorno de desarrollo web utilizado y se complementa perfectamente con la tecnología de Microsoft *ASP.NET*, en la cual está basada dicho entorno.

6.6.4. Implementación

Debido a su complejidad y a la cantidad de tablas que contiene la base de datos "ASPNETDB" de las cuales sólo nos interesan las relacionadas con el login de usuarios y la creación de nuevas cuentas de usuario, con el añadido de que gracias a los controles que proporciona la tecnología de desarrollo web utilizada estas tareas son transparentes de cara al programador, abstrayéndose así de la implementación de dicha base de datos para el fin que se ha utilizado, nos limitaremos a describir la implementación de la base de datos "usuarios", creada específicamente para el proyecto actual.

Las tres tablas que componen la base de datos "usuarios" son independientes entre sí, no están unidas por ningún tipo de relación o dependencia. Por lo tanto el **diagrama entidad-relación** de "usuarios" sería el siguiente (ver figura 38):

Figura 38 – diagrama E-R de la base de datos "usuarios".

Tampoco se han nombrado claves primarias en ninguna de las tres tablas, ya que la programación de la capa de presentación y el tipo de datos de algunos de los campos (el campo "orden" en las tablas "tanda" y "nodos_trayectoria") impiden que se produzcan duplicidades en los registros (filas) de cada tabla. En la tabla "equivalencias" tampoco debe preocuparnos esto ya que como se dijo anteriormente, se han introducido los datos en el momento de la programación y no se verán modificados en tiempo de ejecución.

Los **tipos de datos** de los campos de cada tabla son los siguientes:

tanda:

- **username:** *nvarchar*. Nombre del usuario cuya trayectoria se encuentra en la cola de espera para ser ejecutada por el robot.

- **orden:** *int.* Indica el orden en que han sido añadidos los usuarios a la cola de espera. Se le ha establecido la propiedad de incrementarse en 1 cada vez que se añade un nuevo registro a la tabla.

nodos_trayectoria:

- **circunferencia,radio,plano,altura,zona.pinza:** *nvarchar.* Coordenadas referentes a la posición del efector final del manipulador en su espacio de trabajo (espacio cartesiano).

- **tiempo:** *nvarchar.* Indica el tiempo que se desea que tarde el robot en trasladarse de un nodo de la trayectoria al siguiente.

- **username:** *nvarchar.* Nombre del usuario al que pertenecen los nodos de trayectoria introducidos.

- **orden:** *int.* Indica el orden en que han sido añadidos los nodos a la lista de nodos que componen la trayectoria. Se le ha establecido la propiedad de incrementarse en 1 cada vez que se añade un nuevo registro a la tabla.

equivalencias:

- **botones:** *nvarchar.* Indica en una sola cadena de caracteres las coordenadas referentes a la posición del efector final del manipulador en su espacio de trabajo (espacio cartesiano).

  Ejemplo: C3REP1LZ1 indicaría circunferencia 3, radio E, plano 1, altura L y zona 1.

- **comando_sc32:** *nvarchar.* Es el comando en formato procesable por el servocontrolador SSC-32 equivalente a las coordenadas indicadas en el campo "botones".

  Ejemplo: El equivalente a C3REP1LZ1 del campo "botones" sería #0P1568#1P1050#2P1130#3P500#5P1415.

6.6.5. Problemas surgidos y soluciones

Durante la primera implementación de la base de datos "usuarios", no se crearon los campos "orden" en las tablas "tanda" y "nodos_trayectoria". Pero al poner en funcionamiento el sistema, se observó que cuando había más de un usuario en la tanda de espera, estos no aparecían en dicha tanda en orden de aparición. En ocasiones aparecían desordenados y un fallo como este afectaría al sistema en el sentido de que las trayectorias de los usuarios no se ejecutarían en el orden perteneciente al momento en que cada usuario se añadió a la cola.

También se observó que cuando un usuario añadía varios nodos a la lista de nodos para indicar una trayectoria, a menudo estos no se añadían en el orden correcto. En este caso el error sería mucho más grave ya que la trayectoria realizada por el robot no tendría nada que ver con la que el usuario pretendiese.

Para solucionar estos problemas, se creó el campo "orden", de tipo entero autonumérico, que se inicializa con valor "1" y se incrementa una unidad en cada nuevo registro añadido a las tablas. A continuación se añadió el código SQL necesario en la programación de la capa de presentación para que al inicializarse las tablas para ser mostradas en los controles correspondientes de la interfaz gráfica, lo hiciesen siempre ordenadas por el campo "orden".

6.7. PÁGINAS WEB

Como se vio en el capítulo 5, el sistema dispone de dos páginas web para la interacción con el usuario (una para el registro y login de usuarios, y otra para la interacción con el robot por parte de los usuarios logueados). A continuación se estudiarán más a fondo su funcionamiento e implementación, así como la tecnología empleada en el desarrollo de las webs. También se comentarán los problemas acontecidos durante el desarrollo y el modo en que han sido solventados.

6.7.1. Descripción del módulo

El módulo se divide a su vez en dos submódulos:

1- **Interfaz gráfica.**

2- **Código para el funcionamiento del robot y la interacción con el usuario (capa de negocio).**

El submódulo que conforma la interfaz gráfica está formado por lo que el usuario ve en la pantalla cuando accede a la dirección de la página del laboratorio remoto y por el código HTML que implementa cada elemento gráfico de la interfaz. En el capítulo 5 se ha mostrado su aspecto visual (figuras 9A y 9B).

El segundo submódulo está formado por dos tipos de archivos: uno para la configuración de las páginas que conforman el sitio web, llamado "web.config", y el otro tipo son archivos donde se implementan en un lenguaje orientado a objetos (C#) las funciones que hacen que todo el sistema funcione (tratamiento de los datos ingresados en la interfaz gráfica y evío de órdenes de movimiento al robot).

El siguiente esquema muestra el mapa del sitio web completo incluyendo los dos submódulos comentados (ver figura 39):

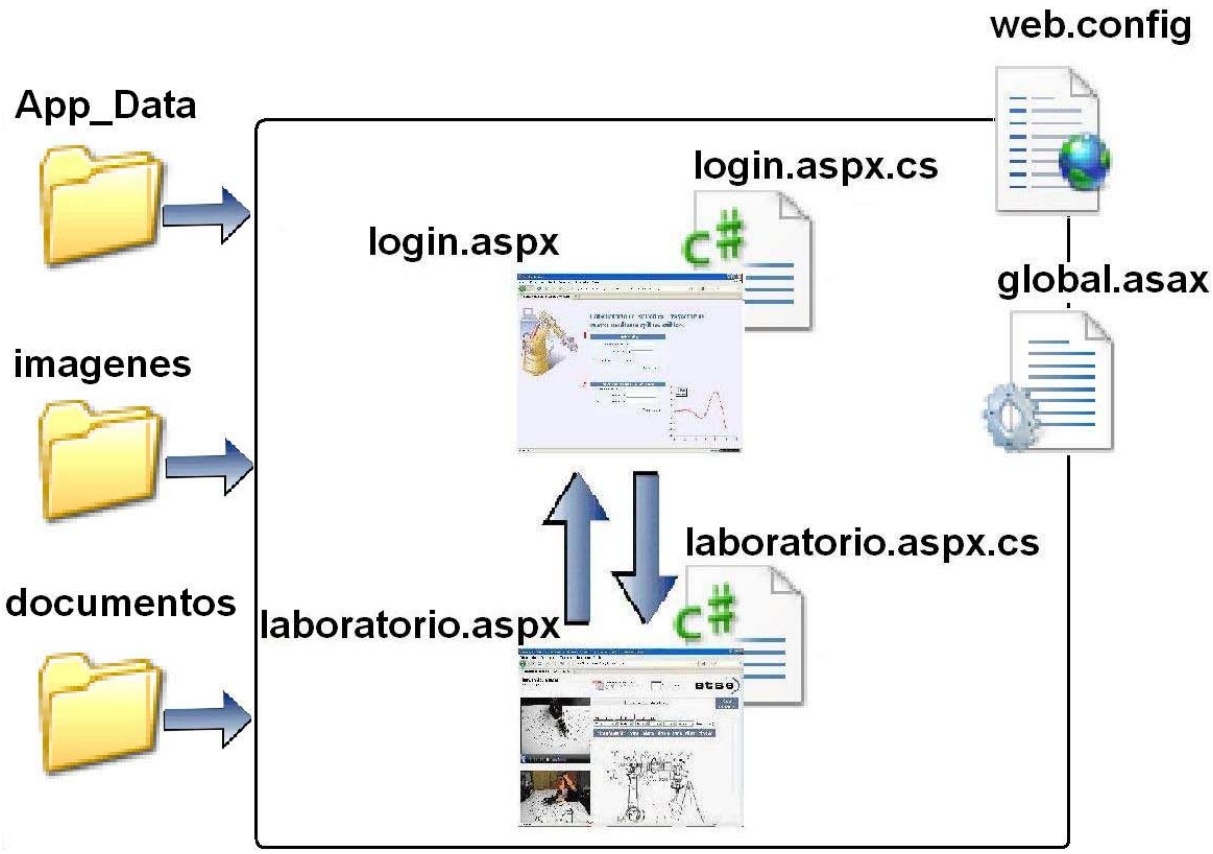


Figura 39 – mapa web del sistema.

