



**Universitat Autònoma  
de Barcelona**

**DISEÑO Y PUESTA EN MARCHA DE UN  
LABORATORIO REMOTO DE ROBÓTICA**

Memoria del proyecto  
de Ingeniería Técnica en  
Informática de Sistemas

Realizado por

**Ramón González Zambrano**

Y dirigido por

**Asier Ibeas Hernández**

**Escola Universitaria d'Informàtica**

**Sabadell, Julio 2009**

El/la sotasignant, Asier Ibeas Hernández,  
professor/a de l'Escola Universitària d'Informàtica de la UAB,

**CERTIFICA:**

Que el treball al que correspon la present memòria  
ha estat realitzat sota la seva direcció  
per en Ramón González Zambrano

I per a que consti firma la present.  
Sabadell, Julio de 2009

-----

Signat: Asier Ibeas Hernández



## RESUMEN

Este proyecto surge de la iniciativa de mejorar la calidad docente de las prácticas en la asignatura Robótica y Automatización Industrial impartida en la ETSE (Escola Tècnica Superior d'Enginyeria) de la UAB (Universidad Autónoma de Barcelona), mediante un sistema innovador.

El objetivo es sustituir las actuales prácticas de dicha asignatura, las cuales se basan en la realización de simulaciones en entorno MATLAB para verificar las ecuaciones que gobiernan a los robots manipuladores, por un entorno de prácticas más atractivo consistente en un robot manipulador real, el cual podrá ser programado para la realización de tareas de PPO (*Pick and Place Operation*), que consiste en trasladar objetos de un punto a otro y manipularlos.

Para ello se tratarán dos aspectos bien diferenciados, por un lado la parte de hardware que consistirá en realizar el diseño y puesta en marcha de una plataforma hardware, a la cual se le podrán dar diversos usos, y por otro lado la parte de software que consistirá en desarrollar el software de control necesario para la plataforma hardware, además del diseño de un lenguaje de programación para el robot. Por último se ha dotado al sistema de una webcam para darle la funcionalidad remota y que la plataforma pueda ser utilizada desde el exterior del aula del laboratorio.

Así pues el alumno dispondrá de una aplicación para verificar los programas que realice, y una interfaz web para poder enviar dichos programas al servidor desde cualquier ordenador conectado a Internet, pudiendo observar gracias a la webcam, los movimientos y acciones del robot de forma remota, en el caso que se encontrase fuera del aula del laboratorio.

## INDICE

<b>1. INTRODUCCIÓN.....</b>	<b>11</b>
1.1 ÁMBITO .....	11
1.2 PRÓPOSITO DEL PROYECTO Y OBJETIVOS .....	11
1.3 RESULTADOS ESPERADOS .....	13
1.4 INNOVACIÓN DOCENTE.....	13
<b>2. ESTUDIO TEÓRICO .....</b>	<b>15</b>
2.1 ROBOTS MANIPULADORES.....	15
2.1.1 Estructura .....	15
2.1.2 Principales características de los robots.....	16
2.1.3 Tipos de configuraciones morfológicas .....	18
2.2 EL PUERTO SERIE RS232.....	19
2.2.1 El puerto serie en el PC.....	20
<b>3. ESTUDIO DE VIABILIDAD .....</b>	<b>22</b>
3.1 OBJETO.....	22
3.1.1 Descripción de la situación actual .....	22
3.1.2 Perfil del cliente – usuario .....	22
3.1.3 Objetivos .....	22
3.1.4 Fuentes de información.....	23
3.2 SISTEMA A REALIZAR .....	24
3.2.1 Descripción .....	24
3.2.2 Modelo de desarrollo .....	26
3.2.3 Recursos .....	27
3.2.4 Viabilidad técnica .....	27
3.2.5 Análisis coste .....	29
3.2.6 Presupuesto.....	31
3.2.7 Viabilidad económica .....	32
3.2.8 Evaluación de riesgos .....	33
3.2.9 Alternativas .....	34
3.3 PLANIFICACIÓN .....	35
3.4 CONCLUSIONES .....	37
<b>4. DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA.....</b>	<b>38</b>
4.1 MÓDULO 1 – ROBOT.....	39
4.1.1 Descripción del módulo .....	39
4.1.2 Entradas y salidas.....	39
4.1.3 Tecnologías utilizadas .....	40
4.1.4 Tiempo de desarrollo.....	43
4.1.5 Implementación .....	43
4.1.6 Problemas y soluciones .....	57
4.2 MÓDULO 2 - CIRCUITO DE CONTROL SSC-32 .....	58
4.2.1 Descripción del módulo .....	58
4.2.2 Entradas y Salidas .....	58

4.2.3	Tecnologías utilizadas .....	59
4.3	MÓDULO 3 – SECUENCIADOR DE OPERACIONES .....	61
4.3.1	Descripción del módulo .....	61
4.3.2	Entradas y salidas.....	62
4.3.3	Tecnologías utilizadas .....	62
4.3.4	Tiempo de desarrollo.....	62
4.3.5	Implementación .....	63
4.3.6	Problemas y soluciones .....	80
4.4	MÓDULO 4 – VERIFICADOR .....	81
4.4.1	Descripción del módulo .....	81
4.4.2	Entradas y salidas.....	82
4.4.3	Tecnologías utilizadas .....	82
4.4.4	Tiempo de desarrollo.....	82
4.4.5	Implementación .....	82
4.4.6	Problemas y soluciones .....	83
4.5	MÓDULO 5 – ARCHIVO.....	84
4.5.1	Descripción del módulo .....	84
4.6	MÓDULO 6 – EDITOR DE TEXTO .....	85
4.6.1	Descripción del módulo .....	85
4.6.2	Entradas y salidas.....	85
4.7	MÓDULO 7 – SERVIDOR .....	86
4.7.1	Descripción del módulo .....	86
4.7.2	Entradas y salidas.....	86
4.7.3	Tecnologías utilizadas .....	86
4.7.4	Tiempo de desarrollo.....	86
4.7.5	Implementación .....	86
4.7.6	Problemas y soluciones .....	92
4.8	MÓDULO 8 – INTERFAZ WEB.....	93
4.8.1	Descripción del módulo .....	93
4.8.2	Entradas y salidas.....	100
4.8.3	Tecnologías utilizadas .....	101
4.8.4	Tiempo de desarrollo.....	101
4.8.5	Implementación .....	101
4.8.6	Problemas y soluciones .....	116
4.9	MÓDULO 9 – APLICACIÓN DE TRANSMISIÓN DE VIDEO.....	117
4.9.1	Descripción del módulo .....	117
4.9.2	Entradas y salidas.....	117
4.9.3	Tecnologías utilizadas .....	117
4.9.4	Tiempo de desarrollo.....	117
4.9.5	Implementación .....	117
4.9.6	Problemas y soluciones .....	119
4.10	MÓDULO 10 – WEBCAM.....	120
4.10.1	Descripción del módulo .....	120
4.10.2	Entradas y salidas.....	120
4.10.3	Tecnologías utilizadas .....	120
4.11	MÓDULO 11 – GESTOR DE ARCHIVOS.....	121
4.11.1	Descripción del módulo .....	121
4.11.2	Entradas y salidas.....	122
4.11.3	Tecnologías utilizadas .....	122
4.11.4	Tiempo de desarrollo.....	122

4.11.6	<i>Problemas y soluciones</i> .....	126
4.12	MÓDULO 12 – PC.....	127
4.12.1	<i>Descripción del módulo</i> .....	127
4.12.2	<i>Entradas y salidas</i> .....	127
<b>5.</b>	<b>DOCUMENTACIÓN</b> .....	<b>128</b>
<b>6.</b>	<b>AMPLIACIONES FUTURAS</b> .....	<b>129</b>
<b>7.</b>	<b>CONCLUSIONES</b> .....	<b>130</b>
<b>8.</b>	<b>ENLACES WEB</b> .....	<b>133</b>
<b>9.</b>	<b>GLOSARIO</b> .....	<b>138</b>
<b>10.</b>	<b>INDICE DE ANEXOS</b> .....	<b>141</b>

## **1. INTRODUCCIÓN**

### **1.1 ÁMBITO**

Este proyecto se basa en el diseño y la puesta en marcha de un laboratorio remoto de robótica que servirá para realizar las prácticas de laboratorio de la asignatura Robótica y Automatización Industrial impartida en la ETSE a partir del curso 2009/2010.

Las prácticas actuales se basan en realizar simulaciones bajo entorno MATLAB para verificar las ecuaciones que gobiernan a los robots manipuladores, estas prácticas serán sustituidas por las prácticas diseñadas para el laboratorio remoto, las cuales mediante este sistema, permitirán la mejor asimilación de la materia por parte del alumno, mejorará la calidad de la docencia y atraerá a nuevos alumnos hacia el ámbito de la robótica industrial.

Para ello se desarrollará un sistema que constará de una plataforma hardware (robot y entorno), el software de control del robot (que permitirá la programación y el uso compartido dentro del aula) y por último se le dotará de la funcionalidad remota que permitirá el uso de la plataforma desde el exterior del aula del laboratorio.

### **1.2 PRÓPOSITO DEL PROYECTO Y OBJETIVOS**

Este proyecto pretende sustituir las prácticas actuales basadas en MATLAB, por unas nuevas diseñadas bajo un sistema basado en un robot manipulador de bajo coste (Ver Fig. 1).

Para ello los objetivos a alcanzar son los siguientes:

- a. Montaje de la plataforma hardware.
  - a.1 Ensamblaje de todas las piezas hasta construir un robot manipulador
  - a.2 Montaje de un entorno de trabajo sobre el que actuará el robot.
  
- b. Comunicaciones
  - b.1 Crear una aplicación para la comunicación serie con el robot.
  
- c. Secuenciador de operaciones
  - c.1 Diseñar un lenguaje de programación para el robot.
  - c.2 Crear una aplicación que traduzca los programas creados mediante etiquetas al código ensamblador que entiende el circuito SSC-32.



## d. Interfaz web

- d.1 Creación de la página web para habilitar la funcionalidad remota.
- d.2 Integración de los perfiles de usuarios para la autenticación.
- d.3 Incorporación de una funcionalidad para el envío de ficheros.
- d.3 Integración de la aplicación de video *streaming*.

## e. Servidor web

- e.1 Instalación de un servidor para *hosting* web.
- e.2 Diseño de una aplicación para la captura de imágenes desde la webcam y enviarlas a la página web.

## f. Documentación

- f.1 Creación del manual del usuario alumno.
- f.2 Creación del manual del usuario administrador.

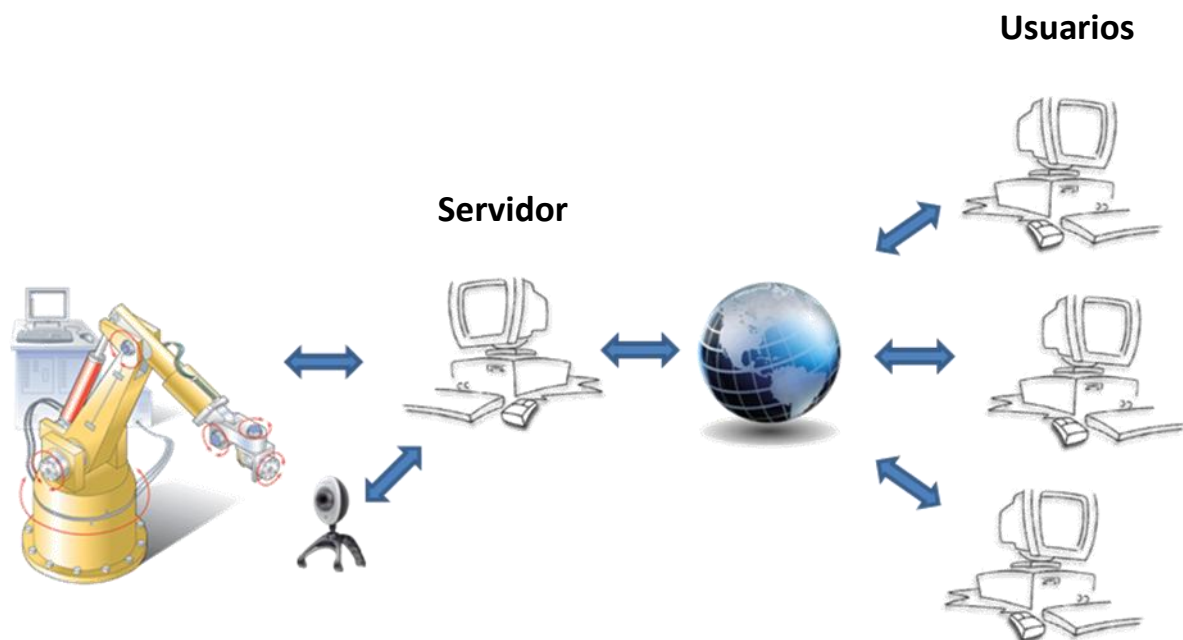


Fig. 1 – Esquema global del sistema

### 1.3 RESULTADOS ESPERADOS

Se espera conseguir un sistema con el cual, un alumno cree un programa en un fichero de texto, lo verifique con el software desarrollado, y si el resultado es correcto se conecte a la página web del laboratorio remoto donde deberá autenticarse y una vez realizada la autenticación tenga acceso al laboratorio, donde podrá enviar su archivo de texto con el código poniéndose automáticamente una cola de espera de archivos para ser ejecutados. A través de las imágenes emitidas por una webcam el alumno podrá observar en la página web las acciones que está realizando el robot y la misma página le avisará en el momento en que su archivo sea el que está siendo ejecutado, para así poder observar si realmente el robot realiza lo que él pretendía.

### 1.4 INNOVACIÓN DOCENTE

Hay que remarcar el carácter innovador de este proyecto, ya que es un sistema actualmente inexistente. Tras una intensiva búsqueda, se han encontrado dos proyectos en marcha que persiguen un objetivo similar al de este proyecto, un proyecto ya desarrollado por la Universidad de Granada, que consiste en una plataforma para la realización de proyectos en el ámbito de la robótica móvil basados en FPGAs (del inglés *Field Programmable Gate Array*), pero carece de la vertiente remota, y el otro proyecto está propuesto por el departamento de Ciencias de la Computación de la Facultad de Economía y Administración de la Universidad Nacional de Comahue, quienes pretenden realizar un sistema muy similar al que se ha desarrollado aquí, para incidir positivamente sobre campos problemáticos que son transversales en sus escuelas: fracaso escolar, debilidad en la definición del área informática y acceso de los alumnos a las TICs con el único fin del esparcimiento, entre otros.

El sistema que se pretende desarrollar persigue el objetivo de poder enseñar mejor a los alumnos las características que tiene un robot manipulador, los métodos de programación existentes y sus problemas, por esto se ha escogido un robot de bajo coste que no tiene altas prestaciones pero sí las suficientes para alcanzar los objetivos planteados, además de poder ser una plataforma colaborativa para el posible uso compartido con otros centros docentes y para la iniciación de nuevas líneas de proyecto.

Aquí se muestra una imagen del robot manipulador que constituye la base de este proyecto. (Ver Fig. 2)

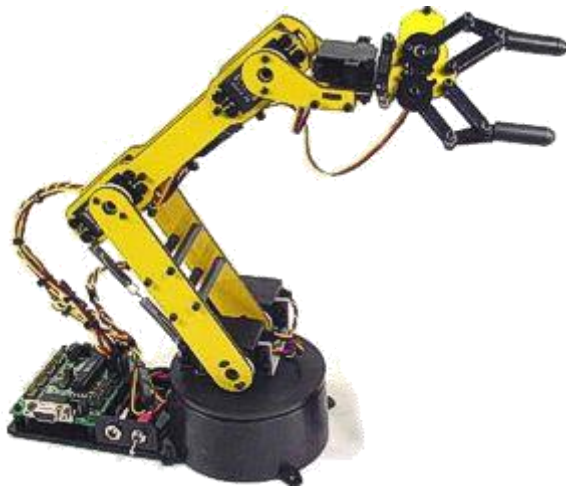


Fig. 2 – Robot manipulador

Dado que se diseñará el software necesario para controlar el robot de forma remota, esto trae consigo una serie de ventajas, como son: la posibilidad de utilizar el laboratorio en cualquier momento y desde cualquier lugar, la de ofrecer acceso a quien se desee gracias a la aplicación de autenticación de la página web, la de guardar todos los archivos ejecutados sobre el robot por cada alumno, para así realizar una evaluación continua y analizar la evolución de su aprendizaje, entre otras.

Ahora se muestra una imagen del actual entorno de prácticas de la asignatura (Ver Fig.3):

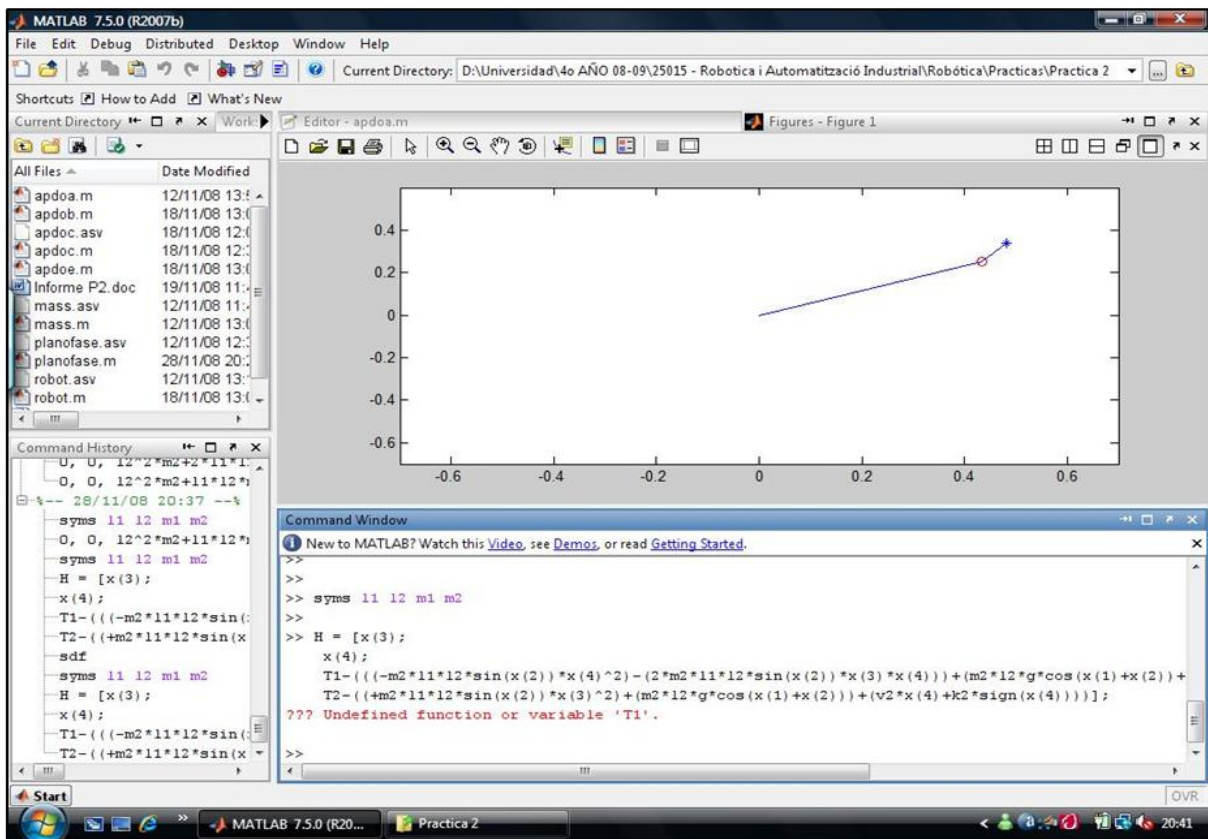


Fig. 3 – Captura de prácticas en Matlab

## 2. ESTUDIO TEÓRICO

A continuación se introducen una serie de conceptos que serán necesarios para la discusión de la viabilidad técnica del proyecto.

### 2.1 ROBOTS MANIPULADORES

#### 2.1.1 Estructura

Cabe destacar que la característica antropomórfica más común en nuestros días es la de un brazo mecánico, el cual realiza diversas tareas industriales. Existen en el mercado diversas empresas dedicadas a la fabricación de robots industriales por lo que existen diferentes marcas y modelos.

Un robot está formado por los siguientes elementos: estructura mecánica, transmisiones, actuadores, sensores, elementos terminales y controlador. Aunque los elementos empleados en los robots no son exclusivos de estos (máquinas herramientas y otras muchas máquinas emplean tecnologías semejantes), las altas prestaciones que se exigen a los robots han motivado que en ellos se empleen elementos con características específicas.

La constitución física de la mayor parte de los robots industriales guarda cierta similitud con la anatomía de las extremidades superiores del cuerpo humano, por lo que, en ocasiones, para hacer referencia a los distintos elementos que componen el robot, se usan términos como cintura, hombro, brazo, codo, muñeca, etc. (Ver Fig. 4) (Enlace 1).

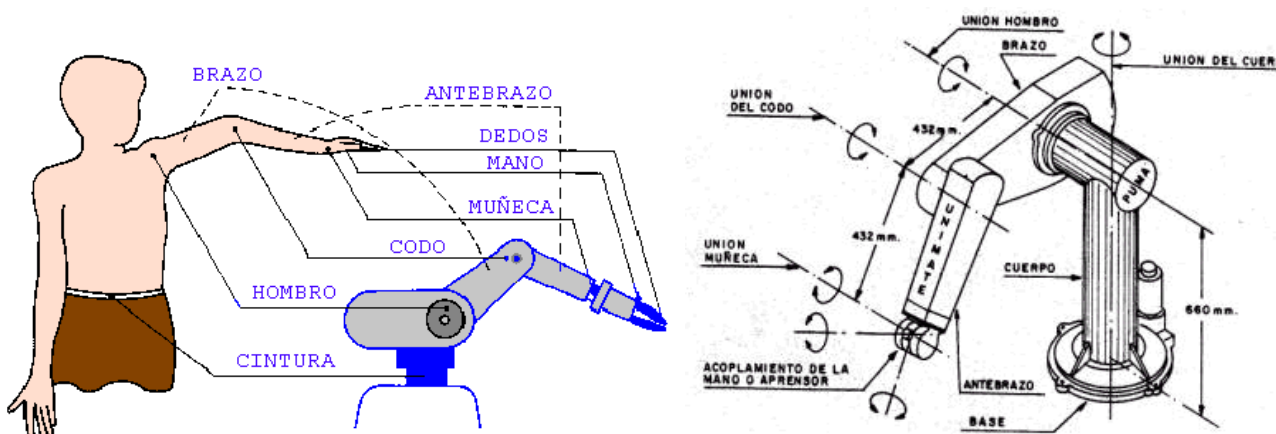
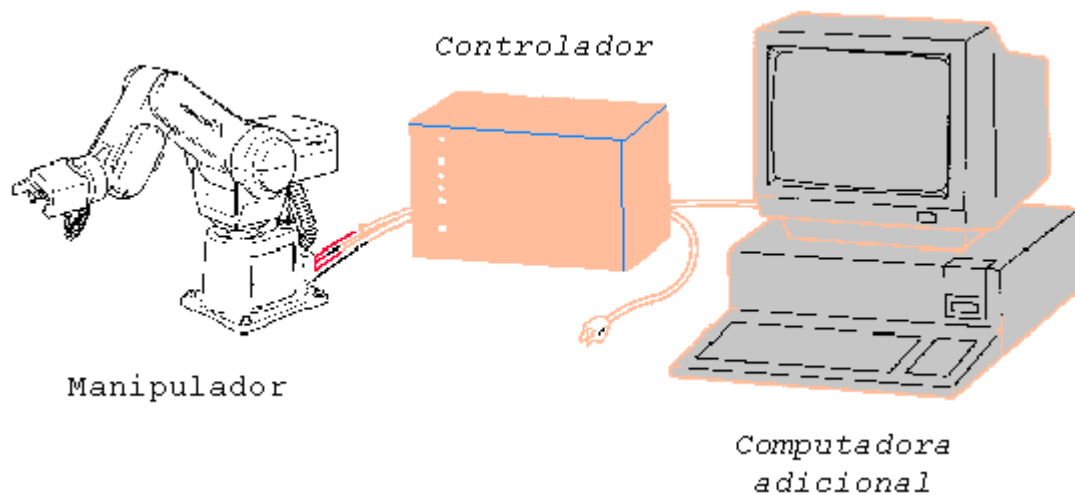


Fig. 4 – Esquema antropomórfico

Los elementos que forman parte de la totalidad del robot son (Ver Fig. 5):

- Manipulador
- Controlador
- Dispositivos de entrada y salida de datos



**Fig. 5 – Elementos del robot**

### **2.1.2 Principales características de los robots**

A continuación se describen las características más relevantes propias de los robots:

#### **2.1.2.1 Grados de libertad**

Cada uno de los movimientos independientes (giros y desplazamientos) que puede realizar cada articulación con respecto a la anterior. Son los parámetros que se precisan para determinar la posición y la orientación del elemento terminal del manipulador. El número de grados de libertad del robot viene dado por la suma de los GDL de las articulaciones que lo componen. Puesto que las articulaciones empleadas suelen ser únicamente de rotación y prismáticas, con un solo grado de libertad cada una, el número de GDL del robot suele coincidir con el número de articulaciones que lo componen.

#### **2.1.2.2 Espacio de trabajo**

El volumen de trabajo de un robot se refiere únicamente al espacio dentro del cual puede desplazarse el extremo de su muñeca. Para determinar el volumen de trabajo no se toma en cuenta el actuador final. La razón de ello es que a la muñeca del robot se le pueden adaptar *grippers* de distintos tamaños.

### 2.1.2.3 Precisión de los movimientos

La precisión de movimiento en un robot industrial depende de tres factores:

- Resolución espacial
- Exactitud
- Repetibilidad

### 2.1.2.4 Capacidad de carga

El peso que puede transportar la pinza del manipulador recibe el nombre de capacidad de carga. A veces, este dato lo proporcionan los fabricantes, incluyendo el peso de la propia pinza.

### 2.1.2.5 Velocidad

Se refiere a la velocidad máxima alcanzable por el TCP (del inglés *Tool Center Point*) o Punto Central de la Herramienta, o por las articulaciones. En muchas ocasiones, una velocidad de trabajo elevada, aumenta extraordinariamente el rendimiento del robot, por lo que esta magnitud se valora considerablemente en la elección del mismo.

### 2.1.2.6 Tipo de actuadores

Los actuadores son los elementos motrices que generan el movimiento de las articulaciones, y pueden ser, según la energía que consuman, de tipo oleohidráulico, neumático o eléctrico (Enlace 2)

### 2.1.2.7 Programabilidad

La inclusión del controlador de tipo microelectrónica en los robots industriales, permite la programación del robot de muy diversas formas.

En general, los modernos sistemas de robots admiten la programación manual, mediante un modulo de programación.

Las programaciones gestual y textual, controlan diversos aspectos del funcionamiento del manipulador:

- Control de la velocidad y la aceleración.
- Saltos de programa condicionales.
- Temporizaciones y pausas.
- Edición, modificación, depuración y ampliación de programas.
- Funciones de seguridad.
- Funciones de sincronización con otras maquinas.
- Uso de lenguajes específicos de Robótica.

### 2.1.3 Tipos de configuraciones morfológicas

La estructura del manipulador y la relación entre sus elementos proporcionan una configuración mecánica, que da origen al establecimiento de los parámetros que hay que conocer para definir la posición y orientación del elemento terminal. Fundamentalmente, existen cuatro estructuras clásicas en los manipuladores, que se relacionan con los correspondientes modelos de coordenadas en el espacio y que se citan a continuación: cartesianas, cilíndricas, esféricas, angulares. Así, el brazo del manipulador puede presentar cuatro configuraciones clásicas:

- Cartesiana
- Cilíndrica
- Esférica
- De brazo articulado,

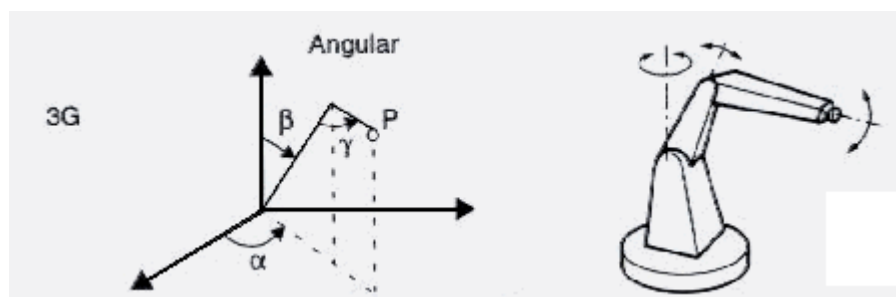
Y una no clásica:

- SCARA (Selective Compliance Assembly Robot Arm).

A continuación se describe con más detalle la configuración que más interesa, puesto que es la que posee el robot manipulador del proyecto.

#### **Configuración de brazo articulado**

También llamada Articulación esférica, Articulación coordinada, Rotación o Angular. El robot usa 3 juntas de rotación para posicionarse. Generalmente, el volumen de trabajo es esférico. Estos tipos de robot se parecen al brazo humano, con una cintura, el hombro, el codo, la muñeca. Presenta una articulación con movimiento rotacional y dos angulares. Aunque el brazo articulado puede realizar el movimiento llamado interpolación lineal (para lo cual requiere mover simultáneamente dos o tres de sus articulaciones), el movimiento natural es el de interpolación por articulación, tanto rotacional como angular. (Ver Fig. 6)



**Fig. 6 – Esquema de la configuración de brazo articulado**

## 2.2 EL PUERTO SERIE RS232

El puerto serie RS-232C (Enlace 3), presente en todos los ordenadores actuales, es la forma más comúnmente usada para realizar transmisiones de datos entre ordenadores. El RS-232C es un estándar que constituye la tercera revisión de la antigua norma RS-232, propuesta por la EIA (Asociación de Industrias Electrónicas), realizándose posteriormente una versión internacional por el CCITT (Comité Consultivo Internacional Telegráfico y Telefónico), conocida como V.24. Las diferencias entre ambas son mínimas, por lo que a veces se habla indistintamente de V.24 y de RS-232C (incluso sin el sufijo "C"), refiriéndose siempre al mismo estándar.

El RS-232C consiste en un conector tipo DB-25 de 25 pines (Ver Fig. 7), aunque es normal encontrar la versión de 9 pines DB-9 (Ver Fig. 8), mas barato e incluso más extendido para cierto tipo de periféricos (como el ratón serie del PC (del inglés *Personal Computer*). En cualquier caso, los PCs no suelen emplear más de 9 pines en el conector DB-25. Las señales con las que trabaja este puerto serie son digitales, de +12V (0 lógico) y -12V (1 lógico), para la entrada y salida de datos, y a la inversa en las señales de control. El estado de reposo en la entrada y salida de datos es -12V. Dependiendo de la velocidad de transmisión empleada, es posible tener cables de hasta 15 metros.

Cada pin puede ser de entrada o de salida, teniendo una función específica cada uno de ellos. Las más importantes son:

Pin	Función
TXD	(Transmitir Datos)
RXD	(Recibir Datos)
DTR	(Terminal de Datos Listo)
DSR	(Equipo de Datos Listo)
RTS	(Solicitud de Envío)
CTS	(Libre para Envío)
DCD	(Detección de Portadora)

Las señales TXD, DTR y RTS son de salida, mientras que RXD, DSR, CTS y DCD son de entrada. La masa de referencia para todas las señales es SG (Tierra de Señal).



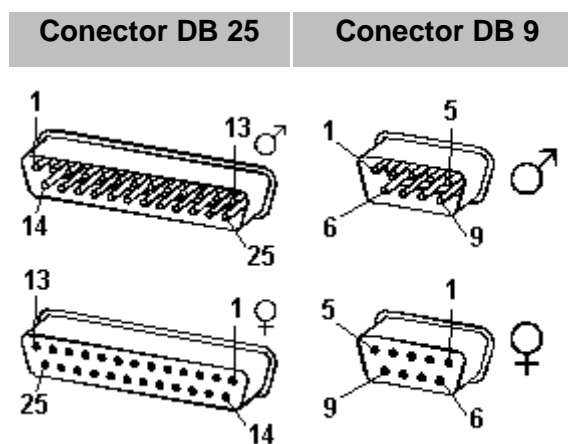


Fig. 7 – Conector DB25

Fig. 8 – Conector DB9

### 2.2.1 El puerto serie en el PC

El ordenador controla el puerto serie mediante un circuito integrado específico, llamado UART (Transmisor-Receptor-Asíncrono Universal). Normalmente se utilizan los siguientes modelos de este chip: 8250 (bastante antiguo, con fallos, solo llega a 9600 baudios), 16450 (versión corregida del 8250, llega hasta 115.200 baudios) y 16550A (con buffers de E/S). A partir de la gama Pentium, la circuitería UART de las placa base son todas de alta velocidad, es decir UART 16550A. De hecho, la mayoría de los módems conectables a puerto serie necesitan dicho tipo de UART, incluso algunos juegos para jugar en red a través del puerto serie necesitan de este tipo de puerto serie. Por eso hay veces que un 486 no se comunica con la suficiente velocidad con un PC Pentium... Los portátiles suelen llevar otros chips: 82510 (con buffer especial, emula al 16450) o el 8251 (no es compatible).

Para controlar al puerto serie, la CPU emplea direcciones de puertos de E/S y líneas de interrupción (IRQ). En el AT-286 se eligieron las direcciones 3F8h (o 0x3f8) e IRQ 4 para el COM1, y 2F8h e IRQ 3 para el COM2. El estándar del PC llega hasta aquí, por lo que al añadir posteriormente otros puertos serie, se eligieron las direcciones 3E8 y 2E8 para COM3-COM4, pero las IRQ no están especificadas. Cada usuario debe elegir las de acuerdo a las que tenga libres o el uso que vaya a hacer de los puertos serie (por ejemplo, no importa compartir una misma IRQ en dos puertos siempre que no se usen conjuntamente, ya que en caso contrario puede haber problemas).

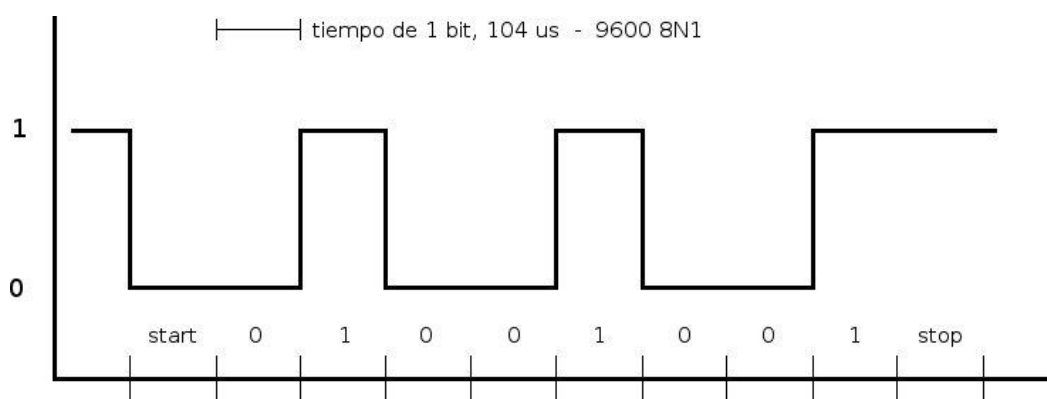
Mediante los puertos de E/S se pueden intercambiar datos, mientras que las IRQ producen una interrupción para indicar a la CPU que ha ocurrido un evento (por ejemplo, que ha llegado un dato, o que ha cambiado el estado de algunas señales de entrada). La CPU debe responder a estas interrupciones lo más rápido posible, para que dé tiempo a recoger el dato antes de que el siguiente lo sobrescriba. Sin embargo, las UART 16550A incluyen unos buffers de tipo FIFO, dos de 16 bytes (para recepción y transmisión), donde se pueden guardar varios datos antes de que la CPU los recoja. Esto también disminuye el número de interrupciones por segundo generadas por el puerto serie.

El RS-232 puede transmitir los datos en grupos de 5, 6, 7 u 8 bits, a unas velocidades determinadas (normalmente, 9600 bits por segundo o más). Después de la transmisión de los datos, le sigue un bit opcional de paridad (indica si el número de bits transmitidos es par o impar, para detectar fallos), y después 1 o 2 bits de Stop. Normalmente, el protocolo utilizado es 8N1 (que significa, 8 bits de datos, sin paridad y con 1 bit de Stop).

Una vez que ha comenzado la transmisión de un dato, los bits tienen que llegar uno detrás de otro a una velocidad constante y en determinados instantes de tiempo. Por eso se dice que el RS-232 es asíncrono por carácter y síncrono por bit. Los pines que portan los datos son RXD y TXD. Las demás se encargan de otros trabajos: DTR indica que el ordenador está encendido, DSR que el aparato conectado a dicho puerto está encendido, RTS que el ordenador puede recibir datos (porque no está ocupado), CTS que el aparato conectado puede recibir datos, y DCD detecta que existe una comunicación, presencia de datos.

Tanto el aparato a conectar como el ordenador (o el programa terminal) tienen que usar el mismo protocolo serie para comunicarse entre sí. Puesto que el estándar RS-232 no permite indicar en qué modo se está trabajando, es el usuario quien tiene que decidirlo y configurar ambas partes. Como ya se ha visto, los parámetros que hay que configurar son: protocolo serie (8N1), velocidad del puerto serie, y protocolo de control de flujo. Este último puede ser por hardware (el que ya hemos visto, el *handshaking* RTS/CTS) o bien por software (XON/XOFF, el cual no es muy recomendable ya que no se pueden realizar transferencias binarias). La velocidad del puerto serie no tiene por qué ser la misma que la de transmisión de los datos, de hecho debe ser superior. Por ejemplo, para transmisiones de 1200 baudios es recomendable usar 9600, y para 9600 baudios se pueden usar 38400 (o 19200).

Este es el diagrama de transmisión de un dato con formato 8N1 (Ver Fig. 9). El receptor indica al emisor que puede enviarle datos activando la salida RTS. El emisor envía un bit de START (nivel alto) antes de los datos, y un bit de STOP (nivel bajo) al final de estos.



**Fig. 9 – Diagrama de transmisión de un dato con formato 8N1**

### **3. ESTUDIO DE VIABILIDAD**

#### **3.1 OBJETO**

##### **3.1.1 Descripción de la situación actual**

Actualmente la ETSE dispone de dos robots manipuladores que forman parte de una célula de fabricación flexible que hay montada en uno de los laboratorios, pero dichos robots llevan años inoperativos debido a un problema con el servicio técnico de los mismos. Debido a esto las prácticas de la asignatura Robótica y Automatización Industrial se realizan sobre entorno MATLAB.

Dichas prácticas permiten realizar simulaciones sobre la dinámica de los robots manipuladores y verificar que MATLAB es una herramienta de cálculo útil a la hora de resolver ecuaciones y los problemas de CD (Cinemática Directa) y CI (Cinemática Inversa), resueltos de forma manual en las clases de problemas.

Pero en cierto modo estas prácticas sirven para verificar si los cálculos que se realizaron a mano eran correctos o no, pero impiden abordar problemas tales como, la secuenciación de operaciones, planificación del trabajo a realizar por un robot e incluso la propia programación de un robot.

##### **3.1.2 Perfil del cliente – usuario**

El perfil de usuario se corresponde con el de un alumno estudiante de Ingeniería en Informática de 4º curso, o en general de un estudiante que esté adquiriendo conocimientos sobre el campo de los robots manipuladores y deba realizar prácticas con éstos.