Los archivos con extensión **".aspx"** contienen el código HTML de la capa de presentación, y los que tienen extensión **"aspx.cs"** contienen el código C# que implementa funciones pertenecientes a la capa de negocio tales como la gestión de colas de nodos de las trayectorias y de usuarios en espera, o calcular los splines cúbicos de las trayectorias y enviar órdenes de movimiento al robot.

El archivo **"Global.asax"** contiene código (también en C#) referente a las variables de aplicación. Este archivo es opcional, y en nuestro caso ha sido necesario porque se ha creado una variable global con el nombre "OCUPADO" cuya utilidad es indicar si el robot se encuentra ocupado realizando la trayectoria de un usuario o si de lo contrario está libre y listo para ejecutar la siguiente trayectoria. En este archivo se inicializa dicha variable cuando se inicia el sistema (cuando se pone en marcha el servidor web).

En el archivo con nombre **"web.config"** se definen, en lenguaje XML, aspectos de la configuración de las páginas web tales como las cadenas de conexión de las bases de datos utilizadas en el proyecto o el método de autenticación, entre otros.

En la carpeta **"App_Data"** se encuentran las bases de datos del sistema.

La carpeta **"imágenes"** contiene los archivos en formato JPG correspondientes a las imágenes que se muestran en la capa de presentación.

Por último, la carpeta **"documentos"** guarda el manual de usuario del laboratorio en formato PDF y el ejecutable correspondiente a un programa llamado "verificador.exe" cuya función es comprobar que los archivos de texto que el usuario desea enviar con la trayectoria deseada, sean sintácticamente correctos.

El diagrama de casos de uso que describe el funcionamiento del sitio web sería el siguiente (ver figura 40):

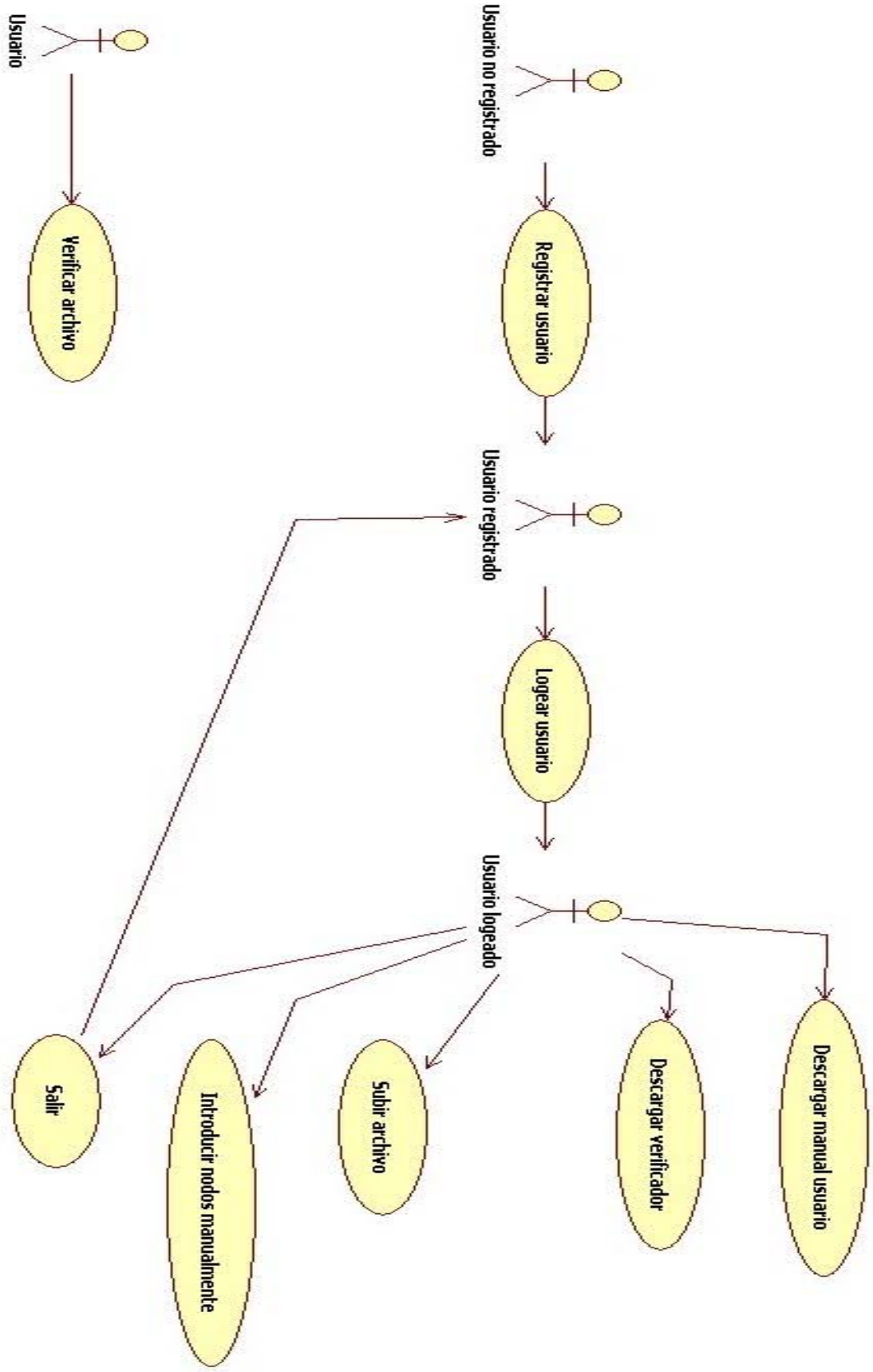


Figura 40 – Diagrama de casos de uso del sitio web

Nombre:	Registrar usuario
Autor:	Marcos Mohedano
Fecha:	21/05/2010

Descripción:

Permite obtener una nueva cuenta de usuario en el sitio web del laboratorio remoto de robótica.

Precondiciones:

Tener habilitada y operativa la conexión a Internet y tener acceso a la red de la UAB.

Flujo Normal:

1. El actor introduce la URL en el navegador.
2. El navegador solicita la página al servidor
3. Se muestra por pantalla un menú que permite registrar un nuevo usuario o loguearse si ya se encuentra registrado.
4. El actor introduce el nombre de usuario y (dos veces) la contraseña deseados en el menú “Regístrese para obtener una nueva cuenta” y pulsa el botón “Crear usuario”.
5. El actor pasa a ser un usuario registrado y accede a la web del laboratorio (“laboratorio.aspx”) con su nueva cuenta.

Flujo Alternativo:

Si ya existe el nombre de usuario o falta algún campo por rellenar, se muestra un mensaje de error.

Postcondiciones:

El actor se convierte en usuario registrado y obtiene acceso a la página del laboratorio con su nueva cuenta.

Nombre:	Logear usuario
Autor:	Marcos Mohedano
Fecha:	21/05/2010

Descripción:

Permite el acceso a la web “laboratorio.aspx” a usuarios registrados.

Precondiciones:

Poseer una cuenta de usuario del laboratorio remoto.

Flujo Normal: 1. El actor introduce la URL en el navegador. 2. El navegador solicita la página al servidor 3. Se muestra por pantalla un menú que permite registrar un nuevo usuario o loguearse si ya se encuentra registrado. 4. El actor introduce el nombre de usuario y contraseña en el menú “Iniciar sesión” y pulsa el botón “Inicio de sesión”. 5. El actor accede a la web del laboratorio (“laboratorio.aspx”).
Flujo Alternativo: Si el nombre de usuario o la contraseña no son correctos, o falta algún campo por rellenar, se muestra un mensaje de error.
Postcondiciones: El actor obtiene acceso a la página del laboratorio.

Nombre:	Descargar manual usuario
Autor:	Marcos Mohedano
Fecha:	21/05/2010
Descripción: Permite descargar un manual de usuario del laboratorio en formato PDF.	
Precondiciones: Encontrarse logueado en la página del laboratorio (“laboratorio.aspx”).	
Flujo Normal: 1. El actor hace click en la imagen con la inscripción “Manual Usuario Alumno.pdf”. 2. Aparece un cuadro de diálogo preguntando si se desea descargar el archivo. 3. El actor pulsa “Aceptar”. 4. Comienza la descarga del manual de usuario en el PC del actor.	
Flujo Alternativo: 1. El actor hace click en la imagen con la inscripción “Manual Usuario Alumno.pdf”. 2. Aparece un cuadro de diálogo preguntando si se desea descargar el archivo. 3. El actor pulsa “Cancelar”.	

4. Se cancela la descarga del archivo.
Postcondiciones: El actor descarga en su PC el manual de usuario.

Nombre:	Descargar verificador
Autor:	Marcos Mohedano
Fecha:	21/05/2010
Descripción: Permite descargar un programa para verificar la sintaxis de los archivos de texto con las trayectorias a enviar al servidor.	
Precondiciones: Encontrarse logueado en la página del laboratorio (“laboratorio.aspx”).	
Flujo Normal: 1. El actor hace click en la imagen con la inscripción “Verificador.exe”. 2. Aparece un cuadro de diálogo preguntando si se desea descargar el archivo. 3. El actor pulsa “Aceptar”. 4. Comienza la descarga del verificador en el PC del actor.	
Flujo Alternativo: 1. El actor hace click en la imagen con la inscripción “Verificador.exe”. 2. Aparece un cuadro de diálogo preguntando si se desea descargar el archivo. 3. El actor pulsa “Cancelar”. 4. Se cancela la descarga del archivo.	
Postcondiciones: El actor descarga en su PC el programa verificador de archivos de texto con las trayectorias a enviar al servidor.	

Nombre:	Subir archivo
Autor:	Marcos Mohedano
Fecha:	21/05/2010
Descripción: Permite enviar un archivo de texto con la trayectoria deseada al servidor.	

Precondiciones: Encontrarse logueado en la página del laboratorio.
Flujo Normal: 1. El actor hace click en el botón “Examinar…”. 2. Aparece una ventana del explorador de archivos del PC para buscar el archivo de texto a subir. 3. El actor selecciona el archivo y pulsa “Aceptar”. 4. El actor pulsa el botón “Subir archivo”. 5. Se envía el archivo de texto al servidor y el actor pasa a la cola de usuarios en espera para la ejecución e trayectorias por parte del robot. Además, se muestran por pantalla los nodos de la trayectoria que contenía el archivo de texto.
Flujo Alternativo: 1. El actor hace click en el botón “Examinar…”. 2. Aparece una ventana del explorador de archivos del PC para buscar el archivo de texto a subir. 3. El actor selecciona el archivo y pulsa “Aceptar”. 4. El actor pulsa el botón “Subir archivo”. 5. Si el archivo contiene nodos que no existen entre los puntos característicos del espacio de trabajo del robot o la sintaxis es incorrecta, se muestra un mensaje de error.
Postcondiciones: El actor ha subido el archivo de trayectorias al servidor y se encuentra en espera para que el robot ejecute su trayectoria.

Nombre:	Introducir nodos manualmente
Autor:	Marcos Mohedano
Fecha:	21/05/2010
Descripción: Permite introducir los nodos de la trayectoria deseada a través de las listas desplegables de la interfaz web.	
Precondiciones: Encontrarse logueado en la página del laboratorio.	

Flujo Normal: 1. El actor elige los elementos de las listas desplegables correspondientes a las coordenadas y al tiempo en milisegundos de uno de los nodos que formarán parte de la trayectoria a enviar. 2. El actor pulsa el botón “Nuevo nodo”. 3. Se muestra por pantalla el nodo introducido. 4. El actor repite los pasos 1 y 2 hasta que se hayan introducido como mínimo tres nodos. 5. El actor pulsa el botón “Enviar”. 6. Se guardan los nodos del actor en la base de datos y este pasa a la cola de usuarios en espera para la ejecución e trayectorias por parte del robot.
Flujo Alternativo: Si cuando el actor pulsa el botón “Enviar” (paso 5 del flujo normal), este ya se encuentra en la cola de espera porque aún no se ha ejecutado una trayectoria enviada con anterioridad o hay menos de 3 nodos en la cola de nodos, se muestra el mensaje de error correspondiente.
Postcondiciones: El actor se encuentra en espera para que el robot ejecute su trayectoria.

Nombre:	Salir
Autor:	Marcos Mohedano
Fecha:	21/05/2010
Descripción: Permite cerrar la sesión del usuario logueado.	
Precondiciones: Encontrarse logueado en la página del laboratorio.	
Flujo Normal: 1. El actor pulsa el enlace “Cerrar sesión”. 2. Se cierra la sesión de usuario del actor y vuelve a la página inicial.	
Flujo Alternativo:	
Postcondiciones:	

El actor ha cerrado su sesión de usuario y se encuentra en la página inicial del sitio web.

Nombre:	Verificar archivo
Autor:	Marcos Mohedano
Fecha:	21/05/2010
Descripción: Permite verificar la sintaxis de los archivos de texto que contienen las trayectorias a enviar al servidor.	
Precondiciones: Poseer el ejecutable “Verificador.exe” y tener el archivo de texto a analizar en la misma carpeta que el verificador.	
Flujo Normal: 1. El actor ejecuta la aplicación verificador.exe. 2. La aplicación solicita al actor el nombre del archivo. 3. La aplicación verifica la existencia del archivo y lo abre. 4. La aplicación verifica la sintaxis del código del archivo 5. Se muestra un mensaje por la pantalla indicando si es correcta o no, y en caso de no ser correcta indica el tipo de error encontrado y la línea dentro del programa.	
Flujo Alternativo: 1. El actor ejecuta la aplicación verificador.exe. 2. La aplicación solicita al actor el nombre del archivo. 3. Si el archivo no se encuentra dentro de la carpeta de la aplicación muestra un mensaje de error por pantalla.	
Postcondiciones: Se ha verificado el archivo e informado al actor del resultado.	

6.7.2. Entradas/Salidas

- **Entradas:** Datos ingresados por los usuarios del laboratorio (nombres de usuario/contraseñas y nodos de las trayectorias enviados a través de un archivo de texto o ingresados manualmente en la interfaz gráfica) .

- **Salidas:** Comandos de movimiento del robot enviados al puerto serie del PC servidor y datos mostrados a los usuarios a través de la interfaz.

6.7.3. Tecnologías utilizadas

Para el desarrollo de las webs que conforman este módulo se ha utilizado la tecnología de *Microsoft* **ASP.NET.**

ASP.NET es un *framework* para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Apareció en enero de 2002 con la versión 1.0 del .NET Framework, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET Permite a los programadores escribir código usando cualquier lenguaje admitido por el .NET Framework (plataforma de desarrollo software creada por Microsoft), como C# y VisualBasic.NET.

Esta tecnología posee muchas ventajas frente a otras posibles elecciones como por ejemplo PHP, las cuales han sido ya descritas en el capítulo 5. Una de ellas es que permite la separación de la parte del código perteneciente a la capa de presentación con respecto al de la capa de negocio en archivos distintos. Los archivos de la capa de presentación tienen la extensión “aspx”, y los de la capa de negocio tendrán la extensión del lenguaje de programación utilizado (en nuestro caso el lenguaje es C#, así que tendrán extensión “.cs”).

Otra de las características más importantes que posee esta arquitectura de desarrollo web, es el concepto de **controles de servidor web.** Los controles de servidor web ASP.NET son objetos de páginas web ASP.NET que se ejecutan cuando se solicita la página y que representan marcado en un explorador. Muchos controles de servidor web son similares a elementos HTML conocidos, como botones y cuadros de texto. Otros controles abarcan

comportamiento complejo, por ejemplo controles de calendario y controles que administran las conexiones de datos.

Existen varias herramientas gratuitas de programación para aplicaciones web en ASP.NET, pero se ha escogido la proporcionada por Microsoft, **Visual Web Develper Express Edition** por ser la más completa e intuitiva y por disponer de una mayor cantidad de manuales en la red.

6.7.4. Implementación

En este apartado se mostrará el código de los dos submódulos (interfaz y capa de negocio) y se comentará la funcionalidad de los controles, clases más importantes, funciones y procedimientos utilizados en dicho código.

6.7.4.1. Implementación de la interfaz gráfica

El código HTML que implementa la interfaz gráfica de la página principal de nuestro sitio web ("**Login.aspx**"), es el siguiente:

Los elementos más importantes del código HTML que implementa la interfaz gráfica (ver Anexo 1) de la página principal de nuestro sitio web ("**Login.aspx**") son los siguientes controles de servidor:

- **Login:** Se define mediante las etiquetas *<asp:Login/> </asp:Login>*. Este control muestra una interfaz para la autenticación de usuario. Contiene cuadros de texto para el nombre de usuario y la contraseña, y una casilla de verificación que permite a los usuarios indicar si quieren que el servidor almacene su identidad y que los autentique automáticamente la próxima vez que visiten el sitio.

  Entre el código que se ha creado para este control, destaca el siguiente fragmento:

```
onauthenticate="Login1_Authenticate"
```

Esta propiedad indica que cuando se pulse el botón "Inicio de sesión", se ejecutará el procedimiento implementado en la capa de negocio con nombre "`Login1_Authenticate`".

- **CreateUserWizard:** Definido mediante las etiquetas *<asp:CreateUserWizard> </asp:CreateUserWizard>*, proporciona la interfaz de usuario para el objeto *MembershipProvider* (clase que proporciona un servicio integrado para validar y almacenar las credenciales del usuario) que se comunica con el almacén de datos del usuario del sitio Web para crear nuevas cuentas de usuario.