##### **3.1.3 Objetivos**

Como objetivo global que se pretende alcanzar es que el alumno, conozca uno de los tipos de programación que puede utilizarse para programar un robot manipulador, y finalmente que realicen un programa sobre el robot para realizar una determinada secuencia de operaciones del tipo PPO (del inglés *Pick & Place Operation*), es decir, operaciones en las que el robot debe coger piezas y trasladarlas a otro punto del espacio de trabajo.

Los objetivos generales que se deben cumplir son:

- Diseño de una interfaz amigable para el usuario.
- Las prácticas propuestas a realizar deberán ser realizables en el tiempo asignado a una sesión de prácticas.
- El proyecto será ampliable, por esto el diseño del mismo estará enfocado a facilitar el inicio de las futuras líneas de proyectos.
- Establecer unas medidas de seguridad que cumplan las normativas de usos de robots.

- La interfaz web deberá estar habilitada para la visualización de las imágenes del robot a través de la webcam.

Para ello es necesario desarrollar los siguientes puntos.

- Montaje y puesta en marcha del robot manipulador y su entorno de trabajo.
- Diseño de las prácticas de PPO
- Analizar la interfaz de control suministrada con el robot
- Creación del secuenciador de operaciones
- Creación del gestor de archivos (Colas y verificación)
- Creación de una aplicación de verificación del código del programa.
- Realización de las prácticas de PPO
- Creación de la aplicación web para las prácticas online
- Configuración de la aplicación de transmisión de video.
- Pruebas y test
- Desarrollo del manual de usuario.
- Documentación del proyecto.

#### **3.1.4 Fuentes de información**

Como fuentes de información documental de las que se dispondrán, se tienen los siguientes libros: Robots y Sistemas Sensoriales.

– F. Torres, J. Pomares, P. Gil, S.T. Puente, R. Aracil. Prentice-Hall, 2002.

Introduction to Robotics: Mechanics and Control.

– J.J. Craig, Addison-Wesley, 1989.

Robótica: Manipuladores y Robots Móviles.

– A. Ollero, Marcombo, 2001.

Sistemas de Control no-Lineal y Robótica.

– V. Etxebarria, Editorial UPV-EHU, 1999.

Fundamentos de Robótica.

– A. Barrientos, L.F. Peñín, C. Balaguer, R. Aracil, Mc-Graw-Hill, 1997.

Dichos libros son parte de la bibliografía de la asignatura Robótica y Automatización Industrial, además se dispondrá de los propios apuntes de la asignatura, así como manuales de programación y toda la información disponible en Internet.

En relación a las fuentes de información no documental, se puede contar con las consultas a profesores de los diversos ámbitos que abarca este proyecto (robótica, programación, redes y comunicaciones, diseño web, etc.) i los foros de ayuda en internet.

### 3.2 SISTEMA A REALIZAR

#### 3.2.1 Descripción

Se quiere realizar el montaje de un robot manipulador, que será la base para la realización de las prácticas de robótica, así como el diseño de un guión de prácticas, de una interfaz de programación de robots y de una aplicación web para la realización tele-presencial de las prácticas, para tener una visión general del sistema se adjunta el siguiente diagrama de bloques (Ver Fig. 10).

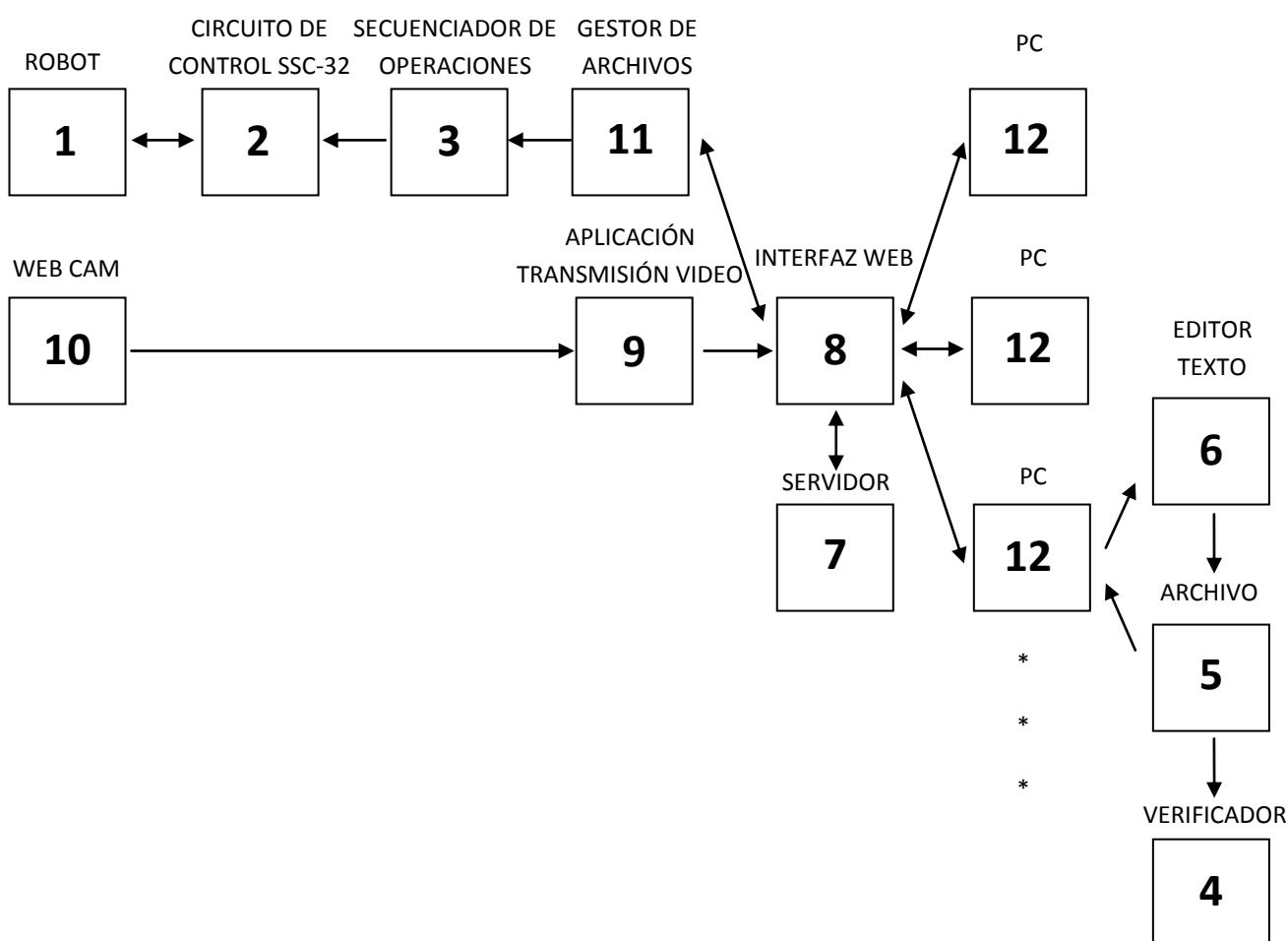


Fig. 10 – Diagrama de bloques del sistema

Descripción de los módulos:

1. **ROBOT.** Será el sistema mecánico que realizará las acciones, y estará compuesto por elementos rígidos y móviles, así como servo-motores como actuadores de tipo eléctrico.
2. **Circuito de control SSC-32.** Es el circuito electrónico que se encargará de controlar los servo-motores además de establecer la comunicación con el PC a través del puerto serie, este circuito está basado en el micro-controlador Atmel ATMEGA168-20PU
3. **Secuenciador de operaciones.** Este módulo consistirá en una aplicación desarrollada en lenguaje C, que tendrá como funciones las de leer un archivo de texto que contendrá el programa realizado por los alumnos y traducir cada instrucción al conjunto de instrucciones que es capaz de interpretar el circuito SSC-32, además el módulo determinará en que momentos debe enviar las instrucciones al circuito, mediante el protocolo de comunicación serie RS232.
4. **Verificador.** Este módulo consistirá en una aplicación desarrollada en lenguaje C que tendrá la función de analizar el archivo de texto que le introduzca el alumno e indicará a través de un mensaje en la pantalla si el programa que contiene el archivo es sintácticamente correcto, para poder ser enviado al servidor y posteriormente ejecutado sobre el robot.
5. **Archivo.** Este archivo será generado al guardar el texto editado por el editor de texto y contendrá el programa de control realizado por el alumno.
6. **Editor de texto.** Este modulo representa cualquier aplicación que permita la edición de texto para la realización del programa de control a realizar en la sesión de prácticas.
7. **Servidor.** Será el ordenador donde estará alojada la página web, y donde estarán conectados la webcam y el robot, con sus aplicaciones correspondientes.
8. **Aplicación Web.** Este módulo consistirá en una página web programada en lenguaje php (del inglés *Hypertext Preprocessor*), que utilizará además *Javascript*. Su función será la de permitir la autenticación del alumno mediante un sistema de acceso basado en usuario y contraseña, permitiendo el envío de archivos de texto al servidor además de la visualización de las acciones realizadas por el robot, mediante el video enviado por la aplicación de transmisión de video, e indicando que alumno es el que está ejecutando su programa en ese momento además de una cola de alumnos en espera.

9. **Web Cam.** Será el dispositivo físico que servirá para la captura de video. Los requisitos de este dispositivo son que tenga conexión del tipo USB (del inglés *Universal Serial Bus*) y una resolución mínima de 320x240 pixels.
10. **Aplicación de transmisión de video.** Este módulo consistirá en una aplicación desarrollada en lenguaje *JavaScript*, que será capaz de leer el video captado por la web cam y transmitirlo a través de la página web.
11. **Gestor de archivos (Colas y verificación).** Este módulo consistirá en una aplicación desarrollada en lenguaje C, que permanecerá a la escucha de un puerto de comunicaciones a la espera de recibir un archivo desde internet, al momento de recibir el archivo lo analizará y determinará si es correcto o no, si no es correcto ignorará el archivo y enviará el mensaje correspondiente a la aplicación web, en caso de que se correcto colocará el archivo en una cola de archivos a la espera de ser enviados al módulo 3, e irá informando a la aplicación web que archivo es el que se está ejecutando en cada momento.
12. **PC.** Serán los ordenadores remotos desde los cuales se conectan los alumnos a la página web.

Para generalizar aún más se pueden agrupar estos módulos en dos súper módulos, el módulo offline y el módulo online.

Dentro del módulo offline se encontrarán los módulos: 1, 2, 3, 4, 5, 6 y 11.

Dentro del módulo online se encontrarán los módulos: 7, 8, 9, 10 y 12.

### **3.2.2 Modelo de desarrollo**

El modelo que se hará servir será un modelo evolutivo, ya que aunque el proyecto se puede dividir en unos bloques bien definidos, no tienen porque distribuirse de forma lineal, pudiendo realizar algunos de ellos en paralelo, y al finalizar cada bloque se realizará un análisis, en el cual posiblemente se hagan modificaciones que obligaran a tener que volver a revisar lo que ya se ha hecho, siendo así un proceso iterativo de depurado.

### 3.2.3 Recursos

Los recursos necesarios para el desarrollo de este proyecto son los siguientes:

- Tiempo del programador
- Tiempo del técnico en robótica
- Tiempo del director de proyecto
- Componentes del robot
- Material para el entorno de trabajo del robot
- Herramientas para el montaje del robot
- PC AMD Athlon XP 1700+ con 256 MB de RAM (del inglés *Random Acces Memory*)
- Software (licencia Windows XP)
- Webcam NGS SpinCam
- Monitor 17"
- Conexión a internet ADSL (Asynchronous Digital Subscriber Line)
- Entorno de programación (Compilador C)

### 3.2.4 Viabilidad técnica

Previamente se han introducido (en el estudio teórico), los conceptos necesarios para la comprensión de este apartado.

#### 3.2.4.1 Robot

El robot escogido para este proyecto es un robot manipulador de configuración antropomórfica de brazo articulado con 6 grados de libertad, con actuadores de tipo eléctrico (servomotores) y un circuito de control de servos con comunicaciones vía puerto serie RS323.

El motivo de la elección de este tipo de robot manipulador con dicha configuración, en lugar de escoger otra, como podrían ser los robots SCARA, cartesiano, cilíndrico, etc... es que es la configuración más común dentro del sector de la robótica industrial, siendo la que se estudia con más profundidad en la asignatura Robótica y Automatización Industrial, además que la aplicación del proyecto está enfocada a las prácticas de dicha asignatura que se centra en el ámbito industrial. Otro motivo es que éste tipo de configuración da un espacio de trabajo más amplio que otras.

En cuanto a la elección de que el robot fuera el modelo lynx6 de la página web de lynxmotion, viene dado puesto que es un robot de bajo coste, con fines didácticos pero con las características básicas suficientes para poder comprender el comportamiento de los robots manipuladores, en sustitución de un robot manipulador industrial que tiene prestaciones superiores pero que tiene un coste muy elevado (a partir de las decenas de miles de euros) que no se amortizaría para la aplicación que se le daría.



Al elegir el tipo de robot este ya era suministrado con sus actuadores y circuito de control, por lo tanto no es una elección que se haya realizado, pero el motivo que se suministre con los actuadores de tipo eléctrico, es que permiten un control mucho más preciso que los neumáticos u oleohidráulicos, por el contrario pueden soportar menos carga que los oleohidráulicos, pero puesto que la aplicación que se le va a dar no es industrial, ese no es un factor determinante. En cuanto al circuito de control, es apropiado debido a que está preparado para establecer comunicaciones con un PC a través del puerto serie RS232 y además permite el control de hasta 32 servos, es más que suficiente puesto que el robot tan solo dispone de 6 servos.

### 3.2.4.2 Lenguajes de programación

Referente a los lenguajes de programación dentro del proyecto, podemos distinguir dos grupos: lenguajes para el desarrollo de aplicaciones y lenguajes para desarrollo web.

Dentro del primer grupo se podían utilizar multitud de lenguajes, C, C++, Delphi, Java, etc., la elección del tipo de lenguaje fue más por motivos de conocimientos que de prestaciones. El lenguaje escogido fue C, ya que se conocía su estructura, se había utilizado para hacer comunicaciones con el puerto serie, además de otras aplicaciones.

Dentro del segundo grupo los lenguajes escogidos fueron PHP, HTML (del inglés *HyperText Markup Language*) y *Javascript* las razones son las mismas que para el grupo anterior, son los lenguajes conocidos que habían sido usados con anterioridad y con los que se sabía que podían implementarse la funcionalidad remota del proyecto.

### 4.2.4.2 Software

En temas de software se intentará en todo momento, al menos en las partes funcionales del proyecto, utilizar software de código abierto o licencia libre, puesto que al ser un proyecto que pretende tener un uso didáctico y de divulgación de conocimiento, no sería adecuado que estuviera limitado por licencias de uso privado.

Los grupos en los que se ha utilizado software son los siguientes: programación de aplicaciones, programación de la web, servidor web, servidor de video, redacción de documentación y diseño y retoque de imágenes

- **Programación de aplicaciones:** El software escogido es un compilador de C y C++ llamado *MingW developer Studio*. El motivo es por compilar en lenguaje C que es el que se utiliza para esta tarea y además es de código abierto.
- **Programación Web:** En este caso hay diversos software utilizados en diversas etapas del desarrollo, el primero es NVU para crear la estructura web de forma gráfica, el segundo es Aptana Studio 1.2 para la programación html y el tercero y último es

Notepad++ 5.2 para la programación en PHP y *Javascript*. Todos ellos han sido escogidos por adaptarse al lenguaje de programación utilizado y ser de código abierto.

- **Servidor web:** El servidor escogido es *appserv*, por ser un software que instala sencillamente un servidor apache con php y mysql, lo que permite trasladar el servidor de forma rápida a cualquier ordenador con sistema operativo Windows, además también es software libre.
- **Servidor de video:** Este software se llama DCam Server se encarga de realizar la captura de las imágenes de la webcam, además tiene la opción de subir la imagen a un servidor de internet (en este caso no se utiliza puesto que se instala en el propio servidor), el motivo de su elección es el de ser software libre además de ser el software con el intervalo de captura más pequeño que se ha encontrado (1 segundo).
- **Redacción y documentación:** Se ha utilizado Microsoft Word 2007. En este caso no es libre, pero no es parte funcional del proyecto.
- **Diseño y retoque de imágenes:** Se ha utilizado Paint de Windows y Adobe Photoshop CS, los cuales tampoco son de licencia libre, pero no son parte funcional del proyecto.

Tras definir que tecnologías se utilizaran dentro del proyecto y comprobar que son compatibles entre sí, en este punto puede afirmarse que el proyecto es técnicamente viable.

#### 4.2.5 Análisis coste

En principio no se prevé obtener ningún beneficio económico con este proyecto, ya que lo que se pretende es mejorar la calidad de las prácticas de robótica, dentro de la UAB, pero podría llegar a comercializarse como un “Laboratorio de robótica tele-presencial”, y como potenciales clientes se podría contar con universidades, institutos y centros docentes en los que abordarían el campo de la robótica y precisarían de la realización de prácticas para la mejor explicación de la materia y/o formación de los alumnos.

Recurso	Coste (h)
Director de proyecto	35,00 €
Profesor de Ingeniería de Sistemas	25,00 €
Analista	20,00 €
Técnico en robótica	18,00 €
Becario programación	7,00 €

Tarea	Trabajo (h)	Coste
Estudio de viabilidad (Analista 100%, Director Proyecto 25%, Técnico en robótica 25%, Técnico en programación 25%)	10	254,00 €

Comprar el material para el montaje (100% Director Proyecto)	2	70,00 €
Montaje y puesta en marcha del robot manipulador. (Director Proyecto 10%, Técnico en robótica 100%)	15	322,50 €
Montaje y puesta en marcha del entorno de trabajo. (Director Proyecto 10%, Técnico en robótica 100%)	10	215,00 €
Pruebas y test fase montaje (Técnico en Robótica 100%)	4	72,00 €
Analizar la interfaz de programación suministrada (Becario de Programación 100%)	20	140,00 €
Creación del secuenciador de operaciones (Becario de Programación 100%)	60	420,00 €
Creación del gestor de archivos (Colas y verificación) (Becario de Programación 100%)	40	280,00 €
Creación del verificador del código del programa (Becario de programación 100%)	40	280,00 €
Pruebas y test del módulo offline (Director de proyecto 10%, Becario de programación 100%)	10	105,00 €
Instalación del servidor (Profesor de Ingeniería de Sistemas 100%)	10	250,00 €
Asignación y configuración del dominio (Profesor de Ingeniería de Sistemas 100%)	5	125,00 €
Creación de la interfaz web (Becario de programación 100%)	70	490,00 €
Creación de la aplicación de transmisión de video (Becario de programación 100%)	20	140,00 €
Pruebas y test del modulo online (Director de proyecto 10%, Becario de programación 100%)	10	105,00 €
Desarrollo de manuales (Técnico en robótica 75%, Becario de Programación 100%)	9	184,50 €
Documentación del proyecto (Director de proyecto 100%)	15	525,00 €
<b>TOTAL</b>	<b>370</b>	<b>3.978,00 €</b>

Total costes:

<b>Recurso</b>	<b>Coste Total</b>	<b>Coste Mensual</b>
Componentes del Robot	395,45 €	
Material y entorno de trabajo del robot	100,00 €	
Herramientas para el montaje del robot	10,00 €	
PC Intel DC 2Ghz, 2Gb RAM, 160 Gb Disco duro	240,00 €	
Software (licencia Windows XP)	85,00 €	
Webcam Conceptronic	15,00 €	

Monitor 17"	120,00 €	
Conexión de banda ancha a internet (ADSL) x 4 meses		30,00 €
Desarrollo del Proyecto	3.978,00 €	
<b>TOTAL</b>	<b>4.943,45 €</b>	<b>120,00 €</b>

En este apartado tan solo se han descrito los costes del proyecto. Sobre los beneficios que se le suponen en función de que uso final se le dé al proyecto, se detalla en el siguiente apartado.

#### 4.2.6 Presupuesto

El siguiente presupuesto sería el que se entregaría a los posibles compradores del laboratorio de robótica tele-presencial, en el caso de que se comercializara el producto. En dicho presupuesto podría excluirse el PC y la webcam si el cliente ya dispusiera de uno, con lo que la oferta sería más atractiva ya que reduciría el importe.

Presupuesto:	
Presupuesto para el laboratorio de robótica tele-presencial.	
Descripción detallada:	
Laboratorio de robótica tele-presencial incluye:	1.526,85 €
<ul style="list-style-type: none"> <li>- Robot Manipulador</li> <li>- Entorno de trabajo del robot</li> <li>- Software de programación y control remoto</li> <li>- Guión de prácticas para alumnos</li> <li>- Manual de usuario</li> </ul>	
PC Intel DC 2Ghz, 2Gb RAM, 160 Gb Disco duro + WebCam	412,00 €
<b>TOTAL</b>	<b>1.938,85 €</b>

Plazo de entrega	2 semanas después de la firma
Condiciones de pago:	
<ul style="list-style-type: none"> <li>- A la aceptación del presupuesto</li> </ul>	40% 60%

- A la entrega	
<p>Todos los precios son sin IVA.</p> <p>El laboratorio puede ser suministrado sin el PC.</p> <p>La oferta tiene una vigencia de 30 días.</p>	

#### 4.2.7 Viabilidad económica

Se conoce que los costes en material hardware ascienden a 965,45 € y el coste en recursos humanos es de 3.978 €. El departamento de ingeniería de sistemas de la ETSE, dispone de un presupuesto anual entorno a los 1.000€ y será quien asumirá los costes de material, también es posible que en lugar de comprar un PC nuevo y una webcam se utilice algunos de los que ya dispone el departamento. El desarrollo del proyecto será llevado a cabo por un alumno de informática de sistemas, que debe realizar su proyecto de final de carrera y que asumirá todos los roles presentados anteriormente (técnico programador, técnico en robótica, etc...), por este motivo los costes de recursos humanos no llegaran a materializarse como coste, asumiendo dicho alumno las horas de desarrollo del proyecto, como las horas necesarias para la obtención de los créditos correspondientes al proyecto de final de carrera, referente al coste del recurso director de proyecto éste será asumido por la UAB, que es la entidad paga el sueldo al profesorado y tiene previsto que ellos tengan que dedicar un número de horas en cada curso a la dirección de proyectos.

Se ha planteado obtener un beneficio del 10% sobre la venta del PC + webcam y un beneficio del 30 % sobre el laboratorio de robótica tele-presencial en caso de comercializarse. Para calcular el precio del laboratorio, solo se ha tenido en cuenta el coste de la puesta en marcha, del robot y del entorno de trabajo, ya que el resto de costes del proyecto solo se tendrán en cuenta para el prototipo, ya que en la realización de un nuevo laboratorio tan solo habría que comprar el robot y el material del entorno de trabajo además de realizar la puesta en marcha. El software, manuales, estudios y demás ya estarán hechos.

Se ha calculado que para amortizar el proyecto sería necesario vender 15 laboratorios en el caso de que no se vendiera con el PC + webcam, si todos los clientes pidieran el equipo completo se amortizaría con 13 ventas.

Puesto que los costes del proyecto son asumibles y además existe la posibilidad de obtener beneficios comercializando el producto, puede decirse que el proyecto es viable económicamente.

#### 4.2.8 Evaluación de riesgos

Los riesgos que se pueden encontrar en este proyecto son los siguientes.

- **Conectividad.** El laboratorio puede funcionar de modo offline, pero debido a que tiene una funcionalidad que le permite operar de modo online, existe el riesgo que por problemas de la red los usuarios que se encuentre en modo remoto no puedan acceder al laboratorio. Durante el diseño del mismo ya se observó que habría un problema en realizar una aplicación del tipo cliente-servidor ya que los usuarios podrían tener problemas con firewalls y los puertos cerrados del enrutador, además la cámara web solo podría ser utilizada por una aplicación a la vez, por esto se eligió una aplicación web, en vez de dos aplicaciones cliente-servidor.
- **Compatibilidad.** Un problema que hay que tener en cuenta es la compatibilidad con los sistemas operativos, en principio el sistema está pensado para trabajar en un entorno Windows, en concreto para las versiones Windows 98, ME, 2000 y XP, por lo que los usuarios de otros sistemas operativos, no podrían utilizarlo.
- **Tecnología.** Si no se utiliza el robot con el PC analizado en el estudio, hay que tener en cuenta los requerimientos que ha de tener el sistema para funcionar, principalmente ha de tener un puerto serie de comunicaciones para la comunicación con el robot, además de disponer de una tarjeta de red para la conectividad a Internet y un procesador mínimo Pentium IV para garantizar la fluidez del sistema.
- **Seguridad en Internet.** Al ser un sistema que permite enviar información para la ejecución de operaciones sobre el robot, y más concretamente prácticas de alumnos, hay que implementar cierta seguridad para garantizar que solo acceden las personas autorizadas al sistema, y que no se alteran los datos por el camino.
- **Formación del usuario.** Como se ha comentado anteriormente este sistema será utilizado por estudiantes de robótica, pero bajo la supervisión de su profesor, es por ello, que la responsabilidad de formar y enseñar a hacer buen uso del sistema queda en manos del profesor, quien deberá leer el manual proporcionado y aprender cómo funciona antes de su uso.
- **Fragilidad.** Básicamente es que el robot es un sistema mecánico y por tanto puede sufrir daños. Para evitar esto se establecerán las medidas de seguridad necesarias para impedir el acceso al robot además para cumplir con la normativa establecida en el uso de robots manipuladores.

#### 4.2.9 Alternativas

Después de una búsqueda intensiva se ha llegado a la conclusión, que no existe un producto similar en formato comercial, además tampoco se ha encontrado ningún referente sobre un uso privado. Se ha encontrado un símil con robots móviles, desarrollado por la universidad de granada (Enlace 4), pero este sistema tiene que ver más con la sensorización de un robot móvil, que con la planificación y secuenciación de operaciones, y los sistemas de coordenadas para representar el espacio de trabajo, que es lo que se imparte en la asignatura de Robótica y Automatización Industrial. También una propuesta por el departamento de Ciencias de la Computación de la Facultad de Economía y Administración de la Universidad Nacional de Comahue quienes para la realización de un sistema muy similar al que se pretende desarrollar aquí, de todo modos aún está en estado de propuesta por lo tanto no hay aún ningún sistema tangible parecido al que aquí se propone (Enlace 5).

Se ha observado que incluso es más fácil encontrar robots móviles comerciales, que robots manipuladores. La mayoría de robots manipuladores que se encuentran, son robots industriales de medianas y grandes dimensiones con un elevado coste, que los hace apto para un uso industrial, pero no para un uso docente.

### 4.3 PLANIFICACIÓN

Para cada día transcurrido, se ha supuesto una jornada de trabajo de 4h.

TAREA	Trabajo (h)
Estudio de viabilidad (Analista 100%, Director Proyecto 25%, Técnico en robótica 25%, Técnico en programación 25%)	10
Comprar el material para el montaje (100% Director Proyecto)	2
Montaje y puesta en marcha del robot manipulador. (Director Proyecto 10%, Técnico en robótica 100%)	15
Montaje y puesta en marcha del entorno de trabajo. (Director Proyecto 10%, Técnico en robótica 100%)	10
Pruebas y test fase montaje (Técnico en Robótica 100%)	4
Analizar la interfaz de programación suministrada (Becario de Programación 100%)	20
Creación del secuenciador de operaciones (Becario de Programación 100%)	60
Creación del gestor de archivos (Colas y verificación) (Becario de Programación 100%)	40
Creación del verificador del código del programa (Becario de programación 100%)	40
Pruebas y test del módulo offline (Director de proyecto 10%, Becario de programación 100%)	10
Instalación del servidor (Profesor de Ingeniería de Sistemas 100%)	10
Asignación y configuración del dominio (Profesor de Ingeniería de Sistemas 100%)	5
Creación de la interfaz web (Becario de programación 100%)	70
Creación de la aplicación de transmisión de video (Becario de programación 100%)	20
Pruebas y test del modulo online (Director de proyecto 10%, Becario de programación 100%)	10
Desarrollo de manuales (Técnico en robótica 75%, Becario de Programación 100%)	9
Documentación del proyecto (Director de proyecto 100%)	15
<b>TOTAL</b>	<b>350</b>

En la siguiente página se puede observar la planificación que se acaba de exponer, mediante el correspondiente diagrama de Gantt (Ver Fig. 11).



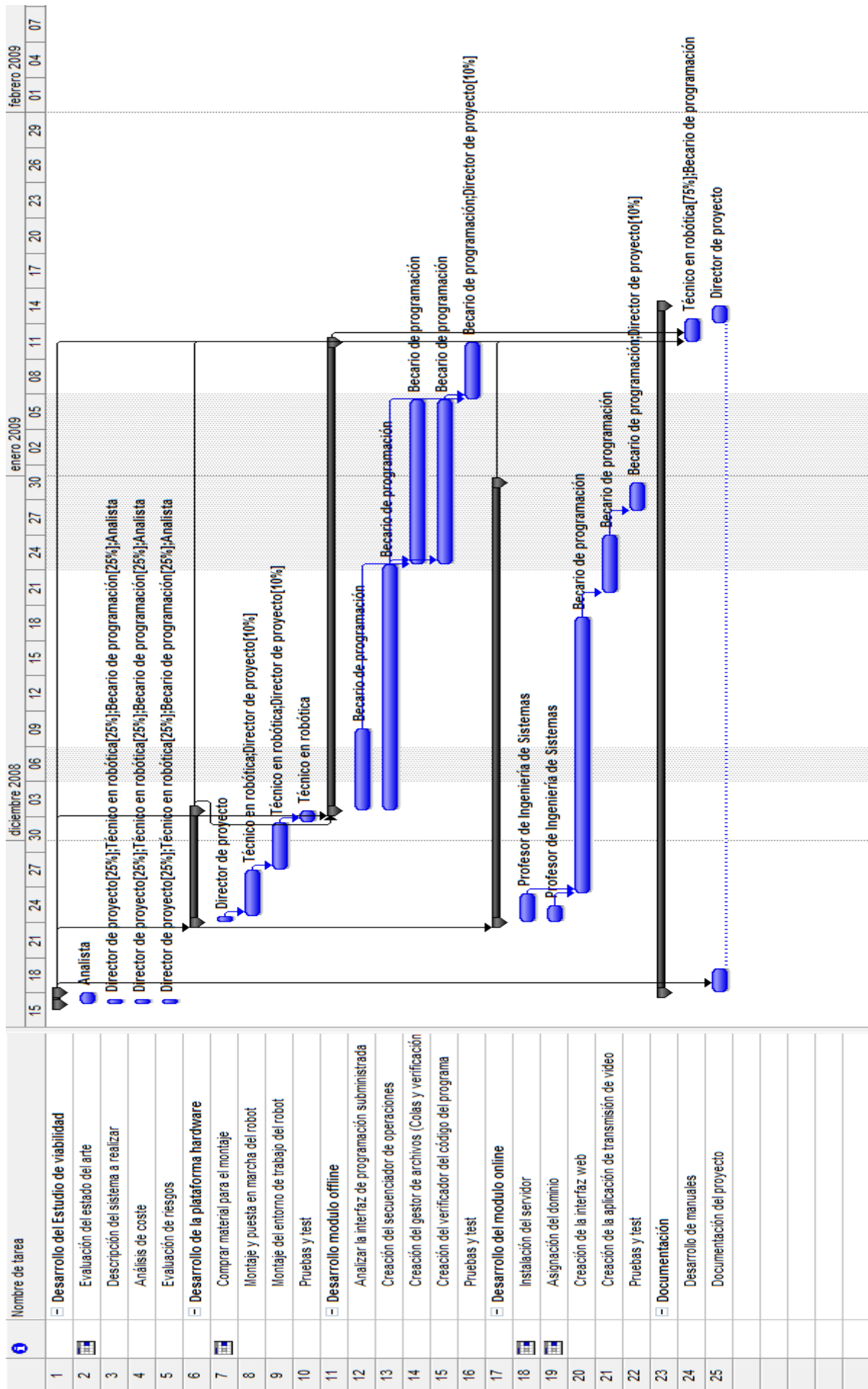


Fig. 11 – Diagrama de Gantt del proyecto

#### 4.4 **CONCLUSIONES**

Las conclusiones que se extraen de este estudio, es que existe una problemática para llevar a cabo prácticas de laboratorio sobre robots reales en la ETSE, y esto comporta que tanto profesor como alumno, se vean obligados a realizar las prácticas en un entorno simulado, en el cual tan solo se ven las ecuaciones dinámicas que rigen el comportamiento de un robot.

También se observa que de comercializar el producto, existe un hueco en el mercado para los robots manipuladores enfocados a uso docente.

El robot que se utilizará para realizar este proyecto, es de bajo coste pero con prestaciones suficientes como para realizar todas las prácticas necesarias para entender el funcionamiento de los robots manipuladores.

Después de todo lo analizado en este estudio, sabiendo que sería posible comercializar el producto, que supondrá una mejora significativa e innovación en la docencia y que es viable tanto técnica como económicamente, se puede decir que este proyecto es viable.

#### 4. DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA

En los siguientes apartados se describirá detalladamente cada módulo que conforma el sistema global, para ello se explicarán los siguientes puntos para cada módulo:

- Descripción del módulo
- Entradas y salidas
- Tecnologías utilizadas
- Tiempo de desarrollo
- Implementación
- Problemas y soluciones

Diagrama de bloques del sistema (Ver Fig. 12):

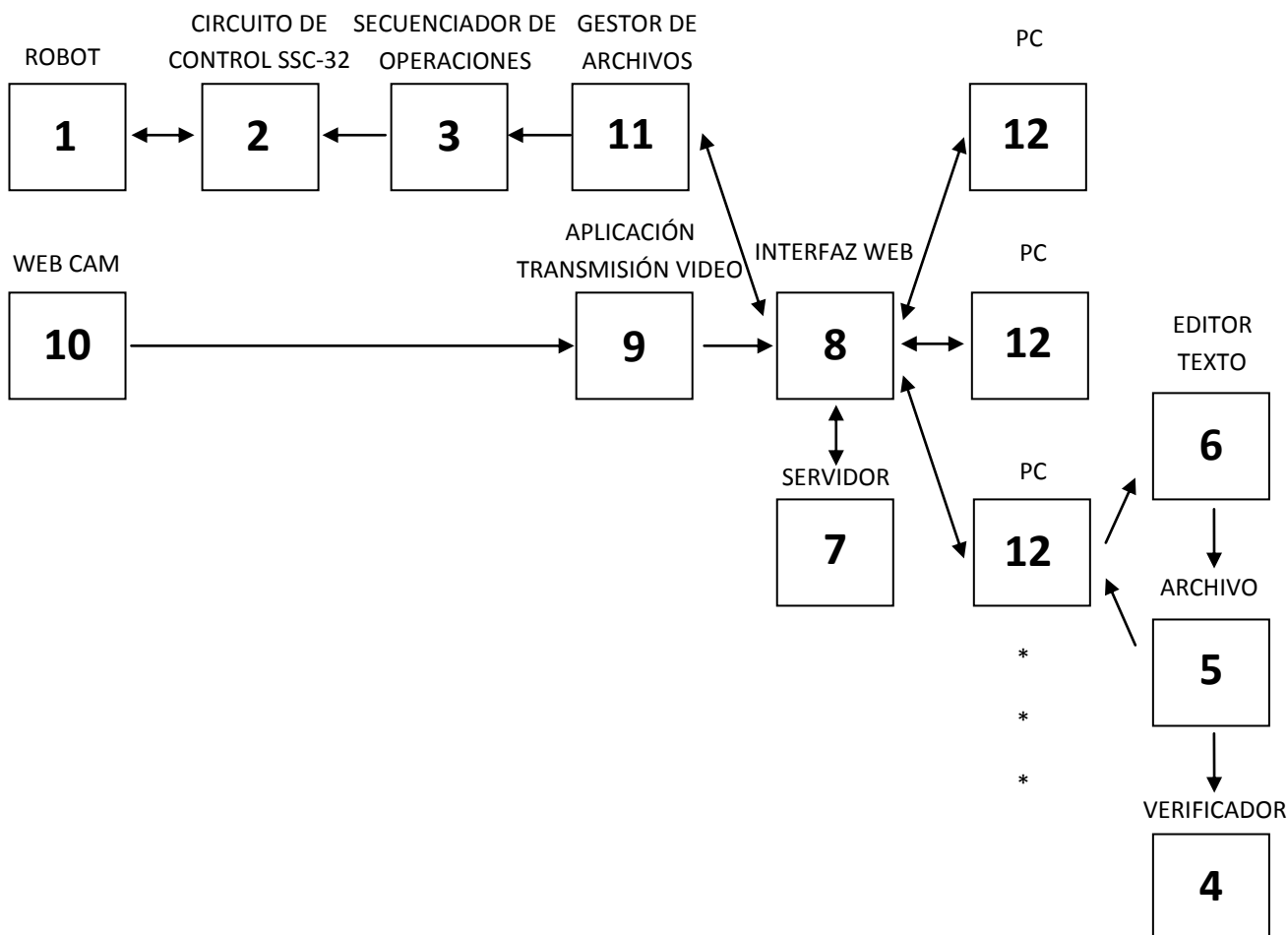


Fig. 12 - Diagrama de bloques del sistema

## 4.1 MÓDULO 1 – ROBOT

### 4.1.1 Descripción del módulo

Se trata del sistema mecánico que realiza las acciones, y está compuesto por elementos rígidos y móviles, así como servo-motores como actuadores de tipo eléctrico. A continuación se muestra un esquema del robot con los sistemas de referencia colocados para cuando se haga alusión a cualquiera de los ejes del sistema (Ver Fig. 13). La base teórica necesaria para realizar dicho esquema se adquirió en la asignatura Robótica y Automatización Industrial impartida en la ETSE.

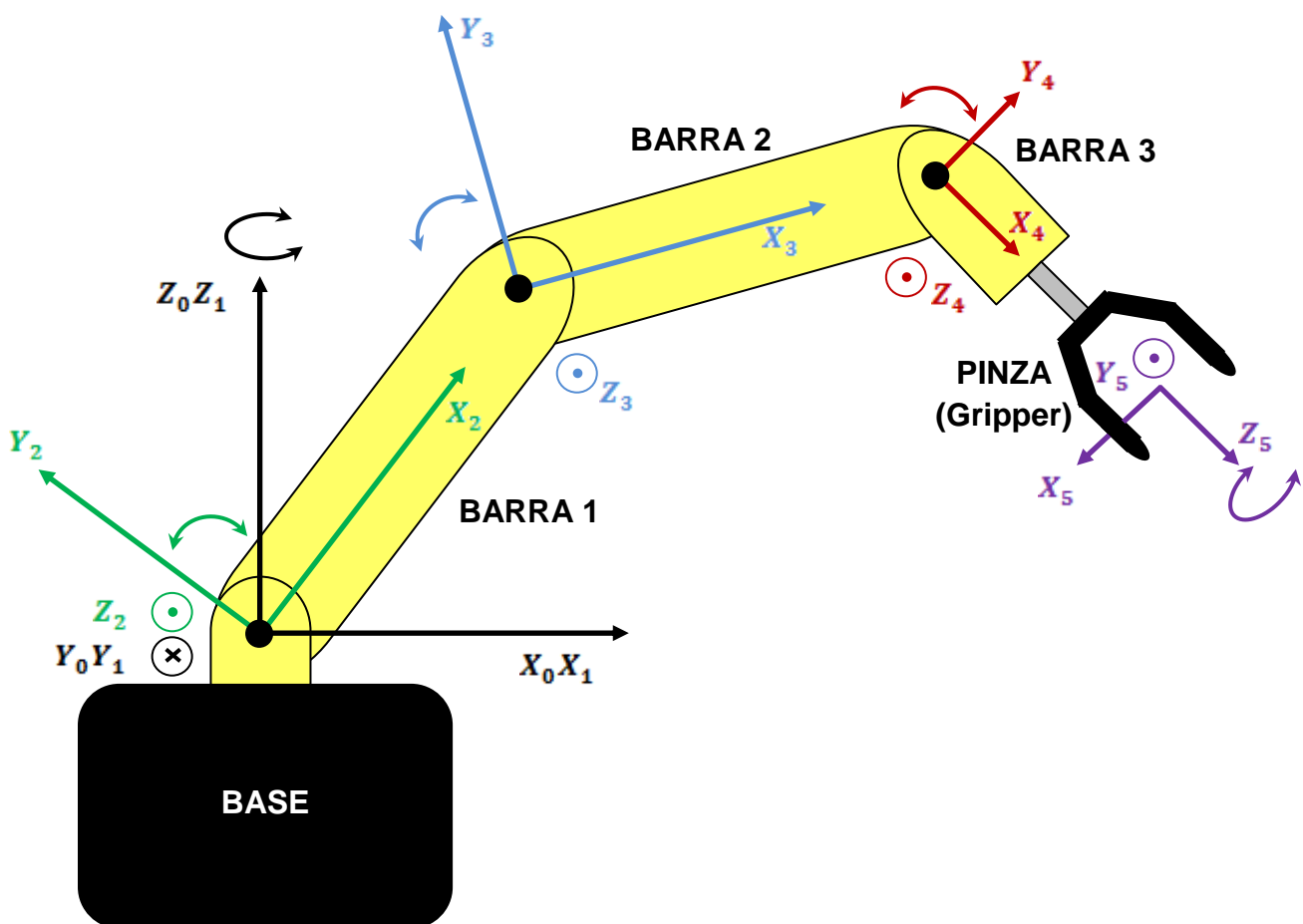


Fig. 13 – Sistemas de referencia del robot

### 4.1.2 Entradas y salidas

**Entradas:** Pulsos emitidos por los PWM (*del inglés Pulse Width Modulator*) o Modulador del Ancho de Pulso, que se encuentran en el circuito de control de servos SSC-32.