  Se han definido dos propiedades importantes en este control. La primera es:

  ContinueDestinationPageUrl.

  Indica la página a la cual se accederá una vez creada la cuenta de usuario. Según nuestro código, al crear la cuenta el nuevo usuario accederá a la página del laboratorio como usuario logueado.

  La siguiente propiedad, que también se ha definido en el control Login, es la siguiente:

  MembershipProvider.

  Esta propiedad especifica al control qué proveedor utilizará para la validación y almacenamiento de las credenciales de usuario. Dicho proveedor se ha definido en el archivo "web.config", perteneciente a la capa de negocio.

Estos dos controles junto con la clase *MembershipProvider* facilitan enormemente las tareas de administración de cuentas de usuario ya que el programador no necesitará implementar procedimientos para dichas tareas, que con estos controles y clases integrados en ASP.NET se realizan de manera automática.

Los controles ASP.NET y fragmentos de código más relevantes en el código HTML que implementa la interfaz de la página "**laboratorio.aspx**" (ver Anexo 2) son los que se muestran a continuación:

- **Control SqlDataSource:** Este control no visible en la interfaz de usuario, representa los datos de una base de datos SQL Server (en nuestro caso las base de datos "usuarios.mdf"). Los controles de servidor enlazados a datos, como por ejemplo los *GridView* que se han utilizado también en el código de la página "laboratorio.aspx", utilizan el control SqlDataSource para obtener dichos datos y mostrarlos en la interfaz.

  Las propiedades más importantes de este control son:

  - **ConnectionString:** Aquí se especifica la cadena de conexión de la base de datos a utilizar. En nuestro caso será la cadena de conexión de la base de datos "usuarios", cuyo nombre es "UsuariosConectionString".

  - **SelectCommand:** Especifica el comando de selección SQL correspondiente a los datos que se desea que el SqlDataSource represente.

  En la página "laboratorio.aspx" se han creado 4 controles del tipo SqlDataSource: uno para representar los datos de la tabla "equivalencias", otro para la tabla "tanda", y los dos restantes para la tabla "nodos_trayectoria". Se han creado dos SqlDataSource para la tabla "nodos_trayectoria" porque uno ("SqlDataSource4") servirá para representar los nodos de la trayectoria del primer usuario de la tanda de espera, y el otro ("SqlDataSource2") para los nodos del usuario logueado en el laboratorio (para cada usuario se mostrarán sólo sus propios nodos en la interfaz web de su navegador).

- **Control GridView:** Muestra en forma de tabla en la interfaz de usuario los datos que alberga un control SqlDataSource.

Se han creado 3 instancias del control Gridview. Una para mostrar la tanda de usuarios en espera ("GridView2"), otra para la cola de nodos de usuarios logueados en la que para cada usuario se mostrarán sólamente sus nodos en la interfaz de su navegador ("GridViewnodos"), y otra, que no se mostrará visible en la interfaz, que se usará a modo de variable auxiliar para guardar los nodos del primer usuario de la tanda ("GridViewPrimerUsuario").

- **Control Timer:** Provoca devoluciones de datos desde el explorador cada ciertos intervalos de tiempo (*ticks*).

  El tiempo en milisegundos se establece en la propiedad "Interval" de la declaración del control. En este caso se ha introducido un intervalo de 5000 ms con la intención de evaluar en cada *tick* del Timer el valor de la variable global "OCUPADO", nombrada en el apartado 6.7.1.

- **Control UpdatePanel:** Este control es una extensión de la tecnología AJAX (*Asynchronous JavaScript And XML*) que permite realizar actualizaciones con devolución de datos sólo en los controles que se hayan declarado en el interior de la declaración de la instancia de UpdatePanel.

  Se ha declarado el Timer dentro de la única instancia de UpdatePanel que posee la web del laboratorio con la finalidad de que cada *tick* del timer dispare el evento de actualización de los controles que se encuentran dentro del UpdatePanel. Estos controles son el Gridview que muestra la cola de usuarios ("Gridview2") y el que muestra la lista de nodos de usuario logueado ("GridViewnodos").

  También se han declarado en su interior los botones que sirven para añadir o borrar elementos de la lista de nodos y de la cola de usuarios de manera que al pulsar dichos botones se desencadene el evento de actualización de los elementos gráficos del UpdatePanel.

  Gracias a la inclusión de este control en nuestra página, se mejora la velocidad de la aplicación web y la interactividad con el usuario debido a que

ya no se actualizará toda la página innecesariamente en el navegador del cliente.

- **Código para incrustar *streaming* de vídeo en la interfaz web:**

Se desea mostrar en directo a través de la interfaz del laboratorio remoto el flujo de vídeo capturado por las dos cámaras de red del sistema. Como se dijo en el apartado 6.5, dicho flujo se recodifica a formato WMA a través del software Windows Media Encoder y se redirige a un puerto específico del servidor (un puerto para cada cámara).

La parte del código HTML del archivo "laboratorio.aspx" cuya finalidad es incrustar dicho flujo de vídeo en la interfaz web es el que se halla encerrado entre las etiquetas <*object*> <*/object*> (ver Anexo 2).

La etiqueta ***object*** hará que se abra el control ActiveX del reproductor *Windows Media Player* en la página web cuando los clientes que lo tengan instalado entren en la página. Dentro de estas eqtiquetas encontramos atributos importantes como:

- **classid:** especifica la versión del reproductor.

- **codebase:** Si el usuario no tiene instalada la versión del reproductor *Windows Media Player* que se especifica en este atributo, se descargará e instalará automáticamente desde la URL indicada en el atributo.

Por último, hay que indicar de donde se obtiene el flujo de vídeo. Esto se hace mediante el siguiente código:

<param name="filename" value=[IP]:[puerto] />

Dónde *[IP] es la dirección IP* del servidor web del sistema, y *[puerto]* el puerto donde se ha redirigido el *streaming* de vídeo a través del software *Windows Media Encoder*.

La IP y el puerto habrá que indicarlos también en el atributo *src* de las etiquetas <*embed*></*embed*> contenidas en el código de *object*.

6.7.4.2. Implementación del código de la capa de negocio

Como ya se vió en la descripción de este módulo (apartado 6.7.1), la parte de la programación web que implementa la capa de negocio está formada por 4 archivos:

- **Global.asax**
- **web.config**
- **Login.aspx.cs**
- **laboratorio.aspx.cs**

El archivo **Global.asax** (ver Anexo 3) contiene el código escrito en lenguaje C# perteneciente al tratamiento de las variables globales de aplicación y de sesión. Las *variables de aplicación* son aquellas que afectan a la aplicación en el servidor, global a todos los usuarios que inicien sesión en el sistema, mientras que los valores de las *variables de sesión* son distintos para cada ventana del navegador. Por lo tanto, si se modifica una variable de aplicación, el cambio de valor se hará patente para todos los usuarios de la aplicación web, mientras que un cambio en una variable de sesión sólo afecta a la sesión del usuario donde ha sido modificada.

En este archivo se encuentran declarados por defecto 5 procedimientos sin código en su interior, que se ejecutan en el inicio de la aplicación, en el final, en caso de error de aplicación, al inicio de sesión o al finalizar una sesión. En nuestro caso nos interesa crear una variable que sea global a todos los usuarios, cuya función será indicar si el robot se encuentra ocupado ejecutando la trayectoria de algún usuario o si por el contrario está libre en estado de reposo. Por lo tanto debemos crear esta variable, a la cual llamaremos “OCUPADO”, en el procedimiento de inicio de aplicación de la siguiente manera:

```
void Application_Start(object sender, EventArgs e)
{
```

```
    Application.Add("OCUPADO","NO");
}
```

El **web.config** (ver Anexo 4) contiene el código XML para la configuración de las páginas web que se encuentren en el mismo directorio que dicho archivo. En nuestro caso sólo necesitamos un archivo web.config para todo el sistema.

Los fragmentos de ódigo más relevantes en este archivo son los que hacen referencia a la base de datos, a la autenticación y a la autorización.

El siguiente código añade una cadena de conexión que enlazará la capa de negocio del sistema con la base de datos creada en la capa de datos (“usuarios.mdf”):

```
<connectionStrings>

<add name="usuariosConnectionString" connectionString="Data
Source=.\SQLEXPRESS;AttachDbFilename=&quot;C:\AppServ\www\App_Data\usuar
ios.mdf&quot;;Integrated Security=True;Connect Timeout=30;User Instance=True"
providerName="System.Data.SqlClient"/>

</connectionStrings>
```

Se desea que no se autorice el acceso a la web a los usuarios anónimos (es decir, los no autenticados), debemos escribir el siguiente código:

```
<authorization>

    <deny users="?"/>

</authorization>
```

Y el métido de autenticación se especifica de la siguiente manera:

```
    <authentication mode="Forms">
```

```
        <forms loginUrl="Login.aspx" protection="All"/>

    </authentication>
```

Con estas líneas se ha especificado que el método de autenticación es mediante una página de login, escribiendo el nombre y contraseña de usuario en dicha página. Los dos últimos fragmentos de código mostrados sirven para evitar que un usuario no autenticado pueda entrar a la página "laboratorio.aspx" sin antes haber iniciado sesión en la página de login. Cuando dicho usuario escriba en su navegador la URL de "laboratorio.aspx" el servidor le redirigirá automáticamente a la página "Login.aspx" para que inicie sesión.

Por último, se ha añadido el siguiente código para automatizar las tareas de Login y creación de usuarios mediante la clase *SqlMembershipProvider*, que utiliza la base de datos *ASPNET.DB* integrada en el entorno de desarrollo de ASP.NET para la administración de cuentas de usuario:

```
<membership defaultProvider="proveedorSql">

<providers>

<add name="proveedorSql" type="System.Web.Security.SqlMembershipProvider,
System.Web, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
connectionStringName="LocalSqlServer" enablePasswordRetrieval="false"
enablePasswordReset="true" requiresQuestionAndAnswer="false"
applicationName="/" requiresUniqueEmail="false" passwordFormat="Hashed"
maxInvalidPasswordAttempts="5" minRequiredPasswordLength="7"
minRequiredNonalphanumericCharacters="1" passwordAttemptWindow="10"
passwordStrengthRegularExpression=""/>

</providers>

</membership>
```

Se ha indicado el nombre del proveedor de suscripciones de SqlServer y se han especificado opciones como el número de intentos para escribir la contraseña correctamente, el método de cifrado de la contraseña, etc.

El archivo **Login.aspx.cs** (ver Anexo 5) contiene el código en lenguaje C# que corresponde a las funciones y procedimientos que se ejecutan cuando el usuario interacciona con la interficie de la página de login.

Nos interesa redirigir al usuario autenticado en el control de la página de login (al que hemos llamado "Login1") hacia la página del laboratorio. Para ello escribiremos el código que se muestra a continuación en el evento *Login1_Authenticate* del control "Login1":

```
protected void Login1_Authenticate(object sender,     AuthenticateEventArgs e)
    {
       if(Membership.ValidateUser(Login1.UserName,Login1.Password)){
          FormsAuthentication.RedirectFromLoginPage(Login1.UserName, false);
       }
    }
```

El método *ValidateUser* de la clase *Membership* devuelve un valor booleano que indica si el nombre de usuario y la contraseña introducidos en el control *Login* son correctas.

El método *RedirectFromLoginPage* de la clase *FormsAuthentication* redirige al usuario autenticado a la página inicialmente solicitada o a la que se ha establecido como página predeterminada, que en nuestro caso será "laboratorio.aspx".

En el archivo **laboratorio.aspx.cs** (ver Anexo 6) están las funciones y procedimientos que permiten la interacción de los usuarios del laboratorio remoto con el robot a través de la interfaz web.

Los usuarios introducen los nodos manualmente a través de las listas desplegables que se muestran por pantalla o envían al servidor un fichero de texto con todos los nodos de la trayectoria y a continuación se añaden a la cola de espera. Mientras, se dispara el evento Timer1_Tick cada 5 segundos y comprueba si el robot se encuentra libre y en reposo (variable de aplicación OCUPADO="NO"). Si se encuentra libre y hay usuarios en espera, se cambia el valor de dicha variable a "SI" y se ejecutan una serie de procedimientos que calculan la trayectoria mediante el uso de *splines* cúbicos a partir de los nodos pertenecientes al primer usuario de la cola de espera, y se envían uno a uno a través del puerto serie los comandos de movimiento del robot pertenecientes a cada punto de la trayectoria conforme se van calculando. Una vez ejecutada la trayectoria, se envía al robot el comando que le devuelve a su posición de reposo y se cambia nuevamente el valor de la variable para indicar que el robot ya está libre (OCUPADO="NO").

A continuación se describirá el funcionamiento de los procedimientos, eventos y funciones pertenecientes al archivo *laboratorio.aspx.cs*:

- **protected void Page_Load(object sender, EventArgs e)**

  Evento que se dispara cada vez que se carga la página web. Se desea distinguir cuándo se carga la página como resultado de una devolución de datos (evento conocido como *postback,* producido al pulsar un botón de la interfaz, por ejemplo) y cuándo se carga por primera vez. Para ello introduciremos el código:

```
if (!IsPostBack)
{

}
else
{

}
```

  La propiedad *IsPostBack* de la clase *Page* de ASP.NET, adquiere el valor booleano *true* en el caso de que la carga de la página sea debida a una devolución de datos, y *false* cuando se carga la página por primera vez. En este último caso (*! IsPostBack*) se ha añadido una variable de sesión que contenga el nombre del usuario logueado. La forma de crear variables de sesión y añadirles un valor es la siguiente:

```
Session.Add("idusuario", User.Identity.Name.ToString());
```

Donde el primer parámetro ("idusuario") es el nombre de la variable, y el segundo hace referencia al valor que se desea darle. En nuestro caso ese valor será el nombre del usuario propietario de la sesión, y se obtiene mediante la propiedad *Name* de la clase *User.Identity*.

- **protected void Button1_Click(object sender, EventArgs e)**

Este evento se dispara cuando el usuario pulsa el botón "Nuevo nodo" de la interfaz, y el código que se ha añadido en su interior realiza las siguientes acciones:

1. Obtiene el comando de movimiento del robot en el formato del circuito SSC-32 equivalente al nodo que representan los elementos seleccionados por el usuario en las listas desplegables de la interfaz. Para la obtención del comando equivalente se ha creado el procedimiento **obtener_comando().**

2. Comprueba que el comando existe entre las posiciones permitidas para el robot.

3. Si existe dicha posición, añade las coordenadas (en formato del lenguage de etiquetas entendible por el usuario) a la lista de nodos del usuario y las muestra por pantalla. El procedimiento creado para insertar el nodo en la lista de nodos se llama **insertar_nodo().**

- **protected string obtener_comando()**

Función que devuelve un *string* con el comando SSC-32 referente a la posición de todos los servos del robot excepto el de la pinza, equivalente a las coordenadas del efector final indicadas por los elementos seleccionados en las listas desplegables de la interfaz. Para ello, primero guarda en un *string* (se le ha puesto el nombre "nodo") la concatenación de los *strings* que representan los elementos seleccionados, y posteriormente llama a la función **equivalente(string cadena)** para obtener el comando en formato SSC-32 que representa dicho nodo.

- **protected string obtener_comando(int i)**

Devuelve una cadena de texto con el comando SSC-32 equivalente a las coordenadas cartesianas indicadas en la fila i-ésima del GridView oculto (*GridViewPrimerUsuario*) que guarda los nodos del primer usuario de la tanda.

- **protected string equivalente(string cadena)**

Retorna una cadena de texto con el valor del campo *comando_sc32* de la tabla "equivalencias" de la base de datos "usuarios" perteneciente a la fila cuyo campo *botones* se corresponda con el valor del parámetro de entrada *cadena*.

Para realizar transacciones con una base de datos en ASP.NET se necesitan crear sentencias SQL. En el fragmento de código que se muestra a continuación se puede observar cómo se han declarado la conexión con la representación de la base de datos referente a la tabla en la cual se quiere realizar la operación, y el comando ASP.NET que representa la sentencia SQL de la transacción. Esto se logra mediante el uso de las clases *SqlConnection* y *SqlCommand*:

```
SqlConnection conexion = new SqlConnection(SqlDataSource3.ConnectionString);

SqlCommand comando = new SqlCommand("select [comando_sc32] from
equivalencias where botones='"+cadena+"'", conexion);
```

Para ejecutar la sentencia, hay que abrir la conexión con la base de datos, ejecutar el comando creado con anterioridad, y finalmente cerrar la conexión. Para ejecutar comandos SQL que devuelven valores (como en este caso, el resultado de una sentencia SELECT) se utiliza el método *Execute Scalar()* de la clase SqlCommand:

```
conexion.Open();
    string resultado = (string)comando.ExecuteScalar();

conexion.Close();
```

- **protected void insertar_nodo()**

Inserta en la tabla "nodos_trayectoria" de la base de datos "usuarios" las coordenadas referentes a la semicircunferencia, radio, plano, altura y zona, el tiempo en milisegundos en que se desea que el robot llegue a dichas coordenadas, y el nombre de usuario del usuario logueado.