**Salidas:** Movimientos realizados por el robot.

### 4.1.3 Tecnologías utilizadas

En este módulo se ha utilizado el material proporcionado por la empresa lynxmotion para el montaje de un robot manipulador de 6 grados de libertad.

Básicamente el módulo está compuesto por un repertorio de tornillería, piezas hechas en material de plástico y servo-motores, que será el elemento que se analizará con mayor detalle.

#### 4.1.3.1 Servomotor HITEC HS475HB

El hitec HS475HB (Ver Fig. 15) es un servo de dimensiones estándar pero en que destaca 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. Se suministra con tornillos y varios platos y brazos de montaje. Conexiones: Amarillo-señal, Rojo- positivo y Negro-negativo (Ver Fig. 14).

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

El robot del proyecto dispone de 5 servomotores de este modelo, uno en la base, 2 en la primera articulación, otro en la segunda y el último en la tercera articulación que corresponde a la muñeca.

Características técnicas (Enlace 6):

- **Sistema de Control:** Control por Anchura de Pulso. 1,5ms al centro
- **Rango de giro:** 180°
- **Tensión de funcionamiento:** 4,8V a 6V
- **Velocidad a 6V:** 0,18 Seg / 60 grados sin carga
- **Par a 6V:** 5,5 Kg · cm
- **Corriente en reposo:** 7,7 mA
- **Corriente en funcionamiento:** 180 mA sin carga
- **Corriente Máxima:** 1100 mA
- **Zona Neutra:** 5 µseg
- **Rango de Trabajo:** 1100 a 1900 µseg
- **Dimensiones:** 38,8 x 19,8 x 36 mm (Ver Fig. 16 y Fig. 17)
- **Peso:** 40 g
- **Rodamiento Principal:** Metálico
- **Engranajes:** Karbonite
- **Longitud del cable:** 300 mm

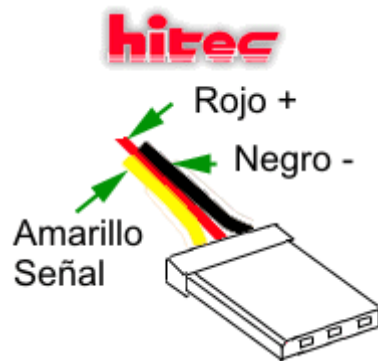


Fig. 14 – Ilustración del conector del servo



Fig. 15 – Servo HS475HB

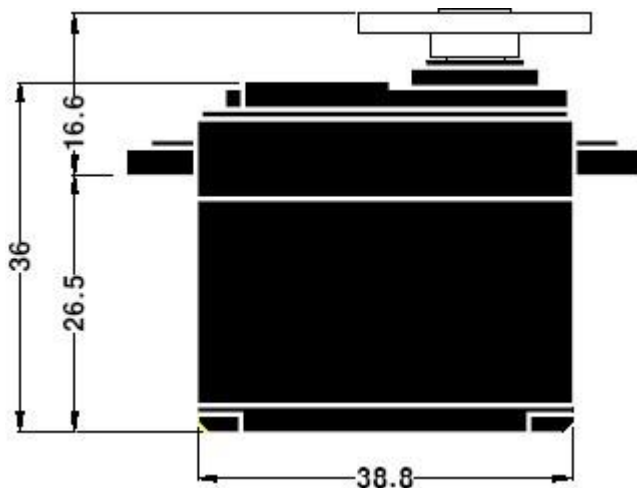


Fig. 16 – Esquema vista de perfil del servo

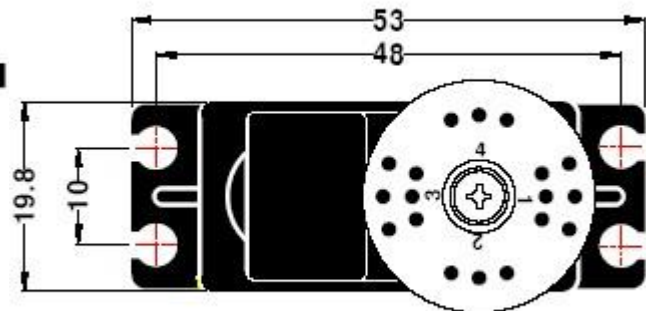


Fig. 17 – Esquema vista superior del servo

#### 4.1.3.2 Servomotor HITEC HS-85BB

El servomotor HS-85BB (Ver Fig. 18) tiene una gran potencia para ofrecer una mayor velocidad y par. Se encuentra disponible en dos versiones, fabricado en resina y metal. Este motor es idóneo si se desean obtener altas prestaciones y movimientos precisos. Los rodamientos de bolas y el eje hacen son robustos esto sumado a la precisión de movimientos que se había comentado 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  $Z_5$ .

Características técnicas (Enlace 7):

- **Sistema de Control:** Control por Anchura de Pulso. 1,5ms al centro
- **Motor:** 3 Polos de Ferrita
- **Tipo de rodamientos:** Rodamientos de bola en la parte superior
- **Par a 4,8V / 6,0V:** 3,0 · kg / 3,5 · kg
- **Velocidad 4,8V / 6,0V:** 0,16 Seg / 0,14 Seg
- **Rango de giro:** 180°
- **Dimensiones:** 29 x 13 x 30 mm
- **Peso:** 19,2 g.



Fig. 18 – Servo HS-85BB

#### 4.1.3.3 Servomotor HITEC HS81

El motor HS-81 (Ver Fig. 19) es uno de los servos más populares de Hitec. Está disponible en dos materiales: resina y metal. Con un fino equilibrio entre velocidad y par, el HS-81 está designado para ser económico pero fiable. Al utilizar el tamaño estándar Hitec *spline*, cualquier anclaje o el brazo que se ajusta a una norma Hitec, encajará en el HS81 perfectamente.

Este motor se encuentra situado en la pinza o *gripper* del robot.

Características técnicas (Enlace 8):

**Sistema de control:** Control por Anchura de Pulso. 1,5ms al centro (Ver Fig. 20)

**Pulso requerido:** 3-5 Volt de pico a pico en onda cuadrada

**Tensión de funcionamiento:** 4.8 a 6.0 V

**Rango de Temperaturas de Funcionamiento:** -20 a +60 °C

**Velocidad a 4.8V:** 0.11sec/60° sin carga

**Velocidad a 6.0V:** 0.09sec/60° sin carga

**Par a 4.8V:** 2.6kg.cm

**Par a 6.0V:** 3kg.cm

**Angulo de funcionamiento:** 45 °. *one side pulse traveling* 450usec

**Modificable a 360°:** No

**Consumo de corriente a 4.8V:** 8.8mA/reposo y 220mA en operaciones sin carga

**Consumo de corriente a 6.0V:** 9.1mA/reposo y 280mA en operaciones sin carga

**Zona Neutra:** 8usec

**Motor:** 3 Polos de Ferrita

**Control de potenciómetro:** Control Directo

**Rodamientos:** Ninguno, la parte exterior sirve como rodamiento.

**Gear Type:** Todo nylon

**Longitud del cable:** 160mm

**Dimensiones:** 29,8 x 12 x 29,6mm (Ver Fig. 21)

**Peso:** 16,6g



Fig. 19 – Servo HS-81

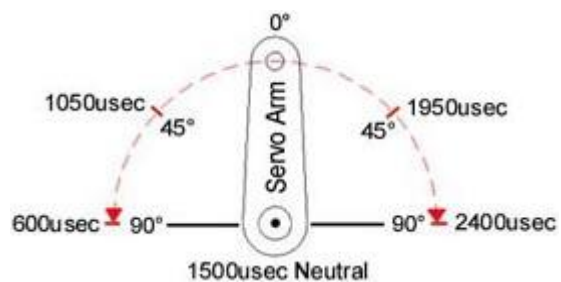


Fig. 20 – Posiciones del servo según pulso PWM

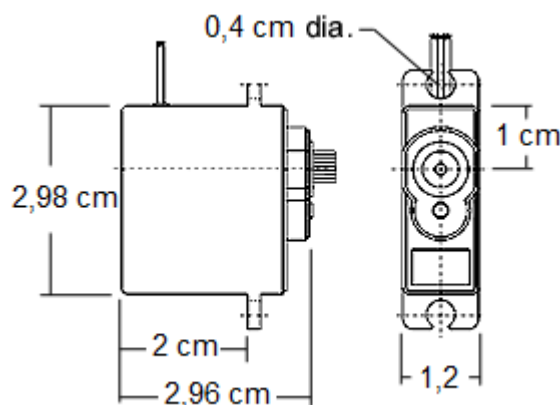


Fig. 21 – Esquemas vista superior y perfil del servo

#### 4.1.4 Tiempo de desarrollo

Para llevar a cabo el desarrollo de este módulo se han invertido un total de 23,5 horas, 11 horas para el montaje del robot y 12,5 horas para la creación del entorno de trabajo.

#### 4.1.5 Implementación

El material para el montaje del robot, se recibe en un pequeño embalaje de cartón, el contenido del cual son todos los elementos para llevar a cabo el montaje del robot, como son los componentes y herramientas (destornillador de plano y llave Allen del nº 5), todos los componentes están clasificados por bolsas y cada bolsa contiene una parte del robot, en las siguientes ilustraciones se puede ver dos de las 5 bolsas que venían en la caja (Ver Fig. 22 y Fig. 23). Cabe destacar que se tuvo que utilizar una llave de tubo hexagonal del nº 6 y un destornillador de estrella que no venían



incluidos en el paquete, para apretar mejor los tornillos, así como 10 bridas negras que tampoco se incluían.



Fig. 22 – Contenido del paquete para el montaje de la barra 2



Fig. 23 – Contenido del paquete para el montaje de la pinza

A continuación se irá explicando el proceso que se ha seguido para el ensamblaje de los componentes del robot hasta que éste estaba completamente montado. El montaje del robot se divide en 4 bloques:

- Montaje de la base y el soporte para el SSC-32
- Montaje de la barra 1 y barra 2
- Montaje de la pinza o *gripper*
- Ensamblaje de la pinza con la barra 2

Antes de iniciar el montaje se procedió a la descarga de los manuales de montaje desde la pagina web de lynxmotion (Enlace 9), dichos manuales están adjuntos a este documento y serán referidos en los módulos pertinentes, los manuales se encuentran redactados en inglés.

Al finalizar la parte del montaje del robot, se verá un último bloque:

- Creación del entorno de trabajo del robot

#### **4.1.5.1 Montaje de la base y el soporte para el SSC-32**

Para montar este bloque se procedió al desembalaje de todos los componentes que se encontraban en la bolsa correspondiente y se comenzó a leer el manual (Anexo 1), un detalle que indicaba el manual era que los rodamientos de la base al ser de plástico tenían una pequeña rebaba que debía pulirse, el sistema que proponía el manual era girar la base de modo que los rodamientos quedaran hacia abajo, apoyar la base en el suelo y realizar un movimiento circular para que se desgastase la rebaba de los rodamientos por la fricción ejercida con el suelo. Se probó esta indicación pero se desechó por no ser eficaz y pulida. En su lugar se optó por utilizar un cúter y cortar la rebaba con cuidado.

Una vez hecho esto se procedió al montaje del primer servo modelo HS475HB, el cual proporciona al robot un movimiento circular alrededor del eje  $Z_1$ , al engrase de los rodamientos mediante lubricante 6 en 1 y al montaje del soporte para los motores de la primera articulación.

Una vez acabada la parte anterior se procedió al montaje del soporte del circuito de control de servos SSC-32 y posteriormente al ensamblaje de las dos partes anteriores fijando el conjunto sobre una madera provisional para utilizarla como base (Ver Fig. 24).



**Fig. 24 – Base y circuito ensamblados**

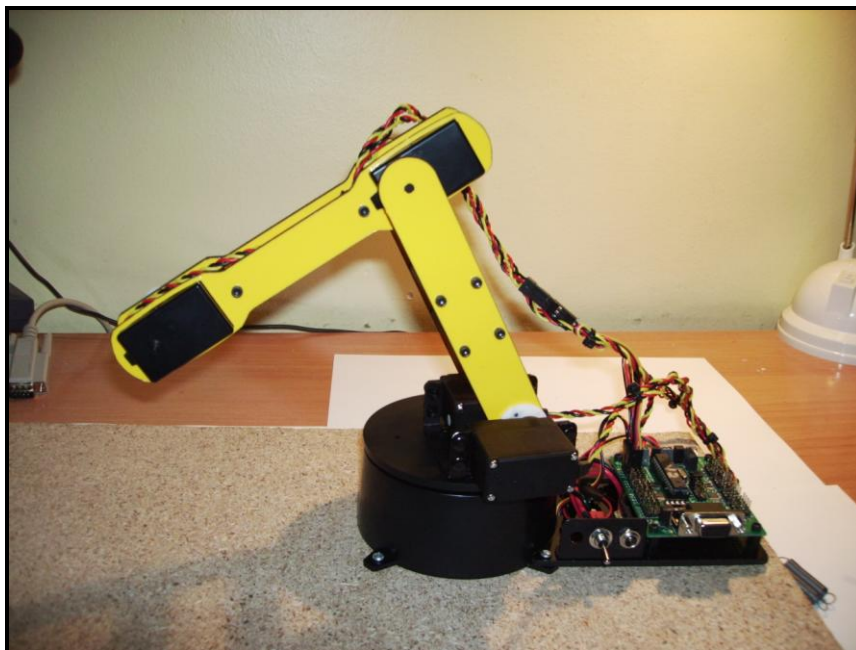
#### **4.1.5.2 Montaje de la barra 1 y barra 2**

En este bloque al igual que en el anterior, se procedió al desembalaje de todos los componentes que se encontraban en la bolsa correspondiente y se comenzó a leer el manual (Anexo 2).

Estando ya fijada la base y el circuito SSC-32 sobre la madera provisional, se montó la barra 1, y se procedió al acoplamiento de los dos servos del modelo HS475HB, que son los que otorgan la movilidad a la articulación 1, con la barra 1 y la base.

Después de tener acabada la parte anterior, se procedió al montaje de la barra 2. En esta barra se colocaron 2 servomotores del modelo HS475HB, uno correspondiente a la articulación 2 y otro a la articulación 3 (articulación de la muñeca) hay que decir en este punto, que los componentes de plástico de color amarillo que son la estructura de las barras, iban cubiertos con un plástico protector que quedo fusionado con los cantos del componente, en principio se supone que debido al corte de la pieza en la fábrica de origen, por lo tanto fue una tarea laboriosa que requirió bastante tiempo para quitar totalmente el plástico de todos los componentes de color amarillo, especialmente en los de este bloque, ya que los componentes tenían más agujeros y el plástico protector se partía por esas zonas.

Tras haber montado la barra 2, se procedió al ensamblaje de ésta con la barra 1 por medio de la articulación 2 (Ver Fig. 25), que como se ha indicado antes el servo-motor que otorga el movimiento a esta articulación es del modelo HS745HB, y rota alrededor del eje  $Z_2$ .



**Fig. 25 – Barra 2 y barra 1 ensambladas**

Por último se colocan unos muelles desde la articulación 1 a la articulación 2 para compensar el peso del brazo.

#### **4.1.5.3 Montaje de la pinza o gripper**

En este bloque de nuevo se procedió al desembalaje de todos los componentes que se encontraban en la bolsa correspondiente y se comenzó a leer el manual (Anexo 3).

En primer lugar se procedió al montaje del soporte de la pinza. En este soporte se encuentra el servomotor HS-85BB y es el que otorga un movimiento de rotación alrededor del eje  $Z_5$ , a la pinza.

Después de haber montado el soporte para la pinza se comenzó a realizar el montaje de la pinza, este proceso fue también laborioso debido a que la pinza del robot consta de múltiples barras y engranajes. En la pinza se encuentra el último servo-motor de este sistema, que corresponde al modelo HS-81 y proporciona la movilidad necesaria para abrir y cerrar la pinza. Hay que decir que en esta parte los tornillos de todas las uniones de la pinza no debían apretarse al máximo, ya que si se hacía esto el servo-motor no era capaz de abrir la pinza ya que los elementos pasaban de ser móviles a rígidos, lo que dañaría el servomotor si se pusiera en funcionamiento.

Después de montar la pinza y unirlo con el soporte, se procedió al ensamblaje de ambos, dando lugar al bloque que permite coger objetos y rotarlos (Ver Fig. 26)



**Fig. 26 – Montaje pinza finalizado**

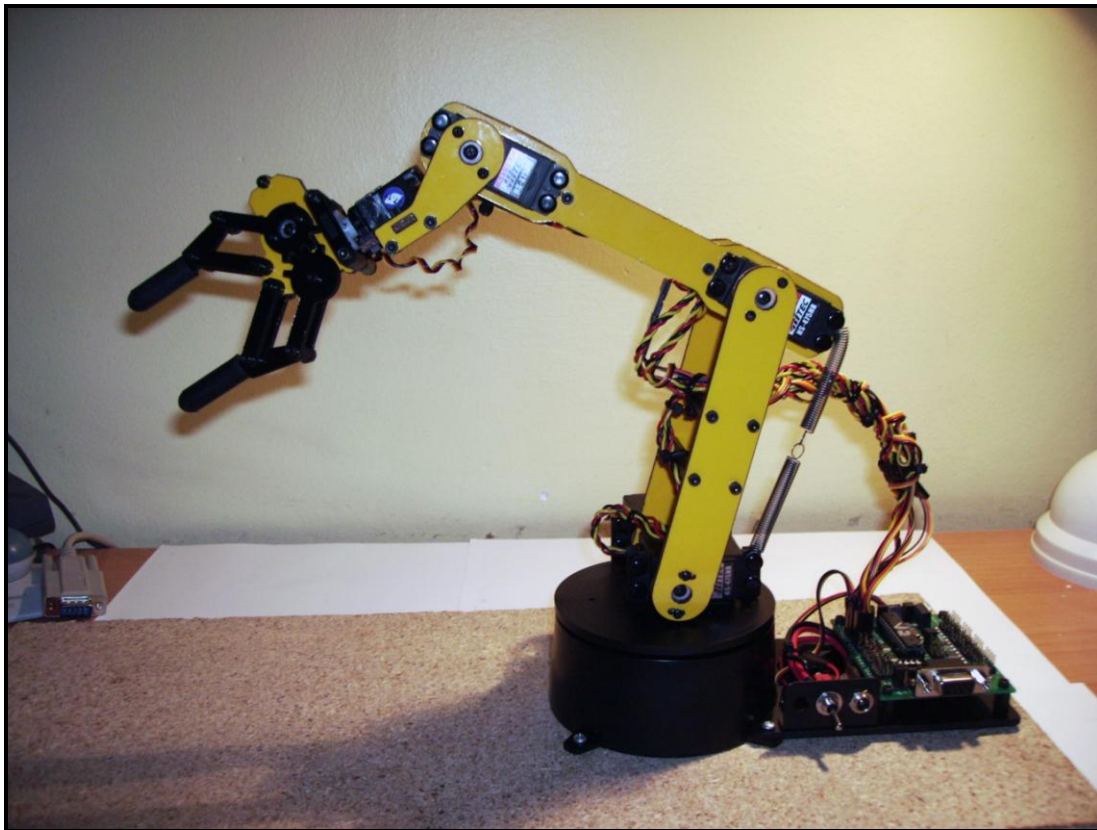
#### **4.1.5.4 Ensamblaje de la pinza con la barra 2**

Después de haber finalizado los tres bloques anteriores se tiene, por un lado la base con las barras 1 y 2 unidas y por otro lado la pinza.

En este punto se procedió al ensamblaje de la pinza con la barra 2 mediante la articulación 3, en esta articulación se encuentra un servo-motor del modelo HS-475HB y otorga a la articulación un movimiento rotacional alrededor del eje  $Z_4$ , que permite la inclinación de la pinza.

Después de ensamblar los dos bloques tan solo faltaba conectar los servos de la pinza y muñeca a los pines correspondientes del circuito SSC-32.

Una vez realizadas las tareas anteriores se finalizó el montaje del robot, y ya estaba operativo para su utilización. Resultado (Ver Fig. 27)



**Fig. 27 – Robot manipulador finalizado**

Tras el montaje se realizaron las pruebas y test pertinentes mediante el programa lynxterm de lynxmotion, para testear el correcto funcionamiento de todos los servos.

Estas son las herramientas que se utilizaron para el montaje del robot:

- Destornillador de estrella
- Destornillador plano
- Llave Allen del nº 5
- Llave de tubo hexagonal nº 6 – 7
- Cúter

#### **4.1.5.5 Creación del entorno de trabajo**

En este bloque se explicará el proceso de elaboración del entorno de trabajo sobre el que interactuará el robot.

Materia prima utilizada para este bloque:

- Madera laminada de 65x85x10 cm
- Plancha de plástico tipo PVC (Poli-cloruro de Vinilo) 100x100x0,4 cm color negro.
- Plancha de plástico tipo PVC 150x150x0,4 cm color blanco
- 3 tornillos 15mm con cabeza de estrella para enroscar en madera

La idea era crear un soporte sobre el que atornillar el robot, de manera que hubiera una zona sobre la cual pintar unos puntos característicos, que más adelante se utilizarían para la programación de tareas PPO, y que dicha zona pudiera variar, de manera que se podrían tener diferentes escenarios sobre los que el robot podría trabajar.

Para llevar a cabo esta idea, en un inicio se procedió a diseñar como debería ser la forma física del entorno de trabajo. Para ello se realizaron dos esquemas para el corte de las planchas de PVC.

El primero (Ver Fig. 28) corresponde al corte que debía realizarse sobre la plancha de PVC de color negro. (Nota: el dibujo del esquema es simétrico)

El segundo (Ver Fig. 29) corresponde al corte que debía realizarse sobre la plancha de PVC de color blanco, que sería la que no estaría fijada a la madera, de manera que se pudiese intercambiar y, de este modo tener diversos escenarios. (Nota: el dibujo del esquema es simétrico)

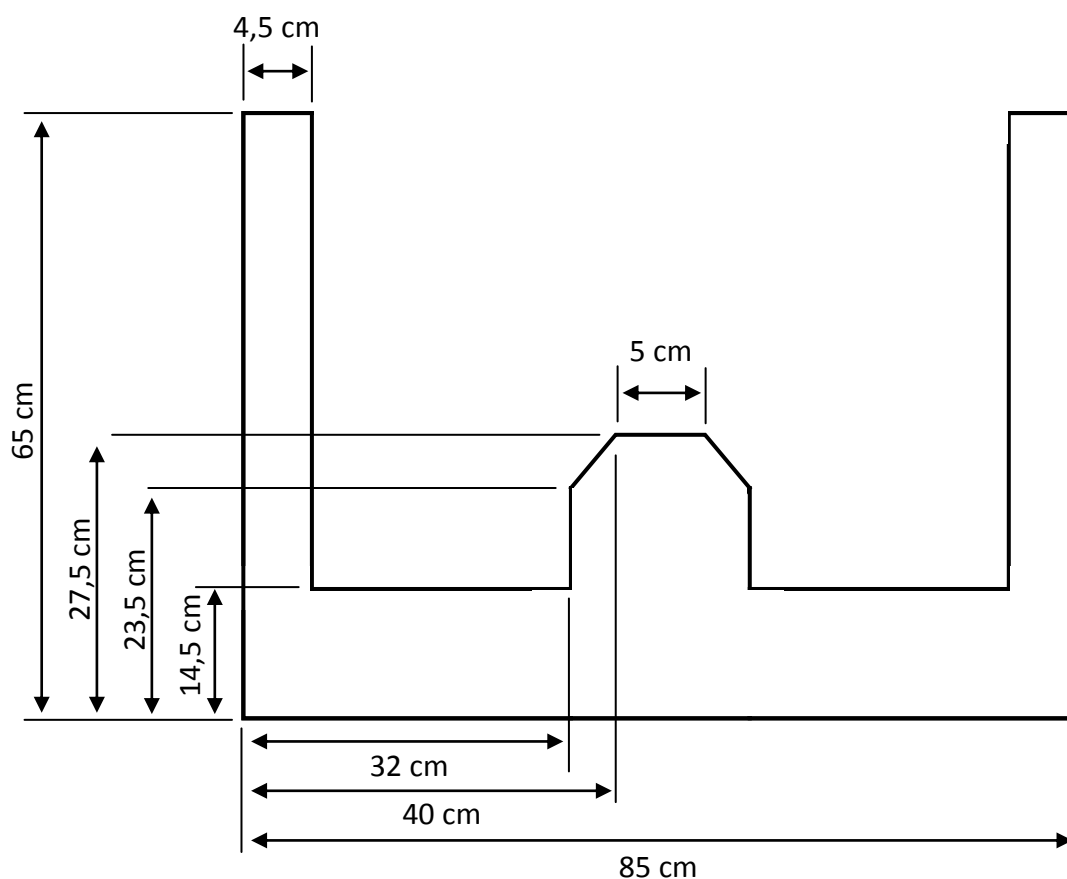
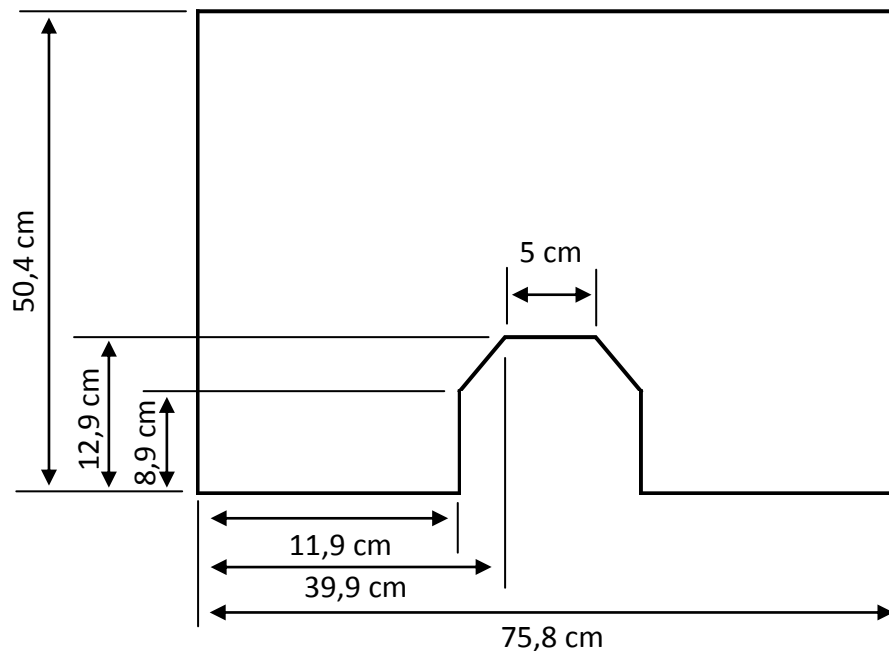


Fig. 28 – Esquema plancha PVC color negro



**Fig. 29 – Esquema plancha PVC color Blanco**

Una vez estuvieron listos los esquemas, se procedió al corte de la plancha de PVC de color negro, para realizar los cortes se utilizaron un cúter y una sierra para corte de hierro, el problema que apareció es que el arco de la sierra no era lo suficientemente grande como para cortar la plancha y llegó un punto en que el arco de la sierra tocaba el canto de la plancha y no se podía seguir cortando, así que se optó por desmontar la hoja de la sierra, enrollar un trapo en un extremo y realizar el corte cogiendo la hoja con la mano.

Otro problema que se presentó fue el no poder cortar con la sierra hasta la esquina, ya que luego no se podría realizar el siguiente corte que era perpendicular a la esquina, debido a que el espacio entre la hoja y la plancha de plástico provocado por la realización del corte no era suficiente para realizar el giro con la sierra, así que se optó por realizar un corte con forma curva para evitar llegar hasta la esquina y posteriormente cortar la parte no deseada con el cúter.

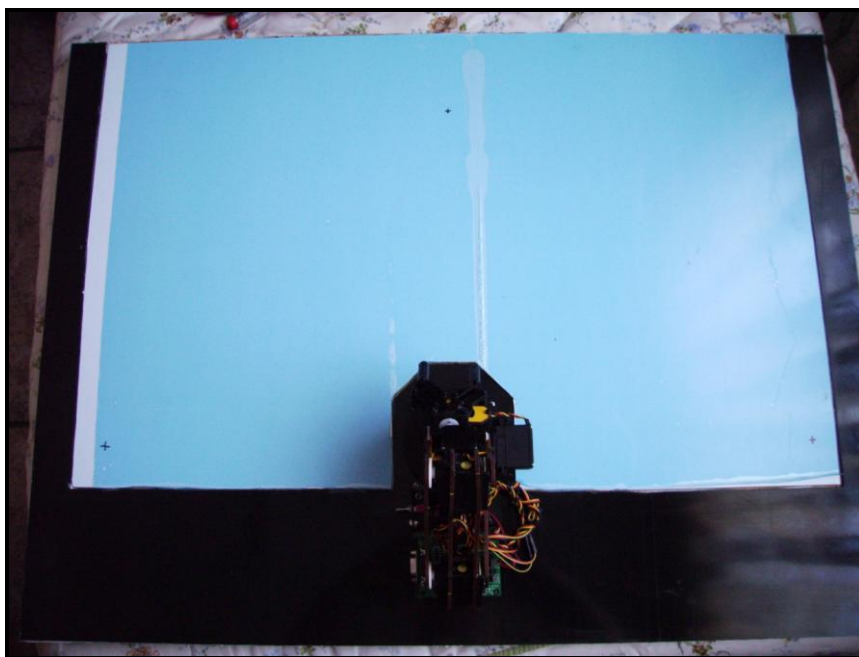
Una vez que estuvo cortada a medida la plancha negra de plástico PVC, se procedió a eliminar las rebabas que habían surgido del corte con la sierra, utilizando un papel de lija. A continuación se tomaron las medidas del esquema para la plancha blanca de PVC, y se procedió al corte mediante el uso de un cúter y un regle de madera como guía para realizar los cortes rectos. Seguidamente se comprobó si la plancha blanca encajaba dentro de la negra, como esto no sucedió así en un primer momento, se rebajó con el cúter las zonas que lo requerían.

A continuación se presentó sobre la madera la plancha negra y una vez comprobado que la plancha encajaba correctamente sobre la madera, se giró y se aplicó silicona blanca para una fuerte fijación con la madera.



Para asegurar que la fijación de la plancha negra de PVC con la madera fuese lo más perfecta posible, sin que en un lugar hubiese más cantidad de silicona que en otra, lo que podría provocar desniveles, se repartió la silicona de forma uniforme sobre la plancha utilizando un recorte de la plancha blanca a modo de espátula. Estando fijada la plancha negra de PVC, se colocó la plancha blanca para asegurar que la plancha negra no pudiera moverse, y utilizando unos listones de madera y unos sargentos, se inmovilizó la plancha negra para que la adhesión a la madera fuera correcta y permanente.

Al día siguiente se retiraron los sargentos y los listones de madera y una vez se comprobó que la fijación era correcta, se procedió a la fijación del robot mediante 4 tornillos de cabeza de estrella que venían incluidos con los componentes del robot. (Ver Fig. 30)



**Fig. 30 – Vista superior del robot fijado a la madera**

El siguiente paso fue realizar un esbozo en papel, para plasmar cuál sería la forma que se le daría al conjunto de puntos característicos que se utilizarían para programar posteriormente las tareas de PPO. Una vez realizado dicho esbozo, se digitalizó y el resultado es el que se muestra a continuación (Ver Fig. 31).

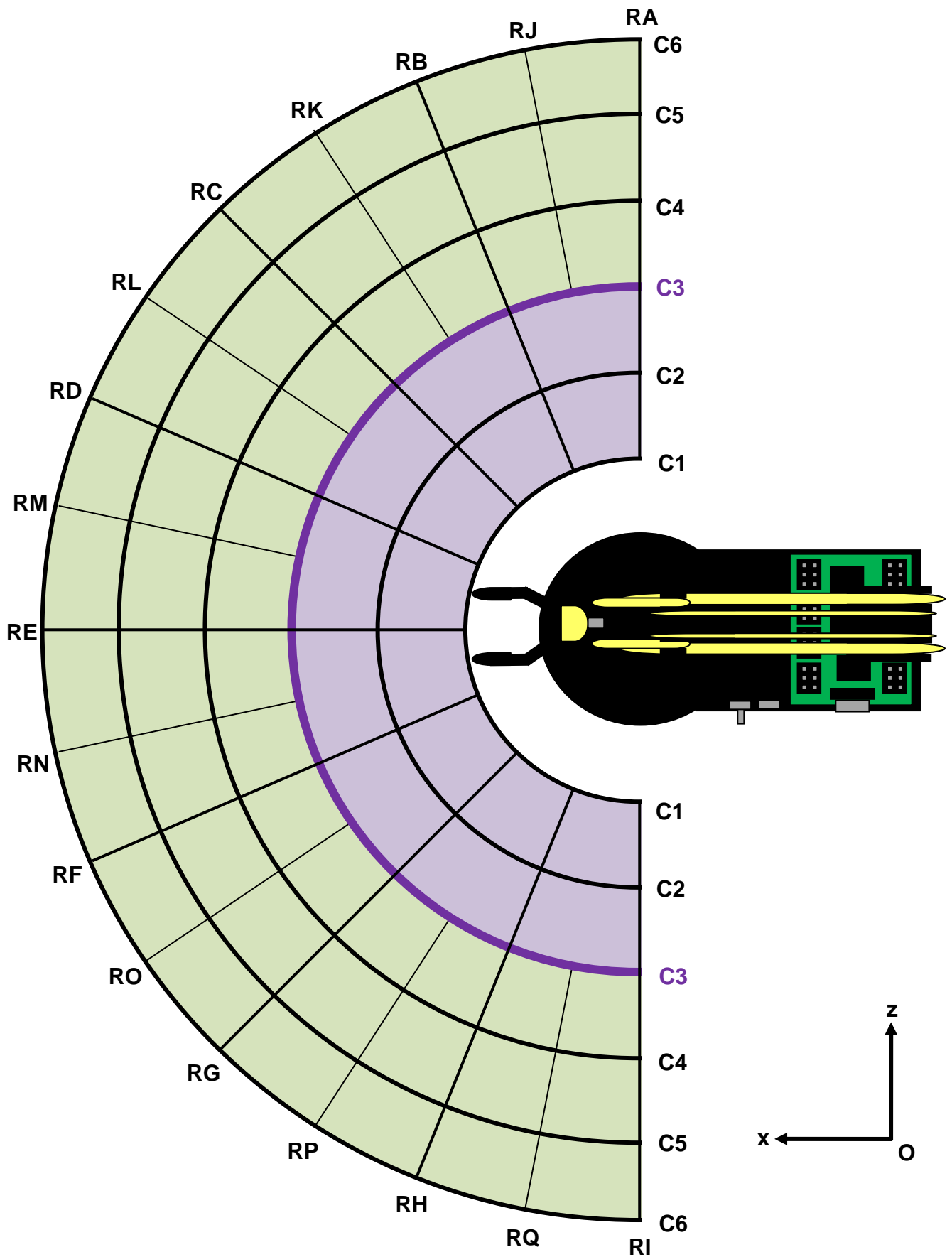
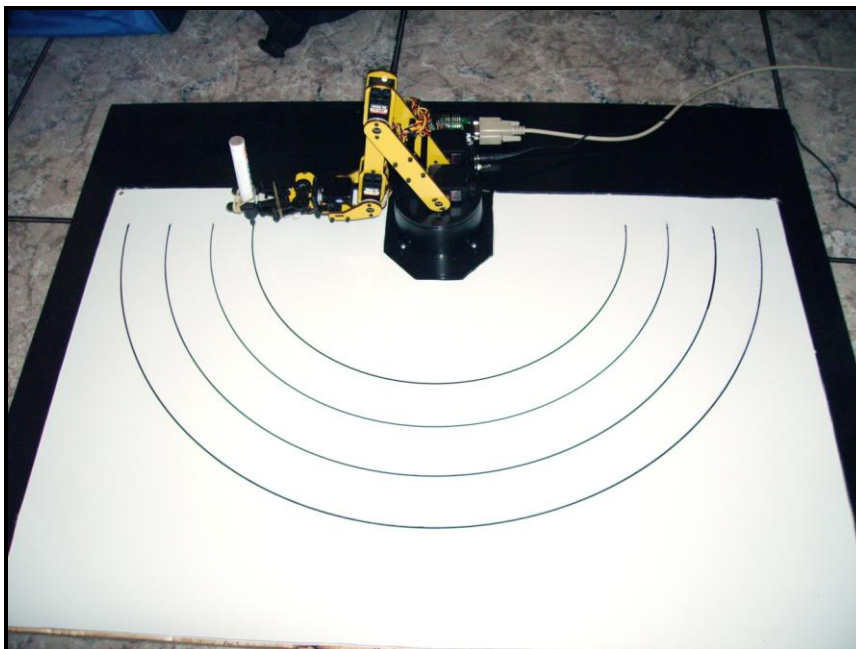


Fig. 31 – Esquema del los puntos característicos (vista superior)

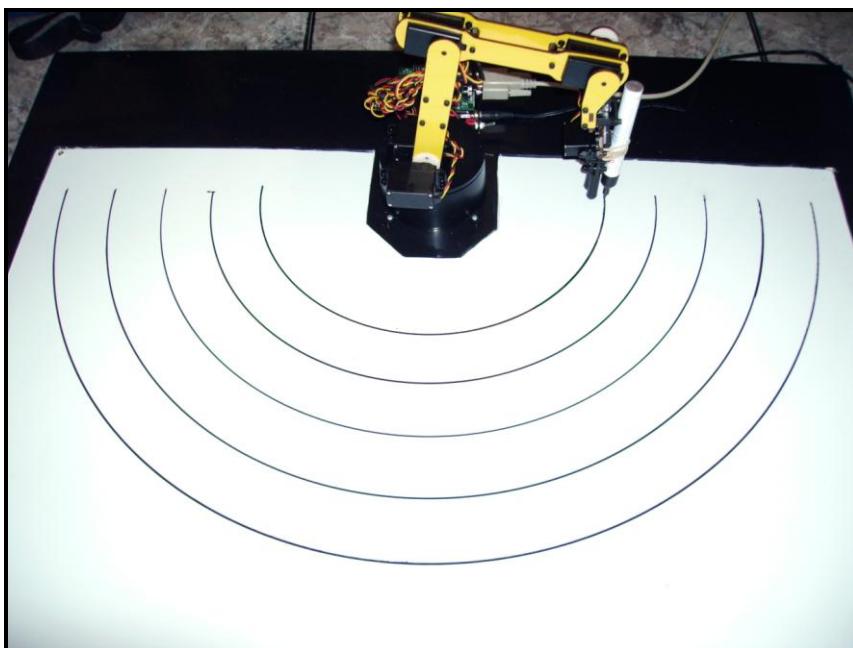
El siguiente paso era dibujar el conjunto de puntos sobre la plancha blanca de PVC, cada punto es el que definen las intersecciones entre líneas, el problema que surgió fue, que no se disponía de un compás lo suficientemente grande como para realizar las circunferencias, además debían realizarse con algún tipo de tinta permanente, por lo que se pensó en utilizar un rotulador permanente. Se planteó la idea de utilizar una cuerda y atar el rotulador en un extremo para utilizarlo a modo de compás, pero dicha idea se desechó puesto que no era un método preciso, y la situación lo requería.