En el siguiente fragmento del código de este procedimiento se muestra cómo se ha declarado la sentencia necesaria para insertar el nodo en la lista mediante el uso de las clases *SqlConnection* y *SqlCommand*:

```
SqlConnection conexion = new SqlConnection(SqlDataSource2.ConnectionString);
SqlCommand comando_insert = new SqlCommand("insert into
nodos_trayectoria(circunferencia,radio,plano,altura,zona,pinza,tiempo,username)
values('" + DLcircunferencia.SelectedValue.ToString() + "','" +
DLradio.SelectedValue.ToString() + "','" + DLplano.SelectedValue.ToString() + "','"
+ DLaltura.SelectedValue.ToString() + "','" + DLzona.SelectedValue.ToString() + "','"
+ DLpinza.SelectedValue.ToString() + "','" +
(Convert.ToInt32(TextBox1.Text)).ToString() + "','" + User.Identity.Name.ToString()
+ "')", conexion);
```

1. Se ha creado una instancia de la clase *SqlConnection* especificando la cadena de conexión de la base de datos con la que se desea trabajar.

2. A continuación se crea un comando SQL. En este caso, se desea insertar en la tabla "nodos_trayectoria" el valor de cada uno de los elementos seleccionados en las listas desplegables de la interfaz.

Para ejecutar la sentencia, hay que abrir la conexión con la base de datos, ejecutar el comando creado con anterioridad, y finalmente cerrar la conexión:

```
conexion.Open();
        comando_insert.ExecuteNonQuery();
conexion.Close();
```

El método *ExecuteNonQuery()* de la clase *SqlCommand* sirve para ejecutar comandos SQL que no devuelven ningún valor, es decir para comandos de inserción o eliminación.

Finalmente, para mostrar por pantalla la lista de nodos actualizada:

```
GridViewnodos.DataBind();
```

El método *DataBind()* del control *GridView* actualiza dicho control para mostrar en la interfaz los cambios realizados sobre la base de datos a la que esté enlazado.

- **protected void Button2_Click(object sender, EventArgs e)**

Es el evento que se dispara cuando un usuario pulsa el botón "Enviar". El código creado inserta el nombre del usuario logueado en la cola de espera, siempre y cuando haya insertado (manualmente o a través de un fichero de texto) almenos dos nodos y no se encuentre ya en espera para la ejecución de una trayectoria enviada con anterioridad.

Para verificar que el usuario no se encuentra ya en espera se comprueba el número de apariciones del nombre de usuario en la tabla *tanda* de la base de datos "usuarios" llamando a la función **num_apariciones()**.

- **protected int num_apariciones()**

Devuelve un entero con el número de veces que aparece el nombre del usuario logueado en la tabla *tanda* de la base de datos "usuarios".

- **protected void insertar_usuario_tanda()**

Inserta el nombre del usuario logueado en la tabla *tanda*.

- **protected void Button3_Click(object sender, EventArgs e)**

En este evento desencadenado al pulsar el botón "Borrar nodos", borra todos los nodos pertenecientes al usuario logueado de la tabla *nodos_trayectoria*, y muestra en su interfaz el control *GridView* de la lista de nodos en blanco

- **protected void borrar_cola_nodos()**

Borra de la tabla *nodos_trayectoria* los nodos pertenecientes al primer usuario de la tanda.

- **protected int num_filas_nodos(string texto)**

Devuelve un entero con el número de filas de la tabla *nodos_trayectoria* cuyo campo *username* se corresponda con el valor de la cadena *texto*.

- **protected int num_filas_nodos_novacias(string texto)**

Devuelve un entero con el número de filas de la tabla *nodos_trayectoria* cuyo campo *username* se corresponda con el valor de la cadena *texto* en las que el resto de campos de dichas filas no se encuentren vacíos. Es decir, devuelve el número de nodos que hay en la lista de nodos de un usuario.

- **protected int num_filas_tanda()**

Retorna un entero que representa la cantidad de filas que posee la tabla *tanda*.

- **protected void botonsubir_Click(object sender, EventArgs e)**

Este evento se desencadena cuando el usuario pulsa el botón "Subir archivo" de la interfaz. Verifica que no hay nodos en la cola de nodos del usuario logueado, que este no se encuentra ya en la tanda de espera, que la sintaxis del archivo de texto es correcta y que este contiene almenos dos nodos. Si cumple las condiciones introduce en la lista de nodos del usuario logueado los nodos que representaban las líneas del fichero, la muestra por pantalla e inserta el nombre de usuario en la tanda de espera.

- **protected bool validar_texto(string path)**

Devuelve un booleano con el valor "true" si el texto del fichero subido al servidor por el usuario logueado cumple la sintaxis adecuada, y "false" en caso contrario. Para ello, comprueba que el nodo que representa cada línea del fichero existe en el campo *botones* de la tabla "equivalencias". Si la sintaxis de alguna de las líneas es incorrecta o si a pesar de estar escrita correctamente no representa un nodo perteneciente a una de las posibles coordenadas cartesianas del espacio de trabajo del robot accesibles por este, se mostrará por pantalla un mensaje de error y no se aceptará el fichero.

- **private int num_lineas_fichero(string path)**

  Retorna en un entero la cantidad de líneas del fichero de texto.

- **private void transcribir_fichero(string path)**

  Inserta en la cola de nodos del usuario logueado los nodos representados por cada línea del fichero de texto enviado al servidor.

- **protected void ImageButton1_Click(object sender, ImageClickEventArgs e)**

  Evento desencadenado cuando el usuario pulsa sobre la imagen de la interfaz que corresponde al manual de usuario. Al pulsar sobre esta, se descarga el manual en formato *pdf* al ordenador del usuario mediante una llamada al procedimiento **descargar_archivo(filename)** donde *filename* es el nombre del archivo**.**

- **protected void ImageButton4_Click(object sender, ImageClickEventArgs e)**

  Se desencadena cuando el usuario pulsa sobre la imagen de la interfaz que corresponde a la aplicación "Verificador.exe". Al pulsar sobre esta, se descarga el el ejecutable al ordenador del usuario mediante una llamada al procedimiento **descargar_archivo(filename)** donde *filename* es el nombre del archivo**.** Este verificador es un programa realizado durante el curso académico 2008-2009 por el ex-alumno de la EUIS Ramón González para su proyecto final de carrera, cuya función es comprobar en el PC del usuario que la sintaxis del fichero de texto a enviar es correcta.

- **descargar_archivo(string filename)**

  Inicia la descarga al PC del usuario, del archivo contenido en la carpeta "documentos/" del servidor cuyo nombre sea el que indique el parámetro *filename*.

- **protected void Timer1_Tick(object sender, EventArgs e)**

  Este evento se dispara cada vez que transcurra el tiempo indicado en milisegundos en la propiedad *tick* del control *timer1* añadido a la página, que en nuestro caso es de 5000ms. Cada vez que se dispare, comprobará el estado de la variable de aplicación "OCUPADO". Si su valor es "NO" y la tanda de usuarios en espera no está vacía, cambia el estado a "SI" y lanza un nuevo hilo de ejecución para el cálculo de los *splines* que interpolarán los nodos del primer usuario de la tanda y el envío al circuito SSC-32 del robot a través del puerto serie de los comandos de movimiento pertinentes. El nuevo hilo se crea y lanza su ejecución de la siguiente manera:

```
Thread hilo = new Thread(new ThreadStart(ejecutar_comandos));

hilo.Start();
```

  En este caso la llamada al método *Start* de la clase *Thread* ejecutará en un proceso a parte el procedimiento **ejecutar_comandos()**.

  El hecho de que se quiera ejecutar este procedimiento en un hilo distinto al de la aplicación principal es debido a que durante el envío de comandos de movimiento al robot se han realizado llamadas al método S*leep* de la clase *thread* para hacer pausas entre comando y comando. Estas pausas se han provocado para evitar pérdidas de datos debido a que se envíen a más velocidad de la que el circuito SSC-32 sea capaz de procesar y almacenar en el *búffer*. Las llamadas a *Thread.Sleep* no se pueden realizar en el mismo hilo de ejecución de la aplicación principal ya que si esto ocurriera, toda la aplicación quedaría paralizada para todos los usuarios del sistema mientras el robot ejecute una trayectoria. Entonces, mientras el robot estuviese ocupado ningún usuario podría interactuar de ninguna manera con la web (enviar fichero, introducir nodos manualmente, etc).

- **protected void ejecutar_comandos()**

El procedimiento *ejecutar_comandos()* no es más que una adaptación a lenguaje C# del algoritmo de cálculo de splines descrito en el capítulo 4 (apartado 4.3). El funcionamiento es el siguiente:

1. Primero se almacenan en dos vectores las posiciones y tiempos que indican los nodos del primer usuario de la tanda, mediante una llamada al procedimiento **rellenar_vectores(vcomandos, vtiempos)**, donde *vcomandos* es un vector inicialmente vacío en el cual se almacenarán las cadenas de texto que representan las posiciones (parte del comando SSC-32 correspondiente a la posición angular de todos los servos) de cada nodo, y *vtiempos* el vector de los tiempos correspondientes a cada posición. Cuando un usuario introduce el tiempo en milisegundos de un nodo, está haciendo referencia al tiempo en que desea que el robot cambie de la posición del nodo anterior a la posición actual, sin embargo el vector *vtiempos* almacenará los tiempos referentes al tiempo transcurrido desde el instante inicial de la trayectoria y no desde el nodo anterior.