Tras varias deliberaciones se pensó que quizás el robot era lo suficientemente preciso como para que fuese el mismo quien dibujara las líneas. Así que se instaló un rotulador en la pinza del robot de forma que quedara fijo, se conectó el robot al PC y se le enviaron las instrucciones necesarias para realizar los movimientos que hicieran los dibujos que se pretendían. Se observó que si era posible dibujar las semicircunferencias de forma precisa, así que se quitó el plástico protector que traía la plancha blanca de PVC para comenzar a dibujar las semicircunferencias, esta vez ya sobre la plancha blanca de PVC.

Se fueron realizando las semicircunferencias del modo en que se ve en la Fig. 55 hasta la circunferencia llamada "C3" en el diagrama de la Fig. 31, aun faltaban dos circunferencias por dibujar pero del modo en que el robot cogía el rotulador ya no podía acceder a las posiciones donde se debían dibujar las circunferencias "C1" y "C2" (Ver Fig. 32), así que se cambió la forma en que se sujetaba el rotulador para poder realizarlas. (Ver Fig. 33)



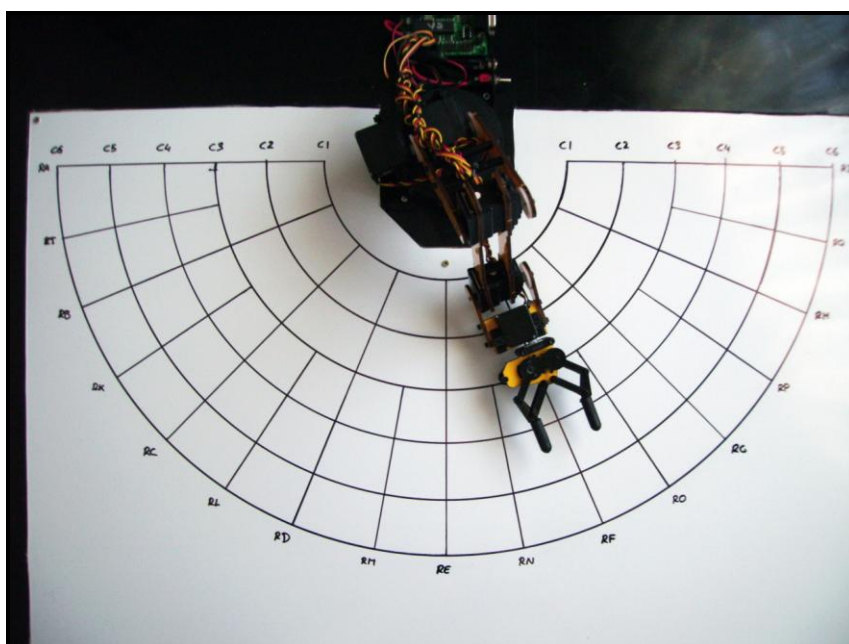
**Fig. 32 – Dibujando la semicircunferencia C3**



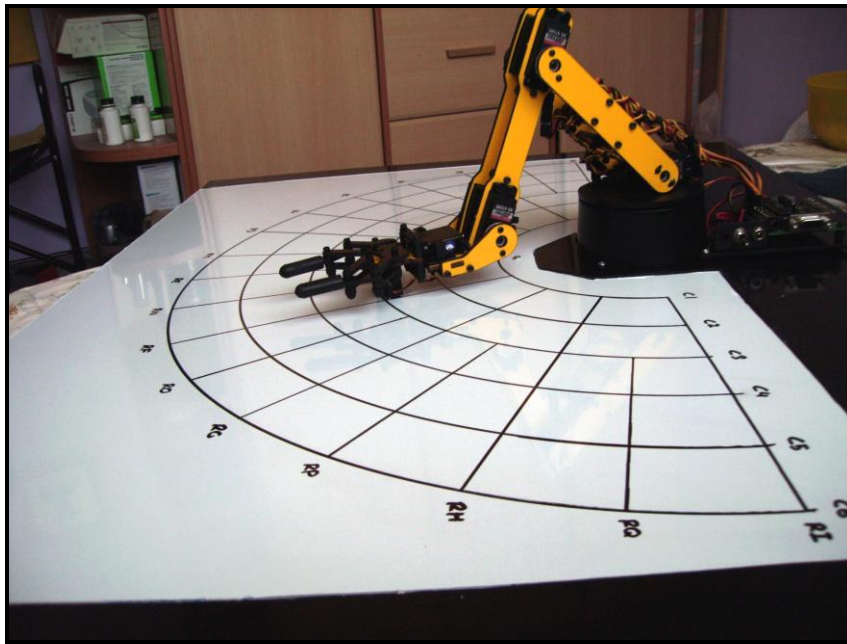
**Fig. 33 – Dibujando la semicircunferencia C2**

Después de haber dibujado todas las circunferencias con el robot, se procedió a desmontarlo de la madera para realizar las líneas rectas utilizando una regla, puesto que era más sencillo que hacérselo hacer al robot. Para ello se adhirió con celo un transportador de ángulos justo en la zona donde estaba el centro de la base del robot para de este modo trazar las líneas rectas según los ángulos que se habían utilizado en el esquema de la Fig. 31 y de este modo se fueron trazando las líneas utilizando la regla y el transportador.

Finalmente cuando se dibujaron todas las líneas, se fijó la plancha blanca de PVC a la madera mediante tres tornillos, uno en la esquina superior izquierda, otro en la inferior derecha y otro justo delante del robot visto desde detrás del robot. Y el resultado final de este bloque se puede observar en las siguientes figuras (Ver Fig. 34 y Fig. 35)



**Fig. 34 – Resultado final entorno trabajo (visión superior)**



**Fig. 35 – Resultado final entorno trabajo (visión lateral)**

Estas son las herramientas que se utilizaron para la creación del entorno de trabajo del robot:

- Sierra con hoja para corte de hierro
- Cúter
- Alicates
- Destornillador de estrella
- Metro
- Rotulador permanente de color negro
- 5 sargentos
- Papel de lija fino
- Aplicador de silicona

#### 4.1.6 Problemas y soluciones

A continuación se muestra un resumen de los problemas con las respectivas soluciones de este módulo:

##### MONTAJE ROBOT

###### **Problemas:**

- Existencia de rebaba en los rodamientos.
- Falta de bridas para recoger los cables de los servos
- Falta de una llave de tubo hexagonal 6-7 y un destornillador de estrella.
- Dificultad para arrancar el plástico protector de los elementos del robot.
- Al apretar al máximo los tornillos de la pinza esta se quedaba bloqueada.

###### **Soluciones:**

- Eliminar la rebaba utilizando un cúter.
- Comprar un paquete de bridas de 2x100mm
- Utilización de llave de tubo hexagonal 6-7 y un destornillador de estrella propios.
- Dedicar más tiempo e incrementar la delicadeza a la hora de retirar el plástico.
- Apretar los tornillos de la pinza de forma gradual, hasta el punto máximo en que no se bloquea la pinza.

##### CREACIÓN ENTORNO DE TRABAJO

###### **Problemas:**

- El arco de la sierra no permitía realizar todo el corte que era necesario.
- No se podía llegar cortando hasta el rincón, ya que luego no se podía girar.
- No se disponía de un compás adecuado para realizar las semicircunferencias.

###### **Soluciones:**

- Desmontar la hoja, enrollarle un trapo en un extremo y realizar el corte con la mano.
- Realizar un corte en forma curva para no llegar hasta la esquina y luego cortar con el cúter la parte que se había esquivado.
- Se utilizó el robot para que fuera el quien dibujara las semicircunferencias

## 4.2 MÓDULO 2 - CIRCUITO DE CONTROL SSC-32

En este módulo tan solo se explicarán los apartados de descripción del módulo y entradas y salidas, ya que el resto de apartados giran en torno a la implementación del módulo, y puesto que este módulo ha sido proporcionado junto con los componentes para el montaje del robot, esta situación anula la información que pudiera incluirse en los apartados afectados. Para el análisis de este módulo se ha utilizado la base teórica adquirida en la asignatura Diseño de Sistemas digitales.

### 4.2.1 Descripción del módulo

Este módulo consiste en un circuito electrónico que hace de enlace entre el sistema físico (Módulo 1 – Robot) y el PC, para ello hace servir el protocolo de comunicaciones serie RS232 para establecer la comunicación con el PC y envía pulsos utilizando los PWM para enviar las ordenes a los actuadores o servomotores del robot.

### 4.2.2 Entradas y Salidas

**Entradas:**

- Instrucciones procedentes del PC codificadas en ASCII (del inglés *American Standard Code for Information Interchange*).
- Tensión continua de 6V (Alimentación)
- Conector para pila de 9V.

**Salidas:**

- Pulsos emitidos por los PWM
- Respuestas codificadas en ASCII a ciertas consultas realizadas desde el PC

**4.2.3 Tecnologías utilizadas**

El circuito SSC-32 (del inglés *serial servo controller*) o controlador serie de servos es un pequeño controlador de servos pre-ensamblado basado en el microcontrolador Atmel ATMEGA 168-20PU que dispone de unas características interesantes. Tiene una alta resolución (1uS) para un posicionamiento preciso, y un movimiento muy suave. El rango está entre 0,50mS y 2,50 mS para obtener un margen de giro de 180°.

El control de movimiento puede ser con respuesta inmediata, con control de velocidad, por tiempo de movimiento o una combinación de ambos. El movimiento en grupo permite tener cualquier combinación a los servos al empezar y que todos terminen el movimiento al mismo tiempo, incluso si los servos se tienen que recorrer diferentes distancias. Esto es una característica muy potente para crear movimientos complejos.

La posición del servo o su movimiento pueden ser consultados para proporcionar información al PC.

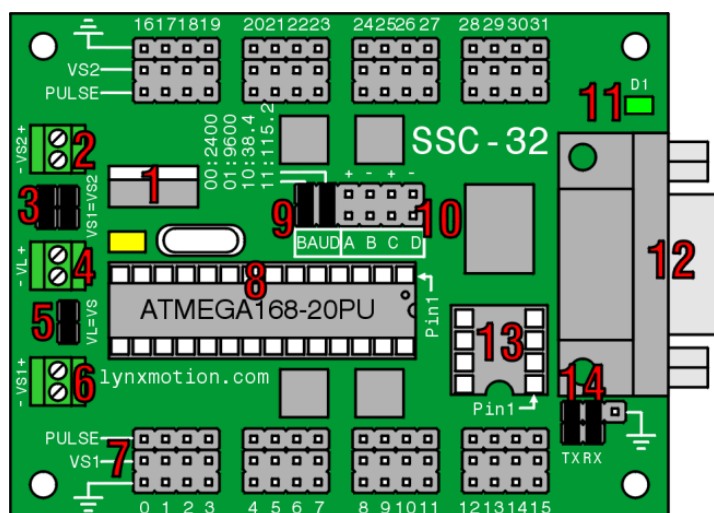
Cualquier salida puede ser utilizada como un nivel TTL de salida. Dispone de 4 entradas digitales que son estáticas o conmutadas, por lo que no hay que preocuparse por la pérdida de un corto evento. Dichas entradas pueden ser utilizadas también como entradas analógicas.

Hay dos bloques terminales para opciones de control. La entrada DB-9 como RS-232 para la comunicación serie con un PC y el socket para una EEPROM 24LC32P el cual será utilizado para futuras versiones del firmware.

A continuación se mostrará una foto de la placa (Ver Fig. 36) un esquema de los módulos de la placa (Ver Fig. 37) y las especificaciones técnicas que detalla el fabricante.



**Fig. 36 – Circuito SSC-32**



**Fig. 37 – Esquema del circuito SSC-32**



Especificaciones técnicas para la versión del firmware 2.01XE (Enlace 10):

- **Micro-controlador** = Atmel ATMEGA168-20PU
- **EEPROM** = 24LC32P (Requerido para 2.01GP)
- **Velocidad** = 14.75 MHz
- **Secuenciador Interno** = 12 Servo *Hexapod (Alternating Tripod)*
- **Entrada Serie** = True RS-232 or TTL, 2400, 9600, 38.4k, 115.2k, N81
- **Salidas** = 32 (Servo o TTL)
- **Entradas** = 4 (Estáticas o Conmutadas, Analógicas o Digitales)
- **Requerimientos de corriente** = 31mA
- **Interfaz de PC** = DB9F
- **Interfaz del micro-controlador** = *Header posts*
- **Control de servos** = Hasta 32 servos conectados directamente
- **Tipo de servo soportado** = Futaba o Hitec
- **Rango de movimiento del servo** = 180°
- **Resolución del servo** = 1uS, .09°
- **Resolución de la velocidad del servo** = 1uS / Segundos
- **Control del movimiento del servo** = Inmediato, por tiempo, por velocidad o Sincronizado.
- **Tamaño de la placa del PC** = 7,6cm x 5,8cm
- **VS capacidad de corriente** = 15 amps per side, 30 amps max

**Nota:** Para ampliar más información sobre este circuito consultar el manual del circuito SSC-32 (Anexo 4) (Enlace 11). Y el esquema electrónico de la placa (Incluido en el CD)

### 4.3 MÓDULO 3 – SECUENCIADOR DE OPERACIONES

#### 4.3.1 Descripción del módulo

Este módulo consiste en una aplicación desarrollada en lenguaje C, que tiene como funciones las de leer un archivo de texto que contiene el programa realizado por los alumnos para la realización de tareas de PPO y traducir cada instrucción al conjunto de instrucciones que es capaz de interpretar el circuito SSC-32. El programa a realizar por los alumnos consistirá en un conjunto de etiquetas (Nombre de las posiciones predefinidas) con el tiempo de ejecución por instrucción, concatenando dichas etiquetas será posible hacer secuencias de operaciones con el robot. Además el módulo determina en que momentos debe enviar las instrucciones al circuito, mediante el protocolo de comunicación serie RS232. Retorna 1 si la ejecución es correcta y 0 si se produce algún error.

El código de este modulo puede consultarse en cd, archivo “secuenciador.cpp”.

Funciones internas:

- **enviar(int k, char Temp[]):** Esta función se encarga de enviar la instrucción seleccionada a través del puerto serie, la función recibe un entero que es el índice del vector de instrucciones PosVECT, y una cadena con el tiempo de ejecución. Devuelve 1 si el resultado es correcto y 0 si ha habido algún error.
- **iniciar():** Esta función se encarga de enviar el robot a la posición de reposo, para asegurarnos que todos los motores han recibido un pulso y el robot no haga movimientos bruscos en la siguiente instrucción. Esta función no recibe ni retorna ningún valor.
- **finalizar():** Esta función se encarga de enviar al robot a la posición de reposo y poner todos los servos a nivel bajos, es decir, desconectarlos. Esta función no recibe ni retorna ningún valor.
- **hash(char pos[]):** Esta función recibe una cadena de texto que representa una etiqueta y se encarga de realizar la búsqueda de la existencia dicha etiqueta dentro del vector PosET, si la encuentra retorna el índice donde se encuentra dicha etiqueta si no devuelve el valor - 1.

Para entender un poco mejor el objetivo de este módulo, se detalla un ejemplo de un sencillo programa a realizar por el alumno:

PosC1RIP1L,2000; → Esta instrucción situaría al robot en las coordenadas C1RIP1L en un tiempo de 2 segundos.

Gripp,500; → Esta instrucción indicaría al robot que cerrara la pinza en un tiempo de 0,5 segundos.

PosC1RIP1H,1000; → Esta instrucción situaría al robot en las coordenadas C1RIP1H en un tiempo de 1 segundo.

PosC3RAP1H,2000; → Esta instrucción situaría al robot en las coordenadas C3RAP1H en un tiempo de 1 segundo.

PosC3RAP1L,1000; → Esta instrucción situaría al robot en las coordenadas C3RAP1L en un tiempo de 1 segundo.

UnGripp,500; → Esta instrucción indicaría al robot que abriera la pinza en un tiempo de 0,5 segundos.

Reposo,1000; → Esta instrucción devolvería al robot al estado de reposo en un tiempo de 0,5 segundos.

Lo que hay antes de la “,” es la etiqueta de posición que será traducida al conjunto de instrucciones que interpreta el circuito SSC-32 para enviar las posiciones a los servos, y lo que hay entre la “,” y el “;” es el tiempo que será traducido para indicar cuanto tiempo debe durar la ejecución de la instrucción tanto por el robot como por el secuenciador.

#### **4.3.2 Entradas y salidas**

**Entradas:** - Archivo de texto en formato “TXT”.

**Salidas:** - Instrucciones para el circuito SSC-32 codificadas en ASCII.

#### **4.3.3 Tecnologías utilizadas**

Para el desarrollo de este módulo se ha hecho servir:

- Compilador de lenguaje C “MinGW Developer Studio”
- Analizador del tráfico RS232 “ComDebud de Windmill”
- Circuito SSC-32 y robot Manipulador Lynx6
- Estándar de comunicaciones Serie RS-232C
- Cable cruzado para comunicación serie con conectores hembra tipo DB-9

#### **4.3.4 Tiempo de desarrollo**

Para llevar a cabo el desarrollo de este módulo se han invertido un total de 77 horas.

### 4.3.5 Implementación

La implementación de este módulo empieza al terminar el montaje del robot manipulador, y pretender programar el robot de modo que realice tareas sin el uso del software comercial que venía incluido junto con las piezas. Para llegar a concluir este modulo se han tenido que superar diversas etapas que se nombran a continuación:

- Definición de los límites físicos de los servos del robot.
- Análisis de la forma de comunicación del circuito SSC-32
- Pruebas con el puerto serie RS232
- Desarrollo del RoboTerminal
- Análisis del software comercial RIOS
- Realización de una secuencia fija utilizando temporizadores
- Realización de una secuencia de tiempo variable
- Creación de una función de lectura de archivos
- Realizar la conversión de etiqueta a instrucción SSC-32
- Codificar el conjunto puntos del espacio de trabajo
- Pruebas y Test

#### 4.3.5.1 Definición de los límites físicos de los servos del robot

En esta etapa se definieron cuales eran los límites físicos de los servos del robot y se analizó que pulso máximo y mínimo se le podía enviar por cada canal al circuito SSC-32, sin que se forzase a cada servo, a llegar a posiciones a las que mecánicamente no era capaz de alcanzar.

Por este motivo y utilizando el software lynxterm, se fueron probando los servos uno a uno, y enviando diferentes pulsos a los PWM para observar visualmente si se había llegado al límite mecánico del servo.

Después de realizar este proceso se elaboró el siguiente esquema (Ver Fig. 38) en el que se detallan las posiciones que toma cada servo en función de los pulsos característicos, que son: máximo, centro y mínimo.

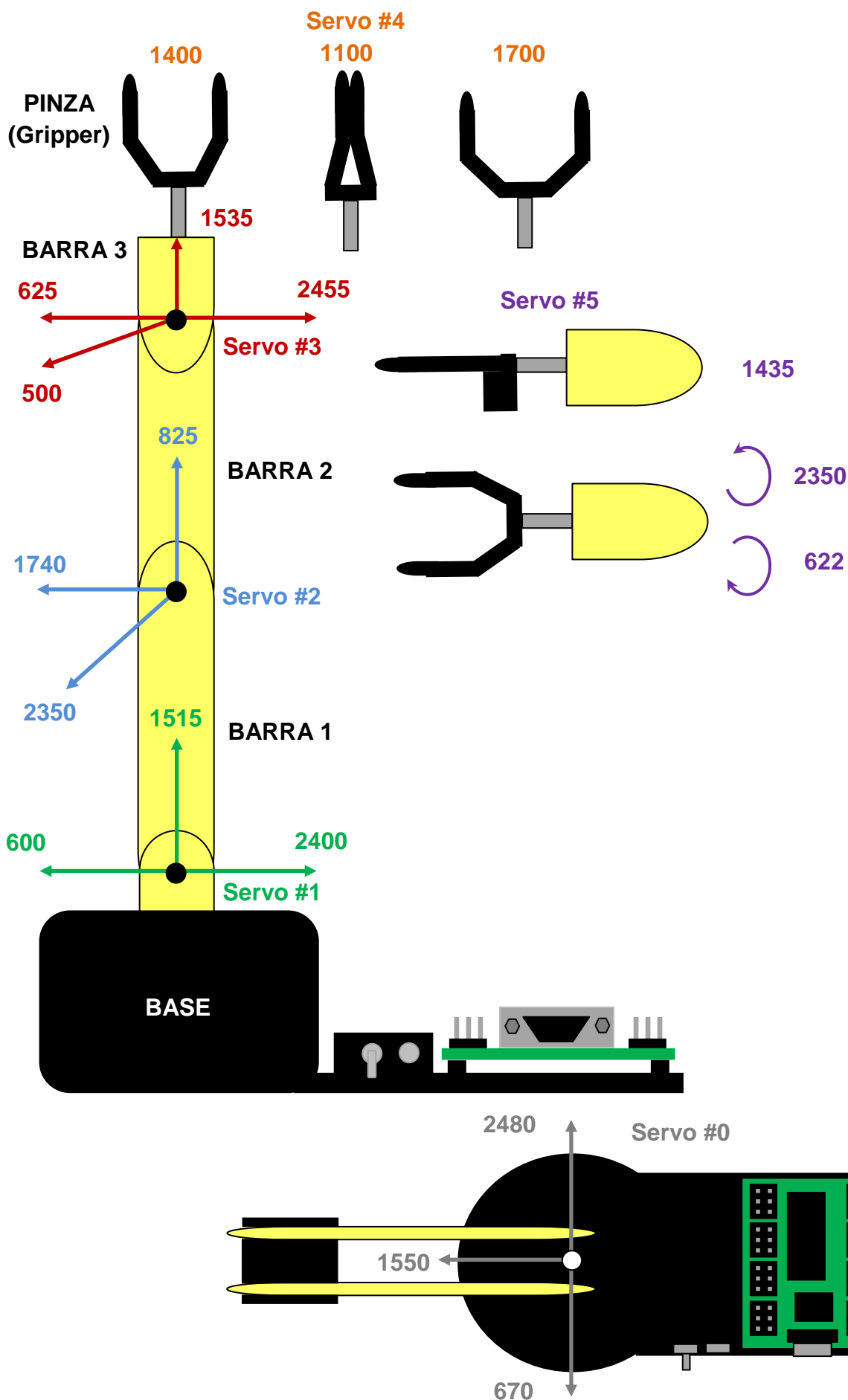


Fig. 38 – Esquema de posiciones de los servos según el pulso enviado por los PWM

#### 4.3.5.2 Análisis de la forma de comunicación del circuito SSC-32

Después de realizar las pruebas y test pertinentes, para verificar el correcto funcionamiento de los servos del robot mediante el software lynxterm una vez finalizado el montaje del robot manipulador, se sabía la configuración con la que se abría el puerto serie, y posteriormente se intentó analizar el código de dicho programa, para ver cómo utilizarlo dentro del proyecto, cosa que no fue posible dado que era de código cerrado.

Este hecho obligó a desechar esa posibilidad y contemplar el hecho de que habría que establecer la comunicación con el robot desde 0, sin ninguna otra aplicación ya creada como intermediaria.

Se inició así una investigación sobre el funcionamiento del circuito SSC-32, para analizar su arquitectura y el tipo de trama que enviaba a través del puerto serie.

Para ello hubo que registrarse en el foro de ayuda de la web lynxmotion (Enlace 12) y también en el de ARDE (Asociación de Robótica y Domótica de España) (Enlace 13), además de buscar los manuales del circuito SSC-32 (Anexo 4).

Tras haber revisado el manual del circuito SSC-32 y haber realizado ciertas consultas en los foros mencionados, sobre todo en el de lynxmotion, se determinó el formato de la trama que se enviaba como una colección de caracteres ASCII seguido del retorno de carro para iniciar la ejecución de la instrucción.

Datos relevantes de esta etapa:

- Configuración del puerto serie para la comunicación con el circuito SSC-32

<b>Puerto:</b>	COM 1
<b>Baudios:</b>	115200
<b>Bits:</b>	8
<b>Paridad:</b>	Ninguna
<b>Bits de parada:</b>	1
<b>Control de flujo:</b>	Ninguno

- Formato de la instrucción (Extracto del Anexo 4 pág. 5 (traducido))

	# <ch> P <pw> S <spd> ... # <ch> P <pw> S <spd> T <time> <cr>
<ch>	Número de Canal en decimal, 0-31
<pw>	Ancho del Pulso en microsegundos, 500-2500
<spd>	Velocidad del movimiento en uS por Segundo para cada canal (Opcional)
<time>	Tiempo en mS para completar un movimiento, afecta a todos los canales, 65535 máx. (Opcional)
<cr>	Carácter de retorno de carro, ASCII 13 (Requerido para iniciar la acción)
<esc>	Cancela la acción en curso, ASCII 27

#### 4.3.5.3 Pruebas con el puerto serie RS232

Al llegar a esta etapa ya se sabía qué tipo de comunicación había que establecer y qué tipo de trama había que enviar. En este punto surgía una nueva pregunta, que era ¿cómo realizar esto?.

Para ello se estuvo buscando a través de internet todo lo referente al puerto serie RS232, y entre otra información se encontraron unas prácticas (Anexos 6 y 7) (Enlace 14) desarrolladas por el profesor José Antonio Rodríguez Mondéjar de la Universidad Pontificia Comillas, que sirvieron como base teórica para realizar las pruebas, así como unas APIs del puerto serie RS232 desarrolladas por él mismo en lenguaje C.

En este punto se recordaron unas prácticas realizadas en la asignatura Informática Industrial, en las que era necesario establecer una comunicación serie para enviar comandos a un PLC. Ya que el principio era el mismo, se decidió modificar esas prácticas, realizadas con el programa Labview, para enviar las tramas en el formato que requería el circuito SSC-32 en sustitución de las que se le enviaban PLC.

Después de modificar el programa se realizó una prueba sin éxito, y se decidió analizar exactamente la trama que recibía el circuito SSC-32. Para ello se hizo uso de la práctica 1 “Herramientas básicas para manejar la RS232” (Anexo 5) encontrada en Internet. Esta práctica sirvió como base para fabricar un cable RS232 cruzado con dos conectores hembra tipo DB-9, que se muestra en las siguientes figuras. (Ver Fig. 39 y Fig. 40).

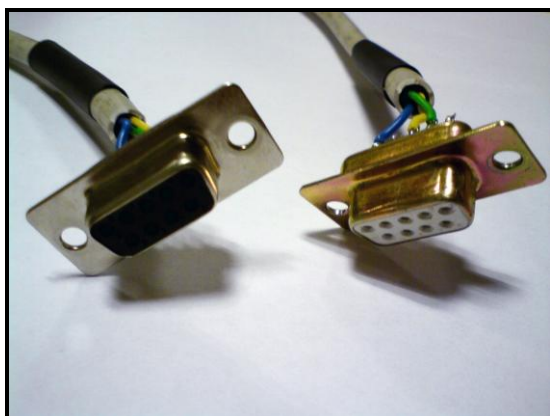


Fig. 39 – Cable RS232 Cruzado



Fig. 40 – Cable RS232 Cruzado

También sirvió para instalar y configurar correctamente el software ComDebug para poder analizar el tráfico byte a byte.

Una vez estaba listo el cable serie cruzado, se hizo una prueba rápida con el Hyperterminal de Windows XP. Se configuró el Hyperterminal en el puerto COM2 para recibir lo que se enviaba por el COM1, pero aparentemente las tramas enviadas eran correctas.

Es en este punto donde se instala ComDebug y se configura bajo el mismo criterio que el Hyperterminal, y aquí si se observa cual era el problema por el cual el robot no respondía a los comandos enviados desde el programa hecho en Labview, ya que con este programa muestra byte a byte lo que se transmite por el puerto serie. El problema estaba en el carácter de retorno de carro, el cual no se enviaba correctamente, en lugar de este carácter lo que se enviaba era el carácter "0x0A" correspondiente al "ENTER", una vez conocido este se modificó el programa para enviar correctamente el retorno de carro (Ver Fig. 41 Circulo verde). En las siguientes figuras se muestra el programa realizado en Labview para el envío de tramas por el puerto serie en formato reconocible por el circuito SSC-32. (Ver Fig. 41, Fig. 42, Fig. 43, Fig. 44 y Fig. 45).

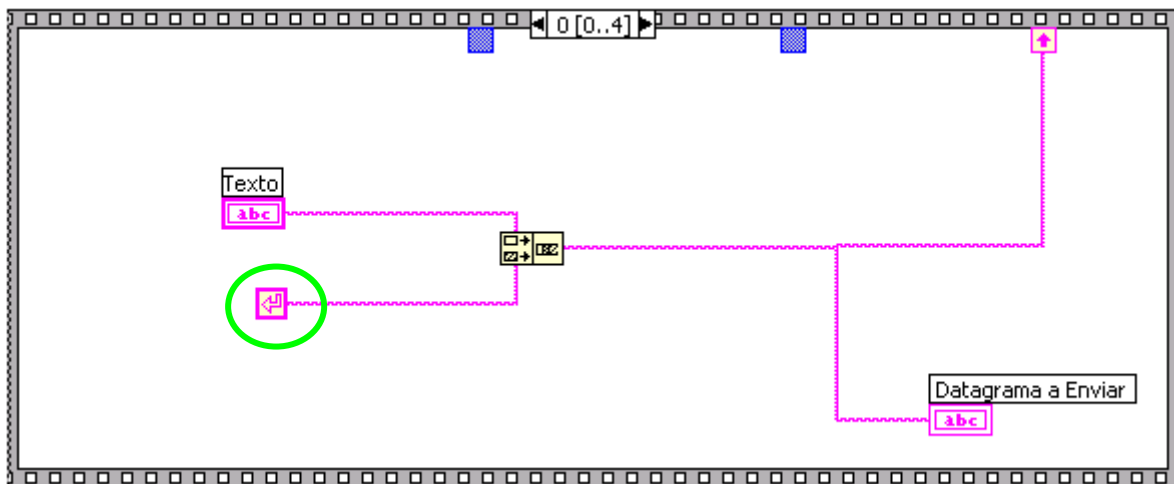


Fig. 41 – Secuencia 0 del programa Labview (Construcción del datagrama)

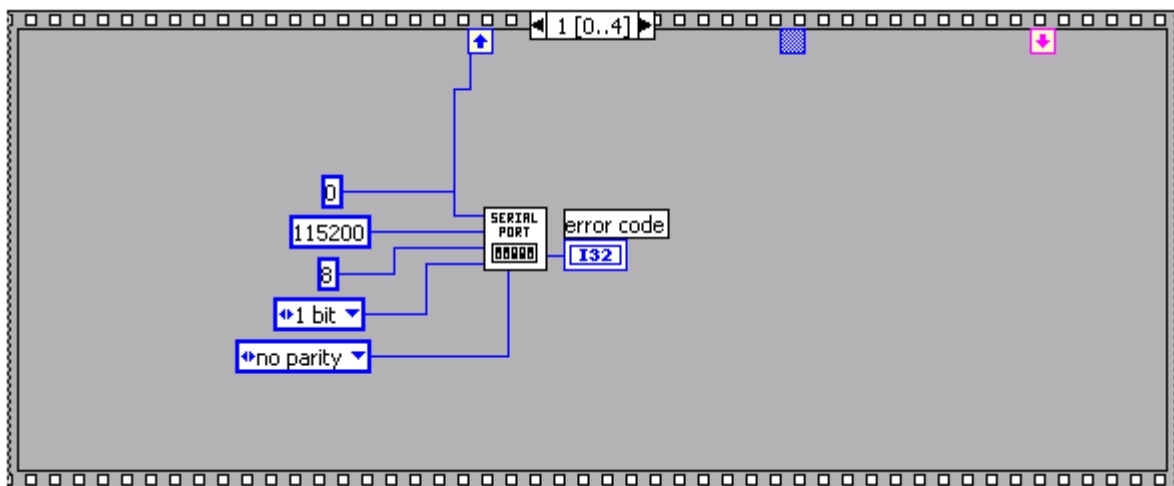


Fig. 42 – Secuencia 1 del programa Labview (Configuración del puerto serie)



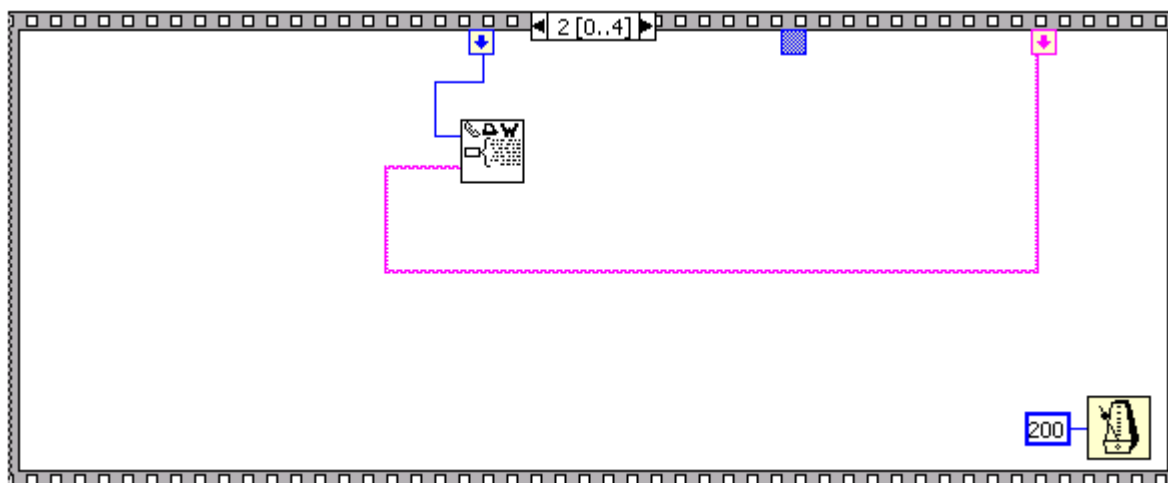


Fig. 43 – Secuencia 2 del programa Labview (Envío de caracteres ASCII)

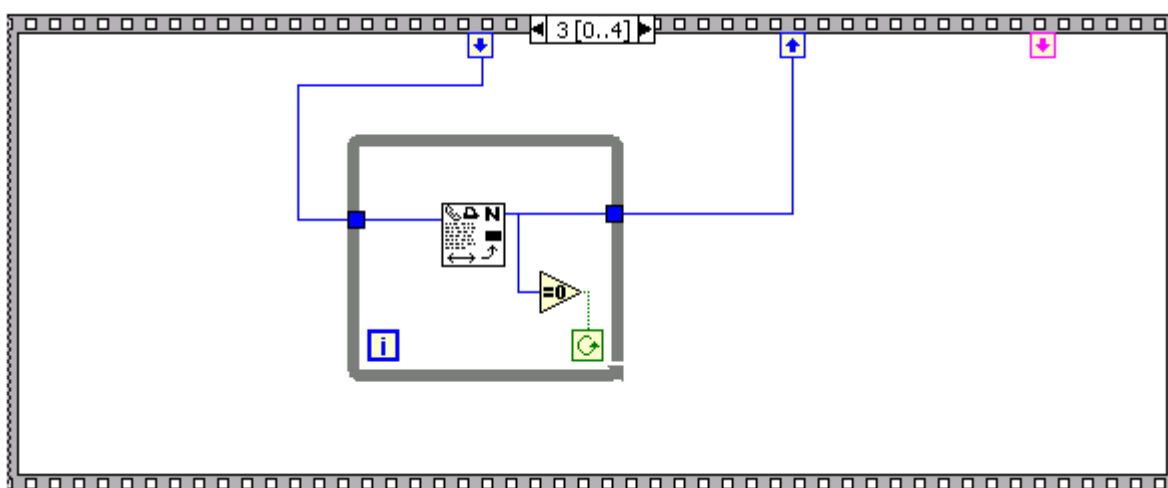


Fig. 44 – Secuencia 3 del programa Labview(Comprobación resto caracteres)

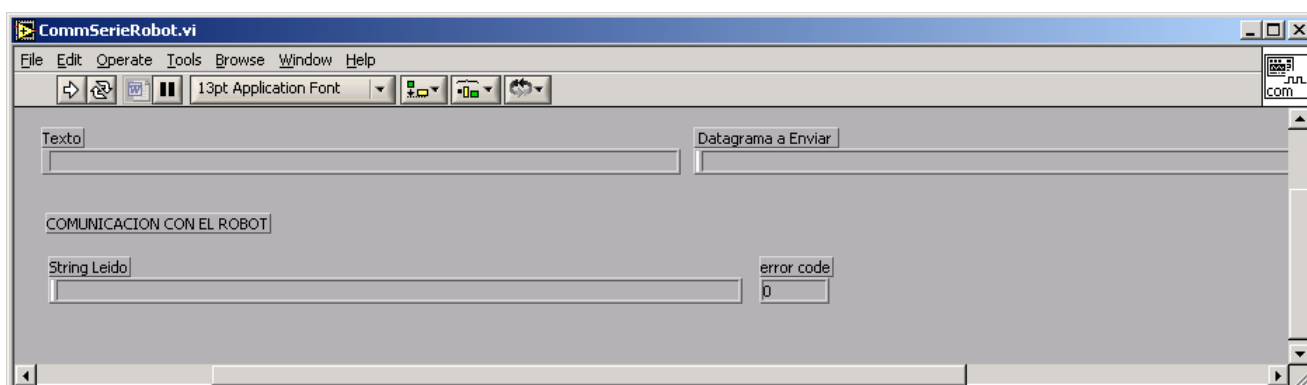


Fig. 45 – Parte del programa Labview desde donde se introduce la instrucción a enviar

Después de realizar esta modificación se analizó el tráfico con el programa ComDebug y efectivamente la trama enviada ya era correcta e incluía al final el carácter “Retorno de Carro” o “0x0D” en hexadecimal (Ver Fig. 46 Circulo Verde).

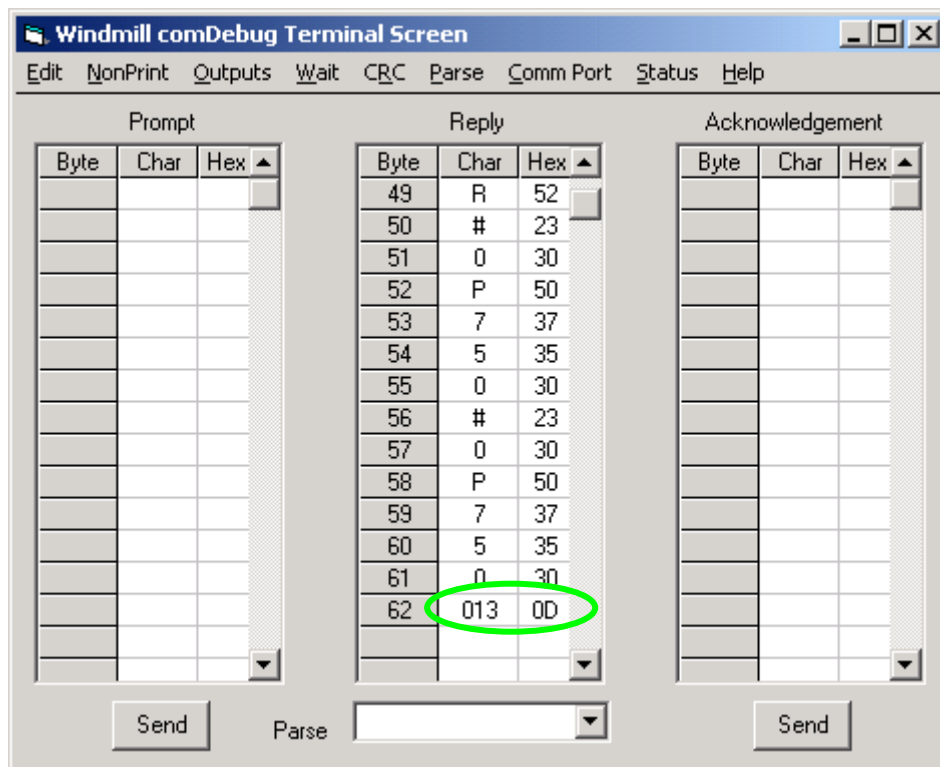


Fig. 46 – Captura del programa comDebug (recibiendo trama del Labview)

Tras realizar esta comprobación, se conecto el robot al PC y efectivamente ya respondía a las instrucciones enviadas desde la práctica modificada de la asignatura Informática Industrial.

Antes de pasar a la siguiente etapa se aprovechó para realizar el mismo tipo de prueba pero con el software lynxterm, es decir, se configuró el hyperterminal y el ComDebug (Ver Fig. 47) para ver que enviaba dicho programa.

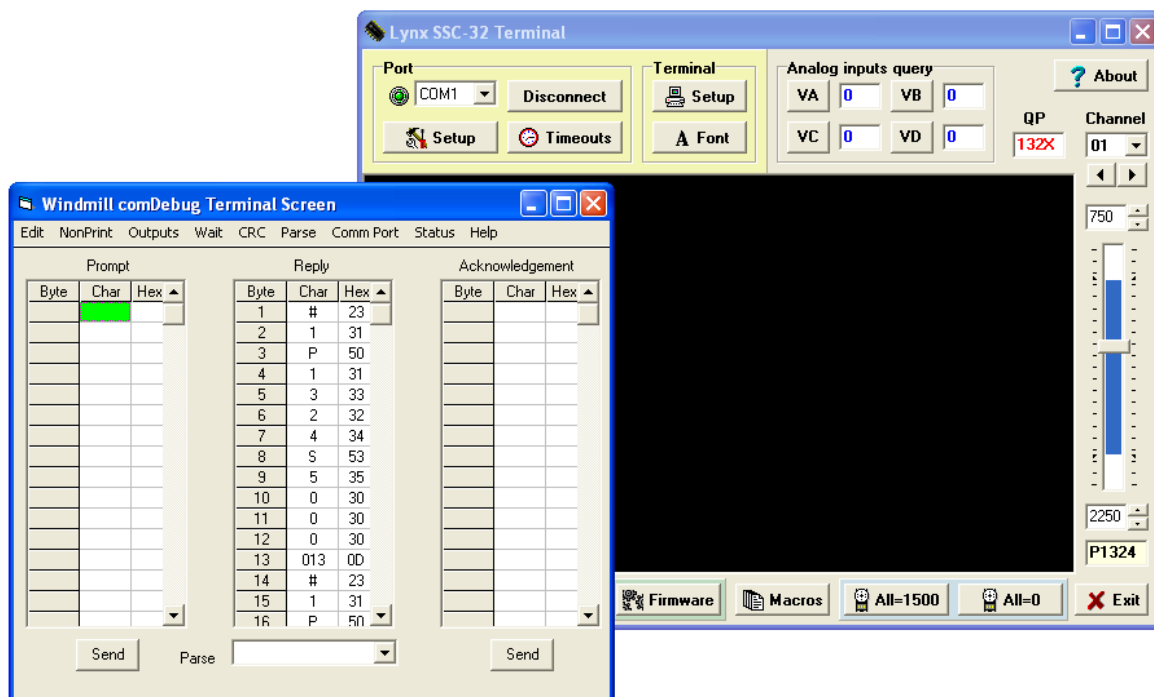


Fig. 47 – Lynxterm enviando y comDebug recibiendo

#### 4.3.5.4 Desarrollo del RoboTerminal

Una vez establecido el control del robot sin hacer uso del software que se incluía con las piezas, se decide realizar un terminal propio para el control del robot, programado en lenguaje C e inspirado en la idea del lynxterm.

En esta etapa se hizo uso de la práctica 3 “Programación básica de la RS232” (Anexo 6) encontrada en Internet. Esta práctica sirvió para recordar los conceptos básicos sobre la programación de la comunicación serie RS232, cuya base teórica se adquirió en la asignatura Diseño de Sistemas Digitales, y para descargar el compilador MinGW Developer (Enlace 15) con el que desarrollar programas en C.

Además se hizo uso de las APIs que se incluían en esa práctica, para poder manejar el puerto serie de forma que se podía abrir el puerto, enviar un dato, cerrar el puerto, etc. tan solo llamando a dichas funciones pasándole los parámetros correctos.

El funcionamiento del RoboTerminal es sencillo, (para consultar el código Ver CD, archivo “RoboTerminal.cpp”), lo que hace básicamente es abrir el puerto serie COM1 con la configuración adecuada para establecer comunicación con el circuito SSC-32, leer los caracteres entrados por teclado y enviarlos a través del puerto serie, en el momento que detecta la pulsación de “ENTER” sustituye el carácter “0x0A” asociado a esa tecla por el carácter “0x0D” o “Retorno de Carro” y a continuación lo envía por el puerto serie, de este modo se ejecuta el comando que se había escrito. Si en algún momento se introduce el carácter “x” seguido de “ENTER”, esta acción finaliza la ejecución del RoboTerminal.

#### 4.3.5.5 Análisis del software comercial RIOS

Tras haber completado las etapas anteriores con éxito, en este punto lo que se pretendía conseguir, era ver de qué manera el software comercial RIOS (del inglés *Robotic arm Interactive Operating System*) de lynxmotion, decidía en que momentos enviar las instrucciones correspondientes para la ejecución secuencial de movimientos desde un programa realizado con este software.

Se intentó aplicar el mismo procedimiento que en la etapa anterior, se realizó una pequeña secuencia de movimientos con el software RIOS que se reproducían en el robot, posteriormente se utilizó el cable serie cruzado y el software ComDebug para analizar que tramas se enviaban. Lamentablemente no se pudo llevar a cabo la prueba debido a que, el software comercial RIOS no enviaba datos por el puerto serie si detectaba que el circuito SSC-32 no estaba conectado, y al ser de código cerrado no se pudo analizar lo que se pretendía.

Por otra parte se sabía, por haber leído el manual del circuito SSC-32 (Anexo 4), que existía una instrucción, que enviada al circuito SSC-32, informaba si se había terminado la instrucción anterior o no, pero por desgracia, esta instrucción nunca se pudo utilizar de forma satisfactoria.

#### **4.3.5.6 Realización de una secuencia fija utilizando temporizadores**

Tras ver el problema surgido en la etapa anterior, se estuvo pensando de qué manera se podría controlar cuando iniciar la siguiente instrucción sin que el circuito SSC-32 informara de ello.

Revisando el manual del circuito SSC-32, se observó que a la hora de enviar una instrucción al circuito SSC-32, podía indicarse el tiempo en el que se debía realizar la acción, de modo que todos los servos acabaran el movimiento al mismo tiempo, aunque para ello debieran moverse a distintas velocidades. Gracias a esta funcionalidad se pensó que si se conocía el tiempo que tardaría en llevarse a cabo la acción, también se sabía cuando iniciar la siguiente, y por eso se inició la investigación de cómo utilizar temporizadores en C para indicar cuándo enviar la siguiente instrucción a través del puerto serie.

Primero se tuvo que revisar la forma de construir cadenas de caracteres en lenguaje C para almacenar las posiciones que se utilizarían, la base teórica fue adquirida en la asignatura Lenguajes de Programación impartida en la ETSE, y vuelta a revisar mediante la información disponible en una página web (Enlace 16).

Después de haber revisado el punto anterior, se procedió a la búsqueda de cómo crear un retardo determinado en lenguaje C, y navegando por internet se encontró una página web (Enlace 17) donde se hablaba de una instrucción llamada Sleep() que servía para generar un retardo de tantos milisegundos como se le pasasen por argumento.

Utilizando estas herramientas se realizó un programa piloto que enviaba una secuencia de instrucciones fijas a través del puerto serie, y esperaba entre instrucción e instrucción, el mismo tiempo en que debía llevar a cabo la acción que se le había indicado al robot. De este modo fue posible la secuenciación de operaciones.

#### **4.3.5.7 Realización de una secuencia fija de tiempo variable**

El objetivo en esta etapa era el de poder modificar una variable en la que se guardaría el tiempo en el que se deberían llevar a cabo todas las operaciones de la secuencia.

Para ello se creó una variable de tipo cadena de texto en la que se guardaba el tiempo en formato texto, entonces dicha variable era utilizada tal cual para ser enviada por el puerto serie carácter a carácter y al mismo tiempo utilizando la función atoi() incluida en la librería stdlib.h se convertía la variable a entero para enviárselo a la función Sleep().

De este modo se podía cambiar el tiempo en el que se realizarían todas las acciones cambiando solo una variable.

Este cambio fue realizado sobre programa de la etapa anterior con éxito.

#### 4.3.5.8 Creación de una función de lectura de archivos

Tras concluir la etapa anterior se tenían los medios necesarios para realizar secuencias de operaciones con el robot, en las que se podía controlar el tiempo de ejecución de todas las operaciones a la vez, de manera que todas las operaciones tardaran lo mismo.

Llegados a este punto se pretendía poder leer el contenido de un archivo, para de este modo, realizar la secuencia de operaciones a realizar por el robot en un archivo de texto, y así poder realizar secuencias variables, es decir, que no estuvieran preestablecidas dentro del programa como se había hecho hasta el momento.

Se definió cual sería la estructura de las instrucciones que serían leídas desde el archivo, que fue la siguiente:

FORMATO DE LA INSTRUCCIÓN			
<i>Etiqueta de posición</i>	,	<i>Tiempo de ejecución</i>	;

En primer lugar se consulto que opciones daba C para el manejo de archivos, para ello se hizo uso de un manual en C redactado por el profesor José Luis Asín Buñuel del IES (Instituto de Enseñanza Secundaria) Barañain (Incluido en el CD), en concreto del capítulo 13 de dicho documento.

Después de haber consultado dicho manual se escribió el código correspondiente que abría el archivo y que iba leyendo carácter a carácter el contenido del mismo mediante la función "*getc()*". Tras realizar algunas pruebas se observó que este no era un buen método para analizar posteriormente el contenido de cada línea del archivo, así que se decidió utilizar la función "*gets()*". Esta función fue muy útil puesto que permitía la lectura de una línea completa del archivo y asignar lo leído a una variable de tipo cadena de caracteres, que posteriormente era la que se analizaba.

La cadena de caracteres era recorrida carácter a carácter guardando lo que leía en una variable hasta encontrar el carácter ",", tras este carácter guardaba lo que iba leyendo en otra variable diferente hasta encontrar el carácter ";".

Habiendo conseguido esto, ya era posible escribir un programa basado en etiquetas, mediante un editor de texto para ser guardado en un archivo, y, que el programa secuenciador abriera este archivo leyéndolo línea a línea y separando en cada línea la etiqueta de posición del tiempo de ejecución.

#### 4.3.5.9 Realizar la conversión de etiqueta a instrucción SSC-32

Habiendo superado las etapas anteriores surge una nueva pregunta, que era: ¿Como realizar la traducción de la etiqueta de posición leída del archivo al conjunto de caracteres que interpreta el circuito SSC-32 y que representan dicha posición en el espacio de trabajo del robot?

La respuesta a esta pregunta es: utilizando un método de *hashing*, mediante el cual pueden realizarse búsquedas de elementos pudiéndonos basar en índices no numéricos.

Inicialmente se intentó buscar si había alguna librería de C que ya incluyera esta función, se encontró dicha función para C++, pero no para C, así que hubo que pensar la manera de implementar esta función en C, para ello se hizo uso de la base teórica adquirida en la asignatura Estructura de Datos.

La idea era crear dos vectores de manera que en el primer vector contuviera todas las etiquetas de posición del entorno de trabajo, y un segundo vector que contuviera la instrucción o configuración de los servos a enviar al circuito SSC-32. La clave está en que los vectores están ordenados de forma que la etiqueta de posición que se encuentra en la posición "i" del primer vector, se corresponde con la instrucción que hay almacenada en la misma posición pero en este caso del vector 2.

Teniendo en cuenta esto se realizó una función a la que se le pasaba una cadena de caracteres que representaba la etiqueta de posición y bucle recorría el primer vector, llamado (VectET) en el programa, realizando comparaciones entre la etiqueta que se había pasado y el contenido de cada posición del vector, en el momento que se encontraba una coincidencia la función retornaba el índice correspondiente. De este modo se podía utilizar dicho índice para acceder a la posición del segundo vector, llamado (VectPos) en el programa, y extraer la instrucción correspondiente que se enviaría al circuito SSC-32.

Durante la implementación de esta etapa, surgió la pregunta de cómo crear un vector de cadenas de caracteres en C, así que tras una búsqueda por internet se encontró información útil (Enlace 18), que llevó a la forma de definir dicha estructura de datos mediante la definición de apuntadores.

#### 4.3.5.10 Codificar el conjunto de puntos del espacio de trabajo

Para la realización de este bloque se diseñó un esquema con una visión del plano vertical de los puntos característicos del espacio de trabajo del robot (Ver Fig. 48), dicho esquema es una proyección del realizado en el “Módulo 1. Robot” (Ver Fig. 49), en el cual se muestra la visión del plano horizontal.

Con dichos planos puede identificarse cualquiera de los puntos característicos tan solo refiriéndose a las coordenadas que los definen. Cada plano tiene una parte alta “H” y otra baja “L”, donde la parte alta sirve para situar al robot correctamente sobre el objeto que se encuentre en el plano correspondiente y la parte baja “L” sitúa al robot en una posición en la cual cerrando la pinza, ésta coge el objeto de dicho plano.

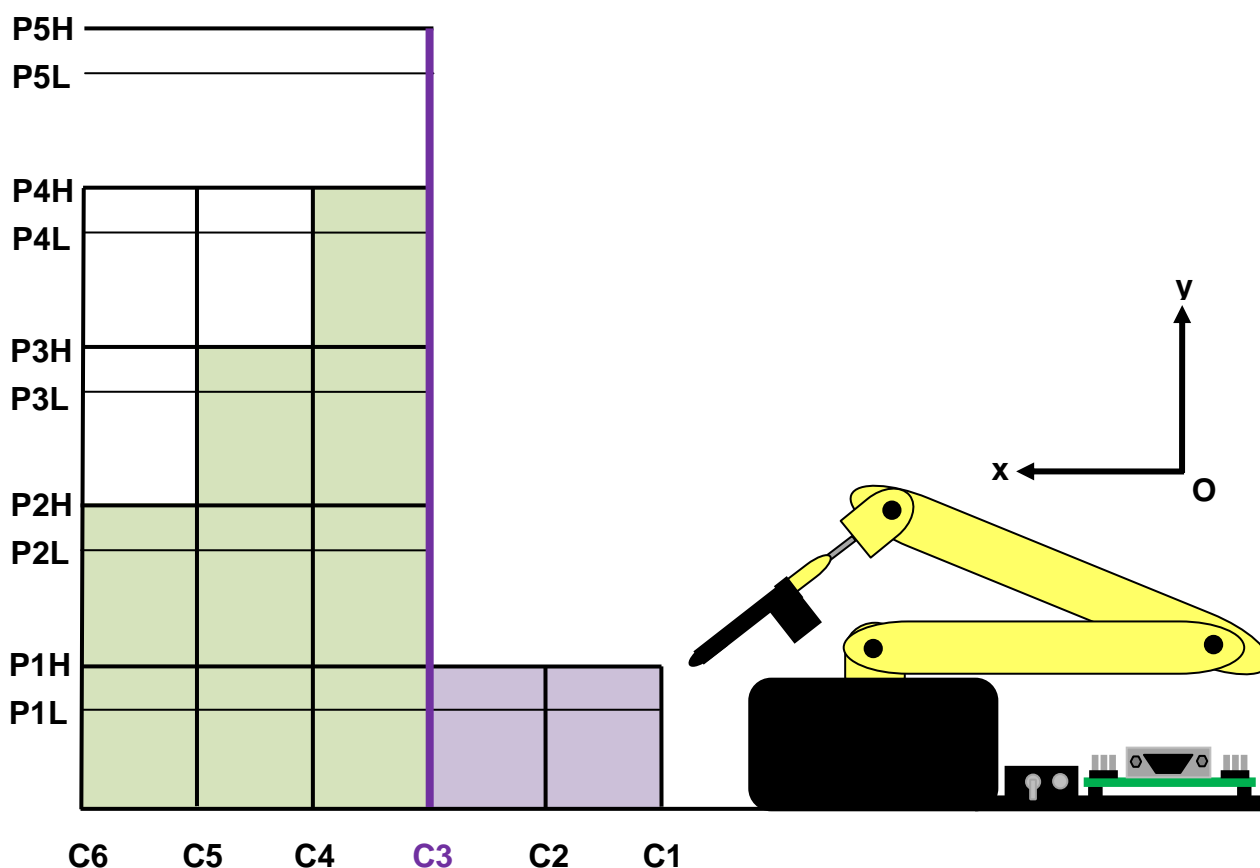


Fig. 48 – Esquema de puntos característicos (Visión vertical)

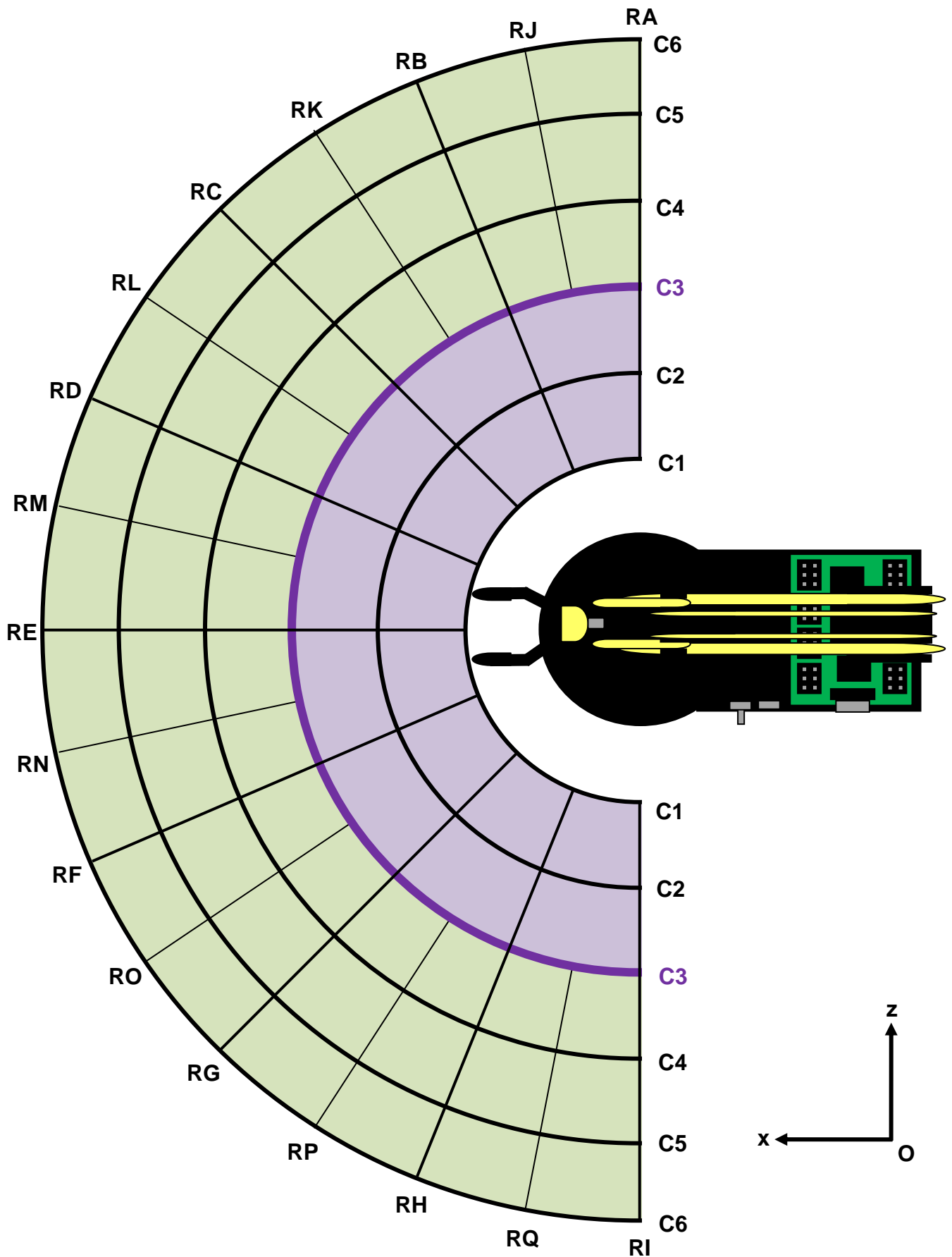
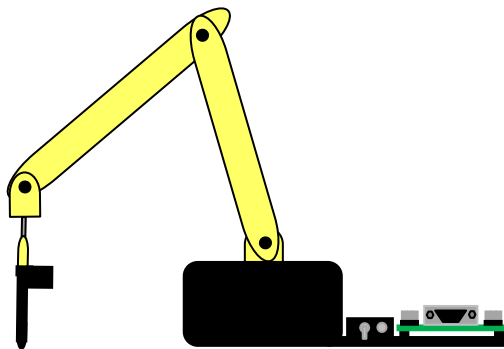


Fig. 49 – Esquema de puntos característicos (Visión horizontal)



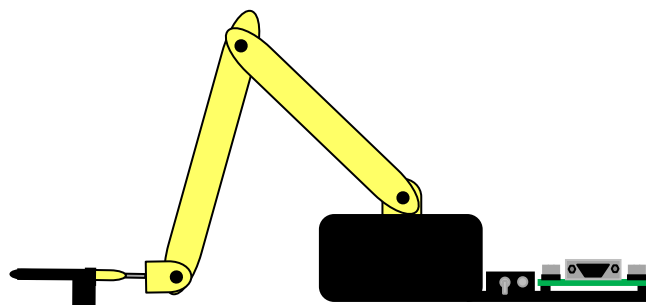
Existen dos zonas de trabajo distintas dentro del espacio de trabajo del robot, en las cuales el robot adquiere una configuración (inclinación) distinta para situarse sobre los puntos característicos y así poder coger los objetos cerrando la pinza.

La “Zona 1” está identificada sobre el esquema mediante el color lila, y en esta zona están contenidos los radios “RA, RB, RC, RE, RF, RG, RH y RI”, las semicircunferencias “C1, C2 y C3” y el Plano 1 con su parte baja y alta “P1L y P1H”. La configuración que adquiere el robot para situarse sobre los puntos característicos en esta zona, es como la que se observa en la siguiente figura (Ver Fig. 50).



**Fig. 50 – Configuración robot en zona 1**

La “Zona 2” está identificada sobre el esquema mediante el color verde, y en esta zona están contenidos los radios “RA, RB, RC, RE, RF, RG, RH, RI, RJ, RK, RL, RM, RN, RO, RP y RQ”, las semicircunferencias “C3, C4, C5 y C6” y los Planos del 1 al 5, aunque no todos los planos son accesibles desde todas las semicircunferencias, 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” y “P5L y P5H: C3”. Esta limitación de la altura viene dada por las propias limitaciones físicas del robot, las cuales no permiten alcanzar los puntos que no se han incluido, ya que se hayan fuera de su espacio de trabajo. La configuración que adquiere el robot para situarse sobre los puntos característicos en esta zona, es como la que se observa en la siguiente figura (Ver Fig. 51).



**Fig. 51 – Configuración robot en zona 2**

Teniendo ya bien identificados todos los puntos característicos del espacio de trabajo, se definió cual sería el formato de las etiquetas para referirse a cada uno de ellos, que fue el siguiente:

	Cabecera	S.Circunferencia	Radio	Plano	Altura	Zona
<b>Ejemplo 1</b>	Pos	C1	RA	P1	L	
<b>Ejemplo 2</b>	Pos	C3	RA	P1	L	Z1
<b>Ejemplo 3</b>	Pos	C3	RE	P1	H	Z2

Las etiquetas de los ejemplos anteriores quedarían de la siguiente forma:

- PosC1RAP1L
- PosC3RAP1LZ1
- PosC3REP1HZ2

La “zona” solo se utiliza para identificar con que configuración se sitúa el robot sobre los puntos contenidos en el plano “P1H” y “P1L” de la semicircunferencia “C3”, ya que ésta es el nexo de unión entre la zona de trabajo 1 y la zona de trabajo 2 y sirve para poder llevar objetos de una zona a la otra sin alterar la inclinación del objeto.

En total se han definido 563 puntos característicos, además de dos configuraciones predefinidas que se identifican con las etiquetas “Reposo”, la cual coloca al robot en posición de reposo (lo más retraído posible) (Ver Fig. 52) , “Depie” que sitúa al robot completamente extendido en posición vertical (Ver Fig. 53), y tres instrucciones especiales que son “GiroH” para realizar un giro de 90° de la muñeca en sentido horario, “GiroAH” para realizar un giro de 90° de la muñeca en sentido anti horario y Sleep para realizar una pausa en la que el robot mantiene su configuración durante el tiempo indicado hasta de recibir la siguiente instrucción.

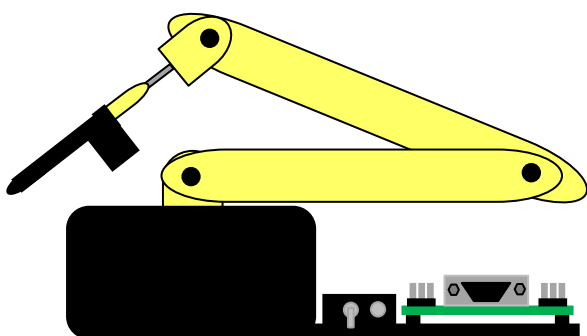


Fig. 52 – Robot posición reposo

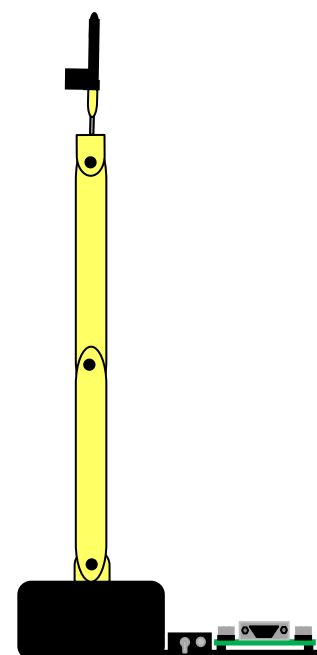


Fig. 53 – Robot posición vertical

#### 4.3.5.11 Pruebas y test

Al finalizar todas estas etapas, se creó la aplicación “Testeador.exe” que no es más que el “Secuenciador.exe”, pero que permite la ejecución paso a paso, es decir, que para cada instrucción leída del archivo de texto, requiere que se pulse una tecla para enviar la orden al robot, código de la aplicación puede consultarse en el archivo “testeador.cpp” del CD.

Teniendo lista la aplicación anterior, se creó un programa cuyo propósito era que el robot recorriera todos y cada uno de los puntos característicos, de modo que con el “Testeador.exe” se podría observar si el robot se situaba correctamente en los puntos que se le indicaban, y en caso contrario depurar los posibles errores que hubieran en las coordenadas.

Tras verificar y ajustar todos los puntos, se realizó un breve programa para testear definitivamente el funcionamiento del robot, con el que se pudo observar que efectivamente robot iba realizando correctamente las acciones indicadas en el archivo de texto que se habían creado y que el secuenciador estaba leyendo.

Hay que decir que esta prueba, tan solo era la prueba final para verificar que este módulo funcionaba correctamente. Pero al finalizar cada etapa se realizaban las pruebas pertinentes para verificar que lo que se había realizado en dicha etapa era correcto.

Durante las pruebas finales, se observó que según qué posición se le hacía alcanzar al robot en la primera instrucción, los cables se enganchaban en los servos de la primera articulación y provocaban que el robot se atascase e incluso que se desconectase el cable del servo de la articulación 2, para ello se modificó en el “secuencializador.exe” para que por defecto el robot se situara en la posición “PosC1REP1L” de manera que desde esta posición se podía ir a cualquier punto sin que surgiese este problema.

También se detectó que existía un offset, es decir, un desfase en la posición según el sentido de giro del servo 0, (el de la base del robot). Todos los puntos característicos han sido codificados siguiendo un orden anti horario, empezando desde el radio “RA” y acabando en el radio “RI”. Así cuando indicas al robot que alcance un punto de su espacio, si se mueve en sentido anti horario, al ser el sentido sobre el que se codificaron los puntos, el robot se sitúa correctamente, pero si para alcanzar el mismo punto, el robot debe realizar un giro en sentido horario, éste se detiene 1 cm antes de llegar a alcanzar el punto.

Dicho problema es originado por el propio mecanismo, con lo cual podría arreglarse mediante hardware, sustituyendo el tipo de servo que lleva el motor, o mediante software, teniendo que tener en cuenta la existencia de este offset. Obviamente se descartó la solución mediante hardware, ya que era más costosa, y requeriría la revisión y posible modificación de todo el proyecto, así que se investigó sobre las soluciones vía software.

Se plantearon dos soluciones alternativas a este problema:

1. Definir de nuevo las etiquetas y añadir el sentido en el que se moverá el robot por ejemplo PosC1REP1HSH o PosC1REP1HSAH. Con lo cual el número de puntos codificados sería prácticamente el doble que los físicos, ya que habría que definir dos etiquetas para cada punto que se pudiera alcanzar desde los dos sentidos.
2. Definir este problema como un hándicap en la programación del robot, que el alumno deberá tener en cuenta, y cuya solución sería que a la hora de programar la secuencia de operaciones que tiene que realizar el robot, si debe alcanzar un punto "P" y previamente se envió al robot a una posición desde la que tendrá que realizar un giro horario para alcanzarlo, en lugar de enviar directamente al robot al punto "P" se le envía al siguiente punto más cercano a su izquierda (mirando de frente al robot) y desde ahí se le envía al punto "P", asegurando así que alcanza dicho punto mediante un giro anti horario.

De las dos soluciones planteadas, se escogió la segunda, puesto que era la que implicaba no retocar el código del programa y puesto que añadía un grado más de complejidad a la hora de que los alumnos realizaran los programas, ya que tendrían que tener en cuenta desde donde se mueve el robot.

Para finalizar la explicación de este módulo se indica que el programa se ha diseñado de forma que sea sumamente sencillo incluir nuevas etiquetas de posición, para poder utilizarlas en los programas a realizar por los alumnos.

Para llevar a cabo la ampliación del número de posiciones predefinidas que alcanza el robot, tan solo basta con añadir la etiqueta de posición con el nombre que se desee al final del vector "VectET" y la correspondiente configuración de los servos para esa posición dentro del espacio de trabajo del robot, al final del vector "VectPos. Una vez realizado esto, dicha etiqueta de posición podrá ser utilizada dentro de los futuros programas.

#### 4.3.6 Problemas y soluciones

##### Problemas:

- El programa proporcionado junto con el robot era de código cerrado.
- No se sabía cómo se establecía la comunicación entre el circuito SSC-32 y el PC
- Al enviar instrucciones al robot, con el programa hecho en Labview, este no las ejecutaba.
- El software comercial RIOS no enviaba tramas por el puerto serie si no estaba conectado el circuito SSC-32.
- Como realizar una conversión de etiquetas a instrucciones SSC-32
- Los cables se enganchan en los servos en determinadas puntos moviéndose el robot desde el reposo.
- Existe un offset en la posición que hace que el robot no se situé exactamente sobre el punto, dependiendo si para alcanzarlo se mueve en sentido horario o anti horario.

##### Soluciones:

- Desarrollo de un programa de control propio
- Consulta en foros de lynxmotion y ARDE y lectura del manual del circuito SSC-32.
- Análisis de las tramas enviadas por el puerto serie mediante el software ComDebug.
- Al no poder analizar las tramas que enviaba el software RIOS, se optó por la utilización de temporizadores para realizar la secuenciación de operaciones.
- Utilización de técnicas de *hashing*
- Por defecto al iniciar un programa el robot se sitúa en la posición "PosC3REP1H"
- Hacer que el robot alcance siempre los puntos mediante un giro anti horario del servo 0, esto se refleja en el programa realizado para tareas de PPO por los alumnos.

#### 4.4 MÓDULO 4 – VERIFICADOR

##### 4.4.1 Descripción del módulo

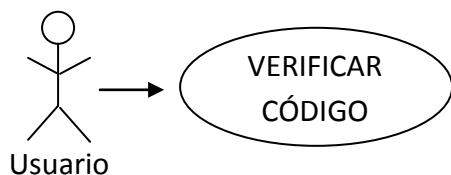
Este módulo consiste en una aplicación desarrollada en lenguaje C que tiene la función de analizar el archivo de texto que le introduce el alumno e indica a través de un mensaje en la pantalla si el programa que contiene el archivo es sintácticamente correcto o no, indicando en caso de no serlo, la línea que contiene el fallo, además del tipo de error.

Para consultar el código de este módulo ver archivo “verificador.cpp” incluido en el CD.

Funciones internas:

- **hash(char pos[]):** Esta función recibe una cadena de texto que representa una etiqueta y se encarga de realizar la búsqueda de la existencia dicha etiqueta dentro del vector PosET, si la encuentra retorna el índice donde se encuentra dicha etiqueta si no devuelve el valor - 1.

#### DIAGRAMA DE CASOS DE USO



#### CASOS DE USO

<b>Nombre:</b>	Verificar Código
<b>Autor:</b>	Ramón González
<b>Fecha:</b>	01/04/2009
<b>Descripción:</b>	Permite verificar la sintaxis del código de un programa de PPO.
<b>Precondiciones:</b>	Tener el archivo a analizar en la misma carpeta que el ejecutable “verificador.exe”
<b>Flujo Normal:</b>	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. 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:**

3. La aplicación verifica la existencia del archivo, si el archivo no se encuentra dentro de la carpeta de la aplicación muestra un mensaje de error por pantalla.

**Poscondiciones:**

Se ha verificado el archivo e informado al actor del resultado.

#### 4.4.2 Entradas y salidas

**Entradas:** - Archivo de texto en formato "TXT".

**Salidas:** - Mensaje por pantalla indicando si la sintaxis del archivo es correcta o no.

#### 4.4.3 Tecnologías utilizadas

Para el desarrollo de este módulo se ha hecho servir:

- Compilador de lenguaje C "MinGW Developer Studio"

#### 4.4.4 Tiempo de desarrollo

Para llevar a cabo el desarrollo de este módulo se han invertido un total de 6 horas.

#### 4.4.5 Implementación

La implementación de este módulo ha resultado menos complicada de lo que podría haber sido, ya que gran parte del trabajo ya se había realizado durante la implementación del Módulo 3 (Secuenciador de operaciones), como son la apertura del archivo, la lectura del mismo, etc.

La implementación de este módulo está más centrada en analizar la sintaxis de las líneas leídas desde el archivo para verificar que cumplen las siguientes reglas:

- La etiqueta de posición debe existir en el vector VectET
- La etiqueta de posición y el tiempo deben ir separados por una ",",
- No puede haber espacios ni antes ni después de la ",",
- Al finalizar cada instrucción se debe incluir el carácter ":",
- No puede haber un espacio antes del carácter ":",

- El tiempo máximo que puede indicarse para cada instrucción es de 3000 ms.
- El tiempo mínimo que puede indicarse para cada instrucción es de 500 ms.
- El tiempo máximo que puede ejecutarse un retardo entre instrucciones es de 5000ms.
- Debe añadirse la instrucción "END;" al finalizar el código del programa.

Para realizar dichas verificaciones tan solo se han incluido en el código ciertas condiciones utilizando la instrucción "if" del conjunto de sentencias condicionales, para determinar, mediante el análisis de las instrucciones leídas desde el archivo, si se cumplían las reglas que han sido listadas anteriormente.

#### **4.4.6 Problemas y soluciones**

Este módulo no alberga la existencia de problemas a los cuales hacer referencia.



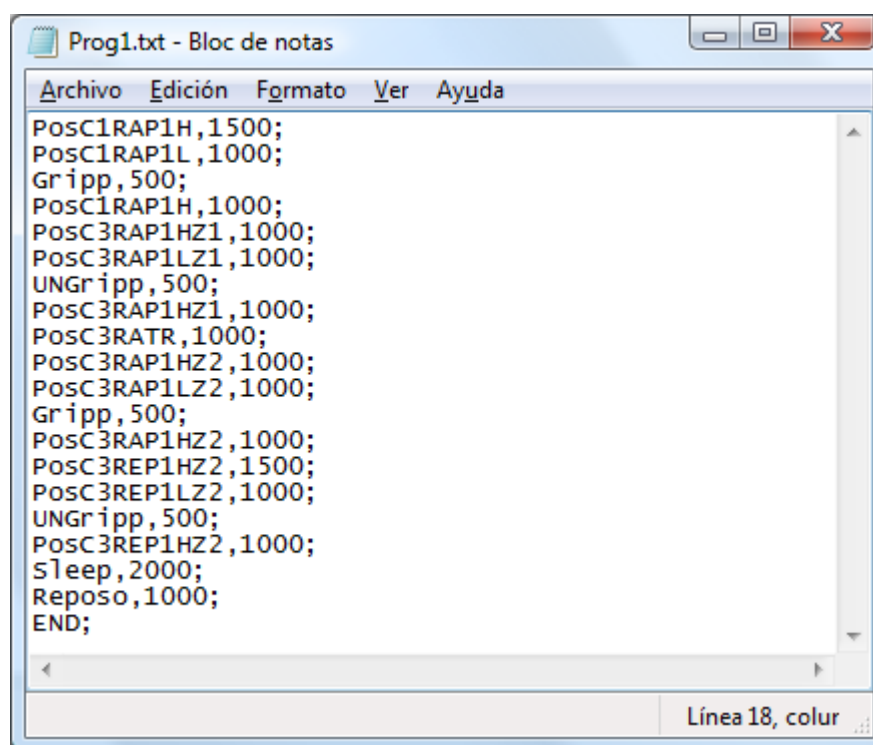
## 4.5 MÓDULO 5 – ARCHIVO

En este módulo tan solo se explicará la descripción del mismo, ya que aunque es un módulo que forma parte del sistema, no ha habido una implementación formal, ya que los archivos creados eran tan solo para las pruebas y test de otros módulos.