2. Se realiza la llamada a **enviar_splines(vcomandos, vtiempos)** para calcular los *splines* cúbicos que interpolan los nodos de la trayectoria y enviar los comandos correspondientes al robot.

3. Se borran de la tabla *nodos_trayectoria* los nodos pertenecientes al primer usuario de la tanda, cuya trayectoria ha sido ejecutada por el robot, se borra su nombre de la tabla *tanda* y se cambia el estado de la variable "OCUPADO" a "NO" para indicar que el robot está listo para ejecutar la trayectoria del siguiente usuario.

- **protected void rellenar_vectores(string[] v_comando, int[] v_tiempo)**

Modifica los vectores a los que hacen referencia los parámetros *v_comando* y *v_tiempo* rellenándolos con las posiciones de los servos del robot en cada nodo y los tiempos en los que se desea que el robot se situe en dichas posiciones,

respectivamente. Los tiempos contenidos en cada posición de v_tiempo corresponderán al tiempo en milisegundos transcurrido desde el instante de inicio de la ejecución de la trayectoria por parte del robot (instante 0 ms) hasta que se encuentre en la posición representada por el contenido de la posición i-ésima del vector *v-comando*.

- **protected void enviar_splines(string[] v_comandos, int[] v_tiempos)**

Implementa el algoritmo de generación de trayectorias mediante *splines* cúbicos descrito en el apartado 4.3 del capítulo 4 y envía a traves del puerto serie del servidor los comandos de movimiento al circuito SSC-32 del robot.

En el capítulo 4 se representa una trayectoria como una sucesión de valores para determinados instantes de tiempo. Estos valores serían en nuestro caso la posición angular de uno de los servos del robot en cada instante del intervalo. Como el robot tiene 6 servos, debemos ejecutar el algoritmo tantas veces como servos existen.

Por lo tanto, en nuestra modificación del algoritmo de cálculo de *splines* cúbicos, se hará lo siguiente:

1. Se crean 6 vectores de posiciones, uno para cada servo, a partir del vector *v_comandos* (cada posición *v_comandos* contiene el fragmento del comando SSC-32 perteneciente al posicionamiento angular de todos los servos del robot).

2. Se realiza el cálculo de las incógnitas h_i y z_i del siguiente sistema de ecuaciones:

$$\begin{pmatrix} u_1 & h_1 & & & & & \\ h_1 & u_2 & h_2 & & & & \\ & h_2 & u_3 & h_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & h_{n-3} & u_{n-2} & h_{n-2} & \\ & & & & h_{n-2} & u_{n-1} & \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n-2} \\ z_{n-3} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{pmatrix}$$

cuyo significado fue ya descrito en el capítulo 4. Este cálculo se realiza 6 veces, una para cada servo, mediante llamadas al procedimiento **protected void velocidades(int n, int[] t, int[] q, double[] z, double[] h).**

3. Se crea un bucle que evalúa cada 10 ms el *spline* que representa la posición angular de cada uno de los servos en el instante actual y envía en ese instante el comando SSC-32 correspondiente a través del puerto serie. Se ha dividido el tiempo total en intervalos de 10 ms porque es la resolución mínima de los servos pertenecientes a las articulaciones del robot (equivale aproximadamente a 1º de rotación).

4. Una vez enviados todos los splines al robot, se provoca una pausa de 5 segundos en el proceso que ejecuta el procedimiento *enviar_splines* para que el robot se mantenga unos instantes en la última posición de la trayectoria ya completada. A continuación se envía el comando que manda al robot a su estado de reposo y se ejecuta otra pausa de 5 segundos.

- **protected void velocidades(int n, int[] t, int[] q, double[] z, double[] h)**

Es una adaptación en lenguaje C# del siguiente algoritmo descrito en pseudocódigo en el apartado 4.3 del capítulo 4:

input n,(t),(y)

for i=0,1,….,n-1 **do**
    $h_i := t_{i+1}-t_i$
    $b_i := 6(y_{i+1}-y_i)/h_i$
end

$u_1 := 2(h_0+ h_1)$
$v_1 := b_1- b_0$
for i=2,3,….,n-1 **do**
    $u_i := 2(h_i+ h_{i+1}) – (h_{i-1}^2 / u_{i-1})$
    $v_i := b_i - b_{i-1}- h_{i-1}(v_{i-1}/u_{i-1})$
end

$z_n :=0$
$z_0 :=0$
for i=n-1,…,1 **do**
    $z_i := (v_i - h_i z_{i+1})/u_i$
end

output (z)

- **protected double splinecubico(int n, int[] t, int[] q, double[] z, double[] h, double x)**

Devuelve el valor del *spline* cúbico que representa la posición angular de uno de los servos del robot en el instante de tiempo que marca el valor de la variable de entrada *x*, utilizando una transcripción a C# de la fórmula descrita en el capítulo 4.3. :

$$S_i(t) = y_i + (t - t_i)[C_i + (t - t_i)[B_i + (t - t_i)A_i]]$$

Donde:

$$A_i = \frac{1}{6h_i}(z_{i+1} - z_i)$$

$$B_i = \frac{z_i}{2}$$

$$C_i = -\frac{h_i}{6}z_{i+1} - \frac{h_i}{3}z_i + \frac{1}{h_i}(y_{i+1} - y_i)$$

6.7.5. Problemas surgidos y soluciones

Haciendo pruebas con el sistema completo en funcionamiento, se ha observado una situación en la que podría dañarse alguno de los servos del robot. Este problema es inherente a la naturaleza de los *splines* cúbicos, y se puede comprender fácilmente observando la siguiente figura (figura 41) que corresponde a la simulación de una trayectoria tipo *spline* cúbico:

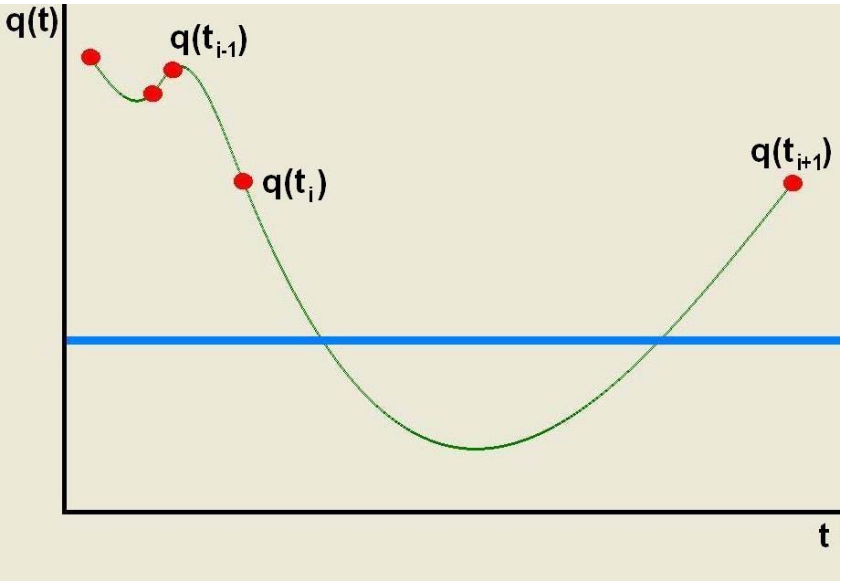


Figura 41 – Situación problemática en una trayectoria de tipo *spline* cúbico.

Si entendemos que *t* es el tiempo en milisegundos, q(t) la posición angular de uno de los servos pertenecientes a las articulaciones del robot en ese instante de tiempo, y la línia azul de la figura representa un límite que no se debería traspasar porque corresponde a una posición no accesible físicamente para el robot, deducimos lo siguiente:

Si se pretende interpolar mediante splines cúbicos entre tres nodos $q(t_{i-1})$, $q(t_i)$ y $q(t_{i+1})$, donde

- **$q(t_{i-1}) \leq q(t_i)$ y $q(t_i) \geq q(t_{i+1})$ o $q(t_{i-1}) \geq q(t_i)$ y $q(t_i) \leq q(t_{i+1})$.**

- **t_{i+1} es mucho mayor que t_i (es decir, que los dos últimos nodos se encuentren muy alejados en el tiempo).**

Si q_i se encuentra muy cerca de una frontera que separa los valores permitidos y prohibidos para q(t), la interpolación hará que en algún momento q(t) tome valores prohibidos durante un periodo de tiempo que dependerá de lo alejados que se encuentren t_i y t_{i+1}. Mientras más alejados se encuentren, mayor será ese periodo.

Esto extrapolado a nuestro manipulador implica que si los nodos de las trayectorias se encuentran demasiado alejados en el tiempo unos de otros, pueden ocurrir problemas como por ejemplo que el efector final del robot choque contra el suelo si uno de los nodos se

encuentra en una posición alta, el siguiente nodo cerca del suelo y el siguiente a este de nuevo en una posición alta con una distancia temporal elevada entre los dos últimos.