### 4.5.1 Descripción del módulo

Este módulo corresponde al archivo que deberá generar el alumno, en las prácticas de la asignatura Robótica y Automatización Industrial, para que el robot realice las operaciones PPO que se indiquen en el guión de prácticas.

La estructura de las instrucciones y el formato del archivo se han explicado detalladamente en los módulos 3 y 4, secuenciador de operaciones y verificador respectivamente, de modo que en este apartado, tan solo se mostrará un pequeño ejemplo de lo que sería un breve programa que podría ejecutarse en el robot. (Ver Fig. 54)



```
Prog1.txt - Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
PosC1RAP1H,1500;
PosC1RAP1L,1000;
Gripp,500;
PosC1RAP1H,1000;
PosC3RAP1HZ1,1000;
PosC3RAP1LZ1,1000;
UNGripp,500;
PosC3RAP1HZ1,1000;
PosC3RATR,1000;
PosC3RAP1HZ2,1000;
PosC3RAP1LZ2,1000;
Gripp,500;
PosC3RAP1HZ2,1000;
PosC3REP1HZ2,1500;
PosC3REP1LZ2,1000;
UNGripp,500;
PosC3REP1HZ2,1000;
Sleep,2000;
Reposo,1000;
END;
Línea 18, colur
```

**Fig. 54 – Ejemplo de programa de PPO**

Lo que realiza este programa es, coger el objeto que hay en la posición “PosC1RAP1L” y lo deja en la posición “PosC3REP1LZ2”, después espera 2 segundos antes de volver a la posición de reposo. El resto de instrucciones son para evitar colisiones con el objeto o para situar correctamente el objeto al cambiar de zona.

## 4.6 MÓDULO 6 – EDITOR DE TEXTO

Al igual que en el módulo anterior, este módulo no requiere de una implementación, por tanto tan solo se describirán los apartados descripción y entradas y salidas, omitiendo el resto de apartados generales.

### 4.6.1 Descripción del módulo

Este módulo corresponde al software necesario para generar el archivo de texto que contendrá el programa que se ejecutará sobre el robot.

Hay que diferenciar que se hace referencia a editores de texto y no a procesadores de texto, ya que estos últimos incluyen caracteres especiales no visibles al usuario, y que provocan que el verificador (módulo 4) no acepte el archivo como válido.

Entre algunos de los editores de texto que pueden hacerse servir, se encuentran:



Bloc de Notas de Windows



Editor de MS-DOS



Notepad++



GVim (editor de Linux)

### 4.6.2 Entradas y salidas

**Entradas:** - Caracteres ASCII procedentes del teclado

**Salidas:** - Archivo en formato de texto (extensión .txt)

## 4.7 MÓDULO 7 – SERVIDOR

### 4.7.1 Descripción del módulo

Este módulo corresponde al montaje del servidor web, que es el que se encarga de ofrecer accesibilidad a las páginas web mediante el protocolo http (del inglés *HyperText Transfer Protocol*) utilizando el puerto 80 del PC. Además este servidor incorpora un intérprete de PHP, para poder generar páginas web de forma dinámica.

### 4.7.2 Entradas y salidas

**Entradas:** - Páginas web en formato PHP

**Salidas:** - Código HTML mediante el código PHP interpretado

### 4.7.3 Tecnologías utilizadas

Para el desarrollo de este módulo se ha hecho servir:

- Servidor Web + PHP: Appserv 2.6.0 (incluidos en el paquete Apache 2.2.8 y PHP 6.0.0 – dev)
- PC AMD Athlon XP 1700+ con 256 MB de RAM y disco duro de 24.5 GB. bajo Windows 2000 SP4

### 4.7.4 Tiempo de desarrollo

Para llevar a cabo el desarrollo de este módulo se han invertido un total de 13 horas.

### 4.7.5 Implementación

Para el desarrollo de este módulo se pueden distinguir tres etapas:

- Instalación del servidor
- Acceso al servidor
- Configuración de directivas de seguridad

#### 4.7.5.1 Instalación del servidor

En este módulo lo que se quería conseguir era poder publicar páginas web, así que se pensó que Apache sería una buena opción, ya que es de código abierto uno de los requisitos de todo el software empleado en este proyecto, y además es uno de los más utilizados.

Habiendo decidido cuál sería el servidor que se instalaría, se pasó a decidir qué tipo de páginas web albergaría el servidor. Para la funcionalidad que debían tener las páginas web, se requería un grado de dinamismo en las mismas, así que se pensó que las páginas estarían codificadas en PHP. Este punto llevó al planteamiento que tan solo con Apache no sería suficiente, ya que apache no interpreta código PHP, así que se procedió a buscar un intérprete de PHP para que trabajara conjuntamente con Apache y de este modo poder albergar páginas codificadas en PHP en el servidor.

En un primer momento se busco en Internet un servidor Apache y un intérprete de PHP, las últimas versiones que se encontraron fueron Apache 2.2.11 y Php 5.2.9-1 respectivamente. Una vez descargados se procedió a su instalación y configuración, para ello se consultaron diversos tutoriales y foros en de Internet, pero por desgracia no se consiguió hacer funcionar los dos programas de forma conjunta. Viendo esta situación, se decidió buscar una alternativa.

Tras una búsqueda intensiva por Internet, se encontró un paquete que incluía un servidor Apache, un intérprete de PHP y un gestor de BDD (Base de Datos) de MySQL, este paquete se llama Appserv 2.6 (Enlace 19). En principio no se necesitaba el gestor de BDD para lo que se pretendía hacer, así que se estuvo probando dicho paquete y se observó que era posible elegir que componentes se instalaban y cuáles no, de modo que se aceptó el uso de dicho paquete para la instalación y configuración del servidor Apache con PHP.

Hay que decir que es muy sencillo de instalar, ya que no requiere configuración y se instala como cualquier aplicación, además esto facilita la migración del servidor a otra máquina, en el caso de que se decidiera cambiar de ubicación.

#### 4.7.5.2 Acceso al servidor

Una vez se tenía todo el software disponible, se realizó una petición al servicio informático de la ETSE, para que se asignara un nombre de dominio o subdominio a la maquina en que se instalaría el servidor, para poder tener acceso desde el exterior, pero al indicarles que tan solo era necesario que abrieran el puerto 80 y que el protocolo de comunicaciones sería tan solo el HTTP, ya que el diseño del sitio web estaba pensado para que en el lado del cliente las operaciones fueran las mínimas, y fuera lo más sencillo posible para agilizar la carga de las pagina, decidieron que de momento no se asignaría ningún dominio, y que para hacer referencia a la maquina se utilizase la IP fija que tenía asignada, la cual era 158.109.69.189

Accediendo al servidor de este modo, es válido para accesos desde la intranet de la UAB, pero no es posible acceder directamente desde Internet, así que mientras no haya un nombre de dominio al que hacer referencia, cualquier acceso externo se debe realizar mediante VPN (del inglés *Virtual Private Network*) o red virtual privada, de este modo se puede extender la intranet a través de Internet, o en términos más sencillos, la máquina de un usuario remoto se conectaría a la intranet con una IP que pertenece al rango de IPs de la intranet de la UAB, y de este modo hacer referencia a la IP fija que identifica el servidor y poder visitar las páginas web del laboratorio remoto.

#### 4.7.5.3 Configuración de directivas de seguridad

Teniendo ya el servidor montado y accesible, se pensó que sería necesario otorgar de cierta seguridad al mismo, así que se decidió que para acceder al sitio web debería realizarse una autenticación mediante usuario y contraseña, de este modo se restringía el acceso a cualquier usuario, y en un segundo nivel restringir el acceso vía web a ciertas carpetas del servidor. La base teórica para realizar esta parte fue adquirida en la asignatura de Redes.

Estas tareas podían realizarse de diversas maneras, pero el propio Apache ofrece una solución buena y segura, así que se optó por utilizarla.

Para las dos funcionalidades que se han nombrado, apache hace uso del archivo `.htaccess`. En dicho archivo se pueden establecer directivas de forma rápida y sencilla, de manera que tan solo afectan al directorio en que se encuentra dicho archivo y a sus subdirectorios, de este modo se evita tener que estar realizando cambios constantemente en el archivo `httpd.conf`, que es el archivo de configuración principal de Apache. Estas directivas van, desde pedir una autenticación mediante usuario y contraseña al intentar acceder a una página o carpeta, hasta redirigir el navegador al solicitar una URL (del inglés *Uniform Resource Locator*) determinada.

Antes de proceder a la creación de los archivos .htaccess, hay que tener en cuenta que hay que realizar un pequeño cambio en el archivo httpd.conf (Enlace 20), para indicar a Apache que se utilizaran los archivos .htaccess, el cambio es el siguiente:

1. Se busca la línea en la que se hace referencia al directorio local y se cambia el parámetro *AllowOverride None* por *AllowOverride All*.

```
<Directory /var/www/>  
Options Indexes FollowSymLinks MultiViews  
AllowOverride All  
  
Order allow,deny  
allow from all  
</Directory>
```

2. Se reinicia el Apache, y entonces ya se pueden hacer uso de los archivos .htaccess

### Autenticación mediante usuario y contraseña

Para la autenticación mediante usuario y contraseña se generó un archivo .htaccess. Hay que decir que en Windows no puede crearse un archivo con esa estructura de nombre y extensión, ya que como se observa el archivo carece de nombre y esto es algo que permite LINUX pero no Windows, e indica al servidor que dicho archivo es oculto y por tanto no debe mostrarse. Así que para poder crear dicho archivo, se tuvo que iniciar la consola de Windows y crear el archivo mediante comandos de DOS, una vez creado el archivo contenía el siguiente código:

```
AuthName "Acceso Restringido"  
AuthType Basic  
AuthUserFile C:/Appserv/recursos/passwd/.htpasswd  
require valid-user  
  
ErrorDocument 404 /error.php  
ErrorDocument 500 /nopass.php  
ErrorDocument 401 /nopass.php
```

Donde las cuatro primeras líneas corresponden a la autenticación de usuario y las tres últimas muestran unas páginas codificadas en PHP para determinados mensajes de error generados en el servidor.

Éste archivo .htaccess se guardó en la carpeta donde se encontraban las páginas web, de este modo al intentar acceder a cualquiera de ellas, aparecía un formulario solicitando usuario y contraseña. Pero aún faltaba un paso más, y era definir el archivo de usuarios y contraseñas.

Para crear el archivo de contraseñas, se utilizó el comando “htpasswd” ejecutado desde la consola de Windows. Para ello hubo que situarse en una carpeta no accesible desde el navegador y allí ejecutar dicho comando con la siguiente estructura: “htpasswd -c archivo usuario”, al ejecutarlo pide al usuario que introduzca dos veces una contraseña, que guarda en el archivo que se le ha indicado codificándola con una variante del algoritmo de hash MD5.

De este modo quedaba protegida la web mediante un sistema de autenticación de usuario.

### **Restricción en el acceso a determinadas carpetas**

En esta parte lo que se pretendía era, que el usuario no pudiese introducir la ruta a una carpeta del servidor en el navegador, y que se mostrara la lista de archivos de la carpeta en el navegador.

Para ellos se utilizaron dos soluciones conjuntas:

La primera fue colocar aquellas carpetas más sensibles a ser accedidas fuera del alcance del navegador, como podían ser la carpeta de usuarios, la de archivos enviados, etc. De este modo dichas carpetas tan solo son accesibles desde el código PHP del sitio web, pero no directamente desde el navegador introduciendo la ruta de acceso.

La segunda solución se tuvo que utilizar en algunas carpetas, en las que no funcionaba la primera, como eran las carpetas de imágenes y archivos *javascript*, ya que al sacarlos del alcance del navegador, no se cargaban las imágenes ni se ejecutaban las funciones *javascript*, de modo que la solución consistió en volver a hacer uso del archivo .htaccess en este caso se creó un archivo para la carpeta raíz, de modo que se restringiría el acceso directo a la misma y todas las subcarpetas, para no mostrar el contenido. El código de dichos archivos es el siguiente:

```
Options -Indexes  
ErrorDocument 403 /denegado.php
```

La primera línea indica que no está permitido mostrar el contenido a través del navegador, provocando un error de permisos con código 403, y la segunda línea captura dicho error y redirige el navegador a una página que muestra un mensaje de error personalizado.

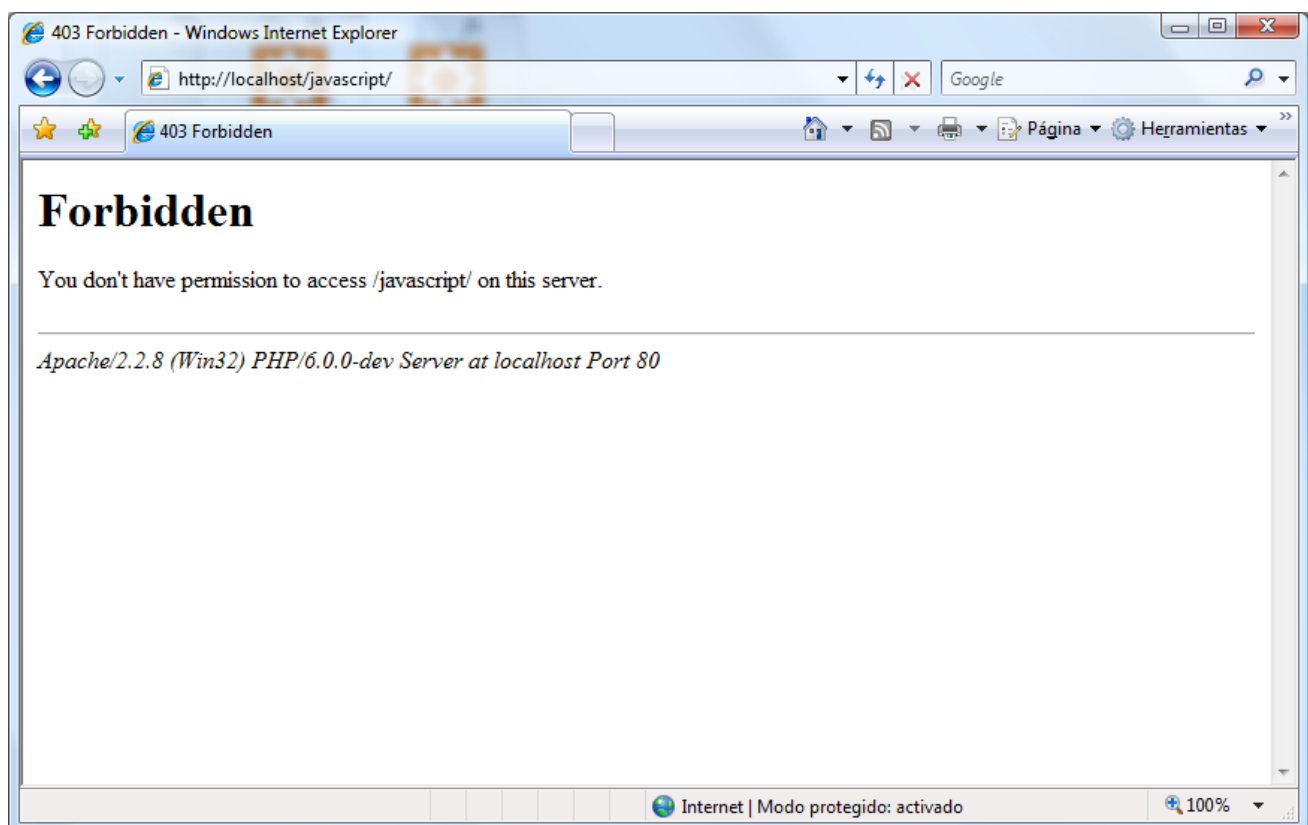
#### 4.7.5.4 Pruebas y test

Al finalizar los apartados anteriores, se realizaron las pruebas pertinentes para testear el correcto funcionamiento del sistema, y fueron las siguientes:

- Intentar acceder a una página web
- Intentar acceder a una carpeta sin protección
- Intentar acceder a una carpeta protegida
- Comprobar los mensajes de error que mostraba el servidor

En la primera prueba se verificó que, al intentar acceder a una página cualquiera que estuviera dentro de la carpeta web, ésta solicitara introducir el usuario y contraseña correctos para permitarnos visualizar la página.

En las siguientes pruebas se testeó lo siguiente comprobar lo que muestra el navegador al intentar acceder a una carpeta sin protección, al acceder a una carpeta protegida (Ver Fig. 55), y al acceder a una carpeta protegida redirigiendo el mensaje de error para personalizarlo (Ver Fig. 56)



**Fig. 55 – Acceso a un directorio del sitio web después de restringir el acceso**





Fig. 56 – Acceso a un directorio del sitio web después de configurar los mensajes de error

#### 4.7.6 Problemas y soluciones

##### Problemas:

- No se consigue configurar correctamente el servidor Apache 2.2.11 con el interprete Php 5.2.9-1
- No funcionaban las configuraciones realizadas en el archivo .htaccess
- No aceptaba las contraseñas para los usuarios creados en el archivo .htpasswd desde el navegador

##### Soluciones:

- Se instala el paquete Appserv 2.6 que instala y configura el servidor Apache junto con PHP de forma automática y en pocos minutos.
- Se modificó la línea del archivo httpd.conf donde se hace referencia al directorio local y se cambia el parámetro *AllowOverride None* por *AllowOverride All*.
- Se especificó correctamente la ruta al archivo de contraseñas .htpasswd dentro del archivo .htaccess

## 4.8 MÓDULO 8 – INTERFAZ WEB

### 4.8.1 Descripción del módulo

Éste módulo consiste, en un conjunto de páginas web que conforman la interfaz de acceso al laboratorio de robótica, dándole así la funcionalidad de acceso remoto.

Se accede por defecto a la pagina “*indice.php*” desde la cual se puede elegir la resolución de visualización de las páginas, así como el idioma. Las resoluciones disponibles son 1024x768, 1280x1024 o 1280x800, para resoluciones mayores eligiendo la de 1280 se visualiza la página completa, en cuanto a los idiomas disponibles el entorno web se encuentra realizado en castellano, catalán e inglés.

Una vez seleccionado el idioma y la resolución, el sitio web realiza una redirección a la página de *login*, desde la que se puede dar de alta un usuario, en el caso de que aun no se tenga, o si ya se dispone de uno, se puede proceder a la autenticación y posteriormente acceder a los menús de cambio de contraseña o acceso al laboratorio. Durante todo el proceso, el sistema lanza los mensajes correspondientes para mantener informado al usuario en todo momento.

Estando ya dentro del laboratorio el usuario podrá observar a través de la webcam los movimientos que está realizando el robot, también tiene la opción de enviar un archivo de texto con su programa, situándose en la cola de usuarios a la espera de ejecución. Una vez que su programa vaya a ser ejecutado, verá que su número de usuario se encuentra al principio de la lista, y la cabecera de la tabla donde se muestra la imagen de la webcam cambia a color verde.

El acceso al laboratorio está limitado a 3 usuarios simultáneos debido a los requisitos de ancho de banda, por eso las sesiones dentro del laboratorio se han limitado temporalmente, aunque el límite de usuarios es fácilmente configurable. En la misma página del laboratorio el usuario puede consultar el manual de programación, o realizar un *logout*. Transcurridos 10 minutos de inactividad, el servidor cierra la sesión y redirige al usuario a la página de *login*, permitiendo así que nuevos usuarios tengan acceso al laboratorio, este tiempo es suficiente para que los tres usuarios que se encuentran dentro del laboratorio puedan ejecutar sus programas.

En la siguiente figura se muestra la distribución real que tiene el sitio web para una resolución y un idioma (Ver Fig.128), la distribución es igual para el resto de idiomas y resoluciones. Las paginas *inici.php*, *inicio.php* e *index.php* son las que permiten cambiar entre idiomas y resoluciones.

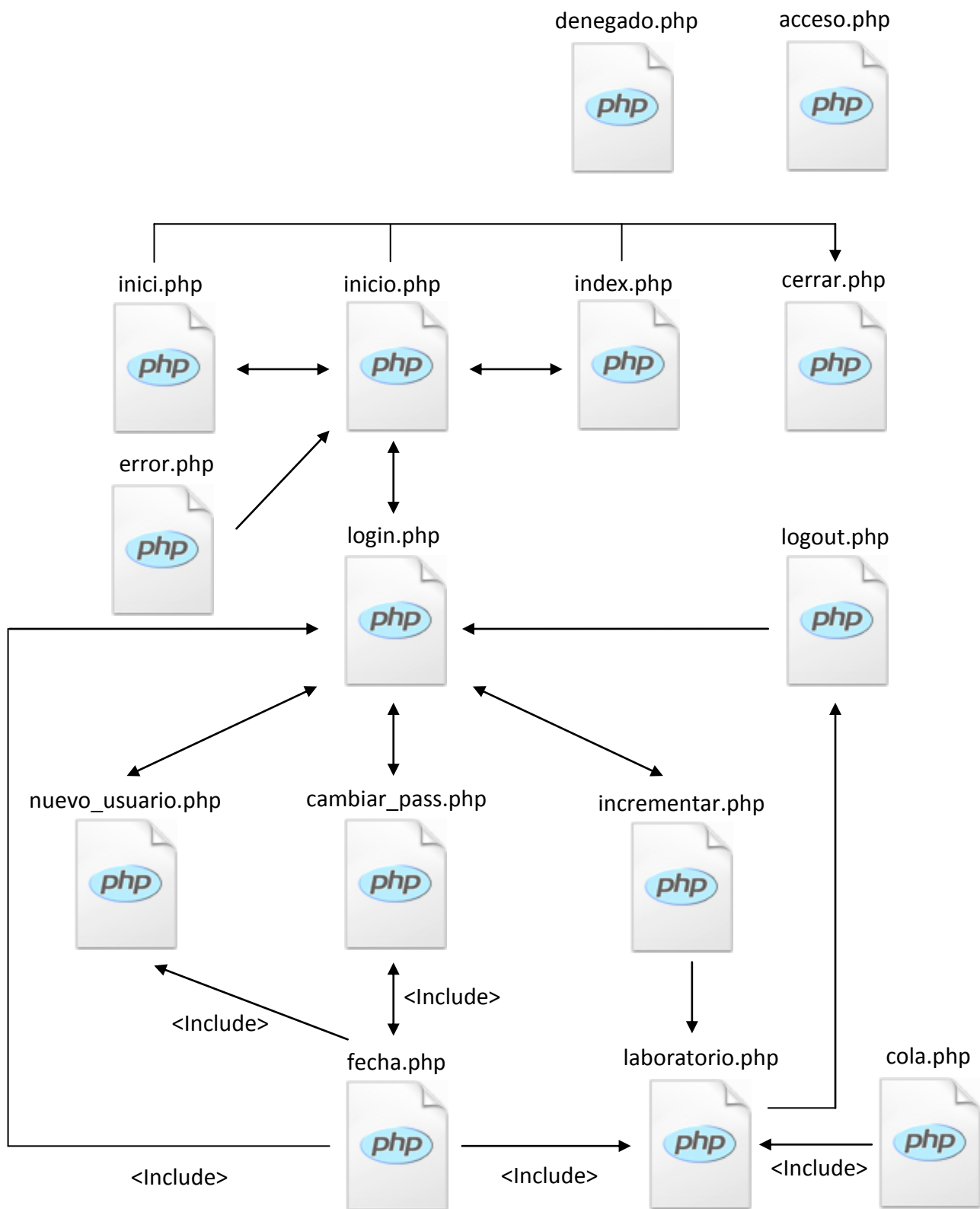


Fig. 57 – Esquema global de las páginas del sitio we

Por último se muestra la estructura que percibiría el usuario al navegar por el sitio web (Ver Fig. 58), al igual que en la figura anterior tan solo se muestra la estructura para un idioma y resolución puesto que la estructura del resto es la misma.

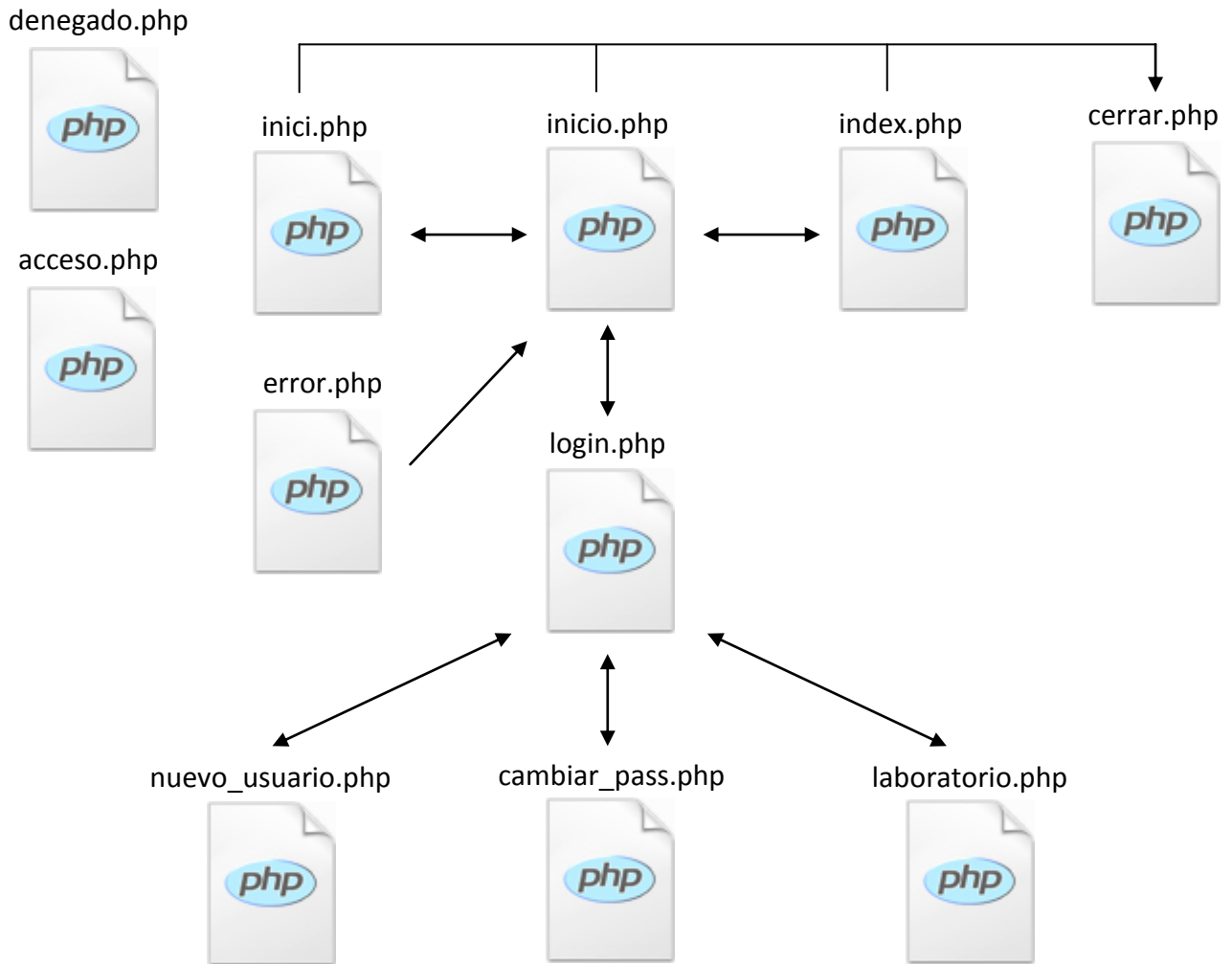
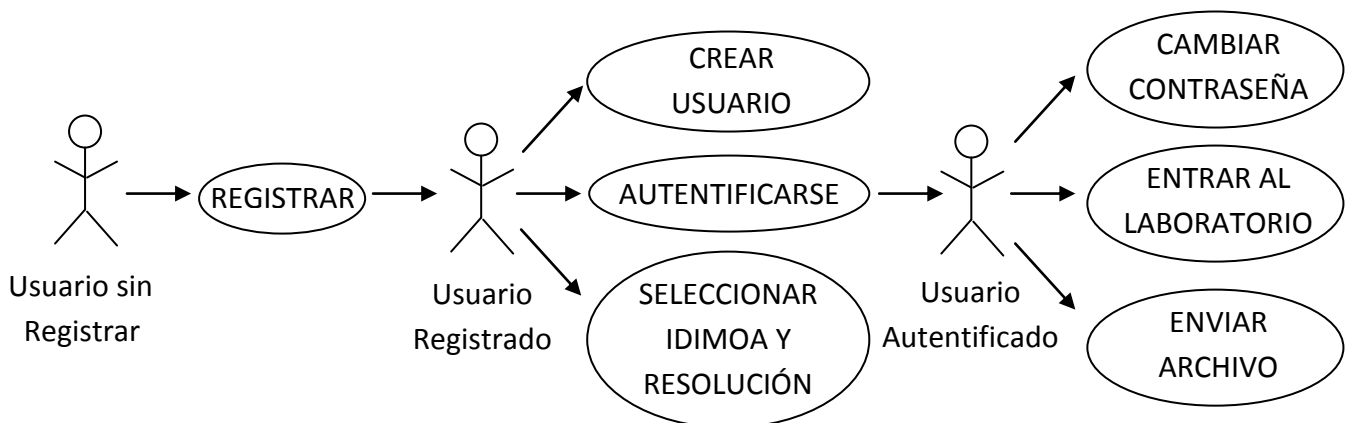


Fig. 58 – Esquema de las páginas del sitio web que percibe el usuario

**DIAGRAMA DE CASOS DE USO**



**CASOS DE USO**

<b>Nombre:</b>	Registrarse
<b>Autor:</b>	Ramón González
<b>Fecha:</b>	01/04/2009
<b>Descripción:</b> Permite obtener acceso al sitio web del laboratorio remoto de robótica.	
<b>Precondiciones:</b> Tener habilitada y operativa la conexión a Internet.	
<b>Flujo Normal:</b> <ol style="list-style-type: none"> <li>1. El actor introduce la URL en el navegador.</li> <li>2. El navegador solicita la página al servidor</li> <li>3. Se muestra por pantalla una caja de texto para introducir usuario y contraseña</li> <li>4. El actor introduce el usuario y la contraseña y pulsa el botón aceptar.</li> <li>5. El servidor verifica la validez de los datos y redirige a la página de inicio.php</li> </ol>	
<b>Flujo Alternativo:</b> <ol style="list-style-type: none"> <li>5. El servidor verifica la validez de los datos, si no son correctos tras tres intentos, redirige a la página nopass.php</li> <li>6. Se muestra un mensaje de error por pantalla.</li> <li>7. Se solicita al actor permiso para cerrar el navegador.</li> <li>8. Si el actor pulsa aceptar se cierra el navegador, si no se queda en la página de nopass.php donde no puede ir a otro sitio.</li> </ol>	
<b>Poscondiciones:</b> El actor obtiene acceso a la página inicio.php	

<b>Nombre:</b>	Elegir idioma y resolución
<b>Autor:</b>	Ramón González
<b>Fecha:</b>	01/04/2009
<b>Descripción:</b> Permite seleccionar el idioma y la resolución que tendrá el sitio web.	
<b>Precondiciones:</b> Estar registrado y tener activadas las cookies.	

**Flujo Normal:**

1. El actor pulsa sobre el icono del idioma deseado.
2. Se redirige a la página del idioma seleccionado.
3. El actor selecciona una de las resoluciones del menú desplegable y pulsa el botón validar.
4. Aparece en la pantalla en botón “Entrar” o “Enter”, según el idioma escogido.
5. El actor pulsa el botón “Entrar” – “Enter”.
6. El navegador solicita al servidor la página login.php y se muestra por pantalla.

**Flujo Alternativo:****Poscondiciones:**

El actor obtiene acceso a la página login.php, con el idioma y resolución seleccionado.

<b>Nombre:</b>	Autenticarse
<b>Autor:</b>	Ramón González
<b>Fecha:</b>	01/04/2009
<b>Descripción:</b>	Permite autenticarse en el sistema.
<b>Precondiciones:</b>	Estar registrado y haber seleccionado el idioma y la resolución.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El actor introduce su NIU y contraseña y pulsa el botón “validar”.</li> <li>2. El servidor comprueba los datos, y marca el usuario como autenticado.</li> <li>3. Se muestran las opciones “cambiar contraseña” y “acceso al laboratorio”.</li> </ol>
<b>Flujo Alternativo:</b>	<ol style="list-style-type: none"> <li>1. El servidor comprueba los datos, si los datos no son correctos, no autentifica al usuario.</li> <li>2. Se muestra por pantalla el mensaje de error correspondiente.</li> </ol>
<b>Poscondiciones:</b>	El usuario está autenticado.

<b>Nombre:</b>	Crear Usuario
<b>Autor:</b>	Ramón González
<b>Fecha:</b>	01/04/2009
<b>Descripción:</b> Permite crear un usuario en el sistema.	
<b>Precondiciones:</b> Estar registrado y haber seleccionado el idioma y la resolución.	
<b>Flujo Normal:</b> <ol style="list-style-type: none"> <li>1. El actor pulsa sobre el icono de nuevo usuario.</li> <li>2. El navegador solicita al servidor la página nuevo_usuario.php y la muestra.</li> <li>3. El actor introduce el NIU, y la contraseña dos veces y pulsa el botón “crear”</li> <li>4. El servidor comprueba la validez de los datos y crea el usuario.</li> <li>5. Se muestra un mensaje por pantalla para informar al actor.</li> </ol>	
<b>Flujo Alternativo:</b> <ol style="list-style-type: none"> <li>4. El servidor comprueba los datos, si los datos no son correctos, no crea el usuario.</li> <li>5. Se muestra por pantalla el mensaje de error correspondiente.</li> </ol>	
<b>Poscondiciones:</b> Se ha creado el usuario en el sistema.	

<b>Nombre:</b>	Cambiar contraseña
<b>Autor:</b>	Ramón González
<b>Fecha:</b>	01/04/2009
<b>Descripción:</b> Permite cambiar la contraseña de un usuario.	
<b>Precondiciones:</b> Estar autenticado.	
<b>Flujo Normal:</b> <ol style="list-style-type: none"> <li>1. El actor pulsa el botón “cambiar password”</li> <li>2. El navegador solicita la página “cambiar_pass.php” al servidor y la muestra.</li> <li>3. El actor introduce la nueva contraseña dos veces.</li> <li>4. El servidor comprueba la validez de los datos y cambia la contraseña</li> <li>5. Se muestra un mensaje por pantalla para informar al actor.</li> </ol>	

**Flujo Alternativo:**

4. El servidor comprueba los datos, si los datos no son correctos, no cambia la contraseña.
5. Se muestra por pantalla el mensaje de error correspondiente.

**Poscondiciones:**

Se cambia la contraseña del usuario autenticado.

<b>Nombre:</b>	Entrar al laboratorio
<b>Autor:</b>	Ramón González
<b>Fecha:</b>	01/04/2009
<b>Descripción:</b>	Permite entrar a la página del laboratorio remoto de robótica.
<b>Precondiciones:</b>	Estar autenticado, no haber llegado al límite de usuarios dentro del laboratorio.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El actor pulsa el botón “ir al laboratorio”</li> <li>2. El navegador solicita la página “laboratorio.php” al servidor y la muestra.</li> </ol>
<b>Flujo Alternativo:</b>	
<b>Poscondiciones:</b>	El actor obtiene acceso a la página del laboratorio remoto de robótica

<b>Nombre:</b>	Enviar archivo
<b>Autor:</b>	Ramón González
<b>Fecha:</b>	01/04/2009
<b>Descripción:</b>	Permite enviar un archivo de texto al servidor
<b>Precondiciones:</b>	Estar autenticado y estar dentro de la página del laboratorio remoto de robótica.



**Flujo Normal:**

1. El actor pulsa el botón “examinar”
2. Se muestra una ventana para explorar los archivos del disco.
3. El actor busca, selecciona el archivo y pulsa el botón aceptar.
4. La página guarda y muestra la ruta al archivo seleccionado.
5. El actor pulsa el botón “enviar”.
6. La página web verifica los datos y envía el archivo al servidor.
7. El servidor recibe el archivo y lo guarda en la carpeta “archivos”.
8. El navegador muestra el NIU del actor en la cola de usuarios.

**Flujo Alternativo:**

4. La página web verifica los datos, si no son correctos, o el archivo no es de tipo texto o si aún existe en el servidor un archivo del mismo usuario que fue enviado con anterioridad, no se envía el archivo.
5. Se muestra por pantalla el mensaje de error correspondiente.

**Poscondiciones:**

Se envía el archivo al servidor.

Para la definición de esta parte se ha hecho servir la base teórica adquirida en la asignatura Ingeniería del software.

**4.8.2 Entradas y salidas**

- Entradas:**
- Instrucciones desde el ratón.
  - Cadenas de texto desde el teclado.
  - Archivos en formato “TXT” desde los clientes
  - Información de los archivos del servidor (sesion, usuarios, etc.)

- Salidas:**
- Información visual por pantalla.
  - Archivo de texto recibido.
  - Archivos de contraseña de usuario
  - Archivo de sesión

### 4.8.3 Tecnologías utilizadas

Para el desarrollo de este módulo se ha hecho servir:

- Editor de HTML WYSING: “Nvu”
- Editor de HTML soporte para Explorer y Firefox: “Aptana Studio 1.2”
- Editor de programación multilenguajes: “Notepad ++ 5.2”
- Editor de imágenes: “Adobe Photoshop CS 8.0.1”
- Servidor web + PHP: Appserv 2.6.0

### 4.8.4 Tiempo de desarrollo

Para llevar a cabo el desarrollo de este módulo se han invertido un total de 80 horas.

### 4.8.5 Implementación

Para realizar la implementación de este módulo, inicialmente se definieron cuales serían los bloques a tratar:

- Estructura básica en HTML
- Páginas web en PHP
  - o Diseño de la estructura global
  - o Subir un archivo al servidor
  - o Gestión de colas de usuario
  - o Retransmisión de las imágenes de la webcam
  - o Control de acceso
  - o Últimos ajustes

Durante la realización de este módulo se hizo servir las bases teóricas adquiridas en las asignaturas Redes, Ampliación de Redes y Bases de Datos. (En la asignatura Bases de Datos tan solo en la parte de las prácticas de la asignatura).

También hay que decir que en un primer momento, se pensó de utilizar una BDD para gestionar los usuarios y los archivos que se enviaban al servidor, pero se pensó que era hacer uso de una herramienta muy potente para la funcionalidad que se requería, y al ver que se podía realizar la gestión mediante archivos y carpetas se descartó el uso de una BDD, ya que esta alternativa era más sencilla y evitaba los posibles errores que pudieran surgir de la interacción con la BDD, además que simplificaba el mantenimiento del sistema.

#### 4.8.5.1 Estructura básica en HTML

En este apartado, se definió cual sería la estructura o apariencia de la página web que sería la del laboratorio remoto, para ello no era necesario codificar aún en PHP, tan solo con código HTML ya se podía dar una primera forma a la página.

Como lo que se pretendía hacer eran una serie de pruebas, para ver cuál era la mejor forma de estructurar la página web del laboratorio remoto, se pensó que sería buena idea utilizar un editor web basado en WYSING (*What You See Is What You Get*) o traducido del inglés, que sería: lo que ves es lo que obtienes, que consiste en un editor web gráfico mediante el cual, es posible ir construyendo la web de forma rápida sin escribir código. Tan solo insertando una tabla, una imagen o escribiendo texto, el editor se encarga de escribir el código html que corresponde a lo que se está haciendo.

Para este cometido se utilizó un editor web WYSING de código abierto llamado Nvu 1.0 (Enlace 21).

Tras hacer los primeros diseños se vio que, crear una página web de este modo tenía ciertas limitaciones que no tenían los editores de código, como podían ser la definición de CSS (*Cascading Style Sheets*), o escribir código PHP en cualquier parte del código HTML, aunque este editor también podía cambiarse al modo código, al pasar del modo código al modo gráfico para observar como quedaba la página, y posteriormente volver al modo código para continuar con el diseño, se perdían líneas de código o, si quería importarse un trozo de código de otra página que contenía algún script o código PHP, el editor NVU los eliminaba al pasar al modo gráfico. Este hecho provocó la búsqueda de un segundo editor también de código abierto y que no presentase dichos inconvenientes.

Como además se pretendía que el sitio web fuese compatible con los navegadores Explorer y Firefox, se aprovechó para buscar un editor que permitiese la pre-visualización de la página web en ambos navegadores. Tras la búsqueda por Internet el editor que se escogió fue Aptana Studio 1.2.5 (Enlace 22), ya que cumplía todos los requisitos que se habían prefijado.

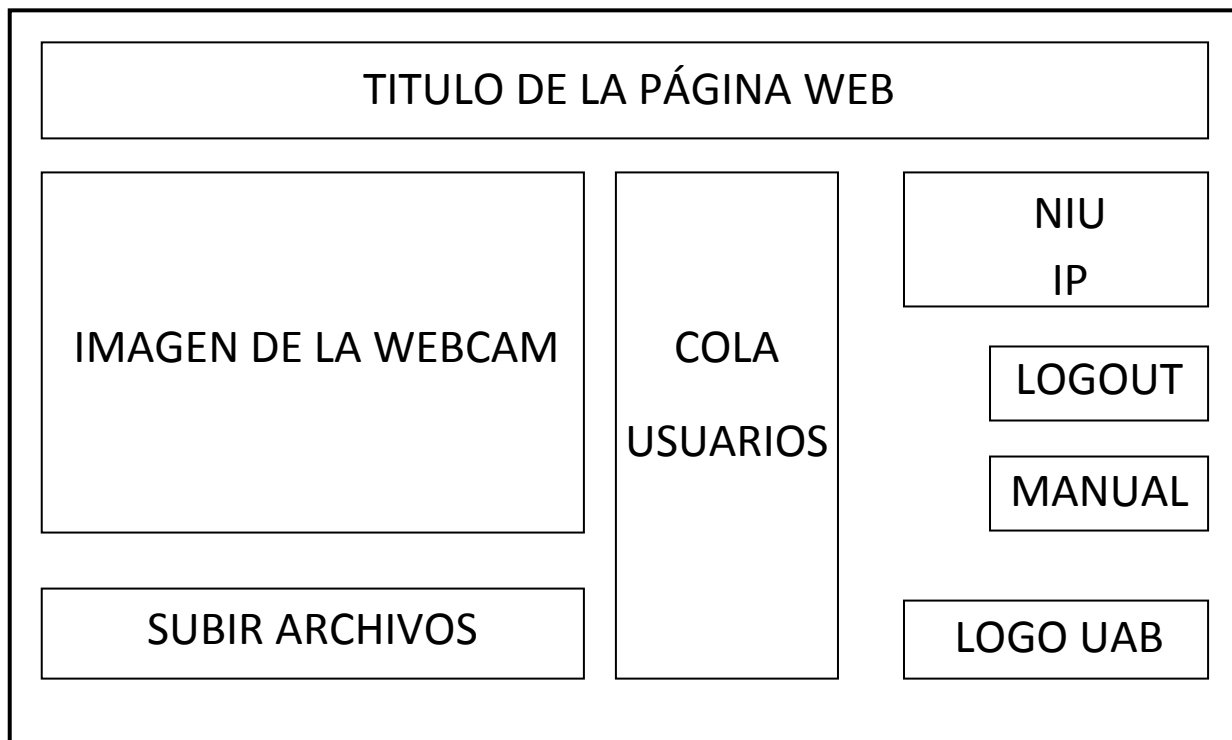
Teniendo estos dos editores instalados, se pudo hacer el diseño principal de lo que sería la página web del laboratorio remoto, que se fue codificando mediante el editor Aptana Studio, y se iba comprobando que todo lo que se incluía se visualizaba correctamente, tanto en Explorer como en Firefox. De este modo se encontraron casos en que por ejemplo, los "fieldset" (cuadros con título) o la definición de divs con posición "Relative" (posición relativa de una división respecto a los bordes de la ventana del navegador), no eran soportados por Firefox, y obligaba a buscar otra solución que fuera aceptada por ambos navegadores.

Hay que decir que el editor Nvu no fue del todo inútil, ya que permitió crear la estructura más básica. También sirvió como referencia, a la hora de incluir código del que no se estaba muy

seguro de cuál era su estructura. Por ejemplo, si se quería crear una tabla que tuviese celdas combinadas, podía hacerse la tabla gráficamente en Nvu, y luego consultar el código que generaba el editor, para utilizarlo posteriormente en Aptana Studio.

En la implementación de este apartado también fue muy útil un manual online de HTML (Enlace 23)

Al finalizar este apartado se tenía una estructura como la que se observa en la siguiente figura (Ver Fig. 59)



**Fig. 59 – Esquema de la distribución de la página laboratorio.php**

#### 4.8.5.2 Páginas web en PHP

Teniendo definida cual iba a ser la estructura y apariencia de la web, ahora se tenía que dotar de la funcionalidad necesaria a las páginas web, para que la vertiente remota del laboratorio fuera posible. Para ello se decidió que las páginas se codificarían con lenguaje PHP, el cual permite generar contenidos en HTML de forma dinámica, y al ser un lenguaje del lado del servidor, permite realizar una serie de acciones sobre el mismo servidor que serían de gran utilidad.

En este punto se vio que el editor web Aptana Studio no soportaba código PHP, por lo que se optó por buscar un editor que sí que lo soportara, refiriéndonos a la posibilidad de resaltar con colores las partes básicas del código y poder ocultar trozos del mismo como podrían ser sentencias if, un div, etc., haciendo más amena la codificación de las páginas.

El editor elegido para esta tarea fue Notepad++ 5.3.1 (Enlace 24), que es un editor multilenguaje de código abierto, que permite programar en multitud de lenguajes.

Para la visualización de las páginas web ya era necesario alojarlas en el servidor y visualizarlas desde los navegadores Explorer y Firefox, ya que era necesario el interprete de PHP del servidor para que pudieran ser visualizadas completamente y con todas sus funciones.

### **Diseño de la estructura global**

Para definir la estructura de páginas que tendría el sitio web, se pensó que sería buena idea partir desde la página del laboratorio remoto y a partir de ahí empezar a añadir paginas en base a las funciones que se fueran añadiendo.

De este modo se comenzó con una página llamada "laboratorio.php", como se quería controlar el acceso debería haber una página de *login* "login.php" y por consiguiente otra de *logout* "logout.php", además debía haber la posibilidad de crear un usuario con su correspondiente contraseña, en caso de que no existiese "nuevo\_usuario.php", al poder crear usuarios con contraseña estos también debían tener la posibilidad de cambiar su contraseña de acceso "cambiar\_pass.php" y como por limitaciones del ancho de banda quería limitarse el acceso al laboratorio a partir de cierto número de usuarios, hubo que crear una página que guardara el número de usuarios conectados al laboratorio "incrementar.php".

Después de tener claras que estas páginas eran necesarias, se pensó que además el sitio debería estar en diversos idiomas, en castellano y catalán, puesto que son las lenguas de la región, y en inglés puesto que es el lenguaje estándar, y al haber alumnos extranjeros que estudian en la UAB mediante becas Erasmus. También se pensó que los usuarios que accedieran al laboratorio remoto, podrían tener monitores con diversas resoluciones, por ello se decidió también ofrecer la posibilidad de elegir visualizar la web entre las resoluciones más comunes. Estas nuevas premisas hicieron que se generaran las páginas para seleccionar el idioma y resolución, "inicio.php", "inici.php" e "index.php"

Y por último además de estas páginas, estaban las páginas de mensajes de error del servidor, de las que ya se hablaron en el módulo 7. "denegado.php", "error.php" y "nopass.php".

La forma en que se relacionan todas estas páginas viene ilustrada en las figuras que se mostraron en el apartado "5.8.1 Descripción del módulo". (Ver Fig. 57 y Fig. 58).

### Subir un archivo al servidor

Con el diseño de la página del “laboratorio.html” realizado como base para la codificación a “laboratorio.php”, se empezó a dotando a la web de una de las funcionalidades más importantes para lograr el funcionamiento deseado, que es la de poder enviar los archivos de texto al servidor.

Se estuvieron consultando algunos tutoriales online y al final se encontró la solución en un foro de PHP (Enlace 25), donde había un ejemplo de cómo subir una imagen que no excediera de cierto tamaño mediante el protocolo http, así que lo que hubo que hacer es adaptar dicho código a las necesidades propias del laboratorio.

Los cambios realizados en el código fueron para que la función fuera la siguiente:

- a. El usuario busca un archivo en su PC para enviarlo.
- b. La web recoge la ruta de dicho archivo.
- c. Si la ruta está en blanco, envía el mensaje correspondiente.
- d. Si el usuario ha seleccionado un archivo que no es de texto, se envía el mensaje correspondiente.
- e. En el caso de que se hayan superado los 10 usuarios, envía el mensaje correspondiente.
- f. Si los puntos 2-5 eran correctos:
  - i. La página captura el NIU (Número de identificación de usuario), haciendo uso de las variables de sesión.
  - ii. Abre el directorio de archivos
  - iii. Verifica que no exista algún archivo del mismo usuario, si existiese envía el mensaje correspondiente
  - iv. Si no había ningún archivo del usuario en el directorio, la página renombra el archivo que se ha subido con el NIU de modo que se guarda como “NIU”.txt

Con estos cambios realizados, la funcionalidad de subir un archivo al servidor teniendo en cuenta una serie de restricciones como son: que aún exista un archivo del mismo usuario, o que el archivo no sea txt, etc., ya era posible.

### Gestión de colas de usuario

Llegados a este punto, los usuarios podían enviar sus archivos al servidor, teniendo la garantía que tan solo había un archivo por usuario. Ahora la necesidad era la de poder mostrar una lista ordenada de los usuarios que habían enviado archivos siguiendo el método FIFO (*First Input First Output*), de modo que el primero en enviar un archivo sería el primero de la cola.

Al tener la garantía que tan solo había un archivo por usuario y además los archivos de los usuarios se habían renombrado con el NIU, lo que se había pretendido con esto era allanar el camino, ya que ahora tan solo habría que consultar el directorio, ordenar los archivos por fecha de creación y guardar los nombres de los archivos en dicho orden en un vector, para luego ser mostrados en una tabla en la página web.

Como ya se sabía la manera de manejar archivos y directorios, puesto que se había hecho servir en a la hora de implementar el envío de un archivo, se empezó buscando la manera de ordenar los archivos por fecha, para ello se buscó información por Internet, en un primer momento se encontró un ejemplo que realizaba dicha ordenación, pero cuando se implemento y se probó se vio que la precisión tan solo era de días, es decir que archivos enviados dentro del mismo día se ordenaban por orden alfabético. Esto hizo descartar dicho ejemplo, ya que no servía para lo que se quería lograr, así que se continuó la búsqueda y se encontró otro ejemplo en una web (Enlace 26) en la que se realizaba la ordenación de archivos de un directorio del sitio web, según el criterio de la fecha de última modificación, y se iba guardando el nombre con extensión en un vector, de forma que el archivo mas nuevo quedaba el primero, en este caso la precisión en la fecha era hasta los segundos.

Teniendo dicho ejemplo al alcance, se modificó para que el criterio fuera por fecha de creación y el orden fuera inverso al del ejemplo, es decir, que el archivo más antiguo fuera el primero, y además que en el vector tan solo se guardara el nombre del archivo sin la extensión. Así teniendo la lista de archivos ordenados en un vector, se podía asignar cada posición del vector a una posición de la tabla en la página web. Para consultar el código de esta parte ver el código del archivo "cola.php" del CD.

Después de poder mostrar el contenido de una carpeta del servidor con los archivos en el orden deseado, se planteaba un nuevo problema, que era el refresco de la cola, para que cuando un usuario añadiese un archivo la cola se actualizara, al igual que si se eliminaba un archivo del servidor por ya haber sido ejecutado en el robot, que dicho cambio se reflejara en la cola de la página del laboratorio.

La primera idea que se planteó fue la de refrescar la página web cada segundo utilizando un tag en el body, pero rápidamente se vio que no sería una buena solución, ya que lo óptimo sería que tan solo se refrescara la parte de la página web que interesa. Por ello, se estuvo investigando de qué modo se podría realizar esta operación, y la solución encontrada recibe el nombre de AJAX (*Asynchronous JavaScript And XML*).

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma

es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Navegando por Internet, se encontró una página web donde había un pequeño ejemplo del uso de AJAX, mediante el uso de una librería de funciones *javascript* llamada JQuery (Enlace 27).

Basándonos en dicha técnica, se pensó que sería apropiado extraer el código correspondiente a la cola del archivo “laboratorio.php”, y generar una nueva página llamada “cola.php”, de modo que en la página “laboratorio.php” se incluyera el contenido de la página “cola.php” dentro de un “div” mediante la función <include>. De este modo se adaptó el ejemplo encontrado en Internet y se utilizó para llevar a cabo el refresco del div, usando además unas funciones *javascript* para definir el intervalo de refresco.

### **Retransmisión de las imágenes de la webcam**

Llegados a este punto la página del laboratorio, ya era capaz de enviar archivos al servidor para ser ejecutados, y mostrar una lista de los usuarios que había en cola. Ahora el próximo paso era poder mostrar en la página del laboratorio, las imágenes guardadas por el servidor de la webcam.

Como ya se conocía la existencia de AJAX, se hizo uso de esta técnica y mediante una función *javascript* que define el intervalo de refresco, se iba realizando el refresco de la imagen en la página web. Se encontró un ejemplo en un foro (Enlace 28), que fue de utilidad. Para consultar el código de esta parte ver el archivo “laboratorio.php”, en concreto las funciones *javascript* `principal()` y `recargarImg()`.

El resultado del desarrollo de esta parte puede verse en la figura que se mostrará a continuación. (Ver Fig. 60)

Habiendo implementado esta funcionalidad de este modo, se desvincula a la página web del tipo de software que realice la captura de imágenes, ya que la página web tan solo se encarga de recoger la imagen que va sobrescribiendo el servidor de la webcam, en el intervalo que definimos en la función *javascript* cuya resolución es de milisegundos. Hay que indicar que la web se ha dejado preparada para poder retransmitir desde dos webcam.

Una vez finalizada esta parte, se realizaron unas modificaciones en el archivo “cola.php”, para que detectar que archivo que se está ejecutando, corresponde al usuario que está observando desde el laboratorio remoto, cambie al color verde de la cabecera de la tabla que muestra las imágenes de la web, para darle un aviso visual.





Fig. 60 – Página web laboratorio.php

### Control de acceso

Para finalizar este módulo, faltaba por poder controlar el acceso de usuarios al laboratorio, para saber quien enviaba los archivos al servidor.

Con la página del laboratorio terminada, se inició la creación de la página “login.php”, cuyo objetivo era el de poder realizar una autenticación de usuario, para ofrecer a posteriori acceso al laboratorio.

Antes de comenzar, se decidió que se mantendría una distribución de los elementos dentro de la página, lo más parecida a la que tenía la página del laboratorio, de modo que al cambiar de una página a otra, el usuario percibiera pocos cambios a la hora de cargarlas. Además, de este modo se agilizaría un poco la carga, puesto que muchos de los elementos que no varían, ya se encontrarían en la memoria caché y no tendrían que ser cargados.

Se empezó incluyendo en la página un formulario para introducir el NIU y la contraseña, en el mismo formulario se crearon unos campos ocultos con un NIU y contraseña prefijados, para poder realizar pruebas y verificar que, a no ser que se introdujesen dicho NIU y contraseña en el formulario, la página no daba acceso al laboratorio.

Como ya se había dicho, no se iba a utilizar una base de datos en el proyecto, puesto que era una herramienta muy potente para el uso que se le iba a dar, por esto y puesto que ya se sabían manejar archivos desde código php, se decidió implementar la parte del control de usuarios mediante archivos. Una vez que el control de acceso funcionaba, se dio el siguiente paso, que era leer la información del NIU y contraseña desde una carpeta de usuarios, en vez de una entrada oculta del formulario. Entonces se decidió que cada usuario dado de alta en el sistema sería un archivo, el nombre del cual sería el NIU y el contenido sería la contraseña.

En este momento se creó un archivo llamado “2108463” y el contenido del cual era “Pepito” dentro de la carpeta usuarios, la cual estaba fuera del alcance del navegador, por los motivos de seguridad explicados en el módulo 7 – Servidor, de modo que una vez modificado el código php para que se consultaran los datos accediendo a los archivos (archivo “login.php”, función acceso()), introduciendo el NIU “2108463” y la contraseña “Pepito” obteníamos acceso al laboratorio. En esta parte cuando el usuario pulsa el botón validar del formulario de acceso, se realizan las verificaciones pertinentes, como son: que se haya introducido un usuario al igual que una contraseña, que el usuario exista, etc., y en cualquier caso se muestran los mensajes pertinentes para mantener informado al usuario.

Al finalizar esta parte se detectó un error. Era necesario enviar el NIU a la página “laboratorio.php”, pero no se quería realizar mediante el método GET, el cual toma los parámetros desde la URL, ya que no es un método muy seguro por mostrar la información que se desea enviar al usuario, así que se decidió hacer uso del método POST, el cual realiza el traspaso de parámetros de forma transparente al usuario, de modo que el NIU se guardaba en un input oculto de un formulario y se recuperaba mediante el método POST cuando era necesario. Pero se observó que cada vez que pasaba de un formulario a otro, se le añadían unos caracteres invisibles al final, los cuales tan solo se podían detectar si se verificaba la longitud de la cadena. Este hecho causó muchos problemas y no se ha podido explicar que lo causaba, pero si se encontraron un par de soluciones:

La primera solución consistía en aplicar una expresión regular que eliminaba cualquier carácter no alfanumérico de la variable NIU cada vez que se obtenía mediante el método POST.

La segunda solución consistía en iniciar una sesión al entrar en el sitio web, de modo que se podían declarar variables de sesión, las cuales pueden consultarse desde cualquier página mientras se mantenga abierta la sesión, es decir, mientras no se cierre el navegador y se corte la conexión con el servidor.

Las dos soluciones anteriores se probaron y funcionaron correctamente, pero de las dos, se escogió la segunda, puesto que era más sencilla, ahorra líneas de código y además nos permitiría dar otro nivel de seguridad al sitio web, puesto que con dichas variables de sesión podía

restringirse el acceso directo a una página desde el navegador, en caso de no estar autenticado en el sistema, pudiendo redirigir el navegador a la página de *login* para accesos no autorizados.

Habiendo conseguido la autenticación de un usuario desde un archivo, se creyó oportuno ofrecer la posibilidad de crear usuarios desde la página web, en lugar de realizarlo de modo manual, así los usuarios que accedieran al sistema, los cuales ya tendrían ciertos permisos, puesto que habrían superado la autenticación del servidor, podrían crearse un usuario personalizado con su NIU y contraseña, descargando de esta tarea al administrador del laboratorio (profesor), el cual tan solo debería eliminar los usuarios una vez finalizado el año académico, que correspondería a eliminar los archivos de la carpeta usuarios. Por motivos de seguridad se estimó que sería el administrador el único con permisos para eliminar usuarios, los usuarios comunes tan solo pueden crear usuarios y modificar sus propias contraseñas.

En este paso se creó la página “nuevo\_usuario.php”, manteniendo el formato y la estructura homogénea con las otras páginas como se había indicado anteriormente. La página solicita al usuario, que introduzca el NIU del usuario que desea crear y la contraseña desea asociar a dicho NIU por duplicado. Una vez pulsado el botón de validación del formulario, se realizan las comprobaciones correspondientes: que se hayan introducido tanto el NIU como dos veces la contraseña, que el NIU no exista en el sistema, que el NIU tan solo contenga caracteres alfanuméricos, que las dos contraseñas coincidan, etc., además de mostrar los mensajes pertinentes en cada caso.

Si el NIU y contraseñas superan las comprobaciones anteriores, se procede a la creación del usuario haciendo uso del código php para crear un archivo (Enlace 29) (Ver CD: archivo “nuevo\_usuario.php”, funcion guardar\_pass()).

Para dotar de una mayor seguridad al sistema, se decidió que habiendo finalizado esta parte, sería conveniente hacer uso de un algoritmo hash a la hora de guardar las contraseñas.

La función que se le va a dar al algoritmo de hash, es la de guardar la cadena resultante al aplicar el algoritmo de hash con la contraseña como entrada, en lugar de guardar la propia contraseña. Posteriormente al introducir una contraseña en el sistema esta no se compara directamente con la que hay almacenada, ya que no coincidirían, en su lugar se aplica el algoritmo hash a la contraseña introducida, y el resultado obtenido es el que se compara con el almacenado.

Una función hash tiene las siguientes propiedades:

1. Todos los hashes generados con una función de hash tienen el mismo tamaño, sea cual sea el mensaje (contraseña) utilizado como entrada.
2. Dado un mensaje, es fácil y rápido mediante un ordenador, calcular su hash.
3. Es imposible reconstruir el mensaje original a partir de su hash
4. Es imposible generar un mensaje con un hash determinado

Sabiendo esto, se investigó como utilizar dichas funciones de hash. En dicha investigación se observó que no hay un único algoritmo para generar un hash, así que además había que escoger un algoritmo apropiado, para ello se consultó una página web (Enlace 30) donde analizaban dos algoritmos de hash que además eran soportados por PHP, que son el MD5 (del inglés *Message-Digest Algorithm 5* o Algoritmo de Firma de Mensajes 5) y el SHA-1 (del inglés *Secure Hash Algorithm 1* o Algoritmo de Hash Seguro 1). Así que tras consultar la información de la página web, se concluyó que el algoritmo SHA-1 era más seguro, y por este motivo fue el que se eligió para ser aplicado.

Con esto decidido, se realizaron los cambios tanto en la página “nuevo\_usuario.php” como en la página “login.php”, para poder almacenar las contraseñas de forma segura y que la autenticación de usuario continuase funcionando correctamente.

El próximo paso fue crear la página “cambiar\_pass.php”, dicha página tiene como función la de cambiar la contraseña de un usuario ya existente. Esta página no generó complicaciones, puesto que las partes complicadas, que eran la de crear el usuario y generar el hash, ya estaban creadas. A esta página se accede desde la página de *login* después que un usuario se ha autenticado correctamente, y una vez en ella se solicita que introduzca dos veces la nueva contraseña, tras realizar las comprobaciones correspondientes y lanzar los mensajes pertinentes al usuario, la página actualiza la contraseña de dicho usuario.

Para concluir la parte del control de acceso, se dotó a la página web de una última funcionalidad, que era la de restringir el acceso al laboratorio para un número limitado de usuarios. El motivo de realizar tal restricción, tiene que ver con el ancho de banda disponible, que al ser limitado provoca a su vez que el número de usuarios conectados también sea limitado, ya que el servidor debe retransmitir las imágenes de la webcam para cada usuario dentro del laboratorio.

Para implementar esta parte, primero se pensó como debería restringirse el acceso al laboratorio, y lo que se planteó fue que a la hora de realizar la autenticación si se había superado el máximo de usuarios en el laboratorio, que no apareciese el botón de acceso al laboratorio y se mostrase el mensaje correspondiente. También se pensó que para llevar el control de los usuarios que había dentro del laboratorio, se haría servir un archivo que haría la función de contador y otro archivo que definiría el máximo de usuarios que podrían acceder al laboratorio, de este modo no sería necesario tocar el código de las páginas si se quería variar dicho límite. Ambos archivos estarían en una carpeta llamada sesión fuera del alcance del navegador.

Haciendo uso de todas estas ideas, estaba definida la forma en que debería funcionar la página, y que era la siguiente:

Un usuario se autentifica correctamente, en ese momento la página de *login* accede al archivo de sesión para verificar cuantos usuarios hay dentro, luego accede al archivo de máximo de usuarios

para verificar si se ha llegado al límite, en caso afirmativo muestra el mensaje correspondiente no dando opción de acceso al laboratorio, también se muestran los mensajes correspondientes en caso de que se haya borrado el archivo de limite de usuarios bloqueando el acceso, en caso negativo aún no se habrá llegado al límite de usuarios, por lo tanto el acceso al laboratorio está permitido (Ver Fig. 61), justo antes de acceder al laboratorio se incrementa el número del archivo sesión mediante la página “incrementar.php”, que no es visible al usuario, ya que es inmediatamente redirigido a la página “laboratorio.php”. Al entrar dentro del laboratorio se activa un contador de 10 min, que es el tiempo máximo que estará activa la sesión dentro del laboratorio, una vez transcurrido ese tiempo, el navegador redirige automáticamente al usuario a la página de *login* pasando por la página “logout.php”, que se encarga de cerrar la sesión y decrementar el número del archivo sesión. También se cierra la sesión mediante este proceso si, se pulsa el botón de *logout*, se actualiza el navegador, se cambia de página o se cierra el navegador. Esto es posible gracias al evento *javascript* `onUnload` y `onBeforeUnload()`.



Fig. 61 – Página web login.php después de autenticar sin superar límite usuarios

Dichos eventos se activan en cualquiera de las situaciones anteriormente descritas, e indican que se está abandonando la página actual, incluso si es por una recarga.

De los dos eventos se utilizó el segundo “`onBeforeUnload()`”, puesto que el primero tan solo funciona en Internet Explorer, y el segundo además también es soportado por Firefox, y como se ha comentado anteriormente, se buscaba una compatibilidad en ambos navegadores.

Este evento es útil y necesario puesto que hace que detectemos cualquier salida de la página que no sea mediante el botón de *logout*, de modo que si no se controlaran este tipo de salidas, que además son las más habituales por los usuarios, como puede ser la de cerrar directamente el navegador sin cerrar la sesión, tendríamos el archivo de sesión con un número de usuarios que no se correspondería con el real, ya que se incrementaría el número de usuarios al entrar, pero nunca se decrementaría el usuario que saliese de forma anormal.

Así que se implementó dicha funcionalidad en el sitio web, pero al realizar diversas pruebas, se detectó un gran fallo, y era que al intentar enviar un archivo al servidor, se cerraba la sesión. Esto era debido por el modo en que funcionaba el evento *onBeforeUnload*, ya que al pulsar el botón de enviar del formulario, se estaba realizando una petición al servidor, y por lo tanto recargando la página.

Se estuvo investigando mucho como poder discriminar la forma en que se salía de la página, en concreto, se quería detectar cuando se pulsaba el botón de cerrar se encontraron algunas soluciones “caseras”, puesto que no había ningún evento que indicara esta acción, pero dichas soluciones solo eran aplicables a Internet Explorer.

Este hecho hizo replantear la forma en que se abordaba el problema, de modo que al final se pensó que en vez de detectar cuando el usuario pulsaba el botón de cerrar del navegador, podría detectarse cuando el usuario pulsara el botón “enviar” del formulario de envío de archivos, y en el momento de activarse el evento *onBeforeUnload*, verificar si dicho evento se había producido por haber pulsado el botón del formulario, o por cualquier otra causa.

En este punto el problema era como indicar a una función *javascript* que se había pulsado un botón de un formulario que se encontraba en la parte de php, se probó pasando parámetros por POST además de otras pruebas varias, pero todas ellas sin obtener resultados satisfactorios. Tras agotar las pruebas se pensó en cambiar de estrategia, si no se conseguía indicar el evento de pulsar el botón del formulario, mediante el uso de variables a la función *javascript*, se descartarían las variables y se pasaría al uso de cookies.

Las cookies son pequeños archivos generados desde una página web que se almacenan en el ordenador y que pueden contener diversa información, como puede ser la dirección IP, nombres de usuario y contraseñas, etc.

Para el caso que se planteaba, no era necesario que en la cookie se almacenara información, puesto que la estrategia era que al pulsar el botón “enviar” del formulario, se creara una cookie, y al saltar el evento *onBeforeUnload* se verificara si existía dicha cookie, en el caso de que no existiera significaría que no se ha pulsado el botón “enviar” y por lo tanto se procedería al cierre de la sesión, en caso de que sí existiera significaría que se ha pulsado el botón “enviar” y por lo tanto no debía cerrarse la sesión y en ese momento se eliminaría la cookie. De este modo que la cookie se crea y

se destruye al instante de pulsar el botón “enviar” y su única función es evitar que se cierre la sesión.

Sabiendo que de este modo era posible controlar de forma correcta el cierre de sesión, sin perder funciones de la página web, se procedió a buscar la manera de gestionar dichas cookies. Se encontró una página web donde había un ejemplo del uso de cookies, además de las funciones necesarias para su gestión (Enlace 31), así que después de realizar los cambios necesarios sobre la página “laboratorio.php”, se consiguió finalmente el objetivo.

Aunque el uso de este método planteó una nueva problemática, y era que sería necesario tener activado el uso de cookies desde el navegador, en caso contrario el sitio web no funcionaría correctamente y se perdería el control de usuarios en el laboratorio. Por este motivo se decidió que era necesario controlar si el usuario tenía activada las cookies desde el momento en que llegaba al sitio web.

En una última instancia, se incluyó el código en la página “inicio.php”, para realizar dicha comprobación, al igual que en sus homólogas en los idiomas catalán e inglés. La estrategia era crear una cookie nada más acceder a la página e inmediatamente leer la cookie, si se podía leer la página eliminaba la cookie y permitía navegar al usuario, en caso contrario significaba que el navegador tenía deshabilitadas las cookies, de modo que realizaba una redirección a una página que muestra al usuario un mensaje, informándole que debe habilitar el uso de cookies para acceder al laboratorio remoto de robótica e inmediatamente envía la solicitud de cerrar el navegador, si el usuario acepta se cierra el navegador, en caso de cancelar se quedaría en una página donde tan solo se muestra el fondo de la web sin ninguna otra opción.

En la siguiente figura se muestra la apariencia de la página “inicio.php” (Ver Fig. 62)

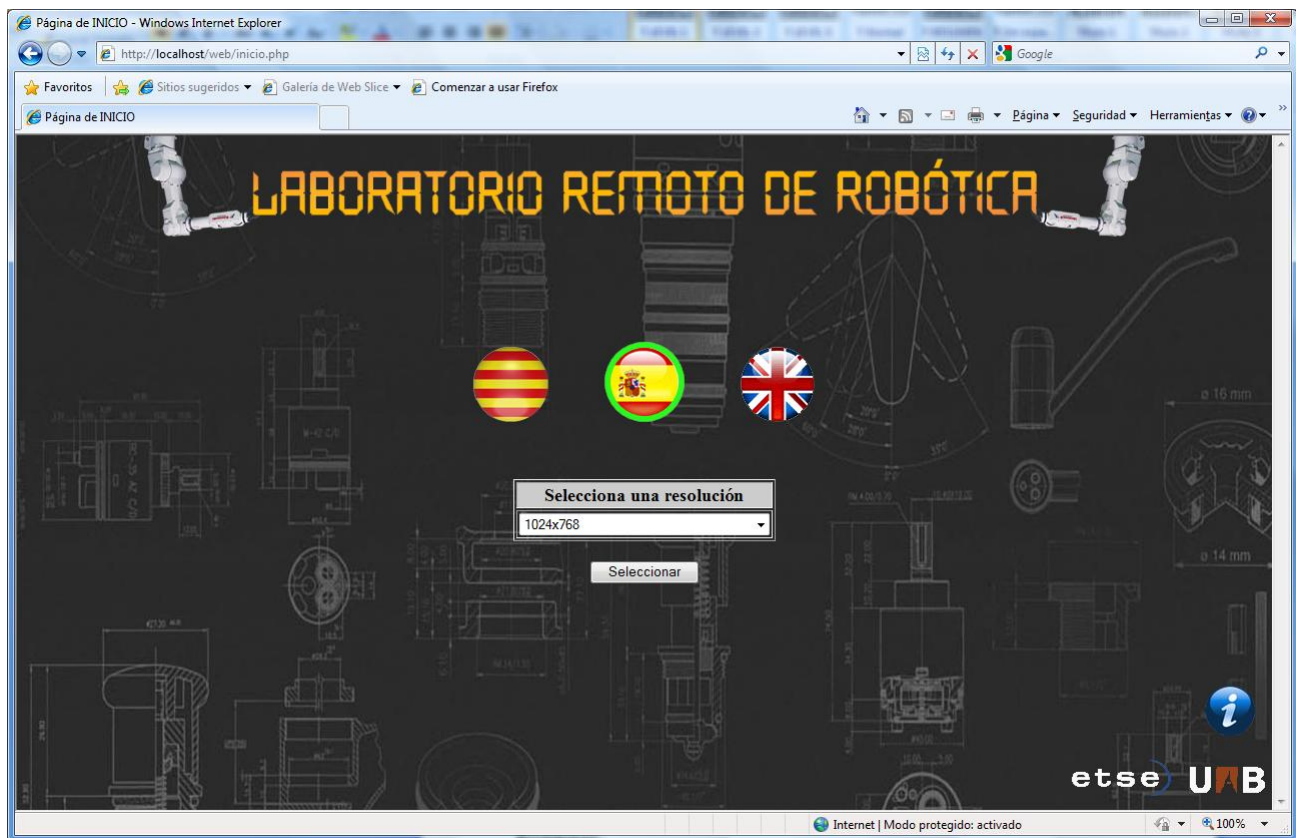


Fig. 62 – Página web inicio.php

### Últimos ajustes

Esta última etapa, se centra en la traducción de la web a los idiomas catalán e inglés, en el ajuste de la resolución y en la inclusión de la fecha y hora dinámica en la página web.

Para realizar los ajustes en cada página y adaptarla para su correcta visualización en función del tipo de resolución se hizo uso de un simulador de resoluciones (Enlace 32) que adaptaba una ventana del navegador a la resolución escogida.

Para realizar la traducción al inglés se utilizó en parte lo aprendido en la asignatura Introducción al Inglés Técnico. A continuación se muestran las páginas “inici.php” e “index.php”, las cuales fueron traducidas al catalán e inglés respectivamente.

Para incluir la fecha y hora de forma que se fuera refrescando cada segundo, se creó una página llamada “fecha.php”, la cual se incluía en el resto de páginas mediante la instrucción `<include>` y era recargada mediante AJAX.

La fecha varía en función del idioma, así que hubo que utilizar una instrucción que transforma la fecha incluyendo los días de la semana al idioma seleccionado, esta función se llama “*setlocale*”. Después de poder mostrar la fecha y hora en los tres idiomas, se observó que la página mostraba el siguiente aviso:



“Strict Standards: date() [function.date]: It is not safe to rely on the system's timezone settings. Please use the date.timezone setting, the TZ environment variable or the date\_default\_timezone\_set() function...”

Esto sucedía porque no estaba definida la zona horaria, así que se buscó información sobre la función que aparecía en el aviso, y se encontró una página web (Enlace 33) donde se mostraba como manejar dicha función, y de este modo se consiguió eliminar dicho aviso.

También se diseñó la página de error que lanzaría el servidor, si se hacía referencia a una página que no existía. “error.php”.

Como comentario para toda esta parte, hay que decir que el retoque y diseño gráfico de todos los elementos que aparecen se realizó gracias al uso de Adobe Photoshop CS, Paint (de Windows) y Microsoft Word 2007.

#### 4.8.6 Problemas y soluciones

##### Problemas:

- Firefox no soporta los cuadros con título o *fieldset*.
- Firefox no soporta que en un CSS se defina la posición de un *div* como “*relative*”.
- Actualizar partes de la página web sin recargar toda la página.
- Refrescar las imágenes captadas por la webcam de forma periódica.
- Se añaden caracteres invisibles en la variable del NIU usando el método POST.
- El ancho de banda es limitado y se envían imágenes constantemente.
- Se debe detectar cuando un usuario sale de forma anormal del laboratorio
- El evento *onBeforeUnload* anula la funcionalidad de subir archivos.
- Si el usuario desactiva el uso de cookies el control de acceso no funciona.
- Aparecía un aviso al hacer uso de las funciones de tiempo en php

##### Soluciones:

- Se sustituyó el uso del *fieldset* por tablas.
- Se define la posición como “*absolute*” y se coloca el *div* de la mejor manera posible.
- Utilización de AJAX.
- Utilización de AJAX.
- Se hace uso de las variables de sesión.
- Limitar el acceso hasta un número máximo de usuarios.
- Utilización del evento *onBeforeUnload* de *javascript*
- Utilización de cookies para detectar cuando se envía un archivo y no cerrar la sesión
- Detectar si el uso de cookies está activado desde la página de inicio
- Se hizo uso de la función *date\_default\_timezone\_set()* para eliminar dicho aviso.

## **4.9 MÓDULO 9 – APLICACIÓN DE TRANSMISIÓN DE VIDEO**

### **4.9.1 Descripción del módulo**

Este módulo consiste en una aplicación desarrollada de código abierto, que se encarga de realizar capturas desde la webcam y guardarlas posteriormente en un archivo en formato “jpg”. El intervalo más pequeño que puede definirse para la toma de imágenes es de 1 segundo, además todas las capturas realizadas se sobrescriben en el mismo archivo.

### **4.9.2 Entradas y salidas**

**Entradas:** - *Stream* de la webCam

**Salidas:** - Imagen en formato “jpg”

### **4.9.3 Tecnologías utilizadas**

Para el desarrollo de este módulo se ha hecho servir:

- Servidor de video: DCAM Server 8.2.6

### **4.9.4 Tiempo de desarrollo**

Para llevar a cabo el desarrollo de este módulo se han invertido un total de 15 horas.

### **4.9.5 Implementación**

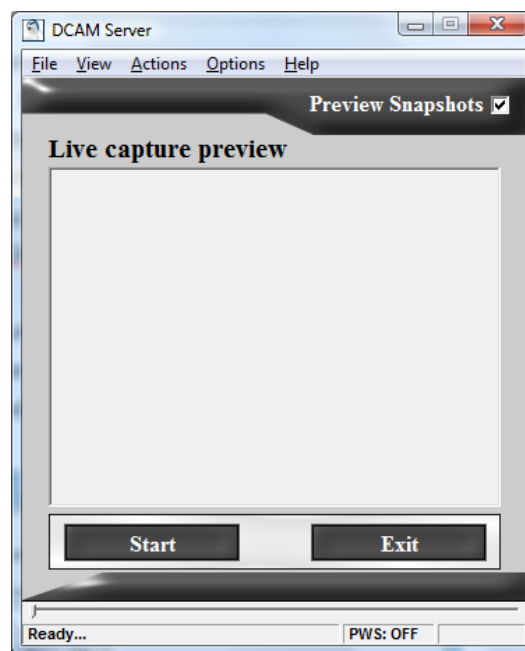
La implementación de esta parte, corresponde a la búsqueda y configuración del software apropiado para poder retransmitir las imágenes captadas a través de la webcam a través de la página web.

Las condiciones para la elección de este software eran:

- Software de código abierto
- Realización capturas de la webcam en formato “jpg”
- Guardar todas las capturas sobre el mismo archivo
- Poder ajustar el intervalo de capturas con resolución de milisegundos.