La solución a este problema se basa en restringir al usuario en la interfaz web la posibilidad de introducir tiempos demasiado elevados en el momento de la inserción de un nodo. Si el alumno inserta un tiempo de más de 1000ms se mostrará un mensaje de error y no se aceptará el nodo introducido.

7. CONCLUSIONES

El objetivo de este proyecto era crear un laboratorio remoto para las prácticas de la asignatura Robótica y Automatización Industrial impartida en la ETSE, en el cual los alumnos pudiesen acceder a este mediante una interfaz web desde cualquier lugar con conexión a internet y observar en directo cómo el robot situado en el laboratorio ejecuta trayectorias suaves de tipo *spline* cúbico que pasen por los puntos introducidos por ellos.

Este objetivo se ha cumplido en su totalidad, y el sistema se encuentra listo para poder utilizarse en las prácticas de la mencionada asignatura correspondientes al curso académico 2010/2011.

En cuanto al tiempo previsto para su desarrollo, se ha dilatado debido a ciertos imprevistos personales, pero no ha significado un problema ya que se ha podido terminar dentro del plazo oficial. A pesar de que se ha retrasado su finalización, realmente ha costado menos horas de trabajo de las inicialmente previstas. En el apartado “COSTES TEMPORALES” del capítulo 3 se calculaba que serían necesarias unas 240 h para la realización del proyecto. Sin embargo finalmente el coste ha sido de 180 h.

En resumen, podemos decir que se han alcanzado satisfactoriamente todos los objetivos previstos.

8. FUTURAS LÍNEAS DE TRABAJO

El sistema se ha diseñado de manera que sea relativamente sencillo modificarlo o ampliarlo. La tecnología utilizada para el desarrollo de la aplicación web, que como ya se ha dicho facilita la separación por capas de la programación y la modularidad, ha contribuído enormemente a ello.

Podrían añadirse en un futuro interesantes funcionalidades como las siguientes:

- Hacer que los usuarios puedan manipular el robot en tiempo real, mediante una interfaz gráfica que simule un *joystick* analógico.

- Realizar un sistema multi-robot para colaboración en tareas PPO.

- Añadir al robot sensores e implementar técnicas de corrección de la trayectoria para evitar obstáculos físicos dentro del entorno de trabajo del manipulador.

BIBLIOGRAFÍA

[1] "Ventajas de ASP.NET 2.0". *icreativos* [artículo en línea]. [Fecha de consulta: 3 de Febrero del 2010].

<http://www.icreativos.com/articulos/-ventajas-de-asp-net-2-0.aspx?AspxAutoDetectCookieSupport=1>

[2] (2010, 26 de Mayo**). "**Robot Industrial". *Wikipedia* [artículo en línea]. [Fecha de consulta: 5 de Febrero del 2010].

<http://es.wikipedia.org/wiki/Robot_industrial>

[3] "Robótica". *El rincón del vago* [artículo en línea]. [Fecha de consulta: 5 de Febrero del 2010].

<http://html.rincondelvago.com/robotica_1.html>

[4] Craig, John J. (2006). *Introduction to Robotics: Mechanics and Control* ['Robótica'] (3ª ed.). Pearson - Prentice Hall.

[5] Cerquera Rojas, Yamil Armando**. "**Interpolación con trazadores o splines". *Universidad Surcolombiana* [documento PDF].

[6] (2007, 20 de Septiembre**).** Candelas Herías, Francisco A.; Corrales Ramón, Juan A. "Servomotores". *Universidad de Alicante* [publicación interna].

[7] Vargas del Valle, Ricardo J.; Maltés Granados, Juan P. "Programación en capas". *Universidad de Costa Rica* [documento PDF].

[8] (2010, 31 de Mayo). "Programación por capas". *Wikipedia* [artículo en línea]. [Fecha de consulta: 20 de Febrero del 2010].

<http://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas>

[9] (2001, 3 de Febrero). Domingo, Juan. "Robótica. Apuntes para la asignatura". *Universidad de Valencia* [Apuntes en formato PDF].

[10] (2009, 18 de Febrero). Frye, Jym (traducción al español por Alicia Bernal). "Manual de SSC-32". S*uperrobotica.com* [documento en línea]. [Fecha de consulta: 3 de Enero del 2010].

<http://www.superrobotica.com/download/S310185/S310185_Esp.pdf>

[11] "Axis 297/207W/207MW Network Cameras". *www.axis.com* [documento en línea]. [Fecha de consulta: 2 de Marzo del 2010].

<http://www.axis.com/files/datasheet/ds_207_combo_32848_en_0904_lo.pdf>

[12] (2010, 23 de Abril). "Internet Information Services". *Wikipedia* [artículo en línea]. [Fecha de consulta: 5 de Marzo del 2010].

< http://es.wikipedia.org/wiki/Internet_Information_Services>

[13] (2002, 16 de Diciembre).Alvarez, Miguel Angel. "Instalación de IIS en Windows XP Profesional". *Desarrolloweb.com* [artículo en línea]. [Fecha de cosulta: 5 de Marzo del 2010].

<http://www.desarrolloweb.com/articulos/1001.php>

[14] (2010, 2 de Junio). "Microsoft SQL Server". *Wikipedia* [artículo en línea]. [Fecha de consulta: 15 de Marzo del 2010].

<http://es.wikipedia.org/wiki/Microsoft_SQL_Server>

[15] (2003, 10 de Abril). "CÓMO: Configurar permisos de servidor Web para contenido Web en IIS". *Soporte Microsoft* [artículo en línea]. [Fecha de consulta: 6 de Marzo del 2010].

<http://support.microsoft.com/kb/313075/es>

[16] (2007, Noviembre). "Controles de servidor ASP.NET". *MSDN Library* [artículo en línea]. [Fecha de consulta: 10 de Diciembre del 2010].

<http://msdn.microsoft.com/es-es/library/bb386416%28v=VS.90%29.aspx>

[17] "How To Embed WMV (Windows Media Video) File to Playback with WMP Player on Web Page". *My Digital Life* [artículo en línea]. [Fecha de consulta: 28 de Noviembre del 2010].

<http://www.mydigitallife.info/2009/07/23/how-to-embed-wmv-windows-media-video-file-to-playback-with-wmp-player-on-web-page/>

[18] "SERVO MOTOR HITEC HS475HB S330170". *Superrobotica.com* [artículo en línea]. [Fecha de consulta: 28 de Febrero del 2010].

< http://www.superrobotica.com/S330170.htm>

[19] "HS-85BB Premium Micro Servo". *Hitec.com* [artículo en línea]. [Fecha de consulta: 28 de Febrero del 2010].

<http://www.hitecrcd.com/products/servos/analog/micro-mini/hs-85bb.html>

[20] "HiTec HS-85BB Ball Bearing Micro Servo". *Hobby Horse* [artículo en línea]. [Fecha de consulta: 28 de Febrero del 2010].

<http://www.hobbyhorse.com/hitec_hs85bb.shtml>

[21] "HiTec HS-81 Standard Micro Servo". *Hoby Horse* [artículo en línea]. [Fecha de consulta: 28 de Febrero del 2010].

<http://www.hobbyhorse.com/hitec_hs81.shtml>

[22] J.Angeles, (2003) *Fundamentals of Robotc Mechanical Systems*. New York: Springer-Verlag.

ÍNDICE DE ANEXOS

ÍNDICE DE FIGURAS

Firmado: **Marcos Mohedano Gómez**

Bellaterra, Junio 2010

La solución a los problemas de disponibilidad horaria para la realización de sesiones prácticas por parte de los estudiantes se encuentra en los laboratorios remotos, que permiten a estos interactuar con los elementos instalados en los laboratorios sin necesidad de estar presentes físicamente. Este proyecto pretende crear un laboratorio remoto para la asignatura "Robótica y Automatización Industrial" impartida en la ETSE, UAB, en el cual los estudiantes puedan ejecutar trayectorias de tipo *spline* cúbico en un brazo robot y observar a través de vídeo en tiempo real los movimientos del robot desde cualquier lugar con conexión a Internet.

La solució als problemes de disponibilitat horària per a la realització de sessions pràctiques per part dels estudiants es troba en els laboratoris remots, que permeten a aquests interactuar amb els elements instal·lats en els laboratoris sense necessitat d'estar presents físicament. Aquest projecte pretén crear un laboratori remot per a l'assignatura "Robòtica i Automatització Industrial" impartida a l'ETSE, UAB, en el qual els estudiants puguin executar trajectòries de tipus spline cúbic en un braç robot i observar a través de vídeo en temps real els moviments del robot des de qualsevol lloc amb connexió a Internet.

The solution to the problems of time availability for conducting practical sessions by students is in remote laboratories, which allow them to interact with the elements installed in laboratories without being physically present. This project aims to create a remote laboratory for the course "Robòtica i Automatització Industrial" given at ETSE, UAB, in which students can perform cubic spline trajectories in a robot arm and watch through real-time video the movements of the robot from anywhere with Internet access.