Buscando por Internet se vio que había bastantes programas que realizaban capturas desde la webcam, tanto privados como gratuitos. Aunque no se encontró ninguno que reuniera las características que se requerían, había unos cuantos que cumplía las tres primeras, y entre estos había uno que era el que más se acercaba a cumplir la cuarta característica, ya que podía definirse

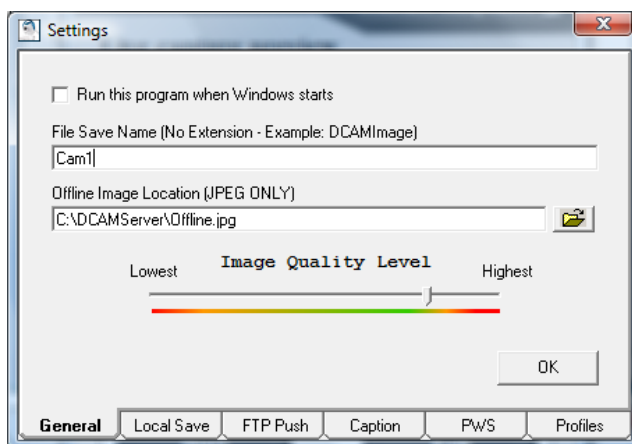
un intervalo de captura mínimo de 1 segundo, dicho software se llama DCam Server (Enlace 34) (Ver Fig. 63).



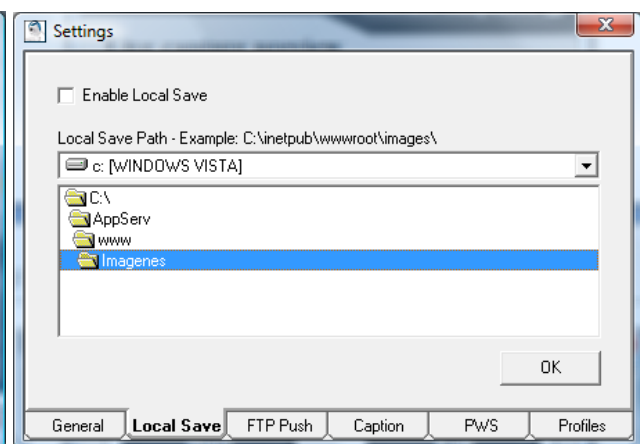
**Fig. 63 – Aplicación DCM Server**

Se instaló y configuró el DCam Server (Ver Fig. 64 y Fig. 65), para que guardara una captura cada segundo en la carpeta de imágenes de la página web, con el mismo nombre que se le había indicado a la página, de este modo las imágenes aparecerían en la web.

Después de tenerlo configurado correctamente, se realizó un pequeño programa para el robot y se ejecuto de forma local para ver como se veía a través de la webcam con esa frecuencia de imagen. El resultado rozaba lo aceptable, era posible ver las tareas que estaba desempeñando el robot, pero no se apreciaban algunos desplazamientos de un punto a otro.



**Fig. 64 – DCM Server (nombre captura)**



**Fig. 65 – DCM Server (directorio captura)**

Quería mejorarse esta característica, por ello se estuvo buscando la manera de desarrollar una aplicación que realizara la captura con los requisitos planteados. Se encontraron Apis en C#, Delphi, C++ y Java. Pero por desgracia no se disponía del tiempo suficiente para implementar dicha aplicación, puesto que aún faltaban módulos y documentación por terminar, e iniciar implementación de dicha aplicación significaría sobrepasar la fecha de entrega del proyecto. Así que la mejora de esta parte queda para una posible ampliación futura.

#### 4.9.6 Problemas y soluciones

##### **Problemas:**

- Podría mejorarse la captura de imágenes para enviar más imágenes y que la visualización sea un poco más fluida.

## 4.10 MÓDULO 10 – WEBCAM

### 4.10.1 Descripción del módulo

Este módulo representa al dispositivo físico que realiza la captura del video o imágenes del entorno que es enfocado por el objetivo de la cámara. Para este módulo no están los puntos de tiempo de desarrollo, implementación y problemas y soluciones, ya que el dispositivo ya está creado y tan solo hay que proceder a su instalación.

### 4.10.2 Entradas y salidas

**Entradas:** - Luz reflejada por el entorno

**Salidas:** - Imagen o video del entorno digitalizado.

### 4.10.3 Tecnologías utilizadas

Para el desarrollo de este módulo se ha hecho servir:

- Webcam: NGS SpinCam (Ver Fig. 66)



**Fig. 66 – Webcam NGS SpinCam**

<b>Marca:</b>	NGS
<b>Modelo:</b>	SpinCam
<b>Conexión:</b>	USB 2.0
<b>Sensor:</b>	CMOS 300 Kpx
<b>Resolución:</b>	- Vídeo: CIF 320x240 @30fps - Imagen: 640x480
<b>Foco:</b>	3 cm a infinito, F2.8 / f=5.3mm
<b>Otras características:</b>	- Normativa CE - Botón de captura de imagen fija - Micrófono incorporado (Solo Win xP SP2) - Gira 360° - Adaptable a pantallas TFT

## 4.11 MÓDULO 11 – GESTOR DE ARCHIVOS

### 4.11.1 Descripción del módulo

Este módulo consiste en una aplicación desarrollada en lenguaje C, que se encarga de comprobar si se han subido archivos al servidor, en cuyo caso busca el archivo más antiguo basándose en la fecha de creación del mismo, seguidamente analiza la sintaxis del archivo, si es incorrecta borra el archivo y en caso de ser correcta, realiza una copia del archivo en una carpeta del servidor llamada BDA (Base de archivos) añadiendo al nombre del archivo la fecha y la hora, a continuación copia el archivo en la carpeta del secuenciador de operaciones y se hace una llamada al programa secuenciador.exe, de este modo el robot ejecuta las operaciones indicadas en el programa, una vez finalizada la ejecución del programa el gestor de archivos borra el archivo de la carpeta del servidor, para que desaparezca de la página web, e inicia de nuevo el proceso.

Para consultar el código de este modulo ver archivo “GestorArchivos.cpp” incluido en el CD.

Funciones internas:

- **hash(char pos[]):** Esta función recibe una cadena de texto que representa una etiqueta y se encarga de realizar la búsqueda de la existencia dicha etiqueta dentro del vector PosET, si la encuentra retorna el índice donde se encuentra dicha etiqueta si no devuelve el valor -1.
- **verificador():** Esta función se encarga de analizar el archivo para comprobar que la sintaxis del mismo es correcta, si es correcta devuelve el valor 1, en caso contrario devuelve el valor 0.
- **mas\_antiguo(char pos[]):** Esta función realiza una búsqueda dentro del directorio de archivos del servidor, para encontrar cual es el archivo más antiguo. Si no hay archivos en el directorio devuelve el valor -1 en caso de haber devuelve 0 y guarda el nombre del archivo en el atributo “cFileName” de la variable global “FindFileData”.
- **ahora():** Esta función se encarga de guardar la fecha y hora actual cada vez que es llamada, en la variable global “Actual”.
- **llamada(char comando[]):** Esta función recibe un comando en formato *string*, y se encarga de realizar la ejecución de dicho creando un nuevo proceso.

#### 4.11.2 Entradas y salidas

**Entradas:** - Archivos de texto en formato "TXT"

**Salidas:** - Archivos de texto en formato "TXT"  
- Ejecución de la aplicación "secuenciador.exe"

#### 4.11.3 Tecnologías utilizadas

Para el desarrollo de este módulo se ha hecho servir:

- Compilador de lenguaje C "MinGW Developer Studio"

#### 4.11.4 Tiempo de desarrollo

Para llevar a cabo el desarrollo de este módulo se han invertido un total de 11 horas.

#### 4.11.5 Implementación

Para llevar a cabo el desarrollo de esta parte, en primer lugar se plantearon cuales serían los puntos a abordar para conseguir el correcto funcionamiento de la aplicación, de modo que tras analizar el problema se definieron los siguientes puntos:

- Buscar el archivo más antiguo
- Verificar la sintaxis del archivo
- Copiar el archivo a otro directorio renombrándolo
- Enviar el archivo al secuenciador
- Eliminar el archivo
- Pruebas y test

##### 4.11.5.1 **Buscar el archivo más antiguo**

Esta tarea consistía en buscar el archivo más antiguo, ya que de ese modo se tendría el archivo del usuario que realizó que hizo antes el envío y además el que está en primer lugar de la cola de usuarios de la página web, ya que la ordenación de dicha cola se realiza bajo el mismo criterio.

La idea inicial era buscar todos los archivos del directorio e ir guardando en un vector su nombre y fecha para luego realizar una búsqueda dentro del vector del archivo más antiguo, pero luego se pensó que sería más óptimo no tener que guardar la información de todos los archivos si al final tan solo nos interesa la información de uno de ellos. De este modo se pensó que lo ideal sería realizar la comprobación de cual era más antiguo a medida que se iban encontrando los archivos.

Así pues, la primera tarea era poder acceder a un directorio y listar sus archivos, omitiendo los dos archivos “.” y “..” como se hizo en la parte web. Después de una búsqueda por Internet se halló un ejemplo donde se listaban todos los archivos de un directorio y se imprimía por pantalla (Enlace 35). Este ejemplo fue la base para realizar la búsqueda de los archivos, luego se modificó para que mientras buscaba, fuese comparando la fecha de creación mediante la función “*CompareFileTime()*” a la cual se le pasaba el atributo de fecha de creación de ambos archivos retornándonos 1, si el primera era más antiguo que el segundo.

Para saber cómo acceder a los atributos de los archivos que se iban encontrando, hubo que consultar las características y estructura del tipo de dato WIN32\_FIND\_DATA (Enlace 36).

Teniendo claras las dos partes anteriores, la estrategia fue declarar una variable global del tipo WIN32\_FIND\_DATA, llamada “*FindFileDataAnt*”, para guardar los atributos del archivo más antiguo, entonces se iniciaba la búsqueda de archivos dentro del directorio, cuando encontraba el primero se guardaban sus atributos en la variable global como si fuera el más antiguo, a partir de aquí cada vez que se encontraba un archivo se comparaba con el más antiguo, si el nuevo resultaba ser más antiguo que el que se tenía, se sustituía este último y el nuevo pasaba a ser el más antiguo. De este modo al finalizar la búsqueda en el directorio, la variable global “*FindFileDataAnt*” contenía los atributos del archivo más antiguo.

Como curiosidad en esta parte, se averiguó que el formato en que se guarda la fecha de los archivos depende del formato de la partición del disco, esto quiere decir que si nuestro disco tiene un sistema de archivos basado en FAT, la fecha se guarda en base a la hora local, y si el sistema de archivos es NTFS la fecha se guarda en formato UTC (Tiempo Universal Coordinado “en español”), además también cambia la resolución de los atributos de fecha, por ejemplo, el atributo “último acceso al archivo” en FAT tiene resolución de 1 día mientras que en NTFS la resolución es de 1 hora. Esta curiosidad no afecta a la función “*más\_antiguo()*”, ya que en cualquier caso los archivos con los que trabaja están bajo el mismo sistema.

El código de esta parte puede ser consultado en: (Ver CD: archivo “gestorArchivos.cpp”, función *mas\_antiguo()* ).

#### 4.11.5.2 Verificar la sintaxis del archivo

Esta parte fue rápida de implementar, puesto que se aprovechó el código de la aplicación “Verificador.exe” encapsulándolo en una función llamada “*verificador()*”. Del resultado de la función “*mas\_antiguo()*” ya teníamos el nombre del archivo más antiguo, así que al llamar a la función “*verificador()*”, tan solo tenía que acceder a la variable global y a su atributo “nombre de archivo”, “*FindFileDataAnt.cFileName*”, para de este modo abrir el archivo y proceder a su verificación. Código: (Ver CD: archivo “gestorArchivos.cpp”, función *verificador()* ).



#### 4.11.5.3 Copiar el archivo a otro directorio renombrándolo

La idea de este apartado era la de conseguir poder realizar copias de un archivo, con el objetivo de poder copiar el archivo que se había enviado al servidor, a la carpeta del “secuenciador.exe” para ser ejecutado, y además realizar una copia del mismo archivo en la carpeta BDA, añadiendo al nombre del archivo la fecha y hora actual, para así almacenarlo.

Para empezar a desarrollar este apartado, se consultó en las páginas de MSDN (*del inglés Microsoft Developer Network*) para ver si había alguna función para la copia de archivos, y efectivamente la función existía y se llamaba “CopyFile()”, además dicha función permitía renombrar el archivo al mismo tiempo que era copiado, funcionalidad muy útil para el problema que se había planteado.

Las acciones de copia y renombrado de archivo se llevan a cabo desde el programa principal del gestor de archivos, pero además se definió una nueva función llamada “actual()” la cual guardaba en una variable global llamada del mismo modo, una cadena de texto que contenía la fecha y hora actual, el código de dicha función se obtuvo de una página que se encontró en Internet (Enlace 37).

#### 4.11.5.4 Enviar el archivo al secuenciador

En esta parte la función que se quería conseguir era poder ejecutar el “secuenciador.exe” desde la aplicación de gestión de archivos.

Para ello se consultó el modo de hacerlo y finalmente se encontró. Había que hacer uso de una función llamada “createProcess” (Enlace 38). Gracias al ejemplo que se encontró en la página de MSDN, se modificó y creó la función “llamada()” la cual recibía un cadena de texto con la ruta de la aplicación a ejecutar.

De este modo cuando se llamaba a la función “llamada” desde el código principal, se creaba un nuevo proceso y se ejecutaba la aplicación que se hubiera indicado, quedando el programa principal a la espera de la finalización de la aplicación.

#### 4.11.5.5 Eliminar el archivo

En esta parte se quería conseguir la eliminación del archivo de la carpeta “archivos” del servidor.

Se buscó nuevamente dentro de la página web MSDN, para ver si existía alguna función que realizara esta operación, y se encontró la función “DeleteFile()” la cual eliminaba el archivo que se le indicaba.

Se aprovechó que en la variable global (FindFileDataAnt.cFileName) teníamos el nombre del archivo que se estaba tratando, para construir la ruta al archivo mediante el uso de la función

“*strcat()*” para concatenar cadenas de texto, guardando el resultado en una variable llamada *archivo*, la cual enviada a la función “*DeleteFile*” indicaba la ruta al archivo que debía eliminarse.

Así que teniendo listo este apartado, estaban concluidos todos los puntos para el correcto y completo funcionamiento del gestor de archivos.

#### 4.11.5.6 Pruebas y test

Durante esta fase se detectaron varios problemas que provocaban que la aplicación quedara bloqueada.

El primero fue al intentar eliminar archivos, después de varios análisis se detectó que este problema era porque la carpeta que contenía los archivos estaba marcada como solo lectura junto con todo su contenido, desmarcada esta opción se solucionó el problema, incluso si se enviaba un archivo ya marcado como solo lectura desde la página web, al guardarse en el directorio se desmarcaba esta propiedad.

El segundo fue cuando la aplicación eliminaba todos los archivos del directorio, al estar continuamente en ejecución, la aplicación llamaba a la función “*verificador()*”, que al no encontrar archivos que procesar lanzaba una excepción y bloqueaba la aplicación. Así que se incluyó una condición en el programa principal, mediante la cual si no había archivos en el directorio, no llegaba a realizarse la llamada a la función “*verificador()*”.

El tercero y último fue tras corregir el segundo problema. Se pensaba que ya estaba solucionado el segundo problema, pero se detectó que aunque se había añadido la condición para no llamar al *verificador* si no existían archivos que procesar en el directorio, había un breve periodo de tiempo en el cual se había dado la orden de eliminar el último archivo, pero a la aplicación *mas\_antiguo()* le daba tiempo de ejecutarse de nuevo antes de la eliminación total del archivo. Este hecho provocaba que cuando se verificaba la condición de existencia de archivos en el directorio esta fuera cierta, y por lo tanto se llamaba a la función “*verificador()*”, pero al entrar dentro de esta función el archivo ya había sido eliminado, y de nuevo provocaba el bloqueo de la aplicación. Así que se pensó en añadir un retardo de un segundo antes de llamar a la función “*mas\_antiguo()*”, para dar tiempo a actualizar el estado del directorio y llamar a la función “*verificador()*” cuando realmente existan archivos que no estén en proceso de eliminación.

#### 4.11.6 Problemas y soluciones

##### Problemas:

- No se podía asignar el atributo "*cFileName*" o nombre de archivo directamente.
- La aplicación se quedaba bloqueada al intentar eliminar los archivos.
- La aplicación se quedaba bloqueada si no había archivos en el directorio.
- La aplicación se quedaba bloqueada al eliminar todos los archivos, porque entraba dentro de la función "mas\_antiguo()".

##### Soluciones:

- Se hizo uso de la función *strcpy()*
- El directorio que contenía los archivos estaba marcado como "solo lectura", así que se desmarcó esta opción.
- Se incluyó una restricción para no llamar a la función "verificador()" si no había archivos.
- Se incluyó un retardo de 1 segundo, para que diera tiempo a refrescar el estado del directorio.

## 4.12 MÓDULO 12 – PC

### 4.12.1 Descripción del módulo

Este módulo representa al ordenador personal de los usuarios remotos. Su función es la de permitir la realización del programa de tareas PPO, poder acceder a la página web de laboratorio para de este modo realizar el envío del archivo y posteriormente la visualización de los movimientos del robot.

### 4.12.2 Entradas y salidas

**Entradas:** - Información procedente del servidor.

**Salidas:** - Archivo TXT con el programa para PPO.  
- Información visual por la pantalla.

## **5. DOCUMENTACIÓN**

Tras haber finalizado la parte de implementación se procedió a la redacción de la documentación correspondiente que fue la siguiente:

Manual de usuario alumno (Anexo 7): Es el manual del funcionamiento básico del laboratorio, dirigido a los usuarios comunes del laboratorio.

Manual de usuario administrador (Anexo 8): Es el manual del funcionamiento avanzado del laboratorio, dirigido al usuario administrador del laboratorio, es decir, el profesor.

Prácticas de Robótica y Automatización Industrial 09-10 (Anexo 9): Es el guión de prácticas de la asignatura mencionada para el curso 2009 - 2010, el cual deberán seguir los alumnos que estén matriculados en la misma.

En este apartado también encontraríamos la propia redacción de esta memoria, aunque en este caso el desarrollo no ha sido tan localizado como el de la documentación anterior, sino que se ha llevado a cabo durante todas las fases del proyecto.

## 6. AMPLIACIONES FUTURAS

Una de las premisas de la realización de este proyecto es que fuese ampliable. Teniendo en cuenta esto se exponen una serie de líneas de proyecto que podrían iniciarse en el futuro:

- Realización de una aplicación para la realización de programas de seguimiento de trayectorias.
- Control remoto del robot a través de un dispositivo aptico.
- Creación de un sistema multi-robot, para la colaboración en tareas de PPO.
- Dotar al robot de inteligencia artificial para jugar a juegos de mesa contra humanos.
- Hacer uso de la plataforma hardware junto con el software de control para sustituir a los robots inoperativos de la célula de fabricación flexible del laboratorio de la ETSE.

En cuanto a las mejoras que podrían realizarse en este propio proyecto encontraríamos lo siguiente:

- Realizar la validación W3C a la página web del laboratorio.
- Reducir la resolución del intervalo de tiempo de captura de la aplicación servidor de video hasta los milisegundos.

## 7. CONCLUSIONES

Actualmente el proyecto se encuentra totalmente operativo y será posible su utilización para el fin que se había previsto que es: utilizarlo como laboratorio de robótica para la asignatura robótica y automatización industrial a partir del curso 09-10, y servir como base para iniciar nuevas líneas de proyecto en el ámbito de la robótica industrial, además de poder utilizarse para la colaboración con otros centros en referencia a poder compartir recursos docentes, gracias a la funcionalidad remota de la plataforma.

En cuanto a la planificación y costes previstos, estos han diferido de los reales y a continuación se detallaran las diferencias:

Si comparamos el diagrama de Gantt inicial (Ver Fig. 11) con el diagrama de Gantt final (Ver Fig. 67), la primera discrepancia que se observa es la de la morfología del diagrama, el primer diagrama nos muestra una estructura con paralelismos entre tareas, mientras que en el segundo la tendencia es a realizar una cascada. Esto es debido a que durante la fase del estudio de viabilidad, se definieron una serie de figuras (roles), los cuales representaban a personas distintas y por lo tanto podían trabajar en paralelo, sin embargo, la realidad ha sido que todos esos roles han sido asumidos por una misma persona, lo que ha implicado que la distribución del trabajo se realizase de una forma secuencial, aún cuando las propias tareas no ofrecían tal restricción.

Otra diferencia que puede observarse es la de la fecha de fin del proyecto, que en la realidad ha tendido a dilatarse. Esto ha sido provocado por dos factores: el primero, como se comentaba anteriormente, por la realización secuencial de las tareas, y el segundo, por el tiempo que requirió la burocracia necesaria para la adquisición de los componentes del robot, agravado posteriormente por un retraso en la entrega de éstos, provocado por un error administrativo en el pago del pedido que se había solicitado, generando un retraso de dos meses.

En cuestión de horas previstas la diferencia no ha sido significativa, teniendo un balance de 5 horas extra (horas previstas 350, horas realizadas 355). En cambio en el coste previsto sí que ha habido mayor diferencia en exceso, en concreto de 1.935,50 € (coste previsto 3.978€, coste real 5.913,50 €). El incremento del coste del proyecto no ha sido proporcional al incremento por las 5 horas extra en la duración del proyecto, la explicación de este suceso es que, puesto que ha habido tareas que han requerido menor tiempo del previsto y otras que han requerido más, en el computo global el exceso de unas era compensado por el déficit de otras, pero las tareas que se han incrementado en tiempo, han sido las de los recursos que tenían mayor coste, sin embargo las que se han reducido eran las de los recursos con un coste menor, y de ahí este incremento en el coste real aun habiendo reducido las 5 horas.

Estos cambios en el coste del proyecto, hacen que al final para amortizarlo vendiendo el laboratorio en versión simple, sean necesarias 17 ventas, en el caso de la versión completa “laboratorio + PC” sean necesarias 16 ventas.

En cuanto a cambios realizados en el proyecto respecto a la planificación inicial, el único ha sido el no contar con un dominio para la página web del proyecto, ya que el servicio informático de la UAB no lo creyó necesario a priori, y por lo tanto el acceso a la web del laboratorio se realiza mediante la IP del servidor web, y para accesos externos es necesaria la configuración de una VPN.

Por último, hay que decir que se han cumplido todos los requerimientos planteados inicialmente, habiendo alcanzado satisfactoriamente todos los objetivos previstos.



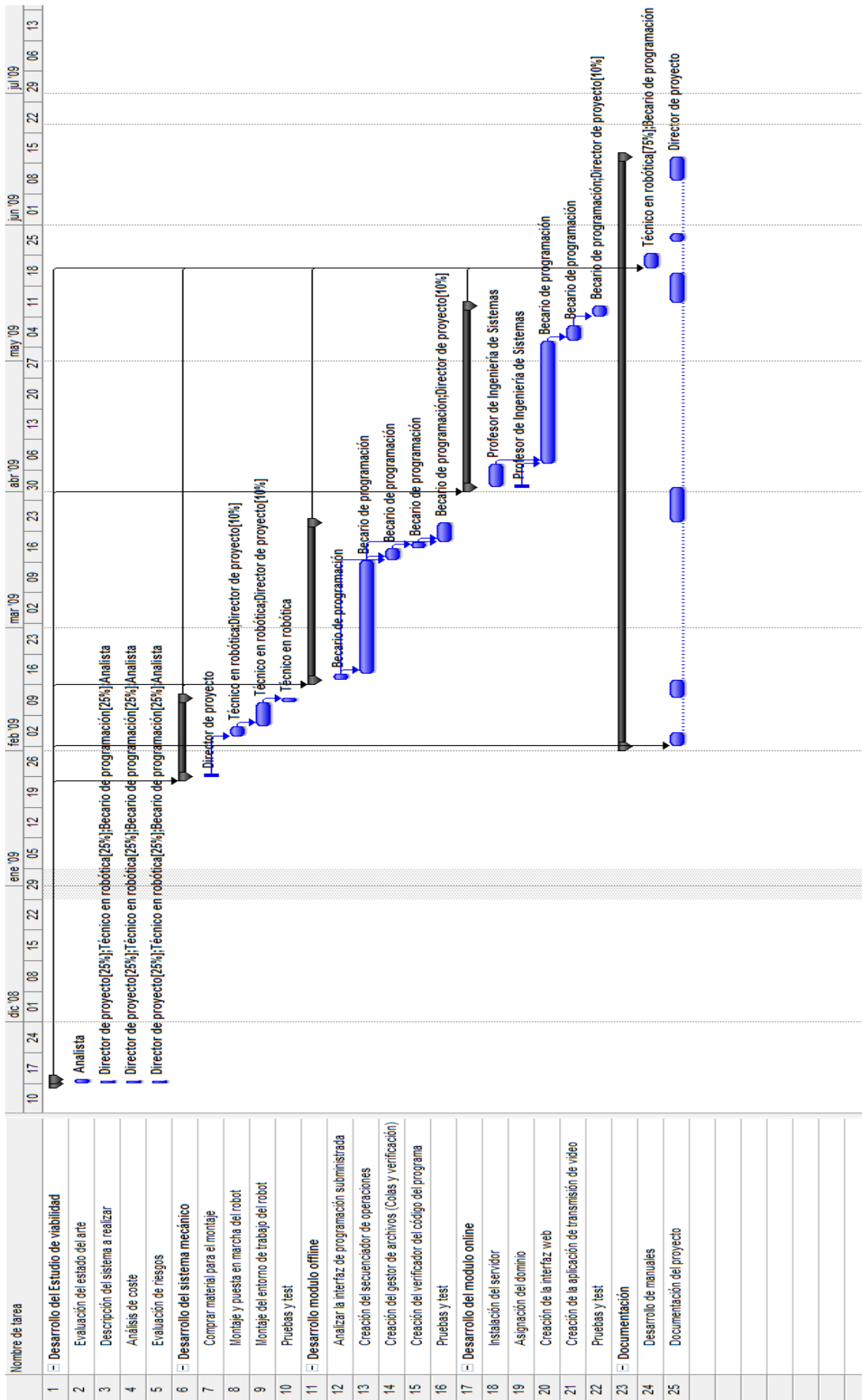


Fig. 67 – Diagrama de Gantt Final

## 8. ENLACES WEB

- **Enlace 1. Morfología de Robots Industriales:**

Última visita: 07/01/2009

Enlace: [http://cfievalladolid2.net/tecno/ctrl\\_rob/robotica/sistema/morfologia.htm](http://cfievalladolid2.net/tecno/ctrl_rob/robotica/sistema/morfologia.htm)

- **Enlace 2. Actuadores:**

Última visita: 07/01/2009

Enlace: <http://www.aie.cl/files/file/comites/ca/abc/actuadores.pdf>

- **Enlace 3. Estándar RS232:**

Última visita: 26/02/2009

Enlace: [http://catarina.udlap.mx/u\\_dl\\_a/tales/documentos/lep/bazan\\_h\\_ja/apendiceA.pdf](http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/bazan_h_ja/apendiceA.pdf)

- **Enlace 4. Proyecto robótica móvil universidad de Granada:**

Enlace: [http://prensa.ugr.es/prensa/innovacion\\_docente/verNota/prensa.php?nota=796](http://prensa.ugr.es/prensa/innovacion_docente/verNota/prensa.php?nota=796)

Última visita: 18/11/2008

- **Enlace 5. Proyecto laboratorio remoto Universidad Nacional de Comahue :**

Enlace: <https://dc.exa.unrc.edu.ar/wicc/papers/InformaticaEducativa/106.PDF>

Última visita: 18/11/2008

- **Enlace 6. Características servomotores HS475HB**

Última Visita: 01-03-2009

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

- **Enlace 7. Características servomotores HS-85BB**

Última Visita: 01-03-2009

Enlace: [http://www.hobbyhorse.com/hitec\\_hs85bb.shtml](http://www.hobbyhorse.com/hitec_hs85bb.shtml)

- **Enlace 8. Características servomotores HS-81**

Última Visita: 01-03-2009

Enlace: [http://www.servocity.com/html/hs-81\\_micro.html](http://www.servocity.com/html/hs-81_micro.html)

- **Enlace 9. Web Lynxmotion:**

Última visita: 05-02-2009

Enlace: <http://www.lynxmotion.com/ViewPage.aspx?ContentCode=assem01&CategoryID=19>

- **Enlace 10. Circuito SSC-32:**

Última visita: 23/01/1009

Enlace: <http://www.lynxmotion.com/Product.aspx?productID=395>

- **Enlace 11. Manual SSC-32:**

Última Visita: 08-02-2009

Enlace: <http://www.lynxmotion.com/images/html/build136.htm#srvmov>

- **Enlace 12. Foro Lynxmotion:**

Última Visita: 06-02-2009

Enlace: <http://www.lynxmotion.net/viewtopic.php?t=4057&sid=fa487100937134f9a611b3d12f6acdd6>

- **Enlace 13. Foro de ARDE:**

Última visita: 08-02-2009

Enlace: <http://www.webdearde.com/modules.php?name=Forums&file=viewtopic&p=26996#26996>

- **Enlace 14. Prácticas RS232:**

Última visita: 07-02-2009

Enlace: <http://www.iit.upcomillas.es/~carlosrg/Docencia/LFCI/LFCI.html>

- **Enlace 15. MinGW Developer:**

Última visita: 10-02-2009

Enlace: <http://www.programatium.com/en/programs-windows/programas.php?id=1241>

- **Enlace 16. Cadenas de caracteres en C:**

Última visita: 03-03-2009

Enlace: <http://www.desarrollador-web.com/2009/02/programacion-en-c-strings-cadena-de-caracteres/>

- **Enlace 17. Función Sleep() para retardos:**

Última visita: 03-03-2009

Enlace: <http://taichi.obolog.com/delay-sleep-c-14299>

- **Enlace 18. Vector de cadenas de caracteres:**

Última visita: 10-03-2009

Enlace: <http://www.kirupa.com/forum/archive/index.php/t-233004.html>

- **Enlace 19. Software Appserv 2.6**

Última visita: 31-03-2009

Enlace: <http://appserv.softonic.com/>

- **Enlace 20. Configuración Apache con .htaccess**

Última visita: 01-04-2009

Enlace: <http://www.tru3n0.com/index.php/2007/11/07/proteger-directorios-con-contrasena-en-apache-htaccess/>

- **Enlace 21. Editor web WYSING – Nvu 1.0**

Última visita: 01-04-2009

Enlace: <http://es.kioskea.net/telecharger/telechargement-185-nvu>

- **Enlace 22. Editor web Aptana Studio 1.2.5**

Última visita: 01-04-2009

Enlace: <http://www.updatestar.com/es/detail/aptana-studio/description>

- **Enlace 23. Tutorial online de HTML**

Última visita: 01-04-2009

Enlace: <http://www.entraenlared.com/html/>

- **Enlace 24. Editor multilenguaje Notepad++ 5.3.1**

Última visita: 01-04-2009

Enlace: <http://notepad.softonic.com/>

- **Enlace 25. Foro de PHP – Subir archives via http**

Última visita: 01-04-2009

Enlace: <http://cl2.php.net/manual/es/features.file-upload.php>

- **Enlace 26. Ordenación de archivos vía PHP**

Última visita: 03-04-2009

Enlace: <http://cl2.php.net/manual/es/features.file-upload.php>

- **Enlace 27. Ejemplos de AJAX**

Última visita: 14-04-2009

Enlace: <http://www.miguelmanchego.com/2009/actualizar-un-elemento-sin-recargar-con-jquery/>

- **Enlace 28. Refresco de imagen con AJAX**

Última visita: 07-04-2009

Enlace: <http://www.forosdelweb.com/f13/refrescar-imagen-generada-por-php-672766/>

- **Enlace 29. Función fopen en PHP**

Última visita: 07-04-2009

Enlace: <http://es.php.net/manual/es/function.fopen.php>

- **Enlace 30. Funciones hash**

Última visita: 07-04-2009

Enlace: <http://gaussianos.com/algoritmos-hash-ii-atacando-md5-y-sha-1/>

- **Enlace 31. Tratamiento de cookies**

Última visita: 23-04-2009

Enlace: <http://gaussianos.com/algoritmos-hash-ii-atacando-md5-y-sha-1/>

- **Enlace 32. Simulador de resoluciones**

Última visita: 25-04-2009

Enlace: [http://www.mclibre.org/consultar/php/lecciones/php\\_fecha\\_hora.html](http://www.mclibre.org/consultar/php/lecciones/php_fecha_hora.html)

- **Enlace 33. Funciones de tiempo en PHP**

Última visita: 24-04-2009

Enlace: [http://www.mclibre.org/consultar/php/lecciones/php\\_fecha\\_hora.html](http://www.mclibre.org/consultar/php/lecciones/php_fecha_hora.html)

- **Enlace 34. Servidor de video DCam Server**

Última visita: 08-05-2009

Enlace: <http://sourceforge.net/projects/dcamserver/>

- **Enlace 35. Listar archivos de un directorio en C**

Última visita: 05-05-2009

Enlace: [http://foro.elhacker.net/programacion\\_cc/listar\\_directorio\\_en\\_windows-t231742.0.html;msg1105689](http://foro.elhacker.net/programacion_cc/listar_directorio_en_windows-t231742.0.html;msg1105689)

- **Enlace 36. Estructura del tipo de dato WIN32\_FIND\_DATA**

Última visita: 05-05-2009

Enlace: [http://msdn.microsoft.com/en-us/library/aa365740\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/aa365740(VS.85).aspx)

- **Enlace 37. Fecha y hora actual en C**

Última visita: 05-05-2009

Enlace: <http://www.holamundo.es/lenguaje/c/articulos/fecha-hora-c.html>

- **Enlace 38. Llamada a aplicación externa en C**

Última visita: 05-05-2009

Enlace: <http://msdn.microsoft.com/en-us/library/ms682425.aspx>

## 9. GLOSARIO

**Actuador:** Es un dispositivo inherentemente mecánico cuya función es proporcionar fuerza para mover o “actuar” otro dispositivo mecánico. La fuerza que provoca el actuador proviene de tres fuentes posibles: Presión neumática, presión hidráulica, y fuerza motriz eléctrica (motor eléctrico o solenoide). Dependiendo del origen de la fuerza el actuador se denomina “neumático”, “hidráulico” o “eléctrico”.

**Asíncrono:** hace referencia al suceso que no tiene lugar en total correspondencia temporal con otro suceso.

**Brazo del robot:** Una de las partes del manipulador. Soportado en la base de éste, sostiene y maneja la muñeca (donde va instalado el útil de toma de objetos).

**Cartesianas, coordenadas:** Método para definir la posición de un punto por medio de su distancia perpendicular a dos o más líneas de referencia.

En geometría plana, dos líneas rectas, llamadas eje x y eje y, forman la base de un sistema de coordenadas Cartesianas en dos dimensiones. Por lo general, el eje x es horizontal y el eje y es perpendicular a él. Al punto de intersección de los dos ejes se le llama origen (O). Cualquier punto en este plano se puede identificar por un par ordenado de números que representan las distancias a los dos ejes.

**Cinemático:** En robótica se utiliza este término para referirse a los accionamientos de un manipulador que suponen una unión física directa entre los mandos del operador y el elemento terminal.

**Control remoto, manipulador de:** Aquél en que cada grado de libertad está actuado por un dispositivo independiente, con lo que puede no estar unido cinemáticamente al actuador del operador.

**Coordenadas:** Sistema de ejes para el posicionamiento de un punto en el plano o en el espacio. Pueden ser: a) Angulares. Si la referencia de un punto se hace mediante la definición de ángulos a partir de los ejes (origen de los ángulos). b) Polares. Se establece un punto mediante la indicación de un ángulo y un valor escalar (numérico). c) Rectangulares. Cuando los puntos están definidos por varios números (dos o tres).

**Eje:** Cada una de las líneas según las cuales se puede mover el robot o una parte de él (algún elemento de su estructura). Pueden ser ejes o líneas de desplazamiento longitudinal sobre sí mismo (articulación prismática) o ejes de giro (rotación). Cada eje define un “grado de libertad” del robot.

**Elemento:** Cada uno de los componentes de la estructura de un manipulador. Pueden ser elemento maestro, esclavo, de unión, terminal, etc.

**Enrutador:** (del inglés *router*) ruteador o encaminador es un dispositivo de hardware para interconexión de red de ordenadores que opera en la capa tres (nivel de red). Este dispositivo permite asegurar el enrutamiento de paquetes entre redes o determinar la ruta que debe tomar el paquete de datos

**Giro:** Movimiento básico de un manipulador. (Ver Eje.)

**Grado de libertad:** Cada uno de los movimientos básicos que definen la movilidad de un determinado robot. Puede indicar un movimiento longitudinal o de rotación. (Ver Eje.)

**Hidráulico:** Es un manipulador cuya energía de movimiento viene proporcionada por un fluido que presiona émbolos. Se consigue una gran potencia en la operación del robot, aunque se pierda precisión.

**HTML:** es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

**Informática:** Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de computadoras.

**Internet:** es un conjunto descentralizado de redes de comunicación interconectadas, que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial.

**Javascript:** es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

**Manipulador:** En general, cualquier dispositivo mecánico capaz de reproducir los movimientos humanos para la manipulación de objetos. En particular, suele referirse a los elementos mecánicos de un robot que producen su adecuado posicionamiento y operación.

**Máquina:** Artificio o conjunto de aparatos combinados para recibir cierta forma de energía, transformarla y restituirla en otra más adecuada o para producir un efecto determinado.

**Muñeca:** Dispositivo donde se articula el elemento terminal (garfio, pinza, etc.) de un manipulador. Es un elemento básico para la definición de la flexibilidad y precisión del manipulador. Las posiciones del elemento terminal vienen dadas por los grados de libertad de la muñeca.



**Neumático:** Es un manipulador cuya energía de movimiento viene proporcionada por un sistema de aire comprimido (conductos que lo contienen, émbolos de empuje, sistema compresor, etc.).

**Paso a paso, motor:** Motor eléctrico que gira un número exacto de grados al recibir una adecuada secuencia de comandos de control. Son motores sumamente precisos.

**PHP:** es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (*server-side scripting*).

**Pinza:** Una de las configuraciones típicas del elemento terminal de un manipulador. Es un elemento de precisión y potencia medias. Se articula con el resto de la estructura a través de la muñeca.

**Polares, coordenadas:** Una forma de definir la posición de un punto en términos de la distancia  $r$  de un punto fijo llamado origen al punto y el ángulo  $\theta$  entre la línea del origen al punto y una línea fija llamada eje. Las coordenadas del punto se expresan como  $(r, \theta)$ .

**Puerto serie:** es una interfaz de comunicaciones de datos digitales, frecuentemente utilizada por computadoras y periféricos, en donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits simultáneamente.

**Robot:** Manipulador mecánico, reprogramable y de uso general.

**Robot Industrial:** En particular, los robots utilizados en la fabricación o procesamiento de objetos (los más numerosos) se suelen llamar robots industriales.

**Rotación:** Movimiento básico en un manipulador.

**Servidor:** En informática, un servidor es una computadora que, formando parte de una red, provee servicios a otras computadoras denominadas clientes.

**URL:** Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

**Web:** documento o fuente de información, generalmente en formato HTML y que puede contener hiperenlaces a otras páginas web. Dicha página web, podrá ser accesible desde un dispositivo físico, una intranet, o Internet.

**10. INDICE DE ANEXOS**

- Anexo 1.** Manual para el montaje de la base del robot
- Anexo 2.** Manual para el montaje de las barras 1 y 2 del robot
- Anexo 3.** Manual para el montaje de la pinza del robot
- Anexo 4.** Manual del circuito SSC-32
- Anexo 5.** Práctica 1 del puerto serie RS232 por José Antonio Rodríguez Mondéjar
- Anexo 6.** Práctica 3 del puerto serie RS232 por José Antonio Rodríguez Mondéjar
- Anexo 7.** Manual usuario alumno.
- Anexo 8.** Manual usuario administrador.
- Anexo 9.** Prácticas de Robótica y Automatización Industrial 09-10.